**Visualforce And Integration**

**1.Create & Edit Visualforce Pages**

**Q. Create a Visualforce page without the standard Salesforce header and display an image using the Visualforce image component.**

**Output :**

<apex:page showHeader="false">

<apex:image url="https://developer.salesforce.com/files/salesforce-developer-network-logo.png">

</apex:image>

</apex:page>

**2.Use Simple Variables and Formulas**

**Q. Create a Visualforce page that displays the first name of the currently logged-in user.**

**Output:**

<apex:page >

<apex:pageBlock >

{!$User.FirstName}

</apex:pageBlock>

</apex:page>

**3.Use Standard Controllers**

**Q. Using the Contact standard controller, create a Visualforce page that displays a Contact's First Name, Last Name and the Email address of the Contact's Owner.**

**Output:**

<apex:page standardController="Contact">

<apex:pageBlockSection >

First Name:{!Contact.FirstName}

Last Name:{!Contact.LastName}

Owner Email:{!Contact.Owner.Email}

</apex:pageBlockSection>

</apex:page>

**4. Display Records, Fields, and Tables**

**Q. Create a page that displays a subset of Opportunity fields using apex:outputField components. Bind the Name, Amount, Close Date and Account Name fields to the apex:outputField components.**

**Output :**

<apex:page standardController="Opportunity">

<apex:outputField value="{!Opportunity.Name}"/>

<apex:outputField value="{!Opportunity.Amount}"/>

<apex:outputField value="{!Opportunity.CloseDate}"/>

<apex:outputField value="{!Opportunity.Account.Name}"/>

</apex:page>

**5. Input Data Using Forms**

**Q. Using the Visualforce apex:form component, create a page that inserts a Contact record based on First Name, Last Name and Email. After submitting the form, the user should be redirected to the detail page of the new Contact record.**

**Output :**

<apex:page standardController="Contact">

<apex:form >

<apex:pageBlockSection >

<apex:inputField value="{!Contact.FirstName}"/>

<apex:inputField value="{!Contact.LastName}"/>

<apex:inputField value="{!Contact.Email}"/>

</apex:pageBlockSection>

<apex:commandButton action="{! save}" value="Save"/>

</apex:form>

</apex:page>

**6.Use Standard List Controllers**

**Q. Using a Standard List Controller, create a Visualforce that displays a list of Accounts with links to their respective record detail pages.**

**Output:**

<apex:page standardController="Account" recordSetVar="accounts">

<apex:repeat var="a" value="{!accounts}">

<li>

<apex:outputlink value="/{!a.ID}">

<apex:outputText value="{! a.Name}">

</apex:outputText>

</apex:outputlink>

</li>

</apex:repeat>

</apex:page>

**7.Use Static Resources**

**Q. Upload the specified zip file as a static resource and display an image from the file on a Visualforce page.**

**Output :**

<apex:page >

<apex:image url="{! URLFOR($Resource.vfimagetest,'cats/kitten1.jpg')}"/>

</apex:page>

**8. Create & Use Custom Controllers**

**Q. Create a Visualforce page that uses a custom controller to display a list of cases with the status of New.**

**Output :**

<apex:page controller="NewCaseListController">

<apex:repeat var="case" value="{!NewCases}">

<li>

<apex:outputLink value="/{!case.id}">{!case.id}</apex:outputLink>

{!case.CaseNumber}

</li>

</apex:repeat>

</apex:page>

public class NewCaseListController {

list<case> newcase = new list<case>();

public list<case> GetNewCases()

{

newcase = [Select Id,CaseNumber from case where status='New'];

return newcase;

}

}