Introduction to Apex

Bootcamp

Where does Apex apply in salesforce application

* Apex can be used to perform DML operations on sobjects or list of sobjects
* Apex can be used to store and process data From sobjects
* Used to create Sobject triggers in salesforce
* Usually apex code is written as class methods
* Apex is used to create the controllers logic behind custom UI like visualforce and Lightning components ,example custom save or update button
* Apex can be used to invoke integration http callouts to access application from third party endpoints .
* Apex code itself can be encapsulated as a REST or SOAP Api,invocable webservices for external systems
* Can be used to code Database batch jobs
* Apex code can be executed synchronously or asynchronously
* Perform complex validation over multiple objects.
* Create complex business processes that are not supported by workflow.
* Build applications for Salesforce IOT and blockchain.

Comparison with java

→Apex is strongly typed, case-insensitive language

→Apex allows you to create classes, interfaces, properties and collections (arrays)

→Apex has typical conditional and control flow statements

→Apex can be written using Eclipse - Force.com IDE plugin and also using SFDC Developer console

Key differences

→Apex code is compiled and executed on the Cloud virtual machine unlike Java where you initialise a JVM and control the memory allocated to Java and C heap etc.

→Cannot do any explicit garbage collection operations

→Apex is aware of the underlying SFDC schema/data model/objects and no need establish any explicit connections etc. to fetch data or perform DMLs and supports transactions and rollbacks

→Apex does not support multiple inheritance

→Has inbuilt unit testing framework

Code Editor

* Force .com APP
* Developer console
* External IDE – eclipse,vsstudio,aside.io

Data types 1

→All variables that are used to store and process data during run-time are of type primitive, Collection, sObject, Enum

→Primitives

- » Apex is a strongly typed language , which means the variable data type should be declared before something is stored in it
- » Integer, Date, Boolean, Double, Long, String, ID are some of the basic data types

→sObjects

- » sObject variable represents a SFDC standard or custom object
- » It can be a generic sObject or specific sObject type

→Collections

- » Lists, maps and sets are different types of collections data types
- » They can store an array of primitives, sObjects

→Enums

- » Enum is an abstract data type, for ex, four seasons WINTER, SUMMER, SPRING, FALL can be stored in enum variable

Data type 2

→ There are 3 types of collections in Apex – List, Map, Sets

→They used to store the records returned from a SOQL query, serve a group of records indexed by ID ,lookup criteria

→List

» List can be multi-dimensional, you can have a list of integers or a list of Sobjects.

→Map

» Map is a collection of key-value pairs

» Key can be ID fields of a records and value can be another data type

→Sets

» Set is an unordered collection of elements that do not contain any duplicates

» All these collection types can be nested up to 4 levels and go up to 5 levels in total

Data types

https://docs.google.com/document/d/1dcYvlKWHtmPrrseSIjIW4UTFTOBuG2N0QTw2CJivprE/edit?usp=sharing

Labs using developer console and debug log

* Create a log for the current user and enable the log
* Write to this log from developer console anonymous block
* system.debug('test');
* Try out the data types and soqls

* Maps can be looped using keyset() or values()

Sobject variables

→SObject variables are used to store a single record of a particular standard or custom object type in Apex

→This provides a very powerful and easy way to create new records or update existing records

→Standard objects like account, contact can be declared as shown below:

» Account acc = new Account ();

» Account acc = new Account (Name = 'New account') ;

» Account acc = new Account (Name = 'New account', Phone = '1234567899') ;

» Custom objects can be declared as shown below

» Student_master__c SM = new Student_master__c ();

» Generic SObject can be declared and be casted to a specific object type later

» Dot notation cannot be used to access fields in Generic SObject variables

Classess and objects

→Class is the container in which all apex code is stored and also a blue print on which an instance (loaded into memory) is created. Instance of a class – Object

→Classes have attributes (or variables) and methods that change the attributes or get the current value of the attributes. All of these happen on the instance of the class which is the object

→Object has state and also has methods to control the state or query the state

→Access modifiers Public or Global or Private. Top level class always need either Public or Global access modified

→Inner classes need not have the modifier and by default it is Private

Class syntax

```
private | public | global
[virtual | abstract | with sharing | without sharing]
class ClassName [implements InterfaceNameList] [extends ClassName]
{  // You can define the attributes and methods of the class here
}
```

→ Access Modifier like private, public, global defines the scope of the class:

- » Private - Class is accessible only locally
- » Public - Class is accessible at the application level or namespace level
- » Global - class is accessible globally and usually if class contains methods defined as web service the class need to be defined as Global

Method definition

Method definition

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

{ // Method body}

→ public - this method is accessible by any apex code within this application or namespace

→private -this is the default option and means this method is accessible only within the class where it is defined

→global - this keyword is used if you need to expose this method to applications outside using SOAP API as a web service

→ protected this method is visible to only to inner classes and in classes that extend this particular class

→Static variables, methods

→Instance variables , methods

→Refer to the Force.com Apex developers guide (salesforce apex language reference) for more details about this

Example class

```
public class Casesharingapex {
   public s void caseshared(){
   caseshare cs = new caseshare();
   cs.CaseId ='50090000022TNJZ';
   cs.UserOrGroupId  = '00590000001T5WO';
   cs.CaseAccessLevel ='Read';
       cs.RowCause = 'Manual';

      insert cs;

   }

}
```

Labs convert all the anonyous snippets created so far to into class methods
Execute and check in debug log
Shit tab for indent

Soql

→ SOQL stands for Salesforce Object Query Language to fetch data from the database similar to SQL in Oracle or SQL Server.

