Liferay Developer Certification Practice Exam (Sample)

Study Guide

Everything you need from our exam experts!

Questions

1. What is the JavaScript method called after all portlets on a page have finished loading?

A. Liferay.Portlet.ready(fn)

B. AUI().ready(fn)

C. Liferay.on('allPortletsReady',fn)

D. None of the above

2. Which component allows for the display of dependent objects based on conditions in Liferay?

A. Asset Publisher

B. Service Builder

C. Content Targeting

D. Custom Fields

3. How does Liferay's Access Control function?

A. It manages user permissions based on predefined roles

B. It encrypts sensitive user data

C. It ensures high availability of the server

D. It tracks user behavior and analytics

4. What method should be used to get the value of a custom field for a User object named "user"?

A. ExpandoLocalServiceUtil.getAttribute(user)

B. user.getExpandoBridge().getAttribute()

C. user.getExpando()

D. PortalUtil.getExpando(user)

5. What functionality does a Layout Template NOT provide in Liferay?

A. Setting up the initial layout structure

B. Managing user roles

C. Defining page layout grids

D. Creating visual designs for pages

6. Which tag in liferay-hook.xml is used to create a Struts action path?

A. <struts-action-path>

B. <struts-action-impl>

C. <struts-hook>

D. <override-struts-path>

7. What functionality is used to create custom fields in Liferay?

A. Theme Management

B. Object Management

C. Site Configuration

D. User Administration

8. Which statements about plugins are true?

A. All plugins are hot-deployable

B. Plugins can be used to modify core portal behavior

C. Plugins can be used to modify the layout of a page

D. Liferay supports six different types of plugins

9. How does Liferay define a User?

A. An entity that can generate new content

B. An individual who can access the system and its resources

C. A role assigned to manage content only

D. A temporary session for accessing Liferay features

10. Which file does the Plugins SDK use?

A. portal-service.jar

B. JAR files from the user's home directory

C. portal-impl.jar

D. portal-plugin.jar

Answers

1. C
2. C
3. A
4. B
5. B
6. A
7. B
8. B
9. B
10. A

Explanations

1. What is the JavaScript method called after all portlets on a page have finished loading?

A. Liferay.Portlet.ready(fn)

B. AUI().ready(fn)

C. Liferay.on('allPortletsReady',fn)

D. None of the above

The correct response references the JavaScript method `Liferay.on('allPortletsReady', fn)`, which is specifically designed to execute a function when every portlet on a Liferay page has successfully finished loading. This is crucial in scenarios where you need to ensure that all components are fully initialized before performing further actions like manipulating the DOM or triggering events. The reason this method is particularly valuable is that loading multiple portlets can lead to variable loading times, influencing how and when scripts interact with the page. Using `Liferay.on('allPortletsReady', fn)` ensures that the subsequent code runs only after all portlets are available, thereby reducing the chances of errors related to elements not being present. In contrast, the other options do not serve the same purpose. While `Liferay.Portlet.ready(fn)` and `AUI().ready(fn)` are both useful for specific loading scenarios, they do not guarantee that every single portlet on the page is ready. They might only relate to individual portlet readiness or a more generalized document readiness, rather than the comprehensive wait for all portlets. Therefore, `Liferay.on('allPortletsReady', fn)` stands out as the correct method for the scenario described.

2. Which component allows for the display of dependent objects based on conditions in Liferay?

A. Asset Publisher

B. Service Builder

C. Content Targeting

D. Custom Fields

The component that allows for the display of dependent objects based on conditions in Liferay is Content Targeting. Content Targeting enables developers to show or hide content dynamically based on user attributes or specific conditions, such as user roles, location, or behavior. It uses rules and targeting criteria to ensure that the right content reaches the appropriate audience, enhancing personalization and user engagement on the platform. By using Content Targeting, developers can create sophisticated experiences that update in real time, ensuring that each user gets relevant information tailored to their needs. This capability is particularly valuable in marketing campaigns where presenting tailored content can significantly increase conversion rates. Other components, while useful for different purposes, do not primarily focus on the dynamic display of dependent objects based on specific criteria. For instance, having an Asset Publisher would help in aggregating and displaying various content assets but does not inherently manage conditional display based on user interactions or criteria. Similarly, Service Builder helps in generating services and data access layers, and Custom Fields are useful for adding additional metadata to objects but do not facilitate dynamic display conditions on their own.

