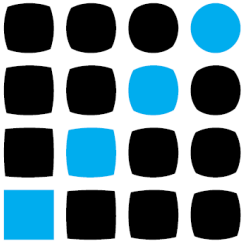


adaptTo()

APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 26-28 SEPTEMBER 2016

Unit Testing with Sling & AEM Mocks

Stefan Seifert, pro!vision GmbH

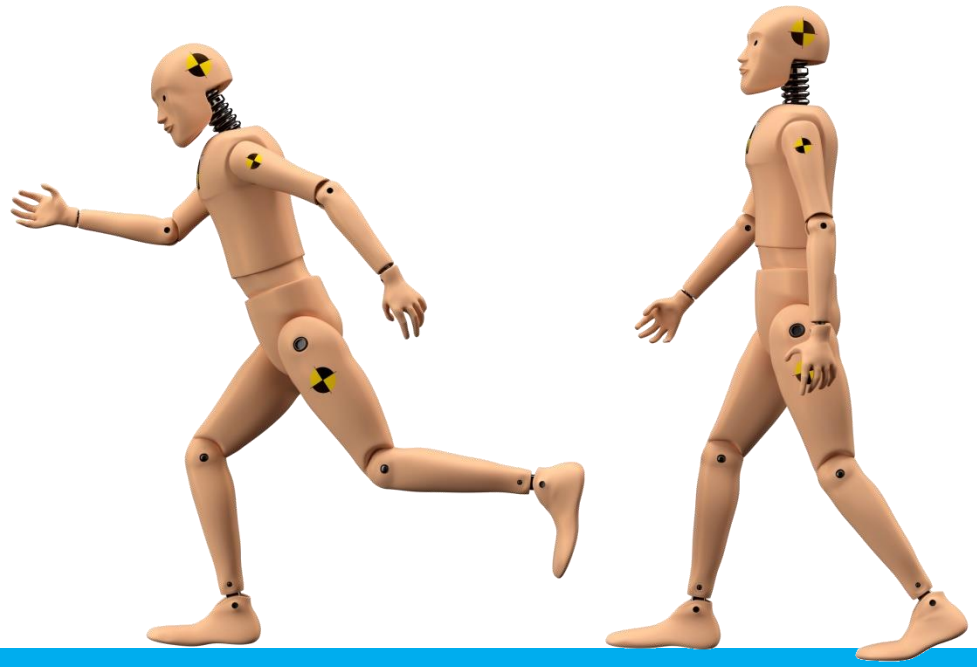
About the Speaker

- AEM Developer
- Apache Sling PMC
- CTO of pro!vision GmbH

PRO!VISION
SOFTWARE CRAFTSMANSHIP

https://www.pro-vision.de

adaptTo()



Recap: Sling & AEM Mocks

What is Sling Mocks, AEM Mocks & Co

- Mock implementation of most important parts of Sling, AEM, OSGi, JCR
- JUnit Rule to easy access the mock context
- Makes it easy to test Java Code facilitating the AEM and Sling APIs
- Very fast test execution

SlingContext / AemContext Junit Rule

```
public class ExampleTest {

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

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

}
```

```
public class ExampleAemTest {

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

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

Choose Resource Resolver implementation

Resource Resolver Type	Sling API	JCR API	Node Types	Obser-vation	JCR Query	Lucene Fulltext
RESOURCE RESOLVER_MOCK	✓	✗	✗	✗ (Sling only)	✗	✗
JCR_MOCK	✓	✓	✗	✗	✗ (mocked)	✗
JCR_OAK	✓	✓	✓	✓	✓	✗

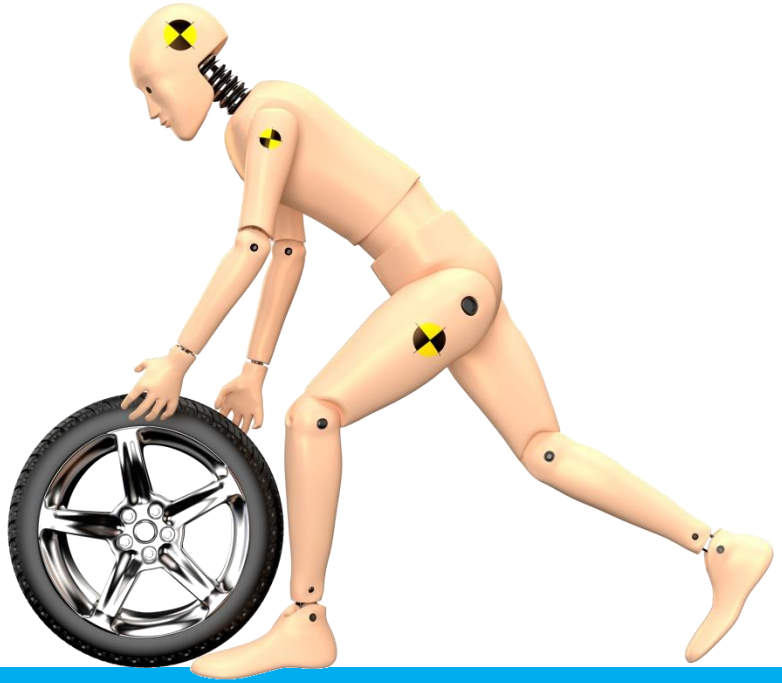
Sling Mocks Features

- Mock Sling Resource API
- Easy resource creation
- Import resources from JSON files
- Simulate requests, capture response output
- Use Sling Models

AEM Mocks Features

- Support most-important parts of the AEM WCM and DAM APIs
- Page and Asset creation and manipulation
- Supports AEM 6.0, 6.1 and 6.2

adaptTo()



New Features

OSGi Companion services

- EventAdmin
  - Send and receive OSGi events
- ConfigAdmin
  - Provide OSGi configurations at runtime

OSGi R6 support

- OSGi R6 API
- Field-based reference bindings
  - Support collections of services and references
- Component property types for config
  - Annotation class-based configuration

OSGi R6 support

