Visualforce Cheatsheet

Overview

Visualforce is a user interface framework with automatic data bindings and API access to Salesforce. It allows developers to rapidly build sophisticated Model-view-controller web applications and custom user interfaces, hosted natively on the Force.com platform.

Getting Started

Turn on Development Mode to enable an inline Visualforce editor. From your personal settings, enter **Advanced** in the Quick Find box, then select **Advanced User Details**.

Enable the Development Mode checkbox and save.

All Visualforce page markup lies within an `<apex:page>` tag. Here's a simple page:

```
<apex:page showHeader="false">
 <h1>Hello World</h1>
</apex:page>
```

All tags take an optional or required set of attributes, such as `showHeader` above.
Use the inline editor auto-completion, or the Component Reference, to determine
the attributes for components.

List all Visualforce pages and Apex classes by navigating to **Develop** or **Custom Code** in **Setup**–whichever one appears.

Example

Visualforce supports auto-generated standard controllers, as well as extensions, to minimize the amount of coding needed. Here's an example of a Visualforce page that uses a custom controller:

```
<apex:page showHeader="false" controller="Hello">
    <apex:form>
       <apex:inputText value="{!theText}"/>
       <apex:commandButton value="Go">
        action="{!action}" reRender="dynamic"/>
    </apex:form>
    <apex:outputPanel id="dynamic">
         {!theText}
    </apex:outputPanel>
</apex:page>
```

This Visualforce page displays an input field and a button labeled **Go**. When clicked,
it sends the value of the field to the controller, which performs an action on it. Here, it duplicates the input. The page renders the result using an AJAX update. Here's the controller:

```
public class Hello {
   public PageReference action() {
      theText = theText + theText;
      return null;
   }
   public String theText {get;set;}
}
```

Core Tags

Here are core tags that often make up a Visualforce page:

`<apex:page>`	The obligatory outer tag.
`<apex:includeScript>`	A link to a JavaScript library, often in a static resource.
`<apex:stylesheet>`	A link to a CSS stylesheet, often in a static resource.

Template System

Template components let you create Visualforce pages that act as templates, having named areas, inserts that must be filled when the template is used.

`<apex:insert>`	Declares a named area that must be defined.
`<apex:composition>`	Adds the defined template.
`<apex:define>`	Provides content for the inserts.

Ajax and JavaScript

The reRender attribute on action tags such as `<apex:commandButton>` lets you refresh a part of a page.
The following tags provide additional AJAX and JavaScript support:

`<apex:actionFunction>`	Define JavaScript functions to be called from JavaScript code.
`<apex:actionPoller>`	A timer that sends an AJAX update request at specified interval.
`<apex:actionRegion>`	Defines an area in a page that demarcates whi ch components should be processed when AJAX requests generate.
`<apex:actionStatus>`	A component that displays the status of an AJAX update request.
`<apex:actionSupport>`	Adds asynchronous refresh support to a component.

Visualforce Cheatsheet

Core Form Components

Here are the core components used in forms. These should be embedded within a single <apex:form> component on a page:

`<apex:form>`	Container for input components.
`<apex:commandButton>`	A form button used to submit or reset a form.
`<apex:commandLink>`	An HTML link that performs an action.
`<apex:inputCheckbox>`	An HTML input element of type checkbox.
`<apex:inputField>`	Input element that corresponds to a field on a standard or custom object.
`<apex:inputFile>`	An input field to upload a file.
`<apex:inputHidden>`	An HTML input element of type hidden.
`<apex:inputSecret>`	An HTML input element of type password.
`<apex:inputText>`	An HTML input element of type text.
`<apex:inputTextArea>`	An HTML input element of type text area.
`<apex:selectList>`	A list of options for radio buttons or checkboxes.
`<apex:selectRadio>`	A set of related radio button input elements, displayed in a table.
`<apex:selectOption>`	A possible value for the <apex:selectCheckboxes> or <apex:selectList> components.
`<apex:selectCheckboxes>`	A set of related checkbox input elements, displayed in a table.
`<apex:selectOptions>`	A collection of values for the <apex:selectCheckboxes>, <apex:selectRadio>, or <apex:selectList> components.

