Quick answers to common problems

CiviCRM Cookbook

Master this web-based constituent relationship management software for nonprofit and civic sector organizations

Foreword by Dave Greenberg, Co-founder of CiviCRM

Tony Horrocks

[PACKT] PUBLISHING open source community experience distilled

CiviCRM Cookbook

Master this web-based constituent relationship management software for nonprofit and civic sector organizations

Tony Horrocks

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

CiviCRM Cookbook

First published: June 2013

Production Reference: 1310513

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

ISBN 978-1-78216-044-1

www.packtpub.com

Cover Image by Abhishek Pandey (abhishek.pandey1210@gmail.com)

Credits

Author
Tony Horrocks

Reviewers
Erik Hommel
Kurund Jalmi
Andrew Wasson

Acquisition Editor
Usha Iyer

Lead Technical Editor
Dayan Hyames

Technical Editors
Jalasha D'costa
Pushpak Poddar
Varun Pius Rodrigues
Lubna Shaikh

Project Coordinator
Anugya Khurana

Proofreaders
Maria Gould
Paul Hindle

Indexer
Tejal Soni

Production Coordinator
Nitesh Thakur

Cover Work
Nitesh Thakur

Foreword

This year CiviCRM celebrated its eight birthday. Leveraging the open source model of collaboration and transparency, a global network of passionate people have built an enterprise quality CRM solution, which provides a compelling alternative to closed source proprietary products, and supports the mission critical activities of thousands of nonprofit and civic-minded organizations in more than 25 countries and five continents. Nonprofits of all sizes are adopting CiviCRM, from local arts groups (such as San Francisco Center for the Book), to multinational membership associations (International Mountain Biking Association), political parties (British Columbia NDP), advocacy organizations (Electronic Frontier Foundation), national charities (Leukemia & Lymphoma Research), and government entities (New York State Senate).

This "Cookbook" represents another exciting milestone in the evolution of the project. As an enthusiastic chef, I learned long ago that cookbook recipes provide a launching point for creativity. Good cooks take a recipe, test it out, and then modify and improve it based on their personal taste *and* knowing their "audience" (family, friends, and guests). Cookbook recipes are a perfect analog for sharing, leading to innovation.

CiviCRM's strength is based on shared innovation. In the two years since Packt's *Using CiviCRM* was published, we've seen an explosion of invention as users and implementers shape CiviCRM-based solutions to increasingly complex problems. Some of these are one-off customizations, but many have developed into full-fledged projects such as the CiviCRM-Webform integration module highlighted in this book. The power of these tools was brought home to me at a recent CiviCRM meetup, where Lisa Hubbert demonstrated the complex summer camp management interface she had built as a volunteer for San Francisco Arts Ed – a wonderful nonprofit that runs arts programs for inner-city kids. Lisa is not a software engineer, but a curious and passionate "cook". She developed an effective solution for her organization, *and* she taught and inspired others by sharing her work at a meetup and on the `CiviCRM.org` blog.

The introduction this year of "native" CiviCRM extensions, a built-in extension browser for site administrators, and a searchable Extensions Directory (`http://civicrm.org/extensions`) on `CiviCRM.org`, will facilitate even more shared innovation—including sharing major new extension-based functionality such as the forthcoming CiviVolunteer module across all three CMS platforms.

For those of you working with CiviCRM in a Drupal environment, this book includes a wide array of techniques. Take advantage of the integration capabilities and openness of both platforms. For those of you working with CiviCRM in WordPress or Joomla!, my hope is that these recipes will stimulate you to explore, build, and share analogous integrations with those CMSs.

This Cookbook is well-suited to bridge the gap between nontechnical end users and software engineers. Whether you are a volunteer, in-house staff person, or a consultant—I'm confident it will provide you with ideas for using CiviCRM more effectively.

Ultimately, the strength of any open source project is the strength of the community behind it. If CiviCRM helps your organization (or your clients' organizations) with mission critical tasks, I urge you to participate actively in the community. Sponsor new features and improvements via the "make it happen" campaigns (`http://civicrm.org/mih`), post new recipes and modules on the Extensions Directory (`http://civicrm.org/extensions`), use social media to share success stories, introduce your peers at other nonprofits to CiviCRM, join a local meetup (or start one), help others who are getting started, and ensure the long-term sustainability of the project with a recurring contribution at `http://civicrm.org/contribute`!

David Greenberg,
Co-founder of CiviCRM

Looking for more learning resources? Check out:

- *Using CiviCRM* by *Packt Publishing*
- CiviCRM User guild and Developer guide (`http://book.civicrm.org`)
- Extension Developer guide and reference (`http://documentation.civicrm.org`)

And remember, CiviCRM is continually evolving and growing, so make sure you're on top of the latest news, by subscribing to the community newsletter at `http://civicrm.org`.

About the Author

Tony Horrocks is the owner of Fabriko Limited (`http://fabriko.co.uk`), a web development company that specializes in CiviCRM and Drupal. Tony has worked for membership organizations for over 25 years and has been developing websites since 1994.

He now works primarily as a Development Consultant for the nonprofit sector.

Thanks, of course, to the superstars of Packt Publishing for their assistance and encouragement, and the reviewers far and wide who I have never met.

Also, thanks to the CiviCRM core development team and the wider CiviCRM community for their dedication.

Thank you to all those people and organizations who donate to the CiviCRM project (`http://civicrm.org/content/make-it-happen`).

Lastly, thanks to Jackie, without whom none of this would have been possible, and also thanks to Rosie, who now has a book dedicated to her.

About the Reviewers

Erik Hommel has been an active member of the CiviCRM community since 2009. As project manager and developer with EE-atWork (`http://www.ee-atwork.nl`), he has worked on several projects implementing CiviCRM and developing customizations to CiviCRM. You can spot Erik regularly in the CiviCRM community on IRC or on the forum.

Kurund Jalmi is one of the core developers of CiviCRM, and he has been associated with the project since its inception. He has also worked on a CiviCRM book at `flossmanuals.net`. When not coding, he likes to spend his time outdoors exploring nature. He loves traveling and is very passionate about photography. For more information, check out `kurund.com`.

Andrew Wasson is partner and lead developer at Luna Design, a graphic design and web development studio in North Vancouver, British Columbia, Canada.

Keenly interested in electronics and technology from an early age, Andrew built his first computer from scratch while in high school in the early 80s. His journey in computer programming began with machine language assemblers, graduating to variations of Basic, C, and he eventually made the leap to web technologies in the mid-1990s when the Internet burst on to the scene.

Andrew has been developing and producing websites since 1998, and today he specializes in developing online membership management systems using Drupal and CiviCRM. Andrew was the technical reviewer for *Designing Next Generation Web Projects with CSS3* (2013), Packt Publishing.

When he is not sharing the responsibilities of running their business with his wife Fiona, Andrew can be found riding or restoring his vintage ex-racing motorcycles.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

PACKTLIB

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface

A good implementation of CiviCRM can transform your organization. Online management of contacts, members, communications, campaigns, funding, and casework used to be beyond the means of small non-governmental organizations (NGOs). But not anymore.

CiviCRM is loaded with features designed and developed by NGOs that make it a second-to-none management tool.

This book takes you from a CiviCRM installation and guides you through, by example, how to exploit the features that make it so popular.

We cover the post-installation setup and all the core and component parts of CiviCRM. In some cases, the recipes focus on CiviCRM, while in others, we cover using CiviCRM with Drupal.

The recipes in this book are not just meant to provide solutions to specific problems. They are there for you to explore and adapt to your own situation.

You don't need to be—and are not expected to be—a CiviCRM expert or a coding ninja. Far from it. What you do need, however, is the will and enthusiasm to use CiviCRM to take your organization from where it is now to where you want it to be.

What this book covers

Chapter 1, *Setting Up CiviCRM*, covers the important post-installation tasks that will get you going quickly. We look at some of the hard-to-do and hard-to-find settings and explore some of the ways of implementing workflows using Scheduled Reminders and CiviCase.

Chapter 2, *Organizing Data Efficiently*, covers the role of tags and groups. We also explore importing and exporting data, and some techniques to make these processes trouble-free.

Chapter 3, *Using the Power of Profiles*, covers how you can exploit the power of CiviCRM profiles to improve usability, speed up data entry, and control listings and directories.

Chapter 4, *Controlling Permissions*, demystifies permissions and shows you how you can use them in a variety of contexts to control access to viewing and editing data.

Chapter 5, *Managing Communications*, covers how to get the best out of CiviMail. We explore the mail templating system and a Drupal-based alternative to authoring your mailings. We also cover techniques for managing mailing subscriptions and allowing users to update information easily.

Chapter 6, *Searching and Reporting*, focuses on the search capabilities of CiviCRM. We look at how you can use searching to find and group data easily. We also explore how you can customize the search result display, and finally we will look at a search technique in Drupal that is not possible within CiviCRM itself.

Chapter 7, *Integrating CiviCRM with Drupal*, covers integrating CiviCRM with Drupal Views and using the power of the Drupal Webform CiviCRM module to do things CiviCRM can't. Finally, we explore some experimental modules that enable you to create user accounts on the fly and to organize contacts using Drupal taxonomy terms.

Chapter 8, *Managing Events Effectively*, uses CiviEvents to explore how you can use jQuery to alter the display and behavior of CiviCRM forms. We also look at how you can use Webform CiviCRM to control registration workflow for paid-for events.

Chapter 9, *Using Campaigns, Surveys, and Petitions Effectively*, covers in detail how to set up campaigns, surveys, and petitions. We also look at how you can use Drupal Views to create a Campaign Dashboard so you can get at-a-glance information about the progress of your campaigns.

Chapter 10, *Working with CiviMember*, explores **CiviMember**, a CiviCRM component used for membership management. We look at a popular requirement—displaying a membership directory—and then explore how to link common membership tasks with other CiviCRM components.

Chapter 11, *Developing with CiviCRM*, looks at the software, skills, and resources you need to start developing CiviCRM in earnest. We also cover developing a simple Drupal module and exploring the CiviCRM API.

What you need for this book

You will need an installed version of CiviCRM. For several recipes, the CMS of choice should be Drupal. There are no specific recipes for Joomla! or WordPress.

For some of the recipes you should have a good text editor.

Who this book is for

This book is for the nontechnical CiviCRM user. You will know how to get CiviCRM installed, but will now want to find out the tips, tricks, and techniques to get the best out of CiviCRM for your particular situation. You should understand the basic operation of CiviCRM and Drupal. For some recipes, it helps if you are familiar with a coding environment as we will be doing some PHP scripting, but you do not need any programming or technical skills as you will learn everything you need in this book.

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 the previous recipe, we used a URL to access the profile we created, that is, `civicrm/profile/create?gid=N&reset=1`, where `N` was the ID of our profile."

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: "Navigate to **Administer** | **Customize Data and Screens** | **Profiles** and add a new profile that will contain the fields you wish to display on your directory."

> 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 on `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.

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 of 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

Setting Up CiviCRM

In this chapter we will cover:

- Setting up a CiviCRM theme in Drupal
- Setting up cron using cPanel
- Adding items to the CiviCRM navigation menu
- Refreshing the dashboard
- Changing display preferences
- Replacing words
- Setting up geocoding
- Autofiling e-mails
- Creating new activities
- Adding custom fields
- Using Scheduled Reminders for activities
- Using CiviCase to create an HR system
- Installing languages and localizing CiviCRM

Introduction

This chapter provides recipes to help you set up your CiviCRM installation. You will find that most of them work in Drupal, Joomla!, and WordPress. Some recipes are **Content Management System** (**CMS**) specific and we have chosen Drupal to illustrate these.

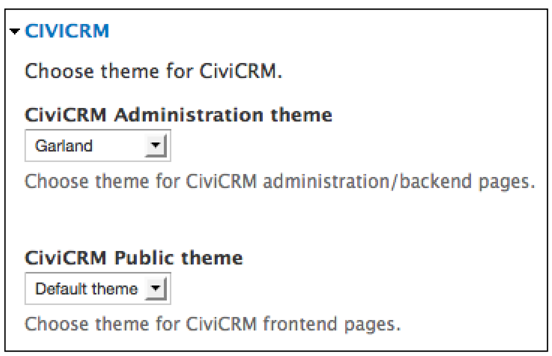
Setting up a CiviCRM theme in Drupal

CiviCRM administration screens take up a lot of browser real estate. How CiviCRM looks is determined by what themes you are using in your CMS. Problems arise when you use your main website theme to display CiviCRM pages. All the customizations, blocks of information, and layouts suddenly get in the way when you want to administer CiviCRM. The trick is to use a different theme for CiviCRM.

How to do it...

This is very easy to accomplish, and just uses a configuration screen in Drupal.

1. Make sure you have the CiviCRM theme module enabled.
2. Navigate to `admin/appearance` in Drupal by clicking on the **Appearance** button. This page shows the themes that are currently installed within our CMS—in this case, Drupal.
3. Make sure that any themes you wish to use are enabled.
4. At the foot of the screen, configure **CiviCRM Administration theme**.



How it works...

Drupal uses the page URL to check if you are administering CiviCRM. If you are, the pages are displayed using the CiviCRM administration theme.

It's a good idea to select a flexible-width theme with sidebars. Garland is a good example. The flexible width accommodates CiviCRM displays nicely.

Once the administration theme is selected, navigate to `admin/structure/blocks`. Here you will see various blocks provided by the CiviCRM module. You can now place these blocks within your administrative theme.

Pay special attention to the visibility settings for these blocks, so that they only appear when using CiviCRM.

There's more...

In Drupal, there is an additional setting that controls which theme is used to display public CiviCRM pages, for example, event sign-up pages.

See also

- You can explore hundreds of contributed Drupal themes at `http://drupal.org/project/themes`

Setting up cron using cPanel

Cron is a time-based scheduler that is used extensively throughout CiviCRM. For example, you might want to use CiviCRM to send out an e-mail newsletter at a particular time, or you might want to send out a reminder to participants to attend an event. CiviCRM has settings to accomplish all these tasks, but these, in turn, rely on having "master" cron set up. Cron is set up on your web server, not within CiviCRM.

How to do it...

There are many different ways of setting up cron, depending on your site-hosting setup. In this example, we are using cPanel, a popular control panel that simplifies website administration.

1. Make a note of your CMS site administrator username and password.
2. Make a note of your CiviCRM site key, which is a long string of characters used to uniquely identify your CiviCRM installation. It is automatically generated when CiviCRM is installed, and is stored in the `civicrm_settings.php` file. Using a text editor, open up the CiviCRM settings file located at `/sites/default/civicrm_settings.php`. Around line 170, you will see the following entry:

   ```
   define( 'CIVICRM_SITE_KEY',
     '7409e83819379dc5646783f34f9753d9' );
   ```

   Make a note of this key.

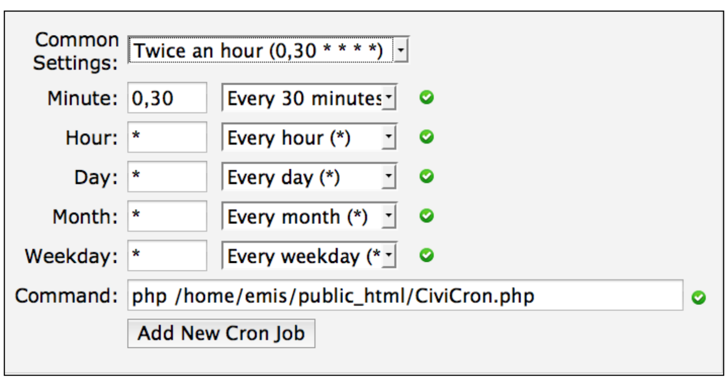
3. Log in to cPanel and use the cPanel File Manager to explore the folders and files that are stored there. You are going to create a file that contains all the necessary information for cron to work. You can choose to create the cron file anywhere you like. It makes sense to keep it in the home directory of your webserver—that is, the first directory you get to once you start exploring.

4. Create a file called `CiviCron.php`. The naming does not particularly matter, but it must be a PHP file.
5. Insert the following code:

```
<?php
// create a new cURL resource
$ch = curl_init();
// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL,
  "http://myDrupalsite.com/sites/all/modules/civicrm/bin/
  cron.php?name=admin&pass=adminpassword&key
  =01504c43af550a317f3c6495c2442ab7");
curl_setopt($ch, CURLOPT_HEADER, 0);
// grab URL and pass it to the browser
curl_exec($ch);
curl_close($ch);
?>
```

   - Substitute `http://myDrupalsite.com` with your own domain
   - Substitute `admin` with your own CMS admin username
   - Substitute `adminpassword` with your own CMS admin password
   - Substitute the key value with the site key from `civicrm_settings.php`

6. Save this file and then navigate to `cron` in cPanel.



7. Select an appropriate cron interval from the **Common Settings** list. Choosing an appropriate cron interval may take some experimentation, depending on how your site is set up. In the **Command** field, enter the following address:

   php /home/site_account_name/public_html/CiviCron.php

   The portion after `php` is the absolute path to the `CiviCron.php` file you created in step 4.

8. Click on **Add New Cron Job**.

How it works...

All cron does is execute the URL that is constructed in the cron file.

The following piece of code does the work:

```
curl_setopt($ch, CURLOPT_URL,
  "http://myDrupalsite.com/sites/all/modules/civicrm/bin/
  cron.php?name=admin&pass=adminpassword&key=
  01504c43af550a317f3c6495c2442ab7");
```

The URL contains the information on permissions (the username, the password, and the site key) to execute the `cron.php` file provided by the CiviCRM module.

Getting cron to work is critical to getting CiviCRM working properly. If you get into difficulties with it, the best solution is to contact your hosting company and seek guidance.

> To test that your cron job is actually working, carry out the following instructions. In the cPanel cron screen, set it to send you an e-mail each time the cron command is run. The e-mail will contain an error message if the cron fails. Failures are generally due to an incorrect setting of the path, or a permissions problem with the username, password, or site key.

Adding items to the CiviCRM navigation menu

As you begin to use CiviCRM, you will want to provide administrative shortcuts. You can do this by adding custom menu blocks within your CMS or editing the navigation menu in CiviCRM.

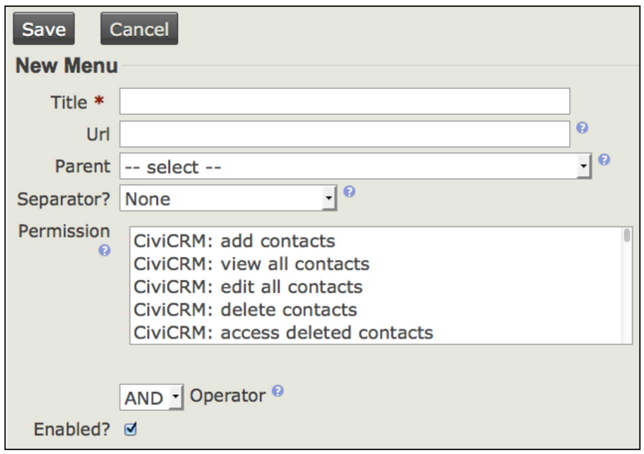
How to do it...

CiviCRM has a fully customizable navigation menu. You can edit this menu to get one-click access to the features you use most.

1. Navigate to a page that you want to use as the link destination for a menu item. For example, you could navigate to **Contacts** | **Manage Groups**, and then select a suitable group.
2. Copy the page URL in the browser location. In this example, it would be as follows:

   `civicrm/group/search?reset=1&force=1&context=smog&gid=2`

3. Navigate to **Administer** | **Customize Data and Screens** | **Navigation Menu**. This displays the CiviCRM navigation menu in tree form.
4. Click on the left arrow on each **Parent** menu item to expand it. You can now explore all the child menu items.
5. Click on the **Add Menu item** button at the top of this screen. This brings up the **Add Menu Item** edit screen.



6. Enter the name of the menu item in the **Title** field.
7. Enter the URL (that you copied) into the **URL** field.
8. Select a parent to make the menu item appear as the child of another menu item. If you don't select a parent, the item will appear on the main CiviCRM menu bar.
9. Select one or more permissions in the **Permission** field to control who can use the menu item. These are CMS permissions, so we must ensure that these are set correctly in our CMS for the menu item to behave properly.

How it works...

CiviCRM stores new menu items, and displays them according to where they are placed in the menu tree and what permissions a user may have to use them.

See also

- You can fully explore CiviCRM customization at `http://book.civicrm.org/user/current/initial-set-up/customizing-the-user-interface/`

Refreshing the dashboard

By default, CiviCRM sets the auto-refresh period for the home page dashboard to `1 hour`. In a busy setting, this is too long, and you constantly have to click on the **Refresh Dashboard data** button to get the information on the dashboard up to date.

How to do it...

Changing the setting is simply a matter of visiting the CiviCRM administration pages:

1. Navigate to **Administer** | **System Settings** | **Undelete, Logging and ReCAPTCHA**.
2. Change the **Dashboard cache timeout** value from **1440** (that's 1 hour in seconds) to a smaller figure.

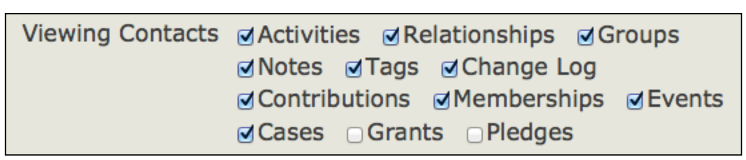
Changing display preferences

By default, CiviCRM displays a lot of data on the **contact summary** screen. Sometimes, this can lead to a cluttered display that is hard to use and slow to load.

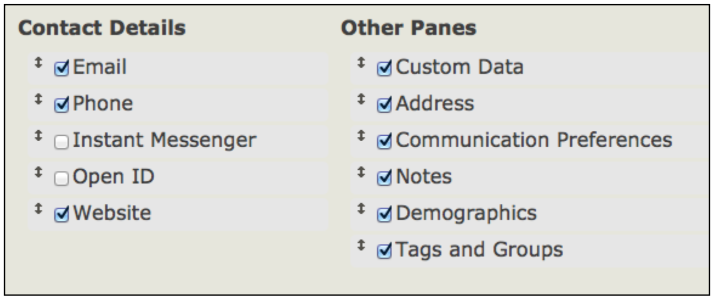
How to do it...

CiviCRM components can add to the clutter on the screen. Here we can disable unwanted components and then fine-tune the display of other elements in the contact summary screen.

1. Navigate to **Administer** | **System Settings** | **Enable CiviCRM Components**, and disable any unused CiviCRM components.
2. Navigate to **Administer** | **Customize data and screens** | **Display preferences**.
3. Control which tabs are displayed in the detail screen (for each contact), using the checkboxes.



4. Control which sections you want to see when editing an individual contact, by checking the checkboxes in the **Editing Contacts** section.



5. Drag the double-arrow icon to move the sections up and down the contact editing screen.

See also

- You can fully explore the display preferences at `http://book.civicrm.org/user/current/initial-set-up/customizing-the-user-interface/`

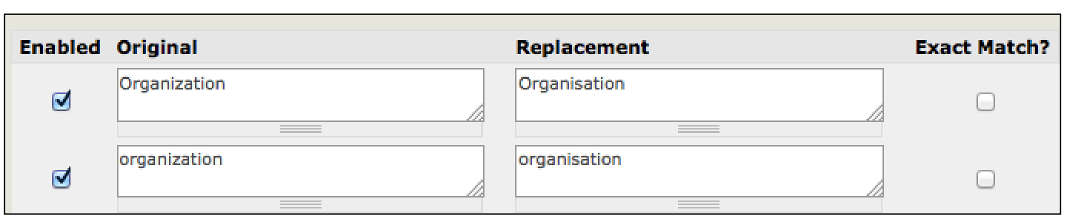
Replacing words

This is useful for fine-tuning your website. For example, you could replace US spelling with UK spelling (thus avoiding installing the UK language translation). Or you might want to change the wording on parts of a standard form without having to make a custom template.

How to do it...

The words—or sentences—that we want to replace are called **strings**. In CiviCRM, we can enter the strings we don't want, and replace them with strings we do want.

1. Navigate to **Administer** | **System Settings** | **Customize Data and Screens** | **Word Replacement**.



In this example, I am replacing the US spelling of "Organization" with the UK version, "Organisation".

2. Use the **Exact Match** checkbox to match words precisely. This would then exclude plurals of the word from being matched. All word replacements are case sensitive.

Setting up geocoding

Geocoding allows you to do location-based searching and to display the maps of contacts.

How to do it...

You need to set a mapping provider—that is a service that will provide you with the visual maps—and a geocoding provider, which will translate your contact addresses into latitude and longitude coordinates.

1. Navigate to **Administer** | **Localization** | **Address settings**. In **Address Display**, make sure that the **Street Address Parsing** checkbox is ticked.
2. Navigate to **Administer** | **System Settings** | **Mapping and Geocoding**. Set **Mapping Provider** to **Google** or **Openstreetmap**. Set **Geocoding Provider** to **Google**.
3. Navigate to **Administer** | **System Settings** | **Scheduled Jobs**. The **Geocode and Parse Addresses** scheduled job should now be enabled. You can set how regularly you want CiviCRM to geocode your address data.

How it works...

Geocoding Provider finds latitude and longitude coordinates for each contact address. **Mapping Provider** uses this information to draw a local map, with a pointer for the contact. **Geocode** and **Parse Addresses** do the geocoding work each day, though you can change this in the settings.

There's more...

Google currently limits geocoding requests to 2,500 per 24 hours. So, if you exceed this limit, Google may not process requests; it may even restrict access to their geocoding service should you continue to break this limit. This is a problem when you have thousands of addresses to process—for example, after a big import of address data.

CiviCRM does not have a tool to place a daily limit on the number of contacts that are processed each day. But you can put parameters into the Geocode and Parse Addresses scheduled job that provide a range of contact IDs to process. You would have to change this each day to work your way though all your contacts.

1. Navigate to **Administer** | **System Settings** | **Scheduled Jobs**, and edit the Geocode and Parse Addresses scheduled job.
2. In the **Command Parameters** box, enter:

   ```
   start= 1
   end=2500
   ```

   `1` would be the ID of your first contact. If you have access to your database tables, check the database table `civicrm_contact` to know what the first value for your contacts is.

See also

- Further details about geocoding in CiviCRM are available at `http://wiki.civicrm.org/confluence/display/CRMDOC43/Mapping+and+Geocoding`

Autofiling e-mails

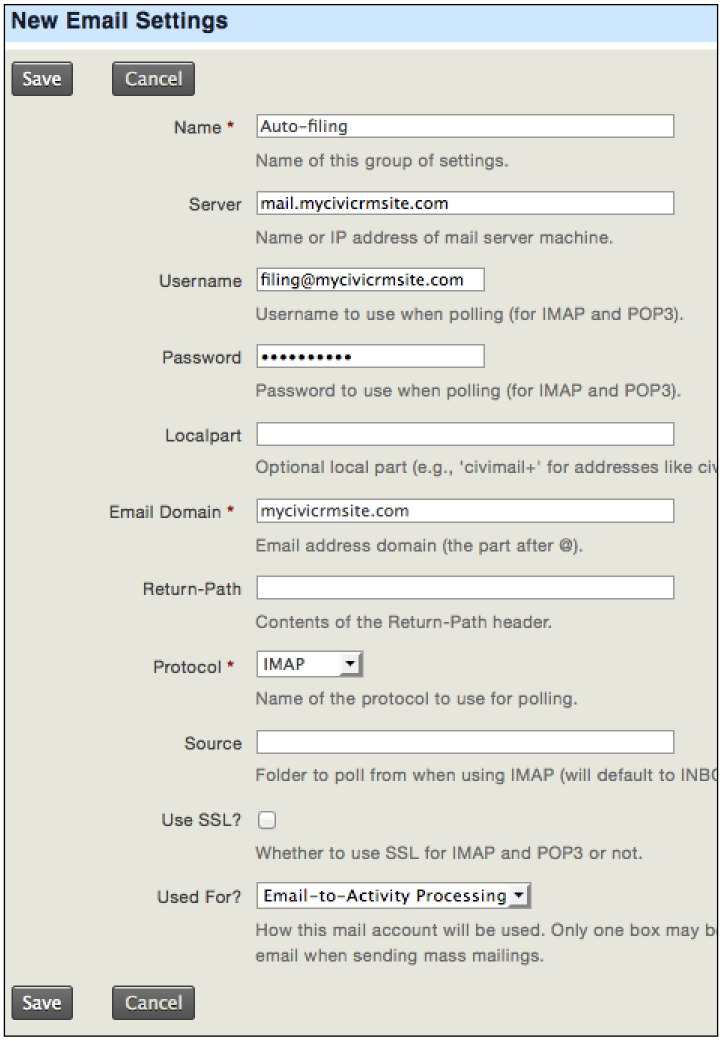
Interactions between your contacts and your organization are many and complex. A lot of these interactions will involve exchanges of e-mail. You may want to keep a record of these exchanges for each of your contacts. This is particularly useful in situations where you are dealing with a contact and you need to see a history of correspondence relating to the contact and other members of your organization. CiviCRM lets you do this by filing e-mail correspondence as an activity on each contact record.

How to do it...

We will set up an e-mail account that will act as a "dropbox" for messages that we want to file, and link this to CiviCRM.

1. Set up an e-mail account. You can use Gmail or an account provided by your hosting provider. In this recipe we will use an account called `filing@mycivicrmsite.com`.
2. Navigate to **Administer** | **System Settings** | **Enable CiviCRM components**, and make sure that **CiviMail** is enabled.
3. Navigate to **Administer** | **CiviMail** | **Mail Accounts**.

4. Click on the **Add Mail Account** button and complete the details for each account you are adding. Getting this right can sometimes be a matter of trial and error. Leave the **Source** field blank.



5. Create a test e-mail message in your e-mail client, and Bcc it to `filing@mycivicrmsite.com`.

6. Navigate to **Administer** | **System Settings** | **Scheduled Jobs**, and execute the job titled **Process Inbound Emails**.

7. Click on the **View Job Log** link to see the log entry. If the log error message is `Failure`, this is highly likely to be a connection problem, so you must go back to **Administer** | **CiviMail** | **Mail Accounts**, and make the necessary changes.
8. Navigate to **Reports** | **Contact Reports** | **Activities**. You will see that CiviCRM has recorded e-mail activities for the sender and the recipient of the e-mail.

How it works...

Each time CiviCRM processes inbound e-mails, it checks the e-mail account you had set up. It then processes each message. If the sender or recipient e-mail address is not held within CiviCRM, it will create a new contact record for each, and will file the e-mail activity.

If the contacts do exist, it files the e-mails as an activity for the sender and an activity for the receiver.

See also

- You can find detailed guidance on configuring CiviMail accounts at `http://wiki.civicrm.org/confluence/display/CRMDOC43/Autofiling+email+activities+via+EmailProcessor`

Creating new activities

Activities are fundamental to how CiviCRM works. They are a record of all interactions between your organization and your contacts. CiviCRM comes with a ready-made set of activity types, such as phone calls, meetings, and e-mails, that can be used in most circumstances.

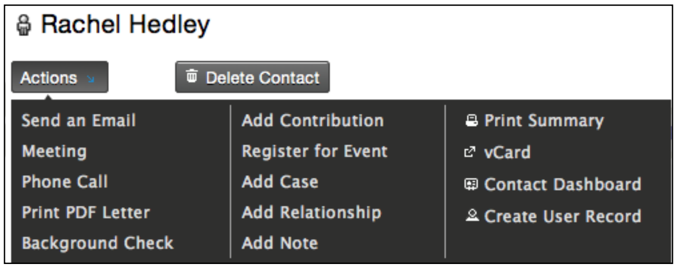
You can add your own activity types to suit your organization's needs. For example, yours may be an organization that performs background checks on volunteers before they are allowed to work with its clients. You could create an activity type called `Background Check` that would help you manage this process.

How to do it...

You should consider what activity types you need as part of the planning stage of your CiviCRM deployment. Adding a new activity type in CiviCRM is easy.

1. Navigate to **Administer** | **Customize Data and Screens** | **Activity Types**. Click on the **Add activity type** button. In the **Label** field, enter your activity. In this case we will use **Background Check**.
2. In the **Component** field, we can choose to have the new activity set against **Contacts**.

3. Save the new activity type.



How it works...

Test your new activity type by going to a contact and clicking on the **Actions** button present at the top left of the contact screen. This will show a drop-down list of available actions. Activities are listed in the first column.

See also

- The *Adding custom fields* recipe in this chapter
- You can find further details about CiviCRM Activities at `http://book.civicrm.org/user/current/organising-your-data/activities/`

Adding custom fields

Custom fields are a great way of storing and organizing data in CiviCRM. Custom fields are contained in custom datasets, and you apply each set to an object in CiviCRM, such as a contact type, an activity, or an event. For example, if you were organizing soccer teams, you might want to have a custom fieldset called `Soccer Data` that contains custom fields for playing position, goals scored, games played, and so on. Custom fields are searchable using advanced search.

Getting ready...

Custom fields need a bit of planning because once you have created a custom fieldset and applied it to an object, you cannot re-edit it and apply it to a different object. For example, let's say you are organizing a boat race. You want to collect information on boat size and boat type. You could choose to collect this custom information for each individual who applies to race, or for each actual participant in the race, or for each team in the race. So if you applied it to an individual contact and then changed your mind and only wanted to collect it for each participant, you would have to recreate the whole custom set of fields.

So you need to think about the following questions:

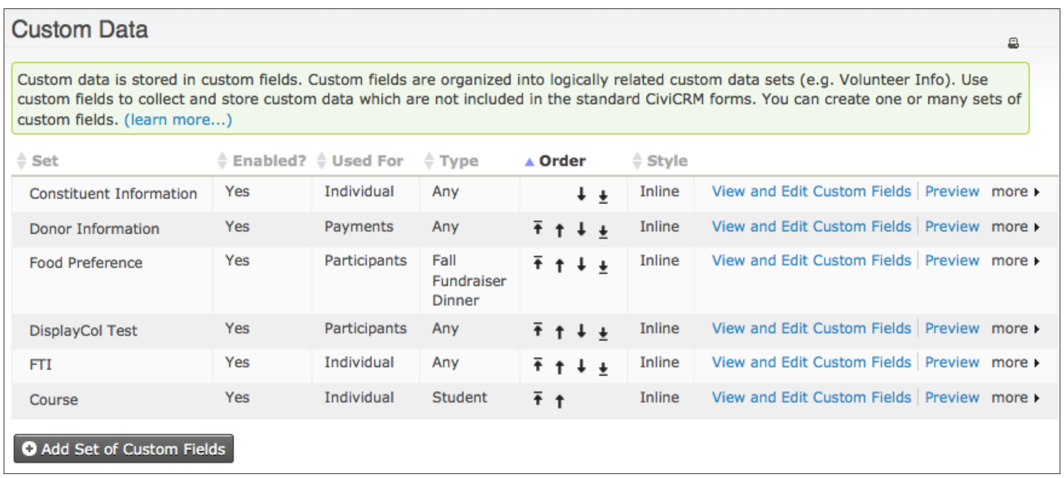
- What sort of unique data do you want to collect?
- What object do you want to apply the custom data set to?

In this example, we will add a simple custom field to the `Phone Call` activity. So when a phone call activity is recorded with a contact, the custom field will record if the call was general, a membership enquiry, or an event enquiry.

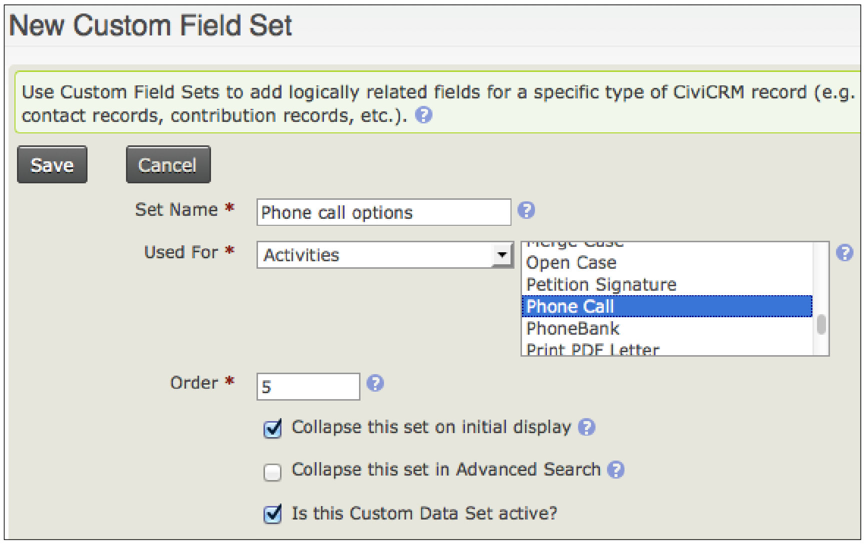
How to do it...

First we will create a custom data set, and then we will add some custom fields to it. In this recipe, we will add some fields to get data about phone calls.

1. Navigate to **Administer** | **Customize Data and Screens** | **Custom Fields**. You will see a screen that contains the current listing of custom datasets.



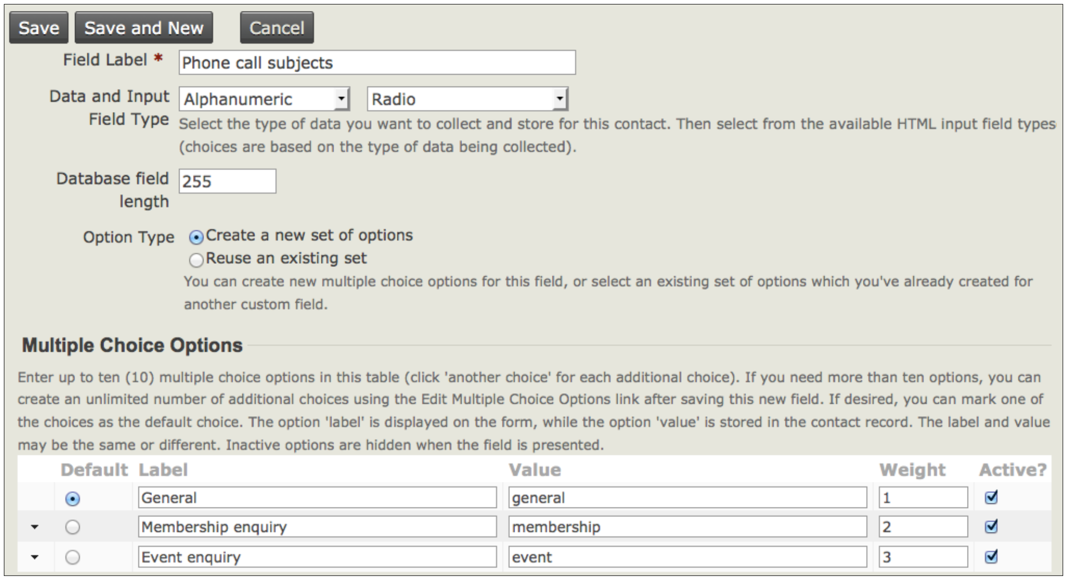
2. Click on the **Add Set of Custom Fields** button.



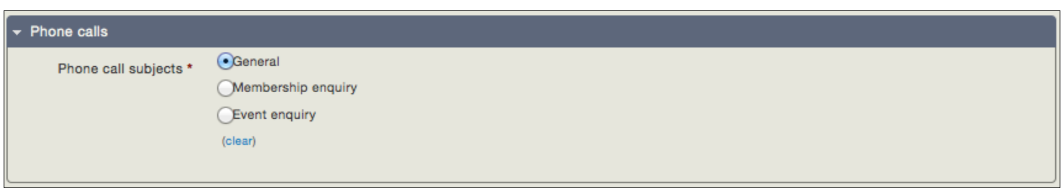
In this recipe, call the custom data set **Phone call options**, or substitute your own label.

3. In the **Used For** field, choose **Activities** and then choose **Phone Call**. This means that when a `Phone Call` activity is created, the fields will appear on the activity form for the user to complete.
4. The **Collapse this set on initial display** checkbox is checked by default. This means that when you look at the contact record, the custom fields will be hidden until you click on the custom fieldset title. Uncheck it.

5. Save the new custom fieldset and add custom fields.



6. Add a set of three options to record the nature of the phone call. It is beyond the scope of this book to go into the details of the various field types available.
7. Save the custom fields.
8. Navigate to a contact and add the `Phone Call` activity. The custom field is available.



There's more...

If you add custom fieldsets to contacts, you will get more options. You can add a fieldset multiple times to the same record. This is useful for recording employment histories or academic achievements.

You can create a custom fieldset for cases, relationships, groups, events, and memberships.

See also

- You can find further details about creating custom fields at `http://book.civicrm.org/user/current/organising-your-data/custom-fields/`

Using Scheduled Reminders for activities

Scheduled Reminders are a great new feature in CiviCRM. For example, you might have created an **Activity** for a colleague—perhaps a meeting that you have scheduled for next Friday. With Scheduled Reminders, you can send an e-mail reminder (say, the day before), reminding them to read the agenda for the meeting, and about the meeting itself.

You can also use Scheduled Reminders to accomplish the activity itself, if it involves e-mailing something. For example, you might schedule an activity, called `Welcome Information Pack`, for a new contact. `Welcome Information Pack` is an e-mail message that contains useful information and links back to your site for resources and so forth. You can configure a Scheduled Reminder as the `Welcome Information Pack` itself and have it sent to the contact at the scheduled time. This is how the Scheduled Reminder e-mail actually accomplishes the task.

Getting ready

You do not need to have cron running on your site to test this recipe. But, for this recipe to work on your live site, you must have cron running.

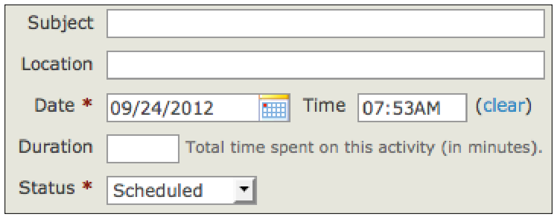
You will need to know how to set up and use mail templates within CiviCRM.

How to do it...

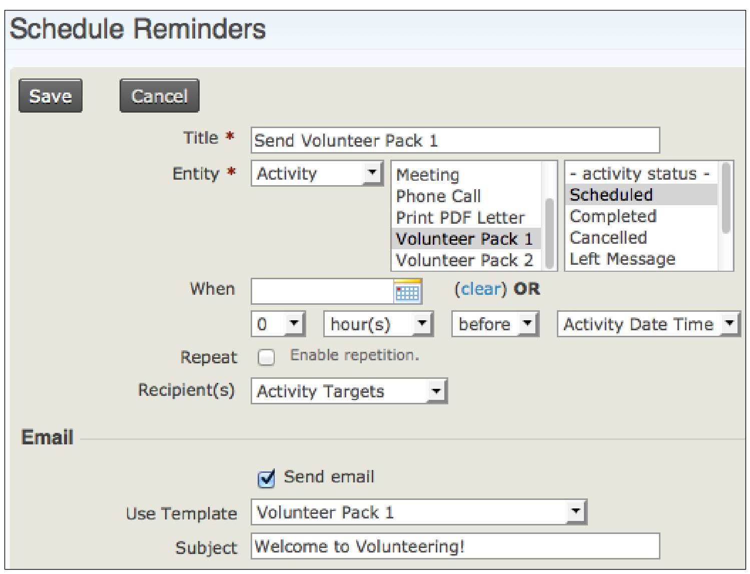
First we will set up the activities we want to schedule and then we will create the Scheduled Reminder for each activity. Every Scheduled Reminder uses a mail template. We will configure the mail template to accomplish the original activity.

1. Navigate to **Administer** | **Customized Data and Screens** | **Activity types**.

2. Set up two new activities, **Volunteer Pack 1** and **Volunteer Pack 2**. Make sure that the **Component** field is set to **Contact**. Now go to a test contact. Click on the **Actions** button and schedule the **Volunteer Pack 1** activity.



3. You do not need to fill in the **Subject** and **Location** fields. Fill in the **Date** field. For testing purposes set this to 10 minutes from the current time.
4. You do not need to fill in the **Duration** field. Make sure the **Status** field is set to **Scheduled**.
5. Now set up the **Volunteer Pack 2** activity in the same way. For testing purposes, set the date and time to 15 minutes from the current time.
6. Click on the **Activities** tab on the contact screen to check if your activities have been scheduled.
7. Navigate to **Communications** | **Schedule Reminders**, and click on **Add a reminder**.



8. Give the reminder a title such as **Send Volunteer Pack 1**. Select the **Volunteer Pack 1** activity and select **Scheduled** for the **activity status** field.

9. For **When**, select the default values: **0**, **hour(s)**, **before**, **Activity Date Time**. For **Recipient(s)**, select **Activity Targets**. Make sure the **Send email** checkbox is checked.
10. Now select an e-mail template, or prepare an e-mail using the rich text editor. Save the schedule.
11. Now add another schedule for the **Volunteer Pack 2** activity.
12. Navigate to **Administer** | **System Settings** | **Scheduled Jobs**. Navigate to the **Send Scheduled Reminder** job.
13. Check that the job is enabled, and for testing purposes, set the interval to **every time cron is run**.

How it works...

CiviCRM will now send out your reminders at the scheduled test times. Check your test e-mail account for incoming mails at regular intervals.

There's more...

The Schedule Reminders system does not automate your workflow.

If you look at the **Activities** tab for your test contact, you will see that the status is still set to **Scheduled** even after the reminder has been sent. This is because Scheduled Reminder is sending out a reminder about the activity, not accomplishing the activity itself.

You may wish to investigate writing a module that switches the activity to the status **Completed** once the reminder is sent.

See also

- You can find further information about Scheduled Reminders at `http://book.civicrm.org/user/current/email/scheduled-reminders/`

Using CiviCase to create an HR system

CiviCase was developed to manage and track interactions between an organization and it's clients in case management situations. It can be adapted to suit any internal or external processes that have regular, predictable, and reasonably well-defined workflows or procedures. Many NGOs generally have human resource functions such as hiring and training staff. Using CiviCase to manage these processes provides consistency, compliancy, and accountability to our human resource procedures. In this recipe we will configure a CiviCase type that will enable us to create and manage the employment records of staff.

How to do it...

CiviCase does not have a user interface for configuring CiviCase types. Instead, we create all the activity types and relationships that we need, and then we create an XML file that generates and schedules these activities when a case is opened. The CiviCase type we are going to create will handle three activities:

- **Contract acceptance**: This activity happens when our new employee signs the employment contract
- **Annual appraisal**: This activity happens when our employee is appraised for performance each year
- **Exit interview**: This activity happens when our employee leaves employment with our organization

We will use the XML file to also generate the relationship types associated with the employment record. These are:

- Line Manager
- HR Officer

1. Enable CiviCase by navigating to **Administer** | **System Settings** | **Enable CiviCRM Components**.
2. Check that you have set up a **Custom Templates** path for CiviCRM. This is a directory on your server that stores custom files for CiviCRM. This is where you will store the CiviCase XML file. Create a directory on your web server for your custom CiviCRM files called `custom_civicrm`. You can give it any name you like. Navigate to **Administer** | **System Settings** | **Directories** to set the path to the directory you just created.

3. Create the following directory path in your `Custom Templates` directory:

   `custom_civicrm/CRM/Case/xml/configuration`

4. Create a text file called `StaffRecord.xml` in the `custom_civicrm/CRM/Case/xml/configuration` directory, so the path to the file will be `custom_civicrm/CRM/Case/xml/configuration/StaffRecord.xml`.

