mvigilCS263technology Documentation

Release 1

Morgan Vigil

May 06, 2013

CONTENTS

In this tutorial, I guide you through setting up your own local ownCloud server and discuss the application API. We will explore the App Framework modules by creating and enabling an app based on the Advanced App Template.

For more information about ownCloud in general, please see http://owncloud.org/about/.

Contents:

CHAPTER

ONE

GETTING STARTED

1.1 Installing ownCloud

1.1.1 Server

At the heart of the ownCloud is the idea of hosting your own cloud. In this part of the tutorial, we will describe installing and running your own ownCloud server.

There are two options:

1. Installing the development version of ownCloud from source https://github.com/owncloud/core
2. Installing the stable version of ownCloud from http://owncloud.org/install/

Hint: the stable version is an easier intall process.

1.1.2 Client (optional)

After you have installed your ownCloud server, download the appropriate client. You can choose a mobile or desktop client at http://owncloud.org/install/. However, ownCloud is accessible over the web via browser.

1.2 Connecting to your ownCloud server via browser

If you are on the machine where the ownCloud server is installed, start a new browser session and go to:

```
http://localhost/owncloud/index.php
```

If you are accessing a remote ownCloud server, visit:

```
http://REMOTE_OWNCLOUD_SERVER_IP/owncloud/index.php
```

In case you are using Mac OS X for development and set up your web server using MAMP, connect to your ownCloud server with:

```
http://localhost:8888/owncloud/index.php
```

Note that if you have configured your server to only be accessible via SSL, you need to access it using 'https' instead of 'http'. However, the default server is configured to operate using standard HTTP.

1.3 Helpful resources

http://doc.owncloud.org/server/5.0/admin_manual/installation.html gives thorough installation guides for different operating systems (including various Linux distributions).

CHAPTER

TWO

OVERVIEW OF OWNCLOUD API'S

In building an ownCloud application, there are two approaches: the ownCloud App API and the App Framework API. This tutorial focuses on developing apps using the App Framework API. However, I will discuss the main differences between the APIs and reference users to another tutorial on creating an ownCloud app using the App API.

In building an ownCloud application, there are two approaches: the ownCloud App API and the App Framework API. This tutorial focuses on developing apps using the App Framework API. However, I will discuss the main differences between the APIs and reference users to another tutorial on creating an ownCloud app using the App API.

In deciding between using the ownCloud App API and the App Framework API, consider that the ownCloud App API facilitates the app-creating process for developers new to programming. Rather than use the MVC architecture that the App Framwork API employs, ownCloud App API uses templates.

The main differences between the App API and the AppFramework API are summarized in a table below.

Criteria	ownCloud App API	App Framwork API
Difficulty	easy	medium
Architecture	routes and templates	routes and MVC
Testability	hard	easy
Maintainability	hard	easy
Templates	OC_Template	OC_Template and Twig
Security	manual checks	escapse XSS, default CSRF and Authentication checks

2.1 ownCloud App API

Components:

- OCS
- OCS_Result
- OC Templates
- View

Please follow this link to get the full ownCloud App API listing: http://api.owncloud.org/namespaces/OCP.html

2.2 App Framework API

Componenets:

- Main

- API Layer
- Request
- Controllers
- Database
- Responses
- Middleware
- Security and Authentication
- Twig Templates
- Testing

Please follow this link to get a full listing of the ownCloud API: http://doc.owncloud.org/server/master/developer_manual/app/appframew

CHAPTER

THREE

OWNCLOUD APP TUTORIAL

This is an ownCloud app tutorial that will result in a simple application for ownCloud based on the Advanced App Template. This tutorial is written for the stable version of ownCloud (v. 5) and with the support of the stable app repository. Prior to developing this app, you will need to have your ownCloud server set up based on the stable branch.

3.1 Setup

Clone the ownClouds apps repository into either:

- /var/www
- /var/www/html
- /srv/http
- /Users/[USER]/Sites

Change into the appropriate web server directory:

```
cd /Users/[USER]/Sites
```

And clone the appropriate version of the apps:

```
sudo git clone git://github.com/owncloud/apps.git -b stable5 apps
```

Now, change into the apps/ directory:

```
cd apps/
```

To finish our setup, we need to make a modification to two files, *apptemplateadvanced/appinfo/info.xml* and *appframework/appingo/info.xml*. There is currently a bug in ownCloud core that removes app directories when an app is disabled. In order to prevent that, add this line in *both* files after the <*author*>*AUTHOR NAME*</*author*>:

```
<shipped>true</shipped>
```

3.2 Creating an App based on Advanced App Template

ownCloud has made using the App Framework even easier with the Advanced App Template. The following steps will show you how to make your own app from the Advanced App Template.

