Plataform Developer 1 notes

- **RemoteAction Annotation**
  a. Adding a controller or controller extension grants access to all @RemoteAction methods in that Apex class, even if those methods aren't used in the page.
  b. Anyone who can view the page can execute all @RemoteAction methods and provide fake or malicious data to the controller.

- **Schedule Trigger Flow flow**
  a. Can be scheduled as follow:

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **Login Limits**
    a. The limit is 3,600 calls to login() per user per hour. Exceeding this limit results in a "Login Rate Exceeded" error.
    b. After reaching the hourly limit, Salesforce blocks the user from logging in.
    c. Users can try to log in again an hour after the block occurred.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **Controlling Access Using Hierarchies**
  a. Beyond setting the organization-wide sharing defaults for each object, you can specify whether users have access to the data owned by or shared with their subordinates in the hierarchy.
  b. By default, the Grant Access Using Hierarchies option is enabled for most standard objects, and it can only be changed for custom objects.
  c. The Grant Access Using Hierarchies is enabled for most standard objects, but not all of them

- **Test and Anonymous blocks**
  a. Execute Anonymous always executes in user mode.

- **Loading Test Data**
  a. Using the Test.loadData method, you can populate data in your test methods without having to write many lines of code.
  b. For example, for Account records and a static resource name of myResource, make the following call:
  c. List<sObject> ls = Test.loadData(Account.sObjectType, 'myResource');

- **Access Client Form Factor**
  a. To access the form factor of the hardware the browser is running on, import the @salesforce/client/formFactor scoped module.
  b. import FORM_FACTOR from '@salesforce/client/formFactor';

- **Tooling API**
  a. Tooling API can be used to access ApexLog for debugging.
  b. Tooling API can be used to access code coverage results.

- **Which permission is required to view, retain, and delete debug logs in Salesforce?**

a. You can retain and manage debug logs for specific users, including yourself, and classes and triggers.
b. User permission needed to view, retain, and delete debug logs: View All Data

- **Triggers don't fire**
  a. Trigger on sObject fires whenever we perform the DML operation on the sObject entity.
  b. There are some exceptions when the trigger doesn't fire, and they are listed below:
    - Mass campaign status changes
    - Mass division transfers
    - Mass address updates
    - Mass approval request transfers
    - Mass email actions
    - Modifying custom field data types
    - Renaming or replacing picklist
    - Managing price books
    - Changing a user's default division with the transfer division option checked
    - Changes to the following objects:
      1. BrandTemplate
      2. MassEmailTemplate
      3. Folder
    - Update account triggers don't fire before or after a business account record type is changed to a person account (or a person account record type is changed to business account.)
    - Update triggers don't fire on FeedItem when the LikeCount counter increases.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **Roll-Up Summary Fields with Formulas Fields**
  a. Formula fields cannot be used in the roll-up summary.
  b. If there is a formula field, we cannot use it directly in the roll-up summary field.
  c. Hence, we first need to copy the value from the formula field to a custom currency field to use it in the roll-up summary field.
  d. We can also do it by writing a trigger and performing an aggregate SUM operation.
  e. But as this functionality can be achieved using a declarative option, we will not use programmatic customization.
- **getStandardPricebookId**
  a. This method returns the ID of the standard price book in your organization regardless of whether the test can query organization data. By default, tests can't query organization data unless they're annotated with @isTest(SeeAllData=true).
  b. Creating price book entries with a standard price requires the ID of the standard price book.
  c. Use this method to get the standard price book ID so that you can create price book entries in your tests.

Validation Rules at Mass Transfer Records
  a. Validation rules continue to run on individual records if the owner is changed.
  b. If the Mass Transfer tool is used to change the ownership of multiple records, however, validation rules don't run on those records.

Flows Tests Coverage
  a. A flow version corresponds to a process built in Process Builder or a flow built in Flow Builder. For a process, Apex tests execute only the active version. For a flow, Apex tests execute the active version. When a flow has no active version, Apex tests execute the latest version.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

b. To deploy a process or flow as active, your org must have 75% flow test coverage. To calculate your org's flow test coverage, Salesforce divides the number of covered flows and processes by the sum of the number of active processes and active autolaunched flows.

c. Make sure that Deploy processes and flows as active is enabled in your org's process automation settings. Otherwise, when you deploy active flows and processes via change sets or Metadata API, they're deployed as inactive.

d. Get the names of all flows and processes that have test coverage.
   - SELECT FlowVersion.Definition.DeveloperName
   - FROM FlowTestCoverage
   - GROUP BY FlowVersion.Definition.DeveloperName

e. Get the names of all active autolaunched flows and processes that don't have test coverage.
   - SELECT Definition.DeveloperName
   - FROM Flow
   - WHERE Status = 'Active'
   - AND (ProcessType = 'AutolaunchedFlow' OR ProcessType = 'Workflow' OR ProcessType = 'CustomEvent' OR ProcessType = 'InvocableProcess')
   - AND Id NOT IN (SELECT FlowVersionId FROM FlowTestCoverage)

f. Get overall test coverage for a flow version.
   - SELECT Id, ApexTestClassId, TestMethodName, FlowVersionId, NumElementsCovered, NumElementsNotCovered FROM FlowTestCoverage WHERE flowversionid='301RM0000004GiK'

- **Field History Tracking**
  a. If Process Builder, an Apex trigger, or a Flow causes a change on an object the current user doesn't have permission to edit, that change isn't tracked.
  b. Field history honors the permissions of the current user and doesn't record changes that occur in system context.

- **LightningMessageChannel**
  a. Represents the metadata associated with a Lightning Message Channel.
  b. A Lightning Message Channel represents a secure channel to communicate across UI technologies (Lightning Web Components, Aura Components, and Visualforce).
  c. an example of a LightningMessageChannel component with LightningMessageFields.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningMessageChannel xmlns="http://soap.sforce.com/2006/04/metadata">
   <masterLabel>SampleMessageChannel</masterLabel>
   <isExposed>true</isExposed>
   <description>This is a sample Lightning Message Channel.</description>
   <lightningMessageFields>
     <fieldName>recordId</fieldName>
     <description>This is the record Id that changed</description>
   </lightningMessageFields>
   <lightningMessageFields>
```

