Free Questions for AD0-E134

Shared by Baxter on 18-05-2023

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Which attribute must be present in all filter rules in AEM dispatcher configuration?

Options:

A- /type

B- /selectors

C- /url

D- /glob

Answer:

D

Explanation:

The /glob attribute is mandatory for all filter rules in the dispatcher configuration. It specifies a pattern that is matched against the incoming request URL. If the pattern matches, the rule is applied. The /glob attribute can use wildcards (*) and placeholders ($1, $2, etc.) to match parts of the URL. Reference:
https://experienceleague.adobe.com/docs/experience-manager-dispatcher/using/configuring/dispatcher-configuration.html?lang=en#filter-rules

Question 2

Question Type: MultipleChoice

A developer needs to create a new Title component. The requirements are:

1. The layout must be the same as the Title core component

2. The text property must have the page title as prefix (e.g., Page Title -)

3. The component must be reusable

Which approach is recommended?

Options:

A- 1. Create a Proxy Component of Title core component

2. Create a Custom Sling Model that overrides the default behavior

3. Customize the component template

B, 1. Create a custom component from scratch

2. Create a Custom Sling Model for the component that follows the requirement

3. Create a Model Exporter

C- 1. Create a Proxy Component from Title core component

2. Create a Custom Sling Model that overrides the default behavior

Answer:

A

Explanation:

A proxy component is a site-specific component that inherits from a core component and allows customization of the component name, group, dialog, and behavior. A proxy component can refer to any version of the core component by changing the sling:resourceSuperType property. A custom sling model can be used to implement the logic for adding the page title as prefix to the text property. A component template can be used to define the layout of the component.

Question 3

Question Type: MultipleChoice

An AEM application is expected to export a content fragment in JSON format without any customization for a headless

implementation.

What is the recommended approach?

Options:

A- Use AEM Assets HTTP API

B- Use Core components to export JSON

C- Use Sling Exporter framework

Answer:

A

Explanation:

AEM Assets HTTP API is a RESTful API that allows access to content fragments in JSON format without any customization. The API supports CRUD operations on content fragments and their variations, as well as querying and searching for content fragments based on metadata or full-text search. Reference: https://experienceleague.adobe.com/docs/experience-manager-65/assets/extending/assets-api-content-fragments.html?lang=en

Question 4

Question Type: MultipleChoice

A developer is on an AEM application that is being used to calculate an employee's salary. The calculation is done in an OSGi service called CalculationService. This service class has a dependency on one other service, called the EmployeeService.

How should the developer make sure that the critical code in the CalculationService has a high unit test coverage?

Options:

A- Use a mock framework in the unit test to inject the CalculationService

B- Instantiate the EmployeeService in the unit test and pass it to the constructor of the CalculationService

C- Use a mock framework in the unit test to inject the EmployeeService

D- Use the feature flag in the unit test to disable the calls to the EmployeeService

Answer:

C

Explanation:

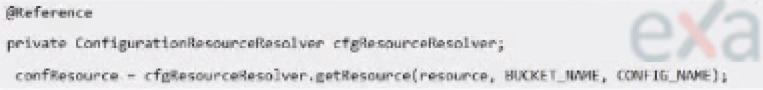
A mock framework is a tool that allows creating mock objects that simulate the behavior of real objects in a controlled way. A mock framework can be used in a unit test to inject the EmployeeService dependency into the CalculationService and verify its interactions. This way,

the unit test can focus on testing the logic of the CalculationService without relying on the actual implementation of the EmployeeService. Reference:
https://sling.apache.org/documentation/development/sling-testing-tools.html
https://wcm.io/testing/aem-mock/usage.html

Question 5

Question Type: MultipleChoice

A developer is using sling context-aware configuration trying to get the configuration resource using:



```
@Reference
private ConfigurationResourceResolver cfgResourceResolver;
 confResource = cfgResourceResolver.getResource(resource, BUCKET_NAME, CONFIG_NAME);
```

This works as intended in author and in publish when logged in lo publish as admin. However this gives a null when run as anonymous. Which method is going to fix the issue?

Options:

A- Applying read permissions to anonymous user for /content directory

B- Applying write permissions to anonymous user for/conf directory.

