Components

Overview

The largest and most complex of the extension types, a component is in fact a separate application. You can think of a component as something that has its own functionality, its own database tables and its own presentation. So if you install a component, you add an application to your website. Examples of components are a forum, a blog, a community system, a photo gallery, etc. You could think of all of these as being a separate application. Everyone of these would make perfectly sense as a stand-alone system. A component will be shown in the main part of your website and only one component will be shown. A menu is then in fact nothing more then a switch between different components.

Throughout these articles, we will be using {ComponentName} to represent the name of a component that is variable, meaning the actual component name is chosen by the developer. Notice also that case is important. {componentname} will refer to the lowercase version of {ComponentName}, eg. "CamelCasedController" -> "camelcasedcontroller". Similarly, {ViewName} and {viewname}, {ModelName} and {modelname}, {ControllerName} and {controllername}.

We also strongly encourage developers to take a look at [Joomla!'s documentation](#).

Directory Structures & Files

Components follow the Model-View-Controller (MVC) design pattern. This pattern separates the data gathering (Model), presentation (View) and user interaction (Controller) activities of a module. Such separation allows for expanding or revising properties and methods of one section without requiring additional changes to the other sections.

In its barest state, no database entry or other setup is required to "install" a component. Simply placing the component into the /components directory will make it available for use. However, if a component requires the installation of database tables or configuration (detailed in the config.xml file), then an administrator must install the component using one of the installation options in the administrative back-end.

Note: Components not installed via one of the installation options or without a database entry in the #__components table will not appear in the administrative list of available components.

To illustrate the typical component directory structures and files:

```
/hubzero
  /administrator
    /components
      /com_example
```

```
...
/components
  /com_example
      /controllers
         example.php
      /models
         foo.php
      /views
         /index
            /tmpl
               display.php
               display.xml
      example.php
      router.php
```

In the above example, all component related files and sub-directories are split between the administrator components and front-end components. In both cases, the files are contained within directories titled "com_example". Some directories and files are optional but, for this example, we've included the most common setup.

The file structure in the administrative portion of the component is exactly the same as in the front side. Note that the view, models, controllers etc. of the front and admin parts are completely separated, and have nothing to do with each other - the front part and the admin part can be thought of as two different components! A view in the /administrator/components/com_example folder may have a counterpart with the same name in the /components/com_example folder, yet the two views have nothing in common but their name.

Directory & File Explanation

/com_{componentname}/{componentname}.php
        This is the component's main file and entry point *for the front-end part*.
/com_{componentname}/controller.php
        This file holds the default frontend controller, which is a class called
        "{ComponentName}Controller". This class must extend the base class "JController".
/com_{componentname}/views

        This folder holds the different views for the component.

        /com_{componentname}/views/{viewname}
                This folder holds the files for the view {ViewName}.
        /com_{componentname}/views/{viewname}/view.html.php

This file is the entry point for the view {ViewName}. It should declare the class {ComponentName}View{ViewName}. This class must extend the base class "JView".

/com_{componentname}/views/{viewname}/tmpl

This folder holds the template files for the view {ViewName}.

/site/views/{viewname}/tmpl/default.php
This is the default template for the view {ViewName}.

/com_{componentname}/models

This folder holds additional models, if needed by the application.

/com_{componentname}/models/{modelname}.php
This file holds the model class {ComponentName}Model{ModelName}. This class must extend the base class "JModel". Note that the view named {ViewName} will by default load a model called {ViewName} if it exists. Most models are named after the view they are intended to be used with.

/com_{componentname}/controllers

This folder holds additional controllers, if needed by the application.

/com_{componentname}/controllers/{controllername}.php
This file holds the controller class {ComponentName}Controller{ControllerName}. This class must extend the base class "JController".

Naming Conventions

Classes

The model, view and controller files use the jimport function to load classes from the Joomla! framework, JModel, JView and JController, respectively. Each class is then extended with a new class specific to the component.

The base controller class for the site is named {ComponentName}Controller. For the administrative section, an "s" is added to the ComponentName, giving {ComponentName}sController. Classnames for additional controllers found within the controllers/ subdirectory are {ComponentName}Controller{ControllerName} for site/ and {ComponentName}sController{ControllerName} for admin/.

The view class is named {ComponentName}View{ViewName}.

The model class is named {ComponentName}Model{ModelName}. Remember that the {ModelName} and the {ViewName} should be the same.

Reserver Words

There are reserved words, which can't be used in names of classes and components.

An example is word "view" (in any case) for view class (except "view" that must be second part of that class name). Because first part of view class name is the same as controller class name, controller class name also can't contain word "view". And because of convension (although violating of it won't produce an error) controller class name must contain component name, so component name also can't contain word "view". So components can't be named "com_reviews", or if thay are, thay must violate naming convention and have different base controller class name (or have some other hacks).

Examples

Here we have a basic front-end component that simply displays a "Hello, World!" message. We present it using both standard Joomla! 1.5+ methods and HUBzero methods, which differ in a few key ways. Note, however, that despite any differences from standard Joomla! methods, all HUBzero methods will still work on a stock Joomla! 1.5+ install.

Download: [Hello World component (Joomla! method)](#)

Download: [Hello World component (HUBzero method)](#)