

Salesforce.B2B-Commerce-Developer.v2024-08-20.q191

Exam Code:	B2B-Commerce-Developer
Exam Name:	Salesforce Accredited B2B Commerce Developer
Certification Provider:	Salesforce
Free Question Number:	191
Version:	v2024-08-20
# of views:	659
# of Questions views:	1910
https://www.freepdfdumps.com/Salesforce.B2B-Commerce-Developer.v2024-08-20.q191.html	

NEW QUESTION: 1

Which three attributes are true regarding Subscriber pages? (3 answers)

- A. Subscriber pages can include additional standard Salesforce B2B Commerce components such as featured products, category tree, and mini cart.
- B. All the user interface components must be created manually.
- C. Subscriber pages allow customers to quickly create new custom pages for their storefront.
- D. Out of the Box, Subscriber Pages are completely blank HTML pages.
- E. Standard Salesforce B2B Commerce components are automatically included on the page e.g. Header links, images, menu items, containers, etc.

Answer: A,C,D (LEAVE A REPLY)

Three attributes that are true regarding Subscriber Pages are:

- * Subscriber Pages can include additional standard Salesforce B2B Commerce components such as featured products, category tree, and mini cart. These components can be added to the Subscriber Pages by using the CCRZ.pagevars.pageLabels and CCRZ.pagevars.pageSections objects, which contain the page labels and page sections that are defined in CCAdmin.
- * Subscriber Pages allow customers to quickly create new custom pages for their storefront. These pages can be used to display any content or functionality that is not available in the default pages, such as FAQs, testimonials, or promotions.
- * Out of the Box, Subscriber Pages are completely blank HTML pages. These pages do not have any predefined layout or content, so the user can customize them according to their requirements. The user can use Visualforce, HTML, CSS, JavaScript, or any other web technologies to create their own Subscriber Pages. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Subscriber Pages

NEW QUESTION: 2

Which two Salesforce B2B Commerce visualforce pages must be enabled at a Salesforce Community level to make the out of the box SEO functionality available? (2 answers)

- A. CCSizeIndex
- B. SizeMap
- C. CCCatSiteMap
- D. ProductMap

Answer: (SHOW ANSWER)

Two Salesforce B2B Commerce Visualforce pages that must be enabled at a Salesforce Community level to make the out of the box SEO functionality available are:

CCSizeIndex: This page generates a sitemap.xml file, which is a file that lists all the pages and resources on a site that can be crawled by web crawlers. The page uses the configuration settings CO.SiteMapIncludeProducts and CO.SiteMapIncludeCategories to specify which products and categories should be included in the sitemap.

CCCatSiteMap: This page generates a category sitemap file, which is a file that lists all the categories on a site that can be crawled by web crawlers. The page uses the configuration setting CO.SiteMapCategoryDepth to specify how many levels of subcategories should be included in the category sitemap. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Sitemap Files

NEW QUESTION: 3

Which two components in a B2B store template should a developer use to customize a storefront page?

- A. My Lists
- B. Product List
- C. Order List
- D. Address List

Answer: (SHOW ANSWER)

Explanation

Two components in a B2B store template that a developer can use to customize a storefront page are My Lists and Product List. My Lists is a standard component that displays and manages lists of products that customers can create, save, or share in the storefront. A developer can use this component to enable customers to organize their favorite or frequently purchased products into lists and access them easily. Product List is a standard component that displays and filters products from one or more product lists in the storefront. A developer can use this component to showcase products from different categories or collections and allow customers to browse or search for products based on various criteria. Order List is not a component in a B2B store template, but rather an object that stores information about orders placed by customers in the storefront.

Address List is not a component either, but rather an object that stores information about addresses associated with customers in the storefront. Salesforce References: [B2B Commerce Developer Guide: My Lists Component], [B2B Commerce Developer Guide: Product List Component], [B2B Commerce Developer Guide: Order List Object], [B2B Commerce Developer Guide: Address List Object]

NEW QUESTION: 4

A developer is working on an existing checkout implementation built against the Lightning Web Runtime (LWR) and wants to implement a custom child checkout component to modify out-of-the-box functionality.

Which interface must the developer implement for the child component?

- A. CheckoutSavable
- B. Checkoutoutinterface
- C. CustomCheckout
- D. CheckoutStep

Answer: D ([LEAVE A REPLY](#))

Explanation

To implement a custom child checkout component for a B2B or B2C store created with an LWR template, the developer must implement the CheckoutStep interface from the commerce/checkout module. This interface defines the methods and properties that the child component must have to communicate with the parent checkout component and the checkout data provider. The CheckoutStep interface extends the UseCheckoutComponent interface, which provides the common functionality for all checkout components, such as loading and saving data, reporting validity, and handling errors.

The other options are not correct because:

- * A. CheckoutSavable is not a valid interface for a custom child checkout component. There is no such interface in the commerce/checkout module or the Lightning web component framework.
- * B. Checkoutoutinterface is not a valid interface for a custom child checkout component. There is no such interface in the commerce/checkout module or the Lightning web component framework.
- * C. CustomCheckout is not a valid interface for a custom child checkout component. There is no such interface in the commerce/checkout module or the Lightning web component framework.

References:

- * Create a Custom Checkout Component for a B2B or B2C Store (LWR)
- * CheckoutStep Interface
- * UseCheckoutComponent Interface

NEW QUESTION: 5

Which Global JavaScript Object should be extended when writing custom Remote Actions?

- A. CCRZ.
- B. CCRZ.cc
- C. CCRZ.cc_CallContext
- D. CCRZ.RemoteInvocation

Answer: B ([LEAVE A REPLY](#))

The Global JavaScript Object that should be extended when writing custom Remote Actions is CCRZ.cc. This object contains all the Remote Actions that are defined in the cloudcraze managed

package, which can be overridden or extended by subscriber code. The object also provides a mechanism for registering custom Remote Actions that can be invoked by the user interface components. For example, `CCRZ.cc.customAction = function(params, callback){ // do something }` will define a custom Remote Action named `customAction` that can be called by `CCRZ.cc.customAction(params, callback)`. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Remote Actions

NEW QUESTION: 6

Which two behaviors does a target value of `lightning__FlowScreen` in metadata allow for a Lightning web component?

- A. It allows the Lightning web component to replace standard functionality in flows and subflows
- B. It allows the Lightning Web component to be dragged onto a page in Lightning AppBuilder
- C. It allows the Lightning web component to be used in guided user experiences to gather input
- D. It automatically generates configuration properties for the Lightning web component

Answer: [\(SHOW ANSWER\)](#)

A target value of `lightning__FlowScreen` in metadata allows for a Lightning web component to be used in guided user experiences to gather input and to automatically generate configuration properties for the Lightning web component. The `lightning__FlowScreen` target value specifies that the component can be used on a flow screen, which is a type of flow element that displays information and collects user input in a flow. A flow is a guided user experience that can automate business processes and guide users through screens, actions, and decisions. By using the `lightning__FlowScreen` target value, the developer can expose their Lightning web component as a custom screen component that can be added to any flow screen. The developer can also use the `@api` decorator to expose public properties of their Lightning web component as configuration properties that can be set in Flow Builder. Flow Builder is a tool that allows the developer to create and modify flows using a drag-and-drop interface. A target value of `lightning__FlowScreen` does not allow the Lightning web component to replace standard functionality in flows and subflows or to be dragged onto a page in Lightning App Builder, as these are not related to the flow screen target value. Salesforce Reference: Lightning Web Components Developer Guide: Configure Components for Flows, Lightning Web Components Developer Guide: Communicate with Flow, Lightning Flow Developer Guide: Screen Components Basics

NEW QUESTION: 7

What are two common and maintainable ways the content layout of a Lightning Web Component can be implemented?

- A. Spreading layout styles across several separate components
- B. Styling the `:host` pseudo-element (or other elements) via the CSS file
- C. Using inline styles
- D. Applying SLDS classes to internal elements

Answer: B,D [\(LEAVE A REPLY\)](#)

For maintainable and scalable Lightning Web Component development, it's recommended to style components by targeting the `:host` pseudo-element in the component's CSS file and by applying Salesforce Lightning Design System (SLDS) classes to internal elements. These practices ensure consistent styling that aligns with Salesforce's design standards and provides a clear separation of concerns between component structure and styling.

NEW QUESTION: 8

Which two different types of services do Salesforce B2B

- A. Data services which are responsible for querying and transforming data from entities
- B. Global services which are responsible for querying and transforming data from entities
- C. Commerce APIs leverage? (2 answers) A) Logic services which are responsible for implementing business logic associated with entities.
- D. Hook services which are extension points for subscribers to tie into.

Answer: A,C ([LEAVE A REPLY](#))

NEW QUESTION: 9

Numerous flags when set, have a direct impact on the result set provided by the Global API's. Which conversion flag allows for sObjects to be returned from the Global API's when provided as a Boolean parameter with a value of true?

- A. `ccrz.ccAPISizing.SKIPTRZ`
- B. `ccrz.ccAPISizing.SOBJECT`
- C. `ccrz.ccAPI.SZ_SKIPTRZ`
- D. `ccrz.ccAPI.SZ_SOBJECT`

Answer: ([SHOW ANSWER](#))

Explanation

The conversion flag that allows for sObjects to be returned from the Global API's when provided as a Boolean parameter with a value of true is `ccrz.ccAPI.SZ_SOBJECT`. This flag indicates that the API should return the raw sObjects instead of the transformed objects that are usually returned by the API. For example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_SOBJECT,true)` will return the Product2 sObjects instead of the `ccrz__E_Product__c` objects. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Data Sizing Conventions

NEW QUESTION: 10

A developer has written the logic to import products from an enterprise resource pi products are in Salesforce, but they are not visible in the store.

What did the developer forget to assign to the imported products?

- A. Entitlement policy
- B. Storefront
- C. Account
- D. Promotion

Answer: (SHOW ANSWER)

Explanation

A developer has written the logic to import products from an enterprise resource planning (ERP) platform into B2B storefront products. The imported products are in Salesforce, but they are not visible in the store. The developer forgot to assign the storefront to the imported products. A storefront is a logical grouping of products, categories, and price books that defines what products are available for purchase in a B2B Commerce site¹. A product must be associated with at least one storefront to be visible in the store². The developer can use the Data Loader to insert or update the storefront assignments for the imported products³.

Option A, Option C, and Option D are incorrect because they are not required for the products to be visible in the store. An entitlement policy is a set of rules that determines the availability of products based on the buyer's account, contract, or order history⁴. An account is a record that represents a business or person involved in a business transaction. A promotion is a marketing tool that offers discounts or incentives to buyers. These are optional features that can be used to customize the B2B Commerce experience, but they are not necessary for the products to be visible in the store. References:

- * Storefronts
- * Product Visibility
- * Insert new Products and custom Price Books via Data Loader
- * Entitlement Policies
- * [Accounts]
- * [Promotions]

NEW QUESTION: 11

How can a developer establish communication between components that are not in the same DOM (Document Object Model) tree?

- A. Use publish-subscribe pattern.
- B. Configure targets property.
- C. Use dispatch events.
- D. Use @api decorators.

Answer: (SHOW ANSWER)

To establish communication between components that are not in the same DOM (Document Object Model) tree, a developer can use the publish-subscribe pattern. The publish-subscribe pattern is a messaging pattern that allows components to communicate with each other without being directly connected or aware of each other. The components can publish events to a common channel and subscribe to events from that channel. The channel acts as a mediator that delivers the events to the subscribers. The developer can use a custom library or a Salesforce platform service, such as Lightning Message Service or Platform Events, to implement the publish-subscribe pattern. Configuring the targets property is not a way to communicate between components that are not in the same DOM tree, as it only defines where a component can be used in an app. Using dispatch events is not a way either, as it only works for components that

are in the same DOM tree or have a parent-child relationship. Using @api decorators is not a way either, as it only exposes public properties or methods of a component to other components that use it. Salesforce Reference: Lightning Web Components Developer Guide: Communicate Across Salesforce UI Technologies, Lightning Web Components Developer Guide: Communicate with Events, [Lightning Web Components Developer Guide: Communicate with Properties]

NEW QUESTION: 12

How does a project implement the process to persist payment information data in the Checkout flow for Salesforce B2B Commerce version 4.2 and beyond?

- A. Trigger a remote action when the process payment button is selected to capture the payment.
- B. Trigger a remote action to store the payment information in the URL query parameters.
- C. Trigger the processPayment event and pass in the payment information object as an argument.
- D. Trigger the externalProcessedPayment and pass in the payment information object as an argument.

Answer: (SHOW ANSWER)

Explanation

To persist payment information data in the Checkout flow for Salesforce B2B Commerce version 4.2 and beyond, the project needs to trigger the processPayment event and pass in the payment information object as an argument. This event will invoke the processPayment method of the ccServicePayment class, which will validate and process the payment information and return a payment result object. The payment result object will contain the status and details of the payment transaction.

NEW QUESTION: 13

Which two blocks of code are needed to implement a custom getter in a Lightning web component?

```
set rows(value) {  
  this.state.rows = value;  
}
```

A.

```
get rows() {  
  return this.state.rows;  
}
```

B.

```

@api
set rows(value){
this.state.rows = value;
}

```

C.

```

@api
get rows() {
return this.state.rows;
}

```

D.

Answer: (SHOW ANSWER)

Explanation

custom getter in a Lightning web component is a JavaScript function that executes logic each time a public property is accessed. A custom getter must start with the keyword `get` and be followed by a name for the property. A custom getter must also have a corresponding custom setter, which is a function that executes logic each time a public property is assigned a value. A custom setter must start with the keyword `set` and have the same name as the getter. One of the getter or setter functions must be annotated with `@api`, which makes the property public and reactive.

Option A and Option D show the correct syntax for defining a custom getter and setter for a public property called `name`. Option A shows the getter function, which returns the value of a private property called `_name`.

Option D shows the setter function, which assigns the value of the parameter `value` to the private property `_name`. The getter function is annotated with `@api`, which makes the `name` property public and reactive.

Option B and Option C are incorrect because they do not follow the syntax for a custom getter and setter.

Option B shows a regular function declaration, not a getter function. Option C shows a regular assignment statement, not a setter function. Neither option uses the `@api` decorator, which is required for a public property. References:

* Getters and Setters

* Understand getter in Lightning Web component

NEW QUESTION: 14

How is a price group dynamically set?

A. By overriding the `ccLogicProductPrice` class

- B. By using contract pricing
- C. By extending the ccApiPriceList API
- D. By extending the cc_hk_pricing hook

Answer: D ([LEAVE A REPLY](#))

Explanation

A price group can be dynamically set by extending the cc_hk_pricing hook. This hook allows modifying the price group based on various factors, such as the user, cart, product, or storefront. For example, the hook can apply a different price group for a specific product category or for a specific user segment.

NEW QUESTION: 15

Which event should be triggered when user facing info, warning or error messages need to be displayed on a Visualforce page?

- A. showMessage
- B. displayPageMessage
- C. displayMessage
- D. pageMessage

Answer: D ([LEAVE A REPLY](#))

Explanation

The event that should be triggered when user facing info, warning or error messages need to be displayed on a Visualforce page is pageMessage. This event is triggered by the displayPageMessage method of the CCRZ.util class, which is a utility class that provides various helper methods for common tasks. The event can be used to display a message on the page with a specified type, title, and text. For example, CCRZ.util.displayPageMessage('error', 'Oops!', 'Something went wrong.') will trigger the pageMessage event and display an error message on the page. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Events, Util Class

NEW QUESTION: 16

Which two are considered discrete units of work (code units) within a transaction in the debug logs?

- A. Validation rule
- B. Lightning component load
- C. Web service invocation
- D. Apex class

Answer: ([SHOW ANSWER](#))

Explanation

Two data types that are considered discrete units of work (code units) within a transaction in the debug logs are web service invocation and Apex class. A discrete unit of work (code unit) is a segment of executable code that runs as part of a transaction in Salesforce. A transaction is a sequence of operations that are treated as a single unit of work and are executed under certain

isolation and consistency rules. A transaction can consist of one or more discrete units of work (code units) that are executed sequentially or concurrently depending on various factors such as triggers, asynchronous calls, or limits. A debug log is a record of database operations, system processes, and errors that occur when executing a transaction or running unit tests in Salesforce. A debug log can capture information about each discrete unit of work (code unit) within a transaction, such as its start time, end time, duration, events, variables, and limits. A web service invocation is a type of discrete unit of work (code unit) that involves calling an external web service from Apex code using SOAP or REST protocols. A web service invocation can be synchronous or asynchronous depending on the method used to make the callout. A web service invocation can be captured in a debug log with its details and results. An Apex class is another type of discrete unit of work (code unit) that involves executing Apex code that defines a class with properties and methods. An Apex class can be invoked from various sources such as triggers, Visualforce pages, Lightning components, or API calls. An Apex class can be captured in a debug log with its details and results. A validation rule is not a type of discrete unit of work (code unit) within a transaction in the debug logs, as it is not a segment of executable code but rather a formula expression that defines a business rule for a field or object. A validation rule can be evaluated during a transaction and cause an error if the rule condition is not met, but it cannot be captured as a separate code unit in a debug log. A Lightning component load is not a type of discrete unit of work (code unit) within a transaction in the debug logs either, as it is not a segment of executable code but rather an event that occurs when a Lightning component is rendered on a web page or application. A Lightning component load can be measured by various performance tools such as Lightning Inspector or Lighthouse, but it cannot be captured as a separate code unit in a debug log. Salesforce References: [Salesforce Developer Blog: Transactions and Request Processing], [Salesforce Help: Debug Logs], [Salesforce Developer Guide: Invoking Callouts Using Apex], [Salesforce Developer Guide: Apex Classes], [Salesforce Help: Validation Rules], [Salesforce Developer Blog: Measuring Lightning Component Performance]

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 17

What are two purposes of the Shadow DOM in a Lightning web component?

A. It encapsulates the internal document object model (DOM) structure of a web component

- B. It allow components to be shared while protecting them from being manipulated by arbitrary code
- C. It allows direct access to the document object model of the component
- D. It allows older JavaScript libraries to manipulate the tagging structure

Answer: (SHOW ANSWER)

According to the Lightning Web Components Developer Guide, Shadow DOM is a standard that encapsulates the internal document object model (DOM) structure of a web component. Encapsulating the DOM gives developers the ability to share a component and protect the component from being manipulated by arbitrary HTML, CSS, and JavaScript. Shadow DOM also provides style and behavior isolation for a web component, which means that the styles and scripts defined inside a component do not affect the rest of the page, and vice versa. Therefore, the purposes of the Shadow DOM in a Lightning web component are to encapsulate the internal DOM structure of a web component (A) and to allow components to be shared while protecting them from being manipulated by arbitrary code (B). The other options are incorrect because Shadow DOM does not allow direct access to the document object model of the component , nor does it allow older JavaScript libraries to manipulate the tagging structure (D). References: Lightning Web Components Developer Guide, Understand the Shadow DOM Unit

NEW QUESTION: 18

A query containing a subquery is executed. What is appended to the subquery name as part of its transformation by default in Salesforce B2B Commerce?

- A. A subscriber-supplied token
- B. "__ccrz"
- C. The "*" symbol
- D. The letter "S"

Answer: B (LEAVE A REPLY)

A query containing a subquery is executed. By default, in Salesforce B2B Commerce, "__ccrz" is appended to the subquery name as part of its transformation. This is done to avoid conflicts with the standard Salesforce fields and relationships. For example, SELECT Id, Name, (SELECT Id, Name FROM Contacts) FROM Account will be transformed to SELECT Id, Name, (SELECT Id, Name FROM Contacts__ccrz) FROM Account__ccrz. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 19

What are three ways to implement custom post Order processing? (3 answers)

- A. Use a Salesforce workflow rule that executes when an Order record is created.
- B. Extend cc_hk_invoice to handle custom business logic post Order processing
- C. Use cc_hk_Order.placeTarget to define a new Order Confirmation page which executes additional business logic.
- D. Modify or add custom Cart formula fields to handle logic.

E. Use Process builder to implement business processes that execute when an Order record is created.

Answer: B,C,E (LEAVE A REPLY)

Three ways to implement custom post Order processing are:

Extend `cc_hk_invoice` to handle custom business logic post Order processing. This hook allows modifying the invoice data or performing additional actions after an Order is placed. For example, the hook can send an email notification or update a custom field on the invoice record.

Use `cc_hk_Order.placeTarget` to define a new Order Confirmation page which executes additional business logic. This hook allows specifying a custom Visualforce page that will be displayed after an Order is placed. The custom page can include additional business logic or user interface elements.

Use Process Builder to implement business processes that execute when an Order record is created. Process Builder is a tool that allows creating workflows and actions based on criteria and conditions. For example, Process Builder can create a task, send an email, or update a record when an Order record is created. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Hooks, Process Builder

NEW QUESTION: 20

What are two guidelines for logging that are used within the core Salesforce B2B Commerce product? (2 answers)

- A. It is okay to log any data on the server that is already logged on the client side.
- B. The close method of `ccrz.ccLog` must be called at the end of the remote action.
- C. No calls to `ccrz.ccLog` can be made before `cc_CallContext.initRemoteContext` is executed.
- D. Items or data within computational intensive loops should be logged.

Answer: B,C (LEAVE A REPLY)

NEW QUESTION: 21

The sizing keys used in the Salesforce B2B Commerce Global APIs five distinct operations. What are three of these operations? (3 answers)

- A. Refetch data (used on some Logic classes)
- B. Return formats as `Map<String, Object>` or `SObjects` lists
- C. Override static DAO classes and methods
- D. Related Query to call (sub queries or direct queries)
- E. Object type casting

Answer: A,D,E (LEAVE A REPLY)

The sizing keys used in the Salesforce B2B Commerce Global APIs perform five distinct operations. Three of these operations are:

Refetch data (used on some Logic classes): This operation indicates that the data should be refetched from the database instead of using the cached data. For example, `ccrz.ccServiceCart.getCart(ccrz.ccAPI.SZ_REFETCH)` will refetch the cart data and refresh the cache.

Related Query to call (sub queries or direct queries): This operation indicates that the related entities should be retrieved by using sub queries or direct queries. For example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_SUBQUERY)` will use sub queries to fetch the related entities for each product, while `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_DIRECTQUERY)` will use direct queries to fetch the related entities separately.

Object type casting: This operation indicates that the data should be cast to a specific object type. For example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_SOBJECT)` will cast the data to `sObjects` instead of transformed objects. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Data Sizing Conventions

NEW QUESTION: 22

Which static method invocation is used to initialize `ccrz.cc_CallContext` with information from `ccrz.cc_RemoteActionContext` and return an instance of `ccrz.cc_RemoteActionResult` in an apex `@RemoteAction` method?

- A. `ccrz.cc_CallContext.init(ccrz.cc_RemoteActionContext)`
- B. `ccrz.cc_CallContext.initCallContext(ccrz.cc_RemoteActionContext)`
- C. `ccrz.cc_CallContext.initRemoteActionContext(ccrz.cc_RemoteActionContext)`
- D. `ccrz.cc_CallContext.initializeCallContext(ccrz.cc_RemoteActionContext)`

Answer: (SHOW ANSWER)

The static method invocation that is used to initialize `ccrz.cc_CallContext` with information from `ccrz.cc_RemoteActionContext` and return an instance of `ccrz.cc_RemoteActionResult` in an Apex `@RemoteAction` method is `ccrz.cc_CallContext.initCallContext(ccrz.cc_RemoteActionContext)`. This method takes in a `ccrz.cc_RemoteActionContext` object as a parameter and returns a `ccrz.cc_RemoteActionResult` object. The `ccrz.cc_RemoteActionContext` object contains information about the current storefront, user, cart, price list, currency, locale, and session. The `ccrz.cc_RemoteActionResult` object contains information about the status, result, and errors of the remote action. The other methods are not valid or do not exist. Salesforce Reference: B2B Commerce Developer Guide: `cc_CallContext` Class, B2B Commerce Developer Guide: `cc_RemoteActionContext` Class, B2B Commerce Developer Guide: `cc_RemoteActionResult` Class

NEW QUESTION: 23

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?

