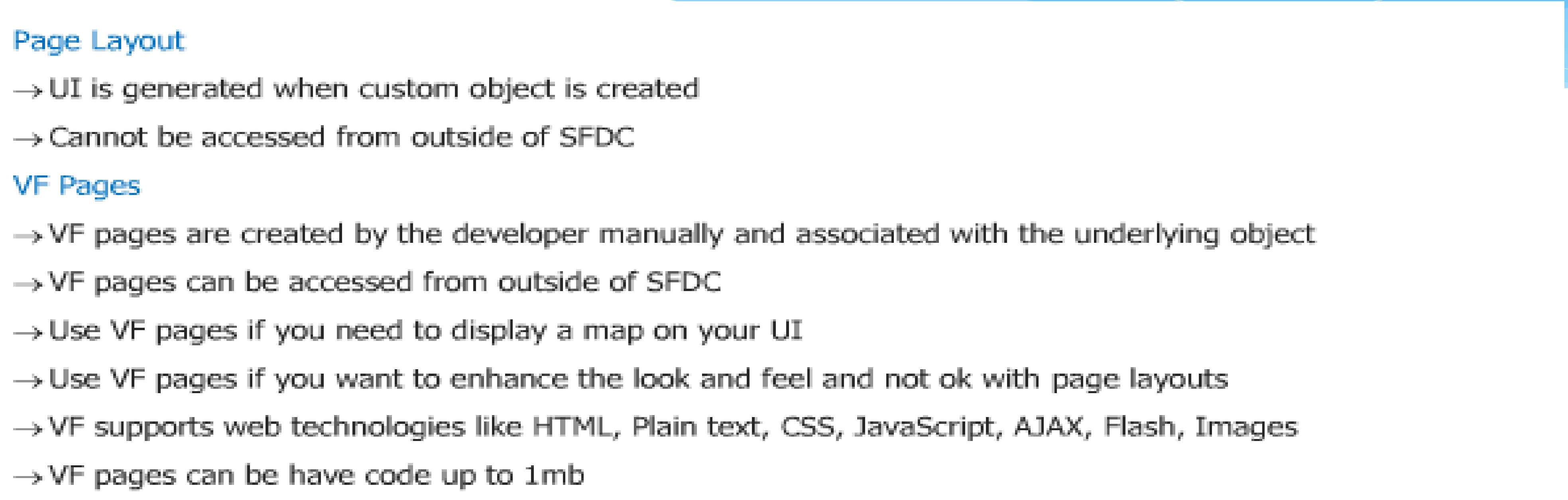
Introduction to visualforce

Pagelayout vs Visualforce



Page Layout

→ UI is generated when custom object is created

→ Cannot be accessed from outside of SFDC

VF Pages

→ VF pages are created by the developer manually and associated with the underlying object

→ VF pages can be accessed from outside of SFDC

→ Use VF pages if you need to display a map on your UI

→ Use VF pages if you want to enhance the look and feel and not ok with page layouts

→ VF supports web technologies like HTML, Plain text, CSS, JavaScript, AJAX, Flash, Images

→ VF pages can be have code up to 1mb

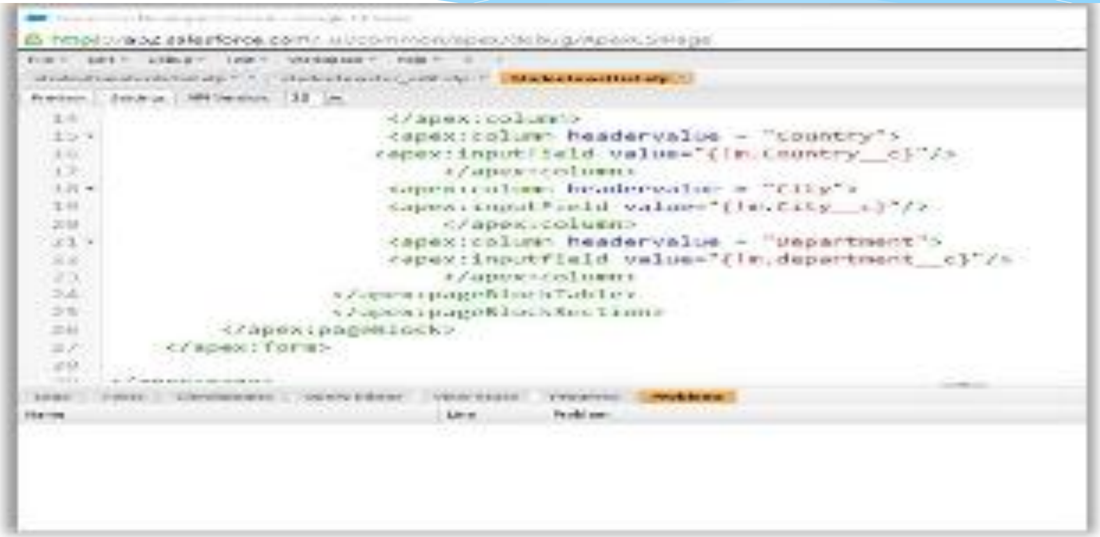
Developer tools to create vf pages

→Inline Editor

→Force.com IDE

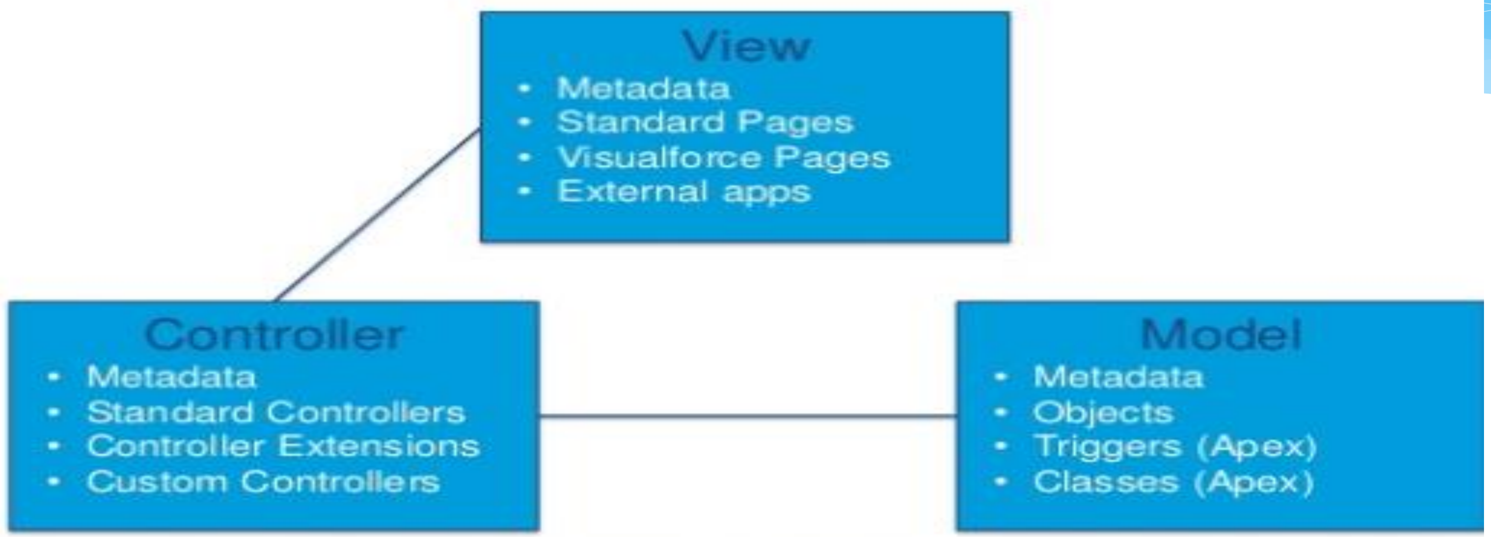
- » Can create a VF page
- » VF components
- » Static Resources and Controllers

