PRO!VISION
SOFTWARE CRAFTSMANSHIP

wcm.io-based Unit Tests and Mocking

29.09.2015

Unit Testing with Mockito

Example: Controller uses a business class

```
/**
 * Controller for footer navigation.
 */
@Model(adaptables = SlingHttpServletRequest.class)
public class FooterNavPageLink {

  private NavigationPageItem root;

  @Inject
  public FooterNavPageLink(@Self NavigationManager navigationManager) {
    root = navigationManager.getFooterNavigation();
  }

  /**
   * @return Root navigation page item
   */
  public NavigationPageItem getRoot() {
    return root;
  }

}
```

- Injects a business class (which is also a Sling Model)
- Does not contain much logic in itself

Example: Controller uses a business class - Test

```
@RunWith(MockitoJUnitRunner.class)
public class FooterNavPageLinkTest {

  @Mock
  private NavigationManager navigationManager;
  private NavigationPageItem navigationPageItem = new NavigationPageItem("dummy");

  private FooterNavPageLink underTest;

  @Before
  public void setUp() throws Exception {
    when(navigationManager.getFooterNavigation()).thenReturn(navigationPageItem);
    underTest = new FooterNavPageLink(navigationManager);
  }

  @Test
  public void testGetRoot() throws Exception {
    assertSame(navigationPageItem, underTest.getRoot());
  }

}
```

- Mocks NavigationManager away
- Test only logic of model, not business logic

Further Examples – pure Mockito

- Editor Configuration Controller
  - https://github.com/wcm-io/wcm-io-config/blob/develop/editor/src/main/java/io/wcm/config/editor/controller/EditorConfiguration.java
  - https://github.com/wcm-io/wcm-io-config/blob/develop/editor/src/test/java/io/wcm/config/editor/controller/EditorConfigurationTest.java
- Sightly Configuration Bindings Value Provider
  - https://github.com/wcm-io/wcm-io-config/blob/develop/core/src/main/java/io/wcm/config/core/impl/ConfigBindingsValueProvider.java
  - https://github.com/wcm-io/wcm-io-config/blob/develop/core/src/test/java/io/wcm/config/core/impl/ConfigBindingsValueProviderTest.java
- Request Header Override Provider
  - https://github.com/wcm-io/wcm-io-config/blob/develop/core/src/main/java/io/wcm/config/core/override/impl/RequestHeaderOverrideProvider.java
  - https://github.com/wcm-io/wcm-io-config/blob/develop/core/src/test/java/io/wcm/config/core/override/impl/RequestHeaderOverrideProviderTest.java
- RequestPath Utility
  - https://github.com/wcm-io/wcm-io-sling/blob/develop/commons/src/main/java/io/wcm/sling/commons/request/RequestPath.java
  - https://github.com/wcm-io/wcm-io-sling/blob/develop/commons/src/test/java/io/wcm/sling/commons/request/RequestPathTest.java

Further Examples – problematic with pure Mockito

- Superimposing Manager Implementation – Negative Example:
  - https://github.com/apache/sling/blob/trunk/contrib/extensions/superimposing/src/main/java/org/apache/sling/superimposing/impl/SuperimposingManagerImpl.java
  - https://github.com/apache/sling/blob/trunk/contrib/extensions/superimposing/src/test/java/org/apache/sling/superimposing/impl/SuperimposingManagerImplTest.java
  - This is 75% Mocking code, 25% Test code
  - Risk of error in mocking code is higher than error in production code
- In general problematic with pure Mockito:
  - Iterating over resource and page hierarchies
  - Interaction with objects that have multiple different methods for similar things, e.g. parameter array in a request, accessing Resource or JCR Node properties
  - Code that needs a complex content structure as text fixture

Unit Testing with AEM Mocks

Introducing AEM Mocks

- AEM Mocks is a mock implementation of the most important parts of AEM API (esp. Page/PageManager, DAM and some more)
- Heavily depends on Sling Mocks, OSGi Mocks, JCR Mocks (now part of the Sling project)
- A JUnit Rule object “AemContext” gives easy access to all AEM/Sling context objects and allows easy setup of test fixtures, e.g.
  - Registering OSGi services and Sling Models
  - Loading content from JSON Files
  - Creating test content with simplified builder API
  - Manipulating current request state/investigating response state
- Targets everything *below* the presentation layer. JSP or Sightly scripts are not tested, but the controller and business logic used by them.