» [select id, name from Account] ; fetches account records. You cannot use [Select * from account] in SOQL

» SOQL can be embedded in Apex code to retrieve data and store in Collections like List, Set or Map

» You can also use conditions like WHERE clause and sort records using ORDER BY clause

» Bind variables are also supported

» You can also query related records. Here SOQL differs from SQL. There are no JOINs in SQL

» The query below retrieves data from student master and student marks objects using relationship studenmarks__r which is a custom relationship

» "Select s.Id, s.FirstName__c, (Select subject__c, marks__c, subjectpassfail__c From studentmarks__r) From studentmaster__c s"

» You can clearly see that JOIN is not used in SOQL but this query gets you records from two related objects

Full reference -> https://developer.salesforce.com/docs/atlas.en-us.soql_sosl.meta/soql_sosl/sforce_api_calls_soql_select_examples.htm

SOQL contd..

```
SELECT Id, Name, Phone
    FROM Contact
    WHERE Phone != null
    AND Name LIKE '%rose%'
    ORDER BY Name
    LIMIT 50
```

SOQL Master detail

```
SELECT Name,
    (SELECT FirstName, LastName, Phone
        FROM Contacts)
    FROM Account
```

Inlining SOQL

Inlining SOQL in Apex

```
List<String> levels = new List<String>();
levels.add('Intermediate');
levels.add('Advanced');
List<Session__c> sessions =
    [SELECT Name, Level__c FROM Session__c
        WHERE Level__c IN :levels];
```

Remember sharing rules

→ Apex code runs in system context which means the code has access to all objects and fields and records, which means sharing rules are not applied when the code is trying to retrieve records. All records can be access by the Apex methods

→If you need to restrict access to the Apex class/method and implement sharing rules as per the current user you need to use the "with sharing" keyword

→Anonymous block uses the current users privileges, with sharing , without sharing does not make any sense there

→If a class declared as "without sharing" calls a method in a class declared as "with sharing" method will execute with sharing rules implemented

Sample

public with sharing class myclass { } - Sharing rules implemented for current user

public without sharing class myclass { } - sharing rules not implemented

Looping through soql

```
for (Speaker__c s : [select email__c from Speaker__c])
{
    System.debug(s.email__c);
}
```

Miscellaneous SOQL

Group By

SELECT Name, COUNT(Id) FROM Account GROUP BY Name HAVING COUNT(Id) > 1

Aggregate

SELECT CampaignId, AVG(Amount) FROM Opportunity GROUP BY CampaignId

Limit and offset

SELECT Name FROM Merchandise__c WHERE Price__c > 5.0 ORDER BY Name LIMIT 100 OFFSET 10

Locking records from apex

SELECT Id FROM Account LIMIT 2 FOR UPDATE]

Locks get released at end of transaction

Data Structures

* List<Account> acc = new list <Account>([select id from account])
* Map<id,account> acmap= new Map<Id,Account> >([select id from account]);
* Looping through the constructs
* For (Account a:acc)

* For (String ac:acmap.keyset() or ac:acmap.values()
* List of Sobjects can be used to overcome governor limits

Using Sobjects in apex (anonymous blocks)

* Account ac= new account(); Creates new empty record instance of account
* Account[] acarray = new account[]; Array of accounts
* Sobject variables can be used to create records or list of records
* List<Account> aclist = new List<account>();
* Aclist.add(ac);
* Use the debug log to view data system.debug(‘’);
* Labs Create data using structures,
* Loop through records

Sosl

→SOSL stands for Salesforce Object Search Language, can be used for text based search across multiple salesforce objects in one go

→Similar to SOQL , SOSL can be embedded in Apex code and results can be stored in a List

→FIND {Marc} IN ALL FIELDS RETURNING student_master__c(Name,department__c)

→The above SOSL query returns all records that have the value 'MARC' in any field

→Search query can be a single word or a phrase

→You can specify ALL FIELDS or specific field types NAME FIELDS, EMAIL FIELDS , PHONE FIELDS etc

→You can also specify filter criteria like

→FIND {Marc} IN ALL FIELDS RETURNING student_master__c(Name,department__c WHERE department__c= 'MECH')

→ORDER BY and LIMIT can also be used

→FIND {Marc} IN ALL FIELDS RETURNING student_master__c(Name,department__c WHERE department__c= 'MECH' ORDER BY name, LIMIT 10)

Governor limits

Description	Synchronous Limit	Asynchronous Limit
Total number of SOQL queries issued[1] (This limit doesn't apply to custom metadata types. In a single Apex transaction, custom metadata records can have unlimited SOQL queries.)	100	200
Total number of records retrieved by SOQL queries	50,000	
Total number of records retrieved by `Database.getQueryLocator`	10,000	
Total number of SOSL queries issued	20	
Total number of records retrieved by a single SOSL query	2,000	
Total number of DML statements issued[2]	150	
Total number of records processed as a result of DML statements, `Approval.process`, or `Database.emptyRecycleBin`	10,000	

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm

Debugging

→Debug logs! Contains all the details about the execution of a piece of code and all errors and many more details

→Developer console is a very useful tool to analyze the debug logs

→System.debug is a statement that you can use to insert your own custom messages into the logs for easy troubleshooting

→Debug log can be seen from SF UI Developer console or Force.com IDE

→Debug log size is limited to 2mb per log and 50mb at org level

→There are different activities that generate logs

- » Database, Apex code, Callouts
- » Workflow rules, validation rules
- » Visualforce, System.debug calls
- » Apex profiling

Stand alone code snippets

→Anonymous block is very handy to test a small block of code and iron out all problems in that small block

→This piece of code is not stored in the metadata

→Cannot use static keyword

→You can write simple apex code blocks that perform DMLs, SOQLs, SOSLs and most of the apex features

→Anonymous block can be launched from Developer console, Force.com IDE, or API call