5. Using a suitable text editor, enter the following XML code:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<CaseType>
  <name>Staff Record</name>
  <ActivityTypes>
    <ActivityType>
      <name>Open Case</name>
      <max_instances>1</max_instances>
    </ActivityType>
    <ActivityType>
      <name>Contract acceptance</name>
        <max_instances>1</max_instances>
    </ActivityType>
    <ActivityType>
      <name>Annual appraisal</name>
    </ActivityType>
     <ActivityType>
      <name>Exit interview</name>
        <max_instances>1</max_instances>
    </ActivityType>
       <ActivityType>
      <name>Change Case Type</name>
    </ActivityType>
    <ActivityType>
      <name>Change Case Status</name>
    </ActivityType>
    <ActivityType>
      <name>Change Case Start Date</name>
    </ActivityType>
    <ActivityType>
      <name>Link Cases</name>
    </ActivityType>
  </ActivityTypes>
  <ActivitySets>
    <ActivitySet>
      <name>standard_timeline</name>
      <label>Standard Timeline</label>
      <timeline>true</timeline>
      <ActivityTypes>
        <ActivityType>
          <name>Open Case</name>
          <status>Completed</status>
        </ActivityType>
        <ActivityType>
```

```
          <name>Contract acceptance</name>
          <reference_activity>Open Case</reference_activity>
          <reference_offset>1</reference_offset>
        </ActivityType>
<ActivityType>
          <name>Annual appraisal</name>
          <reference_activity>Open Case</reference_activity>
          <reference_offset>365</reference_offset>
          <reference_select>newest</reference_select>
        </ActivityType>
      </ActivityTypes>
    </ActivitySet>
  </ActivitySets>
  <CaseRoles>
    <RelationshipType>
        <name>HR Manager</name>
        <creator>1</creator>
    </RelationshipType>
    <RelationshipType>
        <name>Line Manager</name>
    </RelationshipType>
 </CaseRoles>
</CaseType>
```

6. Save the XML file.
7. Navigate to **Administer** | **Customized Data and Screens** |**Activity types**, and create the three activity types described in the XML document:
   - Contract acceptance
   - Annual appraisal
   - Exit interview

   Make sure the names of the activity types are exactly the same as shown in the XML document.

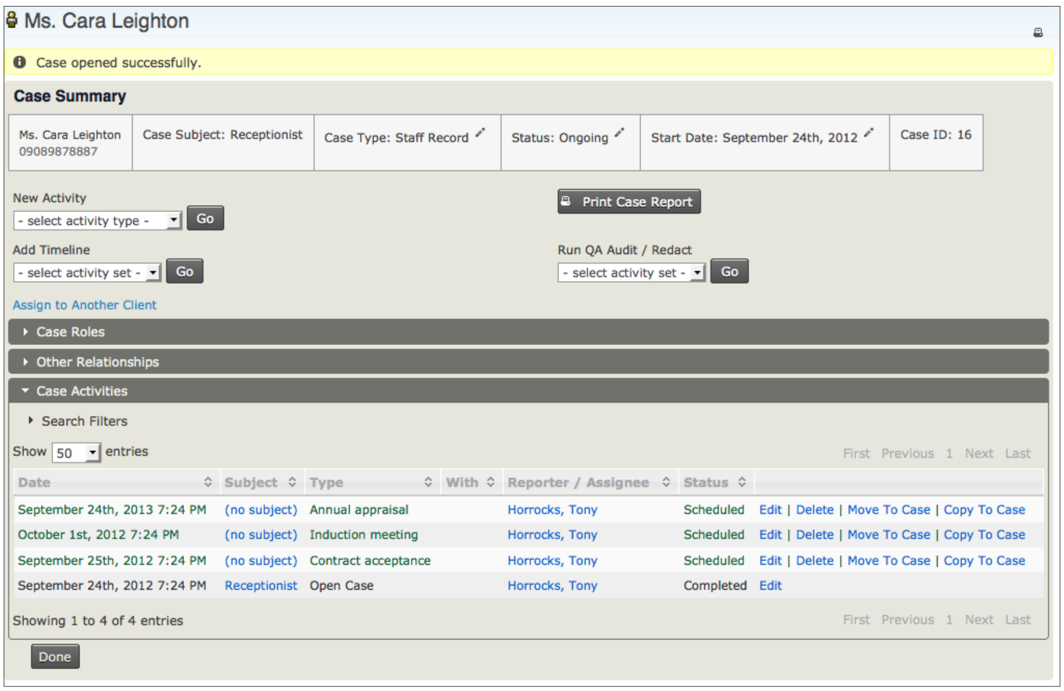
   Make sure that you select CiviCase as the component for each activity type.
8. Create the relationship types that are described in the XML document. Navigate to **Administer** | **Customize Data and Screens**, and create two relationship types, `HR Officer` and `Line Manager`. Make sure that these relationships have *exactly* the same names and capitalizations that you used in the XML file. Make sure you name the **Relationship Label** from **B** to **A**—exactly the same as the relationship type.

9. Navigate to this file on your web server: `sites/modules/civicrm/CRM/Case/xml/configuration.sample/settings.xml`, and copy it to the configuration directory, `custom_civicrm/CRM/Case/xml/configuration` you previously created.

   This is a global settings file for CiviCase. You do not need to alter it.

10. Navigate to **Administer** | **CiviCase**. Add a case type called `Staff Record`.

11. Navigate to a test contact, click on the **Actions** button and add a case, choosing **Staff Record**.



How it works...

The XML code does all the work for us once it is set up. Let's go through the structure. This provides the name of the case type that we will use:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<CaseType>
<name>Staff Record</name>
```

We have a section called `ActivityTypes` (note the plural!). It is a container for each activity type that is going to be associated with the `Staff Record` case.

```
<ActivityTypes>
<ActivityType>
      <name>Open Case</name>
      <max_instances>1</max_instances>
    </ActivityType>
</ActivityTypes>
```

CiviCase always starts with `<ActivityType>` named `Open Case`.

`<max_instances>` tells CiviCase how many instances of the activity to create. As a case is opened only once, there is only one instance.

```
<ActivityType>
    <name>Contract acceptance</name>
    <max_instances>1</max_instances>
</ActivityType>
<ActivityType>
    <name>Annual appraisal</name>
</ActivityType>
<ActivityType>
    <name>Exit interview</name>
    <max_instances>1</max_instances>
</ActivityType>
```

The three activity types that we will use in our CiviCase are described next. You can see that the activity type named `Annual appraisal` does not have a `<max_instances>` tag. This is because annual appraisals take place each year and there is no defined maximum.

Now that we have set up what activities we will use for our case, we can schedule some of them on a timeline. For this, we create another section, called `ActivitySets`, in the XML file.

```
<ActivitySets>
<ActivitySet>
<name>standard_timeline</name>
<label>Standard Timeline</label>
<timeline>true</timeline>
<ActivityTypes>
<ActivityType>
<name>Open Case</name>
<status>Completed</status>
</ActivityType>
<ActivityType>
<name>Contract acceptance</name>
```

```
<reference_activity>Open Case</reference_activity>
<reference_offset>7</reference_offset>
<reference_select>newest</reference_select>
</ActivityType>
</ActivityTypes>
</ActivitySet>
</ActivitySets>
```

Here we have the section called `ActivitySets`. It is a container for one or more instances of `ActivitySet`.

`ActivitySet` is a set of scheduled activities that CiviCase will generate when our `Staff Record` case is opened. When the case is first generated, CiviCase uses the `<standard_timeline>` activity set to generate the initial set of activities. You can have additional `ActivitySet` instances that use a different timeline. This is used to create activity branches within a case. In our example it could be the case that if an employee has a poor annual appraisal, we need to generate another set of activities to deal with the outcome. We can do this by having it configured in our XML file and applying it in the `Add Timeline` section of the CiviCase screen.

Within each `<ActivitySet>` instance, we have `<ActivityType>` again, and we have some tags to schedule each type.

`<reference_offset>` is the time in days that the activity will be scheduled. The offset is measured from whatever activity is entered in the `<reference_activity>` tag.

If the referenced activity has multiple instances, such as a training course, then we use the `<reference_select>` tag to pick the newest instance of the activity. If we do not want an activity schedule, we do not include it in `<ActivitySet>`.

Finally, we have a `<status>` tag that allows us to see the initial status of the activity when it is scheduled.

In our previous example, we have set the `Contract acceptance` activity to be scheduled seven days after the `Open Case` activity.

```
<CaseRoles>
<RelationshipType>
<name>Human Resources Manager</name>
<creator>1</creator>
</RelationshipType>
<RelationshipType>
<name>Line Manager</name>
</RelationshipType>
</CaseRoles>
```

Finally, there is an XML section where we can create our relationships for each case. Each relationship we create becomes a role within the case.

There's more...

This is just a very simplified example of what can be achieved using CiviCase. There are other ways you could apply the same principles: training schedules, volunteer induction programs, membership induction programs, as well as traditional casework applications.

See also

- *Chapter 1*, *Creating Activity Types*
- You can find more about CiviCRM relationships at `http://book.civicrm.org/user/current/organising-your-data/relationships/`
- You can find more about CiviCase at `http://book.civicrm.org/user/current/case-management/what-is-civicase/`

Installing languages and localizing CiviCRM

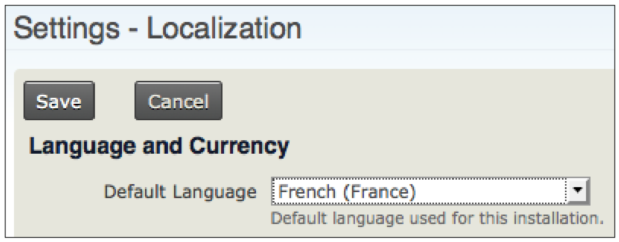
You might want your users to be able to use CiviCRM in a different language than US English. The CiviCRM user interface is available in many languages. This recipe shows you how to install these languages and localize your CiviCRM installation.

How to do it...

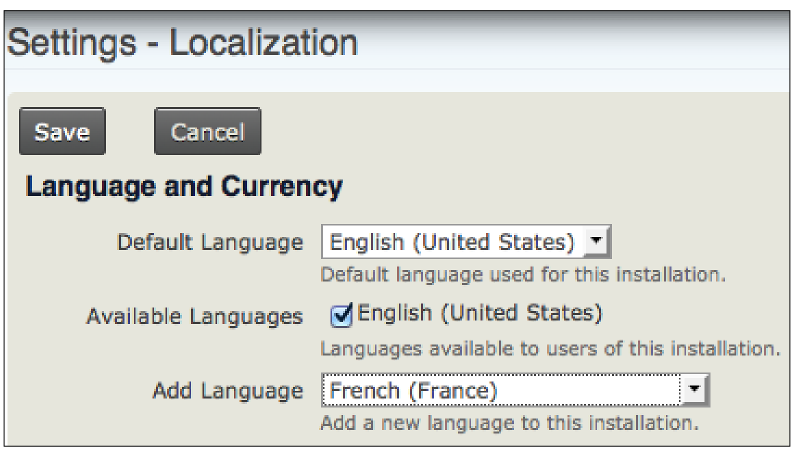
There are two stages to localizing CiviCRM. First we will download and install an interface translation and then we will configure CiviCRM using the **Localization admin** screens.

1. Go to `http://sourceforge.net/projects/civicrm/files/civicrm-stable` and open the latest stable version of CiviCRM. There you will see an archive that contains the translation files. It is called `civicrm-<version number>-l10n.tar.gz`. If you uncompress this archive you will see there are two folders, `l10n` and `sql`.
2. Copy the `l10n` folder into the CiviCRM module folder.
3. Copy the contents of the `sql` folder into the `sql` directory that already exists in your CiviCRM module folder. You can now explore what is in the `l10n` folder. It contains subdirectories for each language that CiviCRM supports. The name of each subdirectory is a locale code for the translation. These are reasonably easy to understand. For example, `es_ES` is mainland Spanish, `es_MX` is Mexican Spanish.
4. Remove languages that you do not wish to use from the `l10n` folder. Do the same for unused languages in the `sql` folder.
5. Navigate to **Administer** | **Localization** | **Languages, Currency, Location**.

6. At the top of the screen, select a new **Default Language**. In this case we will use French.

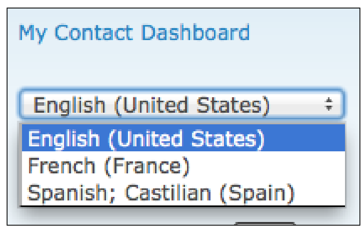


7. Complete the other settings on this page for currency, date formats, and addressing. Save your settings. The site is now displayed in French and supports the date, currency, and addressing formats that were configured.
8. Optionally under **Multiple Languages Support** you can check the **Enable Multiple Languages** checkbox. This allows the CiviCRM user to switch between two or more languages using the CiviCRM **Language Switcher** block available in your CMS. It also enables data entry in a different language.
9. If you have checked **Enable Multiple Languages**, save your settings and then at the top of the screen add in the extra languages that you require.



Here we are adding French. We can add extra ones by checking the checkboxes under **Available Languages**.

10. Save the settings.
11. In your CMS—in this case Drupal—navigate to **Structure** | **Blocks** and enable the CiviCRM Language Switcher block. The user can now switch between languages within CiviCRM.



See also

- You can find more about CiviCRM localization at `http://book.civicrm.org/user/current/the-civicrm-community/localising-civicrm/`

2
Organizing Data Efficiently

In this chapter we will cover:

- Adding contact types
- Adding a time-limited relationship
- Using tag sets to quickly organize data
- Using tags and groups to segment data
- Changing option lists
- Creating and updating a smart group
- Using Google Refine to prepare data
- Importing into CiviCRM using an import script
- Using external identifier deduping rules to import contacts
- Using Google Refine to create a unique ID
- Importing relationship data
- Exporting related data
- Batch updating using profiles

Introduction

Organizing data is a critical planning step in your transition to CiviCRM. What sort of contacts will you be managing and what are your data needs? How will your contacts be grouped and tagged? What relationships will they have between each other and you? What information will you want to get out of CiviCRM? Once you have a plan in place, you will want to get your existing data into CiviCRM the way you want. This often exposes the poor quality of your data, and the effort required to clean it up and get it into shape. This can sometimes take longer than the data planning stage. This chapter shows you some recipes that don't even use CiviCRM but will help get your data in shape. In other recipes we'll explore some of the techniques of organizing your data once you get it into CiviCRM.

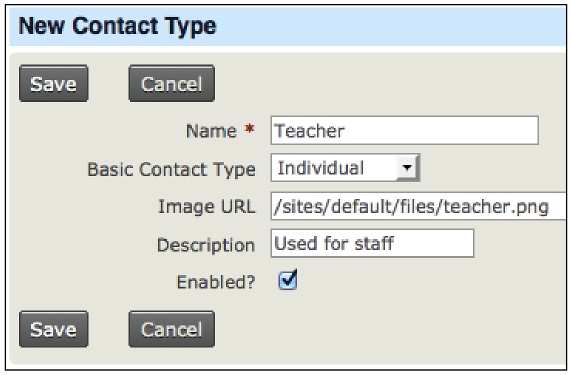
Adding contact types

Imagine you are managing a soccer club using CiviCRM. For some contacts—players—you would want to record the goals scored, attendance, age, playing position, injuries, and other data. For other contacts such as coaches, you would want to collect different data: first aid qualifications, coaching qualifications, and so on. Having both these sets of data showing up on every contact summary screen does not make sense. What you need is to have two different sorts of contacts; contacts who are players and contacts who are coaches so that they both hold separate data. CiviCRM makes this easy to accomplish.

How to do it...

CiviCRM comes with three main contact types: **individuals**, **households**, and **organizations**. As we are dealing with individuals, we will create two subtypes of the the individual contact type, teachers and students. These subtypes inherit all the information you can collect from their parent contact type and can be extended to collect more information using custom field sets. This allows you to segregate your contact data more efficiently and makes data input easier. Perform the following steps to add different contact types:

1. Navigate to **Administer** | **Customize Data and Screens** | **Contact Types** and click on **Add a Contact Type**. You will then find the edit screen that looks like this:



2. Give your new contact type a name and pick the parent contact type. You can also provide a link to an icon for your contact type.

How it works...

Contact subtypes are most useful in situations where you want to collect specific and unique data for each type.

Contact subtypes are useful in building CiviCRM relationships. For example, you can build a relationship between a teacher and a student where the teacher has to be a teacher subcontact and the student a student subcontact.

There's more...

You can create custom field sets targeted at each subcontact you create.

Navigate to **Administer** | **Customized Data and Screens** | **Custom data** to do this.

> Contact subtypes become available for use in CiviCRM core functions such as importing, exporting, and searching data.

See also

- The *Adding custom fields* recipe in *Chapter 1*, *Setting Up CiviCRM*
- Find out more about contact types at `http://book.civicrm.org/user/current/organising-your-data/contacts/`

Adding a time-limited relationship

Relationships are a central feature of CiviCRM. For example, you may run an organization that has contacts related to it because they are volunteers, or you may have individual contacts that are the children of other contacts who are their parents. Some relationships are time-limited. For example, the chair of an organization may have a one year fixed term of office.

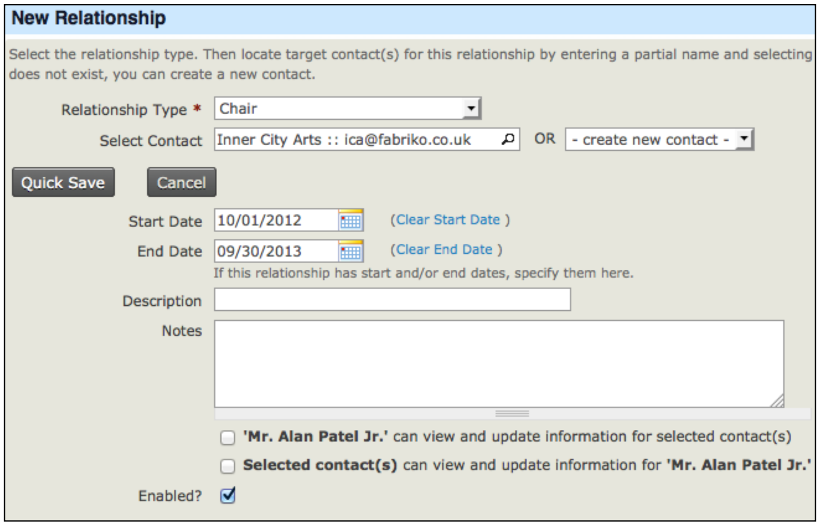
How to do it...

In this recipe we will set up a time-limited relationship between two contacts using the following steps:

1. Navigate to **Administer** | **Customize Data and Screens** | **Relationship Types** and click on **Add relationship**:

2. Set **Relationship Label-A to B** to `Chair`.
3. Set **Relationship Label-B to A** to `Chair is`.
4. Set **Contact Type A** to **Individual**.
5. Set **Contact Type B** to **Organization**.
6. Set a suitable description in the **Description** field.
7. Check if the relationship is enabled.
8. Save the relationship.

9. Navigate to an individual contact, click on the **Relationship** tab, and add the relationship.



10. Select the relationship type **Chair** and select the contact to relate to. CiviCRM autocompletes your entry from the organizations available in the database.
11. Enter a start date and an end date for the relationship and add a description.
12. Navigate to **Administer** | **System Settings** | **Scheduled Jobs** and enable and set the **Disable Expired Relationships** job.

How it works...

Contact Type A is set to **Individual**. **Contact Type B** is set to **Organization**. This means that only individuals can be chairs of organizations.

The Disable Expired Relationship job checks the dates on relationships and disables them if they have expired. This requires cron in order to function properly.

There's more...

You can allow the two related contacts to be able to view and edit each other's records by using the checkboxes below the **Notes** field. For example, you could allow the chair to be able to edit the organization record. This would also require the individual contact to have an active account on your CMS and permission to access their CiviCRM contact dashboard.

> Relationships become available for use in CiviCRM core functions such as importing, exporting, and searching data. They are also used in creating membership types.

See also

- The *Setting up cron using cPanel* recipe in *Chapter 1*, *Setting Up CiviCRM*
- Find more about CiviCRM relationships at `http://book.civicrm.org/user/current/organising-your-data/relationships/`

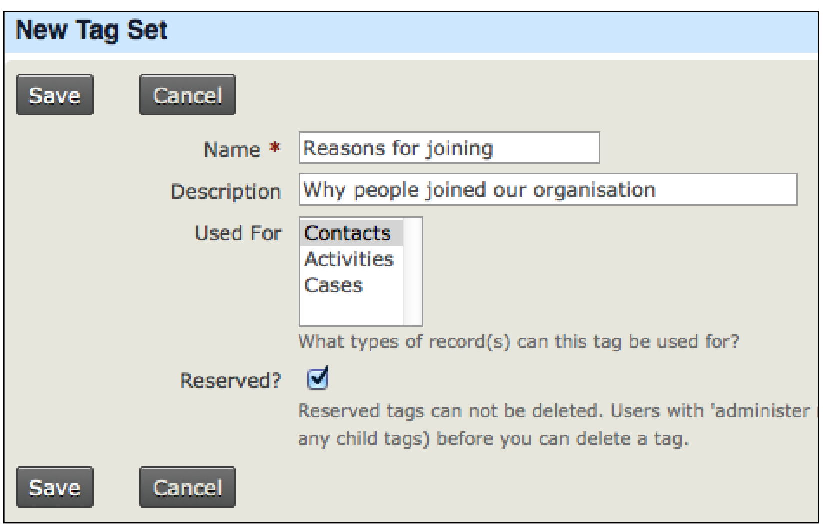
Using tag sets to quickly organize data

Tags are a way of organizing your contacts, activities, and cases within CiviCRM. You can regard a tag as a way of *describing* data and as a powerful way to *segment* your contacts. Any tags you create are visible on the contact edit screen so they are great with getting summary information about a contact quickly. You need to include tags in your data plan for CiviCRM, otherwise your screens can get overrun by redundant tags.

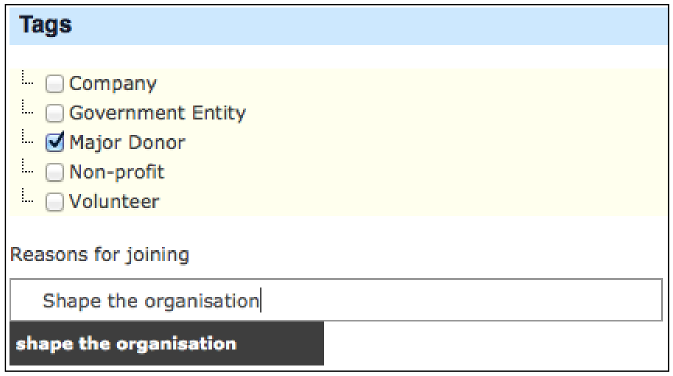
How to do it...

Tag sets allow you to add tags on the fly without going to the **Manage Tags** screen. Here are the following steps to do so:

1. Navigate to **Contacts** | **Manage tags** and click on **Add a Tag Set**.



2. Enter the name and description for your tag set, and apply it to **Contacts**, **Activities**, **Cases**, or any combination of these.
3. Navigate to a contact record and click on the **Tags** tab.



4. Now start typing in new tags into the **Reasons for joining** box on the edit screen.
5. CiviCRM turns this into a tag option (in black). Click on the tag option to add it.
6. You can add as many tags as you like. CiviCRM will autocomplete any existing tags to reduce duplication.
7. Navigate to **Search** | **Advanced search**. You will see a screen like this:

8. Here you can enter one or more of the tags contained within the tag set.

> We could also enter the word `Shape` into the **All Tags** field. This would pick up any items that have tags that contain the word "Shape".

See also

- Find out more about CiviCRM groups and tags at `http://book.civicrm.org/user/current/organising-your-data/groups-and-tags/`

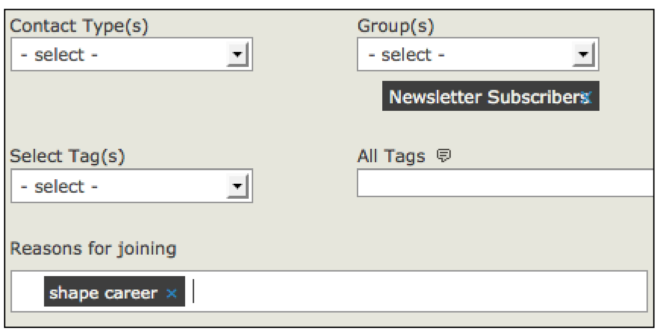
Using tags and groups to segment data

Tags are ways of describing your contacts. When you look at a contact summary screen, all the tags applied to the contact are visible, providing at-a-glance information without having to drill down. Contacts can also be segmented into groups. Groups are not visible on the contact summary screen so they are not useful for getting instant information. They come into their own when they are used for other actions within CiviCRM. For example, they can be used to control permissions, or as mailing lists. Once you get to a group listing there are a wide range of actions you can apply to contacts within the group.

How to do it...

This recipe shows you how to combine group and tag data in a search to target newsletter readers.

1. Create a group in CiviCRM and allocate contacts to it. In this example we have a group called **Newsletter Subscribers**.
2. Create a tag or a tag set and apply tags to some of the contacts within the group.
3. Now navigate to **Search** | **Advanced Search**.



4. Here you can see that we are searching within the **Newsletter Subscribers** group to see which ones are tagged with **shape career**.

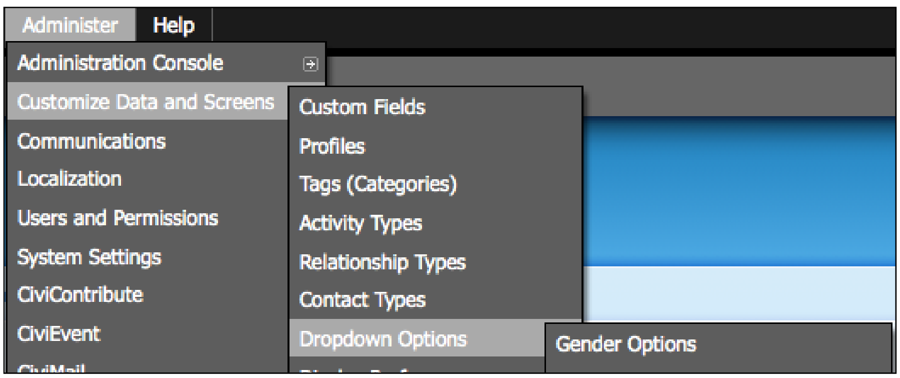
From the result set you could make a decision on whether to write an article about careers in the next edition of the newsletter.

Changing option lists

Option lists are used throughout CiviCRM to ensure data integrity and to make CiviCRM core functions work. Some of these are not available from the menu system.

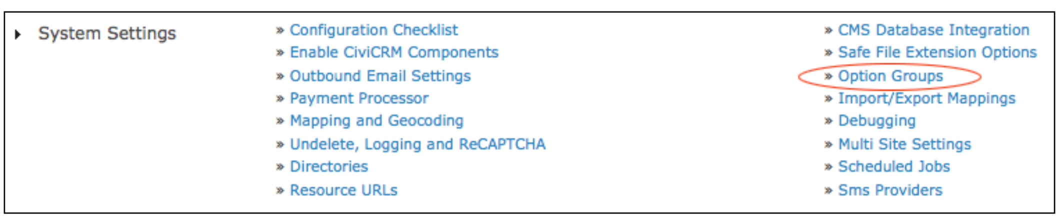
How to do it...

The option lists available through the menu system are limited. There is a main screen where you can edit all option lists:



Here there are only a few drop-down options available to edit.

1. Navigate to **Administer** | **Administration Console**.



2. In **System Settings** you will see the **Option Groups** link in the second column. This provides you with the full list. Many of these option lists are generated internally by CiviCRM so you need to be very careful about which ones you alter.

Creating and updating a smart group

Searching for contacts and doing something with the set of results is a day-to-day task in CiviCRM. For example, you might want to find all your contacts that live in Alaska and send them a mail shot. This is pretty easy to do in Advanced Search, and you can add all the contacts you find into a group called Alaska. This means that you can always go back to the group and find the contacts you added. The group represents a snapshot of your data that you took when you did the search and added contacts to the group.

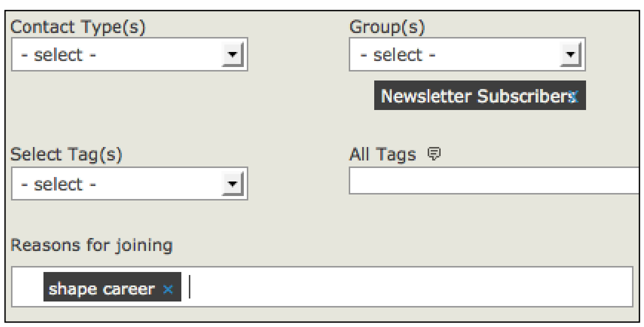
The problem is that if new contacts are added into CiviCRM who live in Alaska, you will have to remember to add them to the group, and you will have to remove them from the group if they move from Alaska.

To overcome this problem, CiviCRM has smart groups. Smart groups automatically contain any contacts that match your search criteria. So, if the data in your contacts changes, or contacts are added or removed, this is reflected dynamically in the smart group without you having to do anything.

How to do it...

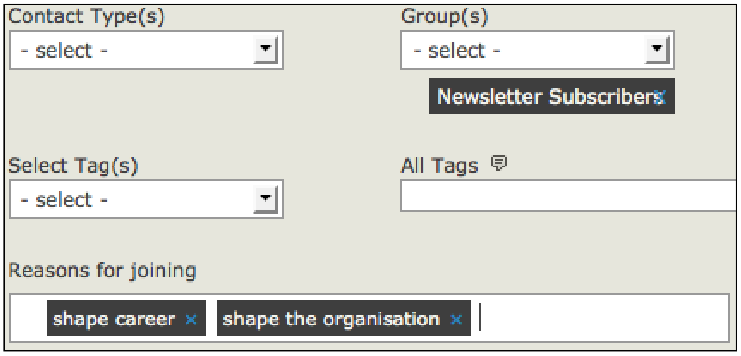
Creating a smart group is easy. We just perform a search and create the smart group directly from the search results screen:

1. Navigate to **Search** | **Advanced Search**.



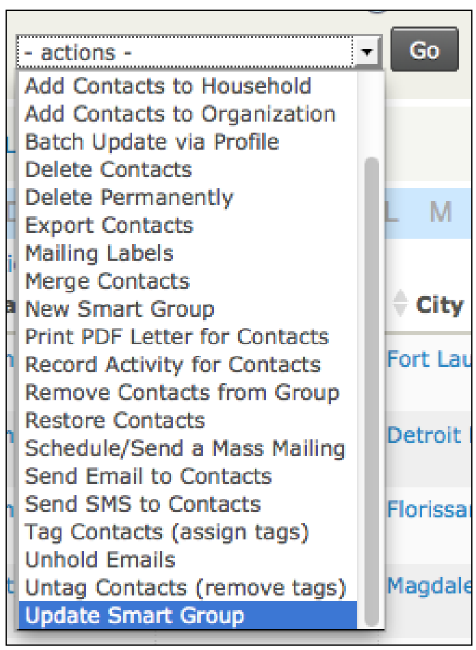
2. In this example we are looking inside an existing group called **Newsletter Subscribers** for contacts tagged with "shape career".
3. From the search results, use the **actions** drop-down menu and add these contacts to a new smart group and call it **Newsletter Shapers**.
4. You can access the smart group by navigating to **Contacts** | **Manage Groups**.
5. Now let's suppose we want to change the criteria. Locate the smart group you created and click on the **Contacts** link. This will list the contacts in the smart group.

6. Click on the **Edit Search Criteria** button at the top of the group screen. This will display the **Advanced Search** pane with the existing criteria loaded.
7. Amend the search criteria to get a different result set.



Here we added an extra tag to our search criteria. Now click on **Search** to see the new set of results.

8. If the search results are what you want, you must use the **actions** drop-down menu to update the smart group as shown in the following screenshot:



How it works...

Smart groups are based on search criteria. They are effectively saved searches. So if you find yourself repeatedly doing the same search, save the results as a smart group to save yourself time.

See also

- Find more about CiviCRM groups and tags at `http://book.civicrm.org/user/current/organising-your-data/groups-and-tags/`

Using Google Refine to prepare data

Preparing data for CiviCRM import can be a time-consuming, frustrating, and traumatic experience. But it is a job that has to be done. CiviCRM does an enormous amount of error checking on data import and will not import records with errors it spots.

Consider this data:

Preesall	Poulton Le Fylde	Lancashire
Preesall	Poulton Le Fylde	Lancashire
	Poulton-le-Flyde	Lancashire
Carlton	Poulton-Le-Fylde	Lancashire
Furness Driv	Poulton-le-fylde	Lancashire

Here you can see that there are data inconsistencies in the center column. The town Poulton-le-Fylde has five different ways of spelling and presenting the data. This is quite a common problem in legacy systems that were designed to hold addressing data for label printing rather than for searching or geocoding. Another common problem is having data in the wrong columns. Towns, cities, and postcodes are often spread across many columns. The result is you cannot guarantee accurate search results or do any geocoding.

How to do it...

Google Refine is an excellent tool for cleaning your data, which is free and easy to use. This recipe shows you some of the basics of Google Refine.

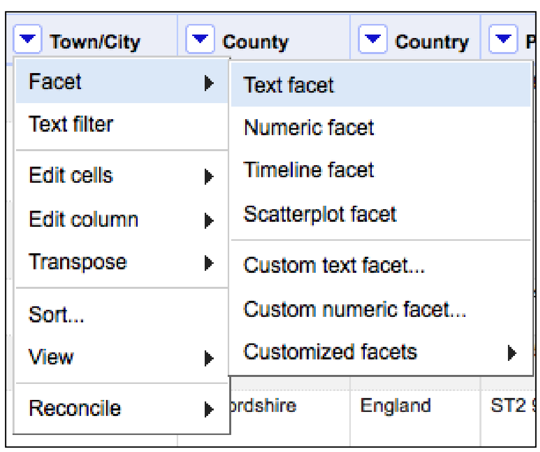
1. Download and install Google Refine from `http://code.google.com/p/google-refine/wiki/Downloads?tm=2`.

2. You can import data from CSV, Excel, or a variety of other files into Google Refine to see how the import mechanism works. It is incredibly easy. Here we have launched Google Refine and have imported this data:

Street	Area	Town/City	County	Country	Post Code
6 Longton Grove Road		Weston Super Mare	Somerset	England	BS23 1LT
437 Beechdale Road		Nottingham	Nottinghamshire	England	NG3 3LF
439 Beechdale Road			Nottingham	England	NG8 3LF
61 Hatfield Road		St Albans	Hertfordshire	England	AL1 4JE edit
11 Church Street	Harston	Cambridge	Cambridgeshire	England	CB2 5NP

Here you can see that the **Area** column is mostly empty. The **County** column contains a city, **Nottingham**, which should be in the city column. If you were importing thousands of addresses it would be very time-consuming to try and fix the data using a spreadsheet or a database. Google Refine makes it easy.

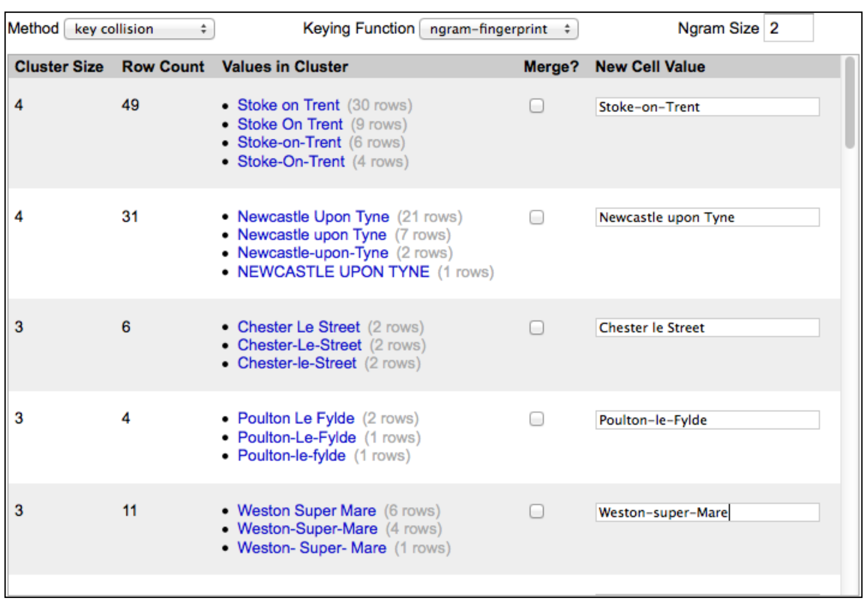
3. At the top of the **Town/City** column we can choose **Text Facet** as shown in the following screenshot:



This creates a list in the left-hand pane of the Google Refine screen:

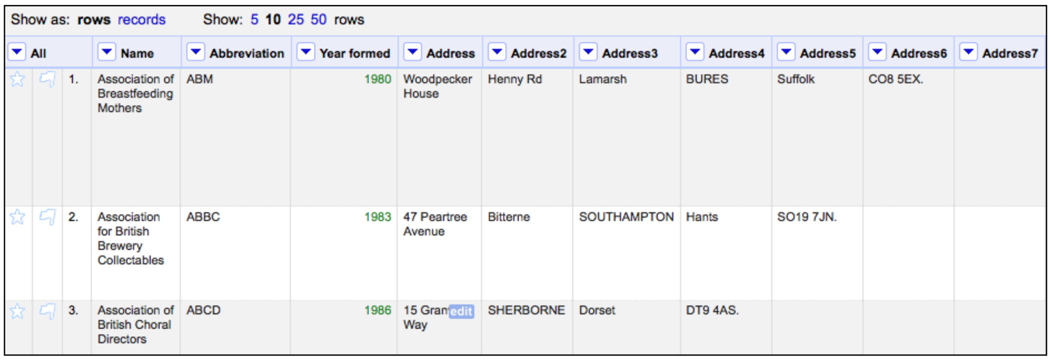
The list contains the unique values in the **Town/City** column and how many rows contain each unique value. So in this example we can see there are 1482 different town/city choices, and that for the choice **Aberdeen** there are 13 rows that have that value.

4. Click on the **Cluster** button in the top-right of our list; we can group these values based on algorithms that Google Refine provides:



You can play around with the **Method** and **Keying Function** values to see which values in the cluster suit you best. In the preceding screenshot you can see that each cluster contains spelling and formatting errors in the address data.

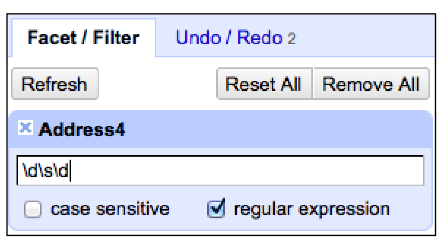
For each cluster you can then enter a value in **New Cell Value** and then update the data. You can work through your data in this way and clear up errors very quickly.



5. In the preceding data, some postal codes contain a "period", and both postal code and city data are scattered in different columns, so if you wanted to use this to geocode data it would not work.
6. Select the **Address2** column and pick up postal code data and store a copy in the **Address6** column.
7. From the drop-down menu on the **Address4** column, select **Text Filter**. If you look at the postal codes, you can see that there is a pattern within them. The postal code prefix ends with a number, then there is a whitespace, then the postal code suffix begins with a number. So you can search for that pattern using a regular expression. You can find out more about regular expressions at `http://www.regular-expressions.info`.

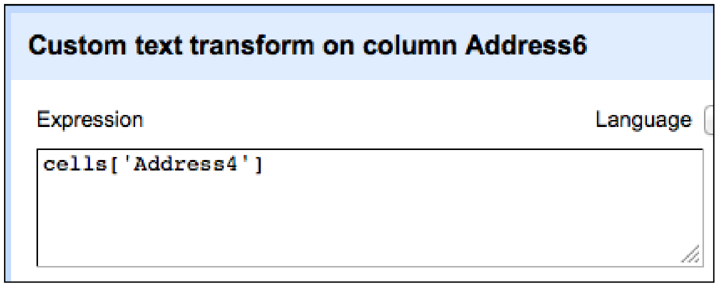
   The expression for any digit is `\d`, and for a space it is `\s`.

8. Search on `\d\s\d`. That is the same as saying "find cells in this column that contain the pattern of a number followed by a space followed by another number."

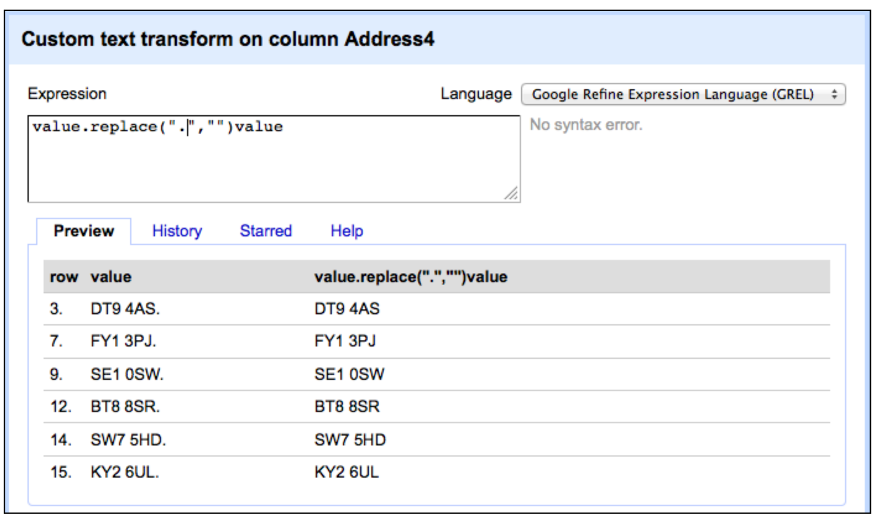


9. Tick the **regular expression** box.

10. As you type in the filter, Google Refine automatically refreshes the data to meet the filter criteria. You will see that this filters the postal code data properly.
11. Copy your found postal code values to the **Address6** column. Go to **Edit Cells** | **Transform** from the **Address6** drop-down menu and copy the data from the **Address4** column using Google Refine's syntax, as shown in the following screenshot:



12. Repeat this procedure for every column you need to fix for the postal codes.
13. Once you have them all in the **Address6:** we can rename the column. We go to the **Address6** column menu and select **Edit Column** and rename the column to `Postal Code`.
14. Finally we can remove the period from the data. We will have to go to **Edit Cells** | **Transform** to do so. And the resultant page is shown in the following screenshot:



How it works...

Google Refine gives you a dynamic view of your data directly as you transform it.

Google Refine stores all of your transformations. This means they can be reapplied to data again. In situations where the same data needs to be imported several times during the course of a development project, this is a godsend.

See more

- Find more about Google Refine at `http://code.google.com/p/google-refine/`
- Find the Google Refine documentation at `http://code.google.com/p/google-refine/wiki/DocumentationForUsers`

Importing into CiviCRM using an import script

There are occasions where you want to get some data into CiviCRM but there is no quick way of doing it. For example, your existing contacts may all be tagged. You want to get these tag values into CiviCRM so that when you import your contacts your tags work properly.

The CiviCRM interface only allows you to add one tag at a time. So this could be very time-consuming if you have hundreds of tags. This recipe introduces the use of the command-line interface to rapidly add data to CiviCRM.

The recipe can be used to migrate data into most CiviCRM tables. It's not as terrifying as it sounds.

Getting ready

First, you must have a local testing environment.

A local testing environment is a CiviCRM installation that runs on your own computer rather than the Internet.

In this recipe our local testing environment was set up on a Mac using the MAMP software. There are similar setups for Windows-based machines.

Once your local testing environment is set up you need to be able to execute PHP from the command line. PHP is the scripting language that powers CiviCRM. We want to be able to type in a command that will run a PHP file that will control our data import.

Navigate to where MAMP stores the PHP executable file, normally `/Applications/MAMP/bin/php/php5.3.14/bin/php`. Make a note of this path.

Now let's open the Terminal application and enter the following command:

```
open -a TextEdit .bash_profile
```

Hit the *Enter* key.

This makes the **TextEdit** application open the `.bash_profile` file, which is normally a hidden file. Now let's add the following line to `.bash_profile`.

```
alias phpmamp="<path> "
```

Here, `path` is the path to the PHP executable file.

In this example the line reads:

```
alias phpmamp="/Applications/MAMP/bin/php/php5.3.14/bin/php"
```

Now save the file.

What this means is that when you type `phpmamp` into the Terminal application it will execute the version of PHP held in the MAMP installation.

Test it by going to the Terminal application and typing `phpmamp –help`.

If all is well, it will return a list of help options for PHP.

How to do it...

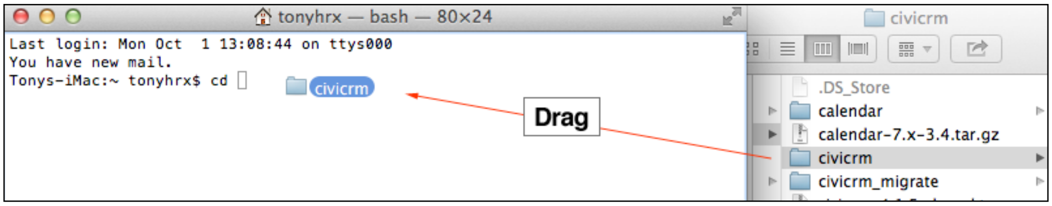
We will prepare a CSV file for import, and call the `import.php` CiviCRM file using a command-line interface.

1. Prepare tag data as a `.csv` file. Each tag must have a name and a description. Your file might look like this:

   ```
   name,description
   Learner, People who join to learn things
   Influencer, People who want to influence our organization
   Participator, People who want to attend our events
   Status booster, People who gain professional status
   Like minders, People who agree with our politics
   Passives, People who just want our resources
   ```

2. Open the Terminal application and navigate to the `site module` folder that contains your CiviCRM installation in your local testing environment. This is easy.

3. Navigate to the CiviCRM module folder using the Finder and make sure it is visible when you use Terminal. Then type `cd` into the Terminal application window. Then drag the `civicrm` folder onto the Terminal window. This accomplishes the navigation with the minimum of work:



4. Enter the command to import the data. In the Terminal application, enter:

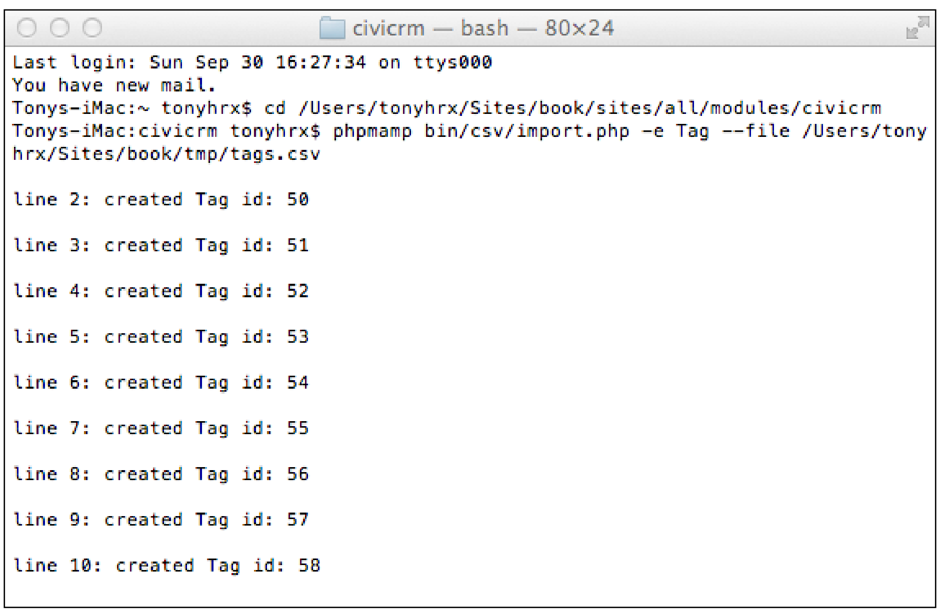
```
phpmamp bin/csv/import.php -e Tag --file <path_to_file>
```

   Here, `path_to_file` is the place where you stored your CSV file. You can grab this path by dragging the file from the Finder into the Terminal window.

   In this example the finished text will be:

```
phpmamp bin/csv/import.php -e Tag -file /Users/tonyhrx/Sites/book/
tmp/tags.csv
```

5. Copy and paste this into the Terminal window and press *return*. You should get the following result:



How it works...

