LearnCubiByExamplesC2
Learn Cubi By Examples Chapter 2 - Create A Module
id , **en**

Updated Jan 13, 2012 by jack@openbiz.me

Learn Cubi By Examples Chapter 2 - Create A Module

At the end of previous chapter, we have database schema ready. This chapter will start the initial implementation. We call it phase 1, the prototyping phase.

Write Trac Module

In Cubi, an application can be one or more Cubi modules. To make it simple to manage, we put all Trac application into 'trac' module.

Module Directories

With a module, you can choose a directories the way you want. Typically, each business component may have its own sub-directory. In trac module, we have the following sub-directories for business components as

```
modules
- trac
-- attach (ticket attachment)
-- comments (ticket comments)
-- component
-- enum (enumerations)
-- milestone
-- product
-- ticket
-- version
```

Other sub-directories are typical Cubi module folders:

```
modules
- trac
-- message (contains module message files)
-- resource (contains module resource files)
-- view (contains module View files, view is same as page)
-- websvc (contains module public web service files)
-- do (contains module Data Object files)
-- form (contains module Form files)
```

Generate Metadata

A Cubi module can be created manually or with a command line tool called “gen_meta”. In this chapter, we will use gen_meta tool which is a command line metadata and module generator. This tool guide user to create a new module in steps. It accomplishes the all manual editing in one shot.

First, launch a console - a DOS window or a Linux/Unix console. Let's assume Cubi is installed under /cubi/ folder. Change director to /cubi/bin/tools/. Then type in the following command.

```
# php gen_meta.php Default trac_ticket trac.ticket
```

Where “Default” is the first database name defined in /cubi/application.xml, “trac_ticket” is the table name, “trac” is the module name. "trac.ticket" means we want to create metadata under /cubi/modules/trac/ticket/ folder.

Following the steps prompted by “gen_meta”, always press “enter” to make choice suggested by the tool.

```
# php gen_meta.php Default trac_ticket trac.ticket
----------------------------------------
Please select metadata naming:
1. module path: \trac\ticket, object name: TracTicket, module name: trac.ticket
2. module path: \trac\ticket, object name: Ticket, module name: trac.ticket
S. specify a custom module path, object name and module name
Please select: [1/2/s] (1) : 2

Access control options:
1. Access and Manage (default)
2. Access, Create, Update and Delete
3. No access control
Please select access control type [1/2/3] (1) :

----------------------------------------
Target dir: C:\xampp\htdocs\test\cubi4\modules\trac\ticket
Medata file to create:
  do/TicketDO.xml
  form/Ticket...Form.xml
  view/TicketView.xml
Do you want to continue? [y/n] (y) :
```

```
---------------------------------------
Do you want to generate data Object? [y/n] (y) :
Generate Data Object metadata file ...
Start generate dataobject TicketDO.
Create directory C:\xampp\htdocs\test\cubi4\modules/trac/ticket/do
        /trac/ticket/do/TicketDO.xml is generated.
---------------------------------------
Do you want to generate form Object? [y/n] (y) :
Generate Form Object metadata files ...
Start generate form object TicketListForm.
Create directory C:\xampp\htdocs\test\cubi4\modules/trac/ticket/form
        /trac/ticket/form/TicketListForm.xml is generated.

Start generate form object TicketNewForm.
        /trac/ticket/form/TicketNewForm.xml is generated.

Start generate form object TicketEditForm.
        /trac/ticket/form/TicketEditForm.xml is generated.

Start generate form object TicketDetailForm.
        /trac/ticket/form/TicketDetailForm.xml is generated.

Start generate form object TicketCopyForm.
        /trac/ticket/form/TicketCopyForm.xml is generated.

---------------------------------------
Do you want to generate view Object? [y/n] (y) :
Generate view Object metadata files ...
Start generate form object TicketListView.
Create directory C:\xampp\htdocs\test\cubi4\modules/trac/view
        /trac/view/TicketListView.xml is generated.
---------------------------------------
Do you want to generate module dashboard files? [y/n] (y) :
Generate Module Dashboard ...
Start generate DashboardForm.xml .
Create directory C:\xampp\htdocs\test\cubi4\modules/trac/widget
        /trac/widget/DashboardForm.xml is generated.

Start generate DashboardView.xml .
        /trac/view/DashboardView.xml is generated.

Start generate LeftMenu.xml .
        /trac/widget/LeftMenu.xml is generated.

Start modify view.tpl to enable module left menu supports .
        /trac/template/view.tpl is modified.

---------------------------------------
Do you want to create mod.xml? [y/n] (y) :
Generate mod.xml ...
Start generate mod.xml.
        /trac/mod.xml is generated.
```

