Unit Test

Unit Test

Unit testing with Wicket is done by mocking the application. This means that you do not have a webserver running. This section will be about how to get started using unit testing with Wicket's testing framework.

Simple example

```
public class AgencyPageTest extends TestCase {

    public void testBasicRender() {
         WicketTester tester=new WicketTester();
         tester.startPage(myPage.class);
         tester.assertRenderedPage(myPage.class);
        }
}
```

Above example makes sure that your page can render. This in itself is a very powerful test & and should be the minimum of tests on all your pages (in this author's opinion).

Basic Test Case for some labels and elements ...

```
/**
 *
 * <p>
 * Note : This is a TestNG Test case, a JUNIT 4.x implementation should be similiar to do.
 * For more on testNG refer to <a href="http://testng.org/doc/">TestNG</a>.
 * Also, note that i have kept the underneath stuff to minimal logic, but more
 * to show what the framework testing is like ...
 * </p>
 *
 * Code for Actor Panel below is :<br/>
 *
 * <code><br/>
 * public ActorPanel(String id) {<br/>
 *                super(id);<br/>
 *
 * // add actor name<br/>
 *        add(new Label("actorId", "" + getActorData().getId()));<br/>
 *        add(new Label("actorName", getActorData().getActorName()));<br/>
 *        add(new Label("actorDepartment", getActorData().getDepartment()));<br/>
 *        add(new DropDownChoice("department", Arrays.<String>asList("HR","IS","FINANCE/ACCOUNTING")));<br/>
 *}<br/>
 * </code>
 *
 * In HomePage class you do a simple add(new ActorPanel());<br/>
 *
 *
 * @author Vyas, Anirudh
 *
 */
public class HomePageTest {

        public WicketTester wicketTester = null;

        /**
         * Basic Setup of our test
         */
        @BeforeClass
        public void testSetup(){
                wicketTester  = new WicketTester();  // <-- A utility inside Wicket Framework for testing ...
                wicketTester.startPage(HomePage.class); // <--- You dont have to start a web server to test
your stuff because of this ... =)
```

```
	}

	/**
	 * Basic Home Page Test case for our application...
	 */
	@Test
	public void testHomePage(){


		// $Id - Test Case 1.0 : ... is Home Page rendered?
		wicketTester.assertRenderedPage(HomePage.class);

		// $Id - Test Case 2.0 : Elements test cases ... ( Starts the Panel ... prepares a DummyPanel
first for testing ).
		wicketTester.startPanel(ActorPanel.class);
		System.out.println(wicketTester.getApplication().getHomePage());

		//$Id - Assertion ( Element - Label - actorDepartment ) --> Not Null
		Assert.assertNotNull(wicketTester.getTagByWicketId("actorDepartment").getValue());

		//$Id - Assertion ( Element - Label - actorDepartment ) --> Same as "HR"
		Assert.assertEquals(wicketTester.getTagByWicketId("actorDepartment").getValue(), "HR");

		//$Id - Assertion ( Element - Label - actorName ) --> Not Null
		Assert.assertNotNull(wicketTester.getTagByWicketId("actorName").getValue());

		// $Id - Assertion( Element - Label - actorName ) --> Same as "|| OM Gam Ganapataye Namaha ||"
		Assert.assertEquals(wicketTester.getTagByWicketId("actorName").getValue(), "|| OM Gam
Ganapataye Namaha ||");

		//$Id - Assertion ( Element - DropdownChoice (select) - department ) --> Not Null
		Assert.assertNotNull(wicketTester.getTagByWicketId("department").getValue());

		//$Id - Assertion ( Element - DropdownChoice (select) - department ) --> Not Null
		Assert.assertNotNull(wicketTester.getTagByWicketId("department").getValue());
		DropDownChoice dChoice = (DropDownChoice) wicketTester.getComponentFromLastRenderedPage("panel:
department"); // thanks Igor =)

		//$Id - Assertion ( Element - DropdownChoice (select) - department ) --> 3 values
		Assert.assertEquals(dChoice.getChoices().size(),3);


	}

}
```

Note that in above test, should you have panel without a default constructor, then you'd have to do the following :

```
TestPanelSource testPanelSource = new TestPanelSource(){

			private static final long serialVersionUID = 1L;

			public Panel getTestPanel(String panelId){

				// prepare the SUT (which is Panel in this case)
				return new AssociateInformationPanel(panelId);

			}
		};

		tester.startPanel(testPanelSource);

		// $Id - Assertion( Element - label - associateName) --> Assert label Values
		tester.assertLabel("panel:labelValueAssociateName", "LastName,FirstName");
```

Similiar to above, if you have a page without a default constructor; you would do the same; this time creating an anonymous ITestPageSource implementation.

In above test, as you can make out, first a WicketTester class is instantiated, This would create a DummyWebApplication instance, which will have new SessionStore, Response etc. and a DummyHomePage in it. A call to startPage method on wicket Tester initiates the cycle. First Mock request and response are setup,then a new RequestCycle is created followed by regular request processing (for details see Lifecycle section of documentation).

A note on paths : As in above assertLabel( ) we have to precede the actual label name or component name with container,in this case a panel.