C- Applying read permissions to anonymous user for /etc directory.

D- Applying read permissions to anonymous user for /cont directory

Answer:

A

Explanation:

The issue arises because the anonymous user does not have the necessary read permissions for the /content directory. When using Sling context-aware configurations, access to the configuration resource is required. If the anonymous user lacks read permissions, the configuration cannot be retrieved, resulting in a null value.

To fix this issue, you need to grant read permissions to the anonymous user for the /content directory.

Steps to apply read permissions:

Access CRXDE Lite: Log into your AEM instance and navigate to CRXDE Lite (http://localhost:4502/crx/de).

Navigate to the /content Directory: In the CRXDE Lite interface, browse to the /content directory.

Set Permissions:

Right-click on the /content directory and select 'Permissions'.

Add the anonymous user (if not already present).

Grant read permissions to the anonymous user.

Save Changes: Apply the changes and ensure that they are saved correctly.

Test the Configuration: Log out of the AEM instance and test the context-aware configuration as an anonymous user to ensure that the configuration resource can now be retrieved.

By granting read permissions to the anonymous user for the /content directory, you allow access to the necessary configuration resources, resolving the issue.

Adobe Experience Manager Security Permissions

Question 6

Question Type: MultipleChoice

An AEM server is overloaded with too many concurrently running workflows. The developer decides to reduce the number of concurrent workflows.

What should be configured to reduce the number of concurrent workflows?

Options:

A- The number of threads in Scheduler

B- The number of threads in Apache Felix Jetty Http Service

C- Launchers for each workflow

D- Maximum Parallel Jobs in OSGI console

Answer:

D

Explanation:

Maximum Parallel Jobs is a configuration property that controls how many workflows can run concurrently on an AEM instance. Reducing this value can help to avoid overloading the server with too many workflows.

Question 7

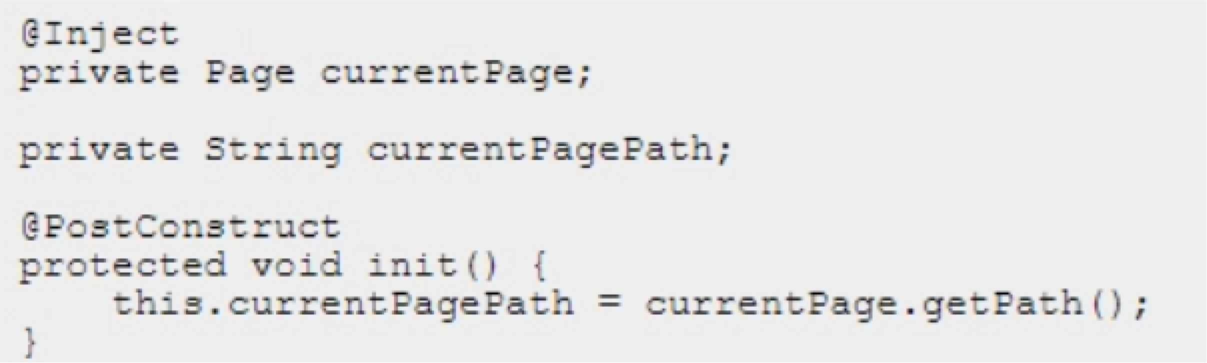
Question Type: MultipleChoice

A snippet throws an exception at runtime:

@Model(adaptables = {Resource.class}) public class MyCustomModel {

(SSlingObject

private Resource resource;



```
@Inject
private Page currentPage;

private String currentPagePath;

@PostConstruct
protected void init() {
    this.currentPagePath = currentPage.getPath();
}
```

What should the developer add to fix it?

