JakartaTurbine2ScreensAndActions

The Turbine framework provides a model of classes that helps you represent the structure and behavior of a web-application, the execution of such elements is "controlled" by a MVC controller, which is the Turbine servlet. In order to start working with Turbine it is a good idea to have a conceptual model of the elements available in the Turbine framework. The functional specification provides a good description of the available elements in Turbine.

If you have read the abovementioned document, here is more information about how to use Screens and Actions:

Screens: As mentioned in the functional specification the Screen module is essentially considered the "body" of the webpage. Typically, the screen is used to put in the Context the information required by the template to be shown. This will occur in the doBuildTemplate(RunData, Context) which you must implement in your Screen, here is an example:

```
package homepage.modules.screens;

import org.apache.turbine.modules.screens.VelocityScreen;
import org.apache.turbine.util.RunData;
import org.apache.velocity.context.Context;

public class MyHomePage extends VelocityScreen {

        protected void doBuildTemplate(RunData data, Context context)
                throws Exception {

                context.put("title", "My Home Page");

{{{                //my list of favorite musicians will appear as a select in the webpage
```

List musicians = new Vector();

```
                musicians.add("Jimi Hendrix");
                musicians.add("Edvard Grieg");
                musicians.add("Mirways");
```

```
                //the velocity template will render the list as a select
```

context.put("musicians", musicians);

}

}
}}}

If you have configured the TurbineResources to find this class according to its package declaration, then everytime you call a template named MyHomePag
(e.g.: MyHomePage.vm) this Screen will be executed automatically. If you do not wish your template to be atached to a Screen, then you have to define a Default and inform this to the TurbineResources

Actions: Actions represent the execution of a specific task, typically, it's name is a verb such as "Login", "Search", et cetera. So as you can imagine, this allows you to separate the code that has to do with the functionality of your webapp from the code that configures how it looks. The following is an example of an Action:

```
// Turbine Stuff
import org.apache.turbine.util.RunData;
import org.apache.velocity.context.Context;
import org.apache.turbine.modules.actions.VelocityAction;

public class SendEmail extends VelocityAction {
{{{    public void doPerform( RunData data, Context context ) throws Exception {

        //here goes the code that sends an email...

    }
```

}
}}}

As always, you have to set in the TurbineResources file the location of the modules. If you haven't done this properly, Turbine won't find the Action and it will throw an exception. Otherwise, you may call it with an URL like this one: http://yourserver/yourapp/action/SendEmail