Testing

Approaches to Testing

There are different kinds of tests and different approaches to testing. No one technology or approach will work for everyone. Different tools are able to run different kinds of tests. For example, one can run database tests with junit. Anything that can be run from a command-line might be a good candidate for a junit test suite. If you need to verify that certain specific things show up on the web pages served by your application, Selenium may be what you want. If you separate the testing of the logic of your application and the display of your application, which an MVC design would encourage, you will see which tool to use where. WOUnit seems to have elements of both junit-ish testing and Selenium-ish testing. (Is this true? Can a WOUnit user explain?)

"Framework testing" is testing that is oriented to testing the WebObjects frameworks or the WOnder frameworks themselves. "Application testing" is testing of an application or applications that are built using WebObjects or WOnder frameworks. There is obviously some overlap here. If one has an application, but with Wonder, one is obviously using and therefore implicitly testing both the WebObjects and WOnder frameworks. "Framework testing" is explicitly testing just the framework functionality in as general a way as possible. There are tests which seem to straddle these two. But one can look at the intent. For example, two of the applications in Project WOnder are the "AjaxExample" and "BugTracker." The AjaxExample application is just a list of pages showing things that can be done. Nobody would construct an app like this for real work. On the other hand, the BugTracker app was created to be used by real people for tracking real bugs. It also demonstrates and provides a test for advanced features in WebObjects and WOnder. An application like AjaxExample can be easily changed so that it is easy to test. BugTracker cannot be so easily changed. On the other hand, BugTracker is more like a real-world application and so we care about it a bit more than a demonstration project. It may end up being less convenient to test BugTracker, but it may be more important to test it.

There may also be a distinction between API tests, functional tests and performance tests. In functional testing, one looks at some interface to an application and tests it to see if it does what it does. For example, if one launches the BugTracker app and clicks things and checks what they do, that is functional testing. If one looks at the API of a class, such as the er.extensions.foundation.ERXThreadStorage class, and determines what methods can be called on it and calls them, that is API testing. One can usually do API testing from a command-line interface and junit is probably the tool of choice for this. Functional testing is not so simple. One can manually exercise an app, which may be called the "clicks and eyeballs" approach. This works well, but does not scale. It becomes dull, people miss things, and we do not have robots to do it for us. One can use a test runner to interact with a running WebObjects/WOnder application. Performance testing often looks like functional testing, but the focus is different. It is not on seeing if things work, but how fast they work. In performance testing, one may have to use measurement tools which are not resistant to errors. In other words, performance testing may only work when everything else has been tested. One can do performance and functional testing at the same time, but perhaps one should not.

Tools

Here are some relevant tools. It may be useful to search for their names in this wiki.

- Junit and TestNG - for testing of user applications and for WOnder framework testing
- Selenium - for both user and WOnder framework testing
- WOUnit - for unit and integration testing
- WOUnitTest - for functional testing (Is WOUnitTest current and being maintained? -rrk)
- JMeter - for performance testing
- Thrash Testing - for testing threaded operations in EOF

Questions still to be answered include the different approaches one must take to different kinds of applications. For example, testing a "regular" WOnder app may be different than testing a "DirectToWeb" app, which is also different than testing a Java Client or a "Web Services" application. Different application types make some things easier and some things harder. One may need different approaches or different tools. Also, one may be deploying an app as a regular java application, or as a servlet in a J2EE container (e.g. Tomcat) or the application may be managed in some other way. How an application is deployed may change how it needs to be tested.

Document Review

There are many documents relevant to testing that should be evaluated for veracity, usefulness and relevance:

- Testing-JUnit and TestNG
- Web Services-Testing Services with Terminal
- Web Applications-Development-Testing and JUnit
- Testing-Load Testing WO Apps with JMeter
- Testing-WOUnitTest
- Web Applications-Development-Testing and JUnit