- A. Big objects
- B. Lightning Canvas
- C. External objects
- D. Lightning Out

Answer: (SHOW ANSWER)

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 References: Salesforce Help: Define BigObjects, Salesforce Help: Lightning Canvas Developer's Guide, Salesforce Help: External Objects, Salesforce Developer Blog: Lightning Out

NEW QUESTION: 24

What is one requirement to keep in mind when including additional JavaScript files in a Lightning Web Component?

- A. The files must be ES6 modules and must have names that are unique within the component's folder.
- B. Only five of the files can be used with an import statement
- C. All the files must be imported to a singleton.js file and the singleton.js file can be used with an import statement
- D. Only one of the files can be used with an import statement

Answer: (SHOW ANSWER)

When including additional JavaScript files in a Lightning Web Component, it is required that these files are ECMAScript 6 (ES6) modules and have unique names within the component's folder. This ensures proper module resolution and avoids namespace conflicts. Salesforce LWC documentation provides guidelines on organizing component resources, including JavaScript modules, to ensure they are correctly recognized and utilized within the LWC framework.

NEW QUESTION: 25

What is default behavior for how the Salesforce B2B Commerce Global APIs transform Salesforce data?

- A. Fields names can be mapped to any naming convention desired

- B. Fields names are returned with "c." prepended in their name.
- C. Fields names are returned with a lowercase first letter, camelcase convention
- D. Fields names are returned using the Salesforce naming convention.

Answer: C ([LEAVE A REPLY](#))

NEW QUESTION: 26

A developer is trying to troubleshoot why a field is not displaying on the Product Detail Page. What should be typed in the Developer Tools Console in the browser to view the fields available for the Product Detail Page?

- A. CCRZ.productSearchView
- B. CCRZ.cartView
- C. CCRZ.productDetailModel
- D. CCRZ.productDetailView

Answer: C ([LEAVE A REPLY](#))

Explanation

To view the fields available for the Product Detail Page, the developer should type CCRZ.productDetailModel in the Developer Tools Console in the browser. This will display the product detail model object, which contains the product data and attributes that are rendered on the page. The other options are either not valid or not relevant for the Product Detail Page.

NEW QUESTION: 27

What is the difference between Gross Layout Overrides and Subscriber Templates?

- A. Gross Layout Overrides allow modification to CSS of a page, while Subscriber Templates allows for modification of the entire page including header and footer.
- B. Subscriber Templates allows for modification of the header, the footer and the content in between them.

Gross Layout Overrides only allow for modification of the header and footer.

- C. Subscriber Templates allow for modification of the header and the footer, while Gross Layout Overrides allow for modification everything inside the header and footer.
- D. Gross Layout Overrides allow for the modification of the footer, while Subscriber Templates allow for modification of everything inside the header and footer.

Answer: C ([LEAVE A REPLY](#))

The difference between Gross Layout Overrides and Subscriber Templates is that Subscriber Templates allow for modification of the header and the footer, while Gross Layout Overrides allow for modification everything inside the header and footer. Subscriber Templates are custom Visualforce pages that can replace the standard header and footer components on a page. Gross Layout Overrides are custom Visualforce pages that can replace the default layout of a page, such as the number of columns, the position of widgets, or the content of sections. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Subscriber Templates, Gross Layout Overrides

NEW QUESTION: 28

Which two aspects are applicable to Page Includes? (2 answers)

- A. Standard Visualforce controls such as apex:form should not be used within a page include
- B. If a controller is used for an included page, then a merge variable must be present on the page.
- C. Page Includes can be configured as Body Includes Begin.
- D. Page Includes must be assigned to an OOTB Page, i.e. Home, Product Detail, etc., and enabled

Answer: A,B (LEAVE A REPLY)

NEW QUESTION: 29

How can a developer make an integration available for selection?

- A. Modify the StoreIntegrated Service to map to an Apex class id using workbench
- B. Enter the integration class name and version in Store Administration
- C. Create a RegisteredExternalService record using Workbench
- D. The integration is available once it is uploaded

Answer: C (LEAVE A REPLY)

To make an integration available for selection in Salesforce B2B Commerce, a developer must create a RegisteredExternalService record. This can be accomplished using tools like Workbench, which allow for direct interaction with Salesforce's backend database. By registering the external service, the integration becomes selectable within the B2B Commerce setup, allowing it to be associated with specific stores or contexts. Salesforce documentation on integrating external services with B2B Commerce provides step-by-step instructions on registering these services to enable seamless integration and functionality within the platform.

NEW QUESTION: 30

The ccUtil apex class in Salesforce B2B Commerce provides numerous utility functions that can be leveraged in subscriber classes.

What are two ways to check the input or return data of the Global API's? (2 answers)

- A. `ccrz.ccUtil.isEmpty(Map<String, Object>)` and `ccrz.ccUtil.isEmpty(List<Object>)`
- B. `ccrz.ccUtil.isValid(Map<String, Object>)` and `ccrz.ccUtil.isValid(List<Object>)`
- C. `ccrz.ccUtil.isValid(Map<String, Object>)` and `ccrz.ccUtil.isValid(List<Object>)`
- D. `ccrz.ccUtil.isEmpty(Map<String, Object>)` and `ccrz.ccUtil.isEmpty(List<Object>)`

Answer: A,D (LEAVE A REPLY)

Explanation

The ccUtil apex class provides two methods to check the input or return data of the Global API's: `ccrz.ccUtil.isEmpty(Map<String, Object>)` and `ccrz.ccUtil.isEmpty(Map<String, Object>)`.

These methods return true if the map is not null and contains at least one entry, or if the map is null or empty, respectively.

Similarly, `ccrz.ccUtil.isEmpty(List<Object>)` and `ccrz.ccUtil.isEmpty(List<Object>)` return true if the list is not null and contains at least one element, or if the list is null or empty, respectively.

These methods are useful for validating the input parameters or the output results of the Global API's.

NEW QUESTION: 31

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

- A. Text Area(Long)
- B. Picklist (Multi-Select)
- C. Lookup Relationship
- D. Address
- E. Currency

Answer: A,C,E (LEAVE A REPLY)

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]

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: <https://www.actual4test.com/B2B->

NEW QUESTION: 32

A developer has created a custom Lightning web component to display on the Product Detail page in the store.

When the developer goes to add the component to the page in Experience Builder, it is missing from the list of custom components.

Which XML fragment should the developer include in the component's configuration XML file to ensure the custom component is available to add to the page?

```
<isExposed>true</isExposedTrue>
<targets>
<target>lightningCommunity_Page</target>
<targets>
```

A.

```
<builder>ExperienceCloud</builder>
<target>RecordPage</target>
```

B.

```
<isExposed target="ExperienceCloud">
<pageType>RecordPage</pageType>
</isExposed>
```

C.

```
<isAvailable>true</isAvailable>
<targets>lightningCommunity_RecordPage<targets>
```

D.

Answer: B (LEAVE A REPLY)

Explanation

The XML fragment that the developer should include in the component's configuration XML file to ensure the custom component is available to add to the page is option B. Option B uses the <targets> tag to specify where the component can be used in an app. The <targets> tag contains a list of <target> tags, each of which defines a valid target for the component. The value of the <target> tag for the Product Detail page in the store is lightningCommunity__RecordPage, which means that the component can be used on any record page in a Lightning community. Option A is incorrect because it uses an invalid value for the <target> tag. There is no such target as lightningCommunity__ProductDetailPage. Option C is incorrect because it uses an invalid syntax for the <targets> tag. The <targets> tag should not have any attributes, such as isAvailable.

Option D is incorrect because it uses an invalid syntax for the <target> tag. The <target> tag should not be self-closing, but rather have a closing tag (</target>). Salesforce References: Lightning Web Components Developer Guide: Configure Components for Lightning Communities, Lightning Web Components Developer Guide: Configure Components for Different Pages and Apps

NEW QUESTION: 33

A developer is working in Visual Studio Code on a previously deployed project which is rather large and deployments are time consuming. The developer wants to know if a CSS file containing small changes was actually deployed to the org. What is one way this can be accomplished?

- A. Right-click the CSS file and choose Diff File Against Org
- B. Click the Tools menu and select Diff Styles Against Org...
- C. Right-click the folder for the component and choose Diff Styles Against Org
- D. Right-click the folder for the component and choose Diff Files Against Org

Answer: A (LEAVE A REPLY)

To know if a CSS file containing small changes was actually deployed to the org, one way that a developer can accomplish this is by right-clicking the CSS file and choosing Diff File Against Org. Diff File Against Org is an option that allows the developer to compare a local file with its remote version in the org using Salesforce CLI commands. The developer can use Visual Studio Code to execute these commands by right-clicking on files or folders in the project and choosing from various diff options. Right-clicking the CSS file and choosing Diff File Against Org allows the developer to see the differences between the local CSS file and the remote CSS file in the org side by side in Visual Studio Code. This way, the developer can verify if their changes were deployed successfully or not. Clicking the Tools menu and selecting Diff Styles Against Org... is not a valid way to know if a CSS file was deployed to the org, as there is no such option in Visual Studio Code or Salesforce CLI. Right-clicking the folder for the component and choosing Diff Styles Against Org is not a valid way either, as there is no such option in Visual Studio Code or Salesforce CLI.

Right-clicking the folder for the component and choosing Diff Files Against Org is not a valid way either, as it will compare all the files in the folder, not just the CSS file, which may not be efficient or necessary.

Salesforce References: [Salesforce CLI Command Reference: force:source:diff], [Salesforce Developer Tools for Visual Studio Code]

NEW QUESTION: 34

Which format is the custom Salesforce field with the API name "My_Fiels_Name__c" transformed onto by default in Salesforce B2B Commerce?

- A. MyFieldName
- B. myFieldName
- C. myfieldname
- D. My_Field_Name__c

Answer: B (LEAVE A REPLY)

Explanation

The format that the custom Salesforce field with the API name My_Field_Name__c is transformed onto by default in Salesforce B2B Commerce is myFieldName. This is done to follow the JavaScript naming convention and to avoid conflicts with the standard Salesforce fields and relationships. The transformation rules are as follows:

- * Remove any namespace prefix from the field name, such as ccrz__ or cloudcraze__.
- * Remove any underscores from the field name and capitalize the first letter of each word after an underscore, such as MyFieldName.
- * Lowercase the first letter of the field name, such as myFieldName. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 35

What is the difference between Gross Layout Overrides and Subscriber Templates?

- A.** Gross Layout Overrides allow modification to CSS of a page, while Subscriber Templates allows for modification of the entire page including header and footer.
- B.** Subscriber Templates allows for modification of the header, the footer and the content in between them. Gross Layout Overrides only allow for modification of the header and footer.
- C.** Subscriber Templates allow for modification of the header and the footer, while Gross Layout Overrides allow for modification everything inside the header and footer.
- D.** Gross Layout Overrides allow for the modification of the footer, while Subscriber Templates allow for modification of everything inside the header and footer.

Answer: C (LEAVE A REPLY)

The difference between Gross Layout Overrides and Subscriber Templates is that Subscriber Templates allow for modification of the header and the footer, while Gross Layout Overrides allow for modification everything inside the header and footer. Subscriber Templates are custom Visualforce pages that can replace the standard header and footer components on a page. Gross Layout Overrides are custom Visualforce pages that can replace the default layout of a page, such as the number of columns, the position of widgets, or the content of sections. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Subscriber Templates, Gross Layout Overrides

NEW QUESTION: 36

Which three attributes are true regarding Subscriber pages? (3 answers)

- A.** Subscriber pages can include additional standard Salesforce B2B Commerce components such as featured products, category tree, and mini cart.
- B.** All the user interface components must be created manually.
- C.** Subscriber pages allow customers to quickly create new custom pages for their storefront.
- D.** Out of the Box, Subscriber Pages are completely blank HTML pages.
- E.** Standard Salesforce B2B Commerce components are automatically included on the page e.g. Header links, images, menu items, containers, etc.

Answer: A,C,D (LEAVE A REPLY)

Explanation

Three attributes that are true regarding Subscriber Pages are:

- * Subscriber Pages can include additional standard Salesforce B2B Commerce components such as featured products, category tree, and mini cart. These components can be added to the Subscriber Pages by using the CCRZ.pagevars.pageLabels and CCRZ.pagevars.pageSections objects, which contain the page labels and page sections that are defined in CCAdmin.
- * Subscriber Pages allow customers to quickly create new custom pages for their storefront. These pages can be used to display any content or functionality that is not available in the default pages, such as FAQs, testimonials, or promotions.
- * Out of the Box, Subscriber Pages are completely blank HTML pages. These pages do not have any predefined layout or content, so the user can customize them according to their requirements. The user can use Visualforce, HTML, CSS, JavaScript, or any other web technologies to create their own Subscriber Pages. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Subscriber Pages

NEW QUESTION: 37

Which practice is allowed when it comes to naming a Lightning web component's folder and associated files?

- A. Including whitespace
- B. Using a single underscore
- C. Using consecutive underscores
- D. Using a single hyphen (dash)

Answer: D (LEAVE A REPLY)

Explanation

To name a Lightning web component's folder and associated files, the developer should follow these naming rules:

- * Must begin with a lowercase letter
- * Must contain only alphanumeric or underscore characters
- * Must be unique in the namespace
- * Can't include whitespace
- * Can't end with an underscore
- * Can't contain two consecutive underscores
- * Can't contain a hyphen (dash)

The only exception to the last rule is when separating the namespace from the component name in markup. For example, if the component is in the example namespace and has the name myComponent, the folder name should be myComponent, but the markup tag should be <example-my-component>. The hyphen character (-) is required by the HTML standard for custom element names. The other options are either invalid or not recommended. For example, including whitespace or using consecutive underscores will cause errors, and using a single underscore will look odd and may confuse developers consuming the component. References:

NEW QUESTION: 38

What is essential for a Salesforce B2B Commerce theme to show up in the theme section in CC Admin?

- A. The theme needs to be set as a Custom Setting in Salesforce.
- B. The theme needs to be set in the Configuration Settings.
- C. The theme needs to have "theme" in the name of the Static Resource.
- D. The theme needs to be referred to in the head element on the page

Answer: C (LEAVE A REPLY)

An essential requirement for a Salesforce B2B Commerce theme to show up in the theme section in CC Admin is that the theme needs to have "theme" in the name of the Static Resource. For example, a theme named "MyTheme" will not appear in CC Admin, but a theme named "MyTheme_theme" will. This is how the framework identifies which static resources are themes and which are not. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Themes

NEW QUESTION: 39

What are two considerations to keep in mind when including additional JavaScript files in a Lightning web component?

- A. Each additional file needs a corresponding .js-meta.xml file.
- B. The files must be ES6 modules and must have names that are unique within the component's folder.
- C. A module can export named functions or variables
- D. Additional JavaScript files should be minified before deployment

Answer: (SHOW ANSWER)

When including additional JavaScript files in a Lightning web component, a developer should keep in mind two considerations: the files must be ES6 modules and must have names that are unique within the component's folder and a module can export named functions or variables. The files must be ES6 modules because LWC uses ES6 modules as the standard for modular JavaScript code. The files must have names that are unique within the component's folder because LWC uses the file name as the module identifier and does not allow duplicate identifiers. A module can export named functions or variables because LWC supports named exports, which allow a module to export multiple values with different names. Each additional file does not need a corresponding .js-meta.xml file, as this is only required for the main JavaScript file of the component. Additional JavaScript files should not be minified before deployment, as this is not necessary or recommended for LWC. Salesforce References: [Lightning Web Components Developer Guide: Include JavaScript Files], [Lightning Web Components Developer Guide: ES6 Modules]

NEW QUESTION: 40

Which event is invoked by any CCRZ Salesforce B2B CommerceView after the view is rendered?

- A. view:*.load
- B. view:*.refresh
- C. view:*.onload
- D. view:*.rendered

Answer: D (LEAVE A REPLY)

The event that is invoked by any CCRZ Salesforce B2B Commerce View after the view is rendered is view:*.rendered. This event is triggered by the render method of the CCRZ.View class, which is the base class for all views in the framework. The event can be used to perform any actions or logic that depend on the view being rendered, such as initializing widgets, binding events, or updating the user interface. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Events, View Class

NEW QUESTION: 41

A dev at Northern Trail Outfitters (NTO) exported Order Summary records via Data Loader, but noticed that some orders were missing. What is the most likely cause?

- A. The export job did not fully complete
- B. Order Life Cycle Type was Managed
- C. The user does not have rights to some of the records
- D. The Status was still set to Draft

Answer: (SHOW ANSWER)

Explanation

The most likely cause of why some orders were missing from the Data Loader export is that the Status was still set to Draft. The Status is a field on the Order Summary object that indicates the current state of the order.

The Status can have values such as Draft, Submitted, Confirmed, or Cancelled. A Draft order is an order that has not been submitted or confirmed by the customer or the seller. A Draft order is not considered a completed or valid order and is not included in reports or exports. When using Data Loader to export data from an org, Data Loader will only include orders that have a Status other than Draft, such as Submitted or Confirmed. If an order has a Status of Draft, Data Loader will not include it in the CSV file, which may result in missing orders.

The export job did not fully complete is not a likely cause of why some orders were missing from the Data Loader export, as it is not related to the order status or data filtering. Order Life Cycle Type was Managed is not a likely cause either, as it is not related to the order status or data filtering. The Order Life Cycle Type is a field on the Order Summary object that indicates whether the order is managed by Salesforce Order Management or by an external system. The user does not have rights to some of the records is not a likely cause either, as it contradicts the fact that some orders were exported successfully. If the user did not have rights to some of the records, Data Loader would not be able to access or export any orders at all. Salesforce References: B2B Commerce Developer Guide: Order Summary Object, B2B Commerce Developer Guide:

NEW QUESTION: 42

Which wire adapter should a developer use to retrieve metadata about a specific picklist?

- A. getPicklistMetadataValues
- B. getPicklistMetadata
- C. getPicklistValues
- D. getPicklist

Answer: C ([LEAVE A REPLY](#))

To retrieve metadata about a specific picklist, a developer should use the getPicklistValues wire adapter. The getPicklistValues wire adapter imports data from the @salesforce/ui-api module and returns an object that contains information such as the picklist's label, value, default value, validity, and controlling field values.

The getPicklistMetadataValues wire adapter does not exist. The getPicklistMetadata wire adapter does not exist either. The getPicklist wire adapter does not exist either. Salesforce References: [Lightning Web Components Developer Guide: getPicklistValues], [Lightning Web Components Developer Guide: Import User Interface API]

NEW QUESTION: 43

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

- 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: (SHOW ANSWER)

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]

NEW QUESTION: 44

Which three attributes are true regarding Subscriber pages? (3 answers)

- A.** Subscriber pages can include additional standard Salesforce B2B Commerce components such as featured products, category tree, and mini cart.
- B.** All the user interface components must be created manually.
- C.** Standard Salesforce B2B Commerce components are automatically included on the page e.g. Header links, images, menu items, containers, etc.
- D.** Out of the Box, Subscriber Pages are completely blank HTML pages.
- E.** Subscriber pages allow customers to quickly create new custom pages for their storefront.

Answer: ([SHOW ANSWER](#))

NEW QUESTION: 45

A developer needs to create a scheduled job in another system to move data into the B2B Commerce org. How can the developer do this without additional third party tools?

- A.** Install a minimal set of dev tools on a machine such as the Command Line Interface (CLI) and create appropriate scripts to import files containing the data
- B.** Set up an SFTP server as a waystation, drop the files there using the off-platform job and schedule a job in-platform to process the file
- C.** Set up WebDAV with SFTP as a waystation, drop the files there using the off-platform job and schedule a job in-platform to process the file
- D.** Create a job in the org (on-platform) to drop a file of existing data, Use the off-platform machine to generate a file and identify the details between the two, Push the changes to the org's "Import" directory

Answer: **B** ([LEAVE A REPLY](#))

Explanation

Option B is the correct answer because it describes a way to create a scheduled job in another system to move data into the B2B Commerce org without additional third party tools. The developer can set up an SFTP server as a waystation, drop the files there using the off-platform job and schedule a job in-platform to process the file. This way, the developer can use the built-in

integration capabilities of the B2B Commerce platform, such as the Windows Integration Service (WIS) or the Integration Job Definitions, to import the data from the SFTP server into the org. The other options are incorrect because they either require additional third party tools, such as WebDAV or CLI, or they do not create a scheduled job in another system, but rather in the same org. References: Creating and editing integration jobs, Create a B2B Commerce Org and Checkout Flow, B2B Commerce on Lightning Experience Developer Guide, B2B Commerce and D2C Commerce Developer Guide

NEW QUESTION: 46

A Developer created a custom field that a project wants to expose on a given page.

How does the Developer ensure that the field is available to display on a given page?

- A.** Override the Service Class that the page uses and update the ServiceManagement in CCAAdmin for the given storefront to use this new Service Class.
- B.** Override the Logic Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class
- C.** Create a new Service Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class
- D.** Create a new Logic Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class

Answer: (SHOW ANSWER)

To ensure that a custom field is available to display on a given page, the Developer needs to override the Service Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class. The Service Class is responsible for retrieving and setting data for a given page. The Logic Class is responsible for implementing business logic and validation rules for a given page. Creating a new Service Class or Logic Class is not necessary, as overriding an existing one can achieve the same result. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Service Classes, Logic Classes

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 47

A Developer created a custom field that a project wants to expose on a given page.

How does the Developer ensure that the field is available to display on a given page?

- A.** Create a new Service Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class
- B.** Override the Logic Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class
- C.** Override the Service Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class.
- D.** Create a new Logic Class that the page uses and update the Service Management in CCAAdmin for the given storefront to use this new Service Class

Answer: C ([LEAVE A REPLY](#))

NEW QUESTION: 48

Universal Containers (UC) needs to display data from standard objects (entities) in a different format than what comes with B2B Commerce out of the box. In doing this, what is one advantage of using the Lightning Data Service vs using a custom Controller class?

- A.** Lightning Data Service translates the developer's component implementation to a VisualForce page for backward compatibility.
- B.** JavaScript proxies for transport objects are created in the developer's IDE automatically.
- C.** The developer can read, create, or modify single records or metadata without writing Apex code.
- D.** There is a Visual Studio add-in that accelerates the layout process

Answer: C ([LEAVE A REPLY](#))

Lightning Data Service (LDS) is a Salesforce UI API layer that empowers developers to perform CRUD operations and manage Salesforce data without the need for custom Apex controllers. LDS is designed to improve performance, user interface consistency, and developer productivity by handling data operations and sharing rules, thus reducing the need for server-side code. For more details, refer to the Salesforce documentation on Lightning Data Service:Salesforce Lightning Data Service Documentation.

NEW QUESTION: 49

A developer is setting up a storefront from scratch. They need to create a storefront, push store sources, create buyer users, import products, and create and search index.

Which command allows the developer to accomplish this task?

- A.** `sfdx commerce:store:quickstart:create -templatename 'b2c-lite-storefront'`
- B.** `sfdx commerce:store:quickstart:setup -definitionfile store-scratch-def.json`
- C.** `sfdx commerce:store:create -store-name test-store`
- D.** `sfdx commerce:store:open -store-name test-store`

Answer: ([SHOW ANSWER](#))

Explanation

The `sfdx commerce:store:quickstart:setup` command allows the developer to set up a store from scratch using a definition file that specifies the store configuration. This command performs the following tasks:

- * Pushes the store sources to the scratch org
- * Creates a buyer user
- * Imports products
- * Creates a search index
- * Opens the store in a browser

The other options are not correct because:

- * A. `sfdx commerce:store:quickstart:create` only creates a community for the store, but does not perform the other tasks required to set up a store from scratch.
- * C. `sfdx commerce:store:create` creates and sets up a store, but does not use a definition file to specify the store configuration. Instead, it uses command-line flags to provide the store parameters.
- * D. `sfdx commerce:store:open` only opens an existing store in a browser, but does not create or set up a store from scratch.

References:

- * store Commands
- * Build a B2B and B2B2C Commerce Store

NEW QUESTION: 50

Which option is the correct syntax to render a property in a Lightning web component template?

