Liferay 7 Knowledge Checks

Describe OSGi within the context of Liferay 7 development.

OSGi is a set of standards for building modular systems. A module is the single unit of distribution and deployment in a modular architecture. OSGi defines, among other things, how modules can depend on each other and communicate. It also defines the packaging format for modules: OSGi bundles. An OSGi module is just a typical JAR file. when a service is deployed, it becomes part of a service registry maintained by Liferay’s OSGi container. Liferay supports a wide range of the OSGi family of standards through its own implementations and also integrates the high quality implementations of the Apache Felix and Eclipse Equinox projects. The OSGi container manages module life cycles dynamically. Modules can be installed, started, updated, stopped, and uninstalled while Liferay DXP is running, making deployment a snap.

Describe an example of the Liferay Service Builder and its purposes and outcomes.

Liferay Service Builder is a model-driven code generation tool that lets us define custom object models called entities. Service Builder generates a service layer through object-relational mapping (ORM) technology that provides a clean separation between object model and code for the underlying database.

It generates following layers of code:

* The model layer is responsible for defining objects to represent your project’s entities,
* The persistence layer is responsible for saving entities to and retrieving entities from the database,
* The service layer is responsible for exposing CRUD and related methods for your entities as an API.

Each entity Service Builder generates contains a model implementation class. Each entity also contains a local service implementation class, a remote service implementation class, or both, depending on how we configure Service Builder in service.xml file.

Explain the difference between a theme and themelet in Liferay 7.

A Liferay Theme is the overall look and feel for a site. Themes are a combination of CSS, JavaScript, HTML, and FreeMarker templates. Liferay has its own set of base themes, called styled and unstyled that create the default look and feel we see at first start. Liferay DXP provides several tools and environments that we can use to create custom themes.

Themelets are small, extendable, and reusable pieces of code. Whereas themes require multiple components, a themelet only requires the files we wish to extend. This creates a more modular approach to theme design. Themelets can be used to modify one aspect of the UI, that we can then reuse in other themes.

A themelet can consist of CSS and JavaScript. Themelets do not support theme templates.

Describe the portlet lifecycle from the perspective of a simple form input portlet, beginning with navigating to the page where the portlet is deployed.

Web apps in Liferay Portal are called portlets. Like many web apps, portlets process requests and generate responses.

Lifecycle of Portlet:

* **Init**: The init() method is called when portlet is deployed and instantiated by portlet container.
* **Render** : In this phase portlet generates content and renders on webpage.

The render phase is called in below cases:

* + The page that contains portlet is rendered on web page
  + After completing Action Phase
  + After completing Event Processing phase
* **Action**: This phase will be called when any action performed, its result of user actions such as add,edit, delete etc.

**processEvent()** : This method will be called when any event is triggered.

**serveResource()** : This method will be called when any resource is served with resource URL.

* **Destroy**: The destroy() method will be called when portlet is undeployed from portlet container.

When we navigate to the page where portlet deployed, then first its call render phase and based on user interaction it call action phase.