Scenario2:- Recursive triggers error.

To explain about recursive trigger.

```
trigger Recursive on Account(after insert, after update)
    {
    //if (Trigger.new, flag)
      if(Trigger.isBefore && Trigger.isInsert)
        {
    Account acc = [select id, name from Account where
                   name = 'Madhuri' limit 1];
      acc.name = 'Y Madhuri';
    update acc;
    }
  if(Trigger.isAfter && Trigger.isUpdate)
      {
    Account a = new Account(name = 'satish');
    insert a;
    }
```

→ the above trigger would generate Recursive triggers error as when you try to update one account record it will call after update trigger on Account object, which would

→ ~~When~~ you call insert new record into Account object. which inturn again call After update trigger on Account object. This would continue recursively.

TO avoid Recursive trigger error we will use Static Variable.

```
global class TrigerVar
    {
    global static boolean flag = false;
        }
```

Trigger:-

```
trigger Recursive on Account (after insert, after update)
    {
  if(TrigerVar.flag != true)
    {
   if(Trigger.isBefore && Trigger.isInsert)
      {
       TrigerVar.flag = true;
  Account acc = [select id, name from Account where
                  name = 'madhuri' limit 1];
     acc.name = 'y Madhuri';
       update acc;
    }
```

```
if(Triggers.isAfter && Triggers.isUpdate)
    {
  TrigersVar.flag = true;
  Account a = new Account (name = 'satish');
   insert a;
     }
}
```

→ In this case the triggers operation will be executed only when flag value will be false. once the triggers is executed for the first time the flag values are set to true.

NOTE:- The flag variable should be a static variable.

Q:- can we call batch apex in the triggers?

Ans:- yes, But we can call batch apex from the triggers. But Salesforce allows only 5 batch jobs in the queue. so if the triggers raises an error if more than 5 jobs.

Q:- can we call callouts from the triggers?

Ans:- We can not call the callouts directly from the triggers, instead we will define callouts in the future annotated methods & this future annotated methods can be called from the triggers.

NOTE:- We can only call 10 callouts in a single transaction.

Order of Execution in triggers :-

When we save a record with an Insert update or upsert Statement salesforce performs the events in order because when the event happens, order of execution is very important because they are multiple things field to a single event and when the event gets fired we need to know which processor is running first and which processes is running last.

The order of execution is in the following order.

1. Executes all before triggers.
2. System validations are performed.
3. Custom validations are performed.
4. Saves the record but does not Commit.
5. Executes all after triggers.
6. Executes assignment rules.
7. Executes auto response rules.
8. Executes workflow rules.
9. If the record was updated with workflow field updates, fires before and after triggers are more time in addition to Standard validations. Custom validations rules are not ran again.

10. Executes escalation rules.

11. Commits to the database.

Trigger Context Variables :-

There are some implicit variables that are defined within a trigger. That allows the developers to access run time context.

→ these variables are defined in System.Trigger class.

1) Trigger.New :- this stores list of new records of sobjects which we are trying to insert into database & list of records with which we are going to update.

→ this variable is available in Triggers which perform insert events & update events.

→ To.

2) Trigger.NewMap :- This is a map of IDs as keys the new versions of the sobject records as values.

Map<ID, SObject>

These will be available only in after insert, after update and before update triggers.

3) Trigger.old :- Returns a list of the old versions of the Sobject records. (This will store the list of old records with old values what it has before the operation).

this is availabe in before update, after update, before delete, after delete.

NOTE:- We can only perform read operation on Trigger.old records.

Trigger.oldMap :- A map of IDs to the old versions of the Sobject records.

Syntax:- Map<ID, SObject>

Note that this map is only available in update and delete triggers.

Trigger.isExecuting :- Returns true if the current Context for the Apex code is trigger, not a Visualforce page, a Webservice or an executeanonymous() API call.

Trigger.isInsert :- Returns true if this trigger was fired due to an insert operation, from the Salesforce user interface, Apex, or the API.

Trigger.isUpdate :- Returns true if this trigger was fired due to an update operation, from the Salesforce user interface, Apex or the API.

Trigger.isDelete :- Returns true if this trigger was fired due to a delete operation, from the Salesforce user interface, Apex or the API.

Triggers :-

Scenarios :-

Scenario1 :-

Create "Sales Rep" field with data type (Text) on the Account object. When we create the Account record, the Account owner will be automatically added to sales Rep field. When we update the Account owner of the record, then also the sales Rep will be automatically updated.

```
trigger UpdateSalesRep on Account (before insert, Before update)
    {
  Set<Id> setAccOwner = new Set<Id>();

    for (Account Acc : trigger.New)
        {
         setAccOwner.add (Acc.OwnerId);
        }

Map<Id, User> user_map = new Map<Id, User>([select Name
                         from User where id in : setAccOwner]);

     for (Account Acc : Trigger.New)
          {

     User usr = user_map.get (Acc.OwnerId);
       Acc.Sales_Rep1__c = usr.Name;
          }
    }
```

Scenario2:-

Create the field called "Contact Relationship" checkbox on the Contact object and Create the object called "contact Relationship" which is related list to the Contacts. (Lookup Relationship).

Now logic is when we Create Contact by checking Contact Relationship checkbox, then Contact Relationship will be created automatically for that contact.

```
trigger CreateCRonContactCreation on Contact (after insert)
    {
    if (trigger.isAfter)
        {
        if (trigger.isInsert)
            {
        ContactMastersHandler ConIns = New ContactMastersHandler();
        ConIns.CreateContactRelationshipByContact (trigger.New);
            }
        }
```

Class:-

```
public class ContactMastersHandler
    {
Public void createContactRelationshipByContact (list<Contact>
                                                list_Contacts)
```

```
{
List<Contact_Relationship__c> conList = new List<Contact_Relationship__c>
                                                    ();
for(Contact newconts : List_Contacts)
    {
    if(newConts.Contact_Relationship__c == true)
        {
    Contact_Relationship__c CR = new Contact_Relationship__c();
    CR.Name = newConts.lastName;
    CR.Contact__c = newConts.id;
    conList.add(CR);
        }
}
insert conList;
}
```

Scenario3 :-

When we change the owner of the Contact Relationship, then the owner name will be automatically populated in the Contact Relationship Name field.

Trigger :-

```
trigger ContactRelationshipMasterTrigger on Contact_Relationship__c
                                                (before update)
    {
    if(trigger.isBefore)
        {
        if(trigger.isUpdate)
            {
            //call the handler for the before update trigger event
    updateCROwnerName ConRelUpd = new updateCROwnerName();
    conRelUpd.updateContactRelationshipNameByOwner(trigger.New);
            }
        }
    }
```

Class :-

```
Public class updateCROwnerName {
public void updateContactRelationshipNameByOwner(
                    list<Contact_Relationship__c> cont_Rel)
```

```
    {
    map<Id,Id> map_Id_OWN = new map<id,id>();
    map<Id,String> map_id_Name = new map<id,String>();
    set<id> Idset = new set<id>();
for(Contact_Relationship__c list_recs : cont_Rel)
    {
     Idset.add(list_recs.Ownesid);
    }
list<users> u = [select id,Name from users where id in :Idset];
    for(users list_users : u)
    {
    map_id_Name.put(list_users.Id,list_users.Name);
    }
    if(u!=null && u.size()>0)
    {
for(Contact_Relationship__c list_recs : cont Rel)
    {
    if(list_recs.Ownesid!=null)
    {
    list_recs.Name = map_id_Name.get(list_recs.Ownesid);
    }
    }
    }
}
```

Scenario 4 :-

Create the field called "Contact Relationship" checkbox on the Contact object and create the object called "Contact Relationship" which is related list to the Contacts. (Look up Relationship).

Triggers Scenario 12 logic will says that when we create contact by checking Contact Relationship checkbox, then Contact Relationship will be created automatically for that Contact.

Now, this logic will for when we delete the Contact, then Contact Relationship will be deleted automatically.

```
trigger DeleteCRonContactDelction on Contact (before delete)
{
if(trigger.isBefore)
{
if(trigger.isDelete)
{
for(Contact c: Trigger.old)
{
for(Contact c: Trigger.old)
{
Contact_relationship__C CR = new Contact_relationship__C();
```

```
Contact_relationship__c CR = new Contact_relationship__c();

CR = [select Id from Contact_Relationship__c where
        Contact__c where Contact_c IN : GlobalUtility.
                getUniqueIds(Trigger.old)];

  delete CR;
    }
  }
}
```

Global utility class :-

```
public static set<Id> getUniqueIds(list<SObject> sobs)
  {
  set<Id> Ids = new set<Id>();
  for(SObject sob : sobs)
    {
    Ids.add(sob.Id);
    }
  return Ids;
  }
```

Scenario 5 :-

Create the field called "Contact Relationship" checkbox on the Contact Object and Create the object called "Contact Relationship" which is related list to the Contacts. (Lookup Relationship).

→ Triggers scenario 14 will says that when we delete the Contact, then Contact Relationship will be deleted automatically.

. Now the logic is when we undelete the Contact, then Contact Relationship will be undeleted automatically.

Trigger :-

```
trigger unDeleteCRonContactUnDeletion on Contact (After
                                                  undelete)
{
  if(trigger.isUndelete)
  {
// call the handler for the after undelete trigger event

ContactMastersHandler_undelete conIns = New
                                    ContactMastersHandler_undelete();

conIns.undeleteContactRelationshipByContact(Trigger.New);

  }

}
```

Class :-

```
public class ContactMastersHandler_undelete
  {
public void undeleteContactRelationshipByContact(list<Contact>
                                        list_contacts)
    {
  Set<Id> ContactIds = New Set<Id>();
  if(list_Contacts!=null && list_contacts.size()>0)
    {
list<Contact_Relationship__c> list_conRels = new
                                list<Contact_Relationship__c>();

list_conRels = [select id from Contact_Relationship__c where
          isDeleted=TRUE and Contact__c in :
          GlobalUtility.getUniqueIds(list_contacts);
    undelete list_conRels;
    }
```

Scenario6 :-

Create field called "Count of Contacts" On Account object. when we add the Contacts for that Account then Count will populate in the field on Account details page. When we delete the Contacts for that Account, then count will update automatically.

NOTE:- the above logic will be applicable when we have look up relationship. But when we have the Master-Detail relationship, then we can create Rollup summary field to get the Count of child records using "Count" function.

```
trigger CountofContacts on Contacts (after insert, after delete)
    {
  set<id> accid = new set<id>();
  list<contact> contactlist = new list<contact>();
  list<contact> listcon = new list<contact>();
  list<account> acclist = new list<account>();
  list<account> listAcc = new list<account>();
  map<id,integer> mapCount = new map<id,integer>();
     if(trigger.isinsert)
        {
     for(Contact con : trigger.New)
           {
```

```
accid.add(con.accountid);
    }
  }
 if(trigger.isdelete)
    {
 for(Contact con:trigger.old)
       {
        accid.add(con.accountid);
          }
       }
acclist=[select id, name, accountid from Contact where
          accountid in:accid];
for(account acc:acclist)
   {
   listcon.clear();
 for(Contact c:contactlist)
     {
     if(c.accountid ==acc.id)
         {
          listcon.add(c);
   mapCount.put(c.accountid, listcon.size());
       }
   }
}
```

```
if(acclist.size()>0)
  {
 for(Account a:acclist)
    {
     if(mapCount.get(a.id)==null)
          a.count_of_contacts__c=0;
 else
 a.count_of_contacts__c = mapCount.get(a.id);
    listAcc.add(a);
    }
  }
 if (listAcc.size()>0)
  update listAcc;
  }
```

Future annotation :- (@future) :-

1. use the future annotation to specify that these methods that are executed asynchronously.
2. Methods with future annotation must be static methods.
3. Methods with future annotation can only return a void type.

Syntax :-
```
global class className
{
@future
Static void methodname(parameters)
{
//body of the method.
}
```

Eg :-
```
global class MyFutureclass
{
@future
Static void myMethod (String a, Integer i)
{
System.debug('Method called with: '+a+' and '+i);
//do callout, other long running code.
}
}
```

4. @future(callout = true), this indicates that this future methods can use callouts.

Eg:- @future (callout = true)

```
public static void docalloutfromfuture()
{
//Add code to perform callout
}
```

5. @future(callout = false) this is used to prevent a method from making callouts.

6. The parameters specified must be primitive datatypes, arrays of primitive datatypes & collections of primitive datatypes.

7. Methods with the future annotation can not take Sobjects or objects as arguments.

Imp:- How to pass Sobjects/Apex class object as parameters in the @future methods.

Step1:- First create Apex class Address.

```
public with sharing class Address
{
public String Street { set; get; }
public String city { set; get; }
```

```
public String State { set; get; }
public String zip { set; get; }

Public Address(String s, String c, String st, String z)
    {
    Street = s;
    city = c;
    State = st;
    zip = z;
    }
}
```

Step2:- create AddressFuture class.

1. With in the Constructor create object for Address class and serialize the objects.
2. create a future method callFuture(String) with String as parameter in the method deserialize the String to objects and use them.

```
Public with Sharing class AddressFuture
    {
    public AddressFuture()
        {
    List<String> addresses = new List<String>();
Address Helper ahl = new AddressHelper('1 base st',
                    'san Francisco', 'CA', '94105');
```

```
AddressHelper ah2 = new AddressHelper('2 here st',
                                      'sanfransisco', 'CA', '94105');
AddressHelper ah3 = new AddressHelper('3 here st',
                                      'san francisco', 'CA', '94105');
//serialize my objects ah1, ah2, ah3
    addresses.add(JSON.serialize(ah3));
    addresses.add(JSON.serialize(ah2));
    addresses.add(JSON.serialize(ah3));
    callfuture(addresses); // invoke the future method by passing
                           a string address where it a serialized form of
                           Apex class objects.
}
@future
static void callfuture(List<String> addressess)
{
    Address currAddress = null;
    for(String ser : addressess)
    {
        currAddress = (Address) JSON.deserialize(ser,
                                      AddressHelper.class);
```

System.debug('Deserialized in future:'+custAddress.street);

}

}

}

NOTE :-

→ No more than 10 method calls per Apex invocation.

→ you can not call a method annotated with future from a method that also has the future annotation, nor you can call triggers from an annotated method that calls another annotated method.

→ All asynchronous calls made after the startTest method are collected by the system. When StopTest is executed, all asynchronous processes are run synchronously.

Apex page sharing rules :-

To share a record programmatically we use Apex sharing (sharing record using Apex).

→ The sharing objects are named as ObjectName__Share. Where object name can be a name of custom object or standard object.

→ For standard objects Account.

Object Name	ShareObjectName
Account	AccountShare
Contact	ContactShare
Opportunity	OpportunityShare

→ For custom objects

ObjectName	ShareObjectName
Customers__c	Customers__Share
Student_details__c	Student_details__Share
Sample__c	Sample__Share

→ Share object includes record supporting all 3 types of sharing.

1) Force.com managed sharing
2) User managed sharing
3) Apex managed sharing.

→ Sharings are granted to the users implicitly by

i) organization wide default

ii) Role hierarchy

iii) View all & modify all permissions.

NOTE :- using share object we can not track view all data & modify all data permissions of the object.

→ Every share object has the following properties.

i) Access level :- this specifies level of access to the granted users & group on an object.

Access levels are

Edit
Read
All

NOTE :- This field must be set to an access level that is higher than organization default access level.

ii) parent Id :- Id of the object on which we are performing sharing.

iii) users & Group Id :- The users & Group id's to which you are granting access the grand group can be public group, role & territory.

iv) RowCause :- the reason why the users of group is being granted access.

Sharing Reason Field :-

i) Force.com Managed Sharing :- this includes sharing access granted by the force.com based on role hierarchy, record ownership and sharing rules.

Reason field value	RowCause Value
Account Sharing	Implicit child
Associated record owner of sharing	Implicit parent
Owner	Owner
Opportunity Team	Team
Sharing Rule	Rule
Territory Assignment Rule	Territory Rule.

ii) Users managed Sharing :- It is also called as manual sharing. This allows record owner of any user with full access to a record to share the record with a user or a group of users.

→ this is generally done by an end user for single record.

→ All the record owners and users above the owner, the role hierarchy is granted full access.

Reason field value	RowCause value
Manual sharing	Manual
Territory Manual	Territory Manual

iii) Apex managed sharing :- Apex managed sharing provides developers with the ability to support an application to share a requirements.

→ This type of sharing available to only users with modify all data permissions. only this users can add/change apex managed Sharing.

→ Apex managed sharing uses a sharing reason (Apex sharing reason).

Steps to create Apex sharing reason :-

1) setup → Build → create → objects.

2) select the custom object.

3) click New in Apex sharing reason related list.

4) Enter Label Name : This label is displayed in Reason column when viewing in user interface.

5) Enter name: This name is used when referencing the reason in API Apex programming.

Eg:- If Reason name is 'Test Sharing' it is referred in the Apex program as Schema.customObject__share.rowCause.

TestSharing__c.

Schema.Customer__share.rowCause.TestSharing__c

Schema.Student__share.rowCause.TestSharing__c

NOTE:- Apex sharing reasons are only available for custom objects.

```
public class JobSharing
{
public static boolean manualShareRead(Id recordId,
                                      Id userOrGroupId)

    Job__share jobshr = new Job__share();
    Set the ID of record being shared.
        obshr.parentId = recordId;
    //Set the Id of user & group being granted access
        obshr.parentId = recordId;
        obshr.userOrGroupId = userOrGroupId;
    //set the access level
        obshr.AccessLevel = 'Read';
```

```
//set rowCause to 'manual' for manual sharing.
jobshr.RowCause = Schema.Job__Share.RowCause.Manual;
Database.SaveResult sr = database.insert(jobshr, false);
    //process the saved result
        if (sr.isSuccess())
            {
        //Indicates Success
            return true;
                }
        else
            {
            return false;
                }
Database.Error err = sr.getErrors()[0];
if(err.getStatusCode() == StatusCode.FIELD_FILTER_VALIDATION_
                                    EXCEPTION &&
    err.getMessage().contains('AccessLevel'))
            {
        return true;
                }
    else
        {
        return false;
            }
```

→ the following triggers grants the recruiter and hiring_manager access when the job record is created. this example requires a custom object called job, with two lookup fields associated with users records called Hiring_Manager & Recruiter. Also, the job custom object should have two sharing reasons added called Hiring_Manager and Recruiter.

```
trigger JobApexSharing on Job__c (after insert)
    {
    //create if(trigger.isInsert)
        {
    //Create a new list of sharing objects for job.
    List<Job_share> jobShrs = new List<Job_share>();
    //Declare variables for recruiting and hiring managers sharing
        Job_share recruiterShr;
        Job_share hmShr;
    for(Job__c job : trigger.new)
        {
    //Instantiates the sharing objects
    recruiterShr = new Job__share();
    hmShr = new Job__share();
```

```
hmshr = new Job__Share();
//set the ID of the record being shared.
    recruitershr.parentId = job.Id;
hmshr.parentId = job.Id;
//set the ID of user & group being granted access.
recruitershr.UserOrGroupId = job.Recruiter__c;
hmshr.UserOrGroupId = job.Hiring_Manager__c;
    //set the access level.
recruitershr.AccessLevel = 'Edit';
    hmshr.AccessLevel = 'Read';
//set the Apex sharing reason for hiring manager and
recruiter.
recruitershr.RowCause = Schema.Job_share.RowCause.Recruiter__c;
hmshr.RowCause = Schema.Job__Share.RowCause.Hiring_
                                    Manager__c;
    //Add objects to list for insert
    jobShrs.add(recruitershr);
    jobshrs.add(hmshr);
        }
Database.saveResult[] lsr = Database.insert(jobshrs, false);
    }
```

Flows :-

When the process involves the number of steps when the data need to be transferred between the pages we preffered to use flows.

Navigation :-

Setup → Build → Create → Workflows → Flows → New Flow

We have

1. palettes
2. Resources
3. Explorer

Palettes :- We have

DRAFT TOOLS
USER INTERFACE
LOGIC
DATA

DRAFT TOOLS :- It's a place holder element which we can use to quickly sketch out a flow and convert into screen element.

Screen :- It's a user facing screen where we can take the inputs and give the outputs. When you click on the screen we are going to get

General Ifo
Add a field
Field settings

General Info :-

Name of screen:
Unique name of screen:
Navigation options: → No navigation restrictions
Don't show previous button
Don't show Finish button

Add a field :-

This is to add a fields to the screen. The type of fields are Input field, choice, multi-select Fields and outputs.

Input Fields :-

* Textbox
* Long Text Area
* Number
* Currency
* Date
* password
* checkbox

Textbox :- Drag the textbox from inputs and drop it into the viewScreen.

Double click on the textbox

Enter Label name, unique name* [ ]

Input type : Textbox

Default value : We can assign any default value to this field.

Required ☑ By checking this we can make the field required

Input validation

By clicking on Input validation we can write the validation rules.

Currency :- Add the currency field to the screen.

Label :

unique name* :

Input type :

Default value :

scale : This specifies how many decimal points we should have currency.

Required ☑

Similarly we can create the rest of the input fields given above.

Choice Fields :-

* Radio Buttons
* Dropdown list

Radio Buttons :- Add the Radio Button fields to the Screen.

Enter the label name : [ ]

unique Name* : [ ]

choice Type : Radio Buttons

Value DataType : We have an options like Text, Number, currency, Date, Boolean.

Required ☐

Default Type : [ ]

Creating choice

click on Create New

↓

choice

↓

Enter Label : [ ]

unique Name* : [ ]

value DataType : [ ] ▽

Stored value : When you select the choice what is it is going to return the value stored in the stored value

Show Input on selection ☑

When you select this you want to show some additional fields by checking this checkbox.

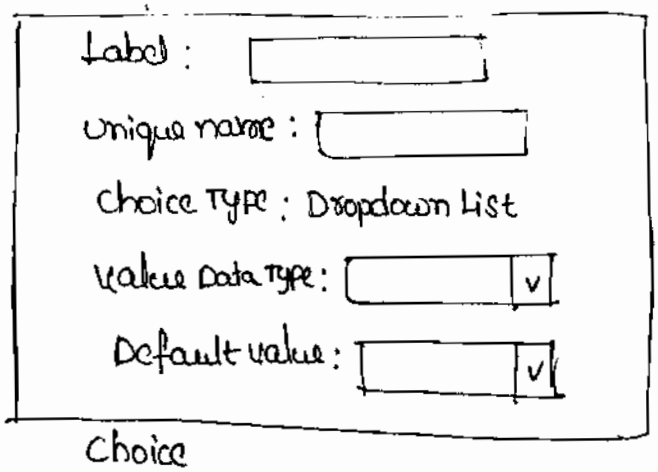
Whe we click on above checkbox we get

Input Field settings

Label : [ ]

Required ☑

Dropdown list :- Dropdown list is nothing but a picklist.

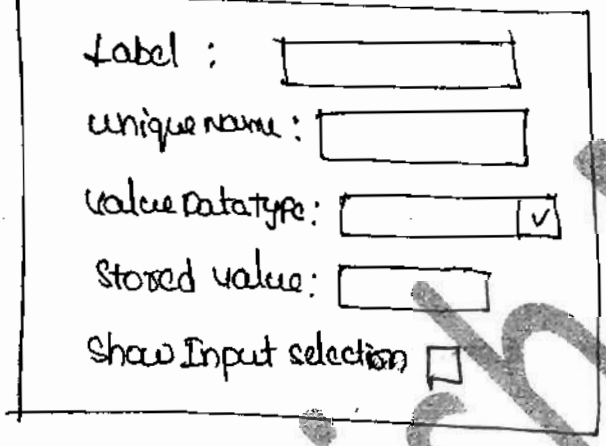


Choice

↓

Create New choice

↓

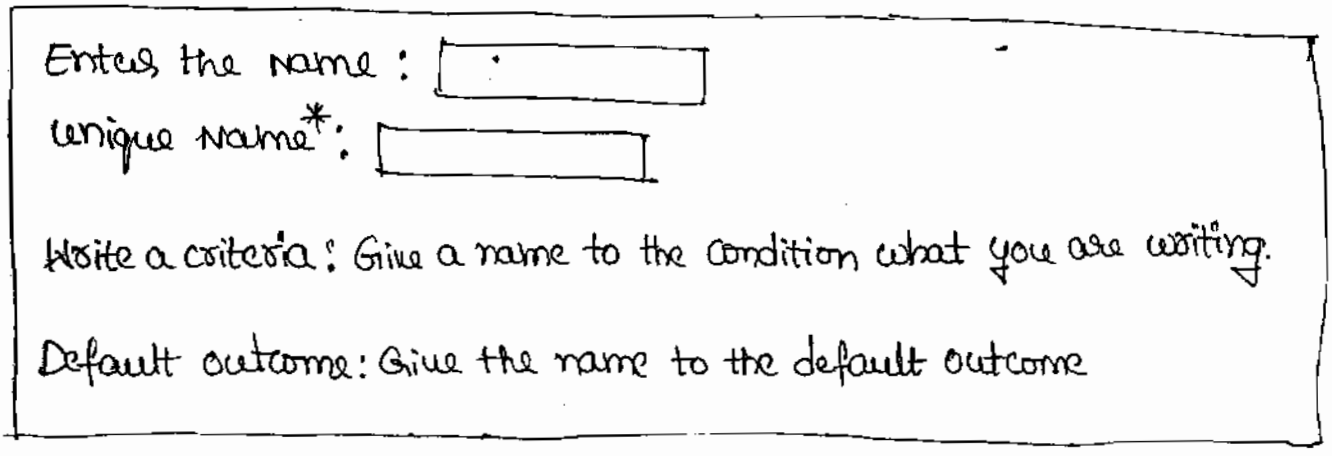


Logic :-

Decission :-

Writing the logical conditions to verify whether the data what we have entered is valid & invalid.

Navigating from one page to another page based on the input what we given.

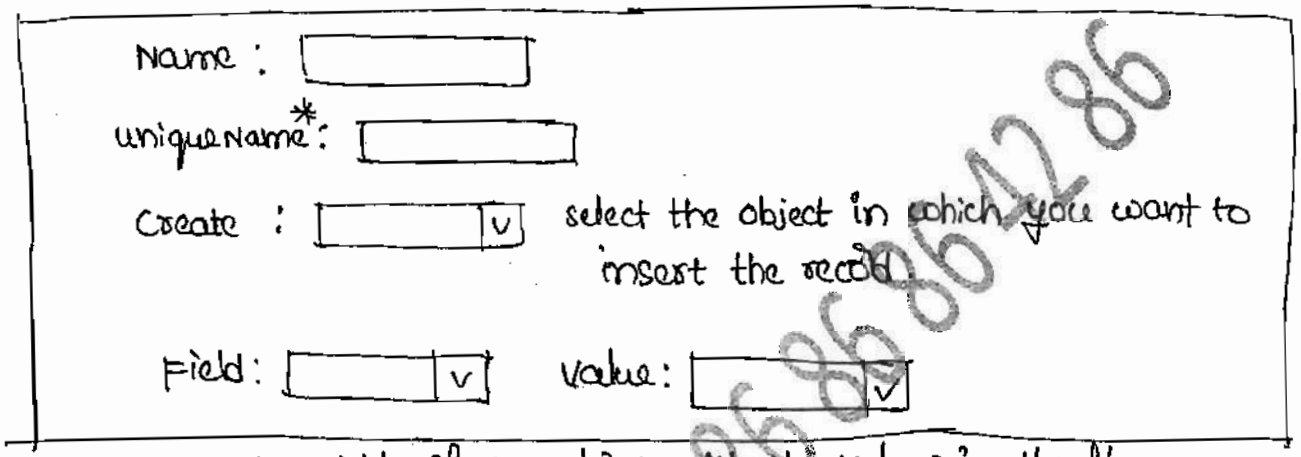


Database operations from visualforce :-

We can perform Record create, Record update, Lookup, delete operations on salesforce objects from visualforce.

Record create :- When you want to create a new record in the flow

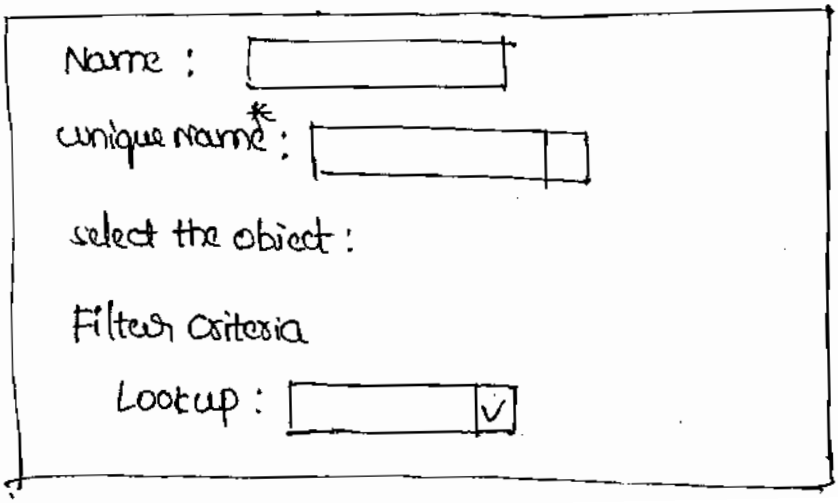
click on → Record create & Enter



once the record is successfully created it is going to return id of the record. store that id to variable so that we can refer to this record with this id in future context.

Record Lookup :- When you want to perform some seaarch operation and fetch the fields.

→ perform search on object based on the input values given in the flow and fetch the records and stored into a variable. So we can use this variables in the flow.



→ the records which are satisfying this filters are fetched and assigned to the variables.

Map the field and variable:

Lookup* [ ]

Field	operator	Variable value
[ ]	[ ]	[ ]

Record update:- Give the filter criteria and select the object the records which are satisfies the condition are fetched now map the fields and values.

Record delete:- select the object then give the filter criteria. The records which are satisfying the condition are deleted.

Adding flows to visualforce page:-

```
<apex:page>
<apex:sectionHeader title = "user login" subtitle = "verification"/>
<flow:interview name = "usertest" ></flow:interview>
</apex:page>
```

process.plugin:-

process.plugin is a built-in interface that allows you to pass data between your organization and specified flow.

The process.plugin interface contains two methods. They are

describe &
invoke

Any class which is implementing this interface should define this two methods.

1)
global
* Process.pluginResult invoke (Process.pluginRequest request)
{
}

→ This is the first method which will be invoked by the system when we implement the plugin.interface.
→ This method is used to perform the business logic what we want to implement.

2) global process.pluginDescribeResult describe()
{
}

→ this method defines what type of inputs the plugin requires how many inputs plugin requires and how many outputs its going to return.

Process.pluginDescribeResult :-

→ This class is used to determine the input parameters and output parameters needed by process.pluginResult.
→ The process.plugin interface describe() dynamically provides both input and output parameters for the flow. That's why the method returns process.pluginDescribeResult.

Properties of process.pluginDescribeResult :-

Description :- This is an optional field which will describe about purpose of the plugin.
We can take upto 255 characters.

Name :- It's the name given for a plugin.
The length can be 40 characters.

Input parameters :- It's a list of input parameters passed by the process.pluginRequest class from the flow to the class that implements process.plugin interface.

Syntax :- List<process.pluginDescribeResult.Inputparameter>Inputparameters

Describing the process.describeResult.Inputparameters :-

process.pluginDescribeResult.Inputparameter one = new

process.pluginDescribeResult.Inputparameters ('name', optionalDescription,

process.pluginDescribeResult.parameterType.Enum, Boolean - required)

name :- Name of the input parameter.

Description :- This is an optional field where you can give the description of field.

parameterType :- This gives the datatype of the parameter.

Required :- If it is true input value is required.

Creating the input parameters

Process.pluginDescribeResult result = new process.pluginDescribeResult();

result.Inputparameters = new List<process.pluginDescribeResult.Inputparameter>

{

new Process.pluginDescribeResult.Inputparameter ('sname', 'Student name',

process.pluginDescribeResult.parameters.String, true),

new process.pluginDescribeResult.Inputparameters ('Age', 'studentAge',

process.pluginDescribeResult.parameters.Decimal, true)

};

Process.pluginDescribeResult.parameterType can take

Boolean
Date
Date & time
Decimal
Double
Float
Id
Integer
Long
String

NOTE:- By default every currency field is a decimal field.

Output parameters :- It's a list of output parameters passed by the process.pluginResult class to the flow.

Describe process.pluginDescribeResult.outputParameters :-

```
process.pluginDescribeResult.outputParameters one = new
    process.pluginDescribeResult.outputParameters (name, Description,
                                process.pluginDescribeResult.parameterType)

process.pluginDescribeResult result = new process.pluginDescribeResult();
result.outputParameterss = new List<process.pluginDescribeResult.outputParameters
        {
    new process.pluginDescribeResult.outputParameters ('Myval', 'this is result',
                                process.pluginDescribeResult.parameterType.String)
        };
```

Process.pluginRequest class :-

process.pluginRequest class process the input parameters from the class that implements the interface to the flow.

Syntax :-

```
process.pluginRequest(Map(String,object))
```

This going to pass the map object to the flow.

Reading the parameter values from the flow to the plugin :-

```
invoke (Process.pluginRequest request)
        {
datatype  var = (datatype)request.inputparameters.get(parametername);
        }
```

Process.pluginResult :-

This class returns the output parameters from the class that implements interface to the flow.

```
Map <String, String> m = new Map<string, String>();
        m.put ('value1', 'Sam');
        m.put ('value2', 'Ram');
Process.pluginResult pr = new process.pluginResult (m);
Process.pluginResult (Map<String, object>)
process.pluginResult (String, object)
```

We can add all the output values that we want to return it to the flow, from plugin in the below form.

```
Map<String, String> m = new Map <String, String>();
        m.put ('value1', 'Sam');
        m.put ('value2', 'Ram');
```

Write a plugin program to read the values of subject1 & subject2 from the flow and calculate the total and return the result as Result :-

```
global class satyaFlow implements process.plugin
    {
    global process.pluginResult invoke(process.pluginRequest request)
        {
        Integer m1 = (Integer)request.inputparameters.get('sub1');
        Integer m2 = (Integer)request.inputparameters.get('sub2');
        Integer c = m1+m2;
    Map<String, Integer> output = new Map<String, Integer>();
        output.put('total', c);
    process.pluginResult result = new process.pluginResult(output);
        return result;
        }
global process.pluginDescribeResult describe()
    {
    process.pluginDescribeResult result = new process.pluginDescribeResult();
result.Inputparameters = new List<process.pluginDescribeResult.Inputparamete
                                                        >{
    new process.pluginDescribeResult.Inputparameters('sub1',
                    process.pluginDescribeResult.parameterType.Integer, true)
new process.pluginDescribeResult.Inputparameters('sub2',
                    Process.pluginDescribeResult.parameterType.Integer, true)
    };
```

```
result.outputparameterss = new List<process.pluginDescribeResult.
                                    outputparameter>
                {
    new process.pluginDescribeResult.outputparameters('total',
        process.pluginDescribeResult.parameterstype.Integer, true)
                }
            }
        }
```

Replace "Decimal" in the place of "Integer" in the above program.

Passing the values from visualforce to flow :-

When you want to pass the values from visualforce to flows use the tag called <apex:param> in between

```
<apex:page>
  <flow:interview name="usertest">
   <apex:param value="12" name="one"/>
   <apex:param value="14" name="two"/>
  </flow:interview>
</apex:page>
```

Overwrite the finish button :-

When you click on the finish button of the flow specify to which page you want to navigate back.

Error :-

An unhandled fault error

this is cause when datatype mismatches.

Plugin program :-

```
global class satyaflow implements process.plugin
        {
global process.pluginResult invoke(process.pluginRequest request)
            {
Decimal m1 = (Decimal) request.inputparameters.get('sub1');
Decimal m2 = (Decimal) request.inputparameters.get('sub2');
 Decimal c = m1+m2;
Map <String, Decimal> output = new Map <String, Decimal> ();
   output.put('total', c);
Process.pluginResult result = new process.pluginResult (output);
    return result;
            }
global process.pluginDescribeResult describe()
            {
  Process.pluginDescribeResult result = new process.pluginDescribeResult();
result.Inputparameters = new List < process.pluginDescribeResult.
                                    Inputparameter > {
 new process.pluginDescribeResult.Inputparameter('sub1',
                process.pluginDescribeResult.parameterType.Decimal, true),
 new process.pluginDescribeResult.Inputparameter('sub2',
                process.pluginDescribeResult.parameterType.Decimal, true)
                    };
result.outputparameters = new list < process.pluginDescribeResult.
                                    outputparameter >
                {
```

```
new process.pluginDescribeResult.outputparameter('total',
        process.pluginDescribeResult.parameterType.Decimal)
        };
    return result;
    }
}
```

Selectoption :-

Selectoption object specifies one of the possible values for visualforce select checkboxes, selectlist, select radio Component.

It Consist of label that is displayed to the end user, and the value that will be written to the Controller.

NOTE:- The Selectoption can be displayed in disable state.

Constructors :-

We have 2 Constructors in this class.

Instantiating

```
Selectoption one = new Selectoption(value, label, isDisabled)
                                    ↓       ↓        ↓
                                  String  String   String
```

If isDisabled is true, the option is disabled we can not select.

```
Selectoption one = new Selectoption(value, label)
                                    ↓       ↓
                                  String  String
```

Methods :-

1. String getlabel()

   This method returns the label of the option displayed to the user.

   Eg:- one.getlabel();
   O/P:- one

2. String getValue()

   This method returns the value of the object.

   Eg:- one.getValue()
   O/P:- one

3. void setDisabled(boolean)

   This will set the value of isDisabled attribute in the selectoption.

   Eg:- one.setDisabled(false);

4. Boolean getDisabled()

   This will return the isDisabled attribute value.

5. void setLabel (String labelname)
This will set the value for the given label.

6. void setValue

SelectOption Example program:-

selectExample :-

```
Public class SelectExample {
 String[] countries = new String[] {};
 Public pageReference test() {
      return null;
       }
 public List<selectoption> getItems() {
 List<Selectoption> options = new List<Selectoption>();
   options.add (new Selectoption('US', 'US'));
   options.add (new SelectOption('CANADA', 'canada'));
    options.add (new SelectOption ('MEXICO', 'Mexico'));
       return options;
        }
   public String[] getCountries() {
       return Countries;
         }
   public void setCountries (String[] countries) {
          this.countries = Countries;
              }
        }
```

```
<apex:page controller = "SelectExample" >
<apex:form>
```

```
<apex: selectCheckboxes value="{!countries}"/>
    <apex:selectoptions value="{!items}"/>
</apex: selectCheckboxes> <bo/>
<apex: commandButton value="Test" action="{!test}" rerender="out"
                                                    Status="Status"/>
</apex:form>
<apex:outputpanel id="out">
        <apex:outputpanel>
            <p> you have selected: </p>
<apex: datalist value="{!countries}" var="c">
                    {!c}
    </apex:datalist>
    </apex:outputpanel>
    </apex:outputpanel>
</apex:page>
```

Schema programming:-

Schema gives the meta data information about the data (Object, fields).

Schema methods:-

1) Map<String, SchemaSObjectType> getGlobalDescribe()

this method returns a map of all the Sobject names as a keys and Sobject Val tokens as values.

Write a program to display all the list of objects available in the Salesforce Organization in the visualforce page:-

```
public class DescribeExample {
  Public List<selectoption> options;
  public List<Selectoption> getoptions()
        {
      return options;
        }
  public DescribeExample()
        {
   Options = new List<Selectoption>();
  Map<String, Schema.SObjectType> m = Schema.getGlobalDescribe();
                                  Set<String> obj = m.keySet();
    for(String s: obj)
        {
     Selectoption op = new Selectoption(s,s);
        options.add(op);
          }
    }
}
```

Visualforce page :-

```
<apex:page controller="schema" "DescribeExample">
 <apex:form>
 <apex:selectList size="1">
<apex:selectoptions value="{!options}"/>
 </apex:selectList>
</apex:form>
</apex:page>
```

Schema.DescribeSObjectResult :-

This describes sobject methods returns an array of DescribeSobjectResult Objects. Where each object has the following properties.

String name : This is the name of the object.

String label : label text for the tab of an object.

String labelplural : label text for an object that represents plural version of object name.

String keyprefix : Object 'id's are prefixed with 3 character quotes that specify the type of object.

Eg :- Account obj has 001, opportunity object has 006.

When we call the key prefix it will return 3 character prefix code for the object which we called.

Field[] fields :- This will return array of the fields associated with an object.

Boolean Custom :- Indicates whether the object is a custom object or not.

Boolean Creatable :- Indicates whether the object can be created via create method.

Boolean deletable :- Indicates whether the object can be deleted or not.

Boolean mergable :- Indicates whether the object can be merged with objects of it's type.

Replicatable :- Indicates whether the object can be replicated via getUpdate function & getDeleted.

Boolean Retrievable :- Indicates whether the object can be retrieved using retrieve method or not.

Boolean Searachable :- Indicates whether object can be searched via Search method.

Search layoutable :- Indicates whether layout information can be retrieved via DescribeSearch layouts or not.

Boolean triggerable :- verifies whether the object can be triggered or not.

Boolean Updatable :- Checks whether the object can be updatable or not.

TO fetch the properties of an object

```
Schema.DescribeSObjectResult res = Account.SobjectType.getDescribe();
                    (or)
Schema.DesribeSObjectResult result = Schema.SObjectType.Account;
```

Example program :-

Field Example (Apex class) :-

```
public class FieldExample {
    public FieldExample()
    {
Schema.DescribeSobjectResult res = Account.SObjectType.getDescribe();
    result = '' + res;
    }
public String result {get; set;}
}
```

In the above program 'res' contains description about the object Account

→ If you want to know the properties individually we can use

```
String getLabel();
String getKeyKeyprefix();
String getLabelplural();
```

```
String getName();
Boolean isAccessable();
Boolean isCreatable();
Boolean isCustom();
Boolean isCustomSetting();
Boolean isDeletable();
Boolean isDeprecated&Hidden();
Boolean isMergable();
Boolean isQueryable();
Boolean isSearchable();
Boolean isUndeletable();
Boolean isUpdatable();
```

Child Relationship methods for given object :-

```
List<Schema.ChildRelationship> getChildRelationships();
```

If an SObject is a parent object we can access the child relationships as well as the child objects using childRelationship object.

→ The above method returns the list of child objects for the given SObject.

List of child records :-

Write a program to display list of child objects for a given object :-

```
public class FieldExample {
    List<selectoption> options;
public list<selectoption> getOptions()
        {
        return options;
        }
public FieldExample()
        {
```

```
Options = new List <SelectOption>();
Schema.DescribeSobjectResult R = Account.SobjectType.getDescribe();
List<Schema.childRelationship> C = R.getchildRelationships();
    for(schema.childRelationship X:C)
        {
      String name = ' '+X.getchildObject();
      Selectoption op = new Selectoption(name,name);
       Options.add(op);
            }
    }
}
```

```
<apex:page controller = "FieldExample">
 <apex:form>
 <apex:selectList size="1">
 <apex:selectoption value="{!options}"> </apex:Selectoption>
 </apex:selectlist>
 </apex:form>
</apex:page>
```

Recordtype info:-

List<Schema.RecordTypeInfo>.getRecordTypeInfos();

this returns a list of record types created on this object. Users may or may not have access can see this record type information.

Write a program to display list of record types for a given object:-

```
Public class FieldExample {
  Public List<Selectoption> options;
```

```
public List<selectoption> getOptions()
        {
        return options;
        }
public fieldExample()
        {
 options = new List<selectoption>();
  String myobie = 'Account';
Schema.DescribeSobjectResult R = one_c.SobjectType.getDescribe();
List<Schema.RecordTypeInfo> RT = R.getRecordTypeInfos();
  for(Schema.RecordTypeInfo x : RT)
        {
  Selectoption op = new Selectoption(x.getName(), x.getName());
        options.add(op);
        }
}
```

RecordTypeInfoByID:-

Map<Id, Schema.RecordTypeInfo>.getRecordTypeInfosByID();

Returns the Map, matches the records id's of an object to their associated record types.

Write a program to display Id's of Record types of a given object:-

```
Public fieldExample() {
    options = new List<selectoption>();
Schema.DescribeSobjectResult d = Schema.SobjectType.one_c;
. Map<Id,Schema.RecordTypeInfo> rtMapById = d.getRecordTypeInfosById();

    Set<Id> S = rtMapById.keySet();
      for(Id x:s) {
```

```
Selectoption op = new Selectoption(x,x);
    Options.add(op);
        }
```

Methods in the RecordTypeInfo class :-

* String getName();
  Returns the name of the record type
* Id getRecordTypeId();
  Returns the id of record.
* Boolean isAvailable();
  Returns true if the record type is available to the current user.
* Boolean isDefaultTypeRecordMapping();
  Returns true if this record type is the default record type for the user.

DescribeFieldResult :-

```
Schema.DescribefieldResult obj = Object.Fields.getDescribe();
```

Eg:-

DescribefieldResult stores the description about the field of an object. to get the description

```
Schema.DescribefieldResult obj = Account.Name.getDescribe();
                                  ↓       ↓
                              obj name  FieldName
```

DescribefieldResult methods :-

```
String getLabel();
Integer getLength();
String getLocalName();
String getName();
Integer precission();
Integer getScale();
```

* Schema.Displaytype getType();

This method returns the datatype of the field.

Boolean isAccessible();

Boolean isAutoNumber();

Boolean isCalculated(); This returns true if the field is userdefined custom formula field.

Boolean isCustom();

Boolean isDependentPicklist();

Boolean isIdLookup();

Boolean isRestrictedDelete();

Create a dropdown list in the visualforce page with the picklist values (options) of particular field of an object :-

Schema.picklistEntry :-

We have a class called schema.picklistEntry which can store one picklist field option.

Schema.DescribeFieldResult obj = Account.Industry.getDescribe();

This statement gets the description about 'industry field and stores to object obj.

