Robots.txt file:

Robots.txt file advice search engines how to crawl your website. It works in a similar way as the [robots meta tag](https://www.elegantthemes.com/blog/tips-tricks/how-to-stop-search-engines-from-indexing-specific-posts-and-pages-in-wordpress) . The main difference being that the robots.txt file will stop search engines from seeing a page or directory, whereas the robots meta tag only controls whether it is indexed.

Placing a robots.txt file in the [root of your domain](https://support.google.com/webmasters/answer/40361) lets you stop search engines indexing sensitive files and directories. For example, you could stop a search engine from crawling your images folder or from indexing a PDF file that is located in a secret folder.

Search engines will look for a robots.txt file at the root of your domain whenever they crawl your website. Please note that a separate robots.txt file will need to be configured for each subdomain and for other protocols such as <https://www.yourwebsite.com>.

The two main directives of the standard are:

User-agent – Defines the search engine that a rule applies to

Disallow – Advises a search engine not to crawl and index a file, page, or directory

An asterisk (\*) can be used as a wildcard with User-agent to refer to all search engines. For example, you could add the following to your website robots.txt file to block search engines from crawling your whole website.

|  |  |
| --- | --- |
| User-agent: \* | |
| Disallow: / |

Some websites use the disallow directive without a forward slash to state that a website can be crawled. This allows search engines [complete access to your website](http://www.robotstxt.org/robotstxt.html).

The following code states that all search engines can crawl your website. There is no reason to enter this code on its own in a robots.txt file, as search engines will crawl your website even if you do not define add this code to your robots.txt file. However, it can be used at the end of a robots.txt file to refer to all other user agents.

|  |  |
| --- | --- |
| User-agent: \* | |
| Disallow: |

You can see in the example below that I have specified the images folder using /images/ and not www.yourwebsite.com/images/. This is because robots.txt uses relative paths, not absolute URL paths. The forward slash (/) refers to the root of a domain and therefore applies rules to your whole website. Paths are case sensitive, so be sure to use the correct case when defining files, pages, and directories.

|  |
| --- |
| User-agent: \* |
| Disallow: /images/ | |

In order to define directives for specific search engines, you need to know the name of the search engine spider (aka the user agent). Googlebot-Image, for example, will define rules for the Google Images spider.

|  |  |
| --- | --- |
| User-agent: Googlebot-Image | |
| Disallow: /images/ |

Below are some common search engine spiders that you may want to use:

Bingbot – Bing

Googlebot – Google

Googlebot-Image – Google Images

Googlebot-News – Google News

Teoma – Ask

Non Standard Robots.txt Rules

User-agent and Disallow are supported by all crawlers, though a few more directives are available. These are known as non-standard as they are not supported by all crawlers. However, in practice, most major search engines support these directives too.

Allow – Advises a search engine that it can index a file or directory

Sitemap – Defines the location of your website sitemap

Crawl-delay – Defines the number of seconds between requests to your server

Host – Advises the search engine of your preferred domain if you are using mirrors

It is not necessary to use the allow directive to advise a search engine to crawl your website, as it will do that by default. However, the rule is useful in certain situations. For example, you can define a directive that blocks all search engines from crawling your website, but allow a specific search engine, such as Bing, to crawl. You could also use the directive to allow crawling of a particular file or directory; even if the rest of your website is blocked.

|  |  |
| --- | --- |
| User-agent: Googlebot-Image | |
| Disallow: /images/ |
| Allow: /images/background-images/ | | | |
| Allow: /images/logo.png | | |

Please note that this code:

|  |  |
| --- | --- |
| User-agent: \* | |
| Allow: / |

Produces the same outcome as this code:

|  |  |
| --- | --- |
| User-agent: \* | |
| Disallow: |

A sitemap can be placed anywhere in your sitemap. Generally, website owners list their sitemap at the beginning or near the end of the robots.txt file.

|  |  |  |
| --- | --- | --- |
| Sitemap: http://www.yourwebsite.com/sitemap\_index.xml | | |
| Sitemap: http://www.yourwebsite.com/category-sitemap.xml | | | | |
| Sitemap: http://www.yourwebsite.com/page-sitemap.xml | |
| Sitemap: http://www.yourwebsite.com/post-sitemap.xml | |
| Sitemap: http://www.yourwebsite.com/forum-sitemap.xml |
| Sitemap: http://www.yourwebsite.com/topic-sitemap.xml |
| Sitemap: http://www.yourwebsite.com/post\_tag-sitemap.xml | | | |

Advanced Robots.txt Techniques

The larger search engines, such as Google and Bing, support the use of wildcards in robots.txt. These are very useful for denoting files of the same type.

An asterisk (\*) can be used to match occurrences of a sequence. For example, the following code will blog a range of images that have logo at the beginning.

|  |
| --- |
| User-agent: \* |
| Disallow: /images/logo\*.jpg | |

The code above would disallow images within the images folder such as logo.jpg, logo1.jpg, logo2.jpg. logonew.jpg, and logo-old.jpg.

Be aware that the asterisk will do nothing if it is placed at the end of a rule. For example, *Disallow: about.html\** is the same as *Disallow: about.html*. You could, however, use the code below to block content in any directory that starts with the word test. This would hide directories named test, testsite, test-123 etc.

|  |
| --- |
| User-agent: \* |
| Disallow: /test\*/ | |

Wildcards are useful for stopping search engines from crawling files of a particular type or pages that have a specific prefix.

For example, to stop search engines from crawling all of your PDF documents within your downloads folder, you could use this code:

|  |
| --- |
| User-agent: \* |
| Disallow: /downloads/\*.pdf | |

And you could stop search engines from crawling your wp-admin, wp-includes, and wp-content directories, by using this code:

|  |
| --- |
| User-agent: \* |
| Disallow: /wp-\*/ | |

You can also use wildcards to refer to part of the URL that contains a certain character or series of characters. For example, you can block any URL that contains a questions mark (?) by using this code:

|  |
| --- |
| User-agent: \* |
| Disallow: /\*?\* | |

The following command would stop search engines from crawling any URL that begins with a quote:

|  |  |
| --- | --- |
| User-agent: \* | |
| Disallow: /" |

One thing that I have not touched upon until now is that robots.txt uses prefix matching. What this means is that using Disallow: /dir/ would block search engines from a directory named /dir/ and from directories such as /dir/directory2/, /dir/test.html, etc.

This also applies to file names. Consider the following command for robots.txt:

|  |
| --- |
| User-agent: \* |
| Disallow: /page.php | |

As you know, the above code would stop search engines from crawling page.php. However, it would also stop search engines from crawling /page.php?id=25 and /page.php?id=2&ref=google. In short, robots.txt will block any extension to the URL you block. So blocking www.yourwebsite.com/123 will also block www.yourwebsite.com/123456 and www.yourwebsite.com/123abc.

In many cases, this is the desired effect; however it is sometimes better to specify the end of a path so that no other URL’s are affected. To do this, you can use the dollar sign ($) wildcard. It is frequently used when a website owner wants to block a particular type of file type.

In my previous example of blocking page.php, we can ensure that only page.php is blocked by adding the $ wildcard at the end of the rule.

|  |
| --- |
| User-agent: \* |
| Disallow: /page.php$ | |

And we can use it to ensure that only the /dir/ directory is blocked, not /dir/directory2/ or /dir/test.html.

|  |
| --- |
| User-agent: \* |
| Disallow: /dir/$ | |

Commenting Your Robots.txt Code

It is in your best interest to get into the habit of documenting the code in your robots.txt file. This will help you quickly understand the rules you have added when you refer to it later.

You can publish comments in your robots.txt file using the hash symbol #:

|  |  |
| --- | --- |
| # Block Google Images from crawling the images folder | |
|  |

|  |  |
| --- | --- |
| User-agent: Googlebot-Image | |
| Disallow: /images/ |

What to Place in Your Robots.txt File

Disallowed and allowed directories and files

The Maximum Size of a Robots.txt File

ccording to an article on [AskApache](http://www.askapache.com/seo/updated-robotstxt-wordpress.html), you should never use more use more than 200 disallow lines in your robots.txt file. Unfortunately, they do not provide any evidence in the article that states why this is the case.

