

Force.com Visualforce Pages

salesforce

global strategic
consulting partner

Lesson 01 : Introduction to Visualforce



Application Building Blocks

Applications
Tabs
Page Layouts
Record Types



User
Interface

Force.com Pages
Web controls
Sites



Workflow
Validation Rules
Approval Processes



Business
Logic

Force.com Page Controllers
Force.com code
Web Services API



Objects
Fields
Relationships



Data
Model

Web Services API
Metadata API



Declarative

Programmatic

Simplicity + Speed

Control + Flexibility

By the end of this lesson, you will be able to:

- Describe the Visualforce framework, including its advantages and capabilities and how it fits into the Force.com platform
- Apply the model-view-controller pattern to Salesforce and understand its advantages
- Identify the different parts of the Visualforce framework and when to use each
- Describe how to incorporate Visualforce pages into Salesforce

Visualforce Overview

Create Any Application and Interface for Any Device

visualforce™

Any application
Any user interface
Any device



Two types of Force.com User Interface Technologies

Page Builder

- UI generated automatically, no technical skills required
- Limited/no control of UI behavior
- Limited control over look and feel, but all UIs are consistent

Visualforce

- UI generated by developer/technologist
- Full control of UI behavior
- Full “pixel level” control over UI

Visualforce Overview

Visualforce

Visualforce was officially released as part of the Summer '08 release. It allows developers to completely customize the standard page layouts within the Salesforce UI with completely custom pages. It uses Apex to incorporate advanced business logic functionality.

Visualforce Overview

Visualforce Inline Editor

Visualforce offers a user-friendly, split-screen development environment. Simply click Save to immediately test your code! The inline editor also includes auto-completion and limited syntax highlighting, as well as online documentation. Access to the inline editor is available only in Development Mode. Visualforce pages can also be edited through the Force.com IDE.

Visualforce Pages are ultimately rendered into markup.

This means that developers can include:

- Visualforce tags
- Force.com expressions
- HTML
- JavaScript
- Flash
- Any other code that can execute within an HTML page on Force.com

Visualforce pages can have up to 1 MB of contents and can bring back up to 15 MB.

Visualforce Overview

Visualforce Component Rendering

Each Visualforce component tag generates markup or other Web-enabled code behind the scenes to create the page.

The simplest example of this is the image component:

- This: `<apex:image id="theImage" value="/img/myimage.gif" width="220" height="55"/>`
- Becomes this: `<image id="theImage" src="/img/myimage.gif" width="220" height="55"/>`

In most other examples, the component generates much more code and simplifies development.

To view the resulting code from any Visualforce page, right-click the page and select View Source.

Important when maintaining state across multiple pages or server calls
View state inspector shows components contributing to view state

- Must be enabled on user profile
- Only displayed when using `<apex:form>`

view state: space in server memory for data necessary to maintain the state of the database between requests