AEM Context JUnit Rule

```
public class ExampleTest {

  @Rule
  public final AemContext context = new AemContext();

 @Test
  public void testSomething() {
    Resource resource = context.resourceResolver().getResource("/content/sample/en");
    Page page = resource.adaptTo(Page.class);
    // further testing
  }

}
```

```
context.bundleContext()
context.componentContext()
context.currentPage()
context.currentResource()
context.pageManager()
context.request()
context.requestPathInfo()
context.response()
context.slingScriptHelper()
context.runMode("author");
...
```

Resource Resolver Types

- **RESOURCERESOLVER_MOCK (default)**
  - Mocked Resource Resolver from Sling Testing
  - Fastest, but does not support JCR
  - Eventing support, but no Search
- **JCR_MOCK**
  - Mocked JCR with real Sling Resource-JCR Mapping
  - Still very fast
  - Not all JCR features support (e.g. no Search, Observation)
- **JCR_OAK**
  - Real OAK with real Sling Resource-JCR (“CRX3”)
  - Full support for Node Types, Search, Observation etc.
- **JCR_JACKRABBIT**
  - Real Jackrabbit with real Sling Resource-JCR (“CRX2”)
  - Problem: Is not cleaned up for each test run, Tests have to use unique paths and clean up themselves

Getting and Manipulating Pages

```
@Test
public void testSomething() {
  Page page = context.pageManager().getPage("/content/sample/en");
  Template template = page.getTemplate();
  Iterator<Page> childPages = page.listChildren();
  // further testing
}

@Test
public void testPageManagerOperations() throws WCMException {
 Page page = context.pageManager().create("/content/sample/en", "test1",
      "/apps/sample/templates/homepage", "title1");
  // further testing
  context.pageManager().delete(page, false);
}
```

- Nearly all methods or Page and PageManager API are supported
- Behavior as close to real implementation as possible

Simulate Sling Request

```
// prepare sling request
context.request().setQueryString("param1=aaa&param2=bbb");

context.requestPathInfo().setSelectorString("selector1.selector2");
context.requestPathInfo().setExtension("html");

// set current page
context.currentPage("/content/sample/en");

// set WCM Mode
WCMMode.EDIT.toRequest(context.request());
```

- Request Mock: Parameters, RequestPathInfo, Attributes, Session, Headers, Cookies, Methods, Server Info
- Response Mock: Content Type, Headers, Cookies, Response Body
- Can be used standalone without AemContext as well

Register OSGi Services

```
// register OSGi service
context.registerService(MyClass.class, myService);

// or alternatively: inject dependencies, activate and register OSGi service
context.registerInjectActivateService(myService);

// get OSGi service
MyClass service = context.getService(MyClass.class);

// or alternatively: get OSGi service via bundle context
ServiceReference ref = context.bundleContext().getServiceReference(MyClass.class.getName());
MyClass service2 = context.bundleContext().getService(ref);
```

- You have to take care of the dependency hierarchy yourself (order of registration)
- Automatic injection of dependency and activation methods only works if SCR Metadata is in Classpath → IDE needs Maven Support for SCR Plugin

Sling Models

```
@Before
public void setUp() {
  // register models from package
  context.addModelsForPackage("com.app1.models");
}

@Test
public void testSomething() {
  RequestAttributeModel model = context.request().adaptTo(RequestAttributeModel.class);
  // further testing
}

@Model(adaptables = SlingHttpServletRequest.class)
interface RequestAttributeModel {
  @Inject
  String getProp1();
}
```

- All types of models and injectors supported
- Registration of package from which models should be loaded has to be done manually (package includes all subpackages)