Options:

A- defaultInjectionStrategy = DefaultInjectionStrategy property to @Model Class annotation
B- (Optional annotation to page field
C- throws Exception at the end of the init method declaration
D- SlingHttpServletRequest.class to adaptables property of Model Class annotation

Answer:

A

Explanation:

The developer should add the defaultInjectionStrategy = DefaultInjectionStrategy property to the @Model Class annotation to fix the snippet. The defaultInjectionStrategy property defines how the Sling Model handles missing or null values for the injected fields. By default, the Sling Model uses the REQUIRED injection strategy, which means that all fields must have a non-null value or else an exception is thrown. By setting the defaultInjectionStrategy property to OPTIONAL, the Sling Model allows null values for the injected fields and does not throw an exception. This way, if the page field is null because the resource is not a page, the Sling Model can still work without errors. Reference: https://sling.apache.org/documentation/bundles/models.html https://sling.apache.org/documentation/bundles/models.html#optional-injection

Question 8

Question Type: MultipleChoice

Which practice should be used to push a code fix to make it into the current release candidate?

Options:

A- Make the fix in CRX where the current release candidate is currently deployed.

B- Create a new release candidate from the master branch.

C- Make the fix locally and upload the package to where the release candidate is deployed.

D- Cherry-pick the fix commit into the release candidate.

Answer:

D

Explanation:

To push a code fix into the current release candidate, the best practice is to cherry-pick the fix commit into the release candidate branch. Cherry-picking allows you to apply specific changes from one branch to another, ensuring that only the necessary fixes are included without introducing unrelated changes.

Here's how to cherry-pick a commit:

Identify the Commit: Locate the commit hash of the fix you need to apply. This can be found in your version control system (e.g., Git).

Checkout the Release Candidate Branch:

```
git checkout release-candidate-branch
```

Cherry-pick the Commit:

git cherry-pick <commit-hash>

Resolve Conflicts (if any): If there are conflicts, resolve them manually and then continue the cherry-pick process:

git add <resolved-files>

git cherry-pick --continue

Test the Changes: Ensure that the changes are tested thoroughly in the release candidate environment to confirm the fix works as expected.

Push the Changes:

git push origin release-candidate-branch

Cherry-picking ensures that only the required changes are applied to the release candidate, maintaining the stability and integrity of the codebase.

Git Cherry-pick Documentation

Question 9

Question Type: MultipleChoice

A developer has to create a Logger and Writer pair for the company's application logging. Which OSGi configurations should the developer use?

Options:

A- Apache Sling Logging Logger Configuration and Apache Sling Logging Configuration

B- Apache Sling Request Logger and Apache Sling Logging Writer Configuration

C- Apache Sling Logging Logger Configuration and Apache Sling Logging Writer Configuration

Answer:

C

Explanation:

The Apache Sling Logging Logger Configuration and Apache Sling Logging Writer Configuration are the OSGi configurations that the developer should use to create a Logger and Writer pair for the company's application logging. The Logger Configuration defines the log level and the log file name for a given logger name or category. The Writer Configuration defines the file size, number of files, and file location for a given log file name. Reference: https://experienceleague.adobe.com/docs/experience-manager-65/deploying/configuring/configure-logging.html?lang=en#configuring-log-files

Question 10

Question Type: MultipleChoice

In a non-optimized website, the final HTML generated for a typical page by publish instance includes a relatively large number of elements that refer to other script files loaded from AEM environment. The developer wants to minimize these network calls by combining all required client library code into a single file to reduce the number of back-and-forth requests on page load.

Which step should a developer take to solve this issue?

Options:

A- Embed the required libraries into an app-specific client library using the allowProxy property of the cq:Clientl_ibraryFolder node

B- Add the categories property of the cq:Clientl_ibraryFolder node into an app-specific client library folder

C- Embed the required libraries into an app-specific client library using the dependencies property of the cq:Clientl_ibraryFolder node

D- Embed the required libraries into an app-specific client library using the embed property of the cq:ClientLibraryFolder node

Answer:

C

Explanation:

The embed property of the cq:ClientLibraryFolder node allows embedding code from a client library into another client library. At runtime, the generated JS and CSS files of the embedding library include the code of the embedded library. This reduces the number of network calls and improves performance. Embedding code is useful for providing access to libraries that are stored in secured areas of the repository. Reference: https://experienceleague.adobe.com/docs/experience-manager-cloud-service/content/implementing/developing/full-stack/clientlibs.html?lang=en#embed https://experienceleaguecommunities.adobe.com/t5/adobe-experience-manager/embed-property-in-client-libs/m-p/426858