# Salesforce Trigger and controllers Assignment

## Trigger Assignment

### Ques 1: How to create triggers? Explain with Example.

**Ans:** A trigger in Salesforce is a piece of Apex code that is executed before or after specific database events, such as insert, update, delete, or undelete, on a record. Triggers are created on objects and are stored as metadata.
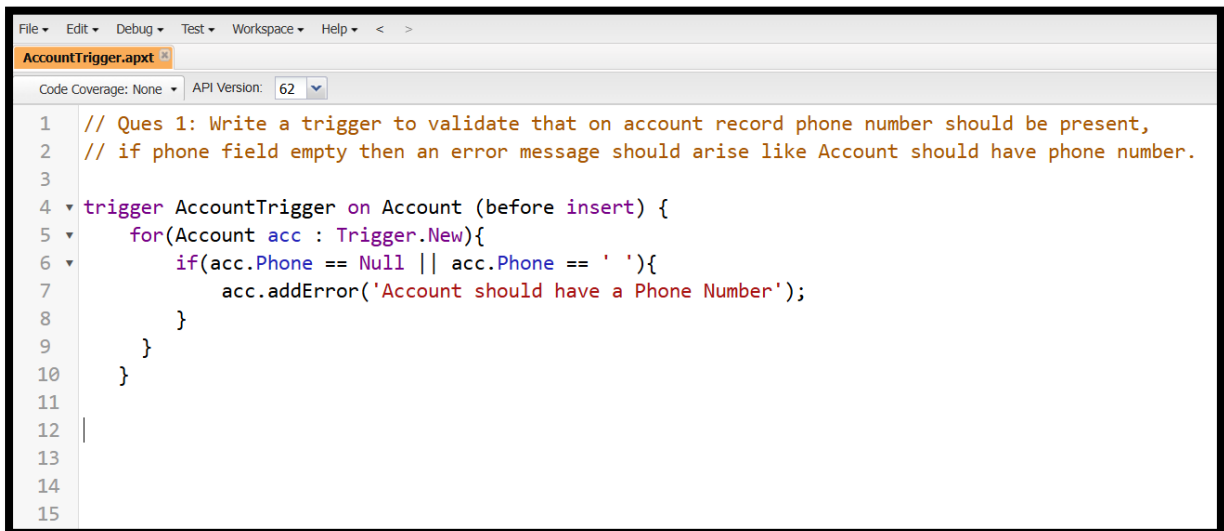
**Steps to Create a Trigger:**

1. Navigate to **Setup → Object Manager →** Choose the Object (e.g., Account).
2. Go to **Triggers** under the object settings.
3. Click **New** to create a trigger.

**Syntax:** trigger TriggerName on ObjectName (trigger_events) {

        // Trigger logic here

    }

**Example:**

```
File ▾  Edit ▾  Debug ▾  Test ▾  Workspace ▾  Help ▾  <   >
AccountTrigger.apxt ✕
Code Coverage: None ▾ | API Version:  62 ▾

1   // Ques 1: Write a trigger to validate that on account record phone number should be present,
2   // if phone field empty then an error message should arise like Account should have phone number.
3
4 ▾ trigger AccountTrigger on Account (before insert) {
5 ▾     for(Account acc : Trigger.New){
6 ▾         if(acc.Phone == Null || acc.Phone == ' '){
7               acc.addError('Account should have a Phone Number');
8           }
9       }
10    }
11
12 |
13
14
15
```

