

**adaptTo()**

APACHE SLING & FRIENDS TECH MEETUP  
BERLIN, 28-30 SEPTEMBER 2015

So how do I test my Sling Application?  
Robert Munteanu, Adobe Systems Romania

# About the speaker

- Apache Sling PMC member
- Long-time Open Source contributor
- Working with Adobe on AEM

# Agenda

- How do I test my application?
- What about Sling?
- Demo

```
questionQueue.add(anyQuestion());
```

# How do I test my application?

# Unit testing

```
@Test public void isNull() {  
    assertThat( StringUtils.isNull(  
null ), is(true));  
}
```

# Unit testing with mocks

```
@Test public void persist() {  
    MyDao dao = mock(MyDao.class);  
    MyService service = new MyService(dao);  
    service.persist(new  
ServiceObject()); // must not fail  
}
```

# Unit testing with mocks

```
@Test(expected=ServiceException.class)
public void persist() {
    MyDao dao = mock(MyDao.class);
    MyService service = new MyService(dao);
    when(dao.persist(anyObject)).thenThrow(new
    DaoUnavailableException("mocked"));
    service.persist(new ServiceObject());
}
```



# Integration testing

```
@Test public void persist() {  
    MyDao dao = new MyRealDao(/* config */);  
    MyService service = new MyService(dao);  
    service.persist(newServiceObject()); //  
must not fail  
}
```

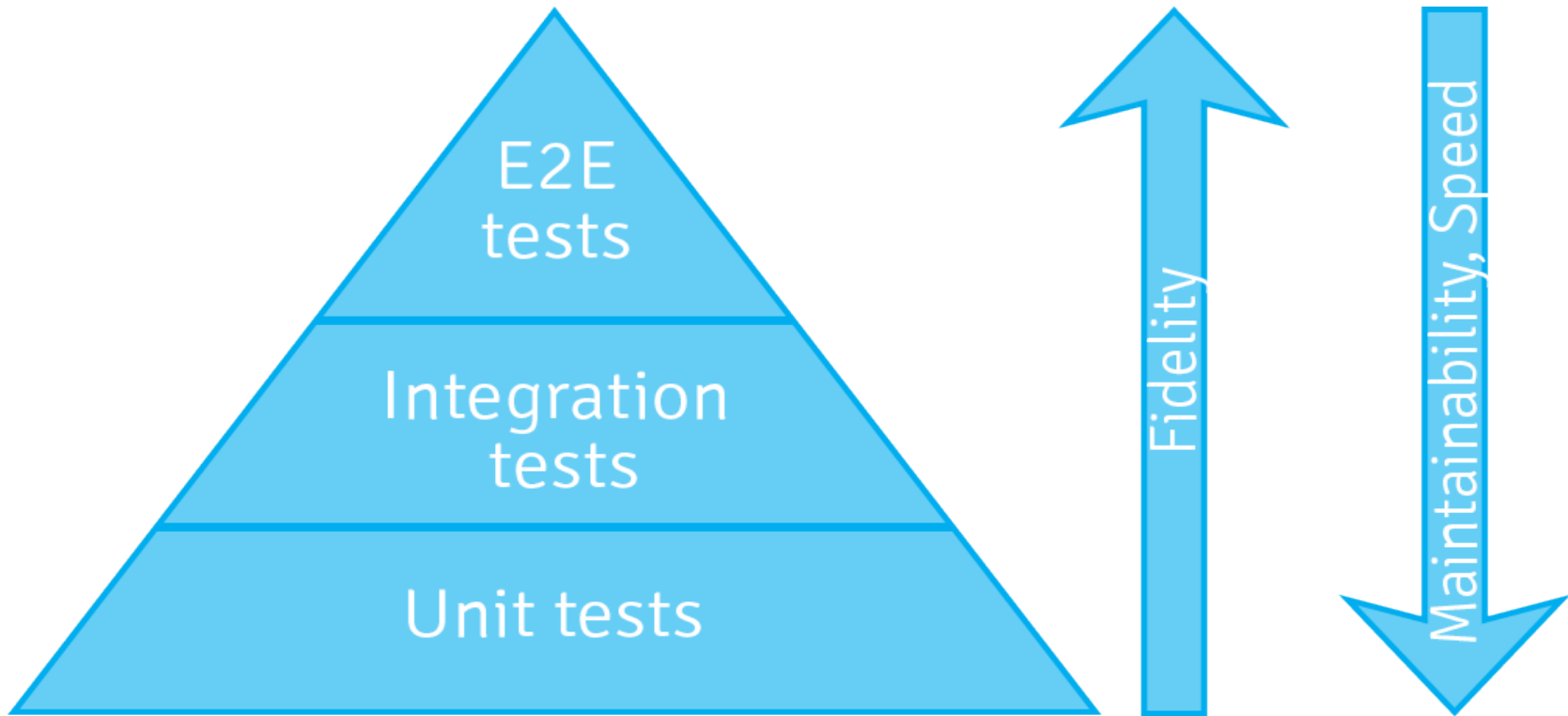
# Integration testing - clean slate

```
@Before public void ensureCleanSlate()
{
    MyDao dao = new MyRealDao(/* config
*/);
    dao.deleteAll();
}
```

