Hello world demo

This is a short tutorial that will help you understand how to develop yourself the well known “Hello World!” demo, included with the release.

Unlike the demo release, though, we will implement the demo as though it was a complete new application you are developing.

We will assume you already have the Yii framework installed, up and running, if not, check this section in the manual: http://www.yiiframework.com/doc/guide/quickstart.installation

We will use the powerful `yiic` tool which can be used to automate code creation for several purposes. We assume that `YiiRoot` is the directory where Yii is installed.

Run yiic on the command line as follows:

```
% YiiRoot/framework/yiic webapp WebRoot/helloworld
```

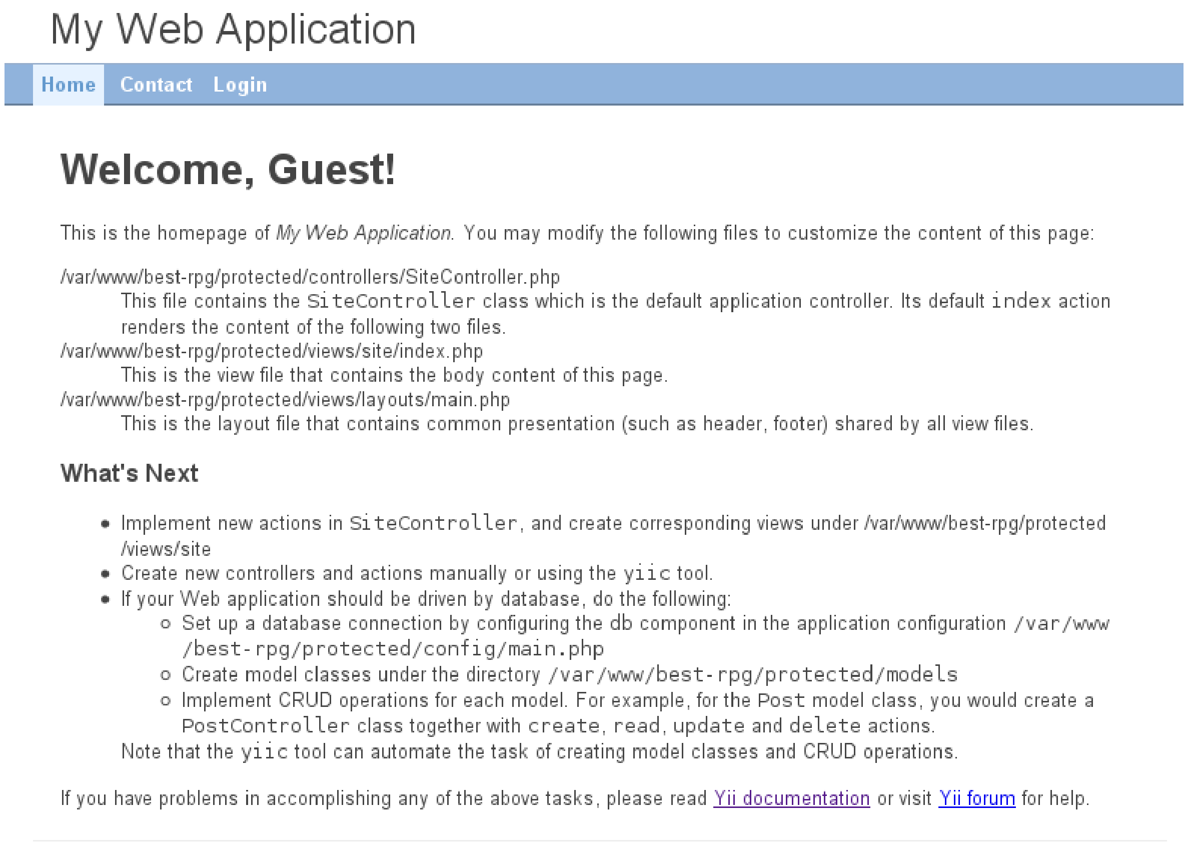
Note: When running `yiic` on Mac OS, Linux or Unix, you may need to change the permission of the `yiic` file so that it is executable. You may also use `php YiiRoot/framework/yiic.php` to replace `yiic`. Make also sure your user hasd permission to create directories under WebRoot.

Without writing a single line of code, we can test drive our first Yii application by accessing the following URL in a Web browser:

`http://hostname/helloworld/index.php`

Note: If you followed the installation guide, in order to check your server meets the necessary requirements, your WebRoot will probably be the YiiRoot. This is a bad practice, the framework files should not be exposed to web access, only your application files should; this will not be a problem to continue this tutorial.

If everything worked you should get something like this:



Welcome, Guest!

This is the homepage of *My Web Application*. You may modify the following files to customize the content of this page:

/var/www/best-rpg/protected/controllers/SiteController.php
This file contains the `SiteController` class which is the default application controller. Its default `index` action renders the content of the following two files.
/var/www/best-rpg/protected/views/site/index.php
This is the view file that contains the body content of this page.
/var/www/best-rpg/protected/views/layouts/main.php
This is the layout file that contains common presentation (such as header, footer) shared by all view files.

What's Next

- Implement new actions in `SiteController`, and create corresponding views under /var/www/best-rpg/protected/views/site
- Create new controllers and actions manually or using the `yiic` tool.
- If your Web application should be driven by database, do the following:
  - Set up a database connection by configuring the db component in the application configuration `/var/www/best-rpg/protected/config/main.php`
  - Create model classes under the directory `/var/www/best-rpg/protected/models`
  - Implement CRUD operations for each model. For example, for the `Post` model class, you would create a `PostController` class together with `create`, `read`, `update` and `delete` actions.