`phpmamp` invokes PHP to run `bin/csv/import.php`.

`-e` tells the import script what CiviCRM entity we are going to target, in this case **Tag**.

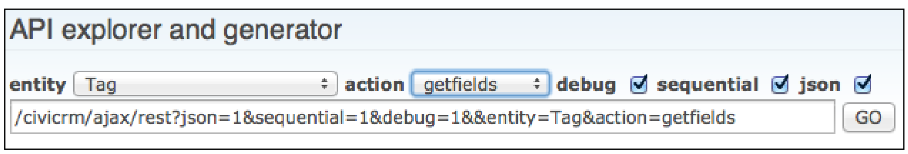
`--file` is the absolute path to the file of data we are going to import, in this case `/Users/tonyhrx/Sites/book/tmp/tags.csv`.

Then the CiviCRM API does the rest.

There's more...

This simple but powerful technique can take hours off import times and accomplish most of your import routines.

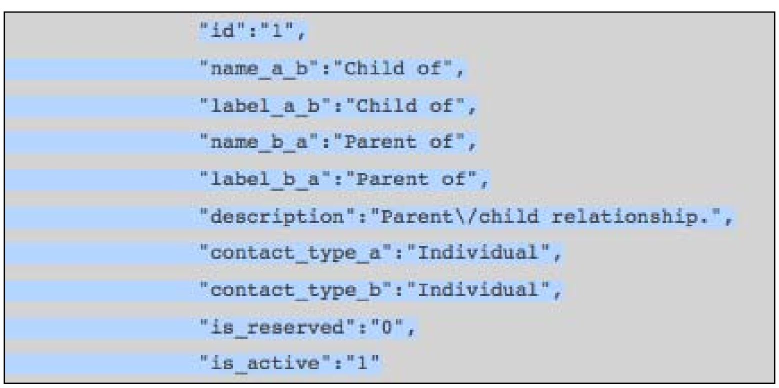
You can explore the CiviCRM API on your own installation. Just type in `civicrm/api/explorer` after your domain URL:



If we look at the Tag entity we can see what fields it contains by selecting **getfields** from the **actions** drop-down menu.

You can do this for any entity that is listed.

For example, let's look at the Relationships entity and see what the fields are:



Using this, we could construct another CSV file and add relationship data to it such as:

```
name_a_b,label_a_b,name_b_a,label_b_a,description,contact_
type_a,contact_type_b,is_reserved,is_active
Secretary,Secretary,Secretary is,Secretary is,Used for Organization
secretaries,Individual,Organization,0,1
```

And then run the import using Terminal:

```
phpmamp bin/csv/import.php -e RelationshipType --file /Users/tonyhrx/
Desktop/relationships.csv
```

See more

- Refer to *Chapter 11*, *Developing for CiviCRM* for more information
- Find out more about this technique at `http://civicrm.org/blogs/xavier/api_batch_tools`

Using external identifier deduping rules to update contacts

When data is imported to update contacts, CiviCRM has a set of rules to match contacts already in the database. By default, CiviCRM uses the e-mail address. If it finds a matching e-mail address it will update the matching contact. A problem will arise because not all of your contacts will have an e-mail address. So if we try to import some updates, only those contacts with an e-mail address will be updated.

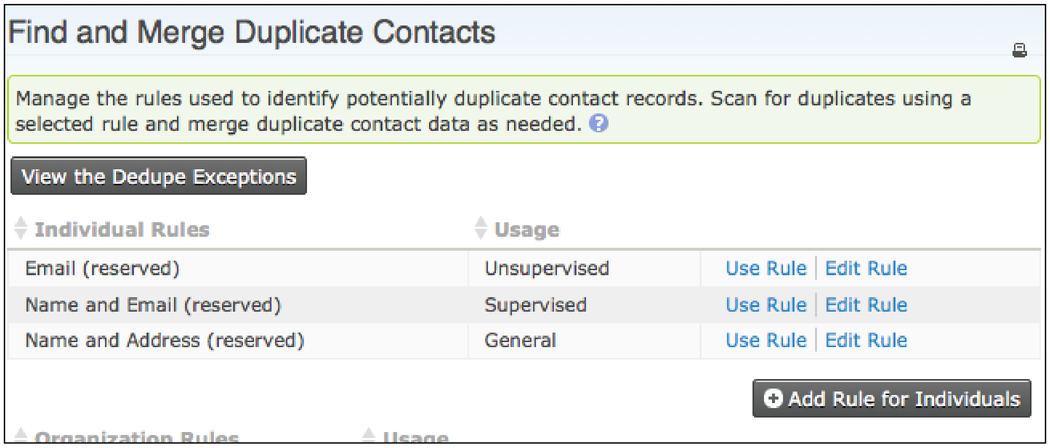
In these situations we need to have a different unique identifier that we can let CiviCRM use to match records.

How to do it...

We will use the **External Identifier** field in CiviCRM as our unique identifier. This field has to have a unique value for each contact. This will provide us with a good alternative to e-mail addresses to match contact data.

1. Navigate to **Contacts** | **Find and Merge Duplicate Contacts**.

The screenshot shows the deduping rules that are available to use for each contact type:

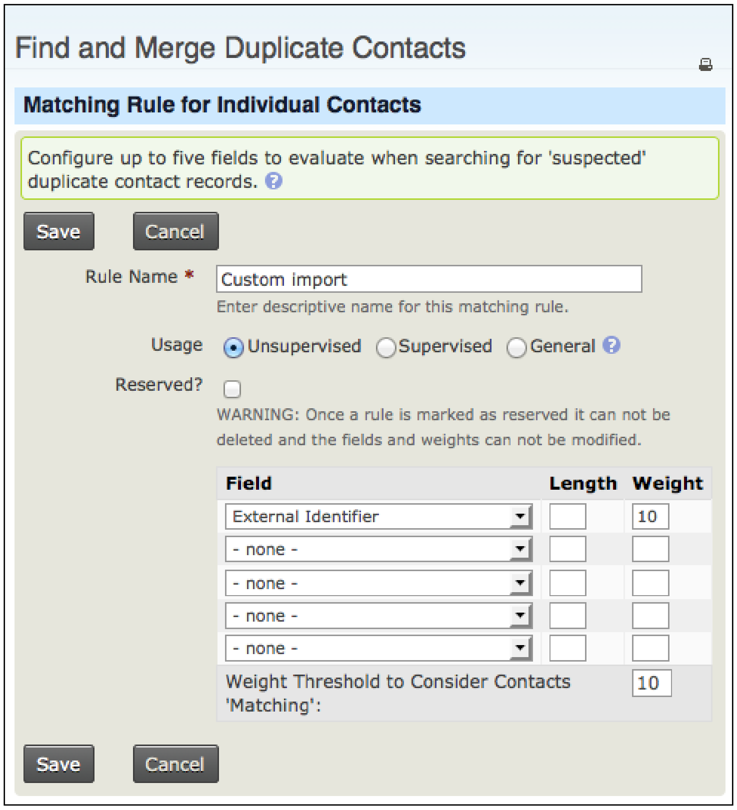


If we were importing data for individuals then CiviCRM would use the **Email (reserved)** rule that matches on e-mail addresses. So we need a different unique identifier for each contact that we can use instead of the e-mail address.

Quite often legacy data will already have a unique identifier, such as a client ID or membership number. If you already have an existing unique identifier then it is a good idea to have mapped this to CiviCRM's **External Identifier** field when your data is first imported.

If you do not have a unique identifier, then you can construct one using Google Refine.

2. Click on **Add Rule**. The following screenshot is what you will find:



3. Give the rule a name. Here we used **Custom import**.
4. Select **Unsupervised** for **Usage**. Unsupervised rules are used for imports and entries through online forms such as event signups or contact information using profiles. Supervised rules are used when user information is added manually through the contact screen.
5. Enter the external identifier as the first matching field with a weight of 10. Make the weight threshold to trigger the rule at 10. That's it. You can now use this rule to match contacts when you are importing data.

See also

- The *Using Google Refine to create a unique ID* recipe
- Find out more about CiviCRM deduping and merging at `http://book.civicrm.org/user/current/common-workflows/deduping-and-merging/`

Using Google Refine to create a unique ID

Having a unique ID for your data is important. When you first import your data into CiviCRM you can store this unique ID in the **External Identifier** field. When you subsequently update this data, you can use the external identifier as the field on which to match records. But sometimes you simply don't have a unique ID for contact data you want to import into CiviCRM.

How to do it...

Here we will use Google Refine to create our unique ID. Once we have this, we can use it to match contact data during CiviCRM import operations:

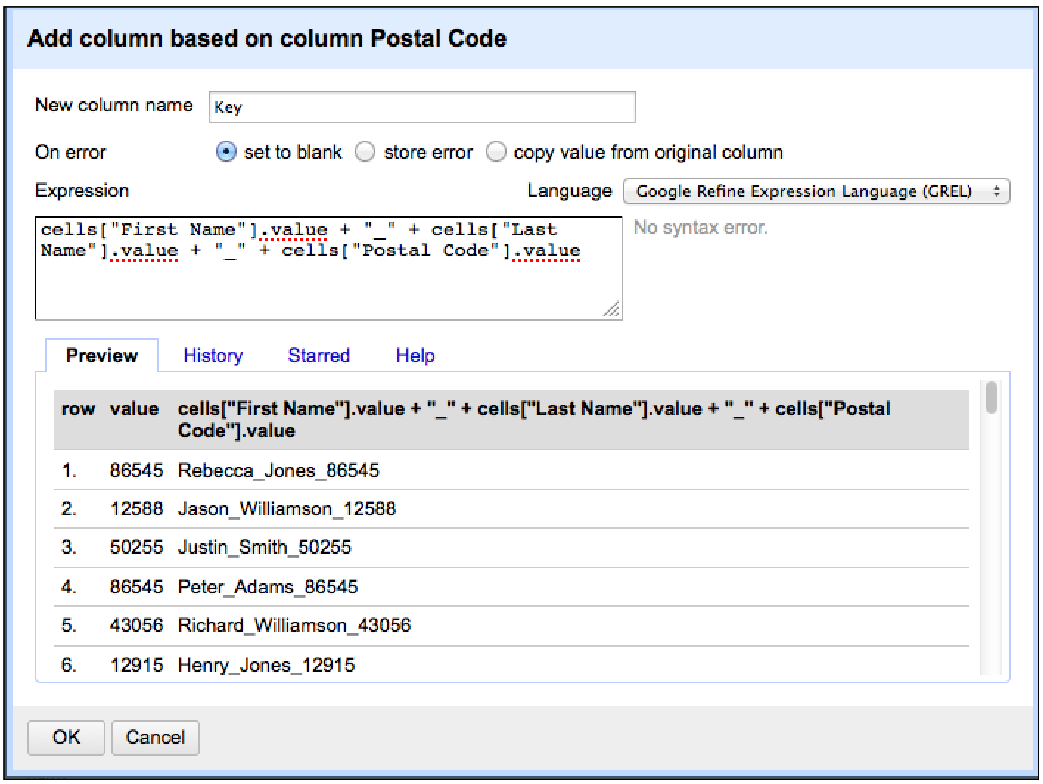
1. Download Google Refine from `http://code.google.com/p/google-refine/wiki/Downloads?tm=2`.
2. Import data from CSV, Excel, or a variety of other files into Google Refine to see how the import mechanism works. It may look something like this:

First Name	Last Name	Postal Code
Peter	Adams	86545
Chris	Adams	33359
Peter	Adams	63034
Greg	Adams	56502
Bruce	Grant	87825
Andrew	Jameson	85378
Charles	Jameson	14812
Walter	Jameson	1540
John	Jameson	60012
Rebecca	Jones	86545
Henry	Jones	12915

You can see that we cannot use **Last Name** values as a unique identifier as it is not unique. If we did an import using the last names, CiviCRM would only update the first matching record it found.

A combination of **First Name** and **Last Name** values is not unique either. There are two people called Peter Adams. But if we combine all three columns then we can get a unique identifier as the value is unique for each record.

3. In the **Postal Code** column, click on the drop-down menu and select **Edit Column** | **Add column based on this column**.
4. In the edit window enter the following:



The data will now look like this:

First Name	Last Name	Postal Code	Key
Peter	Adams	86545	Peter_Adams_86545
Chris	Adams	33359	Chris_Adams_33359
Peter	Adams	63034	Peter_Adams_63034
Greg	Adams	56502	Greg_Adams_56502
Bruce	Grant	87825	Bruce_Grant_87825
Andrew	Jameson	85378	Andrew_Jameson_85378
Charles	Jameson	14812	Charles_Jameson_14812
Walter	Jameson	1540	Walter_Jameson_1540
John	Jameson	60012	John_Jameson_60012
Rebecca	Jones	86545	Rebecca_Jones_86545
Henry	Jones	12915	Henry_Jones_12915

5. You can now use the data in the **Key** column as a unique identifier that CiviCRM can store in the **External Identifier** field.

How it works...

Google Refine has joined together the data in three columns to create a unique identifier for each record in the **Key** column.

For this to work properly, each of the fields in use needs to contain a value. With big datasets, it would be possible that even this key would not be unique. You may also have data such as date of birth to improve the uniqueness.

Importing relationship data

A relationship is a connection between two or more contacts. For example, some contacts may be children of other contacts, who are the parents. Some contacts will be employers of other contacts—employees.

How to do it...

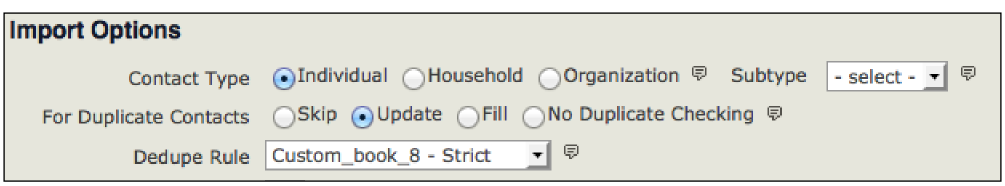
This recipe will show you how to import relationships properly. We will use the built-in employer-employee relationship.

1. Navigate to **Administer** | **Customize Data and Screens** | **Relationship Types** and check that the employer relationship exists and is enabled.
2. Import all the contacts that will be employers. These will generally be organization contacts.
3. Now import all the contacts that will be employees. Make sure you have a unique identifier for each contact.

First Name	Last Name	Postal Code	Key	Organization
Peter	Adams	86545	Peter_Adams_86545	The Center for Sustainability
Chris	Adams	33359	Chris_Adams_33359	Green Earth West
Peter	Adams	63034	Peter_Adams_63034	Parents for the Environment
Greg	Adams	56502	Greg_Adams_56502	Williamsburg Neighborhood Group
Bruce	Grant	87825	Bruce_Grant_87825	The Center for Sustainability
Andrew	Jameson	85378	Andrew_Jameson_85378	Bay Ridge Residents' Group
Charles	Jameson	14812	Charles_Jameson_14812	The Center for Sustainability
Walter	Jameson	1540	Walter_Jameson_1540	Green Earth West
John	Jameson	60012	John_Jameson_60012	Parents for the Environment
Rebecca	Jones	86545	Rebecca_Jones_86545	Borough Park Residents Group

In the preceding data, the **Key** column contains a unique identifier. The **Organization** column contains the name of the employer.

4. Now import the individual contact data again, but this time we will import the relationship.
5. Navigate to **Contacts** | **Import contacts** and set your import.



6. Note we are *updating* our original individual contacts and are using the external identifier as the matching field. In the following screenshot, the options should look like this:

Column Names	Import Data (row 1)	Import Data (row 2)	Matching CiviCRM Field
First Name	Rebecca	Jason	- do not import -
Last Name	Jones	Williamson	- do not import -
Postal Code	86545	12588	- do not import -
Key	Rebecca_Jones_86545	Jason_Williamson_12588	External Identifier (match to contact) *
Organization	Borough Park Residents Group	Organization	Employee of Organization Name (match to contact) *

7. Map the **Organization** field for your import with the relationship type **Employee of**, and use the **Organization Name** as the match.

How it works...

Our contacts are already in the database and we are simply going to update them. So when CiviCRM encounters a matching contact for the **External Identifier** field, we want it to update that contact. That is why **Update** is chosen for the duplicate contacts action.

When it updates, it will create an "Employee of" relationship between the individual contact and the organization that has a name that matches the data in the organization field in the CSV file.

There's more...

Only import one piece of relationship data at a time. Trying to import more than one simply does not work.

One problem with this recipe is that matching an organization name to create the relationship will create errors if organizations share names. For example, there may be branches of the same organization at different addresses, each with its own contact record in CiviCRM. In these situations, you need to get a unique identifier for each organization record set against each record in the individual data and use that as the matching field.

See also

- Find out more about CiviCRM importing at `http://book.civicrm.org/user/current/common-workflows/importing-data/`

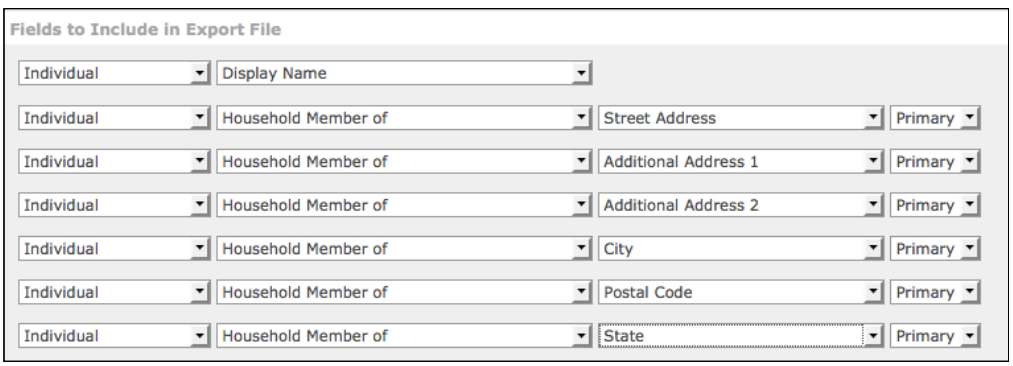
Exporting related data

This recipe shows you how export data that is accessible through a relationship. For example, you might have contacts that are employees of other contacts and you want to send all the employees a mail shot at their workplace address. Or you might want to send all members of the same household a mail shot.

How to do it...

In this recipe we will merge the individual contact data with household contact data to create a CSV file that we can use in a mail merge.

1. Search for the individuals you wish to target and select **Export contacts** from the **actions** drop-down menu at the top of the search results listing.
2. Click on the **Select fields for export** button and then set up your export fields as follows:



3. Save your field mapping and click on **Export**.

How it works...

CiviCRM uses the "Household member of" relationship to grab the address data from the house contact record.

This is a good way of preserving data integrity as there is no need to record an address for the individual contact record.

See also

- Find out more about CiviCRM exporting at `http://book.civicrm.org/user/current/common-workflows/exporting-data/`

Batch updating using profiles

CiviCRM has a useful feature called **batch updating**. For example, you might have a group of contacts where you want to update the value in a custom field. You could go to each contact and update them one at a time, but this would be time consuming. This is where you can use batch updating.

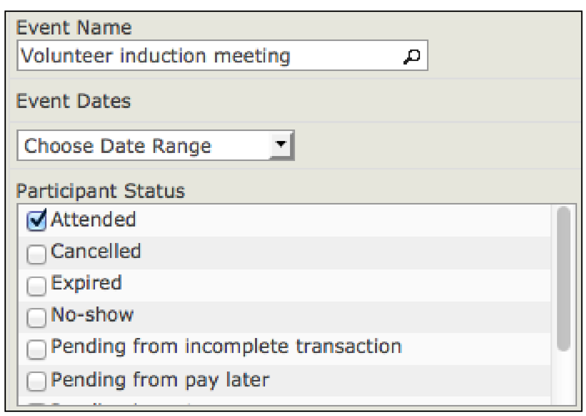
How to do it...

In this recipe we will update some participant information from a recent event.

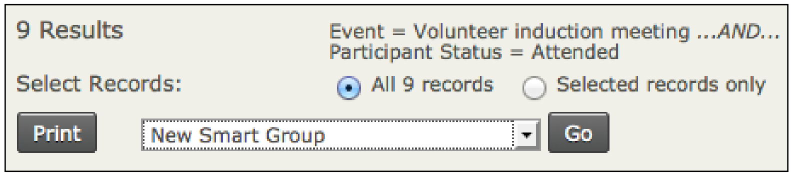
1. Here, prepare a set of custom fields for use with individuals and have them added to a profile that we will use for our batch update. Batch updating works with any contact fields.

   In this example we created a custom field called **Event outcome** with some checkbox options. We then added the field to a new profile called **Volunteer event outcome**.

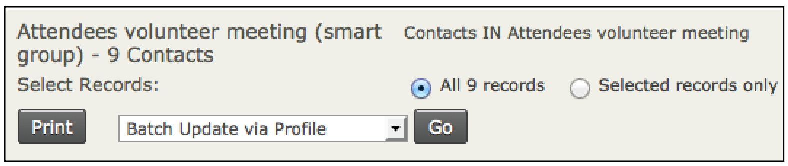
2. Navigate to **Search** | **Find Participants** and search for contacts that attended a volunteer meeting.



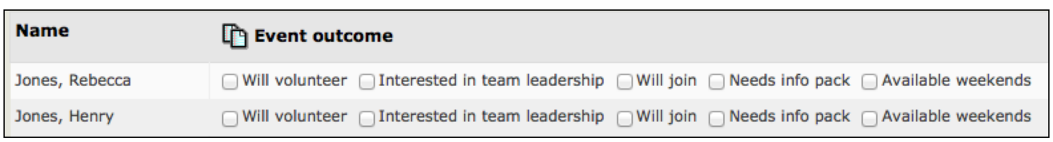
3. Use the **actions** drop-down menu to create a smart group.



4. Navigate to **Contacts** | **Manage groups** and select the smart group you just created.
5. From the **actions** menu at the top of the listing, select **Batch Update via Profile**:



6. Select the profile prepared previously and enter the appropriate options for each contact in the list:



How it works...

CiviCRM loops through each contact and updates each one with the options you have chosen.

Batch updating via profiles has a limit: if there are more than 100 contacts to update then you will need to use an import file instead. You cannot mix contact types in a batch profile. They must all be the same type; for example, individuals or organizations.

There's more...

In this recipe we have been batch updating contacts. CiviCRM contains other batch data entry systems for contributions and memberships.

See also

- The *Creating Custom Fields* recipe in *Chapter 1*, *Setting Up CiviCRM*
- The *Creating and updating a smart group* recipe in this chapter
- Find out more about CiviCRM profiles at `http://book.civicrm.org/user/current/the-user-interface/profiles/`

3

Using the Power of Profiles

In this chapter we will cover:

- Speeding up data entry
- Using URLs to change profile displays
- Creating a membership directory
- Controlling the search result columns using profiles
- Using the Profile Pages and Listings setting to improve usability
- Setting up reCAPTCHA for user profiles

Introduction

You will find yourself using profiles a lot in CiviCRM. A profile is a custom-made collection of contact fields made for a specific purpose. Profiles can be used throughout CiviCRM to display and collect information.

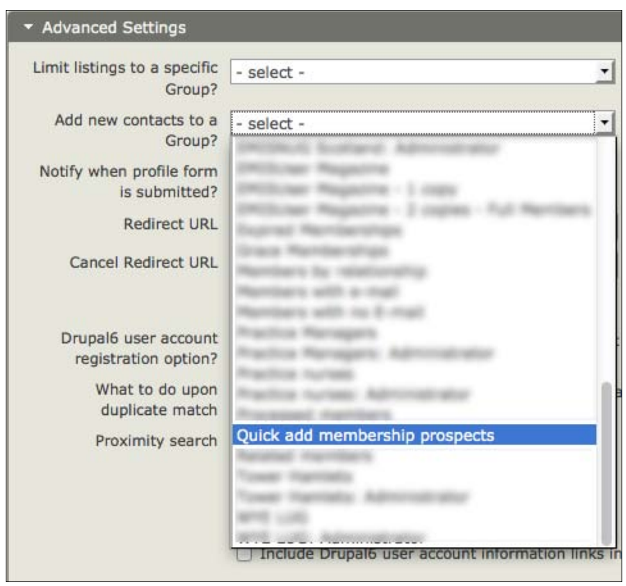
Speeding up data entry

You can use profiles to speed up data entry and to control access to contact data. For example, you might have a set of volunteers who are responsible for maintaining specific data about contacts. You will want to give them access to the data they require but not other sensitive data.

How to do it...

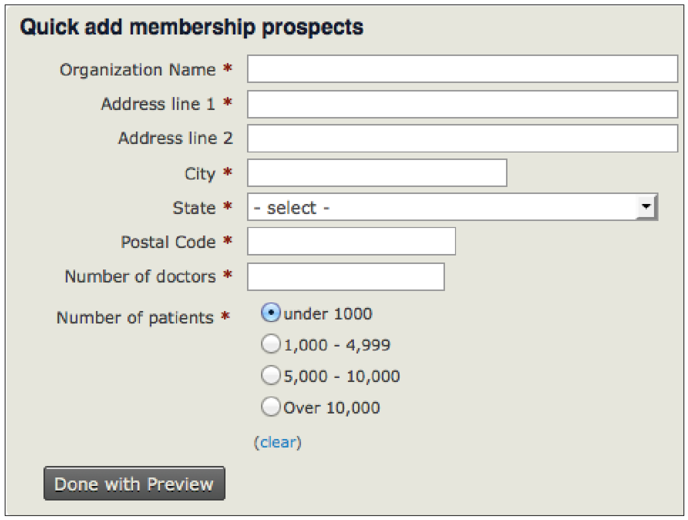
In this recipe we will use a profile to enter some data for potential new members for our organization.

1. Create a group that will store all the new contacts you are going to add.
2. Navigate to **Contacts** | **New Group** and create a new group. Call it `Quick add membership prospects`.
3. Navigate to **Administer** | **Customize Data and Screens** | **Profiles** and create a new profile. Give the profile a sensible name that reflects its purpose. Call it `Quick add member prospects`.
4. In the **Used For** field select **Standalone Form or Directory**.
5. Add in some guidance to the top and the bottom of the profile to help the people who are going to be doing the data entry.
6. Click on the **Advanced** field set at the foot of the profile entry screen.
7. In **Add Contact to a Group** select the group we created as shown in the following screenshot:



8. Save the profile and begin to add fields to it.

9. For this recipe we are going to add some fields for an **Organization** contact type. We will also include a couple of custom fields. See the *Creating custom fields* recipe in *Chapter 1, Setting Up CiviCRM*. From the **Organization** option list, choose **Organization Name**.



10. From the **Contact** option list, choose a few address fields for the organization.
11. Here we have added in a couple of custom fields, **Number of doctors** and **Number of patients**.
12. For each field added, tick the **Required** checkbox for data that *must* be entered; the organization name is a must, as well as the **City**, **State**, and **Postal Code** fields.
13. Click on the **Preview All Fields** button to check the fields and play around with them to get the order right.
14. Now click on the **Use Create mode** button to create a contact entry.
15. Copy the create mode URL from your browser navigation bar (`civicrm/profile/create?gid=N&reset=1`, where `N` is the ID of your profile). Use it to create a menu entry in your CiviCRM navigation menu.

See also

- The *Adding items to the CiviCRM navigation menu* recipe in *Chapter 1, Setting Up CiviCRM*
- Find more about CiviCRM profiles at `http://book.civicrm.org/user/current/the-user-interface/profiles/`

Using URLs to change profile displays

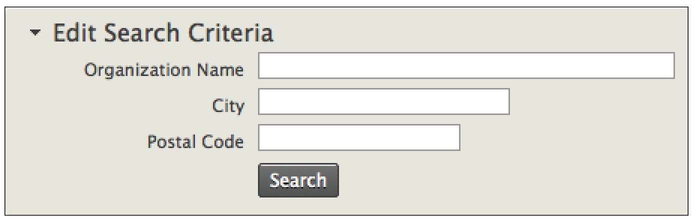
In the previous recipe we used a URL to access the profile we created, that is, `civicrm/profile/create?gid=N&reset=1`, where `N` was the ID of our profile. This URL made the profile appear as a form where we could create a contact.

How to do it...

By using different URLs you can create search forms and listings:

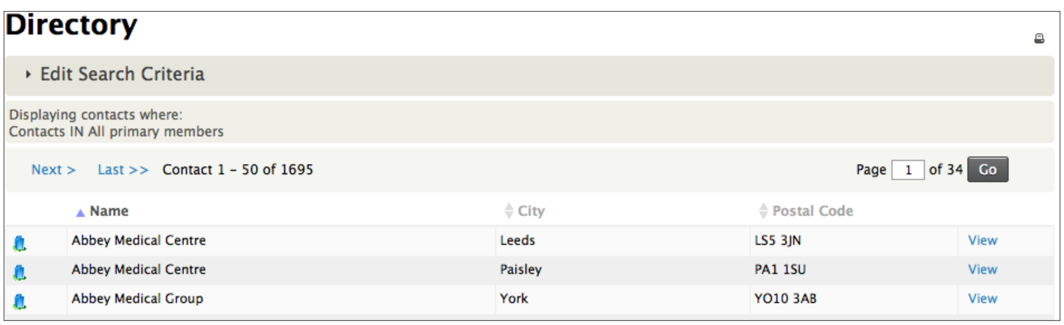
1. Create a search form with this URL: `civicrm/profile?gid=N&reset=1`.

   This provides a search form. When you create your profile, you must make sure that field visibility is set to **Public Pages** or **Public Pages and Listings**.



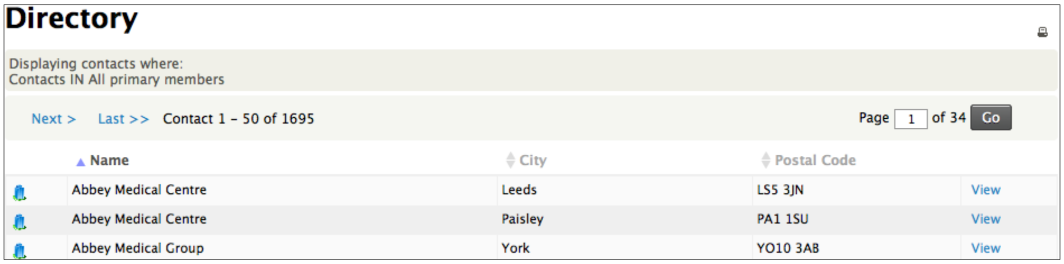
2. Create a listing with an editable search with this URL: `civicrm/profile?gid=N&reset=1&force=1`.

   This provides a listing with editable search criteria at the top. When creating the profile for this, under the **Advanced** settings, you can restrict the contacts listed using **Limit listings to a specific Group**.



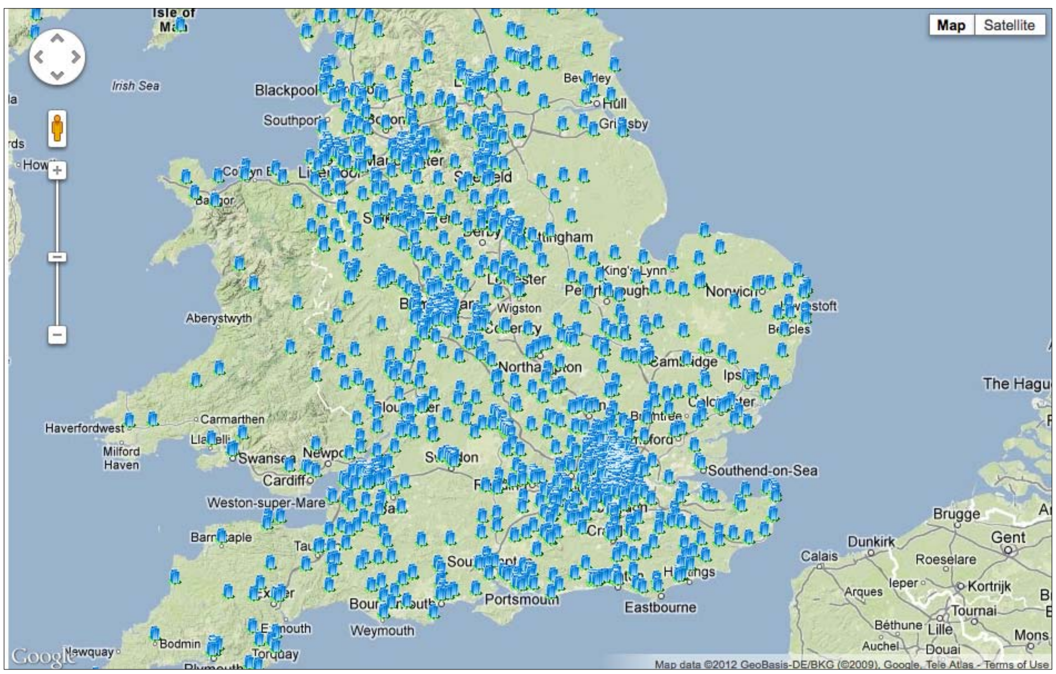
3. Create a listing without a search with this URL: `civicrm/profile?gid=17&reset=1&force=1&search=0`.

   This provides a listing with no editable search criteria.



4. Create a contact map with this URL: `civicrm/profile/map?map=1&gid=N&reset=1`.

   This provides a map of contacts. You must have geocoded your contacts and you must have **Enable mapping for this profile?** selected in the **Advanced** settings for the profile.



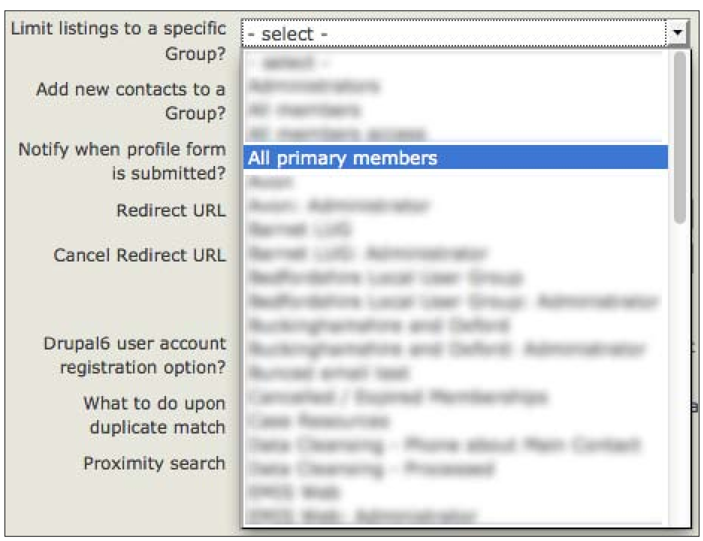
Creating a membership directory

Profiles can be used to create listings of contacts. You can display these lists publicly on your website so it's a quick way to show a membership directory.

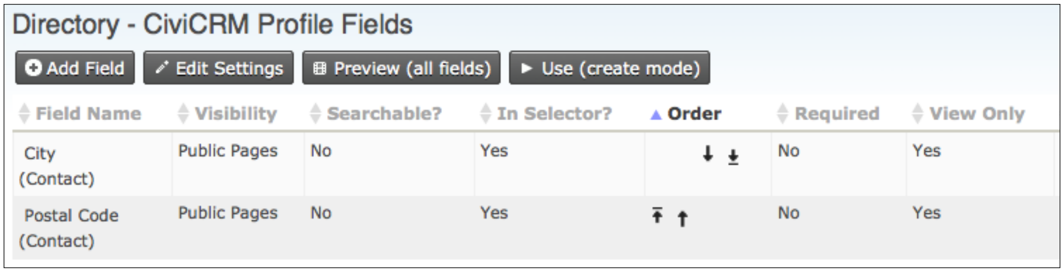
How to do it...

Setting up a membership directory is easy. We just create a smart group to list our contacts and then create a profile that lists contacts within the group.

1. Set up a group that will contain the contacts that you want to list. It's a good idea to make this a **smart group**. This means any contact that matches the smart group criteria will automatically be added to your group and thus to your listing. When you create your smart group, make sure that **Visibility** is set to **public pages**.
2. Navigate to **Administer** | **Customize Data and Screens** | **Profiles** and add a new profile that will contain the fields you wish to display on your directory.
3. For the **Used for:** field, select **Standalone Form or Directory**.
4. Navigate to the **Advanced** settings. In the **Limit listings to a specific Group?** field, select the smart group that you created previously as shown in the following screenshot:



5. Click on **Save** and begin adding fields to your profile. For this recipe we will list the organizations that are members with their city and postal code:



   The fields have been set for public pages. Note that you do not have to include the name of the organization itself. CiviCRM puts the contact display name in the search result set by default.

6. Create a profile URL, such as `civicrm/profile?gid=17&reset=1&force=1&search=0`, to display the directory.

   Here `GID` is the ID of the profile we created. This will display the directory as a list.

There's more...

You might want to control permissions for profiles. For example, you may only want members to be able to see the member directory.

See also

- The *Creating and updating a smart group* recipe in *Chapter 2*, *Organizing Data Efficiently*
- *Chapter 4*, *Controlling Permissions*

Controlling the search result columns using profiles

CiviCRM uses a default template to display search results. There are columns for the contact name, addressing, and other contact data. What if you want it to display other data instead, such as custom data fields you have created? We can use CiviCRM profiles to achieve this.

How to do it...

In this recipe we couple the power of CiviCRM profiles with Advanced Search to create customized search result pages.

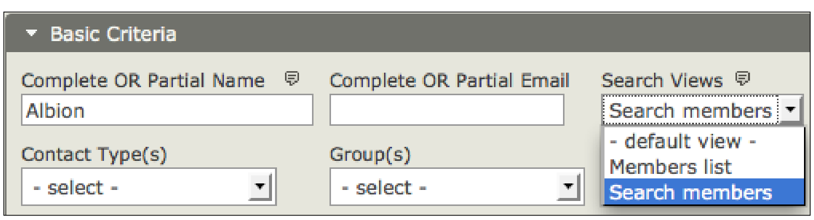
1. Navigate to **Administer** | **Customize Data and Screens** | **Profiles** and add a new profile.
2. Select **Search Views** in the **Used For** field.
3. Save the profile and then add in the fields you want to show in your search results. You can only add in fields available to Contacts. In the following example we added in some custom data fields for organization contacts.
4. Make sure that each field set for **Public Pages** is included in the results table. You only need to set them to **View Only** for the **Search Views** profiles:

Field Name	Visibility	Searchable?	In Selector?	Order	Required	View Only
City (Contact)	Public Pages	No	Yes	↓ ⤓	No	Yes
Number of doctors (Organization)	Public Pages and Listings	No	Yes	⤒ ↑ ↓ ⤓	No	Yes
Number of patients (Organization)	Public Pages and Listings	No	Yes	⤒ ↑	No	Yes

5. Navigate to **Search** | **Advanced Search**. You can now perform a search and choose the profile you created to display the search results.

How it works...

Here we use the **Search members** profile to display search results:



The search result set looks like the following screenshot:

Name	City	Number of doctors	Number of patients
Albion			
Albion House	Dudley	6	Over 10,000
Albion Street Group Practice	London	0	under 1000
The Albion Road Surgery	Broadstairs	1	1,000 - 4,999

CiviCRM only displays the fields we added to the profile.

Using the Profile Pages and Listings setting to improve usability

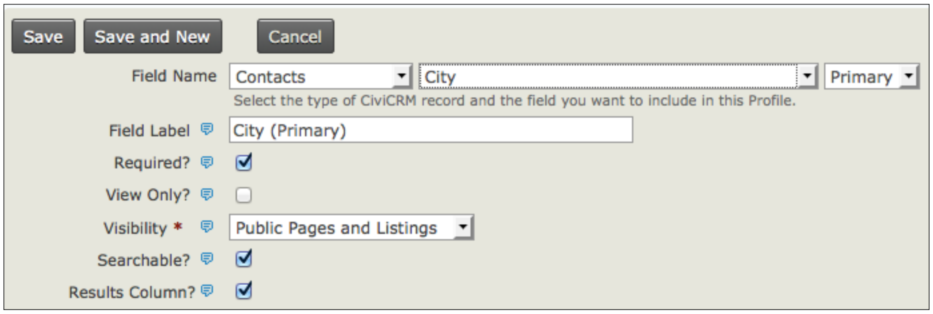
CiviCRM has great search facilities. By using Advanced Search or Search Builder you can find pretty much anything within your database. You can use profiles to provide useful links to list your data without having to go back and forth between searches.

How to do it...

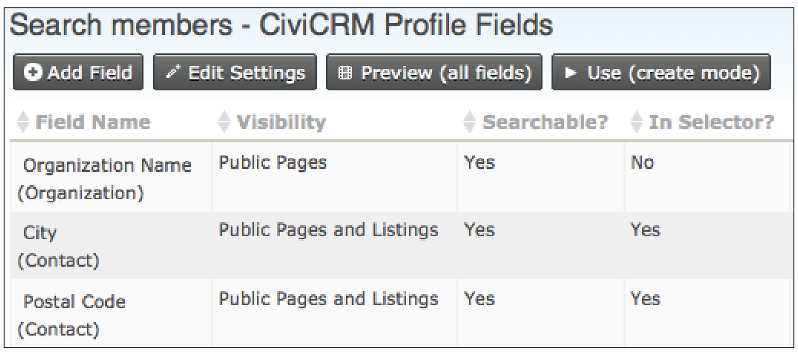
We will use a setting in profiles that creates more search links to explore data further.

1. Navigate to **Administer** | **Customize Data and Screens** | **Profiles** and add a new profile.
2. Save your profile and add the fields that interest you.
3. In the **Used For** field, select **Standalone Form or Directory**.
4. When you add fields to your profile make sure that the fields are set to:
    - **Public Pages and Listings**
    - **Show in Results Column**

- **Searchable**



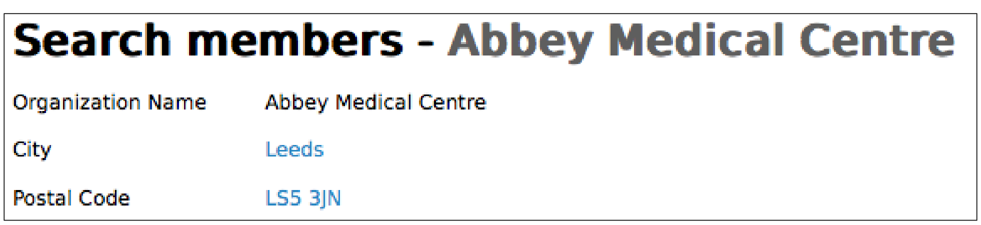
In the following example we have added three fields:



5. Once you have done this, use the search URL `civicrm/profile?gid=N&reset=1`, where `N` is the ID of your profile, and perform a search. From the result set, you will see there is a column containing the **View** links.
6. Click on **View** on any of the contacts.

How it works...

You can now see there are links to search for the fields you set to **Public Pages and Listings**. In this case, they are **City** and **Postal Code**:



These allow you to explore the result set further without having to navigate back and change your search criteria.

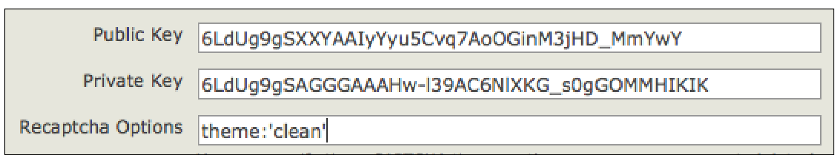
Setting up reCAPTCHA for user profiles

reCAPTCHA is a useful tool to reduce spam submissions to your site.

How to do it...

You will need to register your website with Google and obtain **public** and **private** keys for reCAPTCHA.

1. Navigate to **Administer** | **System Settings** | **Undelete, Logging and ReCaptcha**.
2. Get your public and private reCAPTCHA keys from `http://www.google.com/recaptcha`.
3. Enter them into the settings fields:



4. You can also add in theme settings.
5. You can then enable reCAPTCHA by editing the **Advanced** settings for any profile where you wish to use it.

See also

- Find more about reCAPTCHA customization at `https://developers.google.com/recaptcha/docs/customization`

4

Controlling Permissions

In this chapter, we will cover the following recipes:

- Integrating profiles into Drupal user accounts
- Restricting access to custom fields
- Using CRM profile permissions correctly
- Creating permissions for administrators
- Managing event registrations using CiviCRM Access Control Lists

Introduction

CiviCRM provides two levels of permission control. At the CMS level, for example, Drupal, there are global CiviCRM permissions that can be applied to your users. If you disable these permissions, you can use CiviCRM's **Access Control Lists** (**ACLs**) to achieve fine-grained permissions for viewing and editing your database. These recipes explore how to get the best out of both permissioning systems, using Drupal as a CMS.

Integrating profiles into Drupal user accounts

It's quite easy to add a CiviCRM profile to Drupal's user account page. In the **Settings** page for your profile, you can choose to have the profile used for **View** and **Edit Drupal User Account**. This means that when the user logs in to Drupal and visits their user page, the CiviCRM profile is exposed. This is a great way to allow users to edit their CiviCRM data without giving full access to CiviCRM. The trouble comes when you want to make different profiles available to different sorts of users. For example, in CiviCRM, you could have two different sorts of individual contacts, students and teachers. You would not want to expose the teacher profile to the student and vice–versa.

How to do it...

In this recipe, we are going to have two individual contact subtypes, namely Boat Skippers and Boat Crew.

Boat Skippers will have custom fields that hold information about what sort of boats they have. **Boat Crews** will have custom fields about what sailing skills they have. When they log in to our website, they will have the correct profiles exposed on their user page.

1. Make sure that the **CiviGroup Roles Sync** module is enabled.
2. Navigate to **Administer** | **Customize Data and Screens** | **Contact Types**, and create an **Individual contact** subtype for **Skipper** and for **Crew**.
3. Create custom datasets for each content type. For **Skippers**, you can create **Boat Name**, **Boat Type**, and **Sail Number** fields. For **Crew**, you can create a competence multiple choice field and enter a few choices.
4. Navigate to **Administer** | **Customize Data and Screens** | **Profiles**, and add a new profile.
5. Name the profile `Skipper`.
6. In the **Used For** field, select **View/Edit Drupal User Account**.
7. Save the profile and add in custom fields for **Skippers**.
8. Create another profile for **Crew**. Name it `Crew`.
9. In the **Used For** field, select **View/Edit Drupal User Account**.
10. Save the profile and add in custom fields for **Crew**.

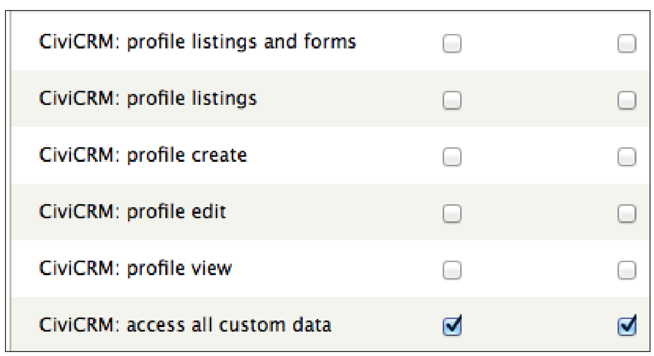
Profile Title	Type	ID	Used For
Crew	Crew	16	Standalone Form or Directory, View/Edit Drupal User Account
Skipper	Skipper	15	Standalone Form or Directory, View/Edit Drupal User Account

You now have two profiles, with custom fields added for the two contact types.

11. Navigate to a **Drupal User Page**, and edit the account. You will now see both profiles available as tabs on the user profile.
12. In Drupal, navigate to `admin/people/permissions/roles`, and create a **Skipper** role and a **Crew** role. Allocate users to each role to test the recipe.
13. In CiviCRM, navigate to **Contacts** | **New Group**, and create a group for **Skippers** and a **Group for Crew**. Make sure that the checkbox for **Access Control** is checked. Make sure that visibility is set to **User** and **User Admin** only.

