APACHE SLING & FRIENDS TECH MEETUP
BERLIN, 26-28 SEPTEMBER 2016

Test Driven Development with AEM

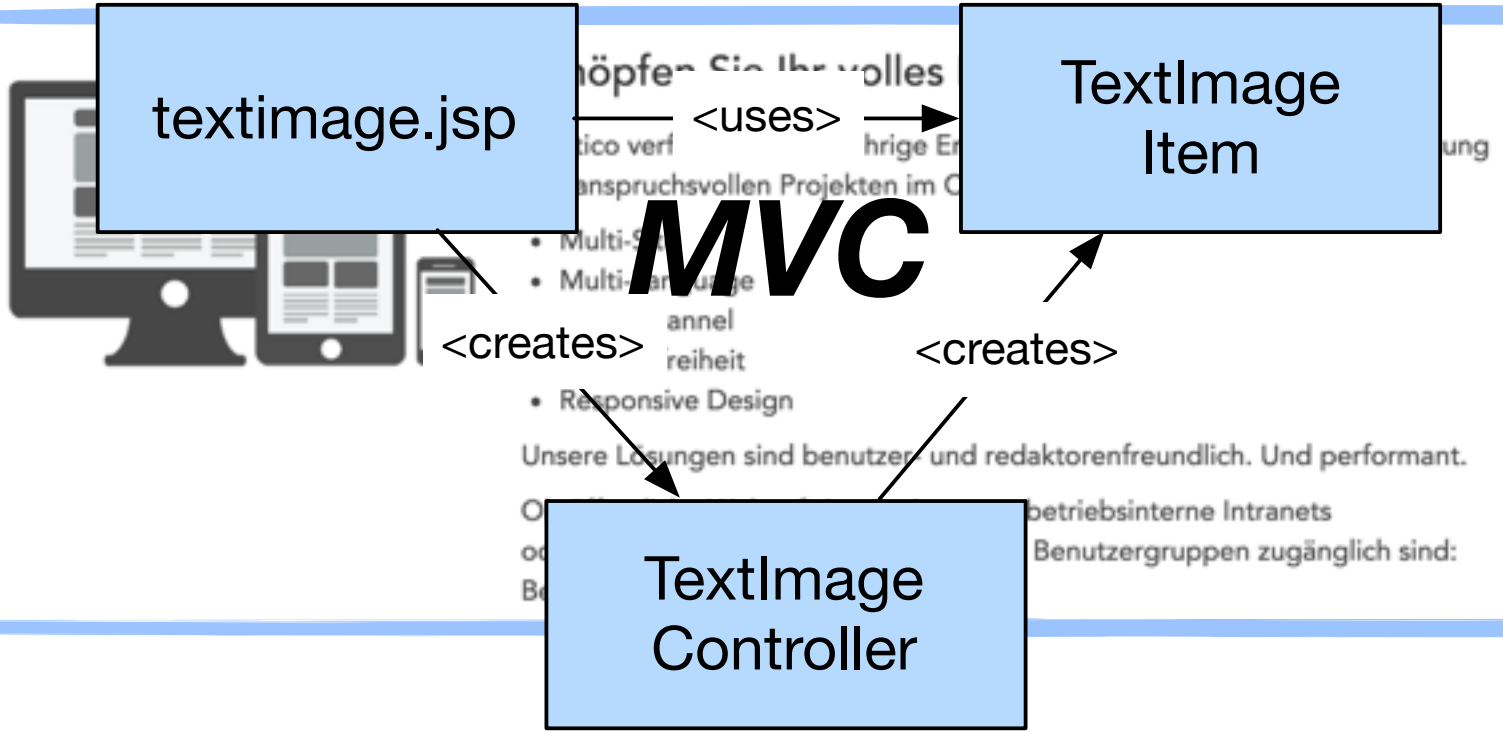
Jan Wloka, Quatico Solutions Inc.