Loading Content

```
@Before
public void setUp() throws Exception {
  // load JSON file into repository
  context.load().json("/sample-data.json", "/content/sample/en");

  // load binary file into repository
  context.load().binaryFile("/sample-file.gif", "/content/binary/sample-file.gif");
}

@Test
public void testSomething() {
  Resource resource = context.resourceResolver().getResource("/content/sample/en");
  Page page = resource.adaptTo(Page.class);
  // further testing
}
```

- JSON content can be created by downloading from AEM instances using ".tidy.999.json" suffix on a page hierarchy. But: this creates very huge and verbose JSON files that are difficult to read and maintain.
  - Example: https://github.com/wcm-io/wcm-io-samples/blob/develop/bundles/sample-app/src/test/resources/sample-content.json
- Better solution: handcraft small JSON snippets per group of related test cases.
  - Still JSON download can be a good start for this, remove everything not needed

Building Content

```
// create page
context.create().page("/content/sample/en", "/apps/sample/template/homepage");

// create resource
context.create().resource("/content/test1", ImmutableValueMap.builder()
    .put("prop1", "value1")
    .put("prop2", "value2")
    .build());
```

- Easy-to-use and Exception-free fluent API to create pages and resources
- Uses Guava-style ImmutableValueMap for providing property maps
- Missing nodes in hierarchy are created automatically