  Note that the `yiic` tool can automate the task of creating model classes and CRUD operations.

If you have problems in accomplishing any of the above tasks, please read Yii documentation or visit Yii forum for help.

And this should be the directory structure that the yiic tool created under your application directory: WebRoot/helloworld.

```
helloworld/
   index.php                 Web application entry script file
   assets/                   containing published resource files
   css/                      containing CSS files
   images/                   containing image files
   themes/                   containing application themes
   protected/                containg protected application files
      yiic                   yiic command line script
      yiic.bat               yiic command line script for Windows
      commands/              containing customized 'yiic' commands
         shell/              containing customized 'yiic shell' commands
      components/            containing reusable user components
         MainMenu.php        the 'MainMenu' widget class
         Identity.php        the 'Identity' class used for authentication
         views/              containing view files for widgets
            mainMenu.php     the view file for 'MainMenu' widget
      config/                containing configuration files
         console.php         the console application configuration
         main.php            the Web application configuration
      controllers/           containing controller class files
```

```
         SiteController.php  the default controller class
      extensions/            containing third-party extensions
      messages/              containing translated messages
      models/                containing model class files
         LoginForm.php       the form model for 'login' action
         ContactForm.php     the form model for 'contact' action
      runtime/               containing temporarily generated files
      views/                 containing controller view and layout files
         layouts/            containing layout view files
            main.php         the default layout for all views
         site/               containing view files for the 'site' controller
            contact.php      the view for 'contact' action
            index.php        the view for 'index' action
            login.php        the view for 'login' action
         system/             containing system view files
```

If you aren't familiar with MVC design pattern you should have a look at the corresponding section under the Yii framework guide, in general, M (the model) deal with data and business rules (typically DB interaction), V (the view) is used to deal with user-interaction (displaying data, forms, pages), C (the controller) manages the communication between the model and the view.

In this case what we need to look for is the default controller class which can be found at: WebRoot/helloworld/protected/controllers and it's called: **SiteController.php**.

Just for the purpose of this tutorial we'll delete the file, so go ahead and delete it (at this point your web application will not work anymore); then open a text editor of your choice, we need to recreate the default controller.

A controller is a php class, its methods are commonly called “actions” and they usually define the possible user actions when interacting or visiting a web page. Yii controllers all need to extend the framework controller class which is called CController.

```
<?php
class SiteController extends CController
{
```

Quite selfexplanatory, we need to rebuild the default controller which is called SiteController.

> **Note:** I suggest you check the guide's section about controllers to understand their naming convention and what they are used for.

Yii controllers have a default action which is called **actionIndex**, usually, when accessing a Yii webappl page we should specify the controller and the action we are accessing (ex www.example.com/helloworld?r=controller/action), if no action is specified, the default action (actionIndex) is accessed, if no controller and no action are specified (our case) the default controller and the default action will be accessed.

So, in order to complete our webapp all we need to do is add the default action to the controller and make it do something:

```
    public function actionIndex()
```

```
        {
                echo 'Hello World';
        }
```

Now, all you have to do is remember to close your class (**}**) and the php file (**?>**) and save your new file as: WebRoot/helloworld/protected/controllers/SiteController.php

Access your application and...

Hello World

At this point your tutorial would be done, although, we can expand it just a little more in order to obtain the same result but use a view instead. In fact, the procedure used for this demo does not reflect the standard development procedures. In my experience with MVC I learned that you never want a controller output user data (our echo), the controller should, as we said before, be just a mean of communication between the model and the view. In this case, a model is obviously not needed as no data needs to be fetched from a DB or a file, no business logic is needed.

The fact that no communication is needed between our view and our model (no model) doesn't mean the controller won't do anything, our default action (actionIndex) will change to do one simple thing: render the view. That can be achieved by simply doing this:

```
        public function actionIndex()
        {
                $this->render('index');
        }
```

The "render()" method comes from the Ccontroller class that we extended and its work is to render a view, in this case we are telling it to dender the "index" view. Which we'll create right now. Save the controller and open a new file in the text editor and type:

```
<b>Hello World!</b>
```

Save the file as WebRoot/helloworld/protected/views/site/index.php overwriting the current index.php which is already placed there.

We'll do a small hack here which I will explain later on: rename the directory:

WebRoot/helloworld/protected/views/layouts

to

WebRoot/helloworld/protected/views/layouts-backup

Then access your page and there you are, same result, a bit prettier, using a view.
Happy Yiiing!

Notes

By default yii applications make use of the default layout file which is called main.php in the layouts directory we renamed. The file contains a few elements like a header and a footer as well as a “component” which is the menu we saw on top when first accessing our application. Those parts were not needed for our application, the fastest way to remove them is to get rid of the layout, or to edit it to whatever you want to.

The directory structure of our application is much fatter than the directory structure of the demo under YiiRoot/demos/helloworld. You can actually skin it down to obtain a very similar result although you have to pay attention when removing certain directories:

- - the default demo doesn't have views, we added our own index.php view, so the views directory should not be removed (although you can remove the other views).
- - you will always need the file WebRoot/helloworld/index.php, but the only directory needed there is the “protected” directory.
- - the default index.php file includes the default configuration file which is found at WebRoot/helloworld/protected/config/main.php. Hint: compare your index.php with the demo one and find out why we need more directories than the demo does.