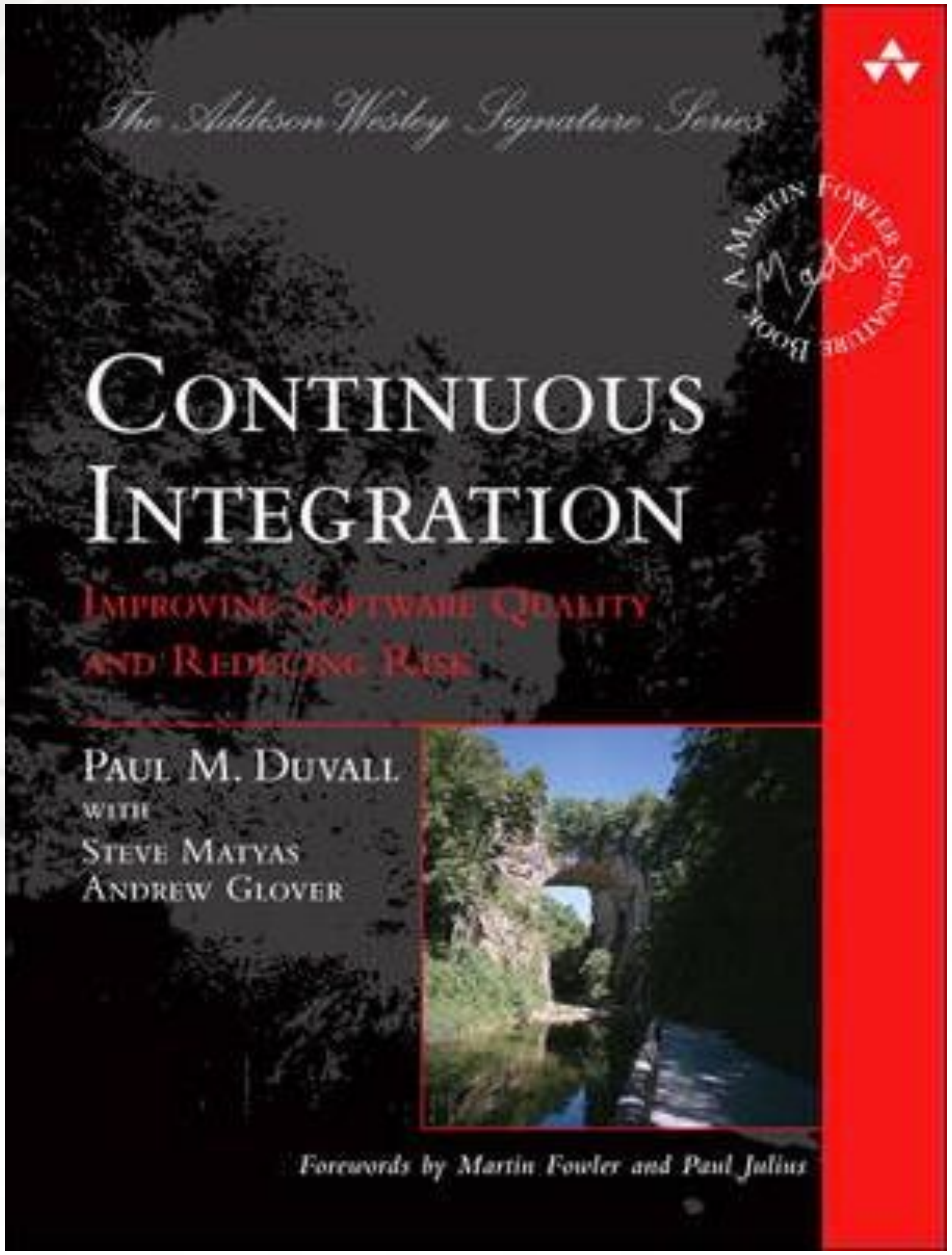
Continuous Testing

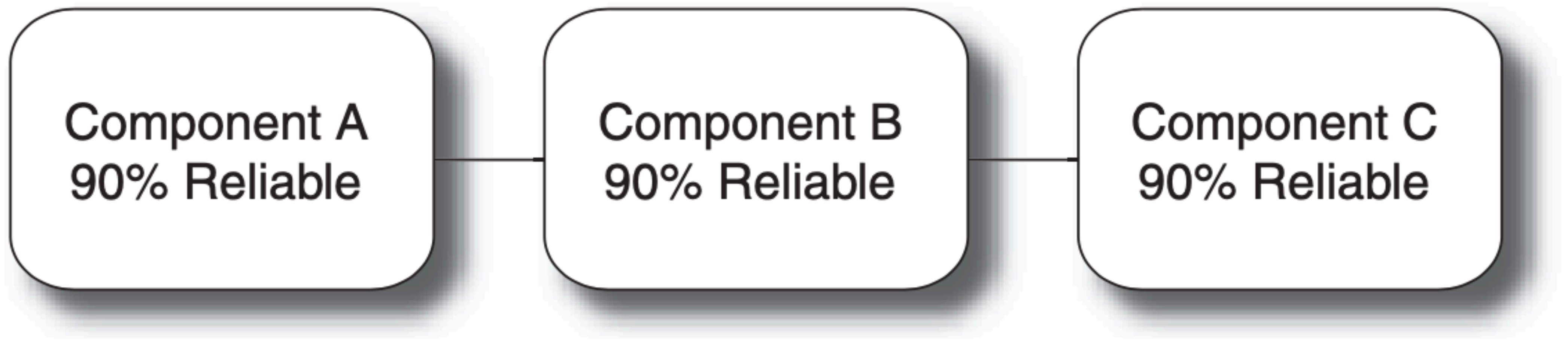
Continuous Integration


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Integration
Improving Software Quality and Reducing Risk
Paul M. Duvall
with
Steve Matyas
Andrew Glover
Forewords by Martin Fowler and Paul Julius

