# Developing first OSGi bundle

In this post, we will be creating and deploying our custom bundle into AEM.  
While doing so, we will be learning concepts of **OSGi Components** and **OSGi Services**.

**Software Modularity**

In modern times, complex software can be thought of as a collection of various modules or components. These modules are normally independent of each other and modification in one module does not affect the other modules.

These modules interact with each other via an API. The API is defined as a set of classes and methods which can be used from other components.

If a module uses an API from another module, it has a dependency on the other module, i.e., it requires the other module exists and works correctly.

A module that is used by other components should try to keep its API stable. This avoids that a change affects other modules. But it should be free to change its internal implementation.

**What is OSGi?**

OSGi stands for ***Open Service Gateway initiative.***It is a Java framework for developing and deploying modular software programs and libraries.

OSGi has two parts.

* The first part is a specification for modular components called bundles, which are commonly referred to as plug-ins. The specification defines an infrastructure for a bundle's life cycle and determines how bundles will interact.
* The second part of OSGi is a Java Virtual Machine (JVM)-level service registry that bundles can use to publish, discover and bind to services in a service-oriented architecture (SOA).

The components and services can be dynamically installed, activated, de-activated, updated and uninstalled.

The OSGi specification has several implementations, for example, **Equinox**, **Knopflerfish**,and **Apache Felix**. AEM uses [Apache Felix](http://felix.apache.org/) implementation. A bundle is the smallest unit of modularization which means in OSGi, a software component is a bundle.

In AEM, along with out of the box bundles, we can also install our custom bundles very easily. So without further ado, let us start creating an OSGi bundle.

**Creating a Bundle**

We will now create an OSGi bundle that has OSGi service which reads JSON data using HTTP GET request from the URL - <https://jsonplaceholder.typicode.com/todos/>. Following are the steps to create an OSGi bundle -

*package org.redquark.demo.core.utils;*

*import java.io.BufferedReader;*

*import java.io.IOException;*

*import java.io.InputStreamReader;*

*import java.net.MalformedURLException;*

*import java.net.URL;*

*import javax.net.ssl.HttpsURLConnection;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* This class has all the Network related calls*

*\*/*

*public final class Network {*

*private static final String USER\_AGENT = "Mozilla/5.0";*

*public static String readJson(String url) {*

*try {*

*/\*\**

*\* Get the URL object from the passed url string*

*\*/*

*URL requestURL = new URL(url);*

*/\*\**

*\* Creating an object of HttpURLConnection*

*\*/*

*HttpsURLConnection connection = (HttpsURLConnection) requestURL.openConnection();*

*/\*\**

*\* Setting the request method*

*\*/*

*connection.setRequestMethod("GET");*

*/\*\**

*\* Setting the request property*

*\*/*

*connection.setRequestProperty("User-Agent", USER\_AGENT);*

*/\*\**

*\* Get the response code*

*\*/*

*int responseCode = connection.getResponseCode();*

*if (responseCode == HttpsURLConnection.HTTP\_OK) {*

*/\*\**

*\* Getting an instance of BufferedReader to read the response returned*

*\*/*

*BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));*

*/\*\**

*\* String which will read the response line by line*

*\*/*

*String inputLine;*

*/\*\**

*\* StringBuffer object to append the string as a whole*

*\*/*

*StringBuffer response = new StringBuffer();*

*/\*\**

*\* Read until empty line is encountered*

*\*/*

*while ((inputLine = in .readLine()) != null) {*

*/\*\**

*\* Append each line to make the response as a whole*

*\*/*

*response.append(inputLine);*

*}*

*/\*\**

*\* Closing the BufferedReader to avoid memory leaks*

*\*/*

*in .close();*

*/\*\**

*\* Return the response*

*\*/*

*return response.toString();*

*}*

*} catch (MalformedURLException e) {*

*e.printStackTrace();*

*} catch (IOException e) {*

*e.printStackTrace();*

*}*

*return "";*

*}*

*}*

* Create an AEM Multimodule Project in Eclipse.
* Create an interface **ReadJsonService** in the package **org.redquark.demo.core.services**. This will be exposed to our service.

*package org.redquark.demo.core.services;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* Service which will be exposed*

*\*/*

*public interface ReadJsonService {*

*/\*\**

*\* @return JSON String*

*\*/*

*public String getData();*

*}*

* Now we need to create an implementation class for this service which will have the business logic. Hence, create a class **ReadJsonDataImpl** in the package **org.redquark.demo.core.services.impl**.

*package org.redquark.demo.core.services.impl;*

*import org.osgi.service.component.annotations.Component;*

*import static org.redquark.demo.core.constants.AppConstants.URL;*

*import org.redquark.demo.core.services.ReadJsonService;*

*import org.redquark.demo.core.utils.Network;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* Implementation of ReadJsonService*

*\*/*

*@Component(immediate = true, service = ReadJsonService.class)*

*public class ReadJsonDataImpl implements ReadJsonService {*

*/\*\**

*\* Overridden method which will read the JSON data via an HTTP GET call*

*\*/*

*@Override*

*public String getData() {*

*String response = Network.readJson(URL);*

*return response;*

*}*

*}*

* Now create constants class for storing project constants and Network class for the HTTP requests

*package org.redquark.demo.core.constants;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* This class has all the project related constants*

*\*/*

*public final class AppConstants {*

*public static final String URL = "https://jsonplaceholder.typicode.com/todos/";*

*}*

* Let us understand this code. If you see the **ReadJsonServiceImpl.java**closely, you will see there is **a @Component**annotation. This annotation signifies that the given class is a component in AEM and the property **service = ReadJsonService.class**signifies that this is a service that is exposed via the **ReadJsonService** interface.

**OSGi Components and Services**

A service is an object that is registered in the OSGi Service Registry and can be looked up using its interface name(s). The only prerequisite is that a service should implement some interface. For example, I could register a runnable object under the java.lang.Runnable interface and clients could look it up using that interface name.

A "component" tends to be an object whose lifecycle is managed, usually by a component framework such as Declarative Services (DS), Blueprint or iPOJO.

A component may have any of the following features, in combination or alone:

1. A component may be started and stopped; this would be considered an "active" component, though that is also an informal term. A component that doesn't need to be started or stopped is called passive.
2. A component may publish itself as an OSGi service.
3. A component may bind to or consume OSGi services.
4. In general, using a component framework is the easiest way to work with OSGi services because the framework will manage the binding to the services that you want to consume. For example, you could say that your component "depends on" a particular service, in which case the component will only be created and activated when that service is available -- and also it will be destroyed when the service becomes unavailable.

* Once the Java code is done, we can now deploy the code using eclipse and maven.
* Now go to the AEM instance and in CRXDE, you should be seeing your project under the **/apps**folder.

|  |
| --- |
| [aem19](https://3.bp.blogspot.com/-uQ7GmKR1dkU/W7zndsoNz0I/AAAAAAAACSc/VSFO9bPF8oEa7Ulco1XkzzxpohucprHsgCLcBGAs/s1600/aem19.png) |
| Demo project |

* Right-click on **/apps/<your-project>/content/**, then **Create...**and then **Create Component...**
* In the dialog, fill the values as shown in the below screenshot

|  |
| --- |
| [aem20](https://4.bp.blogspot.com/-9XwfQHnPbyM/W7zoZJcyP-I/AAAAAAAACSk/s8dINUSeMq4xJss3QSIi2l7mZBGMPJQjgCLcBGAs/s1600/aem20.png) |
| Create a Component |

* Click **Next**, then **OK**
* Go to **/apps/<your-project>/components/content/sampleComponent**and open the **sampleComponent.jsp** file and add the below code and save. Note: We are using JSP only for simplicity. In the latest versions of AEM, we should use HTL (formerly called Sightly) and the use of JSP is discouraged.

*<%@include file="/libs/foundation/global.jsp"%>*

*<% org.redquark.demo.core.services.ReadJsonService service = sling.getService(org.redquark.demo.core.services.ReadJsonService.class);*

*String result = service.getData();*

*%>*

*<h2>This page invokes the AEM ReadJsonService</h2>*

*<h3>RESPONSE: <%=result%></h3>*

* Add the sample component on a page and you should be able to see the JSON string returned on that page as follows

*[{*

*"userId": 1,*

*"id": 1,*

*"title": "delectus aut autem",*

*"completed": false*

*},*

*{*

*"userId": 1,*

*"id": 2,*

*"title": "quis ut nam facilis et officia qui",*

*"completed": false*

*},*

*{*

*"userId": 1,*

*"id": 3,*

*"title": "fugiat veniam minus",*

*"completed": false*

*},*

*{*

*"userId": 1,*

*"id": 4,*

*"title": "et porro tempora",*

*"completed": true*

*}, ...*

*]*

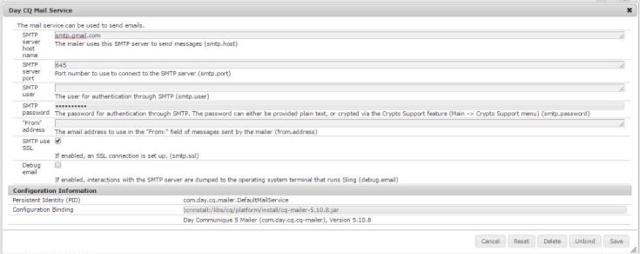
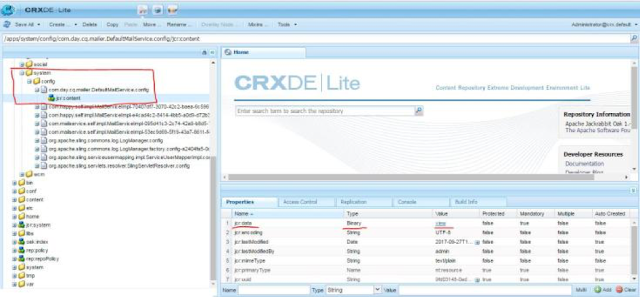
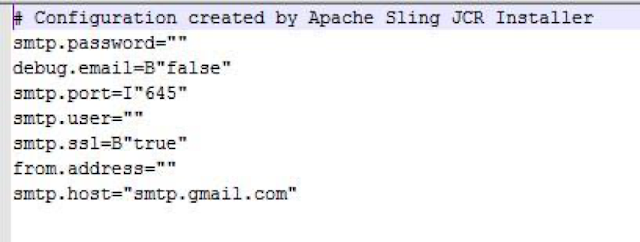
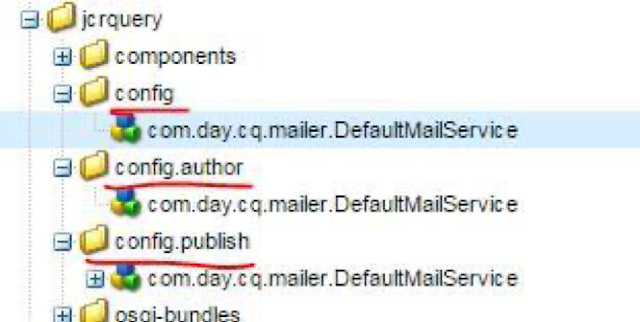
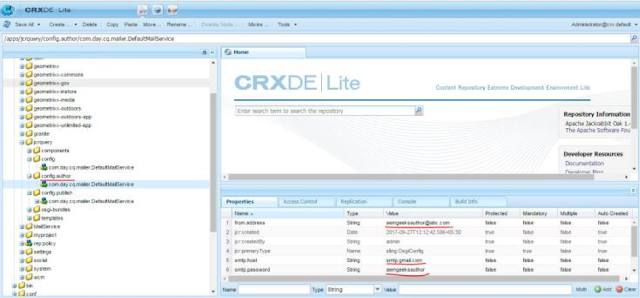
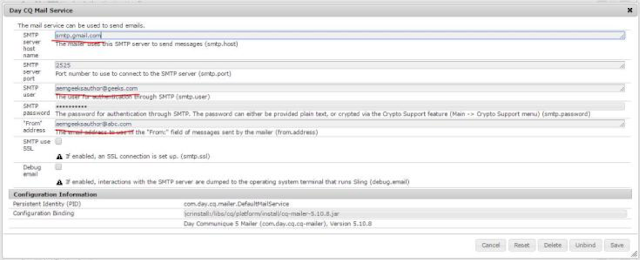
* You can check if your component and service are installed in the AEM server at <http://localhost:4502/system/console/components/> and <http://localhost:4502/system/console/services/> respectively.

**Conclusion**

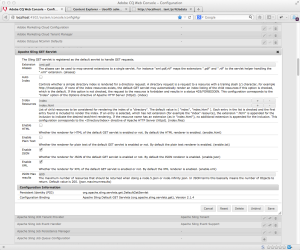
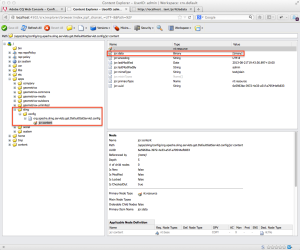
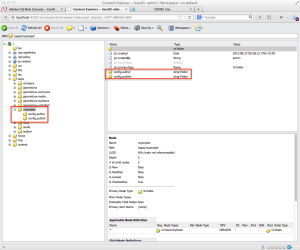
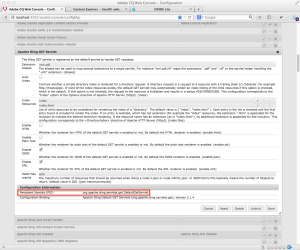
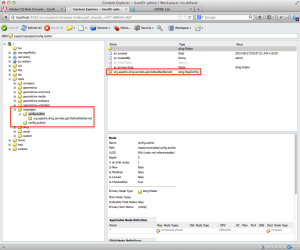
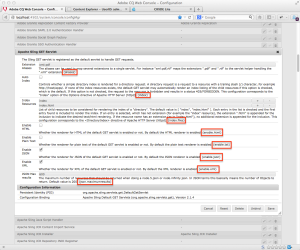
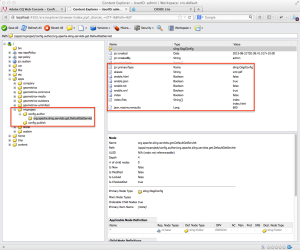
Congratulations!! you have created your first bundle with a component and service and deployed it in the AEM server. We have battled with the custom code in AEM and on the same lines we can create multiple components and services in one bundle.

You can find the complete code of this project on my [GitHub](https://github.com/ani03sha/aem-demo). Feel free to fork or open issues, if any.

# Managing OSGI bundle configurations using content-nodes in the repository

OSGi is used to control the configurations of bundles in AEM. OSGi allows easy management of bundles as the bundles can be started, stopped and installed at any time. The inter-dependencies between bundles are automatically handled.  
There are majorly two ways to configure the bundles in AEM. Using,  
· Adobe CQ Web Console  
· Content nodes in Repository  
But, I would suggest never using the CQ web console to configure the bundles in AEM. Rather, always use [make it a habit] to use content-nodes [sling:osgiconfig] in the repository to configure the bundles in AEM.  
Now the question would arise like, why one should not use CQ web console to configure bundles?  
Below are some of my observations, while accessing the values of the bundle parameters configured via the CQ web console.  
· Every value configured in the CQ web console gets saved in the repository itself.  
· If you configure bundles using the CQ web console, the configurations are saved under app/sling/config or apps/system/config.  
· We cannot control the location where this node will be saved.  
· The node will always be saved under config [sling:Folder].  
· As it is saved under the config folder and not under config.author/config.publish folder, the configurations will load for all run modes. It will not be specific for an author or publish mode.  
· Actual configuration data gets stored inside jcr:content node as a binary file.  
· New configuration node will be created under config[sling:folder] as <pid>.config [nt:file]  
· The actual configuration data is stored in jcr:content node’s properties. A jcr:data property will store configurations in binary format.  
Example:  
Day CQ Mail Service bundle configurations are changed from CQ web console [https://localhost:4502/system/console/configMgr](https://www.blogger.com/)  
[](https://www.blogger.com/)  
  
  
In [CRXDE](https://www.blogger.com/), configurations are reflected under apps/system/config folder [after refresh]. Actual configuration is stored under jcr:data property as binary data file.  
[](https://www.blogger.com/)  
  
  
[](https://www.blogger.com/)  
  
  
Keeping all the above points in mind, when you try to access the configuration parameters programmatically, you have to  
· Search for specific configuration using the <pid> of the bundle in apps/system/config and apps/sling/config, as you are not sure about its location.  
· Then write a code to access and read the binary data file from jcr: data property of the jcr:content node under config.  
· Once accessed, manipulate and adjust the desired parameter in binary file programmatically  
· And then again write back the values to jcr: data of the node.  
This seems to be a very complex task to achieve.  
  
  
Majority of the above mentioned challenges can be overcome by configuring the bundles using content-nodes (sling:osgiConfig) in the repository.  
Where,  
· Every value configured in CQ web console can be saved in the repository under folder specified by the developer  
· You can save the configuration nodes under your project folder (i.e. apps/<myproject>/ config)  
· Now we have full control of the location where this node will be saved.  
· The configurations can be Run mode-specific.  
· As we have the flexibility to save under config.author/config.publish folder, the configurations can be run mode specific now. Configurations under config.author will load for author instance and configurations under config.publish will load for the author instance.  
· Actual configuration data gets stored as properties of [sling:osgiconfig] node.  
· Configuration node needs to be created under a folder of type [sling:folder] as node of type [sling:osgiconfig].  
· The actual configuration data is stored in properties under [sling:osgiconfig] which can be accessed easily  
Example:  
1. Three configurations are added.  
· config:- apply for all instances.  
· config.author:- apply for the author's instance.  
· config.publish:- apply for publish instance.  
[](https://www.blogger.com/)  
  
  
2. Configurations added as properties for config.author instance  
[](https://www.blogger.com/)  
  
  
3. Once we save the new configurations in [CRXDE](https://www.blogger.com/). We need to refresh the [CQ web console](https://www.blogger.com/) to see the changes.  
[](https://www.blogger.com/)  
  
  
If the developer wants to manage the configurations [content-nodes (sling:osgiConfig) in the repository] programmatically, it will be an easy task for a developer to get access to properties of a node using Sling API or JCR API. Developers can easily access, manipulate and save the properties of a configuration done using content-nodes.

# AEM Best Practice: OSGi Configurations

One of the best practices I’ve come across recently in AEM is to never use the OSGi console directly to edit configurations. People may argue “But that’s what it’s there for!”, and indeed it is an easy interface to use for configuration changes. However there are drawbacks to managing your configuration changes this way in a AEM project.  
  
Let’s demonstrate this by changing the configuration of the Apache Sling GET Servlet.  
  
[](https://kristiancoding.files.wordpress.com/2013/08/slinggetservlet.png)  
  
When you edit a configuration in the OSGi console and save it, the updated configuration is saved to the JCR, generally under /apps/sling/config or /apps/system/config depending on the component you configure. First thing to note here is that you have no control over where this node is saved to, or if this configuration should be for a specific runmode – the parent folder is called config, not config.author or config.publish (for more information on runmodes, see [How to set up run mode](http://helpx.adobe.com/cq/kb/RunModeSetUp.html)).  
  
