Practice Exam Questions

Adobe
Certified Professional

AD0-E718

Adobe Commerce Architect Master

EXAMKILLER

Help Pass Your Exam At First Try

Adobe

Exam AD0-E718

Adobe Commerce Architect Master

Version: 4.1

[Total Questions: 50]

Question No : 1

A client is migrating to Adobe Commerce Cloud and has approximately 800 existing redirects that must be implemented. The number of redirects cannot be reduced because all redirects are specific, and do not match any pattern.

How should the redirects be configured to ensure performance?

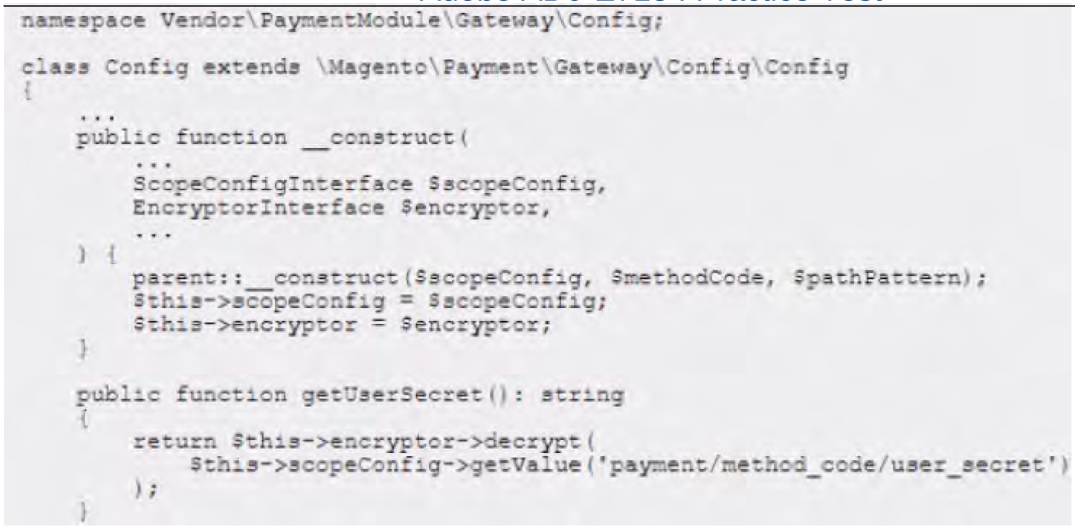
A. Use VCL snippets to offload the redirect to Fastly.
B. Add each redirect in the .magento/routes.yaml file.
C. Add each redirect as a URL rewrite via the admin Ul.

Answer: A

Explanation:
The best option for configuring the redirects is to use VCL snippets to offload the redirects to Fastly. This is a Content Delivery Network (CDN) that can handle large numbers of requests quickly and efficiently, ensuring that your redirects will be processed quickly and reliably. Furthermore, VCL snippets are easy to set up and can be reused for other redirects, making them an efficient and cost-effective solution for managing large numbers of redirects.
For Adobe Commerce on cloud infrastructure projects, configuring numerous non-regex redirects and rewrites in the routes.yaml file can cause performance issues. If your routes.yaml file is 32 KB or larger, offload your non-regex redirects and rewrites to Fastly. See Offload non-regex redirects to Fastly instead of Nginx (routes) in the Adobe Commerce Help Center**1**. References:
https://experienceleague.adobe.com/docs/commerce-cloud-service/user-guide/configure/routes/redirects.html?lang=en**21**

Question No : 2

While reviewing a newly developed pull request that refactors multiple custom payment methods, the Architect notices multiple classes that depend on \Magento\Framework\Encryption\EncryptorInterf ace to decrypt credentials for sensitive data. The code that is commonly repeated is as follows:

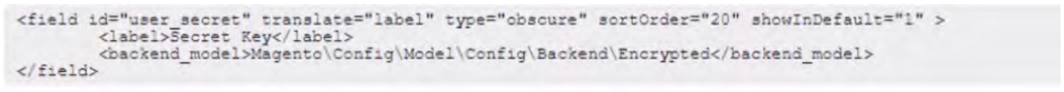


```
namespace Vendor\PaymentModule\Gateway\Config;

class Config extends \Magento\Payment\Gateway\Config\Config
{
    ...
    public function __construct(
        ...
        ScopeConfigInterface $scopeConfig,
        EncryptorInterface $encryptor,
        ...
    ) {
        parent::__construct($scopeConfig, $methodCode, $pathPattern);
        $this->scopeConfig = $scopeConfig;
        $this->encryptor = $encryptor;
    }

    public function getUserSecret(): string
    {
        return $this->encryptor->decrypt(
            $this->scopeConfig->getValue('payment/method_code/user_secret')
        );
    }
```

In each module, the user_secret config is declared as follows:



```
<field id="user_secret" translate="label" type="obscure" sortOrder="20" showInDefault="1" >
        <label>Secret Key</label>
        <backend_model>Magento\Config\Model\Config\Backend\Encrypted</backend_model>
</field>
```

The Architect needs to recommend an optimal solution to avoid redundant dependency and duplicate code among the methods. Which solution should the Architect recommend?

A. Replace all Vendor\PaymentModule\Gateway\Config\Config Classes With virtualTyp- Of Magento\Payxer.t\Gateway\Conflg\Config and Set <user_secret backend_Model="Magento\Config\Model\Config\Backend\Encrypted" /> under ccnfig.xml
B. Add a plugin after the getvalue method of $sccpeConfig, remove the $encryptor from dependency and use it in the plugin to decrypt the value if the config name is 'user.secret?
C. Create a common config service class vendor\Payment\Gateway\config\Config under Vendor.Payment and use it as a parent class for all of the Vender \EaymentModule\Gateway\Config\Config Classes and remove $sccpeConfig and $encryptor dependencies
Explanation: To avoid redundant dependency and duplicate code among the methods, you need to use the following solution:

- Create a standard controller route and mapping the internal URLs (such as news/article/view/id/1) to rewrites that are generated on save and then stored in the URL rewrites table. This solution will leverage the built-in URL rewrite functionality of Magento to handle the news requests. You can create a standard controller route that handles the internal URLs and loads the news article by its ID. Then, you can generate URL rewrites for each news article based on its date and URL key and store them in the URL rewrites table. This way, you can avoid using a custom router or a plugin and reduce the code complexity.

Reference:

: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/routing.html : https://devdocs.magento.com/guides/v2.4/extension-dev-guide/url-rewrite.html

Question No : 3

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.)

A. Write a plugin for \Magento\Framework\Config\Data::get() and read the custom xml files
B. Append the custom xml file name in "Magento\Config\Model\Config\Structure\Reader" in di.xml
C. Create a Data class that implements "\Magento\Framework\Config\Data'
D. Inject a "reader" dependency for "Magento\Framework\Config\Data" in di.xml
E. Make a Reader class that implements "\Magento\Framework\Config\Reader\Filesystem"

Answer: C,E

Explanation: Based on web searches, it seems that Magento uses different classes and interfaces to interact with configuration files, such as Data, Reader, and Converter**12**.
According to the documentation**1**, Data is a class that provides access to configuration data using a scope. Reader is an interface that reads configuration data from XML files. Converter is an interface that converts XML data into an array representation.
Based on these definitions, I would say that two possible steps that the Architect should make to meet the requirement are:

- C. Create a Data class that implements “\Magento\Framework\Config\Data”
- E. Make a Reader class that implements “\Magento\Framework\Config\Reader\Filesystem”

These steps would allow the custom module to read all the XML configuration files declared in the system, merge them together, and use their values in PHP class.