→Developer Console



```
                                        </apex:column>
                                        <apex:column headervalue = "Country">
                                       <apex:inputField value="{!m.Country__c}"/>
                                            </apex:column>
                                        <apex:column headervalue = "City">
                                        <apex:inputField value="{!m.City__c}"/>
                                            </apex:column>
                                        <apex:column headervalue = "Department">
                                        <apex:inputField value="{!m.department__c}"/>
                                            </apex:column>
                                  </apex:pageBlockTable>
                                  </apex:pageBlockSection>
                        </apex:pageBlock>
                </apex:form>
```

MVC model


View
• Metadata
• Standard Pages
• Visualforce Pages
• External apps
Controller
• Metadata
• Standard Controllers
• Controller Extensions
• Custom Controllers
Model
• Metadata
• Objects
• Triggers (Apex)
• Classes (Apex)

Visualforce Pages -benefits

→ Apart from using VF pages to replace standard page layouts VF pages can be used in many other areas

→ It can be used in dashboards

→ Can be used as a part of section in a standard page layout

→ It can be used to display a map, say Google map

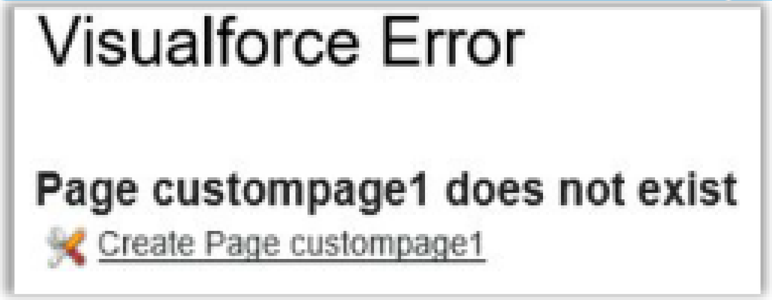
→ Declarative features cannot allow you to edit multiple records - VF page is required

→ Want a different look and feel and want to leverage HTML, CSS, Flash and JavaScript and JS libraries

→ VF adheres to the MVC pattern like many standard enterprise class platforms

Creating a page

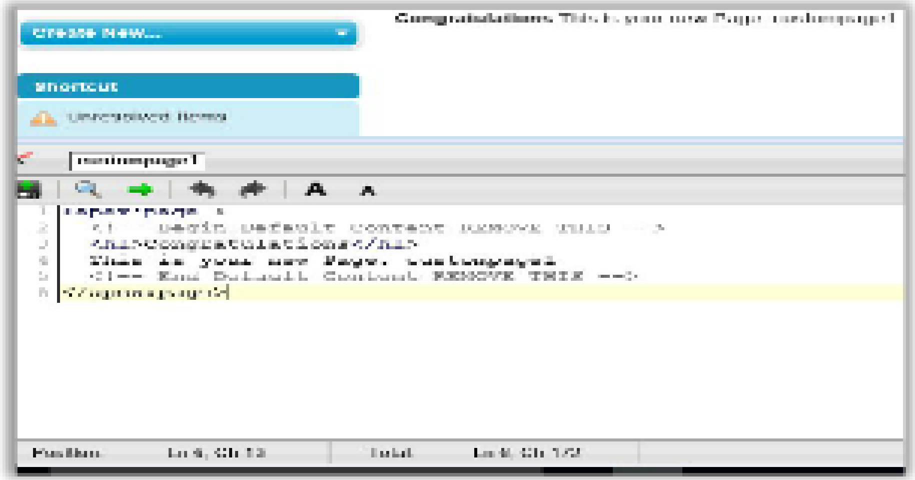
- Ensure your user record has "*Development mode*" enabled
- This helps in editing and seeing the page at the same time
- Login to SFDC dev environment and
- Type the following in the URL and press enter
  - » https://c.ap2.visual.force.com/apex/custompage1
- You will get a message as shown in the screenshot
- Click on "*Create Page custompage1*" link
- A new VF page will be created for you with some custom VF tags as shown in the screenshot



Visualforce Error

Page custompage1 does not exist

Create Page custompage1



Congratulations This is your new Page: custompage1

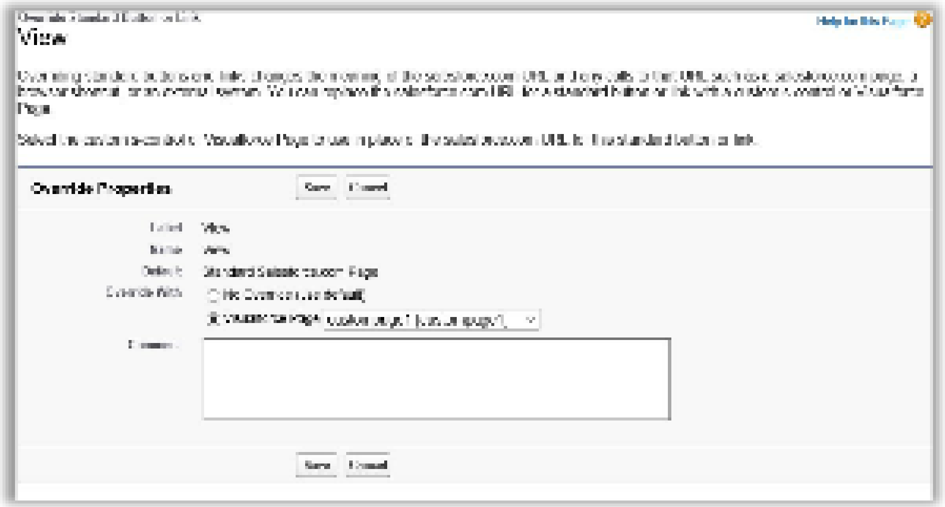
```
<apex:page >
  <!-- Begin Default Content REMOVE THIS -->
  <h1>Congratulations</h1>
  This is your new Page: custompage1
  <!-- End Default Content REMOVE THIS -->
</apex:page>
```

Linking pages to standard controller

→ We have created the page in the previous slide, we will link the page to the custom object

→ After linking you can click on the tab for the *student_master* object and click a particular record, your new VF page will be displayed

```
<apex:page standardController="student_master__c" >
  <!-- Begin Default Content REMOVE THIS -->
  <h1>Welcome to Visual Force, {!$User.FirstName} </h1>
  This is your new Page: custompage1
  <!-- End Default Content REMOVE THIS -->
</apex:page>
```



View

[illegible]

[illegible]

Override Properties

Label: View
Name: View
Default: Standard Salesforce.com Page
Override With: No Override (use default)
[✓ Visualforce Page] custompage1 [custompage1]
Comment:

Vf components

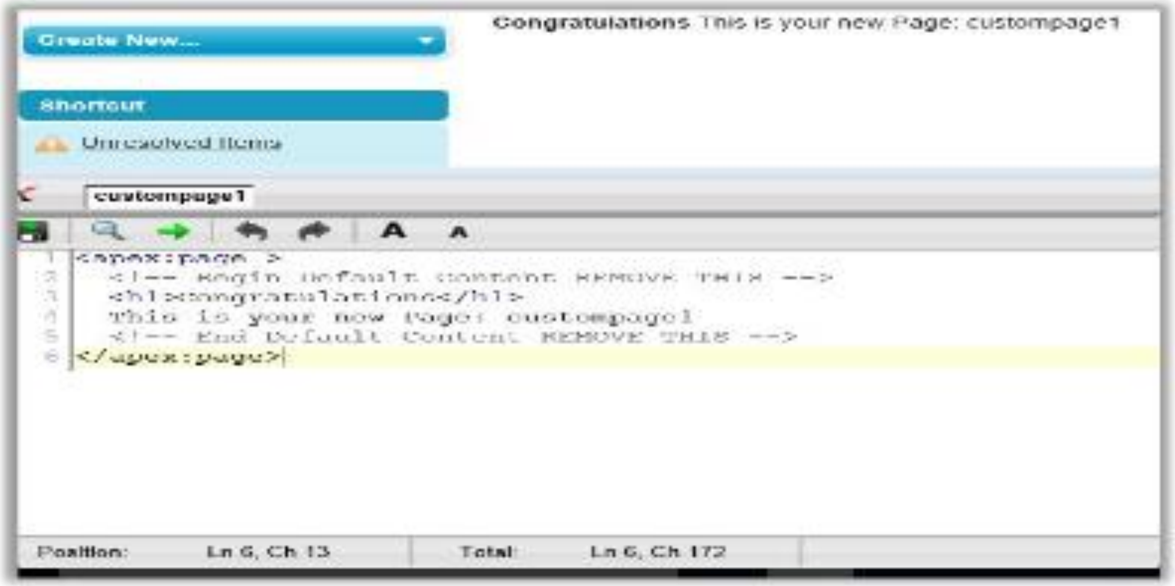
→ <apex:page title ="studentmasterpage" > -->