In 2006, some members of Webmaster World reported [seeing a message from Google](http://www.webmasterworld.com/forum30/34740.htm) that the robots.txt file should be no more than 5,000 characters. This would probably work out to be around 200 lines if we assume an average of 25 characters per line; which is probably where AskApache got this figure of 200 disallow lines from

For reference, here is a list of the most common directives that are available to you:

all – No restrictions on indexing or linking

index – Show the page in search results and show a cached link in search results

noindex – Do not show the page in search results and do not show a cached link in search results

follow – Follow links on the page

nofollow – Do not follow links on the page

none – The same as using “noindex, nofollow”

noarchive – Do not show a cached link in search results

nocache – Do not show a cached link in search results

nosnippet – Do not show a snippet for the page in search results

noodp – Do not use the meta data from the [Open Directory Project](http://dmoz.org/) for titles or snippets for this page

noydir – Do not use the meta data from the [Yahoo! Directory](http://dir.yahoo.com/) for titles or snippets for this page

notranslate – Do not offer translation for the page in search results

noimageindex – Do not index images from this page

unavailable\_after: [RFC-850 date/time] – Do not show the page in search results after a date and time specified in the [RFC 850 format](http://www.ietf.org/rfc/rfc0850.txt)

Java Performance Optimization:

## Don't Optimize Before You Know It's Necessary

We should follow java standards and best practices while coding in java. And measure the response time for all API calls by specifying maximum response time, So that you will come to know which part of code is making the application slowdown. Based on the we should make the required changes.

## Use StringBuilder to Concatenate Strings Programmatically

There are lots of different options to concatenate Strings in Java. You can use a simple + or StringBuffer or a[StringBuilder](https://docs.oracle.com/javase/9/docs/api/java/lang/StringBuilder.html).

But out of all String builder provides best performance while concatenating a string in a for loop. But make sure It is not thread safe.

StringBuilder sb = new StringBuilder(“This is a test”);

for (int i=0; i<10; i++) {

sb.append(i);

sb.append(” “);

}

log.info(sb.toString());

you can provide the first element of your String to the constructor method. That will create a new StringBuilder containing the provided String and a capacity for 16 additional characters. When you add more characters to the StringBuilder, your JVM will dynamically increase the size of the StringBuilder.

Strings are immutable, and the result of each String concatenation is stored in a new String object. That requires additional memory and slows down your application, especially if you’re concatenating multiple Strings within a loop.

## Use Primitives Where Possible

Another quick and easy way to avoid any overhead and improve the performance of your application is to use primitive types instead of their wrapper classes. So, it’s better to use an int instead of an Integer, or a double instead of a Double. That allows your [JVM](https://stackify.com/jvm-metrics/) to[store the value in the stack instead of the heap](https://www.javaworld.com/article/2150208/java-language/a-case-for-keeping-primitives-in-java.html) to reduce memory consumption and overall handle it more efficiently.

## Use Apache Commons StringUtils.Replace Instead of String.replace

// replace this

test.replace(“test”, “simple test”);

// with this

StringUtils.replace(test, “test”, “simple test”);

Try to call functions if same code is required in multiple places.

### Java-level Deadlocks

True Java-level deadlocks, while less common, can also greatly affect the performance and stability of your application. This problem is triggered when two or more threads are blocked forever, waiting for each other.

Garbage collection.

Need to identify unused object and release the space holding by them.

**Using optimal functions**

Java has multiple functions to handle algorithmic performance. If you use StringBuilder instead of simple String, you will gain few improvements in performance. However, there are other ways to handle optimization at the code level. Let's look at them below.

* Use StringBuilder instead of the + operator.
* Avoid using the iterator().
* Take maximum benefit of the stack.
* Avoid regular expressions and instead use [Apache Commons Lang](http://commons.apache.org/proper/commons-lang/).
* Stay away from recursion. Recursions are very resource intensive!

Sitemap.

* Which will display all the pages of a site in sitemap.
* To do this create a component and iterate the page under /content and display page title and link to it. Drag and drop this component in the sitemap page.

Bundle context Methods

Bundle context is used to access the services in java or jsp.

* There are a few ways by which you may be able to use your defined Service:
* Use SCR annotations to let SCR inject the service in your component:
* @Reference
* private SlingRepository repository;
* Use Bundle Context to get the service in your Java/Jsp file

|  |  |
| --- | --- |
|  | * BundleContext bundleContext = FrameworkUtil.getBundle(MyClass.class).getBundleContext(); * ServiceReference factoryRef = bundleContext.getServiceReference(ResourceResolverFactory.class.getName()); * ResourceResolverFactory resolverFactory = (ResourceResolverFactory) bundleContext.getService(factoryRef); * /\*Always make sure to use a null check whenever getService is used\*/ |

* Use sling.getService() method

|  |  |
| --- | --- |
| * 1 | * SampleService service=sling.getService(SampleService.class); |

A BundleContext object will be created for a bundle when the bundle is started.

The BundleContext object will be passed to the [BundleActivator.start(BundleContext)](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleActivator.html#start(org.osgi.framework.BundleContext)) method during activation of the context bundle. The same BundleContext object will be passed to the[BundleActivator.stop(BundleContext)](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleActivator.html#stop(org.osgi.framework.BundleContext)) method when the context bundle is stopped.

The BundleContext object is only valid during the execution of its context bundle; that is, during the period from when the context bundle is in the STARTING, STOPPING, and ACTIVE bundle states.

**package** **com.bw.osgi.consumer**;

**import** **javax.swing.Timer**;

**import** **java.awt.event.ActionEvent**;

**import** **java.awt.event.ActionListener**;

**import** **com.bw.osgi.provider.able.HelloWorldService**;

**public** **class** **HelloWorldConsumer** **implements** ActionListener {

**private** **final** HelloWorldService service;

**private** **final** Timer timer;

**public** HelloWorldConsumer(HelloWorldService service) {

**super**();

**this**.service = service;

timer = **new** Timer(1000, **this**);

}

**public** void startTimer(){

timer.start();

}

**public** void stopTimer() {

timer.stop();

}

@Override

**public** void actionPerformed(ActionEvent e) {

service.hello();

}

}

**package** **com.bw.osgi.consumer**;

**import** **org.osgi.framework.BundleActivator**;

**import** **org.osgi.framework.BundleContext**;

**import** **org.osgi.framework.ServiceReference**;

**import** **com.bw.osgi.provider.able.HelloWorldService**;

**public** **class** **HelloWorldActivator** **implements** BundleActivator {

//bean class

**private** HelloWorldConsumer consumer;

@Override

**public** void start(BundleContext bundleContext) **throws** Exception {

ServiceReference reference = bundleContext.getServiceReference(HelloWorldService.class.getName());

consumer = **new** HelloWorldConsumer((HelloWorldService) bundleContext.getService(reference));

consumer.startTimer();

}

@Override

**public** void stop(BundleContext bundleContext) **throws** Exception {

consumer.stopTimer();

}

}

[**getAllServiceReferences**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#getAllServiceReferences(java.lang.String, java.lang.String))(java.lang.String clazz, java.lang.String filter)

will return the all the services which are registered in that bundle.

[**getBundle**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#getBundle())()   
          Returns the Bundle object associated with this BundleContext.

[**getBundles**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#getBundles())()   
          Returns a list of all installed bundles.

[**getProperty**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#getProperty(java.lang.String))(java.lang.String key)   
          Returns the value of the specified property.

[**getService**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#getService(org.osgi.framework.ServiceReference))([ServiceReference](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/ServiceReference.html)<S> reference)   
          Returns the service object referenced by the specified ServiceReference object.

[**installBundle**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#installBundle(java.lang.String))(java.lang.String location)   
          Installs a bundle from the specified location identifier.

[**registerService**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#registerService(java.lang.Class, S, java.util.Dictionary))(java.lang.Class<S> clazz, S service, java.util.Dictionary<java.lang.String,?> properties)   
          Registers the specified service object with the specified properties under the name of the specified class with the Framework.

