

AD0-E724 New Guide Files - Reliable AD0-E724 Exam Sims



BTW, DOWNLOAD part of Braindumpsqa AD0-E724 dumps from Cloud Storage: <https://drive.google.com/open?id=1zQLmF9iRYMGWp5e6CTNVjzzoOntSISV>

Braindumpsqa's products are developed by a lot of experienced IT specialists using their wealth of knowledge and experience to do research for IT certification exams. So if you participate in Adobe certification AD0-E724 exam, please choose our Braindumpsqa's products, Braindumpsqa can not only provide you a wide coverage and good quality exam information to guarantee you to let you be ready to face this very professional exam but also help you pass Adobe Certification AD0-E724 Exam to get the certification.

Candidates all around the globe use their full potential only to get Adobe AD0-E724 certification. Once the candidate is a Adobe certified, he gets multiple good career opportunities in the Adobe sector. To pass the AD0-E724 Certification Exam a candidate needs to be updated and reliable Commerce Developer Professional (AD0-E724) prep material.

>> AD0-E724 New Guide Files <<

Reliable AD0-E724 Exam Sims - Valid AD0-E724 Exam Sims

Not only that our AD0-E724 exam questions can help you pass the exam easily and smoothly for sure and at the same time you will find that the AD0-E724 guide materials are valuable, but knowledge is priceless. These professional knowledge will become a springboard for your career, help you get the favor of your boss, and make your career reach it is peak. What are you waiting for? Come and take AD0-E724 Preparation questions home.

Adobe Commerce Developer Professional Sample Questions (Q39-Q44):

NEW QUESTION # 39

An Adobe Commerce developer has added a new configuration field to the admin area. The path for this option is `general/store_information/out_of_hours_phone`.

Keeping simplicity in mind, how would the developer ensure this option contains a valid US telephone number?

- A. Add `<validate>phoneUS</validate>` to the field in `system.xml`.

- B. Add <validate type="phoneUS"/> to the field in system.xml.
- C. Create a backend model to check the validity of the phone number entered.

Answer: A

Explanation:

According to the Magento Stack Exchange answer, system.xml is a file that defines the configuration fields for the admin area. Each field can have a validate attribute that specifies a validation rule for the field value.

Magento provides some built-in validation rules, such as phoneUS, which validates a US telephone number.

Therefore, to ensure that the option contains a valid US telephone number, the developer needs to add <validate>phoneUS</validate> to the field in system.xml. Verified References:<https://magento.stackexchange.com/questions/104570/magento-2-system-xml-validation-rules>

When adding a new configuration field to the admin panel and needing to ensure it contains a valid US telephone number, you can leverage Magento's built-in validation options within the system.xml configuration file. The correct approach is to use the <validate> tag with the desired validation type.

* Built-in Validation with system.xml:

* Adobe Commerce provides various validators that can be directly applied in the system.xml file for configuration fields. These validators are handled through JavaScript on the client side, ensuring real-time validation.

* Using <validate>phoneUS</validate>:

* The tag <validate> specifies the type of validation that should be enforced. For US phone numbers, you can simply set <validate>phoneUS</validate>.

* Magento interprets this and applies a validation check to ensure the entered data conforms to the US phone number format.

* Why Option A is Correct:

* Option A leverages existing Magento client-side validators, which are simpler and more efficient than creating a custom backend model for a straightforward validation task.

* Options B and C do not apply the correct method for a simple validation task as per Magento standards.

* Example system.xml Configuration:

```
<field id="out_of_hours_phone" translate="label comment" type="text" sortOrder="100" showInDefault="1" showInWebsite="1" showInStore="1">
<label>Out of Hours Phone</label>
<validate>phoneUS</validate>
<comment>Enter a valid US phone number</comment>
</field>
```

NEW QUESTION # 40

An international merchant is complaining that changes are taking too long to be reflected on the frontend after a full product import. Thinking it may be database issues, the Adobe Commerce developer collects the following entity counts:

- * Categories: 900
- * Products: 300k
- * Customers: 700k
- * Customer groups : 106
- * Orders: 1600k
- * Invoices: 500k
- * Creditmemos: 50k
- * Websites : 15
- * Stores : 45

What is a probable cause for this?

