LearnCubiByExamplesC8
Learn Cubi By Examples Chapter 8 - Beyond Development

Updated Oct 25, 2011 by rockys...@gmail.com

Learn Cubi By Examples Chapter 8 - Beyond Development

After all the work done in previous chapters, we may claim code complete on trac module. While there are work other than coding needs to be done before making it to production. This chapter will cover the following topics.

- Build and deploy
- Multi-language support
- Make a new theme
- Secure your work
- Tune performance
- Manage users and roles

Build and Deploy

After the code and metadata are finished and unit tested, it is the time to build it and deploy to test or production environment.

Cubi provides a build tool that helps creating a tarball of Cubi based applications. Cubi Build Tool includes more details of how to use the tool as well as a sample of RPM builder spec.

As all Cubi source is available, one can use his/her favorite build tool to create deployment package.

Please keep in mind after the application source is copied to the target application directory, it is recommended to run post actions including:

- clean the cache directories
- load your application modules
- give write permission to proper folders

The following snippet is copied form the sample rpm builder spec

```
cd %{contentdir}/%{name}/cubi
rm -rf files/cache/*
php bin/tools/load_module.php module1
php bin/tools/load_module.php module2
chmod -R 777 log/ session/ files/
```

Multi Language Support

If your applications are used in a global environment, it is natural to enable multiple languages support. Cubi language package generation tool would make such work easy. Please find more details in Cubi Multi-Language Support page.

Make a New Theme

Cubi release includes a default theme. It is fairly simple to create a custom theme. Here are the steps:

1. Create a subfolder under /themes folder.
2. Copy the subfolders from /themes/default to the newly created subfolder.
   - css - contains the stylesheet files
   - images - contains images files
   - js - contains theme related js files
   - template - contain shared template files. (Each module can have its own template file under modules/modname/template/)
3. Modify the files in above 4 folder to customize your theme.

A theme can be also created by theme generator command line tool and theme management screen.

The command line tool can be found at /cubi/bin/tool/gen_theme.php

```
# php gen_theme.php new_theme_name
```

The script will copy the default theme into /cubi/theme/new_theme/ and create theme.xml under the new theme folder. The theme.xml describes theme's basic information that is used in Cubi for managing theme and selecting theme. The theme.xml of default theme is

```
<?xml version="1.0" encoding="utf-8"?>
<metafile version="1.5"  client="site" >
  <name>Openbiz Blue</name>
  <preview>theme_screenshot.gif</preview>
  <icon>theme_icon.gif</icon>
  <version>1.5</version>
  <creationDate>2010-06-20</creationDate>
```

```
  <author>Jixian</author>
  <authorEmail>jixian2003@qq.com</authorEmail>
  <authorUrl></authorUrl>
  <copyright>Copyright (C) 2005 - 2010 Open Source Matters. All rights reserved.</copyright>
  <license>http://www.gnu.org/licenses/gpl-2.0.html GNU/GPL</license>
  <description>Cubi default theme in light blue style.</description>
</metafile>
```

Once a new theme is added, it can be set as default theme at theme management screen or at my account preference screen.

Secure Your Work

Once the code is completed and deployed, of course we want to protect our effort by blocking others to

- hack the application server
- steal the source code of your specific application logic

Protect the Application Server

First we want to block navigation of all metadata xml files including /cubi/application.xml, as well as .inc files. It can be typically done by changing web server configuration file. For example, in apache conf file or /cubi/.htaccess, we set

```
<FilesMatch "\.xml$">
    Order deny,allow
    Deny from all
</FilesMatch>

<FilesMatch "\.inc$">
    Order deny,allow
    Deny from all
</FilesMatch>
```