[**ungetService**](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/BundleContext.html#ungetService(org.osgi.framework.ServiceReference))([ServiceReference](https://osgi.org/javadoc/r4v43/core/org/osgi/framework/ServiceReference.html)<?> reference)   
          Releases the service object referenced by the specified ServiceReference object.

Reponse Headers:

1. **Accept:**

application/json, text/javascript, \*/\*; q=0.01

1. **Accept-Encoding:**

gzip, deflate, br

1. **Accept-Language:**

en-US,en;q=0.9

1. **Application:**

Business Policy

1. **Connection:**

keep-alive

1. **Content-Length:**

225

1. **Content-Type:**

application/x-www-form-urlencoded; charset=UTF-8

1. **Cookie:**
2. **Host:**

sales-stg.johnhancockinsurance.com

1. **Origin:**

<https://sales-stg.johnhancockinsurance.com>

1. **Referer:**

<https://sales-stg.johnhancockinsurance.com/financial-professionals/PRD/new_my_business.html?ghbf=undefined>

1. **User-Agent:**

Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36

1. **X-Requested-With:**

XMLHttpRequest

Request Headers

1. Connection:

keep-alive

1. Content-Length:

388

1. Content-Type:

application/json

1. Date:

Wed, 27 Dec 2017 10:15:04 GMT

1. p3p:

CP="NON CUR OTPi OUR NOR UNI"

1. Server:

Apache/2.4.20 (Win64) OpenSSL/1.0.2h ASG RV Module 1.12

1. timestamp:

2017-12-27T05:15:04.932-05:00

1. wbs\_instance:

anderpmx01\_aem

Difference between doGet and doPut.

The doPut() method handles requests send by using the HTTP PUT method. The PUT method allows a client to store information on the server. For an example, you can use it to post an image file to the server.

A) doPost() method:  
1.It overcomes the aforementioned inconveniences caused by Get method as it sends the values inside the request body.  
2.There are no limitations on the number of values to be sent across and no security threats.  
3.If we want to run servlet using any html form or JSP then we can use doPost() method.

B) doGet() Method:  
1.doGet() method appends the name-value pairs on the address bar. 2.Thereby it has got the limitations on the number of characters (hence the values) to be passed.  
3.As it is directly making the values visible, certain sensitive information like our credit card number, password etc are not suggested to be sent this way  
4.Since the complete values are available in the address bar, it can be easily bookmarked but at times it may be disadvantageous.  
5. The doGet() method is used generally to retrieve the data from a server machine.

C) doPut() Method:  
1.A servlet handles PUT request made by the client to put a file,web page or a document in the server and is similar to sending a file by FTP.

Dialog and design dialog.

Custom workflow.

Polymorphism in Java

How to install bundle

* 1. mvn clean install –PautoInstallBundle
  2. through felix console.

JCR version

http://localhost:4502/crx/explorer/config/index.jsp

All the information related to licence, repository properties and java system properties and memory info will be present.

Content and experience fragments

How to list tags of a page

Example code:

        private List<String> tagsList = new ArrayList<String>();

        try{  
            ResourceResolver resourceResolver = getRequest().getResourceResolver();  
            // Session session = resourceResolver.adaptTo(Session.class);  
            TagManager tagManager = resourceResolver.adaptTo(TagManager.class);  
            Resource resource = resourceResolver.getResource("pagepath");  
            Tag[] pageTags = tagManager.getTags(resource);  
   
            for(Tag tag : pageTags){  
                tagsList.add(tag.getTitle());  
            }  
        } catch(Exception e) {  
            LOG.error("Error getting tags",e);  
            throw new ServletException(e);  
        }

How to list tags of a page

Example code:

        private List<String> tagsList = new ArrayList<String>();

        try{  
            ResourceResolver resourceResolver = getRequest().getResourceResolver();  
            // Session session = resourceResolver.adaptTo(Session.class);  
            TagManager tagManager = resourceResolver.adaptTo(TagManager.class);  
            Resource resource = resourceResolver.getResource("/etc/tags/JHINS");  
            Tag[] pageTags = tagManager.getTags(resource);  
   
            for(Tag tag : pageTags){  
                tagsList.add(tag.getTitle());  
            }  
        } catch(Exception e) {  
            LOG.error("Error getting tags",e);  
            throw new ServletException(e);  
        }

### **GETTING A JCR-BASED TAGMANAGER**

To retrieve a TagManager instance, you need to have a JCR Session and to call getTagManager(Session):

|  |  |
| --- | --- |
| 1  2  3  4 | @Reference  JcrTagManagerFactory jcrTagManagerFactory;    TagManager tagManager = jcrTagManagerFactory.getTagManager(session); |

Code samples are intended for illustration purposes only.

In the typical Sling context you can also adapt to a TagManager from the ResourceResolver:

|  |  |
| --- | --- |
| 1 | TagManager tagManager = resourceResolver.adaptTo(TagManager.class); |

Code samples are intended for illustration purposes only.

### **RETRIEVING A TAG OBJECT**

A Tag can be retrieved through the TagManager, by either resolving an existing tag or creating a new one:

|  |  |
| --- | --- |
| 1  2  3 | Tag tag = tagManager.resolve("my/tag"); // for existing tags    Tag tag = tagManager.createTag("my/tag"); // for new tags |

Code samples are intended for illustration purposes only.

For the JCR-based implementation, which maps Tags onto JCR Nodes, you can directly use Sling's adaptTomechanism if you have the resource (e.g. such as /etc/tags/default/my/tag):

|  |  |
| --- | --- |
| 1 | Tag tag = resource.adaptTo(Tag.class); |

Code samples are intended for illustration purposes only.

While a tag may only be converted *from*a resource (not a node), a tag can be converted *to*both a node and a resource :

|  |  |
| --- | --- |
| 1  2 | Node node = tag.adaptTo(Node.class);  Resource node = tag.adaptTo(Resource.class); |

### **GETTING AND SETTING TAGS**

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Getting the tags of a Resource:  Tag[] tags = tagManager.getTags(resource);    // Setting tags to a Resource:  tagManager.setTags(resource, tags); |

### **SEARCHING FOR TAGS**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // Searching for the Resource objects that are tagged with the tag object:  Iterator<Resource> it = tag.find();    // Retrieving the usage count of the tag object:  long count = tag.getCount();    // Searching for the Resource objects that are tagged with the tagID String:   RangeIterator<Resource> it = tagManager.find(tagID); |

### **DELETING TAGS**

|  |  |
| --- | --- |
| 1 | tagManager.deleteTag(tag); |

Code samples are intended for illustration purposes only.

### **REPLICATING TAGS**

It is possible to use the replication service (Replicator) with tags because tags are of type nt:hierarchyNode:

|  |  |
| --- | --- |
| 1 | replicator.replicate(session, replicationActionType, tagPath); |

OSGI Service

Scheduler

Workflow

Sitemap

#### [**How are different ways to register servlet in AEM?**](http://aeminterviewquestions.com/servlet.html#collapse2)

* Registering the servlet by path

@SlingServlet(   
paths={"/bin/customservlet/path"} )   
@Properties({   
@Property(name="service.pid", value="com.day.servlets.SampleServlet",propertyPrivate=false),   
@Property(name="service.description",value="SampleDescription", propertyPrivate=false),   
@Property(name="service.vendor",value="SampleVendor", propertyPrivate=false)   
})   
public class SampleServletname extends SlingAllMethodsServlet   
{   
@Override   
protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException, IOException   
{   
}   
}

* 2. Register servlet by ResourceType

@SlingServlet(   
resourceTypes = "sling/servlet/path",   
selectors = "json",   
extensions = "html",   
 methods = "GET")   
public class MyServlet extends SlingSafeMethodsServlet {   
@Override   
protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException, IOException {   
...   
}   
}

* 1. The @SlingServlet annotation

@SlingServlet(   
resourceTypes = "sling/servlet/default",   
selectors = "hello",   
extensions = "html",   
methods = "GET")   
public class MyServlet extends SlingSafeMethodsServlet {   
@Override   
protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException, IOException {   
...   
}   
}