NOTE :- As we have list of options & more than one option in industry we create

List <Schema.picklistEntry> P = obj.getpicklistValues();

Example program :-

```
Public class FieldDes {
    Public List<Selectoption> options;
    Public List<Selectoption> getOptions()
        {
        return options;
        }
```

```
Public String result {get; set;}
    public FieldDes()
          {
     options = new List<selectoption>();
 Schema.DescribeFieldResult obj = Account.Industry.getDescribe();
 List<Schema.picklistEntry> p = obj.getpicklistValues();
    for(Schema.pickListEntry x : p)
          {
    Selectoption op = new Selectoption(x.getlabel(), x.getlabel());
        options.add(OP);
             }
         }
      }
```

```
<apex:page controller = "FieldDes">
 <apex:form>
 <apex:selectlist size = "1">
 <apex:selectoptions value = "{!options}"/>
 </apex:SelectList>
 </apex:form>
</apex:page>
```

We can also get the description about the field

Schema.DescribeFieldResult obj = schema.sObjectType.Account.Fields.Industry

Write a program to display list of all the fields of an object :-

Map<String, schema.sObjectType> m = schema.sObjectType.Account.fields.getMap

It is going to get all the field names along with properties.

```
public class fieldDes {

  public List<selectoption> options;

  public List<selectoption> getOptions()

        {

    return options;

        }

  public fieldDes()

        {

  options = new List<selectoption>();

Map<String, schema.sobjectField> m = schema.sobjectType.Account.fields.
                                                    getMap();

  Set<String> keys = m.keySet();

    for(String s : keys)

        {

    Selectoption op = new selectoption(s,s);

        options.add(op);

        }

    }

}
```

```
<apex:page Controller="fieldDes">
 <apex:form>
 <apex:selectlist size="1">
 <apex:selectoption value="{!options}" />
 </apex:selectlist>
 </apex:form>
</apex:page>
```

Display the list of fields of an object based on the object selected.

public class FieldDes {

Write a program to display list of custom objects available based on the object selected by user from the displayed list display the fields of corresponding object.

```
public class FieldDes{
Map <String, Schema.SObjectType > my;
Public String oname {get; set;}
public List<selectoption> myoptions;
Public List<SelectOption> options;
Public FieldDes()
    {
Options = new List<selectoption>();
myoptions = new List<selectoption>();
my = schema.getGlobalDescribe();
Set<string> myobi = my.keySet();
for(String s : myobi)
    {
if(my.get(s).getDescribe().iscustom())
    {
Selectoption ab = new Selectoption(s,s);
myoptions.add(ab);
    }
  }
}
public pageReference show()
  {
Map<String, Schema.SObjectField> m = my.get(oname).getDescribe().
                                     Fields().getMap();
```

```
set(String s : keys)
        {
    selectoption op = new selectoption(s,s);
          options.add(op);
              }
    return null;
      }
public List<selectoption> getOptions()
          {
          return options;
              }
public List <selectoption> getMyoptions()
              {
          return myoptions;
                  }
          }
```

```
<apex:page controller="FieldDes">
 <apex:form>
  <apex:pageBlock>
   <apex:pageBlockSection>
    <apex:pageBlockSectionItem>
     <apex:outputLabel>object Name </apex:outputLabel>
     <apex:selectList value="{!oname}" size="1">
      <apex:selectoptions value="{!myoptions}"> </apex:selectoptions>
     </apex:selectList>
    </apex:pageBlockSectionItem>
   <apex:pageBlockSectionItem id="one">
   <apex:outputLabel >{!oname} - Fields</apex:outputLabel>
   <apex:selectList size="1" style="width:150px;">
    <apex:selectoptions value="{!options}"/>
```

```
</apex:selectList>
</apex:pageBlockSectionItem>
<apex:commandButton value="click" action="{!show}"/>
</apex:pageBlockSectionItem
</apex:pageBlockSection>
</apex:pageBlock>
</apex:form>
</apex:page>
```

Write a program on page Block Table with Dynamic columns in Salesforce :-

Visualforce page :-

```
<apex:page controller = "DynamicTableController">
<apex:pageBlock>
 <apex:form>
  <apex:actionFunction name = "ObjectFields" action = "{!ObjectFields}"/>
  <apex:commandButton value = "Show Table" action = "{!showTable}"/>
 <apex:pageBlockSection>
  <apex:pageBlockSectionItem>
   <apex:outputLabel value = "Select Object"/>
   <apex:selectList multiselect = "false" size = "1" value = "{!selectedObject}"
                    onchange = "ObjectFields();">
    <apex:selectOption itemLabel = "--None--" itemValue = "--None--"/>
    <apex:selectOptions value = "{!supptedObject}"/>
   </apex:selectList>
 </apex:pageBlockSectionItem>
 <apex:pageBlockSectionItem>
<apex:outputLabel value = "Select field"/>
<apex:selectList multiselect = "true" size = "5" value = "{!selectedFields}">
<apex:selectOption itemLabel = "--None--" itemValue = "--None--"/>
<apex:selectOptions value = "{!fieldTableApI}"/>
</apex:selectList>
</apex:pageBlockSectionItem>
```

```
<apex:pageBlockTable rendered="{!IF(ObjectList.size>0,true,false)}"
                     value="{!ObjectList}" var="rec">
<apex:column value="{!rec.Id}" rendered="{!IF(selectedFields.size==0,
                                              true,false)}"/>
<apex:repeat value="{!selectedFields}" var="fieldTable">
 <apex:column value="{!rec[fieldTable]}" rendered="{!IF(fieldTable
                                        !='--None--',true,false)}"/>
 </apex:repeat>
</apex:pageBlockTable>
 <apex:outputpanel rendered="{!IF(ObjectList.size<1,true,false)}">
<apex:pageMessage severity="Error" summary="No records to
                                            display"/>
</apex:outputpanel>
</apex:pageBlockSection>
</apex:form>
</apex:pageBlock>
</apex:page>
```

DynamicTableController.Apex (Apex class):-

```
public class DynamicTableController
{
  //List displayed on UI
public List<selectoption> SupptedObject{get; set;}
public String SelectedObject{get; set;}
Map<String, Schema.SobjectType> gd = Schema.getGlobalDescribe();
Set<String> objectkeys = gd.keySet();
```

```
public List <Selectoption> fieldLabelApi {get; set;}
public List<String> SelectedFields {get; set;}
public List<SObject> ObjectList {get; set;}

public DynamicTableController()
    {
 SupportedObject = new List<selectoption>();
  SelectedObject = '';
 fieldLabelApi =new List<selectoption>();
 SelectedFields =new list<String>();
 ObjectList = new list<sobject>();
for(Schema.SobjectType item: processInstance.targetObjectId.getDescribe().
                                        getReferenceTo())
      {
      //Excluding custom setting objects
  if(!item.getDescribe().customSetting)
        {
   SupportedObject.add(new Selectoption(item.getDescribe().getLocalName().
                toLowerCase(), item.getDescribe().getLabel()));
        }
      }
  public void ObjectFields()
      {
    if(SelectedObject != '--None--')
      {
 Schema.SobjectType SystemObjectType = gd.get(SelectedObject);
```

```
Schema.DescribeSobjectResult r = systemObjectType.getDescribe();
Map<String, Schema.SobjectResultField> M = r.fields.getMap();
for(Schema.SobjectFileld fieldAPI : M.values())
        {
    fieldLabelAPI.add(new selectoption(fieldAPI.getDescribe().getName(),
                                       fieldAPI.getDescribe().getLabel()));
                }
            }
        }
public void ShowTable()
    {
    String myQuery = 'Select Id';
  for(String field :SelectedFields)
        {
    if(field.toLowerCase() != 'id' && field.toLowerCase() != '--none--')
      myQuery += ',' +field+ ' ';
            }
  myQuery += 'from' + SelectedObject + 'LIMIT 100';
   ObjectList = Database.query(myQuery);
        }
    }
```

Write a program on dynamic multiselect picklist on visualforce page.

visualforce:page:-

```
<apex:page controller="multiselect1">
<apex:form>
<apex:pageBlock tabStyle = "Contact">
```

```
<apex:outputLabel >Account Name</apex:outputLabel><apex:inputText
                                                value="{!accName}"/>
 <apex:panelGrid  columns="10"  id="abcd">
 <apex:outputLabel >Type :</apex:outputLabel>
 <apex:selectList  id="sel1"  value="{!leftselected}" multiselect="true"
        style="width:150px"  size="5">
 <apex:selectOptions  value="{!unselectedvalues}"/>
 </apex:selectList>
   <apex:panelGroup>
   <br/>
   <apex:image  value="{!$Resource.multiselect}">
    <apex:actionSupport event="onclick" action="{!selectclick}" reRender
                                                = "abcd"/>
 </apex:image>
  <br/> <br/>
 <apex:image  value="{!$Resource.multiunselect}">
   <apex:actionSupport event= "onclick" action="{!unselectclick}"
                        reRender = "abcd"/>
   </apex:image>
   </apex:panelGroup>
<apex:selectlist  id = "sel2"  value="{!rightselected}" multiselect="true"
   style = "width:150px"  size ="5" >
     <apex:selectOptions  value = "{!SelectedValues}"/>
   </apex:selectlist>
  </apex:panelGrid>
    <apex:commandButton  value = "Save"  action = "{!Save}"/>
```

```
</apex:pageBlock>
</apex:form>
</apex:page>
```

multiselect1.apex (Apex class) :-

```
public class multiselect1
    {
  String strget='';
 public String Accid{get; set;}
public String accName{get; set;}
Set<String> originalvalues = new Set<String>();
 public List<String> leftselected{get; set;}
 public List<string> rightSelected{get; set;}
Set<String> leftValues=new Set<String>();
Set<String> rightValues = new Set<String>();
List<selectoption> options=new List<selectoption>();
String selectedvalue=''; List<String> lstfinal =new List<String>();
 public List<selectoption> getCountries()
    {
      List<selectoption> options = new List<selectoption>();
Schema.DescribefieldResult fieldResult = Schema.sObjectType.Account.
            fields.Type.getSobjectfield().getDescribe();
  List<Schema.picklistEntry> ple = fieldResult.getpicklistValues();
  for(Schema.picklistEntry f:ple)
      {
   options.add(new SelectOption(f.getLabel(), f.getValue()));
       originalvalues.add(f.getlabel());
    }
```

```
    return options;
    }
public multiselect1()
    {
    getCountries();
    GetEdit();
  leftSelected = new List<String>();
  rightSelected = new List<String>();
  leftValues.addAll(originalValues);
    }
Public pageReference selectClick()
    {
  rightSelected.clear();
  for(String s : leftSelected)
    {
    leftValues.remove(s);
    rightValues.add(s);
    }
  return null;
    }
Public pageReference unselectclick()
    {
  leftselected.clear();
  for(String s: rightselected)
    {
    rightValues.remove(s);
    leftValues.add(s);
    }
  return null;
    }
Public List<selectoption> getunselected values()
    {
```

```
}
List <Selectoption> options =new List<Selectoption >();
List < String> templist =new List<String>();
templist.addAll (leftvalues);
templist.sort();
for(String s:templist)
options.add (new Selectoption(s,s));
  return options;
      }
List <Selectoption> options1 =new List<Selectoption>();
public List <Selectoption> getSelected Values()
      {
  options1.clear();
List <String> templist = new List<String>();
templist.addAll (rightvalues); templist.sort();
for( String s:templist)
 Options1.add (new Selectoption(s,s));
 System.debug('***** Selected values'+options1);
for(integer i=0; i<options1.size(); i++)
      {
    if(i==0)
 Selectedvalue = options1[i].getValue();
   else
    Selectedvalue = ',' + options1[i].getValue();
          }
   return options1;
    }
Public void save()
      {
```

JSON (Javascript Object notation) :-

→ JSON is a shortform of javascript object notation.

→ It is easy way of storing information in a organized way.

→ Easy to access and human readable.

→ It is smaller than XML and language independent.

→ We represent javascript object notation (JSON) in name, value pairs. Where name is attribute name and value can be any one of the JSON supported values.

{ "Name" : "Rahul", "Branch" : "CSE" }
Field Name / value / Field Name / value

→ JSON values can be; Numbers (Integer, Decimal, Long, Double)

Eg:- { "Age" : 20 }
{ "Salary" : 10000.00 }

2) String

Eg:- { "Name" : "Sam" }
{ "Branch" : "CSE" }
String is represents in double quotes.

3) Boolean
It takes (true, false)

Eg:- { "Active" : true }
{ "status" : false }

JSON Object :-

Object can contain name, value pairs.

Eg:- If you have a class student and if you want to create an object in apex, we write in the form of

Eg:-
```
Class Student
  {
  public String name;
  public Integer Age;
  }
Student s = new Student();
  s.name = 'sam';
  s.age = 20;
  }
```

Now if the same object is represented in the JSON

{"name":"sam","Age":20}

Eg:-
```
class Student
  {
  public String branch;
  class Name
    {
    String firstName;
    String lastName;
    }
  Name c;
  Integer age;
  }
```

attributes

In JSON we represent like below

{"name":{"firstname":"myla","LastName","satish"},"branch":"CSE","Age":28}

NOTE:- The object representation is not only for apex classes, we can also represent a group of attributes as a object. That's why JSON starts with curly brasis.({).

Array:-

Arrays are represent in the square brackets. ([ ]).

Eg:- ["one", "two", "three"]

{"options": ["one", "two", "three"]}

{["one", "two", "three"]

{[{"Name": "Sam", "Age": 20}, {"name": "Ram", "Age": 27}]}

JSON Class:-

This class contains methods for Serializing Apex object into JSON format and deserializing method to convert back the JSON format into object format. (Apex object).

1. public Static String Serialize (Anytype of object)

This method takes apex object as an input and gives String of form which contains object in the JSON format.

```
Account a = new Account (Name = 'sam', Industry = 'Banking', phone = '123');
result = JSON.serialize(a);
```

{"attributes":{"type": "Account"}, "Name": "sam", "phone": "123",
"Industry": "Banking"}

2. public Static String serializepretty (object anyType)

This serializes the apex object into JSON content and generates indented content using the pretty print format.

3. public Static Object deserialize (String jsonString, System.type apexType)

Here System.Type means is apex object type that this method is going to create after deserializing the JSON object.

NOTE:-
If the JSON contains the attributes not present in the apex type specified in the argument such as missing field of object this method ignores this attributes & parses the rest of the JSON Content.

4. public static object deserializeStrict(String jsonString, System.Type apexType

NOTE:-
All the attributes in JSON string must be specified type. In JSON Content (JSON String) that we are passing Contains any attribut that are in the apex:type (apex:class & apex:object) specified in the argument in than this method throws run time exception.

Eg1:- public class car{
public String make;
public String year;
}
String s = '{"make":"tea","year":"2000","Age":20}';

In the above JSON String we have attribute age which is not in the class car.

Car c= (car)JSON.deserialize(s, car.class);

Eventhough age attribute is not in the car.class. It leaves that age field and Converts other fields in the form of car.class.

O/P:- car : [make=Tea , year=2000]

Eg2:-
String s= '{"make":"tea","year":"2000","Age":20}';
Car c = (car)JSON.deserializeStrict(s, car.class);

Age field is not there in the class. so when we use deserializeStrict(), if the JSON String contains any of the attribute which are not there in class we then it throws run time exception.

op: visualforce : runtime error

```
String s = {"name":"tea","year":"2000"};
Car c=(Car)JSON.deserializeStrict (s, Car.class);
```

op :- car:[make = Tea, year =2000]

5. public static object deserializeUntyped(String JsonString)

This method deserializes the JSON string into collection of premitive data types)

Deserialize the JSON String using JSON.deserializeUnTyped :-

JSONUntyped :-

```
public class JSONuntyped {
    public String result {get; set;}
    public JSONuntyped()
        {
    String jsoninput = '{\n'+
    '"description" : "Anappliance",\n'+
    '"accessories" : [powercord", '+
    '{"right" : "door handle1", '+
    '"left" : "door handle2"}],\n'+
    '"dimensions" : '+
    '{"height" : 5.5,'+
    '"width" : 8.0,'+
    '"depth" : 2.2},\n'+
    '"type" : null,\n'+
    '"Inventory" : 2000,\n'+
    '"price" : 1023.45,\n'+
    '"isShipped" : true,\n'+
    '"model Number" : "123"\n'+
    '}';
Map<String,Object> m = (<Map(String,Object>)JSON.deserializeUnTyped
                                                (JSONinput);
Set<String> S = m.keyset();
Myfields = new List<String>();
for(String a:s)
    {
```

```
        Myfields.add(a);
            }
    description = m.get('description');
     price =m.get('price');
Map<String,Object> my = (Map<String,Object>) m.get('dimension');
    for(String y : my.keyset())
            {
        result = result + '==' +y;
            }
    Accessories = (List<Object >m.get('accessories');
            }
List<Object> accessories =new List<Object >();
public String description {get; set;}
public Decimal price {get;set}
 public List<String> getMyFields()
            {
            return myfields;
            }
Public List<Object> getAccessories()
            {
            return accessories;
            }
```

JSON.createGenerator() :-

public static System.JSONGenerator createGenerator(Boolean pretty)

This method is going to return new JSON generator, the boolean value in the parameter list specifies whether JSON content in pretty print format or not.

JSON.createparser :-

public static System.JSONparser createparser(String jsonString)

System.JSONGenerator :-

This class contains the methods used to serialize the objects into JSON content using standard JSON encoding.

writeStartObject() :- This will write the starting mark JSON object.

writeEndObject() :-

```
JSONGenerator jn = JSON.createGenerator(true);
        jn.writeStartObject();
        jn.writeEndObject();
      String result = jn.getAsString()
```

o/p:- '{}'

JSON Generator methods :-

isclosed()
writeBlob(Blob)
writeBlobfield(String, Blob)
writeBoolean(Boolean)
writeBooleanfield(String, Boolean)
writeDate(Date)
writeDatefield(String, Date)

WriteDateTime (Datetime)

WriteDateTimefield (String, Datetime)

WriteEndArray ()

WriteEndObject ()

WriteFieldName (String)

Write

WriteNull ()

WriteNullField (String)

WriteNumber (Decimal)

WriteNumber (Double)

WriteNumber (Double)

WriteNumber (Integer)

WriteNumber (Long)

WriteNumberField (String, Decimal)

WriteNumber (String, Long)

WriteStartArray ()

WriteNumberField (String, Decimal)

WriteNumber (String, Long)

WriteStartArray ()

WriteStartArray ()

WriteObject Field (String, Anytype)

WriteObject (Anytype)

WriteString (String)

WriteTime (time)

Description of JSONGenerator methods :- Is-6

isClosed() :- Returns true if the JSON generator is closed ; otherwise returns false.

writeBlob(Blob) :- Writes the specified Blob value as a base64-encoded String.

writeBlobField(String, Blob) :- Writes a field name and value pair using the specified fieldname and BLOB value.

writeBoolean(Boolean) :- Writes the specified boolean value.

writeBooleanField(String, Boolean) :- Writes a field name and value pair using the specified field name and BLOB value.

writeDate(Date) :- Writes the specified date value in the ISO-8601 format.

writeDateField(String, Date) :- Writes a field name and value pair using the specified field name and boolean value.

writeDate(Date) :- Writes the specified date value in the ISO-8601 format.

writeDateTime(Datetime) :- Writes the specified date and time value in the ISO-8601 format.

writeDateTimeField(String, Datetime) :- Writes a field name and value pair using the specified field name and date and time value. the date and time value is written in the ISO-8601 format.

writeEndArray() :- Writes the ending marker of a JSON array (']')

writeEndObject() :- Writes the ending marker of a JSON object ('}')

writeFieldName(String) :- Writes a field name.

write(ID) :- Writes the specified ID value.

writeIDField(String, ID) :- Writes a field name and value pair using the specified field name and identifier value.

writeNull():- Writes the JSON null literal value.

writeNullField(String):- Writes a field name and value pair using the specified field name and the JSON null literal value.

writeNumber(Decimal):- Writes the specified decimal value.

writeNumber(Double):- Writes the specified integer value.

writeNumber(Long):- Writes the specified long value.

writeNumberField(String, Decimal):- writes a field name and value pair using the specified field name and decimal value.

writeNumberField(String, Double):- Writes a field name and value pair using the specified field name and double value.

writeNumberField(String, Integer):- Writes a field name and value pair using the specified field name and integer value.

writeNumberField(String, Long):- Writes a field name and value pair using the specified field name and integer value.

writeObject(Any type):- writes the specified Apex object in JSON format.

writeObjectField(String, Anytype):- writes a field name and value pair using the specified field name and apex object.

writeStartArray():- Writes the starting marker of a JSON array ('[')

writeStartObject():- Writes the starting marker of a JSON object ('{')

writeString(String):- Writes the specified String value.

writeStringField(String, String):- writes the specified field name and value pair using the specified field name and String value.

writeTime(Time):- Writes the specified time value in the ISO-8601 format.

writeTimeField(String, Time):- writes a field name and value pair using the specified field name and time value in the ISO-8601 format.

Generate the following JSON format using JSON Generator :-

Ex1: JSON format : {"one":["sam","ram"]}

```
jn.writeStartObject();
jn.writeFieldName('one');
jn.writeStartArray();
jn.writeString('sam');
jn.writeString('ram');
jn.writeEndObject();
jn.writeEndArray();
```

Ex2:

```
class Student
    {
public String name;
public Integer Age;
        }
```

{"Student":{"name":"ravi","age":20},"MARKS":[20,30]}

```
jn.writeStartObject();
jn.writeFieldName('Student');
jn.writeStartObject();
jn.writeStringField('name','ravi');
jn.writeNumberField('Age',20);
jn.writeEndObject();
jn.writeFieldName("marks");
jn.writeStartArray();
jn.writeNumber(20);
jn.writeNumber(30);
jn.writeEndArray();
jn.writeEndObject();
```

JSONGeneratorExam :- (Apex class) :-

```
public class JSONGeneratorExam {
    public String result {get; set;}
    Public JSONGeneratorExam() {
        JSONGenerator jn = JSON.createGenerator(true);
            jn.writeStartObject();
            jn.writeStringField('Name', 'satish');
            jn.writeNumberField('Age', 28);
            jn.writeFieldName('Account');
        List<Account> acc = [select name, industry from Account limit 2];
            jn.writeObject(acc);
            jn.writeFieldName('myarray');
            jn.writeStartArray();
            jn.writeNumber(10);
            jn.writeNumber(20);
            jn.writeEndArray();
            jn.writeEndObject();
            result = jn.getAsString();
    }
}
```

JSONGeneratorExample :- (Visualforce page) :-

```
<apex:page controller = "JSONGeneratorExam">
<apex:outputLabel> {!result} </apex:outputLabel>
</apex:page>
```

```
public class JSONDemo
    {
  public String show()
        {
    JSONGenerator jn = JSON.createGenerator(true);
  List<Account> acc = [select name,Industry from Account limit 2];
      jn.writeObject(acc);
    String result = jn.getAsString();
     return result;
        }
    }
```

JSONGeneratorExam (Apex class):-

```
public class JSONGeneratorExam{
  public String result {get; set;}
  public JSONGeneratorExam()
        {
   JSONDemo d = new JSONDemo();
   String sto = d.show();
 List<Account> acc = (List<Account>) JSON.deserialize(sto, List<Account>.
                                                          class);
   for(Account x:acc)
        {
  result = result + '==' + x.name;
        }
  }
```

JSON Generator Example (visualforce page):-

```
<apex:page controller="JSONGeneratorExam">
 <apex:outputLabel>{!&result&}</apex:outputLabel>
</apex:page>
```

NOTE:-

We can deserialize the JSON String generated by JSON generator only when entire string is in the form of apex type what we have given in the deserialize method.

NOTE:-

We can not deserialize apex generated JSON generator created String if it is not matching with the apex type what we have given.

Eg:-

```
List<Account> acc =[select name, Industry from Account limit 2];
    jn.WriteStartObject();
    jn.WriteFieldName('one');
    jn.WriteObject(acc);
    jn.WriteEndObject();
String result = jn.getAsString();
    return result;
}
List<Account> acc = (List<Account>) JSON.deserialize(result, List<Account>.
                                                      class);
}
```

```
List<Account> acc = (List<Account>) JSON.deserialize(result, List<Account>.
                                                      class);
```

The above function can be deserialize by using JSON.deserializeUntyped.

```
Map<String, Object> acc = (Map<String, Object>) JSON.deserializeUntyped(str);
```

System.JSONToken :-

This class contains all the token values parsing JSON content.

END_ARRAY :- The ending of an array value. This token is returned when ']' is encountered.

END_OBJECT :-

FIELD_NAME :-

NOT_AVAILABLE :-

START_ARRAY :-

START_OBJECT :-

VALUE_EMBEDDED_OBJECT :-

VALUE_FALSE :-

VALUE_NULL :-

VALUE_NUMBER_FLOAT :-

VALUE_NUMBER_INT :-

VALUE_STRING :-

VALUE_TRUE :-

JSONParser class :-

this will parse the JSON encoded content

nextToken() :-

public system.JSONToken nextToken()

It's going return the next token in the form of system.JSONToken. If there is no next token it returns null.

getCurrentName() :-

It's going to return the name of the current token. In case if the current token is a value it returns corresponding field name.

Eg :-
```
String str = '{"Name":"Sam","Age":20}';
JSONParser p = JSON.createParser(str);
p.nextToken();
p.nextToken();
result = p.getCurrentName(); // result = Name
p.nextToken();
result = p.getCurrentName(); // current token in "Sam" which is a value
                             so it returns the name of this value field
                             : result : name
```

getCurrentToken() :-

public system.JSONToken getCurrentToken();

It returns the current token on which the cursor is.

Eg :-
```
String str = '{"Name":"Sam","Age":20}';
JSONParser p = JSON.createParser(str);
p.nextToken();
result = '' + p.getCurrentToken(); // TOKEN.START_OBJECT
p.nextToken();
result = '' + p.getCurrentToken();
p.nextToken();
result = '' + p.getCurrentToken();
```

nextValue():

nextValue():-

It returns the next token that is value type & null. If the parser has reached the end of the input stream.

Eg:- String str = '{"name":"sam","Age":20}';

```
JSONparser p = JSON.createparser(str);
p.nextToken();
P.nextValue();
result = "" + p.getText(); //result = sam;
P.nextValue();
result = " " + p.getText(); // result = 20
```

getText():-

It returns the textual representation of the current token & null if there is no current token.

Eg:- String str = '{"name":"sam","Age":20}';

```
JSONparser p = JSON.createparser(str);
p.nextToken();
result = " " + p.getText(); //result = null
p.nextToken();
result = " " + p.getText(); //result = name
p.nextToken();
result = " " + p.getText(); // result = sam
```

→ Write a logic to read elements of an Array.

```
String str = {"name":"Raj", "subject":["cloud", "Java", "oracle"]};
    List<String> sub = new List<String>();
    JSONparser p = JSON.createparser(str);
    while(p.nextToken()!=null)
        {
        if(p.getText() == 'subject')
            {
            if(p.nextToken()== JSONToken.START_ARRAY)
                {
                while(p.nextToken()!=JSONToken.END_ARRAY')
                    {
                    sub.add(p.getText());
                    }
                }
            }
```

getIntegerValue():-

Syn:-

public Integer getIntegerValue()

If the current token is of type JSONToken.VALUE_NUMBER_INT then it returns the integer format of value.

Eg:- String JSONContent = '{"recordCount":10}';

```
    JSONparser parser = JSON.createparser(JSONContent);
    parser.nextToken();
    parser.nextValue();
    Integer count = parser.getIntegerValue();
```

getDecimalValue():-

Syn:-

public Decimal getDecimalValue()

This method returns the current token as decimal value provided if the current token is JS-11

JSONToken.VALUE_NUMBER_FLOAT & JSONToken.VALUE_NUMBER_INT.

Eg:- String JSONContent =
'{"GPA":3.8}';
JSONparser parser = JSON.createparser(JSONContent);
parser.nextToken();
parser.nextValue();
Decimal gpa = parser.getDecimalValue();

getDoubleValue():-

syn:-
Public Double getDoubleValue()

Eg:- String JSONcontent =
'{"GPA":3.8}';
JSONParser parser = JSON.createparser(JSONContent);
parser.nextToken();
parser.nextValue();
Double gpa = parser.getDoubleValue();

getLongValue():-

syn:-
Public Long getLongValue();

Return type:- Long.

getBlobvalue():-

syn:-
Public Blob getBlobValue();

getBooleanValue():-

Syn:-
public Boolean getBooleanValue()

Must be in the form of JSONToken.VALUE_TRUE (or) JSONToken.VALUE_FALS

getDatetimeValue():-

Syn:-
public Datetime getDatetimeValue()

Must be type of JSONToken.VALUE_STRING

Eg:- String JSONContent =
'{"transactionDate":"2011-03-22T13:01:28"}';

JSONParser parser = JSON.createParser(JSONContent);
parser.nextToken();
parser.nextValue();
Datetime transactionDate = parser.getDatetimeValue();

getDateValue():-

Syn:-
public Date getDateValue()
Must be type of JSONToken.VALUE_STRING

Eg:- String JSONContent =
'{"dateOfBirth":"2011-03-22"}';

JSONParser parser = JSON.createParser(JSONContent);
parser.nextToken();
parser.nextValue();
Date transactionDate = parser.getDateValue();

getTimeValue():-

Syn:- public Time getTimeValue()

Token must be type of JSONToken.VALUE_STRING.

Eg:- String JSON

String JSONContent = '{"arrivalTime": "18:05"}';

Time arrivalTime = parser.getTimeValue();

readValueAs():-

syn:-

Public object readValueAs(System.Type apexType)

This method deserializes the JSON Content into an object of the specified apex type and returns the deserialized object.

Eg:-

```
Str = {"name":"sam", "one":{"Age":20, "branch":"CSE"}}'

JSONparser p = JSON.createparser(str);
    p.nextToken();
    p.nextToken();
    p.nextToken();
    p.nextToken();
    p.nextToken();
    p.nextToken();
Bened b = (Bened)p.readValueAs(Bened.class)
    System.assertEquals(b.Age, 20);

class Bened
{
Integer age;
String branch;
}
```

readValueAsStrict():-

syn:-

public object readValueAsStrict(System.type apexType)

NOTE:-

All the attributes in the JSON Content must be present in specified apex type. If not it gives run time error.

skipChildren():-

syn:- Public void skipChildren()

This method skips all the child tokens of JSONToken.START_ARRAY, JSONToken.START_OBJECT that the cursor currently points to.

clearCurrentToken() :-

Syn:-

public void clearCurrentToken()

This will remove the current token.

Example program:-

parseExample :- (Apex class):-

```
public class parseExample
    {
public String nextToken{get; set;}
public String name{get; set;}
public Integer age{get; set;}
public Decimal salary{get; set;}
public List<Bened> result;
public List<Bened> getResult()
        {
        return result;
        }
public parseExample()
    {
result = new List<Bened>();
String str = {"bened":[{"Name": "Ram", "branch": "CSE"},{"Name":
        "kiran", "branch":"ECE"}], "Name": "Sam", "Age":20, "salary":20.
            "test":["one", "two"]}';
JSONParser p = JSON.createParser(str);
p.nextToken();
p.nextToken();
p.nextToken();
result = (List<Bened>)p.readValueAs(List<bened>.class);
p.nextToken();
p.nextValue();
```

```
name = p.getText();
p.nextValue();
age = p.getIntegerValue();
p.nextValue();
Salary = p.getIntegerValue();
p.nextValue();
Salary = p.getDecimalValue();
p.nextToken();
p.nextToken();
p.skipChildren();
nextToken = p.getText();
    }
}
```

Bened.Apex :-

```
public class Bened {
    String name;
    String branch;
        }
```

passesExample (visualforce page) :-

```
<apex:page controller="passesExample">
<apex:pageBlock>
    <apex:pageBlockTable value="{!result}" var="a">
        <apex:column value="{!a}"/>
    </apex:pageBlockTable>
<apex:outputLabel>My name == {!name}</apex:outputLabel><br/>
<apex:outputLabel>Age == {!age} </apex:outputLabel><br/>
<apex:outputLabel>Salary == {!Salary} </apex:outputLabel><br/>
<apex:outputLabel>nextToken == {!nextToken} </apex:outputLabel><br/>
```

Invoking Http callouts :-

Apex provides several built-in classes to work with HTTP services and create HTTP requests like GET, POST, PUT and DELETE.

→ You can use these HTTP classes to integrate to REST based services. They also allow you to integrate to SOAP based webservices as an alternate option to generating Apex code from a WSDL.

→ By using the HTTP classes, instead of starting with a WSDL, you take on more responsibility for handling the construction of the SOAP message for the request and response.

HTTP classes :-

These classes expose the general HTTP request/response functionality.

Http class :- use this class to initiate an HTTP request and response.

HttpRequest class :- use this class to programmatically create HTTP requests like GET, POST, PUT and DELETE.

HttpResponse class :- use this class to handle the HTTP response returned by HTTP.

The HttpRequest and HttpResponse classes support the following elements.

HttpRequest :-

→ HTTP request types such as GET, POST, PUT, DELETE, TRACE, CONNECT, HEAD and OPTIONS.

→ HTTP Request headers if needed.

→ Read and Connection timeouts.

→ Redirects if needed.

→ Content of the message body.

HttpResponse :-

→ The HTTP status code.

→ Response headers if needed.

→ Content of the response body.

The following example shows an HTTP GET request made to the external services specified by the value of url that gets passed into the getContent method. This

Eg:-

```
public class HttpcalloutSample
    {
//pass in the end point to be used using the String url.
  public String getContent(String url)
      {
  //Instantiate a new http object
    Http h = new Http();
//Instantiate a new HTTP request, specify the method (GET)
as well as the end point
    HttpRequest req = new HttpRequest();
    req.setEndpoint(url);
    req.setMethod('GET');
//Send the request, and return a response
    HTTPResponse res = h.send(req);
      return res.getBody();
        }
    }
```

usage of jquery:-

Jquery is fast and concise javascript framework created by Jhon Resig in 2006.

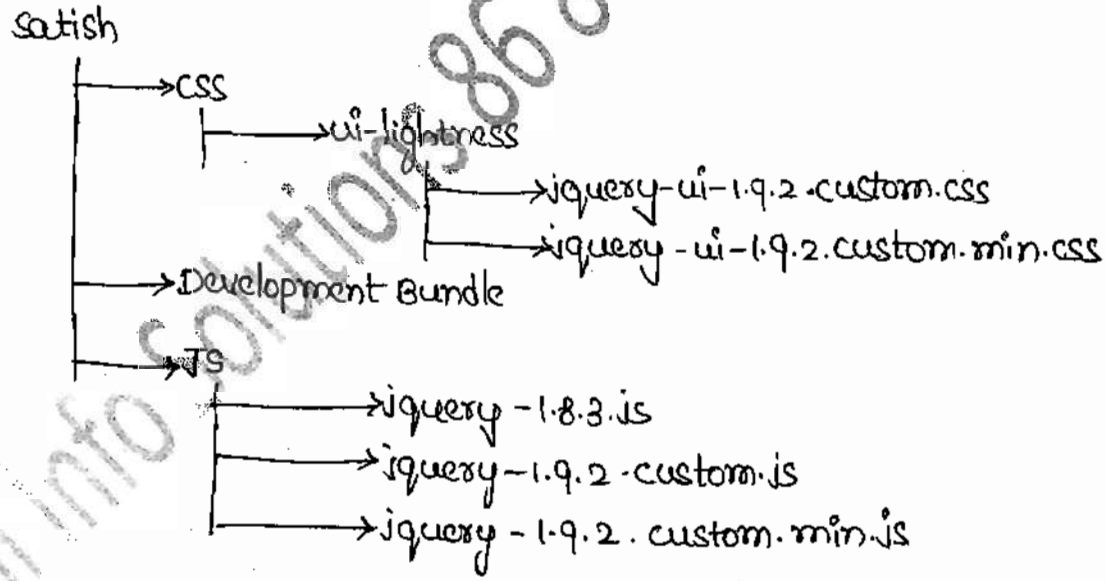
Jquery simplifies HTML document traversing, event handling, animating and AJAX interactions for rapid web development.

When you want to use the jquery in salesforce

→ Download jquery-ui-1.9.2 custom.zip file.

→ Rename this file after extracting the zip as satish.

→ verify the structure in satish folder. This should be like below



→ zip the folder as satish.zip.

→ Create a new static resource in the salesforce with the file satish.zip.

Steps to create static resource in the salesforce:

Setup → Build → Develop → Static Resources → New

Resource Name: satish

Description: jquery Bundle

File : choose satish.zip file

cache control : public → Save.

Include jquery resource files into the visualforce page :-

→ Include the JS, CSS files in to visualforce page by using <apex:includeScript> and <apex:stylesheet>

Example program :-

```
<apex:page>
 <apex:includeScript value="{!URLFOR($Resource.satish, '/satish/js
                                /jquery-1.8.3')}"/>
 <apex:includescript value="{!URLFOR($Resource.satish, '/satish/js
                                /jquery-ui-1.9.2.custom.js')}"/>
 <apex:includeScript value="{!URLFOR($Resource.satish, '/satish/js
                                /jquery-ui-1.9.2.custom.min.js')}"/>
 <apex:stylesheet value="{!URLFOR($Resource.satish, '/satish/css
                                /ui-lightness/jquery-ui-1.9.2.custom.css')}"/>
 <apex:stylesheet value="{!URLFOR($Resource.satish, '/satish/css
                                /ui-lightness/jquery-ui-1.9.2.custom.min.css")}"/>
   <script type="text/javascript">
    j$ = jQuery.noConflict();
     if(j$)
       {
      alert('success');
          }
     else
       {
      alert('not loaded');
          }
</apex:page>
```

→ Jquery should be written within the <Script> tag.

```
<script type="text/javascript">
```

→ Jquery statement starts with '$'. But in the Salesforce we use '$' to indicate global data. so it leads conflict between '$' of jquery & '$' of salesforce. so to remove this conflict we write

```
j$ = jQuery.noConflict();
```

which specifies wherever we use j$ in the visualforce page it is replaced with '$' of jquery.

Verify whether jquery was properly loaded into the visualforce page or not:

```
if(j$){
 alert('success');
    }
else{
 alert('not loaded');
    }
```

→ If 'j$' in the visualforce is replaced by '$' symbol in the jquery it returns true and gives the alert of success else not loaded.

The jquery statement start execution with the functionality

```
j$(document).ready(function()
                    {
                    write the logic;
                    }
                  );
```

→ What ever the logic we want to write we should write with in the function(){ }

Refering to a tag/components in visualforce :

1. Refering the html tag with id.

```
<div id="one">
</div>
j$('#one').click()
```

2. Refering the html tag with class.

```
<div class="one">
</div>
j$('.one').click()
```

```
j$(document).ready(
        function()
          {
            j$("#one").click
                  (function()
                    {j$("#two").hide();
                    }
                  );
          }
      );
```

→ In the above code

```
j$(document).ready(
                function()
                  {
                  }
              );
```

This indicates when the document is loaded successfully perform the function defined.

→ When you click on the component whose id is 'one' then perform the function defined within the click.

j("#two").hide(); It hides the component whose id is 'two'.

Sample program :-

```
<apex:page>
<apex:includeScript value="{!URLFOR($Resource.satish,'/satish/js/
                                    jquery.1.8.3.js')}"/>
    <script type="text/javascript">
        j$ = jquery.noConflict();
        if(j$) {
          alert('hello');
              }
        j$(document).ready(function()
                          {
                    j$('#one').click(
                                function()
                                  {
                              j$('#two').html('<h1>Sample<h1>')
                                    }
                                );
                          }
                        );
    </script>
    <div id="one">Hello</div>
    <div id="two">thank you</div>
</apex:page>
```

toggle() :-

toggles each of the set of matched elements. If they are shown, toggle makes them hidden. If they are hidden, toggle makes them shown.

Example :-

```
$(document).ready(function()
                  {
                    $("#one").click(function()
                                    {
                                      $("#two").toggle();
                                    });
                  });
```

slideDown(speed, callback) :-

only the height is adjusted for this animation, causing all matched elements to be revealed in a "sliding" manner.

Example :-

```
$(document).ready(function()
                  {
                    $("#one").click(function()
                                    {
                                      $("#two").slideDown("slow");
                                    });
                  });
```

slideUp(speed, callback) :-

Example :-

```
$(document).ready(function()
                  {
                    $("#one").click(function()
                                    {
                                      $("#two").slideUp("slow");
                                    });
                  });
```

```
    });
});
```

fadeOut(speed, callback) :-

only the opacity is adjusted for this animation, meaning that all of the matched elements should already have some form of height and width associated with them.

Example :-

```
$(document).ready(function()
{
    $("#one").click(function()
    {
        $("#two").fadeOut("slow");
    });
});
```

fadeTo(speed, opacity, callback) :-

Example :-

```
$(document).ready(function()
{
    $("#one").click(function()
    {
        $("#two").fadeOut fadeTo("slow", 0.5);
    });
});
```

mousemove(fn) :-

Bind a function to the mousemove event of each matched element.

Parameters:-

fn:(Function): A function to bind to the mousemove event on each of the matched elements.

Example:-

```
$(document).ready(function()
                  {
                    $("#butn").click(function()
                                     {
                                       $("p").mousemove(function(){alert("Hello");});
                                     }
                                    );
                  }
                 );
```

mouseover(fn):-

Example:-

```
$( $(document).ready(function()
                     {
                       $("#butn").click(function()
                                        {
                                          $("p").mouseover(function(){alert("Hello");});
                                        }
                                       );
                     }
                    );
```

click(fn):-

Bind a function to the click of event of each matched element.

Parameters:-

fn: (Function): A function to bind to the click event on each of the matched

elements.

Example:-

```
j$(document).ready(function()
                    {
                    j$("#butn").click(function(){alert("Hello");});
                    }
                    );
```

remove(expr):-

Removes all matched elements from the DOM. this does not remove them from the jQuery object, allowing you to use the matched elements further. can be filtered with an optional expressions.

Example:-

```
j$(document).ready(function()
                    {
                    j$("#butn").click(function()
                                      {
                                      j$("p").remove();
                                      });
                    });
```

appendTo(expr):-

Append all of the matched elements to another, specified, set of elements. this operation is, essentially, the reverse of doing a regular j$(A).append(B), in that instead of appending B to A, you're appending A to B.

Example:-

```
j$(document).ready(function()
                    {
                    j$("#butn").click(function()
                                      {
                                      j$("p").appendTo("#two");
                                      });
                    });
```

Flows Introduction:-

The Flow Designer, the tool for creating flows, lets you configure screens and define branching logic for your flows without writing any code.

Elements are the building blocks of flows. Each element represents an action, such as representing information to, or collecting information from, flow users, or even querying, creating, updating and deleting information in salesforce. By connecting elements together in the flow designer, you can create a flow, which is a series of screens, inputs and outputs through which users navigate.

The following elements are available in the cloud Flow Designer:

- Step
- screen
- Desig
- Decision
- Assignment
- Record create
- Record update
- Record lookup
- Record delete
- subflows
- Apex plug-In
- Connector

NOTE:- Every time you add an element or resource to a flow, it's also added to the Explorer tab.

Q: How to on the development mode on?
A: Name-->MySettings-->Personal-->Advanced user Details-->Edit-->
Enable Development mode checkbox

Q::HOw many ways we can create the visualforce pages?
A:: There are two ways in which we can create a visualforce page.

1. Setup-->Build-->Develop-->Pages-->New Page-->Create the code

Note: if you want to run the code call page in the URL https://ap1.salesforce.com/apex/pagename
If any page exists with above pagenamge it will open otherwise it show a link to create a page on this name

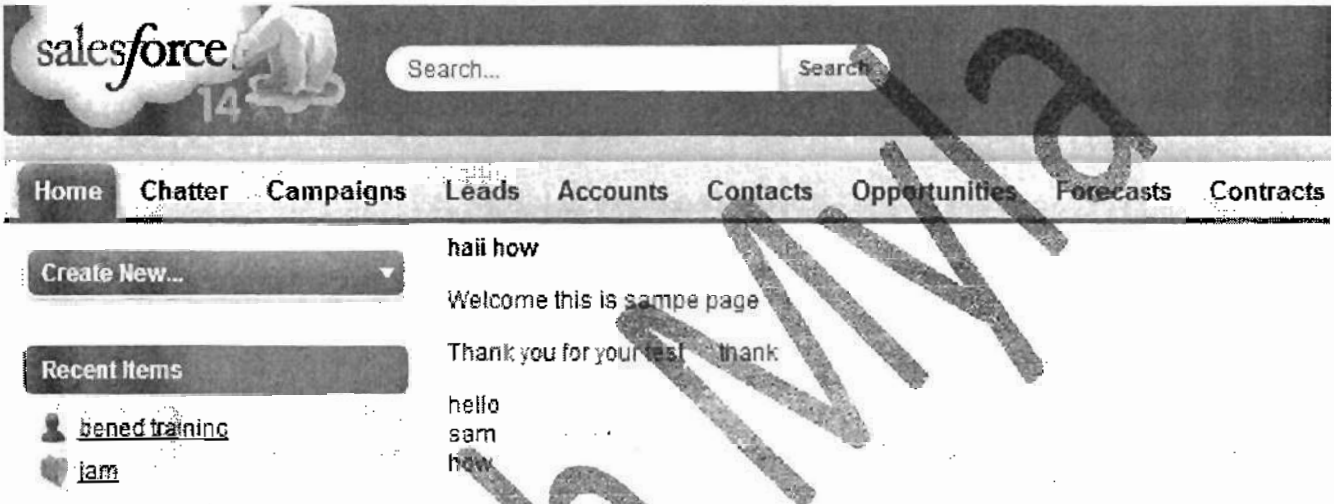
1. <apex:page> : Every visualforce page will start with this component.

Attributes:
Id : This is used to recognize the component uniquly
apiVersion : This will indicate on which version of api we are using by default we will be using latest Version of api.i.e29.0
show header : This is a boolean value , if we give true it show it will show salesforce provided header on your page false it will hide.
sidebar : It is a boolan value ,if we give true it shows sidebar components on your page.
setup : It is boolean value if we give true it will show setu menu components in sidebar
rendered : It is a boolean value if we give true it will display the component. If it is false it will hide the Component
renderAs : It will specify how the page has to be displayed like pdf xls/zip
showChat : It is a boolean value when we give true it will show chatter on the page.

Example:

```
<apex:page>
  <h1> hai</h1>
  <h1> how</h1>
  <p> Welcome this is sampe page</p>
  <p> Thank you for your test      thank</p>
  hello<br/> sam<br/> how<br/>
</apex:page>
```

Output Screen:



➔<apex : sectionHeader> : This is used to create header for the page .

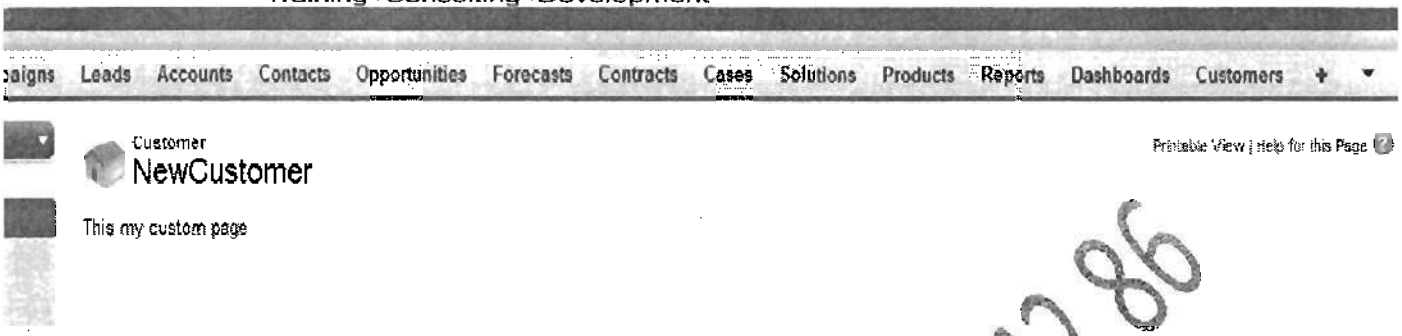
Attributes :

title :
subtitle:
help :
description :
printUrl :

Example :

```
<apex:page
    <apex:sectionHeader title="Customer" subtitle="NewCustomer"
                help="http://www.google.com"      description="This my custom page"
                printUrl="/apex/firspage"/>
</apex:page>
```

Output screen :

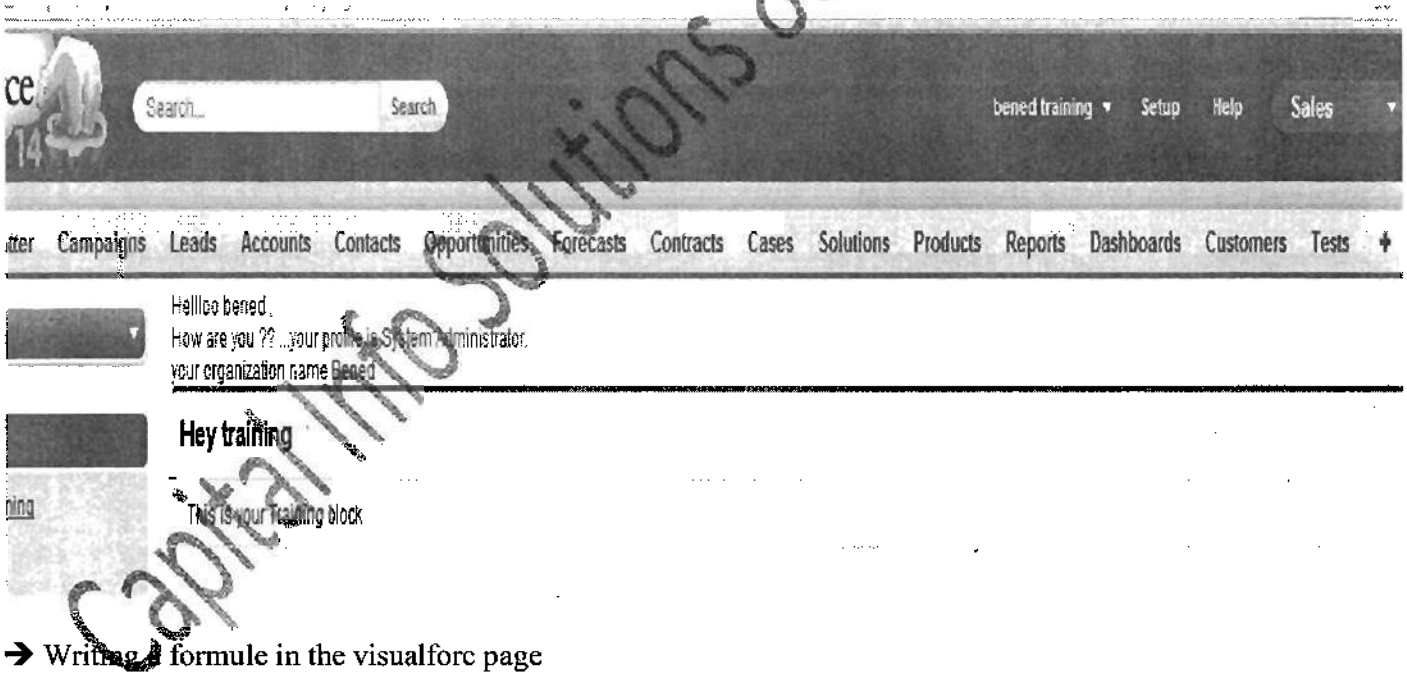


➔ Referering To global data in the visualforce page:

When we want to refer the global data we have call the data in expression format . i.e {!$ObjectName.FieldName}//Note every global object is prefixed by '$'.

Example :{!$User.FirstName}

{!$Profile.Name}



➔ Writing a formule in the visualforc page

when we want to write a formule we have to declare it in expression {!}

Example

{! 10}

{!10+20}

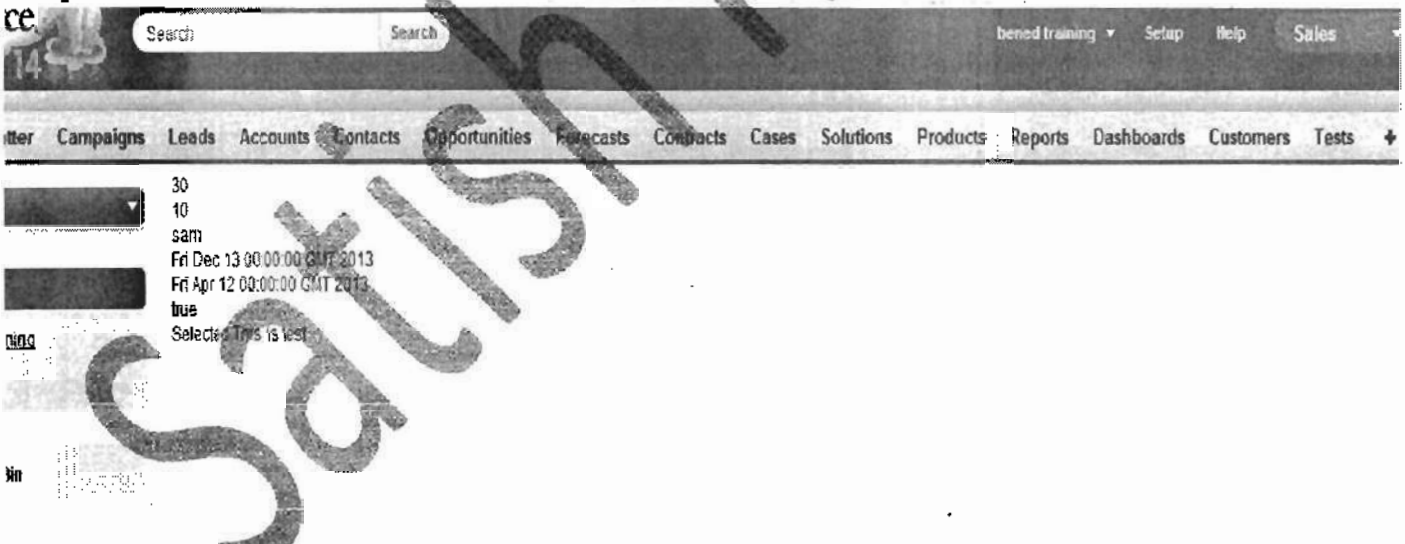
{!'sam'}

{! TODAY()}
{!NOW()}
{!ISBLANK('')}
{!ISBLANK(Accoount.Name)}

Example :

```
<apex:page >
    {!10+20}
    <br/> {!10}<br/> {!'sam'}<br/>{!TODAY()}<br/>
      {!DATE(2013,4,12)}
    <br/>{!ISBLANK('')}<br/>
     {!IF(ISBLANK(''),'Selected','rejected')}
    <apex:outputLabel rendered="{!ISBLANK('')}">This is test</apex:outputLabel>
  </apex:page>
```

OutputScreen :



➔ <apex:pageMessage>

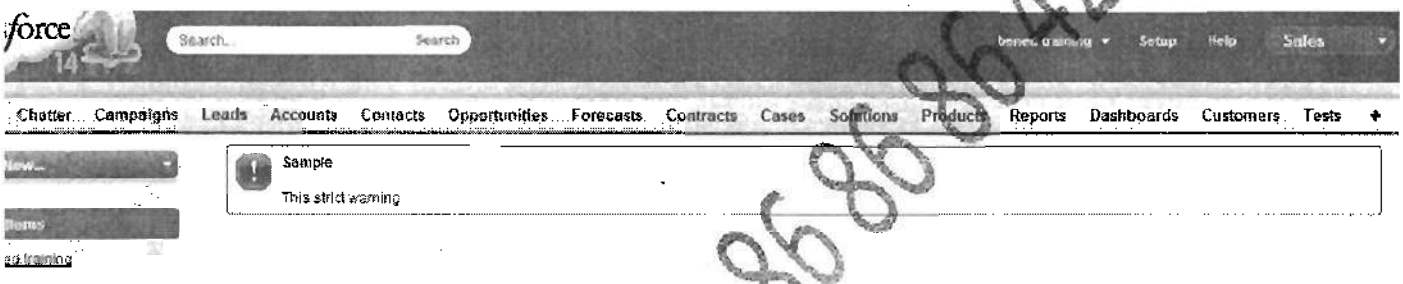
This is used for generating the error message .