```
    <fieldName>recordData</fieldName>
    <description>The current data representing the record that changed</description>
  </lightningMessageFields>
</LightningMessageChannel>
```

- **Log Inspector**
  a. Stack Tree
  b. Execution Stack
  c. Execution Log
  d. Source
  e. Variables
  f. Execution Overview

- **Events in Aura**
  a. In the Aura Components programming model, events are fired from JavaScript controller actions.
  b. Events can contain attributes that can be set before the event is fired and read when the event is handled.
  c. Events are declared by the aura: event tag in a **.evt resource**, and they can have one of two types: c**omponent or application.**
  d. **Component Events**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ A component event is fired from an instance of a component.
- ■ A component event can be handled by the component that fired the event or by a component in the containment hierarchy that receives the event.

   e. **Application Events**
- ■ Application events follow a traditional publish-subscribe model.
- ■ An application event is fired from an instance of a component.
- ■ All components that provide a handler for the event are notified.

● Lightning Message Service

   a. is the first Salesforce technology designed to enable communication between Visualforce and Lightning Components anywhere on a Lightning Page.

● Declarative over programmatic tools

   a. Declarative tools receive automatic upgrades when tools are improved

   b. Declarative development is done though user interface and is easy, can be maintained by System administrator as well.

● maximum number of Roles

   a. Salesforce Support can increase the number of roles allowed in your org.
- ■ By default, a Salesforce org can have up to 500 Roles.
- ■ The current Maximum is 10,000.

   b. A documented business case, including the specific amount of roles required when requesting a higher limit.

● OWD

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

a. **Private**
  - Only the record owner, and users above that role in the hierarchy, can view, edit, and report on those records.

b. **Public Read Only**
  - All users can view and report on records but not edit them. Only the owner, and users above that role in the hierarchy, can edit those records.

c. **Controlled by Parent**
  - A user can perform an action (such as view, edit, or delete) on a contact based on whether he or she can perform that same action on the record associated with it.

d. **Public Read/Write Transfer:**
  - All users can view, edit, transfer, and report on all records. Only available for **cases or leads.**

● Indexed fields in SF

a. Below are the fields which are indexed by default in Salesforce:
  - Name.
  - ID.
  - Owner fields
  - Master-detail relationship fields
  - Lookup fields
  - Audit fields like created date, last modified date, etc.

● lAppExchange

a. Extend Salesforce functionality.

b. Can be distributed to others.

c. Run on the Salesforce platform.

d. Accelerate solution development

e. Bolt solutions are complete industry solutions that are quick to deploy on Salesforce.

f. Trending

 ■ Trends reflect what's popular in the moment.

g. Recomendations

 ■ The recommendations are based on **where you're located**, **what you've installed**, the community profile info you've shared, and more

h. You can filter results by:

 ■ Solution type

 ■ Price

 ■ Salesforce edition

 ■ Customer rating

 ■ Language

i. All AppExchange solutions share a few important qualities.

 ■ **Easy to install:** Thousands of solutions install in just a few clicks.

 ■ **Seamless to set up:** Integrate and configure solutions with clicks not code.

 ■ **Peer reviewed:** There are more than 80,000 peer reviews of AppExchange solutions.

 ■ **Tested for security:** To get listed on AppExchange, every solution passes a rigorous security review.

- **Platform Events**
  a. Use platform events to connect business processes in Salesforce and external apps through the exchange of real-time event data.
  b. Platform events are secure and scalable messages that contain data.
  c. Publishers publish event messages that subscribers receive in reall time.
  d. To customize the data published, define platform event fields.
  e. Subscribers can receive custom event messages from Salesforce or an external system using
    - Apex,
    - CometD clients
    - processes
    - or flows.
  f. By using platform events, publishers can send custom event data through
    - Apex
    - a process
    - a flow
    - or an API.
  g. Multiple subscribers can listen to the same event and carry out different actions.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **Write SOQL Queries**
  ## a. SOQL for loop
    - By using SOQL for loops, you can avoid hitting the heap size limit.
    - SOQL for loops iterate over all of the sObject records returned by a SOQL query. The syntax of a SOQL for loop is either:

```
for (variable : [soql_query])
{

}

for (Account[] tmp : [SELECT Id FROM Account])
{

}
```

- **Iteration components**
  a. Iteration components work on a collection of items instead of a single value. **<apex:pageBlockTable>** is a type of iteration component that can be used to generate a table of data, complete with platform styling.
  b. The **<apex:dataTable>, <apex:dataList>,<apex:repeat>** component can be used if custom styling is required.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

● Web service classes

a. Define any method that uses the **webservice keyword** as **static**.

b. Access to web service methods is only checked at the entry point.

   ■ If access is allowed to the web service, all subsequent code will execute in **system mode.**

c. **Global**

   ■ All classes that contain methods defined with the webservice keyword must be declared as **global**.

   ■ If a method or inner class is declared as **global**, the outer, top-level class must also be defined as **global**.

   ■ Methods defined with the **webservice** keyword are inherently **global**.

   ■ Any Apex code that has access to the class can use these methods.

   ■ You can consider the **webservice keyword** as a type of access modifier that enables **more access than global.**

d. You cannot use the webservice keyword

   ■ in a trigger.

   ■ to define an interface

   ■ to define an interface's methods and variables.

● Cross-object formula fields

a. You can't reference cross-object formulas in roll-up summary fields.

b. In cross-object formulas,

   ■ you can't reference **merge fields f**or objects related to activities.

   ■ you can't reference **fields from contacts through person accounts.**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

c. Salesforce allows a maximum of
   - 10 unique relationships per object in cross-object formulas.
d. Cross-object formula fields can pull
   - data from a record even if the user does not have access to it.
   - field values from master-detail or lookup parent records.
   - field values from objects that are up to 10 relationships away.

● Getting Email: User and Queue

a. if you need owner email and you don't use queues, your formula would be **Owner:User.Email**.
b. If you do use queues, your formula could be
   - **IF(ISBLANK(Owner:User.Id), Owner:Queue.QueueEmail, Owner:User.Email)**

● Multiple Currencies

a. If an organization uses multiple currencies
   - **the currency of the master record determines the currency of the roll-up summary field.**
b. For example
   - **if the master and detail records use different currencies, the detail record value is converted into the currency of the master record.**

● Formulas

a. **CURRENCYRATE**
   - Purpose:
     1. This formula returns the conversion rate used for a record's currency code.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ Syntax:
    1. CURRENCYRATE(/soCode)
        - ○ IsoCode-the currency code in the ISO format:
        - ○ you are required to enter the quotes(e.g."EUR") as the expected value is Text.
        - ○ The return format of the result is Number
- ■ Example:
    1. If you have corporate (Org's default) currency as USD.
    2. But the record's currency is **GBP**. you can now retrieve the conversion rate between those currencies and display it as à formula field or use it in reports.
    3. The returned value will be à number showing the currency conversion rate, eg 0.717412 (you will be able to control how many decimals are shown).
- ■ Note:
    1. Your ORG
        - ○ Should be multi currency enabled to use this function.

b. ADDMONTHS

- ■ Purpose:
    1. Returns the date that is the indicated number of months before or after à specified date.If the resulting month has fewer days than the start month,then the function returns the last day of the resulting month. Otherwise,the result has the same day component as the specified date.
- ■ Syntax:
    1. ADDMONTHS(date, num)

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Date:
  - Specify the date field to be used as the base
- Num:
  - Specify the number of months added to the base date field
- Example:
  1. Calculate the contract renewal date which is 12 months after the Opportunity close date.So the formula for contract renewal date would be-Contract Renewal Date=ADDMONTHS(CloseDate,12)

● MyDomain

a. My domain must be configured and deployed to the org in order to use custom Lightning components.
b. Lightning components do not need to be activated after being deployed to the org.
c. Lightning components are defined as public by default.

● Data Loader vs Data Import Wizard

a. **Data Loader**
  - It can load higher data volumes than the Data Import Wizard.
  - Mappings CAN be saved using Data Loader
  - It is more suitable for loading data multiple times.
  - It supports all standard objects
b. **Data Import Wizard**
  - It's good practice to use the more simple tool whenever you can
  - It supports the upload of up to 50,000 records.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ It provides an option to prevent workflow rules and processes from firing when records are created or updated.
- ■ Mappings cannot be saved using the Data Import Wizard
- ■ Does not support all standard objects, unlike Data Loader.
  1. Supports only Lead, Contacts, Accounts and Solutions
- ■ The data import wizard will allow importing all custom objects.
  c. Triggers
  - ■ They will always run regardless of which tool is used.
  d. The data storage capacity
  - ■ it has no impact on deciding which data import tool is used.

● Roll-up summary fields

a. Roll-up summary fields are used to display a calculated value of related records.

b. The roll-up summary field can be created on any object on
- ■ the Master side of a Master-Detail relationship.

c. Case records are related to Accounts and Contacts via lookup relationships, they cannot be used in roll-up summary fields on those two objects.

d. Special scenarios where roll-up summary fields can be used in a lookup relationship
- ■ Opportunity-Opportunity Product
- ■ Account-Opportunity
- ■ Campaign-Campaign Member.

● System objects

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

a. Activity
b. Event
c. Task
d. User

● View state

a. Salesforce includes a pilot of a tool called the View State Inspector, which lets you view the contents of the view state.
b. This information can help you in optimizing the view state size.
c. This feature needs to be enabled by support - please file a support ticket to get it enabled for your organization.
d. Once enabled, it shows up as a tab in development mode.
e. **Best practice for optimizing the view state:**
   - *Minimize the number of forms on a page*
     1. instead of having multiple forms on a page, have a single form and use <apex:actionRegion> to submit portions of the form.
     2. This will ensure that only a single copy of the view state is associated with that page.
   - *Declare variable as transient*
     1. An instance variable declared as transient is not saved and is not transmitted as part of the view state.
     2. If a certain field is needed only for the duration of the page request and does not need to be part of the view state, declare it as transient.
   - *Recreate state versus storing it in view state*
     1. If you can reconstruct the data during postback, via a SOQL query or a web services call, implement that instead of storing it in controller data members.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- *Refactor your pages to make its view stateless:*
  1. Instead of using **apex:commandLink** or **apex:commandButton** components (which needs to be inside a apex:form component) to invoke an action, use an **apex:outputLink** or other non action method instead and implement the action through an apex:page action attribute - where it makes sense.
- *Use custom objects or custom settings to store large quantities of read-only data*
  1. Let us say that your controller needs to call a web service and parse a large response object.
  2. Storing it in view state may increase the page size.
  3. Marking it as transient would incur the cost of an additional web service call and parsing it again.
  4. In such cases, you could store the parsed response in a custom object and just use the stored record id to get to the parsed response.
  5. **Custom settings provide another mechanism to cache data needed by your controller.**
  6. **Accessing custom settings is faster than access to custom objects since custom settings are part of your application's cache and does not require a database query to retrieve the data.**
- *Consider doing your own state management in certain case:*
  1. In certain cases you may want to bypass the view state mechanism offered by Visualforce and do your own state management.
  2. In such cases, use a **HTML FORM instead of apex:form**.
  3. This technique is useful to deal with Visualforce pages that may have to be served to mobile devices where the view state may be too large for the embedded browsers.
- *Refine your SOQL to only retrieve the data needed by the page*

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

● Visualforce

a. **Overriding actions**
- When **overriding buttons** with a Visualforce page, the standard controller for the object on which the button appears must be used.
- For example, if one wants to use a page to override the View button on Opportunity, the page markup must include the standardController="Opportunity" attribute on the **<apex:page>** tag.
- A controller extension can also be used when one needs to add extra functionality to the Visualforce page that they are using as an override.

b. **DML in some kinds of methods**
- DML statements can't be used **in getter or constructor methods**, but they can be used in setter and static methods.

c. **Visualforce Controllers**
- A Visualforce controller is a set of instructions that specify what happens when a user interacts with the components specified in associated Visualforce markup, such as when a user clicks a button or link.
- *Standard controller*
  1. In order to add a Visualforce page to an object's page layouts, the object's Standard Controller must be used.
  2. Supported actions
     - **Save**
     - **QuickSave**
     - **Edit**
     - **Delete**
     - **Cancel**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- List

3. **In order to use custom Apex logic on a Visualforce page with a Standard Controller, a Controller Extension must be used.**
4. The Controller Extension's constructor must receive the Standard Controller as a parameter, which is of type **ApexPages.StandardController.**
5. It consists of the same functionality and logic that is used for a standard Salesforce page.
6. **Parameter of Constructor**
   - A StandardSetController can be instantiated from either:
     - a list of sObjects or

