

# Apex Basics & Database

# Agenda

- Introducción a APEX
- sObjects
- Manipular registros con DML
- Querys SOQL
- Querys SOSL

Apex Basics & Database

# Características de Apex

# Apex es un lenguaje orientado a objetos

Similar a C# y Java, Apex es un lenguaje de programación orientado a objetos, que soporta Encapsulación, Abstracción, Herencia y Polimorfismo.

Apex es guardado, compilado y ejecutado en la nube.

El manejo de la base de datos es parte de Apex a través de un DML (Data Manipulation Language) y acceso nativo a SOQL (Salesforce Object Query Language).

Es un lenguaje no sensible a mayúsculas y minúsculas.

Soporte de Triggers, transacciones y Rollbacks.

# Ejemplo de clase Apex

Debe llevar el tipo de acceso, seguido por la palabra clave class y el nombre de la clase. Dentro de la clase se declaran los métodos, de nuevo con su respectivo tipo de acceso, tipo de dato que retorna y el nombre del método.

```
1 public class HelloWorld {  
2     public void printMessage() {  
3         String msg = 'Hello World';  
4         System.debug(msg);  
5     }  
6 }
```

# Sintaxis básica Apex

Enter Apex Code

```
1 ▼ if(condicion) {  
2     //se ejecuta si la condicion se cumple  
3 ▼ } else {  
4     //se ejecuta si la condicion no se cumple  
5 }  
6
```

Enter Apex Code

```
1 ▼ do {  
2     //se ejecuta el codigo al menos una vez  
3     //se evalua si el codigo se vuelve a ejecutar  
4 } while (condition);  
5
```

# Sintaxis básica Apex


Enter Apex Code

```
1 //se evalua antes si el codigo se ejecuta
2 while (condicion) {
3     //codigo a ejecutar al cumplir la condicion
4 }
5
```

Enter Apex Code

```
1 for (Integer i = 0; i < 10; i++) {
2     System.debug(i+1);
3 }
4
5 for (variable : list_or_set) {
6     code_block
7 }
```

# Ejecutar Código Apex

1. Clic en el engranaje (  ), clic en “Developer console”
2. En la Developer Console, clic en File | New | Apex Class entrar el nombre de la clase (HelloWorld) y click en Ok.
3. Reemplazar el código por defecto con el siguiente:

```
public class HelloWorld {  
    public void printMessage() {  
        String msg = 'Hello World';  
        System.debug(msg);  
    }  
}
```



4. Clic en **Debug | Open Execute Anonymous Window**

5. Ingresar el siguiente código, activar la casilla **Open Logs** y clic en **Execute**.

```
HelloWorld hw = new HelloWorld();
```

```
hw.printMessage();
```

6. Clic en la pestaña **Log** y doble clic en el log mas reciente de la lista.

7. Seleccionar **Debug Only** para ver únicamente las líneas generadas por los `System.debug()` del código introducido.

Apex Basics & Database

# Tipos de Datos

# Tipos de Datos

Integer	Valor numérico sin punto decimal
Long	Valor numérico de gran tamaño sin punto decimal
Double	Valores grandes con punto decimal
Decimal	Valor preciso con punto decimal
Date	Almacenamiento de fechas
String	Cadenas de texto
Boolean	Falso o Verdadero

# Tipos de Datos

ID	Identificador de un registro (18 caracteres)
sObject	Objetos de Salesforce que puede verse como una representación de una tabla de una base de datos.
Blob	Colección de datos binarios de un objeto.

Apex Basics & Database

# Colecciones de Datos

# List

Colección ordenada de datos primitivos y no primitivos

Ejemplo 1:

```
List<String> myStrings = new List<String> {'String1', 'String2', 'String3'};
```

Ejemplo 2:

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

```
myStrings.add('String1');
```

```
myStrings.add('String2');
```

```
myStrings.add('String3');
```

# Set

Colección desordenada de datos únicos para manejar valores no duplicados

Ejemplo 1:

```
Set<ID> accountIds = new Set<ID>{'004d000000B0aHSAA1', '004d000000B0aHTAA2'};
```

Ejemplo 2:

```
Set<ID> accountIds = new Set<ID>();  
accountIds.add('004d000000B0aHSAA1');  
accountIds.add('004d000000B0aHTAA2');
```

# Map

Colección de elementos compuestos por una clave y un valor asignado a cada clave.

