The controllers and extensions english language essay

Linguistics, English

A set of instructions that can react with the user interaction of Visualforce markup (like button click or link click) is called a controller. A controller can control the behavior of a page and that can be used to access the data which should display in the page. This chapter will introduce you to few types of controllers and extensions which can be used for Visualforce pages. We will understand the controller types with examples. The agenda for this chapter is as follows: Standard controllersStandard List ControllersCustom Controllers & Controller ExtensionsWorking with Large Sets of Data on Visualforce pageOrder of execution of a Visualforce pageValidation Rules and Standard Controllers / Custom ControllersUsing the transient KeywordConsiderations for Creating Custom Controllers and Controller ExtensionsLet's get closer to controllers & extensions….. This chapter includes a set of examples to explain important elements and features of Visualforce. Starting from this chapter we will build an Order Processing application. There are four custom objects (API Names : Customer__c, Item__c, Order__c, Order_Line__c) in this application. Following is the E-R diagram of Order processing application whish is going to build on Force. com platform. 9818_02_01. png

Standard Controllers

The Force. com platform provides few types of controllers. First one is Standard controllers and every sObject has a standard controller. They have the same logic and functionalities as they originally used in standard pages. Therefore we can use standard controllers with Visualforce pages. For a example, if we use Contact standard controller for a Visualforce page, we can implement the standard Save method for Contact without writing any additional Apex codes. That behaviour is same as the Save on standard Contact edit page.

How to use a Standard controller with Visualforce page

Above code is show the usage of standardController attribute. Note : You cannot use standardController and controller attributes at the same time.

Standard Controller Actions

In Visualforce pages, we can define action attribute in following standard visualforce components. An action method can be called from the page using the {!} notation. For a example, if your action methods name is " MyFirstMethod", then the you can use {! MyFirstMethod} notation for calling the action method from the page markup. Note : This action method can be from standard controller or custom controller or controller extension. A standard controller has few standard action methods as follows; save – Insert / update a record. Upon successful completion it will be redirected to the standard detail page or custom Visualforce page. quicksave – Insert / update a record. There is no any redirection to a detail page or a custom Visualforce page. edit – Navigates the user to the edit page with current record. Upon successful completion it will be returned to the page which invoked the action. delete – Delete the current record. Redirect to the list view page with selecting most recently viewed list filter. cancel – Cancels an edit operation. Upon successful completion it will be returned to the page which invoked the edit action. list - Redirect to the list view page with selecting most recently viewed list filter. For example, the following page allows us to insert a new customer or update a customer record. If we are going to use this page for updating a customer record, then the URL must be specified with the id query string parameter. Every standard controller has a getter method that returns the record specified by the id query string parameter in the page URL. When we click Save, the save action is triggered on the standard controller, and the customer is updated. If we are going to use this page to inserting a customer record, then the URL must not be specified with the id query string parameter. In this scenario, when we click the Save, the save action is triggered on the standard controller, and a new customer record is inserted. Note: The page markup allows you to access fields of a particular sObject using {! sObjectAPIName. FieldAPIName}. For example, if you want to access the Email field of Customer object, the page that uses the Customer__c standard controller can use {! Customer__c. Email__c} to return the value of Email field on the Customer which is in current context. The following page allows us to view a customer record. In this page also, the URL must be specified the id query string parameter. The getter method of Customer__c standard controller returns the record specified by the id query string parameter in the page URL. Note: To check the accessibility of particular object for the logged user, you can use {!$ObjectType. objectname. accessible} notation. This expression returns a Boolean value. For a example, if you want to check the accessibility of the Customer object, you can use {!$ObjectType. Customer__c. accessible}.

Standard List Controllers

Second controller type is Standard List Controller which can be used for displaying or performing a action on a set of records (including related lists, list pages and mass action pages). It allows us to filter records in particular page. We can use Standard list controllers for Account, Asset, Campaign, Case, Contact, Contract, Idea, Lead, Opportunity, Order, Product2, Solution, User, and all the Custom objects.

