

No 1 Training Institute for Salesforce.com in India

Development Material (Apex, VisualForce, Deployments)

Completed 150+ Batches

300+ Students Placed

500+ Students Certified

13 Corporate Trainings in MNC's

80+ Workshops on Integrations & Webservices

***** 3 Months Internship in MNC's*****

on Salesforce.com

Trainer: Satish Myla

New Batches & Workshops starts on every Saturday

Contact for Demo: +91-7799458889

INDEX

<u>S.NO</u>	<u>TOPIC NAME</u>	<u>PAGE NO</u>
1.	When Should I Use Apex	1
2.	How Does Apex Work	2
	Object Oriented Programming (Oops)	
3.	Data Types	3
4.	Class	5
5.	Access Modifiers	6
6.	Class Variables	8
7.	Object	9
8.	Constructors	10
9.	Usage of Apex Program With in VF Page	11
10.	Reffering to the Apex Class In VF Page	12
11.	Example for Getter Method	13
12.	Apex Class to Demonstrate Setter Method	14
13.	Call Apex Methods in A VF Page	15
14.	Simple Apex Class	16
15.	Array	17

16. Pick List Using Select Option from Apex	18
17. Collection	
• List	20
• Set	24
• Map	27
18. SOQL Queries	32
19. Child to Parent Relationship On Standard, Custom Objects	40
20. Example Program	41
21. Parent to Child Relationship On Standard, Custom Objects	43
22. SOSL Queries	46
23. Email Programming	49
• Outbound Email Service	50
• PDF File Attachment	54
• Inbound Email Service	58
24. DML Operations	62
25. Interface Iterator	67
26. Interface Iterable	70
27. Database.QueryLocator Class	72
28. Batch Apex	73
• Start Method	
• Execute Method	
• Finish Method	

29. Invoking Of Batch Apex Job	78
30. Order Of Execution Of Batch Apex Job	78
31. Database.Stateful	81
32. Governing Limits,Limitations	83
33. Apex Scheduler	84
34. Real Time use cases for Batch, Schedule Apex	89
34. Testing	100
35. Batch Apex Example With Test Case	106
36. Schedule Apex Example With Test Case	107
37. Triggers & Examples	109
38. Invokong Apex methods in Triggers	124
39.Recursive Triggers	127
40. Order of Execution Of Triggers	131
41. Future Annotation	139
42. Apex Sharing Rules	142
43. Flows Introducation	146
• Process.PlugIn Interface	
• Input parameters	
• Output Parameters	
44. Plug in Program	150
45. Select Option	154
46. Schema Programming	156
47. JSON(JavaScript Object Notation)	167
48. http Callouts	180
49. JQUERY	183

50. Introduction to VisualForce Page	189
51. Apex Components	190
• Apex Page	
• Formula & Expressions in VF	
• Apex PageBlock	
• Formula Expressions	
• PageBlock Section	
• PageBlock Buttons	
• Command Button	
• Command Link	
• PageBlock Section Item	
52. ApexInput Components	197
• Input Text	
• Input Secret	
• Input Hidden	
• Input Checkbox	
• Input Text Area	
• Select List	
• Select Option	
• Select Options	
• Select Radio	
• Input Field	
• Select CheckBox	
53. Java Script in VF Page	205
• Introduction to Java Script	
• Arrays in Java Script	
• Methods in Java Script	
• Reading Sobject data in Java Script	
• Reading VF input component data in Java Script	
• Validations in Java Script	
54. Page Block Table	217

55. Data Table	223
56. Data List	224
57. Apex Tab	226
58. Insert & Include	229
59. VF Page in PDF Format	235
60. Action function, Action Region, Action Support	238
61. Real Time Scenarios	240
62. Reports & Dashboards in VF	249
63. Google Maps in Salesforce	256
64. CSS	261
65. Remote Method invocation Java Script	270

66. Deployments

- Using Eclipse
- Using Change Set

67. Integrations

NOTE:- you can collect Integrations & Webservices hand book (65 pages) & work book of 20 pages from front desk. This covers all integrations examples (sap, SIEBEL, Net Suite, Java, .Net, Jira & many more).

Salesforce Integration 2 Days Workshop Batch Starts on every Saturday

1. Introduction to API's

- Meta Data API
- SOAP API
- REST API
- Chatter API
- Bulk API
- Streaming API

2. Meta Data API

- introduction Meta Data API
- Advanced Programs
- Deployments into different org's.

3. Introduction to Web Services

- Difference between Http 1.0 and Http 2.0
- Introduction to SOAP API
- Types of WSDL files and their purposes
- Creating Web Service classes in SFDC
- Consuming external WSDL file in Salesforce
- Salesforce to Salesforce integration using Enterprise WSDL
- Salesforce to Salesforce intergration using Partner WSDL
- Salesforce to Salesforce Integration using Apex WSDL
- Consuming Salesforce resource in java using Enterprise WSDL
- Consuming Salesforce resource in java using Partner WSDL
- Callouts using SOAP services from Apex
- Chatter Example using SOAP API

4. Introduction to HTTP

- 1.HttpRequest
- 2.HttpResponse
- 3.JSON Parsing
- 4.XML Parsing

5. Introduction to Rest API

- Creating a Rest methods in Salesforce
- Consuming Salesforce Rest Resource in Salesforce
- Introduction Authorization

- Authorization using API keys
- YouTube Integration with Salesforce using API Key
- Face book integration using Salesforce
- Difference between OAuth 1.0 and OAuth 2.0
- Authorization using the OAuth2.0 Web Server
- Authorization using the OAuth2.0 User-Agent
- Authorization using the OAuth2.0 Username –Password
- Authorization using the OAuth2.0 refresh token
- Calling Third-Party services using HTTP Callouts(Two examples)

6. I Google Maps and Google API's using HTTP callouts and using JavaScript
7. Single Sign On Integration using OKTA
8. Marketo Integration with salesforce
9. OnDemand integration with salesforce
10. Salesforce.com sites integration as Twitter/Facebook
11. Informatica Integration using Rest API
12. CAST IRON Integration
13. Outbound Messages
14. Chatter API
 - Introduction to Chatter API
 - Advanced program
15. JIRA CRM and Net suite CRM integrations Formats

Many More to

Interview Questions

Every Bened Student can answer these questions

1) Visualforce Basic Questions :

1. What is view state in visual force?
2. Which api used to design visual force page?
3. What is the difference between actionSupport and actionFunction
4. What is the actionRegion?
5. What is difference between insert and include?
6. How do you use static resource in VF page?
7. What is remote action?
8. How many records we can print a pageBlock
9. What is the difference between related List ,enhanced List ,detail
10. What is the difference between controller and extension?
11. What is Ajax? Have you used it?if so tell the scenario ?
12. What is Jquery ? Where you have used?
13. What is S-Controls?
14. What is the use Static Resource in Visual force?
15. Can I pass parameters from VF page to apex method?
16. How do you refer to current page id
17. Tell me something \$Action
18. How do you embed Google map in visual force?
19. How do you pass the parameters from page to page ?
20. What are custom components?
21. How do you make a VF page available for Salesforce1 .

2) Apex fundamentals

1. What is Apex?
2. What API is used in the apex?
3. What are the access modifiers in the apex?
4. What is the difference between With Sharing and Without Sharing?
5. What is a constructor?
6. What is the use of the static variables?

7. What are reference variables in apex?
8. What are Sobjects?
9. What is the difference between List and Set?
10. What is Map in apex?
11. Can we have duplicate Keys in Map
12. How many objects we can store in list?
13. What are setter and getter methods?
14. How do you refer to current page id in apex?
15. How to do you invoke standard actions in apex class?
16. What is page reference?
17. How do you pass the parameters from one apex class to another to another?
18. What is virtual class?
19. What is interface?
20. What is abstract class?
21. What is overloading?
22. What is overriding?
23. When we invoke with sharing method in without sharing class .Now method is Executed as?
24. Will the inner class inherits the sharing properties of outer class?
25. Base class is declared as With Sharing and Derived class is declared as without Sharing what will happen?
26. Can I have constructor with parameters in apex?
27. Dereferncing a Null pointer value error?
28. Variable is not available?
29. Too many Records: 10001

3) Batch Apex and Schedule apex questions.

1. What are the Soql limitations in apex?
2. What are transaction limits in apex?
3. What is the need of batch apex?
4. What is Database.Batchable interface?
5. Define the methods in Batchable interface?
6. What is purpose of Start method in batch apex?
7. What is the Database.QueryLocator?
8. What is the Iterable<Sobject>?
9. How to define the custom Iterable?
10. What is the use of execute method?
11. How many times execute method is called?
12. What is scope of execute method?

13. Can we call callouts from batch apex?
14. Can we call another batch apex from batch apex?
15. How many callouts we can call in batch apex?
16. If you get Callouts Governing limits error how do you rectify?
17. Batch is synchronous or Asynchronous operations?
18. How to synchronize the batch apex?
19. How do you call batch apex from the batch apex in earlier versions of API 26.0?
20. What all the general errors/ exception we will get in executing batch apex?
21. What is the maximum size of the batch and minimum size of the batch?
22. Tell some of the scenario's that you have developed using batch apex?
23. What is Database.BatchableContext ?
24. How to track the details of the current running Batch using BatchableContext?
25. How many batch jobs can be added to queue?
26. What is Database.Statefull interface
27. What is Database.AllowCallouts
28. Why should we call Database.execute() and future methods in Test.startTest() and Test.StopTest()
29. What is ASyncApexJob object?
30. When a BatchApexWorker record is created?

4) Schedule Apex :

1. What is Schedule apex?
2. How many ways we can schedule the batch apex?
3. What is Schedulable interface?
4. What is the order of execution?
5. How can we schedule the batch apex?
6. How many schedule jobs we can schedule at a time?
7. What is cronTrigger?
8. How to identify the jobs next schedule?
9. What is the difference between Synchronous and Asynchronous jobs
10. How many future calls we can make in a day?
11. What is the difference between manual schedule and apex schedule?
12. What is the best scenario that you have designed using schedule apex?
13. What are the problems that you have encountered while using schedule apex?

5) Triggers:

1. What is trigger?
2. What are different types of Triggers in sfdc?
3. What are Trigger Context variable?

4. What is the difference between Trigger.New and Trigger.NewMap
5. What is the difference between Trigger.New and Trigger.old
6. What is the difference between Trigger.New and Trigger.Old in update Triggers
7. Can we call the Batch apex from the trigger?
8. What are the problems you have encountered when calling batch apex from the trigger.
9. Can we call the callouts from triggers?
10. What are the problems that you encountered while calling apex callouts in triggers.
11. What is the recursive triggers?
12. What is bulkifying triggers?
13. What is the use of future methods in triggers?
14. What is the order of executing the trigger apex?
15. What is trigger handler?
16. How do you avoid recursive triggers?
17. How many triggers we can define on a object?
18. Can we define two triggers with same event on single object?
19. Tell me some scenarios' where you have written the triggers?
20. What is the best scenario that you have developed on triggers?

6. WebServices:

1. What is the difference between HTTP1.0 and HTTP 2.0
2. Difference between REST API and SOAP API?
3. What do you mean by statefull?
4. How many types of WSDL we have in salesforce?
5. What is the difference between Enterprise, Partner WSDL?
6. What is difference between Webservice and Callouts?
7. What is the use of Metadata API?
8. How to fetch data from another Salesforce instance using API?
9. What the annotations in salesforce?
10. In How many ways we get the session id using SOAP API?
11. Can we use Http callouts in SOAP API?
12. How to pass the session in REST API?
13. How many ways we can provide security in REST API?
14. What is diffenect between OAuth 1.0 and OAuth 2.0?
15. How many ways we can specify the Request Header in REST API to get accesstoken?
16. What types of response you have taken from the external server?
17. Can we call any resource from external server using url?
18. What is future annotation?
19. Did you work with heroku?
20. By default webservices are synchronous / asynchronous?



salesforce.com
PARTNER

21. Have you worked in Siebel to Salesforce data Migration?
22. How to implement Salesforce to Salesforce connection?
23. How to call Apex method from a Custom Button?
24. Have you worked in integrating Google chart?
25. Any idea on Salesforce API Integration Using SOAP-based Web Services?

7. Deployment and Change Sets:

1. What are the diffent types of changesets?
2. How to create a sandbox?
3. How many times we can refresh a(configuration /develope/full)
4. Which things we cannot pass using the change sets to production?
5. Can we move the record types using the change sets to production?
6. Can we move approval using change sets to production?
7. Can we move role /users/profiles using change set to production?
8. In how many ways we can deploy the code from sandbox to production?
9. What all the problems that you have when we are deploying?
10. What should you use command based deployment

Project Round:

APEX INTRODUCTION (1 - 16)

1. Introduction to Apex.
2. OOPS fundamentals.
3. Data types in Apex.
4. Access Modifiers.
5. Methods.
6. Apex classes.
7. Object creation.
8. Constructors.
9. Getters & setters methods.
10. Calling Apex members in visualforce.
11. visualforce with Apex Examples

Satish Anila

Apex is a strongly typed object oriented programming language.

→ It allows the developers to execute flows and transaction Control statements.

→ Apex enables developers to add business logic to most system events like button clicks related record updates and visualforce pages.

→ Apex language

i) Integrated :- It provides built in support for DML calls.

ii) Inline salesforce object query language :-

iii) Easy to use, etc

iv) Easy to test.

v) Version

vi) Multi tenant aware

When should i use Apex :-

→ To create email service.

→ Create webservices.

→ perform complex validation over multiple objects.

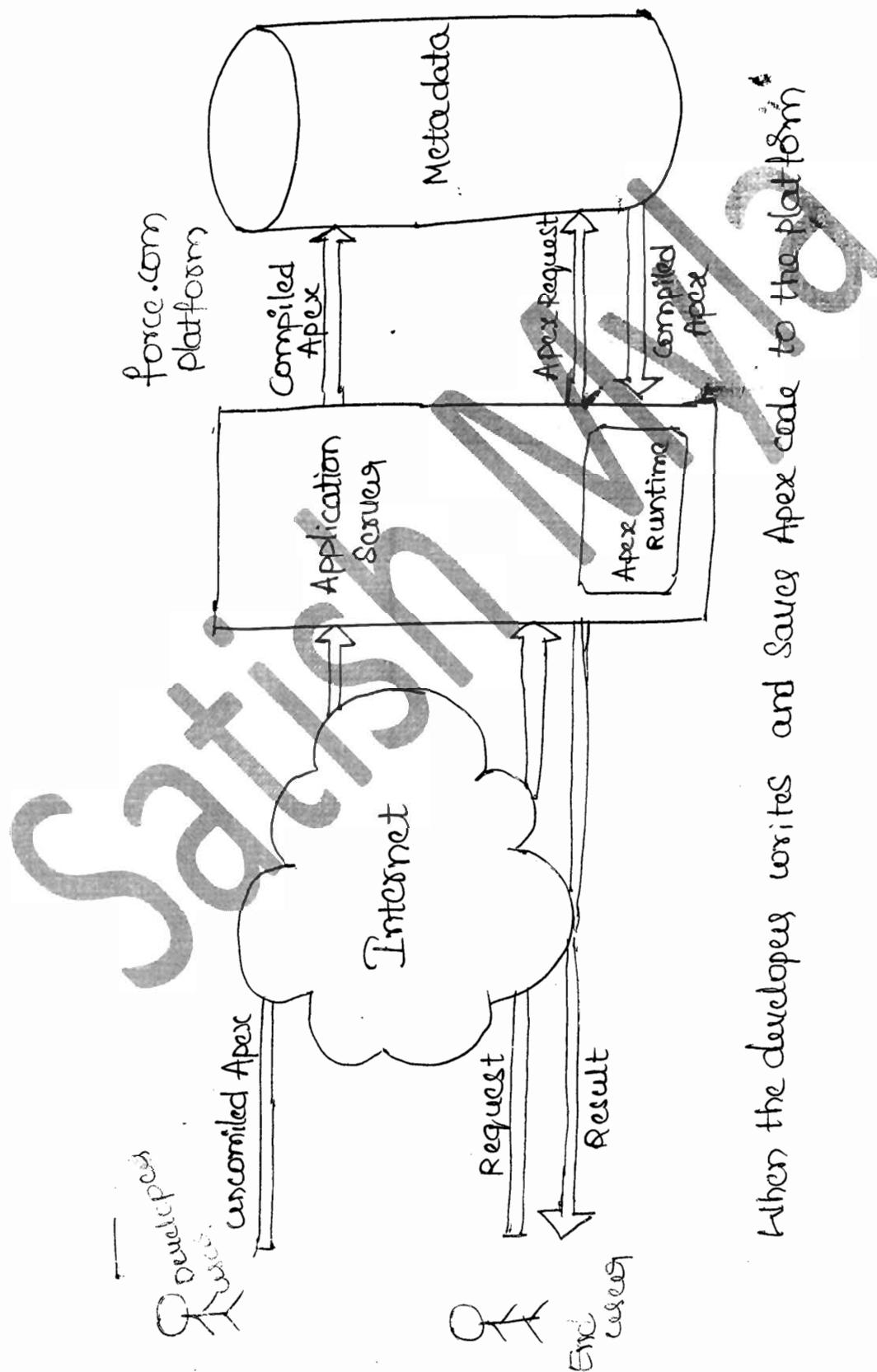
→ To create complex business processes that are not supported by work flow.

→ Create custom transaction logic.

→ Attach custom logic to another operation.

How does Apex Work :-

All Apex programs runs entirely ON-demand on Force.com platform.



(2)

A2

→ First the platform application server compiles the code into abstract set of instructions that can be understood by Apex runtime interpreter.

→ The compiled code is stored to metadata.

→ When the end user triggers the execution of Apex by clicking button of visualforce page the application server retrieves the compiled instructions from the metadata and send them to runtime interpreter before returning the result.

Object oriented programming (oops) :-

oop (object oriented programming) is a methodology that provides a way of modularizing a program by creating partitioned memory area for both data and methods that can be used as template for creating copies of such modules (objects) on demand.

unlike procedural programming, here in the oop programming model, programs are organised around objects and data rather than action and logic.

The main oops principles are

Encapsulation

Inheritance

Polymorphism.

Encapsulation :- the wrapping up of data and methods together is called encapsulation. for example, if we take a class, we write the variables and methods inside the class. Thus, class is binding them together. So class is an example for encapsulation.

Inheritance:- It creates new classes from existing classes, so that the new classes will acquire all the features of the existing classes is called Inheritance.

A good example for Inheritance in nature is parents producing the children and children inheriting the qualities of the parents.

Polymorphism:- Polymorphism represents one form in multiple forms. In programming, we can use a single variable to refer to objects of different types and thus, using that variable we call the methods of different objects. Thus a method call can perform different tasks depending on the type of the object.

Apex fundamentals :-

Data type :-

- Data type in the Apex tells about what type of data can be stored.
- What is the range of the data that can be stored.

- i) primitive data types
- ii) collections
- iii) Enums

i) primitive data types :-

These are the data types which are predefined by the Apex.

- A primitive data types such as an Integer, Double, Long, Date, DateTime, String, ID & Boolean.

(3)

- All primitive data types are passed by value, not by reference.
- All Apex variables, whether they are class member variables, are initialized to null. Make sure that we initialize variables to appropriate values before using them.

Apex primitive datatypes include :-

Boolean :- A value that can only be assigned true, false or null.

Eg:- Boolean isActive = false;

Date :- A value that indicates a particular day. Date values contain no information about time. Date values must always be created with a system static method.

Eg:- Date myDate = Date.newInstance(2013, 05, 15);

Output is 2013-05-15 00:00:00

Time and DateTime :- These are date types associated with dates and times along with Date data type. The Time data types stores times (hours, minutes, second and milliseconds). The Date data types stores dates (Year month and day). The Datetime data type stores both dates and times.

Each of these classes has a newInstance method with which we can construct particular date and time values.

Eg:- Time t1 = newInstance(19, 20, 1, 20);

O/p is 19:20:01

Apex - I

→ We can also create dates and times from the current clock.

Date my = DateTime.now();

Date t = Date.today();

→ The date and time classes also have instance methods for converting from one format to another.

Eg:- Time tq = DateTime.now().time();

→ We can also manipulate the values by using a range of instance methods.

Eg:- Date t3 = Date.today(0);

Date Next = mydate t3.addDays(30);

We will get something like this as the output.

2013-05-15 00:00:00

2013-06-16 00:00:00

Integer, long, Double and Decimal :- To store numeric values in variables, declare variables with one of the numeric data types. Integer, long, Double and Decimal.

Integers :- A 32-bit number that doesn't include a decimal point.

Integers have a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647.

Eg:- Integer i=1;

Apex-I

A4

Long :- A 64 bit number that doesn't include a decimal point. Longs have a minimum value of -2⁶³ and a maximum value of 2⁶³-1.

Eg:- Long l = 2147483648L;

Double :- A 64-bit number that includes a decimal point. Doubles have a minimum value of -2⁶³ and a maximum value of 2⁶³-1.

Eg:- Double d = 3.14159;

Decimal :- A number that includes a decimal point. Decimal is an arbitrary precision number. Currency fields are automatically assigned the type decimal.

Eg:- Decimal dec = 19.23;

Null variables :- If we declare a variable and don't initialize it with a value, it will be null. Null means the absence of a value. We can also assign a null to any variable declared with a primitive type.

Both of these statements result in a variable set to null.

Boolean x = null;

Decimal d;

String :- Strings are set of characters and are enclosed in a single quote. They store text values such as a name or an address.

Eg:- Date d1 = Date today();

String s = String value of(d1);

The O/P of above example should be today's date. 2013-06-16.

Object Types:-

An Object, can be a generic Object or be a specific Objects, such as an Account, Contact, or MyCustom__c.

- Objects (short for "Salesforce Objects") are standard & custom objects that store record data in the Force.com database. There is also an Object data type in Apex that is the programmatic representation of these Objects and their data in code.
- Developers refer to Objects and their fields by their API names.

Eg:- Account a = new Account();

MyCustomObject__c c0 = new MyCustomObject__c();
↓

API name of the custom object.

- The following example creates an invoice Statement with some initial values for the Description__c fields and assigns it to variables of type InvoiceStatement__c, which is an Object type also.

Eg:- InvoiceStatement__c inv = new InvoiceStatement__c

(Description__c = 'TestInvoice', Status__c = 'Pending')

- Object variables are initialized to null, but can be assigned a valid object reference with the new operator.

class :-

class is a collection of datamembers and methods.

Eg:- class Student

{

integer no;

String name;

public void getDetails()

{

System.debug('rollno' + no);

System.debug('name' + name);

{

{

class Employee

{

integer exp;

String department;

void show()

{

// Write the logic

{

These are datamembers of the class.

This is the method of the class.

Variables / datamembers of the class.

Method of the class.

→ To define an Apex class specify the following.

i) Access modifiers:-

→ You must use one of the access modifiers for top level class. (public & global)

→ you do not have to use access modifiers in the declaration of inner classes.

ii) optional definition modifiers such as virtual, abstract.

iii) Required : the keyword class followed by the class name.

iv) optional extensions : AND/OR implementation.

Syntax:-

private | public | global [virtual | abstract] [with sharing] |
(none)]

class className [implements InterfaceNameList |
(none)] [extends className | (none)]

{

// The body of the class.

}

Access modifiers :-

- 1) private :- If you declare a class as private it is only known to the block in which it is declared.
→ By default all the inner classes are private.
 - 2) public :- If you declare class as a public, this class is visible throughout your application and you can access the application anywhere.
 - 3) global :- If you declare a class as global this apex class is visible to all the apex applications in the application or outside the application.
- NOTE:-
If a method or a class (inner) is declared as global then the top level class also must be declared as global.
- 4) With sharing :- If you declare a class as with sharing, sharing rules given to the current user will be taken into the consideration and the user can access & perform the operations based on the permissions given to him on objects & fields. (Field level security, sharing rules).
 - 5) Without sharing :- If you declare a class as a without sharing then this apex class runs in system mode which means apex code has access to all the objects and fields irrespective of current user's sharing rules, field level security, object permissions.

NOTE :-

- 1) If the class is not declared as With sharing or without sharing then by the class is by default taken as
- 2) Both inner classes and outer classes can be declared as Withsharing.
- 3) If innerclass is declared as Withsharing and top level class is declared as Without sharing. then by default entire context will run in Withsharing Context.
- 4) If a class is not declared as With/Without sharing and if this class is called by another class in which sharing is enforced then both the classes run with Withsharing.
- 5) Outerclass is declared as Withsharing and inner class is declared as Without sharing then inner class runs in Without sharing Context only. (Inner classes don't take the properties (sharing properties) from outer class).
- 6) Virtual :- If a class is declared with keyword virtual then this class can be extended (Inherited) & this class methods can be overridden by using a class called overridden.
- 7) Abstract :- This class contains Abstract methods.

Eg1:-

```
public class outerclass
```

```
{
```

```
//code
```

```
class innerclass
```

```
{
```

```
// Innerclass code
```

```
{
```

```
}
```

Eg2:-

```
public with sharing class sharingclass
```

```
{
```

```
//code
```

```
{
```

Eg3:-

```
public without sharing class nosharing
```

```
{
```

```
//code
```

```
{
```

Eg4 :-

public Without sharing class outer

{

//outer class code

Without sharing class inner

{

// Inner class code

{

{

In the above code outer class runs with current user sharing rules. But inner class runs with system context.

Eg5 :-

public Without sharing class outer

{

//outer class code

With sharing class inner

{

// Inner class code

{

{

In this both inner and outer classes runs with current user permissions.

Class Variables :-

The variables in the class should specify the following properties when they are defined.

- i) optional : modifiers such as public or final as well as static.
- ii) required : the data type of the variable , such as String or Boolean.
- iii) optional : The value of the variable
- iv) optional : the name of the variable

Syntax :-

[public | private | protected | global | final] [static]
data_type Variable_name

Eg:-

private static final Integer MY_INT;
private final Integer i = 1;

Class Methods :-

To define a method , specify the following .

- i) optional : modifiers , such as public or protected.
- ii) required : The data type of the value returned by the method , such as String or Integer . use void if the method does not return a value.
- iii) required : A list of input parameters for the method , separated by commas , each preceded by its data type , and enclosed in parentheses () . If

there are no parameters, use a set of empty parentheses.

A method can only have 32 input parameters.

iv) Required : The body of the method, enclosed in braces {} .

All the code for the method, including any local variable declarations, is contained here.

Syntax:-

(public|private|protected|global) [override] [static]
data_type method_name (input parameters)

{

// the body of the method.

}

Eg:-

public static Integer getInt()

{

return MY_INT;

}

Eg2:- public class Example

{

public Integer show(Integer age)

{

System.debug('my age is ' + age);

}

}

Object :-

Object is a instance of the class. This has both State and behaviour.

→ Memory for the data members are allocated only when you create a object.

Syntax:-

Classname objectname = new Classname();

This is the name of class for which we are creating an object.

It's a reference variable

It is a keyword which are allocating the memory

Constructor

Eg:-

```
class Example
{
    // code
}
```

Example e = new Example();

Constructor:-

Constructor is a special method which have the following properties.

- Method name will be same as class.
- Access specifier will be public.
- This method will be invoked only once that is at the time of creating an object.

iv) this is used to instantiate the data members of the class.

Eg:-

```
public class TestObject
```

```
{
```

```
// the no argument constructor
```

```
public TestObject()
```

```
{
```

```
// Code
```

```
{
```

```
}
```

These are 3 types of constructors.

- 1) Default Constructors
- 2) Non-parameterized Constructors
- 3) Parameterized Constructors

1) Default Constructors:

If an Apex class doesn't contain any constructor then Apex Compiler by default creates a dummy constructor on the name of class when we create an object for the class.

Eg:-

```
public class Example
```

```
{
```

Example e = new Example();

In the above example the apex class doesn't contain any constructor. so when we create object for Example class the Apex compiler creates a default constructor.

Eg:-

```
public Example()
{
```

}

89

2) Non-parameterized constructor & parameterized constructor :-

It is a constructor that doesn't have any parameters, or constructor that has parameters.

Eg:- public class Example

{

Integer xno;

String Name;

public Example(Integer x, String myname)

{

xno = x;

name = myname;

{

public Example()

{

//code x.no=10;
 name=sam;

{

} } → parameterized
constructor

→ this is non-parameterized
constructor

Write a Apex program to demonstrate usage of Constructor.

- 1) Open developer console under by clicking the name on the salesforce page.
- 2) click file & select Apex class.
- 3) Enter the class name.
- 4) Write the Apex class.

Eg:-

```
Public class Employee
{
    String EmployeeName;
    Integer EmployeeNo;
    Public Employee()
    {
        EmployeeName = 'Hari';
        EmployeeNo = 10;
    }
    Public void Show()
    {
        System.debug('EmployeeName is '+EmployeeName);
        System.debug('EmployeeNo is '+EmployeeNo);
    }
}
```

5) Open the anonymous block.

```
Employee e1 = new Employee();
Employee e2 = new Employee();
e1.show();
e2.show();
```

This will give an output of Employee Name is Hari and EmployeeNo is 10.

EmployeeName is Hari.
EmployeeNo 10.

usage of Apex program with within Visualforce page :-

1) When you want to call Apex class in Visualforce page we have to declare in the following format.

<Apex:page Controller="class name">

Whenever we call a Visualforce page in which Controller attribute is defined it will first create an object for the apex class which is defined in Controller.

2) When object is created for the Apex class first it invokes the Constructor.

Referring to the apex class members in visualforce :-

When you want to refer apex class variables in the visualforce page we need to use getter & setter methods.

~~get()~~

public class Example

{

String name;

}

get method :-

When visualforce page want to get the value of a variable defined in the Apex. It will invoke get method of that variable.

Eg:-

<apex:outputlabel>{!myname}</apex:outputlabel>



this is a variable defined in apex class.

In the above statement visualforce page is trying to use myname variable which is declared in Apex class. so it is invoke automatically getMyname() method in the apex class and this method will return the value of that.

public class Example

{

String name ;

public void set(String name)

{

this.name = name;

{

public String getName()

{

return name;

{

{

→ setter method. This will take the value from the Visualforce page and stores to the Apex variable name.

→ getter method. This method will return a value to a Visualforce page whenever a name variable is called.

Ex:-

public class Example

{

Integer no;

public void set(String Integer no)

{

this.no = no;

{

public Integer getNo()

{

return no;

{

{

Write an example for getters method using visualforce and apex class:-

Example class:-

```
public class Example
```

```
{
```

```
String name;
```

```
public String getName()
```

```
{
```

```
return 'Sam';
```

```
}
```

```
}
```

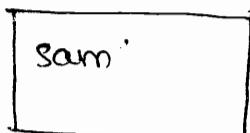
Example Page:-

```
<apex:page controller="Example">  
  <apex:outputLabel> ${!name} </apex:outputLabel>  
</apex:page>
```

→ In the above program when the page is loaded first it creates an object for the example class.

→ When outputlabel calls label in the visu \${!name} in the VF page it invokes getName() method in the controller class. which will return "Sam".

Output:-



Apex class Example :-

public class Example

{

 String name;

 public Example() →

{

 name = 'Hari';

}

 public String getName()

{

 return name;

}

}

Example page :-

<apex:page controller="Example">

<apex:outputlabel> Your name is {!name} </apex:outputlabel>

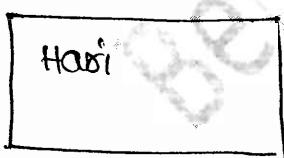
</apex:page>

- ① When visualforce page is loaded it creates an object for the Example class.

- ② At the time of creating the object it first calls the constructor and assign name = 'Hari'.

- ③ When visualforce page called {!name} it invoke getName() in the apex class.

Output :-



③ Writing the values into Apex variables from visualforce

Page :-

This is called read/write operation on the variable.

Eg:- {!age}

public void setAge(Integer age)

{
 this.age = age;
}

Eg:- {!name}

public void setName(String name)

{
 this.name = name;
}

Write an apex class to return the value to visualforce

Page :-

Apex class :-

public class Example1

{
 public Integer age;

 public Example1()
 {

 age = 10;
 }

}

```

public Integer getAge()
{
    return age;
}

public String getName()
{
    return 'Sam Kumar';
}

```

If page:-

```

<apex:page controller="Example1">
    <apex:outputLabel>{!age}</apex:outputLabel>
    <apex:outputLabel>{!name}</apex:outputLabel>
</apex:page>

```

Write an apex class to demonstrate setter method i.e., passing the values and saving the values to apex variables:-

Apex class:-

```

public class Example1
{
    public String name;

    public String getName()
    {
        return name;
    }
}

```

```
public void setName(String name)
{
    this.name=name;
}
```

If page :-

```
<apex:page controller="Example1">
<apex:form>
    <apex:outputLabel> Enter Name </apex:outputLabel>
    <apex:inputText value="{!name}" />
    <apex:commandButton value="click" reRender="one"/>
    <apex:outputLabel id="one"> Your name is {!name} </apex:outputLabel>
</apex:form>
</apex:page>
```

→ We can also define setter and getter methods in a single line.

```
public Integer{set;get;}
```

How to call the apex methods in a Visualforce page:-

public class Demo

{

 public PageReference show()

{

 return null; // when we give return null it will come
 & back to the same page.

}

<apex:commandbutton value = "click" action = "={!show}" />

→ When we click on the "click" button it will invoke
PageReference show() method.

→ PageReference is the return type of every method that we
have called from visualforce page.

Public class Example1

{

 public String name;

 public String getName()

{

 return name;

}

```
public void setName(String name)
{
    this.name = name;
}

public PageReference show()
{
    name = 'This is my name' + name;
    return null;
}
```

~~Satis~~

```
<apex:page controller="Example1">
<apex:form>
    <apex:outputLabel> Enter Name </apex:outputLabel>
    <apex:inputText value="{!name}" />
    <apex:commandButton value="click" reRender="one"
        action="{!show}" />
    <apex:outputLabel id="one">{!name}</apex:outputLabel>
</apex:form>
</apex:page>
```

Eg:- Simple apex class to perform addition and subtraction based on the button you have clicked.

public class Example1

```
{
    public Integer bvalue { get; set; }
```

```
    public Integer avalue { get; set; }
```

```
    public Integer result { get; set; }
```

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

```
    public PageReference subbc()
```

```
{
    result = avalue - bvalue;
```

```
    operation = 'SUBTRACTION';
```

```
    return null;
```

```
}
```

```
    public PageReference subbc()
```

```
{
```

```
    result = avalue + bvalue;
```

```
    operation = 'ADDITION';
```

```
    return null;
```

```
}
```

```
}
```

```
<apex:page controller="Example1">
<apex:form>
<apex:pageBlock title="calculator">
<apex:pageBlockSection columns="1" title="Simple Operations"
collapsible="false">
<apex:pageBlockSectionItem>
<apex:outputLabel> Enter A value </apex:outputLabel>
<apex:inputText value=" {!aValue} "/>
</apex:pageBlockSectionItem>
<apex:pageBlockSectionItem>
<apex:outputLabel> Enter B value </apex:outputLabel>
<apex:inputText value=" {!bValue} "/>
</apex:pageBlockSectionItem>
<apex:pageBlockSectionItem>
<apex:outputLabel> You have performed {!operation}
of {!aValue} and {!bValue} and the result is
{!result} </apex:outputLabel>
</apex:pageBlockSectionItem>
</apex:pageBlockSection>
</apex:pageBlock>
</apex:form>
</apex:page>
```

COLLECTIONS (17-31)

1. Arrays
2. Program to display array of records in pageblock table
3. program to create picklist field in VF page using arrays in Apex.
4. Difference between Array & collection
5. List introduction
6. Methods in the List
7. Program to demonstrate usage of List in Apex.
8. Example programs using List, Apex & visualforce
9. Set introduction
10. Methods in set
11. Program to demonstrate usage of Set in Apex.
12. Example programming using set, Apex & visualforce.
13. Map introduction.
14. List of Methods in Map
15. program to demonstrate usage of Map in Apex
16. Example programs using Map, Apex & visualforce
17. Example to create dynamic picklist using collections.

Satish Mwia

Array:-

Array is a collection of similar elements , where the memory is allocated sequentially .

`Datatype[] arrayname = new Datatype[size]; //this is called dynamic declaration.`

~~`Datatype[] array name = new Datatype[] { value1, value2 }; // Static declaration`~~

~~`Integer[] marks = new Integer[] { 10, 20, 30 };`~~

~~`Account a1 = new Account(name = 'Sam');`~~

~~`Account a2 = new Account(name = 'ram');`~~

~~`Account[] acc = new Account[] { a1, a2 };`~~

~~`String[] s1 = new String[] { 'ram', 'sam', 'ram' };`~~

~~`String[] s1 = new String[4];`~~

~~`s1[0] = 'kumar';`~~

~~`s1[1] = 'Ravi';`~~

Q:- Write a program to display array of strings in pageBlockTable

Ans:- Public class ArrayExample {

```
public String[] myval {set;get;}  
public String name {get;set;}  
public ArrayExample()  
{  
    name = 'prasad';  
    myval = new String[] {'sam', 'ram', 'kiran'};  
}
```

```
<apex:page Controller="ArrayExample">  
<apex:form>  
<apex:pageblock>  
<apex:pageblockTable value="{!myval}" var="a">  
    <apex:column value="{!a}"/>  
</apex:pageblockTable>  
<apex:outputLabel>{!name}</apex:outputLabel>  
</apex:pageblock>  
</apex:form>  
</apex:page>
```

Q:- Write a program to display array of account records.

Ans:-

```
public class ArrayExample {
```

```
    public Account[] myVal{set;get;}
```

```
    public ArrayExample()
```

```
}
```

```
    Account a1=new Account(name='Sashi',
```

89

```
        industry='Banking');
```

```
    Account a2=new Account(name='Ravi', Industry='Banking');
```

```
    Account a3=new Account(name='Praveen', Industry='Banking');
```

```
    myval=new Account[]{a1,a2,a3};
```

```
}
```

```
}
```

~~Output~~ ~~Sample Output~~

```
<apex:page controller="ArrayExample">
```

```
    <apex:form>
```

```
        <apex:pageBlock>
```

```
            <apex:pageBlockTable value="<%myVal%>" var="a">
```

```
                <apex:column value="<%a.name%"/>
```

```
                <apex:column value="<%a.industry%"/>
```

```
</apex:pageBlockTable>
```

```
<apex:outputLabel>{!name}</apex:outputLabel>
</apex:pageBlock>
</apex:form>
</apex:page>
```

Picklist using Selectoption from Apex :-

i) <apex:selectList size="1">
 <apex:selectOption itemLabel="Java" itemValue="scrp"/>
 <apex:selectOption itemLabel="SF" itemValue="SDFC"/>
</apex:selectList>

In the apex program we can create same selectoption using a object selectoption.

Selectoption op1 = new Selectoption(itemvalue, itemlabel)

public class SelectExample {

 public Selectoption[] myoptions {set; get;}

 public SelectExample()

{

 Selectoption op3 = new Selectoption('null', '-None-');

 Selectoption op1 = new Selectoption('one', 'Jan');

```

selectoption op2 = new selectoption('two', 'Feb');
myoptions = new selectoptions[] {op3, op1, op2};
}
}

```

~~DATA~~

```

<apex:page controller="SelectExample">
<apex:form>
<apex:selectlist size="1">
<apex:selectoptions value=" {! myoptions } ">
</apex:selectoptions>
</apex:selectlist>
<apex:selectlist size="1">
<apex:selectoption itemlabel="Java" itemValue="Java">
</apex:selectoption>
<apex:selectoption itemlabel="SFDC" itemValue="SF">
</apex:selectoption>
</apex:selectlist>
</apex:form>
</apex:page>

```

Collections :-

Difference between Array and collections.

Array

1. Array is a collection of homogeneous (similar) elements.
2. Arrays can not grow and shrink dynamically.
3. Arrays can be accessed faster and less memory.

collections.

1. It is a collection of homogeneous & heterogeneous elements.
2. It can grow and shrink dynamically.
3. Collections are slow compared to arrays. They consume more memory.

List :-

- List is an interface.
- A list is an ordered collection of elements that are distinguished by their indices. i.e.
- List elements can be of any datatype - primitive types, collections, objects, user-defined types and built-in apex types.

Index 0	Index 1	Index 2	Index 3	Index 4
Green	Blue	Yellow	Red	Black

- Insertion Order is preserved.
- can grow dynamically at run time.
- Duplicate values are allowed.
- null values are accepted.

Methods in List class :-

add(Object) :- Adds an element to the end of the list.

add(Integer, Object) :- Inserts an element into the list at the specified index position.

addAll(List) :- Adds all of these elements in the specified list to the list that calls the method. Both lists must be of the same type.

addAll(Set) :- Add all of the elements in specified set to the list that calls the method. The set and the list must be of the same type.

clear() :- Removes all elements from a list, consequently setting the list's length to zero.

clone() :- Makes a duplicate copy of a list.

deepClone(Boolean, Boolean, Boolean) :- Makes a duplicate copy of a list of subject records, including the subject records

themselves.

equals(List) :- compares this list with the specified list and returns true if both lists are equal . otherwise returning false.

get(Integer) :- Returns the list element stored at the specified index.

getObjectType() :- Returns the token of the object type that makes up a list of subjects.

hashCode() :- Returns the hashCode corresponding to this list and its contents.

isEmpty() :- Returns true if the list has zero elements.

iterator() :- Returns an instance of an iterator for this list.

remove(Integer) :- Removes the list element stored at the specified index, returning the element that was removed.

set(Integer,object) :- sets the specified value for the element at the given index.

size() :- Returns the number of elements in the list .

sort() :- sorts the items in the list in ascending Order.

