# Adobe

## AD0-E716 Exam

### Adobe Commerce Developer Expert

### Questions & Answers
### Demo

# Version: 4.0

## Question: 1

An Adobe Commerce developer has added an iframe and included a JavaScript library from an external domain to the website. After that, they found the following error in the console:
Refused to frame [URL] because it violates the Content Security Policy directive.
In order to fix this error, what would be the correct policy ids to add to the csp_whitelist.xml file?

A. frame-src and script-src
B. default-src and object-src
C. frame-ancestors and connect-src

**Answer: C**

Explanation:

The frame-ancestors directive specifies the domains that are allowed to embed the current page in an iframe. The connect-src directive specifies the domains that are allowed to be loaded by the current page through a <script> tag or XMLHttpRequest.
In this case, the developer has added an iframe that embeds a page from an external domain. The Content Security Policy (CSP) is preventing the iframe from being loaded because the domain of the external page is not listed in the frame-ancestors directive.
To fix this error, the developer needs to add the domain of the external page to the frame-ancestors directive. They can do this by adding the following line to the csp_whitelist.xml file:
<frame-ancestors>https://www.example.com</frame-ancestors>

## Question: 2

An Adobe Commerce Developer is tasked with creating a custom form which submits its data to a frontend controller They have decided to create an action and have implemented the \Magento\Framework\App\Action\HttpPostActioninterface class, but are not seeing the data being persisted in the database, and an error message is being shown on the frontend after submission.
After debugging and ensuring that the data persistence logic is correct, what may be cause and solution to this?

A. Magento does not allow POST requests to a frontend controller, therefore, the submission functionality will need to be rewritten as an API endpoint.
B. The developer forgot to implement a validatePostDataQ method in their action. They should implement this method: all non-validated POST data
gets stripped out of the request and an error is thrown.

C. Form key validation runs on all non-AJAX POST requests, the developer needs to add the for_key to their requests.

**Answer: C**

Explanation:

According to the Magento Stack Exchange answer, form key validation is a security feature that prevents CSRF attacks by checking if the form key in the request matches the one generated by Magento. If the developer does not include the form_key in their custom form, the validation will fail and an error will be shown. Therefore, the developer needs to add the form_key to their requests by using <?= $block->getBlockHtml ('formkey') ?> in their template file. Verified Reference: https://magento.stackexchange.com/questions/95171/magento-2-form-validation

## Question: 3

An Adobe Commerce developer is working on a module to manage custom brand entities and wants to replicate the following SQL query using SearchCriteria:

A)

```
$filter1->setField('featured')
    ->setValue(1)
    ->setConditionType('eq');

$filter2->setField('logo_image')
    ->setConditionType('notnull');

$filterGroup1->setFilters([$filter1, $filter2]);

$filter3->setField('enabled')
    ->setValue(1)
    ->setConditionType('eq');

$filterGroup2->setFilters([$filter3]);

$searchCriteria->setFilterGroups([$filterGroup1, $filterGroup2]);
```

B)

```
$filter1->setField('featured')
    ->setValue(1)
    ->setConditionType('eq');

$filter2->setField('logo_image')
    ->setConditionType('notnull');

$filter3->setField('enabled')
    ->setValue(1)
    ->setConditionType('eq');

$searchCriteria->setFilter($filter3)
    ->setOrFilter([$filter1, $filter2]);
```

C)

```
$filter1->setField('enabled')
    ->setValue(1)
    ->setConditionType('eq');

$searchCriteria->setFilter($filter1);
$filter2->condition('featured = ? OR logo_image ?, [1, 'IS NOT NULL'])
$searchCriteria->setSelectFields($filter2);
```

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

**Answer: B**

Explanation:

The following SearchCriteria query will replicate the SQL query:
$searchCriteria = new \Magento\Framework\Api\SearchCriteriaBuilder();
$searchCriteria->addFilter('name', 'Brand 1', 'eq');
$searchCriteria->addFilter('status', 1, 'eq');

$brandCollection = $this->brandRepository->getList($searchCriteria);

## Question: 4

The di. xml file of a module attaches two plugins for the class Action.
The PluginA has the methods: beforeDispatch, aroundDispatch, afterDispatch. The PluginB has the methods: beforeDispatch, afterDispatch.

```
<config>
    <type name="Magento\Framework\App\Action\Action">
        <plugin name="vendor_module_plugina" type="Vendor\Module\Plugin\PluginA" sortOrder="10" />
        <plugin name="vendor_module_pluginb" type="Vendor\Module\Plugin\PluginB" sortOrder="20" />
    </type>
</config>
```

The around plugin code is:

```
class PluginA
{
    public function aroundDispatch(\Magento\Framework\App\Action\Action $subject, $next, $request)
    {
        // custom code
        return $request;
    }
}
```

What would be the plugin execution order?
A)

```
PluginA::beforeDispatch()
PluginA::aroundDispatch()
PluginB::beforeDispatch()
Action::dispatch()
PluginB: afterDispatch()
PluginA::aroundDispatch()
```

B)

```
PluginA::beforeDispatch()
PluginA::aroundDispatch()
PluginA: afterDispatch()

PluginA::beforeDispatch()
PluginA::aroundDispatch()
PluginB::beforeDispatch()
```

C)

```
Action::dispatch()
PluginB: afterDispatch()
PluginA::aroundDispatch()
PluginA::afterDispatch()
```

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

**Answer: C**

Explanation:

The plugin execution order is as follows:
PluginA::beforeDispatch()
PluginB::beforeDispatch()
PluginA::aroundDispatch()
The code in the around plugin
PluginB::afterDispatch()
PluginA::afterDispatch()
The aroundDispatch() method is executed in a separate scope, so the code in the around plugin will be executed after the beforeDispatch() methods of both plugins, but before the afterDispatch() methods of

both plugins.
Here is a diagram that shows the plugin execution order:
PluginA
  beforeDispatch()
  aroundDispatch()
  afterDispatch()
PluginB
  beforeDispatch()
  afterDispatch()

## Question: 5

An Adobe Commerce developer adds a new extension attribute to add an array of values to the invoices that are fetched through the APIs.
After a while, their technical manager reviews their work and notices something wrong with the extension_attributes. xml file that the developer created in their module:
What is the problem with this xml snippet?

A. The extension attribute references the wrong interface, it should have referenced the Magento\saies\Api\data\invoiceinterface.
B. The extension attribute references the repository instead of the interface it implements (Magento\saies\Api\invoiceRepositorymterface).
C. The type is wrong, string [] should be replaced with array.

**Answer: B**

Explanation:

The extension attribute is referencing the repository instead of the interface it implements. The correct XML snippet should be:
XML
```
<extension_attributes>
  <attribute code="custom_values" type="string[]"
    group="General"
    translate="true">
    <description>This attribute stores an array of custom values for the invoice.</description>
    <source_model>Magento\Sales\Api\Data\InvoiceInterface</source_model>
  </attribute>
</extension_attributes>
```
The source_model attribute specifies the interface that the extension attribute is associated with. In this case, the extension attribute is associated with the Magento\Sales\Api\Data\InvoiceInterface interface.