```
List<account> accountList = [SELECT Name FROM Account LIMIT 20];
ApexPages.StandardSetController ssc = new ApexPages.StandardSetController(accountList);
```

     - a query locator.

```
ApexPages.StandardSetController ssc =
new ApexPages.StandardSetController(Database.getQueryLocator([SELECT Name,CloseDate FROM Opportunity]));
```

   - Using other data types throws an error during compile time.

- *Standard Set Controller*
  1. It allows pages to perform mass updates of records
  2. The getSelected()
     - method returns a list of sObjects representing the selected records.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

3. The getRecord()
   ○ method return the SObject thaxt represent the changes to selected records
4. StandardSetController objects can allow the creation of list controllers similar to, or as extensions of, the pre-built Visualforce list controllers provided by Salesforce.
5. The maximum record limit for StandardSetController is 10,000 records.
   ○ If a query locator is used for instantiating the StandardSetController to return more than 10,000 records, then a LimitException will be thrown.
   ○ However, instantiating StandardSetController with a list of more than 10,000 records will not throw an exception, and instead trims the records down to the limit.
6. **StandardSetController Methods**
   ○ cancel()
      ■ Returns the PageReference of the original page, if known, or the home page.
   ○ first()
      ■ Returns the first page of records.
   ○ getCompleteResult()
      ■ Indicates whether there are more records in the set than the maximum record limit. If this is false, there are more records than you can process using the list controller. The maximum record limit is 10,000 records.
   ○ getFilterId()
      ■ Returns the ID of the filter that is currently in context.
   ○ getHasNext()

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ Indicates whether there are more records after the current page set.
  - ○ getHasPrevious()
    - ■ Indicates whether there are more records before the current page set.
  - ○ getListViewOptions()
    - ■ Returns a list of the listviews available to the current user.
  - ○ getPageNumber()
    - ■ Returns the page number of the current page set. Note that the first page returns 1.
  - ○ getPageSize()
    - ■ Returns the number of records included in each page set.
  - ○ getRecord()
    - ■ Returns the sObject that represents the changes to the selected records.
    - ■ This retrieves the prototype object contained within the class, and is used for performing mass updates.
  - ○ getRecords()
    - ■ Returns the list of sObjects in the current page set.
    - ■ This list is immutable, i.e. you can't call clear() on it.
  - ○ getResultSize()
    - ■ Returns the number of records in the set.
  - ○ getSelected()
    - ■ Returns the list of sObjects that have been selected.
  - ○ last()
    - ■ Returns the last page of records.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- next()
  - Returns the next page of records.
- previous()
  - Returns the previous page of records.
- save()
  - Inserts new records or updates existing records that have been changed. After this operation is finished, it returns a PageReference to the original page, if known, or the home page.
- setFilterID(filterId)
  - Sets the filter ID of the controller.
- setpageNumber(pageNumber)
  - Sets the page number.
- setPageSize(pageSize)
  - Sets the number of records in each page set.
- setSelected(selectedRecords)
  - Set the selected records.

- ***Standard list controller***
  1. It enables you to create Visualforce pages that can display or act on a set of records.
  2. To use a standard list controller, the standard Controller and **recordSetVar** attributes are defined on the Visualforce page component.
     - <apex:page standardController="Opportunity" recordSet Var="opportunities">

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

3. **Benefits**
   - ○ Existing List View filters can be applied
   - ○ Pagination of records
   - ○ A dynamic number of records can be rendered on the page

- ■ ***Custom Controller (REPLACE)***
  1. It is a class written in Apex that implements all of a page's logic, without leveraging a standard controller.
  2. You can define new navigation elements or behaviors, but you must also reimplement any functionality that was already provided in a standard controller.
  3. Like other Apex classes, **custom controllers execute entirely in system mode**, in which the object and field-level permissions of the current user are ignored.
  4. Custom controller executes in system mode, but we can use the "with sharing" keyword and execute it in user mode.
  5. If one needs to apply a user's organization-wide defaults,role hierarchy,and sharing rules,the [with sharing] keywords should be used in the class definition.
  6. You can specify whether a user can execute methods in a custom controller based on the user's profile
  7. **Valid use cases to use:**
     - ○ The functionality of the standard controller of the Visualforce page needs to be **replaced**.
     - ○ A Visualforce page's functionality needs to run in system mode.

- ■ **Controller Extension (OVERRIDE)**
  1. If an extension works in conjunction with a standard controller, the standard controller methods will also be available.
  2. A controller extension is an Apex class that extends the functionality of a standard or custom controller.
  3. In order to use custom Apex logic on a Visualforce page with a Standard Controller, a Controller Extension must be used.

4. The Controller Extension's constructor must receive the Standard Controller as a parameter, which is of type ApexPages.StandardController.
5. It is a class written in Apex that adds to or **overrides behavior in a standard or custom controller.**
6. Extensions **allow you to leverage the functionality of another controller** while adding your own custom logic.
7. If an extension works in conjunction with a standard controller, the standard controller methods will also be available.
8. The extension is associated with the page using the 'extensions' attribute of the <apex:page> component.
9. Multiple controller extensions can be defined for a single page through a comma-separated list.
10. Overrides are defined by whichever methods are defined in the 'leftmost' extension, or, the extension that is first in the comma-separated list.

● Apex

a. Future annotations

- Future methods must be annotated by static
- Future is not allowed on constructors
- Future does not support return of Type String

b. Accessing sharing programmatically

- To access sharing programmatically, you must use the share object associated with the standard or custom object for which you want to share.
- Objects on the detail side of a master-detail relationship do not have an associated sharing object.
- The detail record's access is determined by the master's sharing object and the relationship's sharing setting.
- A share object includes records supporting all three types of sharing:

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

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

- ■ **Default objects**
    1. **AccountShare** is the sharing object for the Account object,
    2. **ContactShare** is the sharing object for the Contact object,
    3. and so on
- ■ **Custom Objects**
    1. all custom object sharing objects are named as follows, where MyCustomObject is the name of the custom object:
        - ○ **MyCustomObject__Share**

c. Miscellaneous

- ■ The **isFormulaTreatNullNumberAsZero**() method of the DescribeFieldResult class returns true if null is treated as zero in a formula field, and false otherwise.

d. Class definition

- ■ Virtual
    1. The virtual definition modifier declares that this class allows extension and overrides.
    2. You **cannot** override a method with the override keyword unless the class has been defined as virtual.

e. Properties

- ■ An Apex property is similar to a variable; however, you can do additional things in your code to a property value before it is accessed or returned.

■ **Automatic Properties**