* 2. The @Properties and @Property annotations

@Component(metatype = true)   
@Service(Servlet.class)   
@Properties({   
@Property(name = "sling.servlet.resourceTypes", value = "sling/servlet/default"),   
@Property(name = "sling.servlet.selectors", value = "hello"),   
@Property(name = "sling.servlet.extensions", value = "html"),   
@Property(name = "sling.servlet.methods", value = "GET")   
})   
public class MyServlet extends SlingSafeMethodsServlet {   
@Override   
protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException, IOException {   
...   
}   
}

#### [**7. Registering the servlet by path vs ResourceType?**](http://aeminterviewquestions.com/servlet.html#collapse7)

Registering the servlet by resourceType is more preferential than path , because

* use of resourceType is that the Sling Engine will take care of permissions for you. Users who cannot access a particular resource will not be able to invoke the servlet. Hence register by resourcetype is more secure.
* While defining a path , you must be specific what all paths are allowed to be used in the ServletResource OSGi service. If you define something randomly, your servlet might not be fucntional. Only a limited paths are allowed and the rest are blocked unless you open them up. This is resolved using resourceType.

|  |  |
| --- | --- |
| sling.servlet.paths | A list of absolute paths under which the servlet is accessible as a Resource. The property value must either be a single String, an array of Strings or a Vector of Strings. A servlet using this property might be ignored unless its path is included in the *Execution Paths* (servletresolver.paths) configuration setting of the SlingServletResolver service. Either this property or the sling.servlet.resourceTypes property must be set, or the servlet is ignored. If both are set, the servlet is registered using both ways. Binding resources by paths is discouraged, see [caveats when binding servlets by path](https://sling.apache.org/documentation/the-sling-engine/servlets.html#caveats-when-binding-servlets-by-path) below. |
| sling.servlet.resourceTypes | The resource type(s) supported by the servlet. The property value must either be a single String, an array of Strings or a Vector of Strings. Either this property or the sling.servlet.paths property must be set, or the servlet is ignored. If both are set, the servlet is registered using both ways. |

How to remove a property in aem

. Get the property and remove it by calling the remove() method over the Node property.

node.getProperty("propertyName").remove();

2. Set the property to NULL in this manner. (Do not try to set it directly without typecasting)

node.setProperty("propertyName",(Value)null);

How to add a property and node in aem

Node root = session.getRootNode();

            Node a    = root.addNode("node", "A");

            a.setProperty("name", "aaa");

            session.save();

How to remove a node in aem

Node root = session.getRootNode();

            Node a    = root.addNode("node", "A");

            a.setProperty("name", "aaa");

a.remove();

DAM Internal processing or Custom DAM

<http://blogs.adobe.com/contentmanagement/tag/dam-update-asset/>

<https://blogs.perficient.com/adobe/2017/04/14/the-power-of-transient-workflow-in-aem/>

How to get resource resolver in java

## [Within an OSGI Service/Compoment](https://sling.apache.org/documentation/tutorials-how-tos/getting-resources-and-properties-in-sling.html" \l "within-an-osgi-service-compoment)

You can access a resource through the ResourceResolverFactory service:

@Reference

**private** ResourceResolverFactory resolverFactory;

**public** **void** **myMethod**() {

**try** {

String resourcePath = "path/to/resource";

ResourceResolver resourceResolver = resolverFactory.getAdministrativeResourceResolver(**null**);

Resource res = resourceResolver.getResource(resourcePath);

// do something with the resource

// when done, close the ResourceResolver

resourceResolver.close();

} **catch** (LoginException e) {

// log the error

}

}

## [Within a Servlet#](https://sling.apache.org/documentation/tutorials-how-tos/getting-resources-and-properties-in-sling.html" \l "within-a-servlet)

String resourcePath = "path/to/resource";

// req is the SlingHttpServletRequest

ResourceResolver resourceResolver = req.getResourceResolver();

Resource res = resourceResolver.getResource(resourcePath);

## [Within a JSP file](https://sling.apache.org/documentation/tutorials-how-tos/getting-resources-and-properties-in-sling.html" \l "within-a-jsp-file)

When you use the <sling:defineObjects> tag in a JSP file, you have access to a few handy objects, one of them is resource, the resource that is resolved from the URL. Another one is resourceResolver, the ResourceResolver defined through the SlingHttpServletRequest.

To access a resource:

<**sling:defineObjects**>

<**%**

String resourcePath = "path/to/resource";

Resource res = resourceResolver.getResource(resourcePath);

%>

## How to get resource resolver in html using use js:

You can directly use resourceResolver.

var resourceResolver = resource.getResourceResolver();

## getting resource resolver using sub service

<http://www.aemcq5tutorials.com/tutorials/resourceresolver-from-resourceresolverfactory/>

How to return String Array from java

@Override

**public** **void** activate() **throws** Exception {

String[] str=new String[4];

str[0]=1;

str[1]=2;

str[2]=3;

str[3]=4;

**public** String[] getHyperLinks()

{

**return** hyperLinks;

}

<div data-sly-use.sightlyQueryBuilder="com.query.builder.Search">

<ul data-sly-list="${sightlyQueryBuilder.hyperLinks}"><li>${item}</li> </ul>

How to invoke third party services in AEM:

Rest:

package com.jh.jhins.servlet;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.Servlet;

import javax.servlet.ServletException;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Properties;

import org.apache.felix.scr.annotations.Property;

import org.apache.felix.scr.annotations.Service;

import org.apache.http.HttpResponse;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingAllMethodsServlet;

import org.apache.sling.commons.json.JSONArray;

import org.apache.sling.commons.json.JSONObject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Component(immediate = true, metatype = true)

@Service(Servlet.class)

@Properties({ @Property(name = "service.description", value = "NewBusinessGetPolicyStatusServlet"),

              @Property(name = "sling.servlet.paths", value = { "/bin/sling/NewBusinessGetPolicyStatus" }),

              @Property(name = "service.vendor", value = "JHINS"),

              @Property(name = "sling.servlet.methods", value = "POST", propertyPrivate = true) })

public class NewBusinessGetPolicyStatusServlet extends SlingAllMethodsServlet {

       private static final long serialVersionUID = 1L;

private static final Logger LOG = LoggerFactory.getLogger(NewBusinessGetPolicyStatusServlet.class);

       protected final void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)

                     throws ServletException, IOException {

              this.doPost(request, response);

       }

       protected final void doPost(SlingHttpServletRequest request, SlingHttpServletResponse response)

                     throws ServletException, IOException  {

              String insuredCaseId=request.getParameter("insuredCaseId");

              String insuredCaseCode=request.getParameter("insuredCaseCode");

              /\*JSONObject jsonObj = new JSONObject();

              HttpResponse resp=null;

                                 PrintWriter out = response.getWriter();

              try{

                                               jsonObj.put("policyId","1-10DG9R");

                                               jsonObj.put("userRole","SuperUser");

                                               jsonObj.put("UUID","1000001123456");

                                               jsonObj.put("PolNumber","57638884");

                                               jsonObj.put("ProductLineCode","100");

                                               jsonObj.put("partyRole","insured");

                                               DefaultHttpClient httpClient = new DefaultHttpClient();

                                               HttpPost postRequest = new HttpPost("<https://b04amxd02.dev.manulifeusa.com:7081/Insurance/NewBusinessPolicy/GetPolicyStatus>");

                                               postRequest.addHeader("Application","BusinessPolicy");

                                               //String encodedCredentials = new String(Base64.encodeBase64("reddbhu:Bhuvana@2017".getBytes("UTF-8")));

                                               postRequest.addHeader("Authorization","Basic cmVkZGJodTpCaHV2YW5hQDIwMTc=");

                                               StringEntity input = new StringEntity(jsonObj.toString());

                                               input.setContentType("application/json");

                                               postRequest.setEntity(input);

                                               resp = httpClient.execute(postRequest);

                                               LOG.info("output@@@@"+resp);

