Quick answers to common problems

Axure RP Prototyping Cookbook

Over 70 practical recipes to take your wireframing and prototyping skills to the next level using Axure

John Henry Krahenbuhl

[PACKT] PUBLISHING

Axure RP Prototyping Cookbook

Over 70 practical recipes to take your wireframing and prototyping skills to the next level using Axure

John Henry Krahenbuhl

[PACKT] PUBLISHING

BIRMINGHAM - MUMBAI

Axure RP Prototyping Cookbook

First published: January 2014

Production Reference: 1170114

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84969-798-9

www.packtpub.com

Cover Image by Suresh Mogre (suresh.mogre.99@gmail.com)

Credits

Author
John Henry Krahenbuhl

Reviewers
Anant Athale
Ildikó Balla
Melissa Krahenbuhl
Jessamyn Smallenburg
Wahne Lducia Tubman

Acquisition Editors
Anthony Albuquerque
Harsha Bharwani
Usha Iyer

Lead Technical Editor
Susmita Panda

Technical Editors
Shiny Poojary
Tarunveer Shetty
Sonali Vernekar

Copy Editors
Sarang Chari
Brandt D'Mello
Mradula Hegde

Project Coordinator
Leena Purkait

Proofreader
Mario Cecere

Indexer
Tejal Soni

Production Coordinator
Alwin Roy

Cover Work
Alwin Roy

About the Author

John Henry Krahenbuhl has over 20 years of experience in architecting practical, cost-effective, innovative solutions. Being a creative thinker and having an entrepreneurial spirit has enabled him to be the lead or co-inventor on numerous utility patent applications. He is a multifaceted, collaborative management professional, highly skilled at managing products through the entire life cycle, from design to obsolescence, including specification and use case definitions, schematic and PCB layouts, production software implementation, and hardware implementation. He's a passionate, resourceful leader who demands and delivers excellence in design and user experience.

The love and encouragement of my family, made this work possible. I would like to thank my wonderful wife, Melissa, for her limitless love, motivation, faith, and support. I would also like to thank my children Matthew, Jason, Lauryn, and Henry. Seeing the world through your eyes inspires me. I would like to thank my Aunt Teddi Robinson for her determination and drive, and for always inspiring us to follow our dreams.

And lastly, I would like to thank Nikola Tesla for his vision and contribution to modern society. May your accomplishments one day be fully appreciated by the masses.

About the Reviewers

Anant Athale has a passion for software engineering. His areas of practice include web and mobile user interface architecture, enterprise Java architecture, and big data architecture. He has two issued patents and has a Master's degree in Engineering from Arizona State University.

Ildikó Balla is a user experience consultant / interaction designer currently living in Sydney, Australia. Her experience of over six years in the field includes working on mobile, web, and desktop applications ranging from simple sites to complex back office solutions and e-commerce platforms. Specializing in interaction design and medium-fidelity prototyping, she has used Axure as her tool of choice for the past five years. She is a proficient-to-expert user of Axure 6.0 and all versions above that. As a part of her dedication and desire to become an expert user of Axure, she successfully completed several training courses in Axure prototyping for web and mobile led by Ritch Macefield—Europe's only English language trainer at the time (`www.axperts.com`).

She is currently working as UX consultant for reInteractive (`www.reinteractive.net`), Australia's largest Ruby on Rails-focussed development company producing custom web-based software for a diverse range of clients. More specifically, in this role she is in charge of requirements analysis, information architecture, interaction design, and user testing for complex web applications and business intelligence solutions. Some of her short articles about UX, interaction design, and prototyping can be found on the reInteractive company blog. Before moving to Australia, she was a long-term employee of REEA (`www.reea.net`), a web development company based in Tirgu-Mures, Romania. During her six years of employment, she had a great opportunity not only in her role as a UX/interaction designer on exciting and challenging projects, such as brand campaigns, interactive social engagement platforms, and complex administrative solutions, but also in establishing and leading a small team of junior interaction designers.

She is a regular attendee of the yearly Axure World web event and similar Axure and UX-related events around the virtual space. She is reasonably active on LinkedIn forums and of course, the Axure forum as well, sharing knowledge, asking questions, and sometimes discussing controversial user experience topics. She has also just recently been interviewed by one of the Axure community's most well-known authors, Ezra Schwartz, in regard to her experience with shared projects. A version of this interview will be published in Ezra's latest book about prototyping with Axure RP Pro.

In her spare time, she is often found taking pictures, travelling, riding, and learning new languages, though usually not all at the same time.

Melissa Krahenbuhl is a certified professional life coach and has over 20 years of experience in accounting, administration, and organizational development.

She received numerous acknowledgements during the 10 plus years she spent at Motorola Mobility.

I would like to thank my husband, John, and our children for encouraging me every day to love unconditionally, dream big, and laugh often.

Jessamyn Smallenburg earned a Bachelor of Arts degree in Psychology, followed by a Master of Information Science degree with a specialization in Human-Computer Interaction from the University of Michigan, Ann Arbor. She has consulted on usability, user experience, and information architecture for both the health insurance and automobile industries.

Her goals are to create top-notch IA structures and user experiences to facilitate learning and interaction. When she is not working, she is inquiring into psychological concepts and pursuing the perfection of photographic technique. Her graduate school projects and nature photography portfolio are located on her website, `www.jessamynsmallenburg.com`.

I am thankful for the stellar education provided at my alma mater and for the opportunity to contribute to Packt Publishing's book on Axure RP Pro. Ultimately, it is all rooted in my thankfulness for the love of learning instilled in me by my parents and their parents before them.

Wahne Lducia Tubman is a user experience designer, user researcher, strategist, and web developer. She has over 14 years of experience in designing software interfaces, conducting user research, managing design teams, and spearheading the design phase of technology projects for companies including Goldman Sachs, JP Morgan Chase, HBO, and Standard Bank.

A global citizen with deep cultural insights, she now lives in Johannesburg. She is passionate about making applications easy to use and leveraging the power of the Internet to help companies grow their businesses online.

She founded Enwah Interactive, a software design studio. Enwah Interactive specializes in building customer-friendly solutions for banks and financial services companies. Enwah employs customer-centric design methodologies to designs and builds easy-to-use and enjoyable e-business and mobile applications for African consumers.

She studied Interactive Telecommunications at NYU's Tisch School of the Arts and holds a BA in marketing and communications from Pepperdine University.

Table of Contents

Preface

Axure has rapidly become one of the leading tools for rapid prototyping in use today. There are many reasons for Axure's popularity. You can easily create wireframes as well as generate specification documentation. Axure also provides the ability to quickly develop prototypes that can be leveraged in web browsers or as native Android, iPhone, or iPad applications. It is no wonder that Axure has become the tool of choice for a large percentage of the Fortune 100 corporations as well as User Experience professionals worldwide.

This book provides fast, practical, step-by-step recipes to create your own custom prototypes for mobile and desktop. You can now easily create your mobile-first designs as well as responsive and adaptive web design interactions. Using real-world examples developed from today's hottest technological trends, you can take your skills to the next level!

You will learn how to create custom templates for specifications to impress your clients, create Masters, and build your own custom widget libraries. You will also learn to use AxShare to quickly share password-protected prototypes with clients and teammates, and explore integrate Axure prototypes with external JavaScript libraries.

With the latest enhancements in Axure RP 7, you can easily design **Adaptive Web Layouts** (**AWD**) as well as create widgets for **Responsive Web Layouts** (**RWD**). You will learn how to include jQuery Mobile in Axure prototypes; connect to social media sites such as Facebook, Twitter, YouTube, and Pinterest; and create e-commerce order checkout flows to make your prototypes come to life. With this book and Axure RP Pro, you will witness increased productivity in your team by leveraging Axure team projects.

There are plentiful resources available to assist you no matter what your skill level. With a passionate community and exceptional technical support, any question you may have regarding Axure will quickly be answered. Visit `http://www.axure.com/community` for access to widget libraries, the Axure forum, and more resources. Welcome to the Axure community and enjoy your journey!

What this book covers

Chapter 1, *Prototyping Recipes*, introduces methods to bring paper prototypes to life, create a dynamic Breadcrumb Master, create a carousel, and build a fixed Contact Us widget.

Chapter 2, *Enhanced Prototyping Recipes*, covers using external HTML and CSS in your prototypes, using inline frames to embed YouTube and Vimeo video sources, and making active connections to social media channels such as Facebook, Twitter, and Pinterest.

Chapter 3, *Specifications*, explains how to customize specifications and standardize annotations.

Chapter 4, *Wireframes and Design*, introduces reusable custom Masters and libraries.

Chapter 5, *E-commerce Solutions*, covers business-to-business and business-to-consumer fundamentals, checkout flows, and prototyping examples.

Chapter 6, *Using Axure in Teams*, explains how to manage shared projects, work with version control, and streamline collaborative workflows.

Chapter 7, *Adaptive and Responsive Web Design with Axure*, introduces responsive design, jQuery media queries, and jQuery Mobile.

Chapter 8, *Prototyping for Mobile*, covers optimizing prototypes to be viewed as native Android, iPhone, or iPad applications as well as native web browsers.

Chapter 9, *Miscellaneous Explorations*, explains how to set up SDKs for mobile platforms as well as how to set up an equivalent LAMP (Linux, Apache, MySQL, and PHP) environment for prototyping.

What you need for this book

You will need Axure RP 7, an Internet connection, and a desire to enhance your prototyping abilities. If you do not currently have Axure, please visit `http://www.axure.com` to download a free trial version.

[A few recipes will also require Axure RP 7 Pro, a graphics editing program (for example, Adobe Photoshop), and a word processing program that can open Microsoft Word formatted documents (for example, MS Word).]

Who this book is for

This book is suitable for you if you are a User Experience professional, designer, information architect, or business analyst who wants to gain advanced prototyping skills with Axure. You can also use this book if you have some experience with prototyping and want to take your prototypes to the next level.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text are shown as follows: "In your prototype directory, open `start.html` in a browser to view the prototype."

A block of code is set as follows:

```
<VirtualHost *:80>
  ServerName svn.yourserver.com
  <Location /repos>
    DAV svn
    SVNListParentPath on
    SVNParentPath /svn/repos/
    AuthType Basic
    AuthName "Authorization Realm"
    AuthUserFile /etc/auth-file-svn
    Require valid-user
  </Location>
</VirtualHost>
```

Any command-line input or output is written as follows:

```
sudo chown www-data:www-data /svn
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "Click on the **Launch** button to start the instance.

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com` and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased through your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section for that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors' and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Prototyping Recipes

In this chapter, we will cover a few basic Axure prototype recipes. You will learn:

- Sketching, scanning, and prototyping
- Creating a dynamic Breadcrumb Master
- Generating a dynamic welcome message
- Creating a carousel with Dynamic Panel state actions – next and previous
- Building a fixed Contact Us widget
- Prototyping a Frequently Asked Questions (FAQ) page
- Validating user input
- Adding an enabled submit button to form fields
- Sending form data to Salesforce

Introduction

Axure is a leading tool for creating requirement specifications as well as generating interactive HTML wireframes and UI mockups. It is easy to use Axure and add advanced interactions to your wireframes. This chapter will show you a simple, effective method to make your sketches come to life as interactive prototypes. You will also learn how to create a carousel as well as intermediate concepts.

Sketching, scanning, and prototyping

Most folks start the design process by developing quick sketches of the concepts. These sketches can be elaborate or rudimentary. Oftentimes, these sketches evolve into paper prototypes that illustrate the flow or steps a user would take to complete a task. By scanning your drawings, making adjustments with your favorite image editing software (Gimp, Adobe Photoshop, and so on), and Axure, you can quickly create a clickable prototype.

Getting ready

To go through this recipe, you will need to have digital scans of your sketches and access to the image editing software of your choice.

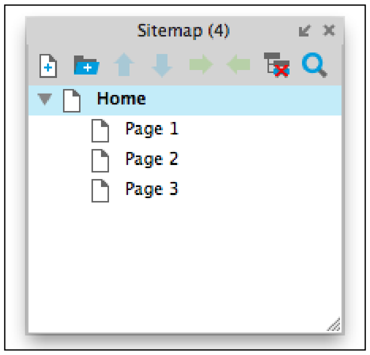
How to do it...

You will now create a carousel including thumbnails from digital scans of simple, freehand-drawn sketches.

1. Using your image-editing tool, first organize your images and crop them appropriately. You will have to organize the images and visualize the user flow just as you would do for paper prototypes.
2. Start Axure and under **Create New** select **RP File**.

    If you already have Axure open, select **File** in the main menu, and then click on **New**, in the drop-down menu to create a new RP document.

3. In the **Sitemap**, add additional child or sibling pages as necessary to complete your flow by clicking on the **Add Page** button icon or by right-clicking on any page in the sitemap. In the menu that appears, mouse over **Add**, and then click on the **Child** or the **Sibling** page.



4. Double-click on any page title in **Sitemap** to select that page. You will see the wireframe for the associated page shown.
5. While holding down the mouse button, drag the **Image** widget, and place it on the wireframe.
6. Double-click on the **Image** widget on the wireframe, and select the appropriate scanned sketch.
7. While holding down the mouse button, drag the **Hot Spot** widget, and place it over the item you would like to make clickable.
8. While holding down the mouse button, drag the corners of the **Hot Spot** widget on the wireframe to the desired size.
9. With **Hot Spot** selected, in the **Widget Interactions and Notes** pane, click on **Create Link...**.
10. In the **Sitemap** pop up, click on the associated page in the user flow.
11. Repeat steps 7 through 10 for each region on your wireframe that you would like to make clickable.
12. Repeat steps 4 through 11 for each page in **Sitemap** that you would like to make a part of the prototype.
13. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

Using this recipe, you are able to convert your paper sketches into clickable digital prototypes. Each paper sketch becomes a page in the **Sitemap** through the use of the **Image** widget. To accomplish this, you opened the scanned image with the **Image** widget to display your paper sketch on the page.

To create clickable regions, you used **Hot Spot** and associated the next page in the flow using **Create Link...**. You used as many image map regions as clickable elements needed for the interactions on a page.

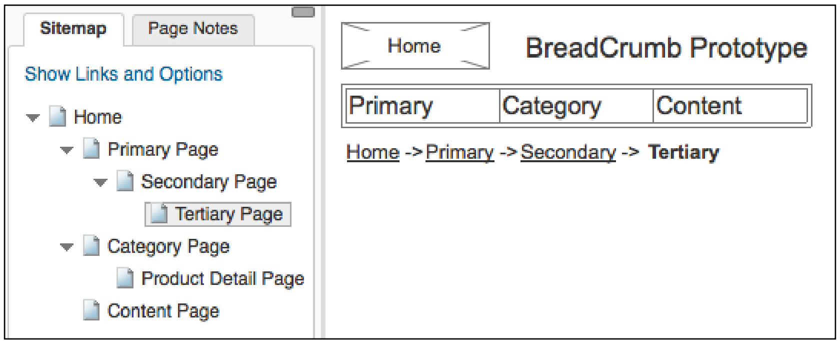
Creating a dynamic Breadcrumb Master

Using Masters in Axure allows you to create reusable components. When you make a change to a Master, the change is applied to all wireframes that contain that Master. Leveraging Masters can ensure the consistency of elements across your prototypes.

Getting ready

In this recipe you will create a dynamic Breadcrumb Master.

In Axure, verify that the **Widget Manager** and **Page Properties** panes are shown. To verify, click on **View** in the main menu and mouse over **Panes**. In the pop-up menu, make sure that a check mark is next to all items, including the **Widget Manager** and **Page Properties** panes.

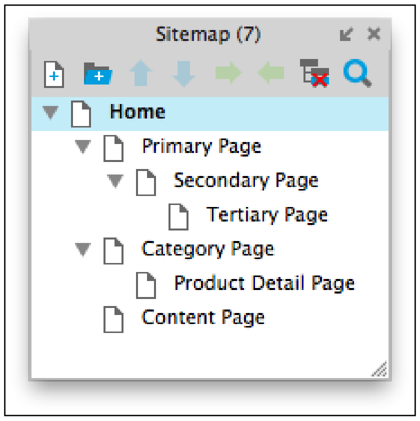


How to do it...

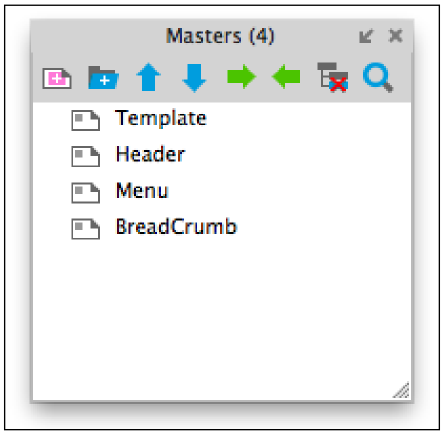
To create a dynamic Breadcrumb Master, first you will create new pages in your sitemap and three empty Masters (for example, Template, Header, Menu, and BreadCrumb). Next, you will place widgets on the Header, Menu, and BreadCrumb Masters. You will then place the Header, Menu, and BreadCrumb Masters onto the Template Master. Finally, you will drag the Template Master to all of the pages in **Sitemap**.

1. Start Axure and under **Create New** select **RP File**.

2. In the **Sitemap** create pages as follows:

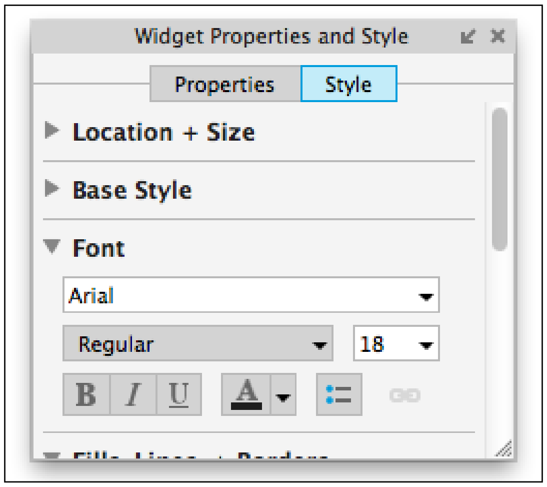


3. In the **Masters** pane, create four individual Masters, titled: **Template**, **Header**, **Menu**, and **BreadCrumb**, respectively, as shown in the following screenshot:



4. Right-click on each Master you created in step 3, mouse over **Drop Behaviour**, and click on **Lock to Master Location**. This will cause the widgets in each Master to maintain the *x* and *y* coordinates no matter where the Master is placed in a wireframe.
5. In the **Masters** pane, double-click on the **Header** Master to select it.
6. While holding down the mouse button, drag the **Placeholder** widget, and place it on the wireframe.
7. With the **Placeholder** widget selected, type `Home`, and change the **x:** `10`, **y:** `12`, **w:** `96`, and **h:** `30` (present at the top-left of the window).

8. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and type `HomeLink`.
9. While holding down the mouse button, drag the **Label** widget, and place it at the coordinates (130,18) on the wireframe.
10. With the **Label** widget selected, perform the following steps:
    1. Type `BreadCrumb Prototype`.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and type `HeaderLabel`.
11. In the **Widget Properties and Style** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **18** by clicking on the font size dropdown, mouse over **18**, and click to select:



12. In the **Masters** pane, double-click on the **Menu** Master to select it.
13. While holding down the mouse button, drag the **Classic Menu - Horizontal** widget, and place it at the coordinates (10,52) on the wireframe.
14. In the **Widget Interactions and Notes** pane, click on the **Menu Name** text field, and type `MainMenu`.
15. To name and link the primary menu item, perform the following steps:
    1. Click on the first menu item labeled **File** to select it, and type `Primary`.
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field, and type `MenuPrimary`.
    3. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.

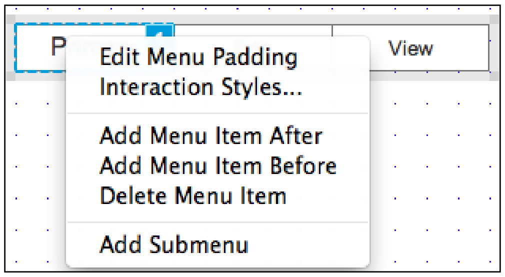
4. In the **Case Editor (OnClick)** pop up, in **Case description**, rename the case description `OpenPrimaryPage`.
5. In **Click to add actions**, click on **Open Link**.
6. In **Organize actions**, you will see the interaction description update to **Open Link in Current Window**.
7. In **Configure actions**, click on the radio button next to **Link to a page in this design**, and then click on **Primary Page**.
8. Click on **OK**.
9. In the **Widget Properties and Style** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **16** by clicking the font size dropdown, mouse over **16**, and then click on it to select.

16. To name and link the category menu item, perform the following steps:
    1. Click on the second menu item, labeled **Edit**, to select it, and type `Category`.
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field, and type `MenuCategory`.
    3. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    4. In the **Case Editor (OnClick)** pop up, in **Case description**, rename the case description `OpenCategoryPage`.
    5. In **Click to add actions**, click on **Open Link**.
    6. In **Organize actions**, you will see the interaction description update to **Open Link in Current Window**.
    7. In **Configure actions**, click on the radio button next to **Link to a page in this design**, and then click on **Category page**.
    8. Click on **OK**.
    9. In the **Widget Properties and Style** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **16** by clicking the font size dropdown, mouse over **16**, and then click on it to select.

17. To name and link the content menu item, perform the following steps:
    1. Click on the third menu item labelled **View** to select it, and type `Content`.
    2. In the **Widget Interactions and Notes** pane, click on **the Menu Item Name** text field, and type `MenuContent`.
    3. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    4. In the **Case Editor (OnClick)** pop up, in **Case description**, rename the case description `OpenContentPage`.

5. In **Click to add actions**, click on **Open Link**.
6. In **Organize actions**, you will see the interaction description update to **Open Link in Current Window**.
7. In **Configure actions**, click on the radio button next to **Link to a page in this design**, and then click on **Content Page**.
8. Click on **OK**.
9. In the **Widget Properties and Style** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **16** by clicking the font size dropdown, mouse over **16**, and click on it to select it.

18. To add a submenu item, right-click on the **Primary** menu item, and click on **Add Submenu**:



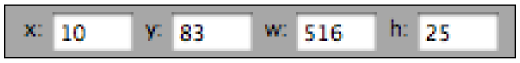
19. Click on the first submenu item, and enter `Secondary`, and then perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field, and type `MenuSecondary`.
    2. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    3. In the **Case Editor (OnClick)** pop up, in **Case description**, rename the case description `OpenSecondaryPage`.
    4. In **Click to add actions**, click on **Open Link**.
    5. In **Organize actions**, you will see the interaction description update to **Open Link in Current Window**.
    6. In **Configure actions,** click on the radio button next to **Link to a page in this design**, and then click on **Secondary Page**.
    7. Click on **OK**.

20. Right-click on the second and third submenu items, and click on **Delete Menu Item**.

21. Click on the submenu item **Secondary** to select it, and perform the following steps:
    1. Right-click on **Secondary**, and click on **Add Submenu**.
    2. Click on the first submenu item, and enter `Tertiary`.
    3. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field, and type `MenuTertiary`.
    4. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    5. In the **Case Editor (OnClick)** pop up, in **Case description**, rename the case description `OpenTertiaryPage`.
    6. In **Click to add actions**, click on **Open Link**.
    7. In **Organize actions**, you will see the interaction description update to **Open Link in Current Window**.
    8. In **Configure actions**, click on the radio button next to **Link to a page in this design**, and then click on **Tertiary Page**.
    9. Click on **OK**.
    10. Right-click on the second and third submenu items, and click on **Delete Menu Item**.

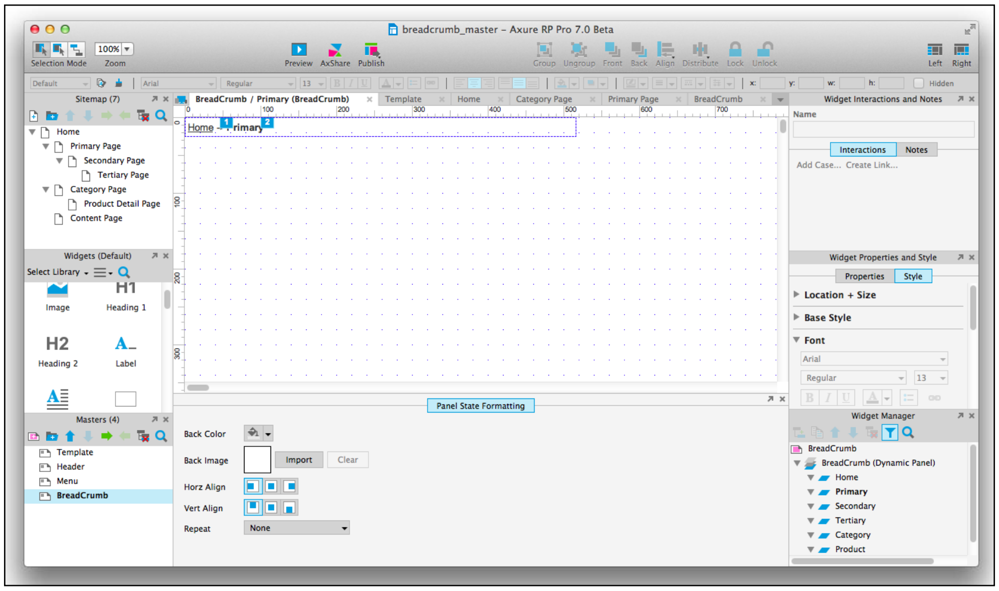
22. Click on the submenu item **Category** to select it, and perform the following steps:
    1. Right-click on **Category**, and click on **Add Submenu**.
    2. Click on the first submenu item, and enter `Product Detail`.
    3. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field, and then type `MenuCategory`.
    4. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    5. In the **Case Editor (OnClick)** pop up, in **Case description**, rename the case description to `OpenCategoryPage`.
    6. In **Click to add actions**, click on **Open Link**.
    7. In **Organize actions**, you will see the interaction description update to **Open Link in Current Window**.
    8. In **Configure actions**, click on the radio button next to **Link to a page in this design**,and then click on **Category Page**.
    9. Click on **OK**.
    10. Right-click on the second and third submenu items, and then click on **Delete Menu Item**.

23. In the **Masters** pane, double-click on the **BreadCrumb** Master to select it.
24. While holding down the mouse button, drag the **Dynamic Panel** widget, and place it on the wireframe.
25. Change the **x:** and **y:** coordinates and **w:** and **h:** to be:



x: 10 y: 83 w: 516 h: 25

26. With the **Dynamic Panel** selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Dynamic Panel Name** text field, and then type `BreadCrumb`.
    2. In the **Widget Manager**, rename **State1** `Home`.
    3. Add states to **Dynamic Panel** as follows: `Primary`, `Secondary`, `Tertiary`, `Category`, `Product`, and `Content`.
27. With **Dynamic Panel** selected, perform the following steps:
    1. Double-click on the state labeled **Primary** in the **Dynamic Panel Manager**.
    2. While holding down the mouse button, drag a **Label** widget and place it on the wireframe at coordinates (0,6).
    3. Enter `Home` as the text on the **Label** widget.
    4. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and then type `HomeBreadCrumbLink`.
    5. In the **Widget Interactions and Notes**, pane click on the **Interactions** tab, and then click on **Add Case...**.
    6. In the **Case Editor (OnClick)** pop up, in **Case description**, rename the case description `OpenHomePage`.
    7. In **Click to add actions**, click on **Open Link**.
    8. In **Organize actions**, you will see the interaction description update to **Open Link in Current Window**.
    9. In **Configure actions**, click on the radio button next to **Link to a page in this design**, and click on **Home** page.
    10. Click on **OK**.
28. You will now focus on building the Dynamic Panel states Primary, Secondary, Tertiary, Category, Product, and Content. The following screenshot shows what the Primary state will look like:



29. To build the **Dynamic Panel** states **Primary**, **Secondary**, **Tertiary**, **Category**, **Product**, and **Content**, with the **Dynamic Panel** selected perform the following step:

    1. Repeat step 27, changing the step each time with the following information:

Panel State	Coordinates for Label Widget	Label Text	Shape Name	Case description	Configure Actions Link to
Primary	(0,6)	Home	HomeBreadCrumbLink	OpenHomePage	Home
Primary	(55,6)	Primary	PrimaryBreadCrumbLink	OpenPrimaryPage	Primary Page
Secondary	(0,6)	Home	HomeBreadCrumbLink	OpenHomePage	Home
Secondary	(55,6)	Primary	PrimaryBreadCrumbLink	OpenPrimaryPage	Primary Page
Secondary	(115,6)	Secondary	SecondaryBreadCrumbLink	OpenSecondaryPage	Secondary Page
Tertiary	(0,6)	Home	HomeBreadCrumbLink	OpenHomePage	Home
Tertiary	(55,6)	Primary	PrimaryBreadCrumbLink	OpenPrimaryPage	Primary Page
Tertiary	(115,6)	Secondary	SecondaryBreadCrumbLink	OpenSecondaryPage	Secondary Page
Tertiary	(200,6)	Tertiary	TertiaryBreadCrumbLink	OpenTertiaryPage	Tertiary Page
Category	(0,6)	Home	HomeBreadCrumbLink	OpenHomePage	Home
Category	(55,6)	Category	CategoryBreadCrumbLink	OpenCategoryPage	Category Page
Product	(0,6)	Home	HomeBreadCrumbLink	OpenHomePage	Home
Product	(55,6)	Category	CategoryBreadCrumbLink	OpenCategoryPage	Category Page
Product	(125,6)	Product Detail	DetailBreadCrumbLink	OpenDetailPage	Product Detail Page
Content	(0,6)	Home	HomeBreadCrumbLink	OpenHomePage	Home
Content	(55,6)	Content	ContnetBreadCrumbLink	OpenContentPage	Content Page

30. To populate the **Template** Master with the component masters (for example, **Header**, **Menu**, and **BreadCrumb** Masters), perform the following steps:
    1. In the Masters pane, double-click on the **Template** Master to select it.
    2. While holding down the mouse button, drag the **Header** Master, and place it anywhere on the wireframe.

    In step 4, you specified **Lock to Master Location** for the **Drop Behaviour** of each Master. This causes the widgets in each Master to maintain their x and y coordinates no matter where the Master is placed in a wireframe.

    3. While holding down the mouse button, drag the **Menu** Master, and place on the wireframe.
    4. While holding down the mouse button, drag the **BreadCrumb** Master, and place it on the wireframe.
31. While holding down the mouse button, drag the **Template** Master, and place it anywhere on the wireframe. The **Template** Master will align to the fixed **X** and **Y** coordinates.
32. Below the wireframe, click on the **Page Interactions** tab, and double-click on the **OnPageLoad** interaction. In the **Case Editor (OnPageLoad)** pop up, perform the following steps:
    1. In **Case description**, rename the case description `SetBreadCrumbState`.
    2. In **Click to add actions**, click on **Dynamic Panels** to expand, and then click on **Set Panel State**.
    3. In **Organize actions**, you will see the interaction description update to **Set Panel to State**.
    4. In **Configure actions** under **Select the panels to set the state**, click on the checkbox next to the **Label** for the **Breadcrumb (Dynamic Panel)**.
    5. Click on the **Select the state** dropdown, and mouse over **Home**. Click on **Home** to select it.
    6. You will see the interaction description under **Organize actions** update to **read**.
    7. Set Template/BreadCrumb/BreadCrumb Home.
    8. Click on **OK**.
33. Repeat steps 30 to 32 for the remaining pages in **Sitemap**, modifying each **OnPageLoad** case to set the **BreadCrumb** state to the appropriate panel state corresponding to the selected page in **Sitemap** (for example, for the **Primary** page, the corresponding state would be **Primary**, and so on).

34. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

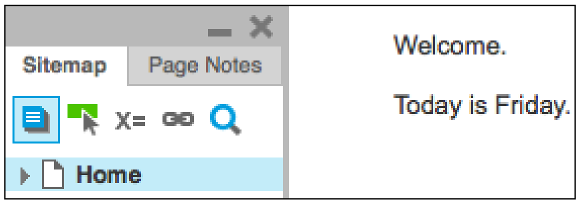
For this recipe, you used Masters, a dynamic panel, a menu widget, and text widgets to create a dynamic BreadCrumb Master. You created four Masters: Template, Header, Menu, and BreadCrumb and set the behavior of each to **Lock to Master Location**. This retained the coordinates of the widgets placed on each Master when used on a page. The Template Master contained the Header, Menu, and BreadCrumb Masters. The Menu was labeled and linked to the corresponding pages in **Sitemap**. The **BreadCrumb** Master contained a dynamic panel that had a corresponding panel state for each of the pages in **Sitemap**. For each individual panel state, label widgets were used and linked to the corresponding page in the sitemap. When a page is loaded, the **Page Interaction OnPageLoad** event sets the state of the **BreadCrumb** dynamic panel to show the correct **BreadCrumb** state.

Generating a dynamic welcome message

Using variables, you can set widget values and text dynamically. For example, when a page loads, you could show the user a welcome message based on the day of the week.

Getting ready

In this recipe, you are going to explore using built-in variables in expressions. You will show the user a welcome message using the `DayOfWeek` variable.



Welcome.

Today is Friday.

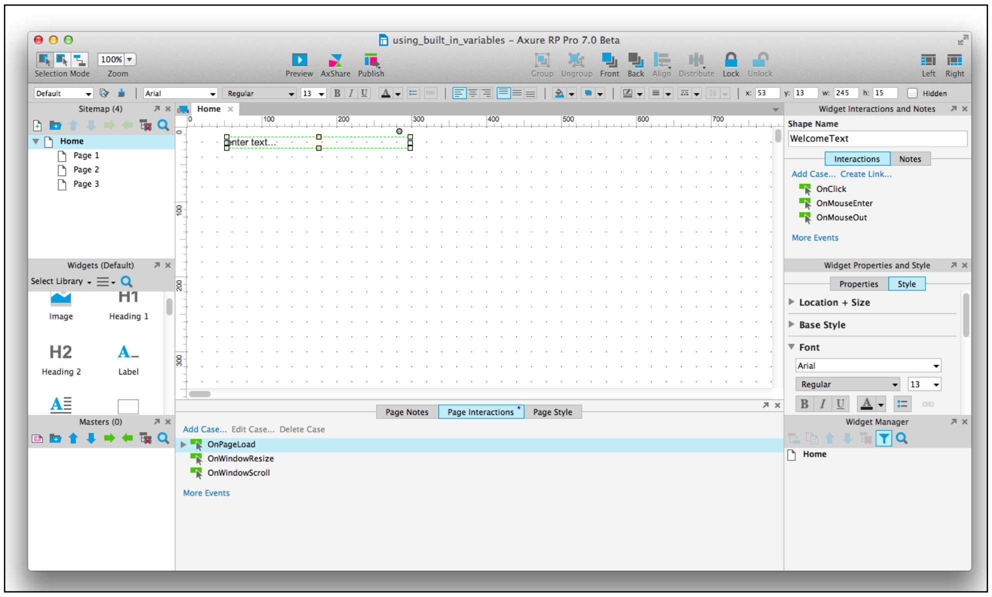
How to do it...

Perform the following steps:

1. Start Axure and under **Create New** select **RP File**.

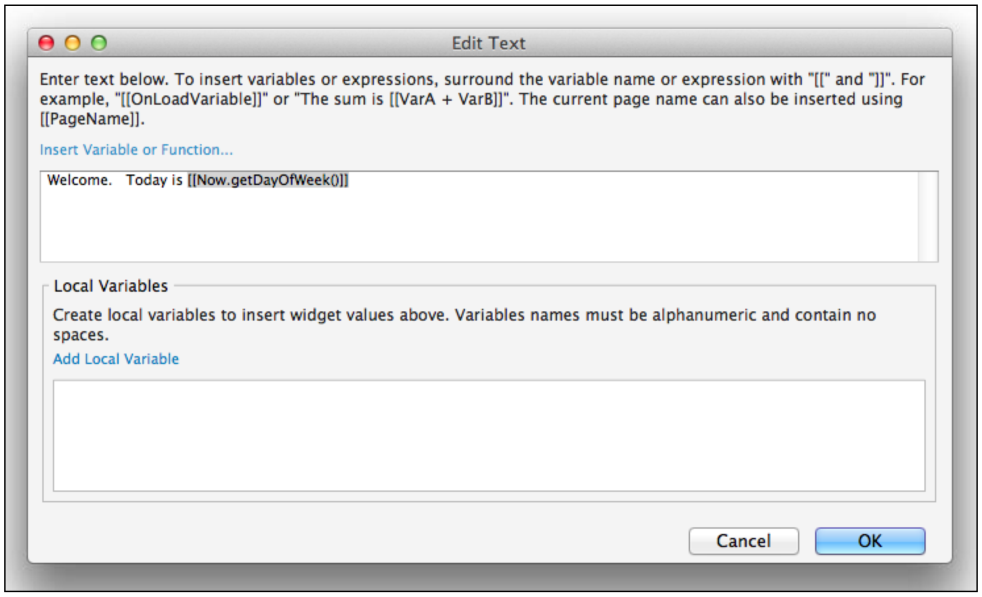
> If you already have Axure open, select **File** in the main menu, and then click on **New** in the drop-down menu to create a new RP document.

2. While holding down the mouse button, drag the **Label** widget, and place it on the wireframe at (53,13).
3. With the **Label** widget selected, in the **Widget Interactions and Notes** pane, click on the field below **Shape Name**, and then type `WelcomeText`.
4. In the **Page Properties** pane, click on the **Page Interactions** tab and double-click on the **OnPageLoad** interaction. The page will appear as shown in the following screenshot:



5. In the **Case Editor (OnPageLoad)** pop up, in **Case description**, type `DisplayMessage`.
6. In **Click to add actions**, click on **Set Text**.
7. In **Organize actions**, you will see the interaction description update to **Set Text**.

8. In **Configure actions**, under **Select the widgets to set text**, click in the checkbox next to **Label** for the **Welcome Text (Shape)**.
9. In **Configure actions** under **Set text to**, set the dropdown to **value**, and click on the **fx** button to bring up the **Edit Text** pop up.
10. In the **Edit Text** popup, enter in the text field `Welcome. Today is`.
11. Click on the **Insert Variable** or **Function...** link to open the drop-down menu; scroll to **Date**; click on **Date** to expand the selection; and click on **getDayOfWeek()**, as shown in the following screenshot:



Enter text below. To insert variables or expressions, surround the variable name or expression with "[[" and "]]". For example, "[[OnLoadVariable]]" or "The sum is [[VarA + VarB]]". The current page name can also be inserted using [[PageName]].

Insert Variable or Function...

Welcome. Today is [[Now.getDayOfWeek()]]

Local Variables

Create local variables to insert widget values above. Variables names must be alphanumeric and contain no spaces.

Add Local Variable

12. Click on **OK**.
13. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and then clicking on **Generate HTML Files...**.

How it works...

For this recipe, you used the **Label** widget, and using the **Widget Interactions and Notes** pane, you applied a label to the widget. Next in the **Page Properties** pane, you set a case on the **OnPageLoad** interaction to **Set Variable/Widget value(s)**. Finally, in the **Edit Text** popup, you used a built-in variable.

There's more...

At times you may find that your **Label** widget is not displaying the built-in variable. One possible cause is that the length of the **Label** widget is not long enough to display all of the characters.

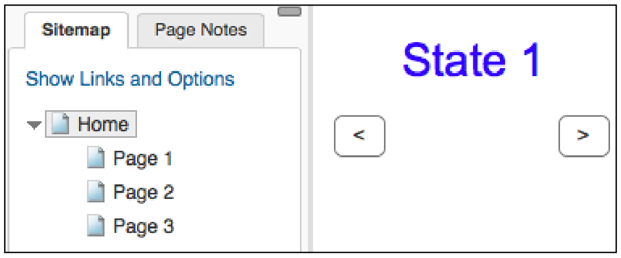
> For a list of variables available in Axure 7, visit `http://www.axure.com/forum/tips-tricks-examples/8030-v7-variables-list.html`

Creating a carousel with Dynamic Panel state actions – next and previous

Dynamic Panel widgets allow you to demonstrate interactions by showing, hiding, or swapping content. A Dynamic Panel consists of one or more panel states of which only one panel state can be shown at a time. When using Dynamic Panels, you can set the current panel state based on user interaction with additional widgets (for example, button).

Getting ready

You will use a Dynamic Panel and two button widgets to create a carousel. The left-hand side button widget will set the Dynamic Panel state to previous. The right-hand side button widget will set the Dynamic Panel state to next. There will be four states of the Dynamic Panel, states 1 to 4. You will also set the interactions on the button widgets to enable looping. For example, if the Dynamic Panel is at state 4, and the left-hand side button widget is clicked, the Dynamic Panel action will loop and show **State 1**, as shown in the following screenshot:



How to do it...

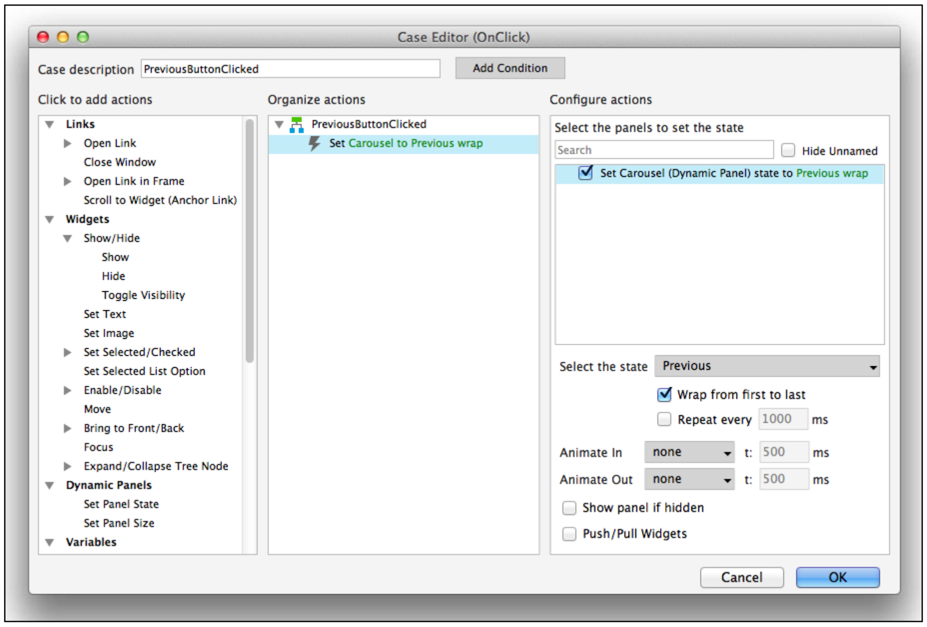
In this recipe, you will set the next and previous state of a Dynamic Panel based on user interaction with the button widgets.

1. Start Axure and under **Create New** select **RP File**.
2. While holding down the mouse button, drag the **Dynamic Panel** widget, and place it on the **Home** wireframe at (0,0).
3. With **Dynamic Panel** selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and then type `Carousel`.
    2. In the **Widget Manager**, double-click on **State1** to select it.
    3. While holding down the mouse button, drag a **Label** widget, and place it on the wireframe at coordinates (55,20).
    4. With the **Label** widget selected, type `State 1` as the text on the **Label** widget.
    5. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and then type `LabelState1`.
    6. In the **Widget Properties and Style** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **28** by clicking on the **Font** size dropdown, mouse over **28**, and then click on it to select it.
4. With the **Dynamic Panel** selected, perform the following steps:
    1. In the **Widget Manager,** right-click on **State1**, and click on **Add State**. This will create a second state **State2**.
    2. Right-click on **State1**, and click on **Add State two more times** to create **State3** and **State4**.
5. With **Dynamic Panel** selected, perform the following steps:
    1. In the **Widget Manager**, double-click on **State2** to select it.
    2. While holding down the mouse button, drag a **Label** widget, and place it on the wireframe at coordinates (55,20).
    3. With the **Label** widget selected, type `State 2` as the text on the **Label** widget.

  4. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and then type `LabelState2`.
  5. In the **Widget Properties and Style** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **28** by clicking on the Font size dropdown, mouse over **28**, and click on it to select it.

6. Repeat step 5, creating new **Label** widgets on **State3** and **State4** and labeling as appropriate.
7. In **Sitemap**, double-click on **Home** to select the **Home** wireframe.
8. While holding down the mouse button, drag the **Button Shape** widget, and place it on the wireframe at (13,69), changing the text to **<**, and **w:** to **31**.
9. With the **Button Shape** widget selected, in the **Widget Interactions and Notes** pane, click on the **Shape Name** field, and then type `PreviousButton`.
10. In the **Widget Interactions and Notes** pane, with the **Interactions** tab selected, double-click on the **OnClick** interaction, and perform the following steps:
    1. In the **Case Editor (OnClick)** pop up, rename the **Case description** `PreviousButtonClicked`.
    2. In **Click to add actions** click on **Set Panel State**.
    3. In **Organize actions**, you will see the interaction description update to **Set Panel to State**.
    4. In **Configure actions**, under **Select the panels to set the state**, click on the checkbox next to the label for **Carousel Dynamic Panel**.
    5. Click on the **Select the state** dropdown, and mouse over **Previous**. Click on **Previous** to select.
    6. Click on the checkbox next to **Wrap from first to last**.
    7. You will see the interaction description under **Organize actions** update to read **Set Carousel state to Previous wrap**.

8. Click on **OK**. The page will look like the following screenshot:



11. While holding down the mouse button, drag the **Button Shape** widget, and place it on the wireframe at (152,69), changing the text to **>**, and the **w:** to **31**.
12. With the **Button Shape** widget selected, in the **Widget Interactions and Notes** pane, click on the **Shape Footnote and Name** field, and then type `NextButton`.
13. In the **Widget Interactions and Notes** pane, with the **Interactions** tab selected, double-click on the **OnClick** interaction, and perform the following steps:
    1. In the **Case Editor (OnClick)** pop up, rename the **Case description** `NextButtonClicked`.
    2. In **Click to add actions**, click on **Set Panel State**.
    3. In **Organize actions**, you will see the interaction description update to **Set Panel to State**.
    4. In **Configure actions** under **Select the panels to set the state**, click on the checkbox next to the label for **Carousel Dynamic Panel**.
    5. Click on the **Select the state** dropdown, and mouse over **Next**. Click on **Next** to select.
    6. Click on the checkbox next to **Wrap from last to first**.
    7. You will see the interaction description under **Organize actions** update to read **Set Carousel state to Previous wrap**.
    8. Left-click on **OK**.

14. You can now choose to preview or save a copy of the prototype. To preview the prototype, click the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish** and clicking on **Generate HTML Files...**.

How it works...

For this recipe, you used a **Dynamic Panel**, two **Button Shape** widgets and two text widgets. You used the two **Button Shape** widgets to control the previous and next states of **Dynamic Panel**. By using the **Widget Interactions and Notes** pane, you applied a label to the widget. For the **Button Shape** widgets, you used the **Widget Interactions and Notes** pane to create an **OnClick** interaction. Using **Case Editor**, you defined a **Set Panel state(s) to State(s)** action and selected either the **Previous** or the **Next** state. You also enabled the looping or wrapping of panel states from first to last.

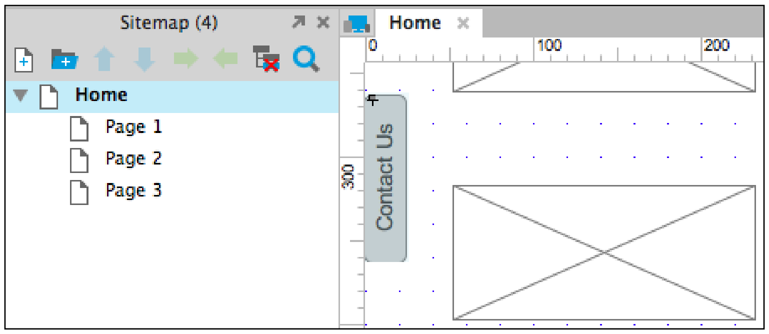
You also created a number of states for our **Dynamic Panel** and edited the wireframes for each state. Clicking our **Button Shape** widgets would cause the Dynamic Panel to change state as defined for that **Button Shape**.

Building a fixed Contact Us widget

One popular design pattern in use today is leveraging fixed widgets on a page. A fixed widget remains in the same place on a web page when the user scrolls in the browser window. In Axure, you can create this effect by **pinning** a Dynamic Panel on a page.

Getting ready

In this recipe, you are going to build a fixed **Contact Us** widget.



How to do it...

Perform the following steps:

1. Start Axure and under **Create New** select **RP File**.
2. While holding down the mouse button, drag the **Placeholder** widget, and place it on the wireframe.
3. Repeat step 2 to create a long vertical list (that is, 16 or more) of **Placeholder** widgets that will enable our prototype to have a vertical scroll bar.
4. While holding down the mouse button, drag the **Button Shape** widget, and place it on the wireframe at (0,300).
5. With the **Button Shape** widget selected, perform the following steps:
    1. Enter `Contact Us`.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and then type `ContactUsButtonShape`.
    3. In the **Widget Properties and Style** pane, click on the **Style** tab, and then click on **Location + Size** to expand.
    4. Click on the **Left:** field and enter `-38`.
    5. Click on the **Rot:** field and enter `90` to rotate the **Button Shape** widget.
    6. Click on the **Text:** field, and enter `270` to rotate the **Button Shape** widget text.
    7. Scroll down to **Fills, Lines, + Borders**, and click on the paint bucket icon drop-down menu.
    8. Click on the fifth vertical gray color block on the left-hand side. You will see the **#** field update with a value of **CCCCCC**.
6. Right-click on the **Button Shape** widget in the wireframe pane, and click on **Convert To Dynamic Panel**.
7. Right-click on the **Button Shape** widget in the wireframe pane, and click on **Pin to Browser...**.
8. In the **Pin to Browser** pop up, click on the checkbox next to **Pin to browser window**.
9. Under the **Horizontal Pin** section, click on the radio button next to **Left**.
10. Under the **Vertical Pin** section, click on the radio button next to **Middle**.
11. Click on **OK**.
12. You can now choose to preview or save a copy of the prototype. To preview the prototype, click the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

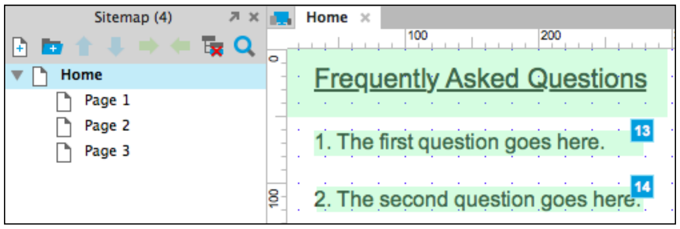
For this recipe, you used a **Button Shape** widget. You used the **Widget Interactions and Notes** pane to rotate the **Button Shape** widget, rotated the Button Shape widget text, and applied a background color to the Button Shape widget. After moving the Button Shape widget in the wireframe pane, you then converted the Button Shape widget to a Dynamic Panel. Finally, you used the **Pin to Browser...** functionality of the Dynamic Panel to fix the converted Button Shape widget to the browser.

Prototyping a Frequently Asked Questions (FAQ) page

To prototype a **Frequently Asked Questions** (**FAQ**) page that jumps to an answer when a question is clicked, you can use the Scroll to Widget (Anchor link) action. The Scroll to Widget (Anchor link) action causes the browser to automatically advance to predetermined areas of a page. You will use **Label** widgets as **anchors** to scroll to.

Getting ready

In this recipe, you are going to build an FAQ page, as shown in the following screenshot, that advances to the answer when the user clicks on a question:


Sitemap (4)
Home
Page 1
Page 2
Page 3
Home
100
200
0
100
Frequently Asked Questions
1. The first question goes here.
13
2. The second question goes here.
14

How to do it...

For this recipe, you will use Label widgets and a Horizontal Line widget. Perform the following steps:

1. Start Axure and under **Create New** select **RP File**.
2. While holding down the mouse button, drag the **Label** widget, and place it on the wireframe at (35,10).
3. With the **Label** widget selected, perform the following steps:
    1. Type `Frequently Asked Questions`.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and then type `FAQTop`.

4. In the **Widget Properties and Style** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, mouse over **16**, and then click on it to select it.
5. While holding down the mouse button, drag the **Label** widget, and place it on the wireframe (for example, at coordinates (35,60)) and type `1. The first question goes here`. Change **w:** to `300`.
6. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `Question1`.
7. Perform the following step to create Questions 2 to 12.
    1. Repeat steps 5 and 6 changing the steps each time using the following information:

Coordinates	Label Text	Shape Name
(35,100)	2. The second question goes here.	Question2
(35,140)	3. The third question goes here.	Question3
(35,180)	4. The fourth question goes here.	Question4
(35,220)	5. The fifth question goes here.	Question5
(35,260)	6. The sixth question goes here.	Question6
(35,300)	7. The seventh question goes here.	Question7
(35,340)	8. The eighth question goes here.	Question8
(35,380)	9. The ninth question goes here.	Question9
(25,420)	10. The tenth question goes here.	Question10
(25,460)	11. The eleventh question goes here.	Question11
(25,500)	12. The twelfth question goes here.	Question12

8. While holding down the mouse button, drag the **Horizontal Line** widget, and place it on the wireframe at (35, 550), and then change **w:** to `265`.
9. While holding down the mouse button, drag the **Label** widget and place it on the wireframe at (35,600), and type `1. The first question goes here`. Change **w:** to `300`.
10. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `QuestionA1`.
11. Perform the following step to create Question and Answer Labels 2 to 12.
    1. Repeat steps 9 and 10, changing the steps each time using the following information:

Coordinates	Label Text	Shape Name
(35,750)	2. The second question goes here.	QuestionA2
(35,900)	3. The third question goes here.	QuestionA3
(35,1050)	4. The fourth question goes here	QuestionA4
(35,1200)	5. The fifth question goes here.	QuestionA5
(35,1350)	6. The sixth question goes here.	QuestionA6
(35,1500)	7. The seventh question goes here.	QuestionA7
(35,1650)	8. The eighth question goes here.	QuestionA8
(35,1800)	9. The ninth question goes here.	QuestionA9
(25,1950)	10. The tenth question goes here.	QuestionA10
(25,2100)	11. The eleventh question goes here.	QuestionA11
(25,2250)	12. The twelfth question goes here.	QuestionA12

12. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at (55,630), and then type `The answer goes here`. There could be many lines of text here. Change **w:** to `195`.
13. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `Answer1`.
14. Perform the following step to create Answer Labels 2 to 12.
    1. Repeat steps 12 and 13 using the same answer label text in step 11. Change the steps each time with the following information:

Coordinates	Shape Name
(55,780)	Answer2
(55,930)	Answer3
(55,1080)	Answer4
(55,1230)	Answer5
(55,1380)	Answer6
(55,1530)	Answer7
(55,1680)	Answer8
(55,1830)	Answer9
(55,1980)	Answer10
(55,2130)	Answer11
(55,2280)	Answer12

15. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at (55,680), and type `Back to Top`.

16. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `BackToTop1`.
17. Perform the following step to create Back to Top Label widgets 2 to 12.
    1. Repeat steps 15 and 16 using the same answer label text in step 15. Change the steps each time with the following information:

Coordinates	Shape Name
(55,830)	BackToTop2
(55,980)	BackToTop3
(55,1130)	BackToTop4
(55,1280)	BackToTop5
(55,1430)	BackToTop6
(55,1580)	BackToTop7
(55,1730)	BackToTop8
(55,1880)	BackToTop9
(55,2030)	BackToTop10
(55,2180)	BackToTop11
(55,2330)	BackToTop12

18. Click on the **Label** widget named **Question1** at (35,60). With the **Label** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and double-click on the **OnClick** interaction.
    2. In the **Case Editor** popup, in **Case description**, type `Question1Clicked`.
    3. In **Click to add** actions, click on **Scroll to Widget (Anchor Link)**.
    4. In **Organize actions**, you will see the interaction description update to **Scroll to Widget**.
    5. In **Configure actions**, under **Select widget**, scroll to **QuestionA1 (Shape)**, and click on the checkbox next to **QuestionA1 (Shape)**.

    [The box must be checked for the step to work properly.]

    6. Click on the radio button next to **Scroll vertically only**.
    7. Click on the **Animate** dropdown, mouse over **bounce**, and then click to select.
    8. You will see the interaction description under **Organize actions** update to read **Scroll to FAQ Top (y only) bounce 500 ms**.

  9. Click on **OK**.

19. Click on the **Label** widget named **BackToTop1** at (55,680). With the **Label** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and double-click on the **OnClick** interaction.
    2. In the **Case Editor** popup, in **Case description**, type `BackToTopClicked`.
    3. In **Click to add actions**, click on **Scroll to Widget (Anchor Link)**.
    4. In **Organize actions**, you will see the interaction description update to **Scroll to Widget**.
    5. In **Configure actions** under **Select widget**, click on the checkbox next to **FAQTop (Shape)**.
    6. Click on the radio button next to **Scroll vertically only**.
    7. Click on the **Animate** dropdown, mouse over **bounce**, and click on it to select it.
    8. You will see the interaction description under **Organize actions** update to read **Scroll to FAQ Top (y only) bounce 500 ms**.
    9. Click on **OK**.

20. You will now finish creating scroll to interactions for Questions 2 to 12 and the associated Back To Top labels. Perform the following step:
    1. Repeat steps 18 and 19. In the table that follows, use Coordinates and Question Name (columns 1 and 2) to change step 18. Use Coordinates and BackToTop Name (columns 3 and 4) to change step 19. Change the steps each time with the following information:

Coordinates	Question Name	Coordinates	BackToTop Name
(35,750)	QuestionA2	(55,780)	BackToTop2
(35,140)	QuestionA3	(55,930)	BackToTop3
(35,180)	QuestionA4	(55,1080)	BackToTop4
(35,220)	QuestionA5	(55,1230)	BackToTop5
(35,260)	QuestionA6	(55,1380)	BackToTop6
(35,300)	QuestionA7	(55,1530)	BackToTop7
(35,340)	QuestionA8	(55,1680)	BackToTop8
(35,380)	QuestionA9	(55,1830)	BackToTop9
(35,420)	QuestionA10	(55,1980)	BackToTop10
(35,460)	QuestionA11	(55,2130)	BackToTop11
(35,500)	QuestionA12	(55,2280)	BackToTop12

21. You can now choose to preview or save a copy of the prototype. To preview the prototype, click the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

For this recipe, you used **Label** widgets to create **Scroll to Widget (Anchor Link)** animation. The Label widgets became anchors for the browser to scroll to.

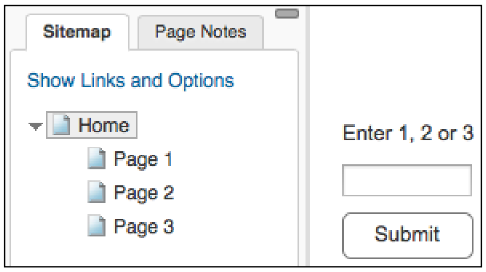
Label widgets were also used as clickable regions. For these Label widgets, you used the **Widget Interactions and Notes** pane to create an **OnClick** interaction. Using the **Case Editor**, you defined a **Scroll to Widget (Anchor Link)** action with animation.

Validating user input

A common interaction with web forms is to process user input and display a confirmation message. This can be accomplished in Axure using a Dynamic Panel, Text Field and Button widget.

Getting ready

In this recipe, you will validate user input and set the current state of a Dynamic Panel to display a message to the user. You will use a Dynamic Panel, Text Field, and Button widget to create a user input form.

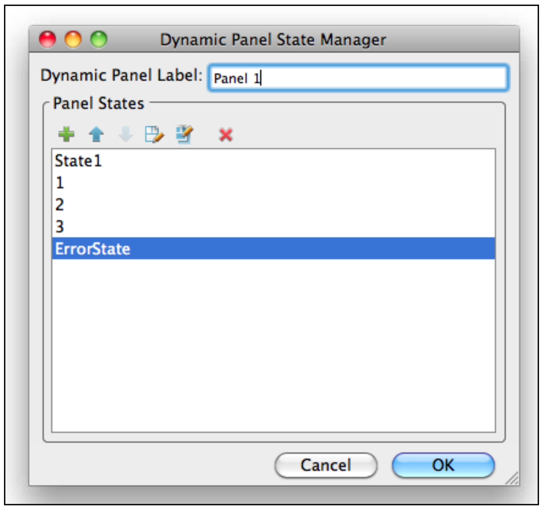


How to do it...

In this recipe, you will set the state of a Dynamic Panel based on user interaction with a Button Shape widget.

1. Start Axure and under **Create New** select **RP File**.

2. While holding down the mouse button, drag the **Dynamic Panel** widget, and place it on the **Home** wireframe at (0,0).
3. With the **Dynamic Panel** selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Dynamic Panel Name** text field, and then type `Panel1`.
    2. In the **Widget Manager**, double-click on the **Panel1 (Dynamic Panel)** icon to open the **Dynamic Panel State Manager**.
    3. In the **Dynamic Panel State Manager** popup, click on the green **+** sign and type `1` to name the new state.
    4. In the **Dynamic Panel State Manager** popup, click on the green **+** sign and type `2` to name the new state.
    5. In the **Dynamic Panel State Manager** popup, click on the green **+** sign and type `3` to name the new state.
    6. In the **Dynamic Panel State Manager** popup, click on the green **+** sign and type `ErrorState` to name the new state.
    7. Click on **OK.**



Dynamic Panel Label: Panel 1

State1
1
2
3
ErrorState

4. With the **Dynamic Panel** selected, perform the following steps:
    1. In the **Widget Manager** double-click on the state labeled **1** to select it.
    2. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at (16, 26), and type `You Entered 1!`.
    3. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `TextPanel1`.

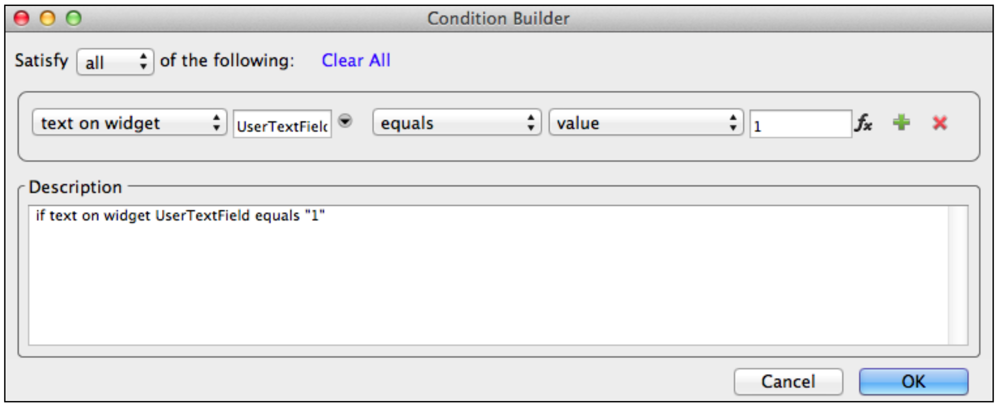
    4. In the **Widget Interactions and Notes** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **16** by clicking the **Font** size dropdown, mouse over **16**, and click on it to select it.
    5. In the **Widget Manager**, double-click on the state labeled **2** to select it.
    6. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at (16, 26) and then type `You Entered 2!`.
    7. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `Label2`.
    8. In the `Widget Interactions and Notes` pane, click on the **Formatting** tab, and increase the **Font** size to **20**.
    9. In the **Widget Manager** double-click on the state labeled **3** to select it .
    10. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at (16, 26), and type `You Entered 3!`.
    11. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `Label3`.
    12. In the **Widget Interactions and Notes** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, mouse over **16**, and click on it to select it.

5. With the **Dynamic Panel** selected, perform the following steps:
    1. In the **Widget Manager** double-click on the state labeled `ErrorState` to select.
    2. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at (16, 26), and then type `You did not enter 1, 2 or 3 !`.
    3. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Shape Name** field, type `LabelErrorState`.
    4. In the **Widget Interactions and Notes** pane, click on the **Style** tab, and then scroll to the **Font** section. Increase the font size to **16** by clicking the **Font** size dropdown, mouse over **16**, and click on it to select it.

6. Double-click on the **Home** page in **Sitemap**, and perform the following steps:
    1. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at (20, 72), and then type `Enter 1, 2 or 3`.
    2. With the **Label** widget selected, in the **Widget Interactions and Notes** pane in the **Label** field, type `UserPromptLabel`.
    3. While holding down the mouse button, drag the **Text Field** widget, and place it on the wireframe at (20,99).
    4. With the **Text Field** widget selected, in the **Widget Interactions and Notes** pane in the **Text Field Name** field, type `UserTextField`.

5. While holding down the mouse button, drag the **Button Shape** widget, and place it on the wireframe at (20,129).
6. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and type `SubmitButton`.

7. With the **Button Shape** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    2. In the **Case Editor (OnClick)** pop up, in **Case description** rename the case description `UserEntered1`.
    3. Click on **Add Condition**.
    4. In the **Condition Builder** pop up from the drop-down menus, select **text on widget**, **UserTextField (Text Field)**, **equals**, and **value**. Enter `1` in the **Text Field**.
    5. In the **Description Label**, you should see the condition **if text on UserTextField equals "1"**, as shown in the following screenshot.
    6. Click on **OK**.



if text on widget UserTextField equals "1"

8. To continue defining the case you started in Step 7, perform the following steps in **Case Editor**:
    1. In **Click to add actions** in the **Dynamic Panel** dropdown, click on **Set Panel State**.
    2. In **Organize actions**, you will see the interaction description update to **Set Panel to State**.
    3. In **Configure actions** under **Select the panels to set the state**, click on the checkbox next to **Panel1 (Dynamic Panel)**. You will see the text next to the checkbox update to read **Set Panel1 (Dynamic Panel) state to State1**.

    4. Click on the **Select the state** dropdown, and mouse over the state labeled **1**. Click on it to select it.
    5. You will see the interaction description under **Organize actions** update to read **Set Panel1 state to 1**.
    6. Click on **OK**.

9. With the **Button Shape** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    2. In the **Case Editor (OnClick)** pop up, in **Case description** rename the case description `UserEntered2`.
    3. Click on **Add Condition**.
    4. In the **Condition Builder** pop up from the drop-down menus, select **text on widget**, **UserTextField (Text Field)**, **equals**, and **value**. Enter `2` in the **Text Field**.
    5. In the **Description Label**, you should see the condition **if text on UserTextField equals "2"**.
    6. In **Click to add actions** in the **Dynamic Panels** dropdown, click on **Set Panel State**.
    7. In **Organize actions**, you will see the interaction description update to **Set Panel to State**.
    8. In **Configure actions** under **Select the panels to set the state**, click on the checkbox next to **Panel1 (Dynamic Panel)**. You will see the text next to the checkbox update to read **Set Panel1 (Dynamic Panel) state to State1**.
    9. Click on the **Select the state** dropdown and mouse over the state labeled **2**. Click to select.
    10. You will see the interaction description under **Organize actions** update to read **Set Panel1 state to 2**.
    11. Click on **OK**.

10. With the **Button Shape** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    2. In the **Case Editor** popup, in Case description, rename the case description `UserEntered3`.
    3. Click on **Add Condition**.
    4. In the **Condition Builder** pop up from the drop-down menus, select **text on widget**, **UserTextField (TextField)**, **equals**, and **value**. Enter `3` in the **Text Field**.

    5. In the **Description Label**, you should see the condition **if text on UserTextField equals "3"**.
    6. In **Click to add actions** in the **Dynamic Panel** dropdown, click on **Set Panel State**.
    7. In **Organize actions**, you will see the interaction description update to **Set Panel State**.
    8. In **Configure actions** under **Select the panels to set the state**, click on the checkbox next to **Panel1 (Dynamic Panel)**. You will see the text next to the checkbox update to read **Set Panel1 (Dynamic Panel) state to State1**.
    9. Click on the **Select the state** dropdown, and mouse over the state labeled **3**. Click to select.
    10. You will see the interaction description under **Organize actions** update to read **Set Panel1 state to 3**.
    11. Click on **OK**.

11. With the **Button Shape** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Add Case...**.
    2. In the **Case Editor** popup, in **Case description**, rename the case description to `UserError`.
    3. In **Click to add actions** in the **Dynamic Panel** dropdown, left-click on **Set Panel State**.
    4. In **Organize actions** you will see the interaction description update to **Set Panel to State**.
    5. In **Configure actions** under **Select the panels to set the state**, click on the checkbox next to **Panel1 (Dynamic Panel)**. You will see the text next to the checkbox update to read **Set Panel1 (Dynamic Panel) state to State1**.
    6. Click on the **Select the state** dropdown, and mouse over the state labeled `ErrorState`. Click to select.
    7. You will see the interaction description under **Organize actions** update to read **Set Panel1 state to ErrorState**.
    8. Click on **OK**.

12. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish** , and clicking on **Generate HTML Files...**.

How it works...

For this recipe, you used a Dynamic Panel, a Text Field, a Button Shape, and several Text widgets. The **Text Field** widget provides the user an area to input a value. You used the Button Shape widget to control the states of the Dynamic Panel based on the value entered into the Text Field widget.

For the Button Shape widget, you used the **Widget Interactions and Notes** pane to create OnClick interactions. Using **Case Editor**, you defined conditions to set a **Set Panel state(s) to State(s)** action depending on which values were entered in the Text Field widget. You also provided a case that would set the Dynamic Panel to an **ErrorState** if no defined conditions were met.

Adding an enabled submit button to form fields

As you build your prototypes, you often will want to add buttons and form fields. This recipe will show you how to quickly build a simple form with an enabled **Submit** button.

How to do it...

In this recipe, you will build a simple form with a pair of labels with text fields and a Button widget.

1. Start Axure and under **Create New** select **RP File**.
2. While holding down the mouse button, drag the **Label** widget, place it on the wireframe at coordinates (15,37), and then type `Name`.
3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and type `LabelName`.
4. While holding down the mouse button, drag the **Text Field** widget and place on the wireframe at (74,32).
5. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** text field, and type `TextFieldName`.
6. While holding down the mouse button, drag the **Label** widget, place on the wireframe at coordinates (15,91), and then type `Email`.
7. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and type `TextPanelEmail`.
8. While holding down the mouse button, drag the **Text Field** widget, and place it on the wireframe at (74,86).
9. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** text field, and type `TextFieldEmail`.

10. While holding down the left mouse button, drag the **Button Shape** widget, place it on the wireframe, and then type **Submit**.
11. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field, and type `SubmitButton`.
12. Right-click on the **TextFieldEmail** text field, click on **Assign Submit Button** in the pop-up menu.
13. Click on the checkbox next to **SubmitButton (Shape)**.
14. Click on the **Button Shape** widget to select, in the **Widget Interactions and Notes** pane, click on the **Interactions** tab, and then click on **Create Link...**.
15. In the **Sitemap** pop up, click on **Page 1**.
16. Double-click on **Page 1** in the **Sitemap**.
17. While holding down the mouse button, drag the **Label** widget, place it on the wireframe, and type `Thank You for Your Comments!`.
18. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

For this recipe, you used the Label, Text Field, and Button widgets to create an enabled submit button. You enabled the submit button by enabling the **OnClick** interaction to open in your current window the page that contained our message to the user. While having the submit button selected, you used **Create Link...** in the Widget Interactions and Notes pane to complete the association.

There's more...

At times you may find that your Text Field widget is not responding in your prototype when a user left-clicks on the Text Field. This can be caused by other widgets overlapping the Text Field widget. For this recipe, make sure that the Label widget does not overlap the Text Field widget.

Sending form data to Salesforce

Salesforce is one of the most popular cloud-based Customer Relationship Management solutions available today. In this recipe, you will enable an Axure prototype to submit form data to Salesforce.

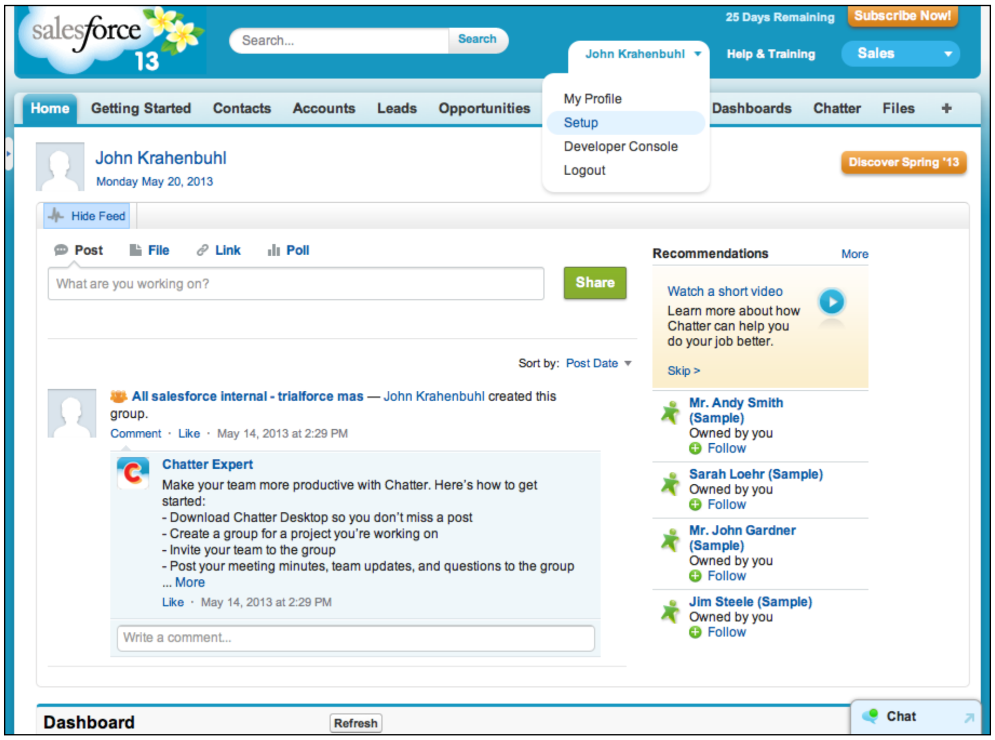
Getting ready

To step through this recipe, you will need access to Salesforce. If you do not have a Salesforce account, you can sign up for a free, 30-day trial by visiting `https://www.salesforce.com/form/signup/freetrial-lb.jsp`.

How to do it...

In this recipe, you will use an Inline Frame widget, and an external HTML to create a form that will populate Salesforce when submitted.

1. Login to Salesforce.
2. Click on **Setup** as shown in the following screenshot:



John Krahenbuhl
Monday May 20, 2013

What are you working on?

All salesforce internal - trialforce mas — John Krahenbuhl created this group.
Comment · Like · May 14, 2013 at 2:29 PM

Chatter Expert
Make your team more productive with Chatter. Here's how to get started:
- Download Chatter Desktop so you don't miss a post
- Create a group for a project you're working on
- Invite your team to the group
- Post your meeting minutes, team updates, and questions to the group
... More
Like · May 14, 2013 at 2:29 PM

Write a comment...

Watch a short video
Learn more about how Chatter can help you do your job better.
Skip >

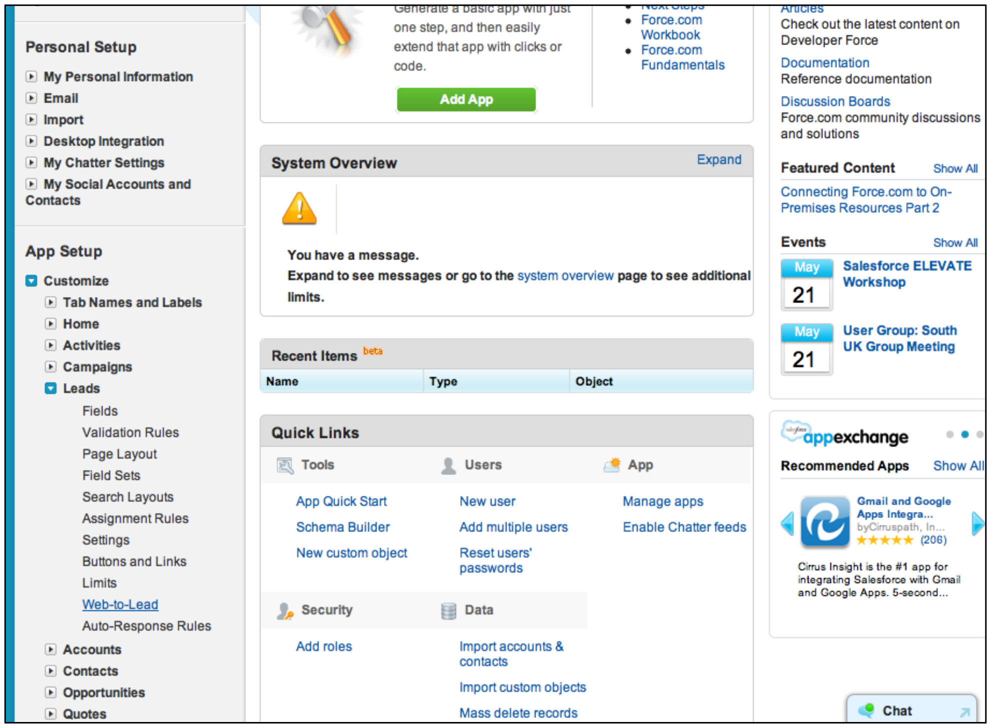
Mr. Andy Smith (Sample)
Owned by you
Follow

Sarah Loehr (Sample)
Owned by you
Follow

Mr. John Gardner (Sample)
Owned by you
Follow

Jim Steele (Sample)
Owned by you
Follow

3. On the left-hand side in the **Build** pane, navigate to **Customize** | **Leads** | **Web-to-Lead**, as shown in the following screenshot:



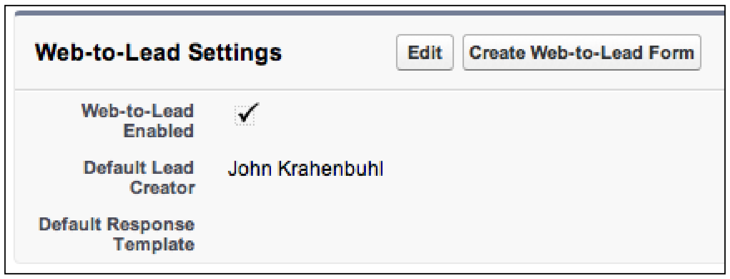
Generate a basic app with just one step, and then easily extend that app with clicks or code.

You have a message.
Expand to see messages or go to the system overview page to see additional limits.

Name	Type	Object

Check out the latest content on Developer Force
Documentation
Reference documentation
Discussion Boards
Force.com community discussions and solutions
Connecting Force.com to On-Premises Resources Part 2
May 21 Salesforce ELEVATE Workshop
May 21 User Group: South UK Group Meeting
Gmail and Google Apps Integra... byCirruspath, In... (206)
Cirrus Insight is the #1 app for integrating Salesforce with Gmail and Google Apps. 5-second...

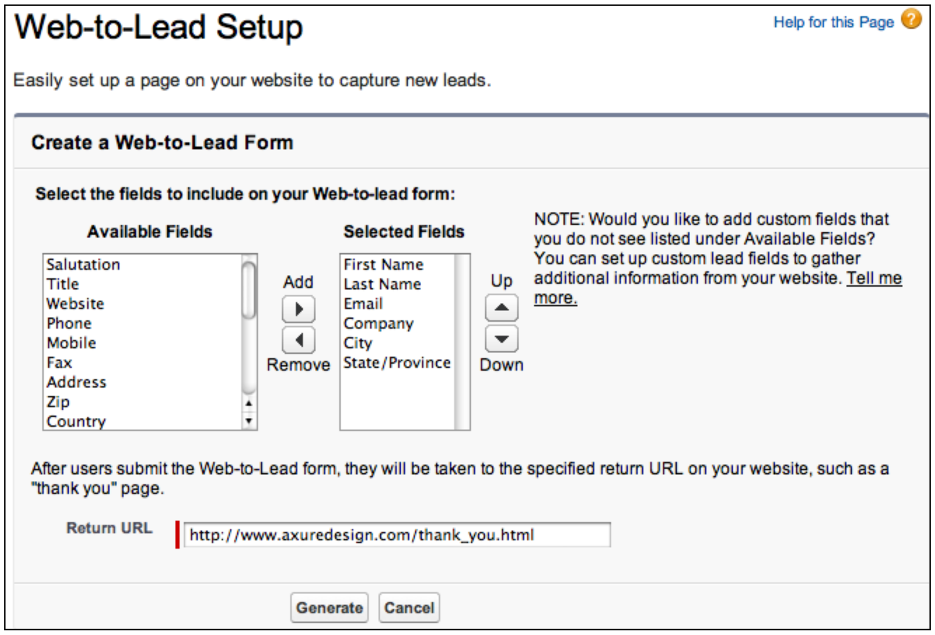
4. Click on **Edit** to modify the **Web-to-Lead** Settings if desired.



Web-to-Lead Enabled	✓
Default Lead Creator	John Krahenbuhl
Default Response Template	

5. Click on **Create Web-to-Lead Form**.
6. You can now add or remove additional fields from the form if you want by clicking either on the **Add** button or the **Remove** button. The order of the fields can be changed by clicking on the selected field and clicking on the up or down buttons.

7. Enter a **Return URL**. This is the page that will be displayed in the Inline Frame widget when the user has completed submitting the form. In our case you will create a basic "Thank You" HTML page and save it to a web server. The page will look like the following screenshot:



NOTE: Would you like to add custom fields that you do not see listed under Available Fields? You can set up custom lead fields to gather additional information from your website. Tell me more.

After users submit the Web-to-Lead form, they will be taken to the specified return URL on your website, such as a "thank you" page.

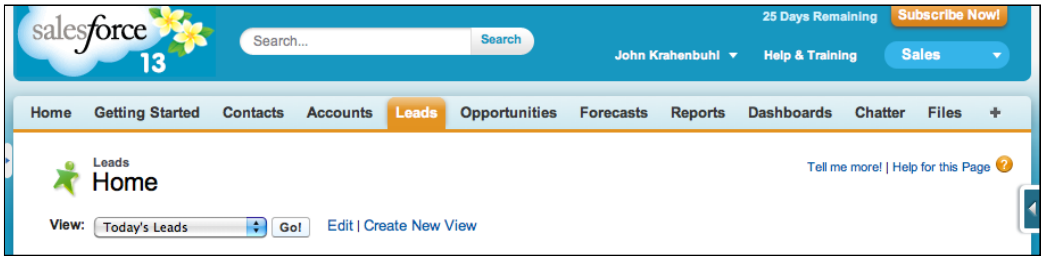
Return URL: http://www.axuredesign.com/thank_you.html

8. Click on **Generate**.
9. Copy and paste the sample HTML, and save to a new HTML file in your Axure prototype directory. The following is a sample HTML output from the Web-to-Lead Form generator:

```
<!--  ------------------------------------------------------------
----------  -->
<!--  NOTE: Please add the following <META> element to your page
<HEAD>.      -->
<!--  If necessary, please modify the charset parameter to specify
the        -->
<!--  character set of your HTML page.  -->
<!--  ------------------------------------------------------------
----------  -->

<META HTTP-EQUIV="Content-type" CONTENT="text/html;
charset=UTF-8">

<!--  ------------------------------------------------------------
----------  -->
```

```
<!--  NOTE: Please add the following <FORM> element to your page.
-->
<!--  ------------------------------------------------------------
----------  -->

<form action="https://www.salesforce.com/servlet/servlet.
WebToLead?encoding=UTF-8" method="POST">

<input type=hidden name="oid" value="00Di0000000YVW0">
<input type=hidden name="retURL"
value="http://www.axuredesign.com/thank_you.html">

<!--  ------------------------------------------------------------
----------  -->
<!--  NOTE: These fields are optional debugging elements. Please
uncomment    -->
<!--  these lines if you wish to test in debug mode.  -->
<!--  <input type="hidden" name="debug" value=1>  -->
<!--  <input type="hidden" name="debugEmail"  -->
<!--  value="john.krahenbuhl@gmail.com">  -->
<!--  ------------------------------------------------------------
----------  -->

<label for="first_name">First Name</label><input  id="first_name"
maxlength="40" name="first_name" size="20"
type="text" /><br>
<label for="last_name">Last Name</label><input
id="last_name" maxlength="80" name="last_name" size="20"
type="text" /><br>

<label for="email">Email</label><input  id="email"
maxlength="80" name="email" size="20" type="text" /><br>

<label for="company">Company</label><input  id="company"
maxlength="40" name="company" size="20" type="text" /><br>

<label for="city">City</label><input  id="city"
maxlength="40" name="city" size="20" type="text" /><br>

<label for="state">State/Province</label><input  id="state"
maxlength="20" name="state" size="20" type="text" /><br>

<input type="submit" name="submit">

</form>
```

10. To enable debug, uncomment either line of code just below the comment: `NOTE: These fields are optional debugging elements.`
11. Start Axure and under **Create New** select **RP File**.
12. While holding down the mouse button, drag the **Inline Frame** widget, and place it at coordinates (0,0) on the wireframe.
13. With the **Inline Frame** widget selected, to the top-right of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the Inline Frame widget. Enter `800` in the width field and `500` in the height field.
14. Double-click on the **Inline Frame** widget on the wireframe.
15. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.
16. Click on the **Hyperlink** field, and enter the filename of your HTML document created in step 10.
17. Right-click on the **Inline Frame** widget and click on **Toggle Border**.
18. Right-click on the **Inline Frame** widget, mouse over **Scrollbars**, and then click on **Never Show Scrollbars**.
19. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.
20. Click on **OK**.
21. Enter all of the field data and click on **submit**.
22. To verify that the data was entered into Salesforce, log in to your Salesforce account.
23. Click on **Leads**.
24. On the Leads Home page, **Go!**. The **Go!** button can be found next to **View:** dropdown (for example, in the screenshot, the **View:** dropdown contains the value **Today's Leads**).



25. You will see all of the leads generated from your prototype.

How it works...

In this recipe, you used the **Inline Frame** widget with the Default Target set as an external HTML file that contains the HTML code for your Salesforce form. You also uploaded a `thank_you` HTML file to a web server. When the prototype is run, the Inline Frame widget displays the external HTML file that contains our Salesforce form. After the user clicks on the **Submit** button, the form data is passed to Salesforce.

2
Enhanced Prototyping Recipes

In this chapter, you will cover several advanced Axure prototype recipes. You will learn the following:

- Adding external HTML and CSS files
- Incorporating inline frames
- Embedding external videos
- Including WordPress blog interactions
- Adding Google photo spheres
- Product visualization with flyout zoom
- Using Google Maps with Geolocation
- Leveraging social media logos – Facebook, Twitter, and Pinterest
- Adding app store badges – Apple iTunes and Google Play

Introduction

As your confidence and comfort level with Axure increases, you may want to add even more robust interactions to your prototypes. You can accomplish this by leveraging advanced concepts within Axure as well as utilizing external libraries and frameworks. In this chapter, you will first see how to use external HTML and CSS files to enhance an Axure prototype. You will explore recipes that detail how to interact with YouTube, WordPress blogs, and visualize products with flyout zoom. We'll also leverage social media logos and badges.

Adding external HTML and CSS files

Let's say that you started working on a design using HTML and CSS. You would like to include these in your Axure prototype. There are several ways to accomplish this task. In this recipe, you will explore a couple of those methods.

Getting ready

To complete this recipe, you will need to have external HTML and CSS files.

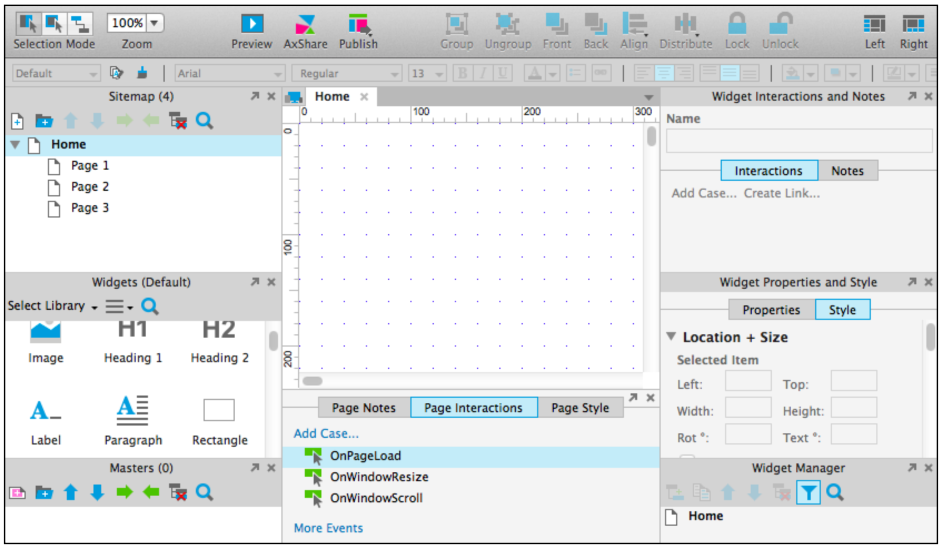
How to do it...

You will now link a prototype to the external HTML and CSS files.

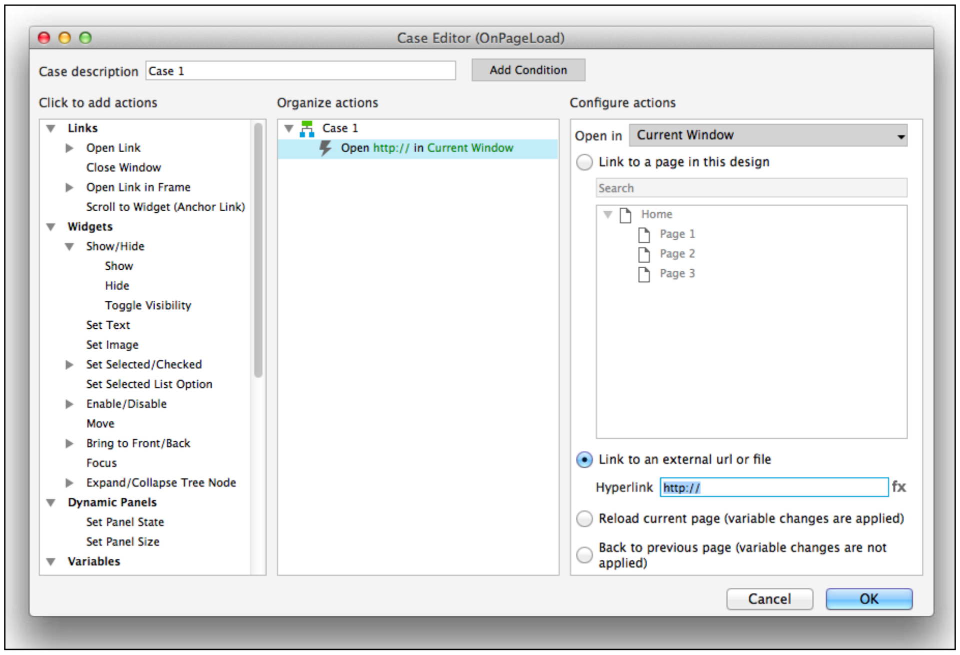
1. Start Axure and under **Create New** select **RP File**.

> If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. Double-click on **Home** in the **Sitemap** window to select the home page. The blank home page will be displayed in the wireframe pane.
3. Next, click on the **Page Interactions** tab in the **Page Properties** pane. Double-click on **OnPageLoad**.



4. Expand the **Links** button under **Click to add actions** on the **Case Editor (OnPageLoad)** pop up. Click on **Open Link**.
5. You will see, under **Organize actions**, that **Case 1** has been updated to have the action **Open http:// in Current Window**.
6. Under **Configure actions**, click on the radio button next to **Link to an external URL or file**.

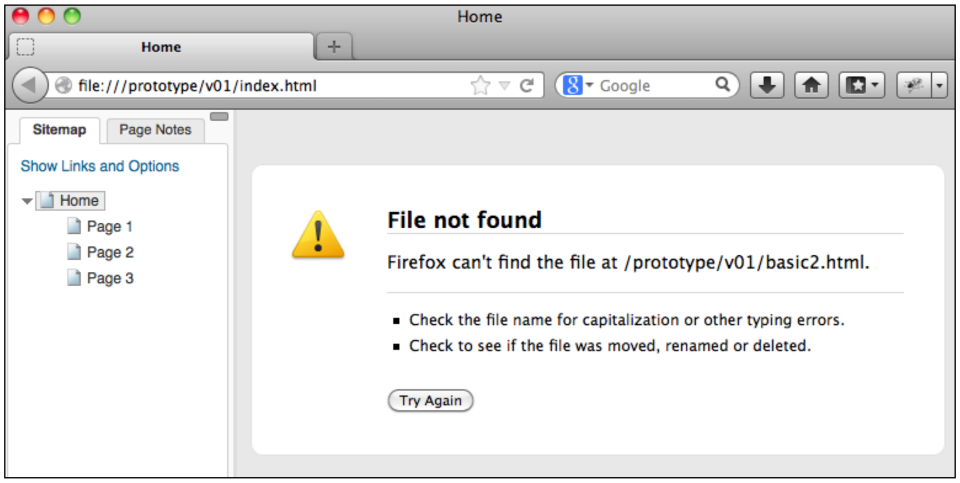


7. In the **Hyperlink** field, enter the path to the external `HTML` file.
8. Click on **OK**.
9. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and clicking on **Generate HTML Files...**.

How it works...

When you view the Axure prototype, the link that is shown in the site map for our given page links to our external HTML file. That external HTML file references the external CSS file.

Also, make sure to have your HTML and CSS files in the same folder as your prototype. If the prototype cannot find the HTML and CSS files when it loads, you will see an error message similar to the one in the following screenshot:



File not found

Firefox can't find the file at /prototype/v01/basic2.html.

- Check the file name for capitalization or other typing errors.
- Check to see if the file was moved, renamed or deleted.

There's more...

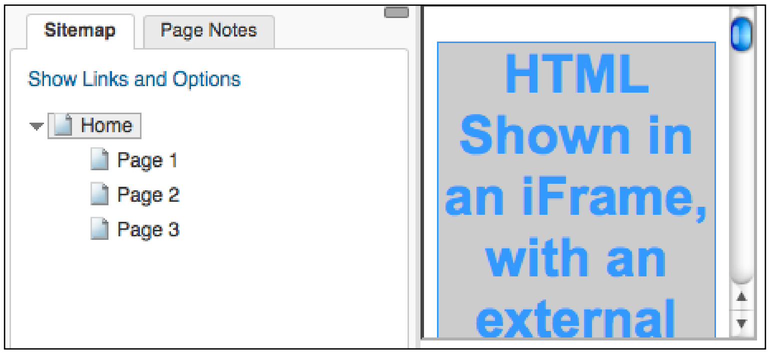
In this recipe, you learned how to link your prototype to external HTML and CSS files. There is another approach to accomplishing this task if you just want to link external HTML and CSS files. This approach involves modifying the Axure-generated HTML page. You would reference the external CSS file in the `head` tag and add your additional HTML in the `body` tag.

Incorporating inline frames

Leveraging the inline frame widget in your prototypes opens a world of possibilities. The inline frame widget allows you to link external HTML pages in your prototypes.

Getting ready

In this recipe, you will use an inline frame widget to open an external HTML page.



How to do it...

1. Start Axure and under **Create New** select **RP File**.

   If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. While holding down the mouse button, drag the **Inline Frame** widget and place it at these coordinates on the wireframe: (0,0).
3. Click on the **Inline Frame** widget on the wireframe. In the top right corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Inline Frame** widget. Adjust the width and height as needed.
4. Double-click on the **Inline Frame** widget on the wireframe.
5. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.
6. Click in the **Hyperlink** field and enter the filename of your HTML document.
7. Click on **OK**.
8. Click on the **Publish** button on the toolbar and then on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and clicking on **Generate HTML Files...**.
9. Note the directory where the prototype is saved.
10. Copy your HTML document referenced in step 6 and save it in the same directory as your prototype.
11. In your prototype directory, you will see a file labeled `start.html`. Axure created this file when the prototype was generated.
12. Open `start.html` in a browser to view the prototype.

How it works...

In this recipe, you use the inline frame widget and set the default target to link to an external file. When the prototype is loaded into the browser, the default HTML file is shown in the inline frame widget.

Embedding external videos

Embedding external video can make your prototype more vibrant and interactive. This recipe will show you how to embed YouTube and Vimeo videos into your prototype.

Getting ready

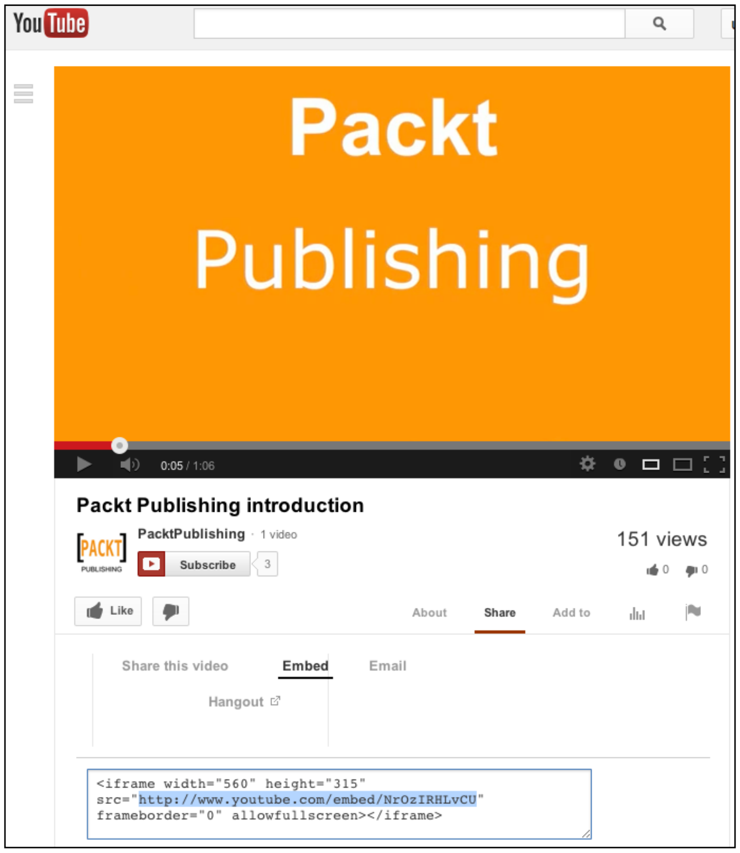
For this recipe, all you will need is a YouTube URL and a Vimeo URL for the videos you wish to link to. If you do not have a YouTube or Vimeo URL, don't worry; we will walk you though the process of getting what you need in this recipe.

How to do it...

In this recipe, you will first use an inline frame widget to open an external YouTube video. Next, you will view the YouTube video in your prototype. Then, you will add a second inline frame widget on another page in the prototype to open an external Vimeo video. Here's how you do it:

1. Open a browser and go to `http://www.youtube.com`.
2. Search for and select the video you would like to include in your prototype.
3. On the YouTube page for your selected video, click on the **Share** menu item under the video.
4. Next, click on the **Embed** menu item.
5. Copy the URL shown in the `embed` HTML tag provided. It will typically be shown on the second line starting with `src=`.

The `src` attribute is used to specify the location or URL of a video file. Please make sure to add `http:` at the beginning of the URL if it is not given in the `embed` HTML tag.



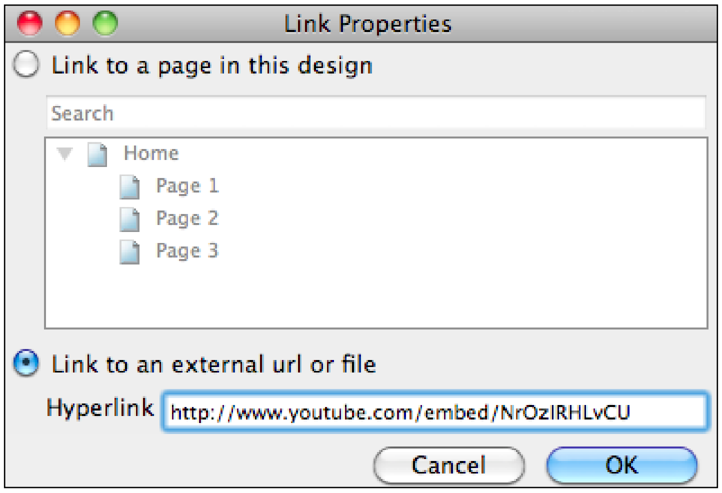
```
<iframe width="560" height="315"
src="http://www.youtube.com/embed/NrOzIRHLvCU"
frameborder="0" allowfullscreen></iframe>
```

6. Start Axure and click on **Create New RP Document**.

> If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

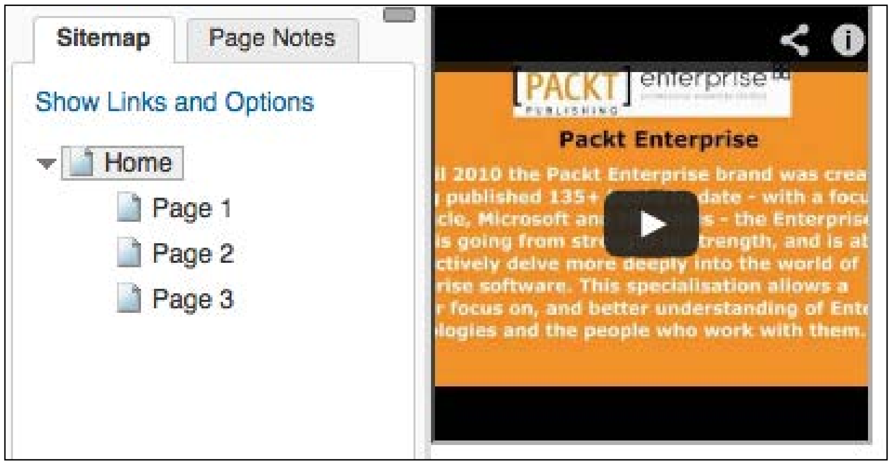
7. While holding down the mouse button, drag the **Inline Frame** widget and place it at the coordinates (0,0) on the wireframe.
8. Click on the **Inline Frame** widget on the wireframe.
9. Double-click on the **Inline Frame** widget on the wireframe.
10. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.

11. Click in the **Hyperlink** field and enter the URL for the YouTube video you copied in step 5.

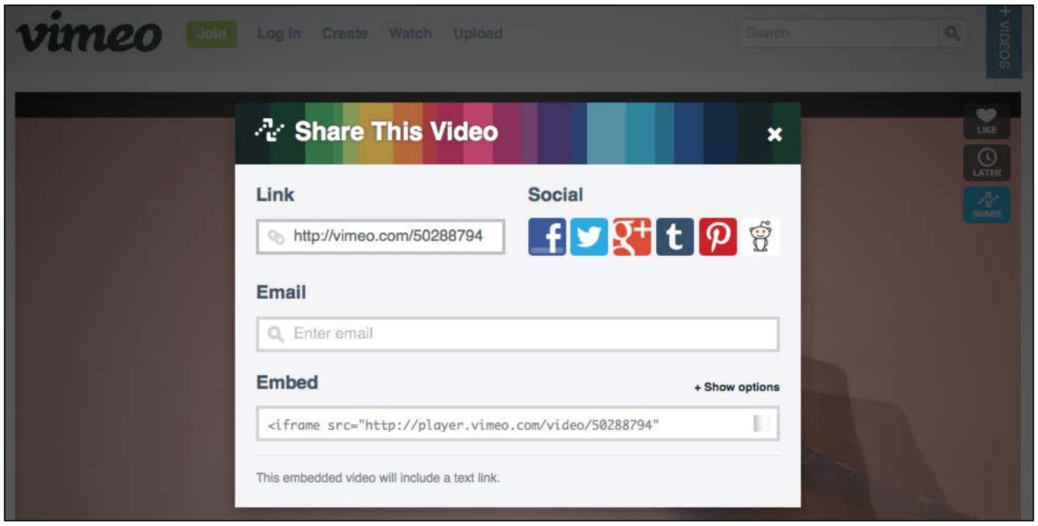

Link Properties
Link to a page in this design
Search
Home
Page 1
Page 2
Page 3
Link to an external url or file
Cancel
OK

Hyperlink: http://www.youtube.com/embed/NrOzIRHLvCU

12. Click on **OK**.
13. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu, and clicking on **Publish** and clicking on **Generate HTML Files...**. Now both videos will be viewable in the prototype.



14. Next, you will add an **Inline Frame** widget to **Page 1** and link to a Vimeo video.
15. Open a browser and go to `http://vimeo.com/`.
16. Search for and select the video you would like to include in your prototype.
17. On the Vimeo page for your selected video, click on the **Share** menu item to the right of the video.
18. An overlay titled **Share This Video** will appear.
19. Copy the URL shown in the `embed` tag provided. It will typically be shown in the line starting with `src=`.



http://vimeo.com/50288794

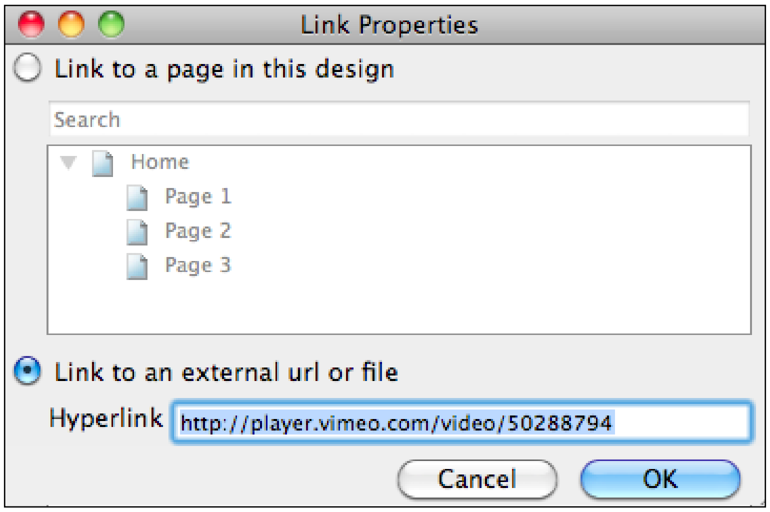
```
<iframe src="http://player.vimeo.com/video/50288794"
```

This embedded video will include a text link.

20. In Axure, click on **Page 1** in the **Sitemap** window.
21. While holding down the mouse button, drag the **Inline Frame** widget and place it at coordinates (0,0) on the wireframe.
22. Click on the **Inline Frame** widget on the wireframe.
23. Double-click on the **Inline Frame** widget on the wireframe.
24. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.

25. Click in the **Hyperlink** field and enter the URL for the Vimeo video you copied in step 5.

> Please include `http://` with your URL.


Link Properties

Link to a page in this design

Search

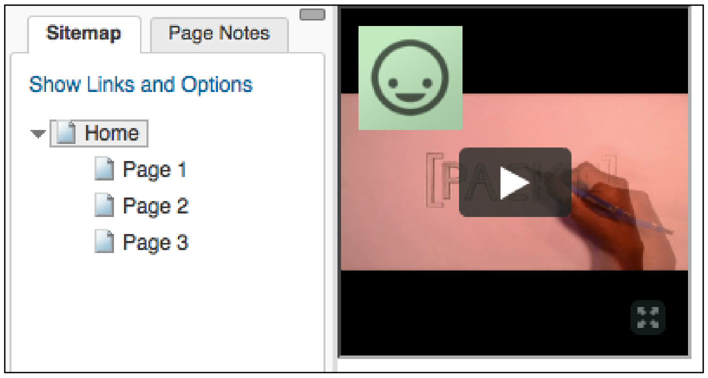
Home
Page 1
Page 2
Page 3

Link to an external url or file

Hyperlink http://player.vimeo.com/video/50288794

Cancel OK

26. Click on **OK**.
27. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish**, and clicking on **Generate HTML Files...**. Now both videos will be viewable in the prototype.
28. You can view the YouTube video by clicking on the **Home** page in the prototype. To view the Vimeo video, click on **Page 1** in the prototype.



How it works...

In this recipe, you use the inline frame widget and set the default target to link to the embedded URL you copied from the selected YouTube or Vimeo video. When the prototype is loaded into the browser, the YouTube video is shown in the inline frame widget on the **Home** page. Clicking on the **Page 1** page shows the Vimeo video. The YouTube and Vimeo links provide all of the media controls for the videos.

Including WordPress blog interactions

Adding live feeds to WordPress blogs is a nice way of providing realistic user interactions for flows that include blogs. This recipe will show you how to add a working WordPress blog feed to your prototypes.

How to do it...

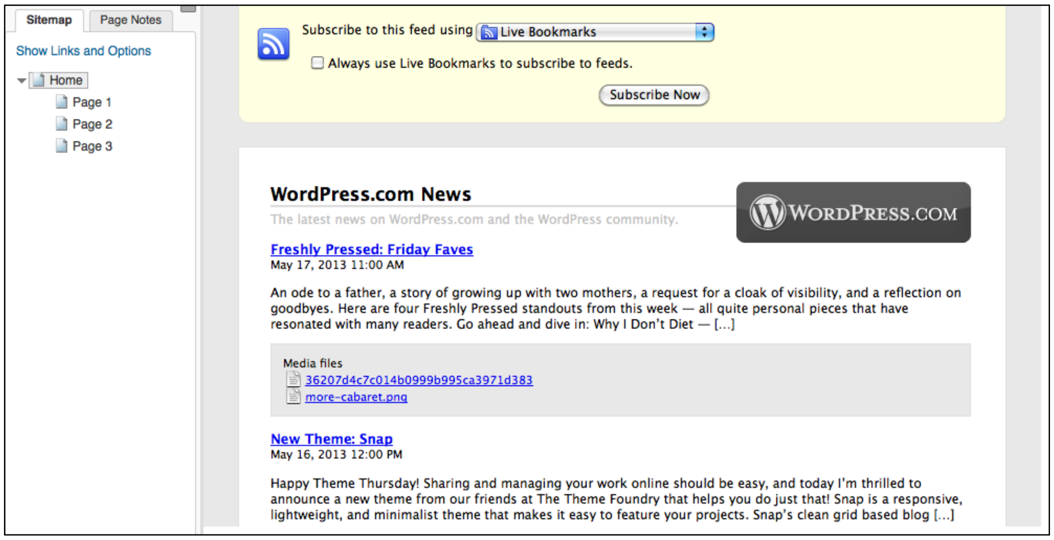
In this recipe, you will perform the following steps to link your Axure prototype to the WordPress feed:

1. Start Axure and under **Create New** select **RP File**.

   If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. While holding down the mouse button, drag the **Inline Frame** widget and place at this coordinates on the wireframe: (0,0).
3. With the **Inline Frame** widget selected, you will see two fields marked **w:** and **h:** in the top right-hand of the wireframe. These are for the width and height of the **Inline Frame** widget. Enter `800` in the width field and `500` in the height field.

4. Click on the **Inline Frame** widget on the wireframe.
5. Double-click on the **Inline Frame** widget on the wireframe.
6. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.
7. Click in the **Hyperlink** field and enter the link to the official WordPress feed at `http://en.blog.wordpress.com/`.
8. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button on the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and clicking on **Generate HTML Files...**.



WordPress.com News

The latest news on WordPress.com and the WordPress community.

Freshly Pressed: Friday Faves
May 17, 2013 11:00 AM

An ode to a father, a story of growing up with two mothers, a request for a cloak of visibility, and a reflection on goodbyes. Here are four Freshly Pressed standouts from this week — all quite personal pieces that have resonated with many readers. Go ahead and dive in: Why I Don't Diet — […]

Media files
36207d4c7c014b0999b995ca3971d383
more-cabaret.png

New Theme: Snap
May 16, 2013 12:00 PM

Happy Theme Thursday! Sharing and managing your work online should be easy, and today I'm thrilled to announce a new theme from our friends at The Theme Foundry that helps you do just that! Snap is a responsive, lightweight, and minimalist theme that makes it easy to feature your projects. Snap's clean grid based blog […]

How it works...

The key to this recipe is using the inline frame widget with a default target set to the WordPress feed link at `http://en.blog.wordpress.com/`. When the prototype is running, the inline frame widget displays the link to the WordPress feed.

There's more...

You can learn more about WordPress feeds at `http://en.support.wordpress.com/feeds/`. Another popular method of using feeds on websites today is to use **Google FeedBurner**. More information about Google FeedBurner can be found at `https://support.google.com/feedburner/answer/79408`.

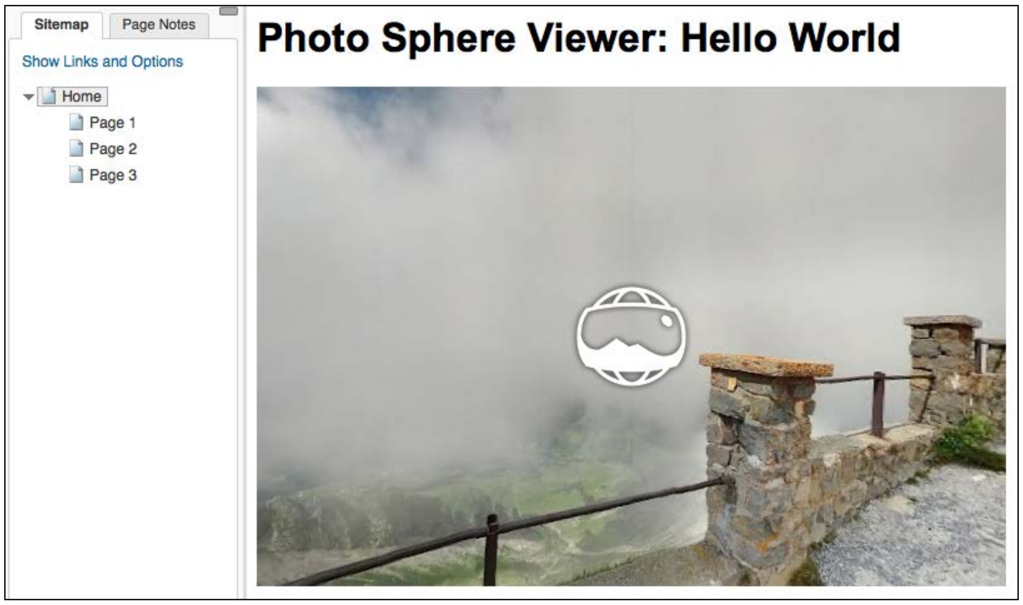
Adding Google photo spheres

An update to Google's mobile operating system Jelly Bean (that is, Android 4.2) provides the ability to capture photo spheres. Photo spheres are 360-degree panoramic images. Google recently shared with developers a viewer that allows everyone to enjoy photo spheres on their website. This recipe will show you how to add Google photo spheres to your prototypes. See `https://developers.google.com/photo-sphere/web/` for more information.

> You may have to register as an Android developer to access the preceding URL. To register as an Android developer, visit `https://support.google.com/googleplay/android-developer/answer/113468?hl=en`.

Getting ready

To complete this recipe, you will need a photo sphere image. You may use the Hello World example provided by Google. If you have a photo sphere image, we'll show you where to reference it in your external HTML.



How to do it...

In this recipe, you will use an inline frame widget to display a Google photo sphere image in an Axure prototype. Perform the following steps:

1. If you have a photo sphere image, copy the file to your Axure prototype directory.
2. Using your favorite text editor, create a blank `.html` file.
3. Place the following HTML code into the blank `.html` file. Change the reference in `imageURL` to point to the local file you copied to your Axure prototype directory in step 1.

```
<html>
  <head>
    <title>Photo Sphere Viewer: Hello World</title>
    <script type="text/javascript"
      src="https://apis.google.com/js/plusone.js"></script>
    <style> body {font-family: Arial,Verdana}</style>
  </head>
  <body>
    <h1>Photo Sphere Viewer: Hello World</h1>
    <g:panoembed imageURL="https://lh5.googleusercontent.com/-
  kr97Eucg6sM/UKGEuvo_eBI/AAAAAAAAi0s/adq8uqyhm_k/
    photo.jpg" fullsize="4096,2048" croppedsize="4096,1380"
      offset="0,480" displaysize="600,400"/>
    <script>
      gapi.panoembed.go();
    </script>
  </body>
</html>
```

4. Start Axure and under **Create New** select **RP File**.

If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

5. While holding down the mouse button, drag the **Inline Frame** widget and place it at coordinates (0,0) on the wireframe.
6. With the **Inline Frame** widget selected, you will see two fields marked **w:** and **h:** in the top right-hand of the wireframe. These are for the width and height of the **Inline Frame** widget. Enter `800` in the width field and `500` in the height field.

7. Click on the **Inline Frame** widget on the wireframe.
8. Double-click on the **Inline Frame** widget on the wireframe.
9. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.
10. Click in the **Hyperlink** field and enter the filename of the HTML document you created in step 2.
11. To save a copy of the prototype, click on the **Publish** button in the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and clicking on **Generate HTML Files...**.
12. Note the directory where the prototype is saved.
13. Copy the HTML document you created in step 3 and save it in the same directory as your prototype.
14. In your prototype directory, open `start.html` in a browser to view the prototype.

How it works...

In this recipe, you used the inline frame widget with an external HTML file set as the default target. The external HTML file contains the HTML code that references the JavaScript file for the Google photo sphere viewer. You place the external HTML file and photo sphere file in the same folder as the root of the prototype output. When the prototype is run, the inline frame widget runs the external HTML file that calls the photo sphere JavaScript that in turn renders the photo sphere image. When the user holds down the left mouse button and moves the cursor, the photo sphere image rotates in the viewer shown in the inline frame widget.

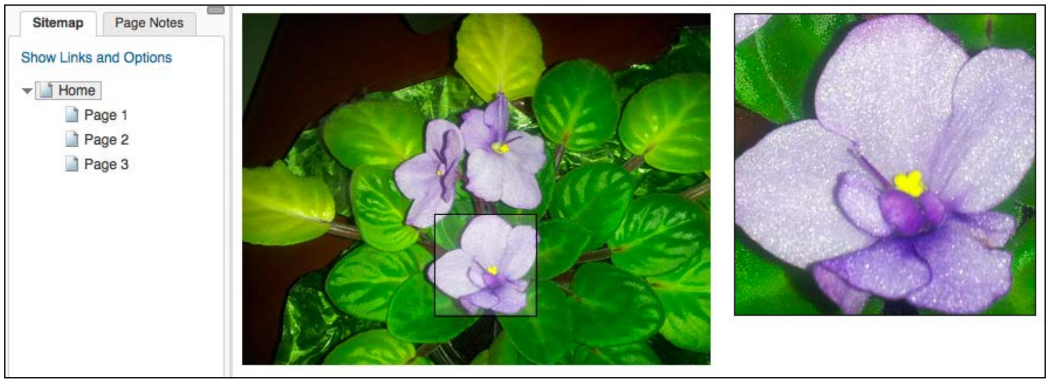
Product visualization with flyout zoom

Product visualization with flyout zoom is popular on many websites. Here is how it works: a product image is displayed in the browser window, and when the user hovers their mouse pointer over a portion of the product image, the portion of the image under the mouse is magnified in a flyout window.

There are several commercial solutions available (for example, Adobe Scene 7) that provide this effect. There also are free open source solutions as well. In this recipe, you will use an open source flyout zoom solution.

Getting ready

To complete this recipe, you will need the **MojoZoom** CSS and JavaScript files available at `http://www.nihilogic.dk/labs/mojozoom/`, an image saved as two different files, and Axure. For this recipe, you are using a resolution of 400 x 300 pixels for the small image file and a resolution of 1200 x 900 pixels for the large image file. You can learn more about MojoZoom at `http://www.nihilogic.dk/labs/mojozoom/`.



How to do it...

In this recipe, you are going to explore using **MojoZoom Javascript Image Zoom**, which is free and licensed under the MPL license.

1. Start Axure and under **Create New** select **RP File**.

> If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. While holding down the mouse button, drag the **Inline Frame** widget and place it at coordinates (0,0) on the wireframe.
3. With the **Inline Frame** widget selected, you will see two fields marked **w:** and **h:** in the top right-hand of the wireframe. These are for the width and height of the **Inline Frame** widget. Enter `800` in the width field and `500` in the height field.
4. Click on the **Inline Frame** widget on the wireframe.
5. Double-click on the **Inline Frame** widget on the wireframe.
6. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.
7. Click in the **Hyperlink** field and enter `mojozoom.html`.

8. To save a copy of the prototype, click on the **Publish** button in the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and clicking on **Generate HTML Files...**.
9. Note the directory where the prototype is saved.
10. Download the MojoZoom ZIP file from `http://www.nihilogic.dk/labs/mojozoom/`. Unzip the `MojoZoom.zip` file and copy the `mojozoom.css` and `mojozoom.js` files to your prototype directory.
11. Create two image files of the same JPG image. A small image file with a resolution of 400 x 300 pixels and a large image file with a resolution of 1200 x 900 pixels. Name the files `small_image1_400x300.jpg` and `large_image1_1200x900.jpg`, respectively. Copy your small and large image files to your prototype directory.
12. Using your favorite text editor, create an HTML file named `mojozoom.html` with the following HTML code and save in your prototype directory:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Axure RP Prototyping Cookbook
      Zoom Demo</title>
    <script src="mojozoom.js"
      type="text/javascript" ></script>
    <link rel="stylesheet" href="mojozoom.css"
      type="text/css"></link>
  </head>

  <body>

    <img id="myimage" style="padding:0px;border:0px;"
      data-zoomsrc="large_image1_1200x900.jpg"
        src="small_image1_400x300.jpg">
    <div style="position: absolute; left: 0px; top: 0px;
      width: 400px; height: 299px; overflow: hidden;
        display: none;">
    <div class="mojozoom_marker" style="width: 90px;
      height: 90px; left: 308px; top: 152px;">
    <div class="mojozoom_fill"></div>
    <div class="mojozoom_border"></div>
    </div>
    </div>

  </body>

</html>
```

13. In your prototype directory, open `start.html` in a browser to view the prototype.

How it works...

In this recipe, you used the inline frame widget with an external HTML file set as the default target. The external HTML file contains the HTML code that references the CSS and JavaScript files for the flyout zoom solution. You place the external HTML file, image files, and flyout zoom CSS and JavaScript files in the same folder as the root of the prototype output. When the prototype is run, the inline frame widget runs the external HTML file that shows the small image, and the flyout zoom JavaScript renders the zoom window when the mouse pointer is over the image shown in the inline frame widget.

There's more...

There are many open source flyout zoom solutions available. This recipe has shown you an approach to integrate flyout zoom into your prototype using the MojoZoom solution. Feel free to explore other open source solutions as well.

Using Google Maps with Geolocation

You can also use Google Maps with Geolocation to create dynamic prototypes. You will leverage the latest Google Maps APIs to make our prototypes come to life. You will use the Google Maps Geolocation example found at `https://developers.google.com/maps/documentation/javascript/examples/map-geolocation`.

How to do it...

In this recipe, you will use an inline frame widget to utilize Google Maps Geolocation in an Axure prototype. Perform the following steps:

1. Using your favorite text editor, create a `.html` file containing the following `HTML` code and place it in your Axure prototype directory:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Geolocation</title>
    <meta name="viewport" content="initial-scale=1.0,
      user-scalable=no">
    <meta charset="utf-8">
    <link href="http://code.google.com/
      apis/maps/documentation/javascript/examples/
        default.css" rel="stylesheet">
    <!--
    Include the maps javascript with sensor=true because this code
is using a
```

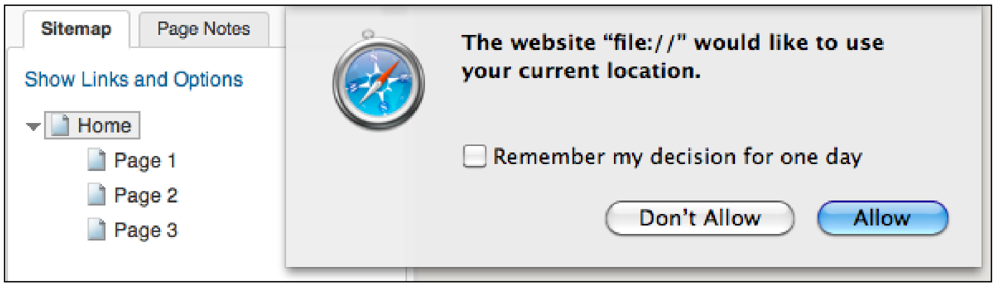
```
    sensor (a GPS locator) to determine the user's location.
    See: https://developers.google.com/apis/maps/documentation/
javascript/basics#SpecifyingSensor
    -->
    <script src="https://maps.googleapis.com/maps
      /api/js?v=3.exp&sensor=true"></script>

    <script>
var map;

function initialize() {
  var mapOptions = {
    zoom: 6,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
  map = new google.maps.Map(document.getElementById
    ('map-canvas'),
      mapOptions);

  // Try HTML5 geolocation
  if(navigator.geolocation) {
    navigator.geolocation.getCurrentPosition
      (function(position) {
      var pos = new google.maps.LatLng
        (position.coords.latitude,
          position.coords.longitude);

      var infowindow = new google.maps.InfoWindow({
        map: map,
        position: pos,
        content: 'Location found using HTML5.'
      });

      map.setCenter(pos);
    }, function() {
      handleNoGeolocation(true);
    });
  } else {
    // Browser doesn't support Geolocation
      handleNoGeolocation(false);
  }
}

function handleNoGeolocation(errorFlag) {
  if (errorFlag) {
          var content = 'Error: The Geolocation service failed.';
        } else {
          var content = 'Error: Your browser doesn\
            't support geolocation.';
        }

        var options = {
          map: map,position:
            new google.maps.LatLng(60, 105),
          content: content
        };

        var infowindow = new google.maps.InfoWindow(options);
        map.setCenter(options.position);
      }

      google.maps.event.addDomListener
        (window, 'load', initialize);

          </script>
        </head>
        <body>
          <div id="map-canvas"></div>
        </body>
      </html>
```

2. Start Axure and under **Create New** select **RP File**.

> If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

3. While holding down the mouse button, drag the **Inline Frame** widget and place it at coordinates (0,0) on the wireframe.
4. With the **Inline Frame** widget selected, you will see two fields marked **w:** and **h:** in the top right-hand of the wireframe. These are for the width and height of the **Inline Frame** widget. Enter `800` in the width field and `500` in the height field.
5. Click on the **Inline Frame** widget on the wireframe.
6. Double-click on the **Inline Frame** widget on the wireframe.
7. In the **Link Properties** pop up click on the radio button next to **Link to an external URL or file**.

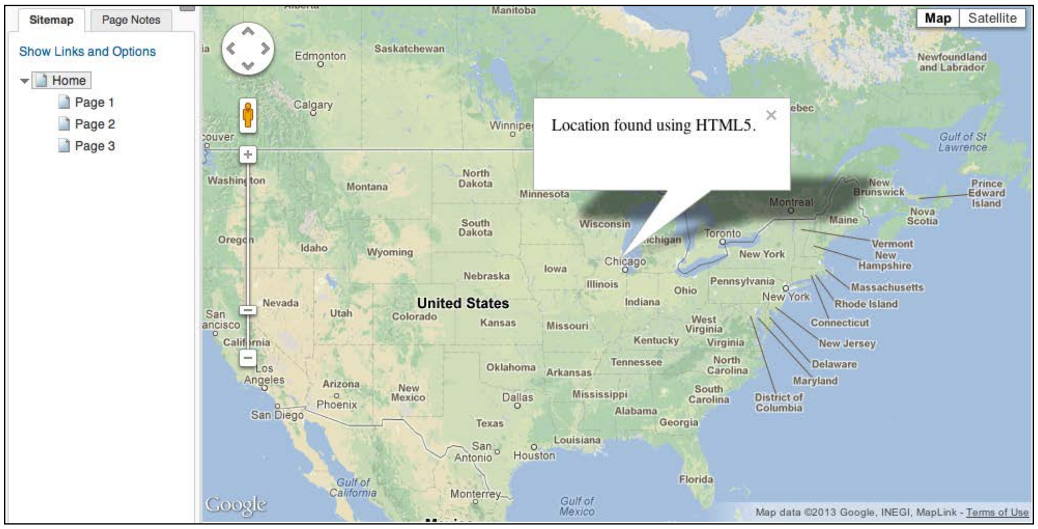
8. Click in the **Hyperlink** field and enter the filename of your HTML document created in step 1.
9. To save a copy of the prototype, click on the **Publish** button in the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and click on **Publish**, then click on **Generate HTML Files...**.
10. Note the directory where the prototype is saved.
11. Copy your HTML document created in step 1 and save it in the same directory as your prototype.
12. In your prototype directory, open `start.html` in a browser to view the prototype.
13. Click on **Allow** to enable the browser to share your current location with Google to update the Google Maps.



The website "file://" would like to use your current location.

Remember my decision for one day

14. The map will now be displayed and you can move and zoom in on the map.



Location found using HTML5.

How it works...

In this recipe, you used the inline frame widget with an external HTML file set as the default target. The external HTML file contains the HTML code that references the Google Maps API. You placed the external HTML file in the same folder as the root of the prototype output. When the prototype is run, the inline frame widget runs the external HTML file that requests location data from the browser. After the user allows the browser to share this data, the Google Maps location is shown in the inline frame widget.

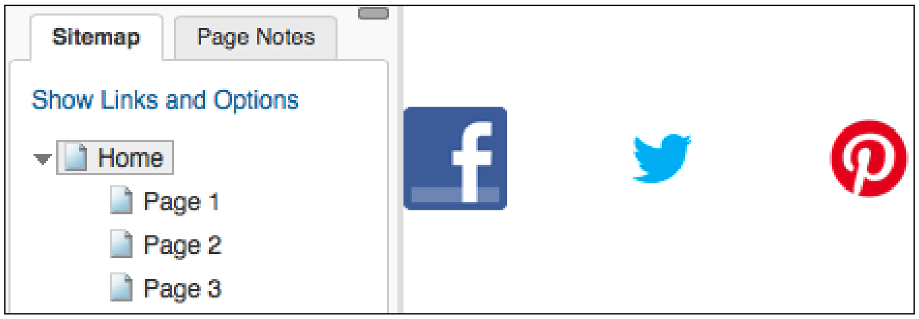
Leveraging social media logos – Facebook, Twitter, and Pinterest

Allowing clients to interact with familiar social media properties can help to make your prototypes feel more realistic. This recipe will show you how to open external Facebook, Twitter, and Pinterest pages using social media logos.

Getting ready

For this recipe, you will need the social media logos and the URL links you would like to reference. Logos can be found from the following links:

- Facebook: `www.facebook.com/brandpermissions/logos.php`
- Twitter: `https://twitter.com/logo`
- Pinterest: `http://business.pinterest.com/logos-and-marketing-guidelines/`



How to do it...

For this recipe, you will create three image widgets linked to Facebook, Twitter, and Pinterest.

1. Start Axure and under **Create New** select **RP File**.

   If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. While holding down the mouse button, drag the **Image** widget and place it at the coordinates (0,50) on the wireframe.
3. Double-click on the **Image** widget on the wireframe and click on the file with the Facebook icon.

   Do not autosize the image.

4. With the **Image** widget selected, perform the following steps:
   1. In the **Widget Interactions and Notes** pane, click in the **Image Name** text field and type `FacebookLogo`.
   2. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and double-click on the **OnClick** interaction.
   3. In the **Case Editor (OnClick)** pop up, in **Case description**, change the case description to `FacebookLogoClicked`.
   4. In **Click to add actions**, click on **Open Link**.
   5. In **Organize actions**, you will see the interaction description change to **Open http:// in Current Window**.
   6. In **Configure actions**, click on the radio button next to **Link to an external URL or file** and type `https://www.facebook.com/PacktPub` in the **Hyperlink** field.
   7. You will see the interaction description under **Organize Actions** change to **Open https://www.facebook.com/PacktPub in Parent Window**.
   8. Click on **OK**.
5. While holding down the mouse button, drag the **Image** widget and place it at the coordinates (100,50) on the wireframe.
6. Double-click on the **Image** widget on the wireframe and click on the file with the Twitter icon.

   Do not autosize the image.

7. With the **Image** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click in the **Label** text field and type `TwitterLogo`.
    2. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and double-click on the **OnClick** interaction.
    3. In the **Case Editor (OnClick)** pop up, in **Case Description**, change the case description to `TwitterLogoClicked`.
    4. Under **Click to add actions**, click on **Open Link in New Window/Tab**.
    5. Under **Organize actions**, you will see the interaction description change to **Open Link in New Window/Tab**.
    6. Under **Configure actions**, click on the radio button next to **Link to an external URL or file** and type `https://twitter.com/packtpub` in the **Hyperlink** field.
    7. You will see the interaction description under **Organize actions** changes to **Open https://twitter.com/packtpub in Parent Window**.
    8. Click on **OK**.
8. While holding down the mouse button, drag the **Image** widget and place at the coordinates (200,50) on the wireframe.
9. Double-click on the **Image** widget on the wireframe and click on the file with the Pinterest icon.

Do not autosize the image.

10. With the **Image** widget selected, perform the following steps:
    1. In the **Widget Properties and Notes** pane, click in the **Label** text field and type `PinterestLogo`.
    2. In the **Widget Properties and Notes** pane, click on the **Interactions** tab and double-click on the **OnClick** interaction.
    3. In the **Case Editor (OnClick)** pop up, in **Case description**, change the case description to `PinterestLogoClicked`.
    4. In **Click to add actions**, click on **Open Link in New Window/Tab**.
    5. In **Organize actions** you will see the interaction description change to **Open Link in New Window/Tab**.
    6. In **Configure actions**, click on the radio button next to **Link to an external URL or file** and type `http://pinterest.com/search/boards/?q=Axure+RP+Prototyping+Cookbook` in the **Hyperlink** field.

7. You will see the interaction description under **Organize actions** change to **Open http://pinterest.com/search/boards/?q=Axure+RP+Prototyping+Cookbook in Parent Window**.
8. Click on **OK**.

11. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and click on **Publish**, then click on **Generate HTML Files...**.

How it works...

The key to this recipe is using the image widget with the `OnClick` interaction to open a link in a new window/tab. You created an image widget for each of the social media logos and created an `OnClick` interaction to open a link in a new window/tab.

There's more...

For the **Case Editor** window' s **Click to add actions** you could have selected from the actions **Open Link in Popup Window**. **Open link in Parent Window** and **Open Link in Current Window** do not work properly when using the Axure-generated **Sitemap** and **Page Notes** pane of the prototype.

Another approach would be to use a button, badge, or social plugin. To use a social plugin, you would use an inline frame widget and reference the specific code for the social plugin. Buttons, badges, or social plugins can be found at the following links:

- Facebook: `http://www.facebook.com/badges`
- Twitter: `https://twitter.com/about/resources/buttons`
- Pinterest: `http://business.pinterest.com/pin-it-button/`

Adding app store badges – Apple iTunes and Google Play

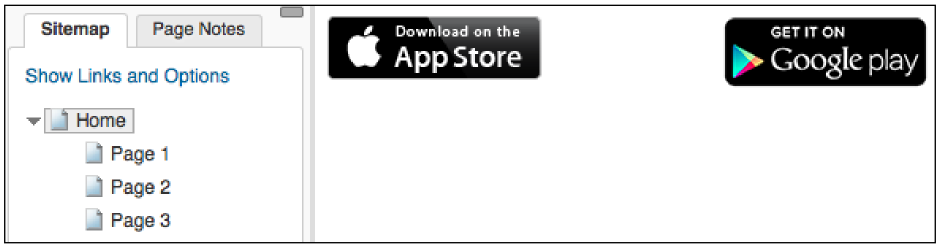
Adding direct links to app stores is a neat way to make your prototypes appear more realistic. This recipe will show you how to add working Apple iTunes and Google Play badges to your prototypes.

Getting ready

For this recipe, you will need an HTML editor and will need to generate the code for the Apple iTunes and Google Play badges. The guidelines for linking to iTunes can be found at `http://www.apple.com/itunes/link/` and the **Link Maker** tool can be found at `http://linkmaker.itunes.apple.com/us/`.

The guidelines for linking and creating badges can be found at the following links:

- Apple iTunes: `http://www.apple.com/itunes/link/`
- Apple Link Maker tool can be found at `http://linkmaker.itunes.apple.com/us/`
- Google Play: `http://developer.android.com/distribute/googleplay/promote/linking.html`
- Google Play badge generator tool can be found at `http://developer.android.com/distribute/googleplay/promote/badges.html`



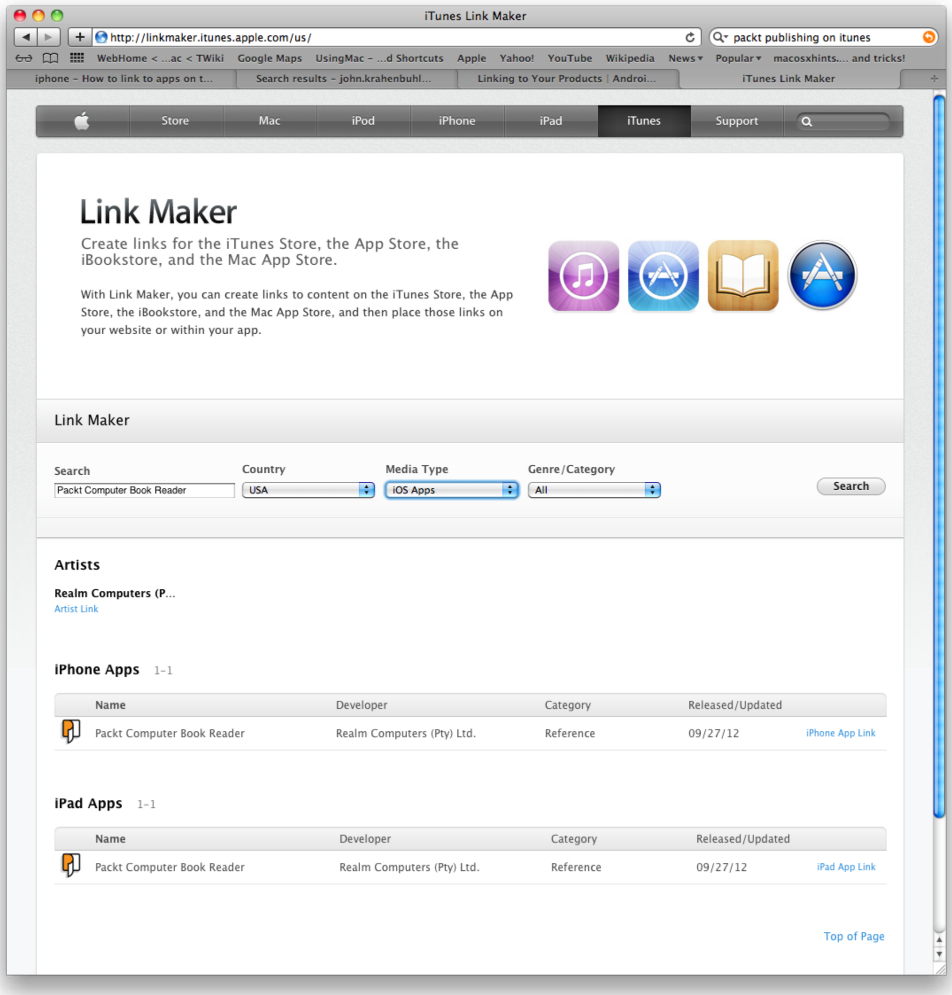
How to do it...

Perform the following steps:

1. Create and save an empty HTML document called `itunes_badge.html` with the following content:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>


  </body>
</html>
```

2. Point your web browser to the Apple Link Maker tool.
3. Enter your search term in the **Search** field, select your **Country**, **Media Type**, and **Genre/Category**, and left-click on **Search**.
4. Select the appropriate result and click on the link.


iTunes Link Maker
http://linkmaker.itunes.apple.com/us/
packt publishing on itunes
Store
Mac
iPod
iPhone
iPad
iTunes
Support

Link Maker

Create links for the iTunes Store, the App Store, the iBookstore, and the Mac App Store.

