DUMPSBOSS.

Adobe Commerce Developer Expert

Adobe AD0-E709

Version Demo

Total Demo Questions: 10

Total Premium Questions: 50

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

QUESTION NO: 1

An Adobe Commerce Developer is tasked with writing an importer for a custom entity. After the work is complete and deployed, they start receiving complaints from their client that the importer does not work and fails every time they use it.

The developer realizes that the client is importing a file which does not match the format required for the importer to process correctly.

What two features would the developer add to this importer to prevent this? (Choose two.)

A. Set up an example file and inject it into the ExampleFileProvider .

B. Set up a try/catch in the importer to display a warning the file is not in the correct format.

C. Set up validation by implementing a validateRow method in their importer.

D. Set $needColumnCheck = true; and add column names to the $validColumnNames class properties.

ANSWER: C D

QUESTION NO: 2

A customer is creating a new website, store and store view that will have a new category structure. How would an Adobe Commerce developer implement this?

A. All stores have unique category structures, select which root category to duplicate to the new store during creation.

B. Create a new root category in the admin and then select it when creating the website.

C. Create a new root category in the admin and then select it when creating the store.

ANSWER: A

QUESTION NO: 3

When paying by Bank Transfer, there is a requirement to send an email to customer service with payment details, after the order is placed successfully. Which two events can be used to send an email during the order placement process? (Choose two.)

A. sales_order_payment_pay

B. sales_model_service_auote_submit_before

C. sales_model_service_quote_submit_success

D. sales_order_place_after

ANSWER: A D

QUESTION NO: 4

An Adobe Commerce developer is working on an Acme_Exceptions module which is supposed to overwrite logic inside some of Magento native exceptions such as \Magento\Frame-work\Exception\NoSuchEntityException or \Mogento\Framwork\GraphQ1\Exception\GrephQIInputException , The module is open-source and will be available on packagist.org.

The build of the codebase of projects, including the module, will sometimes take place in docker containers with full access to filesystem. but then it is deployed to a read-only filesystem.

Which two approaches would the developer use to overwrite logic in those exceptions? (Choose two.)

A. 1. Create a version of those exceptions inside the module using the original namespaces and classes, e.g. \Magento\Framewrk\Exception\NoSuchEntityException .
2. Use composer's extra > nap node to copy those exceptions to their original destinations like vendor/Magento/framework/exception/ to replace original files.

B. 1. Create a version of those exceptions inside the module using new namespace, e.g. \Acme\Exceptions\Exception\NoSuchEntityException .

C. 1. Create a version of those exceptions inside the module using the original namespaces and Classes, e.g. \Magento\Framework\Exceptioin\NoSuchEntityException .
2. Use composer's scripts > post-install-and and scripts > post-update-cmd nodes to copy those exceptions to their original destinations like
vendor/Magento/framework/exception/ to replace original files.

D. 1. Create a version of Those exceptions inside the module using the original namespaces and classes, e.g. \Magento\Framework\Exception\NoSuchEntityException .
2. Create a data patch copying those exceptions to their original destinations like vendor/magento/ framework /except Ion/ to replace original files.

ANSWER: A B

QUESTION NO: 5

An Adobe Commerce developer has just finished creating a new custom entity, a block that extends \Mangento\Framework\View|Element\abstractBlock that lists all of the existing entities and a controller with the appropriate handle to display the block.

The developer now wants to implement cache on the block so that when one of the custom entities is saved, the cache of the block is automatically invalidated.

According to best practices- what are the two steps to accomplish this? (Choose two.)

A. 1. Override AbstractBlock: :getCecheKeyInfo and return an array containing the ids of all displayed entities.
2. Override AbstractBlock: :getCacheTags and return an array containing, for all displayed entities, the value returned by the getcachetag method of the model

B. 1. Create an around plugin on the save( ) method of the model of the entity.
2. Use the cleanCacheByTags( ) method Of \MagentoVFramework\App\Cache\FlushCaehebyTae with a single argument containing the concatenation of a chosen key and the td of the entity.

C. 1. Implement \Magento\framework\DataObject\identityInterface on the block that lists the entities.
2. Implement the getidentitiesO method on the block to return an array containing, for all displayed entities, the value returned by the getidentities() method of the model.

D. 1. Implement \magento\Frame\DataObject\identityInterface on the model of the entity.
2. Implement the getidentities() method to return the concatenation of a chosen key and the id of the entity.

ANSWER: A C

QUESTION NO: 6

An Adobe Commerce developer wants to cover their custom modules with Integration Tests. However, the project they are working on includes a 3rd party module that introduces a new search engine which needs to be used in Integration Tests as well. To do so. catalog/search/engine in the core.config_data table needs to be set to the customSearchEngine on the default scope.

They already created a phpunit.xml file in [m2 base dir/dev/tests/integration by copying unmodified content of phpunit.xml.dist from the same directory and will be using it for their tests.

How do they make sure that this setting is used for all the Integration Tests in their project using best practices?

A. Modify the phpunit.xml file they will be using and add the following node inside the phpunit node:

```
<config>
    <default>
        <catalog>
            <search>
                <engine>customSearchEngine</engine>
            </search>
        </catalog>
        <!-- other configuration here -->
    </default>
</config>
```

B. Modify or create a magento base dir]\ dev/test/integration/etc\config-global.php file and ensure that it contains the following content:

```
<?php
return [
    'catalog/search/engine' => 'customSearchEngine',
    // other configuration here
];
```

C. Include the following annotation in class-level docboack for event integration test class in the project:

```
/**
 * @magentoConfigFixture default catalog/search/engine customSearchEngine
 */
```

ANSWER: B

QUESTION NO: 7

An integration named Marketing is created on the Adobe Commerce instance. The integration has access on Magento_Customer::customer resources and the access token is xxxxxx .