1.Title : This is the title name for the error
2.Summary :This prints the summary of the title error message.
3.detail :This will print the detail description of the error.
4.Severity : This is logo type of the error
1.error 2.Warning 3.info 4.confirm
5.Strength : This size of the severity message logo ranges from 1-3
6.Rendered : Wheater the component should be displayed or not.

Example :

```
<apex:page >
 <apex:pageMessage title="Sample" severity="error" strength="3" detail="strict warning" >
  </apex:pageMessage>
</apex:page>
```

Output Screen :



<apex:pageBlock> :This component will create a block in the page .We can create no of blocks in the page with title ,body .

Attributes :

Title: This will create title for the pageBlock

helpTitle: This will display the help link with the given name in the pageBlock .We can give Context related help

helpURL: If we click on the helpTitle the URL what we have given here will be opened.

dir :This will specify the direction in which content of the pageBlock Should b displayed

rendered : This is a Boolean value which specifies whether the pageBlock should be displayed on the page or not

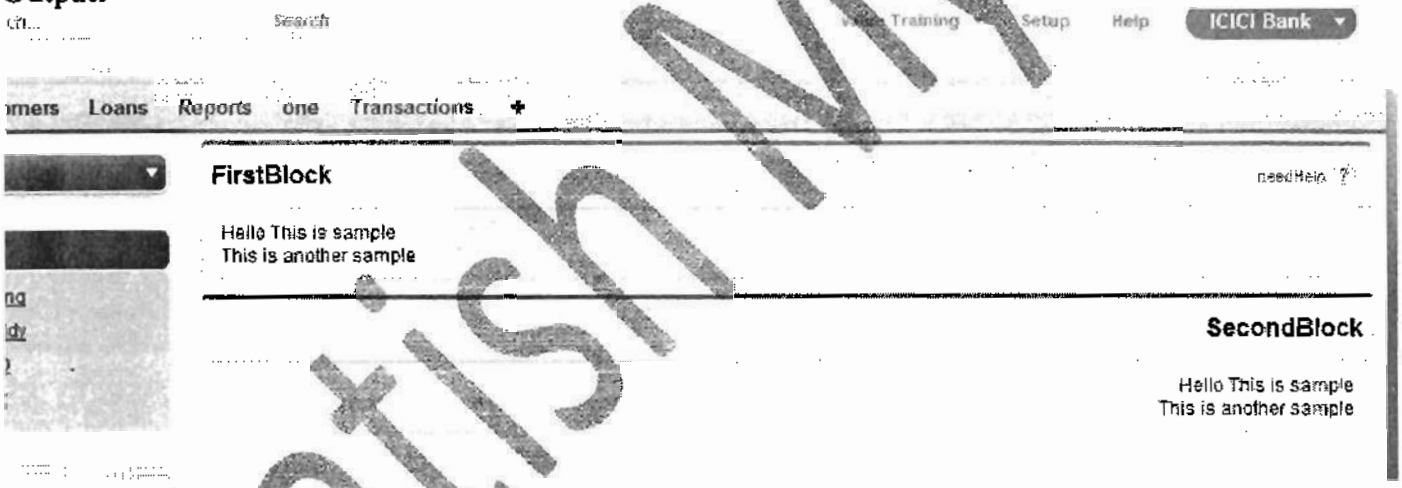
id : This will specify id of the component to recognize the component in the page.

Example:

```
<apex:page >
 <apex:pageBlock title="FirstBlock" helpTitle="needHelp" helpUrl="http://www.google.com">
   Hello This is sample<br/>
   This is another sample
```

```
</apex:pageBlock>
<apex:pageBlock title="SecondBlock" dir="RTL" tabStyle="Loan__c">
  Hello This is sample<br/>
   This is another sample
</apex:pageBlock>
<apex:pageBlock title="ThirdBlock" rendered="{!NOT(true)}">
</apex:pageBlock>
</apex:page>
```

Output:



<apex:commandButton> : This is used to create a button on the page. Button should be created in the <apex:form> component.

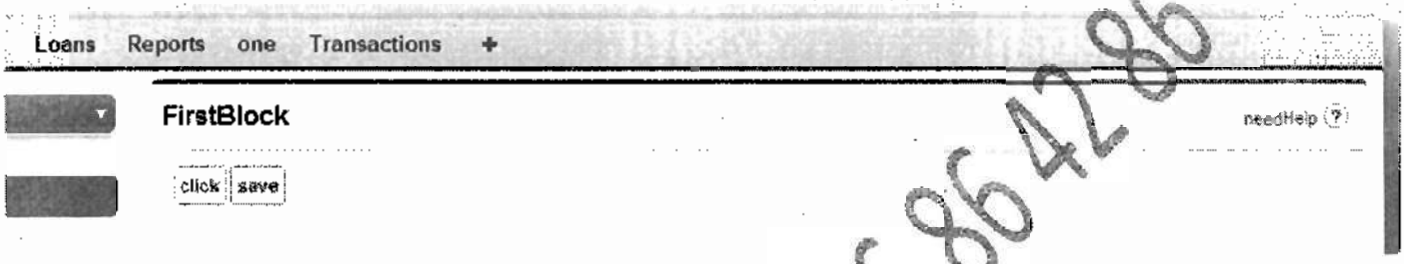
Attributes:

1. Value : The Name that should e be displayed on the button should be given as value.
2. Action : When we click on the button what action should be performed is given here.
3. Disable : It is a Boolean values If we want to disable the button we can give true.
4. Dir : Direction of the text that should be displayed on the button.

Example:

```
<apex:page >
<apex:form >
```

```
<apex:pageBlock title="FirstBlock" helpTitle="needHelp" helpUrl="http://www.google.com">
<apex:commandButton value="click"  action="http://www.google.com"/>
<apex:commandButton value="save" action="{! save}" disabled="false"/>
 </apex:pageBlock>
</apex:form>
</apex:page>
```



<apex:pageBlockButton> :This will create buttons on the pageBlock.ie with respect to particular pageBlock .This should be child component of the <apex:pageBlock>

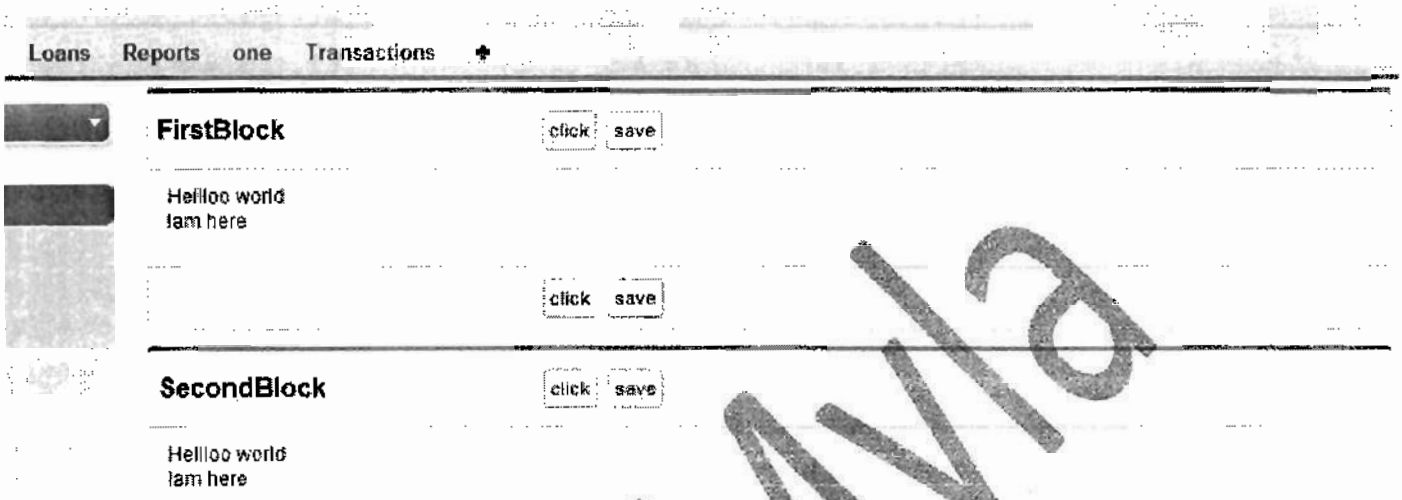
Attributes:

Location : This specify whether the buttons should be displayed on the top| bottom | on both .

Example:

```
<apex:page >
<apex:form >
 <apex:pageBlock title="FirstBlock" >
     <apex:pageBlockButtons >
        <apex:commandButton value="click"  action="http://www.google.com"/>
        <apex:commandButton value="save" action="{! save}" disabled="false"/>
    </apex:pageBlockButtons>
     Hellloo world <br/> Iam here
 </apex:pageBlock>
  <apex:pageBlock title="SecondBlock" >
     <apex:pageBlockButtons location="top" >
        <apex:commandButton value="click"  action="http://www.google.com"/>
        <apex:commandButton value="save" action="{! save}" disabled="false"/>
    </apex:pageBlockButtons>
    Hellloo world <br/> Iam here
 </apex:pageBlock>
</apex:form>
</apex:page>
```

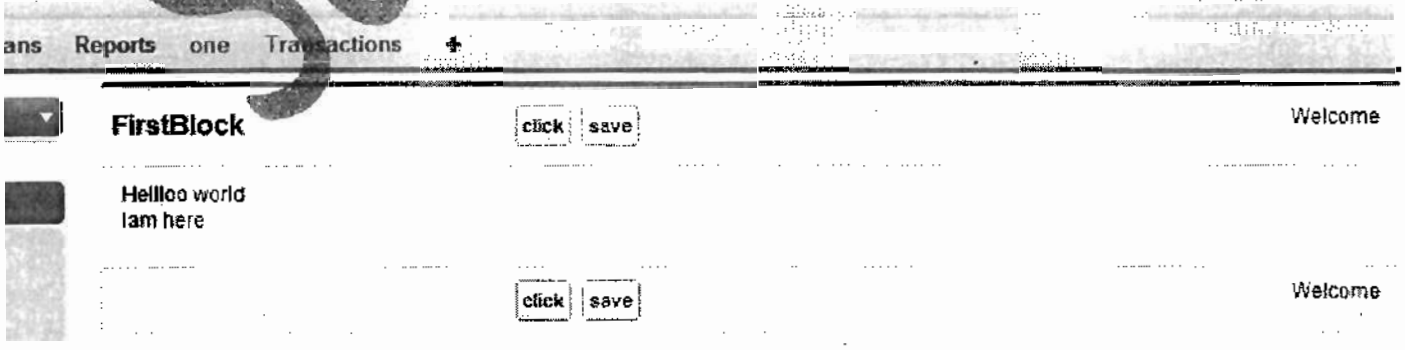
OutputScreen:



Example :

```
<apex:page >
<apex:form >
 <apex:pageBlock title="FirstBlock">
  <apex:pageBlockButtons dir="LTR">
    <apex:commandButton value="click" action="http://www.google.com"/>
     <apex:commandButton value="save" action="{! save}" disabled="false"/>
     <span style="float:right;">Welcome</span>
  </apex:pageBlockButtons>
  Hellloo world <br/> Iam here
 </apex:pageBlock>
</apex:form>
</apex:page>
```

Output Screen :



<apex:pageBlockSection> : This is a child component of a <apex:pageBlock> .We can define the no of columns for a row .

->In every column we can print one apex component .

Attributes:

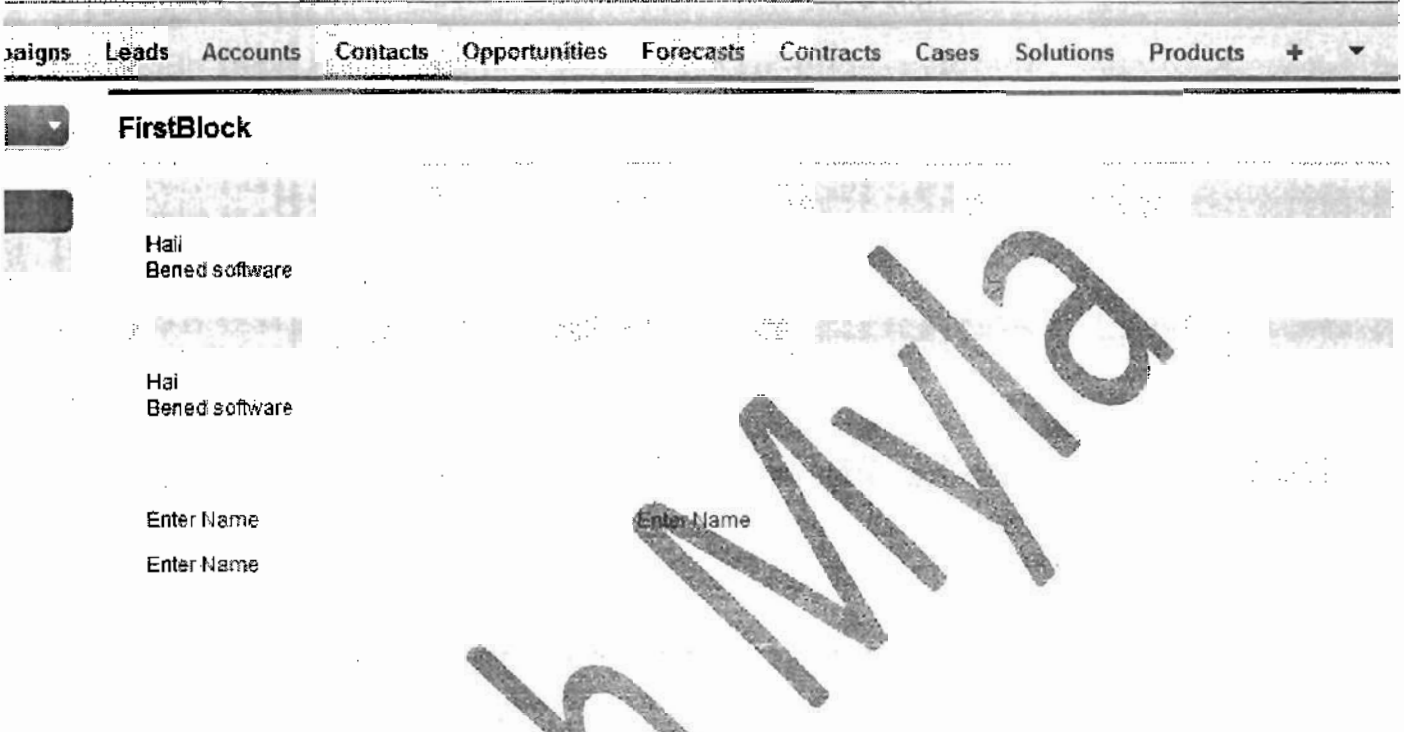
Title : Title that we want to display in the pageBlockSection

Collapsable : This is a Boolean value that specifies whether pageBlockSection is collapsible are not.

Columns :This specifies how many columns should be displayed in a row.

Example:

```
<apex:page >
<apex:form >
  <apex:pageBlock title="FirstBlock" >
     <apex:pageBlockSection title="FirstSection" >
       Haii<br/> Bened software
     </apex:pageBlockSection>
     <apex:pageBlockSection title="Second Section" collapsible="false">
      Hai<br/> Bened software
     </apex:pageBlockSection>
     <apex:pageBlockSection title="Output Labels" >
        <apex:outputLabel >Enter Name</apex:outputLabel>
         <apex:outputLabel >Enter Name</apex:outputLabel>
          <apex:outputLabel >Enter Name</apex:outputLabel>
     </apex:pageBlockSection>
  </apex:pageBlock>
</apex:form>
</apex:page>
```



Input Tags in Visualforce:

1.<apex:inputText>

2.<apex:inputSecret>

3.<apex:inputHidden>

4.<apex:inputCheckbox>

5.<apex:inputTextArea>

6.<apex:selectList>

7.<apex:selectOption>

8.<apex:selectOptions>

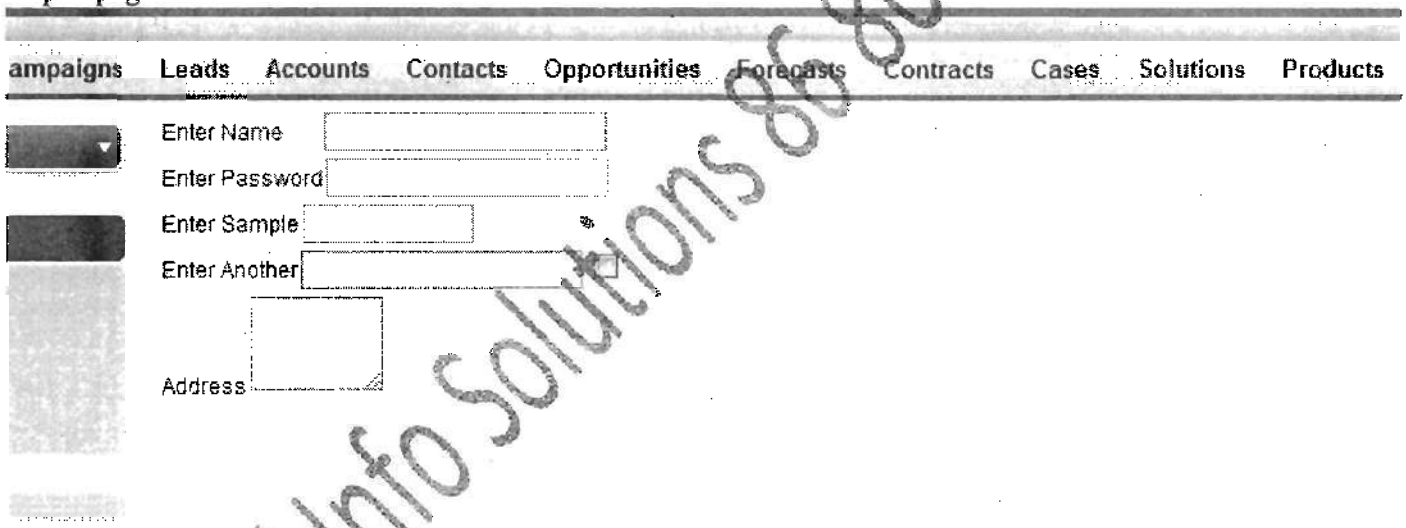
9.<apex:inpuField>

10.<apex:selectRadio>

11.<apex:selectCheckBoxes>

Example:

```
<apex:page >
 <apex:form >
    <apex:outputLabel >Enter Name</apex:outputLabel>
    <apex:inputText tabindex="2"/><br/>
    <apex:outputLabel >Enter Password</apex:outputLabel>
    <apex:inputSecret tabindex="1"/><br/>
    <apex:outputLabel > Enter Sample</apex:outputLabel>
    <apex:inputText size="10"/><br/>
    <apex:outputLabel >Enter Another</apex:outputLabel>
    <apex:inputText maxlength="4"/>
    <apex:inputCheckbox /><br/>
    <apex:outputLabel >Address</apex:outputLabel>
    <apex:inputTextarea cols="10" rows="3"/>
 </apex:form>
</apex:page>
```

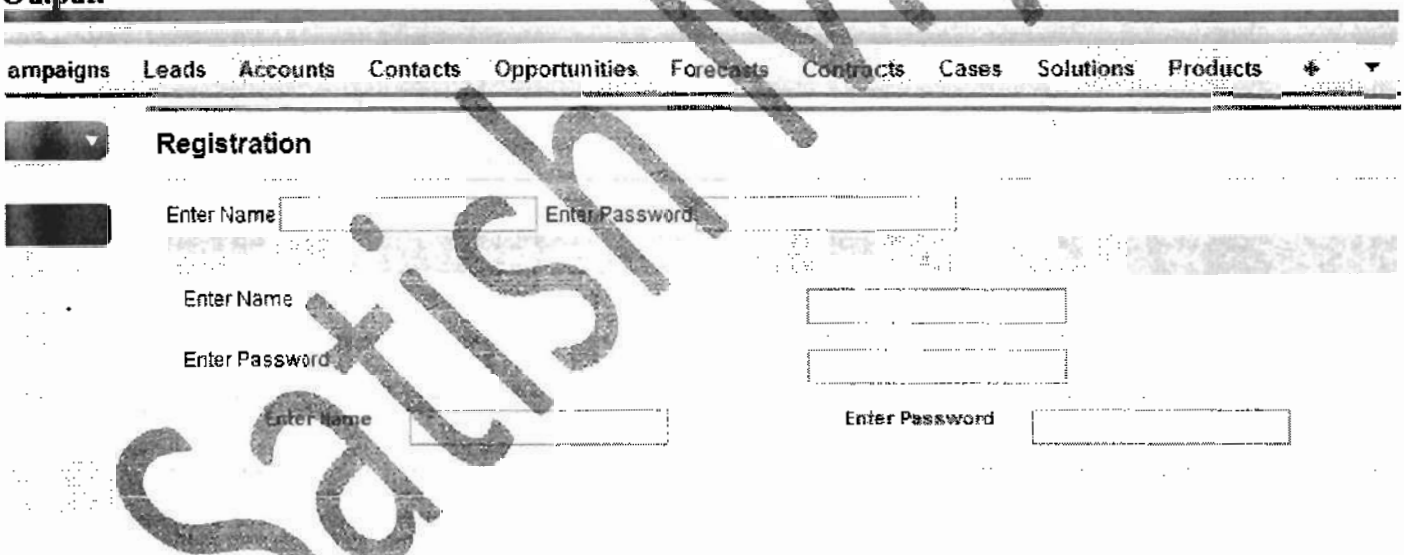


Example 2:

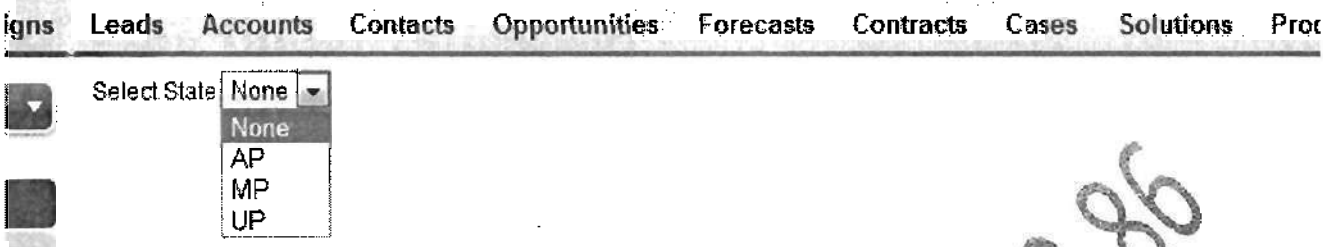
```
<apex:page>
 <apex:form>
   <apex:pageBlock title="Registration">
     <apex:outputLabel >Enter Name</apex:outputLabel>
     <apex:inputText />
     <apex:outputLabel >Enter Password</apex:outputLabel>
     <apex:inputText />
     <apex:pageBlockSection title="MySection">
       <apex:outputLabel >Enter Name</apex:outputLabel>
       <apex:inputText />
```

```
      <apex:outputLabel >Enter Password</apex:outputLabel>
      <apex:inputText />
      <apex:pageBlockSectionItem >
         <apex:outputLabel >Enter Name</apex:outputLabel>
         <apex:inputText />
      </apex:pageBlockSectionItem>
      <apex:pageBlockSectionItem >
          <apex:outputLabel >Enter Password</apex:outputLabel>
           <apex:inputText />
      </apex:pageBlockSectionItem>
    </apex:pageBlockSection>
  </apex:pageBlock>
</apex:form>
</apex:page>
```

Output:

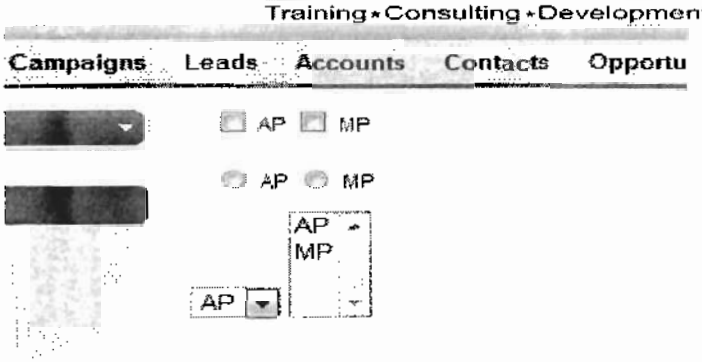



```
<apex:selectList>
<apex:page >
 <apex:form >
 <apex:outputLabel >Select State</apex:outputLabel>
   <apex:selectList size="1">
     <apex:selectOption itemLabel="None" itemValue="none"></apex:selectOption>
     <apex:selectOption itemLabel="AP" itemValue="one"></apex:selectOption>
     <apex:selectOption itemLabel="MP" itemValue="two"> </apex:selectOption>
     <apex:selectOption itemLabel="UP" itemValue="three"></apex:selectOption>
   </apex:selectList>
 </apex:form>
</apex:page>
```



Example :

```
<apex:page >
 <apex:form >
    <apex:selectCheckboxes >
      <apex:selectOption itemLabel="AP" itemValue="AP"/>
      <apex:selectOption itemLabel="MP" itemValue="MP"/>
    </apex:selectCheckboxes>
     <apex:selectRadio >
       <apex:selectOption itemLabel="AP" itemValue="ap"></apex:selectOption>
       <apex:selectOption itemLabel="MP" itemValue="mp"></apex:selectOption>
     </apex:selectRadio>
     <apex:selectList size="1">
       <apex:selectOption itemLabel="AP" itemValue="ap"></apex:selectOption>
       <apex:selectOption itemLabel="MP" itemValue="mp"></apex:selectOption>
     </apex:selectList>
      <apex:selectList multiselect="true">
       <apex:selectOption itemLabel="AP" itemValue="ap"></apex:selectOption>
       <apex:selectOption itemLabel="MP" itemValue="mp"></apex:selectOption>
     </apex:selectList>
 </apex:form>
</apex:page>
```



<apex:inputField>:

This will create a field in the visualforce page with exact properties of the fields what we have we in the object.

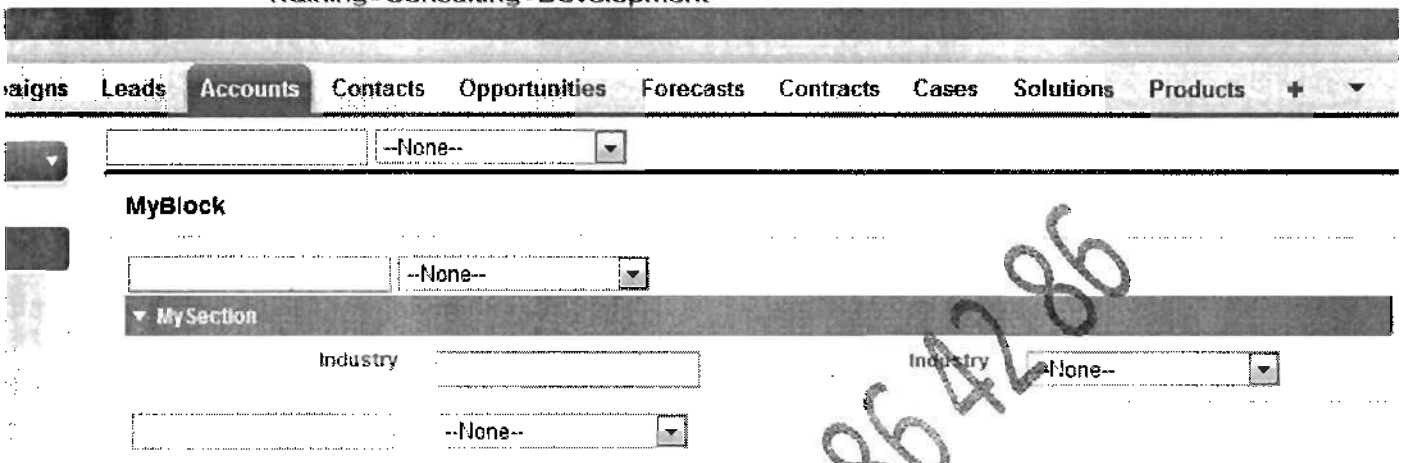
Note:When we want to refer to a particular object in the visualforce page we should use

"standardController= objectname"

Example:

```
<apex:page standardController="Account">
 <apex:form >
    <apex:inputText value="{!Account.Industry}"/>
    <apex:inputField value="{!Account.Industry}"/>
    <apex:pageBlock title="MyBlock">
      <apex:inputText value="{!Account.Industry}"/>
      <apex:inputField value="{!Account.Industry}"/>
      <apex:pageBlockSection title="MySection">
        <apex:inputText value="{!Account.Industry}"/>
        <apex:inputField value="{!Account.Industry}"/>
        <apex:pageBlockSectionItem >
          <apex:inputText value="{!Account.Industry}"/>
          <apex:inputField value="{!Account.Industry}"/>
        </apex:pageBlockSectionItem>
      </apex:pageBlockSection>
    </apex:pageBlock>
 </apex:form>
</apex:page>
```

Output :

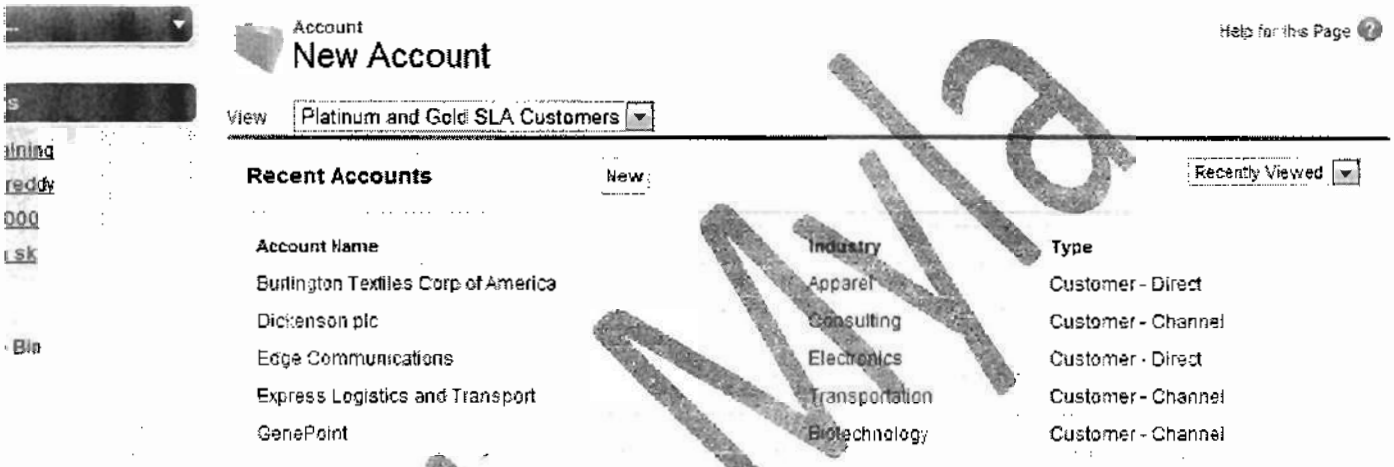


Create a Account Home Page:

```
<apex:page standardController="Account"  recordSetVar="items">
 <apex:sectionHeader title="Account" subtitle="New Account" help="http://www.google.com"/>
 <apex:form >
  <apex:outputLabel >View</apex:outputLabel>
      
    <apex:selectList size="1" value="{!filterid}">
       <apex:selectOptions value="{!listviewoptions}"/>
       <apex:actionSupport event="onchange" reRender="pb"/>
    </apex:selectList>
   <apex:pageBlock title="Recent Accounts" id="pb">
     <apex:pageBlockButtons location="top" >
       <apex:commandButton value="New" action="{!create}"/>
       <span style="float:right;">
         <apex:selectList size="1" >
           <apex:selectOption itemLabel="Recently Viewed" itemValue="one"/>
           <apex:selectOption itemLabel="Recently Modified" itemValue="two"/>
           <apex:selectOption itemLabel="Recently Created" itemValue="three"/>
         </apex:selectList>
       </span>
     </apex:pageBlockButtons>
     <apex:pageBlockTable value="{!items}" var="a" rows="5">
       <apex:column value="{!a.name}"/>
       <apex:column value="{!a.industry}"/>
       <apex:column value="{!a.type}"/>
     </apex:pageBlockTable>
```

```
  </apex:pageBlock>
 </apex:form>
</apex:page>
```

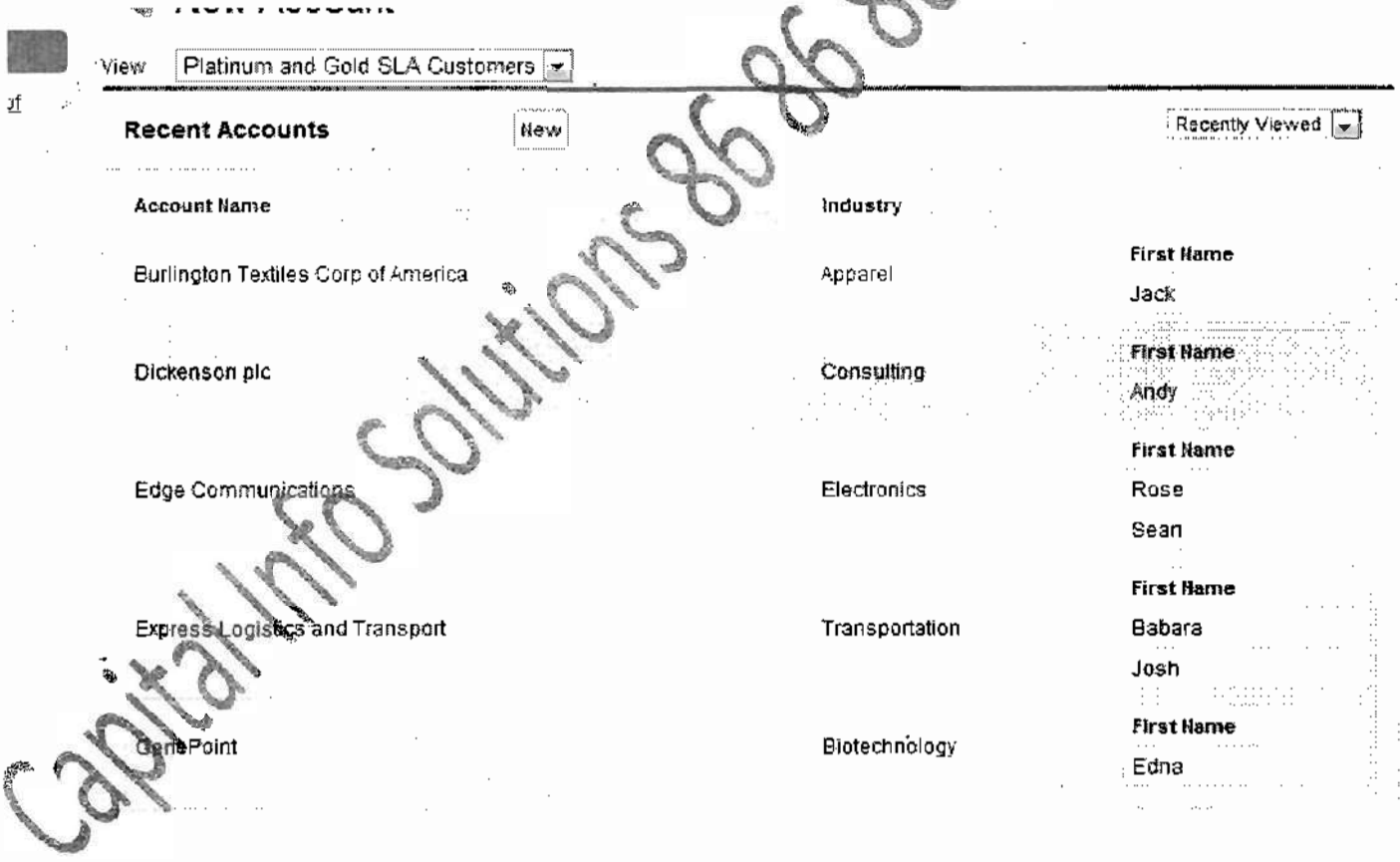
Output Screen :



Example :Create a visualforce page to display Account records with their respective contact details using pageblockTable

```
<apex:page standardController="Account" recordSetVar="items">
 <apex:sectionHeader title="Account" subtitle="New Account" help="http://www.google.com"/>
 <apex:form >
  <apex:outputLabel >View</apex:outputLabel>
      
    <apex:selectList size="1" value="{!filterid}">
      <apex:selectOptions value="{!listviewoptions}"/>
      <apex:actionSupport event="onchange" reRender="pb"/>
    </apex:selectList>
  <apex:pageBlock title="Recent Accounts" id="pb">
    <apex:pageBlockButtons location="top" >
      <apex:commandButton value="New" action="{!create}"/>
      <span style="float:right;">
        <apex:selectList size="1" >
          <apex:selectOption itemLabel="Recently Viewed" itemValue="one"/>
          <apex:selectOption itemLabel="Recently Modified" itemValue="two"/>
          <apex:selectOption itemLabel="Recently Created" itemValue="three"/>
        </apex:selectList>
      </span>
    </apex:pageBlockButtons>
```

```
    <apex:pageBlockTable value="{!items}" var="a" rows="5">
      <apex:column value="{!a.name}"/>
      <apex:column value="{!a.industry}"/>
      <apex:column >
      <apex:pageBlockTable value="{!a.contacts}" var="b">
         <apex:column value="{!b.firstname}"/>
      </apex:pageBlockTable>
     </apex:column>
    </apex:pageBlockTable>
  </apex:pageBlock>
 </apex:form>
</apex:page>
```



➔ Creating output Lables and TextFields in in VF page

1.<apex:outputLabel> : This will create label on the page .

Ex:<apex:outputLabel> LabelName1</apex:outputLabel>

Ex: <apex:outputLabel value="Label Name 2"/>

Note : we create the new custom label under create and use them in visualforce page

Steps to create New Custom Label :

1. setup-->Build-->Create-->Custom Label--> New
2. Enter Label Details
   Ex: Name :Test Label
   Label :NewLabel
3. <apex:outputLable value="{!$Label.TestLabel}"/>

➔ **Javascript :** when we want to declare any scripting code we have to write with in <script> tag

Syntax:

```
<apex:page
    <script>
       // body code
    </script>
</apex:page>
```

Popup Boxes : This will open new popup window and print message on it.

Alert Box : An alert box is often used if you want to make sure information comes through to the user.When an alert box pops up, the user will have to click "OK" to proceed.

Ex: alert('Helloworld');

Confirm Box : A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Ex: var r=confirm("Press a button");

Note: The statements written in the <Script> are executed in the top down passion when the page is loaded

Example :

```
<apex:page>
    <apex:outputLabel> This is line one</apex:outputLabel>
    <script>
        aler('This is line two');
    </script>
    <apex:outputLabel> This is line Three</apex:outputLabel>
<apex:page>
```

Varibales in javascript: All the variables in the javascript are declared as var;

```
var a=10;
var name='sam';
var ages=new Array(10);
```

Arrays in javascript :

```
var arrayname=new Array(size);
Ex: var names=new Array(){'sam','ram'};
Ex: var ages=new Array();
    ages[0]=10;
    ages[1]=20;
```

Objects in Javascript:

```
Ex: var student={name:'sam',age:27};
Ex: var emp=new Object();
    emp.name='Prasad';
    cmp.age=27;
```

Functions in javascript:

syntax:

```
function functionname( parameters)
{
// body ;
}
```

```
Ex: function show()
{
  var name='hello';
  alert(name);
}
```

Note: javascript function will be invoked only when some event occur events like onclick, onfocus,onblur,onchage,ondclick etc.

Example :

```
<apex:page >
 <apex:form >
   <script>
     function show()
     {
```

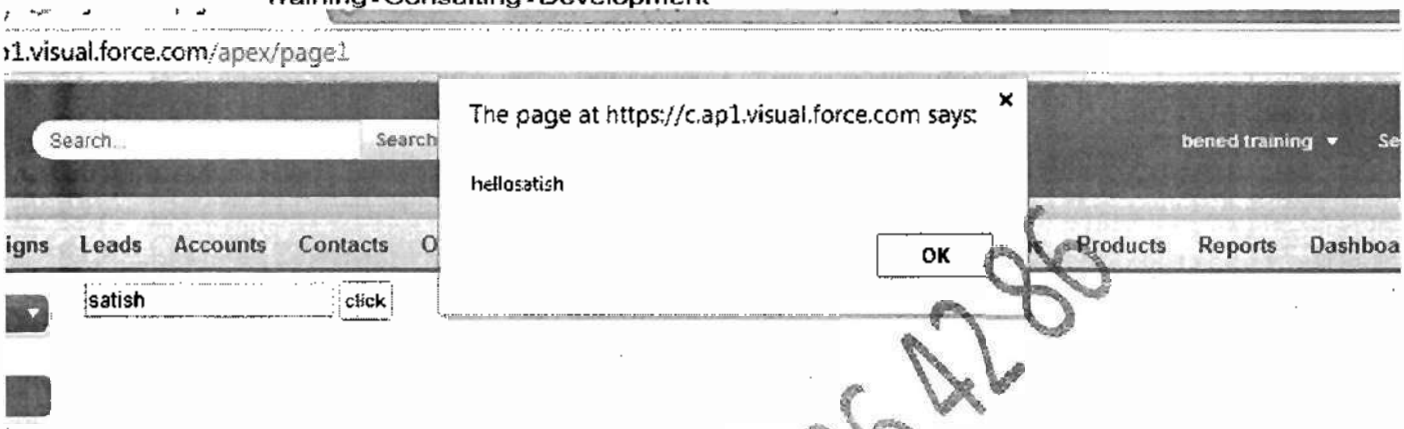
```
        alert('hello');
      }
    </script>
    <apex:commandButton value="click" onclick="show()"/>
  </apex:form>
</apex:page>
```

Output Screen :

Example: Reading a html component values in the javascript

```
<apex:page >
  <apex:form >
    <script>
      function show()
      {
       var name=document.getElementById("one").value;
        alert('hello'+name);
      }
    </script>
    <input type="text" id="one" onchange="show()"/>
    <apex:commandButton value="click" onclick="show()"/>
  </apex:form>
</apex:page>
```

Output Screen :



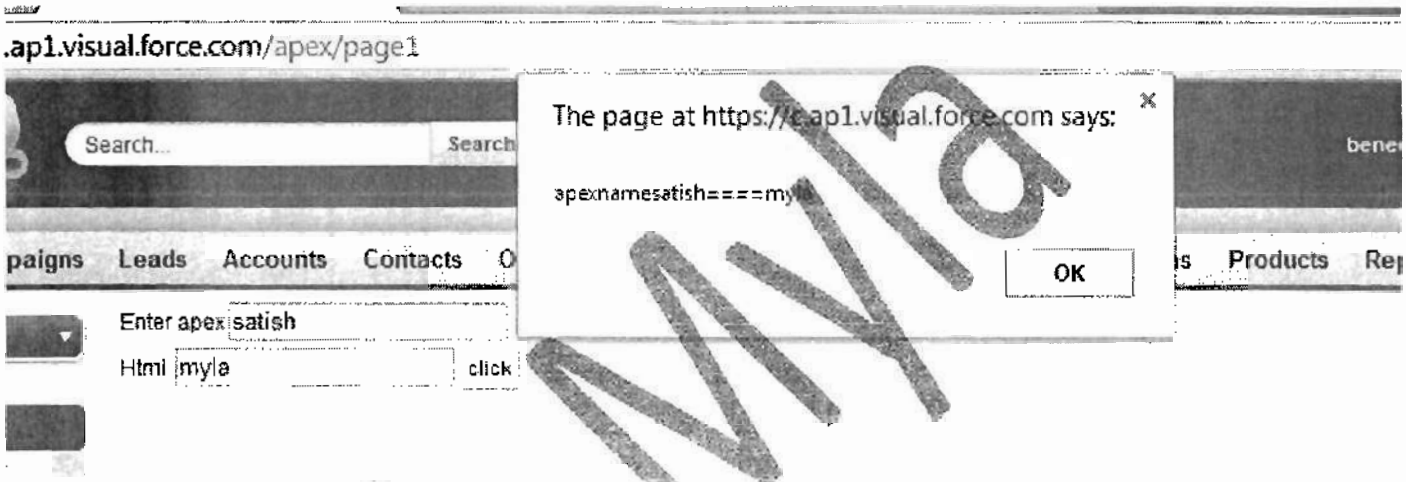
==>$Component: This is a global object which is used to refer to a visualforce component in the java script .
-->Every component in the visualforce page will have separate id .
$Component.id
ex: document.getElementById('{!$Component.id}').value;
Ex 2: This is an example to read html input text value and Visualforce component
<apex:inputText > value in the javascript

Example : To read the apex component values using $component in javascript

```
<apex:page >
  <apex:form >
    <script>
     function show()
       {
     var apexname=document.getElementById('{!$Component.one}').value;
     var name=document.getElementById("two").value;

     alert('apexname'+apexname+'====='+name);

    </script>
     <apex:outputLabel >Enter apex</apex:outputLabel>
     <apex:inputText id="one" />
      <br/>
     <apex:outputLabel >Html </apex:outputLabel>
     <input type="text" id="two"  />
      <apex:commandButton value="click" onclick="show()"/>
  </apex:form>
```

</apex:page>

Output Screen :



Note: when the script and visualforce component are not at the same level then component should be refered with complete path

Ex: Page

```
|
|----Form
|       |
|       |----Script
|       |
|       |----InputText id="one"
|       |
|       |----PageBlock id="PB"
|       |       |
|       |       |----inputText id="two"
|       |       |
|       |       |----PageBlockSection id="PBS"
|       |       |       |
```

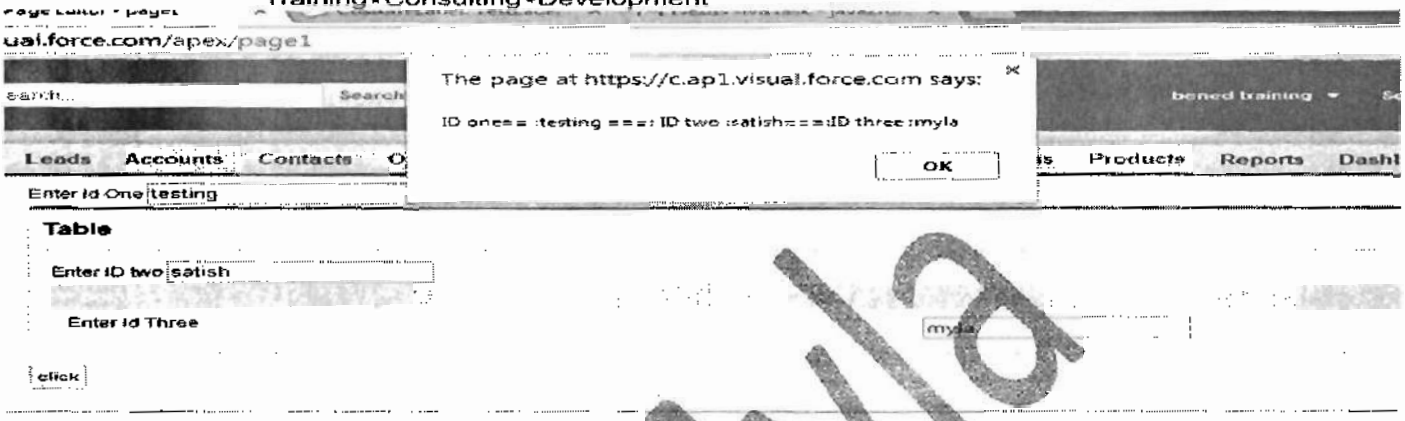
```
|        |---inputText id="three"
|        |
|
```

```
document.getElementById('{!$Component.one}').value;
document.getElementById('{!$Component.PB.two}').value;
document.getElementById('{!$Component.PB.PBS.three}').value;
```

Example:: This is to display the different component from different sections

```
<apex:page >
   <apex:form >
     <script>
         function show()
          {
           var one=document.getElementById('{!$Component.one}').value;
           var two=document.getElementById('{!$Component.PB.two}').value;
           var three=document.getElementById('{!$Component.PB.PBS.three}').value;
           alert('ID one== :'+one+'===: ID two :'+two+'===:ID three :'+three);
          }
   </script>
   <apex:outputLabel >Enter Id One</apex:outputLabel>
   <apex:inputText id="one"/>
    <apex:pageBlock id="PB" title="Table">
      <apex:outputLabel > Enter ID two</apex:outputLabel>
      <apex:inputText id="two"/>
      <apex:pageBlockSection id="PBS" title="Section 1">
        <apex:outputLabel > Enter Id Three </apex:outputLabel>
        <apex:inputText id="three"/>
      </apex:pageBlockSection>
    </apex:pageBlock>
   <apex:commandButton value="click" onclick="show()"/>
  </apex:form>
 </apex:page>
```

Output Screen :

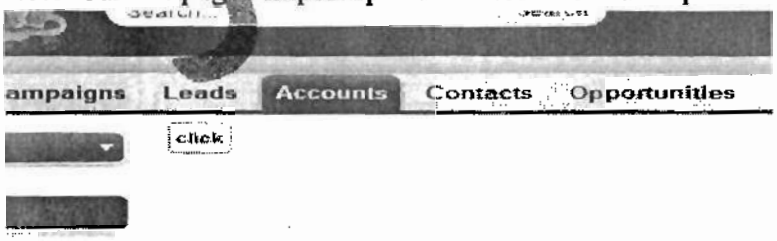


Example: Create a visualforce page to display the firstName of the current user and his profile in popup box using javascript

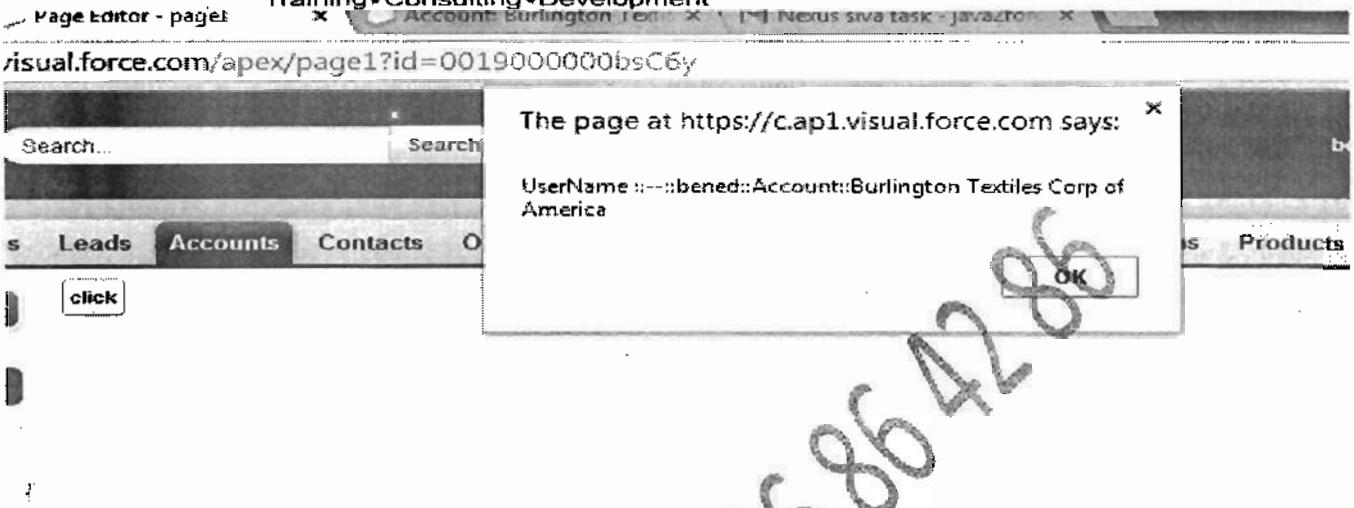
Page: JExample

```
<apex:page>
  <apex:form>
   <script>
    function show()
    {
     var username='{!$User.firstName}';
     var accountname='{!Account.Name}';
     alert('UserName ::-::'+username+'::Account::'+ accountname);
    }
   </script>
    <apex:commandButton value="click" onclick="show()"/>
  </apex:form>
</apex:page>
```

Note :Call the page : https://ap1.salesforce.com/JExample?id=0019000000bsC6y

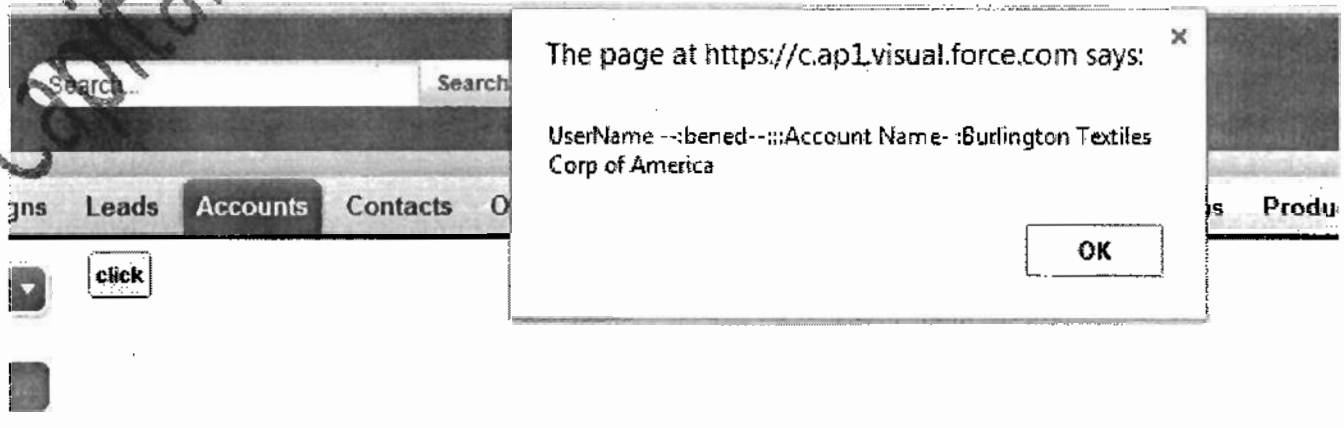


When we click on the button



Example : Pass the parameters to javascript from visualforce component:

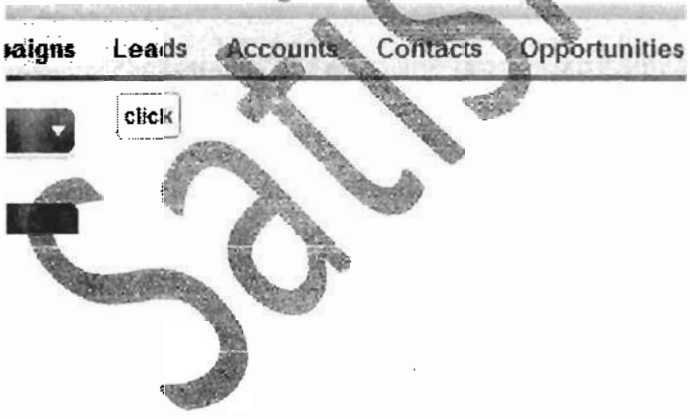
```
<apex:page standardController="Account" >
 <apex:form >
  <script>
   function show(userName,accountname)
   {
    alert('UserName --:'+userName+'--:::Account Name- :'+accountname);
   }
  </script>
 <apex:commandButton value="click"
      onclick="show('{!$User.FirstName}','{!Account.Name}')"/>
 </apex:form>
</apex:page>
```



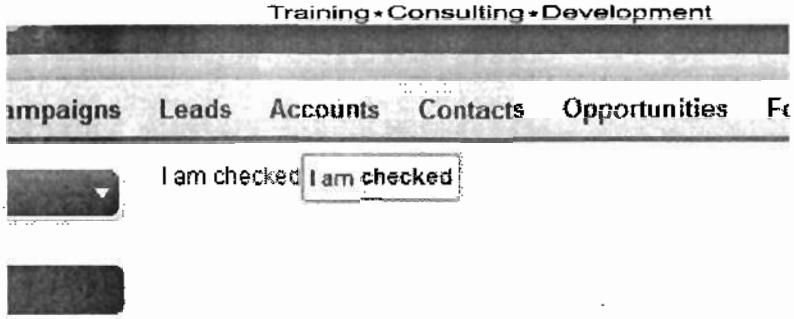
Example: Change the CSS properties /component values of the Visualorce page using javascript

```
<apex:page >
<apex:form >
<script>
 function show()
  {
   var abc='{!$User.FirstName}';
   document.getElementById('{!$Component.two}').style.width='100px';
   document.getElementById('{!$Component.two}').innerHTML='I am checked';
   document.getElementById('{!$Component.three}').value='I am checked';
 }
</script>
 <apex:outputLabel id="two"> </apex:outputLabel>
 <apex:commandButton id="three" value="click" oncomplete="show()"/>
</apex:form>
</apex:page>
```

Screen 1:



When we click on the button :



Validations in javascript :

Regular Expression : Regular expressions are patterns used to match character combinations in strings. In JavaScript, regular expressions are also objects.

