1. **Explain about AEM?**

AEM (Adobe Experience Manager) is a java based content management system from Adobe.  It is an enterprise grade WCMS that allows you to build compelling content centric applications that combine Web Content Management, Digital Asset Management, Analytics and Social Collaboration. AEM can be used to perform below tasks:s

* Build, author and publish websites, complete with enforcement of corporate design and user access control of editing and publishing rights.
* Manage a repository of Digital Assets such as Images, Videos & Documents and integrate these assets into your website.
* Supports inbuilt search mechanism to search content, no matter where it is stored in your repository.
* Set up Social Collaboration Tools like Blogs, User Groups and Calendars.
* Organize your Digital Assets & Web Pages using Tagging.
* Plan, design, launch and optimize marketing campaigns.
* It is based on a content repository to store the content of a website and use JCR (java content repository) specification to access the content repository.
* It uses Restful Apache Sling framework to map request url to the corresponding node in content repository
* It uses powerful OSGi framework internally to allow modular application development. It means individual pieces of your application (called bundles in terms of OSGi) can be independently started and stopped.
* It uses Apache Felix as the OSGi container. Therefore different parts of cq5 can be independently started and stopped.

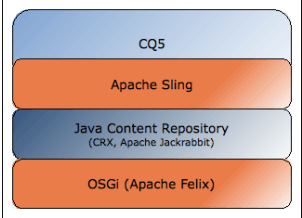
1. **What are the different versions of AEM so far?**

Communiqué 1, Communiqué 2, Communiqué 3 (2002), Communiqué 4 (2005), Communiqué 5 (2008), CQ5.0, CQ5.2.0, CQ5.2.1, CQ5.3, CQ5.4, CQ5.5, AEM 5.6, AEM 5.6.1, AEM 6.0, AEM 6.1, AEM 6.2, AEM 6.3

1. **What is the technology stack used in AEM?**

AEM uses the following technologies:

* **JCR –** Java specification for accessing a content repository JSR-283 specification jcr 2.0, cq5 uses its own implementation of jcr called CRX. Apache Jackrabbit is an open-source implementation of jcr 2.0 specification.
* **Apache Sling –** RESTful framework to access a jcr over http protocol. It maps the request url to the node in jcr.
* **OSGi (Apache Felix) –** Framework for modular application development using java. Each module called bundle can be independently started and stopped. OSGi container which provides implementation classes for OSGi framework.



1. **What are the advantages of**[**AEM**](http://www.aemcq5tutorials.com/)**over other CMS?**

Below are the advantages of AEM/CQ5 over other CMS (Content management System):-

* A well defined end to end solution for Multi Brand, Multi Site, Multi Locale CMS.
* AEM is more of a cloud clubbed together with multiple Open Source Technologies like OSGi, JCR, Apache Sling, Java, etc such that it is easily scalable in any direction.
* Concept of Author & Publish Servers and the handling of the content among them is really good and well defined (Replication & Reverse Replication).
* Completely loosely coupled architecture where Content and Code are totally separate and scalable.
* Inbuilt high performance Repository such that no additional database is required in the system.
* Implementation of workflows for creating, editing and publishing of content.
* Managing a repository of digital assets like images, documents and integrating them to the websites.
* Usage of search queries to find content no matter where it is stored in your organization.
* Setting up easily the social collaboration blogs, groups.
* Tagging utility to organize the digital assets such as images.

1. **What is Apache Sling Framework?**

AEM is built using Sling, a Web application framework based on REST principles that provides easy development of content-oriented applications. Sling uses a JCR repository, such as Apache Jackrabbit, or in the case of AEM, the CRX Content Repository, as its data store.

Apache Sling in five bullets points.

* REST based web framework.
* Content-driven, using a JCR content repository.
* Powered by OSGi
* Scripting inside, multiple languages (JSP, server-side javascript, Scala, etc.)
* Apache Open Source project

1. **Is Sling Content Centric? Explain.**

Sling is content-centric. This means that processing is focused on the content as each (HTTP) request is mapped onto content in the form of a JCR resource (a repository node):

* the first target is the resource (JCR node) holding the content
* secondly, the representation, or script, is located from the resource properties in combination with certain parts of the request (e.g. selectors and/or the extension)

1. **Explain about Repository in AEM.**

AEM is built on top of Adobe's CRX platform. CRX is a data storage system specifically designed for content-centric applications. AEM uses this content repository to store all its web content, digital assets, scripts, Java libraries, configuration information and other data. CRX implements the Content Repository API for Java Technology (JCR). This standard defines a data model and application programming interface (that is, a set of commands) for content repositories.

A content repository, as defined by JCR, combines features of the traditional relational database with those of a conventional file system.

File system-like features supported by JCR include:

* Hierarchy: Content in a JCR repository can be addressed by path. This is useful when delivering content to the web since most websites are also organized hierarchically.
* Semi-structured content: JCR can store structured documents, like XML, either as opaque files (as a file system would) or as structures ingested directly into the JCR hierarchy.
* Access Control and Locking: JCR can restrict access to different parts of the content hierarchy based on policies or ACLs. It also supports locking of content to prevent conflicts.

1. **Why to use OAK Indexing?**

Unlike Jackrabbit 2, Oak does not index content by default. Custom indexes need to be created when necessary, much like with traditional relational databases. If there is no index for a specific query then the whole repository will be traversed.

1. **What is the difference between CRX2 & CRX3?**

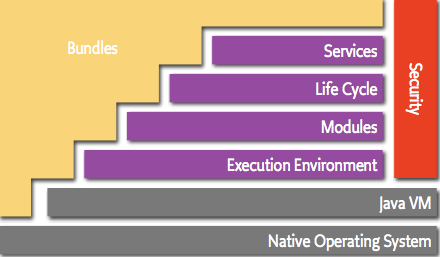
| **CRX 2** | **CRX 3** |
| --- | --- |
| CRX 2 is extended from Jackrabbit. | CRX3 is extended from Jackrabbit OAK. |
| JackRabbit is a pure JCR implementation. | OAK uses a three tier architecture with NODE STATE MODEL that uses JCR just as a facade |
| Persistence Manager is used to store data in JackRabbit that allows the content to be written to the persistence layer as a blob. | In OAK, Microkernels write data as native structures of the underlying Database used. For eg. Mongo DB data is written as documents |
| Jackrabbit runs on LUCENE. | OAK supports SOLR indexing implicitly |
| CRX2 datastore is on the filesystem by default. | CRX3 supports multiple configurations on DataStore (binary data storage). By default it is implicit |

1. **What is OSGi?**

The OSGi Alliance, formerly known as the Open Services Gateway initiative. The OSGi specification describes a modular system and a service platform for the Java programming language that implements a complete and dynamic component model, something that does not exist in standalone Java/VM environments. Applications or components, coming in the form of bundles for deployment, can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot; management of Java packages/classes is specified in great detail.

Application life cycle management is implemented via APIs that allow for remote downloading of management policies. The service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly.

The OSGi has a layered model that is depicted in the following figure.



1. **What are the layers present in OSGi?**

The OSGi has a layered model. The following list contains a short definition of the terms:

* Bundles – Bundles are normal jar components with extra manifest headers.
* Services – The service layer, which hold the service-side of the framework, keeps the service registry and manages it.
* Life-Cycle – The lifecycle layer manages and keeps track of the frameworks and bundles lifecycle state. It is used to install or uninstall framework objects and start or stop them.
* Modules – The module layer, which is the bundle space, holds the bundles that are installed on the framework and are managed through the lifecycle layer.
* Security – The security layer, which extends the jave 2 security architecture, is optional. When active, it validate the bundle signatures and controls the component access rights .
* Execution Environment – The execution environment layer, which is the bottom layer on which the bundles live, is selected to fit the underlying hardware or operating system.

1. **What are the lifecycle of the OSGi bundles?**

The bundle life cycle states are as follows.

* INSTALLED: The bundle has been successfully installed. The framework knowsenough about this bundle to attempt to load it.
* RESOLVED: All resources needed for this bundle have been loaded successfully andthe bundle is ready to be started. This is also the state the bundle would be in, once successfully stopped.
* STARTING: The bundle is being started, but has not finished starting.
* ACTIVE: The bundle has been successfully activated and is running, ready to be used.
* STOPPING: The bundle is being stopped, but has not finished stopping.
* UNINSTALLED: The bundle has been uninstalled. Once uninstalled, nothing can be done with the module.

