Practice Set 2

Your results are here!! for" Salesforce Platform Developer 1 Practice Test 2 [New] "

0 of 41 questions answered correctly

Your time: 00:00:09

Your Final Score is : 0

You have attempted : 0

Number of Correct Questions : 0 and scored 0

Number of Incorrect Questions : 0 and Negative marks 0

Average score	63.28%
Your score	0%

You can review your answers by clicking view questions.

Important Note : Open Reference Documentation Links in New Tab (Right Click and Open in New Tab).

Restart Test View Answers

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

Answered Review

Review question Pause Summary

1. Question

In a single record, a user selects multiple values from a multi-select picklist. How are the selected values represented in Apex?

- As a String with each value separated by a comma
- As a Set with each value as an element in the set
- **As a String with each value separated by a semicolon**
- As a List with each value as an element in the list Previous

Unattempted

In a single record, a user selects multiple values from a multi – select pick list and the selected values are represented in apex as a string with each value separated by semi colon.

2. Question

Which trigger event allows a developer to update fields in the Trigger.new list without using an additional DML statement? Choose 2 answers

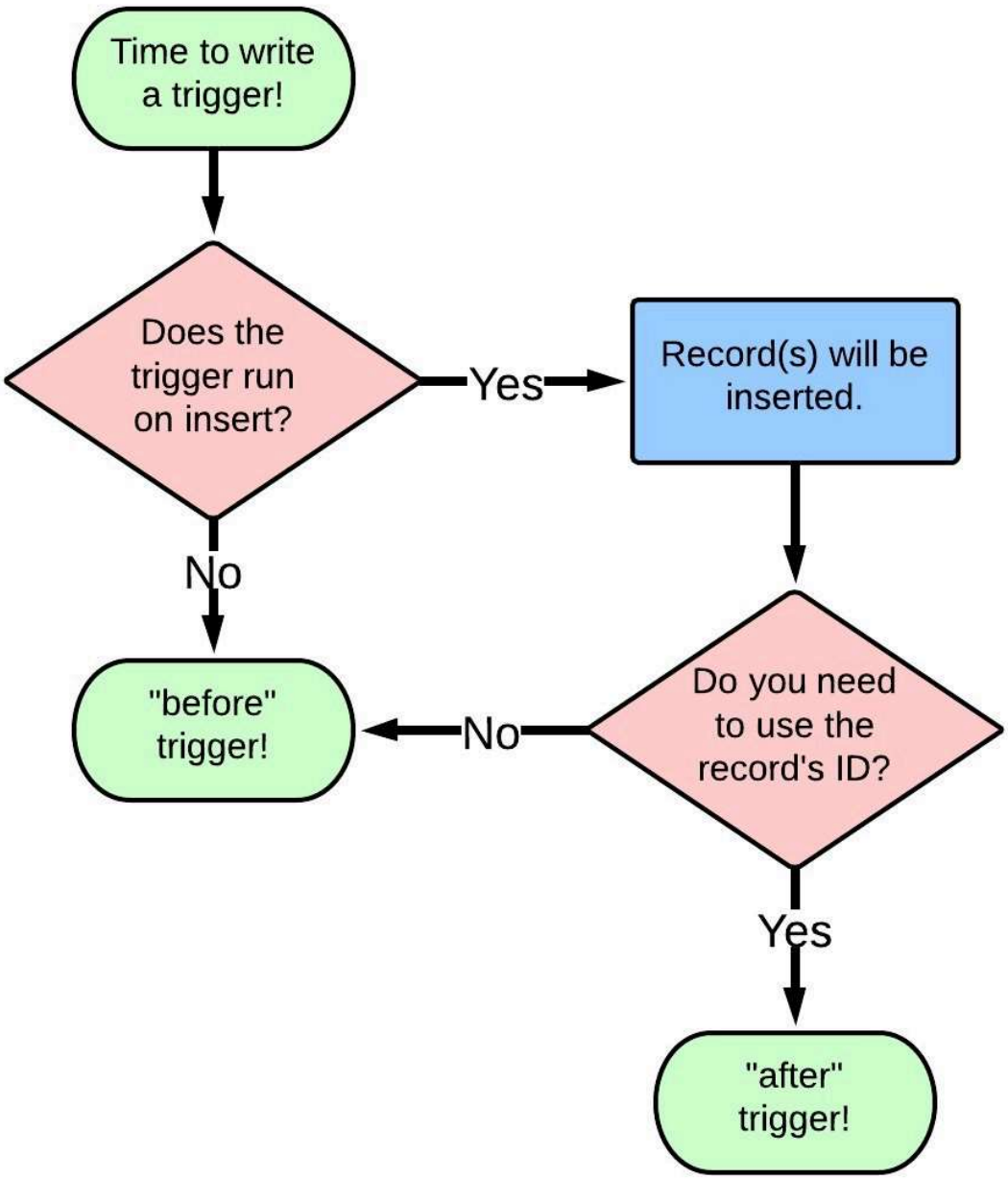
- [ ] **Before Insert**
- [ ] **Before Update**
- [ ] After Update
- [ ] After Insert

Unattempted

Before triggers: used to update or validate record values before they're saved to the database.

After triggers: used to access field values that are set by the system (such as a record's Id or LastModifiedDate field) and to effect changes in other records. The records that fire the after trigger are read-only.

We cannot use After trigger if we want to update a record because it causes read only error. This is because after inserting or updating, we cannot update a record.



3. Question

Which statement would a developer use when creating test data for products and price books?

- Id pricebookId= Test.getStandardPricebookId();
- Pricebook pb= new Pricebook();
- IsTest(SeeAllData= false);
- List objList=Test.loadData(Account.sObjectType,'myResource');

Unattempted

The tag PricebookId tells the system to filter product search results by a single price book ID. Developers can therefore create price book entries for standard and custom price books in Apex tests.

This method allows the system to pull the standard price book ID and then uses that ID to create a price book entry for a product that has a standard price. A custom price book can then be created based on this entry, with its own ID.

4. Question

Where would a developer build a managed package?

- Developer Sandbox
- Unlimited Edition
- Partial Copy Sandbox
- **Developer Edition**

Unattempted

The correct answer to this question is Developer Edition. This edition, used in the cloud based software, Salesforce is used to test and develop applications. The data in this edition is not critical to the business. Developer editions are free and they take up less storage.

They also have less users than other editions. There is no limit on how many developer editions one can sign up for and it is primarily used as the Initial stage of development. Using developer edition allows one to create an application, which can be used in any production environment on the Salesforce software.

A managed package is a collection of application components that are posted as a unit on AppExchange, and are associated with a namespace and a License Management Organization.

You must use a Developer Edition organization to create and work with a managed package.

Difference between managed and unmanaged package in Salesforce:

Unmanaged Package

Unmanaged packages are typically used to distribute open-source projects or application templates to provide developers with the basic building blocks for an application.

Once the components are installed from an unmanaged package, the components can be edited in the organization they are installed in.

Managed Package

Managed packages are typically used by salesforce.com partners to distribute and sell applications to customers.

Once the components are installed from a managed package, the components cannot be edited in the organization they are installed in.

