# Ask the Experts – Using Sling and Adobe Experience Manager

Scott Macdonald | AEM Community Manager

Lokesh BS | Top AEM Community Member



Sagmeister & Walsh

# Agenda

- What is Sling and cheatsheet - http://docs.adobe.com/docs/en/cq/5-6-1/developing/sling_cheatsheet.html (L)
- Sling Selectors within AEM
- Default Sling Post Servlet
- Sling APIs (S)
- Sling Models (L)

# What is Sling

- REST based web framework
- Content-driven, using a JCR content repository
- A resource is a central part of Sling and it assumes everything in the JCR is a resource
- Powered by OSGi
- Scripting inside, multiple languages (JSP, server-side javascript)
- Apache Open Source project

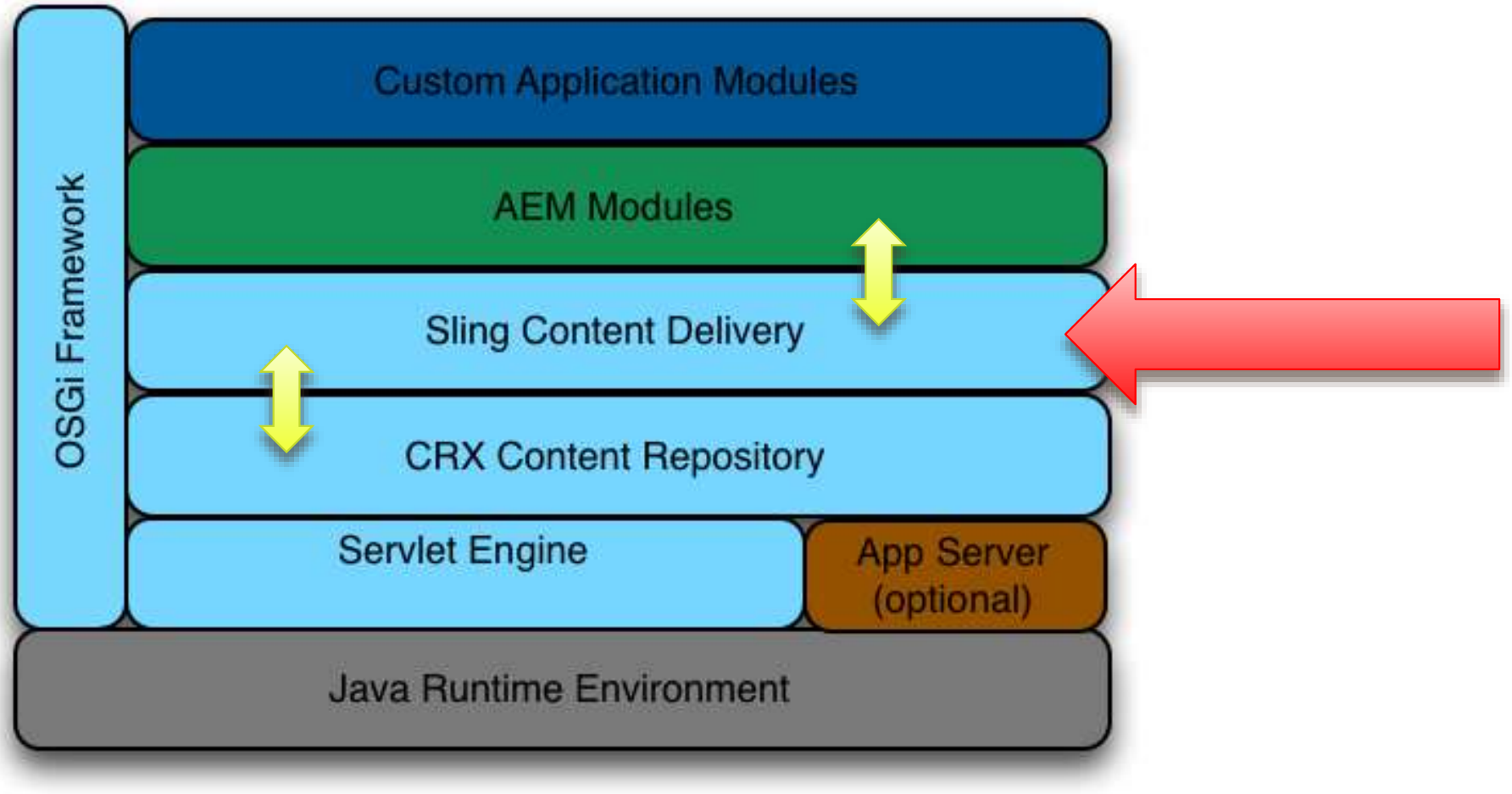
* With some exceptions

# Sling Architecture

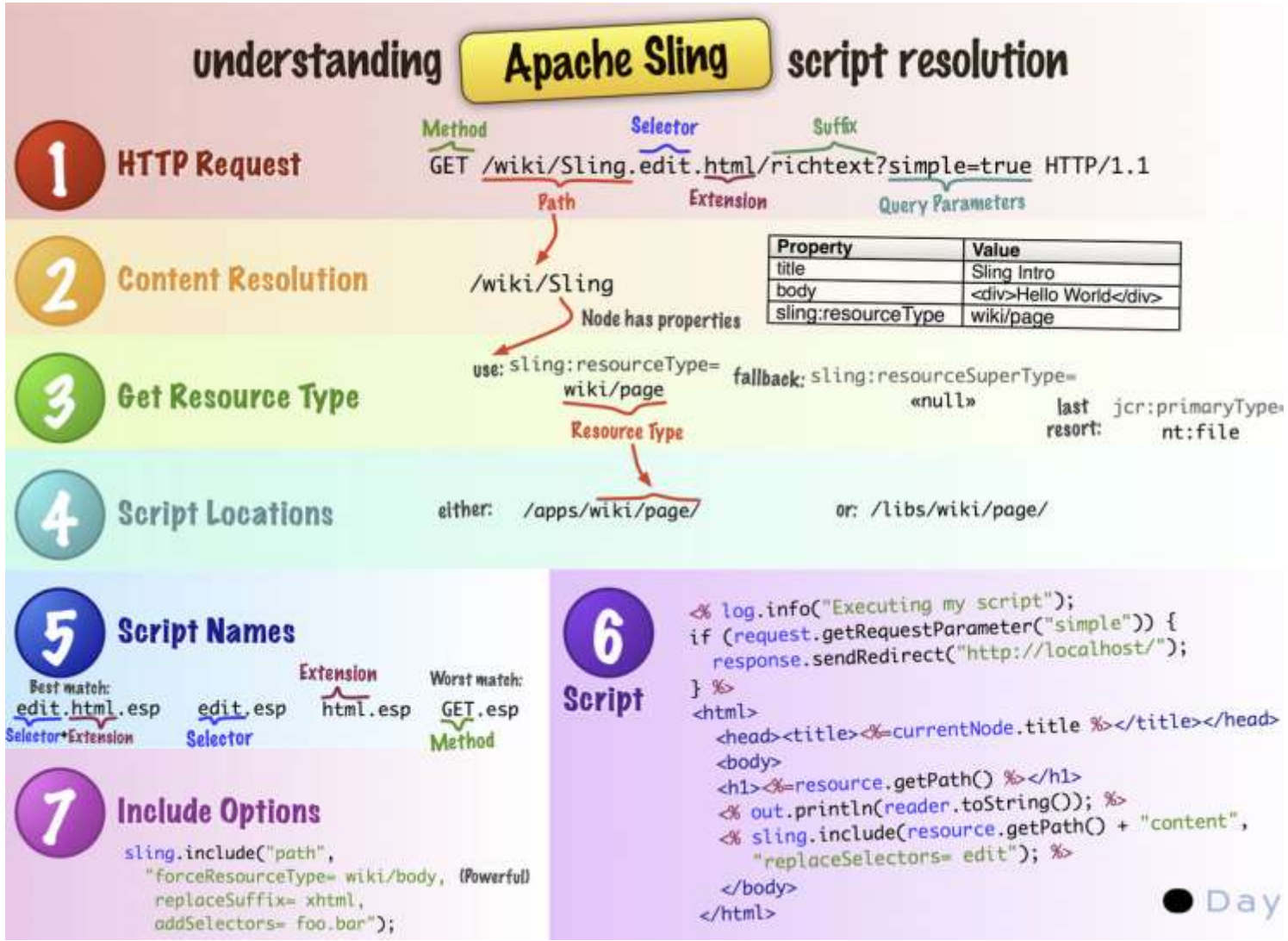
- **OSGi** —Sling applications are built as OSGi bundles and makes heavy use of a number of OSGi core and services.
- **Sling API** — The Sling API extends the Servlet API and provides more functionality to work on the content.
- **Request Processing** — Sling takes a unique approach to handling requests in that a request URL is first resolved to a resource, then based on the resource (and only the resource) it selects the actual servlet or script to handle the request.
- **Resources** —Sling considers everything in the JCR as a Resource.  Based on the resource, a servlet or script is accessed to actually handle the request.
- **Servlets and Scripts** — Servlets and Scripts are handled uniformly in that they are represented as resources themselves and are accessible by a resource path.