First, copy the apptemplateadvanced/ directory to a directory with the name of your app:

```
cp -r apptemplateadvanced/ myapp
```

Change into your app directory:

```
cd myapp/
```

Now, you need to customize the files to refer to your app id and name instead of apptemplateadvanced. We will do this by executing:

```
find . -type f -exec sed -i .tmp 's/apptemplateadvanced/myapp/g' {} \;
find . -type f -exec sed -i .tmp 's/apptemplate_advanced/myapp/g' {} \;
find . -type f -exec sed -i .tmp 's/AppTemplateAdvanced/MyApp/g' {} \;
find . -type f -exec sed -i .tmp 's/Advanced App Template/My App/g' {} \;
find . -name "*.tmp" -type f -delete
```

All that's left is changing the author's name in myapp/appinfo/info.xml to your name.

3.3 Enabling App

To enable your app, you need to link it with the owncloud/apps/ directory:

```
ln -s /path/to/file/apps/myapp /path/to/file/owncloud/apps/myapp
```

Go to the ownCloud URL at http://localhost/owncloud/index.php or http://localhost:8888/owncloud/index.php. Navigate to the Apps page.

First, you will need to enable the App Framework app.

'My App' should be in the nav bar on the left side of the page. Enable it and an icon with the words "My App" should appear in the ownCloud dock on the leftmost part of the screen. If you click it, it will take you to a page with a text field and a button. The logic for this app is dictated by myapp/templates/main.php. To get an idea of how the components of myapp work, see *Modules of the Advanced App Template*.

That's it! That's all it takes to get an application set up with the ownCloud App Framework and Advanced App Template. Happy coding!

CHAPTER

FOUR

MODULES OF THE ADVANCED APP TEMPLATE

The following components are part of Advanced App Template:

- 3rdparty
- admin
- appinfo
- cache
- coffee
- controller
- css
- db
- dependencyinjection
- img
- js
- templates
- tests

All of these components are necessary for an app based on the Advanced App Template to work.

4.1 3rdparty

3rd party javascript that can be integrated into the app.

4.2 admin

Contains settings.php, which calls *main()* and connects main with the owncloud/admin page.

4.3 appinfo

Contains all metadata pertaining to the app. The three main files are:

- *info.xml*
- *routes.php*
- *app.php*

4.3.1 info.xml

info.xml specifies:

- *License* This file also dictates the license used for the app. In order for the app to function, it must be AGPL compatible and *MUST NOT* be a proprietary license.
- *Author*
- *Version*
- *ownCloud version requirements*
- *App ID*
- *App Name*
- *App Description*

4.3.2 routes.php

Maps file pathnames to URLs. This facilitates method calling or value extraction from certain URLs.

4.3.3 app.php

Provides navigation entry code that provides ownCloud with App ID, App Name, App Description, App Author, and App URL so that it can produce the App in the App Nav bar.

4.4 coffee

CoffeeScript files that compile into JavaScript files in *js*.

4.5 controller

Contains the Controller functions that are used as part of the MVC framework.

4.6 css

Contains CSS files for your app. To include an image or css in CSS, prepend the following to your path:

- %appswebroot%: gets the absolute path to your app
- %webroot%: gets the absolute path to owncloud

Example:

```
.folder > .title {
        background-image: url('%webroot%/core/img/places/myimage.png');
}
```

ownCloud uses formfactors for different platforms. ownCloud automatically detects the formfactor your app employs. The diffe

- mobile
- tablet
- standalone

In order to use the formfactor, add it as part of the filename. For example:

style.mobile.css mystyle.standalone.css

4.7 db

Contains database layer for your app. Separates data entries from database queries.

4.8 dependencyinjection

Enables you to create testable code. This is done by decoupling class dependencies. Dependency Injection facilitates the automatic creating of unit tests. You can add your own created classes to dicontainer.php in order to enable the creation and maintenance of clean, testable code. See ownCloud: Dependency Injection for more details.

4.9 img

Contains all image files used by application.

4.10 js

Contains all JavaScript for the application.

4.11 templates

There are two types of templates available to ownCloud apps–OC Templates and Twig Templates.

OC Templates provide template functions from the class OC_Templates.

Twig templates are the preferred template because they provide better prevention of XSS. In order to facilitate integration with ownCloud, the App Framework provides additional functions for Twig. These can be found here.

4.12 tests

Contains unit tests for your application. Tests usually have the suffix **Test** appended to the filename of the corresponding code to be tested. For example, controllers/ItemController.php would have a test file named tests/controllers/ItemControllerTest.php. PHPUnit executes all files ending in Test.php. See Unit Testing for more information.

CHAPTER

FIVE

SEARCH

- *search*