1. **What are the differences between a JAR & an OSGi Bundle?**

| **JAR** | **Bundle** |
| --- | --- |
| have manifest file | have manifest file |
| They are not meant to be run in an OSGi runtime container | They are meant to be run in an OSGi runtime container |
| To create jar, we have to select jar option into packaging tag in pom.xml | To create bundle, we have to put "bundle" as value into packaging tag in pom.xml |
| you can consume jar by calling its public methods | You can consume bundle by calling their services. Which are mentioned in export packages of manifest file |

1. **What are Configuration Services in OSGi?**

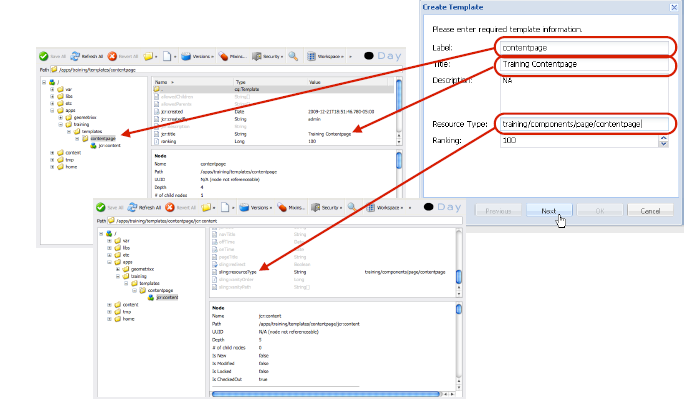
The OSGi Componendium Configuration Admin Service specifies a service, which allows for easy management of configuration data for configurable components. Basicaly configuration is a list of name-value pairs. Configuration is managed by management applications by asking the Configuration Admin Service for such configuration. After updating the configuration, it is sent back to the Configuration Admin Service. The Configuration Admin Service is like a central hub, which cares for persisting this configuration and also for distributing the configuration to interested parties. One class of such parties are the components to be configured. These are registered as ManagedService services. There is also a notion of ManagedServiceFactory, which allows for multiple configurations of the same kind to be applied.

1. **What are the benefits of using OSGi?**

**The following are the benefits of using OSGi :**

* **Reduced Complexity** – Developing with OSGi technology means developing bundles: the OSGi components. Bundles are modules. They hide their internals from other bundles and communicate through well defined services. Hiding internals means more freedom to change later. This not only reduces the number of bugs, it also makes bundles simpler to develop because correctly sized bundles implement a piece of functionality through well defined interfaces.
* **Reuse** – The OSGi component model makes it very easy to use many third party components in an application. An increasing number of open source projects provide their JARs ready made for OSGi. However, commercial libraries are also becoming available as ready made bundles.
* **Real World** – The OSGi framework is dynamic. It can update bundles on the fly and services can come and go. Developers used to more traditional Java see this as a very problematic feature and fail to see the advantage. However, it turns out that the real world is highly dynamic and having dynamic services that can come and go makes the services a perfect match for many real world scenarios. For example, a service could model a device in the network. If the device is detected, the service is registered. If the device goes away, the service is unregistered. There are a surprising number of real world scenarios that match this dynamic service model. Applications can therefore reuse the powerful primitives of the service registry (register, get, list with an expressive filter language, and waiting for services to appear and disappear) in their own domain. This not only saves writing code, it also provides global visibility, debugging tools, and more functionality than would have implemented for a dedicated solution. Writing code in such a dynamic environment sounds like a nightmare, but fortunately, there are support classes and frameworks that take most, if not all, of the pain out of it.
* **Easy Deployment** – The OSGi technology is not just a standard for components. It also specifies how components are installed and managed. This API has been used by many bundles to provide a management agent. This management agent can be as simple as a command shell, a [TR-69](http://www.ecma-international.org/publications/techreports/E-TR-069.htm) management protocol driver, [OMA DM](http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html) protocol driver, a cloud computing interface for Amazon’s EC2, or an [IBM Tivoli](http://www-306.ibm.com/software/tivoli/) management system. The standardized management API makes it very easy to integrate OSGi technology in existing and future systems.
* **Dynamic Updates** – The OSGi component model is a dynamic model. Bundles can be installed, started, stopped, updated, and uninstalled without bringing down the whole system. Many Java developers do not believe this can be done reliably and therefore initially do not use this in production. However, after using this in development for some time, most start to realize that it actually works and significantly reduces deployment times.
* **Adaptive** – The OSGi component model is designed from the ground up to allow the mixing and matching of components. This requires that the dependencies of components need to be specified and it requires components to live in an environment where their optional dependencies are not always available. The OSGi service registry is a dynamic registry where bundles can register, get, and listen to services. This dynamic service model allows bundles to find out what capabilities are available on the system and adapt the functionality they can provide. This makes code more flexible and resilient to changes.
* **Transparency** – Bundles and services are first class citizens in the OSGi environment. The management API provides access to the internal state of a bundle as well as how it is connected to other bundles. For example, most frameworks provide a command shell that shows this internal state. Parts of the applications can be stopped to debug a certain problem, or diagnostic bundles can be brought in. Instead of staring at millions of lines of logging output and long reboot times, OSGi applications can often be debugged with a live command shell.
* **Versioning** – OSGi technology solves JAR hell. JAR hell is the problem that library A works with library B;version=2, but library C can only work with B;version=3. In standard Java, you’re out of luck. In the OSGi environment, all bundles are carefully versioned and only bundles that can collaborate are wired together in the same class space. This allows both bundle A and C to function with their own library. Though it is not advised to design systems with this versioning issue, it can be a life saver in some cases.
* **Simple** – Using OSGi is surprisingly simple, despite the powerful dependency management, configuration, and dynamics, OSGi code looks almost identical to classic Java code. A number of easy-to-use annotations tell the runtime how a particular class wants to use the dynamics, configuration, and dependencies on other services. The defaults completely hide the dynamics and OSGi. This very simple model allows the gradual use of more advanced features.
* **Small** – The OSGi Release 4 Framework can be implemented in about a 300KB JAR file. This is a small overhead for the amount of functionality that is added to an application by including OSGi. OSGi therefore runs on a large range of devices: from very small, to small, to mainframes. It only asks for a minimal Java VM to run and adds very little on top of it.
* **Fast** – One of the primary responsibilities of the OSGi framework is loading the classes from bundles. In traditional Java, the JARs are completely visible and placed on a linear list. Searching a class requires searching through this (often very long, 150 is not uncommon) list. In contrast, OSGi pre-wires bundles and knows for each bundle exactly which bundle provides the class. This lack of searching is a significant speed up factor at startup.
* **Lazy** – Lazy in software is good and the OSGi technology has many mechanisms in place to do things only when they are really needed. For examples, bundles can be started eagerly, but they can also be configured to only start when another bundle is using them. Services can be registered but only created when they are used. The specifications have been optimized several times to allow for these kind of lazy scenarios that can save tremendous runtime costs.
* **Secure** – Java has a very powerful fine grained security model at the bottom but it has turned out very hard to configure in practice. The result is that most secure Java applications are running with a binary choice: no security or very limited capabilities. The OSGi security model leverages the fine grained security model but improves the usability (as well as hardening the original model) by having the bundle developer specify the requested security details in an easily audited form while the operator of the environment remains fully in charge. Overall, OSGi likely provides one of the most secure application environments that is still usable short of hardware protected computing platforms.
* **Humble** – Many frameworks take over the whole VM, they only allow one instance to run in a VM. The flexibility of the OSGi specifications is demonstrated by how it can even run inside a J2EE Application Server. Many developers wanted to run OSGi but their companies did not allow them to deploy normal JARs. Instead, they included an OSGi framework in their WAR file and loaded their bundles from the file system or over the network. OSGi is so flexible that one application server can easily host multiple OSGi frameworks.
* **Non Intrusive** – Applications (bundles) in an OSGi environment are left to their own. They can use virtually any facility of the VM without the OSGi restricting them. Best practice in OSGi is to write Plain Old Java Objects and for this reason, there is no special interface required for OSGi services, even a Java String object can act as an OSGi service. This strategy makes application code easier to port to another environment.
* **Runs Everywhere** – Well, that depends. The original goal of Java was to run anywhere. Obviously, it is not possible to run all code everywhere because the capabilities of the Java VMs differ. A VM in a mobile phone will likely not support the same libraries as an IBM mainframe running a banking application. There are two issue to take care of. First, the OSGi APIs should not use classes that are not available on all environments. Second, a bundle should not start if it contains code that is not available in the execution environment. Both of these issues have been taken care of in the OSGi specifications.
* **Widely Used** – The OSGi specifications started out in the embedded home automation market but since 1998 they have been extensively used in many industries: automotive, mobile telephony, industrial automation, gateways & routers, private branch exchanges, fixed line telephony, and many more. Since 2003, the highly popular Eclipse Integrated Development Environment runs on OSGi technology and provides extensive support for bundle development. In the last few years, OSGi has been taken up by the enterprise developers. Eclipse developers discovered the power of OSGi technology but also the Spring Framework helped popularize this technology by creating a specific extension for OSGi. Today, you can find OSGi technology at the foundation of IBM Websphere, SpringSource dm Server, Oracle (formerly BEA) Weblogic, Sun’s GlassFish, and Redhat’s JBoss.
* **Supported by Key Companies** – OSGi counts some of the largest computing companies from a diverse set of industries as its [members](https://www.osgi.org/about-us/members/). Members include Oracle, IBM, Samsung, Nokia, Progress, Motorola, NTT, Siemens, Hitachi, Deutsche Telekom, Redhat, Ericsson, and many more.

