



**adaptTo()**

EUROPE'S LEADING AEM DEVELOPER CONFERENCE

27<sup>th</sup> – 29<sup>th</sup> SEPTEMBER 2021

# What's new in AEM Mocks

Stefan Seifert, diva-e

# About the Speaker

- AEM Developer
- Apache Sling PMC
- Apache Member

**diva<sup>e</sup>**

<https://diva-e.com/>



Stefan Seifert



**PRO!VISION**  
SOFTWARE CRAFTSMANSHIP



## What's AEM Mocks?



# About AEM Mocks

- Unit Tests for your AEM Application
- Provides an **in-memory AEM environment** suitable for Unit Tests
- Covers 90% of what is required for typical AEM applications
- Can be combined with Mockito and others
- Fast test execution

# Tech Stack

AEM Mocks

wcm.io



Sling Mocks

Apache Sling



OSGi Mocks

Apache Sling

Resource  
Resolver Mocks

Apache Sling

JCR Mock

Apache Sling

JUnit 5 or JUnit 4

# Choose Resource Resolver implementation

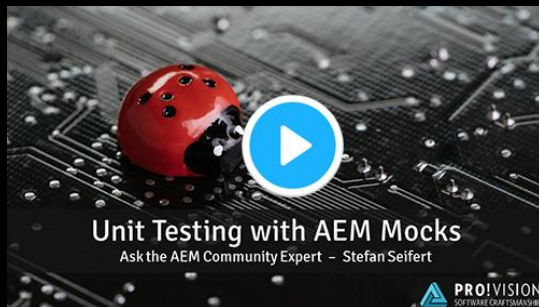
**FASTER**

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

**more features**

\* Requires additional dependency: `org.apache.sling.testing.sling-mock-oak`

## Ask the AEM Community Expert Session March 2019



<https://github.com/stefanseifert/2019-atace-unit-testing-with-aem-mocks>



# Your Project – Test Dependency Management



# AEM Mocks Test Dependency

```
<dependency>  
  <groupId>io.wcm</groupId>  
  <artifactId>io.wcm.testing.aem-mock.junit5</artifactId>  
  <version>4.1.4</version>  
  <scope>test</scope>  
</dependency>
```

- Always use latest version
- Do not define additional dependencies to Sling Mocks
- Do not exclude transitive dependencies
  - Transitive dependencies reference required Sling implementation bundles that are not included in AEM API JAR
- POM: Define AEM Mocks dependencies **before** AEM API dependency

# AEM Dependencies

- AEM Mocks compatible with AEM 6.4, 6.5, AEMaaCS
  - Default transitive dependencies for AEM 6.4 (from 2018)
- Ideally, you should test with transitive dependencies exactly from AEM Version you are deploying to
- Use AEM (Cloud) Dependencies from wcm.io
  - <https://wcm.io/tooling/maven/aem-dependencies.html>
    - For AEMaaCS, AEM 6.x, all service packs

# AEM Cloud Dependencies Example

Replace

```
<dependencyManagement>
  <dependency>
    <groupId>com.adobe.aem</groupId>
    <artifactId>aem-sdk-api</artifactId>
    <version>2021.9.5852.20210917T073206Z-210800</version>
  </dependency>
</dependencyManagement>
```

[See AEM Cloud Dependencies POM](#)



with

```
<dependencyManagement>
  <dependency>
    <groupId>io.wcm.maven</groupId>
    <artifactId>io.wcm.maven.aem-cloud-dependencies</artifactId>
    <version>2021.9.5852.20210917T073206Z-210800.0000</version>
    <type>pom</type>
    <scope>import</scope>
  </dependency>
</dependencyManagement>
```

AEM 6.5:

io.wcm.maven:io.wcm.maven.aem-dependencies:6.5.10.0000



# AEM Mocks Plugins

- Provide “Test Lifecycle Extensions” for non-Standard AEM extensions
    - Registers required OSGi services
    - Provides helper methods
    - List of all plugins
- <https://wcm-io.atlassian.net/wiki/x/EQBjPg>

# Plugin Definition Example

```
import static org.apache.sling.testing.mock.caconfig.ContextPlugins.CACONFIG;
import static com.adobe.cq.wcm.core.components.testing.mock.ContextPlugins.CORE_COMPONENTS;

@ExtendWith(AemContextExtension.class)
class MyUnitTest {

    private final AemContext context = new AemContextBuilder()
        // register plugins
        .plugin(CACONFIG)
        .plugin(CORE_COMPONENTS)
        // shared context setup code for all tests
        .<AemContext>afterSetUp(context -> {
            // add more setup code here
        })
        .build();
}
```