**Output:**



**Ques 2: Write a trigger example to insert contact when you create account.**

**Ans:  Trigger**



```
//The trigger creates a default contact whenever an account is created.


trigger AccountTrigger on Account (after insert){
    List<Contact> conList = new List<Contact>();
    for(Account acc : Trigger.New){
        Contact con = new Contact(FirstName = 'Default', Lastname = acc.Name, AccountId = acc.Id);
        conList.add(con);
        }
    if(!conList.isEmpty()){
        insert conList;
    }


}
```
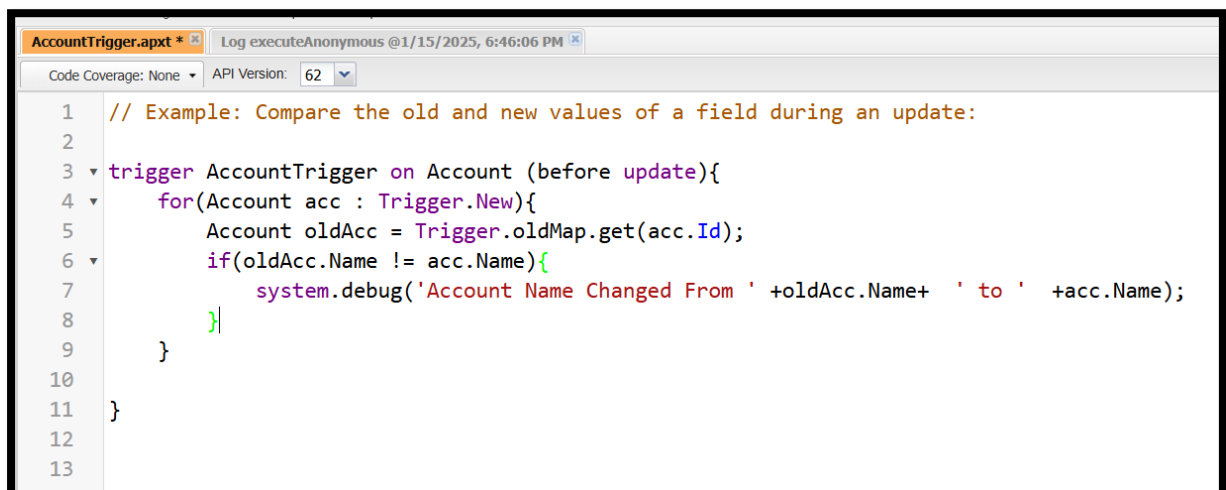
**Output:-**
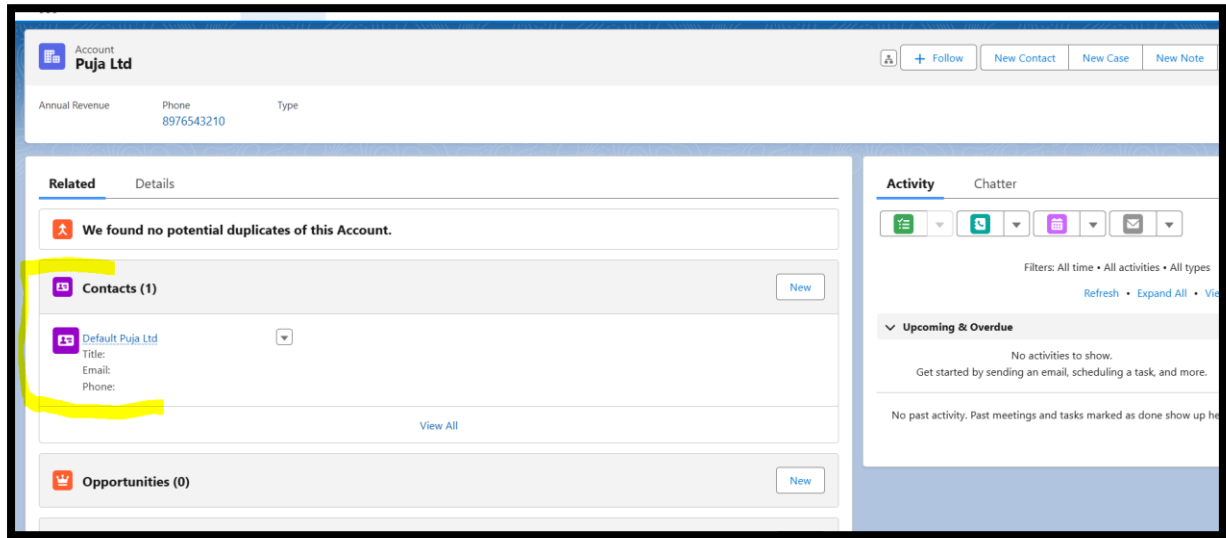


**Ques 3: Explain Trigger.oldmap with example.**

**Ans: `Trigger.oldMap`** is a map of IDs to the old versions of the records that were updated or deleted. It is available in the following contexts:

- **update (before and after)**
- **delete (before and after)**

**Example:** Compare the old and new values of a field during an update:



```
// Example: Compare the old and new values of a field during an update:

trigger AccountTrigger on Account (before update){
    for(Account acc : Trigger.New){
        Account oldAcc = Trigger.oldMap.get(acc.Id);
        if(oldAcc.Name != acc.Name){
            system.debug('Account Name Changed From ' +oldAcc.Name+ ' to ' +acc.Name);
        }
    }
}
```

## Output:



| Timestamp | Event | Details |
|---|---|---|
| 18:50:40:008 | USER_DEBUG | [23]|DEBUG|Account Name Changed From Puja Ltd to Nidhi Ltd |

