Adobe

AD0-E718 Exam

Adobe Commerce Architect Master Questions & Answers Demo

Version: 5.1

Question: 1

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. Reference: https://devdocs.magento.com/guides/v2.4/howdoi/checkout/checkout_new_total.html

Question: 2

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: 3

An Adobe Commerce Architect is asked by a merchant using B2B features to help with a configuration issue.
The Architect creates a test Company Account and wants to create Approval Rules for orders. The Approval Rules tab does not appear in the Company section in the Customer Account Menu when the Architect logs in using the Company Administrator account.
Which two steps must be taken to fix this issue? (Choose two.)

A. Set 'Enable Purchase Orders' in the B2B Admin to TRUE
B. Merchant needs to log out of frontend and then log back in to load new permissions
C. Set Enable Purchase Orders' on the Company Record to TRUE
D. Make sure that the 'Purchase Order' payment method is active
E. Set 'Enable B2B Quote" in the B2B Admin to TRUE

Answer: A, C

Explanation:

Enabling Purchase Orders at both the B2B Admin and the Company Record levels is necessary for Approval Rules to appear in the Company section of the Customer Account Menu. When 'Enable Purchase Orders' is set to TRUE, the system assumes that the company will be making purchases using purchase orders, and the Approval Rules tab becomes visible.

To create Approval Rules for orders, the Architect needs to enable Purchase Orders both in the B2B Admin and on the Company Record. This will allow the Company Administrator to access the Approval Rules tab in the Customer Account Menu and create rules based on various criteria. The Purchase Order payment method and the B2B Quote feature are not required for this functionality. Reference: https://docs.magento.com/user-guide/customers/account-dashboard-approval-rules.html

Question: 4

An external system integrates functionality of a product catalog search using Adobe Commerce GraphQL API. The Architect creates a new attribute my_attribute in the admin panel with frontend type select.
Later, the Architect sees that Productinterface already has the field my_atcribute, but returns an mc value. The Architect wants this field to be a new type that contains both option id and label.
To meet this requirement, an Adobe Commerce Architect creates a new module and file etc/schema.graphqls that declares as follows:

```
interface ProductInterface {
    my_attribute: SelectableOption @resolver(class:"Vendor\\CatalogGraphQl\\Model\\Resolver\\SelectableOption")
}
```

After calling command setup:upgrade, the introspection of ProductInterface field xy_attribute remains int. What prevented the value type of field my_attribute from changing?

A. The fields of ProductInterface are checked during processing schema.graphqls files. If they have a corresponding attribute, then the backendjype of product attribute is set for field type.
B. The interface ProductInterface is already declared in Magento.CatalogGraphQI module. Extending requires use of the keyword -xceni before a new declaration of ProductInterface.
C. The Magento.CatalogGraphQI module occurs later in sequence than the Magento.GraphQI module and merging output of dynamic attributes schema reader overrides types declared in schema.graphqls

Answer: A

Explanation:

According to the Adobe Commerce Developer Guide1, the fields of ProductInterface are checked during processing schema.graphqls files. If they have a corresponding attribute, then the backend_type of product attribute is set for field type. Therefore, the value type of field my_attribute remains int because it corresponds to the attribute my_attribute that has a frontend type select and a backend_type int.

Question: 5

An Adobe Commerce Architect is creating a new GraphQL API mutation to alter the process of adding configurable products to the cart. The mutation accepts configurable product ID. If the given product has only one variant, then the mutation should add this variant to the cart and return not nullable cart type. If the configurable product has more variants, then the mutation should return not nullable conf igurableProduct type.
The mutation declaration looks as follows:

```
type Mutation {
    addConfigurableToCart(product_id: Int!): AddToCartOutput!
        @resolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddConfigurableToCart")
}
```

How should the Adobe Commerce Architect declare output of this mutation?

A)

```
interface AddToCartOutput
@typeResolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddToCartOutputTypeResolver") {
}

type ConfigurableProduct implements AddToCartOutput {
}

type Cart implements AddToCartOutput {
}
```

B)

```
union AddToCartOutput
@typeResolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddToCartOutputTypeResolver")
= ConfigurableProduct | Cart
```

C)

```
type AddToCartOutput {
    product: ConfigurableProduct
    cart: Cart
}
```

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

Answer: C

Explanation:

According to the Adobe Commerce Developer Guide2, a union type is a special kind of object that can be one of several types. It is useful for returning disjoint data types from a single field. In this case, the output of the mutation can be either Cart or ConfigurableProduct, depending on the number of variants of the given product. Therefore, a union type should be declared for the output of this mutation, as shown in option C. The other options are not valid syntax for union types.