Sling Models

Sling Models are annotation driven java bean classes which are mapped automatically with Sling Objects *(like Resource, Request….)* and allow us to access JCR node property values directly into java classes via annotations. Sling Models supports both class and interface.

A Sling Model is implemented as an OSGi bundle. A Java class located in the OSGi bundle is annotated with @Model and the adaptable class (for example, *@Model(adaptables = Resource.class)*. The data members (Fields) use @Inject annotations. These data members map to node properties.

In order for these classes to be picked up, there is a header which must be added to the bundle’s manifest: *(add sling models package in core/pom.xml)*

|  |  |
| --- | --- |
| 1  2  3  4 | <**Sling-Model-Packages**>    com.aem.explained.platform.core.models,    com.aem.explained.platform.core.components  </**Sling-Model-Packages**> |

**SLING MODELS ANNOTATIONS:**

| **Annotation** | **Description** |
| --- | --- |
| @Model | declares a model class or interface |
| @Inject | marks a field or method as injectable |
| @Self | Injects the adaptable itself. If the field type does not match with the adaptable it is tried to adapt the adaptable to the requested type |
| @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 | explictly tie an injected field or method to a particular injector (by name). can also be on other annotations. |
| @PostConstruct | methods to call upon model option creation (only for model classes) |
| @Via | change the adaptable as the source of the injection |
| @Default | set default values for a field or method |
| @ScriptVariable | Injects the script variable defined via [Sling Bindings](https://cwiki.apache.org/confluence/display/SLING/Scripting+variables). If name is not set the name is derived from the method/field name |
| @ValueMapValue | Injects a ValueMap value. If via is not set, it will automatically take resource if the adaptable is the SlingHttpServletRequest. If name is not set the name is derived from the method/field name |
| @RequestAttribute | Injects a request attribute by name. If name is not set the name is derived from the method/field name |
| @OSGiService | Injects an OSGi service by type. The filter can be used give an OSGi service filter |
| @SlingObject | Injects commonly used sling objects if the field matches with the class: request, response, resource resolver, current resource, SlingScriptHelper |

**SLING MODELS VS WCMUSEPOJO**

1. WCMUsePojo will need to be extend from WCMUsePojo class, whereas Sling Models can be standalone class with @Model annotation and no keyword
2. With Sling Models, it’s simpler and cleaner to retrieve common objects or property values, instead of writing more line of code to use API
3. You may use Felix annotation @Reference to reference to an available OSGI service, whereas in Sling Models, you will use @Inject or @OSGiService
4. With Sling Models API 1.3, you can serialize the model and export it as a JSON file with Jackson exporter, so your front-end application can leverage the same model. It’s not available for WCMUsePojo.
5. For WCMUsePojo, you will need to overwrite the activate() method, whereas in Sling Models, your init method will be called in the @PostConstruct annotation

***HAPPY LEARNING….!***

Sources:  
1. <https://sling.apache.org/documentation/bundles/models.html>  
2. <https://helpx.adobe.com/experience-manager/using/sling_models.html>  
3. <https://blogs.perficientdigital.com/2017/05/30/how-to-switch-from-wcmusepojo-to-sling-models-in-aem/>

# HTML Template Language

HTML Template Language (also referred to as HTL, formerly known as Sightly) is Adobe Experience Manager’s preferred and recommended server-side template system for HTML. As in all HTML server-side templating systems, a HTL file defines the output sent to the browser by specifying the HTML itself, some basic presentation logic and variables to be evaluated at runtime.

*It is recommended for new AEM projects to use the HTML Template Language, as it offers multiple benefits compared to JSP.*

The HTML Template Language uses an expression language to insert pieces of content into the rendered markup, and HTML5 data attributes to define statements over blocks of markup (like conditions or iterations).

**BLOCKS AND EXPRESSIONS**

Here’s a first example, which could be contained as is in a  **template.html**

|  |  |
| --- | --- |
| 1  2  3 | <**h1** data-sly-test="${properties.jcr:title}">      ${properties.jcr:title}  </**h1**> |

Two different kind of syntaxes can be distinguished:

* [**Block Statements**](https://docs.adobe.com/content/help/en/experience-manager-htl/using/htl/block-statements.html)
  + To conditionally display the **<h1>**element, a [data-sly-test] HTML5 data attribute is used. HTL provides multiple such attributes, which allow to attach behavior to any HTML element, and all are prefixed with data-sly.
* [**Expression Language**](https://docs.adobe.com/content/help/en/experience-manager-htl/using/htl/expression-language.html)
  + HTL expressions are delimited by characters ${ and }. At runtime, these expressions are evaluated and their value is injected into the outgoing HTML stream.

**BLOCK STATEMENTS:**

**data-sly-use:**  
Initializes a helper object (defined in JavaScript or Java) and exposes it through a variable.

|  |
| --- |
| <**div** data-sly-use.nav="org.example.Navigation">${nav.foo}</**div**> |

**data-sly-unwrap:**  
Removes the host element from the generated markup while retaining its content.

|  |
| --- |
| <**div** class="popup" data-sly-unwrap="${isPopup}">content</div |

**data-sly-text**:  
Replaces the content of its host element with the specified text.

|  |
| --- |
| <**p** data-sly-text="${properties.jcr:description}">Lorem ipsum</**p**> |

**data-sly-attribute**:  
Adds attributes to the host element.

|  |
| --- |
| <**div** title="Lorem Ipsum" data-sly-attribute.title="${properties.jcr:title}"></**div**> |

**data-sly-element**:  
Replaces the element name of the host element.

|  |
| --- |
| <**h1** data-sly-element="${titleLevel}">text</**h1**> |

**data-sly-test**:  
Conditionally removes the host element and it’s content. A value of false removes the element; a value of true retains the element.

|  |
| --- |
| <**p** data-sly-test="${isShown}">text</**p**> |

**data-sly-repeat**:  
With data-sly-repeat you can *repeat*an element multiple times based on the list that is specified.

|  |
| --- |
| <**div** data-sly-repeat="${currentPage.listChildren}">${item.name}</**div**> |

**data-sly-list**:  
Repeats the content of the host element for each enumerable property in the provided object.

|  |  |
| --- | --- |
| 1  2  3  4 | <**dl** data-sly-list="${currentPage.listChildren}">      <**dt**>index: ${itemList.index}</**dt**>      <**dd**>value: ${item.title}</**dd**>  </**dl**> |

Advertisements

REPORT THIS AD

The following default variables are available within the scope of the list:  
*item : The current item in the iteration.  
itemList : Object holding the following properties:  
index : zero-based counter ( 0..length-1 ).  
count : one-based counter ( 1..length ).  
first : true if the current item is the first item.  
middle : true if the current item is neither the first nor the last item.  
last : true if the current item is the last item.  
odd : true if index is odd.  
even : true if index is even.*

**data-sly-resource**:  
Includes the result of rendering the indicated resource through the sling resolution and rendering process.

|  |  |
| --- | --- |
| 1  2  3  4 | <**article** data-sly-resource="path/to/resource"></**article**>  <**article** data-sly-resource="${'path/to/resource' @ wcmmode='disabled'}"></**article**>  <**article** data-sly-resource="${'path/to/resource' @ decorationTagName='span',  cssClassName='className'}"></**article**> |

**data-sly-include**:  
Replaces the content of the host element with the markup generated by the indicated HTML template file (HTL, JSP, ESP etc.) when it is processed by its corresponding template engine.

|  |
| --- |
| <**section** data-sly-include="path/to/template.html"></**section**> |

**template & call:**  
data-sly-template: Defines a template. The host element and its content are not output by HTL.  
data-sly-call: Calls a template defined with data-sly-template. The content of the called template (optionally parameterized) replaces the content of the host element of the call.