Eg:- List<String> str = new List<String>();

String s1 = 'sam';

String s2 = 'Ram';

String s3 = 'Ravi';

str.add(s1);

str.add(s2);

str.add(1, s3);

List<String> finalist = new List<String>();

finalist.addAll(str);

String x = str.get(1); // Ravi

→ Write a program to demonstrate the list.

Public class ListExample {

public List<String> result {set; get;}

public ListExample() {

}

result = new List<String>();

result.add('sam');

result.add('ram');

result.add('hari');

result.add(1, 'kumar');

} {

```
<apex:page Controller="ListExample">  
    <apex:pageblock>  
        <apex:pageblockTab value="${!default}" var="a">  
            <apex:column value="${!a}"/>  
        </apex:pageblockTab>  
    </apex:pageblock>  
</apex:page>
```

List of Objects Demo :-

```
Public class ListExample {
```

```
    Public List<Account> account {set; get;}
```

```
    Public ListExample() {
```

```
        {
```

```
            Account a1 = new Account(name='sam' Industry='Banking');
```

```
            Account a2 = new Account(name='zam' Industry='Energy');
```

```
            result = new List<Account>();
```

```
            result.add(a1);
```

```
            result.add(a2);
```

```
        }
```

```
}
```

```

<apex:page Controller = "ListExample">
  <apex:pageBlock>
    <apex:pageBlockTable value = "{!result}" var = "a">
      <apex:column value = "{!a.name}"/>
    </apex:pageBlockTable>
  </apex:pageBlock>
</apex:page>

```

Create a list of Apex class objects:

1. Student Apex:-

global class Student

{

public String name {get; set;}

public Integer age {get; set;}

public Student (String name, Integer age)

this.name = name;

this.age = age;

}

}

2. List Example Apex :-

```
public class ListExample
```

```
{
```

```
public List<Student> result {set;get;}
```

```
public ListExample()
```

```
{
```

```
result = new List<Student>();
```

```
Student s1 = new Student('sam', 20);
```

```
Student s2 = new Student('Ram', 40);
```

```
Student s3 = new Student('praveen', 40);
```

```
result.add(s1);
```

```
result.add(s2);
```

```
result.add(s3);
```

```
}
```

```
}
```

List Example visualforce :-

```
<apex:page controller="ListExample">
```

```
<apex:pageblock>
```

```
<apex:pageblockTable value="<% result %>" var="a">
```

```
<apex:column value="<% a.name %"/>>
```

```

<apex:column value="<%! e.age %"/>"
</apex:pageBlockTable>
</apex:pageBlock>
</apex:page>

```

Write an apex program to generate list of selectoptions :-

public class ListSelect

{

 public List String myval {get;set;}

 public List<SelectOption> myoptions;

 public List<SelectOption> getMyoptions()

{

 return myoptions;

}

 public ListSelect()

{

 myoptions = new List<SelectOption>();

 Selectoption s = new Selectoption('null', '-none-');

 Selectoption s1 = new Selectoption('one', 'Jan');

 myoptions.add(s);

 myoptions.add(s1);

```
myoptions.add(new Selectoption('two','Feb'));  
myoptions.add(new Selectoption('three','Mar'));
```

3
3

```
<apex:page controller="ListSelect">  
<apex:form>  
<apex:selectList size="1" value="<%! myval %>">  
<apex:selectOptions value="<%! myoptions %>">  
<apex:actionSupport event="onchange" dependency="one"/>  
</apex:selectList>  
<apex:outputLabel id="one"> You have Selected Satish   
 </apex:outputLabel>  
</apex:form>  
</apex:page>
```

Set :-

It is an unordered collection of elements where elements can be of any data type. primitive types, collections, objects, user-defined types, and built-in Apex types.

1. Set don't allow the duplicate values.
2. Insertion order is not preserved in the set.
3. It grows dynamically at run time.

Eg:-

```
Set<String> names = new Set<String>();
```

```
Set<Account> acc = new Set<Account>();
```

```
Set<Customer__c> mycustomers = new Set<Customer__c>();
```

public void add(Object) // This method will add elements to the set.

```
Set<String> names = new Set<String>();
```

```
names.add('one');
```

```
names.add('two');
```

```
names.add('one');
```

NOTE:- Set will not allow duplicates, but if we insert it will not raise any error it will not take value.

public void addAll(List) // This method will add all elements to the set.

```
List<String> mylist = new List<String>();  
mylist.add('one');  
mylist.add('two');  
  
Set<String> mynames = new Set<String>();  
mynames.add('sam');  
mynames.addAll(mylist);  
  
public Integer size() // This method will return no. of  
elements in the set.
```

Set<String> names = new Set<String>();

```
names.add('one');  
names.add('two');  
names.add('one');
```

Integer mysize = names.size(); // This will return 3 and
Stored to my size variable.

Public void remove(Integer index) // This method will remove
the elements at the specified index.

```
names.remove(1);
```

Example:- write an apex class to demonstrate set.

```
public class SetExample
{
    public Set<String> names {get; set;}
    public SetExample()
    {
        names = new Set<String>();
        names.add('one');
        names.add('two');
        names.add('sam');
        names.add('one');
        names.add('one');
    }
}
```

If page :-

```
<apex:page controller="SetExample">
    <apex:pageBlock>
        <apex:pageBlockTable value="{!names}" var="a">
            <apex:column value="{!!a}"/>
        </apex:pageBlockTable>
    </apex:pageBlock>
</apex:page>
```

→ Write an apex class to add list of elements to set.

public class SetExample

{

public set<String> names {get;set;}

public SetExample()

{

List<String> mylist = new List<String>();

mylist.add('sashi');

mylist.add('sashi');

mylist.add('saksh');

mylist.add('saksh');

names = new set<String>();

names.add('one');

names.add('two');

names.add('one');

names.add('one');

names.add(mylist);

}

}

public void retainAll(List) // this will keep only the values
that are existing in list and set.

remove rest of the values.

public class SetExample {

}

public Set<String> names { get; set; }

public boolean test { get; set; }

public SetExample ()

{

List<String> myList = new List<String>();

myList.add('sashi');

myList.add('rakesh');

names = new Set<String>();

names.add('sashi');

names.add('ram');

names.add('kumar');

test = names.retainAll(myList);

}

}

visualforce page:-

```
<apex:page controller="setExample">
<apex:pageBlock>
<apex:outputLabel> ${!test}</apex:outputLabel>
<apex: dataList value="${!names}" var="a"> ${!a}
</apex: dataList>
</apex: pageBlock>
</apex: page>
```

Satish Anvia

Map :-

A map is a collection of key-value pairs where each unique key maps to a single value. Keys and values can be any datatype - primitive types, collections, objects, user-defined types, and built-in Apex types.

For Eg, the following table represents a map of Countries and currencies.

Country (key)	'United States'	'Japan'	'France'	'England'	'India'
Currency (value)	dollar	'yen'	'Euro'	'pound'	'Rupee'

clear() :- Remove all of the key-value mappings from the map.

clone() :- Makes a duplicate copy of the map.

containsKey(Object) :- Returns true if the map contains a mapping for the specified key.

deepClone() :- Makes a duplicate copy of a map, including object records if this is a map with object record values.

equals(Map) :- Compare this map with the specified map & returns true if both maps are equal. Otherwise returns false.

get(Object) :- Returns the value to which the specified key is mapped, or null if the map contains no value for this key.

getSubjectType() :- Returns the token of the subject type that makes up the map values.

hashCode() :- Returns the hashCode corresponding to this map.

isEmpty() :- Returns true if the map has zero key-value pairs.

keySet() :- Returns a set that contains all of the keys in the map.

put(Object, Object) :- Associates the specified value with the specified key in the map.

putAll(Map) :- copies all of the mappings from the specified map to the original map.

putAll(List) :- Adds the list of subject records to a map declared as Map<ID, Subject> & Map<String, Subject>.

remove(key) :- Removes the mapping for the specified key from the map, if present, and returns the corresponding value.

size() :- Returns the no. of key-value pairs in the map.

values() :- Returns a list that contains all of the values in the map in arbitrary order.

Set of key: key can not be duplicate

List of values: It allows duplicates.

`Map<Integer, String> my = new Map<Integer, String>();`

`Set<Integer> keys = List<String> values`

`void put(key, value)`

`m.put(1, 'sam');`

`m.put(2, 'Ram');`

`m.put(3, 'kumar');`

$\{1 \rightarrow 'sam', 2 \rightarrow 'Ram', 3 \rightarrow 'kumar'\}$

→ `Object get(key)` when we give key it will return values.

`String x = m.get(1) \Rightarrow sam`

`String y = m.get(3) \Rightarrow kumar`

→ `Set<Object> keyset()` :- If we use this method we will get set of keys.

`Set<Integer> s = m.keySet(); $\Rightarrow \{1, 2, 3\}$`

→ `List<Object> values()` :- It will return list of values in the map.

`List<String> val = m.values(); $\Rightarrow \{'sam', 'sam', 'kumar'\}$`

```
public List<String> myresult {get; set;}
```

```
Map<String, String> mymap;
```

```
public MapExample()
```

```
{
```

```
myresult = new List<String>();
```

```
mymap = new Map<String, String>();
```

```
mymap.put('Hyd', 'Satish');
```

```
mymap.put('Warangal', 'Reddy');
```

```
mymap.put('Korimnagar', 'Chaitu');
```

```
}
```

```
public PageReference show()
```

```
myresult.clear();
```

```
Set<String> keys = mymap.keySet();
```

```
myresult.addAll(keys);
```

```
return null;
```

```
}
```

```
public PageReference display()
```

```
{
```

```
myresult = mymap.values();
```

```
return null;
```

```
}
```

```
}
```

```
Map<String, List<String>> mybranches = new Map<String,
List<String>>();
```

```
List<String> hyd = new List<String>();
```

```
Hyd.add('SR');
```

```
Hyd.add('Jb');
```

```
Mybranches.put('Hyd', Hyd);
```

```
List<String> war = new List<String>();
```

```
Hyd.add('Kazipet');
```

```
Hyd.add('Bhalampali');
```

```
Mybranches.put('war', war);
```

```
Mybranches.put('war', war);
```

```
List<String> adi = new List<String>();
```

```
Adi.add('Manchiyedal');
```

```
Adi.add('godavari');
```

```
Mybranches.add('adi', adi);
```

```
Set<String> keys = Mybranches.keySet();
```

```
for (String s : keys)
```

```
{}
```

```
Selectoption op = new Selectoption(s, s);
```

```
}
```

```
List<String> val = mybranches.get(city);
```

```
Public class DependExample {
```

```
    public pageReference show() {
```

```
        List<String> bran = mybranches.get(mycity);
```

```
        branch.clear();
```

```
        for(String x : bran)
```

```
    {
```

```
        SelectOption op1 = new Selectoption(x, x);
```

```
        branch.add(op1);
```

```
}
```

```
    return null;
```

```
}
```

Apex class :-

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

```
Map<String, List<String>> mybranches = new Map<String,
```

```
    List<String>>();
```

```
    public List<Selectoption> city {set; get; }
```

```
    public List<Selectoption> branch {set; get; }
```

```
    Public DependExample()
```

```
{
```

```
    List<String> hyd = new List<String>();
```

```
    hyd.add('soragarh');
```

```
    hyd.add('lbinagar');
```

```

List<String> bang = new List<String>();
bang.add('Ecity');
bang.add('Marthali');
mybranches.put('Hyd', hyd);
mybranches.put('Bang', bang);
Set<String> keys = mybranches.keySet();
city = new List<Selectoption>();
branch = new List<Selectoption>();
city.add(new Selectoption('null', 'None'));
List<String> my = new List<String>();
my.add('None');
for(String a: keys)
{
}
    
```

Selectoption opt = new Selectoption(a, a);
 city.add(opt);

If page :-

<apex:page Controller="DependExample">

<apex:form>

```
<apex:selectList value="={!myCity}" size="1">  
<apex:selectOptions value=" {!city} "></apex:selectOptions>  
  
<apex:actionSupport event="onchange" action=" {!show} ">  
    renderer="one"/>  
</apex:selectList>  
  
<apex:selectList size="1" id="one">  
    <apex:selectOptions value=" {!branch} ">  
        </apex:selectOptions>  
</apex:selectList>  
  
<apex:form>  
</apex:page>
```

Example :-

Apex class :-

```
public class List&MapsController {  
    public Map<String, String> inputFields {get; set;}  
  
    public List&MapsController() {  
        inputFields = new Map<String, String>();  
        'firstName' => 'Tommy', 'lastName' => 'Applesed', 'age' =>  
        '42';  
    }  
}
```

```
public PageReference submitFieldData() {
    doSomethingInterestingWithInput();
    return null;
}
```

```
public void doSomethingInterestingWithInput()
```

```
    inputFields.put('age', Integer.valueOf(inputFields.get('age'))
        + 10).format());
```

```
<apex:page controller="ListMapsController">
```

```
<apex:outputpanel id="box" layout="block">
```

```
<apex:pageMessage />
```

```
<apex:form>
```

```
<apex:repeat value="{!inputFields}" var="fieldKey">
```

```
<apex:outputText value=" {!fieldKey} "/>
```

```
<apex:inputText value=" {!inputfields[fieldKey]} "/> <br/>
```

```
</apex:repeat>
```

```
<apex:commandbutton action=" {!submitFieldData}">
    value="Submit" id="button" rerender="box"/>
```


SOQL, SOSL Query (32-48)

1. Introduction about SOQL
2. Difference between Static & Dynamic SOQL
3. SOQL using select query examples
4. Querying multiselect picklist values in SOQL
5. SOQL Queries using Having, not in, like etc..
6. SOQL Sub Queries
7. Date formats, Limits Offset, Group by examples
8. Relationship Queries (parent to child, child to parent)
9. Example programs using Relationship querying and Apex.
10. Introduction to SOSL
11. Dynamic SOSL Examples

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

SOQL Queries :-

Salesforce object query language is used to query the records from the database.com based on the requirement.
There are 2 types of SOQL statements.

- 1) Static SOQL
- 2) Dynamic SOQL

1) Static SOQL :- the Static SOQL Statement is written in [] (array brackets).

→ This statements are similar to LINQ.

Eg:- String Searchfor = 'Jones';

```
Contact[] Contacts = [select testfield_c, firstname,  
                         lastname from Contact where  
                         lastname = :Searchfor];
```

2) Dynamic SOQL :- It is used to refer to the creation of a SOQL String at run time with Apex code.

→ Dynamic SOQL enables you to create more flexible application.

- To create Dynamic SOQL query at run time use Database.Query() method, in one of the following ways.
- Return a single object when the query returns a single record.
- object s = Database.query(String_limit-1);
- return a list of objects when the query returns more than a single record.

Examples :-

Eg1:-

```
String myTestString = 'TestName';
List<Object> l = Database.Query('SELECT Id FROM
    MYCustomObject__c WHERE Name=:myTestString');
```

Eg2:-

```
String resolvedField1 = myvariable.field1__c;
List<Object> l = Database.Query('SELECT Id FROM MYCustomObject__c
    WHERE field1__c = ' + resolvedField1);
```

Syntax of SOQL query:-

SELECT field1, field2.....

FROM objectType

[WHERE condition]

Examples :-

SELECT Id, Name

List<Account> acc = [Select Id, Name from Account];

List<Account> acc = [Select id, name from Account where
annual_revenue < 100000];

Q) Write a query to fetch customer name, Balance from
customer object where balance is more than 100000.

List<Customer__c> customers = [Select Id, customer_name__c,
Balance__c from Customer__c
where Balance__c > 100000];

Q) Write a query to fetch TID , type__c from transaction
if mode is 'cash'.

List<Transaction__c> tra = [Select Name, type__c from
Transaction__c where
Mode__c = 'cash'];

Querying Multiselect picklist values :-

When you want to write a query with a condition based on multi select picklist in which multiple item values are selected then we can use the following operators to write the condition.

NOTE:- First let us create proof^{field} with datatype multiselect with the values

- i) passport
- ii) Aadhar
- iii) Voter Id
- iv) PAN card

Q) Write a query to fetch all the customer names & account types who has submitted PAN card as a proof.

List<customer_c> customers = [Select id, customer_name_c,
Account_Type_c from customer_c where
proof_c = 'PAN card'];

Q) Write a query to fetch customer name, balance from Customer object whose proof is pan card or Aadhar card.

NOTE:- When you want to verify whether the value what we have selected is in the list or not by using INCLUDES.

```
List<Customer__c> customers = [SELECT Customer_Name__c,  
                                Balance__c FROM Customer__c WHERE  
                                Proof__c INCLUDES ('Aadharcard', 'pancard')]
```

Q) Write a query to fetch the list of Ids & customer names from customer object where proof is Aadhar card or passport and voter id.

When you want to verify whether we have selected more than one value then use ;

Proof__c =: 'passport; voterid'

```
List<Customer__c> customers = [SELECT Customer_Name__c,  
                                Balance__c FROM Customer__c WHERE  
                                Proof__c INCLUDES ('Aadharcard', 'passport; voterid')]
```

LIKE :-

Expression is true if the value in the specified field name matches the characters of the text string in the specified value.

→ The LIKE operator in SOQL and SOSL is similar to the LIKE operator in SQL.

It provides a mechanism for matching partial text strings and includes support for wildcards.

- The % and _ wildcards are supported for the LIKE operator.
- The % wildcard matches zero or more characters.
- The _ wildcard matches exactly one character.
- The text string in the specified value must be enclosed in single quotes.
- The LIKE operator is supported for string fields only.
- The LIKE operator performs a case insensitive match, unlike case sensitive match in SQL.

Eg:- SELECT AccountId, FirstName, LastName
FROM Contact
WHERE LastName LIKE 'app%'

IN :-

If the value equals any one of the specified values in a WHERE clause.

Eg:- SELECT Name FROM Account
WHERE BillingState IN ('california', 'Newyork')

NOT

NOT IN :-

If the value does not equal any of the specified values in a WHERE clause. for eg

Eg:- SELECT Name FROM Account

WHERE BillingState NOT IN ('California', 'New York')

Subquery :-

The query within a query is called subquery.

i. Write a query to return list of Accounts that do not have any open Opportunities.

```
SELECT Id  
FROM Account  
WHERE Id NOT IN
```

(SELECT AccountId → AccountId is a lookup field

 FROM Opportunity in the opportunity.

 WHERE IsClosed = false

)

Eg2:- SEL

Eg 2: SELECT Id

FROM Opportunity
WHERE AccountId NOT IN

(
SELECT AccountId
FROM Contact
WHERE LeadSource = 'Web'
)

NOTE: Whenever we create a master-detail or lookup field id of the master record is stored in this field.

Date formats:-

When you want to use the date formats in query, it should be any one of the below formats.

<u>Format</u>	<u>Format Syntax</u>	<u>Example</u>
Date only	YYYY-MM-DD	1991-01-01
Date, time & time zone offset	YYYY-MM-DDThh:mm:ss±hh:mm ss±hh	Eg:- 1991-01-01T23:01:01+01:00
	YYYY-MM-DDThh:mm:ss±hh:mm	

LIMIT:

Use LIMIT to specify the maximum number of rows to return.

Syntax:-

~~SELECT~~ SELECT fieldList

FROM objectType

[WHERE conditionExpression]

LIMIT number_of_rows

Eg:- SELECT name

FROM Account

WHERE industry = 'Media' LIMIT 125

NOTE:- You can't use a LIMIT clause in a query that uses an aggregate function, but does not use a GROUP BY clause. For example, the following query is invalid.

SELECT MAX(createdDate)

FROM Account LIMIT 1

OFFSET:

We OFFSET to specify the starting row offset into the result set returned by your query.

→ Using OFFSET is helpful for paging into large result sets, in scenarios where you need to quickly jump to a particular subset of the entire results.

Syntax:-

```

SELECT fieldList
FROM objectType
[WHERE conditionExpression]
ORDER BY fieldOrderByList
LIMIT number_of_rows_to_return
OFFSET number_of_rows_to_skip

```

Eg:-

```

SELECT Name
FROM Merchandise_c
WHERE price_c > 5.0
ORDER BY Name
LIMIT 100
OFFSET 10

```

GROUP BY:-

With API version 18.0 and later, you can use GROUP BY with aggregate functions, such as SUM() or MAX(), to summarize the data and enable you to rollup query results rather than having to process the individual records in your code.

Syntax:-

```
[GROUP BY fieldGroupByList]
```

Eg1:-

SELECT Leadsource FROM Lead.

Eg2:-

SELECT Leadsource, COUNT(Name)
FROM Lead
GROUP BY Leadsource.

Eg3:- SELECT LeadSource

FROM Lead

GROUP BY LeadSource.

NOTE:- You must use a GROUP BY clause if your query uses a LIMIT clause and an aggregated function.

Eg:- SELECT Name, MAX(CreatedDate)
FROM Account
GROUP BY Name
LIMIT 5

~~Group By~~

Group By ROLLUP :- This allows the query to calculate subtotals so you don't have to maintain that logic in your code.

Syntax:-

[Group By ROLLUP (fieldName1, fieldName2, fieldName3)]

Eg1: rolls the results up by one field

```
SELECT LeadSource, COUNT(Name) cnt
FROM Lead
GROUP BY ROLLUP (LeadSource)
```

Eg2: rolls the results up by two fields

```
SELECT Status, LeadSource, COUNT(Name) cnt
```

```
FROM Lead
```

```
GROUP BY ROLLUP (Status, LeadSource)
```

Group By CUBE: This is particular useful if you need to compile cross-tabular reports of your data. use Group By CUBE with aggregate functions, such as SUM() and COUNT(fieldName)

Syntax:-

[Group By CUBE [fieldName1, fieldName2, fieldName3]]

Eg: Returns subtotals of accounts for combination of type & Billing city.

```
SELECT Type, BillingCountry,
```

```
GROUPING (Type) grpType, GROUPING (BillingCountry) grpCity,
COUNT(id) accts
```

```
FROM Account
```

```
GROUP By CUBE (Type, BillingCountry)
```

```
ORDER BY GROUPING (Type), GROUPING (BillingCountry)
```

HAVING :-

With API version 18.0 and later, you can use a HAVING clause with a GROUP BY clause to filter the results returned by aggregate functions, such as SUM().

Syntax:-

[HAVING havingConditionExpression]

Eg1 :- Determine how many leads are associated with each leadsource.

SELECT Leadsource, COUNT(Name)

FROM Lead

GROUP BY Leadsource

Eg2 :- Generate more than 100 Leads

SELECT Leadsource, COUNT(Name)

FROM Lead

GROUP BY Leadsource

HAVING COUNT(Name) > 100

Eg3 :- Returns Accounts with duplicate names.

SELECT Name, Count(Id)

FROM Account

GROUP BY Name

HAVING Count(Id) > 1

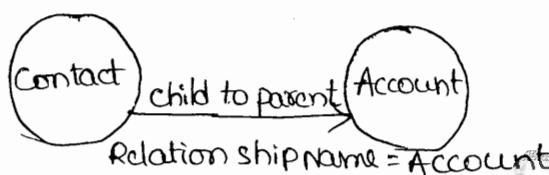
Relationship queries :-

When you want to write a queries based on parent child relationship will be using relationship queries on standard & custom objects.

→ There are two types of relationship queries.

- 1) child to parent relationship
- 2) parent to child relationship.

1) child to parent relationship :-



→ In the above diagram Contact is a child of Account object.

→ In the child to parent relationship relationship name will be parent object which means foreign key is ~~an~~ in Account object.

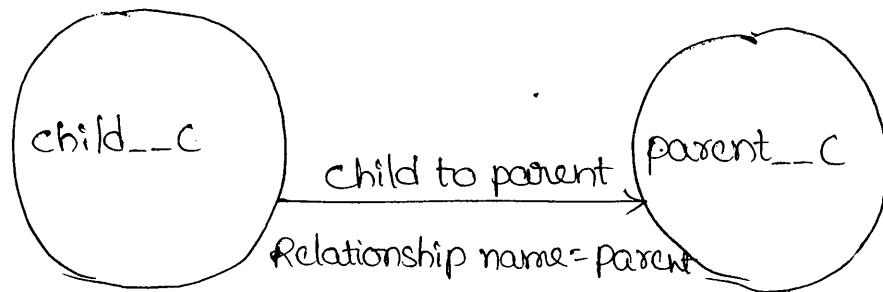
Eg:-

```
SELECT Contact.FirstName, Contact.Account.Name from Contact
```

Eg2:- Write a query to fetch list of contacts and Account names from Contact objects where Account.Industry = 'Media'.

```
SELECT Id, Name, Account.Name FROM Contact
WHERE Account.Industry = 'Media'
```

Child to parent on custom objects :-

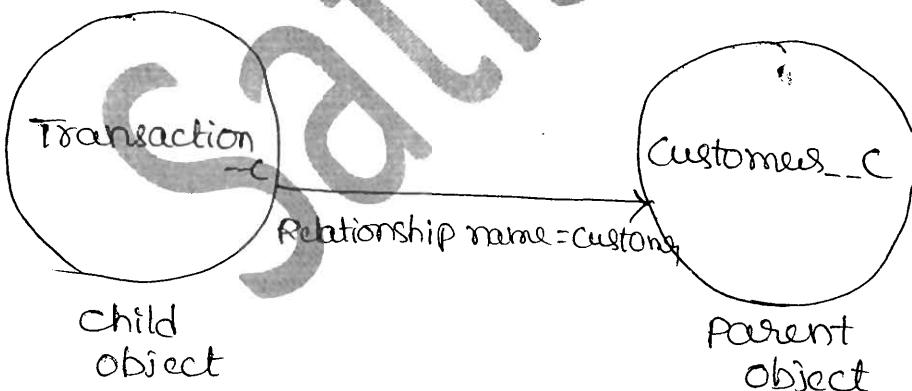


NOTE:- When we use relationship name for the custom objects in SOQL query we should append `ObjectName__r` to the relationship name.

Eg:-

```
List<child__c> ch = [Select Id, Name, parent__r.FirstName,  
parent__r.LastName__c from child__c  
where age__c < 25];
```

Q2:-



NOTE:- When you are writing a query from child to parent relationship always ~~parent object name will be~~ ^{most} relation name will be ~~parent object name will be~~ ^{most} master-detail field name (or) lookup field name.

3) Write a query to fetch list of transactions along with Account type and customer names whose transaction type is deposit.

NOTE: Master-detail field in the transaction is customerDetails_c

List <transactions_c> tx = [select Id, customerDetails_c.
CName_c, customerDetails_c.
AccountType_c, name from
Transaction_c];

Write a apex program to display first five contact details from Contact object along with Account Name and industry.

Apex class :-

public class childparent {

public List <Contact> cont {get; set;}

public pageReference show()

cont = [select id, Name, Account.Name, Account.Industry
from Contact Limit 5];

return null;

}

}

Vf page :-

```
<apex:page controller="childparent">  
<apex:form>  
    <apex:pageBlock rendered=" {!NOT(ISNULL(cont))} " >  
        <apex:pageBlockTable value=" {!cont} " var="a" >  
            <apex:column value=" {!a.Name} " />  
            <apex:column value=" {!a.Account.Name} " />  
            <apex:column value=" {!a.Account.Industry} " />  
        </apex:pageBlockTable>  
    </apex:pageBlock>  
    <apex:CommandButton value="click" action=" {!show} " />  
</apex:form>
```

A42

Q:- write a Apex program to display the transaction details along with customers name , Account type, city using transaction object.

Apex class :-

public class childparent

{

public List<Transaction__c> tran {get; set; }

public pageReference show()

{

tran = [select Type__c, customer_Details__x,

Account_Type__c, customer_Details__x. customer_Name__c,

customer_Details__x. city__c from Transaction__c];

return null;

}

}

If page :-

<apex:page controller = "childparent" >

<apex:form>

<apex:commandButton value = "click" action = " {!show } "/>

<apex:pageBlock>

<apex:pageBlockTable value = " {!tran } " var = "a" >

<apex:column value = " {!a. customer_Details__x. customer_Name__c } "/>

.

```

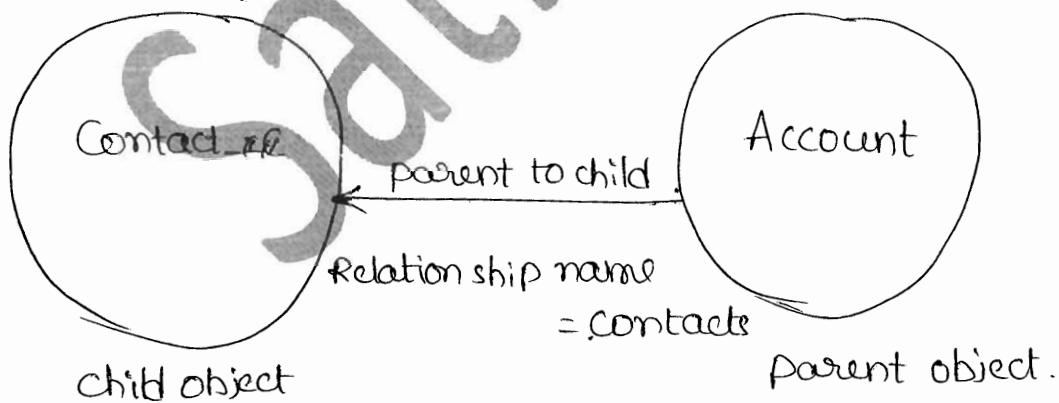
<apex:column value = "${!a.Customer_Details__r.Account_Type__c}">
<apex:column value = "${!a.Customer_Details__r.City__c}">
</apex:pageBlock>
</apex:form>
</apex:page>

```

Parent to child relationship on standard objects :-

When you are trying to write a query to refer the child object records from the parent object then the child object name would be the relationship name.

→ the child object name should be referred from with plural name.



Syntax:-

```

SELECT Account.Name , SELECT Contact.FirstName ,
       Contact.LastName FROM Account.Contacts)
      FROM Account

```

Eg1:

```

SELECT Name,
(
    SELECT LastName
    FROM Contacts
)
FROM Account

```

Eg2:- ~~194889~~

```

SELECT Amount, Id, Name,
(
    SELECT Quantity, ListPrice,
    PricebookEntry.UnitPrice, PricebookEntry.Name
    FROM OpportunityLineItems
)
FROM Opportunity

```

Eg3:- ~~194889~~

```

SELECT Amount, Id, Name, (
    SELECT Quantity, ListPrice,
    PricebookEntry.UnitPrice, PricebookEntry.Name,
    PricebookEntry.Product2.Family
    FROM OpportunityLineItems
)
FROM Opportunity

```

Q:- Write an apex program to display list of accounts along with their corresponding contact details.

Apex class :-

public class parentchild

{

 public List<Account> acc {get; set;}

 public pageReference show()

{

 acc = [select Name, Industry, (select AssistantName,
 Email from Contacts) from Account];

 return null;

}

}

VF page :-

<apex:page controller="parentchild">

 <apex:form>

 <apex:commandButton value="click" action="!show"/>

 <apex: dataList value="!acc" var="a">

```

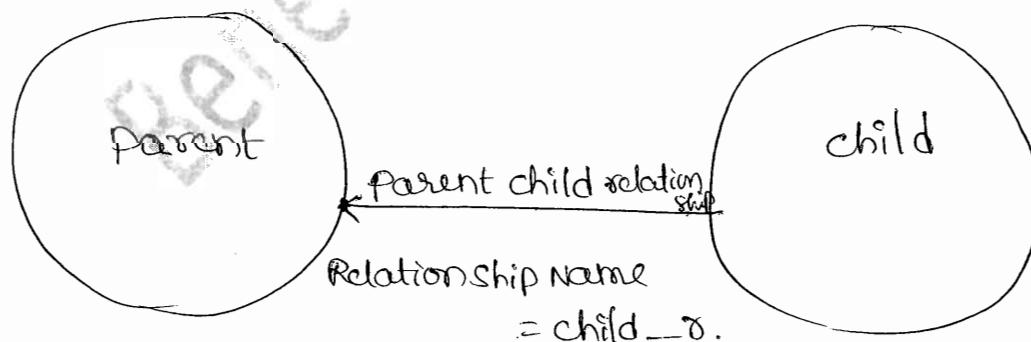
<apex:outputLabel> Account Name : ${!a.name} </apex:outputLabel>
    Account.Industry ${!a.industry} </apex:outputLabel>
<apex: dataList value = "${!a.contacts}" var = "b">
    ${!b.Email} --- ${!b.AssistantName}
</apex: dataList>
</apex: dataList>
</apex: form>
</apex: page>

```

Q:-

Parent to child relationship on custom objects :-

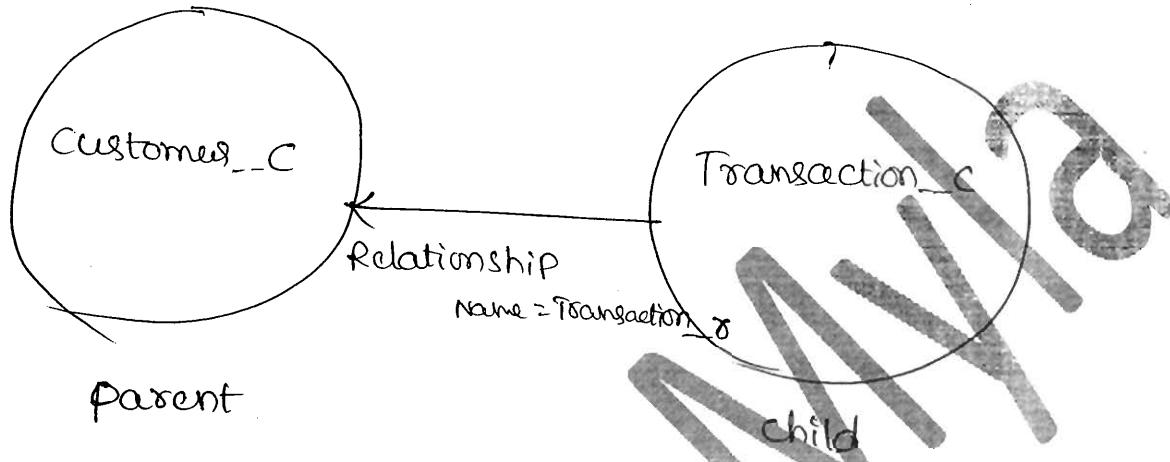
When you are writing a query to refer custom child object fields from the parent child relationship then we have to append child object plural name _s.



Eg:-

SELECT customer_name, Account_Type

[SELECT ParentName_c, Age_c, (select child_name_c, Dob_c from childs_8) from parent_c]



∴ Write an Apex program to print customer details along with corresponding transaction details of the customer whose account type is saving.

Apex class:

public class parentchild

{

Public List<Customer_c> acc {get;set;}

Public PageReference show()

{

acc = [Select customer_name_c, Account_Type_c,

(Select Type__c , Amount__c from Transactions__r)
from Customer__c)] ;

return null;

}

{

<apex:page Controller="parentchild">

<apex:form>

<apex:commandbutton value="click" action=" {!show}" />

<apex:pageblock>

<apex:pageblocktable value=" {!acc}" var="a">

<apex:column value=" {!a.Customer_Name__c}" />

<apex:column value=" {!a.Account_Type__c}" />

<apex:column>

<apex:pageblocktable value=" {!a.Transactions__r}" var="b">

<apex:column> {!b.Type__c} </apex:column>

<apex:column> {!b.Amount__c} </apex:column>

</apex:pageblocktable>

</apex:column>

</apex:pageblocktable>

</apex:pageblock>

</apex:form>

</apex:page>

Satish Mania

SOSL Statements :-

SOSL statements evaluate to a list of Sobjects, where each list contains the search results for a particular object type. The result lists are always returned in the same order as they were specified in the SOSL query.

- If a SOSL query does not return any records for a specified object type, the search results include an empty list for that object.
- For example, you can return a list of accounts, contacts, opportunities and leads that begin with the phrase map.

~~Sample~~
~~Code~~
~~Search~~

```
List<List<Sobject>> searchList = [FIND 'map*' IN ALL FIELDS
    RETURNING Account(Id,Name), Contact, Opportunity, Lead];
```

NOTE:-

The syntax of the FIND clause in Apex differs from the syntax of the FIND clause in the SOAP API.

- In Apex, the value of the FIND clause is demarcated with single quotes. For Eg.

FIND 'map*' IN ALL FIELDS RETURNING Account(Id,Name),
 Contact, Opportunity, Lead.

→ In the Force.com API, the value of the FIND clause is demarcated with single quotes braces. ↴

For Eg:-

FIND {map*} IN ALL FIELDS RETURNING Account (Id, name),
Contact, Opportunity, Lead.

From searchList, you can create arrays for each object returned

Account [] accounts = ((List<Account>) searchList[0]);

Contact [] contacts = ((List<Contact>) searchList[1]);

Opportunity [] opportunities = ((List<Opportunity>) searchList[2]);

Lead [] leads = ((List<Lead>) searchList[3]);

Dynamic SOSL :-

Dynamic SOSL refers to the creation of a SOSL String at runtime with Apex code.

→ Dynamic SOSL enables you to create more flexible applications.
For example, you can create a search based on input from an end user, or update records with varying field names.

→ To create a dynamic SOSL query at runtime, use the Search query method. For example

```
List<List<sObject>> myQuery =
    search.query(SOSL_Search_String);
```

The following example exercises a simple SOSL query string.

```
String searchquery = 'FIND \'Edge*\' IN ALL FIELDS RETURNING
    Account(id, name), Contact, Lead';
```

```
List<List<sObject>> searchList = search.query(searchquery);
```

→ Dynamic SOSL statements evaluate to a list of lists of sObjects, where each list contains the search results for a particular sObjectType. The result lists are always returned in the same order as they were specified in the dynamic SOSL query.

→ The Search query method can be used wherever an inline SOSL query can be used, such as in regular assignment statements and for loops.

SOSL example :-

public with sharing class DeferenceDemoController

{

 public List<Opportunity> optyList {get; set;}

 public List<Lead> leadList {get; set;}

 public List<Contact> conList {get; set;}

 public List<Account> accList {get; set;}

 public DeferenceDemoController()

{

&

 public void sosldemo_method()

{

 OptyList = New List<Opportunity>();

 leadList = New List<Lead>();

 ConList = New List<Contact>();

 accList = New List<Account>();

 List<List<SObject>> SearchList = [FIND 'test' IN ALL FIELDS

 RETURNING Account (Id, Name, type),

 Contact (name, email), Opportunity (name, StageName),

 Lead (Company, name, Status)];

```

acctList = ((List<Account>)searchList[0]);
contactList = ((List<Contact>)searchList[1]);
opptyList = ((List<Opportunity>)searchList[2]);
leadList = ((List<Lead>)searchList[3]);

```

}

}

Vf page :-

< apex:page controller="DefenceDemoController">
 < apex:form >
 < apex:CommandButton value="Show records using SOSL"
 action = "\${!SOSLDemo_method}" />
 < apex:pageBlock title="Accounts" >
 < apex:pageBlockTable value = "\${!acctList}" var = "acc" >
 < apex:column value = "\${!acc.name}" />
 < apex:column value = "\${!acc.type}" />
 </apex:pageBlockTable>
 </apex:pageBlock>
 < apex:pageBlock title = "contacts" >

```
<apex:pageBlockTable value = "${!ContList}" var = "con">  
    <apex:column value = "${!con.name}"/>  
    <apex:column value = "${!con.email}"/>  
</apex:pageBlockTable>  
</apex:pageBlock>  
<apex:pageBlock title = "Leads">  
    <apex:pageBlockTable value = "${!leadList}" var = "lead">  
        <apex:column value = "${!lead.name}"/>  
        <apex:column value = "${!lead.Company}"/>  
</apex:pageBlockTable>  
</apex:pageBlock>  
<apex:pageBlock title = "Opportunities">  
    <apex:pageBlockTable value = "${!optyList}" var = "opty">  
        <apex:column value = "${!opty.name}"/>  
        <apex:column value = "${!opty.StageName}"/>  
</apex:pageBlockTable>  
</apex:pageBlock>  
</apex:form>  
</apex:page>
```

EMAIL PROGRAMMING (50-61)

1. Introduction to Email
2. Types of Emails
3. Single Email Message & Methods
4. Single Email Message Example using Apex & Visualforce
5. Mass Email Message Introduction
6. Mass Email Message Example using Apex & Visualforce
7. Attaching files (PDF, XLS) to Outbound Emails
8. Attaching documents to Outbound Email Message
9. Introduction to Inbound Email Message
10. Example programs

Satish Mawa

Email programming :-

AUG
SD

When you want to send an email from the Salesforce to the external system or receiving a mail from the external system to the Salesforce then we use Email Services. There are 2 types of emails what we have

- 1) Inbound email messaging
- 2) Outbound email messaging

Outbound Email :- This is used to send an email to external system using apex code. There are 2 types of Outbound Emails.

- 1) SingleEmailMessage
- 2) MassEmailMessage.

NOTE :- Messaging namespace provides classes and methods for Salesforce Outbound and Inbound Email functionality.

1) SingleEmailMessage :-

This is a instantiates an email object used for sending a single email message.

Syntax :-

```
Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
```

This class contains methods of sending single email message. When we want to send a message first we have to create the object of `SingleEmailMessage`.