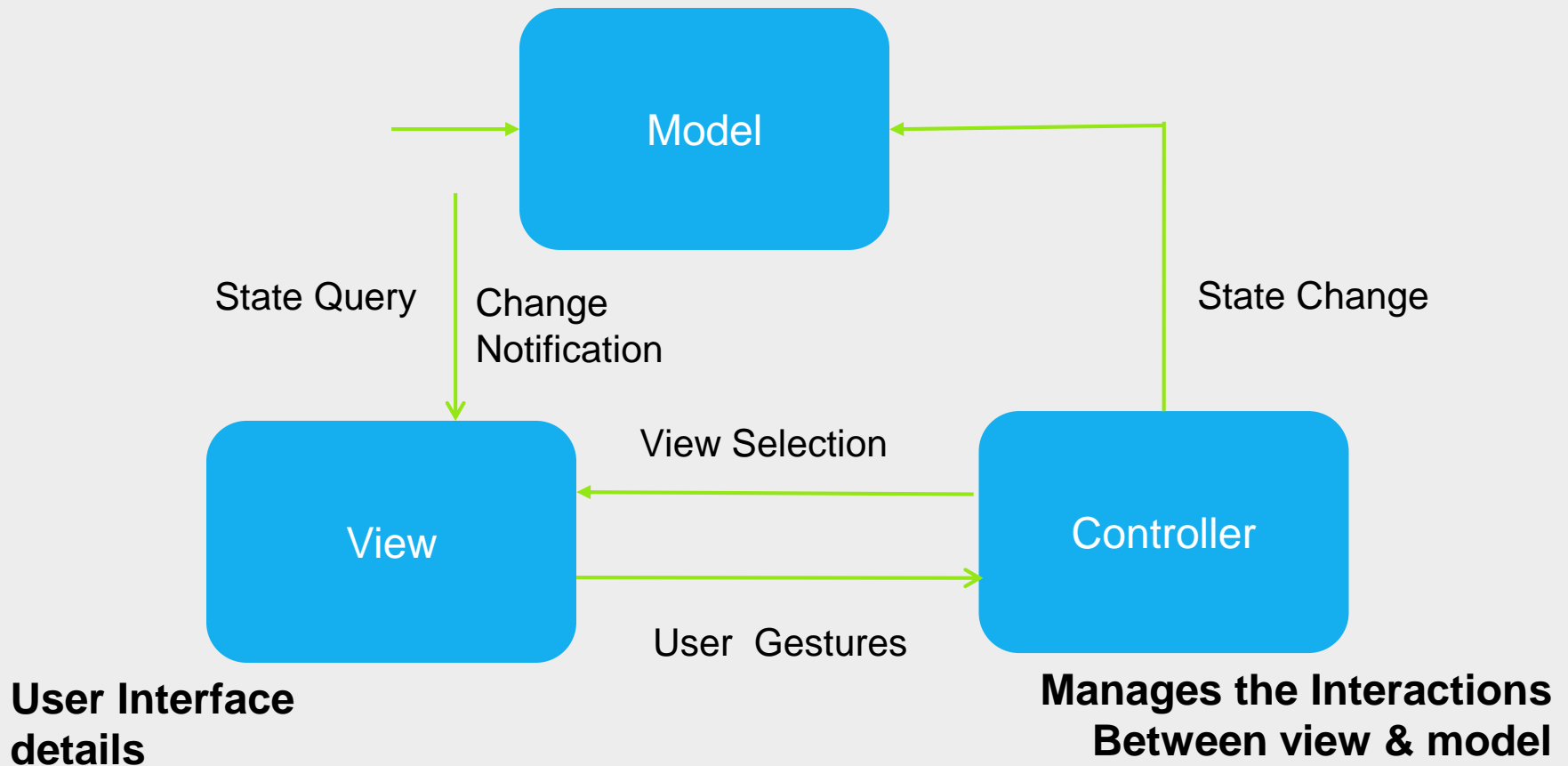
Visualforce understands Salesforce metadata and provides access to the respective user interface elements.

- For example, it automatically adds a calendar picker for date entry fields. It is hosted by Salesforce and tightly coupled with the Force.com platform.
- Because of this, Visualforce pages display the same performance as standard Salesforce pages.
- Many Visualforce components automatically re-create the standard Salesforce look and feel.
- Visualforce pages are automatically upgraded to the next Salesforce release. Visualforce separates the view of information from the navigation control and the data model.
- It conforms to the model-view-controller development pattern.

Visualforce Overview

First, a Primer on Model-View-Controller

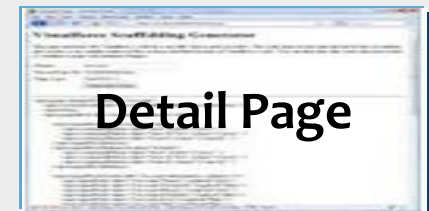
Business data and the rules For how to use the data



Visualforce Overview

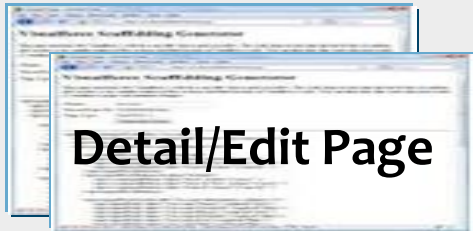
MVC Example: Opportunities Tab

Model	Opportunity object (master) Opportunity Product object (detail)
View	Opportunity overview page Opportunity list views page Opportunity detail page Opportunity edit page Opportunity Product detail page Opportunity Product multi-line edit page
Controller	Tab Navigation -> Overview page Overview list view picker -> list view page View links -> detail page New button -> edit page Edit button & links -> edit page Save button -> save action Delete buttons & links -> delete action



Visualforce Overview

MVC Example: Positions View (Page Layout)



Hard-wired representation of data model

- Detail sections (parent)
- Related lists (children)

