Custom Controller in Salesforce

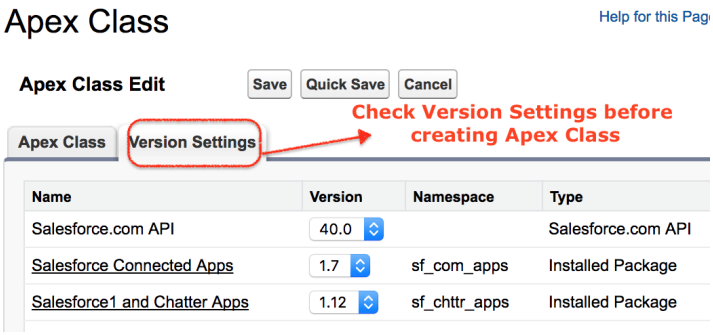
Custom controller is an Apex class which is used to implement all the logic of Visualforce without leveraging the standard functionality. In Salesforce.com, Standard controllers provides all the functionalities that are required for Visualforce pages. If we want to override the existing functionalities , then we can write our own Custom controller in Salesforce or a Controller extension using Apex class.

When Custom Controller in Salesforce is used?

- Custom Controller in Salesforce is used to override the built in functionalities of a standard controller.
- To add new actions.
- To build visualforce pages that respects user permissions.
- Custom controller and Controller extensions classes executes in system mode.
- User Permissions, field level security, Organization wide defaults, Role hierarchy and Sharing rules can be done using keyword called ***with sharing.***

How to build Custom Controller in Salesforce?

Custom Controller in Salesforce is an Apex Class. To create Custom controller navigate to**Setup | Build | develop | Apex Classes | New.**



- Click on Version settings before creating Custom controller in Salesforce. This version settings specify version of Apex and the API used.
- Now click on Apex class editor to write Apex class for Custom controller in Salesforce.
- In general, In single apex class we can add upto 1 million characters in length.

Apex Class for CustomController

```
public class
```

```
public class CustomController {

    private final Account account;

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

    public Account getAccount() {
        return account;
    }

    public PageReference save() {
        update account;
        return null;
    }
}
```

```
1  public class Customcontroller {
2
3      private final Account account;
4
5      public CustomController() {
6          account = [SELECT Id, Name, Site FROM Account
7                     WHERE Id = :ApexPages.currentPage().getParameters().get('id')];
8      }
9
10     public Account getAccount() {
11         return account;
12     }
13
14     public PageReference save() {
15         update account;
16         return null;
17     }
18 }
```

Controller name

- Customcontroller is the name of the controller.
- Click on Save button.

Apex Class

Customcontroller

« Back to List: Apex Classes

Apex Class Detail Edit | Delete | Download | Security | Show Dependencies

Name	Customcontroller	Status	Active
Namespace Prefix	tutorialkart	Code Coverage	0% (0/8)
Created By	Prasanth Kumar , 8/26/2017 3:53 AM	Last Modified By	Prasanth Kumar , 8/26/2017 3:53 AM

Class Body | Class Summary | Version Settings | Trace Flags

```
1  public class Customcontroller {
2
3      private final Account account;
4
5      public CustomController() {
6          account = [SELECT Id, Name, Site FROM Account
7                  WHERE Id = :ApexPages.currentPage().getParameters().get('id')];
8      }
9
10     public Account getAccount() {
11         return account;
12     }
13
14     public PageReference save() {
15         update account;
16         return null;
17     }
18 }
```

How to implement Custom controller in Visualforce pages.

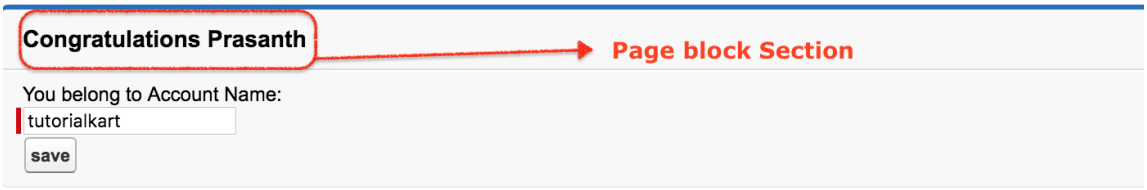
To implement Custom Controller in Visualforce pages, first we have to create new visualforce page in Salesforce.Follow the steps given below to implement custom controller.