**Ques 4: Write an example to create list in Apex Trigger.**

**Ans:**

```apex
//  To create list in Apex Trigger.

trigger AccountTrigger on Account (before update){
    List<Account> AccList = new List<Account>();
    for(Account acc : Trigger.New){
        acc.Name = acc.Name +  '-Updated';
        AccList.add(acc);
    }
    system.debug('Account List -->' +AccList);

}
```
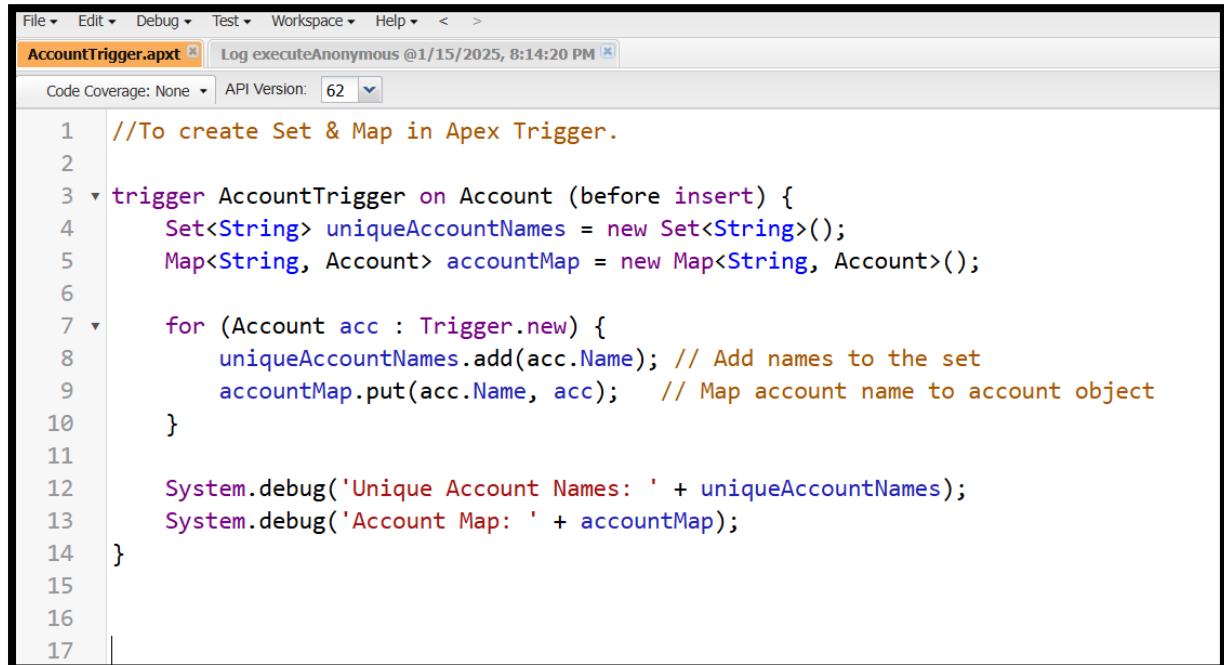
## Output:

| Timestamp | Event | Details |
|---|---|---|
| 19:15:18:012 | USER_DEBUG | [9]|DEBUG|Account List -->(Account:{Id=001WU00000dzor3YAA, IsDeleted=false, MasterRecordId=null, Name=Puja Test-Updated, Type=null, RecordTypeId=null, ParentId=null, BillingStreet=null, BillingCity=null, BillingState=null, BillingPostalCode=null, BillingCountry= |

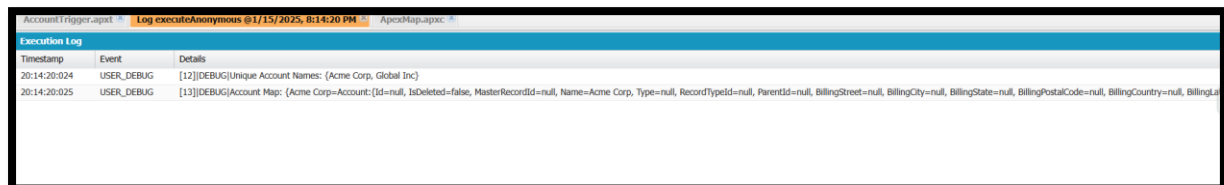| Timestamp | Event | Details |
|---|---|---|
| 19:13:46:007 | USER_DEBUG | [9]|DEBUG|Account List -->(Account:{Id=001WU00000dycNMYAY, IsDeleted=false, MasterRecordId=null, Name=Test Account 1-Updated, Type=null, RecordTypeId=null, ParentId=null, BillingStreet=null, BillingCity=null, BillingState=null, BillingPostalCode= |

**Ques 5: Write an example to create Set & Map in Apex Trigger.**

**Ans:** Following is the example to create set and map on Account Object

```
//To create Set & Map in Apex Trigger.

trigger AccountTrigger on Account (before insert) {
    Set<String> uniqueAccountNames = new Set<String>();
    Map<String, Account> accountMap = new Map<String, Account>();

    for (Account acc : Trigger.new) {
        uniqueAccountNames.add(acc.Name); // Add names to the set
        accountMap.put(acc.Name, acc);   // Map account name to account object
    }

    System.debug('Unique Account Names: ' + uniqueAccountNames);
    System.debug('Account Map: ' + accountMap);
}
```