With Link Maker, you can create links to content on the iTunes Store, the App Store, the iBookstore, and the Mac App Store, and then place those links on your website or within your app.

Link Maker

Search: Packt Computer Book Reader
Country: USA
Media Type: iOS Apps
Genre/Category: All
Search

Artists

Realm Computers (P...
Artist Link

iPhone Apps 1-1

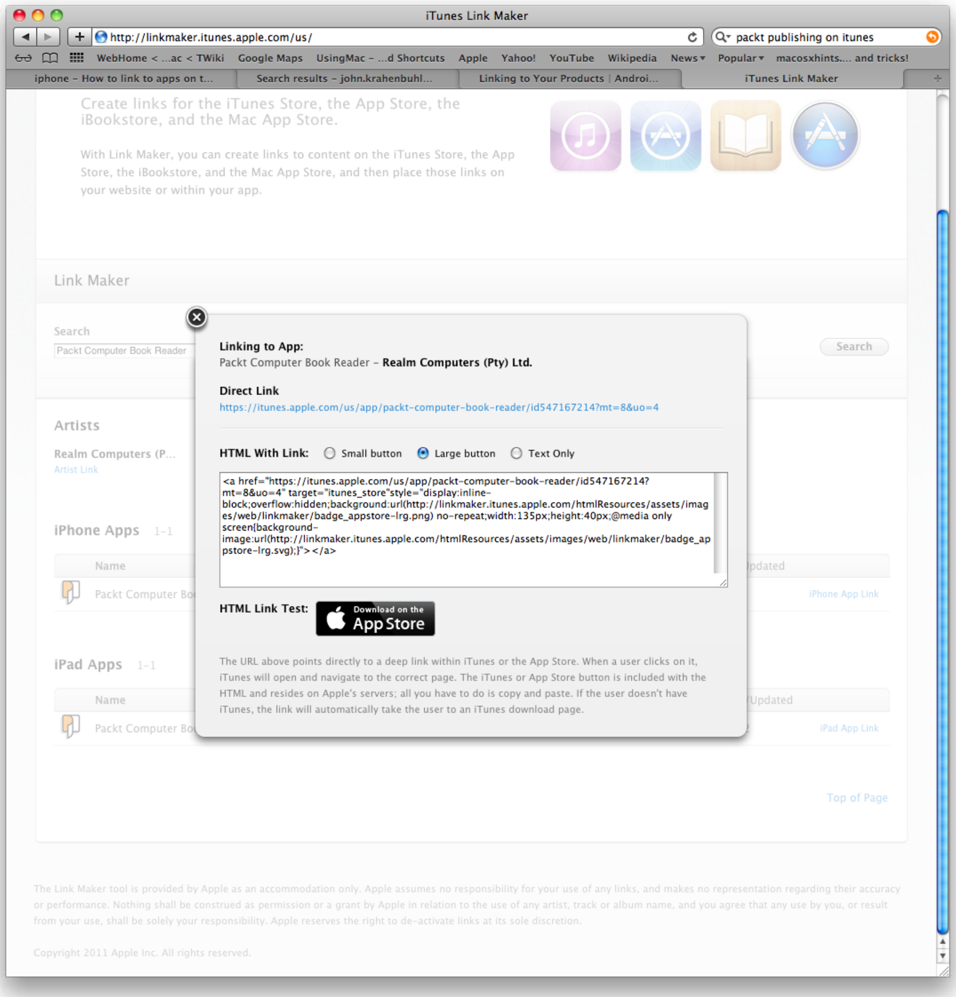
Name	Developer	Category	Released/Updated	
Packt Computer Book Reader	Realm Computers (Pty) Ltd.	Reference	09/27/12	iPhone App Link

iPad Apps 1-1

Name	Developer	Category	Released/Updated	
Packt Computer Book Reader	Realm Computers (Pty) Ltd.	Reference	09/27/12	iPad App Link

Top of Page

5. In the pop-up window, next to **HTML with Link:**, click on the radio button that corresponds with either **Small button** or **Large button**.
6. Copy the HTML provided by the Link Maker tool.


iTunes Link Maker
http://linkmaker.itunes.apple.com/us/
packt publishing on itunes
Create links for the iTunes Store, the App Store, the iBookstore, and the Mac App Store.
Link Maker
Search
Artists
iPhone Apps
iPad Apps
Top of Page

Linking to App:
Packt Computer Book Reader – **Realm Computers (Pty) Ltd.**

Direct Link
https://itunes.apple.com/us/app/packt-computer-book-reader/id547167214?mt=8&uo=4

HTML With Link: ( ) Small button [✓ Large button] ( ) Text Only

```
<a href="https://itunes.apple.com/us/app/packt-computer-book-reader/id547167214?
mt=8&uo=4" target="itunes_store"style="display:inline-
block;overflow:hidden;background:url(http://linkmaker.itunes.apple.com/htmlResources/assets/imag
es/web/linkmaker/badge_appstore-lrg.png) no-repeat;width:135px;height:40px;@media only
screen{background-
image:url(http://linkmaker.itunes.apple.com/htmlResources/assets/images/web/linkmaker/badge_ap
pstore-lrg.svg);}"></a>
```

HTML Link Test: Download on the App Store

The URL above points directly to a deep link within iTunes or the App Store. When a user clicks on it, iTunes will open and navigate to the correct page. The iTunes or App Store button is included with the HTML and resides on Apple's servers; all you have to do is copy and paste. If the user doesn't have iTunes, the link will automatically take the user to an iTunes download page.

7. Insert the HTML from step 1 between the `<body>` and `</body>` tags in the HTML document you created in step 1 and save the HTML document.
8. Start Axure and under **Create New** select **RP File**.

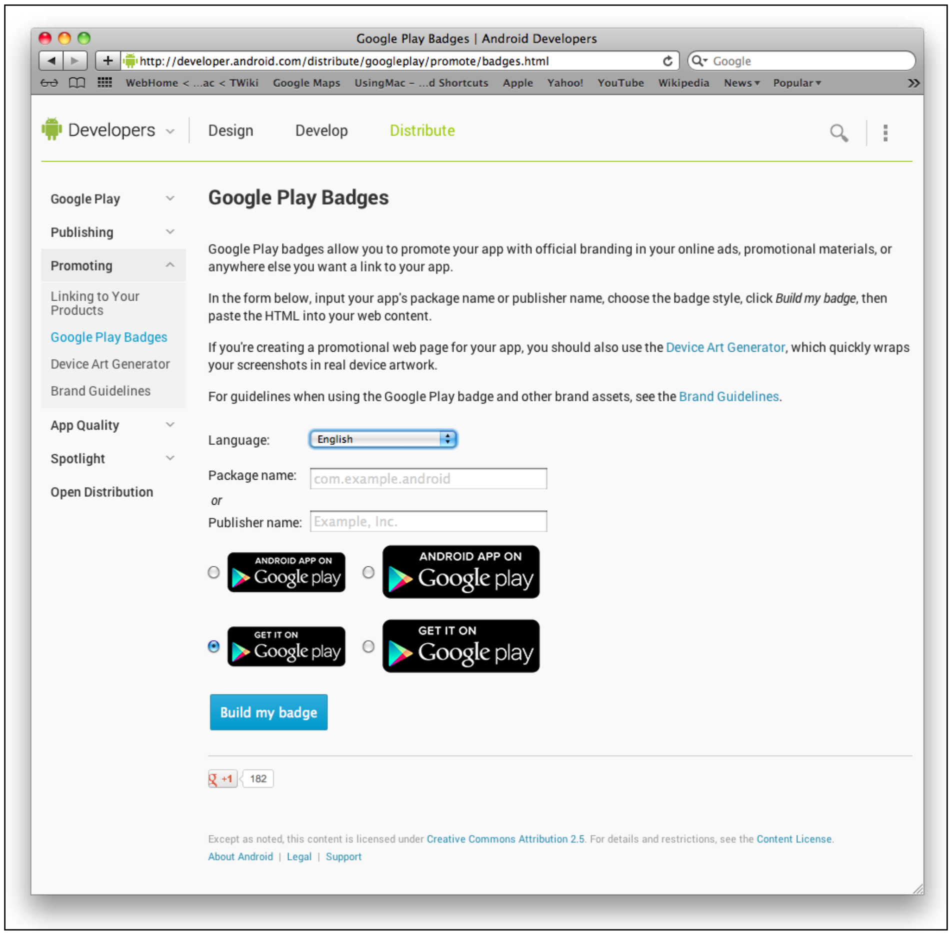
> If you already have Axure open, click on **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

9. While holding down the mouse button, drag the **Inline Frame** widget and place it at the coordinates (0,0) on the wireframe.
10. Right-click on the **Inline Frame** widget and click on **Toggle Border**.
11. Right-click on the **Inline Frame** widget, click on **Scrollbars**, and click on **Never Show Scrollbars**.
12. Double-click on the **inline frame** on the wireframe.
13. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.
14. Left-click in the **Hyperlink** field and type **itunes_badge.html**.
15. Click on **OK**.
16. Create and save an empty HTML document called `google_play_badge.html` with the following content:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>