From the talk “Get the Flow”

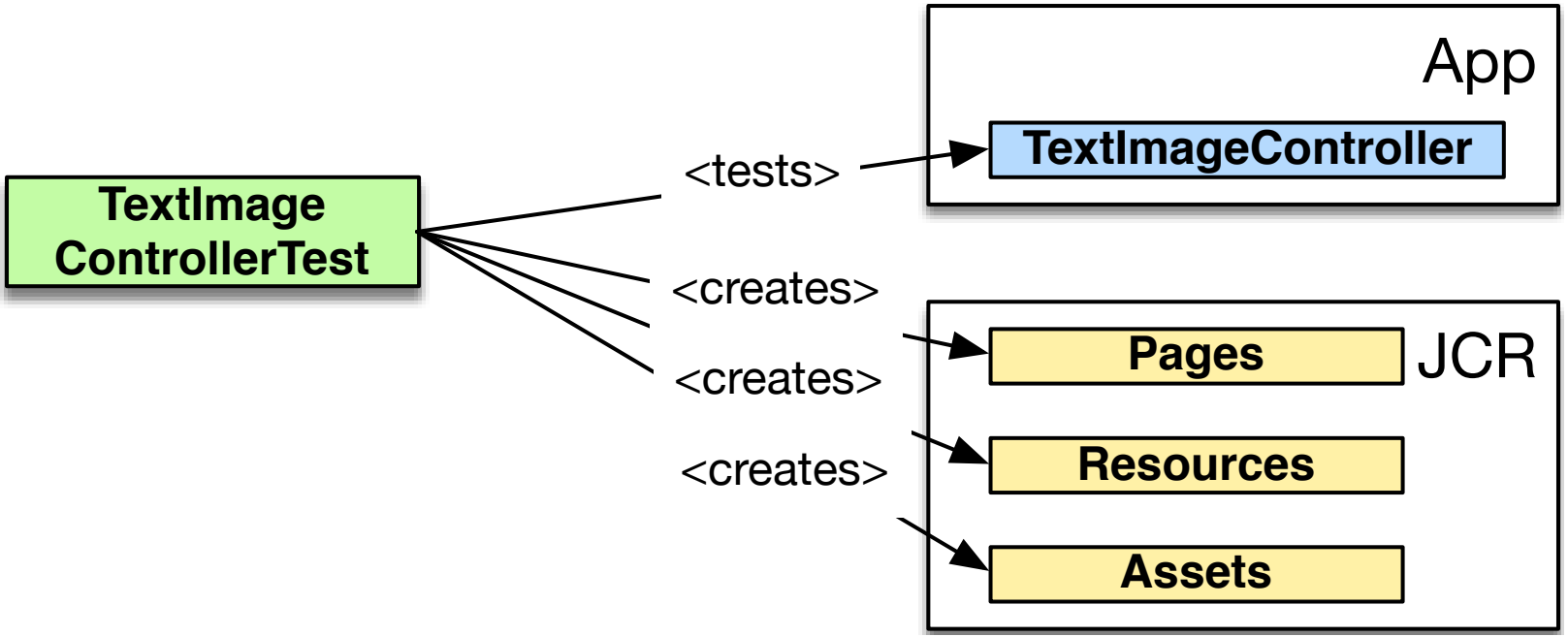
“Test coverage is the natural enemy of fast coding. [...] It is a challenge that will be around forever.”

Web Application with Components

Text Image Component



Testing Text Image Component



Unit Test: getImageLegend() returns caption?

```
@Test
public void getImageLegenWithNoSourceButTitleReturnsCaption() {
  Resource page   = $.aDetailPageWithParents("/content/test/ko/home/page");
  Resource target = $.aTextImage(page, "/jcr:content/foobar");
  $.anImage(target, "image", "caption", "expected");

  TextImageController testObj = new  TextImageController().setup(
    $.aPageContext().component(target).page(page).build()
  );

  assertEquals("expected", testObj.getImageLegend());
}
```

Application specific Testing API

- Creation / Builder methods tailored to your application
  - Create Resources

```
$.aResource(String, Object…) : Resource
$.aResource(Resource, String, Object…) : Resource
```