How to use Standard List Controller with Visualforce

Note: The standardController attribute specifies the type of records that we want to access. The recordSetVar attribute indicates that the page uses a list controller and the variable name (used to access data in the record collection) of the record collection. Following markup explains how the page can access list of records when the page is associated with a list controller. In the following example, you can refer list of customer records. Figure 2. 2 illustrates the result of following page. 9818_02_02. png

Standard List Controller Actions

All the standard visualforce components which have the action attribute can be used with a visualforce page with standard list controller. The usage of those components is same as standard controller. Following action methods are supported by all standard list controllers. save – Insert / update a record. Upon successful completion it will be redirected to the standard detail page or custom visualforce page. quicksave – Insert / update a record. There is no any redirection to a detail page or custom visualforce page. List – Redirect to the list view page with selecting most recently viewed list filter when the filterid is not specified by the user. cancel –. Cancels an edit operation. Upon successful completion it will be returned to the page which invoked the edit action. first - Displays the first page of records in the set. last - Displays the last page of records in the set. next - Displays the next page of records in the set. previous - Displays the previous page of records in the set. List views in salesforce standard pages can be used for filtering records which displayed on the page. For example, on the customer home page, you can select " start with c" view from the list view drop down and view customers whose name is start with letter " c". You can implement that functionality on a page associated with a list controller. Pagination can be added to a page associated with a list controller. The pagination feature allows you to implement the next and previous actions. For example, to create a simple list of customers with a list view and pagination, create a page with the following mark-up:{! a. name} 9818_02_02. png

Custom Controllers & Controller Extension

Custom Controllers and Controller extension are written using Apex..

What is a Custom Controller?

Custom Controllers are used to implement logics & functionalities without using a standard controller. Use custom controllers when you want to; Implement completely different functionality and don’t need rely on standard controller behaviour. Override existing functionalitiesMake new actions for the pageCustomize the navigationUse HTTP callouts or web servicesUse a wizardsHave greater control of information access on the pageRun your page without applying permissionNote : A page is allowed for only one controller.

Building a Custom Controller

You can build a custom controller in two ways. Via setup page – Your Name ☾ Setup ☾ Develop ☾ Apex Classes ☾ NewNote: You have the choice to write controller classes with sharing or without sharing keyword which is influence to run the particular page in system mode or user mode. 9818_02_04. pngFollowing class is an example of custom controllers. This custom controller has the functionalities for retrieving existing item list from Item__c custom object and adding a new item record. insertNewItem is the action method of ItemController. ExistingITems is a list of Item property which is used to retrieve existing item records. ExistingITems property has an override get method. public with sharing class ItemController {//public item property for new insertionpublic Item__c NewItem{get; set;}public ItemController(){NewItem = new Item__c();

}

//get existing items to show in a tablepublic List ExistingITems{get{ExistingITems = new List (); ExistingITems = [SELECT Id, Name, Item_Name__c, Unit_Price__c FROM Item__c LIMIT 100]; return ExistingITems;

}

set;

}

public PageReference insertNewItem() {try{insert NewItem;//reset public property for new insertNewItem = new Item__c();}catch(DmlException ex){ApexPages. addMessages(ex);

}

return null;

}

}

Note: A Custom controller uses no-argument constructor. You cannot create a constructor that includes parameters for a custom controller.

What is Controller Extension?

Controller Extensions are used to extend the logics & functionalities of a standard controller or a custom controller. A controller extension cannot be on a page without a standard controller or a custom controller. Use controller extensions when you want to; Keep majority of functionalities of a standard or custom controller as it is and add more functionalities. Build a Visualforce page which shoud run according to the user permissions.

Building a Controller Extension

We can build a controller extension using ways that used to build the custom controller. Note: An extensions cannot live by themselves on a page. It can be used on a Visualforce page with a custom controller or a standard controller. The following class is a simple example of a controller extension. This controller extension is used to extend the logics & functionalities of the

Order__c custom object’s standard controller. In this extension, we have a single argument constructor to fetch the order record from the standard controller. getRecord() is the method for fetching record from standard controller. prepareFullOrder() method is a custom method that implemented for query the order lines of particular order.

```
public with sharing class
OrderViewExtension{public Order__c CurrentOrder{get; set;}public List
OrderLines{get; set;}public OrderViewExtension(ApexPages.
StandardController controller){CurrentOrder = new Order__c(); this.
CurrentOrder = (Order__c)controller. getRecord(); prepareFullOrder();

}
public void prepareFullOrder(){OrderLines = new List (); OrderLines =
[SELECT Id, Name, Price__c, Item__c, Item__r. Unit_Price__c, Item__r.
Item_Name__c, Order__c, Quantity__c FROM Order_Line__c WHERE Order__c
=: this. CurrentOrder. Id];

}
}
```

Note: A controller extension uses one-argument constructor with the argument type of ApexPages. StandardController or a custom controller type. Following Visualforce page uses the above controller extension. On the page, we have page block with two sections. First section is shown us order header details. Second section is there to show the order lines of a particular order.

