Adobe

AD0-E123
Adobe Experience Manager Sites Developer Professional

- **Up to Date products, reliable and verified.**
- **Questions and Answers in PDF Format.**

Full Version Features:

- **90 Days Free Updates**
- **30 Days Money Back Guarantee**
- **Instant Download Once Purchased**
- **24 Hours Live Chat Support**

For More Information:

https://www.testsexpert.com/

- **Product Version**

Latest Version: 6.0

Question: 1

A content author will be using live copies on AEM.
Which two factors must the content author now consider? (Choose two.)

A. When the inheritance Is re-enabled, the page is automatically synchronized with the source.
B. When (everting a canceled inheritance on a paragraph system, the order of components will be automatically restored from the blueprint.
C. If the component is marked as a container, the cancellation and suspend actions do not apply to Its child components.
D. Changes made locally to a component marked as a container will not be overwritten by the content of the blueprint on a rollout.

Answer: A, D

Explanation:
In Adobe Experience Manager (AEM), when dealing with live copies, a content author must be aware of various factors related to the synchronization of content between the source (blueprint) and the live copies. Option A is correct because when inheritance is re-enabled on a live copy page, synchronization with the blueprint page occurs, which may include automatic updates to the live copy page's content.
Option D is also correct because local changes made to a component marked as a container are protected during rollouts, meaning that these local changes will not be overwritten by content from the blueprint. The container concept in AEM allows for more granular control over which parts of the page are updated during synchronization.
For Option B, the term 'paragraph system' seems incorrect. It should likely refer to 'paragraph systems' as in the ParSys (Paragraph System), a component that allows authors to add components to a page. There is no automatic restoration of the order of components from the blueprint upon re-enabling canceled inheritance; instead, components are managed according to the rules set in the rollout configurations.
Option C is incorrect because the cancel and suspend actions can be applied to individual child components within a container. When inheritance is canceled for a container, the inheritance status of child components within that container can be controlled individually.

Question: 2

A developer wants to be able to execute the following query:

```
SELECT
*F
ROM [ntbase] AS s
WHERE
s.status ='STARTED'
```

Which two options are mandatory additions to the Index? (Choose two.)

A. ntbase index rule
B. status properly with property Index = false
C. ntbase aggregate
D. status property with propertylndex = true

Answer: A, D

Explanation:
When creating a custom query in AEM's JCR (Java Content Repository), the query's performance is highly dependent on the indexing configuration. For the given query that selects all nodes with a 'status' property equal to 'STARTED' from the 'nt:base' node type, the index must be set up correctly:
Option A, "nt:base index rule", is correct. The index rule for 'nt:base' must be added to define which properties of nodes of this type are indexed.
Option D, "status property with propertyIndex = true", is also correct. This index ensures that queries filtering on the 'status' property are executed efficiently. The property index should be set to true, which means that this property is indexed and the query will use this index to filter the results.
Option B is incorrect because setting the property index to false would mean that the property is not indexed, making the query less efficient since the repository would need to scan each node to find matches.
Option C, "nt:base aggregate", is not mandatory. Aggregates are used to include properties of related nodes in the index, but they are not required for a simple property match like the one in the given query.

Question: 3

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. htmI'search=searchparam
B. /myproduct/myrecipe. html?search- s&ordet=asc&brand=ad
C. /myproduct/myrecipe. html?brand=mybrand

Answer: C

Explanation:
Given the dispatcher configuration snippet provided in the image, we can understand the URL patterns

that will be ignored (not cached) by the dispatcher. The configuration under /ignoreUrlParams shows patterns to match query parameters in the URLs:
/0001 { /glob "*" /type "deny" } means that by default, all query parameters are ignored (not cached).
/0002 { /glob "search" /type "allow" } specifically allows caching for URLs with the 'search' parameter.
/0003 { /glob "order" /type "allow" } specifically allows caching for URLs with the 'order' parameter.
Based on this, let's evaluate the options:
A) /myproduct/myrecipe.html?search=searchparam — This URL will not be cached because the 'search' parameter is allowed, but the actual value 'searchparam' does not match any allow pattern. B) /myproduct/myrecipe.html?search=s&order=asc&brand=ad — This URL will not be cached because although 'search' and 'order' parameters are allowed, the 'brand' parameter is not allowed according to the configuration. C. /myproduct/myrecipe.html?brand=mybrand — This URL will be cached because there are no allowed parameters, so the default deny does not apply, and the page can be cached without considering the 'brand' parameter.
Therefore, the page that will be cached on the dispatcher is the one in option C, as it does not contain any of the explicitly allowed query parameters ('search' or 'order'), and all other parameters are ignored by default.

Question: 4

A developer wants to see debug logs for SAML. Which logger must be created for this purpose?

A. com.adobe.granlte.debug.auth.saml
B. com.adobe.sling.auth.saml
C. com. adobe, granite.aulh.saml

Answer: C

Explanation:
The correct logger to be created for debugging SAML (Security Assertion Markup Language) within AEM (Adobe Experience Manager) is the one pertaining to the Granite platform, which is part of AEM's internal framework dealing with authentication. The Granite framework provides services that include user, group, and permission management, and is integral to AEM's SAML authentication process. Thus, the logger com.adobe.granite.auth.saml is the appropriate logger that should be created and configured to capture and display debug logs related to SAML authentication processes.

Question: 5

A developer has to enable the indexing of multiple properties asynchronously. Which type of index would the developer use?

A. Property
B. Lucene
C. Ordered

Answer: B

Explanation:
In AEM, Lucene is often used for indexing due to its powerful full-text search capabilities, and it also supports asynchronous indexing. Asynchronous indexing allows the system to handle other tasks while indexing is being processed in the background, which can improve performance and scalability. This is especially useful when multiple properties need to be indexed without impacting the immediate response times of the system. Therefore, when a developer needs to enable the indexing of multiple properties asynchronously, a Lucene index is the most suitable choice.

For More Information – **Visit link below:**

https://www.testsexpert.com/

16$ Discount Coupon: 9M2GK4NW

Features:

- Money Back Guarantee……..……...…. 

- 100% Course Coverage…… 

- 90 Days Free Updates…… 

- Instant Email Delivery after Order……