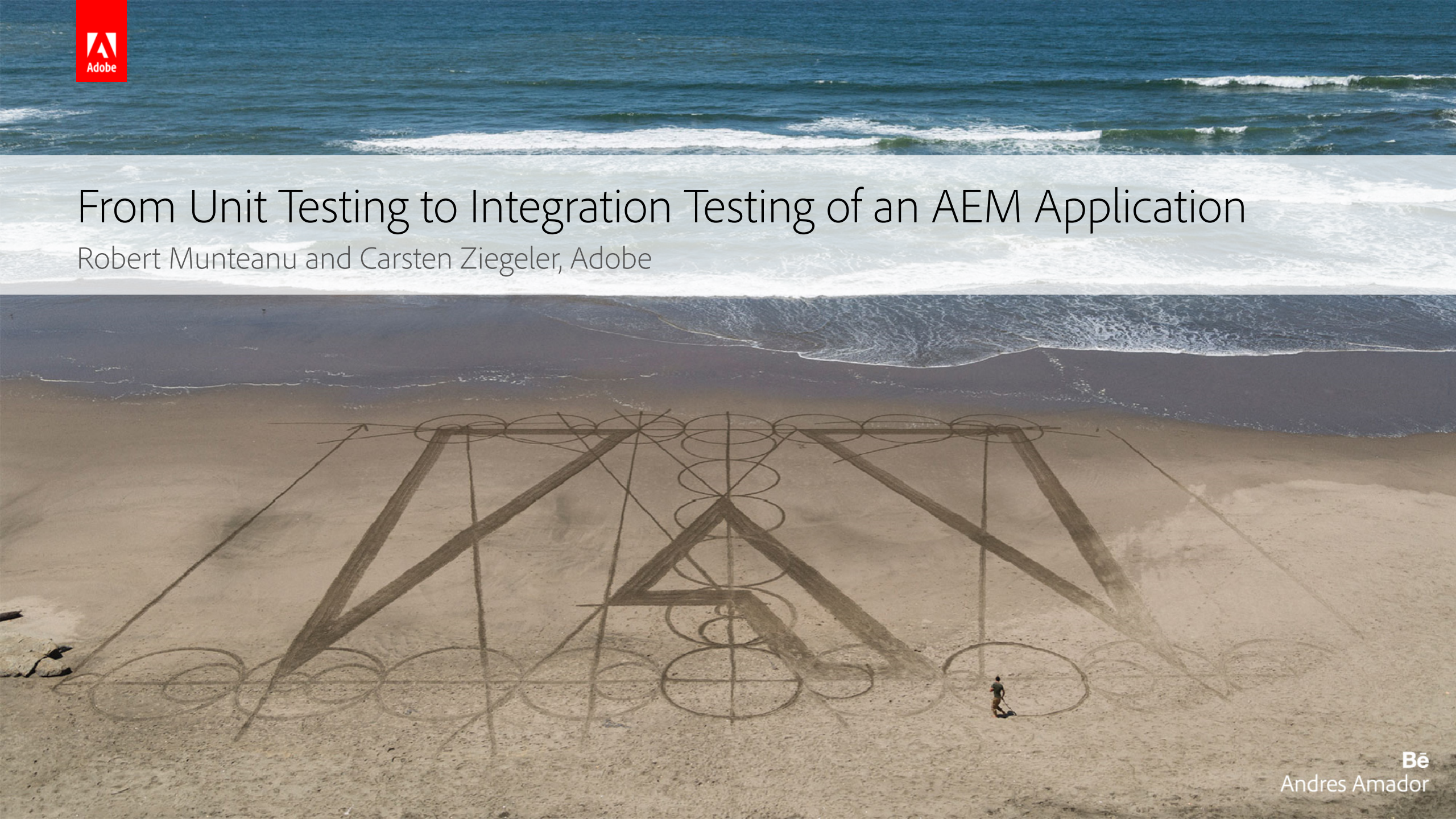
From Unit Testing to Integration Testing of an AEM Application

Robert Munteanu and Carsten Ziegeler, Adobe



Robert Munteanu

- Adobe Systems Romania
- AEM Developer, focusing on lower layers of the stack
- Apache Sling PMC member