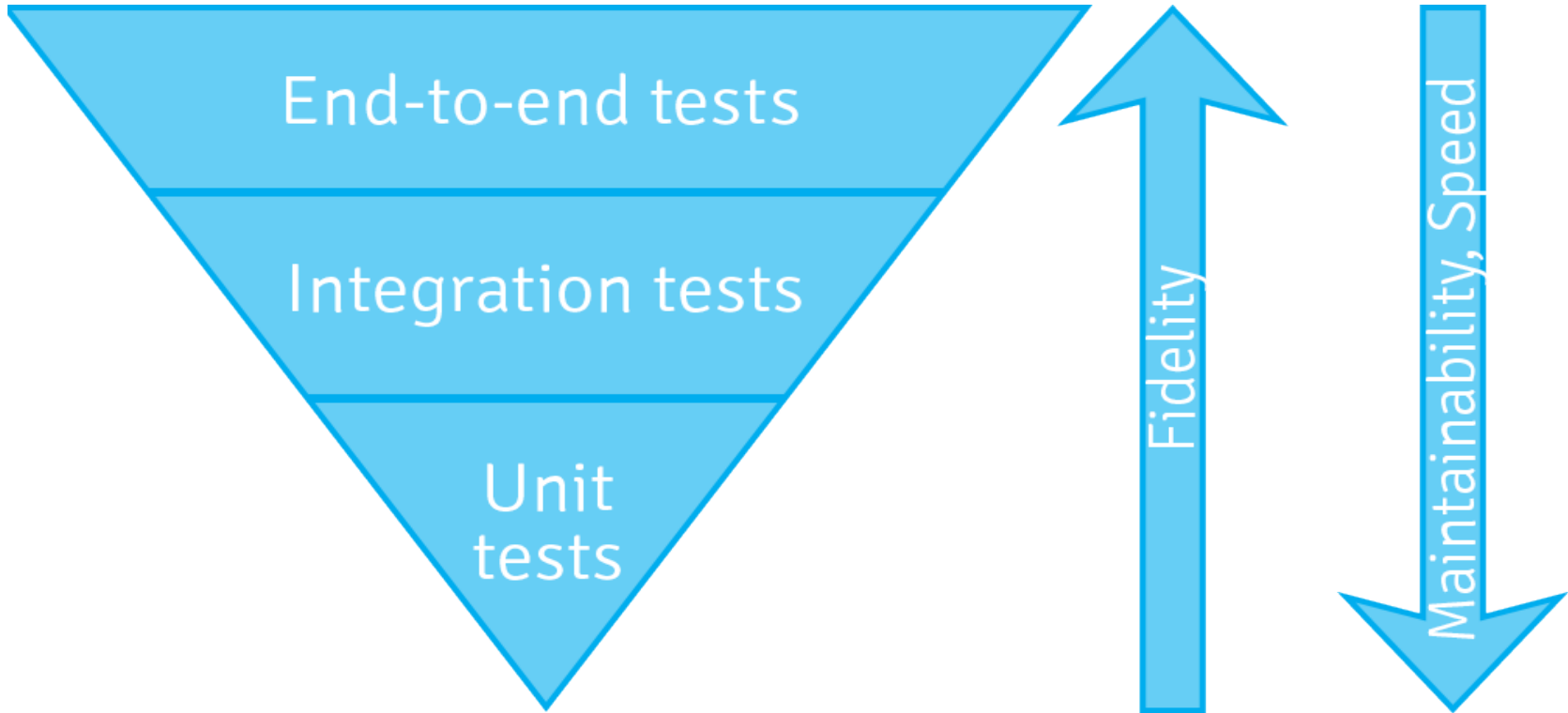
# End-to-end testing

```
@Test public void login() {  
    Client client = new MyBrowserBasedClient();  
    AuthResult result = client.login("admin",  
    "admin");  
    assertThat(result.isLoggedIn(), is(true));  
}
```