- A. Surround the property with curly braces: `{property}`
- B. Surround the property with brackets: `[property]`
- C. Surround the property with an exclamation point and curly braces: `!{property}`
- D. Surround the property with curly braces and exclamation point: `!{!property}`

Answer: A (LEAVE A REPLY)

Explanation

The correct syntax to render a property in a Lightning web component template is to surround the property with curly braces: `{property}`. This syntax allows you to access the property value from the JavaScript class of the component and display it in the HTML template. The other options are incorrect because they either use invalid characters or do not follow the standard syntax for rendering a property in a Lightning web component template. References: Use JavaScript Properties in Templates | Lightning Web Components Developer Guide

NEW QUESTION: 51

A developer needs to make a call to a long running web service which is critical to finalizing their checkout process. Which three items should the developer consider in their implementation?

- A. A new CORS entry may need to be created in Setup
- A new Named Credential may need to be created in Setup
- B. An Apex method returning a Continuation will need to be created

C. Requests to the service should be brokered to prevent limit exceptions

D. A new Remote Site may need to be created in Setup

Answer: A,B,C (LEAVE A REPLY)

A developer needs to make a call to a long running web service which is critical to finalizing their checkout process. The developer should consider the following items in their implementation:

* A new CORS entry may need to be created in Setup. CORS stands for Cross-Origin Resource Sharing, which is a mechanism that allows web browsers to make requests to servers on different domains. If the web service is hosted on a different domain than the B2B Commerce site, the developer may need to add

* a CORS entry in Setup to allow the browser to access the web service. This entry specifies the origin, path, and method of the request, as well as any headers or cookies that are allowed¹.

* A new Named Credential may need to be created in Setup. A Named Credential is a type of metadata that stores the URL and authentication settings of a web service. By using a Named Credential, the developer can avoid hardcoding the web service URL and credentials in their code, and instead reference the Named Credential by name. This makes the code more secure and easier to maintain².

* An Apex method returning a Continuation will need to be created. A Continuation is a special type of Apex object that allows the developer to make asynchronous calls to long running web services. By using a Continuation, the developer can avoid blocking the main thread and improve the user experience. A Continuation method must return a Continuation object, specify the web service URL and parameters, and register a callback method that handles the response³.

Option D and Option E are incorrect because they are not relevant to the scenario. Option D suggests that the developer should broker the requests to the web service, which means to use an intermediary service that manages the requests and responses. This may be useful for some scenarios, but it is not required for making a call to a long running web service. Option E suggests that the developer should create a new Remote Site in Setup, which is a way to whitelist the domains that can be accessed from Apex code. However, this is not necessary if the developer uses a Named Credential, which automatically handles the Remote Site setting².

References:

* Set Up CORS

* Named Credentials

* Continuation Class

NEW QUESTION: 52

Which method is used to override when extending the Salesforce B2B Commerce logic providers?

A. doAction

B. fetch

C. doLogic

D. process

Answer: (SHOW ANSWER)

NEW QUESTION: 53

What is a valid way of referencing the global `cc_api_CartExtension` apex class via subscriber code?

- A. `cloudcraze.cc_api_CartExtension`
- B. `ccrz__cc_api_CartExtension`
- C. `ccrz.cc_api_CartExtension`
- D. `c__cc_api_CartExtension`

Answer: C ([LEAVE A REPLY](#))

NEW QUESTION: 54

Which two statements are accurate about the Cart Item with a Type of Charge?

- A. It is created with the Cart Delivery Group Method after the shipping integration
- B. It is created with the Cart Delivery Group Method after the freight integration
- C. It is linked directly to a Cart Id
- D. It is linked directly to a Catalog Id

Answer: C,D ([LEAVE A REPLY](#))

Explanation

Two statements that are accurate about the Cart Item with a Type of Charge are that it is linked directly to a Cart Id and that it is linked directly to a Catalog Id. A Cart Item with a Type of Charge is a special type of Cart Item that represents an additional charge or fee that is applied to a Cart, such as shipping, handling, or tax. A Cart Item with a Type of Charge is linked directly to a Cart Id, which means that it belongs to a specific Cart and can be retrieved or updated along with other Cart Items. A Cart Item with a Type of Charge is also linked directly to a Catalog Id, which means that it references a specific Catalog that contains the products and prices for the store. A Cart Item with a Type of Charge is not created with the Cart Delivery Group Method after the shipping integration or after the freight integration, as these are not related to the creation of Cart Items. The Cart Delivery Group Method is a method that determines how products are grouped into delivery groups based on their shipping methods and addresses. The shipping integration and the freight integration are integrations that calculate and apply shipping costs and freight charges to a Cart or an Order. Salesforce References: B2B Commerce Developer Guide: Cart Item Object, B2B Commerce Developer Guide: Shipping Integration, B2B Commerce Developer Guide: Freight Integration

NEW QUESTION: 55

What is default behavior for how the Salesforce B2B Commerce Global APIs transform Salesforce data?

- A. Fields names are returned using the Salesforce naming convention.
- B. Fields names are returned with `;` prepended in their name.
- C. Fields names are returned with a lowercase first letter, camelcase convention
- D. Fields names can be mapped to any naming convention desired

Answer: C ([LEAVE A REPLY](#))

The default behavior for how the Salesforce B2B Commerce Global APIs transform Salesforce data is to return field names with a lowercase first letter, camelcase convention. For example, the field name `ccrz__E_Product__c` in Salesforce will be transformed to `eProduct` in the API. This is done to follow the JavaScript naming convention and to avoid conflicts with the standard Salesforce fields and relationships. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 56

What two kinds of queries do the methods in Salesforce B2B Commerce services perform by default? (2 answers)

- A. SOSL
- B. SQL
- C. SOQL
- D. Schema-less queries

Answer: C,D ([LEAVE A REPLY](#))

Explanation

Two kinds of queries that the methods in Salesforce B2B Commerce services perform by default are SOQL and schema-less queries. SOQL is the query language that is used to retrieve data from Salesforce objects and fields. Schema-less queries are queries that do not specify the object or field names explicitly, but use placeholders instead. For example, `ccrz.ccServiceDao.getQuery('SELECT Id FROM Account WHERE Name = :name')` is a schema-less query that uses `:name` as a placeholder for the field name. The framework will transform this query to use the actual field name based on the query transformation rules. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 57

While in the process of gathering requirements from a customer about how they would like to set up their net new storefront checkout experience, a consultant learns that the customer needs the ability to add new shipping and billing addresses during checkout.

Which approach should a developer take to meet this requirement?

- A. Create a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component.
- B. Enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow.
- C. Enable Buyer Managed Contact Point Addresses within Commerce Administration.
- D. Create a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow.

Answer: B ([LEAVE A REPLY](#))

To enable the ability to add new shipping and billing addresses during checkout, a developer should enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow. The Buyer Managed Contact Point Addresses is a feature that allows customers to add, edit, or delete their shipping and billing addresses during checkout. The developer can enable this feature by setting the `buyerManagedContactPointAddressesEnabled` attribute to true in the Shipping Address standard component in the Checkout subflow. The Shipping Address standard component is a component that displays and collects the shipping address information for the cart or order. The Checkout subflow is a subflow that defines the steps and components for the checkout process in the storefront. Creating a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component is not a valid way to enable this feature, as there is no such component as Buyer Managed Contact Point Addresses. Enabling Buyer Managed Contact Point Addresses within Commerce Administration is not a valid way either, as this feature is not configurable in Commerce Administration. Creating a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow is not a valid way either, as this is unnecessary and complicated when there is already a standard component that supports this feature. Salesforce References: [B2B Commerce Developer Guide: Buyer Managed Contact Point Addresses], [B2B Commerce Developer Guide: Shipping Address Component], [B2B Commerce Developer Guide: Checkout Subflow]

NEW QUESTION: 58

A developer is on a tight timeline and needs to implement a Lightning web component which can read, create and modify single records. What is the recommended path forward?

- A. Use base components
- B. Write custom functions against a wire adapter
- C. Create an Apex Controller
- D. Use Lightning Data Service

Answer: (SHOW ANSWER)

To implement a Lightning web component which can read, create and modify single records, the recommended path forward is to use Lightning Data Service. Lightning Data Service is a service that provides access to Salesforce data and metadata, cache management, and data synchronization across components.

Lightning Data Service allows developers to use standard components or base components to perform CRUD (create, read, update, delete) operations on single records without writing Apex code or SOQL queries.

Lightning Data Service also handles data caching, performance optimization, and conflict resolution automatically. Using base components is not a sufficient way to implement a Lightning web component which can read, create and modify single records, as base components are only user interface elements that do not provide data access or manipulation functionality by

themselves. Base components need to be used with Lightning Data Service or other services to perform CRUD operations on single records. Writing custom functions against a wire adapter is not a recommended way to implement a Lightning web component which can read, create and modify single records, as it involves writing complex and error-prone JavaScript code that may not be efficient or scalable. Writing custom functions against a wire adapter also requires handling data caching, performance optimization, and conflict resolution manually. Creating an Apex controller is not a recommended way either, as it involves writing Apex code that may not be necessary or optimal for performing CRUD operations on single records. Creating an Apex controller also requires exposing Apex methods using `@AuraEnabled` or `@RemoteAction` annotations and invoking them from JavaScript code using imperative calls or promises. Salesforce References: Lightning Web Components Developer Guide: Access Salesforce Data, Lightning Web Components Developer Guide: Base Components, [Lightning Web Components Developer Guide: Call Apex Methods]

NEW QUESTION: 59

Which event should be triggered when user facing info, warning or error messages need to be displayed on a Visualforce page?

- A. showMessage
- B. displayPageMessage
- C. displayMessage
- D. pageMessage

Answer: D (LEAVE A REPLY)

The event that should be triggered when user facing info, warning or error messages need to be displayed on a Visualforce page is pageMessage. This event is triggered by the displayPageMessage method of the CCRZ.util class, which is a utility class that provides various helper methods for common tasks. The event can be used to display a message on the page with a specified type, title, and text. For example, `CCRZ.util.displayPageMessage('error', 'Oops!', 'Something went wrong.')` will trigger the pageMessage event and display an error message on the page. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Events, Util Class

NEW QUESTION: 60

In which three ways should useful debugging information in Salesforce B2B Commerce implementation be garnered? (3 answers) A) Enabling the logging token via

- A. Admin and subsequently inspecting the logs via the browser console.
- B. Logging a case with Salesforce support to enable advanced debugging options.
- C. Enabling debugging options for the current user and visually inspecting the Salesforce debug logs.
- D. Placing a `System.debug()` statement anywhere in the class being debugged.
- E. Logging into the community as a system administrator to identify any potential permissions or Visualforce exceptions.

Answer: (SHOW ANSWER)

Useful debugging information in Salesforce B2B Commerce implementation can be garnered in three ways:

Enabling the logging token via Admin and subsequently inspecting the logs via the browser console. This will enable logging messages and errors to the browser console, which can be viewed by opening the Developer Tools in the browser. The logging token can be enabled by setting the value of CO.logToken to true in CCAAdmin.

Enabling debugging options for the current user and visually inspecting the Salesforce debug logs. This will enable logging messages and errors to the Salesforce debug logs, which can be viewed by opening the Debug Logs page in Salesforce Setup. The debugging options can be enabled by creating a Debug Level and a Trace Flag for the current user in Salesforce Setup. Logging into the community as a system administrator to identify any potential permissions or Visualforce exceptions. This will allow viewing any errors or warnings that may occur on the community pages due to insufficient permissions or Visualforce issues. The system administrator can also access CCAAdmin and other tools from within the community. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Logging, Debug Your Code

NEW QUESTION: 61

A new payment type for the Checkout flow has been implemented. Which three descriptors follow best practice for possible configuration metadata are needed to enable a flow? (3 answers)

- A. *.pay
- B. Cart
- C. Checkout
- D. *.Edit
- E. *.New

Answer: A,D,E (LEAVE A REPLY)

Explanation

To enable a new payment type for the Checkout flow, three possible configuration metadata descriptors are needed:

* *.pay: This descriptor defines the payment type name, label, description, icon, and handler class. For example, CO.pmts.CreditCard.pay defines the payment type for credit card payments.

* *.Edit: This descriptor defines the Visualforce page that is used to edit or update an existing payment of this type. For example, CO.pmts.CreditCard.Edit defines the page that allows editing a credit card payment.

* *.New: This descriptor defines the Visualforce page that is used to create a new payment of this type.

For example, CO.pmts.CreditCard.New defines the page that allows creating a new credit card payment.

Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Payment Configuration Settings

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 62

A user wants to have a Contact Us page in the storefront. This page will be a web-to-lead form and it should have the header and footer, essentially the same look and feel as all the pages in the application. How can this requirement be fulfilled?

- A. Page Include
- B. Subscriber Page (CC Page)
- C. Subscriber Template
- D. Body Include Begin

Answer: B (LEAVE A REPLY)

Explanation

To have a Contact Us page in the storefront that is a web-to-lead form and has the same look and feel as all the pages in the application, the requirement can be fulfilled by creating a Subscriber Page (CC Page). This is a custom Visualforce page that can be added to the storefront and use the standard header and footer components. The page can also include a web-to-lead form that submits data to Salesforce as leads. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Subscriber Pages

NEW QUESTION: 63

Which three actions must a developer take, in a B2B Commerce store, to accept credit card payments using a client's chosen payment provider?

- A. Create a named credential for authentication with the payment provider.
- B. Create a RegisteredExternalService record for the custom payment provider class.
- C. Create an Apex class that implements the `sfdc_checkout.PaymentGatewayAdapter`
- D. Create a PaymentProviderGateway record for the custom payment provider class.
- E. Create an Apex class that implements the `commercepayments.PaymentGatewayAdapter`.

Answer: (SHOW ANSWER)

Explanation

Three actions that a developer must take, in a B2B Commerce store, to accept credit card payments using a client's chosen payment provider are: create a named credential for authentication with the payment provider, create an Apex class that implements the `sfdc_checkout.PaymentGatewayAdapter` interface, and create a PaymentProviderGateway

record for the custom payment provider class. Creating a named credential for authentication with the payment provider allows the developer to securely store and manage authentication information, such as username, password, token, or certificate, for connecting to the payment provider's API or service. Creating an Apex class that implements the `sfmc_checkout.PaymentGatewayAdapter` interface allows the developer to define custom logic for processing credit card payments using the payment provider's API or service. The interface provides methods for validating credit card information, authorizing payments, capturing payments, voiding payments, and refunding payments. Creating a `PaymentProviderGateway` record for the custom payment provider class allows the developer to register the custom payment provider class as a payment gateway in the store and associate it with a payment method. Creating a `RegisteredExternalService` record for the custom payment provider class is not a required action, as this is only used for registering external services that are not related to payment processing, such as tax or shipping services. Creating an Apex class that implements the `commercepayments.PaymentGatewayAdapter` interface is not a required action either, as this is only used for D2C Commerce stores, not B2B Commerce stores. Salesforce References: B2B Commerce Developer Guide: Payment Integration, B2B Commerce Developer Guide: Payment Gateway Adapter Interface, B2B Commerce Developer Guide: Payment Provider Gateway Object

NEW QUESTION: 64

Which category can receive signaling from a Lightning Message Channel?

- A. only descendents (i.e. children)
- B. only direct siblings
- C. page (anywhere on it)
- D. only ancestors (i.e. parents)

Answer: C (LEAVE A REPLY)

A Lightning Message Channel can receive signaling from any component on a page (anywhere on it). A Lightning Message Channel is a service that allows components to communicate with each other across different Salesforce UI technologies, such as Aura, LWC, Visualforce, and Lightning web apps. A component can subscribe to a message channel and receive events that are published by another component on the same or different page. The message channel acts as a mediator that delivers the events to the subscribers. A Lightning Message Channel does not receive signaling only from descendants (i.e., children), only from direct siblings, or only from ancestors (i.e., parents), as these are limitations of event-based communication. Salesforce Reference: [Lightning Web Components Developer Guide: Communicate Across Salesforce UI Technologies], [Lightning Web Components Developer Guide: Communicate with Events]

NEW QUESTION: 65

An administrator has just provided a developer with a completely new org and a username. Assuming the username is `me@my-new-org.org`, what is the correct set of steps to authorize the org for Command Line Interface (CLI) access so the developer can deploy Lightning web components?

- A. Run the command: 'sfdx force:login -r "https://login.salesforce.com" and supply the credentials in the browser when it opens.
- B. Run the command 'sfdx force:auth:web:login -a "https://login.salesforce.com"' and then supply the credentials in the browser when it opens.
- C. Run the command: 'sfdx force:auth:web:login -r "https://login.salesforce.com" and then supply the credentials in the ^ browser when it opens ^
- D. Run the command 'sfdx force:auth:web:login -r "https://login.salesforce.com" -username^'mefaJmy-new-org.org''

Answer: C (LEAVE A REPLY)

To authorize the org for Command Line Interface (CLI) access so the developer can deploy Lightning web components, the developer should run the command: 'sfdx force:auth:web:login -r "https://login.salesforce.com" and then supply the credentials in the browser when it opens. The sfdx force:auth:web:login command is a Salesforce CLI command that authorizes an org using the web server flow.

The web server flow is an OAuth 2.0 authentication flow that opens a browser window and prompts the user to log in to Salesforce and allow access to the CLI. The -r flag specifies the login URL of the org, which is

https://login.salesforce.com for production or developer orgs. Running this command will open a browser window and ask the developer to enter their username and password for the org. After successfully logging in, the developer will be able to use the CLI to perform various tasks with the org, such as deploying or retrieving metadata, running tests, or executing commands. Running the command: 'sfdx force:login -r

"https://login.salesforce.com" is not a valid way to authorize the org for CLI access, as there is no such command as sfdx force:login. Running the command 'sfdx force:auth:web:login -a "https://login.salesforce.com"' is not a valid way either, as the -a flag specifies an alias for the org, not a login URL. Running the command 'sfdx force:auth:web:login -r "https://login.salesforce.com" -username'me@my-new-org.org'' is not a valid way either, as there is no such flag as -username. Salesforce References: Salesforce CLI Command Reference: force:auth:web:login, Salesforce Developer Tools for Visual Studio Code

NEW QUESTION: 66

How do the REST APIs in Salesforce B2B Commerce support pass-through parameter handling

- A. An exception is generated for unknown API keys
- B. Parameters are filtered out before the request is processed
- C. Parameters are passed through the service handlers
- D. Parameters are separated, but unused

Answer: C (LEAVE A REPLY)

NEW QUESTION: 67

When a user buys 10 units of product B, the user wants 1 unit of Product A to be automatically added to the cart. How can this requirement be fulfilled?

- A. Override the prepareToAdd method in ccrz.cc_api_CartExtension
- B. Override the preprocess method in ccrz.cc_api_CartExtension
- C. Override the prepareForSave method in ccrz.cc_api_CartExtension
- D. Override the AllowCheckout method in ccrz.cc_api_CartExtension

Answer: B (LEAVE A REPLY)

NEW QUESTION: 68

A dev at Northern Trail Outfitters (NTO) exported Order Summary records via Data Loader, but noticed that some orders were missing. What is the most likely cause?

- A. Status was not in Created
- B. Billing Contact was missing
- C. Original Order information was missing
- D. Order Life Cycle Type was Managed
- E. Currency mapping was not done correctly

Answer: A (LEAVE A REPLY)

When exporting Order Summary records via Data Loader, if some orders are missing, a common cause could be that those orders did not have a status that was included in the export criteria, such as 'Created'. Salesforce data exports can be filtered based on specific criteria, and if the status of some orders doesn't match the criteria, those orders will not be included in the export. The other options listed are less likely to be direct causes for missing orders in an export operation. For more information on using Data Loader and understanding export criteria, refer to the Salesforce Data Loader Guide:Salesforce Data Loader Guide.

NEW QUESTION: 69

A user wants to have a Contact Us page in the storefront. This page will be a web-to-lead form and it should have the header and footer, essentially the same look and feel as all the pages in the application. How can this requirement be fulfilled?

- A. Page Include
- B. Subscriber Page (CC Page)
- C. Subscriber Template
- D. Body Include Begin

Answer: B (LEAVE A REPLY)

To have a Contact Us page in the storefront that is a web-to-lead form and has the same look and feel as all the pages in the application, the requirement can be fulfilled by creating a Subscriber Page (CC Page). This is a custom Visualforce page that can be added to the storefront and use the standard header and footer components. The page can also include a web-to-lead form that submits data to Salesforce as leads. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Subscriber Pages

NEW QUESTION: 70

A developer needs to implement a custom Lightning web component (LWC) for the storefront.

The LWC contains language-specific text values.

How should the developer translate the text values?

- A. Import static resources for the text values and add them into the LWC.
- B. Use a CustomLabel xml file in the LWC to add the text values there.
- C. Create custom labels for the text values and import them in the LWC.
- D. Create a custom Metadata object for the text values and query it in the LWC.

Answer: C (LEAVE A REPLY)

Explanation

Custom labels are text values that can be translated into any language that Salesforce supports. They are useful for displaying language-specific text in Lightning web components. To use custom labels in a LWC, the developer needs to create them in the Setup menu and assign them to a language and a value. Then, the developer can import them in the LWC using the `@salesforce/label` scoped module. For example, if the developer has a custom label named `welcomeHeader`, they can import it as follows:

```
import welcomeHeader from '@salesforce/label/c.welcomeHeader';
```

Then, they can use it in the HTML template or the JavaScript file of the LWC. For example, in the HTML template, they can use it as follows:

HTMLAI-generated code. Review and use carefully. More info on FAQ.

```
<template>
```

```
<h1>{welcomeHeader}</h1>
```

```
</template>
```

The custom label will automatically display the translated value based on the user's language preference. The developer can also use the `lightning-formatted-text` component to format the custom label value with HTML tags.

The other options are not correct because:

- * A. Importing static resources for the text values is not a recommended way to translate text values in a LWC. Static resources are files that are stored in Salesforce and can be referenced by applications. They are not designed for storing language-specific text values and they do not support automatic translation based on the user's language preference.
- * B. Using a CustomLabel xml file in the LWC to add the text values there is not a valid option. Custom labels are not stored in xml files, but in the Setup menu. They cannot be added directly to the LWC, but they need to be imported using the `@salesforce/label` scoped module.
- * D. Creating a custom Metadata object for the text values and querying it in the LWC is not a feasible option. Custom Metadata objects are records that store configuration data that can be deployed and packaged. They are not intended for storing language-specific text values and they do not support automatic translation based on the user's language preference. Querying them in the LWC would also require an Apex class and a wire service, which would add unnecessary complexity to the solution.

References:

- * Use Custom Labels in Lightning Web Components
- * Custom Labels

* Internationalizing Your Lightning Web Component (LWC)

NEW QUESTION: 71

Which three actions must a developer take, in a B2B Commerce store, to accept credit card payments using a client's chosen payment provider?

- A. Create a named credential for authentication with the payment provider.
- B. Create a RegisteredExternalService record for the custom payment provider class.
- C. Create an Apex class that implements the `sfdc_checkout.PaymentGatewayAdapter`
- D. Create a PaymentProviderGateway record for the custom payment provider class.
- E. Create an Apex class that implements the `commercepayments.PaymentGatewayAdapter`.

Answer: A,C,D (LEAVE A REPLY)

Three actions that a developer must take, in a B2B Commerce store, to accept credit card payments using a client's chosen payment provider are: create a named credential for authentication with the payment provider, create an Apex class that implements the `sfdc_checkout.PaymentGatewayAdapter` interface, and create a PaymentProviderGateway record for the custom payment provider class. Creating a named credential for authentication with the payment provider allows the developer to securely store and manage authentication information, such as username, password, token, or certificate, for connecting to the payment provider's API or service. Creating an Apex class that implements the `sfdc_checkout.PaymentGatewayAdapter` interface allows the developer to define custom logic for processing credit card payments using the payment provider's API or service. The interface provides methods for validating credit card information, authorizing payments, capturing payments, voiding payments, and refunding payments. Creating a PaymentProviderGateway record for the custom payment provider class allows the developer to register the custom payment provider class as a payment gateway in the store and associate it with a payment method. Creating a RegisteredExternalService record for the custom payment provider class is not a required action, as this is only used for registering external services that are not related to payment processing, such as tax or shipping services. Creating an Apex class that implements the `commercepayments.PaymentGatewayAdapter` interface is not a required action either, as this is only used for D2C Commerce stores, not B2B Commerce stores. Salesforce References: B2B Commerce Developer Guide: Payment Integration, B2B Commerce Developer Guide: Payment Gateway Adapter Interface, B2B Commerce Developer Guide: Payment Provider Gateway Object

NEW QUESTION: 72

Which three actions are applicable when modifying the number of steps required in the Salesforce Commerce Checkout flow? (3 answers)

- A. Perform a template override on the Checkout page.
- B. Add a page include to the checkout page.
- C. Build and activate a new configuration cache setting via CC admin.
- D. Set the value of the configuration setting defined as `CO.useDef` to TRUE
- E. Set the value of the configuration setting defined as `CO.overrideFlow` to TRUE.