```
public class AutomaticProperty {
  public integer MyReadOnlyProp { get; }
  public double MyReadWriteProp { get; set; }
  public string MyWriteOnlyProp { set; }
}
```

■ **Notes**
1. We recommend that your get accessor not change the state of the object that it is defined on.
2. Properties cannot be defined on interfaces.

```
Public class BasicClass {

  // Property declaration
  access_modifier return_type property_name {
    get {
      //Get accessor code block
    }
    set {
      //Set accessor code block
    }
```

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

```
    }
```

f. Access Modifiers

- ■ By default, a method or variable is visible only to the Apex code within the defining class.
- ■ Modifiers
    1. **private**
        - ○ This access modifier is the default
        - ○ If you don't specify an access modifier, the method or variable is private
    2. **protected**
        - ○ This means that the method or variable is visible to any **inner classes** in the defining Apex class, and to the classes that **extend** the defining Apex class.
        - ○ You can only use this access modifier for instance methods and member variables.
    3. **public**
        - ○ Default
    4. **global**
        - ○ This means the method or variable can be used by any Apex code that has access to the class, not just the Apex code in the same application.
   - ○ If you declare a method or variable as global, you must also declare the class that contains it as global.

● Aura Components

a. An Aura component has up to <mark>eight code artifacts</mark> plus metadata today (nine total)
   - **CMP**: The only required resource in a bundle. Contains markup for the component or app. Each bundle contains only one component or app resource.
   - **Design**: File required for components used in Lightning App Builder, Lightning pages, Experience Builder, or Flow Builder.
   - **Renderer**: Client-side renderer to override default rendering for a component.
   - **SVG**: Custom icon resource for components used in the Lightning App Builder or Experience Builder.

b.
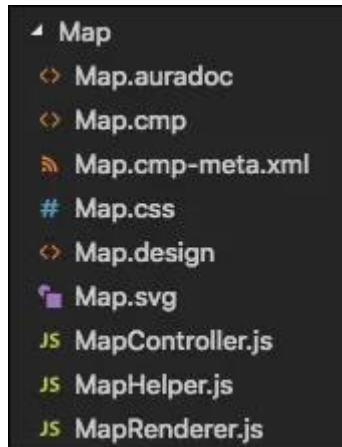
● Lightning Web Component

a. Lightning web components are custom HTML elements built using HTML and modern JavaScript.
b. Lightning web components and Aura components can coexist and interoperate on a page.
c. To admins and end users, they both appear as Lightning components.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

d. Enabling LWC in flows
■ adding lightning__FlowScreen in XML files

● Salesforce Fundamentals

a. Metadata-driven development model is one of the key differences between developing on the platform and developing outside of Salesforce.
b. Salesforce platform encourages you to minimize code
c. The platform's metadata-driven architecture lets you complete most basic development tasks without ever writing a line.
d. Salesforce offers a host of tools for point-and-click—or **declarative**—development.
e. Some development tasks, like writing validation rules or hooking up components with UI elements, are considered low code. That means they require some basic programmatic knowledge to complete, but aren't so rigorous that they're considered programmatic
f. If you're working on a team with non-coders, you can leave the declarative development tasks to them while you double down on more code-intensive projects.
g. There are **three core programmatic technologies** to learn about as a Salesforce developer.
   ■ **Lightning Component Framework:**
      1. A UI development framework similar to AngularJS or React.
      2. The Lightning Component framework is a user interface development framework for desktop and mobile.
   ■ **Apex:**
      1. Salesforce's proprietary programming language with Java-like syntax.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ **Visualforce:**
    1. A markup language that lets you create custom Salesforce pages with code that looks a lot like HTML, and optionally can use a powerful combination of Apex and JavaScript.
    2. **Caveats**
        - ○ Limited interactivity (aside from added JavaScript functionality)
        - ○ Higher latency, which degrades mobile performance
- h. Developer Console
    - ■ Developer Console is the Salesforce integrated development environment (IDE) that you can use to develop, debug, and test code in your org.
- ● **Merge Field Syntax**
    - a. A merge field
        - ■ is a field you can put in an email template, mail merge template, custom link, or formula to incorporate values from a record.
    - b. When you insert a merge field in a custom button or link, the syntax consists of an open curly brace and exclamation point, followed by the object name, a period, the field name, and a closing curly brace.
        - ■ {!Object_Name.Field_Name}
- ● **Shared Records**
    - a. How to know?
        - ■ By querying **CustomObject__Share**
        - ■ All Custom object shared objects are named **MyCustomObject_Share**

- **Constructor**
  - a. **CustomController**
    - To create a constructor for a custom controller, the constructor cannot have a parameter
    - public MyController ()

      {

        account = new Account();

      }
  - b. **ControllerExtension**
    - public MyController (ApexPages.StandardController stdController)

      {

        account = (Account) stdController.getRecord();

      }
- **Database Class**
  - a. **Transaction Control**
    - Sometimes during the processing of records, your business rules require that partial work (already executed DML statements) be "rolled back" so that the processing can continue in another direction.
    - Apex gives you the ability to generate a savepoint, that is, a point in the request that specifies the state of the database at that time.
    - Any DML statement that occurs after the savepoint can be discarded, and the database can be restored to the same condition it was in at the time you generated the savepoint.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■

```
Account a = new Account(Name = 'xxx');
insert a;
System.assertEquals(null, [SELECT AccountNumber FROM Account WHERE Id = :a.Id].
            AccountNumber);

// Create a savepoint while AccountNumber is null
Savepoint sp = Database.setSavepoint();

// Change the account number
a.AccountNumber = '123';
update a;
System.assertEquals('123', [SELECT AccountNumber FROM Account WHERE Id = :a.Id].
             AccountNumber);

