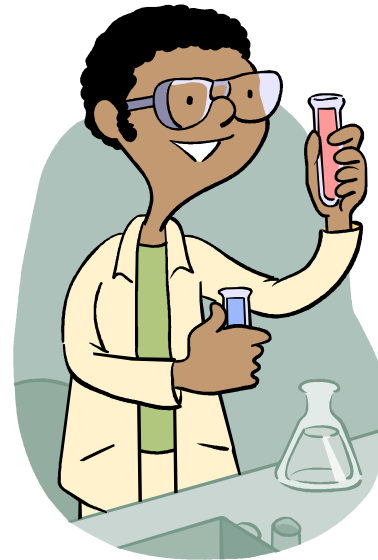


Testing OpenMRS Modules

With JUnit and Spring

Why test?

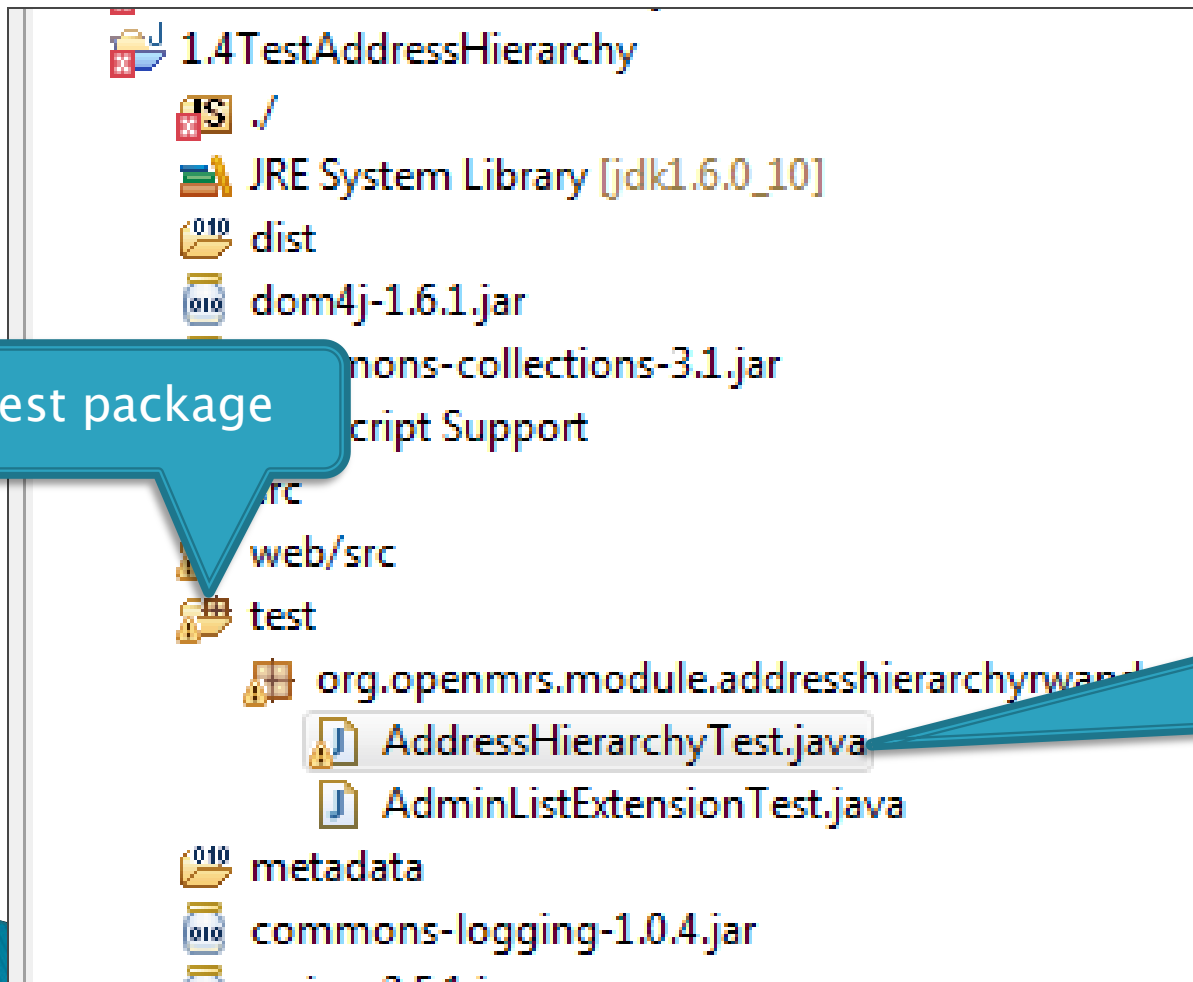
- ▶ Like any software, we need to KNOW it works the way it should
- ▶ Testing helps you out
 - It helps you think about what your code is doing
 - It's faster than loading a module every time you add functionality



What should you test?