[](https://kristiancoding.files.wordpress.com/2013/08/contentexplorerfile.png)  
  
Secondly, and more importantly, the configuration node that has been created is of type nt:file. Therefore it contains a child called jcr:content, which has a property called jcr:data which contains the binary contents of the file. So as you can see (or more accurately NOT see), the details of the configuration itself are not readily apparent. When viewing this binary, we see the following:  
  
aliases=”xml:pdf”  
index=B”false”  
enable.txt=B”false”  
index.files=[“index”,”index.html”]  
enable.html=B”false”  
enable.json=B”true”  
enable.xml=B”true”  
json.maximumresults=I”600″  
  
Why is this bad? What happens when you want to change a single configuration value here programatically, or even through CRXDE lite, CRX Explorer, or curl for that matter? You would need to write code that extracts the binary contents of the file, manipulate this content to adjust the values to the desired configuration, then write this back to the jcr:data property of the file contents. This is quite messy, if all you want to do is change a single value of a complex configuration and leave the rest the same.  
  
So what’s the alternative? Create the configuration node manually, and use properties for each of the configuration settings. The configuration screen in the OSGi console gives you virtually all the information you need to create this node yourself. In addition, you also have the ability to choose the location of these settings, and therefore you can store them under you project branch, and also control the runmode in which they are used. This can be done in 3 simple steps:  
1 – Create the folder structure  
  
Create folders /apps/myproject/config.author and /apps/myproject/config.publish. Make sure the config.author and config.publish folders are of type sling:Folder  
  
[](https://kristiancoding.files.wordpress.com/2013/08/contentexplorerconfigwithrunmodes.png)  
2 – Create the configuration node  
  
Create a node of type sling:OsgiConfig under the runtime folder you wish to configure. The name of the node should be the same name as the Persistent Identity (PID) of the configuration in the OSGi Console. HINT: If this is a factory class and can have multiple instances, use the Factory Persistent Identifier (Factory PID) and append the name with ‘-xxxxx’, where xxxxx is a unique identifier.  
  
[](https://kristiancoding.files.wordpress.com/2013/08/slinggetservletpid.png)  
  
[](https://kristiancoding.files.wordpress.com/2013/08/contentexplorerauthorconfig.png)  
3 – Add the names and values from the OSGi configuration as properties  
  
Using the label in brackets at the end of the configuration description as the property name, and the value of the configuration as the values of the properties.  
  
[](https://kristiancoding.files.wordpress.com/2013/08/slinggetservletproperties.png)  
  
[](https://kristiancoding.files.wordpress.com/2013/08/contentexplorerauthorconfigwithproperties.png)  
  
That is it! Now, if you change the values of the configuration nodes’ properties, you can refresh the OSGi console and see these changes reflected to prove the configuration is taking effect. So from now on, whenever you need to change a configuration, you can make the change directly in the JCR, as well as easily make any configuration changes via curl or java, without needing to deal with file binaries.

# Running and debugging CQ5 using CRXDE

The problem you want to debug your code on a CQ5 server using Breakpoints in CRXDE, but the Debug option is greyed out.  
  
The SolutionChances are, you are not running CQ5 in debug mode. Here is a simple way to start your CQ5 instance in debug mode, and how to attach CRXDE to it.  
  
Assuming a Windows machine and a local copy of CQ5 for the purposes of this tutorial, located at c:\cq5\author running on the default port 4502.  
  
1. Ensure that all local instances of CQ5 are shut down.  
  
2. Under windows. open up command from the start menu, navigate to the c:\cq5\author directory  
  
3. copy and paste the following command:  
  
java -debug -XX:MaxPermSize=256M -Xnoagent -Xmx1024M -Djava.compiler=NONE -Xrunjdwp:transport=dt\_socket,server=y,suspend=n,address=30303 -jar cq5-author-4502.jar -nofork  
  
4. IMPORTANT: It can take quite a while for the server to start in debug mode. Wait until your browser window opens up at the author's login screen before continuing.  
  
5. Open up CRXDE. Search for it in your supplied CQ5 materials.  
  
6. Enter http://localhost:4502 in the URL field, and admin/admin for the username and password.  
  
7. Click on OK, and wait for CRXDE to load.  
  
8. To test, we can navigate to a piece of code that you can break into.  
  
A good place to test is to put a breakpoint on line 31 in /apps/geometrix/components/title/title.jsp  
  
When you have put the breakpoint in, navigate to:  
http://localhost:4502/cf#/content/geometrixx/en/products.html  
  
You will now be able to step through your code.  
  
**No code complete in CRXDE**  
The Problem In CQ5.5 SP1, CRXDE does not have code complete, classes can't be found, the laptop is in serious risk of defenestration.  
  
The Solution Go to Package Share in CQ5 admin on the author.  
  
<http://localhost:4502/crx/packageshare/>  
  
Log in if required with your adobe id.  
  
**Click on Download for:**  
1. Missing CRXDE-Libs  
2. CRX Hotfix Pack  
  
And install these packages.  
  
Restart CQ5 and CRXDE, and hopefully, this will have resolved the issue.

# How to Configure an OSGi Service before Starting an AEM Instance

1. Unpack AEM by running the following command.
2. java –jar cq-quickstart-6.jar -unpack
3. Create a folder named crx-quickstart\install in the installation directory.
4. Create a file with the name <service pid>.cfg. [From AEM 6.0 this should be .config] For example – org.apache.jackrabbit.oak.plugins.segment.SegmentNodeStoreService.cfg in the newly created folder.
5. Edit the file and set the configuration options line by line.
6. For example – tarmk.size=256MB

# OSGi Design Patterns

The trail on OSGi design patterns.  
This trail provides a set of application-specific OSGi design patterns along with basics for easy understanding. The sample source code given is implemented using AEM 6.x running on Java 8. Note that AEM uses Apache Felix OSGi container with the Declarative Services component model.  
  
**Basics**

* [OSGi – A simplistic view](http://www.computepatterns.com/116/what-is-osgi/)
* [OSGi dynamic component model](http://www.computepatterns.com/129/dynamic-component-programming-model-in-osgi/)
* [Component Vs Services](http://www.computepatterns.com/76/osgi-component-vs-service-in-aem/)
* Slicing a bundle – Sneak into what’s inside.
* Hiding classes within bundles.
* Bundles of multiple versions co-existing.
* Fragment.

**Application Design Patterns**

* Service publisher and service consumer.
* Components which requires mandatory configuration object.
* [Service listener pattern.](http://www.computepatterns.com/89/service-listener-osgi-pattern-in-aem/)
* Filtered service listener.
* Factory pattern.
* Config factory pattern.
* Extender pattern.
* Service deployment.
* Event admin
* Wire admin
* Named dynamic extension pattern (Query builder case).
* Dynamic implementation selection of a service.
* [Overriding OOTB servlet in AEM.](http://www.computepatterns.com/153/overriding-out-of-the-box-servlet-in-aem-sling/)

Source code for a [sample implementation of OSGi design patterns](https://github.com/Sivaramvt/ComputePatterns-AEM/tree/master/bundle/src/main/java/com/computepatterns/apppatterns/osgi) used in this trail is available in github.  
  
**Reference**

* OSGi Spec – Link [here](http://www.osgi.org/Specifications/HomePage).
* Apache Felix. Link [here](http://felix.apache.org/).
* Apache Felix SCR annotations – Link [here](http://felix.apache.org/documentation/subprojects/apache-felix-maven-scr-plugin/scr-annotations.html).
* Handy book on OSGi. Link [here](http://njbartlett.name/osgibook.html).

# OSGi – A Simplistic View

**OSGi**  
OSGi is a specification describing dynamic modular system for java platform. Essentially, two key aspects here – dynamic modules called bundles in OSGi and the OSGi runtime facilitating these bundles.  
More details on OSGi can be found [here](http://www.osgi.org/Main/HomePage).  
  
**Dynamic modules – Bundles**

* Bundle represents an unit of deployment that implements highly cohesive, loosely coupled functionality.
* Bundle explicitly states its dependencies.
* Bundles are versioned.
* Bundles allow hiding information at package level.
* Classpath per bundle.
* Each bundle has a name called ‘symbolic name’ for its unique identity.

**Dynamic modular system – OSGi runtime.**

* Allows bundles to be installed, started, stopped, configured at run time.
* Allows bundle to collaborate with each other to meet its dependencies.
* Bundle with different versions can co-exist at run time. Solves [JAR hell](http://en.wikipedia.org/wiki/Java_Classloader#JAR_hell).
* Classloader per bundle.

**A glimpse of goodness we gain with this dynamic modular setup in OSGi.**  
You can think of your application as a set of modules. Each module can be installed, started, stopped, configured at the runtime without any restart. If one of your module fails, only the dependent modules will be impacted; rest of your application can work fine. Your module could refer multiple versions of the dependent module. You’ve the options to hide your implementations within the module. All these are feasible with OSGi dynamic module setup.  
  
To make all these possible, OSGi defines [lifecycle of bundles](http://en.wikipedia.org/wiki/OSGi#Life-cycle) and goes with late binding (runtime binding). The need for late binding for meeting dependencies at runtime spawns the dynamic component model in OSGi.

# OSGi Component vs Service in AEM

AEM ships with an OSGi container Apache felix that implements Declarative Services (DS) component model. Most of the developers who are new to AEM often gets confused between OSGi components and services.  
  
**OSGi Component**  
If you want the life of your object to be managed by the OSGi container, you should declare it as a component. Using DS annotations, you could make a POJO a OSGi component by annotating it with@Component With this, you will get the ability to start, stop and configure the component using the felix web console.  
  
**OSGi Service**  
OSGi components can be made as OSGi service by marking it with @Service annotation. All it mandates is that an interface – Services should implement an interface (1 or more). When you mark a component as service, you could refer (call) this service from other osgi components.  
  
**OSGi Component Vs Service**  
All objects managed by OSGi container are components. You qualify components as services. This means that all services are components but not vice-versa.

Components can refer/call (using container injection – @Reference) other services but not components. In other words, a component cannot be injected into another component / service. Only services can be injected into another component.  
  
**OSGi Component Example**  
We have a weather printing component that creates a thread when activated (started) and calls the weather service (an OSGi service) to pull the weather details from the yahoo server.  
  
*package com.computepatterns.apppatterns.osgi;  
  
import java.io.IOException;  
import java.util.Map;  
  
import org.apache.felix.scr.annotations.Activate;  
import org.apache.felix.scr.annotations.Component;  
import org.apache.felix.scr.annotations.Reference;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@Component(  
label = "Compute Patterns - Weather details printing osgi component.",  
description = "Sample OSGi component that uses thread and a OSGi service to print weather details in log.",  
immediate = true)  
public class WeatherPrintingComponent {  
private static final Logger log = LoggerFactory.getLogger(WeatherPrintingComponent.class);  
  
/\* Yahoo weather api end point \*/  
private static final String weatherApiEndpoint =  
"http://weather.yahooapis.com/forecastrss?p=80020&u=f";  
  
@Reference  
WeatherService weatherService;  
  
@Activate  
protected void activate(Map<String, String> config) {  
log.info("Weather printing component - activiated");  
// Set up a thread which wakes up every 5s and make a make a service call to fetch weather info  
// and print it in the log.  
Runnable task = () -> {  
try {  
while (!Thread.currentThread().isInterrupted()) {  
Thread.sleep(5000);  
try {  
log.info(weatherService.getWeatherFeed(weatherApiEndpoint));  
} catch (IOException e) {  
log.error("Unable to get weather details.", e);  
}  
}  
} catch (InterruptedException e) {  
log.error("Weather printing thread interrupted", e);  
}  
};  
  
Thread weatherThread = new Thread(task);  
weatherThread.setName("Compute Patterns - Weather printing");  
weatherThread.start();  
}  
}*  
  
WeatherPrintingComponent class has been marked with @Component annotation (line# 12). A service WeatherService is injected into the component (line #.24). Remember that a component can refer to other services.  
Here’s the source code for the WeatherService interface and its implementation.  
  
*package com.computepatterns.apppatterns.osgi;  
  
import java.io.IOException;  
  
/\*\*  
\* Service to provide weather details.  
\*/  
public interface WeatherService {  
  
/\*\*  
\* Get weather feed using given endpoint.  
\*  
\* @param apiEndPoint Url endpoint to hit and get the weather feed. Example - Yahoo weather end  
\* point.  
\* @return Weather feed in xml format.  
\* @throws IOException Exception in connecting to the url.  
\*/  
String getWeatherFeed(String apiEndPoint) throws IOException;  
}  
  
  
package com.computepatterns.apppatterns.osgi.impl;  
  
import java.io.IOException;  
import java.util.Map;  
  
import org.apache.felix.scr.annotations.Activate;  
import org.apache.felix.scr.annotations.Component;  
import org.apache.felix.scr.annotations.Service;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.apache.commons.httpclient.HttpClient;  
import org.apache.commons.httpclient.HttpStatus;  
import org.apache.commons.httpclient.methods.GetMethod;  
import org.apache.commons.lang.StringUtils;  
  
import com.computepatterns.apppatterns.osgi.WeatherService;  
  
/\*\*  
\* Weather service implementation. Connects to the provided end point using apache http client to  
\* fetch the feed.  
\*  
\*/  
@Component(label = "Compute Patterns - Weather Service.",  
description = "Connects to the weather apis and fetches weather details.")  
@Service  
public class WeatherServiceImpl implements WeatherService {  
private static final Logger log = LoggerFactory.getLogger(WeatherServiceImpl.class);  
  
@Activate  
protected void activate(Map<String, String> config) {  
log.info("Weather Service - ACTIVATED");  
}  
  
@Override  
public String getWeatherFeed(String apiEndPoint) throws IOException {  
// Sanity check the arguments.  
if (StringUtils.isBlank(apiEndPoint)) {  
return StringUtils.EMPTY;  
}  
// Create a http client and hit the server.  
HttpClient httpClient = new HttpClient();  
GetMethod httpMethod = new GetMethod(apiEndPoint);  
// Return the response body if the request is successfully executed.  
if (httpClient.executeMethod(httpMethod) == HttpStatus.SC\_OK) {  
log.trace("Successfully fetched data from the endpoint.");  
return httpMethod.getResponseBodyAsString();  
}  
log.trace("Connection not successful.");  
return StringUtils.EMPTY;  
}  
}*  
  
**OSGi Component – Use cases**  
Now, you must be wondering, why you shouldn’t be marking every component as service. Yes, you want your objects to be used outside its body. However, in certain use-cases modeling your object as component (not marking it as service) makes sense. Here are a few of them.

* A server object in your application which listens to a socket.
* A filter object which intercepts the requests.
* An object which monitors a resource and report.
* An Objects which pulls data from the external system and writes to the repository.

# Dynamic Component Programming Model In OSGI

In OSGi deployment model for java, bundles come and go at run time. Hence, dependencies between the bundles would have to be resolved at run time. This would lead to inconsistency as programs cannot be written with confidence referring to types from other bundles. Hence a dynamic component programming model has been introduced for bundle developers to addresses the late binding of dependencies.  
  
**The dynamic component programming model**

* Introduced the notion of component and service.
* The mechanism to bind services at run time.

**Components**

* A component is a POJO with additional meta information and managed by the OSGi container. With plugin such as [Apache Felix SCR annotations](http://felix.apache.org/documentation/subprojects/apache-felix-maven-scr-plugin/scr-annotations.html#service), all these metadata can be added via annotations which get converted into required XMLs at build time.
* Every component has a unique name which is also its PID (Persistence Identifier).
* The component has a lifecycle and it can be started or stopped.
* Supports extending model – Component can be made abstract. The component can inherit other components.
* A component can be declared as a factory object. Meaning, Any number of instances of that component can be made at programmers will.
* The runtime can be instructed to activate a component immediately or lazily (delayed – on the first invocation)
* A component can be marked as configuration factory. A new instance could be created for every configuration supplied.
* A component can be made to be active only when a configuration object is present.
* A component may have multiple properties (including configurations). Property can be single-valued or multivalued.

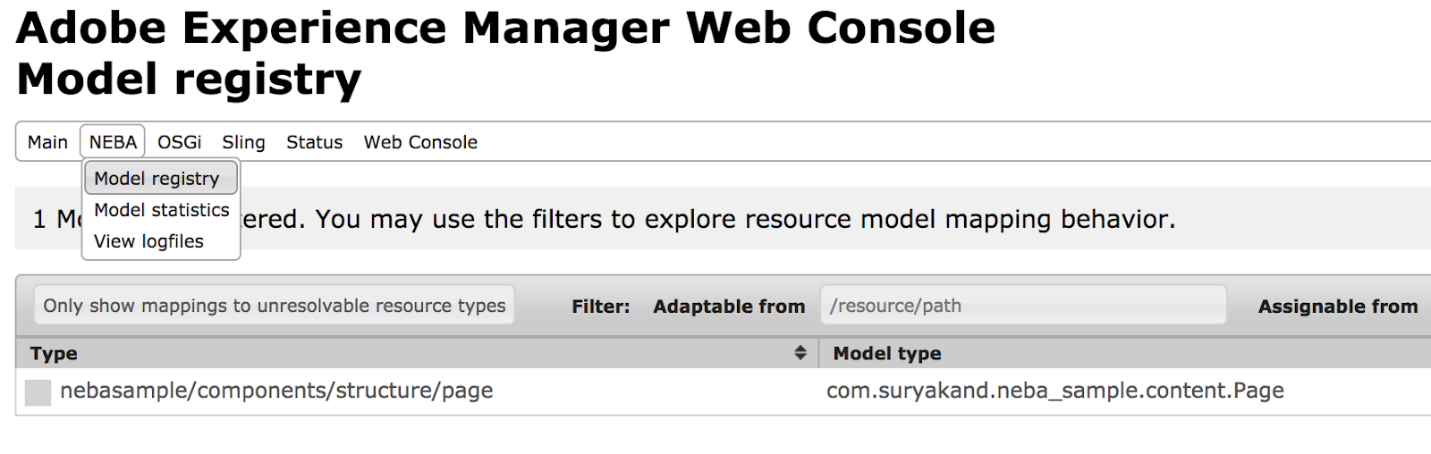
**Services**

* Services are special components that can be injected into other components.
* In addition to the mechanics of components, service needs an interface. Hence, interfaces are decoupled from the implementations.
* Interfaces are used to bind services. The selection of the right implementation for the given service can be deferred to run time.
* Services can be injected into components. Using the component property as filters, target filters for the selection of the right implementation can be defined.
* An implementation can be registered as a service against multiple interfaces.
* Services are usually singletons. So, thread safety for the mutable state needs to be taken care of by the bundle developers.
* A service can be marked as a service factory. The service factory creates a new instance for every bundle which uses the service.

# Create a OSGI Configuration Listener in AEM 6.1

* Following article demonstrate business scenario where you need to execute a job, call a method or simply log when there in change in XYZ OSGI configuration. Taking forward the [OSGI Configuration Management](https://aem.adobemarketingclub.com/osgi-configuration-management-aem/) and [Reading of OSGI Configuration](https://aem.adobemarketingclub.com/reading-osgi-configuration-properties/), I will show you how you can create a OSGI Configuration listener. Before that’s take a look at below…  
    
  ConfigurationListener[Interface] objects are registered with the Framework service registry and are notified with a ConfigurationEvent object when an event is fired. ConfigurationListener objects can inspect the received ConfigurationEvent object to determine its type, the pid of the Configuration object with which it is associated, and the Configuration Admin service that fired the event.  
    
