

# CiviCRM Developer Training 2014

Parvez Saleh



# Resources

**Extension Installation** - <http://wiki.civicrm.org/confluence/display/CRMDOC/Create+an+Extension>

<http://wiki.civicrm.org/confluence/display/CRMDOC/Create+a+Module+Extension>

## Test Cases:

PHPUnit: <http://wiki.civicrm.org/confluence/display/CRM/Setting+up+your+personal+testing+sandbox+HOWTO>

Webtests: <http://wiki.civicrm.org/confluence/display/CRM/Setting+yourself+up+to+work+with+Selenium+tests>

# Aims and Objectives

## Day 1

- **How CiviCRM works** - code layout and architecture
- **Extending CiviCRM** - hooks, APIs, custom PHP files and templates.
- **Native Extension** - civix and more.
- **Internationalization** - including translation tools and multilingual sites

## Day 2

- **Advanced features** - message templates, batch update via profile, checksum authentication, command-line scripts.
- **Testing** - unit tests and webtests
- **Debugging** - tips and tricks
- **Integration with Drupal and Joomla** - including user/contact synchronization, Profiles, Organic Groups, Views2, Joomla extensions

# How CiviCRM Works - Components

CiviCRM is pear based.

- Quick Form
- Quick Form controller (for wizards, import mailing etc)
- BD\_DataObject (precursor to active record. Think about data in terms of objects, rather than columns)
- Mail
- Smarty (used to be pear)

Templating is smarty based

Jquery for Javascript

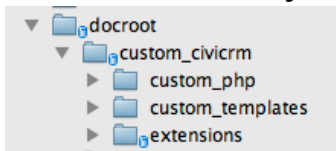
# How CiviCRM Works - Codebase

## Directory structure

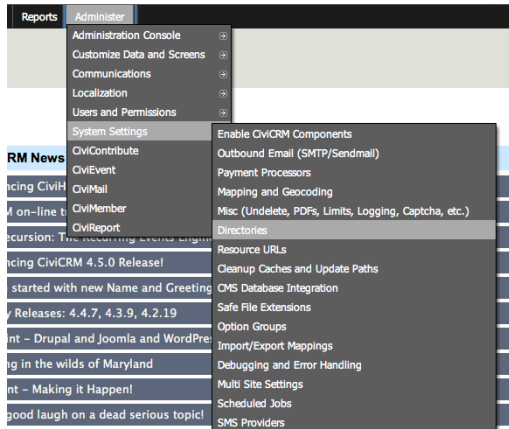
- top level is CRM for php, contains Business logic
  - a. DAO stands for data access object
  - b. BAO stands for business access object.
  - c. Form - In general each page in CiviCRM which contains a form that maps to a file in one of the form directories
  - d. Page - If a CiviCRM screen isn't a Form, it is probably a page
  - e. xml - This directory contains a menu directory which maps urls to CRM form or page classes and controls access to these URLs using permissions.
- templates for tpl
- packages for third party libraries (this is the reason why CiviCRM is pretty large - 2/3 of the stuff is PEAR JS, etc.

# Prepare CiviCRM for amendments

Create Directories in your file system for custom code



Set CiviCRM directories accordingly



Default values will be supplied for these upload directories the first time you access CiviCRM – based on the CIVICRM\_TEMPLATE\_C

Save Cancel

Temporary Files   
File system path where temporary CiviCRM files – such as import data files – are uploaded.

Images   
File system path where image files are uploaded. Currently, this path is used for images associated with premiums (CiviCor

Custom Files   
Path where documents and images which are attachments to contact records are stored (e.g. contact photos, resumes, con

Custom Templates   
Path where site specific templates are stored if any. This directory is searched first if set. Custom JavaScript code can be ad  
CiviCase configuration files can also be stored in this custom path. ([learn more...](#))

Custom PHP Path Directory   
Path where site specific PHP code files are stored if any. This directory is searched first if set.