**HTL GLOBAL OBJECTS:**

Advertisements

REPORT THIS AD

Below are some of available global objects in HTL.

properties, pageProperties, inheritedPageProperties, component, componentContext, currentDesign, currentNode, currentPage, currentSession, log, out, pageManager, reader, request, resolver, resource, response, sling, slyWcmHelperproperties, pageProperties, inheritedPageProperties, component, componentContext, currentDesign, currentNode, currentPage, currentSession, log, out, pageManager, reader, request, resolver, resource, response, sling, wcmmode, xssAPI

# Apache Sling Model

The Apache Sling Model enables injector specific annotations which aggregate the standard annotations for each of the available injector, which are: [Script Bindings](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ScriptVariable), [Value Map](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ValueMapValue), [Resource Path](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ResourcePath), [Child Resources](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ChildResource), [Request Attributes](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#RequestAttribute), [OSGI Services](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#OSGiService), [Self](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#Self), and the [Sling Object](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#SlingObject).

Sure we can invoke injectors by the @inject, followed by the @source annotation (with an injector name) as so, @Inject @Source(“script-bindings”), but invoking such injectors introduces many more lines of code which is tedious and repetitive. Using the @inject annotation freely may cause injector collisions.

**Thankfully Apache’s Sling Model library delivered the injector specific annotations!**  
The injector specific annotations enable us, developers, to write less code, enables stability with injectors to demise injector collisions, and enables better IDE support.

This article will provide examples (used in practice) which will include both ways to invoke injectors in Sling Models, using the @Inject & @Source annotations, and also the Apache Sling Model injector specific annotations approach.

## Available Injectors

1. [@ScriptVariable](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ScriptVariable)
2. [@ValueMapValue](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ValueMapValue)
3. [@ResourcePath](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ResourcePath)
4. [@ChildResource](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#ChildResource)
5. [@RequestAttribute](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#RequestAttribute)
6. [@OSGiService](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#OSGiService)
7. [@Self](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#Self)
8. [@SlingObject](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide#SlingObject)

#### 1. Script Bindings (name=”script-bindings”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) 1000  
**Annotation:** @ScriptVariable  
**Description:** Injects objects via script variable defined from [Sling Bindings](https://cwiki.apache.org/confluence/display/SLING/Scripting+variables#Scriptingvariables-Sling'sscriptingvariablesandscriptlanguages); Lookup objects in the script bindings object by name.

As you can see, the example below indicated that there are many ways to inject within the POJO:

**Without the injector specific annotations:**

* 1. @Inject @Source(“script-bindings”) @Named(“component”)
* 2. @Inject @Source(“script-bindings”)

**With the injector specific annotations:**

* 3. @ScriptVariable(name = “component”)
* 4. @ScriptVariable

**Note:** If the name is not set (using the @Named annotation or name property), then the name is derived from the method/property/variable/field name.

A full list of the scripting variables can be found [here](https://sourcedcode.com/aem-global-objects-for-backend-and-front-end-sightly-htl#mappings).

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 | @Model(adaptables = Resource.class,     defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("script-bindings") @Named("component")     // @Inject @Source("script-bindings")     // @ScriptVariable(name = "component")     @ScriptVariable     private Component component;      // @Inject @Source("script-bindings") @Named("componentContext")     // @Inject @Source("script-bindings")     @ScriptVariable     private ComponentContext componentContext;      // @Inject @Source("script-bindings") @Named("currentDesign")     // @Inject @Source("script-bindings")     @ScriptVariable     private Design currentDesign;      // @Inject @Source("script-bindings") @Named("currentNode")     // @Inject @Source("script-bindings")     @ScriptVariable     private Node currentNode;      // @Inject @Source("script-bindings") @Named("currentPage")     // @Inject @Source("script-bindings")     @ScriptVariable     private Page currentPage;      // @Inject @Source("script-bindings") @Named("currentSession")     // @Inject @Source("script-bindings")     @ScriptVariable     private HttpSession currentSession;      // @Inject @Source("script-bindings") @Named("currentStyle")     // @Inject @Source("script-bindings")     @ScriptVariable     private Style currentStyle;      // @Inject @Source("script-bindings") @Named("designer")     // @Inject @Source("script-bindings")     @ScriptVariable     private Designer designer;      // @Inject @Source("script-bindings") @Named("editContext")     // @Inject @Source("script-bindings")     @ScriptVariable     private EditContext editContext;      // @Inject @Source("script-bindings") @Named("log")     // @Inject @Source("script-bindings")     @ScriptVariable     private Logger log;      // @Inject @Source("script-bindings") @Named("out")     // @Inject @Source("script-bindings")     @ScriptVariable     private PrintWriter out;      // @Inject @Source("script-bindings") @Named("pageManager")     // @Inject @Source("script-bindings")     @ScriptVariable     private PageManager pageManager;      // @Inject @Source("script-bindings") @Named("pageProperties")     // @Inject @Source("script-bindings")     @ScriptVariable     private ValueMap pageProperties;      // @Inject @Source("script-bindings") @Named("reader")     // @Inject @Source("script-bindings")     @ScriptVariable     private BufferedReader reader;      // @Inject @Source("script-bindings") @Named("request")     // @Inject @Source("script-bindings")     @ScriptVariable     private SlingHttpServletRequest request;      // @Inject @Source("script-bindings") @Named("resolver")     // @Inject @Source("script-bindings")     @ScriptVariable     private ResourceResolver resolver;      // @Inject @Source("script-bindings") @Named("resource")     // @Inject @Source("script-bindings")     @ScriptVariable     private Resource resource;      // @Inject @Source("script-bindings") @Named("resourceDesign")     // @Inject @Source("script-bindings")     @ScriptVariable     private Design resourceDesign;      // @Inject @Source("script-bindings") @Named("resourcePage")     // @Inject @Source("script-bindings")     @ScriptVariable     private Page resourcePage;      // @Inject @Source("script-bindings") @Named("response")     // @Inject @Source("script-bindings")     @ScriptVariable     private SlingHttpServletResponse response;      // @Inject @Source("script-bindings") @Named("sling")     // @Inject @Source("script-bindings")     @ScriptVariable     private SlingScriptHelper sling;      // @Inject @Source("script-bindings") @Named("slyWcmHelper")     // @Inject @Source("script-bindings")     @ScriptVariable     private WCMScriptHelper slyWcmHelper;      // @Inject @Source("script-bindings") @Named("wcmmode")     // @Inject @Source("script-bindings")     @ScriptVariable     private SightlyWCMMode wcmmode;      // @Inject @Source("script-bindings") @Named("xssAPI")     // @Inject @Source("script-bindings")     @ScriptVariable     private XSSAPI xssAPI; } |

#### 2. Value Map (name=”valuemap”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) 2000  
**Annotation:** @ValueMapValue  
**Description:** Gets a property from a ValueMap by name; If @Via is not set, it will automatically take resource if the adaptable is the SlingHttpServletRequest. If name is not set the name is derived from the method/field name.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | @Model(adaptables = SlingHttpServletRequest.class,     defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("valuemap") @Named("jcr:title")     @ValueMapValue(name = "jcr:title")     private String titleText;      // @Inject @Source("valuemap")     @ValueMapValue     private String titleDescription; } |