# Testing pyramid



# ... or the testing icecream cone?



# What about Sling?

# Unit testing with Sling

# Unit testing

```
@Test public void test_isRedirectValid_null_empty() {  
    TestCase.assertFalse(AuthUtil.isRedirectValid(null,  
null));  
    TestCase.assertFalse(AuthUtil.isRedirectValid(null,  
""));  
}
```

Actual code from [AuthUtilTest.java](#)



# Unit testing with ad-hoc mocking

```
@RunWith(JMock.class) public class AuthUtilTest {
    final Mockery context = new JUnit4Mockery();
    final ResourceResolver resolver = context.mock(ResourceResolver.class);
    final HttpServletRequest request = context.mock(HttpServletRequest.class);

    @Test public void test_isRedirectValid_invalid_characters() {
        context.checking(new Expectations() { /* 8 lines of code */ })
            TestCase.assertFalse(AuthUtil.isRedirectValid(request,
"/illegal/</x"));
    }
}
```

Actual code also from [AuthUtilTest.java](#)

# Unit testing with ad-hoc mocking

```
context.checking(new Expectations() {
    {
        allowing(request).getContextPath();
        will(returnValue("/ctx"));

        allowing(request).getAttribute(with(any(String.class)));
        will(returnValue(null));
    }
});
```

# Unit testing with Sling mocks

```
public class ModelAdapterFactoryUtilTest {  
  
    @Test public void testRequestAttribute() {  
        MockSlingHttpServletRequest request = new MockSlingHttpServletRequest();  
        request.setAttribute("prop1", "myValue");  
        RequestAttributeModel model =  
request.adaptTo(RequestAttributeModel.class);  
        assertNotNull(model);  
        assertEquals("myValue", model.getProp1());  
    }  
}
```

Actual code from [ModelAdapterFactoryUtilTest.java](#).

The background of the slide is a close-up photograph of a stone wall. The stones are irregular in shape and size, with various shades of grey, brown, and tan. The mortar between the stones is a light, weathered grey. The overall texture is rough and aged.

An ideal test runtime?

# Unit testing OSGi code with Sling mocks

```
public class ExampleTest {

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

    @Test public void testSomething() {

        // register and activate service
        MyService service1 = context.registerInjectActivateService(new MyService(),
            ImmutableMap.<String, Object>of("prop1", "value1"));

        // get service instance
        OtherService service2 = context.getService(OtherService.class);
    }
}
```

Picked up from the [OSGi mocks documentation](#).

# Sling mocks

- donated to the Sling project in October 2014 by Stefan Seifert
- focus on running Apache Sling in an in-memory environment suitable for unit tests

# The humble object pattern with OSGi

```
public interface RouterAdmin {
    void doStuff();
}

public class RouterAdminImpl implements RouterAdmin {
    // constructor and field elided
    public void doStuff() { // implementation }
}

@Component @Properties({ @Property(name="url") })
public class RouterAdminComponent implements RouterAdmin {
    private RouterAdmin delegate;

    protected void activate(ComponentContext ctx) throws Exception {
        delegate = new RouterAdminImpl(new URL(requireString(ctx, "url")));
    }

    public void doStuff() {
        delegate.doStuff();
    }
}
```



A large, leafy tree stands in the center of a vast, green field. The background is a hazy, overcast sky, creating a soft, atmospheric setting. The text "An active repository" is overlaid in white, centered horizontally and partially overlapping the tree's canopy.

An active repository



# Testing JCR code with Sling mocks

```
public class FindResourcesTest {

    @Rule public SlingContext context = new SlingContext(ResourceResolverType.JCR_MOCK);

    @Before public void setUp() {
        Resource resource = context.create().resource("test",
            ImmutableMap.<String, Object> builder().put("prop1", "value1")
                .put("prop2", "value2").build());
        // snip ...
        MockJcr.setQueryResult(session, Collections.singletonList(node));
    }

    @Test public void testFindResources() {
        Resource resource = context.resourceResolver().getResource("/test");
        Assert.assertNotNull("Resource with name 'test' should be there", resource);

        Iterator<Resource> result = context.resourceResolver().findResources("/test", Query.XPATH);
        Assert.assertTrue("At least one result expected", result.hasNext());
        Assert.assertEquals("/test", result.next().getPath());
        Assert.assertFalse("At most one result expected", result.hasNext());
    }
}
```

Code taken from [FindResourceTest.java](#) .