Limited Layout options

- 1 or 2 column details
- Related lists at bottom

Detail & edit page share 1 layout

Position Edit Save Save & New Cancel

Information ! = Required Information

Title	<input type="text"/>	Owner	vaishali kunchur
Type	--None--	Priority	--None--
Department	--None--	Status	--None--
Location	--None--	Sub-Status	--None-- i
Pay Grade	--None-- i	Date Opened	<input type="text"/> [11/20/2012 8:36 PM]
Hiring Manager	<input type="text"/>	Date Closed	<input type="text"/> [11/20/2012 8:36 PM]
Duration	<input type="text"/>		
Legacy Position Number	<input type="text"/>		
Start Date	<input type="text"/>		

Description

Job Description	<input type="text"/>
Education	<input type="text"/>
Skills Required	<input type="text"/>

View link

View link

Edit button

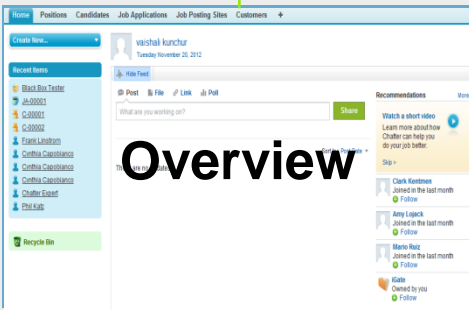
New button

Overview

List View

Detail Page

Edit Page



Recent Positions

New

Title	Location	Type	Hiring Manager	Status	Date Opened
Black Box Tester	San Francisco, CA	Full Time	Amy Lopez	Open	11/4/2012 11:29 PM
Technical Writer	Atlanta, GA	Full Time	Amy Lopez	New	
Developer	New York, NY	Full Time	vaishali kunchur	New	11/7/2012 11:35 PM

Position: Black Box Tester

Position Details: 11/4/2012 11:29 PM | Details | History | Add | Edit | Delete | Close | Sharing

Owner: vaishali kunchur (Chatter)

Type: Full Time

Department: Engineering

Location: San Francisco, CA

Pay Grade: E-00001

Hiring Manager: Amy Lopez

Status: Open

Legacy Position Number: 11/4/2012 11:29 PM

Start Date: 11/4/2012 11:29 PM

Created By: vaishali kunchur (11/4/2012 11:35 PM)

Last Modified By: vaishali kunchur (11/7/2012 11:35 PM)

Description: Black box tester for testing. Test Apps

Position Edit

Save | Save & Notify | Cancel

Information

Title:

Type:

Department:

Location:

Pay Grade:

Hiring Manager:

Status:

Legacy Position Number:

Start Date:

Owner: vaishali kunchur

Priority:

Status:

Date Opened:

Date Closed:

Description:

Visualforce Overview

Visualforce: Model/View/Controller Pattern

Model

What schema and data does the app use?

Standard & Custom Objects

Visualforce shares the same data as the rest of the platform; objects can have both Visualforce and standard UIs

View

How is the user interface and data presented?

Pages

Components

Visualforce pages are just like HTML/Web pages; components are reusable UI building blocks

Controller

How are the user interface actions and logic represented?

Standard & Custom Controllers

Controller Extensions

Controllers contain Apex that defines how pages interact with each other, and the rest of the platform

Visualforce Basics

Visualforce Pages

The screenshot shows the 'Visualforce Pages' setup page in Salesforce. On the left is a navigation sidebar with links for 'Force.com Home', 'System Overview', 'Personal Setup' (including 'My Personal Information', 'Email', 'Import', 'Desktop Integration', 'My Chatter Settings', and 'My Social Accounts and Contacts'), and 'App Setup' (including 'Customize', 'Create', and 'Develop'). The main content area is titled 'Visualforce Pages' and includes a 'Quick Find' search bar, a 'Help for this Page' link, and a description: 'Visualforce Pages provide a robust and easy to use mechanism to create new and exciting user experiences for your application or to enhance existing applications to optimize your users' productivity.' Below this is a 'View' dropdown set to 'All' and a 'Create New View' link. A table with columns 'Label', 'Name', 'Namespace Prefix', 'Api Version', 'Description', 'Created By Alias', and 'Created Date' is shown, with a 'New' button above it. The table currently displays 'No records to display.' and a filter bar with letters A-Z and 'Other'.

```
<apex:page standardController="Account">
  <apex:pageBlock title="Hello {!$User.FirstName}!">
    You are viewing the {!account.name} account.
  </apex:pageBlock>
  <apex:detail/>
</apex:page>
```