// Rollback to the previous null value
Database.rollback(sp);
System.assertEquals(null, [SELECT AccountNumber FROM Account WHERE Id = :a.Id].
            AccountNumber);
```

- ■
- **LimitsClass**

a. **getDMLStatements()**
   - Returns the number of DML statements
     1. such as insert,
     2. update or the database.EmptyRecycleBin method that have been called.
     3. **tells you how many DML statements you've mad**
b. **getLimitDMLStatements()**
   - Returns the total number of DML statements or the database.EmptyRecycleBin methods that can be called.
   - **tells you the most you're allowed to make.**
c. **Other methods**
   - **getAggregateQueries()**
   - **getLimitAggregateQueries()**
   - **getAsyncCalls()**
   - **getLimitAsyncCalls()**
   - **getCallouts()**
   - **getChildRelationshipsDescribes()**
   - **getLimitCallouts()**
   - **getCpuTime()**
   - **getLimitCpuTime()**
   - **getDMLRows()**
   - **getLimitDMLRows()**

- **getDMLStatements()**
- **getLimitDMLStatements()**
- **getEmailInvocations()**
- **getLimitEmailInvocations()**
- **getFindSimilarCalls()**
- **getLimitFindSimilarCalls()**
- **getFutureCalls()**
- **getLimitFutureCalls()**
- **getHeapSize()**
- **getLimitHeapSize()**
- **getMobilePushApexCalls()**
- **getLimitMobilePushApexCalls()**
- **getPublishImmediateDML()**
- **getLimitPublishImmediateDML()**
- **getQueries()**
- **getLimitQueries()**
- **getQueryLocatorRows()**
- **getLimitQueryLocatorRows()**
- **getQueryRows()**
- **getLimitQueryRows()**

- ■ **getQueueableJobs()**
- ■ **getLimitQueueableJobs()**
- ■ **getRunAs()**
- ■ **getLimitRunAs()**
- ■ **getSavepointRollbacks()**
- ■ **getLimitSavepointRollbacks()**
- ■ **getSavepoints()**
- ■ **getLimitSavepoints()**
- ■ **getSoslQueries()**
- ■ **getLimitSoslQueries()**
- **Lightning Data Service**
  a. Lightning Data Service (LDS) to serve as the data layer for Lightning
  b. Lightning Data Service identifies and eliminates requests that involve the same record data, sending a single shared data request that updates all relevant components.
  c. it also provides a way to cache data to work offline in case the user gets disconnected, intelligently syncing the data once the connection is restored.
- **Polymorphic fields**
  a. Polymorphic fields are relationship fields that can point to more than one entity.
  b. A polymorphic relationship is one where the current object can be one of several object types.
  c. In a polymorphic relationship, the referenced object of the relationship can be one of several different types of object.

d. Some fields are relationship fields, which means they can be used to get information about a related object.

e. And some of those relationship fields are polymorphic fields.

f. **To determine what kind a field is, call describeSObjects() on the object and examine the properties for the field.**

   1. *If **relationshipName** is not null, the field is a relationship field.*
   2. If, in addition, **namePointing** is true, then the field is polymorphic.

g. *The following example only returns records where the **What** field of **Event** is referencing an Account or Opportunity.*

h. *If an Event record referenced a Campaign in the What field, it wouldn't be returned as part of this SELECT.*

```
SELECT Id
FROM Event
WHERE What.Type IN ('Account', 'Opportunity')
```

- Common Polymorphic Fields
  - Owner
  - Who
  - What
- Example of a Polymorphic Field.
  - The OwnerId field of the Event object has the following properties:
    - relationshipName = Owner
    - **namePointing = true**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- referenceTo = Calendar, User

- **TYPEOF**
  a. Use TYPEOF in a SELECT statement to control which fields to query for each object type in a polymorphic relationship.
  b. The following SELECT statement returns a different set of fields depending on the object type associated with the What polymorphic relationship field in an Event.

     SELECT
       TYPEOF What
         WHEN Account **THEN** Phone, NumberOfEmployees
         WHEN Opportunity **THEN** Amount, CloseDate
         ELSE Name, Email
       END
     FROM
             Event

  c. At run time, this SELECT statement checks the object type referenced by the What field in an Event.
  d. If the object type is Account, the referenced Account's Phone and NumberOfEmployee fields are returned.
  e. If the object type is Opportunity, the referenced Opportunity's Amount and CloseDate fields are returned.
  f. If the object type is any other type, the Name and Email fields are returned.

- **SECURITY_ENFORCED**
  a. Use the **WITH SECURITY_ENFORCED** clause to enable field- and object-level security permissions checking for SOQL SELECT queries in Apex code, including subqueries and cross-object relationships.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

b. Insert the **WITH SECURITY_ENFORCED** clause:
- After the WHERE clause if one exists, else after the FROM clause.
- Before any ORDER BY, LIMIT, OFFSET, or aggregate function clauses.

```
List<Account> act1 =
[
        SELECT
                Id, (SELECT LastName FROM Contacts)
        FROM
                Account
        WHERE
                Name like 'Acme' WITH SECURITY_ENFORCED
]
```

- **Sharing keywords**

    a. **Inner classes**
       - Both inner classes and outer classes can be declared as with sharing
       - Inner classes do not inherit the sharing setting from their container class.
    b. **executeAnonymous**
       - Apex code that is executed with the executeAnonymous call and Connect in Apex always execute using the sharing rules of the current user.
    c. **No declaration**

- ■ Apex without an explicit sharing declaration is insecure by default
- ■ We strongly recommend that you always specify a sharing declaration for a class.

d. **with sharing**
- ■ "with sharing" keyword ensures that Apex code runs in the current user context. Therefore, it is always advisable to use the "with sharing" keyword when declaring a class to enforce sharing rules of the current user.
- ■ Use this mode ==as the default== unless your use case requires otherwise.
- ■ Use the with sharing keyword when declaring a class to enforce sharing rules of the current user.
- ■ Explicitly setting this keyword ensures that Apex code runs in the current user context.
- ■ An Apex class with inherited sharing runs as with sharing when used as:
    1. An Aura component controller
    2. A Visualforce controller
    3. An Apex REST service
    4. Any other entry point to an Apex transaction

e. **without sharing**
- ■ Use this mode with caution.
- ■ the class must be declared with the without sharing keyword in order ==**to ensure that sharing rules are not enforced**==
- ■ Ensure that you don't inadvertently expose sensitive data that would normally be hidden by the sharing model.
- ■ This sharing mechanism is best used to grant targeted elevation of sharing privileges to the current user.
- ■ **For example**
    1. ==**use without sharing to allow community users to read records to which they wouldn't otherwise have access.**==

f. **Inherited sharing**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Use this mode for service classes **that have to be flexible** and support **use cases with different sharing modes** while also defaulting to the more secure with sharing mode.
- <mark>Using inherited sharing is an advanced technique to determine the sharing mode **at runtime** and design Apex classes that can run in either with sharing or without sharing mode.</mark>
- Using inherited sharing, along with other appropriate security checks, **facilitates in passing AppExchange security review** and ensures that your privileged Apex code isn't used in unexpected or insecure ways.

g. **Without and Inherited Sharing**
- There's a distinct difference between an Apex class that is marked with inherited sharing and one with an omitted sharing declaration.
    1. If the class is used as the entry point to an Apex transaction, an omitted sharing declaration runs as without sharing.
    2. However, inherited sharing ensures that the default is to run as with sharing.
    3. A class declared as inherited sharing runs as without sharing only **when explicitly called from an already established without sharing context.**

- **ApexWebServices**
    a. Rest methods
    - Every method should have the

        @httpVerb

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

