Salesforce visualforce pages complete reference

Various field types in visualforce

Hello world with external css and js

Show list of records

Show one record detail view

Record edit view

Create new record – normal save, ajax based insert

VF components

Controllers – standard, custom, extensions

Search functionality

Related lists

Hello world with external css and js

Go to quick find and search for visualforce pages. Go to visualforce pages.

Create new page.

```
<apex:page>
  <apex:includeScript value="{! $Resource.jquery331 }"/>
  <apex:stylesheet value="{!URLFOR($Resource.AgentDemoCSS)}"/>
  <apex:includeScript value="{! $Resource.AgentDemoJS }"/>
  <script>
  alert('here');
  </script>
</apex:page>
```

Show list of records

To show list of records you need to use a controller.

Notice the additional code. We have defined record set var in apex:page. This var is used in the loop apex repeat.

```
<apex:page standardController="Account" recordSetVar="AccountsList">

  <apex:includeScript value="{! $Resource.jquery331 }"/>
  <apex:stylesheet value="{!URLFOR($Resource.AgentDemoCSS)}"/>
  <apex:includeScript value="{! $Resource.AgentDemoJS }"/>

  <apex:pageblock>
    <apex:repeat var="a" value="{!AccountsList}" rendered="true"  id="account_list">
      <li>
        <apex:outputLink value="/{!a.ID}" >
          <apex:outputText value="{!a.Name}"/>
        </apex:outputLink>
      </li>
    </apex:repeat>
  </apex:pageblock>

  <script>
  //alert('here');
  </script>

</apex:page>
```

Save code and preview

Show one record detail view

Now if you click on the account name in above page, it takes you to standard detaile view of account record. We will create a custom detail view for account record. This page will get the record id from URL, pull its details and display in visualforce.

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

  <apex:includeScript value="{! $Resource.jquery331 }"/>
  <apex:stylesheet value="{!URLFOR($Resource.AgentDemoCSS)}"/>
  <apex:includeScript value="{! $Resource.AgentDemoJS }"/>

  <apex:pageBlock title="Record Detail Page">
  <apex:detail subject="{!Account.Id}" relatedList="false" title="false"/>
  </apex:pageBlock>

</apex:page>
```

Above code will display detail view using standard layout. To implement your own layout replace apex:detail tag with your code such as <apex:outputText>{!Account.Name}</apex:outputText>

Record edit view

Let create an edit button in above detail page.

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

  <apex:includeScript value="{! $Resource.jquery331 }"/>
  <apex:stylesheet value="{!URLFOR($Resource.AgentDemoCSS)}"/>
  <apex:includeScript value="{! $Resource.AgentDemoJS }"/>

  <apex:pageBlock title="Record Detail Page">
  <apex:form>
  <apex:inputField value="{!Account.Name}" />
  <apex:commandButton action="{!Save}" value="Save and return to standard detail page" />
  </apex:form>
  </apex:pageBlock>

</apex:page>
```

Above submit button will save and return to standard detail page. Ideally you would want to retrun to your custom detail page. To do so change submit button and add extension controller.

```
<apex:page standardController="Account" extensions="CustomSave">

  <apex:pageBlock title="Record Detail Page">
  <apex:form >
  <apex:inputField value="{!Account.Name}" />
  <apex:commandButton action="{!saveAndGo}" value="Save and return to custom detail page" />
  </apex:form>
  </apex:pageBlock>

  </apex:page>
```

CustomSave class:

```
public class CustomSave {
   private ApexPages.StandardController stdController;
   public CustomSave(ApexPages.StandardController stdController) {
   this.stdController = stdController;
}

   public PageReference saveAndGo () {
      stdController.save();
      PageReference v = Page.Account_Detail_View;
      v.getParameters().put('id',stdController.getId());
      v.setRedirect(true);
      return v;
   }
}
```