re·li·a·ble—adjective—Giving the same result in successive trials

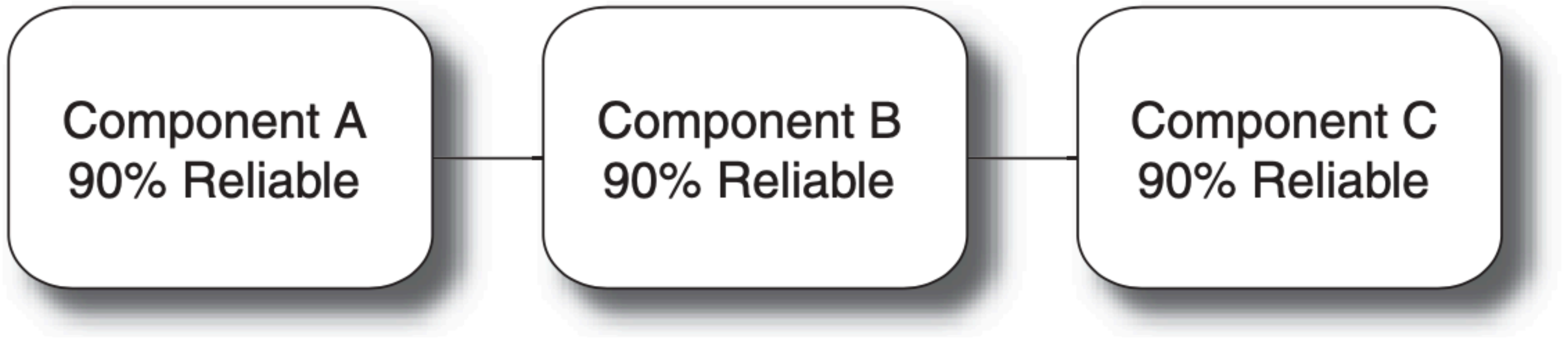
From www.m-w.com/cgi-bin/dictionary?va=reliable.