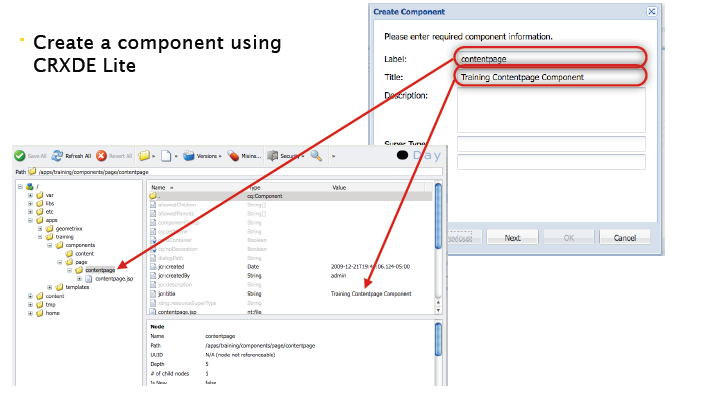
1. **What is a Template?**

An AEM template enables you to define a consistent style for the pages in your application. A template comprises of nodes that specify the page structure. A template defines the structure of content on a page, including images, font types and other properties. Templates are comprised of components and can be applied to newly created pages or existing pages. Whenever you create a new page, you need to select the correct template. This is an important step, as the templates define the design, structure and content of the webpages. Also, once you’ve selected a template and created a new page, you cannot change the template agai

1. **What is a Component?**

Components are re-usable modules that implement specific application logic to render the content of your web site. You can think of a component as a collection of scripts (JSP or Sightly or Handlebars - 3 types of script supported in AEM). Components are building blocks of content with specific functionality and they differ based on templates. When you create a new page using a template, you can add content to the page by using Components. Can include other components.

2 types of components: Page Components (typically referenced by a template) & Other Components



| **dialog** | **design-dialog** |
| --- | --- |
| Dialog will change the content at the page level. | Design dialog will change the content at the template level. |
| authored in edit mode. | authored in design mode. |
| node name should be dialog. | node name should be design\_dialog. |
| stored under pages jcr:content node. | stored under design page located under /etc/design/default. |
| we can get value from properties object. | we can get design dialog value from currentStyle object. |
| jcr:primaryType is cq:Dialog. | jcr:primaryType is cq:Dialog. |

1. **What is Dialog, Design Dialog & CQ:Dialog?**

cq:dialog - It is the dialogs for the touch-optimized UI(editor.html).

* It uses the Granite UI framework.
* Node name is cq:dialog.
* jcr:primaryType = nt:unstructured, sling:resourceType = cq/gui/components/authoring/dialog for cq:dialog

1. **Why we need to include global.jsp if we are creating a component (in JSP or Sightly)?**

  The global.jsp script which adobe provides by default declares Sling, AEM and JSTL taglibs to make component creation easy in AEM. Also, if we do not include this, the Sidekick and Component Edit Tools are not rendered on the page.

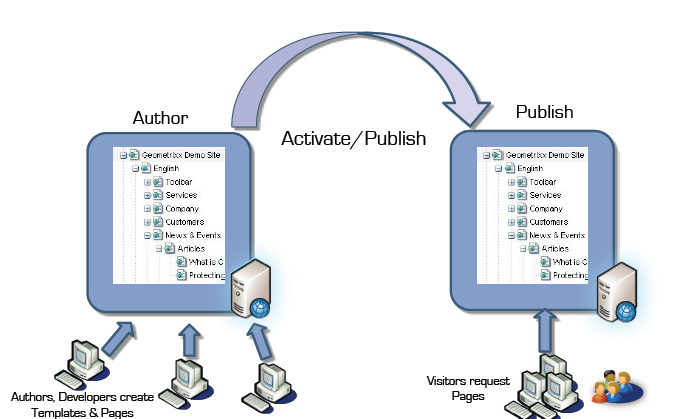
1. **Which script you should include to display sidekick?**

 init.jsp should be included in our JSP or script file to display sidekick. If you are including global.jsp, init.jsp is usually called from inside the global.jsp. But, for some reason if you have overridden the global.jsp in your project, you should make sure that the new custom global.jsp has init.jsp included in it.

1. **What are the Run Modes in AEM?**

AEM has basically two Run Modes - Author and Publish.

Author is meant for the Business Users/Authors to create and edit Content (Pages & Assets). Once an Author is done with a page, he can Activate the Page. Activation causes a Page/Asset to be replicated on a Publish Instance, that is meant for the Public Users to view. This process is called REPLICATION. Content cannot be edited in the Publish Instance (here its READ ONLY). Only User Generated content like Comments and Blogs can be entered in the Publish and these are transported to the Author using the concept called REVERSE REPLICATION.



1. **What are the different interfaces available in AEM?**

The important interfaces/screens available in AEM are:

* **CRX Explorer - http://localhost:4502/crx/explorer/index.jsp** - Create Users & System Users (Click on User Administration Link on this page)
* **CRX DE Lite - http://localhost:4502/crx/de/index.jsp -** IDE for coding through Components, Templates and manage Clientlibs, Debug Content, etc.
* **CRX Package Manager - http://localhost:4502/crx/packmgr/index.jsp -** IDE for creating and installing packages to import and export repository nodes (can be code or content, depending on the filters defined in the package definition)
* **Apache Felix Console - http://localhost:4502/system/console/bundles -** OSGi Bundles (JAR Files for different modules of AEM Internal Functionality & Custom Project) are installed and managed here.
* **Apache Felix Config Manager - http://localhost:4502/system/console/configMgr -** Configurable OSGi Services managed here.
* **Site Admin - http://localhost:4502/siteadmin -** (Main WCM Page for the Author to manage pages & websites)
* **DAM Admin - http://localhost:4502/damadmin -** Digital Asset Manager (Manage all assets from here, along with renditions & metadata for the Asset - restrictions will be handled as per User Rights across the Content Page & Respective DAM Folder)
* **User Admin - http://localhost:4502/useradmin -** Manage user groups & permissions
* **Misc Admin - http://localhost:4502/miscadmin -** Extra tools can be accessed from here
* **Campaign Manager - http://localhost:4502/mcmadmin#/content/dashboard** (Personalization & Segmentation & Campaigns are managed from here)
* **Tagging - http://localhost:4502/tagging -** Create & Manage Tags for the websites
* **Workflow Console - http://localhost:4502/libs/cq/workflow/content/console.html -** Workflow Console

1. **What is the use of EditConfig node in creating a component?**

  A cq: EditConfig node is used to define the behavior of the component

1. **What are the basic SCR Annotations used for creating an OSGI component?**

  Basic SCR Annotation used for developing a component or service in OSGi are:-

**@Component –** defines the class as a component.

**@ Service –** defines the service interface that is provided by the component.

**@Reference –** injects a service into the component.

**@ Property –** defines a property that can be used in the class.

1. **Difference between Dialog and Design Dialog?**

Both dialog and design dialog are used by the user to configure the component.

**Design Dialog:** of a component can be seen/edited in design mode of the page. Design dialog is present at template level so all the page of the template will share same design dialog.

**Dialog:** of a component is present at page level so each component instance will have its own dialog and information entered in dialog will be stored under content folder.

1. **Where dialog and design dialogue data is stored?**

Design dialog data is stored under **/etc/designs** folder

V/S

Dialog data is stored under **/content** folder

1. **What is Adaptive Form? Explain Adaptive Form? What do you mean by Adaptive Form?**

  Adaptive forms, is used to break down a form into logical sections, basically it enables end users to focus on filling out the form. When we require to take input from user we use Adaptive Form.

1. **What is Adaptive Document? Explain Adaptive Document? What do you mean by Adaptive Document?**

Adaptive Document is used to display output to the end user. For example- a bank statement is an adaptive document as all its content remain same only name and amount changes. Basically we put place holder text in Adaptive Document which are filled dynamically at run time.

1. **What do you mean by Site Page?**

 A site is basically a website, where we can place an Adaptive Form, Adaptive Document or a static text.

1. **How to put multiple files in CRX repository?**

 To put multiple files we can use many tools that supports **WebDAV Protocol** like Net Drive.

1. **How do you resolve a resource?**

You can resolve using Resource Resolver which you can get from method getResourceResolver() and then use resolve() method to resolve a resource.

1. **How do you adapt a resource?**

You can adapt a resource to any another type using adaptTo() method, which accepts class type in which you want to adapt your resource. i.e. Page page = resource.adaptTo(Page.class);

1. **What is the purpose of clientlibs?**

 It is used for adding site specific js and css files to the page and also third-party js and css files. jcr:primaryType of clientlibs folder is cq:ClientLibraryFolder. It takes cares of dependency management, merging files and  minification of all js and css files stored under it.

1. **How to connect to external Database in CQ?**

To connect to external DB, we need to configure a connection pool by creating a node of type sling:OsgiConfig. Please have a look at below screenshot for connecting to hsqldatabase, but if you want to connect to other DB then properties values needs to be changed accordingly.