  </body>
</html>
```

17. Point your web browser to the Google Play badge generator tool.

18. Select your language and enter the publisher name for your app.



Google Play Badges | Android Developers

http://developer.android.com/distribute/googleplay/promote/badges.html

WebHome < ...ac < TWiki Google Maps UsingMac – ...d Shortcuts Apple Yahoo! YouTube Wikipedia News Popular

Developers Design Develop Distribute

Google Play
Publishing
Promoting
Linking to Your Products
Google Play Badges
Device Art Generator
Brand Guidelines
App Quality
Spotlight
Open Distribution

Google Play Badges

Google Play badges allow you to promote your app with official branding in your online ads, promotional materials, or anywhere else you want a link to your app.

In the form below, input your app's package name or publisher name, choose the badge style, click *Build my badge*, then paste the HTML into your web content.

If you're creating a promotional web page for your app, you should also use the Device Art Generator, which quickly wraps your screenshots in real device artwork.

For guidelines when using the Google Play badge and other brand assets, see the Brand Guidelines.

Language: English

Package name: com.example.android

or

Publisher name: Example, Inc.

Build my badge

+1 182

Except as noted, this content is licensed under Creative Commons Attribution 2.5. For details and restrictions, see the Content License.
About Android | Legal | Support

19. Click the radio button next to the badge you would like to generate.
20. Click on **Build my badge**.
21. Copy and paste the generated HTML from step 20 between the `<body>` and `</body>` tags of the HTML document you created in step 16 and save the HTML document.
22. While holding down the left mouse button, drag the **Inline Frame** widget and place at coordinates (250,0) on the wireframe.
23. Right-click on the **Inline Frame** and click on **Toggle Border**.
24. Right-click on the **Inline Frame** widget, mouse over **Scrollbars**, and click on **Never Show Scrollbars**.

25. Double-click on **Inline Frame** in the wireframe.
26. In the **Link Properties** pop up, left-click on the radio button next to **Link to an external URL or file**.
27. Left-click in the **Hyperlink** field and type `google_play_badge.html`.
28. Click on **OK**.
29. To save a copy of the prototype, click on the **Publish** button on the toolbar and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and then clicking on **Generate HTML Files...**.
30. Note the directory where the prototype is saved.
31. Copy `itunes_badge.html` and `google_play_badge.html` to the same directory as your prototype.
32. In your prototype directory, open `start.html` in a browser to view the prototype.

How it works...

The key to this recipe is using the inline frame widget with an external HTML file set as the default target. The external HTML file contains the HTML code for the badge between the `<body>` and `</body>` tags. You place the external HTML file in the same folder as the root of the prototype output. When the prototype is run, the inline frame widget runs the external HTML file and generates the badge. Clicking on the badge opens the referenced link in a new window.

There's more...

Google also provides a device art generator that will take screenshots of your app and skin them in a device. This is great for helping others visualize your app on their device. The Device Art Generator can be found at `http://developer.android.com/distribute/promote/device-art.html`. For the Device Art Generator to work properly, you will also need to have Google Chrome installed. Google Chrome can be downloaded at `https://www.google.com/intl/en/chrome/browser/`.

3
Specifications

In this chapter, we will discuss ways to streamline your workflow in Axure as well as customize documentation for your organization. You will learn the following:

- Customizing the document template
- Standardizing your annotations
- Annotations for widgets
- Annotations for pages
- Specification customization

Introduction

Axure is a powerful tool not only for communicating design and prototyping, but also for generating specifications. We will cover quick recipes to customize the document template and define standard nomenclature for annotations and will walk through generating specifications. Note that the specification generation feature and the recipes outlined in this chapter are only available with Axure RP Pro.

Customizing the document template

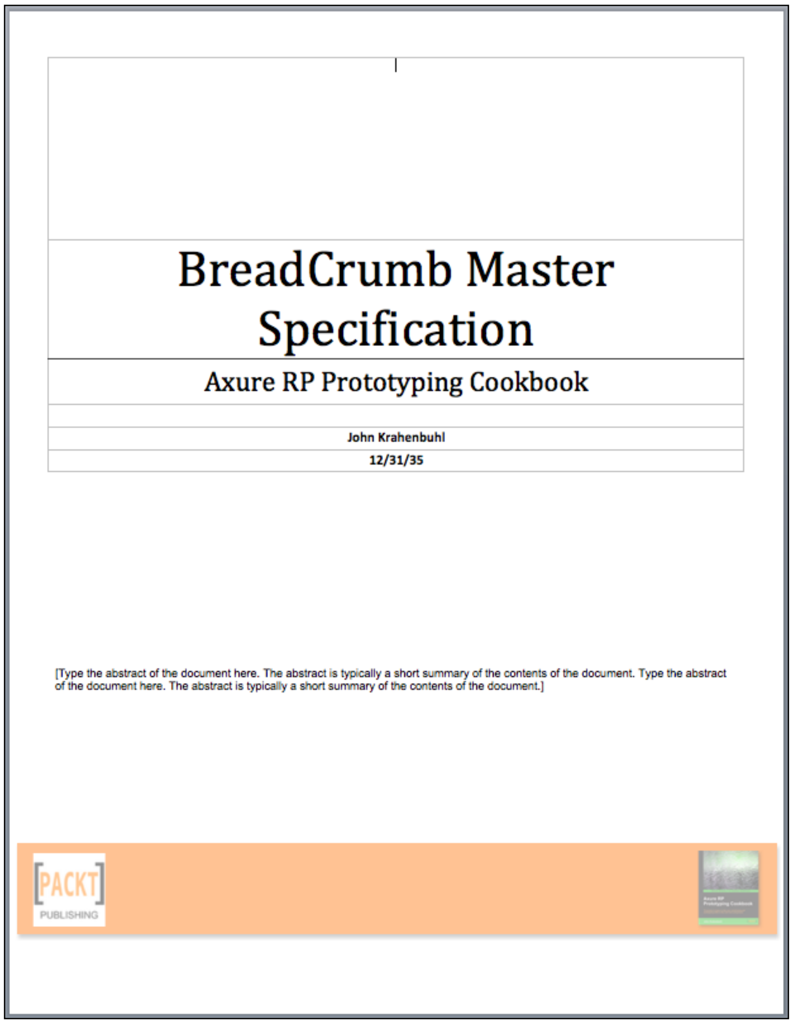
One of the beautiful features of Axure is its ability to generate annotated wireframes and to document specifications. With Axure, you can use the default Microsoft Word template to generate specifications, edit the default template, or import your own customized template, including headers and footers. In this recipe, we will explore how to customize the document template.

Getting ready

To follow this recipe, you will need:

- Microsoft Word or an open source word processor that can open and save files in Microsoft Word format, such as OpenOffice, LibreOffice, NeoOffice, or Google Docs
- Axure
- A previously created Axure RP file populated with your wireframes

The following is a screenshot from an example specification document:



BreadCrumb Master Specification

Axure RP Prototyping Cookbook

John Krahenbuhl

12/31/35

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

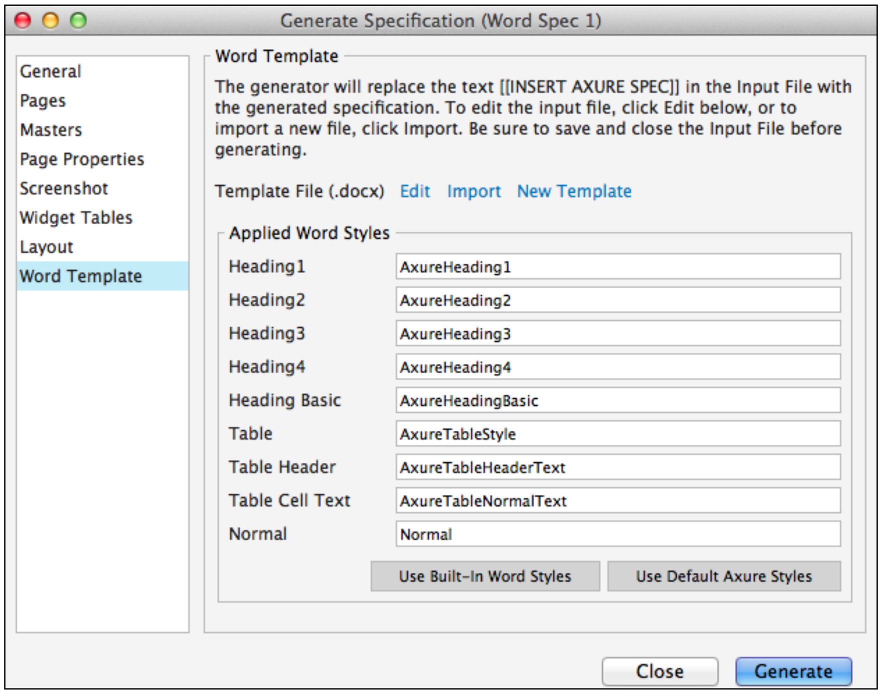
How to do it...

For this recipe, you will create a custom document template from the Axure base specification template. We will use the Axure RP file from the *Creating a dynamic Breadcrumb Master* recipe in *Chapter 1*, *Prototyping Recipes*, to create a specification document.

1. Start Axure, click on **Open...**, and select `breadcrumb_master.rp`.

> If you already have Axure open, click on **File** in the main menu and then click on **Open...** in the drop-down menu to create a new RP document.

2. For Axure 7.0, click on **Publish** above the wireframe pane. You can also select **File** in the main menu and then click on **Generate Specification...** in the drop-down menu.
3. In the **Generate Specification** pop up, click on **Word Template**.
4. Under the **Word Template** option, click on **Edit**. Axure will open the base specification template in Microsoft Word.



Word Template

The generator will replace the text [[INSERT AXURE SPEC]] in the Input File with the generated specification. To edit the input file, click Edit below, or to import a new file, click Import. Be sure to save and close the Input File before generating.

Template File (.docx) Edit Import New Template

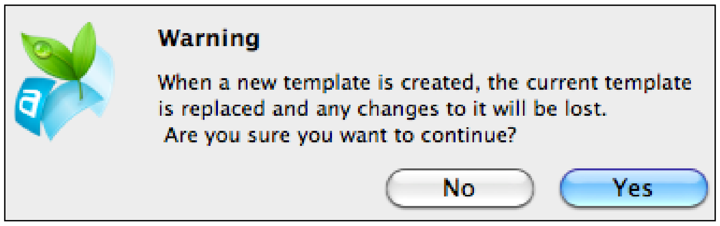
Applied Word Styles

Heading1	AxureHeading1
Heading2	AxureHeading2
Heading3	AxureHeading3
Heading4	AxureHeading4
Heading Basic	AxureHeadingBasic
Table	AxureTableStyle
Table Header	AxureTableHeaderText
Table Cell Text	AxureTableNormalText
Normal	Normal

5. Make modifications to the template (for example, to the document cover page, header, footer, and so on) and then save it as `AxureCookbookInputFile.docx`.

> In the template, note the text **[[INSERT AXURE SPEC]]**. The specification generator replaces this text with the Axure-generated specification.

6. To import the template file, click on **Import**.
7. An **Open** pop up will be displayed.
8. Select `AxureCookbookInputFile.docx` and click on **Open**.
9. A **Warning** pop up will be shown letting you know that the current template will be replaced. Click on **Yes**.



Warning

When a new template is created, the current template is replaced and any changes to it will be lost.
Are you sure you want to continue?

10. Click on the **General** menu item.
11. To the right of the **Destination file** field, click on the icon with the three dots (ellipsis) to change the location in which the destination file will be saved.
12. Enter `breadcrumb_master.docx` in the **Destination file** field, select the folder you wish to save the specification in, and click on **Save**.
13. Click on the **Publish** button on the toolbar and select **Generate Prototype Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and then on **Generate Prototype Files...**.
14. The specification document will be generated using the `AxureCookbookInputFile.docx` file you specified in step 8. The file will be saved to the destination file location as `AxureCookbookInputFile.docx`, and the specification file will open automatically.

How it works...

By using a custom template, you can generate a customized specification document for your organization. You can modify the default template provided by Axure or use your own. Just remember to place the text `[[INSERT AXURE SPEC]]` in the document where you want the generator to place the specification.

There's more...

Be sure to explore the other options in the **Generate Specification** pop up. You can specify which items to include in the generate specification document. Options provided in the **Generate Specification** menu are as follows:

- **Masters**
- **Page Properties**
- **Screenshot**
- **Widget Properties**
- **Layout**

See also

- The *Creating a dynamic BreadCrumb Master* recipe in *Chapter 1*, *Prototyping Recipes*

Standardizing your annotations

To streamline your workflow and assist others in understanding your functional specifications, it helps to follow a nomenclature. In this recipe, we will discuss the basic methods to standardize annotations for your wireframes.

Many individuals have used various naming conventions throughout their careers. Currently, two naming conventions stand out: **CamelCase** and **Underscore**.

Since most programming languages do not let a programmer use spaces in names, the CamelCase and Underscore methods have become good alternatives.

In CamelCase, most characters are in lowercase. The first word is lowercase and the first letter of subsequent words is upper case. For example, "menuContent" has the first word in lowercase and the first letter of the second word in uppercase.

In Underscore, words are separated with underscores. For example, "menu_content" has the first and second words separated by an underscore.

Axure uses modified CamelCase in that the first letter of each word is capitalized. For example, "OnClick" has the first letter of the first and second words capitalized.

Getting ready

All you will need is a word processor or text editor to create a document.

How to do it...

For this recipe, you will create a document with your proposals for naming conventions and annotations.

1. Select several naming conventions for your annotations.
2. Create a document with example widgets and interactions showing the various proposals of naming conventions.
3. List the pros and cons of the various proposals.

> Sometimes, the names of elements can get lengthy. If you see this as an issue, explore several different standards for shortening the names of common elements (for example, `dynamic_panel_carousel` becomes `dp_carousel`).

4. Document and discuss these options throughout your organization and collect feedback.
5. Refine your document based on the feedback and suggestions gathered.
6. Finalize your proposal and obtain sign offs from the decision makers in your organization.

How it works...

Standardizing your annotations can help others embrace the evolution of different designs and interactions. By using and communicating a standard nomenclature for the description of elements, you can create an atmosphere of comfort, enabling others to get beyond the documentation and focus on the vision.

Annotation for widgets

Labelling widgets in your wireframes helps to communicate the intent for those widgets. Labels provide you with a reminder as to the purpose of each widget as well. Consistently labelling widgets is another best practice as it assists others in understanding your functional specifications.

How to do it...

In this recipe, you will place the **Image** widget on a wireframe and label the widget.

1. Start Axure, and under **Create New**, select **RP Document**.

[If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.]

2. While holding down the mouse button, drag the **Image** widget and place it at coordinates (0,0) on the wireframe.
3. In the **Widget Properties and Notes** pane, select the **Image** widget, click on the **Image Name** text field, and enter the name of the image (for example, `CompanyLogo`).
4. Experiment by dragging other widgets onto the wireframe and naming the respective widget name in the **Widget Properties and Notes** pane.

How it works...

Axure associates the label entered in the **Widget Properties and Notes** pane's **Label** text field with the widget. As you further explore Axure and its features, you will see that labels are referenced throughout your prototype. Labels also appear in widget tables displayed in your specification document.

Annotations for pages

Labelling each page in your design helps to communicate the intent for that page. Axure RP Pro enables the use of page notes. Notes can be separated into separate fields (for example, developers, testers, clients, and so on).

How to do it...

1. Start Axure, and under **Create New**, select **RP Document**.
2. Double-click on **Home** in the **Sitemap** pane.
3. Below the wireframe, click on the **Page Notes** tab.
4. Click on the **Customize Notes** link.
5. In the **Page Notes Fields** pop up, click on the green **+** and type a new field name. Add new fields and rename them as appropriate (for example, developers, testers, clients, and so on).
6. Click on **Page Notes** and you will see each of the note fields you added in step 5.
7. Select the appropriate note field and type `page notes`.

[Page notes will be shown in your generated specification typically just above the screenshot.]

How it works...

For Axure RP Pro users, you can add **Page Notes** to each page in your **Sitemap**. Click on the **Page Notes** tab to view current **Page Notes**. Click on **Page Notes** to select from a currently defined note field and add a new page note.

Specification customization

Axure provides several options for customization of the specification generated. In this recipe, we will cover some of the options in the **Generate Specification** dialog. You will be shown the options for a specific menu item and a brief explanation of the properties available will follow. We will discuss **Pages**, **Masters**, **Page Properties**, **Screenshot**, **Widget Properties**, and **Layout**.

Getting ready

To step through this recipe, you will need:

- Microsoft Word or an open source word processor that can open and save files in Microsoft Word format, such as OpenOffice, LibreOffice, NeoOffice or Google Docs
- Axure
- A previously created Axure RP file populated with your wireframes

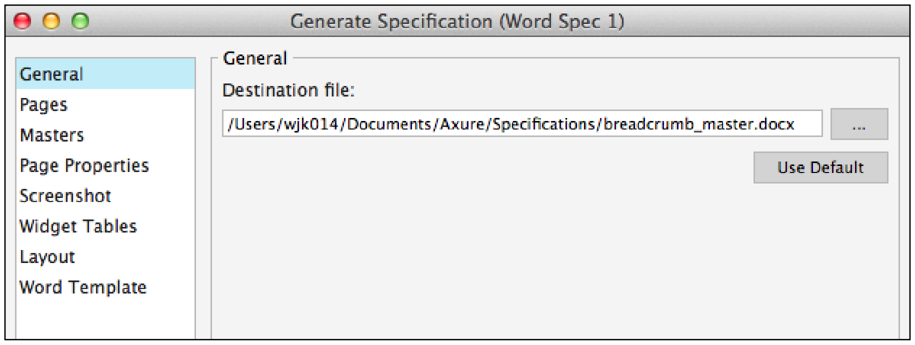
How to do it...

For this recipe, you will use the Axure RP file from the *Create a dynamic Breadcrumb Master* recipe in *Chapter 1*, *Prototyping Recipes*, to create a specification document.

1. Start Axure, click on **Open**, and select `breadcrumb_master.rp`.

> If you already have Axure open, select **File** in the main menu and then click on **Open...** in the drop down menu to create a new RP document.

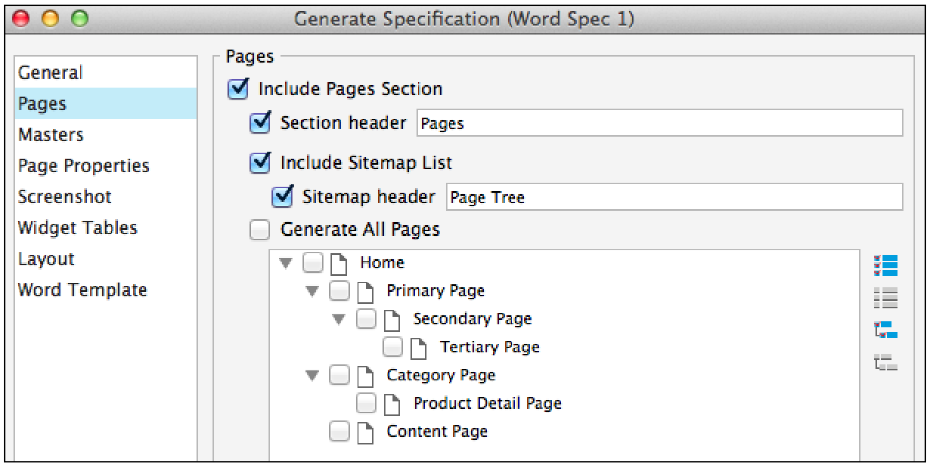
2. Click on **Publish** above the wireframe pane. You can also select **File** in the main menu and then click on **Generate Specification...** in the drop-down menu.
3. In the **Generate Specification** pop up, click on **Word Template**.
4. You will see the **Generate Specification** dialog open.



General

Destination file: /Users/wjk014/Documents/Axure/Specifications/breadcrumb_master.docx

5. Click on the **Pages** menu item.
6. You will see the **Pages** dialog open.



Pages

☑ Include Pages Section

☑ Section header: Pages

☑ Include Sitemap List

☑ Sitemap header: Page Tree

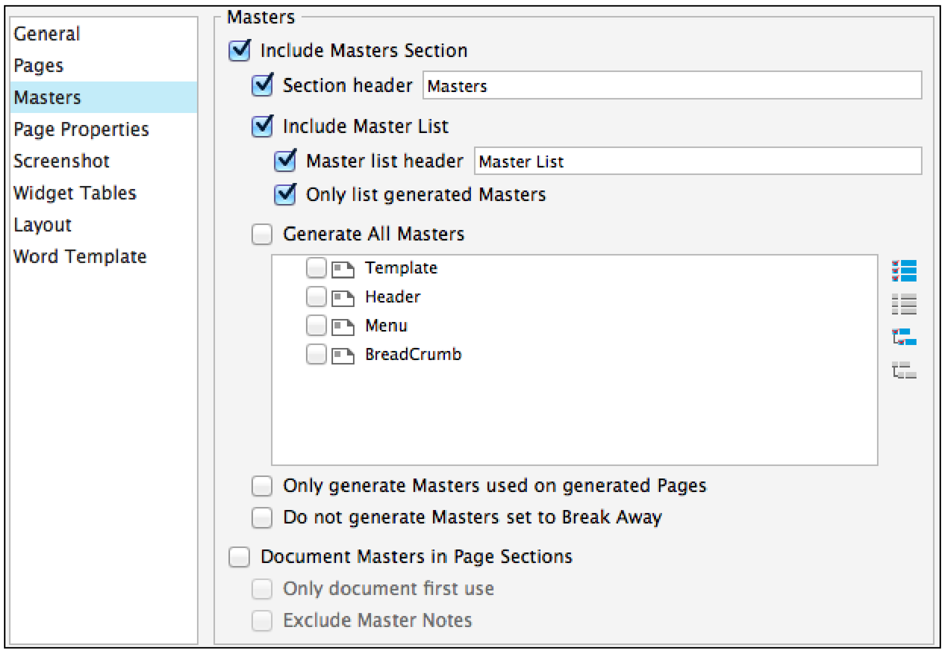
☐ Generate All Pages

7. Click on the check boxes to select the pages to include in the specification. You can also choose from the following options:
   - **Include Pages Section**
   - **Section Header**
   - **Include Sitemap List**
   - **Sitemap Header**
   - **Generate All Pages** or select individual pages to generate
   - **Screenshot header**
   - **Show footnotes on screenshot**
   - **Exclude footnotes not in widget tables**

To the right of **Generate All Pages**, just next to the sitemap shown, there are four icons. Clicking on an icon performs the action. The icons, from top to bottom, are:

- Check All
- Uncheck All
- Check All Children
- Uncheck All Children

8. Click on the **Masters** menu item.
9. You will see the **Masters** dialog open.



General
Pages
Masters
Page Properties
Screenshot
Widget Tables
Layout
Word Template

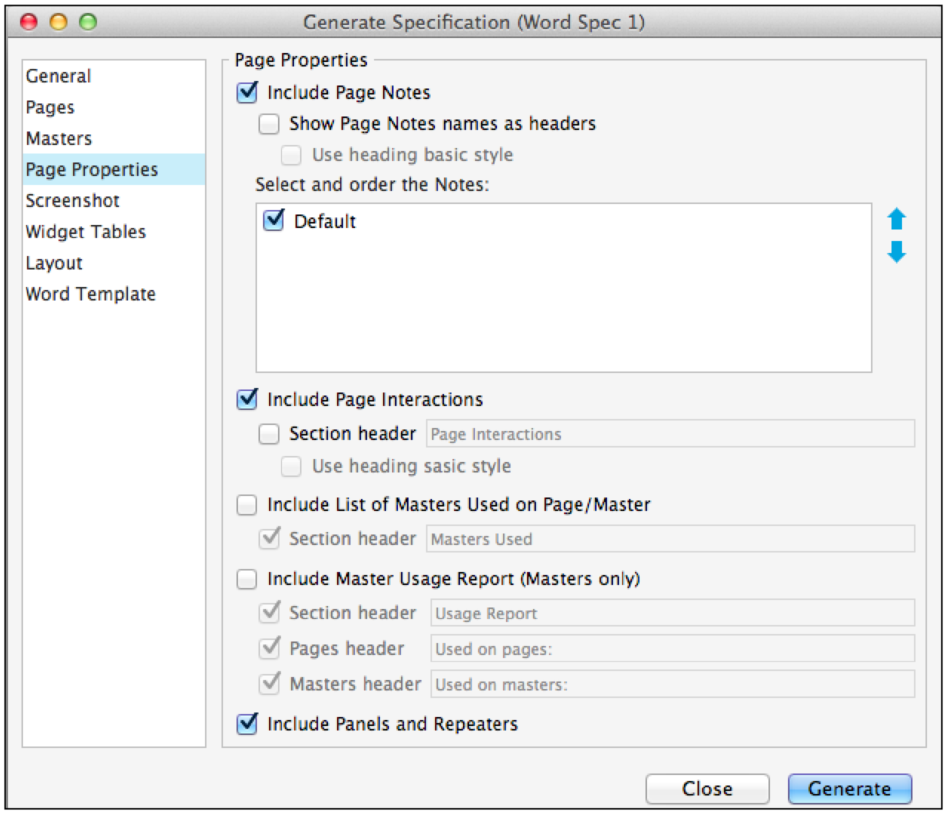
Masters
- [✓] Include Masters Section
  - [✓] Section header: Masters
  - [✓] Include Master List
    - [✓] Master list header: Master List
    - [✓] Only list generated Masters
  - [ ] Generate All Masters
    - [ ] Template
    - [ ] Header
    - [ ] Menu
    - [ ] BreadCrumb
  - [ ] Only generate Masters used on generated Pages
  - [ ] Do not generate Masters set to Break Away
- [ ] Document Masters in Page Sections
  - [ ] Only document first use
  - [ ] Exclude Master Notes

10. Click on the check boxes to select the masters to include in the specification. You can choose from the following options:
    - **Include Masters Section**
    - **Section header**
    - **Include Master List**
    - **Master list header**
    - **Only list generated Masters**
    - **Generate All Masters** or select individual masters to generate

To the right of the **Generate All Pages** option, just next to the sitemap shown, there are four icons. Left-clicking on an icon performs the action. The icons, from top to bottom, are:

- Check All
- Uncheck All
- Check All Children
- Uncheck All Children

11. You also can select from:
    - **Generate Masters Used on Generated Pages**
    - **Do Not Generate Masters Set As Custom Widgets**
    - **Document Masters in Page Sections** provides you with the option to **Only document first use** and **Exclude Master Notes**
12. Click on the **Page Properties** menu item.
13. You will see the **Page Properties** dialog open.


Generate Specification (Word Spec 1)

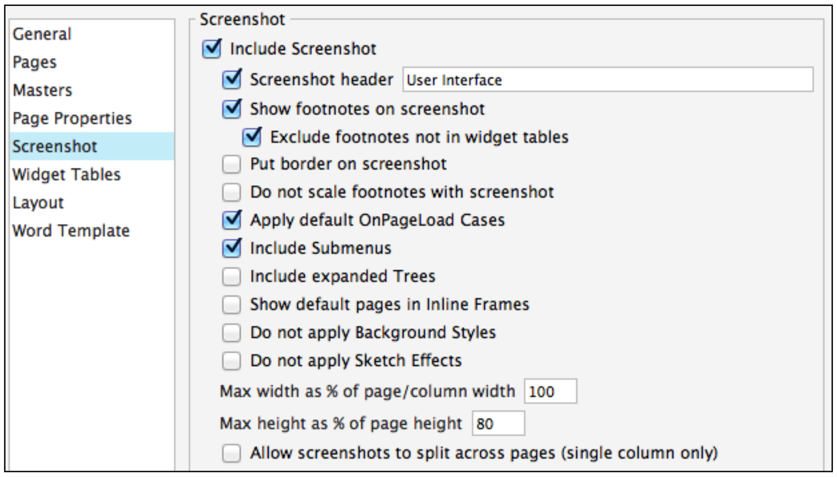
General
Pages
Masters
Page Properties
Screenshot
Widget Tables
Layout
Word Template

Page Properties

[✓] Include Page Notes
[ ] Show Page Notes names as headers
[ ] Use heading basic style
Select and order the Notes:
[✓] Default

[✓] Include Page Interactions
[ ] Section header: Page Interactions
[ ] Use heading sasic style
[ ] Include List of Masters Used on Page/Master
[✓] Section header: Masters Used
[ ] Include Master Usage Report (Masters only)
[✓] Section header: Usage Report
[✓] Pages header: Used on pages:
[✓] Masters header: Used on masters:
[✓] Include Panels and Repeaters

14. Click on the check boxes to select the page property options for the specification. You can choose **Include Page Notes** and **Show Page Notes names as headers**. If you select **Show Page Notes names as headers**, you are given the option to select **Use heading basic style**. You can also use the **Select and order the Notes** box. Click on a note and use the up and down arrows to change the order of the notes.
15. If you select **Include Page Interactions**, you can also select **Section header** and **Use heading basic style**.
16. If you select **Include List of Masters Used on Page/Master**, you can choose **Section header**.
17. If you select **Include Master Usage Report**, you can choose to include **Section header**, **Pages header**, and **Masters header**.
18. You can also choose **Include Dynamic Panels**.
19. Click on the **Screenshot** menu item.
20. You will see the **Screenshot** dialog open.

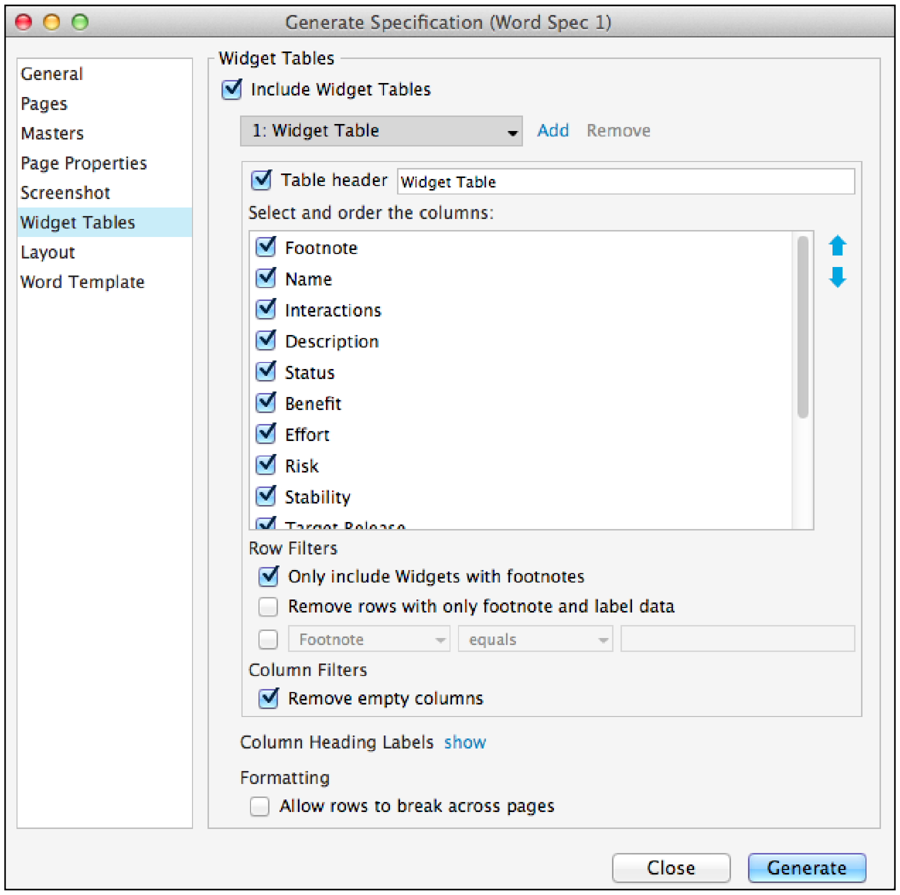


Screenshot
- [✓] Include Screenshot
  - [✓] Screenshot header: User Interface
  - [✓] Show footnotes on screenshot
    - [✓] Exclude footnotes not in widget tables
  - [ ] Put border on screenshot
  - [ ] Do not scale footnotes with screenshot
  - [✓] Apply default OnPageLoad Cases
  - [✓] Include Submenus
  - [ ] Include expanded Trees
  - [ ] Show default pages in Inline Frames
  - [ ] Do not apply Background Styles
  - [ ] Do not apply Sketch Effects
  - Max width as % of page/column width: 100
  - Max height as % of page height: 80
  - [ ] Allow screenshots to split across pages (single column only)

21. Click on the check boxes to select the screenshot options for the specification. You can choose from the following options:
    - **Include Screenshot**
    - **Screenshot header**
    - **Show footnotes on screenshot**
    - **Exclude footnotes not in widget tables**
    - **Put border on screenshot**
    - **Do not scale footnotes with screenshot**
    - **Apply default OnPageLoad Cases**

- **Include Submenus**
- **Include expanded Trees**
- **Show default pages in Inline Frames**
- **Do not apply Background Styles**
- **Do not apply Sketch Effects**

22. You can also set **Max width as % of page/column width** and **Max height as % of page height**. You also can enable **Allow screenshots to split across pages** for single-column specifications only.
23. Click on the **Widget Tables** menu item.
24. You will see the **Widget Tables** dialog open.



Widget Tables

☑ Include Widget Tables

1: Widget Table

☑ Table header: Widget Table

Select and order the columns:

☑ Footnote
☑ Name
☑ Interactions
☑ Description
☑ Status
☑ Benefit
☑ Effort
☑ Risk
☑ Stability
☑ Target Release

Row Filters

☑ Only include Widgets with footnotes
☐ Remove rows with only footnote and label data
☐ Footnote equals

Column Filters

☑ Remove empty columns

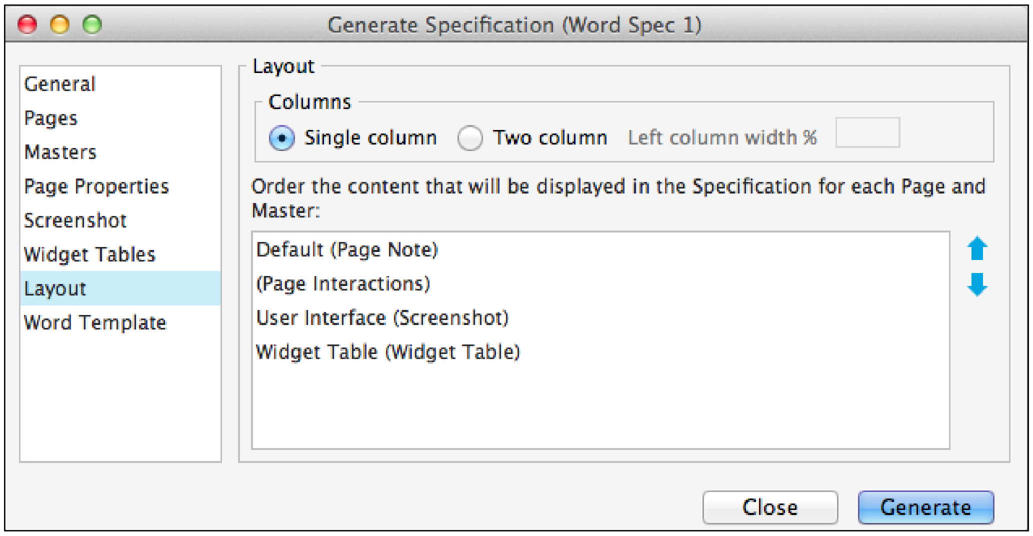
Column Heading Labels show

Formatting

☐ Allow rows to break across pages

25. Click on the checkbox for **Include Widget Tables** to show widget tables in your specification. You can add and remove widget tables by clicking on the **Add** and **Remove** links. Select the widget table you would like to modify by clicking on the widget table drop-down list.
26. Click on **Table header** to show a widget table header on the specification. You can choose from the following options:
    - **Footnote**
    - **Label**
    - **Interactions**
    - **Description**
    - **Status**
    - **Benefit**
    - **Effort**
    - **Risk**
    - **Stability**
    - **Target Release**
    - **Assigned To**
    - **Widget Text**
    - **Widget ToolTip**
    - **List Options**
27. You can set **Row Filters** to **Only include Widgets with footnotes** and **Remove rows with only footnote and label data**. You are given the option to customize a row filter by any of the preceding columns shown and testing it for equality to a string you enter into a text field.
28. For **Column Filters**, you can select **Remove empty columns**.
29. To see the **Column Heading Labels** options in your specification document, click on the **show** link. **Column Heading Labels** options can be changed from default for:
    - **Footnote**
    - **Label**
    - **Interactions**
    - **Text**
    - **Tool Tip**
    - **List Options**
30. For **Formatting**, you can choose **Allow rows to beak across pages**.
31. Click on the **Layout** menu item.

32. You will see the **Layout** dialog open.



Layout

Columns

(● Single column) ○ Two column   Left column width %

Order the content that will be displayed in the Specification for each Page and Master:

Default (Page Note)
(Page Interactions)
User Interface (Screenshot)
Widget Table (Widget Table)

33. Click on the radio button to specify a **Single column** or **Two column** layout. If you have selected two-column layout, you can change **Left column width %**. You can order the content displayed in the columns by clicking on the content type shown and clicking on the up and down arrows. Use **---Column Break---** to specify where the column break should be applied.

For Two-column layout, most folks choose to put **User Interface (Screenshot)** in a column by itself and the rest of the content in the other column. This leads to a cleaner look and makes the specification easier to review.

How it works...

As we have seen, there are many options you are provided with to customize your functional specifications. Feel free to explore these options and optimize the specification for your audience.

There's more...

Axure provides an excellent overview to customize and generate functional specifications. Navigate to `http://www.axure.com/learn/spec/functional-specifications` to learn more.

4

Wireframes and Design

In this chapter, you will create custom widgets for common page elements. You will learn about the following topics:

- Leveraging freely available widget libraries
- Creating a header Master
- Creating a main navigation Master
- Creating a left navigation Master
- Creating a footer Master
- Using multiple Masters on a page
- Enabling rounded turns on connectors
- Using stacked flow shapes
- Creating custom widget libraries
- Editing custom widgets

Introduction

Axure enables you to create widgets, export those widgets into a library, and share libraries. You can also create Masters. Masters can be leveraged as building blocks for multiple pages. A Master enables you to make a change on a given Master and have that change automatically updated across all of the pages containing that Master.

Axure also provides a flow library enabling you to create flow diagrams and sitemaps. You will enable rounded turns on connectors and leverage stacking flow shapes as well as cover quick recipes to import freely available widget libraries, create custom widget libraries, and edit custom widgets.

Leveraging freely available widget libraries

Axure has a robust, active community that is constantly creating new widget libraries. Axure.com provides an overview and links to some of the most popular current libraries at `http://www.axure.com/download-widget-libraries`.

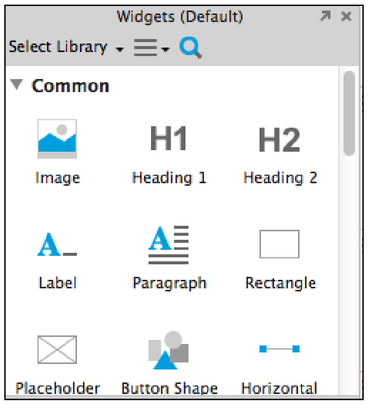
Getting ready

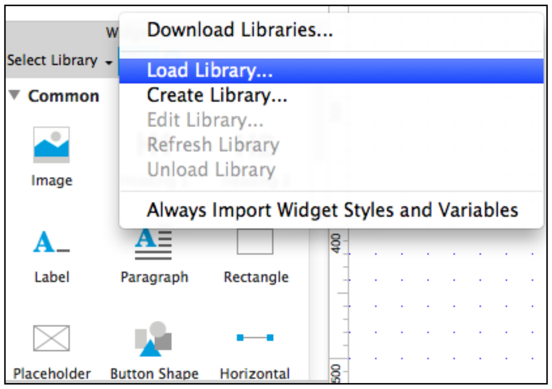
To step through this recipe, you will need a new widget library that you will have to download.

How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. In the **Widgets** pane toolbar, click on the second down arrow to open the drop-down menu, as shown in the following screenshots:





3. Left-click on **Load Library...**.
4. The system will open a window to browse for the Axure library file.

> Axure library files have a `.rplib` extension.

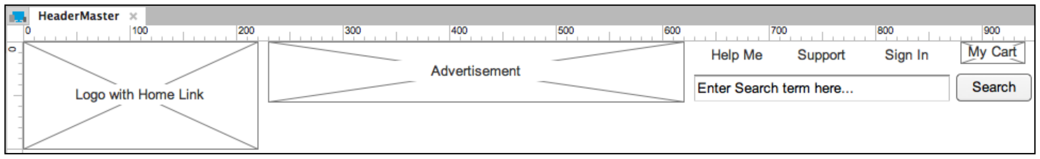
5. You will now see the widget library selected and the available widgets in the **Widgets** pane.
6. The new widget library will continue to load each time you restart Axure.
7. Axure widget library files are traditionally stored in `~/Documents/Axure/Libraries` for Mac users and the `~/My Documents/My Axure RP Libraries` folder for PC users.
8. If you would like to remove an Axure widget library, perform the following steps:
    1. In the **Widgets** pane toolbar, click on **Select Library** and then click on the library, you want to remove, from the drop-down menu.
    2. In the **Widgets** pane toolbar, click on the second down arrow to open the drop-down menu and click on **Unload Library**.

Creating a header Master

You can create a header Master to be used on each page of your design. This enables you to make a change on your header Master and have that change automatically updated across all of the pages containing that Master.

Getting ready

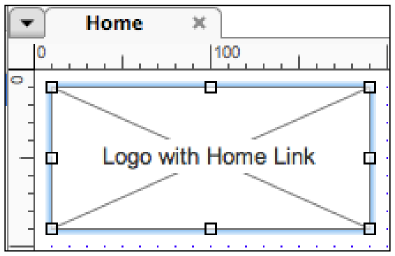
You will use **Placeholder**, **Label**, **Text Field**, and **Button** widgets to build a header Master, as shown in the following screenshot:



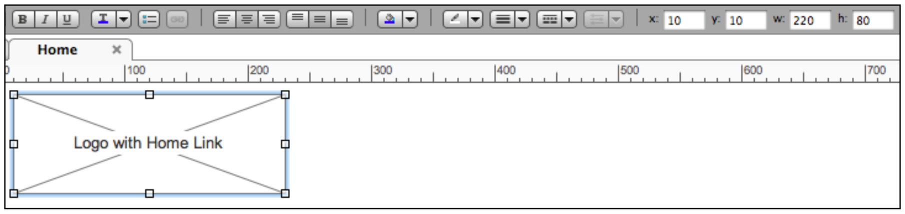
How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. While holding down the mouse button, drag the **Placeholder** widget and place it at coordinates (0,0) on the wireframe. Perform the following steps:
    1. Type in `Logo with Home Link`. The text will appear in the middle of the **Placeholder** widget as shown in the following screenshot:



    2. With the **Placeholder** widget selected, change the width **w:** to `220` and the height **h:** to `80` in the toolbar, as shown in the following screenshot:



x: 10 y: 10 w: 220 h: 80

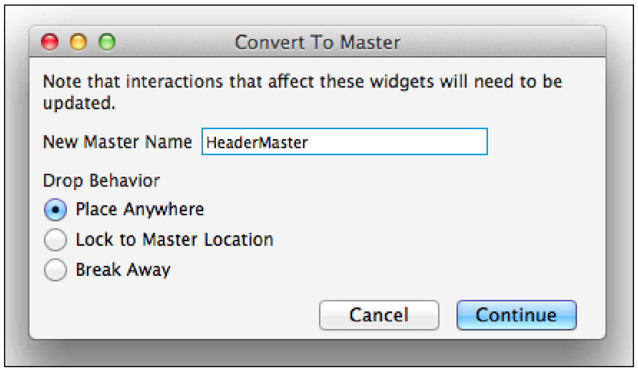
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `CompanyLogo`.
3. While holding down the mouse button, drag the **Placeholder** widget and place it at coordinates (230,0) on the wireframe. Perform the following steps:
    1. Type in `Advertisement`. The text will appear in the middle of the **Placeholder** widget.
    2. In the toolbar, change the width **w:** to `390` and the height **h:** to `60`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `AdvertisementCallToAction`.

4. While holding down the mouse button, drag the **Label** widget and place it at coordinates (640,5) on the wireframe. Perform the following steps:
    1. Type in `Help Me`. The text will appear in the middle of the **Label** widget.
    2. In the toolbar, change the width **w:** to `60`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `HelpMeLink`.
5. Repeat step 4 to create the Support and Sign In labels. Change the step using the following table as a reference for placement of the label widgets, text to display, and shape names for the label widgets.

Coordinates	Text Displayed	Shape Name
(720,5)	`Support`	`SupportLink`
(800,5)	`Sign In`	`SignInLink`

6. While holding down the mouse button, drag the **Placeholder** widget and place it at coordinates (880,0) on the wireframe. Perform the following steps:
    1. Type in `My Cart`. The text will appear in the middle of the **Placeholder** widget.
    2. In the toolbar, change the width **w:** to `60` and the height **h:** to `25`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `MyCartLink`.
7. While holding down the left mouse button, drag the **Text Field** widget and place it at coordinates (630,35) on the wireframe. Perform the following steps:
    1. Type in `Enter Search term here...`. The text will appear in the middle of the **Placeholder** widget.
    2. In the toolbar, change the width **w:** to `230`.
    3. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type in `SearchTextField`.
8. While holding down the mouse button, drag the **Button Shape** widget and place it at coordinates (870,35) on the wireframe. Perform the following steps:
    1. Type in `Search`. The text will appear in the middle of the **Placeholder** widget.
    2. In the toolbar, change the width **w:** to `70`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `SearchButton`.

9. You will now convert all widgets on the wireframe into a Master. In the main Axure menu, click on **Edit**, hover the mouse over **Select All** and click to select all the widgets on the wireframe. Perform the following steps:
    1. With all the widgets selected on the wireframe, right-click on any of the selected widgets.
    2. Click on **Convert to Master**.
    3. In the dialog box that pops up, type in `HeaderMaster` in the text field next to the **New Master Name** label. For **Drop Behavior**, click on the radio button next to **Place Anywhere**, as shown in the following screenshot:



Note that interactions that affect these widgets will need to be updated.

New Master Name: HeaderMaster

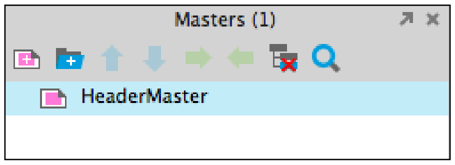
Drop Behavior

[✓ Place Anywhere]

Lock to Master Location

Break Away

    4. Click on the **Continue** button.
    5. You will see all of the widgets on the wireframe turn pink and a new Master titled **HeaderMaster** will be shown in the **Masters** pane, as shown in the following screenshot:



In the **Masters** pane, the pink color indicates that the Master can be placed at any location on the wireframe. In the **Convert To Master** dialog, if you would have selected **Lock to Master Location**, the Master would have been in gray, and when placed on a wireframe, the Master would lock to the coordinates in the Master. **Break Away** would appear as three gray boxes. **Break Away** means that you can edit widgets from the Master on each page of the wireframe where the Master has been used.

10. Save the Axure RP file as `header_master.rp`.

How it works...

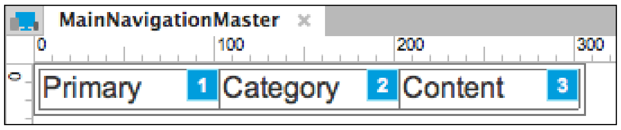
Selecting all of the widgets in the wireframe you wish to convert to a master, right-clicking on any of the selected widgets, hovering the mouse over the **Convert** menu, and then right-clicking on **Convert to Master**, brings up the **Convert To Master** dialog box. Entering a new name for the Master and then clicking on the **Continue** button creates the Master. All Masters in an Axure RP file are visible in the **Masters** pane.

Creating a main navigation Master

You can create a main navigation Master to be used on each page of your design. This enables you to make changes on your main navigation Master and have those changes automatically updated across all of the pages containing that Master.

Getting ready

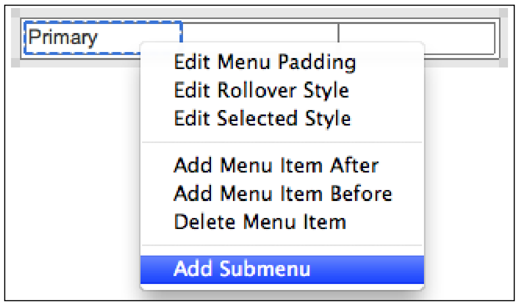
You will use a **Classic Menu - Horizontal** widget to build a main navigation Master, as shown in the following screenshot:



How to do it...

1. Start Axure and select **Create New RP Document**.
2. While holding down the left mouse button, drag the **Classic Menu - Horizontal** widget and place it at coordinates (0,0) on the wireframe. Perform the following step:
3. In the **Widget Interactions and Notes** pane, click on the **Menu Name** text field and type in `MainMenu`.
4. You will now name the **Primary**, **Category**, and **Content** main menu items.
5. To name the **Primary** menu item, perform the following steps:
    1. Click on the menu item labeled **File** to select and type in `Primary`.
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Name** text field and type in `MenuPrimary`.
    3. In the **Widget Properties and Style** pane, click on the **Style** tab and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, hover the mouse over **16**, and click on it to select.

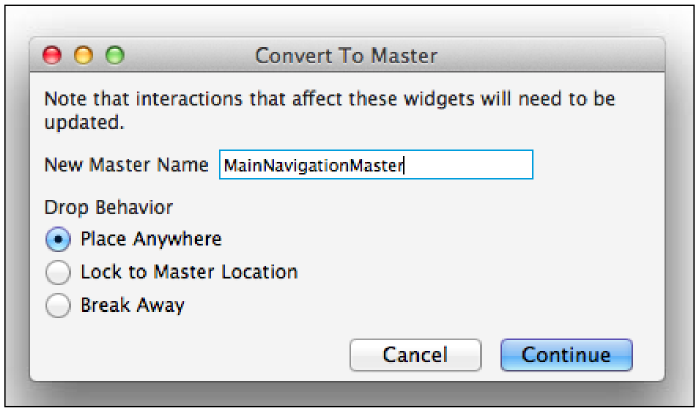
6. To name the **Category** menu item, perform the following steps:
    1. Click on the menu item labeled **Edit** to select it and type in `Category`.
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Name** text field and type in `MenuCategory`.
    3. In the **Widget Properties and Style** pane, click on the **Style** tab and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, hover the mouse over **16**, and click on it to select.
7. To name the **Content** menu item, perform the following steps:
    1. Click the menu item labeled **View** to select and type in `Content`.
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Name** text field and type in `MenuContent`.
    3. In the **Widget Properties and Style** pane, click on the **Style** tab and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, hover the mouse over **16**, and click on it to select.
8. Next, you will add submenu items to the **Primary**, **Category**, and **Content** menu items.
9. To add submenu items to the **Primary** menu item, perform the following steps:
    1. Right-click on the **Primary** menu item, and click on **Add Submenu** as shown in the following screenshot:



    2. Click on the first submenu item and enter `Secondary`.
    3. Right-click on the second submenu item and click on **Delete Menu Item**, do the same for the third submenu item.
    4. Click on the submenu item `Secondary` to select it.
    5. Right-click on **Add Submenu**.
    6. Click on the first submenu item and enter `Tertiary`.
    7. Right-click on the second and third submenu items and click on **Delete Menu Item**.

10. To add a submenu item to the **Category** menu item, perform the following steps:
    1. Click on the submenu item **Category** to select it.
    2. Right-click on **Add Submenu**.
    3. Click on the first submenu item and enter `Product Detail`.
    4. Right-click on the second and third submenu items and click on **Delete Menu Item**.

11. You will now convert all widgets on the wireframe into a Master. In the main Axure menu, click on **Edit**, hover the mouse over **Select All**, and click on it to select all the widgets on the wireframe. Perform the following steps:
    1. With all the widgets on the wireframe selected, right-click on the gray highlighted border.
    2. Click on **Convert to Master**.
    3. In the dialog box that pops up, type in `MainNavigationMaster` in the text field next to the **New Master Name** label. For **Drop Behavior**, click on the radio button next to **Place Anywhere** as shown in the following screenshot:



Convert To Master

Note that interactions that affect these widgets will need to be updated.

New Master Name: MainNavigationMaster

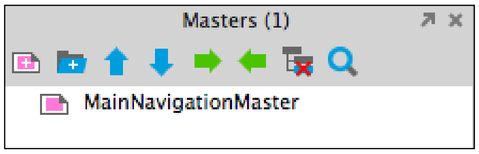
Drop Behavior

[✓ Place Anywhere]

Lock to Master Location

Break Away

    4. Click on the **Continue** button.
    5. You will see that all of the widgets on the wireframe turn pink, and a new Master titled **MainNavigationMaster** will be shown in the **Masters** pane, as shown in the following screenshot:



12. Save the Axure RP file as `main_navigation_master.rp`.

How it works...

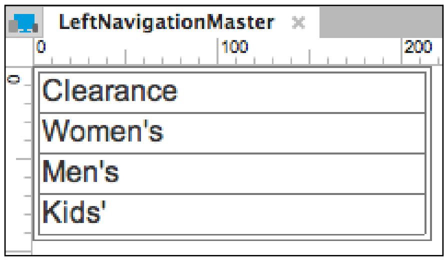
Selecting all of the widgets in the wireframe you wish to convert to a Master, right-clicking on any of the selected widgets, mousing over the **Convert** menu and then right-clicking on the **Convert to Master** menu brings up the **Convert To Master** dialog box. Entering a new name for the Master and then left-mouse clicking on the **Continue** button creates the Master. All Masters in an Axure RP file are visible in the **Masters** pane.

Creating a left navigation Master

You can create a left navigation Master to be used on each page of your design. This enables you to make changes on your left navigation Master and have those changes automatically updated across all of the pages containing that Master.

Getting ready

You will use **Classic Menu - Vertical** widgets to build a left navigation Master, as shown in the following screenshot:



How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. While holding down the left mouse button, drag the **Classical Menu - Vertical** widget and place it at coordinates (0,0) on the wireframe.
3. In the **Widget Interactions and Notes** pane, click on the **Menu Name** text field and type in `MainMenu`.
4. You will now name the **Clearance**, **Women's**, **Men's**, and **Kids'** main menu items.
5. To name the **Clearance** menu item, perform the following steps:
    1. Click on the menu item labeled **Item 1** to select and type in `Clearance`.

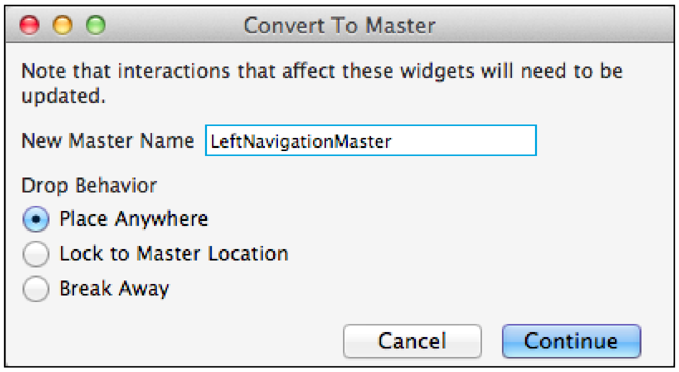
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field and type in `ClearanceLeftMenuItem`.
    3. In the **Widget Properties and Style** pane, click on the **Style** tab and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, hover the mouse over **16**, and click on it to select.

6. To name the **Women's** menu item, perform the following steps:
    1. Click on the menu item labeled **Item 2** to select and type in `Women's`.
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field and type in `WomensLeftMenuItem`.
    3. In the **Widget Properties and Style** pane, click on the **Style** tab and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, hover the mouse over **16**, and click on it to select.
7. To name the **Men's** menu item, perform the following steps:
    1. Click on the menu item labeled **Item 3** to select and type in `Men's`.
    2. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field and type in `MensLeftMenuItem`.
    3. In the **Widget Properties and Style** pane, click on the **Style** tab and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, hover the mouse over **16**, and click on it to select.
8. To add and name the **Kids'** menu item, perform the following steps:
    1. Right-click on the **Men's** menu item and click on **Add Menu Item After**.
    2. Click on the blank menu item to select and type in `Kids'`.
    3. In the **Widget Interactions and Notes** pane, click on the **Menu Item Name** text field and type in `KidsLeftMenuItem`.
    4. In the **Widget Properties and Style** pane, click on the **Style** tab and then scroll to the **Font** section. Increase the font size to **16** by clicking on the **Font** size dropdown, hover the mouse over **16**, and click on it to select.
9. Next, you will add submenu items to the **Women's**, **Men's**, and **Kids'** menu items.
10. To add submenu items to the **Women's** menu item, perform the following steps:
    1. Right-click on the **Women's** menu item and click on **Add Submenu**.
    2. Click on the first submenu item and enter `Clothing`.
    3. Click on the second submenu item and enter `Shoes`.
    4. Click on the third submenu item and enter `Accessories`.

11. To add a submenu item to the **Men's** menu item, perform the following steps:
    1. Right-click on the **Men's** menu item and click on **Add Submenu**.
    2. Click on the first submenu item and enter `Clothing`.
    3. Click on the second submenu item and enter `Shoes`.
    4. Click on the third submenu item and enter `Accessories`.

12. To add submenu items to the **Kids'** menu item, perform the following steps:
    1. Right-click on the **Kids'** menu item and click on **Add Submenu**.
    2. Click on the first submenu item and enter `Clothing`.
    3. Click on the second submenu item and enter `Shoes`.
    4. Click on the third submenu item and enter `Accessories`.

13. You will now convert all widgets on the wireframe into a Master. In the main Axure menu, click on **Edit**, hover the mouse over **Select All**, and click on it to select all the widgets on the wireframe. Perform the following steps:
    1. With all widgets on the wireframe selected, right-click on the gray highlighted border.
    2. Click on **Convert to Master**.
    3. In the dialog box that pops up, type in `LeftNavigationMaster` in the text field, next to the **New Master Name** label. For **Drop Behavior**, click on the radio button next to **Place Anywhere**, as shown in the following screenshot:

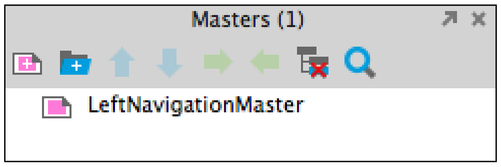

Convert To Master

Note that interactions that affect these widgets will need to be updated.

New Master Name: LeftNavigationMaster

Drop Behavior
- [✓ Place Anywhere]
- Lock to Master Location
- Break Away

    4. Click on the **Continue** button.
    5. You will see all of the widgets on the wireframe turn pink, and a new master titled **HeaderMaster** will be shown in the **Masters** pane, as shown in the following screenshot:



14. Save the Axure RP file as `left_navigation_master.rp`.

How it works...

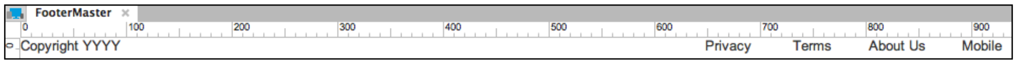
Selecting all of the widgets in the wireframe you wish to convert to a Master, right-clicking on any of the selected widgets, hovering the mouse over the **Convert** menu and then right-clicking on **Convert to Master** brings up the **Convert To Master** dialog box. Entering a new name for the Master and then clicking on the **Continue** button creates the Master. All Masters in an Axure RP file are visible in the **Masters** pane.

Creating a footer Master

You can create a footer Master to be used on each page of your design. This enables you to make changes on your footer Master and have those changes automatically updated across all of the pages containing that Master.

Getting ready

You will use **Label** widgets to build a footer Master as shown in the following screenshot:



How to do it...

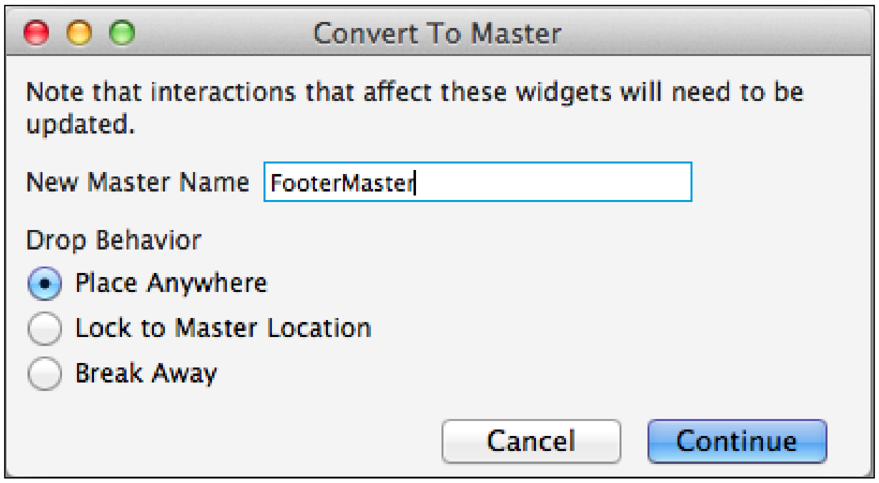
Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. While holding down the mouse button, drag the **Label** widget and place it at coordinates (0,0) on the wireframe. Perform the following steps:
    1. Type in `Copyright YYYY`. The text will appear in the middle of the **Label** widget.
    2. In the toolbar, change the width **w:** to `140`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `CopyrightLink`.

3. Repeat step 2 to create the Privacy, Terms, About Us, and Mobile Labels. Change the step using the following table as a reference for the placement of the label widgets, width, text to display, and shape names for the label widgets:

Coordinates	Width	Text Displayed	Shape Name
(640,0	60	`Privacy`	`PrivacyLink`
(720,0)	60	`Terms`	`TermsLink`
(800,0)	60	`About Us`	`AboutUsLink`
(880,0)	60	`Mobile`	`MobileLink`

4. You will now convert all widgets on the wireframe into a Master. In the main Axure menu, click on **Edit**, hover the mouse over **Select All** and click to select all widgets on the wireframe. Perform the following steps:
    1. With all widgets on the wireframe selected, right-click on the gray highlighted border.
    2. Click on **Convert to Master**.
    3. In the dialog box that pops up, type in `MainNavigationMaster` in the text field next to the **New Master Name** label. For **Drop Behavior**, click on the radio button next to **Place Anywhere**, as shown in the following screenshot:



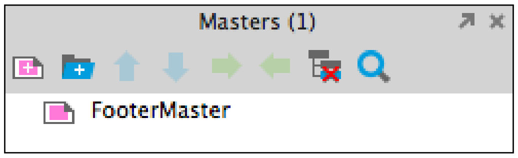
Convert To Master

Note that interactions that affect these widgets will need to be updated.

New Master Name: FooterMaster

Drop Behavior
[✓ Place Anywhere]
Lock to Master Location
Break Away

    4. Click on the **Continue** button.
    5. You will see that all of the widgets on the wireframe turn pink, and a new master titled **HeaderMaster** will be shown in the **Masters** pane, as shown in the following screenshot:



5. Save the Axure RP file as `footer_master.rp`.

How it works...

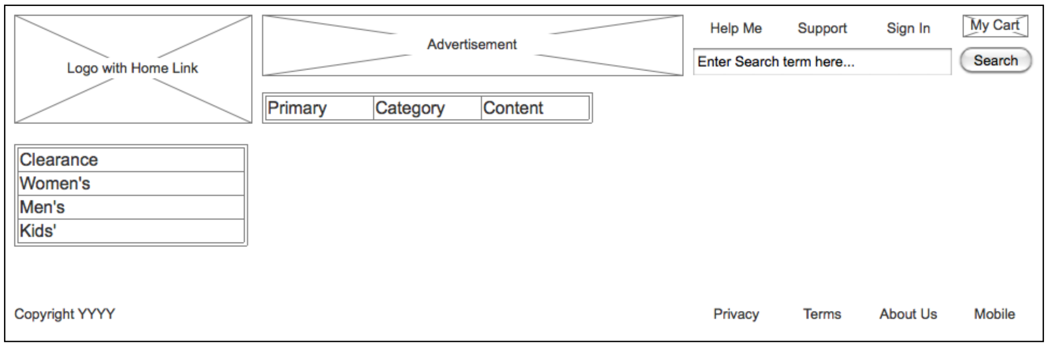
Selecting all of the widgets in the wireframe you wish to convert to a Master, right-clicking on any of the selected widgets, mousing over the **Convert** menu and then right-clicking on the **Convert to Master** brings up the **Convert To Master** dialog box. Entering a new name for the **Master** and then left-mouse clicking on the **Continue** button creates the Master. All Masters in an Axure RP file are visible in the **Masters** pane.

Using multiple Masters on a page

You can use multiple Masters to create a single page or multiple pages in your design. This enables you to use Masters as building blocks, knowing that you only have to update the Master once to automatically make the same change across all of the pages containing that Master.

Getting ready

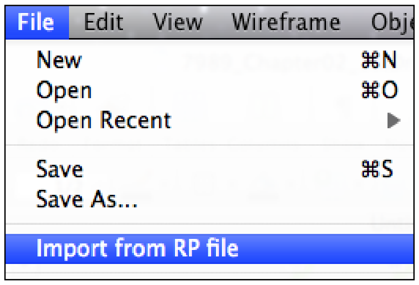
For this recipe, you will use the individual Axure RP files from the individual recipes for header Master, main navigation Master, left navigation Master, and footer Master, as shown in the following screenshot:



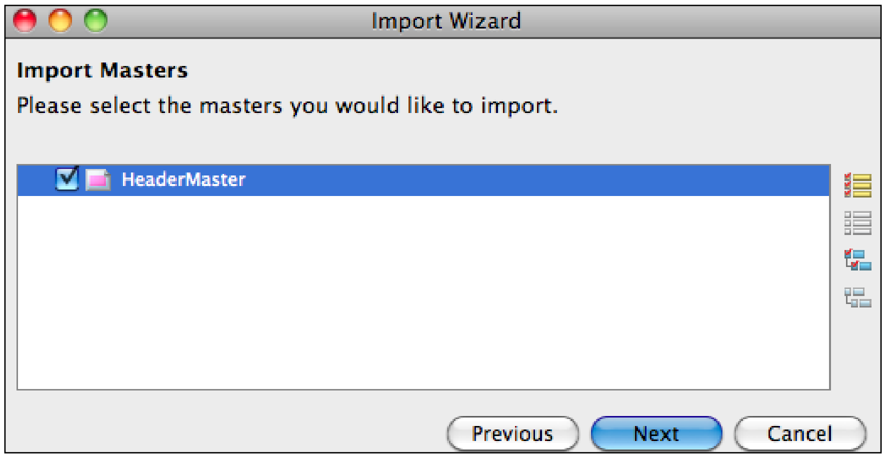
How to do it...

Perform the following steps:

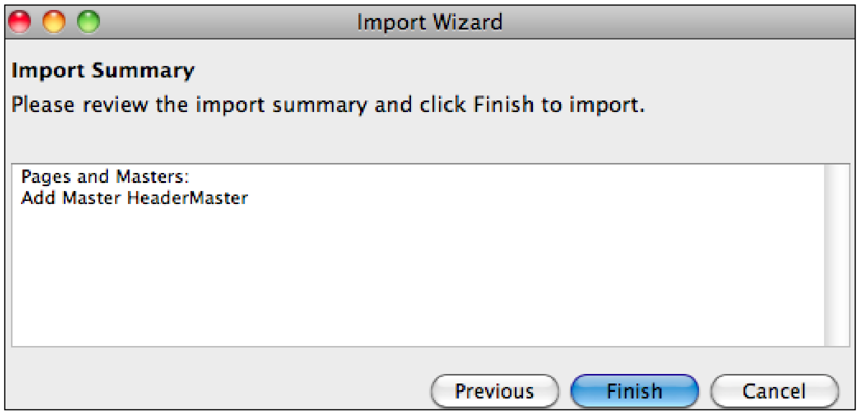
1. Start Axure and under **Create New**, select **RP File**.
2. You will now import the **HeaderMaster** Master. Perform the following steps:
    1. In the main Axure menu, click on **File** and then click on **Import from RP file** as shown in the following screenshot:



    2. Navigate to the Axure `header_master.rp` file and click on **Open**.
    3. The **Import Wizard** dialog will open in the **Import Pages** dialog. Since you intend to import the **HeaderMaster** Master and not pages, click on the **Next** button.
    4. The **Import Wizard** dialog will open to the **Import Masters** dialog. Click on the checkbox next to **HeaderMaster** and then click on the **Next** button, as shown in the following screenshot:



5. The **Import Wizard** dialog will open in the **Review Import Actions** dialog. Click on the **Skip to End** button.
6. The **Import Wizard** dialog will open to the **Import Summary** dialog showing the pages and Masters that are to be imported, as shown in the following screenshot:



Import Summary
Please review the import summary and click Finish to import.

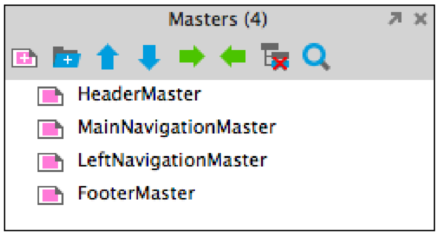
Pages and Masters:
Add Master HeaderMaster

7. Left-click on the **Finish** button.
8. You will see the imported Masters listed in the **Masters** pane.

3. Repeat step 2 to import the **MainNavigationMaster**, **LeftNavigationMaster**, and **FooterMaster** Masters. Change the step using the following table as a reference for the filename and Master to import:

Filename	Master
`main_navigation_master.rp`	MainNavigationMaster
`left_navigation_master.rp`	LeftNavigationMaster
`footer_master.rp`	FooterMaster

4. In the **Masters** pane, you should now see **HeaderMaster**, **MainNavigationMaster**, **LeftNavigationMaster**, and **FooterMaster** listed, as shown in the following screenshot:



5. You will now build the home page using the Masters you have imported. In the **Sitemap** pane, double-click on the page icon next to the **Home** page. Perform the following steps:
    1. While holding down the mouse button, in the **Masters** pane, click on the **HeaderMaster** Master and drag it to coordinates (10,10) on the **Home** wireframe.
    2. While holding down the mouse button, in the **Masters** pane, click on the **MainNavigationMaster** Master and drag it to coordinates (240,82) on the **Home** wireframe.
    3. While holding down the mouse button, drag the **LeftNavigationMaster** Master and place it at coordinates (10,130) on the **Home** wireframe.
    4. While holding down the mouse button, drag the **FooterMaster** Master and place it at coordinates (10,530) on the **Home** wireframe.
6. Save the Axure RP file as `multiple_masters.rp`.
7. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the preview button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar and then click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

By using the **Import from RP** file feature, you can import pages and Masters from individual Axure RP files. In this recipe, you imported Masters from four different Axure RP files. You then dragged each Master you wanted to include onto the same wireframe and adjusted the original Master when you needed to make refinements. Remember that all of the Masters and imported Masters in an Axure RP file are visible in the **Masters** pane.

Enabling rounded turns on connectors

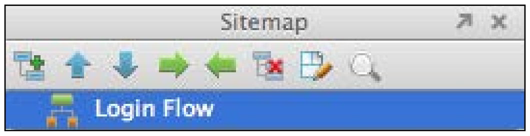
Several enhancements have been made to enhance flow diagrams in Axure. One of those enhancements is the option to enable rounded turns on connectors. In this recipe, you will create a basic flow diagram and enable the option to use rounded turns on connectors.

How to do it...

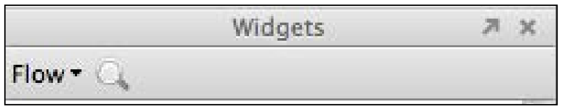
Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. In the **Sitemap** pane, click on the Add page icon in the toolbar.

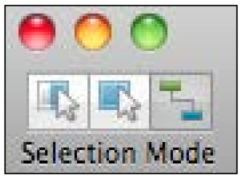
3. Click on the left arrow in the **Sitemap** pane toolbar to outdent the new page.
4. Right-click on the page icon next to the new page, a pop-up menu will appear.
5. Click on **Rename** and type in `Login Flow` and press *Enter*.
6. Click on the up arrow in the **Sitemap** pane toolbar to move the **Login Flow** page so that it is the first page in the **Sitemap** pane.
7. Right-click on the page icon next to the new page, and a pop-up menu will appear.
8. Hover your mouse over **Diagram Type** and in the flyout menu, click on **Flow**. You will see the **Page** icon change to a **Flow** icon.



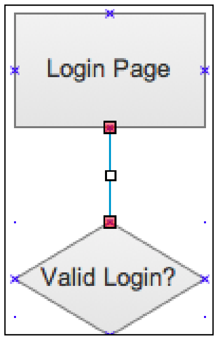
9. In the **Widgets** pane toolbar, click on the down arrow and select the **Flow** library as shown in the following screenshot:



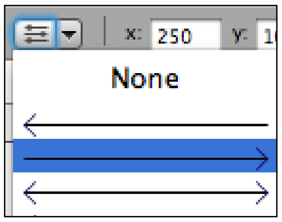
10. While holding down the mouse button, drag the **Rectangle** widget and place it at coordinates (200,40) on the wireframe.
11. With the **Rectangle** widget selected, type in `Login Page`.
12. While holding down the mouse button, drag the **Diamond** widget and place it at coordinates (200,150) on the wireframe.
13. With the **Rectangle** widget selected, type in `Valid Login?`.
14. While holding down the mouse button, drag the **Rectangle** widget and place it at coordinates (200,258) on the wireframe.
15. With the **Rectangle** widget selected, type in `Account Landing Page`.
16. For **Selection Mode**, click on the **Connector Mode** button to select it, as shown in the following screenshot:



17. Hover the mouse over the blue x at the bottom of the **Login Page** Rectangle widget. You will see a red square appear.
18. While holding down the mouse button, drag it to the blue **x** at the top of the **Valid Login?** Diamond widget. A blue connecting line will be placed on the wireframe connecting the two widgets, as shown in the following screenshot:

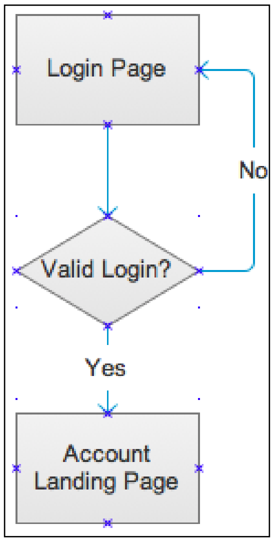


19. With the connecting line segment selected, in the main toolbar, left-click on the arrow style dropdown and hover the mouse over the line segment with the right arrow. You will see that the connected line segment changes to show the arrow style that is selected, as shown in the following screenshot:



20. Hover the mouse over the blue **x** at the bottom of the **Valid Login?** Diamond widget. You will see a red square appear.
21. While holding down the mouse button, drag it to the blue x at the top of the **Account Landing Page** Rectangle widget. A blue connecting line will be placed on the wireframe connecting the two widgets.
22. Note that the arrow style has already been applied to the connecting line segment since you selected this arrow style in step 19.
23. With the connecting line segment selected, type in `Yes`.
24. Mouse over the blue **x** on the right-hand side of the **Valid Login?** Diamond widget. You will see a red square appear.

25. While holding down the mouse button, drag it to the blue **x** on the right-hand side of the **Login Page** Rectangle widget. A blue connecting line with an arrow will be placed on the wireframe connecting the two widgets.
26. With the connecting line segment selected, type in `No`.
27. You will now see that the connecting line segment has rounded turns, as shown in the following screenshot:



[For Axure RP 7.0, rounded-angle connectors are enabled by default. For Axure RP 6.5 to enable rounded-angle connectors, right-click on the connecting line segment, and a pop-up menu will appear. Hover the mouse over **Edit Connector** and click on **Rounded-Angle Connector** in the flyout menu.]

How it works...

Enabling rounded turns on connectors is one of the new ways you can enhance flow diagrams in Axure. You can select a connecting line segment, right-click on it to bring up the pop up, hover the mouse over **Edit Connector** to bring up the flyout menu and left-click on **Rounded-Angle Connector** to enable rounded turns on connectors.

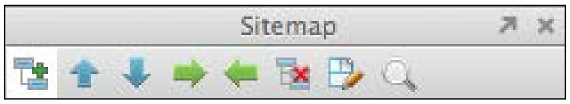
Using stacked flow shapes

Several enhancements have been made to enhance flow diagrams in Axure. One of those enhancements is the option to use a stacked flow shape widget in your flow diagrams. In this recipe, you will create a basic flow diagram and use a stacked flow shape widget.

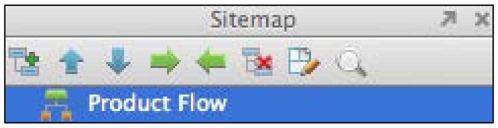
How to do it...

Perform the following steps:

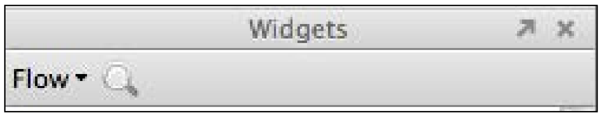
1. Start Axure and under **Create New**, select **RP File**.
2. In the **Sitemap** pane, click on the Add page icon in the toolbar, as shown in the following screenshot:



3. Click the left arrow in the **Sitemap** pane toolbar to outdent the new page.
4. Right-click on the page icon next to the new page, and a pop-up menu will appear.
5. Click on **Rename** and type in `Product Flow`.
6. Click the up arrow in the **Sitemap** pane toolbar to move the **Product Flow** page to be the first page in the **Sitemap** pane.
7. Right-click on the Page icon next to the new page, and a pop-up menu will appear.
8. Hover the mouse over **Diagram Type**, and in the flyout menu, click on **Flow**. You will see the **Page** icon change to a **Flow** icon, as show in the following screenshot:

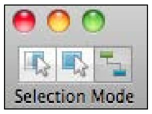


9. In the **Widgets** pane toolbar, click on the down arrow and select the **Flow** library as shown in the following screenshot:

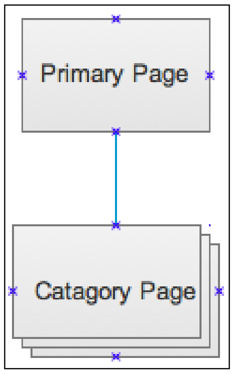


10. While holding down the mouse button, drag the **Rectangle** widget and place it at coordinates (200,40) on the wireframe. Type in `Primary Page`.
11. While holding down the mouse button, drag the **Stacked Rectangle** widget and place it at coordinates (195,150) on the wireframe. Type in `Category Page`.
12. While holding down the mouse button, drag the **Stacked Rectangle** widget and place at coordinates (200,264) on the wireframe. Type in `Product Landing Page`.

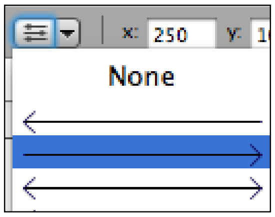
13. To select the **Selection Mode**, click on the **Connector Mode** button, as shown in the following screenshot:



14. Mouse over the blue **x** at the bottom of the **Primary Page** Rectangle widget. You will see a red square appear.
15. While holding down the mouse button, drag the blue **x** at the top of the **Category Page** Stacked Rectangle widget. A blue connecting line will be placed on the wireframe connecting the two widgets:



16. With the connecting line segment selected, in the main toolbar, click on the arrow style dropdown and hover the mouse over the line segment with the right arrow. You will see that the connected line segment changes to show the arrow style selected:



17. Hover the mouse over the blue **x** at the bottom of the **Category Page** Stacked Rectangle widget. You will see a red square appear.
18. While holding down the mouse button, drag the blue **x** at the top of the **Product Landing Page** Stacked Rectangle widget. A blue connecting line will be placed on the wireframe connecting the two widgets.

19. Note that the arrow style has already been applied to the connecting line segment since you selected this arrow style in step 16.

How it works...

The inclusion of stacked flow shape widgets is one of the new ways you can enhance flow diagrams in Axure. When widgets from the Flow library is shown in the **Widgets** pane while holding down the left mouse button, drag the stacked widget and place it on the wireframe.

There's more...

Note that the sizes of the standard widgets and the stacked widgets are different. For example, the default Rectangle widget has a width of 100 and a height of 60. The default Stacked Rectangle widget has a width of 110 and a height of 70. This is important because you can right-click on a Flow Shape widget, hover the mouse over **Edit Flow Shape**, and you can easily change between flow shape widgets (for example, **Rectangle** to **Stacked Rectangle**, **Stacked Rounded** to **Rounded Rectangle**). Knowing this difference up front enables us to adjust our flow diagrams to keep the connecting line segments flowing nicely.

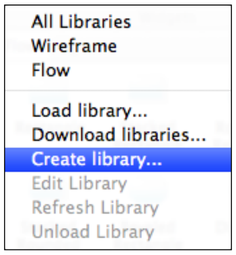
Creating custom widget libraries

You can create custom widget libraries to share within your team or with the greater part of the Axure community. A custom widget library can include custom widgets that you have created as well as existing widgets from other libraries.

How to do it...

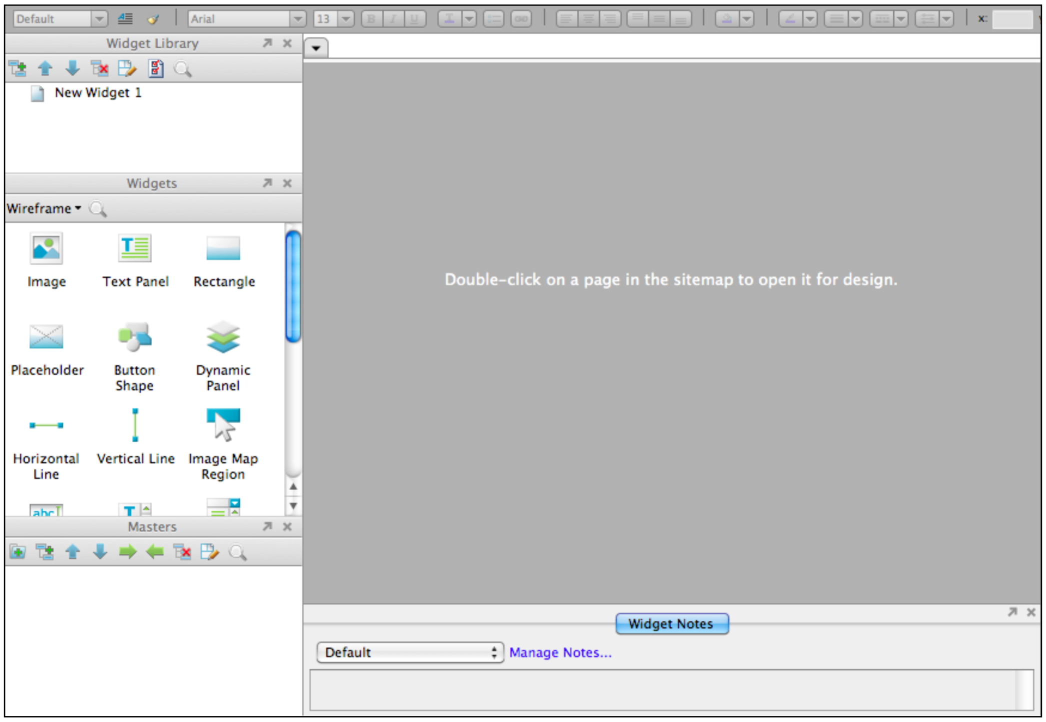
Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. In the **Widgets** pane toolbar, click on the down arrow and select **Create library...**, as shown in the following screenshot:



3. A **Save Axure RP Library** dialog box will open with the initial path defaulted to `~/Libraries`. Type a name for your library file and click on the **Save** button.

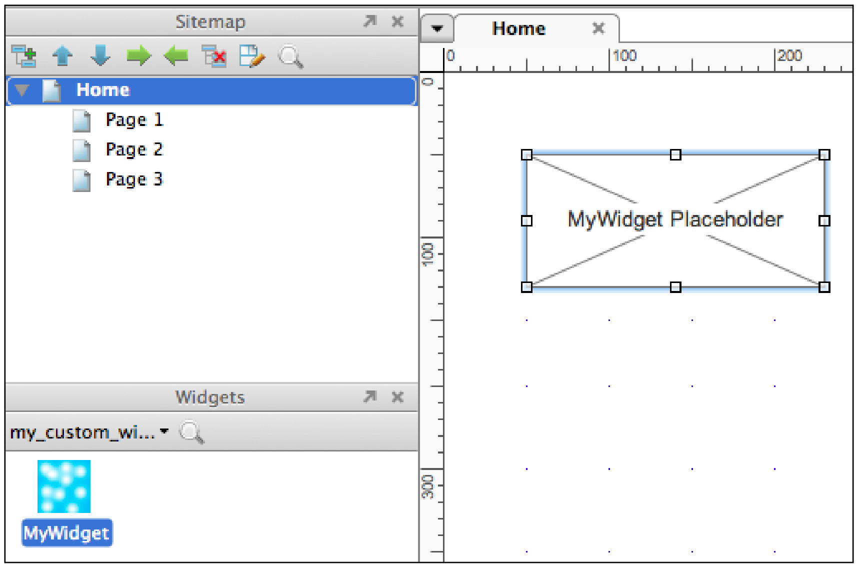
4. You will see that the **Sitemap** pane changes to the **Custom Widget Library** pane, and the wireframe design view will now be the area where you design and view your widgets.
5. As with pages in the **Sitemap** pane, double-click on widgets in the **Custom Widget Library** pane to open them in the widget library design view as shown in the following screenshot:



6. Right-click on the page icon next to the new widget, a pop-up menu will appear. Click on **Rename** and type in `MyWidget`.
7. In the **Custom Widget Library** pane, double-click on the Page icon next to **MyWidget**. You will see the **MyWidget** page open in the widget library design view.
8. Under the Widget Library Design view, click on the **Widget Properties** tab. Under **Icon**, you can click on the radio button next to **Use the thumbnail** to have Axure create a thumbnail of the widget screen. Click on the radio button next to **Custom Icon** and click on the **Import** button to import custom icons.

Standard icons are 28 x 28 pixels. High resolution icons for Mac retina displays are 56 x 56 pixels.

9. In the **Widgets** pane, click on the **Select Library** drop-down menu and click on **All Libraries**.
10. In the **Widgets** pane, while holding down the mouse button, drag the **Placeholder** widget and place it at coordinates (50,50) on the wireframe.
11. With the **Placeholder** widget selected, type in `MyWidget Placeholder`.
12. Save and close the custom Axure widget library.
13. Start Axure and under **Create New**, select **RP File**.
14. In the **Widgets** pane toolbar, click on the down arrow.
15. In the pop-up menu, you will see the filename listed for the custom Axure widget library you created.
16. Hover the mouse over the filename listed for the custom Axure widget library you created, and click on it to select.
17. You will see the widget named **MyWidget** with the icon you created.
18. In the **Widgets** pane, while holding down the mouse button, drag the **Placeholder** widget and place it at coordinates (50,50) on the wireframe.
19. You will see the **MyWidget** Placeholder widget on the wireframe, as shown in the following screenshot:



How it works...

In this recipe, you saved and created a custom widget library. You learned that you also can include a custom icon to be displayed in the widgets pane as well as define a custom tool tip for each new widget. You also learned that the new widget library continues to load each time Axure is restarted.

There's more...

To include Masters from other Axure RP files in your custom widget library, in the main Axure menu, left click on **File** and then click on **Import from RP file**. Follow the steps in the **Import** dialog that opens to import Masters and pages to be included in your custom widget library.

Editing custom widgets

A custom widget library can include custom widgets that you have created as well as existing widgets from other libraries. You can edit custom widgets stored in custom widget libraries.

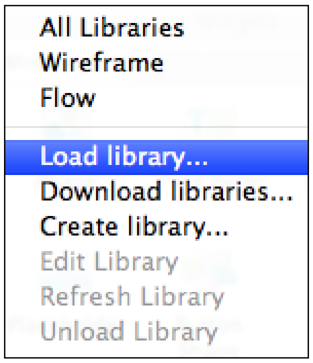
Getting ready

For this recipe, you will edit a custom widget loaded from a custom widget library.

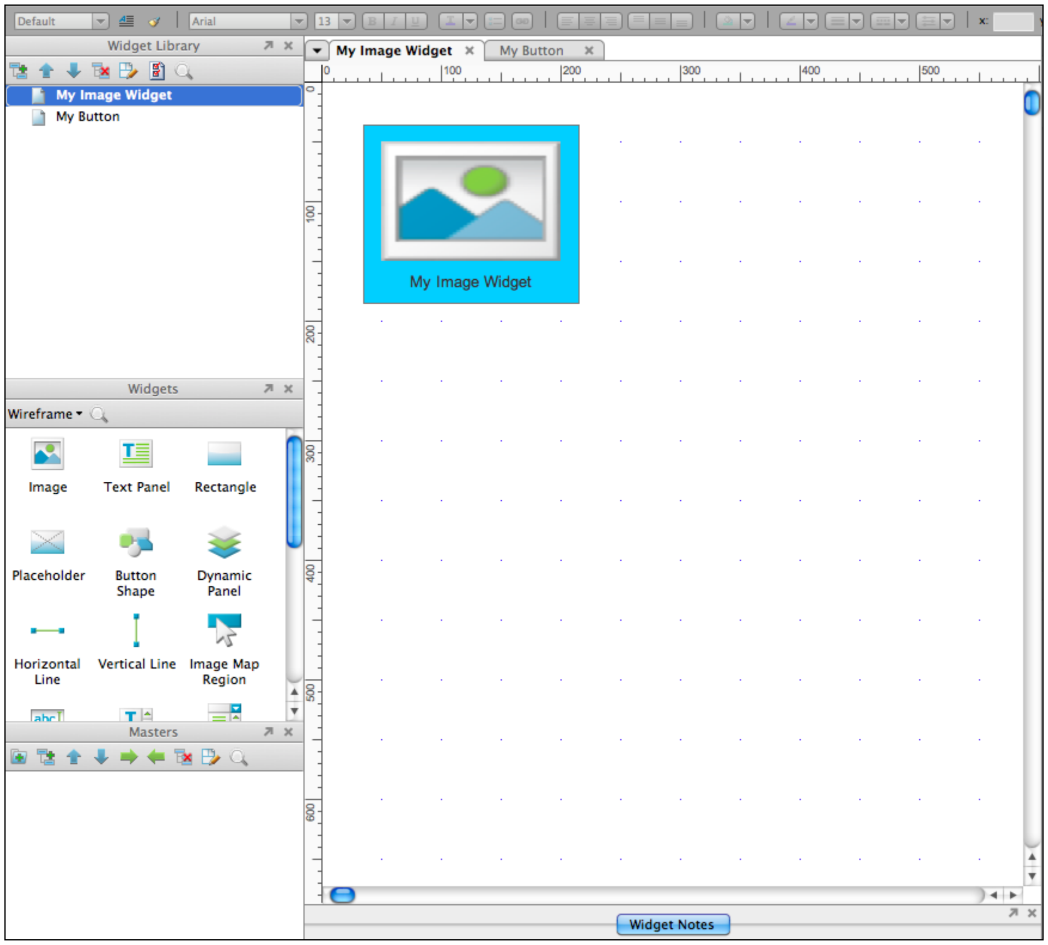
How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. In the **Widgets** pane toolbar, click on the down arrow and select **Load library...** as shown in the following screenshot:



3. An **Open** dialog box will open with the initial path defaulted to `~/Libraries`. Browse to your Axure RP library file and click on the **Open** button.
4. As with pages in the **Sitemap** pane, double-click on widgets in the **Widget Library** pane to open them in the widget library design view as shown in the following screenshot:



5. Make your changes in the open widget design view just as you would in a standard wireframe.
6. In the main menu, click on **Save** to save the updated custom library file.

How it works...

You can create, load, and save custom widget libraries. You also can make changes to custom widgets by loading the library, making changes to custom widgets, and then using create library to save the updated custom widget library.

There's more...

When creating a new custom library file, do not use the same filename as an existing library file. If you do, you will overwrite the existing custom library file with a blank library file, and your changes will be lost.

5

E-commerce Solutions

In this chapter, we will discuss e-commerce and explore several of the leading solutions in the market today. You will learn about the following topics:

- Building a Contact Us form
- Business-to-business (B2B) use cases
- A B2B invoice inquiry
- A B2B order checkout
- Business-to-consumer (B2C) use cases
- A B2C order status
- A B2C order checkout
- A Magento example
- A Zen Cart example

Introduction

Consumer spending is often one of the pillars of a growing economy. With the increasing number of Internet-connected devices activated each day, we have just begun to see the potential of e-commerce.

While the processes we follow for defining **business-to-business** (**B2B**) and **business-to-consumer** (**B2C**) use cases as well as representative user flows may be similar at times, the methods used to process information, browse and search for items, and the overall goals of the business-to-business customer and business-to-consumer customer may differ greatly.

While creating user personas, you should keep in mind the social styles and personality types in addition to use cases and outcomes. For your use case definitions, you should define the user's task, any actions or desires to consider, and finally develop test cases to further define the user's motivation and validate the outcome.

Building a Contact Us form

Giving visitors to your site a means to provide contact information is a great way to enable direct communication with future customers. In this recipe, you will build a Contact Us form.

How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP Document**.

> If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. In the **Sitemap** pane, delete the pages labeled **Page 1**, **Page 2**, and **Page 3** by right-clicking on each page, and click on the **Delete** option in the pop-up menu.
3. In the **Sitemap** pane, right-click on the page labeled **Home**, click on **Rename** in the pop-up menu, and rename the page to `Contact Us`.
4. While holding down the mouse button, drag the **Placeholder** widget and place it at coordinates (10,10) on the wireframe.
5. With the **Placeholder** widget selected, type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
6. With the **Placeholder** widget selected, in the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CompanyLogo`.
7. You will now focus on building out the body of the **Contact Us** page. While holding down the mouse button, drag the **Rectangle** widget and place it at coordinates (10,120) on the wireframe.
8. With the **Rectangle** widget selected, perform the following steps:
    1. In the toolbar, change the width **w:** to `525` and the height **h:** to `360`
    2. In the **Widget Interactions and Notes pane**, click on the **Shape Name** field and type in `ContactUsRectangle`
9. Drag the **Label** widget and place it at coordinates (20,135) on the wireframe.
10. With the **Label** widget selected, perform the following steps:
    1. Type in `Contact Us`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field-and type in `ContactUsH2Tag`.

   3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down list and selecting **16**.
   4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.

11. Drag the **Label** widget and place it at coordinates (20,170) on the wireframe.
12. With the **Label** widget selected, perform the following steps:
    1. Type in `Full Name`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `FullNameLabel`.
    3. Under **Alignment + Padding**, click on the third icon to align the text to the right within the label.
13. Drag the **Text Field** widget and place it at coordinates (200,165) on the wireframe.
14. With the **Text Field** widget selected, perform the following steps:
    1. Type in `Enter First and Last Name`. The text will appear in the **Text Field** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type `FullNameField`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the fourth icon with an A to bring up the color chooser drop-down menu.
    4. To change the color of the text, enter the hex value `CCCCCC` in the field and press *Enter*, or click anywhere outside the drop-down menu to select it.
15. While holding down the *Shift* key, click on both, the **Label** widgets you dragged onto the wireframe in step 11 and the **Text Field** widgets dragged onto the wireframe in step 13. In the toolbar, click on the Group icon.
16. Click on the group you created in step 15 and select **Copy** from the pop-up menu.
17. Click on the wireframe and select **Paste** from the pop-up menu.
18. Drag the new group to align it under the previous group.
19. Repeat steps 16 to 18 to create seven additional groups. You should now have a total of eight groups that comprises a label and text field. You should name the **Label** widgets and have the text displayed on the widget as follows: `Full Name`, `Address Line 1`, `Address Line 2`, `City`, `State/Providence`, `ZIP`, `Country`, and `Phone Number`. Name the **Text Field** widgets in each group and change the text displayed on the widget as appropriate (for example, for the **Address Line 1** label, the corresponding text field should display **Street**, **P.O. Box**, and **Company**).
20. Drag the **Button Shape** widget and place it at coordinates (420,442) on the wireframe.

21. With the **Button Shape** widget selected, type in `Submit`. The text will appear in the **Button Shape** widget.
22. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `SubmitButton`.
23. Click on the **Publish** button in the toolbar and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish**, and then clicking on **Generate HTML Files...**.

How it works...

In this recipe, we created a Contact Us form. We used the **Placeholder**, **Rectangle**, **Label**, **Text Field**, and **Button Shape** widgets to do this.

There's more...

We could add additional interactions to this simple prototype by designing a **Confirmation** page and assigning the **OnClick** action to the **Button Shape** widget.

Business-to-business (B2B) use cases

With the growth of e-commerce, more businesses have created web portals as well as sites for business-to-business communication and transactions. Often, a business-to-business customer will have a solid idea of what they are looking for. In addition, these customers may be more educated about your product offerings. A business-to-business customer will often demand a robust search experience.

Search solutions have moved beyond autocomplete and now focus on predictive search, including using GPS information on mobile devices, to further refine results. One such feature offered by Google is Google Now for predictive search. Another feature is NEXT's predictive search app that can be leveraged by iOS users. Robust predictive search has become a primary requirement to streamline the user experience and help widen the conversion funnel.

Another factor to consider is that many business-to-business customers will be repeat customers. This means that these customers may also wish to rapidly place the items and quantities from a previous order, or start with a completed order by adjusting quantities and augmenting the order with additional items. For this reason, we should provide a means for these customers to not only check order status but also allow them to use completed orders as a baseline for new orders. In addition, we can provide a means to automatically place a new order based on previous orders at a given interval (for example, ordering of office supplies every three months).

How to do it...

In this recipe, we will explore a process to identify, optimize, and communicate solutions for business-to-business use cases.

1. You will begin by defining requirements to understand the needs of your business-to-business customer.
2. Use one-on-one interviews with current business-to-business customers to understand their unique and collective use cases. Always remember to ask open-ended questions in order to receive unbiased responses.

A great resource on e-commerce topics, user testing, and more is `http://www.nngroup.com/articles`.

3. Use online surveys and questionnaires to collect primary data from your current business-to-business customers.
4. Conduct brainstorming sessions with your customers, sales, customer support, and other internal stakeholders.
5. Analyze all of the data collected and generate an optimized initial draft of your requirements.
6. Review your initial requirement draft with stakeholders and other key decision makers.
7. Complete revisions to your requirement draft.
8. Iterate through steps 6 to 7 until you have a final requirements draft.
9. Obtain sign off of your final requirements draft.
10. Perform competitive analysis to evaluate competitors within your industry as well as the best-in-class companies outside your industry.
11. Develop several user personas from demographic data that is representative of typical current and future customers.
12. Create scenarios and storyboards using various user personas.
13. Develop use cases highlighting the interaction between the user and the proposed system features.
14. Create user flows that highlight the streamlined and optimized execution of your use cases.
15. Develop your information architecture by organizing these user flows into an optimized, logical, and efficient sitemap and navigation schema.
16. Use the **Sitemap** pane in Axure to add/remove placeholder pages to communicate your page hierarchy and navigation schema.

17. You can now use Axure to quickly create wireframes for each of the pages shown in your **Sitemap**.
18. Iterate through steps 15 to 17 and gain approval from your stakeholders for the final sitemap and navigation schema.
19. You can now use Axure to create interactive prototypes to refine the proposed **user experience** (**UX**).
20. Conduct internal and external user studies to solicit feedback on your solution.
21. Analyze the amount of time it takes for users to complete individual tasks while interacting with your prototype and make improvements.
22. Use Axure to create annotated wireframes and generate the specification document.
23. Share the specification document and optimized prototype with your stakeholders.
24. Incorporate feedback and gain final approval for the final wireframes and UX.
25. You can now focus on the visual design for your project.

How it works...

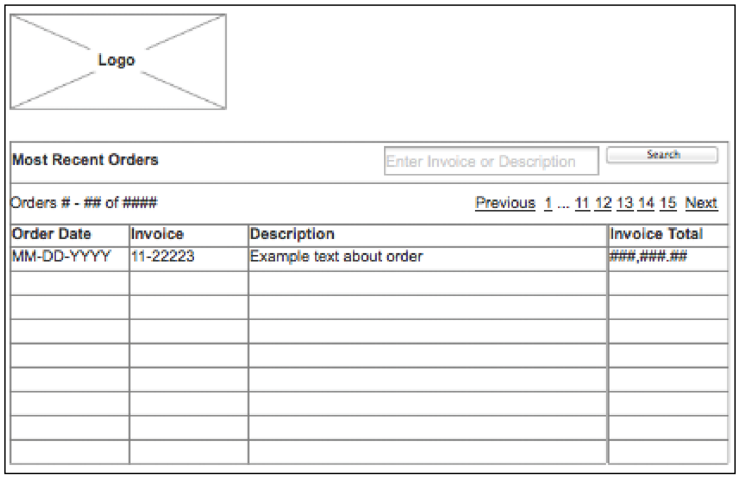
By using a user-centered design and process, we keep the focus on our customers and optimize the solutions we create. This approach helps enhance the user experience for our business-to-business customers, and as a result can lead to increased conversion rates.

A B2B invoice inquiry

It is important for business-to-business customers to quickly access information on completed and active orders. By providing a means to perform invoice inquiry, customers can quickly access the data they require.

Getting ready

In this recipe, we will create an **Invoice Inquiry** page and a sample invoice for business-to-business customers.


Logo
Most Recent Orders
Enter Invoice or Description
Search
Orders # - ## of ####
Previous 1 ... 11 12 13 14 15 Next

Order Date	Invoice	Description	Invoice Total
MM-DD-YYYY	11-22223	Example text about order	###,###.##

How to do it...

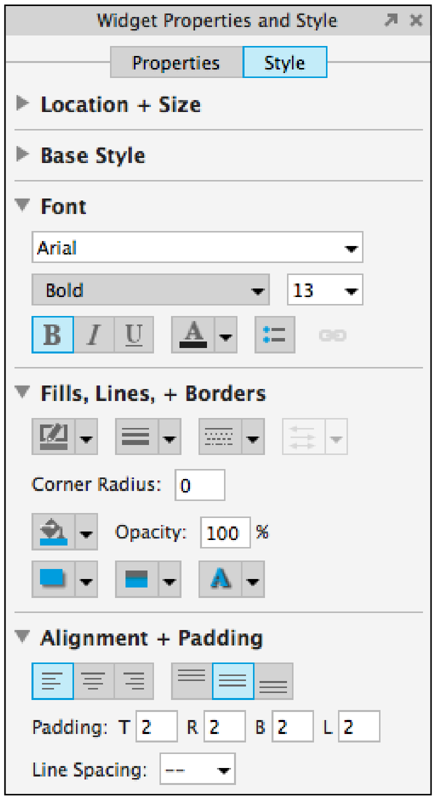
To create an **Invoice Inquiry** page and a sample **Invoice** page, you will use the **Placeholder**, **Rectangle**, **Text Field** and **Table** widgets. Perform the following steps:

1. Start Axure and under **Create New**, select **RP Document**.

> If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. In the **Sitemap** pane, right-click on the page labelled **Page 1**, then click on **Rename** in the pop-up menu, and type in `Invoice`.
3. In the **Sitemap** pane, right-click on the page labelled **Home**, click on **Rename** in the pop-up menu, and type in `Invoice Inquiry`.
4. Drag the **Placeholder** widget and place it at coordinates (10,13) on the wireframe.
5. With the **Placeholder** widget selected, type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
6. With the **Placeholder** widget selected, in the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CompanyLogo`.
7. In steps 4 to 6, you created a **Placeholder** widget with the **CompanyLogo** label. Feel free to add as many additional widgets to the right of the **Placeholder** widget to finish building out your header. Additional widgets that you might consider adding include **Site Title** with tag line, **Navigation**, and **Search**.
8. You will now focus on building out the body of the **Invoice Inquiry** page. Drag the **Rectangle** widget and place it at coordinates (10,120) on the wireframe.

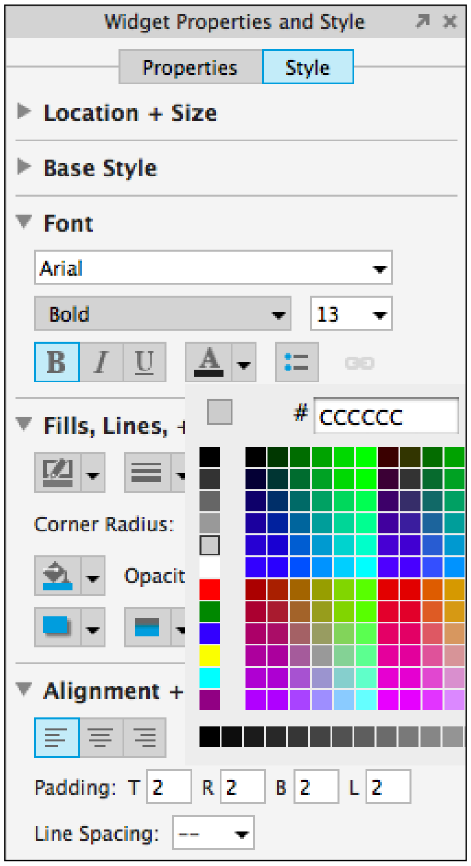
9. With the **Rectangle** widget selected, perform the following steps:
    1. Type in `Most Recent Orders`. The text will appear in the middle of the **Placeholder** widget.
    2. In the toolbar, change **w:** to `600` and **h:** to `35`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type `MostRecentOrdersHeadline`.
    4. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the first icon with a B to make the text bold within the rectangle. Under **Alignment + Padding**, click on the first icon to align the text to the left within the rectangle.



Font: Arial
Bold, 13
Corner Radius: 0
Opacity: 100 %
Padding: T 2 R 2 B 2 L 2
Line Spacing: --

10. Drag the **Text Field** widget and place it at coordinates (322,125) on the wireframe.
11. With the **Text Field** widget selected, perform the following steps:
    1. Type in `Enter Invoice or Description`. The text will appear in the **Text Field** widget.

2. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type in `SearchTextField`.
3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the fourth icon with an A to bring up the color chooser drop-down menu.
4. To change the color of the text, enter the hex value `CCCCCC` in the field and press *Enter*, or click anywhere outside the drop-down menu to select it.



12. Drag the **Button Shape** widget and place it at coordinates (508,125) on the wireframe.
13. With the **Button Shape** widget selected, perform the following steps:
    1. Type in `Search`. The text will appear in the **Button Shape** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **HTML Button Name** field and type in `SearchButton`.

    3. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and then click on **Create Link...**.
    4. In the **Sitemap** pop up, double-click on the page labeled **Invoice**.

14. Drag the **Rectangle** widget and place it at coordinates (10,155) on the wireframe.
15. With the **Rectangle** widget selected, perform the following steps:
    1. Type in `Orders # - ## of ####`. The text will appear in the middle of the **Rectangle** widget.
    2. In the toolbar, change the width **w:** to `600` and the height **h:** to `35`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `OrdersTotalSubhead`.
    4. In the **Widget Properties and Style** pane, click on the text formatting icon. Under **Alignment + Padding**, click on the first icon to align the text to the left within the rectangle.
16. Drag the **Label** widget and place it at coordinates (398,165) on the wireframe.
17. With the **Label** widget selected, perform the following steps:
    1. Type in `Previous  1 ... 11 12 13 14 15  Next`. The text will appear in the middle of the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape** field and type in `PaginationControls`.
    3. Add underscores to the text in the label by highlighting each individual string or number in the label.
    4. In the **Widget Properties and Style** pane, click on the **Style** tab and scroll to the **Font** section. Click on the third icon with a U to underline the selected substring. Repeat for each substring you would like to apply an underscore to.
18. Drag the **Table** widget and place it at coordinates (10,190) on the wireframe.
19. With the **Table** widget selected, perform the following steps:
    1. In the **Table** widget, right-click on the first row of the first column, and click on **Insert Column Left**. You should now have four columns.
    2. In the **Table** widget, right-click on the first row of the first column and click on **Insert Row Below**. You should now have four rows. Repeat to create a total of 10 rows.
    3. In the **Widget Interactions and Notes** pane, click on the **Table Name** field and type in `InvoiceInquiryTable`.
    4. Click on the first column of the first row of the **Table** widget to select and type in `Order Date`.

    5. In the **Widget Interactions and Notes** pane, click on the **Table Cell Name** field and type in `IITHeaderRowOrderDateCell`.
    6. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the first icon with a B to make the text bold within the rectangle. Under **Alignment + Padding**, click on the first icon to align the text to the left within the table cell.
    7. For each cell in the first row (that is, header row) of the table, type a table cell name and the correct column headings as follows: `Invoice`, `Description`, and `Invoice Total`.
    8. Click on the cell in the first column of the second row and type in `MM-DD-YYYY`.
    9. Click on the cell in the second row of the second column and type in `11-22223`.
    10. In the **Widget Interactions and Notes** pane, click on the **Table Cell Footnote and Name** field and type in `InvoiceRow2`.
    11. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and click on **Create Link...**.
    12. In the **Sitemap** pop up, click on the page labeled **Invoice**.
    13. Click on the cell in the first row of the third column and change the width **w:** to `300`.
    14. Click on the cell in the third column of the second row and type in `Example text about order`.
    15. Click on the cell in the fourth column of the second row, and type in `###,###.##`.

20. In the **Sitemap** pane, double-click on the page named **Invoice**.
21. Drag the **Placeholder** widget and place it at coordinates (10,13) on the wireframe.
22. With the **Placeholder** widget selected, perform the following steps:
    1. Type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CompanyLogo`.
23. Drag the **Label** widget and place it at coordinates (810,13) on the wireframe.
24. With the **Label** widget selected, perform the following steps:
    1. Type in `Invoice                  11-22223`. The text will display on the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `InvoiceTextPanel`.

   3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the first icon with a B to make the text bold within the rectangle. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.

25. Repeat step 24 to create two additional **Label** widgets. One at coordinates (810, 38) with the **Date   MM-DD-YYYY** text and a second **Label** widget at coordinates (810, 63) with the **Customer   1111111111** text.
26. Repeat step 25 to create one additional **Label** widget at coordinates (10, 120) with the text as follows:

    Sold To

    Name of Customer

    Division

    1111 Street Address

    Suite 222

    Anywhere, State 44444-4444

27. With the **Label** widget selected, in the toolbar, change the width **w:** to `180`.
28. Repeat step 25 to create one additional **Label** widget at coordinates (810, 100) with the text as follows:

    Ship To

    Name of Customer

    Division

    1111 Street Address

    Suite 222

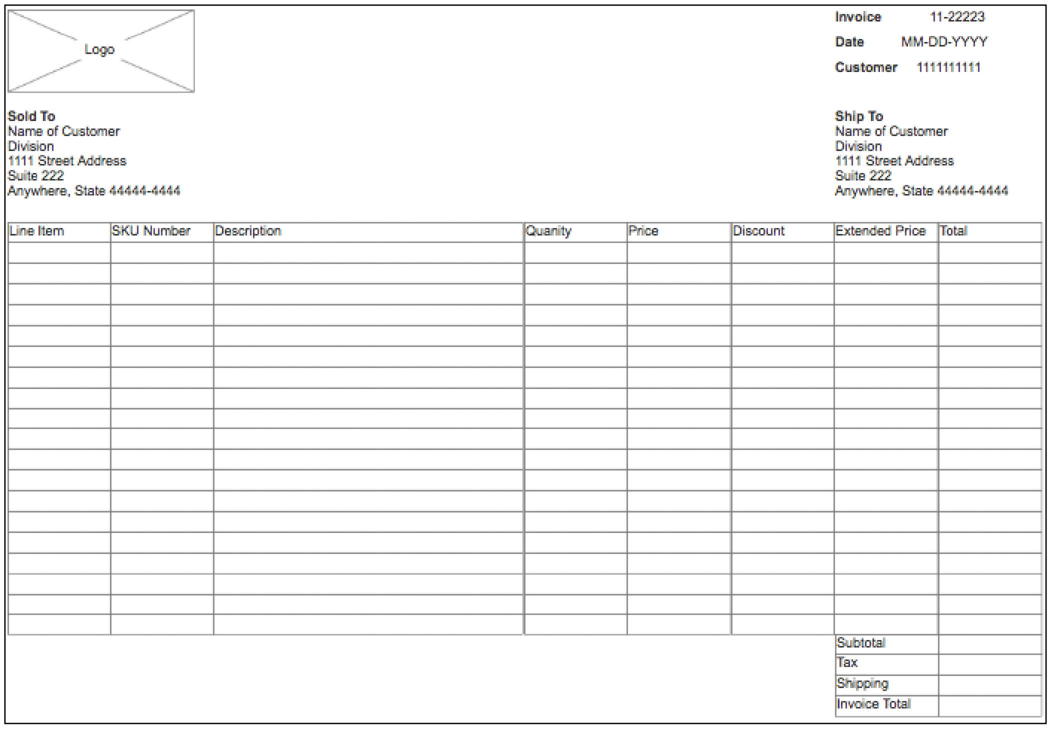
    Anywhere, State 44444-4444

29. With the **Label** widget selected, in the toolbar, change the width **w:** to `180`.
30. Drag the **Table** widget and place it at coordinates (10,219) on the wireframe.
31. With the **Table** widget selected, perform the following steps:
    1. In the **Table** widget, right-click on the first row of the first column, and click on **Insert Column Left**. You should now have four columns. Repeat this to create a total of eight columns.
    2. In the **Table** widget, right-click on the first row of the first column, and click on **Insert Row Below**. You should now have four rows. Repeat this to create a total of 20 rows.
    3. In the **Widget Interactions and Notes** pane, click on the **Table Name** field and type in `InvoiceTable`.

   4. Click on the first column of the first row of the **Table** widget and type in `Line Item`.
   5. Click on the second column of the first row of the **Table** widget and type in `SKU Number`.
   6. Click on the third column of the first row of the **Table** widget and type in `Description`. Change the width **w:** to `300`.
   7. Click on the fourth column of the first row of the **Table** widget to select and type in `Quantity`.
   8. Click on the fifth column of the first row of the **Table** widget and type in `Price`.
   9. Click on the sixth column of the first row of the **Table** widget and type in `Discount`.
   10. Click on the seventh column of the first row of the **Table** widget and type in `Extended price`.
   11. Click on the eighth column of the first row of the **Table** widget and type in `Total`.

32. Drag the **Table** widget and place it at coordinates (810,619) on the wireframe.
33. With the **Table** widget selected, perform the following steps:
   1. In the **Table** widget, right-click on the first row of the first column and click on **Delete Column(s)**. You should now have two columns.
   2. In the **Table** widget, right-click on the first row of the first column and click on **Insert Row Below**. You should now have four rows.
   3. In the **Widget Interactions and Notes** pane, click on the **Table Name** field and type in `InvoiceTotalTable`.
   4. Click on the first column of the first row of the **Table** widget and type in `Subtotal`.
   5. Click on the first column of the second row of the **Table** widget and type in `Tax`.
   6. Click on the first column of the third row of the **Table** widget and type in `Shipping`.
   7. Click on the first column of the fourth row of the **Table** widget and type in `Invoice Total`.
34. Save the file as `b2b_invoice_inquiry.rp`.

35. Click on the **Publish** button in the toolbar and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu and selecting **Publish** and clicking on **Generate HTML Files...**.



Invoice 11-22223
Date MM-DD-YYYY
Customer 1111111111

Sold To
Name of Customer
Division
1111 Street Address
Suite 222
Anywhere, State 44444-4444

Ship To
Name of Customer
Division
1111 Street Address
Suite 222
Anywhere, State 44444-4444

Line Item	SKU Number	Description	Quanity	Price	Discount	Extended Price	Total
						Subtotal	
						Tax	
						Shipping	
						Invoice Total	

How it works...

In this recipe, you created an Invoice Inquiry page and a sample Invoice page. You used the **Placeholder**, **Rectangle**, **Text Field** and **Table** widgets. On the Invoice Inquiry page, we linked the invoice number to the Invoice page.

There's more...

One additional point to consider while using the **Table** widget is that if you plan on having a header row in your table, you need to take this into account while assigning the row table properties. For example, if you would like to show the status of the last 10 orders, you would need to have 11 rows to allow for the column headers.

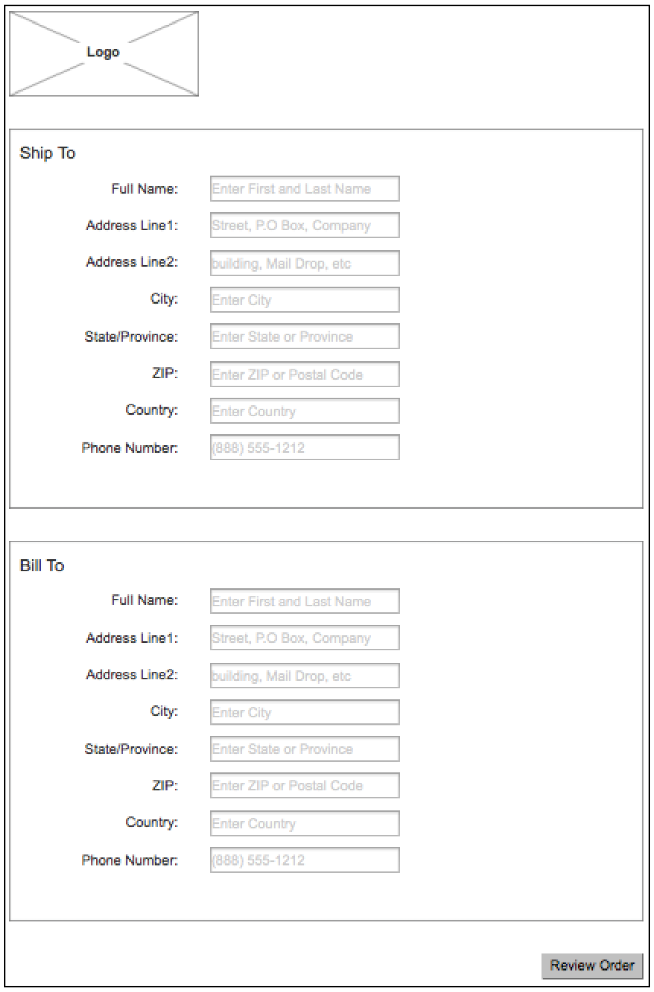
[If you find that you need to change the number of rows or columns in a table, right-click on the table to bring up a pop-up menu and select the desired option.]

A B2B order checkout

Business-to-business customers appreciate being able to quickly place an order. You must streamline the order checkout process to accomplish this task. In this recipe, you will create a B2B order checkout for business-to-business customers.

Getting ready

Our intention in this recipe is to focus on the optimized B2B order checkout flow. You will assume that the flow for account creation has already been defined, and you will only focus on order checkout. For this recipe, you will need the Axure RP file `b2b_invoice_inquiry.rp` created in the *A B2B invoice inquiry* recipe.



Ship To

Full Name:	Enter First and Last Name
Address Line1:	Street, P.O Box, Company
Address Line2:	building, Mail Drop, etc
City:	Enter City
State/Province:	Enter State or Province
ZIP:	Enter ZIP or Postal Code
Country:	Enter Country
Phone Number:	(888) 555-1212

Bill To

Full Name:	Enter First and Last Name
Address Line1:	Street, P.O Box, Company
Address Line2:	building, Mail Drop, etc
City:	Enter City
State/Province:	Enter State or Province
ZIP:	Enter ZIP or Postal Code
Country:	Enter Country
Phone Number:	(888) 555-1212

How to do it...

To create an optimized B2B order checkout flow, you will use the **Placeholder**, **Rectangle**, and **Text Field** widgets. Perform the following steps:

1. Start Axure and under **Create New**, select **RP Document**.