- ▶ Generally, you should:
 - Write tests for your domain objects
 - Write tests for each of your controllers
 - Write tests for each of your service methods

Test Classes and Package



Test package

Testing class
for service
methods

JUnit Assert Statements

| Statement | Description |
|--|--|
| <code>fail(String)</code> | Let the method fail, might be usable to check that a certain part of the code is not reached. |
| <code>assertTrue(true);</code> | True |
| <code>assertEquals([String message], expected, actual)</code> | Test if the values are the same. Note: for arrays the reference is checked not the content of the arrays |
| <code>assertEquals([String message], expected, actual, tolerance)</code> | Usage for float and double; the tolerance are the number of decimals which must be the same |

| Statement | Description |
|--|--|
| <code>assertNull([message], object)</code> | Checks if the object is null |
| <code>assertNotNull([message], object)</code> | Check if the object is not null |
| <code>assertSame([String], expected, actual)</code> | Check if both variables refer to the same object |
| <code>assertNotSame([String], expected, actual)</code> | Check that both variables refer not to the same object |
| <code>assertTrue([message], boolean condition)</code> | Check if the boolean condition is true. |

JUnit TestCase

```
public class AdminListExtensionTest extends TestCase {  
  
    public void testValidatesLinks() {  
        AdminList ext = new AdminList();  
  
        Map<String, String> links = ext.getLinks();  
  
        assertNotNull("Some links should be returned", links);  
  
        assertTrue("There should be a positive number of links",  
links.values().size() > 0);  
    }  
  
    public void testMediaTypeIsHtml() {  
        AdminList ext = new AdminList();  
  
        assertTrue("The media type of this extension should be html",  
ext.getMediaType().equals(MEDIA_TYPE.html));  
    }  
}
```

JUnit Annotations

| Annotation | Description |
|--|---|
| <code>@Test public void method()</code> | Annotation <code>@Test</code> identifies that this method is a test method. |
| <code>@Before public void method()</code> | Will perform the method() before each test. This method can prepare the test environment, e.g. read input data, initialize the class) |
| <code>@After public void method()</code> | Test method must start with test |
| <code>@BeforeClass public void method()</code> | Will perform the method before the start of all tests. This can be used to perform time intensive activities for example be used to connect to a database |

| Annotation | Description |
|---|--|
| <code>@AfterClass public void method()</code> | Will perform the method after all tests have finished. This can be used to perform clean-up activities for example be used to disconnect to a database |
| <code>@Ignore</code> | Will ignore the test method, e.g. useful if the underlying code has been changed and the test has not yet been adapted or if the runtime of this test is just too long to be included. |
| <code>@Test(expected=IllegalArgumentException.class)</code> | Tests if the method throws the named exception |
| <code>@Test(timeout=100)</code> | Fails if the method takes longer than 100 milliseconds |

JUnit TestCase with Annotations

```
public class AppointmentWithTestCaseTest {  
  
    private Appointment appointment;  
  
    @Before  
    public void setUp() throws Exception {  
        appointment = new Appointment();  
    }  
  
    @After  
    public void tearDown() throws Exception {  
        appointment = null;  
    }  
}
```

```
@Test
public void testReason() {
    String reason = appointment.getReason();
    String reasonText = "I need an appt!";

    assertNull("Reason should be null", reason);

    appointment.setReason(reasonText);

    reason = appointment.getReason();

    assertNotNull("Reason should not be null", reason);
    assertTrue("Reason should be "+ reason,
reason.equals(reasonText));
}
}
```

JUnit TestSuite

- ▶ You can run each test manually one by one.
- ▶ If you have a large number of tests, this becomes time-consuming.
- ▶ Use a Test Suite to automatically group tests and run them together.

JUnit Test Suite

```
import junit.framework.Test;
import junit.framework.TestSuite;

public class AllTestsOld {

    public static Test suite() {
        TestSuite suite = new TestSuite("Test for
org.openmrs.module.appointments");
        //$JUnit-BEGIN$
        suite.addTestSuite(AdminListExtensionTest.class);
        //$JUnit-END$
        return suite;
    }
}
```

Junit Test Suite with Annotations

```
package org.openmrs.module.appointments;
```

```
import org.junit.runner.RunWith;
```

```
import org.junit.runners.Suite;
```

```
@RunWith(Suite.class)
```

```
@Suite.SuiteClasses({  
    AdminListExtensionTest.class,  
    AppointmentWithTestCaseTest.class  
})
```

```
public class AllTests {  
}
```

Testing OpenMRS Services

- ▶ If you want to test your services or anything that uses the Context class:
 - You must extend `BaseModuleContextSensitiveTest`
- ▶ OpenMRS has a testing framework:
 - Need to use it if you want to test services!
 - Sets up the database connections for you and initializes the context services

Extend BaseModuleContextSensitive

- ▶ Two Options
 - Use a Dynamically Created In – Memory Database
 - Use database specified by runtime properties file
- ▶ Required Configuration
 - You must put the three OpenMRS jars on your buildpath
 - You must put all the jars that OpenMRS uses on your buildpath as well!