A system with three components



90%

A system with three components



~~90%~~

70%

Continuous Testing

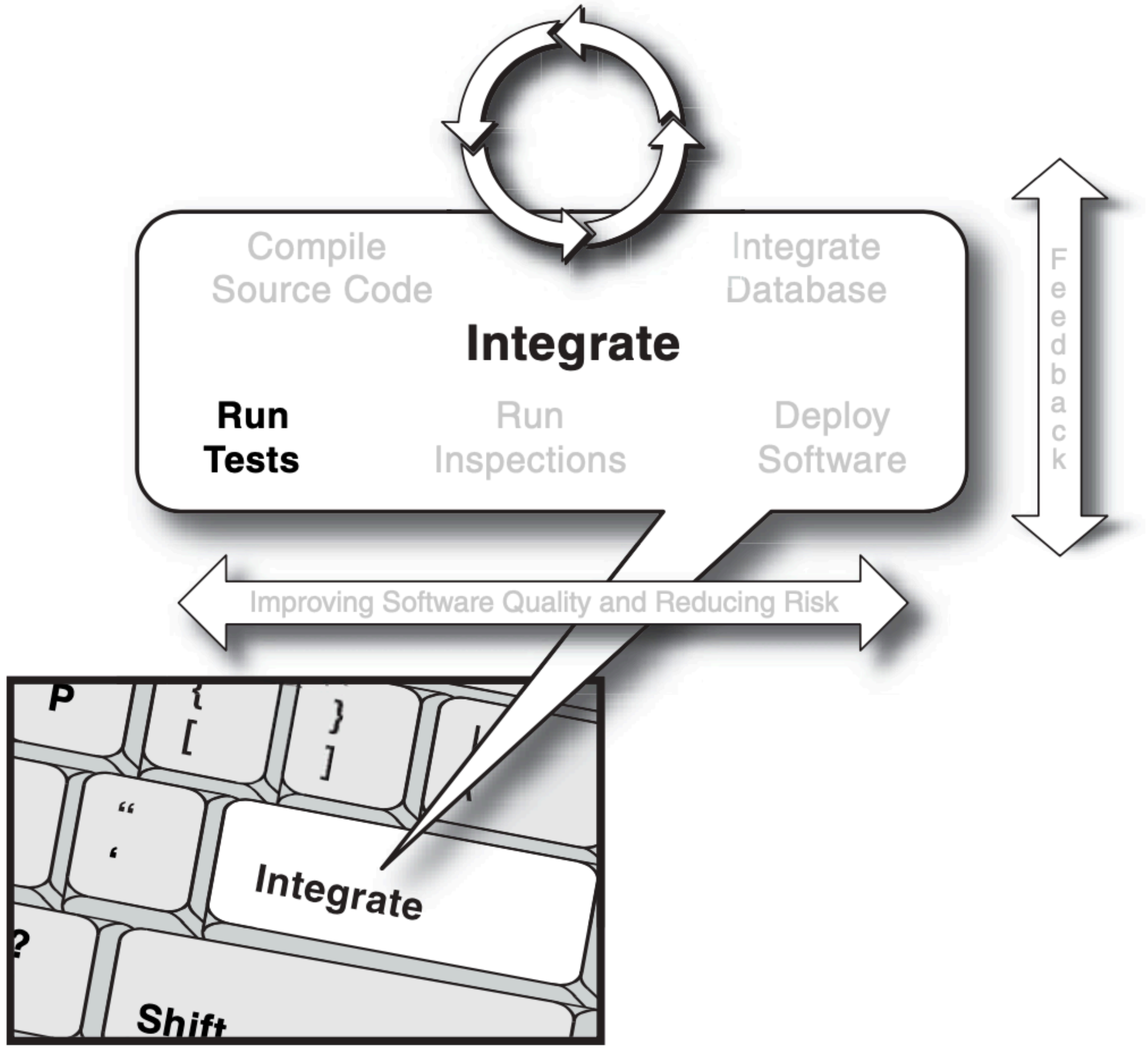
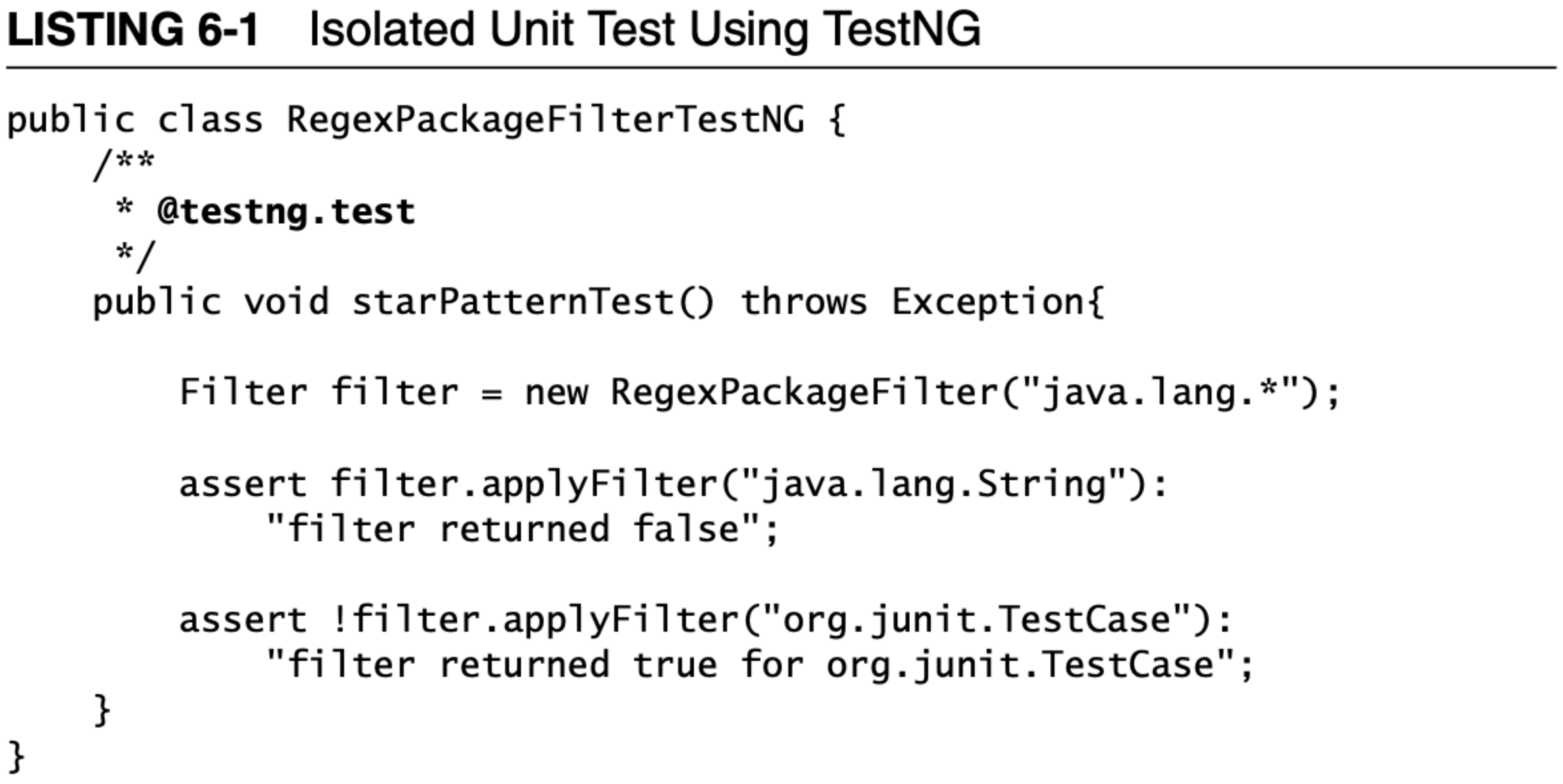