The list of methods available in the `SingleEmailMessage` are

⇒ `SetBcc Addresses (String [])`

This method will set Bcc Address to whom the mail should be sent. We can set upto 25 email address.

Eg:- `String [] toBccAddress = new String [] {'a1@gmail.com', 'a2@gmail.com'};`

`myemail.setBccAddresses (toBccAddress);`

⇒ `SetCc Addresses (String [])`

This method will set cc Address to whom the mail should be sent. We can set upto 25 email address.

Eg:- `String [] toCcAddress = new String [] {'c1@gmail.com'}`

`myemail.setCcAddresses (toCcAddress);`

⇒ `SetTo Addresses (String [])`

This method will set the address, we can set upto 100 addresses.

~~Ans~~
Eg:- String[] toaddress = new String[] { 'abc@gmail.com',
'xyz@gmail.com' } ;
myemail.setToAddress (toaddress);

⇒ setSubject (String)

this method will set the Subject of the mail.

Eg:- myemail.setSubject ('Test Email');

⇒ setPlainTextBody (String)

this method will set the TextBody of the mail.

Eg:- myemail.setPlainTextBody ('Hi Sam');

⇒ setHtmlBody (String)

this method will set the main body of the mail.

Eg:- myemail.setHtmlBody ('<h1> this is the link for
Registration </h1>');

Example :-

Apex class :-

```
public class mailer
```

```
{
```

```
public pageReference show()
```

```
{
```

```
    Messaging.SingleEmailMessage msg = new Messaging.
```

```
        .SingleEmailMessage();
```

```
    String[] toaddress = new String[]{'abc@gmail.com',  
                                     'xyz@gmail.com'};
```

```
    msg.setToAddresses(toaddress);
```

```
    String[] tobccaddress = new String[]{'a1@gmail.com',  
                                         'a2@gmail.com'};
```

```
    msg.setBccAddresses(tobccaddress);
```

```
    String[] toccaddress = new String[]{'c1@gmail.com'};
```

```
    msg.setCcAddresses(toccaddress);
```

```
    msg.setSubject('Test');
```

```
    msg.setPlainTextBody('Hai Ram');
```

```

msg.setHtmlBody ('<h1> This is the link for registration </h1>');
Messaging.sendEmail (new Messaging.SingleEmailMessage []
{msg});
return null;
}
}

```

Vf page :-

```

<apex:page controller="mailgen">
<apex:form>
<apex:commandbutton value="click" action="{!!show!!}" />
</apex:form>
</apex:page>

```

→ In the above program when we want to send an Email we use the following syntax.

```

Public Messaging.sendEmailResult[] sendEmail (Messaging.Email[]
emails, Boolean allToNothing)

```

This method sends list of email objects instantiated with either singleEmailMessage or massEmailMessage.

→ In the method the boolean parameter is optional. When you give the boolean value is false, if any one of email is failed it stops processing all other emails, if boolean value is true if any one of the email is failed, it discarded only that mail and the rest of the mails processed as usual.

SendEmailMessage :-

Syntax :-

```
Public Messaging.SendEmailResult[] sendEmailMessage(List<ID>  
emailMessageIds, Boolean allOrNothing)
```

This method sends 10 draft email messages as specified by the email message Id's.

MassEmailMessage :-

We can send a mass email message to a recipient list that consists of contacts, leads, person accounts, or users you can view in Salesforce.

Messaging.MassEmailMessage :-

This class has all the methods defined in the Email class and also Email base class methods.

Methods :-

public void setDescription(String description) :- This will give the description about the mail.

public void setTargetObjectIds(ID[] targetObjectIds) :- We can add upto 250 IDs per email. If you specify a value for the target object IDs field optionally specify a whatID's as well.

public void setWhatIds(ID[] whatIds) :- The values can be any one of the following , Contact, case, opportunity and product.

Syntax:-

Messaging. MailEmail message = new Messaging. MailEmail
Message();

Example:-Apex Class :-

```
public class testemail
{
    private final List<Id> Contactids;
    public List<Contact> Con;
```

Public testemail (Apexpages. StandardController controller)

{

con = [select Id from Contact limit 250];

for(Integer i=0; i<250; i++)

{

contactids.add (con[i].Id);

}

{

public void sendEmail()

{

Messaging.MassEmailMessage mail = new Messaging.MassEmail
Message();

mail.setTargetObjectIds (contactids);

mail.setFrom

Messaging.sendEmail (new Messaging.MassEmailMessage[] {mail});

}

VF Page :-

<apex:page StandardController="contact" extensions="testemail"/>

<apex:form>

SK Aso

```
<apex:CommandButton value="Send Email" action=" {!sendEmail} "/>  
</apex:form>  
</apex:page>
```

PDF Emailfile Attachment :-

Messaging.Emailfile Attachment :-

Emailfile Attachment is used in single email message to specify attachments passed in as a part of request in other terms sending file attachments in the email.

Methods :-

Public void setfileName(String name) :- This method will set the name of the file which we attach.

public void setBody(Blob, attachment) :- this method will set the attachments, when we want to set the attachments in the Blob format, we need to use the Blob methods.

Public static Blob topdf(String) :- this method cast the Specified String into pdf format.

Public static Blob Valueof(String) :- This method cast the Specified String into Blob format.

public String toString(Blob b) :- If we give a Blob value it is converted into String.

public void setContentType(String) :- This will specify the what type of attachment Content we have added.

public void setInline(Boolean b) :- When you give true it doesn't require any action when the Content is open.

To attach a pdf file in email we have to write the code like below.

```
Pdf.getParameters().put('id', (String)acc.id);
```

```
pdf.setRedirect(true);
```

```
Blob b = pdf.getContent();
```

```
Messaging.EmailFileAttachment myem = new
```

```
Messaging.EmailFileAttachment();
```

```
Messaging.SingleEmailMessage em = new Messaging.SingleEmail  
Message();
```

```
em.setFileAttachments(new Messaging.EmailfileAttachment[]  
{myem});
```

Create a visualforce page with in a attachment pdf.

```

<apex:page StandardController="Account" renderAs="pdf">
    <apex:pageBlock>
        <apex:pageBlockSection>
            <apex:outputField value="{!account.name}"/>
            <apex:outputField value="{!account.Industry}"/>
        </apex:pageBlockSection>
    </apex:pageBlock>
</apex:page>

```

Write a program to add visualforce page as a attachment to mail in pdf format.

Apex class :-

```

public class MyAttachment {
    Account acc;
    public String myname {set;get;}
    public MyAttachment()

```

```

acc = [select id, name from Account where
       name = :myname limit 1];

```

}

```
public PageReference showc )
```

```
{
```

```
    Messaging. SingleEmailMessage mymail = new Messaging.
```

```
        SingleEmailMessage( );
```

```
    Messaging. EmailFileAttachment fmail = new Messaging.
```

```
        EmailFileAttachment( );
```

```
    PageReference pdf = page. attachmentpdf;
```

```
    pdf. getparameters(). put('id', (String) accid);
```

```
    pdf. setRedirect(true);
```

~~```
Blob b = pdf. getcontent();
```~~~~```
Blob b = pdf. getContent();
```~~~~```
fmail. setfilename ('attachment. pdf');
```~~~~```
fmail. setBody(b);
```~~~~```
mymail. setSubject ('Month Bill');
```~~~~```
String toadd = new String [ ] { 'abc@gmail.com' };
```~~~~```
mymail. setTOAddresses (toadd);
```~~~~```
mymail. setBody ('Plz find the attachment');
```~~~~```
mymail. setfileAttachments (new Messaging. EmailFile
 Attachment [] { mail });
```~~

```
Messaging. sendEmail (new Messaging. Email[] { mymail }),
```

```
return null;
```

```
}
```

```
}
```

Vf page:-

```

<apex:page Controller="MyAttachment">
 <apex:form>
 <apex:inputText value="!myname"/>
 <apex:commandButton action="!show" value="click"/>
 </apex:form>
</apex:page>

```

Sending the documents through mail

public void SetDocumentAttachments(IDE) :- This method will send the documents that we got in the Salesforce in the email using messaging concept.

Step1:- Create a document in Salesforce

1. Go to Document object click on new document.
2. Enter the name, description of the document.
3. Select External file, choose the document file & save.

Step2:- In Apex program write a query to fetch the id of the document where document name = 'Satish'.

Attach this document to single EmailMessage.

Document d = [select id from Document where name='satish'];  
Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();  
mail.setDocumentAttachments(new ID[] {d.id});

Example :-

Apex class :-

public class mailer

{

    public PageReference Show()

{

    Document d = [select id from Document where name='satish'];

    Messaging.SingleEmailMessage msg = new Messaging.SingleEmailMessage();

    msg.setDocumentAttachments(new ID[] {d.id});

    msg.set

        String[] krish = new String[] {'abc@gmail.com'};

        msg.setToAddresses(krish);

        msg.setSubject('Hai');

```
msg.setPlainTextBody('Hello');
msg.setHtmlBody('<h1>How r you </h1>');
```

Message

```
Messaging.sendEmail(new Messaging.SingleEmailMessage[]{msg});
```

```
return null;
```

```
}
```

```
}
```

VF page :-

```
<apex:page Controller="mailgen">
 <apex:form>
 <apex:commandbutton value="click" action="!show"/>
 </apex:form>
</apex:page>
```

## Inbound Email Service :-

Inbound email service is nothing but this will receive a mail from external system to Salesforce and the apex class will process the email and attachments and perform requested operation.

→ To perform this operation we got some classes defined in messaging name space.

### ) Messaging. InboundEmail class :-

This class object represents email received by the Salesforce. This class has following properties.

public String textAttachments

public String subject

public String replyTo

public String toAddress

public String plainTextBody

public String messageId

public String inReplyTo {get; set; }

This inReplyTo field of the incoming email, identifies the email or emails to which this one is rep (parent emails) contains the parent email or email message.

public String htmlBodyIsTruncated {get; set;}

This indicates the whether the html body is truncated or not.

public String CcAddress

public String fromAddress

public String fromName

## ii) Messaging. Inbound Envelope :-

The object of this class stores the envelope information (:

from address & to address) associated with inbound Email. this class has got two properties. They are

public String fromAddress

public String toAddress

Eg:- Messaging. InboundEnvelope em = new

Messaging. InboundEnvelope ();

em.fromAddress = 'xyz@gmail.com';

em.toAddress = 'abc@gmail.com';

## Headers :-

A list of RFC 2822 headers. In RFC 2822 includes following data.

- \* Received from

- \* Custom headers

- \* Message Id

Inbound Email Header class we have 2 attributes.

the object of this class refers to the header contents of RFC 2822 format. Apart from that it also has 2 attributes. They are  
name & value.

## ii) Messaging. InboundEmailResult Class:-

The object of this class is used to return the result of the email service.

NOTE :- If the object is null the result is considered to be successful. This class has 2 properties.

1) public boolean Success :- This attribute indicates whether the email received was successfully processed or not. In case if the email was not successfully processed, salesforce rejects the inbound email and sends reply to the original setup.

2) public String message :- This message stores the message written by the salesforce in the reply mail.

A59

- Any Apex class which want to process inbound email  
should implement an interface `Messaging.InboundEmailHandler`.  
This interface has a method.

global `Messaging.InboundEmailResult` handleInboundEmail  
(`Messaging.InboundEmail email,`  
~~`Messaging.InboundEnvelope envelope)`~~)

- In the above method first parameter `email` object stores  
the email received from email object to the salesforce.  
→ The second parameter stores the `to & from` address.

Eg:- Write a program to handle the inbound email and  
create a new task based on the address from which it has  
received.

global class CreateTaskEmailExample implements  
~~`Messaging.InboundEmailHandler`~~

global `Messaging.InboundEmailResult` handleInboundEmail  
(`Messaging.inboundEmail email, Messaging.`  
`inboundEnvelope envelope)`

{

`Messaging.InboundEmailResult result = new`  
`Messaging.InboundEmailResult(),`

String ename = email.fromName;

String edescription = email.plainTextBody;

String eindustry = email.Subject;

Account a = new Account(name = ename, description = edescription,  
industry = eindustry);

insert a;

{

}

Eg2:-

global class CreateTaskEmailExample implements  
Messaging.InboundEmailHandler {

global Messaging.InboundEmailResult handleInboundEmail()

(Messaging.InboundEmail email, Messaging.Inbound  
Envelope envelope)

Messaging.InboundEmailResult result =

new Messaging.InboundEmailResult();

try {

Contact vcon = [SELECT Id, Name, Email FROM Contact

WHERE Email = :email.fromAddress LIMIT 1];

A/B  
60

```
Task t = new Task(Description = email.plainTextBody,
 priority = 'Normal', status = 'Inbound Email',
 subject = email.subject, IsReminderSet = true,
 ReminderDateTime = System.now() + 1,
 WhoId = vcon.Id
);
```

```
insert t;
```

```
System.debug('New Task Object : '+new Task);
```

```
}
```

```
catch(QueryException e)
```

```
{
```

```
System.debug('Query Issue : '+e);
```

```
}
```

```
{
```

```
}
```

## Email Service:

Email Service is an automated process that use Apex classes to process the contents, headers, Attachments of Inbound Email.

NOTE:- visualforce email templates can not be used for mass emails.

- i) We can associate each email service with one or more salesforce generated email addresses to which the user can send messages for processing.

Navigation :-

Setup → Build → Develop → EmailService → click on new EmailService.

→ Give the Service name, now select the Apex class to which the mail should be forwarded to process the mail and the class should implement InboundEmail interface.

→ Select which type of attachment we have to select list of address/domains whenever we want to receive the mails.

→ Convert text to Binary fields. click on Active & save.

Steps to create Email Service :-

- 1) Go to Setup → Develop → EmailServices
- 2) click the New EmailService button.
- 3) on the EmailService page, fill the form like below.

- a) Email Service Name : sbx\_EmailAWSResponseHandler
- b) Apex class : sbx\_EmailAWSResponseHandler.
- c) Accept Attachments : None
- d) Advanced Email security settings : (unchecked)
- e) Accept Email From : (leave blank)
- f) Convert Text Attachments to Binary Attachments : (unchecked)
- g) Active : (checked)
- h) over Email Rate Limit Action : Discard message
- i) Deactivated Email Address Action : Discard message
- j) Deactivated Email Service Action : Discard message
- k) Unauthenticated Sender Action : Discard message
- l) Enable Error Routing : (unchecked)
- m) Route Error Emails to This Email Address : (leave blank)
- (e) click Save and New Email Address button.
  - a) on the Email Service Address page, fill in the form as follows.
  - b) Email Address : sbx\_emailawsresponsehandler
  - c) Active : (checked)
  - d) Contact User : (choose the user that will have full access)

To the soap box mailer app, and to all related services) <sup>AOS</sup>

- ③) Accept Email From : (leave blank)
- 5) click save.
- 6) Copy the full email address provided by Salesforce, which will be utilized in the next step.

Satish Anwala

## DML OPERATIONS &amp; GOVERNING LIMITS (62-73)

1. Introduction to DML Operations
2. Types of DML operations
3. SOQL & DML Governing limits
4. Bulkifying DML operations
5. Iterator Interface & its methods
6. Introduction to Iterable interface
7. Example to create custom Iterable

Satish Nanda

68  
1948  
Software

## DML Operations :-

The operations are

Insert

update

Delete

UpSert

There are two ways to perform DML operations.

1) By using DML Statements

Eg:- List <Account> acctlist = new List <Account>();  
 acctlist.add(new Account (Name = 'Acme1'));  
 acctlist.add(new Account (Name = 'Acme2'));  
 insert acctlist;

2) By using Database class

Eg:- List <Account> acctlist = new List <Account>();  
 acctlist.add(new Account (Name = 'Acme1'));  
 acctlist.add(new Account (Name = 'Acme2'));

Database . SaveResult [ ] sr = Database . insert (acctlist, false);

→ there is one difference between above two operations.

In the Database class method you can specify whether to allow partial processing of the records if any errors are

encountered.

By passing +

By passing the boolean value as a parameter to Database.insert.

→ If you give the parameter as true if any error occurs it doesn't allow the operation to continue.

→ If you specify false the remaining DML operations can still succeed , whereas insert in DML if any one of the record fails the total operation is discarded.

Eg :-

public pagReference show()

{

List<Account> acc = new List<Account>();

Account a1 = new Account (Name = 'Today4' , Industry = 'Banking');

Account a2 = new Account (Industry = 'Banking');

Account a3 = new Account (Name = 'Today4' ,

Industry = 'Banking');

acc.add(a1);

acc.add(a2);

acc.add(a3);

Database . SaveResult [ ] r = Database . insert (acc , true) ;

r . return null;

3

- In the above program when we give Database.insert(acc, true), if any error occurs in any one of the records a1, a2, a3 the entire operation of insert is rolledback.
- When you give Database.insert(acc, false) for above statements, if any error occurs in any one of the records a1, a2, a3 only that record is terminated and status is saved to saveResult class rest of the operations are processed normally.

### SOQL DML and loops:-

SOQL for loops iterate over all of the object records returned by a SOQL query. The syntax of a SOQL for loop is either.

```

for(variable : [soql_query])
{
 code_block
}
OR
for(variable_list : [soql_query])
{
 code_block
}

```

Eg1:-

```
String s='Bened';
for(Account a:[SELECT Id, Name from Account Where
 Name=:s])
{
 //your code
}
```

Eg2:- write a code to fetch list of accounts whose account name  
is 'Siebel' and update them by oracle.

```
List<Account> accs = [SELECT Id, Name FROM Account
 WHERE Name = 'Siebel'];
```

//Loop through the list and update the Name field

```
for(Account a:accs)
```

```
{
```

```
a.Name = 'Oracle';
```

```
}
```

//update the database

```
update accs;
```

Eg3:- write a code to fetch list contacts for a given account &  
update Contact email with account email id.

```
Account acc=[select email,(select id, email_c from contact)
 from Account where Name='Sam' limit 1];
```

Agg

List<Contact> con = acc.contacts;

List<Contact> myContacts = new List<Contact>();

for(Contact x : con)

{

x.email = acc.emailId - c;

myContacts.add(x);

}

update myContacts;

Eg4:- Write a code to fetch all the transaction records which are created yesterday and delete them.

List<Transaction\_c> tran = [select id from Transaction\_c  
where createddate = YESTERDAY];

delete tran;

Eg5:- Write a code to create new student records by fetching the data from college object where branch is CSE.

List<Student\_c> std = new List<Student>();

List<College\_c> colg = database.query('select name, Branch\_c,  
Year\_c from college\_c where branch\_c = 'CSE');  
for(College\_c : colg)

{

Student\_c s = new Student\_c();

s.name = c.name;

```
S.Branch_c = c.Branch_c;
Std.add(s);
```

{

```
insert std;
```

How to Count number of DML statements in a transaction:-

Eg1:-

```
Account a = new Account (Name = 'Sam', Industry = 'Banking');
insert a;
```

```
Account a1 = new Account (Name = 'Sam', Industry = 'Banking');
insert a1;
```

Eg2:- void show()

{

```
Account a = new Account (Name = 'Sam', Industry = 'Banking');
insert a;
```

```
Account x = [select id, name from Account limit 1];
```

```
x.name = 'Sam';
```

```
update x;
```

```
Account y = [select id, name from Account limit 1];
```

```
delete y;
```

}

Eg3:- void showc()

{

for (integer i=1; i<=100; i++)

{

Account a = new Account (Name = 'Sam', Industry = 'Banking');

insert a;

}

}

NOTE:- In a single transaction we can not make more than 150 DML statements.

Eg4:- void showc()

{

for (integer i=1; i<=160; i++)

{

Account a = new Account (Name = 'Sam', Industry = 'Banking');

insert a;

}

In the above transaction Show we made 160 DML statements. so we get an errors like below.

System.LimitException : Too many DML statements: 151

Error is in expression '{!showc}' in page dmlStatementCount

## Bulkify the DML operation:-

When we are performing DML operations on any record we have governance limits of 150 records for every statement.

→ To support the large no. of DML statements in a single transaction without bypassing the governance limits. To meet this requirement we are bulkifying the operation by adding all the records to a list and calling the DML on the list.

NOTE :- If you call DML operation on the List object it is considered as single DML statement. This operation's called bulkifying operation.

Governance limit :- In a single transaction we can perform DML operations on maximum 10,000 rows. If you perform more than 10,000 rows it gives an exception like below.

System.LimitException: Too many rows (1000).

Ex:- List<Account> acc = new List<Account>();  
for(Integer i=1; i<=4000; i++)  
    {  
        }

Account a = new Account(name = 'elctoday',  
                          industry = 'Banking');  
acc.add(a);

```

insert acc;
List<Customer__c> acc1 = new List<Customer__c>();
for(Integer i=1; i<=6002; i++)
{
 Customer__c a = new Customer__c (Customer_Name__c =
 'customer');
 acc1.add(a);
}
insert acc1;
return null;
}

```

### Interface in Apex:-

Interface is a collection of unimplemented methods. This will specify the signature of the method, types of inputs that we pass the method specifies what type is given as an output.

NOTE:- Generally the interface methods are give it as global.

Syntax:- public interface Exam

{
 Account Show(String s);

}

All the methods in the interface are public and abstract methods. Interface also contains final static data members.

→ The class which is implementing this interface should define all the methods in the interface.

Ex: interface test

```
public String show();
void disp();
```

global class Example implements test

```
public String show()
{
 return 'sum';
}
public void disp()
{
}
```

```
Account a = new Account (name = 'sum');
```

→ If you don't define any methods in the interface it gives an error. we can create an object for class.

class Test

{

}

Test t = new Test();

→ interface Demo

{

Void show();

}

To interface we can not create an object.

Demod = new Democ(); is not allowed. Bcoz interface

Contains only unimplemented methods and static methods.

→ class Exam implements Demo

{

public Void show()

{

}

g

Demod d = e; is allowed. Bcoz 'e' is the object of the class  
which is implementing the demo interface.

NOTE: Interface reference variable can store the object of  
the class which has successfully implemented interface.

In the above scenario

demo dt; is not allowed. Bcoz test class is not implemented interface demo.

Interface Iterator<AnyType>:-

this is the interface defined by the salesforce in apex. It has 2 methods.

1. Boolean hasNext()

2. AnyType next()

1. Boolean hasNext() :- this method returns true if there is another item in the collection being traversed else returning false.

2. AnyType next() :- Returns the next item in the collection.

Both the methods given above should be declared as a global or public.

NOTE :- we can use only custom iteratos in while loop, it is not supported for loop.

Write a program to create Custom iterator:-

class satish implements Iterator<Account>

{}

A69

public List<Account> acc;

public Integer i;

public satisfies()

{

acc = [select name from Account];

i = 0;

{

public boolean hasNext()

{

if (i > acc.size())

return false;

else

return true;

{

public Account next()

{

if (i == 0)

return null;

i++;

return a[i - 1];

{

{

NOTE :-

satish s = new satish();

Iterator<Account> it = s;

since satish class has implemented iterator interface we can store the object of the satish class in Iterator it.

Interface Iterable :-

Iterable is a interface in which we have a method

Iterator<class> iterator()

Eg:-

class Bened implements Iterable<Account>

{

public Iterator<Account> iterator()

{

Satish s = new Satish(); // satish class is implementing

Iterator<Account> it = s;

return it;

}

interface Iterator, so we  
can store this object in  
interface iterator.

{

Eg2:- class Example

{

```
List<Account> acc;
```

```
public Example() {
```

{

```
acc = new List<Account>();
```

{

```
public PageReference show()
```

{

```
Bened b = new Bened();
```

```
Iterator<Account> it = b.iterator();
```

```
//satisfy s = b.iterator();
```

```
while (it.hasNext()) {
```

{

```
Account a = (Account) it.next();
```

```
acc.add(a);
```

```
return null;
```

{

{

Eg 3:

global class SatisfIterator implements Iterator<Account>

{

public List<Account> acc;

public Integer i;

public SatisfIterator()

{

acc = [select name from Account limit 5];

i = 0;

{

public boolean hasNext()

{

if (i > acc.size())

return false;

else

return true;

{

public Account next()

{

if (i == 5)

return null;

i++;

return acc[i - 1];

{

{

Eg4:-

public class BankIterable implements Iterable<Account>

{

public Iterator<Account> iterator()

{

return new SatisfIterator();

{

{

Eg5:-

public class ExampleIterable

{

public List<Account> acc;

public List<Account> getAcc()

{

return acc;

{

public ExampleIterator()

{

acc = new List<Account>();

{

public pageReference Show()

{

```
Iterable<Account> my = new BenedIterable();
```

```
Iterator<Account> test = my.iterator();
```

```
while (test.next() != null)
```

```
{
```

```
 Account a = test.next();
```

```
 acc.add(a);
```

```
}
```

```
return null;
```

```
{
```

```
}
```

If page :-

```
<apex:page Controller="ExampleController">
```

```
 <apex:form>
```

```
 <apex:commandButton value="click" action="!show?"/>
```

```
 <apex:pageBlock>
```

```
 <apex:pageBlockTable value="!acc?" var="a">
```

```
 <apex:column value="!a?." />
```

```
 <apex:pageBlockTable>
```

```
 </apex:pageBlock>
```

```
</apex:form>
```

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

## Database.QueryLocate Class:-

Database.QueryLocate class stores the record set return by the database.getQueryLocate.

### Methods:-

There are 2 methods in the QueryLocate class.

⇒ getQuery():

Syn: public String getQuery()

This method returns the query used to instantiate the Database.QueryLocate object.

This is very much useful when testing the start method.

Eg: Database.QueryLocate dq = Database.getQueryLocate  
 ([Select name from Account]);

String str = dq.getQuery();

// str = 'Select name from Account';

⇒ iterator():

Syn: public Database.QueryLocateIterator iterator()

This will return new instance of queryLocate Iterable.

Eg: List

Eg:-

List<Account> acc = new List<Account>();

Database.QueryLocators dq = Database.getQueryLocators();

Database.QueryLocatorsIterable iq = dq.iterator();

while(iq.hasNext())

b

Account a = (Account) iq.next();

acc.add(a);

c

Satish Mwia

## BATCH APEX (74-83)

1. Introduction to Batch Apex
2. Introduction to Batchable interface index methods
3. Difference Between Database.Query Locatable & Iterable
4. Example programs for Batch Apex
5. Practical Scenarios on Batch Apex
6. Steps to Execute Batch Apex
7. Database.Stateful interface
8. Database.allowCallouts interface
9. Governing Limits & Limitations
10. Introduction to Database-Batchable Context & Asynchronous jobs

examples

## Batch Apex :-

Batch Apex allows you to define a single job that can be broken up into manageable chunks, where every chunk can be processed separately.

Eg:- If you need to make a field update of every record of Account object in your organization, then we have governing limits that would restrict us from achieving the above task.

Reason:- In a single transaction we can process only 10,000 records. Now, in the above case if we have more than 10,000 records in the organization then we can not perform this field update.

Batch Apex:- In the Batch Apex it will fetch all the records on which you want to perform the field update and divide them into list of 200 records and on every 200 records operation is performed separately.

→ This would help us to execute on more than 10,000 records as, it won't perform an operation on all the records in a single transaction instead it dividing them into no. of

subtasks where each subtask may contain the records upto 2000.

### Database.Batchable Interface :-

To use the Batch Apex concept the Apex class should implement Database.Batchable interface.

→ Database.Batchable interface consists of 3 methods, that must be implemented.

1. Start() method

2. execute method.

3. finish method.

i. Start method :- Start method is automatically called at the begining of the batch apex job.

→ This method will collect the records of objects on which the operation should be performed.

→ This records are broken down into subtasks and given to execute method.

### Syntax:-

```
global (Database.QueryLocator | Iterable<Object>)
```

```
start(Database.BatchableContext bc) { }
```

→ the return type of the Start method can be

i) Database.QueryLocator  
(D)

ii) Iterable<Object>

i) Database.QueryLocator :-

use Database.QueryLocator as return type to the start method when you are fetching the records using a simple select query.

NOTE:- The governing limit says

i) Total number of records retrieved by SQL queries 50,000.

ii) Total number of records retrieved by  
Database.QueryLocator 10,000.

But, in the Batch Apex the governing limits for SQL query are bypassed and we can fetch upto 50 million records using Database.QueryLocator.

Example:-

```
global Database.QueryLocator start(Database.BatchableContext bc)
```

{

```
 return Database.getQueryLocator('select id, name from Account');
```

}

Eg2: Write a start method to fetch all customer Name, AccountType, from customer object whose AccountType is 'Saving'.

global Database.QueryLocator start(Database.BatchableContext bc)

{  
String sql = 'select id, customer\_name\_c, Account\_Type\_c  
from customer\_c where Account\_Type\_c  
= '+ 'saving';

ii) Iterable :- use Iterable as a return type when you want to use complex logic & scope to fetch the records for the batch job.

Eg: global class MyRet implements Iterable<Account>

global Iterable<Account> start(Database.BatchableContext bc)

{  
return new MyRet(); // This should return object of the class that has implemented iterable interface.

## 2) execute method :-

The records which are fetched from the Start method are divided into batches of 200 records each. Now every batch of 200 records are separately pass to the execute method separately and the operation what we want to perform on these records are written in this execute method.

### Syntax :-

global void execute (Database.BatchableContext BC, List<P>) { }

This method takes two parameters.

→ A reference to the Database.BatchableContext object.

→ A list of objects such as List<sobject>, or a list of parameterized types.

i.e, set of 200 records which are passed to the execute method are stored in this list.

## 3) finish method :-

When the Start method has fetched 1000 records on which we want to perform the operations they are divided into 5 batches/Groups of size 200.

- on every batch ~~in~~ the group the execute method is called.
- which means execute method is called 5 times.
- After executing every group/batch the governing limits are reset for the execute method.
- once all the batches are executed then finish method will be called to send email notification of post execution work.

NOTE:-

Syntax:-

global void finish(Database.BatchableContext BC){ }

NOTE:- All the 3 methods of the Batchable interface refers to Database.BatchableContext object. Where this object is used to track the progress of the batch job.

Example:- Write a Batch Apex program to update Account Name with 'in' the Account object with Suffix Ms. before the AccountName.

Example:- Write a batch apex program to update Account name in the Account object with suffix 'Mo' before the Account Name.

```
global class AccountBatch implements Database.Batchable<Account>
```

{

```
 global Database.QueryLocator Start(Database.BatchableContext bc)
```

{

```
 String query = 'Select id, name from Account';
 return Database.getQueryLocator(query);
 }
```

```
 global void execute(Database.BatchableContext bc,
```

List<Account> scope)

{

```
 List<Account> acc = new List<Account>();
```

```
 for (Account a:acc)
```

{

```
 a.name = 'Mo.'+a.name;
```

```
 acc.add(a);
```

}

```
 update acc;
```

}

```
global void finish(Database.BatchableContext bc)
```

```
{}
```

```
}
```

```
{}
```

Example 2:- Write a batch apex program to phone no. in the Contact object with the phone no. in the corresponding Account Object. where Contact is a child of Account.

```
global class ContactUpdate implements Database.Batchable<Contact>
```

```
{}
```

```
global Database.QueryLocator start(Database.BatchableContext bc)
```

```
{}
```

```
String query = 'select id, phone, Account.phone from
Contact';
```

```
return Database.getQueryLocator(query);
```

```
{}
```

```
global void execute(Database.BatchableContext bc,
List<Contact> scope)
```

```
{}
```

```
List<Contact> con = new List<Contact>();
```

```
for(Contact c:con)
```

```
{}
```

c.phone = c.Account.phone;

Con.add(c);

}

update con;

}

global void finish(Database.BatchableContext bc)

}

}

}

How to invoke batch apex job (or) how to execute the batch apex job programmatically.

We can use Database.executeBatch() method to programmatically begin the batch job.

Syntax:-

public static ID executeBatch(Sobject className)

public static Id executeBatch(Sobject className, Integer scope)

→ The above two methods are static methods of database class. We can use any one of the method to execute

the batch job.

NOTE:- the class name what we are passing to the Database.executeBatch() method should be object of the class which has implemented Database.Batchable interface.

Order of execution of Batch Apex Job when we invoke through Database.executeBatch() :-

Step1:- Create a object for the class which has implemented Database.Batchable interface.

Step2:- pass this object which you have created in the first step as a parameter to the Database.Batchable interface.

Step3:- When database.executeBatch() method is called it will add the batch job to the queue.

Step4:- Once the resource is available in the queue automatically Start() method will be invoked and it will collect all the records which we need to perform the operation.

Step5:- All the records that are fetched from the Start method are divided into small group/batch of size given in the

`Database.execute()` method.

NOTE:- In case if you don't specify the size by default it takes 200.

Step 6:- On every batch of records execute method will be invoked.

Step 7:- Once all the batches are executed, then finish method will be called.

NOTE:- 1) `Database.execute` method is going to return id of the batch job. using which we can monitor & abort the operation.

2) All the asynchronous jobs are stored to 'AsyncApexJob' object, from this we can monitor, no of jobs processed for our asynchronous job and status of the job.

Example:-

```
ID batchprocessid = Database.executeBatch(reassign);
```

```
AsyncApexJob aa = [select id, status , jobitems processed
from AsyncApexJob where
ID = :batchprocessid];
```

## Scenario :-

1. Create a visualforce page with Input text button.
2. When we click on the Replace button in the custom visualforce page.
3. Fetch All the records in the Customer object with the Customer name matching with name given in the visualforce page.
4. Update their Account Type as 'saving' account.

Step1: Create a batch apex class to perform update of the Account Type.

global class CustomerBatch implements Database.Batchable<Customer>

{

    public String myname;

    global CustomerBatch(String myname)

{

        this.myname = myname;

}

    global Database.QueryLocator Start(Database.BatchableContext BC)

{

```
return Database.getQueryLocator('select id, Account_Type__c
 from Account where name = '+myname);
```

{}

```
global void execute(Database.BatchableContext bc,
 List<Customer__c> scope)
```

{}

```
List<Customer__c> cust = new List<Customer__c>();
```

```
for(Customer__c c : cust)
```

{}

```
c.Account_Type__c = 'saving';
```

```
cust.add(c);
```

{}

```
update cust;
```

{}

```
global void finish(Database.BatchableContext bc)
```

{}

```
Messaging.SingleEmailMessage myemail = new
```

```
Messaging.SingleEmailMessage();
```

```
String[] toadd = new String[] {'abc@gmail.com'};
```

```
myemail.setToAddresses(toadd);
myemail.setSubject('Batchprocessed');
myemail.setPlainTextBody('Batch completed successfully');
Messaging.sendEmail(new Messaging.Email[] {myemail});
```

Step 2: Create Apex class to invoke the batch apex job.

```
public class TestMybatch
```

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

```
 public PageReference Show()
```

```
{CustomerBatch mybatch = new CustomerBatch(cname);
```

```
ID = id = Database.executeBatch(mybatch, 400);
```

```
System.debug('My job id '+id);
```

```
}
```

Step 3: Create a visualforce page to call the TestMybatch class

~~A75~~  
`<apex:page controller="TestMyBatch">`

81

`<apex:form>`

`<apex:outputLabel> Enter name </apex:outputLabel>`

`<apex:inputText value="! myname!"/>`

`<apex:commandButton value="click" action="! $show!"/>`

`</apex:form>`

`</apex:page>`

## Database. Stateful :-

Each execution of batch apex job is considered as discrete transaction. Which means when you have a batch job with 1000 records executed with optional scope of 200 records then

↳ 5 batch      Batch1 → 1-200

Batch2 → 201-400

Batch3 → 401-600

⇒ When we call execute on batch1 to summarize the value

Integers sum = 0;

public void execute(Database.BatchableContext bc, List<Account> scope)

{

for(Account a : scope)

sum = sum + a.AnnualRevenue;

}

→ first Batch of 1-200 records call the execute method : with 200 records.

Before calling execute() : sum=0

After calling execute() : sum = 30000 (assume it)

→ When the execute() is called on batch 2 of records 201-400 then  
Initial value of sum again set to zero.

Before calling execute() on Batch1 : 1-200 records : sum=0;

After calling execute() on Batch1 : 1-200 records : sum=30000

Before calling execute() on Batch2 : 201-400 records : sum=30000

When we call execute() on Batch2 : 201-400 records.

First sum is set =0 again : sum=0.

then again first summary value is calculated again.

→ Which means the state of the batch is forwarded from one  
execute() to another execute().

→ If you specify Database.Stateful in the class definition,  
you can maintain State across these transactions. This is  
useful for counting or summarizing records as they are  
processed. For eg. Suppose your job processed opportunity  
records. You could define a method in execute to aggregate

totals of the Opportunity amounts as they were processed.

global class SummarizeAccountTotal implements

Database.Batchable<Subject>, Database.Stateful

{

global final String query;

global integer summary;

global SummarizeAccountTotal(String q)

{

query = q;

summary = 0;

}

global Database.QueryLocator start(Database.BatchableContext BC)

{

return Database.getQueryLocator(query);

}

global void execute(Database.BatchableContext BC, List<Subject> scope)

{

for(Subject s:scope)

{

summary = Integer.valueOf(s.get('total\_c')) + summary;

}

}

```
global void finish(Database.BatchableContext BC)
```

{

}

{

### Governing Limits :-

1. only one batch apex job's Start method can run at a time in an organization.
2. up to 5 queued or active batch jobs are allowed for apex.
3. the maximum number of batch apex method executions per a 24-hour period is 2,50,000.
4. the batch ApexStart method can have up to 15 query cursors open at a time per user.
5. A maximum of 50 million records can be returned in the Database.QueryLocator object.
6. the Start, execute, and finish methods can implement up to 10 callouts each.

NOTE :- If we have 1000 records with scope of 200 records then they are divided into 5 batches.

so execute() method is called 5 times. Which means in every execute() we call 10 callouts. so in this scenario we call

start() → 10 callouts

execute() →  $10 \times 5 = 50$  callouts

finish() → 10 callouts

70 callouts in entire operations

### Limitations :-

- Methods declared as future aren't allowed in the classes that implement Database.Batchable interface.
- Methods declared as future can't be called from Batch Apex class.
- For sharing & recalculation, we recommend that the execute method delete and then re-create all Apex managed sharing for the records in the batch.
- For each 10,000 AsyncApexJob records, Apex creates one additional AsyncApexJob record of type BatchApexWorker for internal use.

## SCHEDULE APEX (84-98)

1. Introduction to schedulable Apex
2. Methods in Database.Schedulable interface
3. Schedulable Apex Examples
4. Creating Time frame
5. Executing the Schedulable Apex
6. Governing limits.
7. Real-time scenarios

~~INTERFACE IN APEX & Custom ITERABLE(91-98)~~

*Satish Mania*

## Apex Scheduler :-

It will invoke the Apex class to run at specific time.

Anybody who want to schedule their class they have to implement Schedulable interface.

### Schedulable interface :-

The class that implements this interface can be scheduled to run at different intervals. This interface has several methods they are

~~public void execute(SchedulableContext sc)~~

Eg: public class Myschedule implements schedulable

{

public void execute(SchedulableContext sc)

{

Account a = new Account(Name = 'Faraz').

insert a;

}

}

- The scheduler will run in systemcontext, which means all the classes are executed whether the user has permission or not.
- We can monitor or stop the execution of scheduled apex

job using salesforce legacy interface from setup.

### Navigations:-

Setup → Monitoring → Jobs → Scheduled Jobs

→ the execute() method must be declared as public or global.

→ using System.Schedule.

### System.Schedule :-

once you are implemented schedulable interface use  
System.Schedule method to execute the class.

System.Schedule() method takes 3 parameters.

1) Name of the job.

2) Expression that is used to represent time and date of the operation.

3) the object of the class which you want to execute

> Expression is written in the form of 'Seconds, minutes, hours, day of the month, 1st month day of the week, optional year.'

'seconds min hours DAY-Month Month DAY-Week optional year'

0-60 0-60 0-24 1-31 1-12 1-7

## special characters!

85 ~~A79~~

? :- specifies no specific value. This is only available for day of the month and day of the week.

Eg1:- Write the expression to schedule an operation 10th of August at 12:30 PM.

Ans:- '0 30 12 10 ? ?'

'0 30 12 10 Aug ?'

'0 30 12 10 Aug ? 2013'.

Eg2:- Write an expression to schedule an operation on 10th Monday 12:30.

Ans:- '0 30 12 ? 1 MON'

'0 30 12 ? 1 2'

\* :- specifies all the values.

Eg:- Write the expression to schedule an operation on every day of the Aug at 12:30 PM.

'0 30 12 \* AUG ?'

'0 30 12 \* 8 ?'

Eg:- Expression to schedule on every hour on 10th Aug.

'0 30 \* 10 ? ?'

L :- specifies the end of the range. This is available only day of the month or day of the week.

Eg:- Write an expression to schedule the operation on last Friday of the March at 10:20.

'O 20 10 ? 3 6L'.

W :- specifies nearest weekday of the given day. This is available for only day of the month.

Eg:- If we specifies 20W and 20th is a Saturday so the class runs on 19th. If i give 1W and 1st is a Saturday then it runs on Monday not on previous month.

Eg:- Write an expression to schedule an operation on nearest weekday of March 20th at 10:20.

'O 20 10 20W 3 6L'

This is to specify the last working day of last week day of last of month.

# :- this specifies nth day of the month.

Week-Day # Day-Month

2#2 it will run Monday of every 2nd month.

Q:- JAN, MAR means JAN-MAR.

86 ~~180~~

→ If you want to schedule any operation we have to create an object for the class which has implemented for schedulable interface.

class Myschedule implements Schedulable

{

    public void execute(SchedulableContext sc)

{

{

{

        Myschedule my = new Myschedule();

        String sto = '0 0 10 \* 3 ?';

        System.schedule('myjob', sto, my);

            ↓      ↓      ↓  
            Name   Time   Object.  
            format

System.scheduleBatch() :-

System.scheduleBatch() is used to run a schedule a batch job only once for a future time. This method has got 2 parameters.

Param1 :- Instance of a class that implements Database.Batchable interface.

Param2 :- Job name.

Param3 :- Time interval after which the job should start execute.

Param4 :- It's an optional parameter which will define the no.of that processed at a time.

The System.ScheduleBatch() returns the scheduled job Id (CronTrigger id).

→ We can use this job id to abort the job.

Example :- you have to first implement the Schedulable interface for the class then specify the Schedule using Schedule Apex page or Apex class :- system.schedule method.

global class purge implements Schedulable

{

global void execute(SchedulableContext sc)

{

List<Credit\_Card\_\_c> creditcard = new List<Credit\_Card\_\_c>();

for

```
for(credit_card_c cc:[select Id, Name from credit_card_c
 where Age_c > 2])
```

{

```
 creditCard.add(cc);
```

```
 cc.ch_c = '0000';
```

{

```
Database.update(creditCard);
```

{

{

Eg:- If we want to schedule

- i) Create an object for the class which has implemented the schedulable interface.
- ii) Create the time frame.
- iii) Invoke the System.Schedule method with job name, Schedule object, timeframe.

global class MyBatch implements

) global class MyBatch implements Database.Batchable<Subject>

{

global Database.QueryLocator start(Database.BatchableContext bc)

{

return Database.getQueryLocator('select id, name from Account');

}

global void execute(Database.BatchableContext bc,

List<Subject> scope)

{

List<Account> acc = new List<Account>();

for (Subject x: scope)

{

Account a = (Account)x;

a.name = 'Nb' + a.name;

acc.add(a);

{

update acc;

}

global void finish(Database.BatchableContext bc)

{

{

}

i) global class Myschedule implements schedulable

```

 {
 global void execute(SchedulableContext sc)
 {
 MyBatch mb = new MyBatch();
 Database.execute(mb);
 }
 }

```

ii) global class Textschedule

```

 {
 public PageReference Show()
 {
 String timeframe = '0 10 & 10 * ?';
 Myschedule ms = new Myschedule();
 System.schedule('myjob', timeframe, ms);
 }
 }

```

Vf page:-

```

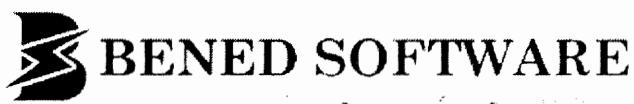
<apex:page Controller='Textschedule'>
 <apex:form>
 <apex:CommandButton value="click" action="={!Show}"/>
 </apex:form>
</apex:page>

```

## Schedulable Apex limitations:-

- We can schedule only 100 jobs at a time.
- Max. no. of apex schedule jobs in 24 hours is 2,50,000 number of jobs.
- Synchronous Web service callouts are not supported in Schedulable Apex.

Satish Minala



## Real Time Scenario's and UseCases

### Real Time Scenario 1:

#### Database.Stateful Interface

I have batch apex code updating two fields in Opportunity line Item(OLI). While I run the code for complete batch the values in the field are updating correctly. However, If I split the batch apex in two or many, I am facing the issue of the last batch value getting updated in Opportunity line item instead of complete one.

for ex:- Consider 3 record of value 50 each, I am expecting the field in OLI to be updated by 150 and it is updating rightly while I run the batch for `Database.executeBatch(job,3);` where as if I split the same as `Database.executeBatch(job,2);` then the field is getting updated by 50 instead of 150.

