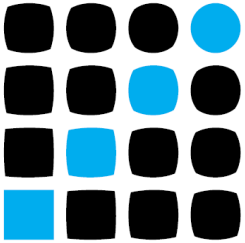


adaptTo()

APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 22-24 SEPTEMBER 2014

Application architecture with Sling Models

Stefan Seifert, pro!vision GmbH

About the Speaker

- CQ5/AEM6 Developer
- Apache Sling Committer
- CTO of pro!vision GmbH

PRO!VISION
SOFTWARE CRAFTSMANSHIP

http://www.pro-vision.de

Overview

- Architecture requirements
- How Sling Models can help
- Real-world Examples
- Summary

Sling Models 1.1 required

Architecture requirements

Architecture Requirements

- Separation of Concerns
- Re-use of common functionality
- Reduce boilerplate code
- Testability
- Multitenancy and Multi-Application

Separation of Concerns

Content	Resources
View	Sightly, JSP, Servlet
Model	Value Map, Java Object
Controller	Java Object
Business Logic	Java Object

Re-use of common functionality

- Examples for common functionality
  - Link resolving
  - Media resolving
  - Navigating resource hierarchy
  - Inheritance
  - Sending HTTP Headers

Reduce boilerplate code

- No boilerplate code for
  - Accessing content
  - Passing over request/resource-context
  - Looking up and injecting dependencies

Testability

- Every Java class should be easily unit-testable on its own
- Use POJOs with Dependency Injection (DI)
- Mock dependencies with tools like Mockito

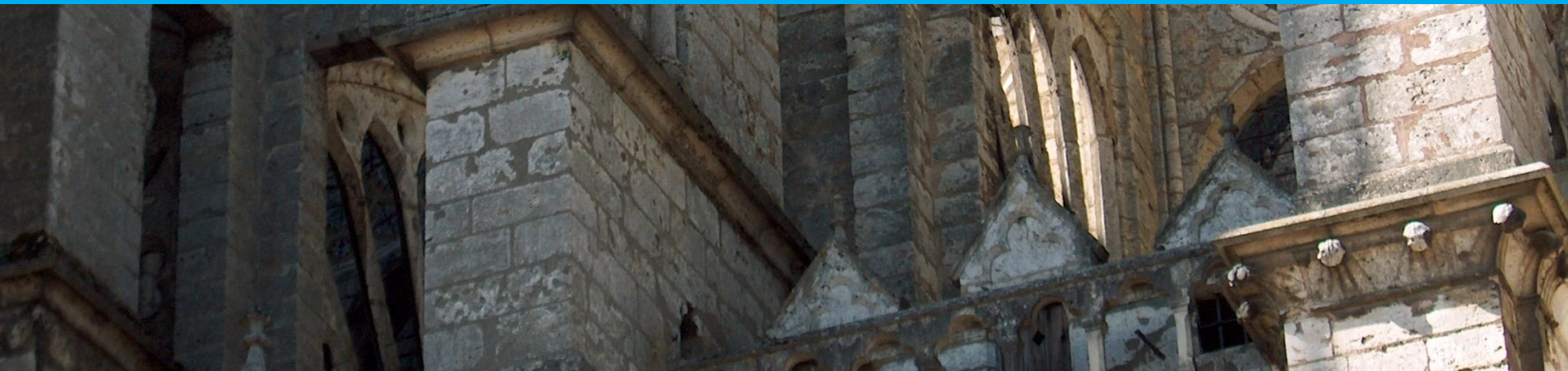
Multitenancy and Multi-Application

- Multiple applications on a single instance
- Multiple tenants
- Different implementations of interfaces
- Isolate applications and tenants

Why not use OSGi services?

- Use OSGi services wherever possible
- Not possible to inject “context objects” depending on current request/resource
- Multitenancy difficult
- Need solution for “context-aware” Dependency Injection

How Sling Models can help



Separation of Concerns

- Use Sling Models for Controllers and Business Classes that are “context-aware”
- Use Value Map to read resource data
  - Use Slings Models to read resource data only when it makes sense