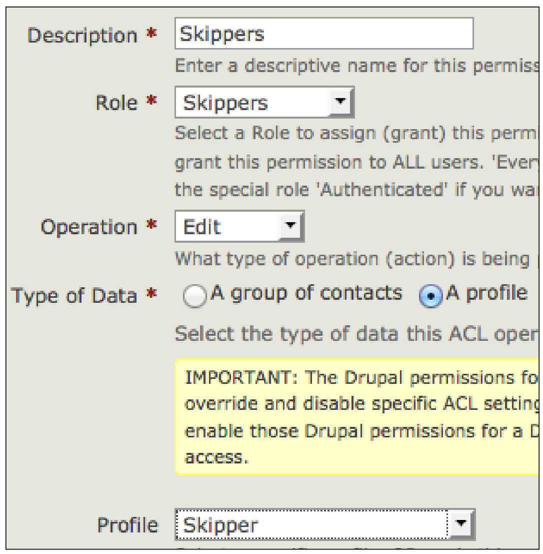
14. In Drupal, navigate to `admin/config/civicrm/civicrm_group_roles`. You want to match the Drupal roles to the CiviCRM groups you set up:

15. Select the **Manually Synchronize** tab and synchronize the contacts. Check the CiviCRM groups to see if the Drupal users have been added. Now, when a Drupal user is allocated to the role **Skipper**, they are automatically added to the CiviCRM group. The same goes for the crew. You can now use the CiviCRM group to control access to the CiviCRM profiles you created.
16. In Drupal, navigate to `admin/people/permissions`. Drupal permissions overrule any permissions that we set up in CiviCRM. CiviCRM assigns a default set of Drupal permissions on installation, so we need to remove these first.



17. Remove the permissions to create, edit, and review profiles. This means the profiles you created will no longer appear on the user account edit form. We left the CiviCRM access to all custom data permissions checked for the **Anonymous** and **Authenticated** roles. In your own situation, you can also remove these global permissions to have a further layer of control. We can now refine and reinstate these permissions in CiviCRM.

18. Navigate to **Administer** | **Users and permissions** | **Permissions (Access Control)**, and add an ACL role. This is not the same as a Drupal role. An ACL role acts as a container to hold groups of contacts.
19. Create an ACL role for **Skippers** and an ACL role for **Crew**.
20. Now, click on **Assign Users to ACL Roles**. For each ACL role, we can add one or more CiviCRM groups of users. These groups must have **Access Control** set as the **Group Type**.
21. Add the **Skippers** group to the **Skippers ACL** role, and add the **Crew Group** to the **Crew ACL** role. In your situation, you can add extra groups to these roles. This makes CiviCRM permissions very flexible.
22. On the home screen for **Permissions**, click on the **Manage ACLs** link and then **Add ACL**.



23. Make the ACL description `Skippers`.
24. Add the ACL role `Skipper` that you created in step 16.
25. Set **Operation** as `Edit`.
26. Make the type of data **A profile**. This will now automatically provide you with a drop-down list of CiviCRM profiles.
27. Set **Profile** as **Skipper**.
28. Repeat the procedure for the **Crew** profile.

How it works

If you now edit a Drupal user account, for each role, you will see the tab for **Skippers** is only available for users in the **Skipper** role and similarly for the crew role.

The CiviCRM **Civi Group Roles Sync** module synchronizes Drupal roles to CiviCRM Groups. So, when a user is given the **Skipper** role in Drupal, they are added to the CiviCRM Group called **Skipper**.

The CiviCRM **Skipper** group is set to be used for access control. We created an ACL role called **Skippers** and added all the contacts in the CiviCRM skipper group to it. We then gave the ACL role permissions to edit the skipper profile.

> Do not use the Drupal Masquerade module to test CiviCRM profile visibility. Masquerade only takes into account Drupal permissions.

See also

- The *Adding custom fields* recipe in *Chapter 1*, *Setting Up CiviCRM*
- The *Adding contact types* recipe in *Chapter 2*, *Organizing Data Efficiently*

Restricting access to custom fields

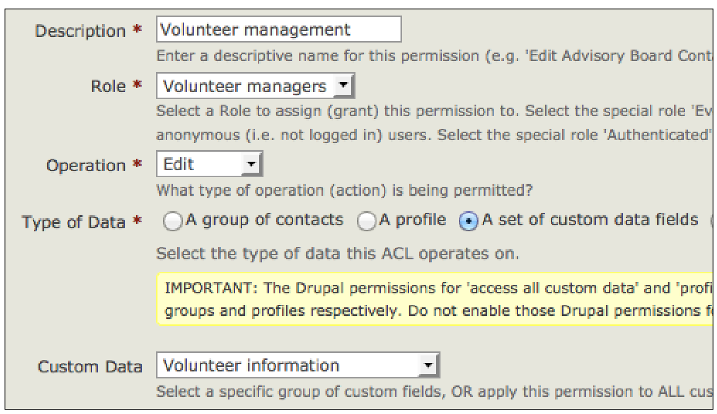
There may be situations where you will want to restrict sensitive confidential data to certain roles. For example, in a drug rehabilitation center it would be critical to ensure that any client confidential data is only viewable and editable by client caseworkers and other authorized people.

How to do it...

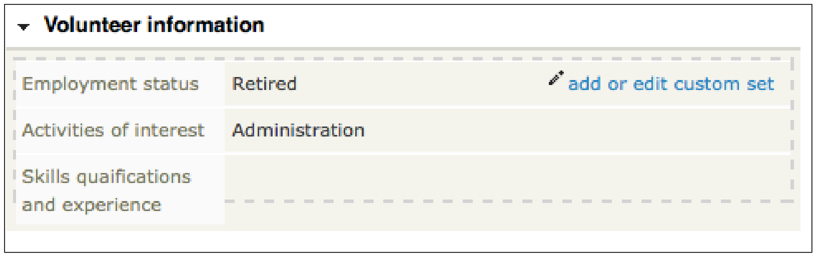
In this recipe, we will have a group of volunteer managers who will be able to edit custom fields for volunteer information. We will remove any overriding CMS permissions and use CiviCRM ACLs to provide permissions for custom data.

1. In Drupal, navigate to **People** | **Permissions** | **Roles**. Create a role called `CiviCRM Admin`.
2. In Drupal, navigate to **People** | **Permissions**, and remove the **Access all custom data** permission for all roles. Removing the **Access all custom data** permission is not without it's difficulties. It means that every time you add a new custom field, you will have to add permissions using CiviCRM, rather than globally in your CMS. This is why data planning is such a critical step when planning your CiviCRM deployment.

3. Add the permissions **Access CiviCRM**, **View All Contacts**, and **Edit all contacts** to the **CiviCRM Admin** role.
4. Add some Drupal users to the role **CiviCRM Admin** to test the recipe.
5. In CiviCRM, navigate to **Administer** | **Customize Data and Screens**, and add a custom field set. In this case, we have added a custom field set called **Volunteer Information** and some custom fields to hold volunteering data.
6. Navigate to **Contacts** | **New Group**, and create a **Volunteer Admin** group, making sure that the **Access Control** checkbox is ticked.
7. Navigate to **Administer** | **Users and permissions** | **Permissions (Access Control)**, and add an ACL role called `Volunteer Managers`.
8. From the **Permissions** home screen, click on **Assign Users to Roles**.
9. Add the **CiviCRM Volunteer Admin** group to the CiviCRM ACL role `Volunteer Managers`.
10. On the **Permissions** home screen, click on the **Manage ACLs** link and **Add ACL**.



11. Make the ACL description **Volunteer Management**.
12. Set **Role** as **Volunteer Managers**.
13. Make the operation **Edit**.
14. Make the type of data **A set of custom data fields**.
15. Make the custom data **Volunteer Information**.
16. Add some contacts to the **CiviCRM Volunteer Manager** group. They must also have the Drupal role **CiviCRM Admin**.
17. Log in as a user that has the Drupal CiviCRM admin role and is in the CiviCRM group **Volunteer Manager**. You will see that the **Volunteer information** custom field set is editable, where it appears on the contact summary screen:



How it works

The Drupal permissions system is used to check whether the user has the correct admin rights for CiviCRM. We removed the permissions to view all custom fields. So, we then used CiviCRM's permissions to see if the user can access the **Volunteer information** custom field set.

See also

- The *Adding custom fields* recipe in *Chapter 1*, *Setting Up CiviCRM*

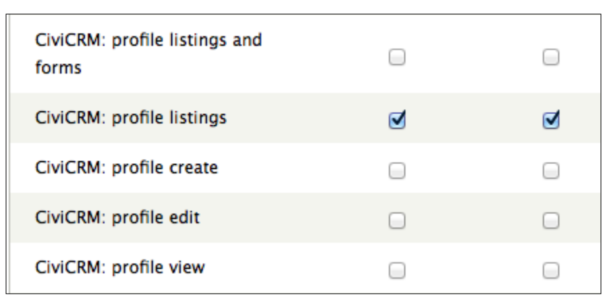
Using CRM profile permissions correctly

When CiviCRM is installed for the first time, it provides a standard set of permissions with the CMS. The biggest source of security problems with CiviCRM is permissioning, so it is critical that you visit your CMS permissions and check them.

How to do it...

Setting permissions is an important part of the CiviCRM planning process. From the outset, you should understand what each permission does so that you can configure your installation properly.

1. Navigate in Drupal to **People** | **Permissions**, and scroll down to **CiviCRM Profile permissions**.



CiviCRM: profiles listings is used in situations where you provide listings of contacts on your website, for example, a membership directory. If you want everyone to be able to search for and see the lists of contacts on your site, you can give the anonymous user and the authenticated user this permission. Note that these permissions are global. If you want only some lists to be available, then you can remove this permission and control the access by using CiviCRM access control lists.

CiviCRM grants this permission to anonymous and authenticated users by default, and one of the most common errors in a CiviCRM site is that sensitive data is exposed to anonymous users because this permission has not been removed.

CiviCRM: profile view is used in situations where you want to display profile information on a page. For example, you may have a site where you want to allow users to view other users' Drupal user pages, which also contain CiviCRM data. You can expose it using this permission.

CiviCRM: profile edit is used when you want users to be able to search for and edit profile fields that are not a part of an event registration form. A typical example might be a short survey. Anonymous users can be given this permission but cannot actually edit the information if it were just presented on a screen. They would need to navigate to the survey by using a URL that contains a checksum token, which gives them a unique URL, so they can edit this information.

CiviCRM: profile create is used when you want users to be able to complete a profile as part of an event registration.

CiviCRM: profile listings and forms is a powerful permission that you should only assign with care. It gives global permissions to all the online forms and listings.

See also

- `http://book.civicrm.org/user/current/initial-set-up/access-control/`

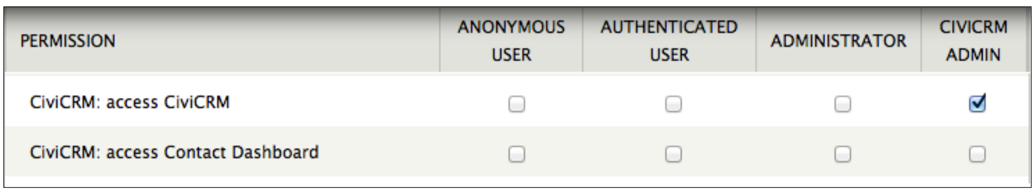
Creating permissions for administrators

If you look at the CMS permissions for CiviCRM, there are 60 or so permissions that you can control, depending upon what CiviCRM components you have enabled. If you are new to CiviCRM, it can be quite a daunting proposition to allocate these permissions correctly. For example, you might want to create a role in your CMS for CiviCRM Admin and let those users the admin CiviCRM. What permissions do you give them in the CMS?

How to do it...

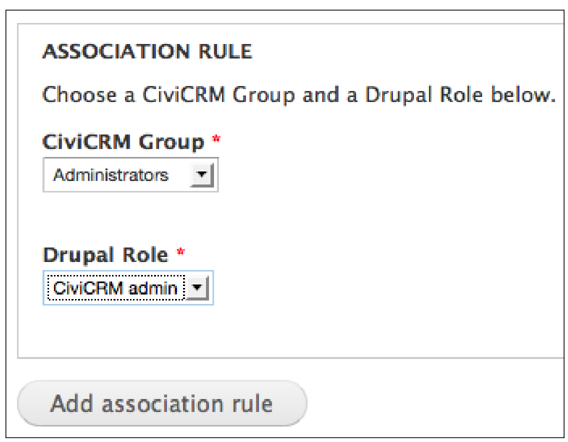
CiviCRM has a preconfigured ACL role called **Administrators** linked to a CiviCRM group called **Administrators**. It is already set up with the main permissions to administer CiviCRM. All we need to do is to hook our users up to it and use it for our admin permissions rather than using CMS permissions.

1. In Drupal, create a role called `CiviCRM Admin`.
2. In your Drupal, disable the **CiviCRM: access CiviCRM** permission for everyone except the **CIVICRM ADMIN** role.



PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	ADMINISTRATOR	CIVICRM ADMIN
CiviCRM: access CiviCRM	☐	☐	☐	☑
CiviCRM: access Contact Dashboard	☐	☐	☐	☐

3. In Drupal, ensure that the **CiviCRM Group Roles Sync** module is enabled.
4. In Drupal, navigate to `admin/config/civicrm/civicrm_group_roles`, and set up a new association rule:



5. Match the CiviCRM **Administrators** group to the Drupal **CiviCRM admin** role.

How it works...

The **Administrator** group in CiviCRM is already set up with CiviCRM access control permissions to administer CiviCRM. Anyone who has the Drupal CiviCRM admin role now has CiviCRM administrative rights to the site.

There's more...

You can set up extra roles in Drupal to refine your CiviCRM administration. For example, you can set up an **Events Administration** role in Drupal and synchronize it with an events manager group in CiviCRM. You can then give the members of that group CiviCRM ACL permissions to manage CiviCRM events.

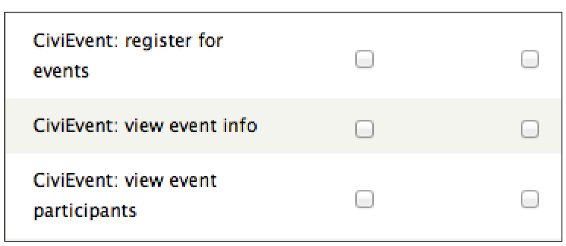
Managing event registrations using CiviCRM Access Control Lists

This recipe is used for situations where you want to restrict access to event registrations to a group of contacts. For example you might be holding an event that is exclusive to members.

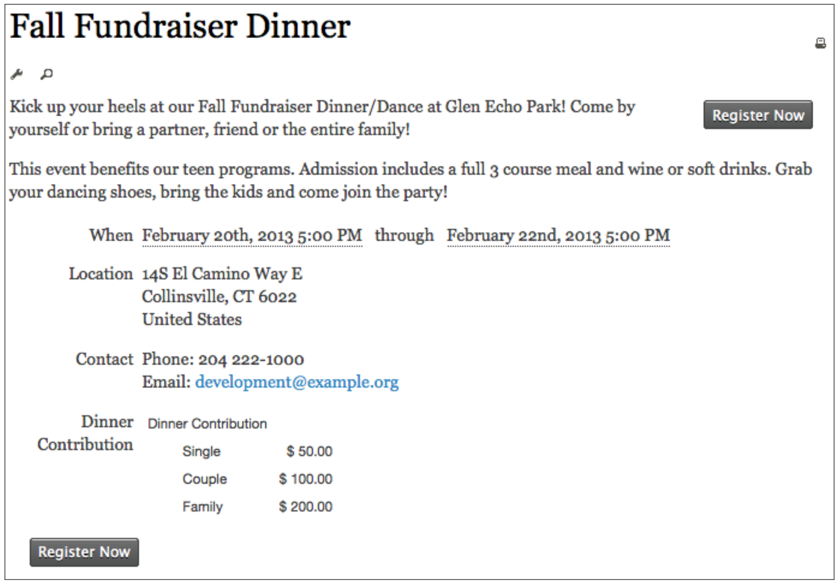
How to do it...

Here, we remove the global permissions set by the CMS—in this case Drupal—and replace them with CiviCRM access control list permissions.

1. Navigate to **Drupal** | **Administer** | **People** | **Permissions**, and look at the permissions for **Events**. You need to remove global permissions for **CiviEvent: Register for Events**.



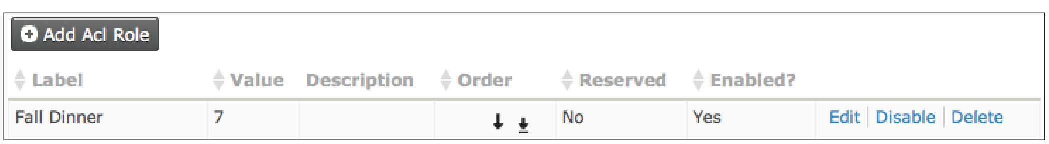
2. Navigate to **Administer** | **Events** | **New Event**, and create an event.



Note that events can include profile fields and a contribution. Both these can also be controlled by permissioning.

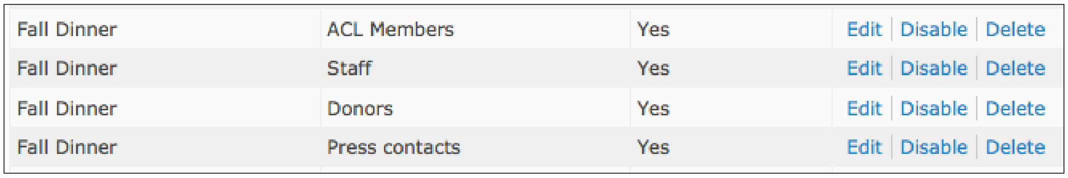
3. Navigate to **Search** | **Advanced Search**, and search for a group of contacts that are to be targeted for invitations. Save these contacts to a group. For example, you can do a search for current members of your organization and add them into a group called **Members**. You might want to create other groups such as **Staff**, **Press Contacts**, and **Donors**. Each group that you create needs to have **Access Control** checked. Note also that you cannot use smart groups for access control.

4. Navigate to **Administer** | **Users and Permissions** |**ACL(Access Control)**, and create a new ACL role called `Fall Dinner`.



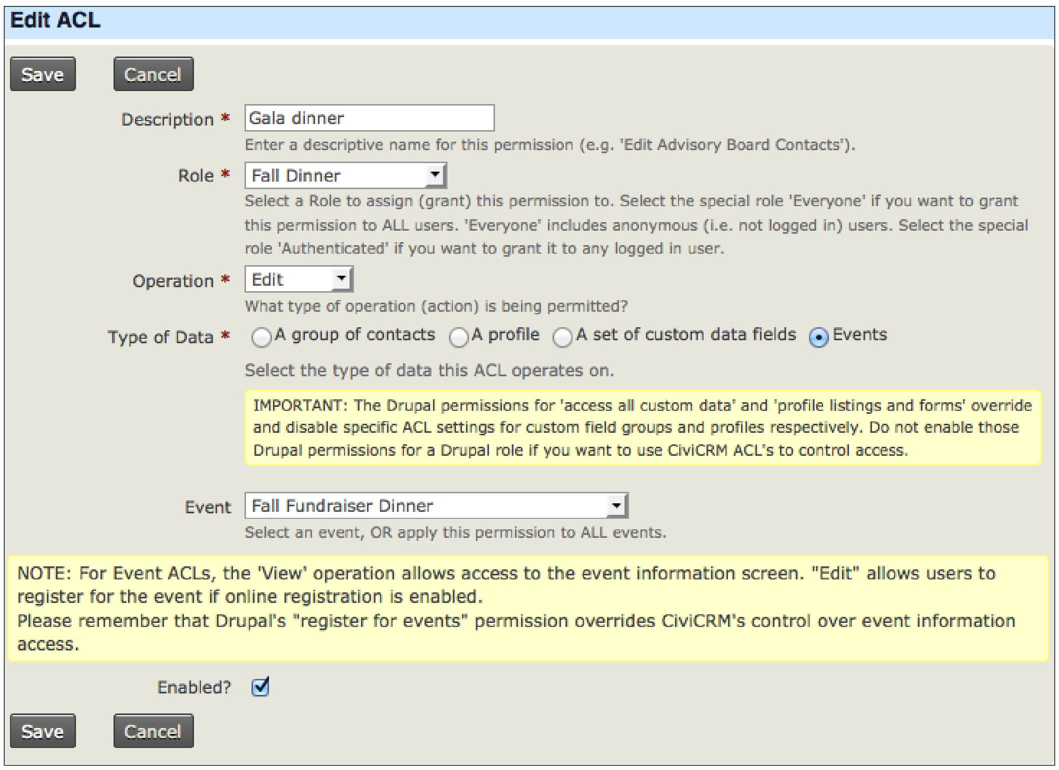
5. Navigate to **Administer** | **Users and Permissions | ACL (Access Control)**, and click on **Assign Users to CiviCRM ACL Roles**.

6. Add the CiviCRM groups you created to the **Fall Dinner** role:



Fall Dinner	ACL Members	Yes	Edit Disable Delete
Fall Dinner	Staff	Yes	Edit Disable Delete
Fall Dinner	Donors	Yes	Edit Disable Delete
Fall Dinner	Press contacts	Yes	Edit Disable Delete

Now, we can apply permissions to register for our event to the **Fall Dinner** role:



7. Make the description `Gala dinner`.
8. Make the role `Fall Dinner`.
9. Make the operation **Edit**.
10. Set **Type of Data** to `Events`.
11. Make the event the **Fall Fundraiser Dinner** event.

How it works

By removing the CiviCRM global permissions in the CMS, we are able to fine-tune permissions to register for the event by using CiviCRM's ACL permissioning.

5
Managing Communications

In this chapter, we will cover:

- Setting up a bounced e-mail account using Gmail
- Creating mail templates for CiviMail
- Creating mail templates for CiviMail in Drupal
- Using tokens in templates
- Creating custom date tokens
- Scheduling CiviMail
- Throttling mailings to comply with hosting restrictions
- Creating newsletter subscription services using profiles
- Creating newsletter subscriptions using URLs
- Creating a standalone newsletter subscription form
- Getting a CiviMail report
- Mailing attachments in e-mails and CiviMail
- Allowing users to update information without logging in

Introduction

CiviCRM comes with very sophisticated mail services that will form the basis of how you communicate with your contacts. It is vital that you provide consistent, personalized, and high quality communications—and this chapter shows you how.

Setting up a bounced e-mail account using Gmail

CiviCRM comes with two flavors of mail. You can choose to send mail to your contacts using the **Send email** action available for every set of contact search results. Or for a lot of contacts you can choose to use CiviMail, which provides additional tracking, notifications for bounced e-mail, and subscription features not available in ordinary mail. If you plan to use CiviMail to communicate with your contacts, then having a **bounced e-mail account** is essential. Contact e-mail addresses are constantly changing and become out of date very quickly. A bounced e-mail account is required so that CiviMail can disable contact e-mails that reply with bounced e-mail messages. Getting this set up can be quite frustrating, but this recipe works every time.

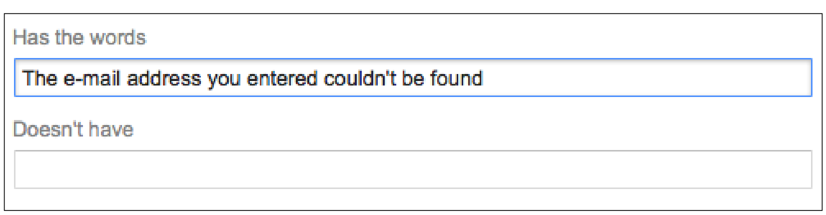
Getting ready...

Ensure that CiviMail is enabled in CiviCRM components, and ensure that you have set up cron to manage **Fetch Bounces**.

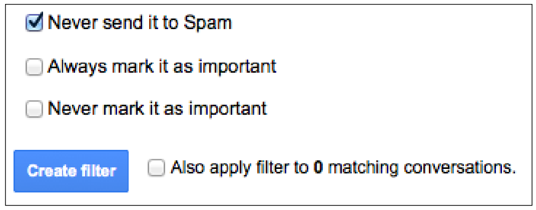
How to do it...

We will set up an e-mail account to capture bounced e-mails and then configure CiviCRM to check the account periodically.

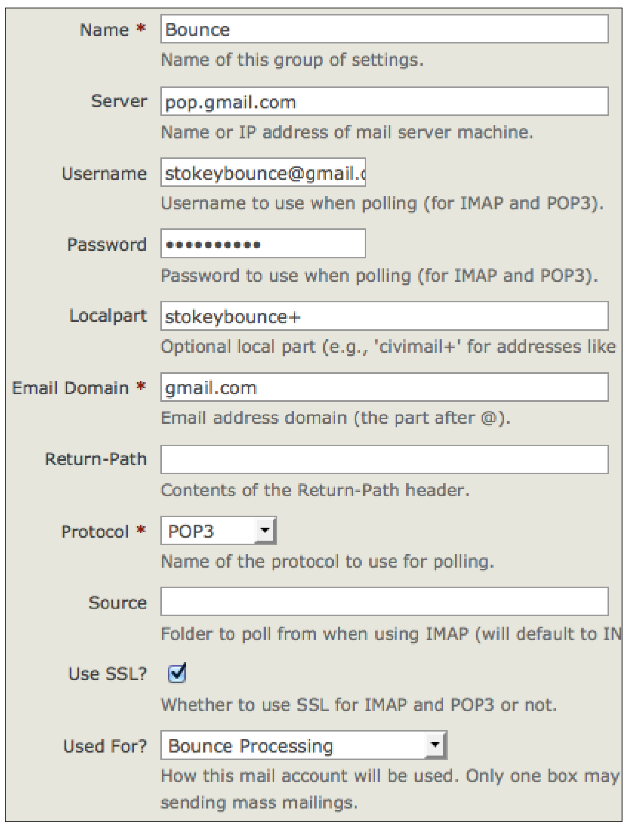
1. Set up an account in Gmail to manage the bounced e-mails. Give it a name such as `mysite.bounce@gmail.com`.
2. Log in to the Gmail account and navigate to the settings page.
3. Click on the **Filter** tab. You do not want Gmail to filter any bounced messages into the **Spam** folder, otherwise CiviCRM will not have access to them. Create filters for phrases such as **The e-mail address you entered couldn't be found**, or from accounts containing `mailer-daemon` or `postmaster`.



4. For each filter you create, ensure that the **Never send it to spam** checkbox is checked so that messages are never sent to the **Spam** folder.



5. Navigate to **Administer** | **CiviMail** | **Mail accounts**.
6. Add a new account and complete the details as shown in the following screenshot, entering the information from the Gmail account you created.



7. Test the setup by navigating to **Administer** | **System Settings** | **Scheduled Jobs** and run the **Fetch Bounces** scheduled job.

8. Click on the **View Job Log** link and see if bounce processing was successful.

9. Configure the **Fetch Bounced mails scheduled job** settings to check the account regularly.

How it works...

The bounced e-mail system only works for mailings sent out using CiviMail, so it does not work if you use the **Send Email to Contacts** option available in the **Action** drop-down list when viewing contact search results.

If your mailing is bounced from a particular e-mail account, the bounced message is sent to the Gmail account you set up.

CiviCRM checks this account. It will put on hold any e-mail accounts that bounce messages and provides you with a report.

You can access these reports at **Administer** | **Reports** | **Mail bounce report**, or by navigating to **Administer** | **Mailings** | **Scheduled and Sent Mailings**.

The report shows the contacts' e-mail addresses affected, and gives an explanation of why the bounce occurred.

It is worth going back to the bounced e-mail messages in your Gmail account to check why they are getting rejected and make edits to your contacts accordingly.

See also

- `http://book.civicrm.org/user/current/initial-set-up/email-system-configuration/`

Creating mail templates for CiviMail

Developing e-mail templates for anything used to be a long, hard, and frustrating process because not all e-mail clients work in the same way. With the growth of mobile platforms, this became even more difficult. Now there are freely available services that will let you create a tested e-mail template that you can use in CiviCRM.

How to do it...

There are freely available, tested templates that you can use with CiviCRM.

1. Navigate to `http://www.campaignmonitor.com`; this website provides a free templating service with a visual editor. Other mail templates are available, such as MailChimp templates at `http://mailchimp.com/resources/html-email-templates/`. Using the Campaign Monitor service you can create an excellent e-mail template that can be downloaded as an HTML file. Alternatively, you can simply choose one from the MailChimp collection.
2. Open the HTML file in a text editor.
3. Navigate to **Administer** | **CiviMail** | **Message templates** and create a new template.
4. Paste in this template the HTML code from your downloaded file and you have an instant tested template that you can now edit.
5. Make sure that you include CiviCRM's **Unsubscribe and Domain** tokens in your template.

> When you compose your message, it's always better to create and test the plain text message first. Once you have perfected your message it's easy to copy and paste the basics into the HTML version. It's much harder to do it the other way round.

Creating mail templates for CiviMail in Drupal

CiviCRM's mailing system is excellent, but the templating system demands a certain degree of skills and knowledge of HTML, particularly if things go wrong.

Add to that the complexity of modern mail templates and it can become very challenging. When planning your CiviCRM deployment, you should consider the skill set of the users who are going to be responsible for sending out mailings. If the HTML-savvy skill set is not there, this recipe shows a different technique for creating perfect e-mail newsletters that use the CMS to create the e-mail. As a developer you will need to know a little about creating templates within your CMS system. In this example we will use Drupal.

How to do it...

Here we will create a new content type in our CMS to handle the composition of each mailing. Users will then be able to cut and paste the HTML into CiviMail.

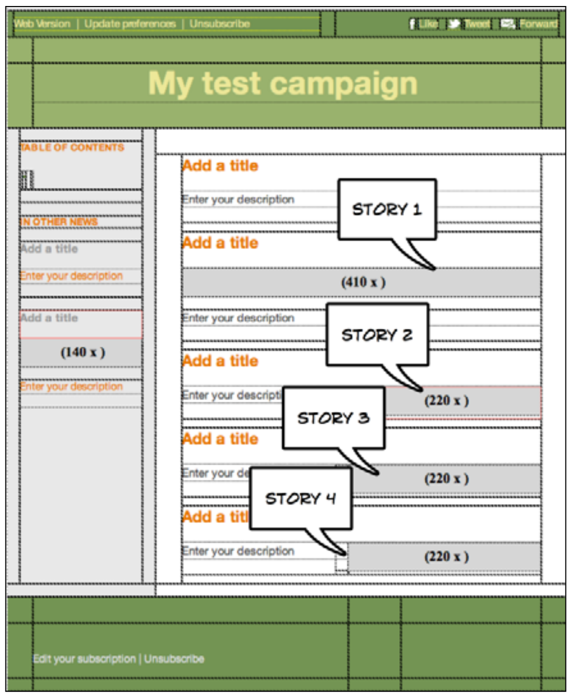
1. Create a new Drupal content type. Call it `email news` or something similar.
2. Plan your e-mail newsletter content. What are the maximum number of stories you will want to publish at any time? Which ones will have pictures? For each story, create a `text area` field and an `image` field on your `e-mail news` content type. So if you have 10 stories, you will have 10 `text area` fields and 10 `image` fields.

> For usability, you can contain the fields for each story inside a fieldset.

3. Create a node using your new content type. The edit screen may look something like this:

Here we have included only three stories and have collapsed the **STORY 2** and **STORY 3** fields.

4. Add the text and image for each story. Notice that you do not have to size the images at all, and you can use a Rich Text Editor such as CKEditor to style your text, so adding a story is a snap because you do not have to worry about design.
5. Download freely available templates from the Mailchimp website `http://mailchimp.com/resources/html-email-templates/` or Campaign Monitor, `http://www.campaignmonitor.com`.
6. Edit these in a suitable editor such as Dreamweaver and add into your design the number of stories that you have planned for in your `email news` content type. So if you have 10 stories, then you need to plan and place the 10 stories in your e-mail template.



Here we have a very simplified plan for four stories. You can see that you have to add the title, image, and story text for each story at specific places within the templates.

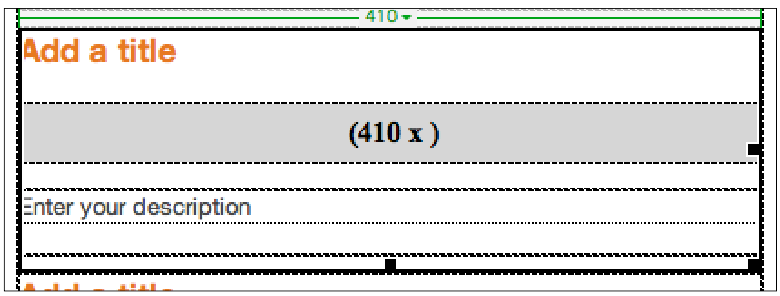
7. This e-mail template conveniently gives you the width for the images you need to use. So for **STORY 1** you need to use images that are 410 pixels wide. In Drupal, navigate to `admin/config/media/image-styles` and create the image styles for each width.

8. In Drupal, navigate to `admin/structure/types/manage/email-news/display` and allocate the image styles to the images for each story. So now you have an e-mail newsletter that displays the headlines, stories, and images in the right size with no fiddling around. But we have not added in the design we created in step 6.

9. In Drupal, follow the template-naming conventions and create a node template for the e-mail newsletter. In this case it would be `node–email_news.tpl.php`. Store it in your theme's `templates` folder.

10. In Dreamweaver, you have your e-mail template designed to hold the stories you are going to create with the `email news` content type. The template is composed of complex nested tables that have placeholders for your stories. In Dreamweaver it is easy to highlight a table that contains a content placeholder and then look at the underlying code.

    Here is its screenshot:



And here is the underlying code:

```
<layout label="Text with full-width image">
  <table class="w410" border="0" cellpadding="0"
    cellspacing="0" width="410">
    <tbody>
      <tr><td class="w410" width="410"><p class=
        "article-title" align="left"><singleline
        repeatertitle="true" label="Article Title">
        Add a title</singleline></p></td></tr>
      <tr><td class="w410" width="410"><img
        editable="true" label="Image"
        class="w410" border="0" width="410">
        Place image</td></tr>
      <tr><td class="w410" height="15"
        width="410"></td></tr>
      <tr><td class="w410" width="410">
      <div class="article-content" align="left">
        <multiline label="Description">
        Enter your description</multiline></div></td></tr>
```

```
      <tr><td class="w410" height="10"
        width="410"></td></tr>
    </tbody>
  </table
</layout>
```

The placeholders are highlighted.

11. Substitute your Drupal fields into the placeholders. This one is for the first story.

> Using the **Devel** module in Drupal you can isolate all the field values you want.

For the first story the substitutions would look as follows:

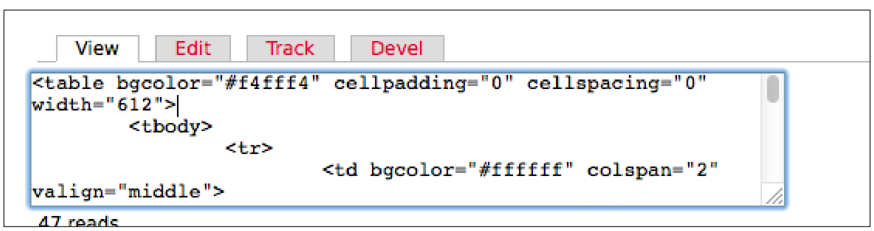
```
<layout label="Text with full-width image">
  <table class="w410" border="0" cellpadding="0" cellspacing="0"
width="410">
    <tbody>
      <tr><td class="w410" width="410"><p class="article-title"
align="left"><singleline repeatertitle="true" label="Article
Title"> <php print $node->title;?> </singleline></p></td></tr>
      <tr><td class="w410" width="410">
        <?php print theme('image_style',
        array('style_name' => '410', 'path' =>
        $base_path. $node->field_image_1
        ['und'][0]['filename'], 'alt' =>$node-
        >field_image_1['und'][0]['alt']  ));?> </td></tr>
      <tr><td class="w410" height="15" width="410"></td></tr>
      <tr><td class="w410" width="410">
        <div class="article-content"
        align="left"><multiline label="Description">
        <?php print  $node->field_story_1
        ['und'][0]['safe_value'];?>
        </multiline></div></td></tr>
      <tr><td class="w410" height="10"
        width="410"></td></tr>
    </tbody>
  </table
</layout>
```

For the images you need to add in the `$base_path` variable, otherwise our `image src` values will only show local paths.

12. Repeat this for each story placeholder.
13. Check that your code is working by removing the print render (`$content`) line from `node–email_news.tpl.php`. Copy the code between the body tags of your now-edited e-mail template and paste it into the `node–email_news.tpl.php` template. There may be some style problems, but the important thing is to make sure the field values are showing.
14. Once you are satisfied that the code is working correctly, add some form tags to `node–email_news.tpl.php`. Then, add in a `textarea` tag and paste the full code, including the opening and closing HTML tags from your e-mail template. Your code should look something like this:

```
<form action="none" method="get">
  <textarea id="pasteup" rows="5" cols="60"
    readonly>
    <!--All the email template code goes here-->
  </textarea>
</form>
```

15. Now complete all the stories in your `email news` node. The HTML text can now be copied from the text area of the form and pasted directly into a CiviMail mailing.



How it works...

This recipe exploits the content management features of the CMS to produce e-mail newsletters easily. It is quite an effort to set up and test, but the payoff is that users responsible for newsletter production require no HTML or design skills; yet they can produce very high quality mailings.

There's more...

You can add further code in your node template to account for **plain text** mailings—again a major reduction in the time and effort required to create mailings.

This recipe also stores the newsletter as content in your CMS. This means you can maintain a newsletter archive for website visitors. You can theme the content for use on the website and link back to it from your newsletter.

By adding some permission checks you can stop the form element from being visible to ordinary users.

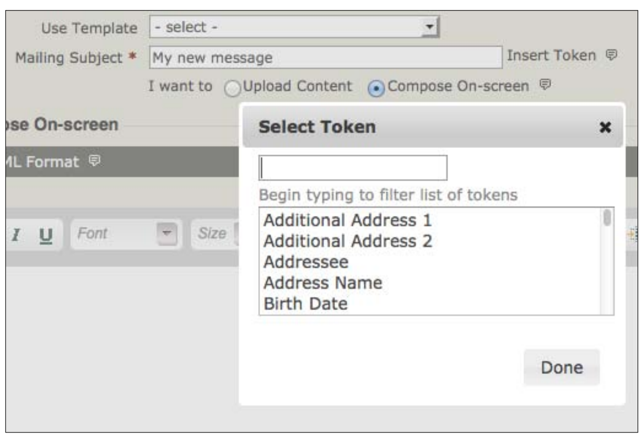
Using tokens in templates

In CiviMail you can substitute tokens for contact data so that your recipients get a personalized message.

How to do it...

CiviCRM has a comprehensive collection of tokens that you can use to personalize mailings.

1. Navigate to **Mailings** | **New Mailings** and create a new mailing.
2. Place your cursor in the main edit screen and click where you wish to place a token.
3. At the top right corner of the main edit screen, click on the **Insert Tokens** link.
   A pop-up box appears containing all the tokens that are available.
4. Click on the token you wish to use. This is now inserted into your message.



There's more...

You can also mail contacts by doing a search and selecting **Send Email to Contacts** in the **Actions** menu. You can put tokens into these mails too. CiviCRM restricts you to 50 contacts or less using this method.

Creating custom date tokens

Sometimes there is a token that is not available to you, so you will have to create it programmatically. This recipe shows a very simple example of adding a date token for the current date. It uses **plugin code**, which means you can substitute different token values.

Getting ready...

You don't really need to know much PHP but it helps if you know the basics of how to set up a basic Drupal module.

How to do it...

In this recipe we will create our own Drupal module to do the work. This is not too intimidating, particularly as our code is readily available online.

1. Create a folder in `/sites/all/modules` and give it a suitable name. Let's call this one `Cookbook`.
2. Inside this folder create a file called `cookbook.info`.
3. Open `cookbook.info` and add in the basic code that Drupal needs to identify the module:

   ```
   name = Cookbook
   description = Custom CiviCRM functions
   core = 7.x
   files[] = cookbook.module
   ```

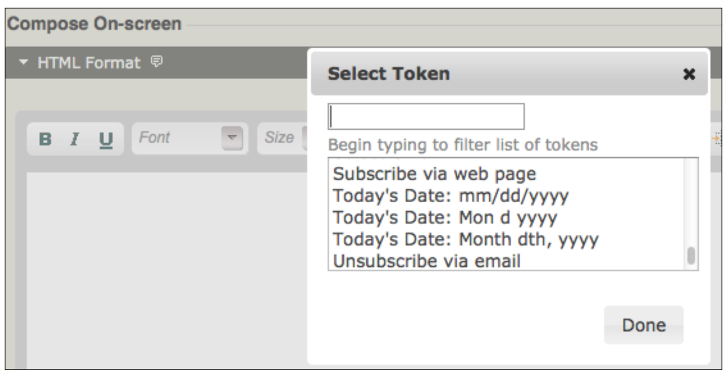
   Easy!

4. Create a file called `cookbook.module` and add in this code:

   ```
   <?php
   function cookbook_civicrm_tokens(&$tokens) {
     $tokens['date'] = array(
       'date.date_short' => 'Today\'s Date: mm/dd/yyyy',
       'date.date_med' => 'Today\'s Date: Mon d yyyy',
       'date.date_long' => 'Today\'s Date: Month dth, yyyy',
     );
   }

   function cookbook_civicrm_tokenValues(&$values, $cids, $job =
   null, $tokens = array(), $context = null) {
     // Date tokens
     if (!empty($tokens['date'])) {
       $date = array(
   ```

```
        'date.date_short' => date('m/d/Y'),
        'date.date_med' => date('M j Y'),
        'date.date_long' => date('F jS, Y'),
      );
      foreach ($cids as $cid) {
        $values[$cid] = empty($values[$cid]) ? $date : $values[$cid]
  + $date;
      }
    }
  }
```

5. Enable the module and test it by seeing if the tokens are available for placing into a test message.



How it works...

CiviCRM has a range of **hooks**. When CiviCRM runs a process such as getting a list of tokens, you can "hook" into that process with your own function and alter what CiviCRM does.

The first hook used here is as follows:

```
function cookbook_civicrm_tokens(&$tokens)
```

Here's the function in full:

```
function cookbook_civicrm_tokens(&$tokens) {
  $tokens['date'] = array(
    'date.date_short' => 'Today\'s Date: mm/dd/yyyy',
    'date.date_med' => 'Today\'s Date: Mon d yyyy',
    'date.date_long' => 'Today\'s Date: Month dth, yyyy',
  );
}
```

All this function is doing is adding the names of our tokens into the list of available tokens. We have three tokens for the current date, available in different formats, so adding token names is really easy.

The second hook used in our module is shown here:

```
function cookbook_civicrm_tokenValues(&$values,
  $cids, $job = null, $tokens = array(), $context = null)
```

Here is the first part of the function:

```
function cookbook_civicrm_tokenValues(&$values, $cids, $job = null,
$tokens = array(), $context = null) {
  // Date tokens
  if (!empty($tokens['date'])) {
    $date = array(
      'date.date_short' => date('m/d/Y'),
      'date.date_med' => date('M j Y'),
      'date.date_long' => date('F jS, Y'),
    );
```

Here we fill the date token with an array of three date formats for the current date.

Here is the second part of the function:

```
    foreach ($cids as $cid) {
      $values[$cid] = empty($values[$cid]) ? $date : $values[$cid] +
$date;
    }
  }
}
```

Here, for each contact ID that we cycle through during the mail process, we check if `$values[$cid]` has anything in it. If it is empty, we put in our date token. If it is not empty, we add our date token.

See also

- `http://civicrm.org/blogs/colemanw/create-your-own-tokens-fun-and-profit`

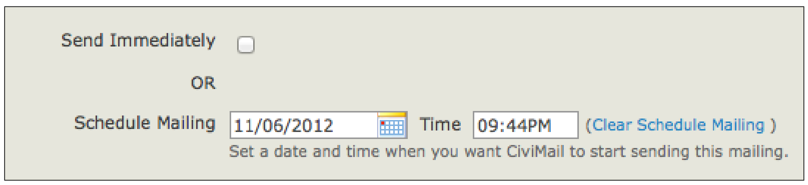
Scheduling CiviMail

You should consider scheduling mail particularly if you mail out to thousands of addresses. Mailings hog server resources, so it is always best to schedule them to your server when your server is least busy.

How to do it...

CiviCRM has an easy–to–configure scheduling system for CiviMail.

1. Ensure that you have set up a cron job on your web server.
2. Navigate to **Administer** | **System Settings** | **Scheduled Jobs** and make sure that the **Mailings Scheduler** option is set to run at regular intervals.
3. Navigate to **Mailings** | **New mailing** and create your new mailing.
4. In the last screen, ensure the **Send Immediately** box is unchecked. Then schedule your mail.



See also

- The *Setting up cron using cPanel* recipe in *Chapter 1, Setting Up CiviCRM*

Throttling mailings to comply with hosting restrictions

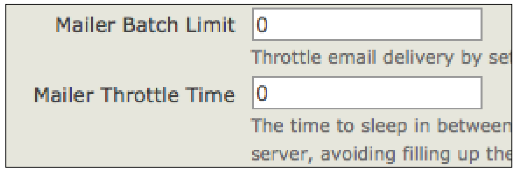
Many hosting companies put a limit on how many e-mails you can send out per hour. If you exceed this total then your hosting company will bounce all your e-mails and it rapidly becomes impossible to manage your system.

How to do it...

CiviCRM can set limits to how many mailings are sent out per hour.

1. Navigate to **Administer** | **System Settings** | **Scheduled Jobs** and make sure that the **Mailings Scheduler** option is set to run at hourly intervals.
2. Navigate to **Administer** | **CiviMail** | **Mail settings**.

3. Configure the **Mailer Batch Limit** option to a figure slightly below the hourly limit allowed by your hosting provider.



Creating newsletter subscription services using profiles

Newsletter subscription services are a great way of communicating with your contacts, as your audience is opting for information that they want. You can gain more information about your users if you add a couple of extra fields into your subscription form. For example, you might want to see what extra topics they might be interested in. Provided you do not overdo it, users will be willing to transact this information in return for the subscription service.

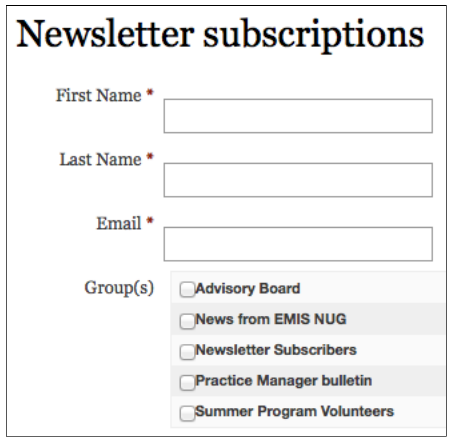
How to do it...

You can exploit the power of CiviCRM profiles to collect the extra data you want and combine it with the subscription checkbox.

1. Navigate to **Contacts** | **New Group** and create a group for each newsletter you wish to publish. You must ensure that the **Group Type** is a **Mailing List** and that **Visibility** is set to **Public Pages**.

2. Navigate to **Administer** | **Customize Data and Screens** | **Profiles** and add a new profile. Give the profile the title **Newsletter subscriptions**.
3. Add the **Contact email** field and the **Contact Group(s)** field.

4. Add **Contact fields** for **First Name**, **Last Name**, and any other extra fields you need. Try and limit the amount of extra information you want to gather otherwise it will put subscribers off.
5. Navigate to **Administer** | **CiviMail** | **CiviMail Component Settings**. Tick the **Enable Double Opt-in for Profile Group(s)** field if you wish subscribers to confirm their group subscriptions by e-mail. Your profile may look something like this:



How it works...

CiviCRM adds contacts to the group that have been selected. If you configured **Enable Double Opt-in for Profile Group(s)** then it will send out a confirmation e-mail for the subscriptions. You must have an e-mail field included in the profile to account for **Anonymous user** subscriptions.