```
global class SummarizeAccountTotal implements Database.Batchable<sObject>, Database.Stateful {
```

```
 global final String query;
```

```
 global Map<Id, Account> accountmap;
```

```
 global SummarizeAccountTotal() {
```

```
 accountmap = new Map<Id, Account>();
```

```
}
```

```
 global Database.QueryLocator start(Database.BatchableContext BC) {
```

```
 return Database.getQueryLocator(query);
```

```
}
```

```
 global void execute(Database.BatchableContext BC, List<sObject> scope) {
```

```
 List<Opportunity> ops = (List<Opportunity>) scope;
```



```
for (Opportunity o : ops) {

 if (accountmap.containsKey(o.AccountId)) {

 Account a = accountmap.get(o.AccountId);

 a.Test_Amount__c += o.Amount;

 accountmap.put(o.AccountId, a);
 }

 else{

 accountmap.put(o.AccountId, new Account (Id = o.AccountId, Test_Amount__c = o.Amount))

 }

}

}

global void finish(Database.BatchableContext BC){

 try {

 update accountmap.values();
 }

 catch (Exception Ex) {

 system.debug(Ex);
 }

}
```



## Allow Callout interface Scenario with example

I had to integrate a client's Salesforce and NetSuite System. The requirement was to implement a web-service that synchronises the financial-data between two systems everyday. In dev terms, the client's Salesforce has to make a GET request to the NetSuite System, parse the returned JSON response and save the data in respective Salesforce fields. When I looked into this, I found there was an Apex Governor Limit of 10 HTTP callouts in any context, i.e., you cannot make more than 10 HTTP request in one session. For the current requirement, I had to make around 3500 calls a day.

So after a lot of digging, I figured out that the workaround to this problem is using Batch Apex and implementing HTTP requests inside the same. Batch Apex was introduced to empower the developers to be able to build complex, long-running processes on the Force.com platform. By inserting the HTTP request in the same and it works like a background web-service that runs asynchronously and needs no supervision. Sounds sweet, right?? So let's get started.

To use batch Apex, you must write an Apex class that implements the Salesforce-provided interface *Database.Batchable*, and then invoke the class programmatically. In order to make the callouts, you also have to include *Database.AllowsCallouts* in class definition.

Let us first create an Apex class the  
implements *Database.Batchable* and *Database.AllowCallouts*.

```
global class BatchSync implements Database.Batchable<sObject>, Database.AllowsCallouts {
 public String query = 'Select ID, Name from Account';
 global Database.QueryLocator start(Database.BatchableContext BC) {
 return Database.getQueryLocator(query);
 }
 global void execute(Database.BatchableContext BC, List<Account> records) {
 String endpoint;
 for (integer i = 0; i < records.size(); i++){
 try {
 HttpRequest req = new HttpRequest();
 HttpResponse res = new HttpResponse();
 Http http = new Http();

```

```

// Set values to Params

endpoint = 'Your endpoint';

req.setHeader('Authorization', header);
req.setHeader('Content-Type', 'application/json');
req.setEndpoint(endpoint);
req.setMethod('POST');
req.setBody('Information you wanna send');
req.setCompressed(true); // This is imp according to SF, but please check if
 // the webservice accepts the info. Mine did not :P
 // Had to set it to false

if (!Test.isRunningTest()) {
 res = http.send(req);
 String sJson = res.getBody();
 System.debug('Str:' + res.getBody());
}
// now do what u want to with response.
}
catch (Exception e) {
 System.debug('Error:' + e.getMessage() + 'LN:' + e.getLineNumber());
}
}

global void finish(Database.BatchableContext BC){
 //you can also do some after-the-job-finishes work here
}
}
}

```

You can query any valid SF sObject in *Database.QueryLocator*. In this example, you can use the **records** Account object to use any data-field from Accounts and even perform DML operations.

Now, to execute the above Batch Job programatically, you need to use *Database.executeBatch* method. The *Database.executeBatch* method takes two parameters:

1. An instance of a class that implements the *Database.executeBatch* interface.
2. The *Database.executeBatch* method takes an optional parameter *scope*. This parameter specifies the number of records that should be passed into the *executemethod*. Use this parameter when you have many operations for each record being passed in and are running into governor limits.

Q\



Test Class:

```
@isTest
public with sharing class TestBatchSync {

 static testmethod void m1() {

 List<Account> accns = new List<Account>();
 for(integer i = 0; i<10; i++){
 Account a = new Account(Name='testAccount'+i');
 accns.add(a);
 }

 Test.StartTest();
 insert accns;
 BatchSync sync = new BatchSync();
 sync.query = 'Select ID, Name from Account';
 ID batchprocessid = Database.executeBatch(sync);
 Test.StopTest();
 }

}

@isTest
private class TestScheduleSync {
 static testmethod void testSync() {

 Test.StartTest();
 List<Account> accns = new List<Account>();
 for(integer i = 0; i<10; i++){
 Account a = new Account(Name='testAccount'+i');
 accns.add(a);
 }

 String CRON_EXP = '0 0 23 * * ?';

 insert accns;
 scheduleDailySync sync = new scheduleDailySync();

 // Schedule the test job
 String jobId = System.schedule('testBasicScheduledApex', CRON_EXP, sync);
 }
}
```

```
// Get the information from the CronTrigger API object
CronTrigger ct = [SELECT Id, CronExpression, TimesTriggered, NextFireTime
FROM CronTrigger WHERE id = :jobId];

// Verify the expressions are the same
System.assertEquals(CRON_EXP, ct.CronExpression);
System.assertEquals(0, ct.TimesTriggered);
System.assertEquals('2022-09-03 00:00:00', String.valueOf(ct.NextFireTime));

Test.StopTest();
}

}
```

### Real Time scenario where you can use Batch apex:

#### Business Use Case 1 :

We use batch apex for bulk processing. Let's say you want to update thousands to accounts on a regular basis or (even if just one time) and real time update is not a MUST HAVE requirement. You know how you are bound by governor limits for all the retrievals , updates , inserts.

Batch class has much wider governor limits . Its an async operation and you can run a nightly job and process all these records without hitting limits.

#### Examples :

I used batch class for a scenario where I had to send email to all contract owners 15 days before payment due date. There were thousands of contracts nad at one time there could be many contract owners meeting this criteria. Also, apart from sending email ,I had to update a flag on contract. So, I schedules a daily nightly batch job for this.

#### Business Use Case 2:

An organization's sales team is using Salesforce CRM primarily to manage their Leads, Contacts, and opportunities. This is akin to a retail setting whereby there are multiple physical locations and each location has its own sales team and sales manager (think car dealership). In this setting, Leads (and/or Contacts) enter through multiple channels such as internet marketing, phone calls, walking through the front door, etc and are assigned to a sales rep based on a round-robin format. That sales rep becomes the owner of the Lead, but only has protection for a certain period of time before the Lead is redistributed.



The protected period of time is based on the sales rep's last activity to the Lead/Contact. In other words, if the rep is not actively working the Lead/Contact then they lose protection, and if that Lead/Contact ends up buying a product then the rep will not be credited for the commission.

The question then became, how can we automate this process of removing Lead protection based on the sales rep's activity? One of the great features of working with the Force.com platform is that there are always multiple ways to solve these types of problems. And these are good problems to solve because the solutions free up people's time and help advance business. In this situation, the basic design for this scenario is that when the number of days since the last activity is greater than 5 days, revert the ownership of the Lead/Contact from the sales rep to the sales manager. This lets the sales manager redistribute the Lead/Contact to another sales rep.

The first step was to create a new formula field that returns a Number and call it Days\_Since\_Last\_Activity. The formula is this: Today() - LastActivityDate. At this point, you might think about creating a Workflow Rule that says when Days\_Since\_Last\_Activity is greater or equal to 5 then Update the Owner field. However, a Workflow Rule is only fired under three circumstances: a) Only when a record is created; b) Every time a record is created or edited or; c) When a record is edited and did not previously meet the criteria. This means that the workflow rule won't get fired until the record gets edited, and we want the ownership to change immediately upon the passing of the protected period regardless of whether or not someone edits the record.

The solution in this case is to use Batch Apex to query the database for all Leads/Contacts (we use Contacts in the example code) that have crossed the protected period, i.e. the Days\_Since\_Last\_Activity fields is greater than 5. Then, we reassign ownership to the correct sales manager based on which physical location this Contact is associated with. Next, we create a new Task associated with this Contact so that the Days\_Since\_Last\_Activity gets reset. Lastly, we schedule this Batch Apex to run each night so that the ownership is being recalculated on a daily basis.

Below are the Apex and Test Classes that make this work.

```
global class UpdateAllContacts implements Database.Batchable<sObject> {
```

```
 //This is the query that is passed to the execute method. It queries all of the Contacts who
 //have passed
 //the protected period.
```



```
String query = 'Select Id, Club_Location__C, OwnerId FROM Contact WHERE
Days_Since_Last_Activity__c > 5';

global database.queryLocator start(Database.BatchableContext BC) {
 return database.getQueryLocator(query);

} //close start method

global void execute(Database.BatchableContext BC, list <Contact> scope) {

List <Task> taskList = new List<Task>();

// Iterate through the whole query of Contacts and transfer ownership based on the location.
// This example only has two locations.
// Create a Task that's associated with each Contact. This resets the Days Since Last
Activity formula field.
for(Contact c : scope) {
 if(c.Location__c == 'Location 1') {
 c.OwnerId = 'XXXXXXXXXXXXXXXXXX';
 Task tsk = new Task();
 tsk.WhoId = c.Id;
 tsk.ActivityDate = System.today();
 tsk.Status = 'Completed';
 tsk.Subject = 'Ownership Transferred';

 taskList.add(tsk);

 } //close if statement
 else {
 c.OwnerId='XXXXXXXXXXXXXXXXXX';
 Task tsk = new Task();
 tsk.WhoId = c.Id;
 tsk.ActivityDate = System.today();
 tsk.Status = 'Completed';
 tsk.Subject = 'Ownership Transferred';

 taskList.add(tsk);
 } //close else
} //close for-loop

try {
 insert taskList;
} catch (system.dmlexception e) {
 System.debug('Tasks not inserted: ' + e);
}
```



```

try {
 update scope;
} catch (System.Dmlexception e) {
 System.debug('Scope not updated: ' + e);
}

} //close execute method

global void finish(Database.BatchableContext BC) {

 AsyncApexJob a = [Select Id, Status, NumberOfErrors, JobItemsProcessed,
 TotalJobItems, CreatedBy.Email
 from AsyncApexJob where Id =
 :BC.getJobId()];

 // Create and send an email with the results of the batch.
 Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

 mail.setToAddresses(new String[] {a.CreatedBy.Email});
 mail.setReplyTo('batch@mycompany.com');
 mail.setSenderDisplayName('Batch Processing');
 mail.setSubject('Contact Update ' + a.Status);
 mail.setPlainTextBody('The batch apex job processed ' + a.TotalJobItems +
 ' batches with ' + a.NumberofErrors + ' failures.');

 Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });

} //close finish method
} //close class

```

### **Apex Class Used to Schedule the Batch Apex Class:**

```

global class ScheduleUpdateContacts implements Schedulable
{
 global void execute(SchedulableContext SC) {
 UpdateAllContacts uac = new UpdateAllContacts();
 database.executebatch(uac);
 } //close execute method
} //clo

```

@isTest



.....Drive for Excellence

```
private class UpdateAllContactsTest {
```

```
 static testMethod void TestUpdateAllContacts() {
```

```
 // User Id's for the two sales managers.
```

```
 String aId = 'XXXXXXXXXXXXXXXXXXXX';
```

```
 String bId = 'XXXXXXXXXXXXXXXXXXXX';
```

```
 List <Contact> contacts = new List <Contact>();
```

```
 List <Task> tasks = new List <Task>();
```

```
 Test.StartTest();
```

```
 // Create 50 Contacts and assign them to the sales manager of the opposite location.
```

```
 for (integer i=0; i<50; i++) {
```

```
 Contact c = new Contact(FirstName='Test',
 LastName='Contact'+ i,
 Location__c = 'Location 1',
 OwnerId = bId);
 contacts.add(c);
```

```
} //close for-loop
```

```
 // Create 50 more Contacts and assign them to the sales manager of the opposite location.
```

```
 for (integer i=0; i<50; i++) {
```

```
 Contact c = new Contact(FirstName='Test',
 LastName='Contact' + i + i,
 Location__c = 'Location 2',
 OwnerId = aId);
 contacts.add(c);
```

```
} //close for-loop
```

```
 insert contacts;
```

```
 List <Contact> cont = [Select ID, FirstName from Contact Where FirstName=: 'Test' limit 200];
```

```
 // Create a Task for each Contact that was just inserted. Set the date of the task so that it will set the
```

```
 // Last Activity Date to a date older than your protection period.
```

```
 for (Integer i=0; i<100; i++) {
```

```
 Task tsk = new Task();
```

```
 tsk.WhoId = cont.get(i).Id;
```

```
 tsk.ActivityDate = System.today() - 15;
```





.....Drive for Excellence



```

tsk.Status = 'Completed';
tsk.Subject = 'Test Subject';

tasks.add(tsk);

} // close for-loop

try {
 insert tasks;
} catch (System.DMLEexception e) {
 System.debug('Task List not inserted: ' + e);
}

// Call the Batch Apex method.
UpdateAllContacts uac = new UpdateAllContacts();
ID batchprocessid = Database.executeBatch(uac);
Test.StopTest();

AsyncApexJob async = [Select Id, Status, NumberOfErrors, JobItemsProcessed,
TotalJobItems from AsyncApexJob where Id = :batchprocessid];
System.debug('Final results are ' + async);

System.AssertEquals(async.NumberOfErrors, 0);
System.AssertEquals([Select count() from Contact Where OwnerId=:aId AND
FirstName='Test'], 50);
System.AssertEquals([Select count() from Contact Where OwnerId=:bId AND
FirstName='Test'], 50);
System.AssertEquals([Select count() from Task Where Subject = 'Test Subject'], 100);

} //close testmethod

} //close Class

global class TestScheduledApexFromTestMethod implements Schedulable
{
// This test runs a scheduled job at midnight Sept. 3rd. 2022
public static String CRON_EXP = '0 0 0 3 9 ? 2022';

global void execute(SchedulableContext ctx)
{
 CronTrigger ct = [SELECT Id, CronExpression, TimesTriggered,NextFireTime
 FROM CronTrigger WHERE Id = :ctx.getTriggerId()];

 System.assertEquals(CRON_EXP, ct.CronExpression);
 System.assertEquals(0, ct.TimesTriggered);
}

```



```
System.assertEquals('2022-09-03 00:00:00', String.valueOf(ct.NextFireTime));

Account a = [SELECT Id, Name FROM Account WHERE Name =
 'testScheduledApexFromTestMethod'];

a.name = 'testScheduledApexFromTestMethodUpdated';
update a;
}
}
```

**Note** : you must guarantee that the trigger won't add more scheduled classes than the 100 that are allowed.

### Test Class to for schedule Apex:

```
@istest
class TestClass {

 static testmethod void test() {
 Test.startTest();

 Account a = new Account();
 a.Name = 'testScheduledApexFromTestMethod';
 insert a;

 // Schedule the test job
 String jobId = System.schedule('testBasicScheduledApex',
 TestScheduledApexFromTestMethod.CRON_EXP,
 new TestScheduledApexFromTestMethod());

 // Get the information from the CronTrigger API object
 CronTrigger ct = [SELECT Id, CronExpression, TimesTriggered,
 NextFireTime
 FROM CronTrigger WHERE id = :jobId];

 // Verify the expressions are the same
 System.assertEquals(TestScheduledApexFromTestMethod.CRON_EXP, ct.CronExpression);
 // Verify the job has not run
 System.assertEquals(0, ct.TimesTriggered);
 // Verify the next time the job will run
 System.assertEquals('2022-09-03 00:00:00',
 String.valueOf(ct.NextFireTime));
 System.assertNotEquals('testScheduledApexFromTestMethodUpdated',
 [SELECT id, name FROM account WHERE id = :a.id].name);
 Test.stopTest();
 }
}
```



.....Drive for Excellence



```
System.assertEquals('testScheduledApexFromTestMethodUpdated',
 [SELECT Id, Name FROM Account WHERE Id = :a.Id].Name);

}
```

To schedule an Apex class to run at regular intervals, first write an Apex class that implements the Salesforce-provided interface Schedulable.

The scheduler runs as system: all classes are executed, whether the user has permission to execute the class or not. For more information on setting class permissions, see “Apex Class Security Overview” in the Salesforce online help.

To monitor or stop the execution of a scheduled Apex job using the Salesforce user interface, go to **Setup -->Monitoring --> Scheduled Jobs**. For more information, see “Monitoring Scheduled Jobs” in the Salesforce online help.

The Schedulable interface contains one method that must be implemented, execute.  
global void execute(SchedulableContext sc){}

#### Example:

```
global class scheduledAccountUpdate implements Schedulable
{
 global void execute(SchedulableContext SC)
 {
 AccountUpdate acc = new AccountUpdate('Description','Updated Account');
 }
}
```

To get details about AccountUpdate Class go to <http://www.infallibletechie.com/2012/05/batch-apex.html>

Go to Setup à Develop à Apex Classes and then click ‘Schedule Apex’ button.

**My Chatter Settings**

|           | Action                                                  | Name              | Namespace Prefix         | API Version | Is Valid | Status | Site Without Comments | Code Coverage |
|-----------|---------------------------------------------------------|-------------------|--------------------------|-------------|----------|--------|-----------------------|---------------|
| App Setup | Install App                                             | benedsoft         |                          | 22.0        | ✓        | Active | 110                   | 100%          |
|           | Create                                                  | benedsoft         |                          | 22.0        | ✓        | Active | 110                   | 100%          |
|           | Development                                             |                   |                          |             |          |        |                       |               |
|           | <input checked="" type="checkbox"/> Apex Classes        | Set Class Set     | ChangePasswordController | 22.0        | ✓        | Active | 375                   | 100%          |
|           | <input checked="" type="checkbox"/> Apex Triggers       | Set Trigger Set   | ForgotPasswordController | 22.0        | ✓        | Active | 644                   | 88%           |
|           | <input checked="" type="checkbox"/> Apex Test Execution | Set Test Set      | ProfilePageController    | 22.0        | ✓        | Active | 4381                  | 82%           |
|           | API Components                                          | Set API Component |                          |             |          |        |                       |               |

## Schedule Apex

Enter an Apex class that implements the Schedulable interface to be automatically executed on a weekly or monthly interval.

Save Cancel

|                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| <b>Job Name</b> <input type="text" value="Account Update"/>                                                                                                                                                                                                                                                                                                                         | <b>Apex Class</b> <input style="background-color: #f0f0f0;" type="text" value="scheduledAccountUpdate"/> |
| <b>Schedule Apex Execution</b>                                                                                                                                                                                                                                                                                                                                                      |                                                                                                          |
| Frequency<br><input checked="" type="radio"/> Weekly<br><input type="radio"/> Monthly                                                                                                                                                                                                                                                                                               |                                                                                                          |
| <div style="border: 1px solid #ccc; padding: 5px; width: 100%;"> Returns every week on<br/> <input type="checkbox"/> Sunday<br/> <input checked="" type="checkbox"/> Monday<br/> <input type="checkbox"/> Tuesday<br/> <input type="checkbox"/> Wednesday<br/> <input type="checkbox"/> Thursday<br/> <input type="checkbox"/> Friday<br/> <input type="checkbox"/> Saturday </div> |                                                                                                          |
| Start <input type="text" value="5/21/2012"/> (5/21/2012)<br>End <input type="text" value="6/21/2012"/> (5/21/2012)<br>Preferred Start Time <input style="background-color: #f0f0f0;" type="text" value="None"/>                                                                                                                                                                     |                                                                                                          |

Exact start time will depend on job queue activity.

Save Cancel

Qb



Figure clearly explains how to schedule Batch Apex. The frequency may be set to Weekly or Monthly. Start and End dates are important. Preferred Start Time is also important. The job will be sent to Apex Job queue. The preferred Start Time won't be the exact time because it depends upon the job queue. The job will be executed after other jobs in the job queue have been executed.

Once you've successfully saved your Apex Classes go to Setup -> Develop -> Apex Classes -> Schedule Apex. Use the class above that implements the *Schedulable* interface and select the frequency that you want the class to run.

## Batch Apex Governor Limits

Keep in mind the following governor limits for batch Apex:

- Up to five queued or active batch jobs are allowed for Apex.
- The maximum number of batch Apex method executions per a 24-hour period is 250,000 or the number of user licenses in your organization multiplied by 200, whichever is greater.
- The batch Apexstart method can have up to 15 query cursors open at a time per user.
- A maximum of 50 million records can be returned in the Database.QueryLocator object. If more than 50 million records are returned, the batch job is immediately terminated and marked as Failed.
- If the start method of the batch class returns a QueryLocator, the optional scope parameter of Database.executeBatch can have a maximum value of 2,000.
- of Database.executeBatch, Salesforce chunks the records returned by the startmethod into batches of 200, and then passes each batch to the execute method. Apex governor limits are reset for each execution of execute.
- The start, execute, and finish methods can implement up to 10 callouts each.
- Only one batch Apex job's start method can run at a time in an organization. Batch jobs that haven't started yet remain in the queue until they're started. Note that this limit doesn't cause any batch job to fail and execute methods of batch Apex jobs still run in parallel if more than one job is running.



## TEST CLASSES IN APEX (99-107)

1. Introduction to Test classes
2. Example Test cases for Apex classes
3. Test class for Batch Apex
4. Test class for Schedule Apex

Satish Mwla

Bened Software-7799458889

## Testing :-

- If you want to validate whether the application what we have created is working as expected.
  - There are two ways of testing an application.
    - i) Using salesforce user interface.
    - ii) Testing bulk functionality. (upto 200 records can be passed through your code)
  - Before you deploy your code or package to the Force.com App exchange the following conditions should be satisfied.
    - i) Atleast 75% of your Apex Code must be covered by unit test.
    - ii) All the test cases will be successful.
- NOTE :- When you deploy any code production will be executed. Test methods and the test classes are not part of Apex code coverage.
- Every trigger must have some test case.
  - All classes & triggers should compile successfully.

What are the factors that need to be tested in Apex programming:-

- 1) Single Action :- This is to test a single record, produces the correct expected result.
- 2) Bulk actions :- Any Apex code, whether a trigger, a class or an extension, may be invoked for 1 to 200 records. You must test not only the single record case, but the bulk cases as well.
- 3) Positive behaviour :- Test to verify that the expected behaviour occurs through every expected permutation, that is, that the user filled out everything correctly and did not go past the limits.
- 4) Negative behaviour :- There are likely limits to your applications, such as not being able to add a future date, not being able to specify a negative amount & so on.
- 5) Restricted user.

Apex unit test :-

- Unit test are the class methods that verify whether a particular piece of code is working properly or not.
- Unit test method takes no arguments & commits no data to the database, will not send any emails.

## Test method keyword :-

- When you want to create a test method them should be prefixed with keyword testmethod.
- All the test methods are static.

Eg:- public class myClass

{

    Static testMethod void myTest()

{

        // Add test method logic using System.assert(),  
        System.assertEquals()

    // and System.assertNotEqual() here

{

{

NOTE :- Test methods can not be defined in Apex trigger.

## @isTest Annotation :-

If you define any method as @isTest then the method is terminated test method just like what we have defined.

→ If any class is defined with annotation @isTest then that class is defined as test class.

Eg1:- @isTest

```
private class MyClass
{
 static void myTest()
 {
 //code-block
 }
}
```

This is the same test class as in previous example but it defines the test method with the isTest annotation instead.

Eg2:- @isTest

```
private class MyClass
{
 @isTest static void myTest()
 {
 //code-block
 }
}
```

NOTE:- class defined with the isTest annotation don't count against your organization limit of 3 MB for all apex code.

Eg:-

A85

102

@isTest

private class MyTestClass

{

//Methods for testing

@isTest static void test1()

{

//Implement test code

{

@isTest static void test2()

{

//Implement test code

{

}

→ Test classes must be defined as isTest with access specifier as public/private.

→ By default access level is private.

→ Methods of the test class can be called only from the running method.

→ A test method can not be invoked in non-test method.

Ex:-

### Apea class (TV Remote Control) :-

public class TVRemoteControl

{  
    //volume to be modified  
    Integer volume;

    //constant for maximum volume value

    Static final Integer MAX\_VOLUME = 50;

    //constructor

    public TVRemoteControl(Integer v)

    {  
        //set initial value for volume  
        volume = v;

    public Integer increaseVolume(Integer amount)

    {  
        volume += amount;  
        if (volume > MAX\_VOLUME)

            volume = MAX\_VOLUME;

        return volume;

    public Integer decreaseVolume(Integer amount)

    {  
        volume -= amount;

```
if(volume < 0)
{
 volume = 0;
}
return volume;
```

```
public static String getMenuOptions()
{
 return 'AUDIO SETTING - VIDEO SETTINGS';
}
```

Test class :- (TVRemoteControlTest) :-

@istest

```
class TVRemoteControlTest .
```

```
{
```

@istest static void testVolumeIncrease()

```
{
```

```
 TVRemoteControl rc = new TVRemoteControl(10);
```

```
 Integer newVolume = rc.increaseVolume(15);
```

```
 System.assertEquals(25, newVolume);
```

```
}
```

@istest static void testVolumeDecrease()

```
{
```

```
 TVRemoteControl rc = new TVRemoteControl(20);
```

```
Integer newVolume = rc.decreaseVolume(15);
```

```
System.assertEquals(5, newVolume);
```

}

@isTest static void testVolumeDecreaseUnderMin()

{

```
TVRemoteControl rc = new TVRemoteControl(10);
```

```
Integer newVolume = rc.decreaseVolume(100);
```

```
System.assertEquals(0, newVolume);
```

}

@isTest static void testGetMenuOptions()

{

//Static method call : No need to create a class instance

```
String menu = TVRemoteControl getMenuOptions();
```

```
System.assertNotEquals(null, menu);
```

```
System.assertNotEquals("", menu);
```

{  
}

NOTE:- From API 28.0 test methods no longer be reside in  
the non-test class.

## Accessing private members of the class in test class :-

- Private members of the class can't be accessed outside the class. This will create a problem in checking private members.
- If you want to check the private members of the class add annotation `@TestVisible` to before the private/protected members of the class.
- When you keep the annotation `@TestVisible` before the private members those are visible with in a test class.

Eg:-

### Apex class (visible sample class) :-

```
public class VisibleSampleClass {
 //private member variables
 @TestVisible private Integer recordNumber = 0;
 @TestVisible private String areaCode = '(415)';
 //public member variable
 public Integer maxRecords = 1000;
 //private inner class
```

```
@TestVisible class Employee {
```

```
}
```

```
String fullname;
String phone;
//constructor
```

```
@TestVisible Employee (String s, String ph)
```

```
{
```

```
 fullname = s;
```

```
 phone = ph;
```

```
}
```

```
{
```

```
//private method
```

```
@TestVisible private String privateMethod (Employee e)
```

```
{
```

```
 System.debug ('I am private.');
```

```
 recordNumber++;
```

```
 String phone = areaCode + ' ' + e.phone;
```

```
 String s = e.fullname + ' ' + 's phone number is ' + phone;
```

```
 System.debug(s);
```

```
 return s;
```

```
{
```

```
//public method
```

```
public void publicMethod () .
```

```
{
```

```
 maxRecords++;
```

System.debug('From public');

8

Ans  
105

(@TestVisible private class MyException extends visibleSampleClass()

8

TestClass (visible sample class Test) :-

// Test class for visible sample class

@isTest

private class visibleSampleClassTest

{

// this test method can access private members of another class

// that are annotated with @TestVisible

Static testmethod void test1()

{

visibleSampleClass Sample = new visibleSampleClass();

// Access private inner class.

visibleSampleClass.Employee emp = new visibleSampleClass

Employee ('Joe Smith', '555-1212');

// call private method.

String s = Sample.privateMethod(emp);

11

```
// verify result
```

```
System.assert(
```

```
 s.contains('(510)') &&
 s.contains('Joe Smith') &&
 s.contains('555-1212'));
```

```
{
```

```
Static testmethod void test2()
```

```
{
```

```
try
```

```
{
```

```
 throw new VisibleSampleClass.MyException('Thrown from a test');
```

```
}
```

```
Catch(VisibleSampleClass.MyException e)
```

```
{
```

```
// Handle Exception
```

```
{
```

```
}
```

```
Static testmethod void test3()
```

```
{
```

```
// Access public method
```

```
// NO @TestVisible is used
```

```
VisibleSampleClass Sample = new VisibleSampleClass();
```

```
Sample.publicMethod();
```

```
{
```

```
}
```

## Batch Apex example with testcase :-

### Batch Apex class :-

global class Batchclass implements Database.Batchable<Subject>

{

    public String query = 'select id from Account';

    global Database.QueryLocator start(Database.BatchableContext BC)

{

        return Database.getQueryLocator(query);

{

    global void execute(Database.BatchableContext BC, List<Subject> scope)

{

        for(Account a : (List<Account> scope))

{

            System.debug(a);

{

    global void finish(Database.BatchableContext BC)

{

        System.debug('finished');

{

{

Test case :-

private class TestBatchClass

{

Static testMethod void testBatchClass()

{

BatchClass bc = new BatchClass();

bc.query = 'select Id from Account Limit 200';

Test.startTest();

Database.executeBatch(bc, 200);

Test.stopTest();

}

}

## Schedule Apex example with testcase :-

Ago  
107

### Batch class :-

global class customerBatch implements Database.Batchable<Sobject>

{

    global final String Query;

    global final String Entity;

    global final String Field;

    global final String value;

    global customerBatch (String q, String e, String f, String v) {

        Query=q; Entity=e; Field=f; value=v;

}

    global Database.QueryLocator Start (Database.BatchableContext BC)

{

        return Database.getQueryLocator(Query);

}

    global void execute (Database.BatchableContext BC, List<Sobject> scope)

{

        for(Sobject s: scope)

{

            s.put(Field, value);

}

        update scope;

}

    global void finish (Database.BatchableContext BC) {

{

}

global class CustomerSchedule implements Schedulable

{

global void execute(SchedulableContext sc)

{

CustomerBatch cb = new CustomerBatch();

database.executeBatch(cb);

{

}

Test case :-

@isTest

private class ScheduleTest {

static testMethod void myUnitTest()

{

Test.startTest();

String CRON\_EXP = '0 00 11 2025';

String jobId = System.schedule('testScheduledApex',  
CRON\_EXP, new CustomerReportRevokeScheduler());

Contriggers ct = [select id, cronExpression, timesTriggered,  
nextFireTime from Contriggers where id=:jobId];

System.assertEquals(CRON\_EXP, ct.cronExpression);

System.assertEquals(0, ct.timesTriggered);

System.assertEquals('2025-01-01 00:00:00', string.valueOf  
(ct.nextFireTime));

Test.stopTest();

{

## TRIGGERS (108-138)

1. Introduction to Triggers
2. Types of Triggers
3. Types of Events in Triggers
4. Triggers.New in Before Insert
5. Examples to demonstrate Before Insert Trigger
6. Test class for Before Insert trigger
7. Triggers.New in After Insert
8. Examples to demonstrate After Insert Trigger
9. Test class for After Insert Trigger.
10. Triggers.New, Triggers.Old in update Events in Trigger.
11. Examples to demonstrate Before & After Trigger Events
12. Test classes for above scenario
13. Triggers.Old in Before & After Delete
14. Examples to demonstrate Before & After Delete Triggers
15. Test class for above scenarios
16. Triggers.New in After undelete

17. Examples in Testcase

18. Trigger, OldMap & Trigger, New map in Trigger Events

19. Trigger Context Variables

20. Recursive triggers

21. Circular triggers

22. Invoking Apex Methods in Triggers

23. Passing the Trigger Context Variables as a parameter to

24. Apex methods

25. Invoking Future methods in Triggers

26. Governing limits

27. FUTURE ANNOTATIONS (139-141)

## Triggers :-

Trigger is a Apex code that executes before & after on the following types of DML operations.

- Insert
- Update
- Delete
- Merge
- Upsert
- Undelete

Triggers are divided into 2 types.

- 1) Before Triggers.
- 2) After Triggers.

### 1) Before Triggers :-

Before triggers can be used to update or validate values of a record before they are saved to the database.

### 2) After Triggers :-

After triggers can be used to access field values of the records that are stored in the database and use this values to make changes in other records.

## Syntax :-

triggers triggerName on ObjectName (trigger\_events)

{

code-block

}

Where trigger\_events can be Comma Separated list of events.

## Types of events in the triggers :-

- 1) Before insert
- 2) Before update
- 3) Before delete
- 4) After insert
- 5) After update
- 6) After delete
- 7) After undelete

Q:-

trigger Sample on Account (before insert, after delete)

{

//code-block

}

NOTE :- Triggers can only contains keywords applicable to an inner class.

→ you do not have to commit the data manually , it automatically save into database.

### Trigger.New :-

Trigger.New is a context variable which contains list of new records which has caused the trigger to fire.

→ Trigger.New can be used in the trigger events.

1. Before insert
2. Before update.
3. After insert
4. After update
5. After undelete

NOTE :- There is no concept called Trigger.New in delete operations.

### Create a table customers

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 1234  |
| 222 | bbb  | 34  | 3455  |
| 333 | ccc  | 45  | 9876  |

Trigger.New in before insert :-

We have an object with three records.

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 1234  |
| 222 | bbb  | 34  | 3455  |
| 333 | ccc  | 45  | 9876  |

In this object if we are trying to insert new records, into Customer object.

|     |     |    |      |
|-----|-----|----|------|
| 444 | ddd | 34 | 1234 |
| 555 | eee | 23 | 3456 |

Then the new records which we are trying to insert are stored in Trigger.New in before insert event. Which means

List<Customer\_\_c> cus = Trigger.New; // In 4

|     |     |    |      |
|-----|-----|----|------|
| 444 | ddd | 34 | 1234 |
| 555 | eee | 23 | 3456 |

These records are stored into Trigger.New.

NOTE:- The records which are stored in the Trigger.New we can directly perform changes in before insert.

Eg: `for(customer_c c:Trigger.New)`

{

`c.Age--c=30;`

{

NOTE :- Before insert event will occur before new records are inserted into the database. So we can not retrieve the new records using DML operations in before trigger. i.e,

When we have customers table with following records.

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 1234  |
| 222 | bbb  | 34  | 3455  |
| 333 | ccc  | 45  | 9876  |

When we are trying to perform insert these two new records

|     |     |    |      |
|-----|-----|----|------|
| 444 | ddd | 34 | 1234 |
| 555 | eee | 23 | 3456 |

If there is any before insert trigger and we have written any DML in it.

trigger example on customer\_c (before insert)

{

List<customer\_c> my = [select cid\_c, Name, Age\_c,  
Phone\_c from customer\_c];

//this query will only fetch three records

|     |     |    |      |
|-----|-----|----|------|
| 111 | aaa | 23 | 1234 |
| 222 | bbb | 34 | 3455 |
| 333 | ccc | 45 | 9876 |



as remaining two records are not yet inserted.

}

Before insert :-

These triggers will fire when we are trying to insert a new records into a specified object.

- Operations which we have written in trigger will be implemented before new records are saved to the database.
- In before insert, Trigger.New stores the list of new records which we are trying to insert.

## Before Insert :-

These triggers will be fired when we are trying to insert a new records into a specified object.

- operations which we have written in triggers will be implemented before new records are saved to the database.
- In before insert , Triggers.New Stores the list of new records which we are trying to insert.

## Scenarios :-

When we are trying to insert new record into object . If there is any record existing with same account name it should prevent duplicate record.

trigger accountinsert on Account (before insert)

```
for (Account a : Triggers.New)
```

```
List<Account> mynew = [select id, Name from Account
```

```
where Name = :a.name];
```

```
if (mynew.size() > 0)
```

```
a.Name.addError('Account with name is existing');
```

## Testcase :-

@isTest

public class AccountInsert

{

public static void testInsert()

{

String addError;

String myname = 'madhuri';

Account a2 = new Account(name = myname);

List<Account> x = [select name from Account where  
name = :myname];

if (x.size() < 1)

{

System.assertEquals(0, x.size());

insert a2;

{

else

{

addError = 'Existing';

{

System.assertEquals('Existing', addError);

{

{

Scenario 2 :-

Write a trigger to prefix Account Name with 'Mr' when new record is inserted.

Trigger :-

trigger accountprefix on Account(before insert)

{

for(Account a : triggers.New)

{

a.Name = 'Mr' + a.name;

{

{

Test Case :-

@isTest

public class AccountInsert

{

public static testMethod void testInsert()

{

Account a = new Account(name = 'Sam');

a.name = 'Mr.' + a.name;

insert a;

{

{

## Scenarios :-

When ever a new record is ~~created~~ inserted into account object. Before this new record is inserted into Account , delete all the Contacts records with this account name.

## Triggers :-

trigger contactDeletion on Account (before insert)