After the command is executed, the following files are generated.

```
Module configuration file
/cubi/modules/trac/mod.xml

Module DO file
/cubi/modules/trac/ticket/do/TicketDO.xml

Module Form file
/cubi/modules/trac/ticket/form/TicketListForm.xml
/cubi/modules/trac/ticket/form/TicketDetailForm.xml
/cubi/modules/trac/ticket/form/TicketEditForm.xml
/cubi/modules/trac/ticket/form/TicketNewForm.xml
/cubi/modules/trac/ticket/form/TicketCopyForm.xml

Module View file
/cubi/modules/trac/view/TicketListView.xml
/cubi/modules/trac/view/DashboardView.xml

Module template file
/cubi/modules/trac/template/*

Module widget file
/cubi/modules/trac/widget/*
```

Understand module description file

Every Cubi module has its own description file mod.xml. Let's see what is in mod.xml.

```
<?xml version="1.0" standalone="no"?>
<Module Name="trac" Description="trac module" Version="0.1" OpenbizVersion="2.4">
    <ACL>
        <Resource Name="trac">
            <Action Name="Access" Description="Access Trac Module Dashboard"/>
        </Resource>
        <Resource Name="trac.ticket">
            <Action Name="Access" Description="Access Trac Ticket"/>
            <Action Name="Manage" Description="Manage Trac Ticket"/>
        </Resource>
    </ACL>
    <Menu>
        <MenuItem Name="Trac" Title="Trac" Description="Trac Description" URL="/trac/dashboard" Parent="" Order="10">
            <MenuItem Name="Trac.Ticket" Title="Ticket" Description="Trac Ticket description" URL="" Parent="" Order="10">
                <MenuItem Name="Trac.Ticket.List" Title="Ticket Manage" Description=""  URL="/trac/ticket_list" Order="10'
            </MenuItem>
        </MenuItem>
```

```
    </Menu>
    <Dependency>
        <Module Name="system"/>
    </Dependency>
</Module>
```

mod.xml contains 3 sections

- Access control. "ACL" section defines the resources and their actions. These definitions will be used to control permissions of given roles. You can find more details at Cubi Access Control.
- Menu. "Menu" section defines the page links on Cubi navigation system - application tabs, breadcrumb and left side menus. You can find more details at Cubi Menu.
- Dependency. "Dependency" defines modules the current module depends on. The depended modules should be installed first.