Core Output

Mirroring many of the Core Form Components, these components are used for outputting text:

<apex:outputField>	A read-only display of a label and value for fields on custom objects.
<apex:outputLabel>	A label for an input or output field.
<apex:outputLink>	A link to a URL.
<apex:outputPanel >	A set of content that is grouped together, rendered with an HTML <span> tag, <div> tag, or neither.
<apex:outputText>	Displays styled or escaped text in a page.

Tables and List Output

These components are used to generate tables or lists by iterating over a collection:

`<apex:dataTable>`	An HTML table.
`<apex:column>`	A single column in a table.
`<apex:dataList>`	An ordered or unordered list of values.
`<apex:facet>`	A placeholder for content that is rendered in a specific part of the parent component.
`<apex:panelGrid>`	An HTML table element that lays out its components in consecutive cells.
`<apex:panelGroup>`	A container for multiple child components so that they can be displayed in a single panelGrid cell.

Miscellaneous HTML

These components generate HTML for embedding flash, iframes and images:

`<apex:flash>`	A Flash movie, rendered with object and embed tags.
`<apex:iframe>`	Creates an iframe in a page.
`<apex:image>`	A graphical image, rendered with an img tag.

Miscellaneous Visualforce

These components provide Visualforce messages, iteration, and include functionality:

`<apex:repeat>`	An unformatted iteration component.
`<apex:message>`	A message for a specific component, such as a warning or error.
`<apex:messages>`	All messages that were generated for all components on the current page.
`<apex:include>`	Inserts a second Visualforce page into the current page.
`<apex:param>`	Lets parameters be fed into other components.
`<apex:scontrol>`	An inline frame that displays an s-control.
`<apex:variable>`	A local variable representing an expression.

Visualforce Cheatsheet

salesforce force.com

Default Look and Feel

CRUD and other screens have a standard look and feel. The following components generate output that conforms to that look and feel:

`<apex:detail>`	The standard detail page for a particular object.
`<apex:enhancedList>`	The list view picklist for an object, including its associated list of records.
`<apex:listViews>`	The list view picklist for an object, including its associated list of records.
`<apex:relatedList>`	A list of records that are related to a parent record with a lookup or master-detail relationship.
`<apex:pageBlock>`	Defines a visual block within a page.
`<apex:pageBlockButtons>`	Defines buttons related to a block.
`<apex:pageBlockSection>`	A section of data within a `<apex:pageBlock>` component.
`<apex:page BlockSectionItem>`	A single piece of data in a `<apex:pageBlockSection>` that takes up one column in one row.
`<apex:pageBlockTable>`	A list of data displayed as a table within either a `<apex:pageBlock>` or `<apex:pageBlockSection>` component.
`<apex:pageMessage>`	Displays custom messages.
`<apex:pageMessages>`	Displays all messages that were generated for all components on the page.
`<apex:panelBar>`	A page area that includes one or more `<apex:panelBarItem>` tags that can expand.
`<apex:panelBarItem>`	A section of a `<apex:panelBar>` that can expand or collapse.
`<apex:sectionHeader>`	A title bar for a page.
`<apex:tab>`	A single tab in a `<apex:tabPanel>`.
`<apex:tabPanel>`	A page area that displays as a set of tabs.
`<apex:toolbar>`	A toolbar that can contain any number of child components.
`<apex:toolbarGroup>`	A group of components within a toolbar.

Custom Components

Create your own components that will reside in the c namespace, for example `<c:helloWorld/>`:

`<apex:component>`	The outer tag that starts the definition of a component.
`<Vapex:attribute>`	A definition of a typed attribute on a custom component.
`<apex:componentBody>`	A place holder for content injected into a component.