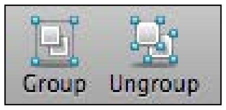
[If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.]

2. In the **Sitemap** pane, delete the pages labelled **Page 1**, **Page 2**, and **Page 3** by right-clicking on each page, and clicking on **Delete** in the pop-up menu.
3. In the **Sitemap** pane, right-click on the page labelled **Home**, click on **Rename** in the pop-up menu, and rename the page to `B2B Order Checkout`.
4. Import the page labeled **Invoice** from the Axure RP file `b2b_invoice_inquiry.rp`, which you created in the *A B2B invoice inquiry* recipe of this chapter. To accomplish this, perform the following steps:
    1. Click on **File** and then click on **Import from RP File**.
    2. Browse to the Axure RP file `b2b_invoice_inquiry.rp` and click on **Open**.
    3. Click on the checkbox next to the page labeled **Invoice**.
    4. Click on **Next** through the prompts and then click on **Finish**.
5. With the page labeled **Invoice** selected, right-click on the Add Page icon.
6. In the **Sitemap** pane, right-click on the page labeled **New Page 1**, click on **Rename** in the pop-up menu, and rename the page to `Confirmation`.
7. In the **Sitemap** pane, double-click on the page icon next to the page labeled **Confirmation**.
8. Drag the **Placeholder** widget and place it at coordinates (10,10) on the wireframe.
9. With the **Placeholder** widget selected, type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
10. With the **Placeholder** widget selected, in the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CompanyLogo`.
11. Drag the **Rectangle** widget and place it at coordinates (10,120) on the wireframe.
12. With the **Rectangle** widget selected, perform the following steps:
    1. Type in `Your order has been received. We appreciate your business. Thank you for your Order!`. The text will appear in the middle of the **Placeholder** widget.

   2. In the toolbar, change the width **w:** to `600` and the height **h:** to `360`.
   3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ConfirmationMessage`.

13. In the **Sitemap** pane, double-click on the page icon next to the page labeled **B2B Order Checkout**.
14. Drag the **Placeholder** widget and place it at coordinates (10,10) on the wireframe.
15. With the **Placeholder** widget selected, type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
16. With the **Placeholder** widget selected, click on the **Shape Name** field and type in `CompanyLogo` in the **Widget Interactions and Notes** pane.
17. You will now focus on building out the body of the B2B Order Checkout page. While holding down the mouse button, drag the **Rectangle** widget and place it at coordinates (10,120) on the wireframe.
18. With the **Rectangle** widget selected, perform the following steps:
    1. With the **Rectangle** widget selected, change the width **w:** to `600` and the height **h:** to `360` in the toolbar
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ShipToRectangle`
19. Drag the **Label** widget and place it at coordinates (20,135) on the wireframe.
20. With the **Label** widget selected, perform the following steps:
    1. Type in `Ship To`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ShipToH2Tag`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down list and clicking on **16** to select it.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
21. Drag the **Label** widget and place it at coordinates (20,170) on the wireframe. In the toolbar, change the width **w:** to `150`.
22. With the **Label** widget selected, perform the following steps:
    1. Type in `Full Name`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `FullNameLabel`.
    3. Under **Alignment + Padding**, click on the third icon to align the text to the right within the label.

23. Drag the **Text Field** widget and place it at coordinates (200,165) on the wireframe. In the toolbar, change the width **w:** to `180`.
24. With the **Text Field** widget selected, perform the following steps:
    1. Type in `Enter First and Last Name`. The text will appear in the **Text Field** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type in `FullNameField`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the fourth icon with an A to bring up the color chooser drop-down menu.
    4. To change the color of the text, enter the hex value `CCCCCC` in the field and press *Enter*, or click anywhere outside the drop-down menu to select it.
25. While holding down the *Shift* key, click on both, the label you dragged onto the wireframe in step 21 and the **Text Field** widget dragged onto the wireframe in step 23. In the toolbar, click on the Group icon.



26. Click on the group you created in step 25 and select **Copy** from the pop-up menu.
27. Click on the wireframe and select **Paste** from the pop-up menu.
28. Drag the new group to align under it the previous group.
29. Repeat steps 25 to 27 to create seven additional groups. You should now have a total of eight groups that comprise the label and text field. You should name the **Label** widgets and have the text displayed on the widget as follows: **Full Name**, **Address Line 1**, **Address Line 2**, **City**, **State/Providence**, **ZIP**, **Country**, and **Phone Number**. Name the **Text Field** widgets in each group and change the text displayed on the widget as appropriate (for example, for the **Address Line 1** label, the corresponding text field should display **Street, P.O. Box, Company**).
30. Select the **Rectangle** widget and label it with the **Ship To** text, and each of the eight groups.
31. In the toolbar, click on the Group icon.
32. Click on the group you created in step 31, and select **Copy** from the pop-up menu.
33. Click on the wireframe and select **Paste** from the pop-up menu.
34. Drag the new group to align it under the previous group.
35. Click on the label in the second rectangle and rename it from **Ship To** to `Bill To`.

36. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and rename it to `BillToH2Tag`.
37. Drag the **Button Shape** widget and place it at coordinates (513,900).
38. With the **Button Shape** widget selected, perform the following steps:
    1. Type in `Review Order`. The text will appear at the **Button Shape** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **HTML Button Name** field and type in `ReviewOrderButton`.
    3. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and click on **Create Link...**.
    4. In the **Sitemap** pop up, click on the page labeled **Invoice**.
39. Save the file as `b2b_order_checkout.rp`.
40. Click on the **Publish** button in the toolbar and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

In this recipe, you created an optimized B2B order checkout flow. You used the **Placeholder**, **Rectangle**, and **Text Field** widgets. By assigning the **OnClick** actions to the **Button Shape** widgets on each page, you linked the pages in the B2B order checkout flow.

Business-to-consumer (B2C) use cases

New e-commerce sites open daily by offering their goods and services directly to the customers. Many of these businesses have both an online and offline presence. Often a business-to-consumer customer will not have a solid idea of the exact item they are looking to purchase. As a result, business-to-consumer sites will use robust search as well as images and galleries to allow customers to browse and visualize product offerings.

How to do it...

In this recipe, you will explore a process to identify, visualize, and communicate product offerings for business-to-consumer use cases.

1. You will begin by understanding the needs of your business-to-consumer customer by defining requirements.
2. Use one-on-one interviews with current business-to-consumer customers to understand their unique and collective use cases. Always remember to ask open-ended questions in order to receive unbiased responses from your participants.

3. Use online surveys and questionnaires to collect primary data from your current business-to-consumer customers. One service that is free to try is `http://freeonlinesurveys.com`.
4. Conduct brainstorming sessions with your customers, sales, customer support, and other internal stakeholders.
5. Analyze all of the data collected. Leverage this data as input while creating an optimized initial draft of your requirements.
6. Review your initial requirement draft with stakeholders and other key decision makers.
7. Complete revisions to your requirement draft.
8. Iterate through steps 6 to 7 until you have a final requirements draft.
9. Obtain sign off of your final requirements draft.
10. Perform competitive analysis. Evaluate competitors within your industry as well as the best-in-class companies outside your industry.
11. Develop several user personas from the demographic data that is representative of typical current and future customers.
12. Create scenarios and storyboards using various user personas.
13. Develop use cases highlighting the interaction between the user and the proposed system features.
14. Create user flows that highlight the streamlined and optimized execution of your use cases.
15. Develop your information architecture by organizing these user flows into an optimized, logical, and efficient sitemap and navigation schema. Use Axure to perform the following steps:
    1. In the **Sitemap** pane, add/remove placeholder pages to communicate your page hierarchy and navigation schema
    2. Create a wireframe for each page shown in your **Sitemap**
16. Iterate through step 15 and gain approval from your stakeholders for the final sitemap and navigation schema.
17. You can now use Axure to create interactive prototypes to refine the proposed user experience (UX).
18. Conduct internal and external user studies to solicit feedback on your solution. One such service is `http://www.usertesting.com`.
19. Analyze the amount of time it takes for users to complete individual tasks while interacting with your prototype and make improvements.
20. Use Axure to create annotated wireframes and generate the specification document.
21. Share the specification document and optimized prototype with your stakeholders.

22. Incorporate feedback and gain final approval for the final wireframes and UX.
23. You can now focus on the visual design for your project.

How it works...

By using a user-centered design process, we keep the focus on our customers and optimize the solutions we create. This approach helps enhance the experience for our business-to-consumer customers, and as a result can lead to increased conversion rates.

There's more...

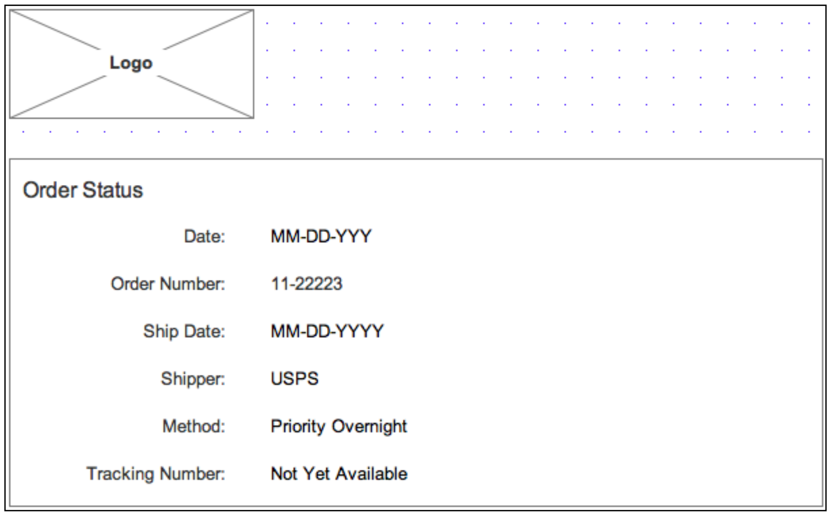
You will see that this recipe is very similar to the recipe for creating business-to-business use cases. The key difference between these two approaches is based on the different user goals and user objectives between the B2B and B2C customer. With a business-to-consumer customer, you will provide more opportunities for visualization of the product (for example, carousels, galleries, and feature images).

A B2C order status

Business-to-consumer customers expect to be able to track the order status. In this recipe, we will create a B2C **Order Status** page for business-to-consumer customers.

Getting ready

To create an optimized B2C **Order Status** page, you will use the **Placeholder**, **Rectangle**, and **Text Field** widgets, as shown in the following screenshot:



Order Status

Date:	MM-DD-YYY
Order Number:	11-22223
Ship Date:	MM-DD-YYYY
Shipper:	USPS
Method:	Priority Overnight
Tracking Number:	Not Yet Available

How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP Document**.

> If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. In the **Sitemap** pane, delete the pages labeled **Page 1**, **Page 2**, and **Page 3** by right-clicking on each page and clicking on **Delete** in the pop-up menu.
3. In the **Sitemap** pane, right-click on the page labeled **Home**, click on **Rename** in the pop-up menu, and rename the page to `Order Status`.
4. Drag the **Placeholder** widget and place it at coordinates (10,10) on the wireframe.
5. With the **Placeholder** widget selected, type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
6. With the **Placeholder** widget selected, in the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CompanyLogo`.
7. You will now focus on building out the body of the B2C Order Status page. Drag the **Rectangle** widget and place it at coordinates (10,120) on the wireframe.
8. With the **Rectangle** widget selected, perform the following steps:
    1. With the **Rectangle** widget selected, change the width **w:** to `600` and the height **h:** to `290` in the toolbar
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `OrderStatusRectangle`.
9. Drag the **Label** widget and place it at coordinates (20,135) on the wireframe.
10. With the **Label** widget selected, perform the following steps:
    1. Type in `Order Status`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `OrderStatusH2Tag`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down list and selecting **16**.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
11. Drag the **Label** widget and place it at coordinates (20,170) on the wireframe.
12. With the **Label** widget selected, perform the following steps:
    1. Type in `Date`. The text will appear in the **Label** widget.

    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `DateLabel`.
    3. Under **Alignment + Padding**, click on the third icon to align the text to the right within the label.

13. Drag the **Text Field** widget and place it at coordinates (200,165) on the wireframe.
14. With the **Text Field** widget selected, perform the following steps:
    1. Type in `MM-DD-YYYY`. The text will appear in the **Text Field** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type in `DateField`.
    3. In the **Widget Properties and Style** pane, scroll to the font section. Click on the fourth icon with an A to bring up the color chooser drop-down menu.
    4. To change the color of the text, enter the hex value `CCCCCC` in the field and press *Enter*, or click anywhere outside the drop-down menu to select it.
15. While holding down the *Shift* key, click on both, the labels you dragged onto the wireframe in step 11 and the text field you dragged onto the wireframe in step 13. In the toolbar, click on the Group icon.
16. Click on the group you created in step 15 and select **Copy** from the pop-up menu.
17. Click on the wireframe and select **Paste** from the pop-up menu.
18. While holding down the mouse button, drag the new group to align it under the previous group.
19. Repeat steps 16 to 18 to create five additional groups. You should now have a total of six groups that comprise a label and text field. You should name the **Label** widgets and have the text displayed on the widgets as follows: **Date**, **Order Number**, **Ship Date**, **Shipper**, **Method**, and **Tracking Number**. Name the **Text Field** widgets in each group and change the text displayed on the widget as appropriate (for example, for the **Order Number** label, the corresponding text field should display **11-222223**).
20. Save the file as `b2c_order_status.rp`.
21. Click on the **Publish** button in the toolbar and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and then clicking on **Generate HTML Files...**.

How it works...

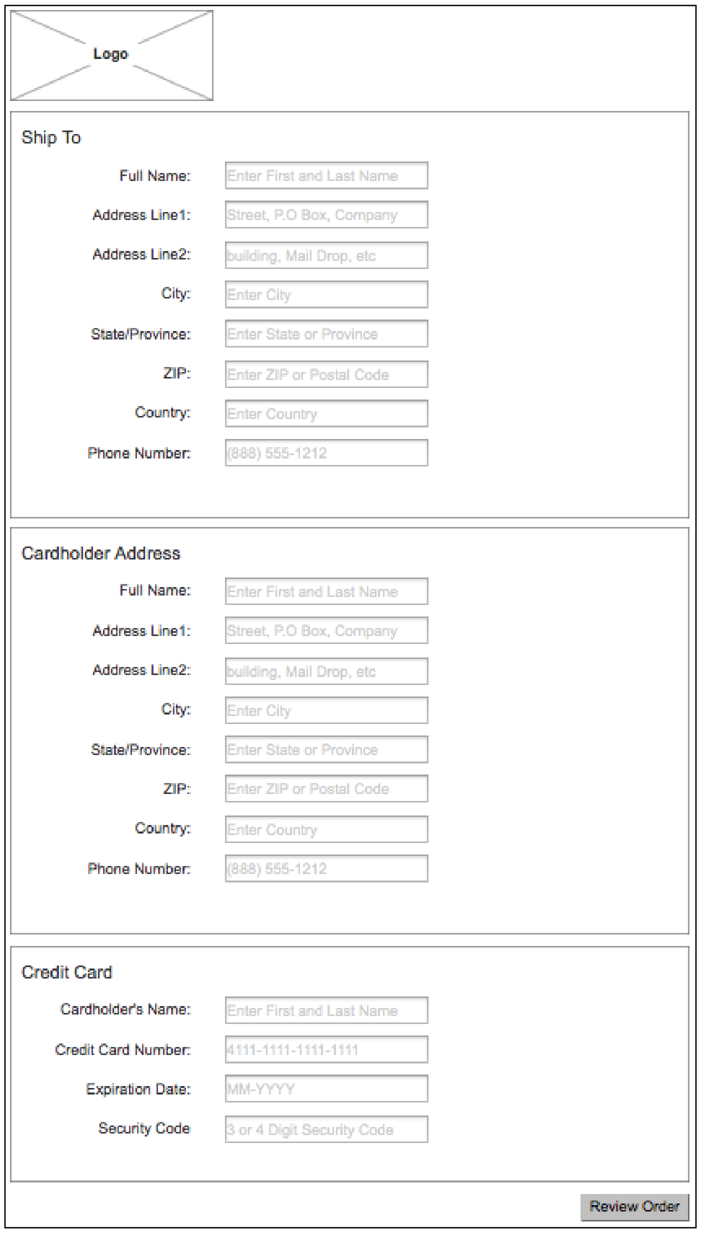
In this recipe, you created an optimized B2C **Order Status** page. You used the **Placeholder**, **Rectangle**, and **Text Field** widgets.

A B2C order checkout

In this recipe, you will create a **B2C Order Checkout** page for business-to-consumer customers.

Getting ready

For this recipe, you will need the Axure RP file `b2b_order_checkout.rp`, which was created in the *A B2B order checkout* recipe of this chapter.



Logo

Ship To

Full Name:	Enter First and Last Name
Address Line1:	Street, P.O Box, Company
Address Line2:	building, Mail Drop, etc
City:	Enter City
State/Province:	Enter State or Province
ZIP:	Enter ZIP or Postal Code
Country:	Enter Country
Phone Number:	(888) 555-1212

Cardholder Address

Full Name:	Enter First and Last Name
Address Line1:	Street, P.O Box, Company
Address Line2:	building, Mail Drop, etc
City:	Enter City
State/Province:	Enter State or Province
ZIP:	Enter ZIP or Postal Code
Country:	Enter Country
Phone Number:	(888) 555-1212

Credit Card

Cardholder's Name:	Enter First and Last Name
Credit Card Number:	4111-1111-1111-1111
Expiration Date:	MM-YYYY
Security Code	3 or 4 Digit Security Code

Review Order

How to do it...

You will open the `b2b_order_checkout.rp` file, rename pages, and add widgets to visualize credit card entry fields.

1. Start Axure and open `b2b_order_checkout.rp`.
2. In the **Sitemap** pane, right-click on the page labeled **B2B Order Checkout**, click on **Rename** in the pop-up menu, and rename the page to `B2C Order Checkout`.
3. You will now focus on adding a credit card section to the B2B Order Status page. Drag the **Rectangle** widget and place it at coordinates (10,900) on the wireframe.
4. With the **Rectangle** widget selected, perform the following steps:
    1. With the **Rectangle** widget selected, change the width **w:** to `600` and the height **h:** to `208` in the toolbar.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `BillToRectangle`.
5. Drag the **Label** widget and place it at coordinates (20,915) on the wireframe.
6. With the **Label** widget selected, perform the following steps:
    1. Type in `Credit Card`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CreditCardH2Tag`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down list and selecting **16**.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
7. Drag the **Label** widget and place it at coordinates (20,950) on the wireframe.
8. With the **Label** widget selected, perform the following steps:
    1. Type in `Cardholder's Name`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CardholdersNameLabel`.
    3. Under **Alignment + Padding**, click on the third icon to align the text to the right within the label.
9. Drag the **Text Field** widget and place it at coordinates (200,945) on the wireframe.
10. With the **Text Field** widget selected, perform the following steps:
    1. Type in `Enter First and Last Name`. The text will appear in the **Text Field** widget.

   2. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type in `CardholdersNameField`.
   3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the fourth icon with an A to bring up the color chooser drop-down menu.
   4. To change the color of the text, enter the hex value `CCCCCC` in the field and press *Enter*, or click anywhere outside the drop-down menu to select it.

11. While holding down the *Shift* key, click on both, the label you dragged onto the wireframe in step 7 and the text field dragged onto the wireframe in step 9. In the toolbar, click on the Group icon.
12. Click on the group you created in step 11, and select **Copy** from the pop-up menu.
13. Click on the wireframe and select **Paste** from the pop-up menu.
14. Drag the new group to align it under the previous group.
15. Repeat steps 12 to 14 to create three additional groups. You should now have a total of four groups that comprise of a label and text field. You should name the **Label** widgets and have the text displayed on the widgets as follows: **Cardholder's Name**, **Credit Card Number**, **Expiration Date**, and **Security Code**. Name the **Text Field** widgets in each group and change the text displayed on the widget as appropriate (for example, for the **Credit Card Number** label, the corresponding text field should display **4111-1111-1111-1111**).
16. Save the file as `b2c_order_checkout.rp`.
17. Click on the **Publish** button in the toolbar and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and then clicking on **Generate HTML Files...**.

How it works...

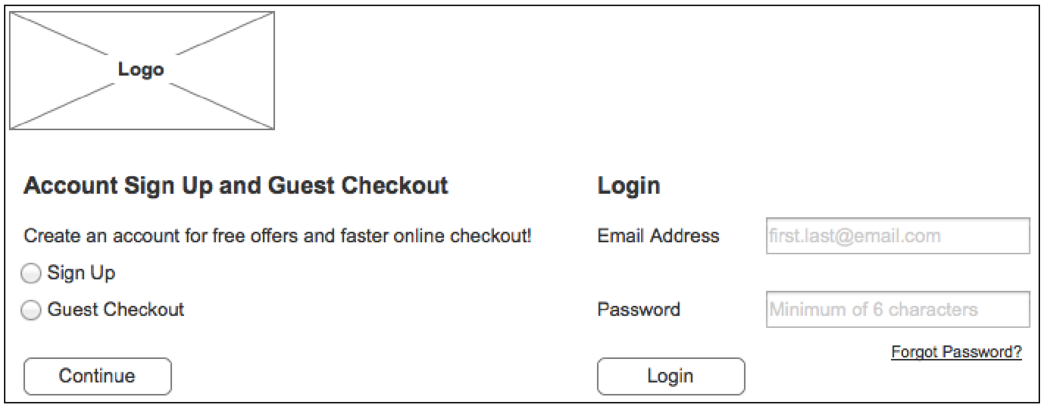
In this recipe, you created a B2C order checkout flow. You used the Axure RP file `b2b_order_checkout.rp` created in the *A B2B order checkout* recipe of this chapter. You added credit card entry fields to the **B2C Order Checkout** page by adding additional **Placeholder**, **Rectangle**, and **Text Field** widgets.

A Magento example

Magento is one of the most popular open source e-commerce applications in use today. You can learn more about Magento, the e-commerce platform, at `http://www.magentocommerce.com/`. In this recipe, you will create a Magento checkout flow.

Getting ready

For this recipe, you will need the `b2c_order_checkout.rp` file created in the *A B2C order checkout* recipe.



How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP Document**.

> If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. In the **Sitemap** pane, perform the following steps:
    1. Delete the pages labeled **Page 1**, **Page 2**, and **Page 3** by right-clicking on each page and clicking on **Delete** in the pop-up menu
    2. In the **Sitemap** pane, right-click on the page labeled **Home**, click on **Rename** in the pop-up menu, and rename the page to `Magento Account Sign Up and Login`
    3. Click on the Add Page icon
    4. Right-click on the page labeled **New Page 1**, click on **Rename** in the pop-up menu, and rename the page to `Magento Shipping Method`

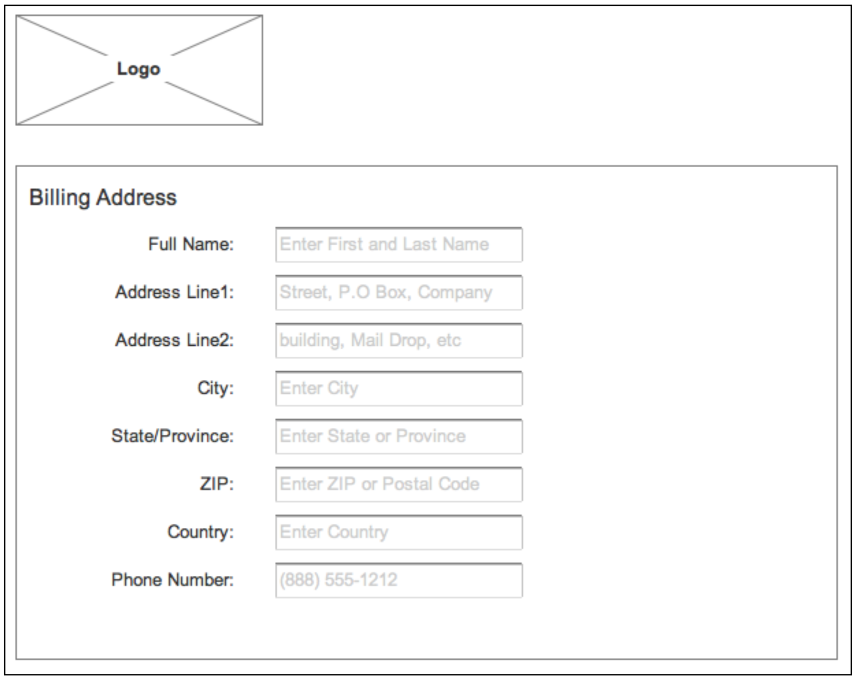
3. Next, you will import pages from the Axure RP file `b2c_order_checkout.rp`, which was created in the *A B2C order checkout* recipe of this chapter. To accomplish this, perform the following steps:
    1. Click on **File** and then click on **Import from RP File**. Browse to the Axure RP file `b2c_order_checkout.rp` and click on **Open**. Click on the checkboxes next to the pages labeled **B2C Order Checkout** and **Confirmation**. Click on **Next** through the prompts and then click on **Finish**.
    2. In the **Sitemap** pane, right-click on the page labeled **B2C Order Checkout**, click on **Rename** in the pop-up menu, and rename the page to `Magento Billing Address`.
    3. Perform steps 1 and 2 again and import the page labeled **B2C Order Checkout**, renaming the page to `Magento Shipping Address`.
    4. Perform steps 1 and 2 again and import the page labeled **B2C Order Checkout**, renaming the page to `Magento Payment Info`.
    5. In the **Sitemap** pane, organize the pages by clicking and dragging the pages in the following order: **Magento Account Sign Up and Login**, **Magento Billing Address**, **Magento Shipping Address**, **Magento Shipping Method**, **Magento Payment Info**, and **Confirmation**.
4. You will now focus on building the **Magento Account Sign Up and Login** page. In the **Sitemap** pane, double-click on the page icon next to the page labeled **Magento Account Sign Up and Login**.
5. Drag the **Placeholder** widget and place it at coordinates (10,10) on the wireframe. With the **Placeholder** widget selected, perform the following steps:
    1. With the **Placeholder** widget selected, type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
    2. With the **Placeholder** widget selected, in the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CompanyLogo`.
6. Drag the **Label** widget and place it at coordinates (20,120) on the wireframe. With the **Label** widget selected, perform the following steps:
    1. Type in `Account Sign Up and Guest Checkout`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `AccountSignUpH2Tag`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down list and selecting 16.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.

7. Drag the **Label** widget and place it at coordinates (410,120) on the wireframe.
8. With the **Label** widget selected, perform the following steps:
    1. Type in `Login`. The text will appear in the **Label** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `LoginH2Tag`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down list and selecting 16.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
9. Drag the **Label** widget and place it at coordinates (20,155) on the wireframe.
10. With the **Label** widget selected, perform the following steps:
    1. Type in `Create an account for free offers and faster online checkout!`. The text will appear in the **Label** widget.
    2. In the toolbar, change the width **w:** to `370` and the height **h:** to `15`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CreateAccountCopy`.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
11. Drag the **Label** widget and place it at coordinates (410,155) on the wireframe.
12. With the **Label** widget selected, perform the following steps:
    1. Type in `Email Address`. The text will appear in the **Label** widget.
    2. In the toolbar, change the width **w:** to `95` and the height **h:** to `15`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type `EmailLabel`.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
13. Drag the **Text Field** widget and place it at coordinates (525,150) on the wireframe.
14. With the **Text Field** widget selected, perform the following steps:
    1. Type in `first.last@email.com`. The text will appear in the **Text Field** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type in `EmailField`.
    3. In the **Widget Properties and Style** pane, scroll to the **Font** section. Click on the fourth icon with an A to bring up the color chooser drop-down menu.

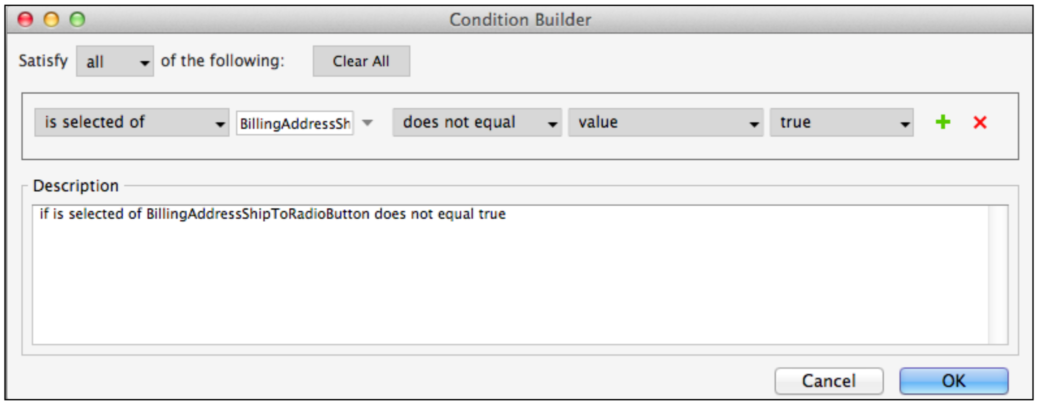
4. To change the color of the text, enter the hex value `CCCCCC` in the field and press *Enter*, or click anywhere outside the drop-down menu to select it.

15. Repeat steps 13 to 16 to create a **Password** label at (410, 205) showing **Password** as the text and **Password** text field at (525, 200) showing **Minimum of 6 Characters** as the text.
16. Drag the **Label** widget and place it at coordinates (610,235) on the wireframe.
17. With the **Label** widget selected, perform the following steps:
    1. Type in `Forgot Password?`. The text will appear in the **Label** widget.
    2. In the toolbar, change the width **w:** to `95` and the height **h:** to `12`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ForgotPasswordLink`.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
18. Drag the **Radio Button** widget and place it at coordinates (20,180) on the wireframe. With the **Radio Button** widget selected, perform the following steps:
    1. Type in `Sign up`. The text will appear in the **Radio Button** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Radio Button Name** field and type in `SignUpRadioButtonLabel`.
    3. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
19. Drag the **Radio Button** widget and place it at coordinates (20,205) on the wireframe. With the **Radio Button** widget selected, perform the following steps:
    1. In the toolbar, change the width **w:** to `135`.
    2. Type in `Guest Checkout`. The text will appear in the **Radio Button** widget.
    3. In the **Widget Interactions and Notes** pane, click on the **Radio Button Name** field and type in `GuestCheckoutRadioButtonLabel`.
    4. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
20. Hold down the *Shift* key, and click on each of the two **Radio Button** widgets created in steps 20 and 21. Perform the following steps:
    1. Right-click on any of the two **Radio Button** widgets you just selected. In the pop-up menu, click on **Assign Radio Group** from the flyout menu.
    2. In the **Selection Group** dialog box, type in `SignUpCheckout` in the **Group Name** field.
21. Drag the **Button Shape** widget and place it at coordinates (20,245) on the wireframe.

22. With the **Button Shape** widget selected, perform the following steps:
    1. Type in `Continue`. The text will appear in the **Button Shape** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **HTML Button Name** field and type in `ContinueButton`.
    3. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and then click on **Create Link...**.
    4. In the **Sitemap** pop up, click on the page labeled **Magento Billing Address**.
23. Drag the **Button Shape** widget and place it at coordinates (410,245) on the wireframe.
24. With the **Button Shape** widget selected, perform the following steps:
    1. Type in `Login`. The text will appear in the **Button Shape** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **HTML Button Name** field and type in `LoginButton`.
25. You will now focus on building the **Magento Billing Address** page. In the **Sitemap** pane, double-click on the page icon next to the page labeled **Magento Billing Address**.
26. With the **Magento Billing Address** page displayed in the **Wireframe** pane, delete all widgets on the page except the **Logo** and **Ship To** group. Rename the **Group** header from **Ship To** to `Billing Address`. Your wireframe should look like the following diagram:



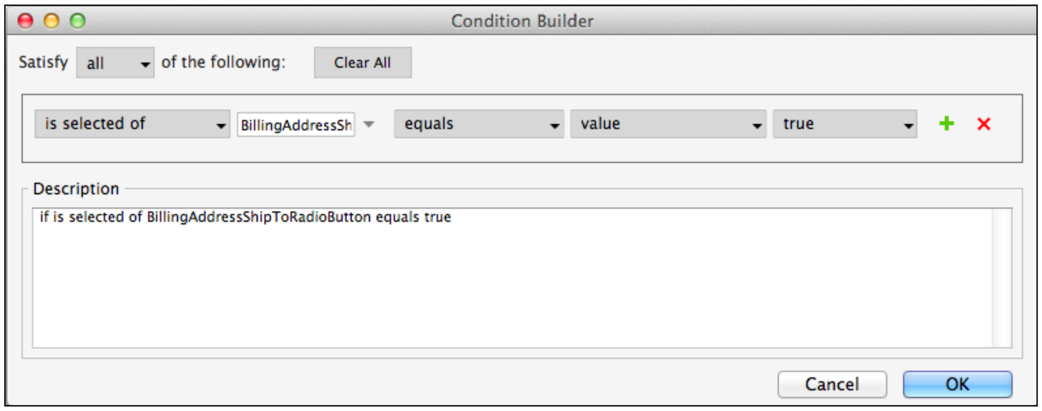
27. Drag the **Radio Button** widget and place it at coordinates (75,451) on the wireframe. With the **Radio Button** widget selected, perform the following steps:
    1. Type in `Billing Address for Ship To`. The text will appear in the middle of the **Radio Button** widget.
    2. In the toolbar, change the width **w:** to `207`.
    3. In the **Widget Interactions and Notes** pane, click on the **Radio Button Name** text field and type in `BillingAddressShipToRadioButton`.
28. Drag the **Button Shape** widget and place it at coordinates (410,441). With the **Button Shape** widget selected, perform the following steps:
    1. Type in `Continue`. The text will appear in the middle of the **Button Shape** widget.
    2. In the toolbar, change the width **w:** to `207`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `ContinueButton`.
    4. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and double-click on the **OnClick** interaction.
    5. In the **Case Editor** pop up, in **Case description**, rename **Case description** to `Do Not Use Billing Address for Ship To`.
    6. In the **Case Editor** pop up, click on the **Add Condition** button.
    7. In the **Condition Builder** pop up, click on each of the five drop-down menus to create the following condition: **if is selected of BillingAddressShipToRadioButton does not equal true**. Click on **OK**.



if is selected of BillingAddressShipToRadioButton does not equal true

8. In **Click to add actions**, click on **Open Link in New Window/Tab**.
9. In **Organize actions**, you will see the interaction description update to **Open Link in New Window/Tab**.

10. In **Configure Actions**, click on the radial button next to **Link to a page in this design**, and click on the page labeled **Magento Shipping Address**.
11. You will see the interaction description under the **Organize actions** option to include the page you selected. Click on **OK**.
12. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and double-click on the **OnClick** interaction.
13. In the **Case Editor** pop up, in **Case description**, rename the **Case description** to `Use Billing Address for Ship To`.
14. In the **Case Editor** pop up, click on the blue hyperlink labeled **Add Condition**.
15. In the **Condition Builder** pop up, click on each of the five drop-down menus to create the following condition: **if is checked of BillingAddressShipToRadioButton equals true**.



if is selected of BillingAddressShipToRadioButton equals true

16. In **Click to add actions**, click on **Open Link in New Window/Tab**.
17. In **Organize actions**, you will see the interaction description update to **Open Link in New Window/Tab**.
18. In **Configure Actions**, click on the radial button next to **Link to a page in this design**, and click on the page labeled **Magento Shipping Method**.
19. You will see the interaction description under the **Organize actions** update to include the page you selected. Click on **OK**.
20. With the **Button Shape** widget selected, type in `Continue`. The text will appear in the middle of the **Button Shape** widget.

29. You will now focus on building the **Magento Shipping Address** page. In the **Sitemap** pane, double-click on the page icon next to the page labeled **Magento Shipping Address**.

30. With the **Magento Billing Address** page displayed in the **Wireframe** pane, delete all widgets on the page except the **Logo** and **Ship To** group. Rename the group header from **Ship To** to `Shipping Address`.
31. Drag the **Button Shape** widget and place it at coordinates (410,441). With the **Button Shape** widget selected, perform the following steps:
    1. Type in `Continue`. The text will appear in the middle of the **Button Shape** widget.
    2. In the toolbar, change the width **w:** to `97`.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `ContinueButton`.
    4. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and click on **Create Link...**.
    5. In the **Sitemap** pop up, click on the page labeled **Magento Shipping Method**.
32. You will now focus on building the **Magento Shipping Method** page. In the **Sitemap** pane, double-click on the page icon next to the page labeled **Magento Shipping Method**.
33. With the **Magento Shipping Method** page displayed in the **Wireframe** pane, perform the following steps:
    1. Drag the **Placeholder** widget and place it at coordinates (10,10) on the wireframe.
    2. With the **Placeholder** widget selected, type in `Logo`. The text will appear in the middle of the **Placeholder** widget.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `CompanyLogo`.
    4. Drag the **Rectangle** widget and place it at coordinates (10,120) on the wireframe.
    5. In the toolbar, change the width **w:** to `600` and the height **h:** to `360`.
    6. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `ShipToRectangle`.
    7. Drag the **Label** widget and place it at coordinates (20,135) on the wireframe.
    8. With the **Label** widget selected, type in `Shipping Method`. The text will appear in the **Label** widget.
    9. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `ShippingMethodH2Tag`.
    10. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down and selecting 16.
    11. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.

12. Drag the **Radio Button** widget and place it at coordinates (20,170) on the wireframe.
13. Type in `Standard`. The text will appear in the **Radio Button** widget.
14. In the **Widget Interactions and Notes** pane, click on the **Radio Button Name** field and type in `StandardRadioButton`.
15. Under **Alignment + Padding**, click on the first icon to align the text to the left within the radio button label.
16. Drag the **Radio Button** widget and place it at coordinates (20,205) on the wireframe.
17. Type in `Two-Day Shipping`. The text will appear in the **Radio Button** widget.
18. In the **Widget Interactions and Notes** pane, click on the **Radio Button Name** field and type in `TwoDayShippingRadioButton`.
19. Under **Alignment + Padding**, click on the first icon to align the text to the left within the radio button label.
20. Drag the **Radio Button** widget and place it at coordinates (20,240) on the wireframe.
21. Type in `Next Day Air`. The text will appear in the **Radio Button** widget.
22. In the **Widget Interactions and Notes** pane, click on the **Radio Button Name** field and type in `NextDayAirRadioButton`.
23. Under **Alignment + Padding**, click on the first icon to align the text to the left within the radio button label.
24. Hold down the *Shift* key and click on each of the three **Radio Button** widgets you just created.
25. Right-click on any of the three **Radio Button** widgets you just selected. In the pop-up menu, click on **Assign Radio Group** from the flyout menu.
26. In the **Selection Group** dialog box, type in `ShippingMethod` in the **Group Name** field.
27. Drag the **Button Shape** widget and place it at coordinates (410,441).
28. With the **Button Shape** widget selected, type in `Continue`. The text will appear in the middle of the **Button Shape** widget.
29. In the **Widget Interactions and Notes** pane, click on the **HTML Button Footnote and Name** text field and type in `ContinueButton`.
30. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and click on **Create Link...**.
31. In the **Sitemap** pop up, click on the page labeled **Magento Payment Info**.

34. You will now focus on building the **Magento Payment Info** page. In the **Sitemap** pane, double-click on the page icon next to the page labeled **Magento Payment Info**.

35. With the **Magento Payment Info** page displayed in the **Wireframe** pane, delete all widgets on the page except the **Logo** and **Credit Card** group. Perform the following steps:
    1. In the toolbar, click on the first icon in **Selection Mode** to select **Select Intersected Mode**.
    2. Drag the wireframe selecting all of the widgets in the **Credit Card** group.
    3. In the toolbar, change **y:** to `120`. This will move up all of the widgets selected on the page.
    4. Drag the **Button Shape** widget and place it at coordinates (513,360).
    5. With the **Button Shape** widget selected, type in `Place Order`. The text will appear in the middle of the **Button Shape** widget.
    6. In the **Widget Interactions and Notes** pane, click on the **Shape Name** text field and type in `PlaceOrderButton`.
    7. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and then on **Create Link...**.
    8. In the **Sitemap** pop up, click on the page labeled **Confirmation**.
36. Save the file as `magento_order_checkout.rp`.
37. Click on the **Publish** button in the toolbar and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and then clicking on **Generate HTML Files...**.

How it works...

In this recipe, we created a Magento B2C order checkout flow. We used the **Placeholder**, **Rectangle**, **Label**, **Radio Button**, and **Text Field** widgets. By assigning **OnClick** actions to the **Button Shape** widgets on each page, we linked the pages in the Magento order checkout flow.

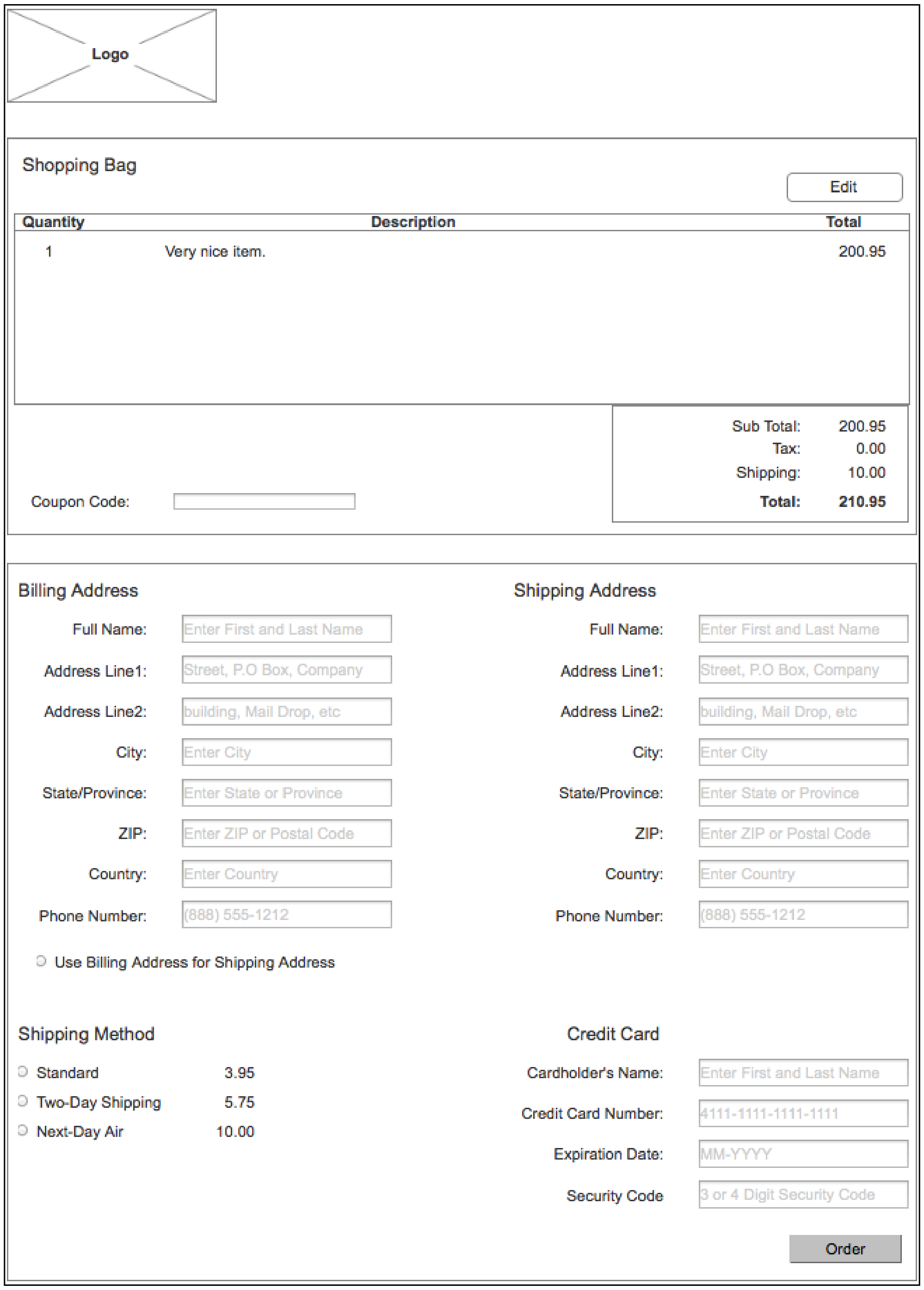
We also assigned a radio group to individual radio buttons, so only one of the radio buttons in the group would be selected at a time. We also checked the state of the **BillingAddressShipTo** radio button to determine which page would be next in our checkout flow.

A Zen Cart example

Zen Cart is another popular open source shopping cart solution. You can learn more about Zen Cart at `http://www.zen-cart.com/`. In this recipe, you will create a single page containing your shopping bag, shipping, and payment Zen Cart checkout flow.

Getting ready

For this recipe, you will need the `magento_order_checkout.rp` file created in the *A Magento example* recipe of this chapter.



Shopping Bag

Quantity	Description	Total
1	Very nice item.	200.95

Sub Total:	200.95
Tax:	0.00
Shipping:	10.00
Total:	**210.95**

Coupon Code:

Billing Address

Full Name:	Enter First and Last Name
Address Line1:	Street, P.O Box, Company
Address Line2:	building, Mail Drop, etc
City:	Enter City
State/Province:	Enter State or Province
ZIP:	Enter ZIP or Postal Code
Country:	Enter Country
Phone Number:	(888) 555-1212

Shipping Address

Full Name:	Enter First and Last Name
Address Line1:	Street, P.O Box, Company
Address Line2:	building, Mail Drop, etc
City:	Enter City
State/Province:	Enter State or Province
ZIP:	Enter ZIP or Postal Code
Country:	Enter Country
Phone Number:	(888) 555-1212

○ Use Billing Address for Shipping Address

Shipping Method

○ Standard	3.95
○ Two-Day Shipping	5.75
○ Next-Day Air	10.00

Credit Card

Cardholder's Name:	Enter First and Last Name
Credit Card Number:	4111-1111-1111-1111
Expiration Date:	MM-YYYY
Security Code	3 or 4 Digit Security Code

How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP Document**.

> If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document.

2. In the Sitemap pane, perform the following steps:
    1. Delete the pages labeled **Page 1**, **Page 2**, and **Page 3** by right-clicking on each page and clicking on **Delete** in the pop-up menu.
    2. Right-click on the page labeled **Home**, click on **Rename** in the pop-up menu, and rename the page to `Zen Cart Single Page Checkout`.
3. Import pages from the `magento_order_checkout.rp` Axure RP file created in the *A B2C order checkout* recipe of this chapter. To accomplish this, perform the following steps:
    1. Click on **File** and then **Import from RP File**. Browse to the `magento_order_checkout.rp` Axure RP file and click on **Open**.
    2. Click on the checkbox next to the pages labeled **Magento Billing Address**, **Magento Shipping Address**, **Magento Shipping Method**, **Magento Payment Info**, and **Confirmation**.
    3. Click on **Next** through the prompts and then click on **Finish**.
4. In the **Sitemap** pane, double-click on the **Zen Cart Single Page Checkout** page.
5. Drag the **Rectangle** widget and place it at coordinates (10,10) on the wireframe. Perform the following steps:
    1. Type in `Logo`. The text will appear in the middle of the **Rectangle** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CompanyLogo`.
6. You will now focus on building the **Shopping Bag** section of the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. Drag the **Rectangle** widget and place it at coordinates (10,120) on the wireframe.
    2. With the **Rectangle** widget selected, change the width **w:** to `780` and the height **h:** to `340` in the toolbar.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ShoppingBagRectangle`.

4. Drag the **Label** widget and place it at coordinates (23,135) on the wireframe.
5. With the **Label** widget selected, type in `Shopping Bag`. The text will appear in the **Label** widget.
6. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ShopingBagH1Tag`.
7. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the font size to 16 by clicking on the Font Size drop-down list and selecting **16**.
8. Under **Alignment + Padding**, click on the first icon to align the text to the left within the label.
9. Drag the **Button Shape** widget and place it at coordinates (678,150).
10. With the **Button Shape** widget selected, type in `Edit`. The text will appear on the **Button Shape** widget.
11. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `EditButton`.

7. You will now focus on building the shopping bag details of the **Shopping Bag** section for the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. Drag the **Rectangle** widget and place it at coordinates (16,185) on the wireframe.
    2. With the **Rectangle** widget selected, change the width **w:** to `767` and the height **h:** to `15` in the toolbar.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ShoppingBagHeader`.
    4. Drag the **Rectangle** widget and place it at coordinates (16,200) on the wireframe.
    5. With the **Rectangle** widget selected, change the width **w:** to `767` and the height **h:** to `150` in the toolbar.
    6. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ShoppingBagDetailsRectangle`.
    7. Drag the **Label** widget and place it at coordinates (23,185) on the wireframe.
    8. With the **Label** widget selected, change the width **w:** to `195` and the height **h:** to `15` in the toolbar.
    9. With the **Label** widget selected, type `Quantity`. The text will appear in the **Label** widget.
    10. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `QuantityHeaderCell`.
    11. Drag the **Label** widget and place it at coordinates (322,185) on the wireframe.

12. With the **Label** widget selected, change the width **w:** to `195` and the height **h:** to `15` in the toolbar.
13. With the **Label** widget selected, type in `Description`. The text will appear in the **Label** widget.
14. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `DescriptionHeaderCell`.
15. Drag the **Label** widget and place it at coordinates (712,185) on the wireframe.
16. With the **Label** widget selected, change the width **w:** to `72` and the height **h:** to `15` in the toolbar.
17. With the **Label** widget selected, type in `Total`. The text will appear in the **Label** widget.
18. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `TotalHeaderCell`.
19. Drag the **Label** widget and place it at coordinates (42,210) on the wireframe.
20. With the **Label** widget selected, change the width **w:** to `72` and the height **h:** to `15` in the toolbar.
21. With the **Label** widget selected, type in `1`. The text will appear in the **Label** widget.
22. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `QuantityRow1`.
23. Drag the **Label** widget and place it at coordinates (146,210) on the wireframe.
24. With the **Label** widget selected, change the width **w:** to `510` and the height **h:** to `15` in the toolbar.
25. With the **Label** widget selected, type in `A very nice item`. The text will appear in the **Label** widget.
26. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `DescriptionRow1`.
27. Drag the **Label** widget and place it at coordinates (692,210) on the wireframe.
28. With the **Label** widget selected, change the width **w:** to `72` and the height **h:** to `15` in the toolbar.
29. With the **Label** widget selected, type in `200.95`. The text will appear in the **Label** widget.
30. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `TotalRow1`.
31. Under **Alignment + Padding**, click on the third icon to align the text to the right within the label.
32. Drag the **Label** widget and place it at coordinates (30,425) on the wireframe.

33. With the **Label** widget selected, change the width **w:** to `117` and the height **h:** to `15` in the toolbar.
34. With the **Label** widget selected, type in `Coupon Code:`. The text will appear in the **Label** widget.
35. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `CouponCodeLabel`.
36. Drag the **Text Field** widget and place it at coordinates (153,425) on the wireframe.
37. In the **Widget Interactions and Notes** pane, click on the **Text Field Name** field and type in `CouponCodeField`.

8. You will now focus on building the shopping bag total of the **Shopping Bag** section for the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. Drag the **Rectangle** widget and place it at coordinates (529,350) on the wireframe.
    2. With the **Rectangle** widget selected, change the width **w:** to `254` and the height **h:** to `100` in the toolbar.
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ShoppingBagTotalRectangle`.
    4. Drag the **Label** widget and place it at coordinates (496,360) on the wireframe.
    5. With the **Label** widget selected, change the width **w:** to `195` and the height **h:** to `15` in the toolbar.
    6. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `SubTotalLabel`.
    7. Under **Alignment + Padding**, click on the third icon to align the text to the right within the label.
    8. Drag the **Label** widget and place it at coordinates (692,360) on the wireframe.
    9. With the **Label** widget selected, change the width **w:** to `72` and the height **h:** to `15` in the toolbar.
    10. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `SubTotalAmount`.
    11. Under **Alignment + Padding**, click on the third icon to align the text to right within the label.
    12. You will now create a group with the **SubTotalLabel** label and **SubTotalAmount** text field. Hold down the *Shift* key and click on the **SubTotalLabel** and **SubTotalAmount** widgets. In the toolbar, click on the Group icon.

13. Right-click on the group you created in the previous step and select **Copy** from the pop-up menu. Click on the wireframe and select **Paste** from the pop-up menu. Drag the new group to align it under the previous group.
14. Repeat the previous step to create two additional groups. You should now have a total of four groups that comprise two labels side by side. Use the following table as a reference for the placement of the groups, text to display, and shape names for the text widgets:

Coordinates	Text Displayed	Shape Name
(496,360)	Sub Total:	`SubTotalLabel`
(692,360)	200.95	SubTotalAmount
(496,380)	Tax:	TaxLabel
(692,380)	0.00	TaxAmount
(496,400)	Shipping:	ShippingLabel
(692,400)	10.00	ShippingAmount
(496,425)	Total	TotalLabel
(692,425)	210.95	TotalAmount

15. Drag the **Horizontal Line** widget and place it at coordinates (621,415) on the wireframe.
16. With the **Horizontal Line** widget selected, change the width **w:** to `146` in the toolbar.

9. You will now focus on building the consolidated checkout rectangle of the consolidated checkout section for the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. Drag the **Rectangle** widget and place it at coordinates (10,485) on the wireframe
    2. With the **Rectangle** widget selected, change the width **w:** to `780` and the height **h:** to `615` in the toolbar
    3. In the **Widget Interactions and Notes** pane, click on the **Shape Name** field and type in `ConsolidatedCheckOutRectangle`.

10. You will now focus on building the billing address of the consolidated checkout section for the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. In the **Sitemap** pane, double-click on the **Magento Billing Address** page.
    2. Click on the **Rectangle** widget in the wireframe, and click on the Ungroup icon in the toolbar.
    3. Click on the **Rectangle** widget in the wireframe and press the *Delete* key. Delete the **Logo** placeholder and the **Continue** button.

   4. You will now create a Billing Address group. In the main menu, click on **Edit** and then click on **Select All**. In the toolbar, click on the Group icon.
   5. In the main menu, click on **Copy**.
   6. In the **Sitemap** pane, double-click on the **Zen Cart Single Page Checkout** page.
   7. In the main menu, click on **Paste**. Click on the Billing Address group to select.
   8. With the Billing Address group selected, change **x:** to `20` and **y:** to `500` in the toolbar.

11. You will now focus on building the shipping address of the consolidated checkout section for the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. In the **Sitemap** pane, double-click on the **Magento Shipping Address** page.
    2. Click on the **Rectangle** widget in the wireframe, and click on the Ungroup icon in the toolbar.
    3. Click on the **Rectangle** widget in the wireframe and press the *Delete* key. Delete the **Logo** placeholder and the **Continue** button.
    4. You will now create a Shipping Address group. In the main menu, click on **Edit** and click on **Select All**. In the toolbar, click on the Group icon.
    5. In the main menu, click on **Copy**.
    6. In the **Sitemap** pane, double-click on the **Zen Cart Single Page Checkout** page.
    7. In the main menu, click on **Paste**. Click on the Shipping Address group to select.
    8. With the Shipping Address group selected, change **x:** to `420` and **y:** to `500` in the toolbar.

12. You will now focus on building the shipping method of the consolidated checkout section for the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. In the Sitemap pane, double-click on the **Magento Shipping Method** page.
    2. Click on the **Rectangle** widget in the wireframe, and click on the Ungroup icon in the toolbar.
    3. Click on the **Rectangle** widget in the wireframe and press the *Delete* key. Delete the **Logo** placeholder and the **Continue** button.
    4. You will now create a Shipping Method group. In the main menu, click on **Edit** and then click on **Select All**. In the toolbar, click on the Group icon.
    5. In the main menu, click on **Copy**.
    6. In the **Sitemap** pane, double-click on the **Zen Cart Single Page Checkout** page.

   7. In the main menu, click on **Paste**. Click on the Shipping Address group to select it.
   8. With the Shipping Address group selected, change **x:** to `20` and **y:** to `880` in the toolbar.

13. You will now focus on building the credit card form of the consolidated checkout section for the **Zen Cart Single Page Checkout** page. Perform the following steps:
    1. In the **Sitemap** pane, double-click on the **Magento Payment Info** page.
    2. Click on the **Rectangle** widget in the wireframe, and click on the Ungroup icon in the toolbar.
    3. Click on the **Rectangle** widget in the wireframe and press the *Delete* key. Delete the **Logo** placeholder and the **Continue** button.
    4. You will now create a Credit Card group. In the main menu, click on **Edit** and click on **Select All**. In the toolbar, click on the Group icon.
    5. In the main menu, click on **Copy**.
    6. In the **Sitemap** pane, double-click on the **Zen Cart Single Page Checkout** page.
    7. In the main menu, click on **Paste**. Click on the Shipping Address group to select.
    8. With the Shipping Address group selected, change **x:** to `420` and **y:** to `880` in the toolbar.
14. Drag the **Button Shape** widget and place it at coordinates (681,1061).
    1. With the **Button Shape** widget selected, type in `Order`. The text will appear on the **Button Shape** widget.
    2. In the **Widget Interactions and Notes** pane, click on the **Interactions** tab and click on **Create Link...**.
    3. In the **Sitemap** pop up, click on the page labeled **Confirmation**.
15. In the **Sitemap** pane, delete the pages labeled **Magento Billing Address**, **Magento Shipping Address**, **Magento Shipping Method**, and **Magento Payment Info**.
16. Save the file as `zen_cart_example.rp`.
17. Click on the **Publish** button in the toolbar and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, selecting **Publish**, and clicking on **Generate HTML Files...**.

How it works...

In this recipe, we created a **Zen Cart Checkout** page. We used the **Placeholder**, **Rectangle**, **Label**, **Radio Button**, and **Text Field** widgets. We also assigned a radio group to individual radio buttons, so only one of the radio buttons in the group would be selected at a time. By assigning the **OnClick** actions to the **Button Shape** widget, we linked **Zen Cart Single Page Checkout** to the **Confirmation** page.

6
Using Axure in Teams

In this chapter, you will discuss using Axure in teams and additional ways to streamline workflows for your organization. You will learn about the following topics:

- Leveraging the team project basics
- Creating a new shared project
- Maximizing collaboration through shared projects
- Checking in and checking out – making changes in a shared project
- Managing shared projects
- Using the team project history
- Exporting a shared project
- Configuring subversion on an Amazon EC2 instance

Introduction

You now know that Axure is a powerful tool for communicating design, prototyping, and generating specifications. What you may not know is that Axure can also empower team collaboration through the use of shared projects. We will cover quick recipes to discuss team project basics, creating a new shared project, how to check in and check out a shared project, collaborating with shared projects, as well as setting up your own **Version Control System** (for example, the **Apache Subversion** (**SVN**) server).

[The shared projects for teams feature and the recipes outlined in this chapter are only available with Axure RP Pro.]

Leveraging the team project basics

Shared projects enable you to leverage the power of team collaboration using a single Axure shared project. Behind the scenes, Axure uses Apache Subversion to maintain version control of each individual project. This allows individual team members to access the same team project concurrently while other team members are making revisions as well. Histories of all revisions are stored, and team members can review the revision history at any time.

How to do it...

For this recipe, you will be provided with an overview of how shared projects work.

Briefly review the following steps:

1. Identify which members in the team will be working on the shared project.
2. Ensure that all members of the team have access to the shared drive or SVN repository that you plan to host the shared directory on. This shared directory will contain the shared project.

   When Axure saves a shared project, the file will have a `.rpprj` file extension indicating that the file is a team project and not a standalone Axure RP file.

3. Typically, a team member will work with two copies of a shared project. One copy will be stored locally on their machine, and the other copy will be stored on a shared drive or in an SVN repository.
4. To open a local copy of a shared project, start Axure and select an Axure team project with the `.rpprj` file extension listed under **Open Recent**, or click on **Open** and browse to the team project stored locally. You can also use your finder or file explorer, browse to the team project file with the `.rpprj` file extension, and double-click to open it.
5. Once the team project is opened, you will check out the page or pages in the sitemap that you wish to edit. This prevents others from checking out the page(s) you are currently editing.
6. Perform your edits and then click on **Check In** to check in the page(s) that contain your modifications back into the shared project. During the check in process, your changes will be updated in the shared project.
7. Now, other members of the team can use the **Get Changes** option to sync changes saved to the team project with their local copy.
8. Other members of the team can now also check out and edit the page(s) you just checked in to the shared project.

How it works...

In this recipe, we provided a couple of key points to consider and a high-level overview of how shared projects work. As mentioned, shared projects in Axure leverage Apache Subversion. A lot of the concepts mentioned in this recipe will be familiar to you if you have had exposure to version control systems in the past.

If you have limited exposure to version control systems, do not be concerned. Just keep in mind that to edit a page or a Master, you will have to first check out the page or Master, make your edits, and then check in the new version of the page or Master. Axure and the shared projects feature will manage the check in/check out process, history, and versioning for you. It really is that easy.

There's more...

Apache Subversion is an open source project and is freely distributed under the Apache license. You can learn more about Apache Subversion at `http://subversion.apache.org/`.

Creating a new shared project

You have already identified the team members who will be working on your project. You are now ready to create the new shared project.

Getting ready

For this recipe, all you will need is a network drive or share mapped to your computer.

> If you do not have a network drive or share mapped to your computer, you can store the team project locally.

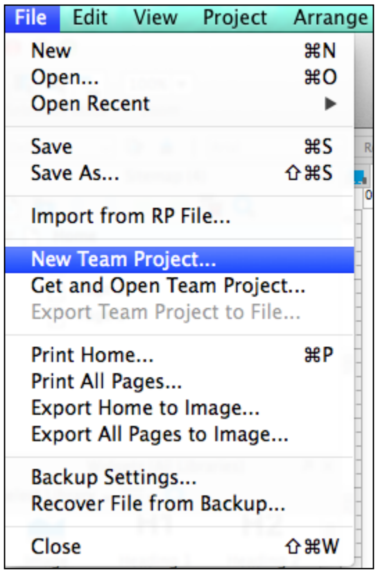
How to do it...

Perform the following steps:

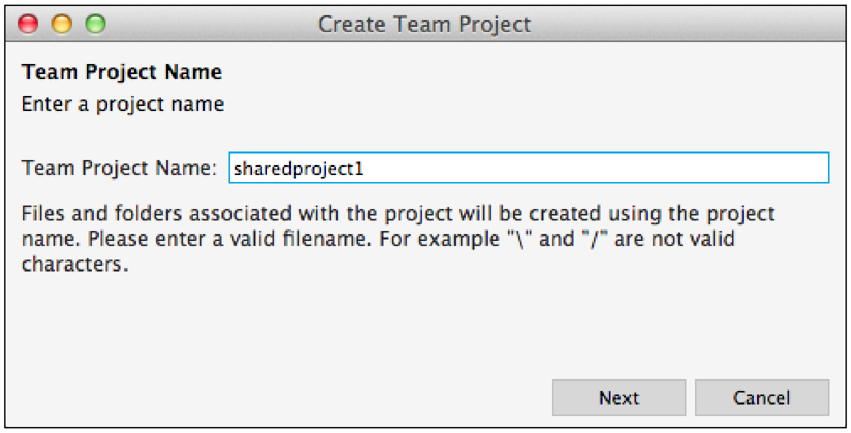
1. Start Axure and under **Create New**, select **RP File** or **Open Recent**.

> If you chose to click on **Open Recent** for this recipe, select an Axure file with the `.rp` extention. If you already have Axure open, select **File** in the main menu and then click on **New** in the drop-down menu to create a new RP document, or click on **Open** in the drop-down menu to open an existing RP document.

2. In the main menu, click on **File** and then on **New Team Project...**as shown in the following screenshot:



3. In the **Create Team Projec**t dialog box, type the name of your team project in the **Team Project Name:** field. Click on the **Next** button as shown in the following screenshot:



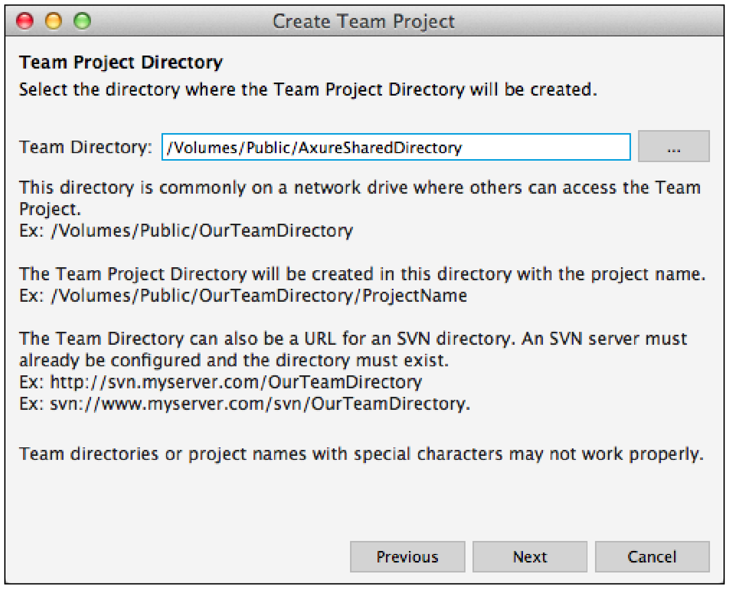
Team Project Name

Enter a project name

Team Project Name: sharedproject1

Files and folders associated with the project will be created using the project name. Please enter a valid filename. For example "\" and "/" are not valid characters.

4. In the **Create Team Project** dialog box, type the name of your team project directory in the **Team Directory:** field, or use the **...** button to browse the directory you wish to use as your team project directory. Click on the **Next** button as shown in the following screenshot:



Team Project Directory

Select the directory where the Team Project Directory will be created.

Team Directory: /Volumes/Public/AxureSharedDirectory

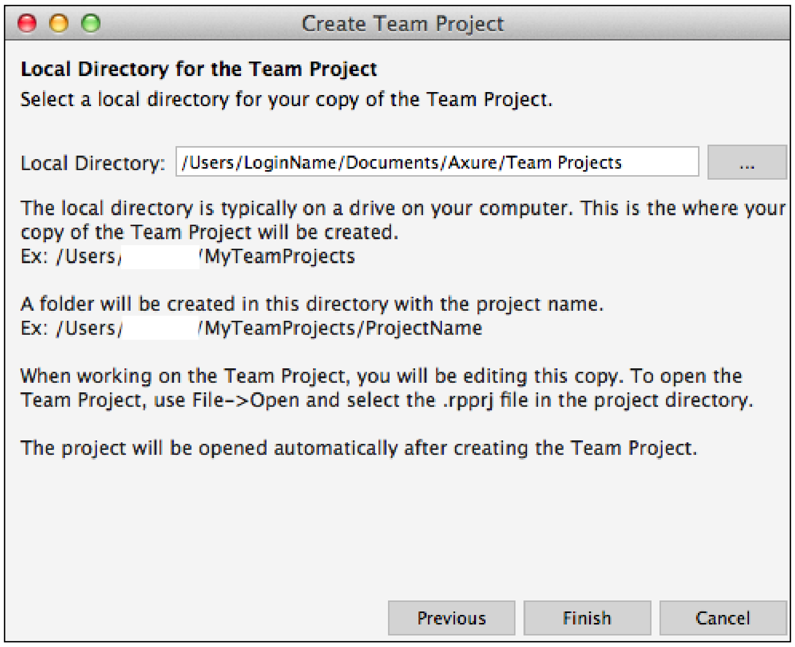
This directory is commonly on a network drive where others can access the Team Project.
Ex: /Volumes/Public/OurTeamDirectory

The Team Project Directory will be created in this directory with the project name.
Ex: /Volumes/Public/OurTeamDirectory/ProjectName

The Team Directory can also be a URL for an SVN directory. An SVN server must already be configured and the directory must exist.
Ex: http://svn.myserver.com/OurTeamDirectory
Ex: svn://www.myserver.com/svn/OurTeamDirectory.

Team directories or project names with special characters may not work properly.

5. In the **Create Team Project** dialog box, type the name for the local directory for the team project in the **Local Directory:** field, or use the **...** button labeled to browse to the directory you wish to use as your local directory. Click on the **Finish** button as shown in the following screenshot:



Local Directory for the Team Project

Select a local directory for your copy of the Team Project.

Local Directory: /Users/LoginName/Documents/Axure/Team Projects

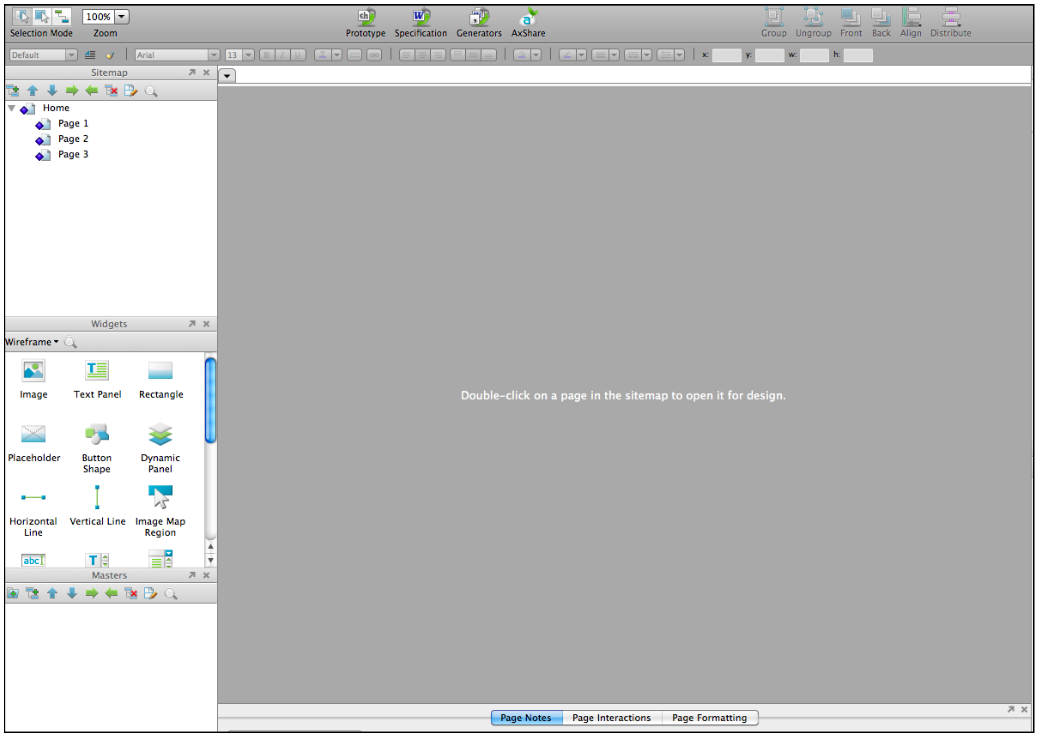
The local directory is typically on a drive on your computer. This is the where your copy of the Team Project will be created.
Ex: /Users/ /MyTeamProjects

A folder will be created in this directory with the project name.
Ex: /Users/ /MyTeamProjects/ProjectName

When working on the Team Project, you will be editing this copy. To open the Team Project, use File->Open and select the .rpprj file in the project directory.

The project will be opened automatically after creating the Team Project.

6. While the team project is being created, a **Status** dialog box will be displayed with a progress bar.
7. Once the team project has been created, a **Success** dialog box will be shown. Click on **OK** to close the **Success** dialog box.
8. Your team project will now open in Axure as shown in the following screenshot:



How it works...

For this recipe, you first decided to create a new Axure RP file or open an existing one. From the main menu, you selected **File** and then **New Team Project ...**. While clicking through the **Create Team Project** dialog boxes, you provided a team project name, shared directory, and local directory. Axure created the team project in your shared directory and created a local copy of the team project in the local directory with the team project name having a `.rpprj` file extention. The local copy of the team project opened automatically after the team project directory was created.

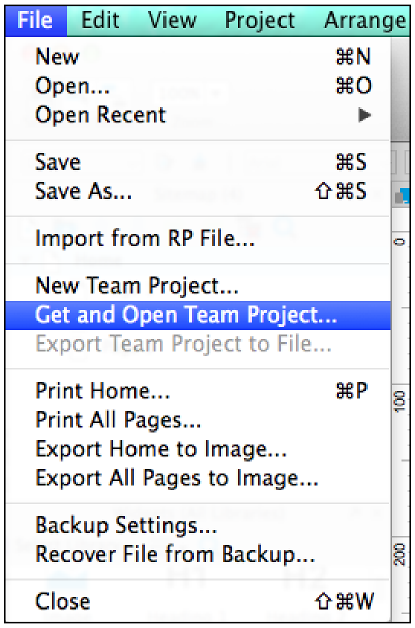
Maximizing collaboration through shared projects

You now know that shared projects enable us to leverage the power of team collaboration using a single Axure shared project. Collaboration for shared projects is enabled through individual team members accessing a local copy of the team project while other team members are making revisions to their individual copies of the same team project as well. This recipe will take you through creating a local copy of a shared project.

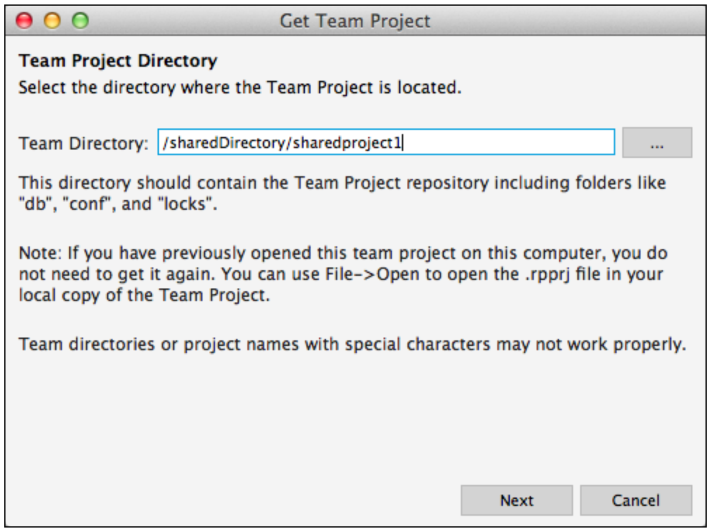
How to do it...

Perform the following steps:

1. Start Axure and under **Create New**, select **RP File**.
2. In the main menu, click on **File**, and click on **Get and Open Team Project...** as shown in the following screenshot:



3. In the **Get Team Project** dialog box, type the name of your team project directory in the **Shared Directory:** field, or use the button labeled with **...** to browse to the directory you wish to use as your team project directory. Click on the **Next** button as shown in the following screenshot:



Team Project Directory

Select the directory where the Team Project is located.

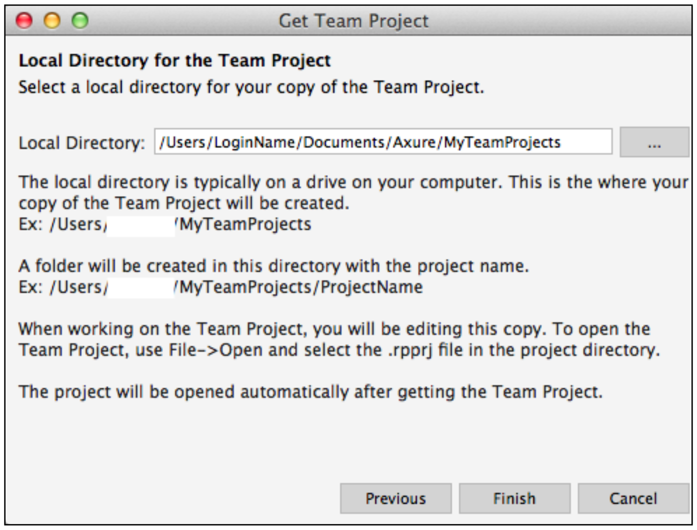
Team Directory: /sharedDirectory/sharedproject1

This directory should contain the Team Project repository including folders like "db", "conf", and "locks".

Note: If you have previously opened this team project on this computer, you do not need to get it again. You can use File->Open to open the .rpprj file in your local copy of the Team Project.

Team directories or project names with special characters may not work properly.

4. In the **Get Team Project** dialog box, type the name of the local directory for the team project in the **Local Directory:** field, or use the button labeled with **...** to browse to the directory you wish to use as your local directory. The local directory you specify should be empty or should not exist. Click on the **Finish** button as shown in the following screenshot:



Local Directory for the Team Project

Select a local directory for your copy of the Team Project.

Local Directory: /Users/LoginName/Documents/Axure/MyTeamProjects

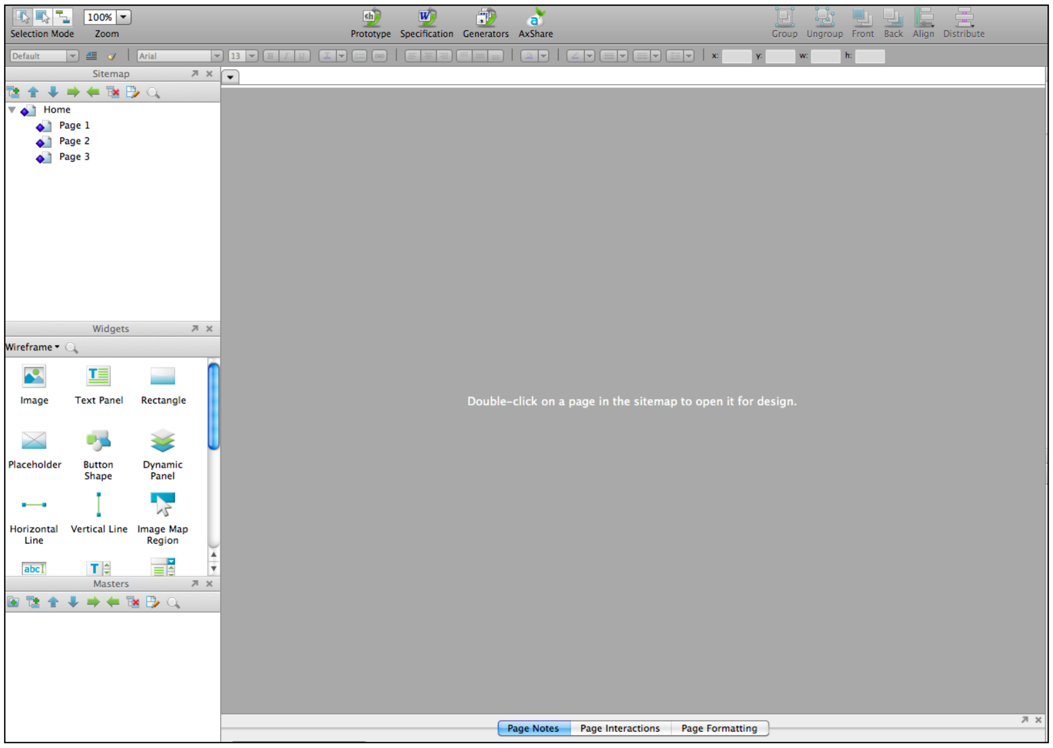
The local directory is typically on a drive on your computer. This is the where your copy of the Team Project will be created.
Ex: /Users/ /MyTeamProjects

A folder will be created in this directory with the project name.
Ex: /Users/ /MyTeamProjects/ProjectName

When working on the Team Project, you will be editing this copy. To open the Team Project, use File->Open and select the .rpprj file in the project directory.

The project will be opened automatically after getting the Team Project.

5. If the local directory does not exist, Axure will prompt an **Alert** dialog box. Click on the **Yes** button to create the directory.
6. Once the team project has been created, a **Success** dialog box will be shown. Left-click on **OK** to close the **Success** dialog box.
7. Your team project will now open in Axure.



How it works...

Collaboration for shared projects works by allowing team members to work on local copies of their team project while other team members are making revisions to their individual local copies. In this recipe, you created a local copy of a team project which is stored in a local `.rpprj` file.

There's more...

If you plan on working on multiple computers, each computer should have its own local copy of the shared project. Just complete this recipe on each computer that you will be using to ensure a local copy of the team project is available when you need it.

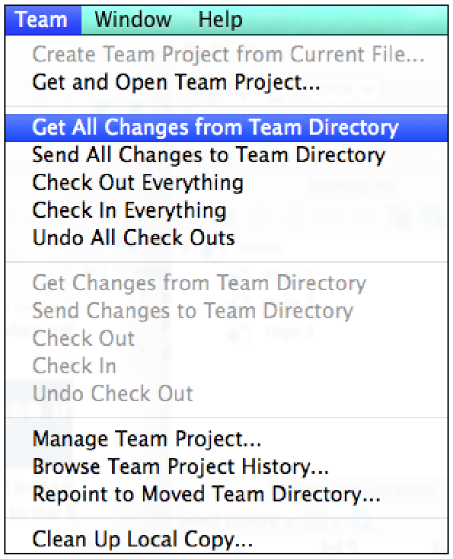
Checking in and checking out – making changes in a shared project

You will now learn how to get changes from the team project and save changes to your local copy of the shared project. You will then send changes to the shared project.

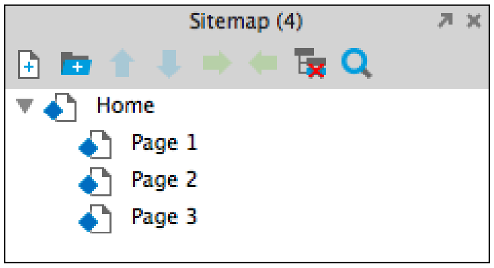
How to do it...

Perform the following steps:

1. To open a local copy of a shared project, start Axure and select an Axure team project with the `.rpprj` file extension listed under **Open Recent**, or select **Open** and browse to the team project stored locally. You can also use your finder or file explorer, browse to the team project file with the `.rpprj` file extension, and double-click to open.
2. Once the team project opens, in the main menu, left-click on **Share** and then on **Get All Changed from Team Directory** as shown in the following screenshot:

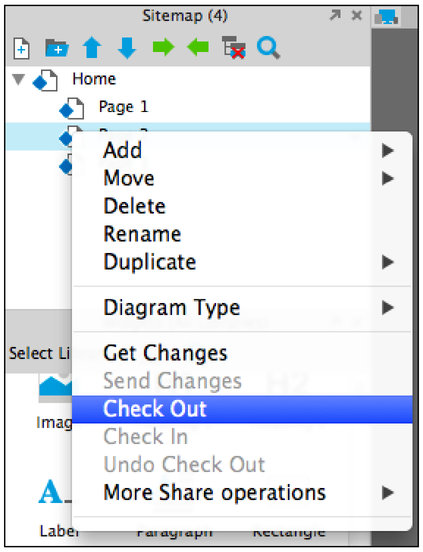

Team
Window
Help
Create Team Project from Current File...
Get and Open Team Project...
Get All Changes from Team Directory
Send All Changes to Team Directory
Check Out Everything
Check In Everything
Undo All Check Outs
Get Changes from Team Directory
Send Changes to Team Directory
Check Out
Check In
Undo Check Out
Manage Team Project...
Browse Team Project History...
Repoint to Moved Team Directory...
Clean Up Local Copy...

3. A progress dialog box will display showing a progress indicator while all changes are being synchronized from the shared directory to your local copy. Once the progress dialog box closes automatically, your local copy will contain the latest changes from the shared directory.
4. In the **Sitemap** pane, you will see that there are team project icons in the lower-left corner of each page, as shown in the following screenshot:



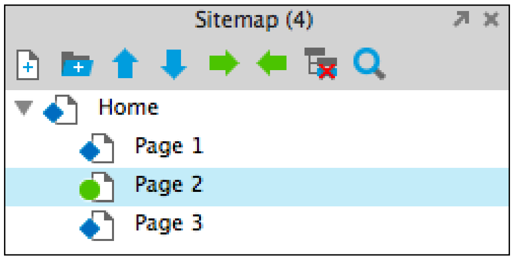
- A blue diamond means that the page is checked in
- A green circle means that the page is checked out
- A green plus means that the page is new
- A red square means that the page has a conflict
- A yellow triangle means that the page has been checked out unsafely

5. To check out a page for editing, right-click on the page icon of the page that you would like to check out in the **Sitemap** pane. In the pop-up menu, click on **Check Out**. For example, right-click on the **Page 2** icon, and click on **Check Out** in the pop-up menu, as shown in the following screenshot:

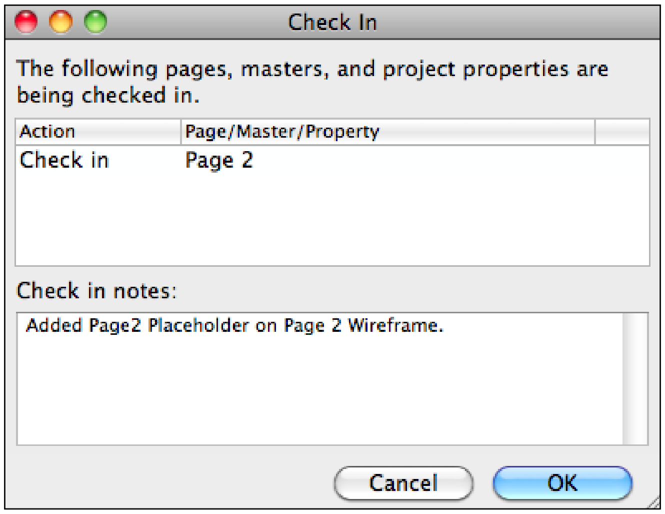


6. When you check out a page(s), your local copy is updated with the latest changes to that page(s), including any Master(s) on that page from the shared directory. The page(s) is also checked out in your user name in the shared directory. This prevents others from checking out the page(s) you are currently editing.

7. When you have checked out a page(s), you will see that the team project icons are shown in **Sitemap** in the lower-left corner of the page(s) and are updated to a green circle. This indicates that the page(s) have been checked out as shown in the following screenshot:



8. Perform your edits to the wireframe.
9. To check in a page, click on the page icon in the **Sitemap** pane that you would like to check in. In the pop-up menu, click on **Check In**.
10. The **Check In** dialog box will open. Click on the **Check in notes:** text area and enter your notes. Click on the **OK** button as shown in the following screenshot:



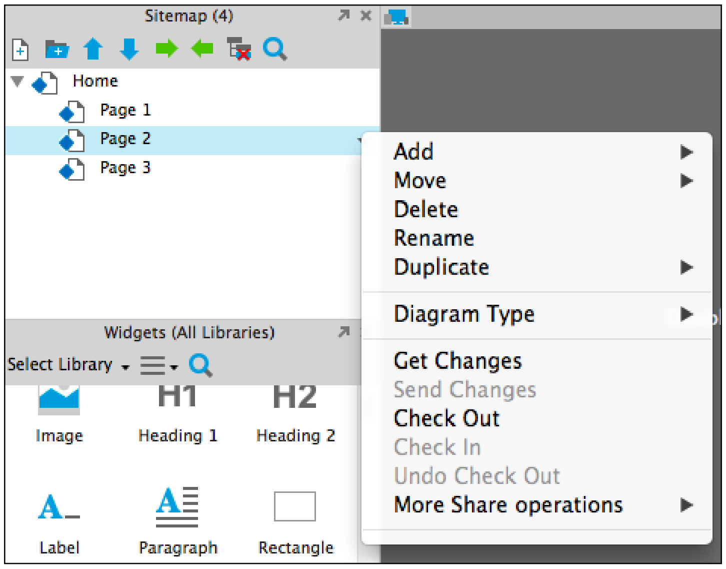
The following pages, masters, and project properties are being checked in.

Action	Page/Master/Property
Check in	Page 2

Check in notes:

Added Page2 Placeholder on Page 2 Wireframe.

11. The team project icon for **Page 2** shown in **Sitemap**, in the lower-left corner of the page icon, is now updated to a blue diamond indicating that the page is checked in. You will also see that the wireframe toolbar has an updated status indicating that the **Page has been Checked in**.
12. There is another way to get changes or check out a page. In the wireframe toolbar to the right, there is a drop-down menu where you can **Get Changes** or **Check Out** the page as shown in the following screenshot:



How it works...

In this recipe, you used **Get Changes** to sync your local copy of the shared project. You then used **Check Out** to check out the page(s). After completing your edits, you used **Check In** to check in the page(s) with all your modifications back into the shared project.

Managing shared projects

Axure provides a **Manage Team Project** dialog to assist in managing shared projects. In this recipe, you will explore the Manage Team Project dialog.

Getting ready

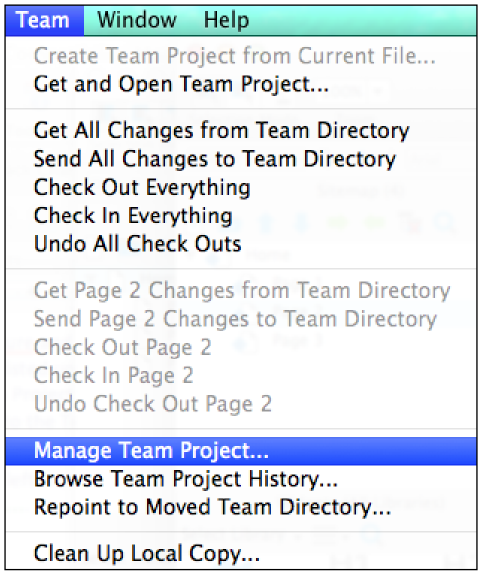
For this recipe, all you will need is a local team project file and a shared project.

How to do it...

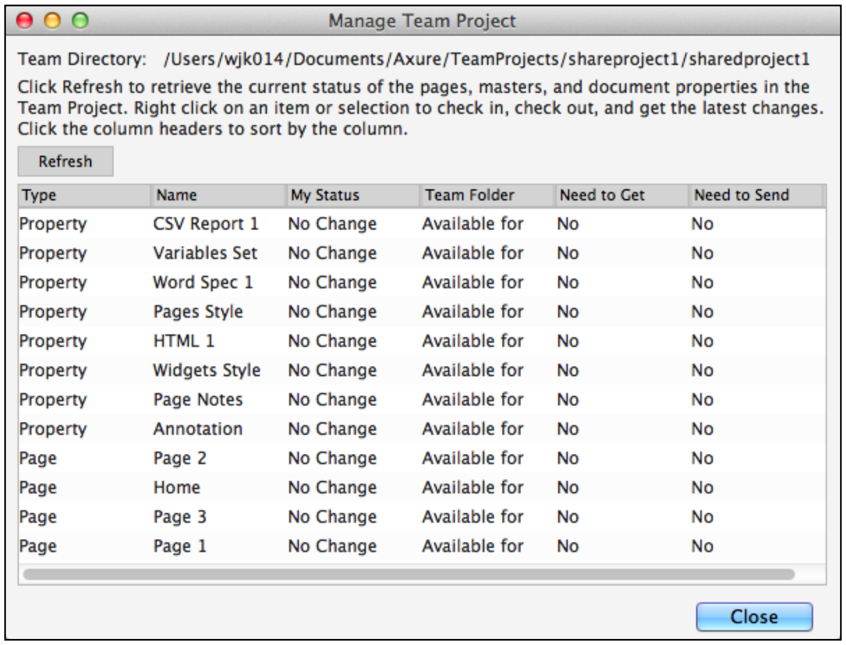
Perform the following steps:

1. To open a local copy of a shared project, start Axure, and select an Axure team project with the `.rpprj` file extension listed under **Open Recent**, or select **Open** and browse to the team project stored locally. You can also use your finder or file explorer, browse to the team project file with the `.rpprj` file extension, and double-click to open.

2. Once the team project opens, in the main menu, left-click on **Share** and then on **Manage Team Project...** as shown in the following screenshot:


Team
Window
Help
Create Team Project from Current File...
Get and Open Team Project...
Get All Changes from Team Directory
Send All Changes to Team Directory
Check Out Everything
Check In Everything
Undo All Check Outs
Get Page 2 Changes from Team Directory
Send Page 2 Changes to Team Directory
Check Out Page 2
Check In Page 2
Undo Check Out Page 2
Manage Team Project...
Browse Team Project History...
Repoint to Moved Team Directory...
Clean Up Local Copy...

3. A **Manage Team Project** dialog box will display. Click on the **Refresh** button to get the current status for pages, Masters, and document properties of your shared project.

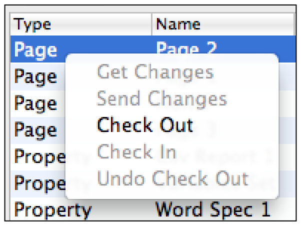


Team Directory: /Users/wjk014/Documents/Axure/TeamProjects/shareproject1/sharedproject1

Click Refresh to retrieve the current status of the pages, masters, and document properties in the Team Project. Right click on an item or selection to check in, check out, and get the latest changes. Click the column headers to sort by the column.

Type	Name	My Status	Team Folder	Need to Get	Need to Send
Property	CSV Report 1	No Change	Available for	No	No
Property	Variables Set	No Change	Available for	No	No
Property	Word Spec 1	No Change	Available for	No	No
Property	Pages Style	No Change	Available for	No	No
Property	HTML 1	No Change	Available for	No	No
Property	Widgets Style	No Change	Available for	No	No
Property	Page Notes	No Change	Available for	No	No
Property	Annotation	No Change	Available for	No	No
Page	Page 2	No Change	Available for	No	No
Page	Home	No Change	Available for	No	No
Page	Page 3	No Change	Available for	No	No
Page	Page 1	No Change	Available for	No	No

4. To get the latest changes or check in/check out of an item, right-click on the item. In the pop-up menu, click on the menu item that you wish to choose. If you decide not to complete an operation on an item, click on the **Close** button to close the **Manage Team Project** dialog.



How it works...

The **Manage Team Project** dialog assists you with managing shared projects. You are able to get changes, send changes, check out, check in, and undo check out for individual items or selections. When you first open the **Manage Team Project** dialog box, remember to left-click on the **Refresh** button to get the current status for your shared project.

There's more...

In the **Manage Team Project** dialog, click on any column header to sort the column.

Using the team project history

By using the team project history feature of shared projects, you can view the change in notes of all revisions within a specified date range. You can then select individual revisions and export that revision to an Axure RP file. In this recipe, you will explore the team project history feature.

Getting ready

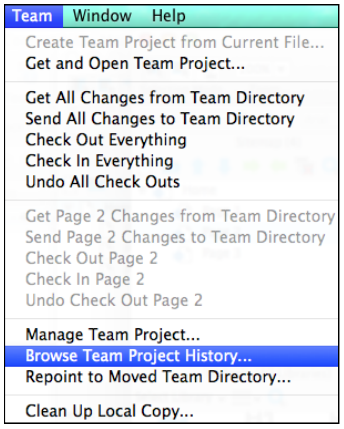
For this recipe, all you will need is a local team project file and a shared project.

How to do it...

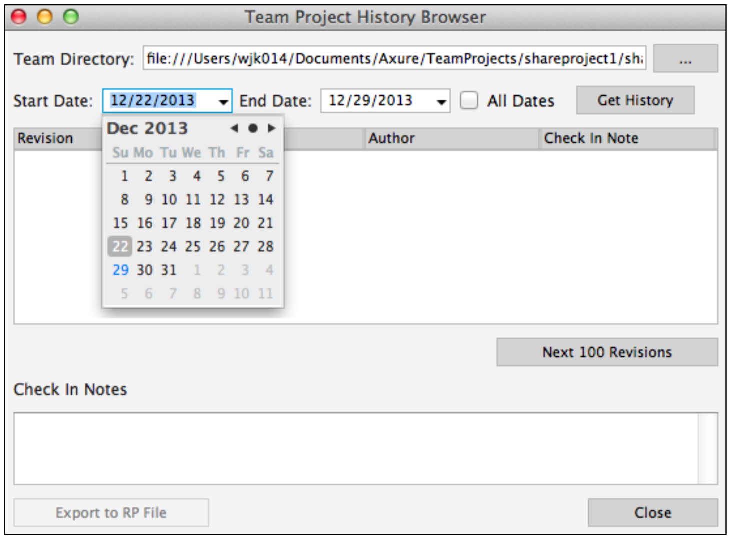
Perform the following steps:

1. To open a local copy of a shared project, start Axure and select an Axure team project with the `.rpprj` file extension listed under **Open Recent**, or select **Open** and browse to the team project stored locally. You can also use your finder or file explorer, browse to the team project file with the `.rpprj` file extension, and double-click to open.

2. Once the team project opens, in the main menu, click on **Share** and then on **Browse Team Project History...** as shown in the following screenshot:

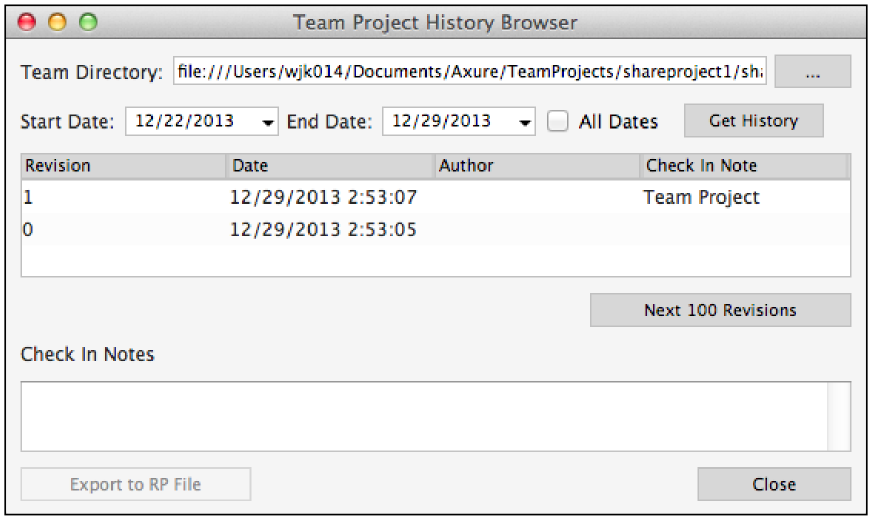

Team
Window
Help
Create Team Project from Current File...
Get and Open Team Project...
Get All Changes from Team Directory
Send All Changes to Team Directory
Check Out Everything
Check In Everything
Undo All Check Outs
Get Page 2 Changes from Team Directory
Send Page 2 Changes to Team Directory
Check Out Page 2
Check In Page 2
Undo Check Out Page 2
Manage Team Project...
Browse Team Project History...
Repoint to Moved Team Directory...
Clean Up Local Copy...

3. A **Team Project History Browser** dialog box will display.
4. To select a date range to display **Team Project History**, click on the **Start Date:** field and enter the start date. You can also click on the **Start Date** dropdown and select **Month** and **Day** for the start date, as shown in the following screenshot:



5. Click on the **End Date:** field and enter the end date. You can also click on the **End Date** dropdown and select **Month** and **Day** for the end date.
6. If you want to select the entire history of the shared project, click on the checkbox next to **All Dates**.

7. To get the history for the shared project, click on the **Get History** button to get the **Team Project History Browser**, as shown in the following screenshot:



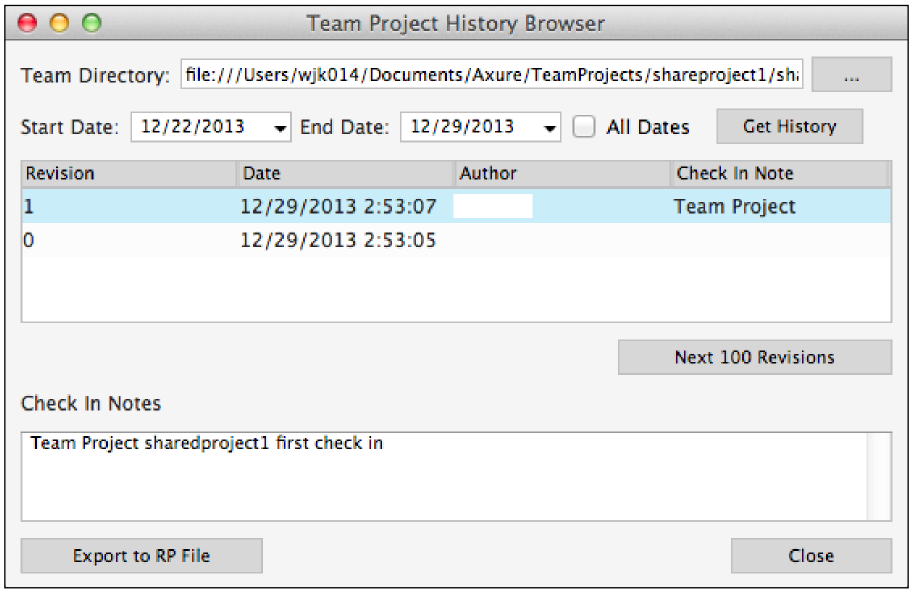
Team Directory: file:///Users/wjk014/Documents/Axure/TeamProjects/shareproject1/sh

Start Date: 12/22/2013 End Date: 12/29/2013

Revision	Date	Author	Check In Note
1	12/29/2013 2:53:07		Team Project
0	12/29/2013 2:53:05		

8. Click on **Revision** to review the check in notes in the **Check In Notes** text area.

9. To get a specific revision, click on **Revision** in the list, and click on the **Export to RP File** button to export that version of the shared project, as shown in the following screenshot:



Team Directory: file:///Users/wjk014/Documents/Axure/TeamProjects/shareproject1/sh

Start Date: 12/22/2013 End Date: 12/29/2013

Revision	Date	Author	Check In Note
1	12/29/2013 2:53:07		Team Project
0	12/29/2013 2:53:05		

Check In Notes: Team Project sharedproject1 first check in

10. When you left-click the **Export to RP File** button, an **Export To Axure RP File** window will open. In the **Save As:** field, enter the name you wish to give to the version of the team project you just selected.
11. Browse to the location where you would like to save the version of the team project, and click on the **Save** button.
12. Once saved, you will be prompted with a dialog box asking if you want to open the exported file. Click on the **Yes** button to open the file and review it.

How it works...

The team project history feature of shared projects allows you to view the change in notes of all revisions within a specified date range. Each revision can be exported to an Axure RP file.

You first select **Share** from the main menu and then browse **Team Project History...**. Next, you select the date range in the **Team Project History Browser** dialog and click on the **Get History** button. This caused the revisions to be listed with the date, author and check in note summary. You are then able to click on a specific revision and review the detailed check in notes in the **Check In Note** text area. You can now click on the **Export to RP file** button to store a local copy of the revision for further review.

Exporting a shared project

You can also export a team project into a standalone Axure RP file.

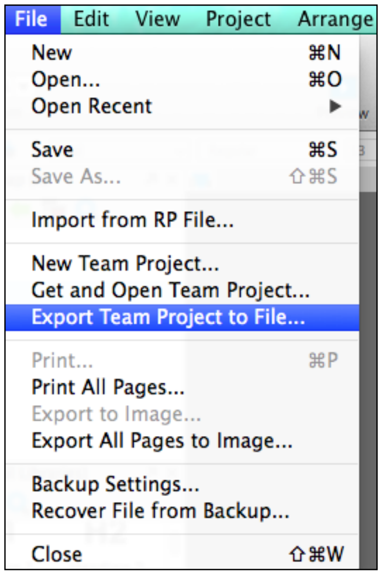
Getting ready

To step through this recipe, you will need to have Microsoft Word or an open source word processor that can open and save files in the Microsoft Word format, such as Open Office, LibreOffice, NeoOffice, or Google Docs; Axure; and a previously created Axure RP file populated with your wireframes.

How to do it...

Perform the following steps:

1. To open a local copy of a shared project, start Axure and select an Axure team project with the `.rpprj` file extension listed under **Open Recent**, or select **Open** and browse to the team project stored locally. You can also use your finder or file explorer, browse to the team project file with the `.rpprj` file extension, and double-click to open.
2. Once the team project opens, in the main menu, click on **File** and then on **Export Team Project to File...**.



3. An **Export Current Team Project To Axure RP File** window will open. In the **Save As:** field, enter the name that you wish to give to the version of the team project you just selected.
4. Browse to the location you would like to save the version of the team project and left-click on the **Save** button.
5. Once saved, you will be prompted with a dialog box asking if you want to open the exported file. Left-click on the **Yes** button to open the file and review it.

How it works...

In this recipe, you learned how to create a local Axure RP file from a shared project. To accomplish this, you first selected **File** from the main menu and then **Export Team Project to File...**. You then entered the name for the exported Axure RP file and browsed to the location where you wanted the file to be saved. You now can make changes to the Axure RP file as you would do to a standalone Axure RP file.

There's more...

Remember that once you export a shared project, the saved Axure RP file is no longer associated to the shared project. If you would like to merge the changes from the Axure RP file you saved in this recipe back into the shared project, you would left-click on **File** in the main menu and then on **Import from RP file**. The **File Import** wizard will open, and you can select what items you want to import into your local copy of the project file. Items that will be updated during the import process must be checked out.

Configuring subversion on an Amazon EC2 instance

As mentioned before, Axure uses Apache Subversion to maintain version control of each individual project. While using a standard file server with shared projects, sometimes the response time of the server can be slower than you would like it to be. A couple of solutions to this problem are: to set up your own SVN server or use a cloud-based SVN provider. In this recipe, you will set up an SVN server on Amazon Web Services Elastic Computing Cloud.

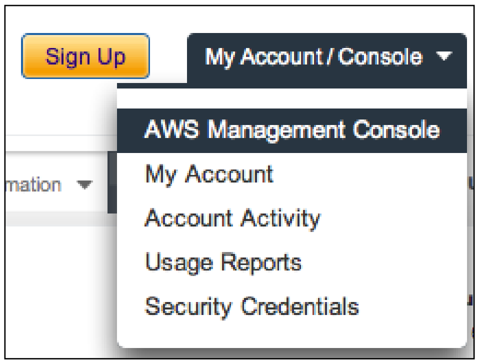
Getting ready

To step through this recipe, you will need to have an account on Amazon's AWS that runs an EC2 instance with Ubuntu and Axure. You can create a free account for Amazon AWS at `http://aws.amazon.com/`. Learn more about the AWS Free Usage Tier at `http://aws.amazon.com/free/`.

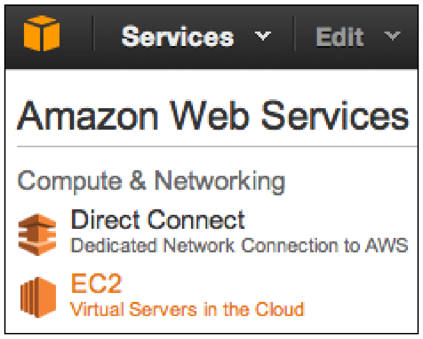
How to do it...

Perform the following steps:

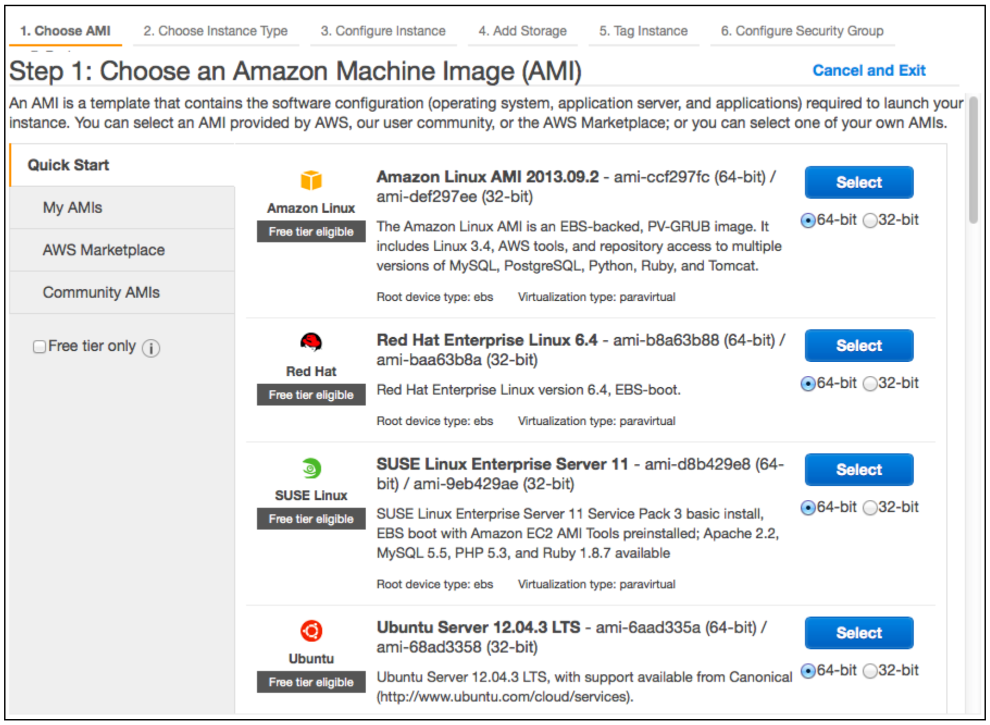
1. Log in to your Amazon AWS account.
2. Click on **My Account/Console**, and click on **AWS Management Console** as shown in the following screenshot:



3. Click on **EC2 Virtual Servers in the Cloud** under the **Compute & Networking** section to launch the **EC2** dashboard as shown in the following screenshot:



4. You will now create an Ubuntu Amazon EC2 instance.
5. Under **Create Instance**, click on the blue **Launch Instance** button.
6. In **Step 1: Choose an Amazon Machine Image (AMI)**, scroll to **Ubuntu Server 12.04.3 LTS** and click on the **Select** button as shown in the following screenshot:



Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Amazon Linux — **Amazon Linux AMI 2013.09.2** - ami-ccf297fc (64-bit) / ami-def297ee (32-bit)
The Amazon Linux AMI is an EBS-backed, PV-GRUB image. It includes Linux 3.4, AWS tools, and repository access to multiple versions of MySQL, PostgreSQL, Python, Ruby, and Tomcat.
Root device type: ebs Virtualization type: paravirtual
[✓ 64-bit] 32-bit

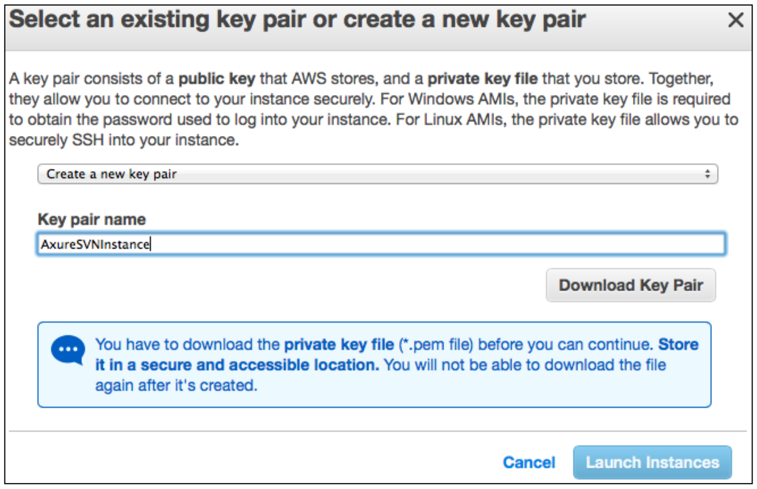
Red Hat — **Red Hat Enterprise Linux 6.4** - ami-b8a63b88 (64-bit) / ami-baa63b8a (32-bit)
Red Hat Enterprise Linux version 6.4, EBS-boot.
Root device type: ebs Virtualization type: paravirtual
[✓ 64-bit] 32-bit

SUSE Linux — **SUSE Linux Enterprise Server 11** - ami-d8b429e8 (64-bit) / ami-9eb429ae (32-bit)
SUSE Linux Enterprise Server 11 Service Pack 3 basic install, EBS boot with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.5, PHP 5.3, and Ruby 1.8.7 available
Root device type: ebs Virtualization type: paravirtual
[✓ 64-bit] 32-bit

Ubuntu — **Ubuntu Server 12.04.3 LTS** - ami-6aad335a (64-bit) / ami-68ad3358 (32-bit)
Ubuntu Server 12.04.3 LTS, with support available from Canonical (http://www.ubuntu.com/cloud/services).
[✓ 64-bit] 32-bit

7. Click on the **Review** and **Launch** button.

8. In the **Select an existing key pair or create a new key pair** dialog pop up, click on the drop-down menu and select **Create a new key pair**. In the **Key pair name** field enter `AxureSVNInstance`. Click on the **Download Key Pair** button. Click on the **Launch Instances** button as shown in the following screenshot:



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

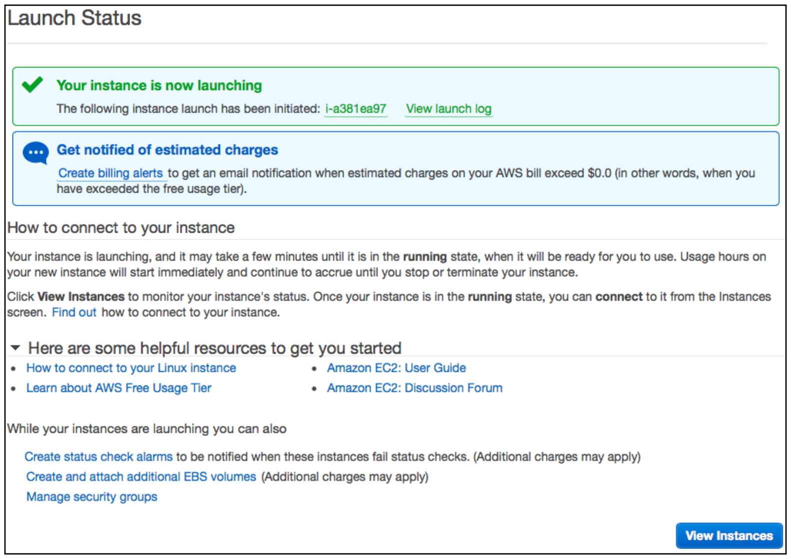
Create a new key pair

Key pair name: AxureSVNInstance

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

On some systems, the downloaded key pair file will be of the form `filename.pem.txt`. Please change the filename to have only a `.pem` extension (that is, `filename.pem`).

9. In **Launch Status**, under **Your instance is now launching**, click on the **Instance ID** as shown in the following screenshot:



Launch Status

Your instance is now launching

The following instance launch has been initiated: i-a381ea97 View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed $0.0 (in other words, when you have exceeded the free usage tier).

How to connect to your instance

Your instance is launching, and it may take a few minutes until it is in the **running** state, when it will be ready for you to use. Usage hours on your new instance will start immediately and continue to accrue until you stop or terminate your instance.

Click **View Instances** to monitor your instance's status. Once your instance is in the **running** state, you can **connect** to it from the Instances screen. Find out how to connect to your instance.

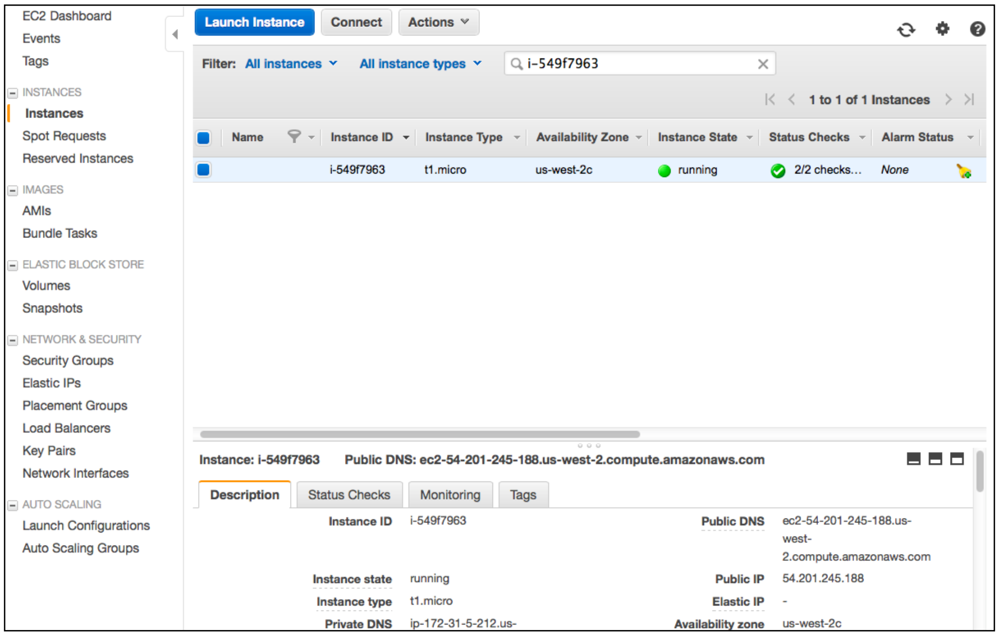
Here are some helpful resources to get you started

- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

While your instances are launching you can also

Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
Create and attach additional EBS volumes (Additional charges may apply)
Manage security groups

10. In the EC2 **Instances** dashboard, click on the box in the first column to select the instance. Next, click on the **Connect** button.



Filter: All instances | All instance types | i-549f7963

1 to 1 of 1 Instances

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
	i-549f7963	t1.micro	us-west-2c	running	2/2 checks...	None

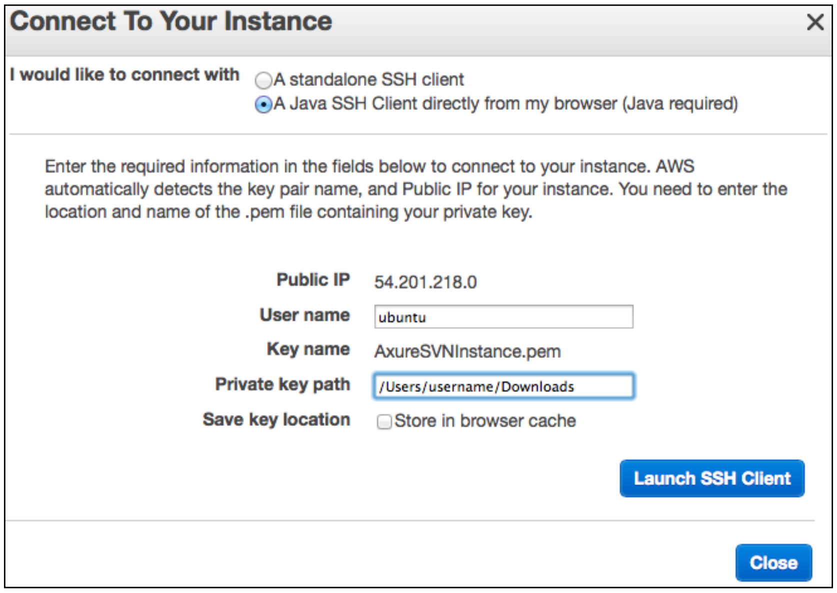
Instance: i-549f7963 Public DNS: ec2-54-201-245-188.us-west-2.compute.amazonaws.com

Instance ID	i-549f7963	Public DNS	ec2-54-201-245-188.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	54.201.245.188
Instance type	t1.micro	Elastic IP	-
Private DNS	ip-172-31-5-212.us-	Availability zone	us-west-2c

11. When your instance started, a `.pem` file that contained your private key was downloaded to the `download` directory for your browser. On some computers, this path is `/home/username/downloads` or `/Users/username/downloads` where `username` is your login. You can also search for `*.pem` to identify the location of the `.pem` file.

[The `.pem` extension is an acronym for privacy enhanced e-mail and may contain an entire certificate chain, including public key, private key, and root certificates.]

12. In the **Connect To Your Instance** dialog window, click on the **A Java SSH Client directly from my browser (Java required)** radio button, as shown in the following screenshot:



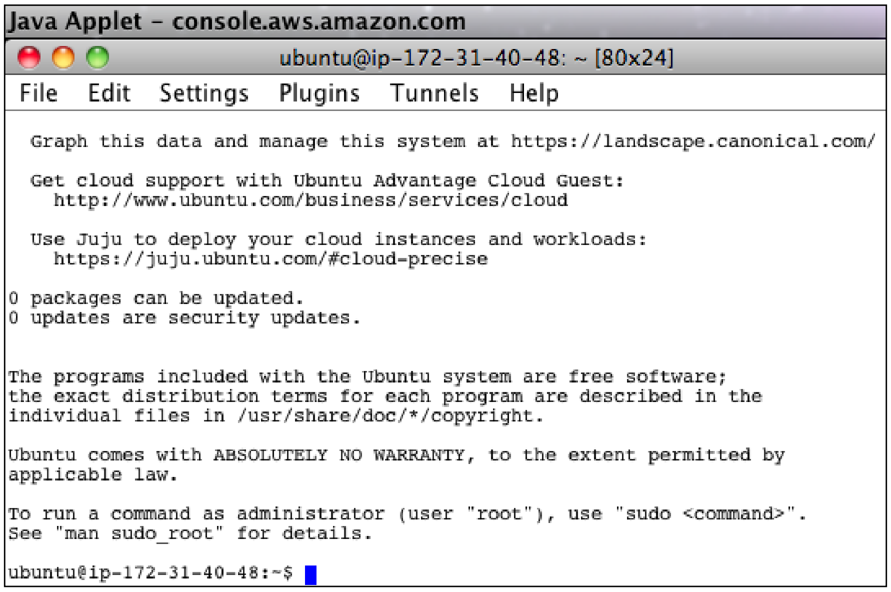
Connect To Your Instance

I would like to connect with
( ) A standalone SSH client
(•) A Java SSH Client directly from my browser (Java required)

Enter the required information in the fields below to connect to your instance. AWS automatically detects the key pair name, and Public IP for your instance. You need to enter the location and name of the .pem file containing your private key.

Public IP	54.201.218.0
User name	ubuntu
Key name	AxureSVNInstance.pem
Private key path	/Users/username/Downloads
Save key location	☐ Store in browser cache

13. In the **Private key path:** field, enter the path to your `.pem` file and the filename of your `.pem` file; for example, `/Users/mylogin/axuresvn.pem`.
14. Click on the **Launch SSH Client** button.
15. A **MindTerm – License agreement** dialog window will open. After reviewing the terms, click on the **Accept** button.
16. A **MindTerm – Confirmation** dialog window will open, click on the **Yes** button to create the `MindTerm` home directory.
17. A **MindTerm – Confirmation** dialog window will open, click on the **Yes** button to have the `MindTerm` hosts added to your known hosts.
18. A Java applet titled **console.aws.amazon.com** will open with a terminal window logged into your AWS EC2 instance, as shown in the following screenshot:


Java Applet – console.aws.amazon.com
ubuntu@ip-172-31-40-48: ~ [80x24]
File Edit Settings Plugins Tunnels Help

