TestingAndBuilding

Introduction

Making a change to the JDO source, testing it and building a new version can involve quite a few steps. This page intends to guide you through the process, and documents some of the things I've learned along the way. Much thanks to the other members of the JDO crew for all your help in mentoring me into this process.

Assumptions

I make a few assumptions that you should know about:

1. You've already checked out or downloaded the JDO source code. Get it at http://db.apache.org/jdo/svn.html.
2. You have your build tools installed. You'll need Ant (the latest version is fine), and Maven 1.1 (and **NOT** Maven 2). get Ant Here. get Maven 1.1 here.
3. You have successfully built the JDO code without your modifications first, and then with your modifications.
4. You have built a JUnit test before. So, get used to JUnit if you haven't already. See The JUnit Website for more download.

If you haven't done these things already, then stop reading for now and go do them. Then, come back to this page. Otherwise, these instructions aren't going to help you too much - they will merely provide some insight on the build process with little practical experience.

Steps In Adding A New Test To the JDO Source Tree

You've checked out the latest version of the JDO source code from the SVN repository (or downloaded it), and have made modifications which you have successfully built using Maven command line tools. Your work isn't done yet! Now you need to build a test to be certain your change works. This is as important as making the change itself, and a requirement of the JDO project to get your changes accepted.

The following steps, which are discussed below, are what you'll be doing to set up and execute a successful JDO test:

1. Familiarize yourself with the test environment
2. Write your test code.
3. Set up configuration files.
4. Run your test.
5. Examine the output of your test.

1. Familiarize yourself with the test environment

You're going to save yourself a lot of time if you take a few moments to examine the directory structure associated with the test code. Actually, I just lied. You're going to take more than a few moments - you going take as much time as you need to visit each directory in the test environment and get familiar with the files and their reasons for being there. Give yourself a day or two to go through all that. And, as much as it sounds like you're getting lazy, don't just rely on your command line tools and text editors to explore the JDO code base. You should use your favorite IDE just to cruise around the directory tree and familiarize yourself with what you see.

My preferred way of working code on a project is to just use the lowest common denominator of tools to get the job done. This usually means the VI editor for writing code, and build tools (e.g. compilers and repository management tools) that can be run from a simple UNIX command line. Keeping things simple like this means you don't need more than that to get your work done, and quite frankly, it's a much more efficient way to work because you don't have to mess around with windowing GUIs, IDEs and the like. And, you get the added benefit of having your working build environment be identical to the production build environment (which runs nightly in batch on a different box). But, there are exceptions where using an IDE can help you learn what you need to learn about a project as quickly as possible. Working the JDO code is one such exception. There are a lot of files in the JDO code base, and it's helpful to have the IDE interpret the project's files so it can lay things out graphically.

So, that's what we'll do in the first step. I'll walk you through the code base as it can be seen from the Netbeans IDE.

Once you are familiar with the JDO code base and test environment, you can go back to the old school approach of command lines and text editors.

2. Write your test code.

Now that you've familiarized yourself with the test environment, you are ready to write some test code.

More coming soon ...

3. Set up configuration files.

More coming soon ...

4. Run your test.

More coming soon ...

5. Examine the output of your test.

More coming soon ...