#### 3. Resource Path (name=”resource-path”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) 2500  
**Annotation:** @ResourcePath  
**Description:** Injects one or multiple resources. The resource paths are either given by @Path annotations, the element path or paths of the annotation @ResourcePath or by paths given through a resource property being referenced by either @Named or element name of the annotation @ResourcePath.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | @Model(adaptables = Resource.class,         defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("resource-path") @Path("/content/sourcedcode/en/home")     @ResourcePath(path = "/content/sourcedcode/en/home")     Resource sourcedCodePageResource;      // @Inject @Source("resource-path") @Path("/content/we-retail/language-masters/en")     @ResourcePath(name = "/content/we-retail/language-masters/en")     Resource weRetailPageResource;      // @Inject @Source("resource-path") @Path(paths = {"/content/sourcedcode/en/home","/content/we-retail/language-masters/en"})     @ResourcePath(paths = {"/content/sourcedcode/en/home","/content/we-retail/language-masters/en"})     Resource[]  resources; } |

#### 4. Child Resources (name=”child-resources”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) 3000  
**Annotation:** @ChildResource  
**Description:** Gets a child resource by name.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 | @Model(adaptables = Resource.class,         defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("child-resources") @Named("links")     // @ChildResource(name="links")     @ChildResource     private Resource links;      // @Inject @Source("child-resources") @Named("links")     // @ChildResource(name="links")     @ChildResource     private List<Resource> links;      // @Inject @Source("child-resources") @Named("social")     // @ChildResource(name="social")     @ChildResource     private Resource social; } |

#### 5. Request Attributes (name=”request-attributes”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) 4000  
**Annotation:** @RequestAttribute  
**Description:** Injects a request attribute by name. If name is not set the name is derived from the method/field name.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | @Model(adaptables = Resource.class,         defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("request-attributes") @Named("social")     @RequestAttribute(name = "social")     private String socialParam;      // @Inject @Source("request-attributes") @Named("social")     // @RequestAttribute(name = "social")     @RequestAttribute     private String link;      public String getSocialParam() {         return socialParam;     }         public String getLink() {         return socialParam;     } } |

The example below calls the Sling Model using input parameter:

|  |  |
| --- | --- |
| 1 2 3 | <div data-sly-use.exampleComponent="${'com.sourcedcode.core.models.ExampleComponent' @ social='facebook'}">     ${exampleComponent.socialParam} </div> |

A really great example of the use of this annotation can be found in this [article](https://sourcedcode.com/blog/aem/how-to-pass-parameters-to-sling-modal-from-sightly-htl-component).

#### 6. OSGi Services (name=”osgi-services”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) 5000  
**Annotation:** @OSGiService  
**Description:** Injects an OSGi service by type; Lookup services based on class name. Since Sling Models Impl 1.2.8 (SLING-5664) the service with the highest service ranking is returned. In case multiple services are returned, they are ordered descending by their service ranking (i.e. the one with the highest ranking first).

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | @Model(adaptables = Resource.class,         defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("osgi-services")     @OSGIService     private SlingSettingsService slingSettingsService;      // @Inject @Source("osgi-services")     @OSGiService     private MyCustomOSGIService myCustomOSGIService;      // @Inject @Source("osgi-services")     @OSGiService     private MyCustomOSGISConfigurationervice myCustomOSGISConfigurationervice; } |

#### 7. Self (name=”self”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) Integer.MAX\_VALUE  
**Annotation:** @Self  
**Description:** Injects the adaptable object itself (if the class of the field matches or is a supertype). If the @Self annotation is present it is tried to adapt the adaptable to the field type.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 | @Model(adaptables = Resource.class,         defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("self")     @Self     private Node node;      // @Inject @Source("self")     @Self     private MyCustomSlingModel myCustomSlingModel; }  /////// /////// /////// Example below highlights that the @self annotation can minimize the lines of code that needs to be written. /////// /////// @Model(adaptables = Resource.class,         defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      @SlingObject     private Resource currentResource;      Node node;      @PostConstruct     public void init() {         // adapts the current resource to a node class         node = currentResource.adaptTo(Node.class);     } } |

#### 8. Sling Object (name=”sling-object”) Injector

[**Service Ranking:**](https://sourcedcode.com/aem-sling-models-injectors-service-ranking) Integer.MAX\_VALUE  
**Annotation:** @SlingObject  
**Description:** Injects commonly used sling objects if the field matches with the class: request, response, resource resolver, current resource, SlingScriptHelper. This works only if the adaptable can get the according information, i.e. all objects are available via SlingHttpServletRequest while ResourceResolver can only resolve the ResourceResolver object and nothing else. A discussion around this limitation can be found at SLING-4083. Also Resources can only be injected if the according injector specific annotation is used (@SlingObject).

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | @Model(adaptables = Resource.class,         defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL) public class ExampleComponent {      // @Inject @Source("sling-object")     @SlingObject     private SlingHttpServletRequest slingHttpServletRequest;      // @Inject @Source("sling-object")     @SlingObject     private SlingHttpServletResponse slingHttpServletResponse;      // @Inject @Source("sling-object")     @SlingObject     private Resource currentResource;      // @Inject @Source("sling-object")     @SlingObject     private ResourceResolver resourceResolver;  } |

**Note:**  
If the name is not set (using the @Named annotation or name property), then the name is derived from the method/property/variable/field name. An example for setting the @Named annotation would time you as a developer encounter a clash between the method/property/variable/field name or when developer not wanting to use the scripting variable names as the variables in the POJO.

As you can see, using the Apache Sling Model’s injector specific annotations during implementation will help you stay organised, write less code, and speed up the development process.

### Was this post helpful?

Let us know if you liked the post. That’s the only way we can improve.

Yes



##### [Briankasingli](https://sourcedcode.com/blog/author/sourcedc)

Hello, I am an Adobe AEM MVP & a certified Lead AEM Developer who is currently working as a Senior AEM Full Stack Developer at Whitbread, UK, London. I have 10 years of overall web engineering experience and 6 years of AEM experience in practice. I hope to give back to the AEM Full Stack Development community by sharing my knowledge with the world.

Website <https://www.linkedin.com/in/briankasingli>

## Post navigation

[AEM Sling Models Injectors Service Ranking](https://sourcedcode.com/blog/aem/aem-sling-models-injectors-service-ranking)

[AEM Sling Model Field Injection vs Constructor Injection Memory Consumption](https://sourcedcode.com/blog/aem/aem-sling-model-field-injection-vs-constructor-injection-memory-consumption)

### Leave a Reply

Top of Form

Your email address will not be published. Required fields are marked \*

Comment 

Name \* 

Email \*

Website

 Save my name, email, and website in this browser for the next time I comment.