There's more...

Navigate to **Administer** | **Customize Data and Screens** | **Profiles**. Click on the **More** link against the **Newsletter Management** profile and you will get an HTML snippet that you can paste into any web page and use straight away for your subscription service.

Creating newsletter subscriptions using URLs

You may want to provide your users with a central location where they can manage their CiviCRM subscriptions.

How to do it...

CiviCRM uses different URLs so that you can target some or all of your users' subscription services. You simply need to provide a link to this on your CMS user page.

1. `civicrm/mailing/subscribe` will provide a page to manage all group subscriptions.
2. `civicrm/mailing/subscribe?reset=1&gid=N` will provide a subscription link to a specific group where `N` is the group ID.

How it works...

Users have to be logged in to your website for these links to work. When they click on the links they are taken to a CiviCRM page where they can choose to subscribe or unsubscribe from groups set up as mailing lists.

Creating a standalone newsletter subscription form

You may want to provide a standalone form that allows your users to subscribe to a CiviCRM mailing group rather than providing a URL as in the previous recipe.

How to do it...

It's easy to set up a bit of HTML code that can be pasted into your CMS. In this example we will use Drupal. All you need to know are your group IDs.

1. Navigate to **Contacts** | **Manage Groups** and make a note of any group ID that you wish to use. The groups must be configured for use as mailing lists.
2. Open up a text editor, create a text file, and enter the following code:

```
<form action="http://book.dev/civicrm/mailing/
  subscribe" method="post">
<p>Email: <input name="email" type="text"
  id="email" /></p>
<table> <tr> <td><input name="groupID1"
  type="checkbox" value="1" /></td>
  <td>Advisory Board</td></tr>
  <tr><td><input name="groupID2" type="checkbox"
    value="1" /></td><td>Newsletter Subscribers</td></tr>
  <tr><td><input name="groupID3" type=
    "checkbox" value="1" /></td><td>Summer Program
    Volunteers</td></tr>
```

```
</table><p>
<input name="_qf_Subscribe_next" value="Subscribe"
  type="submit" /></p></form>
```

3. Substitute your own domain for `http://book.dev`.
4. Substitute your own group IDs for `groupID1`, `groupID2`, and `groupID3`.
5. Cut and paste the text into your website. In Drupal you could create a custom block to hold the text. This works well, but anonymous users are not sent to the confirmation page because Drupal does not create a session ID. You can fix this by adding a function into a custom module.
6. Create a folder in `/sites/all/modules` and give it a suitable name. Call it `Cookbook`.
7. Inside the folder create a file called `cookbook.info`.
8. Open `cookbook.info` and add in the basic code that Drupal needs to identify the module:

```
name = Cookbook
description = Custom CiviCRM functions
core = 7.x
files[] = cookbook.module
```

9. Create a file called `cookbook.module` and add in this code:

```
<?php
function cookbook_base_init() {
  if (!isset($_SESSION['cookbook'])) {
    $_SESSION['cookbook'] = 'session_initialized';
  }
}
```

10. Enable the module.

How it works...

If the user is already logged in the form works well but will fail for the anonymous user. This is why we create a simple function to provide a session ID for the anonymous user.

See also

- `http://drupal.org/developing/modules`

Getting a CiviMail report

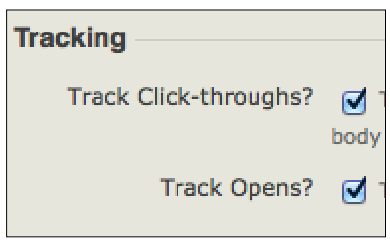
Getting useful information about your users' interests is critical if you wish to communicate effectively and keep them interested in what you are offering on your website. Some of these techniques involve forms, surveys, quizzes, and polls. All these methods rely on the user taking the trouble to actively complete forms and become less effective when there is a lot to fill in or when time is not available to the user.

CiviMail reports an indirect way of assessing what interests your contacts.

How to do it...

CiviCRM has an extensive and sophisticated reporting system. By setting up your mailings properly you can get CiviCRM to record what users click on in your mailings and provide a report.

1. Navigate to **Mailings** | **New mailing**. On the second screen of the mailing wizard, **Track and Respond**, make sure that **Track Click-throughs?** and **Track Opens?** are checked.



2. Once your mailing has been sent, navigate to **Mailings** | **Scheduled and Sent Mailings** and check the **mailing report** checkbox. Or you can navigate to **Reports** and select the various **Mail report** options that are available.
3. On the **Click Through Summary** tab on the report you can see what links the recipients clicked on when they read the mailing. In this way you can see what they are interested in.

How it works...

CiviCRM routes all links in your e-mail back to itself so that it can register what your contacts are looking at. From these reports you can see what sort of content is popular. Combine this information with some intelligent profiling and you can begin to segment and target your contacts much more effectively.

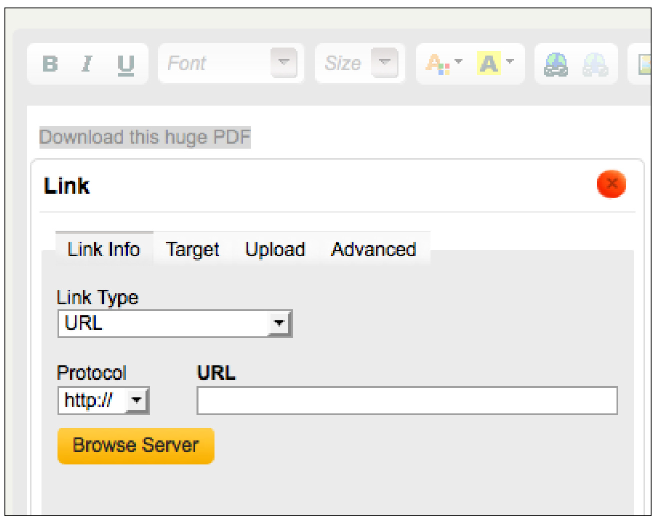
Mailing attachments in e-mails and CiviMail

Generally it is not a good idea to add attachments to mass e-mails in CiviCRM. Delivering each attachment consumes your bandwidth. If you have 1,000 contacts and you deliver each one a 1 MB attachment then you will have used 1 GB of bandwidth.

How to do it...

CiviCRM has controls that allow you to limit attachments in e-mail messages.

1. Navigate to **Administer** | **System Settings** | **Undelete, Logging and ReCAPTCHA** and set the e-mail attachment limits.
2. You can set the number of files and the maximum file size.
3. A less bandwidth-intensive method is to use the **Link** tool in the rich text editor to upload large files to your server and simply put links to them in your mail.



Allowing users to update information without logging in

Your users are busy people. Time is a commodity. You might be in a situation where you want to get users to provide you with data but they have to log in to your site to do it. The act of having to log in is a huge barrier to interaction. Luckily CiviCRM provides a way around this.

How to do it...

Using the CiviCRM checksum token in combination with a profile allows you to bypass the requirement to log in.

1. Create a profile you wish to use. The profile contains all the fields of information that you want the user to complete.
2. Make a note of the profile ID.
3. Enable **Profile Listings and Forms access** for **anonymous** and **authenticated** users in your CMS.
4. Now create a CiviMail mailing in the normal way.
5. You now need to use the `{contact.checksum}` token and the `{contact.contact_id}` token to construct a link back to edit the profile you created:

   ```
   http://www.myorganization.org/civicrm/profile/
   edit?reset=1&gid=N&id={contact.contact_id}&{contact.checksum}
   ```
6. Substitute your website for `http://www.myorganization.org`.

How it works...

When your contacts click on the link from within your mailing, they can update the profile without logging in. CiviCRM uses the checksums to link the profile to the contact ID for each user receiving the mailing.

There's more...

You need to be cautious about using checksums. For example, if one user forwards the e-mail to another user, then the second user would still be able to use the checksum and alter the first user's data. So make sure that any profile data available using a checksum is non-sensitive.

By default, CiviCRM sets a checksum lifespan to seven days. You can alter this by navigating to **Administer** | **System Settings** | **Undelete, Logging and reCaptcha** and changing the **Checksum Lifespan** setting.

See also

- `http://book.civicrm.org/user/current/common-workflows/tokens-and-mail-merge/`

6

Searching and Reporting

In this chapter, we will cover:

- Creating a membership mailing list using Advanced Search
- Using Search Builder to create a smart group
- Adding the external identifier to full-text searching
- Adding custom fields to a report
- Adding an extra display field to a report template
- Creating a dynamic relationship report using Drupal Views

Introduction

CiviCRM has powerful search features. **Advanced Search** provides the interface to accomplish most searches. **Search Builder** is slightly more technical and is useful when you cannot use Advanced Search. **CiviReports** is a component that adds reporting features to CiviCRM. The reports are in themselves excellent, and you can use them as templates to build your own, more customized versions.

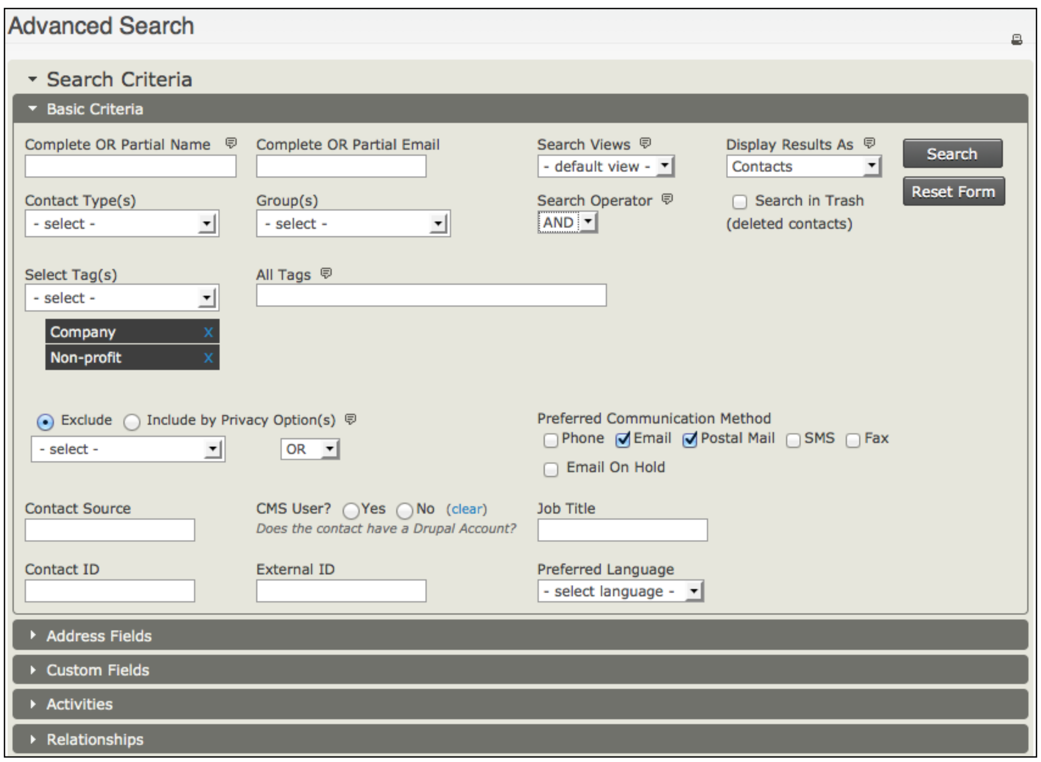
Creating a membership mailing list using Advanced Search

Advanced Search is a powerful tool that allows you to search across all CiviCRM components for useful information about your contacts. These searches can be saved as a smart group.

How to do it...

This recipe will show you how to create a smart group for a newsletter mailing and will provide extra hints for searching using Advanced Search.

1. Navigate to **Search** | **Advanced Search**. The main window shows the basic search interface:

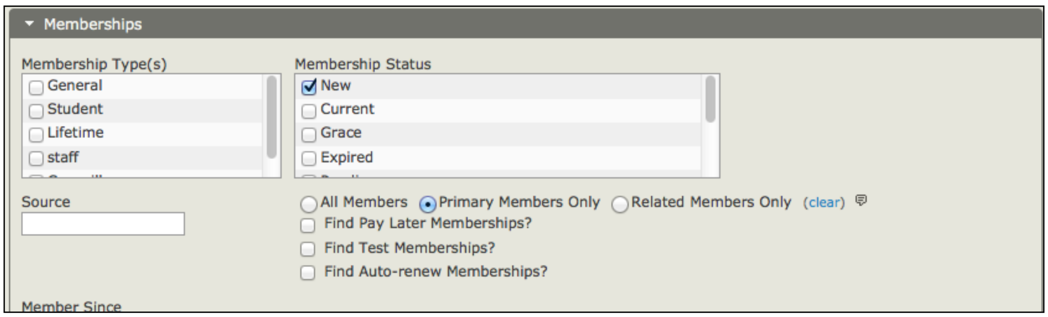


The **Search Operator** option in the second row is, by default, set to **AND**. This means that contacts will have to fulfil | any search criteria you select here *and* any criteria that we select in any of the other search sections, for example, Relationships. You can change this to **OR** if you want to. That would mean contacts would have to fulfill either criteria that you select in search sections.

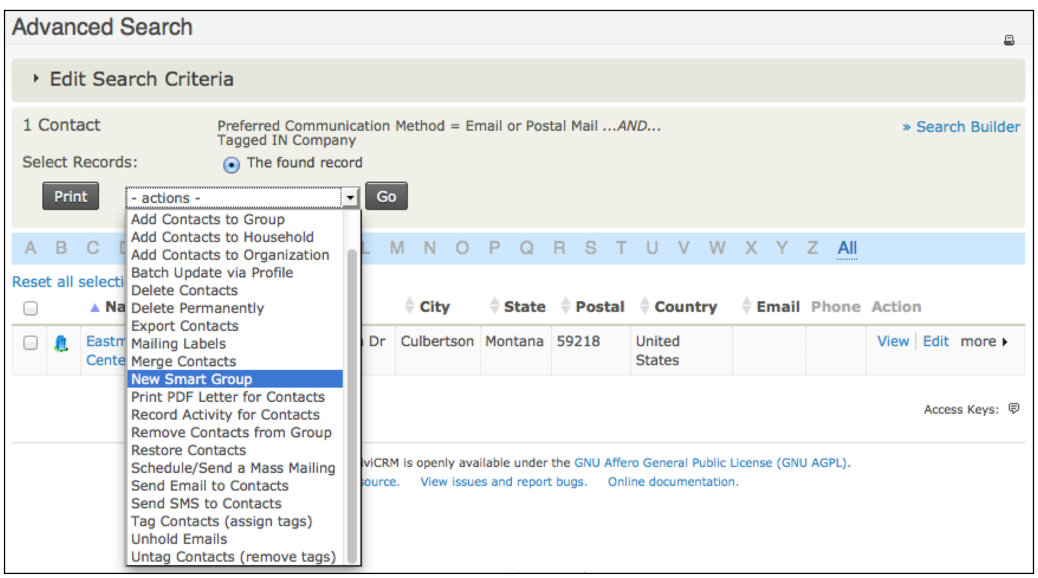
For the **Select Tag(s)** option in the third row, you can select more than one tag. When you do this, you are creating an OR selection for the tags. In this example, contacts will show if they are tagged **Company** OR **Non-profit**.

The **Preferred Communication Method** checkboxes, in the fourth row, is an AND search for the criteria selected. So, in this example, contacts would have to have e-mail AND postal mail as preferred communication methods to be shown in the results list. Once you decide your search criteria, we can perform the search.

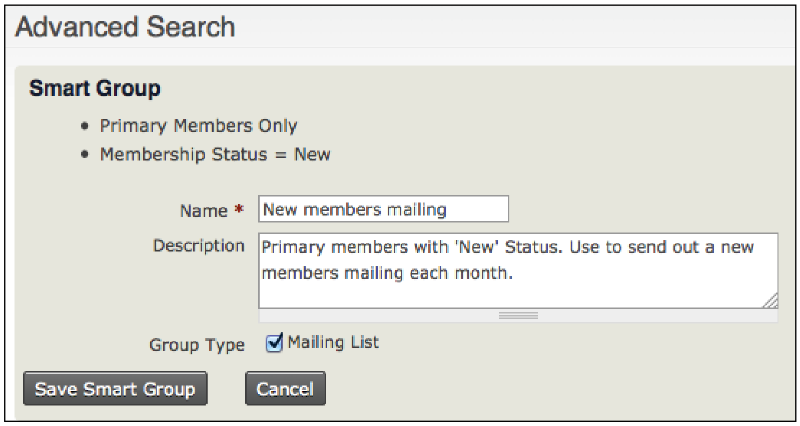
2. Use the **Advanced Search Membership** search section to find primary members who have the status of **New**:



3. From the **actions** drop-down menu, select **New Smart Group** and save your contacts:



4. Give the smart group a name.
5. Give the smart group a description.
6. If you want to use it for mailings, check the group type **Mailing list** checkbox as shown in the following screenshot:



This means you can send new members a mailing. They will be automatically removed from the group when their membership status changes to **Current** or another value that is not new.

How it works...

Advanced Search is pretty awesome. It combines core CiviCRM contact searches with search sections provided by other components, such as **CiviMember** and **CiviContribute**, and accomplishes most of your search tasks.

There's more...

If you create custom field sets, these become available in Advanced Search, provided you set each custom field as searchable.

See also

- The *Adding custom fields* recipe in *Chapter 1*, *Setting Up CiviCRM*
- Find out more about Advanced Search at `http://book.civicrm.org/user/current/the-user-interface/searching/`

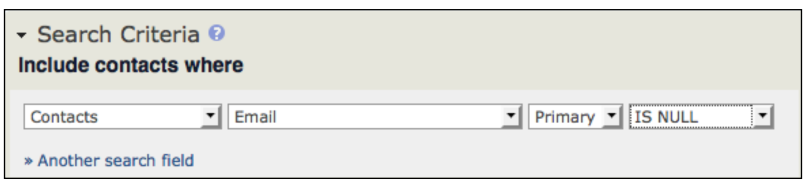
Using Search Builder to create a smart group

Search Builder is an alternative method of creating smart groups in CiviCRM. It differs from Advanced Search in that you can search for NULL values: for example, finding contacts with no e-mail address. It also allows OR searching with groups and tags.

How to do it...

Search Builder also has a slightly different syntax, which means you have to look up some values by visiting other administrative pages within CiviCRM. This recipe explores the Search Builder interface to create a smart group.

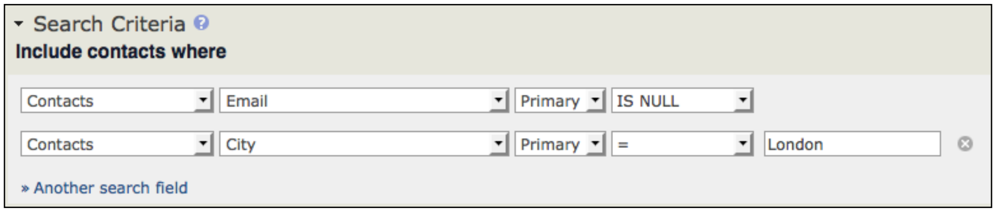
1. Navigate to **Search** | **Search Builder**.
2. Under **Include contacts where**, select **Contact** from the drop-down menu. CiviCRM now dynamically changes the search options available.
3. Select **Email** from the first drop-down menu.
4. Select **Primary** from the next drop-down menu.
5. Select **IS NULL** from the last drop-down menu.



This is a search for all contacts with no e-mail address.

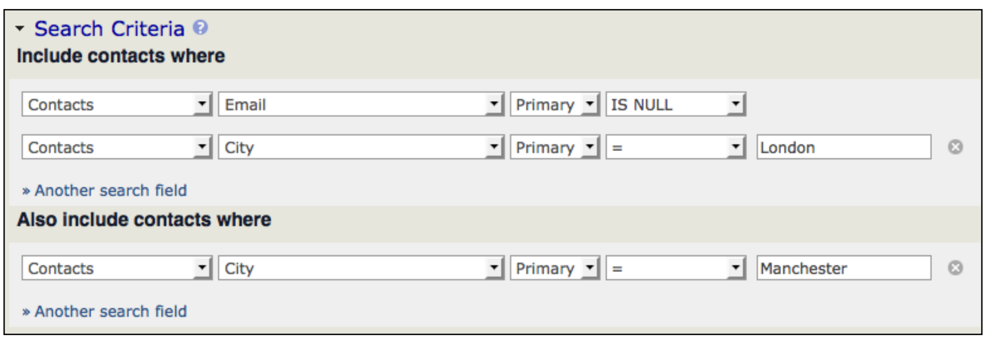
6. Do an AND search. Click on the **Another search field** link.
7. Under **Include contacts where**, select **Contact** from the drop-down menu.
8. Select **City** from the first drop-down menu.
9. Select **Primary** from the second drop-down menu.
10. Select **=** as the operator.

11. Enter `London` as the search value.



Now you have searched for contacts with no e-mail address *and* who live in London.

12. Do an OR search. Click on the **Also include contacts where** link.
13. Under **Include contacts where**, select **Contacts** from the drop-down menu.
14. Select **City** from the first drop-down menu.
15. Select **Primary** from the second drop-down menu.
16. Select **=** as the operator.
17. Enter `Manchester` as the search value.



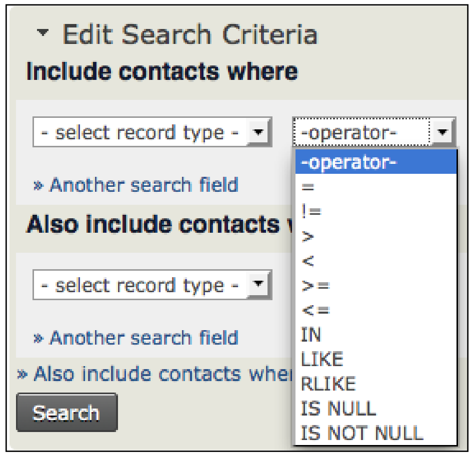
Now you have searched for contacts with no e-mail address *and* who live in London *or* contacts that live in Manchester.

> You can combine AND and OR searches to create very complex queries and save them as smart groups.

18. Once you are happy with your search results, you can save them to a smart group.

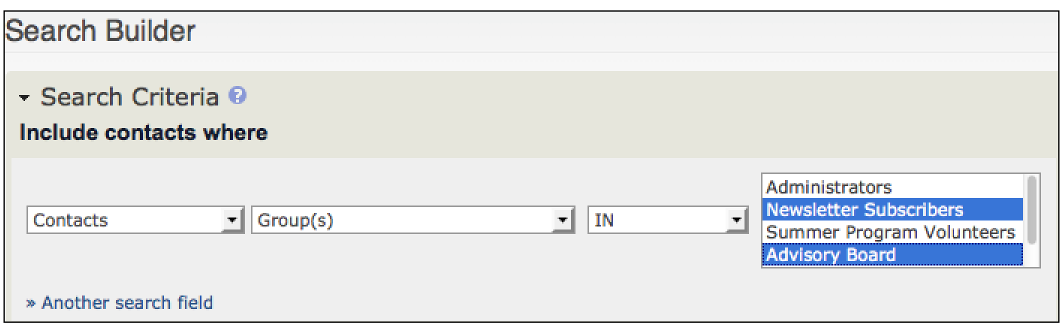
There's more...

The **operator** drop-down menu has some options that may seem unfamiliar:



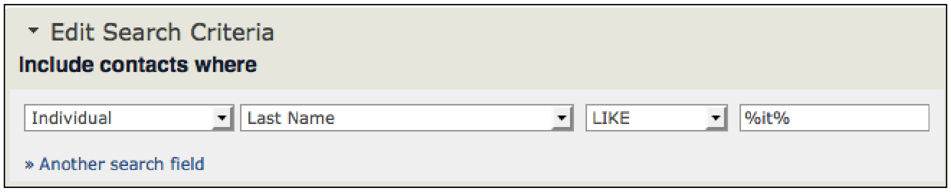
The **IN** operator allows you to test whether a value is contained in a list of other values. For example, you might want to find contacts that are in one or more groups in a list of groups.

You can enter these group IDs into a search as follows:



This would return a list of contacts that are in **Newsletter Subscribers** or **Advisory Board**.

The **LIKE** operator allows you to use wildcards and perform **fuzzy searching**. The symbol for the wildcard in CiviCRM is %:

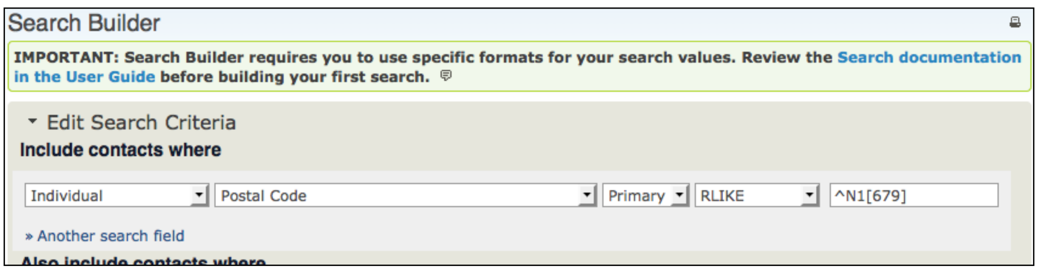


Here you would be looking for contacts with last names that contain the letters "it".

RLIKE is similar to **LIKE** but allows you to search using more complex patterns, using the regular expression syntax `REGEX`.

A regular expression is a pattern that provides a means of matching text.

In this example, you want to search for contacts with UK postcodes of N16 or N17 or N19:



Your regex expression will look as follows:

```
^N1[679]
```

Option values in Search Builder are not the same as in Advanced Search as we saw in the IN example in this chapter. Search Builder requires specific formats for some values, such as IDs, instead of labels for groups and tags.

See also

- Find more about regex at `http://en.wikipedia.org/wiki/Regular_expression for details about Regex/`
- Find more about searching in CiviCRM at `http://book.civicrm.org/user/current/the-user-interface/searching/`

Adding the external identifier to full-text searching

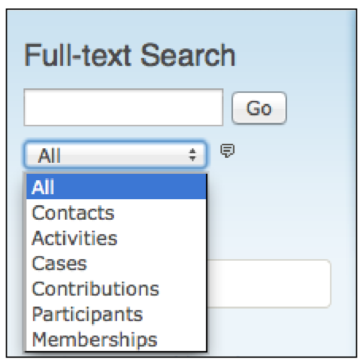
Many CiviCRM installations contain legacy data that has different unique identifiers than CiviCRM's contact ID. For example, you might have a membership system that identifies contacts with a membership number. Your staff will be used to using this identifier to locate data quickly. CiviCRM provides the external identifier field to hold this information and you can continue to use it to find your contacts.

CiviCRM also has a custom full-text search that allows you to search multiple contact fields at the same time. It is available as a block in Drupal. Unfortunately the custom search does not include the external identifier as a searchable field.

How to do it...

We will navigate to and open up the custom search PHP file and make a small edit to include the external identifier field in searches.

1. Make sure that the **Full-text Search** block is visible in Drupal:



2. Navigate to `/sites/all/modules/civicrm/CRM/Contact/Form/Search/Custom/FullText.php`.

3. Find the `fillContactIDs()` function. Inside this function look for the `$tables` array around line 421 as shown in the following screenshot:

```
421   $tables = array(
422     'civicrm_contact' => array(
423       'id' => 'id',
424       'fields' => array(
425         'sort_name' => NULL,
426         'nick_name' => NULL,
427         'display_name' => NULL,
428       ),
429     ),
```

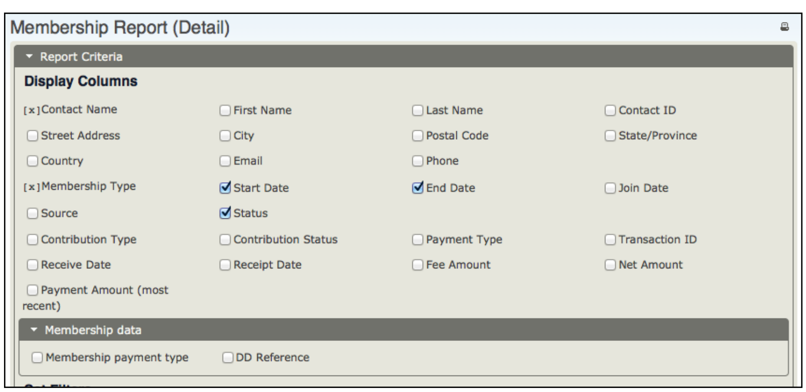
4. Add in the external identifier field:

```
421   $tables = array(
422     'civicrm_contact' => array(
423       'id' => 'id',
424       'fields' => array(
425         'sort_name' => NULL,
426         'nick_name' => NULL,
427         'display_name' => NULL,
428         'external_identifier' => NULL,
429       ),
430     ),
```

5. Save the file. The external identifier is now available to search. Note that this is a change to a core file in CiviCRM, so it will be overwritten by any updates.

Adding custom fields to a report

CiviCRM comes with an excellent suite of reports. If we navigate to **Reports** | **Membership Report (Detail)** and click on the report criteria, we can see CiviCRM divides the report criteria into sections. In the following screenshot you can see that there is a main section (for contact data) and then a section for membership. In the membership section there are a couple of custom fields:



What the report does not allow you to do is to display any custom fields that you created for contacts. For example, you might use CiviCRM to manage a Health IT Association, where the membership consists of health centers. For each health center, you want to know how many patients they have and how many doctors they have. You want to display this data on a membership report but it is not available from the default template.

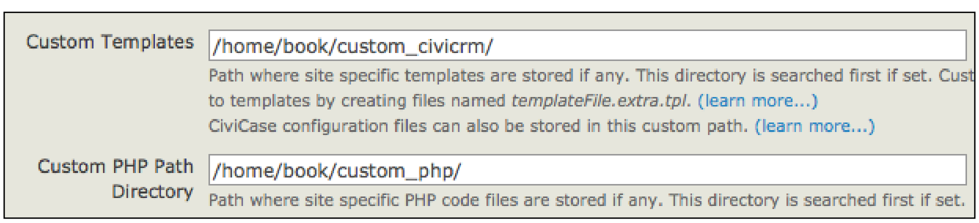
Getting ready

Create a custom field set for an organization and add in some custom fields for number of doctors, and number of patients.

How to do it...

We need to set up and tell CiviCRM where to find any custom files that we create. We will copy an existing CiviCRM membership detail report template and customize it to display the data we need.

1. On your web server, explore the filesystem structure. It varies from host to host. For example, if we look for where the Drupal `index.php` file is held, in some servers it will be `/var/www/vhosts/example.com/httpdocs/`. In others it might be `/home/siteaccount/public_html/`.
2. Create a directory within this structure that is going to hold CiviCRM customized template files. Make sure that the directory is writeable. Name it `custom_civicrm`. You can choose any name that suits you.
3. Create a directory within this structure that is going to hold CiviCRM customized PHP files. Make sure that the directory is writeable. Name it `custom_php`. You can choose any name that suits you.
4. Navigate to **Administer** | **System Settings** | **Directories**.
5. Set the **Custom Templates** path to the directory you created for templates.
6. Set the **Custom PHP Path Directory** field to the directory you created for PHP.



7. Navigate to **Reports** | **Membership Report (detail)** and run the report.

8. In your browser, view the page source and search it for the text "Report class":

```
<!-- Report class: [CRM_Report_Form_Member_Detail] --><div class="clear"></div>
```

This gives you a clue as to where to find the form that is used for the membership detail report. The Report class is `[CRM_Report_Form_Member_Detail]`. This is the directory structure within the CiviCRM module. The underscore (_) represents a backslash (\).

9. Navigate to your CiviCRM module and then `crm/report/form/member/detail.php`.
10. Copy `detail.php` and rename it `special.php`.
11. Store it in the custom PHP directory you created, recreating the exact directory structure used in the CiviCRM module.
    - CiviCRM module path: `/sites/all/modules/member/CRM/Report/Form/Member /detail.php`
    - Custom path: `/custom_php/ CRM/Report/Form/Member/special.php`
12. Return to the page source for the membership detail report.
13. Now search for ".tpl". The results show you directly where to find the report template file.

```
<!-- .tpl file invoked: CRM/Report/Form/Member/Detail.tpl.
    <form  action="/civicrm/report/instance/20" method="po
```

14. Navigate to your CiviCRM module and then the `templates` directory.
15. Within the templates directory navigate to `CRM/Report/Form/Member/detail.tpl`.
16. Copy `detail.tpl`.
17. Store it in the custom templates directory you created, recreating the exact directory structure used in the CiviCRM module. Rename it `special.tpl`.
    - Original path: `/sites/all/modules/civicrm/templates/CRM/Report/Form/Member/Detail.tpl`
    - Custom path: `/custom_civicrm/CRM/Report/Form/Member/Special.tpl`

18. Edit `special.php` and change the opening line from:

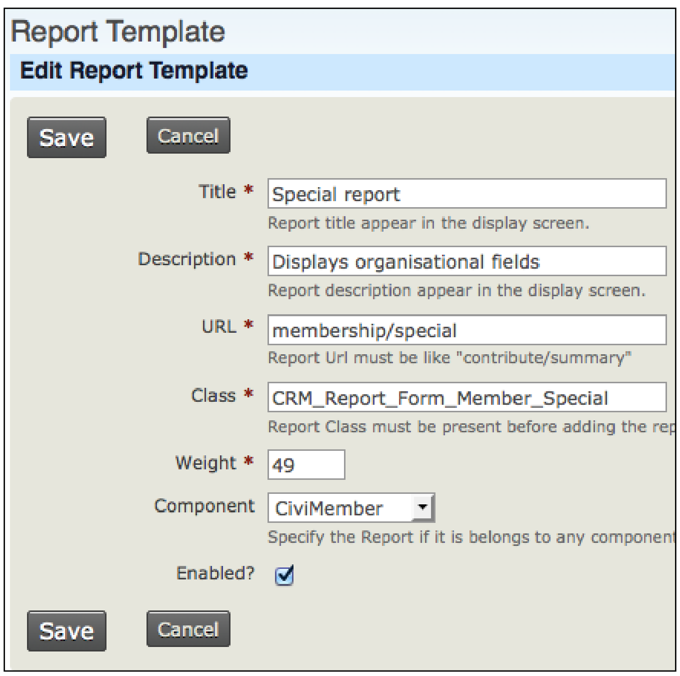
    ```
    class CRM_Report_Form_Member_Detail extends CRM_Report_Form {
    ```

    To:

    ```
    class CRM_Report_Form_Member_Special extends CRM_Report_Form {
    ```

    Note that "class" in this sense is not a class that you might use in CSS. In this case the PHP class is your blueprint for our special report.

19. Navigate to **Administer** | **CiviReport** | **Register Report**.
20. Set **Title** to **Special Report**.
21. Set **Description** to **Display Organizational Fields**.
22. Set the **URL** field for the report to **membership/special**.
23. Set the **Class** field for the report to **CRM_Report_Member_Special**.
24. Set **Component** to **CiviMember**.
25. Enable the report.



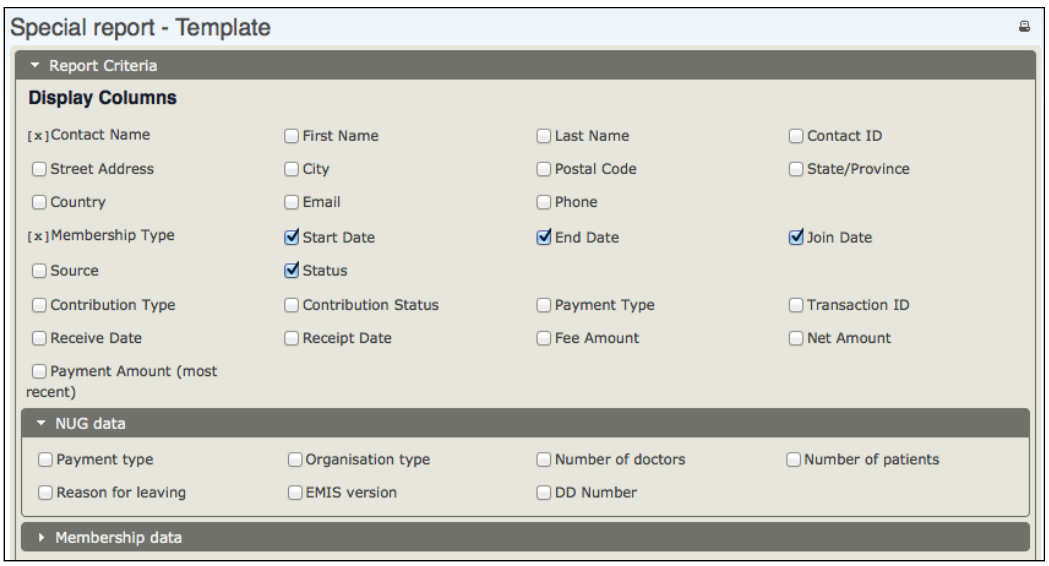
26. Save the template and now go to the URL `special/report/membership/special`. You will notice that it looks exactly the same except it has the title we gave it when we registered it as a report template.
27. Open `special.php` and navigate to the following line:

```
protected $_customGroupExtends = array('Membership',
'Contribution');
```

28. Add `Organization` into the array as follows:

```
protected $_customGroupExtends = array('Membership',
'Contribution', 'Organization');
```

29. Refresh the URL. The custom fields are now available.



How it works...

CiviCRM looks at the class you created when you registered the report. This in effect tells it where to look for the template and PHP files for the report when the URL is run.

See also

- Find more about report customization and report extensions at `http://book.civicrm.org/developer/version/4.1/the-extensions-framework/reports`

Adding an extra display field to a report template

There may be instances when you wish to add display fields to a CiviCRM report that is not available on the report template. For example, you might want to add the external identifier field.

How to do it...

In this example we will add the external identifier as a display option for a report.

1. Set up a custom report template and register it with CiviCRM.
2. Open the report PHP file. In this example we will use the `special.php` template created in the *Adding custom fields to a report* recipe in this chapter.

   Around line 49 we can see the code that generates the Contact checkboxes.

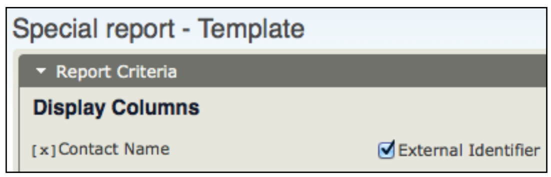
```
protected $_customGroupGroupBy = FALSE; function __construct() {
    $this->_columns = array(
      'civicrm_contact' =>
      array(
        'dao' => 'CRM_Contact_DAO_Contact',
        'fields' =>
        array(
          'sort_name' =>
          array('title' => ts('Contact Name'),
            'required' => TRUE,
            'default' => TRUE,
            'no_repeat' => TRUE,
          ),
          'id' =>
          array(
            'no_display' => TRUE,
            'required' => TRUE,
          ),
          'first_name' =>
          array('title' => ts('First Name'),
            'no_repeat' => TRUE,
          ),
          'id' =>
          array(
            'no_display' => TRUE,
            'required' => TRUE,
          ),
```

You can see that the code has a pattern: an array for the name of the field followed by an array for ID.

3. Copy one in for the external identifier.

```
'external_identifier' =>
          array('title' => ts('External Identifier'),
            'no_repeat' => TRUE,
          ),
          'id' =>
          array(
            'no_display' => TRUE,
            'required' => TRUE,
          ),
```

4. Save the file and reload the report.



The external identifier now shows in display columns and shows in our report.

Contact Name	External Identifier	Membership Type
Aarons, M_F	Aarons_1746	Full
Abbas, M_F	Abbas_3397	Full
Abbey Medical Centre	xxxx124	Full
	SNUG002	SNUG
Abbey Medical Group	xxx6227	Full

There's more...

Have a good look at the form for your report. In this example you can see that under `CRM_Contact_DAO_Contact`—effectively the `Contacts` table—you could have added any field that is defined in the table, for example, the birth date.

In other parts of report forms you can see other tables that reaccessed, for example, the membership table. So all you need to do is identify what extra fields you want to put in by looking at the database table fields and then add them in.

See also

- The *Adding custom fields to a report* recipe

Creating a dynamic relationship report using Drupal Views

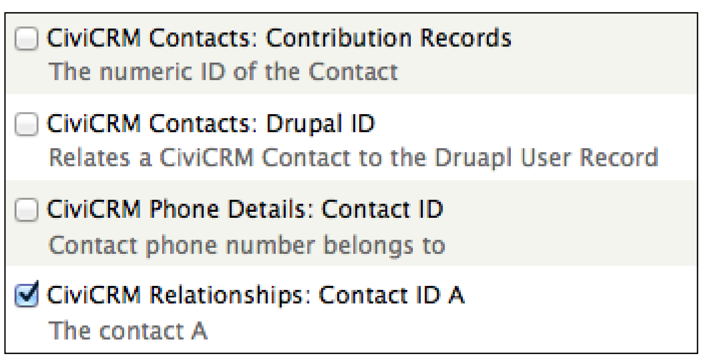
There are some limitations to the CiviCRM searching and reporting. For example, you might have a set of organizations that are tagged "Not For Profit" and you want to list all the employees of those organizations, grouped by each organization.

Such a search or report is not possible in CiviCRM without customization.

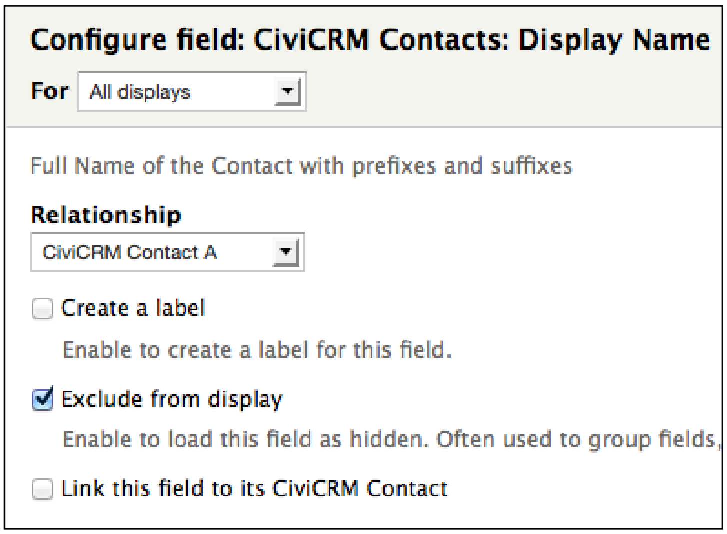
How to do it...

We will use the power of Drupal Views to create a relationship report. We will also expose the tag and relationship filter so that our report is dynamic.

1. In Drupal, navigate to **Structure** | **Views** and add a view. Call it `Relationship and Tag report`.
2. Set **Show** to CiviCRM contacts.
3. Click on **Continue and Edit** to continue and edit the view.
4. Set the view to display 100 items, paged.
5. Remove the CiviCRM **Contact ID** field.
6. Add the **CiviCRM Contact: Display Name** field. The view is now showing a list of contact display names.
7. Add a relationship. Select **CiviCRM Relationships: Contact ID A** as shown in the following screenshot:



8. Select **Limit results only to active relationships**.
9. Apply the relationship.
10. Add the **CiviCRM Contact: Display Name** field, but this time set it to display using the relationship.
11. Exclude the field from the display.



12. Add a filter. Choose **CiviCRM Tags** and select **Tag Name**.
13. Expose the tag filter.
14. Set the relationship for the filter to **CiviCRM Contact A**.
15. Add another filter. Choose **CiviCRM Relationships**.
16. Select **CiviCRM Relationships: Relationship Type A-to-B**.
17. Expose the filter. Do not set a relationship on this filter.
18. In **Format**, choose **Settings**.
19. For **Grouping Field**, select the *second* **CiviCRM Contact: Display Name** value that has the relationship.
20. Test the view and the filters.

How it works...

When we create the relationship in Drupal we do not specify what the relationship is. We do this in the exposed relationship filter.

We do not display the **CiviCRM Contact: Display Name** field that is linked to the relationship, as it would repeat for every record in the view.

Instead we use it as a grouping field. This makes it appear once in the display. So if we choose the employee relationship, the display name linked to the relationship, that is, the employer, is shown only once in summary with the display names of the employees beneath it. The tags filter adds an extra refinement.

7

Integrating CiviCRM with Drupal

In this chapter we will cover:

- Enabling Drupal Views
- Creating user accounts from contacts in CiviCRM
- Mapping contact data
- Creating user accounts on the fly with CiviCRM entities
- Using Webform CiviCRM to update relationship data
- Combining CiviCRM contacts with Drupal content using CiviCRM entities

Introduction

This chapter explores how to present and integrate CiviCRM data into your content management system. The recipes make extensive use of Drupal modules, some of which are still in development.

Enabling Drupal Views

Sometimes it is convenient to display CiviCRM data through Drupal. For example, you might want to display a list of contacts without giving the user access to CiviCRM. You might also want to combine data from CiviCRM with data from Drupal. For example, you have collected custom data about your contacts stored in CiviCRM that you want to display on the Drupal user page.

How to do it...

The key to integrating CiviCRM data is to install the **Drupal Views** module and to give Drupal access to the CiviCRM database tables.

1. Download the Drupal Views module from `http://drupal.org/project/views`.
2. Install and enable the module in the normal way.
3. Navigate to **Administer** | **System Settings** | **CMS Database integration**. Here you will see a page of code that you need to put into the Drupal `settings.php` file.

Views integration settings

To enable CiviCRM Views integration, add the following to the site `settings.php` file:

```
$databases['default']['default']['prefix']= array(

  'civicrm_acl'                              => '`emis_civicrm`.',
  'civicrm_acl_cache'                        => '`emis_civicrm`.',
  'civicrm_acl_contact_cache'                => '`emis_civicrm`.',
  'civicrm_acl_entity_role'                  => '`emis_civicrm`.',
  'civicrm_action_log'                       => '`emis_civicrm`.',
  [...]
  'civicrm_value_membership_data_2'          => '`emis_civicrm`.',
  'civicrm_value_nug_dasta_1'                => '`emis_civicrm`.',
  'civicrm_value_test_sub_activity_3'        => '`emis_civicrm`.',
  'civicrm_value_workshop_options_6'         => '`emis_civicrm`.',
  'civicrm_website'                          => '`emis_civicrm`.',
  'civicrm_worldregion'                      => '`emis_civicrm`.',
);
```

   We have shortened this table for the illustration.

4. Copy the code.
5. Navigate to `sites/default/settings.php`. You may have to change permissions on the `default directory` and on the `settings.php` file so that you can change it.
6. Edit `/site/default/settings.php` and paste the code at the end of the `settings.php` file.
7. Reset the permissions on `/site/default/settings.php` and `/site/default`.
8. Go to your MySQL manager. You need to give the Drupal database user "Select" permission on the CiviCRM database.

> In Drupal 6, replace the line `$databases['default']['default']['prefix']= array(` with `$db_prefix = array(`.
>
> Each time you add a set of custom fields, CiviCRM creates a new table. These fields will not be available to Drupal Views unless you add the table into `settings.php`.

How it works...

When you install and enable the Drupal Views module, it activates the database integration settings at **Administer** | **System Settings** | **CMS Database integration**.

Pasting the code into `settings.php` allows Drupal Views to access the CiviCRM tables. Giving the Drupal database user "Select" permission to the CiviCRM database tables is important for linking CiviCRM contact data with Drupal user accounts.

