# *Recipe 11.12 Test stored procedures*

## *Problem*

You would like to test stored procedures.

## *Background*

When your application implements some of its business logic in stored procedures then you *need* to test against a live database in order to test that business logic. We recognize that this allows the database to optimize execution of business rules for improved performance; however this makes it more difficult to test business logic both by placing it in a difficult-to-test resource and potentially putting some business logic in the database and other business logic in objects. We recommend keeping business logic where it can more easily be tested. It is, after all, one of the most important aspects of your software.

If instead your application uses stored procedures only as a way to hide database schema details through a CRUD-style interface, then we applaud the people who made that decision. These stored procedures relieve the data access layer of the burden of generating correct SQL commands, leaving only data mapping. We think this is a wise choice, since it makes testing easier!

It may not be possible for you to do at this time, but if you can refactor away from the first case towards the second base, we recommend it.

## *Recipe*

If you can help it, do not test the behavior of a stored procedure with JUnit. We find that JDBC is too verbose for such a simple task, so instead use shell scripts and your vendor's SQL command line tool to run tests. Do not worry whether you have a framework for your shell. Instead, simply start with a test that feeds some SQL into your database, queries the result, compares against some known value and uses return codes or error levels to signal pass/fail. We recommend books like *Windows NT Shell Scripting* and *Linux and Unix Shell Programming* for a thorough treatment of using shell scripts do to testing.

**BashUnit?** Curtis Cooley tells the story of writing tests for a Unix shell. The moral of the story is this: you do not need a framework; you need tests. The framework will appear.

I just got done with a fun experiment. I was working with a DBA who groks (Extreme Programming) ... and he was trying to get a bunch of shell scripts working. The scripts were part of the install and backup procedures for the database.

He was struggling with trying to figure out how they worked and what their intent was. I joked "we should unit test these. it shouldn't be that hard to write a shell framework."

After a couple of minutes of frustration he said "let's go for it."

So for the next day and a half, we wrote unit tests for `bash` scripts, developing a little (unit test) framework along the way.

The tests found many minor errors in the scripts that would have taken quite a while to track down.

In a day and a half we completely rewrote and tested a bunch of scripts that originally took another DBA a couple weeks.

Should you be required for some reason to test your stored procedures from Java, the best you can do is set up some data, invoke the `CallableStatement` then check the results. See [Reference:Recipe:Test legacy JDBC code with the database] for an example of such a test. Be careful to maintain your test data and clean up your JDBC resources.

If you want to verify that you *invoke* a stored procedure correctly, then we recommend testing with `MockCallableStatement` objects. Verify that the callable statement receives the appropriate parameters and that the invoking code was able to handle the different kinds of parameters the callable statement passes back. Here is an example.

[JBR: Insert example: Coffee Shop/add order item]

## *Discussion*

We do not recommend using JUnit to test stored procedures, simply because writing them in Java at all puts an unnecessary layer of complexity between the tester (you) and the code being tested. This extra layer of complexity takes effort to build, and we have illustrated throughout this chapter. A JUnit test for a stored procedure will often include inserting data into the database, calling the procedure, then cleaning up the database. The resulting JDBC code presents nothing of value: just a verbose layer around SQL that includes translating data types, handling null values... all the things about JDBC that annoy many Java programmers. On the whole, testing stored procedures is best left to database-centered tools and approaches, such as shell scripting.

Hey, if you *need* to write the tests in Java because it is the path of least resistance, then by all means, but we have experienced more return on investment from implementing these tests as shell scripts.

*Do not*, however, mistake the preceding statement as "do not test your stored procedures." Code is code and you need to test it, otherwise, when the application fails, how will you know whether it is the application or the stored procedures at fault? Test them with JUnit if you do not have the skill to do it through shell scripting, but we believe you will save effort by learning to write shell scripts for these tests.

## *Related*

<Later>