**Visualforce Page Documentation for Case Management**

This Visualforce page is designed for Salesforce to manage **Case Management** by providing a user-friendly interface for interacting with **Case** records. The page uses a custom extension (CaseController) to fetch data dynamically and integrates both visual styles and Salesforce functionality to enhance the user experience.

Here is a breakdown of the page's components and functionality:

**1. Page Definition**

<apex:page standardController="Case" extensions="CaseController" showHeader="false" sidebar="false">

* **standardController="Case"**: This specifies that the page is using the standard Salesforce Case object. It automatically provides functionality to display and edit Case records.
* **extensions="CaseController"**: A custom Apex controller extension is applied here. It allows for custom logic (for example, querying related records such as contacts) to be executed during the page lifecycle.
* **showHeader="false"**: Hides the default Salesforce header (navigation bar) on this page.
* **sidebar="false"**: Hides the default sidebar in the Salesforce interface for a clean, full-screen view.

**2. Styling Section**

<style>

body {

background: linear-gradient(to bottom, #87CEEB, #ffffff);

font-family: Arial, sans-serif;

color: #333;

}

.header {

background-color: #87CEEB;

color: white;

padding: 20px;

text-align: center;

font-size: 24px;

font-weight: bold;

}

.container {

background-color: white;

border-radius: 10px;

box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);

margin: 20px auto;

padding: 30px;

max-width: 800px;

}

.form-section {

margin-bottom: 20px;

}

.form-label {

font-weight: bold;

color: #333;

}

.form-input {

width: 100%;

padding: 10px;

margin-top: 5px;

border: 1px solid #87CEEB;

border-radius: 5px;

}

.buttons {

text-align: right;

}

.btn {

background-color: #87CEEB;

color: white;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

margin-left: 10px;

}

.btn:hover {

background-color: #66B2FF;

}

.contact-info {

background-color: #F0F8FF;

border-radius: 5px;

padding: 15px;

margin-top: 20px;

border: 1px solid #87CEEB;

}

.contact-info h3 {

color: #87CEEB;

margin-top: 0;

}

</style>

* **Background Gradient**: The body tag has a gradient from sky blue (#87CEEB) to white, giving a smooth transition effect to the page.
* **Font and Colors**: The page uses the Arial font for simplicity, and a color scheme of sky blue and white is applied throughout to match the branding of the case management system.
* **Container Styling**: The main content area is styled as a container with rounded corners and a soft shadow for a clean, modern look.
* **Button Styling**: The buttons are styled with a blue background, white text, and rounded corners, providing a consistent, easy-to-press feel.

**3. Main Content Structure**

<div class="header">Case Management</div>

<div class="container">

<apex:form>

<!-- Form Elements Here -->

</apex:form>

</div>

* **header div**: Displays the title "Case Management" in the header, styled with a sky blue background.
* **container div**: Wraps the form fields for a well-contained and visually clean layout.

**4. Form Elements**

Each form field is created inside an apex:form component, which allows users to interact with Salesforce data. The fields are displayed as input elements where users can modify the case record.

<div class="form-section">

<label class="form-label" for="subject">Subject</label>

<apex:inputText id="subject" value="{!Case.Subject}" styleClass="form-input"/>

</div>

<div class="form-section">

<label class="form-label" for="description">Description</label>

<apex:inputTextarea id="description" value="{!Case.Description}" styleClass="form-input"/>

</div>

<div class="form-section">

<label class="form-label" for="priority">Priority</label>

<apex:inputField value="{!Case.Priority}" styleClass="form-input"/>

</div>

<div class="form-section">

<label class="form-label" for="status">Status</label>

<apex:inputField value="{!Case.Status}" styleClass="form-input"/>

</div>

<div class="form-section">

<label class="form-label" for="contact">Contact</label>

<apex:inputField value="{!Case.ContactId}" styleClass="form-input" required="true"/>

</div>

* **Subject, Description, Priority, and Status Fields**: These fields are rendered using apex:inputText, apex:inputTextarea, and apex:inputField components. They allow users to modify the Case's subject, description, priority, and status.
* **Contact Field**: The contact field uses apex:inputField to allow users to select a contact. This field is required, ensuring that a contact is always associated with the case.

**5. Dynamic Contact Information Display**

<apex:actionRegion>

<apex:outputPanel id="contactInfoPanel" styleClass="contact-info">

<apex:outputText rendered="{!selectedContact != null}">

<h3>Contact Information</h3>

<p><strong>Name:</strong> {!selectedContact.Name}</p>

<p><strong>Email:</strong> {!selectedContact.Email}</p>

<p><strong>Phone:</strong> {!selectedContact.Phone}</p>

<p><strong>Open Cases:</strong> {!openCases}</p>

</apex:outputText>

</apex:outputPanel>

</apex:actionRegion>

* **apex:actionRegion**: This component wraps the contact information output and allows the page to dynamically update without reloading.
* **apex:outputPanel**: Contains the contact info. It's styled with a light blue background and rounded corners to differentiate it from other sections.
* **Dynamic Content**: Using the rendered attribute, the contact information (name, email, phone, and open cases) is displayed only if a contact is selected. The variables {!selectedContact.Name}, {!selectedContact.Email}, {!selectedContact.Phone}, and {!openCases} are fetched dynamically from the Apex controller.

**6. Submit and Reset Buttons**

<div class="buttons">

<apex:commandButton value="Submit" action="{!save}" styleClass="btn"/>

<apex:commandButton value="Reset" action="{!reset}" styleClass="btn"/>

</div>

* **Submit Button**: Calls the save method in the controller to save the case record. The button is styled using the btn class.
* **Reset Button**: Calls the reset method in the controller to reset the form fields to their default values.

**7. Apex Controller Logic (CaseController)**

While this documentation doesn't include the Apex code directly, the **CaseController** extension is crucial for:

* Handling dynamic content for selected contacts.
* Implementing logic to fetch and display the open cases related to the selected contact.

**Conclusion:**

This Visualforce page is a fully styled and functional interface that integrates seamlessly with Salesforce's Case management system. It offers users the ability to update case details, view associated contact information, and submit or reset the form, all within an aesthetically pleasing and responsive layout.