                                               String respContent = org.apache.http.util.EntityUtils.toString(resp.getEntity());

                                               LOG.info("output"+ respContent);

                                               response.setContentType("application/json");

                                               response.setCharacterEncoding("UTF-8");

                                               out.write(respContent);

                              }

              catch(Exception e){

                     LOG.info("ERROR IN FETCHING VALUES"+e);

              }\*/

<https://www.mkyong.com/webservices/jax-rs/restful-java-client-with-apache-httpclient/>

|  |  |
| --- | --- |
| **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | **JAX-WS** is the java API for SOAP web services. | **JAX-RS** is the java API for RESTful web services. |
| 6) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 7) | SOAP **requires more bandwidth** and resource than REST. | REST **requires less bandwidth** and resource than SOAP. |
| 8) | SOAP **defines its own security**. | RESTful web services **inherits security measures** from the underlying transport. |
| 9) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
| 10) | SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. |

import javax.xml.soap.\*;

public class SOAPClientSAAJ {

// SAAJ - SOAP Client Testing

public static void main(String args[]) {

/\*

The example below requests from the Web Service at:

http://www.webservicex.net/uszip.asmx?op=GetInfoByCity

To call other WS, change the parameters below, which are:

- the SOAP Endpoint URL (that is, where the service is responding from)

- the SOAP Action

Also change the contents of the method createSoapEnvelope() in this class. It constructs

the inner part of the SOAP envelope that is actually sent.

\*/

String soapEndpointUrl = "http://www.webservicex.net/uszip.asmx";

String soapAction = "http://www.webserviceX.NET/GetInfoByCity";

callSoapWebService(soapEndpointUrl, soapAction);

}

private static void createSoapEnvelope(SOAPMessage soapMessage) throws SOAPException {

SOAPPart soapPart = soapMessage.getSOAPPart();

String myNamespace = "myNamespace";

String myNamespaceURI = "http://www.webserviceX.NET";

// SOAP Envelope

SOAPEnvelope envelope = soapPart.getEnvelope();

envelope.addNamespaceDeclaration(myNamespace, myNamespaceURI);

/\*

Constructed SOAP Request Message:

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:myNamespace="http://www.webserviceX.NET">

<SOAP-ENV:Header/>

<SOAP-ENV:Body>

<myNamespace:GetInfoByCity>

<myNamespace:USCity>New York</myNamespace:USCity>

</myNamespace:GetInfoByCity>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

\*/

// SOAP Body

SOAPBody soapBody = envelope.getBody();

SOAPElement soapBodyElem = soapBody.addChildElement("GetInfoByCity", myNamespace);

SOAPElement soapBodyElem1 = soapBodyElem.addChildElement("USCity", myNamespace);

soapBodyElem1.addTextNode("New York");

}

private static void callSoapWebService(String soapEndpointUrl, String soapAction) {

try {

// Create SOAP Connection

SOAPConnectionFactory soapConnectionFactory = SOAPConnectionFactory.newInstance();

SOAPConnection soapConnection = soapConnectionFactory.createConnection();

// Send SOAP Message to SOAP Server

SOAPMessage soapResponse = soapConnection.call(createSOAPRequest(soapAction), soapEndpointUrl);

// Print the SOAP Response

System.out.println("Response SOAP Message:");

soapResponse.writeTo(System.out);

System.out.println();

soapConnection.close();

} catch (Exception e) {

System.err.println("\nError occurred while sending SOAP Request to Server!\nMake sure you have the correct endpoint URL and SOAPAction!\n");

e.printStackTrace();

}

}

private static SOAPMessage createSOAPRequest(String soapAction) throws Exception {

MessageFactory messageFactory = MessageFactory.newInstance();

SOAPMessage soapMessage = messageFactory.createMessage();

createSoapEnvelope(soapMessage);

MimeHeaders headers = soapMessage.getMimeHeaders();

headers.addHeader("SOAPAction", soapAction);

soapMessage.saveChanges();

/\* Print the request message, just for debugging purposes \*/

System.out.println("Request SOAP Message:");

soapMessage.writeTo(System.out);

System.out.println("\n");

return soapMessage;

}

}

<https://docs.oracle.com/javaee/5/tutorial/doc/bnbis.html>

<http://www.jitendrazaa.com/blog/java/create-soap-message-using-java/>

What is Multi Site Manager (MSM)  
  
Multi Site Manager (MSM) allows aem developers to create copy of existing site and automatically update the copy when changes are done to the source site.

Advantages of using MSM :-

Easily manage multiple websites that share common content.

MSM allows aem developers to define relations between the sites so that content changes in one site are automatically replicated in all other sites.

Reduces the time to manage your websites and increases the re-use of common content.

Maintain common look and feel across all websites.  
  
When we should use MSM

Sites are in multiple countries.

Sites are in multiple languages.

Need to reuse/replicate same content in multiple websites.

Maintain consistent branding and content across websites.  
  
Email validation for a touch UI dialog

(function (document, $, ns) {

    "use strict";

    $(document).on("click", ".cq-dialog-submit", function (e) {

        e.stopPropagation();

        e.preventDefault();

        var $form = $(this).closest("form.foundation-form"),

            emailid = $form.find("[name='./email']").val(),

               message, clazz = "coral-Button ",

         patterns = {

             emailadd: /^([a-z\d!#$%&'\*+\-\/=?^\_`{|}~\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]+(\.[a-z\d!#$%&'\*+\-\/=?^\_`{|}~\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]+)\*|"((([ \t]\*\r\n)?[ \t]+)?([\x01-\x08\x0b\x0c\x0e-\x1f\x7f\x21\x23-\x5b\x5d-\x7e\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]|\\[\x01-\x09\x0b\x0c\x0d-\x7f\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]))\*(([ \t]\*\r\n)?[ \t]+)?")@(([a-z\d\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]|[a-z\d\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF][a-z\d\-.\_~\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]\*[a-z\d\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\.)+([a-z\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]|[a-z\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF][a-z\d\-.\_~\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF]\*[a-z\u00A0-\uD7FF\uF900-\uFDCF\uFDF0-\uFFEF])\.?$/i

        };

        if(emailid != "" && !patterns.emailadd.test(emailid) && (emailid != null)) {

                ns.ui.helpers.prompt({

                title: Granite.I18n.get("Invalid Input"),

                message: "Please Enter a valid Email Address",

                actions: [{

                    id: "CANCEL",

                    text: "CANCEL",

                    className: "coral-Button"

                }],

            callback: function (actionId) {

                if (actionId === "CANCEL") {

                }

            }

        });

        }else{

                 $form.submit();

        }

    });

})(document, Granite.$, Granite.author);

I18n

**b. Normal way to implement i18n in Sightly :**

We can implement i18n directly by using the below syntax :

${'key' @ i18n}

${'key' @ i18n, source='user', hint='Translation Hint'}

${'key' @ i18n, locale='en', hint='Translation Hint'}

The default **source** for the language is ‘resource’, meaning that the text gets translated to the same language as the content. If you specify it to ‘user’ than the language is taken from the browser locale or from the locale of the logged-in user.But when we provide an explicit **locale** than we overrides the source settings and would take the specified locale.

In Sightly if we don’t define the attribute locale than it would automatically convert the text according to current page locale .

The **hint** is used to provide information about the context for the translators.

**Issue and it’s alternative approach :**

When I tried using the above approach for i18n implementation in sightly, it didn’t worked in a usual manner for different locale as it should work on.

**Solution :**So if such a scenario happened with you, you can also achieve it by creating a **local template** to which you can pass the variable identifying your key and than calling it by using  data-sly-call :

Implementation for the same is as below :

<template data-sly-template.i18="${@ key}">

${key @ i18n}

</template>

<div data-sly-call="${i18 @ key='key-name'}"></div>

This approach holds good and worked for me in all use cases .

