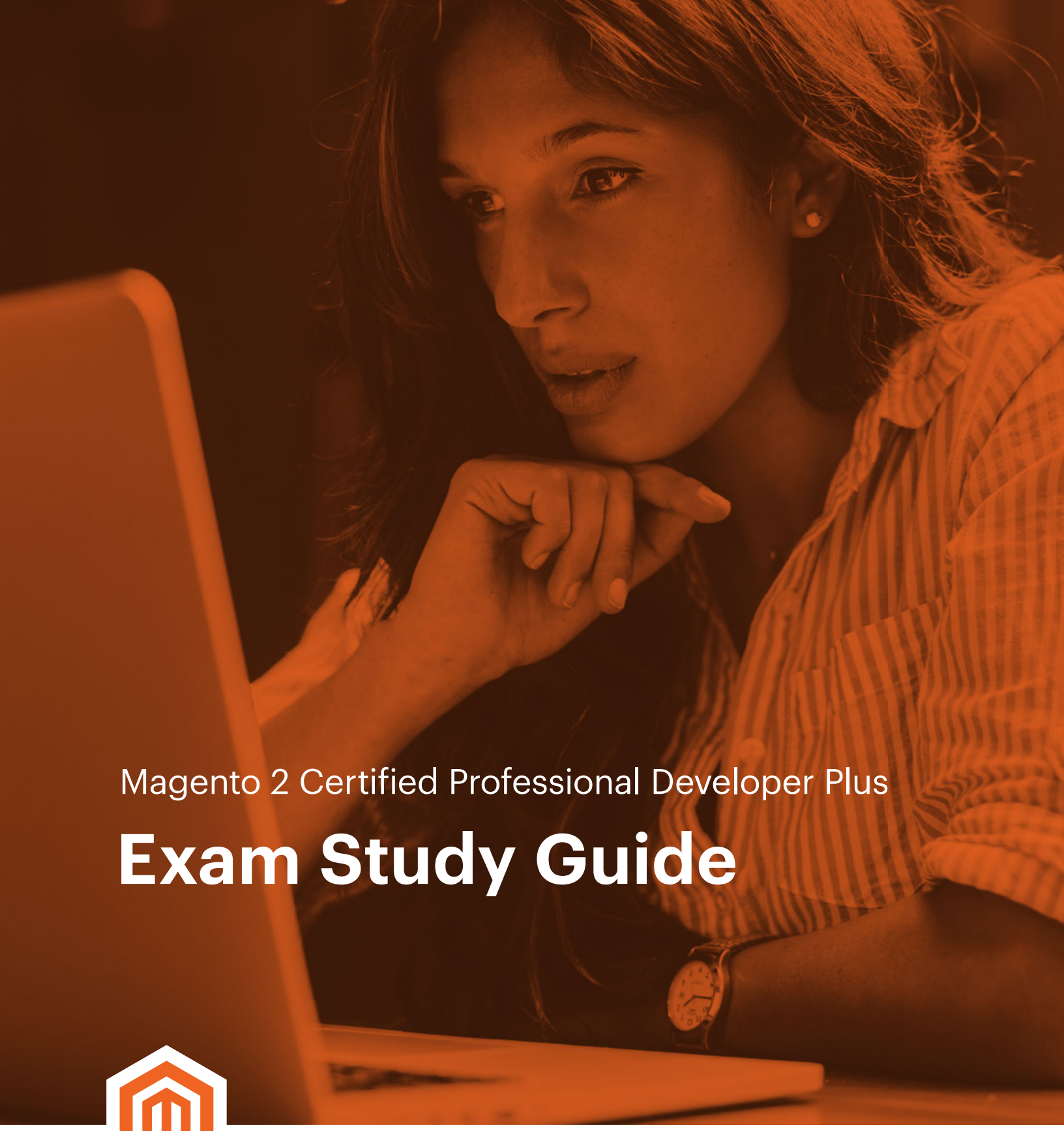

Magento 2 Certified Professional Developer Plus

Exam Study Guide

MAGENTO 2 CERTIFIED
Professional Developer Plus

Magento® U
An Adobe Company

training@magento.com

u.magento.com

u.magento.com/certification

Contents

Introduction

This exam is for a senior Magento 2 developer/architect with 2 years of experience in customizing different areas of Magento Commerce, leading teams of Magento developers, leading projects, making key technical decisions on a Magento project, and working with customers to build project requirements.