See also

- Find out more information about Drupal integration at `http://book.civicrm.org/user/current/website-integration/integrating-with-drupal/`

Creating user accounts from contacts in CiviCRM

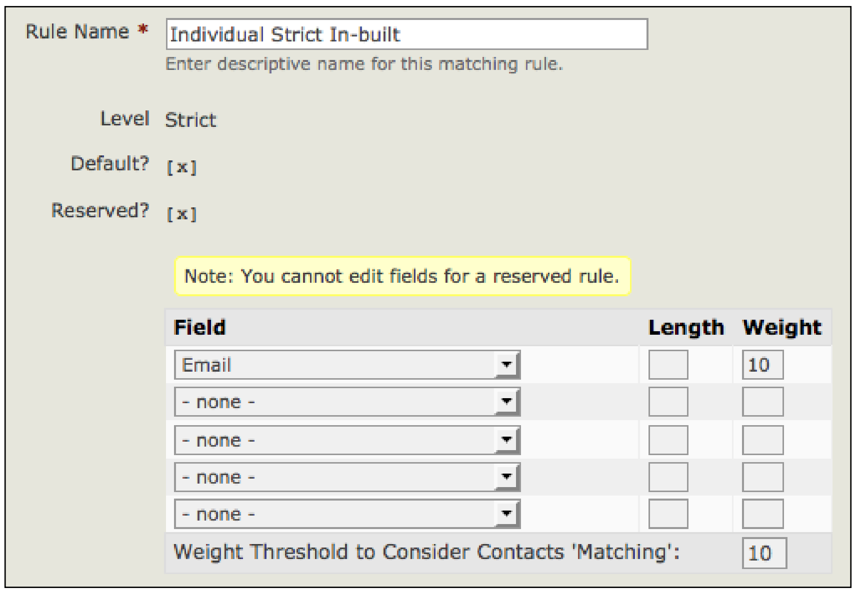
It's pretty easy to bulk create CiviCRM contacts from your Drupal users, but not so easy to bulk create Drupal users from CiviCRM contacts. For example, you can migrate a membership-based website into CiviCRM and Drupal, and you can create login accounts for your members.

How to do it...

We can use another useful Drupal module, **User Import**, to create our user accounts.

1. Install and enable the Drupal User Import module available at `http://drupal.org/project/user_import`.
2. In CiviCRM, navigate to **Search** and perform a search for the contacts you wish to add.
3. On the result set, select **Export Contacts** from the **actions** drop-down menu.

4. Select **export primary fields** and export your records. The exported file will be sent to your browser's `Downloads` folder.
5. Navigate to **Contacts** | **Find and Merge Duplicate Contacts**.
6. Select the **Individual Strict In-built** rule.
7. Ensure that this is set as the **Default** rule.



8. In Drupal, navigate to **People** | **Import**.
9. Use the CiviCRM export file and match its fields to the Drupal **Username** and **Email** fields.
10. Map the CiviCRM **Display Name** field to the Drupal **Username** field.
11. Map the CiviCRM **Email** field to the Drupal **Email** field.

> There are various other settings for your import that are beyond the scope of this book, such as setting roles, sending e-mails, and so on. Read the Drupal documentation carefully so that you set the rest of your imports correctly.

12. Import the users.

How it works...

CiviCRM combines the first name and last name Individual contact fields into the **Display Name** field, making it convenient to use as a Drupal username.

When each username is added, CiviCRM will use the **Individual Strict In-built** default rule to match contacts using just the **email** field. This means your CiviCRM contacts will be updated and no duplicates will be created.

See also

- Find more about the Drupal User Import module at `http://drupal.org/project/user_import`

Mapping contact data

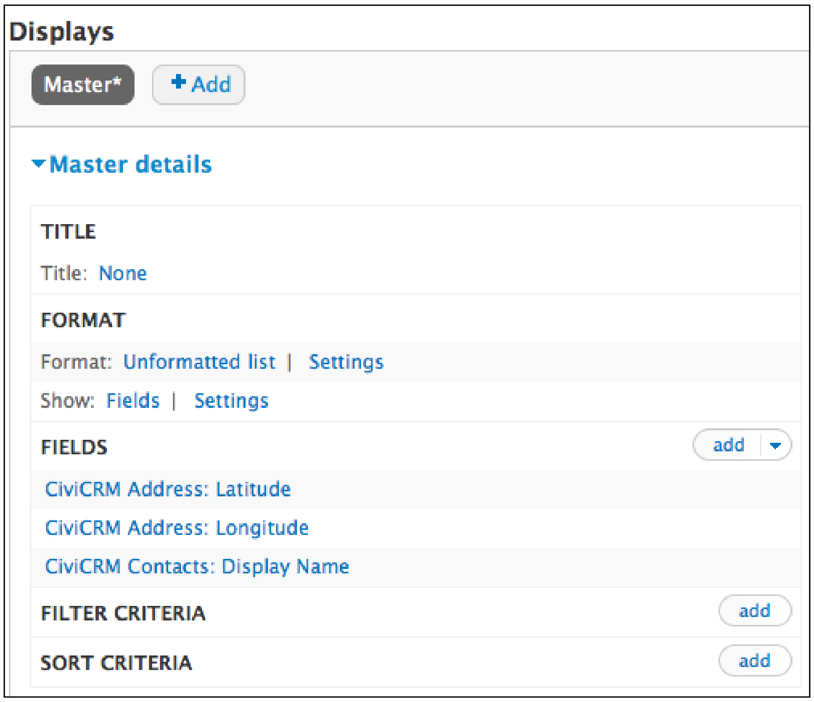
CiviCRM has excellent geolocation features. This means that you can map contact address data. For example, you could create a profile to hold data on voting intention, or political allegiance that you can then map. This means you can *visualize* the voting data.

How to do it...

CiviCRM's mapping features are very rudimentary, but, when linked to Drupal Views, they become very powerful. In this recipe we will use **Drupal Views** and **Drupal Open Layers** modules to produce a dynamic map that filters the display of contacts. In this example we will simply use the contact's last name as our filter.

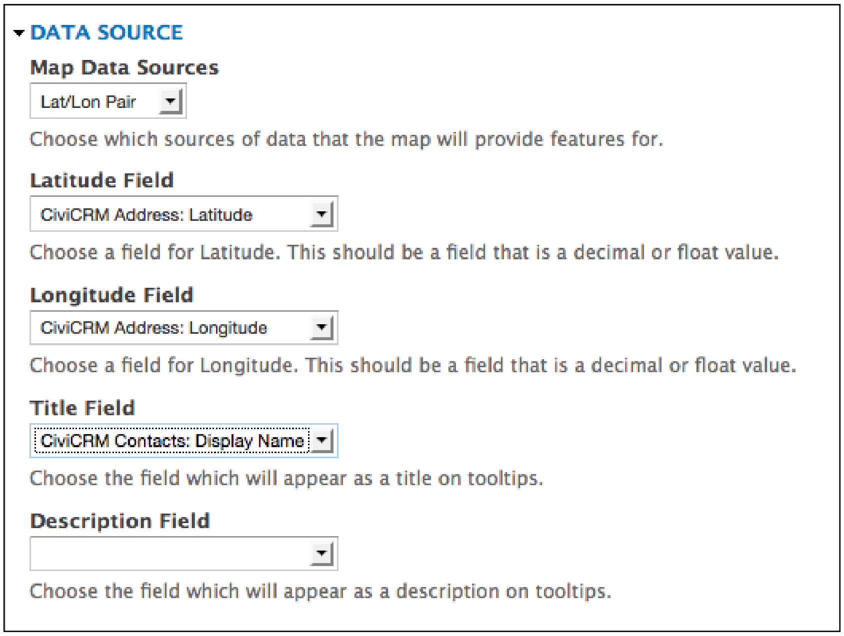
1. Navigate to **Administer** | **System Settings** | **Geocoding** and ensure that geocoding is set up correctly.
2. Navigate to **Administer** | **System Settings** | **Scheduled Jobs** and ensure **Address geocoder** is enabled and scheduled.
3. Install and enable the following modules:
    - Views (`http://drupal.org/project/views`)
    - OpenLayers (`http://drupal.org/project/openlayers`)
    - Libraries (`http://drupal.org/project/libraries`)
    - Proj4js (`http://drupal.org/project/proj4js`)
    - geoPHP (`http://drupal.org/project/geophp`)
    - CTools (`http://drupal.org/project/ctools`)

4. Make sure that you have integrated CiviCRM with Drupal Views.
5. In Drupal, navigate to **Administration** | **Structure** | **Views** and add a new view. Name it `Contacts by Last name`.
6. Show the CiviCRM contacts and save the view.
7. Add CiviCRM address fields for latitude and longitude.
8. Add the **CiviCRM Display Name** field as shown in the following screenshot:



9. Add an **OpenLayers Data Overlay** display to the view.
10. Name the overlay `Contact Map` and make the title `Contact Map`.
11. Change the format of the view from **Unformatted List** to **OpenLayers Data Overlay**.
12. Click on the **Format** | **Settings**.
13. Set **Map Data Sources** to **Lat/Lon Pair**.
14. Set **Latitude Field** to **CiviCRM Address: Latitude**.
15. Set **Longitude Field** to **CiviCRM Address: Longitude**.

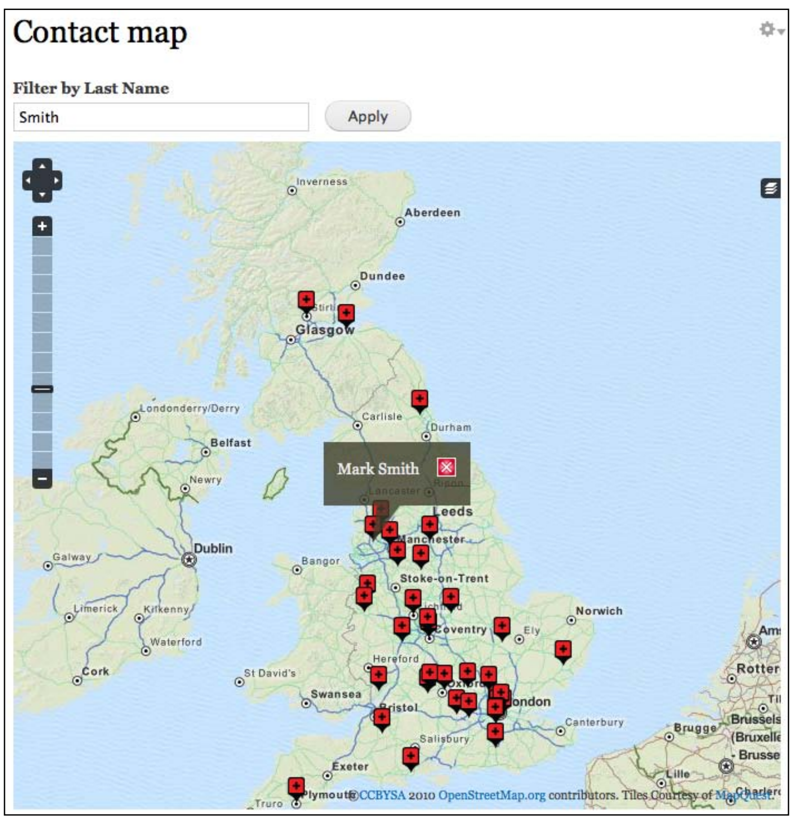
16. Set **Title field** to **CiviCRM Contact: Display Name**.



> You can add more fields to the display from CiviCRM and use Drupal Views re-write to combine field values together. All of these can be styled to present a rich source of information in the pop up.

17. Save the view. What you have created is called an **overlay**. Imagine it as a transparent film with your CiviCRM contacts marked on it.
18. Navigate to **Drupal** | **Structure** | **OpenLayers** and click on the **Layers** tab. The OpenLayer Data Overlay you created in the view is now listed.
19. Click on the **Maps** tab. This lists all available maps.
20. Select the **Default** map and in the **Operations** menu, select **Clone**.
21. Give the clone a name, a title, and a description.
22. Click on the **Centers** and **Bounds** tab. Scroll and zoom to set an initial view of the map.
23. Click on the **Layers** and **Styles** tab. In **Base Layers**, select the **MapQuest OSM** default layer.
24. In **Overlay Layers**, enable and activate the overlay you created in Drupal Views.

25. Set the map marker to **Red**.
26. Click on the **Behaviors** tab. In the pop up for features, select **Popup for Features**.
27. Save your clone.
28. Navigate to **Structure** | **Views** and edit the CiviCRM contacts by the **CiviCRM Last Name** view you created.
29. Add a **Page** display.
30. Set the format to **OpenLayersMap**, making sure that you apply the format only to this display.
31. Click on **Format** | **Settings**.
32. Give the page a URL, `contacts-name`.
33. Remove the page and display all records. Apply this to all displays. Check both displays to ensure that the page has been changed.
34. Add a **starts with** filter for the **CiviCRM Last Name** field and expose it on the view. Check and make sure the filter is applied to all displays.
35. Save the view and test it.



There's more...

These views can also be filtered by CiviCRM Group ID or by custom field values transforming CiviCRM into a powerful data visualization tool.

Using Webform CiviCRM to update relationship data

Updating related contact information is not very straightforward in CiviCRM. Let's imagine you are using CiviCRM to organize a soccer league. You would have three types of contacts: an organization contact type representing a soccer team, an individual contact type representing the team manager, and another individual contact type representing players. You would create a "manager" relationship between the team and the manager and a "player" relationship between the team and the players. Suppose you want to allow the manager to be able to update the team contact? You can do this by allowing managers to update teams by checking the permissions box on the relationship. But the update interface is through the user CiviCRM dashboard, and custom fields are not available through this technique. Enter `CiviCRM Webforms for Drupal`.

This powerful module exposes CiviCRM contact fields to the Drupal Webform module. CiviCRM Webforms can create and update information about contacts, relationships, cases, activities, event participants, group subscriptions, tags, and custom data.

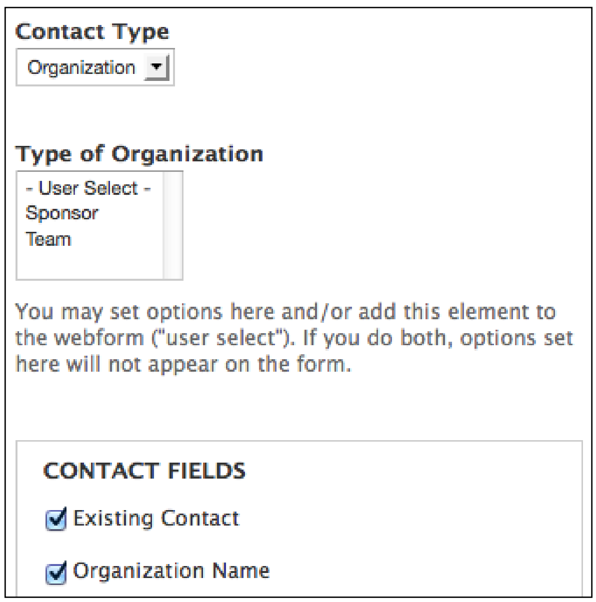
How to do it...

In this recipe we will create a simple form that allows you to update individual contact details and organizational contact details where the relationship between the two is Employee-Employer. We will use the CiviCRM Webform and Drupal Webform modules to achieve this.

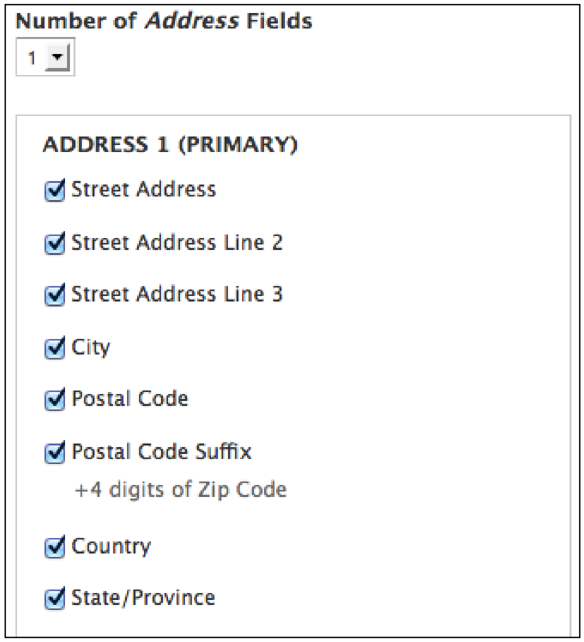
1. Install and enable the Webform module from `http://drupal.org/project/webform`.
2. Install and enable the dependent Libraries module (`http://drupal.org/project/libraries`).
3. Install and enable the Webform CiviCRM module (`http://drupal.org/project/webform_civicrm`).
4. Create a Webform node and call it `Update your details`.
5. Select the **CiviCRM** tab. Tick the **Enable CiviCRM processing** checkbox, and set the number of contacts on the form to **2**. The first contact will contain fields to update an individual contact record and the second will be to update the individual's employer contact record.

6. Click on the **Contact 1** tab. This is going to be set to **Individual**. Webform CiviCRM has prechecked the **Existing Contact**, **First Name**, and **Last Name** fields from CiviCRM. The **Existing Contact** checkbox is used to prefill the form with the logged in user's CiviCRM data.

7. Select the **Contact 2** tab. You need to set this contact to **Organization**.
8. Include the organization name.



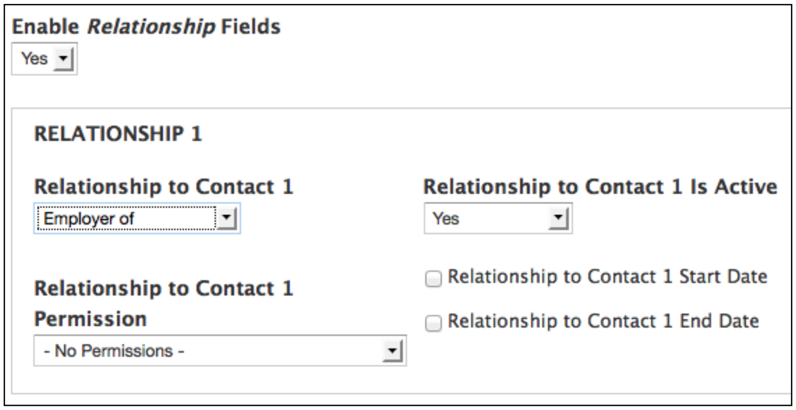
9. Add an address field.



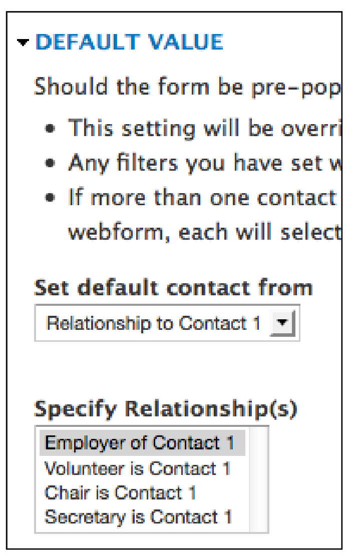
10. Add in a custom field set. Here we added in one called **Employer**.

11. Enable **Relationship Fields**.
12. Set the **Relationship to Contact 1** to **Employer of**.

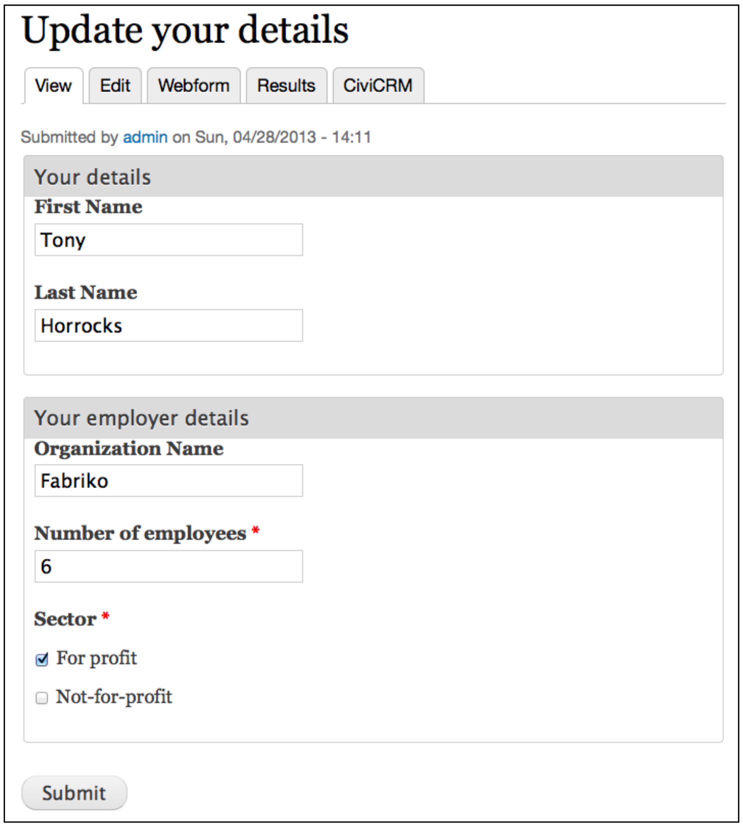
13. Set the **Relationship to Contact 1 Is Active** to **Yes**, as shown in the following screenshot:



14. Save your work so far.
15. Click on the **Webform** tab.
16. Change the label for the **Contact 1 Field set** to **Your details**.
17. Change the label for the **Contact 2 Field set** to **Your employer details**.
18. Click on the **Edit** link on the **Existing Contact Field for Contact 2**, the organization.
19. In the **Default Value** section, select **Relationship to Contact 1** in **Set default contact from**.
20. Set **Specify relationship(s)** to **Employer of Contact 1**:



21. In the **Display** section, set the **form widget** to **Static**.
22. Set **Display Contact Name** to **No**.
23. Save everything and test the form.



How it works...

Setting the **Existing Contact** checkbox for Contact 1 ensures Contact 1 is prepopulated with the logged in user.

Setting the default value for Contact 2 to **Employer relationship to Contact 1** ensures Contact 2 is prepopulated with any employer details related to Contact 1.

This makes the form behavior work for the logged in user.

Enabling **Employer relationship** between Contact 1 and Contact 2 ensures that, if an anonymous user completes the form, both contact details and the relationship are saved to CiviCRM.

See also

- Find out more about Webform CiviCRM at `http://drupal.org/node/1615380`

Creating user accounts on the fly with CiviCRM entities

You might want to add Drupal user accounts to your site when you create a CiviCRM contact. There is a **Create User Account** action on each contact summary screen on the **Actions** button menu. This allows you to create the account once the contact is created. However, this means completing information in an additional screen. Furthermore, Drupal does not send the user a notification that the account has been created. To overcome this, there is a new module by Eileen McNaughton still under development called **civicrm_entity**. This exposes CiviCRM entities to Drupal, which means that they can be recognized by the Drupal Rules module. CiviCRM entities include contacts, activities, memberships, cases, events, and contributions.

How to do it...

We will create a CiviCRM tag called "Create User Account". When we add a contact with this tag, it will trigger a rule that creates the user account:

1. Download, install, and enable the civicrm_entity module (`https://github.com/eileenmcnaughton/civicrm_entity`). Note that it is an experimental module and thus subject to change without notice.
2. Download, install, and enable the Drupal Rules module (`http://drupal.org/project/rules`).
3. Download, install, and enable the Drupal Entity API module (`http://drupal.org/project/entity`).
4. In CiviCRM, navigate to **Contacts** | **New tag** and create a new tag called **Create User Account**.
5. Set **Used For** to **Contacts**.
6. Save the tag.
7. On the **Tag Management** screen, make a note of the tag ID.
8. In Drupal, navigate to **Components** | **Workflow** | **Rules**.
9. Add a new rule. Name it `Create User Record from Civi Contact`.
10. Set **React on Event** to **CiviEntity Tag has been created**.
11. Save the new rule.
12. In the **Conditions** section, add a **Data comparison** condition.

13. Set the data selector to **civicrm-entity-tag:tag-id** (which is currently the last selector in the data selector list).
14. Set the operator to **equals**. Set the value to the tag ID. In this example the tag ID was **15**:

15. Add an action. Select **Create or Load Linked Drupal User Account.**
16. On the action configuration screen, select **Activate account** and **Send account notification email**.
17. Save the action.
18. In CiviCRM, test the rule by adding a new contact with the **Create User Account** tag.

How it works...

The CiviCRM contact is exposed to Drupal Rules and you can configure actions to be undertaken when the tag is created. CiviCRM actions are currently limited, but the whole scope of Drupal actions are available. So, you could add roles for the contact, send them custom e-mails, and so on. This module and others like it make it easy for the nondeveloper to create valuable functionality with no coding skills.

Combining CiviCRM contacts with Drupal content using CiviCRM entities

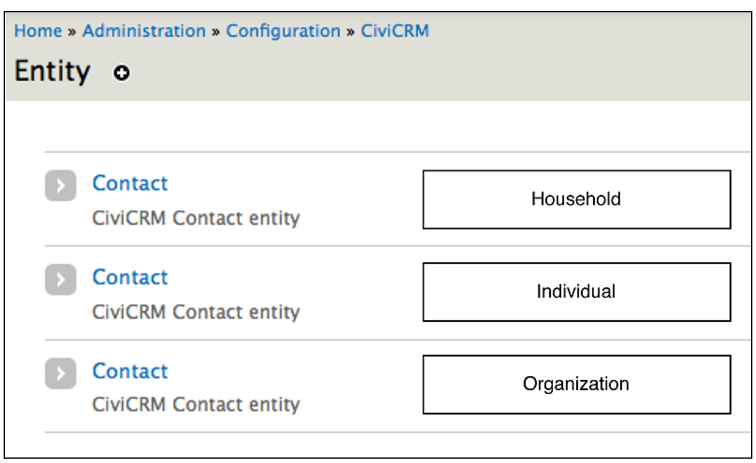
Linking CiviCRM contact data with Drupal data has until now not been possible directly. For example, imagine you run a cycling club. You have some members who are interested in mountain bikes, some in road bikes, and some in BMX bikes. You want to create "category" pages on your website that link users and content together in a community. In other words, if you could tag contacts and content with "Mountain bike", you could create a page that displays the two together.

Before now, you would have to create a user account on your website for each contact and link them both using a tag. Now you can do it directly. This makes it possible to produce category pages that list linked contacts without the requirement for a user account.

How to do it...

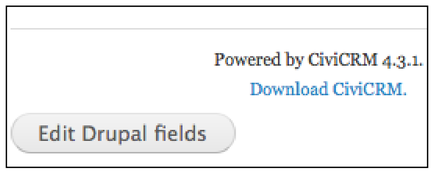
This recipe takes advantage of a module called **civicrmentity**, by Benjamin Doherty. This is not to be confused with the **civicrm_entity** module by Eileen McNaughton. These modules are currently incompatible. civicrmentity exposes CiviCRM entities to Drupal entities. This means that you can add Drupal fields to a CiviCRM contact, or, for this recipe, add taxonomy terms. Here, we will create a taxonomy term to link together CiviCRM contacts and content, and create an online community based around it.

1. Download, install, and enable the following modules:
    - civicrmentity module at `https://github.com/bangpound/civicrmentity/`
    - Entity API module (`http://drupal.org/project/entity`)
    - Views module (`http://drupal.org/project/views`)
    - CTools module (`http://drupal.org/project/ctools`)
    - Panels module (`http://drupal.org/project/panels`)
2. In Drupal, navigate to **Structure** | **Taxonomy** and add a new vocabulary called `Topics`.
3. Add the following terms to the `Topics` vocabulary: `Mountain bikes`, `Road bikes`, and `BMX`.
4. Navigate to **Configuration** | **CiviCRM** | **Entities**. The current version of the module does not label each content type. We have shown these in the following screenshot:

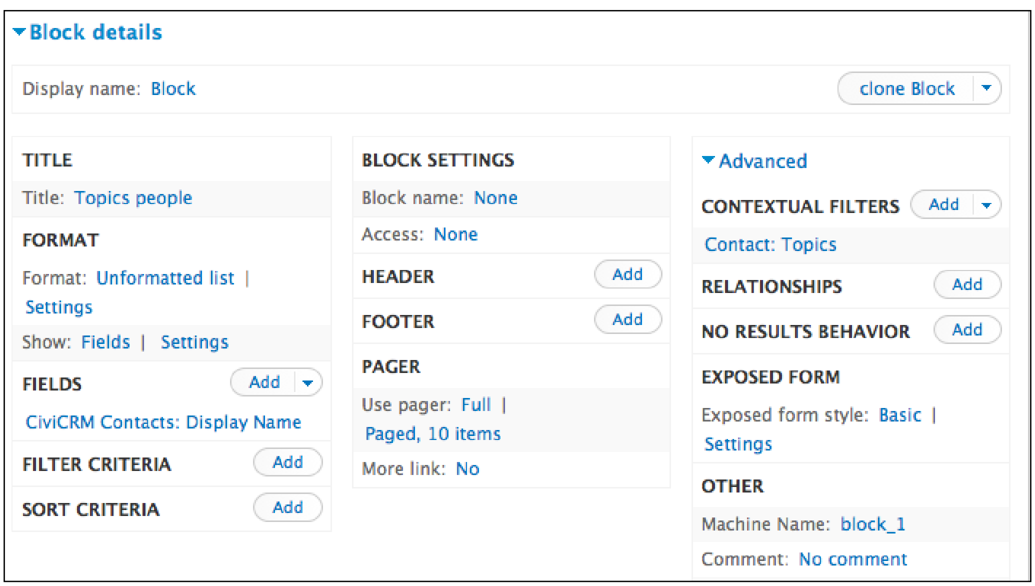


5. Click on the second Contact link for individuals and add a new field to the contact entity, and name it `Topics`.

6. Set the field type to **Term reference**.
7. Set the widget to **Checkboxes/Radio buttons**.
8. Save the field.
9. In the field settings, set **Vocabulary** to **Topics**.
10. Navigate to any CiviCRM contact and edit it. You will see a link to **Edit Drupal fields** at the foot of the page as shown in the following screenshot:



11. Click on the link, check the **Mountain Bike** box and save.
12. Navigate to **Structure** | **Views** and add a CiviCRM Contact view.
13. Add the **CiviCRM Contacts: Display Name** field to the view.
14. Add a contextual filter to the view. Select the **Contact: Topics** filter.
15. Preview the view using the **term ID** for Mountain bike. The contact with the term will appear.



How it works...

The civicrmentity module allows us to apply terms to CiviCRM contacts. This means we can use these terms as arguments in a Drupal view, so that if we filter the view for a term ID, all the CiviCRM contacts with that term will be listed.

There's more...

You can add the same term fields to CiviCRM events and Drupal content types. You could then create views for CiviCRM events and Drupal content that use the term IDs as arguments.

You could then use **Drupal Panels** to gather these views together so that you get a page that displays events, CiviCRM contacts, and Drupal content together. This in effect would be a community of users and content based on a topic of interest.

8
Managing Events Effectively

In this chapter, we will cover:

- Using jQuery to control form elements
- Using jQuery to show and hide form elements by user choices
- Using CiviDiscount with CiviEvents
- Collecting data for a paid event registration with Webform CiviCRM
- Using a shopping cart and Drupal views for event registration

Introduction

CiviCRM documentation for CiviEvents is comprehensive, so in these recipes we will use CiviEvents as a vehicle to explore how we can alter forms and introduce a couple of techniques using jQuery. We will also look at a relative newcomer to CiviCRM, **CiviDiscounts**, and the use of Drupal modules with CiviEvents.

Using jQuery to control form elements

jQuery is a versatile method used to alter any HTML page. In CiviCRM, we can use it to accomplish tasks that are impossible using custom templates or CSS. A common problem is how to expose different price sets on an event registration form.

Getting ready

In this example we have a membership organization called Inner City Arts. We have enabled the **CiviMember Roles Sync** module. This module synchronizes membership status to a Drupal role. In this case, if an individual is a current member of Inner City Arts, CiviMember Role Sync automatically assigns them the role of Member in Drupal. This means that when a member logs in, we can tell that they are a member of Inner City Arts because they have the Member role in Drupal.

If you want to play around with jQuery and see how the components work, you can use Firefox as your browser and install the Firebug extension, or use Google Chrome.

How to do it...

In this recipe, we will use jQuery to show different price sets to users depending on whether the logged-in user is a member of your organization:

1. Set up a price set for an event and include prices for members and non-members.

> Note that two price set items are showing. You want the **General Public** price to show to the anonymous user and the **Inner City Arts Members** price to show to logged-in users who are Inner City Arts members. You accomplish this using jQuery.

2. Create a Drupal block and copy and paste in the following piece of jQuery:

```
<script type="text/javascript">
(function($) {
  $(document).ready(function() {

//This is our custom jQuery code
```

```
  $("#priceset").find("label:not(:contains('Members'))").
each(function() {
      var id = $(this).attr('for');
            if (id) {
              $(this).prev().remove();
                $(this).remove();
              }
          });
//This is the end of our custom jQuery code
  });
  }) (jQuery);
</script>
```

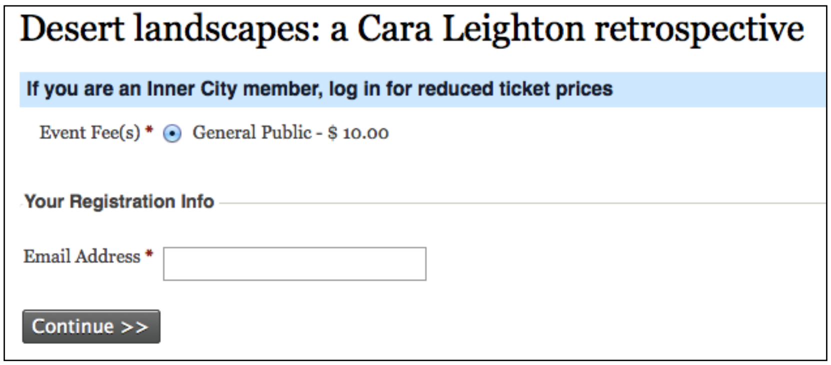
3. Set the block's **Text Format** to **Full HTML**.
4. Set the block's visibility to only show on specific CiviCRM event pages *and* only to users who have the **Member** role.
5. Create another block and copy and paste in the following jQuery:

```
<script type="text/javascript">
  cj(function($) {
  $(document).ready(function() {

//This is our custom jQuery code

$("#priceset").find("label:contains('Members')").each(function(){
      var id = $(this).attr('for');
      if (id) {
        $(this).prev().remove();
        $(this).remove();
         }
     });
  $("#priceset").prepend("<h3>If you are an Inner City member, log
in for reduced ticket prices</h3>");
//This is the end of our custom jQuery code
  });
  }) (jQuery);
</script>
```

6. Set the block's visibility so it only shows on specific CiviCRM event pages *and* only to the **anonymous** user.
7. Test the event registration.

How it works...

`cj(function($)` is used to tell CiviCRM to use the jQuery version that comes with the CiviCRM installation rather than the jQuery that comes with your CMS. If the user is anonymous, the second block jQuery runs and the **General Public** price is displayed with the message for Inner City Arts members to log in, as shown in the following screenshot:



If the user is logged in *and* is a member, the first block jQuery runs and only the member price is displayed.

This technique requires that the price set label for the members' price contains the word "Members".

The first block of jQuery only runs if the user is logged in and has the Drupal "Member" role. jQuery looks at the price set and finds all the items with a `label` element that does not contain the word "Members". It checks that the label is attached to a form input and then removes the input and the label. The effect of this is to remove the **General Public** price set item.

The second block does the reverse. It only runs for the anonymous user. jQuery finds all the items with a label that does contain the word "Members" and removes the price set items. The effect of this is to remove the member price set item. It also adds in a reminder for members to log in to get reduced ticket prices.

> Note that this technique is not secure and that your page can be manipulated so that the member prices could become available.

See also

- Find out more about this technique at `http://civicrm.org/blogs/stoob/using-only-jquery-and-civicrm-create-members-only-pricing`

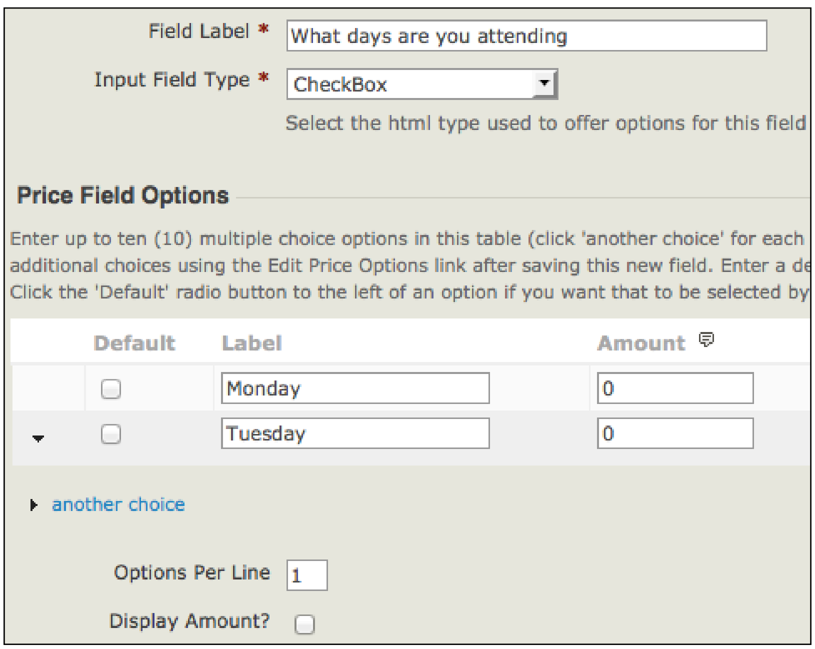
Using jQuery to show and hide form elements by user choices

Sometimes, you may want to expose form elements depending on what previous choices a user has made. For example, you might run a conference over two days. If people only want to attend one day, you will only want to show them further options for that day.

How to do it...

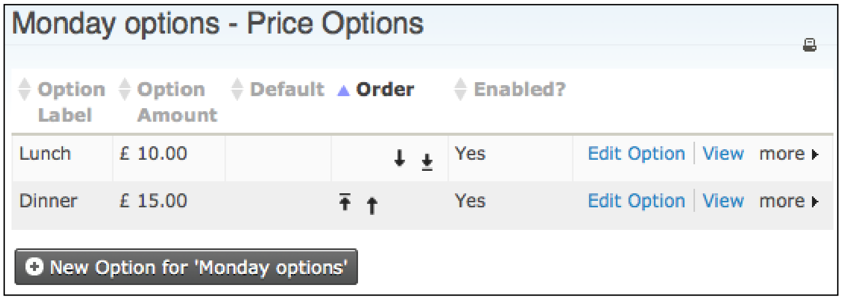
This recipe shows you how to gain control of your event registration forms using jQuery.

1. Set up a pricelist for your event. Navigate to **Administer** | **CiviContribute** | **New Price sets** and create a new set.
2. In the new set, create a checkbox field so users can choose what days they want to attend:

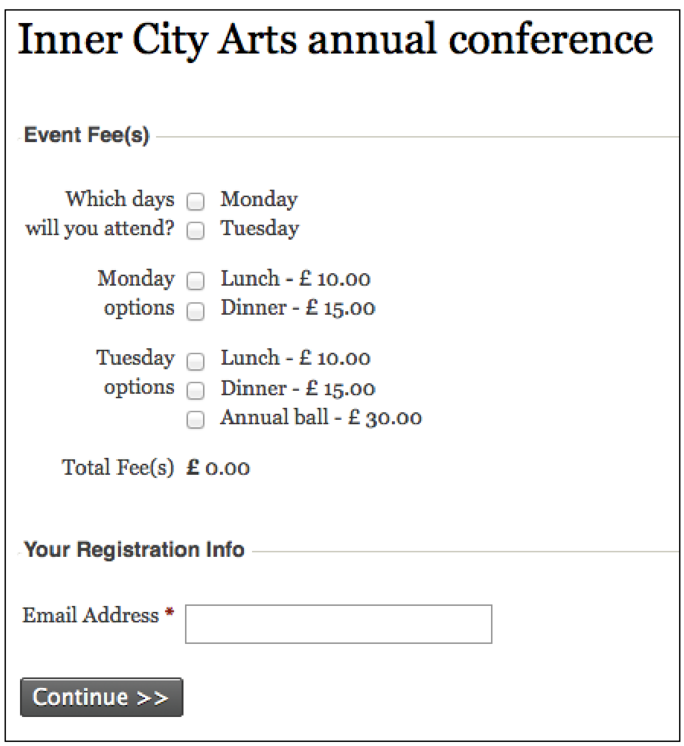


3. Provide two choices: **Monday** and **Tuesday**.
4. Make the **Amount** column for each choice to be free.
5. Uncheck **Display Amount**. This presents the user with two checkboxes with no price display.

6. Add another checkbox field for further choices for Monday, to be used if the user selects **Monday** in the first field.



7. Provide the **Lunch** and **Dinner** options for **Monday**.
8. Repeat this for **Tuesday**.
9. Create an event.
10. Apply the price set to it and make the event available for online registration. It will look something like this:

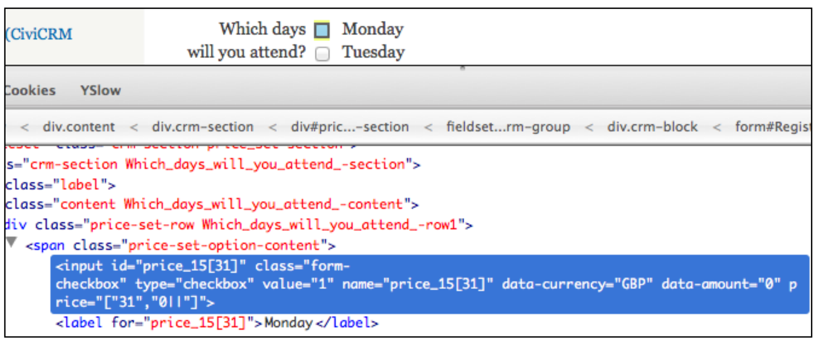


All the options are now displayed. When a user checks the boxes in **Which days will you attend?**, you will want the following behavior:

- If the user chooses **Monday**, display the Monday options, otherwise hide them
- If the user chooses **Tuesday**, display the Tuesday options, otherwise hide them

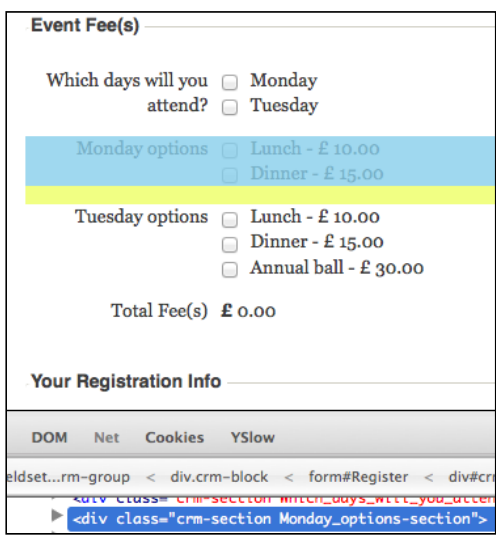
11. Get the ID for each of the checkboxes in **Which days will you attend?**.

    Use Firefox and Firebug to get these values. In your own situation, the values will be different from the ones shown in the following screenshot:



Here you can see that the **Monday** checkbox has an ID of `price_15[31]`. We can find the value for the **Tuesday** checkbox as well.

12. Get the class for each price set.



Here you can see that the **Monday options** field is contained in a `div tag` with a class of `Monday_options_section`.

13. Next, in a text editor, create the following code:

```
<script>
(function($) {
  $(document).ready(function() {
    $('.Tuesday_options-section').hide();
    $('.Monday_options-section').hide();
    $("#price_15\\[31\\]").change(function() {
      if($(this).is(':checked')){
        $('#priceset').find('.Monday_options-section').show();
      }
      else {
        $('#priceset').find('.Monday_options-section').hide();
      }
    });
    $("#price_15\\[32\\]").change(function() {
      if($(this).is(':checked')){
        $('#priceset').find('.Tuesday_options-section').show();
      }
      else {
        $('#priceset').find('.Tuesday_options-section').hide();
      }
    });
  });
  }) (jQuery);
</script>
```

14. Add the code to a Drupal block.
15. Set the block's **Text Format** to **Full HTML**.
16. Set the block's visibility to show on your CiviCRM event.
17. Now test the event registration page.

How it works...

The following code is called a **wrapper**. It just ensures that the code works in Drupal:

```
<script>
(function($) {
$(document).ready(function() {
```

The following code looks for the classes that contain the **Monday** and **Tuesday** options and hides them when the event is loaded:

```
$('.Tuesday_options-section').hide();
$('.Monday_options-section').hide();
```

The following function looks for the **Monday** checkbox ID and tests whether it has been checked. If it has, the price set with the `Monday_options-section` class is displayed. If not, it is hidden:

```
$("#price_15\\[31\\]").change(function() {
if($(this).is(':checked')){
$('#priceset').find('.Monday_options-section').show();
}
else {
$('#priceset').find('.Monday_options-section').hide();
}
});
```

The following function does the same for the **Tuesday** checkbox. If it is checked, the Tuesday price set is shown, if not, it is hidden:

```
$("#price_15\\[32\\]").change(function() {
if($(this).is(':checked')){
$('#priceset').find('.Tuesday_options-section').show();
}
else {
$('#priceset').find('.Tuesday_options-section').hide();
}
```

This code closes the wrapper for the jQuery:

```
});
  });
  }) (jQuery);
</script>
```

There's more...

jQuery is very versatile and powerful for all sorts of development tasks, not just CiviCRM. It is really quite easy to learn the basics and go on from there.

See also

- Find out more about customizing CiviCRM with jQuery at `http://civicrm.org/blogs/hershel/how-customize-civicrm-pages-jquery`

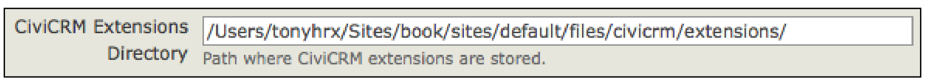
Using CiviDiscount with CiviEvents

CiviDiscount is an example of an agnostic CiviCRM extension, which means that it works in both Drupal and Joomla!. You can use the extension to create discount codes or to provide consistent discounts to your members.

How to do it...

CiviDiscount is an example of a CiviCRM extension. You will have to configure an **extensions directory** for your CiviCRM installation and download the extension before you can use it.

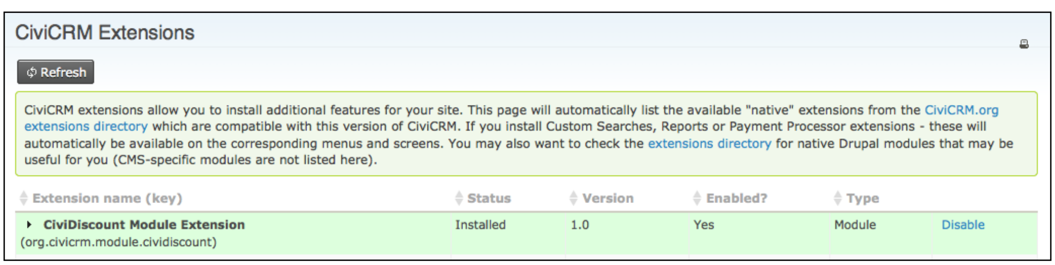
1. Navigate to **Administer** | **System Settings** | **Directories** and set a directory to hold CiviCRM extensions. The directory must exist on the server:



2. Navigate to **Administer** | **System Settings** | **Resource URLs** and set a URL for your CiviCRM extensions:

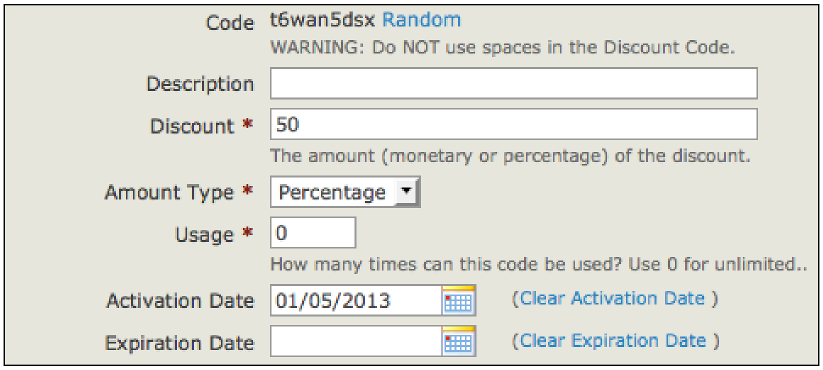
> Note that the URL has to match the path you created for the extensions directory.