Sites

Sites is based on Visualforce pages, and the following tags provide additional functionality:

`<site:googleAnalyticsTracking>`	Used to integrate Google Analytics.
`<site:previewAsAdmin>`	Shows detailed messages in administrator preview mode.

Messaging

Visualforce can also be used to create email templates:

`<messaging:attachment>`	Appends an attachment to the email.
`<messaging:emailHeader>`	Adds a custom header to the email.
`<messaging:emailTemplate>`	Defines a Visualforce email template.
`<messaging:htmlEmailBody>`	The HTML version of the email body.
`<messaging:plainTextEmailBody>`	The plain text (non-HTML) version of the email body.

Ideas

Applications that integrate Ideas have additional components available:

`<ideas:detailOutputLink>`	A link to the page displaying an idea.
`<ideas:listOutputLink>`	A link to the page displaying a list of ideas.
`<ideas:profileListOutputLink>`	A link to the page displaying a user's profile.

Maps

Add interactive maps to your pages with these components:

`<apex:map>`	Adds an interactive, JavaScript-based map to the page.
`<apex:mapMarker>`	Adds a marker to a location on the map.
`<apex:mapInfoWindow>`	Adds a pop-up info window to a marker on a map.

Knowledge

You can embed Knowledge functionality on your pages:

`<knowledge:articleCaseToolbar>`	UI component used when an article is opened from the case detail page.
`<knowledge:articleList>`	A loop on a filtered list of articles.
`<knowledge:article RendererToolbar>`	Displays a header toolbar for an article.
`<knowledge:articleTypeList>`	A loop on all available article types.
`<knowledge:categoryList>`	A loop on a subset of the category hierarchy.

Visualforce Cheatsheet

Chatter

Add Chatter UI widgets:

`<chatter:feed>`	Displays the Chatter feed for a record.
`<chatter:feedWithFollowers>`	An integrated UI component that displays the Chatter feed for a record, as well as its list of followers.
`<chatter:follow>`	Renders a button for a user to follow or unfollow a Chatter record.
`<chatter:followers>`	Displays the list of Chatter followers for a record.

Visual Workflow

Components that present your process visual workflow:

`<flow:interview>`	Embeds a Flow interview in the page.

Apex Support

Here are Apex classes, available in the ApexPages namespace, which can be used to reference Visualforce functionality:

`Action`	Create and invoke actions on an object.
`Message`	Create messages to be displayed, usually for errors.
`PageReference`	A reference to a Visualforce page.
`SelectOption`	Specifies one of the possible values for the selectCheckboxes, selectList, or selectRadio components.
`StandardController`	Reference the standard Visualforce controllers.
`StandardSetController`	Reference standard set controllers for iteration.

Global Variables

These reference general information about the current user and your organization on a Visualforce page:

`$Action`	Reference standard actions on objects.
`$API`	Refer to API URLs or sessions IDs.
`$Component`	Reference a component from within, for example, JavaScript.
`$CurrentPage`	Reference name, parameters, URL of current page.
`$Label`	References a custom label.
`$ObjectType`	Access metadata of objects.
`$Organization`	Access information about the company profile.
`$Page`	The way to reference a Visualforce page.
`$Resource`	Used for accessing static resources.
`$SControl`	References an existing custom s-control by name.
`$Site`	Reference site information, such as domain name.
`$System.OriginDateTime`	Represents the literal value of 1900-01-01 00:00:00.
`$User`	Information about the current user.
`$UserRole`	The current user's role information.
`$Profile`	The current user's profile.

Static Resources

Upload static resources, such as stylesheets and images, into a zip, and then reference them using the URLFOR formula and $Resource merge field. For example, if you've uploaded images/Blue.jpg into a zip called TestZip, reference it like this:

```
<apex:image url="{!URLFOR($Resource.TestZip,'images/Blue.jpg')}"/>
```