By passing this exam the developer will earn Magento 2 Certified Professional Developer Plus credential.

This exam will validate the skills and knowledge needed to customize Magento in these areas: core architecture, UI modifications, catalog, checkout, Magento Commerce features, and security.

The exam will also validate the ability to make architectural decisions and to forecast the impact of a customization, understanding of core mechanisms in the most important areas like price calculation for a product, checkout, and quote operations. The test is built for 2.2.*x* version of Magento Commerce software.

This exam consists primarily of scenario-based questions in a multiple-choice format.

This guide contains several sample questions at the end of the guide.

Exam topics and the percentage covered in the test.

- Magento Architecture 6%
- Magento UI 7%
- Working with Databases 14%
- Using the Entity-Attribute-Value (EAV) Model 10%
- Developing with Adminhtml 5%
- Customizing the Catalog 23%
- Customizing the Checkout Process 17%
- Magento Commerce Features 13%
- Understanding Magento Security 5%

Topics and Objectives

1 Magento Architecture

1.1 Determine advanced uses of the Magento configuration system

- Understand how to create a custom config file. Demonstrate an understanding of the Magento Configuration files framework
- Understand how to create a custom config file with validation and a unique node that is overridden on merging
- Understand how to create a config file with a remote schema

1.2 Demonstrate an ability to design complex customizations using plugins and di.xml

- Plugins sort order, plugin on plugin scenario, plugin debugging techniques. Demonstrate an understanding of virtual types, shared objects, object instantiation process, proxies, factories
- How does an around plugin modify the plugin execution order?
- How do you debug a plugin that is not executed?
- Demonstrate Plugin on Plugin examples
- Which classes are instantiated outside of the ObjectManager so they cannot be customizing using di.xml?
- Demonstrate a use case for a virtual type (different instances of a class with a different set of arguments)

1.3 Demonstrate understanding of Magento events processing

- Demonstrate an understanding of the events processing flow. Influence of Staging on the event processing
- What is a modification of the event processing mechanism introduced by the staging module?

1.4 Demonstrate an ability to use the Magento command-line interface

- Create a new CLI command, emulate different areas within it
- Create a new CLI-command, configure it in di.xml, add optional/required options/keys
- Environment specification using Area class
- Environment emulation for a section of code

2 Magento UI

2.1 Demonstrate understanding UiComponents architecture

- UiComponent workflow, initialization, execution, configuration structure, data loading process
- Retrieving a UiCompnent's instance from the uiRegistry
- Understand the difference between executing a data provider component and loading data
- Describe the role of UiComponent PHP classes
- Understand the uiClass instance, extending uiComponent

2.2 Demonstrate advanced use of Magento layouts

- Non-standard layouts, custom handles, debugging layouts
- Add a custom handle, obtain a list of handles loaded for a page
- Obtain the layout XML for a page
- Containers elements with a wrapping DIV tag
- Dynamically modify the layout tree

2.3 Demonstrate an ability to operate with Magento blocks and templates

- Block caching, fallback debugging, email templates, translations
- Cache all instances of the block, specific instance
- Assign an object to the email template. Render different images depending on a locale in the email template
- Identify the location of block instantiation
- Print out all places where Magento looks for a template
- Demonstrate an understanding of different types of translations working together (inline, phrase in JavaScript code, CSV file)

3 Working with Databases

3.1 Demonstrate understanding of the architectural layers of the database access classes, including models, repositories, and data mappers