  I have used *com.day.cq.wcm.notification.email.impl.EmailChannel* service but you can use any service based on your requirements and will read property email.from when their is a change. Let’s get started..  
    
  **1. Create a Service which Implement *ConfigurationListener* interface**
* *@Component*
* *@Service*
* *public class OSGIConfigurationListener implements ConfigurationListener { }*  
    
  **2. Override *ConfigurationEvent* method and its implementation**
* *@Override*
* *public void configurationEvent(ConfigurationEvent event) {*
* *try {*
* *if (event.getPid().equals(OSGI\_SERVICE\_EMAIl\_CHANNEL)) {*
* *readUpdatedConfig();*
* *}*
* *} catch (Exception exp) {*
* *log.info("Exception found while listening for change on {}, {} ", OSGI\_SERVICE\_EMAIl\_CHANNEL, exp.getMessage());*
* *}*
* *}*
* **3. Invoke a custom define method to read required property**
* */\*\**
* *\* Method will be executed once there is any change in the property of*
* *\* <code> com.day.cq.widget.impl.HtmlLibraryManagerImpl</code> service*
* *\*/*
* *public void readUpdatedConfig() {*
* *try {*
* *Configuration readConfig = configurationAdmin.getConfiguration(OSGI\_SERVICE\_EMAIl\_CHANNEL);*
* *Dictionary < String, String > properties = readConfig.getProperties();*
* *log.info("Updated property is " + properties.get(EMAIL\_FROM));*
* *} catch (IOException e) {*
* *log.info("Exception while reading property of {}, {}", EMAIL\_FROM, e.getMessage());*
* *}*
* *}*
* **4. Change the *email.from* property in service***com.day.cq.wcm.notification.email.impl.EmailChannel*  
    
    
  **5. See the updated log file snippet**  
    
    
    
  **6. Here is complete code you can use**
* *package com.adobeaemclub.adobeaemclub.core.services;*
* *import org.apache.felix.scr.annotations.Component;*
* *import org.apache.felix.scr.annotations.Reference;*
* *import org.apache.felix.scr.annotations.Service;*
* *import org.osgi.service.cm.Configuration;*
* *import org.osgi.service.cm.ConfigurationEvent;*
* *import org.osgi.service.cm.ConfigurationListener;*
* *import org.osgi.service.cm.ConfigurationAdmin;*
* *import org.slf4j.Logger;*
* *import org.slf4j.LoggerFactory;*
* *import java.io.IOException;*
* *import java.util.Dictionary;*
* */\*\**
* *\* Service will act as a listener for any configuration change and will show update property.*
* *\* @author praveen*
* *\**
* *\*/*
* *@Component*
* *@Service*
* *public class OSGIConfigurationListener implements ConfigurationListener {*
* *public static final Logger log = LoggerFactory.getLogger(OSGIConfigurationListener.class);*
* *public final static String OSGI\_SERVICE\_EMAIl\_CHANNEL = "com.day.cq.wcm.notification.email.impl.EmailChannel";*
* *public final static String EMAIL\_FROM = "email.from";*
* *@Reference*
* *ConfigurationAdmin configurationAdmin;*
* *@Override*
* *public void configurationEvent(ConfigurationEvent event) {*
* *try {*
* *if (event.getPid().equals(OSGI\_SERVICE\_EMAIl\_CHANNEL)) {*
* *readUpdatedConfig();*
* *}*
* *} catch (Exception exp) {*
* *log.info("Exception found while listening for change on {}, {} ", OSGI\_SERVICE\_EMAIl\_CHANNEL, exp.getMessage());*
* *}*
* *}*
* */\*\**
* *\* Method will be executed once there is any change in the property of*
* *\* <code> com.day.cq.widget.impl.HtmlLibraryManagerImpl</code> service*
* *\*/*
* *public void readUpdatedConfig() {*
* *try {*
* *Configuration readConfig = configurationAdmin.getConfiguration(OSGI\_SERVICE\_EMAIl\_CHANNEL);*
* *Dictionary < String, String > properties = readConfig.getProperties();*
* *log.info("Updated property is " + properties.get(EMAIL\_FROM));*
* *} catch (IOException e) {*
* *log.info("Exception while reading property of {}, {}", EMAIL\_FROM, e.getMessage());*
* *}*
* *}*
* *}*

# AEM 6.3 - Bundle Whitelisting - Deprecation of administrative authentication

I stumbled on an issue when I was using AEM 6.3. I created few neba ResourceModels and when I tried to access neba Model Registry, I got an error *(java.lang.IllegalStateException: org.apache.sling.api.resource.LoginException: Bundle org.eclipse.gemini.blueprint.extender is NOT whitelisted):*  
  
Image: neba Model Registry Menu  
  
  
  
Image: Error Screen  
  
**NOTE:** Neba team has already fixed it on their development branch and we don’t need to explicitly add whitelisting configuration for neba bundle.  
  
**Here is the reason for the error**  
Originally the ResourceResolverFactory.getAdministrativeResourceResolver and SlingRepository.loginAdministrative methods have been defined to provide access to the resource tree and JCR Repository. These methods proved to be inappropriate because they allow for much too broad access.  
Consequently, these methods are being deprecated and will be removed in future releases of the service implementations.  
The following methods are deprecated:

* ResourceResolverFactory.getAdministrativeResourceResolver
* ResourceProviderFactory.getAdministrativeResourceProvider
* SlingRepository.loginAdministrative

The implementations we have in Sling's bundle will remain implemented in the near future. But there will be a configuration switch to disable support for these methods: If the method is disabled, a LoginException is always thrown from these methods. The JavaDoc of the methods is extended with this information.

**Whitelisting bundles for administrative login**  
In order to be able to manage a few (hopefully legit) uses of the above-deprecated methods, a whitelisting mechanism was introduced with [SLING-5153](https://www.blogger.com/) (JCR Base 2.4.2).  
The recommended way to whitelist a bundle for administrative login is via a whitelist fragment configuration. It can be created as an OSGi factory configuration with the factoryPID org.apache.sling.jcr.base.internal.LoginAdminWhitelist.fragment.  
  
E.g. a typical configuration file might be called org.apache.sling.jcr.base.internal.LoginAdminWhitelist.fragment-myapp.config and could look as follows:  
  
*whitelist.name="myapp"  
whitelist.bundles=[  
"com.myapp.core",  
"com.myapp.commons"  
]*  
  
In general, try to avoid using administrative login if you are writing code in your bundle but, let’s say you are using some third-party bundle then in that case you can add a configuration to get it working as explained above.

# CQ Development - OSGi bundles and Components

Recently one of my blog followers asked about how we can use the OSGi bundle (components and services) in CQ’s JSP/components. In this post, I am going to enlist a few key things about creating and accessing the OSGi bundle in CQ components.

So, a CQ component is nothing but a script file (either a JAVA or JSP) and the primary goal of a component is rending the markup but, a component may need to access OSGi services in order to execute some business logic that is part of OSGi bundle. I am going to use CRXDE (an eclipse flavored IDE for CQ development).

First of all, I’ll enlist the steps to access any component that you have written in a bundle and then I’ll explain it in detail.

**Steps:**

1) Create an OSGi bundle.

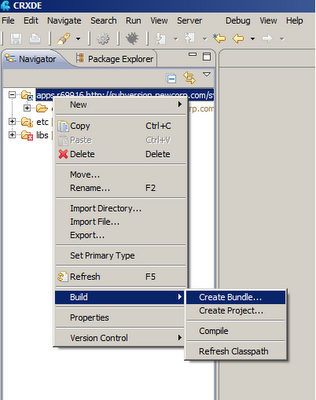
2) Create an OSGi service (using Felix/OSGi annotations).

3) Write a utility class to access the components that we have created in setp#2.

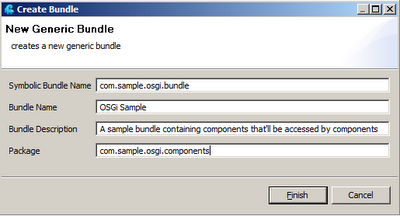
**Explanation:**

**1.** **Create an OSGi bundle**

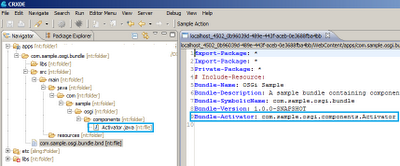
**1.A)** To create a bundle open up your CRXDE and right-click on “apps” folder ->Build -> Create Bundle (as shown in below) screenshot:

[](https://3.bp.blogspot.com/-tIznvUAHAEs/TnzWVaNTZEI/AAAAAAAAAMA/53F_NTx9EoA/s1600/create-bundle-step-1.png)

**1.B)** You’ll get a “Create bundle” pop-up, fill the details as shown below and hit “Finish” button:

[](https://3.bp.blogspot.com/-Skfqzvxaz1Y/TnzWdX9lRQI/AAAAAAAAAME/dNKG5ke-B-w/s1600/create-bundle-step-2.png)

**1.C)** Once the bundle is created successfully you’ll see following folder/package in your CRXDE IDE:

[](https://3.bp.blogspot.com/-3392Cj13ARg/TnzWjYyTcmI/AAAAAAAAAMI/hLGcVgLaBI8/s1600/create-bundle-step-3.png)

So, now we have a bundle with the directory structure (package) that we defined in step 1.B. In step#2 we’ll be creating a new class that we want to access from a component or other java files (within the same bundle or from other bundles).

**2.** **Create an OSGi service**

For this example, we are going to create a “FormattingService” that will be used for formatting dates.

**2.A)** Create an interface “FormattingService” in “com.sample.osgi.components” package.

***package****com.sample.osgi.components;*

***public******interface****FormattingService {*

***public****String getDateByInterval(String WMY, String dateFormat,****int****interval);*

*}*

**2.B)** Create and implementation class “FormattingServiceImpl” in same package  “com.sample.osgi.components”. We need to annotate this class (as shown below) in order to expose it as an OSGi service to other bundles and CQ components.

*package com.sample.osgi.components;*

*import java.text.DateFormat;*

*import java.text.SimpleDateFormat;*

*import java.util.Calendar;*

*import org.apache.commons.lang.StringUtils;*

*import org.apache.sling.commons.osgi.OsgiUtil;*

*import org.osgi.service.component.ComponentContext;*

*/\*\**

*\* A sample OSGi service class that will be used by other OSGi bundles or components*

*\* to get past or future date.*

*\**

*\* @scr.component immediate="false" label="Date formatting service"*

*\* description="An utility service to get past or future date"*

*\* name="com.sample.osgi.components.FormattingServiceImpl"*

*\* @scr.property name="date.format" label = "Expected date format"*

*\* description="Configuration to set expected date format"*

*\* @scr.service*

*\*/*

*public class FormattingServiceImpl implements FormattingService {*

*//this can be configured via OSGi configuration console using "default.date.section" property*

*private String dateFormat = null;*

*/\*\**

*\* This method will be invoked only once when the FormattingService is*

*\* intialized by OSGi container.*

*\* @param componentContext*

*\*/*

*protected void activate(ComponentContext componentContext) {*

*this.dateFormat = OsgiUtil.toString(*

*componentContext.getProperties().get("date.format"), "MM/dd/yyyy");*

*}*

*/\*\**

*\* Utility method to choose a future or back date*

*\* @param WMY W - Week, M - Month, Y - Year*

*\* @param interval integer value that represents future or past interval*

*\* @return String date*

*\*/*

*public String getDateByInterval(String WMY, String dateFormat, int interval) {*

*String formattedDate = "";*

*DateFormat expectedDateFormat = null;*

*Calendar calendar = Calendar.getInstance();*

*if(StringUtils.isBlank(dateFormat)) {*

*dateFormat = getDateFormat();*

*}*

*expectedDateFormat = new SimpleDateFormat(dateFormat);*

*if(StringUtils.equalsIgnoreCase(WMY, "W")){*

*calendar.add(Calendar.WEEK\_OF\_YEAR, interval);*

*} else if(StringUtils.equalsIgnoreCase(WMY, "M")){*

*calendar.add(Calendar.MONTH, interval);*

*} else if(StringUtils.equalsIgnoreCase(WMY, "Y")){*

*calendar.add(Calendar.YEAR, interval);*

*}*

*formattedDate = expectedDateFormat.format(calendar.getTime());*

*return formattedDate;*

*}*

*/\*\**

*\* @return the dateFormat*

*\*/*

*public String getDateFormat() {*

*return dateFormat;*

*}*

*/\*\**

*\* @param dateFormat the dateFormat to set*

*\*/*

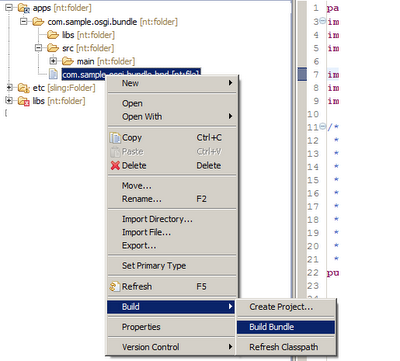
*public void setDateFormat(String dateFormat) {*

*this.dateFormat = dateFormat;*

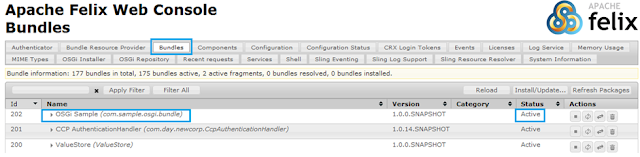
*}*

*}*

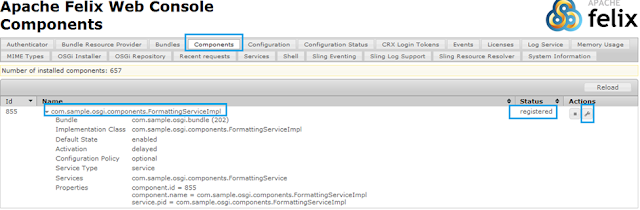
2.C) Build the bundle by right clicking on “com.sample.osgi.bundle.bnd” -> Build -> Build Bundle

[](https://1.bp.blogspot.com/-P0ilktGEBgo/TnzWtG123sI/AAAAAAAAAMM/ka6qr6xPoZ8/s1600/build-bundle.png)

Once the bundle is created go to (Felix web console) using the URL on your machine: <http://localhost:4502/system/console/bundles> and the bundle that we have built should be available in Felix console and it should be in ”Active” as shown below:

[](https://2.bp.blogspot.com/-WWvoOd0R8h8/TnzWzsC3t6I/AAAAAAAAAMQ/N2pObagFHCM/s1600/felix-bundle-console.png)

Now, go to the “**components**” tab as shown below, and you’ll see that the service “FormattingService” that we have written is registered with Felix OSGi container and is ready to consume:

[](https://4.bp.blogspot.com/-6-mJnlW_n5s/TnzW4Ty0y7I/AAAAAAAAAMU/z-2PbD0bEIU/s1600/felix-component-console.png)

**3.** **Code to access component/service from other java classes and components.**

In order to access the “FormattingService”, we’ll write a utility class with a static method so that the code to access service is wrapped inside it and we don’t need to write the same code again and again. This how the utility class should look like:

In order to access the “FormattingService”, we’ll write a utility class with a static method so that the code to access service is wrapped inside it and we don’t need to write the same code again and again. This how the utility class should look like:

*package com.sample.osgi.components;*

*import org.osgi.framework.BundleContext;*

*import org.osgi.framework.FrameworkUtil;*

*import com.newcorp.ccp.resolver.PortalResolver;*

*public class ComponentUtil {*

*/\*\**

*\* Utility method that will return an FormattingService instance from OSGi container*

*\* @return*

*\*/*

*public static FormattingService getFormattingService() {*

*BundleContext bundleContext = FrameworkUtil.getBundle(FormattingService.class).getBundleContext();*

*return (FormattingService)bundleContext.getService(*

*bundleContext.getServiceReference(FormattingService.class.getName()));*

*}*

*}*

That’s it! Now you can use this class to get an instance of your service from any CQ component or any other class in other OSGi bundles, here is sample code that you can write in your CQ component’s JSP:

*<%String formattedDate = ComponentUtil.getFormattingService().getDateByInterval("M", "MM-dd-YYY", -1);%>*

I hope this will help the folks who are working on CQ.

I have created this example just to explain the concept of how we can create OSGi services and access it from the CQ component. For simple things we should avoid creating services. We should create services only for those functionalities that fit into the OSGi definition.

**You can read more about OSGi annotations at:**

<http://felix.apache.org/site/apache-felix-maven-scr-plugin.html>

<http://felix.apache.org/site/scr-javadoc-tags.html>

<http://felix.apache.org/site/scr-annotations.html#SCRAnnotations-Component>

**Source:**<http://suryakand-shinde.blogspot.com/2011/09/cq-development-osgi-bundles-and.html>

# Creating your Custom OSGi configuration

We all know that AEM works on ***Apache Felix*** which is an implementation of ***OSGi***. OSGi provides a way to manage bundles and configurations.  
You can find all the Out of the box OSGi configurations at - http://<host>:<port>/system/console/configMgr. Apart from out of the box configurations, we can also create our custom configurations. In this post, we will be creating a custom out of the box configuration which reads the user input and gets the JSON response from a web service.

## Code in Action

* To make a custom OSGi configuration, we need to first create an interface whose public methods will represent the fields in the configuration.
* Create an interface named ***HttpConfiguration***and paste the following code in it.

*package org.redquark.demo.core.services;*

*import org.osgi.service.metatype.annotations.AttributeDefinition;*

*import org.osgi.service.metatype.annotations.AttributeType;*

*import org.osgi.service.metatype.annotations.ObjectClassDefinition;*

*import org.osgi.service.metatype.annotations.Option;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* This interface represents an OSGi configuration which can be found at -*

*\* ./system/console/configMgr*

*\*/*

*@ObjectClassDefinition(*

*name = "Http Configuration",*

*description = "This configuration reads the values to make an HTTP call to a JSON webservice")*

*public @interface HttpConfiguration {*

*/\*\**

*\* This is a checkbox property which will indicate of the configuration is*

*\* executed or not*

*\**

*\* @return {@link Boolean}*

*\*/*

*@AttributeDefinition(*

*name = "Enable config",*

*description = "This property indicates whether the configuration values will taken into account or not",*

*type = AttributeType.BOOLEAN)*

*public boolean enableConfig();*

*/\*\**

*\* This method returns the protocol that is being used*

*\**

*\* @return Protocol*

*\*/*

*@AttributeDefinition(*

*name = "Protocol",*

*description = "Choose Protocol",*

*options = {*

*@Option(label = "HTTP", value = "http"),*

*@Option(label = "HTTPS", value = "https")*

*})*

*public String getProtocol();*

*/\*\**

*\* Returns the server*

*\**

*\* @return {@link String}*

*\*/*

*@AttributeDefinition(*

*name = "Server",*

*description = "Enter the server name")*

*public String getServer();*

*/\*\**

*\* Returns the endpoint*

*\**

*\* @return {@link String}*

*\*/*

*@AttributeDefinition(*

*name = "Endpoint",*

*description = "Enter the endpoint")*

*public String getEndpoint();*

*}*

* This configuration has a checkbox ***Enable Config***, a drop-down ***Protocol***, two text fields ***Server,*** and ***Endpoint***.
* Now create an interface ***HttpService*** to make the http call as below

*package org.redquark.demo.core.services;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* This interface exposes the functionality of calling a JSON Web Service*

*\*/*

*public interface HttpService {*

*/\*\**

*\* This method makes the HTTP call on the given URL*

*\**

*\* @param url*

*\* @return {@link String}*

*\*/*

*public String makeHttpCall();*

*}*

* Create an implementation class HttpServiceImpl as below.