The Architect should make two steps to meet this requirement: C) Create a Data class that implements “\Magento\Framework\Config\Data”. This class will be responsible for reading and merging the custom xml configuration files and providing access to their values. The Data class should extend \Magento\Framework\Config\Data and use the constructor to inject the Reader class and the cache type. E) Make a Reader class that implements “\Magento\Framework\Config\Reader\Filesystem”. This class will be responsible for loading and validating the custom xml configuration files from different modules. The Reader class should extend \Magento\Framework\Config\Reader\Filesystem and use the constructor to specify the file name, schema file, and validation state of the custom xml configuration files. Option A is incorrect because writing a plugin for \Magento\Framework\Config\Data::get() will not define a new xml configuration file, but rather modify the existing one. Option B is incorrect because appending the custom xml file name in “Magento\Config\Model\Config\Structure\Reader” in di.xml will not define a new xml configuration file, but rather add it to the system configuration structure. Option D is

incorrect because injecting a “reader” dependency for “Magento\Framework\Config\Data” in di.xml will not define a new xml configuration file, but rather use an existing one.
References: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/build/XSD-XML-validation.html

Question No : 4

An Adobe Commerce Architect notices that the product price index takes too long to execute. The store is configured with multiple websites and dozens of customer groups.

Which two ways can the Architect shorten the full price index execution time? (Choose two.)

A. Enable price index customer group merging for products without tier prices
B. Set Customer Share Customer Accounts Option to Global
C. Edit customer groups to exclude websites that they are not using
D. Set MaGE_INDEXER_THREADS_COUNT environment variable to enable parallel mode
E. Move catalog price_index indexer to another custom indexer group

Answer: A,D

Explanation: The two best ways the Architect can shorten the full price index execution time are Option A. Enable price index customer group merging for products without tier prices, and Option D. Set MaGEINDEXERTHREADS_COUNT environment variable to enable parallel mode. Enabling customer group merging will help reduce the number of customer groups that need to be processed, while setting the environment variable will allow the indexer to use multiple threads and run in parallel mode, thus reducing the overall execution time.
Enabling price index customer group merging allows Magento to merge the price index rows for products that have the same price for all customer groups. This reduces the number of rows in the price index table and improves the performance of the indexer.
Setting the MaGE_INDEXER_THREADS_COUNT environment variable allows Magento to run the indexer in parallel mode, which splits the index into multiple batches and processes them simultaneously. This reduces the execution time of the indexer. References: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/indexing.html#customer-group-merging https://devdocs.magento.com/guides/v2.4/extension-dev-guide/indexing.html#parallel-mode

Question No : 5

An Architect agrees to improve company coding standards and discourage using Helper classes in the code by introducing a new check with PHPCS.

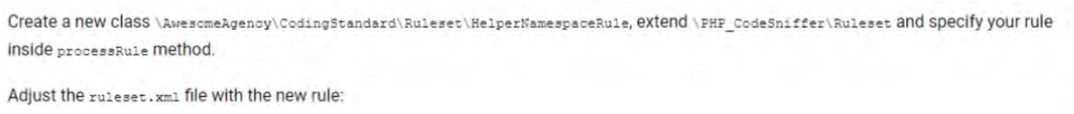
The Architect creates the following:

• A new composer package under the AwesomeAgency\CodingStandard\ namespace

• The ruleset. xml file extending the Magento 2 Coding Standard

What should the Architect do to implement the new code rule?

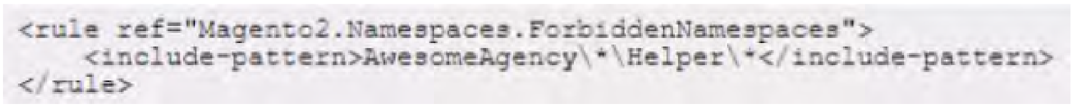
A)



Create a new class `\AwesomeAgency\CodingStandard\Ruleset\HelperNamespaceRule`, extend `\PHP_CodeSniffer\Ruleset` and specify your rule inside `processRule` method.