→<Apex: is the Visualforce component tag

→"page" is the tag name

→"title" is the tag attribute and "studentmasterpage" is the tag value

→</apex:page> is the end tag

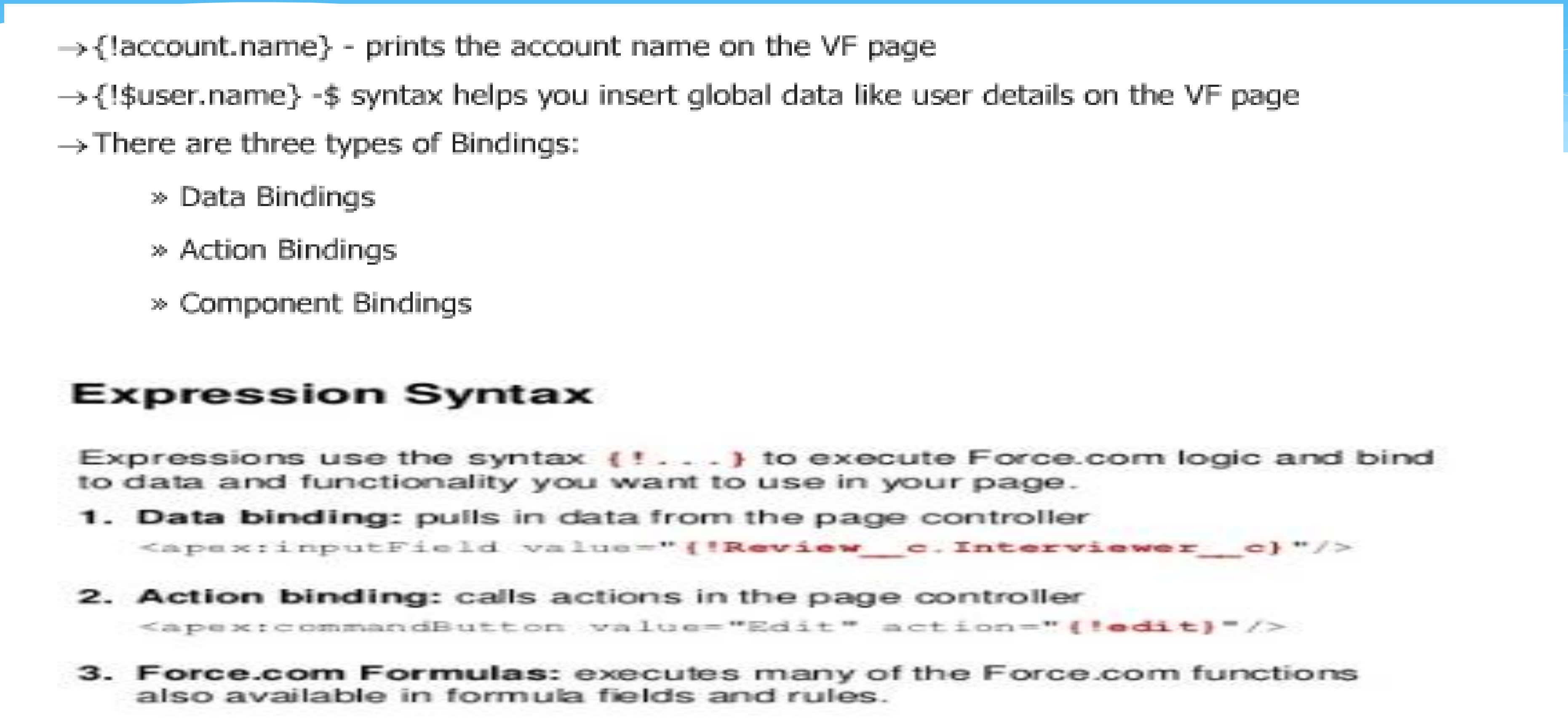


Congratulations This is your new Page: custompage1

```
<apex:page >
  <!-- Begin Default Content REMOVE THIS -->
  <h1>Congratulations</h1>
  This is your new Page: custompage1
  <!-- End Default Content REMOVE THIS -->
</apex:page>
```

Position: Ln 6, Ch 13   Total: Ln 6, Ch 172

Bindings



→{!account.name} - prints the account name on the VF page

→{!$user.name} -$ syntax helps you insert global data like user details on the VF page

→There are three types of Bindings:

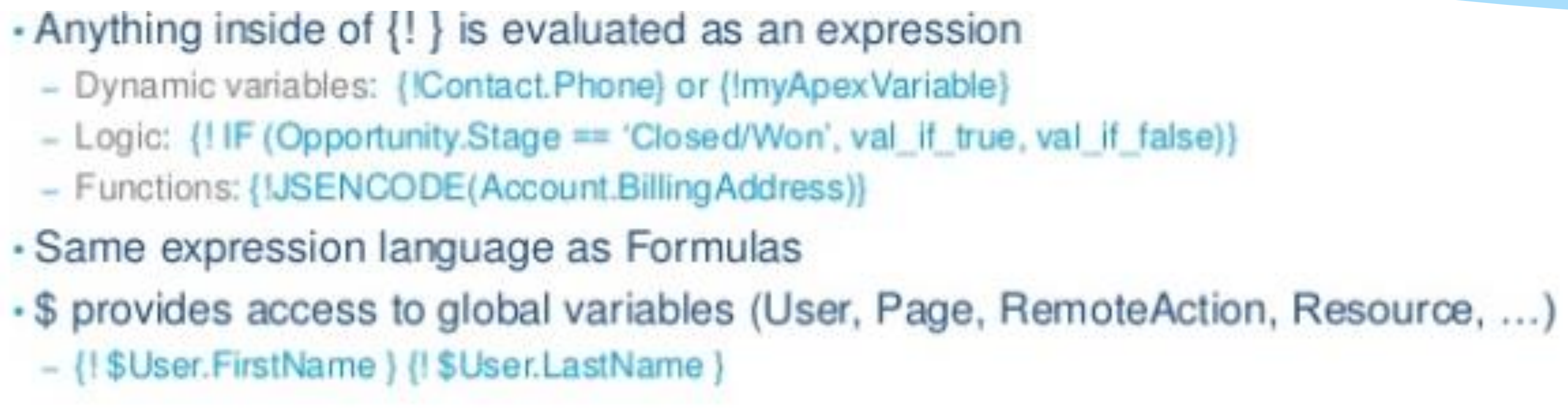
- » Data Bindings
- » Action Bindings
- » Component Bindings

Expression Syntax

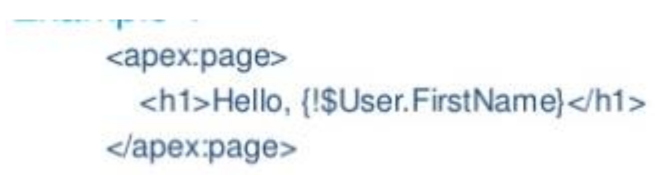
Expressions use the syntax {!...} to execute Force.com logic and bind to data and functionality you want to use in your page.

1. **Data binding:** pulls in data from the page controller
   ```
   <apex:inputField value="{!Review__c.Interviewer__c}"/>
   ```
2. **Action binding:** calls actions in the page controller
   ```
   <apex:commandButton value="Edit" action="{!edit}"/>
   ```
3. **Force.com Formulas:** executes many of the Force.com functions also available in formula fields and rules.