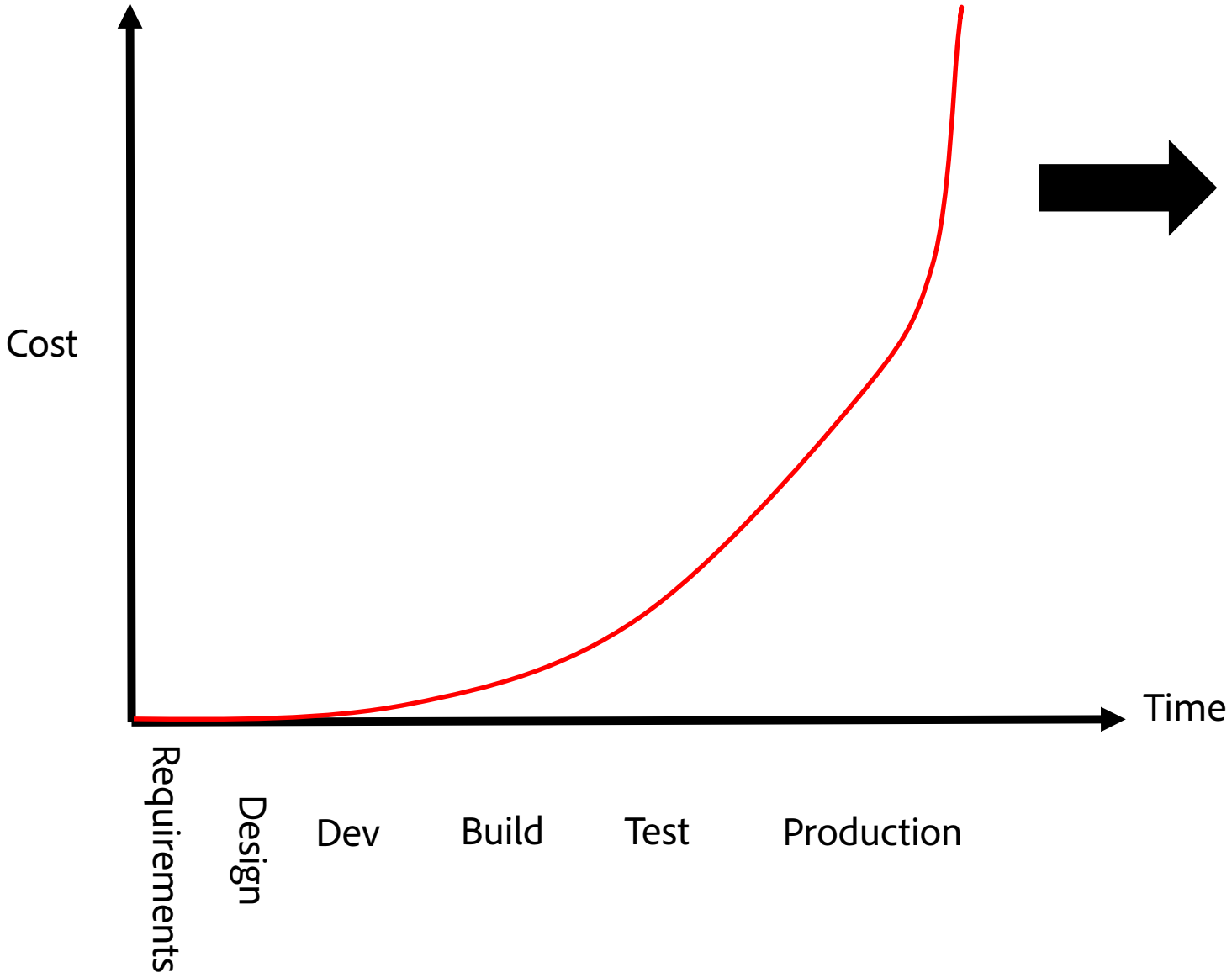
Carsten Ziegeler

- RnD Adobe Research Switzerland
- Project Lead / Founder of Adobe Granite
- Member of the Apache Software Foundation
- VP of Apache Felix and Sling
- OSGi Expert Groups and Board member

Agenda

- Unit testing intro
- OSGi Unit testing
- Sling Unit testing
- AEM Mocks
- Pax-Exam
- End-to-End testing

The Cost of Fixing Defects



The earlier defects are found the less expensive is the fix

Automated testing is imperative!

Frequent and fast testing!