- Models, resource models, and collections in Magento, their impact on performance. Repositories, SearchCriteria, WebAPI, WebAPI access, extension attributes
- How to create an entity that supports extension attributes
- How to implement SearchCriteria processing in the repository::getList method
- How to perform bulk save operations in Magento

- How to extend the Magento data object (Data API class) with an attribute that has values in a remote system
- How to extend exisitng WebAPI calls with a new parameter
- How to create a dynamic WebAPI ACL
- The difference between extension attributes and custom attributes

3.2 Demonstrate understanding of the staging workflow

- Staging modification to the Magento database operations (row_id, entity managers)
- How does data versioning work?
- Different possibilities of data versioning (row/table/database level) and how this is implemented in Magento
- The role of the entity manager
- High level staging implementation overview

3.3 Demonstrate an ability to use different types of setup scripts in Magento

- Schema and data setup scripts, uninstall scripts, recurring scripts, uninstall schema vs. uninstall data
- What happens when an uninstall script is executed: data version change, deleted tables, etc.
- Recurring scripts and their order in the setup:upgrade process
- Accessing areas and system configuration values in setup scripts

4 Using the Entity-Attribute-Value (EAV) Model

4.1 Describe the EAV data access process in Magento

- Getting an attribute instance, impact of attribute sets, large number of attributes and attribute sets
- What is the impact of 10,000 attribute sets? 1,000 attributes in a set?
- How to get information about an attribute
- How to perform attribute operations programmatically: assign it to a set/group, update properties, etc.

4.2 Describe the database tables for EAV entities and how to create them

- The EAV database structure, performance considerations, entity-level attribute properties (catalog_eav_attribute)
- Where are catalog-specific attribute properties stored and what are they used for?
- How does Magento store the attribute to attribute group association?
- What backend types are available? How do you add a new backend type?
- Specifics around static attributes

4.3 Demonstrate an ability to operate with attribute options

- Different ways to store attribute options. Using eav_attribute_option_* tables
- Config base, database base options
- The eav_attribute_option_ table: tables that contain shared options between different entities, pros and cons of using the table

4.4 Demonstrate an ability to use non-catalog EAV entities

- Adding an attribute to Customer, Customer Address and Sales entities. Making an attribute visible in the Admin or the storefront. Pitfalls in attributes operations in non-catalog EAV attributes
- Adding an attribute to customers, saving and loading the attribute, problems related to the save process. What is the role of attribute sets and groups for customer attributes?
- Adding an attribute to customer addresses, the role of the ""is_system"" property and why it only works for the Customer Address entity
- How to make a customer or customer address attribute visible in the My Account, Checkout, and Admin pages
- What is the purpose of the SalesSetup class and why do you use the addAttribute method for sales entities?

5 Developing with Adminhtml

5.1 Demonstrate ability to use ACL

- Complex cases of ACL setup. WebAPI ACL, ACL process customization and debugging
- How to debug an ACL record
- How does a row-based ACL or IP-based ACL work in Magento?
- The connection between admin ACL and WebAPI ACL
- Different ways to access WebAPI resources including admin access

5.2 Demonstrate understanding of the admin login process and admin actions processing

- Admin login, customizing and debugging issues related to admins logging into Magento, the Admin Action class
- Debugging the login process
- Logging in an admin user programmatically
- Customizing the login process: for example, adding 2-factor authentication
- Operations performed by the Magento\Backend\App\Action class, for example, the secret key

5.3 Demonstrate an ability to create complex forms and grids

- Complex forms with custom elements and with tabs. Complex grids with custom columns and inline editing customizations
- Create custom elements for forms
- Create a form with tabs
- Create a form with a grid inside of a tab
- Forms for editing related/nested data
- Customization of inline editing in a grid; for example, a file uploader
- Bookmark filters selection for a grid
- Grid meta information: adding a new column that requires a join to another table