Bindings

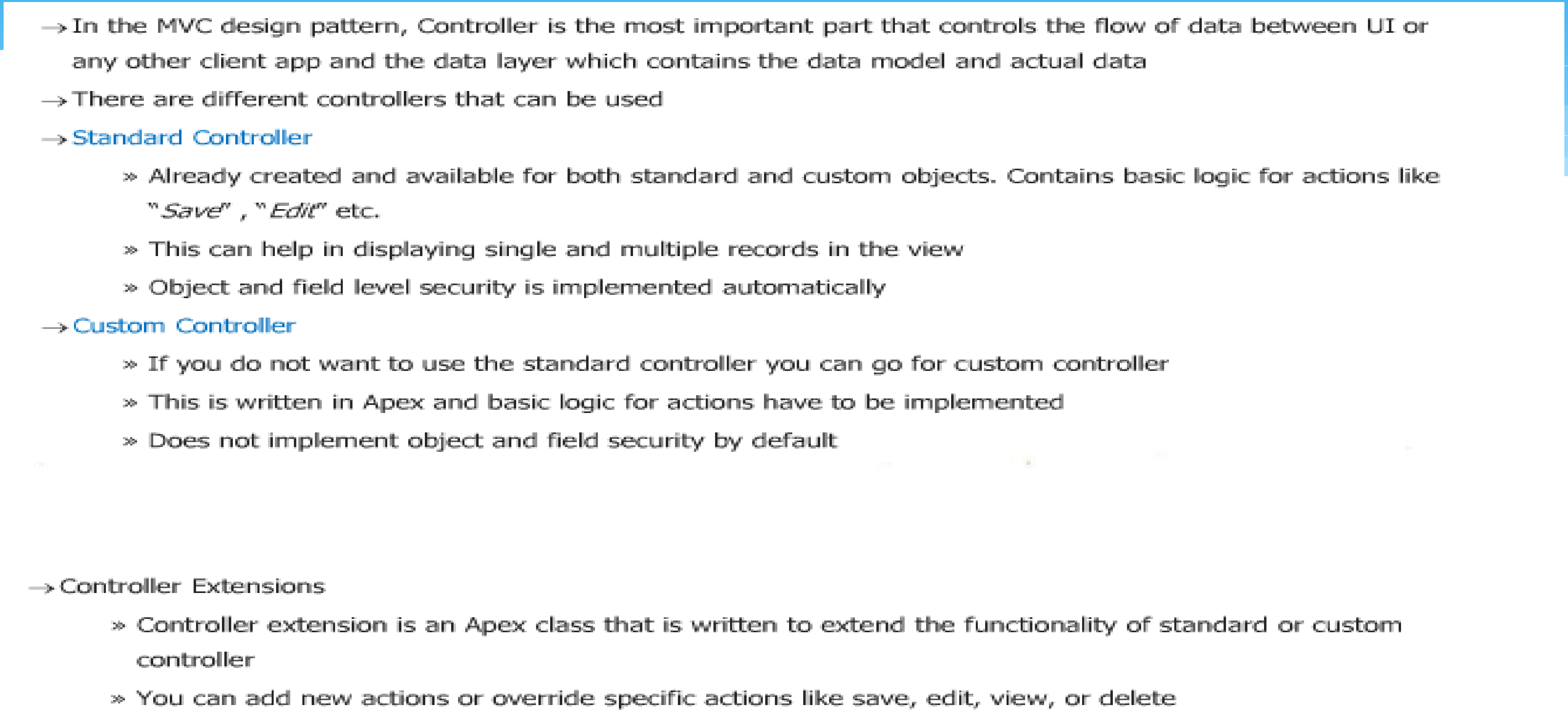


- Anything inside of {! } is evaluated as an expression
  - Dynamic variables: {!Contact.Phone} or {!myApexVariable}
  - Logic: {! IF (Opportunity.Stage == 'Closed/Won', val_if_true, val_if_false)}
  - Functions: {!JSENCODE(Account.BillingAddress)}
- Same expression language as Formulas
- $ provides access to global variables (User, Page, RemoteAction, Resource, …)
  - {! $User.FirstName } {! $User.LastName }



```
<apex:page>
   <h1>Hello, {!$User.FirstName}</h1>
</apex:page>
```

Controllers – Standard/Custom/Extension



→ In the MVC design pattern, Controller is the most important part that controls the flow of data between UI or any other client app and the data layer which contains the data model and actual data

→ There are different controllers that can be used

→ Standard Controller

- » Already created and available for both standard and custom objects. Contains basic logic for actions like "*Save*" , "*Edit*" etc.
- » This can help in displaying single and multiple records in the view
- » Object and field level security is implemented automatically

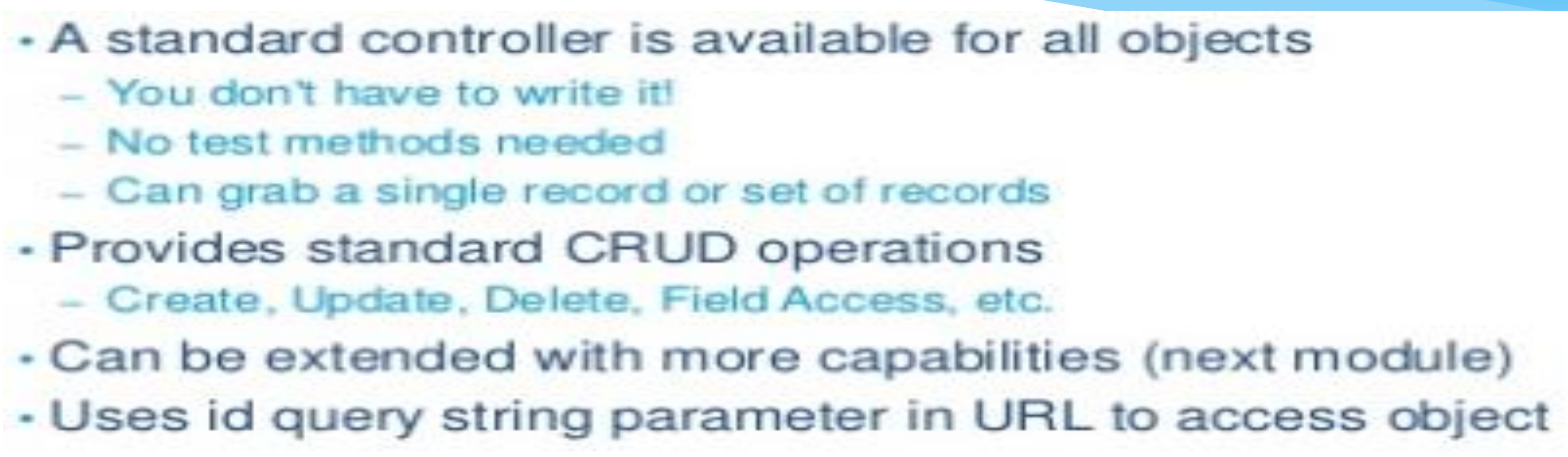
→ Custom Controller

- » If you do not want to use the standard controller you can go for custom controller
- » This is written in Apex and basic logic for actions have to be implemented
- » Does not implement object and field security by default