Adobe

Testing an AEM Application

- Java Code
- Scripts
- Javascript
- REST API
- HTML / CSS

Phase One – Unit Testing

- Maven Setup
  - All tests go at src/test/java
  - Include junit as test dependency
- Create test class, annotate method with org.junit.Test annotation
- Assertions from org.junit.Assert.*
- Use Before / After annotation for setup / clean up

JUnit Test

Source src/main/java: org.apache.sling.resource.ResourceUtil

```
public static @CheckForNull String getParent(@Nonnull String path) {
...
```

Test src/test/java: org.apache.sling.resource.ResourceUtilTest

```
@Test public void testGetParent() {
        assertNull(ResourceUtil.getParent("/"));
        assertNull(ResourceUtil.getParent("/.."));

        assertEquals("/", ResourceUtil.getParent("/b"));
        assertEquals("b/c", ResourceUtil.getParent("b/c/d"));
        assertEquals("/b/c", ResourceUtil.getParent("/b/c/d"));
        ...
}
```

Mocking

Business objects, services, data...

- Use real classes / implementations
- Create own implementations
- Mocks

Mocking

```
public class MyService {

     public String readData(ResourceResolver resolver, String path) {
          Resource r = resolver.getResource(path);
          return r == null ? null : r.adaptTo(String.class);
     }
}
```

```
@Test public void testReadData() {

        MyService service = new MyService();
        assertEquals("OK", service.readData(resolver, "/content/data/testdata"));
}
```

Mocking with mockito

```
public class MyService {

     public String readData(ResourceResolver resolver, String path) {
          Resource r = resolver.getResource(path);
          return r == null ? null : r.adaptTo(String.class);
     }
}

@Test public void testReadData() {
        ResourceResolver resolver = mock(ResourceResolver.class);
        Resource rsrc = mock(Resource.class);
        when(resolver.getResource("/content/data/testdata")).thenReturn(rsrc);
        when(rsrc.adaptTo(String.class)).thenReturn("OK");

        MyService service = new MyService();
        assertEquals("OK", service.readData(resolver, "/content/data/testdata"));
}
```

Mocking with mockito

```
@Test public void testReadData() {
        ResourceResolver resolver = mock(ResourceResolver.class);
        Resource rsrc = mock(Resource.class);
        when(resolver.getResource("/content/data/testdata")).thenReturn(rsrc);
        when(rsrc.adaptTo(String.class)).thenReturn("OK");

        MyService service = new MyService();
        assertEquals("OK", service.readData(resolver, "/content/data/testdata"));

        verify(resolver).getResource("/content/data/testdata");
        verifyNoMoreInteractions(resolver);
}
```

Mocking

- Easy but powerful
- Expectations
- Verifications
- Complex matching possible

- But keep it simple – or do you want to write tests for your tests?
- And/or: „Don‘t mock a type you don‘t own“

Phase Two– Apache Sling's Resource Resolver Mock Library

Testing Code using Resource API

- [redacted]
- [redacted]
- [redacted]

Apache Sling Testing Resource Resolver Mock

Adobe

Apache Sling Testing Resource Resolver Mock

```
ResourceResolverFactory factory = new MockResourceResolverFactory();
ResourceResolver resolver = factory.getResourceResolver(null);


resolver.getResource("/my/path");
resource.adaptTo(ValueMap.class);
resolver.create(parent, "new resource", properties);
resolver.delete(resource);
resource.adaptTo(ModifiableValueMap.class);
resolver.commit();
resolver.revert();
```

Apache Sling Testing Resource Resolver Mock

```
MockHelper.create(resolver)
 .resource("/apps")
   .resource("1").p("a", "1").p("b", "2")
   .resource(".2").p(ResourceResolver.PROPERTY_RESOURCE_TYPE, "apps")
   .resource(".3").p("e", "2")
                  .p(ResourceResolver.PROPERTY_RESOURCE_TYPE, "apps")

 .resource("/libs")
   .resource("1").p("d", "1").p("b", "5")
   .resource(".2").p(ResourceResolver.PROPERTY_RESOURCE_TYPE, "apps")
   .resource(".4").p("f", "2");
```

Phase Three– Apache Sling's Mock Library Family

Andres Amador

Mocks are great, but what about....

- OSGi services
- Especially referencing other services
- OSGi configurations
- More complicated test setups
- ...