1. **Explain the role of Dispatcher in**[**AEM**](http://www.gloryittechnologies.com/Other-Technologies/Adobe-CQ5-Online-Training.html)**?**

Unlike most CMS systems, AEM comes with its own caching system, the [Dispatcher](http://dev.day.com/docs/en/cq/current/deploying/dispatcher.html). The dispatcher is an Apache Httpd module which proxies and caches static HTML and assets served by CQ Publishers. This significantly increases the number of requests which can be served and decreases the load on the publisher.

The major impact of this is components must be created to be cachable manner and should not require server side logic to be executed with every request. If that is not the case, a single component can require the entire page to not be cached and the entire page be rebuilt for every request. Of course, most projects will require components which require server-side logic and developers will need to use tools such as AJAX and SSI to create sites which have server side logic but still enable caching.

To summarize, Dispatcher has two main roles:

* Caching – It is used to cache as much content as possible in the form of a static website which helps to reduce the continuous functioning of layout engine frequently for generating content when website data is dynamic.
* Load-balancing – To increase the performance by load-balancing.

Note:  If there are multiple cq instances configured with a dispatcher, the dispatcher can do a load- balancing and if there is too much load on any cq instance, it can relay the request to another less busy instance.

1. **What is dispatcher.any file?**

By default the Dispatcher configuration is stored in the dispatcher.any text file, though you can change the name and location of this file during installation.

The configuration file contains a series of single-valued or multi-valued properties that control the behavior of Dispatcher:

* Property names are prefixed with a forward slash ("/").
* Multi-valued properties enclose child items using braces ("{ }").

1. **Which requests are not cached by the dispatcher?**

By default the following requests are not cached by dispatcher

* Request that do not return http code 200
* Requests with suffixes
* Requests with request parameter(i.e ?)
* Programatically: send http header   
  response.setHeader("Dispatcher", "no-cache")

1. **How do I delete files from the Dispatcher Cache?**

We can delete files from the CQ cache by using an HTTP request. When the HTTP request is received, Dispatcher deletes the files from the cache. Dispatcher caches the files again only when it receives a client request for the page. Deleting cached files in this manner is appropriate for web sites that are not likely to receive simultaneous requests for the same page.

The HTTP request has the following syntax:

*POST /dispatcher/invalidate.cache HTTP/1.1  
 CQ-Action: Activate  
 CQ-Handle: path-pattern  
 Content-Length: 0*

Dispatcher deletes the cached files and folders that have names that match the value of the CQ- Handle header. For example, a CQ-Handle of /content/geomtrixx-outdoors/en matches the following items:

* All files (of any file extension) named en in the geometrixx-outdoors directory
* Dispatcher only flushes files or directories that match en.\* in the geometrixx-outdoors directory. The directory en itself will only be deleted if the "CQ-Action" is "Delete" or "Deactivate".

1. **Where does the cache directory exists for AEM?**

  The Dispatcher uses a Cache Directory for caching static content. The cached documents are created in the root of a web-server.

1. **Explain the methods of Caching adopted by Dispatcher?**

  Dispatcher uses the following methods for caching:

* Content Updates invalidates those pages whose content has been updated and replaces it with new content.
* Auto-invalidation automatically invalidates the content parts which are out of date – without physically deleting any files.

1. **How Dispatcher performs Load-balancing ?**

* Performance Statistics – Dispatcher keeps statistics on how fast each instance of cq is responding to a particular url. Based on those metrics, dispatcher determines which instance of cq will fetch the quickest response for any request and relays the request to that cq instance.
* Sticky Connections – when a user session is established, then all incoming requests from that user should be served by the same cq instance, because other cq instances cannot recognize the user session and generate personalized pages for him. Dispatcher makes sure all requests for user session are served from the same cq instance.
* Increased fail-safe coverage: If the Dispatcher does not receive responses from an instance, it will automatically relay requests to one of the other instance(s)
* Increased processing power: In practice this means that the Dispatcher shares document requests between several instances of CQ. Because each instance has fewer documents to process, you have faster response times.

1. **Can I implement multiple Dispatchers in a setup?**

 Yes. In such cases, ensure that both the Dispatchers can access the CQ website directly. A Dispatcher cannot handle requests coming from another Dispatcher.

1. **What are the differences between package and bundle?**

  Package: A Package is a zip file that contains the content in the form of a file-system serialization (called “vault” serialization) that displays the content from the repository as an easy-to-use-and-edit representation of files and folders. Packages can include content and project-related data.

Bundle: Bundle is a tightly coupled, dynamically loadable collection of classes, jars, and configuration files that explicitly declare their external dependencies (if any).

1. **Explain life cycle of OSGI [Open Systems Gateway initiative] bundle?**

OSGi is a framework which allows modular development of applications using java. A large application can be constructed using small reusable components(called bundles in terms of OSGi) each of which can be independently started, stopped, and also can be configured dynamically while running without requiring a restart.

1. **What are the advantages of using OSGI?**

Advantages of using OSGI are stated below:-

* Dynamic module system for Java.
* Universal Middleware Category.
* Helps applications to be constructed from small, reusable and collaborative components.
* OSGi bundles can contain compiled Java code, scripts, or any contents to be loaded in the repository.
* Helps the bundles to be loaded, installed. Reduces the complexity of the system.

1. **What are the difference between OSGi bundle and Normal Jar file?**

* OSGi bundles are jar files with metadata inside. Much of this metadata is in the jar’s manifest, found at META-INF/MANIFEST.MF. This metadata, when read by an OSGi runtime container, is what gives the bundle its power.
* With OSGi, just because a class is public doesn’t mean you can get to it. All bundles include an export list of package names, and if a package isn’t in the export list, it doesn’t exist to the outside world. This allows developers to build an extensive internal class hierarchy and minimize the surface area of the bundle’s API without abusing the notion of package-private visibility. A common pattern, for instance, is to put interfaces in one package and implementations in another, and only export the interface package.
* All OSGi bundles are given a version number, so it’s possible for an application to simultaneously access different versions of the same bundle (eg: junit 3.8.1 and junit 4.0.). Since each bundle has it’s own class-loader, both bundles classes can coexist in the same JVM.
* OSGi bundles declare which other bundles they depend upon. This allows them to ensure that any dependencies are met before the bundle is resolved. Only resolved bundles can be activated. Because bundles have versions, versioning can be included in the dependency specification, so one bundle can depend on version junit version 3.8.1 and another bundle depend on junit version 4.0. In OSGi bundle, there will be an Activator.java class in OSGi which is an optional listener class to be notified of bundle start and stop events.

1. **What is the difference between sling:resourceType & sling:superResourceType?**

**sling:resourceSuperType**: It is used to achieve inheritance in cq. When set, it inherits the specified component to this component.

**sling:resourceType**: It is a path, which locates the script to be used for rendering the content. Path used can be absolute or relative.

1. **What are the differences between parsys and iparsys?**

* **parsys –** It is a placeholder called “Paragraph System”, where we can drag and drop or add other components or scripts at page level.
* **iparsys** – The inherited paragraph system is a paragraph system that also allows you to inherit the created paragraphs from the parent. it is similar to parsys except that it allows to inherits parent page “paragraph system” at template level. You can also cancel paragraph inheritance at a level at any time. It has two checkbox options to cancel/disable the inheritance.

**Cancel Inheritance** - If selected, the components in this paragraph system are not passed down to the child pages.

**Disable Inheritance** - If selected, components of this paragraph system on this page are not inherited from the parent page.

#### What is the difference between sling:include, cq:include & cq:import?

**sling:include** - This is the include tag of the Sling JSP Tag library. This tag knows about Sling and also supportsRequestDispatcherOptions.

**cq:include** - This tag is Communiqué specific extension of the Sling JSP Tag library include tag. IIRC it supports callings scripts in addition to just including renderings of resources.

**c:import** - I assume this is the import tag of the Standard Tag Library. This tag is documented at http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/c/import.html and does not know about Sling directly.   
 But -- asuming -- this tag is using a RequestDispatcher to dispatch the request, this tag will also pass Sling and the Sling resource resolver.

#### How to preinclude a component into a template?

**In Sightly** -   
 < div data-sly-resource="${@path='mycomponent', resourceType='foundation/components/mycomponent'}"> 

**In JSP** -   
 < cq:include path="mycomponent" resourceType="foundation/components/mycomponent" />

#### Can you tell something about Widgets & Xtypes in AEM?

**widget** - Adobe Experience Manager (AEM) uses the ExtJS widgets library, which provides the highly polished user interface elements that work across all the most important browsers and allow the creation of desktop-grade UI experiences. These widgets are included within AEM and, in addition to being used by AEM itself, can be used by any website built using AEM.