→ Controller Extensions

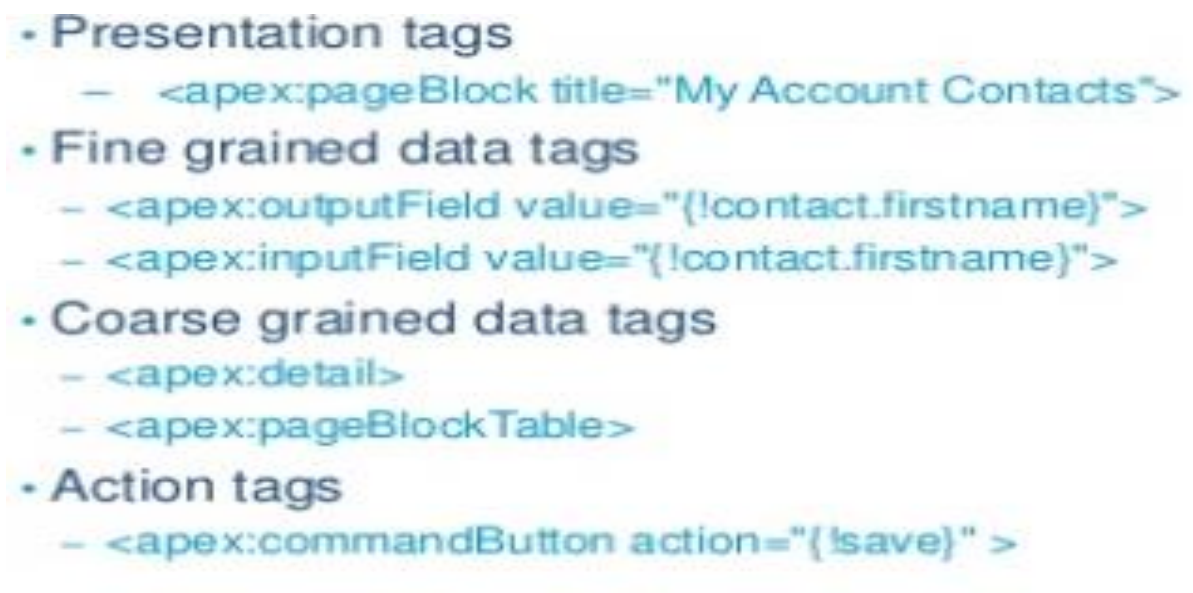
- » Controller extension is an Apex class that is written to extend the functionality of standard or custom controller
- » You can add new actions or override specific actions like save, edit, view, or delete

Standard Controller



- A standard controller is available for all objects
  - You don't have to write it!
  - No test methods needed
  - Can grab a single record or set of records
- Provides standard CRUD operations
  - Create, Update, Delete, Field Access, etc.
- Can be extended with more capabilities (next module)
- Uses id query string parameter in URL to access object

Tags



- Presentation tags
  - <apex:pageBlock title="My Account Contacts">
- Fine grained data tags
  - <apex:outputField value="{!contact.firstname}">
  - <apex:inputField value="{!contact.firstname}">
- Coarse grained data tags
  - <apex:detail>
  - <apex:pageBlockTable>
- Action tags
  - <apex:commandButton action="{!save}" >

Example 1

→ Create a new VF page from developer console "*studentmasterdetail*" "(https://c.ap2.visual.force.com/apex/studentmasterdetail?id="record id") - Replace the record id with a valid record id of existing student master record

→ In this example we are trying to display master and details object --> student master record and student marks

→ There are two ways:

- » Refer to the complete VF page in the next slide. You can copy paste the code and create a new VF page
- » Refer to the demo on how to open this page

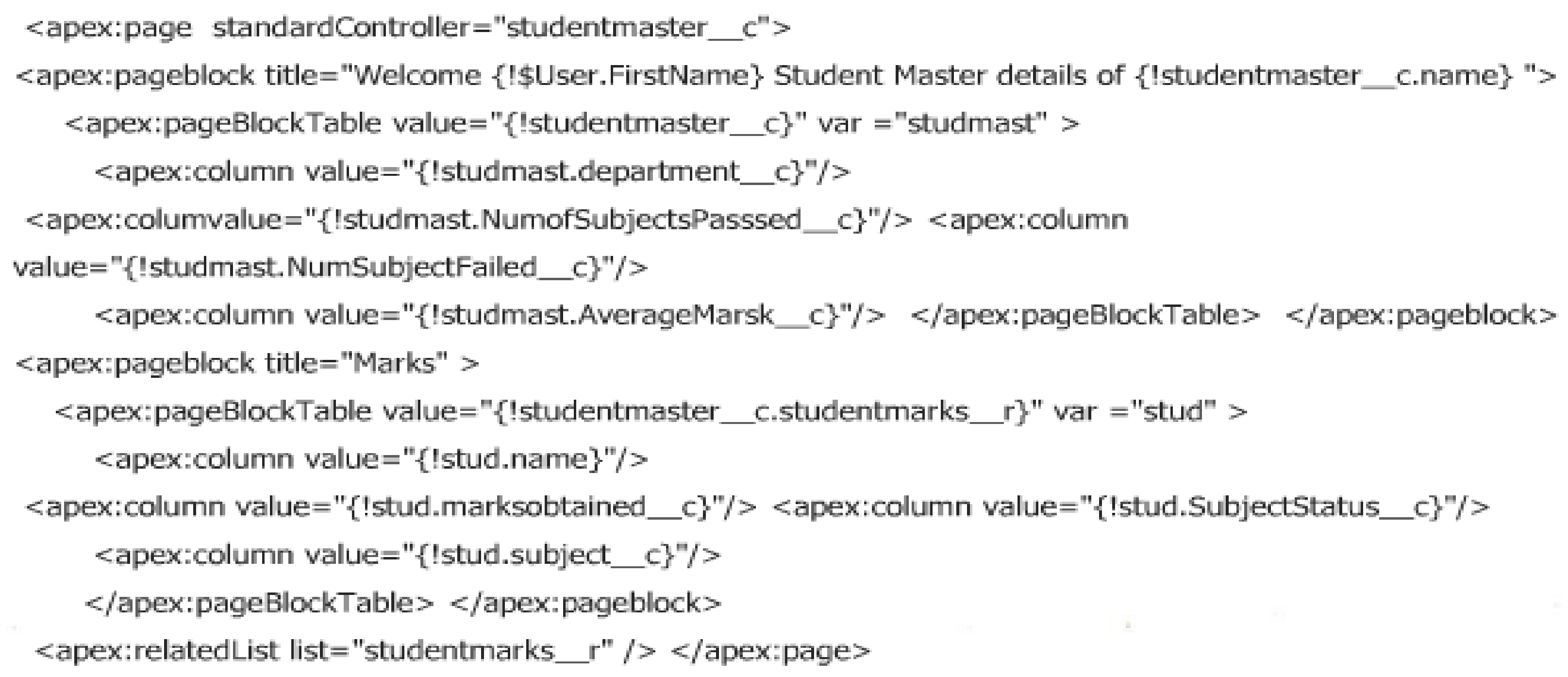
→ One is to display using related list

→ <apex:relatedList list="studentmarks__r" />

→ If you would like to use your own layout and display the fields from the detail object you can use the following approach

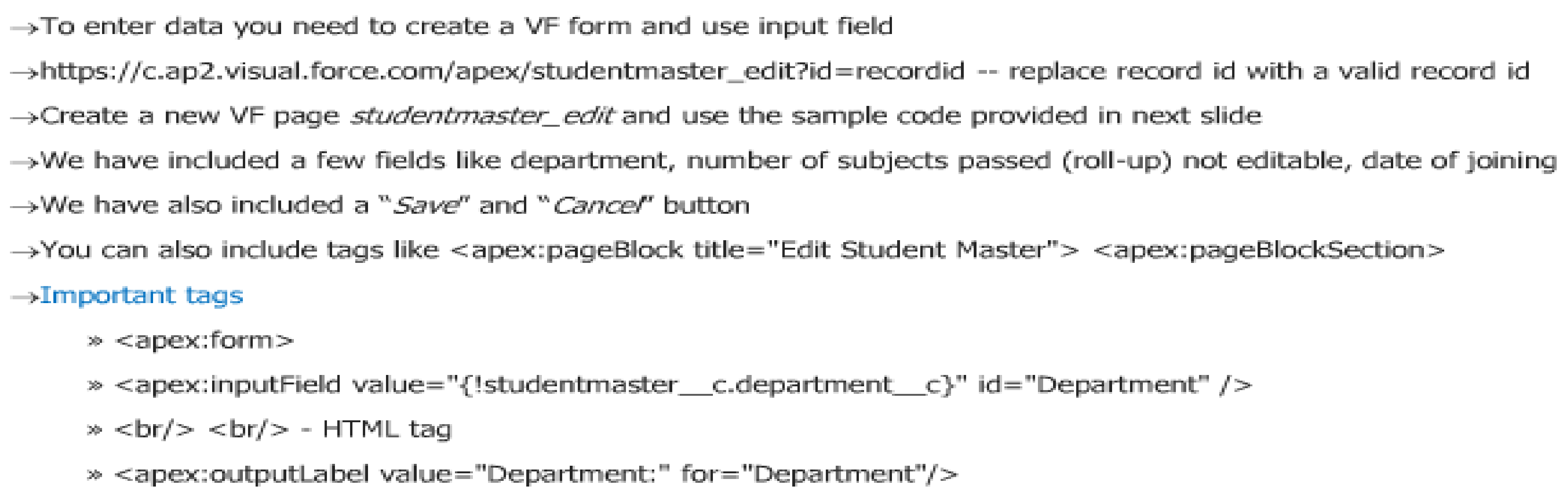
```
<apex:pageblock title="Marks" >
    <apex:pageBlockTable value="{!studentmaster__c.studentmarks__r}" var ="stud" >
        <apex:column value="{!stud.name}"/><apex:column value="{!stud.marksobtained__c}"/>
<apex:column value="{!stud.SubjectStatus__c}"/>
 <apex:column value="{!stud.subject__c}"/>
</apex:pageBlockTable>
</apex:pageblock
```