* With some exceptions

# AEM and Sling



# Sling CheatSheet








Adobe

# Using the SlingPostServlet

The **`SlingPostServlet`** is a front-end to JCR operations. To select a JCR operation to execute, the **`:operation`** request parameter is used.

Out of the box, the **SlingPostServlet** supports the following operations:

- **property not set or empty** -- Create new content or modify existing content
- **delete** -- Remove existing content
- **move** -- Move existing content to a new location
- **copy** -- Copy existing content to a new location
- **import** -- Import content structures from JSON/XML/Zip
- **nop** -- Explicitly requests to do nothing and just sets the response status
- **checkin** - Check in a versionable node
- **checkout** - Check out a versionable node

# SlingPostServlet Examples

- Create a node named *mynode*. Set (or overwrite) *title* and *body* properties.

```
<form action="/mynode" method="POST">
    <input type="text" name="title">
    <textarea name = "body"
 </form>
```

- Create a node below /mynode and make it the first child

```
<form action="/mynode/" method="POST">
    <input type="text" name="dummy">
    <input type="hidden" name=":order">
       <vaule = "first"
 </form>
```

# SlingPostServlet Examples (Continued)

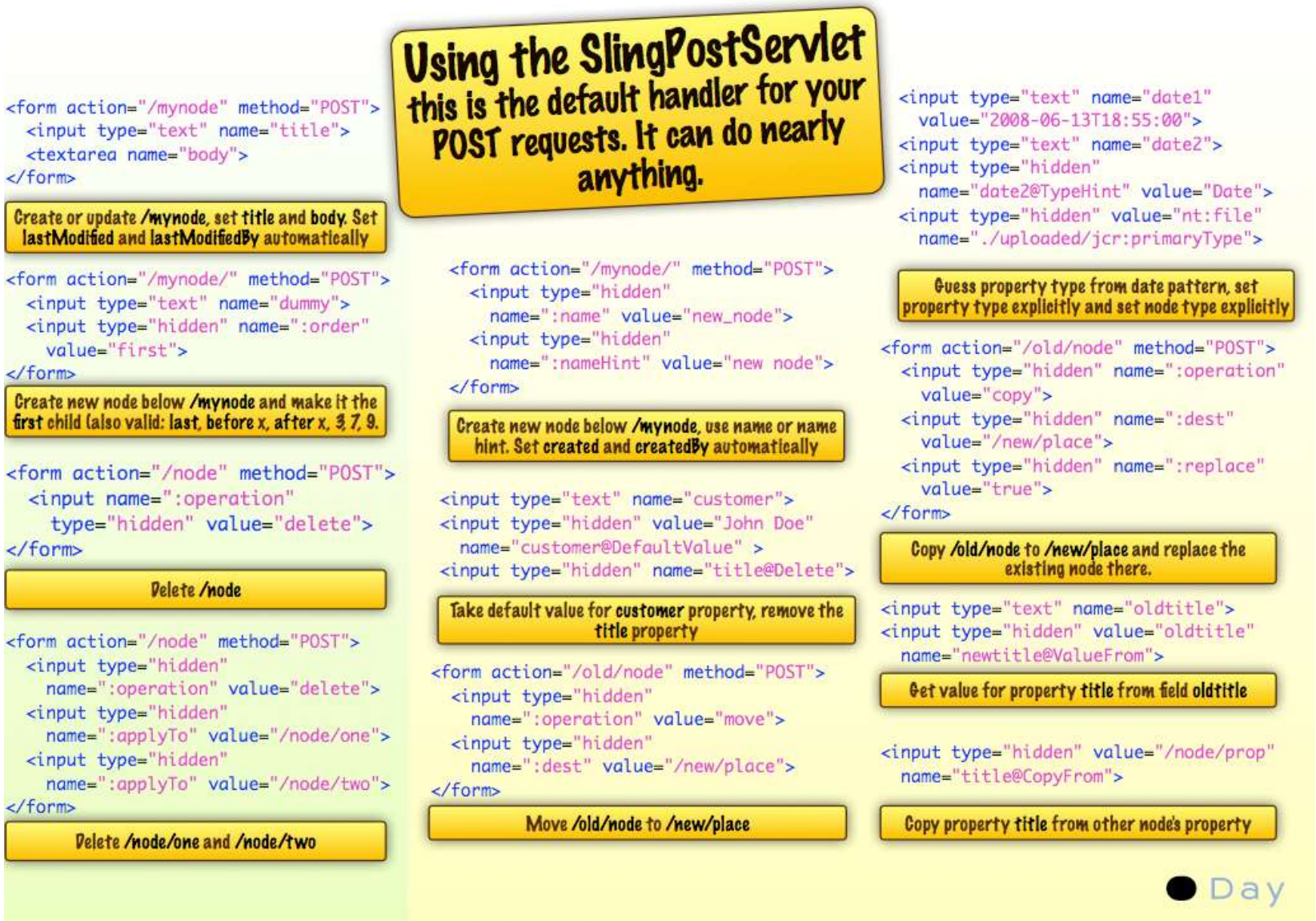
- Delete a node.

```
<form action="/node/" method="POST">
  <input name =":operation" type="hidden">
     <vaule = "delete" >
</form>
```