CiviCRM Extensions Directory   
Path where CiviCRM extensions are stored.

Save Cancel

# The Requirement

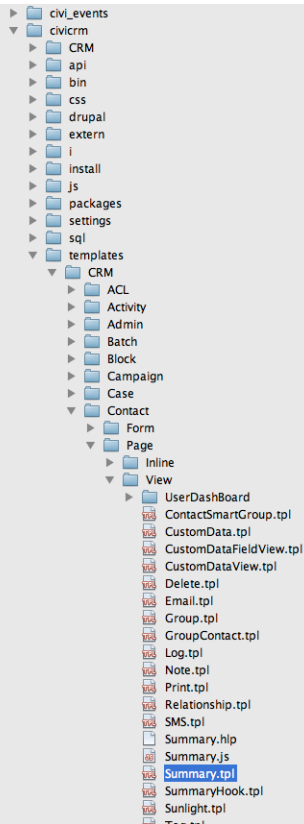
ACME Membership organisation would like to make some changes to the contact summary page.

1. Display the membership details from the membership tab on the summary page
2. Display a summary total of all contributions on the summary page

# Where to start?

View the source of the page in question and find .tpl

```
417 <div id="printer-friendly">
418 <a href="/civicrm/contact/view?cid=66&key=1cb8ab1a7f4828e415f0f195f31f128_2917&context=search&snippet=2" target='_blank' title="Printer-friendly view"
419 </a>
420 </div>
421 </div>
422 <div class="clear"></div>
423
424
425
426
427 <!-- .tpl file invoked: CRM/Contact/Page/View/Summary.tpl. Call via form.tpl if we have a form in the page. -->
428
429
430 <div><input name="entryURL" type="hidden" value="http://drupal.demo.civicrm.org/civicrm/contact/view?reset=1&cid=66&key=1cb8ab1a7f4828e415f0f195f31f128_2917"
431 </div>
432
433
434
435
436
437 <script type="text/javascript" src="/sites/all/modules/civicrm/packages/ckeditor/ckeditor.js"></script>
438
439
440
...
```





# Method 1: Override the template

Requirement:

- HTML
- CiviCRM API call from template (CiviCRM in-built API explorer comes in handy)

After locating the .tpl file, copy the .tpl to the custom template directory, retaining the directory structure as in Core templates directory.

Open the .tpl file & make the below changes

1. Make Contribution get API call for the contact (You can find the contact ID variable using debug) & calculate Contribution summary total and display in a HTML div/table.
2. Make Membership get API call for the contact and loop through the results and display the rows in a HTML table.

# Method 1: Override the template

## Pros

- Lots of customisations to the same file, no conflict of changes
- Easy to manage

## Cons

- Needs to be reviewed during upgrades, as new fields/features might have been added in the upgraded version.
- Hard to spot changes unless commented well.
- Difficult to identify dependency on other work done to deliver the requirement

# Method 2: Extra.tpl

Append HTML/Smarty code & jQuery functions to any template without having to create a customized copy of the entire file.

1. Rename/move the .tpl file in the custom template directory
2. Create a new .extra.tpl file (template\_to\_append\_to.extra.tpl)
3. Redo/Copy the content added in above step (Method 1) into a DIV
4. Move the DIV to the desired location in the summary screen using JQuery functions. Make sure you use smarty {literal} tags.

## Pros

- No need to review the customisation during upgrades, as we are not overriding any core templates.

# Method 3: Using CiviCRM Hooks

Requirement:

Civix

CiviCRM API call from PHP (Refer CiviCRM in-built API explorer)

For implementing CiviCRM hooks, we need to create an extension

## 1. Generate Extension using Civix

```
civix generate:module uk.co.vedaconsulting.training
```

## 2. Update 'info.xml' file in the extension directory

## 3. Enable the extension

## 4. Implement hook\_civicrm\_summary

In summary hook, call CiviCRM API to get the Contribution summary total and Membership details