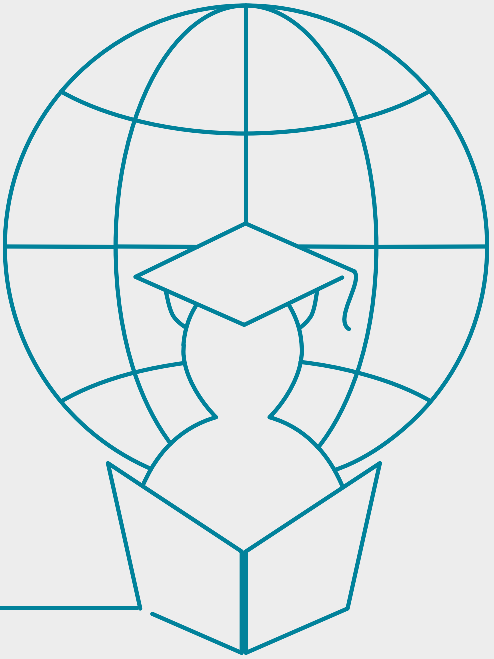
Force.com Visualforce Pages

Lesson 02 : Visualforce Components

Lesson Objectives

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

- Understand basic tag syntax and identify errors in sample pages
- State the commonalities between Visualforce and other tag/web language
- Construct expressions to include salesforce data into Visualforce pages



Components
What can go in a Visualforce page?

Visualforce includes a tag library similar to HTML and XML markup languages

You can include text directly into the Visualforce pages

You can use HTML tags within a Visualforce page, including:

- HTML comment tags <!– comments -- >
- Formatting tags

You can use Javascript within a Visualforce page as well, but you will often find it easier to use the equivalent Visualforce tags

- Note: You cannot use JavaScript line commenting within <script> tags

Components
Components are Well-formed XML

Like XML, Visualforce must be well-formed. At a high level this means:
Every Visualforce start tag *<tagName>* must have a matching end tag *</tagName>* or be a self contained tag *<tagName/>*
Tags are hierarchical and must be closed in the reverse order they were opened

```
    <tagName1>
      <tagName2>
      </tagName2>
   </tagName1>
```

Components
Components

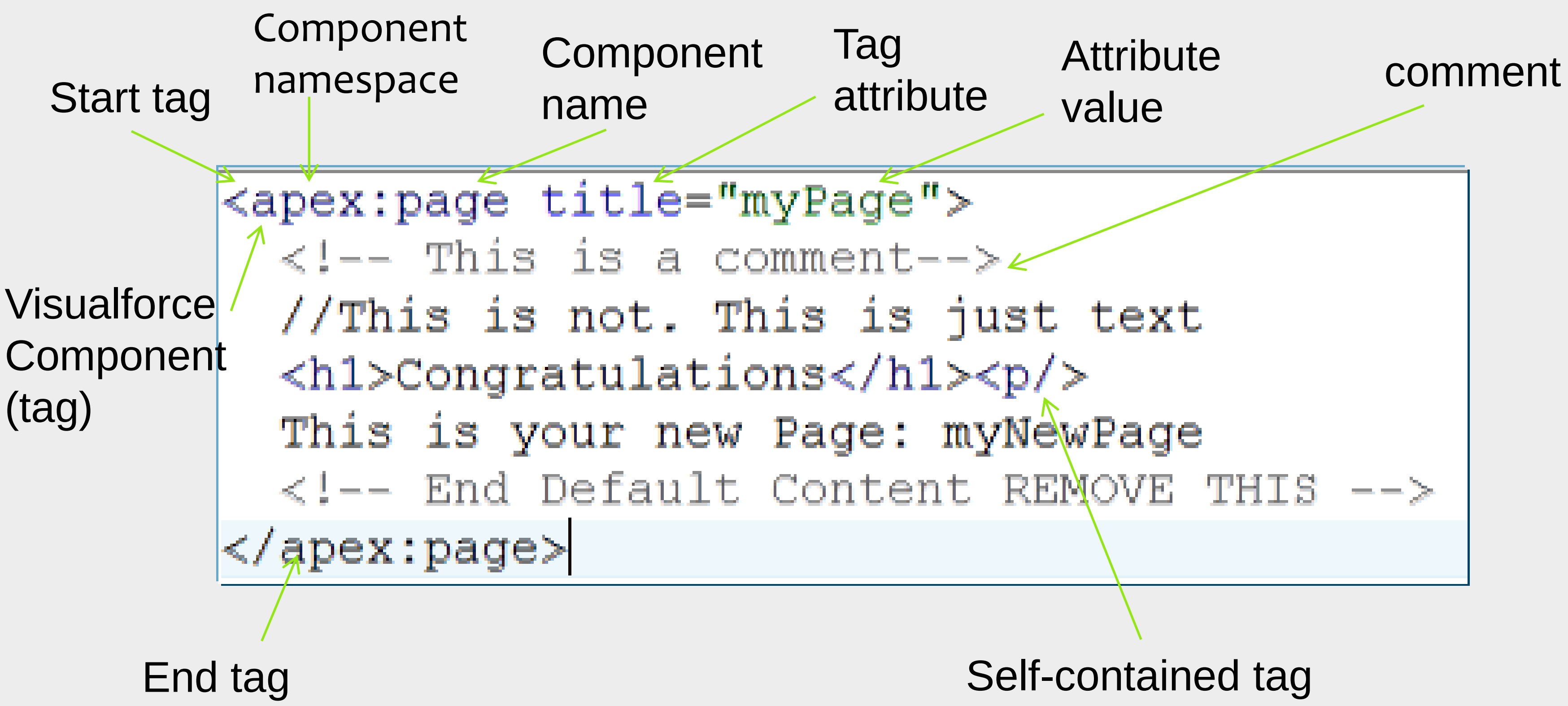
Most Visualforce components (tags) all begin with the apex: prefix