3. How does Liferay's Access Control function?

A. It manages user permissions based on predefined roles

B. It encrypts sensitive user data

C. It ensures high availability of the server

D. It tracks user behavior and analytics

Liferay's Access Control function primarily manages user permissions based on predefined roles. This role-based access control (RBAC) allows administrators to define and customize user permissions efficiently. By categorizing users into different roles, such as administrators, editors, or guests, Liferay enables a structured approach to content management and access rights. This ensures that users only see and interact with the parts of the site that they are authorized to access, enhancing both security and usability. In practice, this means that roles can be assigned various levels of access to content, applications, and features within the portal. For example, an editor might have permission to create and edit content, while a guest might only have permission to view it. This granular control helps organizations maintain their security policies and compliance requirements by clearly defining who can do what within the system. Other options focus on unrelated functions of Liferay. Encrypting sensitive user data pertains to security measures for data protection, ensuring confidentiality. High availability addresses server resilience and uptime, which is crucial for performance but not directly related to user access. Tracking user behavior and analytics relates to monitoring interactions within the portal for insights, which also doesn't fall under access control. Therefore, the role-based management of permissions is essential to maintaining a secure and user

4. What method should be used to get the value of a custom field for a User object named "user"?

A. ExpandoLocalServiceUtil.getAttribute(user)

B. user.getExpandoBridge().getAttribute()

C. user.getExpando()

D. PortalUtil.getExpando(user)

To retrieve the value of a custom field from a User object in Liferay, using the method provided in the selected answer is appropriate because it utilizes the ExpandoBridge associated with the User object. The ExpandoBridge allows access to dynamic fields—those that are not hard-coded in the database schema but rather stored as custom attributes. By calling `user.getExpandoBridge().getAttribute()`, you are specifically referencing the ExpandoBridge tied to the User, which manages the custom fields. This method is designed to fetch the value of a specific custom field based on the field name provided as an argument, ensuring that it looks into the correct structure that holds these custom attributes. Other choices do not effectively target the necessary approach for the context of retrieving custom fields. For instance, using ExpandoLocalServiceUtil requires the specification of the custom field and might involve additional parameters like company ID or class type, making it less straightforward for this specific task. The `user.getExpando()` method provides access to the Expando object itself, but lacks the functionality to directly fetch attribute values. Lastly, the PortalUtil class generally serves as a utility for various portal-related tasks rather than specifically managing custom fields for User objects. Thus, the chosen method correctly fits the

5. What functionality does a Layout Template NOT provide in Liferay?

A. Setting up the initial layout structure

B. Managing user roles

C. Defining page layout grids

D. Creating visual designs for pages

A Layout Template in Liferay is primarily focused on establishing the structure and presentation of a webpage. This includes defining the page layout grids, which determine how content and components are arranged on the page, and setting up the initial layout structure, which sets the foundational organization for sections of content. Additionally, Layout Templates facilitate the creation of visual designs for pages, allowing developers to use CSS and other design elements to enhance the appearance of the site. This emphasizes the purpose of Layout Templates in controlling how content is displayed and ensuring consistency across pages. However, managing user roles falls outside the scope of Layout Templates. User roles pertain to the permissions and access controls within Liferay, which are handled by the role management system rather than layout design. Thus, the functionality surrounding user roles is distinctly separate from the responsibilities of a Layout Template, confirming that this is the correct indication of what a Layout Template does not provide.

6. Which tag in liferay-hook.xml is used to create a Struts action path?

A. <struts-action-path>

B. <struts-action-impl>

C. <struts-hook>

D. <override-struts-path>

The tag used to create a Struts action path in the liferay-hook.xml file is <struts-action-path>. This tag defines a specific action that can be invoked by the Struts framework, which is an integral part of Liferay's web application structure. By utilizing this tag, developers can map user requests to specific actions, allowing for better organization and management of the application's interactions with end users. In the context of a Liferay Hook, the <struts-action-path> provides a means to customize or extend the behavior of existing Struts actions. This can be particularly useful for adding new functionalities or overriding existing ones without modifying the core framework, promoting a modular and maintainable architecture. The other tags mentioned do not serve the purpose of creating an action path within the Struts framework. Thus, understanding how to effectively use <struts-action-path> is vital for those looking to manage Struts actions within their Liferay projects effectively.