Config nodes example

Implimentation class

package com.jh.jhins.impl;

import java.io.IOException;

import java.util.Dictionary;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Reference;

import org.apache.felix.scr.annotations.Service;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.jh.jhins.constants.JHINSConstants;

import com.jh.jhins.interfaces.JHINSConfigService;

import org.osgi.service.cm.Configuration;

import org.osgi.service.cm.ConfigurationAdmin;

@Service

@Component(immediate = true, metatype = true, name = "JHINSConfigServiceImpl", description = "Programatically get  properties of OSGi configurations.")

public class JHINSConfigServiceImpl implements JHINSConfigService {

                /\*\* The service to get OSGi configs \*/

                @Reference

                private ConfigurationAdmin configAdmin;

                private static final Logger LOG = LoggerFactory

                                                .getLogger(JHINSConfigServiceImpl.class);

                /\*\*

     \* Get the value of an OSGi configuration string property for a given PID.

     \*

     \*

     \* @param property The property of the config to retrieve

     \* @return The property value

     \*/

                public String getConfigProperty(final String property) {

                                String propertyValue ="" ;

                                try {

                                                Configuration conf = configAdmin.getConfiguration(JHINSConstants.CONFIG\_PID);

                                                @SuppressWarnings(JHINSConstants.UNCHECKED)

                                                Dictionary<String, Object> properties = conf.getProperties();

                                                if (properties != null) {

                                                                if (!properties.isEmpty()) {

                                                                                if (properties.get(property) != null) {

                                                                                                propertyValue = properties.get(property).toString();

                                                                                }

                                                                }

                                                }

                                } catch (IOException e) {

                                                LOG.error("IOException",e);

                                }

                                return propertyValue;

                }

}

Interface

**package** com.jh.jhins.interfaces;

**public** **interface** JHINSConfigService {

**public** String getConfigProperty(**final** String property);

}

Usage:

package com.jh.jhins.servlet;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.Servlet;

import javax.servlet.ServletException;

import org.apache.commons.lang.StringUtils;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Properties;

import org.apache.felix.scr.annotations.Property;

import org.apache.felix.scr.annotations.Reference;

import org.apache.felix.scr.annotations.Service;

import org.apache.http.HttpResponse;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.DefaultHttpClient;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingAllMethodsServlet;

import org.apache.sling.commons.json.JSONObject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.jh.jhins.interfaces.JHINSConfigService;

@Component(immediate = true, metatype = true)

@Service(Servlet.class)

@Properties({ @Property(name = "service.description", value = "NewBusinessGetPolicyStatusServlet"),

              @Property(name = "sling.servlet.paths", value = { "/bin/sling/NewBusinessGetPolicyStatus" }),

              @Property(name = "service.vendor", value = "JHINS"),

              @Property(name = "sling.servlet.methods", value = "POST", propertyPrivate = true) })

public class NewBusinessGetPolicyStatusServlet extends SlingAllMethodsServlet {

       private static final long serialVersionUID = 1L;

       @Reference

                                private JHINSConfigService JHINSConfig;

private static final Logger LOG = LoggerFactory.getLogger(NewBusinessGetPolicyStatusServlet.class);

       protected final void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)

                     throws ServletException, IOException {

              this.doPost(request, response);

       }

       protected final void doPost(SlingHttpServletRequest request, SlingHttpServletResponse response)

                     throws ServletException, IOException  {

              String policyId=request.getParameter("policyId");

              String userRole=request.getParameter("userRole");

              String UUID=request.getParameter("UUID");

              String policyNumber=request.getParameter("policyNumber");

              String predefinedLineCode=request.getParameter("ProductLineCode");

              String partyRole=request.getParameter("partyRole");

              JSONObject jsonObj = new JSONObject();

              HttpResponse resp=null;

                                 PrintWriter out = response.getWriter();

              try{

                 if(StringUtils.isNotBlank(policyId)){

                                 jsonObj.put("policyId",policyId);}

                 if(StringUtils.isNotBlank(userRole)){

                                 jsonObj.put("userRole",userRole);}

                 if(StringUtils.isNotBlank(UUID)){

                                 jsonObj.put("UUID",UUID);}

                 if(StringUtils.isNotBlank(policyNumber)){

                                 jsonObj.put("PolNumber",policyNumber);}

                 if(StringUtils.isNotBlank(predefinedLineCode)){

                                 jsonObj.put("ProductLineCode",predefinedLineCode);}

                 if(StringUtils.isNotBlank(partyRole)){

                                 jsonObj.put("partyRole",partyRole);}

                                               /\*jsonObj.put("policyId","1-10DG9R");

                                               jsonObj.put("userRole","SuperUser");

                                               jsonObj.put("UUID","1000001123456");

                                               jsonObj.put("PolNumber","57638884");

                                               jsonObj.put("ProductLineCode","100");

                                               jsonObj.put("partyRole","insured");\*/

                                               String getPolicyURL=(String) JHINSConfig.getConfigProperty("newbusinessgetpolicy.service.url");

                                               LOG.info("test@@@@"+getPolicyURL);

                                               DefaultHttpClient httpClient = new DefaultHttpClient();

                                               HttpPost postRequest = new HttpPost(getPolicyURL);

                                               postRequest.addHeader("Application","BusinessPolicy");

                                               postRequest.addHeader("Authorization","Basic cmVkZGJodTpCaHV2YW5hQDIwMTc=");

                                               StringEntity input = new StringEntity(jsonObj.toString());

                                               input.setContentType("application/json");

                                               postRequest.setEntity(input);

                                               resp = httpClient.execute(postRequest);

                                               LOG.info("output@@@@"+resp);

                                               String respContent = org.apache.http.util.EntityUtils.toString(resp.getEntity());

                                               LOG.info("output"+ respContent);

                                               response.setContentType("application/json");

                                               response.setCharacterEncoding("UTF-8");

                                               out.write(respContent);

                              }

              catch(Exception e){

                     LOG.info("ERROR IN FETCHING VALUES"+e);

              }

        }

}

How to populate a dropdown based on the other dropdown classic ui.

We can do this by creating a servlet.

Dialog structure

Suppose we have a dialog with countries US and Australia as dropdown options.

And I have 2nd dropdown till selection node.

Now I want to display US states in the second dropdown when I select us country from the first dropdown accordingly for Australia also.

Now create a second dropdown as shown below.

* **allowBlank** (String) - false
* **fieldLabel**(String) – Selection
* **name**(String) - ./name
* **options**(Sting) - /apps/populatedropdown.json (**This is the value that maps to the Movie sling servlet**)
* **type**(String) - select
* **xtype**(String) -type

Servlet code will be:

@Component(metatype = false)

@Service(value = Servlet.class)

@Properties({

        @Property(name = "sling.servlet.resourceTypes", value = "/apps/populatedropdown"),

        @Property(name = "sling.servlet.methods", value = {"GET", "POST"}),

        @Property(name = "service.description", value = "Populate states dropdown based on country value")

})

public class DropDownPopulator extends SlingAllMethodsServlet {

    private Logger logger = LoggerFactory.getLogger(DropDownPopulator.class);

    protected void doPost(SlingHttpServletRequest request,

                          final SlingHttpServletResponse response)

            throws ServletException, IOException {

        response.setHeader("Access-Control-Allow-Origin", "\*");

        response.setContentType("application/json");

        response.setCharacterEncoding("UTF-8");

        try {

            String US\_STATES[] = {"0=Alabama",

                    "1=Alaska",

                    "2=Arizona",

                    "3=Arkansas",

                    "4=California",

                    "5=Colorado",

                    "6=Connecticut",

                    "7=Delaware",

                    "8=Florida",

                    "9=Georgia",

                    "10=Hawaii",

                    "11=Idaho",

                    "12=Illinois",

                    "13=Indiana",

                    "14=Iowa",

                    "15=Kansas",

                    "16=Kentucky",

                    "17=Louisiana",

                    "18=Maine",

                    "19=Maryland",

                    "20=Massachusetts",

                    "21=Michigan",

                    "22=Minnesota",

                    "23=Mississippi",

                    "24=Missouri",

                    "25=Montana",

                    "26=Nebraska",

                    "27=Nevada",

                    "28=New Hampshire",

                    "29=New Jersey",

                    "30=New Mexico",

                    "31=New York",

                    "32=North Carolina",

                    "33=North Dakota",

                    "34=Ohio",

                    "35=Oklahoma",

                    "36=Oregon",

                    "37=Pennsylvania",

                    "38=Rhode Island",

                    "39=South Carolina",

                    "40=South Dakota",

                    "41=Tennessee",

                    "42=Texas",

                    "43=Utah",

                    "44=Vermont",

                    "45=Virginia",

                    "46=Washington",

                    "47=West Virginia",

                    "48=Wisconsin",

                    "49=Wyoming"};