- A. The combination of the number of orders, customers, invoices and creditmemos is too big. This leads to a huge amount of values being stored in the customer grid index which is too large to be processed at a normal speed.
- B. The combination of the number of products, customer groups and websites is too big. This leads to a huge amount of values being stored in the price index which is too large to be processed at a normal speed.
- **C. The combination of the number of products, categories and stores is too big. This leads to a huge amount of values being stored in the flat catalog indexes which are too large to be processed at a normal speed.**

Answer: C

Explanation:

The combination of a large number of products, categories, and stores directly affects the flat catalog indexing process. Each product needs to be indexed across all categories and stores, which exponentially increases the data size of the index tables. This

can significantly slow down the frontend updates after a full product import due to the high volume of data that needs to be processed.

Impact on Flat Catalog and Indexing:

In Magento, flat catalog tables are used to optimize product data retrieval, especially in stores with a large catalog. However, with high numbers of products, categories, and stores, the flat catalog tables become massive, making them slower to update.

In this scenario, the system must create and maintain an entry for each product in each store across every category, leading to a substantial volume of data.

Why Option A is Correct:

This combination (products, categories, stores) is known to cause performance issues in indexing and data storage. It impacts the catalog indexes, which are critical for reflecting updates on the frontend.

Options B and C do not directly relate to the delay after product imports. Option B involves customer-related data, and Option C is related to price indexing, which affects a different area.

Optimization Recommendations:

Consider disabling the flat catalog and leveraging Elasticsearch for product and category search and filtering.

Additionally, increasing server resources or partitioning the data could help optimize performance.

NEW QUESTION # 41

An Adobe Commerce developer is working on a Magento 2 instance which contains a B2C and a B2B website, each of which contains 3 different store views for English, Welsh, and French language users. The developer is tasked with adding a link between the B2C and B2B websites using a generic link template which is used throughout the sites, but wants these links to display in English regardless of the store view.

The developer creates a custom block for use with this template, before rendering sets the translate locale and begins environment emulation using the following code:

```
/** @var $this->_translate \Magento\Framework\TranslateInterface */
$this->_translate->setLocale($newLocaleCode);

/** @var $this->_emulation \Magento\Store\Model\App\Emulation */
$this->_emulation->startEnvironmentEmulation($storeId, \Magento\Framework\App\Area::AREA_FRONTEND);
```

They find that the template text is still being translated into each store's language. Why does this occur?

- A. `setLocale()` does not change translation locale after it has been initially set, the `$this->_translate->emulate($newLocaleCode)` method exists to temporarily modify this by pushing the new locale to the top of the current emulatedLocales stack.
- B. `startEnvironmentEmulation()` resets the translation locale to the one of the emulated stores, which overrides the locale the developer has set when the order of `setLocale()` and `startEnvironmentEmulation()` is used as displayed above.
- C. `startEnvironmentEmulation()` sets and locks the locale by using the `setLocale()` Optional Second \$lock parameter, i.e. `setLocale($newLocaleCode, true)`, to override and lock the locale of the emulated store. If this is set and locked initially then the environment emulation will not be able to override this.

Answer: B

Explanation:

The `startEnvironmentEmulation()` method resets the translation locale to the one of the emulated stores, which overrides the locale the developer has set when the order of `setLocale()` and `startEnvironmentEmulation()` is used as displayed above.

The correct way to achieve the desired result is to use the `emulate()` method to temporarily modify the translation locale. The following code shows how to do this:

PHP

```
$this->_translate->emulate('en_US');
```

```
// Render the template
```

```
$this->_translate->revert();
```

This code will set the translation locale to English before rendering the template, and then revert the locale back to the default value after the template has been rendered.