FIGURE 6-2 Integrate button—running automated developer tests

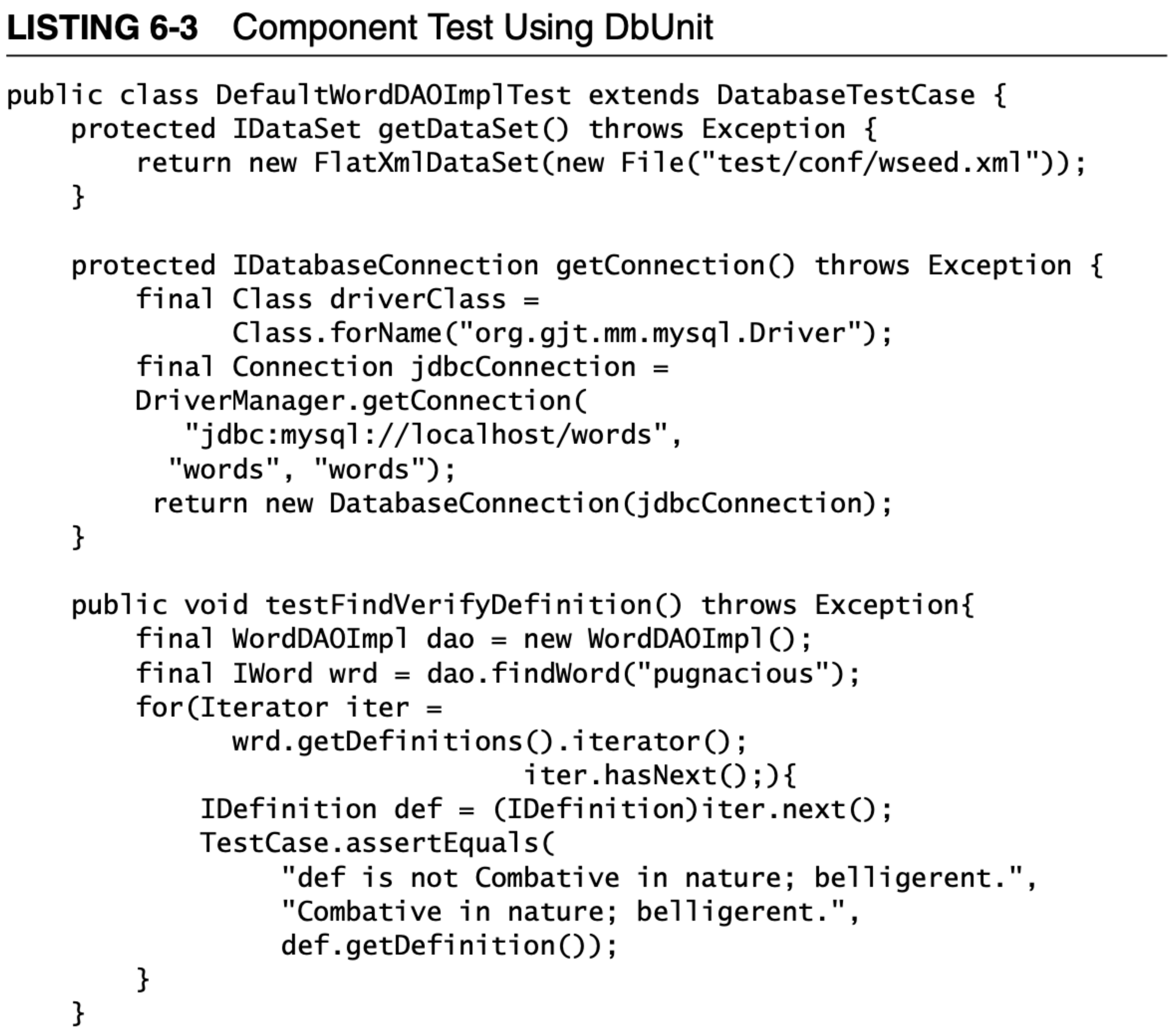
Automate Unit Tests



LISTING 6-1 Isolated Unit Test Using TestNG

```
public class RegexPackageFilterTestNG {
    /**
     * @testng.test
     */
    public void starPatternTest() throws Exception{

        Filter filter = new RegexPackageFilter("java.lang.*");

        assert filter.applyFilter("java.lang.String"):
            "filter returned false";

        assert !filter.applyFilter("org.junit.TestCase"):
            "filter returned true for org.junit.TestCase";
    }
}
```