```
global static returnedType methodName()
{

}
```

@HttpGet	Read	Reads or retrieves record
@HttpPost	Create	Creates records.
@HttpDelete	Delete	Deletes records.
@HttpPut	Upsert	Typically used to update existing records or create records.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

@HttpPatch	Update	Typically used to update fields in existing records.

- **Kinds of DML Statements**
  a. **insert**
  b. **update**
  c. **upsert**
    - The upsert DML operation creates new records and updates sObject records within a single statement, using a specified field to determine the presence of existing objects, or the ID field if no field is specified.
  d. **delete**
  e. **undelete**
  f. **merge**
    - The merge statement merges up to three records of the same sObject type into one of the records, deleting the others, and re-parenting any related records
- **Apex Governor Limits**
  a. DML limit of 150 statements per **Apex transaction.**
  b. https://developer.salesforce.com/docs/atlas.en-us.190.0.salesforce_app_limits_cheatsheet.meta/salesforce_app_limits_cheatsheet/salesforce_app_limits_platform_apexgov.htm

Description	Synchronous Limit	Asynchronous Limit
Total number of SOQL queries issued[1]	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	

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

Total stack depth for any Apex invocation that recursively fires triggers due to `insert`, `update`, or `delete` statements[3]	16	
Total number of callouts (HTTP requests or Web services calls) in a transaction	10	
Maximum timeout for all callouts (HTTP requests or Web services calls) in a transaction	120 seconds	
Total number of methods with the `future` annotation allowed per Apex invocation	10	
Total number of `sendEmail` methods allowed	10	
Total heap size[4]	6 MB	12 MB
Maximum CPU time on the Salesforce servers[5]	10,000 milliseconds	60,000 milliseconds
Maximum execution time for each Apex transaction	10 minutes	
Maximum number of unique namespaces referenced[6]	10	
Maximum number of push notification method calls allowed per Apex transaction	10	
Maximum number of push notifications that can be sent in each push notification method call	2,000	

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **Workbench**
  a. allows users to describe, query, manipulate, and migrate both data and metadata in Salesforce.com organizations directly in their web browser with a simple and intuitive user interface.
  b. provides many advanced features for testing and troubleshooting the Force.com APIs, such as customizable SOAP headers, debug logs for API traffic, backward compatibility testing with previous API versions, and single sign-on integration within the Salesforce application.
- **Anonymous blocks**
  a. An anonymous block is Apex code that doesn't get stored in the metadata, but that can be compiled and executed.
  b. Unlike classes and triggers, anonymous blocks execute as the **current user** and can fail to compile if the code violates the user's object- and field-level permissions.
  c. Compile and execute anonymous blocks using one of the following:
    - Developer Console
    - Salesforce extensions for Visual Studio Code
    - The executeAnonymous() SOAP API call:

- **Types of sandboxes**

Each type has different features to support the activities it's designed for.

Sandboxes Available Per Edition

SANDBOX TYPE	PROFESSIONAL EDITION	ENTERPRISE EDITION	UNLIMITED EDITION	PERFORMANCE EDITION
Developer Sandbox	10	25	100	100
Developer Pro Sandbox			5	5
Partial Copy Sandbox	Not Available	1	1	1
Full Sandbox	Not Available		1	1

- **Refresh intervals**

Sandbox Feature Quick Reference

SANDBOX TYPE	REFRESH INTERVAL	STORAGE LIMIT	WHAT'S COPIED	SANDBOX TEMPLATES
Developer Sandbox	1 day	Data storage: 200 MB File storage: 200 MB	Metadata only	Not available
Developer Pro Sandbox	1 day	Data storage: 1 GB File storage: 1 GB	Metadata only	Not available
Partial Copy Sandbox	5 days	Data storage: 5 GB File storage: Same as your production org	Metadata and sample data	Required
Full Sandbox	29 days	Same as your production org	Metadata and all data	Available

a. **A Scratch org**
   - ■ You can spin up a new scratch org when you want to:
     1. *Start a new project.*
     2. *Start a new feature branch.*
     3. *Test a new feature.*

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

4. *Start automated testing.*
5. *Perform development tasks directly in an org.*
6. *Start from "scratch" with a fresh new org.*

- Scratch orgs don't have the capacity to contain the entire happy soup of metadata contained in your production org.
- Also, they weren't meant to replace sandboxes either.
- When you begin to analyze what pieces of metadata you want to push to a scratch org, ask yourself if all that source is required for your specific project.
- A Scratch org is a source-driven short-lived org that is used in **development and automation**.
- The scratch org is a source-driven and disposable deployment of Salesforce code and metadata, made for developers and **automation (CI/CD).**
- A scratch org is fully configurable, allowing developers **to emulate different Salesforce edition**s with different features and preferences.
- You have two ways of managing your scratch orgs:
  1. **using CLI commands**
  2. **the Salesforce graphical interface.**
- To work with scratch orgs,
  1. first enable the Developer Hub (Dev Hub) in your production or business org.
  2. **You can then use**
     - **Salesforce Extensions for VS Code or**
     - **Salesforce CLI to create scratch orgs**.

b. **Scratch org VS Sandboxes**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ As much as we think scratch orgs are going to rock your world and increase your productivity, sandboxes are still an important part of the package development lifecycle.
- ■ You'll still use them as targets/destinations for testing the installation of the package version you created. Once installed, you continue to use them for user acceptance testing, as a staging environment, and for continuous delivery testing.
- ■ Align source development use cases to your scratch orgs, and align release and deployment testing to your sandboxes.

c. **Partner Developer Edition**
   - ■ A Partner Developer Edition is a licensed version of the free Developer Edition org and comes with **more data storage, licenses, as well as users.**

d. **Trial Production org**
   - ■ A Trial Production org is a trial version of the production org and comes with sample data and features based on the org edition to allow customers to try out and evaluate the platform before subscribing.

e. **Developer sandbox**
   - ■ The Developer Sandbox is designed for testing and development, has a storage limit of **200 MB**, and can be refreshed on a daily basis

f. **The Developer Pro Sandbox**
   - ■ The Developer Pro Sandbox has a storage capacity of **1 GB** and requires an additional license depending on the org edition.
   - ■ Developer and Developer Pro sandboxes are intended for **development and testing in an isolated environment.**

g. **Partial copy sandbox**
   - ■ The Partial Copy Sandbox is used mainly for **testing and training.**
   - ■ A partial copy sandbox can be refreshed with a sample of production data every **5 days** and can store up to **5GB of object records,** Documents, and Attachments and a maximum of 10,000 records per selected object.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ **A Partial Copy sandbox should be used for quality assurance tasks such as user acceptance testing, integration testing, and training.**
  - h. **Full copy sandbox**
    - ■ A full copy sandbox can only be refreshed every **29 days**.
    - ■ A developer or developer pro sandbox does not have any production data when refreshed.
    - ■ **Only Full sandboxes support performance testing, load testing, and staging.**
    - ■ A Full sandbox is a replica of the production org, including all data, such as object records and attachments, and metadata.
    - ■ *The length of the refresh interval makes it difficult to use Full sandboxes for development.*
  - i. **Creating managed packages**
    - ■ Only **Developer Edition** or **Partner Developer Edition** environments can create managed packages.
- ● **Triggers and Order of Execution**
  - a. **Validations in client Side**
    - ■ Before Salesforce executes these events on the server, the browser runs **JavaScript validation** if the record contains **any dependent picklist fields.**
    - ■ The validation limits each dependent picklist field to its available values.
    - ■ **No other validation occurs on the client side.**
  - b. **Order**
    - ■ Before triggers
    - ■ After triggers
    - ■ Workflow fields updates
  - c. Full order

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- When record get created into Salesforce following Order of execution works:
  1. **System Validation Rules**
     - If the request came from a standard UI edit page
     - If the request comes from other sources, such as an Apex application or a SOAP API call, validates only the foreign keys
  2. **Executes record-triggered flows that are configured to run <mark>before</mark> the record is saved.**
  3. **<mark>Apex Before Triggers</mark>**
  4. **Custom Validation Rules**
  5. **Duplicate Rules**
     - if the duplicate rule identifies the record as a duplicate and uses the block action, the record isn't saved and no further steps, such as after triggers and workflow rules, are taken.
  6. **<mark>Saves the record to the database, but doesn't commit yet.</mark>**
  7. **<mark>Apex After Triggers</mark>**
  8. **Assignment Rules**
  9. **Auto-Response Rules**
  10. **<mark>Workflow Rules (If there are workflow field updates)</mark>**
      - <mark>Updates the record again.</mark>
      - <mark>Runs system validations again</mark>
      - <mark>Executes</mark>
        - <mark>before update triggers and</mark>
        - <mark>after update triggers</mark>

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ <mark>regardless of the record operation (insert or update), one more time (and only one more time)</mark>
11. **Escalation Rules**
12. **Executes the following Salesforce Flow automations**
    - ○ Processes
    - ○ Flows launched by processes
    - ○ Flows launched by workflow rules (flow trigger workflow actions pilot)
13. <mark>**Executes record-triggered flows that are configured to run after the record is saved.**</mark>
14. **Roll-Up Summary Fields**
    - ○ **Parent record goes through save procedure.**
15. **Executes Criteria Based Sharing evaluation.**
16. **Commits all DML operations to the database.**
17. **Executes post-commit logic are executed.**
    - ○ **In no particular order**
        - ■ Sending email
        - ■ Enqueued asynchronous Apex jobs, including queueable jobs and future methods
        - ■ Asynchronous paths in record-triggered flows
- ● **Setter and getter methods**
    - a. **Setter methods in VisualForce**
        - ■ A 'set' method should be used to pass data from a **Visualforce page** to its Apex controller.
    - b. **Getter methods in VisualForce**
        - ■ To pass data from an Apex controller to a Visualforce page, a 'get' method must be used.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- The name of the getter method **without the 'get' prefix** should be used to display results from a getter method.
- Getter methods return values from a controller.
- Every value that is calculated by a controller and displayed on a page must have a corresponding getter method, including any Boolean variables.

- **Can't be deployed via API Metadata**
  a. The following components can't be retrieved or deployed with Metadata API, and changes to them must be made manually in each of your organizations:
     - Account Teams
     - Activity Button Overrides
     - Auto-number on Customizable Standard Fields
     - Calendars
     - Campaign Influences
     - *Case Contact Roles*
     - *Case Feed Layouts*
     - *Case Team Roles*
     - Console Layouts
     - Multiline layout fields for contract line items
     - Currency Exchange Rates
     - Data Category Visibility Settings
     - Divisions
     - File Upload and Download Security Settings

- Mail Merge Templates
- Multiline layout fields for opportunity teams
- Offline Briefcase Configurations
- Omni-Channel Queues and Omni-Channel Skills routing types for the LiveChatButton object
- Opportunity Big Deal Alerts
- Opportunity Update Reminders
- **Organization Wide Email Addresses**
- Partner Management
- The following standard picklists:
    1. IdeaTheme.Categories,
    2. Opportunity.ForecastCategoryName,
    3. Question.Origin.
    4. **(All other standard picklists are supported.)**
- Predefined Case Teams
- Quote Templates
- Salesforce to Salesforce
- Self-Service Portal Font and Colors
- Self-Service Portal Users
- Self-Service Public Solutions
- Self-Service Web-to-Case
- Service report templates

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ Social Business Rules
- ■ SoftPhone Layout
- ■ Solution Categories
- ■ Solution Settings
- ■ Standard fields that aren't customizable, such as autonumber fields or system fields
- ■ Web Links on Person Account Page Layouts
- ■ Web-to-Lead

- ● **Constant Variables**
    a. Declaring a constant variable requires using the **'final' keyword.**
    b. Variables that are declared as static are associated with the class and accessed directly without instantiating the class.

- ● **Address compount Field**
    a. Accuracy
    b. City
    c. Country
    d. CountryCode
    e. Latitude
    f. Longitude
    g. Postalcode
    h. State
    i. StateCode
    j. Street

- **Geolocation Compound Field**
    a. Geolocation fields are accessible in the SOAP and REST APIs as a Location—a structured compound data type—or as individual latitude and longitude elements.
    b. This structured data type contains the following fields.
        - **latitude**
        - **longitude**
    c. Geolocation fields are provided on many standard objects, such as
        - Account,
        - Contact,
        - Quote, and
        - User,
            1. as part of their address field or fields.
    d. Geolocation fields can also be added as custom fields to standard or custom objects.
    e. A geolocation compound field is read-only, although its latitude and longitude subfields are editable
    f. Although geolocation fields appear as a single field in the user interface, custom geolocation fields count as *three* **custom fields** towards your organization's limits:
        - one for **latitude**,
        - one for **longitude**,
        - and one for **internal use**.
    g. How to access?
        - SELECT location__c FROM Warehouse__c

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- ■ SELECT
    1. <mark>location__latitude__s</mark>,
    2. <mark>location__longitude__s</mark>
    3. FROM Warehouse__c

- ● **Cross-site scripting**
    - a. **HTML Encoding**
        - ■ HTML Encoding is the process of replacing characters by their character references and HTML decoding is the reverse.
        - ■ Therefore if developers html encode user data prior to HTML rendering, the data will always render as text and never as markup.
- ● Built in VisualForce encoding functions
    - a. The platform provides the following VisualForce encoding functions:
    - ● **JS**ENCODE
        - ○ performs string encoding within a Javascript String context.
    - ● **JS**INHTM*LENCODE*
        - ○ a convenience method that is equivalent to the composition of HTMLENCODE(JSENCODE(x))
    - ● **HTML**ENCODE
        - ○ encodes all characters with the appropriate HTML character references so as to avoid interpretation of characters as markup.
    - ● **URL**ENCODE
        - ○ performs URI encoding (% style encoding) within a URL component context.
        - ■
- ● **VisualForce**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **Charts**
  - Visualforce charting includes a collection of standard components for rendering charts. Visualforce pages support the use of third-party JavaScript charting libraries including Google Charts.
- **Passing parameters as reference and by value**
  - When working with Visualforce, or Apex methods, in particular,
    - primitive types such as Strings are passed by value to a controller,
    - while non-primitive types such as Lists are passed by reference.
  - This allows the controller to act directly on non-primitive types and change the underlying data.
- **Cases to use**
  - for generating PDFs documents
  - to create components for dashboards and layouts. Yes, you can embed à VisualForce page anywhere in your Salesforce dashboard as à standard component
- **getCompleteResult()**
  - Indicates whether there are more records in the set than the maximum record limit.
  - If this is **false**, there are more records than you can process using the list controller.
  - The maximum record limit is **10,000 records.**
- **Metada API**
  - Metada API versus SOAP and Rest API
  - The main purpose of Metadata API is to move metadata between Salesforce orgs during the development process.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Use Metadata API to deploy, retrieve, create, update, or delete customization information, such as custom object definitions and page layouts.
- Metadata API doesn't work directly with business data. To create, retrieve, update, or delete records such as accounts or leads, use SOAP API or REST API.

- **Deploy**
  - **Deploy via ChangeSets**
    - Change Set can be used in related organizations.
    - Change Set requires a deployment connection.
    - Change sets can contain only modifications you can make through the Setup menu.
      - Therefore, you can't use a change set to upload a list of contact records.
      - Change sets contain information about the org. They don't contain data, such as records.
    - How?
      - create an outbound change set
      - Once you send the change set, the receiving org sees it as an inbound change set.
  - **Deploy processes and Flows As Active**
    - By default, active processes and flows are deployed as inactive.
    - After deployment, you manually reactivate the new versions.
    - **If you use a continuous integration and continuous delivery model to deploy metadata changes, enable the option to deploy processes and flows as active.**
    - This option applies to processes and autolaunched flows that are deployed via change sets and Metadata API.
    - **Tests**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **Before you can deploy a process or autolaunched flow as active, make sure to meet flow test coverage requirements.**
- **At least one Apex test must cover the flow test coverage percentage of the active processes and autolaunched flows.**
- *Flow test coverage requirements don't apply to flows that have screens.*
- **DescribeFieldResult Class and Enforcing Object and Field Permissions**
  - Apex generally runs in system context; that is, the current user's permissions and field-level security aren't taken into account during code execution.
  - **fields.isAccessible()**
    - Returns true if the current user **can see** this field, false otherwise.
  - **fields.isCreateable()**
    - Returns true if the field **can be created** by the current user, false otherwise.
  - **fields.isUpdateable()**
    - Returns true if the field **can be edited** by the current user, or child records in a master-detail relationship field on a custom object can be reparented to different parent records; false otherwise.
  - **sObjectType.sObjectType.ObjectName.isDeletable()**
    - you can call the isDeletable method provided by Schema.DescribeSObjectResult to check if the current user has permission to **delete** a specific sObject.
- **Annotations**
  - **JsonAccess Annotation**
    - The @JsonAccess annotation defined at Apex class level controls whether instances of the class can be serialized or deserialized.
    - If the annotation restricts the JSON serialization and deserialization, a runtime JSONException exception is thrown.

- The serializable and deserializable parameters of the @JsonAccess annotation enforce the contexts in which Apex allows serialization and deserialization.
- You can specify one or both parameters, but you can't specify the annotation with no parameters.
- If an Apex class annotated with JsonAccess is extended, the extended class doesn't inherit this property.

The valid values for the parameters to indicate whether serialization and deserialization are allowed:

- **never**: never allowed
- **sameNamespace**: allowed only for Apex code in the same namespace
- **samePackage**: allowed only for Apex code in the same package (impacts only second-generation packages)
- **always**: always allowed for any Apex code

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