5. Question

What are two benefits of the Lightning Component framework? (Choose two)

- [ ] It simplifies complexity when building pages, but not applications.
- [ ] **It provides an event-driven architecture for better decoupling between components.**
- [ ] **It promotes faster development using out-of-box components that are suitable for desktop and mobile devices.**
- [ ] It allows faster PDF generation with Lightning components.

Unattempted

Why Use the Lightning Component Framework?

There are many benefits of using the Lightning Component framework to build components and apps.

Out-of-the-box Components

Comes with an out-of-the-box set of components to kick start building apps. You don't have to spend your time optimizing your apps for different devices as the components take care of that for you.

Rich Component Ecosystem

Create business-ready components and make them available in the Salesforce app, Lightning Experience, and Communities. Salesforce app users access your components via the navigation menu. Customize Lightning Experience or Communities using drag-and-drop components on a Lightning Page in the Lightning App Builder or using Experience Builder. Additional components are available for your org in the AppExchange. Similarly, you can publish your components and share them with other users.

Fast Development

Empowers teams to work faster with out-of-the-box components that function seamlessly with desktop and mobile devices. Building an app with components facilitates parallel design, improving overall development efficiency.

Components are encapsulated and their internals stay private, while their public shape is visible to consumers of the component. This strong separation gives component authors freedom to change the internal implementation details and insulates component consumers from those changes.

Device-aware and Cross Browser Compatibility

Apps use responsive design and support the latest in browser technology such as HTML5, CSS3, and touch events.

6. Question

A method is passed a list of generic sObjects as a parameter. What should the developer do to determine which object type (Account,Lead or Contact for example) to cast each sObject?

- Use the first three characters of the sObject ID to determine the sObject type.
- **Use the getSObjectType method on each generic sObject to retrieve the sObject token**
- Use the getSObjectName method on the sObject class to get the sObject name
- Use a try-catch construct to cast the sObject into one of the three sObject types

Unattempted

SObject Class: SObject methods are all instance methods: they are called by and operate on an sObject instance such as an account or contact. The following are the instance methods for sObjects.

getSObjectType()
Returns the token for this SObject. This method is primarily used with describe information.

```
Account acc = new Account(name = 'Acme', description = 'Acme Account');
Schema.SObjectType expected = Schema.Account.getSObjectType();
System.assertEquals(expected, acc.getSObjectType());
```

7. Question

When viewing a Quote, the sales representative wants to easily see how many discounted items are included in the Quote Line items. What should a developer do to meet this requirement?

- Create a trigger on the Quote object that queries the Quantity field on discounted Quote Line Items
- Create a Workflow Rule on the Quote Line Item object that updates a field on the parent Quote when the item is discounted.
- **Create a roll-up summary field on the Quote object that performs a SUM on the quote Line Item Quantity field, filtered for only discounted Quote Line Items**
- Create a formula field on the Quote object that performs a SUM on the Quote Line Item Quantity field, filtered for only discounted Quote Line Items.

Unattempted

The most efficient and scalable solution to meet the requirement is to create a **roll-up summary field** on the Quote object.

Here's why:

- **Real-time calculation:** Roll-up summary fields automatically calculate the specified aggregation (in this case, SUM) based on the related child records (Quote Line Items). This ensures that the count of discounted items is always up-to-date on the Quote.
- **Filtering:** Roll-up summary fields allow you to apply filters to the child records, so you can easily filter for only discounted Quote Line Items.
- **Performance:** Roll-up summary fields are optimized for performance, especially when dealing with large datasets.
- **Ease of use:** Once the roll-up summary field is created, the sales representative can simply view the Quote to see the count of discounted items.

Other options, like triggers or workflow rules, might be more complex to implement and maintain, and they might not provide real-time updates or handle large datasets as efficiently.

8. Question

Which two Apex data types can be used to reference a Salesforce record ID dynamically? (Choose two)

- [ ] ENUM
- [ ] **sObject**
- [ ] External ID
- [ ] **String**

Unattempted

sObject and String are the two apex date types used for reference a Salesforce record ID dynamically.

Explanation:The sObject type defines an object's existence in force.com, a generic and a unique data type that can be used in either standard or custom objects. The real purpose of the sObject is that it takes the actual meaning for portrayal of any object that it is designated to it. For example: Fruit is a generic type whereas pineapple, guava, orange is all concrete types of sObject fruit. And strings similar to other programming languages, is also used to store characters of infinite length.

9. Question

Where can a developer identify the time taken by each process in a transaction using Developer Console log inspector?

- ( ) Performance Tree tab under Stack Tree panel

Execution Tree tab under Stack Tree panel

Timeline tab under Execution Overview panel

Save Order tab under Execution Overview panel

Unattempted

Log Inspector

The Log Inspector is a context-sensitive execution viewer in the Developer Console. It shows the source of an operation, what triggered the operation, and what occurred next. Use this tool to inspect debug logs that include database events, Apex processing, workflow, and validation logic.

The panels displayed in the Log Inspector depend on the selected perspective. To switch perspectives, select Debug | Switch Perspective.

Some features in the Log Inspector, such as filtering and searching, are available only after a debug log has finished loading and processing. To access a log that is still processing, select File | Open Raw Log.

Log Panels

The Log Inspector can contain the following panels:

-Stack Tree

-Execution Stack

-Execution Log

-Source

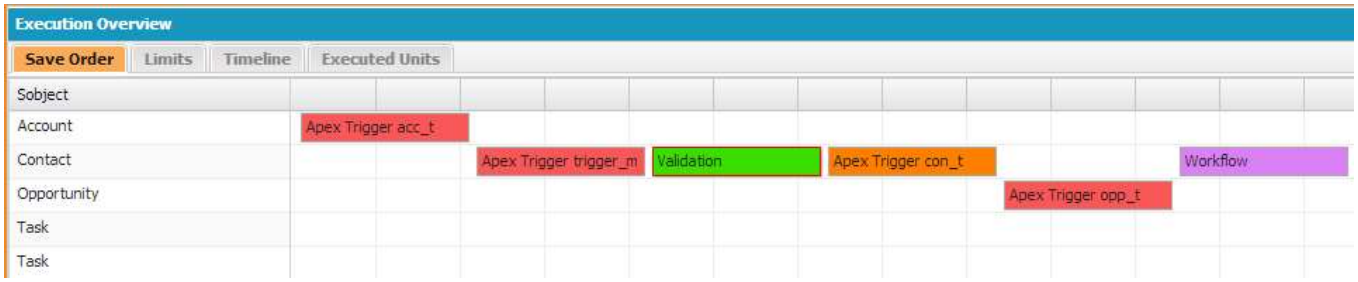
-Variables

-Execution Overview

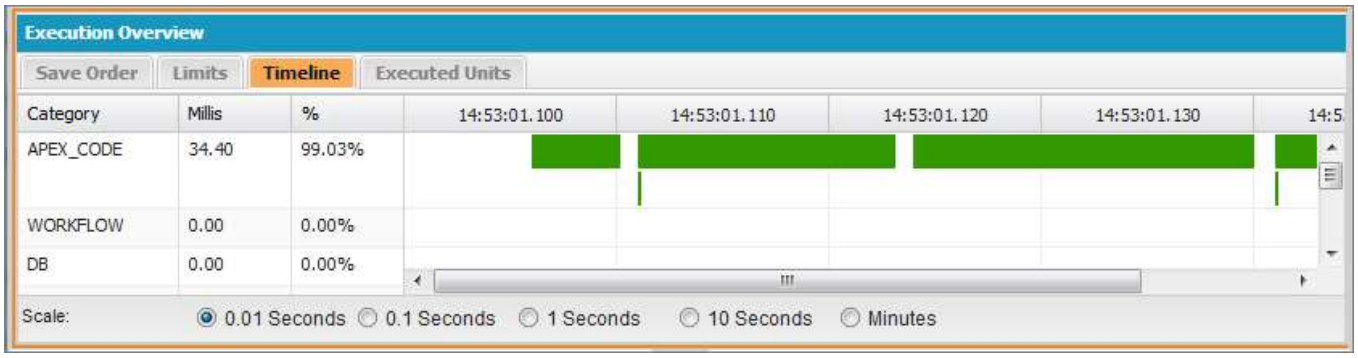
Execution Overview: Save Order, Limits, Timeline, and Executed Units

The Execution Overview panel at the bottom of the Log Inspector contains four tabs:

The Save Order tab displays a color-coded timeline of DML actions. For each DML action taken, save order elements are shown as boxcars in the timeline.



The Timeline tab provides a visual representation of the time taken by each process. Select the Scale option that results in the most useful view.



The Timeline tab contains this information:

COLUMN	DESCRIPTION
Category	Type of process.
Millis	Milliseconds of time taken by the process.
%	Percent the process took of the entire request.

10. Question

Which two platform features align to the Controller portion of MVC architecture? (Choose two)

- [ ] **Process Builder actions**
- [ ] **Workflow rules**
- [ ] Standard objects
- [ ] Date fields

Unattempted

Model View Controller (MVC)

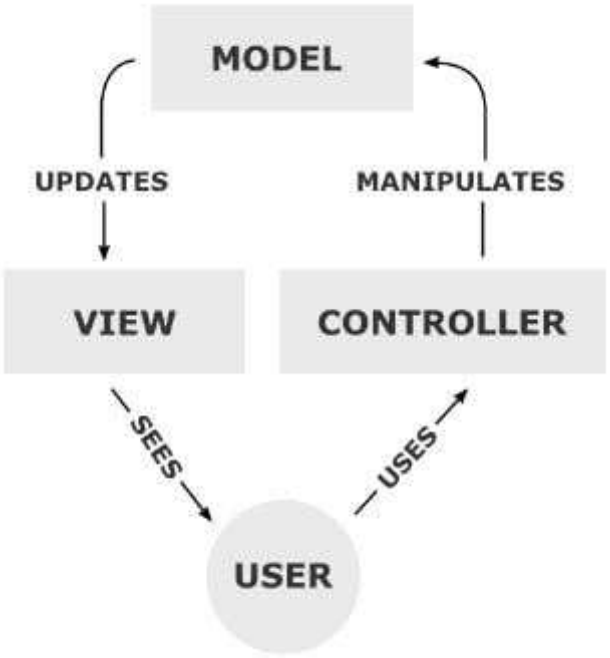
Model view controller (MVC) is a software architecture pattern which separates the representation of information from the user's interaction with it

In addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.

A controller can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).

A model notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications, because the application does not require them or the software platform does not support them.

A view requests from the model the information that it needs to generate an output representation.



Model: What schema and data does salesforce uses to represent the system completely. In salesforce, we can say that sObjects are the model as every entity in salesforce is mapped to some sObject.

View: How the schema and data is represented. Visualforce is used to present the data to users.

Controller: How the interface actions. Controllers are used to perform the actions whenever users interact with visual force.

In SFDC

1. Visual Force pages, Page Layouts, Tabs comes under View Layer of Model View controller .

2. Workflows, Apex Classes, Triggers comes under Controller part in Model View controller .

3. Objects, Fields, Relationships comes under Model Layer of Model View Controller .

11. Question

A developer needs to test an Invoicing system integration. After reviewing the number of transactions required for the test, the developer estimates that the test data will total about 2GB of data storage. Production data is not required for the integration testing. Which two environments meet the requirements for testing ? Choose two

- [ ] Developer Sandbox
- [ ] **Full Sandbox**
- [ ] Developer Edition
- [ ] **Partial Sandbox**
- [ ] Developer Pro Sandbox

Unattempted

Developer Sandbox

A Developer sandbox is intended for development and testing in an isolated environment. A Developer Sandbox includes a copy of your production org's configuration (metadata).

Developer Pro Sandbox

A Developer Pro sandbox is intended for development and testing in an isolated environment and can host larger data sets than a Developer sandbox. A Developer Pro sandbox includes a copy of your production org's configuration (metadata). Use a Developer Pro sandbox to handle more development and quality assurance tasks and for integration testing or user training.

Partial Copy Sandbox

A Partial Copy sandbox is intended to be used as a testing environment. This environment includes a copy of your production org's configuration (metadata) and a sample of your production org's data as defined by a sandbox template. Use a Partial Copy sandbox for quality assurance tasks such as user acceptance testing, integration testing, and training.

Full Sandbox

A Full sandbox is intended to be used as a testing environment. Only Full sandboxes support performance testing, load testing, and staging. Full sandboxes are a replica of your production org, including all data, such as object records and attachments, and metadata.

12. Question

A developer working on a time management application wants to make total hours for each timecard available to application users. A timecard entry has a Master-Detail relationship to a timecard. Which approach should the developer use to accomplish this declaratively?

- A visualforce page that calculates the total number of hours for a timecard and displays it on the page.
- **A Roll-up summary field on the Timecard Object that calculates the total hours from timecard entries for that timecard**
- A process builder process that updates a field on the timecard when a timecard entry is created
- An Apex trigger that uses an Aggregate Query to calculate the hours for a given timecard and stores it in a custom field.

Unattempted

Roll-up summary field

A roll-up summary field calculates values from related records, such as those in a related list. You can create a roll-up summary field to display a value in a master record based on the values of fields in a detail record. The detail record must be related to the master through a master-detail relationship. For example, you want to display the sum of invoice amounts for all related invoice custom object records in an account's Invoices related list. You can display this total in a custom account field called Total Invoice Amount.

You can perform different types of calculations with a roll-up summary field. You can count the number of detail records related to a master record. Or, you can calculate the sum, minimum value, or maximum value of a field in the detail records.

13. Question

A developer encounters APEX heap limit errors in a trigger. Which two methods should the developer use to avoid this error? (Choose two)

- **Use the transient keyword when declaring variables**
- Query and store fields from the related object in a collection when updating related objects.
- Remove or set collections to null after use
- **Use SOQL for loops instead of assigning large queries results to a single collection and looping through the collection**

Unattempted

We can use transient keyword with apex class's instance variables when we want that values of those variables should not be transferred as part of the view state of visualforce page. This helps us reducing the view state of visualforce page. As we all know that, there is a limit of 135KB of view state and many times this "Transient" keyword helps us to reduce the view state.