Controller Methods

There are three types of methods which can be used within a custom controller or a controller extension. Getter methodsSetter methodsAction Methods

Getter methods

Developers can use getter methods to display database or other computed values in Visualforce markup. That means, getter methods are used to pass data from Apex controllers to Visualforce page . There are two ways to define getter methods. Typically, getter methods are named as getVariable, where the variable is the name of the attribute which returns from the getter method. public class GetterSetterExample{String GetterVariable; public String getGetterVariable() {return GetterVariable;

}

}

A getter method can define the attribute with default getter and setter methods. public class GetterSetterExample{public String GetterVariableDefault{get; set;}

}

The variable can be accessed on Visualforce page with the {!} expression.

Setter Methods

Setter methods are used to pass user defined values to the Apex controller. Setter methods are defined in the same way as getter methods were defined. Following example uses default getter and setter methods for searching items which are already in the database. public with sharing class SearchItemController {public List ExistingItems{get; set;}public String Keyword{get; set;}public SearchItemController(){ExistingItems = new List ();

}

public void SearchItems(){ExistingItems = [SELECT Id, Name, Item_Name__c, Unit_Price__c FROM Item__c WHERE Item_Name__c LIKE: ('%'+Keyword+'%')];

}

}

Action Methods

Working with Large Sets of Data on Visualforce page

Note: Custom controllers and controller extensions are adhere to Apex governor limits. Visualforce provides " read only mode" feature to overcome the limit on number of rows which can be queried within one request and the limit on number of collection items which can be iterated on the page. There are two ways to set up the Visualforce read only mode.

Order of execution of a Visualforce page

A Visualforce page has a life cycle that is the life timebetween the page is created and destroyed during the user session. The life cycle is defined by the type of Visuaforce page request and the content of the page. There are two types of Visualforce page requests. Get requestPostback request

Order of Execution for Visualforce Page Get Requests

When we request a new page by entering a URL or by clicking button or a link, a get request is created. In the following diagram, it illustrates how a Visualforce page interacts with a custom controller or a controller extension during a get request. 9818_02_05. pngConstructor methods are called by initiating the controller objectsIf there are any custom components, they are created and constructor methods are called on associated class. If any attribute is specified in a component using an expression, expressions are also evaluated. Send the result HTML to the browser. If there are any client side technologies (such as JavaScript, CSS) , the browser executes them.

Order of Execution for Visualforce Page Postback Requests

Some user interactions (eg:- a save action triggered by user user's button click) require page updates, typically those page updates are performed by postback requests. Following diagram illustrates how a Visualforce page interacts with a custom controller or a controller extension during a postback request. 9818_02_06. pngThe view state is decoded and used as the basis for updating the values on the page during a postback request. Then expressions are evaluated and setters are executed. The action is executed, on the successful completion data is update. If the postback request redirects the user to the same page, the view state is updated. Send the results to the browserNote: If we want to execute an action without performing validations on the input or data changes on the page, we can use immediate attribute with true value for a particular component.

Validation Rules and Standard Controllers / Custom Controllers

Using the transient Keyword

The " transient" key word is used for declaring variables and used in Apex classes. Declaring variable as transient causes to reduce the view state size. Variables with transient keyword cannot be saved and it should not be transmitted as a part of the view state of the particular Visualforce page. Transient variables are needed only for the duration of a page request. The " transient" keyword is used in serializable Apex classes which mean classes that implement the Batchable or Schedulable interfaces. Following Apex objects are natively considered as transient;

PageReferenceXmlStreamClassesCollectionsMost objects generated by system methods such as Schema. getGlobalDescribeStatic variablesJSONParser class instancesFollowing example has a transient datatime variable and non transient datatime variable. This example shows the main feature of transient variables. That is cannot be saved and should not be a part of the view state. When we click on the Refresh button, transient date will be recreated but the non-transient date will be continue with its original value. Non Transient Date: {! t1}

Transient Date : {! t2}

public with sharing class TransientExampleController {DateTime t1; transient DateTime t2; public String getT1() {if (t1 == null) t1 = System. now(); return '' + t1;

}

public String getT2() {if (t2 == null) t2 = System. now(); return '' + t2;

}

Considerations for Creating Custom Controllers and Controller Extensions

When we are creating custom controllers and controller extensions, keep following consideration in your mind. Most important thing to keep in your mind is Apex governor limits. Apex classes can be run in system mode and user mode by using respectively without sharing and with sharing. Sensitive data can be exposed in without sharing controllers. The webservice methods must be defined as global. Other all the methods are publicTry to access the database in less time by using sets, maps, or lists. It will increase the efficiency of your code. Apex methods and variables are not instantiated in a guaranteed order. You cannot implement Data Manipulation Language (DML) in constructor method of a controllerYou cannot define @future annotation for any getter method, setter method, or constructor method of a controller. Primitive data types (String, Integer, etc) are passed by value and non-primitive Apex data types (list, maps, set, sObject, etc)are passed by reference to component’s controller