Ejemplo de creación de Map:

```
Map<ID,String> mapIDstring = new Map<ID, String>();
```

```
mapIDstring.put('004d000000BOaHSAA1','Valor 1');
```

```
mapIDstring.put('004d000000BOaHTAA2','Valor 2');
```

Ejemplo de obtención de datos de Map:

```
String salidaMap = mapIDstring.get('004d000000BOaHTAA2');
```



# Apex Basics & Database

sObjects

# ¿Que es un sObject?

Los Salesforce Objects o sObject son representaciones nativas de cada registro en Salesforce.

Apex está estrechamente integrado con la base de datos, lo que permite generar una abstracción de los datos de una tabla y sus registros, y lo mantiene en memoria como un objeto.

# Creación de variables sObject

**Creación con constructor:**

```
Account acct = new Account(Name='Acme', Phone = '(415)555-1212',  
NumberOfEmployees = 100);
```

**Crear el objeto vacío y añadir campos después:**

```
Account acct = new Account();
```

```
acct.Name = 'Acme';
```

```
acct.Phone = '(415)555-1212';
```

```
acct.NumberOfEmployees = 100;
```

# Apex Basics & Database

## DML Data Manipulation Language

# ¿Qué es DML?

Apex es un lenguaje integrado con la base de datos y este hace uso del DML para realizar acciones sobre los registros sin necesidad de una configuración adicional.

El DML tiene las siguientes declaraciones:

- insert
- update
- upsert
- delete
- undelete
- merge

# Ejemplo de insert con DML

Se crea un sObject (acct) y se utiliza la declaración “Insert” para insertar un nuevo registro en la base de datos:

```
Account acct = new Account();
```

```
acct.Name = 'Acme';
```

```
acct.Phone = '(415)555-1212';
```

```
acct.NumberOfEmployees = 100;
```

```
Insert acct;
```

## Luego de insertar un registro:

Cuando se usa Insert, se crea el registro en la base de datos y Salesforce le asigna un ID de 18 caracteres al registro, podemos acceder a él como una propiedad del sObject recién insertado:

```
ID acctID = acct.Id;
```

Eso almacenará el ID y nos permitirá hacer operaciones como Update o Delete refiriéndonos al registro por medio de su ID

# Actualizar registro

Ya tenemos el ID del registro, podemos usarlo para actualizar usando update

```
acct.Phone = '(415)555-5555';
```

```
Update acct;
```

Podemos verificar la actualización del registro colocando lo siguiente antes y después del update:

```
System.debug('Phone = ' + acct.Phone);
```



# Apex Basics & Database

## SOQL

Salesforce Object  
Query Language

# ¿Qué es el SOQL?

Es un lenguaje similar al SQL pero hecho para trabajar con la plataforma Lightning que nos permite la lectura de registros.

Como la base de datos está integrada con APEX, el siguiente código permite ejecutar una query colocando el resultado devuelto en una lista:

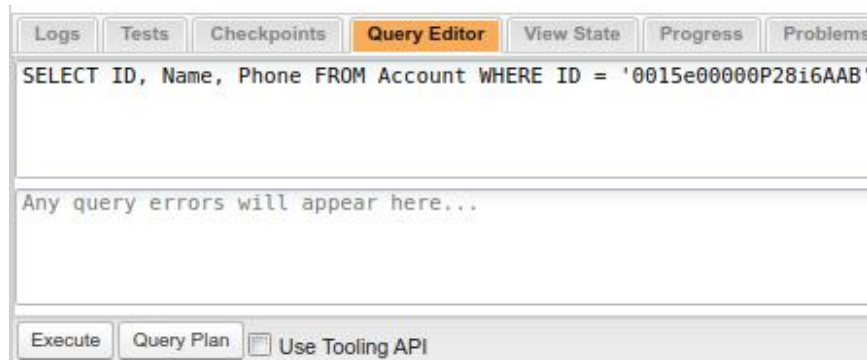
```
List<Account> accts = [SELECT Name, Phone FROM Account];
```

Y la lista accts puede luego ser utilizada con DML para insertar, actualizar o eliminar registros.