There are certain points that we should consider while using the transient keyword:

The value of transient variable is only transferred from controller to visualforce page but not as part of view state. As the value is not part of view state, the changed value of that variable is not transferred back from visualforce page to controller when a new request is made by clicking a button or link.

We should use the transient keyword mostly in case where we have read only Visualforce page and data doesn't need to be sent back to controller in further requests. A common use case for the transient keyword is a field on a Visualforce page that is needed only for the duration of a page request, but should not be part of the page's view state and would use too many system resources to be recomputed many times during a request.

14. Question

Which approach should be used to provide test data for a test class?

- Query for existing records in the database.
- Execute anonymous code blocks that create data
- **Use a test data factory class to create test data.**
- Access data in @TestVisible class variables.

Unattempted

Data Factories are apex classes that are responsible for generating data. They primarily generate Salesforce records in various ways. By far, test classes use Data Factories the most to generate their data.

```
//service provider

public abstract class TestDataFactory {

//Account creation
public static List createAccounts( Integer numberOfAccounts) {
List accounts = new List();
for ( Integer i = 0 ; i < numberOfAccounts ; i++ ) { Account account = new Account( firstname = 'Test Account' + Math.random(), lastname = 'Account', PersonEmail = 'noreplay@email.com',Home_Phone__c= '(415) 419-8873',PersonMailingStreet = '5353 W.Test Rd', PersonMailingCity = 'Testdale', PersonMailingState = 'CA', PersonMailingPostalCode = '94803'); accounts.add( account); } return accounts; } } // Consumer class @isTest private class TestDataFactoryTest { static Account account; //test for creating Accounts static testMethod void createAccountsTest() { account = TestDataFactory.createAccounts(1)[0]; insert account; system.debug('Account Id'+account.id); account = [select firstname, lastname,personmailingpostalcode,personEmail from account where id = :account.id]; system.assertEquals(account.lastname, 'Account'); system.assertEquals(account.personmailingpostalcode, '94803'); system.debug('Account Firstname'+ account.FirstName); } }
```

15. Question

Which approach should a developer take to automatically add a " Maintenance Plan" to each Opportunity that includes an "Annual Subscription" when an opportunity is closed?

- Build a OpportunityLineItem trigger that adds a PriceBookEntry record.
- Build an OpportunityLineItem trigger to add an OpportunityLineItem record
- Build an Opportunity trigger that adds a PriceBookEntry record.
- **Build an Opportunity trigger that adds an OpportunityLineItem record.**

Unattempted
Correct:

✅ D. Build an Opportunity trigger that adds an OpportunityLineItem record. This is the correct approach. The requirement is to add a "Maintenance Plan" (which would be represented as an OpportunityLineItem) to the Opportunity *when the Opportunity is closed*. Therefore, the trigger should be on the Opportunity object. Since the "Maintenance Plan" is a product (and therefore an OpportunityLineItem), you would create a new OpportunityLineItem record within the trigger.

Incorrect:

❌ A. Build an OpportunityLineItem trigger that adds a PriceBookEntry record. This is incorrect for two reasons: 1) The trigger is on the wrong object (it should be on Opportunity, not OpportunityLineItem), and 2) you're adding a PriceBookEntry, which represents a *product* in the price book, not an *instance* of a product on an Opportunity (which is an OpportunityLineItem).

❌ B. Build an OpportunityLineItem trigger to add an OpportunityLineItem record. This is also incorrect because the trigger is on the wrong object. The trigger needs to react to the Opportunity being closed, so it should be on the Opportunity object.

❌ C. Build an Opportunity trigger that adds a PriceBookEntry record. This is incorrect because you're adding a PriceBookEntry. You need to add an OpportunityLineItem (which is an *instance* of a product on an Opportunity), not a PriceBookEntry (which is a *product* in a price book).

16. Question

A visualforce page is required for displaying and editing Case records that includes both standard and custom functionality defined in an Apex class called myControllerExtension. The visualforce page should include attribute(s) to correctly implement controller functionality?

- controller="Case" and extensions="myControllerExtension"
- extensions="myControllerExtension"

controller="myControllerExtension"

standardController="Case" and extensions="myControllerExtension"

Unattempted

What is visualforce controller extension ?

Visualforce controller Extension :-If we want to use both custom controller functionality and standard controller functionality we use extension controllers. Extension Controllers begins with Standard controller and extended or overridden with custom controller with custom apex code.

Syntax :-

17. Question

A newly hired developer discovers that there are multiple triggers on the case object. What should the developer consider when working with triggers?

Developers must dictate the order of trigger execution

Trigger execution order is based on creation date and time

Unit tests must specify the trigger being tested

Trigger execution order is not guaranteed for the same sObject.

Unattempted

Correct:

✅ D. Trigger execution order is not guaranteed for the same sObject. This is a crucial concept for the Platform Developer I exam. When multiple triggers exist on the same object and event (e.g., before insert), Salesforce does *not* guarantee the order in which they will execute. This can lead to unpredictable behavior if the triggers depend on each other.

Incorrect:

❌ A. Developers must dictate the order of trigger execution. Developers *cannot* directly control the order of trigger execution. There's no mechanism to specify a trigger execution sequence.

❌ B. Trigger execution order is based on creation date and time. This is incorrect. The order is *not* reliably based on creation date/time. While it *might* appear that way sometimes, it's not a guarantee and should not be relied upon.

❌ C. Unit tests must specify the trigger being tested. Unit tests do *not* need to specify which trigger is being tested. When a DML operation occurs, *all* relevant triggers on that object will execute (in a non-deterministic

order). Your test should cover the logic in a way that handles any combination of trigger firings. You can't isolate a single trigger for testing.

18. Question

How should a developer prevent a recursive trigger?

- Use a "one trigger per object" pattern
- **Use a static Boolean variable**
- Use a trigger handler
- Use a private Boolean variable

Unattempted
Use a static Boolean variable.

A static Boolean variable is a reliable and efficient way to prevent recursive triggers. Here's how it works:

1. **Initialize the variable:** Declare a static Boolean variable outside of the trigger's code. Initialize it to `false`.
2. **Check the variable:** At the beginning of the trigger's code, check the value of the variable. If it's `true`, exit the trigger to avoid recursion.
3. **Set the variable:** If the variable is `false`, set it to `true` before executing the trigger's logic.
4. **Reset the variable:** After the trigger's logic has finished executing, reset the variable to `false`.

This approach ensures that the trigger will only execute once, even if it's triggered multiple times within the same transaction. It's a simple and effective method to prevent recursive triggers.

Note: While using a trigger handler can also help prevent recursive triggers, it's generally less efficient and requires more complex logic.

19. Question

What is a capability of the tag that is used for loading external Javascript libraries in Lightning Component? (Choose three)

- Loading files from Documents
- **One-time loading for duplicate scripts**
- **Specifying loading order**
- **Loading scripts in parallel**
- Loading externally hosted scripts

Unattempted

https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/js_libs_platform.htm#!

20. Question

A Platform Developer needs to write an Apex method that will only perform an action if a record is assigned to a specific Record Type. Which two options allow the developer to dynamically determine the ID of the required Record Type by its name ? Choose two

- [ ] Make an outbound web services call to the SOAP API
- [ ] Hardcode the ID as a constant in an Apex class
- [ ] **Use the getRecordTypeInfosByName() method in the DescribeSObjectResult class**
- [ ] **Execute a SOQL query on the RecordType object**

Unattempted

getRecordTypeInfosByName()

Returns a map that matches record labels to their associated record type. The current user is not required to have access to a record type to see it in this map.