```
List<String> mynames = new List<String>();
```

```
for (Account a : Triggered.new)
```

```
{
```

```
 mynames.add(a.name);
```

```
}
```

```
List<Contact> mycontacts = [select id, name from Contact
```

```
where name in : mynames].
```

```
delete mycontacts;
```

```
}
```

Testcase:-

@isTest

public class AccountInsert

{

public static testMethod void testdeletion()

{

String myname = 'Sam';

Account a = new Account(name = myname);

Contact con = new Contact(lastname = 'Sam');

insert con;

Contact c = [select id, name from Contact where Name = :myname

limit 1];

if (c != null)

{

System.assertEquals(c.name, a.name);

delete c;

{

insert a;

{

## After insert :-

- this trigger will be fired after new records are successfully saved to the database.
- We can use Triggers.New to refer to the list of new records which we have inserted.
- On Triggers.New we can only perform read only operations.
- On the new list of records we can perform DML operations.

NOTE :- on any records that are successfully saved to database . If we want to perform any changes on those records we have Perform DML operations.

Ex:-

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 1234  |
| 222 | bbb  | 34  | 3455  |
| 333 | ccc  | 45  | 9876  |
| 44  |      |     |       |

When we insert a new records

|     |     |    |      |
|-----|-----|----|------|
| 444 | ddd | 32 | 3456 |
| 555 | eee | 56 | 7655 |

After triggers will be performed after committing the new records into database. which means

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 1234  |
| 222 | bbb  | 34  | 3455  |
| 333 | ccc  | 45  | 9876  |
| 444 | ddd  | 32  | 2456  |
| 555 | eee  | 56  | 7655  |

} After Saving this records then after, insert trigger will be called, so operation written in trigger will be performed after records are successfully inserted.

### Scenario1 :- (Exercise)

Whenever a new record is customer record is successfully created. then update all the customers

When ever a new transaction is performed successfully

then update the customer object balance field based on

if Transaction type = deposit , Balance = Balance + amount;  
= withdraw Balance = balance - amount;

NOTE:- Customer and Transaction has Lookup detail relation.

## Scenario :-

When ever a new Contact is created for a account . update the Corresponding account phone field with the new Contact phone field.

## Trigger:-

trigger updatephone on contact (after insert)

{

List<Account> acc = new List<Account>();

for (Contact c : Triggers.New)

{

Account a = [select id, phone from Account where  
id = :c.AccountId];

a.phone = c.phone;

acc.add(a);

{

update acc;

}

Testcase:-

@isTest

public class Testphone

{

Static testmethod void updatephone()

{

Account a = new Account (name = 'Sam', phone = '(23)'),

insert a;

Account my = [select id from Account where name = 'Sam'];

Contact c = new Contact (lastname = 'Kumar',

accountid = a.id, phone = '456');

insert c;

Account acc = [select id, phone from Account where

id = :c.accountid limit 1];

acc.phone = c.phone;

update acc;

System.assertEquals(acc.phone, c.phone);

{}

{}

## update events in salesforce :-

There are two update events.

1. Before update

2. After update

Trigger.old and Trigger.new are in update events in salesforce.

→ We have a customer object with the following records

### Customer

| cid | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 3455  |
| 222 | bbb  | 34  | 2344  |
| 333 | ccc  | 27  | 9876  |
| 444 | ddd  | 56  | 2346  |

In the above table when we are trying to update the records.

|                 |    |                 |
|-----------------|----|-----------------|
| 333 ccc 27 9876 | to | 333 ccc 29 7654 |
| 444 ddd 78 2346 | to | 444 ddd 35 1234 |

### Trigger.New :-

Trigger.New will store the set of records on which new values on which you are performing update.

i.e, Trigger.New will have

|                 |
|-----------------|
| 333 ccc 29 7654 |
| 444 ddd 35 1234 |

these two records will store into Trigger.New

Triggers.old :-

Triggers.old will store the set of records on which we are performing update. This store the records with old values.

i.e, in the above case Triggers.old will have

|     |     |    |      |
|-----|-----|----|------|
| 333 | ccc | 27 | 9876 |
| 444 | ddd | 78 | 2346 |

Event : Before update :-

When ever we are trying to update any records in the object.

The operations which need to be performed before saving the changes to database are written in before update.

Ex:-

Customers

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 3455  |
| 222 | bbb  | 34  | 2344  |
| 333 | ccc  | 27  | 9876  |
| 444 | ddd  | 56  | 2346  |

Step 1 :-

update these records

333 ccc 29 7654

444 ecc 35 1234

Step 2 :-

Before update operations  
are performed

Step 3 :-

|     |     |    |      |
|-----|-----|----|------|
| 111 | aaa | 23 | 3455 |
| 222 | bbb | 34 | 2344 |
| 333 | ccc | 29 | 7654 |
| 444 | ddd | 35 | 1234 |

- When we modify the value of a record and click on update.
- Before trigger will be called on object and all the operation written it will be performed.
- Records are updated with new values in the database.

VOTE :- If we want to make any changes in the values of new record we can directly perform using Trigger . New in before trigger.

→ We can not perform any changes in the records that are in triggers . New using DML operations as they are not yet committed in before update.

### After update :-

The operations written in the afterupdate trigger will be fired when the changes that we have done are saved to the database.

#### if customer

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 3455  |
| 222 | bbb  | 34  | 2344  |
| 333 | ccc  | 27  | 9876  |
| 444 | ddd  | 56  | 2346  |

step :-

update three records

333 ccc 29 7654

444 ccc 35 1234

| CD  | Name | Age | phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 8455  |
| 222 | bbb  | 34  | 2344  |
| 333 | ccc  | 29  | 7654  |
| 444 | ddd  | 35  | 1234  |

operations written in  
after update performed  
now on new set of data.

→ When we make some changes on the records of the object and click on update.

→ First before trigger operations are performed on the object.

→ All completing before update operations all values in Triggers.New are updated to object.

→ Once records Triggers.New are updated to object then After trigger operations are performed.

NOTE :- In the after update trigger operation we can only read the data from trigger.

→ If we want to perform any changes on the records in after update trigger we have to write DML Statements.

## Scenario:-

When ever Customer record is updated, before updating the record create new record in test object with old values of customer record.

## Triggers:-

trigger CustomerUpdate on Customer\_\_c (before update)

{

List<Test\_\_c> test = new List<Test\_\_c>();

for(Customer\_\_c x : Trigger.old)

{

Test\_\_c t = new Test\_\_c();

t.name = x.name;

t.salary\_\_c = x.salary\_\_c;

t.phone\_\_c = x.phone\_\_c;

test.add(t);

}  
insert test;

}

## Testcase:-

8

## Testcase :-

@isTest

```
public class insertCustomer
```

```
{
```

```
 static testMethod void testCustomer()
```

```
{
```

```
 Customer__c x = new Customer__c();
```

```
 x.name = 'Satya';
```

```
 x.salary__c = 1000;
```

```
 x.phone__c = '123';
```

```
 insert x;
```

```
 Test__c t = new Test__c();
```

```
 t.name = x.name;
```

```
 t.phone__c = x.phone__c;
```

```
 t.salary__c = x.salary__c;
```

```
 insert t;
```

```
 System.assertEquals(t.name, x.name);
```

```
 System.assertEquals(t.salary__c, x.salary__c);
```

```
 x.name = 'Satya k';
```

```
 x.salary__c = 30000;
```

```
 update x;
```

```
}
```

```
}
```

## Scenario 2:-

Trigger :-

triggered CustomerUpdate on customer\_c (before update)

```
{
List<Test_c> test = new List<Test_c>();
```

```
for (Customer_c x : Trigger.New)
```

```
{
}
```

```
Test_c t = new Test_c();
```

```
t.name = x.name;
```

```
t.salary_c = x.salary_c;
```

```
t.phone_c = x.phone_c;
```

```
test.add(t);
```

```
{
}
```

```
insert test;
```

```
{
}
```

Test case :-

```
@isTest
```

```
public class insertCustomer
```

```
{
}
```

static testmethod void testCustomer()

{

Customer\_C x = new Customer\_C();

x.name = 'satya';

x.salary\_C = 1000;

x.phone\_C = '123';

insert x;

x.name = 'satya k';

x.salary\_C = 30000;

update x;

Test\_C t = new Test\_C();

System.assertEquals(t.name, x.name);

System.assertEquals(t.salary\_C, x.salary\_C);

t.name = x.name;

t.phone\_C = x.phone\_C;

insert t;

{

}

### Scenarios:-

To update the owner of a case based on the values selected within a pick list and populate the owner field with the createdBy field data. When we have selected any

Field Name = Status

Pick list values =

Priced - (Initial)

Priced - (Re-priced)

Price file loaded

trigger CaseTrigger on case(before update)

{

for(Case c : Triggers.new())

{

if(c.Status == 'priced - (Initial)' || c.Status == 'priced - (Re-priced)' || c.Status == 'price file loaded')

{

c.OwnerId = c.CreatedById;

}

{

}

Scenario4:-

Write a trigger that will prevent a user from creating a lead that already exists as a Contact. We'll use the lead/contact's email address to detect duplicates.

Lead is created or updated.

1. Lead has an email address
2. Try to find a matching Contact based on email address  
(using SOQL!)
3. If a match is found, give the user an error.
4. If a match is not found, do nothing.

triggers FindDups on Lead (before insert, before update)

```
for(Lead myLead : Trigger.New) {
 if(myLead.Email != null)
```

```
List<Contact> dupes = [select id from Contact where
 Email = :myLead.Email];
 if(dupes == null && dupes.size() > 0) {
```

```
String errorMessage = 'Duplicate Contact found!';
 errorMessage += 'Record ID is ' + dupes[0].Id;
```

```
 myLead.addError(errorMessage);
}
```

## Delete events in Salesforce triggers :-

There are two types of delete events.

1. Before delete
2. After delete

Trigger.old in before or after delete :- this will store the list of records which are trying to delete.

→ On this records we can only perform read only operations.

### Customer

| CID | Name | Age | Phone |
|-----|------|-----|-------|
| 111 | aaa  | 23  | 8455  |
| 222 | bbb  | 34  | 2344  |
| 333 | ccc  | 27  | 9876  |
| 444 | ddd  | 56  | 2346  |

→ If we are trying to delete 333-ccc-27-9876  
444-ddd-56-2346

Then Trigger.old will contain these two records.

Scenario:- When we are trying to delete customer record delete all the corresponding child records from Test object . where Customer is lookup field in the loan object .

Trigger:-

triggers CustomerUpdate on Customer\_c (after delete)

{

List<Test\_c> test = [select id from Test\_c where cdetails\_c in :trigger.old].

delete test;

}

Test case:-

@isTest

public class TestExam

{

Static testmethod void ceestest()

{

Customer\_c c = new Customer\_c (name = 'Satya', salary\_c = 1000);  
insert c;

Test\_c t = new Test\_c (name = 'chandu', cdetails\_c = c.id);

```
insert t;
```

```
List<Test_c> test = [select id from Test_c where
cDetails_c = :c.id];
```

```
if (test.size() > 0)
```

```
{
```

```
 delete test;
```

```
{
```

```
 delete c;
```

```
{
```

```
}
```

NOTE:- there is no concept called triggers.new in after delete,  
before delete triggers.

Event: After undelete:

When we are undelete the records from the recycle bin this  
operations written in after undelete will be performed.

Triggers.New:- The records which we have undeleted are stored in  
Triggers.New.

Scenario:-

when we create the opportunity with probability = 50% then the opportunity owner will be automatically added to AccountTeam of the associated account for the opportunity.

Trigger :-

trigger addToAccountTeam on opportunity (after insert, after update)

{

List<AccountShare> acc\_share = new List<AccountShare>();

List<AccountTeamMember> acc\_team = new List<AccountTeamMember>();

};

for(Opportunity opp : Trigger.New)

{

if(opp.probability == 50)

{

AccountTeamMember team = new AccountTeamMember();

team.accountid = opp.accountid;

team.userid = opp.ownerid;

AccountShare share = new AccountShare();

share.accountid = opp.accountid;

share.accesslevel = 'Read/Write';

share.opportunityAccessLevel = 'Read Only';

```
share.accessLevel = 'read only';
```

```
acc_share.add(share);
```

```
acc_team.add(team);
```

{

{

```
if(acc_team != null)
```

{

```
insert acc_team;
```

{

```
if(acc_share != null && acc_share.size() > 0)
```

{

```
insert acc_share;
```

{

{

Scenario 2 :- Invoking the Apex class from the trigger.

When ever new customer record is created/updated then  
income tax should be calculated based on salary and should  
be updated to field ittax (Currency field).

Trigger :-

```
trigger CustomerUpdate on Customer__c (before insert,
before update)
```

{

```
for(Customer__c cust : Trigger.New)
```

{

IncomeTax it = new IncomeTax();

cust. Ittax\_c = it.calculateTax(cust.salary\_c);

{

{

Testcase:-

@isTest

public class IncomeTest

{

static testMethod void testIncome()

{

Customer\_c c = new Customer\_c(name = 'Sam', salary\_c = 30000);

IncomeTax t = new IncomeTax();

c. Ittax\_c = t.calculateTax(c.salary\_c);

insert c;

Customer\_c cust = [select. id, salary\_c, Ittax\_c from  
Customer\_c where id = :c.id];

cust. Ittax\_c = t.calculateTax(cust.salary\_c);

update cust;

{

{

Scenarios:-

Apex class :-

public class IncomeTax

{

    public decimal tax { set; get; }

    public decimal calculateTax(decimal amount)

{

        if (amount > 500000)

{

            tax = amount \* 0.20;

}

    else

{

        tax = amount \* 0.10;

}

    return tax;

{

}

Scenario 3:-

Before we insert a new record in the customer object calculate the ittax field value based on salary field value and then insert.

→ If we delete any of the existing customer record then first Create new test object record with customer record values then delete customer record.

Apex class:-

global class TriggerExample

{

Global static void calculateTax (Customer\_\_c[] cust)

{

for (Customer\_\_c c : cust)

{

if (c.salary\_\_c < 50000)

{

c.ittax\_\_c = c.salary\_\_c \* 0.10;

}

else

{

c.ittax\_\_c = c.salary\_\_c \* 0.20;

{

}

```
global static void createTest(customer_c[] cust)
```

```
{
```

```
list<test_c> test = new list<test_c>();
```

```
for (customer_c c : cust)
```

```
{
```

```
test_c t = new test_c(name=c.name, salary_c=
```

~~c.salary\_c)~~

```
test.add(t);
```

```
{
```

```
insert test;
```

```
{
```

```
}
```

trigger:-

trigger customerUpdate on customer\_c (before insert,  
before delete)

```
if (trigger.isBefore && trigger.isInsert)
```

```
{
```

```
triggerExample.calculateTax(trigger.New);
```

```
}
```

else  
{  
if (Trigger.isBefore && Trigger.isUpdate)

{

TriggerExample.calculateTax(Trigger.old);

{  
}  
}Test class :-

@isTest

public class TriggerExampleTest

{  
}

Static testmethod void testTriggerExample()

{  
}

Customer\_\_c c = new Customer\_\_c(name = 'satya', salary\_\_c = 40000);

List&lt;Customer\_\_c&gt; cust = new List&lt;Customer\_\_c&gt;();

cust.add(c);

TriggerExample.calculateTax(cust);

insert cust;

Test\_\_c t = new Test\_\_c(name = c.name, salary\_\_c = c.salary\_\_c);

insert t;

delete c;

{  
}{  
}

## Usage of Trigger.NewMap and Trigger.OldMap :-

When ever we try

Scenario :-

When ever we try to update the phone of account record then update the related Contact phone no. with the new Account phone no before account record is updated.

When we delete the account record then delete the corresponding Contact records.

trigger Contactupdate on Account (before update, after delete)

{

if (trigger.isBefore && trigger.isUpdate)

{

Map<Id, Account> mymap = Trigger.newMap;

List<Contact> con = new List<Contact>();

List<Contact> cons = [select id, phone, Account id from Contact

Where accountid in :mymap.keySet()];

for (Contact c: cons)

{

c.phone = mymap.get(c.accountid).phone;

con.add(c);

}

update con;

{

if (Triggers. isAfter && Triggers. isDelete)

{

Map<ID, Account> deleteacc = Triggers.oldMap;

List<Contact> mycontact = [select id from Contact where  
Accountid in : deleteacc.keySet()];

delete mycontact;

{

{

}

## Recursive triggers :-

→ you want to write a trigger that creates a new record as part of its processing logic. However, that record may then cause another trigger to fire, which in turn causes another to fire, and so on. You don't know how to stop that recursion.

→ use a static variable in an Apex class to avoid an infinite loop.

Static variables are local to the context of a web request

(or test method during a call to `runTests()`), so all triggers that fire as a result of a user's action have access to it.

## Scenario :-

Suppose there is a scenario where in one trigger performs update operation, which results in invocation of second trigger and the update operation in second trigger acts as triggering criteria for trigger one.

### Class :-

public class utility

{

    public static boolean isFutureUpdate;

}

Trigger :-

trigger updateSomething on Account (after insert, after update)

{

/\* this trigger performs its logic when the call is not from  
@future \*/

if (utility.isFutureUpdate != true)

{

Set<Id> idsToProcess = new Set<Id>();

for (Account acct : trigger.new)

{

if (acct.NumberOfEmployees >= 500)

{

idsToProcess.add(acct.Id);

{

{

/\* sending Ids to @future method for processing \*/

futureMethods.processLargeAccounts(idsToProcess);

{

{

class:

public class FutureMethods

{

@future

public static void processLargeAccounts(Set<Id> acctIDS)

{

List<Account> accsToUpdate = new List<Account>();

~~/\* if futureUpdate is set to true to avoid recursion \*/~~

Utility.isFutureUpdate = true;

update accsToUpdate;

{

}

Satish MMW

Scenario 2:- Recursive trigger error.

Att  
129

To explain about recursive trigger.

trigger Recursive on Account(After insert, after update)

{

if (~~TriggerName~~.~~ftobj~~).

\* 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 trigger 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 triggers error we will use Static Variable .

global class TriggerVar

{  
global static boolean flag = false;

}

Triggers :-

triggers Recursive on Account (after insert, after update)

{  
if (TriggerVar.flag != true)

{  
if (Trigger.isBefore && Trigger.isInsert)

TriggerVar.flag = true;

Account acc = [select id, name from Account where  
name = 'madhuri' limit 1];

acc.name = 'y Madhuri';

update acc;

}

if (Trigger.isAfter && Trigger.isUpdate)

{

TriggerVar.flag = true;

Account a = new Account(name = 'Satisf');

insert a;

{

{

→ In this case the trigger operation will be executed only when flag value will be false. Once the trigger 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 trigger. But Salesforce allows only 5 batch jobs in the queue. So if the trigger 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 trigger, instead we will define callouts in the future annotated methods & this future annotated methods can be called from the trigger.

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. Save 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 update , fire before and after triggers are more time in addition to Standard validations. Custom validations rules are not run again.

Att  
131

10. Executes escalation rules.
11. Commits to the database.

94 0 89  
stage 1  
stage 2  
stage 3  
stage 4  
stage 5  
stage 6

Satish Awla

## 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 subjects which we are trying to insert into database or list of records with which we are going to update.

→ This variable is available in Triggers which perform insert events or update events.

→ To.

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

Map < ID, Subject >

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 subject records. (This will store the list of old records with old values what it has before the operation).

This is available 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 Subject 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:-

133

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.

triggers UpdateSalesRep on Account (Before insert, Before update)

{

Set<Id> setAccountees = new Set<Id>();

for (Account Acc : triggers.New)

{

setAccountees.add (Acc.OwnerId);

}

Map<Id, user> user\_map = new Map<Id, user>([select Name  
from user where id in :setAccountees]);

for (Account Acc : triggers.New)

{

User us = user\_map.get (Acc.OwnerId);

Acc.Sales\_Rep\_\_c = us.Name;

}

}

### Scenario 02 :-

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)
 {
 ContactMasterHandler ConIns = New ContactMasterHandler();
 ConIns.CreateContactRelationshipByContact(trigger.New);
 }
}
```

### Class :-

```
public class ContactMasterHandler
{
 public void CreateContactRelationshipByContact(List<Contact>
 list_contacts)
```

```
list<Contact_Relationship_c> contList = new List<Contact_Relationship_c>();
for(Contact newConts : List_Contacts)
{
 if(newConts.Contact_Relationship_c == true)
 {
 Contact_Relationship_c CR = new Contact_Relationship_cc();
 CR.Name = newConts.lastname;
 CR.Contact_c = newConts.id;
 contList.add(CR);
 }
}
'insert contList;
```

### Scenario 03 :-

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

### Triggers :-

triggers ContactRelationshipMasterTrigger on Contact\_Relationship\_\_c

{  
if(trigger.isBefore)

{  
if(trigger.isUpdate)

{  
//call the handlers for the before update trigger event

updateCROwnerName ContactRelUpd = new UpdateCROwnerName();

contactRelUpd.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.OwnerId);
}

list<User> u = [select id, Name from user where id in :IdSet];
for(User 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.OwnerId != null)
 {
 list_recs.Name = map_id_name.get(list_recs.OwnerId);
 }
 }
}

```

#### 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. (Lookup 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 DeleteCronContactDeletion on Contact (before delete)

```
if (trigger.isDelete)
 if (trigger.old != null)
 for (Contact c : trigger.old)
 for (ContactRelationship cr : c.getContacts())
 if (cr != null)
 cr.delete();
 c.removeContactRelationship(cr);
 c.save();
```

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).

→ Trigger 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 :-

triggers undeleteCron(ContactUnDeletion on Contact (After undelete))

{

if (trigger.isUndelete)

{

// call the handler for the after undelete trigger event

ContactMasterHandler\_undelete conIns = New

ContactMasterHandler\_undelete();

conIns.undeleteContactRelationshipByContact (trigger.New);

}

}

class :

```
public class ContactMasterHandler_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> U;
```

~~list\_Conrels = [select id from Contact\_Relationship\_\_c where~~

~~isDeleted = TRUE and Contact\_\_c in :~~

~~GlobalUtility.getUniqueIds (list\_contacts);~~

~~undelete list\_Conrels;~~

## Scenario 6 :-

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 lookup 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);

138

```
{
 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 callback, 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 Subjects or objects as arguments.

~~Imp:- How to pass Subjects/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 call future (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> addressu = new List <String>();
```

```
AddressHelper ah = new AddressHelper ('1 base st',
'san Francisco', 'CA', '194105');
```

//Serialize my objects ah1, ah2, ah3

```
address.add(JSON.serialize(ah3));
```

```
addresses.add(JSON.serialize(ah2));
```

addresses.add(JSON.Serialize(ah3))

~~callFuture(address); // invoke the future method by passing a string address where it a serialized form of Apex class objects.~~

४

@fctcse

Static void callFuture(List<String> addressess)

Address curAddress = null;

```
for (String s : addresses)
```

४०

```
CurrAddress = (Address) JSON.deserialize(curr,
 AddressHelper.class);
```

System.debug('Deserialized in future: '+currAddress.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 SHARING RULES (142 - 145)

1. Introduction to Apex sharing rules
2. Apex sharing Rules on Standard objects
3. Apex Sharing Rules on custom objects.

Satish Awana

## 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.

| <u>Object Name</u> | <u>ShareObject Name</u> |
|--------------------|-------------------------|
| Account            | AccountsShare           |
| Contact            | ContactShare            |
| Opportunity        | OpportunityShare        |

→ For custom objects

| <u>ObjectName</u> | <u>ShareObjectName</u> |
|-------------------|------------------------|
| Customer_c        | Customer_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.

→ Sharing are granted to the users implicitly by

- i) organization wide default
- ii) role hierarchy
- iii) New 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 user or 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.

- ii) User or Group Id :- the user's or Group id's to which you are granting access the grant group can be public group, role or territory.

iv) Row cause :- the reason why the user or group is being granted access. Att 143

### Sharing Reason field :-

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

#### Reason field value

Account sharing

Associated record owner  
& sharing

Owner

Opportunity Team

Sharing Rule

Territory Assignment  
Rule

#### Row cause Value

Implicit child

Implicit parent

Owner

Team

Rule

Territory Rule.

ii) User 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.

| <u>Reason field value</u> | <u>RowCause value</u> |
|---------------------------|-----------------------|
| Manual sharing            | Manual                |
| Territory Manual          | Territory Manual      |

- 1) 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.

A119  
144

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 userOrgGroupId)

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.userOrgGroupId = userOrgGroupId;

//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 trigger 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 user records called Hiring\_Manager & Recruiter. Also, the job custom object should have two sharing reasons added called Hiring\_Managers and Recruiters.

Also  
145

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 manager sharing

```
Job_Share recrutorshs;
```

```
Job_Share hmshs;
```

```
for(Job__c job : trigger.new)
```

//Instantiate the sharing objects

```
recrutorshs = new Job_Share();
```

```
hmshs = new Job_Share();
```

```
hmshs = new Job_Share();
//set the ID of the record being shared.
recruitershs.parentId = job.Id;

hmshs.parentId = job.Id;
//set the ID of user or group being granted access.
recruitershs.UserIdOrGroupId = job.Recruiter__c;
hmshs.UserIdOrGroupId = job.Hiring_Manager__c;
//set the access level.
recruitershs.AccessLevel = 'Edit';
hmshs.AccessLevel = 'Read';
//set the Apex sharing reason for hiring manager and
//recruiter.
recruitershs.RowCause = Schema.Job_Share.RowCause.Recruiter__c;
hmshs.RowCause = Schema.Job_Share.RowCause.Hiring_
Manager__c;
//Add objects to list for insert
jobshs.add(recruitershs);
jobshs.add(hmshs);
Database.SaveResult[] lso = Database.insert(jobshs, false);
```

## FLOWS (146 - 155)

1. Introduction to Apex visualforce
2. Examples to create flows
3. Introduction to Plugins
4. Examples to create visualflows using plugins
5. Invoking visualflows in visualforce page
6. Refer to visualflows College project workbook for Examples.

NOTE: You can collect all the 6 workbooks related to College Management System project.

## 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

Design

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.

## Flows:-

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

## Navigation:-

Setup → Build → Create → Workflows → Flows → New flow

We have

1. palettes
2. Resources
3. Explorers

## 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 Info

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 fields to the screen. The type of fields are

Input field, choice, multi-select fields and outputs.

## Input fields :-

F1-2

- \* Textbox
- \* Long Text Area
- \* Number
- \* Currency
- \* Date
- \* Password
- \* Checkbox

Textbox:- Drag the text box from inputs and drop it into the view screen.

Double click on the text box

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 :        | <input type="text"/>                                     |
| unique name* : | <input type="text"/> <input checked="" type="checkbox"/> |
| Input Type:    |                                                          |
| Default value: | <input type="text"/> <input checked="" type="checkbox"/> |
| Scale :        | This specifies how many decimal points we should have    |
| Required       | <input checked="" type="checkbox"/>                      |

currency.

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 : [ ] v

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.

When we click on above checkbox we get

Input Field settings

Label : [ ]

Required

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

|                 |                        |
|-----------------|------------------------|
| Label :         | <input type="text"/>   |
| Unique name :   | <input type="text"/>   |
| Choice Type :   | Dropdown List          |
| Value datatype: | <input type="text"/> ✓ |
| Default value:  | <input type="text"/> ✓ |

Choice



Create new choice



|                      |                          |
|----------------------|--------------------------|
| Label :              | <input type="text"/>     |
| unique name :        | <input type="text"/>     |
| Value datatype:      | <input type="text"/> ✓   |
| Stored value:        | <input type="text"/>     |
| Show Input selection | <input type="checkbox"/> |

Logic :-Decision :-

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

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

|                  |                      |
|------------------|----------------------|
| Enter the name : | <input type="text"/> |
| unique Name* :   | <input type="text"/> |

Write a criteria : Give a name to the condition what you are writing.

Default outcome : Give the name to the default outcome

## Database operations from visualforce:-

128  
149

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

|              |                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------|
| Name :       | <input type="text"/>                                                                          |
| uniqueName*: | <input type="text"/>                                                                          |
| create :     | <input checked="" type="checkbox"/> select the object in which you want to insert the record. |
| Field:       | <input type="text"/> v                                                                        |
| value:       | <input type="text"/> <input checked="" type="checkbox"/>                                      |

Map the fields of an object with the values in the flow.

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 search 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.

|                    |                                                          |
|--------------------|----------------------------------------------------------|
| Name :             | <input type="text"/>                                     |
| uniqueName*:       | <input type="text"/> <input checked="" type="checkbox"/> |
| select the object: |                                                          |
| Filter criteria    |                                                          |
| Lookup :           | <input type="text"/> v                                   |

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

Map the field and variable:

Lookup\*

| Field                                                    | operator                                                 | Variable value                                           |
|----------------------------------------------------------|----------------------------------------------------------|----------------------------------------------------------|
| <input type="text"/> <input checked="" type="checkbox"/> | <input type="text"/> <input checked="" type="checkbox"/> | <input type="text"/> <input checked="" type="checkbox"/> |

Record update :- Give the filter criteria and select the object. The records which are satisfying 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 these two methods.

\* `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.

3) `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 return `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.Inputparameter (name, optional Description,

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

name :- Name of the input parameters.

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

parametersType :- 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.Inputparameters ('sname', 'Student name',

process.pluginDescribeResult.parameters.String, true),

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

process.pluginDescribeResult.parameters.Decimal, true),

?;

process.pluginDescribeResult.parameterType can take

BO  
ISI

- 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.outputparameters = 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.

ax:-

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(parametersname);

}

## 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.Inputparameters>{

{

        new process.pluginDescribeResult.Inputparameters('sub1',

            process.pluginDescribeResult.parameterType.Integer, true)

        new process.pluginDescribeResult.Inputparameters('sub2',

            process.pluginDescribeResult.parameterType.Integer, true)

{};

result.outputparameters = 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.

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

you write 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 will cause when datatype mismatch.

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 describer()

{

process.plugindescriberesult result = new process.plugindescriberesult();

result.inputparameters = new List<process.plugindescriberesult.

Inputparameter>{

new process.plugindescriberesult.Inputparameter('sub1',

process.plugindescriberesult.parametersType.Decimal, true)

new process.plugindescriberesult.Inputparameter('sub2',

process.plugindescriberesult.parametersType.Decimal, true)

};

result.outputparameters = new List<process.plugindescriberesult.

Outputparameter>

{

new process.pluginDescribeResult.outputParameters('total',  
F-8  
process.pluginDescribeResult.parametersType.Decimal)  
};  
return result;  
}  
}

Satish Mania

Selectoption :-

Selectoption object specifies one of the possible values for visualform select checkboxes, selectlist, select radio component.

It consist of label that is displayed to the end user, and the value that can 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)

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

Selectoption one = new Selectoption(value, label)

Methods:-

1. String getlabel()

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

Eg:- one.getlabel();

O/P:- once

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 setLabel

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>

<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, SchemaObjectType> getGlobalDescribe()

This method returns a map of all the Object names as keys and Object 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.ObjectType> m = schema.getGlobalDescribe();~~

~~Set<String> obj = m.keySet();~~

~~for (String s : obj)~~

~~{~~

~~SelectOption op = new SelectOption(s, s);~~

~~Options.add(op);~~

~~}~~

~~}~~

~~}~~

Satish Amna

## Schema programming:-

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

### Schema methods:-

- 1) Map<String, SchemaObjectType> getGlobalDescribe()

This method returns a map of all the Object names as keys and Object 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.ObjectType> m = schema.getGlobalDescribe();
```

```
 Set<String> obj = m.keySet();
```

```
 for (String s : obj)
```

```
{
```

```
 SelectOption op = new SelectOption(s, s);
```

```
 Options.add(op);
```

```
}
```

```
}
```

```
2
```

Visualforce page:-

155-

```
<apex:page controller="SchemaDescribeExample">
 <apex:form>
 <apex:selectList size="1">
 <apex:selectOptions value="{!options}"/>
 </apex:selectList>
 </apex:form>
</apex:page>
```

\$

## Schema . DescribeObjectResult :-

This describes object methods returns an array of DescribeObjectResult 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 mergeable :- Indicates whether the object can be merged with objects of its type.

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

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

Boolean Searchable :- Indicates whether object can be searched via search method. 156

Search Layoutable :- Indicates whether layout information can be retrieved v describeSearchLayouts 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();  
(8)

Schema.DescribeSObjectResult result = Schema.SObjectType.Account;

Example program:-

Field Example (Apex class):-

```
public class FieldExample {
 public FieldExample() {
 System.out.println(result);
 }
```

Schema.DescribeSObjectResult res = Account.SObjectType.getDescribe();  
result = '' + res;

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

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

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

```
String getLabel();
String getKeyPrefix();
String getLabelPlural();
```

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

Child relationship methods for given object :-

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

If an Subject 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 Subject.

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);
}

```

~~1948~~

```

<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. User 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 myobj = '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);
 }
}
```

### RecordTypeInfo By ID :-

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 Recordtypes of a given object :-

```
public fieldExample() {
```

```
 options = new List<Selectoption>();
```

```
 Schema.DescribeSobjectResult d = Schema.SObjectType.one_c;
```

```
 Map<Id, Schema.RecordTypeInfo> stMapById = d.getRecordTypeInfosById();
```

```
 Set<Id> s = stMapById.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.get("Describe");

Ex:-

DescribeFieldResult stores the description about the field of an object. To get the description

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

↓            ↓  
obj name field name

### DescribeFieldResult methods :-

String getLabel();

Integer getLength();

String getLocalName();

String getName();

Integer precision();

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 FieldSet {
```

```
 public List<SelectOption> options;
```

```
 public List<SelectOption> getOptions()
```

```
 {
 return options;
 }
```

159  
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.getDescribe().Account.fields.Industry;

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

Map<String, Schema.SObjectType> m = schema.getDescribe().Account.fields.getMap();

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

```
public class FieldDef {
 public List<SelectOption> options;
 public List<SelectOption> getOptions() {
 return options;
 }
 public FieldDef() {
 Options = new List<SelectOption>();
 Map<String, Schema.SObjectType> 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="FieldDef">
<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. H40 160

public class FieldDef {

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 FieldDef {

Map<String, Schema.SObjectType> my;

public String cname {get; set; }

public List<SelectOption> myoptions;

public List<SelectOption> options;

Public FieldDef()

{

Options = new List<SelectOption>();

myoptions = new List<SelectOption>();

my = schema.getGlobalDescribe();

Set<String> myobj = my.keySet();

for(String s: myobj)

{

if(my.get(s).getDescribe().isCustom())

{

SelectOption ob = new SelectOption(s,s);

myoptions.add(ob);

}

}

public PageReference show()

{

Map<String, Schema.SObjectField> m = my.get(cname).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;
}
```

~~get this now~~

```
<apex:page controller="FieldDef">
<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:selectOptions>
```

161

```
</apex:selectList>
</apex:pageBlockSectionItem>
<apex:commandButton value="click" action="f{!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="${!selectedObject}">
 <apex:selectOption itemLabel="--None--" itemValue="--None--"/>
 <apex:selectOptions value="${!supportedObject}"/>
 </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> supportedObject {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.ObjectType 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.ObjectType SystemObjectType = gd.get(SelectedObject);
```

163

```

Schema.DescribeSObjectResult s = systemObjectType.getDescribe();
Map<String, Schema.SObjectField> M = s.fields.getMap();
for(Schema.SObjectField 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);
}

```

194

Write a program on dynamic multi-select 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>

<apex:image value = "${!resource.multiselect}" />
<apex:actionSupport event = "onclick" action = "${!selectClick}" reRender
= "abcd" />
</apex:image>

<apex:image value = "${!resource.multiselect}" />
<apex:actionSupport event = "onclick" action = "${!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>

164

### 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> getCountry() {
 List<SelectOption> options = new List<SelectOption>();
 Schema.DescribeFieldResult fieldResult = Schema.sObjectType.Account.fields.type.getDescribe();
 List<Schema.PicklistEntry> plc = fieldResult.getPicklistValues();
 for (Schema.PicklistEntry f : plc) {
 options.add(new SelectOption(f.getLabel(), f.getValue()));
 originalvalues.add(f.getLabel());
 }
 }
}
```

```
return options;
}

public multiselect() {
 {
 getCount();
 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> getunselectedValues() {
 }
```

```

 {
 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> getSelectedValue()
 {
 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)

```

~~Selected value = options1[i].getValue();~~

~~else~~

~~Selected value = ',' + options1[i].getValue();~~

~~{~~

~~return options1;~~

~~{~~

public void save()

~~{~~



```
Account acc = new Account();
```

```
acc.Name = accName;
```

```
acc.Selected_Values_c = selectedValue;
```

```
insert acc;
```

}

```
List<Account> selAcc = new List<Account>();
```

```
public List<Account> GetEdit()
```

{

```
selAcc = [select Name, Selected_Values_c from Account where id =:
```

ApexPages.currentPage().getParameters().get('id') and

Selected\_Values\_c != null];

String get = selAcc[0].Selected\_Values\_c;

```
list<string> lstSplitData = get.split(',') ;
```

AccName = selAcc[0].Name;

```
if(lstSplitData.size() != null && lstSplitData.size() != 0 && originalValues.
```

size() != null && originalValues.size() != 0)

{

for(String sSelected : lstSplitData)

{

if(originalValues.contains(sSelected.trim()))

{

lstFinal.add(sSelected);

}

for(String sFinal : lstFinal)

{

```
if (originalvalues.contains(sFinal.trim()))
{
 originalvalues.remove(sFinal);
}

rightValues.addAll(lstFinal);

return selfAcc;
}
}
```

Satish Mania

## JSON(167-180)

1. Introduction to JSON
2. Examples of JSON formats
3. Arrays, Objects in JSON Format
4. JSON class
5. System.JSON Generator (Generating the JSON Format String)
6. Example programs
7. JSON parser class & its methods
8. Example programs for parsing
9. Deserialization of JSON.

## 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": "Rakesh", "Branch": "CSE" }

Field Name      value      Field Name      value

→ JSON values can be numbers (Integers, Decimal, Long, Double)

Eg:- { "Age": 20 }

{ "Salary": 10000.00 }

### 2) String

Eg:- { "Name": "Saurabh" }

{ "Branch": "CSE" }

### 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 e;
 Integer age;
 }
}

```

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 braces. { }.

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(Any type 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 or object this method ignores this attribute & 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 attribute that are in the apex:type (apex:class or apex:object) specified in the argument then 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);

Even though 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 then it throw run time exception.

O/P: visualforce : runtime error

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

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

#### 5. public static Object deserializeUntyped(String jsonString)

This method deserializes the JSON String into collection of primitive data types)

Satish MNM

150 170

180 190 200  
170 180 190 200  
160 170 180 190 200  
150 160 170 180 190 200  
140 150 160 170 180 190 200  
130 140 150 160 170 180 190 200  
120 130 140 150 160 170 180 190 200  
110 120 130 140 150 160 170 180 190 200  
100 110 120 130 140 150 160 170 180 190 200  
90 100 110 120 130 140 150 160 170 180 190 200  
80 90 100 110 120 130 140 150 160 170 180 190 200  
70 80 90 100 110 120 130 140 150 160 170 180 190 200  
60 70 80 90 100 110 120 130 140 150 160 170 180 190 200  
50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200  
40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200  
30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200  
20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200  
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200  
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200

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 ()

Output:- { }

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 ()

WriteObjectField (String, Anytype)

WriteObject (Anytype)

WriteString (String)

WriteTime (time)

## Description of JSONGenerator methods :-

IS - b

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 Boolean 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.

(49)

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 :-

IS-7

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

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

Ex2:

```
class Student
```

```
{
 public String name;
 public Integer age;
}
```

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

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

JSONGeneratorExam :- (Apex class):-

```
public class JSONGeneratorExam {
 public String result {get; set;}
 public JSONGeneratorExam() {
 JSON 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.toJSONString();
}
}
```

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;
 }
}

```

### JSONGenerator Exam (Apex Class)

```

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

```

## JSON Generation Example (visualforce page):-

17/11/19

```
<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];
JSON.serialize(acc);
String result = JSON.serialize(acc);
System.debug(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 :-

I8-9

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-NUL :-

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 to 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.

String str = '{"Name": "Sam", "Age": 20}';

Eg:- JSONparser p = JSON.createparser(str);

```
p.nextToken();
```

```
p.nextToken();
```

result = p.getCurrentName(); // result = Name.

```
p.nextToken();
```

result = p.getCurrentName(); // Current token is "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 or 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 or 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 or 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():-

15/07/17

Syn:-

public boolean getBooleanValue()

Must be in the form of JSONToken.VALUE\_TRUE or JSONToken.VALUE\_FALSE

## getDatetimeValue():-

Syn:-

public Datetime getDatetimeValue()

Must be type of JSONToken.VALUE\_STRING

Eg:- String JSONContent =

{ "transactionDate": "2011-03-22T13:01:23" };

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"}';
```

IS-12

```
Time arrivalTime = parser.getTimeValue();
```

## readValuesAs():-

Syn:-

```
public Object readValuesAs(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();
```

class Bemed  
{  
 Integer age;  
 String branch;

```
Bemed b = (Bemed)p.readValuesAs(Bemed.class);
```

```
System.assertEquals(b.Age, 20);
```

## readValuesAsStrict():-

Syn:-

```
public Object readValuesAsStrict(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&lt;Bened&gt;();

```
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&lt;Bened&gt;)p.readValueAs(List&lt;Bened&gt;.class);

p.nextToken();

p.nextValue();

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

JS-13

}

?