**xtype** - In the ExtJS language, an xtype is a symbolic name given to a class. For list of Xtypes available in AEM, go through: https://docs.adobe.com/docs/en/aem/6-3/develop/components/xtypes.html

#### What is the difference between allowedPaths, allowedChildren & allowedParents?

**allowedPaths** - Path of a page that is allowed to be based on this template.

**allowedChildren** - Path of a template that is allowed to be a child of this template.

**allowedParents** - Path of a template that is allowed to be a parent of this template.

#### How can we reuse a part of dialog in another dialog?

A specific widget with the label cqinclude exists which allows for inclusion of existing dialog elements in other dialog definitions. The generic JSON format is used on the client side to construct the actual dialog. Following is an example which uses the cqinclude widget to include an existing tab from an existing dialog:

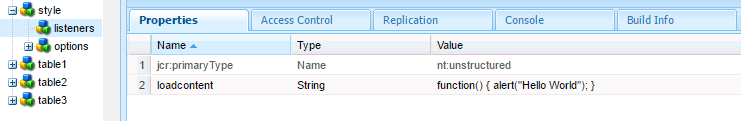
{  
 "jcr:primaryType": "cq:Widget",  
 "xtype": "cqinclude",  
 "path": "/libs/replication/components/agent/tab\_extended.infinity.json"  
 }

The path property needs to point to a dialog-resource that is to be included in JSON format.

#### What is a Listener Node?

We can have listeners node under dialog/widget in AEM component. Listeners is nt:unstructured type node. We can add event(like loadcontent, focus etc) as property into listeners node.

Example :



Listeners will trigger loadcontent event and show "Hello World" in popup.

#### Can you explain the concept of ClientLibs in AEM?

Clientlibs or Client libraries in aem is one of the most widely used features provided by Adobe, it allows us to not only manage our client side resources like ( JavaScript, CSS, images, fonts etc ), but also provide options to debug, minify,merge and gzip the client-side code.

The centralized approach entails bundling all client libraries into monolithic and all-encompassing.JS and .CSS files within /etc/designs/{project}/clientlibs. This type of clientlib is loaded by default for every page within a project and it doesn’t need to be called explicitly. The biggest obstacle to this is large files that load all the JS and CSS regardless of whether or not the page actually needs it, at the expense of page load performance.

#### How to create a Client Folder and call the same in a Template or Component?

* Open CRXDE Lite in a web browser (http://:/crx/de).
* Select the folder where you want to locate the client library folder and click Create > Create Node.
* Enter a name for the library file, and in the Type list select cq:ClientLibraryFolder. Click OK and then click Save All.
* To specify the category or categories that the library belongs to, select the cq:ClientLibraryFolder node, add the following property, and then click Save All:
  + Name: categories
  + Type: String
  + Value: The category name
  + Multi: Select
* Add source files to the library folder by any means.
* Select the client library folder and click Create > Create file.
* In the file name box, type one of the following file names and click OK:
  + **js.txt:** Use this file name to generate a JavaScript file.
  + **css.txt:** Use this file name to generate a Cascading Style Sheet.
* Open the file and type the following text to identify the root of the path of the source files:   
  #base=[root]
* On the lines below #base=[root], type the paths of the source files relative to the root. Place each file name on a separate line.
* Click Save All.

#### What is the difference between categories, dependencies & embed w.r.t. ClientLibs?

**categories** – This is the identifier into which categories a clientlib belongs. A clientlib can have one or more categories.

**dependencies** - This defines the other categories that the current clientlib depends upon. The dependencies will be included in the page along with the dependent clientlib.   
 This property is transitive – if Clientlib A depends on Clientlib B which depends on Clientlib C, then all clientlibs will be included in the page.

**embed** - This defines the categories which will be combined to the current clientlib. AEM will merge all clientlibs into the current clientlib. This is usually used for minimizing requests and for accessing clientlibs which are not supposed to be exposed to public.   
 Take note that the embed property is NOT transitive – If Clientlib A embeds Clientlib B which embeds Clientlib C, then only Clientlib A and B will be included in the page. Clientlib A and B will be combined into one CSS and JS files as well. In order to include Clientlib C, it must be added to the embed property of Clientlib A as well.

#### What are the different ways to include a ClientLib in a Component or Template?

**In JSP:**  
      <cq:includeClientLib categories="yourcategoies"/>

**In Sightly**:

The best example is from the new foundation page component, this is located at        
 **/libs/wcm/foundation/components/page/head.html.**

There, you see the following thing:        
      **data-sly-use.clientlib="/libs/granite/sightly/templates/clientlib.html"**

This declares a “clientlib” object, which is implemented as a template.        
 After that, you see statements like following one:       
      **data-sly-call="${clientlib.all @ categories='cq.jquery'}"**   
 This will output <script> and a <style> includes to all CSS and JS contained in the parameter “categories”.

You can also call “clientlib.css” and “clientlib.js” if you only want to output the CSS or JS:        
            **data-sly-call="${clientlib.js @ categories='clientlib1,clientlib2'}"**  
            **data-sly-call="${clientlib.css @ categories='clientlib1,clientlib2'}"**  
 Notice that the “categories” option can be a comma-separated string, or a list of category names.

#### What are Workflows?

Workflows enable you to automate Experience Manager activities. Workflows consist of a series of steps that are executed in a specific order. Each step performs a distinct activity such as activating a page or sending an email message. Workflows can interact with assets in the repository, user accounts, and Experience Manager services. Therefore, workflows can coordinate complicated activities that involve any aspect of Experience Manager

Many useful workflow models are provided with Experience Manager. In addition, any number of custom workflow models, tailored to the specific needs of your project, can be defined using the Workflow console.

#### What all you can do in the AEM Workflow Console?

The Workflow console is the centralized location for workflow management in AEM. It can be accessed via the Workflows link on the AEM Welcome page (Classic UI), and the Tools section of the touch-optimized UI.

#### Within the Workflow console, there are several tabs:

* **Models -**Lists the workflow models currently available. Here you can create, edit or delete workflow models.
* **Instances -**Shows you details of workflow instances which are currently active. These instances are also version dependent.
* **Archive -**Enables you to access details of workflow instances which have terminated, for whatever reason.
* **Launcher -**Allows you to define a workflow to be launched if a specific node has been updated.
* **Failures -**Enables you to monitor and manage failed worklow instances.

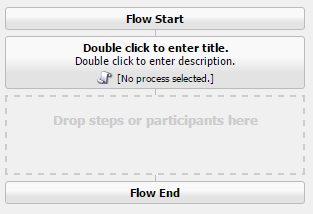
#### How to create a Custom Process Step in AEM Workflow?

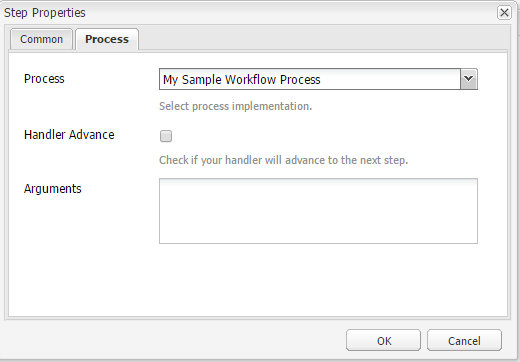
To define a process step as an OSGI service component.

* The OSGI component needs to implement the WorkflowProcess interface with its execute() method.
* Add the SCR property "process.label" and set the value as you please. This will be the name which your process step is listed as when using the generic Process Step component.

import org.apache.felix.scr.annotations.Component;  
import org.apache.felix.scr.annotations.Properties;  
import org.apache.felix.scr.annotations.Property;  
import org.apache.felix.scr.annotations.Service;  
import org.osgi.framework.Constants;  
import com.adobe.granite.workflow.WorkflowException;  
import com.adobe.granite.workflow.WorkflowSession;  
import com.adobe.granite.workflow.exec.WorkItem;  
import com.adobe.granite.workflow.exec.WorkflowProcess;  
import com.adobe.granite.workflow.metadata.MetaDataMap;  
@Component   
@Service   
@Properties({  
@Property(name = Constants.SERVICE\_DESCRIPTION, value = "An example workflow process implementation."),     
@Property(name = Constants.SERVICE\_VENDOR, value = "my vendor"),     
@Property(name = "process.label", value = "My Sample Workflow Process") })     
 public class MyProcess implements WorkflowProcess {    
public void execute(WorkItem arg0, WorkflowSession arg1, MetaDataMap arg2 )throws WorkflowException {    
// TODO Auto-generated method stub    
}    
}

* **execute method** has three parameters :-
* **WorkItem -**A WorkItem is the unit that is passed through an Workflow instance of a WorkflowModel. It contains the WorkflowData the instances acts on and a reference to the WorkflowNode that describes the underlying workflow step. A WorkItem can be considered as a token in a petri net that flows through the net during execution.
* **WorkflowSession -**The WorkflowSession class provides all functionality (depending on the users rights) for managing WorkflowModels, Workflow instances and their execution.
* **MetaDataMap -**A value map for generic access to meta data values.
* In the CQ Workflow console, add the process step to the workflow using the generic Process Step component.