6 Customizing the Catalog

6.1 Demonstrate an ability to understand and customize Magento products

- Selecting the right product type for a given requirement (configurable with custom options, bundle with grouped). Deciding to use non-standard products: for example, for licenses, subscriptions, courseware, or glasses. Product relations (related, upsells).
- Select a product type for a subscription/subscription bundled with a physical product
- Select a product type for courseware
- Select a product type for glasses with a prescription
- Compare custom options with configurable products
- Configurable product with custom options for the associated simple products
- Bundled product with custom options for its associated simple products
- What is the related products database structure? Understand the performance impact of having many related products
- Compare related with upsells
- Programmatically access related products, custom options, configurable parameters
- Dynamic related products

6.2 Demonstrate an ability to perform complex operations with the Magento pricing framework

- Understand the pricing calculation and rendering framework. Which classes are involved in rendering/calculation? What is the role of indexing? How do different price modifiers work together?
- How is a price calculated on the product detail page, the product listing page? Price calculation classes. Relation to the price indexer

- The Magento\Framework\Price* component and its extension in the Catalog module
- How to render a product price: which class to use, which data needs to be provided
- Indexed price vs. prices calculated on the fly
- Price configuration: tier prices, special price, custom option, configurable adjustment, catalog rules, cart rules
- How to change the price rendering process for a given product type/product instance
- How to add a custom price adjustment to modify the calculation process. What happens if the price index is not aligned with this change?

6.3 Customize catalog price rules

- Programmatically create a catalog price rule, the impact of catalog price rules on performance, extending catalog price rule conditions with custom entities

6.4 Determine how to use Magento categories

- Advanced category features: hierarchy, custom attributes, impact on performance
- What happens if a project has many categories?
- Dynamic rules for the order of products in category
- The category is_anchor attribute and its effect on performance

6.5 Demonstrate an understanding of catalog indexers

- The indexing framework, the price indexer, the inventory indexer, the EAV indexer. How does Magento use indexed data?
- Estimate indexing customization efforts when making architectural decisions. Estimate indexing process complexity and time for different given conditions. The impact of indexing for frequent catalog updates.
- What steps does Magento perform when indexing? What is the role of the indexer_state table
- How to register a new indexer
- How does the price indexer work? Which events trigger it? How do different product types declare their indexers? How important is an order of indexers in price indexing?
- Inventory indexer overview, inventory indexing for different product types
- What is the scope of a price in the price index? How difficult is it to extend the scope of an index?
- Impact of many stores/websites on price indexing
- Custom price modifiers: Pros and cons of customizing the index versus adjusting the native features like price rules or custom options

6.6 Demonstrate understanding of catalog staging and its impact on the system

- Flow modifications introduced by staging (triggers, row_id, data versions). Staging-related modifications of the indexing process

- Issues related to different row_id and entity_id
- Catalog triggers

6.7 Demonstrate an understanding of the product search framework

- How to customize Elastic search

6.8 Demonstrate understanding of importing products and categories in Magento

- Frequent imports, massive imports, import with many attributes
- Issues related to a frequent import, for example every minute
- Compare importing products using the native import, save by model, or custom SQL
- Specifics of importing a catalog with many attributes and attribute sets
- Importing categories and product relations

7 Customizing the Checkout Process

7.1 Understand the Magento quote architecture and customizing quote-related functionality

- Quote-related objects, total models and the price calculation process, the add to cart process, custom add to cart operations, customizations of the price calculation, taxes and discount, various display settings
- The quote merge functionality
- Programmatically add a free gift when a certain condition is met
- Programmatically set a price for a product
- Taxes with discounts calculation
- The impact of many shopping cart price rules on performance
- Programmatically create shopping cart price rules
- Importing coupon codes
- Extend the shopping cart price rules with custom entities
- Programmatically separate line items in the shopping cart so there are two line items instead of one with qty=2
- Adding a new total model and evaluating its impact on taxes and discounts
- Shipping discounts behavior and customizations
- WebAPI for quote operations. Create a quote, add an item, create a coupon, add a discount