These patterns are used with the exec and test methods of RegExp, and with the match, replace, search, and split methods of String.

This chapter describes JavaScript regular expressions.

➔We can create a regular expressions in two ways:

1. Var reg= /ab+c/;
   Calling the constructor function of the RegExp object, as follows:
2. Var reg= new RegExp('ab+c');
   Pattern in creating a regular expression

1. ^ : Matches beginning of input.
   ex: ^a : This would accept any string which starts with 'a';
2. [] : This would specify the domain range of the valus from which i chosse to form a expression
   ex: [ABC] : I can choose any one value from A, B, or C
   i:p: A -accepted
   B-Accepted
   C-Accepted
   D-Not accepted
3. {} : This will specify how many charecters
   ex: [ABC]{2}
   This will accept any set of characters that are formed from [ABC]
   ex: Expression that accpets indian phone no
   [789][0-9]{9}
4. * : This will accpet zero and any no of give expressions
   Ex:* a== this will accept { null,a,aa,aaa,aaaa,.....}

5. + : This will accept one or more no of charectes
ex: /a+/ =={a,aa,aaa,aaaa,....}

6. ? : This will accept zero or one time of expression
ex:/ ab?c /
ac--accepted
abc-accepted
abbc-rejected

7.$: This indiacates the expression should end with prefixed character
ex: /a$/: This will accept any string that ends with 'a';

8.\d : This will accept any digit
ex: /a\d c/
a1c ==accepted
a3d==accepted
aac==rejected
ex: /a\d{2} c/ : this will accept any expression with a follwed by two digits and followed by c
a12c --accepted
a34c-accepted
a3c-rejected

9. \D : This will accept any thing other than digit :
ex: /a\D/ :
aa--accepted
a1--rejected

10: x(?=y) :This will accpet x only if x is followed by Y
ex: /(ab)(?=cd)/
abcd==accepted
abxy==rejected
xyac==accepted

11. x(?!y) : This will accept x only if x is not followed by Y

12: \w :This will accept any alpha numeric format

13: \W :This will accept any thing other than alphanumeric

```
<apex:page id="pg">
  <apex:form id="fm">
 <script>
 function show()
```

```
{
 var myname=document.getElementById('pg:fm:one').value;
 var reg=/^a[0-9]{3}-[A-Z]{2}$/;
 if(reg.test(myname))
 {
   alert('hello'+myname);
}
}
</script>
<apex:inputText id="one"/>
<apex:commandButton value="click" onclick="show()"/>
</apex:form>
</apex:page>
```

Method Description

1.exec: A RegExp method that executes a search for a match in a string.
It returns an array of information.

```
var myRe = /ab*/g;
var str = "abbcdefabh";
var myArray;
while ((myArray = myRe.exec(str)) !== null)
  {
  var msg = "Found " + myArray[0] + ".  ";
  msg += "Next match starts at " + myRe.lastIndex;
  console.log(msg);
  }
```

2.test : A RegExp method that tests for a match in a string.
It returns true or false.

```
function testinput(str)
{
 var re=/^abc/;
 var midstring;
 if (re.test(str))
   {
   midstring = " contains ";
  }
 else {
   midstring = " docs not contain ";
```

```
    }
  }
```

3.match : A String method that executes a search for a match in a string.
It returns an array of information or null on a mismatch.

4.search : A String method that tests for a match in a string.
It returns the index of the match, or -1 if the search fails.

5.replace : A String method that executes a search for a match in a string,
and replaces the matched substring with a replacement substring.

6.split: A String method that uses a regular expression or a fixed string to break a string into an array of substrings.

➔Creating a picklist field in the visualforce page

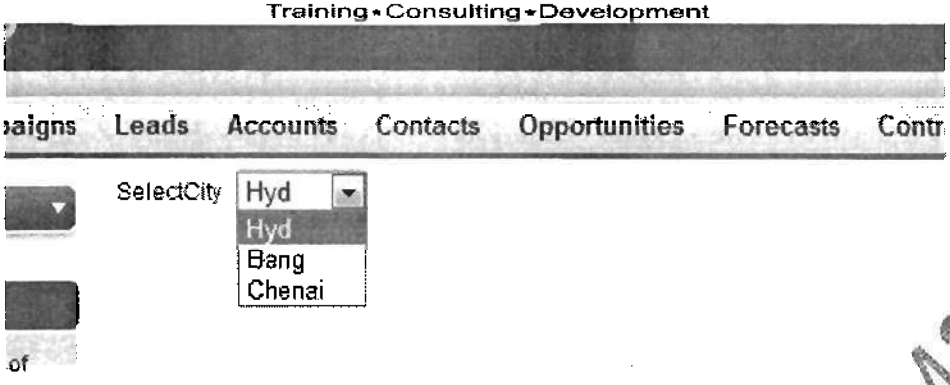
1.apex:selectList

2.apex:selectOption

3.apex:selectOptions

Example: create a picklist field city with the options {hyd,bang,chenai}

```
<apex:page >
 <apex:form >
   <apex:outputLabel >SelectCity  </apex:outputLabel>
   <apex:selectList size="1">
      <apex:selectOption itemLabel="Hyd" itemValue="one"/>
      <apex:selectOption itemLabel="Bang" itemValue="two"/>
      <apex:selectOption itemLabel="Chenai" itemValue="three"/>
   </apex:selectList>
 </apex:form>
</apex:page>
```



Example: Create a picklist field with the listview options for the given object

```
<apex:page standardController="Account" recordSetVar="items">
 <apex:form >
    <apex:outputLabel >View  </apex:outputLabel>
        <apex:selectList size="1" value="{!filterId}">
            <apex:selectOptions value="{!listviewoptions}"/>
         </apex:selectList>
 </apex:form>
</apex:page>
```

==>Creating a commandLink in the visualforce page

```
<apex:page standardController="Account" recordSetVar="items">
 <apex:form >
    <apex:commandLink value="click" action="http://www.google.com"/>
 </apex:form>
</apex:page>
```

➔<apex:pageBlockTable> :
A list of data displayed as a table within an <apex:pageBlock> or <apex:pageBlockSection> component, similar to a related list or list view in a standard Salesforce page. Like an <apex:dataTable>, an <apex:pageBlockTable> is defined by iterating over a set of data,. The Set of data can contain up to 1,000 items.

<apex:pageBlockTable> :This is used to display the list of records by rows wise format in table.

value : This is list of records which need to be displayed.

var : This is variable which will act as cursor .By defualt it points to first record and moves

till the last record.

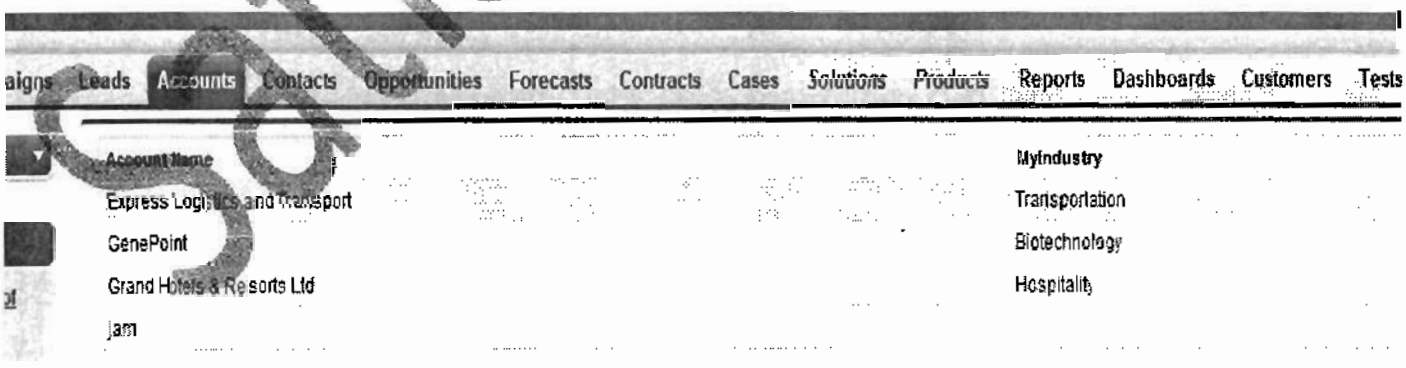
Note: <apex:page standardController="Account" recordServar="items" recordserVar will fetches the records from the standard controller based on the recently used filter in the Standrardcontroller.

first : This will indicate from which row the records should be displayed in pageBlockTable
rows: This will specify how many rows should be displayed in the pageBlockTable

Ex: Create a visualforce page to display 4 records from third recor from Account object based on recently used filter

```
<apex:page standardController="Account" recordSetVar="items">
  <apex:pageBlock >
    <apex:pageBlockTable value="{!items}" var="a" first="3" rows="4">
      <apex:column value="{!a.name}"/>
      <apex:column headerValue="MyIndustry">
          {!a.industry}
      </apex:column>
    </apex:pageBlockTable>
  </apex:pageBlock>
</apex:page>
```

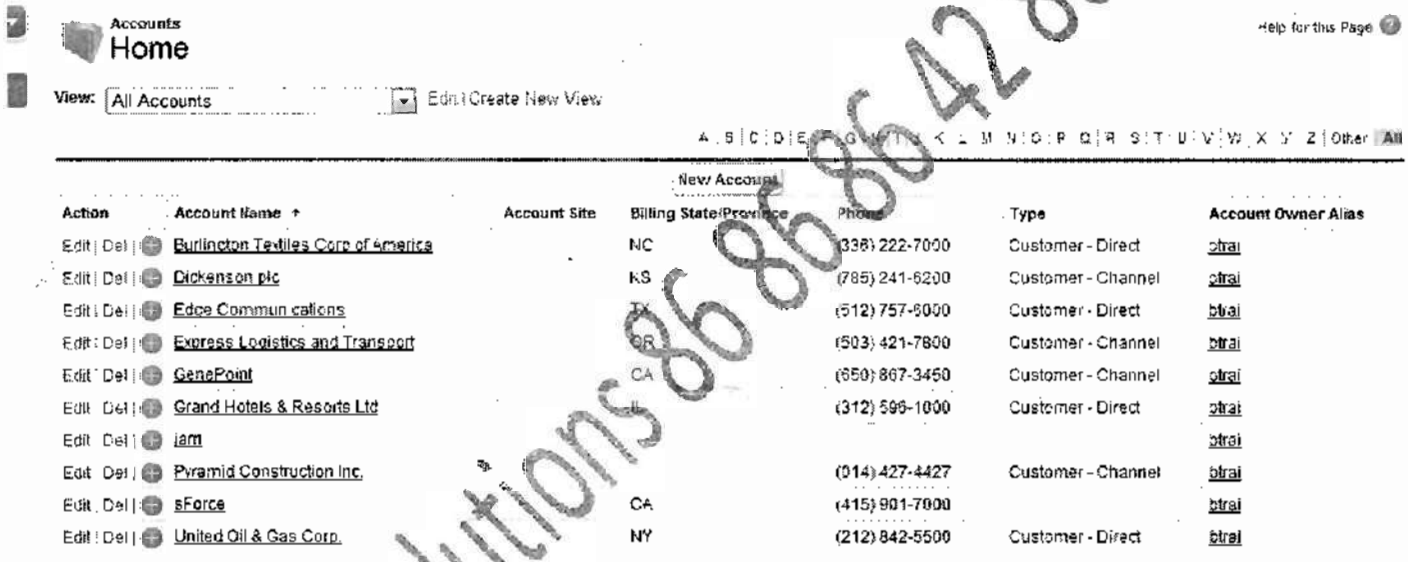
Output Screen :



Example : Create Account Object list view page with out using pageBlockTable

```
<apex:page standardController="Account" recordSetVar="items">
```

```
<apex:sectionHeader title="Accounts" subtitle="Home"
                    help="https://wiki.developer.force.com"/>
  <apex:listViews type="Account"/>
</apex:page>
```

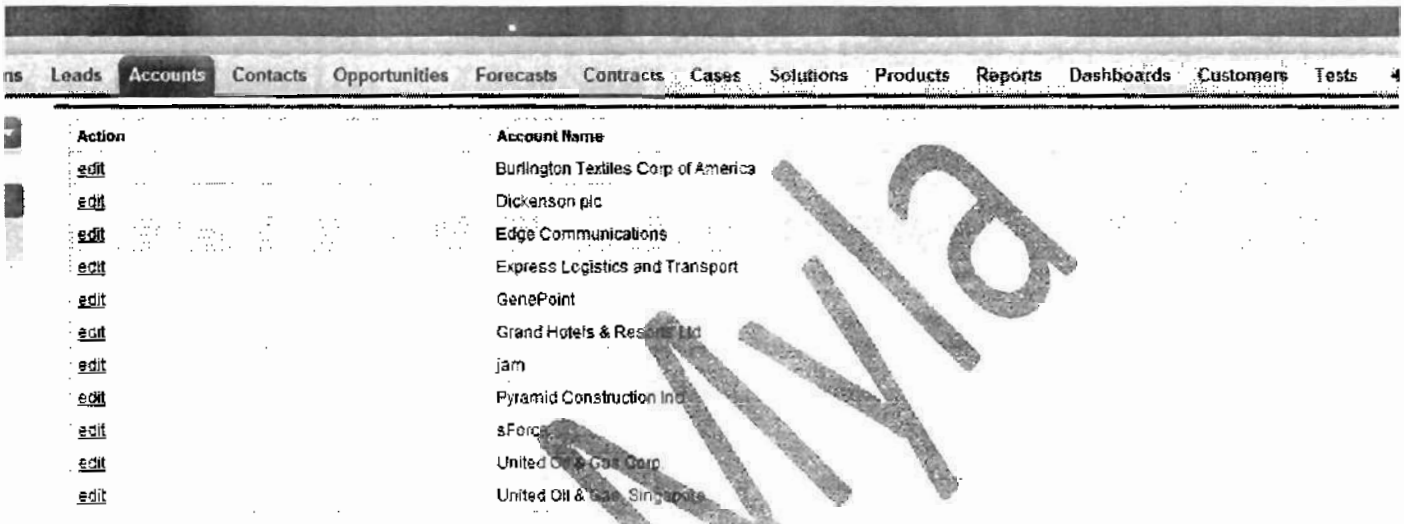


<apex:facet> :

This will print the header and footer values for a component

```
<apex:page standardController="Account" recordSetVar="items">
  <apex:form >
   <apex:pageBlock >
    <apex:pageBlockTable value="{!items}" var="a">
     <apex:column >
       <apex:facet name="header"> Action</apex:facet>
        <apex:commandLink value="edit" action="/{!a.id}/e?retURL={!a.id}" />
        <apex:facet name="footer"> This myvfooter</apex:facet>
      </apex:column>
      <apex:column value="{!a.name}"/>
    </apex:pageBlockTable>
   </apex:pageBlock>
  </apex:form>
```

```
</apex:page>
```

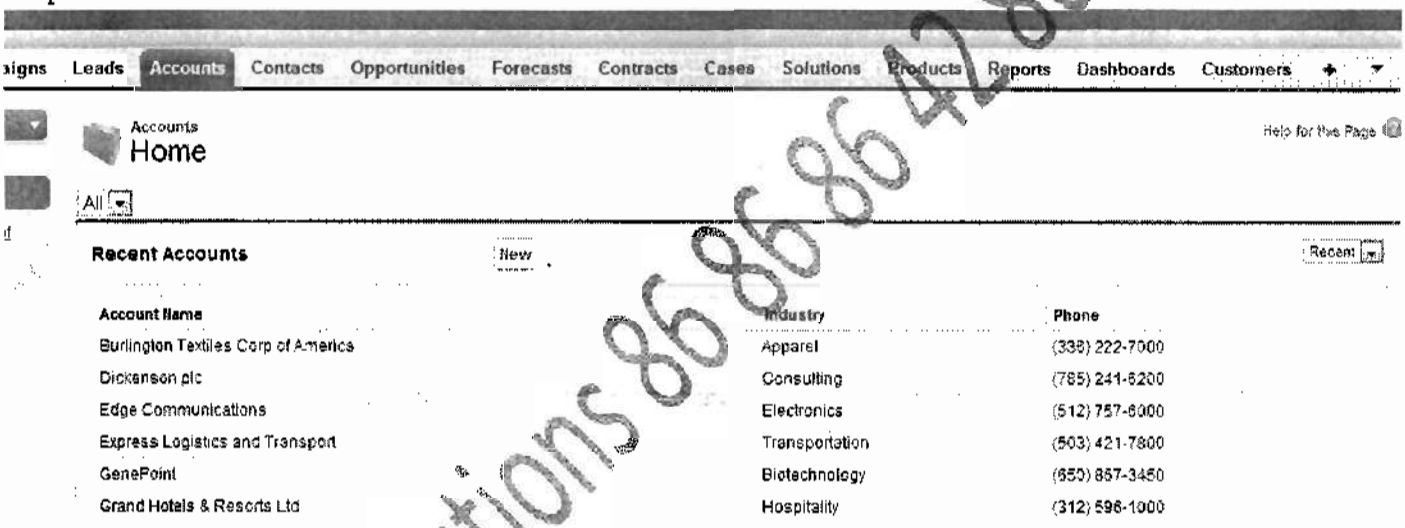


Example :Create Account Object Home Page:

```
<apex:page standardController="Account" recordSetVar="items">
 <apex:sectionHeader title="Accounts" subtitle="Home"
                                 help="https://wiki.developer.force.com"/>
 <apex:form >
   <apex:selectList value="{!filterid}" size="1">
     <apex:selectOptions value="{!listviewoptions}"></apex:selectOptions>
       <apex:actionSupport event="onchage" rendered="pb"/>
   </apex:selectList>
<apex:pageBlock title="Recent Accounts" id="pb">
   <apex:pageBlockButtons >
    <apex:commandButton value="New" action="{!create}"/>
    <span style="float:right;">
      <apex:selectList size="1">
        <apex:selectOption itemLabel="Recent" itemValue="one"/>
        <apex:selectOption itemLabel="Last"  itemValue="two"/>
      </apex:selectList>
     </span>
   </apex:pageBlockButtons>
  <apex:pageBlockTable value="{!items}" var="a">
    <apex:column value="{!a.name}"/>
    <apex:column value="{!a.industry}"/>
```

```
    <apex:column value="{!a.phone}"/>
    </apex:pageBlockTable>
   </apex:pageBlock>
  </apex:form>
</apex:page>
```

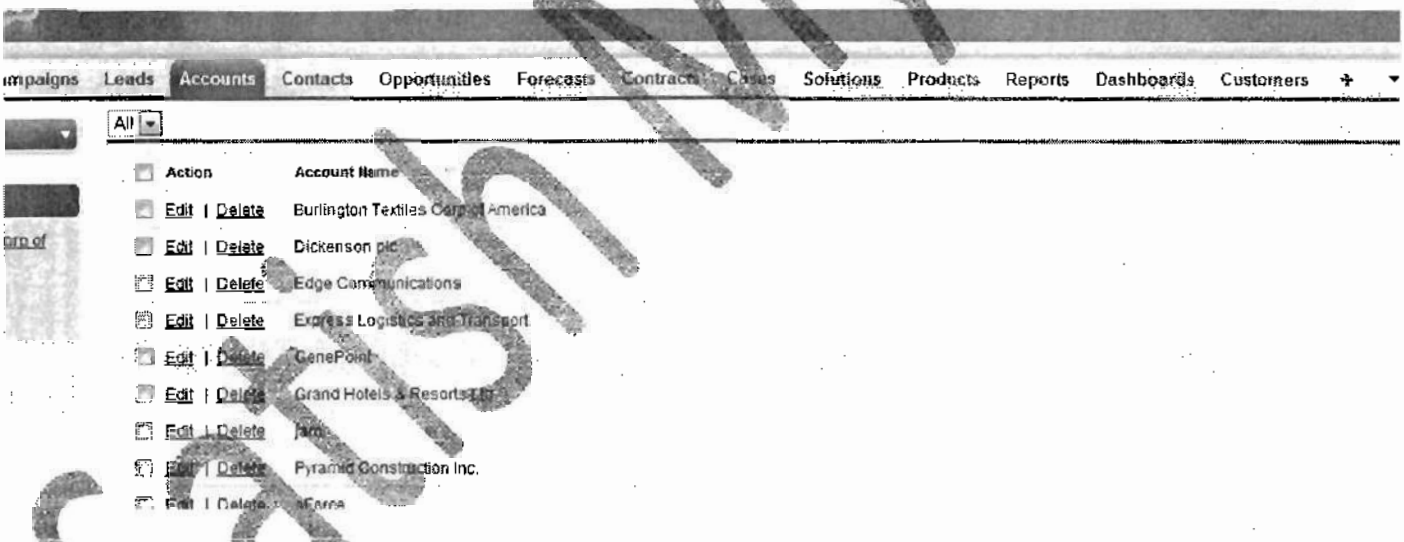
Output Screen :



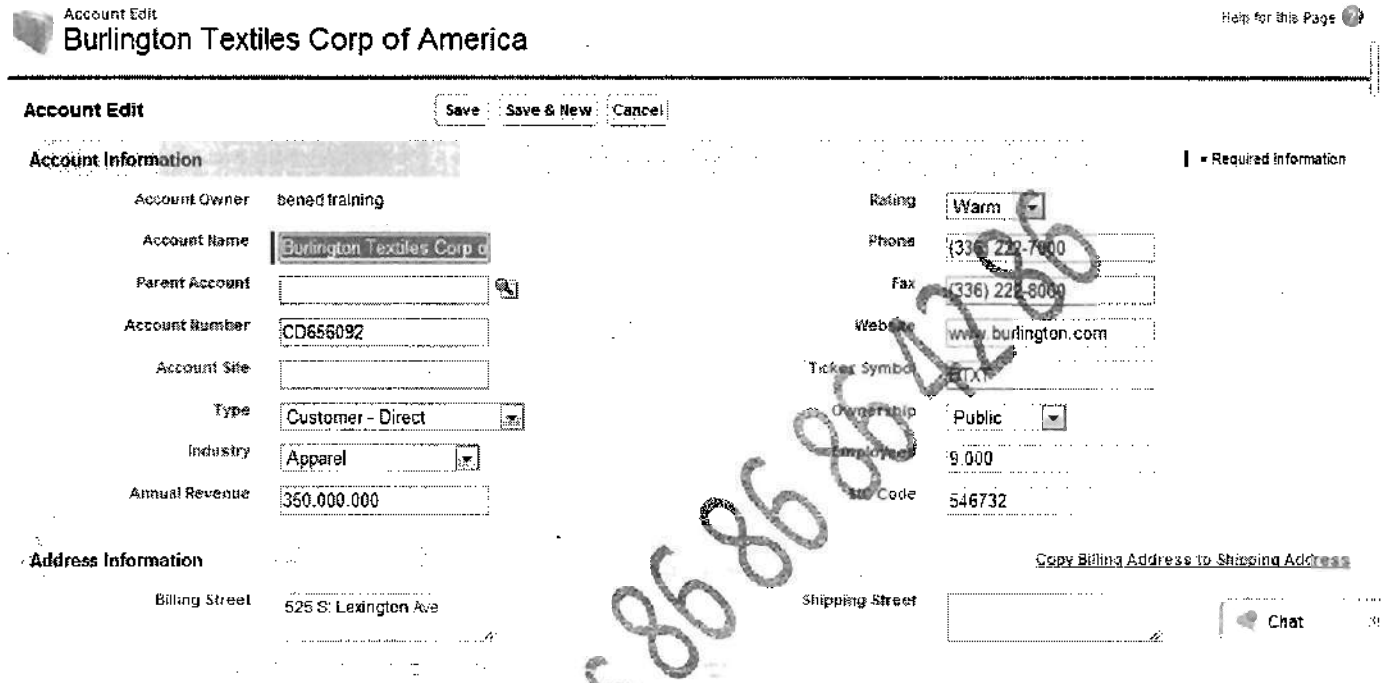
Example: Create account object list view page along with all the functionality

```
<apex:page standardController="Account" recordSetVar="items">
   <apex:form >
     <apex:selectList value="{!filterid}" size="1">
        <apex:selectoptions value="{!listviewoptions}"/>
        <apex:actionSupport event="onchange"  reRender="one"/>
     </apex:selectList>
     <apex:pageBlock >
      <apex:pageBlockTable value="{!items}" var="a" id="one">
       <apex:column width="5px">
        <apex:facet name="header">
          <apex:inputCheckbox />
        </apex:facet>
          <apex:inputCheckbox />
      </apex:column>
      <apex:column width="90px;">
```

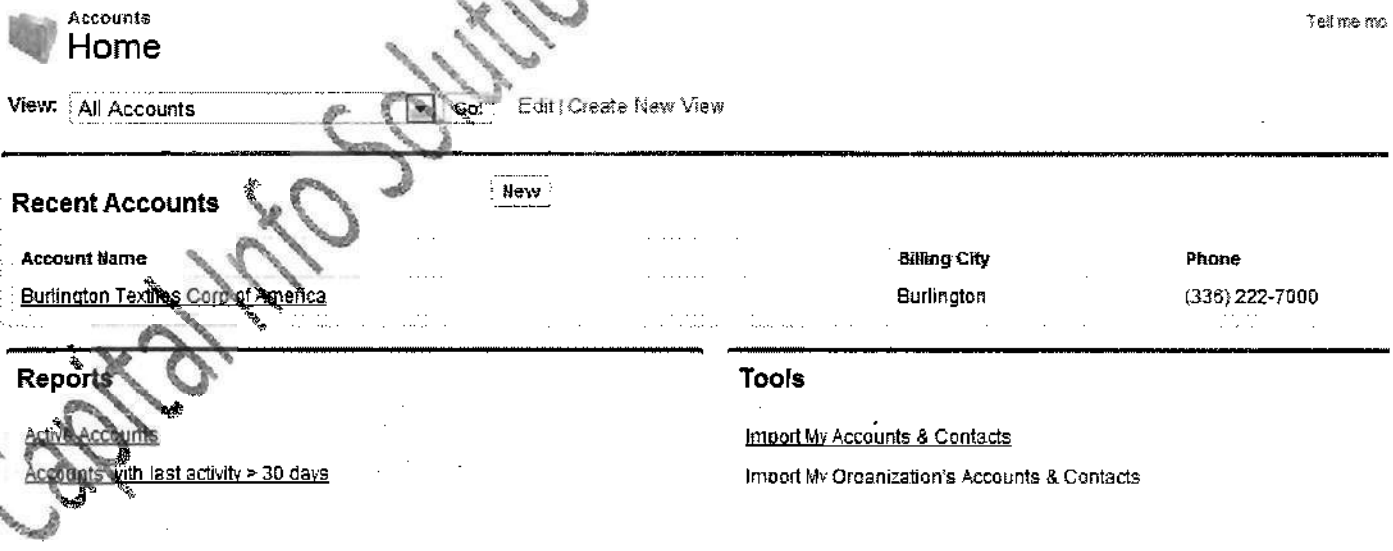
```
        <apex:facet name="header">
            Action
         </apex:facet>
        <apex:commandLink value="Edit" action="/{!a.id}/e?retURl={!a.id}"/>
            |  
        <apex:commandLink value="Delete" action="URLFOR ($Action.Account.delete,a.id)"/>
      </apex:column>
      <apex:column value="{!a.name}"/>
    </apex:pageBlockTable>
  </apex:pageBlock>
  </apex:form>
</apex:page>
```



When we click edit button of Burlington Textiles corp of Amreica



When we click on delete button of Iam record



➔<apex:dataTable> :

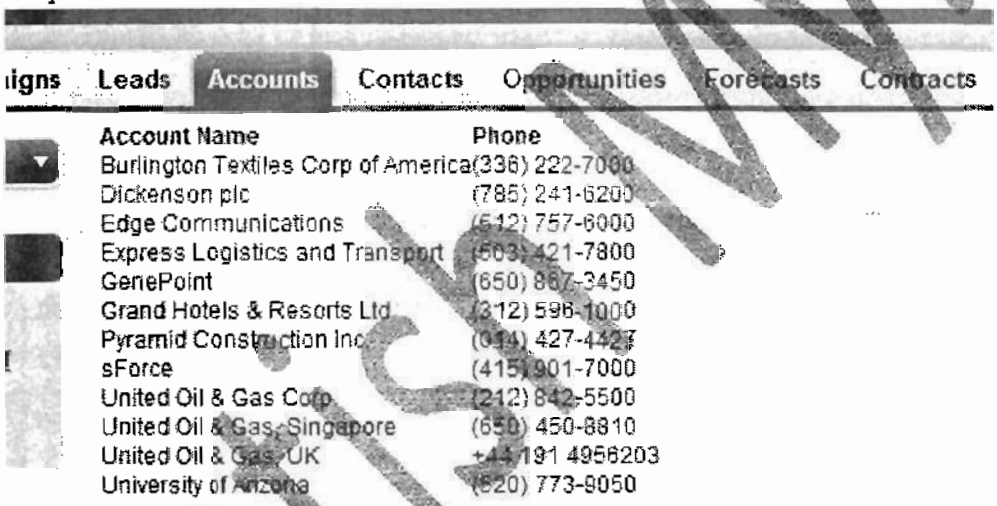
This is used to display the records in the table column wise .

Example :

```
<apex:page standardController="Account" recordSetVar="items">
  <apex:dataTable value="{!items}" var="a">
```

```
    <apex:column headerValue="Account Name">
     {!a.name}
    </apex:column>
    <apex:column >
        <apex:facet name="header">Phone</apex:facet>
        {!a.phone}
    </apex:column>
  </apex:dataTable>
</apex:page>
```

Output Screen :



➔ <apex:dataList> :

An ordered or unordered list of values that is defined by iterating over a set of data. The body of the < apex:dataList > component specifies how a single item should appear in the list. The data set can include up to 1,000 items.

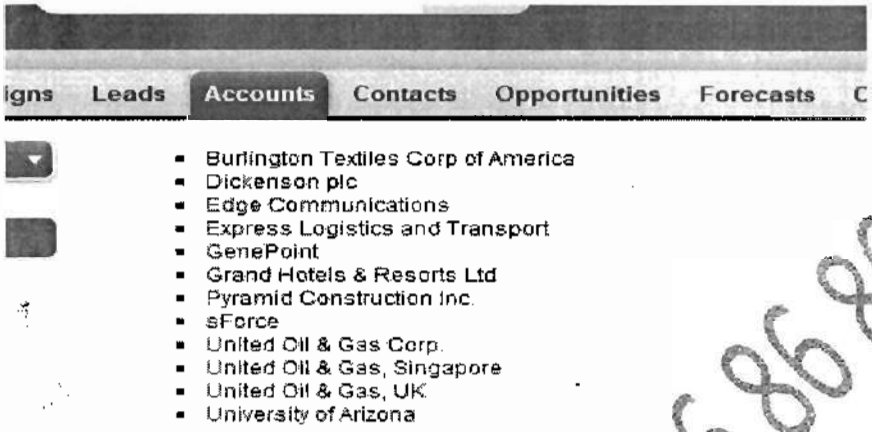
Type : This will specify the format in which the list should be displayed .
Ordered list possible values: "1", "a", "A", "i", or "I".
Unorder List : "disc", "square", and "circle".

Example :

```
<apex:page standardController="Account" recordSetVar="items">
    <apex:dataList value="{!items}" var="a" type="Square">
```

```
        <apex:outputText value="{!a.Name}"/>
      </apex:dataList>
  </apex:page>
```

Output Screen :



➔<apex:repeat> :

In iteration component that allows you to output the contents of a collection according to a structure that you specify. The collection can include up to 1,000 items.

Note that if used within an < apex:pageBlockSection > or < apex:panelGrid > component, all content generated by a child < apex:repeat > component is placed in a single < apex:pageBlockSection > or < apex:panelGrid > cell.

This component can't be used as a direct child of the following components:

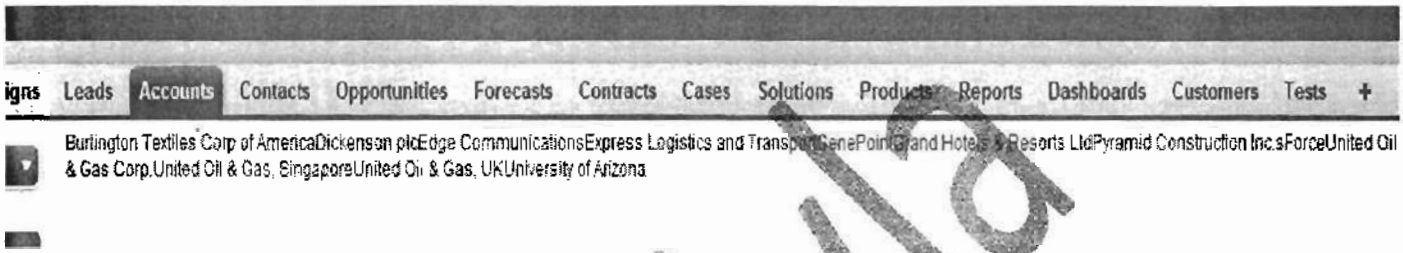
- < apex:panelBar >
- < apex:selectCheckboxes >
- < apex:selectList >
- < apex:selectRadio >
- < apex:tabPanel >

Example :

```
<apex:page standardController="Account" recordSetVar="items">
  <apex:repeat value="{!items}" var="a" >
    <apex:outputText value="{!a.Name}"/>
  </apex:repeat>
```

</apex:page>

Output Screen :



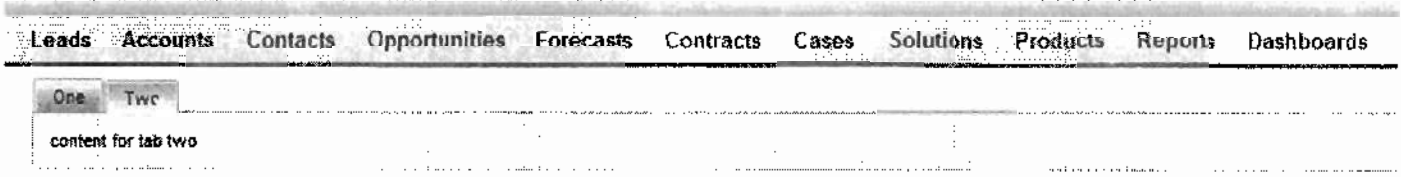
➔<apex:tabPanel>
A page area that displays as a set of tabs. When a user clicks a tab header, the tab's associated content displays, hiding the content of other tabs.

activeTabClass : This will specify in which style class the tab should be displayed when the tab is selected.
disabledTabClass : This will specify in which style class the tab should be displayed when tab is not in selected mode.
Height : The height of the tab bar expressed as a percentage of the available vertical space.
Width : The width of the tab bar.

Example :

```
<apex:page id="thePage">
  <apex:tabPanel switchType="client" selectedTab="name2" id="theTabPanel">
    <apex:tab label="One" name="name1" id="tabOne">content for tab one</apex:tab>
    <apex:tab label="Two" name="name2" id="tabTwo">content for tab two</apex:tab>
  </apex:tabPanel>
</apex:page>
```

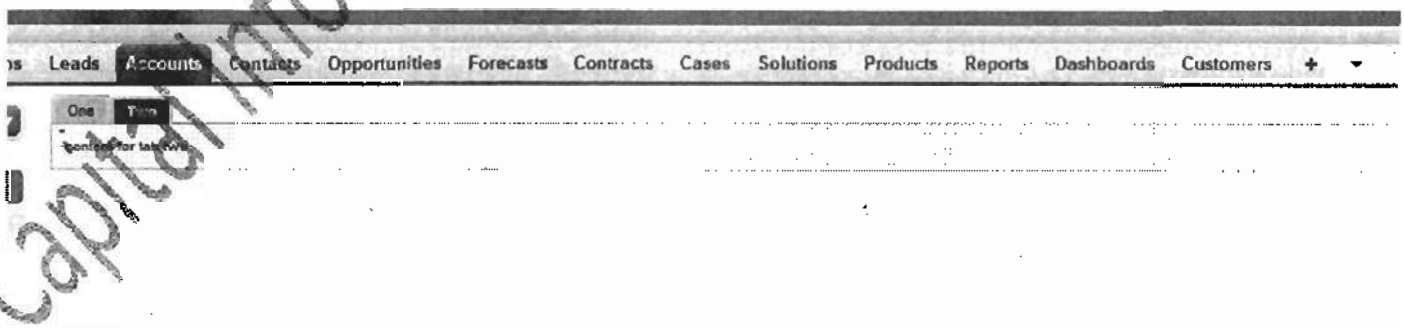
Output Screen :



Example :

```
<apex:page standardController="Account" showHeader="true">
  <!-- Define Tab panel .css styles -->
  <style>
  .activeTab {background-color: #236FBD; color:white; background-image:none}
  .inactiveTab { background-color: lightgrey; color:black; background-image:none}
  </style>
  <!-- Create Tab panel -->
  <apex:tabPanel switchType="client" selectedTab="name2" id="AccountTabPanel"
    tabClass="activeTab" inactiveTabClass="inactiveTab">
    <apex:tab label="One" name="name1" id="tabOne">content for tab one</apex:tab>
    <apex:tab label="Two" name="name2" id="tabTwo">content for tab two</apex:tab>
  </apex:tabPanel>
</apex:page>
```

Output Screen :

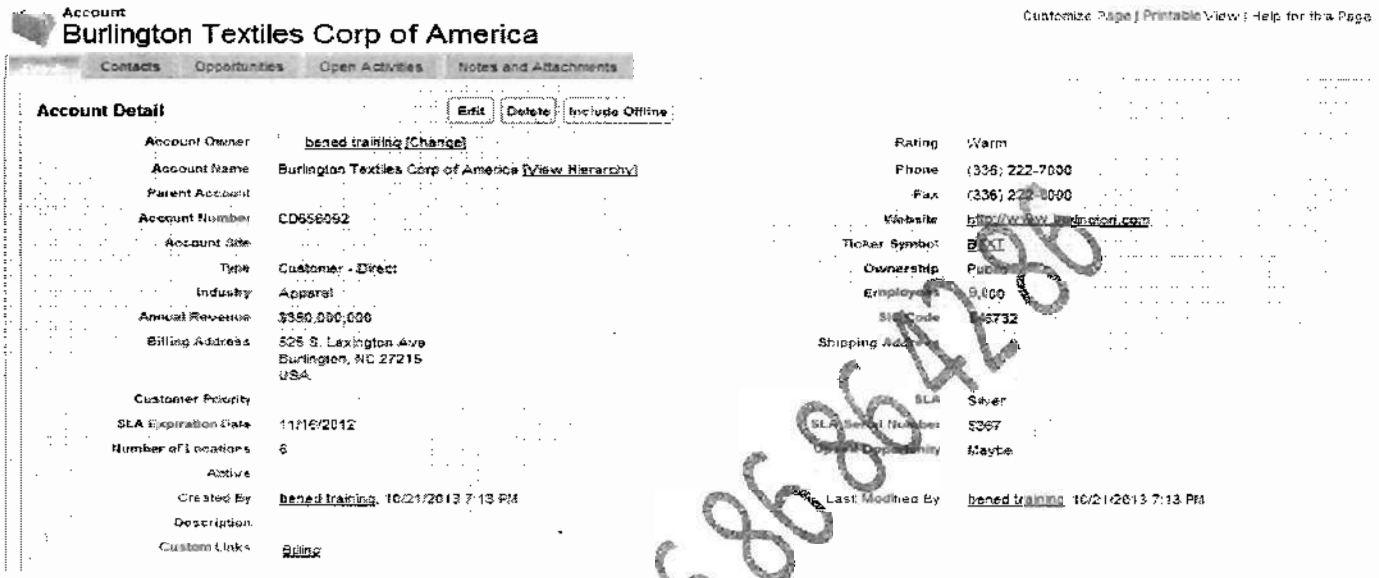


Example :

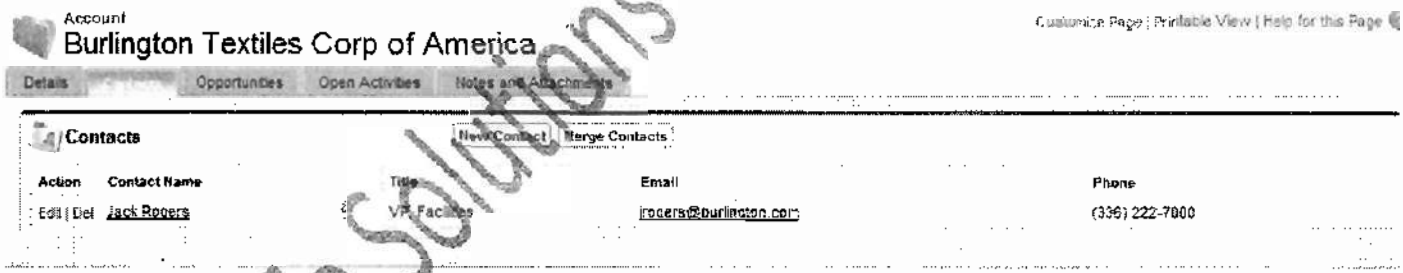
```
<apex:page standardController="Account" showHeader="true" tabStyle="account" >
  <style>
  .activeTab {
        background-color: #236FBD; color:white;
        background-image:none:
```

```
        }
.inactiveTab {
        background-color: lightgrey;
        color:black;
        background-image:none
        }
</style>
<apex:tabPanel switchType="client" selectedTab="tabdetails" id="AccountTabPanel" tabClass="activeTab"
                                                        inactiveTabClass="inactiveTab">
  <apex:tab label="Details" name="AccDetails" id="tabdetails">
    <apex:detail relatedList="false" title="true"/>
  </apex:tab>
  <apex:tab label="Contacts" name="Contacts" id="tabContact">
    <apex:relatedList subject="{!account}" list="contacts" />
  </apex:tab>
  <apex:tab label="Opportunities" name="Opportunities" id="tabOpp">
    <apex:relatedList subject="{!account}" list="opportunities" />
  </apex:tab>
  <apex:tab label="Open Activities" name="OpenActivities" id="tabOpenAct">
    <apex:relatedList subject="{!account}" list="OpenActivities" />
  </apex:tab>
  <apex:tab label="Notes and Attachments" name="NotesAndAttachments" id="tabNoteAtt">
   <apex:relatedList subject="{!account}' list="CombinedAttachments" />
  </apex:tab>
 </apex:tabPanel>
</apex:page>
```

Output Screen :



When we click on the Contacts :



➔<apex: include> :

This component is used to include one page in another page .

Attributes:

ID : This is the id of the component

PageName: This the name of the page which we ant to include in the VF page.

Rendered: This specifies wheather the component should be displayed or not.