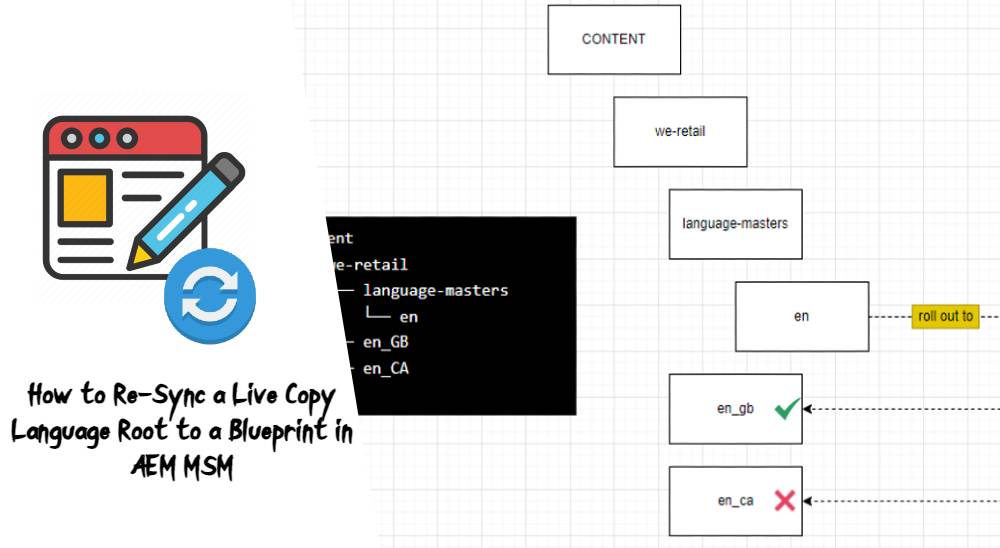
Notify me via e-mail if anyone answers my comment.

I consent to Sourced Code collecting and storing the data I submit in this form. [(Privacy Policy)](https://sourcedcode.com/privacy-policy) \*



Bottom of Form

## RELATED POSTS



[AEM](https://sourcedcode.com/blog/category/aem)

[November 19, 2021](https://sourcedcode.com/blog/aem/how-to-re-sync-a-live-copy-language-root-to-a-blueprint-in-aem-msm)[briankasingli](https://sourcedcode.com/blog/author/sourcedc)

## [How to Re-Sync a Live Copy Language Root to a Blueprint in AEM MSM](https://sourcedcode.com/blog/aem/how-to-re-sync-a-live-copy-language-root-to-a-blueprint-in-aem-msm)



[AEM](https://sourcedcode.com/blog/category/aem) [Development](https://sourcedcode.com/blog/category/aem/development) [Touch UI](https://sourcedcode.com/blog/category/aem/touch-ui)

[October 29, 2021](https://sourcedcode.com/blog/aem/aem-richtext-remove-p-tag-removesingleparagraphcontainer-for-touch-ui)[briankasingli](https://sourcedcode.com/blog/author/sourcedc)

## [AEM RichText Remove P Tag, removeSingleParagraphContainer, for Touch UI](https://sourcedcode.com/blog/aem/aem-richtext-remove-p-tag-removesingleparagraphcontainer-for-touch-ui)



[AEM](https://sourcedcode.com/blog/category/aem) [Development](https://sourcedcode.com/blog/category/aem/development)

[October 24, 2021](https://sourcedcode.com/blog/aem/how-to-pass-parameters-to-wcmusepojo-from-sightly-htl-component)[briankasingli](https://sourcedcode.com/blog/author/sourcedc)

## [How to Pass Data Parameters to WCMUsePojo from Sightly HTL Component](https://sourcedcode.com/blog/aem/how-to-pass-parameters-to-wcmusepojo-from-sightly-htl-component)

#### GET ME HOOKED ON AEM!

Sign up for emails to get the latest AEM tutorials, news, updates & more.

Top of Form

First Name Email AddressSubmit

Bottom of Form

* [**Popular**](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide)
* [**Recent**](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide)
* [**AEM Sling Model Injectors Annotations Reference Guide**](https://sourcedcode.com/blog/aem/aem-sling-model-injectors-annotations-reference-guide)

January 4, 2020

* [**How to use AEM JCR SQL2 query strings to query for nodes in Java Content...**](https://sourcedcode.com/blog/aem/aem-jcr-sql2-tutorial-and-examples-and-cheatsheet)

April 2, 2018

* [**How to make a simple HTTP POST request to AEM with a HTTP Rest Client,...**](https://sourcedcode.com/blog/aem/how-to-make-simple-http-post-methods-in-aem-with-a-http-rest-client)

March 31, 2018

* [**JUnit 4: AEM Sling Models Unit Test Example Using wcm.io AEM Mocks**](https://sourcedcode.com/blog/aem/aem-sling-models-unit-test-junit-4-with-examples)

May 27, 2019

* [**AEM Global Objects for Backend and Front-end Sightly (HTL) Development**](https://sourcedcode.com/blog/aem/aem-global-objects-for-backend-and-front-end-sightly-htl)

May 25, 2019

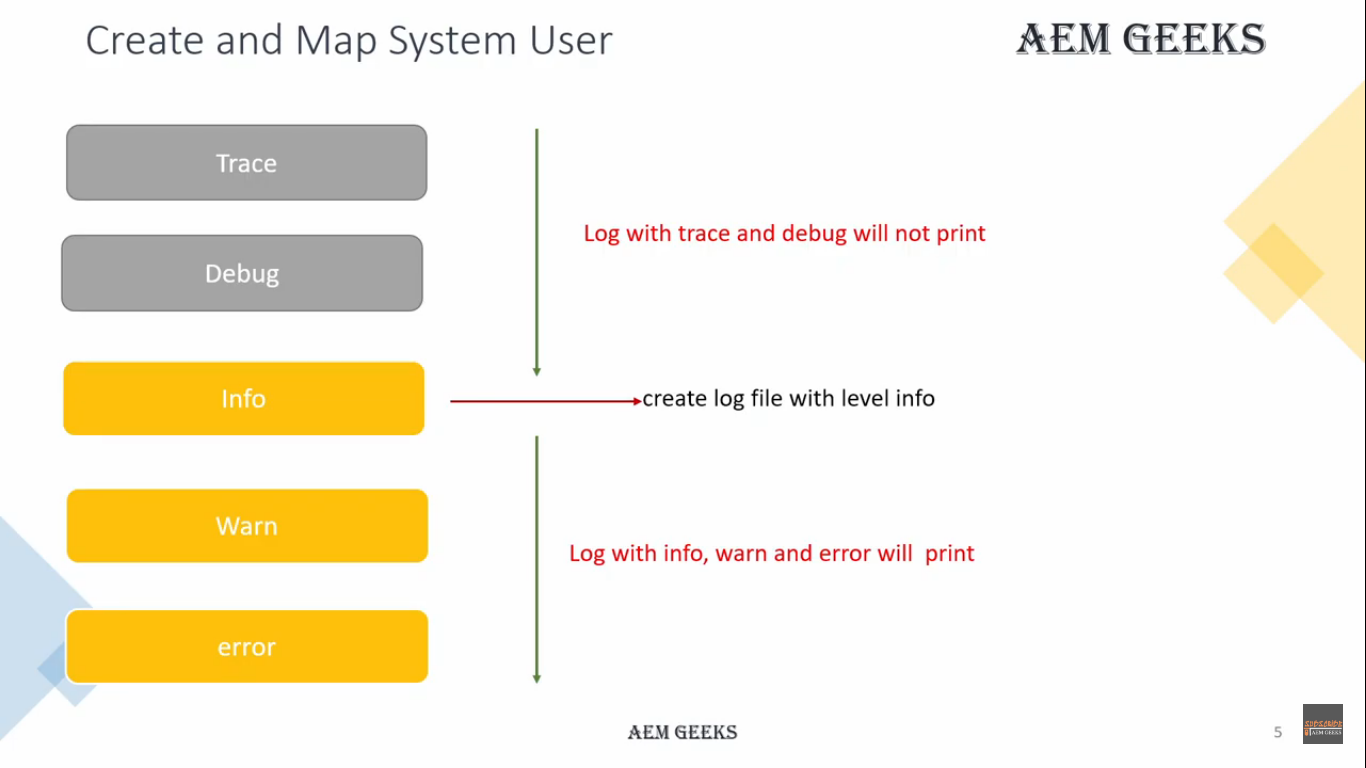
* [**What is the AEM Apache Sling Referrer Filter OSGI configuration?**](https://sourcedcode.com/blog/aem/in-aem-what-is-the-apache-sling-referrer-filter-osgi-configuration)