7.2 Demonstrate an ability to customize and extend the checkout process

- Checkout steps, the REST API, customizations of the checkout API, the order placement process. The impact of many concurrent order placements
- Adding a step to the checkout after the payment step, or between the payment and shipping steps
- Implementing a ""one-click"" checkout, evaluate possible pitfalls such as discounts being applied or canceled when the order is placed
- The checkout REST API: Modifying the native flow (separate calls to save payment and to place the order)
- Extending the checkout REST API. Adding new parameters to different API endpoints. Using extension attributes
- Issues related to simultaneous order placement, inventory locking
- Determining the exact moment when the stock is decremented during an order placement
- Customizing the order placement such that it uses message queues
- Horizontal sharding of orders tables to improve order placement capacity. What challenges need to be resolved?

7.3 Create and debug shipping and payment methods in Magento

- Different types of payment methods: gateway, offline, hosted. The gateway payment methods framework. The payment method availability logic. The shipping rates calculation process, debugging shipping rates and table rates customizations
- The gateway framework vs. AbstractMethod
- The structure of offline payment methods. What makes them "offline"?
- Partial invoices/refunds for payment methods
- Add a hosted-type payment method (redirects, custom order review page)
- Add a new shipping method. Customize the logic for getting a list of available shipping methods
- The shipping rate calculation flow. Debug shipping rates, identify plugins that that influence the shipping rate calculation process
- Customizing table rates. Add a new "column" to the "table", change the logic of selecting the rate.
- Gateway shipping methods. Remote call flow. How to avoid the remote call if it is unnecessary

8 Magento Commerce Features

8.1 Demonstrate an ability to use message queues

- Create a listener/publisher. Use cases for using message queues
- How to register a listener/publisher for a queue
- How to configure a queue
- What is a use case to use message queues?

8.2 Demonstrate understanding of customer segmentation

- Describe the customer segments functionality, its use cases, impact on performance, and programmatical operations with customer segments
- Programmatically create a customer segment
- Extend customer segments with new entities or attributes
- Understand the database structure of customer segments

8.3 Demonstrate understanding of advanced capabilities in Magento Commerce

- Store credits, reward points, RMA, gift cards
- Compare RMA with refunds
- Customizing the RMA process
- Customizing store credits/reward points accrual/spending

8.4 Demonstrate understanding of target rules

- Customer rules database structure, use cases, comparison to other product relations, operate with target rules programmatically
- Target rules vs. related products
- Extending target rules with a custom attribute or a custom entity
- Create a rule programmatically, understand the database structure of a rule
- What is the performance impact of having many target rules in the system?
- How does editing a product or adding new products affect existing target rules?

9 Understanding Magento Security

9.1 Demonstrate understanding of frontend security with Magento

- Filter input and escape output; password and sensitive data hashing; validate user file uploads; how to prevent cross site scripting vulnerabilities
- How to validate field values before inserting them into the database
- How to escape output to remove HTML tags
- Using secure encryption and hashing functions to store sensitive values in the database
- Validate file uploads, the file type, the file contents, and the file metadata

9.2 Demonstrate understanding Adminhtml security with Magento

- CSRF tokens; stored XSS; adminhtml secret key; security configuration; securing email from injection; preventing admin privilege escalation

- How to prevent CSRF attacks; the importance of the form_key field in forms
- How is the admin secret key generated in URLs?
- Security related admin configurations
- Security by obscurity, the custom admin path

9.3 Demonstrate understanding of different types of attacks and preventing them with Magento

- Remote code execution; local and remote file inclusions; session hijacking; SQL injection; insecure functions; directory traversal attacks
- PHP functions that should never be used, per Magento recommendations (eval, serialize, etc.)
- How to prevent directory traversals attacks when uploading a file
- Importance of the HttpOnly property when setting a cookie
- Serving Magento from the base vs. the pub/ directory
- Retrieving the user IP from the REMOTE_ADDR header vs. the X-Forwarded-For header
- File operations influenced by user-submitted request parameters, how to stop directory traversal attacks and restrict access to dirs/files
- How to write secure SQL queries

Magento 2 Certified Professional Developer Plus Exam *Example Questions*

See the Answer Key following the questions for answers and references.

