Liferay OSGI

OSGI –

* Open service gateway initiative
* Runtime container is called osgi container
* It does multiple foundation like eclipse, apache etc.

WHY Used –

* provide development environment for moduler applications. (development karva module provide kare)
* Liferay develop on osgi framework
* It comprises the code module created and tested in parallel way. It’s plateform which module and moduler application started, used, stopped and uninstalling.
* It’s runtime environment so its lightweight, fast and secure.
* Module dependancy manage by automatically.
* Which module classes are exported that visible all others classes are hide
* **Module projects** create basic project skeleton with default deployable components

Sepration bundle – bundle name should be unique(it’s autocomplete but keep in mind) **classes from bundle package/ export package**

* Major changes on version – breaks the API (just like some coding standard change)
* Minor changes on version – without break API change
* Micro – not effect on the api like (change of property etc not a version)
* Version identiify - [1.1,1.2]/[1.2,2.2) 1.1,1.2 including, 1.2 including but 2.2 not

@component annotation have three types

* Immediate – should not be lazy loading
* Property - where the clock-api and clock-impl code execute
* Service – define as self or modal of class

**Classloader** : it’s runtime environment parts of dynamically called to class in jvm and manage by osgi container.

* What is classloader : java class taking care of classloader.
* Portal/DXP web server deployed on app server and osgi bundle deployed on module framework.

build.gradle : dependencies related to particular module

Configuration folder: configure different environment like dev, qa etc.

Module – u can create module by cli like **blade creae –t (portlet-name) -p (package-name like com.stpl.liferay.training) -c (module-class-name using on widget) (module-name)**

Jsp

* Bundler container jsp classloader
* Jsp servlet bundle classloader
* Javax environment language implementation bundle classloader

**Workspace provider**

* Blade cli
  + The methods create,deploy,gw,help,init,convert,install
* Gradlew and maven wrapper

Osgi Lifecycle

* Web application seraches bootstrap
* If classes or resource not there it picks it’s own class. If still not found the class searches the system and common classloader
* It's always pick first parent jar

Gradlew dependancies scope :

**compile module define on either bnd.bnd or build.gradle. compileInclude types are group,name version**

* Compile :
* CompileOnly : compile with another module
* CompileInclude : out sourcing external package added here
* Provided
* Runtime
* TestCompile

Osgi Architecture

* Bundles – A jar file with headers, manageable, testable
* Services Layer – service registry, allow to find, publish and bind the component of the service.(keeping track class of components and what they do with reference act retrieve service)
* LifeCycle Layer – manage the bundle using install, uninstall, start, stop, update
* Modules layer – manage class loader, import-export packages
* Execution Layer – Java run time platform compatibility

Override OSGI Service Reference

* Multiple service wrapper is initialized by property = { "service.ranking:Integer=100" }
* Reference cardinality
  + Mandatory -(default)
  + Optional – reference are not mandatory
  + Multiple – reference are not required, but multiple resource satisfy the reference and component take all of them
  + AtLeastOne – similar to multiple, but at least one reference to start component
* Reference policy
  + Statically – reference is required, and component are notified by become available(default)
  + Dynamically – not required and accept new reference of become available.