21. Question

How should a developer make sure that a child record on a custom object, with a lookup to the Account object, has the same sharing access as its associated account?

- ( ) Create a Sharing Rule comparing the custom object owner to the account owner.
- ( ) Create a validation rule on the custom object comparing the record owners on both records
- ( ) Include the sharing related list on the custom object page layout
- ( ) **Ensure that the relationship between the objects is Master-Detail**

Unattempted

Correct:

✅ D. Ensure that the relationship between the objects is Master-Detail. This is the *most direct and effective* way to ensure that the child record has the same sharing access as its associated Account. In a Master-Detail relationship, the child record *inherits* the sharing settings of the parent record. This is a fundamental concept for the Platform Developer I exam.

Incorrect:

❌ A. Create a Sharing Rule comparing the custom object owner to the account owner. While sharing rules *can* be used to grant access, they are not the ideal solution in this case. Sharing rules are designed to *extend* access beyond what's provided by OWDs and role hierarchies. With a Master-Detail relationship, sharing is handled automatically through inheritance, making sharing rules unnecessary and less efficient for this particular requirement.

❌ B. Create a validation rule on the custom object comparing the record owners on both records. A validation rule can *prevent* users from doing something if a condition isn't met, but it doesn't *grant* access. It won't enforce the same sharing model.

❌ C. Include the sharing related list on the custom object page layout. The sharing related list displays the users who have access to the record, but it doesn't *enforce* any particular sharing model. It simply shows the current sharing settings. It's a view, not a mechanism for controlling sharing.

22. Question

Which approach should a developer use to add pagination to a Visualforce page?

- ○ A StandardController
- ○ The Action attribute for a page
- ○ The extensions attribute for a page
- ○ **A StandardSetController**

Unattempted

Correct:

✅ D. A StandardSetController. The `StandardSetController` in Visualforce is specifically designed for handling collections of records and provides built-in support for pagination. It allows you to easily control which records are displayed on each page and provides methods for navigating between pages. This is the recommended approach for adding pagination to Visualforce pages.

Incorrect:

❌ A. A StandardController. While a `StandardController` can be used with Visualforce, it's designed for working with *single* records, not collections. It doesn't have built-in pagination features. You would have to implement pagination logic yourself if you used a `StandardController`.

❌ B. The Action attribute for a page. The `action` attribute is used to specify an Apex method to be called when the page loads or a button is clicked. It's not directly related to pagination. You would still need a controller (likely a `StandardSetController` or a custom controller) to handle the pagination logic, even if you use the `action` attribute.

❌ C. The extensions attribute for a page. The `extensions` attribute is used to add additional controllers to a Visualforce page alongside a standard controller. While you *could* create a custom controller extension to handle pagination, it's much more efficient and simpler to use the `StandardSetController`, which is built specifically for this purpose. It's the most straightforward and best-practice approach for pagination.

23. Question

A developer is asked to create a PDF quote document formatted using the company's branding guidelines and automatically save it to the Opportunity record. Which two ways should a developer create this functionality? (Choose two)

- [ ] **Install an application from the AppExchange to generate documents**
- [ ] **Create a Visualforce page with custom styling**
- [ ] Create an email template and use it in Process Builder
- [ ] Create a visual flow that implements the company's formatting.

Unattempted
Correct:

✅ B. Create a Visualforce page with custom styling. Visualforce pages are excellent for generating custom-formatted documents, including PDFs. You can use CSS and other styling techniques to match the company's branding guidelines. Then, you can use the `<apex:page renderAs="pdf">` attribute to render the page as a PDF.

✅ A. Install an application from the AppExchange to generate documents. There are many AppExchange apps specifically designed for document generation, including PDF creation. These apps often provide pre-built templates and customization options that can help meet branding requirements. This is a valid approach, especially if the formatting is complex and requires specialized tools.

Incorrect:

❌ C. Create an email template and use it in Process Builder. Email templates are designed for email content, not for creating styled PDF documents. While you can attach files to emails, this approach won't provide the level of custom formatting needed for a branded PDF quote.

❌ D. Create a visual flow that implements the company's formatting. While Flows can automate actions and even generate some text-based output, they are not well-suited for creating complex, styled PDF documents. Visualforce or a dedicated document generation app are much better choices for this requirement. Flows are more for automation and less for document design.

24. Question

Which tool allows a developer to send requests to the Salesforce REST APIs and view the responses?

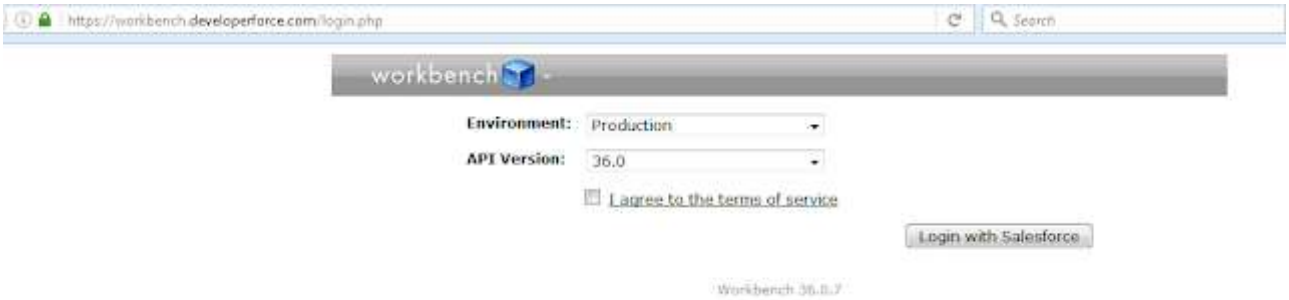
- REST resource path URL
- **Workbench REST Explorer**
- Developer Console REST tab
- Force.com IDE REST Explorer tab

Unattempted

How to open REST Explorer?

1) Log in to your developer organization.

2) Open a new browser tab and navigate to https://workbench.developerforce.com/login.php. and login after click on I agree button.