The `startEnvironmentEmulation()` method is used to emulate a different store view or website. This can be useful for testing purposes, or for developing features that need to work in different environments.

The `emulate()` method is used to temporarily modify the translation locale. This can be useful for rendering templates in a specific language, or for testing features that need to work in different languages.

NEW QUESTION # 42

What is the correct way to inject CMS block in a layout?

- A. `<block class="Magento\Cms\Block\Block" name="block_identifier"> <arguments> q <argumentname="block_id" xsi:type="string">my_cms_block_identifier</argument> </arguments> </block>`
- B. `<referenceBlock name="content"> <block class="Magento\Cms\Block\Block" name="block_identifier" identifier="my_cms_block_identifier" /> </referenceBlock>`
- C. `<block class="Magento\Cms\Block\Block" name="block_identifier"> q <actionmethod="setBlock">my_cms_block_identifier</action> </block>`

Answer: A

Explanation:

The correct way to inject a CMS block into a layout in Adobe Commerce is by using the `<block>` element with the class `Magento\Cms\Block\Block` and specifying the block identifier through an `<argument>` element with the name `"block_id"`. This is shown in option A. The `<block>` tag defines the block class and name, and the `<arguments>` tag contains child `<argument>` tags for each argument, where the `"block_id"` argument specifies the identifier of the CMS block to be injected.

NEW QUESTION # 43

An Adobe Commerce developer is creating a new module to extend the functionality of the cart. The module is installed in `app/code/CompanyName/ModuleName/`.

How would an Adobe Commerce developer extend the existing `CartItemPrices` GraphQL schema to include a custom `base_price` field?

- A. Create and Configure a `<preffrence>` for `Hagento\QuoteGraphQL\Model\Resolver\CartItemPrices` that adds the `base_price` field in the `resolve()` function.
- B. Option B
- C. Option C
- D. Add the following to the module's `etc/schema.graphqls` file:

A black text on a white background AI-generated content may be incorrect.

```
type CartItemPrices {  
  base_price: Money!  
}
```

C)

Add the following to the module's `etc/graphql/di.xml` file:

A screen shot of a computer code AI-generated content may be incorrect.

```
<type name="Magento\QuoteGraphQL\Model\Resolver\CartItemPrices">  
  <arguments>  
    <argument name="extendedConfigData" xsi:type="array">  
      <item name="base_price" xsi:type="number"/>  
    </argument>  
  </arguments>  
</type>
```

- E. Option A

Answer: D

Explanation:

The developer can extend the existing `CartItemPrices` GraphQL schema to include a custom `base_price` field by adding the following code to the module's `etc/schema.graphqls` file:

`extend type CartItemPrices { base_price: Money! @doc(description: "The base price of the cart item") }` This code adds a new field called `base_price` to the `CartItemPrices` type and specifies that it is of type `Money` and it is not nullable. The `@doc` directive adds a description for the field that will be shown in the schema documentation. The developer also needs to create a custom resolver class for the `base_price` field and declare it in the `di.xml` file of the module. Verified References: [Magento 2.4 DevDocs] [Magento Stack Exchange]

NEW QUESTION # 44

.....

Challenge is omnipresent like everywhere. By eliciting all necessary and important points into our AD0-E724 practice materials, their

Reliable AD0-E724 Exam Sims: https://www.braindumpsqa.com/AD0-E724_braindumps.html

Code review for security has come a long way Exam AD0-E724 Materials in the last few years, and commercial tools are now mature enough to be put in use by all software practitioners, Things like AD0-E724 archiving all their tapeless media, tools to facilitate that process, and so on.

With increasingly higher awareness of the importance of the Adobe Commerce certification, people get to know that a reliable AD0-E724 exam study material is really helpful.

- BTW, DOWNLOAD part of Braindumpsqa AD0-E724 dumps from Cloud Storage: <https://drive.google.com/open?id=1zOLmf9iRYMGWp5e6CTNVizzoOntSISV>