GENERAL

salesforce visualforce page developer guide

Published: September 3, 2026 | Index ID: #-

Salesforce Visualforce Page Developer Guide

In the world of Salesforce development, creating custom user interfaces that blend seamlessly with business processes is essential. While Lightning Web Components (LWC) have become the modern standard for new projects, Visualforce remains a powerful tool for legacy systems, complex page layouts, and quick prototyping. This comprehensive guide walks you through the fundamentals and advanced techniques needed to become a proficient Visualforce page developer. Whether you're a seasoned Apex coder or a newcomer to the Salesforce ecosystem, this resource will equip you with the knowledge to design, implement, and maintain robust Visualforce pages.

Why Visualforce Still Matters

Visualforce, introduced in 2010, provides a declarative markup language that mirrors HTML but is tightly integrated with Salesforce's data model. Its strengths include:

- Rapid Development: Drag and drop components and pre built tags reduce coding time.
- Fine grained Control: You can override standard page layouts or embed complex logic.
- Legacy Compatibility: Many enterprises still rely on Visualforce for critical applications.
- Seamless Apex Integration: Controllers written in Apex provide powerful data manipulation.

Because of these benefits, many organizations still maintain or extend Visualforce pages. Understanding how to build and troubleshoot them is a valuable skill for any Salesforce developer.

Getting Started: Prerequisites

Before diving into Visualforce syntax, make sure you have the following:

1. Salesforce Developer Edition: Free access to a sandbox environment.
2. Developer Console or VS Code with Salesforce Extensions: Tools for writing and testing code.
3. Basic knowledge of Apex and Visualforce.
4. Understanding of Salesforce Objects and the data model.

With these in place, you're ready to create your first Visualforce page.

Visualforce Page Structure

A typical Visualforce page consists of three core elements:

- Page Tag: The root element that defines the page.
- Standard Components: Built in tags like <apex:form>, <apex:inputText>, and <apex:commandButton>.
- Custom Components: Reusable components created with <apex:component>.

Below is a minimal example:

<apex:page controller="AccountController">

```
  &lt;apex:form&gt;
    &lt;apex:pageBlock title="Account Details"&gt;
      &lt;apex:pageBlockSection title="Basic Info" columns="2"&gt;
        &lt;apex:inputField value="{!account.Name}"/&gt;
        &lt;apex:inputField value="{!account.Industry}"/&gt;
      &lt;/apex:pageBlockSection&gt;
      &lt;apex:pageBlockButtons&gt;
        &lt;apex:commandButton action="{!save}" value="Save"/&gt;
      &lt;/apex:pageBlockButtons&gt;
    &lt;/apex:pageBlock&gt;
  &lt;/apex:form&gt;
&lt;/apex:page&gt;
```

The controller attribute links the page to an Apex class that supplies data and handles actions.

Controllers: The Backbone of Logic

Controllers in Visualforce are Apex classes that expose properties and methods to the page. They can be:

- Standard Controllers: Automatically provide CRUD operations for a single object.
- Standard Set Controllers: Manage collections of records.
- Custom Controllers: Full control over logic, suited for complex scenarios.
- Controller Extensions: Extend a standard controller with additional functionality.

Custom Controller Example

Here’s a simple controller that retrieves an account and updates it:

```
public class AccountController {
  public Account account { get; set; }

  public AccountController() {
    account = [SELECT Id, Name, Industry FROM Account LIMIT 1];
  }

  public PageReference save() {
    try {
      update account;
      ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.CONFIRM, 'Account saved.'));
    } catch (Exception e) {
      ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.ERROR, e.getMessage()));
    }
    return null;
  }
}
```

Notice the use of ApexPages.addMessage to provide user feedback. Visualforce pages automatically render these messages.

Controller Extensions

When you need to add features to an existing standard controller, extensions are the way to go:

```
public class AccountExtension {
  private final Account acct;

  public AccountExtension(ApexPages.StandardController stdController) {
    acct = (Account)stdController.getRecord();
  }

  public void closeAccount() {
    acct.Status__c = "Closed";
    update acct;
    ApexPages.addMessage(new ApexPages.Message(ApexPages.Severity.CONFIRM, "Account closed."));
  }
}
```

In the Visualforce page, reference the extension via extensions attribute.