7. What functionality is used to create custom fields in Liferay?

A. Theme Management

B. Object Management

C. Site Configuration

D. User Administration

The ability to create custom fields in Liferay is primarily handled through Object Management. This functionality allows developers and administrators to define new object types and customize their attributes, including the creation of custom fields. Objects can be tailored to meet specific business requirements, enabling the addition of fields that capture unique information relevant to a particular application or use case. Object Management encompasses the design of data models that are flexible and extendable, allowing users to specify the types of data they need to gather and how it should be structured. This capability is essential for applications that require specific data handling beyond the default fields provided by Liferay. The other options, such as Theme Management, Site Configuration, and User Administration, focus on different aspects of Liferay's functionality. Theme Management pertains to the visual aspects of sites, Site Configuration is about managing site settings and global configurations, and User Administration deals with managing user accounts and permissions. None of these directly contribute to the creation of custom fields in the same way Object Management does.

8. Which statements about plugins are true?

A. All plugins are hot-deployable

B. Plugins can be used to modify core portal behavior

C. Plugins can be used to modify the layout of a page

D. Liferay supports six different types of plugins

The statement indicating that plugins can be used to modify core portal behavior is true. In Liferay, plugins provide a powerful mechanism to extend the functionality and customize the behavior of the portal. This capability allows developers to create custom applications, enhance existing features, or change how the core functionalities work, which is essential for adapting the portal to meet specific business requirements. Customizing core behavior might include creating new portlets, overriding service behaviors, or developing themes that sustain the unique branding of the organization's portal. This flexibility is one of the reasons why plugins are an integral part of Liferay's architecture, providing ways to tailor the platform for various use cases. The assertion regarding all plugins being hot-deployable is not universally accurate, as certain changes or enhancements may require a restart for the full effect to take place. While many plugins support hot deployment, it does not apply to every plugin type. While plugins do enable layout modifications, they are primarily associated with functionality rather than solely layout control, which is why that statement is more ambiguous. Additionally, while Liferay does support different types of plugins, saying there are exactly six does not capture the full extent of available plugin types and could lead to misunderstandings regarding Liferay's extensibility. Understanding these aspects of L

9. How does Liferay define a User?

A. An entity that can generate new content

B. An individual who can access the system and its resources

C. A role assigned to manage content only

D. A temporary session for accessing Liferay features

Liferay defines a User as an individual who can access the system and its resources. This definition encompasses a broad range of functionalities that a user can perform within the Liferay platform, including logging in, utilizing applications, and accessing content. In the context of Liferay, users are not just content creators; they can also be administrators, contributors, or viewers, which illustrates their varied roles within the system. Access to the system's features and resources is central to what being a User means in Liferay, emphasizing the platform's focus on user interaction and engagement. The other options do not capture the full essence of what a User is in Liferay. While content generation and management roles are important aspects of user functionality, they do not represent the primary definition, which stresses access and engagement with the system.

10. Which file does the Plugins SDK use?

A. portal-service.jar

B. JAR files from the user's home directory

C. portal-impl.jar

D. portal-plugin.jar

The Plugins SDK in Liferay primarily utilizes the "portal-service.jar" file. This file is crucial because it contains the service APIs and models that allow developers to create and extend Liferay functionalities through plugins. By utilizing the classes and methods defined in "portal-service.jar," developers can interact with the core services of Liferay, making it possible to build custom applications or tools that seamlessly integrate with the platform. The other choices refer to different components of Liferay's architecture but are not directly applicable to the functionality of the Plugins SDK. For instance, "portal-impl.jar" involves the implementation layer of the portal, which is not intended for direct use in developing plugins. Meanwhile, "portal-plugin.jar" is not a standard designation in typical Liferay usage, and user home directory JAR files are not standard files that the Plugins SDK accesses for developing plugins. Thus, focusing on "portal-service.jar" is the correct and logical choice for understanding the framework that the Plugins SDK operates upon.