*package org.redquark.demo.core.services.impl;*

*import org.osgi.service.component.annotations.Activate;*

*import org.osgi.service.component.annotations.Component;*

*import org.osgi.service.metatype.annotations.Designate;*

*import org.redquark.demo.core.services.HttpConfiguration;*

*import org.redquark.demo.core.services.HttpService;*

*import org.redquark.demo.core.utils.Network;*

*import org.slf4j.Logger;*

*import org.slf4j.LoggerFactory;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* Implementation class of HttpService interface and this class reads values from the OSGi configuration as well*

*\*/*

*@Component(service = HttpService.class, immediate = true)*

*@Designate(ocd = HttpConfiguration.class)*

*public class HttpServiceImpl implements HttpService {*

*/\*\**

*\* Logger*

*\*/*

*private static final Logger log = LoggerFactory.getLogger(HttpServiceImpl.class);*

*/\*\**

*\* Instance of the OSGi configuration class*

*\*/*

*private HttpConfiguration configuration;*

*@Activate*

*protected void activate(HttpConfiguration configuration) {*

*this.configuration = configuration;*

*}*

*/\*\**

*\* Overridden method of the HttpService*

*\*/*

*@Override*

*public String makeHttpCall() {*

*log.info("----------< Reading the config values >----------");*

*try {*

*/\*\**

*\* Reading values from the configuration*

*\*/*

*boolean enable = configuration.enableConfig();*

*String protocol = configuration.getProtocol();*

*String server = configuration.getServer();*

*String endpoint = configuration.getEndpoint();*

*/\*\**

*\* Constructing the URL*

*\*/*

*String url = protocol + "://" + server + "/" + endpoint;*

*/\*\**

*\* Make HTTP call only if "enable" is true*

*\*/*

*if (enable) {*

*/\*\**

*\* Making the actual HTTP call*

*\*/*

*String response = Network.readJson(url);*

*/\*\**

*\* Printing the response in the logs*

*\*/*

*log.info("----------< JSON response from the webservice is >----------");*

*log.info(response);*

*return response;*

*} else {*

*log.info("----------< Configuration is not enabled >----------");*

*return "Configuration not enabled";*

*}*

*} catch (Exception e) {*

*log.error(e.getMessage(), e);*

*return "Error occurred" + e.getMessage();*

*}*

*}*

*}*

* This is an OSGi component in which we are reading values from the OSGi configuration. Notice that we are using @Designate annotation to link this class to the configuration.
* Now create a simple Sling Servlet to use this component as follows.

*package org.redquark.demo.core.servlets;*

*import javax.servlet.Servlet;*

*import org.apache.sling.api.SlingHttpServletRequest;*

*import org.apache.sling.api.SlingHttpServletResponse;*

*import org.apache.sling.api.servlets.HttpConstants;*

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

*import org.osgi.framework.Constants;*

*import org.osgi.service.component.annotations.Component;*

*import org.osgi.service.component.annotations.Reference;*

*import org.redquark.demo.core.services.HttpService;*

*import org.slf4j.Logger;*

*import org.slf4j.LoggerFactory;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* This method makes an HTTP call and read the value from the JSON webservice via an OSGi configuration*

*\**

*\*/*

*@Component(service = Servlet.class, property = {*

*Constants.SERVICE\_DESCRIPTION + "=HTTP servlet",*

*"sling.servlet.methods=" + HttpConstants.METHOD\_GET,*

*"sling.servlet.paths=" + "/bin/demo/httpcall"*

*})*

*public class HttpServlet extends SlingSafeMethodsServlet {*

*/\*\**

*\* Generated serialVersionUid*

*\*/*

*private static final long serialVersionUID = -2014397651676211439 L;*

*/\*\**

*\* Logger*

*\*/*

*private static final Logger log = LoggerFactory.getLogger(HttpServlet.class);*

*@Reference*

*private HttpService httpService;*

*/\*\**

*\* Overridden doGet() method*

*\*/*

*@Override*

*protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) {*

*try {*

*String jsonResponse = httpService.makeHttpCall();*

*/\*\**

*\* Printing the json response on the browser*

*\*/*

*response.getWriter().println(jsonResponse);*

*} catch (Exception e) {*

*log.error(e.getMessage(), e);*

*}*

*}*

*}*

* Go to the ***./system/console/configMgr*** and search for ***Http Configuration*** and open it and configure it accordingly. Then save.

|  |
| --- |
| [aem36](https://3.bp.blogspot.com/-3HI0sx8zULI/W8Rkc1eYgCI/AAAAAAAACWY/cBYl9vqxoCE-kvd1J1tBk4bLuJbtsJbwQCLcBGAs/s1600/aem36.png) |
| OSGi Configuration |

* Now hit the request - [http://<host>:<port>/bin/demo/httpcall](http://localhost:4502/bin/demo/httpcall) and you will get the JSON response as follows

[{"userId": 1, "id": 1, "title": "delectus aut autem", "completed": false}, {"userId": 1, "id": 2, "title": "quis ut nam facilis et officia qui", "completed": false}, {"userId": 1, "id": 3, "title": "fugiat veniam minus", "completed": false}, {"userId": 1, "id": 4, "title": "et porro tempora", "completed": true}, ...]

**Access to ResourceResolver in OSGi Services: AEM 6.1**

We all know that from AEM 6.0, usage of Admin Session to access the ResourceResolver is deprecated which means we cannot use session = repository.loginAdministrative(null); anymore !  
  
Instead, AEM comes with the concept of Service based authentication to get access to ResourceResolver.  
  
Let us see how to create the Service Users and Use the same to get access to ResourceResolver in the OSGi service  
  
**Step1: Creating Service User Mapping**  
Go to[http://<host>:<port>/system/console/configMgr](https://www.blogger.com/)  
Search for ‘Apache Sling Service User Mapper Service’ and click on edit  
Add an entry by clicking ‘+’  
<bundleId>:<subServiceName> = <systemUserName>  
  
Ex: org.test.core:readService=testreaduser

User Mapping Service  
  
**Step 2: Create a User Mapper Service Amendment**  
Add a new Amendment as shown below

User Mapping Service Amendment  
  
If you are using multiple User Mapping for the same service, then the highest Ranking User will be used to authenticate the access for the ResourceResolver.  
  
**Step 3: Create the System User**  
In AEM 6.0, even the normal user could be used in mapping the service but from AEM 6.1 it is mandatory to use only the ‘System User’ in the Mapping.  
  
Goto [http://<host>:<port>/crx/explorer/index.jsp](https://www.blogger.com/)  
  
Click on ‘User Administration’

CRX Explorer  
  
Click on ‘Create System User’

User Administration  
  
Add a userId ‘testreaduser’ and click the tick mark to create System User  
 **Step 4: Permissions to the System User**  
Once you have created the system User, goto /useradmin  
Select the user you created and click on the ‘Permission’ tab  
Enable the ACLs accordingly and ‘Save’.

User Permission

**Step 5: OSGi Service**  
Now you have successfully created the service users which can be used in your services to get access to ResourceResolver. Below is an example that shows how to use the service user to get [https://www.acheterviagrafr24.com/viagra-online/](https://www.blogger.com/) the access.  
  
**Create an Interface**  
package org.test.test.core.service;  
public interface ReadService {  
public void listTitles();  
}  
  
**Create an Impl class**  
*package org.test.test.core.service.impl;  
import java.util.HashMap;  
import java.util.Map;  
import org.apache.felix.scr.annotations.Activate;  
import org.apache.felix.scr.annotations.Component;  
import org.apache.felix.scr.annotations.Reference;  
import org.apache.felix.scr.annotations.Service;  
import org.apache.sling.api.resource.Resource;  
import org.apache.sling.api.resource.ResourceResolver;  
import org.apache.sling.api.resource.ResourceResolverFactory;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.test.test.core.service.ReadService;  
import com.day.cq.wcm.api.Page;  
@Component(immediate = true) @Service public class ReadServiceImpl implements ReadService {  
private final Logger log = LoggerFactory.getLogger(this.getClass());  
@Reference private ResourceResolverFactory resourceFactory;  
@Override @Activate public void listTitles() {  
Map < String, Object > paramMap = new HashMap < String, Object > (); //Mention the subServiceName you had used in the User Mapping paramMap.put(ResourceResolverFactory.SUBSERVICE, "readService"); log.info("After the param"); ResourceResolver rr = null; try{ rr = resourceFactory.getServiceResourceResolver(paramMap); log.info("UserId : " + rr.getUserID()); Resource res = rr.getResource("/content/geometrixx"); log.info("Resource : " + res.getPath()); Page page = res.adaptTo(Page.class); log.info("page path : " + page.getPath()); log.info("page title : " + page.getTitle()); }catch(Exception e){ log.error(e.getMessage());  
}  
}  
}*  
Once you install the bundle, you should be able to see the mentioned logs in your <project>.log file  
  
This is slightly different from AEM 6.0 were in just having the UserMapping and the User would be sufficient to get access in ResourceResolver. In 6.1 it's changed a bit with Amendments and the compulsory of creating System User only to work.

### Creating JMX MBeans in AEM

***Java Management Extensions*** (JMX) is a Java technology that provides an architecture to manage resources dynamically at runtime. [JMX](https://docs.oracle.com/javase/tutorial/jmx/mbeans/index.html) is used mostly in enterprise applications to make the system configurable or to get the state of the application at any point in time. Those resources are represented by objects called ***MBeans*** (for Managed Bean).  
  
In AEM also, sometimes we need to use this architecture to manage resources dynamically. Although AEM provides a rich set of out of the box MBeans, we also create our own MBeans as and when required.  
  
In this post, we will be creating an MBean that will take a set of input parameters and return the status of all the ***bundles/services/components*** in the AEM instance in a JSON format.  
  
Let us look at the steps of creating such an MBean.

## Create an MBean

* The first step is to create the layout of our MBean. By layout, I mean the input parameters that the MBean requires to function.
* All the parameters are specified by the specified parameters. In our case, we need the server's hostname, port number and the entity which we want to examine (bundles/services/components).
* Create an interface named ***SystemInfo*** and paste the following code in it

*package org.redquark.demo.core.jmx;*

*import com.adobe.granite.jmx.annotation.Description;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* This interface exposes the input parameter for the MBean*

*\*/*

*@Description("Input parameters for getting the System information")*

*public interface SystemInfo {*

*@Description("Enter the protocol, hostname, and port of the server")*

*String getBundles(String protocol, String hostName, String port);*

*@Description("Enter the protocol, hostname, and port of the server")*

*String getComponents(String protocol, String hostName, String port);*

*@Description("Enter the protocol, hostname, and port of the server")*

*String getServices(String protocol, String hostName, String port);*

*}*

* Here we have three methods that take host, port, and protocol and gives us the JSON response of the bundles, services, and components
* Now let us create the implementation class of the ***SystemInfo*** interface and paste the following code in it

*package org.redquark.demo.core.jmx;*

*import java.io.BufferedReader;*

*import java.io.IOException;*

*import java.io.InputStreamReader;*

*import java.net.URL;*

*import java.net.URLConnection;*

*import javax.management.DynamicMBean;*

*import javax.management.NotCompliantMBeanException;*

*import org.apache.commons.codec.binary.Base64;*

*import org.osgi.service.component.annotations.Component;*

*import org.redquark.demo.core.utils.Network;*

*import org.slf4j.Logger;*

*import org.slf4j.LoggerFactory;*

*import com.adobe.granite.jmx.annotation.AnnotatedStandardMBean;*

*/\*\**

*\* @author Anirudh Sharma*

*\**

*\* Implementation class for the MBean*

*\*/*

*@Component(immediate = true, service = DynamicMBean.class, property = {*

*"jmx.objectname = org.redquark.demo.core.jmx:type=System Info MBean"*

*})*

*public class SystemInfoImpl extends AnnotatedStandardMBean implements SystemInfo {*

*/\*\**

*\* Logger*

*\*/*

*private static final Logger log = LoggerFactory.getLogger(SystemInfoImpl.class);*

*/\*\**

*\* Parameters to read user input*

*\*/*

*public SystemInfoImpl() throws NotCompliantMBeanException {*

*super(SystemInfo.class);*

*}*

*@Override*

*public String getBundles(String protocol, String hostname, String port) {*

*log.info("Logging the bundles in JSON");*

*String url = protocol + "://" + hostname + ":" + port + "/system/console/bundles/.json";*

*String result;*

*if (protocol.equalsIgnoreCase("http")) {*

*result = makeHttpCall(url);*

*} else {*

*result = Network.readJson(url);*

*}*

*return result;*

*}*

*@Override*

*public String getComponents(String protocol, String hostname, String port) {*

*log.info("Logging the components in JSON");*

*String url = protocol + "://" + hostname + ":" + port + "/system/console/components/.json";*

*String result;*

*if (protocol.equalsIgnoreCase("http")) {*

*result = makeHttpCall(url);*

*} else {*

*result = Network.readJson(url);*

*}*

*return result;*

*}*

*@Override*

*public String getServices(String protocol, String hostname, String port) {*

*log.info("Logging the services in JSON");*

*String url = protocol + "://" + hostname + ":" + port + "/system/console/services/.json";*

*String result;*

*if (protocol.equalsIgnoreCase("http")) {*

*result = makeHttpCall(url);*

*} else {*

*result = Network.readJson(url);*

*}*

*return result;*

*}*

*private String makeHttpCall(String requestURL) {*

*URL url;*

*try {*

*url = new URL(requestURL);*

*URLConnection uc;*

*uc = url.openConnection();*

*uc.setRequestProperty("X-Requested-With", "Curl");*

*String userpass = "admin" + ":" + "admin";*

*String basicAuth = "Basic " + new String(new Base64().encode(userpass.getBytes()));*

*uc.setRequestProperty("Authorization", basicAuth);*

*BufferedReader reader = new BufferedReader(new InputStreamReader(uc.getInputStream()));*

*StringBuilder builder = new StringBuilder();*

*String line = null;*

*while ((line = reader.readLine()) != null) {*

*builder.append(line);*

*builder.append(System.getProperty("line.separator"));*

*}*

*String result = builder.toString();*

*return result;*

*} catch (IOException e) {*

*e.printStackTrace();*

*}*

*return "";*

*}*

*}*

* Here we are registering this class of service ***DynamicMBean***which exposes a dynamic management interface. We are using property ***jmx.objectname***which defines the name of our MBean. We are reading the passed parameters and creating an appropriate URL to get the JSON response.
* Now deploy the code and navigate to [http://<host>:<port>/system/console/jmx](http://localhost:4502/system/console/jmx). There you will see all the registered MBean. Search for your MBean.

|  |
| --- |
| [aem49](https://2.bp.blogspot.com/-z_dT8vfaIt0/W8hCpvHDctI/AAAAAAAACYs/ivHydJJucpcSGscrPelryMjvfeQT2phYQCLcBGAs/s1600/aem49.png) |
| System Info MBean |

* Open the MBean, you will see three methods each for bundles, components, and services.
* Open the ***getBundles()*** methods, and configure it. After clicking on ***invoke,***you will see the JSON response returned.

|  |
| --- |
| [aem50](https://4.bp.blogspot.com/-c9UtyLI2e-U/W8hDHliyGqI/AAAAAAAACY0/pwEGXCY2f6QQO0sxIjVj6MDSJM05KwJuACEwYBhgL/s1600/aem50.png) |
| JSON Response returned |

# Getting familiar with Sitemaps

## About Sitemap

A sitemap is an XML file that includes URLs of a website and their basic information like last updated, how often it changes and how important it is as compared to other URLs by giving its priority.

### Purpose of Sitemap

The information listed in Sitemap.xml file is used by the search engines to decide efficiently which URLs are available for crawling over a website. So Sitemaps are a URL inclusion protocol that allows the search engines to crawl a site much more effectively and sensibly.

Sitemap being an XML file consists of XML tags. The Sitemap file must be UTF-8 encoded and can be a normal text file or a compressed .gz file. Some of the supported XML tags are:

<?xml version="1.0" encoding="UTF-8"?>

<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

<url>

<loc>http://www.example.com/</loc>

<lastmod>2015-07-29</lastmod>

<changefreq>monthly</changefreq>

<priority>0.6</priority>

</url>

</urlset>

* **<urlset>**: This tag is a required tag and is the topmost element used in the sitemap file after <?xml version> element and the rest of the elements must be contained inside it.
* **<url>**: This tag is a required tag and is the parent tag for each URL entry. The remaining tags are children of this tag.
* **<loc>**: This tag is a required tag and is used to indicate the URL of the page. This URL must begin with the protocol(such as http, https) and end with a trailing slash if required by the webserver. Its value must be less than 2048 characters.
* **<lastmod>**: This tag is an optional tag and indicates the date of the last modification of the file. One of its formats is YYYY-MM-DD.
* **<changefreq>**: This tag is an optional tag and denotes how frequently the page is likely to change. Its values are: never, yearly, monthly, weekly, daily, hourly, always.  
  If there is a document that changes every time it is being accessed, in that case, the document should use “always” value and if there are some archived URLs which we know that never going to change any more than in that case it’s better to use “never” value.
* **<priority>**: This tag is an optional tag and used to describe the priority of this URL relative to other URLs of a website. Valid values are from 0.0 to 1.0. Its default value for a page is 0.5.

There can be multiple Sitemap files, with each Sitemap file not having more than 50,000 URLs or must not be larger than 10MB. One can also compress the Sitemap file but when a Sitemap compressed file is uncompressed it must not be more than 10MB of size. If more URLs need to be included beyond the 50,000 limits than create multiple Sitemap files.

### Sitemap Index File

The main purpose of the Sitemap index file is to group multiple sitemap files i.e if multiple Sitemap files are used then each one should be listed in a Sitemap index file.

### Purpose of using the Sitemap Index File

By doing so it would be easier for the crawler to keep track of multiple Sitemap files created for a website. Also similar to Sitemap files, the Sitemap index file must not list more than 50,000 Sitemaps and must not be larger than 10MB. There can be multiple Sitemap index files as well and can be compressed.

**The following example shows a Sitemap index file:**

<?xml version="1.0" encoding="UTF-8"?>

<sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">

<sitemap>

<loc>http://www.example.com/sitemap1.xml.gz</loc>

<lastmod>2015-07-28</lastmod>

</sitemap>

<sitemap>

<loc>http://www.example.com/sitemap2.xml.gz</loc>

<lastmod>2015-07-29</lastmod>

</sitemap>

</sitemapindex>

* **<sitemapindex>**: This is a required tag and includes information about all the Sitemaps available for a website.
* **<sitemap>**: This is a required tag and includes information about an individual Sitemap.
* **<loc>**: This is a required tag and indicates the location of the Sitemap.
* **<lastmod>**: This is an optional tag and identifies the time a particular Sitemap file was modified.