All pages must be enclosed by a set of <apex:page> tags

Tags may contain attributes that have values (in quotes) to help further define them

- Attribute values are typed to be strings, collections, IDs, etc.
- For a full list of tags and attributes, check the documentation

Components
Components Example



```
<apex:page title="myPage">
  <!-- This is a comment-->
  //This is not. This is just text
  <h1>Congratulations</h1><p/>
  This is your new Page: myNewPage
  <!-- End Default Content REMOVE THIS -->
</apex:page>
```

Component Bindings
Components Bindings

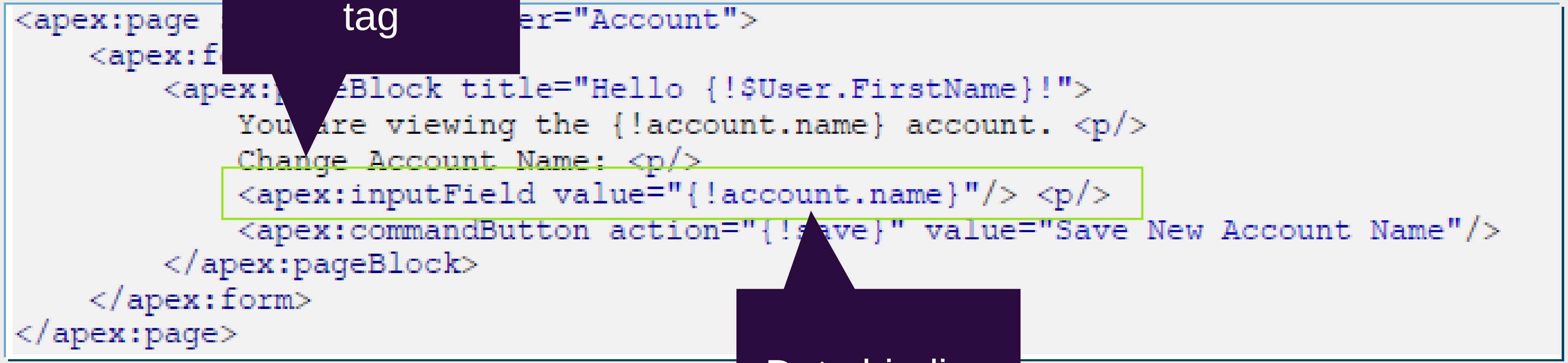
Bindings are ways to relate Visualforce components with either the page controller or other page components

There are primarily three types of binding for Visualforce components

- Data bindings: using the expression syntax to pull in the data from the data set made available by the page controller
- Action bindings: using the expression syntax to call action method for functions coded in the page controller
- Component bindings: using component attribute values to reference other components
  - These other components must have set a value of the id attribute