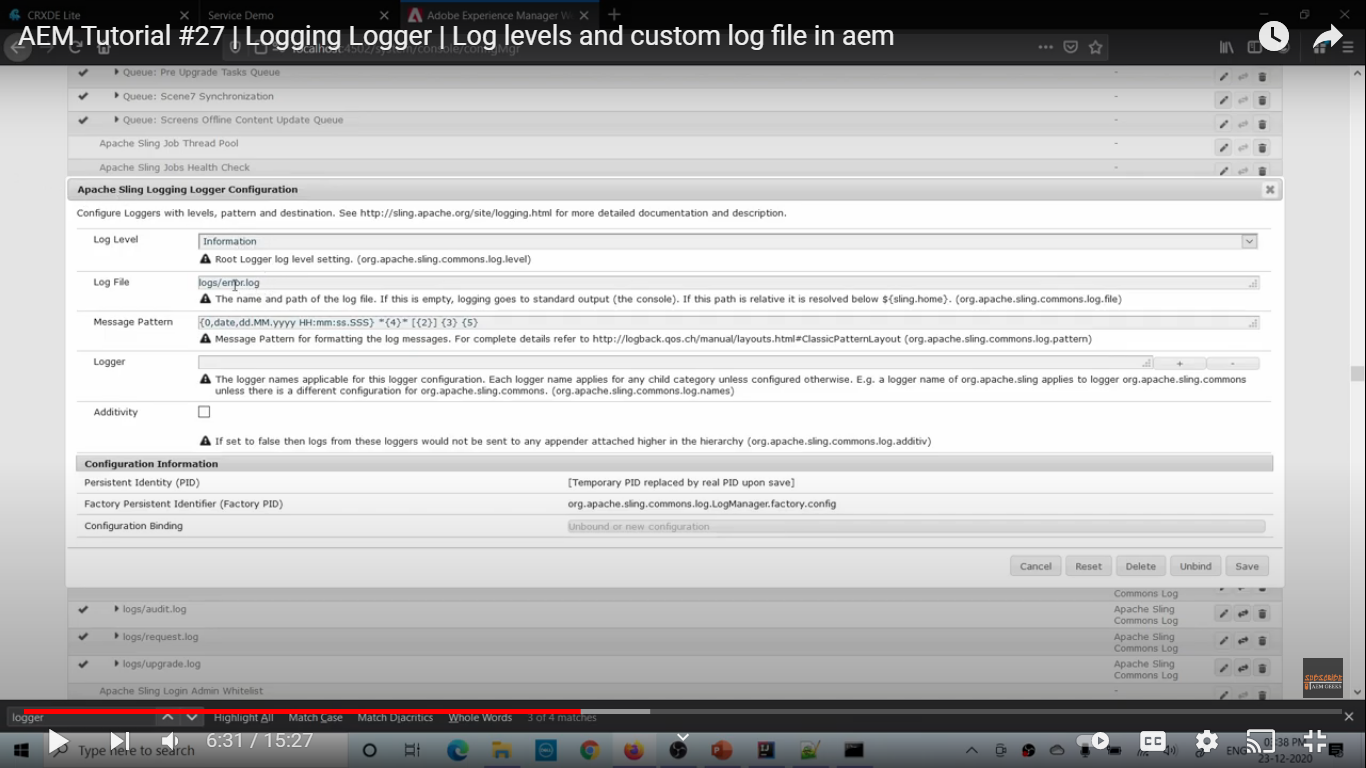
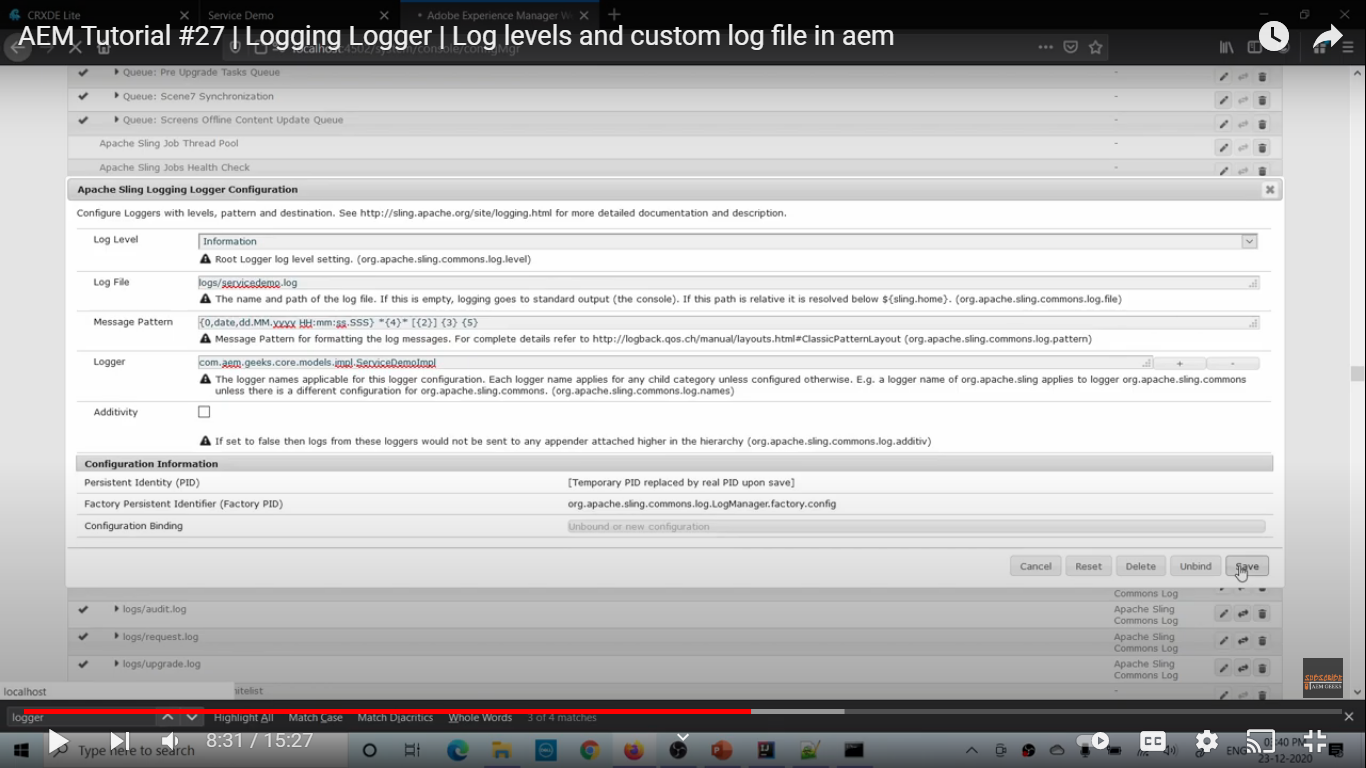