  - Create Pages

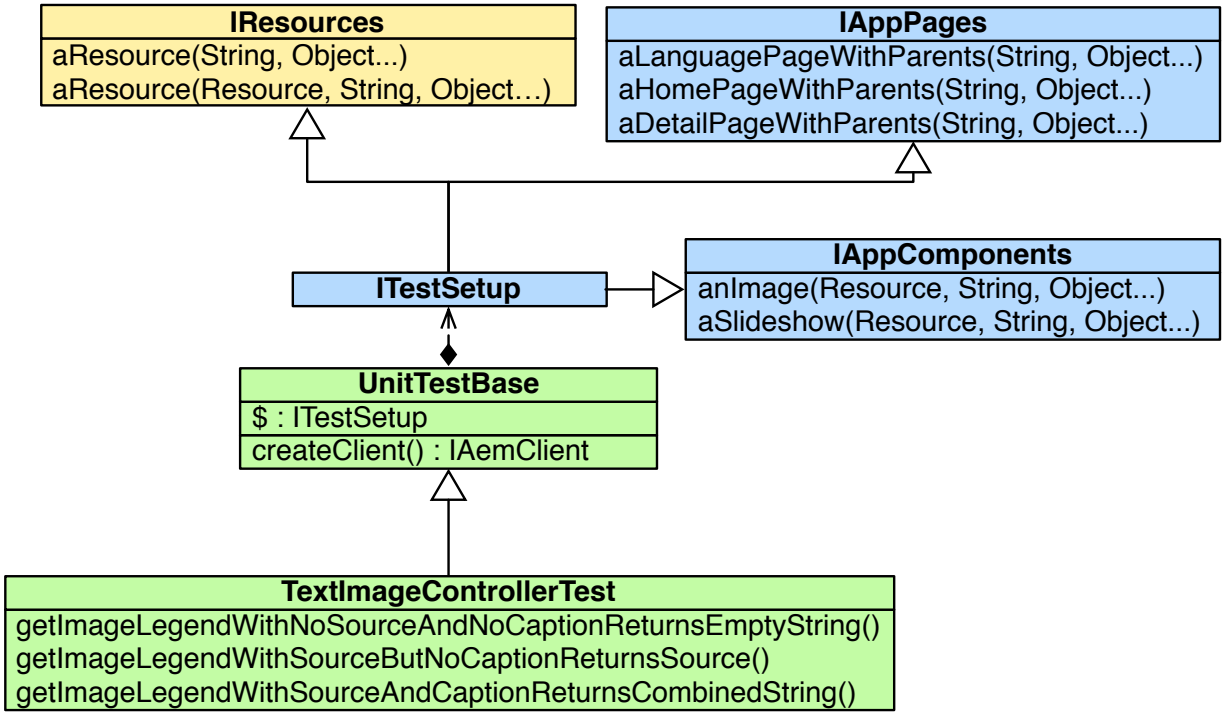
```
$.aLanguagePageWithParents(String, Object..) : Resource
$.aHomePageWithParents(String, Object..) : Resource
$.aDetailPageWithParents(String, Object..) : Resource
$.aPageContext() : PageContextBuilder
```

  - Create Components

```
$.anImage(Resource, String, Object…) : Resource
$.aSlideShow(Resource, String, Object…) : Resource
```

Where is this API coming from?