**Output:**

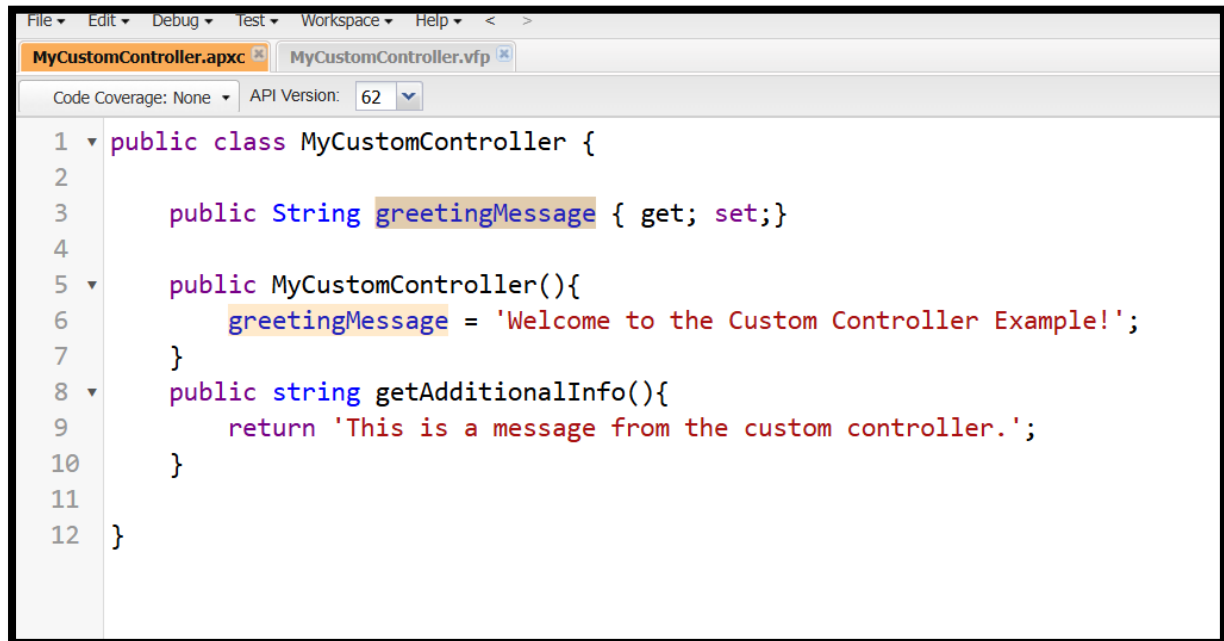# Controller Assignment

**Ques 1: Design VisualForce Page using Custom Controller Class.**
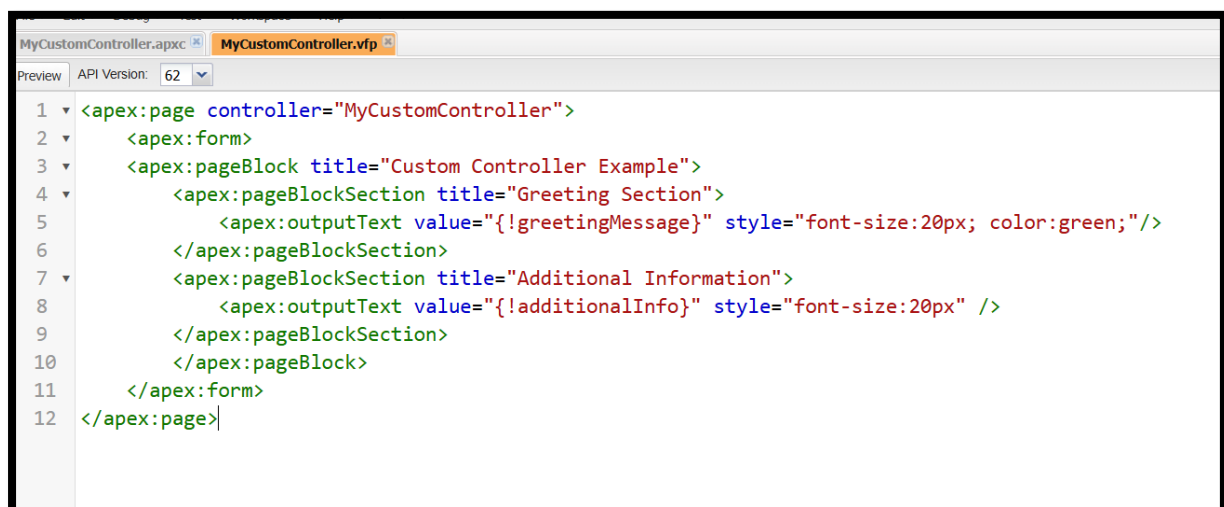
**Ans:**

**MyCustomController Apex Class:**
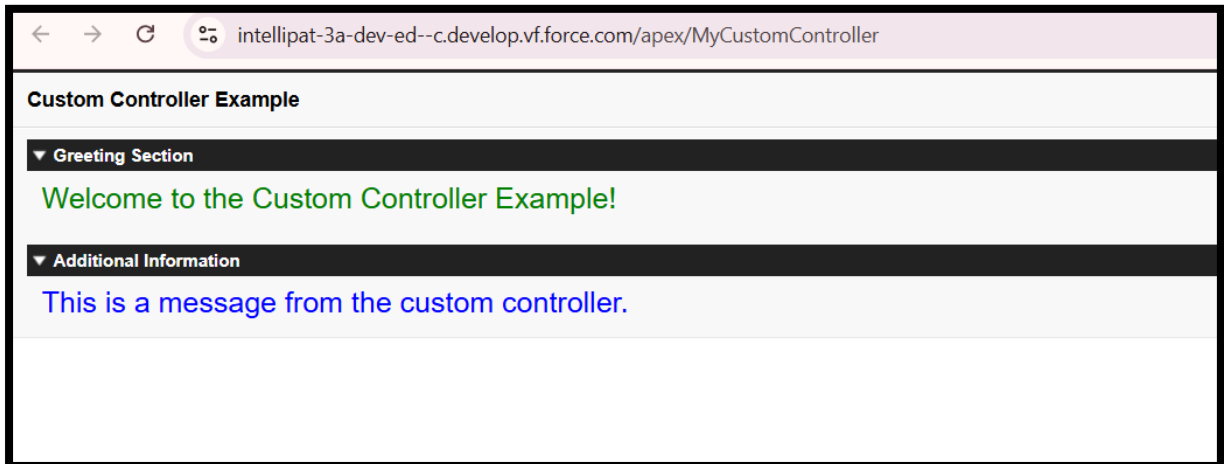
```
File ▾   Edit ▾   Debug ▾   Test ▾   Workspace ▾   Help ▾   <   >
MyCustomController.apxc ⊠   MyCustomController.vfp ⊠
Code Coverage: None ▾   API Version:  62 ▾

1 ▾ public class MyCustomController {
2
3       public String greetingMessage { get; set;}
4
5 ▾     public MyCustomController(){
6           greetingMessage = 'Welcome to the Custom Controller Example!';
7       }
8 ▾     public string getAdditionalInfo(){
9           return 'This is a message from the custom controller.';
10      }
11
12 }
```

## MyCustomController Visualforce Page:

```
MyCustomController.apxc ⊠   MyCustomController.vfp ⊠
Preview   API Version:  62 ▾

1 ▾ <apex:page controller="MyCustomController">
2 ▾     <apex:form>
3 ▾     <apex:pageBlock title="Custom Controller Example">
4 ▾         <apex:pageBlockSection title="Greeting Section">
5               <apex:outputText value="{!greetingMessage}" style="font-size:20px; color:green;"/>
6           </apex:pageBlockSection>
7 ▾         <apex:pageBlockSection title="Additional Information">
8               <apex:outputText value="{!additionalInfo}" style="font-size:20px" />
9           </apex:pageBlockSection>
10          </apex:pageBlock>
11      </apex:form>
12 </apex:page>
```