# Sling

A photograph of two slings lying on a dark asphalt surface. The one on the left is blue with yellow straps, and the one on the right is light-colored wood. The text "The weapon of choice" is overlaid in white, centered across the image.

The weapon of choice

# Sling Mocks for unit testing

```
public class SimpleNoSqlResourceProviderQueryTest {

    @Rule public SlingContext context = new SlingContext(ResourceResolverType.JCR_MOCK);

    @Before public void setUp() throws Exception {
        context.registerInjectActivateService(new SimpleNoSqlResourceProviderFactory(), ImmutableMap.<String, Object>builder()
            .put(ResourceProvider.ROOTS, "/nosql-simple")
            .build());

        // prepare some test data using Sling CRUD API ( not shown here )
    }

    @Test
    public void testFindResources_ValidQuery() {
        Iterator<Resource> result = context.resourceResolver().findResources("all", "simple");
        assertEquals("/nosql-simple", result.next().getPath());
        assertEquals("/nosql-simple/test", result.next().getPath());
        assertEquals("/nosql-simple/test/node1", result.next().getPath());
        assertEquals("/nosql-simple/test/node2", result.next().getPath());
        assertFalse(result.hasNext());
    }
}
```

Code from [SimpleNoSqlResourceProviderQueryTest](#)

# Sling Mocks for unit testing

```
-----  
T E S T S  
-----
```

```
Running org.apache.sling.nosql.generic.simple.SimpleNoSqlResourceProviderQueryTest
```

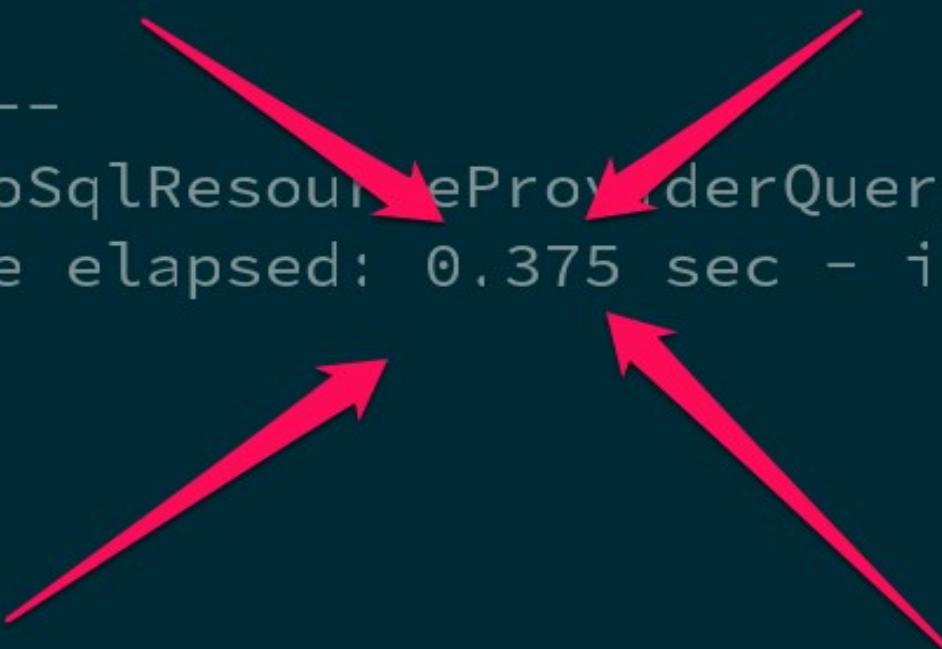
```
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.375 sec - in org.apache.sling.nosql.generic.simple.SimpleNoSqlResourceProviderQueryTest
```

```
Results :
```

```
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
```

# Sling Mocks for unit testing

```
-----  
imple.SimpleNoSqlResourceProviderQueryTest  
ipped: 0; Time elapsed: 0.375 sec - in org  
  
ipped: 0
```

Four red arrows point towards the code. Two arrows point to the word "Provider" in the first line, and two arrows point to the word "Time" in the second line.