# Probar Querys SOQL

Si solo queremos leer registros y probar querys SOQL podemos ir a la **Developer Console**, en la pestaña Query Editor escribir la query y clic en Execute.

El resultado se mostrará en una nueva pestaña:



SELECT ID, Name, Phone FROM Account WHERE ID = '0015e00000P28i6AAB'

Query Results - Total Rows: 1		
Id	Name	Phone
0015e00000P28i6AAB	Cuenta Ejemplo	(415)555-5555

# Ejemplo de SOQL

Se ingresan unos registros pero en algunos registros el nombre de Department está mal escrito y requiere de una corrección.

```
List<Contact> conList = new List<Contact> {  
    new Contact(FirstName='Joe',LastName='Smith',Department='Finance'),  
    new Contact(FirstName='Kathy',LastName='Smith',Department='Technolog'),  
    new Contact(FirstName='Caroline',LastName='Roth',Department='Finance'),  
    new Contact(FirstName='Kim',LastName='Shain',Department='Technolog')};  
  
insert conList;
```

Podemos observar que por error se ingresó 'Technolog' en lugar de 'Technology'

# Corrigiendo con SOQL y DML

Para corregir el error, se puede ejecutar una sentencia SOQL asignando su salida a una lista de tipo contact.

Luego se puede recorrer la lista con un for, la variable interna con nos sirve para actualizar el valor dentro dentro de la lista.

Al final se puede hacer un update a la lista para actualizar todos los registros que se desean corregir.

```
List<Contact> newConList = [SELECT ID, FirstName,
LastName, Department FROM Contact WHERE
Department = 'Technolog'];
```

```
for(Contact con : newConList) {

    if (con.Department == 'Technolog') {

        con.Department = 'Technology';

        System.debug(' ID= ' + con.ID + ' First Name= '
+ con.FirstName + ' Last Name= ' + con.LastName);

    }

}

update newConList;
```

Apex Basics & Database

**SOSL**

Salesforce Object Search Language

# ¿Qué es SOSL?

Es un lenguaje utilizado para buscar en registros en múltiples objetos en Salesforce. Una sentencia SOSL devuelve una lista de listas.

```
List<List<SObject>> searchList = [FIND 'Smith' IN ALL FIELDS RETURNING  
Account(Name), Contact(FirstName, LastName)];
```

En este ejemplo, se devolverán 2 listas, una de tipo Account y otra de Tipo Contact, en ambas listas irán los resultados que contengan "Smith" dentro de todos los campos de ambos objetos. Si no hay resultados para Account o para Contact, se devolverá una lista vacía en ambos casos.

# Ejemplo SOSL

Se buscará en todos los campos, y cuando se encuentre las palabras “Wingo” o bien “SFDC”, se devolverán Account y Contact.

Para acceder a cada lista, puede referirse como espacios de un array [0], [1], ... [n]

Y finalmente se puede recorrer cada lista usando un for para cada uno.

```
List<List<sObject>> searchList = [FIND 'Wingo OR  
SFDC' IN ALL FIELDS RETURNING Account(Name),  
Contact( FirstName, LastName, Department)];  
  
Account[] searchAccounts =  
(Account[])searchList[0];  
  
Contact[] searchContacts =  
(Contact[])searchList[1];  
  
for (Account a : searchAccounts) {  
    System.debug(a.Name); }  
  
for (Contact c : searchContacts) {  
    System.debug(c.LastName + ', ' +  
c.FirstName); }
```