Data Binding and Expression Language

Visualforce uses Expression Language (EL) to bind data between the page and the controller. Key features include:

- Property Binding: {!account.Name} pulls the value from the controller.
- Method Invocation: {!save} calls a method.
- Conditional Rendering: <apex:outputPanel rendered="{!isAdmin}"> displays content only when isAdmin is true.
- Iteration: <apex:repeat> loops over collections.

Example: Rendering a List of Contacts

```
&lt;apex:page controller="ContactListController"&gt;
  &lt;apex:pageBlock title="Contact List"&gt;
    &lt;apex:pageBlockTable value="{!contacts}" var="c"&gt;
      &lt;apex:column headerValue="Name"&gt;{!c.Name}&lt;/apex:column&gt;
      &lt;apex:column headerValue="Email"&gt;{!c.Email}&lt;/apex:column&gt;
    &lt;/apex:pageBlockTable&gt;
  &lt;/apex:pageBlock&gt;
&lt;/apex:page&gt;
```

The controller would expose a contacts property returning a list of Contact records.

Styling Visualforce Pages

Visualforce supports standard CSS and inline styles. Additionally, you can reference external stylesheets, including Salesforce's Lightning Design System (SLDS), to give Visualforce pages a modern look.

Using SLDS with Visualforce

1. Include the SLDS CDN link in the <apex:page> tag: style="{!URLFOR($Resource.SLDS, "assets/styles/

salesforce-lightning-design-system.min.css")}".

2. Apply SLDS classes to components: class="slds-button slds-button--neutral".
3. Use SLDS utility classes for spacing, typography, and layout.

By leveraging SLDS, your Visualforce pages will maintain visual consistency with Lightning pages, improving user experience.

Reusable Components

Custom components encapsulate markup and logic, making them reusable across pages. They are defined with <apex:component> and can accept parameters.

Defining a Component

```
&lt;apex:component&gt;
  &lt;apex:param name="title" type="String" description="Title of the panel" /&gt;
  &lt;apex:param name="content" type="String" description="Content to display" /&gt;

  &lt;div class="slds-card"&gt;
    &lt;header class="slds-card__header slds-media"&gt;
      &lt;div class="slds-media__body"&gt;
        &lt;h2 class="slds-card__header-title"&gt;{!title}&lt;/h2&gt;
      &lt;/div&gt;
    &lt;/header&gt;
    &lt;div class="slds-card__body slds-card__body_inner"&gt;{!content}&lt;/div&gt;
  &lt;/div&gt;
&lt;/apex:component&gt;
```

Using the Component

```
&lt;apex:page&gt;
  &lt;apex:include component="/components/MyCard" title="Welcome" content="Hello, Salesforce!" /&gt;
&lt;/apex:page&gt;
```

Components can also contain their own controllers, enabling complex behavior without cluttering the parent page.

Integration with Other Salesforce Features

Visualforce pages can be embedded in various contexts:

- Lightning App Builder: Add a Visualforce component to Lightning pages.
- Community Builder: Use Visualforce to create custom pages in Experience Cloud.
- Mobile App Builder: Render Visualforce pages in Salesforce mobile.
- Redirects and Actions: Use PageReference to navigate between pages.

Embedding a Visualforce Page in Lightning

In Lightning App Builder, drag the Visualforce Page component onto the layout. Select the page from the dropdown, and it will render within the Lightning page. This integration is particularly useful for legacy UIs that need to coexist with Lightning components.

Baca Juga (Artikel Terkait)

- **automaker mini software version v8 erptoboss**
  URL: http://art-consciousness.com/automaker-mini-software-version-v8-erptoboss.pdf
- **automata and computability**
  URL: http://art-consciousness.com/automata-and-computability.pdf
- **automata and languages theory and applications**
  URL: http://art-consciousness.com/automata-and-languages-theory-and-applications.pdf
- **automata and mechanical toys**
  URL: http://art-consciousness.com/automata-and-mechanical-toys.pdf

Tags: #salesforce #visualforce #page #developer #guide