Page

A canvas similar to standard Web development model

Composed of HTML, page tags, and merge fields

Ability to reference any CSS, Flex, Flash, AJAX, or other Web technology

Support standard query string for parameters, referenced via /apex/pageName URL syntax

Composed on the server, not the client

Visualforce Components

<apex:actionFunction>

<apex:actionPoller>

<apex:actionRegion>

<apex:actionStatus>

<apex:actionSupport>

<apex:areaSeries>

<apex:attribute>

<apex:axis>

<apex:barSeries>

<apex:chart>

<apex:chartLabel>

<apex:chartTips>

<apex:column>

<apex:commandButton>

<apex:commandLink>

<apex:component>

<apex:componentBody>

<apex:composition>

<apex:dataList>

<apex:dataTable>

<apex:define>

<apex:actionFunction>

A component that provides support for invoking controller action methods directly from JavaScript code using an AJAX request. An `<apex:actionFunction>` component must be a child of an `<apex:form>` component.

Unlike `<apex:actionSupport>`, which only provides support for invoking controller action methods from other Visualforce components, `<apex:actionFunction>` defines a new JavaScript function which can then be called from within a block of JavaScript code.

Note: Beginning with API version 23 you can't place `<apex:actionFunction>` inside an iteration component — `<apex:pageBlockTable>`, `<apex:repeat>`, and so on. Put the `<apex:actionFunction>` after the iteration component, and inside the iteration put a normal JavaScript function that calls it.

Dynamic Visualforce Notation

Component: Apex.ActionFunction

Component Details

Usage

Attribute Name	Attribute Type	Description	Required?	Default Value	API Version	Access
action	ApexPages.Action	The action method invoked when the actionFunction is called by a DOM event elsewhere in the page markup. Use merge-field syntax to reference the method. For example, action="{!save}" references the save method in the controller. If an action is not specified, the page simply refreshes.			12.0	global
focus	String	The ID of the component that is in focus after the AJAX request completes.			12.0	global
id	String	An identifier that allows the actionFunction component to be referenced by other components in the page.			12.0	global
immediate	Boolean	A Boolean value that specifies whether the action associated with this component should happen immediately, without processing any validation rules associated with the fields on the page. If set to true, the action happens immediately and validation rules are skipped. If not specified, this value defaults to false.			12.0	global

Components

Create complex standard UI elements, such as detail areas and related lists, with a single tag

Built-in data binding to Salesforce data

Special UI elements to handle common design solutions, such as templates and data iterations

Built-in AJAX functionality supporting partial page refresh

Over 65 components at GA


Accessed through `<apex:tag>` syntax

Components

Visualforce
component tag

```
<apex:page title="Account">
  <apex:form>
    <apex:pageBlock title="Hello {!$User.FirstName}!">
      You are viewing the {!account.name} account. <p/>
      Change Account Name: <p/>
      <apex:inputField value="{!account.name}"/> <p/>
      <apex:commandButton action="{!save}" value="Save New Account Name"/>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

Data binding
expression



```
1 public class MyController {
2     private final Account account;
3     public MyController() {
4         account = [SELECT Id, Name, Site FROM Account
5                     WHERE Id = :ApexPages.currentPage().getParameters().get('id')];
6     }
7
8     public Account getAccount() {
9         return account;
10    }
11
12    public PageReference save() {
13        update account;
14        return null;
15    }
16 }
```

Controllers

Controllers contain the logic and data references a page uses

Because they are created in Apex, they have full access to Apex functionality (API, Web Services, etc.)

Pages interact with controllers through components that call data or actions

A controller can be used to maintain state across page interactions (in wizards, for example)

Controllers

Standard Controllers:

Are available for all API entities/objects, such as Account, Contact, Opportunity, etc., as well as custom objects

Provide access to standard Salesforce data and behavior

- Standard object record data
- Standard actions like save, edit, delete

Are referenced by using:

```
<apex:page standardController="Contact">
```

Controllers

Custom Controllers:

Are coded to create custom behaviors or non-standard data sets

Can be used to create wizards or leverage callouts

Are invoked by using :

```
<apex:page controller="MyController">
```

Controllers

Controllers Extensions:

Add custom behavior or additional data to controllers

Are invoked by using :