Answer: A,C,E (LEAVE A REPLY)

Three actions that are applicable when modifying the number of steps required in the Salesforce Commerce Checkout flow are:

Perform a template override on the Checkout page. This action will allow you to change the structure and content of the Checkout page, such as adding or removing sections, widgets, or fields. For example, you can override the checkout.handlebars template and modify it according to your requirements.

Set the value of the configuration setting defined as CO.overrideFlow to TRUE. This setting will enable you to use your own custom checkout flow instead of the default one. You need to set this value to true before you can modify the checkout flow.

Set the value of the configuration setting defined as CO.useDef to TRUE. This setting will enable you to use a single-page checkout flow instead of a multi-step checkout flow. You need to set this value to true if you want to reduce the number of steps in the checkout flow to one. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Checkout Flow

NEW QUESTION: 73

What is a valid way of referencing the global cc_api_CartExtension apex class via subscriber code?

- A. ccrz__cc_api_CartExtension
- B. c__cc_api_CartExtension
- C. cloudcraze.cc_api_CartExtension
- D. ccrz.cc_api_CartExtension

Answer: (SHOW ANSWER)

Explanation

A valid way of referencing the global cc_api_CartExtension apex class via subscriber code is to use ccrz.cc_api_CartExtension. This is the name of the class that is defined in the cloudcraze managed package.

The class is global, so it can be accessed by subscriber code. The other options are either invalid or incorrect.

Salesforce References: B2B Commerce and D2C Commerce Developer Guide, API Classes

NEW QUESTION: 74

Which event is triggered within Salesforce B2B Commerce whenever a cart's state changes?

- A. cartChange
- B. cart
- C. pageMessage
- D. cartState

Answer: (SHOW ANSWER)

Explanation

The event that is triggered within Salesforce B2B Commerce whenever a cart's state changes is cartChange.

This event is triggered by the `updateCart` method of the `CCRZ.Cart` class, which is responsible for updating the cart data and performing any actions that depend on the cart state, such as applying discounts, taxes, or shipping charges. The event can be used to notify all the components that are listening to it that the cart has changed and they should refresh their data accordingly.

Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Events, Cart Class

NEW QUESTION: 75

A developer has been working on the flow of an Inventory Class for checkout. The developer has handled all the error states and now needs to indicate that inventory is available for all items and amounts in the cart.

Which step should happen next?

- A. Return `TRUE`
- B. Return `sfdc_checkout.IntegrationStatus.Status.SUCCESS`
- C. Return `sfdc.checkout.InventoryStatus.Status.SUCCESS`
- D. Return `sfdc.checkout.InventoryStatus.SUCCESS`

Answer: C (LEAVE A REPLY)

To indicate that inventory is available for all items and amounts in the cart, the developer should return `sfdc.checkout.InventoryStatus.Status.SUCCESS`. The `sfdc.checkout.InventoryStatus.Status.SUCCESS` is a constant value that represents a successful inventory status. It means that the inventory class has checked the availability of all the products in the cart and confirmed that they are in stock and can be fulfilled. The developer should use this value as the return value of the `checkInventory` method, which is a method that implements the `sfdc.checkout.InventoryCheck` interface. The `checkInventory` method is invoked during the checkout flow to validate the inventory availability of the products in the cart. Returning `TRUE` is not a valid way to indicate inventory availability, as it is not a valid inventory status value. Returning `sfdc_checkout.IntegrationStatus.Status.SUCCESS` is not a valid way either, as it is not an inventory status value, but rather an integration status value. Returning `sfdc.checkout.InventoryStatus.SUCCESS` is not a valid way either, as it is not a valid syntax for accessing the inventory status value. Salesforce Reference: B2B Commerce Developer Guide: Inventory Integration, B2B Commerce Developer Guide: Inventory Status Enum, B2B Commerce Developer Guide: Inventory Check Interface

NEW QUESTION: 76

Which method signature is used in the Global API's?

- A. `Map<String, Object>`
- B. `ccrz.cc_Output (ccrz.cc_Input input)`
- C. `List<List<Object>>`
- D. Changes based on API and Method name

Answer: A (LEAVE A REPLY)

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 77

Which cart item type is created with the Cart Delivery Group Method after the shipping integration?

- A. Surcharge
- B. Charge
- C. Shipping
- D. Delivery

Answer: B (LEAVE A REPLY)

A cart item with a type of Charge is created with the Cart Delivery Group Method after the shipping integration. A shipping integration is an integration that calculates and applies shipping costs and options to a cart or an order based on various factors, such as product weight, dimensions, quantity, location, and carrier service level. A shipping integration can use either an external shipping service provider or custom Apex code to perform the shipping calculation. The Cart Delivery Group Method is a method that determines how products are grouped into delivery groups based on their shipping methods and addresses. A delivery group is a logical grouping of products that are shipped together using the same shipping method and address. After performing the shipping calculation for each delivery group, the shipping integration creates a cart item with a type of Charge and a charge type of Shipping for each delivery group. These cart items represent the shipping costs and options that are applied to each delivery group. A cart item with a type of Surcharge, Shipping, or Delivery does not exist or is not related to the shipping integration. Salesforce References: [B2B Commerce Developer Guide: Shipping Integration], [B2B Commerce Developer Guide: Cart Delivery Group Method]

NEW QUESTION: 78

What is a best practice when passing query parameters from user interface to an apex controller?

- A. Query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component.
- B. String parameters should be trimmed using String.trim().
- C. Query parameters should be passed only to Salesforce B2B Commerce classes that you are extending.
- D. Query parameters should be stored on a backbone model prior to passing them to the server

Answer: A ([LEAVE A REPLY](#))

Explanation

A best practice when passing query parameters from user interface to an apex controller is to query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component. This function will encode any special characters in the query parameters to prevent cross-site scripting (XSS) attacks or SOQL injection attacks. For example, `ccrz.ccRemoteActions.getProducts('{!JSINHTMLENCODE(searchTerm)}')` will encode the searchTerm parameter before passing it to the apex controller. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Security

NEW QUESTION: 79

Which three steps are necessary to have subscriberpage added to Salesforce B2B Commerce after creating a custom Visualforce page? (3 answers)

- A. Create a new CC Subscriber Page record that points to your custom Visualforce page.
- B. Create a new Visualforce page, and manually import the Salesforce B2BCommerce JavaScript libraries.

Run in Anonymous Apex `ccrz.cc_util_Reflection.upsertPageUIKey([arg1],[arg2],[arg3]);`

- C. Refresh the Page Keys Index in CC Admin.
- D. Enable the Subscriber Page in CC Admin.

Answer: ([SHOW ANSWER](#))

Explanation

Three steps that are necessary to have a subscriber page added to Salesforce B2B Commerce after creating a custom Visualforce page are:

* Create a new CC Subscriber Page record that points to your custom Visualforce page. This record will store information about the subscriber page, such as the name, description, URL, and Visualforce page.

For example, you can create a new record named MySubscriberPage that points to your custom Visualforce page named MyPage.

* Refresh the Page Keys Index in CC Admin. This action will update the page keys index, which is a cache that stores the mapping between the page keys and the subscriber pages. You need to do this whenever you create or modify a subscriber page record.

* Enable the Subscriber Page in CC Admin. This action will allow you to select the subscriber page from the CC Page Settings configuration and assign it to a CC Page. For example, you can enableMySubscriberPage and assign it to the Home page. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Subscriber Pages

NEW QUESTION: 80

What is a method to resolve if the current storefront customer is a Salesforce B2B Commerce guest user in an apex class?

- A. `ccrz.cc_CallContext.isGuest`
- B. ... `UserType`

- C. UserInfo.getUserType()
- D. ccrz.cc_CallContext.currUser.isGuest

Answer: A ([LEAVE A REPLY](#))

NEW QUESTION: 81

A product is assigned to the entitlement policy but is missing from the Price Book related to the Buyer Group.

The entitlement policy has View products and View prices in catalog checked.

How will the product behave on the B2B Portal?

- A. The product will appear on the Portal with Price Unavailable status and can also be added to the cart.
- B. The product will appear on the Portal with Price Unavailable status but cannot be added to the cart.
- C. The product will not appear on the Portal and therefore cannot be added to the cart.
- D. The product will not appear on the Portal but can be searched since it is part of the entitlement policy.

Answer: C ([LEAVE A REPLY](#))

NEW QUESTION: 82

What interface must a developer implement to override Tax in Checkout?

- A. sfdc.checkout.CartTaxCalculations
- B. sfdc.commerce.TaxCalculations
- C. sfdc_commerce.CartTaxCalculations
- D. sfdc_checkout.TaxCalculations
- E. sfdc.commerce.CheckoutTaxCalculations

Answer: ([SHOW ANSWER](#))

To override tax in checkout, a developer must implement the sfdc_checkout.TaxCalculations interface. The sfdc_checkout.TaxCalculations interface is an Apex interface that provides methods for customizing the tax calculation logic for a cart or an order. The interface allows the developer to define how to retrieve tax rates and rules, how to apply tax amounts and adjustments, and how to handle tax errors and exceptions. The developer can use the sfdc_checkout.TaxCalculations interface to integrate with a third-party tax service provider or to implement their own tax calculation logic. The other interfaces do not exist or are not related to tax calculation. Salesforce Reference: B2B Commerce Developer Guide: Tax Integration, B2B Commerce Developer Guide: Tax Calculations Interface

NEW QUESTION: 83

How are variables bound when services use the ccServiceDao class to execute queries?

- A. String substitution
- B. Apex local variables
- C. Global variables

D. Apex class variables

Answer: ([SHOW ANSWER](#))

NEW QUESTION: 84

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 when to export data from Salesforce or import data into Salesforce?

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: ([SHOW ANSWER](#))

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 References: [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]

NEW QUESTION: 85

What is a best practice when passing query parameters from user interface to an apex controller?

- A. Query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component.
- B. String parameters should be trimmed using String.trim().
- C. Query parameters should be passed only to Salesforce B2B Commerce classes that you are extending.
- D. Query parameters should be stored on a backbone model prior to passing them to the server

Answer: A (LEAVE A REPLY)

A best practice when passing query parameters from user interface to an apex controller is to query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component. This function will encode any special characters in the query parameters to prevent cross-site scripting (XSS) attacks or SOQL injection attacks. For example, ccrz.ccRemoteActions.getProducts('{!JSINHTMLENCODE(searchTerm)}') will encode the searchTerm parameter before passing it to the apex controller. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Security

NEW QUESTION: 86

What is true regarding adding more Configuration Settings to Salesforce B2B Commerce?

- A. Metadata can be added to existing modules, but you cannot add new modules.
- B. Configuration settings can only be extended through API's
- C. Select "New" in your storefront's Configuration Settings and create a custom setting.
- D. More modules and metadata can be added to Salesforce B2B Commerce.

Answer: D (LEAVE A REPLY)

Explanation

More modules and metadata can be added to Salesforce B2B Commerce by creating custom configuration settings. Configuration settings are custom settings that store various values and parameters that affect the functionality and appearance of the storefront. They are organized into

modules, which group related settings together. To create a custom configuration setting, the user needs to create a custom setting record in Salesforce and specify its module, name, value, and description. The custom setting will then appear in CCAdmin under the specified module. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Configuration Settings

NEW QUESTION: 87

What is a best practice when passing query parameters from user interface to an apex controller?

- A.** Query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component.
- B.** String parameters should be trimmed using String.trim().
- C.** Query parameters should be passed only to Salesforce B2B Commerce classes that you are extending.
- D.** Query parameters should be stored on a backbone model prior to passing them to the server

Answer: A (LEAVE A REPLY)

A best practice when passing query parameters from user interface to an apex controller is to query parameters should be properly sanitized by using JSINHTMLENCODE within the VisualForce Page or Component. This function will encode any special characters in the query parameters to prevent cross-site scripting (XSS) attacks or SOQL injection attacks. For example, `ccrz.ccRemoteActions.getProducts('{!JSINHTMLENCODE(searchTerm)}')` will encode the searchTerm parameter before passing it to the apex controller. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Security

NEW QUESTION: 88

Which technique can be used with Lightning web components to expose them outside of an org in another web container?

- A.** Slot elements
- B.** Heroku
- C.** Lightning Out
- D.** Lightning Canvas

Answer: C (LEAVE A REPLY)

Explanation

According to the Lightning Web Components Developer Guide, Lightning Out is a technique that allows developers to run Lightning web components outside of Salesforce servers, such as in a Node.js app running on Heroku or a department server inside the firewall. Lightning Out uses a script tag to load the Lightning web components framework and the custom components into the web container. Lightning Out also handles authentication, event handling, and data access between the web container and the Salesforce org. Slot elements, Heroku, and Lightning Canvas are not techniques for exposing Lightning web components outside of an org in another web container. References: Lightning Web Components Developer Guide, Use Components Outside Salesforce, B2B Commerce and D2C Commerce Developer Guide

NEW QUESTION: 89

Salesforce B2B leverages global API's for encapsulating business logic into blocks that can be extended and modified by subscribers. Which three statements are true regarding extending ccServiceProduct and exposing custom fields on the Product Detail Page? (3 answers)

- A. Override the getFieldsMap method and add subscriber specific code.
- B. Ensure that any helper methods are defined as private and static only.
- C. Create a global with sharing class that extends ccrz.ccServiceProduct.
- D. Create a public with sharing class that extends ccrz.ccServiceProduct.
- E. Override the fetch method and add your subscriber specific code here.

Answer: A,C,E (LEAVE A REPLY)

To extend ccServiceProduct and expose custom fields on the Product Detail Page, three statements are true:

Override the getFieldsMap method and add subscriber specific code. This method returns a map of field names and field types for the product entity and its related entities. By overriding this method, the subscriber can add their custom fields to the map and make them available for the API.

Create a global with sharing class that extends ccrz.ccServiceProduct. This class will inherit all the methods and properties of the ccServiceProduct class and allow overriding or extending them. The class should be global and with sharing to ensure that it can be accessed by the API framework and respect the sharing rules.

Override the fetch method and add your subscriber specific code here. This method executes the query to fetch the product data and returns a result object. By overriding this method, the subscriber can modify the query or the result object to include their custom fields or logic.

Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Service Classes, ccServiceProduct Class

NEW QUESTION: 90

A developer needs to create an event listener on a parent component programmatically. With the script below, what should replace the text <EVENT_LISTENER_UNE>?

```
import { LightningElement } from 'lwc';
export default class Parent extends LightningElement {
  constructor() {
    super();
    <EVENT_LISTENER_UNE>
  }
  handleNotification = () => {};
}
```

- A. `this.template.addEventListener(handleNotification);`
- B. `this.template.addEventListener(this.handleNotification);`
- C. `this.template.addEventListener(notification1, this.handleNotification);`
- D. `addEventListener('notification', this.handleNotification);`

Answer: C (LEAVE A REPLY)

To create an event listener on a parent component programmatically, the developer should use the following line of code:

`this.template.addEventListener('notification', this.handleNotification);` This line of code adds an event listener to the template element of the parent component, which is the root element that contains all the child elements. The event listener listens for an event named 'notification', which is a custom event that can be dispatched by any child component. The event listener invokes a method named `handleNotification`, which is an arrow function defined in the parent component class. The `handleNotification` method receives the event object as a parameter and can perform any logic based on the event data. The other lines of code are either incorrect or incomplete.

Salesforce Reference: Lightning Web Components Developer Guide: Communicate with Events, Lightning Web Components Developer Guide: Create and Dispatch Events

NEW QUESTION: 91

Which two statements are true for Mass Order (2 answers)

- A. Mass Order pricing is done via a batch job.
- B. Mass order works with the default wishlists
- C. The variation product is leveraged for SKUs.
- D. Mass Order is mobile ready with the ccrz templates.

Answer: A,C (LEAVE A REPLY)

Explanation

Mass Order pricing is done via a batch job, which means that the prices are not calculated in real time, but rather at a scheduled time. Mass order works with the variation product, which is a product that has multiple SKUs with different attributes, such as size or color. Mass order does not work with the default wishlists, which are used to store products that the customer wants to buy later. Mass order is not mobile ready with the ccrz templates, which are the default templates for the storefront pages. Salesforce References: B2B Commerce Developer Guide: Mass Order, B2B Commerce Developer Guide: Variation Products

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: <https://www.actual4test.com/B2B->

NEW QUESTION: 92

A configuration value, CO.NewOrder, is set to TRUE. What is one way of preventing an existing payment page from being shown on the checkout payment page?

- A. Delete the Visualforce page from the code base.
- B. Remove the value matching the page name from the pmt.whitelist configuration setting, then rebuild and activate a new Configuration cache
- C. Remove the payment type associated with the payment page from CO.pmts, then rebuild and activate a new cache.
- D. Override the front end template and modify the way the embedded payment page gets loaded from the payment list configuration.

Answer: C ([LEAVE A REPLY](#))

Explanation

One way of preventing an existing payment page from being shown on the checkout payment page is to remove the payment type associated with the payment page from CO.pmts, then rebuild and activate a new cache. This will exclude the payment type from the list of available payment options on the checkout page.

The other options are either not feasible or not effective in hiding the payment page. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Payment Configuration Settings

NEW QUESTION: 93

In which three ways can Salesforce B2B Commerce API sizing blocks support multiple API sizing requests? (3 answers)

- A. When different entities are specified in the method invocation.
- B. The sizing block is not removed.
- C. SZ_ASSC is used.
- D. The sizing block is removed after the first handler.
- E. SZ_ASSC is not used.

Answer: A,C,E ([LEAVE A REPLY](#))

Explanation

Salesforce B2B Commerce API sizing blocks can support multiple API sizing requests in three ways:

* When different entities are specified in the method invocation. For

example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M, ccrz.ccAPI.SZ_L)` will use the SZ_M sizing block for the product entity and the SZ_L sizing block for the related entities.

* SZ_ASSC is used. This flag indicates that the associated entities should use the same sizing block as the primary entity. For example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M,`

ccrz.ccAPI.SZ_ASSC) will use the SZ_M sizing block for both the product entity and the related entities.

* SZ_ASSC is not used. This flag indicates that the associated entities should use the default sizing block, which is SZ_M, unless otherwise specified. For example, ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M) will use the SZ_M sizing block for the product entity and the SZ_M sizing block for the related entities. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Data Sizing Conventions

NEW QUESTION: 94

What are the templating, Javascript, and CSS frameworks what the cloudcraze managed package leverages?

- A. Angularjs, Backbonejs, and handlebarsjs
- B. Angularjs, react.js, and handlebarsjs
- C. Bootstrap, Angularjs, and Backbonejs
- D. Bootstrap, Backbonejs, and handlebarsjs

Answer: D ([LEAVE A REPLY](#))

NEW QUESTION: 95

Which two scoped modules should a developer import in Lightning web components to check user permissions?

- A. @salesforce/permission
- B. @salesforce/customPermission
- C. @salesforce/hasPermission
- D. @salesforce/userPermission

Answer: A,B ([LEAVE A REPLY](#))

To check user permissions in Lightning web components, a developer should import two scoped modules: @salesforce/permission and @salesforce/customPermission. The @salesforce/permission module allows the developer to check if the user has access to a standard permission, such as View Setup or Modify All Data. The @salesforce/customPermission module allows the developer to check if the user has access to a custom permission, such as Enable Debug Mode or Manage Orders. The other modules do not exist or are not related to user permissions. Salesforce Reference: Lightning Web Components Developer Guide: Check User Permissions, Lightning Web Components Developer Guide: Import Salesforce Modules

NEW QUESTION: 96

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?

- 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: (SHOW ANSWER)

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

NEW QUESTION: 97

Which two log levels does a developer need to include to debug platform events?

- A.** Apex Code
- B.** Database
- C.** Callout

D. Workflow

Answer: (SHOW ANSWER)

Explanation

According to the Platform Events Developer Guide, to debug platform events, a developer needs to include the Apex Code and Workflow log levels in the trace flag entry for the Automated Process entity or the overridden user. The Apex Code log level captures the execution of the Apex trigger that subscribes to the platform event. The Workflow log level captures the execution of the event process or the resumed flow interview that subscribes to the platform event. The other log levels are not relevant for debugging platform events. The Database log level captures database operations, such as DML statements and SOQL queries. The Callout log level captures the details of HTTP requests and responses, such as web service callouts. References: Platform Events Developer Guide, Set Up Debug Logs for Event Subscriptions, Set Up Debug Logging

NEW QUESTION: 98

A developer exports data from an org on a standard entity which has a custom attribute. When they launch Data Loader, select the entity, click the Select All Fields button and click Finish, the custom field they added called MyCustomField__c has no values and no column header in the CSV file. What is the root cause?

- A. The user needs to install a specific Zulu JDK that is recommended by Salesforce.
- B. A mapping file was not used when the data was loaded in
- C. The user does not have access to the field
- D. The user has rights to the field but there are no values in it

Answer: C (LEAVE A REPLY)

The root cause of why the custom field MyCustomField__c has no values and no column header in the CSV file is that the user does not have access to the field. A user's access to a field is determined by their profile and permission sets, which define their field-level security settings. Field-level security settings control whether a user can see, edit, or delete the value for a particular field on an object. If a user does not have access to a field, they will not be able to view or modify its value in any interface, including Data Loader. Data Loader is a tool that allows users to import or export data between Salesforce and CSV files. When using Data Loader to export data from an org, Data Loader will only include fields that are accessible to the user based on their field-level security settings. If a user does not have access to a field, Data Loader will not include that field in the CSV file, neither as a column header nor as a value. The user needs to install a specific Zulu JDK that is recommended by Salesforce is not the root cause of why MyCustomField__c has no values and no column header in the CSV file, as it is not related to field access or Data Loader functionality. A mapping file was not used when the data was loaded in is not the root cause either, as it is not related to field access or Data Loader functionality. A mapping file is an optional file that maps fields between Salesforce objects and CSV files when using Data Loader to import or export data. The user has rights to the field but there are no values in it is not the root cause either, as it contradicts the fact that MyCustomField__c has no column header in the CSV file. If the user had rights to the field but there were no values in it,

Data Loader would still include MyCustomField__c as a column header in the CSV file, but leave its values blank. Salesforce Reference: [Data Loader Guide: Export Data from Salesforce], [Data Loader Guide: Field Mapping], [Salesforce Help: Set Field-Level Security]

NEW QUESTION: 99

How is a price group dynamically set?

- A. By overriding the ccLogicProductPrice class
- B. By using contract pricing
- C. By extending the ccApiPriceList API
- D. By extending the cc_hk_pricing hook

Answer: D (LEAVE A REPLY)

A price group can be dynamically set by extending the cc_hk_pricing hook. This hook allows modifying the price group based on various factors, such as the user, cart, product, or storefront. For example, the hook can apply a different price group for a specific product category or for a specific user segment.

NEW QUESTION: 100

Which three statements are true about Global API versioning? (3 answers)

- A. Calling in with an API version set to lower than 1 will result in an exceptional case where the exception class `ccrz.BelowMinAPIVersionException` will be returned to callers.
- B. There is no need to pass `API_VERSION` to the Global APIs, and based on the Salesforce B2B Commerce Managed Package version, Global APIs are able to figure out what version of the API to use.
- C. The API version is scoped at the Class API level and NOT at the method level.
- D. Minimum `API_VERSION` is 1 and the Maximum API version follows the releases. E.g. The maximum was 4 as of Salesforce B2B Commerce Release-4.5, 5 as of Salesforce B2B Commerce Release 4.6, etc.
- E. Calling in with an API version set to more than current maximum will result in exception case where the exception class `ccrz.ExceedsMaxAPIVersionException` will be returned to callers.