### Bened Apex :-

```
public class Bened {
```

```
 String name;
```

```
 String branch;
```

}

### parseExample (visualforce page) :-

```
<apex:page controller="parseExample">
<apex:pageBlock>
 <apex:pageBlockTable value="{!result}" var="a">
 <apex:column value="{!a}"/>
 </apex:pageBlockTable>
 <apex:outputLabel>My Name =={!name}</apex:outputLabel>

 <apex:outputLabel>Age =={!age}</apex:outputLabel>

 <apex:outputLabel>Salary =={!salary}</apex:outputLabel>

 <apex:outputLabel>nextToken =={!nextToken}</apex:outputLabel>

```

Deserialize the JSON String using `JSON.deserializeUntyped` :-

JSONUntyped :-

```
public class JSONUntyped {
 public String result {get; set;}
 public JSONUntyped () {
 }
 String jsoninput = '{\n' +
 ' "description": "An appliance",\n' +
 ' "accessories": ["powercord", '+
 ' {"right": "door handle", '+
 ' {"left": "door handle2"}],\n' +
 ' "dimensions": '+
 ' {"height": 5.5, '+
 ' {"width": 3.0, '+
 ' {"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;
```

{

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

180





## 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 web services 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 service specified by the value of `url` that gets passed into the `getContent` method. This

## JQUERY (183 -187)

1. Introduction to Jquery
2. Example Programs
3. Refer to the real time scenarios using jquery workbook given by Benuel for more examples.

Satish Mwia

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();

}

}

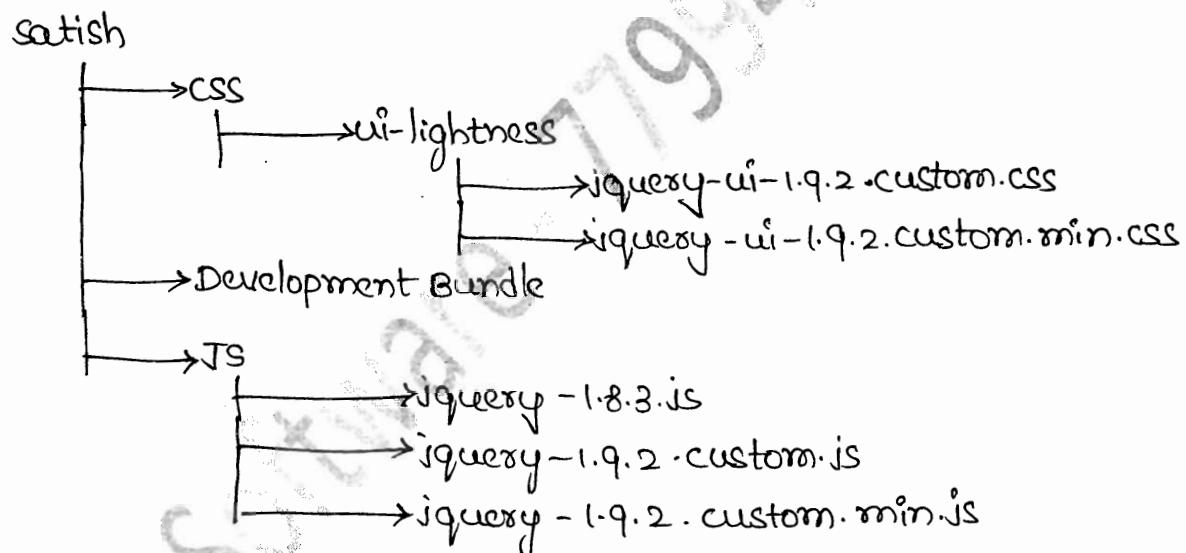
## usage of jquery:-

Jquery is fast and concise javascript framework created by John 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">

  if (!\$ = jQuery.noConflict());

    if (\$)

      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

```
if ($) {
```

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

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

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

→ If '\$' 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

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

→ Whatever the logic we want to write we should write with in the `function() {}`

Referring to a tag / components in visualforce :

1. Referring the html tag with id .

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

2. Referring the html tag with class .

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

if (document).ready (

functions)

```
 {
 $('#one').click
 (
 function()
 {
 $('#two').hide();
 }
);
 }
);
```

→ In the above code

```
if (document).ready(
 functions)
{
 .
}
);
```

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.

`if ("#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">

`if = jquery.noConflict();`

`if (if) {`

`alert('hello');`

`}`

`if (document).ready(function()`

`{`

`if ('#one').click(`

`function()`

`{`

`if ('#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 :-

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

slide Down(speed,callback) :-

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

Example :-

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

slide Up(speed,callback) :-

Example :-

```
if (document).ready(function()
 {
 if ("#one").click(function()
 {
 if ("#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 :-

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

## mouseleave(fn):-

### Example :-

```
if(if(document).ready(function()
{
 if ("#butn").click(function()
 {
 if ("p").mouseleave(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.

125 10-8

Example:-

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

187

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 expression.

Example:-

```
if(document).ready(function()
{
 if("#butn").click(function()
 {
 if("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 `if(A).append(B)`, in that instead of appending B to A, you're appending A to B.

Example:-

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



Q: How to turn 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 pagename it will open otherwise it will show a link to create a page on this name

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

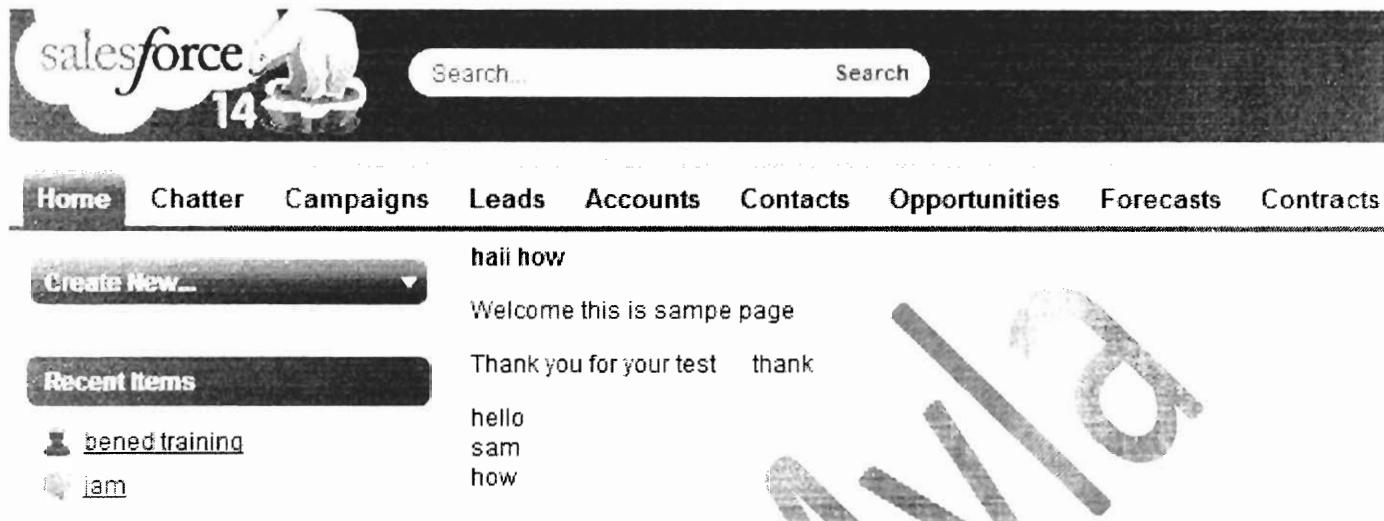
Attributes:

Id : This is used to recognize the component uniquely.  
 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  
 showHeader : This is a boolean value , if we give true it will show salesforce provided header on  
 your page false it will hide.  
 sidebar : It is a boolean value ,if we give true it shows sidebar components on your page.  
 setup : It is boolean value if we give true it will show setup 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> haii</h1>
 <h1> how </h1>
 <p> Welcome this is sample page</p>
 <p> Thank you for your test thank</p>
 hello
 sam
 how

</apex:page>
```

**Output Screen:**

salesforce 14

Search... Search

Home Chatter Campaigns Leads Accounts Contacts Opportunities Forecasts Contracts

Create New...

Recent Items

bened training jam

haii how  
Welcome this is sampe page  
Thank you for your test thank  
hello  
sam  
how

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

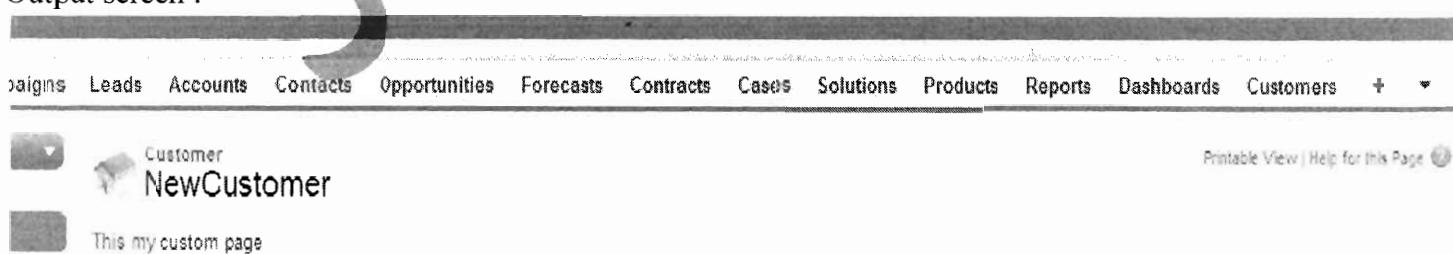
Attributes :

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 :



Lead Accounts Contacts Opportunities Forecasts Contracts Cases Solutions Products Reports Dashboards Customers +

Customer NewCustomer

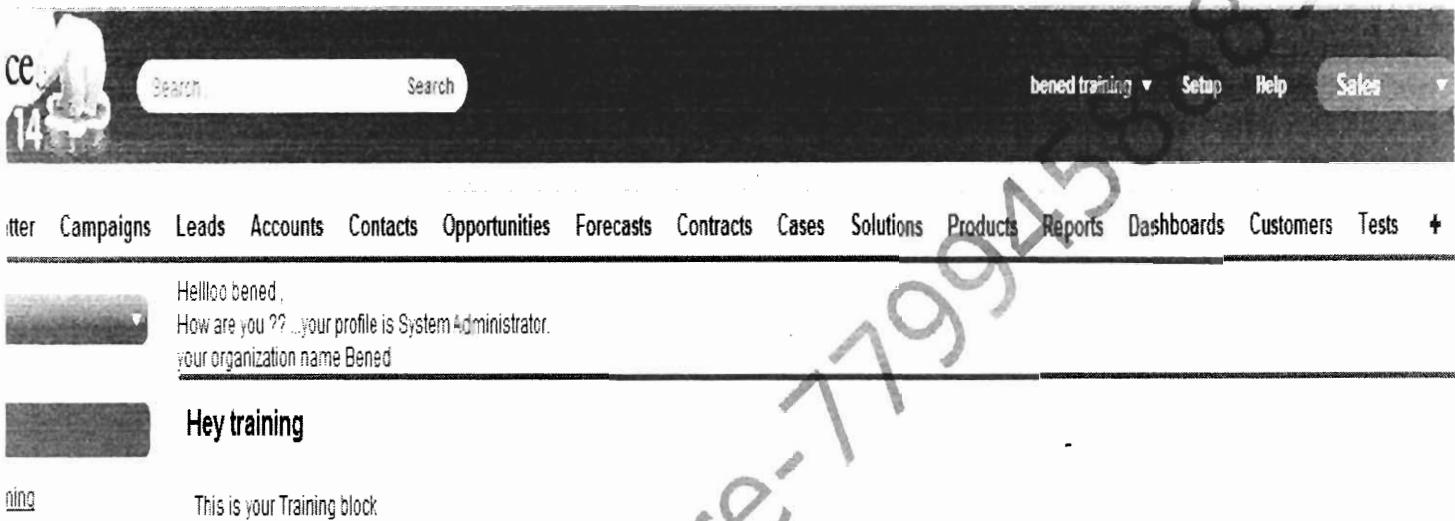
This my custom page

Printable View | Help for this Page

→ Refering 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 visualforce page

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

Example

```
{! 10}
{!10+20}
{'sam'}
{! TODAY()}
{!NOW()}
{!ISBLANK("")}
{!ISBLANK(Account.Name)}
```

Example :

```
<apex:page>
{!10+20}

 {!10}
 {'sam'}
{!TODAY()}

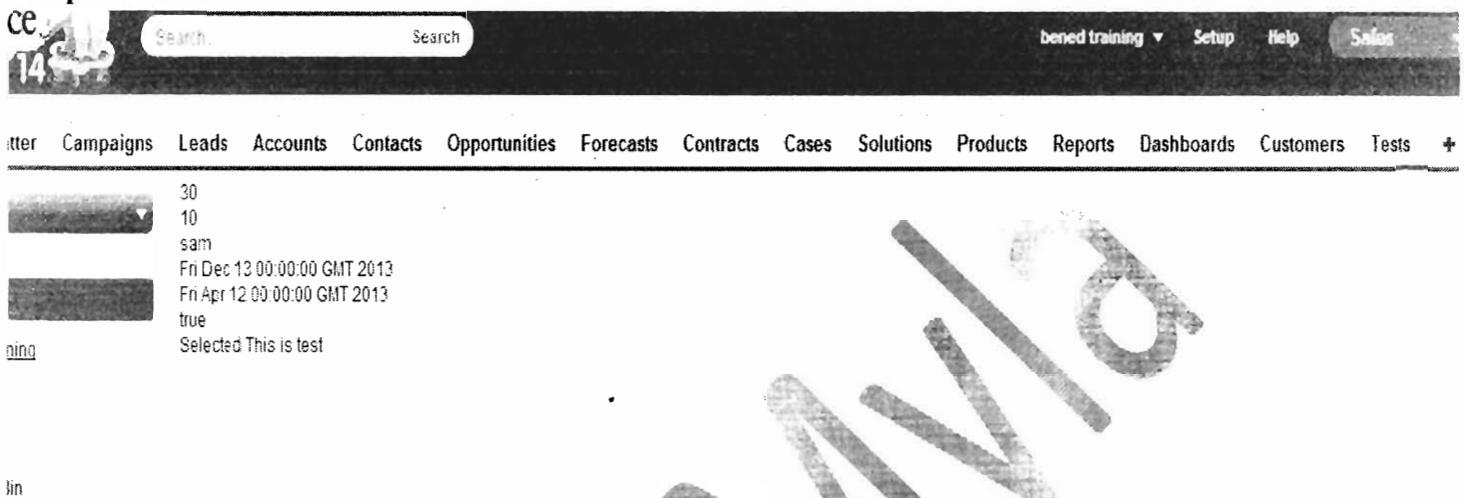
{!DATE(2013,4,12)}

{!ISBLANK("")}

{!IF(ISBLANK(),"Selected",'rejected')}
```

<apex:outputLabel rendered="{!!ISBLANK("")}">This is test </apex:outputLabel>  
</apex:page>

## OutputScreen :



### → <apex:pageMessage>

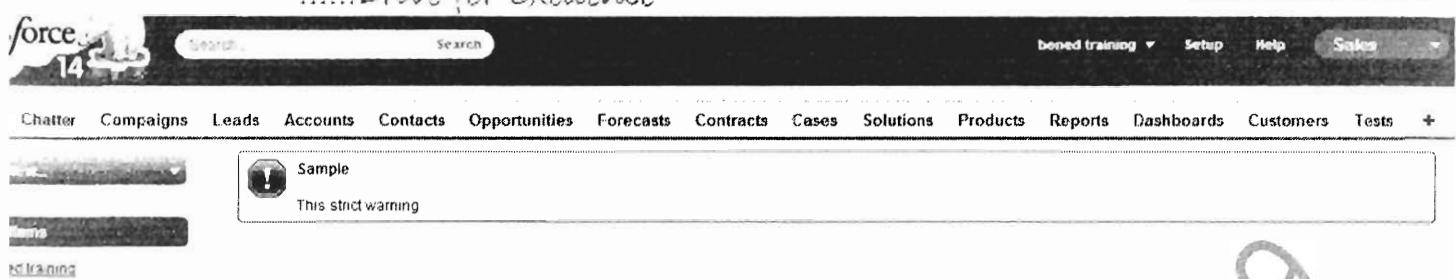
This is used for generating the error message .

- 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

 This is another sample
</apex:pageBlock>
<apex:pageBlock title="SecondBlock" dir="RTL" tabStyle="Loan_c">
 Hello This is sample

 This is another sample
</apex:pageBlock>
<apex:pageBlock title="ThirdBlock" rendered="{!!NOT(true)}">
</apex:pageBlock>
</apex:page>
```

## Output:

The screenshot shows a Salesforce page with a header containing 'Search', 'value Training', 'Setup', 'Help', and a 'ICICI Bank' button. Below the header is a navigation bar with links for 'Customers', 'Loans', 'Reports', 'one', 'Transactions', and a plus sign. The main content area has a title 'FirstBlock' and a help link 'needHelp ?'. Below the title, there is some sample text: 'Hello This is sample' and 'This is another sample'. At the bottom of the page, there is a large watermark reading 'Spiral IT Solutions'.

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

### Attributes:

- Value : The Name that should be displayed on the button should be given as value.
- Action : When we click on the button what action should be performed is given here.
- Disable : It is a Boolean values If we want to disable the button we can give true.
- 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>
```

This screenshot shows a modified version of the previous page. The 'click' button now has a different background color compared to the 'save' button. The rest of the page structure and content remain the same, including the 'FirstBlock' title and sample text.

**<apex:pageBlockButton>** : This will create buttons on the pageBlock.i.e 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
 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
 Iam here
</apex:pageBlock>
</apex:form>
</apex:page>
```

**OutputScreen:**

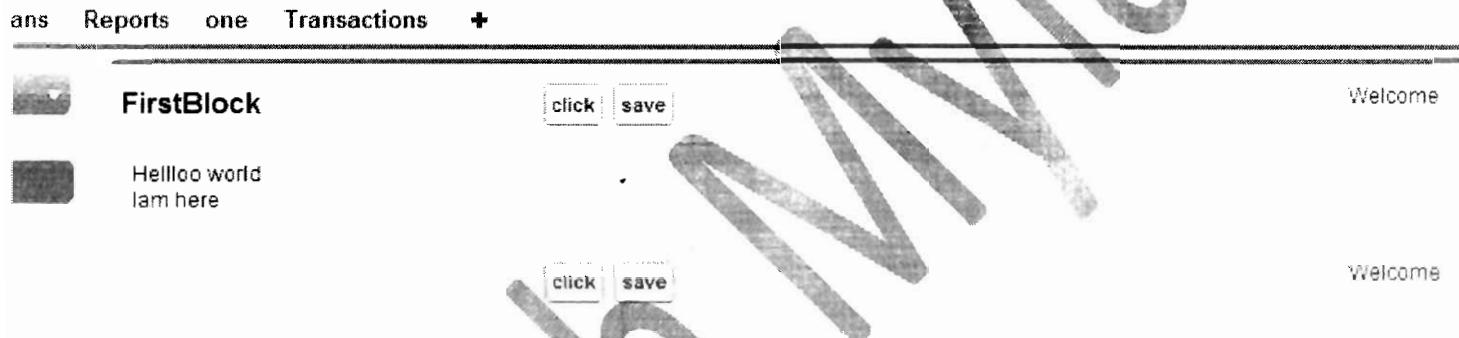
The screenshot shows a Salesforce page with a navigation bar at the top containing 'Loans', 'Reports', 'one', and 'Transactions'. Below the navigation bar are two sections labeled 'FirstBlock' and 'SecondBlock'. Each section contains the text 'Hellloo world' followed by 'Iam here' and a pair of buttons labeled 'click' and 'save'. The 'FirstBlock' section is positioned above the 'SecondBlock' section.

**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"/>
 Welcome
</apex:pageBlockButtons>
Helloo world
 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

Collapsible : 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
 Bened software
 </apex:pageBlockSection>
 <apex:pageBlockSection title="Second Section" collapsible="false">
 Hai
 Bened software
 </apex:pageBlockSection>
```



.....Drive for Excellence

```
<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>
```



Designs Leads Accounts Contacts Opportunities Forecasts Contracts Cases Solutions Products + ▾

FirstBlock

Hai  
Bened software

Hai  
Bened software

Enter Name Enter Name  
Enter Name

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:inputField>

10.<apex:selectRadio>

11.<apex:selectCheckboxes>

**Example:**

```
<apex:page>
<apex:form>
 <apex:outputLabel>Enter Name</apex:outputLabel>
 <apex:inputText tabIndex="2"/>

 <apex:outputLabel>Enter Password</apex:outputLabel>
 <apex:inputSecret tabIndex="1"/>

 <apex:outputLabel> Enter Sample</apex:outputLabel>
 <apex:inputText size="10"/>

 <apex:outputLabel>Enter Another</apex:outputLabel>
 <apex:inputText maxLength="4"/>
 <apex:inputCheckbox />

 <apex:outputLabel>Address</apex:outputLabel>
 <apex:inputTextarea cols="10" rows="3"/>
</apex:form>
</apex:page>
```

campaigns Leads Accounts Contacts Opportunities Forecasts Contracts Cases Solutions Products

Enter Name	<input type="text"/>
Enter Password	<input type="password"/>
Enter Sample	<input type="text"/>
Enter Another	<input type="text"/>
Address	<input type="text"/>

**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 />
```



.....Drive for Excellence

199  
salesforce.com  
PARTNER

```
<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:

ampaigns Leads Accounts Contacts Opportunities Forecasts Contracts Cases Solutions Products + ▾

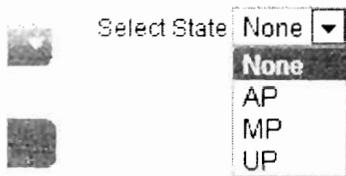
Registration

Enter Name  Enter Password

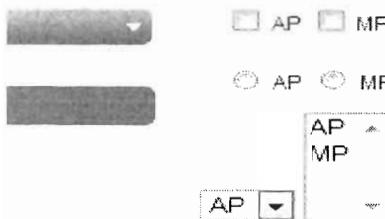
Enter Name   
Enter Password

Enter Name  Enter Password

```
<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 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 :

The screenshot shows a Salesforce Visualforce page with the title "MyBlock". At the top, there is a navigation bar with links for Accounts, Contacts, Opportunities, Forecasts, Contracts, Cases, Solutions, and Products. Below the navigation bar, there is a search bar with the placeholder "-None--". The main content area contains a section titled "MySection" which includes two input fields for "Industry". Each input field has a dropdown menu next to it with the option "-None--".

## Create a Account Home Page:

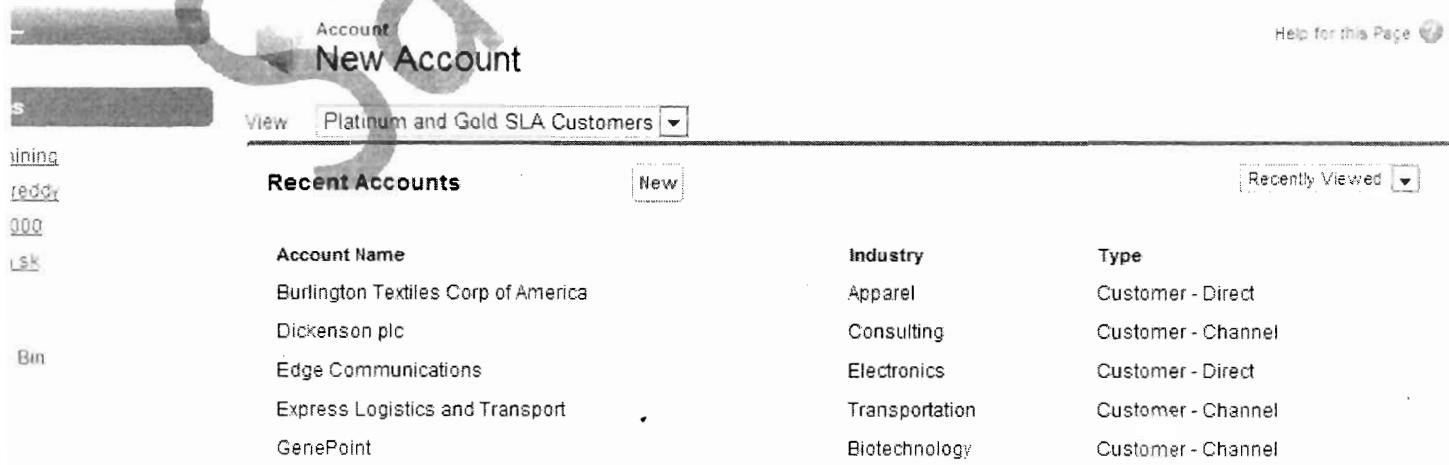
```
<apex:page standardController="Account" recordSetVar="items">
<apex:sectionHeader title="Account" subtitle="New Account" help="http://www.google.com"/>
<apex:form>
<apex:outputLabel>View</apex:outputLabel>

<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}" />

 <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>

</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 :



The screenshot shows the 'New Account' page in Salesforce. At the top, there's a header with 'Account' and 'New Account'. Below it, a navigation bar includes 'View' and a dropdown menu set to 'Platinum and Gold SLA Customers'. A toolbar on the left has user names: 'simina', 'reddy', 'gopu', and 'lsk'. On the right, there are links for 'Help for this Page' and 'Recently Viewed'. The main content area is titled 'Recent Accounts' and contains a table with five rows of account data:

	Account Name	Industry	Type
Burlington Textiles Corp of America	Apparel	Customer - Direct	
Dickenson plc	Consulting	Customer - Channel	
Edge Communications	Electronics	Customer - Direct	
Express Logistics and Transport	Transportation	Customer - Channel	
GenePoint	Biotechnology	Customer - Channel	

**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}" />

 <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>

 </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>
```

View Platinum and Gold SLA Customers
**Recent Accounts**
New
Recently Viewed

Account Name	Industry	First Name
Burlington Textiles Corp of America	Apparel	Jack
Dickenson plc	Consulting	Andy
Edge Communications	Electronics	Rose
Express Logistics and Transport	Transportation	Sean
GenePoint	Biotechnology	Babara
		Josh
		Edna

→ Creating output Labels and TextFields 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:outputLabel 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 fashion when the page is loaded

### Example :

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

Variables 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';
```

emp.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, onchange, ondblclick etc.

### Example :

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

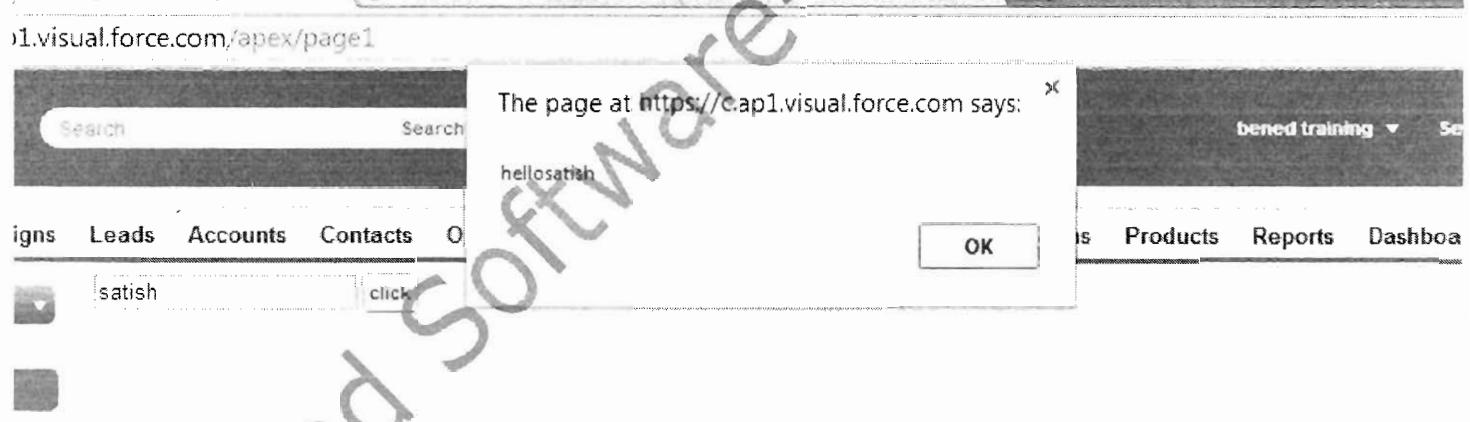
Output Screen :



**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 >
```

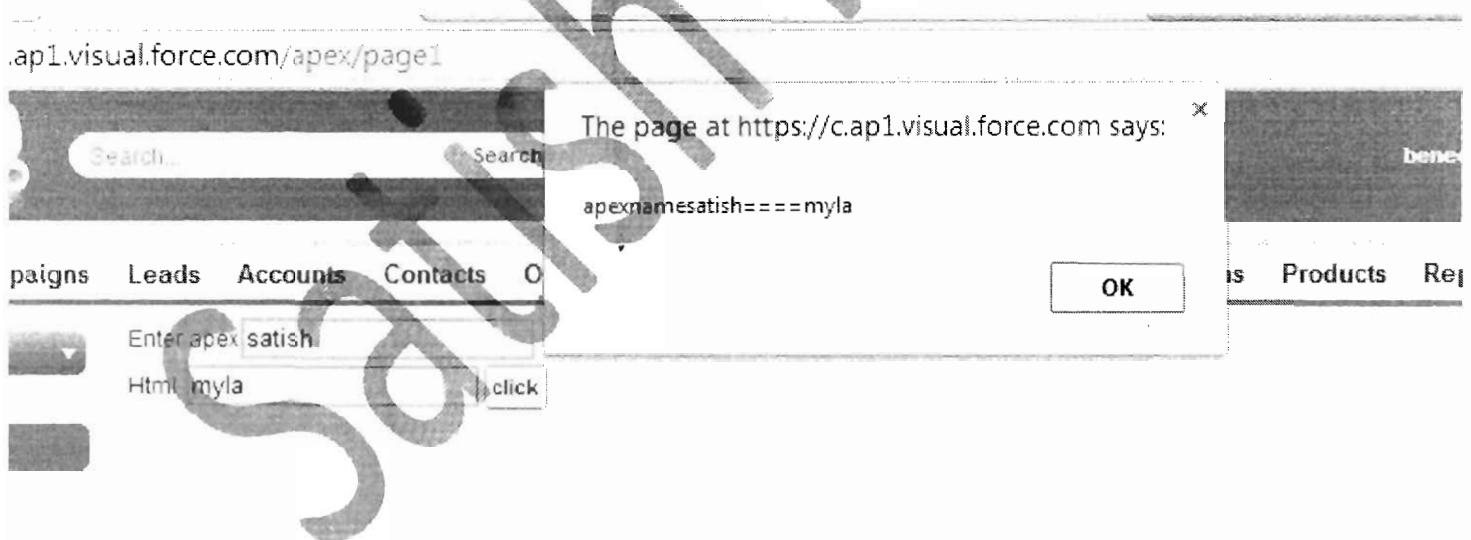
208

```
<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" />

<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 referred 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

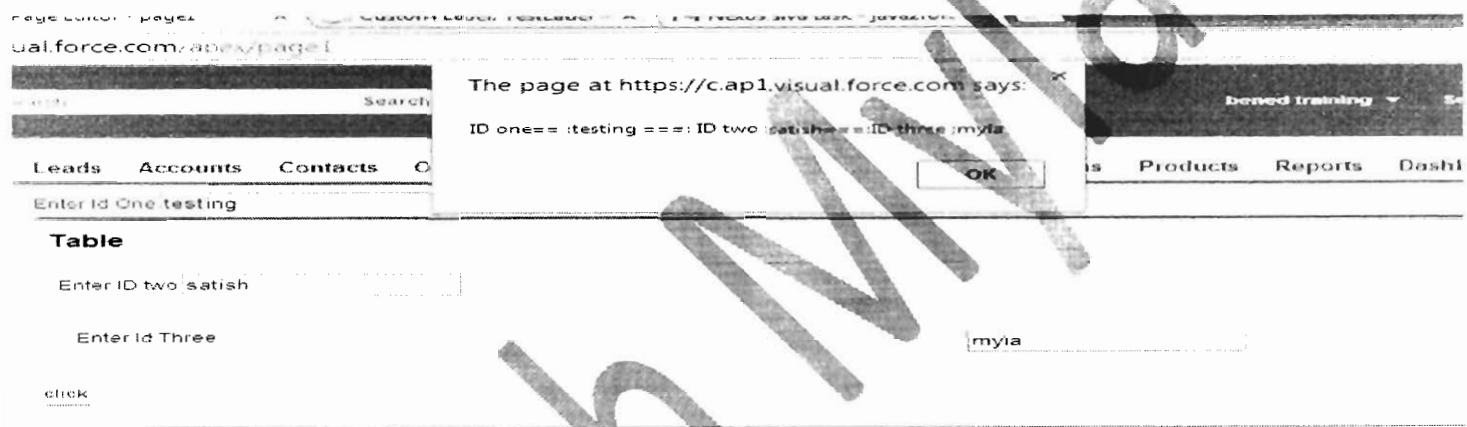
```
<apex:page>
 <apex:form>
 <script>
 function show()
 {
 var one=document.getElementById('{$Component.one'}).value;
 var two=document.getElementById('{$Component.PB.two'}).value;
 var three=document.getElementById('{$Component.PB.PBS.three'}).value;
 alert('ID one== :'+one+'==: ID two :'+two+'==:ID 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

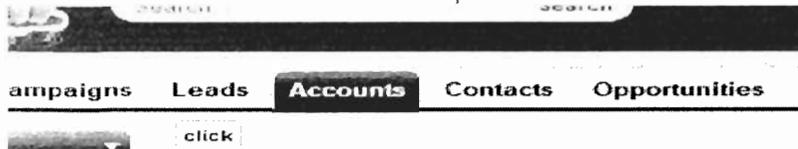
Page: JExample

```

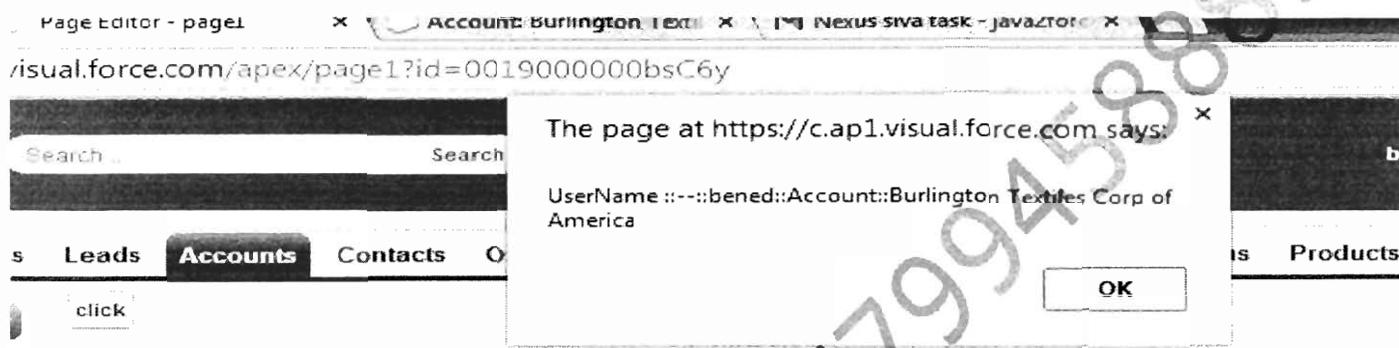
<apex:page>
 <apex:form>
 <script>
 function show()
 {
 var username='!$User.firstName';
 var accountname='!Account.Name';
 alert('UserName ::--::'+username+':Account::'+ accountname);
 }
 </script>
 <apex:commandButton value="click" onclick="show()"/>
 </apex:form>
</apex:page>

```

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



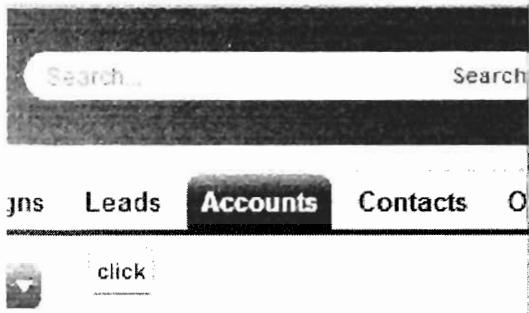
When we click on the button



**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>
```

L.visual.force.com/apex/page1?id=0019000000bsC6y



The page at https://c.ap1.visual.force.com says:

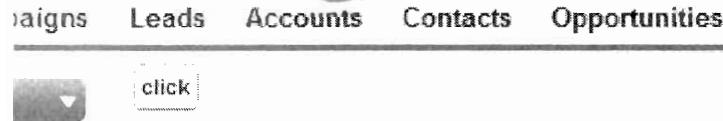
UserName --:bened--::Account Name- :Burlington Textiles  
Corp of America

OK

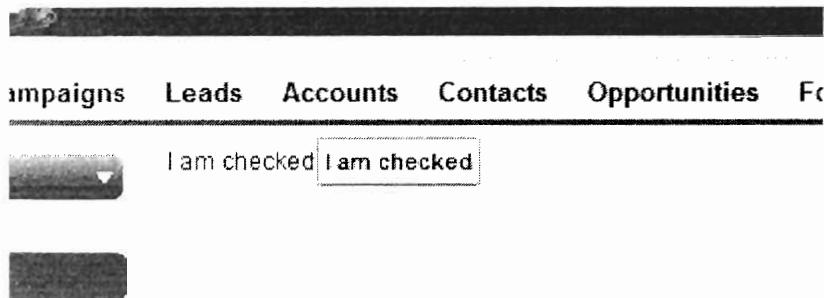
**Example:** Change the CSS properties /component values of the Visualorce page using javascript

```
<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 value 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 accept 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 indicates 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 followed 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 accept 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 = " does 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

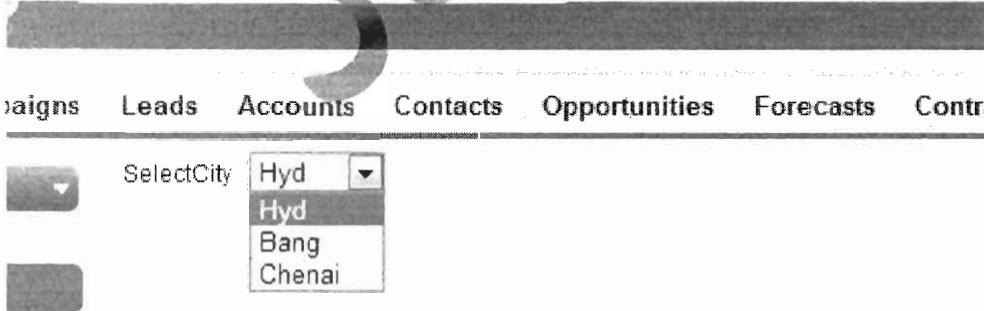
1.apex:selectList

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>
```



of

**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" recordServerVar="items" recordserVar="` will fetches the records from the standard controller based on the recently used filter in the Standardcontroller.

