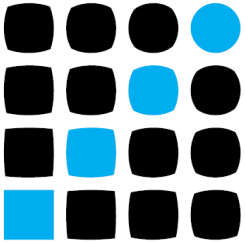

adaptTo()

APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 22-24 SEPTEMBER 2014

Mock AEM & Co for Unit Tests

Stefan Seifert, pro!vision GmbH

Terminology: Test Doubles

- **Dummy:** passed around – never used
- **Fake:** working implementations with shortcuts (e.g. in-memory database)
- **Stubs:** provide canned answers
- **Mocks:** objects pre-programmed with expectations

Source: http://martinfowler.com/articles/mocksArentStubs.html

Limits of Mocking

- Mockito et al are great Tools for Unit Tests
- Use them wherever possible
- Limited if code interacts a lot with resource resolver
- Need for a Fake Environment for Sling – and JCR, OSGi, AEM

Resource Resolver Mock

- If you just need to mock Sling Resource Access – use Apache Sling Testing Resource Resolver Mock
- If you need more…

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
  }

}
```

```
// set runmode for unit test
context.runMode("author");
```

```
context.bundleContext()
context.componentContext()
context.currentPage()
context.currentResource()
context.pageManager()
context.request()
context.requestPathInfo()
context.response()
context.slingScriptHelper()
...
```

Resource Resolver Types

- **JCR_MOCK**
  - Mocked JCR with real Sling Resource-JCR Mapping
- **RESOURCERESOLVER_MOCK**
  - Mocked Resource Resolver from Sling Testing
- **JCR_JACKRABBIT**
  - Real Jackrabbit with real Sling Resource-JCR

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);
}
```

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());
```

Registering OSGi Service

```
// 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.slingScriptHelper().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);
```

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();
}
```

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
}
```

Building Content

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

// create resource
context.create().resource("/content/test1", ImmutableMap.<String, Object>builder()
    .put("prop1", "value1")
    .put("prop2", "value2")
    .build());
```

Get it

- Available at Maven Central

```
<dependency>
  <groupId>io.wcm</groupId>
  <artifactId>io.wcm.testing.aem-mock</artifactId>
  <version>1.0.0</version>
  <scope>test</scope>
</dependency>
```

- Documentation: http://wcm.io/testing