Answer: A,D,E (LEAVE A REPLY)

Explanation

Three statements that are true about Global API versioning are:

- * Calling in with an API version set to lower than 1 will result in an exceptional case where the exception class `ccrz.BelowMinAPIVersionException` will be returned to callers. This exception indicates that the API version is not supported by the framework and the caller should use a higher API version.
- * The API version is scoped at the Class API level and NOT at the method level. This means that all the methods in a class will use the same API version that is specified by the caller. For example, if the caller passes an API version of 4 to `ccrz.ccServiceProduct.getProducts()`, then all the other methods in `ccrz.ccServiceProduct` will also use API version 4.

* Calling in with an API version set to more than current maximum will result in exception case where the exception class `ccrz.ExceedsMaxAPIVersionException` will be returned to callers. This exception indicates that the API version is not supported by the framework and the caller should use a lower API version. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, API Versioning

NEW QUESTION: 101

What does a developer need to do to modify the out-of-the-box checkout flow template?

- A. Clone, modify, activate and refer in Experience Builder
- B. Modify directly and save to activate
- C. Create each flow from scratch
- D. Clone, modify and rename to Checkout Flow

Answer: A (LEAVE A REPLY)

To modify the out-of-the-box checkout flow template in Salesforce B2B Commerce, a developer should clone the existing template, make the necessary modifications, activate the modified template, and then reference it in the Experience Builder. This approach ensures that the original template remains intact and provides a fallback option. Salesforce documentation on customizing the checkout flow in B2B Commerce emphasizes the importance of using the Experience Builder for such customizations, providing a visual interface to manage and reference different checkout flow templates.

NEW QUESTION: 102

Numerous flags ... have a direct impact on the result set provided by the Global API's. What Global API Data-Sizing convention flag prevents an API request from propagating to further requests when provided as a Boolean parameter with a value of true?

- A. `ccrz.ccAPI.SZ_REL`
- B. `ccrz.ccAPI.SZ_ASSC`
- C. `ccrz.ccAPISizing.ASSC`
- D. `ccrz.ccAPISizing.REL`

Answer: (SHOW ANSWER)

The Global API Data-Sizing convention flag that prevents an API request from propagating to further requests when provided as a Boolean parameter with a value of true is `ccrz.ccAPI.SZ_ASSC`. This flag indicates that only one API request should be executed with the specified sizing block, and any subsequent requests should use their own default sizing blocks. For example, `ccrz.ccServiceCart.getCart(ccrz.ccAPI.SZ_L,true)` will use the `SZ_L` sizing block for retrieving the cart data, but any other requests that are triggered by this method will use their own default sizing blocks. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Data Sizing Conventions

NEW QUESTION: 103

Which element can be used to pass HTML from a parent component to a child component? 03m
19s

- A. <html></html>
- B. <template></template>
- C. <p></p>
- D. <slot></slot>

Answer: D ([LEAVE A REPLY](#))

Explanation

To pass HTML from a parent component to a child component, the child component needs to have a <slot></slot> element in its template. A slot is a placeholder for markup that a parent component passes into a component's body. Slots are part of the Web Component specification. The parent component can use the slot name or the default slot to specify which HTML content goes into which slot of the child component. The other options are not valid elements for passing HTML from parent to child. References:

- * Slots
- * Pass HTML Markup From Parent TO CHILD In Lightning Web Component
- * How parent component pass HTML to child component in Angular?

NEW QUESTION: 104

Which two aspects are applicable to Page Includes? (2 answers)

- A. Standard Visualforce controls such as apex:form should not be used within a page include
- B. Page Includes must be assigned to an OOTB Page, i.e. Home, Product Detail, etc., and enabled
- C. Page Includes can be configured as Body Includes Begin.
- D. If a controller is used for an included page, then a merge variable must be present on the page.

Answer: ([SHOW ANSWER](#))

Explanation

Two aspects that are applicable to Page Includes are:

- * Standard Visualforce controls such as apex:form should not be used within a page include. This is because the page include is rendered inside an existing Visualforce page that already has a form element. Using another form element inside the page include will cause conflicts and errors. Instead, the page include should use HTML elements or custom components that do not require a form element.
- * Page Includes can be configured as Body Includes Begin. This means that the page include will be rendered at the beginning of the body section of the page, before any other content or widgets. This can be useful for adding some custom content or functionality that applies to the whole page, such as a banner, a modal, or a script. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Page Includes

NEW QUESTION: 105

A developer is trying to integrate a new shipping provider to use during checkout in a storefront. Which two steps must the developer take to make an integration available for selection?

- A. Create a RegisteredExternalService record using Workbench.
- B. Create an Apex class that uses the integration framework.
- C. Modify the StoreIntegratedService to map to an Apex class ID using Workbench.
- D. Enter the integration class name and version in the store administration.

Answer: A,B (LEAVE A REPLY)

To make an integration available for selection, a developer must create a RegisteredExternalService record using Workbench and create an Apex class that uses the integration framework. Creating a RegisteredExternalService record using Workbench allows the developer to register their custom integration class as an external service in Salesforce B2B Commerce. The RegisteredExternalService record contains information such as the class name, version, display name, description, and category of the integration class.

The category determines where and how the integration class can be used in B2B Commerce, such as ShippingService or TaxService. Creating an Apex class that uses the integration framework allows the developer to define custom logic for integrating with an external service provider's API or service. The integration framework provides interfaces and classes for various types of integrations, such as shipping, tax, payment, inventory, and freight. The developer can implement these interfaces and classes in their custom Apex class and override their methods with their own logic. Modifying the StoreIntegratedService to map to an Apex class ID using Workbench is not a required step for making an integration available for selection, as it is only used for registering internal services that are provided by Salesforce B2B Commerce out-of-the-box.

Entering the integration class name and version in store administration is not a required step either, as it is only used for selecting an existing integration class that has already been registered as an external service.

Salesforce References: [B2B Commerce Developer Guide: Integration Framework], [B2B Commerce Developer Guide: RegisteredExternalService Object]

NEW QUESTION: 106

A developer is trying to troubleshoot why a field is not displaying on the Product Detail Page. What should be typed in the Developer Tools Console in the browser to view the fields available for the Product Detail Page?

- A. CCRZ.cartView
- B. CCRZ.productDetailView
- C. CCRZ.productSearchView
- D. CCRZ.productDetailModel

Answer: D (LEAVE A REPLY)

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 107

A query containing a subquery is executed. What is appended to the subquery name as part of its transformation by default in Salesforce B2B Commerce?

- A. A subscriber-supplied token
- B. "__ccrz"
- C. The "*" symbol
- D. The letter "S"

Answer: B (LEAVE A REPLY)

A query containing a subquery is executed. By default, in Salesforce B2B Commerce, "__ccrz" is appended to the subquery name as part of its transformation. This is done to avoid conflicts with the standard Salesforce fields and relationships. For example, SELECT Id, Name, (SELECT Id, Name FROM Contacts) FROM Account will be transformed to SELECT Id, Name, (SELECT Id, Name FROM Contacts__ccrz) FROM Account__ccrz. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 108

Which wire adapter can a developer use to retrieve metadata about a specific object?

- A. getObject
- B. getObjectMetadata
- C. All of the above
- D. getObjectInfo

Answer: D (LEAVE A REPLY)

To retrieve metadata about a specific object in a Lightning Web Component, a developer can use the getObjectInfo wire adapter. This adapter provides access to the metadata of a specified Salesforce object, including fields, record types, and layouts, which is essential for dynamic component rendering based on the object's schema.

NEW QUESTION: 109

A configuration value, CO.NewOrder, is set to TRUE. What is one way of preventing an existing payment page from being shown on the checkout payment page?

- A. Delete the Visualforce page from the code base.

- B.** Remove the value matching the page name from the pmt.whitelist configuration setting, then rebuild and activate a new Configuration cache
- C.** Remove the payment type associated with the payment page from CO.pmts, then rebuild and activate a new cache.
- D.** Override the front end template and modify the way the embedded payment page gets loaded from the payment list configuration.

Answer: C (LEAVE A REPLY)

One way of preventing an existing payment page from being shown on the checkout payment page is to remove the payment type associated with the payment page from CO.pmts, then rebuild and activate a new cache. This will exclude the payment type from the list of available payment options on the checkout page. The other options are either not feasible or not effective in hiding the payment page. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Payment Configuration Settings

NEW QUESTION: 110

Which two statements are true for Mass Order (2 answers)

- A.** The variation product is leveraged for SKUs.
- B.** Mass order works with the default wishlists
- C.** Mass Order is mobile ready with the ccrz templates.
- D.** Mass Order pricing is done via a batch job.

Answer: A,D (LEAVE A REPLY)

NEW QUESTION: 111

Salesforce B2B leverages global API's for encapsulating business logic into blocks that can be extended and modified by subscribers. Which three statements are true regarding extending ccServiceProduct and exposing custom fields on the Product Detail Page? (3 answers)

- A.** Ensure that any helper methods are defined as private and static only.
- B.** Create a global with sharing class that extends ccrz.ccServiceProduct.
- C.** Override the fetch method and add your subscriber specific code here.
- D.** Create a public with sharing class that extends ccrz.ccServiceProduct.
- E.** Override the getFieldsMap method and add subscriber specific code.

Answer: B,C,E (LEAVE A REPLY)

NEW QUESTION: 112

During checkout flow customizations, a developer receives an error on shipping cost calculation integrations with the error code: INSUFFICIENT_ACCESS_OR_READONLY.

What is causing this error?

- A.** The storefront user does not have access to the Cart Delivery Method object.
- B.** An error has occurred during the cart shipping charge integration.
- C.** The storefront user does not have access to custom fields on the Order Delivery Method object.

D. The cart is no longer in a valid Checkout State.

Answer: D ([LEAVE A REPLY](#))

The error code `INSUFFICIENT_ACCESS_OR_READONLY` is caused by the cart being no longer in a valid Checkout State during checkout flow customizations. A cart is an object that represents a collection of products and charges that a customer intends to purchase in the storefront. A cart has a Checkout State field that indicates the current state of the checkout process for the cart. The Checkout State can have values such as Draft, InProgress, Completed, or Cancelled. A cart can only be modified or updated when it is in Draft or InProgress state. A cart cannot be modified or updated when it is in Completed or Cancelled state. If an attempt is made to modify or update a cart that is in Completed or Cancelled state, an error with the code `INSUFFICIENT_ACCESS_OR_READONLY` will be thrown. This error means that the user does not have permission to edit or delete a record because it is read-only or locked. The storefront user does not have access to the Cart Delivery Method object is not a cause of this error code, as it is not related to the cart checkout state or data modification. The Cart Delivery Method object is an object that stores information about the delivery method selected for a cart in the storefront. An error has occurred during the cart shipping charge integration is not a cause of this error code either, as it is not related to the cart checkout state or data modification. The cart shipping charge integration is an integration that calculates and applies shipping charges to a cart based on various factors such as delivery method, location, weight, volume, etc. The storefront user does not have access to custom fields on the Order Delivery Method object is not a cause of this error code either, as it is not related to the cart checkout state or data modification. The Order Delivery Method object is an object that stores information about the delivery method selected for an order summary in the storefront. Salesforce Reference: [B2B Commerce Developer Guide: Cart Object](#), [\[B2B Commerce Developer Guide: Cart Delivery Method Object\]](#), [\[B2B Commerce Developer Guide: Order Delivery Method Object\]](#), [\[Salesforce Help: Common Error Messages\]](#)

NEW QUESTION: 113

A developer attempts to export data from an org by launching Data Loader, selecting a standard entity, clicking the "Select All Fields" button and clicking the Finish button. The developer finds that the CustomField_c field they added to the entity has no values under the header in the CSV file output. What is the root cause?

- A.** The developer does not have the correct JDK that is recommended by Salesforce and this is known to cause issues with exporting custom attributes
- B.** The developer does not have access to the object's metadata
- C.** The field is not populated
- D.** The user does not have rights to the custom field

Answer: (SHOW ANSWER)

The root cause of why the CustomField__c field they added to the entity has no values under the header in the CSV file output is that the user does not have rights to the custom field. A user's access to a field is determined by their profile and permission sets, which define their field-level security settings. Field-level security settings control whether a user can see, edit, or delete the

value for a particular field on an object. If a user does not have access to a field, they will not be able to view or modify its value in any interface, including Data Loader. Data Loader is a tool that allows users to import or export data between Salesforce and CSV files. When using Data Loader to export data from an org, Data Loader will only include fields that are accessible to the user based on their field-level security settings. If a user does not have access to a field, Data Loader will not include that field in the CSV file, neither as a column header nor as a value. The developer does not have the correct JDK that is recommended by Salesforce and this is known to cause issues with exporting custom attributes is not the root cause of why CustomField__c has no values under the header in the CSV file output, as it is not related to field access or Data Loader functionality. The developer does not have access to the object's metadata is not the root cause either, as it is not related to field access or Data Loader functionality. The object's metadata defines its structure and properties, such as fields, relationships, and layouts. The field is not populated is not the root cause either, as it contradicts the fact that CustomField__c has no column header in the CSV file output. If the field was not populated but accessible to the user, Data Loader would still include CustomField__c as a column header in the CSV file, but leave its values blank.

Salesforce References: Data Loader Guide: Export Data from Salesforce, [Data Loader Guide: Field Mapping], [Salesforce Help: Set Field-Level Security]

NEW QUESTION: 114

While in the process of gathering requirements from a customer about how they would like to set up their net new storefront checkout experience, a consultant learns that the customer needs the ability to add new shipping and billing addresses during checkout.

Which approach should a developer take to meet this requirement?

- A.** Create a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component.
- B.** Enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow.
- C.** Enable Buyer Managed Contact Point Addresses within Commerce Administration.
- D.** Create a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow.

Answer: B (LEAVE A REPLY)

To enable the ability to add new shipping and billing addresses during checkout, a developer should enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow. The Buyer Managed Contact Point Addresses is a feature that allows customers to add, edit, or delete their shipping and billing addresses during checkout. The developer can enable this feature by setting the `buyerManagedContactPointAddressesEnabled` attribute to true in the Shipping Address standard component in the Checkout subflow. The Shipping Address standard component is a component that displays and collects the shipping address information for the cart or order. The Checkout subflow is a subflow that defines the steps and components for the checkout process in the

storefront. Creating a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component is not a valid way to enable this feature, as there is no such component as Buyer Managed Contact Point Addresses. Enabling Buyer Managed Contact Point Addresses within Commerce Administration is not a valid way either, as this feature is not configurable in Commerce Administration. Creating a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow is not a valid way either, as this is unnecessary and complicated when there is already a standard component that supports this feature. Salesforce Reference: [B2B Commerce Developer Guide: Buyer Managed Contact Point Addresses], [B2B Commerce Developer Guide: Shipping Address Component], [B2B Commerce Developer Guide: Checkout Subflow]

NEW QUESTION: 115

What are two advantages of using Lightning Data Service?

- A. Communicates with other components
- B. Converts between different data formats
- C. Combines and de-duplicates server calls
- D. Loads record data progressively

Answer: (SHOW ANSWER)

Explanation

Two advantages of using Lightning Data Service are that it combines and de-duplicates server calls and that it loads record data progressively. Lightning Data Service is a service that provides access to Salesforce data and metadata in Lightning web components. It optimizes performance and minimizes server round trips by caching data on the client side and sharing data across components. It also combines and de-duplicates server calls by batching requests for the same record or object and returning a single response. It also loads record data progressively by returning available cached data first and then fetching updated data from the server asynchronously. Communicating with other components and converting between different data formats are not advantages of using Lightning Data Service, as they are not related to its functionality or features. Salesforce References: Lightning Web Components Developer Guide: Lightning Data Service, Lightning Web Components Developer Guide: Work with Salesforce Data

NEW QUESTION: 116

The `ccrz.cc_hk_UserInterface` apex class, HTML Head Include Begin and HTML Head Include End Cloudcraze Page Include sections allow additional content to be added to the HTML `<head>` tag. What are two reasons that it is preferred to use the `ccrz.cc_hk_UserInterface` extension over the Cloudcraze Page Include sections? (2 answers)

- A. HTML does not support `<div>` tags inside the `<head>`
- B. Salesforce `apex:include` is wrapped in `` tags.
- C. HTML does not support `` tags inside the `<head>`
- D. Salesforce `apex:include` is wrapped in tags.

Answer: B,C (LEAVE A REPLY)

NEW QUESTION: 117

Which orders can a user view on Order List Component (Aura) for B2B stores?

- A. Orders placed in the current storefront and related to the current effective account
- B. Orders placed in any storefront and related to the current effective account
- C. Orders placed in the current storefront and related to the current user's account
- D. Orders placed in any storefront and related to the current user's account

Answer: A ([LEAVE A REPLY](#))

Explanation

According to the B2B Commerce Developer Guide, the Order List Component (Aura) for B2B stores displays a list of orders that the user can view and reorder. The component filters the orders based on the current storefront and the current effective account. The current effective account is the account that the user is currently acting on behalf of, which can be different from the user's own account. The component does not show orders placed in other storefronts or related to other accounts, even if the user has access to them.

Therefore, the correct answer is A. Orders placed in the current storefront and related to the current effective account. The other options are incorrect because they do not match the criteria of the component. References: B2B Commerce Developer Guide, Show the Order List Widget

NEW QUESTION: 118

What tool can a developer use to investigate errors during development?

- A. Commerce Diagnostics Event Logging
- B. Checkout Flow Log
- C. Support cases
- D. Browser dev tools

Answer: ([SHOW ANSWER](#))

Browser dev tools are a set of web authoring and debugging tools built into most modern browsers. They allow developers to inspect, edit, and debug the HTML, CSS, JavaScript, and network activity of a web page.

They can also provide useful information about errors, warnings, performance, and accessibility issues.

Browser dev tools are especially helpful for developing and testing Lightning web components, as they can display the component hierarchy, attributes, events, and slots.

The other options are not correct because:

- * A. Commerce Diagnostics Event Logging is a feature that enables developers to capture and analyze events that occur during the execution of B2C Commerce code. It can help identify performance bottlenecks, memory leaks, and unexpected behavior. However, it is not a tool that can be used directly by the developer, but rather a service that requires a support request to enable and access.
- * B. Checkout Flow Log is a log file that shows the details of the checkout flow execution, such as the input and output parameters, the pipeline steps, and the errors and warnings. It can help

troubleshoot issues related to the checkout process, such as payment, shipping, or tax calculation. However, it is not a tool that can be used during development, but rather a log file that can be accessed after the checkout flow has run.

* C. Support cases are requests for assistance from the Salesforce support team. They can help resolve technical issues, provide guidance, or escalate bugs. However, they are not a tool that can be used to investigate errors during development, but rather a communication channel that can be used after the developer has exhausted other resources.

References:

- * Browser Dev Tools
- * Debug Your Lightning Web Components
- * Commerce Diagnostics Event Logging
- * [Checkout Flow Log]

NEW QUESTION: 119

Which service method should be overridden in order to allow "without sharing" queries?

- A. `ccrz.ccService.initSVCDAO()`
- B. `ccrz.ccService.query()`
- C. `ccrz.ccService.queryWithoutSharing()`
- D. `ccrz.ccAPI.queryService()`

Answer: ([SHOW ANSWER](#))

NEW QUESTION: 120

Which method is used to override when extending the Salesforce B2B Commerce logic providers?

- A. `doLogic`
- B. `process`
- C. `doAction`
- D. `fetch`

Answer: ([SHOW ANSWER](#))

The method that is used to override when extending the Salesforce B2B Commerce logic providers is `doLogic`.

This method is responsible for executing the business logic for each logic provider class, such as validating inputs, performing calculations, applying rules, or updating data. By overriding this method, the user can modify or extend the existing logic or add their own custom logic.

Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Logic Classes

NEW QUESTION: 121

What is default behavior for how the Salesforce B2B Commerce Global APIs transform Salesforce data?

- A. Fields names are returned using the Salesforce naming convention.
- B. Fields names are returned with prepended in their name.

C. Fields names are returned with a lowercase first letter, camelcase convention

D. Fields names can be mapped to any naming convention desired

Answer: (SHOW ANSWER)

Explanation

The default behavior for how the Salesforce B2B Commerce Global APIs transform Salesforce data is to return field names with a lowercase first letter, camelcase convention. For example, the field name `ccrz__E_Product__c` in Salesforce will be transformed to `eProduct` in the API. This is done to follow the JavaScript naming convention and to avoid conflicts with the standard Salesforce fields and relationships.

Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 122

Which event is triggered within Salesforce B2B Commerce whenever a cart's state changes?

A. `cartChange`

B. `cart`

C. `pageMessage`

D. `cartState`

Answer: A (LEAVE A REPLY)

The event that is triggered within Salesforce B2B Commerce whenever a cart's state changes is `cartChange`. This event is triggered by the `updateCart` method of the `CCRZ.Cart` class, which is responsible for updating the cart data and performing any actions that depend on the cart state, such as applying discounts, taxes, or shipping charges. The event can be used to notify all the components that are listening to it that the cart has changed and they should refresh their data accordingly. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Events, Cart Class

NEW QUESTION: 123

How can a developer introduce new screen behavior in a checkout flow step?

A. Modify the property mappings in `checkoutSteps.xml`

B. Edit the default subflow directly

C. Clone the appropriate subflow and replace the Lightning web components in it

D. Adjust next-state in previous subflow configuration

Answer: D (LEAVE A REPLY)

To introduce new screen behavior in a checkout flow step, a developer should clone the appropriate subflow and replace the Lightning web components in it. A subflow is a reusable flow that can be invoked from another flow as an element. A checkout flow is a flow that defines the steps and components for the checkout process in the storefront. A checkout flow consists of several subflows, each of which corresponds to a checkout step, such as Cart, Shipping Address, Payment, or Order Summary. To introduce new screen behavior in a checkout flow step, a developer should clone the subflow that matches the checkout step they want to modify and replace the Lightning web components in the cloned subflow with their custom components. The developer can then update the checkoutSteps.xml file to point to the cloned subflow instead of the original subflow. Modifying the property mappings in checkoutSteps.xml is not a valid way to introduce new screen behavior in a checkout flow step, as it only affects how data is passed between subflows, not how screens are displayed or rendered. Editing the default subflow directly is not a valid way either, as it can cause conflicts or errors with other stores or environments that use the same default subflow. Adjusting next-state in previous subflow configuration is not a valid way either, as it only affects how subflows are transitioned or navigated, not how screens are displayed or rendered. Salesforce References: B2B Commerce Developer Guide: Checkout Flow, B2B Commerce Developer Guide: Checkout Subflows, B2B Commerce Developer Guide: Customize Checkout Steps

NEW QUESTION: 124

Which category can receive signaling from a Lightning Message Channel?

A. only descendents (i.e. children)

B. only direct siblings

C. page (anywhere on it)

D. only ancestors (i.e. parents)

Answer: C (LEAVE A REPLY)

A Lightning Message Channel can receive signaling from any component on a page (anywhere on it). A Lightning Message Channel is a service that allows components to communicate with each other across different Salesforce UI technologies, such as Aura, LWC, Visualforce, and Lightning web apps. A component can subscribe to a message channel and receive events that are published by another component on the same or different page. The message channel acts as a mediator that delivers the events to the subscribers. A Lightning Message Channel does not receive signaling only from descendants (i.e., children), only from direct siblings, or only from ancestors (i.e., parents), as these are limitations of event-based communication. Salesforce References: [Lightning Web Components Developer Guide: Communicate Across Salesforce UI Technologies], [Lightning Web Components Developer Guide: Communicate with Events]

NEW QUESTION: 125

In which three ways should useful debugging information in Salesforce B2B Commerce implementation be garnered? (3 answers) A) Enabling the logging token via

A. Admin and subsequently inspecting the logs via the browser console.

B. Logging a case with Salesforce support to enable advanced debugging options.

C. Enabling debugging options for the current user and visually inspecting the Salesforce debug logs.

D. Placing a `System.debug()` statement anywhere in the class being debugged.

E. Logging into the community as a system administrator to identify any potential permissions or Visualforce exceptions.

Answer: A,C,E (LEAVE A REPLY)

Useful debugging information in Salesforce B2B Commerce implementation can be garnered in three ways:

* Enabling the logging token via Admin and subsequently inspecting the logs via the browser console.