An important **note** about Sitemap.xml file that it must be UTF-8 encoded and any URLs or data values must use entity escape codes for the following characters listed below:

| **Character** | **Escape Code** |
| --- | --- |
| * Ampersand (&) | &amp; |
| * Single Quote (‘) | &apos; |
| * Double Quote (“) | &quot; |
| * Greater Than (>) | &gt; |
| * Less Than (<) | &lt; |

I hope you enjoyed it and will explore more about Sitemaps.

# Enabling Vanity URLs with Adobe Experience Manager

**Statement - Handling Vanity URLs Using the AEM Dispatcher Module**

**Solution:**  
Recent updates to the AEM Dispatcher module (since version 4.1.9 of the module) allow authors to directly control vanity URLs from within the Author UI, and these are automatically pushed out to the publishers, which then expose them to the dispatchers.  
  
**Steps to be followed:**

* On your AEM Publish nodes, download and install the [VanityURLS-Components](https://www.adobeaemcloud.com/content/marketplace/marketplaceProxy.html?packagePath=/content/companies/public/adobe/packages/cq600/component/vanityurls-components) (<https://www.adobeaemcloud.com/content/marketplace/marketplaceProxy.html?packagePath=/content/companies/public/adobe/packages/cq600/component/vanityurls-components>)
* Package from Adobe Package Share, or pull it down and install it manually in /crx/packmgr.
* Go to the /useradmin on your Publish instance and allow “Read” permission to */libs/granite/dispatcher/content/vanityUrls* for the “Everyone” group. Do this by double-clicking the “Anonymous” user, Then go to “Permissions” and check the “Read” column for the above path.
* If you don’t have an allow-all in your dispatcher configs, add a filter rule in the dispatcher to allow the vanity URL to be called on Publish instance:

*/0100 { /type "allow" /url "/libs/granite/dispatcher/content/vanityUrls.html" }*

* Add a caching rule to prevent caching of this URL:

*/0001 { /type "deny" /glob*  
*"/libs/granite/dispatcher/content/vanityUrls.html" }*

* Add the vanity\_urls configuration to the farm:

*/vanity\_urls {  
/url "/libs/granite/dispatcher/content/vanityUrls.html"  
/file "/tmp/vanity\_urls"  
/delay 300  
}*

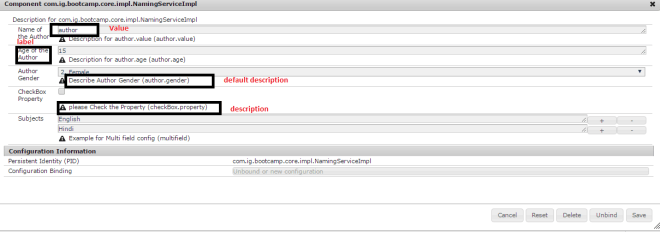
* Re-start Apache.
* The file defined at the */file* setting is not automatically created/updated at the time interval set at */delay*, but only when a request is made that fails the */filter* rules of your dispatcher. On fail, it checks to see if the file is there —
* if not, it will generate and use it by pulling */libs/granite/dispatcher/content/vanityUrls.html* from the publisher. If it is there, and not older than*/delay* seconds, it will use it. Finally, if it is older than */delay* seconds, it will update it from the Publish instance and use it.

# Create Custom OSGi Configuration in AEM

OSGI (Open Service Gateway Initiative) is a major building block in aem architecture which makes it modular. In AEM, there is a common need of creating custom OSGi configuration. These Configurations can be a single value,multi-value,drop down list and checkbox like property.  
  
In this tutorial i have tried to cover all the available options for creating custom OSGi configuration in aem.

* How to Set and Get OSGI configuration values using Text Field.
* How to Set and Get OSGI configuration values using Drop Down List.
* How to Set and Get OSGI configuration values using CheckBox.
* How to Set and Get OSGI configuration values using Multi Field.

**Set and Get OSGI configuration values using Textfield:**  
When user wants to get and set single value in osgi configuration, Text Field is preferred.  
  
**Setting Values:**  
*@Property(label="Name of the Author",value = "author")  
@Property(label="Age of the Author",intValue = 15)*  
**Getting Values:**  
*private static final String AUTHOR\_NAME = "author.value";  
private static final String AUTHOR\_AGE = "author.age";*  
*value:* This property indicates default value of the configuration.  
*intValue:* If you want to take only integer value ,use this property to set default value.  
  
**Note:** @Property annotation helps us to define the properties in felix console.

**Set and Get OSGI configuration values using Drop Down List:**  
Drop Down list is a very common requirement that a developer encounters, when customer wants to display list of countries, states, gender etc to choose from felix console.  
  
**Note:**User can add as many options in the list , but select only once at a time.  
  
**Setting Values:**  
*@Property(  
label = "Author Gender",  
description = "Describe Author Gender",  
options = {  
@PropertyOption(name = "Male", value = "1. Male"),  
@PropertyOption(name = "Female", value = "2. Female")  
},  
value = "Female")*  
**Getting Values:**  
*private static final String AUTHOR\_GENDER = "author.gender";*  
  
**Set and Get OSGI configuration values using CheckBox :**  
The checkBox is a good solution when number of options are limited, and customers wants to select single or multiple values from felix console.  
  
**Note:**The Value of this property is boolean ,we can set default value by the attribute boolValue.  
  
**Setting Values:**  
*@Property(label = "CheckBox Property " ,boolValue = false,description = "please Check the Property")*  
**Getting Values:**  
*public static final String CHECKBOX\_PROPERTY = "checkBox.property";***Set and Get OSGI configuration values using Multi-Field :**  
Multifield Components is the most preffered component by developers when their is a requirement of creating a custom configuration which consist dynamic number of fields. We can add as many values in it as per our requirement  
  
*@Property(value={"English", "Hindi"}, unbounded = PropertyUnbounded.ARRAY, label = "Subjects", cardinality = 50, description = "Example for Multi field config")  
private static final String MULTI\_FIELD = "multifield";*  
**Now in order to fetch the values we require two methods:**  
*Activate:*This method is called only once when the bundle is activated.  
*Modified:* When we made the changes in our configurations, then we need to fetch the values from this method because the bundle is already activated.  
  
*@Activate  
protected void activate(@SuppressWarnings("rawtypes") final Map context) {  
this.name = PropertiesUtil.toString(context.get(AUTHOR\_NAME), "");  
this.gender = PropertiesUtil.toString(context.get(AUTHOR\_GENDER),"");  
this.checkbox = PropertiesUtil.toBoolean(context.get(CHECKBOX\_PROPERTY),true);  
this.multiString = PropertiesUtil.toStringArray(context.get(MULTI\_FIELD));  
this.age = PropertiesUtil.toInteger(context.get(AUTHOR\_AGE),12);  
  
}*  
*@Modified  
protected void modified(ComponentContext context){  
this.name = PropertiesUtil.toString(  
context.getProperties().get(AUTHOR\_NAME), "");  
this.gender = PropertiesUtil.toString(  
context.getProperties().get(AUTHOR\_GENDER), "");  
this.checkbox = PropertiesUtil.toBoolean(  
context.getProperties().get(CHECKBOX\_PROPERTY), true);  
this.multiString = PropertiesUtil.toStringArray(  
context.getProperties().get(MULTI\_FIELD));  
this.age = PropertiesUtil.toInteger(  
context.getProperties().get(AUTHOR\_AGE),12);  
}*  
  
**Testing OSGI Configurations:**  
Go to [Felix Console.](http://localhost:4502/system/console/configMgr)  
Select the bundle that you want to Verify or Configure.  
  
[](https://i0.wp.com/www.aemcq5tutorials.com/wp-content/uploads/2016/05/custom-osgi-confiuration-aem-set-and-get-values-1.png)  
  
You can find the complete code [here](https://github.com/gargshivani111/access-felix-property).

# How to find the repository properties in AEM through Felix console

**Statement** - Repository properties of AEM  
**Environment** - AEM 6.3 GA  
  
**Solution:**  
Go to the Felix console URL: <https://localhost:4502/system/console/status-Repository%20Apache%20Jackrabbit%20Oak>

login with username and password if not logged into the AEM server.  
Below screenshot shows the list of repository properties

Repository Properties:

crx.cluster.id: 26eb16c2-ac36-4631-8783-c229474ae7b22

crx.cluster.master: true

crx.repository.systemid: b6e06f32-20a1-4fa0-869d-7a798282fded

identifier.stability: identifier.stability.method.duration

jcr.repository.name: Apache Jackrabbit Oak

jcr.repository.vendor: The Apache Software Foundation

jcr.repository.vendor.url: http://www.apache.org/

jcr.repository.version: 1.6.1

jcr.specification.name: Content Repository for Java Technology API

jcr.specification.version: 2.0

level.1.supported: true

level.2.supported: true

node.type.management.autocreated.definitions.supported: true

node.type.management.inheritance: node.type.management.inheritance.single

node.type.management.multiple.binary.properties.supported: true

node.type.management.multivalued.properties.supported: true

node.type.management.orderable.child.nodes.supported: true

node.type.management.overrides.supported: true

node.type.management.primary.item.name.supported: true

node.type.management.property.types: -

node.type.management.residual.definitions.supported: true

node.type.management.same.name.siblings.supported: false

node.type.management.update.in.use.suported: false

node.type.management.value.constraints.supported: true

oak.clusterid: 26eb16c2-ac36-4631-8783-c229474ae722

oak.discoverylite.clusterview: {"seq":1,"final":true,"me":1,"id":"26eb16c2-ac36-4631-8783-c229474ae7b6","active":[1],"deactivating":[],"inactive":[]}

option.access.control.supported: true

option.activities.supported: false

option.baselines.supported: false

option.journaled.observation.supported: false

option.lifecycle.supported: false

option.locking.supported: true

option.node.and.property.with.same.name.supported: true

option.node.type.management.supported: true

option.observation.supported: true

option.principal.management.supported: true

option.privilege.management.supported: true

option.query.sql.supported: false

option.retention.supported: false

option.shareable.nodes.supported: false

option.simple.versioning.supported: false

option.transactions.supported: false

option.unfiled.content.supported: false

option.update.mixin.node.types.supported: true

option.update.primary.node.type.supported: true

option.user.management.supported: true

option.versioning.supported: true

option.workspace.management.supported: false

option.xml.export.supported: true

option.xml.import.supported: true

query.full.text.search.supported: false

query.joins: query.joins.none

query.languages: -

query.stored.queries.supported: false

query.xpath.doc.order: false

query.xpath.pos.index: false

write.supported: true

# ncrease the Bundle Cache size in AEM

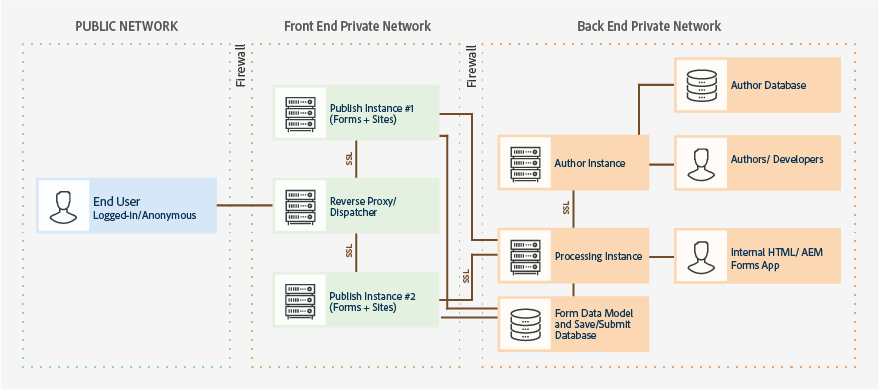
**Steps Increase the Bundle Cache size:**  
**Step1:** repository.xml and all of your workspace.xml  
**Step2:** Edit PersistenceManager elements in repository.xml and workspace.xml file  
**Step3:** add an element  
**Step4:** Increased size to 256 or 512 or 1024, depending upon jvm size.

# Set the JVM Max, Min heap and Perm size of JVM in AEM

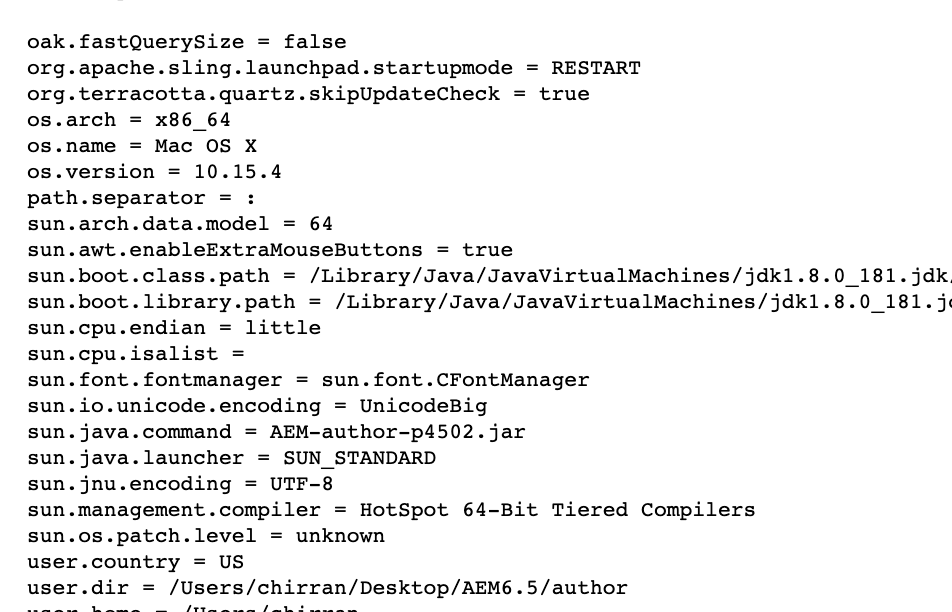
Add the below details in start.bat/start.sh file of AEM:  
CQ\_JVM\_OPTS ='-server –Xms8192m –Xmx8192m -XX:MaxPermSize=1024M  
whereas

* Xms is Min heap - size can be increased depending on your instance RAM size
* Xmx is Max heap - size can be increased depending on your instance RAM size
* MaxPermSize size can be increased depending on your instance RAM size
* finally, restart the instance.

# Hardening and Securing Checklist For AEM forms on OSGi environment

* - AEM FORMS on OSGI Deployment Architecture  
    
    
    
  **Reference URL:**  
  <https://helpx.adobe.com/experience-manager/6-4/forms/using/hardening-securing-aem-forms-environment.html#SecureintranetelementsofanAEMFormsenvironment>  
    
  <https://helpx.adobe.com/experience-manager/6-4/forms/using/hardening-aem-forms-jee-environment.html#protecting_from_cross_site_request_forgery_attacks>

# How to find the OS version, OS architecture and OS name in AEM through Felix console

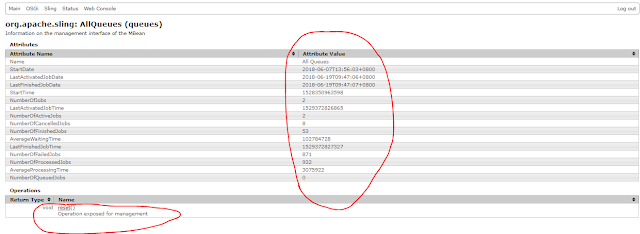
* **Statement -** Assuming if you do not have access to the physical server on AEM is installed and an alternate way to find the OS details  
  **Environment**- AEM 6.3 GA  
    
  **Solution:**  
  Go to the Felix console URL: [http://localhost:4502/system/console/status-System%20Properties](https://www.blogger.com/)  
  login with username and password if not logged into the AEM server.  
  Look for OS details as shown in the below screenshot.  
  [](https://www.blogger.com/)

# How to Check status of All Queues in AEM through JMX console

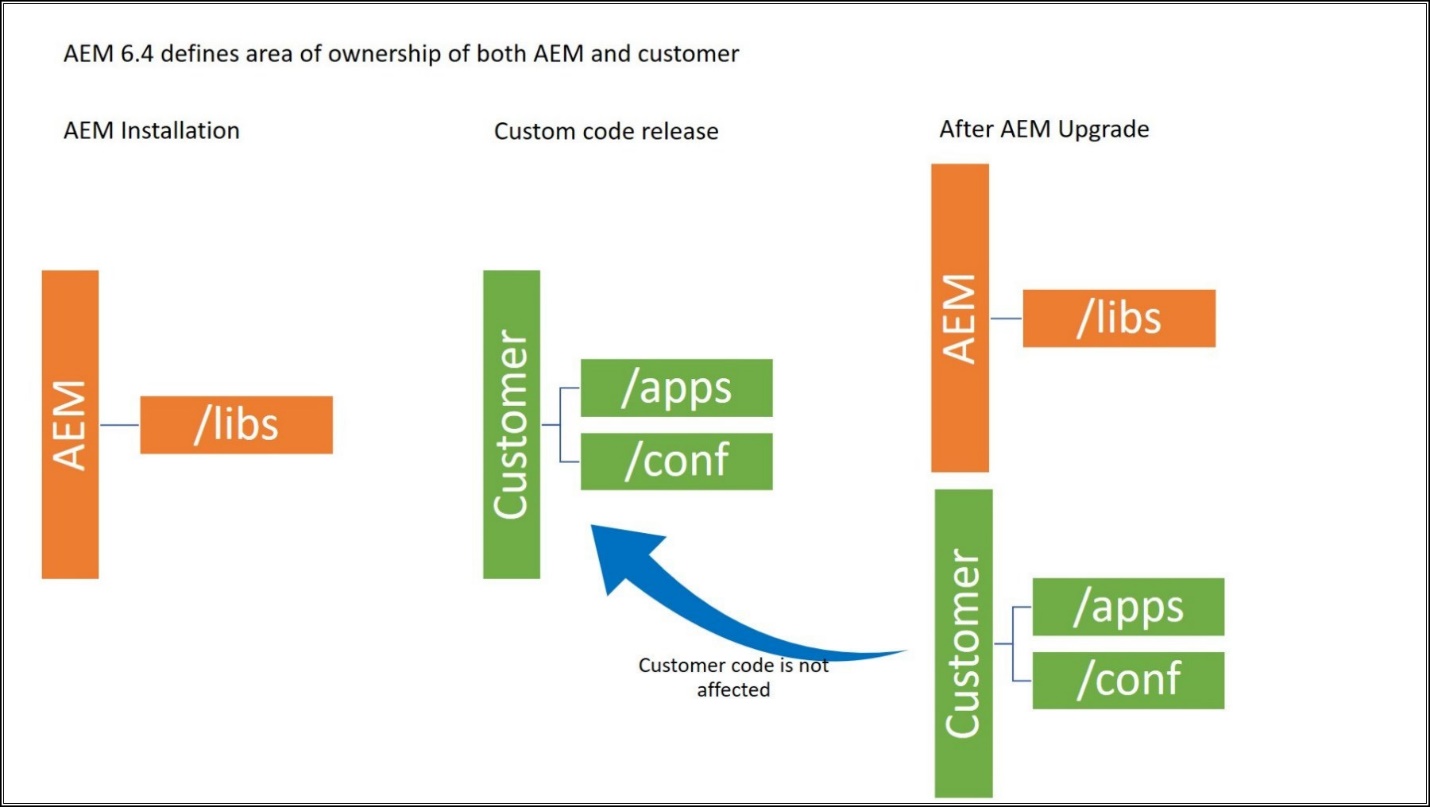
* Statement - All Queues Status  
    
  **Solution:**  
    
  Status of All Queues:

|  |  |
| --- | --- |
| **Attributes** |  |
| **Attribute Name** | **Attribute Value** |
| Name | All Queues |
| StartDate | 2018-06-07T13:56:03+0800 |
| LastActivatedJobDate | 2018-06-19T09:47:06+0800 |
| LastFinishedJobDate | 2018-06-19T09:47:07+0800 |
| StartTime | 1.5E+12 |
| NumberOfJobs | 2 |
| LastActivatedJobTime | 1.5E+12 |
| NumberOfActiveJobs | 2 |
| NumberOfCancelledJobs | 8 |
| NumberOfFinishedJobs | 53 |
| AverageWaitingTime | 1E+08 |
| LastFinishedJobTime | 1.5E+12 |
| NumberOfFailedJobs | 871 |
| NumberOfProcessedJobs | 932 |
| AverageProcessingTime | 3075922 |
| NumberOfQueuedJobs | 0 |

|  |  |
| --- | --- |
| **Operations** |  |
| **Return Type** | **Name** |
| void | reset() |
| Operation exposed for management |

