Free Questions for AD0-E722 by actualtestdumps

Shared by Shaffer on 24-05-2024

Question 1

Question Type: MultipleChoice

An Adobe Commerce Architect is planning to create a new action that will add gift registry items to the customer's quote. What should the Architect do to guarantee that private content blocks are updated?

Options:

A- Mark the controller by setting no-cache HTTP headers

B- Invalidate the status of gift registry indexers

C- Specify a new action in a sections.xml configuration file

Answer:

C

Explanation:

Private content blocks are sections of the page that are specific to each customer and are not cached by the server. To update these blocks when a customer performs an action, such as adding a gift registry item to the quote, the Adobe Commerce Architect needs to specify the new action in a sections.xml configuration file. This file defines which blocks need to be updated for each action and how often they should be updated. By doing this, the Architect can ensure that the private content blocks are refreshed with the latest data from the server.Reference:

Private content | Magento 2 Developer Documentation

Configure private content | Magento 2 Developer Documentation

Question 2

Question Type: MultipleChoice

An Adobe Commerce Architect is investigating a case where some EAV product attributes are no longer updated.

* The catalog is composed of 20.000 products with 100 attributes each.

* The product updates are run by recurring Adobe commerce imports that happen multiple times a day.

* The Architect finds an error in the logs that indicates an integrity constraint while trying to insert row with id 2147483647.

What is causing this error?

Options:

A- Magento framework uses INSERT on DUPLICATE, which leads to reaching the max limit of the increment of the column.

B- Integrity constraints were dropped after upgrading to the latest version, and the integrity checks were missed.

C- EAV attribute import uses REPLACE, which leads to reaching the max limit of the increment of the column

Answer:

C

Explanation:

EAV attribute import uses the REPLACE statement, which deletes and inserts a new row with the same primary key value. This causes the auto-increment column to increase by one for each row, even if the row already exists. If the auto-increment column reaches its maximum value, which is 2147483647 for a signed INT, then any further REPLACE statement will fail with an integrity constraint violation error.Reference:

EAV and extension attributes | Magento 2 Developer Documentation

GitHub - techdivision/import-attribute: This library provides the functionality for the Magento 2 import of EAV attributes

Data integrity in JSON (B) when replacing EAV - Stack Overflow

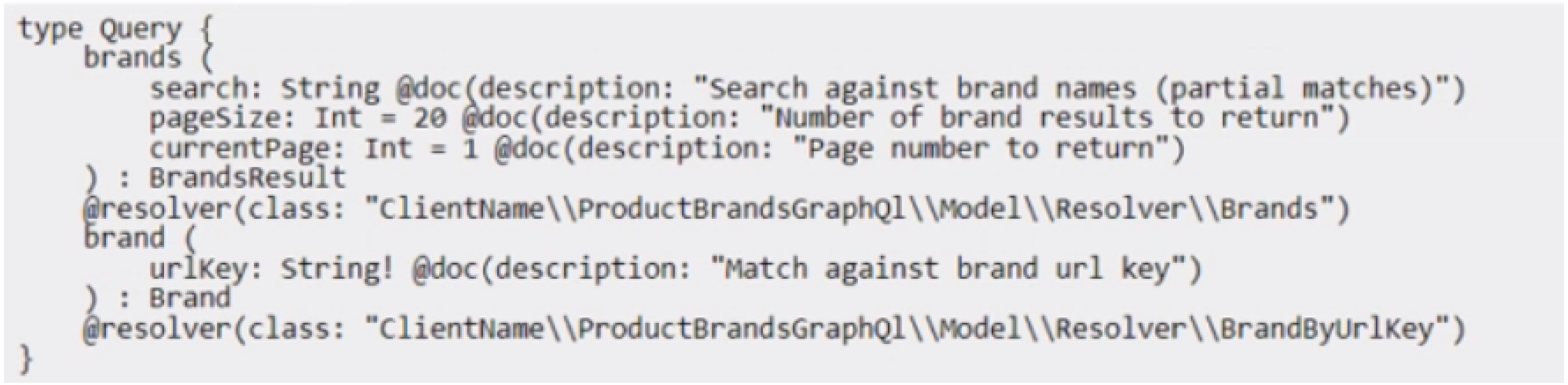
Question 3

Question Type: **MultipleChoice**

An existing Adobe Commerce website is moving to a headless implementation.

The existing website features an "All Brands" page, as well as individual pages for each brand. All brand-related pages are cached in Varnish using tags in the same manner as products and categories.

Two new GraphQL queries have been created to make this information available to the frontend for the new headless implementation:



```
type Query {
    brands (
        search: String @doc(description: "Search against brand names (partial matches)")
        pageSize: Int = 20 @doc(description: "Number of brand results to return")
        currentPage: Int = 1 @doc(description: "Page number to return")
    ) : BrandsResult
    @resolver(class: "ClientName\\ProductBrandsGraphQl\\Model\\Resolver\\Brands")
    brand (
        urlKey: String! @doc(description: "Match against brand url key")
    ) : Brand
    @resolver(class: "ClientName\\ProductBrandsGraphQl\\Model\\Resolver\\BrandByUrlKey")
}
```

During testing, the queries sometimes return out-of-date information. How should this problem be solved while maintaining performance?