This will enable logging messages and errors to the browser console, which can be viewed by opening the Developer Tools in the browser. The logging token can be enabled by setting the value of `CO.logToken` to true in `CCAdmin`.

* Enabling debugging options for the current user and visually inspecting the Salesforce debug logs. This will enable logging messages and errors to the Salesforce debug logs, which can be viewed by opening the Debug Logs page in Salesforce Setup. The debugging options can be enabled by creating a Debug Level and a Trace Flag for the current user in Salesforce Setup.

* Logging into the community as a system administrator to identify any potential permissions or Visualforce exceptions. This will allow viewing any errors or warnings that may occur on the community pages due to insufficient permissions or Visualforce issues. The system administrator can also access `CCAdmin` and other tools from within the community. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Logging, Debug Your Code

NEW QUESTION: 126

In which threeways can Salesforce B2B Commerce API sizing blocks support multiple API sizing requests? (3 answers)

A. When different entities are specified in the method invocation.

B. The sizing block is not removed.

C. `SZ_ASSC` is used.

D. The sizing block is removed after the first handler.

E. `SZ_ASSC` is not used.

Answer: (SHOW ANSWER)

Salesforce B2B Commerce API sizing blocks can support multiple API sizing requests in three ways:

When different entities are specified in the method invocation. For example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M, ccrz.ccAPI.SZ_L)` will use the `SZ_M` sizing block for the product entity and the `SZ_L` sizing block for the related entities.

SZ_ASSC is used. This flag indicates that the associated entities should use the same sizing block as the primary entity. For example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M, ccrz.ccAPI.SZ_ASSC)` will use the SZ_M sizing block for both the product entity and the related entities.

SZ_ASSC is not used. This flag indicates that the associated entities should use the default sizing block, which is SZ_M, unless otherwise specified. For example, `ccrz.ccServiceProduct.getProducts(ccrz.ccAPI.SZ_M)` will use the SZ_M sizing block for the product entity and the SZ_M sizing block for the related entities. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Data Sizing Conventions

NEW QUESTION: 127

What is a valid way of referencing the CC Cart Object whose API name is `E_Cart__c` in a SOQL query?

- A. `ccrz__E_Cart__c`
- B. `c.E_Cart__c`
- C. `cloudcraze__E_Cart__c`
- D. `_Cart__c`

Answer: A ([LEAVE A REPLY](#))

NEW QUESTION: 128

Which Lightning web component path allows a developer to view or edit a record while maintaining control over specifying its layout and set of fields?

- A. `lightning-record-edit-form`
- B. `lightning-record-imperative`
- C. `lightning-record-view-form`
- D. `lightning-record-form`

Answer: A ([LEAVE A REPLY](#))

The `lightning-record-edit-form` component allows a developer to view or edit a record while maintaining control over specifying its layout and set of fields. This component provides a way to create forms that respect the field-level security and layout configuration defined in the Salesforce metadata. The developer can use `lightning-input-field` components inside the form to display and edit the fields of a record. The other options are incorrect because they either do not allow editing the record, do not respect the metadata configuration, or do not exist as valid component paths.

References: `lightning-record-edit-form` | Lightning Web Components Developer Guide

NEW QUESTION: 129

Which event is invoked by any CCRZ Salesforce B2B CommerceView after the view is rendered?

- A. `view:*.load`
- B. `view:*.refresh`
- C. `view:*.onload`
- D. `view:*.rendered`

Answer: D ([LEAVE A REPLY](#))

Explanation

The event that is invoked by any CCRZ Salesforce B2B Commerce View after the view is rendered is view:*.rendered. This event is triggered by the render method of the CCRZ.View class, which is the base class for all views in the framework. The event can be used to perform any actions or logic that depend on the view being rendered, such as initializing widgets, binding events, or updating the user interface. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Events, View Class

NEW QUESTION: 130

Which two Salesforce B2B Commerce visualforce pages must be enabled at a Salesforce Community level to make the out of the box SEO functionality available? (2 answers)

- A. CCSizeIndex
- B. SizeMap
- C. CCCatSiteMap
- D. ProductMap

Answer: A,C ([LEAVE A REPLY](#))

Two Salesforce B2B Commerce Visualforce pages that must be enabled at a Salesforce Community level to make the out of the box SEO functionality available are:

* CCSizeIndex: This page generates a sitemap.xml file, which is a file that lists all the pages and resources on a site that can be crawled by web crawlers. The page uses the configuration settings CO.SiteMapIncludeProducts and CO.SiteMapIncludeCategories to specify which products and categories should be included in the sitemap.

* CCCatSiteMap: This page generates a category sitemap file, which is a file that lists all the categories on a site that can be crawled by web crawlers. The page uses the configuration setting CO.SiteMapCategoryDepth to specify how many levels of subcategories should be included in the category sitemap. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Sitemap Files

NEW QUESTION: 131

Which three file extensions are allowed in a Lightning Web Component folder?

- A. .js-meta.xml
- B. .html
- C. .Js
- D. .gif
- E. .jar

Answer: A,B,C ([LEAVE A REPLY](#))

In a Lightning Web Component folder, the allowed file extensions include.js-meta.xmlfor the component's metadata,.htmlfor the component's markup, and.jsfor the component's JavaScript class. These files are essential for defining the structure, behavior, and metadata of a LWC.

Salesforce LWC documentation provides detailed information on the structure of a LWC bundle and the purpose of each file type within it.

NEW QUESTION: 132

How are variables bound when services use the `ccServiceDao` class to execute queries?

- A. Global variables
- B. Apex local variables
- C. String substitution
- D. Apex class variables

Answer: C ([LEAVE A REPLY](#))

Explanation

When services use the `ccServiceDao` class to execute queries, variables are bound by string substitution. This means that the query string contains placeholders for variables that are replaced by their values at runtime. For example, `ccrz.ccServiceDao.getQuery('SELECT Id FROM Account WHERE Name = :name')` will replace `:name` with the value of the `name` variable.

NEW QUESTION: 133

Where is the API-based record creation generally handled in Salesforce B2B Commerce?

- A. In the methods available in extension hooks
- B. The service-layer responsible for the entity
- C. Data creation is not allowed
- D. Logic classes that implement the business logic for create operations

Answer: B ([LEAVE A REPLY](#))

The API-based record creation is generally handled in the service-layer responsible for the entity in Salesforce B2B Commerce. The service-layer is a set of classes that provide methods for interacting with the data layer and performing business logic. Each entity, such as product, cart, or order, has a corresponding service class that handles the create, read, update, and delete operations for that entity. For example, `ccrz.ccServiceProduct` provides methods for creating and retrieving products. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Service Classes

NEW QUESTION: 134

What are two considerations to keep in mind when including additional JavaScript files in a Lightning web component?

- A. Each additional file needs a corresponding `.js-meta.xml` file.
- B. The files must be ES6 modules and must have names that are unique within the component's folder.
- C. A module can export named functions or variables
- D. Additional JavaScript files should be minified before deployment

Answer: B,C ([LEAVE A REPLY](#))

Explanation

When including additional JavaScript files in a Lightning web component, a developer should keep in mind two considerations: the files must be ES6 modules and must have names that are unique within the component's folder and a module can export named functions or variables. The files must be ES6 modules because LWC uses ES6 modules as the standard for modular JavaScript code. The files must have names that are unique within the component's folder because LWC uses the file name as the module identifier and does not allow duplicate identifiers. A module can export named functions or variables because LWC supports named exports, which allow a module to export multiple values with different names. Each additional file does not need a corresponding .js-meta.xml file, as this is only required for the main JavaScript file of the component. Additional JavaScript files should not be minified before deployment, as this is not necessary or recommended for LWC. Salesforce References: [Lightning Web Components Developer Guide: Include JavaScript Files], [Lightning Web Components Developer Guide: ES6 Modules]

NEW QUESTION: 135

In checkout, what event should the developer's code listen for in order to help troubleshoot and respond to actions?

- A. CommerceBubbleEvents
- B. CommerceErrorEvents
- C. CommerceActionEvents
- D. CommerceDiagnosticEvents

Answer: D (LEAVE A REPLY)

Explanation

To help troubleshoot and respond to actions in checkout, the developer's code should listen for CommerceDiagnosticEvents. CommerceDiagnosticEvents is an event that is fired by Salesforce B2B Commerce when an error occurs in the storefront. CommerceDiagnosticEvents contains information about the error, such as error code, error message, error type, and error details. CommerceDiagnosticEvents can be subscribed by other components or services that want to handle or display the error information in different ways. The developer's code can listen for CommerceDiagnosticEvents using an aura:handler tag in Aura Components or an @wire decorator in Lightning web components. The developer's code can also use the event information to perform custom logic or actions based on the error, such as logging, reporting, or notifying. CommerceBubbleEvents is not an event that the developer's code should listen for in checkout, as it is not related to troubleshooting or responding to actions. CommerceBubbleEvents is an event that is fired by Salesforce B2B Commerce when a user interacts with a bubble component in the storefront. A bubble component is a user interface element that displays information or options in a pop-up window when clicked or hovered over. CommerceBubbleEvents contains information about the user interaction, such as bubble name, bubble type, and bubble value. CommerceErrorEvents is not an event that the developer's code should listen for in checkout either, as it is not related to troubleshooting or responding to actions.

CommerceErrorEvents is an event that is fired by Salesforce B2B Commerce when a validation error occurs in the storefront. CommerceErrorEvents contains information about the validation error, such as field name, field label, and error message. CommerceActionEvents is not an event that the developer's code should listen for in checkout either, as it is not related to troubleshooting or responding to actions. CommerceActionEvents is an event that is fired by Salesforce B2B Commerce when a user performs an action in the storefront.

CommerceActionEvents contains information about the user action, such as action name, action type, and action parameters. Salesforce References: B2B Commerce Developer Guide: Handle Errors, B2B Commerce Developer Guide: Handle User Interactions with Bubble Components, B2B Commerce Developer Guide:

Handle Validation Errors, B2B Commerce Developer Guide: Handle User Actions

NEW QUESTION: 136

Which code statement should a developer use to import the ID of the current Lightning Experience

- A. import id from '@salesforce/network/Id'
- B. import id from '@salesforce/experience/Id'
- C. import id from '@salesforce/site/Id'
- D. import id from '@salesforce/community/Id'

Answer: D (LEAVE A REPLY)

To import the ID of the current Lightning Experience community, a developer should use the following code statement:

```
import id from '@salesforce/community/Id';
```

The @salesforce/community module allows the developer to access information about the current community, such as its ID, name, URL, and base path. The other modules do not exist or are not related to the community ID. The @salesforce/network module is used to access information about the current network, such as its ID and name. The @salesforce/experience module is used to access information about the current user experience, such as whether it is standard or custom. The @salesforce/site module is used to access information about the current site, such as its name and prefix. Salesforce References: [Lightning Web Components Developer Guide: Import Community Information], [Lightning Web Components Developer Guide: Import Salesforce Modules]

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: <https://www.actual4test.com/B2B->

NEW QUESTION: 137

What are three advantages of using ccLog over the Salesforce standard System.debug class? (3 answers)

- A. There is no need to use string concatenation to easily tag log statements with a subject.
- B. ccLog can debug syntax errors found in the JavaScript.
- C. There is no need to create a User Trace Flag.
- D. Append #ccLog=<Logging Token Name> to the end of the storefront URL in order to get logs in the inspector console.
- E. There is no need to manually set a cookie to debug with the Site Guest User.

Answer: ([SHOW ANSWER](#))

Explanation

Three advantages of using ccLog over the Salesforce standard System.debug class are:

- * There is no need to use string concatenation to easily tag log statements with a subject. ccLog allows passing a subject parameter to the log method, which will prepend the subject to the log message. For example, ccLog.log('This is a message', 'Subject') will log [Subject] This is a message.
- * There is no need to create a User Trace Flag. ccLog can be enabled by setting the value of CO.logToken to true in CCAdmin, which will activate logging for all users who access the storefront.
- * There is no need to manually set a cookie to debug with the Site Guest User. ccLog can be enabled for the Site Guest User by appending #ccLog=<Logging Token Name> to the end of the storefront URL in order to get logs in the inspector console. For example, https://my-storefront.com/#ccLog=debug will enable logging for the Site Guest User with the debug level. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Logging

NEW QUESTION: 138

What is likely to happen if a developer leaves debug mode turned on in an environment?

- A. The performance of the org will become slower each day
- B. The user will begin getting JavaScript limit exceptions
- C. The org will turn off debug mode after 72 hours
- D. A banner will be displayed to the user indicating that the org is in debug mode

Answer: ([SHOW ANSWER](#))

If a developer leaves debug mode turned on in an environment, the user will begin getting JavaScript limit exceptions. Debug mode is a setting that enables more detailed logging and error reporting for Lightning web components. However, it also increases the size and complexity of the JavaScript code that is delivered to the browser, which can cause performance issues and JavaScript limit exceptions. The JavaScript limit exceptions are errors that occur when the browser reaches its maximum capacity for executing JavaScript code, such as memory heap size

or script execution time. The performance of the org will not become slower each day, as debug mode only affects the client-side performance, not the server-side performance. The org will not turn off debug mode after 72 hours, as debug mode is a persistent setting that can only be changed manually by an administrator. A banner will not be displayed to the user indicating that the org is in debug mode, as debug mode is a transparent setting that does not affect the user interface. Salesforce Reference: Lightning Web Components Developer Guide: Debug Your Code, Lightning Web Components Developer Guide: JavaScript Limit Exceptions

NEW QUESTION: 139

A developer exports data from an org on a standard entity which has a custom attribute. When they launch Data Loader, select the entity, click the Select All Fields button and click Finish, the custom field they added called MyCustomField__c has no values and no column header in the CSV file. What is the root cause?

- A.** The user needs to install a specific Zulu JDK that is recommended by Salesforce.
- B.** A mapping file was not used when the data was loaded in
- C.** The user does not have access to the field
- D.** The user has rights to the field but there are no values in it

Answer: (SHOW ANSWER)

The root cause of why the custom field MyCustomField__c has no values and no column header in the CSV file is that the user does not have access to the field. A user's access to a field is determined by their profile and permission sets, which define their field-level security settings. Field-level security settings control whether a user can see, edit, or delete the value for a particular field on an object. If a user does not have access to a field, they will not be able to view or modify its value in any interface, including Data Loader. Data Loader is a tool that allows users to import or export data between Salesforce and CSV files. When using Data Loader to export data from an org, Data Loader will only include fields that are accessible to the user based on their field-level security settings. If a user does not have access to a field, Data Loader will not include that field in the CSV file, neither as a column header nor as a value. The user needs to install a specific Zulu JDK that is recommended by Salesforce is not the root cause of why MyCustomField__c has no values and no column header in the CSV file, as it is not related to field access or Data Loader functionality. A mapping file was not used when the data was loaded in is not the root cause either, as it is not related to field access or Data Loader functionality. A mapping file is an optional file that maps fields between Salesforce objects and CSV files when using Data Loader to import or export data. The user has rights to the field but there are no values in it is not the root cause either, as it contradicts the fact that MyCustomField__c has no column header in the CSV file. If the user had rights to the field but there were no values in it, Data Loader would still include MyCustomField__c as a column header in the CSV file, but leave its values blank. Salesforce References:

[Data Loader Guide: Export Data from Salesforce], [Data Loader Guide: Field Mapping],

[Salesforce Help:

Set Field-Level Security]

NEW QUESTION: 140

A developer is creating a component to implement a custom Terms and Conditions checkbox at checkout in the Aura Commerce template.

Which method should the developer implement on the Lightning web component to ensure the user accepts the terms and conditions?

- A. ComponentValidity
- B. Validate
- C. SaveCheckout
- D. CheckValidity

Answer: B (LEAVE A REPLY)

To implement a custom Terms and Conditions checkbox at checkout in the Aura Commerce template, a developer should add a Desired Delivery Date input field during the checkout flow. The Desired Delivery Date input field allows the customer to enter a date when they want their order to be delivered. The developer can use the `@api` decorator to expose this field as a public property of the Lightning web component and bind it to the `ccCheckoutOrder` object. The developer can also use the `@wire` decorator to get the current cart object and use its properties, such as shipping address and shipping method, to calculate and display an estimated delivery date based on the desired delivery date. The developer can also add validation logic to ensure that the desired delivery date is valid and acceptable. Adding the Expected Delivery Date field to the order confirmation email is not a good solution, as it does not allow the customer to choose or see their delivery date before placing their order. Displaying the Expected Delivery Date on the order page with a Lightning web component is not a good solution either, as it does not allow the customer to enter or change their delivery date after placing their order. Configuring an email alert to the customer when the Expected Delivery Date changes is not a good solution either, as it does not provide a consistent or reliable way of informing the customer about their delivery date. Salesforce References: B2B Commerce Developer Guide: Checkout Flow, B2B Commerce Developer Guide: Checkout Order Object, Lightning Web Components Developer Guide: Communicate with Properties

NEW QUESTION: 141

Which two technologies can subscribe to the `CommerceDiagnosticEvents` event?

- A. Aura Components
- B. Processes
- C. Streaming API
- D. Lightning web components

Answer: A,D (LEAVE A REPLY)

Explanation

Two technologies that can subscribe to the `CommerceDiagnosticEvents` event are Aura Components and Lightning web components. `CommerceDiagnosticEvents` is an event that is fired by Salesforce B2B Commerce when an error occurs in the storefront.

CommerceDiagnosticEvents contains information about the error, such as error code, error message, error type, and error details. CommerceDiagnosticEvents can be subscribed by other components or services that want to handle or display the error information in different ways. Aura Components are a type of component that can be used to create custom user interfaces for Salesforce apps. Aura Components can subscribe to CommerceDiagnosticEvents using an aura:handler tag in their markup file. The aura:handler tag specifies an event name, an action attribute that defines a controller function to handle the event, and other optional attributes. Lightning web components are another type of component that can be used to create custom user interfaces for Salesforce apps. Lightning web components can subscribe to CommerceDiagnosticEvents using an @wire decorator in their JavaScript file. The @wire decorator specifies an event name, a function name that defines a handler for the event, and other optional parameters. Processes are not a technology that can subscribe to CommerceDiagnosticEvents, as they are not related to user interface development or event handling. Processes are automated workflows that execute actions based on certain criteria or conditions in Salesforce. Streaming API is not a technology that can subscribe to CommerceDiagnosticEvents either, as it is not related to user interface development or event handling. Streaming API is an API that allows applications to receive notifications of data changes in Salesforce in near real-time. Salesforce References: [B2B Commerce Developer Guide: Handle Errors], [Aura Components Developer Guide: Handle Component Events], [Lightning Web Components Developer Guide: Communicate with Events], [Salesforce Help: Process Automation], [Salesforce Developer Guide: Streaming API]

NEW QUESTION: 142

How does a project implement the process to persist payment information data in the Checkout flow for Salesforce B2B Commerce version 4.2 and beyond?

- A.** Trigger a remote action when the process payment button is selected to capture the payment.
- B.** Trigger a remote action to store the payment information in the URL query parameters.
- C.** Trigger the processPayment event and pass in the payment information object as an argument.
- D.** Trigger the externalProcessedPayment and pass in the payment information object as an argument.

Answer: ([SHOW ANSWER](#))

To persist payment information data in the Checkout flow for Salesforce B2B Commerce version 4.2 and beyond, the project needs to trigger the processPayment event and pass in the payment information object as an argument. This event will invoke the processPayment method of the ccServicePayment class, which will validate and process the payment information and return a payment result object. The payment result object will contain the status and details of the payment transaction.

NEW QUESTION: 143

How can the display of CC Menu Items be customized for different users?

- A. cc_hk_Category extension to pre-process which category items are cached as menu items
- B. cc_hk_Menu extension to post-process any cached menu items
- C. cc_hk_Menu extension to pre-process which menu items are cached
- D. cc_hk_Category extension to post-process any cached menu items

Answer: B (LEAVE A REPLY)

The display of CC Menu Items can be customized for different users by using the cc_hk_Menu extension to post-process any cached menu items. This extension allows modifying the menu items based on the user context, such as the user role, account, or cart. For example, the extension can hide or show certain menu items based on the user's permissions or preferences.

NEW QUESTION: 144

A developer has just deployed a new Lightning web component called myNewLwcComp to an authorized org. The developer tries to find the component in the Lightning Page Builder, but it does not come up in searches. Which two steps should the developer take next?

- A. Ensure that the metadata isExposed property is set properly in source code
- B. Redeploy the component
- C. Close the browser and reopen the page
- D. Ensure it has a target of lightning__FlowScreen

Answer: A,D (LEAVE A REPLY)

Explanation

To make a Lightning web component available in the Lightning Page Builder or Experience Builder, the developer needs to do two things: set the isExposed property to true in the component's metadata file, and define at least one target that specifies where the component can be used, such as a Lightning page type or a flow screen. Redeploying the component or closing and reopening the browser will not make the component appear in the searches if the metadata file is not configured properly. References:

- * XML Configuration File Elements
- * Supported Salesforce Targets and Tools
- * #8: Use Lightning Web Components in Salesforce Targets

NEW QUESTION: 145

How are version related upgrades passed on to subscriber API extensions/overrides?

- A. APIs callback with specific versions specified; the user must know which version number to use.
- B. Copy and paste of specific code is "built-in"
- C. Extensions and overridden APIs don't support-related upgrades.
- D. The "delegate" allows inherited method calls access to the most recently specified service version

Answer: D (LEAVE A REPLY)

Version related upgrades are passed on to subscriber API extensions/overrides by using the "delegate" keyword, which allows inherited method calls access to the most recently specified service version. For example, `delegate.getCart()` will invoke the `getCart()` method of the latest service version that is available for the current storefront. This way, extensions and overrides can leverage the new features and enhancements of the upgraded service versions without modifying their code.

NEW QUESTION: 146

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?

- 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 (LEAVE A REPLY)

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 References: B2B Commerce Developer Guide: Debug Lightning Web Components, Lightning Web Components Developer Guide: Debug Your Code, Salesforce Help: Debug Logs

NEW QUESTION: 147

While working on a commerce rollout, a developer needs to update the checkout process so that buyers can purchase with one of the below payment types.

- * Credit Card
- * Purchase Order
- * Contract Now & Pay Later

Additionally, the developer needs to show only Purchase Order and Contract Now & Pay Later if a custom checkbox field on the account is checked.

How should the developer meet these requirements?

A. Create a custom Lightning web component that can be used with the standard payment component. Use a publish-<g, subscribe (pub-sub) model to listen to events from the standard component to determine which additional payment options should be shown.

B. Create a custom Lightning web component for the checkout flow that has all the options available.

Within that component, pull data from the account to determine which options to show.

C. Modify the standard payment component settings in the checkout screen flow and add the new payment method. Use the component visibility feature in screen flows to fulfill the account-based payment option criteria.

D. Add a new payment gateway through the reference implementation steps so the payment shows up on the checkout payment screen. Configure the different payment options required.

Answer: B (LEAVE A REPLY)

To update the checkout process so that buyers can purchase with one of the below payment types:

- * Credit Card
- * Purchase Order

* Contract Now & Pay Later Additionally, show only Purchase Order and Contract Now & Pay Later if a custom checkbox field on the account is checked, a developer should create a custom Lightning web component for the checkout flow that has all the options available. Within that component, pulldata from the account to determine which options to show. Creating a custom Lightning web component for the checkout flow allows the developer to define custom logic and user interface for processing payments using different payment types. The developer can use Apex methods or third-party APIs to integrate with payment service providers or payment gateways and handle payment authorization, capture, void, and refund. The developer can also

use `@wire` or `@api` decorators to get data from the account object and use its properties, such as the custom checkbox field, to determine which payment options to show or hide based on business logic. Creating a custom Lightning web component that can be used with the standard payment component is not a valid way to meet this requirement, as it does not allow the developer to replace or modify the standard payment component's logic or user interface. Modifying the standard payment component settings in the checkout screen flow and adding the new payment method is not a valid way either, as it does not allow the developer to add custom payment types or conditional logic based on account data. Adding a new payment gateway through the reference implementation steps so the payment shows up on the checkout payment screen is not a valid way either, as it does not allow the developer to add multiple payment options or conditional logic based on account data. Salesforce References: B2B Commerce Developer Guide: Payment Integration, B2B Commerce Developer Guide: Payment Component, B2B Commerce Developer Guide: Checkout Subflow

NEW QUESTION: 148

What is the purpose of `connectedCallback` in a Lightning web component?