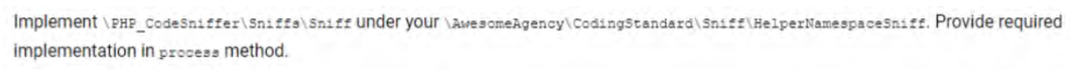
Adjust the `ruleset.xml` file with the new rule:

B)



```
<rule ref="Magento2.Namespaces.ForbiddenNamespaces">
    <include-pattern>AwesomeAgency\*\Helper\*</include-pattern>
</rule>
```

C)



Implement `\PHP_CodeSniffer\Sniffs\Sniff` under your `\AwesomeAgency\CodingStandard\Sniff\HelperNamespaceSniff`. Provide required implementation in `process` method.

A. Option A
B. Option B
C. Option C

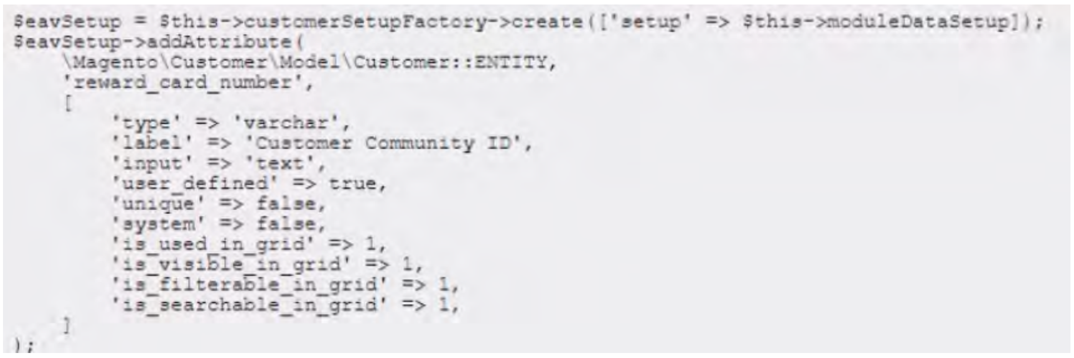
Answer: C

Explanation: To implement the new code rule, the Architect should create a new class that extends the \PHP_CodeSniffer\Sniffs\Sniff interface and implements the register() and process() methods. The register() method should return an array of tokens that the rule applies to, such as T_STRING for class names. The process() method should contain the logic to check if the class name contains Helper and report an error if so. The Architect should also add a reference to the new class in the ruleset.xml file under the <rule> element with a ref attribute. This will enable PHPCS to use the new rule when checking the code.

Question No : 6

A merchant is utilizing an out-of-the-box Adobe Commerce application and asks to add a new reward card functionality for customers. During the code review, the Adobe Commerce

Architect notices the reward_card_number attribute setup created for this functionality is causing the customer attribute to be unavailable in the My account/My rewards page template.



```
$eavSetup = $this->customerSetupFactory->create(['setup' => $this->moduleDataSetup]);
$eavSetup->addAttribute(
    \Magento\Customer\Model\Customer::ENTITY,
    'reward_card_number',
    [
        'type' => 'varchar',
        'label' => 'Customer Community ID',
        'input' => 'text',
        'user_defined' => true,
        'unique' => false,
        'system' => false,
        'is_used_in_grid' => 1,
        'is_visible_in_grid' => 1,
        'is_filterable_in_grid' => 1,
        'is_searchable_in_grid' => 1,
    ]
);
```

What should be added to set the customer attribute correctly?

A. scope property should be added with a value of global
B. group property should be added with a value of 1
C. system property should be added with a value of true

Answer: B

Explanation:

The group property specifies the attribute group ID that the customer attribute belongs to. By setting the group property to 1, the reward_card_number attribute will be added to the default attribute group and will be available in the My account/My rewards page template.
References: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/attributes.html#customer-eav-attribute

Question No : 7

A company wants to build an Adobe Commerce website to sell their products to customers in their country. The taxes in their country are highly complex and require customization to Adobe Commerce. An Architect is trying to solve this problem by creating a custom tax calculator that will handle the calculation of taxes for all orders in Adobe Commerce.

How should the Architect add the taxes for all orders?