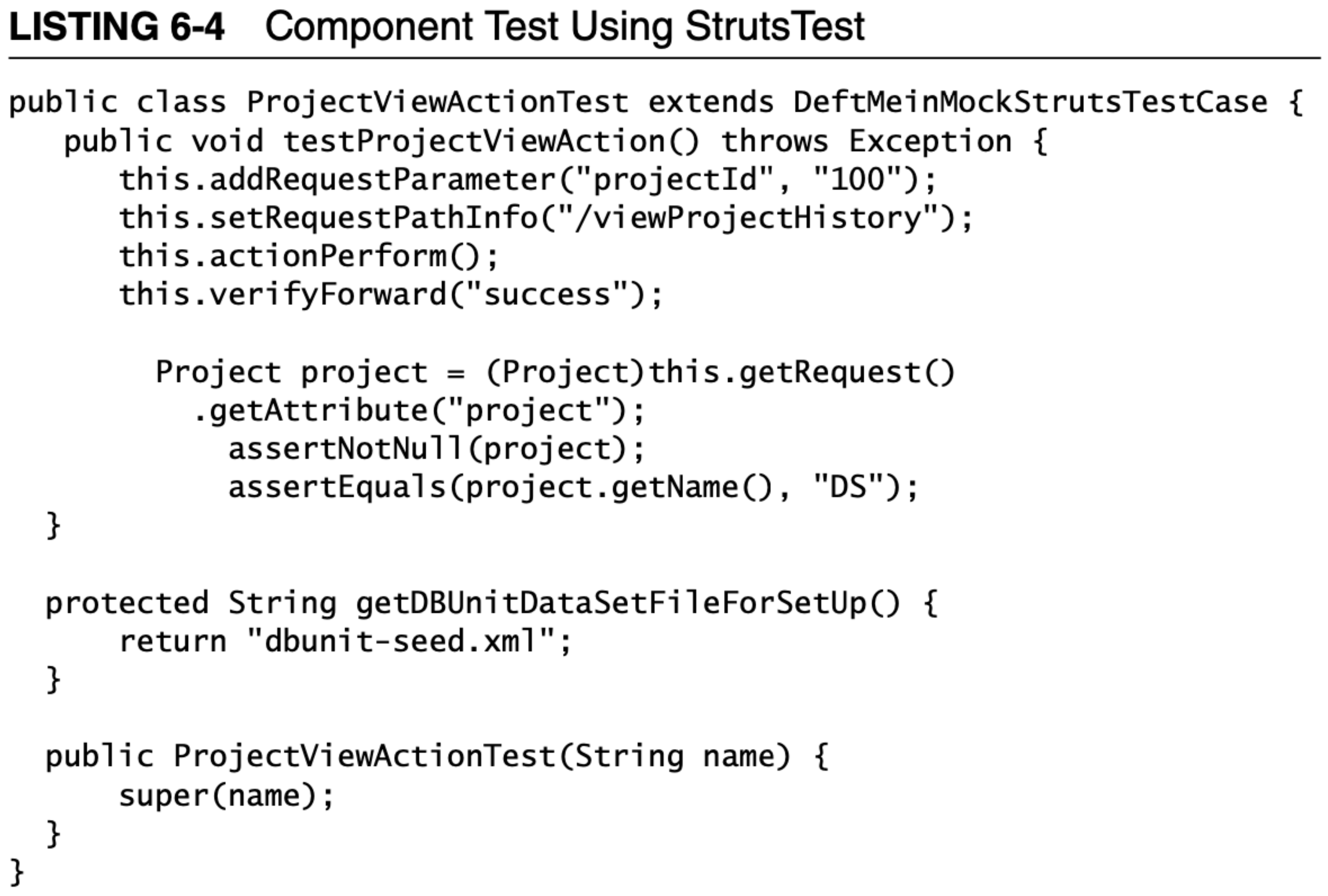
Automate Component Tests



LISTING 6-3 Component Test Using DbUnit

```
public class DefaultWordDAOImplTest extends DatabaseTestCase {
    protected IDataSet getDataSet() throws Exception {
        return new FlatXmlDataSet(new File("test/conf/wseed.xml"));
    }

    protected IDatabaseConnection getConnection() throws Exception {
        final Class driverClass =
              Class.forName("org.gjt.mm.mysql.Driver");
        final Connection jdbcConnection =
        DriverManager.getConnection(
           "jdbc:mysql://localhost/words",
          "words", "words");
         return new DatabaseConnection(jdbcConnection);
    }

    public void testFindVerifyDefinition() throws Exception{
        final WordDAOImpl dao = new WordDAOImpl();
        final IWord wrd = dao.findWord("pugnacious");
        for(Iterator iter =
              wrd.getDefinitions().iterator();
                                  iter.hasNext();){
            IDefinition def = (IDefinition)iter.next();
