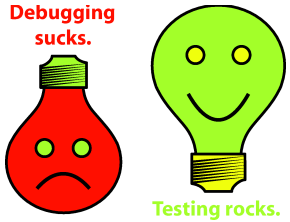


## Testing on the Toilet

# Stubs Speed up Your Unit Tests

April 4, 2007

Michael Feathers defines the qualities of a good unit test as: “they run fast, they help us localize problems.” This can be hard to accomplish when your code accesses a database, hits another server, is time-dependent, etc.

By substituting custom objects for some of your module's dependencies, you can thoroughly test your code, increase your coverage, and still run in less than a second. You can even simulate rare scenarios like database failures and test your error handling code.

A variety of different terms are used to refer to these “custom objects”. In an effort to clarify the vocabulary, Gerard Meszaros provides the following definitions:

- **Test Double** is a generic term for any test object that replaces a production object.
- **Dummy** objects are passed around but not actually used. They are usually fillers for parameter lists.
- **Fakes** have working implementations, but take some shortcut (e.g., **`InMemoryDatabase`**).
- **Stubs** provide canned answers to calls made during a test.
- **Mocks** have expectations which form a specification of the calls they do and do not receive.

For example, to test a simple method like **`getIdPrefix()`** in the **`IdGetter`** class:

```
public class IdGetter {  // Constructor omitted.
  public String getIdPrefix() {
    try {
      String s = db.selectString("select id from foo");
      return s.substring(0, 5);
    } catch (SQLException e) { return ""; }
  }
}
```

You could write:

```
  db.execute("create table foo (id varchar(40))");  // db created in setUp().
  db.execute("insert into foo (id) values ('hello world!')");
  IdGetter getter = new IdGetter(db);
  assertEquals("hello", getter.getIdPrefix());
```

The test above works but takes a relatively long time to run (network access), can be unreliable (db machine might be down), and makes it hard to test for errors. You can **avoid these pitfalls by using stubs**:

```
  public class StubDbThatReturnsId extends Database {
    public String selectString(String query) { return "hello world"; }
  }
  public class StubDbThatFails extends Database {
    public String selectString(String query) throws SQLException {
      throw new SQLException("Fake DB failure");
    }
  }
  public void testReturnsFirstFiveCharsOfId() throws Exception {
    IdGetter getter = new IdGetter(new StubDbThatReturnsId());
    assertEquals("hello", getter.getIdPrefix());
  }
  public void testReturnsEmptyStringIfIdNotFound() throws Exception {
    IdGetter getter = new IdGetter(new StubDbThatFails());
    assertEquals("", getter.getIdPrefix());
  }
```

**More information, feedback, and discussion:**
**http://googletesting.blogspot.com**