A. Write a before plugin to \Magento\Quote\Model\QuoteManagement::placeOrder() and add the custom tax to the quote
B. Declare a new total collector in "etc/sales.xml" in a custom module
C. Add a new observer to the event 'sales_quote_collect_totals_before" and add the custom tax to the quote

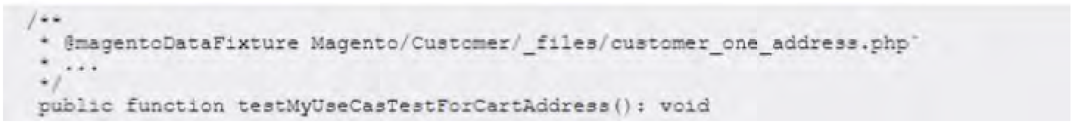
Answer: B

Explanation: you can create tax rules in Magento 2 by going to Stores > Taxes > Tax Rules and choosing or adding tax rates. However, this may not be enough for complex tax scenarios that require customization.
https://amasty.com/knowledge-base/how-to-configure-tax-calculation-in-magento-2.html
To add a new total to the order, the Architect should declare a new total collector in the “etc/sales.xml” file of a custom module. This file defines the order of calculation and rendering of totals. The Architect should also implement a model class that extends \Magento\Quote\Model\Quote\Address\Total\AbstractTotal and overrides the collect() and fetch() methods to handle the logic of adding the custom tax to the quote and order.
References:
https://devdocs.magento.com/guides/v2.4/howdoi/checkout/checkout_new_total.html

Question No : 8

An Adobe Commerce Architect is reviewing api-functional test code. Some tests send errors to indicate that the customer address does not exist.

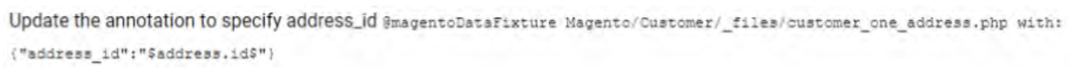
The test codes show the following:



```
/**
 * @magentoDataFixture Magento/Customer/_files/customer_one_address.php
 * ...
 */
public function testMyUseCasTestForCartAddress(): void
```

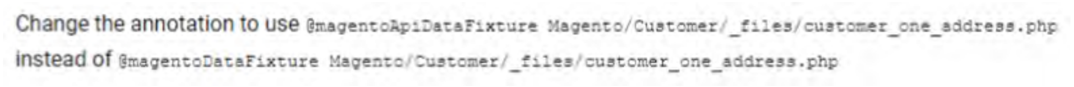
Which steps should the Architect take to fix the test errors?

A)



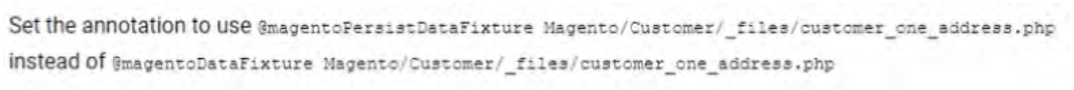
Update the annotation to specify address_id `@magentoDataFixture Magento/Customer/_files/customer_one_address.php with: {"address_id":"$address.id$"}`

B)



Change the annotation to use `@magentoApiDataFixture Magento/Customer/_files/customer_one_address.php` instead of `@magentoDataFixture Magento/Customer/_files/customer_one_address.php`

C)



Set the annotation to use `@magentoPersistDataFixture Magento/Customer/_files/customer_one_address.php` instead of `@magentoDataFixture Magento/Customer/_files/customer_one_address.php`

A. Option A
B. Option B

C. Option C

Answer: B

Explanation: The test errors are caused by using the wrong customer ID and address ID in the request. The correct customer ID and address ID should be obtained from the response of the previous request to create a customer and an address. The test code should use $this->customer->getId() and $this->address->getId() instead of hard-coded values. References: https://devdocs.magento.com/guides/v2.4/get-started/web-api-functional-testing.html

Question No : 9

An Architect needs to integrate an Adobe Commerce store with a new Shipping Carrier. Cart data is sent to the Shipping Carrier's API to retrieve the price and display to the customer. After the feature is implemented on the store, the API hits its quota and returns the error "Too many requests". The Shipping Carrier warns the store about sending too many requests with the same content to the API.