            TestCase.assertEquals(
                 "def is not Combative in nature; belligerent.",
                 "Combative in nature; belligerent.",
                 def.getDefinition());
        }
    }
```

Automate Component Tests



LISTING 6-4 Component Test Using StrutsTest

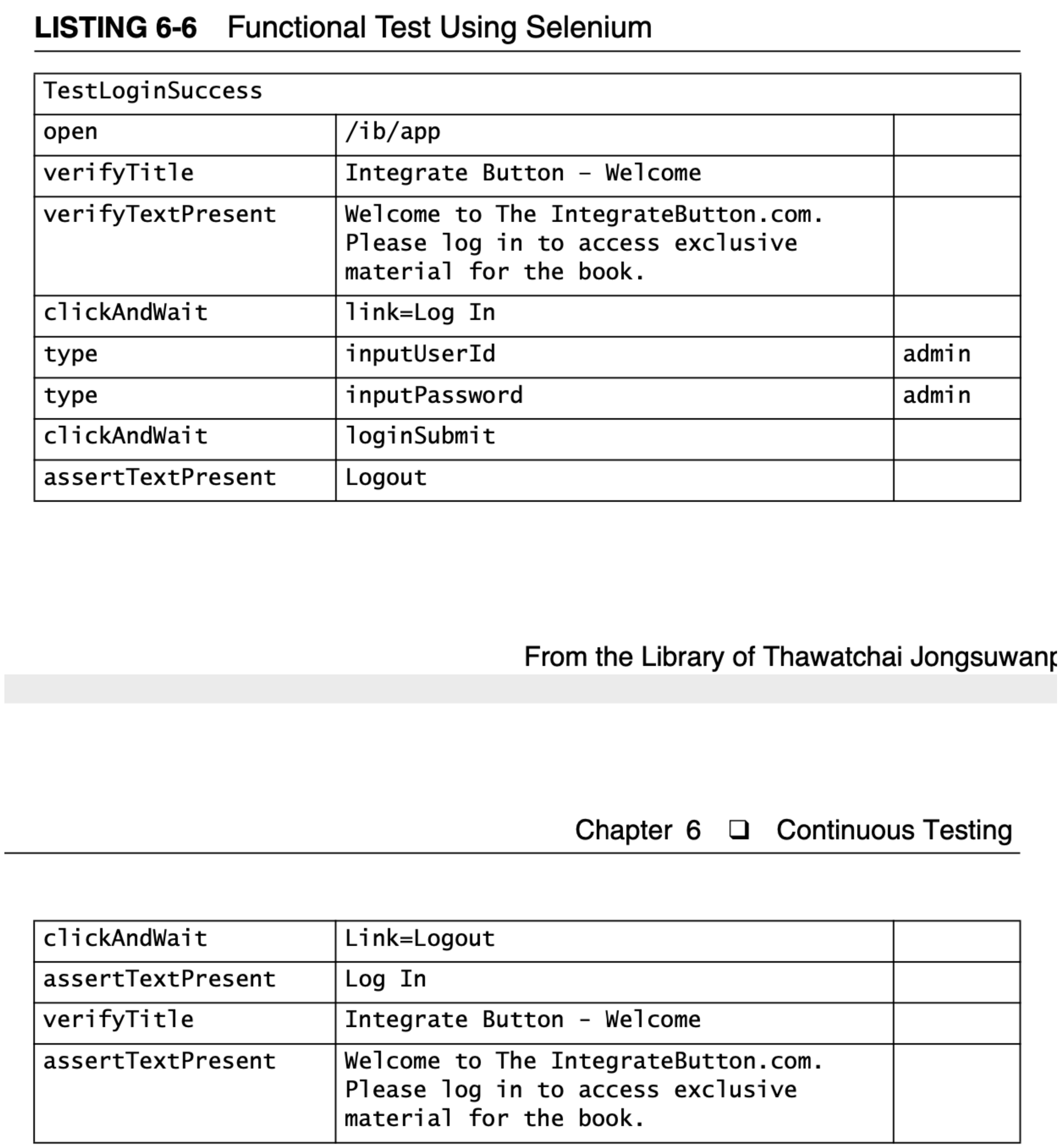
```
public class ProjectViewActionTest extends DeftMeinMockStrutsTestCase {
   public void testProjectViewAction() throws Exception {
      this.addRequestParameter("projectId", "100");
      this.setRequestPathInfo("/viewProjectHistory");
      this.actionPerform();
      this.verifyForward("success");

        Project project = (Project)this.getRequest()
          .getAttribute("project");
            assertNotNull(project);
            assertEquals(project.getName(), "DS");
  }

  protected String getDBUnitDataSetFileForSetUp() {
      return "dbunit-seed.xml";
  }

  public ProjectViewActionTest(String name) {
      super(name);
  }
}
```