**Output:**



**Ques 2:** Design VisualForce Page using Custom Controller Extensions Class ?

**Ans:**

**MyCustomController Apex Class**



```apex
public class MyCustomControllerExtension {
    private ApexPages.StandardController standardController;
    public string customMessage {get; set;}
    public MyCustomControllerExtension(ApexPages.StandardController controller){
        this.standardController = controller;
        customMessage = 'Hello from the Custom Controller Extension!';
    }
    public string getExtendedInfo(){
        return 'This is additional functionality provided by the extension.';
    }

}
```

**MyCustomController Visualforce Page:**

```
MyCustomControllerExtension.apxc ⊠    MyCustomControllerExtension.vfp ⊠
Preview   API Version: 62 ▼

 1 ▼  <apex:page standardController="Account" extensions="MyCustomControllerExtension">
 2 ▼      <apex:form>
 3 ▼      <apex:pageBlock title="Custom Controller Extension Example">
 4 ▼          <apex:pageBlockSection title="Standard Controller Data">
 5               <apex:outputLabel value="Account Name: Puja" />
 6               <apex:outputLabel value="{!Account.Name}" />
 7           </apex:pageBlockSection>
 8 ▼          <apex:pageBlockSection title="Extension Data">
 9               <apex:outputLabel value="{!customMessage}" style="font-size:20px; color:green;" />
10               <br />
11               <apex:outputText value="{!extendedInfo}" style="font-size:16px;" />
12           </apex:pageBlockSection>
13           </apex:pageBlock>
14       </apex:form>
15  </apex:page>
```
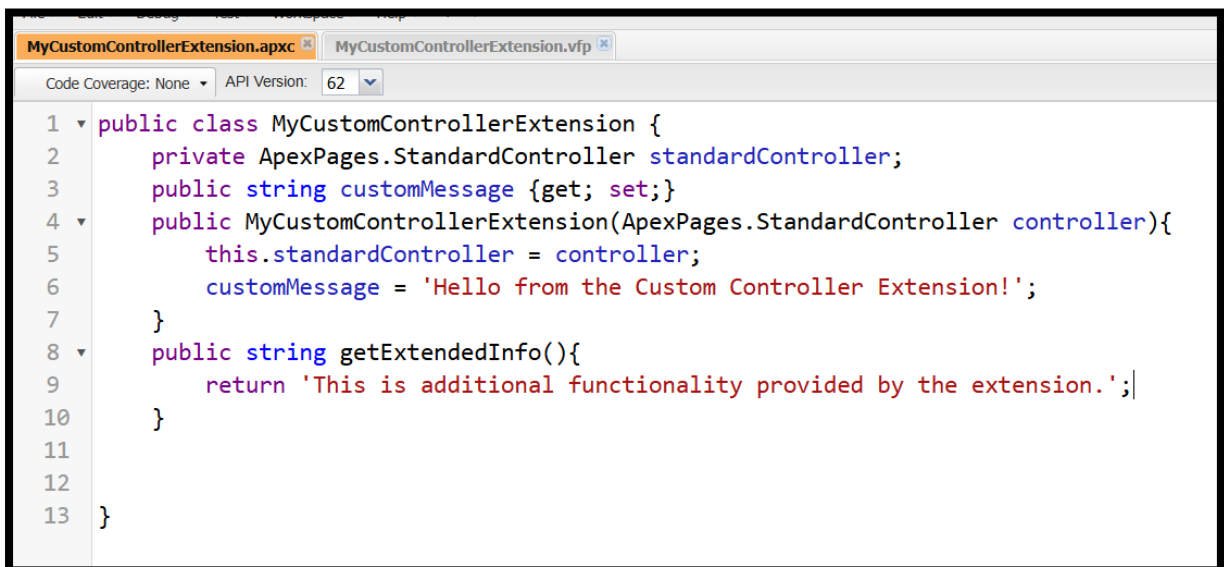
**Output:**