Family of Apache Sling Mocking Libraries

- Sling Mocks
- JCR Mocks
- OSGi Mocks
- Tooling

Simple OSGi Example - Service

```
public interface MyService {

    String getMessage();
}
```

Simple OSGi Example - Implementation

```
@Component(service=MyService.class)
public class MyComponent implements MyService {

    @Reference
    private EventAdmin eventAdmin;

    private String message = "Hello";

    @Activate
    private void activate(final Map<String, Object> config) {
        message = (String) config.get("message");
    }

    @Override
    public String getMessage() {
        return message;
    }
}
```

Simple OSGi Example - Test

```
public class MyComponentTest {

    @Rule
    public SlingContext context = new SlingContext();

    @Test public void testMessage() {
        context.registerService(EventAdmin.class, mock(EventAdmin.class));

        MyService service = context.registerInjectActivateService(
                new MyComponent(),
                Collections.singletonMap("message", "YOU"));

        assertEquals("YOU", service.getMessage());
    }
}
```

Phase Four– AEM Mocks

Andres Amador

AEM Mocks - Introduction

- Natural extension of OSGi/JCR/Sling mocks to AEM
- Provides implementations for key components of AEM
  - PageManager, Page, Template
  - ComponentManager, Component
  - TagManager, Tag
  - Designer
  - ComponentContext, EditContext, EditConfig
  - Asset, Rendition

AEM Mocks – Test Example

```
public class AemAwareTest {

    @Rule public AemContext ctx = new AemContext();

    @Test public void workWithPages() throws WCMException {

        ctx.pageManager().
            create("/", "content", "/apps/my/templates/home", "Home");
        ctx.pageManager().
            create("/content", "welcome", "/apps/my/templates/hello", "Welcome");

        Iterator<Page> children = ctx.pageManager().getPage("/").listChildren();

        assertThat("Number of child pages for /", count(children), equalTo(1));
    }
}
```

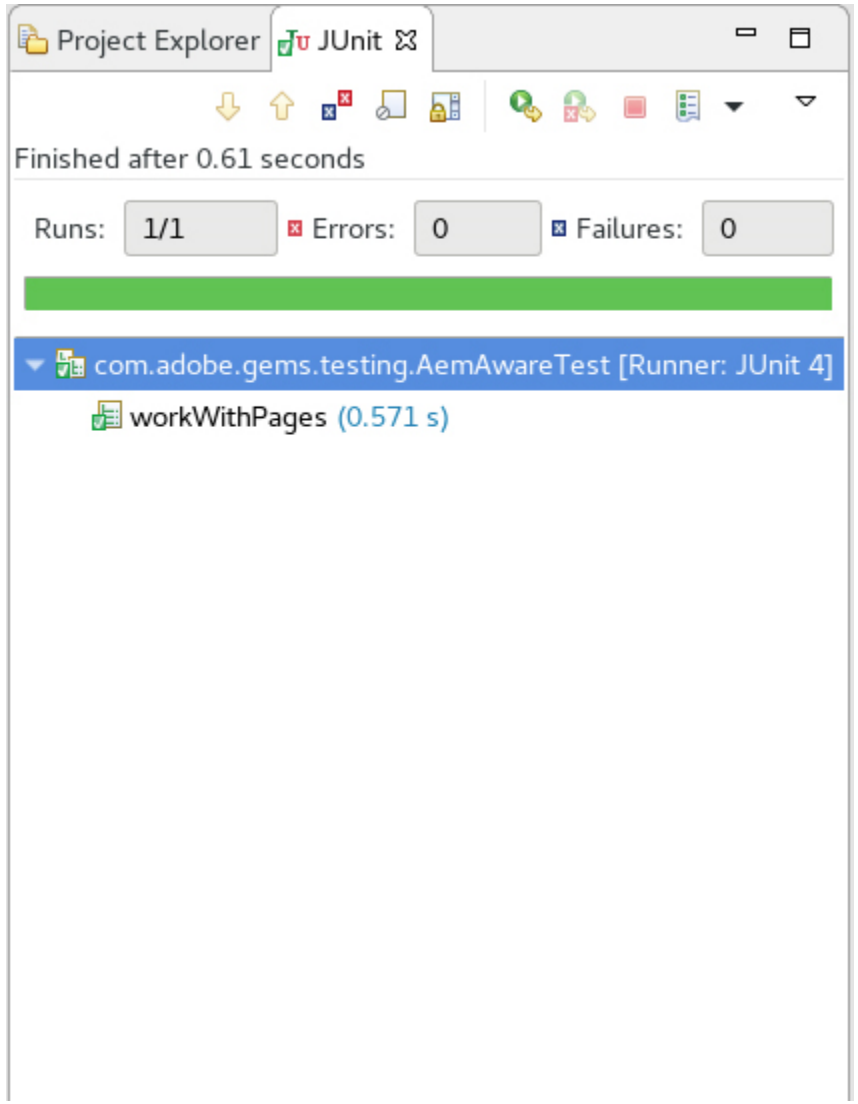
AEM Mocks – Custom Setup Callback

```
private static final class SetUpCallback implements AemContextCallback {

    @Override
    public void execute(AemContext context) throws PersistenceException,
IOException {

        // application-specific services for unit tests
        context.registerService(CryptoSupport.class,
                                new MockCryptoSupport());
        context.registerService(Externalizer.class,
                                new MockExternalizer());

        // import sample content
        context.load().json("/sample-content.json", "/content/sample/en");

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

AEM Mocks – Custom Setup Callback

