1. Introduction to Visualforce Pages

**Enable Visualforce Development Mode**

**1.** Click ***Your Name*** > **Setup** > **My Personal Information** > **Personal Information**.

**2.** Click **Edit**.

**3.** Select the Development Mode checkbox, then click **Save**.

**Create a Visualforce Page**

**1.** In your browser, add /apex/hello to the URL for your Salesforce instance. For example, if your Salesforce instance is https://na1.salesforce.com, the new URL is https://na1.salesforce.com/apex/hello. You will see the following error:

**2.** Click the **Create Page hello** link to create the new page. You will see your new page with some default markup.

**Edit the Visualforce Page**

**1.** You don’t want the heading of the page to say “Congratulations,” so change the contents of the <h1> tag to Hello World,

remove the comments, and the “This is your new page” text. The code now looks like this:

<apex:page>

<h1>Hello World</h1>

</apex:page>

**2.** Click the **Save** button at the top of the Page Editor.

**Find all Visualforce Pages**

Now that you’ve created a Visualforce page, you’ll need to know where to find it.

**1.** Click ***Your Name*** > **Setup** > **Develop** > **Pages**.

**2.** Scroll down to locate the page created in Step 2—hello.

**Alternative Page Creation**

**1.** Go to ***Your Name*** > **Setup** > **Develop** > **Pages**, then click **New**.

**2.** Create and label the page hello2.

**3.** Click **Save**.

**4.** Navigate to the new page using the URL as you did in Step 2: [https://*your-salesforce-instance*/apex/hello2](https://your-salesforce-instance/apex/hello2)

**1.** Add sidebar="false" within the start tag of the <apex:page> component as follows:

<apex:page sidebar="false">

**2.** Click **Save**.

**3.** Position your cursor just after the final quotation mark ("), and hit the space bar. A helpful list of attributes pop up that

are valid for the <apex:page> component. Choose the showHeader attribute.

**4.** The attribute is automatically added to your page, and you now need to supply a value for the attribute. Add false. Your

complete first line should look like this:

<apex:page sidebar="false" showHeader="false">

**5.** Click **Save** (remember, you can also press CTRL+S as a shortcut).

This time your page looks completely different. By setting the showHeader attribute to false, you’ve not only removed the top header, but all the default styling associated with the page.

**6.** Change the showHeader attribute’s value to true.

**7.** Click **Save**.

**Add Additional Components**

**1.** Click the **Component Reference** link in the Page Editor. A help popup window displays with all available components.

**2.** Click <apex:pageBlock>. A description of what the component does, and what attributes you can add to change its behavior displays in the Component Details tab.

**3.** Click the Usage tab to see an example of how to use the component. You’ll notice that the <apex:pageBlock> component is often used with the <apex:pageBlockSection> component. Click <apex:pageBlockSection> to learn more about that component.

In general, you’ll dip into the component reference whenever you need to. You’ll soon learn what the major components do—and while some of them take a large number of attributes, in practice you will only use a handful.

Now add both components to your page. We’re going to go a little faster here—see if you can do this without looking at the final code below:

**4.** Within the <apex:page> component, add an <apex:pageBlock> component with a title attribute set to A Block

Title.

**5.** Within the <apex:pageBlock> component, add an <apex:pageBlockSection> component, with its title attribute

set to A Section Title.

**6.** Within the <apex:pageBlockSection>, add some text, like I'm three components deep!

**7.** Click **Save**. Your final code will look something like this:

<apex:page sidebar="false">

**<apex:pageBlock title="A Block Title">**

**<apex:pageBlockSection title="A Section Title">**

**I'm three components deep!**

**</apex:pageBlockSection>**

**</apex:pageBlock>**

</apex:page>

**Add Nested Components**

Adding additional components is easy.

**1.** Navigate to the end of the <apex:pageBlockSection> component, and add another <apex:pageBlockSection>

component with its own title. Both <apex:pageBlockSection> components must be contained within the same

<apex:pageBlock> component.

**2.** Click **Save** and admire your handiwork.

<apex:page sidebar="false">

<apex:pageBlock title="A Block Title">

<apex:pageBlockSection title="A Section Title">

I'm three components deep!

</apex:pageBlockSection>

<apex:pageBlockSection title="A New Section">

This is another section.

</apex:pageBlockSection>

</apex:pageBlock>

</apex:page>

**Global Variables**

**1.** Modify your existing page to include the following line: {! $User.FirstName}. Remember that any content must lie

within the <apex:page> component (between its open and closing tags).

**2.** Click **Save**.

Your Visualforce page looks something like this:

<apex:page sidebar="false">

{! $User.FirstName}

</apex:page>

**Basic Formulas**

1. Add this to your Visualforce page: {! $User.firstname & ' ' & $User.lastname}

**2.** Add this to your Visualforce page:

<p> Today's Date is {! TODAY()} </p>

<p> Next week it will be {! TODAY() + 7} </p>

Trialhead link:

<https://trailhead.salesforce.com/users/00550000006yDdKAAU/trailmixes/prepare-for-your-salesforce-platform-developer-i-credential>

*DAY 2 Lab Assignment:*

1. Create a Visual Force Page to Display Posion Details Page without related lists.

The Job Application Related list should be shown in a data list form on the same page.

1. Create a Visualforce page to display Account record with detail page with Opportunities shown in Data Table and Cases shown in Data List on the same page

***DAY 3 Lab Assignments:***

1. Create a component to show the text with the color, font and font size of users choice.
2. Implement the above component design using Javascript.

Check the following code and set it working:

* 1. <apex:page id=“thePage”>
  2. <!- - the following script element contains a simple function for changing the font. - - >
  3. <script language=“JavaScript” type=“text/javascript”>
  4. function changeFont(input, textid) {
  5. if (input.checked) document.getElementById(textid).style.fontWeight = “bold”;
  6. else document.getElementById(textid).style.fontWeight = “ normal”;
  7. }
  8. </script>
  9. <!- - the first output panel calls the function, passing in the existing occurrence of the checkbox, as well as the DOM ID of the output component- - >
  10. <apex:outputPanel layout=“block”>
  11. <label for=“checkbox”>Click this box to change text font:</label>
  12. <apex:input id=“checkbox” type=“checkbox” onclick=“changeFont(this,’{!$Component.thePanel}’);”/>
  13. </apex:outputPanel>
  14. <!- - the second output panel contains the text that will be changed - - >
  15. <apex:outputPanel id=“thePanel”layout=“block”>Change me!
  16. </apex:outputPanel>
  17. </apex:page>

AJAX Code

* 1. <apex:page standardController=“Account”>
  2. <! - - Same as previous example - - >
  3. <apex:outputPanel>
  4. **<apex:actionSupport event=“onmouseover” rerender=“detail”>**
  5. <apex:param name=“cid” value=“{!contact.id}”/>
  6. **</apex:actionSupport>**
  7. {!contact.Name}
  8. </apex:outputPanel>
  9. </apex:column>
  10. </apex:dataTable>
  11. </apex:form>
  12. <apex:pageBlock>
  13. <apex:outputPanel id=“detail”>
  14. <apex:actionStatus startText=“Requesting…”>
  15. <apex:facet name=“stop”>
  16. <apex:detail subject=“{!$CurrentPageReference.parameters.cid}” relatedList=“false” title=“false”/>
  17. </apex:facet>
  18. </apex:actionStatus>
  19. </apex:outputPanel>
  20. </apex:page>