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 :

The screenshot shows the Salesforce interface with the 'Accounts' tab selected in the top navigation bar. Below the navigation bar, there is a search bar and a filter section labeled 'MyIndustry' with options: Transportation, Biotechnology, and Hospitality. The main content area displays a list of account records:

Account Name
Express Logistics and Transport
GenePoint
Grand Hotels & Resorts Ltd
jam

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>
```

Accounts Home Help for this Page

View: All Accounts Edit | Create New View

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Other All

Action	Account Name	Account Site	Billing State/Province	Phone	Type	Account Owner Alias
Edit   Del	<a href="#">Burlington Textiles Corp of America</a>	NC		(336) 222-7000	Customer - Direct	btrai
Edit   Del	<a href="#">Dickenson plc</a>	KS		(785) 241-6200	Customer - Channel	btrai
Edit   Del	<a href="#">Edge Communications</a>	TX		(512) 757-6000	Customer - Direct	btrai
Edit   Del	<a href="#">Express Logistics and Transport</a>	OR		(503) 421-7800	Customer - Channel	btrai
Edit   Del	<a href="#">GenePoint</a>	CA		(650) 867-3450	Customer - Channel	btrai
Edit   Del	<a href="#">Grand Hotels &amp; Resorts Ltd</a>	IL		(312) 596-1000	Customer - Direct	btrai
Edit   Del	<a href="#">jam</a>			(014) 427-4427		btrai
Edit   Del	<a href="#">Pyramid Construction Inc.</a>	CA		(415) 901-7000	Customer - Channel	btrai
Edit   Del	<a href="#">sForce</a>			(212) 842-5500	Customer - Direct	btrai
Edit   Del	<a href="#">United Oil &amp; Gas Corp.</a>	NY				btrai

<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>

```

Action	Account Name
<a href="#">edit</a>	Burlington Textiles Corp of America
<a href="#">edit</a>	Dickenson plc
<a href="#">edit</a>	Edge Communications
<a href="#">edit</a>	Express Logistics and Transport
<a href="#">edit</a>	GenePoint
<a href="#">edit</a>	Grand Hotels & Resorts Ltd
<a href="#">edit</a>	jam
<a href="#">edit</a>	Pyramid Construction Inc.
<a href="#">edit</a>	sForce
<a href="#">edit</a>	United Oil & Gas Corp.
<a href="#">edit</a>	United Oil & Gas, Singapore

**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="onchange" rendered="pb"/>
 </apex:selectList>
 <apex:pageBlock title="Recent Accounts" id="pb">
 <apex:pageBlockButtons>
 <apex:commandButton value="New" action="{!!create}">

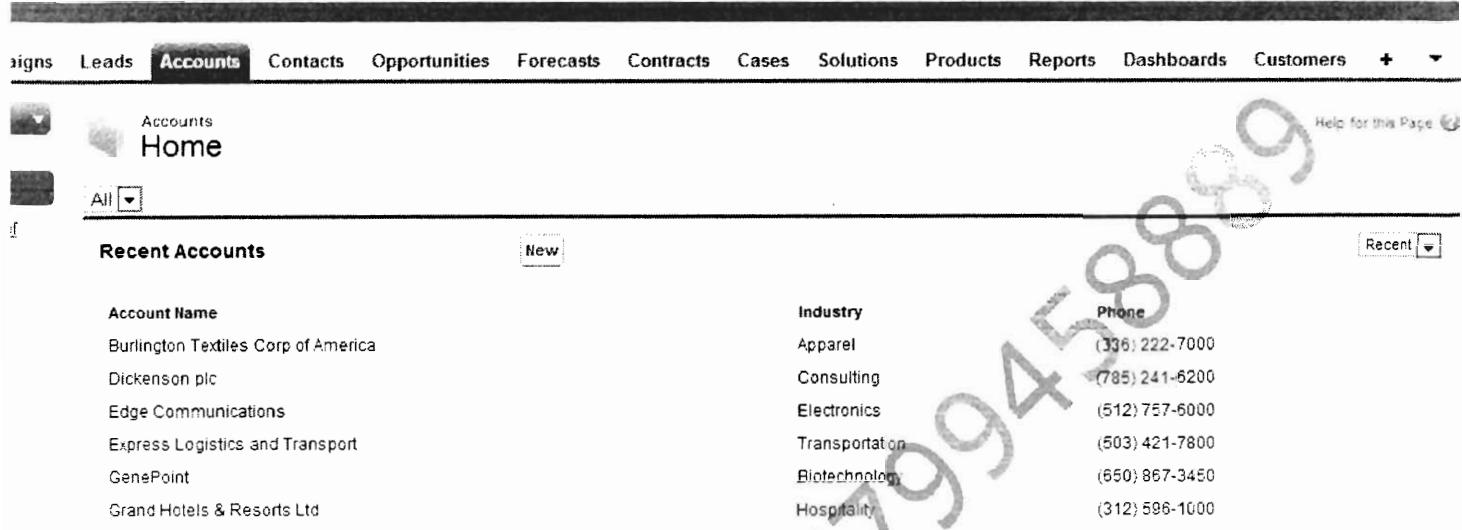
 <apex:selectList size="1">
 <apex:selectOption itemLabel="Recent" itemValue="one"/>
 <apex:selectOption itemLabel="Last" itemValue="two"/>
 </apex:selectList>

 </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>

```

&lt;/apex:page&gt;

Output Screen :



Account Name	Industry	Phone
Burlington Textiles Corp of America	Apparel	(336) 222-7000
Dickenson plc	Consulting	(785) 241-6200
Edge Communications	Electronics	(512) 757-6000
Express Logistics and Transport	Transportation	(503) 421-7800
GenePoint	Biotechnology	(650) 867-3450
Grand Hotels & Resorts Ltd	Hospitality	(312) 596-1000

**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)"/>

```

```

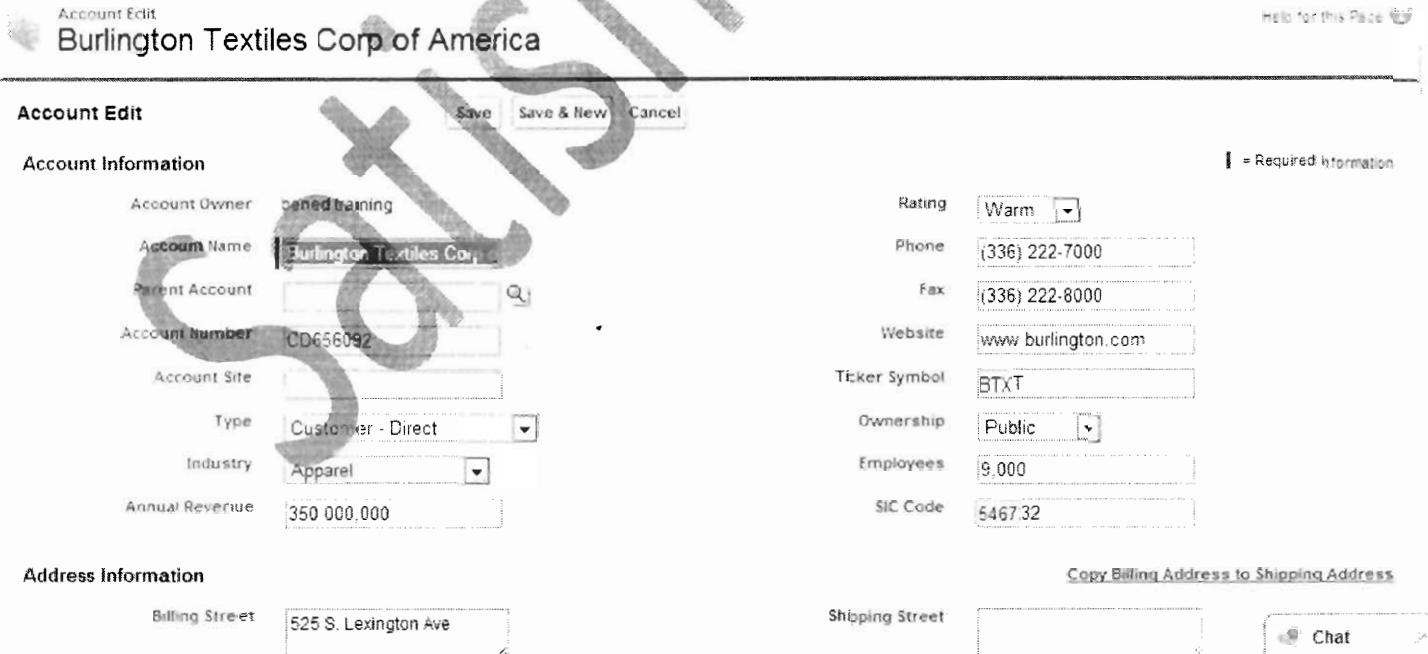
</apex:column>
<apex:column value="{!!a.name}" />
</apex:pageBlockTable>
</apex:pageBlock>
</apex:form>
</apex:page>

```



The screenshot shows the Salesforce Accounts page. The top navigation bar includes links for Campaigns, Leads, Accounts, Contacts, Opportunities, Forecasts, Contracts, Cases, Solutions, Products, Reports, Dashboards, Customers, and a plus sign for creating new records. Below the navigation is a search bar with the placeholder 'All'. The main content area displays a list of accounts with columns for Action, Account Name, and Edit | Delete. The accounts listed are Burlington Textiles Corp of America, Dickenson plc, Edge Communications, Express Logistics and Transport, GenePoint, Grand Hotels & Resorts Ltd, Jam, Pyramid Construction Inc., and sForce.

When we click edit button of Burlington Textiles corp of Amreica



The screenshot shows the Account Edit page for 'Burlington Textiles Corp of America'. The page has a header with 'Account Edit' and 'Help for this Page'. It features two main sections: 'Account Information' and 'Address Information'. The 'Account Information' section contains fields for Account Owner (set to 'bened training'), Account Name ('Burlington Textiles Corp of America'), Parent Account (dropdown menu), Account Number ('CD656092'), Account Site (dropdown menu), Type ('Customer - Direct'), Industry ('Apparel'), and Annual Revenue ('350 000.000'). The 'Address Information' section includes fields for Billing Street ('525 S. Lexington Ave'), Shipping Street (empty field), and a 'Copy Billing Address to Shipping Address' checkbox. On the right side, there are additional fields: Rating ('Warm'), Phone ('(336) 222-7000'), Fax ('(336) 222-8000'), Website ('www.burlington.com'), Ticker Symbol ('BTXT'), Ownership ('Public'), Employees ('9,000'), and SIC Code ('546732'). A note at the bottom right indicates that a required field is marked with a red asterisk (\*).

When we click on delete button of Iam record

Accounts  
Home

Tell me mo

View: All Accounts  Edit | Create New View

### Recent Accounts

New

Account Name

Burlington Textiles Corp of America

Billing City

Burlington

Phone

(336) 222-7000

### Reports

Active Accounts

Accounts with last activity > 30 days

### Tools

Import My Accounts & Contacts

Import My Organization's Accounts & Contacts

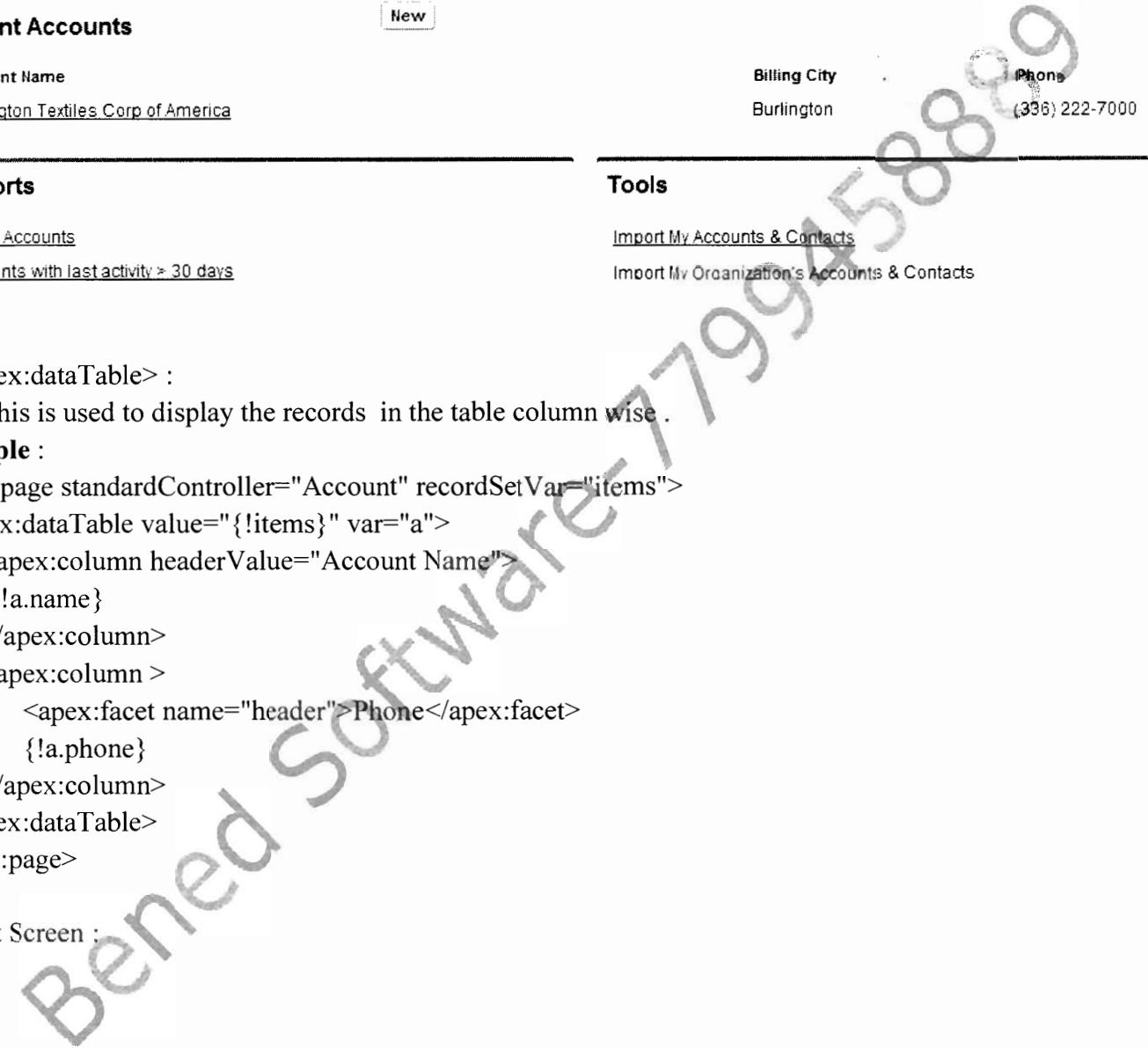
→<apex:dataTable> :

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 :



**Accounts** Leads Contacts Opportunities Forecasts Contracts

	Account Name	Phone
	Burlington Textiles Corp of America	(336) 222-7000
	Dickenson plc	(785) 241-6200
	Edge Communications	(512) 757-6000
	Express Logistics and Transport	(503) 421-7800
	GenePoint	(650) 867-3450
	Grand Hotels & Resorts Ltd	(312) 598-1000
	Pyramid Construction Inc.	(014) 427-4427
	sForce	(415) 901-7000
	United Oil & Gas Corp.	(212) 842-5500
	United Oil & Gas, Singapore	(650) 450-8810
	United Oil & Gas, UK	+44 191 4956203
	University of Arizona	(520) 773-9050

## →&lt;apex:dataList&gt; :

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 :

igns Leads **Accounts** Contacts Opportunities Forecasts C

- Burlington Textiles Corp of America
- Dickenson plc
- Edge Communications
- Express Logistics and Transport
- GenePoint
- Grand Hotels & Resorts Ltd
- Pyramid Construction Inc.
- sForce
- United Oil & Gas Corp.
- United Oil & Gas, Singapore
- United Oil & Gas, UK
- University of Arizona

➔<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 :

igns Leads **Accounts** Contacts Opportunities Forecasts Contracts Cases Solutions Products Reports Dashboards Customers Tests +

Burlington Textiles Corp of America Dickenson plc Edge Communications Express Logistics and Transport GenePoint Grand Hotels & Resorts Ltd Pyramid Construction Inc sForce United Oil & Gas Corp. United Oil & Gas, Singapore United Oil & Gas, UK University of Arizona

#### →<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.

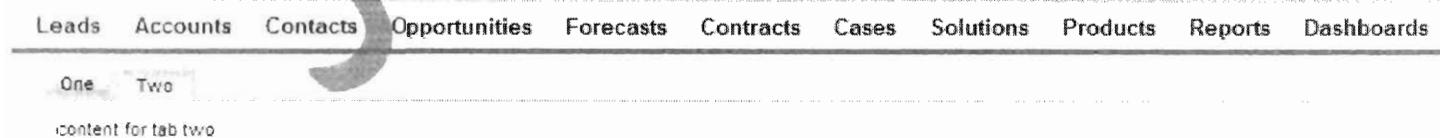
Height : The height of the tab bar expressed as a percentage of the available vertical space.

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 :

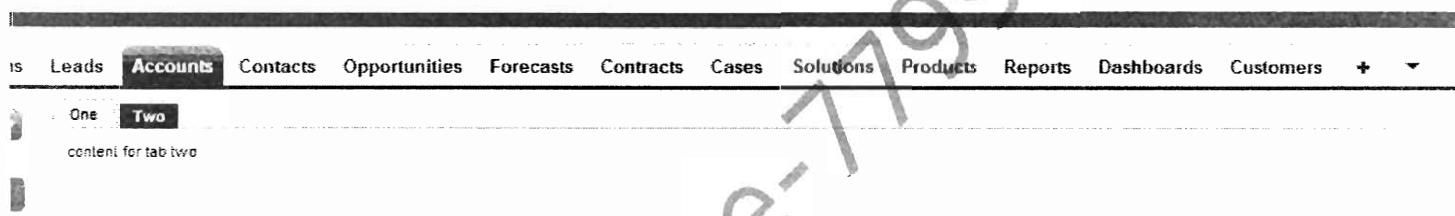
```
<apex:page standardController="Account" showHeader="true">
```

Bened Software, Pet (P) Ltd, 8-212/9/14<sup>th</sup> Floor, DTC Office Building, Near Saradhi Studio, Ameerpet -38  
Bened Software, Pet (P) Ltd, 8-212/9/14<sup>th</sup> Floor, DTC Office Building, Near Saradhi Studio, Ameerpet, Hyd-38.  
Ph: 91 700 658 893 555 566 Email: [training@benedsoft.com](mailto:training@benedsoft.com) <http://www.benedsoft.com>

```
<!-- 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 :



Account

**Burlington Textiles Corp of America**

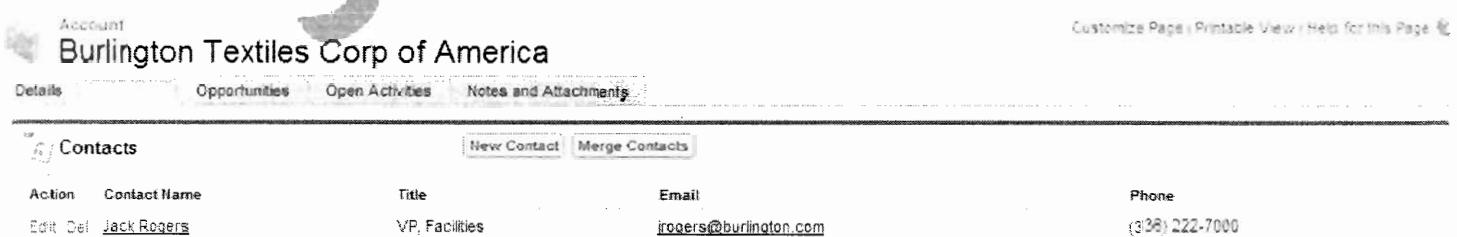
Customize Page | Printable View | Help for this Page

Contacts | Opportunities | Open Activities | Notes and Attachments

**Account Detail**

Account Owner	<a href="#">bened training [Change]</a>	Rating	Warm
Account Name	Burlington Textiles Corp of America <a href="#">[View Hierarchy]</a>	Phone	(336) 222-7000
Parent Account	CD656092	FAX	(336) 222-8000
Account Number	CD656092	Website	<a href="http://www.burlington.com">http://www.burlington.com</a>
Account Site	Type: Customer - Direct Industry: Apparel	Ticker Symbol	BTXT
Annual Revenue	\$350,000.000	Ownership	Public
Billing Address	525 S. Lexington Ave Burlington, NC 27215 USA	Employees	9,000
Customer Priority		SIC Code	546732
SLA Expiration Date	11/16/2012	Shipping Address	
Number of Locations	8	SLA	Silver
Active		SLA Serial Number	S267
Created By	<a href="#">bened training</a> 10/21/2013 7:13 PM	Upsell Opportunity	Maybe
Description	Billing	Last Modified By	<a href="#">bened training</a> 10/21/2013 7:13 PM
Custom Links			

When we click on the Contacts :



Account

**Burlington Textiles Corp of America**

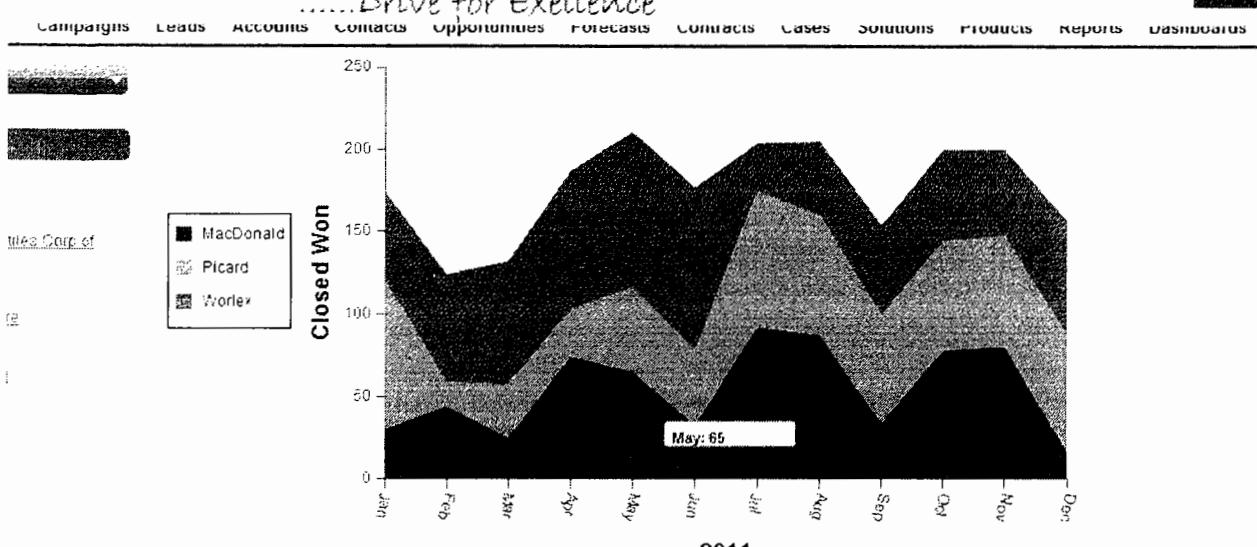
Customize Page | Printable View | Help for this Page

Details | Opportunities | Open Activities | Notes and Attachments

**Contacts**

Action	Contact Name	Title	Email	Phone
Edit   Del	<a href="#">Jack Rogers</a>	VP, Facilities	<a href="mailto:jrogers@burlington.com">jrogers@burlington.com</a>	(336) 222-7000

```
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>
```



### Scenario 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:form>

```

```
<apex:pageBlockSection title="Select Files" rendered="{!!IF(Count != null &&
Count != ", 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>
```

Bened Software




How many files you want to upload?

--None--

1  
2  
3  
**4**  
5  
6  
7  
8  
9  
10

Org of


Select 2 files




How many files you want to upload?

2

 PWF.txt

 PWF.txt

**Scenario 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.getLatLang(
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 : <i> {"+!Account.Name} </i>
Address : "+address);
}
}
);
}
}
</script>

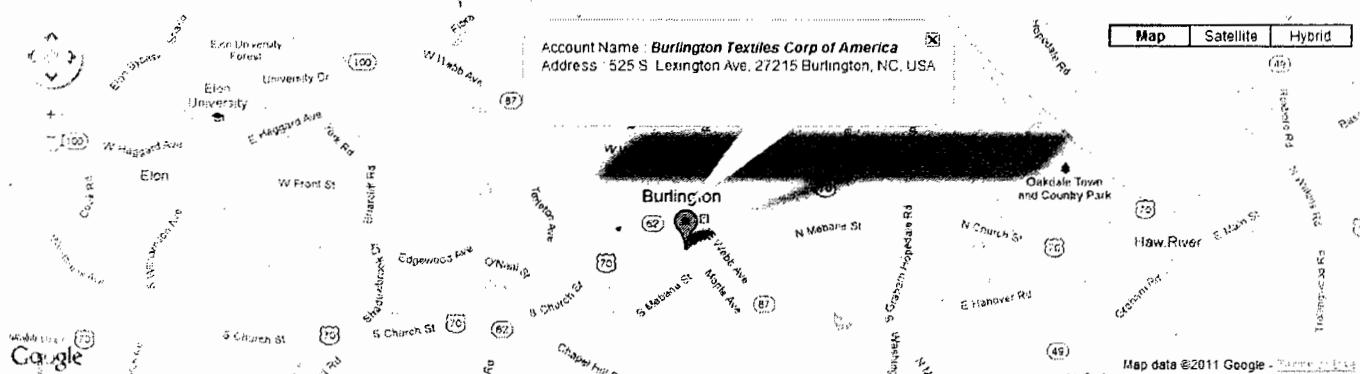
```

```

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

```

Give the id of the Account Record id .



### Scenario 15::Create Dynamic Record in Subject 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 GlobalDescribe 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 ;
```

```
}
```

```
}
```

```
}
```

gns Leads Accounts Contacts Opportunities Forecasts Contracts Cases Solutions Products Reports Dashboards Customers +

Save

Enter Object Name  Enter Name for Record

Save

Satish Example

Customize Page | Edit Layout | Printable View |

Hide Feed Post File New Task More

View All Updates

There are no updates

Check for List Apex Classes

Contacts (0) Opportunities (0) Cases (0) Open Activities (0) Activity History (0) Tickets & Attachments (0) Products (0) Account Types (0)

Account Detail

Account Owner bened training [Change]  
Account Name Satish Example [View Hierarchy]  
Parent Account  
Account Number

Rating  
Phone  
Fax  
Website

## 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 within 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

overline

line-through

underline

text-transform : lowercase

text-transform : uppercase

text-transform : capitalize.

→ In the inline CSS we are going to define the property within the component itself.

Ex:-

```
<apex:page>
```

```
 <apex:form>
```

```
 <h1 style="text-decoration:line-through; background-color: red;
 text-transform: lowercase;"> Welcome to this </h1>

```

```
 <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

→ Whenever we want to use background image in the CSS.

- i) 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 :-

font

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 within the visual force page then it is called as internal css.

→ The internal css should be defined within 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;
```

```
} #id
```

```
{
```

```
.two
```

```
{
```

```
 color: red;
```

```
.className
```

```
{
```

```

 h1 {
 color: green;
 }
}

h1 - Html tag element.

```

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: 50px;

```

```
background-color: #FFFFFF;
```

```
{
```

& recent

```
{
```

```
margin: 20px;
```

```
padding: 10px;
width : 150px;
height : 20px;
background-color : #81D4FA;
}
</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 alt = "square/1383--/download" style =
 "width:10px; height:10px; padding:5px;
 float:left;"/>
 <apex:commandLink value = "?!a.name?" action = "?!/a.edit"/>
 </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; /*
```

- out co

{

```
margin: 10px;
padding: 10px;
width: 250px;
height: 50px;
background-color: #FFFFFF;
-webkit-transform: rotateY(40deg);
```

- decent

{

```
margin: 20px;
padding: 10px;
width: 150px;
height: 20px;
background-color: #81DAF5;
```

```
border-radius: 10px;
}
```

Step2:- Load this mystyle.css file as static resource in Salesforce.

Setup → Build → Develop → Static Resource → New

filename: 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="!$resource.satishcss"/>
 <div>
 <div class="recent">
 <apex:outputLabel style="font-size:20px;">Recent Items</apex:outputLabel>
 </div>
 <div class="one">
 <apex:datatable value="!$items" var="a" rows="5">
```

265

```
first = "3" > |
<apex:column> .
<apex:image url = "resource/1383105071000/download" style = "width:10px; height:10px; 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 existing 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 of 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.

Ex:- 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="fm">
 <apex:pageBlock id="pb">
 <Script type="text/javascript">
 function show()
 {
 var name = document.getElementById('${!$Component.one$}');
 name.value;
 var myname = document.getElementById('pg;fm;pb;one').value;
 alert(name);
 }
 </script>
 <apex:commandButton value="click" action="${!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 user name </apex:outputLabel>

<apex:inputText id = "uid"/>

<apex:outputLabel> Enter password </apex:outputLabel>

<apex:inputText id = "pwd"/>

<apex:CommandButton value = "Login" action = "f!show"/>

</apex:pageblock>

<Script>

function show()

Var user = document.getElementById('{\$!\$Component.pb.  
uid}').value;

Var passwd = document.getElementById('{\$!\$Component.pb.  
pwd}').value;

```

if(password == '' || user == '')
 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 showc )

{

    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="click me" onclick="showc()"/>

</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, text id)
{
 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 controllers 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.

270

→ method is the name of your Apex controller method you are calling.

→ parameters is the comma-separated 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 remotemethod :-

→ In your Controllers, your Apex method declaration is preceded with the @RemoteAction annotation like this,

@RemoteAction

global static String getItemId(String objectName){ ... }

~~Tip:-~~ → Your method can take Apex primitives, collections, typical and generic objects, and user-defined Apex classes & interfaces as arguments.

Ex:-

~~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;

}  
{

Step 2: Load the apex class into the visualforce page. 27

→ If we want to invoke the remote action we have to call a method

visualforce.Remoting.Manager.InvokeAction('{\$!\$RemoteAction.

Example.show}');

↓                    ↓

Controller class name      Remote method name.

function(result, event)

{  
= → Response handle.

}

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
 }
 }
}

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

Second way of calling method in Javascript using VF page:-

Example. myfun(parameters, function(result, event))

```
if (event.status)
{
 else if (event.type == "Exception")
 {
 });
}
```

## Javascript Remoting Example:-

272

```
<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>
```

Satish K. Awad

## Force.com IDE Installation

The Force.com IDE is available as a plug-in for the industry-leading, open-source Eclipse project.

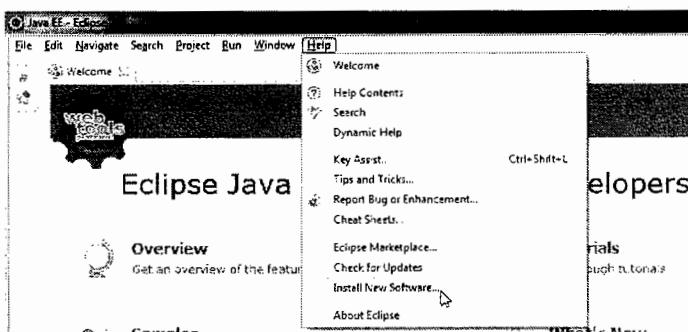
These instructions show you how to install the Force.com IDE into your existing Eclipse distribution or upgrade from a previous version.

### Prerequisites

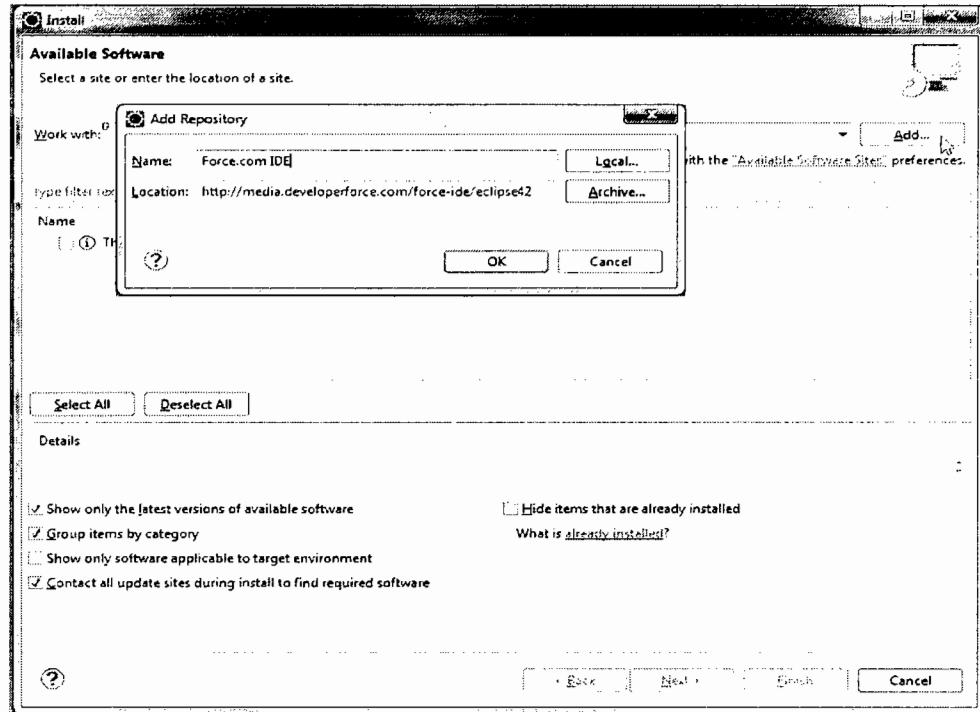
- Java SE Runtime Environment 6 (v1.6) or later
- Eclipse 4.2 "Juno" (Eclipse 4.2 download site) or Eclipse 4.3 "Kepler" (Eclipse 4.3 download site) - The 'Eclipse IDE for Java Developers' distribution is strongly recommended.

### Installation Steps

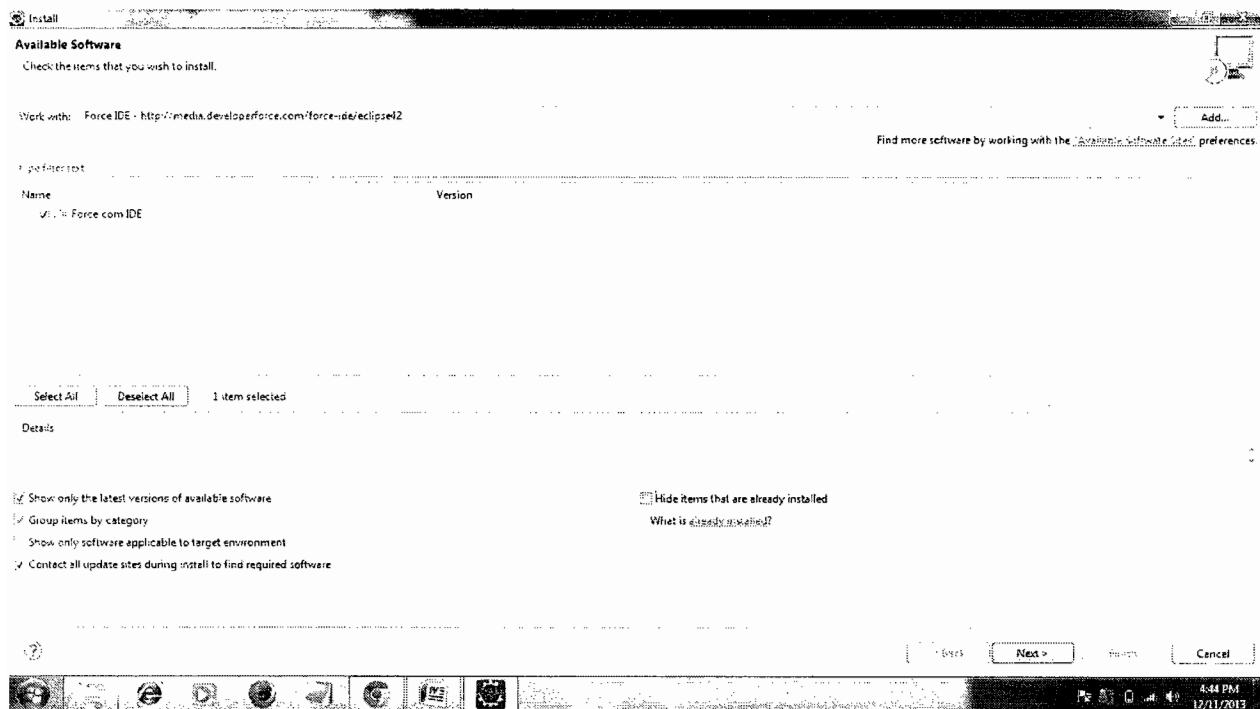
1. Launch Eclipse and click **Help > Install New Software....**



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**.





salesforce.com  
PARTNER

Install

Install Details

Review the items to be installed.

Name	Version	Id
Force.com IDE	29.0.0.201310181333	com.salesforce.ide.feature.feature.group

Size: Unknown

Details



Install

Review Licenses

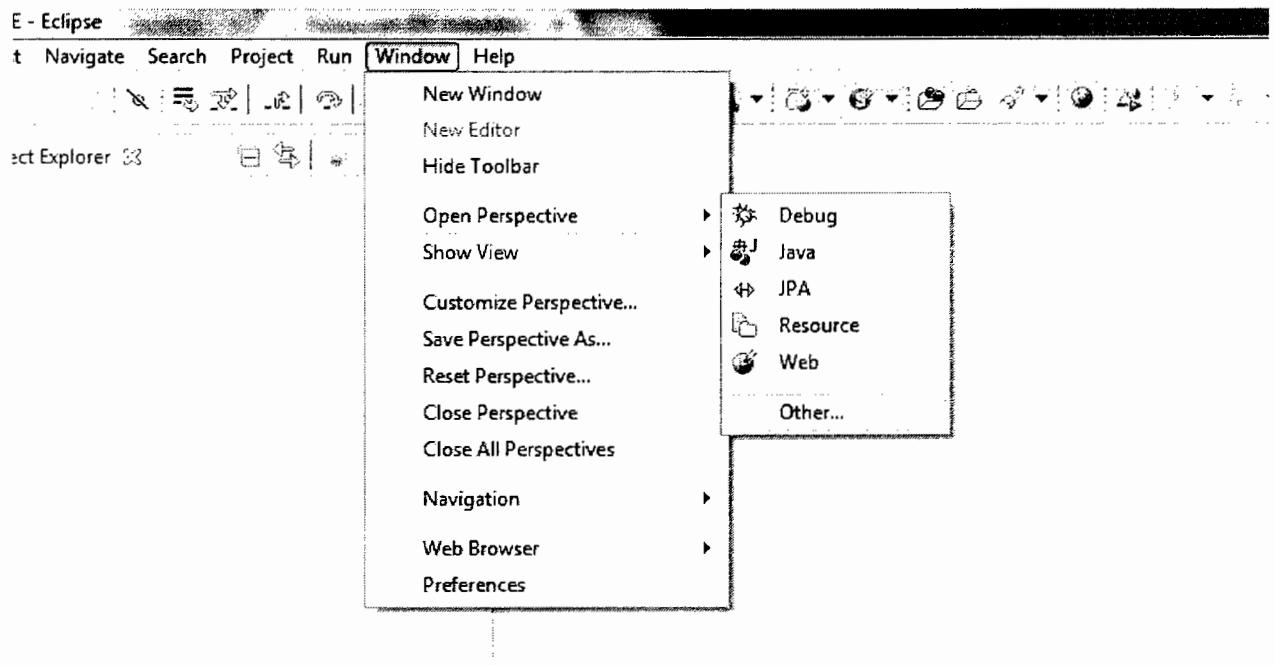
Licenses must be reviewed and accepted before the software can be installed.

License text (for Force.com IDE 29.0.0.201310181333):

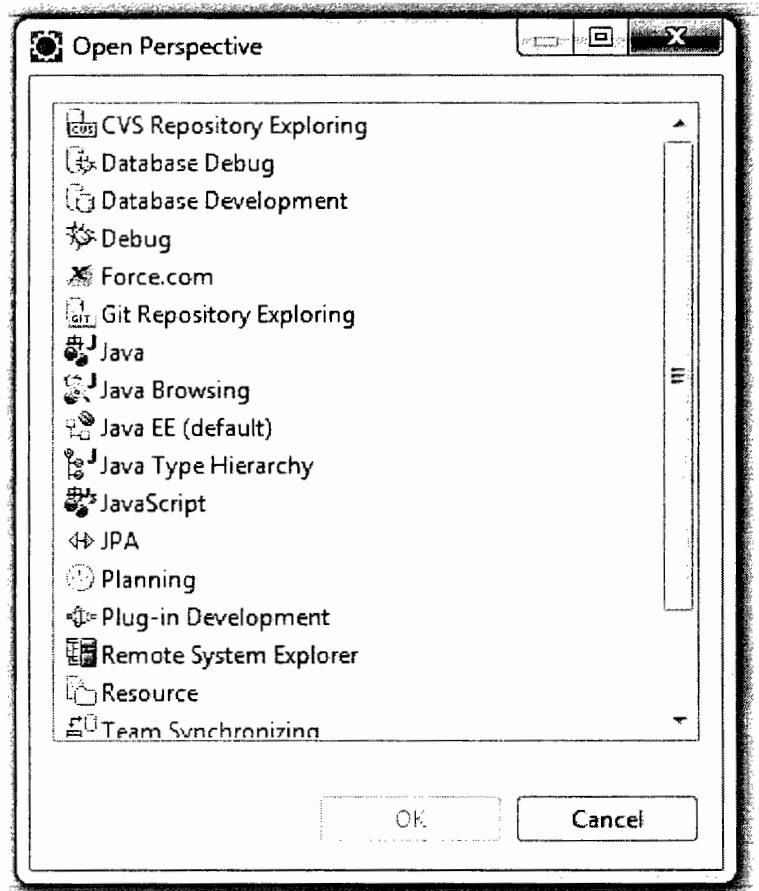
End User License Agreement  
Force.com IDE version 29.0  
Except as described below, com.salesforce.ide,  
com.salesforce.ide.apis,  
com.salesforce.ide.branding,  
com.salesforce.ide.core,  
com.salesforce.ide.deployment,  
com.salesforce.ide.documentation,  
com.salesforce.ide.editor,  
com.salesforce.ide.editor.intro,  
com.salesforce.ide.schemaparser,  
com.salesforce.ide.core.compatibility,  
com.salesforce.ide.core.compatibility.oauth.platformdb  
and com.salesforce.ide.upgrade.jar files are Copyright (c) 2005-2013,  
salesforce.com, inc. All rights reserved.  
Redistribution and use in source and binary forms, with or without  
modification, are permitted provided that the following conditions  
are met:  
\* Redistributions of source code must retain the above copyright  
notice, this list of conditions and the following disclaimer.  
\* Redistributions in binary form must reproduce the above copyright  
notice, this list of conditions and the following disclaimer  
in the documentation and/or other materials provided with the  
distribution.  
\* Neither the name of salesforce.com, inc. nor the names of its  
contributors may be used to endorse or promote products derived  
from this software without specific prior written permission.  
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

I accept the terms of the license agreement  
 I do not accept the terms of the license agreement



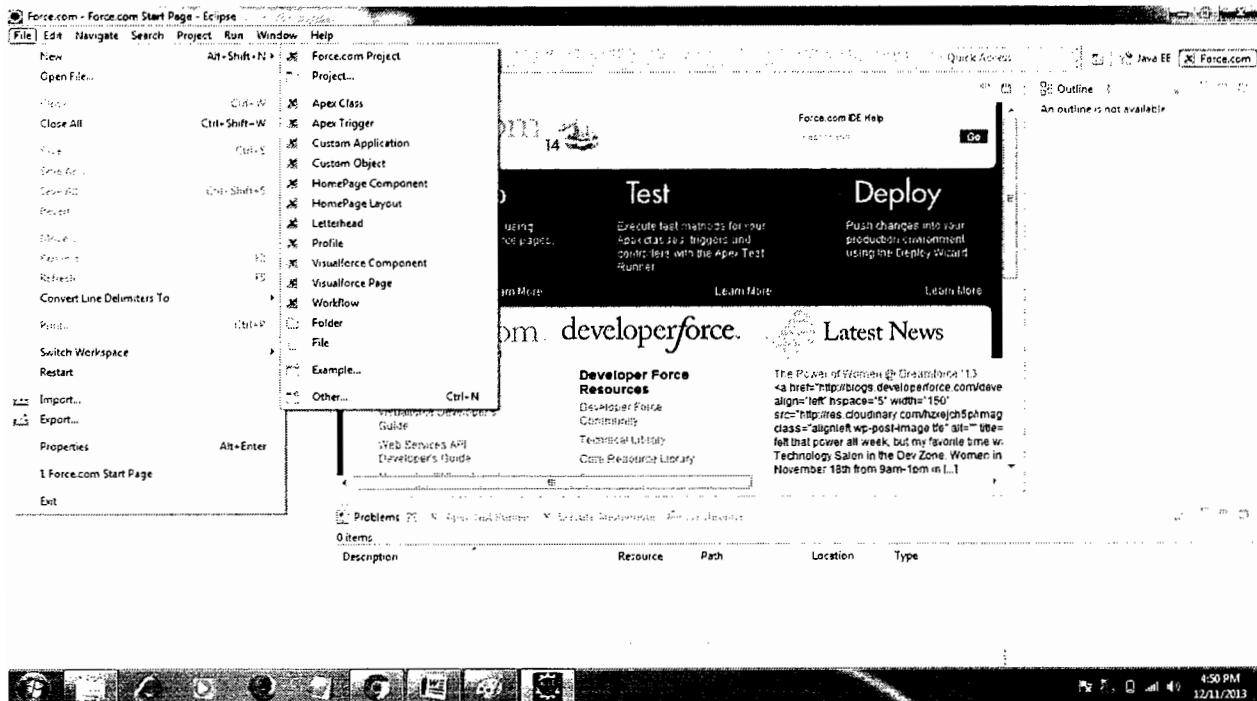


Change the perspective to Force.com



Create a new Force.com Project:

File→New→Force.com Project



## Enter The Project Details Like

1. Project Name :
2. UserName : This is your salesforce username
3. Password : This is salesforce password
4. Security Token : Reset Token from your salesforce account
5. Select Environment : Production / Sandbox / Developer / PreRelease

New Force.com Project

### Create a Force.com Project

Choose a name for your project and configure the connection settings for your organization.