Registers OSGi services  
required by Core Components  
internally

# Write Context-Aware Configuration

```
@Rule
public SlingContext context = new AemContextBuilder()
    .plugin(CACONFIG)
    .build();

@Before
public void setUp() {
    // prepare site with references to conf path
    context.create().resource("/content/region/site", "sling:configRef", "/conf/region/site");
    context.currentResource(context.create().resource("/content/region/site/en"));

    // write a single configuration
    MockContextAwareConfig.writeConfiguration(context, "/content/region/site", SimpleConfig.class,
        "stringParam", "value1");

    // write configuration list
    MockContextAwareConfig.writeConfigurationCollection(context, "/content/region/site", ListConfig.class,
        ImmutableList.of(
            ImmutableMap.<String, Object>of("stringParam", "value1"),
            ImmutableMap.<String, Object>of("stringParam", "value2")));
}
```



# Enhanced OSGi Support



# Constructor Injection

## *OSGi Service*

```
@Component(service = MyService.class)
public class MyServiceImpl implements MyService {
    private final PageManagerFactory pageManagerFactory;

    // inject OSGI service via constructor
    public MyServiceImpl(
        @Reference PageManagerFactory pageManagerFactory) {
        this.pageManagerFactory = pageManagerFactory;
    }
}
```

## *Unit Test*

```
@BeforeEach
void setUp() {
    // register service in Unit Test
    context.registerInjectActivateService(MyServiceImpl.class);
}
```

Register by Class Name:  
Works for Constructor  
and Field Injection

In your `<dependencies>` section, instead of

- `org.osgi:osgi.core`
- `org.osgi:osgi.cmpn`

Use any of

- `org.osgi:org.osgi.framework`
- `org.osgi:org.osgi.service.component`
- `org.osgi:org.osgi.service.metatype`
- `org.osgi:org.osgi.service.cm`
- `org.osgi:org.osgi.service.event`
- ...

[Most used OSGi artifacts  
in AEM Cloud Dependencies POM](#)

AEM Mocks uses individual artifacts.



# Prepare Test Context Data

Points to test class path,  
or relative path in file system

## ■ Single Files (read/write)

```
// load node structure from single JSON file to path in repository  
context.load().json("/json-test-data/content.json", "/content/sample/en");
```

NEW

```
// load node structure from single FileVault XML file to path in repository  
context.load().fileVaultXml("/xml-test-data/en/.content.xml", "/content/sample/en");
```

## ■ Mount Folders (read-only)

Uses FSResource

NEW

```
// mount folder with JSON files and binaries to path in repository  
context.load().folderJson("src/test/resources/json-test-data", "/content/sample/en");
```

NEW

```
// mount folder with FileVault XML files and binaries to path in repository  
context.load().folderFileVaultXml("src/test/resources/xml-test-data", "/content");
```

# Create Test Data on-the-fly

```
// create a page with page properties
Page page = context.create().page("/content/site1/en", TEMPLATE,
    JCR_TITLE, "Test Title",
    "customProperty", "value1");

// create a dummy image with web-enabled rendition
Asset asset = context.create().asset("/content/dam/sample1.jpg", 1600, 900, "image/jpeg");
Rendition webEnabledRendition = context.create().assetRenditionWebEnabled(asset);

// create a child resource "component1" in the page
Resource resource = context.create().resource(page, "component1",
    PROPERTY_RESOURCE_TYPE, "myapp/components/text",
    "text", "My Text",
    "fileReference", asset.getPath());
```

JPEG, PNG, GIF supported by default, TIFF and SVG with [Java ImageIO plugins](#).



# Test Core Component Extensions

## Core Components Models can be extended using the Delegation Pattern

### *Component Definition*

```
slings:resourceSuperType="core/wcm/components/title/v2/title"
```

Needs to be present in  
/apps folder  
in mock repository

### *Model Class*

```
@Model(adaptables = SlingHttpServletRequest.class,  
        adapters = Title.class,  
        resourceType = "myproject/components/customTitle")  
public class CustomTitle implements Title {  
    @Self @Via(type = ResourceSuperType.class)  
    private Title delegate;  
}
```