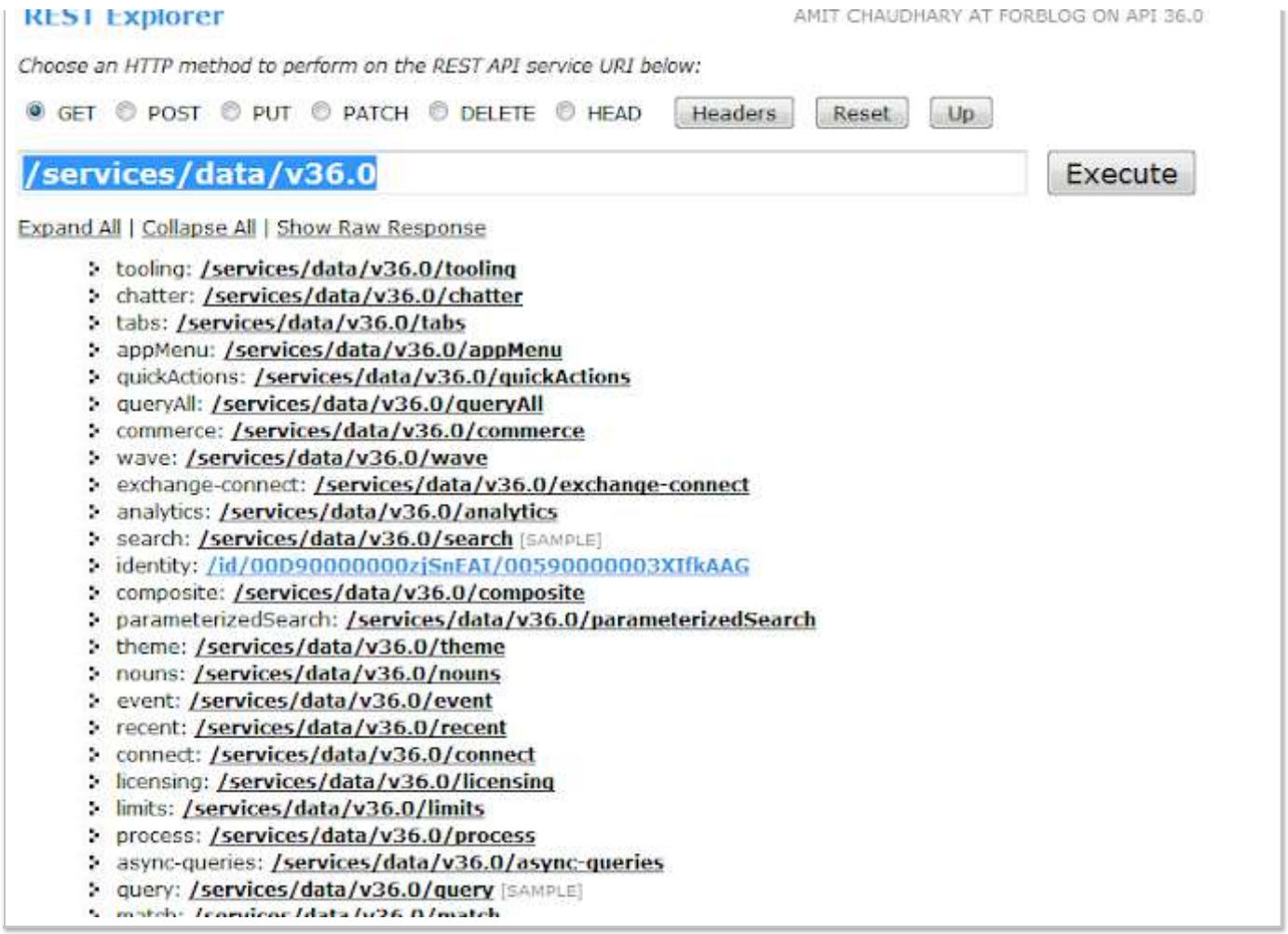


3) Then Open REST Explorer. Click Utilities | REST Explorer.

4) If you want to see REST Explorer service then use below URL and click on execute button

URL:- /services/data/v36.0

Method :- Get



25. Question

What is a benefit of using an after insert trigger over using a before insert trigger?

- An after insert trigger allows a developer to bypass validation rules when updating fields on the new record.
- **An after insert trigger allows a developer to insert other objects that reference the new record.**
- An after insert trigger allows a developer to make a callout to an external service
- An after insert trigger allows a developer to modify fields in the new record without a query

Unattempted

Correct:

✅ B. An after insert trigger allows a developer to insert other objects that reference the new record. This is a key benefit. In a `before insert` trigger, the record is not yet committed to the database, so you cannot use its ID to create related records. The ID is only available *after* the insert. Therefore, if you need to create child records or perform other actions that rely on the newly inserted record's ID, you must use an `after insert` trigger.

Incorrect:

✗ A. An after insert trigger allows a developer to bypass validation rules when updating fields on the new record. This is incorrect. Neither `before` nor `after` triggers can bypass validation rules. Validation rules are always enforced.

✗ C. An after insert trigger allows a developer to make a callout to an external service. While it's *possible* to make callouts from an `after insert` trigger, it's generally *not recommended* due to governor limits and asynchronous processing requirements. Callouts should ideally be done in a separate asynchronous process (e.g., Queueable Apex, Batch Apex) after the trigger completes. The question implies it is *only* possible in an after insert trigger, which is incorrect.

✗ D. An after insert trigger allows a developer to modify fields in the new record without a query. This is incorrect. Both `before` and `after` triggers can modify fields of the new record. However, `before` triggers do it directly by modifying the trigger record itself, whereas `after` triggers generally need to do a DML `update` because they operate on records that already exist in the database (although not yet committed in the same transaction) and are accessed via SOQL queries. `before insert` triggers tend to be more efficient when modifying records than `after insert` triggers.

26. Question

An org has a single account named 'NoContacts' that has no related contacts. Given the query:
List accounts=[Select Id,(select Id, Name from Contacts) from Account where Name='NoContacts'];
What is the result of running this Apex?

- accounts[0].contacts is invalid Apex
- **accounts[0].contacts is an empty Apex**
- accounts[0].contacts is Null
- A QueryException is thrown

Unattempted
Correct:

✅ B. `accounts[0].contacts` is an empty List. The query will execute successfully. Because the Account 'NoContacts' *exists*, the query will return a list containing that Account. However, since there are *no* related Contacts, `accounts[0].contacts` will be an *empty* list (a list with zero elements), not `null`. This is a crucial distinction for the Platform Developer I exam.

Incorrect:

✗ A. `accounts[0].contacts` is invalid Apex. This is incorrect. The syntax is perfectly valid. SOQL queries with parent-child relationships like this are common.

✗ C. `accounts[0].contacts` is Null. This is the key point of confusion. It's an *empty* list, not `null`. An empty list is a valid list object that simply contains no elements. `null` means the variable hasn't been

initialized at all. The query *will* return a list; it will just be an empty list.

✗ D. A QueryException is thrown. This is incorrect. The query is syntactically correct, and the Account 'NoContacts' *does* exist. There's no reason for a `QueryException` to be thrown. The lack of related Contacts does not cause an exception; it simply results in an empty list.

27. Question

Using the Schema Builder, a developer tries to change the API name of a field that is referenced in an Apex test class. What is the end result?

- The API name is not changed and there are no other impacts.
- The API name of the field and the reference in the test class is changed
- **The API name of the field is changed, and a warning is issued to update the class.**
- The API name of the field and the reference in the test class is updated

Unattempted

Correct:

✓ **C. The API name of the field is changed, and a warning is issued to update the class.**

Explanation:

When a developer changes the API name of a field in the Schema Builder, Salesforce will allow the change. However, if the field is referenced in an Apex test class or any other Apex code, Salesforce will issue a warning that the field reference in the class is now outdated and needs to be updated. The API name of the field will be updated, but the test class will still reference the old API name unless manually updated.

Incorrect:

✗ **A. The API name is not changed and there are no other impacts.**

This option is incorrect because changing the API name of a field in Schema Builder will change the name of the field in the schema. However, it will not automatically propagate the change to any code that references the field, which leads to issues with outdated references.