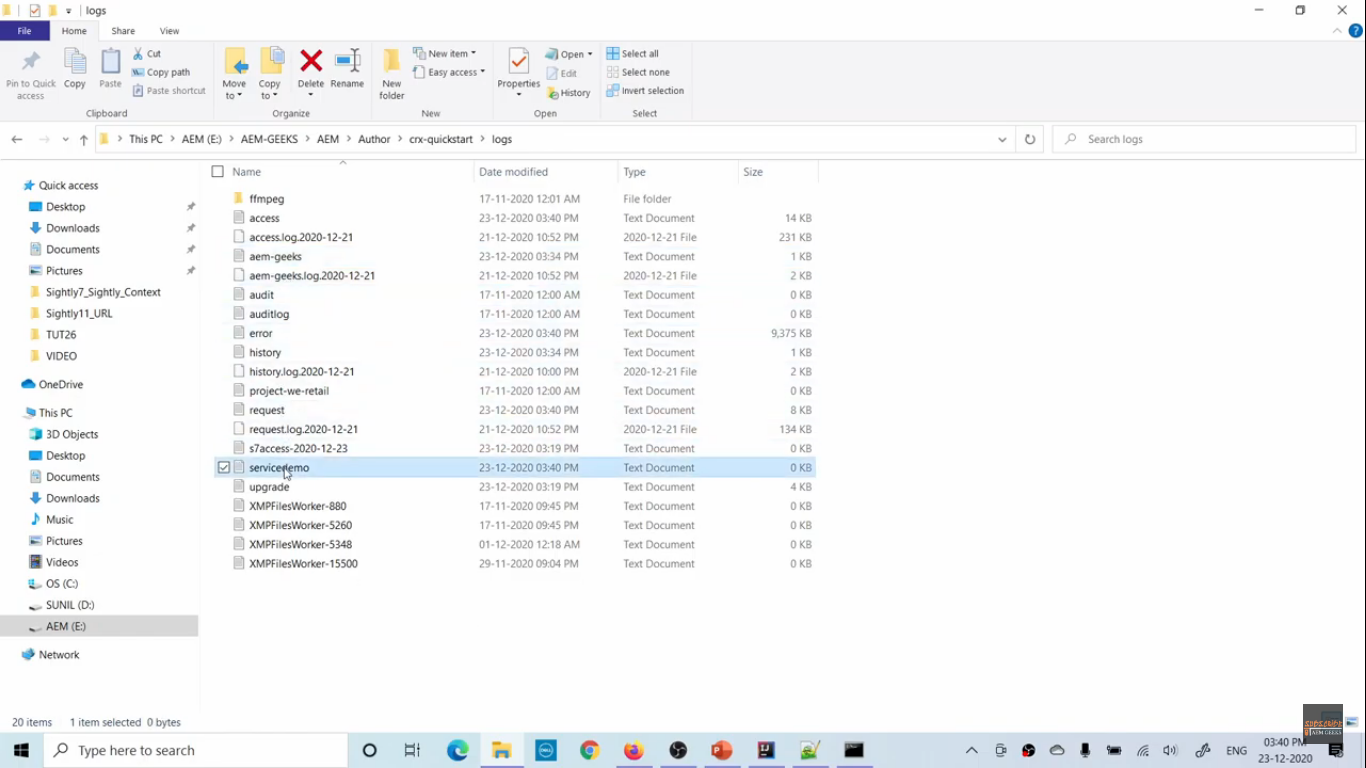
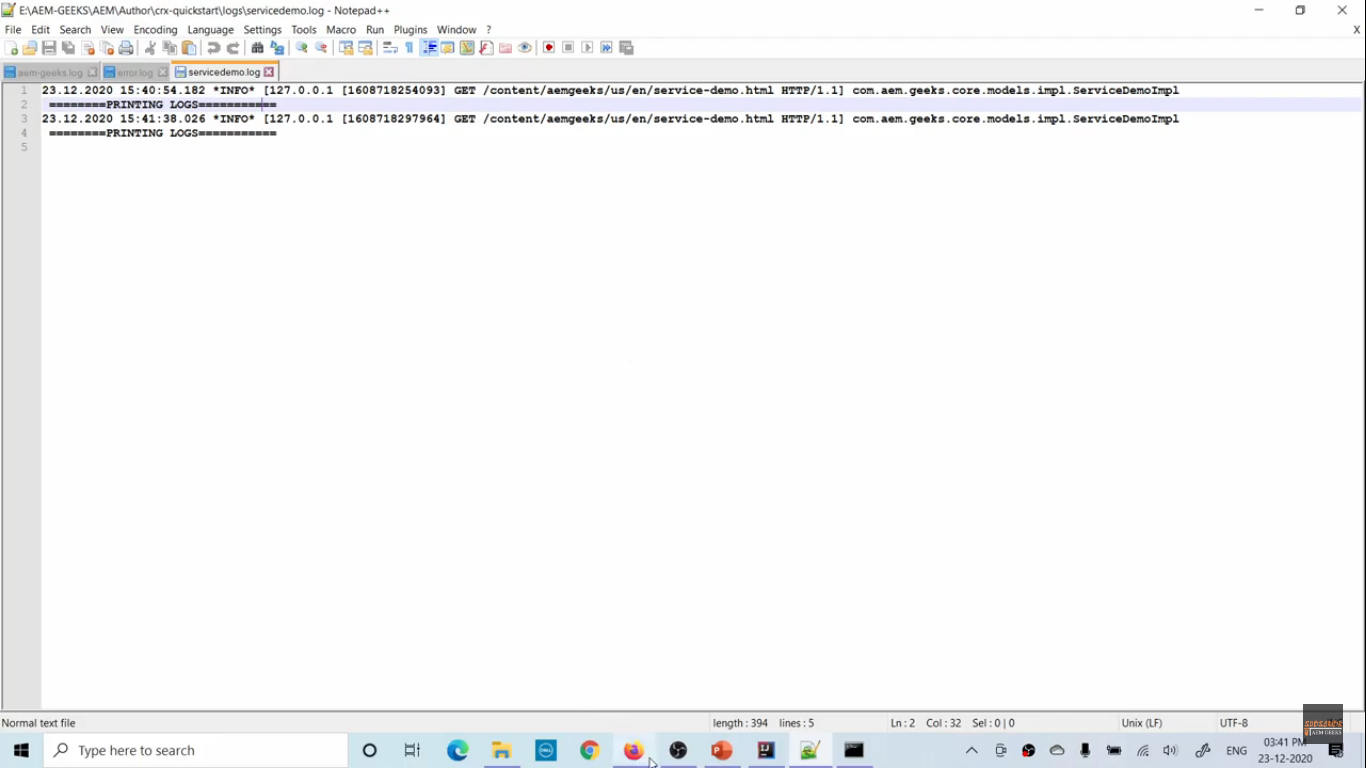
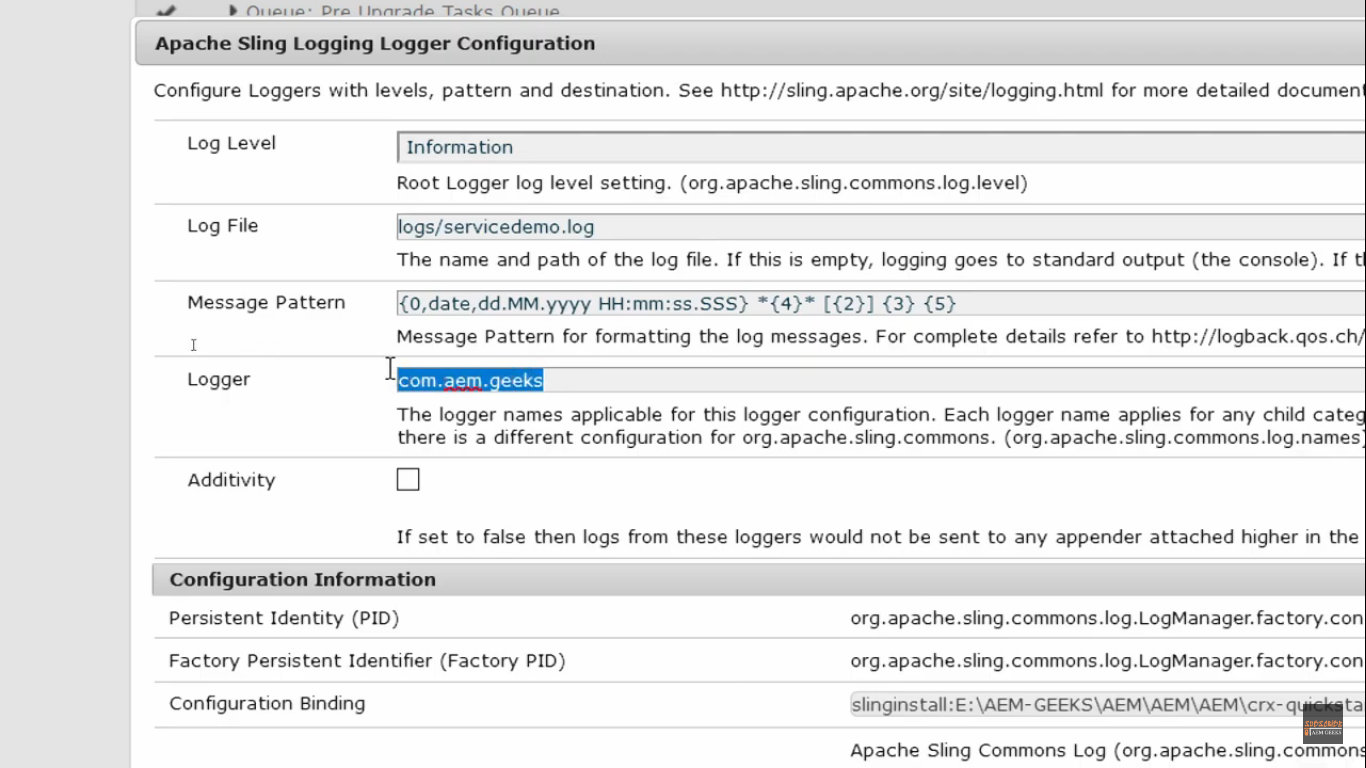
