# Free Questions for AD0-E718 by certsinside

## Question 1

**Question Type: MultipleChoice**

An Adobe Commerce Architect needs to ensure zero downtime during the deployment process of Adobe Commerce on-premises.
Which two steps should the Architect follow? (Choose two.)

### Options:

**A-** Run bin/magento setup:upgrade --keep-generated to Upgrade database

**B-** Run bin/magento setup:upgrade ---dry-run=true to upgrade database

**C-** Rim bin/magento setup:upgrade --convert-old-scripts=true to Upgrade database

**D-** Enable config flag under developer/zere _down_time/enabled

**E-** Enable config flag under deployment/blue_ green/enabled

### Answer:

A, E

### Explanation:

To ensure zero downtime during the deployment process of Magento 2 on-premises, the Architect should follow two steps:

Run bin/magento setup:upgrade --keep-generated to upgrade database. This will skip the regeneration of static content and code files during the upgrade process, which can take a long time and cause downtime. The static content and code files should be generated separately before or after the upgrade process.

Enable config flag under deployment/blue_green/enabled. This will enable the blue-green deployment strategy, which creates a copy of the current production environment (blue) and deploys the new code to it (green). Then, it switches the traffic from the blue environment to the green environment without any downtime. This option can be enabled by adding a line like deployment/blue_green/enabled: true to the .magento.env.yaml file.

## Question 2

**Question Type: MultipleChoice**

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

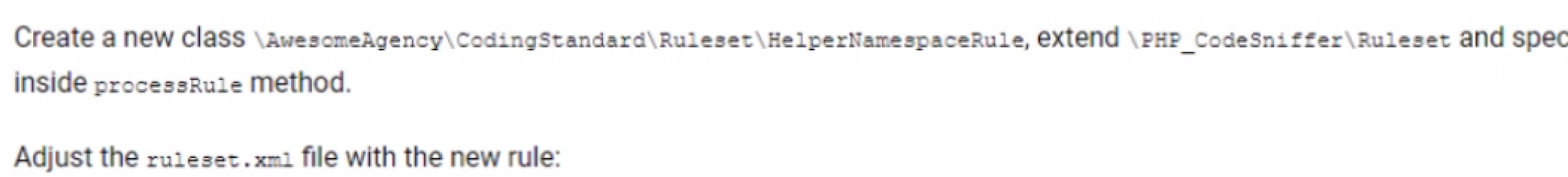
The Architect creates the following:

* 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)

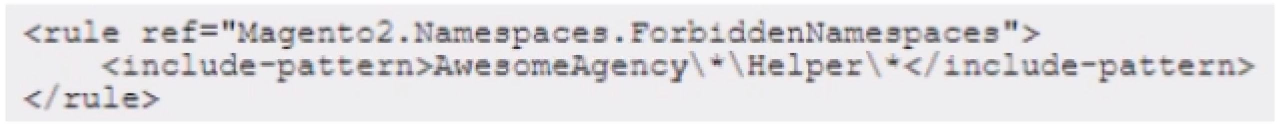


Create a new class `\AwesomeAgency\CodingStandard\Ruleset\HelperNamespaceRule`, extend `\PHP_CodeSniffer\Ruleset` and spec

inside `processRule` method.

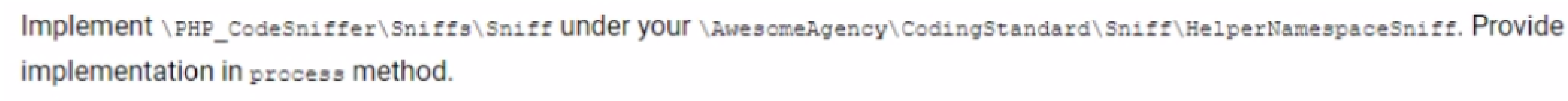
Adjust the `ruleset.xml` file with the new rule:

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

implementation in `process` method.

## Options:

**A-** Option A

**B-** Option B

**C-** Option C

### Answer:

C

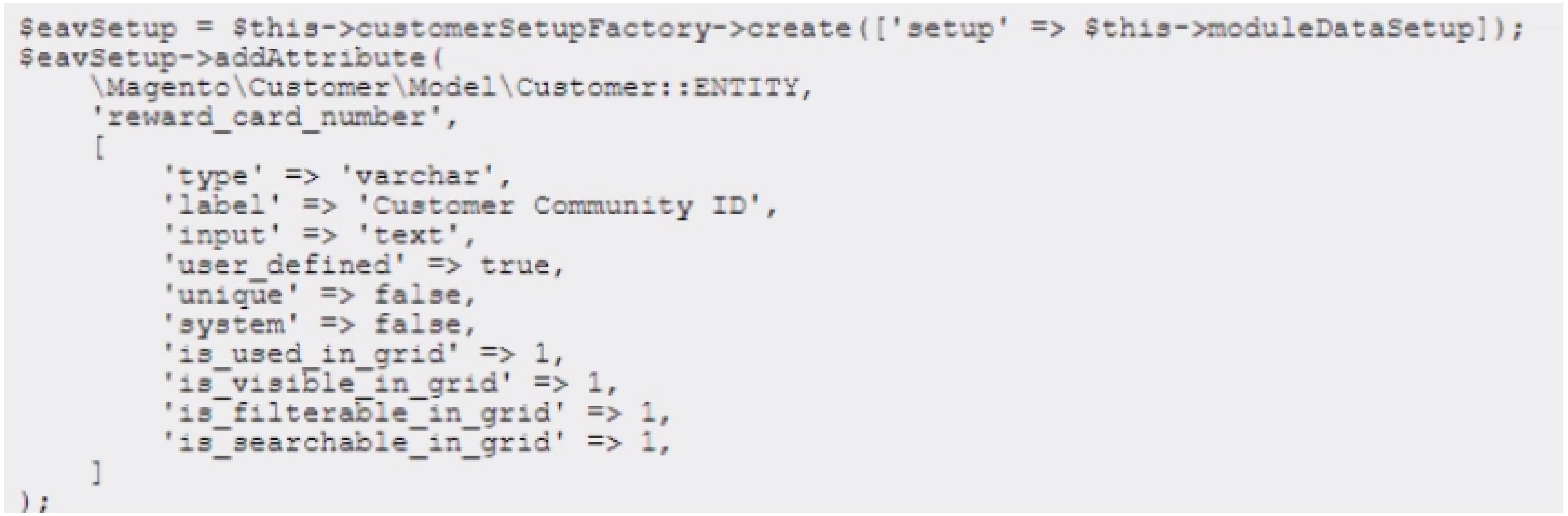
### Explanation:

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 3

**Question Type:** **MultipleChoice**

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?

## Options:

**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

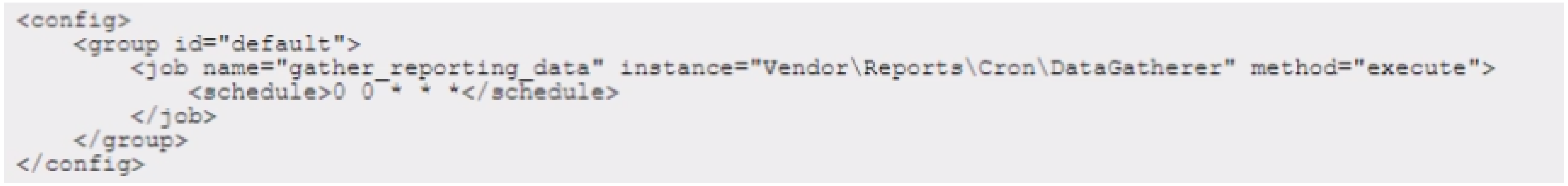
**Explanation:**

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. Reference: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/attributes.html#customer-eav-attribute

## Question 4

**Question Type: MultipleChoice**

A custom cron job has been added to an Adobe Commerce system to collect data for several reports. Its crontab. xml configuration is as follows:



```
<config>
    <group id="default">
        <job name="gather_reporting_data" instance="Vendor\Reports\Cron\DataGatherer" method="execute">
            <schedule>0 0 * * *</schedule>
        </job>
    </group>
</config>
```

The job is data intensive and runs for between 20 and 30 minutes each night.

Within a few days of deployment, it is noticed that the site's sitemap. xml file has not been updated since the new job was added.

What should be done to fix this issue?

## Options:

**A-** Break the data gathering job into a number of smaller jobs, so that each individual job runs for a maximum of 5 minutes.

**B-** Create a new cron group for the reporting job. Specifying <use_separate_process>1/use_separate_process>

**C-** Change the schedule of the sitemap_generate cron job to 30 o * * * so that it runs after the aacher_reporcmg_data job has completed.

## Answer:

B

## Explanation:

This will ensure that the reporting job runs in its own process, separate from other cron jobs, and will not interfere with the sitemapgenerate cron job. This will ensure that the sitemapgenerate cron job runs as soon as the reporting job is finished, ensuring that the sitemap.xml is always up to date.

The issue is caused by the reporting job blocking the default cron group from running other jobs. By creating a new cron group for the reporting job and specifying <use_separate_process>1/use_separate_process>, the reporting job will run in a separate PHP process and will not affect the default cron group. This way, the sitemap_generate cron job will run as scheduled. Reference: https://devdocs.magento.com/guides/v2.4/config-guide/cron/custom-cron-ref.html

## Question 5

**Question Type:** **MultipleChoice**

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.)

### Options:

**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. Reference:
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 6

**Question Type:** **MultipleChoice**

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?

## Options:

**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. Reference:

https://devdocs.magento.com/guides/v2.4/test/integration/annotations/magento-store-context.html

## Question 7

**Question Type:** MultipleChoice

An Adobe Commerce Architect creates a new functionality called Customs Fee, which adds a new total that applies to additional costs for handling customs clearance expenses. The extension allows specifying fee value for every website separately via the Adobe Commerce Configuration System.

The Architect plans to cover new functionality with integration tests. One test case needs to confirm if the total is calculated correctly on different websites.

How should the Architect make sure that test configuration data is added to test methods according to best practices?

### Options:

**A-** Override setuo () method, receive instance of \Magento\TestFramework\App\config, and specify value via setValue () method

**B-** Specify @magentoconfigFixture annotations for the test methods in PHPDoc

**C-** Create a fixture file to configure Adobe Commerce and specify it in test method PHPDoc using the @magentoconfigFixture annotation

## Answer:

C

## Explanation:

The best practice for adding test configuration data is to use a fixture file that contains the configuration values for different websites. The fixture file can be specified in the test method PHPDoc using the @magentoconfigFixture annotation. This way, the configuration data is isolated from the test code and can be reused for other tests. Reference:
https://devdocs.magento.com/guides/v2.4/test/integration/annotations/magento-config-fixture.html

## Question 8

**Question Type: MultipleChoice**

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`

## Options:

**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. Reference: https://devdocs.magento.com/guides/v2.4/get-started/web-api-functional-testing.html