Options:

A- Specify a @cacgecacheable(cacheable: false) directive for each GraphQL query, making sure that the data returned is not cached, and is up to date

B- Specify a $cache(cacheidentity: Path\\To\\identityclass) directive for each GraphQL query, corresponding to a class that adds cache tags for relevant brands and associated products

C- Each GraphQL query's resolver class should inject \Magento\GraphQlcache\Model\cacheableQuery and call setcachevalidity(true) on it as part of the resolver's resolve function.

Answer:

B

Explanation:

This solution ensures that the data returned by the GraphQL queries is up to date, while also maintaining performance. By specifying a $cache(cacheidentity: Path\To\identityclass) directive for each GraphQL query, the relevant brands and associated products will be added as cache tags.

Question 4

Question Type: **MultipleChoice**

An Adobe Commerce Architect needs to customize the workflow of a monthly installments payment extension. The extension is from a partner who is contracted with the default website Payment Service Provider (PSP), which has its own legacy extension (a module using deprecated payment method).

The installment payment partner manages only initializing a payment, and then hands the capture to be executed by the PSP Once the amount is successfully captured, the PSP notifies the website through a webhook. The goal of the webhook is only to create an "invoice" and save the "capture information" to be used later for refund requests through the PSP itself.

The Architect needs the most simple solution to capture the requested behavior.

Which solution should the Architect implement?

Options:

A- Add a plugin before the $invoice->capture() and change Its input to prevent the call of the $Payment->capture()

B- Change the can_capture attribute for the payment method under config.xml to be <can_capture>0</can_capture>

C- Declare a capture Command with type Magento\Payment\Gateway\Command\NullCommand for the payment method CommandPool in di.xml

Answer:

C

Explanation:

Option C is the correct solution because declaring a capture command with type Magento\Payment\Gateway\Command\NullCommand for the payment method command pool in di.xml will prevent the default capture logic from being executed. The NullCommand class is a dummy implementation of the CommandInterface that does nothing.This way, the payment capture will be handled by the PSP webhook, and the invoice will be created accordingly12

Option A is not a correct solution because adding a plugin before the $invoice->capture() and changing its input to prevent the call of the $payment->capture() will require modifying the core Magento code, which is not recommended.Moreover, this solution will affect all payment methods that use the invoice capture logic, not just the monthly installments payment extension3

Option B is not a correct solution because changing the can_capture attribute for the payment method under config.xml to be <can_capture>0</can_capture> will disable the capture functionality for the payment method entirely.This means that the invoice cannot be created or captured, even by the PSP webhook4

1: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/payments-integrations/payment-gateway/gateway-command.html?lang=en2: https://github.com/magento/magento2/blob/2.4-develop/app/code/Magento/Payment/Gateway/Command/NullCommand.php3: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/customization/best-practices.html?lang=en#do-not-modify-core-code4: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/payments-integrations/payment-gateway/payment-gateway-configuration.html?lang=en#payment-method-configuration

Question 5

Question Type: **MultipleChoice**

A merchant is using a unified website that supports native Adobe Commerce B2B and B2C with a single store view.

The merchant's objective is to display the B2B account features, such as negotiable quotes and credit limits, in the header of the site on every page for logged-in users who belong to a B2B company account.

Each B2B company possesses its unique shared catalog and customer group, while numerous customer groups for non-B2B customers undergo changes. The merchant insists that this association should not be linked to customer groups.

Which two solutions should the Architect recommend for consideration, taking into account public data and caching? (Choose two.)

Options:

A- Create a Virtual Type that switches the theme when a user is part of a B2B company so the output can be modified accordingly in the alternate theme.

B- Create a new HTTP Context variable to allow for separate public content to be cached for users in B2B companies where the output can be modified accordingly.

C- Set whether the current user is part of a B2B company in the customer session and use that data directly to modify the output accordingly.

D- Create a new custom condition for customer segments that allow for choosing whether a user is part of a B2B company and then use this segment to modify the output accordingly.

E- Check if the current user is part of a B2B company within a block class and modify the output accordingly.

Answer:

B, D

Explanation:

Option B is a valid solution because creating a new HTTP Context variable can allow for differentiating the public content cache for users who belong to a B2B company account.The HTTP Context variable can be used to modify the output of the header block accordingly, without affecting the performance or scalability of the site1

Option D is also a valid solution because creating a new custom condition for customer segments can enable targeting users who are part of a B2B company account. The customer segment can be used to modify the output of the header block accordingly, using layout updates or dynamic blocks.This solution can also leverage the existing customer segment functionality and avoid custom coding2

Option A is not a valid solution because switching the theme based on a virtual type can cause performance issues and increase the complexity of the site maintenance.Moreover, switching the theme can affect the entire site appearance, not just the header block3

Option C is not a valid solution because using the customer session data directly to modify the output of the header block can prevent the public content cache from working properly.The customer session data is private and cannot be cached, so this solution can negatively impact the performance and scalability of the site4

Option E is not a valid solution because checking if the current user is part of a B2B company within a block class can also prevent the public content cache from working properly.The block class logic is executed on every request, so this solution can negatively impact the performance and scalability of the site5

1: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/architecture/starter-architecture.html?lang=en#http-context2: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/marketing/customer-segments.html?lang=en3: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/design/themes.html?lang=en4: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/architecture/starter-architecture.html?lang=en#private-content5: https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/architecture/starter-architecture.html?lang=en#public-content

Question 6

Question Type: MultipleChoice

A client has multiple warehouses where orders can be fulfilled. The cost of shipping goods from each warehouse varies by day, due to the number of workers available. The Architect needs to make sure that when an order is shipped, it is shipped from the lowest cost warehouse that is open.