            String AUSTRALIAN\_STATES[] = {"0=Ashmore and Cartier Islands",

                    "1=Australian Antarctic Territory",

                    "2=Australian Capital Territory",

                    "3=Christmas Island",

                    "4=Cocos (Keeling) Islands",

                    "5=Coral Sea Islands",

                    "6=Heard Island and McDonald Islands",

                    "7=Jervis Bay Territory",

                    "8=New South Wales",

                    "9=Norfolk Island",

                    "10=Northern Territory",

                    "11=Queensland",

                    "12=South Australia",

                    "13=Tasmania",

                    "14=Victoria",

                    "15=Western Australia"};

            String country = request.getParameter("country");

            JSONArray stateJsonArray = new JSONArray();

            if (country.length() > 0) {

                if ("australia".equalsIgnoreCase(country)) {

                    stateJsonArray = new JSONArray();

                    for (String state : AUSTRALIAN\_STATES) {

                        stateJsonArray.put(state);

                    }

                } else if ("unitedstates".equalsIgnoreCase(country)) {

                    stateJsonArray = new JSONArray();

                    for (String state : US\_STATES) {

                        stateJsonArray.put(state);

                    }

                }

                response.setContentType("application/json");

                response.getWriter().write(stateJsonArray.toString());

            }

        } catch ( Exception e) {

            logger.error(e.getMessage(), e);

        }

    }

}

In classUI to call servlet from dialog add options property to selection widget and provide servlet path to the property.

How to populate a dropdown statically and dynamically.

Static way Classic UI:

Create a json file right under component and add key value pair like below.

[

{

"value": 10,

"text": "A"

}, {

"value": 20,

"text": "B"

}

]

Create a dropdown selection and add this json file path in options property like.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | | false | | false | | false | |
| 2 | fieldLabel | | String | | Select your choice | | false | | false | | false | | false | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | jcr:primaryType | Name | cq:Widget | true | true | false | true |
| 4 | name | String | ./selectionlist | false | false | false | false |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | options | String | /apps/aemtasks/components/content/hideandshowtabs/dropdown.json | false | false | false | false |
| 6 | type | String | select | false | false | false | false |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 | xtype | String | selection |

Dynamic way classic UI:

* **allowBlank** (String) - false
* **fieldLabel**(String) - Movie Name
* **name**(String) - ./name
* **options**(Sting) - /bin/moviename.json (**This is the value that maps to the Movie sling servlet**)
* **optionsRoot**(String) - root
* **type**(String) - select
* **xtype**(String) -type

Servlet

package com.aem.community.sample;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Service;

import org.apache.felix.scr.annotations.sling.SlingServlet;

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingSafeMethodsServlet;

import org.apache.sling.commons.json.JSONArray;

import org.apache.sling.commons.json.JSONException;

import org.apache.sling.commons.json.JSONObject;

import org.apache.sling.commons.json.io.JSONWriter;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

@Service(MovieList.class)

@SlingServlet(paths = {"/bin/moviename"}, generateComponent = false)

@Component(label = "Dropdown Movie data provider", description = "This servlet provides list of movie names in drop down",

enabled = true, immediate = true, metatype = false)

public class MovieList extends SlingSafeMethodsServlet {

    private static final Logger LOGGER = LoggerFactory.getLogger(MovieList.class);

    @Override

    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) {

        try {

            JSONObject eachOption;

            JSONArray optionsArray = new JSONArray();

            String[] movies = { "Terminator", "Kicking & Screaming","Harold and Maude",

                    "Ratatouille", "10 Things I Hate About You", "American Beauty",

                    "The Dark Knight", "The Wolf of Wall Street","Mean Girls",

                    "Inception", "Life Is Beautiful", "No Country for Old Men" };

            String[] returnData = new String[movies.length];

            for (int i = 0; i < movies.length; i++) {

                eachOption = new JSONObject();

                returnData[i] = movies[i];

                eachOption.put("text", returnData[i]);

                eachOption.put("value", returnData[i]);

                optionsArray.put(eachOption);

            }

            JSONObject finalJsonResponse = new JSONObject();

            //Adding this finalJsonResponse object to showcase optionsRoot property functionality

            finalJsonResponse.put("root", optionsArray);

            response.getWriter().println(finalJsonResponse.toString());

        } catch (JSONException e) {

            LOGGER.error("Json Exception occured while adding data to JSON Object : ", e);

        } catch (IOException e) {

            LOGGER.error("IOException occured while getting Print Writer from SlingServletResponse : ", e);

        }

    }

}

Query Builder

List of AEM Standard Predicates:-

Here is the list of available standard predicates, that predicate evaluator uses to resolve at run time.

* path : This is used to search under a particular hierarchy. However we can create multiple path predicates to search under multiple paths (Resolves to PathPredicateEvaluator)
* path.self=true : If true searches the subtree including the main node given in path, if false searches the subtree only.
* path.exact=true : If true exact path is matched, if false all descendants are included.
* path.flat=true : If true searches only the direct children .
* type: It is used for searching particular nodetype only. ( For EX:- dam:Asset)(Resolves to TypePredicateEvaluator)
* property: This is used for search based on specific jcr property.(Resolves to JcrPropertyPredicateEvaluator)
* property.value : the property value to search . Mutilple values of a particular property could be given using N\_property.value=xxx , where N is number from 1 to N and xxx is the value.
* property.depth : The number of additional levels to search under a node. eg. if property.depth=3 then the property is searched till 3 level from base node. It is mostly used for nested search. property.and : If multiple properties are present , by default an OR operator is applied. If you want an AND , you may use property.and=true
* property.operation : “equals” for exact match (default), “unequals” for unequality comparison, “like” for using the jcr:like xpath function , “not” for no match , (value param will be ignored) or “exists” for existence match .(value can be true – property must exist).

1. fulltext: It is used to search terms for fulltext search (This is case in sensitive search. fulltext predicate resolves to FulltextPredicateEvaluator)
2. fulltext.relPath : can specify the relative path to search in (eg. property or subnode) eg. fulltext.relPath=jcr:content or fulltext.relPath=jcr:content/@cq:tags
3. daterange : This predicate is used to search a date property range. (Resolves to DateRangePredicateEvaluator) daterange.property : Specify the date property which on which query needs to run.
4. daterange.lowerBound : Fix a lower bound date range eg. 2010-07-25
5. daterange.lowerOperation : “>” (default) or “>=”
6. daterange.upperBound: Fix a upper bound date range eg. 2013-07-26

daterange.upperOperation: “<” (default) or “<=”  
  
nodename: This is used to search exact nodenames for the result set. It allows few wildcards like: nodename=text\* will search for this and any character after it . nodename=text? will search for all records that starts with text but will not return result that contains only text. (Resolves to NodenamePredicateEvaluator).  
  
Note:- By default Two different predicates are separated by AND operator. Lets understand this scenario by a simple example.

fulltext=Management

group.p.or=true

group.1\_path=/content/geometrixx/en

group.2\_path=/content/dam/geometrixx

Final Xpath query will be created as (fulltext AND (path=… OR path=…)) Example 2 :- 1 2 3 4 5

fulltext=Management

group.p.or=true

group.1\_group.path=/content/geometrixx/en

group.1\_group.type=cq:Page

group.2\_group.path=/content/dam/geometrixx

group.2\_group.type=dam:Asset

Final Xpath query will be created as (fulltext AND ( (path= AND type=) OR (path= AND type=) ))

orderBy: This predicate is used to sort the result sets obtained in the query. e.g. orderby=@jcr:score or orderby=@jcr:content/cq:lastModified

orderby.sort: You may define the sorting way for the search results e.g.

orderby.desc=true  or orderby.sort = desc for descending and orderby.asc=true or orderby.sort=asc for ascending.

orderby.case: support case insensitive orderby.case=ignore (since 6.2)

orderby=mypredicate (eg: orderby=path) : this can also be used to sort by path. Refining the Results: In order to refine the results there are some parameters which could be leveraged:

Multiple Ordering:- Multiple ordering can also be achieved using orderby predicate 1 2 3 1\_orderby=@cq:tags

2\_orderby=@cq:lastModified

3\_orderby=nodename

What is the difference between 5.6 and 6.1/6.2

Jackrabbit Oak: Compared to JCR, Oak offers improved performance, scalability. You also have an option of using NoSQL DB like MongoDB as persistence layer to support clustering and user generated data scenarios.