* **Screenshot for Queue status:**  
  [](https://4.bp.blogspot.com/-8Lsi8dl8J3U/Wyht3199TBI/AAAAAAAASzE/jtnLMSXtpxMPNUHXFfPwM7FNioBfFUQzwCLcBGAs/s1600/AllQueues.PNG)

# Repository restructuring in AEM 6.4

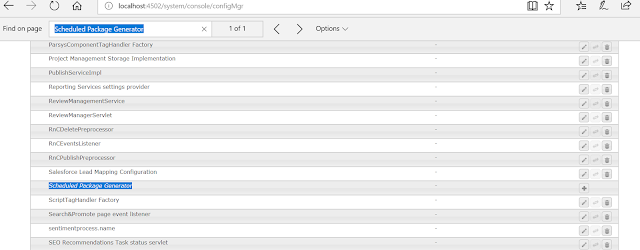
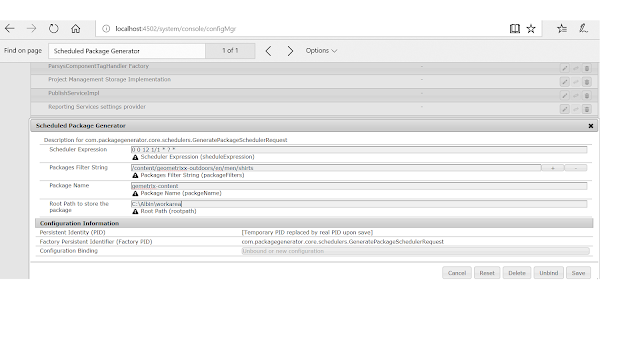
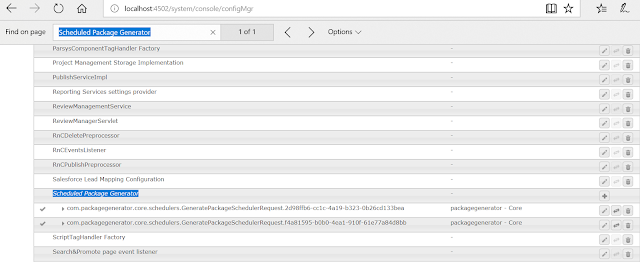
AEM product code will always be placed in /libs, which must not be overwritten by custom code  
Custom code should be placed in /apps, /content, and /conf  
  
  
Starting from AEM 6.4 (see <https://helpx.adobe.com/experience-manager/6-4/sites/deploying/using/repository-restructuring.html>) content repository is to be reorganized prior to 6.5, and finally in 6.5 most probably will follow the rule as below:

1. /etc should not be used (probably to be removed)
2. any application code, clientlibs should be located under /apps
3. any runtime data to be under /var or /content folder,
4. while the configurations should be rather under /conf

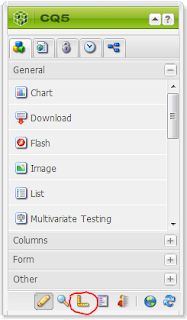
The groovy console should have its own content structure to be reorganised as follows:

1. /etc/clientlibs/groovyconsole should rather go to /apps/groovyconsole/clientlibs
2. /etc/groovyconsole to /apps/groovyconsole
3. /etc/groovyconsole/scripts should go to /conf/groovyconsole/scripts or /var/groovyconsole/scripts
4. /etc/groovyconsole/jcr:content/audit should to to /var/groovyconsole/audit

# How to define a factory of OSGI schedulers (Dynamic Schedulers) in AEM

1. This post explains the approach to define factory of OSGI schedulers(Dynamic Schedulers) in Adobe Experience Manager(AEM)  
     
   Define a factory class to capture the configurations:  
   Define a component and enable as configuration factory(configurationFactory=true)  
   Configure all the required properties to run the scheduler job.  
     
   Retrieve the properties through the activate method and assign the values to a class variable, define the getters to retrieve the property values in Job Class.  
     
   Specify the label to identify the configuration in the OSGI console.  
     
   @Component(configurationFactory = true, policy = ConfigurationPolicy.OPTIONAL, metatype = true, immediate = true, label = "Scheduled Package Generator")  
   @Service(value = GeneratePackageSchedulerRequest.class)  
   public class GeneratePackageSchedulerRequest {  
     
   final Logger logger = LoggerFactory.getLogger(this.getClass());  
     
   @Property(unbounded=PropertyUnbounded.DEFAULT, label="Scheduler Expression", description="Scheduler Expression", value="")  
   private static final String SCHEDULER\_EXPRESSION = "sheduleExpression";  
   private String sheduleExpression;  
     
   @Property(unbounded = PropertyUnbounded.ARRAY, label = "Packages Filter String", cardinality = 50, description = "Packages Filter String")  
   public static final String PACKAGE\_FILTERS = "packageFilters";  
   private String[] packageFilters;  
     
   @Property(unbounded=PropertyUnbounded.DEFAULT, label="Package Name", description="Package Name")  
   private static final String PACKAGE\_NAME = "packgeName";  
   private String packageName;  
     
   @Property(unbounded=PropertyUnbounded.DEFAULT, label="Root Path to store the package", description="Root Path")  
   private static final String ROOT\_PATH = "rootpath";  
   private String rootPath;  
     
   @Activate  
   protected void activate(final ComponentContext ctx) {  
     
   Dictionary<?, ?> props = ctx.getProperties();  
   sheduleExpression = PropertiesUtil.toString(props.get(SCHEDULER\_EXPRESSION), "");  
   packageFilters = PropertiesUtil.toStringArray(props.get(PACKAGE\_FILTERS), null);  
   packageName = PropertiesUtil.toString(props.get(PACKAGE\_NAME), null);  
   rootPath=PropertiesUtil.toString(props.get(ROOT\_PATH), null);  
   }  
     
   public String[] getPackageFilters() {  
   return packageFilters;  
   }  
     
   public String getJobname() {  
   return packageName;  
   }  
     
   public String getSheduleExpression() {  
   return sheduleExpression;  
   }  
     
   public String getRootPath() {  
   return rootPath;  
   }  
     
   }  
     
   Define a Job class to perform the required task with provided data:  
   @Component(immediate = true, metatype = true)  
   @Service(GeneratePackageScheduledTask.class)  
   public class GeneratePackageScheduledTask {  
     
   @Reference  
   private SlingRepository repository;  
     
   @Reference  
   private SlingSettingsService settingsService;  
     
   @Reference  
   private Scheduler scheduler;  
     
   protected final Logger logger = LoggerFactory.getLogger(this.getClass());  
     
   @Reference(cardinality = ReferenceCardinality.OPTIONAL\_MULTIPLE, referenceInterface = GeneratePackageSchedulerRequest.class, policy = ReferencePolicy.DYNAMIC)  
   private final List<GeneratePackageSchedulerRequest> providers = new LinkedList<GeneratePackageSchedulerRequest>();  
     
   protected void bindProviders(final GeneratePackageSchedulerRequest config) throws Exception {  
     
   providers.add(config);  
   final String schedulingExpression=config.getSheduleExpression();  
   final String jobname= config.getJobname();  
     
   final Runnable job = new Runnable() {  
     
   public void run() {  
     
   logger.debug("run() STARTS"+jobname);  
     
   if (isRunMode("author")&& isMasterRepository()) {// this can be removed if the Job can be executed in all nodes  
     
   try {  
     
   //define the Job code here  
     
   } catch (Exception rex) {  
   rex.printStackTrace();  
   logger.error("Error occurred in Job execution..", rex.getMessage());  
   }  
   }  
     
   logger.debug("run() END"+jobname);  
   }  
     
   };  
     
   ScheduleOptions so = scheduler.EXPR(schedulingExpression);  
   so.name(jobname);  
   so.canRunConcurrently(true);// change based on the Job configuration  
   this.scheduler.schedule(job, so);  
   logger.debug("Scheduled Job: " + config.getJobname()+" "+schedulingExpression);  
   }  
     
   protected void unbindProviders(final GeneratePackageSchedulerRequest config) {  
   logger.debug("Removed Job: " + config.getJobname());  
   this.scheduler.unschedule(config.getJobname());  
   providers.remove(config);  
   }  
     
   private Boolean isRunMode(String mode) {  
   Set<String> runModes = settingsService.getRunModes();  
   for (String runMode : runModes) {  
   if (runMode.equalsIgnoreCase(mode)) {  
   logger.debug("Current Runmode is : " + runMode);  
   return true;  
   }  
   }  
   return false;  
   }  
     
   public boolean isMasterRepository(){  
   final String isMaster = repository.getDescriptor("crx.cluster.master");  
   logger.debug("isMaster.."+isMaster);  
   return isMaster!=null && !isMaster.equals("") && Boolean.parseBoolean(isMaster);  
   }  
     
   }  
     
   Define bindProviders and unbindProviders methods  
   bindProviders - this method will be invoked whenever a new Factory configuration is created/modified with the input config object. Schedule the Job with input configurations, the Job will be scheduled based on the schedulingExpression.  
     
   unbindProviders - this method will be invoked whenever the existing factory configuration is removed. unschedule the existing Job with the same Job name.  
     
   After deploying the bundle, login to OSGI configuration and search for Factory configuration - search through the label specified in the factory - Scheduled Package Generator  
     
   [](https://www.blogger.com/)  
   Define the individual configurations - the Job will be scheduled based on the configurations  
   [](https://www.blogger.com/)  
     
   [](https://www.blogger.com/)  
     
   The Jobs will be running based on the scheduled time, refer the log file for more details - better define a separate log file to capture scheduler logs

# How to create a custom multifield component

1. **The Problem**
2. You are trying to capture a list of key/pair values in a custom component in the admin interface.  
     
   **The Solution**
3. Create a custom multified component to capture values.  
     
   1. Create multifield component  
   1.1 Create the component  
   The first thing we are going to do is to create a new component for our tutorial project. This component will be visible in the siteadmin editor.  
     
   Open up CRXDE and create a new components directory in the following location:  
   \apps\tutorial\components\ConfigComponent  
     
   Right click on the apps folder-> New -> Folder. Name this tutorial  
   Right click on the tutorial folder->New -> Folder. Name this components  
   Right click on the components folder -> New -> Component  
     
   Label -> ConfigComponent  
   Title -> Config Component  
   Description -> Key pair configuration component  
   Super Resource Type -> blank  
   Group -> Admin (Or whatever group you want this component to live in)  
     
   Click on next  
     
   in AllowedParents enter  
   \*/\*parsys  
     
   Click on finish.  
     
   You have now created your new skeleton component.  
     
   1.2 Create the component display  
   If you look into the ConfigComponent directory, you will see a ConfigComponent.jsp file.  
     
   This jsp will be the component as it renders for display.  
     
   In this case, I have decided both for backwards compatibility reasons, and simplicity, to use a table. You can style this in whatever fashion that you desire, however, for the purposes of this tutorial, we will keep it vanilla.  
     
   Note, we could also abstract this functionality out into seperate java classes etc, but again, for simplicity, we will just do this in a vanilla fashion.  
     
   <%@include file="/libs/foundation/global.jsp"%>Configuration Settings  
   <table>  
   <tr>  
   <th>Key</th>  
   <th>Value</th>  
   </tr><%  
   PropertyIterator propertyIterator= currentNode.getProperties();  
   while(propertyIterator.hasNext())  
   {  
   Property currentProperty=propertyIterator.nextProperty();  
   String currentValue="";  
   if(currentProperty!=null &&  
   currentProperty.getName()!=null &&  
   currentProperty.getName().toLowerCase().compareTo("keyvaluepairs")==0)  
   {  
   if(currentProperty.isMultiple())  
   {  
   Value[] values=currentProperty.getValues();  
   for(Value value:values)  
   {  
   currentValue=value.getString();  
   String[] splitValue=currentValue.split("#");  
   %><tr><%  
   for(int splitLoop=0; splitLoop< splitValue.length; splitLoop++)  
   {  
   %><td><%= splitValue[splitLoop] %></td><%  
   }  
   %></tr><%  
   }  
   }  
   else  
   {  
   currentValue=currentProperty.getString();  
   %><td><%= currentValue %></td><%  
   }  
   }  
   properties.isEmpty();  
   }  
   %></table>  
     
   What we are doing in this code, is taking the current node:  
   PropertyIterator propertyIterator= currentNode.getProperties();  
     
   and iterating through its properties.  
     
   while(propertyIterator.hasNext())  
     
   We then get the values from the property keyvaluepairs, and split the value by the delimitation of a hashtag.  
     
   We follow the route of separating values by hashtag, as we wish to avoid having to rewrite the multi-field component entirely. Note that if the # character is a requirement for the field, you can use a multi character delimeter.  
     
   **1.3 Create the component Dialog**  
   Right click on the ConfigComponent folder -> New Dialog  
     
   Name -> dialog  
   Title -> Key Value Pairs  
     
   Click on finish  
   Navigate down to the tab 1 node, and change the title to Key Value Pairs  
     
   Right click on tab1 -> new -> Node  
   Name -> items  
   Type -> cq:WidgetCollection  
     
   Click Finish  
     
   Right click on items -> new -> Node  
   Name-> trainingmulti  
   Type -> cq:Widget  
     
   Set the following properties on the trainingmulti node  
   fieldLabel -> Key Value Pairs  
   jcr:primaryType -> cq:Widget  
   name -> ./keyvaluepairs  
   xtype -> multifield  
     
   So what have we done so far? We've created a component that displays a table of values coming from ./keyvaluepairs.  
     
   So if we go to our siteadim, open a geometrixx page, and create a template with a parsys such as wide page.  
     
   [](https://2.bp.blogspot.com/-zTClIVTzbEk/UPiG1t6vdPI/AAAAAAAAAIQ/JhiXMlYD6Rs/s1600/switchtodesginmode.PNG)  
     
   On the sidekick, click on switch to design mode.  
     
   [parsyseditbtn](https://2.bp.blogspot.com/-go-yCotRido/UPiHZzkdnAI/AAAAAAAAAIY/qSAPfeD31w0/s1600/parsyseditbtn.PNG)  
     
   Click on edit, scroll down to Admin, or whatever group you've added the component to, and enable the component for inclusion in the sidekick.  
     
   If you expand the sidekick again, you should be able to see the component either in Admin section, or the Other section, depending on how many other groups you have enabled.  
     
   Add the component to parsys on the left by dragging and dropping it on.  
     
   You won't see any key pairs or information at this point in time.  
     
   Double click on the component to open the edit dialog.  
     
   Click the add button to add a new row, and enter the following:  
   Test Pair one  
   Test#Pair two  
     
   2. Create custom dual textfields  
   2.x Hooking in the custom xtype  
   create new node on the trainingmulti node of an nt:unstructured type  
     
   Add the xtype property of trainingSelection.  
     