```
1   // SomeSerializableClass is serializable in the same package and deserializable in the wider namespace
2
3   @JsonAccess(serializable='samePackage' deserializable='sameNamespace')
4   public class SomeSerializableClass { }
5
6
7   // AlwaysDeserializable class is always deserializable and serializable only in the same namespace (default value
8
9   @JsonAccess(deserializable='always')
10  public class AlwaysDeserializable { }
```

- **InvocableMethod Annotation**
  - It is an attribute that allows other tools like Process Builder to execute the method.
  - Rules
    - Use the InvocableMethod annotation to identify methods that can be run as invocable actions.
    - Invocable methods **are called with REST API** and used to invoke a single Apex method.
    - Supported Modifiers

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Label
  - The label for the method, which appears as the action name in Flow Builder. The default is the method name, though we recommend that you provide a label.
- Description
  - The description for the method, which appears as the action description in Flow Builder. The default is Null.
- Callout
  - The callout modifier identifies whether the method makes a call to an external system. If the method makes a call to an external system, add callout=true. The default value is false.
- Category
  - The category for the method, which appears as the action category in Flow Builder. If no category is provided (by default), actions appear under Uncategorized.
- configurationEditor
  - The custom property editor that is registered with the method and appears in Flow Builder when an admin configures the action. If you don't specify this modifier, Flow Builder uses the standard property editor.
- Invocable methods have dynamic input and output values and support describe calls.