- Creation Methods for Resources, custom Components and Pages



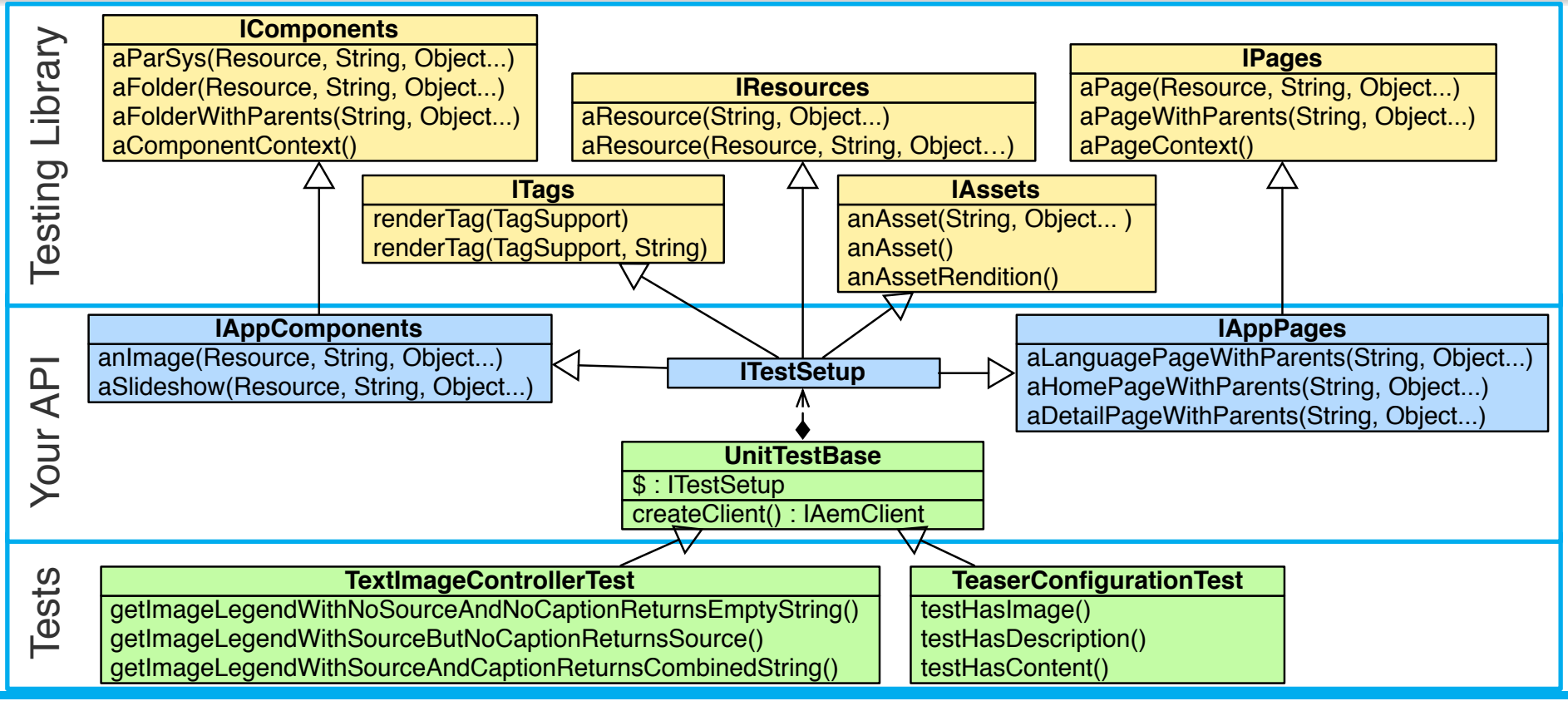
Implementing your creation methods

```
public class AppComponents extends Components implements IAppComponents {
...
public Resource aTeaserConfiguration(Resource parent, TeaserType type, Object… p){
  String configPath = type != null
          ? type.getConfigPath()
          : PageTeaser.PATH_EXTENSION);

  Properties props = new Properties(p).appendIfNotSet(
          "sling:resourceType",
          ComponentType.TEASER_CONFIGURATION);

  return resources.aResource(parent, configPath, props.toArray());
}
...
}
```

Your Application Testing API



AEM Testing Library

- Specifically build for tailored testing APIs
- Based on AEM Mocks (wcm.io/testing)
- Provides common API for creating
  - Resources, Pages, Components, Assets, Tags, Services
- Run Unit (JCR_MOCK) and Integration Tests (JCR_OAK)
- Get it from https://github.com/quatico-solutions/aem-testing

Getting Started: Create your Test Setup

- Setup your testing API, make the '$' available to your tests

```
public class UnitTestBase {

  protected ITestSetup $;
  private IAemClient client;

  @Before
  public void setUpTestContext() throws Exception {
    this.client = new AemClient(Type.Unit).startUp();  // Type.Integration for integrated tests
    IResources resources = new Resources(client);
    IAppPages  pages     = new AppPages(resources, client);
    $ = SetupFactory.create(ITestSetup.class).getSetup(
                  client, resources, pages, new AppComponents(resources, client),
                  new Assets(client), new Tags(pages), new AppServices(client));
  }
  @After
  public void tearDownTestContext() throws Exception {
    this.client.shutDown();
  }
}
```