* In the edit dialog, go to the Process tab and select your process implementation.(Here My Sample Workflow Process).  
     
* If you use arguments in your code set the Process Arguments
* Save the changes

#### What is a Dynamic Participant Step?

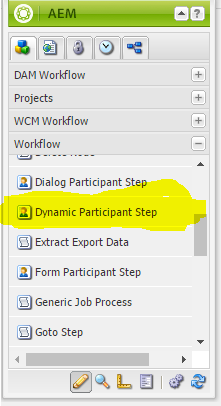
The Dynamic Participant Step component is similar to Participant Step except the participant is selected automatically at run time.

To define a dynamic participate step as an OSGI service component.

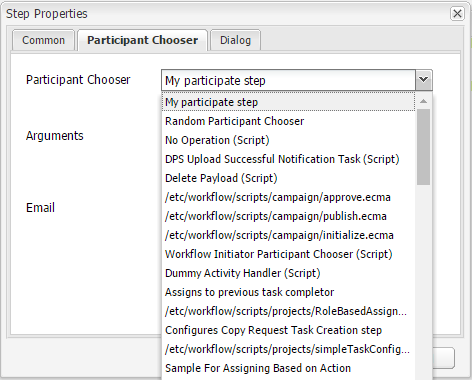
* **T**he OSGI component needs to implement the ParticipantStepChooser interface with its getParticipant() method.
* Add the SCR property "chooser.label" and set the value as you please. This will be the name which your Participant chooser is listed as when using the Dynamic Participant Step component.

import org.apache.felix.scr.annotations.Component;  
import org.apache.felix.scr.annotations.Properties;  
import org.apache.felix.scr.annotations.Property;  
import org.apache.felix.scr.annotations.Service;  
import org.osgi.framework.Constants;   
import com.day.cq.workflow.WorkflowException;  
import com.day.cq.workflow.WorkflowSession;  
import com.day.cq.workflow.exec.ParticipantStepChooser;  
import com.day.cq.workflow.exec.WorkItem;  
import com.day.cq.workflow.metadata.MetaDataMap;  
@Component  
@Service  
@Properties({  
 @Property(name = Constants.SERVICE\_DESCRIPTION, value = "An example of a dynamic participant chooser."),     
 @Property(name = ParticipantStepChooser.SERVICE\_PROPERTY\_LABEL, value = "My participate step") })     
 public class InitiatorParticipantChooser implements ParticipantStepChooser {   
public String getParticipant(WorkItem arg0, WorkflowSession arg1,MetaDataMap arg2) throws WorkflowException {       
 return "admin";        
}   
}

* In the CQ Workflow console, add the participant step to the workflow using the dynamic particpant Step component.



* In the edit dialog, go to the Participant chooser tab and select your Participant chooser implementation.



* If you use arguments in your code set the Arguments
* Save the changes

#### What are the different built in users/groups present in AEM?

We have two built-in workflow groups.

* **workflow-editors**- Group that is allowed to create and modify workflow models.
* **workflow-users**- A user participating in a workflow must be member of group workflow-users. This gives him or her full access to: /etc/workflow/instances so that he or she can update the workflow instance.

#### How to trigger a workflow programmatically?

Procedure to trigger a workflow programmatically.

* Create a workflow session

WorkflowSession wfSession = workflowService.getWorkflowSession(session);

* Get the workflow model

WorkflowModel wfModel = wfSession.getModel(model); # your workfllow model name

* Get the workflow data.

WorkflowData wfData = wfSession.newWorkflowData("JCR\_PATH", path);

* Run the Workflow.

wfSession.startWorkflow(wfModel, wfData);

#### [What is new in AEM 6.2?](http://aeminterviewquestions.com/#collapse3)

Adobe Experience Manager 6.2 is an upgrade release to the Adobe Experience Manager 6.1 code base. It provides new and enhanced functionality, key customer fixes, high priority customer enhancements and general bug fixes oriented toward product stabilization. It also includes all Adobe Experience Manager 6.1 feature pack, hot fix, and service pack releases.

**Security Features:**

* Added support for password history.
* Configurable authentication token expiration
* Ongoing effort: Switched usage of the Sling loginAdministrative API to Service Users it various areas of the product.

**Main repository enhancements are**:

* Support for MongoDB Enterprise 3.2
* Cold standby enhancements to support a procedural failover for high availability in TarMK.
* Oak search enhancements like Faceted Search, Suggestions, Spellchecker and more.
* Performance, Scalability and Resilience in general.
* Revision Cleanup Support(Offline revision cleanup is the recommended way of performing revision cleanup)

**User Interface:**

AEM 6.2 implements the 2016 Adobe Marketing Cloud UI design (also known as Shell 3). Further - the user interface is in transition from Coral UI 2 to the Web Component based Coral 3 UI library.

**Operations Dashboard**:

* "Explain Query" provides insight to the mechanics of your queries to support diagnosis and optimization.
* Specific repository aspects can be monitored on a configurable timeline view in the Tools/Operations section.
* A configurable series of Java thread dumps can be downloaded now with the Status.zip from the Tools/Operations/Diagnosis section.
* User Sync Diagnostics to support consistency for User/Groups across AEM instances.

**Content Distribution**:

* Package replication to support extra large activation volumes
* Priority-Queuing configuration to allow split between urgent activations and backlog.
* A configurable series of Java thread dumps can be downloaded now with the Status.zip from the Tools/Operations/Diagnosis section.
* Auto-unlocking stuck replication queues incl advanced notifications

1. **What is Scaffolding?**

With scaffolding we can create a form (a scaffold) with fields that reflect the structure we want for our pages and then use this form to easily create pages based on this structure.

1. **What are the Use-API in Sightly?**

**Java Use-API** - The HTML Template Language (HTL) Java Use-API enables a HTL file to access helper methods in a custom Java class. This allows all complex business logic to be encapsulated in the Java code, while the HTL code deals only with direct markup production.

**JavaScript Use-API** - The HTML Template Langugae (HTL) JavaScript Use-API enables a HTL file to access helper code written in JavaScript. This allows all complex business logic to be encapsulated in the JavaScript code, while the HTL code deals only with direct markup production.

1. **How to call a Service in Java Use-Api Class?**

Use SlingScriptHelper.getService() method (into Java Use-API class)  
SampleService service=getSlingScriptHandler().getService(SampleService.class);

1. **What is the use of unwrap, element and attribute in sightly?**

* **Attribute attribute** - Use an array to make things easy. Example   
  attrMap = {        
  title: "myTitle",       
  class: "myClass",       
  id: "myId"        
  }        
        <div data-sly-attribute="${attrMap}"></div>       <div title="myTitle" class="myClass" id="myId"></div>
* **Element attribute** - Replaces the element of the host element
* **Unwrap attribute** - Remove the outer elements (i.e. HTML Tags) leaving just the output of the Expression)

1. **What is the difference between Sightly & JSP?**

Sightly Offers below advantages over JSP for better development in AEM

* Protection against cross-side scripting injection.
* Easily development of AEM Projects by front-end developers.
* Flexible and powerful templating and logic binding features
* Need to write less code in Sightly thus productivity increases.
* Wider range of implicit objects as compared to JSP.

1. **What are the advantages of Sightly/HTL?**

The HTML Template Language has been introduced with AEM 6.0, and takes the place of JSP (JavaServer Pages) as the preferred and recommended server-side template system for HTML. Folowing are the benefits of HTML Template Language

* **Increased Security** - The HTML Template Language increases the security of sites that use it in their implementation, as compared to JSP and to most other template systems, because HTL is capable of automatically applying the proper context-aware escaping to all variables being output to the presentation layer. HTL makes this possible because it understands the HTML syntax, and uses that knowledge to adjust the required escaping for expressions, based on their position in the markup. This will for instance result in expressions placed in href or src attributes to be escaped differently from expressions placed in other attributes, or elsewhere.

While the same result can be achieved with template languages like JSP, there the developer must manually ensure that the proper escaping is applied to each variable. As a single omission or mistake on the applied escaping is potentially sufficient to cause a cross-site scripting (XSS) vulnerability, we decided to automate this task with HTL.

* **Simplified Development** - The HTML Template Language is easy to learn and its features are purposely limited to ensure that it stays simple and straight-forward.
* **Reduced Costs** - Increased security, simplified development and improved team collaboration, translates for AEM projects in reduced effort, faster time to market (TTM), and lower total cost of ownership (TCO).

1. **What is a Sling Servlet?**

Servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers.

Servlets can be registered as OSGi services. For a Servlet registered as an OSGi service to be used by the Sling Servlet Resolver, the following restrictions apply:

* Either the sling.servlet.paths or the sling.servlet.resourceTypes service reference property must be set. If neither is set, the Servlet service is ignored.
* If the sling.servlet.paths property is set, all other sling.servlet.\* properties are ignored.
* Otherwise a Resource provider is registered for the Servlet for each permutation resource types, selectors, extensions and methods.