```
  Graph this data and manage this system at https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

  Use Juju to deploy your cloud instances and workloads:
    https://juju.ubuntu.com/#cloud-precise

0 packages can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-40-48:~$
```

19. To update your Ubuntu EC2 instance with the latest updates, type the following command and press *Enter*:

    `sudo apt-get update dist-upgrade`

20. To update your Ubuntu EC2 instance with Apache, type the following command and press *Enter*:

    `sudo apt-get install apache2 apache2-doc apache2-utils`

21. To update your Ubuntu EC2 instance with Subversion and the Apache SVN module, type the following command and press *Enter*:

    `sudo apt-get install subversion libapache2-svn`

22. You may be prompted to press *Y* to continue the update process. Once the updates have finished, you will see a message saying **Complete!**, and the cursor will appear at the command prompt.

23. Now, you will create a default directory for the SVN repository and a repository named `axurerepo1`. Enter the following commands:

    `sudo mkdir /svn`

    `sudo chown www-data:www-data /svn`

```
sudo -u www-data mkdir /svn/repos
sudo -u www-data svnadmin create /svn/repos/axurerepo1
```

24. Next, create an Apache `conf` file. Enter the following command:

    ```
        sudo vi /etc/apache2/conf.d/my_svn.conf
    ```

25. Hold down the *Shift* key and press *O*. Enter the following lines of code:

    ```
    <VirtualHost *:80>
      ServerName svn.yourserver.com
      <Location /repos>
        DAV svn
        SVNListParentPath on
        SVNParentPath /svn/repos/
        AuthType Basic
        AuthName "Authorization Realm"
        AuthUserFile /etc/auth-file-svn
        Require valid-user
      </Location>
    </VirtualHost>
    ```

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

26. In the preceding code, `yourserver` is the hostname of your EC2 instance.
27. Create a login and password for the SVN repository. Enter the following command:

    ```
    sudo htpasswd -c /etc/auth-file-svn username
    ```

28. Change the username to the one you would like to use for Axure user in order to access the SVN repository.
29. Restart Apache for the changes to take effect. Enter the following command:

    ```
    sudo service apache2 restart
    ```

30. In a browser, enter `http://svn.yourserver.com/repos/`, where `yourserver` is the hostname of your EC2 instance, to access your SVN repository.

How it works...

You first created an Ubuntu server in an Amazon EC2 instance. You then installed and configured an Apache web server and SVN.

7
Adaptive and Responsive Web Design with Axure

In this chapter, we will discuss how to leverage Axure prototypes to communicate adaptive and responsive designs. You will learn how to do the following:

- Designing for adaptive and responsive layouts
- Applying your responsive grid to Axure prototypes
- Using jQuery to emulate media queries with Axure
- Using jQuery Mobile with Axure
- Building a responsive navigation example
- Building a responsive home page prototype
- Building a responsive gallery grid
- Building a responsive landing page

Introduction

Users expect rewarding experiences on mobile and tablet devices when visiting a site. Adaptive web designs will reposition screen elements and reconfigure menus based on the screen size of a device. Responsive web designs go one step further and may limit functionality to optimize the user experience as well as resize and reposition screen elements by utilizing fluid grids, flexible images, and media queries.

Designing for adaptive and responsive layouts

There are two approaches to starting your layout. In one approach, you design first for the desktop. Next, select each arbitrary breakpoint, and design a layout for each successively smaller breakpoint. Finally, complete the overall design with mobile. In the desktop-first approach, you focus on graceful degradation as the screen size decreases.

In the second approach, you start with mobile first. Next, design each breakpoint of increasing width. Finally, finish the design with the desktop layout. For a mobile-first design, you will focus on progressive enhancements as the screen size increases.

Getting ready

For this recipe, we will walk through the steps to design for an adaptive or responsive layout.

How to do it...

1. Decide which design approach you will utilize for your project—desktop-to-mobile or mobile-first.
2. Determine the pixel width for desktop and mobile.
3. In Axure, set guides to assist with the spacing of elements by selecting **Arrange** from the main menu, moving the mouse over **Grids and Guides**, and left-clicking on **Create Guides...** from the flyout menu.
4. Breakpoints are browser widths at which a layout transition occurs (for example, large display, desktop, tablet, and mobile). Determine how many breakpoints will be required for your design and at what pixel dimensions the breakpoints will occur (for example, 1400 pixels, 1024 pixels, 960 pixels, 810 pixels, 620 pixels, 480 pixels, and 320 pixels).
5. Complete each design, adapting the layout for each breakpoint identified in step 4.
6. Evaluate your various designs across multiple devices by adjusting breakpoints and tweaking the designs as necessary to accommodate the most prevalent devices. Following is a brief table based on current trends:

Type	Device	Screen resolution
Large display	MacBook Pro (Retina 15")	2880-by-1800
Desktop	Desktop monitors	1920-by-1200
Desktop	Desktop monitors	1920-by-1080
Tablet	iPad Air	2048-by-1536
Tablet	Samsung Galaxy Note 10.1	2560-by-1600
Tablet	Windows Surface Pro 2	1920-by-1080
Mobile	iPhone 5S	1136-by-640
Mobile	Samsung Galaxy S4	920-by-1080
Mobile	Nokia Lumnia 1520	1920-by-1080

How it works...

In this recipe, we discussed two examples enabling design for adaptive and responsive layouts. In the initial approach, we designed for the desktop-first and then each subsequent breakpoint until we finished our design with mobile. In the second approach, we started with mobile-first and increased the screen size for each breakpoint until we finalized our design with the desktop.

In the design community, there are numerous advocates of both approaches. The key to either approach is to determine the most effective breakpoints and complete the initial designs. Next, evaluate the layouts for each breakpoint across the most popular devices, and hone the design when possible. This approach ensures that the majority of users have an optimal experience.

Applying your responsive grid to Axure prototypes

Axure RP 7 and beyond enables you to create adaptive prototypes with less effort than previous versions. This is accomplished through a new feature called adaptive views.

Adaptive views inherit properties from the Parent or Base view. Making changes in Child views does not impact the Parent views.

You will first decide if your Base view is going to be desktop or mobile. You will then create views for each breakpoint and then adjust the individual widgets in each Child view as necessary.

In this recipe, you will learn how to lay out an adaptive grid and make it behave in a responsive fashion.

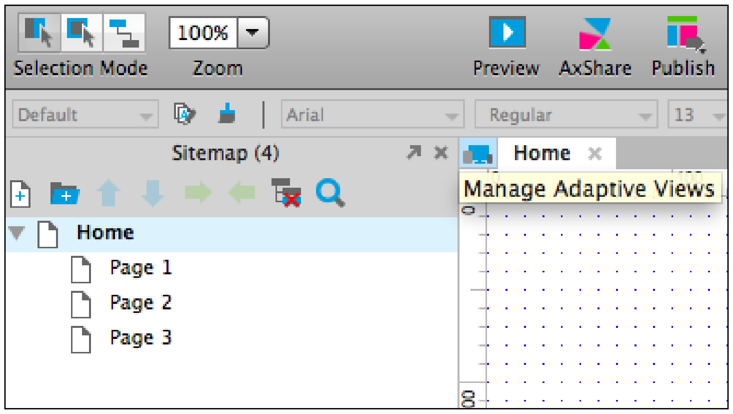
How to do it...

You will create three views. A Base view that will be shown for browser widths of 980 pixels and larger, a tablet view that is shown for browser widths between 979 pixels and 768 pixels, and a mobile view that is shown for browser widths of 767 pixels and below. To create the views, perform the following steps:

1. Start Axure and under **Create New** select **RP File**.

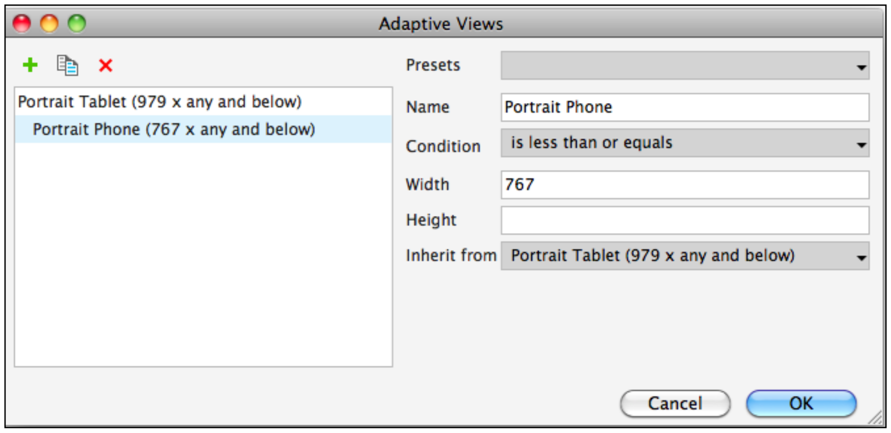
[If you already have Axure open, click on **File** in the main menu, and then click on **New** in the drop-down menu to create a new RP document.]

2. In the main menu, click on **Project**, and then either click on **Adaptive Views...**, or click on the **Manage Adaptive Views** icon in the wireframe pane.



3. In the **Adaptive Views** dialog box, click on the green *plus* sign to add a view. Click on the **Presets** dropdown, and click on the **Portrait Tablet** preset. Change the **Width** to `979`.

4. Repeat step 3, and add a view for **Portrait Phone**. Change the **Width** to `767`. Click on the **OK** button.

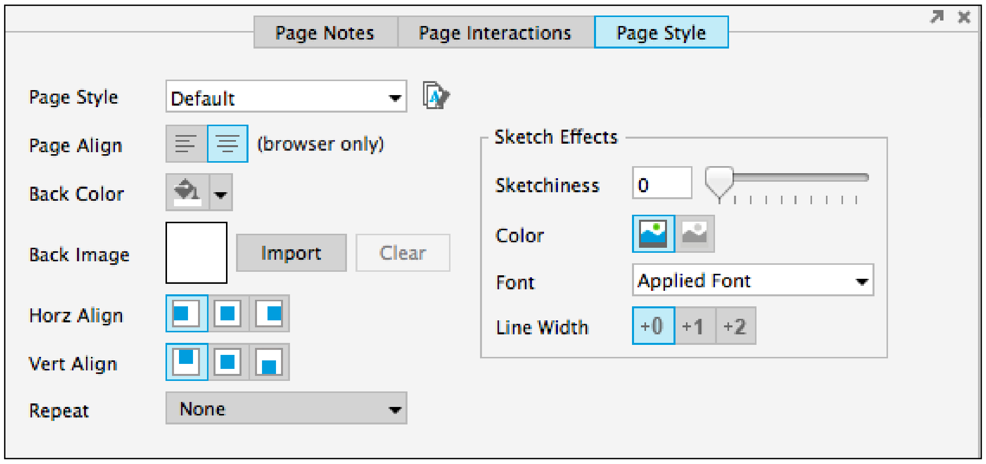


Adaptive Views

Portrait Tablet (979 x any and below)
Portrait Phone (767 x any and below)

Presets:
Name: Portrait Phone
Condition: is less than or equals
Width: 767
Height:
Inherit from: Portrait Tablet (979 x any and below)

5. Under the **wireframe pane**, click on the **Page Style** tab. In the **Page Style** pane, click on the **Page Align** icon on the right-hand side. The page content will now be centered for the browser only.

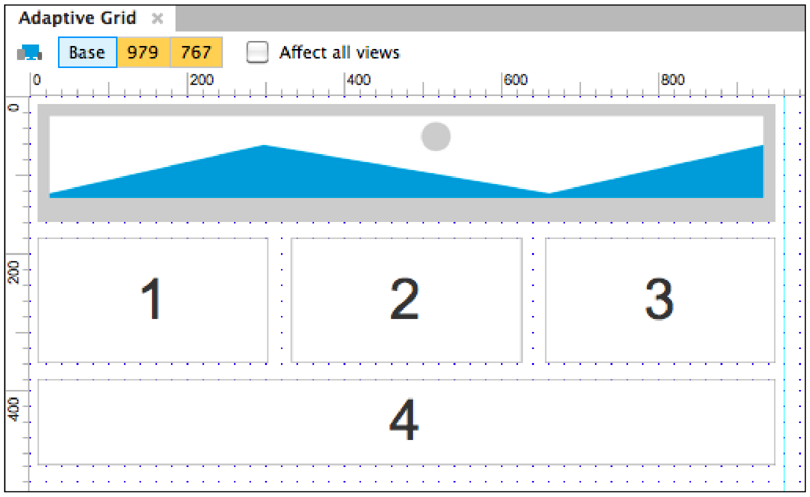


Page Style: Default
Page Align: (browser only)
Back Color
Back Image

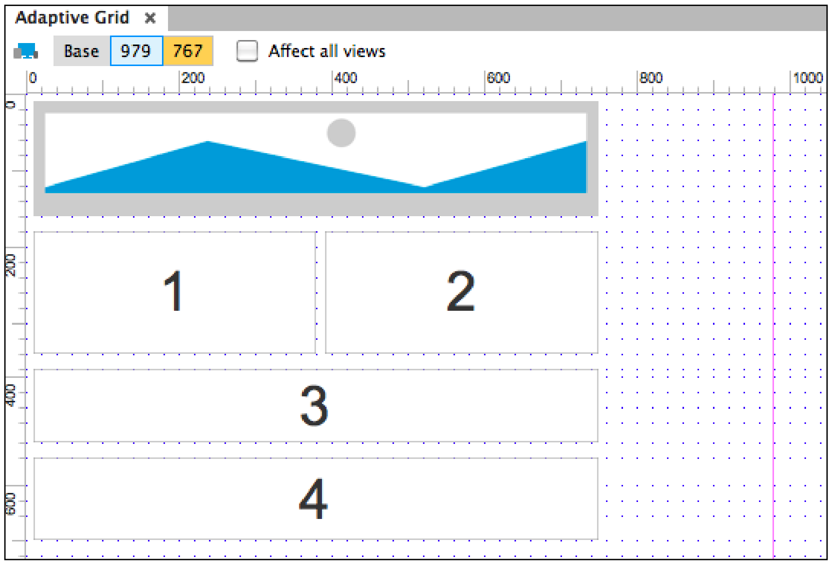
Horz Align
Vert Align
Repeat: None

Sketch Effects
Sketchiness: 0
Color
Font: Applied Font
Line Width: +0 +1 +2

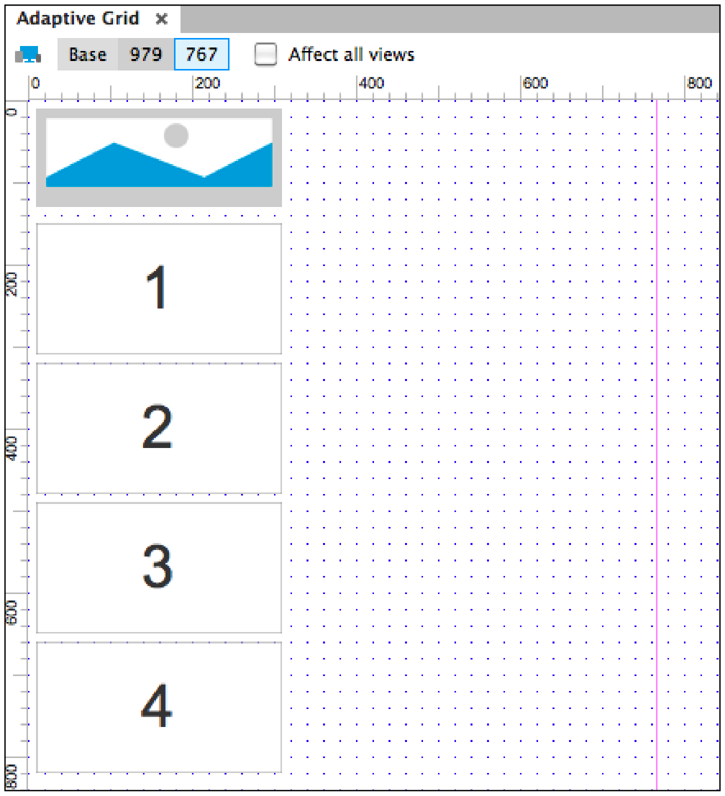
6. The **Base** view will be shown when the viewport or browser window is sized at 980 pixels and above. Click on the **Base** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow. Now place and size the widgets you would like to show for this view.



7. The **979** view will be shown when the viewport or browser window is sized from 979 pixels to 768 pixels. For this view, we will place our design optimized for 768 pixels. Click on the **979** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow. Now place and size the widgets you would like to show for this view.



8. The **767** View will be shown when the viewport or browser window is sized from 767 pixels to 0 pixel. For this view, we will place our design optimized for 480 pixels. Left-click on the **767** view icon in the header of the wireframe pane. The selected view will turn blue and views that inherit from the selected view will be yellow. Now place and size the widgets you would like to show for this view.



9. Click on the **Preview** icon above the wireframe pane to preview the HTML prototype.

[To see the adaptive views, resize your browser window, or click on **Select Adaptive Views** in the **Sitemap** pane.]

10. If you are satisfied with the preview, you can generate a prototype.
11. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish**, and then clicking on **Generate HTML Files...**.

How it works...

For this recipe, you learned how to lay out an adaptive grid and made the grid appear to behave in a responsive fashion. This was accomplished through using the new feature adaptive views. You defined breakpoints just below your target viewport sizes. Next, you adjusted each design by optimizing the content for the smallest viewport shown for a given view. To make the prototype feel more like a responsive layout, you also set the page alignment to center in the **Page Style** pane.

Using jQuery to emulate media queries with Axure

One of the most popular JavaScript libraries for websites in use today is jQuery. You can use jQuery to emulate media queries with Axure, making Axure widgets truly responsive in your prototypes. In this recipe, you will use jQuery to create a responsive background image that maintains the aspect ratio and resizes when the size of the browser changes.

How to do it...

Perform the following steps to emulate media queries:

1. Start Axure and under **Create New** select **RP File**.
2. Double-click on the page titled **Home** in the site map. You will see the **Home** wireframe shown in the wireframe pane.
3. Under the wireframe pane, click on the **Page Style** tab. In the **Page Style** pane, click on the **Page Align** icon on the right. The page content will now be centered for the browser only.
4. While holding down the mouse button, drag the **Image** widget, and place it at coordinates (0,0) on the wireframe.
5. Double-click on the **Image** widget on the wireframe, and select the image you would like to use for your background.
6. A **Warning** dialog box will open asking you if you would like to optimize the image. Left-click on the **Yes** button.
7. An **Autosize** dialog box will open asking you if you would like to autosize the image. Left-click on the **Yes** button.
8. To save a copy of the prototype, click on the **Publish** button in the toolbar, and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish**, and then click on **Generate HTML Files...**.
9. Choose the location where you would like to store the prototype and left-click on the **Generate** button.

10. Browse to the location at which you saved the prototype.
11. Open the file labeled `home.html` with a text editor.
12. Insert the following jQuery script just before the `</head>` tag, and save the HTML file:

```
<!-- jQuery Script to Emulate Media Queries on browser resize -->
    <script>
    $(function() {

      var theWindow        = $(window),
          $u0              = $("#u0"),
        $u0_img          = $("#u0_img");

      function resizeU0() {
        if ( (theWindow.width())<=480) {
        $("body").css({"background-color":"purple"});
        $("#u0").css({"position":
          "static","z-index":"-2","margin-top":"1%",
        "margin-bottom":"1%","margin-right":"5%",
          "margin-left":"5%",
        "width":"90%","height":"auto"});
        $("#u0_img").css({"position":"static",
          "z-index":"-1","width":"100%",
        "height":"auto"});
        }
        else if ( (theWindow.width())<=768) {
        $("body").css({"background-color":"green"});
        $("#u0").css({"position":"static",
          "z-index":"-2","margin-top":"1%",
        "margin-bottom":"1%","margin-right":"5%",
          "margin-left":"5%",
        "width":"90%","height":"auto"});
        $("#u0_img").css({"position":"static",
          "z-index":"-1","width":"100%",
        "height":"auto"});
        }
        else if ( (theWindow.width())<=1024) {
        $("body").css({"background-color":"blue"});
        $("#u0").css({"position":"static",
          "z-index":"-2","margin-top":"1%",
        "margin-bottom":"1%","margin-right":"5%",
          "margin-left":"5%",
        "width":"90%","height":"auto"});
        $("#u0_img").css({"position":"static",
          "z-index":"-1","width":"100%",
        "height":"auto"});
```

```
          }
          else {
          $("body").css({"background-color":"pink"});
          $("#u0").css({"position":"static",
            "z-index":"-2","margin-top":"1%",
          "margin-bottom":"1%","margin-right":"5%",
            "margin-left":"5%",
          "width":"90%","height":"auto"});
          $("#u0_img").css({"position":"static",
            "z-index":"-1","width":"100%",
          "height":"auto"});
          }
        }

        theWindow.resize(function() {
          resizeU0();
        }).trigger("resize");

      });
    </script>
```

13. Left-click on the `home.html` file, and open with your favorite browser.
14. As you resize the web browser, notice that the image maintains the correct aspect ratio and that the background image changes color when the browser size reaches 480 pixels, 768 pixels, and 1024 pixels.

How it works...

When Axure generates the prototype, the image widget is assigned the CSS ID `#u0_img` and is placed within a `div` container with the CSS ID `#u0`. When the browser window is resized, the jQuery script changes the default CSS properties for the body tag as well as `#u0` and `#u0_img` based on the size of the browser. For `#u0` and `#u0_img`, the width and height properties are changed from fixed to percentage and auto respectively. This enables the element to float and become responsive to the changes in browser size.

There's more...

There are four browser widths selected as points to change the body background color and CSS properties. These widths are 480 pixels for mobile, 768 pixels for tablet, 1024 pixels for desktop, and greater than 1024 pixels for large desktop. This is accomplished using the `if else` statements in the jQuery `resizeU0` function. You can use additional `if else` statements as well as change the width at which the CSS changes are applied.

[Go to `http://jquery.com/` to learn more about jQuery. Another excellent free resource for learning basic HTML, CSS, and jQuery is `http://www.w3schools.com/`.]

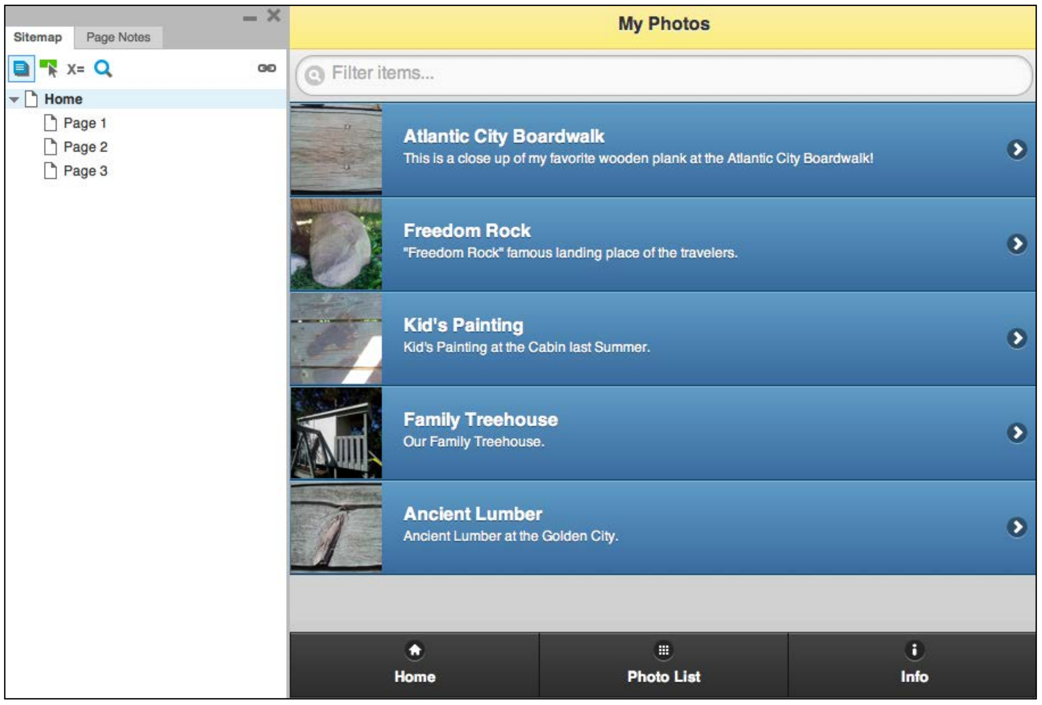
Using jQuery Mobile with Axure

jQuery Mobile is a popular touch-optimized framework for tablets and smartphones that provides a user interface based on HTML5 and CSS. HTML is an acronym for hypertext markup language and provides the structure of a web page. CSS is an acronym for cascading style sheets and provides the layout for a web page.

In this recipe, you will create a responsive inline frame widget in Axure with the frame target as a responsive jQuery Mobile prototype. You can learn more about jQuery Mobile at `http://jquerymobile.com/`.

Getting Ready

In this recipe, you will create a responsive jQuery Mobile prototype using an inline frame widget as shown in the following screenshot:



Atlantic City Boardwalk
This is a close up of my favorite wooden plank at the Atlantic City Boardwalk!

Freedom Rock
"Freedom Rock" famous landing place of the travelers.

Kid's Painting
Kid's Painting at the Cabin last Summer.

Family Treehouse
Our Family Treehouse.

Ancient Lumber
Ancient Lumber at the Golden City.

How to do it...

1. Start Axure and under **Create New** select **RP File**.
2. Double-click on the page titled **Home** in the site map. You will see the **Home** wireframe shown in the wireframe pane.
3. While holding down the mouse button, drag the **Inline Frame** widget, and place it at coordinates (0,0) on the wireframe.
4. With the **Inline Frame** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Inline Frame** widget. Enter `1366` in the width field and `768` in the height field.
5. Right-click on the **Inline Frame** widget on the wireframe, and click on **Frame Target**.
6. In the **Link Properties** pop up, left-click on the radio button next to **Link to an external URL or file**.
7. Left-click on the **Hyperlink** field, and enter `jquery_mobile_example.html`.
8. Click on **OK**.
9. To save a copy of the prototype, click on the **Publish** button in the toolbar, and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish**, and then clicking on **Generate HTML Files...**.
10. Choose the location in which you would like to store the prototype, and left-click on the **Generate** button.
11. Note the directory where the prototype is saved.
12. Copy your HTML document referenced in step 7, and save it in the same directory as your prototype.
13. Download the `jquery.mobile-1.3.2.zip` file from `http://jquerymobile.com/resources/download/jquery.mobile-1.3.2.zip` and extract to a temporary directory.
14. From your temporary directory, copy `jquery.mobile-1.3.2.min.css` to the directory in which you saved the prototype `/resources/css` folder.
15. From your temporary directory, copy `jquery.mobile-1.3.2.min.js` to the directory in which you saved the prototype `/resources/scripts` folder.
16. In the directory you saved the prototype `/resources/css/images` folder, copy from your temporary directory `/jquery/images` all files to the new image folder. There should be five images as follows: `ajax-loader.gif`, `icons-18-black.png`, `icons-18-white.png`, `icons-36-white.png`, and `icons-36-black.png`.
17. In the directory you saved the prototype, create `jquery_mobile_example.html` with the following lines of code:

```
<!-—jQuery Mobile Example for Axure Prototype -->
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>ListApp</title>
  <meta name="viewport" content="width=device-width,
    initial-scale=1, maximum-scale=1" />
<link rel="stylesheet" href=
  "resources/css/jquery.mobile-1.3.2.min.css" />
<script src=
  "resources/scripts/jquery-1.7.1.min.js"></script>
<script src=
  "resources/scripts/jquery.mobile-1.3.2.min.js"></script>
  <style>
    img.fullscreen {
      max-height: 100%;
      max-width: 100%;
    }
  </style>
</head>
<body>
<div data-role="page" id="photo_list" data-theme="b">
  <header data-role="header" data-theme="e">
    <h1>My Photos</h1>
  </header>

  <article data-role="content">
    <ul data-role="listview"  data-filter="true">
      <li>
        <a href="#boardwalk">
        <h1>Atlantic City Boardwalk</h1>
        <img src=
          "images/jqmobile_demo/tn_boardwalk_plank.jpg"
        alt="Atlantic City Boardwalk Wooden Plank" />
        <p>This is a close up of my favorite
          wooden plank at the Atlantic City
        Boardwalk!</p>
        </a>
      </li>
      <li>
        <a href="#freedom_rock">
        <h1>Freedom Rock</h1>
        <img src="images/jqmobile_demo/tn_freedom_rock.jpg"
          alt="Freedom Rock" />
          <p>"Freedom Rock" famous landing place
            of the travelers.</p>
          </a>
        </li>
        <li>
          <a href="#kids_painting">
          <h1>Kid's Painting</h1>
          <img src="images/jqmobile_demo/tn_kids_painting.jpg"
    alt="Kids Painting" />
          <p>Kid's Painting at the Cabin last Summer.</p>
          </a>
        </li>
        <li>
          <a href="#treehouse">
          <h1>Family Treehouse</h1>
          <img src="images/jqmobile_demo/tn_treehouse.jpg"
            alt="Family Treehouse" />
          <p>Our Family Treehouse.</p>
          </a>
        </li>
        <li>
          <a href="#weathered_lumber">
          <h1>Ancient Lumber</h1>
          <img src="images/jqmobile_demo/tn_weathered_lumber.jpg"
  alt="Weathered Lumber" />
          <p>Ancient Lumber at the Golden City.</p>
          </a>
        </li>
      </ul>
    </article>
    <footer data-role="footer" data-position="fixed"
      data-theme="a">
      <nav data-role="navbar">
        <ul>
          <li><a href="#photo_list" data-icon="home">Home</a></li>
          <li><a href="#photo_list" data-icon="grid">
            Photo List</a></li>
          <li><a href="#info" data-icon="info">Info</a></li>
        </ul>
      </nav>
    </footer>
  </div><!-- Page Photo_list -->

  <div data-role="page" id="info" data-theme="b">
    <header data-role="header" data-theme="e">
```

```
      <h1>Info</h1>
      <a href="#photo_list" data-icon="home"
        data-iconpos="notext">Home</a>
    </header>
    <article data-role="content">
      <h1>jQuery Mobile Example</h1>
      <p>Prototyping with jQuery Mobile and Axure
        is awesome!</p>
    </article>
  </div><!-- Page Info -->

    <div data-role="page" id="boardwalk" data-theme="b">
      <header data-role="header" data-theme="e">
        <h1>Atlantic City Boardwalk Plank</h1>
       <a href="#photo_list" data-icon="grid"
         data-iconpos="notext">Photo List</a>
      </header>
      <img src="images/jqmobile_demo/boardwalk_plank.jpg"
        class="fullscreen" alt=
          "Atlantic City Boardwalk Wooden Plank" />
    </div><!-- Page Boardwalk -->

    <div data-role="page" id="freedom_rock" data-theme="b">
      <header data-role="header" data-theme="e">
        <h1>Freedom Rock</h1>
        <a href="#photo_list" data-icon="grid"
          data-iconpos="notext">Photo List</a>
      </header>
      <img src="images/jqmobile_demo/freedom_rock.jpg"
        class="fullscreen" alt="Freedom Rock" />
    </div><!-- Page Freedom Rock -->

    <div data-role="page" id="kids_painting" data-theme="b">
      <header data-role="header" data-theme="e">
        <h1>Kid's Painting</h1>
        <a href="#photo_list" data-icon="grid"
          data-iconpos="notext">Photo List</a>
      </header>
      <img src="images/jqmobile_demo/kids_painting.jpg"
  class="fullscreen" alt="Kid's Painting" />
    </div><!-- Page Kid's Painting -->

    <div data-role="page" id="treehouse" data-theme="b">
      <header data-role="header" data-theme="e">
        <h1>Family Treehouse</h1>
        <a href="#photo_list" data-icon="grid"
          data-iconpos="notext">Photo List</a>
      </header>
      <img src="images/jqmobile_demo/treehouse.jpg"
        class="fullscreen" alt="Family Treehouse" />
    </div><!-- Page Family Treehouse -->

    <div data-role="page" id="weathered_lumber"
      data-theme="b">
      <header data-role="header" data-theme="e">
        <h1>Ancient Lumber</h1>
        <a href="#photo_list" data-icon="grid"
          data-iconpos="notext">Photo List</a>
      </header>
      <img src="images/jqmobile_demo/weathered_lumber.jpg"
        class="fullscreen" alt="Family Treehouse" />
    </div><!-- Page Ancient Lumber -->

  </body>
  </html>
```

18. In the `directory you saved the prototype/images folder`, create a `jqmobile_demo` folder, and place the thumbnails (200 pixels by 200 pixels) and full-size images (1024 pixels by 768 pixels). Make sure to update the HTML in step 17 with the correct thumbnail and main image filenames as well as the `H1` and `P` tag contents to customize your prototype.

> For example, the reference to `tn_boardwalk_plank.jpg` is the thumbnail image, and the full size image is `boardwalk_plank.jpg`.

19. From the directory in which you saved the prototype, open `/files/home/styles.ccs`, and replace all text with the following code:

```
    /* Responsive CSS for home.html */
body {
  margin:0px;
  left:-0px;
  width:100%;
  min-height:100%;
  margin-left:auto;
  margin-top:auto;
  margin-right:auto;
  margin-bottom:auto;
  text-align:left;
}
```

```
#u0 {
  left:0px;
  top:0px;
  width:100%;
  height:99%;
  margin-left:auto;
  margin-top:auto;
  margin-right:auto;
  margin-bottom:auto;
}
```

20. In your prototype directory, open `start.html` with a browser to view the prototype.

How it works...

In this recipe, you created a responsive jQuery Mobile prototype. This was accomplished by creating an Axure prototype that contained a responsive inline frame widget with the frame target as a jQuery Mobile HTML file. You made the default inline frame widget responsive by modifying the CSS height and width properties for the Axure prototype's HTML `<body>` tag and the inline frame widget (`#u0`).

Building a responsive navigation example

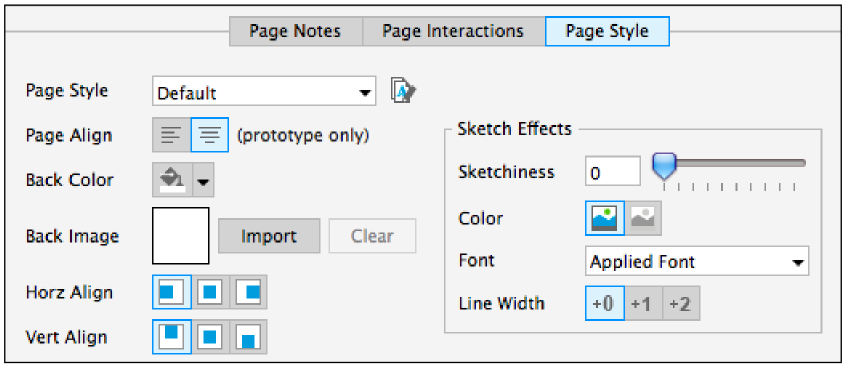
Using Axure RP 7's adaptive views, you can create a navigation prototype that appears to be responsive. In this recipe, you will create an adaptive horizontal navigation widget using adaptive views.

How to do it...

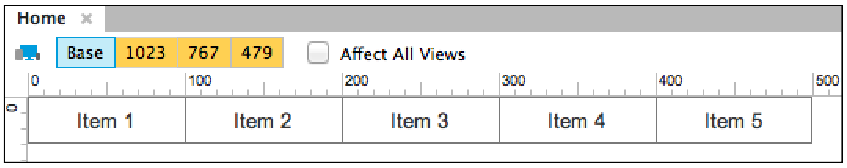
Perform the following steps to create the navigation widget:

1. Start Axure and under **Create New** select **RP File**.
2. In the main menu, click on **Project**, and then either click on **Adaptive Views...**, or click on the **Manage Adaptive Views** icon in the wireframe pane.
3. In the **Adaptive Views** dialog box, click on the green *plus* sign to add a view. Click on the **Presets Dropdown**, and click on the **Landscape Tablet** preset. Change the **Width** to `1023`.
4. Repeat step 3, and add a view for **Portrait Tablet**. Change the **Width** to `767`. Click on the **OK** button.
5. Repeat step 3, and add a view for **Landscape Phone**. Change the **Width** to `479`. Left-click on the **OK** button.

6. Under the wireframe pane, click on the **Page Style** tab. In the **Page Style** pane, click on the **Page Align** icon on the right. The page content will now be centered for the browser only.

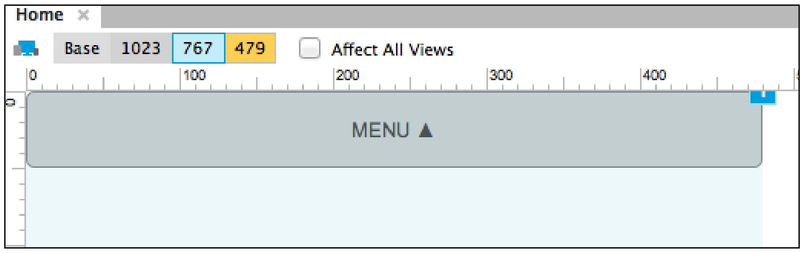


7. The **Base** view will be shown when the viewport or browser window is sized at 1024 pixels and above. Click on the **Base** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow.



8. In the **Widgets** pane, move your mouse over the **Classic Menu – Horizontal** widget. Hold down the mouse button, and drag the **Classic Menu – Horizontal** widget on the wireframe at coordinates (0,0).
9. Click on the menu item labeled **File** to be selected, and type `Item 1`.
10. Click on the menu item labeled **Edit**, and type `Item 2`.
11. Click on the menu item labeled **View**, and type `Item 3`.
12. Right-click on the third menu item, and left-click on **Add Menu Item After** in the submenu.
13. Click on the fourth menu item, and type `Item 4`.
14. Right-click on the fourth menu item, and click on **Add Menu Item After** in the submenu.
15. Click on the fifth menu item, and type `Item 5`.

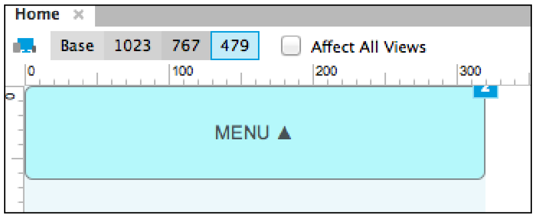
Click on the **767** adaptive view shown near the top of the wireframe pane. You will now perform the following steps to create an adaptive view as shown in the following screenshot:



1. In the main menu, click on **Edit**, and click on **Select All** from the drop-down menu. Right-click on the grey highlighted border, and click on **Unplace from View** to delete **Classic Menu – Horizontal**.
2. In the **Widgets** pane, move your mouse over the **Dynamic Panel** widget. Hold down the mouse button, and drag the **Dynamic Panel** widget on the wireframe at (0,0).
3. With the **Dynamic Panel** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Dynamic Panel** widget. Enter `480` in the width field and `300` in the height field.
4. Click on the **Dynamic Panel** widget in the **Widget Interactions and Notes** pane, and left-click on **Add Case**.
5. With the **Dynamic Panel** widget selected in the **Widget Interaction and Notes** pane, double-click on the **OnClick** interaction.
6. In the **Case Editor** pop up, click on **Set Panel State**.
7. Under **Configure actions**, click on the **This Widget** checkbox to select.
8. Click on the **Select the state** drop-down menu, and click on **Next**. Click on the **Wrap from last to first** checkbox to select.
9. Left-click on the **OK** button.
10. Double-click on **State 1** under **Dynamic Panel** in the **Widget Manager** to open **State 1** in the wireframe pane.
11. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (0,0).
12. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `480` in the width field and `50` in the height field.
13. With the **Button Shape** widget selected, type `MENU` to rename the **Button Shape** widget. You can also add any additional characters to indicate the menu state as well (that is, the up and down arrows).

14. With the **Button Shape** widget selected, in the **Widget Properties and Style** pane in the **Fill, Lines, + Borders** flyout menu, click on the paint bucket icon, and select a fill color.
15. Double-click on **State 2** under **Dynamic Panel** in the **Widget Manager** to open **State 2** in the wireframe pane.
16. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (0,0).
17. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `480` in the width field and `50` in the height field.
18. With the **Button Shape** widget selected, type `MENU` to rename the **Button Shape** widget. You can also add any additional characters to indicate the menu state as well (that is, up and down arrows).
19. With the **Button Shape** widget selected in the **Widget Properties and Style** pane in the **Fill, Lines, + Borders** flyout menu, click on the paint bucket icon, and select a fill color.
20. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (50,0).
21. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `480` in the width field and `50` in the height field.
22. With the **Button Shape** widget selected, type `ITEM 1` to change the **Button Shape** widget's displayed text.
23. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (100,0) changing the width to `480` and height to `50`. With the **Button Shape** widget selected, type `ITEM 2` to change the **Button Shape** widget's displayed text.
24. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (150,0) changing the width to `480` and height to `50`. With the **Button Shape** widget selected, type `ITEM 3` to change the **Button Shape** widget's displayed text.
25. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (200,0) changing the width to `480` and height to `50`. With the **Button Shape** widget selected, type `ITEM 4` to change the **Button Shape** widget's displayed text.
26. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (250,0) changing the width to `480` and height to `50`. With the **Button Shape** widget selected, type `ITEM 5` to change the **Button Shape** widget's displayed text.

Click on the **479** adaptive view shown near the top of the wireframe pane. You will now perform the following steps to create an adaptive view as shown in the following screenshot:



1. In the main menu, click on **Edit**, and click on **Select All** from the drop-down menu. Right-click on the grey highlighted border, and click on **Unplace from View** to delete **Classic Menu – Horizontal**.
2. In the **Widgets** pane, move your mouse over the **Dynamic Panel** widget. Hold down the mouse button, and drag the **Dynamic Panel** widget on the wireframe at (0,0).
3. With the **Dynamic Panel** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Dynamic Panel** widget. Enter `320` in the width field and `390` in the height field.
4. Click on the **Dynamic Panel** in the **Widget Manager** pane and click on **Add Case**.
5. With the **Dynamic Panel** widget selected, in the **Widget Interaction and Notes** pane, double-click on the **OnClick** interaction.
6. In the **Case Editor** pop up, click on **Set Panel State**.
7. Under **Configure actions**, click on the **This Widget** checkbox to select.
8. Click on the **Select the state** drop-down menu, and click on **Next**. Click on the **Wrap from last to first** checkbox to select.
9. Click on the **OK** button.
10. Double-click on **State 1** under the **Dynamic Panel** in the **Widget Manager** pane to open **State 1** in the wireframe pane.
11. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (0,0).
12. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `320` in the width field and `65` in the height field.
13. With the **Button Shape** widget selected, type `MENU` to rename the **Button Shape** widget. You can also add any additional characters to indicate the menu state as well (that is, the up and down arrows).
14. With the **Button Shape** widget selected, in the **Widget Properties and Style** pane in the **Fill, Lines, + Borders** flyout menu, click on the paint bucket icon, and select a fill color.

15. Double-click on **State 2** under **Dynamic Panel** in the **Widget Manager** to open **State 2** in the wireframe pane.
16. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (0,0).
17. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `320` in the width field and `65` in the height field.
18. With the **Button Shape** widget selected, type `MENU` to rename the **Button Shape** widget. You can also add any additional characters to indicate the menu state as well (that is, the up and down arrows).
19. With the **Button Shape** widget selected, in the **Widget Properties and Style** pane in the **Fill, Lines, + Borders** flyout menu click the paint bucket icon, and select a fill color.
20. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (50,0).
21. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `320` in the width field and `65` in the height field.
22. With the **Button Shape** widget selected, type `ITEM 1` to change the **Button Shape** widget's displayed text.
23. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (100,0) changing the width to `320` and height to `65`. With the **Button Shape** widget selected, type `ITEM 2` to change the **Button Shape** widget's displayed text.
24. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (150,0) changing the width to `320` and height to `65`. With the **Button Shape** widget selected, type `ITEM 3` to change the **Button Shape** widget's displayed text.
25. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (200,0) changing the width to `320` and height to `65`. With the **Button Shape** widget selected, type `ITEM 4` to change the **Button Shape** widget's displayed text.
26. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (250,0) changing the width to 320 and height to 65. With the **Button Shape** widget selected, type ITEM 5 to change the **Button Shape** widget's displayed text.

You can now choose to preview, or save a copy of, the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish** and then clicking on **Generate HTML Files...**. Choose the location in which you would like to store the prototype and click on the **Generate** button.

How it works...

For this recipe, you learned how to use Axure RP 7 adaptive views to make a navigation menu appear to behave in a responsive fashion. You defined breakpoints just below your target viewport sizes. Next, you adjusted each navigation menu by optimizing the menu's presentation for the smallest viewport shown for a given view. To make the prototype feel more like a responsive layout, you also set the page alignment to center in the **Page Style** pane.

Building a responsive home page prototype

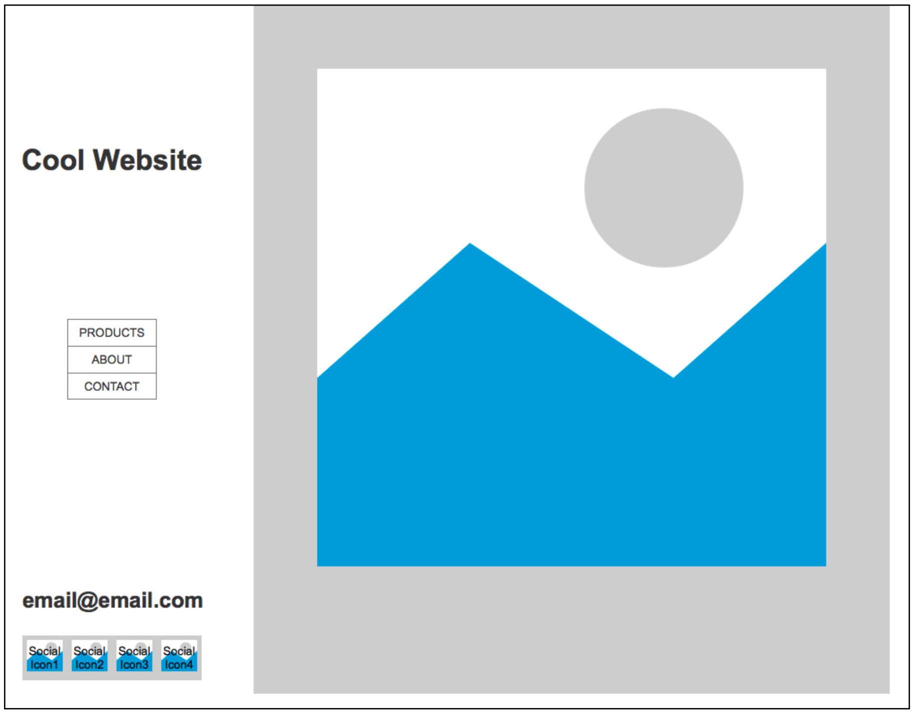
Using Axure RP 7's adaptive views, you can create a home page prototype that appears to be responsive. In this recipe, you will create an adaptive home page using adaptive views.

How to do it...

Perform the following steps to build a responsive home page prototype:

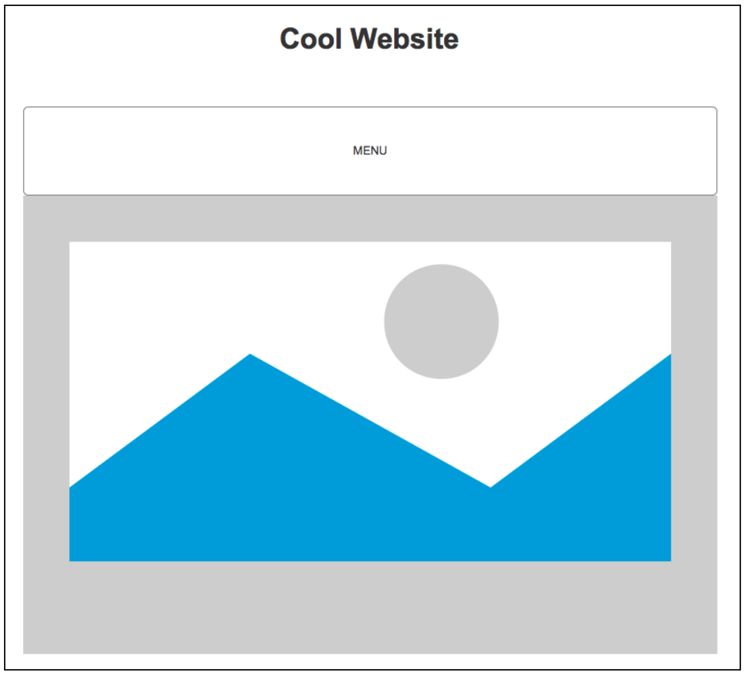
1. Start Axure and under **Create New** select **RP File**.
2. In the main menu, click on **Project**, and then either click on **Adaptive Views...**, or click on the **Manage Adaptive Views** icon in the wireframe pane.
3. In the **Adaptive Views** dialog box, left-click on the green *plus* sign to add a view. Click on the **Presets** drop-down menu, and click on the **Landscape Tablet** preset. Change the **Width** to `1023`.
4. Repeat step 3, and add a view for **Portrait Tablet**. Change the **Width** field to `767`. Click on the **OK** button.
5. Repeat step 3, and add a view for **Landscape Phone**. Change the **Width** field to `479`. Click on the **OK** button.
6. Under the wireframe pane, click on the **Page Style** tab. In the **Page Style** pane, click on the **Page Align** icon on the right. Page content will now be centered for the browser only.
7. The **Base** view will be shown when the viewport or browser window is sized at 1024 pixels and above. Click on the **Base** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow.

8. You will now perform the following steps to create an adaptive view as shown in the following screenshot:



9. In the **Widgets** pane, move your mouse over the **Rectangle** widget. Hold down the mouse button, and drag the **Rectangle** widget on the wireframe at (0,0).
10. Change the width and height to `314` and `768` respectively.
11. In the **Widgets** pane, mouse over the **Image** widget. Hold down the mouse button, and drag the **Image** widget on the wireframe at (314,0).
12. Change the width and height to `710` and `768` respectively.
13. In the **Widgets** pane, mouse over the **Heading 1** widget. Hold down the mouse button, and drag the **Heading 1** widget on the wireframe at (55,155).
14. Click on the **Heading 1** widget, and type `Cool Website`.
15. In the **Widgets** pane, move your mouse over the **Classic Menu – Vertical** widget. Hold down the mouse button, and drag the **Classic Menu – Vertical** widget on the wireframe at (106,350).
16. Click on the first menu item, and type `PRODUCTS`.
17. Click on the second menu item, and type `ABOUT`.
18. Click on the third menu item, and type `CONTACT`.

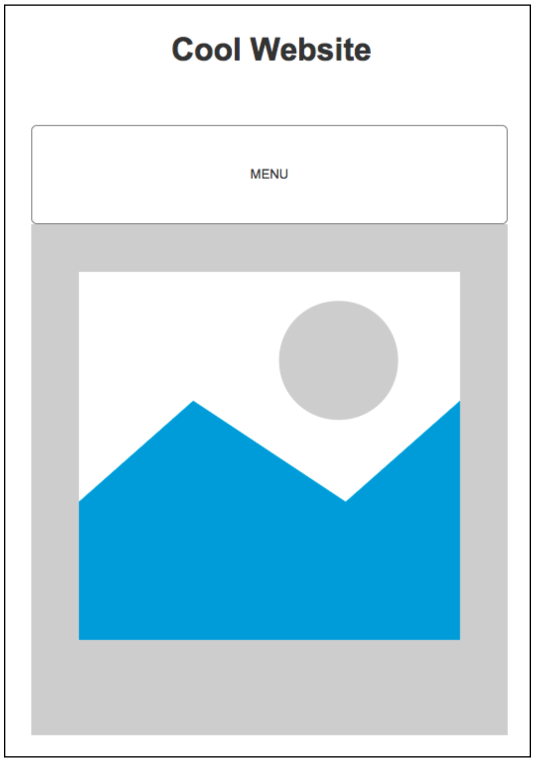
19. In the **Widgets** pane, move your mouse over the **Heading 2** widget. Hold down the mouse button, and drag the **Heading 2** widget on the wireframe at (56,650).
20. Click on the **Heading 2** widget, and type `email@email.com`.
21. In the **Widgets** pane, move your mouse over the **Image** widget. Hold down the mouse button, and drag the **Image** widget on the wireframe at (56,703).
22. Type `Social Icon1`.
23. In the **Widgets** pane, move your mouse over the **Image** widget. Hold down the mouse button, and drag the **Image** widget on the wireframe at (106,703).
24. Type `Social Icon2`.
25. In the **Widgets** pane, move your mouse over the **Image** widget. Hold down the mouse button, and drag the **Image** widget on the wireframe at (156,703).
26. Type `Social Icon3`.
27. In the **Widgets** pane, move your mouse over the **Image** widget. Hold down the mouse button, and drag the **Image** widget on the wireframe at (206,703).
28. Type `Social Icon4`.
29. Click on the **1023** adaptive view shown near the top of the wireframe pane. You will now perform the steps mentioned after the following screenshot to create an adaptive view as shown:



1. In the main menu, click on **Edit**, and then click on **Select All** from the drop-down menu.
2. In the main menu, click on **Edit**, and click on **Delete** from the drop-down menu to delete all widgets on the **Wireframe** pane.
3. In the **Widgets** pane, move your mouse over the **Rectangle** widget. Hold down the mouse button, and drag the **Rectangle** widget on the wireframe at (0,0).
4. With the **Rectangle** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Rectangle** widget. Enter `780` in the width field and `150` in the height field.
5. In the **Widgets** pane, move your mouse over the **Heading 1** widget. Hold down the mouse button, and drag the **Heading 1** widget on the wireframe at (288,56).
6. Click on the **Heading 1** widget, and type `Cool Website`.
7. In the **Widgets** pane, move your mouse over the **Dynamic Panel** widget. Hold down the mouse button, and drag the **Dynamic Panel** widget on the wireframe at (0,150).
8. With the **Dynamic Panel** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Dynamic Panel** widget. Enter `780` in the width field and `400` in the height field.
9. Click on **Dynamic Panel** in the **Widget Manager** pane, and click on **Add State.**
10. With the **Dynamic Panel** widget selected, in the **Widget Interaction and Notes** pane, double-click on the **OnClick** interaction.
11. In the **Case Editor** pop up, click on **Set Panel State**.
12. Under **Configure actions**, click on the **This widget** checkbox.
13. Click on the **Select the state** drop-down menu, and click on **Next**. Click on the **Wrap from last to first** checkbox.
14. Click on the **OK** button.
15. Double-click on **State 1** under **Dynamic Panel** in the **Widget Manager** to open **State 1** in the wireframe pane.
16. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (0,0).

17. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `780` in the width field and `100` in the height field.
18. With the **Button Shape** widget selected, type `MENU` to rename the **Button Shape** widget. You can also add any additional characters to indicate the menu state as well (that is, the up and down arrows).
19. Double-click on **State 2** under **Dynamic Panel** in the **Widget Manager** to open **State 1** in the wireframe pane.
20. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (0,0).
21. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `780` in the width field and `100` in the height field.
22. With the **Button Shape** widget selected, type `MENU` to rename the **Button Shape** widget. You can also add any additional character to indicate the menu state as well (that is, the up and down arrows).
23. In the **Widgets** pane, move your mouse over the **Button Shape** widget. Hold down the mouse button, and drag the **Button Shape** widget on the wireframe at (100,0).
24. With the **Button Shape** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Button Shape** widget. Enter `780` in the width field and `100` in the height field.
25. With the **Button Shape** widget selected, type `PRODUCTS` to rename the **Button Shape** widget.
26. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (200,0) changing the width to `780` and height to `100`. With the **Button Shape** widget selected, type ABOUT to change the **Button Shape** widget's displayed text.
27. While holding down the mouse button, drag the **Button Shape** widget and place at coordinates (300,0) changing the width to `780` and height to `100`. With the **Button Shape** widget selected, type CONTACT to change the **Button Shape** widget's displayed text.

30. Click on the **767** adaptive view shown near the top of the wireframe pane. You will now perform the steps mentioned after the following screenshot to create an adaptive view as shown:



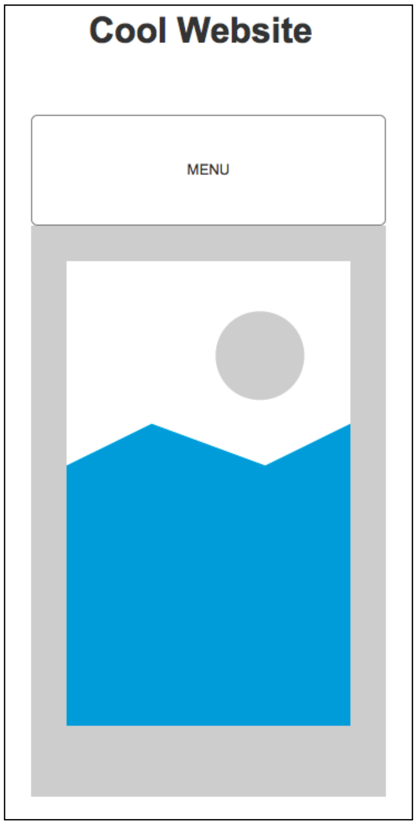
1. In the wireframe pane, click on the **Rectangle** widget in the wireframe at (0,0).
2. With the **Rectangle** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Rectangle** widget. Enter `480` in the width field and `150` in the height field.
3. In the **Widgets** pane, click on the **Heading 1** widget in the wireframe at (288,56).
4. With the **Heading 1** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Heading 1** widget. Enter `141` in the width field and `56` in the height field.
5. In the wireframe pane, click on the **Dynamic Panel** widget in the wireframe at (0,150).

6. With the **Dynamic Panel** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Dynamic Panel** widget. Enter `480` in the width field and `400` in the height field.
7. Click on **Dynamic Panel** in the **Widget Manager** pane, and click on **Add State**.
8. In the **Widget Manager** pane, double-click on **State 1** in the active **Dynamic Panel** widget.

[Active widgets will be in black. Widgets that have been deleted from a specific preset in the adaptive view will be in red.]

9. In the wireframe pane, click on the **Rectangle** widget in the wireframe at (0,0).
10. With the **Rectangle** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Rectangle** widget. Enter `480` in the width field and `100` in the height field.
11. In the **Widget Manager** pane, double-click on **State 2** in the active **Dynamic Panel** widget.
12. In the wireframe pane, click on the **Rectangle** widget in the wireframe at (0,0).
13. With the **Rectangle** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Rectangle** widget. Enter `480` in the width field and `100` in the height field.
14. In the wireframe pane, click on the remaining three rectangle widgets in the wireframe at (100,0), (200,0), and (300,0), and change the width field on each widget to `480`.
15. In the **Sitemap** pane, double-click on the **Home** page. This will open the **767** adaptive view for the **Home** page.
16. In the wireframe pane, click on the **Image** widget in the wireframe at (0,250).
17. With the **Image** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Image** widget. Enter `480` in the width field and `515` in the height field.

31. Click on the **479** adaptive view shown near the top of the wireframe pane. You will now perform the steps mentioned after the following screenshot to create an adaptive view as shown:



1. In the wireframe pane, click on the **Rectangle** widget in the wireframe at (0,0).
2. With the **Rectangle** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Rectangle** widget. Enter `320` in the width field and `150` in the height field.
3. In the **Widgets** pane, click on the **Heading 1** widget in the wireframe at (52,56).
4. With the **Heading 1** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Heading 1** widget. Enter `141` in the width field and `56` in the height field.

5. In the wireframe pane, click on the **Dynamic Panel** widget in the wireframe at (0,150) to select.
6. With the **Dynamic Panel** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Dynamic Panel** widget. Enter `320` in the width field and `400` in the height field.
7. Click on **Dynamic Panel** in the **Widget Manager** pane, and then click on **Add State**.
8. In the **Widget Manager** pane, double-click on **State 1** in the active **Dynamic Panel** widget.
9. In the wireframe pane, click on the **Rectangle** widget in the wireframe at (0,0).
10. With the **Rectangle** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Rectangle** widget. Enter `320` in the width field and `100` in the height field.
11. In the **Widget Manager** pane, double-click on **State 2** in the active **Dynamic Panel** widget.
12. In the wireframe pane, click on the **Rectangle** widget in the wireframe at (0,0).
13. With the **Rectangle** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Rectangle** widget. Enter `320` in the width field and `100` in the height field.
14. In the wireframe pane, click on the remaining three rectangle widgets in the wireframe at (100,0), (200,0), and (300,0), and change the width field on each widget to `320`.
15. In the **Sitemap** pane, double-click on the **Home** page. This will open the **479** adaptive view for the **Home** page.
16. In the wireframe pane, click on the **Image** widget in the wireframe at (0,250).
17. With the **Image** widget selected, to the top right-hand side corner of the wireframe, you will see two fields marked **w:** and **h:**. These are for the width and height of the **Image** widget. Enter `320` in the width field and `515` in the height field.

32. You can now choose to preview or save a copy of the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish**, and then clicking on **Generate HTML Files...**. Choose the location in which you would like to store the prototype, and click on the **Generate** button.

How it works...

For this recipe, you learned how to use Axure RP 7 adaptive views to make a home page appear to behave in a responsive fashion. You defined breakpoints just below your target viewport sizes. Next, you adjusted each widget on the home page by optimizing the widgets presentation for the smallest viewport shown for a given view. To make the prototype feel more like a responsive layout, you also set the page alignment to center in the **Page Style** pane.

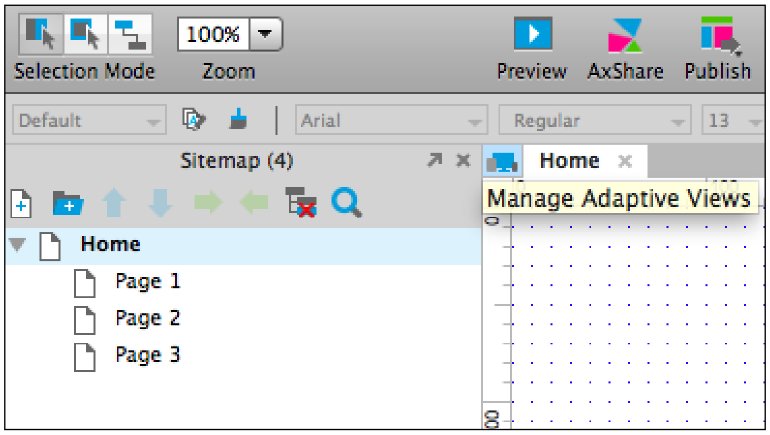
Building a responsive gallery grid

As mentioned earlier, Axure RP 7 enables you to create adaptive prototypes through adaptive views. In this recipe, you will start with a desktop view as your Base view and design a responsive gallery grid for each subsequent breakpoint.

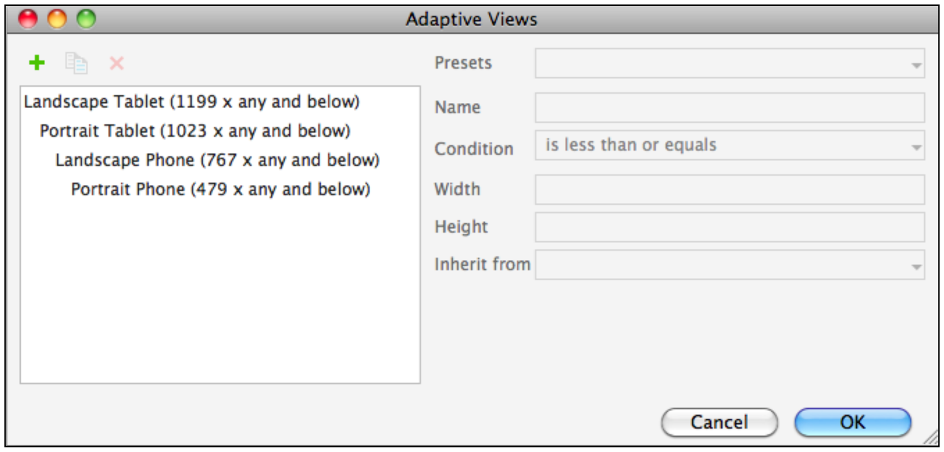
How to do it...

Perform the following steps to design a responsive gallery grid:

1. Start Axure and under **Create New** select **RP File**.
2. In the main menu, click on **Project**, and then either click on **Adaptive Views...**, or click on the **Manage Adaptive Views** icon in the wireframe pane.



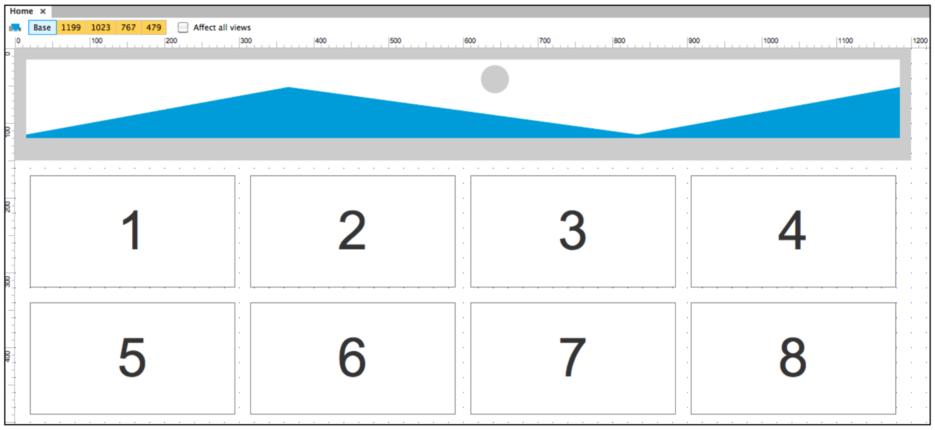
3. In the **Adaptive Views** dialog box, click on the green *plus* sign to add a view. Click on the **Presets** drop-down menu, and then click on the **Landscape Tablet** preset. Change the width to `1199`.
4. Repeat step 3, and add views for **Portrait Tablet**, **Landscape Phone**, and **Portrait Phone** modes, changing each corresponding view width to `1023`, `767`, and `479`. Click on the **OK** button.


Adaptive Views

Landscape Tablet (1199 x any and below)
Portrait Tablet (1023 x any and below)
Landscape Phone (767 x any and below)
Portrait Phone (479 x any and below)

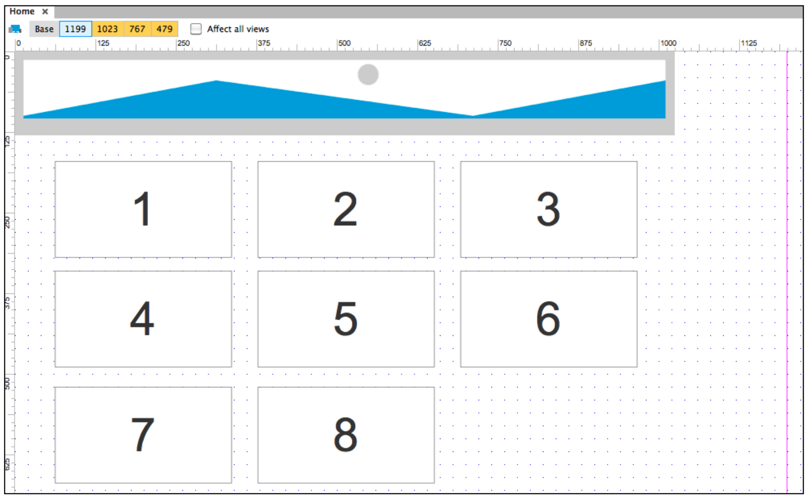
Presets:
Name:
Condition: is less than or equals
Width:
Height:
Inherit from:

5. Under the wireframe pane, click on the **Page Style** tab. In the **Page Style** pane, click on the **Page Align** icon on the right. Page content will now be centered for the browser only.
6. The **Base** view will be shown when the viewport or browser window is sized at 1200 pixels and above. Click on the **Base** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow. Now place and size the widgets you would like to show for this view as shown in the following screenshot:

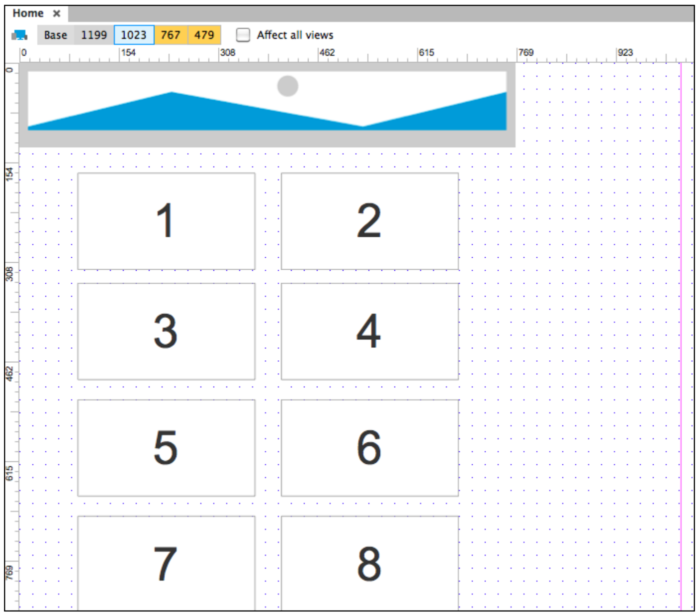


7. The **1199** view will be shown when the viewport or browser window is sized from 1199 pixels to 1024 pixels. For this view, we will place our design optimized for 1024 pixels. Click on the **1199** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow.

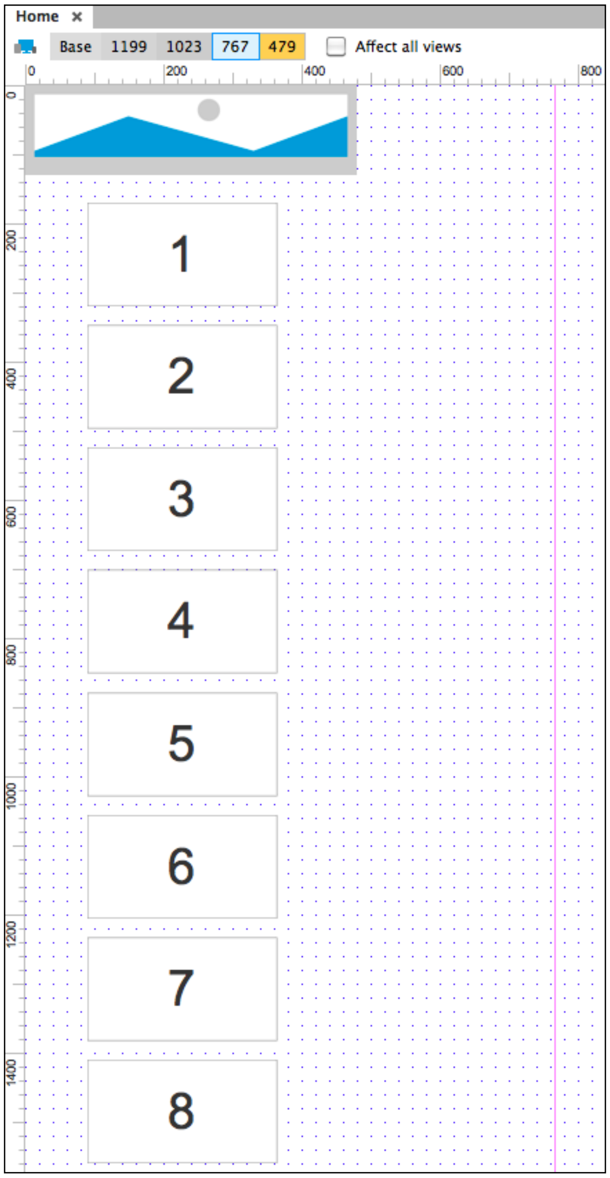
Now place and size the widgets you would like to show for this view as shown in the following screenshot:



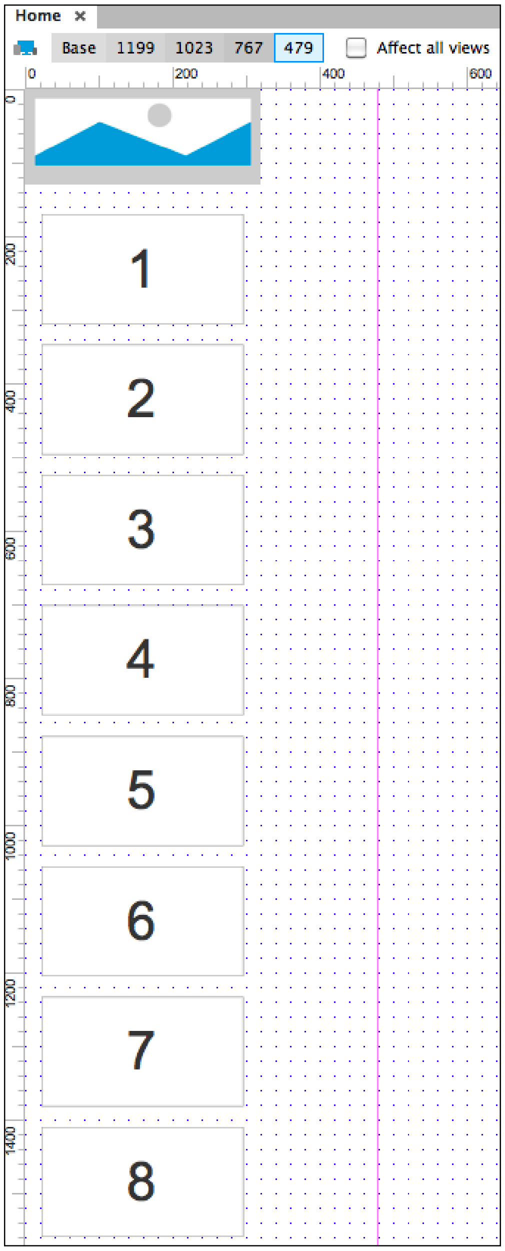
8. The **1023** view will be shown when the viewport or browser window is sized from 1023 pixels to 768 pixels. For this view, we will place our design optimized for 768 pixels. Click on the **1023** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow. Now place and size the widgets you would like to show for this view as shown in the following screenshot:



9. The **767** view will be shown when the viewport or browser window is sized from 767 pixels to 480 pixels. For this view, we will place our design optimized for 480 pixels. Click on the **767** view icon in the header of the wireframe pane. The selected view will turn blue and the views that inherit from the selected view will be yellow. Now place and size the widgets you would like to show for this view as shown in the following screenshot:



10. The **479** view will be shown when the viewport or browser window is sized from 479 pixels to 0 pixels. For this view, we will place our design optimized for 320 pixels. Click on the **479** view icon in the header of the wireframe pane. The selected view will turn blue, and the views that inherit from the selected view will be yellow. Now place and size the widgets you would like to show for this view.



11. You can now choose to preview, or save a copy of, the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and select **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish**, and then clicking on **Generate HTML Files...**. Choose the location in which you would like to store the prototype, and click on the **Generate** button.

How it works...

For this recipe, you learned how to lay out a gallery grid that appears to behave in a responsive fashion. To make the prototype feel more like a responsive layout, you also set the page alignment to center in the **Page Style** pane.

Next, you used adaptive views and defined breakpoints just below your target viewport sizes. Next, you adjusted each design by optimizing the content for the smallest viewport shown for a given view.

Building a responsive landing page

Using Axure RP 7's adaptive views, you can create a landing page prototype that appears to be responsive. In this recipe, you will create an adaptive landing page using adaptive views.

Getting ready

For this recipe, you will need the Axure RP file from the responsive home page prototype.

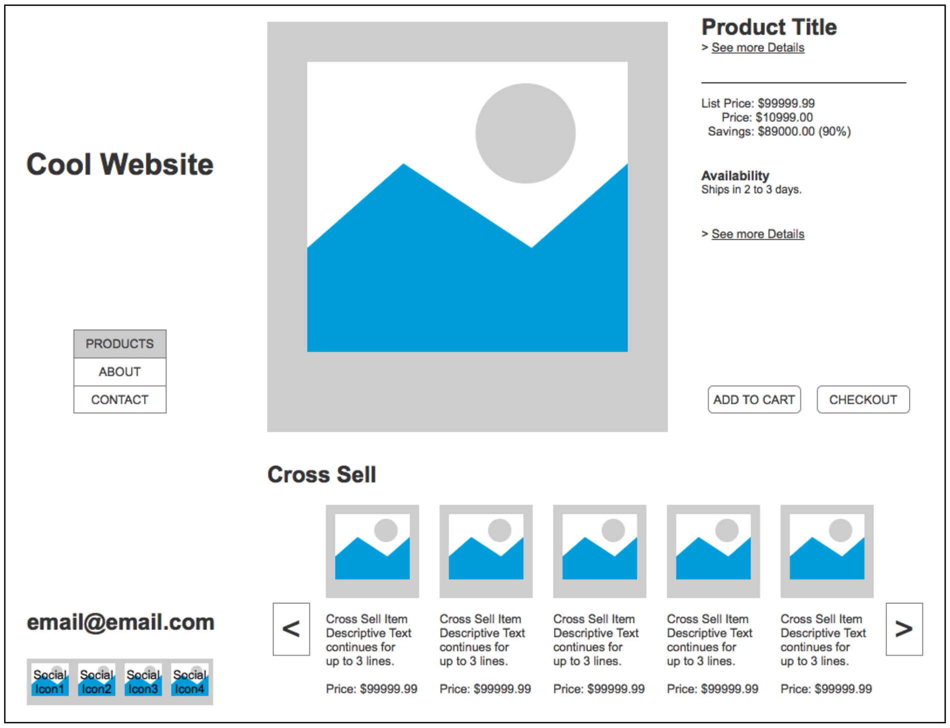
How to do it...

Perform the following steps to build a responsive landing page:

1. Start Axure RP, and open the Axure RP file you created in the responsive home page prototype.
2. Click on the **Base** view icon in the header of the wireframe pane. The selected view will turn blue and the views that inherit from the selected view will be yellow.
3. In the wireframe pane, click on the **Image** widget at (314,0). Move the **Image** widget to (314,20), and change the width and height to 430 by 440.
4. In the **Widgets** pane, move your mouse over the **Heading 2** widget. Hold down the left mouse button, and drag the **Heading 2** widget on the wireframe at (314,492).
5. Click on the **Heading 2** widget, and type `Cross Sell`.
6. In the **Widgets** pane, drag the **Rectangle** widget, and place it on the wireframe at (320,643). Type `<`, and increase the font to **36** in the **Widget Properties and Style** pane.

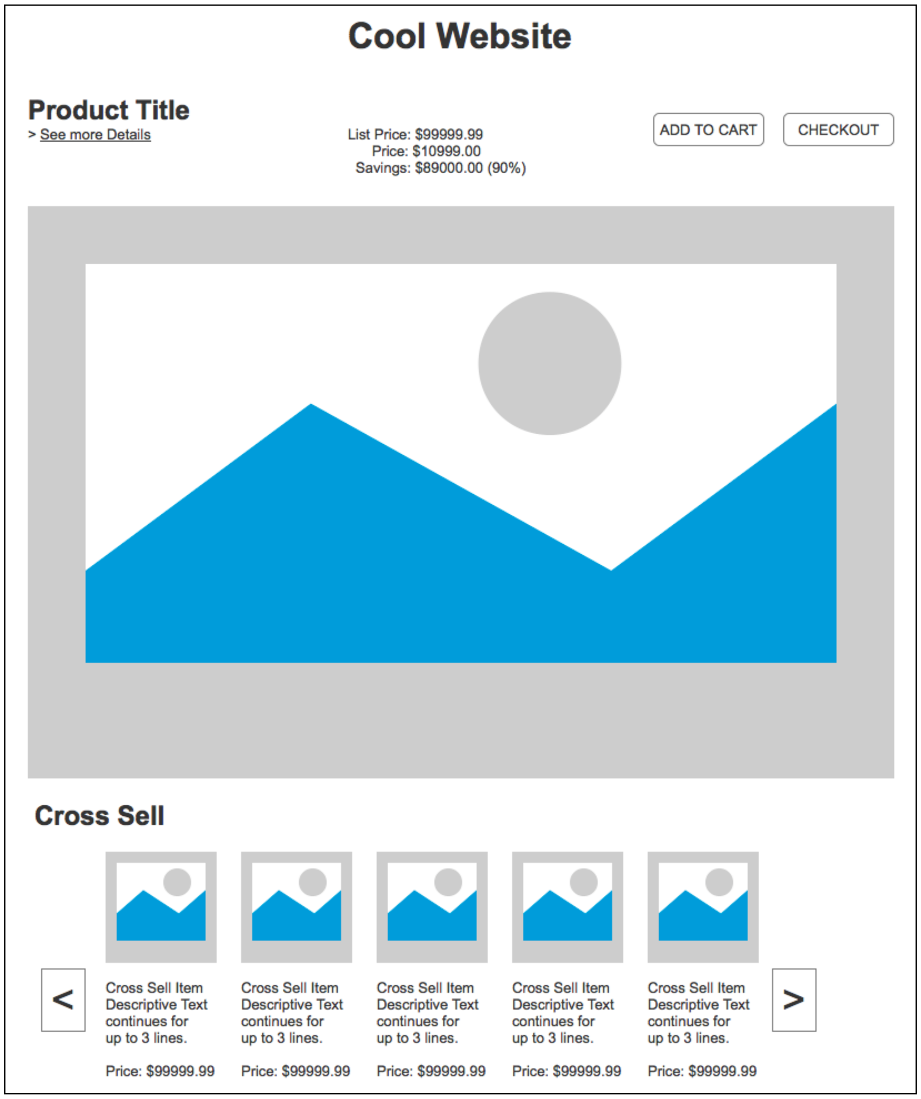
7. Change the width and height to `40` and `57`.
8. In the **Widgets** pane, move your mouse over the **Rectangle** widget. Hold down the mouse button, and drag the **Rectangle** widget on the wireframe at (978,643). Type `>`, and increase the font to `36` in the **Widget Properties and Style** pane.
9. Change the width and height to `40` and `57`.
10. In the **Widgets** pane, move your mouse over the **Rectangle** widget. Hold down the mouse button, and drag the **Rectangle** widget on the wireframe at (978,643). Type `>`, and increase the font to **36** in the **Widget Properties and Style** pane.
11. Change the width and height to `40` and `57`.
12. In the **Widgets** pane, move your mouse over the **Image** widget. Hold down the mouse button, and drag the **Image** widget on the wireframe at (377,538).
13. Change the width and height to `100` and `100`.
14. Repeat steps 12 to 13, placing four more image widgets at (499,538), (621,538), (743,538), and (865,538).
15. In the **Widgets** pane, move your mouse over the **Label** widget. Hold down the mouse button, and drag the **Label** widget on the wireframe at (314,492).
16. Click on the **Label** widget, and type descriptive text and a price for the item.
17. With the **Label** widget selected, change the width and height to `100` and `90` respectively.
18. Copy the **Label** widget you created in step 15 four times placing the copies at (499,653), (621,653), (743,653), and (865,653).
19. In the **Widgets** pane, move your mouse over the **Heading 1** widget. Hold down the mouse button, and drag the **Heading 1** widget on the wireframe at (780,12).
20. Click on the **Heading 1** widget, and type `Product Title`.
21. In the **Widgets** pane, move your mouse over the **Label** widget. Hold down the mouse button, and drag the **Label** widget on the wireframe at (780,40). Type `Secondary information about the product`.
22. Change the width and height to `264` and `460`.
23. In the **Widgets** pane, move your mouse over the **Horizontal Line** widget. Hold down the mouse button, and drag the **Horizontal Line** widget on the wireframe at (780,80). Type `Secondary information about the product`.
24. Change the width to `220`.
25. In the **Widgets** pane, move your mouse over the **Label** widget. Hold down the mouse button, and drag the **Label** widget on the wireframe at (780,100). Type `List Price`, `Price`, and `Savings` labels and values.
26. Change the width and height to `220` and `45`.

27. In the **Widgets** pane, move your mouse over the **Label** widget. Hold down the mouse button, and drag the **Label** widget on the wireframe at (780,177). Type `Availability`, and a brief shipping time.
28. Change the width and height to `110` and `30`.
29. In the **Widgets** pane, move your mouse over the **Label** widget. Hold down the mouse button, and drag the **Label** widget on the wireframe at (780,240). Type `Description`, brief selling points, and a link to see more details.
30. Change the width and height to `179` and `121`.
31. Your base product landing page should look similar to this screenshot:

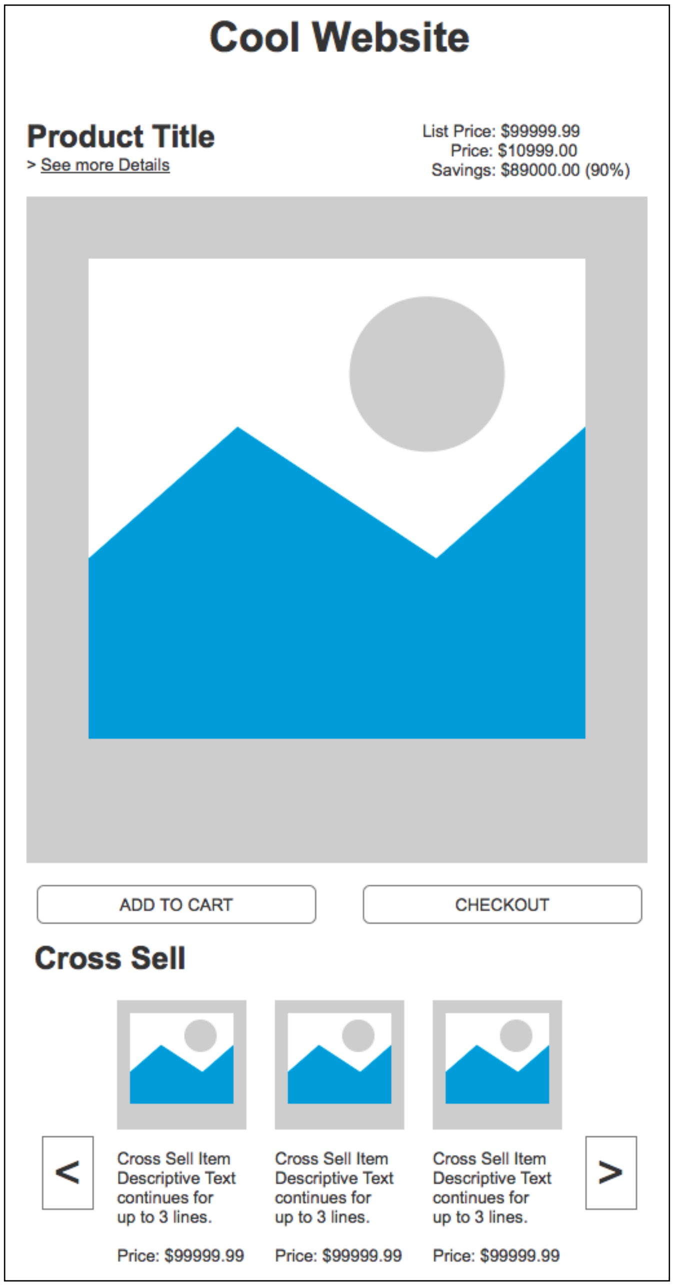


32. Add an **Add to Cart** button widget at (787,410) and a **Checkout** button widget at (904,410).
33. Click on the **1023** adaptive view shown near the top of the **Wireframe pane**.
34. At (0,0), you should see the rectangle widget with the text **Cool Website.**
35. To delete the **Classic Menu – Vertical** widget at (106,350), click on the widget to select. Right-click on the grey highlighted border, and click on **Unplace from View** to delete the **Classic Menu – Horizontal** widget.
36. Delete the **Label** widget at (56,650).

37. Delete the **Image** widgets at (56,703), (106,703), (156,703), and (206,703).
38. Arrange the remaining widgets optimizing the copy and information for 780 width. The following screenshot shows an example layout:

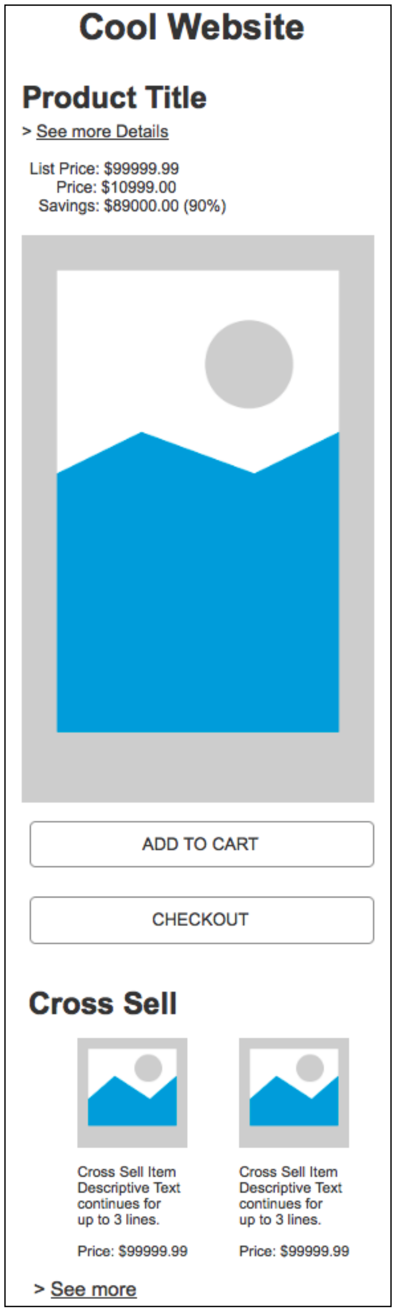


39. Click on the **767** adaptive view shown near the top of the wireframe pane.
40. Arrange the remaining widgets, optimizing the copy and information for 480 width. The following screenshot shows an example layout:


Cool Website
Product Title
> See more Details
List Price: $99999.99
Price: $10999.00
Savings: $89000.00 (90%)
ADD TO CART
CHECKOUT
Cross Sell
<
Cross Sell Item Descriptive Text continues for up to 3 lines.
Price: $99999.99
Cross Sell Item Descriptive Text continues for up to 3 lines.
Price: $99999.99
Cross Sell Item Descriptive Text continues for up to 3 lines.
Price: $99999.99
>

41. Click on the **479** adaptive view shown near the top of the wireframe pane.

42. Arrange the remaining widgets, optimizing the copy and information for 320 width. The following screenshot shows an example layout:



43. You can now choose to preview, or save a copy of, the prototype. To preview the prototype, click on the **Preview** button in the toolbar. To save a copy of the prototype, click on the **Publish** button in the toolbar, and click on **Generate HTML Files...**. You can also generate the prototype by going to the main menu, clicking on **Publish**, and then clicking on **Generate HTML Files...**. Choose the location in which you would like to store the prototype, and click on the **Generate** button.

How it works...

For this recipe, you learned how to use Axure RP 7 adaptive views to make a landing page appear to behave in a responsive fashion. You defined breakpoints just below your target viewport sizes. Next, you adjusted each widget on the landing page by optimizing the widgets presentation for the smallest viewport shown for a given view.

There's more...

To make the prototype feel more like a responsive layout, remember to set the page alignment to center in the **Page Style** pane.

8
Prototyping for Mobile

In this chapter, we will discuss optimizing and running Axure prototypes on mobile devices. You will learn how to do the following:

- Using AxShare
- Setting the viewport for your target mobile device
- Creating a slideshow with the OnSwipeLeft and OnSwipeRight events
- Mimicking a native iOS application
- Leveraging dynamic panel scrolling for mobile devices
- Running your prototype natively on an iOS device
- Running your prototype natively on Android devices
- Running your prototype natively on Microsoft Surface and Windows 8 devices

Introduction

Axure is a great tool for prototyping mobile applications as well as **Adaptive Web Design** (**AWD**) and **Responsive Web Design** (**RWD**). You will learn how to set the viewport, use OnSwipe events, and how to make prototypes appear to run like apps on mobile devices.

Using AxShare

There are several options for viewing your prototypes on mobile devices. One way is to use Axure's AxShare service. With the free tier, you can host up to 10 prototypes. You can even enable discussions and password protection. Learn more at `http://www.axure.com/axshare`.

Getting ready

For this recipe, you will need an AxShare account and a mobile device.

How to do it...

We need to perform the following steps for using AxShare:

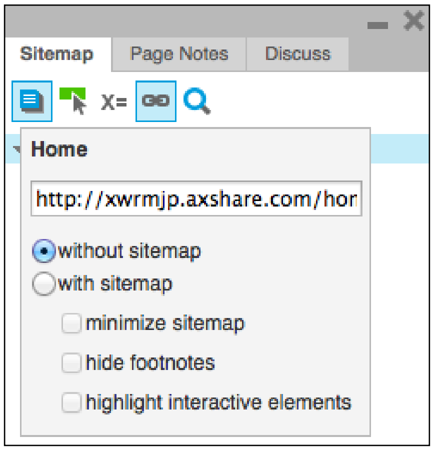
1. Start Axure and select **Open a Recent Item**, and open the RP document that you would like to share.

   > If you already have Axure open, select **File** in the main menu and then click on **Open** in the drop-down menu to open an existing RP document.

2. From the main menu, select **Publish** and click on **Publish to Axure Share...**.
3. In the **Publish to Axure Share** dialog box, click on **Existing Account** and enter your e-mail and password.

   > If you do not have an AxShare account, click on **Create Account**, enter your `email`, `password`, and click on the **I agree to the Share terms** checkbox.

4. To password protect the prototype, type a password in the **Password** field under the **Create a new project** pane.
5. On AxShare, the default directory is the **My Projects** folder. Click on the **...** button next to the **Folder** field to change the default directory.
6. To publish to AxShare, click on the **AxShare** button.
7. A pop-up window will open providing publishing status to AxShare and also the link to your prototype. Click on the blue **Copy** link to copy the link to your prototype, and paste the link onto a browser to view your prototype.
8. Next, you will copy the link without sitemap and view it on a mobile device. To copy the link without sitemap, refer to the following screenshot and perform the steps:
   1. Open the prototype link in a browser window.
   2. Click on the fourth icon that looks like a chain link.
   3. In the drop-down menu, click on the radio button next to the **without sitemap** label.
   4. Copy the link without sitemap.



http://xwrmjp.axshare.com/hor

- [✓ without sitemap]
- with sitemap
  - minimize sitemap
  - hide footnotes
  - highlight interactive elements

5. Next, open the link without sitemap on your mobile device to view the prototype.

How it works...

In this recipe, you learned how to share an Axure prototype using AxShare. To view the Axure prototype on your mobile device, you published the prototype to AxShare and viewed the link to the prototype on the mobile device.

Setting the viewport for your target mobile device

The viewport tag provides the size and scaling options for your target mobile device. Using the viewport tag helps the prototype to scale properly for your device.

Getting ready

For this recipe, you will need an AxShare account or a web server to load your prototype to and a mobile device.

All references to screen resolution and screen elements are in pixels.

How to do it...

In order to set the viewport, you need to perform the following steps:

1. Determine the mobile device you wish to build the prototype for.
2. Identify the screen resolution in pixels (that is, the height and width of the device).
3. Subtract the height of the address bar on your device from the device's height. This will be your visible device height.
4. Start Axure and under **Create New** select **RP File**.
5. From the main menu, navigate to **Arrange** | **Grid and Guides**, and click on **Create Guides...** to create guides.

   [Guides placed at the boundaries of the viewable area for a given device can assist you while placing widgets.]

6. Double-click on **Home** in the sitemap. Place the desired widgets on the wireframe.
7. From the main menu, select **Publish** and click on **Generate HTML Files...**.
8. In the **Generate HTML** dialog box, with the **General** menu item selected, note the directory where the prototype will be saved under **Destination folder for HTML files**,.
9. In the **Generate HTML** dialog box, click on **Mobile/Device**.
10. Click on the checkbox next to the **Include Viewport Tag** label and set the **Width** to **device-width**, **Initial Scale** to **1.0**, **Maximum Scale** to **1.0**, and **User Scalable** to **no**.
11. Click on the **Generate** button to generate your prototype.

    [The prototype should automatically open in a browser window. If it does not open, browse to your prototype directory (for example, **Destination folder** from step 8) and click on **start.html**.]

12. To view the prototype on your mobile device using AxShare, perform the following steps:
    1. To publish to AxShare, click on the **Publish to Axure Share** button.
    2. A pop-up window will open that provides publishing status to AxShare and also the link to your prototype. Click on the blue **copy** link to copy the link to your prototype, and paste the link into a browser to view your prototype.
    3. To copy the link without sitemap, with the prototype displayed in a browser window, click on the fourth icon that looks like a chain link.
    4. In the drop-down menu, click on the radio button next to the **without sitemap** label.
    5. Copy the link without sitemap.

13. Next, open link without sitemap on your mobile device to view it.

How it works...

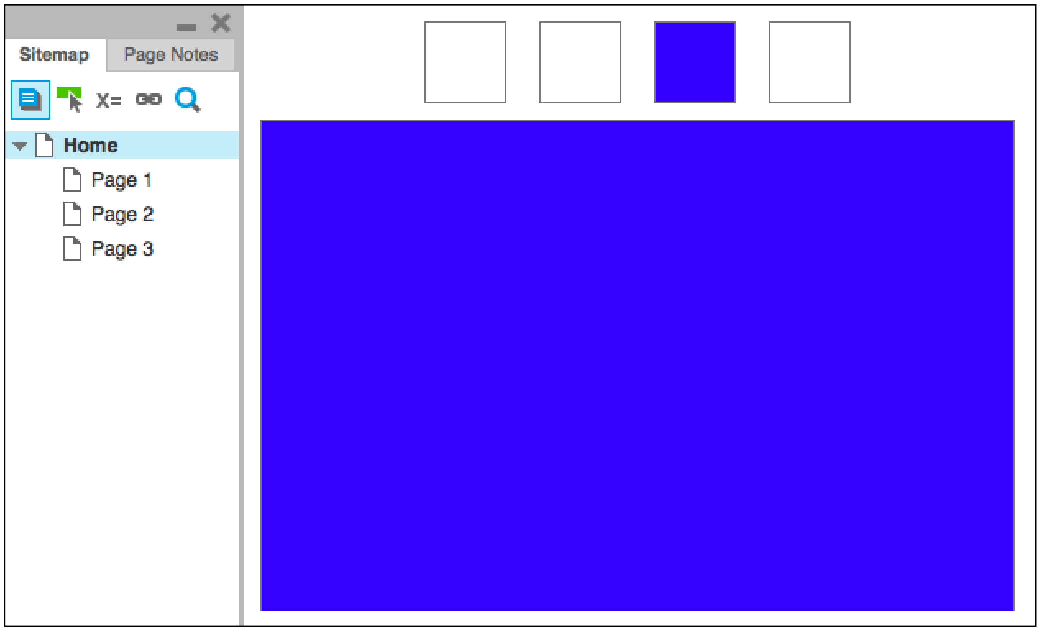
In this recipe, you learned how to set the viewport tag to provide size and scaling options for your Axure prototype. To view the Axure prototype on your mobile device, you will need to place the prototype on a web server, use AxShare, or view the prototype locally on the device.

Creating a slideshow with the OnSwipeLeft and OnSwipeRight events

The **OnSwipeLeft** and **OnSwipeRight** events are extremely useful while creating a prototype for mobile devices. In this recipe, you will use the **OnSwipeLeft** and **OnSwipeRight** events to create a slideshow for a mobile device.

Getting ready

For this recipe, you will need an AxShare account or a web server to load your prototype to and a mobile device. The prototype will look like the following screenshot:



How to do it...

For creating a slideshow, we need to perform the following steps:

1. Start Axure and under **Create New** select **RP File**.
2. From the main menu, navigate to **Arrange** | **Grid and Guides** and click on **Create Guides...** to create guides.
3. Double-click on the page labeled **Home** in the sitemap, and place a **Dynamic Panel** widget on the wireframe naming the widget `IndicatorPanel`.
4. You will now focus on building the indicator panel for the slideshow page. We need to perform the following steps for this:
    1. While holding down the left mouse button, drag the **Dynamic Panel** widget and place it at coordinates (10,0) on the wireframe.
    2. With the **Dynamic Panel** widget selected, change the width **w:** to **460** and the height **h:** to **70** in the toolbar.
    3. In the **Widget Interactions and Notes** pane, click on the **Dynamic Panel Name** field and type in `IndicatorPanel`.
    4. In **Widget Manager**, double-click on **State1**.
    5. While holding down the left mouse button, drag the **Rectangle** widget and place it at coordinates (100,10) on the wireframe. With the **Rectangle** widget selected, change the width **w:** to `50` and the height **h:** to `50` in the toolbar.
    6. Repeat the previous step to create three additional **Rectangle** widgets. Use the following table as a reference for placing the **Rectangle** widgets:

Coordinates
(170,10)
(240,10)
(310,10)

    7. In **Widget Manager**, right-click on **State1** and click on **Duplicate State** to create **State2**.
    8. Repeat the previous step twice for creating **State3** and **State4**. You should now have four states labeled **State1**, **State2**, **State3**, and **State4**.
    9. With the **Dynamic Panel** widget selected, double-click on **State1** in **Widget Manager**.
    10. Click on the first **Rectangle** widget at (100,10).

11. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.
12. To change the color of the **Rectangle** widget, enter the hex value `FF0000` in the field and press *Enter*, or click anywhere outside the drop-down menu.
13. With the **Dynamic Panel** widget selected, double-click on **State2** in the **Widget Manager** pane.
14. Click on the first **Rectangle** widget at (170,10).
15. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.
16. To change the color of the **Rectangle** widget, enter the hex value `008000` in the field and press *Enter*, or click anywhere outside the drop-down menu.
17. With the **Dynamic Panel** widget selected, double-click on **State3** in the **Widget Manager** pane.
18. Click on the first **Rectangle** widget at (240,10).
19. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.
20. To change the color of the **Rectangle** widget, enter the hex value `0000FF` in the field and press *Enter*, or click anywhere outside the drop-down menu.
21. With the **Dynamic Panel** widget selected, double-click on **State4** in the **Widget Manager** pane.
22. Click on the first **Rectangle** widget at (310,10).
23. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.
24. To change the color of the **Rectangle** widget, enter the hex value `800080` in the field and press *Enter*, or click anywhere outside the drop-down menu.

5. You will now focus on building the background panel for the slideshow page. We need to perform the following steps to do this:
    1. In the Sitemap, double-click on the **Home** page.
    2. While holding down the left mouse button, drag a new **Dynamic Panel** widget, and place it at coordinates (10,0) on the wireframe.
    3. With the **Dynamic Panel** widget selected, change the width **w:** to **460** and the height **h:** to **580** in the toolbar.
    4. In the **Widget Interactions and Notes** pane, click in the **Dynamic Panel Name** field and type in `BackgroundPanel`.

5. In **Widget Manager**, double-click on **State1**.
6. While holding down the left mouse button, drag the **Rectangle** widget and place it at coordinates (0,0) on the wireframe. With the **Rectangle** widget selected, change the width **w:** to `460` and the height **h:** to `730` in the toolbar.
7. In **Widget Manager**, right-click on **State1** and click on **Duplicate State** for creating **State2**.
8. Repeat the previous step twice to create **State3** and **State4**. You should now have four states labeled **State1**, **State2**, **State3**, and **State4**.
9. With the **Dynamic Panel** widget selected at (10,70), double-click on **State1** in **Widget Manager**.
10. Click on the **Rectangle** widget at (0,0).
11. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.
12. To change the color of the **Rectangle** widget, enter the hex value `FF0000` in the field and press *Enter*, or click anywhere outside the drop-down menu.
13. With the **Dynamic Panel** widget selected, double-click on **State2** in the **Widget Manager** pane.
14. Click on the **Rectangle** widget at (0,0).
15. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.
16. To change the color of the **Rectangle** widget, enter the hex value `008000` in the field and press *Enter*, or click anywhere outside the drop-down menu.
17. With the **Dynamic Panel** widget selected, double click on **State3** in **Widget Manager** pane.
18. Click on the first **Rectangle** widget at (0,0).
19. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.
20. To change the color of the **Rectangle** widget, enter the hex value `0000FF` in the field and press *Enter*, or click anywhere outside the drop down-menu.
21. With the **Dynamic Panel** widget selected, double-click on **State4** in the **Widget Manager** pane.
22. Click on the first **Rectangle** widget at (0,0).
23. In the **Widget Properties and Style** pane, scroll to the **Fill, Lines, + Borders** section. Click on the bucket icon to bring up the color chooser drop-down menu.

24. To change the color of the **Rectangle** widget, enter the hex value `800080` in the field and press *Enter*, or click anywhere outside the drop-down menu.
25. Double-click on **Home** in the Sitemap.

6. Click on the **Dynamic Panel** widget at (1 0,70). Perform the following steps:
    1. In the **Widget Interactions and Notes** pane, double-click on the **OnSwipeLeft** interaction.
    2. In the **Case Editor** popup, rename the case description to `SwipeLeft` in the **Case Description** field.
    3. In **Click to add actions**, navigate to **Dynamic Panels** | **Set Panel State**.
    4. In **Organize actions**, you will see the interaction description update to **Set Panel to State**.
    5. In **Configure actions**, click on the checkbox next to **This Widget**. Select the **state** drop-down and change it to **Next**. Click on the **Wrap from last to first** checkbox.
    6. In **Configure** actions, click on the checkbox next to **(Dynamic Panel)**. Select the **state** drop-down menu and change it to **Next**. Click on the **Wrap from last to first** checkbox. Click on the **Animate In** drop-down menu and change it to **slide left**. Click on the **Animate Out** drop-down menu and change it to **slide left**.
    7. In the **Widget Interactions and Notes** pane, double-click on the **OnSwipeRight** interaction.
    8. In the **Case Editor** pop up, rename the case description to `SwipeRight` in the **Case Description** field.
    9. In **Click to add actions**, navigate to **Dynamic Panels** | **Set Panel State**.
    10. In **Organize actions**, you will see the **Interaction description** update to **Set Panel to State**.
    11. In **Configure actions**, click the checkbox next to **Set IndicatorPanel**. Select the **state** drop-down menu and change to **Previous**. Click on the **Wrap from last to first** check box.
    12. In **Configure actions**, click on the checkbox next to **Set BackgroundPanel**. Select the state drop-down menu and change it to **Previous**. Click on **Wrap from first to last** the checkbox. Click on the **Animate In** drop-down menu and change it to **slide right**. Click on the **Animate Out** drop-down menu and change it to **slide left**.
7. Save the file as `MobileDemoOnSwipe.rp`.
8. From the main menu, navigate to **Publish** | **Generate HTML Files...**.
9. In the **Generate HTML** dialog box, with the **General** menu item selected, note the directory where the prototype will be saved under **Destination folder for HTML files**.

10. In the **Generate HTML** dialog box, click on **Mobile/Device**.
11. Click on the checkbox next to the **Include Viewport Tag** label, and set the **Width** to **device-width**, **Initial Scale** to **1.0**, **Maximum Scale** to **1.0**, and **User Scalable** to **no**.
12. Click on the **Generate** button to generate your prototype.
13. To view the prototype on your mobile device using AxShare, perform the following steps:
    1. To publish to AxShare, click on the **Publish to Axure Share** button.
    2. A pop-up window will open providing publishing status to AxShare and also the link to your prototype. Click on the blue **Copy** link to copy the link to your prototype, and paste the link into a browser to view your prototype.
    3. To copy the link without sitemap, with the prototype displayed in a browser window, click on the fourth icon that looks like a chain link.
    4. In the drop-down menu, click on the radio button next to the **without sitemap** label.
    5. Copy the link without sitemap.
14. Next, open the link without sitemap on your mobile device to view.

How it works...

For this recipe, you used two **Dynamic Panel** widgets with four states each. Swiping left or right on the **Dynamic Panel** widget caused the indicator panel to show the **Next** or **Previous** state.

Mimicking a native iOS application

An Axure prototype can also mimic a native iOS application. In this recipe, you will learn how to configure your prototype to appear to the user like a native iOS app.

Getting ready

For this recipe, you will need an AxShare account or a web server to load your prototype to and a mobile device.

How to do it...

For mimicking a native iOS app, we need to perform the following steps:

1. Start Axure and under **Create New** select **RP File**.
2. From the main menu, navigate to **Arrange** | **Grid and Guides** and click on **Create Guides...** to create guides.

3. Double-click on **Home** in the sitemap. Place desired widgets on the wireframe. Next, wireframe your mobile design.
4. From the main menu, select **Publish** and click on **Generate HTML Files...**.
5. In the **Generate HTML** dialog box, with the **General** menu item selected, , note the directory where the prototype will be saved under **Destination folder for HTML files**.
6. In the **Generate HTML** dialog box, click on **Mobile/Device** and perform the following steps:
    1. Click on the checkbox next to the **Include Viewport Tag** label and set the **Width** to **device-width**, **Initial Scale** to **1.0**, **Maximum Scale** to **1.0**, and **User Scalable** to **no**.
    2. Click on the checkbox next to **Hide Address Bar**.

    On iOS devices, this will hide the browser URL that makes the prototype appear like a native application.

    3. Click on the checkbox next to **Prevent vertical page scrolling**.

    This blocks the elastic scrolling, keeping your prototype window from being pulled from the edge of the screen.

7. In the **Generate HTML** dialog box, perform the following steps to explore the additional menu options:
    1. Click on the **Home screen icon** section. You can import a PNG icon of 114 x 114 pixels to be used as a shortcut application icon on the **Home** screen.
    2. Click on the **iOS Splash Screens** section to create splash screens that will be shown when the prototype loads. The sizes for the PNG splash screens are 320 x 460 pixels for iPhone, 768 x 1004 pixels for the iPad portrait mode, and 748 x 1024 pixels for the iPad landscape mode.
8. Click on the checkbox next to **Hide browser nav** to hide the browser navigation when the prototype is launched from the iOS home screen.
9. Click on **iOS Status Bar** to specify the value as **default**, **black**, or **black-translucent**.
10. Click on the **Generate** button to generate your prototype.
11. To view the prototype on your mobile device using AxShare, perform the following steps:
    1. To publish to AxShare, click on the **Publish to Axure Share** button.

   2. A pop-up window will open that provides the publishing status to AxShare and also the link to your prototype. Click on the blue **Copy** link to copy the link to your prototype, and paste the link onto a browser to view your prototype.
   3. To copy the link without sitemap, with the prototype displayed in a browser window, click on the fourth icon that looks like a chain link.
   4. In the drop-down menu, click on the radio button next to the **without sitemap** label.
   5. Copy the link without sitemap.

12. Next, open the link without sitemap on your mobile device to view.
13. To add an icon to the home screen on the Apple device, press the center icon, which is the **Options** button located at the bottom navigation page. In the pop-up menu, select **Add to Home Screen** and click on **Add when prompted**.

> If you specified an iOS splash screen, you should see the iOS splash screen while launching from the home screen. If you chose to hide browser navigation when the prototype loads, you should not see the bottom navigation or browser URL.

How it works...

In the **Mobile/Device** section of the **Generate HTML** dialog box, you set the viewport tag, hid the address bar, prevented vertical page scrolling, specified a home screen icon and iOS splash screens. You also selected **Hide browser nav** and set the **iOS Status Bar**. When you launched the prototype, you copied the link for the prototype without the sitemap and launched the prototype on your iOS device using this link. Finally, you created a shortcut to the page on the iOS home screen.

There's more...

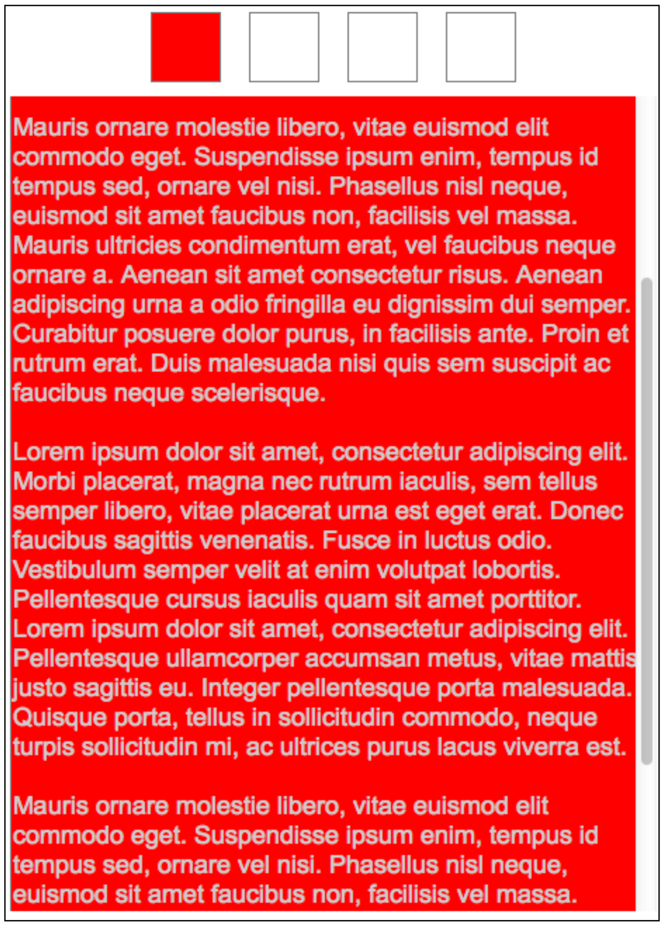
Check out `http://www.axure.com/learn/iphone-app` for more tutorials and articles to help you create interactive iPhone app prototypes using Axure.

Leveraging dynamic panel scrolling for mobile devices

In this recipe, you will learn how to enable dynamic panel scrolling on Axure prototypes targeted for mobile devices.

Getting ready

For this recipe, you will need Axure, an AxShare account or a web server to load your prototype to and a mobile device. The dynamic pane scrolling will look like the following screenshot:



Mauris ornare molestie libero, vitae euismod elit commodo eget. Suspendisse ipsum enim, tempus id tempus sed, ornare vel nisi. Phasellus nisl neque, euismod sit amet faucibus non, facilisis vel massa. Mauris ultricies condimentum erat, vel faucibus neque ornare a. Aenean sit amet consectetur risus. Aenean adipiscing urna a odio fringilla eu dignissim dui semper. Curabitur posuere dolor purus, in facilisis ante. Proin et rutrum erat. Duis malesuada nisi quis sem suscipit ac faucibus neque scelerisque.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi placerat, magna nec rutrum iaculis, sem tellus semper libero, vitae placerat urna est eget erat. Donec faucibus sagittis venenatis. Fusce in luctus odio. Vestibulum semper velit at enim volutpat lobortis. Pellentesque cursus iaculis quam sit amet porttitor. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque ullamcorper accumsan metus, vitae mattis justo sagittis eu. Integer pellentesque porta malesuada. Quisque porta, tellus in sollicitudin commodo, neque turpis sollicitudin mi, ac ultrices purus lacus viverra est.

Mauris ornare molestie libero, vitae euismod elit commodo eget. Suspendisse ipsum enim, tempus id tempus sed, ornare vel nisi. Phasellus nisl neque, euismod sit amet faucibus non, facilisis vel massa.

How to do it...

Perform the following steps to enable dynamic panel scrolling:

1. Start Axure and click on **Create New RP Document**.
2. In the **Sitemap** pane, delete the pages labeled **Home**, **Page 1**, **Page 2**, and **Page 3** by right-clicking on each page, moving your mouse over **Delete** in the pop-up menu, and clicking on it.

3. Import the page labeled **Home** from the `MobileDemoOnSwipe.rp` Axure RP file, which you created in the *Creating a slideshow with OnSwipeLeft and OnSwipeRight events* recipe. To accomplish this, perform the following step:
    1. Click on **File** and then click on **Import from RP File**. Browse to `MobileDemoOnSwipe.rp` the Axure RP file and click on **Open**. Click on the checkbox next to the page labeled **Home**. Click on **Next** through the prompts and click on **Finish**.
4. In the **Sitemap** pane, double-click on the page icon next to the page labeled **Home**.
5. Click on the **Dynamic Panel** at (0,70). Right-click on the **Dynamic Panel** and hover to **Scrollbars**, and in the flyout menu, click on **Show Vertical Scrollbar As Needed**.
6. With the **Dynamic Panel** widget at (10,70) selected, double-click on **State1**. in the **Widget Manager** pane
7. Click on the **Rectangle** widget at (0,0). In the toolbar, change the height **h:** to `920`.
8. In the **Widget Properties and Style** pane, scroll to the **Font** section. Increase the **Font Sizes** to **16** by clicking on the font size drop-down list and and select **16**.
9. In the **Font** section, just below the font size drop-down list, click on the fourth icon with an **A** to bring up the color chooser drop-down menu. To change the color of the text, enter the hex value `CCCCCC` in the field and press *Enter*, or click anywhere outside the drop-down menu.
10. Right-click on the **Rectangle** widget and then on **Edit Text**. Type several paragraphs of text so that the text overflows the length of the **Rectangle** widget. For example, you could type the following text to be shown on the **Rectangle** widget:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi
placerat, magna nec rutrum iaculis, sem tellus semper libero,
vitae placerat urna est eget erat. Donec faucibus sagittis
venenatis. Fusce in luctus odio. Vestibulum semper velit at
enim volutpat lobortis. Pellentesque cursus iaculis quam
sit amet porttitor. Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Pellentesque ullamcorper accumsan metus, vitae
mattis justo sagittis eu. Integer pellentesque porta malesuada.
Quisque porta, tellus in sollicitudin commodo, neque turpis
sollicitudin mi, ac ultrices purus lacus viverra est.

Mauris ornare molestie libero, vitae euismod elit commodo eget.
Suspendisse ipsum enim, tempus id tempus sed, ornare vel nisi.
Phasellus nisl neque, euismod sit amet faucibus non, facilisis
vel massa. Mauris ultricies condimentum erat, vel faucibus
neque ornare a. Aenean sit amet consectetur risus. Aenean
adipiscing urna a odio fringilla eu dignissim dui semper.
Curabitur posuere dolor purus, in facilisis ante. Proin et
rutrum erat. Duis malesuada nisi quis sem suscipit ac faucibus
neque scelerisque.
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi
placerat, magna nec rutrum iaculis, sem tellus semper libero,
vitae placerat urna est eget erat. Donec faucibus sagittis
venenatis. Fusce in luctus odio. Vestibulum semper velit at
enim volutpat lobortis. Pellentesque cursus iaculis quam
sit amet porttitor. Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Pellentesque ullamcorper accumsan metus, vitae
mattis justo sagittis eu. Integer pellentesque porta malesuada.
Quisque porta, tellus in sollicitudin commodo, neque turpis
sollicitudin mi, ac ultrices purus lacus viverra est.

Mauris ornare molestie libero, vitae euismod elit commodo eget.
Suspendisse ipsum enim, tempus id tempus sed, ornare vel nisi.
Phasellus nisl neque, euismod sit amet faucibus non, facilisis
vel massa. Mauris ultricies condimentum erat, vel faucibus
neque ornare a. Aenean sit amet consectetur risus. Aenean
adipiscing urna a odio fringilla eu dignissim dui semper.
Curabitur posuere dolor purus, in facilisis ante. Proin et
rutrum erat. Duis malesuada nisi quis sem suscipit ac faucibus
neque scelerisque.
```