Load Trac Module

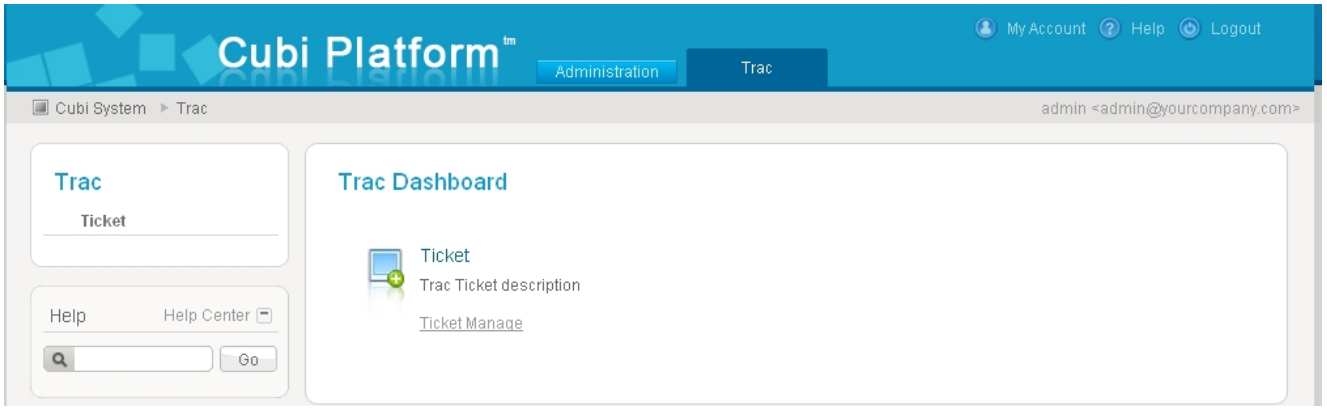
In order to load the newly generated module into Cubi, “load_module” (also called module loader) tool will be used. Module loader does the following work:

load module ACL setting load module menu load module resource give Cubi admin full permissions to access this module The output of running the command

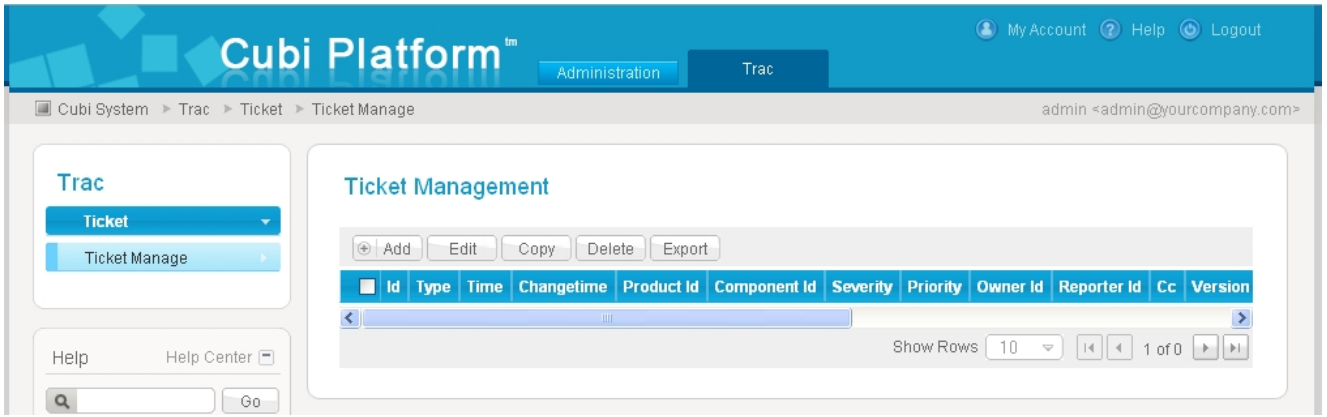
```
# php load_module.php trac
Start loading trac module ...
-------------------------------------------------------
[2011-10-03T22:16:57-07:00] Loading module trac
[2011-10-03T22:16:58-07:00] Install Module Sql.
[2011-10-03T22:16:58-07:00] Install Module trac
[2011-10-03T22:16:58-07:00] Install Module ACL.
[2011-10-03T22:16:58-07:00] Install Module Menu.
[2011-10-03T22:16:59-07:00] Install Module Widget.
[2011-10-03T22:16:59-07:00] Install Module Resource.
[2011-10-03T22:16:59-07:00] Install Module Change Logs.
[2011-10-03T22:16:59-07:00] Copy resource files to /cubi/resources folder.
[2011-10-03T22:16:59-07:00] trac is loaded.

Give admin to access all actions of module 'trac'
-------------------------------------------------------
End loading trac module
```

If you make a change on mod.xml file, please increase the version number to force the module reload. For example, change Version="0.1.0" to Version="0.1.1"

Let’s test this new module. If you have logged in Cubi site, please log out and re-login. You should be able to see a new Tab called "Trac". Clicking "Trac" tab will take user to Trac Dashboard view which is a page lists all menu items in trac module.