- A. It performs actions when a network request is made.
- B. It perform actions when a component makes a call to a Connect APL.
- C. It performs actions when a component is removed from the DOM.
- D. It performs actions when a component is added to the DOM.

Answer: ([SHOW ANSWER](#))

Explanation

The `connectedCallback` is a lifecycle hook that is invoked when a Lightning web component is inserted into the DOM. It is used to perform any initialization tasks that depend on the component being rendered, such as fetching data, setting up event listeners, or creating timers. The `connectedCallback` can fire more than once, for example, when a component is moved or rerendered. The `connectedCallback` should not be used to change the state of a component, such as loading values or setting properties. Use getters and setters instead. The other options are incorrect because they do not describe the purpose of the `connectedCallback`. References: Lifecycle Hooks | Lightning Web Components Developer Guide, `connectedCallback()` in Lightning Web Component - Salesforce Diaries, Chapter 23: `connectedCallback()` in Lightning Web Component

NEW QUESTION: 149

What is the purpose of `connectedCallback` in a Lightning web component?

- A. It performs actions when a network request is made.
- B. It perform actions when a component makes a call to a Connect APL.
- C. It performs actions when a component is removed from the DOM.
- D. It performs actions when a component is added to the DOM.

Answer: D ([LEAVE A REPLY](#))

The `connectedCallback` is a lifecycle hook that is invoked when a Lightning web component is inserted into the DOM. It is used to perform any initialization tasks that depend on the component being rendered, such as fetching data, setting up event listeners, or creating timers. The `connectedCallback` can fire more than once, for example, when a component is moved or rerendered. The `connectedCallback` should not be used to change the state of a component, such as loading values or setting properties. Use getters and setters instead. The other options are incorrect because they do not describe the purpose of the `connectedCallback`. References: Lifecycle Hooks | Lightning Web Components Developer Guide, `connectedCallback()` in Lightning Web Component - Salesforce Diaries, Chapter 23: `connectedCallback()` in Lightning Web Component

NEW QUESTION: 150

Which format is the custom Salesforce relationship with the API name, "My_Relationship_Name__r.My_Name__c" queried and transformed into by default in Salesforce B2B Commerce?

- A. myrelationshipname.myname: value
- B. myRelationshipName.myName: value
- C. myRelationshipNameR=>(myName: value)
- D. My_Relationship_Name__r.My_Name__c: value

Answer: (SHOW ANSWER)

The format that the custom Salesforce relationship with the API name, My_Relationship_Name__r.My_Name__c is queried and transformed into by default in Salesforce B2B Commerce is myRelationshipName.myName: value. This is done to follow the JavaScript naming convention and to avoid conflicts with the standard Salesforce fields and relationships.

The transformation rules are as follows:

Remove any namespace prefix from the field name, such as ccrz__ or cloudcraze__.

Remove any underscores from the field name and capitalize the first letter of each word after an underscore, such as MyRelationshipName and MyName.

Lowercase the first letter of the field name, such as myRelationshipName and myName.

Use a dot (.) to separate the relationship name and the field name, such as myRelationshipName.myName.

Use a colon (:) to separate the field name and the field value, such as

myRelationshipName.myName: value. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 151

Which out of the box Salesforce B2B Commerce page can give instructions to web crawlers from accessing specific Salesforce B2B Commerce pages?

- A. CCCat?SiteMap
- B. cc_RobotsTxT
- C. CCSiteIndex

D. CCPage

Answer: B (LEAVE A REPLY)

Explanation

The out of the box Salesforce B2B Commerce page that can give instructions to web crawlers from accessing specific Salesforce B2B Commerce pages is cc_RobotsTxt. This is a Visualforce page that generates a robots.txt file, which is a text file that tells web crawlers which pages or files they can or can't request from a site. The page uses the configuration settings CO.RobotsTxtAllow and CO.RobotsTxtDisallow to specify which paths are allowed or disallowed for web crawlers. For example, User-agent: * Disallow: /CCCart will instruct web crawlers to not access the CCCart page. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Robots.txt File

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 152

While in the process of gathering requirements from a customer about how they would like to set up their net new storefront checkout experience, a consultant learns that the customer needs the ability to add new shipping and billing addresses during checkout.

Which approach should a developer take to meet this requirement?

- A. Create a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component.
- B. Enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow.
- C. Enable Buyer Managed Contact Point Addresses within Commerce Administration.
- D. Create a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow.

Answer: B (LEAVE A REPLY)

Explanation

To enable the ability to add new shipping and billing addresses during checkout, a developer should enable Buyer Managed Contact Point Addresses within the Shipping Address standard component in the Checkout subflow. The Buyer Managed Contact Point Addresses is a feature that allows customers to add, edit, or delete their shipping and billing addresses during checkout. The developer can enable this feature by setting the

buyerManagedContactPointAddressesEnabled attribute to true in the Shipping Address standard component in the Checkout subflow. The Shipping Address standard component is a component that displays and collects the shipping address information for the cart or order. The Checkout subflow is a subflow that defines the steps and components for the checkout process in the storefront. Creating a new shipping address checkout subflow that utilizes the Buyer Managed Contact Point Addresses component is not a valid way to enable this feature, as there is no such component as Buyer Managed Contact Point Addresses. Enabling Buyer Managed Contact Point Addresses within Commerce Administration is not a valid way either, as this feature is not configurable in Commerce Administration. Creating a Lightning web component that enables this functionality and replaces the current shipping address screen within the Checkout subflow is not a valid way either, as this is unnecessary and complicated when there is already a standard component that supports this feature. Salesforce References: [B2B Commerce Developer Guide: Buyer Managed Contact Point Addresses], [B2B Commerce Developer Guide: Shipping Address Component], [B2B Commerce Developer Guide: Checkout Subflow]

NEW QUESTION: 153

Although Salesforce B2B Commerce and Salesforce recommend against using "without sharing classes" whenever possible, sometimes it is unavoidable. Which three items will open up a major security hole? (3 answers)

- A.** Executing dynamic SOQL inside a without sharing class with a bind variable from `PageReference.getParameters()`.
- B.** Executing dynamic SOQL inside a without sharing class with a bind variable from the `UserInfo` class.
- C.** Executing dynamic SOQL inside a without sharing class with a bind variable from `PageReference.getCookies()`.
- D.** Executing dynamic SOQL inside a without sharing class with a bind variable from `cc_RemoteActionContext` class.
- E.** Executing dynamic SOQL inside a without sharing class with a bind variable from `ccAPI.CURRENT_VERSION`.

Answer: A,C,D (LEAVE A REPLY)

Executing dynamic SOQL inside a without sharing class with a bind variable from `PageReference.getParameters()`, `PageReference.getCookies()`, or `cc_RemoteActionContext` class will open up a major security hole because these sources of input are not sanitized and can be manipulated by malicious users to inject SOQL queries that bypass the sharing rules and access data that they are not supposed to see. For example, a user can modify the URL parameters or cookies to include a SOQL query that returns sensitive data from the database. To prevent this, it is recommended to use static SOQL or escape the bind variables before executing dynamic SOQL.

NEW QUESTION: 154

Ursa Major is planning to implement Salesforce B2B Commerce, and a developer needs to configure taxes for their storefront. The company operates in multiple states, each with different tax rates and tax rules.

What are two ways the developer should configure taxes in B2B Commerce?

- A. Configure a tax engine using third-party software.
- B. Configure tax rates and rules for each state in Salesforce B2B Commerce.
- C. Use a different pricebook for each state.
- D. Use the Salesforce out-of-the-box tax calculator.

Answer: A,B (LEAVE A REPLY)

Two ways that a developer should configure taxes in B2B Commerce for Ursa Major are: configure a tax engine using third-party software and configure tax rates and rules for each state in Salesforce B2B Commerce. Configuring a tax engine using third-party software allows the developer to integrate with an external tax service provider that can calculate and apply accurate and up-to-date tax rates and rules for different states and jurisdictions. The developer can use the `sfdc_checkout.TaxCalculations` interface or the `RegisteredExternalService` object to connect to the third-party tax service provider's API or service.

Configuring tax rates and rules for each state in Salesforce B2B Commerce allows the developer to implement their own custom tax calculation logic using Apex code. The developer can use custom objects, such as `TaxRate__c` and `TaxRule__c`, to store and manage tax rates and rules for different states and products. The developer can also use the `sfdc_checkout.TaxCalculations` interface to define how to retrieve and apply tax rates and rules for a cart or an order. Using a different pricebook for each state is not a good way to configure taxes in B2B Commerce, as it can create complexity and inconsistency in pricing and discounting logic. Using the Salesforce out-of-the-box tax calculator is not a good way either, as it does not support complex or dynamic tax scenarios that may vary by state or jurisdiction. Salesforce References: [B2B Commerce Developer Guide: Tax Integration], [B2B Commerce Developer Guide: Tax Calculations Interface], [B2B Commerce Developer Guide: TaxRate Object], [B2B Commerce Developer Guide: TaxRule Object]

NEW QUESTION: 155

Which wire adapter should a developer use to retrieve metadata about a specific picklist?

- A. `getPicklistMetadataValues`
- B. `getPicklistMetadata`
- C. `getPicklistValues`
- D. `getPicklist`

Answer: C (LEAVE A REPLY)

To retrieve metadata about a specific picklist, a developer should use the `getPicklistValues` wire adapter. The `getPicklistValues` wire adapter imports data from the `@salesforce/ui-api` module and returns an object that contains information such as the picklist's label, value, default value, validity, and controlling field values. The `getPicklistMetadataValues` wire adapter does not exist.

The `getPicklistMetadata` wire adapter does not exist either. The `getPicklist` wire adapter does not exist either. Salesforce Reference: [Lightning Web Components Developer Guide: `getPicklistValues`], [Lightning Web Components Developer Guide: Import User Interface API]

NEW QUESTION: 156

In checkout, what event should the developer's code listen for in order to help troubleshoot and respond to actions?

- A. `CommerceBubbleEvents`
- B. `CommerceErrorEvents`
- C. `CommerceActionEvents`
- D. `CommerceDiagnosticEvents`

Answer: ([SHOW ANSWER](#))

To help troubleshoot and respond to actions in checkout, the developer's code should listen for `CommerceDiagnosticEvents`. `CommerceDiagnosticEvents` is an event that is fired by Salesforce B2B Commerce when an error occurs in the storefront. `CommerceDiagnosticEvents` contains information about the error, such as error code, error message, error type, and error details. `CommerceDiagnosticEvents` can be subscribed by other components or services that want to handle or display the error information in different ways. The developer's code can listen for `CommerceDiagnosticEvents` using an `aura:handler` tag in Aura Components or an `@wire` decorator in Lightning web components. The developer's code can also use the event information to perform custom logic or actions based on the error, such as logging, reporting, or notifying. `CommerceBubbleEvents` is not an event that the developer's code should listen for in checkout, as it is not related to troubleshooting or responding to actions. `CommerceBubbleEvents` is an event that is fired by Salesforce B2B Commerce when a user interacts with a bubble component in the storefront. A bubble component is a user interface element that displays information or options in a pop-up window when clicked or hovered over. `CommerceBubbleEvents` contains information about the user interaction, such as bubble name, bubble type, and bubble value. `CommerceErrorEvents` is not an event that the developer's code should listen for in checkout either, as it is not related to troubleshooting or responding to actions. `CommerceErrorEvents` is an event that is fired by Salesforce B2B Commerce when a validation error occurs in the storefront. `CommerceErrorEvents` contains information about the validation error, such as field name, field label, and error message. `CommerceActionEvents` is not an event that the developer's code should listen for in checkout either, as it is not related to troubleshooting or responding to actions. `CommerceActionEvents` is an event that is fired by Salesforce B2B Commerce when a user performs an action in the storefront. `CommerceActionEvents` contains information about the user action, such as action name, action type, and action parameters. Salesforce References: B2B Commerce Developer Guide: Handle Errors, B2B Commerce Developer Guide: Handle User Interactions with Bubble Components, B2B Commerce Developer Guide: Handle Validation Errors, B2B Commerce Developer Guide: Handle User Actions

NEW QUESTION: 157

Which two statements are true for Mass Order (2 answers)

- A. Mass Order pricing is done via a batch job.
- B. Mass order works with the default wishlists
- C. The variation product is leveraged for SKUs.
- D. Mass Order is mobile ready with the ccrz templates.

Answer: A,C (LEAVE A REPLY)

Mass Order pricing is done via a batch job, which means that the prices are not calculated in real time, but rather at a scheduled time. Mass order works with the variation product, which is a product that has multiple SKUs with different attributes, such as size or color. Mass order does not work with the default wishlists, which are used to store products that the customer wants to buy later. Mass order is not mobile ready with the ccrz templates, which are the default templates for the storefront pages. Salesforce References: B2B Commerce Developer Guide: Mass Order, B2B Commerce Developer Guide: Variation Products

NEW QUESTION: 158

What is a valid way of referencing the CC Cart Object whose API name is E_Cart__c in a SOQL query?

- A. _Cart__c
- B. c.E_Cart__c
- C. ccrz__E_Cart__c
- D. cloudcraze__E_Cart__c

Answer: (SHOW ANSWER)

A valid way of referencing the CC Cart Object whose API name is E_Cart__c in a SOQL query is to use ccrz__E_Cart__c. This is the transformed name of the object that is used by the Salesforce B2B Commerce framework. All custom objects and fields that are part of the cloudcraze managed package have the prefix ccrz__ in their API names. For example, SELECT Id, Name FROM ccrz__E_Cart__c will query the CC Cart Object records. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 159

Which wire adapter should a developer use to retrieve metadata about a specific object?

- A. getObjectMetadata
- B. getObjectInfo
- C. getObject
- D. getObjectDescribe

Answer: B (LEAVE A REPLY)

To retrieve metadata about a specific object, a developer should use the getObjectInfo wire adapter. The getObjectInfo wire adapter imports data from the @salesforce/schema module and returns an object that contains information such as the object's label, key prefix, fields, child relationships, record type infos, and theme. The getObjectMetadata wire adapter does not exist.

The getObject wire adapter does not retrieve metadata, but rather returns a record object based on the record ID. The getObjectDescribe wire adapter does not exist either. Salesforce Reference: [Lightning Web Components Developer Guide: getObjectInfo], [Lightning Web Components Developer Guide: Import Salesforce Schema]

NEW QUESTION: 160

Which three steps are necessary to have subscriberpage added to Salesforce B2B Commerce after creating a custom Visualforce page? (3 answers)

- A.** Create a new CC Subscriber Page record that points to your custom Visualforce page.
- B.** Create a new Visualforce page, and manually import the Salesforce B2BCommerce JavaScript libraries. Run in Anonymous Apex `ccrz.cc_util_Reflection.upsertPageUIKey([arg1],[arg2],[arg3]);`
- C.** Refresh the Page Keys Index in CC Admin.
- D.** Enable the Subscriber Page in CC Admin.

Answer: A,C,D (LEAVE A REPLY)

Three steps that are necessary to have a subscriber page added to Salesforce B2B Commerce after creating a custom Visualforce page are:

Create a new CC Subscriber Page record that points to your custom Visualforce page. This record will store information about the subscriber page, such as the name, description, URL, and Visualforce page. For example, you can create a new record named MySubscriberPage that points to your custom Visualforce page named MyPage.

Refresh the Page Keys Index in CC Admin. This action will update the page keys index, which is a cache that stores the mapping between the page keys and the subscriber pages. You need to do this whenever you create or modify a subscriber page record.

Enable the Subscriber Page in CC Admin. This action will allow you to select the subscriber page from the CC Page Settings configuration and assign it to a CC Page. For example, you can enable MySubscriberPage and assign it to the Home page. Salesforce Reference: B2B Commerce and D2C Commerce Developer Guide, Subscriber Pages

NEW QUESTION: 161

Which three pages should be enabled for the Guest user profile for a storefront to have anonymous checkout?

(3 answers)

- A.** OrderConfirmation
- B.** CCPaymentInfo
- C.** CheckoutNew
- D.** OrderView
- E.** Checkout

Answer: A,B,C (LEAVE A REPLY)

NEW QUESTION: 162

A configuration value, CO.NewOrder, is set to TRUE. What is one way of

preventing an existing payment page from being shown on the checkout payment page?

- A. Delete the Visualforce page from the code base.
- B. Remove the value matching the page name from the `pmt.whitelist` configuration setting, then rebuild and activate a new Configuration cache
- C. Remove the payment type associated with the payment page from `CO.pmts`, then rebuild and activate a new cache.
- D. Override the front end template and modify the way the embedded payment page gets loaded from the payment list configuration.

Answer: B (LEAVE A REPLY)

This approach effectively removes the payment page from the list of allowed pages, ensuring it is not displayed during the checkout process.

NEW QUESTION: 163

In which two ways can events fired from Lightning web components be handled?

- A. Programmatically adding event listeners
- B. Adding callbacks to components
- C. Listening for all possible events at the document root
- D. Attaching handlers to DOM elements

Answer: A,D (LEAVE A REPLY)

Two ways that events fired from Lightning web components can be handled are programmatically adding event listeners and attaching handlers to DOM elements. Programmatically adding event listeners is a way of handling events by using JavaScript code to register functions that are invoked when an event occurs. The developer can use methods such as `addEventListener` or `@wire` to add event listeners to components or services that fire events. Attaching handlers to DOM elements is a way of handling events by using HTML attributes to bind functions that are invoked when an event occurs. The developer can use attributes such as `onclick` or `onchange` to attach handlers to DOM elements that fire events. Adding callbacks to components is not a valid way of handling events fired from Lightning web components, as it is not related to event handling, but rather to asynchronous programming. Listening for all possible events at the document root is not a valid way either, as it is not efficient or recommended for event handling, as it can cause performance issues or conflicts with other event listeners. Salesforce References: [Lightning WebComponents Developer Guide: Handle Events], [Lightning Web Components Developer Guide: Communicate with Events]

NEW QUESTION: 164

What are three ways to test the value of Page Label on any Salesforce B2B Commerce Community Page? (3 answers)

- A. Enable the 'display page label names' in cc admin.
- B. Execute `CCRZ.pagevars.pageLabels['PAGE_LABEL_NAME']` in the JavaScript console.
- C. Execute `(('PAGE_LABEL_NAME'))` in the JavaScript console
- D. Execute `CCRZ.processPageLabelMap('PAGE_LABEL_NAME')` in the JavaScript console.

E. Access the source HTML for the page via the browser developer tools.

Answer: A,B,D ([LEAVE A REPLY](#))

NEW QUESTION: 165

Numerous flags ... have a direct impact on the result set provided by the Global API's. What Global API Data-Sizing convention flag prevents an API request from propagating to further requests when provided as a Boolean parameter with a value of true?

- A. ccrz.ccAPI.SZ_REL
- B. ccrz.ccAPI.SZ_ASSC
- C. ccrz.ccAPISizing.ASSC
- D. ccrz.ccAPISizing.REL

Answer: ([SHOW ANSWER](#))

Explanation

The Global API Data-Sizing convention flag that prevents an API request from propagating to further requests when provided as a Boolean parameter with a value of true is ccrz.ccAPI.SZ_ASSC. This flag indicates that only one API request should be executed with the specified sizing block, and any subsequent requests should use their own default sizing blocks. For example, ccrz.ccServiceCart.getCart(ccrz.ccAPI.SZ_L,true) will use the SZ_L sizing block for retrieving the cart data, but any other requests that are triggered by this method will use their own default sizing blocks. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Data Sizing Conventions

NEW QUESTION: 166

Which three pages should be enabled for the Guest user profile for a storefront to have anonymous checkout?

(3 answers)

- A. CCPaymentInfo
- B. CheckoutNew
- C. OrderView
- D. Checkout
- E. OrderConfirmation

Answer: ([SHOW ANSWER](#))

Three pages that should be enabled for the Guest user profile for a storefront to have anonymous checkout are:

* CCPaymentInfo: This page allows the guest user to enter their payment information, such as credit card number, expiration date, and security code. The page also displays the order summary and total amount.

* CheckoutNew: This page allows the guest user to enter their shipping and billing information, such as name, address, phone number, and email. The page also displays the cart items and shipping options.

* OrderConfirmation: This page displays the confirmation message and order number after the guest user places their order. The page also provides a link to view the order details or print the invoice. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Anonymous Checkout

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 167

How are version related upgrades passed on to subscriber API extensions/overrides?

- A. APIs callback with specific versions specified; the user must know which version number to use.
- B. Copy and paste of specific code is "built-in"
- C. Extensions and overridden APIs don't support-related upgrades.
- D. The "delegate" allows inherited method calls access to the most recently specified service version

Answer: (SHOW ANSWER)

Explanation

Version related upgrades are passed on to subscriber API extensions/overrides by using the "delegate" keyword, which allows inherited method calls access to the most recently specified service version. For example, `delegate.getCart()` will invoke the `getCart()` method of the latest service version that is available for the current storefront. This way, extensions and overrides can leverage the new features and enhancements of the upgraded service versions without modifying their code.

NEW QUESTION: 168

What two things happen with the Cart during tax implementation?

- A. New entries are written to the Cart
- B. Previous entries are copied to another object
- C. Previous entries are deleted from the Cart
- D. New entries are written to the Order Summary

Answer: A,C (LEAVE A REPLY)

Two things that happen with the cart during tax implementation are that new entries are written to the cart and previous entries are deleted from the cart. A tax implementation is an integration that

calculates and applies tax rates and amounts to a cart or an order based on various factors, such as product type, price, quantity, location, and tax rules. A tax implementation can use either an external tax service provider or custom Apex code to perform the tax calculation. When a tax implementation is invoked for a cart, it writes new entries to the cart with a type of Charge and a charge type of Tax. These entries represent the tax amounts and adjustments that are applied to the cart. Before writing new entries to the cart, the tax implementation deletes any existing entries with a type of Charge and a charge type of Tax from the cart. This ensures that the tax calculation is accurate and up-to-date based on the current state of the cart and avoids any conflicts or inconsistencies with previous tax entries. Previous entries are not copied to another object or modified with the new tax calculation, as these are not valid actions for handling existing tax entries. Salesforce References: B2B Commerce Developer Guide: Tax Integration, B2B Commerce Developer Guide: Tax Calculation Flow

NEW QUESTION: 169

In which three different ways can a theme be enabled in Salesforce B2B Commerce? (3 answers)

- A. A Storefront setting
- B. An Account Group field value
- C. A per user setting
- D. Account
- E. Dynamically through a hook

Answer: A,B,E (LEAVE A REPLY)

A theme can be enabled in Salesforce B2B Commerce in three different ways:

A Storefront setting: The theme can be specified in the Storefront Configuration settings in CCAdmin. This will apply the theme to all users who access the storefront.

An Account Group field value: The theme can be specified in the Theme field of an Account Group record in Salesforce. This will apply the theme to all users who belong to that account group.

Dynamically through a hook: The theme can be specified dynamically by extending the `cc_hk_theme` hook. This hook allows changing the theme based on various factors, such as the user, cart, product, or storefront. For example, the hook can apply a different theme for a specific product category or for a specific user segment.

NEW QUESTION: 170

Salesforce B2B Commerce natively provides a robots.txt file, however, a customer can also create its own version. Which three scenarios are valid reasons for customer to create their own robots.txt file? (3 answers)

- A. The customer wants to reference multiple storefront sitemap indexes in a single robots.txt file
- B. The customer wants to have multiple robot.txt files in a single Salesforce Community.
- C. The customer wants to reference a custom sitemap index.
- D. The customer's store is not located at the root of their domain.