11. You will now add the text to states 2 to 4 for the **Dynamic Panel** widget at (10,70). With the **Dynamic Panel** selected, perform the following step:
    1. Repeat step 6 to add text to states 2 to 4 to change the step by using the following table as a reference for panel state and text color:

Panel state	Text color value
State2	FFFF00
State3	00FF66
State4	FF99CC

12. Save the file as `MobileScrollingDemo.rp`.
13. From the main menu, click on **Publish** and then on **Generate HTML Files...**.
14. In the **Generate HTML** dialog box, with the **General** menu item selected, note the directory where the prototype will be saved under **Destination folder for HTML files**.
15. In the **Generate HTML** dialog box, click on **Mobile/Device**.
16. Click the checkbox next to the **Include Viewport Tag** label and set the **Width** field to **device-width**, **Initial Scale** to **1.0**, **Maximum Scale** to **1.0**, and **User Scalable** to **no**.
17. Click the **Generate** button to generate your prototype.

[The prototype should automatically open in a browser window. If it does not open, browse to your prototype directory (for example, destination folder from step 8) and click on **start.html**.]

18. To view the prototype on your mobile device using AxShare, perform the following steps:
    1. To publish to AxShare, click on the **Publish to Axure Share** button.
    2. A pop-up window will open providing publishing status to Axure Share and also the link to your prototype. Click the blue **Copy** link to copy the link to your prototype, and paste the link into a browser to view your prototype.
    3. To copy the link without sitemap, with the prototype displayed in a browser window, click on the fourth icon that looks like a chain link.
    4. In the drop-down menu, click on the radio button next to the **without sitemap** label.
    5. Copy the link without sitemap.
19. Next, open the link without sitemap on your mobile device to view it.

How it works...

In this recipe, you enabled dynamic panel scrolling for mobile devices. This was accomplished by expanding the height of the content for each state of the background in the dynamic panel. When the prototype was generated, this allowed the content displayed in the dynamic panel to be scrolled vertically on the mobile device

Running your prototype natively on an iOS device

Running a prototype natively on an iOS device is a little more challenging than on other devices due to restricted access to the local filesystem. Luckily, there are several free and paid applications that you can download from iTunes to make the process easier. For this recipe, we will leverage **Sites-2-Go Lite**.

Getting ready

To step through this recipe, you will need to have a previously created Axure RP prototype and Sites-2-Go Lite from iTunes must be installed on your iOS device.

How to do it...

Perform the following steps to leverage Sites-2-Go Lite:

1. Browse to the root folder of the Axure RP prototype you would like to copy to the iOS device.

2. Create a standard ZIP archive of all files and folders in the root prototype folder.
3. Start Sites-2-Go Lite on your iOS device.
4. Tap on the **Stored Sites** button at the top-right corner of the application.
5. Next, tap on the **Upload** button in the tab bar at the bottom-right corner of the dialog box.
6. You have two options for uploading the ZIP archive of your prototype. You can upload the ZIP archive from your computer's web browser or from iTunes.
7. To upload from your computer's web browser, both your computer and your iOS device must be on the same wireless network.
8. On your iOS device, make sure that the **With Browser** button is active in the **Upload** dialog box, and point a web browser on your computer to the IP address, as shown in the **Upload** dialog box.
9. Tap on the **Stop Accepting Files** button on the iOS device when the files have been uploaded.
10. If you wish to upload files via iTunes, select the Sites-2-Go Lite app in iTunes in the **File Sharing Apps** list.
11. Click on **Add** and then select the ZIP archive of your prototype. Once the ZIP archive has completely loaded to the iOS device, you will no longer see the file in iTunes.
12. To view the uploaded prototype, tap on the **Stored Sites** button at the top-right corner of the application.
13. Tap on the **Titles** button in the tab bar at the bottom of the dialog box. You should see your uploaded prototype listed.
14. Tap on the title of your prototype to launch it.
15. Next, shake the iOS device to get the application to run in fullscreen mode.

How it works...

In this recipe, you used an application called Sites-2-Go Lite to upload your Axure prototype saved in a ZIP archive to your iOS device. You then viewed the prototype using the application.

There's more...

Applications available on iTunes, such as Sites-2-Go Lite, enable you to upload your Axure prototype directly to iOS devices. This recipe leveraged Sites-2-Go Lite but there are other applications you could use as well. To name a few, additional iOS apps are GoodReader, SiteSucker, and iSaveWeb.

Running your prototype natively on Android devices

You can run any Axure prototype from a local file, USB storage, or network drive on Android devices. In this recipe, you will learn how to run your prototype natively on an Android device.

Getting ready

To step through this recipe, you will need to have a previously created Axure RP prototype and an Android device.

> Since Android is an open source platform, it is much easier to access local files on an Android device. In comparison, Apple's iOS platform typically restricts local file access unless provided through an application.

How to do it...

Perform the following steps to run you prototype natively on an Android device:

1. Copy the prototype directory, including all files and folders, to your Android device.
2. Open the browser on your Android device.
3. In the browser's URL field, enter `file:///` and the path to your HTML file. This will open the local HTML file and launch the prototype. For example, to open the `home.html` file stored on your SD card in a directory called `temp`, type `file:///sdcard/temp/home.html`.

How it works...

Once you have stored your prototype to the internal memory or the SD card on your Android device, you can easily open your prototype files. Just enter `file:///` in your browser's URL field and the path to the HTML file you wish to view.

There's more...

For most Android devices, you can connect them to your USB port on a Windows computer, and your computer will detect the Android device, allowing you to transfer files to your Android device.

For Mac OS X, you will need to use a third-party app to enable you to transfer files. One such app is **Android File Transfer**. You can learn more about this at `http://www.android.com/filetransfer/`.

Running your prototype natively on Microsoft Surface and Windows 8 devices

You can run any Axure prototype from a local file, USB storage, or network drive on the Microsoft Surface or Windows 8 device. In this recipe, you will learn how to run your prototype natively on a Microsoft Surface device.

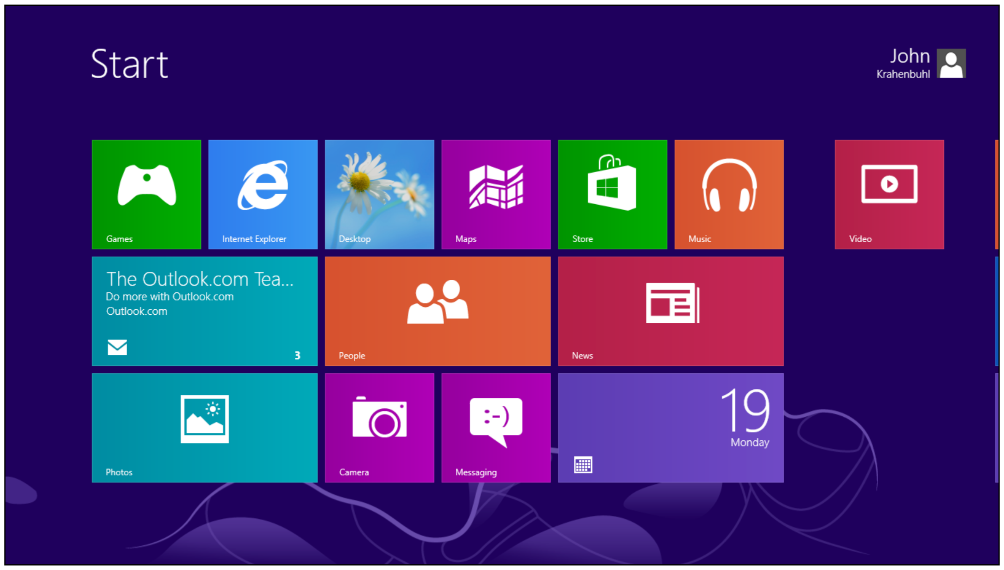
Getting ready

To step through this recipe, you will need to have a previously created Axure RP prototype and a Windows 8 device.

How to do it...

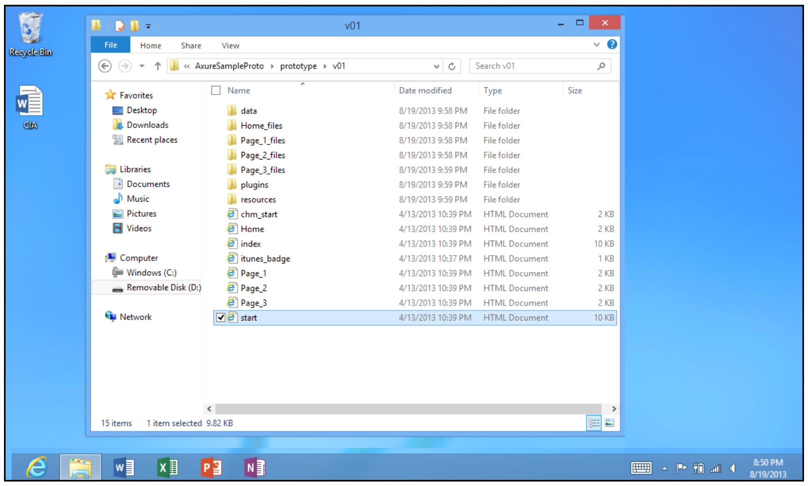
Perform the following steps to run your prototype natively on your Windows device:

1. First, make sure that all Axure RP prototype files can be seen from the file explorer on your Surface.



2. Tap on the desktop tile to open the traditional desktop view.
3. Tap on the file explorer icon at the bottom ribbon to open the file explorer. Browse to your prototype directory to verify that you can see the `start.html` file.

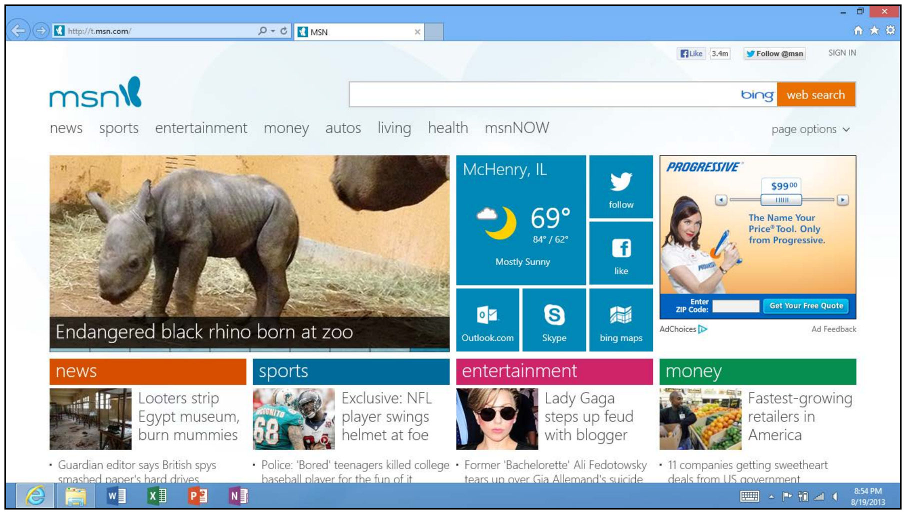
> You must double-tap to open directories and files.



Name	Date modified	Type	Size
data	8/19/2013 9:58 PM	File folder	
Home_files	8/19/2013 9:58 PM	File folder	
Page_1_files	8/19/2013 9:58 PM	File folder	
Page_2_files	8/19/2013 9:58 PM	File folder	
Page_3_files	8/19/2013 9:59 PM	File folder	
plugins	8/19/2013 9:59 PM	File folder	
resources	8/19/2013 9:59 PM	File folder	
chm_start	4/13/2013 10:39 PM	HTML Document	2 KB
Home	4/13/2013 10:39 PM	HTML Document	2 KB
index	4/13/2013 10:39 PM	HTML Document	10 KB
itunes_badge	4/13/2013 10:37 PM	HTML Document	1 KB
Page_1	4/13/2013 10:39 PM	HTML Document	2 KB
Page_2	4/13/2013 10:39 PM	HTML Document	2 KB
Page_3	4/13/2013 10:39 PM	HTML Document	2 KB
✓ start	4/13/2013 10:39 PM	HTML Document	10 KB

15 items 1 item selected 9.82 KB

4. To allow IE to run JavaScript, you must first verify a security setting for IE. With the desktop mode visible, tap on the Internet Explorer icon.
5. Tap on the gear icon in the IE toolbar in the upper-right corner.



Endangered black rhino born at zoo

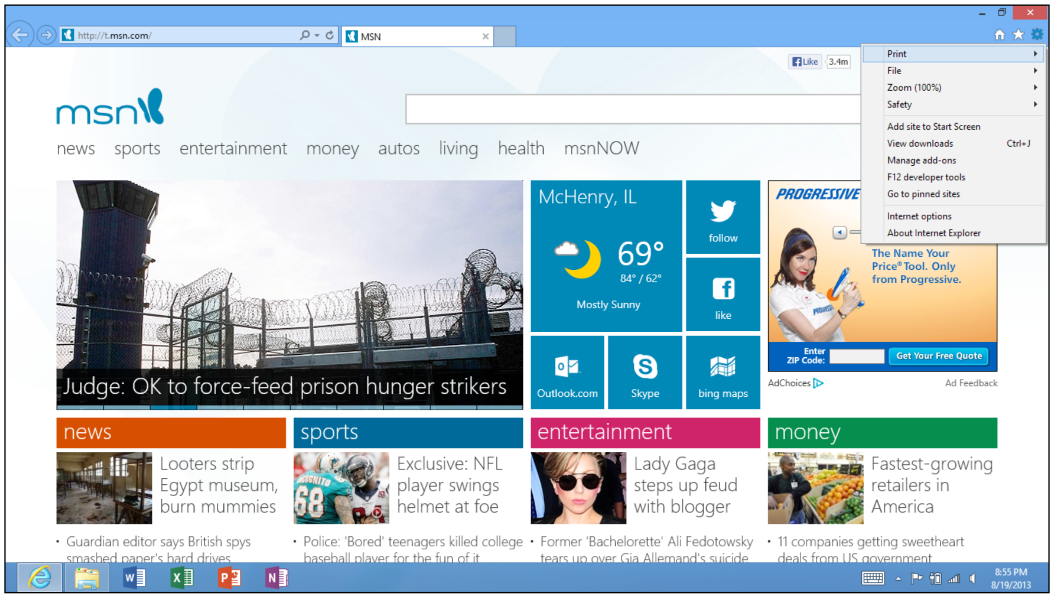
news: Looters strip Egypt museum, burn mummies
• Guardian editor says British spys smashed paper's hard drives

sports: Exclusive: NFL player swings helmet at foe
• Police: 'Bored' teenagers killed college baseball player for the fun of it

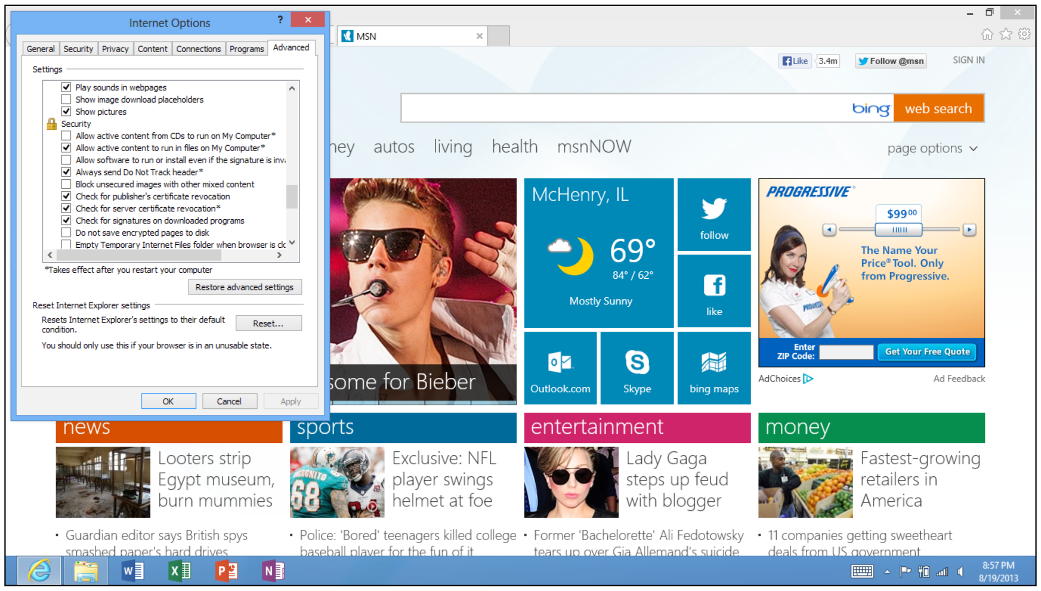
entertainment: Lady Gaga steps up feud with blogger
• Former 'Bachelorette' Ali Fedotowsky tears up over Gia Allemand's suicide

money: Fastest-growing retailers in America
• 11 companies getting sweetheart deals from US government

6. In the flyout menu, tap on the **Internet options**.



7. In the **Internet Options** pop up, tap on the **Advanced** tab. Scroll to the **Security** section and tap on the checkbox next to the option **Allow active content to run in files on My Computer*** option. Tap on the **Apply** button.


Internet Options

General | Security | Privacy | Content | Connections | Programs | Advanced

Settings

- [✓] Play sounds in webpages
- [ ] Show image download placeholders
- [✓] Show pictures
- Security
- [ ] Allow active content from CDs to run on My Computer*
- [✓] Allow active content to run in files on My Computer*
- [ ] Allow software to run or install even if the signature is inv
- [✓] Always send Do Not Track header*
- [ ] Block unsecured images with other mixed content
- [✓] Check for publisher's certificate revocation
- [✓] Check for server certificate revocation*
- [✓] Check for signatures on downloaded programs
- [ ] Do not save encrypted pages to disk
- [ ] Empty Temporary Internet Files folder when browser is clo

*Takes effect after you restart your computer

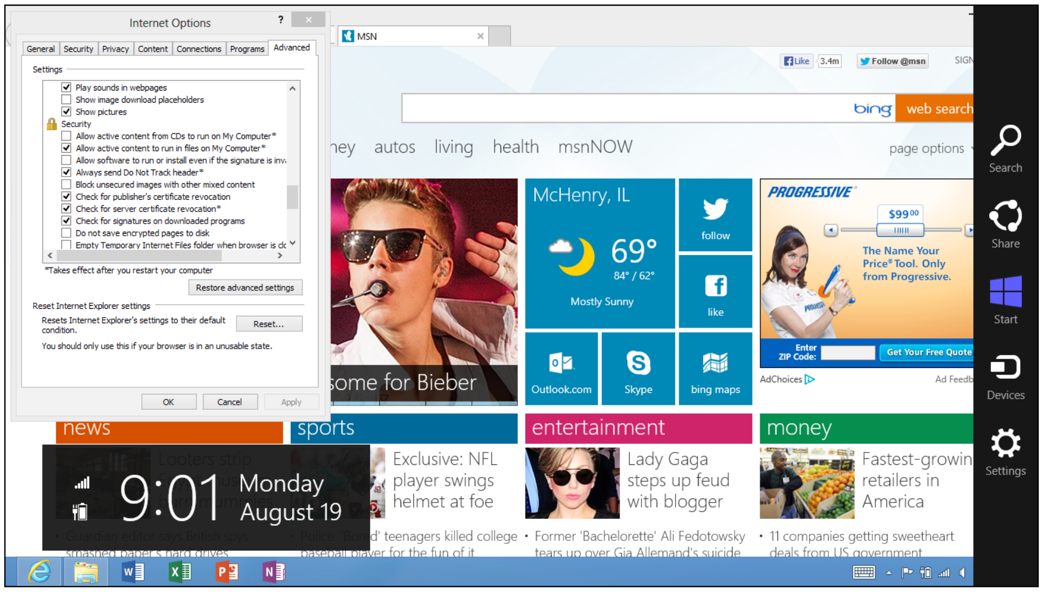
Restore advanced settings

Reset Internet Explorer settings

Resets Internet Explorer's settings to their default condition.

You should only use this if your browser is in an unusable state.

8. Tap on the **OK** button.
9. For these changes to take effect, you must restart your Windows 8 device.
10. The **Charms bar** is a vertical menu that contains five charms: **Search**, **Share**, **Start**, **Devices**, and **Settings**. Display the Charms bar on the right edge of the display by swiping from the right of the screen.



Internet Options

General | Security | Privacy | Content | Connections | Programs | Advanced

Settings

- [✓] Play sounds in webpages
- [ ] Show image download placeholders
- [✓] Show pictures

Security

- [ ] Allow active content from CDs to run on My Computer*
- [✓] Allow active content to run in files on My Computer*
- [ ] Allow software to run or install even if the signature is inv
- [✓] Always send Do Not Track header*
- [ ] Block unsecured images with other mixed content
- [✓] Check for publisher's certificate revocation
- [✓] Check for server certificate revocation*
- [✓] Check for signatures on downloaded programs
- [ ] Do not save encrypted pages to disk
- [ ] Empty Temporary Internet Files folder when browser is cl

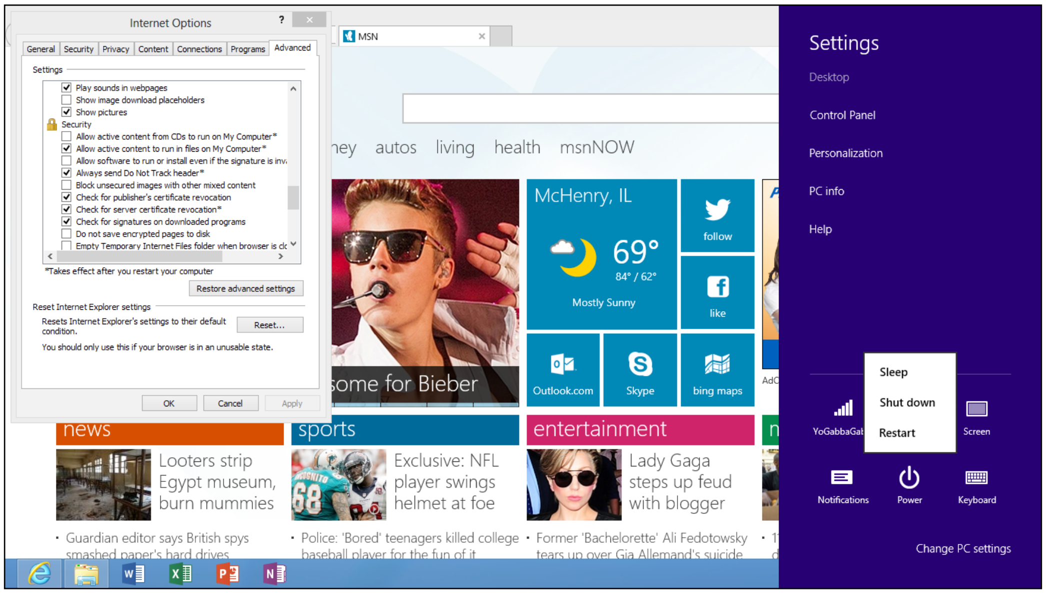
*Takes effect after you restart your computer

Reset Internet Explorer settings

Resets Internet Explorer's settings to their default condition.

You should only use this if your browser is in an unusable state.

11. Tap on the **Settings** icon.
12. Tap on the **Power** icon and then tap on **Restart** to restart your Windows 8 device.



Internet Options

General | Security | Privacy | Content | Connections | Programs | Advanced

Settings

- [✓] Play sounds in webpages
- [ ] Show image download placeholders
- [✓] Show pictures

Security

- [ ] Allow active content from CDs to run on My Computer*
- [✓] Allow active content to run in files on My Computer*
- [ ] Allow software to run or install even if the signature is inv
- [✓] Always send Do Not Track header*
- [ ] Block unsecured images with other mixed content
- [✓] Check for publisher's certificate revocation
- [✓] Check for server certificate revocation*
- [✓] Check for signatures on downloaded programs
- [ ] Do not save encrypted pages to disk
- [ ] Empty Temporary Internet Files folder when browser is cl

*Takes effect after you restart your computer

Reset Internet Explorer settings

Resets Internet Explorer's settings to their default condition.

You should only use this if your browser is in an unusable state.

Restore advanced settings
Reset...
OK
Cancel
Apply
MSN
ney
autos
living
health
msnNOW
some for Bieber
McHenry, IL
69°
84° / 62°
Mostly Sunny
follow
like
Outlook.com
Skype
bing maps
news
Looters strip Egypt museum, burn mummies
Guardian editor says British spys smashed paper's hard drives
sports
Exclusive: NFL player swings helmet at foe
Police: 'Bored' teenagers killed college baseball player for the fun of it
entertainment
Lady Gaga steps up feud with blogger
Former 'Bachelorette' Ali Fedotowsky tears up over Gia Allemand's suicide
Settings
Desktop
Control Panel
Personalization
PC info
Help
Sleep
Shut down
Restart
YoGabbaGab
Screen
Notifications
Power
Keyboard
Change PC settings

13. Once your Windows 8 device has restarted, your changes to IE will take effect.
14. You can now browse to your prototype directory, and tap on the **start.html** file to launch your prototype in IE.

How it works...

You first verified that you could browse to the Axure prototype directory from your Windows 8 device. You then made sure that Internet Explorer was set to allow active content to run in files on My Computer. For this setting to take effect, you had to reboot your Windows 8 device. Once your Windows 8 device was rebooted, you were able to open the `start.html` file in the prototype directory in Internet Explorer.

9 Miscellaneous Explorations

In this chapter, we will discuss miscellaneous prototyping concepts to enhance your Axure prototypes. You will learn the following:

- Using event triggers to play audio files
- Recovering Axure files
- Setting up the Android SDK for virtual application prototype demonstrations
- Setting up Xcode or iPad/iPhone virtual application prototype demonstrations
- Installing XAMPP for Mac OS X for local prototyping
- Installing XAMPP on Windows 8.1 for local prototyping
- Leveraging Google reCAPTCHA and PHP pages with Axure
- Using Ajax, JavaScript, MySQL, and PHP with Axure

Introduction

By utilizing additional free resources for developers, you can create robust local environments to support your prototyping activities with Axure. You will learn how to install and configure the Android SDK, use Xcode for iPad/iPhone simulation, and install the Acquia desktop, which provides you with a local Apache web server as well as PHP and MySQL.

Using event triggers to play audio files

Ever wanted to have an audio file play when the user interacts with your prototype? This recipe will show you how to have an audio file play when a button is clicked in your prototype.

How to do it...

In this recipe, you will enable the **onClick** interaction on a Button Shape widget to play an audio file as follows:

1. Start Axure and under **Create New** select **RP File**.
2. While holding down the mouse button, drag the **Button Shape** widget and place it on the wireframe at (50,50).
3. Click on the **Button Shape** widget on the wireframe and type `Click for Sound`.
4. With the **Button Shape** widget selected, perform the following steps:
    1. In the **Widget Interactions and Notes** pane, click in the **Shape Name** field and type `PlayAudioButton`.
    2. In the **Widget Interactions and Notes** pane, double-click on the **Interactions** tab and double-click the **OnClick** interaction to open the **Case Editor** pop up.
    3. In the **Case Editor** pop up, under **Click to add actions**, click on **Links** to expand it and click on **Open Link** to expand and click **Current Window**.
    4. You will see, in the **Organize actions** pane, **Case 1** change to have the action **Open Link in Current Window**.
    5. In **Configure actions**, click on the radio button next to **Link to an external URL or file**.
    6. Click in the **Hyperlink** field and enter the name of your sound file (for example, `sineWave.wav`).
    7. Click on **OK**.
5. Click on the **Publish** button on the toolbar and select **Generate Prototype Files...**. You can also generate the prototype by going to the main menu and clicking on **Publish** and then on **Generate HTML Files...**.
6. Note the directory where the prototype is saved.
7. Copy your sound file (that is, `sineWave.wav`) to the same directory as your prototype.
8. In your prototype directory, open `start.html` in a browser to view the prototype.
9. When you click on the **Button Shape** widget, the sound file will play with the default media player associated with your browser.

How it works...

The key to this recipe is using the Button Shape widget with an OnClick interaction of the Open Link in Current Window action. The external URL hyperlinked to is the sound file we copied to our prototype directory. When the **Button Shape** widget is clicked, the default media player is called and the sound file is played in the browser.

Recovering Axure files

From time to time, you may need to recover an Axure RP file from a backup. By default, Axure creates backups every 15 minutes. This recipe will teach you how to recover an Axure RP file from a backup.

How to do it...

You will perform the following steps:

1. Start Axure. On the main menu, click on **File**, and in the drop-down menu, select **Recover File from Backup...**.
2. Axure 7, by default, will show you files from the last five days. You can change this increment to show files for up to 15 days.
3. Select the file you would like to recover from the list and click on the **Recover** button.
4. In the **Save As** dialog, enter the new filename to save the recovered file as and click on the **Save** button.

Save the recovered file with a new filename to avoid accidently overwriting your most recently saved file.

How it works...

In this recipe, you learned to recover an Axure RP file from a backup. By default, Axure creates backups every 15 minutes. You also learned how to change this default backup interval.

Setting up the Android SDK for virtual application prototype demonstrations

If you've ever wanted to view your Axure prototypes on various Android devices, this recipe will walk you through downloading the Android SDK and starting the Android emulator.

Getting ready

For this recipe, you will need to download the latest Android SDK from `http://developer.android.com/sdk/index.html`.

How to do it...

In this recipe, you will install the latest Android SDK, create an Android emulator and view an Axure prototype in a browser on the emulator, by using the following steps:

1. Download and install the latest Android SDK from `http://developer.android.com/sdk/index.html`.
2. Open a terminal or command prompt and change the directory to the location where you unzipped the Android SDK. For example, on a Mac where the SDK is in the `Applications` folder, type this at the command prompt:

   ```
   cd /Applications/adt-bundle-mac-x86_64-20131030/sdk/tools
   ```

3. Type `./android list targets` to list all of the available Android target platforms.
4. To create an Android Virtual Device running Android 4.4 (KitKat), type the following command:

   ```
   ./android avd
   ```

5. The **Android Virtual Device Manager** window should open.
6. Click on the **New** button. In the **Create new Android Virtual Device (AVD)** pop up, enter the name, device, target, and other options for the AVD. For example:
   - **AVD Name**: `Nexus4_4-4_API19_Kit-Kat`
   - **Device**: `Nexus 4(4.7", 768x1280: xhdpi)`
   - **Target**: `Android 4.4 – API Level 19`
7. Click on the **OK** button to create the new AVD.
8. You will now see your AVD appear in the **Android Virtual Device Manager** list. Click on the **Start** button.
9. In the **Launch Options** pop up, click on **Launch** to launch the AVD.
10. You will see a pop up called **Starting Android Emulator** with a progress bar.
11. Once started, you will see an emulator window open with your AVD running.
12. If you have a local web server running (for example, XAMPP with Apache web server), you can now open a browser on the AVD and view any Axure prototypes or PHP or HTML pages on the device.

If your local web server is running on port 80, then in the Android browser instead of typing `localhost:80`, type `10.0.2.2:80` to access your local web server.

How it works...

In this recipe, you learned how to install the latest Android SDK. You also learned how to start the Android Virtual Device Manager and create your own Android Virtual Device. If you had a local web server running, you then were able to view Axure prototypes or other web pages on the device's browser.

There's more...

If you are using a Mac, you may run into the issue of not being able to open applications downloaded in the SDK. For example, the first time you try to open Eclipse, you will receive an error message stating that this app is from an unidentified developer. In such a case, press *Ctrl* and click on the Eclipse icon; then select **Open** from the pop-up menu. You will be prompted to type the administrator password to add an exception for Eclipse. You should now be able to open Eclipse as you would any other application.

See also

- The *Installing XAMPP for Mac OS X for local prototyping* recipe
- The *Installing XAMPP on Windows 8.1 for local prototyping* recipe

Setting up Xcode for iPad/iPhone virtual application prototype demonstrations

Apple's Xcode is an integrated development environment that includes an iOS simulator. In this recipe, you will install and set up Apple's Xcode to enable you to view your Axure prototypes on virtual iOS devices.

How to do it...

In this recipe, you will install the latest version of Xcode, launch an iOS simulator and view an Axure prototype in a browser in the simulator.

1. Download and install the latest version of Xcode from `https://developer.apple.com/xcode/`.

2. Open the `Xcode.dmg` file. Once the Xcode installer starts, drag the Xcode icon onto the applications icon to start the install.
3. After Xcode has successfully installed, launch it by clicking on the Xcode icon in the `Applications` folder.
4. On the main menu, navigate to **Xcode** | **Open Developer Tool**, and then click on **iOS Simulator** to launch the iOS simulator.
5. The default simulator opened for Xcode 5.0.1 is the iPhone Retina (4 inch). To change devices, select **Hardware** and then **Device** on the main menu, and click on the device you would like to simulate.
6. Once the simulator has started, launch Safari.
7. If you have a local web server running (for example, XAMPP with Apache web server), you can now view any Axure prototypes or PHP or HTML pages on the iOS Simulator.

[If your local web server is running on port 80, type `localhost:80` in the Safari browser.]

How it works...

In this recipe, you installed and set up Apple's Xcode. This enabled you to view your Axure prototypes on virtual iOS devices when you had a local web server running.

See also

- The *Installing XAMPP for Mac OS X for local prototyping* recipe
- The *Installing XAMPP on Windows 8.1 for local prototyping* recipe

Installing XAMPP for Mac OS X for local prototyping

XAMPP is a non-profit project offered by the Apache Friends organization. This organization's goal is to promote the use of the Apache web server. Installing XAMPP gives you a local Apache web server with MySQL, PHP, and Perl. In this recipe, you will learn how to install XAMPP for Mac OS X for local prototyping.

Getting ready

For this recipe, you will need to download the latest version of XAMPP for Mac OS X from `http://www.apachefriends.org/en/xampp-macosx.html`.

How to do it...

In this recipe you will install the latest version of XAMPP for Max OS X.

1. Click on the DMG image to open it. Double-click on the `xampp-osx` installer icon.
2. If prompted for the administrator password, enter your administrator password to continue installation.
3. XAMPP will be installed to the default directory `/Applications/XAMPP`. Click on the **Next** button to continue.
4. The XAMPP installer will prompt you to install **BitNami**, a library that makes it easier to install other popular open source applications. Click on the **Next** button twice to continue and start the install.
5. Open your `Applications` folder and look for the `XAMPP` folder. Click to expand the `XAMPP` folder and click on the **manager-osx** shortcut to open the XAMPP control panel.
6. Click on the **Manage Servers** tab to start and stop your Apache, MySQL, and ProFTPD servers.
7. To change the configuration of a server or list the port that a server is listening to, click the type of server in the server list (that is, **MySQL Database**, **Apache Web Server**, and so on) and click on the **Configure** button to the right-hand side.

The default port for the XAMPP Apache web server is 80. The default port for the MySQL database is 3306.

8. To test that XAMPP is working properly, open the address `http://localhost:80/xampp/` in a browser. You should see the default start up page, **Welcome to XAMPP**.

How it works...

In this recipe, you installed XAMPP for Mac OS X to enable advanced prototyping. By having a local Apache web server with MySQL, PHP, and Perl you can now explore prototyping options that previously required a separate server. Through using the XAMPP Manager-OSX shortcut, you learned how to start and stop your Apache web server and your MySQL database server. You also learned how to quickly verify the port configurations and change server settings.

There's more...

Learn more about XAMPP at `http://www.apachefriends.org`. Learn more about BitNami for XAMPP at `http://bitnami.com/stack/xampp?utm_source=bitnami&utm_medium=installer&utm_campaign=XAMPP%2BInstaller`.

Installing XAMPP for Windows 8.1 for local prototyping

XAMPP for Windows will provide you with a local Apache web server with MySQL, PHP, and Perl. In this recipe, you will learn how to install XAMPP for Windows 8.1 for local prototyping.

Getting ready

For this recipe, you will need to download the installer for the latest version of XAMPP for Windows from `http://www.apachefriends.org/en/xampp-windows.html`, as well as the Microsoft Visual C++ 2008 Redistributable package from `http://www.microsoft.com/en-us/download/details.aspx?id=5582`.

How to do it...

In this recipe you will install the Windows version of XAMPP.

1. Download and install the Microsoft Visual C++ 2008 Redistribution package and XAMPP for Windows.

   During the installation of XAMPP, you should disable your antivirus software.

2. XAMPP will be installed to the default directory: `C:\xampp`. Click on the **Next** button to continue.
3. The XAMPP installer will prompt you to install BitNami. Click on the **Next** button twice to continue and start the installation.

   BitNami for XAMPP makes it easy to install applications such as WordPress, Drupal, Joomla!, and more.

4. Open a file explorer window and navigate to `C:\xampp`. Click on the `xampp-control` application.

5. The XAMPP Control Panel shows the status of the modules (for example, Apache, MySQL, and so on).
6. To start a service, click on the **Start Action** button associated with the service you would like to start. When the service is running, you will see the module name highlighted in green.
7. To stop a service, click on the **Stop Action** button associated with the service you would like to stop.
8. The acronym PID stands for Windows Process Identifier. Once a module has started, you will see the PID associated with the module under the **PID** column and the port associated with the module under the **Port** column.
9. To change the configuration of a server, click on the **Config** button in the upper-right of the Control Panel. In the **Configuration Panel** pop up, click on the **Service and Port Settings** button. In the **Service Settings** pop up, click on the tab associated with the service (for example, Apache, MySQL, and so on) and make the required changes. Once finished, click on the **Save** button. Click on the **Save** Button on the **Configuration Control Panel**.

> The default port for the XAMPP Apache web server is 80. The default port for the MySQL database is 3306.

10. To test that XAMPP is working properly, visit the address `http://localhost:80/xampp/` in a browser. You should see the default start up page **Welcome to XAMPP**.

How it works...

In this recipe, you installed XAMPP for Windows to enable advanced prototyping. By having a local Apache web server with MySQL and PHP, you can now explore prototyping options that previously required a separate server. Using the XAMPP control application, you learned how to start and stop your Apache web server and your MySQL database server. You also learned how to quickly verify port configurations and change server settings.

Leveraging Google reCAPTCHA and PHP pages with Axure

Google's reCAPTCHA is a free CAPTCHA service that allows you to prevent automated programs (that is, "bots") from spamming websites. When you use reCAPTCHA, you are not only protecting your website but are also helping to digitize newspapers and books.

Getting ready

You need to have a local Apache web server and PHP installed and running in your local environment by using XAMPP or another Apache, MySQL, or PHP stack installed. You will also need to have the reCAPTCHA PHP library downloaded from `https://code.google.com/p/recaptcha/downloads/list?q=label:phplib-Latest`.

How to do it...

In this recipe, you will use an Inline Frame widget to display a reCAPTCHA form.

1. Open a web browser, go to `http://www.google.com/recaptcha/whyrecaptcha`, and click on the **Sign up Now!** button.
2. Sign in with your Google account. If you do not have a Google account, you can create one at `https://accounts.google.com/SignUp`.
3. On the account creation page, enter your domain in the form field (**Domain**) to create your public and private reCAPTCHA keys and click on the **Enable this key on all domains (global key)** option. Click on the **Create Key** button.
4. Save your public and private keys. You will use these in your PHP files to communicate between your server and the reCAPTCHA server.

[For steps 5 to 8 refer to `https://developers.google.com/recaptcha/docs/php`.]

5. Edit the `example-captcha.php` file included with the reCAPTCHA PHP library, changing the value of `$publickey` and `$privatekey` to the values you received in steps 3 and 4.
6. Copy the `recaptchalib.php` and `example-captcha.php` files you downloaded to locations accessible by your local Apache web server (for example, `/Applications/XAMPP/htdocs` or `C:\xampp\htdocs`).
7. Start Axure and under **Create New** select **RP File**.
8. While holding down the mouse button, drag the **Inline Frame** widget and place it at coordinates (0,0) on the wireframe.
9. With the **Inline Frame** widget selected, to the top-right of the wireframe you will see two fields marked **w:** and **h:**. These are for the width and height of the Inline Frame widget. Enter `800` in the width field and `500` in the height field.
10. Click on the **Inline Frame** widget on the wireframe.
11. Mouse over **Edit Inline Frame** and click on **Edit Default Target**.

12. In the **Link Properties** pop up, click on the radio button next to **Link to an external URL or file**.
13. Click on the **Hyperlink** field and enter the filename of your HTML document, `example-captcha.php`, copied in step 8.
14. Click on the **Publish** button on the toolbar and select **Generate Prototype Files...**. You can also generate the prototype by going to the main menu and selecting **Publish** and clicking on **Generate Prototype Files...**.

How it works...

In this recipe, you learned how to use Google's reCAPTCHA with PHP. You first created a public and private key for reCAPTCHA. You then added your `$publickey` and `$privatekey` values to the `example-captcha.php` file that was downloaded with reCAPTCHA library. You then used Axure's Inline Frame widget to display the reCAPTCHA form.

There's more...

You can learn more at `http://www.google.com/recaptcha/learnmore`.

Using Ajax, JavaScript, MySQL, and PHP with Axure

Ajax, JavaScript, MySQL, and PHP are some of the hottest technologies in use today. By using Inline Frame widgets, you can pull into your prototype dynamic elements using Ajax, JavaScript, MySQL, and PHP. In this recipe you will learn how to create a drop-down list by using Ajax to pull data from a SQL database.

Getting ready

You will need to have a local Apache web server, PHP, and MySQL installed and running in your local environment by using XAMPP or another Apache, MySQL, or PHP stack installed.

How to do it...

Perform the following steps:

1. With your Apache web server and MySQL servers running, open a terminal or command prompt.
2. First, we will create a new MySQL database by typing the following command at the command prompt:

```
mysql -h localhost -u root -p
```

3. If prompted, enter your root/administrator password.
4. At the MySQL prompt, type the following:
   ```
   CREATE DATABASE axure_db;
   ```
5. To create a new MySQL user called `axure`, type this line:
   ```
   CREATE USER 'axure'@'localhost' IDENTIFIED BY 'Rocks!';
   ```
6. Next, at the MySQL prompt, type the following:
   ```
   GRANT ALL PRIVILEGES ON axure_db.* to 'axure'@'localhost';
   ```
7. To make the new changes take effect, type this at the MySQL prompt:
   ```
   FLUSH PRIVILEGES;
   ```
8. To select `axure_db` as your current database, type the following at the MySQL prompt:
   ```
   USE axure;
   ```
9. You will now create a table in the `axure_db` database. At the MySQL prompt, type this line:
   ```
   CREATE TABLE Employee(ID INT NOT NULL AUTO_INCREMENT,
   Name CHAR(50) NOT NULL, Age INT, PRIMARY KEY (ID));
   ```
10. To verify that your table was created properly at the MySQL prompt, type this line:
    ```
    DESCRIBE employee;
    ```
11. We will insert data into the `Employee` table by issuing the following commands:
    ```
    INSERT INTO Employee(Name, Age) VALUES ("Joe Dee", 23);
    INSERT INTO Employee(Name, Age) VALUES ("Sarah Yu", 52);
    ```
12. To verify that the data was added to the `Employee` table, type this:
    ```
    SELECT * FROM Employee;
    ```
13. To store your changes in the database, type the following:
    ```
    COMMIT;
    ```
14. Create a `showemployee.html` file containing the following code:
    ```
    <html>
    <head>
    <script>
    function showEmployee(str)
    {
    if (str=="")
      {
      document.getElementById("tempText").innerHTML="";
    ```

```
  return;
  }
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("tempText").innerHTML=xmlhttp.
    responseText;
    }
  }
xmlhttp.open("GET","getemployee.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<form>
<select name="users" onchange="showEmployee(this.value)">
<option value="">Select an employee:</option>
<option value="1">Joe Dee</option>
<option value="2">Sarah Yu</option>
</select>
</form>
<br>
<div id="tempText"><b>Employee info will be shown here.</b></div>

</body>
</html>
```

15. Create a `getemployee.php` file containing the following code:

```
<?php
$q = intval($_GET['q']);

$con = mysqli_connect('localhost:3306','axure',
'Rocks!','axure_db');
```

```
if (!$con)
  {
  die('Could not connect: ' . mysqli_error($con));
  }

mysqli_select_db($con,"axure_db");
$sql="SELECT * FROM Employee WHERE ID = '". $q ."'";
$result = mysqli_query($con,$sql);

echo "<table border='1'>
<tr>
<th>ID</th>
<th>Name</th>
<th>Age</th>
</tr>";

while($row = mysqli_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['ID'] . "</td>";
  echo "<td>" . $row['Name'] . "</td>";
  echo "<td>" . $row['Age'] . "</td>";
  echo "</tr>";
  }
echo "</table>";

mysqli_close($con);
?>
```

16. Open a browser. In the browser's URL field, enter the path to the `showemployee.html` file and verify operation by changing the drop-down options. You should see the employee information update on the page when you change options in the dropdown.
17. Start Axure and under **Create New** select **RP File**.
18. While holding down the mouse button, drag the **Inline Frame** widget and place it at coordinates (0,0) on the wireframe.
19. With the **Inline Frame** widget selected, you will see two fields marked **W:** and **H:** in the top-right corner of the wireframe. These are for the width and height of the Inline Frame widget. Enter `800` in the width field and `500` in the height field.
20. Click on the **Inline Frame** widget on the wireframe.
21. Mouse over **Edit Inline Frame** and click on **Edit Default Target**.
22. In the **Link Properties** pop up, left-click on the radio button next to **Link to an external URL or file**.

23. Click on the **Hyperlink** field and enter the filename of your HTML document, `showemployee.html`.
24. Click on the **Publish** button in the toolbar and select **Generate Prototype Files...**. You can also generate the prototype by going to the main menu and selecting **Publish** and clicking on **Generate Prototype Files...**.

How it works...

You first created a MySQL database and a user and assigned the user permissions to access the database. You used an HTML file to show the drop-down list and Ajax JavaScript to display results from the PHP database query. MySQL and PHP are some of the hottest technologies in use today. You then used Axure's Inline Frame widget to display the HTML form and pull it into an Axure prototype.

Index