```
<apex:page standardController="Contact"  
extensions="MyClass, MyOtherClass">
```

Visualforce Basics

Expressions and Data Binding

Visualforce uses the expression syntax (also found in merge fields and formulas) to bind components to Salesforce data and actions in the page's controller

All contents in `{ ! . . . }` are evaluated as an expression

`{ ! $ User.FirstName }` shows the current user's first name in a page

Data context is provided to controllers by the ID parameter, just as in standard pages

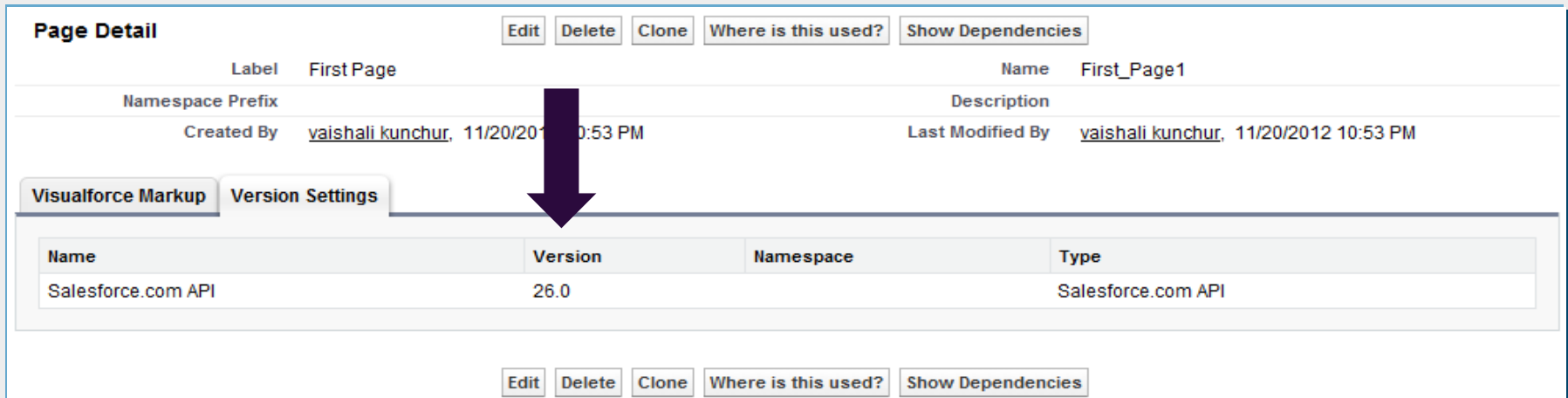
Are invoked by using :

https://ap1.salesforce.com/apex/First_page?id=066900000001cDEF

Visualforce Basics

Versioning

Visualforce pages and components are versioned
Previous versions of Visualforce elements remain available after new implementations are introduced, ensuring that your code works with each new release
The version settings tab displays the version of each page or component



Page Detail [Edit] [Delete] [Clone] [Where is this used?] [Show Dependencies]

Label	First Page	Name	First_Page1
Namespace Prefix		Description	
Created By	vaishali kunchur, 11/20/2012 10:53 PM	Last Modified By	vaishali kunchur, 11/20/2012 10:53 PM

[Visualforce Markup] **Version Settings**

Name	Version	Namespace	Type
Salesforce.com API	26.0		Salesforce.com API

[Edit] [Delete] [Clone] [Where is this used?] [Show Dependencies]

Standard tags begin with the word apex

- **Example:** `<apex:page>`
Apex is the namespace

Custom tags begin with the letter c

- **Example:** `<c:customtag>`
c is the namespace

Application developers can register custom namespaces to be displayed with custom tags instead of the letter c

- **Example:** `<thirdpartycompany:customtag>`
thirdpartycompany is the namespace

Visualforce Basics

Incorporating Visualforce Pages

Visualforce pages can be incorporated into your Salesforce UI by:

- Creating links to reference the unique page URL
- Overriding standard buttons to route to the new page
- Overriding tab overview pages to use the new page
- Creating custom tabs for the new page
- Creating custom buttons and links to route to the new pages
- Embedding pages into page layout
- Adding pages to a dashboard
- Using pages as custom help for a custom object

Visualforce Basics

Standard vs Custom

BEHAVIOR			
LOOK AND FEEL		Standard	Custom Apex
	Standard	Application framework and default UI (page layout)	Page layout and custom Apex classes
	Custom Visualforce	Visualforce pages using standard controllers	Visualforce pages using custom Apex controllers

Creating a Visualforce Hello World



Visualforce

Visualforce Benefits

Visualforce: Model/View/Controller Pattern

What kind of content can be included in a Visualforce page?
What is the MVC pattern and how does it relate to Visualforce?