How would the rest API be called to search the customers?

A)

Passing integration name and access token as http auth credentials:

```
curl -X GET https://Marketing:XXXXXX@magentourl/rest/V1/customers/search?searchCriteria...
```

B)

Using integration name as username and access token as password, get the admin token (YYYYYY) via:

```
curl -X POST https://magentourl/rest/V1/integration/admin/token -d '{"username":"Marketing", "password":"XXXXXX"}' -H 'Content-Type:application/json'
```

Use the admin token as Bearer `curl -X GET https://magentourl/rest/V1/customers/search?searchCriteria... -H 'Authorization: Bearer YYYYYY'`

C)

Using the integration access token as Bearer

```
curl -X GET https://magentourl/rest/V1/customers/search?searchCriteria... -H 'Authorization: Bearer XXXXXX'
```

A. Option A

B. Option B

C. Option C

ANSWER: C

QUESTION NO: 8

An Adobe Commerce developer is being tasked with storing additional data for products added to the cart in the quote. A new column should be added to the Quote_item table to store the value-Following best practices, how would the developer extend the database to accomplish this?

A)

Create a new Schema Patch implenting the `SchemaPatchInterface` to add the new column using the `apply()` method.

```
public function apply()
{
    $this->moduleDataSetup->getConnection()->startSetup();
    $this->moduleDataSetup->getConnection()->addColumn(
        $this->moduleDataSetup->getTable('quote_item'),
        'custom_data', [
            'type'     => 'varchar',
            'length'   => 255,
            'nullable' => true,
            'comment'  => 'New custom data'
        ]
    );
    $this->moduleDataSetup->getConnection()->endSetup();
    return $this;
}
```

B)

Create an UpgradeSchema script adds a new column to the `quote_item` table using the `upgrade()` method.

```
public function upgrade(SchemaSetupInterface $setup, ModuleContextInterface $conte
xt)
{
    $setup->startSetup();
    $setup->getConnection()->addColumn(
        $setup->getTable('quote_item'),
        'custom_data', [
            'type'     => 'varchar',
            'length'   => 255,
            'nullable' => true,
            'comment'  => 'New custom data'
        ]
    );
    $setup->endSetup();
}
```

C)

Create a new `db_schema.xml` file with the new column and a generate the `db_schema_whitelist.json` file.

```
<?xml version="1.0"?>
<schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="urn:magento:framework:Setup/Declaration/Sch
ema/etc/schema.xsd">
    <table name="quote_item">
        <column xsi:type="varchar" name="custom_data" nullable="true" length="255"
comment="New custom data"/>
    </table>
</schema>
```

A. Option

B. Option

C. Option

ANSWER: A

QUESTION NO: 9

A logistics company with an Adobe Commerce extension sends a list of reviewed shipment fees to all its clients every month in a CSV file. The merchant then uploads this CSV file to a "file upload" field in admin configuration of Adobe Commerce.

What are the two requirements to display the "file upload' field and process the actual CSV import? (Choose two.)

A)

Create an observer that listens to the adminhtml_config_system_save_after

```
class MyObserver implements ObserverInterface
{
    public function execute(\Magento\Framework\Event\Observer $observer)
    {
        $config = $observer->getData('config');
        $filePath = $config->getData('import_fees');
        /** @var \My\Module\Model\ImportFeed $importFees */
        $importFees = $this->importFeesFactory->create();
        $importFees->uploadAndImport($filePath);
    }
}
```

B)

Add a new field in etc.adminhtml/system.xml in my_Module with the file type:

```
<field id="import_fees" translate="label" type="file" sortOrder="1000" showInDefau
lt="1">
    <label>Import shipment fees</label>
    ...
</field>
```

C)

Add a custom backend model which extends /Magento\Framework\App\Config\Value and call afterSave:

```
// etc/adminhtml/system.xml
<field id="import_fees" ...>
    <label>Import shipment fees</label>
    <backend_model>My\Module\Model\Config\Backend\ImportFees</backend_model>

    ...

</field>
```

```
// \My\Module\Model\Config\Backend\ImportFees

class \My\Module\Model\Config\Backend\ImportFees extends \Magento\Framework\App\Co
nfig\Value

...
public function afterSave()
{
    /** @var \My\Module\Model\ImportFeed $importFees */
    $importFees = $this->importFeesFactory->create();
    $importFees->uploadAndImport($this);
    return parent::afterSave();
}
```

D)

Add a new field in etc/adminhtml\system.xml in My_Module with a new custom type:

```
<field id="import_fees" translate="label" type="My\Module\Block\Adminhtml\Form\Fie
ld\ImportFees" sortOrder="1000" showInDefault="1">
    <label>Import shipment fees</label>
    ...
</field>
```

A. Option A

B. Option B

C. Option C

D. Option D

ANSWER: B C

QUESTION NO: 10

An Adobe Commerce developer is tasked with adding custom data to orders fetched from the API While keeping best practices in mind, how would the developer achieve this?

A. Create an extension attribute on Magento\Seles\Api\Data\OrderInterface and an after plugin on Magento\Sales\Model\Order: :getExtensionAttributes() to add the custom data.

B. Create a before plugin on Magento\Sales\Model\ResourceMode\Order\Collection:: load and alter the query to fetch the additional data. Data will then be automatically added to the items fetched from the API.

C. Create an extension attribute on Magento\Sales\ApI\Data\Orderinterface and an after plugin on Magento\sales\Api\orderRespositoryinterface on get() and getlist () to add the custom data.

ANSWER: A