Question 1

In your phtml template file you need to output a URL inside a JavaScript context.

```
var url = '<?= /* code here */ ?>';
```

Which two methods allow you to keep the output XSS-safe?

A. escapeUrl
B. escapeHtmlAttr
C. escapeHtml
D. escapeJs

Question 2

You are working on a custom form in the Admin and the form is too lengthy. To organize the form better, you decide to group the fields into multiple tabs.

How do you achieve this?

A. Create a plugin for the method `MyCompany\MyModule\Block\Adminhtml\Form\Edit\Tabs::toHtml()`
B. Add the fields into <tab> nodes in the customer_account_edit.xml layout file
C. Add the fields into a <fieldset> node in the existing form in the ui_component XML file
D. Add a new form for the field group in the ui_component XML file

Question 3

You are making some changes to your existing action controller

`\MyCompany\MyModule\Controller\Index\Index`

```
public function __construct(
    \Magento\Framework\App\Action\Context $context,
    <new dependency here>,
    \Magento\Framework\View\Result\PageFactory $resultPageFactory
) {
```

You want to inject a new dependency to the controller, but you encounter the following error after flushing the cache and reloading the page:

Recoverable Error: *Argument 2 passed to MyCompany\MyModule\Controller\Index\Index::__construct() must be an instance of <new dependency class> ...*

How do you fix the error?

A. New dependencies must be added at the end of the constructor signature because dependencies cannot be added in the middle of existing constructors

B. Remove the generated child class from generated/code that is calling the parent constructor with the old signature

C. Clean the config cache that contains all constructor signatures

D. Configure the new argument in di.xml for the controller class

Question 4

You are working with an OMS that requires you to add an attribute to the order API.

How do you add a field to the existing sales API?

A. You update `Magento\Sales\Api\OrderManagementInterface` and `Magento\Sales\Api\Data\OrderInterface` to add your field

B. You add an extension attribute to `Magento\Sales\Api\Data\OrderInterface` to include the new field

C. You create an install script to add a column to the sales table and Magento will automatically pull in the information

D. You create an etc/webapi.xml file and add the new field as an item to the route, as shown in the following code:

```
<route url="/V1/orders" method="GET">
    <service class="Magento\Sales\Api\OrderRepositoryInterface" method="getList">
        <item name="my_new_attribute" xsi:type="string" />
    </service>
    <resources>
        <resource ref="Magento_Sales::sales" />
    </resources>
</route>
```

Question 5

In your phtml template file you need to output a URL inside a JavaScript context.

```
var url = '<?= /* code here */ ?>';
```

Which two methods allow you to keep the output XSS-safe?

A. `escapeUrl`

B. `escapeHtmlAttr`

C. `escapeHtml`

D. `escapeJs`

Question 6

You are exploring the customer segment module and find this code in `frontend/di.xml`:

```
<type name="Magento\Framework\View\Layout">
    <plugin name="customer-segment-session-depersonalize"
            type="Magento\CustomerSegment\Model\Layout\DepersonalizePlugin" sortOrder="15"/>
</type>
```

What effect does this plugin have for customer segments?

A. It passes the segment information to the Knockout JS model from the customer session

B. It passes the segment information to the HttpContext from the customer session

C. It cleans the intermediate data of the segment-website and the segment-customer mapping in the tables to improve performance

D. It cleans the segment and customer related entities from the session

Answer Key

Answer Key

Question 1

Answers: A and D

Reference:

https://devdocs.magento.com/guides/v2.2/frontend-dev-guide/templates/template-security.html

Question 2

Answer: C

Reference:

https://www.magestore.com/magento-2-tutorial/how-to-insert-new-tab-into-customer-form-in-backend-magento-2/

Question 3

Answer: B

Question 4

Answer: B

Question 5

Answers: A and D

Reference:

https://devdocs.magento.com/guides/v2.0/frontend-dev-guide/templates/template-security.html

Question 6

Answer: D