1. **How to register Servlets using Java Annotations?**

There are two ways of doing this

**1. The @SlingServlet annotation**

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

**2. The @Properties and @Property annotations**

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

We should prefer @SlingServlet annotation.

1. **What are the different ways to register a Servlet in AEM?**

You can register a Servlet using the two Standard approaches:

**1. Registering the servlet by path**

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

**2. Register servlet by ResourceType**

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

The resource type(s) supported by the servlet. The property value must either be a single String, an array of Strings or a Vector of Strings. This property is ignored if the sling.servlet.paths property is set.

1. **What is the difference between SlingSafeMethodsServlet & SlingAllMethodsServlet?**

**SlingSafeMethodsServlet**- Helper base class for read-only Servlets used in Sling. This base class is actually just a better implementation of the Servlet API HttpServlet class which accounts for extensibility. So extensions of this class have great control over what methods to overwrite. It supports GET, HEAD, OPTIONS etc methods.

**SlingAllMethodsServlet**- Helper base class for data modifying Servlets used in Sling. This class extends the SlingSafeMethodsServlet by support for the POST, PUT and DELETE methods.

1. **What are OSGi Components & Services in AEM?**

Components are main building block for OSGI application. Components in OSGI are, by defination, provided by a bundle. A bundle will provide/contain one or more components. A component is like a run-time service. They can publish themselves as a service, and/or they can have depencencies on other components/services. Adding a @Component annotation to a public class will turn it into a component

An OSGi service is a java object instance, registered into an OSGi framework with a set of properties. Any java object can be registered as a service, but typically it implements a well-known interface. The client of a service is always an OSGi bundle, i.e. a piece of java code possible to start via the BundleActivator interface. Each bundle may register zero or more services. Each bundle may also use zero or more services. There exists no limit on the number of services, more than the ones given by memory limits or java security permissions.

Following are the advantages of OSGI services:

* Lightweight services
* Separates interface from implementation
* Lookup is based on Interface name
* Direct method invocation
* Good design practice
* Enables reuse, substitutability, loose coupling and late binding.

1. **How to create a Service Class?**

We can define a class as a service by adding the following scr annotations:

* **@Component** – defines the class as a component
* **@Service** - defines the service interface that is provided by the component

Example

import org.apache.felix.scr.annotations.Component;  
 import org.apache.felix.scr.annotations.Service;  
 //This is a component so it can provide or consume services  
 @Component   
 @Service   
 public class MyserviceImpl implements MyService {   
 }

1. **Brief explanation on different OSGI Annotations?**

**@component** - The @Component annotates an implementation class and is used to declare it as a component type. It is the only required annotation. If this annotation is not declared for a Java class, the class is not declared as a component.

**@service** - The @Service annotation defines whether and which service interfaces are provided by the component. This is a class annotation.

**@reference** - The @Reference annotation defines references to other services made available to the component by the Service Component Runtime.

**@property** - The @Property annotation defines properties which are made available to the component through the ComponentContext.getProperties() method. These tags are not strictly required but may be used by components to defined initial configuration. Additionally properties may be set here to identify the component if it is registered as a service, for example the service.description and service.vendor properties.

1. **What is the difference between OSGI Component & Service?**

**Component** - If you want the life of your object to be managed by the OSGi container, you should declare it as a component. Using 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. All objects managed by OSGi container are components. You qualify components as services. This means that all services are components but not vice-versa.

**Service** - OSGi components can be made as OSGi service by marking it with @Service annotation. When you mark a component as service, you could refer (call) this service from other osgi components. 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.

1. **What are the different ways to call a Service Class?**

* Use SCR annotations to let SCR inject the service in your component:(Generally in Servlet)  
       **@Reference**   
   private MyService myService;
* Use Bundle Context to get the service in your Java/Jsp file  
   BundleContext bundleContext = FrameworkUtil.getBundle(MyService.class).getBundleContext();       
   ServiceReference factoryRef = bundleContext.getServiceReference(ResourceResolverFactory.class.getName());       
   ResourceResolverFactory resolverFactory = (ResourceResolverFactory) bundleContext.getService(factoryRef);
* Use **sling.getService()** method(Generally in JSP)  
   MyService service=sling.getService(MyService.class);
* Use **SlingScriptHelper.getService()** method (into Java Use-API class ie Handler class )  
   MyService service=getSlingScriptHandler().getService(MyService.class);

1. **What is a Service Factory?**

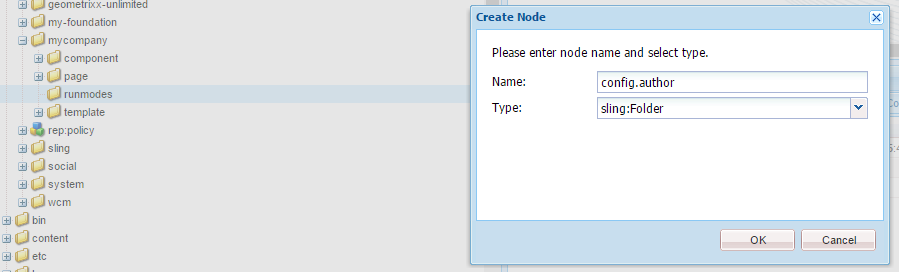
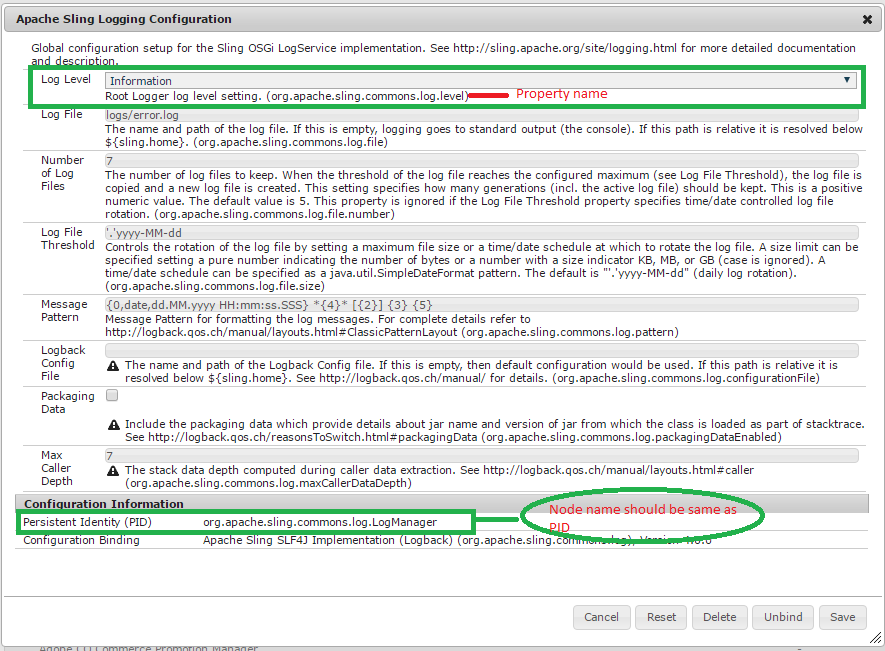
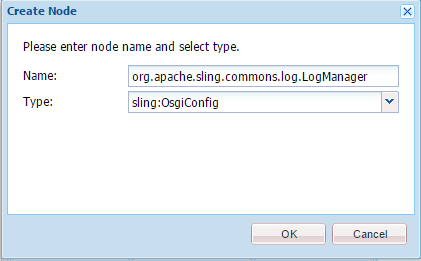
An OSGi service factory is a special class ServiceFactory, which can create individual instances of service objects for different bundles. Sometimes a service needs to be differently configured depending on which bundle uses the service. For example, the log service needs to be able to print the logging bundle’s id, otherwise the log would be hard to read.

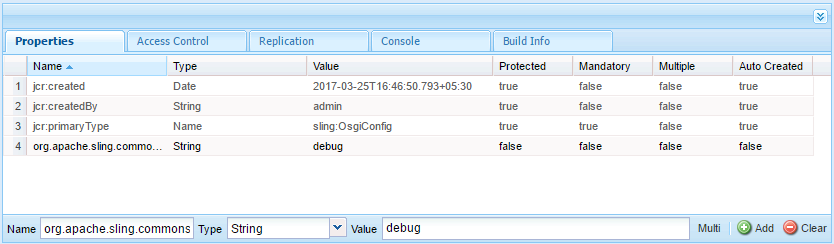
1. **What are Run Modes? How to set up Run Modes in AEM?**

Run modes allow you to tune your AEM instance for a specific purpose; for example author or publish, Devlopment, QA, Production etc. We can divide into two types

* **Installation Run Modes** :- Installation Run Modes cannot be changed. These run modes are used at installation time and fixed for the entire lifetime of the instance. Example author, publish .
* **Customized Run Modes** :- Customized run modes are applied on each startup and can be changed with a restart of AEM instance. These are custom run modes defined for specific environment like development (config.author.dev), or production(config.publish.prod).

Set up a different log level in local.