In the carrier model, what should the Architect change to fix the problem?

A. Implement _setCachedQuotes () and_getCachedQuotes() return the data if the request matches.
B. In _doShipmentRequest (), call canCollectRates() before sending request to the API
C. Override getResponse (), save the response to a variable, check if the response exists, then return.

Answer: A

Explanation: Implementing setCachedQuotes () andgetCachedQuotes() in the carrier model can allow the store to store the cart data in a cache, so that repeated requests with the same content can be retrieved from the cache instead of sending a new request to the API. This can reduce the number of requests and avoid hitting the quota limit. References: [1] https://docs.adobe.com/content/help/en/experience-manager-65/commerce/commerce-payment-shipping-modules/shipping/shipping-carrier-apis.html [2] https://docs.adobe.com/content/help/en/experience-manager-65/commerce/commerce-payment-shipping-modules/shipping/developing-shipping-carrier-integrations/shipping-carrier-model.html

Question No : 10

An Architect wants to create an Integration Test that does the following:

• Adds a product using a data fixture

• Executes $this->someLogic->execute($product) on the product

• Checks if the result is true.

Sthis->someLogic has the correct object assigned in the setup () method-Product creation and the tested logic must be executed in the context of two different store views with IDs of 3 and 4, which have been created and are available for the test.

How should the Architect meet these requirements?

A. Create one test class with one test method. Use the \Magento\testFramework\store\Executionstorecontext class once in the fixture and another time in the test.
B. Create two test Classes With one test method each. Use the @magentoExecuteInStoreContext 3 and @magentoExecuteInStoreContext 4 annotations on the class level.
C. Create one test class with two test methods. Use the @magentoStoreContext 3 annotation in one method and @magentoStoreContext 4 in the other one.

Answer: C
Explanation: The @magentoStoreContext annotation allows the test to run in the context of a specific store view. This annotation can be used on the method level to specify different store views for different test methods. This way, the product creation and the tested logic will be executed in the context of the same store view for each test method.
References:
https://devdocs.magento.com/guides/v2.4/test/integration/annotations/magento-store-context.html

Question No : 11

An Adobe Commerce system is configured to run in a multi-tier architecture that includes:

• A cache server with Varnish installed

• A backend web server with Adobe Commerce installed

• A database server with MySQL installed

When an Adobe Commerce Architect tries to clean the cache from the Store Admin by using the "Flush Magento Cache" in Cache Management, the Full Page Cache does not clear.

Which two steps should the Architect take to make the Full Page Cache work properly? (Choose two.)

A. Set the backend destination host to the frontend server's address in the Store Admin Stores > Configuration > Advanced > System > Full Page Cache > Varnish Configuration > Backend Host
B. Set the backend port destination to the frontend server's Varnish port in the Store Admin Stores > Configuration > Advanced > System > Full Page Cache > Varnish Configuration > Backend Port
C. Set the cache type to "Varnish Caching" in the Store Admin.
Stores > Configuration > Advanced > System > Full Page Cache > Caching Application
D. Use "Flush Cache Storage" instead of "Flush Magento Cache"
E. Set the cache destination host using magento CLI bin/magento setup:config:set --http-cache-hosts<cache_server>:<varrnish port>

Answer: C,E

Explanation: To use Varnish as the full page cache, the cache type must be set to “Varnish Caching” in the Store Admin. This will enable the “Flush Magento Cache” button to send a purge request to Varnish. Additionally, the cache destination host must be specified using the magento CLI command to tell Magento which Varnish servers to purge. References: https://devdocs.magento.com/guides/v2.4/config-guide/varnish/config-varnish.html

Question No : 12

An Adobe Commerce Architect notices that queue consumers close TCP connections too often on Adobe Commerce Cloud server leading to delays in processing messages.

The Architect needs to make sure that consumers do not terminate after processing available messages in the queue when CRON job is running these consumers.

How should the Architect meet this requirement?