Component Bindings
Components Data Bindings

Visualforce component tag



```
<apex:page                  er="Account">
    <apex:f
        <apex:p    eBlock 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

Component Bindings
API Naming Syntax

Before discussing data binding in greater detail, it is important that you understand the API naming syntax

This information is available as a separate resource and should have been completed before coming to class

Tags refer to objects and fields by their API names

- Custom objects and fields use the __c suffix
- Custom objects referenced through relationships use dot notation and the __r suffix

Component Bindings
Expression Syntax

Tags use the same expression syntax as formula fields and other areas of the application

Dynamic object data can be inserted using the *{!objectName.propertyName}* syntax

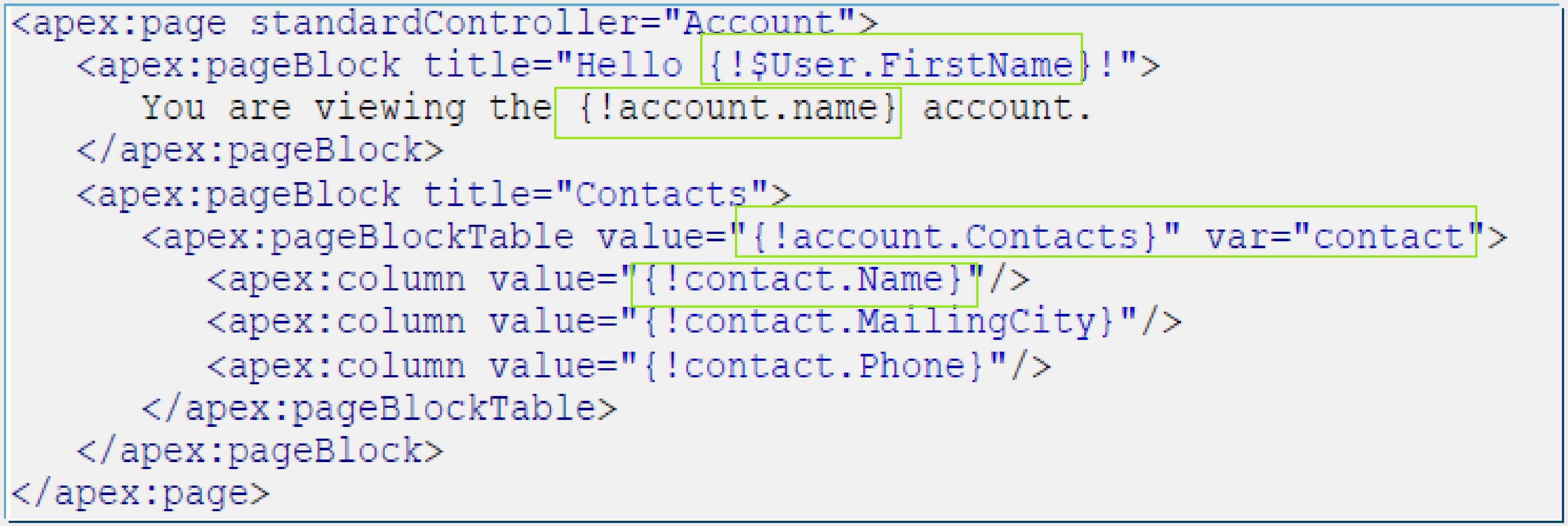
Global data can be inserted with the added $ syntax, such as:

- {!$user.fieldName }
- {!$Page.otherVisualforcePage }
- {!$Component.otherVisualforceComponent }
- Note that you can also access custom labels and translations using the $Label variable

Local variables can be created to stand in for these expressions, as they can become long and unwieldy using the <apex:variable> tag

Bindings can reference custom Apex class types as well as their inner classes using dot notation

Component Bindings
Component Data Binding Examples



```
<apex:page standardController="Account">
   <apex:pageBlock title="Hello {!$User.FirstName}!">
      You are viewing the {!account.name} account.
   </apex:pageBlock>
   <apex:pageBlock title="Contacts">
      <apex:pageBlockTable value="{!account.Contacts}" var="contact">
         <apex:column value="{!contact.Name}"/>
         <apex:column value="{!contact.MailingCity}"/>
         <apex:column value="{!contact.Phone}"/>
      </apex:pageBlockTable>
   </apex:pageBlock>
</apex:page>
```

Component Bindings
Action Binding

Data binding works because the page can access the data made available through the controller

In a similar manner, actions that are available through the controller can be called using the same expression syntax. These can be:

- Standard actions, such as save and edit
- Custom actions that provide custom functionality
  - More on this later in the discussion about controllers. Note that at this point, only actions that are shared across all objects are exposed through standard controllers

Dynamic Visualforce Components
What are Dynamic Visualforce Components?

Create a custom user interface driven by the user’s

- Permission
- Behavior
- Organization preferences
- Data attributes

Dynamic Visualforce Components
How will Dynamic Visuaforce Components be used?

Two users view the same page customized for their needs:
The Sales Engineer sees technical details
The Account Manager sees financial information

Demo

Creating a dynamic pages

Summary

Components are Well-formed XML

API Naming Syntax

Components Bindings

- Data Binding
- Action Binding
- Components Bindings

Module Review

- What kind of content can be included in a Visualforce page?
- What is the MVC pattern and how does it relate to Visualforce?
- The Data Loader is the only way to utilize the Bulk API

  TRUE OR FALSE