Clicking "Ticket Manage" link to land on the ticket management view where you can do the basic create/edit/delete operations.



Please note if you don't see the Trac tab, please refer to the this discussion thread.

How things work

So far so good. Now let's see how Cubi generates the cool AJAX pages from the xml files we just created.

The gen_meta command line tool generates 3 types of xml files.

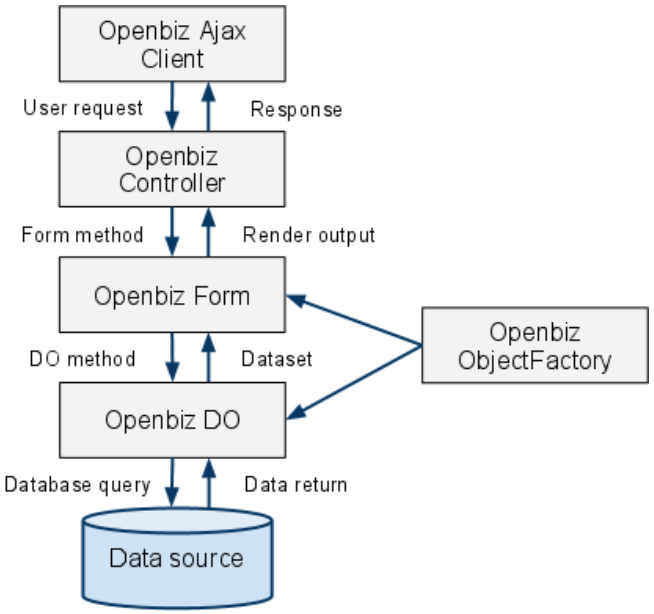
- a Data Object file (DO) that defines the object relational mapping (ORM) on database tables.
- Form files (Form) that defines a group of UI elements on top of a Data Object.
- a View file (View) that defines a web page contains one or more Forms.

For Data Object, you can get more details from Openbiz Data Object, and Openbiz UI for Form and View.

This is what happens on loading a Cubi url http://host/cubi/index.php/trac/ticket_list from browser.

1. index.php calls `_forward.php` to parse the url. It understands "/trac/ticket_list" points to "`/cubi/modules/trac/view/TicketListView.xml`" based on Cubi url naming convention.
2. The render method of the View is invoked.
3. This render method of a view then calls render method of its included Form (`/cubi/modules/trac/ticket/form/TicketListForm.xml`)
4. The render method of the Form calls its Data Object's data query methods
5. The query methods prepare the query SQL and execute query against database
6. Form uses returned data set from DO to generate HTML with its template
7. View uses the output of its Forms to generate HTML with its template

The diagram below describes how Ajax call works in Cubi.



Test ticket page

Let's try it out now.

- Add a ticket. On the ticket list form, click on the "Add" button. In the new ticket form, enter your inputs as below, then click "Save" button.
  - Type: defect
  - Severity: high
  - Priority: P0
  - Summary: test ticket 1
  - Description: This is a test ticket 1
- Edit a ticket. On the ticket detail form, click "Edit" button. Change Priority from P0 to P1. Then click "Save" button.
- Copy a ticket. On the ticket list form, click "Copy" button. Change summary and description as below, then click "Save" button.
  - Summary: test ticket 2
  - Description: This is a test ticket 2
- Sort tickets. On the ticket list form, click on "Summary" column head, verify the records are sorted on summary column.
- Delete a ticket. On the ticket list form, highlight a record, click "Delete" button. Click "OK" on the confirmation dialog, verify the selected ticket is deleted.

Now you are ready to go next chapter.