   Add the xtype component -> list out error
4. **Troubleshooting**
5. Below are the articles regarding AEM project Debugging, Troubleshooting and fixing AEM Performance related issues, will update the blog with new posts.
6. **Debugging:**
7. [How to start AEM in Debug mode](https://aem4beginner.blogspot.com/2020/03/how-to-start-aem-in-debug-mode.html)
8. [Debugging in AEM](https://aem4beginner.blogspot.com/2020/03/debugging-in-aem.html)
9. [AEM Application Debugging with Visual Studio IDE](https://aem4beginner.blogspot.com/2020/03/aem-application-debugging-with-visual.html)
10. [Debugging Dispatcher Authorization Issues](https://aem4beginner.blogspot.com/2020/03/debugging-dispatcher-authorization.html)[Debugging and Vault Configuration in IntelliJ Idea and Eclipse in AEM](https://aem4beginner.blogspot.com/2020/03/debugging-and-vault-configuration-in.html)
11. [Day CQ WCM Debug Filter Configuration Missing](https://aem4beginner.blogspot.com/2020/04/day-cq-wcm-debug-filter-configuration.html)
12. [How to Debug a Particular Java Class in CQ](https://aem4beginner.blogspot.com/2020/04/how-to-debug-particular-java-class-in-cq.html)
13. [Debugging in AEM](https://aem4beginner.blogspot.com/2020/04/debugging-in-aem.html)
14. [Debugging JavaScript on Android using Chrome](https://aem4beginner.blogspot.com/2020/04/debugging-javascript-on-android-using.html)
15. [Debugging in IntelliJ IDEA: Part 1](https://aem4beginner.blogspot.com/2020/04/debugging-in-intellij-idea-part-1.html)  
    [Debug a CQ5/AEM6 app using eclipse](https://aem4beginner.blogspot.com/2020/04/debug-cq5aem6-app-using-eclipse.html)[A command to Run AEM/CQ 5.6.1 Instance In Debug Mode](https://aem4beginner.blogspot.com/2020/04/a-command-to-run-aemcq-561-instance-in.html)
16. **Error Handling:**
17. [How to manage certificates in AEM Truststore](https://aem4beginner.blogspot.com/2020/03/how-to-manage-certificates-in-aem.html)  
    [How to Implement 301 and 302 Redirect in AEM](https://aem4beginner.blogspot.com/2020/03/how-to-implement-301-and-302-redirect.html)  
    [Handing 301 Redirection in AEM using Redirect Map Manager](https://aem4beginner.blogspot.com/2020/03/handing-301-redirection-in-aem-using.html)  
    [Error handling using a Dispatcher](https://aem4beginner.blogspot.com/2020/03/error-handling-using-dispatcher.html)
18. [Implementing 301 and 302 Redirect in AEM](https://aem4beginner.blogspot.com/2020/04/implementing-301-and-302-redirect-in-aem.html)  
    [Approach to 301 redirect the internal URL's to dynamic SEO friendly URL](https://aem4beginner.blogspot.com/2020/04/approach-to-301-redirect-internal-urls.html)  
    [Different approaches to perform Vanity/Redirect URL management in AEM](https://aem4beginner.blogspot.com/2020/04/different-approaches-to-perform.html)  
    [External redirect for an AEM page](https://aem4beginner.blogspot.com/2020/04/external-redirect-for-aem-page.html)
19. [RedirectTarget property is not working in AEM 6.2](https://aem4beginner.blogspot.com/2020/04/redirecttarget-property-is-not-working.html)
20. [502 Bad Gateway while accessing /libs/cq/security/userinfo.json via telnet](https://aem4beginner.blogspot.com/2020/04/502-bad-gateway-while-accessing.html)
21. [Different approaches to define vanity URL in AEM](https://aem4beginner.blogspot.com/2020/04/different-approaches-to-define-vanity.html)
22. [Configuring same short URL/Vanity URL for different sites with https](https://aem4beginner.blogspot.com/2020/04/configuring-same-short-urlvanity-url.html)  
    [How to Customize/Configure the 404 error handler for multi-sites in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-customizeconfigure-404-error.html)
23. [How to Enable Cross-Origin Resource Sharing (CORS) support in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-enable-cross-origin-resource.html)
24. [How to display custom error (404,403 and 500) pages in Multi-Site AEM?](https://aem4beginner.blogspot.com/2020/04/how-to-display-custom-site-specific.html)
25. [HTTPS URL is resulting with 404 - Adobe Experience Manager](https://aem4beginner.blogspot.com/2020/04/https-url-is-resulting-with-404-adobe.html)
26. **AEM Logs:**
27. [Overview of CQ Logs](https://aem4beginner.blogspot.com/2020/03/overview-on-cq-logs.html)  
    [Explore Request Logs in AEM/Adobe CQ5](https://aem4beginner.blogspot.com/2020/03/explore-request-logs-in-aemadobe-cq5.html)  
    [Overview of AEM Logs in detail - Part 1](https://aem4beginner.blogspot.com/2020/03/overview-of-aem-logs-in-detail-part-1.html)[Overview of AEM Logs in detail - Part 2](https://aem4beginner.blogspot.com/2020/03/overview-of-aem-logs-in-detail-part-2.html)
28. [When trying to fake a post request, you get the following error in the logs](https://aem4beginner.blogspot.com/2020/04/when-trying-to-fake-post-request-you.html)
29. [Overview of Workflow, Performance monitoring, Logs in AEM/Adobe CQ5](https://aem4beginner.blogspot.com/2020/03/overview-on-workflow-performance.html)
30. [How to Enable Audit Logging for CQ DAM](https://aem4beginner.blogspot.com/2020/04/how-to-enable-audit-logging-for-cq-dam.html)
31. [How to Configure the Garbage Collection Log to Roll Over](https://aem4beginner.blogspot.com/2020/04/how-to-configure-garbage-collection-log.html)
32. [How to Rotate Logs in CQ/AEM](https://aem4beginner.blogspot.com/2020/04/how-to-rotate-logs-in-cq-aem.html)
33. [How to read Logs from Browser in CQ](https://aem4beginner.blogspot.com/2020/04/how-to-read-logs-from-browser-in-cq.html)
34. [View JSP Logging statements in CQ5/AEM](https://aem4beginner.blogspot.com/2020/04/view-jsp-logging-statements-in-cq5aem.html)
35. [Accessing Logs Through Browser In AEM 6.2](https://aem4beginner.blogspot.com/2020/04/accessing-logs-through-browser-in-aem-62.html)
36. [Send Mail via Log4j with SMTP Appender](https://aem4beginner.blogspot.com/2020/04/send-mail-via-log4j-with-smtp-appender.html)
37. [Multiple log files using Log4j appender](https://aem4beginner.blogspot.com/2020/04/multiple-log-files-using-log4j-appender.html)
38. [Audit Log Search](https://aem4beginner.blogspot.com/2020/04/audit-log-search.html)
39. [Logging Activates, Deactivates and Deletes](https://aem4beginner.blogspot.com/2020/04/logging-activates-deactivates-and.html)
40. [To enable logging for JSP pages](https://aem4beginner.blogspot.com/2020/04/to-enable-logging-for-jsp-pages.html)
41. [log4j.xml for crx (logging QueryImpl)](https://aem4beginner.blogspot.com/2020/04/log4jxml-for-crx-logging-queryimpl.html)
42. [AEM Interpreting the request.log](https://aem4beginner.blogspot.com/2020/04/aem-interpreting-requestlog.html)
43. [AEM Types of Audit log](https://aem4beginner.blogspot.com/2020/04/aem-types-of-audit-log.html)
44. [AEM log Level 0 to level 3 meaning](https://aem4beginner.blogspot.com/2020/04/aem-log-level-0-to-level-3-meaning.html)
45. [AEM Log files Basics](https://aem4beginner.blogspot.com/2020/04/aem-log-files-basics.html)
46. [How to download the AEM server log files through Felix console](https://aem4beginner.blogspot.com/2020/04/how-to-download-aem-server-log-files.html)
47. [How to tail log files of AEM in windows OS using Jtail.jar file](https://aem4beginner.blogspot.com/2020/04/how-to-tail-log-files-of-aem-in-windows.html)
48. [Request Log Analyzer to find the top 10 expensive queries in AEM 6.4](https://aem4beginner.blogspot.com/2020/04/request-log-analyzer-to-find-top-10.html)
49. [Log File rotation in AEM/CQ5](https://aem4beginner.blogspot.com/2020/04/log-file-rotation-in-aemcq5.html)
50. [Debug/Info Logging for a Particular Java Class in Adobe CQ 5](https://aem4beginner.blogspot.com/2020/04/debuginfo-logging-for-particular-java.html)
51. [Consuming AEM Logs into New Relic with FluentD](https://aem4beginner.blogspot.com/2020/04/consuming-aem-logs-into-new-relic-with.html)  
    [Audit Log Maintenance in AEM 6](https://aem4beginner.blogspot.com/2020/04/audit-log-maintenance-in-aem-6.html)
52. [SegmentNotFoundException observed in the AEM log file](https://aem4beginner.blogspot.com/2020/04/segmentnotfoundexception-observed-in.html)
53. [The SegmentNotFound Exceptions in the AEM 6.3 Log](https://aem4beginner.blogspot.com/2020/04/the-segmentnotfound-exceptions-in-aem.html)
54. [Types of Logs in an AEM](https://aem4beginner.blogspot.com/2020/04/types-of-logs-in-aem.html)
55. [How to change the log file name and location in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-change-log-file-name-and.html)
56. [Explore Request Log in AEM](https://aem4beginner.blogspot.com/2020/04/explore-request-log-in-aem.html)
57. [Changing the log level for a package – AEM](https://aem4beginner.blogspot.com/2020/04/changing-log-level-for-package-aem.html)
58. **Trouble Shooting:**
59. [Overview of Troubleshooting in AEM 6.2](https://aem4beginner.blogspot.com/2020/03/overview-of-troubleshooting-in-aem-62.html)
60. [What are the major Vulnerabilities and Security Issues in AEM](https://aem4beginner.blogspot.com/2020/03/what-are-major-vulnerabilities-and.html)  
    [How to unlock a page in AEM?](https://aem4beginner.blogspot.com/2020/03/how-to-unlock-page-in-aem.html)
61. [Enable CORS in AEM](https://aem4beginner.blogspot.com/2020/03/enable-cors-in-aem.html)  
    [AEM / Adobe CQ5: Set up CQ Cluster and Troubleshoot](https://aem4beginner.blogspot.com/2020/03/aem-adobe-cq5-set-up-cq-cluster-and.html)  
    [AEM / Adobe CQ5: Overview on Clustering](https://aem4beginner.blogspot.com/2020/03/aem-adobe-cq5-overview-on-clustering.html)
62. [href=tel issue for international numbers in AEM RTE](https://aem4beginner.blogspot.com/2020/03/hreftel-issue-for-international-numbers.html)
63. [Error handling using a Dispatcher](https://aem4beginner.blogspot.com/2020/03/error-handling-using-dispatcher.html)
64. [How to Debug AEM Page Load Performance Issues](https://aem4beginner.blogspot.com/2020/04/how-to-debug-aem-page-load-performance.html)
65. [How to Solve CQ Port Conflicts on Windows](https://aem4beginner.blogspot.com/2020/04/how-to-solve-cq-port-conflicts-on.html)
66. [Client Libraries are not Refreshing Automatically in AEM/CQ5](https://aem4beginner.blogspot.com/2020/04/client-libraries-are-not-refreshing.html)
67. [Nothing Showing in CRXDE Lite in AEM/CQ5](https://aem4beginner.blogspot.com/2020/04/nothing-showing-in-crxde-lite-in-aemcq5.html)
68. [How to Reduce CQ Disk I/O](https://aem4beginner.blogspot.com/2020/04/how-to-reduce-cq-disk-io.html)
69. [How to increase the Limit of the Largest Uploadable File](https://aem4beginner.blogspot.com/2020/04/how-to-increase-limit-of-largest-upload.html)
70. [How to Assess Disk Performance Using mongoperf](https://aem4beginner.blogspot.com/2020/04/how-to-assess-disk-performance-using.html)
71. [Performance Troubleshooting Case Study – CPU Getting Maxed Out](https://aem4beginner.blogspot.com/2020/04/performance-troubleshooting-case-study_6.html)
72. [Performance Troubleshooting Case Study – Slow Batch Job In AEM](https://aem4beginner.blogspot.com/2020/04/performance-troubleshooting-case-study.html)
73. [Developer Health Checks](https://aem4beginner.blogspot.com/2020/04/developer-health-checks.html)  
    [Health Checks in AEM](https://aem4beginner.blogspot.com/2020/04/health-checks-in-aem.html)
74. [Error Page Handler](https://aem4beginner.blogspot.com/2020/04/error-page-handler.html)
75. [Troubleshooting Slow Queries in AEM](https://aem4beginner.blogspot.com/2020/04/troubleshooting-slow-queries-in-aem.html)
76. [org.osgi.framework.BundleException:osgi.wiring.package=com.adobe.cq.sightly](https://aem4beginner.blogspot.com/2020/04/orgosgiframeworkbundleexceptionosgiwiri.html)
77. [org.osgi.framework.BundleException: osgi.wiring.package=com.adobe.cq.mcm.salesforce](https://aem4beginner.blogspot.com/2020/04/orgosgiframeworkbundleexception.html)
78. [java.lang.NoClassDefFoundError: com/day/cq/commons/Externalizer](https://aem4beginner.blogspot.com/2020/04/javalangnoclassdeffounderror.html)
79. [Failed to execute goal org.apache.felix:maven-scr-plugin](https://aem4beginner.blogspot.com/2020/04/failed-to-execute-goal.html)
80. [Resolving the issues while migrating the packages between AEM instances](https://aem4beginner.blogspot.com/2020/04/resolving-issues-while-migrating.html)
81. [io.jsonwebtoken,version=[0.7,1) -- Cannot be resolved](https://aem4beginner.blogspot.com/2020/04/iojsonwebtokenversion071-cannot-be.html)
82. [org.apache.sling.security.impl.ReferrerFilter Rejected empty referrer header for POST request in AEM](https://aem4beginner.blogspot.com/2020/04/orgapacheslingsecurityimplreferrerfilte.html)
83. [java.lang.RuntimeException: ConfigurationManagerImpl disabled](https://aem4beginner.blogspot.com/2020/04/javalangruntimeexception.html)
84. [java.lang.NullPointerException:null at com.adobe.granite.workflow.core.WorkflowSessionFactory.isSuperUser](https://aem4beginner.blogspot.com/2020/04/javalangnullpointerexceptionnull-at.html)
85. [org.apache.sling.api.resource.PersistenceException: Resource is not modifiable](https://aem4beginner.blogspot.com/2020/04/caused-by-orgapacheslingapiresourcepers.html)
86. [Unresolveable build extension: Error resolving version for plugin 'com.day.jcr.vault:maven-vault-plugin' / Unknown packaging: content-package while deploying through Maven](https://aem4beginner.blogspot.com/2020/04/unresolveable-build-extension-error.html)
87. [com.google.gson,version=[2.2,3) -- Cannot be resolved in bundle](https://aem4beginner.blogspot.com/2020/04/comgooglegsonversion223-cannot-be.html)
88. [org.osgi.framework.ServiceException: Service cannot be cast: javax.servlet.Servlet](https://aem4beginner.blogspot.com/2020/04/orgosgiframeworkserviceexception.html)
89. [NPE:currentDesign.getDoctype(currentStyle).toRequest(request)](https://aem4beginner.blogspot.com/2020/04/nullpointerexception.html)
90. [org.apache.sling.engine.impl.SlingRequestProcessorImpl service: Uncaught Throwable java.lang.LinkageError: loader constraint violation – AEM](https://aem4beginner.blogspot.com/2020/04/orgapacheslingengineimplslingrequestpro.html)
91. [Exception in thread "main" java.lang.NoSuchMethodError: org.slf4j.helpers.MessageFormatter.format(Ljava/lang/String;Ljava/lang/Object;)Lorg/slf4j/helpers/FormattingTuple;Jcr2davRepositoryFactory](https://aem4beginner.blogspot.com/2020/04/exception-in-thread-main.html)
92. [Caused by: java.lang.NoClassDefFoundError: javax/management/InstanceAlreadyExistsException while connecting to Oracle Database from OSGI](https://aem4beginner.blogspot.com/2020/04/caused-by-javalangnoclassdeffounderror.html)
93. [Enable CORS in AEM](https://aem4beginner.blogspot.com/2020/04/enable-cors-in-aem.html)  
    [AEM Troubleshooting](https://aem4beginner.blogspot.com/2020/04/aem-troubleshooting.html)
94. [Non-English Characters not displaying on a page](https://aem4beginner.blogspot.com/2020/04/if-you-are-facing-issue-with-display-of.html)
95. [/etc/cloudsettings/default/contexthub.kernel.js is not loading in AEM 6.4](https://aem4beginner.blogspot.com/2020/04/etccloudsettingsdefaultcontexthubkernel.html)  
    [How to Clear/Recompile the JSP classes in AEM 6.1](https://aem4beginner.blogspot.com/2020/04/how-to-clearrecompile-jsp-classes-in.html)  
    [Time zone difference in Author/Publishers](https://aem4beginner.blogspot.com/2020/04/time-zone-difference-in-authorpublishers.html)  
    [How to disable Geolocation feature](https://aem4beginner.blogspot.com/2020/04/how-to-disable-geolocation-feature.html)
96. [Removing the Cookie is not working in IE10 and IE11](https://aem4beginner.blogspot.com/2020/04/removing-cookie-is-not-working-in-ie10.html)
97. [window.location.href is not working while clicking on a hyperlink in IE](https://aem4beginner.blogspot.com/2020/04/windowlocationhref-is-not-working-while.html)
98. [checkout: com.day.jcr.vault.vlt.VltException: Unable to mount filesystem in AEM](https://aem4beginner.blogspot.com/2020/04/checkout-comdayjcrvaultvltvltexception.html)
99. [Browser cookie values are not set in IE10 and IE11](https://aem4beginner.blogspot.com/2020/04/browser-cookie-values-are-not-setting.html)
100. [Changing a .content.xml file locally is not reflecting in CRX after maven build](https://aem4beginner.blogspot.com/2020/04/changing-contentxml-file-locally-is-not.html)
101. [The bundle is not uploading after maven build](https://aem4beginner.blogspot.com/2020/04/bundle-is-not-uploading-after-maven.html)
102. [Uncaught TypeError: Cannot read property 'extend' of undefined' - cq.extend](https://aem4beginner.blogspot.com/2020/04/uncaught-typeerror-cannot-read-property.html)
103. [java.lang.UnsupportedOperationException: Deserialization not allowed for class [Ljava.lang.Object;](https://aem4beginner.blogspot.com/2020/04/javalangunsupportedoperationexception.html)
104. [com.day.cq.mailer.MailingException: org.apache.commons.mail.EmailException: Sending the email to the following server failed - AEM](https://aem4beginner.blogspot.com/2020/04/comdaycqmailermailingexception.html)
105. [com.adobe.cq.sightly,version=[2.5,3) -- Cannot be resolved](https://aem4beginner.blogspot.com/2020/04/comadobecqsightlyversion253-cannot-be.html)
106. [Plugin org.apache.maven.plugins:maven-dependency-plugin:2.10 or one of its dependencies could not be resolved](https://aem4beginner.blogspot.com/2020/04/plugin-orgapachemavenpluginsmaven.html)
107. [How to disable the minification for client libraries in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-disable-minification-for-client.html)
108. [An issue with turning off location tracker feature (Remove the geolocation store)](https://aem4beginner.blogspot.com/2020/04/issue-with-turning-off-location-tracker.html)
109. [AEM Component is in a satisfied state, not in active](https://aem4beginner.blogspot.com/2020/04/aem-component-is-in-satisfied-state-not.html)
110. [AEM Components not loading in Touch UI](https://aem4beginner.blogspot.com/2020/04/aem-components-not-loading-in-touch-ui.html)
111. [AEM Touch UI issues](https://aem4beginner.blogspot.com/2020/04/aem-touch-ui-issues.html)
112. [Uncaught Reference Error: CQ is not defined](https://aem4beginner.blogspot.com/2020/04/uncaught-referenceerror-cq-is-not.html)
113. [Custom bundles are in the installed state after AEM server restart](https://aem4beginner.blogspot.com/2020/04/custom-bundles-are-in-installed-state.html)
114. [javax.inject, version=[0.0,1) -- Cannot be resolved](https://aem4beginner.blogspot.com/2020/04/javaxinject-version001-cannot-be.html)
115. [Sidekick is not loading in AEM](https://aem4beginner.blogspot.com/2020/04/sidekick-is-not-loading-in-aem.html)
116. [Decompile AEM: How to get a jar from AEM](https://aem4beginner.blogspot.com/2020/04/decompile-aem-how-to-get-jar-from-aem_25.html)  
     [Encrypting/Decrypting data in AEM](https://aem4beginner.blogspot.com/2020/04/encryptingdecrypting-data-in-aem.html)  
     [Exception while browsing the tags in page creation wizard: The method transform(Object) of type new Transformer(){} must override a superclass method – AEM](https://aem4beginner.blogspot.com/2020/04/exception-while-browsing-tags-in-page.html)
117. **Performance Issues:**