Automate System Tests

LISTING 6-5 System Test Using JWebUnit

```
public class LoginTest extends WebTestCase {

  protected void setUp() throws Exception {
     getTestContext().
         setBaseUrl("http://pone.acme.com/meinst/");
  }

  public void testLogIn() {
    beginAt("/");
    setFormElement("j_username", "aader");
    setFormElement("j_password", "a1445");
    submit();
    assertTextPresent("Logged in as aader");
  }
}
```

Automate Functional Tests



LISTING 6-6 Functional Test Using Selenium

TestLoginSuccess		
open	/ib/app	
verifyTitle	Integrate Button – Welcome	
verifyTextPresent	Welcome to The IntegrateButton.com. Please log in to access exclusive material for the book.	
clickAndWait	link=Log In	
type	inputUserId	admin
type	inputPassword	admin
clickAndWait	loginSubmit	
assertTextPresent	Logout	
clickAndWait	Link=Logout	
assertTextPresent	Log In	
verifyTitle	Integrate Button - Welcome	
assertTextPresent	Welcome to The IntegrateButton.com. Please log in to access exclusive material for the book.	

Run Faster Tests First

Unit Test -> Component Test -> System Test (and Functional Test)

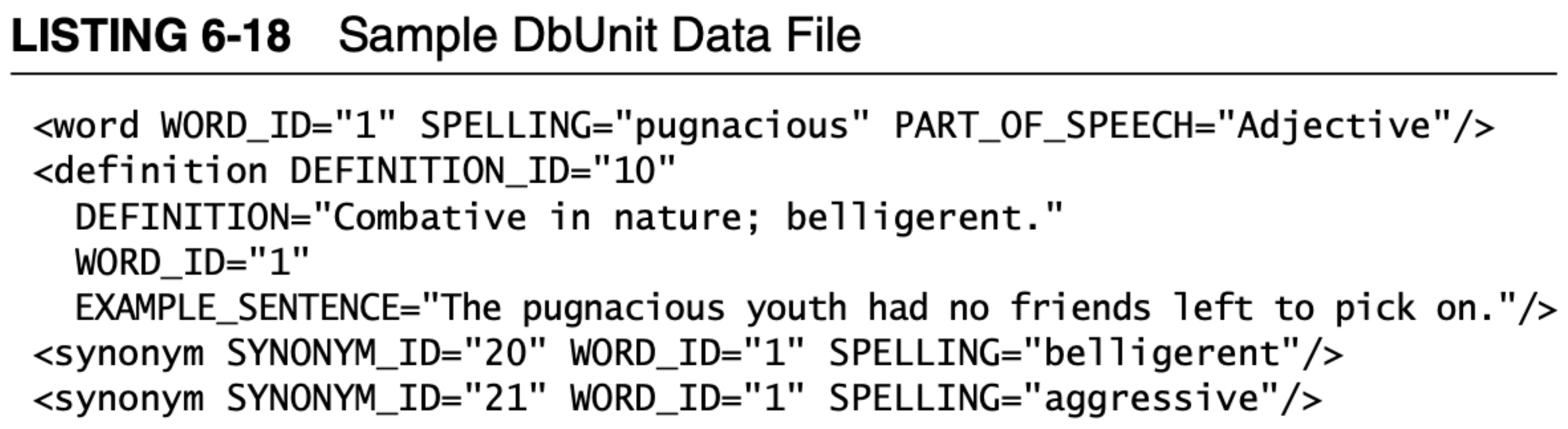
- Small
- Fast

- Integrate

- Running System

- User Behavior

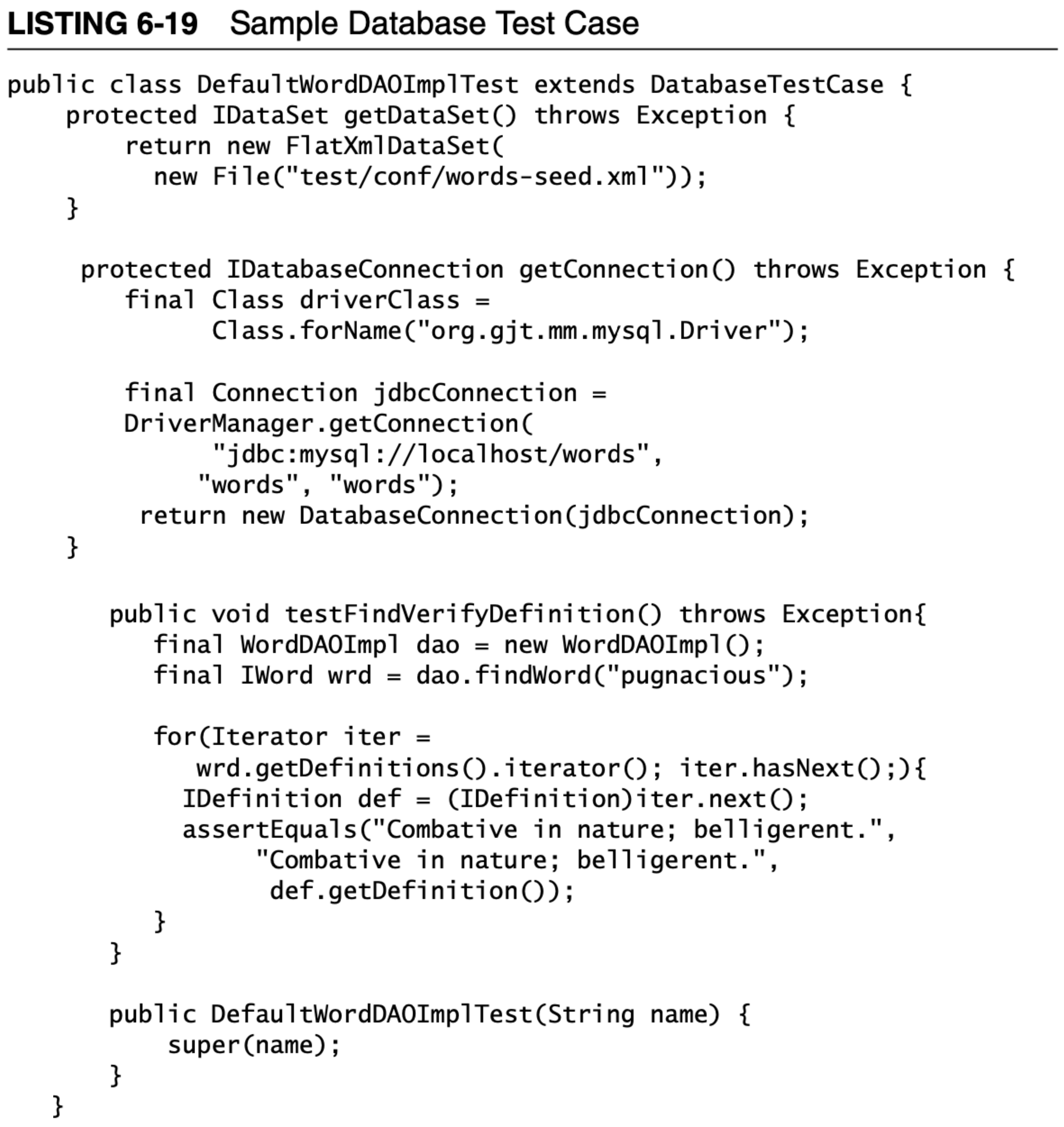
Make Component Tests Repeatable



LISTING 6-18 Sample DbUnit Data File

```
<word WORD_ID="1" SPELLING="pugnacious" PART_OF_SPEECH="Adjective"/>
<definition DEFINITION_ID="10"
  DEFINITION="Combative in nature; belligerent."
  WORD_ID="1"
  EXAMPLE_SENTENCE="The pugnacious youth had no friends left to pick on."/>
<synonym SYNONYM_ID="20" WORD_ID="1" SPELLING="belligerent"/>
<synonym SYNONYM_ID="21" WORD_ID="1" SPELLING="aggressive"/>
```