Examples contd.



```
 <apex:page  standardController="studentmaster__c">
<apex:pageblock title="Welcome {!$User.FirstName} Student Master details of {!studentmaster__c.name} ">
    <apex:pageBlockTable value="{!studentmaster__c}" var ="studmast" >
      <apex:column value="{!studmast.department__c}"/>
 <apex:columvalue="{!studmast.NumofSubjectsPasssed__c}"/> <apex:column
value="{!studmast.NumSubjectFailed__c}"/>
      <apex:column value="{!studmast.AverageMarsk__c}"/>  </apex:pageBlockTable>  </apex:pageblock>
<apex:pageblock title="Marks" >
   <apex:pageBlockTable value="{!studentmaster__c.studentmarks__r}" var ="stud" >
      <apex:column value="{!stud.name}"/>
 <apex:column value="{!stud.marksobtained__c}"/> <apex:column value="{!stud.SubjectStatus__c}"/>
      <apex:column value="{!stud.subject__c}"/>
     </apex:pageBlockTable> </apex:pageblock>
  <apex:relatedList list="studentmarks__r" /> </apex:page>
```

Example contd.



→To enter data you need to create a VF form and use input field

→https://c.ap2.visual.force.com/apex/studentmaster_edit?id=recordid -- replace record id with a valid record id

→Create a new VF page *studentmaster_edit* and use the sample code provided in next slide

→We have included a few fields like department, number of subjects passed (roll-up) not editable, date of joining

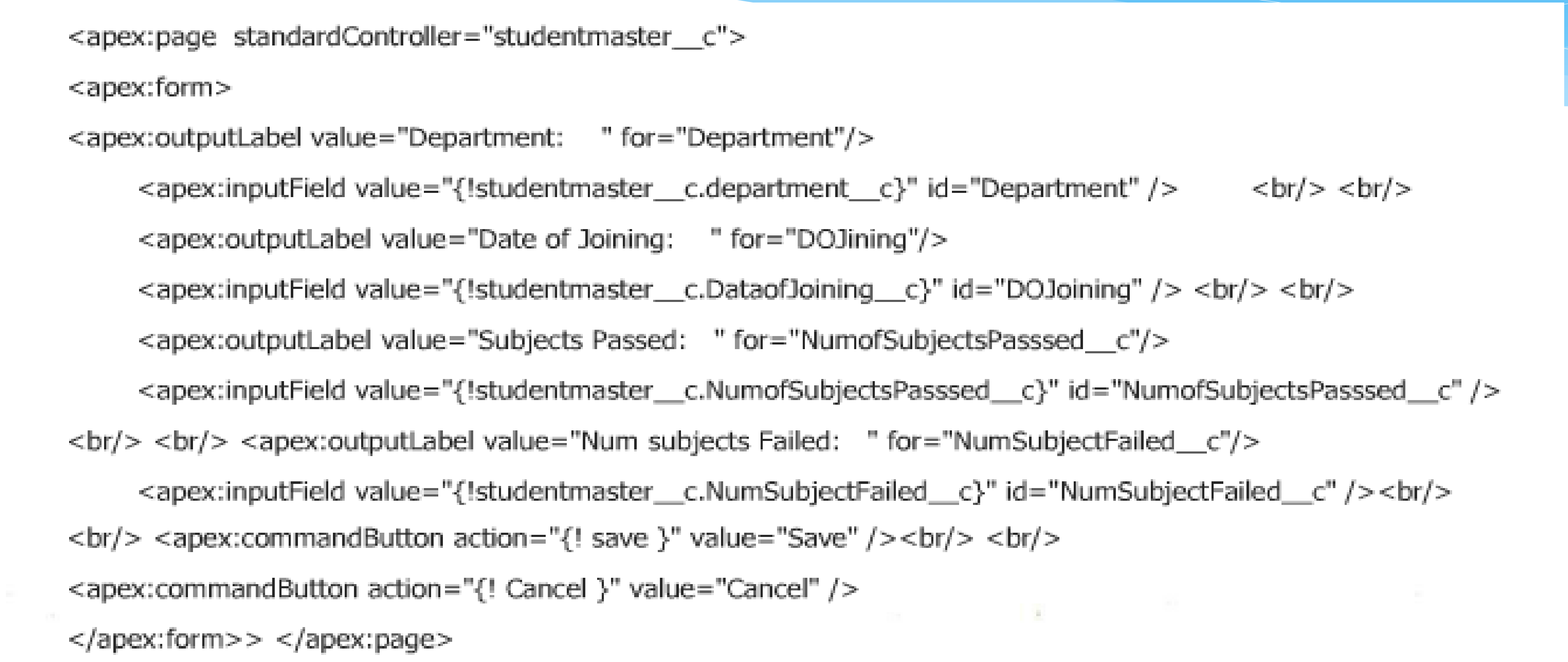
→We have also included a "*Save*" and "*Cancel*" button

→You can also include tags like <apex:pageBlock title="Edit Student Master"> <apex:pageBlockSection>

→Important tags

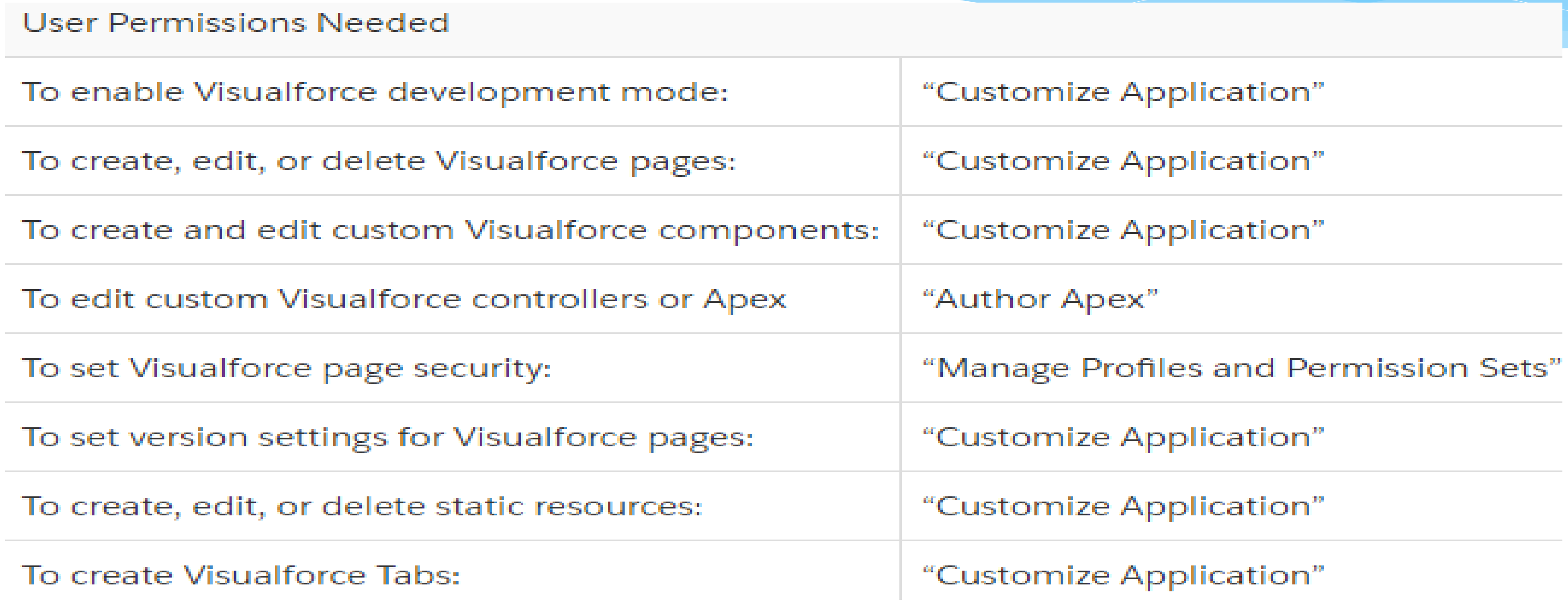
- » <apex:form>
- » <apex:inputField value="{!studentmaster__c.department__c}" id="Department" />
- » <br/> <br/> - HTML tag
- » <apex:outputLabel value="Department:" for="Department"/>