```
public class GemsAemContext {

    public static AemContext newAemContext() {
        return new AemContext(new SetUpCallback());
    }
}

public class AemAwareTest {

    @Rule public AemContext ctx = GemsAemContext.newAemContext();
    /* unchanged code omitted */
}
```

AEM Mocks – Execution Speed



Balance speed with mock implementation coverage

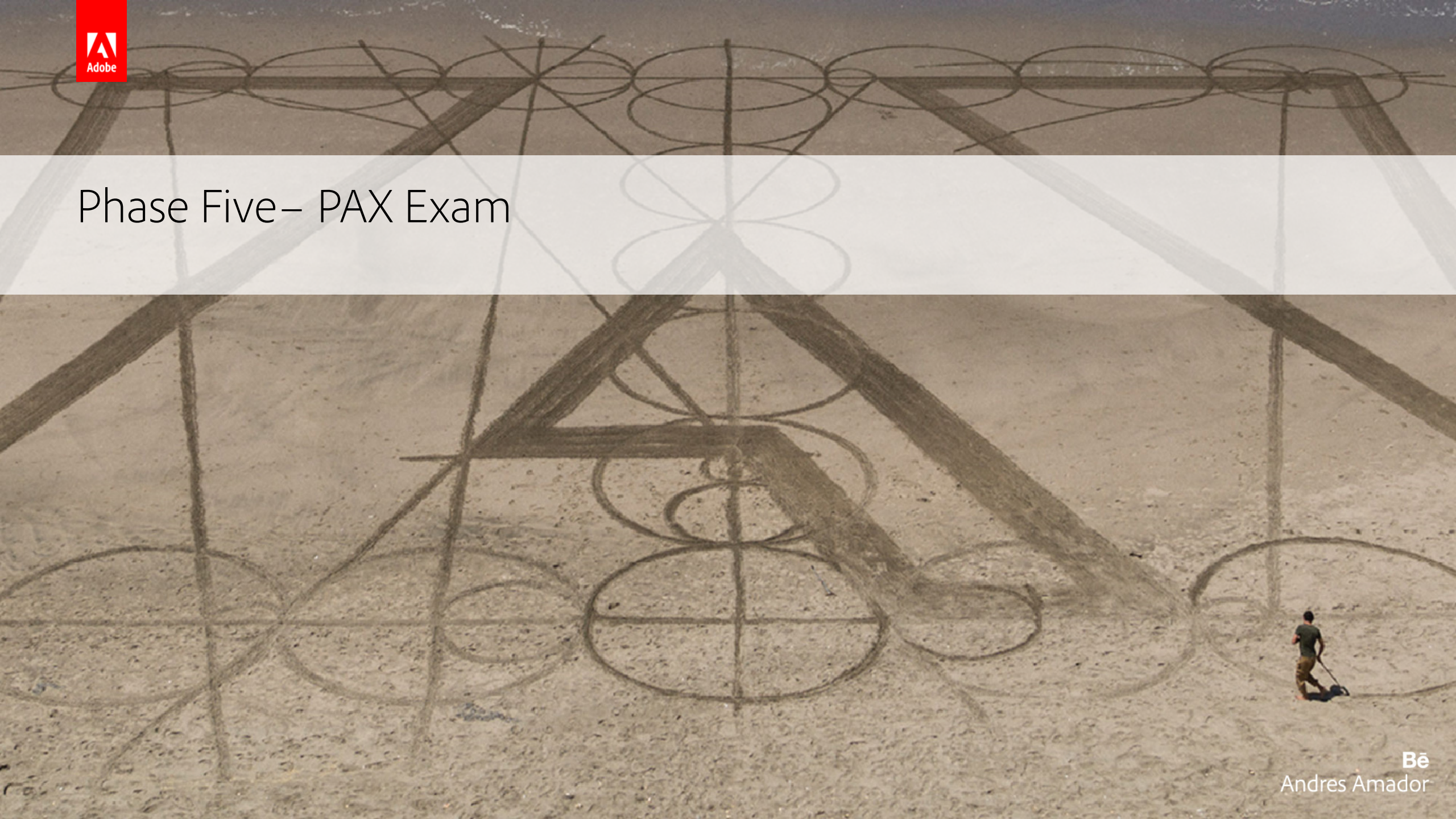
- Resource Resolver Mock (fastest)
- JCR Mock
- ~~JCR Jackrabbit~~
- JCR Oak (feature-complete JCR repository)

Adobe

AEM Mocks – pom.xml snippet