```
@Component(service = MyService.class, immediate = true)
@Designate(ocd = MyServiceImpl.Config.class)
public class MyServiceImpl implements MyService {

  @ObjectClassDefinition(name = "My Example Service")
  @interface Config {
    @AttributeDefinition(description = "URL of webservice.")
    String entryPointUrl();

    @AttributeDefinition(description = "Timeout in seconds.")
    int timeout() default 5;
  }

  @Activate
  private void activate(Config config) {
    // Process configuration
  }

}
```

Supported by OSGi Mocks

Automatic JCR Node Type registration

- When you use Oak, it does not “know” any Sling- or AEM-specific JCR node types
- Detects and registers Node types Definitions (CND) referenced in MANIFEST files
- AEM Mocks provide most-important AEM specific node types (cq:Page, cq:Asset etc.)

Simple mocking of Adapter Factories

- Use one-liner to mock adaptTo() calls

```
// Return fixed value for adaption
context.registerAdapter(Resource.class, Integer.class, 5);

// Return dynamic value for adaption
context.registerAdapter(Resource.class, MyClass.class,
  new Function<Resource,MyClass>() {
    @Override
    public MyClass apply(Resource input) {
        // transform Resource to MyClass
    }
  });
```

New AEM Mocks Features

- Mocks for ComponentManager, Component, TagManager, Tag, Designer
- Easy creation of DAM assets, renditions and tags

Asset and rendition creation

```
// create asset from byte array
asset = context.create().asset("/content/dam/sample1.bin",
    new ByteArrayInputStream(new byte[] { 0x01, 0x02, 0x03  }),
    "application/octet-stream");

// load asset from class path (/sample-image.gif)
asset = context.create().asset("/content/dam/sample1.gif",
    "/sample-image.gif", "image/gif");

// create dummy asset with given with/height and mime type
asset = context.create().asset("/content/dam/sample1.gif",
    100, 50, "image/gif");

// add dummy rendition
rendition = context.create().assetRendition(asset, "sample2.gif",
    20, 20, "image/gif");
```

Simplified properties/map handling

- Simplified properties syntax can be used anywhere within Sling and OSGi mock

```
// before
resource = context.create().resource("/content/test1/resource2",
    ImmutableMap.<String,Object>builder()
    .put("jcr:title", "Test Title")
    .put("intProp", 5)
    .build());