- Create a node below /mynode and make it the first child

```
<form action="/old/node/" method="POST">
  <input type="hidden“ name=":operation" value="move">
  <input type="hidden" name=":dest" vaule = "/new/place">
</form>
```

# SlingPostServlet Cheatsheet








Adobe

# SlingPostServlet DEMO

# AEM Sling APIs

- The *Sling API* defines a presentation framework to build Web Applications.
- The Sling API is resource centric. That is, the request URL does not address a servlet or a portlet but a resource represented by an instance of the **`org.apache.sling.api.resource.Resource`** interface
- The Sling API uses the URL to select a resource to be delivered. You can retrieve content from the AEM JCR using Sling APIs. You can use the Sling API from within an OSGi bundle to retrieve a resource from within the AEM JCR.
- To use the Sling API from within an OSGi component, you inject an **`org.apache.sling.api.resource.ResourceResolverFactory`** instance into the service
- When using the Sling API to query the AEM JCR, you have access to helper methods that are not available when using the JCR API. For example, the **`adaptTo`** method converts a resource into an appropriate object representing a certain aspect of this resource
- For example to translate a `Resource` object to the corresponding **`Node`** object, you invoke the **`adaptTo`** method:

```
Node node = resource.adaptTo(Node.class);
```

# Sling API “Live” Coding





Adobe

# Sling Models

# Design Goals

- Entirely annotation driven. "Pure" POJOs
- Use of standard annotations
- OOTB supports resource properties, SlingBindings, OSGi services
- Adapt multiple objects
- Support both classes and interfaces.
- Work with existing Sling infrastructure

# First Model

- POJO Class

```
@Model(adaptables=Resource.class)
public class TestModel{
        @Inject
        private String propertyName;
}
```

- Interface

```
@Model(adaptables=Resource.class)
public interface TestModel{
        @Inject
        String getPropertyName();
}
```

- Required

  In Manifest file:

```
<Sling-Model-Packages>
        org.apache.sling.models.mymodels
</Sling-Model-Packages>
```

# Annotations

**@Model**

declares a model class or interface

**@Inject**

marks a field or method as injectable

**@Named**

declare a name for the injection (otherwise, defaults based on field or method name).

**@Optional**

marks a field or method injection as optional

**@Source**

explicitly tie an injected field or method to a particular injector (by name). Can also be on other annotations.

**@Filter**

an OSGi service filter

**@PostConstruct**

methods to call upon model option creation (only for model classes)

**@Via**

use a JavaBean property of the adaptable as the source of the injection

**@Default**

set default values for a field or method

# Client Code

## Model Instantiation

- **adaptTo()**

```
TestModel model = resource.adaptTo(TestModel.class)
```

- **ModelFactory**

```
try {
        TestModel model = modelFactory.createModel(resource,
TestModel.class);
} catch (Exception e) {
        //exception
}
```


Adobe