DUMPSARENA

Adobe Experience Manager Sites Developer Professional

Adobe AD0-E123

Version Demo

Total Demo Questions: 7

Total Premium Questions: 71

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Development	44
Topic 2, Deployment	21
Topic 3, Debugging	6
Total	71

QUESTION NO: 1

A developer wants to overwrite an OSGi configuration via the codebase.

Which one of the Maven modules is the place to create this configuration on the codebase?

A. ui.content

B. ui.apps

C. core

D. ui.config

ANSWER: D

Explanation:

ui.config is the dedicated Maven module for source-controlled OSGi configuration in the current Adobe Experience Manager project structure. Configuration files are stored in this module under its JCR source tree, typically in run-mode-specific folders such as `config`, `config.author`, or `config.publish`. They are normally named using the OSGi PID and use the `.cfg.json` format. During the Maven build and deployment process, the module packages these files so AEM installs them as repository-based OSGi configurations. This lets a developer deliberately override a service's effective settings through the application codebase, while keeping configuration separate from Java implementation code and from application content. The separation is useful for environment and run-mode targeting, review in version control, and repeatable deployments through Cloud Manager or Maven. Adobe's AEM Maven project archetype describes `ui.config` as the module containing OSGi configurations, and Adobe's OSGi configuration documentation describes repository-based configuration deployment and run-mode resolution. See the AEM Project Archetype documentation and Configuring OSGi for AEM.

QUESTION NO: 2

A developer is creating an editable template for a new site. Every page created from the template must contain a locked header component, while each new page should begin with an editable promotional component.

Which two actions should the developer take? (Choose two.)

A. Place the header component in the template structure.

B. Place the promotional component in the template initial content.

C. Place the header component in the template initial content.

D. Configure the promotional component as an allowed component policy.

ANSWER: A B

Explanation:

The template **structure** defines content that is locked on pages created from the template. Therefore, placing the header in the structure ensures that page authors cannot remove or alter that structural component on individual pages.

Initial content is copied to a page when the page is created from the editable template. Placing the promotional component in initial content gives every new page a starting component that authors can subsequently edit or remove. Component policies define authoring behavior and allowed components, but they do not themselves create the required header or promotional content.

QUESTION NO: 3

A project component extends a Core Component and inherits its dialog through Sling Resource Merger. The developer must remove one inherited tab and move a project-specific tab before another inherited tab without copying the complete dialog.

Which two Sling Resource Merger properties should be used? (Choose two.)

A. `sling:hideResource`

B. `sling:orderBefore`

C. `jcr:orderBefore`

D. `sling:hideChildren`

ANSWER: A B

Explanation:

`sling:hideResource` is used on the corresponding local dialog resource to suppress an inherited resource, such as an inherited tab. This allows the remainder of the supertype dialog to remain available through resource merging.

`sling:orderBefore` controls the position of a local resource relative to a sibling resource in the merged result. It is appropriate for moving a project-specific tab before an inherited tab. `jcr:orderBefore` is not the Sling Resource Merger property used to control the effective merged order, and `sling:hideChildren` hides child resources rather than removing the tab resource itself.

QUESTION NO: 4

Given this configuration property:

```
/ignoreUrlParams
{
  /0001 { /glob "*" /type "deny" }
  /0002 { /glob "search" /type "allow" }
  /0003 { /glob "order" /type "allow" }
}
```

Which page will be cached on the dispatcher?

A. /myproduct/myrecipe.html?search=searchparam

B. /myproduct/myrecipe.html?search=s&order=asc&brand=ad

C. /myproduct/myrecipe.html?brand=mybrand

ANSWER: C

Explanation:

`/myproduct/myrecipe.html?brand=mybrand` will be cached because the Dispatcher's `/ignoreUrlParams` rules are evaluated to determine whether query-string parameters affect cacheability. The catch-all rule for `*` with `/type "deny"` tells Dispatcher to ignore parameters by default. In this context, "deny" means the parameter is denied from consideration in the cache URL, rather than that the request itself is denied. Since `brand` has no later rule that allows it, it remains ignored.