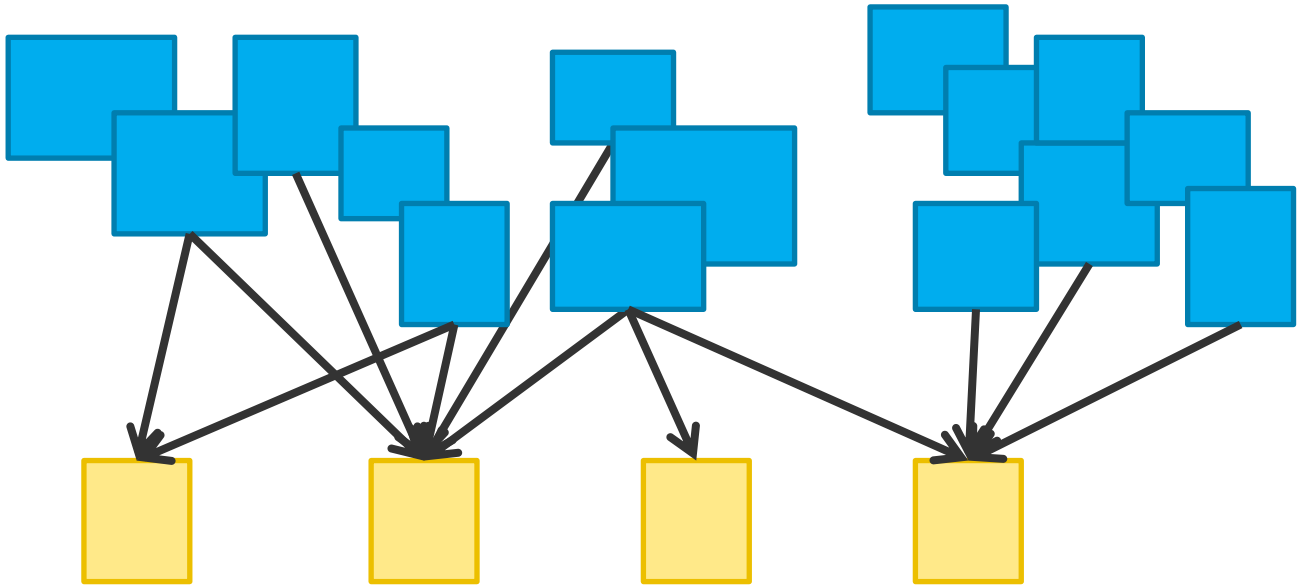
Re-use of common functionality

- Recommended pattern:
  Controller per concern, not per component
- Controllers are Sling Models
- Component/View can (re-)use
  multiple controllers

Re-use of common functionality

Components

Controller per concern



Reduce boilerplate code

- Use Sling Models for Dependency Injection
  - Inject Sling Context Objects via @SlingObject
  - Inject AEM Context Objects via @AemObject
  - Inject Business Classes via @Self
  - Inject Resource Data or Parameters

Testability

- Sling Model Inject Annotations defines the dependency contract
- Dependencies can be mocked with tools like Mockito @InjectMocks

Multitenancy and Multi-Application

- Register models to interfaces using the adapter attribute
- Provide different implementations per application
  - ImplementationPicker chooses the right one
- Provide different configuration per tenant

Real-world examples

Component with Media & Link concern

```
<div data-sly-use.link="o.a.controller.ResourceLink"
    data-sly-use.media="${'o.a.controller.ResourceMedia' @ mediaFormat='content_480'}">

  <figure class="image-small" data-sly-test="${media.valid}">

    <a data-sly-attribute="${link.attributes}" data-sly-unwrap="${!link.valid}">

      <img data-sly-text="${media.markup @ context='html'}" data-sly-unwrap/>

    </a>

  </figure>

</div>
```

Resolve media from current resource

```
@Model(adaptables = SlingHttpServletRequest.class)
public class ResourceMedia {

  @RequestAttribute(optional = true) String mediaFormat;
  @Self MediaHandler mediaHandler;
  @SlingObject Resource resource;

  MediaMetadata media;

  @PostConstruct
  protected void activate() {
    media = mediaHandler.getMediaMetadata(resource, mediaFormat);
  }

  public boolean isValid() { return media.isValid(); }

  public String getMarkup() { return media.getMediaMarkup(); }

}
```

Business class example

```
@Model(adaptables = { SlingHttpServletRequest.class, Resource.class },
    adapters = MediaHandler.class)
public final class MediaHandlerImpl implements MediaHandler {

  @Self
  private Configuration config;
  @SlingObject
  private ResourceResolver resolver;
  @OSGiService
  private SlingSettingsService slingSettings;

  // optional injections (only available if called inside a request)
  @SlingObject(optional = true)
  private SlingHttpServletRequest request;
  @AemObject(optional = true)
  private Page currentPage;

  // ...
}
```

Testability

```
@RunWith(MockitoJUnitRunner.class) public class ResourceMediaTest {
  String mediaFormat = "content_480";
  @Mock Resource resource;
  @Mock MediaHandler mediaHandler;
  @Mock MediaMetadata media;

  @InjectMocks ResourceMedia underTest;

  @Before public void setUp() {
    when(mediaHandler.getMediaMetadata(resource, mediaFormat).thenReturn(media);
    when(media.isValid()).thenReturn(true);
    when(media.getMediaMarkup()).thenReturn("<img/>");
  }

  @Test public void testMedia() {
    underTest.activate();
    assertTrue(underTest.isValid());
    assertEquals("<img/>", underTest.getMarkup());
  }
}
```

Application-specific implementation

```
@Model(adaptables = { SlingHttpServletRequest.class, Resource.class },
    adapters = UrlHandlerConfig.class)
@Application("MyApp")
public class MyAppUrlHandlerConfig implements UrlHandlerConfig {

  @Override
  public UrlMode getDefaultUrlMode() {
    // ...
  }

  @Override
  public boolean isSecure(Page pPage) {
    // ...
  }

}
```

ImplementationPicker Implementation

```
@Component
@Service(ImplementationPicker.class)
@Property(name = Constants.SERVICE_RANKING, intValue = 1000)
public class ApplicationImplementationPicker implements ImplementationPicker {

  public Class pick(Class adapterType, Class[] implTypes, Object adaptable) {
    String applicationId = getApplicationId(adaptable);
    for (Class type : implTypes) {
      Application ann = type.getAnnotation(Application.class);
      if (ann != null && StringUtils.equals(applicationId, ann.value())) {
        return type;
      }
    }
    return null;
  }

}
```

Summary

Summary

- Sling Models not necessary Domain Objects
- Sling Models = context-aware Dependency Injection Framework
- Have Fun!

Resources

- **Sling Models Documentation**
  http://sling.apache.org/documentation/bundles/models.html
- **Sling Models 1.0.x Introduction by Justin Edelson**
  http://slideshare.net/justinedelson/sling-models-overview
- **Sling Models Content Packages**
  https://github.com/Adobe-Consulting-Services/com.adobe.acs.bundles.sling-models/releases
  (Drop-in replacement in existing AEM6 instances, runs in CQ56 as well)
- **AEM Object Injector Implementations**
  http://wcm.io/sling/models/
  http://adobe-consulting-services.github.io/acs-aem-commons/features/aem-sling-models-injectors.html
- **What's new in Sling Models 1.1**
  http://adapt.to/2014/en/schedule/whats-new-in-sling-models-11.html