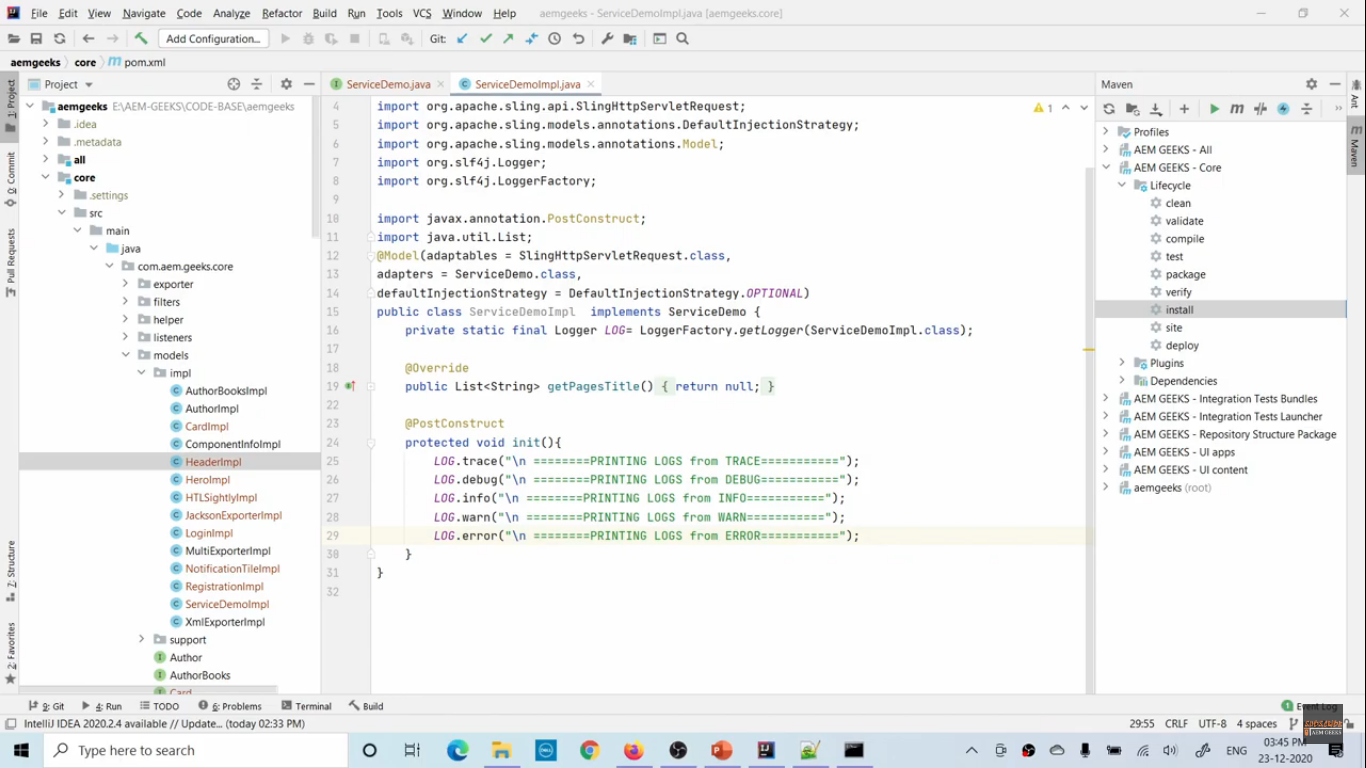
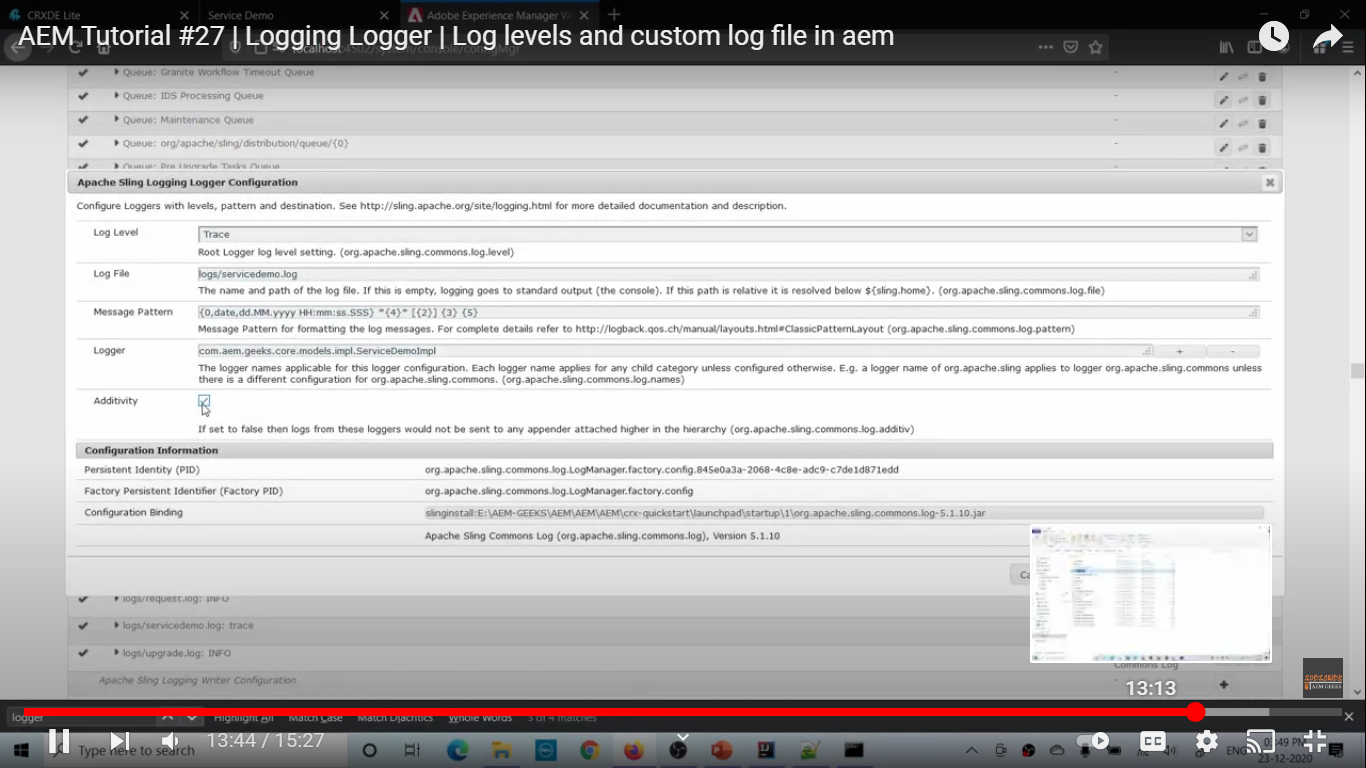
April 1, 2018