# Hamcrest matchers

```
@Test public void loadResources() {  
  
    Map<String, Object> expectedProperties = /* define properties */;  
    Resource resource = /* load resource */ null;  
  
    assertThat(resource, resourceOfType("my/app"));  
    assertThat(resource, hasChildren("header", "body"));  
    assertThat(resource, resourceWithProps(expectedProperties));  
}
```

Taken from the [Hamcrest integration documentation](#).

# Integration testing with Sling

# Pax-Exam

```
public static Option[] paxConfig() {
    final File thisProjectsBundle = new File(System.getProperty( "bundle.file.name",
"UNDLE_FILE_NOT_SET" ));
    final String launchpadVersion = System.getProperty("sling.launchpad.version",
"LAUNCHPAD_VERSION_NOT_SET");
    log.info("Sling launchpad version: {}", launchpadVersion);
    return new DefaultCompositeOption(
        SlingPaxOptions.defaultLaunchpadOptions(launchpadVersion),

CoreOptions.provision(CoreOptions.bundle(thisProjectsBundle.toURI().toString()))

    ).getOptions();
}
```

Taken from [U.java](#)



# Pax-Exam

```
@RunWith(PaxExam.class) public class FileNameExtractorImplIT {  
  
    @Inject private FileNameExtractor fileNameExtractor;  
  
    @Test public void testFileNameExtractor(){  
        String rawPath = "http://midches.com/images/uploads/default/demo.jpg#anchor?query=test";  
        String expectedFileName = "demo.jpg";  
        assertEquals(expectedFileName, fileNameExtractor.extract(rawPath));  
    }  
  
    @Configuration public Option[] config() {  
        return U.paxConfig();  
    }  
}
```

Taken from [FileNameExtractorImplIT.java](#).

# Sling mocks with a JCR backend

- Unit testing with JCR Mocks
  - JCR\_MOCK
  - SLING\_MOCK
- Integration testing with JCR Mocks
  - JCR\_JACKRABBIT
  - JCR\_OAK

# Server-side JUnit tests

```
public class ServerSideInstallerTest {  
  
    @Rule public final TeleporterRule teleporter = TeleporterRule.forClass(getClass(), "Launchpad");  
  
    @Before  
    public void setup() throws LoginException {  
        ip = teleporter.getService(InfoProvider.class);  
        is = ip.getInstallationState();  
    }  
  
    @Test  
    public void noUntransformedResources() {  
        final List<?> utr = is.getUntransformedResources();  
        if(utr.size() > 0) {  
            fail("Untransformed resources found: " + utr);  
        }  
    }  
}
```

# Scriptable server-side tests

```
// test imports that fail on Java 8, SLING-3405

%><%@page import="java.util.Arrays"%><%
%><%@page import="java.lang.CharSequence"%><%
%>TEST_PASSED
```

Actual code from [installer-duplicate.jsp](#).

# End-to-end testing with Sling

# HTTP utilities from the Sling testing tools

```
public class HttpPingTest extends HttpTestBase {
    public void testWebServerRoot() throws Exception {
        // by default, the Launchpad default servlet redirects / to index.html
        final String url = HTTP_BASE_URL + "/";
        final GetMethod get = new GetMethod(url);
        get.setFollowRedirects(false);
        final int status = httpClient.executeMethod(get);
        assertEquals("Status must be 302 for " + url, 302, status);
        final Header h = get.getResponseHeader("Location");
        assertNotNull("Location header must be provided",h);
        assertTrue("Location header must end with index.html",
            h.getValue().endsWith("index.html"));
    }
}
```

Code taken from [HttpPingTest.java](#)

# Code time

# Slingshot





# Final thoughts

- Sling provides a lot of specialised testing tools
- Lots of effort recently invested in the unit testing layer
- Experiment and provide feedback
- Contributions are definitely welcomed

# Resources

- <https://sling.apache.org/documentation/development/sling-mock.html>
- <http://wcm.io/testing/aem-mock/>
- <http://svn.apache.org/repos/asf/sling/trunk/samples/slingshot/>
- <http://xunitpatterns.com/Humble%20Object.html>

# Colophon

- Presentation crafted with Eclipse, GNU Make and [odpdown](#)
- Photo credits
  - [Slingshots](#) by *Anne and Tim* on Flickr
  - [Wall](#) by *Bart Lumber* on Flickr
  - [HTML code](#) by *Marjan Krebelj* on Flickr
  - [Foggy Oak Tree](#) by *Brett Whaley* on Flickr
- Testing pyramid and ice-cream cone concepts courtesy of [watirmelon.com](#)

# Q&A