✗ **B. The API name of the field and the reference in the test class is changed.**

This is incorrect because the field's API name will change, but the reference in the test class is not automatically updated. The developer needs to manually update the reference in the test class.

✗ **D. The API name of the field and the reference in the test class is updated.**

This is incorrect because while the API name will be updated, the reference in the test class will not automatically update. Developers must manually update the references to reflect the new API name in the test class.

28. Question

When is an Apex Trigger required instead of a Process Builder Process?

- When a record needs to be created
- When multiple records related to the triggering record need to be updated
- When a post to Chatter needs to be created
- **When an action needs to be taken on a delete or undelete, or before a DML operation is executed.**

Unattempted

Process Builders cannot handle before DML It executes after a record has been created or updated. Whereas Apex triggers can handle both before and after DML operations.

Process Builder cannot handle delete and undelete DML. Whereas Apex triggers can handle all DML operations.

An error reported in Process Builder is more generic which makes it difficult to find the origin of the error. With Apex triggers, exception handling can be made more specific.

29. Question

A developer needs to join data received from an integration with an external system with parent records in Salesforce. The data set does not contain the Salesforce IDs of the parent records, but it does have a foreign key attribute that can be used to identify the parent. Which action will allow the developer to relate records in the data model without knowing the Salesforce ID?

- Create and populate a custom field on the parent object marked as Unique.
- Create a custom field on the child object of type External Relationship
- **Create and populate a custom field on the parent object marked as an External ID**
- Create a custom field on the child object of type Foreign Key

Unattempted

The correct answer is:

Create and populate a custom field on the parent object marked as an External ID.

Here's why:

- **External ID fields** are designed to uniquely identify records across systems, even if they don't have a Salesforce ID. This is perfect for matching records from an external system with their corresponding parent records in Salesforce.

- **Creating the field on the parent object** ensures that the match can be made based on the data from the external system, which is likely to have the foreign key attribute.
- **Marking the field as an External ID** indicates to Salesforce that it's a unique identifier that can be used to match records across systems.

The other options are not as suitable:

- **Creating a custom field on the parent object marked as Unique:** While this can help ensure uniqueness within Salesforce, it doesn't guarantee a match with the external system's data.
- **Creating a custom field on the child object of type External Relationship:** External Relationships are primarily used for relating records across different Salesforce orgs, not for matching records from an external system.
- **Creating a custom field on the child object of type Foreign Key:** Foreign Key fields are typically used to reference related records within Salesforce, not to match records from an external system.

30. Question

A developer created a Lightning component to display a short text summary for an object and wants to use it with multiple Apex classes. How should the developer design the Apex classes?

- Have each class define method getObject() that returns the sObject that is controlled by the Apex class.
- Extend each class from the same base class that has a method getTextSummary() that returns the summary
- **Have each class implement an interface that defines method getTextSummary() that returns the summary**
- Have each class define method getTextSummary() that returns the summary

Unattempted
Correct:

✅ C. Have each class implement an interface that defines method `getTextSummary()` that returns the summary. This is the best approach. Using an interface establishes a contract. The Lightning component can then interact with *any* Apex class that implements the interface, knowing that it will have a `getTextSummary()` method. This promotes loose coupling and makes the component reusable with different Apex classes without modification.

Incorrect:

❌ A. Have each class define method `getObject()` that returns the sObject that is controlled by the Apex class. While this might work, it's less flexible. The Lightning component would then need to know *which* Apex class it's working with and call the appropriate `getObject()` method. This creates a tighter coupling. The component would need to be updated if you wanted to use it with a different Apex class.

✗ B. Extend each class from the same base class that has a method `getTextSummary()` that returns the summary. While inheritance *can* work, it's not as flexible as an interface. With inheritance, the Apex classes are now tightly coupled to a specific inheritance hierarchy. If you later need to use one of these classes in a different inheritance structure, you'd have problems. Interfaces are preferred for this reason.

✗ D. Have each class define method `getTextSummary()` that returns the summary. While this will technically work if all the classes happen to have the same method signature, it's not a robust or maintainable solution. There's no guarantee that future developers will adhere to this naming convention. An interface provides a *formal contract* and ensures consistency. It's the best practice for loose coupling and reusability.

31. Question

Which two strategies should a developer use to avoid hitting governor limits when developing in a multi-tenant environment? Choose two

- ☐ Use collections to store all fields from a related object and not just minimally required fields
- ☐ **Use methods from the "Limits" class to monitor governor limits.**
- ☐ **Use SOQL for loops to iterate data retrieved from queries that return a high number of rows**
- ☐ Use variables within Apex classes to store large amounts of data

Unattempted

The Limits methods return the specific limit for the particular governor, such as the number of calls of a method or the amount of heap size remaining.

Because Apex runs in a multitenant environment, the Apex runtime engine strictly enforces a number of limits to ensure that runaway Apex doesn't monopolize shared resources.

None of the Limits methods require an argument. The format of the limits methods is as follows:

myDMLLimit = Limits.getDMLStatements();

32. Question

Which set of roll-up types are available when creating a roll-up summary field?

- ○ **COUNT, SUM, MIN, MAX**
- ○ AVERAGE,SUM,MIN, MAX
- ○ SUM, MIN, MAX
- ○ AVERAGE, COUNT, SUM, MIN, MAX

Unattempted

Note : Roll up summary field can only be defined on the master object.

The Roll up Summary field is basically of 4 types:

Count

Sum

Min

Max

33. Question

Which three tools can deploy metadata to production? (Choose three)

- [ ] Change Set from Developer Org
- [ ] **Force.com IDE**
- [ ] Data Loader
- [ ] **Change Set from Sandbox**
- [ ] **Metadata API**

Unattempted
Correct Answers:

✅ **B. Force.com IDE:**
The **Force.com IDE** is a development tool that allows developers to deploy metadata to production. It is particularly useful for deploying Apex classes, triggers, and other metadata components.

✅ **D. Change Set from Sandbox:**
Change Sets are a common method for deploying metadata from a sandbox to a production org. They allow administrators and developers to bundle and deploy metadata changes in a controlled manner.

✅ **E. Metadata API:**
The **Metadata API** is a powerful tool for deploying metadata to production. It is used by various tools (e.g., Salesforce CLI, Workbench) to retrieve and deploy metadata programmatically.

Incorrect Answers:

❌ **A. Change Set from Developer Org:**
This is incorrect because **Change Sets** cannot be sent directly from a **Developer Org** to a production org. Change Sets are only supported between sandbox and production orgs.

✗ C. Data Loader:

This is incorrect because the **Data Loader** is used for importing, exporting, and updating data records, not for deploying metadata. It does not have the capability to deploy metadata components like Apex classes, triggers, or custom objects.

34. Question

How should a developer avoid hitting the governor limits in test methods?

- Use @TestVisible on methods that create records
- Use Test.loadData() to load data from a static resource
- Use @isTest (SeeAllData=true) to use existing data
- **Use Test.startTest() to reset governor limits**

Unattempted

Set up your test data in @testSetup. To avoid using your test governor limits, start with Test.startTest()

```
@isTest class MyUnitTestClass {
@testSetup static void testSetup() {
Test.startTest();
// Do your DML operations here
Test.stopTest();
}
@isTest static void test() {

Test.startTest();
// Test your business logic here
Test.stopTest();
}
}
```

35. Question

Why would a developer consider using a custom controller over a controller extension?