Examples contd



```
<apex:page  standardController="studentmaster__c">
<apex:form>
<apex:outputLabel value="Department:    " for="Department"/>
    <apex:inputField value="{!studentmaster__c.department__c}" id="Department" />        <br/> <br/>
    <apex:outputLabel value="Date of Joining:    " for="DOJining"/>
    <apex:inputField value="{!studentmaster__c.DataofJoining__c}" id="DOJoining" /> <br/> <br/>
    <apex:outputLabel value="Subjects Passed:   " for="NumofSubjectsPasssed__c"/>
    <apex:inputField value="{!studentmaster__c.NumofSubjectsPasssed__c}" id="NumofSubjectsPasssed__c" />
<br/> <br/> <apex:outputLabel value="Num subjects Failed:   " for="NumSubjectFailed__c"/>
    <apex:inputField value="{!studentmaster__c.NumSubjectFailed__c}" id="NumSubjectFailed__c" /><br/>
<br/> <apex:commandButton action="{! save }" value="Save" /><br/> <br/>
<apex:commandButton action="{! Cancel }" value="Cancel" />
</apex:form>> </apex:page>
```

Permissions required for VF



User Permissions Needed	
To enable Visualforce development mode:	“Customize Application”
To create, edit, or delete Visualforce pages:	“Customize Application”
To create and edit custom Visualforce components:	“Customize Application”
To edit custom Visualforce controllers or Apex	“Author Apex”
To set Visualforce page security:	“Manage Profiles and Permission Sets”
To set version settings for Visualforce pages:	“Customize Application”
To create, edit, or delete static resources:	“Customize Application”
To create Visualforce Tabs:	“Customize Application”

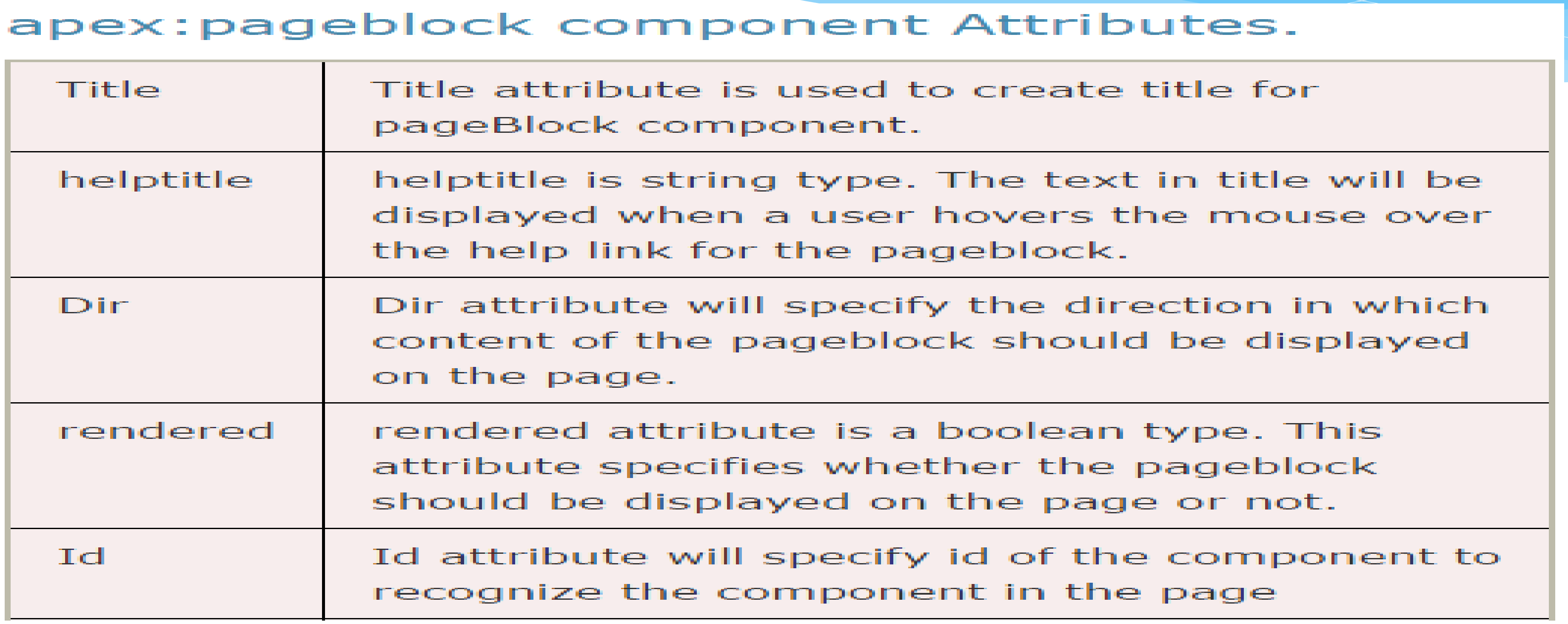
Standcontroller methods

* <apex:commandButton action=”{!list}” value=”List” />
* <apex:commandButton action=”{!view}” value=”View” />
* <apex:commandButton action=”{!cancel}” value=”Cancel” />
* <apex:commandButton action=”{!save}” value=”Save” />
* <apex:commandButton action=”{!quickSave}” value=”Quick Save” />no redirect
* <apex:commandButton action=”{!edit}” value=”Edit” />
* <apex:commandButton action=”{!delete}” value=”Delete” />

Record set var Labs

- Use recordsetvar to create listing of records using standard controller
- <apex:page standardcontroller =“Account” recordsetvar =“acclist”>
- Apex:pageblocktable value=“acclist” var=“acl”>