**Custom Controller Extension Example**

**▼ Standard Controller Data**

Account Name: Puja

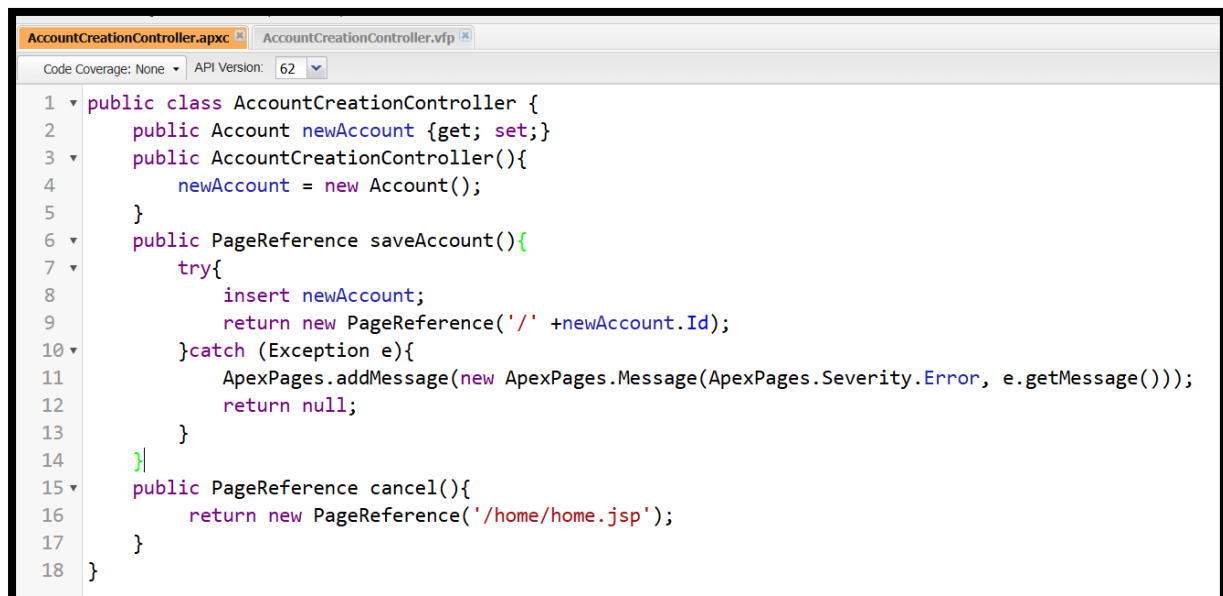**▼ Extension Data**

Hello from the Custom Controller Extension!

This is additional functionality provided by the extension.

**Ques 3:** Write an example to create new records using custom controller.

**Ans:**

**AccountCreationController Apex Class**

```
AccountCreationController.apxc    AccountCreationController.vfp
Code Coverage: None    API Version: 62

1   public class AccountCreationController {
2       public Account newAccount {get; set;}
3       public AccountCreationController(){
4           newAccount = new Account();
5       }
6       public PageReference saveAccount(){
7           try{
8               insert newAccount;
9               return new PageReference('/' +newAccount.Id);
10          }catch (Exception e){
11              ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.Error, e.getMessage()));
12              return null;
13          }
14      }
15      public PageReference cancel(){
16          return new PageReference('/home/home.jsp');
17      }
18  }
```

