Free Questions for B2B-Commerce-Developer by vceexamstest

Shared by Gordon on 12-12-2023

Question 1

Question Type: MultipleChoice

A developer is working on a storefront and is seeing unexpected UI behavior in one of the custom Lightning web components (LWCs) their team has built.

How should the developer investigate the issue?

Options:

A- Enable Debug Mode for a storefront user, log in to the storefront, and use Browser Inspection tools and debugger points.

B- Enable Debug Mode for a storefront user, load the LWC in Visual Studio (VS) Code, attach to session, and view debug logs in VS Code.

C- Enable debug logs for a storefront user, log in to storefront and perform action, and view debug logs in Setup.

D- Identify the user, inputs, and failure, then ask Salesforce support to investigate the issue with the custom LWC.

Answer:

A

Explanation:

To investigate the issue of seeing unexpected UI behavior in one of the custom Lightning web components (LWCs) their team has built, the developer should enable Debug Mode for a storefront user, log in to the storefront, and use Browser Inspection tools and debugger points. Debug Mode is a feature that allows developers to debug and troubleshoot custom LWCs in the storefront by disabling performance optimizations and enabling source maps. Source maps are files that map the minified or obfuscated code to the original source code, making it easier to read and debug. To enable Debug Mode for a storefront user, the developer can go to Setup, enter Users in the Quick Find box, select Users, click Edit next to the user name, and select Debug Mode. After enabling Debug Mode, the developer can log in to the storefront as the user and use Browser Inspection tools and debugger points to inspect and debug the custom LWC. Browser Inspection tools are tools that are built into web browsers that allow developers to examine and modify the HTML, CSS, JavaScript, and other aspects of a web page. Debugger points are statements that are added to the JavaScript code of a LWC that pause the execution of the code at a certain point and allow the developer to inspect the values of variables, expressions, and other elements. Enable Debug Mode for a storefront user, load the LWC in Visual Studio (VS) Code, attach to session, and view debug logs in VS Code is not a valid way to investigate the issue of seeing unexpected UI behavior in one of the custom LWCs their team has built, as it is not possible to attach to a session or view debug logs for LWCs in VS Code. Enable debug logs for a storefront user, log in to storefront and perform action, and view debug logs in Setup is not a valid way either, as debug logs do not capture information about LWCs or UI behavior. Debug logs are records of database operations, system processes, and errors that occur when executing a transaction or running unit tests. Identify the user, inputs, and failure, then ask Salesforce support to investigate the issue with the custom LWC is not a valid way either, as it is not a recommended or efficient way of debugging or troubleshooting custom LWCs. Salesforce support may not be able to provide assistance or guidance for custom LWCs that are developed by third-party developers. Salesforce Reference:B2B Commerce Developer Guide: Debug Lightning Web Components,Lightning Web Components Developer Guide: Debug Your Code,Salesforce Help: Debug Logs

Question 2

Question Type: MultipleChoice

Northern Trail Outfitters (NTO) has acquired a company and is looking to manage product data across the org seamlessly. The company has a governance policy to not install any tool or use third-party API applications to export or import the data into Salesforce. However, users have access to Salesforce CLI.

Which set of tasks must a developer perform whento export data from Salesforce or import data into Salesforce?

Options:

A- sfdx force:data:bulk:export -Product2 -all 0 and
sfdx force:data:bulk:Import -f Product2.json -all

B- sfdx force:data;tree:export -Product2 -all q and
sfdx force:data:tree:Import -f Product2.json -all

C- sfdx force:tree:data:export -q 'SELECT Id, Name FROM Product2' -u <your username> and
sfdx force:tree:data:import -f Product2Json -all

D- sfdx force:data:tree:export -q 'SELECT Id, Name FROM Product2' -u '<your username>' and sfdx force:data:tree:import -f Product2.json -u w<your username>'

Answer:

D

Explanation:

The correct answer for how to export data from Salesforce or import data into Salesforce using Salesforce CLI commands is running a command like: sfdx force:data:tree:export -q "SELECT Id, Name FROM Product2" -u "<your username>" and sfdx force:data:tree:import -f Product2.json -u "<your username>". The sfdx force:data:tree:export command is a Salesforce CLI command that exports data from an org into JSON files that conform to the SObject Tree API specification. The SObject Tree API specification is a format that defines how records are represented in JSON files for data import or export. The -q flag specifies the SOQL query that selects the records and fields to be exported. The -u flag specifies the username or alias of the org where the data will be exported from. Running this command will generate JSON files that contain the data from the org based on the SOQL query. The sfdx force:data:tree:import command is a Salesforce CLI command that imports data into an org using JSON files that conform to the SObject Tree API specification. The -f flag specifies the path of the JSON file that contains the data to be imported. The -u flag specifies the username or alias of the org where the data will be imported to. Running this command will create records in the org based on the data in the JSON file. Running a command like: sfdx force:data:bulk:export -Product2 -all 0 and sfdx force:data:bulk:import -f Product2.json -all is not a correct answer, as it uses invalid syntax and flags for the sfdx force:data:bulk:export and sfdx force:data:bulk:import commands. The correct syntax and flags for these commands are sfdx force:data:bulk:upsert -s Product2 -f Product2.csv -w 10 -u <your username> and sfdx force:data:bulk:status -i <job ID> -u <your username>. Running a command like: sfdx force:data;tree:export -Product2 -all q and sfdx force:data:tree:import -f Product2.json -all is not a correct answer either, as it uses invalid syntax and flags for the sfdx force:data:tree:export and sfdx force:data:tree:import commands. The correct syntax and flags for these commands are sfdx force:data:tree:export -q "SELECT Id, Name FROM Product2" -u <your username> and sfdx force:data:tree:import -f Product2.json -u <your username>. Running a command like: sfdx force:tree:data:export -q "SELECT Id, Name FROM Product2" -u <your username> and sfdx force:tree:data:import -f Product2Json -all is not a correct answer either, as there is no such command as sfdx force:tree:data:export or sfdx force:tree:data:import. The correct commands are sfdx force:data:tree:export and sfdx force:data:tree:import. Salesforce Reference: [Salesforce CLI Command Reference: force:data:tree:export], [Salesforce CLI Command Reference: force:data:tree:import], [Salesforce CLI Command Reference: force:data:bulk], [Salesforce Developer Tools for Visual Studio Code]

Question 3

Question Type: **MultipleChoice**

Which three data types are supported for custom fields while using CSV file format for importing data for a store?

Options:

A- Text Area(Long)

B- Picklist (Multi-Select)

C- Lookup Relationship

D- Address

E- Currency

Answer:

A, C, E

Explanation:

Three data types that are supported for custom fields while using CSV file format for importing data for a store are Text Area(Long), Lookup Relationship, and Currency. A custom field is a field that is added by a developer or an administrator to an object to store additional information or data. A data type is a property that defines the type, format, and validation rules of a field. A CSV file is a file format that stores tabular data in plain text using commas to separate values. A store is a record that represents a B2B or B2C storefront in Salesforce. Text Area(Long) is a data type that allows users to enter up to 131,072 characters on separate lines. Text Area(Long) is supported for custom fields while using CSV file format for importing data for a store. Lookup Relationship is a data type that allows users to create a relationship between two objects and select a value from another record. Lookup Relationship is supported for custom fields while using CSV file format for importing data for a store. Currency is a data type that allows users to enter currency values and automatically convert them based on the user's locale and currency settings. Currency is supported for custom fields while using CSV file format for importing data for a store. Picklist (Multi-Select) is a data type that allows users to select one or more values from a predefined list of values. Picklist (Multi-Select) is not supported for custom fields while using CSV file format for importing data for a store. Address is a data type that allows users to enter address values and automatically format them based on the user's locale settings. Address is not supported for custom fields while using CSV file format for importing data for a store. Salesforce Reference: [Salesforce Help: Custom Field Attributes], [Salesforce Help: Data Types], [Data Loader Guide: Import Data into Salesforce], [B2B Commerce Developer Guide: Store Object]

Question 4

Question Type: MultipleChoice

A developer has the task to bring some historical data into an org. The data must reside in the org, but cannot be populated in standard or custom objects. The customer is fine with developers building UI components to surface this data on a case-by-case basis.

Which option allows a developer to meet all of these requirements?

Options:

A- Big objects

B- Lightning Canvas

C- External objects

D- Lightning Out

Answer:

A

Explanation:

To bring some historical data into an org, the data must reside in the org, but cannot be populated in standard or custom objects, and the customer is fine with developers building UI components to surface this data on a case-by-case basis, the option that allows a developer to meet all of these requirements is big objects. Big objects are a type of object that can store and manage massive amounts of data on the Salesforce platform. Big objects can store up to billions of records and are accessible through a subset of SOQL or custom user interfaces. Big objects are not subject to the same storage limits or performance issues as standard or custom objects and are suitable for storing historical or archived data that does not need to be updated frequently. Big objects can be defined using Metadata API or declaratively in Setup. Lightning Canvas is not an option that allows a developer to meet all of these requirements, as it is a framework that allows developers to integrate third-party applications into Salesforce. Lightning Canvas does not store data in the org, but rather displays data from external sources using an iframe. External objects are not an option either, as they are a type of object that map to data stored outside Salesforce. External objects do not store data in the org, but rather access data from external systems using OData services. Lightning Out is not an option either, as it is a feature that allows developers to use Lightning components outside Salesforce. Lightning Out does not store data in the org, but rather renders components on external web pages or applications. Salesforce Reference:Salesforce Help: Define Big Objects,Salesforce Help: Lightning Canvas Developer's Guide,Salesforce Help: External Objects,Salesforce Developer Blog: Lightning Out

Question 5

Question Type: **MultipleChoice**

A developer needs to import some new product data contained in a JSON file one time. What are two viable ways to do this? .