```
<dependency>
    <groupId>io.wcm</groupId>
    <artifactId>io.wcm.testing.aem-mock</artifactId>
    <!-- for AEM 6.0 or 6.1 use version 1.x -->
    <version>2.1.0</version>
    <scope>test</scope>
</dependency>
```

Phase Five– PAX Exam



Andres Amador

OSGi Integration Testing with PAX Exam

- Launch a full OSGi application
  - **Framework**
  - **Bundles**
  - **Configurations**
- Run your test within that framework as a separate bundle

Simple Example with PAX

```
@RunWith(PaxExam.class)
public class SampleTest {

    @Inject
    private HelloService helloService;

    @Configuration
    public Option[] config() {
        return options(
            mavenBundle("com.example.myproject", "myproject-api", "1.0.0-SNAPSHOT"),
            junitBundles());
    }

    @Test
    public void getHelloService() {
        assertNotNull(helloService);
        assertEquals("Hello Pax!", helloService.getMessage());
    }
}
```


Source:https://ops4j1.jira.com/wiki/display/PAXEXAM4/Getting+Started+with+OSGi+Tests

Phase Six – End-to-end Tests

End-to-end Tests – general concepts

- Require one or more AEM instances to be started
- Slower, more difficult to write correctly
  - Slower hardware often reveals timing issues
  - Sensitive to environment changes
- Two variations supported by AEM
  - Black-box HTTP based via the Apache Sling Testing Clients and Rules
  - White-box testing via the Apache Sling Test Tools

Apache Sling Testing Clients and Rules

```
public class NewsletterCSVExportIT {

    @ClassRule
    public static SlingInstanceRule slingInstanceRule = new SlingInstanceRule();