Fields marked with an asterisk (\*) are required.

\*Project name: SatisfProject

Organization Settings

\*Username: batch128@bened.com

\*Password: \*\*\*\*\*

Security Token: \*\*\*\*\*

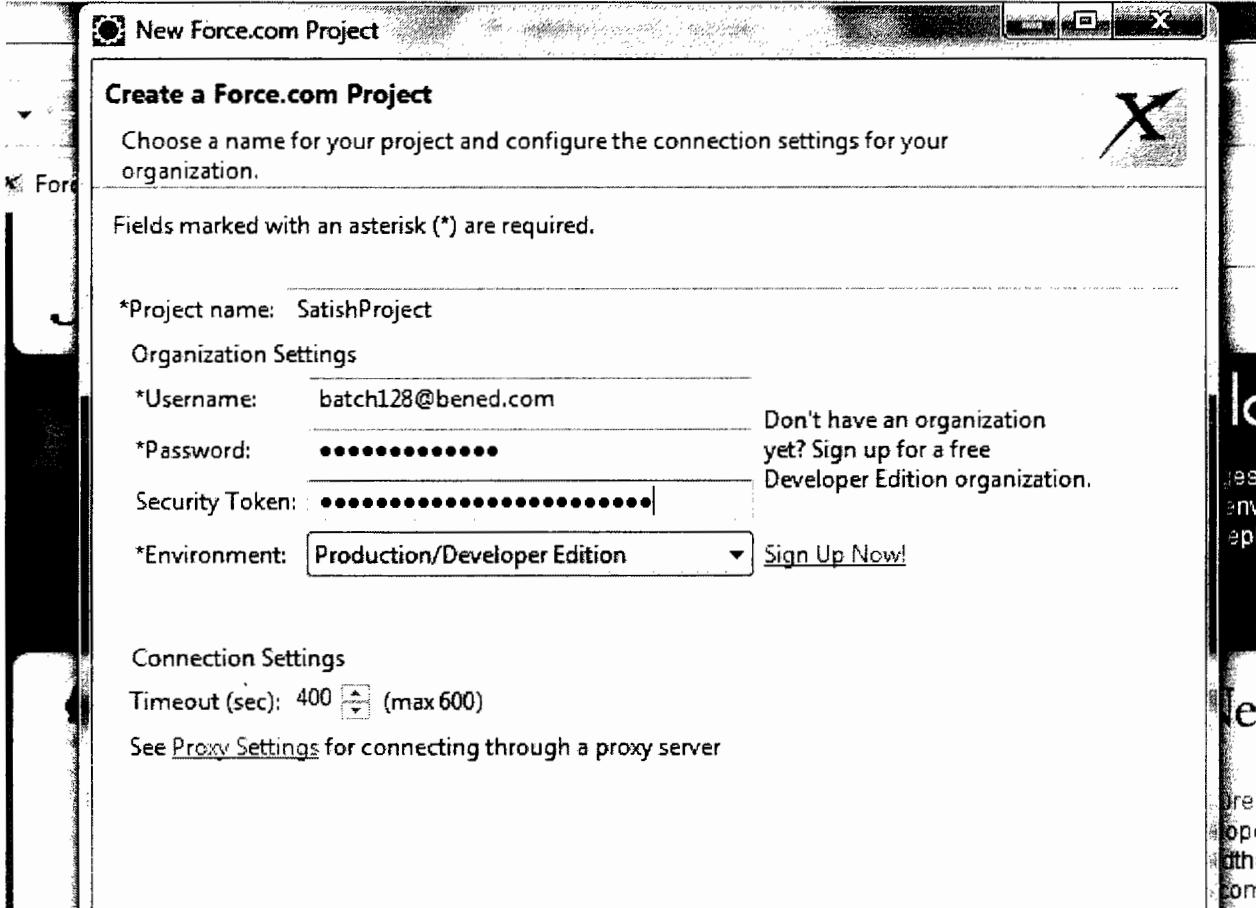
\*Environment: Production/Developer Edition ▾ [Sign Up Now!](#)

Don't have an organization yet? Sign up for a free Developer Edition organization.

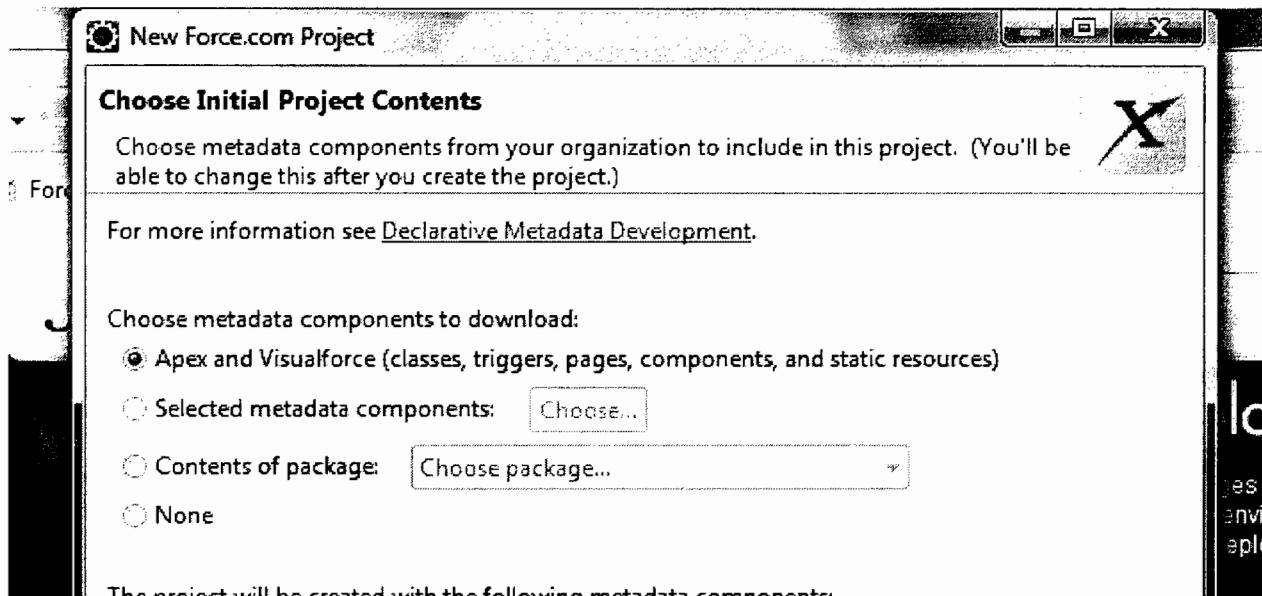
Connection Settings

Timeout (sec): 400 ▾ (max 600)

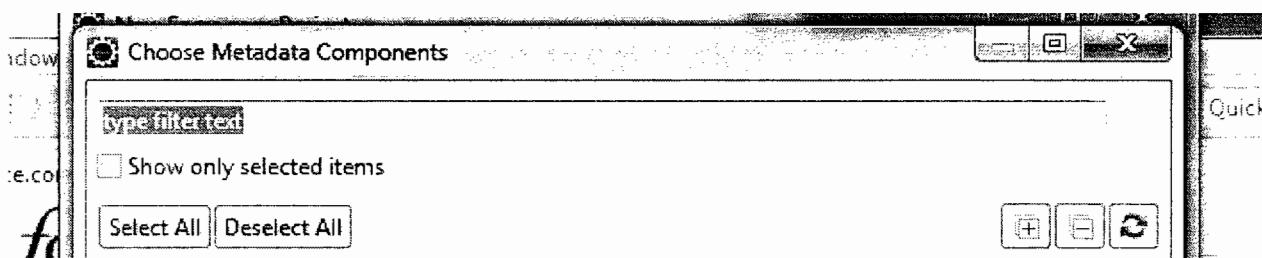
See [Proxy Settings](#) for connecting through a proxy server



Choose which type project contents



Select Metadata components and click choose



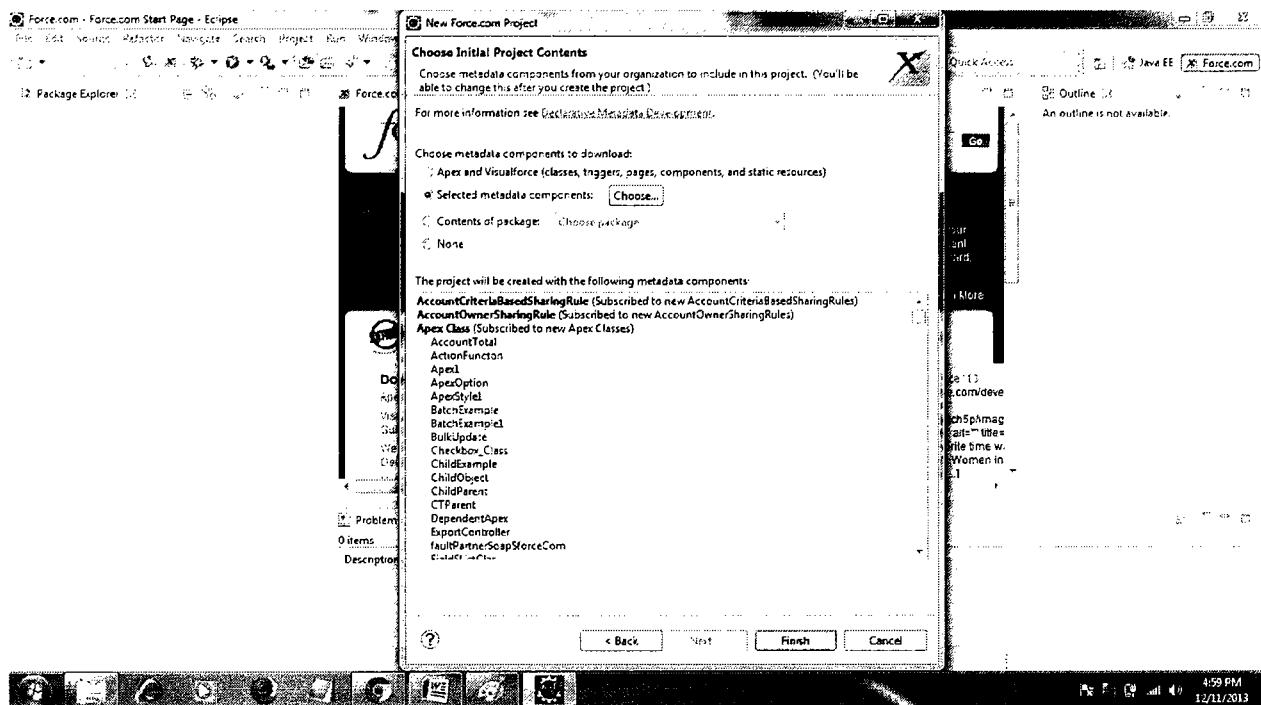
Click Select All button ;

LAST UPDATED: 12/11/13 4:50:50 PM

Component	Subscription
<input type="checkbox"/> accountcriteriabasedsharingrule	
<input type="checkbox"/> accountownersharingrule	
<input checked="" type="checkbox"/> applications	
<input checked="" type="checkbox"/> approvalprocesses	
<input checked="" type="checkbox"/> assignmenrules	
<input checked="" type="checkbox"/> authproviders	
<input checked="" type="checkbox"/> callcenters	
<input checked="" type="checkbox"/> campaigncriteriabasedsharingrule	
<input checked="" type="checkbox"/> campaignownersharingrule	
<input checked="" type="checkbox"/> casecriteriabasedsharingrule	
<input checked="" type="checkbox"/> caseownersharingrule	
<input checked="" type="checkbox"/> classes	
<input checked="" type="checkbox"/> communities	
<input checked="" type="checkbox"/> components	
<input checked="" type="checkbox"/> connectedapps	
<input checked="" type="checkbox"/> contactcriteriabasedsharingrule	
<input checked="" type="checkbox"/> contactownersharingrule	
<input checked="" type="checkbox"/> customapplicationcomponents	
<input checked="" type="checkbox"/> datacategorygroups	
<input checked="" type="checkbox"/> email	

Click finish button.

After



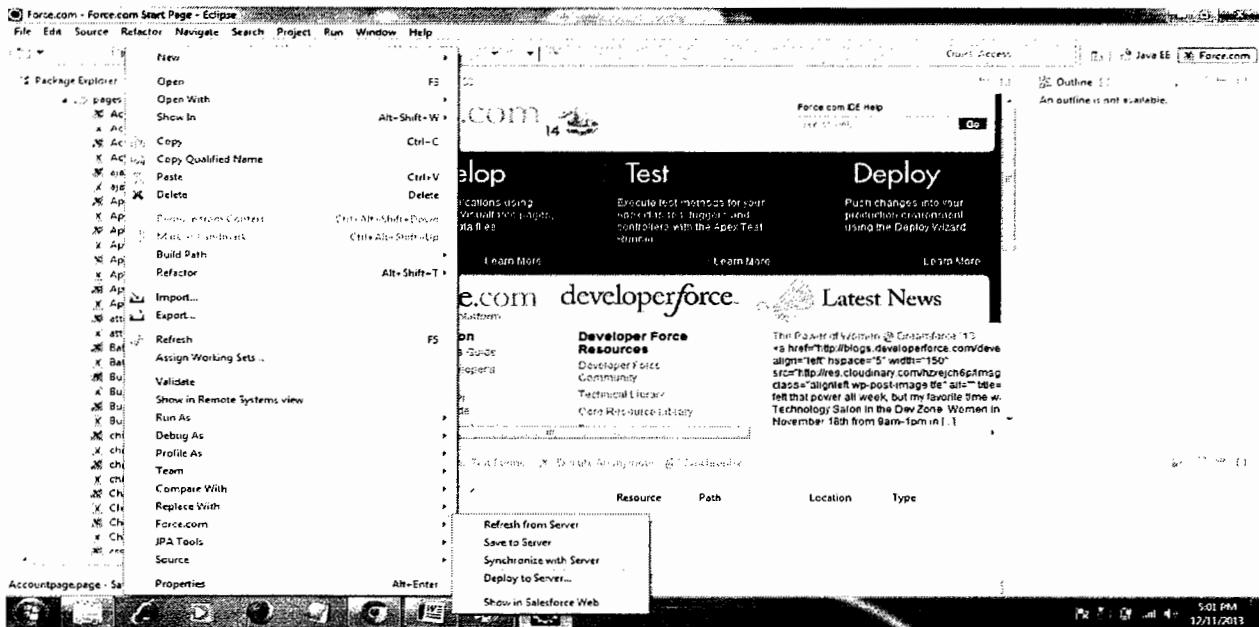
### Steps to deploy from Eclipse to Salesforce Productions: .

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

Force.com Component Deployment

### Destination Details (Step 1 of 4)

This wizard deploys Force.com components to a Salesforce destination.

Fields marked with an asterisk (\*) are required.

**Organization Settings**

\*Username: batch135@sew.com

\*Password: \*\*\*\*\*

Security Token: \*\*\*\*\*

\*Environment: Production/Developer Edition ▾ [Sign Up Now!](#)

Don't have an organization yet? Sign up for a free Developer Edition organization.

Sign Up Now!

Uncheck the Destination Archive checkbox and click next

Force.com Component Deployment

### Archive Options (Step 2 of 4)

This page presents options for creating archives of project and destination Force.com components.

**Archives**

Prior to deployment, create archive (zip) of project and destination Force.com components. (Recommended)

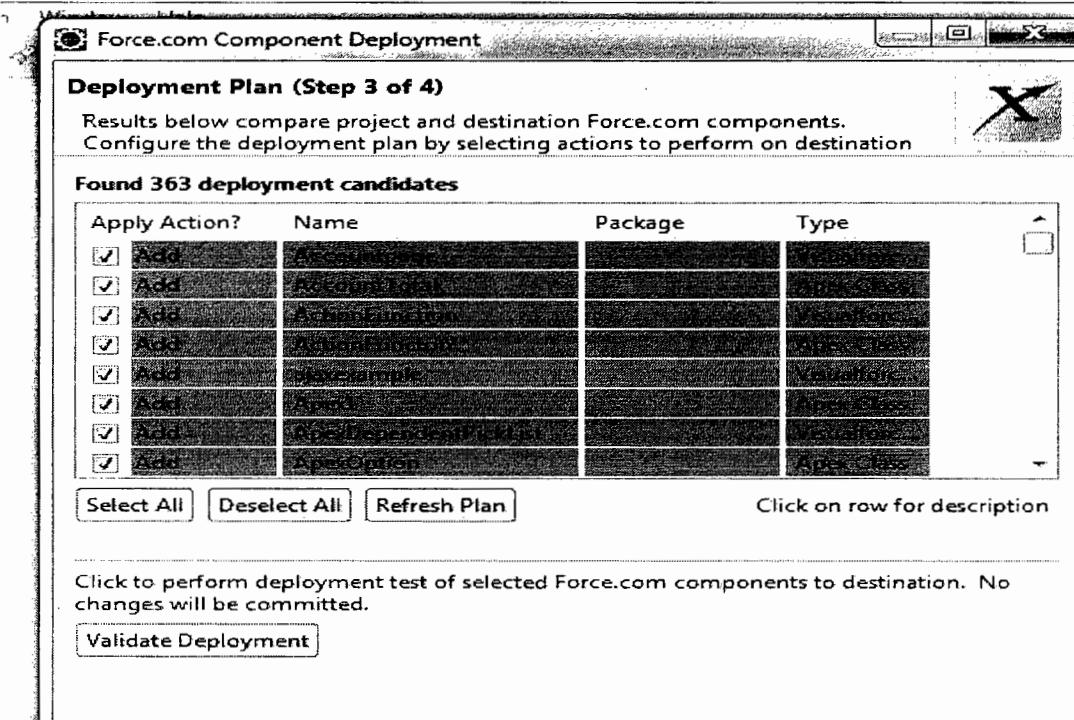
Project archive  
Directory:

Destination archive  
Directory:

Step : Green color indicates that those components are not there in the destination environment.

Yellow Color : This indicates that those components are already available in the destination environment.

Red Color : This indicates that we cannot deploy these components .



The screenshot shows the 'Deployment Plan (Step 3 of 4)' page. At the top, it says 'Results below compare project and destination Force.com components. Configure the deployment plan by selecting actions to perform on destination'. Below this, a table lists 'Found 363 deployment candidates' with columns: 'Apply Action?' (checkboxes), 'Name', 'Package', and 'Type'. A large red 'X' icon is visible in the top right corner of the table area. At the bottom left are buttons for 'Select All', 'Deselect All', and 'Refresh Plan'. To the right of these buttons is the text 'Click on row for description'. At the bottom center is a button labeled 'Validate Deployment'. Below the table, a note says 'Click to perform deployment test of selected Force.com components to destination. No changes will be committed.'

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
<input type="checkbox"/>			
<input type="checkbox"/>			
<input type="checkbox"/> Delete	Task		QuickActi...
<input type="checkbox"/>			

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

**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 Force.com components.

**Found 60 deployment candidates**

Apply Action?	Name	Package	Type
<input checked="" type="checkbox"/>	Add		
<input checked="" type="checkbox"/>	Overwrite Admin.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite ContractManager.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite Custom%3A Marketing Profile.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite Custom%3A Sales Profile.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite Custom%3A Support Profile.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite MarketingProfile.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite ReadOnly.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite SolutionManager.profile	unpackaged	Profile
<input checked="" type="checkbox"/>	Overwrite Standard.profile	unpackaged	Profile

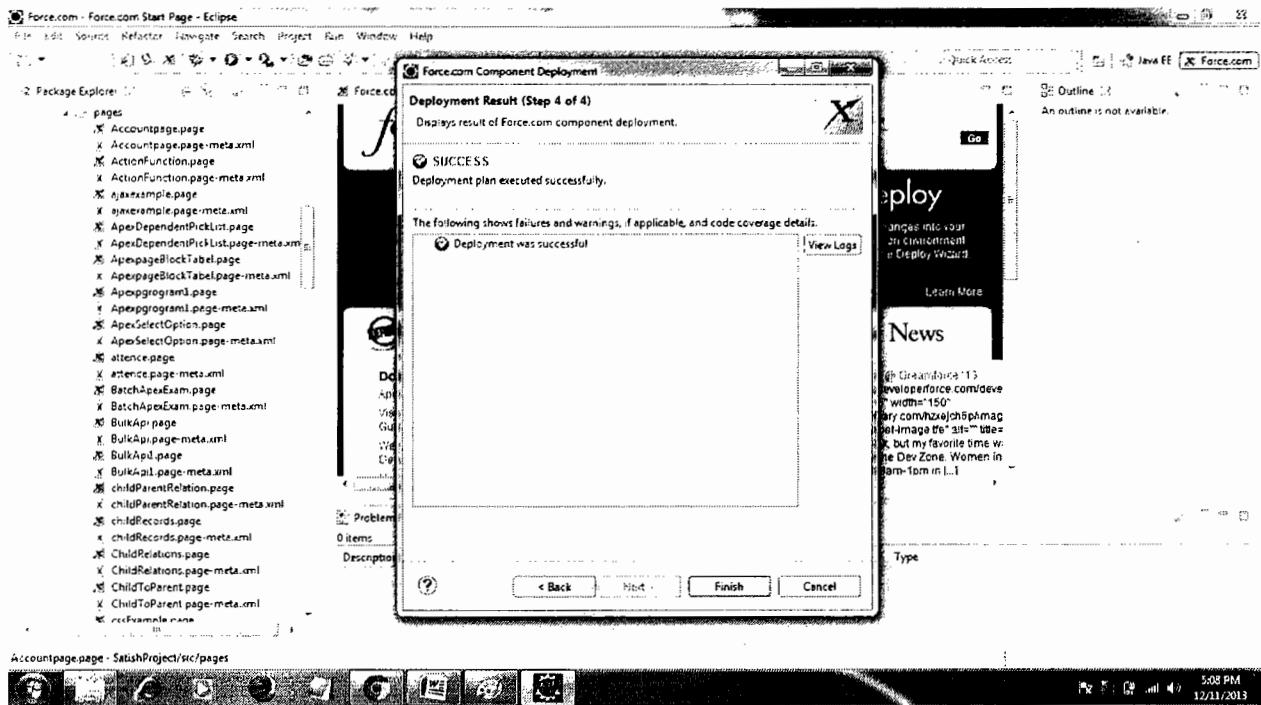
**Select All** **Deselect All** **Refresh Plan** Click on row for description

Click to perform deployment test of selected Force.com components to destination. No changes will be committed.

**Validate Deployment**

Clicking NEXT will execute the deployment plan.

**?** < Back Next > Finish Cancel



## **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.

Manual 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:**

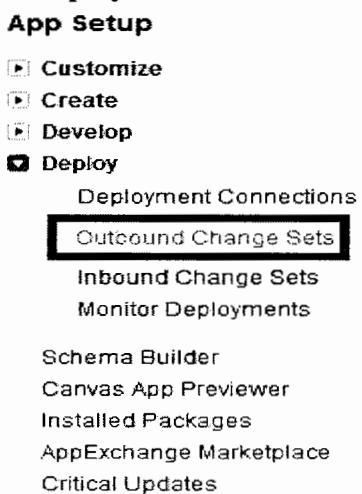
This is first step to the deployment through Change set. Outbound change set is nothing but creation of connection or select the changes you done in Sandbox like Object, Custom Fields, Validation, Workflow, Classes, Trigger etc. For that you have to follow below steps.

- 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.



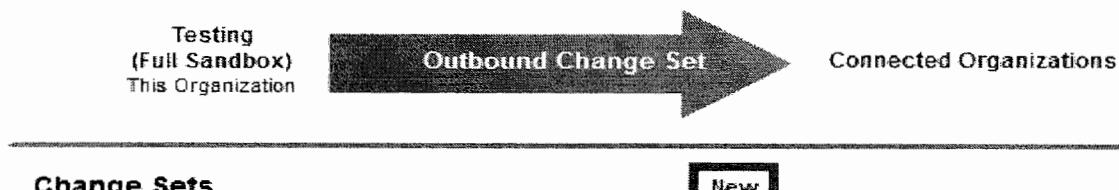
### 2. Click "New" button.

## Outbound Change Sets

An outbound change set contains customizations that you want to send from this organization to another organization or modifications to existing components, such as apps, objects, reports, or Apex classes delete or rename components in another organization.

Example uses:

- Deploy Apex classes and triggers developed in sandbox to production
- Copy custom objects and other customizations to a sandbox without refreshing it
- Migrate changes across environments, e.g. dev sandbox to QA sandbox to production



No change sets have been created in this organization. Click New to create a new change set.

3. Enter a Change Set Name and click "Save" button.

Change Set Edit

## New Change Set

---

**Change Set Edit**

Name

Description

---

---

4 . Click on “Add “ button in the Change set Components section to add the Components

## Sandbox to Production

[Back to List Outbound Change Sets](#)

A change set contains customizations to components such as apps, objects, reports or emails another.

Once a change set has been uploaded, you can't add or remove components. However, you can

### Change Set Detail

[Edit](#) | [Delete](#) | [Upload](#) | [Clone](#)

Change Set Name **Sandbox to Production**

Description

Created By [Maqulan 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.

**Change Set**  
**Sandbox to Production**  
[Back to List: Outbound Change Sets](#)

A change set contains customizations to components such as apps, objects, reports or email templates to another.

Once a change set has been uploaded, you can't add or remove components. However, you can

---

<b>Change Set Detail</b>		<a href="#">Edit</a>	<a href="#">Delete</a>	<b>Upload</b>	<a href="#">Clone</a>
Change Set Name	Sandbox to Production				
Description					
Created By	<a href="#">Magulan Duraipandian, 12/7/2013 7:08 PM</a>				
		<a href="#">Edit</a>	<a href="#">Delete</a>	<b>Upload</b>	<a href="#">Clone</a>

---

<b>Change Set Components</b>		<a href="#">Add</a>	<a href="#">View/Add Dependencies</a>
Action	Name	Parent Object	Type
Remove	<a href="#">Test</a>		Visualforce

7. Select the Target organization and click "Upload" button.

**Upload Change Set**  
**Sandbox to Production**

Choose the organization that will receive the change set. Once the upload completes, you won't be able to edit it or recall it from the history.

 Once you upload this change set, you won't be able to edit it or recall it from the history.

---

<b>Upload Details</b>		<b>Upload</b>	<a href="#">Cancel</a>
<b>Target Organization</b>			
	Name	Description	
	Production	Production organization	
		<b>Upload</b>	<a href="#">Cancel</a>

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.

Expand All | Collapse All

C4 Quick Find

**Force.com Home**

**System Overview**

**Personal Setup**

- My Personal Information
- Email
- Import
- Desktop Integration
- My Chatter Settings
- My Social Accounts and Contacts

**App Setup**

- Customize
- Create
- Develop
- Deploy**

- Deployment Connections
- Outbound Change Sets
- Inbound Change Sets**
- Monitor Deployments

## Inbound Change Sets

An Inbound change set contains customizations sent from another organization's existing components, such as apps, objects, reports, or Apex classes and triggers.

Example uses:

- Deploy Apex classes and triggers developed in sandbox to production
- Apply changes from other environments to this organization, such as a test environment
- When planning to deploy on a schedule, validate pending changes ahead of time

CSC (Global SI and ISV)  
(Production)  
This Organization

Inbound Change Set



### Change Sets Awaiting Deployment

Action	Change Set Name	Description	Source Depic
Delete	Sandbox to Production		Testing

### Deployed Change Sets

There are no deployed change sets.

3. Click "Validate" button.

### Inbound Change Set **Sandbox to Production**

[Back to List: Inbound Change Sets](#)

This change set contains customizations that have been uploaded from a connected organization without committing any changes. It isn't necessary to validate before deploying, as long as the source connection is valid.

#### Change Set Detail

**Validate** | **Deploy** | **Delete**

Change Set Name    Sandbox to Production

Description

#### Source Information

Source Deployment Connection    Testing

**Validate** | **Deploy** | **Delete**

#### Deployment History

This change set hasn't been deployed.

4. Click "Deploy" button.

Inbound Change Set  
**Sandbox to Production**  
[« Back to List: Inbound Change Sets](#)

This change set contains customizations that have been uploaded from a connected organization without committing any changes. It isn't necessary to validate before deploying, as

---

**Change Set Detail**

Change Set Name    **Sandbox to Production**

Description

**Source Information**

Source Deployment Connection    Testing

---

**Deployment History**

Action	Start Time	End Time	Validate Only
View Results	15/7/2013 12:49 PM	15/7/2013 12:49 PM	✓



Project Round:

## Real Time Development Issues:

Schedule Apex Exception – “No Apex Classes Found”?

So I ran across this issue a couple of months ago and forgot to blog about it for my reference and anyone else that runs across this problem. So here's the issue, I have an Apex class scheduled in production that runs another Apex class. I've made some changes to this second class and need to deploy it to production. Before you do that, for Force.com platform requires you to delete the scheduled job for that class since they are related.

No problem. So I deleted the scheduled class, pushed my new code to production (passes all tests! w00t!) and went back to schedule this class to run and saw this error:

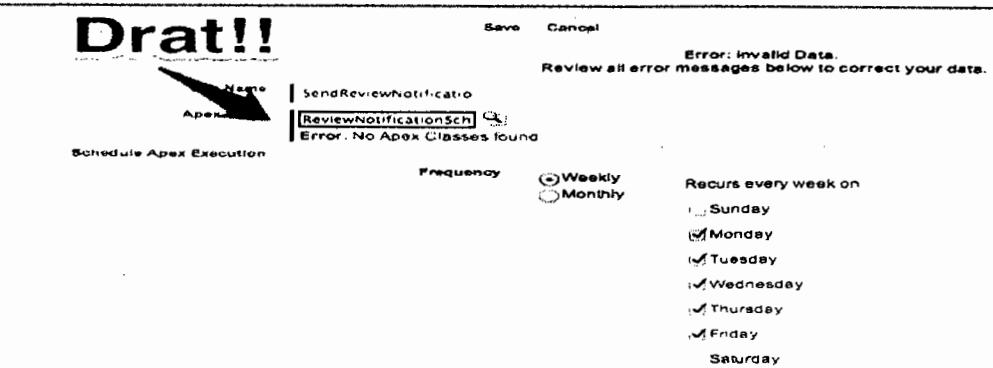
Continued in next page...

Bened Software (P) Ltd, 8-3-219/1, 4<sup>th</sup> Floor, DTDC Opp building, Near Saradhi Studio, Ameerpet, Hyd-38.  
Ph: +91-40-66781355/56 ~~training@benedsoft.com~~, www.benedsoft.com

Frequently asked interview questions with  
real time Scenarios given in the next pages

### Schedule Apex

Schedule an Apex class that implements the 'Schedulable' interface to be automatically executed on a weekly or monthly

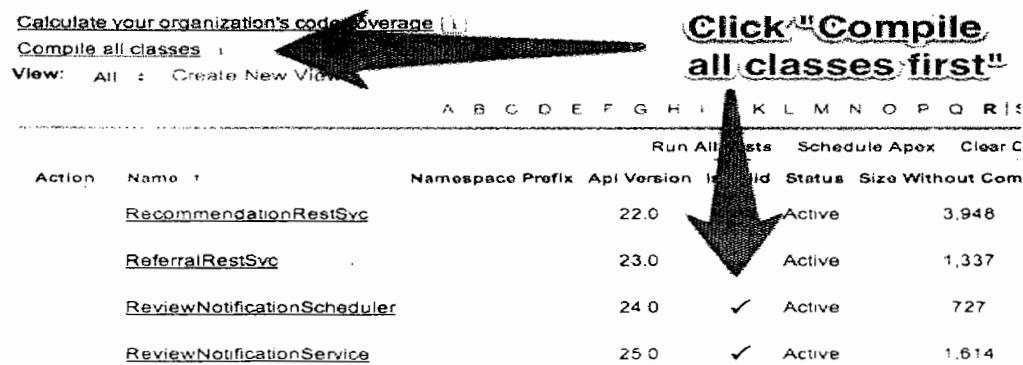


The screenshot shows the 'Schedule Apex Execution' page. A red arrow points to the 'Name' field where 'Apex' is typed. Another red arrow points to the 'SendReviewNotification' class under the 'SendReviewNotificationSch' section. An error message 'Error: No Apex Classes found' is displayed. The 'Frequency' section has 'Weekly' selected. The 'Recur every week on' dropdown is open, showing days from Sunday to Saturday, with 'Tuesday' checked.

I went and looked at the class and noticed that it was no longer valid:

Name	Namespace Prefix	Api Version	Is Valid	Status
<a href="#">RecommendationRestSvc</a>		22.0	✓	Active
<a href="#">ReferralRestSvc</a>		23.0	✓	Active
<a href="#">ReviewNotificationScheduler</a>		24.0		Active
<a href="#">ReviewNotificationService</a>		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.



The screenshot shows the 'Calculate your organization's code coverage' page. A large black arrow points to the 'Compile all classes' button. Another large black arrow points to the 'Run All Tests' button. The table below shows the results of the tests run.

Action	Name	Namespace Prefix	Api Version	Is Valid	Status	Size Without Com
	<a href="#">RecommendationRestSvc</a>		22.0		Active	3,948
	<a href="#">ReferralRestSvc</a>		23.0		Active	1,337
	<a href="#">ReviewNotificationScheduler</a>		24.0	✓	Active	727
	<a href="#">ReviewNotificationService</a>		25.0	✓	Active	1,614

### 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
```

```
{
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 datatypes, 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:

Bened Software (P) Ltd, 8-3-219/1, 4<sup>th</sup> Floor, DTDC Opp bulding, Near Saradhi Studio, Ameerpet, Hyd-38.

Ph: +91-40-66781355/56 [training@benedsoft.com](mailto:training@benedsoft.com), [www.benedsoft.com](http://www.benedsoft.com)

- 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

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, onclick 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

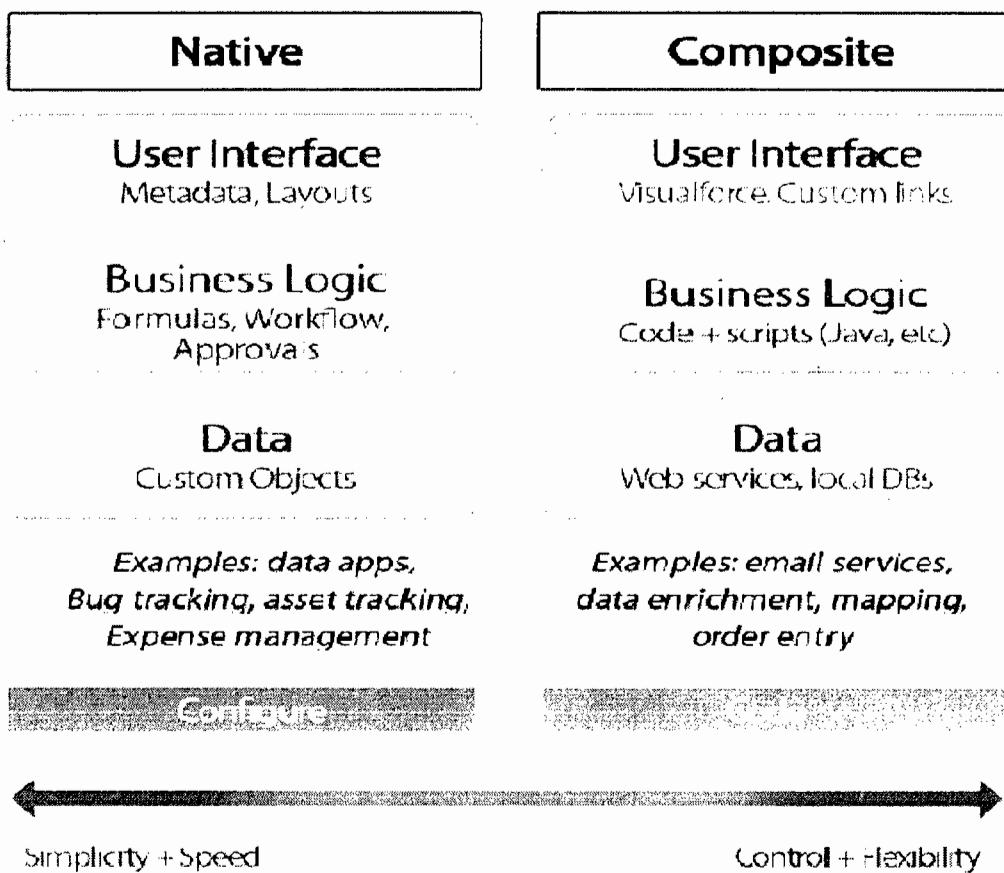
**39. What is AppExchange directory?**

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?**

Bened Software (P) Ltd, 8-3-219/1, 4<sup>th</sup> Floor, DTDC Opp bulding, Near Saradhi Studio, Ameerpet, Hyd-38.

Ph: +91-40-66781355/56 [training@benedsoft.com](mailto:training@benedsoft.com), [www.benedsoft.com](http://www.benedsoft.com)

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 no longer 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.

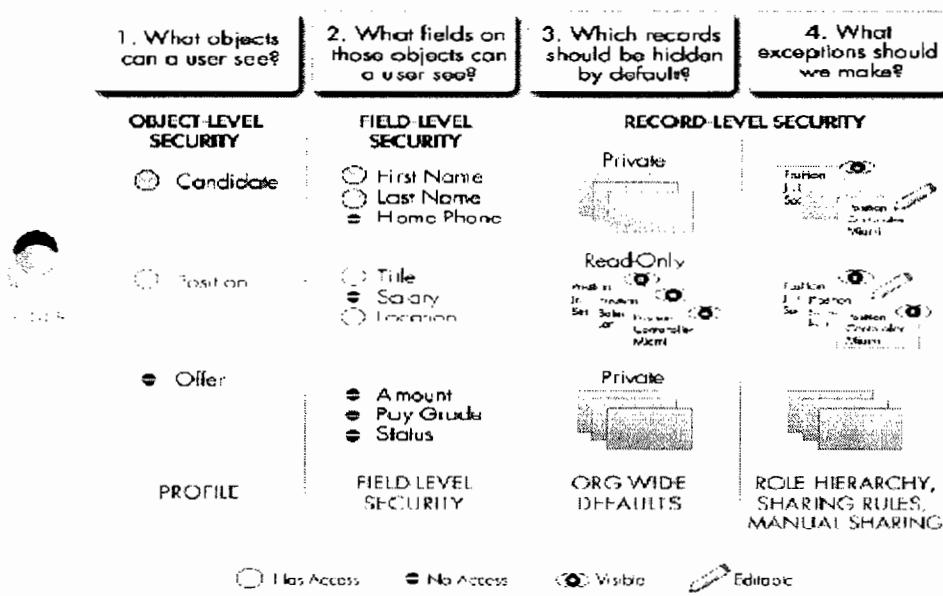
#### 58. What is the difference between Object-Level Security , Field-Level Security and Record-level Security ?

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 extra permissions 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.

## 81. What is a Web tab?

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 Access 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