Options:

A- Convert the JSON to an xlsx file and use Workbench to import it

B- Run a command like: sfdx force:data:tree:import -f NewProducts.json -u <your username>

C- Convert the JSON to a CSV file and use Data Loader to import it

D- Run a command like: sfdx force:data;import:bulk -f NewProducts.json -u <your username>

Answer:

B, C

Explanation:

Two viable ways that a developer can import some new product data contained in a JSON file one time are running a command like: sfdx force:data:tree:import -f NewProducts.json -u <your username> and converting the JSON to a CSV file and using Data Loader to import it. Running a command like: sfdx force:data:tree:import -f NewProducts.json -u <your username> allows the developer to import data from a JSON file into an org using Salesforce CLI commands. The sfdx force:data:tree:import command is a Salesforce CLI command that imports data into an org using JSON files that conform to the SObject Tree API specification. The SObject Tree API specification is a format that defines how records are represented in JSON files for data import or export. The -f flag specifies the path of the JSON file that contains the data to be imported. The -u flag specifies the username or alias of the org where the data will be imported. Running this command will create records in the org based on the data in the JSON file. Converting the JSON to a CSV file and using Data Loader to import it allows the developer to import data from a CSV file into an org using Data Loader. Data Loader is a tool that allows users to import or export data between Salesforce and CSV files. The developer can use an online converter or a spreadsheet application to convert their JSON file into a CSV file that matches the structure and format of their Salesforce object. The developer can then use Data Loader to import the CSV file into their org and create records based on the data in the CSV file.
Converting the JSON to an xlsx file and using Workbench to import it is not a viable way to import some new product data contained in a JSON file one time, as Workbench does not support xlsx files for data import or export. Workbench is a web-based tool that provides access to various Salesforce features and functionalities, such as data manipulation, REST Explorer, and Apex Execute. Running a command like: sfdx force:data;import:bulk -f NewProducts.json -u <your username> is not a viable way either, as there is no such command as sfdx force:data;import:bulk. The correct command for importing data using bulk API is sfdx force:data:bulk:upsert.
Salesforce Reference: [Salesforce CLI Command Reference: force:data:tree:import], [Salesforce Developer Tools for Visual Studio Code], [Data Loader Guide: Import Data into Salesforce], [Workbench], [Salesforce CLI Command Reference: force:data:bulk:upsert]

Question 6

Question Type: MultipleChoice

What is true about mapping custom fields from Cart to Order Summary?

Options:

A- A custom field must exist in the Cart and Order Summary objects only to be mapped successfully.

B- The automatic Cart to Order mapping of custom fields can be disabled.

C- All data types are supported for custom fields to be mapped from Cart to Order.

D- There is a limit of 25 custom fields on a Cart that can be mapped to Order.

Answer:

B

Explanation:

The correct answer for what is true about mapping custom fields from Cart to Order Summary is that the automatic Cart to Order mapping of custom fields can be disabled. A custom field is a field that is added by a developer or an administrator to an object to store additional information or data. A Cart is an object that represents a collection of products and charges that a customer intends to purchase in the storefront. An Order Summary is an object that represents a confirmed purchase of products and charges by a customer in the storefront. A Cart can be converted to an Order Summary when the customer completes the checkout process and confirms their order. By default, Salesforce B2B Commerce automatically maps custom fields from Cart to Order Summary when converting a Cart to an Order Summary. This means that any custom fields that exist on both Cart and Order Summary objects with identical API names and data types will have their values copied from Cart to Order Summary during the conversion. The automatic Cart to Order mapping of custom fields can be disabled by setting the B2BCommerce.CartToOrderMappingEnabled custom setting to false. This will prevent any custom fields from being copied from Cart to Order Summary during the conversion. A custom field must exist in the Cart and Order Summary objects only to be mapped successfully is not true, as it is not the only requirement for mapping custom fields from Cart to Order Summary. The custom fields must also have identical API names and data types, and the automatic Cart to Order mapping of custom fields must be enabled. All data types are supported for custom fields to be mapped from Cart to Order is not true, as some data types are not supported for mapping custom fields from Cart to Order Summary. The supported data types are Boolean, Date, DateTime, Double, Integer, Long, Percent, String, and TextArea. There is a limit of 25 custom fields on a Cart that can be mapped to Order is not true, as there is no such limit for mapping custom fields from Cart to Order Summary. Any number of custom fields that meet the mapping requirements can be mapped from Cart to Order Summary. Salesforce Reference: [B2B Commerce Developer Guide: Custom Field Mapping], [B2B Commerce Developer Guide: Cart Object], [B2B Commerce Developer Guide: Order Summary Object]