Consequently, Dispatcher can serve or create the cached representation for `/myproduct/myrecipe.html` without treating `brand=mybrand` as a cache-distinguishing parameter. This is appropriate for parameters that do not alter the rendered page content, such as tracking or non-functional URL parameters. The more specific allow rules for named parameters take precedence over the general wildcard behavior for those parameter names, ensuring that parameters intentionally designated as cache-relevant are not silently ignored.

Adobe documents `/ignoreUrlParams` as the Dispatcher configuration mechanism for controlling which request parameters are ignored during caching. See the Dispatcher configuration documentation for ignoring URL parameters and the Adobe Experience Manager Dispatcher documentation.

QUESTION NO: 5

A developer wants to render Hello World' only in edit mode. Which approach would be used?

A.

Hello World

B.

Hello World

ANSWER: A

Explanation:

`<p data-sly-test="${wcmmode.edit}">Hello World</p>` is the appropriate HTL implementation because AEM exposes the current authoring mode through HTL's global `wcmmode` object. Its `edit` property evaluates to true when the page is being viewed in Edit mode in the authoring environment. The `data-sly-test` block statement evaluates that expression before rendering its host element. Therefore, when an author is editing the page, the paragraph and its "Hello World" text are included in the generated output; when the page is in another WCM mode, the element is omitted. This is useful for author-only hints, placeholders, and component guidance that should not be presented in normal viewing or published output. The expression must use the exact `wcmmode.edit` property and standard HTL expression delimiters, `${...}`. Adobe documents `wcmmode` as an HTL global object and lists `edit` as the indicator for Edit mode. Adobe's HTL specification also documents `data-sly-test` as the conditional rendering mechanism. See HTL Global Objects and the HTL block statements specification.

QUESTION NO: 6

A developer wants to send a SAML Authentication Request to a specific URL of a system entity that creates, maintains, and manages identity information.

Which property of SAML Authentication Handler configuration must be configured with this URL?

A. Identity Provider URL

B. Path

C. Default Redirect

ANSWER: A

Explanation:

Identity Provider URL is the required SAML Authentication Handler property because the described system entity is the Identity Provider (IdP). In a SAML single sign-on flow, AEM acts as the service provider and creates a SAML authentication request when a user needs to authenticate. The Identity Provider URL identifies the IdP endpoint to which AEM sends that request, typically the IdP's SSO service endpoint. The IdP then authenticates the user, manages the relevant identity information, and returns a signed SAML response/assertion to AEM. Configuring this value accurately is essential: it establishes the destination used by the authentication handler for the redirect or request exchange and must correspond to the endpoint published in the IdP's SAML metadata or configuration. Adobe's SAML 2.0 authentication documentation identifies the IdP URL as a configuration element for connecting AEM to an external identity provider. See Adobe Experience Manager SAML 2.0 Authentication Handler documentation and the Adobe Experience League SAML authentication overview.

QUESTION NO: 7

A developer needs to remove all the selectors from the linkValue.

Assuming linkValue = 'path/page.woo.too.html', which two HTL approaches would be used to complete this task? (Choose two.)

A. ${linkValue @ removeSelectors=['woo', 'too']}

B. ${linkValue @ removeSelectors=['foo', 'bar']}

C. ${linkValue @ selectors=''}

D. ${ linkValue @ selectors}

ANSWER: A C

Explanation:

HTL provides URI-manipulation options that operate on a link's Sling URI components, including selectors. Using `removeSelectors=['woo', 'too']` explicitly removes each selector from `path/page.woo.too.html`, producing `path/page.html` while preserving the resource path and the `html` extension. This is appropriate when the selectors to remove are known and should be named directly.

Using `selectors=''` replaces the URI's selector list with an empty value. Because the replacement list is empty, all existing selectors are cleared, again yielding `path/page.html`. This is especially useful when the requirement is to remove every selector rather than selectively remove named values. Both forms are URI-manipulation expressions and are applied through HTL's expression language using the `@` option syntax. The HTL specification documents URI manipulation, including the `selectors` and `removeSelectors` options, at the HTL Specification. Adobe's HTL documentation also describes the expression-language options used to safely construct and modify rendered URIs: Getting Started with HTL.