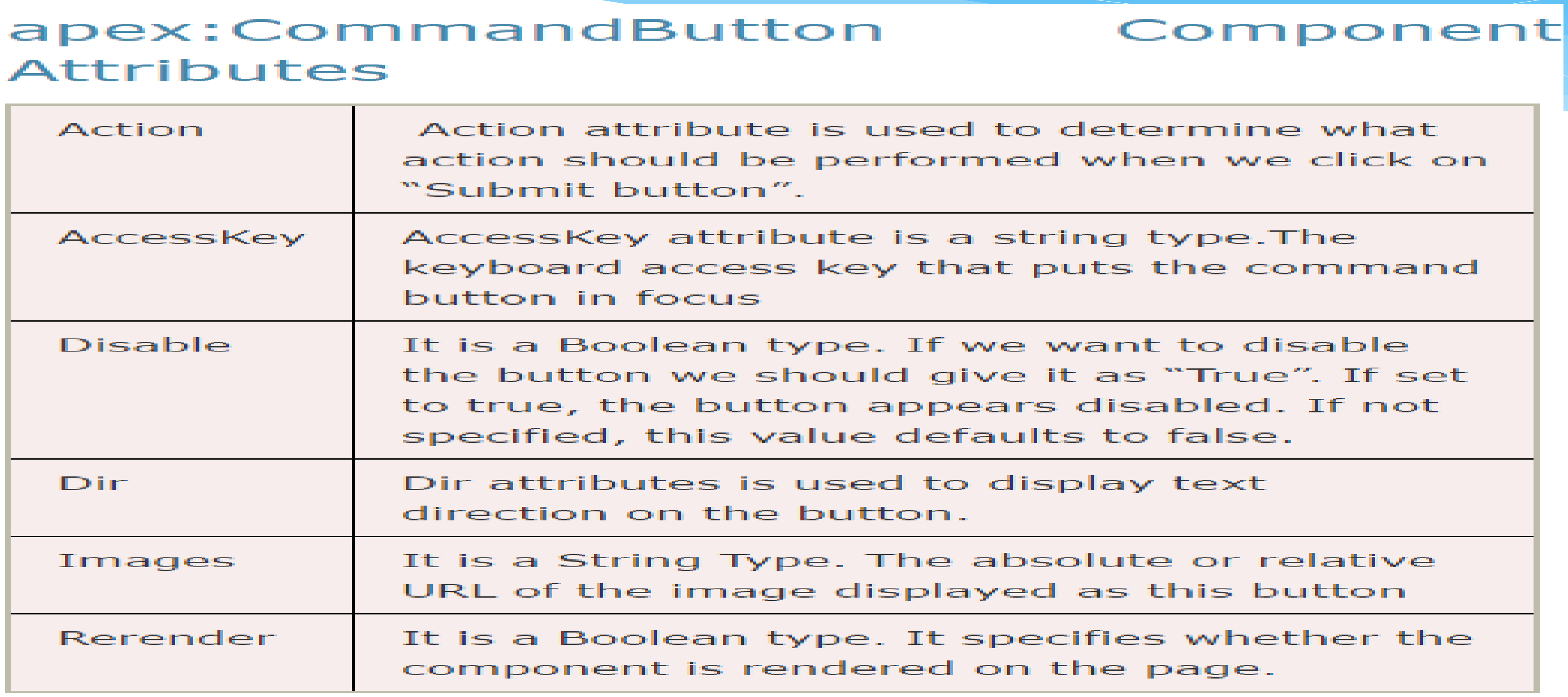
Pageblock



apex:pageblock component Attributes.

Title	Title attribute is used to create title for pageBlock component.
helptitle	helptitle is string type. The text in title will be displayed when a user hovers the mouse over the help link for the pageblock.
Dir	Dir attribute will specify the direction in which content of the pageblock should be displayed on the page.
rendered	rendered attribute is a boolean type. This attribute specifies whether the pageblock should be displayed on the page or not.
Id	Id attribute will specify id of the component to recognize the component in the page

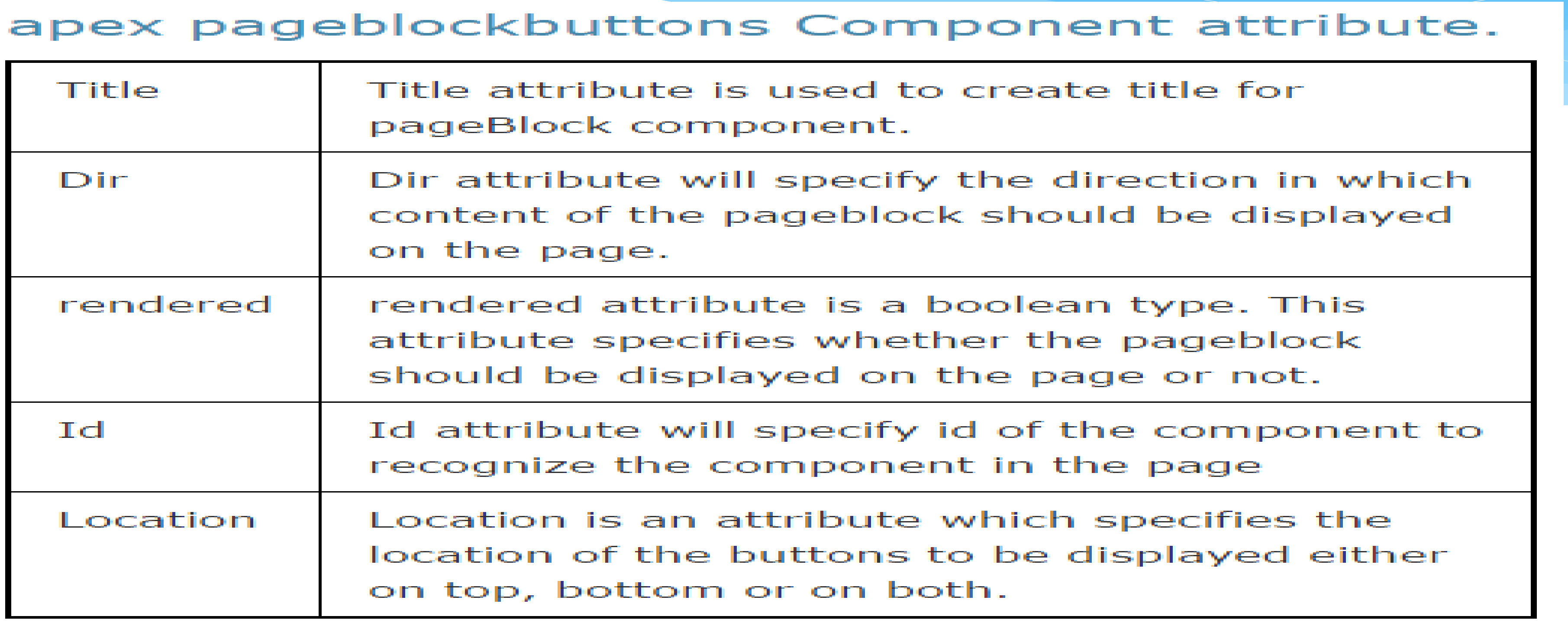
Command Button



apex:CommandButton Component Attributes

Action	Action attribute is used to determine what action should be performed when we click on "Submit button".
AccessKey	AccessKey attribute is a string type.The keyboard access key that puts the command button in focus
Disable	It is a Boolean type. If we want to disable the button we should give it as "True". If set to true, the button appears disabled. If not specified, this value defaults to false.
Dir	Dir attributes is used to display text direction on the button.
Images	It is a String Type. The absolute or relative URL of the image displayed as this button
Rerender	It is a Boolean type. It specifies whether the component is rendered on the page.

Pageblock Buttons



apex pageblockbuttons Component attribute.

Title	Title attribute is used to create title for pageBlock component.
Dir	Dir attribute will specify the direction in which content of the pageblock should be displayed on the page.
rendered	rendered attribute is a boolean type. This attribute specifies whether the pageblock should be displayed on the page or not.
Id	Id attribute will specify id of the component to recognize the component in the page
Location	Location is an attribute which specifies the location of the buttons to be displayed either on top, bottom or on both.

Additional variables

* Global action variable example:$action.account.new action will redirect to new tab
* value=”{!URLFOR(‘apex/CustomAccountPage’)
* Value=“URLFOR(‘/’+Contact.accountId)}””)
* Above needs actionlink apex tag
* {!$User.FirstName}
* subject="{!$CurrentPage.parameters.relatedId}"/> accessing url params in page
* In apex: WHERE Id = :ApexPages.currentPage(). getParameters(). get('id')

Variables contd..

- Today's date is: {!TODAY()}
- {!IF(AND(Price < 1, Quantity < 1), "Small", null)}
- **HTMLENCODE, JSENCODE**
- **Encodes text and merge field values for use in JavaScript by inserting escape characters, such as a backslash (\), before unsafe JavaScript characters, such as the apostrophe (')**
- **Custom Settings reference:** rendered="{!$Setup.App_Prefs__c.Show_Help_Content__c
- Look and feel detection:Theme3—Salesforce Classic 2010 user interface theme
- Theme4d—Modern "Lightning Experience" Salesforce theme
- rendered="{!$User.UITheme == 'Theme2'}
- <apex:outputLink value="{!$Page.otherPage}" link to another page
- Styleclass ="style1,style2 marker"

Labs

- Create block wise contact and student master editable vf pages
- Blocks should be rendered conditionally and should have ability to be
- Closed.