Ex:

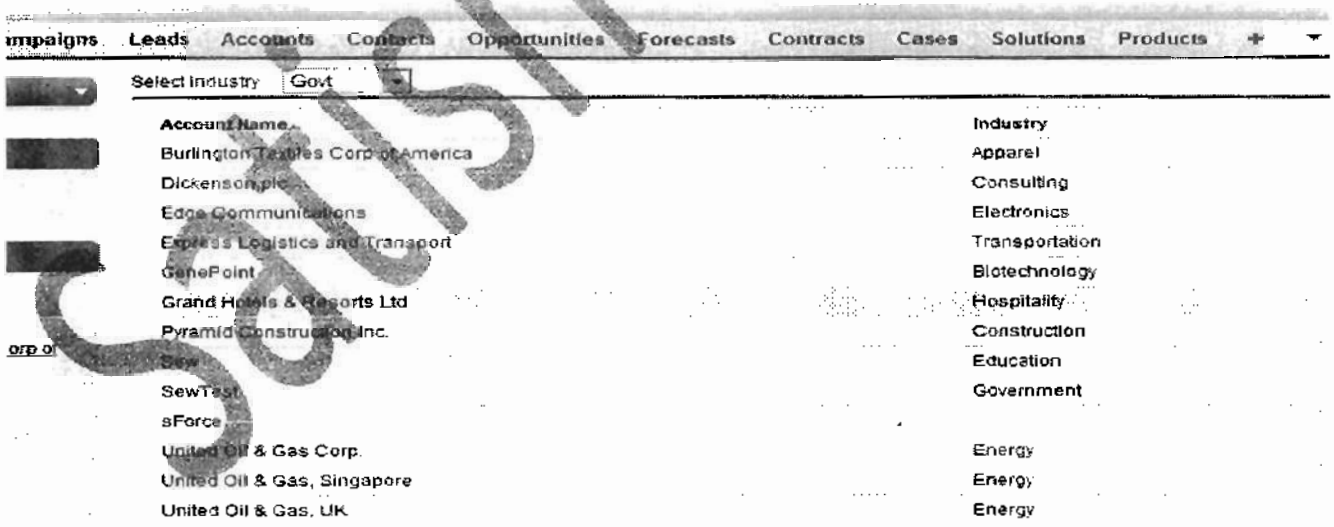
Step 1 :First Create Visualforce page with name :" Include Page"

```
<apex:page standardController="Account"  recordSetVar="items">
  <apex:pageBlock >
     <apex:pageBlockTable value="{!items}" var="a">
        <apex:column value="{!a.name}"/>
```

```
            <apex:column value="{!a.industry}"/>
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

Step 2: Create a visualforce which will include above VF in this : Name : FinalPage

```
<apex:page >
    <apex:form >
        <apex:outputLabel >Select Industry</apex:outputLabel>    
        <apex:selectList size="1">
                <apex:selectOption itemLabel="Govt" itemValue="govt"/>
                <apex:selectOption itemLabel="Banking" itemValue="bank"/>
                <apex:selectOption itemLabel="IT" itemValue="it"/>
                <apex:selectOption itemLabel="Finanace" itemValue="fin"/>
        </apex:selectList>
        <apex:include pageName="includepage"/>
    </apex:form>
</apex:page>
```



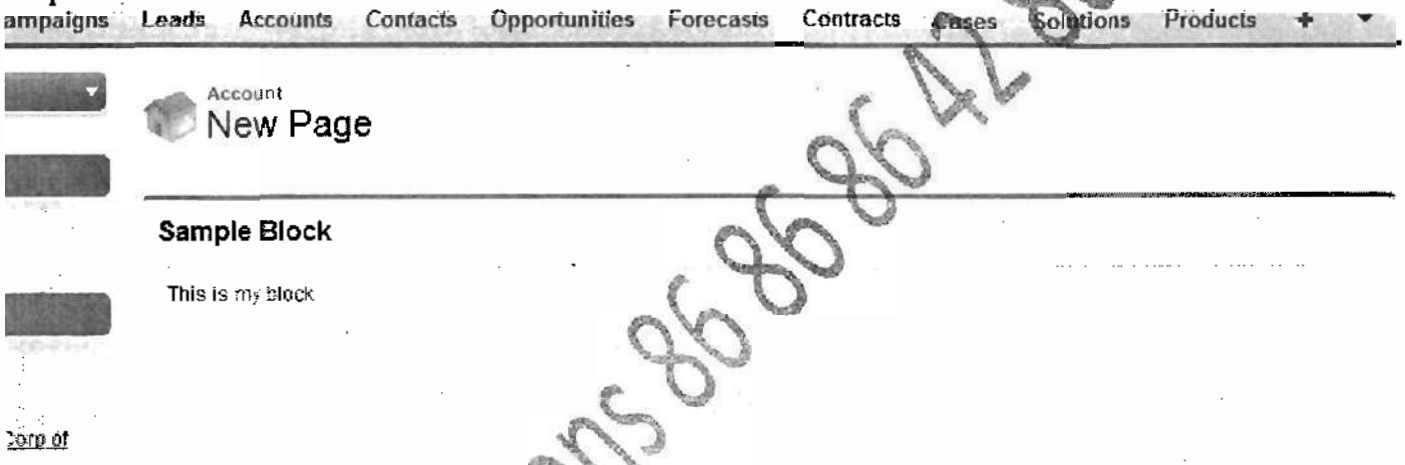
➔Creating Visualforce Template page using <apex:insert>

Step 1:

```
<apex:page >
    <apex:sectionHeader title="Account" subtitle="New Page"/>
```

```
<apex:insert name="header"/><br/>
<apex:pageBlock title="Sample Block">
        This is my block
</apex:pageBlock><br/>
<apex:insert name="body"/>
</apex:page>
```

Output :



Any page which using above template can define its own text in place of the <apex:insert > by referring the name given at .

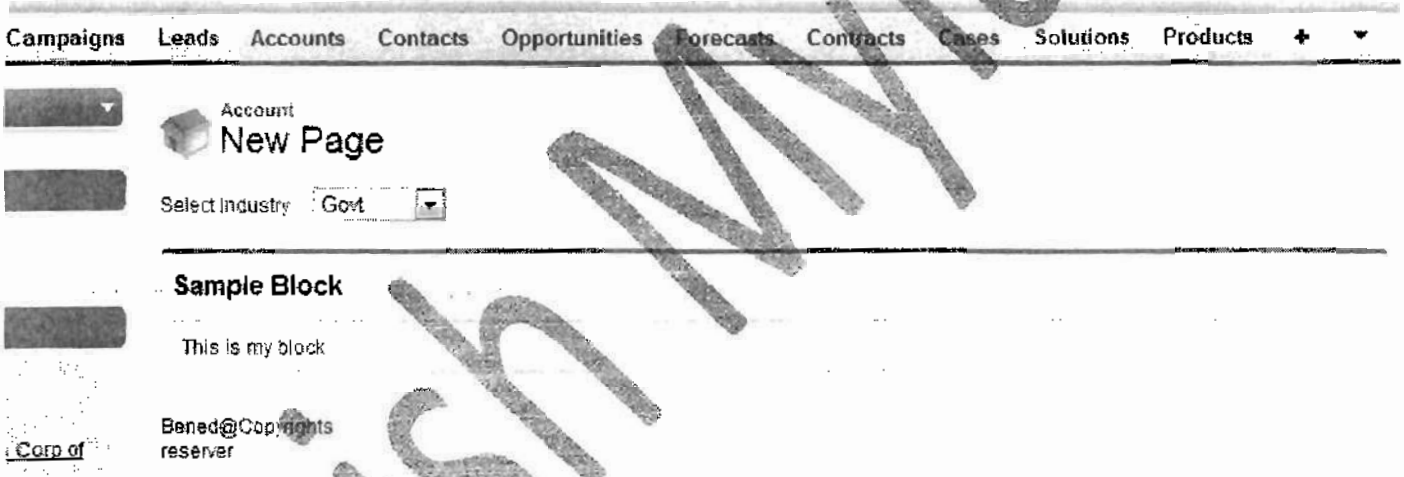
Step 2:

A template component that declares <apex:insert name=xxx> area that must be defined by an < apex:define > component in another Visualforce page.

Use this component with the <apex:composition > and < apex:define > components to share data between multiple pages.

```
<apex:page>
  <apex:composition template="templatepage">
   <apex:define name="header">
    <apex:form >
     <apex:outputLabel >Select Industry</apex:outputLabel>    
     <apex:selectList size="1">
        <apex:selectOption itemLabel="Govt" itemValue="govt"/>
         <apex:selectOption itemLabel="Banking" itemValue="bank"/>
                <apex:selectOption itemLabel="IT" itemValue="it"/>
                <apex:selectOption itemLabel="Finanace" itemValue="fin"/>
       </apex:selectList><br/>
```

```
    </apex:form>
  </apex:define>
  <apex:define name="body">
    <div style="background-color:yellow;width:200px;">
      Bened@copyright reserved  </div>
      </apex:define>
</apex:composition>
</apex:page>
```



➔<apex:iframe> : This is used for embedding entire page into another page .

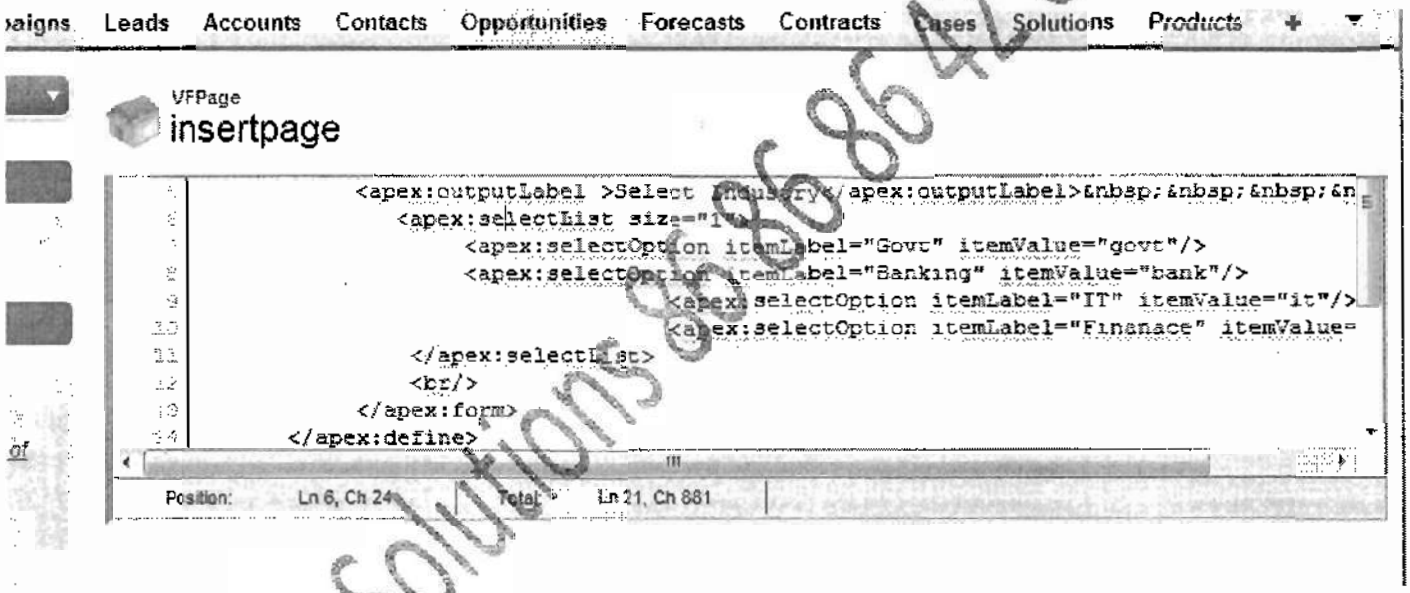
Height : height of the iframe
Border : This is a Boolean value which would specify wheater the border for the Frame should be displayed or not.
Scroll : this is Boolean value which specifies wheater the scroll for the page is enabled or Not
Src : we have the specify the url of the page or frame which need to be displayed in the Iframe

Example :

```
<apex:page >
  <apex:sectionHeader title="VFPage" subtitle="insertpage"/>
```

```
    <apex:form >
        <apex:iframe frameborder="true" height="200px" scrolling="true"
                     src="https://c.ap1.visual.force.com/apex/insertdefinepage" />
    </apex:form>
</apex:page>
```

Output Screen :



➔<apex:details> : The standard detail page for a particular object, as defined by the associated page layout for the object in Setup. This component includes attributes for including or excluding the associated related lists.

relatedList : It is a Boolean value that specifies whether the related list are included in the rendered component. If true related lists are displayed .

relatedListHover :It is Boolean value that specified wheather the related list hover links are included in the rendered component .If true the related list hover links are displayed .

subject :This is the ID of the record that should provide data for this component;

Example : Example to display the detail page of the account owner

```
<apex:page standardController="Account">
  <apex:detail subject="{!account.ownerId}" relatedList="false" title="false"/>
```

</apex:page>

Output Screen :

➔<apex:relatedList> :

A list of Salesforce records that are related to a parent record with a lookup or master-detail relationship.

List : This will specify the related list to display .This does not need to be on the object's page layout.
To specify this value use the name of the child relationship to the related object.
Ex: List="contacts"

pageSize : The number of records to display by default in the relatedlist .If not specified , defaults to 5.

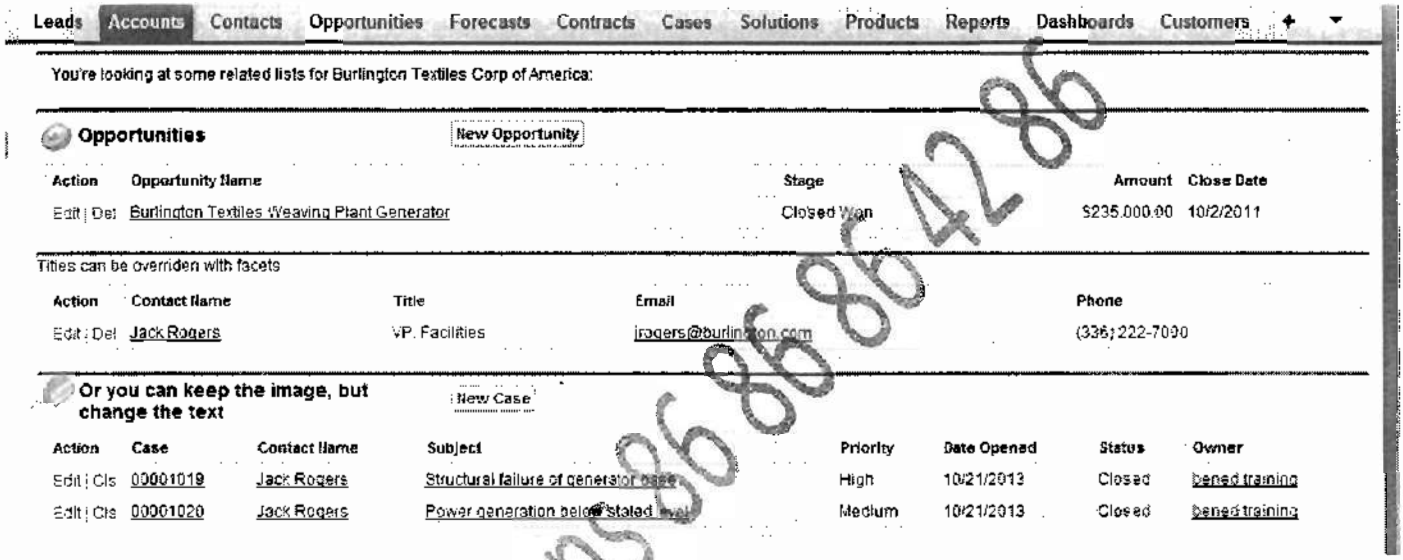
Subject : The parent record from which the data and relatedlist definition are derived .If not specified ,and if Using a standard controller ,this value is automatically set to the value of the ID .

Example :

```
<apex:page standardController="Account">
  <apex:pageBlock >
      You're looking at some related lists for {!account.name}:
  </apex:pageBlock>
  <apex:relatedList list="Opportunities" />
  <apex:relatedList list="Contacts">
    <apex:facet name="header">Titles can be overriden with facets</apex:facet>
  </apex:relatedList>
  <apex:relatedList list="Cases" title="Or you can keep the image, but change the text" />
```

</apex:page>

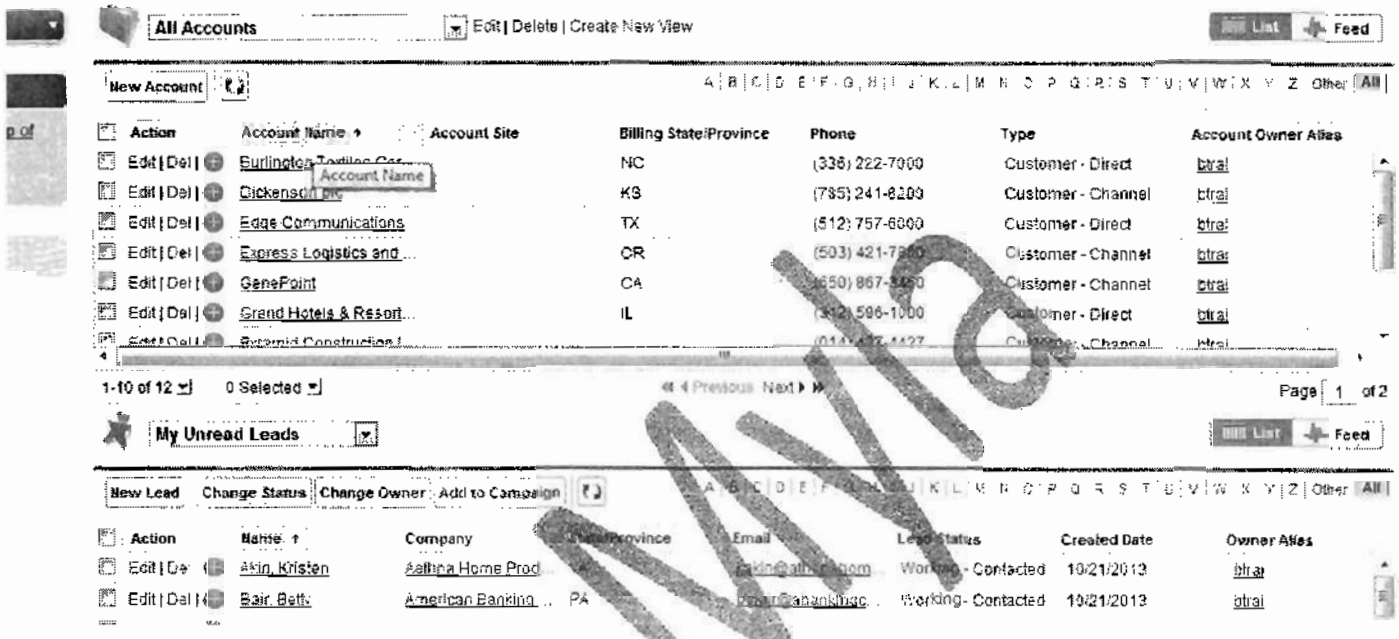
Output Screen :



< apex:enhancedList > : This is rerendered through another component's rerender attribute, the < apex:enhancedList > must be inside of an < apex:outputPanel > component that has its layout attribute set to "block". The < apex:enhancedList > component is not allowed on pages that have the attributeshowHeader set to false.

Example :

```
<apex:page >
    <apex:enhancedList type="Account" height="300" rowsPerPage="10" id="AccountList" />
    <apex:enhancedList type="Lead" height="300" rowsPerPage="25"   id="LeadList" customizable="False" />
</apex:page>
```



Senario 1: Print the visualforce page in PDF format .

```
<apex:page standardController="Account" renderAs="pdf" applyBodyTag="false">
<head>
 <style>
  body { font-family: 'Arial Unicode MS'; }
  .companyName { font: bold 30px; color: red; }
  </style>
 </head>
 <body>
 <center>
  <h1>New Account Name!</h1>
  <apex:panelGrid columns="1" width="100%">
    <apex:outputText value="{!account.Name}" styleClass="companyName"/>
    <apex:outputText value="{!NOW()}"></apex:outputText>
    </apex:panelGrid>
    </center>
 </body>
</apex:page>
```

Output Screen :

New Account Name!

Bened Software

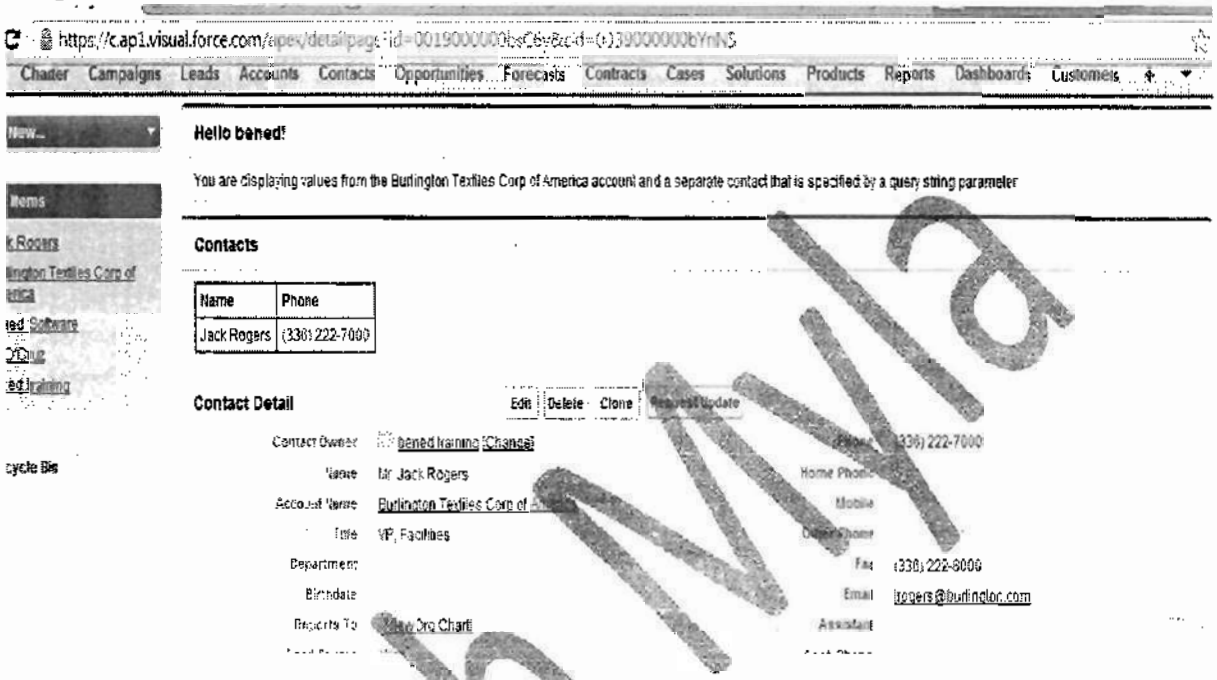
Sat Dec 14 06:56:34 GMT 2013

Senario 2: How to use the parameters used in the currentpage url

https://c.ap1.visual.force.com/apex/detailpage?id=0019000000bsC6y&cid=0039000000bYnN5

```
<apex:page standardController="Account">
  <apex:pageBlock title="Hello {!$User.FirstName}!">
    You are displaying values from the {!account.name} account and a separate contact
    that is specified by a query string parameter.
   </apex:pageBlock>
  <apex:pageBlock title="Contacts">
  <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4" border="1">
    <apex:column >
    <apex:facet name="header">Name</apex:facet>
      {!contact.Name}
    </apex:column>
    <apex:column >
      <apex:facet name="header">Phone</apex:facet>
        {!contact.Phone}
      </apex:column>
   </apex:dataTable>
  </apex:pageBlock>
  <apex:detail subject="{!$CurrentPage.parameters.cid}" relatedList="false" title="false"/>
</apex:page>
```

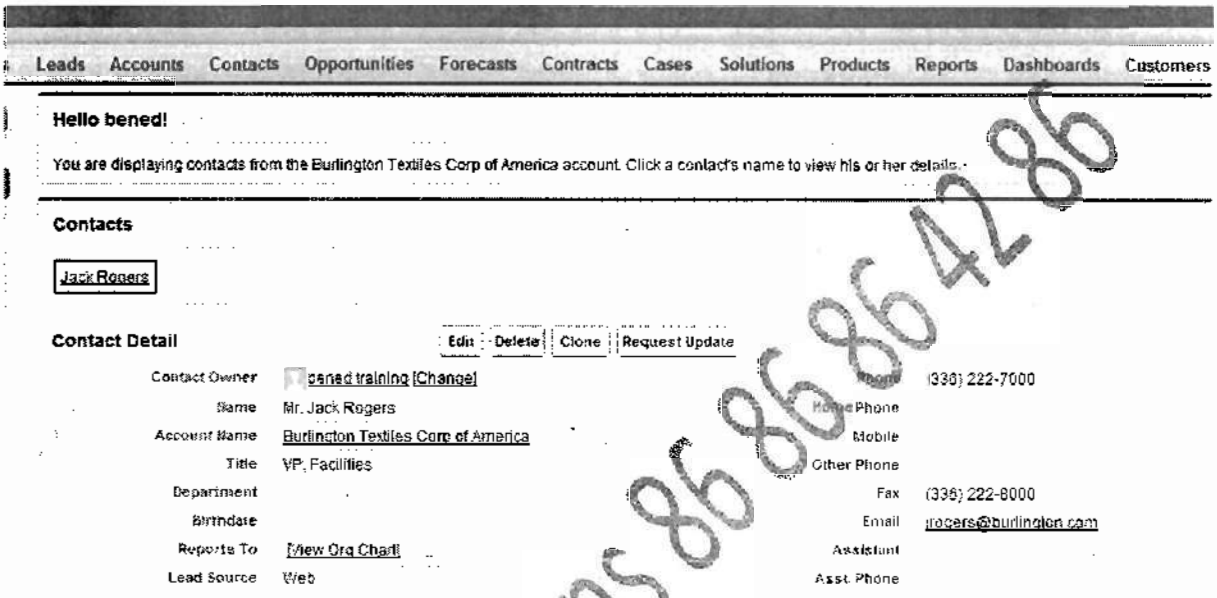
Output Screen :



Senario 3 : Display the detail of the record based on record id provided

```
<apex:page standardController="Account">
  <apex:pageBlock title="Hello {!$User.FirstName}!">
    You are displaying values from the {!account.name} account and a separate contact
    that is specified by a query string parameter.
  </apex:pageBlock>
  <apex:pageBlock title="Contacts">
   <apex:dataTable value="{!account.Contacts}" var="contact" cellPadding="4" border="1">
    <apex:column >
    <apex:facet name="header">Name</apex:facet>
      {!contact.Name}
    </apex:column>
    <apex:column >
     <apex:facet name="header">Phone</apex:facet>
        {!contact.Phone}
     </apex:column>
  </apex:dataTable>
 </apex:pageBlock>
 <apex:detail subject="{!$CurrentPage.parameters.cid}" relatedList="false" title="false"/>
```

```
</apex:page>
```



Senario 4: Genreating visaulforce in Excel sheet:

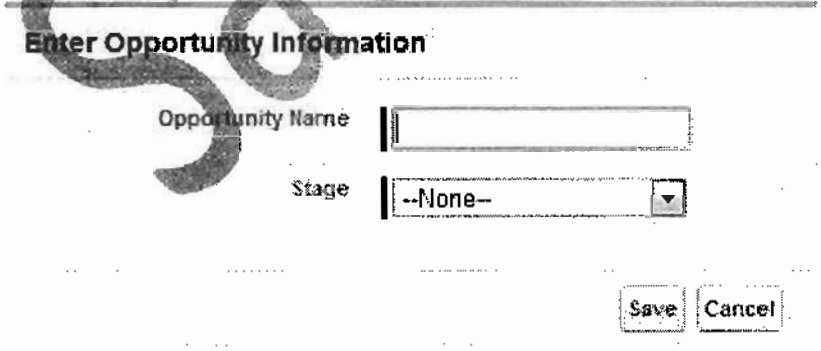
```
<apex:page standardController="Account" contentType="application/vnd.ms-excel">
<apex:pageBlock title="Contacts">
<apex:pageBlockTable value="{!account.Contacts}" var="contact">
<apex:column value="{!contact.Name}"/>
<apex:column value="{!contact.MailingCity}"/>
<apex:column value="{!contact.Phone}"/>
</apex:pageBlockTable>
</apex:pageBlock>
</apex:page>
```

Senario 5:How to Re-render a part of VisualForce Page using <apex:actionRegion>

How to re-render pageblock / section of a page ?

```
<apex standardController="Opportunity">
 <apex:form >
 <apex:pageBlock title="Enter Opportunity Information">
 <apex:pageBlockButtons location="bottom">
```

```
  <apex:commandButton value="Save" action="{!Save}"/>
  <apex:commandButton value="Cancel" action="{!Cancel}" immediate="true"/>
</apex:pageBlockButtons>
<apex:PageblockSection columns="1" >
   <apex:inputField value="{!Opportunity.Name}"/>
<apex:PageBlockSectionItem >
    <apex:outputLabel value="Stage"/>
    <! ======= Field to be re-rendered ======= />
    <apex:actionRegion >
     <apex:inputField value="{!Opportunity.StageName}">
      <apex:actionSupport event="onchange" reRender="ajaxrequest" />
     </apex:inputField>
    </apex:actionRegion>
</apex:PageBlockSectionItem>
</apex:PageblockSection>
  <! ======= What to Re-render and on What Condition ======= />
<apex:outputPanel id="ajaxrequest">          <apex:pageBlockSection
rendered="{!Opportunity.StageName=='Prospecting'}" >
   <apex:inputField value="{!Opportunity.CloseDate}"/>
</apex:pageBlockSection>
</apex:outputPanel>
</apex:pageBlock>
</apex:form>
</apex:page>
```



Senario 6:Checking for Object Accessibility using {!$Object} in visualforce page

If a user has insufficient privileges to view an object, any Visualforce page that uses a controller to render that object will be inaccessible. To avoid this error, you should ensure that your Visualforce components will only

render if a user has access tothe object associated with the controller.You can check for the accessibility of an object like this:

{!$ObjectType.objectname.accessible} This expression returns a true or false value.

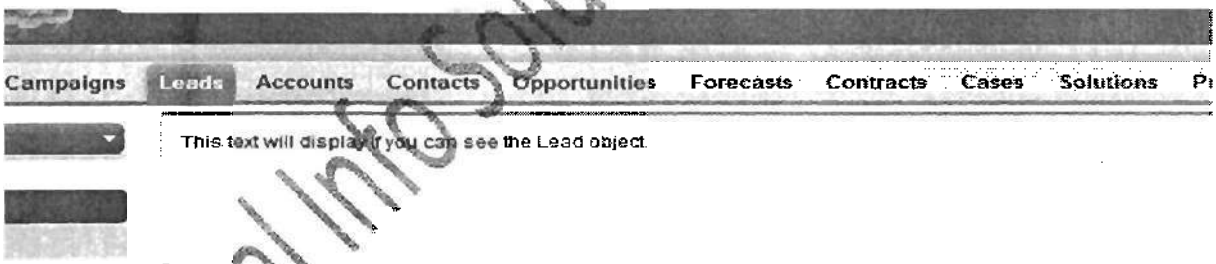
For example, to check if you have access to the standard Lead object, use the following code:
{!$ObjectType.Lead.accessible}

For custom objects, the code is similar:
{!$ObjectType.MyCustomObject__c.accessible} where MyCustomObject__c is the name of your custom object

Example :

```
<apex:page standardController="Lead">
  <apex:pageBlock rendered="{!$ObjectType.Lead.accessible}">
    <p>This text will display if you can see the Lead object.</p>
  </apex:pageBlock>
</apex:page>
```

Output Screen :

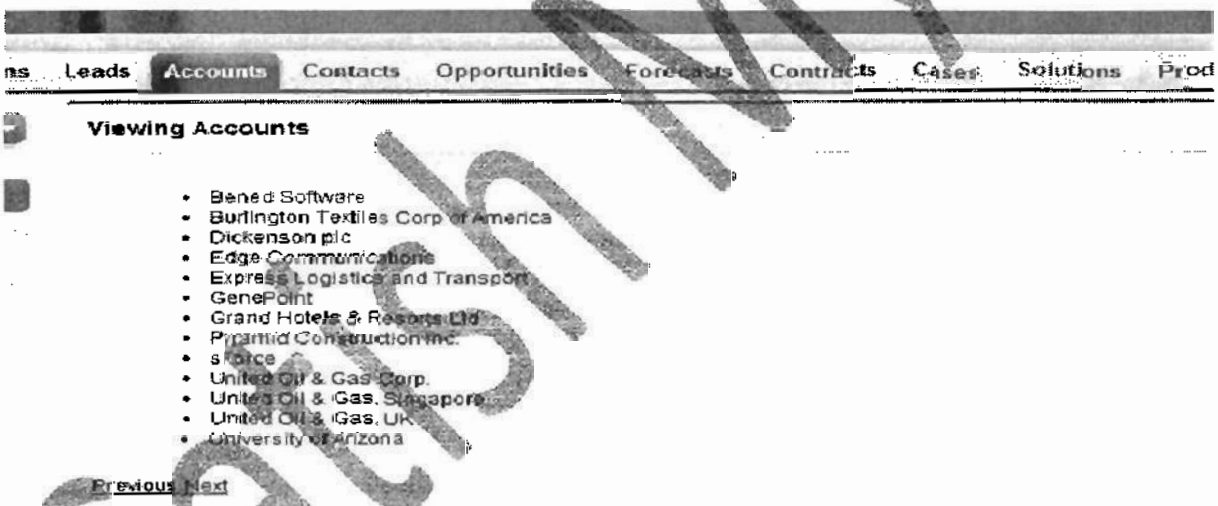


Senario 7: Pagination with a List Controller

You can add pagination to a page using a list controller by utilizing the next and previous actions. For **Example:**

```
<apex:page standardController="Account" recordSetvar="accounts">
  <apex:pageBlock title="Viewing Accounts">
    <apex:form id="theForm">
     <apex:pageBlockSection >
```

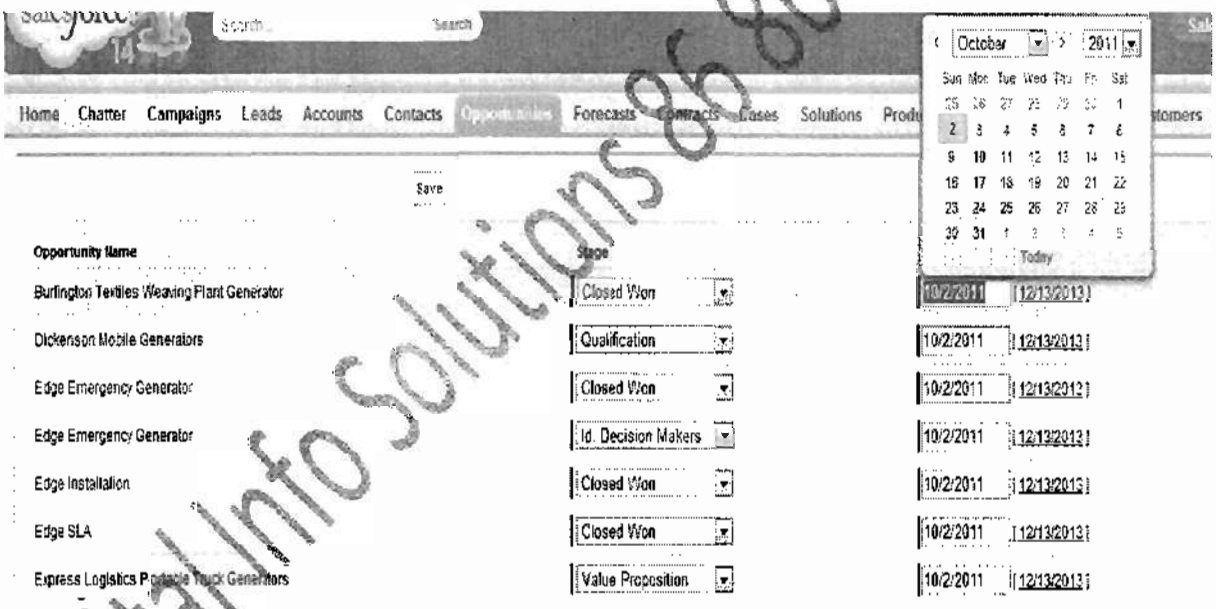
```
      <apex:dataList var="a" value="{!accounts}" type="1">
        {!a.name}
      </apex:dataList>
    </apex:pageBlockSection>
    <apex:panelGrid columns="2">
      <apex:commandLink action="{!previous}">Previous</apex:commandlink>
      <apex:commandLink action="{!next}">Next</apex:commandlink>
    </apex:panelGrid>
    </apex:form>
  </apex:pageBlock>
</apex:page>
```



Senario 8:Editing Records with List Controller:

```
<apex:page standardController="Opportunity" recordSetVar="opportunities" tabStyle="Opportunity"
sidebar="false">
<apex:form >
 <apex:pageBlock >
  <apex:pageMessages />
    <apex:pageBlockButtons >
      <apex:commandButton value="Save" action="{!save}"/>
    </apex:pageBlockButtons>
  <apex:pageBlockTable value="{!opportunities}" var="opp">
```

```
    <apex:column value="{!opp.name}"/>
    <apex:column headerValue="Stage">
        <apex:inputField value="{!opp.stageName}"/>
    </apex:column>
    <apex:column headerValue="Close Date">
      <apex:inputField value="{!opp.closeDate}"/>
    </apex:column>
  </apex:pageBlockTable>
  </apex:pageBlock>
</apex:form>
</apex:page>
```



Senario 9: Create a visuaforce page with one picklist with list of object in your account as options in the picklist and one Multiselect which is display list of fields available in the selected object.
When click on the show Table it should the list of the records from the object which we have choosen with selected fields

```
public class DynamicTableController
{
  //List displayed on UI
  public List<selectoption> supportedObject {get; set;}
```

```
//Selected Object
public String SelectedObject {get; set;}

//Global describe
Map<String, Schema.SObjectType> gd = Schema.getGlobalDescribe();
Set<String> objectKeys = gd.keySet();

//Field Select List
public List<SelectOption> fieldLableAPI {get; set;}

//Selected fields to be displayed in table
public List<String> SelectedFields {get; set;}

//List to maintain dynamic query result
public List<sObject> ObjectList {get; set;}

//Constructor
public DynamicTableController()
{
  //Initialize
  supportedObject = new List<selectoption>() ;
  SelectedObject = '';
  fieldLableAPI = new List<SelectOption>() ;
  SelectedFields = new List<String>() ;
  ObjectList = new List<sObject>() ;

  //Get only reference to objects
  for(Schema.SObjectType item : ProcessInstance.TargetObjectId.getDescribe().getReferenceTo())
  {
    //Excluding custom setting objects
    if(!item.getDescribe().CustomSetting)
    {
      //Adding to list
      supportedObject.add(new SelectOption(item.getDescribe().getLocalName().toLowerCase() ,
item.getDescribe().getLabel() ));
    }
  }
```

```
}

//Get fields of selected object
public void ObjectFields()
{
    if(SelectedObject != '--None--')
    {
        //Creating sObject for dynamic selected object
        Schema.SObjectType systemObjectType = gd.get(SelectedObject);
        //Fetching field results
        Schema.DescribeSObjectResult r = systemObjectType.getDescribe();

        Map<String, Schema.SObjectField> M = r.fields.getMap();
        //Creating picklist of fields
        for(Schema.SObjectField fieldAPI : M.values())
        {
            fieldLableAPI.add(new SelectOption(fieldAPI.getDescribe().getName() ,
fieldAPI.getDescribe().getLabel())) ;
        }
    }
}

public void ShowTable()
{
    //Creating dynamic query with selected field
    String myQuery = 'Select Id ' ;

    for(String field : SelectedFields)
    {
        if(field.toLowerCase() != 'id' && field.toLowerCase() != '--none--')
        myQuery += ','+ field + ' ' ;
    }

    //Limit is 100 for now you can change it according to need
    myQuery += ' from ' + SelectedObject + ' LIMIT 100' ;

    //Executing the query and fetching results
```

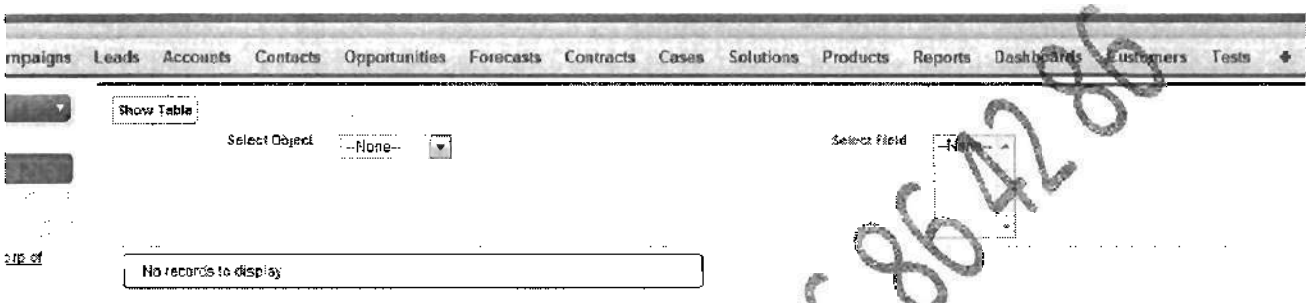
```
    ObjectList = Database.query(myQuery) ;
  }
}
```

Visualforce Page:

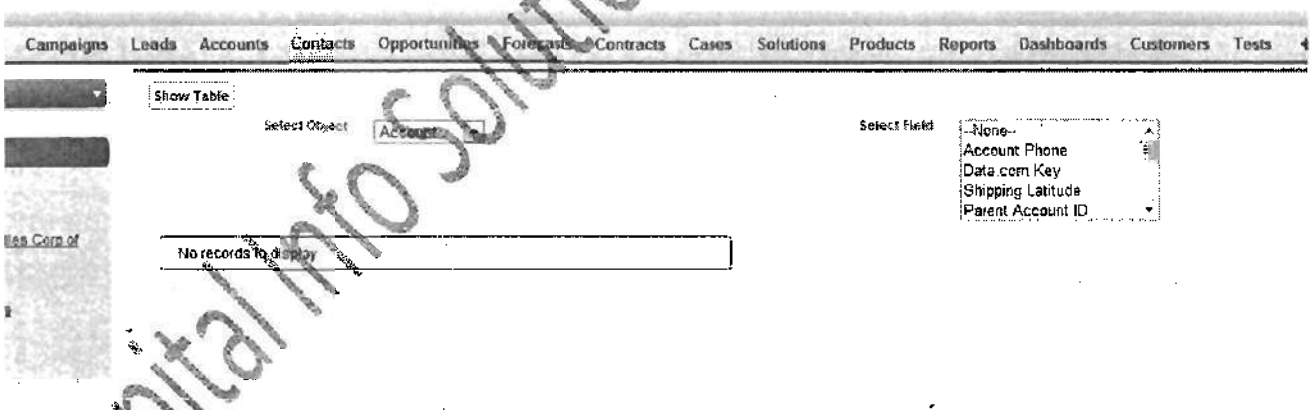
```
<apex:page controller="DynamicTableController">
<apex:pageBlock >
  <apex:form >
    <apex:actionFunction name="ObjectFileds" action="{!ObjectFields}"/>
    <apex:commandButton value="Show Table" action="{!ShowTable}"/>
     <apex:pageBlockSection >
        <apex:pageBlockSectionItem >
          <apex:outputLabel value="Select Object"/>
    <apex:selectList multiselect="false" size="1" value="{!SelectedObject}"
                                                  onchange="ObjectFileds();">
        <apex:selectOption itemLabel="--None--" itemValue="--None--"/>
        <apex:selectoptions value="{!supportedObject}" />
     </apex:sclectlist>
      </apex:pageBlockSectionItem>
      <apex:pageBlockSectionItem >
        <apex:outputLabel value="Select Field"/>
        <apex:selectList multiselect="true" size="5" value="{!SelectedFields}">
          <apex:selectOption itemLabel="--None--" itemValue="--None--"/>
          <apex:selectoptions value="{!fieldLableAPI}" />
        </apex:selectlist>
      </apex:pageBlockSectionItem>
      <apex:pageBlockTable rendered="{!IF(ObjectList.size > 0 , true , false)}"
                                                  value="{!ObjectList}" var="rec">
      <apex:column value="{!rec.Id}" rendered="{!IF(SelectedFields.size == 0 , true, false)}"/>
      <apex:repeat value="{!SelectedFields}" var="FieldLable">
<apex:column value="{!rec[FieldLable]}" rendered="{!IF(FieldLable != '--None--' , true, false)}"/>
</apex:repeat>
      </apex:pageBlockTable>
      <apex:outputPanel rendered="{!IF(ObjectList.size < 1 , true , false)}">
        <apex:pageMessage severity="ERROR" summary="No records to display"/>
      </apex:outputPanel>
    </apex:pageBlockSection>
  </apex:form>
```

</apex:pageBlock>
</apex:page>

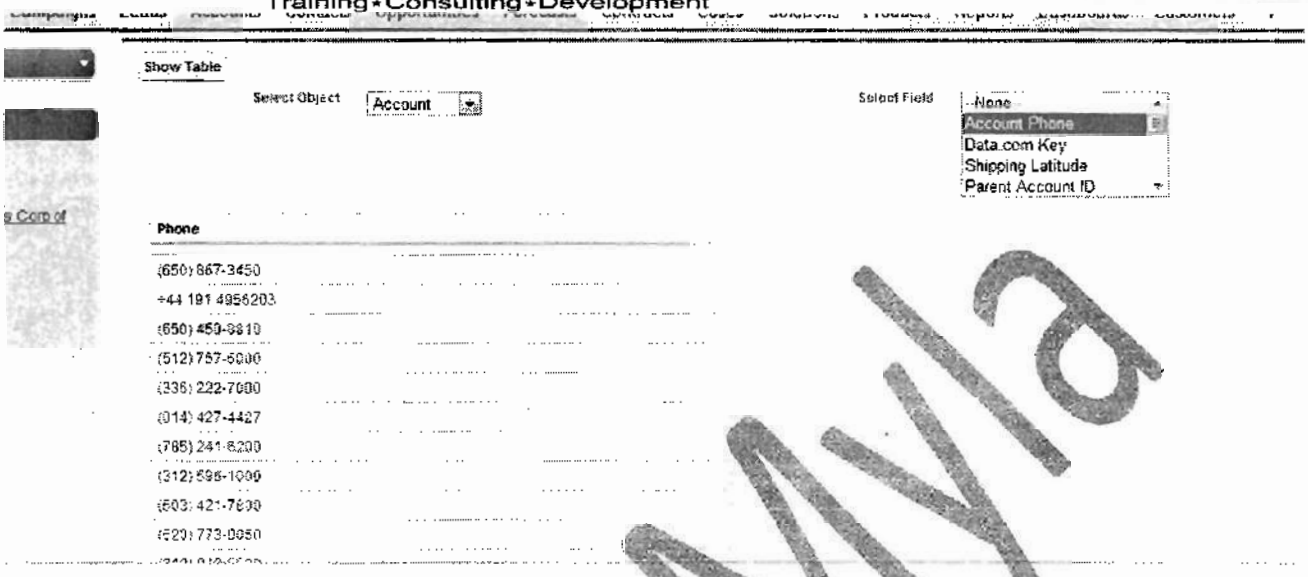
OutputScreen :



When we select Account object then Account object fields will be populated in the select Field multiselct picklist.



Select Account Phone filed in the Multiselect picklist field and click on ShowTable button then we Will get the list Account records with Account Phone no field.



Senario 10:Create a visualforce page that will read the code of specified visualforce page and display as output .

Apex Class: UtilClass:

```
public class UtilClass
{
  //Method will take visualforce page name as parameter and returns the code written in it
  public static String VFPageCode(String PageName)
  {
    ApexPage testPage = new ApexPage();
    String PageText = ''

    if(PageName != '')
    {
      //Fetching visualforce page code
      testPage = [Select Id, Markup from ApexPage where name =: PageName];
      PageText = testPage.markup ;
    }
    return PageText ;
  }
}
```

Ape x class:VisualforceCodeController:

```
public class VisualforceCodeController
{
   //To display section which hold code as text of VF page
   public boolean showCodeSection {get; set;}

   //To hold VF Code as text
   public String PageText {get; set;}

   //Hold current page name
   public String CurrentPageName {get; set;}

   public VisualforceCodeController()
   {
      CurrentPageName = '' ;
      showCodeSection = false ;
   }

   public PageReference DisplayCode()
   {
      if(CurrentPageName != '')
      {
         //Fetching VF code
         PageText = UtilClass.VFPageCode(CurrentPageName) ;
         showCodeSection = true ;
      }
      return null ;
   }
}
```

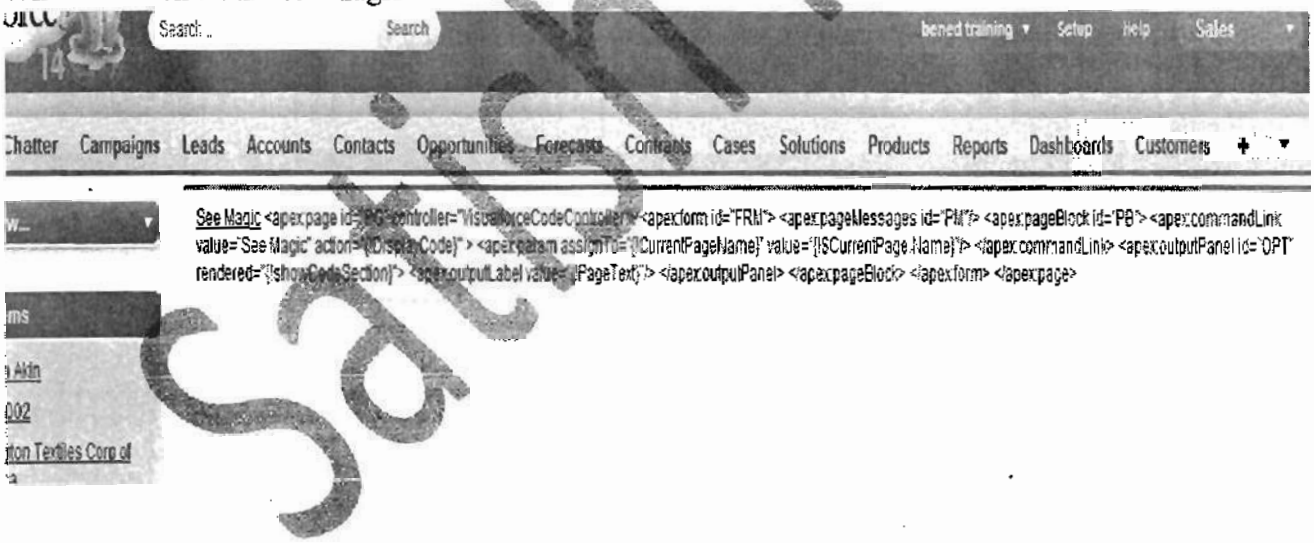
Visualforce Code:

```
<apex:page id="PG" controller="VisualforceCodeController">
<apex:form id="FRM">
   <apex:pageMessages id="PM"/>
   <apex:pageBlock id="PB">
         <apex:commandLink value="See Magic" action="{!DisplayCode}" >
      <apex:param assignTo="{!CurrentPageName}" value="{!$CurrentPage.Name}"/>
```

```
    </apex:commandLink>
    <apex:outputPanel id="OPT" rendered="{!showCodeSection}">
      <apex:outputLabel value="{!PageText}"/>
    </apex:outputPanel>
  </apex:pageBlock>
</apex:form>
</apex:page>
```

Output Screen :

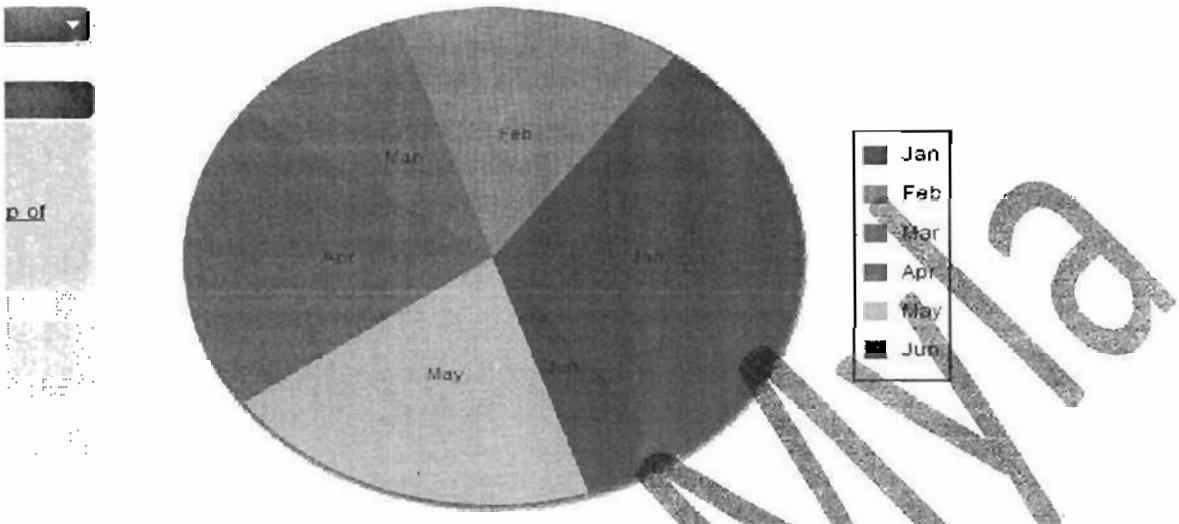
When we click on the see magic



Senario11 :Create a visualforce page to display the pie chart based on the data provide.