- To increase the SOQL query governor limits
- **To implement all of the logic for a page and bypass default Salesforce functionality**
- To leverage built-in functionality of a standard controller
- To enforce user sharing settings and permissions

Unattempted

Controller extension leverages the built-in-functionality of Standard Controller, but overrides one or more actions.

Where as, Custom Controller replaces the functionality of Standard Controller without leveraging it.

36. Question

A developer wants to override a button using Visualforce on an object. What is the requirement?

- The controller or extension must have a PageReference method
- **The standardController attribute must be set to the object**
- The action attribute must be set to a controller method
- The object record must be instantiated in a controller or extension

Unattempted

When a developer wants to override a button using visualforce on an object, the standardController attribute must be set to the object. In general, a controller is an Apex class which developers use to implement many of the positive features of visualforce, such as its logic.

Standard controllers are often used in coding to provide guidelines for standard functions like delete, save, and create records.

37. Question

How should a developer create a new custom exception class?

- **public class CustomException extends Exception{}**
- CustomException ex=new (CustomException) Exception();
- public class CustomException implements Exception {}
- (Exception)CustomException ex=new Exception();

Unattempted

Create Custom Exceptions

You can't throw built-in Apex exceptions. You can only catch them. But with custom exceptions, you can throw and catch them in your methods. Custom exceptions enable you to specify detailed error messages and have more custom error handling in your catch blocks.

Exceptions can be top-level classes, that is, they can have member variables, methods and constructors, they can implement interfaces, and so on.

To create your custom exception class, extend the built-in Exception class and make sure your class name ends with the word Exception, such as “MyException” or “PurchaseException”. All exception classes extend the system-defined base class Exception, and therefore, inherits all common Exception methods.

This example defines a custom exception called MyException.

public class MyException extends Exception {}

38. Question

How can a developer set up a debug log on a specific user?

- It is not possible to setup debug logs for users other than yourself
- Ask the user for access to their account credentials, log in as the user and debug the issue
- Create Apex code that logs code actions into a custom object
- **Set up a trace flag for the user,and define a logging level and time period for the trace**

Unattempted

Correct:

✅ D. Set up a trace flag for the user, and define a logging level and time period for the trace. This is the correct way. Trace flags allow you to specify which user’s debug logs you want to capture, the level of detail (e.g., ERROR, WARNING, INFO, DEBUG, FINE, FINER, FINEST), and the duration for which the debug log should be active.

Incorrect:

❌ A. It is not possible to setup debug logs for users other than yourself. This is incorrect. Trace flags allow you to set up debug logs for *any* user in the org, provided you have the necessary permissions.

❌ B. Ask the user for access to their account credentials, log in as the user and debug the issue. While this might *seem* like a solution, it’s *strongly discouraged* for security reasons. Sharing credentials is a bad practice. Trace flags are the proper way to debug user-specific issues.

❌ C. Create Apex code that logs code actions into a custom object. While you *could* create custom logging, it’s not the standard or recommended approach for debug logs. Debug logs provide a built-in, centralized mechanism for capturing detailed information about code execution, and using trace flags is the most efficient way to capture user-specific debugging information. Custom logging should be reserved for very specific scenarios where standard debug logs are insufficient.

39. Question

What are three characteristics of static methods? Choose three

- [ ] **Initialized only when a class is loaded**
- [ ] A static variable outside of the scope of an Apex transaction
- [ ] **Allowed only in outer classes**
- [ ] Allowed only in inner classes
- [ ] **Excluded from the view state for a Visualforce page**

Unattempted
A static method or variable doesn't require an instance of the class in order to run.

Static methods, variables, or initialization code are associated with a class, and are only allowed in outer classes. When you declare a method or variable as static, it's initialized only once when a class is loaded. Static variables aren't transmitted as part of the view state for a Visualforce page.

40. Question

What are two uses for External IDs? Choose two

- [ ] **To create relationships between records imported from an external system.**
- [ ] To create a record in a development environment with the same Salesforce ID as in another environment
- [ ] To identify the sObject type in Salesforce
- [ ] **To prevent an import from creating duplicate records using Upsert**

Unattempted
Correct:

✅ A. To create relationships between records imported from an external system. External IDs are crucial for relating records when data is coming from an external system. Since the external system's IDs won't match Salesforce IDs, you use External IDs to establish those relationships during import.

✅ D. To prevent an import from creating duplicate records using Upsert. The `upsert` DML operation uses External IDs to determine whether a record already exists. If a matching External ID is found, the existing record is updated; otherwise, a new record is inserted. This prevents duplicates during data imports.

Incorrect:

❌ B. To create a record in a development environment with the same Salesforce ID as in another environment. Salesforce IDs are *automatically generated* by the platform and cannot be manually set. You cannot create a record with a specific Salesforce ID. External IDs are used for *cross-system* identification, not for replicating Salesforce IDs.

❌ C. To identify the sObject type in Salesforce. The sObject type is determined by the object's definition (e.g., Account, Contact, etc.), not by External IDs. External IDs are for identifying *specific records*, not object types.

41. Question

A developer wrote a unit test to confirm that a custom exception works properly in a custom controller, but the test failed due to an exception being thrown. Which step should the developer take to resolve the issue and properly test the exception?

- Use try/catch within the unit test to catch the exception
- Use the finally block within the unit test to populate the exception
- Use the database methods with all or none set to FALSE
- Use Test.isRunningTest() within the custom controller

Unattempted

Correct:

✅ A. Use try/catch within the unit test to catch the exception. This is the standard and correct way to test for expected exceptions in Apex unit tests. You should wrap the code that is *expected* to throw the exception in a `try` block. Then, in the `catch` block, you assert that the caught exception is of the correct type and/or contains the expected message. This verifies that the custom exception is working as designed.

Incorrect:

❌ B. Use the finally block within the unit test to populate the exception. The `finally` block is used for code that *must* execute, regardless of whether an exception is thrown or not. It's not the place to *handle* or *assert* exceptions. The `catch` block is what you would use to handle the exception, not the `finally` block.

❌ C. Use the database methods with all or none set to FALSE. Setting `allOrNone` to `false` is related to partial record processing in DML operations. It doesn't have anything to do with testing custom exceptions. It's irrelevant to the scenario.

❌ D. Use `Test.isRunningTest()` within the custom controller. `Test.isRunningTest()` is used to conditionally execute code *only* during testing. It's not directly related to exception handling. While you *might* use it in conjunction with exception testing (e.g., to set up test data differently), it's not the core mechanism for verifying that an exception is thrown. The `try/catch` block is what you use to test the exception itself.

Use Page numbers below to navigate to other practice tests

Pages: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Skillcertpro

Quick Links

ABOUT US
FAQ
BROWSE ALL PRACTICE TESTS
CONTACT FORM

Important Links

REFUND POLICY
REFUND REQUEST
TERMS & CONDITIONS
PRIVACY POLICY

Privacy Policy