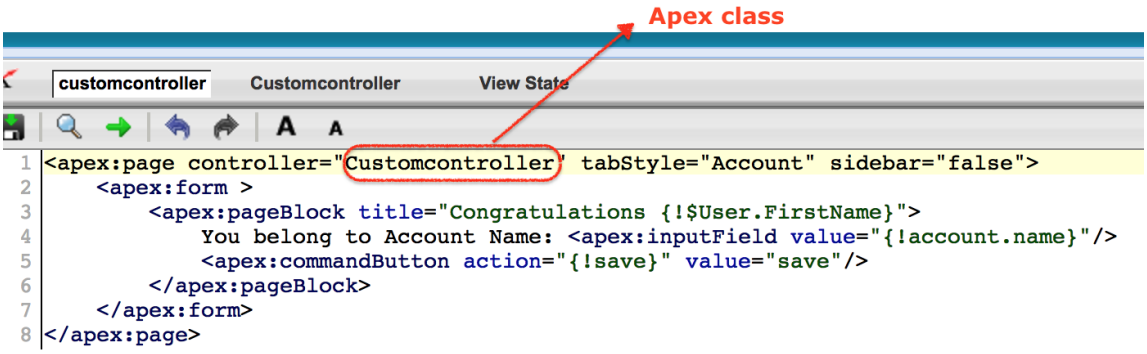
- Click on the link to create new Visualforce page in salesforce.

- In this visualforce page, we writing apex code to open Account in form. Account id must be added at the end of the URL
- https://tutorialkart-dev-ed–tutorialkart.ap4.visual.force.com/apex/customcontroller?**id=0016F000026m5ZB**

```
<apex:page
```

```
<apex:page controller="CustomController" tabStyle="Account">
    <apex:form>
        <apex:pageBlock title="Congratulations {!$User.FirstName}">
            You belong to Account Name: <apex:inputField value="{!account.name}"/>
            <apex:commandButton action="{!save}" value="save"/>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```





The Custom controller is associated with the page because of the *Controller* attribute of the **<apex:page>** component. The custom controller method in Salesforce must be referenced with {!}.

- **{!User.Firstname}** – This method retrieves the firstname of the User.

In above apex class, getAccount method is referenced by ***<apex:inputfield>*** tag in visualforce page and ***<apex:commandButton>*** tag references the save method with its ***action*** attribute.

- Finally click on Save.

In output section, If we rename the account name and Saved. The account name will be renamed with the new name.

Visualforce Tutorials.

- What is Visualforce?
- Difference between Salesforce.com, Force.com and Salesforce1.
- Why should we learn Visualforce?.
- Creating first visualforce page

Visualforce basics

- Building Visualforce page using MVC Model.
- Visualforce Standard Controllers
- Visualforce Custom Controllers
- Visualforce Controller extensions.
- Visualforce standard list controllers.

Visualforce components.

- apex:page.
- apex:pagemessage.
- apex:Form
- apex:pageblock.
- apex:Commandbutton.
- apex:pageblockbuttons.
- apex:pageblockSection.
- apex:pageblockSectionItem.
- apex:detail.
- apex:tab.
- apex:tabpanel.
- apex:panelbar
- apex:panelbarItem.
- apex:facetname

Visualforce - Input Components.

- apex:inputText.
- apex:inputSecret.
- apex:inputHidden.
- apex:inputCheckbox
- apex:inputTextArea.
- apex:selectList
- apex:selectoption
- apex:inputField
- apex:selectRadio
- apex:selectCheckBoxes

Enchancing your Organization.

Using Developer Console.

Unit Test