```
public class ChartingExample
{
  public List<PieWedgeData> getPieData()
  {
   List<PieWedgeData> data = new List<PieWedgeData>();
```

```
    data.add(new PieWedgeData('Jan', 30));
    data.add(new PieWedgeData('Feb', 15));
    data.add(new PieWedgeData('Mar', 10));
    data.add(new PieWedgeData('Apr', 20));
    data.add(new PieWedgeData('May', 20));
    data.add(new PieWedgeData('Jun', 5));
    return data;
  }
// Wrapper class
  public class PieWedgeData
   {
     public String name { get; set; }
     public Integer data { get; set; }
     public PieWedgeData(String name, Integer data)
      {
        this.name = name;
        this.data = data;
      }
   }
}
<apex:page controller="ChartingExample" title="PieChart">
<apex:chart height="350" width="450" data="{!pieData}">
 <apex:pieSeries dataField="data" labelField="name"/>
  <apex:legend position="right"/>
</apex:chart>
</apex:page>
```

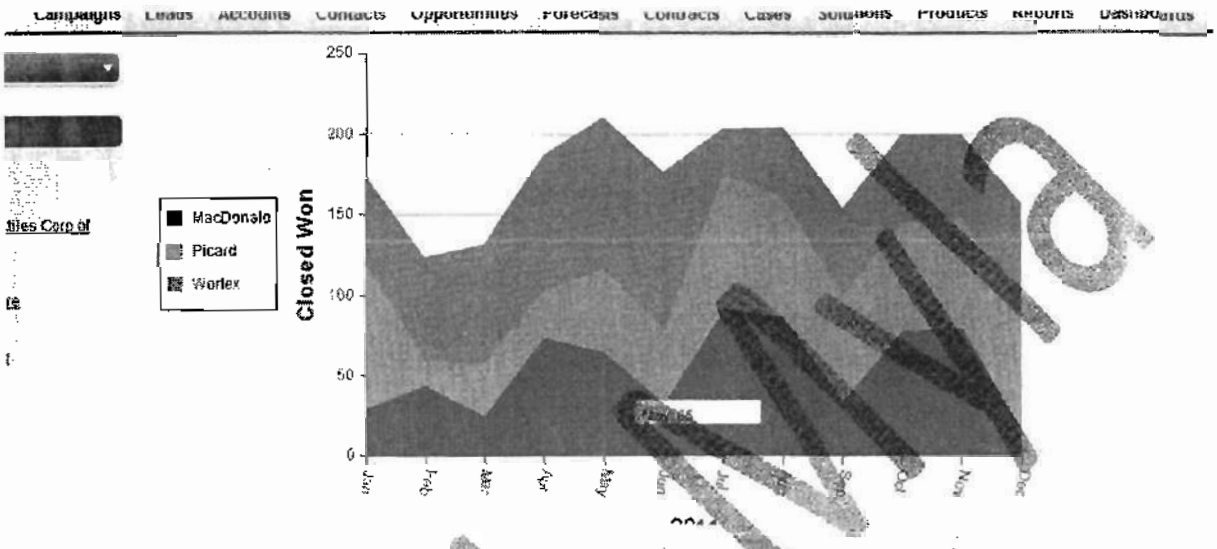


Senario 12: Create a chart

```
public class ChartController
{
  public List<Data> getData()
   {
     return ChartController.getChartData();
   }
// Make the chart data available via JavaScript remoting
@RemoteAction
public static List<Data> getRemoteData()
{
   return ChartController.getChartData();
}
public static List<Data> getChartData()
{
   List<Data> data = new List<Data>();
   data.add(new Data('Jan', 30, 90, 55));
   data.add(new Data('Feb', 44, 15, 65));
   data.add(new Data('Mar', 25, 32, 75));
```

```
    data.add(new Data('Apr', 74, 28, 85));
    data.add(new Data('May', 65, 51, 95));
    data.add(new Data('Jun', 33, 45, 99));
    data.add(new Data('Jul', 92, 82, 30));
    data.add(new Data('Aug', 87, 73, 45));
    data.add(new Data('Sep', 34, 65, 55));
    data.add(new Data('Oct', 78, 66, 56));
    data.add(new Data('Nov', 80, 67, 53));
    data.add(new Data('Dec', 17, 70, 70));
    return data;
}
// Wrapper class
public class Data
{
    public String name { get; set; }
    public Integer data1 { get; set; }
    public Integer data2 { get; set; }
    public Integer data3 { get; set; }
    public Data(String name, Integer data1, Integer data2, Integer data3)
    {
    this.name = name;
    this.data1 = data1;
    this.data2 = data2;
    this.data3 = data3;
    }
 }
}
<apex:page controller="ChartController" >
  <apex:chart height="400" width="700" animate="true" data="{!data}">
   <apex:legend position="left"/>
    <apex:axis type="Numeric" position="left" fields="data1,data2,data3" title="Closed Won" grid="true">
     <apex:chartLabel />
    </apex:axis>
    <apex:axis type="Category" position="bottom" fields="name" title="2011">
      <apex:chartLabel rotate="100"/>
   </apex:axis>
     <apex:areaSeries axis="left" xField="name" tips="true" yField="data1,data2,data3"
title="MacDonald,Picard,Worlex" />
```

```
</apex:chart>
</apex:page>
```



Senario 13:Uploading Multiple Attachments into Salesforce - Simple Code

Okay, so here is the simple code to upload multiple attachments into Salesforce. Please note that I've used standard controller ("Account" in the code) so you need to pass the record Id in parameter. You can also change this according to your need as code is dynamic.

So first option allow you to select how many files you want to upload. Say, you have selected "5", then it will give you 5 options to select files and once you click on "Upload" it will upload the files as attachments to the associated record.

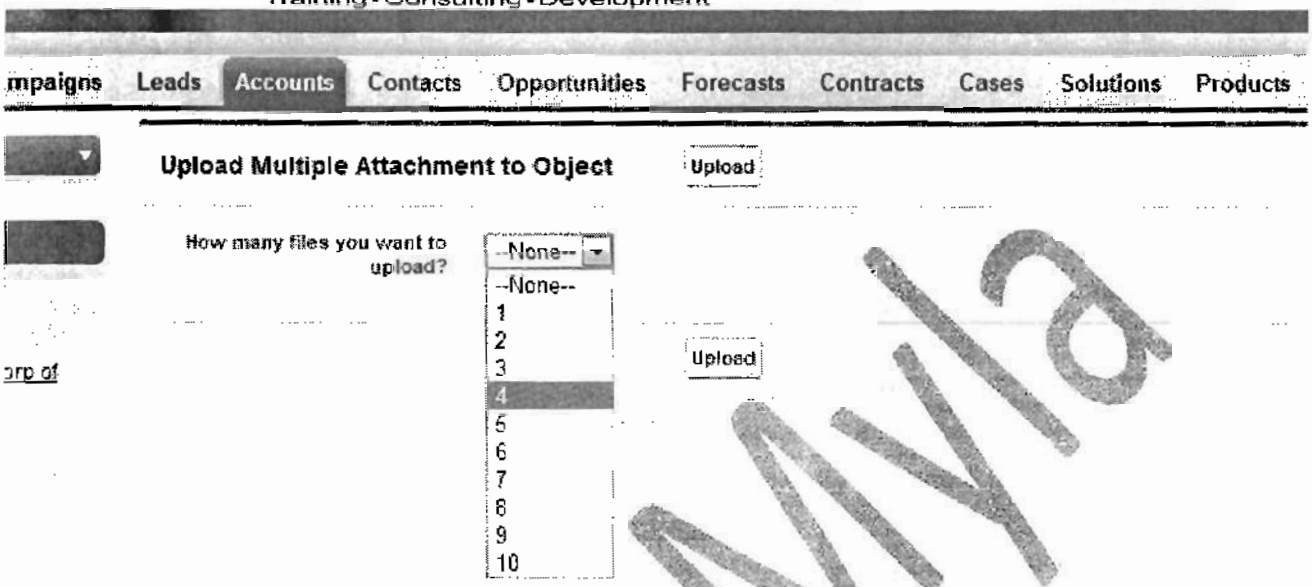
```
<apex:page standardController="Account" extensions="MultipleUploadController">
  <apex:form>
    <apex:pageBlock title="Upload Multiple Attachment to Object">
      <apex:pageBlockButtons>
        <apex:commandButton value="Upload"  action="{!SaveAttachments}"/>
      </apex:pageBlockButtons>
      <apex:pageMessages id="MSG"/>
      <apex:actionFunction name="ChangeCount" action="{!ChangeCount}"/>
      <apex:pageblocksection>
        <apex:pageBlockSectionItem>
          <apex:outputLabel value="How many files you want to upload?"/>
    <apex:selectList onchange="ChangeCount() ;" multiselect="false" size="1" value="{!FileCount}">
```

```
                <apex:selectOption itemLabel="--None--" itemValue=""/>
                <apex:selectOptions value="{!filesCountList}"/>
              </apex:selectList>
            </apex:pageBlockSectionItem>
          </apex:pageblocksection>
 <apex:pageBlockSection title="Select Files" rendered="{!IF(FileCount != null &&
                                                          FileCount != '', true , false)}">
            <apex:repeat value="{!allFileList}" var="AFL">
              <apex:inputfile value="{!AFL.body}" filename="{!AFL.Name}" />
            </apex:repeat>
          </apex:pageBlockSection>

      </apex:pageBlock>
    </apex:form>
</apex:page>

public class MultipleUploadController
{
   //Picklist of tnteger values to hold file count
   public List<SelectOption> filesCountList {get; set;}
   //Selected count
   public String FileCount {get; set;}

   public List<Attachment> allFileList {get; set;}

   public MultipleUploadController(ApexPages.StandardController controller)
   {
      //Initialize
      filesCountList = new List<SelectOption>() ;
      FileCount = '' ;
      allFileList = new List<Attachment>() ;

      //Adding values count list - you can change this according to your need
      for(Integer i = 1 ; i < 11 ; i++)
         filesCountList.add(new SelectOption('+i , '+i)) ;
   }

   public Pagereference SaveAttachments()
```

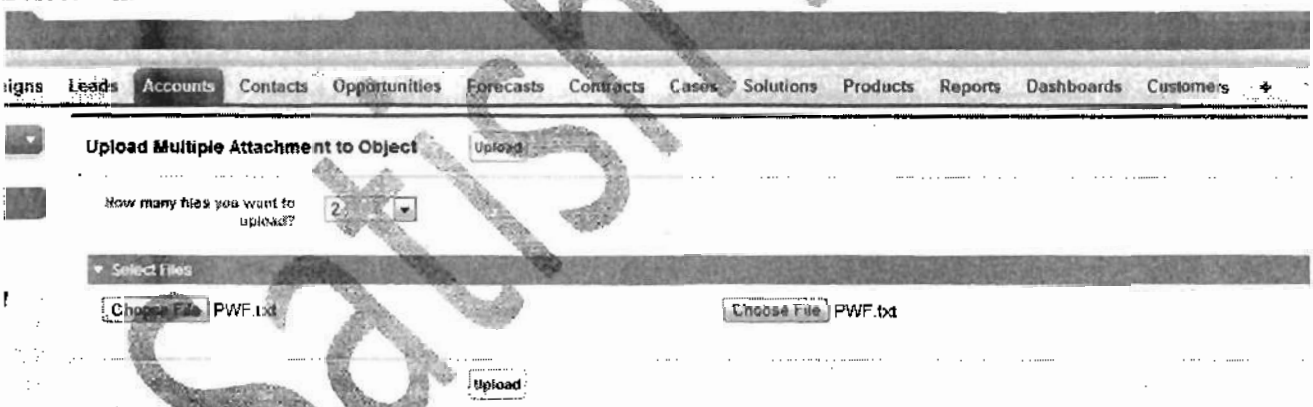
```
{
    String accId = System.currentPagereference().getParameters().get('id');
    if(accId == null || accId == '')
        ApexPages.addmessage(new ApexPages.message(ApexPages.Severity.ERROR,'No record is associated.
Please pass record Id in parameter.'));
    if(FileCount == null || FileCount == '')
        ApexPages.addmessage(new ApexPages.message(ApexPages.Severity.ERROR,'Please select how many
files you want to upload.'));

    List<Attachment> listToInsert = new List<Attachment>() ;

    //Attachment a = new Attachment(parentId = accid, name=myfile.name, body = myfile.body);
    for(Attachment a: allFileList)
    {
      if(a.name != '' && a.name != '' && a.body != null)
        listToInsert.add(new Attachment(parentId = accId, name = a.name, body = a.body)) ;
    }

    //Inserting attachments
    if(listToInsert.size() > 0)
    {
      insert listToInsert ;
ApexPages.addmessage(new ApexPages.message(ApexPages.Severity.INFO, listToInsert.size() +
                                                    ' file(s) are uploaded successfully'));
      FileCount = '';
    }
    else
      ApexPages.addmessage(new ApexPages.message(ApexPages.Severity.ERROR,'Please select at-least
one file'));

    return null;
  }
  public PageReference ChangeCount()
  {
    allFileList.clear() ;
    //Adding multiple attachments instance
    for(Integer i = 1 ; i <= Integer.valueOf(FileCount) ; i++)
      allFileList.add(new Attachment()) ;
```

```
    return null ;
  }
}
```

```
<apex:page standardController="Account" extensions="MultipleUploadController">
  <apex:form>
    <apex:pageBlock title="Upload Multiple Attachment to Object">
            <apex:pageBlockButtons>
      <apex:commandButton value="Upload"  action="{!SaveAttachments}"/>
      </apex:pageBlockButtons>

      <apex:pageMessages id="MSG"/>
      <apex:actionFunction name="ChangeCount" action="{!ChangeCount}"/>

      <apex:pageblocksection>

        <apex:pageBlockSectionItem>
          <apex:outputLabel value="How many files you want to upload?"/>
          <apex:selectList onchange="ChangeCount() ;" multiselect="false" size="1" value="{!FileCount}">
            <apex:selectOption itemLabel="--None--" itemValue=""/>
            <apex:selectOptions value="{!filesCountList}"/>
          </apex:selectList>
        </apex:pageBlockSectionItem>

      </apex:pageblocksection>

      <apex:pageBlockSection title="Select Files" rendered="{!IF(FileCount != null && FileCount != '', true ,
false)}">
        <apex:repeat value="{!allFileList}" var="AFL">
          <apex:inputfile value="{!AFL.body}" filename="{!AFL.Name}" />
                  </apex:repeat>
      </apex:pageBlockSection>
       </apex:pageBlock>
  </apex:form>
</apex:page>
```



Select 2 files

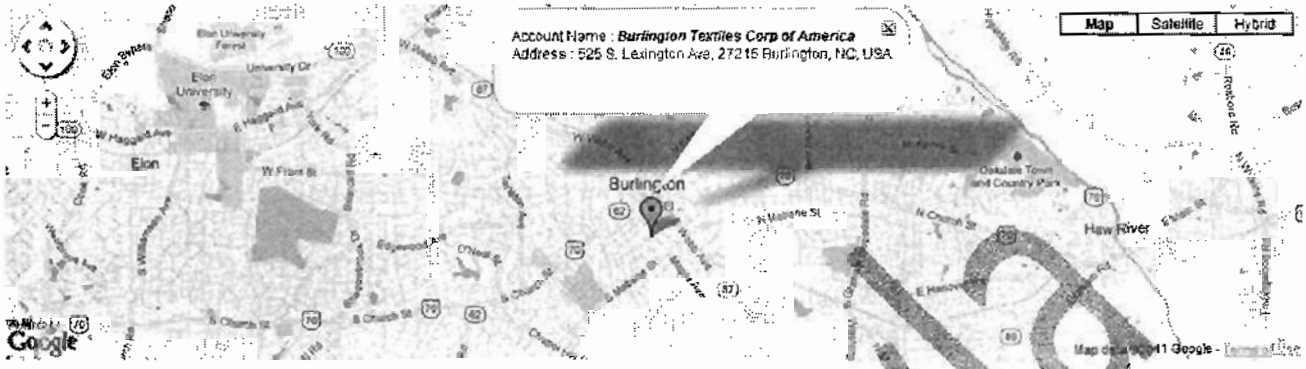


Senario 14: Google Maps with salesforce:

```
<apex:page standardController="Account">
<script src="http://maps.google.com/maps?file=api">
</script>
<script type="text/javascript">
var map = null;
var geocoder = null;
```

```
var address = "{!Account.BillingStreet}, {!Account.BillingPostalCode} {!Account.BillingCity},
{!Account.BillingState}, {!Account.BillingCountry}";
function initialize() {
if(GBrowserIsCompatible())
{
 map = new GMap2(document.getElementById("MyMap"));
 map.addControl(new GMapTypeControl());
 map.addControl(new GLargeMapControl3D());

 geocoder = new GClientGeocoder();
 geocoder.getLatLng(
  address,
  function(point) {
   if (!point) {
    document.getElementById("MyMap").innerHTML = address + " not found";
   } else {
    map.setCenter(point, 13);
    var marker = new GMarker(point);
    map.addOverlay(marker);
    marker.bindInfoWindowHtml("Account Name : <b><i> {!Account.Name} </i></b>
Address : "+address);
   }
  }
 );
}
}
</script>


<div id="MyMap" style="width:100%;height:300px"></div>
<script>
  initialize();
</script>

</apex:page>
```

Give the id of the Account Record id .



Senario 15::Create Dynamic Record in Sobject selected .

```
<apex:page controller="InsertDynamicSobjectController">
  <apex:form >
    <apex:pageBlock >
      <apex:pageBlockButtons >
        <apex:commandButton value="Save" action="{!Save}"/>
      </apex:pageBlockButtons>
      <apex:pageMessages />
      <apex:pageBlockSection >
        <apex:pageBlockSectionItem >
          <apex:outputLabel value="Enter Object Name"/>
          <apex:inputText value="{!ObjectName}"/>
        </apex:pageBlockSectionItem>
        <apex:pageBlockSectionItem >
          <apex:outputLabel value="Enter Name for Record"/>
          <apex:inputText value="{!RecordName}"/>
        </apex:pageBlockSectionItem>
      </apex:pageBlockSection>
    </apex:pageBlock>
  </apex:form>
</apex:page>
```

Apex Class:

```
public class InsertDynamicSobjectController
{
```

```
public String ObjectName {get; set;}

public String RecordName {get; set;}

public InsertDynamicSobjectController()
{
   ObjectName = '' ;
   RecordName = '' ;
}

public PageReference Save()
{
   //use GlobalDecribe to get a list of all available Objects
   Map<string, schema.sobjecttype> gd = Schema.getGlobalDescribe();
   Set<string> objectKeys = gd.keySet();
   if(objectKeys.contains(Objectname.toLowerCase()))
   {
      try
      {
         //Creating a new sObject
         sObject sObj = Schema.getGlobalDescribe().get(ObjectName).newSObject() ;
         sObj.put('name' , RecordName) ;
         insert sObj ;

         PageReference pg = new PageReference('/'+sObj.Id) ;
         return pg ;
      }
      Catch(Exception e)
      {
         ApexPages.addMessages(e) ;
         return null ;
      }
   }
   else
   {
ApexPages.addmessage(new ApexPages.message(ApexPages.severity.ERROR,'Object API name is
                                    invalid')) ;
      return null ;
```

```
    }
  }
}
```


Leads Accounts Contacts Opportunities Forecasts Contracts Cases Solutions Products Reports Dashboards Customers
Save
Enter Object Name
Account
Satish Example
Save
Satish Example
Post File New Task More
Follow
Followers
Show All Updates
There are no updates.
Account Detail
Edit Delete Include Offline
Account Owner
Account Name
Rating
Phone
Fax
Website

Example on pageBlockTable :-

```
<apex:page StandardController = "Account" recordSetVar = "items" >
  <apex:pageBlock>
    <apex:pageBlockSection>
     <apex:pageBlock>
       <apex:pageBlockTable value = "{!items}" var="a"
                            frame = "border">
        <apex:column value = "{!a.name}" />
        <apex:column value = "{!a.industry}" />
       </apex:pageBlockTable>
     </apex:pageBlock>
     <apex:pageBlock>
       <apex:pageBlockTable value = "{!items}" var = "a"
                            frame = "border" >
        <apex:column value = "{!a.name}" />
        <apex:column value = "{!a.industry}" />
       </apex:pageBlockTable>
     </apex:pageBlock>
    </apex:pageBlockSection>
  </apex:pageBlock>
</apex:page>
```

Example :-

```
<apex:page standardController="Account" recordSetVar="items">
 <table>
  <tr>
   <td width="50%">
       <span style="width:100px;">
       <apex:pageBlock>
       <apex:dataList value="{!items}" var="a"
                      style="width=50%;">
              {!a.name}
       </apex:dataList>
      </apex:pageBlock>
     </span>
</td><td width="10%">
  </td>
 <td width="50%">
  <span>
<apex:pageBlock>
<apex:dataList value="{!items}" var="a" style="width=50%;">
        {!a.name}
  </apex:dataList>
</apex:pageBlock>
 </span>
 </td>
 </tr>
 </table>
</apex:page>
```

Example :-

```
<apex:page StandardController="Account" recordSetVar="items">
  <apex:pageBlock>
    <apex:pageBlockSection>
      <apex:pageBlockTable value="{!items}" var="a">
       <apex:column value="{!a.name}"/>
      </apex:pageBlockTable>
      <apex:dataTable value="{!items}" var="a">
        <apex:column value="{!a.name}"/>
        </apex:dataTable>
      <apex:dataList value="{!items}" var="a">
           {!a.name}
        </apex:dataList>
      <apex:repeat value="{!items}" var="a">
         {!a.name}
       </apex:repeat>
    </apex:pageBlockSection>
  </apex:pageBlock>
</apex:page>
```

usage of css within visualforce :-

css → cascading style sheet.

There are 3 types of css.

1) Inline
2) Internal
3) External.

1) Inline :- If we define a style with in a component we call it as inline.

Eg:- `<apex:outputLabel style="color:blue;">`

2) Internal :- using def<style> tage with in the visual force page.

Eg:-
```
<apex:page>
    <style>
        .one
        {
            color:blue;
        }
    </style>
    <apex:outputLabel styleclass="one" value="mytext"/>
    <apex:outputLabel styleClass="two" value="myvalue"/>
</apex:page>
```

3) External :- Defining a css file outside the page & outside the salesforce and using that within a visualforce page is called External css.

1) Internal css file

Internal Style sheet can be define in 3 ways by using

i) Id
ii) class
iii) Component selector

→ Style properties can be assigned to a component by using id

```
#one → id name
{
color: blue;
}
```

→ Style properties can be assigned to a component by using class

```
.one → class name
{
color: blue;
}
```

Borders properties :-

- border-style
- border-width
- border-color
- border-top-style
- border-right-style
- border-left-style

NOTE :-> Box styles are
→ Dotted
→ Dashed
→ Double
→ Solid.

When we want to make a box the properties are given as
→ Width
→ Height
→ Padding
→ Margin.

<apex:detail> :-

This will display the standard detail page for the particular object.

Attributes :-

Subject :- Id of the record that should provide the data for this component.

Relatedlist :- Boolean value that specifies whether it included in given component.

Relatedlisthoverylink :- A boolean value that specifies whether the related list hover links are included in the rendered Component.

Example :-

```
<apex:page StandardController = "Customers__c">
 <apex:detail relatedlist = "true" Subject = "{!Customers__c.id}"
      relatedListHovers = "true" inlineEdit = "true"
        showChatter = "true"/>
 </apex:page>
```

→ Create a one more visual force that displays the detail page of the record along with the related list.

```
<apex:page StandardController = "Account">
 <apex:detail relatedlist = "true" Subject = "{!Customer Account.id}"
  relatedListHovers = "true" inlineEdit = "true"/>
 </apex:page>
```


<apex:enhancedlist> :-

The list view picklist for an object including its associated records for accessing selecting views in the standard salesforce application will be displayed.

Attributes :-

ListId :- Salesforce object for which views are displayed.

Height :- An integer value that specifies the height of the list in pixels. This is required value.

Customizable :- Boolean value that specifies whether the list can be customized & not. Default value is true.

rowsperpage :- An integer value that specifies number of rows per page. The possible values are 10, 25, 50, 100 & 200.

type :- The salesforce object for which views are created.

Eg :-

```
<apex:page>
  <apex:enhancedList type="Account" height="300"
    rowsperpage="10" id="AccountList"/>
  <apex:enhancedList type="Lead" height="300"
    rowsperpage="25" id="LeadList" customizable="False"/>
</apex:page>
```

<apex:listviews> :-

This list views displays the listviews picklist for an object including its associated list of records for currently selecting views.

Attributes :-

type :- salesforce object for which the listviews are displayed.

Eg :- type="Customer__c"

Eg:-

```
<apex:page showHeader="true" tabStyle="Case">
    <apex:enhancedList type="Customers__c" height="300"/>
    <apex:listViews type="Customer__c"/>
</apex:page>
```

<apex:include>:-

This Component which inserts another visualforce page into Current page.

Attributes:-

pageName:- the visualforce page whose content should be inserted into the current page. For this value, Specify the name of the visualforce page.

Eg:-

```
    pageName : Firstpage
<apex:page>
    <apex:form>
        <apex:pageBlock>
            <apex:outputLabel value="Enter Name"/>
            <apex:inputText id="one"/>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

pageName : InsertExample

```
<apex:page>
    <apex:sectionHeader title="Homepage" Subtitle="Mypage"/>
    <apex:insert include pageName="Firstpage"/>
    <apex:pageBlock title="footer"/>
        thank you
    </apex:pageblock>
</apex:page>
```

<apex:iframe> :-

This will create entire source as a inline frame within a visualforce page.

Attributes :-

Src :- the URL that specifies the initial content of the inline frame. the URL can be either a external website, or another page in the application.

Width, height, id.

Scrolling :- it's a boolean value that specifies whether inline frame can be scrolled & not.

Eg:-

```
<apex:page>
 <apex:outputText> Example for this </apex: outputText>
  <apex:include pageName = "Mypage"/>
  <apex:iframe src = "https://ap1.Salesforce.com/a009000oMPAR2"/>
</apex:page>
```

<apex: insert>:-

With in a template we can created named area which can be defined by the area based on his requirement.

Name:- name of the area in the template.

Eg:- <apex:page> pageName: Composition.

```
    <apex:outputText value = "{!template) this is before the
                                        header "/> <br/>
  <apex:outputT
  <apex:insert name = "header"/> <br/>
  <apex:outputText value = "(template) this is between the header
                                    and body"/> <br/>
  <apex:insert name = "body"/>
  </apex:page>
```

pageName : page

<apex:page>

<apex:composition template="composition">

<apex:define name="header">(page) this is the header of mypage </apex:define>

<apex:define name="body">(page) this is the body of mypage </apex:define>

</apex:composition>

</apex:page>

<apex:selectoption list> :-

value :- A merge field that references the controller class variable this will store the value of the picklist that you have selected.

NOTE :- If the value is a variable which will store the value of the option that you have selected from the variable.

size :- Is a number of selected list options displayed one time on vf page.

Readonly

Multiselect :- This is a boolean value, if we give true, it converts picklist field into multipicklist field.

Disabled :- If we give this boolean value as true then picklist will be displayed in disabled mode.

Example :-

```
<apex:page StandardController = "Account" recordsetVar="items">
  <apex:form>
    <apex:pageBlock>
      <apex:outputLabel > Select State </apex:outputLabel>
      <apex:selectList size="1">
          <apex:selectOption itemLabel = "Andhra" itemValue="Ap"/>
          <apex:selectOption itemLabel = "Tamil" itemValue = "TN"/>
          <apex:selectOption itemLabel = "Karnataka" itemValue = "Ka"/>
          <apex:selectOption itemLabel = "Utarpradesh"
                                          itemValue = "UP"/>
      </apex:selectList>
    </apex:pageBlock>
    <apex:pageBlock>
      <apex:selectList size = "1" value = "{! filterId }"
                              label = "Account View">
      <apex:selectOptions value = "{! listViewOptions }" >
      </apex:selectOptions>
      <apex:actionSupport event = "onchange" reRender = "one"/>
    </apex:selectList>
    <apex:pageBlockTable value = "{! items }" var = "a". id = "one">
```

```
<apex:column value="{!a.name}"/>
<apex:column value="{!a.industry}"/>
</apex:pageBlockTable>
<apex:selectList size="3" multiselect="true">
<apex:selectoption itemlabel="Andhra" itemValue="AP"/>
<apex:selectoption itemlabel="Tamil" itemvalue="TN"/>
<apex:selectoption itemlabel="karnataka" itemvalue="ka"/>
</apex:selectlist>
</apex:pageBlock>
</apex:form>
</apex:page>
```

→ In the above example value = "{!listviewoptions}" will fetch list view options of Account object.

→ value = "{!filterid}" which will display the recently selected filters from the list view options.

<apex:selectcheckboxes>:-

Borders :- The width of the frame around the rendered HTML table in pixels.

Bisp Bordervisible :- It is a boolean value controls whether the borders around the <fieldset> i

Layout :- This specifies whether the borders has to be displayed as horizontally & vertically.

legendtext :- Text that will be displayed on the group of checkboxes.

Example :-

```
<apex:page>
  <apex:form>
    <apex:selectCheckboxes borderVisible = "true" layout="PageDirection"
        legendText = "course" legendValue = "
    <apex:selectoption itemlabel = "Java" itemValue = "Java" >
        </apex:selectoption>
    <apex:selectoption itemlabel= "oracle" itemValue = "oracle">
        </apex:selectoption>
    <apex:selectoption itemlabel = "sfdc" itemvalue = "sfdc">
        </apex:selectoption>
    </apex:selectCheckboxes>
  </apex:form>
</apex:page>
```

<apex:selectradio> :-

```
<apex:page>
<apex:form>
  <apex:selectRadio legendtext = "MyCourse" borderVisible="true">
    <apex:selectoption itemlabel = "Java" itemValue = "Java">
        </apex:selectoption>
```

```
<apex:selectoption itemlabel="SFDC" itemValue="SFDC">
                </apex:selectoption>
<apex:selectoption itemlabel="cloud" itemValue="cloud">
        </apex:selectoption>
</apex:selectradio>
</apex:form>
</apex:page>
```

<apex:pageMessage>:-

Severity:- severity of the message values supported are confirm, info, warning error.

Strength:- The strength of the message. this controls the visibility and size of the icon displayed next to the message. 0 represents no image.

Example :-

```
<apex:page>
<apex:form>
    <apex:pageMessage severity="Warning" strength="2"
                Summary="This Warning is Example"
detail="In this VF page we created page message example">
```

```
</apex:pageMessage>
</apex:form>
</apex:page>
```

Css :-

→ Css stands for cascading style sheet. Css will be used to define the properties of the attribute.

→ Css can be defined in 3 ways.

1) Inline Css :- Where the style of the Component is defined with in the Component by using an attribute style.

Eg :-

```
<apex:page>
    <apex:form>
        <h1 style="text-align:center; background-color:red;">
                                    Welcome </h1>
        <apex:outputLabel Style="color:blue; text-align:center;">
                                    Helloo </apex:outputLabel>
    </apex:form>
</apex:page>
```

Text properties :-

Centered, Right, Justify

Over-line

line-through

underline

text-transform : lowercase

text-transform : uppercase

text-transform : capitalize.

→ In the inline css we are going to define the property with in the Component it self.

Eg:-

```
<apex:page>
   <apex:form>
   <h1 Style ="text-decoration:line-through; background-color:red;
        text-transform : lowercase; " > Welcome to this </h1><br/>
   <apex:outputLabel  Style = "color:blue; text-decoration:line-through; "
                        value ="Hello"/>
   <apex:commandButton  value = "click"  Style ="background : blue;
        Color ; red; width : 100px; height : 100px; font-size : 20px;"/>
   </apex:form>
</apex:page>
```

Background effects:-

Background-color

Background-image

Background-repeat

Background-attachment

Background-position

→ When ever we want to use background image in the css.

1) Load the image into static resources.

2) Click on view file in the static resources. Then you will get URL. Then you will get the URL.

Eg:-

```
<apex:page>
  <apex:form>
    <apex:commandButton value="click" Style="background-image:
          URL('/resource/1383105671000/download');
  background-repeat:no-repeat; color:red; width:200px;
        height:100px; font-Size:20px;"/>
  </apex:form>
</apex:page>
```

Fonts :-

font:family; Verdana, Times new roman

font-style ; italic, Bold.

font-Size ; 1em=60px;

Font properties :-

~~fon~~

font: family

font: style

font: Size

font: Weight

Eg:-

```
<apex:page>
  <apex:form>
    <apex:outputlabel style="font-family:serif; font-size:40px;
        font-style:Bold "/>
  </apex:form>
  </apex:page>
```

2) Internal css :-

→ If you define the css with in the visualforce page then it is called as internal css.

→ The internal css should be defined with in the <script> tag.

→ The css properties can be defined in 3 ways.

1) By using id.
2) By using class
3) By using tag.

Eg:-

```
<style type="text/css">
    #one
      {
       color:blue;
      }
    .two
      {
       color:red;
      }
```

#id

.className

```
h1
{
  color:green;
}
</style>
```

h1 - HtmlTagElement.

Example :-

```
<apex:page StandardController="Contact" recordSetVar="items"
                                        sidebar="false">
  <apex:form>
    <style>
      .one
      {
        margin:20px;
        padding:10px;
        width:150px;
        height:100px;
        background-color:#81DAF5;
            border-radius:20px;
        }   border:2px solid red;
      .outer
      {
        margin:10px;
        padding:10px;
        width:250px;
        height:500px;
        background-color:#FFFFFF;
          }
      .record
        {
          margin:20px;
```

```
        Padding : 10px;
        Width : 150px;
        height : 20px;
        background-color : #810AF5;
        }
    </style>
    <div>
      <div class="recent">
          <apex:outputLabel style="font-size:20px;">
              Recent Items </apex:outputLabel>
      </div>
    <div class="one">
    <apex:dataTable value="{!items}" var="a" rows="5"
                    first="3">
    <apex:column>
    <apex:image url="resource/1383--/download" Style="
          width:10px; height:10px; padding:5px;
                    float:left;"/>
    <apex:commandLink value="{!a.name}" action="{!a.}"/>
    </apex:column>
  </apex:dataTable>
  </div>
</apex:form>
</apex:page>
```

3) External css :-

In External css we can define the css file outside the salesforce and we can load this file as a static resource and use it in all the visualforce pages.

step1 :- Create Mystyle.css file.

Eg:

```
.one
{
margin:20px;
padding: 10px;
width : 150px;
height : 100px;
background-color : #81DAF5;
border-radius :20px;
border : 2px solid red;
    -webkit-perspective :250; /*
}
.outer
{
margin : 10px;
padding : 10px;
width : 250px;
height : 500px;
background-color : #FFFFFF.
    -webkit-transform : rotateY(40deg);
}
.accent
{
margin : 20px;
padding : 10px;
width : 150px;
height : 200px;
background-color : #81DAF5;
```

```
border-radius : 10px;
}
```

Step2:- Load this MyStyle.css file as Static resource in Salesforce.

Setup → Build → Develop → Static Resource → New

File name : satish css

Choose File : MyStyle.css

Cache : public

Step3 :- Include the Stylesheet in Vf page

```
<apex:StyleSheet value="{!$Resource.satishcss}"/>
```

NOTE :- Any resource/file stored to static resource will be called by $Resource.ResourceName.

External css.vf :-

```
<apex:page standardController="Contact" recordSetVar="items">
 <apex:form>
  <apex:Stylesheet value="{!$Resource.satishcss}"/>
    <div>
      <div class="recent">
       <apex:outputLable style="font-size:20px;">Recent Items
                                    </apex:outputlabel>
      </div>
    <div class="one">
     <apex:datatable value="{!items}" var="a" rows="5"
```

```
        first="3">
    <apex:column>
<apex:image  url="/resource/1383105071000/download"
        style="width:100px; height:100px; padding:5px; float:left;"/>
  <apex:commandLink value="{!a.name}" action="{!a.id}"/>
  </div>
  </apex:column>
 </apex:dataTable>
 </apex:form>
</apex:page>
```

Javascript :-

→ Using javascript in visualforce pages gives you access to a wide range of exisisting javascript functionality, such as javascript libraries, and other ways to customize the functionality of your Pages.

→ Action tags such as <apex:actionfunction> and <apex:actionsupport> Support Ajax requests.

→ The best method for Including javascript in a visualforce page is placing the javascript in a Static resource, then calling it from there.

Eg:- <apex:includescript value="{!$Resource.jscriptfile}"/>

→ you can then use the functions defined within that javascript file with in your page using <script> tag.

Eg:-
```
<apex:page>
    <apex:form>
     <script>
       alert('Hello');
     </script>
    </apex:form>
</apex:page>
```

→ To define a function in javascript we use the following syntax.

Syntax:- function functionname (parameters)

{

}

NOTE:- In javascript we will store all types of variables in "var" whenever we want we can typecaste it.

→ When ever the event is raised then the javascript function will be called. (onclick, onchange etc.)

Using javascript to Reference Components:-

To refer to a Visualforce Component in javascript or another web-enabled language, you must specify a value for the id attribute for that component.

→ To bind visualforce Components together, this id is used to form part of the document object model (DOM) id for the Component when the page is created.

Eg:- 1) this example uses the DOM ID for an <apex:outputpanel> tag. The page contains two panels. First holds a checkbox that fires a DOM event, and the second contains some text that's changed in response to the event.

```
<apex:page id="thepage">
    <script>
      function changefont(input, textid)
          {
           if(input.checked){
          document.getElementById(textid).style.fontweight
                                          ="bold";
                }
          else {
          document.getElementById(textid).style.fontweight
                                          ="normal";
                }
           }
      </script>
  <apex:outputpanel layout="block">
    <label for="checkbox">click</label>
      <input id="checkbox" type="checkbox"
      onclick="changefont(this,'{!$Component.thepanel}');"/>
  </apex:outputpanel>
  <apex:outputpanel id="thepanel" layout="block">
          Change my font weight!
    </apex:outputpanel>
</apex:page>
```

NOTE:- The $!$Component.thepanel$ expression is used to obtain the DOM ID of the HTML element generated by the <apex:outputpanel id="thepanel"> Component.

Eg:-2)

```
<apex:page id="pg">
  <apex:form id="frm">
  <apex:pageBlock id="pb">
 <Script type="text/JavaScript">
    function show()
      {
    var name = document.getElementById('$!$Component.one$').
                                                value;
Var myname = document.getElementById('pg:frm:pb:one').value;
      alert(name);
      }
  </script>
<apex:commandButton value="click" action='$$!show$'/>
  </apex:pageBlock>
</apex:form>
</apex:page>
```

Eg3:- Create a vf page to check whether the entered username & password are correct or not, if entered one is not correct give alert message.

```
<apex:page>
  <apex:form>
   <apex:pageBlock id = "pb">
    <apex:outputLabel > Enter username </apex:outputLabel>
    <apex:inputText  id = "Uid" />
  <apex:outputLabel > Enter password </apex:outputLabel>
   <apex:inputText  id = "pwd" />
<apex:commandButton value = "login" action = "{!show}" />
  </apex:pageBlock>
  <Script>
   function show()
      {
  var uses = document.getElementById ('{!$Component.pb.
                                        Uid}').value;
  Vas passwd = document.getElementById ('{!$Component.pb.
                                        pwd}').value;
```

```
if(password=='' || uesr==' ')
    alert('check the user id and password');
    else
     alert('Login Success')
</script>
</apex:form>
</apex:page>
```


Usage of external javascript file in visualforce page:-

→ you can include the javascript files in your visualforce pages to take advantage of functionality provided by these libraries.

→ The best way to include javascript libraries is by creating a static resource, and then including the library by adding an <apex:includescript> Component to your page.

Eg:- 1)

Step1:- Create external script file with the name Myname.js.

```
function show()
    {
    alert("Welcome");
     }
```

Step2:- Now load this file as a static resource in the visualforce page with the resource name as MyScript.

Step3:- Create a visualforce page.

```
<apex:page>
 <apex:form>
 <apex:includescript value = "{!$Resource.MyScript}"/>
<apex:CommandButton value = "clickme" onclick="show()"/>
</apex:form>
</apex:page>
```

Eg2:-

For Eg., if you are using MOOTools, then create a static resource from the library called mootools. Then

```
<apex:page>
 <apex:includescript value = "{!$Resource.mootools}"/>
```

```
<script>
    function changeFont(input,textid)
     {
     if(input.checked) $(textid).style.fontweight='bold';
     else  $(textid).style.fontweight="normal";
          }
     </script>

<h1> Congratulations </h1>
 <apex:outputpanel  layout="block" >
   <label  for="checkbox" > click </label>
<input  id ="checkbox"  type ="checkbox"
     onclick = "changeFont (this, '{!$Component.thepanel}');"/>
</apex:outputpanel>
<apex:outputpanel id="thepanel" layout="block">
                    change me!
  </apex:outputpanel>
 </apex:page>
```

Javascript Remoting for Apex Controllers :-

→ use Javascript remoting in visualforce to call methods in Apex controllers from Javascript.

Javascript remoting has 3 parts.

→ the remote method invocation you add to the visualforce page, written in Javascript.

→ the remote method definition in your Apex controller class. This method definition is written in Apex, but there are few differences from normal action methods.

→ the response handler callback function you add to or include in your vf page, written in Javascript.

Adding Javascript remoting to a vf page :-

→ To use javascript remoting in a vf page, add the request as a javascript invocation with the following form.

```
[namespace.]Controller.method(
    [parameters... ,]
    callbackfunction,
    [Configuration]
);
```

→ Name space is the name space of the Controller class.

→ controller is the name of your Apex controller.

→ method is the name of your Apex Controller method you are calling.

→ parameters is the comma-seperated list of parameters that your method takes.

→ callback function is the name of the Javascript function that will handle the response from the controller.

→ Configuration configures the handling of remote call and response.

Configuring Javascript remoting requests :-

Configure a remoting request by providing an object with configuration settings when you declare the remoting request.

→ Javascript remoting supports the following configuration parameters.

Name	DataType	Description
buffer	Boolean	Whether to group requests executed close to each other in time into a single request. The default is true.
escape	Boolean	Whether to escape the Apex method's response. The default is true.
timeout	Integer	The timeout for the request in milliseconds. Default is 30000 (30 seconds)

Declaring a remote method :-

→ In your Controller, your Apex method declaration is preceded with the @RemoteAction annotation like this

```
@RemoteAction
global static String getItemId(String objectName){...}
```

Eg:-
→ your method can take Apex primitives, collections, typed and generic sobjects, and user-defined Apex classes & interfaces as arguments.

Eg:-

Step1:- create an apex class which has got remote method.

If we want to declare any method as remote we have to use @RemoteAction annotation.

→ The Remote method must be Static method.

```
global class Example
{
@RemoteAction
global static String myfun(String xyz)
{
return 'Myname' +xyz;
}
}
```

Step2:- Load the apex class into the visualforce page.

→ If we want to invoke the remote action we have to call a method

visualforce.Remoting.Manager.InvokeAction('{!$RemoteAction.Example.Show}',

Example → Controller class name
Show → Remote Method name.

function(result, event)

{ = { → Response handler.

}

Usage of Remote method in javascript using VF page:-

Eg:-

```
<apex:page Controller="Example">
  <apex:form>
    <script type="text/javascript">
      function myJS()
        {
      Var name = document.getElementById(
                  '{!$Component.myname).value;
    visualforce.Remoting.Manager.InvokeAction
          -('{!$RemoteAction.Example.myfun()'},
                  name,
```

```
function (result, event)
    {
    if (event.status)
        {
    document.getElementById ('{!$(component.one).
                        InnerHTML = result);
            }
        else
            {
            if (event.type = 'Exception')
                {
                }
            }
        }
</script>
</apex:form>
</apex:page>
```

Second way of calling method in javascript using VF page :-

```
Example.myfun(parameter, function(result, event))
            {
            if (event.status)
                {
                }
            else if (event.type = "Exception")
                {
                }
            }
        );
```

Javascript Remoting Example :-

```
<apex:page Controller = "RemoteExample">
  <script type = "text/javascript">
    function myjs()
      {
      var name = document.getElementById("myname").value;
    RemoteExample.show(name, function(result, event))
        {
        if(event.status)
          {
        document.getElementById("one").innerHTML=result;
        }
      });
    }
  </script>
  <input type = "text" id = "myname"/>
  <button onclick = "myjs()">click</button>
  <p id = "one"> </p>
</apex:page>
```

Force.com IDE Installation

The Force.com IDE is available as a plug-in for the industry-leading, open-source Eclipse project.

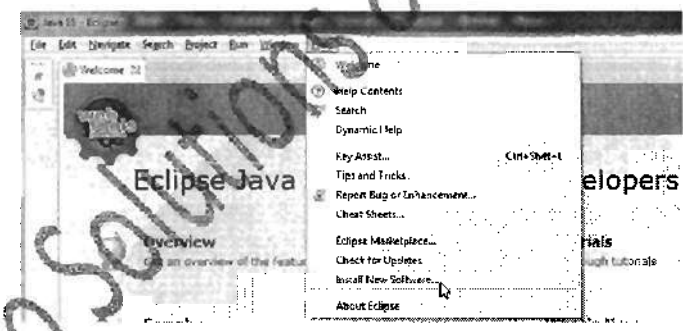
These instructions show you how to install the Force.com IDE into your existing Eclipse distribution or upgrade from a previous version.

Prerequisites

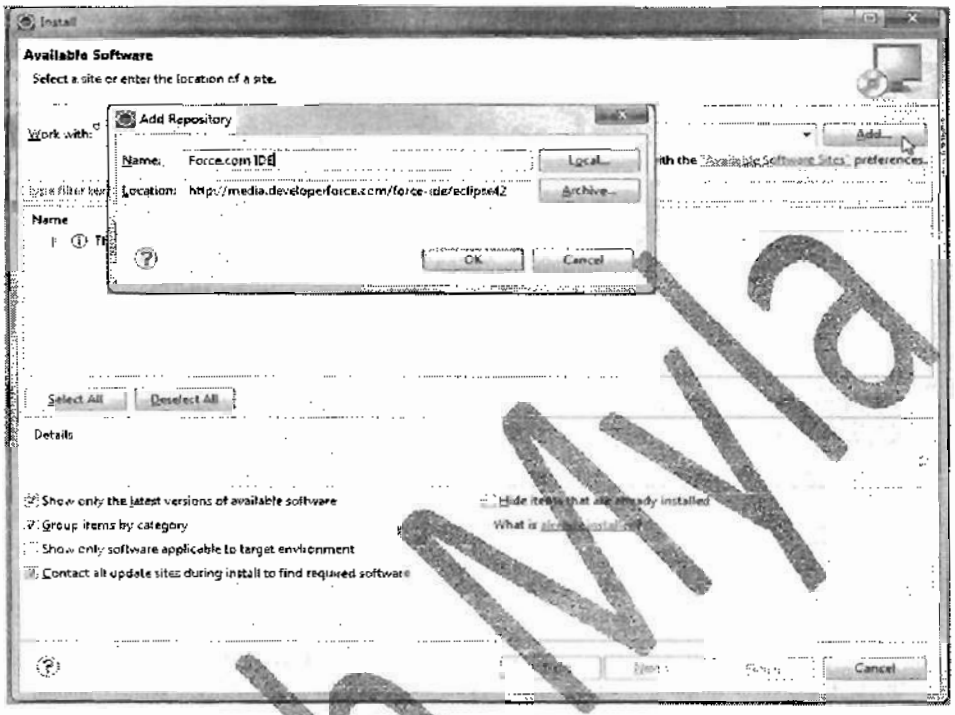
- Java SE Runtime Environment 6 (v1.6) or later
- Eclipse 4.2 "Juno" (Eclipse 4.2 download site) or Eclipse 4.3 "Kepler" (Eclipse 4.3 download site) - The 'Eclipse IDE for Java Developers' distribution is strongly recommended.