* Go to CRXDE lite.
* Go to /apps/(mycompany)
* Create runmodes folder.
* Create config.author folder  
    
  
* Create a node under the config.author folder   
  Node name: Persistent Identity(PID) of the configuration.  
  Type: sling:OsgiConfig.  
    
    
    
  
* Add property org.apache.sling.commons.log.level on org.apache.sling.commons.log.LogManager for changing log level specific to environment.



* click on save

1. **Explain about Multi Site Manager.**

Multi Site Manager (MSM) enables us to easily manage multiple web sites that share common content. MSM lets you define relations between the sites so that content changes in one site are automatically replicated in other sites.

MSM reduces the time it takes to manage us websites and increases the re-use of common content:

* Efficiently manage different language versions of a website.
* Automatically update one or more sites based on a source site:
* Enforce a common base structure and use common content across multiple sites.
* Maximize the use of available resources.
* Focus efforts on managing the content that differs between the sites.
* Maintain a common look and feel.

Multi Site Manager (MSM) allows us to create copy of existing site and automatically update the copy when changes are done to the source site. Multi Site Manager (MSM) enables us to easily manage multiple web sites that share common content. We should use MSM when our website is :

Multinational Site - Sites are in multiple countries

Multilingual Site - Sites are in multiple languages.

Multinational Multilingual Site - Sites are in multiple countries and in multiple languages

1. **What is a Blueprint? How to create a Blueprint?**

Blueprints target the rollout of complete multilingual website projects and are a tool to control multiple rollout configs and live copies. Blueprints allow you to control multiple live copies and centrally consistent rollout configs for the blueprint's live copies. A blueprint rollout will push modifications to all it's live copies.The default Blueprint Template assumes that the source web site has the following characteristics:

* The web site has a root page.
* The immediate child pages of the root are language branches of the web site. The Name of each page is a language code. When creating a Live Copy, the languages are presented as optional content to include in the copy.
* Each language page contains one or more child pages. When creating a Live Copy, child pages are presented as a chapter that you can include in the copy.

**To create a blueprint:**

* Open the Tools console.
* Select Tools, then MSM Control Center. Click New... in the top-middle toolbar.
* In the Create Page dialog, define the blueprint:
  + **Title**, the page title,
  + **Name**, the page (node) name
  + Select the Blueprint Template

click **create**

* Open the newly created page and click the **Edit** button beside **Settings**.
* In the Blueprint Settings dialog, define the blueprint:
  + **Name**, the blueprint name that was defined earlier can be changed.
  + **Description**, e.g. This is my blueprint
  + **Source Path:** set the path of the blueprint, e.g. /content/geometrixx
  + **Thumbnail Image** (optional): this thumbnail will appear in the live copy creation.

1. **What is a Live Copy? How to create a Live Copy?**

Live Copy is the creation of a new site based on the content and structure of an existing site. Changes or additions to the main site can then cascade down throughout Live Copies. When used to its full potential, this is a very powerful tool that can simplify management across a large number of sites with little effort.

Create a Live Copy of any page and its child pages, or of a single page. When you create the Live Copy, you can optionally specify the rollout configurations to use for automatically updating the content:

* The selected rollout configurations apply to all of the Live Copy pages that are created for the source page and its child pages.
* If you specify no rollout configurations, either the system default rollout configuration is used, or the default rollout configuration for the branch is used.

The following procedure creates a **Live Copy** using the classic UI.

* Open the Websites console.
* Select the folder or page below which you want to locate the Live Copy pages.
* Click New > New Live Copy.
* In the Source selection tab, define the Live Copy:
  + **Title**: The title of the root Live Copy page
  + **Name**: The name of the page node
  + **Live Copy From**: Browse to select the page to use as the source of the Live Copy
  + **Exclude sub pages**: Select this option to exclude the child pages from the Live Copy.
* On the Sync config tab, specify one or more Rollout Configs to use for the Live Copy
* Click **Create**.

1. **What are Rollout Configurations? How to create custom Rollout Configurations?**

If we run Live Copy straight out of the box, we get an exact duplicate of our original site. That can be good, but probably doesn’t fulfill a project’s requirements. Chances are we’d need to change the template and resource types of our Live Copy pages. These changes can be made using rollout configurations in Live Copy. With a rollout configuration, CQ can automatically create a new mobile page anytime we create a new page in our main site, taking the exact content from the new main page while still using the mobile site’s designs. We could always do this manually, but configuring our Live Copy rollouts will automate the process—and it really isn’t too difficult. **Rollout configurations** can be found by navigating to **/etc/msm/rolloutconfigs**.

There are some rollout configurations are installed by default in AEM . e.g Standard rollout config, Activate on Blueprint activation, Deactivate on Blueprint deactivation, Push on modify [more](https://docs.adobe.com/docs/en/cq/5-6-1/administering/multi_site_manager.html#Installed Rollout Configurations)

We need to create a rollout configuration when the installed rollout configurations do not meet your application requirements.   
 Create the rollout configuration, then add the trigger and synchronazion actions. The new rollout configuration is available when setting rollout configurations on a blueprint or on a Live Copy page

**CREATE ROLLOUT CONFIGURATIONS**

* 1. Go to the Tools console
* 2. In the folder tree, select the Tools/MSM/Rollout Configurations folder.
* 3. Click New > New Page and provide values for the Rollout Configuration properties:   
  + **Title**: The title of the Rollout Configuration, such as My Rollout Configuration
  + **Name**: The name of the node that stores the property values, such as myrolloutconfig
  + Select **RolloutConfig Template**.
* 4. Click **Create**.
* 5. Open the rollout configuration that you created.
* 6. Click **Edit**.
* 7. In the Rollout Config dialog, use the Sync Trigger drop-down menu to select the action that causes the rollout to occur.
* 8. Click **OK** to save the changes.

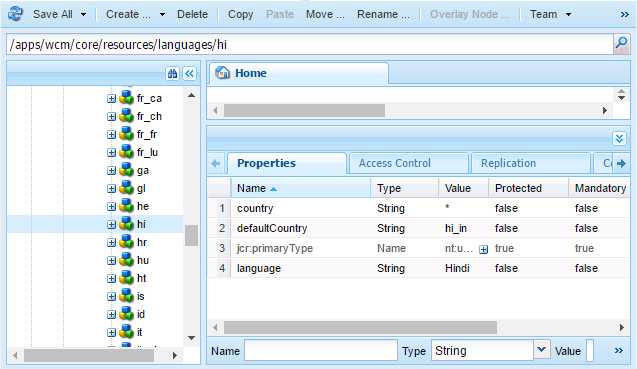
**PROCEDURE TO ADD SYNCHRONIZATION ACTIONS TO THE ROLLOUT CONFIGURATIONS**

* 1. Open CRXDE Lite.
* 2. Select the jcr:content node below your rollout configuration node. For example, for the rollout configuration with the Name property of myrolloutconfig, select the /etc/msm/rolloutconfigs/myrolloutconfig/jcr:content node.
* 3. Click Create > Create Node, configure the following node properties and click OK:
  + **Name**: The node name of the synchronization action. The name must be the same as teh Action Name in the table Synchronisation Actions, for example contentCopy, or workflow
  + **Type**: cq:LiveSyncAction
* 4. Select the action node that you created and add the following property to the node:
  + **Name**: The property name of the action. The name must be the same as the Property Name in the table Synchronisation Actions, for example enabled
  + **Type**: String
  + **Value**, the property value of the action. For valid values, see the Property Value column in Synchronisation Actions, for example true.
* 5. Add and configure as many syncronization action nodes as you require. Rearrange the action nodes so that their order matches the order in which you want them to occur. The topmost action node occurs first.
* 6. Click Save All.

1. **How to change Language Names and Default Countries?**

Procedure to modify the languages

* 1. Open **CRXDE Lite** (http://host:port/crx/de/index.jsp)
* 2. Select the **/apps** folder and click Create > Create Folder. Name the folder wcm.
* 3. Repeat the previous step to create the **/apps/wcm/core** folder tree. Create a node of type sling:Folder in core called resources.
* 4. Right-click the **/libs/wcm/core/resources/languages** node and click **Copy**. Right-click the **/apps/wcm/core/resources** folder and click **Paste**.
* 5. Each child node of languages node represents a language or a language-country:  
  + The name of the node is the languge code (such as en or hi), or the language\_country code (such as en\_us or de\_ch).
  + The language property of the node stores the full name of the language for the code.
  + The country property of the node stores the full name of the country for the code.
  + When the node name consists only of a language code (such as en), the country property is \*, and an additional defaultCountry property stores the code of the language-country to indicate the country to use.



* Modify the child nodes as required.
* Click Save All.
* Go to OSGI console. Click configuration
* Open the CQ WCM Language Manager.(http://host:port/system/console/configMgr/com.day.cq.wcm.core.impl.LanguageManagerImpl)
* Locate and click Day CQ WCM Language Manager, and change the value of Language List to /apps/wcm/core/resources/languages, then click Save.