Getting Started: Add your unit tests

- Extend UnitTestBase, access your API at '$', write more tests.

```
public class TextImageControllerTest extends UnitTestBase {
@Test
public void isShowTitleBelowWithImagePresentAndSizeSmallAndPositionLeftReturnFalse() throws Exception {
  Resource asset  = $.anAsset("/content/dam/test/foo.jpg");
  Resource page   = $.aDetailPageWithParents("/content/test/ko/home/page");
  Resource target = $.aTextImage(page, "/jcr:content/foobar", "text", "<b>hello</b>");
  $.anImage(target, "image", FILE_REFERENCE, asset.getPath(), "imageSize", SMALL, "imagePosition", LEFT_ABOVE);

  assertFalse(new TextImageController().setup($.aPageContext().component(target).page(page).build()).isShowTitleBelow());
}

@Test
public void getImageLegendWithNoSourceButTitleReturnsCaption() throws Exception {
  Resource page   = $.aDetailPageWithParents("/content/test/ko/home/page");
  Resource target = $.aTextImage(page, "/jcr:content/foobar");
  $.anImage(target, "image", "caption", "expectedValue");

  TextImageController testObj = new TextImageController().setup($.aPageContext().component(target).page(page).build());

  assertEquals("expectedValue", testObj.getImageLegend());
}
```

adaptTo()