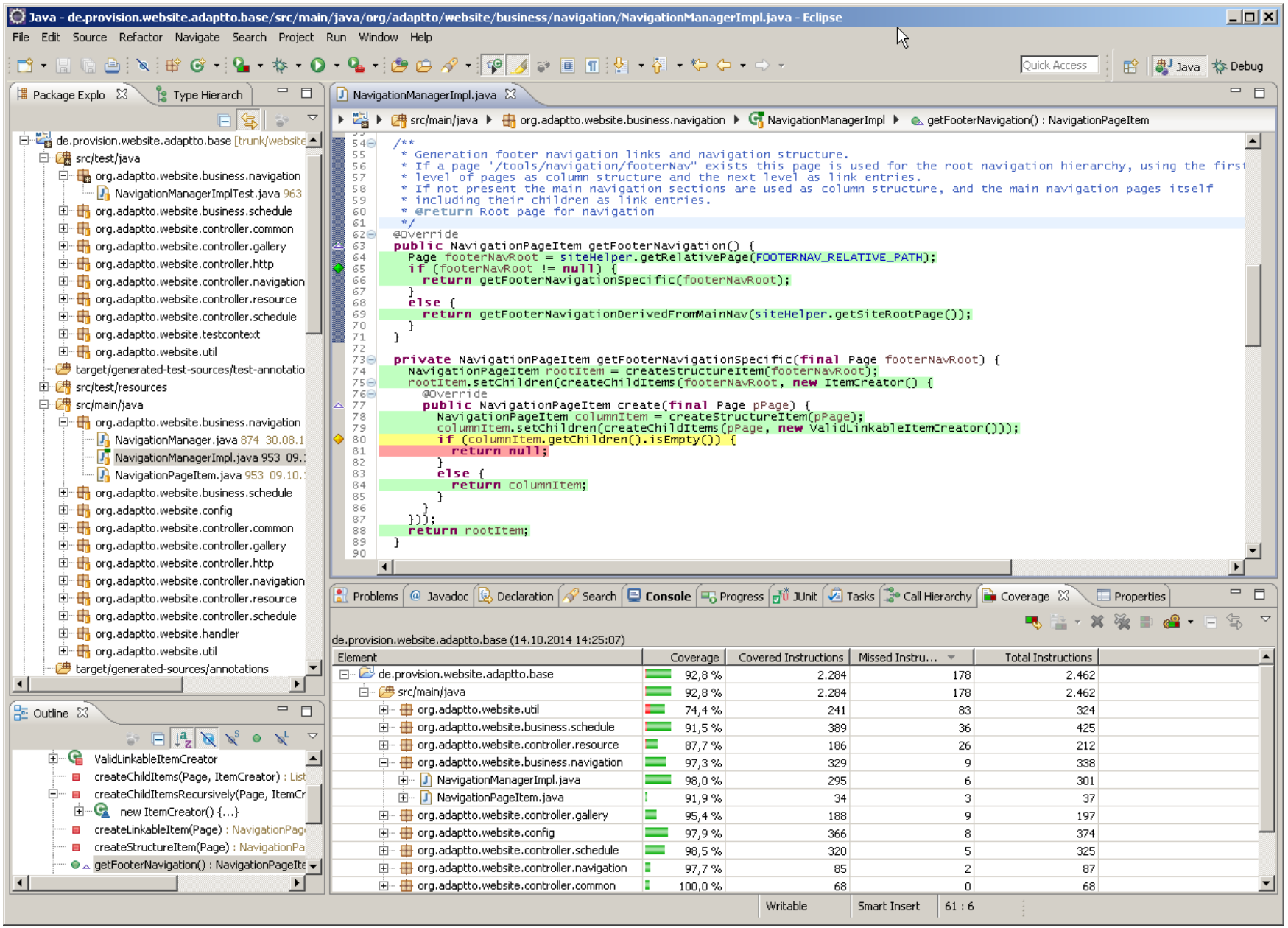
AEM Mock Examples

- Navigation Manager
  - https://github.com/wcm-io/wcm-io-samples/blob/develop/bundles/sample-app/src/main/java/io/wcm/samples/app/business/navigation/impl/NavigationManagerImpl.java
  - https://github.com/wcm-io/wcm-io-samples/blob/develop/bundles/sample-app/src/test/java/io/wcm/samples/app/business/navigation/impl/NavigationManagerImplTest.java
- Resource Link Controller
  - https://github.com/wcm-io/wcm-io-samples/blob/develop/bundles/sample-app/src/main/java/io/wcm/samples/app/controller/resource/ResourceLink.java
  - https://github.com/wcm-io/wcm-io-samples/blob/develop/bundles/sample-app/src/test/java/io/wcm/samples/app/controller/resource/ResourceLinkTest.java
- Combination of Mockito and AEM Mocks
  - https://github.com/wcm-io/wcm-io-config/blob/develop/editor/src/main/java/io/wcm/config/editor/impl/EditorParameterPersistence.java
  - https://github.com/wcm-io/wcm-io-config/blob/develop/editor/src/test/java/io/wcm/config/editor/impl/EditorParameterPersistenceTest.java

Summary

Summary

- There is no “best way” for all cases
  - Use Mockito and/or AEM Mocks whatever suites best for your unit test
  - Goal: Tests that are easy to write *and* easy to maintain
  - Tests that focus on the logic of the testable, and only this
- If you find a missing/incompatible feature in the Mocks:
  please report them, most can be fixed easily and promptly
- Goal: **80% unit test coverage** for all projects
- Constantly test your test coverage in your IDE!
- Use Jenkins or SONAR to have a broader look on your project to find the weak spots in unit test coverage