3. Navigate to **Administer** | **Customize Data and Screens** | **Manage extensions**.

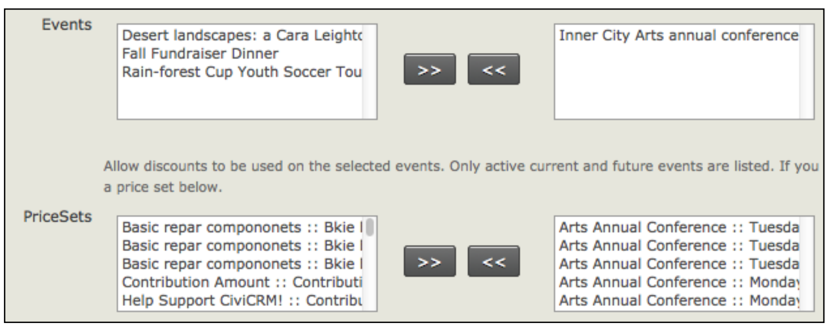


4. Download the CiviDiscount extension.
5. Add a navigation menu item for CiviDiscount. The URL is `civicrm/cividiscount/discount?&reset=1`.

6. Add a discount.
7. Add a 50 percent discount through the use of a discount code.

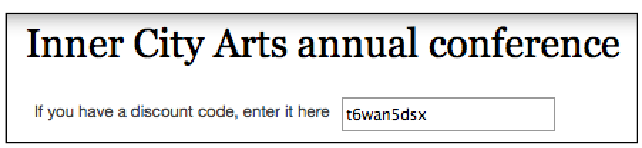


8. Apply this to an event.



Here there is an event called **Inner City Arts annual conference** and the discount has been added to each field in the event price set.

9. Save the discount and see how it affects event registration.



How it works...

The event now has a box to enter the discount code.

You can now make your selections from the event price set. These discounts appear on the event registration confirmation screen.

There's more...

CiviDiscount has many other options available, and it is a good alternative to using jQuery and different price set fields for different sorts of users. It is also an example of where organizations have contributed to CiviCRM to fund a much-requested feature.

See also

- Find out more about supporting new CiviCRM features at `http://civicrm.org/make-it-happen`

Collecting data for a paid event registration with Webform CiviCRM

Collecting complex organizational data and user data on one form during event registration is difficult in CiviCRM. You can create profiles to gather individual contact data, but you can only collect the organization name for organizational contact data.

How to do it...

In this recipe, we will use the Drupal Webform CiviCRM module to collect the data we require and then pass the user onto the CiviCRM event registration screen, with details already filled in.

1. Download, install, and enable the following Drupal modules:
   - Webform CiviCRM (`http://drupal.org/project/webform_civicrm`)
   - Webform (`http://drupal.org/project/webform`)
   - Libraries (`http://drupal.org/project/libraries`)
2. Create a simple event in CiviCRM. The event will show a single registration fee and collect the first name and last name of the registrant and the organization they work for. Make a note of the event registration URL.
3. Create a new web form using Webform CiviCRM.

4. Enable the form for CiviCRM processing.
5. Set the form for two contacts. Contact 1 will be the individual making the registration, and Contact 2 will be the organization they work for.
6. For Contact 1, select the **Checksum** field, the **Existing Contact** field, the **Contact ID** field, and well as other fields such as **First Name** and **Last Name**.
7. You can add as many built-in or custom CiviCRM fields as you like to both contacts, and therefore, collect a lot of detailed information.
8. Enable relationship fields for Contact 2 and set Contact 2 as the employer of Contact 1.
9. Save the form.
10. Click on the **Webform** tab.
11. Change the label for Contact 1 to About you.
12. Change the label for Contact 2 to About your organization.
13. For Contact 2, the organization, click on the Edit link for the Existing Contact field.
14. In the Default Value pane, change Set the default contact from to Relationship to Contact 1.
15. Set Specify Relationships(s) to Employer of Contact 1.
16. For Contact 1, the individual, click on the Edit link for the Checksum field. You will see that Webform CiviCRM provides you with snippets of code that you can use to redirect the user once the form has been submitted. You can adapt these to redirect the user to the event registration form.
17. In a text editor, enter the following code:

    ```
    civicrm/event/register?reset=1&id=16&cid=%value[civicrm_1_
    contact_1_fieldset_fieldset][civicrm_1_contact_1_contact_contact_
    id]&cs=%value[civicrm_1_contact_1_fieldset_fieldset][civicrm_1_
    contact_1_contact_cs]
    ```

    Here, the `ID` value is set to the ID of the CiviCRM event.

18. Click on the form settings.
19. Under **Redirection location**, select **Custom URL** and copy and paste in the code you created earlier:

20. Test the web form.

How it works...

When the user has completed filling in the individual details and organization details, they are sent to the paid event registration screen that displays minimum data prefilled through the use of the checksum.

If this were a free event, you could have added as many contacts as you like, have them mapped to the event, and then use the web form as your event registration screen instead of CiviCRM. You would not need to use the checksum at all.

Using a shopping cart and Drupal views for event registration

CiviCRM 4.2 includes a shopping cart system for event registration. This can be combined with a Drupal view of events to provide a quick and convenient event registration page.

How to do it...

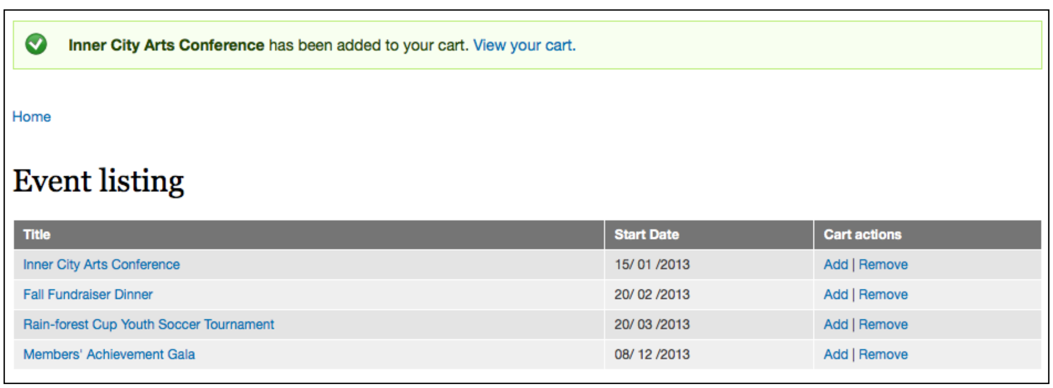
We will create a list of available CiviCRM events in a Drupal view. None of the events should include any special profile or custom field requirements. We will then add shopping cart links in the view.

1. Navigate to **Administer** | **CiviEvent** | **CiviEvent component settings** and tick the **Use Shopping Cart Style Event Registrations** checkbox.
2. Create a page view of CiviCRM events in Drupal views.
3. Add a filter to show only future events.
4. Sort the list in date order.
5. Add field values for **Event Title**, **Event Date**, and **Event ID**.
6. Set **Format** to **Table**.
7. We can rewrite the **Event ID** field in the Drupal view to provide us with add/remove links to the event shopping cart:

8. Save the view and test it.

How it works...

The view will now look something like this. When a user adds or removes an event from the shopping cart, a status message is displayed with a link to cart checkout:



9
Using Campaigns, Surveys, and Petitions Effectively

In this chapter, we will cover:

- Using activities for campaign planning
- Designing campaign dashboards in Drupal Views
- Using surveys effectively
- Recording survey results
- Using get out the vote effectively
- Using petitions effectively

Introduction

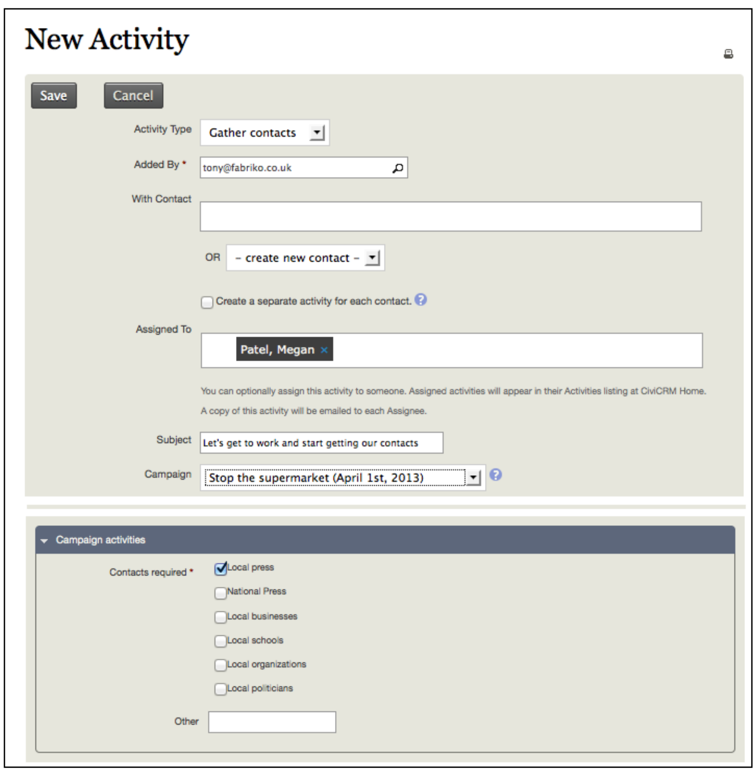
CiviCampaign acts as a *container* for activities, events, petitions, and surveys based around a particular *organizational* goal. This chapter explores how to set up campaigns, how to display them, and how to use some of the more specialized campaigning features of CiviCRM.

Using activities for campaign planning

In any campaign, there are a million things to do and usually not enough people to do them! You can use CiviCRM to define these tasks, allocate them to people, schedule their completion, and check their progress.

How to do it...

1. Navigate to **Administer** | **System Settings** and enable **CiviCampaign**.
2. Navigate to **Administer** | **CiviCampaign** | **Campaign types** and add a campaign type. Name it `Community Campaign`.
3. Navigate to **Campaigns** | **New campaign** and add a campaign.
4. Set the title to `Stop the Supermarket`.
5. Set the **Campaign Type** field to `Community Campaign`.
6. Leave the start date as the current date.
7. Set the **Campaign status** to **In progress**.
8. Save the new campaign.
9. Navigate to **Administration** | **Customize Data and Screens** | **Activity Types** and add some activity types. For example, add an **Activity Type** called **Gather contacts**.
10. Add a custom field set for the **Gather Contacts** activity type and create a custom field to show what sort of contacts you want gathered.
11. Add an activity and assign it to a contact.
12. Set the **Campaign** field to **Stop the Supermarket**.



How it works...

Adding CiviCampaign allows you to gather related activities under one umbrella. By adding custom activity types, you can create, allocate, and schedule administrative tasks associated with your campaign.

There's more...

1. Navigate to **Communications**| **Schedule Reminders**.
2. Create a **two days before** reminder for each assigned contact to get scheduled activities completed.
3. Create a **two days after** reminder to contacts who have allocated tasks that activities have not been completed to schedule.
4. Navigate to **Reports** | **Contact Reports** | **Activity Reports**.
5. Create a to-do list. Change the activity report criteria. Set the assignee to you, and activity status to **Scheduled**. Save a copy of the report and add it to your CiviCRM dashboard.

See also

- The *Adding activity types* recipe in *Chapter 1*, *Setting Up CiviCRM*
- The *Adding custom fields* recipe in *Chapter 1*, *Setting Up CiviCRM*
- The *Using scheduled reminders for activities* recipe in *Chapter 1*, *Setting Up CiviCRM*

Find out more about setting up CiviCampaign at `http://book.civicrm.org/user/current/campaign/setup/`.

Designing campaign dashboards in Drupal Views

CiviCRM does have a campaign report, but it is based around a contact listing rather than a list of activities associated with a campaign. There is also a **Campaign Dashboard**, but this does not provide appropriate listings.

In any campaign you might want to know:

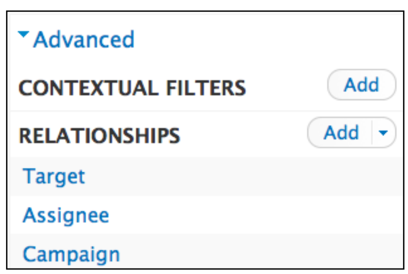
- What administrative activities are completed, scheduled, or overdue?
- What events are scheduled?
- Were events successful?
- What mailings were sent out?

- How effective were mailings?
- What petitions were organized?
- What surveys were organized?

How to do it...

We will use Drupal Views recipes to create the necessary listings. We can also use Drupal Panels or Drupal Context modules to organize our views into *dashboards*. These recipes assume that you have a working knowledge of Drupal Views.

1. Download, install, and enable the Drupal Views module: `http://drupal.org/projects/views`.
2. Add a new view of CiviCRM Activities.
3. Save the view.
4. In the **Advanced** section of the view, add a relationship.
5. Select the **CiviCRM Activity Targets: Target Contact ID** relationship. This will enable you to display the name of the target contact. Set the **Identifier** field to **Target**.
6. Add another relationship. Select the **CiviCRM Activity Assignments: Assignee Contact ID** relationship. This will enable you to display the name of the contact that was assigned the activity. Set the **Identifier** to **Assignee**.
7. Add another relationship. Select the **CiviCRM Activities: Campaign** relationship. This will enable you to display the title of the campaign for each activity. Set the **Identifier** to **Campaign**.



8. Add the **CiviCRM Activities: Subject (Subject)** field.
9. Add the **CiviCRM Activities: Scheduled Date** field.
10. Add the **CiviCRM Activities: Scheduled Activity Date (Scheduled Activity Date)** field.
11. Add the **CiviCRM Contacts: Display Name** field and set the relationship to **Target**. Set the label to **Target Contact**.
12. Add the **CiviCRM Contacts: Display Name** field and set the relationship to **Assignee**. Set the label to **Assigned Contact**.

13. Add the **CiviCRM Campaigns: Title** field and set the **relationship** to **Campaign**. Disable the **label** and exclude the field from the display.
14. In the **Format** section, set **Format** to **Table**.
15. In the **Table** settings, set **Campaign Title** as **Grouping field**.
16. Save the view.

Support safer cycling

SUBJECT	SCHEDULED ACTIVITY DATE	TARGET CONTACT	ASSIGNED TO
Consult with city authorities	08/04/2013	Mr. Allen Ivanov	Ms. Juliann Terry
Call to check security arrangements for protest.	30/04/2013	Billy Prentice II	Troy Jameson

Stop the supermarket

SUBJECT	SCHEDULED ACTIVITY DATE	TARGET CONTACT	ASSIGNED TO
Get sponsorship from this business	29/04/2013	Creative Technology Partnership	Nobody
Lets start gathering some data	23/05/2013	Nobody	Mr. Clint Terry
Start gathering contact data	24/05/2013	Nobody	Bernadette Cooper

How it works...

Drupal Views has powerful relationship tools that allow you to extract data from different tables and bring them into the same view.

It also allows you to summarize the data on a field—in this case, the **Campaign Title** field. The effect is a view that shows activities grouped under each campaign.

There's more...

- Remove the grouping field and add a filter for the campaign ID. This would create a display for a single campaign. Clone the display and change the campaign ID to get displays for different campaigns.
- Add filters for the Activity scheduled date. For example, you could set a filter to display activities that have passed their scheduled date and are still scheduled; in effect, overdue activities.
- Add the Contact ID for each relationship and use the Rewrite field feature in Views to make links back to the related CiviCRM contacts.
- Add the Activity ID and use the Rewrite field feature in Views to make links back to each activity.
- Add a Drupal User relationship and combine it with other relationships to get lists of activities you have assigned, or that have been assigned to you.

- Create another view of mailings. Add in fields for click throughs, openings, and other useful data. Link it to campaigns using the Campaign relationship. This provides a mailing report for each campaign.
- Create a custom field set for events. Add a field called **Event report**. Create a view of events that includes the field for the event report. Link it to campaigns using the Campaign relationship. This provides an event report for the campaign.
- For each view, set the campaign ID as a **Contextual filter**. Gather each view into a Panel or a page using the Contexts module. This creates a dashboard that unites events, mailing, and activities into one page for each campaign.

Using surveys effectively

In any campaign you might want to get quick answers to simple questions. For example, in an election campaign you might want to ask:

- Who do you intend to vote for?
- What issues are important to you?

CiviSurvey is used in these sorts of situations where you have a very limited set of questions to ask, and where the contacts you intend to survey are already in your database.

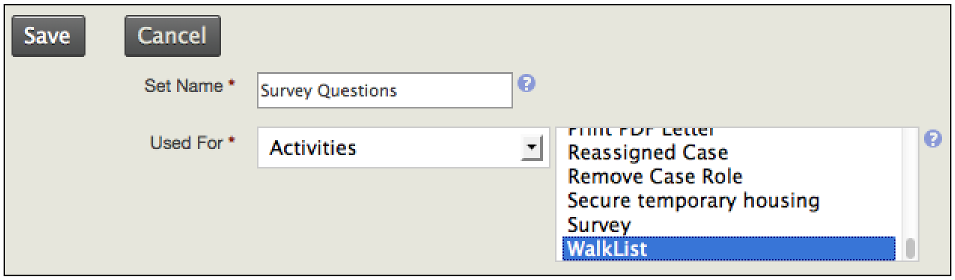
In **CiviCampaign**, we can use surveys and petitions to collect data to support our campaign objectives. Surveys are used to collect data from contacts that are already in CiviCRM. In this recipe we shall create a survey to ask contacts about their voting intentions. We will distribute the survey to our volunteers so that they can conduct the survey from door-to-door (a **walklist** survey).

How to do it...

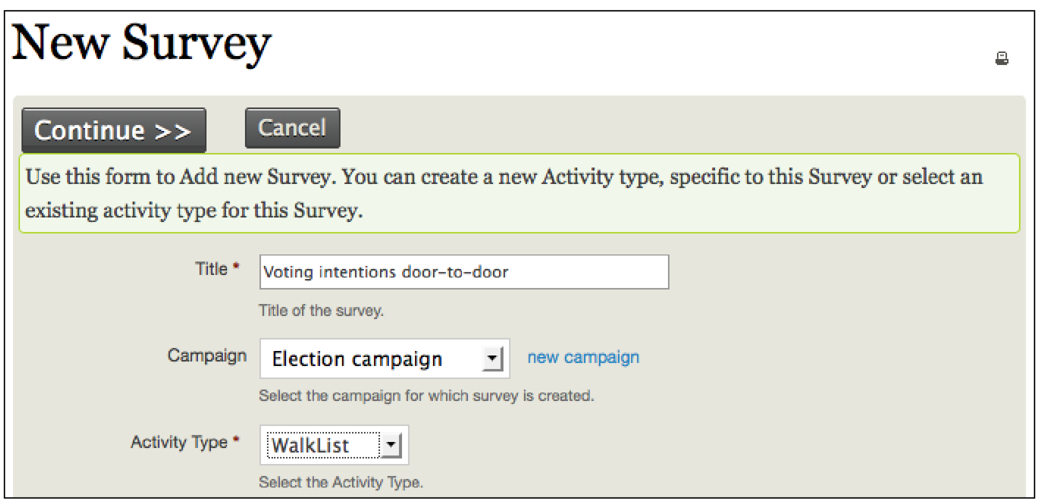
We need to set up some custom fields to hold our survey questions and then link them to a survey. In this example, we will create a single question and distribute it as a door-to-door survey managed by our volunteers.

1. Navigate to **Campaigns** | **New Campaign**. Name the campaign `Election Campaign`.
2. Navigate to **Administer** | **Custom Fields** and create a custom field set. Name it `Survey Questions`.
3. Set **Used For** to **Activities**.

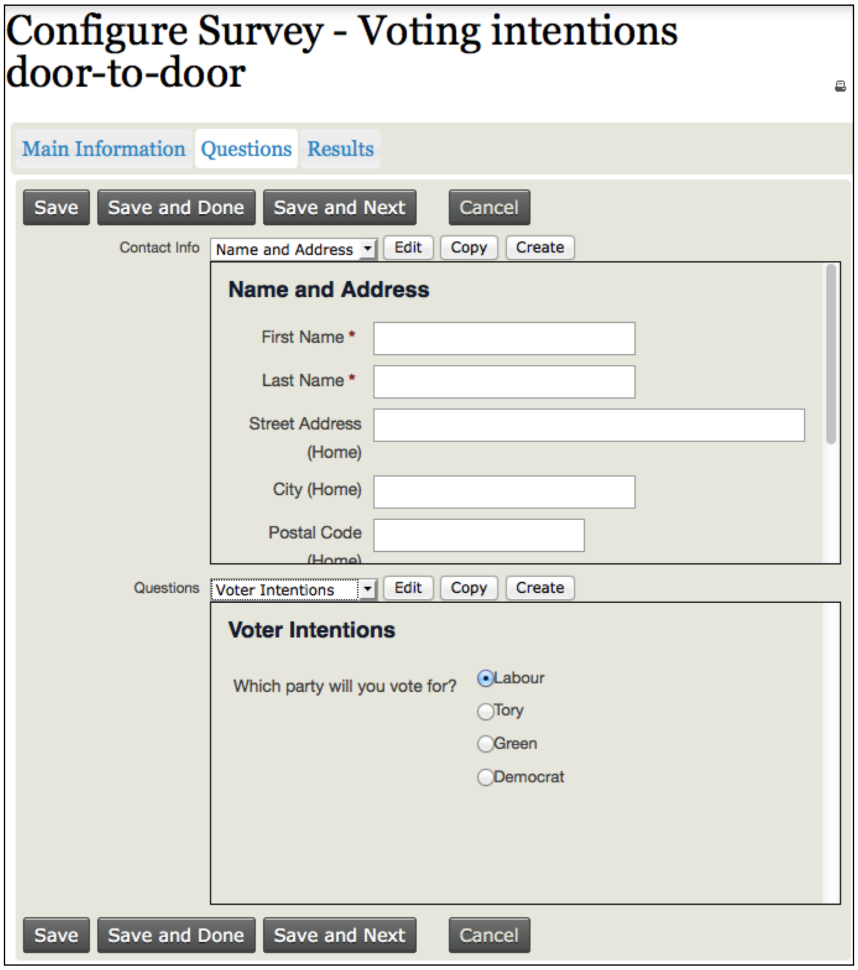
4. Set **Activities Options** to **WalkList**.



5. Add a custom alphanumeric radio button field. Name it `Which party will you vote for?`.
6. Add choices for **Labour**, **Tory**, **Green**, and **Democrat**.
7. Navigate to **Administer** | **Customize Data and Screens** and create a new profile. Name it `Voting Intentions`.
8. Add the **Survey Questions** custom field set to the profile.
9. Navigate to **Campaigns** | **New Survey** and create a new survey. Name the survey `Voter intentions door-to-door`.
10. Set the **Campaign** field to **Election campaign**.
11. Set **Activity Type** to **WalkList**.
12. Set **Maximum contacts reserved at one time** to `10`.



13. Click on **Save and Next**.
14. Add the **Name and Address** profile to the **Contact Info** section of the survey.
15. Add the **Voter Intentions** profile to the **Questions** section of the survey.



16. Click on **Save and Next**.
17. Enter the following result options: **Not in**, **Changed address**, **Come back later**, and **Deceased**.
18. Set **Report Title** to **Voter Intention Survey**.
19. Save the survey.
20. Navigate to **Contacts** | **New Group** and add a group. Name it `Canvassers`.
21. Add some contacts to the `Canvassers` group for testing purposes.

22. Navigate to **Contacts** | **New Group** and add a group. Name it `Voters`.
23. Add some contacts to the `Voters` group for testing purposes.
24. Navigate to **Campaigns** | **Reserve Respondents**.
25. Select the `Canvassers` group and click on **Search.**
26. Select up to 10 contacts from the list. Select **Reserve respondents** from the **Action** drop-down list and click on **Go**.
27. Navigate to **Campaigns** | **Campaign reports**.
28. Select the **Voter Intention Result** report.
29. Print out the report.

Voter Intention Result.

April 29th, 2013 10:06 PM

Survey Is Voting intentions door-to-door

Respondent Status Is equal to Reserved

Respondent Name	Street Number	Street Name	Street Unit	Survey Result	Which party will you vote for?
Ivanov, Jackson				ni | mo | cbl | de	lab | tor | gre | dem
Ivanov, Andrew	829	Caulder		ni | mo | cbl | de	lab | tor | gre | dem
Ivanov, Herminia	931	Lincoln		ni | mo | cbl | de	lab | tor | gre | dem
Ivanov, Alida	54	Main		ni | mo | cbl | de	lab | tor | gre | dem
Ivanov, Allen	277	Second		ni | mo | cbl | de	lab | tor | gre | dem
González, Ivey	299	Second		ni | mo | cbl | de	lab | tor | gre | dem
Ivanov, Elizabeth	757	Van Ness		ni | mo | cbl | de	lab | tor | gre | dem
Ivanov, Delana	515	Woodbridge		ni | mo | cbl | de	lab | tor | gre | dem

Row(s) Listed 8

The report printout includes pages for canvassers to record the survey.

How it works...

We created a set of custom fields that are used for the walklist activity and added them to a profile for walklist activities. We then created a new survey and made it a walklist survey. We then added in a profile that contained the names and addresses of contacts and the profile we created for the walklist.

We then added some options for the survey result such as a change of address. We then chose a group of respondents and printed off the walklist. The names and addresses profile populates the walklist with address data and the walklist profile populates it with the question.

See also

Find out more about CiviSurvey at `http://book.civicrm.org/user/current/survey/setup/`.

Recording survey results

CiviCRM surveys are conducted offline, and so results must be entered *manually*. This recipe shows you how to manually enter results.

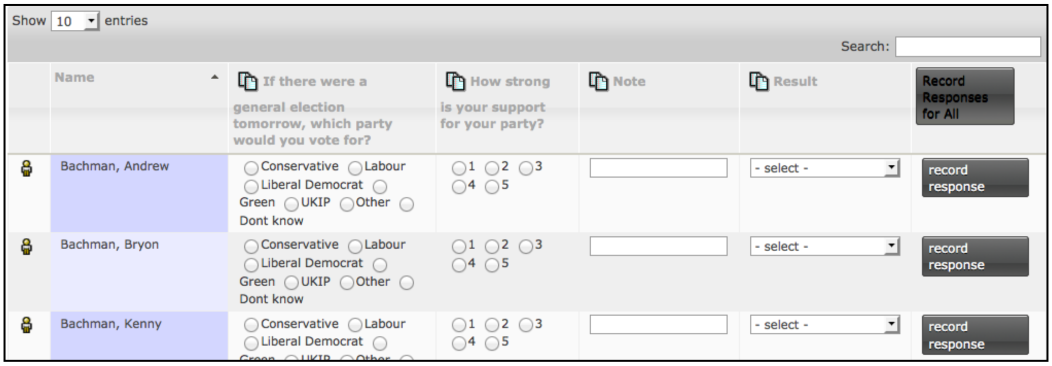
How to do it...

CiviSurvey has a well-developed interface for recording offline survey results. We will use this in our recipe as follows:

1. Navigate to **Campaigns** | **Dashboard** and click on the **Surveys** button.
2. Select a survey. In the preceding recipe, we created a survey called `Voter Intentions`.
3. In the **More** menu, select **Interview Respondents**. You are now presented with a list of possible respondents for the survey.

How it works...

At the top of the screen there is an interface to order the results. So, if a volunteer comes in with the results of a walklist survey, you can quickly navigate to the related respondents. You also have a screen where you can enter the survey results. Note that the head of the columns allows you to downfill the results.



Using get out the vote effectively

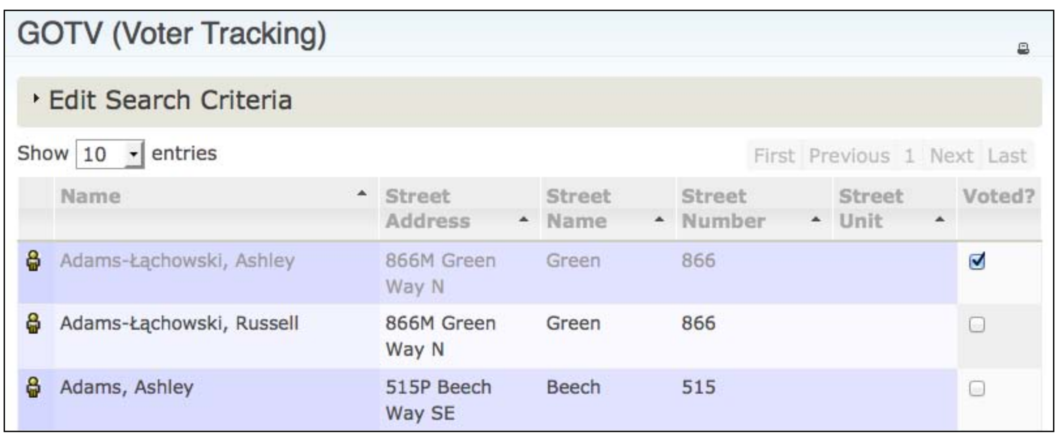
Get out the vote (GOTV) is a handy tool to very quickly check off people you know have voted in an election. For example, imagine you are running a local election campaign for your party. You would want to maximize your vote, and the best way to do this is to know if your supporters have voted or not. This recipe shows you how to create reports that show how well you and your opponents are doing.

How to do it...

The CiviCRM database needs to hold party affiliation data for each individual that is going to vote for your party. The best way to hold this data is not within a survey, but in a set of custom fields organized into an individual profile. Most local political organizations have gathered this valuable data over the years, and it can be imported into CiviCRM to form the basis of your election campaign. In this recipe, we will create a set of custom fields and add them to a profile. We will then use the profile in a batch update for all the individual contacts in the sample data that is available when CiviCRM is first installed. We will then create a survey to contain our voter list and finally use a neat survey trick to record their vote.

1. Navigate to **Administer** | **Customize Data and Screens** | **Custom Fields**. Add a new custom field set. Name it `Voter allegiance`.
2. Set **User For** to **Contacts**.
3. Add a **checkbox custom field**. Name it `Party`. Add options for **Labor**, **Tory**, **Green**, and **Democrat**.
4. Navigate to **Administer** | **Customize Data and Screens** | **Profile** and add a new profile. Name it `Party allegiance`.
5. Add in the `Party` custom field and save the profile.

6. Navigate to **Search** | **Find Contacts**. Search for individual contacts.
7. There should be 161 records. Select 30 or so records.
8. Select **Batch Update via profile** from the **Action** drop-down list and click on **Go**.
9. Select the `Party allegiance` profile and click on **Continue**.
10. Enter a value for the first contact, for example, `Green`. At the top of the **Party** column, click on the downfill icon so that all the contacts have the `Green` value. Click on **Continue**.
11. Navigate to **Advanced Search**. In the custom fields section, select the **Green** option. Add the result set of contacts to a new smart group named **Green**. You now have a smart group that contains contacts who you know would vote Green.
12. Navigate to **Campaigns** | **New survey**.
13. Name the survey `Have you voted?`. You do not need to fill in any other details or add any other options.
14. Navigate to **Campaigns** | **Reserve Respondents**.
15. Select the `Have you voted?` survey and select the **Green** group. Click on **Search**.
16. Select all the contacts and click on **Reserve**.
17. Navigate to **Campaigns** | **GOTV**.
18. Select the **Have you voted?** survey and click on **Search**.
19. In the last column you can click on the checkbox for each contact you know has voted.



How it works...

The `Have you voted?` survey is just a container that holds the respondents that we are interested in. GOTV provides displays for the respondents and 'interviews them' when the **Voted?** checkbox is selected. This removes them as a respondent and this is used to show they have voted.

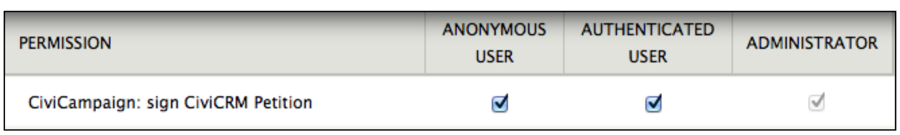
Using petitions effectively

CiviCRM Petition allows you to gather data online. You can use the data simply as survey data, or you can use it to demonstrate support for your cause in the form of a petition. CiviCRM Petition allows you to store, enrich, and leverage the data you hold and that alone gives it a huge advantage over external petitioning applications. This recipe shows you how to set up a petition.

How to do it...

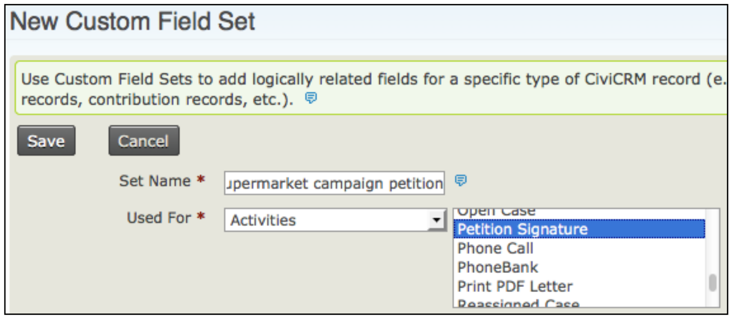
We will need to set up some custom fields to hold our petition question and put this into a profile. We will also need to create another profile to hold details of the person completing the petition. We then combine these two profiles to create the petition itself.

1. Navigate to **Campaigns** | **New Campaign** and set the title to `Stop the Supermarket`.
2. Navigate to **People** | **Permissions** and add the CiviCRM Petition permission for anonymous and authenticated users.

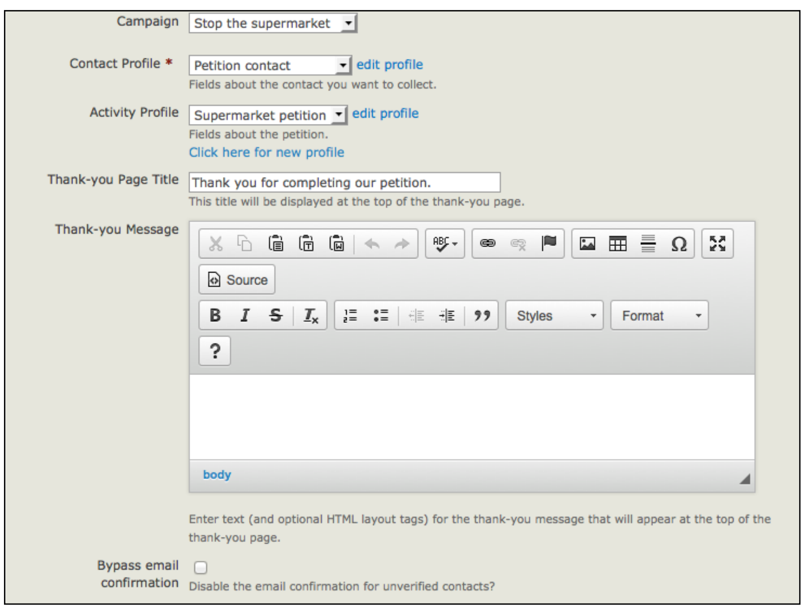


3. Navigate to **Administer Customize Data and Screens** | **Profiles** and create a new profile. Name it `Petition contact`.
4. Add first name, last name, and e-mail address fields.
5. In **Advanced Settings**, set **What to do upon duplicate match** to **Allow duplicate contact to be created**.
6. Navigate to **Administer Customize Data and Screens** | **Custom Fields** and add a new field set. Name it `Supermarket campaign petition`.
7. Set **Used For** to **Activities**.

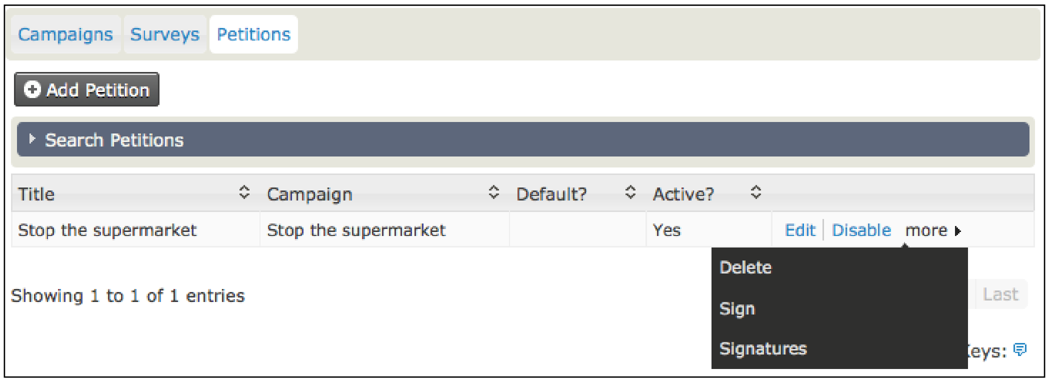
8. Select **Petition Signature** in the **Activities** options.



9. Add a custom checkbox field to the set. Name it `Do you want a new supermarket in town?` and set **checkbox options** for **Yes** and **No**.
10. Navigate to **Administer Customize Data and Screens** | **Profiles** and create a new profile. Name the profile `Supermarket petition`.
11. Add in the supermarket question.
12. In **Advanced Settings**, set **What to do upon duplicate match** to **Allow duplicate contact to be created**.
13. Navigate to **Campaigns** | **New Petition**.
14. Set the **Campaign** to **Stop the supermarket**.
15. Set the **Contact Profile** to **Petition contact**.
16. Set the **Activity Profile** to **Supermarket petition**.



17. Navigate to **Campaign** | **Dashboard** | **Petition**.
18. Select the **Stop the supermarket** petition.
19. In the **More** menu, select **Sign**. This provides the URL for the petition.
20. In the **More** menu, select **Signatures**. This provides a report on petition signatures.



See also

Find out more about CiviCRM Petition at `http://book.civicrm.org/user/current/petition/what-is-civipetition/`.

10
Working with CiviMember

In this chapter, we will cover:

- Creating a membership directory using Drupal Views
- Updating memberships by bulk data entry
- Effective membership communications using reminders
- Using price sets for complex memberships
- Using CiviCase for membership induction

Introduction

This chapter explores CiviMember, a CiviCRM component used for membership management. We will look at a popular requirement, displaying a membership directory, and then explore linking common membership tasks with other CiviCRM components.

Creating a membership directory using Drupal Views

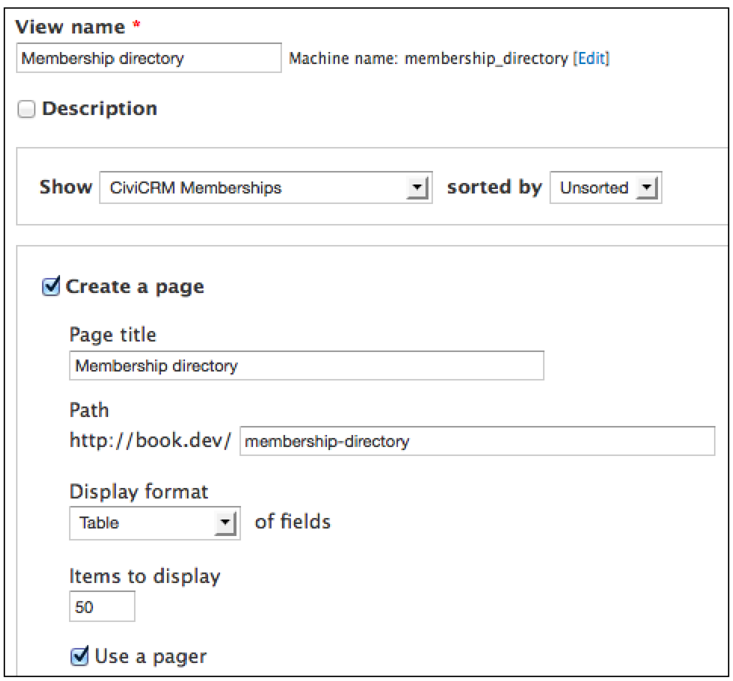
CiviCRM uses **profiles** to display contacts, and in most situations this is an excellent solution to displaying and sorting your membership contacts.

However, displaying membership contacts can be a problem in CiviCRM. For example, you may have a membership setup that allows both individuals and organizations to join. CiviCRM does not allow you to have a profile that combines both individual and organizational data. So, you cannot use the profile solution for your directory.

How to do it...

Using this recipe, we can use Drupal's **Views** module to create a membership directory that includes organizations and individuals.

1. Download, install, and enable Drupal's `Views` module, available at `http://drupal.org/project/views`.
2. In Drupal, navigate to **Admin** | **Structure** | **Views** and add a new view.
3. Set the view to show **CiviCRM Memberships**.
4. Name the view **Membership directory**.
5. Set the view to display a page and give the page a URL.
6. In the **FORMAT** section, set the **Display format** field to **Table**.



7. In the **FIELDS** section, add the following fields:

   CiviCRM Contacts: Display Name (Display Name)

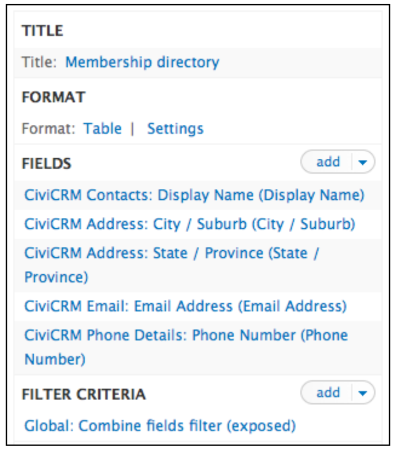
   CiviCRM Address: City / Suburb (City / Suburb)

   CiviCRM Address: State / Province (State / Province)

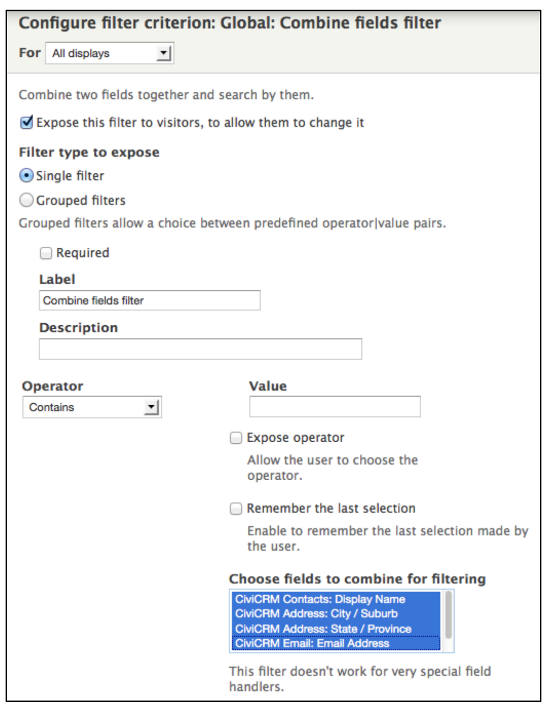
   CiviCRM Email: Email address (Email Address)

   CiviCRM Phone details: Phone Number (Phone Number)

8. In the **Filters** section, add a global filter.



9. In **Global filter configuration**, set the filter to **Exposed**.
10. Set **Operator** to **Combined**.
11. Set **Choose fields to combine for filtering** to all the fields added earlier.
12. Save the filter.



13. In the **Advanced** section, set **Query options** to **Distinct**.
14. **Preview** the view.

Membership directory

Combine fields filter

Apply

Display Name	City / Suburb	State / Province	Email Address	Phone Number
Mr Sandy J Reynolds Sr	North Waterford	Maine		
Mrs Amar M Smith Jr	Arlington	Tennessee		94927088
Mr Peter J Smith Jr	West Hyannisport	Massachusetts		65041724
Ms Sheila P Roberts Jr	Trenton	Michigan	robertssheila@redimail.net.in	
Mrs Milan A Smith Sr	Saint Stephens Church	Virginia		14531182
Mr John K Yadav Jr	Allgood	Alabama	yadavjohn@npo.edu.in	

How it works...

The **CiviCRM Contact Display Name** field is available for individuals and organizations. So, all membership names will be displayed.

The global filter combines all the fields together so that you can search across them.

Setting **Query options** to **Distinct** reduces the duplication of entries.

See also

- Find out more about reducing duplicates at `http://dropbucket.org/node/153`

Updating memberships by bulk data entry

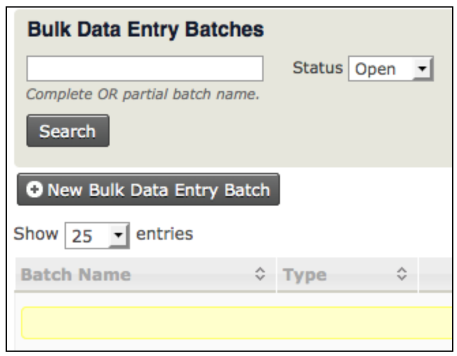
You may run a membership organization that takes membership payments offline.

For example, in the UK, many organizations process membership payments using the **Direct Debit** payment system. Entering these payments is time-consuming, and hitherto has required custom import scripts for large sets of payments.

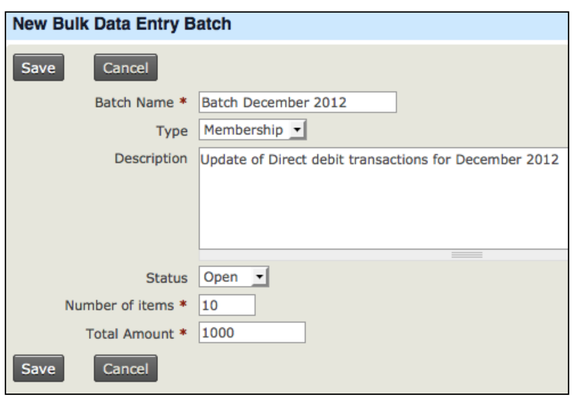
How to do it...

In this recipe, we will use a new batch entry system that enables you to automate membership payments.

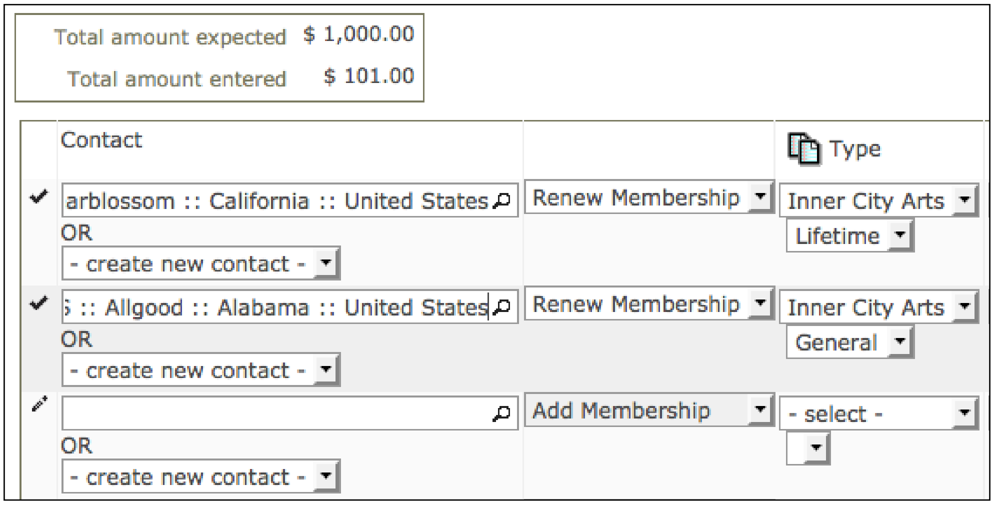
1. First, you will need to prepare your data for entry. You will need the contact name and the amount paid. You also need to know how many entries you are going to make and the total payment amount.
2. Navigate to **Memberships** | **Bulk Data Entry** and create a new bulk data entry batch.



