Controllers of a Visualforce Page

Share the information with your friends

The Controllers of a Visualforce Page are the logical part that we must know to create and manage quality applications.

Controller Types

Standard Drivers

- Standard controllers are used to manage **custom** and **default objects** .
- Contains the **functionality** and **logic** used by **standard Salesforce pages** .
- They are included in the **apex: page** tag as follows: **<apex: page standardController = ”Account”>**
- Its values can be accessed via force.com API as **{! Account.Name}**

```
<!-- Visualforce Page Example -->
<apex:page standardController="Account">
  Name: {! Account.Name }
</apex:page>
```

Custom Drivers

- They can be used to overwrite functionality, navigation and calls to web services.
- It uses its default **constructor** , without arguments, that is, **without parameters** .
- The methods of a standard controller can be **getters** , **setters,** and **actions** .

```
//MyCustom Controller
public class MyCustomController {
    Account acc;

    public MyCustomController () {
      acc = [SELECT Id, Name FROM Account
                WHERE Id =: ApexPages.currentPage().getParameters().get('id')];
    }

    public Account getAccount () {
      return acc;
    }
}
```

```
<!-- Visualforce page -->
<apex:page controller="MyCustomController">
  <apex:pageBlock>
    <apex:pageBlockSection>
      <apex:pageBlockTable value="{! account }" var="acc">
        <apex:column value="{! acc.Id }" />
        <apex:column value="{! acc.Name }"/>
      </apex:pageBlockTable>
    </apex:pageBlockSection>
  </apex:pageBlock>
</apex:page>
```

Getter methods

- Every **value calculated** and **to be displayed** in a visualforce page **must have** a **getter** method in the controller.
- The **name** of the getter methods should always be prefixed **get** at startup. For example: **getAccounts** .
- With the markup **{! name_of_expression}** you can **automatically connect** a visualforce page with the **getter** method . For Example: **{! accounts}** connects with the getter method **getAccounts.**
- The getter methods **do not have parameters** but they must **return an object** .
- They must have at least a **Public access level.**
- A getter method is **required** to **access the data** of an object from a controller to a visualforce page.
- A getter method can get data using **SOQL** queries .

```
//Getter example code
public class MyCustomController {
  Account acc;

  public Account getAccount () {
    return acc;
  }
}
```

Setter methods

- They are used to **send data** from a visualforce page **to the controller** .
- Setter methods are not required.
- These methods are **executed** automatically **before of** the methods **actions** .
- Your name must start with the prefix **set** . For example: **setVariable** , the **variable** value must be obtained from a visualforce page.

- They must have at least one **level of public access** .
- The setter method **has one parameter** and **does not return a value** .

```
//Setter example code
public class MyCustomController {
  Account acc;
  String accountName;

  public Account getAccount () {
    acc = [SELECT Id, Name FROM Account
           WHERE Id =: ApexPages.currentPage().getParameters().get('id')];
    return acc;
  }

  public String getAccountName () {
    return accountName;
  }
  //Setter Method
  public void setAccountName(String value) {
    accountName = value;
  }

  public PageReference save () {
    acc.Name = accountName;
    update acc;
    return null;
  }
}
```

```
<!-- Setter visualforce page example -->
<apex:page controller="MyCustomController">
  <apex:form>
    <apex:outputLabel value="Enter account: " />
    <apex:inputText value="{!accountName}" />
    <apex:commandButton value="Save" action="{!Save}" /> <p/>
    <apex:outputText value="Account Name entered: {!accountName}" />
  </apex:form>
</apex:page>
```

Properties for using Getter and Setter methods

- We can represent a method get and set the sentences **get** and **September** . For example: **public dataType propertyName {get; set;}**
- If a property has a single **get** statement , it is **read-only.**
- If a property has a single **set** statement , it is **write-only.**
- Properties with **get** and **set statements** are considered **read** and **write.**

```
public class MyCustomController {

  public Account acc {get; set;}

  public String accountName {get; set;}

  public String accountPhoneNumber {get;}

  public String accountFile {set;}

}
```

Driver Extensions

Controller extensions are classes in Apex that can **extend** the **functionality** of standard or custom controllers.

- We can call a controller extension using the **extension** keyword in the **<apex: page>** tag of our visualforce page.
- **Multiple** controller **extensions** can be used in a visualforce page by **separating them** with **commas** .
- They must have a **constructor** with a single argument of type **ApexPages.StandardController** to initialize their values.

```
public class MyControllerExtension {

  private final Account acc;

  public MyControllerExtension(ApexPages.StandardController stdController) {
    this.acc = (Account)stdController.getRecord();
  }

  public String getGreeting () {
    return 'This is a greeting message for: ' + this.acc.Name;
  }

}
```

```
<apex:page standardController="Account" extensions="MyControllerExtension">
  <h2>{!greeting}</h2>
  <p>
    <apex:form>
      <p><apex:inputField value="{!account.name}"/></p>
      <apex:commandButton value="Save" action="{!save}"/>
    </apex:form>
  </p>
</apex:page>
```