Installation Steps

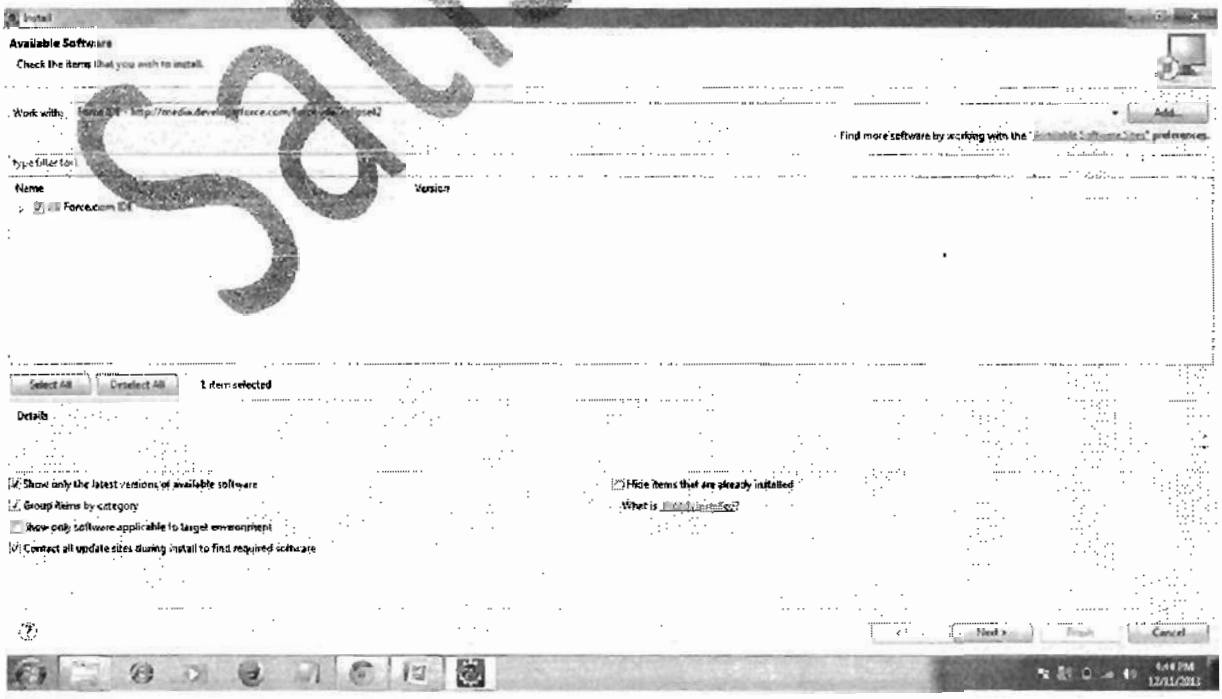
1. Launch Eclipse and click **Help > Install New Software....**

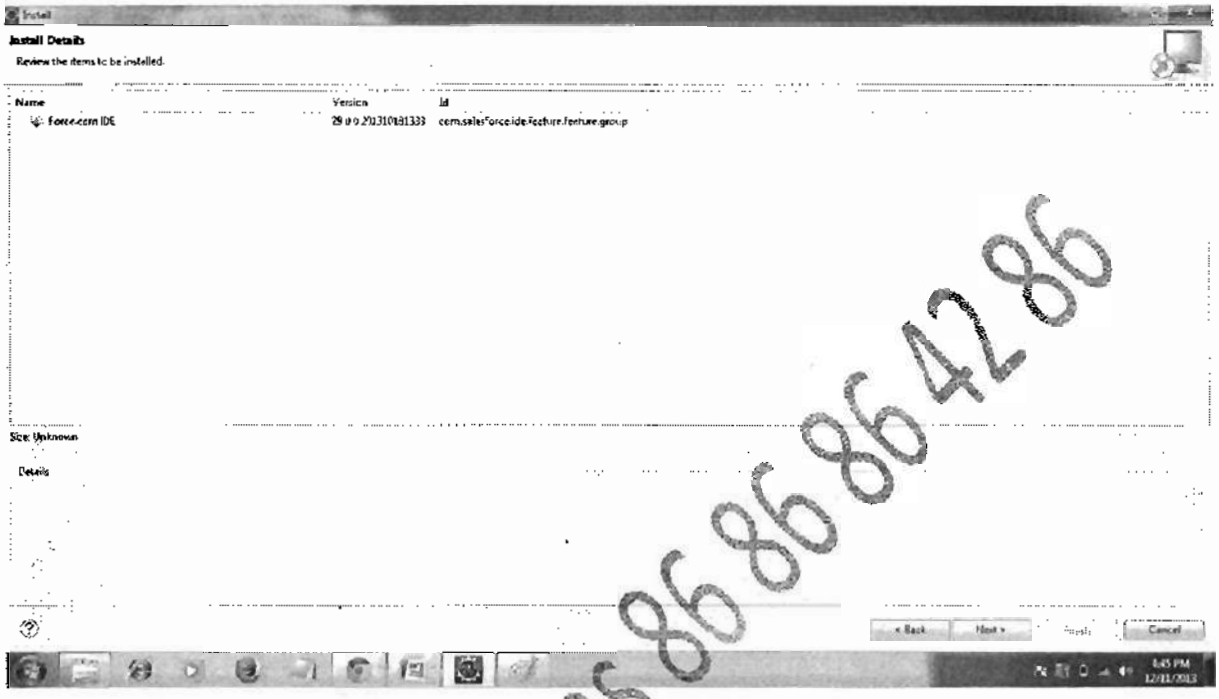


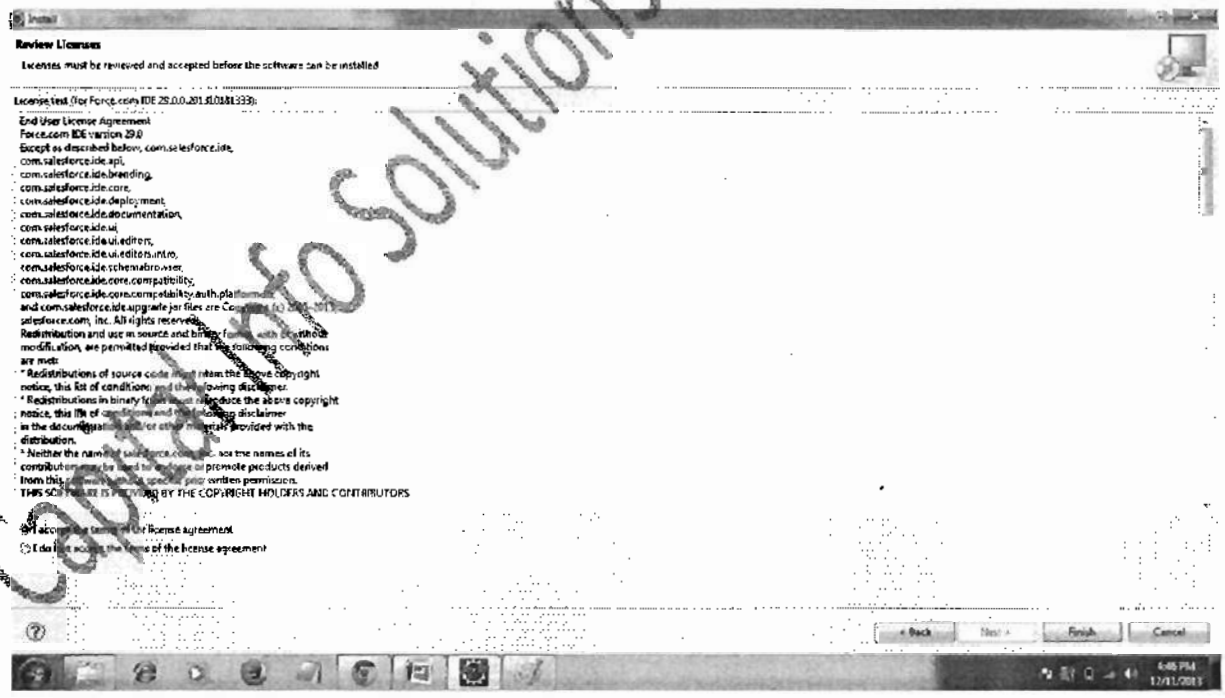
2. Click **Add...**
3. In the **Add Repository** dialog, set the **Name** to "Force.com IDE" and the **Location** to "http://media.developerforce.com/force-ide/eclipse42" and click **OK**. (Use the same URL for Eclipse 4.3.)

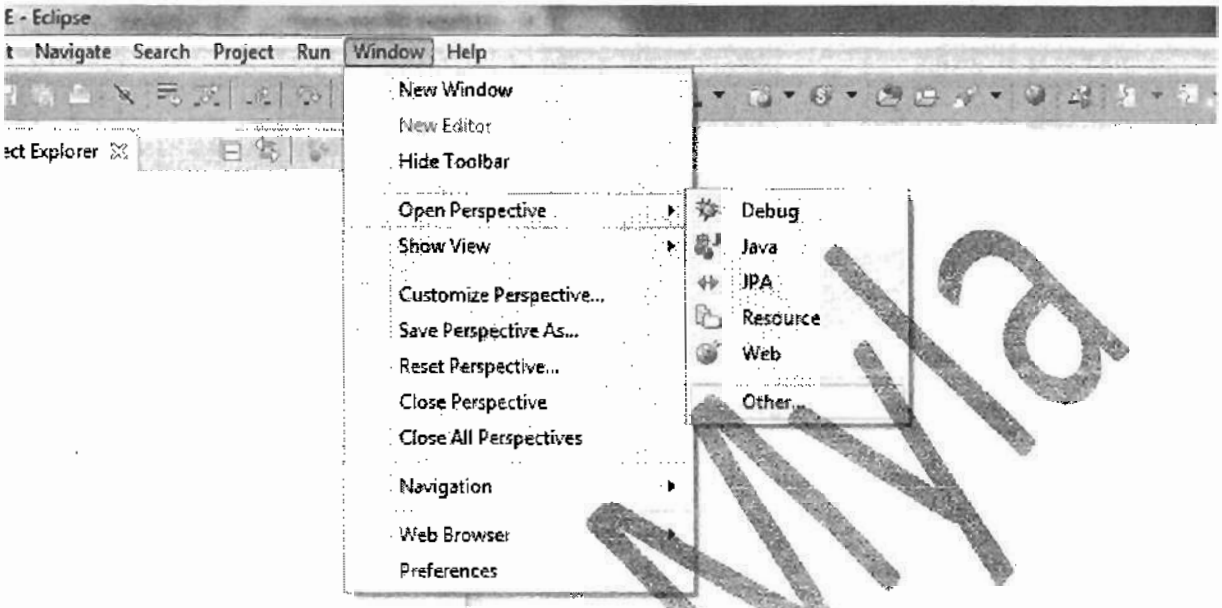


4. Eclipse downloads the list of available plugins and displays them in the Available Software dialog.
5. Check the box next to the Force.com IDE plugin and click **Next**.

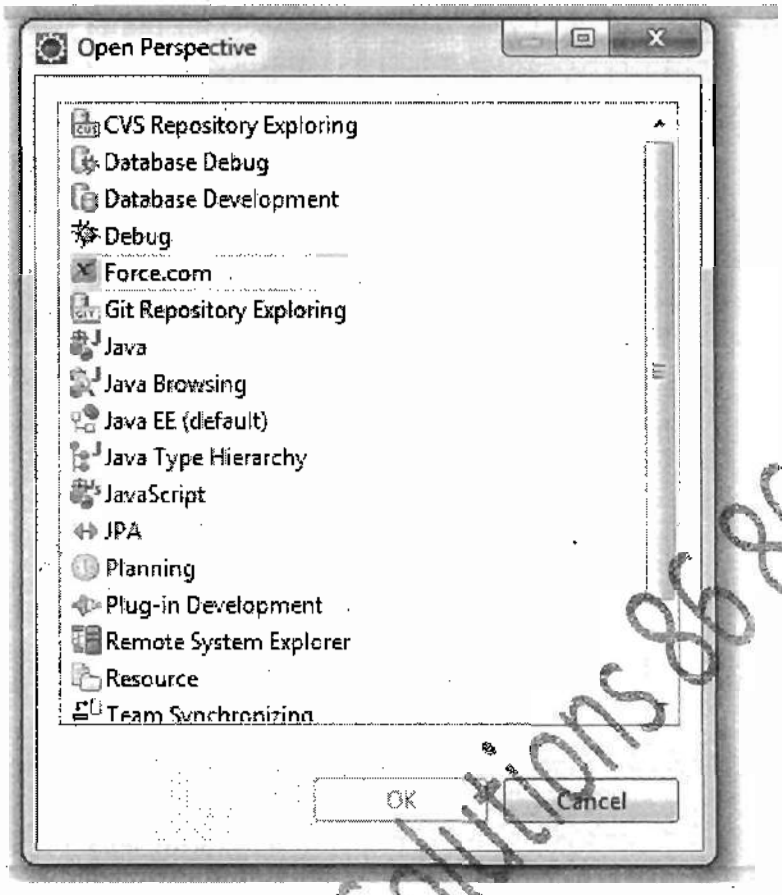








Change the proppective to Force.com



Create a new Force.com Project:

File→New→Force.com Project

Enter The Project Deatilss Like

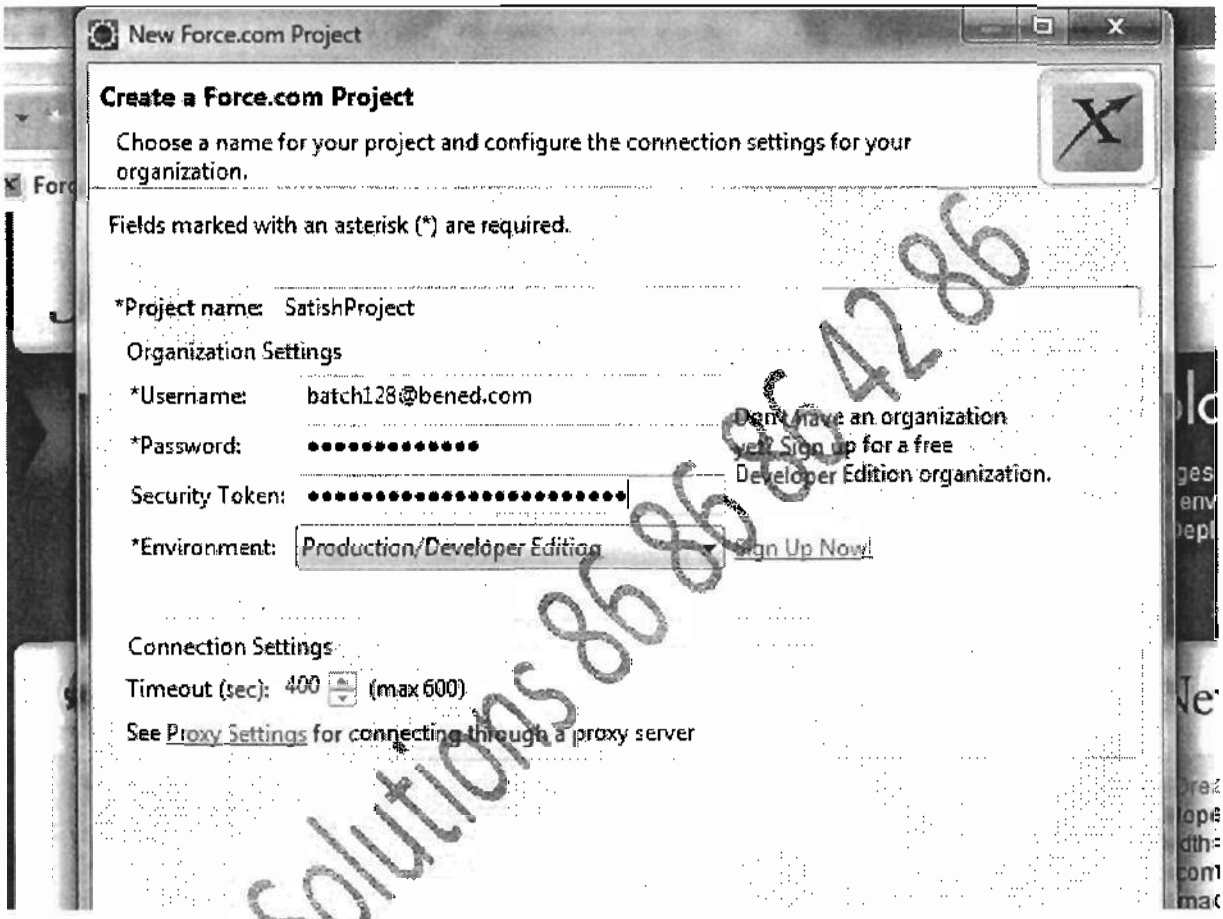
1.Project Name :

2.UserName : This is your salesforce username

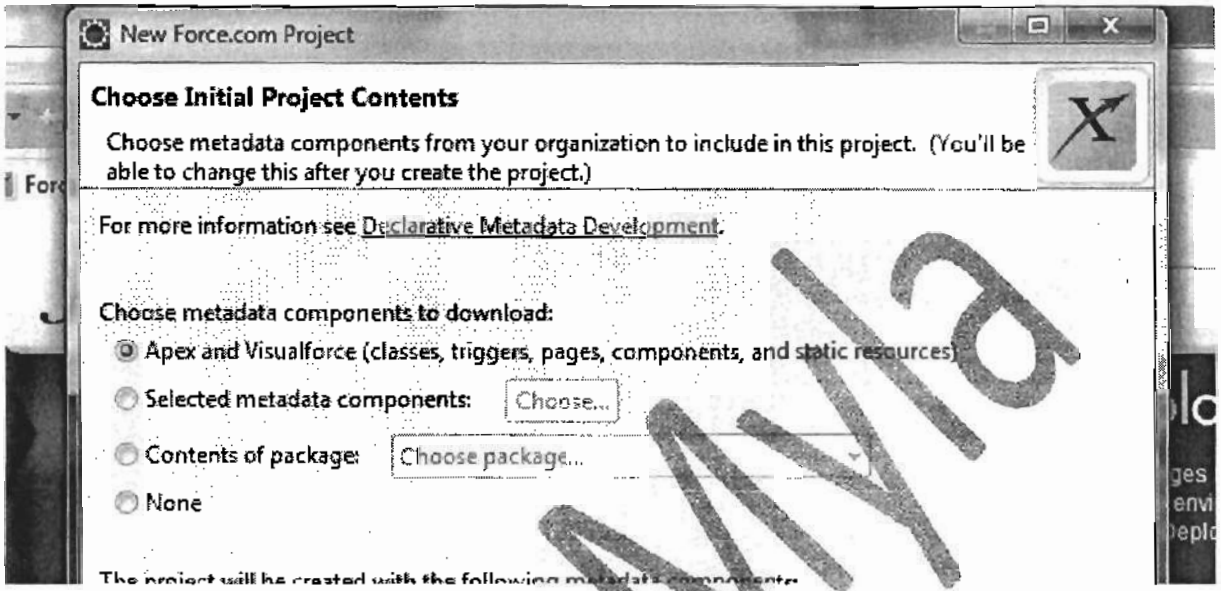
3.Password : This is salesforce password

4.Secuity Token : Reset Token from your salesforce account

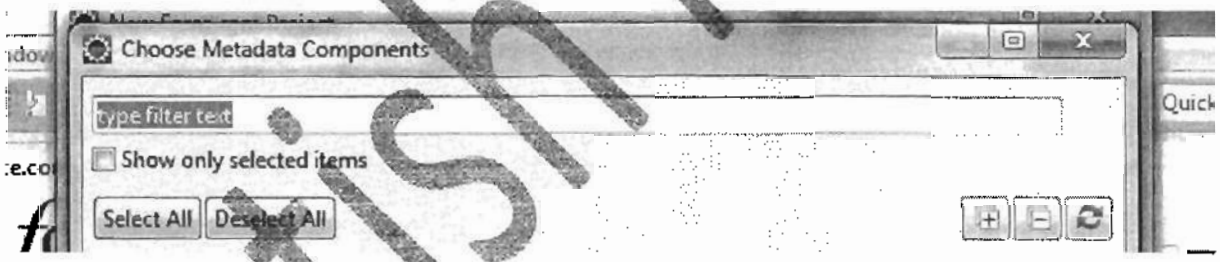
5.Select Environment :Production /Sandbox/Developer/PreRelease



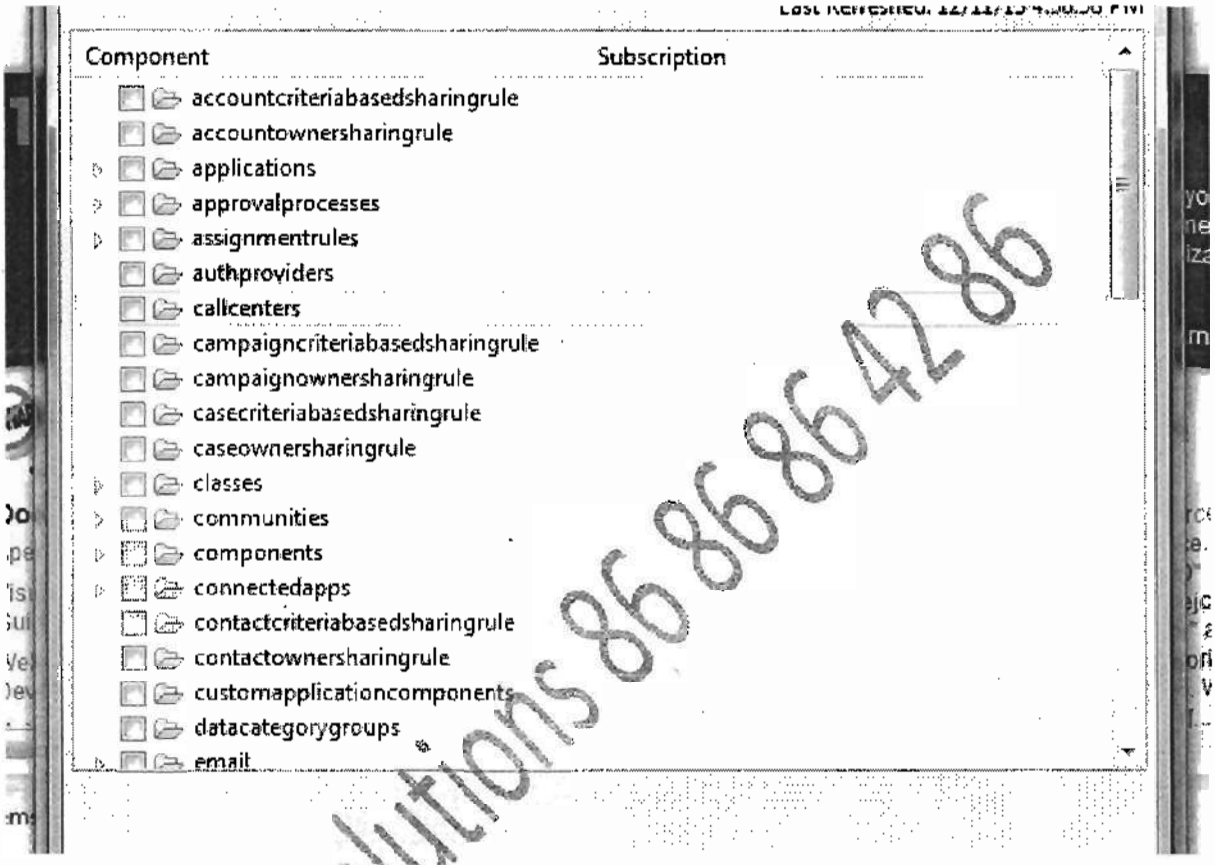
Choose which type project contents



Select Metadata components and click choose

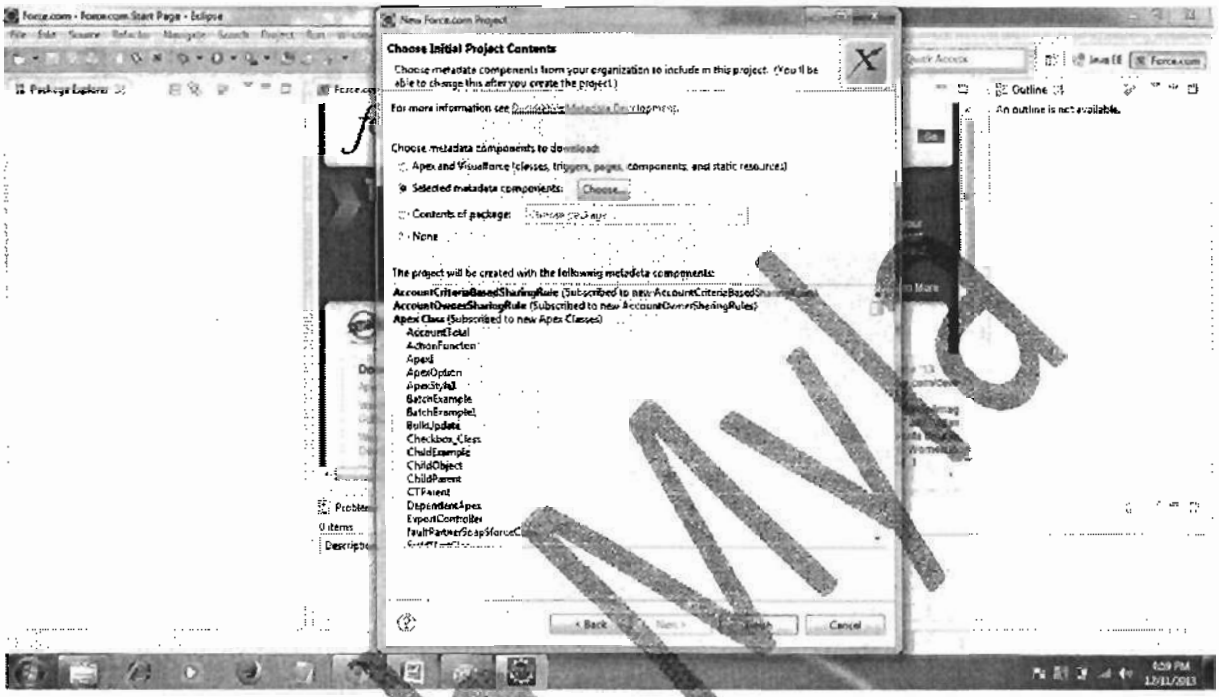


Click Select All button ;



Click finish button.

After



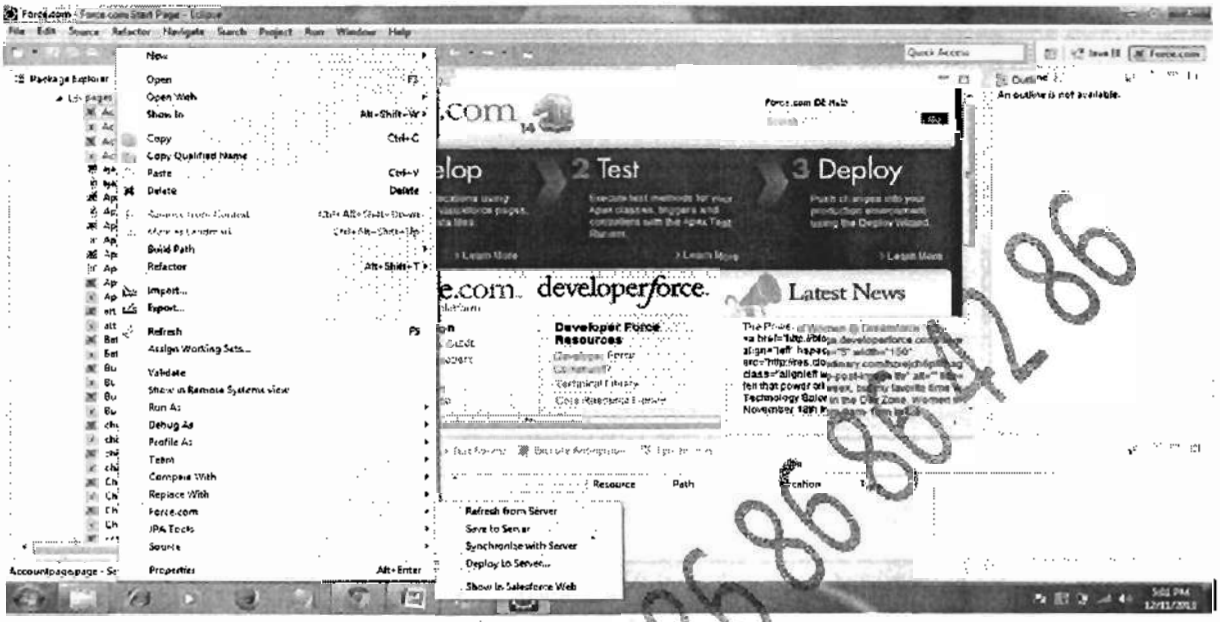
Steps to deploy from Eclipse to Salesforce Productions: .

Step 1: Select the project /component which you want to migrate from eclipse

Step 2 :Right Click on the component

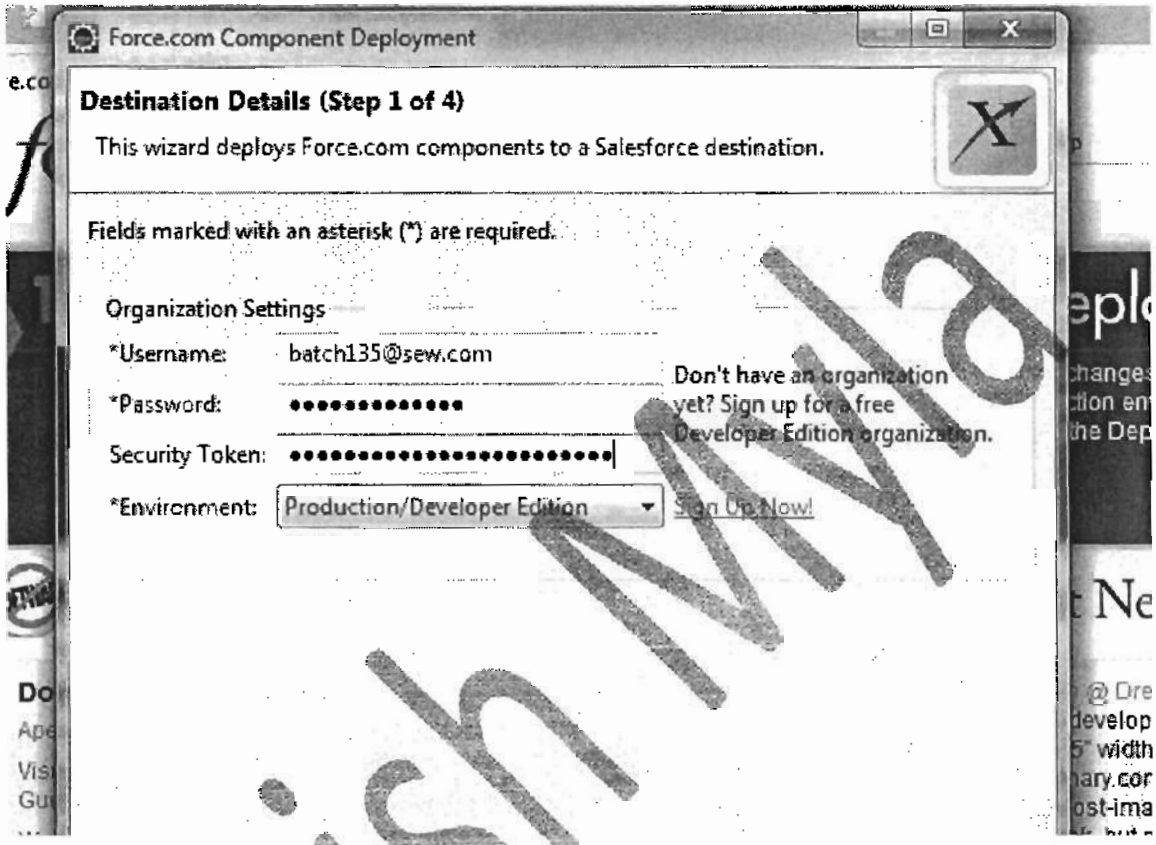
Step 3: select Force.com

Step 4: select Deploy to server

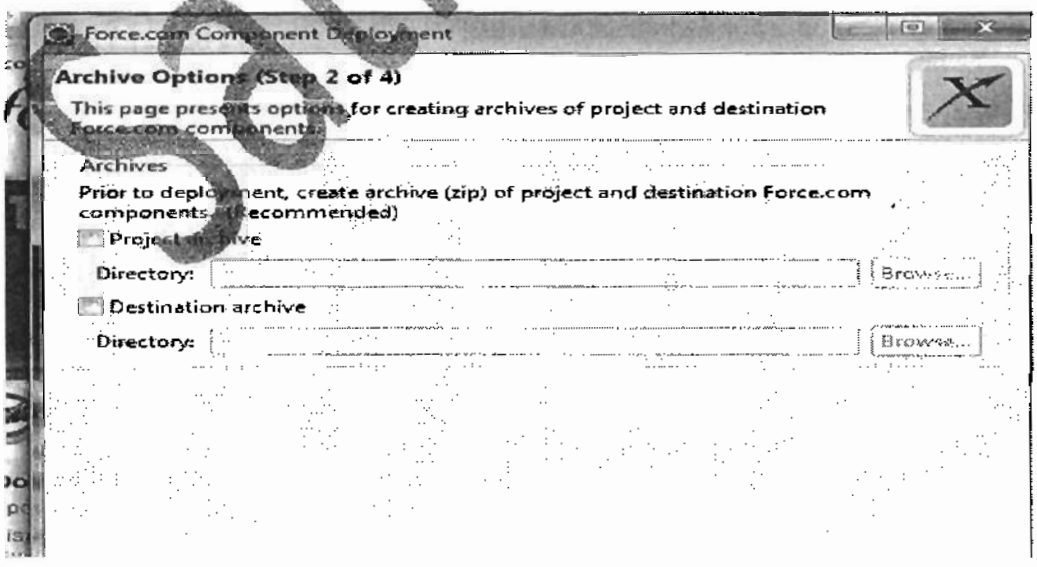


Enter the Production org details

1.User Name
2.Password
3.Security Token
4.Select the environment



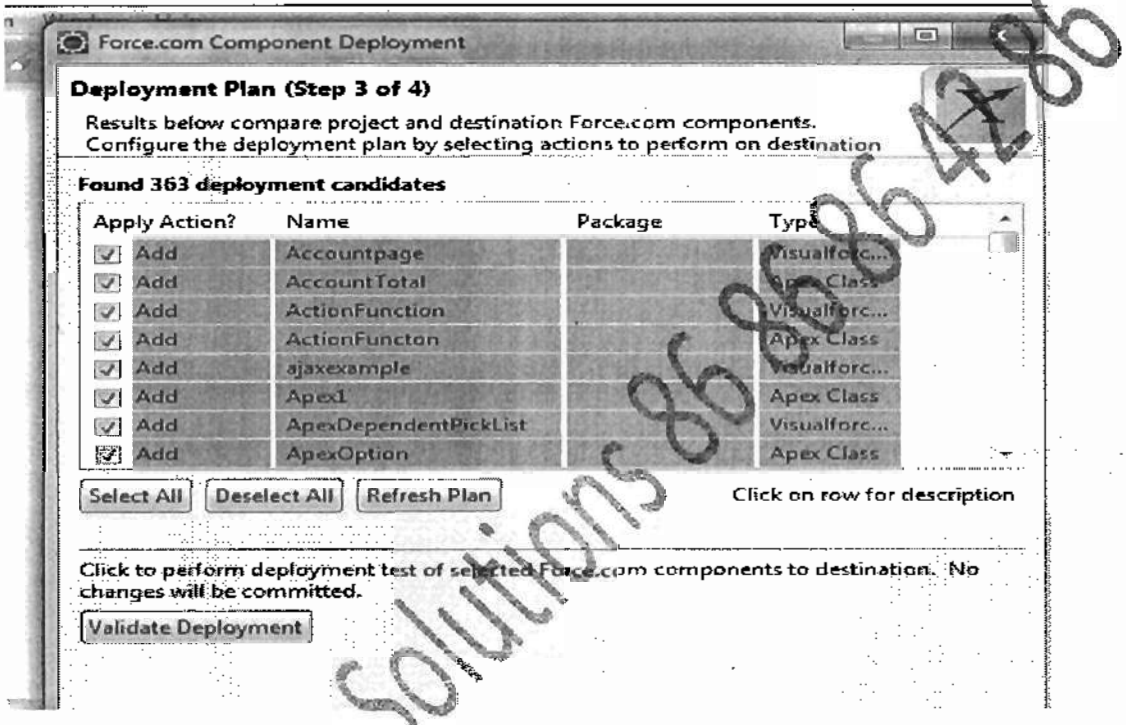
Uncheck the Destination Archive checkbox and click next

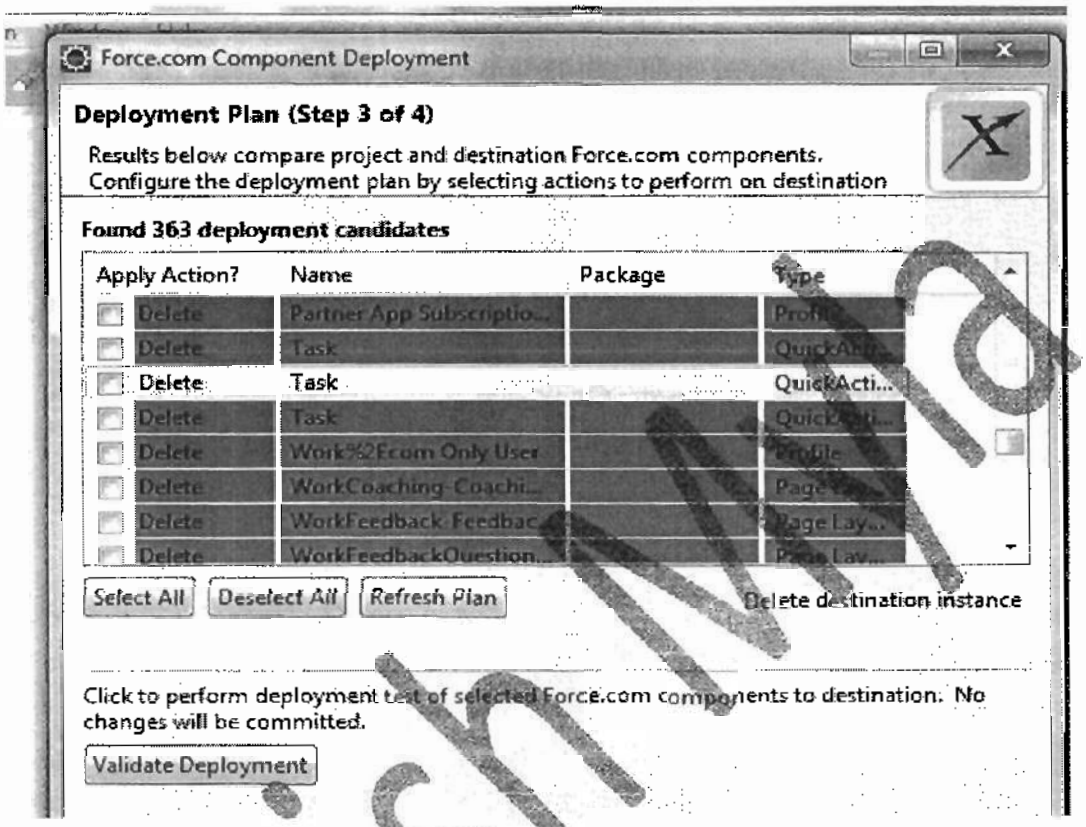


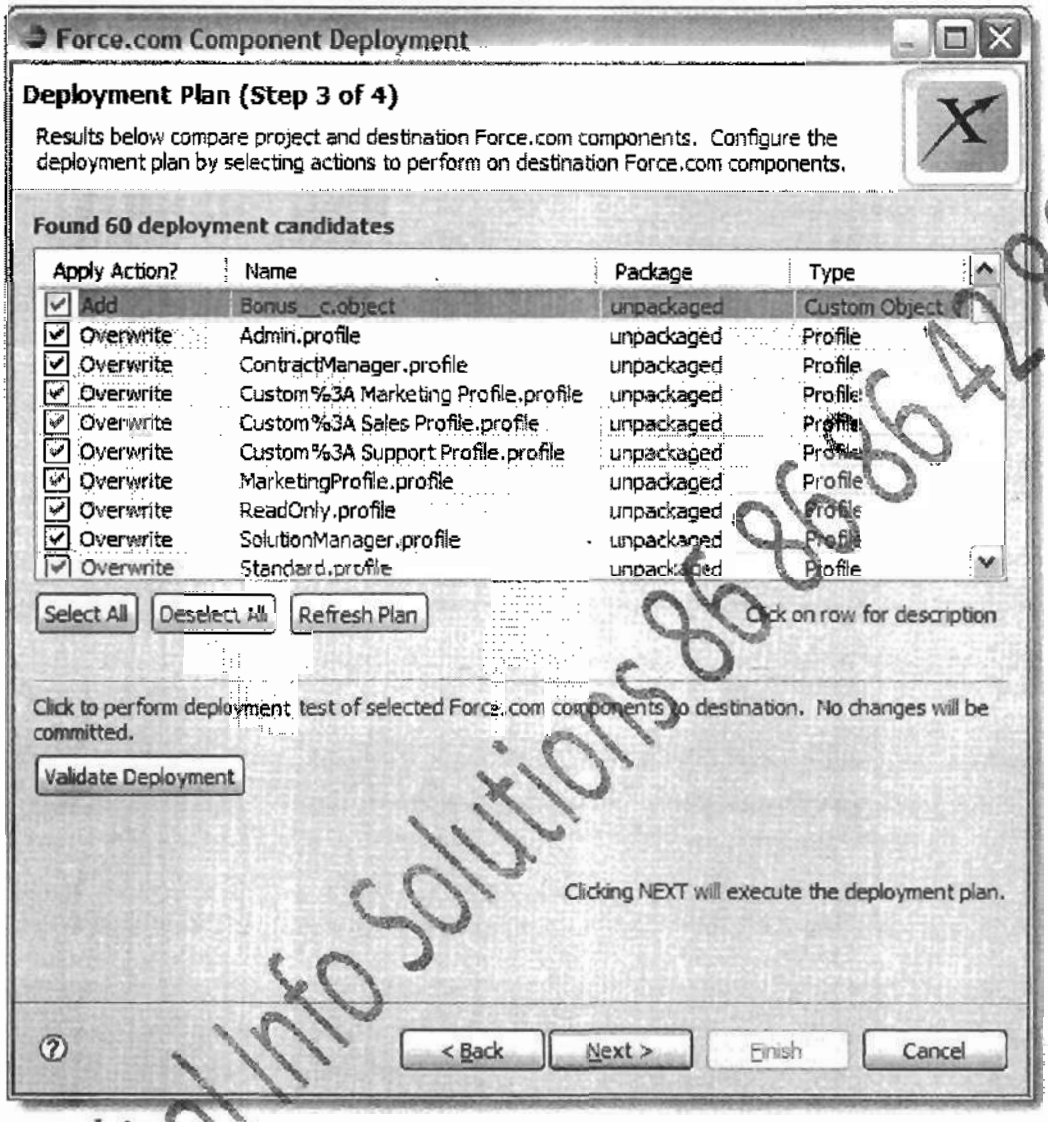
Step : Green color indicates that those components are not there in the destination environment.

Yellow Color : This dicates that those components are already available in the destination environment.

Red Color : This indicates that we cannot deploy these components .

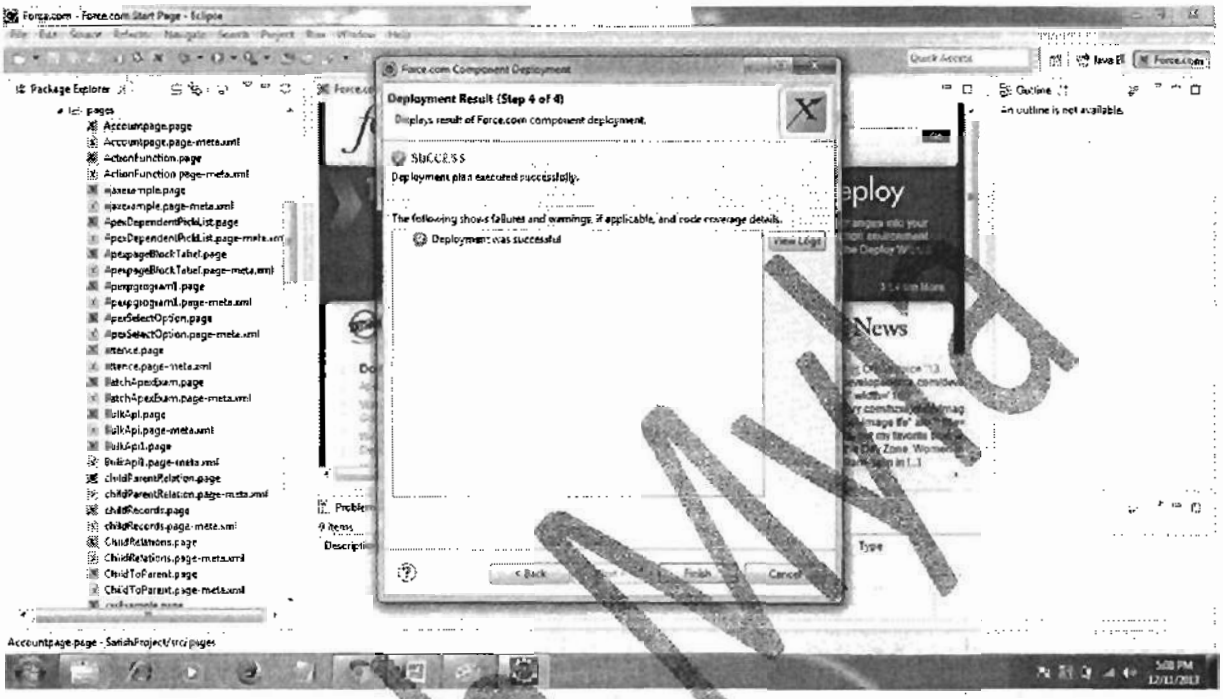



Force.com Component Deployment
Deployment Plan (Step 3 of 4)
Results below compare project and destination Force.com components.
Configure the deployment plan by selecting actions to perform on destination
Found 363 deployment candidates
Apply Action?
Name
Package
Type
Delete
Partner App Subscriptio...
Task
Work%2Ecom Only User
Profile
WorkCoaching-Coachi...
WorkFeedback-Feedbac...
WorkFeedbackQuestion...
Select All
Deselect All
Refresh Plan
Delete destination instance
Click to perform deployment test of selected Force.com components to destination. No changes will be committed.
Validate Deployment



Capital info SOLUTIONS
Training • Consulting • Development

salesforce.com PARTNER



Shyamala Plaza, Opp. Annapurna Block Behind Huda Maitrivanam Ameerpet,Hyderabad-16
040-66028688, +91 86 86 86 42 86.
www.capitalinfosol.com capitalinfosol@gmai.com

How to create Outbound and Inbound changeset in Salesforce?

Migration is the act of moving configuration changes from one Salesforce organization to another. During the development cycle, you might migrate many times, either to keep development organizations in sync or to move changes through development organizations toward production.
Migration can happen in two ways: manually or through the Metadata API.

- Manual migration—Changes to components that are not available in the Metadata API must be manually migrated in each environment. That is, you must repeat the exact same modifications in every production or development organization. Manual migration is made easier by diligently tracking changes.

- Metadata migration—Components that are available in the Metadata API can be migrated using desktop tools or change sets.

The typical steps to migrate changes are:

1. Determine which components need to be migrated first. This is especially important if you have both manual and metadata migrations to perform. For example, if you need to create a queue against a custom object, the custom object must be migrated first.

2. Migrate components in the order required:
   - Look at your change list to determine if anything needs to be manually migrated. If so, log into the organization you will be migrating changes into and make those setup changes manually.
   - Retrieve the latest metadata changes from the server.

3. Optionally modify your Force.com project or outbound change set to deploy only a subset of components.

4. Deploy.

Manulal Deployment is Done through change sets

What is Change Set?

Change Set is the deployment tool by which Sales force Developer/Administrator can upload/deploy the changes to Sandbox (Test Environment) to Production. You can go in Setup>Deploy>Select any option of deployment. You can deploy changes in between Sandbox to Sandbox also.

There are three main topics and steps to understand first before deployment:

1. Outbound Change set:

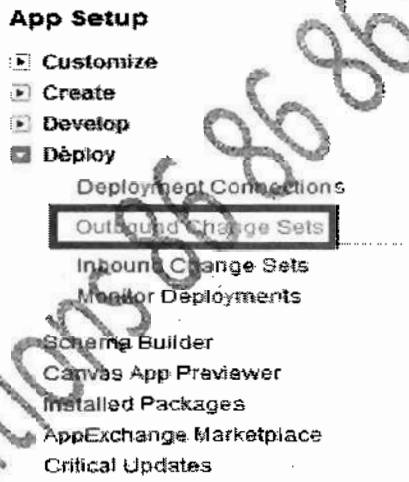
This is first step to the deployment through Change set. Outbound change set is nothing but creation of connection or select the changes you done in Sandbox like Object, Custom Fields, Validation, Workflow, Classes, Trigger etc. For that you have to follow below steps.

- Login in to Sandbox account of Salesforce.com.
- Go to Setup>Deploy>Outbound Change set: It will show you information on Change set and Outbound/Inbound change set information.
- Press Continue button on this page.
- Click on New button and create the outbound change set. Enter the Name of outbound change set and description of this and Click on Save.
- Once you get outbound change set detail page click Add button on Change Set Components.
- This page will show you Component Type (App, Analytical Snapshot, Apex Class, Apex Sharing Reason, Apex Trigger, Button or Link, Custom Fields, Custom Label, Object, Report Type, Custom Setting, Dashboard, Document, Email Template, Field Set, Folder, Home page Component etc.) Select any of above part and Click on Add To Change Set Button.
- After above step you will get the list of components added on change set component section.
- You can view or add dependencies in this section.
- Click on Add Profile in Profile Setting for included components means you can ass profile your can see or share the changes whatever you have done.
- After completing above steps click on Upload button. This will do the actual deployment to the Production/other Sandbox.
- Select any option from given list of Sandbox and Production.
- Click on Upload button to selected environment.

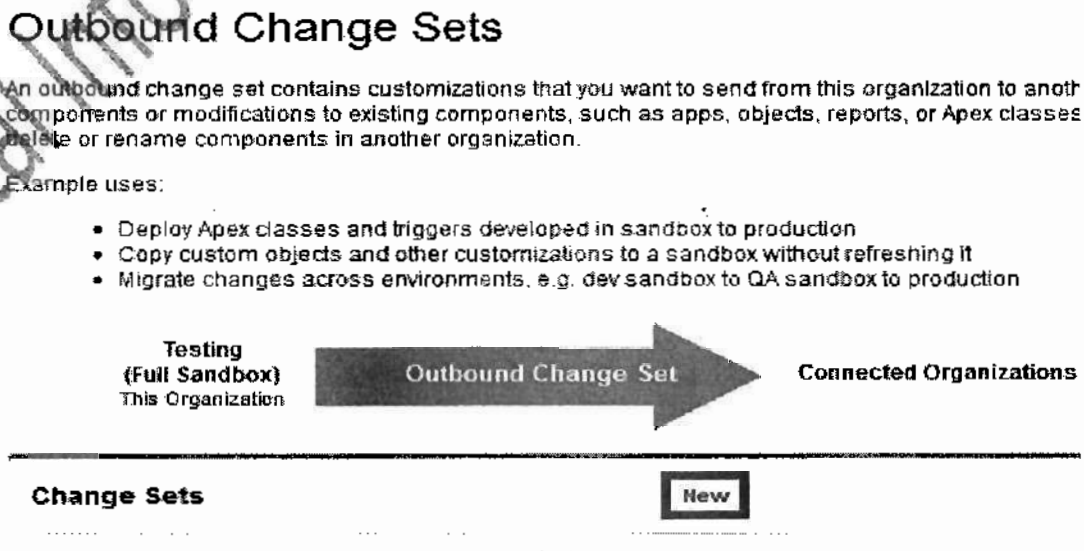
- After above step you will get Completion Email on your given email id. Means you have successfully uploaded the outbound change set.