A. Increase multiple_process limit to spawn more processes for each consumer.
B. Set CONSUMER_WAIT_FOR_MAX_MESSAGES variable true in deployment stage.
C. Change max_messages from 10,000 to 1,000 for CRON_CONSUMER_RUNNERvariable.

Answer: B

Explanation: The best way to meet this requirement is to set the

CONSUMERWAITFORMAXMESSAGES variable to true in the deployment stage. This variable will ensure that the consumer will not terminate when there are no more messages in the queue and will instead wait until a new message is available, preventing it from closing the connection prematurely. Additionally, the multiple_processes limit can be increased to spawn more processes for each consumer, which will help ensure that messages can be processed faster.
Setting CONSUMER_WAIT_FOR_MAX_MESSAGES variable true in deployment stage will make sure that consumers do not terminate after processing available messages in the queue when CRON job is running these consumers. This will prevent consumers from closing TCP connections too often and improve performance. See Start message queue consumers in the Adobe Commerce Help Center**1**. References:
https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/message-queues/consumers.html?lang=en**2**
https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/cli/start-message-queues.html**1**

Question No : 13

An Adobe Commerce Architect needs to log the result of a ServiceClass : : ge-Dara method execution after all plugins have executed. The method is public, and there are a few plugins declared for this method. Among those plugins are after and around types, and all have sortOrder specified.

Which solution should be used to meet this requirement?

A. Declare a new plugin with the sortOrder value higher than the highest declared plugin sortOrder and implement afterGetData method.
B. Declare a new plugin with the sortOrder value lower than the lowest declared plugin sortOrder and implement aroundGetData method
C. Declare a new plugin with the sortOrder value higher than the highest declared plugin sortOrder and implement aroundGetData method

Answer: C
Explanation: the **sortOrder** property from the plugin node declared in di.xml determines the plugin's prioritization when more than one plugin is observing the same method. The Magento\Framework\Interception\PluginListInterface which is implemented by Magento\Framework\Interception\PluginList\PluginList is responsible to define when to call the before, after, and around methods for each plugin.
Therefore, to log the result of a ServiceClass::getData method execution after all plugins have executed, the solution should be:
C. Declare a new plugin with the sortOrder value higher than the highest declared plugin

sortOrder and implement aroundGetData method.

https://devdocs.magento.com/guides/v2.3/extension-dev-guide/plugins.html

Question No : 14

An Adobe Commerce Architect gets a request to change existing payment gateway functionality by allowing voided transactions only for a certain range of paid amounts.

In the vendor module file etc/config.xml, payment method has an option can,_void set to 1.

How should this customization be done?

A. Extend Magento\Payment\Model\\Method\Adapter and reimplement method void. Use this new class as a new type of payment method facade configuration overriding virtualType type for adapter.
B. Declare a new plugin for class Magento\Payment\ Gateway\Config\ConfigValueHandler and using the afterHandle method, change the result for Subject can_void.
C. Add new handler with name can_void to virtualType based on typeMagento payment\Gateway\config\ValueHandlerPool In payment method facade configuration.

Answer: C

Explanation: The Architect should add a new handler with name can_void to virtualType based on type Magento\Payment\Gateway\Config\ValueHandlerPool in payment method facade configuration. This handler will be responsible for determining whether the payment method can void transactions or not, based on the custom logic of the paid amount range. The handler should implement \Magento\Payment\Gateway\Config\ValueHandlerInterface and override the handle() method to return true or false depending on the payment amount. Option A is incorrect because extending Magento\Payment\Model\Method\Adapter and reimplementing method void will not change the can_void option, but rather the logic of voiding transactions. Option B is incorrect because declaring a new plugin for class Magento\Payment\Gateway\Config\ConfigValueHandler and using the afterHandle method will affect all payment methods that use this class, not just the specific one that needs customization. References: https://devdocs.magento.com/guides/v2.4/payments-integrations/base-integration/integration-model.html

Question No : 15

An Architect is configuring the preload.keys for Redis on an Adobe Commerce on-premise instance.