Example: Extending BaseModuleContextSensitiveTest

```
public class AddressHierarchyTest extends BaseModuleContextSensitiveTest {
```

```
    @Override
```

```
    public Boolean useInMemoryDatabase() {
```

```
        return false;
```

```
    }
```

```
    @Test
```

```
    public void leafNodesShouldBeUmudugudus() throws Exception{
```

```
        AddressHierarchyService ahs = ((AddressHierarchyService) Context.getService(AddressHierarchyService.class));
```

```
        Assert.assertNotNull(ahs);
```

```
        AddressHierarchy ah = ahs.getNextComponent(30).get(0);
```

```
        List<AddressHierarchy> children = ahs.getLeafNodes(ah);
```

```
        for(AddressHierarchy leaf: children){
```

```
            Assert.assertTrue(leaf.getHierarchyType().getName().equals("Umudugudu"));
```

```
        }
```

```
    }
```

```
    @Test
```

```
    public void topOfListShouldBeCountries() throws Exception{
```

```
        AddressHierarchyService ahs = ((AddressHierarchyService) Context.getService(AddressHierarchyService.class));
```


```
        List<AddressHierarchy> countries = ahs.getTopOfHierarchyList();
```

```
        for(AddressHierarchy country: countries){
```

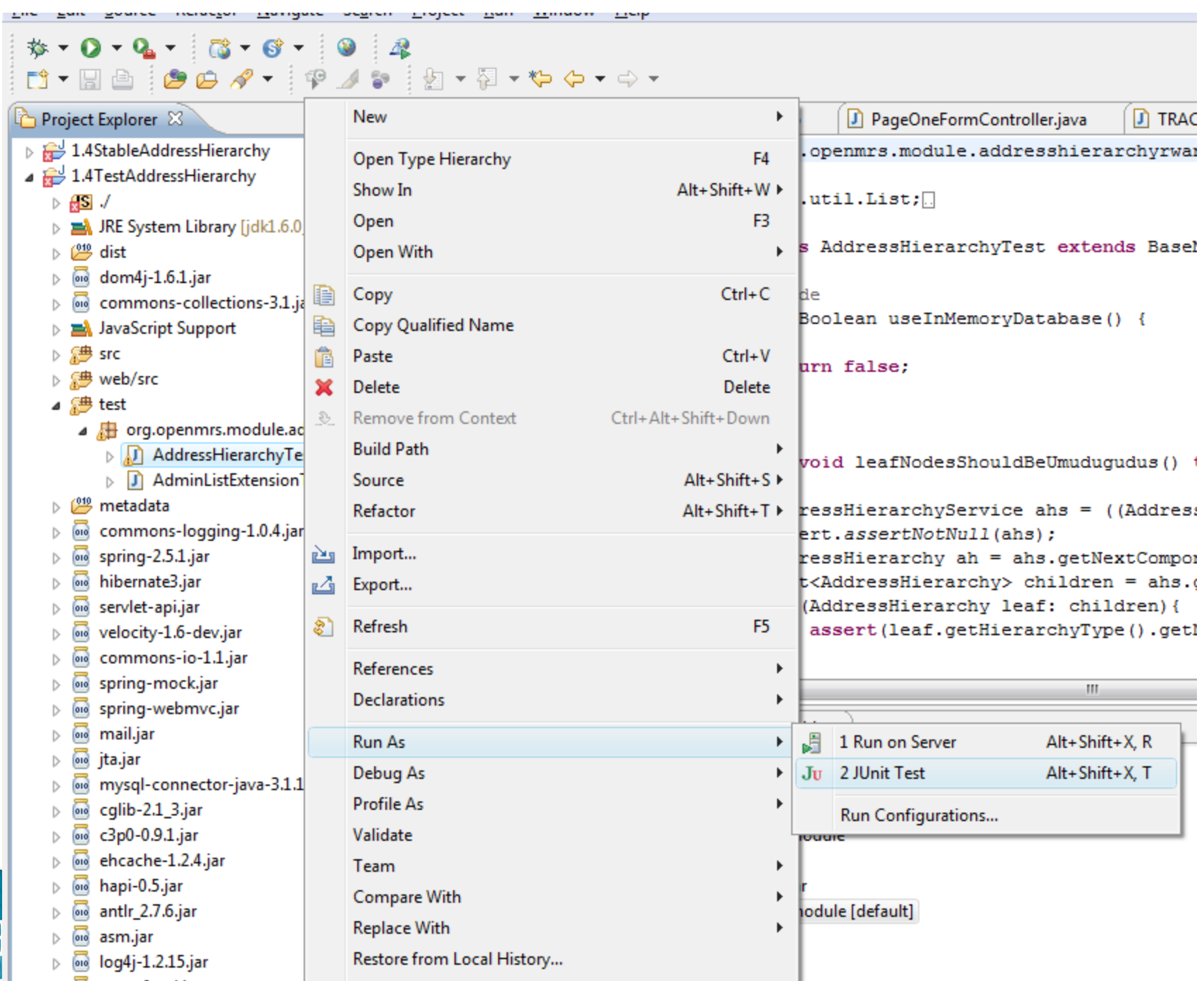
```
            Assert.assertTrue(country.getHierarchyType().getName().equals("Country"));
```

```
        }
```

```
    }
```



Forces use of database specified by the runtime properties file



A Successful JUnit Run