TitleImpl Core Component  
class needs to be present  
in class path



# Test Core Component Extensions

```
@BeforeEach
void setUp() {
    context.create().resource("/apps/myproject/components/customTitle",
        "sling:resourceSuperType", "core/wcm/components/title/v2/title");

    /* alternative:
    context.load().fileVaultXml("../ui.apps/src/main/content/jcr_root/apps/myproject/components/customTitle/.content.xml",
        "/apps/myproject/components/customTitle"); */

    page = context.create().page(SITE_ROOT_PATH);
}
```

```
@Test
void testEmpty() {
    context.currentResource(context.create().resource(page, "title",
        PROPERTY_RESOURCE_TYPE, "myproject/components/customTitle",
        JCR_TITLE, "My Title"));

    Title underTest = context.request().adaptTo(Title.class);

    assertEquals("My Title", underTest.getText());
}
```

Resource type hierarchy:

myproject/components/customTitle  
↳ core/wcm/components/title/v2/title

Prerequisites:

- Use AEM Mocks Plugin for Core Components
- **core.wcm.components.core** as test dependency



A close-up photograph of a blue ladybug with black spots on its back, resting on a dark green printed circuit board (PCB). The PCB is covered in intricate silver-colored traces and numerous small, circular solder points. The lighting is dramatic, highlighting the texture of the ladybug and the metallic sheen of the board.

# Enhanced AEM Feature Support

## Test component with different content policies.

```
// create a content policy with mapping for resource type
context.contentPolicyMapping("app1/componenty/component1",
    "prop1", "value1",
    "prop2", 123);
```

Resource type of component  
to be tested

## Creates structured and unstructured content fragments.

```
// Create structured content fragment
ContentFragment cf = context.create().contentFragmentStructured("/content/dam/cf1",
    "param1", "value1",
    "param2", 123,
    "param3", true,
    "param4", new String[] { "v1", "v2" });

// Create unstructured content fragment
ContentFragment cf = context.create().contentFragmentText("/content/dam/cf2",
    "<p>Text</p>", "text/html");
```

## Test code that expects i18n resources

### Model Class

```
@Model(adaptables = SlingHttpServletRequest.class)
public class MyModel {
    @AemObject
    private I18n i18n;

    @PostConstruct
    private void activate() { // read i18n
        String translation = i18n.get("key1");
        // ...
    }
}
```

Uses [wcm.io AEM Sling Models Extension](#)

### Unit Test

```
@BeforeEach
void setUp() { // set i18n key
    ResourceBundle bundle = context.request().getResourceBundle(Locale.ENGLISH);
    ((MockResourceBundle)bundle).put("key1", "My Translation");
}
```

Mock i18n Resource Bundles can be manipulated in Unit Tests

# Other new features

- Support for JUnit 5 `@BeforeAll` and `@AfterAll` annotations
- Resource Resolver Mock: Provided mocked results for `findResources/queryResources`
- Mock Asset Manager sends OSGi events
- New Mock implementations of:
  - Sling XSSAPI
  - Sling Feature Flag Service
  - Granite ResourceCollectionManager/ResourceCollection
  - AEM PageManagerFactory
  - AEM LanguageManager
  - AEM Dynamic Media PublishUtils
  - AEM SlingModelFilter
  - AEM Externalizer



## References

- wcm.io AEM Mocks

- <http://wcm.io/testing/aem-mock/>

- Talks

- [Ask the AEM Community Expert Session March 2019](#)

- [adaptTo\(\) 2018 Talk: JUnit 5 and Sling/AEM Mocks](#)

- [adaptTo\(\) 2016 Talk: Unit Testing with Sling & AEM Mocks](#)

- [adaptTo\(\) 2014 Lightning Talk: Mock AEM & Co for Unit Tests](#)

- Apache Sling Mocks

- <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>

- Core Components

- <https://github.com/adobe/aem-core-wcm-components/tree/master/testing/aem-mock-plugin>

- Need Help? Use the Mailing Lists  
<http://wcm.io/mailling-lists.html>
- File a bug or extension requests (PRs welcome!)  
<https://wcm-io.atlassian.net/browse/WTES>
- Trouble Shooting Articles
  - [List of AEM Mock Context Plugins](#)
  - [How to use AEM Mocks in Adobe AEM Project Archetype](#)
  - [Migrate AEM Mocks Unit Tests from JUnit 4 to JUnit 5](#)
  - [AEM Mocks fails with "No OSGi SCR metadata found for class ..."](#)
- Examples for Unit Tests using AEM Mocks:  
Look at the repositories in <https://github.com/wcm-io> - most of them contains lots of test code using AEM Mocks





Questions?