3. Name the batch.
4. Set the **Number of items** field to the number of entries you are going to make.
5. Set the **Total Amount** field to the total amount of money being paid.



6. Save the batch.



How it works...

As you enter the data into each row, CiviCRM checks the membership details for each contact and makes the second column **Add Membership** or **Renew Membership**, depending on what membership data is held.

It also auto-enters the **Amount** column with the membership fee (not shown).

It also alters the figure entered in **Total Amount** at the top of the form entry.

The interface provides many other fields for data entry. For renewals, these do not need to be completed, as CiviCRM does the calculations for you. For new members, you will have to enter the data manually.

At the top of some columns, you can click on the column header to fill the values for each column.

Please note that caution is needed here, particularly if you are dealing with mixed membership types.

CiviCRM allows you to save the batch and continue data entry or to validate and process the batch once you have finished.

Effective membership communications using reminders

In CiviCRM 4.3, the reminder system for membership renewals has been moved from membership management to **Scheduled Reminders**. This gives us the opportunity to use these reminders in our member workflow. For example, you may have a process for membership induction. This may involve sending out information at fixed intervals.

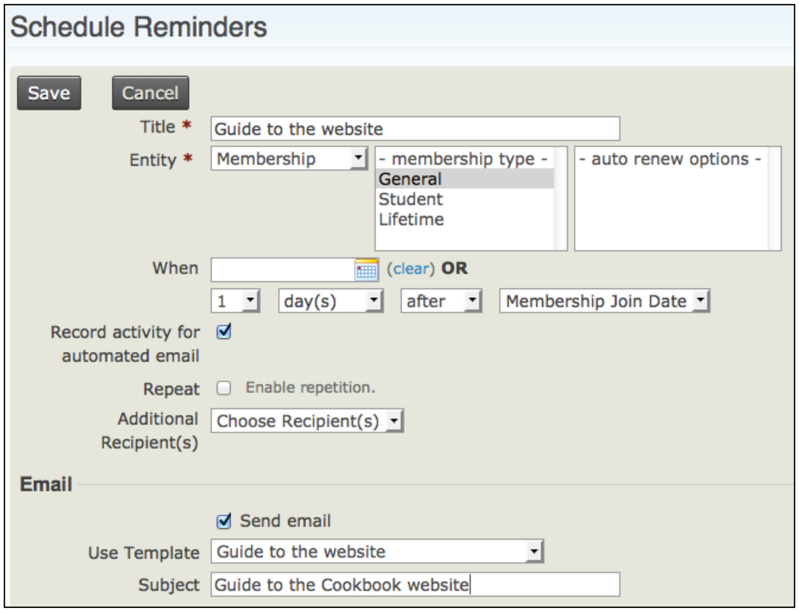
Examples of this include a guide to the organization's website, a personalized message from the head of the organization, or a request to share some information to provide a better service for the member.

How to do it...

We will create two mail templates. We will use one to send out a guide to the website, and the other to send out a guide to local involvement. We will then schedule these mailings using **Scheduled Reminders**.

1. Navigate to **Administer** | **CiviMail** | **Message Templates**. Add a new template.
2. Set **Title** to **Guide to the website**.
3. Set the **Subject** to **Guide to the Cookbook website**. Substitute **Cookbook** with the name of your organization.
4. In the **plain text** field, enter the full details of your website guide. Insert tokens to personalize the message.
5. Copy the plain text entry into the **HTML** field and apply HTML formatting to the message.
6. Save the template.
7. Create a new template for the guide to local involvement and repeat the process.
8. Navigate to **Administer** | **Communications** | **Scheduled Reminders** and create a new reminder.
9. Set the **Title** field to **Guide to the website**.
10. Set the **Entity** field to **Membership**.
11. Set the **membership type** field to **General**.
12. Schedule the reminder for **1 day(s) after Membership Join Date**.
13. In the **Email** section, check the **Send email** checkbox.

14. Set **Use Template** to **Guide to the website**.



15. Set up another **Scheduled Reminder** instance for the second mailing.
16. Navigate to **Administer** | **System Settings** | **Scheduled Jobs** and enable the **Send Scheduled Reminders** job.

See also

- The *Using Scheduled Reminders for activities* recipe in *Chapter 1*, *Setting Up CiviCRM*
- The *Creating mail templates for CiviMail* and *Using tokens in templates* recipes in *Chapter 5*, *Managing Communications*
- Find out more about Scheduled Reminders at `http://book.civicrm.org/user/current/email/scheduled-reminders/`

Using price sets for complex memberships

Membership **price sets** are used in situations where you have a national membership plus regional sub-memberships. For example, you may run a national cycling association with chapters in each state.

How to do it...

We can create a CiviCRM price set and link it to a contributions page for online membership. The price set allows us to apply complex membership options.

1. Navigate to **Contacts** | **New organization**. Create a contact called **National Cycling Association**. Click on **Save and New**.
2. Add the following organizations. These will be NCA Chapters:

   NCA Delaware

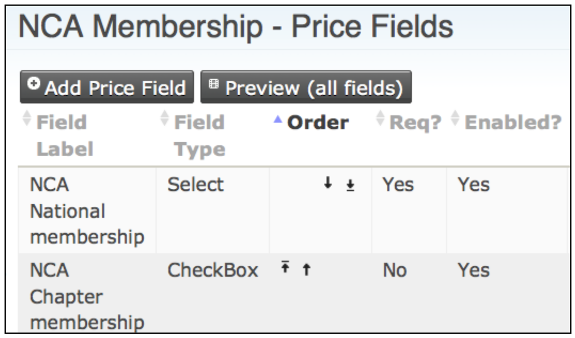
   NCA California

   NCA Oregon

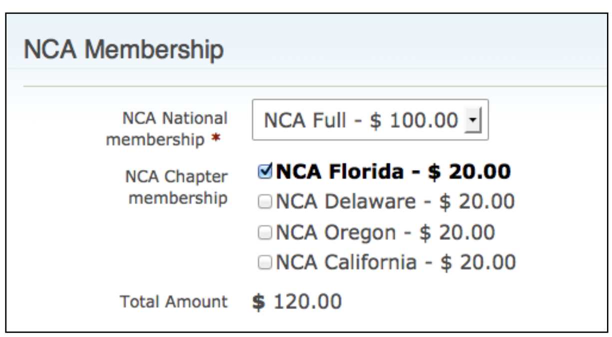
   NCA Florida

3. Navigate to **Administer** | **CiviMember** | **Membership Types** and add a membership.
4. Set **Name** to **NCA Full**.
5. Set **Membership Organization** to **National Cycling Association**.
6. Set **Minimum Fee** to **$100**.
7. Set **Financial Type** to **Member Dues**.
8. Set **Duration** to **1 year rolling**.
9. Click on **Save and New**.
10. Create four more membership types, one for each NCA Chapter. Set the fee for each chapter to **$20**.
11. Navigate to **Administer** | **CiviMember** | **New Price Set**.
12. Set **Name** to **NCA Membership**.
13. Set **Used For** to **Membership**.
14. Set **Default Financial Type** to **Member Dues**.
15. Save the price set and add a field.
16. Set **field label** to **NCA National Membership**.
17. Set **Input Field Type** to **Select**.
18. In **Membership Options**, set **Membership Type** to **NCA Full**.
19. Select **Required**.
20. Click on **Save and New**.
21. Set **field label** to **NCA Chapter Membership**.
22. Set **Input Field Type** to **Checkbox**.

23. In **Membership Options**, set **Membership Type** to **NCA Florida**.
24. Add in the other three chapters.
25. Save the price field.



26. Navigate to **Contributions** | **New Contribution Page**.
27. Set **Title** to **NCA Membership**.
28. Set **Financial Type** to **Member Dues**. Click on **Save and Next**.
29. Set **Processor** to **Test Payment Processor**.
30. Uncheck **Contribution Amounts section enabled**.
31. Click on **Save and Next**.
32. Check **Membership Section enabled**.
33. Set **Title – New Membership** to **NCA Membership**.
34. Set **Title – Renewals** to **NCA Membership renewal**.
35. Set **Membership Price Set** to **NCA Membership**.
36. Click on **Save and Done**.
37. Locate the **NCA Membership** entry on the **Contribution** listing. On the **More** link, select **Test-Drive**.



How it works...

The **National Membership price** field is set to **required**. This means members have to join the national organization. NCA Chapters are not required. So, users can join none, one, or more of the chapters. CiviCRM recalculates the cost according to the users' choices.

Using CiviCase for membership induction

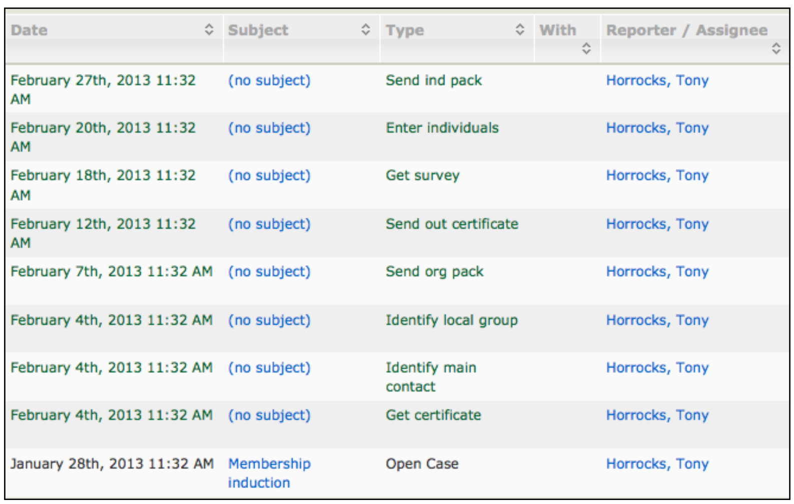
In this recipe, we will use CiviCase to provide a membership communication scheme for complex organizations. For example, you may have a large organization where different staff members, volunteers, and representatives have responsibility for the different stages in a membership induction program. There might be a membership officer, membership office staff, regional office staff, and specialist office staff. CiviCase allows us to allocate activities to everyone involved in this process.

How to do it...

We will create a CiviCase to handle the activities associated with a membership induction program. In our imaginary scenario, a new member induction scheme might be implemented as follows:

Relative time	Activity	Detail	Assignee
7 days	Get certificate	Get a membership certificate printed and signed by the President	Membership Officer
7 days	Identify the main contact	Contact the applicant and update our records to get the main contact details for the organization that joined	Membership Officer
7 days	Identify the local group	From the application identify the local group for the organization. Schedule a contact from the local group leader.	Local group leader
10 days	Send out the organization pack	Send out the organization pack personalized with details collected in first week	Membership Officer
14 days	Send out the certificate	Send the certificate out with a note from the President	Membership Officer
20 days	Send out the survey	Send out the online survey that collects more data about the organization	Membership Officer
27 days	Collate information and contact individuals	Collate the survey information and contact other individuals within the member organization.	Membership Officer

1. Navigate to **Administer** | **System Settings** | **CiviCRM Components** and enable CiviCase.
2. Navigate to **Administer** | **Customize Data and Screens** | **Activity Types**. Create activity types to cater to all the activities involved in your membership program.
3. Set **User For** to **CiviCase** for each activity.
4. Navigate to **Customize Data and Screens** | **Relationships**. Create relationships for the contacts that will be involved in the membership induction process, for example, **Membership Officer**.
5. Create the XML file that will create the activities, the activity schedules, and the necessary relationships when a new case is added to a contact.
6. Navigate to **Administer** | **CiviCase** | **Case Types** and create a new CiviCase type that uses the XML file. Name it **Member Induction**.
7. When a new member joins, add **Member Induction Case** in the contact summary screen.



Date	Subject	Type	With	Reporter / Assignee
February 27th, 2013 11:32 AM	(no subject)	Send ind pack		Horrocks, Tony
February 20th, 2013 11:32 AM	(no subject)	Enter individuals		Horrocks, Tony
February 18th, 2013 11:32 AM	(no subject)	Get survey		Horrocks, Tony
February 12th, 2013 11:32 AM	(no subject)	Send out certificate		Horrocks, Tony
February 7th, 2013 11:32 AM	(no subject)	Send org pack		Horrocks, Tony
February 4th, 2013 11:32 AM	(no subject)	Identify local group		Horrocks, Tony
February 4th, 2013 11:32 AM	(no subject)	Identify main contact		Horrocks, Tony
February 4th, 2013 11:32 AM	(no subject)	Get certificate		Horrocks, Tony
January 28th, 2013 11:32 AM	Membership induction	Open Case		Horrocks, Tony

How it works...

CiviCase generates and schedules all the activities. All that remains is to allocate each activity to the appropriate person responsible.

See also

- The *Using Scheduled Reminders for activities* recipe in *Chapter 1*, *Setting Up CiviCRM*
- The *Adding custom fields* and *Creating new activities* recipes in *Chapter 1*, *Setting Up CiviCRM*

11
Developing for CiviCRM

In this chapter, we will cover:

- Setting up a local development environment
- Finding developer resources
- Exploring Drupal hooks
- Exploring the CiviCRM API
- Developing a CiviCRM Drupal module
- Exploring CiviCRM extension development using Civix

Introduction

This chapter looks at the software, skills, and resources you need to start developing CiviCRM in earnest. We will also cover developing a simple Drupal module and exploring the CiviCRM API.

Setting up a local development environment

A local development environment means installation of CiviCRM on your own computer so that you can do development without having to connect to a remote server on the Internet. We have seen in other recipes that having a local installation of CiviCRM makes things such as importing contacts faster and easier.

In this recipe—more of a set of guidelines—we explore how to set up such an environment.

How to do it...

We need five ingredients for our local installation to work:

- A web server: Apache will serve our CiviCRM pages
- A database: MySQL will store our CiviCRM data
- A PHP interpreter: This works with Apache and MySQL to interpret the PHP code we write and sends us back pages of HTML
- A good text editor
- A good MySQL manager

Our own computers will have an operating system such as Linux, Windows, or Mac OS, and you may have come across acronyms such as a **LAMP** environment (Linux, Apache, MySQL, PHP). So, we also have **WAMP** for Windows, and **MAMP** for Mac OS.

By far the easiest way to create these environments is to download an application called `XAMPP` (`http://www.apachefriends.org/en/xampp.html`). This can be used for Windows, Linux, and Mac OS operating systems.

This runs your local environment as an application. This means when you start `XAMPP`, Apache, MySQL, and PHP all start up for you. It is easy to download and install, and it is free.

For Mac OS, there is an alternative application `MAMP`, available at `http://www.mamp.info`.

Each of these applications have their own configurations, and it is not possible here to provide installation details. Once they are set up, they will do the job of serving our CiviCRM installation.

We now need to have some programs to help us develop.

There are all sorts of text editors available for Windows or Mac OS. The better ones will have syntax highlighted to show you where you are making mistakes in your code. Some will also have code-completion shortcuts, which are also useful.

Text editors for Windows include:

- UltraEdit (`http://www.ultraedit.com/`)
- Komodo Edit (`http://www.activestate.com/komodo-edit`)
- Aptana (`http://www.aptana.com/`)
- PSPad (`http://www.pspad.com/en/`)
- Notepad++ (`http://notepad-plus-plus.org/`)

Text editors for Mac OS include:

- Komodo Edit (`http://www.activestate.com/komodo-edit`)
- Aptana (`http://www.aptana.com/`)
- BBEdit (`http://www.barebones.com`)
- Textmate (`http://macromates.com/`)
- Coda (`https://panic.com/coda/`)

MySQL managers, XAMPP and MAMP, come with PHPmyAdmin, which is good enough for most purposes.

Once you have these tools installed, you are good to go for doing some development.

An **Integrated Development Environment** (**IDE**) provides a total environment for your development needs. These can be quite complex to use as they normally cater to several development environments, such as Java, C++, and so on. Popular IDE software includes:

- Zend Studio (`http://www.zend.com/en/products/studio/`)
- Netbeans (`http://netbeans.org/`)
- Aptana (`http://www.aptana.com/`)
- Komodo (`http://www.activestate.com/komodo-ide`)

Finding developer resources

CiviCRM has excellent documentation for the basic needs of most people who want to create a good CiviCRM installation.

As tasks and requirements get more and more specific, guidance becomes a little bit trickier to find. Luckily, CiviCRM has a thriving online community that can help through forums as well as an IRC channel where you can communicate with the core team. Support does not come much better than that.

How to do it...

In this recipe, we'll look at what resources are available to beginner and experienced developers.

The following table shows the CiviCRM developer continuum. Progressively more skills, time, and commitment is required as the tasks become more and more complex.

Task	Skills needed	Resources
Configuring the CiviCRM core and its components	An understanding of CiviCRM	CiviCRM books, forums, blogs, and meetups

Task	Skills needed	Resources
Basic customization of report templates and search templates	Knowledge of template structures	CiviCRM books, forums, blogs, Wiki, and meetups
Advanced customization of reports	Knowledge of Smarty templates, SQL, and PHP	CiviCRM developer resources and Wiki
CMS-specific modules	Knowledge of CMS API, OOP, and PHP	CiviCRM developer resources, CiviCRM API Explorer, CiviCRM module examples, Wiki, and IRC
Agnostic modules (extensions)	High level knowledge of CiviCRM architecture, OOP, and PHP	CiviCRM developer resources, Wiki, CiviCRM API Explorer, IRC, and code sprints

You can find basic guidance using the following resources:

- Find the latest CiviCRM announcements at `http://civicrm.org/blog-news`
- Read the authoritative guide for administrators and users of CiviCRM at `http://book.civicrm.org/user/` `http://forum.civicrm.org/`
- Read the authoritative developer guide at `http://book.civicrm.org/developer/`

You can also find some advanced guidance. Here are some online resources that provide advanced guidance:

- The guide to using hooks in CMS-specific modules at `http://wiki.civicrm.org/confluence/display/CRMDOC42/Hook+Reference`
- The guide for making and customizing templates at `http://book.civicrm.org/developer/current/techniques/templates/`
- The guide to the CiviCRM API at `http://book.civicrm.org/developer/current/techniques/api/`
- The guide to advanced importing techniques at `http://book.civicrm.org/developer/current/techniques/imports/`
- Guidance for creating extensions at `http://wiki.civicrm.org/confluence/display/CRMDOC42/Create+a+Module+Extension`

Exploring Drupal hooks

You can customize the behavior of CiviCRM by developing a module that takes advantage of the **hook system**.

CiviCRM is written in a scripting language called PHP, and you use PHP to create all the functions that make CiviCRM work. Some of these functions are exposed to your CMS—in this case, Drupal. For example, there is a function called `civicrm_postProcess` that runs every time a form is submitted.

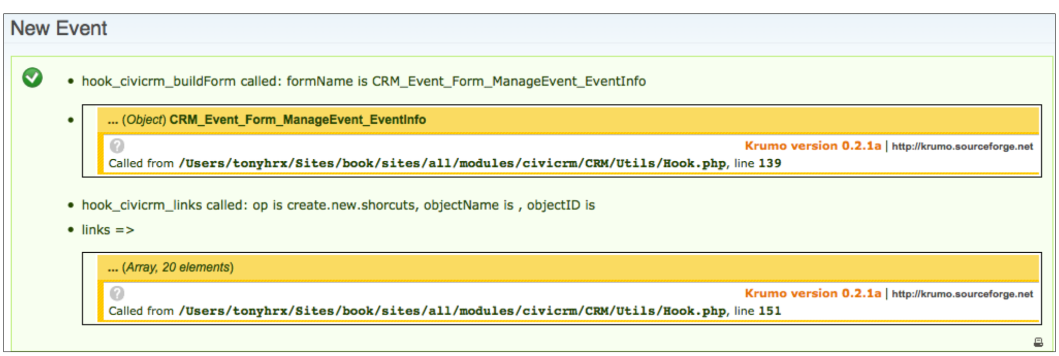
You can copy this function and customize it. Each time `civicrm_postProcess` runs, CiviCRM checks whether there is a customized version of the function, and if there is, it runs that instead. Not all CiviCRM functions are exposed in this way. If they are, they are called **hooks**.

This recipe shows you how to explore these hooks in Drupal using the `civicrm_developer` module.

How to do it...

Hooks allow the CMS to extend the functionality of CiviCRM without having to alter any core CiviCRM files. Here, we will add a module that allows us to see which hooks are available to us:

1. Install and enable the Drupal Devel module, available at `http://drupal.org/project/devel`.
2. Navigate to `https://github.com/eileenmcnaughton/civicrm_developer`.
3. Download the ZIP file for the module and expand it. It expands as a directory called `civicrm_developer-master`.
4. Rename the directory `civicrm_developer`.
5. Install and enable the module.
6. Now visit CiviCRM pages within your site.



How it works...

The module shows all the Drupal hooks that can be called and allows the developer to examine objects and their values.

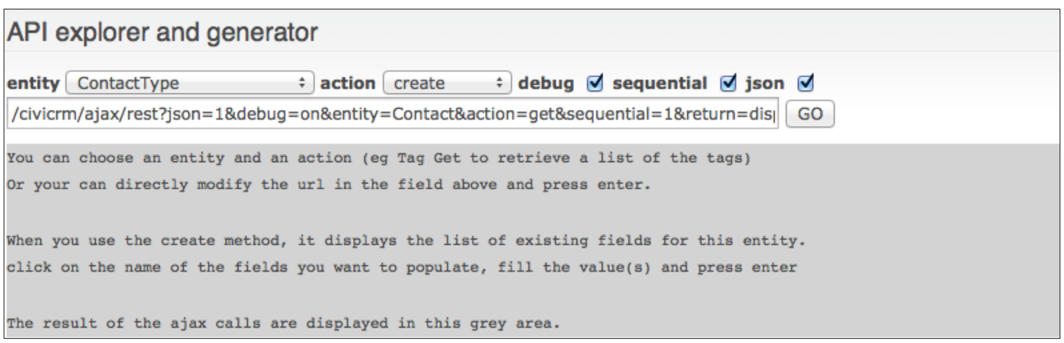
Exploring the CiviCRM API

Writing code is laborious. It takes a lot of skill, a lot of trial and error, and most importantly, a lot of time and cost. Rather than write the code from scratch, you can use prewritten functions that do the work. These functions, when gathered together, are called an **Application Programming Interface** (**API**).

How to do it...

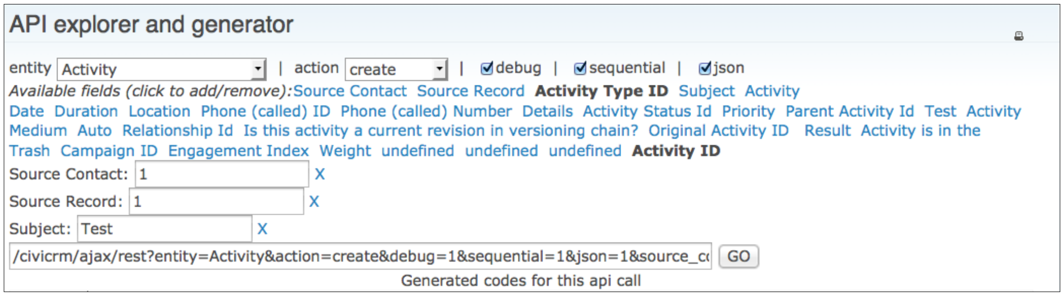
CiviCRM has an API. This recipe shows you how to explore it:

1. Navigate to `http://<mycivicrm.com>/civicrm/ajax/doc/api#explorer`. Substitute `<mycivicrm.com>` with your own domain.



2. Select the **Activity** entity and perform a **create action** on it.
3. Click on the **Source Contact** field, the **Source Record** field, and the **Subject** field. The fields are added dynamically to the API Explorer.

4. Set **Source Contact** to **1**, **Source Record** to **1**, and **Subject** to **Test**. The values you put in each field are called **parameters**. Click on **Go**.



How it works...

CiviCRM provides us with ready-made code examples with the fields and parameters added.

URL	ajax query |REST query.
smarty	`smarty uses only 'get' actions`

php	`$params = array(`
	`  'version' => 3,`
	`  'sequential' => 1,`
	`  'source_contact_id' => 1,`
	`  'source_record_id' => 1,`
	`  'activity_subject' => 'Test',`
	`);`
	`$result = civicrm_api('Activity', 'create', $params);`

javascript	`CRM.api('Activity', 'create', {'sequential': 1, 'source_contact_id': 1, 'source_record_id': 1, 'activity_subject': 'Test'},`
	`  {success: function(data) {`
	`      cj.each(data, function(key, value) {// do something });`
	`    }`
	`  }`
	`);`

CiviCRM also provides error checking on-the-fly when you make mistakes in your API exploration.

```
{
        "fields":["one of (activity_name, activity_type_id, activity_label)"],
        "error_code":"mandatory_missing",
        "entity":"Activity",
        "action":"create",
        "is_error":1,
        "error_message":"Mandatory key(s) missing from params array: one of (activity_name, activity_type_id, activity_label
        "trace":"#0 \/home\/entity\/public_html\/sites\/all\/modules\/civicrm\/api\/v3\/utils.php(66): civicrm_api3_verify_m
}
```

> The API really does execute on your data, so any API exploration is best done locally on sample data.

There's more...

Navigate to http://<mycivicrm.com>/civicrm/api/doc. Substitute <mycivicrm.com> with your own domain.

Click on **Activity**. You will get an expanded list of the available fields and the sort of data that CiviCRM expects for the Activity entity.


API Parameters

You can see the list of parameters for each entity by clicking on its name.
You can explore and try the api directly on your install.

Activity

Attribute	Name	type
source_contact_id	Source Contact	integer
source_record_id		integer
activity_type_id	Activity Type ID	integer
activity_date_time	Activity Date	date time

For example, **Source Contact ID** is expected to be an **integer**.

You can see that there are many CiviCRM entities that are exposed to the API, and it is not obvious what the API does for each one. For example, if you use the API call for an activity and fill in the correct parameters, what exactly does it do?

In your web server file system, navigate to /sites/all/modules/civicrm/api/v3/.

When a CiviCRM API runs, these are the PHP files that are called. So, for the Activity entity, there is a corresponding PHP file called activity.php that runs when the Activity API is called.

Open this file and look through the commented text at the start of the file; this explains what the file does. In this case, * creates or updates an activity See the example for usage. This is a good way to see what each API does.

See also

- Find out more about the CiviCRM API at http://book.civicrm.org/developer/current/techniques/api/

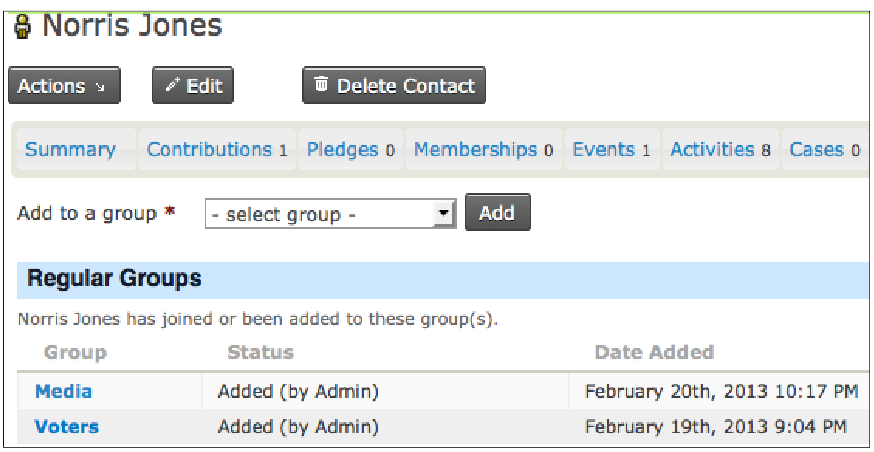
Developing a CiviCRM Drupal module

In this recipe, we will explore how a CiviCRM `Drupal` module works. The basic code for this module is already available online.

How to do it...

In this example, we have a contact that is in a specific group. We are going to fire off an e-mail if someone edits that contact.

1. Set up your own local development environment.
2. Download, install, and enable the Drupal Devel module, available at `http://drupal.org/project/devel`.
3. Navigate to `https://github.com/eileenmcnaughton/civicrm_developer`.
4. Download the ZIP file for the module and expand it. It expands as a directory called `civicrm_developer-master`.
5. Rename it `civicrm_developer`.
6. Install and enable the module.
7. In CiviCRM, add a contact to a group. Make a note of the contact ID; in this case, it was **51**.



Here, a contact was added to the **Media** group.

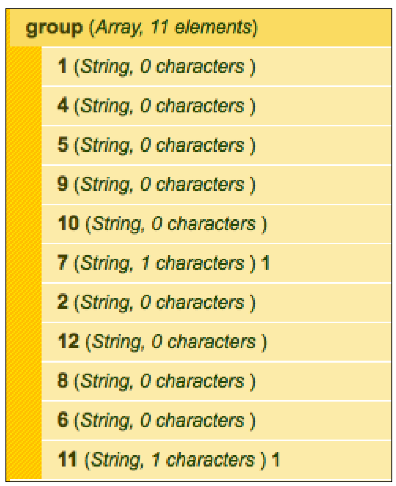
8. Navigate to **Contacts** | **Manage groups**. Note the ID of the **Media** group. In this case, the ID was **7**. In your own implementation, the ID is likely to be different.
9. Click on the **Edit** button on the contact summary screen, then click on the **Save** button.

10. Click on the yellow-colored **CiviCRM Development** display at the top of the page.

    You will see that the CiviCRM Developer lists the hooks that have been called (some of them are called several times):

    - hook_civicrm_buildForm
    - hook_civicrm_links
    - hook_civicrm_pre
    - hook_civicrm_post

11. Look closely at hook_civirm_pre. In one instance, it is called when the operation is Edit and when the object name is Individual. This means it is a good candidate for the module, as you want the module to do something when a contact is edited.
12. Click on the yellow bar where hook_civicrm_pre is shown and expand it.
13. Locate the **group** array and click on it to expand it.

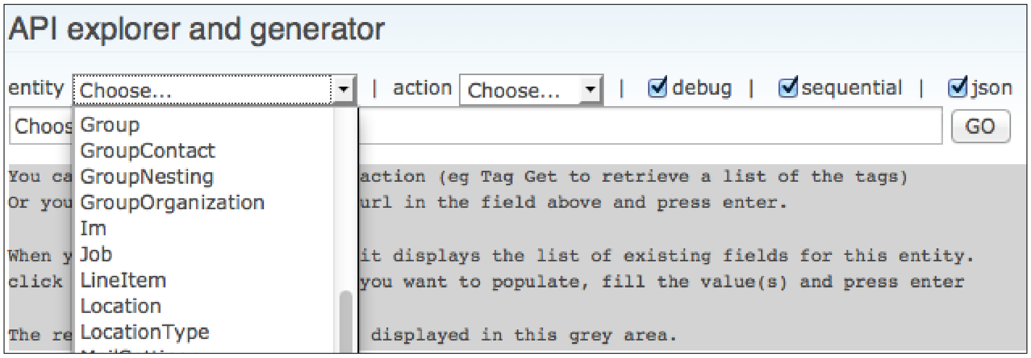


    Note there is a **1** against group ID **7** and group ID **11**. This means the contact is a member of both these groups. The group set up earlier had an ID of **7** (your own ID may be different). So, you can now tell that the contact is in the **Media** group.

    When hook_civicrm_pre is called, the module will test whether the contact is in group ID **7**. If it is, then it sends the e-mail. If it isn't, then it just carries on as usual.

14. Navigate to http://<mycivicrm.com>/civicrm/api/doc. Substitute <mycivicrm.com> with your own domain.

15. Click on **entity** and scroll down for group entities.



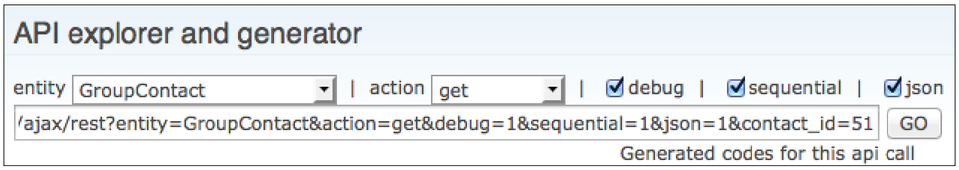
There are four entities for groups. You need to discover which one to use. The entity **GroupContact** looks like a good candidate.

16. On your web server, navigate to `/sites/all/modules/civicrm/api/v3/GroupContact.php` and open it.

```
/**
 * This API will give list of the groups for particular contact
 * Particualr status can be sent in params array
 * If no status mentioned in params, by default 'added' will be used
 * to fetch the records
 *
 * @param  array $params  name value pair of contact information
 * {@getfields GroupContact_get}
 *
 * @return  array  list of groups, given contact subsribed to
 */
function civicrm_api3_group_contact_get($params) {

  if (empty($params['contact_id'])) {
    if (empty($params['status'])) {
      //default to 'Added'
      $params['status'] = 'Added';
    }
    //ie. id passed in so we have to return something
    return _civicrm_api3_basic_get('CRM_Contact_BAO_GroupContact', $params);
  }
  $status = CRM_Utils_Array::value('status', $params, 'Added');

  $values = &CRM_Contact_BAO_GroupContact::getContactGroup($params['contact_id'], $status, NULL, FALSE, TRUE);
  return civicrm_api3_create_success($values, $params);
}
```

The code comments tell you that this API will give a list of groups for a particular contact. The code also shows what parameters the GroupContact API expects—`contact_id`.

17. Navigate to the API Explorer and select the **GroupContact** entity and the **get** action.
18. Add the contact ID parameter, `contact_id`, to the query field and give it the value of the contact ID that was edited earlier. In this case, it was **51**. Yours will be different. So, the added parameter code is **&contact_id=51**. Click on **Go**.



CiviCRM makes the API call and returns a result.


```
{
        "is_error":0,
        "undefined_fields":["contact_id"],
        "version":3,
        "count":2,
        "values":[{
                "id":"250",
                "group_id":"7",
                "title":"Media",
                "visibility":"User and User Admin Only",
                "is_hidden":"0",
                "in_date":"2013-02-20 22:17:13",
                "in_method":"Admin"
        },
        {
                "id":"145",
                "group_id":"11",
                "title":"Voters",
                "visibility":"User and User Admin Only",
                "is_hidden":"0",
                "in_date":"2013-02-19 21:04:40",
                "in_method":"Admin"
        }],
```


You can see that the contact **51** belongs to two groups, **7** and **11**. This is exactly what was expected. Your own results may be different as your contact ID and group ID will not be the same as in this recipe.

19. On your web server, navigate to `sites/all/modules`.
20. Create a new folder called `civicrm_custom`.

21. Using a text editor, create a file called `civicrm_custom.info` and enter the following code:

```
name = CiviCRM custom module
description = Provides custom functions
for CiviCRMcore = 7.x
package = civicrm
dependencies[] = civicrm
files[] = civicrm_custom.module
```

22. Create a file called `civicrm_custom.module` and add the following code:

```
<?php
/**
 * Implements hook_civicrm_pre().
 */
function civicrm_custom_civicrm_pre( $op,
  $objectName, $objectId ) {
  $theGroupId = 7;
  $emailRecipient = 'johndoe@example.org';
  if ($objectName == "Individual" && $op == "edit") {
    require_once 'api/v3/utils.php';
    require_once 'api/v3/GroupContact.php';
    $params = array('contact_id' => $objectId);
    $groups = civicrm_api3_group_contact_get($params);
    $found = false;
    foreach ($groups['values'] as $group) {
      if ($group['group_id'] == $theGroupId) {
        $found = true;
      }
    }
    if (!$found) {
      return;
    }
    $emailSubject = "Contact was edited";
    $emailBody = "Someone edited contactId $objectId\n";
    mail( $emailRecipient, $emailSubject, $emailBody );
  }
}
```

23. In Drupal, navigate to **Modules** and enable the **custom civicrm** module.
24. Navigate to CiviCRM and edit a contact that is not in the Media group.
25. Check the Media group to see if the contact was edited.

How it works...

The following line stores your target `GroupID`. Your own `GroupID` may be different:

```
$theGroupId = 51;
```

This stores the e-mail address for the person we wish to e-mail:

```
$emailRecipient = 'johndoe@example.org';
```

This if statement is used to check that we are actually acting on `Individual` and that we are in `Edit` mode:

```
if ($objectName == "Individual" && $op == "edit")
```

The API is dependent on `utils.php` in order to work:

```
require_once 'api/v3/utils.php';
```

This calls the `GroupContact.php` file:

```
require_once 'api/v3/GroupContact.php';
```

This passes the contact ID into `civicrm_api3_group_contact_get`:

```
$params = array('contact_id' => $objectId);
```

The function `civicrm_api3_group_contact_get($params)` is the API call to get the contact groups:

```
$groups = civicrm_api3_group_contact_get($params);
```

The result is placed in the variable `$groups`. A contact can belong to many groups, so the values in `$groups` will be stored in an array.

```
$found = false;
    foreach ($groups['values'] as $group) {
      if ($group['group_id'] == $theGroupId) {
        $found = true;
      }
    }
    if (!$found) {
      return;
    }
```

This code cycles through the array that is stored in `$groups`. It checks the first group in the array and stores that in `$group`. It then gets `group_id` from `$group` and checks if that is equal to `groupID` stored in `$theGroupId`. This is repeated for every item in the array. If there is a match, the value of `$found` is set to `TRUE`.

If there is no match, then nothing is returned.

```
$emailSubject = "Contact was edited";
$emailBody = "Someone edited contactId $objectId\n";
mail( $emailRecipient, $emailSubject, $emailBody );
```

If there is a match, then the e-mail is sent using the function mail.

See also

- The *Exploring Drupal hooks* recipe
- The *Setting up a local development environment* recipe
- Find out more about Drupal module development at `http://drupal.org/node/361112`
- Find out more about CiviCRM hooks at `http://wiki.civicrm.org/confluence/display/CRMDOC40/CiviCRM+hook+specification#CiviCRMhookspecification-hook_civicrm_pre`

Exploring CiviCRM extension development using Civix

CiviCRM extensions are agnostic. This means that they will work regardless of what CMS you are using: Drupal, Joomla!, or WordPress. If you have ever explored the contents of your CiviCRM module directory, you'll see there are a bewildering number of directories and files. Many of these files work together. So, if you want to start developing your own CiviCRM extensions, you will need to know how the files work together.

How to do it...

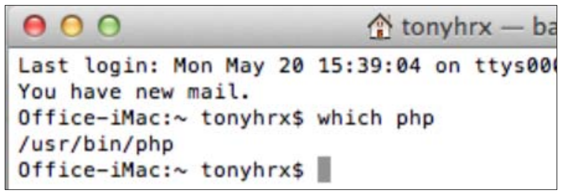
Civix is a command-line tool that helps you develop CiviCRM extensions. In this recipe, we will use a MAMP local development environment available on Mac OS X and install Civix. Then, we will create our own CiviCRM extension and add a page. The recipe can be adapted to other environments.

1. Start up your local environment and set up a CiviCRM development site using your CMS of choice. In this recipe, we will use MAMP.

2. Check where PHP is running so that you can run PHP from the command line in your local development environment. Open a command-line tool, such as `Terminal`, and start typing.

   `which php`

   This will show the path to PHP. This is the PHP that runs when you type `php` on the command line.

   In this example, PHP is running from the version that is installed with Mac OS X. We need to change this.

3. In your computer file system, navigate to your local development environment files to where MAMP runs PHP. In this case, it is `/Applications/MAMP/bin/php/php5.3.14`.

   In your own installation, the path may be different.

4. In your computer file system, locate the file `/Users/<yoursusername>/.bash_profile` and open it in a text editor. Add the following line:

   `export PATH=/Applications/MAMP/bin/php/php5.3.14/bin:$PATH`

   Note that this recipe is for Mac OS, so details for Windows machines will be different.

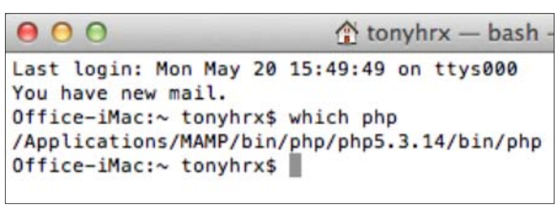
5. In `Terminal`, type in the following command:

   `source .bash_profile`

   This reloads the profile you just edited.

6. In `Terminal`, type in the following command:

   `which php`

   This shows that the correct PHP is running.

7. In your browser, navigate to `https://github.com/totten/civix/` and download the `civix .zip` archive.
8. Unzip the archive. It will unzip to a directory called `civix-master`. Rename the directory `civix` and store it in your `Applications` directory.
9. In your browser, navigate to `http://getcomposer.org/download/`. Composer is a tool that adds all the up-to-date resources that Civix needs in order to work properly.
10. In `Terminal`, enter the following commmand:

    ```
    curl -sS https://getcomposer.org/installer | php
    ```

    This downloads Composer.
11. In `Terminal`, change the directory to where you place Civix with this command:

    ```
    cd /Applications/civix
    ```

12. Now enter this command to make Composer install all the extra resources that Civix needs:

    ```
    php $HOME/composer.phar install
    ```

    In the `Terminal` window, you will see Composer downloading all the files. Civix is now installed.
13. Open the `.bash_profile` again and enter the following line:

    ```
    export PATH=/Applications/civix:$PATH
    ```

    This ensures that Civix runs whenever you type `civix` into the command line.
14. In CiviCRM, navigate to **Administer** | **System Settings** | **Directories**. Check that you have configured the **CiviCRM Extensions Directory** setting properly.
15. In your local development environment, copy the path to the default directory that contains your `civicrm_settings.php` file.
16. In `Terminal`, type the following command, changing the path to the location of your own `civicrm_settings.php` file:

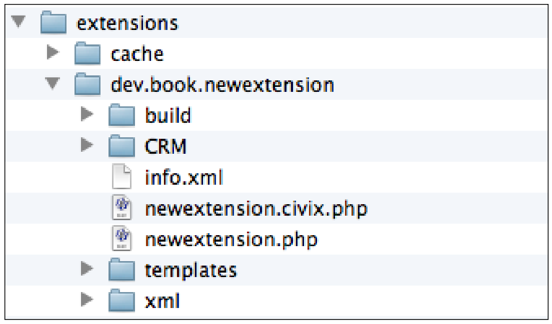
    ```
    civix config:set civicrm_api3_conf_path /var/www/drupal/sites/
    default/
    ```

    This "connects" Civix to your development site.
17. In `Terminal`, change directories to **CiviCRM Extensions Directory**.
18. You are now ready to start your new CiviCRM extension. In `Terminal`, type in the following command:

    ```
    civix generate:module dev.book.newextension
    ```

The name of the module follows CiviCRM's extension-naming conventions.

Civix generates the skeleton files for the extension.



19. In `Terminal`, change directories to your new extension directory. Type in the following command:

    ```
    civix generate:page MyPage civicrm/my-page
    ```

    This generates all the files required to develop the page.

How it works...

Civix does all the basic work for you. It creates the structure and the files necessary to begin building your extension. Furthermore, each file contains template code that you can adapt to your own purpose.

This saves a huge amount of development time. Furthermore, it provides a means of exploring how CiviCRM is structured. You can generate pages, forms, searches, reports, API calls, and unit tests, and at each stage see what files Civix adds and how they work together.

See also

- The *Setting up a development environment* recipe
- Find out more about CiviCRM extension development at `http://wiki.civicrm.org/confluence/display/CRMDOC43/Create+a+Module+Extension`

Index

D

R

S

T

Thank you for buying

CiviCRM Cookbook

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

About Packt Open Source

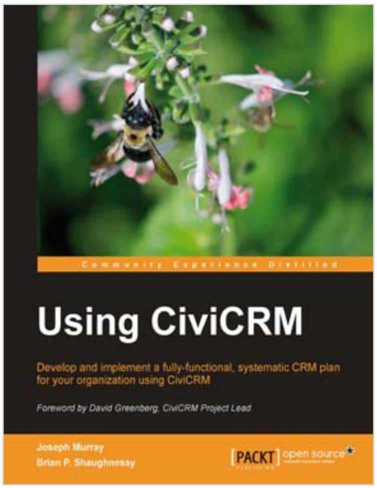
In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] open source
PUBLISHING
community experience distilled

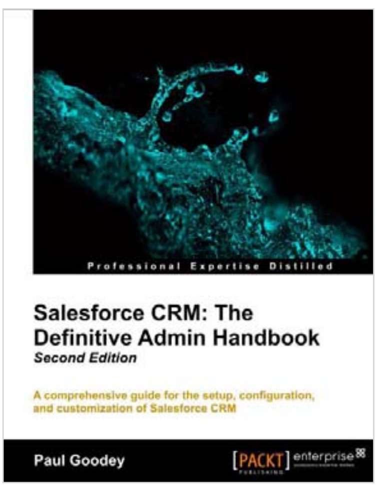


Using CiviCRM

ISBN: 978-1-84951-226-8 Paperback: 464 pages

Develop and implement a fully-functional, systematic CRM plan for your organization using CiviCRM

1. Build a CRM that conforms to your needs from the ground up with all of the features that you want
2. Develop an integrated online system that handles contacts, donations, event registration, bulk e-mailing, case management and other functions such as activity tracking, grants, reporting, and analytics
3. Integrate CiviCRM with Drupal and Joomla!



Salesforce CRM: The Definitive Admin Handbook Second Edition

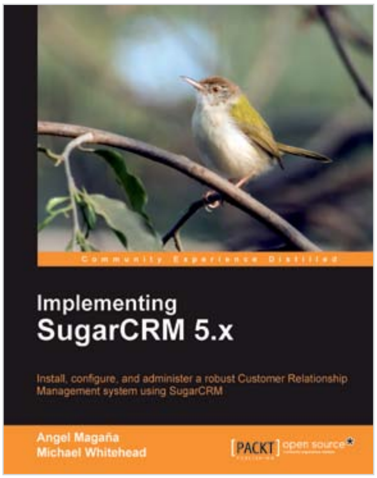
ISBN: 978-1-78217-052-5 Paperback: 420 pages

A comprehensive guide for the setup, configuration, and customization of Salesforce CRM

1. Updated for Spring '13, this book covers best practice administration principles, real-world experience, and critical design considerations for setting up and customizing Salesforce CRM
2. Analyze data within Salesforce by using reports, dashboards, custom reports, and report builder
3. A step-by-step guide offering clear guidance for the customization and administration of the Salesforce CRM application

Please check **www.PacktPub.com** for information on our titles

[PACKT] open source
community experience distilled
PUBLISHING



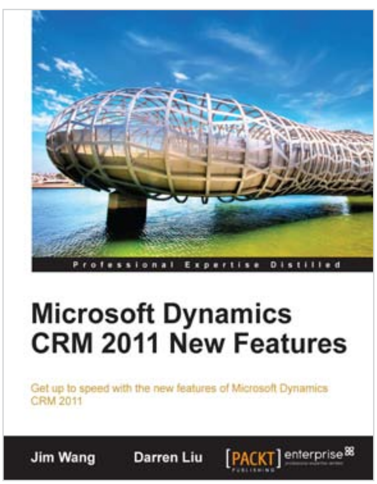
Implementing SugarCRM 5.x

ISBN: 978-1-84719-866-2 Paperback: 352 pages

Install, configure, and administer a robust Customer Relationship Management system using SugarCRM

1. Analyze and weigh deployment options based on your needs and resources
2. A brief overview of the benefits of SugarCRM 6.0
3. Review powerful built-in customization tools and popular third-party enhancements for your SugarCRM system
4. Learn about on-going maintenance needs such as backups and user management

---



Microsoft Dynamics CRM 2011 New Features

ISBN: 978-1-84968-206-0 Paperback: 288 pages

Get up to speed with the new features of Microsoft Dynamics CRM 2011

1. Master the new features of Microsoft Dynamics 2011
2. Use client-side programming to perform data validation, automation, and process enhancement
3. Learn powerful event driven server-side programming methods: Plug-Ins and Processes (Formerly Workflows)
4. Extend Microsoft Dynamics CRM 2011 in the Cloud

Please check **www.PacktPub.com** for information on our titles