UnitTestingWithTorque

Unit-testing with Torque can take two forms:

- the DbUnit approach
- testing the contents of the Criteria object

DbUnit Testing

This is a straight-forward application of the DbUnit package. Populate a test environment in the database using DbUnit, run the code under test, determine if the correct records were returned. It's easy enough but the runtime costs add up, and (in my case) leads to development of a fairly large body of code that does nothing but prepare datasets for testing. The biggest downside of this approach is that it tests Torque, and we'd like to assume that Torque works, and test our instructions to Torque without testing Torque itself. Perhaps the DbUnit tests have a place as occasionally run validations of the schema, but even there I think it would be better to deal with the schema file directly and assume that Torque does the right thing with it.

Criteria Testing

Unit-testing with the Criteria object satisfies the desire to test the instructions given to Torque without testing Torque. It is a little tricky but very fast at runtime. I've struggled with one simple fact about the Criteria object: the predicate (OR, AND, IN, etc.) information is in a package-scoped class. As a result, my natural tendency to dig around in the Criteria object and find what I need has been blocked. Hmmmm, there must be another way.

Here are a variety of methods for testing the contents of the Criteria class:

- simple assertions
- a helper class devoted to introspecting the Criteria class
- creating a Criteria that the Criteria-under-test should match, and comparing them for equality.
- comparing the toString() output of Criteria

Simple Assertions

Simple assertions work great unless dealing with a non-trivial query. They rapidly become exercises in understanding the internal construction of the Criteria object, which eventually run into the packaged-scope predicate problem.

Helper Class

I've been experimenting with a helper class that knows how to introspect the Criteria object, but it's an ugly job and of course vulnerable to changes in the Criteria class. Not to mention that once again, the packaged-scope predicates will prevent complete testing.

Criteria.equal()

Could be the elegant solution, if equal() is written to ignore ordering issues. Simply create the correct Criteria object, and check if the Criteria object generated by the code under test is equal() to it. As of today (2004/05/17) I am experiencing mismatches between two Criteria containing identical IN values for the same field, the difference being that one was built from a List of Integer, and the other built from an array of int. According to the dev list, this (if true of course 🙂 would be a bug. I'll attempt to find the problem and will update this entry accordingly.

Criteria.toString()

Haven't looked at the code to see if there's any canonicalization being done, so that differently entered, but identical, statements generate identical strings. As of today (2004/05/17) I have one example of a pair of Criteria which are identical except for details of construction, for which the toString() values match but equal() returns false.