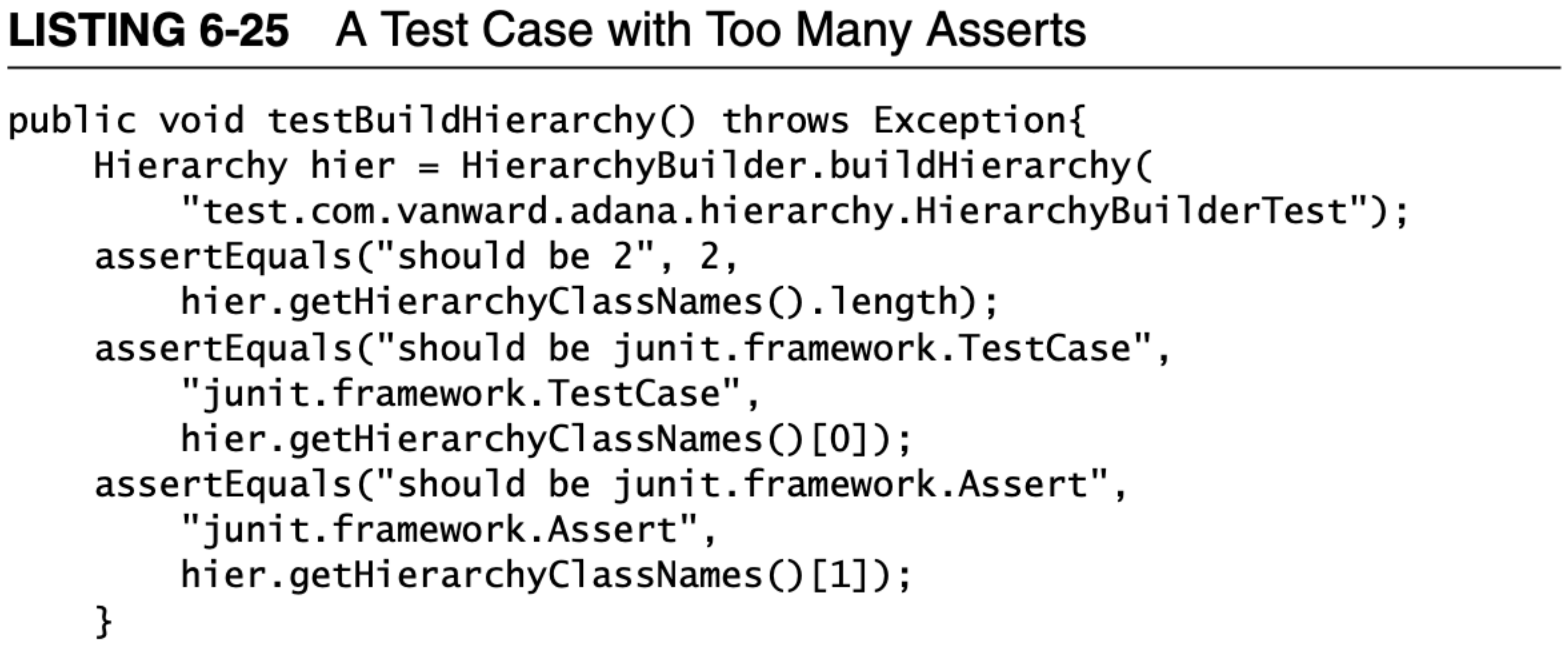
Make Component Tests Repeatable



LISTING 6-19 Sample Database Test Case

```
public class DefaultWordDAOImplTest extends DatabaseTestCase {
    protected IDataSet getDataSet() throws Exception {
        return new FlatXmlDataSet(
          new File("test/conf/words-seed.xml"));
    }

     protected IDatabaseConnection getConnection() throws Exception {
        final Class driverClass =
              Class.forName("org.gjt.mm.mysql.Driver");

        final Connection jdbcConnection =
        DriverManager.getConnection(
              "jdbc:mysql://localhost/words",
             "words", "words");
         return new DatabaseConnection(jdbcConnection);
    }

       public void testFindVerifyDefinition() throws Exception{
          final WordDAOImpl dao = new WordDAOImpl();
          final IWord wrd = dao.findWord("pugnacious");

          for(Iterator iter =
             wrd.getDefinitions().iterator(); iter.hasNext();){
            IDefinition def = (IDefinition)iter.next();
            assertEquals("Combative in nature; belligerent.",
                 "Combative in nature; belligerent.",
                  def.getDefinition());
          }
       }

       public DefaultWordDAOImplTest(String name) {
           super(name);
       }
   }
```

Limit Test Cases to One Assert



LISTING 6-25 A Test Case with Too Many Asserts

```
public void testBuildHierarchy() throws Exception{
    Hierarchy hier = HierarchyBuilder.buildHierarchy(
        "test.com.vanward.adana.hierarchy.HierarchyBuilderTest");
    assertEquals("should be 2", 2,
        hier.getHierarchyClassNames().length);
    assertEquals("should be junit.framework.TestCase",
        "junit.framework.TestCase",
        hier.getHierarchyClassNames()[0]);
    assertEquals("should be junit.framework.Assert",
        "junit.framework.Assert",
        hier.getHierarchyClassNames()[1]);
    }
```

Limit Test Cases to One Assert

LISTING 6-26 Test Case Refactoring

```
public final void testBuildHierarchyStrSize() throws Exception{
    Hierarchy hier = HierarchyBuilder.buildHierarchy(
     "test.com.vanward.adana.hierarchy.HierarchyBuilderTest");
     assertEquals("should be 2", 2,
        hier.getHierarchyClassNames().length);
}

public final void testBuildHierarchyStrNameAgain() throws Exception{
    Hierarchy hier = HierarchyBuilder.buildHierarchy(
      "test.com.vanward.adana.hierarchy.HierarchyBuilderTest");
    assertEquals("should be junit.framework.TestCase",
       "junit.framework.TestCase",
       hier.getHierarchyClassNames()[0]);
}

public final void testBuildHierarchyStrName() throws Exception{
    Hierarchy hier = HierarchyBuilder.buildHierarchy(
      "test.com.vanward.adana.hierarchy.HierarchyBuilderTest");
    assertEquals("should be junit.framework.Assert",
       "junit.framework.Assert",
       hier.getHierarchyClassNames()[1]);
}
```

AGILE TESTING FELLOWSHIP

สยาม ชำนาญกิจ

SIAM CHAMNAN KIT

DALCHEMIST

防 25 止 62