118. [Improve the performance of the AEM websites](https://aem4beginner.blogspot.com/2020/04/improve-performance-of-aem-websites.html)
119. [Performance monitoring in AEM projects](https://aem4beginner.blogspot.com/2020/03/performance-monitoring-in-aem-projects.html)  
     [Quality, Security, Performance Checks in Adobe Cloud Manager brief overview](https://aem4beginner.blogspot.com/2020/03/quality-security-performance-checks-in.html)
120. [Overview of Workflow, Performance monitoring, Logs in AEM/Adobe CQ5](https://aem4beginner.blogspot.com/2020/03/overview-on-workflow-performance.html)  
     [Performance issues on AEM web sites](https://aem4beginner.blogspot.com/2020/03/performance-issues-on-aem-web-sites.html)  
     [Performance tuning](https://aem4beginner.blogspot.com/2020/03/performance-tuning.html)  
     [Performance tuning tips for Adobe CQ 5.X and AEM 6.X](https://aem4beginner.blogspot.com/2020/03/performance-tuning-tips-for-aem-5x.html)
121. [Workflow Purging for AEM Performance optimization](https://aem4beginner.blogspot.com/2020/03/workflow-purging-for-aem-performance.html)  
     [TAR Compaction for AEM Performance optimization](https://aem4beginner.blogspot.com/2020/03/tar-compaction-for-aem-performance.html)
122. [Performance issue in CRXDE IDE of AEM/Adobe CQ5](https://aem4beginner.blogspot.com/2020/03/performance-issue-in-crxde-ide-of.html)  
     [Performance issue in CRXDE IDE of AEM/Adobe CQ5](https://aem4beginner.blogspot.com/2020/03/performance-issue-in-crxde-ide-of.html)
123. [Miscellaneous Administration](https://aem4beginner.blogspot.com/2020/03/miscellaneous-administration.html)  
     [Create and analyze Heap Dump in AEM](https://aem4beginner.blogspot.com/2020/03/create-and-analyse-heap-dump-in-aem.html)
124. [Run online and offline Tar Compaction in AEM/Adobe CQ5](https://aem4beginner.blogspot.com/2020/03/run-online-and-offline-tar-compaction.html)
125. [Monthly Offline Maintenance of the AEM Repository](https://aem4beginner.blogspot.com/2020/04/monthly-offline-maintenance-of-aem.html)
126. [Improve Page Load Performance in Firefox](https://aem4beginner.blogspot.com/2020/04/improve-page-load-performance-in-firefox.html)
127. [Performance Tuning AEM 6 Deployments on MongoDB](https://aem4beginner.blogspot.com/2020/04/performance-tuning-aem-6-deployments-on.html)
128. [How to Improve Query Performance in Oak](https://aem4beginner.blogspot.com/2020/04/how-to-improve-query-performance-in-oak.html)
129. [Recommended Sequence of CQ Maintenance Operations](https://aem4beginner.blogspot.com/2020/04/recommended-sequence-of-cq-maintenance.html)
130. [AEM 6.1 Base Size](https://aem4beginner.blogspot.com/2020/04/aem-61-base-size.html)
131. [How to Set the Landing Page](https://aem4beginner.blogspot.com/2020/04/how-to-set-landing-page.html)
132. [How to Create a CQ “Author” Instance Without Geometrixx Content](https://aem4beginner.blogspot.com/2020/04/how-to-create-cq-author-instance.html)
133. [How to Determine How Long CQ Has Been Running](https://aem4beginner.blogspot.com/2020/04/how-to-determine-how-long-cq-has-been.html)
134. [How to Efficiently Copy Large Amounts of Content Between CQ Repositories](https://aem4beginner.blogspot.com/2020/04/how-to-efficiently-copy-large-amounts.html)
135. [Performance Engineering](https://aem4beginner.blogspot.com/2020/04/performance-engineering.html)
136. [Performance Troubleshooting Case Study – CPU Getting Maxed Out](https://aem4beginner.blogspot.com/2020/04/performance-troubleshooting-case-study_6.html)
137. [Performance Troubleshooting Case Study – Slow Batch Job In AEM](https://aem4beginner.blogspot.com/2020/04/performance-troubleshooting-case-study.html)
138. [Disaster Recovery Planning – Key Terms](https://aem4beginner.blogspot.com/2020/04/disaster-recovery-planning-key-terms.html)
139. [Disaster Recovery Strategies](https://aem4beginner.blogspot.com/2020/04/disaster-recovery-strategies.html)
140. [Horizontally Scaling Systems (Scale-Out)](https://aem4beginner.blogspot.com/2020/04/horizontally-scaling-systems-scale-out.html)
141. [Disk space growing on the CQ author server?](https://aem4beginner.blogspot.com/2020/04/disk-space-growing-on-cq-author-server.html)
142. [How to check disk usage in CRX](https://aem4beginner.blogspot.com/2020/04/how-to-check-disk-usage-in-crx.html)
143. [How to Assess Disk Performance Using mongoperf](https://aem4beginner.blogspot.com/2020/04/how-to-assess-disk-performance-using.html)
144. [How to Debug AEM Page Load Performance Issues](https://aem4beginner.blogspot.com/2020/04/how-to-debug-aem-page-load-performance.html)
145. [AEM (Adobe CQ) Performance on different Hardware](https://aem4beginner.blogspot.com/2020/04/aem-adobe-cq-performance-on-different.html)
146. [Performance list](https://aem4beginner.blogspot.com/2020/04/performance-list.html)
147. [w3c validation](https://aem4beginner.blogspot.com/2020/04/w3c-validation.html)
148. [XSS Protection Service](https://aem4beginner.blogspot.com/2020/04/xss-protection-service.html)
149. [How to Bookmark AEM URLs on Browser?](https://aem4beginner.blogspot.com/2020/04/how-to-bookmark-aem-urls-on-browser.html)
150. [HTML page construction standards for performance](https://aem4beginner.blogspot.com/2020/04/html-page-construction-standards-for.html)
151. [Improving performance with browser caching](https://aem4beginner.blogspot.com/2020/04/improving-performance-with-browser.html)
152. [How to take Online Backups?](https://aem4beginner.blogspot.com/2020/04/how-to-take-online-backups.html)
153. [Fixing a hanging author server](https://aem4beginner.blogspot.com/2020/04/fixing-hanging-author-server.html)
154. [Author server says "Loading" for a long, long, long time](https://aem4beginner.blogspot.com/2020/04/author-server-says-loading-for-long-time.html)
155. [Midnight at the lost and found](https://aem4beginner.blogspot.com/2020/04/midnight-at-lost-and-found.html)
156. [Lots of data\*.tar files in the author](https://aem4beginner.blogspot.com/2020/04/lots-of-datatar-files-in-author.html)
157. [AEM JVM VM Behavioral Options](https://aem4beginner.blogspot.com/2020/04/aem-jvm-vm-behavioral-options.html)
158. [AEM JVM GC options](https://aem4beginner.blogspot.com/2020/04/aem-jvm-gc-options.html)
159. [Performance tuning](https://aem4beginner.blogspot.com/2020/04/performance-tuning.html)
160. [AEM JVM VM Performance Options](https://aem4beginner.blogspot.com/2020/04/aem-jvm-vm-performance-options.html)
161. [AEM JVM VM options for Debugging](https://aem4beginner.blogspot.com/2020/04/aem-jvm-vm-options-for-debugging.html)
162. [AEM 6.3 Maintenance activities or Performance tuning](https://aem4beginner.blogspot.com/2020/04/aem-63-maintenance-activities-or.html)
163. [How to generate Heap dump or run Garbage collection through Felix console](https://aem4beginner.blogspot.com/2020/04/how-to-generate-heap-dump-or-run.html)
164. [AEM6.3 Performance Tuning tips](https://aem4beginner.blogspot.com/2020/04/aem63-performance-tuning-tips.html)
165. [AEM performance Analysis from top to Bottom for each layer of the system](https://aem4beginner.blogspot.com/2020/04/aem-performance-analysis-from-top-to.html)
166. [Performance Tree -Identification of Issues in Different layers of the AEM system](https://aem4beginner.blogspot.com/2020/04/performance-tree-identification-of.html)
167. [AEM Common critical issues and their fixes](https://aem4beginner.blogspot.com/2020/04/aem-common-critical-issues-and-their.html)
168. [AEM Performance analysis using built-in profiler](https://aem4beginner.blogspot.com/2020/04/aem-performance-analysis-using-built-in.html)
169. [AEM Basics of Performance Common issues and how to prevent them](https://aem4beginner.blogspot.com/2020/04/aem-basics-of-performance-common-issues.html)
170. [AEM How to find requests with long duration times](https://aem4beginner.blogspot.com/2020/04/aem-how-to-find-requests-with-long.html)
171. [Apache benchmarking tool to identify the HTTP server performance along with AEM server Load Testing](https://aem4beginner.blogspot.com/2020/04/apache-bench-marking-tool-to-identify.html)
172. [AEM Tools for Monitoring and Analyzing Performance](https://aem4beginner.blogspot.com/2020/04/aem-tools-for-monitoring-and-analyzing.html)
173. [Performance Tuning in AEM](https://aem4beginner.blogspot.com/2020/04/performance-tuning-in-aem.html)
174. [Performance & Load Testing AEM](https://aem4beginner.blogspot.com/2020/04/performance-load-testing-aem.html)
175. [Real full-stack performance monitoring through Dynatrace Tool](https://aem4beginner.blogspot.com/2020/04/real-full-stack-performance-monitoring.html)
176. [Performance tuning for production-ready instance](https://aem4beginner.blogspot.com/2020/04/performance-tuning-for-production-ready.html)
177. [Performance Tuning in Informatica](https://aem4beginner.blogspot.com/2020/04/performance-tuning-in-informatica.html)
178. [Performance Optimization tools for AEM](https://aem4beginner.blogspot.com/2020/04/performance-optimization-tools-for-aem.html)
179. [Performance Dashboard AEM](https://aem4beginner.blogspot.com/2020/04/performance-dashboard-aem.html)
180. [Improve Web Page Performance](https://aem4beginner.blogspot.com/2020/04/improve-web-page-performance.html)
181. [How to Use JMeter for Performance & Load Testing-Chapter 5](https://aem4beginner.blogspot.com/2020/04/how-to-use-jmeter-for-performance-load.html)
182. [Building Health Checks for AEM](https://aem4beginner.blogspot.com/2020/04/building-health-checks-for-aem.html)
183. [How to monitor the Performance of AEM instance using JVisualVM](https://aem4beginner.blogspot.com/2020/04/how-to-monitor-performance-of-aem.html)
184. [AEM Sites Performance Issues](https://aem4beginner.blogspot.com/2020/04/aem-sites-performance-issues.html)
185. [AEM Assets Performance Issues & How to Fix](https://aem4beginner.blogspot.com/2020/04/aem-assets-performance-issues-how-to-fix.html)
186. [AEM Memory Issues and How to Fix](https://aem4beginner.blogspot.com/2020/04/aem-memory-issues-and-how-to-fix.html)
187. [AEM Tune your JVM For Better Performance](https://aem4beginner.blogspot.com/2020/04/aem-tune-your-jvm-for-better-performance.html)
188. [What are the different ways to run the garbage collector in AEM through Felix Console](https://aem4beginner.blogspot.com/2020/04/what-are-different-ways-to-run-garbage.html)
189. [How to refresh all open Session in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-refresh-all-open-session-in-aem.html)
190. [How to perform Re-indexing for the propertyindexAync in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-perform-re-indexing-for.html)
191. [Revision GC through JMX console in AEM](https://aem4beginner.blogspot.com/2020/04/revisiongc-through-jmx-console-in-aem.html)
192. [How to run the Full Garbage collection in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-run-full-garbage-collection-in.html)
193. [Data Store Garbage Collection AEM 6.x](https://aem4beginner.blogspot.com/2020/04/data-store-garbage-collection-aem-6x.html)
194. [How to take online repository backup and perform the Datastore GC and revision GC for AEM 6.3](https://aem4beginner.blogspot.com/2020/04/how-to-take-online-repository-backup.html)
195. [Full Garbage collection for BLOB Garbage](https://aem4beginner.blogspot.com/2020/04/full-garbage-collection-for-blob-garage.html)
196. [What is the repository Backup Sequence in AEM](https://aem4beginner.blogspot.com/2020/04/what-is-repository-backup-sequence-in.html)
197. [How to Troubleshooting –Rapid DataStore growth in AEM 6.3](https://aem4beginner.blogspot.com/2020/04/how-to-troubleshooting-rapid-datastore.html)
198. [Why GC takes a too long time to finish in AEM6.3](https://aem4beginner.blogspot.com/2020/04/why-gc-takes-too-long-time-to-finish-in.html)
199. [How to Troubleshoot Missing Blobs in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-troubleshoot-missing-blobs-in-aem.html)
200. [How to reset the Repository ID in AEM](https://aem4beginner.blogspot.com/2020/04/how-to-reset-repository-id-in-aem.html)
201. [How to run online and offline Consistency Check in AEM through JMX console](https://aem4beginner.blogspot.com/2020/04/how-to-run-online-and-offline.html)
202. [Managing AEM Datastore adobe official document](https://aem4beginner.blogspot.com/2020/04/managing-aem-datastore-adobe-official.html)
203. [AEM OutOfMemoryError: Java heap space or OutOfMemoryError: GC overhead limit exceeded.](https://aem4beginner.blogspot.com/2020/04/aem-outofmemoryerror-java-heap-space-or.html)
204. [Heap Dump Generation in AEM through JMX console](https://aem4beginner.blogspot.com/2020/04/heap-dump-generation-in-aem-through-jmx.html)
205. [Java VM Diagnostic commands to run GC, Finalization, vmSystemProperties, vmVersion, vmUptime, and vmCommand line](https://aem4beginner.blogspot.com/2020/04/java-vm-diagnostic-commands-to-run-gc.html)
206. [How to clear blocked queue or forced-to retry blocked queue in AEM through JMX console](https://aem4beginner.blogspot.com/2020/04/how-to-clear-blocked-queue-or-forced-to.html)
207. [When to perform Re-Indexing in AEM](https://aem4beginner.blogspot.com/2020/04/when-to-perform-re-indexing-in-aem.html)
208. [In AEM why backup\_.tar files are created](https://aem4beginner.blogspot.com/2020/04/in-aem-why-backuptar-files-are-created.html)
209. [AEM How to identify the check session count](https://aem4beginner.blogspot.com/2020/04/aem-how-to-identify-check-session-count.html)
210. [AEM Command to check CPU usage & how to interpret the results](https://aem4beginner.blogspot.com/2020/04/aem-command-to-check-cpu-usage-how-to.html)
211. [AEM CPU at 100% -Analyze slow and blocked processes](https://aem4beginner.blogspot.com/2020/04/aem-cpu-at-100-analyze-slow-and-blocked.html)
212. [AEM How to take a Thread Dump](https://aem4beginner.blogspot.com/2020/04/aem-how-to-take-thread-dump.html)
213. [AEM Analysis of Unclosed sessions](https://aem4beginner.blogspot.com/2020/04/aem-analysis-of-unclosed-sessions.html)
214. [AEM Indexing Issues & How to Fix](https://aem4beginner.blogspot.com/2020/04/aem-indexing-issues-how-to-fix.html)
215. [AEM6.3 How to clear or extend the reporting Cache](https://aem4beginner.blogspot.com/2020/04/aem63-how-to-clear-or-extend-reporting.html)  
     [AEM6.3 How to clear the Facet Cache](https://aem4beginner.blogspot.com/2020/04/aem63-how-to-clear-facet-cache.html)  
     [AEM Common How to identify Slow running Queries.](https://aem4beginner.blogspot.com/2020/04/aem-common-how-to-identify-slow-running.html)  
     [Revision Cleanup](https://aem4beginner.blogspot.com/2020/04/revision-cleanup.html)  
     [AEM Offline Compaction](https://aem4beginner.blogspot.com/2020/04/aem-offline-compaction.html)  
     [How to set the Oak login token session expiration](https://aem4beginner.blogspot.com/2020/04/how-to-set-oak-login-token-session.html)  
     [How to enable Debug mode in AEM & find unclosed Sessions](https://aem4beginner.blogspot.com/2020/04/how-to-enable-debug-mode-in-aem-find.html)
216. [How to find the no of open sessions, segment repo size, and Histograms](https://aem4beginner.blogspot.com/2020/04/how-to-find-no-of-open-sessions-segment.html)
217. [Set the Oak Login token session expiration](https://aem4beginner.blogspot.com/2020/04/set-oak-login-token-session-expiration.html)
218. [Change AEM Default Session timeout value](https://aem4beginner.blogspot.com/2020/04/change-aem-default-session-timeout-value.html)
219. [AEM Common Security checklist for Production Server](https://aem4beginner.blogspot.com/2020/04/aem-common-security-checklist-for.html)
220. [AEM What is the average number of page activations per day](https://aem4beginner.blogspot.com/2020/04/aem-what-is-average-number-of-page.html)
221. [AEM What causes CPU at 0% consumption](https://aem4beginner.blogspot.com/2020/04/aem-what-causes-cpu-at-0-consumption.html)
222. [AEM Typical situations causing high CPU consumption](https://aem4beginner.blogspot.com/2020/04/aem-typical-situations-causing-high-cpu.html)
223. [AEM DISK I/O Running out of Disk Space issues](https://aem4beginner.blogspot.com/2020/04/aem-disk-io-running-out-of-disk-space.html)
224. [How to restart or stop the AEM Framework only](https://aem4beginner.blogspot.com/2020/04/how-to-restart-or-stop-aem-framework.html)
225. [How to find the server uptime or last started](https://aem4beginner.blogspot.com/2020/04/how-to-find-server-uptime-or-last.html)
226. [How to find out the number of processors](https://aem4beginner.blogspot.com/2020/04/how-to-find-out-number-of-processors.html)
227. [How many req per hour do you have on the author or publish system](https://aem4beginner.blogspot.com/2020/04/aem-how-many-requests-per-hour-do-you.html)
228. [AEM How to find how many authors are working with the system](https://aem4beginner.blogspot.com/2020/04/aem-how-to-find-how-many-authors-are.html)
229. [AEM6.3 How to increase the limit for the Uploadable file](https://aem4beginner.blogspot.com/2020/04/aem63-how-to-increase-limit-for.html)
230. [How to protect your AEM instances from Google searches](https://aem4beginner.blogspot.com/2020/04/how-to-protect-your-aem-instances-from.html)
231. [Adobe Experience Manager Crashes during large asset uploads](https://aem4beginner.blogspot.com/2020/04/adobe-experience-manager-crashes-during.html)
232. [How to find the Binaries location on the file system](https://aem4beginner.blogspot.com/2020/04/how-to-find-binaries-location-on-file.html)
233. [Steps to Tune the Sling Job Queues](https://aem4beginner.blogspot.com/2020/04/steps-to-tune-sling-job-queues.html)
234. [Disk usage is in AEM is abnormally and rapidly increasing on an AEM server.](https://aem4beginner.blogspot.com/2020/04/disk-usage-is-in-aem-63-server-is.html)
235. [How to find the number of Processor on which AEM server is running](https://aem4beginner.blogspot.com/2020/04/how-to-find-number-of-processor-on.html)
236. [AEM Version Purging Basics](https://aem4beginner.blogspot.com/2020/04/aem-version-purging-basics.html)
237. [AEM How to Purge Versions of a Web Site](https://aem4beginner.blogspot.com/2020/04/aem-how-to-purge-versions-of-web-site.html)
238. [AEM Interpretation of Version purging results](https://aem4beginner.blogspot.com/2020/04/aem-interpretation-of-version-purging.html)
239. [AEM IBM Thread Analyzer guide](https://aem4beginner.blogspot.com/2020/04/aem-ibm-thread-analyzer-guide.html)
240. [Top 5 server-side programming languages for websites](https://aem4beginner.blogspot.com/2020/04/top-5-server-side-programming-languages.html)  
     [Performance Tuning in AEM](https://aem4beginner.blogspot.com/2020/04/performance-tuning-adobe-cq5adobe-aem.html)  
     [Debugging Slowness in your AEM instance: Part 1](https://aem4beginner.blogspot.com/2020/04/debugging-slowness-in-your-aem-instance_25.html)  
     [Debugging Slowness in your AEM instance: Part 2](https://aem4beginner.blogspot.com/2020/04/debugging-slowness-in-your-aem-instance.html)  
     [Developing a caching service to cache third part elements to improve performance in AEM](https://aem4beginner.blogspot.com/2020/04/developing-caching-service-to-cache.html)