Java - de.provision.website.adaptto.base/src/main/java/org/adaptto/website/business/navigation/NavigationManagerImpl.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explo
Type Hierarch
de.provision.website.adaptto.base [trunk/website
src/test/java
org.adaptto.website.business.navigation
NavigationManagerImplTest.java 963
org.adaptto.website.business.schedule
org.adaptto.website.controller.common
org.adaptto.website.controller.gallery
org.adaptto.website.controller.http
org.adaptto.website.controller.navigation
org.adaptto.website.controller.resource
org.adaptto.website.controller.schedule
org.adaptto.website.testcontext
org.adaptto.website.util
target/generated-test-sources/test-annotatio
src/test/resources
src/main/java
org.adaptto.website.business.navigation
NavigationManager.java 874 30.08.1
NavigationManagerImpl.java 953 09.
NavigationPageItem.java 953 09.10.
org.adaptto.website.business.schedule
org.adaptto.website.config
org.adaptto.website.controller.common
org.adaptto.website.controller.gallery
org.adaptto.website.controller.http
org.adaptto.website.controller.navigation
org.adaptto.website.controller.resource
org.adaptto.website.controller.schedule
org.adaptto.website.handler
org.adaptto.website.util
target/generated-sources/annotations
Outline
ValidLinkableItemCreator
createChildItems(Page, ItemCreator) : List
createChildItemsRecursively(Page, ItemCr
new ItemCreator() {...}
createLinkableItem(Page) : NavigationPag
createStructureItem(Page) : NavigationPa
getFooterNavigation() : NavigationPageIte
NavigationManagerImpl.java
src/main/java
org.adaptto.website.business.navigation
NavigationManagerImpl
getFooterNavigation() : NavigationPageItem
/**
* Generation footer navigation links and navigation structure.
* If a page '/tools/navigation/footerNav" exists this page is used for the root navigation hierarchy, using the firs
* level of pages as column structure and the next level as link entries.
* If not present the main navigation sections are used as column structure, and the main navigation pages itself
* including their children as link entries.
* @return Root page for navigation
*/
@Override
public NavigationPageItem getFooterNavigation() {
Page footerNavRoot = siteHelper.getRelativePage(FOOTERNAV_RELATIVE_PATH);
if (footerNavRoot != null) {
return getFooterNavigationSpecific(footerNavRoot);
}
else {
return getFooterNavigationDerivedFromMainNav(siteHelper.getSiteRootPage());
}
}
private NavigationPageItem getFooterNavigationSpecific(final Page footerNavRoot) {
NavigationPageItem rootItem = createStructureItem(footerNavRoot);
rootItem.setChildren(createChildItems(footerNavRoot, new ItemCreator() {
@Override
public NavigationPageItem create(final Page pPage) {
NavigationPageItem columnItem = createStructureItem(pPage);
columnItem.setChildren(createChildItems(pPage, new ValidLinkableItemCreator()));
if (columnItem.getChildren().isEmpty()) {
return null;
}
else {
return columnItem;
}
}
}));
return rootItem;
}
Problems
Javadoc
Declaration
Search
Console
Progress
JUnit
Tasks
Call Hierarchy
Coverage
Properties
de.provision.website.adaptto.base (14.10.2014 14:25:07)
Element
Coverage
Covered Instructions
Missed Instru...
Total Instructions
de.provision.website.adaptto.base 92,8 % 2.284 178 2.462
src/main/java 92,8 % 2.284 178 2.462
org.adaptto.website.util 74,4 % 241 83 324
org.adaptto.website.business.schedule 91,5 % 389 36 425
org.adaptto.website.controller.resource 87,7 % 186 26 212
org.adaptto.website.business.navigation 97,3 % 329 9 338
NavigationManagerImpl.java 98,0 % 295 6 301
NavigationPageItem.java 91,9 % 34 3 37
org.adaptto.website.controller.gallery 95,4 % 188 9 197
org.adaptto.website.config 97,9 % 366 8 374
org.adaptto.website.controller.schedule 98,5 % 320 5 325
org.adaptto.website.controller.navigation 97,7 % 85 2 87
org.adaptto.website.controller.common 100,0 % 68 0 68
Writable
Smart Insert
61 : 6

Resources

- AEM Mock documentation
  http://wcm.io/testing/aem-mock/

- Sling/OSGi/JCR Mock documentation
  http://sling.apache.org/documentation/development/sling-mock.html
  http://sling.apache.org/documentation/development/osgi-mock.html
  http://sling.apache.org/documentation/development/jcr-mock.html

- Mockito documentation
  http://mockito.github.io/mockito/docs/current/org/mockito/Mockito.html

- Example projects
  https://github.com/wcm-io/wcm-io-samples
  https://github.com/wcm-io/

Contact

Karim Khan
Tel. +49 (69) 8700328-0
E-Mail kkhan@pro-vision.de

Berlin
pro!vision GmbH
Wilmersdorfer Str. 50-51
10627 Berlin
Germany

Tel. +49 (30) 818828-50
E-Mail info@pro-vision.de

Frankfurt
pro!vision GmbH
Löwengasse 27 A
60385 Frankfurt am Main
Germany

Tel. +49 (69) 8700328-0
E-Mail info@pro-vision.de

Wolfsburg
pro!vision GmbH
Schlosserstr. 6a
38440 Wolfsburg
Germany

Tel. +49 (5361) 834937-0
E-Mail info@pro-vision.de