Meanwhile, we can move some writable directories into a non-web folder. To do this, we need to edit /cubi/bin/app_init.php

```
// define a hidden directory that is invisible on web
define("HIDDEN_PATH", "/var/cubi");

/* Log file path */
define("LOG_PATH", HIDDEN_PATH.DIRECTORY_SEPARATOR."log");

/* define session save handler */
define("SESSION_PATH", HIDDEN_PATH.DIRECTORY_SEPARATOR."session"); // for default FILE type session handler

/* file cache.DIRECTORY_SEPARATOR."directory */
define('CACHE_PATH', HIDDEN_PATH.DIRECTORY_SEPARATOR."files".DIRECTORY_SEPARATOR."cache");

/* temopary files directory */
define('TEMPFILE_PATH', HIDDEN_PATH.DIRECTORY_SEPARATOR."files".DIRECTORY_SEPARATOR."tmp");

/* metadata cache files directory */
define('CACHE_METADATA_PATH', HIDDEN_PATH.DIRECTORY_SEPARATOR."files".DIRECTORY_SEPARATOR."cache".DIRECTORY_SEPARATOR."meta

/* secured upload / attachment file path. files cannot be accessed by a direct url */
define('SECURE_UPLOAD_PATH', HIDDEN_PATH.DIRECTORY_SEPARATOR."files".DIRECTORY_SEPARATOR."sec_upload");
```

Secondly, we want to avoid network attacks if your application is on available on internet (not company intranet). The protection can be done with firewall settings. Cubi security module provides an extra protection on application layer. It works better of organization without strong operation force. Please check Cubi security module for more information.

Secure the Source Code

In commercial applications, we don't want give source code to clients. In such cases, we need to encode/encrypt the source code before the packaging.

The two mature PHP encoding solutions available on market are

- Zend Guard. http://www.zend.com/en/products/guard/
- ionCube. http://www.ioncube.com/

Cubi release includes a command line tool that can work ionCube. The script can be found at /cubi/bin/tools/makelic_module.php. This script can encode folder with ionCube license file.

```
<?php
/*
 * create license command line script
 */
if ($argc<3) {
  echo "usage: php makelic_module.php module_name".PHP_EOL;
  exit;
}

include_once ("../app_init.php");
if(!defined("CLI")){
  exit;
}

$ENCODER_PATH = "C:\\Program Files\\ionCube Pro PHP Encoder 7.0";

$module = $argv[1];
$encoder_cmd = $ENCODER_PATH.DIRECTORY_SEPARATOR."ioncube_encoder5";
$source_dir = MODULE_PATH.DIRECTORY_SEPARATOR.$module;
```

```
$target_dir = MODULE_PATH.DIRECTORY_SEPARATOR.$module."_encoded";
$license_file = "license_".$module.".txt";
$pass_code = "pass_".$module;
$callback_file = "callback_".$module.".php";
$properties = "product='cubi-".$module;

$cmd = "\"$encoder_cmd\" \"$source_dir\" -o \"$target_dir\" --with-license $license_file --passphrase $pass_code --license-
echo $cmd."\n";
system($cmd);

?>
```

As you can see, the script is made for Windows. It is easy to port it to other platform. Please check http://www.ioncube.com/USER-GUIDE.pdf to understand how ionCube encoder works.

Tune performance

Cubi application performance can be improved by leveraging cache. Openbiz-Cubi supports cache in the following objects

- View. View can be cached according to the given URL
- Form. Form can be cached by its name
- DataObj. DataObj query (only "Select" query) can be cached per unique SQL statement.

To enable cache on openbiz objects, you need to add "CacheLifeTime" attribute for the metadata element. CachLifeTime has unit as "second". For example,

```
<View Name="MiscView" CacheLifeTime="60">
```

tells that the content of view "Misc" is cached by the system for 60 seconds.

Openbiz-Cubi counts on cache service to cache content. Zend_Cache is used by cache service. Thanks for Zend_Cache, the cache service can support all cache backend including:

- File cache
- SqlLite cache
- MemCache
- APC
- ...

Cache service comes with a configuration file under /cubi/openbiz/metadata/service/cacheService.xml. In the configuration file, user can specify the cache backend.

The default content cache is file cache, Cubi saves the cache data under /cubi/files/cache/data/...

Openbiz-Cubi parses metadata file once and cache it in /cubi/files/cache/metadata/. If PHP APC extension is installed, the metadata will be cached in APC memory for better performance.

Manage users and roles

Once the application is deployed, the next thing is to create roles and grant permissions to each role. With the correctly configured roles, we can control users with proper actions in the application.

Users and roles management are the core modules in Cubi, please check Cubi System Module to fully understand how it works.