E. robot.txt only works if there is one storefront in the org

Answer: A,C,D (LEAVE A REPLY)

NEW QUESTION: 171

Which two methods from the platformResourceLoader module are relevant for including third party JavaScript and CSS in a Lightning web component?

- A. loadClientScript
- B. loadScript
- C. loadCss
- D. loadStyle

Answer: B,D (LEAVE A REPLY)

Two methods from the platformResourceLoader module that are relevant for including third party JavaScript and CSS in a Lightning web component are loadScript and loadStyle. The platformResourceLoader module is a module that provides methods for loading JavaScript or CSS files from static resources or external URLs into a Lightning web component. The loadScript method is used to load a JavaScript file and execute it in the component. The loadStyle method is used to load a CSS file and apply it to the component. The loadClientScript method does not exist or is not relevant for including third party JavaScript in a Lightning web component. The loadCss method does not exist or is not relevant for including third party CSS in a Lightning web component. Salesforce Reference: Lightning Web Components Developer Guide: Load Scripts and Style Sheets, [Lightning Web Components Developer Guide: platformResourceLoader Module]

NEW QUESTION: 172

When a developer configures a tax integration for a store, what happens to the previously calculated tax entries during the checkout flow?

- A. Ignored during recalculation
- B. Saved prior to recalculation
- C. Deleted from the Cart
- D. Modified with the new tax calculation

Answer: C (LEAVE A REPLY)

When a developer configures a tax integration for a store, the previously calculated tax entries during the checkout flow are deleted from the Cart. A tax integration is an integration that calculates and applies tax rates and amounts to a Cart or an Order based on various factors, such as product type, price, quantity, location, and tax rules. A tax integration can use either an external tax service provider or custom Apex code to perform the tax calculation. When a developer configures a tax integration for a store, any existing tax entries in the Cart are deleted before calling the tax integration service or method. This ensures that the tax calculation is accurate and up-to-date based on the current state of the Cart and avoids any conflicts or inconsistencies with previous tax entries. The previously calculated tax entries are not ignored during recalculation, saved prior to recalculation, or modified with the new tax calculation, as

these are not valid actions for handling existing tax entries. Salesforce Reference: B2B Commerce Developer Guide: Tax Integration, B2B Commerce Developer Guide: Tax Calculation Flow

NEW QUESTION: 173

A user wants to leverage a three column layout on a page. The user also wants to move the mini-cart widget from the right to the center column. How can this requirement be fulfilled?

- A. Gross Layout Override
- B. Subscriber Template
- C. Page Include
- D. HandleBar Template Override

Answer: A ([LEAVE A REPLY](#))

To leverage a three column layout on a page and move the mini-cart widget from the right to the center column, the requirement can be fulfilled by creating a Gross Layout Override. This is a custom Visualforce page that overrides the default layout of a page and allows changing its structure and content. The user can create a Gross Layout Override that uses a three column layout and places the mini-cart widget in the center column. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Gross Layout Overrides

NEW QUESTION: 174

Although Salesforce B2B Commerce and Salesforce recommend against using "without sharing classes" whenever possible, sometimes it is unavoidable. Which three items will open up a major security hole? (3 answers)

- A. Executing dynamic SOQL inside a without sharing class with a bind variable from `PageReference.getParameters()`.
- B. Executing dynamic SOQL inside a without sharing class with a bind variable from `ccAPI.CURRENT_VERSION`.
- C. Executing dynamic SOQL inside a without sharing class with a bind variable from `PageReference.getCookies()`.
- D. Executing dynamic SOQL inside a without sharing class with a bind variable from `cc_RemoteActionContentex` class.
- E. Executing dynamic SOQL inside a without sharing class with a bind variable from the `UserInfo` class.

Answer: ([SHOW ANSWER](#))

NEW QUESTION: 175

Universal Containers (UC) is ready to build a tax provider class using the interfaces available in the Buyer Experience SDK. When creating a tax provider, what are three things that a developer should consider first?

- A. Steps to complete in the Tax Service
- B. How to handle results

- C. WhethertouseJSONorXML
- D. What to implement
- E. What events to fire in the Lightning web component

Answer: (SHOW ANSWER)

When creating a tax provider, three things that a developer should consider first are: steps to complete in the tax service, how to handle results, and what to implement. Steps to complete in the tax service are the actions that the developer needs to perform to connect to and use the third-party tax service provider's API or service. These steps may include authentication, authorization, request formatting, response parsing, error handling, and logging. How to handle results are the actions that the developer needs to perform to process and apply the tax calculation results from the tax service to the cart or order. These actions may include creating or updating cart items with type of Charge and charge type of Tax, applying tax adjustments or exemptions, and displaying tax amounts and details on the storefront. What to implement are the methods that the developer needs to define in their custom Apex class that implements the `sfdc_checkout.TaxCalculations` interface. The interface provides methods for customizing the tax calculation logic for a cart or an order, such as `getTaxRatesAndRules`, `applyTaxAmountsAndAdjustments`, and `handleTaxErrorsAndExceptions`. Whether to use JSON or XML is not something that the developer should consider first, as it is not a critical or relevant factor for creating a tax provider. It may depend on the format that the tax service provider supports or prefers, but it does not affect the overall functionality or performance of the tax integration. What events to fire in the Lightning web component is not something that the developer should consider first either, as it is not related to creating a tax provider. It may be an optional feature that the developer can add to enhance the user interface or user experience of their storefront, but it does not affect the core logic or functionality of the tax integration. Salesforce Reference: B2B Commerce Developer Guide: Tax Integration, [B2B Commerce Developer Guide: Tax Calculations Interface]

NEW QUESTION: 176

Which two technologies can subscribe to the `CommerceDiagnosticEvents` event?

- A. Aura Components
- B. Processes
- C. Streaming API
- D. Lightning web components

Answer: A,D (LEAVE A REPLY)

Two technologies that can subscribe to the `CommerceDiagnosticEvents` event are Aura Components and Lightning web components. `CommerceDiagnosticEvents` is an event that is fired by Salesforce B2B Commerce when an error occurs in the storefront.

`CommerceDiagnosticEvents` contains information about the error, such as error code, error message, error type, and error details. `CommerceDiagnosticEvents` can be subscribed by other components or services that want to handle or display the error information in different ways.

Aura Components are a type of component that can be used to create custom user interfaces for

Salesforce apps. Aura Components can subscribe to CommerceDiagnosticEvents using an aura:handler tag in their markup file. The aura:handler tag specifies an event name, an action attribute that defines a controller function to handle the event, and other optional attributes. Lightning web components are another type of component that can be used to create custom user interfaces for Salesforce apps. Lightning web components can subscribe to CommerceDiagnosticEvents using an @wire decorator in their JavaScript file. The @wire decorator specifies an event name, a function name that defines a handler for the event, and other optional parameters. Processes are not a technology that can subscribe to CommerceDiagnosticEvents, as they are not related to user interface development or event handling. Processes are automated workflows that execute actions based on certain criteria or conditions in Salesforce. Streaming API is not a technology that can subscribe to CommerceDiagnosticEvents either, as it is not related to user interface development or event handling. Streaming API is an API that allows applications to receive notifications of data changes in Salesforce in near real-time. Salesforce References: [B2B Commerce Developer Guide: Handle Errors], [Aura Components Developer Guide: Handle Component Events], [Lightning Web Components Developer Guide: Communicate with Events], [Salesforce Help: Process Automation], [Salesforce Developer Guide: Streaming API]

NEW QUESTION: 177

Which event is invoked by any CCRZ Salesforce B2B CommerceView after the view is rendered?

- A. view:*.onload
- B. view:*.refresh
- C. view:*.rendered
- D. view:*.load

Answer: (SHOW ANSWER)

NEW QUESTION: 178

Which format is the custom Salesforce relationship with the API name, "My_Relationship_Name__r.My_Name__c" queried and transformed into by default in Salesforce B2B Commerce?

- A. myrelationshipname.myname: value
- B. myRelationshipName.myName: value
- C. myRelationshipNameR=>(myName: value)
- D. My_Relationship_Name__r.My_Name__c: value

Answer: C (LEAVE A REPLY)

Explanation

The format that the custom Salesforce relationship with the API name, My_Relationship_Name__r.My_Name__c is queried and transformed into by default in Salesforce B2B Commerce is myRelationshipName.myName: value. This is done to follow the JavaScript

naming convention and to avoid conflicts with the standard Salesforce fields and relationships.

The transformation rules are as follows:

- * Remove any namespace prefix from the field name, such as ccrz__ or cloudcraze__.
- * Remove any underscores from the field name and capitalize the first letter of each word after an underscore, such as MyRelationshipName and MyName.
- * Lowercase the first letter of the field name, such as myRelationshipName and myName.
- * Use a dot (.) to separate the relationship name and the field name, such as myRelationshipName.myName.
- * Use a colon (:) to separate the field name and the field value, such as myRelationshipName.myName:

value. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Query Transformation

NEW QUESTION: 179

During checkout flow customizations, a developer receives an error on shipping cost calculation integrations with the error code: `INSUFFICIENT_ACCESS_OR_READONLY`.

What is causing this error?

- A.** The storefront user does not have access to the Cart Delivery Method object.
- B.** An error has occurred during the cart shipping charge integration.
- C.** The storefront user does not have access to custom fields on the Order Delivery Method object.
- D.** The cart is no longer in a valid Checkout State.

Answer: D (LEAVE A REPLY)

The error code `INSUFFICIENT_ACCESS_OR_READONLY` is caused by the cart being no longer in a valid Checkout State during checkout flow customizations. A cart is an object that represents a collection of products and charges that a customer intends to purchase in the storefront. A cart has a Checkout State field that indicates the current state of the checkout process for the cart. The Checkout State can have values such as Draft, InProgress, Completed, or Cancelled. A cart can only be modified or updated when it is in Draft or InProgress state. A cart cannot be modified or updated when it is in Completed or Cancelled state. If an attempt is made to modify or update a cart that is in Completed or Cancelled state, an error with the code `INSUFFICIENT_ACCESS_OR_READONLY` will be thrown. This error means that the user does not have permission to edit or delete a record because it is read-only or locked. The storefront user does not have access to the Cart Delivery Method object is not a cause of this error code, as it is not related to the cart checkout state or data modification. The Cart Delivery Method object is an object that stores information about the delivery method selected for a cart in the storefront. An error has occurred during the cart shipping charge integration is not a cause of this error code either, as it is not related to the cart checkout state or data modification. The cart shipping charge integration is an integration that calculates and applies shipping charges to a cart based on various factors such as delivery method, location, weight, volume, etc. The storefront user does not have access to custom fields on the Order Delivery Method object is not a cause of this error

code either, as it is not related to the cart checkout state or data modification. The Order Delivery Method object is an object that stores information about the delivery method selected for an order summary in the storefront. Salesforce References: B2B Commerce Developer Guide: Cart Object, [B2B Commerce Developer Guide: Cart Delivery Method Object], [B2B Commerce Developer Guide: Order Delivery Method Object], [Salesforce Help: Common Error Messages]

NEW QUESTION: 180

A Northern Trail Outfitters (NTO) developer made a tile component. To expose a click event and react to user input using the markup below, what should replace <CLICK_EVENT>?

```
<template>
  <div class="container">
    <a onclick=CLICK_EVENT>
      <div class="title">{product.fields.Name.value}</div>
      <img class="product-img" src={product.fields.Picture_URL__c.value}></img>
    </a>
  </div>
</template>
```

- A. tileClick()
- B. {event:tileClick}
- C. javascript:avoid(0);tileClick();
- D. {tileClick}

Answer: A (LEAVE A REPLY)

Explanation

To expose a click event and react to user input in a Lightning web component, the developer should use a method name as the value of the onclick attribute in the template. The method name should be followed by parentheses, as in tileClick(). This syntax indicates that the method is invoked when the element is clicked.

The method should be defined in the JavaScript file of the component, and it can access the event object as a parameter. The other options are either invalid or incorrect. For example, {event:tileClick} is not a valid syntax for an onclick attribute, javascript:avoid(0);tileClick(); is unnecessary and outdated, and {tileClick} is a property binding expression, not a method invocation. References: B2B Commerce and D2C Commerce Developer Guide, Lightning Web Components Developer Guide

NEW QUESTION: 181

Why is the use of a standard Visualforce control such as apex:form discouraged in Salesforce B2B Commerce page includes and subscriber pages?

- A. The CCRZ Javascript object is not accessible within an apex:form control.
- B. Visualforce "scopes" controls that are present on a page and scope of the control will be set to "ccrz"

- C. Javascript events are not supported within an apex:form control
- D. Apex:form render DOM components slowly

Answer: B ([LEAVE A REPLY](#))

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)

NEW QUESTION: 182

A user wants to leverage a three column layout on a page. The user also wants to move the mini-cart widget from the right to the center column. How can this requirement be fulfilled?

- A. Gross Layout Override
- B. Subscriber Template
- C. Page Include
- D. HandleBar Template Override

Answer: A ([LEAVE A REPLY](#))

Explanation

To leverage a three column layout on a page and move the mini-cart widget from the right to the center column, the requirement can be fulfilled by creating a Gross Layout Override. This is a custom Visualforce page that overrides the default layout of a page and allows changing its structure and content. The user can create a Gross Layout Override that uses a three column layout and places the mini-cart widget in the center column. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Gross Layout Overrides

NEW QUESTION: 183

How should a developer get the grand total amount, including shipping and tax, for items in the cart and in the currency of the cart, when developing a new Lightning web component for an Aura storefront cart page?

- A. `{!Cart.Details.grandTotal}`
- B. `{!Cart.Totals.grand Total}`
- C. `{!Cart.Details.Fields.grandTotal}`
- D. `{!Cart.Fields.grandTotal}`

Answer: ([SHOW ANSWER](#))

According to the B2B Commerce Developer Guide, the ICart interface provides access to the cart object and its related data. The Details property of the ICart interface returns an ICartDetails

object, which contains information about the cart such as the currency, the subtotal, the shipping cost, the tax, and the grand total. The Fields property of the ICartDetails interface returns a map of field names and values for the cart object.

Therefore, to get the grand total amount for items in the cart and in the currency of the cart, a developer should use the expression {ICart.Details.Fields.grandTotal}, which returns the value of the grandTotal field from the cart object. References: B2B Commerce Developer Guide, ICart Interface, ICartDetails Interface

NEW QUESTION: 184

A developer has made some changes to the products of an existing storefront, but they are unable to see the changes in the products from the store.

Which action did the developer forget to take?

- A. Activate the product list.
- B. Publish the storefront.
- C. Rebuild the search index.
- D. Publish the catalog

Answer: B (LEAVE A REPLY)

Explanation

The developer forgot to publish the storefront after making changes to the products. Publishing the storefront is a necessary step to make the changes visible to the store visitors. Publishing the storefront updates the site's metadata and content, and also triggers the site index and search index rebuilds. If the developer does not publish the storefront, the changes will not be reflected on the live site.

The following actions are not required or relevant for the scenario:

- * Activate the product list. This action is only needed when creating a new product list or modifying an existing one. It does not affect the visibility of the products on the storefront.
- * Rebuild the search index. This action is automatically performed when the developer publishes the storefront. It is not a separate step that the developer needs to take manually.
- * Publish the catalog. This action is not applicable for Salesforce B2B Commerce for Visualforce, which does not use catalogs. Catalogs are only used by Salesforce B2B Commerce for Lightning Experience.

References:

- * Finalize and Publish Your Storefront
- * Customize the D2C Store Template
- * Configure Public Pages for a Storefront

NEW QUESTION: 185

Which service method should be overridden in order to allow "without sharing" queries?

- A. ccrz.ccService.queryWithoutSharing()
- B. ccrz.ccAPI.queryService()
- C. ccrz.ccService.query()

D. ccrz.ccService.initSVCDAO()

Answer: D (LEAVE A REPLY)

The service method that should be overridden in order to allow "without sharing" queries is ccrz.ccService.initSVCDAO. This method is responsible for initializing the service data access object (SVCDAO) that is used by the service class to perform queries. By overriding this method, the user can specify the sharing mode of the SVCDAO, which will determine whether the queries respect or ignore the sharing rules of the current user. For example, ccrz.ccService.initSVCDAO(ccrz.ccAPI.SZ_WITHOUTSHARING) will initialize the SVCDAO with the without sharing mode. Salesforce References: B2B Commerce and D2C Commerce Developer Guide, Service Classes, ccService Class

NEW QUESTION: 186

What is the recommended method for increasing the number of required autocomplete characters that are typed before autocomplete works?

- A. Update the...minLength property in CC Admin, then build and activate a new cache.
- B. Override and extend the autoComplete method in cc_hk_Catalog.
- C. Override the autoComplete.search_input.minLength value in the CCRZ.uiProperties file
- D. Override the autocomplete template and create and register a new handlebars helper.

Answer: (SHOW ANSWER)

NEW QUESTION: 187

Northern Tail Outfitters (NTO) is converting an existing aura component into a Lightning Web Component.

The aura component has the following source code:

```
<!-- owner.cmp -->
<aura:component>
  <aura:attribute name="fName" type="String" />
  <c:child firstName="{!v.fName}" />
</aura:component>
```

What is the equivalent of this code in a Lightning Web Component?

```
<!-- owner.html -->
<template>
  <c-child first-name={!fName}></child>
</template>
```

A.

```
<!-- owner.html -->
<template>
  <c-child first-name={fName}></child>
</template>
```

B.

```
<!-- owner.html -->
<template>
  <c-child first-name={<fName>}></child>
</template>
```

C.

```
<!-- owner.html -->
<template>
  <c-child first-name={item.fName}></child>
</template>
```

D.

Answer: B (LEAVE A REPLY)

The equivalent of this code in a Lightning web component is option B. Option B uses the `@api` decorator to expose `firstName` as a public property of the Lightning web component and communicate it with other components or services. The `@api` decorator is a decorator that marks a property or method as public, which means that it can be accessed by other components or services that use or consume this component. The `@api` decorator also makes the property reactive, which means that it can track changes and update the component accordingly. In option B, `firstName` is exposed as a public property using the `@api` decorator and passed to the child element using an attribute. Option A is incorrect because it uses an invalid syntax for exposing `firstName` as a public property. The `@api` decorator should be used before the property declaration, not after it. Option C is incorrect because it uses an invalid syntax for exposing `firstName` as a public property. The `@api` decorator should be used with parentheses, not without them. Option D is incorrect because it uses an invalid syntax for exposing `firstName` as a public property. The `@api` decorator should be used with camel case, not with hyphens. Salesforce References: Lightning Web Components Developer Guide: Communicate with Properties, Lightning Web Components Developer Guide: Decorators

NEW QUESTION: 188

A developer has created a custom Lightning web component to display on the Product Detail page in the store.

When the developer goes to add the component to the page in Experience Builder, it is missing from the list of custom components.

Which XML fragment should the developer include in the component's configuration XML file to ensure the custom component is available to add to the page?

```
<isExposed>true</isExposedTrue>
<targets>
<target>lightningCommunity_Page</target>
<target>sales/force
```

A.

```
<builder>ExperienceCloud</builder>
<target>RecordPage</target>
```

B.

```
<isExposed target="ExperienceCloud">
<pageType>RecordPage</pageType>
</isExposed>
```

C.

```
<isAvailable>true</isAvailable>
<targets>lightningCommunity_RecordPage<targets>
```

D.

Answer: B (LEAVE A REPLY)

The XML fragment that the developer should include in the component's configuration XML file to ensure the custom component is available to add to the page is option B. Option B uses the <targets> tag to specify where the component can be used in an app. The <targets> tag contains a list of <target> tags, each of which defines a valid target for the component. The value of the <target> tag for the Product Detail page in the store is lightningCommunity__RecordPage, which means that the component can be used on any record page in a Lightning community. Option A is incorrect because it uses an invalid value for the <target> tag. There is no such target as lightningCommunity__ProductDetailPage. Option C is incorrect because it uses an invalid syntax for the <targets> tag. The <targets> tag should not have any attributes, such as isAvailable. Option D is incorrect because it uses an invalid syntax for the <target> tag. The <target> tag should not be self-closing, but rather have a closing tag (</target>). Salesforce References: Lightning Web Components Developer Guide: Configure Components for Lightning Communities, Lightning Web Components Developer Guide: Configure Components for Different Pages and Apps

NEW QUESTION: 189

Which interface does a developer have to implement to override Inventory in Checkout?

- A. sfdc_commerce.ValidationCartinventory
- B. sfdc_commerce.CartinventoryValidation
- C. sfdc_checkout.InventoryCartVvalidation
- D. sfdc_checkout.CartinventoryValidation

Answer: (SHOW ANSWER)

To override inventory in the checkout process, a developer must implement the `sfdc_checkout.CartInventoryValidation` interface. This interface provides the necessary methods for custom inventory validation logic, allowing developers to define how inventory checks are conducted during the checkout process. Salesforce documentation on extending and customizing the checkout functionality in B2B Commerce would include guidance on implementing this interface to meet specific inventory validation requirements.

NEW QUESTION: 190

Which two are considered discrete units of work (code units) within a transaction in the debug logs?

- A. Validation rule
- B. Lightning component load
- C. Web service invocation
- D. Apex class

Answer: C,D (LEAVE A REPLY)

Two data types that are considered discrete units of work (code units) within a transaction in the debug logs are web service invocation and Apex class. A discrete unit of work (code unit) is a segment of executable code that runs as part of a transaction in Salesforce. A transaction is a sequence of operations that are treated as a single unit of work and are executed under certain isolation and consistency rules. A transaction can consist of one or more discrete units of work (code units) that are executed sequentially or concurrently depending on various factors such as triggers, asynchronous calls, or limits. A debug log is a record of database operations, system processes, and errors that occur when executing a transaction or running unit tests in Salesforce. A debug log can capture information about each discrete unit of work (code unit) within a transaction, such as its start time, end time, duration, events, variables, and limits. A web service invocation is a type of discrete unit of work (code unit) that involves calling an external web service from Apex code using SOAP or REST protocols. A web service invocation can be synchronous or asynchronous depending on the method used to make the callout. A web service invocation can be captured in a debug log with its details and results. An Apex class is another type of discrete unit of work (code unit) that involves executing Apex code that defines a class with properties and methods. An Apex class can be invoked from various sources such as triggers, Visualforce pages, Lightning components, or API calls. An Apex class can be captured in a debug log with its details and results. A validation rule is not a type of discrete unit of work (code unit) within a transaction in the debug logs, as it is not a segment of executable code but rather a formula expression that defines a business rule for a field or object. A validation rule can be evaluated during a transaction and cause an error if the rule condition is not met, but it cannot be captured as a separate code unit in a debug log. A Lightning component load is not a type of discrete unit of work (code unit) within a transaction in the debug logs either, as it is not a segment of executable code but rather an event that occurs when a Lightning component is rendered on a web page or application. A Lightning component load can be measured by various performance tools such as Lightning Inspector or Lighthouse, but it cannot be captured as a

separate code unit in a debug log. Salesforce References: [Salesforce Developer Blog: Transactions and Request Processing], [Salesforce Help: Debug Logs], [Salesforce Developer Guide: Invoking Callouts Using Apex], [Salesforce Developer Guide: Apex Classes], [Salesforce Help: Validation Rules], [Salesforce Developer Blog: Measuring Lightning Component Performance]

NEW QUESTION: 191

A new payment type for the Checkout flow has been implemented. Which three descriptors follow best practice for possible configuration metadata are needed to enable a flow? (3 answers)

- A. Cart
- B. *.Edit
- C. Checkout
- D. *.pay
- E. *.New

Answer: B,D,E ([LEAVE A REPLY](#))

Valid B2B-Commerce-Developer Dumps shared by Actual4test.com for Helping Passing B2B-Commerce-Developer Exam! Actual4test.com now offer the **newest B2B-Commerce-Developer exam dumps**, the Actual4test.com B2B-Commerce-Developer exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com B2B-Commerce-Developer dumps with Test Engine here: https://www.actual4test.com/B2B-Commerce-Developer_examcollection.html (223 Q&As Dumps, **30%OFF Special Discount: Freepdfdumps**)