**AccountCreationController VisualForce Page**

```
AccountCreationController.apxc    AccountCreationController.vfp
Preview    API Version: 62

1   <apex:page controller="AccountCreationController" >
2       <apex:form>
3       <apex:pageBlock title="Create New Account">
4           <apex:pageMessages />
5           <apex:pageBlockSection title="Account Details">
6               <apex:inputField value="{!newAccount.Name}" label = "Account Name"  />
7               <apex:inputField value="{!newAccount.Phone}" label="Phone Number" />
8               <apex:inputField value="{!newAccount.Active__c}" label="Active" />
9               <apex:inputField value="{!newAccount.Industry}" label="Industry" />
10              <apex:inputField value="{!newAccount.BillingCity}" label="Billing City" />
11          </apex:pageBlockSection>
12          <apex:pageBlockButtons>
13          <apex:commandButton value="Save" action="{!SaveAccount}" />
14              <apex:commandButton value="cancel" action="{!cancel}" immediate="true" />
15          </apex:pageBlockButtons>
16          </apex:pageBlock>
17      </apex:form>
18  </apex:page>
```

**Output:**

**Ques 4:** Write an example to manage list of records by using standard controller.

**Ans:**

**AccountStandardController VisualForce Paage**

```
AccountStandardController.vfp
Preview   API Version: 62

1  ▼ <apex:page standardController="Account" recordSetVar="accounts">
2  ▼     <apex:form>
3  ▼     <apex:pageBlock title="Manage Account Records">
4  ▼         <apex:pageBlockTable value="{!accounts}" var="acc">
5  ▼         <apex:column headerValue="Account Name">
6              <apex:outputLink value="/{!acc.Id}"> {!acc.Name} </apex:outputLink>
7          </apex:column>
8  ▼         <apex:column headerValue="Phone Number">
9              <apex:outputText value="{!acc.Phone}" />
10         </apex:column>
11 ▼        <apex:column headerValue="Industry">
12             <apex:outputText value="{!acc.Industry}" />
13         </apex:column>
14       </apex:pageBlockTable>
15 ▼     <apex:pageBlockButtons>
16     <apex:commandButton value="Save Changes" action="{!save}" />
17             <apex:commandButton value="Cancel" action="{!Cancel}" immediate="true" />
18     </apex:pageBlockButtons>
19     </apex:pageBlock>
20     </apex:form>
21 </apex:page>
```
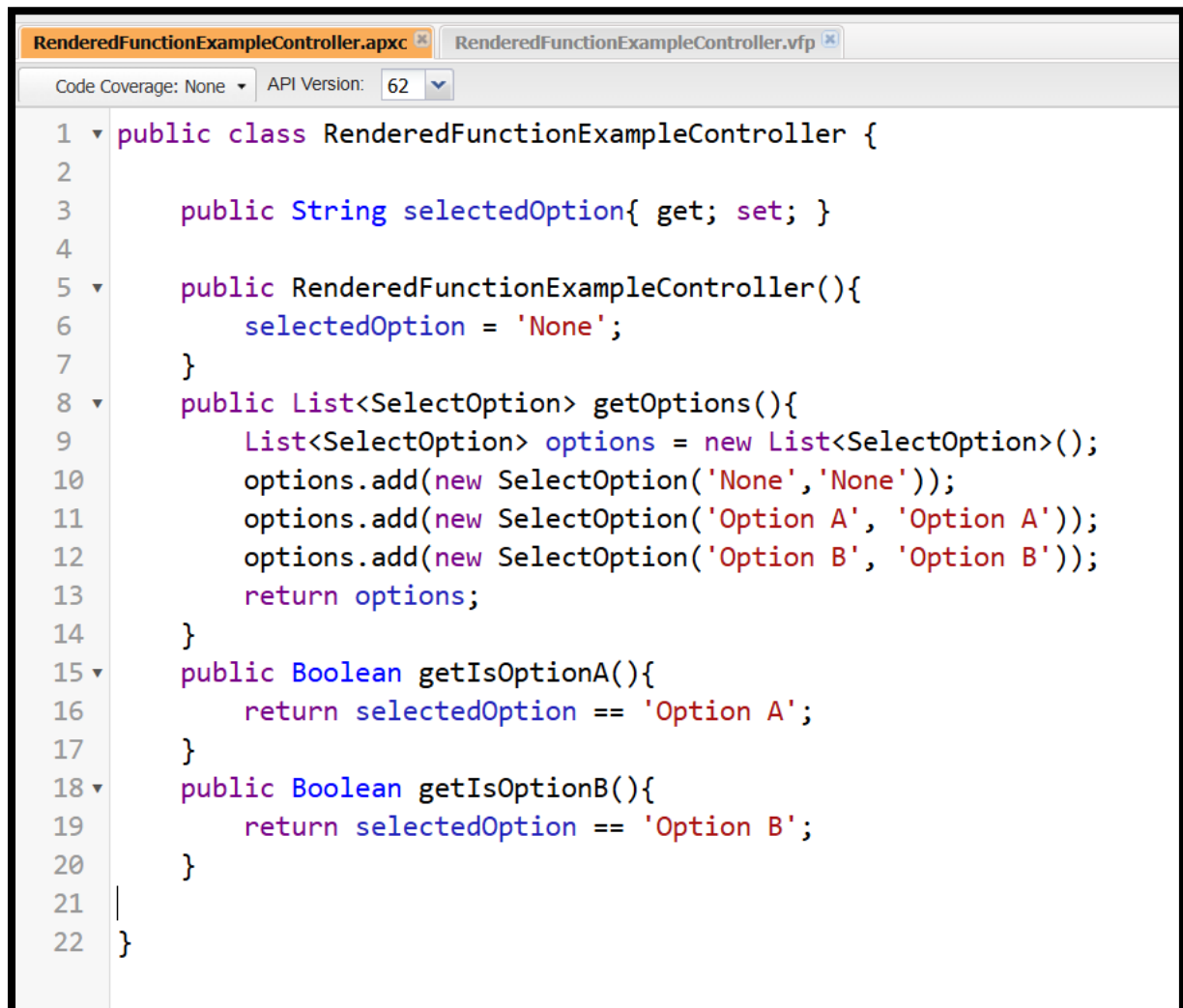
# Output:

| Manage Account Records | Save Changes | Cancel | |
|---|---|---|---|
| **Account Name** | **Phone Number** | | **Industry** |
| ACME corp | | | |
| Acme Corp | | | |
| Acme Corp | | | |
| Global Inc | | | |
| Global Inc | | | |
| Nidhi | | | |
| nidhi | 1234567890 | | |
| Puja LTD | 1234567890 | | Agriculture |
| Puja Pvt | 2345678901 | | Communications |
| Puja Test-Updated | | | |
| Puvi Ltd | | | |
| Test Account 1-Updated | | | |
| Test Trigger | | | |
| Test Trigger 1 | | | |
| Vikash Ltd | | | |
| | Save Changes | Cancel | |

## Ques 5: Explain Rendered Function with example.

**Ans:** The `rendered` attribute in Visualforce determines whether a specific component is displayed on the page. The attribute accepts a Boolean expression, and the component is only displayed if the expression evaluates to `true`.

**Apex Controller Code**

```
RenderedFunctionExampleController.apxc    RenderedFunctionExampleController.vfp
Code Coverage: None ▾   API Version:  62  ▾

1  ▾ public class RenderedFunctionExampleController {
2
3        public String selectedOption{ get; set; }
4
5  ▾     public RenderedFunctionExampleController(){
6            selectedOption = 'None';
7        }
8  ▾     public List<SelectOption> getOptions(){
9            List<SelectOption> options = new List<SelectOption>();
10           options.add(new SelectOption('None','None'));
11           options.add(new SelectOption('Option A', 'Option A'));
12           options.add(new SelectOption('Option B', 'Option B'));
13           return options;
14       }
15 ▾     public Boolean getIsOptionA(){
16           return selectedOption == 'Option A';
17       }
18 ▾     public Boolean getIsOptionB(){
19           return selectedOption == 'Option B';
20       }
21 |
22 }
```

**Visualforce Page Code**

```
RenderedFunctionExampleController.apxc    RenderedFunctionExampleController.vfp
Preview  API Version: 62
 1  <apex:page controller="RenderedFunctionExampleController">
 2      <apex:form>
 3          <apex:pageBlock title="Rendered Function Example">
 4              <apex:pageBlockSection title="Select an Option">
 5                  <apex:selectList value="{!selectedOption}" size="1">
 6                      <apex:selectOptions value="{!options}" />
 7                      <apex:actionSupport event="onchange" reRender="formSection" />
 8                  </apex:selectList>
 9              </apex:pageBlockSection>
10              <apex:pageBlockSection id="formSection" title="Dynamic Form Section" >
11                  <apex:outputPanel rendered="{!isOptionA}">
12                      <apex:outputText value="You selected Option A. Displaying Option A's form." style="color:green; font-size:18px" />
13                  </apex:outputPanel>
14                  <apex:outputPanel rendered="{!isOptionB}">
15                      <apex:outputText value="You selected Option B. Displaying Option B's form." style="color:blue;font-size:18px" />
16                  </apex:outputPanel>
17                  <apex:outputPanel rendered="{!selectedOption == 'None'}">
18                      <apex:outputText value="Please select an option to display the form." style="color:gray;font-size:18px"/>
19                  </apex:outputPanel>
20              </apex:pageBlockSection>
21          </apex:pageBlock>
22      </apex:form>
23  </apex:page>
```

**Outputs**

**For None:**

**For Option A:**

**Rendered Function Example**

▼ Select an Option

Option A ⌄

▼ Dynamic Form Section

You selected Option A. Displaying Option A's form.

**For Option B:**

**Rendered Function Example**

▼ Select an Option

Option B ⌄

▼ Dynamic Form Section

You selected Option B. Displaying Option B's form.