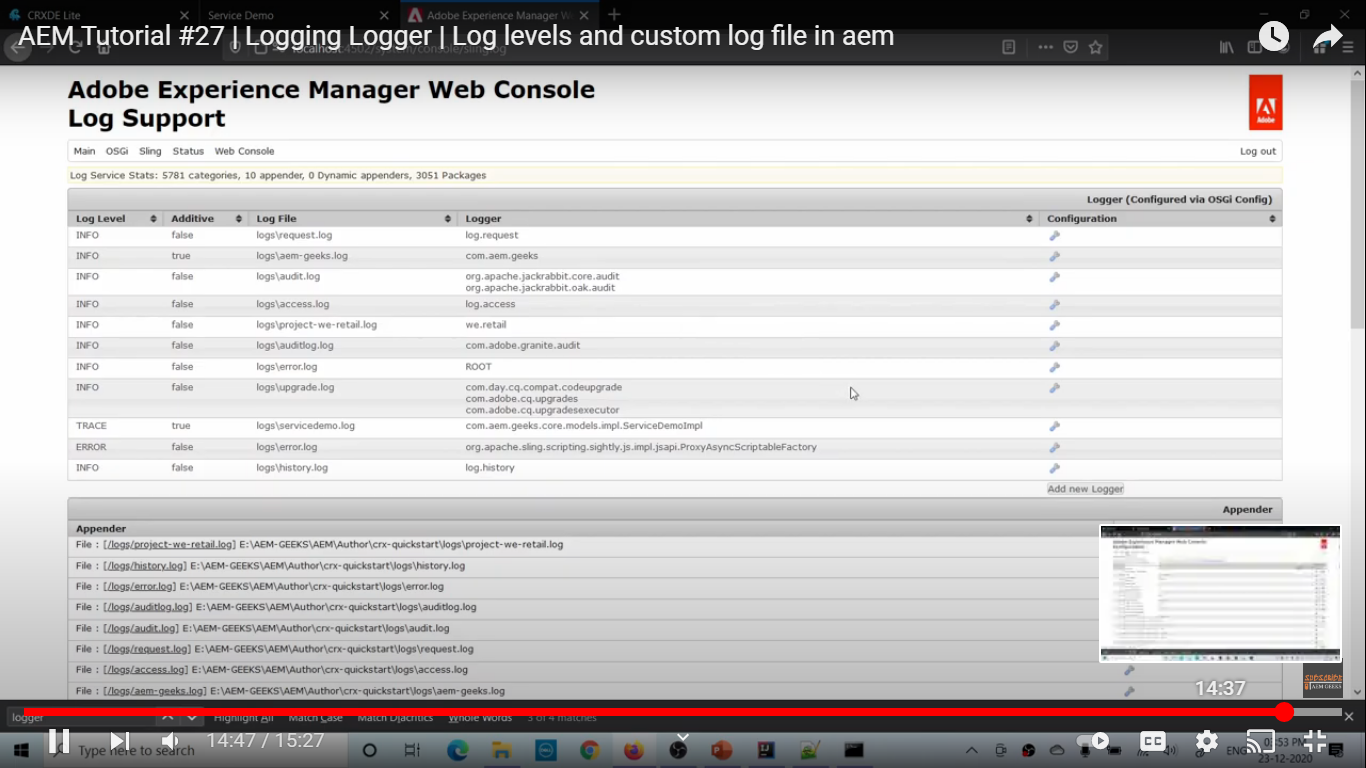
* [**JUnit 4: AEM Sling Servlet Unit Test Example Using wcm.io AEM Mocks, Servlet by Resource...**](https://sourcedcode.com/blog/aem/aem-sling-servlet-osgi-r7-by-resource-type-unit-test-junit-4-with-examples)

December 21, 2019

# Log and Custom Log





# Logging Writer

# OSGI Service

# OSGi service with Rank,Filter and target

# Osgi Configuration R7

# Create OSGi Configuration in separate file in aem

# Create OSGi Factory configuration in aem

# Run Modes | Set installation and Custom run modes in aem

# OSGi Config #4| OSGi configurations with different Run Modes in aem

# Servlet #1| Sling Servlet in aem

# Servlet #2 | Resource Type Sling Servlet in aem

# Servlet #3 | Sling Servlet registered with path in aem

# Servlet #4| Sling Servlet using OSGi R5 annotations in aem

# Scheduler #1 | Scheduler in AEM

# Scheduler #2 | Scheduling multiple jobs using scheduler in aem

# Multifield Dialog | Multifield Dialog with Sling Model in AEM map or bean

# Content as a Service | How to write Sling Model Content Exporter in aem

# Jackson exporter in AEM | Sling Model Exporter using Jackson Exporter in aem

# Sling XML exporter | Custom Sling model exporter using JAXB in aem

# Multiple Sling Exporters | Sling Model with multiple content exporters in aem

# Sling Model Unit Testing | Unit Test for Sling Model using JUnit 5 and AEM Mocks