Sightly: New templating language, makes markup look beautiful, enforces separation of the markup from logic and also offers XSS protection by default.

Touch UI: Classic UI in CQ5 which is ExtJS based has been upgraded to Touch UI which supports touch enabled devices – built using Coral UI framework.

Search – Apache Solr: Default search engine in CQ5 was Lucene, this has been upgraded to Solr. You can now configure Solr server as search engine for your AEM application.

differences between implementing with WCMUsePojo and Sling Models

WCMUsePojo will need to be extend from that class, whereas Sling Models can be standalone class with @Model annotation and no keyword.

1. 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
2. You may use Felix annotation @Reference to reference to an available OSGI service, whereas in Sling Models, you will use @Inject or @OSGiService
3. 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.
4. For WCMUsePojo, you will need to overwrite the activate() method, whereas in Sling Models, your init method will be called in the @PostConstruct annotation

Overall, I think Sling Models are pure POJOs that separate logic and presentation. They are clean and annotation driven, but also extensible with custom injectors and annotations.

In the TitleModel class, the @Model is required to register the Java class as a Sling Model.

# Access OSGI ser­vice from the WCMUse-class in Sightly

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | package com.adobeaemclub.adobeaemclub.core.services;    public interface SightlySerivceInterface {  String getDeveloperName();  String getDeveloperProfile();  String getDeveloperSkills();  String getDeveloperData();  } |

package com.adobeaemclub.adobeaemclub.core.services;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.apache.felix.scr.annotations.Component;

import org.apache.felix.scr.annotations.Service;

@Component

@Service

public class SightlySerivce implements SightlySerivceInterface {

Logger logger = LoggerFactory.getLogger(SightlySerivce.class);

@Override

public String getDeveloperName() {

return "John";

}

@Override

public String getDeveloperProfile() {

return "AEM Developer";

}

@Override

public String getDeveloperSkills() {

return "JAVA, OSGI, HTML, JS";

}

        @Override

public String getDeveloperData() {

String name = this.getDeveloperName();

String profile = this.getDeveloperProfile();

String skills = this.getDeveloperSkills();

return name + " is a " + profile + ", He is expert in skills like " + skills;

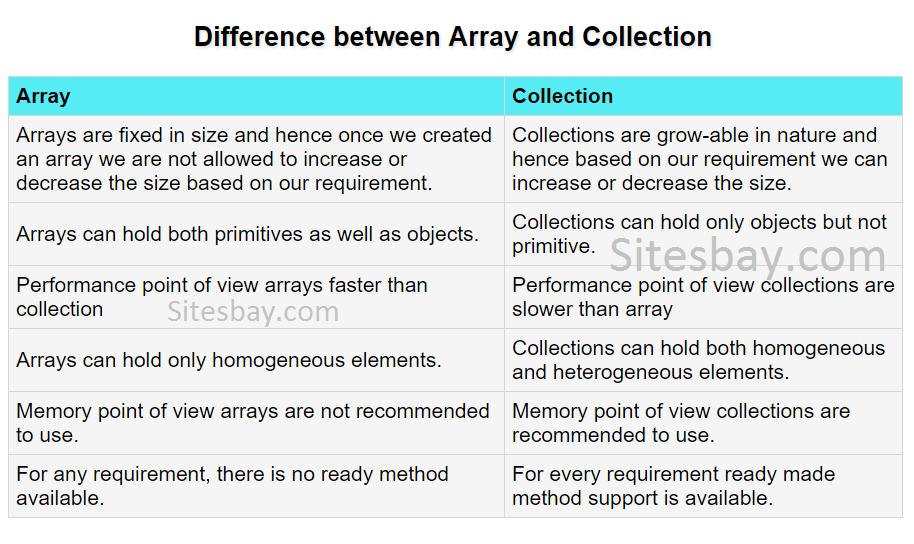
}

}

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | package com.adobeaemclub.adobeaemclub.core.services;    import org.slf4j.Logger;  import org.slf4j.LoggerFactory;    import com.adobe.cq.sightly.WCMUse;    public class Developer extends WCMUse {  Logger logger = LoggerFactory.getLogger(Developer.class);  protected String detail;    @Override    public void activate() {        SightlySerivceInterface service = getSlingScriptHelper().getService(SightlySerivceInterface.class);      detail = service.getDeveloperData();    }      public String getDetails() {      return this.detail;    }  } |

|  |  |
| --- | --- |
| 3  4 | Our service value:-  <div data-sly-use.info="com.adobeaemclub.adobeaemclub.core.services.Developer">      ${info.details}  </div> |

Difference between arrays and collections



Difference between array and collection is 1. Array is group of similar data type object. Collection is group of homogeneous and heterogeneous data type object. 2.Array is fixed in size. Collection is not fixed in size. 3.Array is strong type. Collection is not strong type.We use generic type to make it strong. Note:The difference between array and collection or array and list or array and datalist will be same.

**Fail fast and fail safe iterators:**

Fail-Fast iterators immediately throw ConcurrentModificationException if there is **structural modification** of the collection. Structural modification means adding, removing or updating any element from collection while a thread is iterating over that collection. Iterator on ArrayList, HashMap classes are some examples of fail-fast Iterator.

* They use original collection to traverse over the elements of the collection.
* These iterators don’t require extra memory.

To know whether the collection is structurally modified or not, fail-fast iterators use an internal flag called modCount which is updated each time a collection is modified.Fail-fast iterators checks the modCount flag whenever it gets the next value (i.e. using next() method), and if it finds that the modCount has been modified after this iterator has been created, it throws ConcurrentModificationException.

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

public class FailFastExample {

    public static void main(String[] args)

    {

        Map<String, String> cityCode = new HashMap<String, String>();

        cityCode.put("Delhi", "India");

        cityCode.put("Moscow", "Russia");

        cityCode.put("New York", "USA");

        Iterator iterator = cityCode.keySet().iterator();

        while (iterator.hasNext()) {

            System.out.println(cityCode.get(iterator.next()));

            // adding an element to Map

            // exception will be thrown on next call

            // of next() method.

            cityCode.put("Istanbul", "Turkey");

        }

    }

}

Fail-Safe iterators don’t throw any exceptions if a collection is structurally modified while iterating over it. This is because, they operate on the clone of the collection, not on the original collection and that’s why they are called fail-safe iterators. Iterator on CopyOnWriteArrayList, ConcurrentHashMap classes are examples of fail-safe Iterator.

**Example of Fail-Safe Iterator which does not create separate copy**

|  |
| --- |
| // Java program to illustrate  // Fail-Safe Iterator which  // does not create separate copy  import java.util.concurrent.ConcurrentHashMap;  import java.util.Iterator;    public class FailSafeItr {      public static void main(String[] args)      {            // Creating a ConcurrentHashMap          ConcurrentHashMap<String, Integer> map              = new ConcurrentHashMap<String, Integer>();            map.put("ONE", 1);          map.put("TWO", 2);          map.put("THREE", 3);          map.put("FOUR", 4);            // Getting an Iterator from map          Iterator it = map.keySet().iterator();            while (it.hasNext()) {              String key = (String)it.next();              System.out.println(key + " : " + map.get(key));                // This will reflect in iterator.              // Hence, it has not created separate copy              map.put("SEVEN", 7);          }      }  } |