The Architect discovers that the following cache keys are loaded on each frontend request: eav_entity_types,

GLOBAL_PLUGIN_LIST, DB_IS_UP_TO_DATE , SYSTEM_DEFAULT.

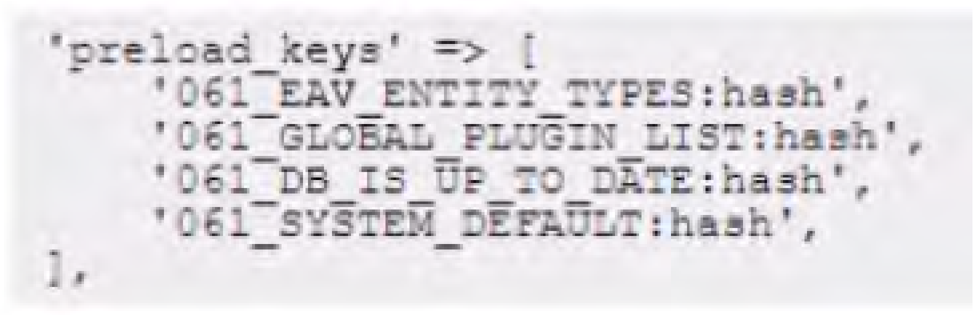
• The id_prefix of the frontend =>page_cache is set to 061_.

• The id_prefix of frontend => default: is not set.

• The Architect has enabled and configured Redis L2 caching.

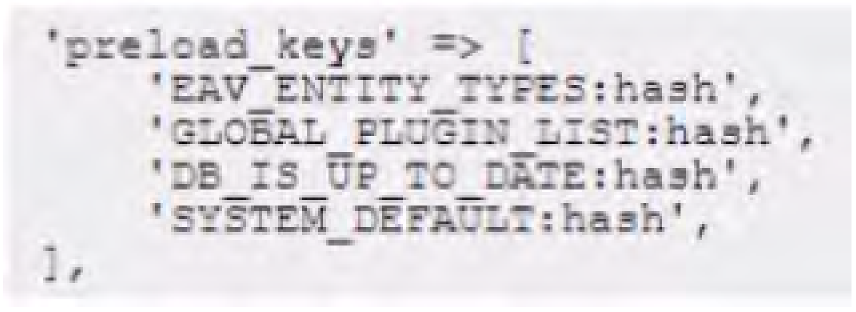
How should the preload.keys be configured?

A)



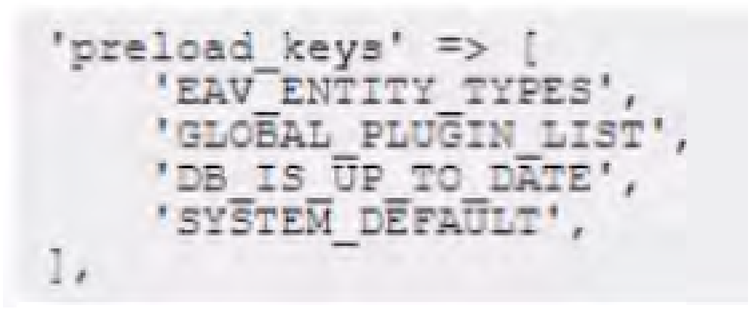
```
'preload_keys' => [
    '061_EAV_ENTITY_TYPES:hash',
    '061_GLOBAL_PLUGIN_LIST:hash',
    '061_DB_IS_UP_TO_DATE:hash',
    '061_SYSTEM_DEFAULT:hash',
],
```

B)



```
'preload_keys' => [
    'EAV_ENTITY_TYPES:hash',
    'GLOBAL_PLUGIN_LIST:hash',
    'DB_IS_UP_TO_DATE:hash',
    'SYSTEM_DEFAULT:hash',
],
```

C)



```
'preload_keys' => [
    'EAV_ENTITY_TYPES',
    'GLOBAL_PLUGIN_LIST',
    'DB_IS_UP_TO_DATE',
    'SYSTEM_DEFAULT',
],
```

D)