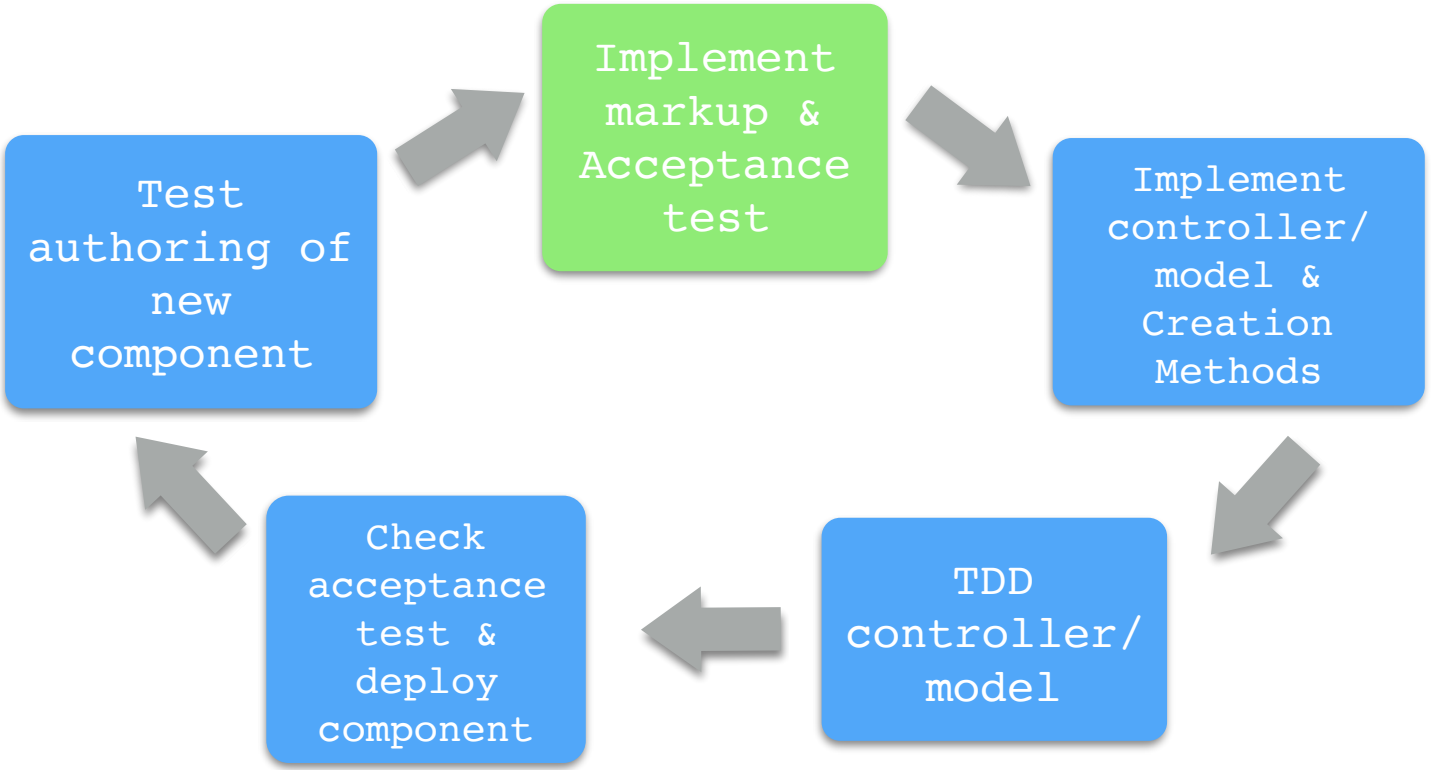
Let's implement a new component

- Write a markup template
- Provide a custom testing API
- [Write an acceptance test, see it fail]*
- Test-drive the component
- [See the acceptance test pass]*
- [Run the component]*
- Sample Code is available on Github
  - https://github.com/quatico-solutions/aem-testing-sample

*omitted here

Demo: Test-drive a new component.

Development Cycle with AEM



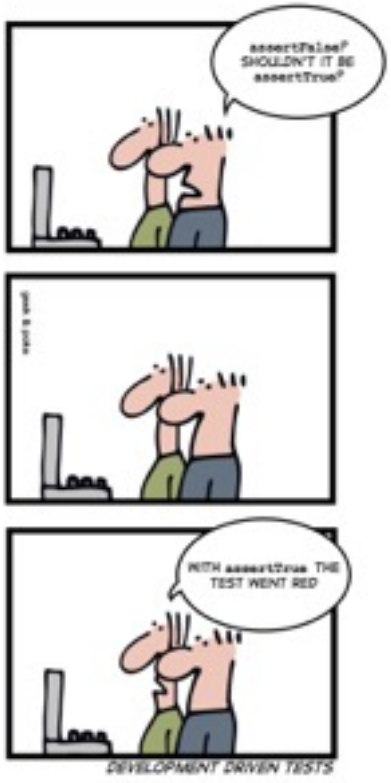
adaptTo()

Conclusion: No excuses

- Consistent creation of AEM resources
- Available across all components / tests
- Test setups cost virtually nothing
- New developers quickly adapt best practices
- You write more simple, reliable, maintainable tests
- Listen to their feedback more frequently

Question & Answers

- Thank you!



Jan Wloka

@crashtester

Appendix.

Unit Testing w/ AEM Mocks

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

@Test
public void getImageLegendWithSourceAndCaptionReturnsBoth() throws Exception {
  context.create().page("/content/foobar", PageType.PRESENCE.getTemplate().getName());
  context.create().page("/content/foobar/ko", PageType.LANGUAGE.getTemplate().getName());
  context.create().page("/content/foobar/ko/home", PageType.HOME.getTemplate().getName());
  Page     page            = context.create().page("/content/foobar/ko/home/content", PageType.DETAIL.getTemplate().getName());
  Resource contentResource = page.getContentResource();

  Resource target = context.create().resource(contentResource.getPath() + "/textimage", new HashMap<>());

  Map<String, Object> imageProps = new HashMap<>();
  imageProps.put("source", "<sourceValue>");
  imageProps.put("caption", "titleValue");
  imageProps.put(JcrResourceConstants.SLING_RESOURCE_TYPE_PROPERTY, ComponentType.IMAGE);
  imageProps.put(JcrConstants.JCR_LASTMODIFIED, new Time().format(DateFormat.GMT));
  imageProps.put(JcrConstants.JCR_LAST_MODIFIED_BY, "admin");
  context.create().resource(contentResource.getPath() + "/textimage/image", imageProps);

  TextImageController testObj = new TextImageController().setup(mockPageContext(target,
                        context.resourceResolver().resolve(page.getPath())));

  assertEquals("titleValue<br />&lt;sourceValue&gt;", testObj.getImageLegend());
}

private PageContext mockPageContext(Resource resource, Resource page) throws Exception { … }
```