How to create Outbound change set in Salesforce?

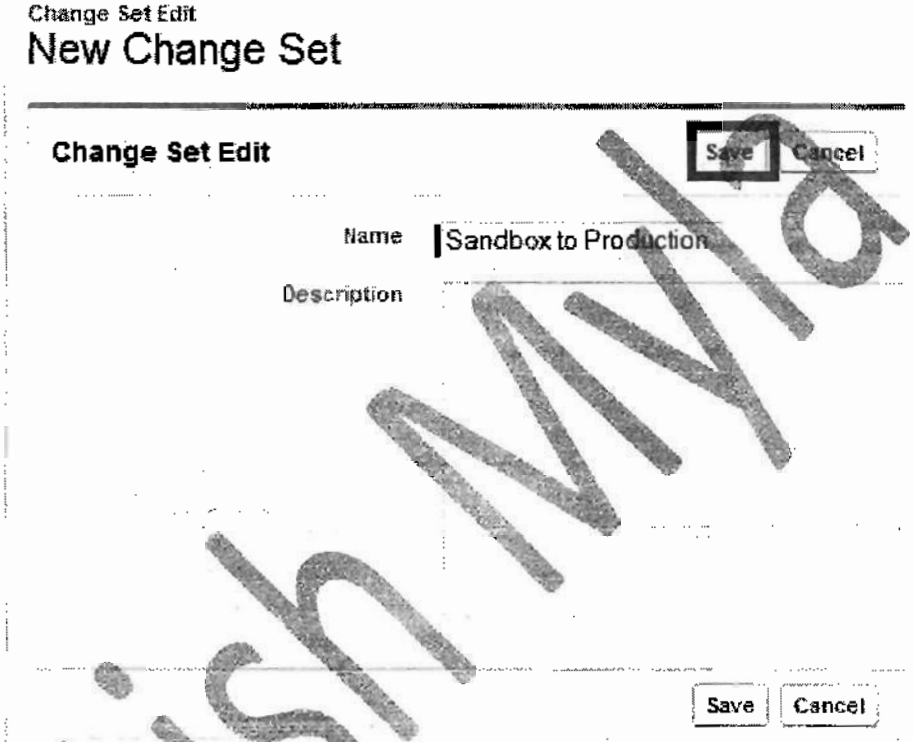
1. Go to Setup --> App Setup --> Deploy --> Outbound Change Sets.



2. Click "New" button.



3. Enter a Change Set Name and click "Save" button.



4 . Click on "Add" button in the Change set Components section to add the Conponents

Sandbox to Production

« Back to List: Outbound Change Sets

A change set contains customizations to components such as apps, objects, reports or emai another.

Once a change set has been uploaded, you can't add or remove components. However, you c

Change Set Detail Edit | Delete | Upload | Clone

Change Set Name: Sandbox to Production

Description:

Created By: Maguian Duraipandian, 12/7/2013 7:08 PM

Edit | Delete | Upload | Clone

Change Set Components Add | View/Add Dependencies

This change set contains no components

Add | View/Add Dependencies

Profile Settings For Included Components Add Profiles

This change set contains no profiles

5. Select a Component type, select the components and click "Add to Change set" button. Ex: if you have selected visualforce page as component Type .Then list of visualforce page will be displayed. Select the visualforce pages that you want to deploy.

Add to Change Set

Sandbox to Production

Choose components to add to your change set

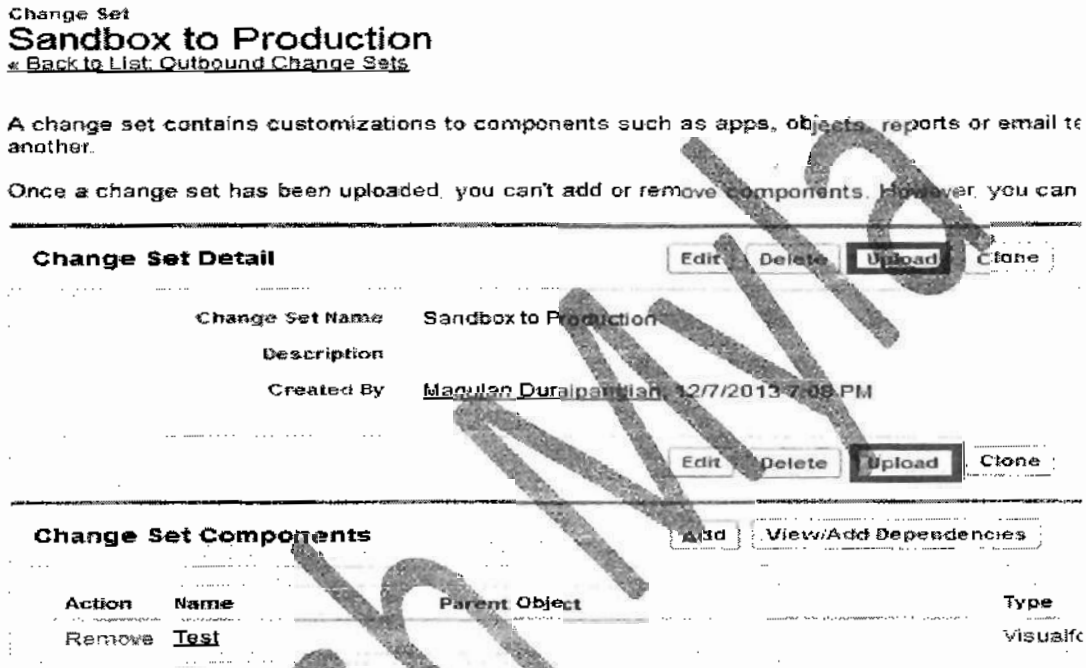
Add To Change Set | Cancel

Component Type: Visualforce Page

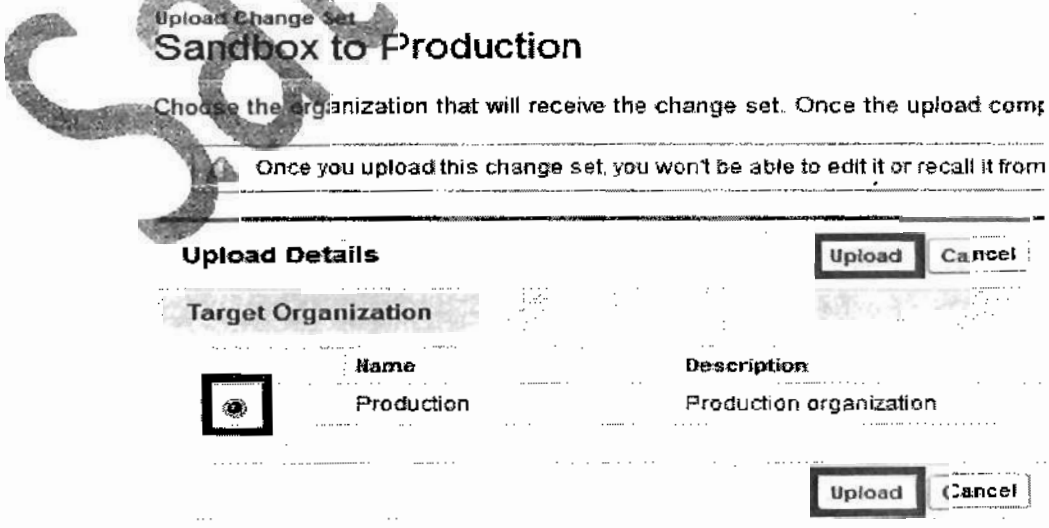
	Name ↑
☐	TeleCommHomePage
☑	Test

Add To Change Set | Cancel

6. Click "Upload" button.



7. Select the Target organization and click "Upload" button.



8.Once we click on the upload button in the above window ,Then outbound changeset what

We have created will be insitaed and confirmation will sent through email

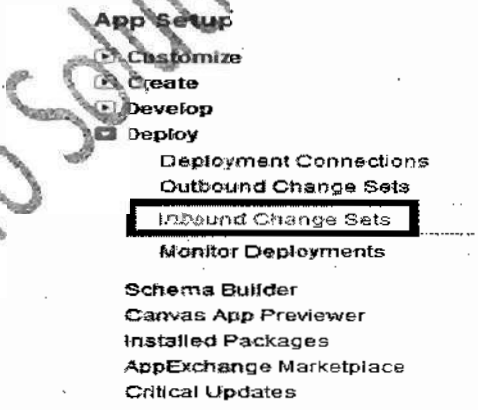
2. Inbound Change Set:

Inbound change set is automatically created once outbound change set it created inbound change set gets the all changes sent by outbound change set.

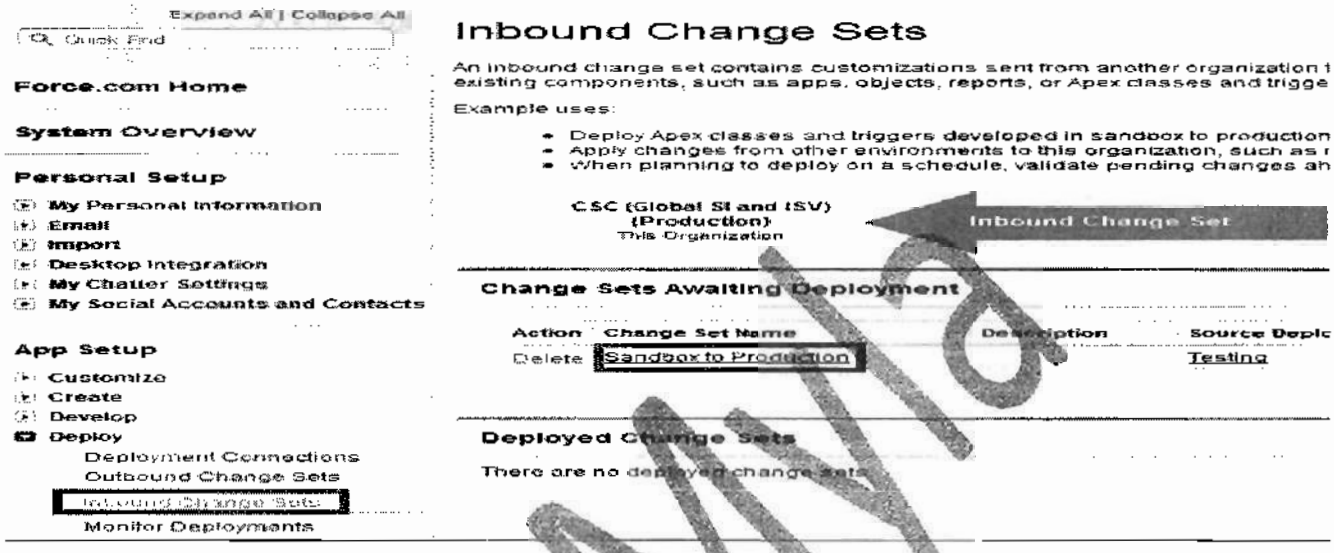
- Select on inbound change set and get detail of outbound.
- You can view change set components list and deployment history.
- Click on validate it will show validation history in deployment History.
- Click on Deployment after successful validation and can see Deployment History.

Steps to be followed in Production

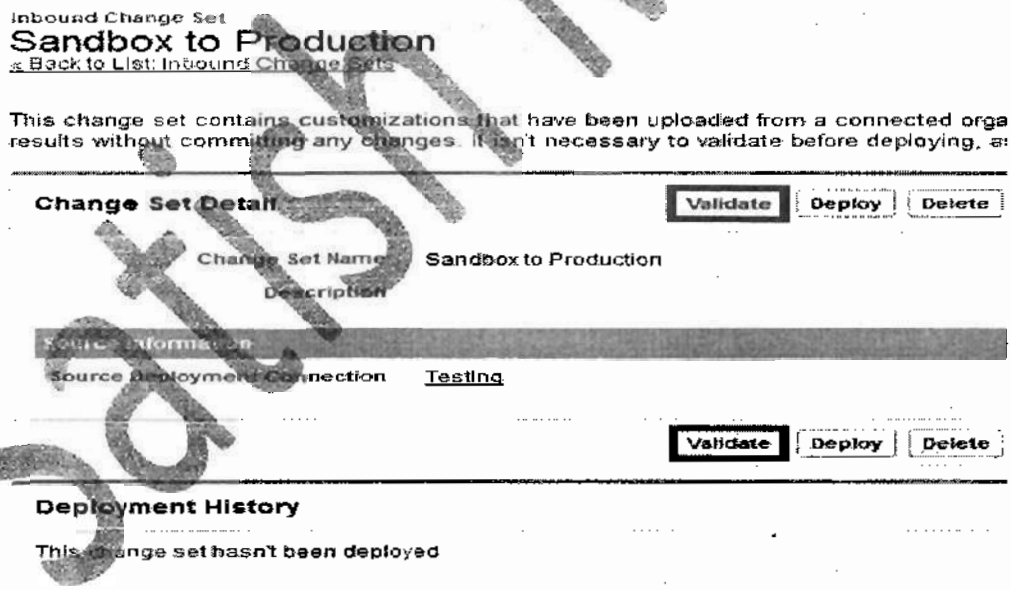
1. Go to Setup --> App Setup --> Deploy --> Inbound Messages.



2. Select the Change Sets Awaiting Deployment.



3. Click "Validate" button.



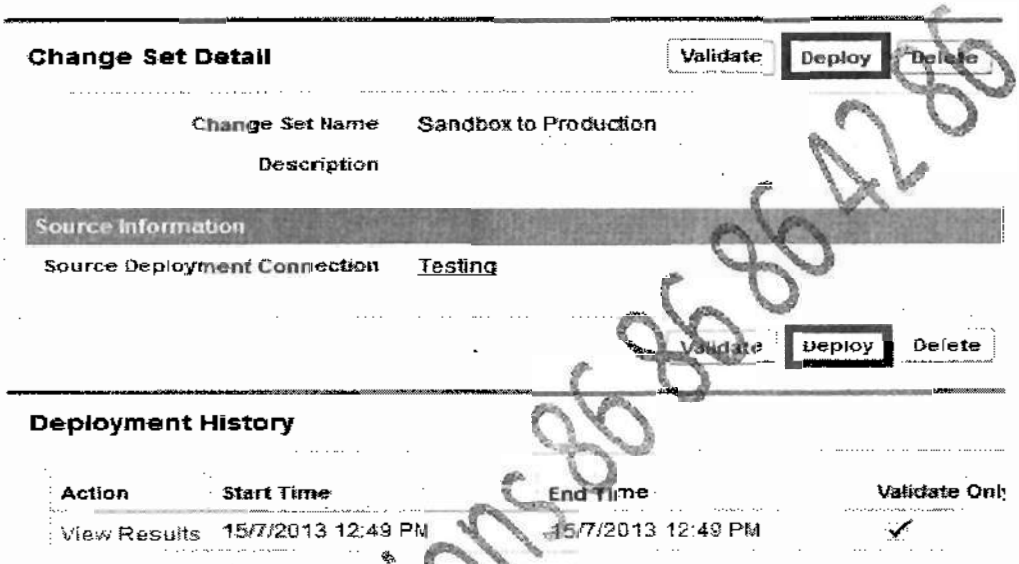
4. Click "Deploy" button.

Capital info SOLUTIONS
Training • Consulting • Development

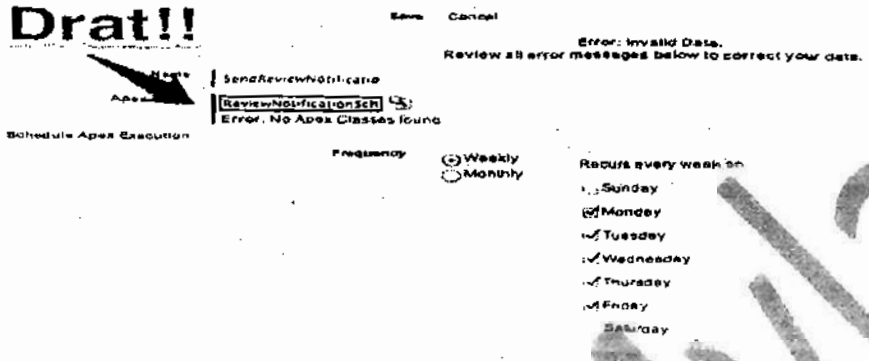
salesforce.com PARTNER

Inbound Change Set
Sandbox to Production
« Back to List: Inbound Change Sets

This change set contains customizations that have been uploaded from a connected orga results without committing any changes. It isn't necessary to validate before deploying, as



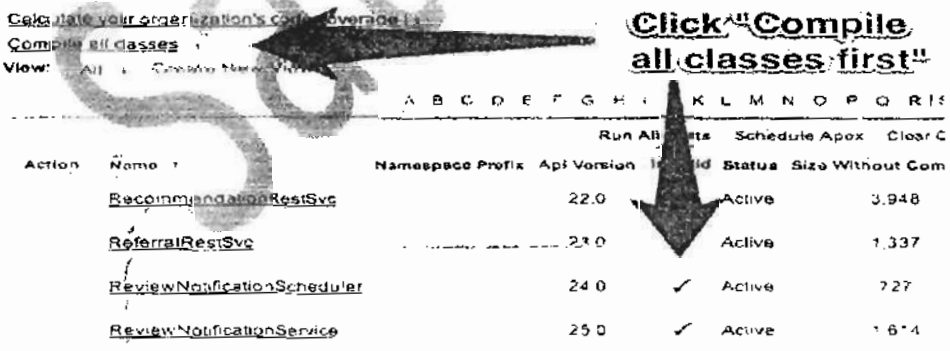
Shyamala Plaza, Opp. Annapurna Block Behind Huda Maitrivanam Ameerpet,Hyderabad-16
040-66028688, +91 86 86 86 42 86.
www.capitalinfosol.com capitalinfosol@gmai.com



I went and looked at the class and noticed that it was no longer valid:

Name ↑	Namespace Prefix	Api Version	Is Valid	Status
RecommendationRestSvc		22.0	✓	Active
ReferralRestSvc		23.0	✓	Active
ReviewNotificationScheduler		24.0	☐	Active
ReviewNotificationService		25.0	✓	Active

So here's how you fix the issue so that the class can be scheduled. You have to simply have to make salesforce recompile the classes and then it will be available to schedule.



Future Methods Real Time Issues Encountered:

Passing Objects to Future Annotated Methods

The future annotation is a great feature on the Salesforce Platform to allow you to execute custom logic asynchronously. I often use them to perform callouts within the context of a database trigger. However, one of the restrictions of future annotations is that you cannot pass sObjects or objects as arguments to the annotated method. I regularly use encapsulation to pass a collection of parameters, or even a sObject, but this won't work in @future method.

```
@future
static void doFutureCall(List<AddressHelper> addresses) {
 //do something
}
```

But thankfully there is a way you can do. The secret sauce is the JSON serialize|deserialize methods.

Take the example of an AddressHelper object referred to above. This is encapsulated convenience object to help me pass around address details. (note: For simplicity, this helper object just includes strings, but it could easily include other types including objects, sobjects, collections etc.)

```
/**
 * Sample encapsulated class
 * $author:  satish*/
public with sharing class AddressHelper {

        public String street {set; get;}
        public String city {set; get;}
        public String state {set; get;}
        public String zip {set; get;}

  public AddressHelper(String s, String c, String st, String z) {
        street = s;
        city = c;
        state = st;
        zip = z;
  }
}
```

Using the JSON serialize method I can easily pass this to an @future method.

```
public with sharing class AddressFuture
```

- Like any asynchronous method. The callout must be in a static method and return a void type.
- The system executes the asynchronous call when it has available resources – this means that you can't assume anything about when the call will be executed.
- The asynchronous method executes in a different transaction context than the trigger. A failure of the callout will have no effect on the transaction controlling the trigger (i.e. changes will not be rolled back). You will have to build in suitable compensation logic yourself to account for any exceptions in the callout.
- Keep the governor limits in mind when making callouts. We are constrained by the interface provided by the remote web service. If the remote web service were to provide any kind of batch processing interface, choose that instead of making individual calls. It helps avoid hitting the governor limits and will be more performant as well since there are fewer round trips to make.

Visualforce Basic Questions:

1. What are expressions used in pages to bind in controllers?

Using methods we can bind.
Getter: Will return value from controller to vf page
Setter: Will pass value from vf page to controller
Action: Will redirect to another page.

2. What is the purpose of controllers?

Controllers provide the data and actions that are available to a Visualforce page.

3. Which objects have associated standard controllers?

All standard and custom objects that can be accessed via the API have associated controllers

4. What is included with each standard controller?

Data: the fields for the associated object record that are API accessible, including the related records (5 up/1 down). Actions: save, delete, view, edit, cancel.

5. When do you need to go beyond a standard controller and code custom Apex?

If you need data beyond the limits of what is available in the standard controller or actions that go beyond the provided standard actions.

6. Compare and contrast custom controllers and controller extensions. How are they the same? How are they different?

Both allow for custom code to be used, allowing for custom data sets and custom actions. Extensions leverage the existing data and actions within a standard or custom

```
public AddressFuture ()
 {
        List<String> addresses = new List<String>();
        AddressHelper ah1 = new AddressHelper('1 here st', 'San Francisco', 'CA', '94105');
        AddressHelper ah2 = new AddressHelper('2 here st', 'San Francisco', 'CA', '94105');
        AddressHelper ah3 = new AddressHelper('3 here st', 'San Francisco', 'CA', '94105');
        //serialize my objects
        addresses.add(JSON.serialize(ah3));
        addresses.add(JSON.serialize(ah2));
        addresses.add(JSON.serialize(ah3));
        doFutureCall(addresses);
  }
}
```

And then, within my future method, all I need to do is deserialize the object, and you are good to go..

```
@future
 static void doFutureCall(List<String> addressesSer)
{
        AddressHelper currAddress = null;
        for (String ser : addressesSer)
         {
         currAddress = (AddressHelper) JSON.deserialize(ser, AddressHelper.class);
         System.debug('Deserialized in future:'+currAddress.street);
    }
 }
```

Limitations of future Annotation

1. No more than 10 method calls per Apex invocation
2. No more than 200 method calls per Salesforce license per 24 hours
3. The parameters specified must be primitive dataypes, arrays of primitive datatypes, or collections of primitive datatypes.
4. Methods with the future annotation cannot take sObjects or objects as arguments.
5. Methods with the future annotation cannot be used in Visualforce controllers in either getMethodName or setMethodName methods, nor in the constructor.

Points to remember:

controller. Custom controllers must contain all data and actions that need to be executed by the page. Extensions that extend standard controller allow for the pages which use those extensions to be used in custom buttons, standard button overrides, and over declarative features.

7. **What identifies a controller as being an extension?**
   The controller must declare a constructor which takes another controller explicitly. For example: public myControllerExtension(ApexPages.StandardController stdController) {this.acct = (Account)stdController.getRecord(); }

8. **Why are properties helpful in controllers?**
   Properties can automatically create standard getters and setters while still allowing for their customizations. They save you from both writing the tedious code and reading the clutter when reviewing code

9. **In what order do methods fire within a controller?**
   The only rule is that setters fire before action methods. Aside from that, there is no guaranteed order.

10. **What are some Apex classes that are commonly used within controllers?**
    StandardController, SelectOption, PageReference, Message, etc.

11. **How are wizard controllers different from other controllers?**
    The two main issues is that they must handle multiple pages and they must maintain the state across those pages.

12. **What are the effects of using the transient key word?**
    The transient key word prevents the data from being saved into the view state. This should be used for very temporary variables.

13. **When is a component controller required for custom components?**
    A component controller is required when business logic is required to decide how to render the component.

14. **What kind of content can be included in a Visualforce page?**
    Any content that can be rendered in a browser (HTML, JavaScript, etc.).

15. **What do {!*expressions*} refer to when used in Visualforce components?**
    Expressions refer to either data or actions that are made available to the page from the controller

Static resources are a new type of storage in Salesforce specifically designed for use in Visualforce pages. They are ways to store images, flash files, stylesheets, and other web resources on the Salesforce servers that can be cached for better page performance.

26. What are some examples of JavaScript Events?

Onmouseover, onclieck etc.

27. What is AJAX typically used for in Visualforce

AJAX is primarily used for partial page updates in Visualforce. In s-controls, the AJAX toolkit was the soap (XML over HTTP) client that gave you access to the force.com Web Services API.

28. What is the purpose of <script> tags?

Script tags allow you to create JavaScript (or other types) functions that can be used within your pages

29. What are the different AJAX action tags? What does each do?

- actionStatus: used to display start and stop statuses of AJAX requests.
- actionSupport: used to call a second component when an event happens to the first component.
- actionPoller: similar to actionSupport, but the event is based on a timer instead of a user action.
- actionFunction: provides support for invoking a controller action from JavaScript code using an AJAXrequest by defining a new JavaScript function.
- actionRegion: used to demarcate which parts of the page the server should reprocess.

30. How can you create partial page refreshes?

Basically, you need to define the section of the page that is going to refresh (typically with a panel of sorts), and then define the event that will cause the refresh. The method changes depending on if the area being refreshed is the same as the one handling the event. It also depends on if you are just processing something on the server, or if you need the UI to change.

31. Which tag is used with both radio buttons and picklists to create the selectable values?

<apex:selectOption>

32. What is the purpose of creating attributes on components?

By allowing the component to take parameters via an attribute, you can make the component more flexible and reusable

33.How can you access visualforce components values into a JavaScript?

16. **What are the ways that Visualpages can be incorporated into the rest of your user interfaces?**
Basically, via links, buttons, tabs, and inline frames.

17. **Is it always necessary to know Apex to create Visualforce pages? When does it become necessary?**
No, it is not always necessary. You can use standard controllers and VF component tags to accomplish quite a bit. Apex becomes necessary when you need either a custom set of data or custom actions to be available from the page.

18. **What are attributes? What is the syntax for including them?**
Attributes are modifiers to the main tag that belong after the tag name in the start tag. The syntax is attributeName="attributeValue"

19. **What are three types of bindings used in Visualforce? What does each refer to?**
Data bindings refer to the data set in the controller.
Action bindings refer to action methods in the controller.
Component bindings refer to other Visualforce components

20. **What is the main difference between using dataTable vs. pageBlockTable tags?**
PageBlock: For default salesforce standard format.
dataTable:To design customformats

21. **Which tag is used with both radio buttons and picklists to create the selectable values?**
<Apex:selectoption> tag

22. **How many controllers can a page have? Where is the controller for a page assigned?**
One main controller (of course, it could have extensions or custom components could have controllers, etc.). The controller is assigned in the <apex:page> tag.

23. **There are a series of layout components that all help recreate the traditional Salesforce page layout style very easily. What name do they share?**
pageBlock.

24. **Which tags should be used to automatically bring in the Salesforce label and default widget for a field?**
Pageblock

25. **What are static resources?**

Using **Component** global variable, we can access visualforce components in javascript. Let us suppose, we want to use id of an apex field with id="afield".
So, we can use the {!$Component.afield} syntax to use properties of the field in javascript.
Let us suppose, we want to store the field's value in java script, then we can use like below:

```
<script>
        Var a =' document.getElementById('{!$component.afield}').value';
</script
```

34. What are the Gov Limits in Salesforce.com?

Because Apex runs in a multitenant environment, the Apex runtime engine strictly enforces a number of limits to ensure that
Runaway scripts do not monopolize shared resources. These limits, or *governors*, track and enforce the statistics outlined in the following table. If a script ever exceeds a limit, the associated governor issues a runtime exception that cannot be handled.
Governor limits can be extended from release to release.

	Trigger	Class	Test
Total number of SOQL's	20	100	100
Total number of SOSL's	0	20	20
Total number of records Retrieved by single SOQL Query	1000	10000	500
Total number of records Retrieved by single SOSL Query	0	200	200
Total Number Of Call out Methods	10	10	10
Total number of send email methods allowed	10	10	10
Total Heap Size	300000 Bytes	3MB	1.5MB

35. What is Multitenant Architecture?
Ans: - An application model in which all users and apps share a single, Common infrastructure And code base

36. What is Metadata-driven development model?
Ans: - An app development model that allows apps to be defined as Declarative "blueprints," With no code required. Data models, objects, forms, workflows, and more are defined by Metadata.

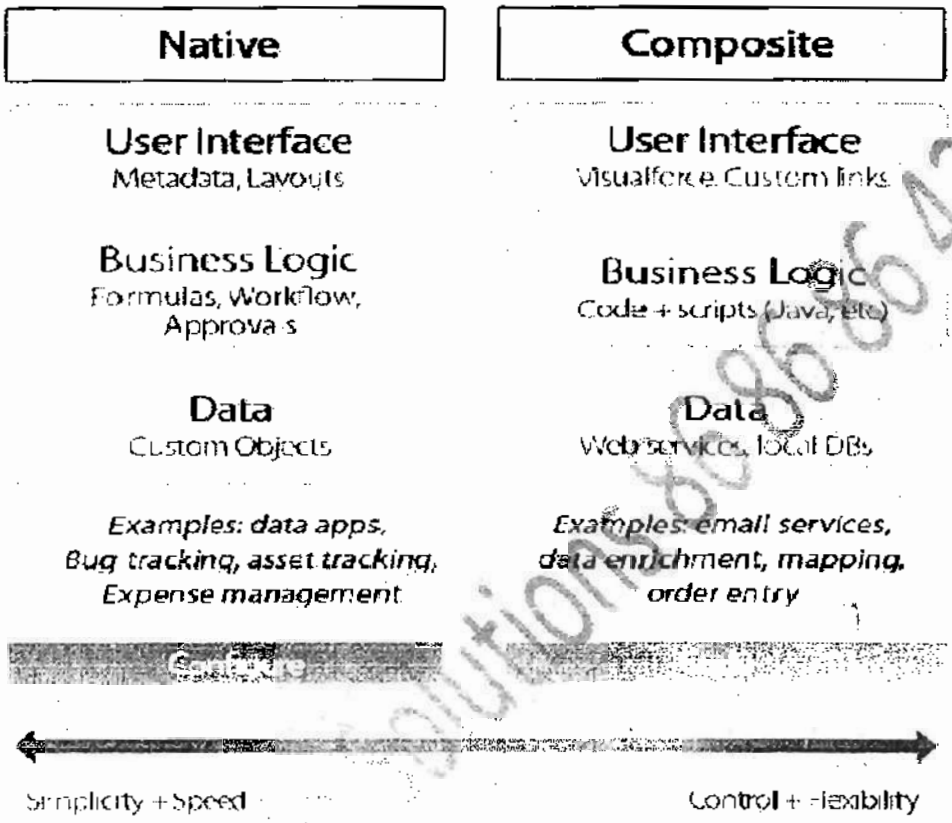
37.What is Apex ?
• Apex is a procedural scripting language in discrete and executed by the Force.com platform.
• It runs natively on the Salesforce servers, making it more powerful and faster than non-server code,such as JavaScript/AJAX.
• It uses syntax that looks like Java
• Apex can written in triggers that act like database stored procedures.
• Apex allows developers to attach business logic to the record save process.
• It has built-in support for unit test creation and execution.

38. What are Force Platform sites?
Ans: - Public websites and applications that are directly integrated with your Salesforce organization - without requiring users to log in with a username and password

39. What is AppExchange directory?
Ans: - A Web directory where hundreds of AppExchange apps are available to Salesforce customers to review, demo, comment upon, and/or install. Developers can submit their apps for listing on the AppExchange directory if they want to share them with the community.

40. What is the difference between Native components and Composite Components?
Ans: - One way to split up the work is to look at which requirements can be implemented by using just the point-and-click tools of the platform and which requirements must be implemented by leveraging other Web applications. The former method, which uses native components, is typically fast and simple to use, requiring only point-and-click setup rather than more traditional coding. The latter method, which uses composite components, gives us more control and flexibility in what we do but requires more work.

The following diagram shows how features implemented with each method are created split out by their user interface layer, business logic, and data model.



41. Name some Native Components?

Ans:-•Custom objects
•Security and sharing rules
•Workflow and approval processes
•Custom reports and dashboards

42. What are Custom Objects?

Ans: - Custom objects are the native components that model the data we need to store in our application. Similar to a database table, a custom object is composed of several fields that store information such as a job applicant's name, or the maximum salary for a particular position. However, unlike traditional database tables, we don't need to write any SQL in order to create custom objects. We can simply point and click in the platform to create as many objects as we need.

43. What is a database table?

Ans: - A database table stores information about a single type of person, thing, or concept—such as a job position. In the Force Platform, we use the term object here

44. What is a database row or record?
Ans: - A database row, or record in Force Platform terms, represents a single instance of an object—such as the SW Engineer position.

45. What is a field?
Ans: - A field stores a particular piece of information on a record.

46. What are Relationships?
Ans: - Relationships define the connection between two objects, and objects are related to each other through the use of common fields.

47. When do you Roll-up Summary fields?
Ans:-Use roll-up summary fields to display the sum, minimum, or maximum Value of a field in a related list, or the record count of all records listed in a related list.

48. What are the different types of Picklists and what is the difference between the two?
Ans:-Picklists come in two flavors: a standard picklist, in whom a user can select only one option, and a multi-select picklist, in which a user can select multiple options at a time. For the purposes of our Position object, we need to define standard picklists for a position's location, status, type of job, functional area, and job level.

49. What are field dependencies?
Ans:-Field dependencies are filters that allow us to change the contents of a picklist based on the value of another field. For example, rather than displaying every value for Country in a single picklist, we can limit the values that are displayed based on a value for another field, like Continent. That way our users can find the appropriate country more quickly and easily.

50. What is the difference between controlling and dependent fields ?
Ans:-Picklist fields can be either controlling or dependent fields. A controlling field controls the available values in one or more corresponding dependent fields. A dependent field displays values based on the value selected in its corresponding controlling field. For example, the Continent picklist is the controlling field, while the Country picklist is the dependent field.

51. What are custom formula fields ?
Ans:-Just as you can use a spreadsheet program like Microsoft Excel to define calculations and metrics specific to your business, we can use custom formula fields to define calculations and metrics that are specific to the application

52. What are Validation rules?
Ans:-Validation rules verify that the data a user enters in the application meets the standards that you specify. If it doesn't, the validation rule prevents the record from being saved, and the user sees an error message that you define either next to the problematic field or at the top of the edit page.

53. What are Page Layouts?

Ans:-A page layout controls the position and organization of the fields and related lists that are visible to users when viewing a record. Page layouts also help us control the visibility and editability of the fields on a record. We can set fields as read-only or hidden, and we can also control which fields require users to enter a value and which don't. Page layouts are powerful tools for creating a good experience for our users, but it's crucial that we remember one important rule: page layouts should never be used to restrict access to sensitive data that a user shouldn't view or edit.

54. What are the different types of relationship fields ? What are the differences between them?

Ans:- There are different types of relationship fields, each with different implications. The simplest and most flexible type is a lookup relationship field, which creates a simple relationship between two objects.

A second type of relationship field, master-detail relationship, is a bit more complex, but more powerful. Master-detail relationships create a special parent-child relationship between objects: the object on which you create the master-detail relationship field is the child or "detail," and the object referenced in the field is the parent or "master." In a master-detail relationship, the ownership and sharing of detail records are determined by the master record, and when you delete the master record, all of its detail records are automatically deleted along with it. Master-detail relationship fields are always required on detail records, and once you set a master-detail relationship field's value, you cannot change it.

55. When do you use master-detail relationship?

Ans:-If you have an object that derives its significance from another object. For example, say you have a Review custom object that contains an interviewer's feedback on a job application. If you delete a job application record, you will probably want all of its review records deleted as well, being that reviews of something that nolonger exists aren't very useful. In this case, you want to create a master-detail relationship on the Review custom object with the Job Application object as the master object.

56. What are search layouts?

Ans:-Search layouts are ordered groups of fields that are displayed when a record is presented in a particular context, such as in search results, a lookup dialog, or a related list. By adding fields, we can give users more information and help them locate records more quickly

57. What is a Junction Object?

Ans:-A junction object is a custom object with two master-detail relationships, and is the key to making a many-to-many relationship.

58. What is the difference between Object-Level Security , Field-Level Security and Record-level Security ?

Ans:-**Object-Level Security** The bluntest way that we can control data is by preventing a user from seeing, creating, editing, and/or deleting any instance of a particular type of object, like a position or review. Object-level access allows us to hide whole tabs and objects from particular users, so that they don't even know that type of data exists. On the platform, we set object-level access rules with object permissions on user profiles.

Field-Level Security A variation on object-level access is field-level access, in which a user can be prevented from seeing, editing, and/or deleting the value for a particular field on an object. Field-level access allows us to hide sensitive information like the maximum salary for a position or a candidate's social security number without having to hide the whole object.On the platform, we set field-level access rules with the field-level security.

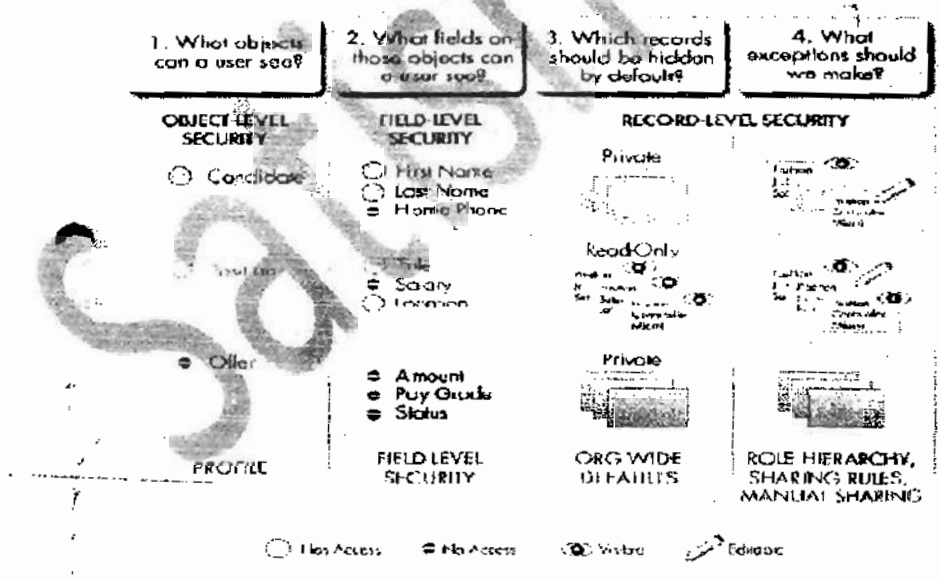
Record-Level Security To control data with a little more finesse, we can allow particular users to view an object, but then restrict the individual object records that they're allowed to see. For example, record-level access allows an interviewer like Melissa Lee to see and edit her own reviews, without exposing the reviews of everyone else on her team. On the platform, we actually have four ways of setting record-level access rules:

• Organization-wide defaults allow us to specify the baseline level of access that a user has in your organization. For example, we can make it so that any user can see any record of a particular object to which their user profile gives them access, but so that they'll need extra permissions to actually edit one.

• Role hierarchies allow us to make sure that a manager will always have access to the same records as his or her subordinates.

• Sharing rules allow us to make automatic exceptions to organization-wide defaults for particular groups of users.

• Manual sharing allows record owners to give read and edit permissions to folks who might not have access to the record any other way.



59. What are Organization-wide defaults ?

Ans:- Organization-wide defaults allow us to specify the baseline level of access that a user has in your organization. For example, we can make it so that any user can see any record of a particular object to which their user profile gives them access, but so that they'll need extrapermissions to actually edit one.

60. What are Role Hierarchies ?

Ans: - Role hierarchies allow us to make sure that a manager will always have access to the same records as his or her subordinates. The first way that we can share access to records is by defining a role hierarchy. Similar to an org chart, a role hierarchy represents a level of data access that a user or group of users needs. Users assigned to roles near the top of the hierarchy (normally the CEO, executives, and other management) get to access the data of all the users who fall directly below them in the hierarchy. The role hierarchy ensures that a manager will always have access to the same data as his or her employees, regardless of the org-wide default settings. Role hierarchies also helpfully define groups of users who tend to need access to the same types of records.

61. What are sharing rules ?

Ans:- Sharing rules allow us to make automatic exceptions to organization-wide defaults for particular groups of users. Sharing rules let us make automatic exceptions to org-wide defaults for particular groups of users The thing to remember with sharing rules is that, like role hierarchies, we can use them only to open up record access to more users. Sharing rules and role hierarchies can never be stricter than our org-wide default settings.

62. What is Manual Sharing ?

Ans:- Manual sharing allows record owners to give read and edit permissions to folks who might not have access to the record any other way.

63. What are Profiles?

Ans:-A profile is a collection of settings and permissions that determine what a user can do in the platform, kind of like a group in a Windows network, where all of the members of the group have the same folder permissions and access to the same software. Profiles control:

- The objects the user can view, create, edit, and delete
- The object fields the user can view and edit (more on that later!)
- The tabs the user can view in the app
- The standard and custom apps the user can access
- The page layouts a user sees
- The record types available to the user
- The hours during which the user can log in to the app
- The IP addresses from which the user can log in to the app

Profiles are typically defined by a user's job function (for example, system administrator or sales representative), but you can have profiles for anything that makes sense for your organization. A profile can be assigned to many users, but a user can be assigned to only one profile at a time.

64. What are the differences between Roles and Profiles ?

Ans:-Profiles control a user's object- and field-level access permissions. Indeed, a user can't be defined without being assigned to a particular profile, since the profiles specifies the apps and tabs that appear when he or she logs in, among a number of other useful things.Roles, on the other hand, primarily control a user's record-level access permissions through role hierarchy and sharing rules. Although a role assignment isn't exactly required when we define a user, it would be foolish of us not to assign a role since it makes it so much easier to define our record-level permissions.Because profiles control object- and field-level access whereas roles influence

record-level access, a user is typically assigned to one of each. To help you remember which controls what, remember: Roles control Records.

65. What is a Public Group?

Ans:-A public group is a collection of individual users, other groups, individual roles, and/or roles with their subordinates that all have a function in common. Using a public group when defining a sharing rule makes the rule easier to create and, more important, easier to understand later, especially if it's one of many sharing rules that you're trying to maintain in a large organization. You'll need to create a public group if you ever want to define a sharing rule that encompasses more than one or two groups or roles, or any individual.

66. What is a workflow?

Ans:-Workflow is a Force Platform business-logic engine that allows us to automatically send email alerts, assign tasks, or update field values based on rules that we define. Any time that changes to a record meet the conditions in a workflow rule, the platform automatically performs any actions associated with the rule.

67. What are Workflow Rules ?

Ans:-In general, a workflow rule is the main container for a set of workflow instructions. It includes the criteria for when the workflow should be activated, as well as the particular tasks, alerts, and field updates that should take place when the criteria for that rule are met.
Every workflow rule must be based on a single object you choose when you define the rule. This object influences the fields that are available for setting workflow activation criteria.

68. What are Workflow Tasks?

Ans:- A workflow task assigns a task to a user according to a particular template. Just as in Microsoft Outlook, tasks include information about something that needs to be done by a certain time, such as making a telephone call or returning a library book. Assigned tasks appear in a user's My Tasks related list on their Home tab and generate reminder messages that pop up when a user logs in. When we define a workflow task, we provide default values for the Assignee, Subject, and Status, Priority, and Due Date fields for tasks that are generated by an associated workflow rule. We can also make sure that a notification email is sent to the assignee when a task is automatically generated.

69. What are Workflow Field Updates ?

Ans:- A workflow field update changes the value of a particular field on the record that initially triggered the workflow rule.

70. What are Workflow Email Alerts?

Ans:- workflow email alert sends an email according to a specified email template. Unlike workflow tasks, which can only be assigned to users of the app, workflow alerts can be sent to any user or contact, as long as they have a valid email address

71. What are Queues?

Ans:- Much like a collection of items in a lost and found drawer, a queue is a collection of records that don't have an owner. Users who have access to the queue can examine every record

that's in it and claim ownership of the ones they want. Queues are traditionally used in sales and support organizations to distribute new leads and support cases to the employees who have the most availability.

72. What are Time Dependent workflows Actions?

Ans:- Time-dependent workflow actions are actions that occur before or after a certain amount of time has elapsed We can use time-dependent workflow actions to fire tasks, field updates, and email alerts while the condition of a workflow rule remains true.

73. What are Email Templates?

Ans:- Email templates allow you to create form emails that communicate a standard message, such as a welcome letter to new employees or an acknowledgement that a customer service request has been received.

74. What is the role of Approval Processes?

Ans:- Approval processes allow you to specify a sequence of steps that are required to approve a new record. Each step allows one or more designated approvers to accept or reject a record. The steps can apply to all records included in the process, or just to records that meet certain requirements. Like workflow, approval processes also allow you to specify actions—like sending an email alert, updating a field value, or assigning a task—that should occur whenever a record is approved, rejected, first submitted for approval, or recalled.

75. What are Approval Actions?

Ans:- Just like workflow actions, approval actions allow you to create and assign tasks, update fields, and send email updates and outbound messages. They can either be associated with the approval process as a whole or with individual approval steps.

76. What are the different types of Reports that Platform supports ? When do you need to use them?

Ans:- The platform supports three different report formats, each with varying degrees of functionality and complexity:

• Tabular reports are the simplest and fastest way to look at your data. Similar to a spreadsheet, they consist simply of an ordered set of fields in columns, with each matching record listed in a row. While easy to set up, they can't be used to create groups of data or graphs. Consequently, they're best used just for tasks such as generating a mailing list.

Tip: Use tabular reports when you want a simple list or a list of items with a grand total.

Summary reports are similar to tabular reports, except that they also allow you to group rows of data, view subtotals, and create graphs. For example, in the sample Employee Interviewer reports that appear in the following screenshot, the summary report groups the rows of reviews by the possible values of the Owner Name field, allowing us to see at a glance subtotals of how many times the two interviewers have talked to candidates and entered reviews for them.

While a little more time-consuming to set up, summary reports give us many more options for manipulating and organizing the data, and, unlike tabular reports, they can be used in dashboards. Summary reports are the workhorses of reporting - you'll find that most of your reports tend to be of this format.

Tip: Use summary reports when you want subtotals based on the value of a particular field or when you want to create a hierarchical list, such as sales organized by year and then by quarter.
Matrix reports are the most complex kind of report available, allowing you to group records both by row and by column. For example, in the following sample Employee Interviewer reports, the matrix report groups the review rows by the possible values of the Owner Name field, and also breaks out the possible values of the Position field into columns. Consequently, the report gives us subtotals for the number of times an interviewer has interviewed candidates and entered reviews for a particular position. These reports are the most time-consuming to set up, but they also provide the most detailed view of our data. Like summary reports, matrix reports can be used in dashboards.
Tip: Use matrix reports when you want to see data by two different dimensions that aren't related, such as date and product.

77. What are Dashboards ?
Ans:-A dashboard is a group of different summary or matrix report charts that graphically display custom report data.

78. What are the different Dashboard components ?
Ans:Components come in five varieties:
- **Charts**—Displays a pie chart, bar chart, line chart, or any other type of chart that can be made in a report.
- **Tables**—Displays a two-column table that contains record counts and values from the top-level row grouping in the report.
- **Metrics**—Inserts the grand total of a report at the end of a label that you customize.
- **Gauges**—Uses the grand total of a report as a point on a scale.
- **Custom S-Controls**—Displays any custom content that can be viewed in a Web browser, such as an ActiveX control, an Excel file, or a custom HTML Web form.

79. What are Custom Report Types ?
Ans:-Custom report types define the report criteria from which your users can run and create custom reports. When you create a custom report type, you specify the objects, relationships, and fields that users can select for their reports.

80. What are Web services ?
Ans:-A Web service is the mechanism by which two applications that run on different platforms, that were written in different languages, and that are geographically remote from each other, can exchange data using the Internet. Web services makes data exchange between two such applications as straightforward as the way at two processes can exchange data on a single computer. The way that data is exchanged between two Web services is similar to the way data is exchanged between a Web browser like Microsoft Internet Explorer and a Web server. Just as a Web browser uses a common network protocol (HTTP over TCIP/IP) to download HTML files hosted on a Web server, a Web service can also use this same network protocol to download data from another Web service. The key difference is the actual data that is sent and received—Web services use XML instead of HTML.

Ans:-A custom tab that allows your users to use external websites from within the application.

82. What is Workflow Outbound Message?
Ans:-A workflow action that sends data to an external Web service, such as another application in the cloud. Outbound messages are used primarily with composite apps.

83. Who is a running user?
Ans:-The user whose security settings determine what data is displayed in a dashboard. Because only one running user is specified per dashboard, everyone who can access the dashboard sees the same data, regardless of their personal security settings.

84. What is SOAP?
Ans:- A protocol that defines a uniform way of passing XML-encoded data. SOAP Stands for Simple Object Acess Protocol.

89. What is SOQL ?
Ans:-A query language that allows you to construct simple but powerful query strings and to specify the criteria that should be used to select the data from the platform database.
SOQL Stands for Salesforce Object Query Language

90. What is a Time Trigger?
Ans:-A setting that defines when time-dependent workflow actions should fire.

91. What is a Primary Key?
Ans:-A relational database concept. Each table in a relational database has a field in which the data value uniquely identifies the record. This field is called the primary key. The relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

92. What is a foreign Key ?
Ans:-A field whose value is the same as the primary key of another table. You can think of a foreign key as a copy of a primary key from another table. A relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

93.What is Merge Field ?
Ans:-A field you can place in an email template, mail merge template, custom link, or formula to incorporate values from a record. For example, Dear {!Contact.FirstName}, uses a contact merge field to obtain the value of a contact record's First Name field to address an email recipient by his or her first name.

94. What is a tab ?
Ans:-An interface item that allows you to navigate around an app. A tab serves as the starting point for viewing, editing, and entering information for a particular object. When you click a tab at the top of the page, the corresponding tab home page for that object appears.

95. What is S-Control?

Ans: S-Controls are the predominant salesforce.com widgets which are completely based on JavaScript. These are hosted by salesforce but executed at client side. S-Controls are superseded by Visualforce now.

96. Will Visual force still supports the merge field's usage like S-control?

Ans: Yes. Just like S-Controls, Visualforce Pages support embedded merge fields, like the {!$User.FirstName} used in the example.

97. What are Apex Governor Limits?

Ans: Governor Limits are runtime limits enforced by the Apex runtime engine. Because Apex runs in a shared, multitenant environment, the Apex runtime engine strictly enforces a number of limits to ensure that code does not monopolize shared resources. Types of limits that Apex enforces are resources like memory, database resources, number of script statements to avoid infinite loops, and number of records being processed. If code exceeds a limit, the associated governor issues a runtime exception.

98.What is static variable ?

Ans: This is an instance of a class .Which will be created only once and this is accessed using class name. this is a global scope