How should this functionality be implemented?

Options:

A- Create a new class as a preference for Magento\inventoryShipping\piugin\Sales\shipment\AssignSourceCodeToShipmentPlugin to set the lowest-cost warehouse on a shipment.

B- Create a new class implementing Magento\invtntorysourceSelectionApi\Modei\sourceSelectioninterfacece. which returns open warehouses sorted by cost.

C- Create an after plugin On Hagento\InventoryDistanceBasedSourceSelection\Hodel\Algorithms\DistanceBasedAlgorithto sort to Warehouse sources by cost

Answer:

B

Explanation:

According to the Adobe Commerce documentation, the Source Selection Interface is the main interface for implementing custom source selection algorithms. The interface defines a method called execute(), which takes a list of items to be shipped and a stock ID as parameters, and returns a SourceSelectionResultInterface object, which contains the recommended sources and quantities for each item. The Architect can create a new class that implements this interface and provides the logic for finding the lowest-cost warehouse that is open for each item. The Architect can then register the new class as an option for the source selection algorithm in the di.xml file of the custom module.

Source Selection Algorithm | Adobe Commerce Developer Guide

Source Selection Interface | Adobe Commerce Developer Guide

Question 7

Question Type: **MultipleChoice**

An Adobe Commerce Architect creates a stopword for the Italian locale named stopwordsjtJT.csv and changes the stopword directory to the following: /app/code/Custovendor/Elasticsearch/etc/stopwords/

What is the correct approach to change the stopwords directory inside the custom module?

Options:

A- Add stopwords to the stopwordsDirectory and CustomerVendor_Elasticsearch to the stopword sModule parameter Of the \Magento\Elasticsearch\SearchAdapter\Query\Preprocessor\Stopwords ClflSS Via di.xml

B- Add a new ClaSS implementing \Magento\Framework\Setup\Patch\PatchInterface to modify the default Value Of elasticsearch\customer\stopwordspath in core.conf ig_data table.

C- Add stopwords to the stopwordsDirectory parameter of the\Hagento\Elasticsearch\Model\Adapter\Document\DirectoryBuilder ClaSS Via stopwords/it.xml and Adobe Commerce will automatically detect the current module.

Answer:

A

Explanation:

According to the Adobe Commerce documentation, the correct approach to change the stopwords directory inside a custom module is to use dependency injection to override the default values of the stopwordsDirectory and stopwordsModule parameters of the \Magento\Elasticsearch\SearchAdapter\Query\Preprocessor\Stopwords class. The stopwordsDirectory parameter specifies the relative path of the stopwords directory from the module directory, while the stopwordsModule parameter specifies the name of the module that contains the stopwords directory. By adding these parameters to the di.xml file of the custom module, the Architect can change the location of the stopwords files without modifying the core code or database.

To change the directory from your module

Configure Elasticsearch stopwords

Question 8

Question Type: MultipleChoice

In a custom module, an Architect wants to define a new xml configuration file. The module should be able to read all the xml configuration files declared in the system, merge them together, and use their values in PHP class.

Which two steps should the Architect make to meet this requirement? (Choose two.)

Options:

A- Inject a 'reader' dependency for 'Magento\Framework\Config\Data' in di.xml

B- Write a plugin for \Magento\Framework\Config\Data::get() and read the custom xml files

C- Create a Data class that implements '\Magento\Framework\Config\Data'

D- Append the custom xml file name in 'Magento\Config\Model\Config\Structure\Reader' in di.xml

E- Make a Reader class that implements '\Magento\Framework\Config\Reader\Filesystem'

Answer:

C, E

Explanation:

According to the Adobe Commerce documentation, to create a new xml configuration file, the Architect needs to create a Data class and a Reader class for the custom module. The Data class is responsible for storing and retrieving the configuration data from the cache or the Reader class. The Data class should implement the ''\Magento\Framework\Config\Data'' interface or extend the ''\Magento\Framework\Config\Data'' class. The Reader class is responsible for reading and validating the xml configuration files from the file system. The Reader class should implement the '\Magento\Framework\Config\Reader\Filesystem' interface or extend the '\Magento\Framework\Config\Reader\Filesystem' class. The Architect also needs to declare the Data class and the Reader class in the di.xml file of the custom module, and specify the name of the xml configuration file, the converter class, and the schema locator class for the Reader class.

Configuration types | Adobe Commerce - Experience League

Create configuration types | Adobe Commerce - Experience League