    @Test
    public void csvExportContainsKnownUser() throws Exception {
    slingInstanceRule
        .getAdminClient()
        .doGet("/content/newsletter/subscribers.csv", HttpServletResponse.SC_OK)
        .checkContentContains("1,John Doe, john@doe.org");
    }
}
```

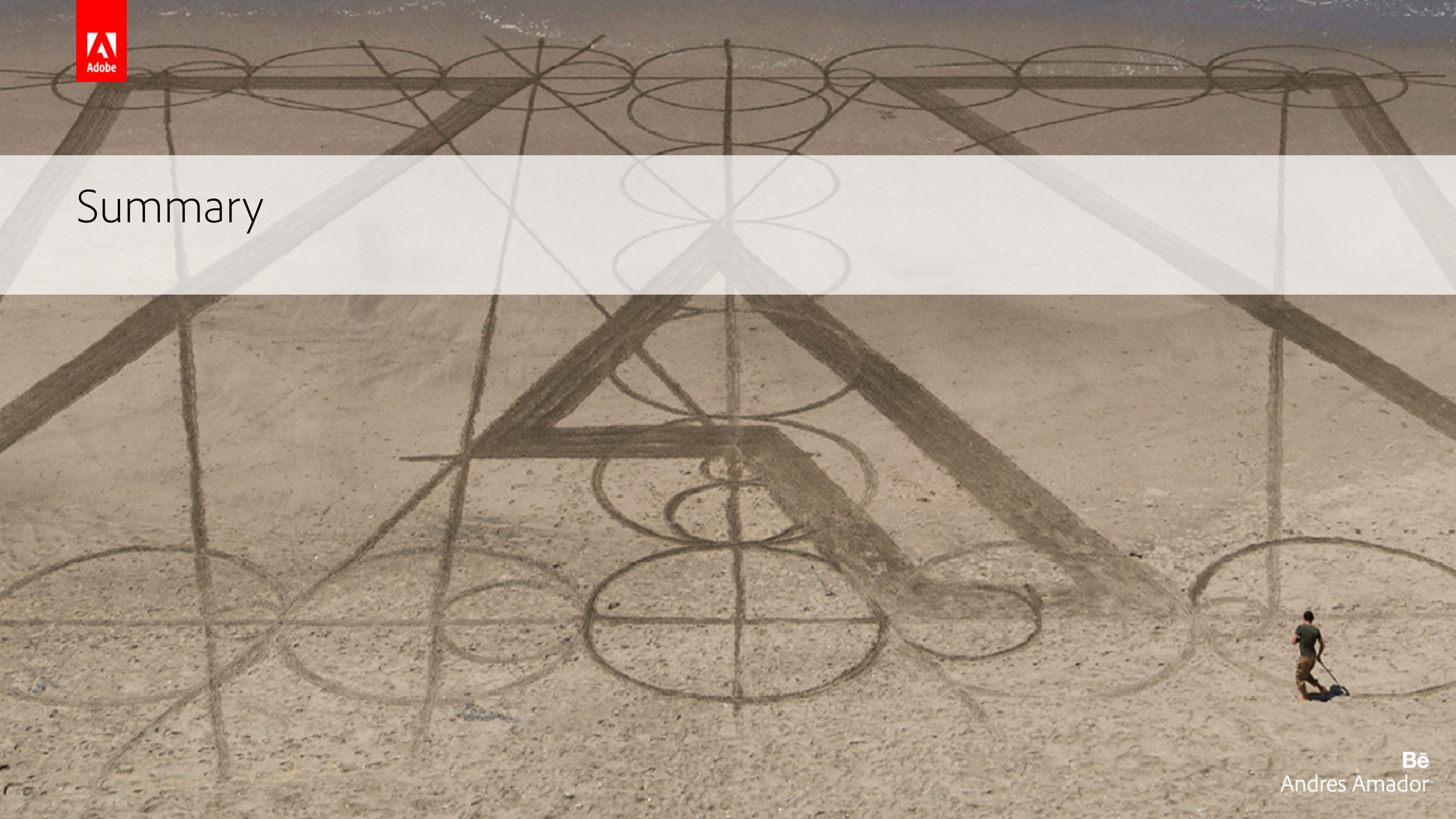
Apache Sling Testing Clients and Rules

```
public class AuditorTest {

    @Rule public final TeleporterRule teleporter = TeleporterRule.forClass(getClass(),
"Launchpad");

    @Test public void auditorHasNoEventsBeforePurge () throws IOException {

        final Auditor auditor = teleporter.getService(Auditor.class);
        assertNotNull(Auditor.class.getSimpleName(), auditor);
        assertThat("No purge events before action", auditor.getPurgeEventsCount(),
equalTo(0));
    }
}
```

Teleporter tests versus Pax-Exam

- Pax-Exam usually based on the „Bill of materials“ pattern
  - Lists all dependencies and assemble an OSGi runtime
- Teleporter tests are usually run against a full AEM instance
  - Dependencies are pulled in via a certain AEM version – for instance AEM 6.2
- Rules of the thumb
  - Prefer Pax-Exam tests for libraries which have a few well-defined dependencies
  - Prefer Teleporter tests for applications which should run on a certain AEM release

Summary



Fast and early testing

- Unit tests
- Mocking
- Special mock libraries
- Integration testing

Links

- Maven: https://maven.apache.org
- JUnit: http://junit.org/junit4/
- Mockito: http://site.mockito.org
- Apache Sling Mocks:
  - https://sling.apache.org/documentation/development/resourceresolver-mock.html
  - https://sling.apache.org/documentation/development/sling-mock.html
  - https://sling.apache.org/documentation/development/osgi-mock.html
  - https://sling.apache.org/documentation/development/jcr-mock.html

QnA