// after
resource = context.create().resource("/content/test1/resource2",
        "jcr:title", "Test Title",
        "intProp", 5);
```

ResourceBuilder integration

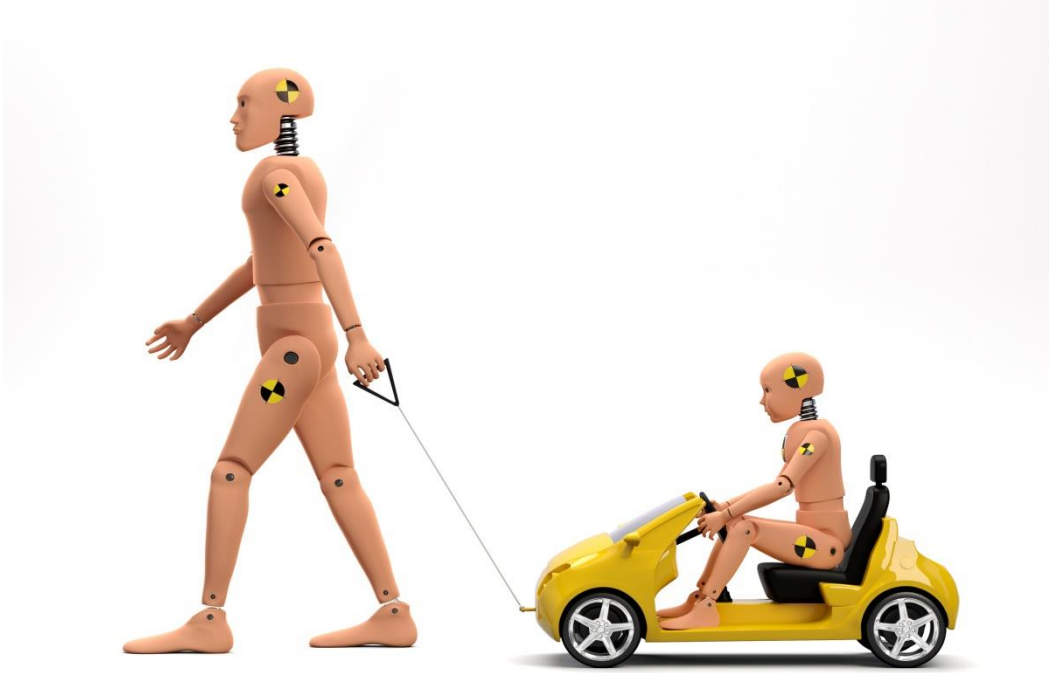
- Fluid syntax to build a resource hierarchy

```
context.build().resource("/content/site1", "prop1", "value1")
               .resource("en")
               .siblingsMode()
               .resource("page1", "jcr:title", "My title")
               .resource("page2");
```

Result

```
/content
    /site1
      @prop1 = "value1"
        /en
            /page1
              @jcr:title = "My title"
            /page2
```

adaptTo()



Version Matrix

API Support Matrix

API	Sling / OSGi / AEM Mock **Version 1.x**	Sling / OSGi / AEM Mock **Version 2.x**
JCR API	2.0	2.0
Servlet API	3.0	**3.1**
OSGi	R4, R5	**R6**
Sling API	2.4.0	**2.11.0**

AEM Version Usage Matrix

AEM Version	Sling Mock Version	OSGi Mock Version	AEM Mock Version
AEM 6.0	1.x	1.x	1.x
AEM 6.1	1.x	1.x **or 2.x**	1.x
AEM 6.2	**2.x**	**2.x**	**2.x**

Pitfalls

#1 Correct versions

- Pick the correct major versions **1.x or 2.x** for your environment e.g. AEM version
- If you encounter problems update to the latest minor version

#2 Unobfuscated APIs

- Never use the AEM “obufuscated-apis.jar” for unit tests – it may fail utterly
  - AEM APIs for 6.0 and 6.2 are available unobfuscated on repo.adobe.com
  - Get 6.1 APIs from Adobe Support

#3 Transitive dependency mismatch

- Transitive Dependencies from sling-mock may be overwritten with incompatible Versions in your pom
  - e.g. transitive from other dependencies
  - Check with
    `mvn dependency:tree`
    and compare with versions in the mock poms

#3 Transitive dependency mismatch

- Important Sling Mock dependencies

Sling Bundle	Sling Mock 1.x	Sling Mock 2.x
org.apache.sling.resourceresolver	1.1.0	1.4.8
org.apache.sling.jcr.resource	2.3.6	2.7.4
org.apache.sling.models.impl	1.1.0	1.2.2
org.apache.sling.commons.osgi	2.2.0	2.4.0

#3 Transitive dependency mismatch

- Possible solutions/workarounds
  - Exclude mismatches from other deps
  - Change dependency order (move mocks to top)
  - Inherit from parent POM which declares all correct versions for each AEM major version, e.g. http://wcm.io/tooling/maven/aem-dependencies.html

#4 Eclipse Gotcha: No SCR metadata

- A unit test fails with this error after refreshing your project in Eclipse:

```
org.apache.sling.testing.mock.osgi.NoScrMetadataException:
        No OSGi SCR metadata found in classpath at OSGI-INF/x.y.z.MyService.xml
        at org.apache.sling.testing.mock.osgi.OsgiServiceUtil.injectServices(OsgiServiceUtil.
        at org.apache.sling.testing.mock.osgi.MockOsgi.injectServices(MockOsgi.java:127)
        at org.apache.sling.testing.mock.osgi.context.OsgiContextImpl.registerInjectActivateS
        at org.apache.sling.testing.mock.osgi.context.OsgiContextImpl.registerInjectActivateS
        at x.y.z.MyServiceTest.testResult(MyServiceTest.java:76)
…
```

- Solution: Project → Clean

#5 Declarative Services OSGi annotations

- You use the new OSGi DS annotations from `org.osgi.service.component.annotations` and `maven-bundle-plugin` to generate them
  - Make sure to export them to the classpath
  - And define an extra „manifest“ goal
  - See Maven Bundle Plugin FAQ for details
    - **“Use SCR metadata generated by BND in Unit Tests”**

References

Mock Documentation

- Sling Mocks: http://sling.apache.org/documentation/development/sling-mock.html
- OSGi Mocks: http://sling.apache.org/documentation/development/osgi-mock.html
- JCR Mocks: http://sling.apache.org/documentation/development/jcr-mock.html
- ResourceResolver Mocks: http://sling.apache.org/documentation/development/resourceresolver-mock.html
- AEM Mocks: http://wcm.io/testing/aem-mock/

Other presentations about Mocks

- adaptTo() 2014 lightning talk: https://adapt.to/2014/en/schedule/lightning-talks.html
- adaptTo() 2015 talk: https://adapt.to/2015/en/schedule/how-do-i-test-my-sling-application.html
- adaptTo() 2015 playground session: https://adapt.to/2015/en/schedule/playground-sesssion.html

References

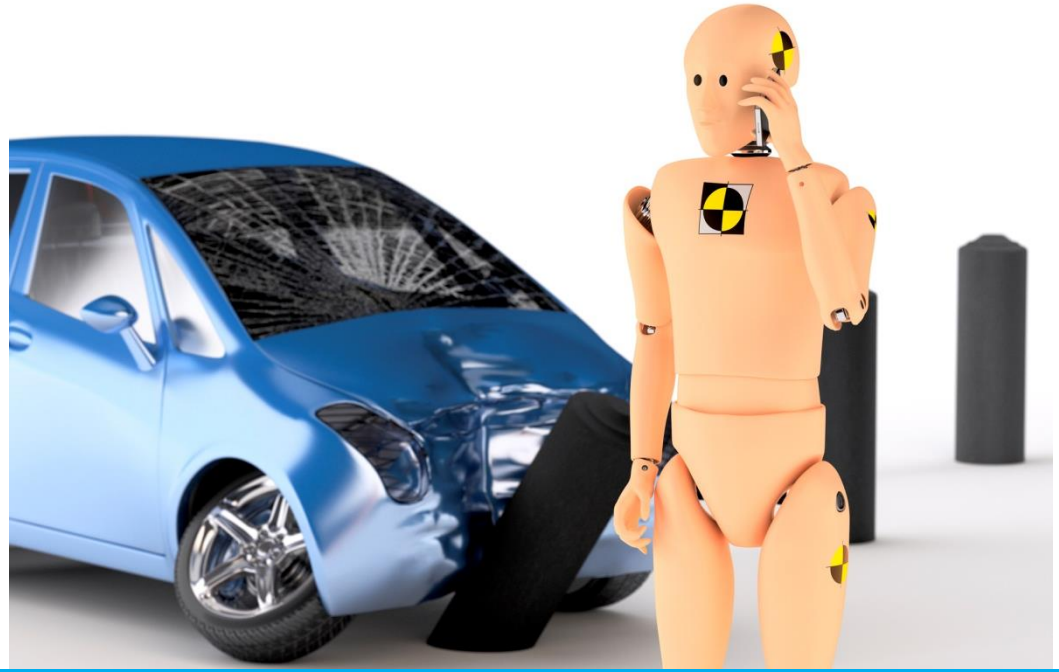
See mock-based unit tests in real projects

- Unit tests of Sling Mock and AEM Mock projects
- Sling Context-Aware Configuration
  https://github.com/apache/sling/tree/trunk/contrib/extensions/contextaware-config
- wcm.io unit tests e.g.
  https://github.com/wcm-io/wcm-io-handler/tree/develop/url
  https://github.com/wcm-io/wcm-io-handler/tree/develop/link
  https://github.com/wcm-io/wcm-io-handler/tree/develop/media
  https://github.com/wcm-io/wcm-io-wcm/tree/develop/parsys
  https://github.com/wcm-io/wcm-io-dam/tree/develop/asset-service
- and a lot of other projects

Felix Maven Bundle Plugin FAQ - Use SCR metadata generated by BND in Unit Tests

- http://felix.apache.org/documentation/faqs/apache-felix-bundle-plugin-faq.html#use-scr-metadata-generated-by-bnd-in-unit-tests

adaptTo()



Questions?