```
 1  public class AccountQueryAction {
 2    @InvocableMethod(label='Get Account Names' description='Returns the list of account names corresponding to the
 3    public static List<String> getAccountNames(List<ID> ids) {
 4      List<String> accountNames = new List<String>();
 5      List<Account> accounts = [SELECT Name FROM Account WHERE Id in :ids];
 6      for (Account account : accounts) {
 7        accountNames.add(account.Name);
 8      }
 9      return accountNames;
10    }
11  }
```

- **Object Relationships**
  - **Hierarchical**
    - A special lookup relationship available for only the user object.
    - It lets users use a lookup field to associate one user with another that does not directly or indirectly refer to itself. For example, you can create a custom hierarchical relationship field to store each user's direct manager.
  - **External and Indirect**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

RELATIONSHIP	ALLOWED CHILD OBJECTS	ALLOWED PARENT OBJECTS	PARENT FIELD FOR MATCHING RECORDS
Lookup	Standard Custom External	Standard Custom	The 18-character Salesforce record ID
External lookup	Standard Custom External	External	The External ID standard field
Indirect lookup	External	Standard Custom	You select a custom field with the External ID and Unique attributes

- ○
  - ■ Each ORG can have **200 external objects**
  - ■ **External is also a custom objec**t, but the data is stored outside the ORG
  - ■ **Indirect and External are available for external objects**
  - ■ Steps to create an external object
    - ● **Create an external data source**
      - ○ When the external data source is created then external object are created when validated and sync
    - ● Create an external object pointing to the external data source

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Next we have to create an external or indirect lookup
  - **External lookup**
    - The **PARENT** is an external object and **THERE** is a field in the external data that references the salesforce records with ids of 18 characters
    - It links
      - **a child standard**
      - **custom**
      - **or external** object
        - to a **parent external object**.
    - When you create an external lookup relationship field, the standard External ID field on the parent external object is matched against the values of the child's external lookup relationship field.
    - External object field values come from an external data source.
  - **Indirect lookup**
    - It is used when the external data does not include salesforce IDs
    - When you create an <mark>indirect lookup relationship</mark> field
      - <mark>on an external object</mark>, you specify the parent object field and the child object field to match and associate records in the relationship.
    - An <mark>indirect lookup</mark> relationship
      - links <mark>a child external</mark> object to
        - a <mark>parent standard</mark>
          - or

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- a **parent custom object.**
  - Specifically, you select a **custom unique**, **external ID** field on the parent object to match against the child's indirect lookup relationship field, whose values come from an external data source.
- **Behaviors of master-detail relationships:**
  - Deleting a detail record moves it to the Recycle Bin and leaves the master record intact;
  - Deleting a master record also deletes related detail and subdetail records.
  - Undeleting a detail record restores it, and undeleting a master record also undeletes related detail and subdetail records.
  - However, if you delete a detail record and later separately delete its master record, you can't undelete the detail record, as it no longer has a master record to relate to.
  - **The Owner field on the detail and subdetail records isn't available and is automatically set to the owner of the master record.**
  - **Custom objects on the detail side of a master-detail relationship can't have sharing rules, manual sharing, or queues, as these require the Owner field.**
  - Detail and subdetail records inherit security settings and permissions from the master record. You can't set permissions on the detail record independently.
  - The master-detail relationship field (which is the field linking the objects) is required on the page layout of the detail and subdetail records.
  - Each custom object can have up **to two master-detail relationships** and up to 25 total relationships
  - Custom objects on the detail side of a master-detail relationship cannot have queues since they require the 'Owner' field.
    - This field is automatically set to the owner of the master record.
  - **By default, records can't be reparented in master-detail relationships.**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Administrators can, however, allow child records in master-detail relationships on custom objects to be reparented to different parent records by selecting the **Allow reparenting option** in the master-detail relationship definition.
- **Upsert**
  - **How Upsert Chooses to Insert or Update**
    - The upsert statement matches the sObjects with existing records by comparing values of one field. If you don't specify a field when calling this statement, the upsert statement uses the sObject's ID to match the sObject with existing records in Salesforce.
    - Upsert uses the sObject record's primary key (the ID), an idLookup field, or an external ID field to determine whether it should create a record or update an existing one:
      - If the key isn't matched, a new object record is created.
      - If the key is matched once, the existing object record is updated.
      - If the key is matched multiple times, an error is generated and the object record isn't inserted or updated.
  -
- **Apex**
  - **Interview Class**
    - The Flow.Interview class provides advanced controller access to flows and the ability to start a flow.
    - To create an Interview object, you have two options.
      - Create the object directly in your class by using:
        - No namespace:
          - **Flow.Interview.flowName**
        - Namespace:
          - **Flow.Interview.namespace.flowName**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Create the object dynamically by using **createInterview**()

```
1  {
2    Map<String, Object> inputs = new Map<String, Object>();
3    inputs.put('AccountID', myAccount);
4    inputs.put('OpportunityID', myOppty);
5
6    Flow.Interview.Calculate_discounts myFlow =
7      new Flow.Interview.Calculate_discounts(inputs);
8    myFlow.start();
9  }
```

Interview Created Dynamically for a Local Flow

```
1  public void callFlow(String flowName, Map <String, Object> inputs) {
2    Flow.Interview myFlow = Flow.Interview.createInterview(flowName, inputs);
3    myFlow.start();
4  }
```

- **DescribeFieldResult Methods. The following are methods for DescribeFieldResult. All are instance methods.**
  - **getByteLength()**

    For variable-length fields (including binary fields), returns the maximum size of the field, in bytes.
  - **getCalculatedFormula()**

    Returns the formula specified for this field.
  - **getController()**

    Returns the token of the controlling field.
  - getDefaultValue()

    Returns the default value for this field.
  - getDefaultValueFormula()

    Returns the default value specified for this field if a formula is not used.
  - getDigits()

    Returns the maximum number of digits specified for the field. This method is only valid with Integer fields.
  - getInlineHelpText()

    Returns the content of the field-level help.
  - getLabel()

    Returns the text label that is displayed next to the field in the Salesforce user interface. This label can be localized.
  - getLength()

    Returns the maximum size of the field for the DescribeFieldResult object in Unicode characters (not bytes).

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- **getLocalName()**

  Returns the name of the field, similar to the getName method. However, if the field is part of the current namespace, the namespace portion of the name is omitted.

- getName()

  Returns the field name used in Apex.

- getPicklistValues()

  Returns a list of PicklistEntry objects. A runtime error is returned if the field is not a picklist.

- getPrecision()

  For fields of type Double, returns the maximum number of digits that can be stored, including all numbers to the left and to the right of the decimal point (but excluding the decimal point character).

- getReferenceTargetField()

  Returns the name of the custom field on the parent standard or custom object whose values are matched against the values of the child external object's indirect lookup relationship field. The match is done to determine which records are related to each other.

- getReferenceTo()

  Returns a list of Schema.sObjectType objects for the parent objects of this field. If the isNamePointing method returns true, there is more than one entry in the list, otherwise there is only one.

- getRelationshipName()

  Returns the name of the relationship.

- getRelationshipOrder()

  Returns 1 if the field is a child, 0 otherwise.
- getScale()

  For fields of type Double, returns the number of digits to the right of the decimal point. Any extra digits to the right of the decimal point are truncated.
- getSOAPType()

  Returns one of the SoapType enum values, depending on the type of field.
- getSObjectField()

  Returns the token for this field.
- getType()

  Returns one of the DisplayType enum values, depending on the type of field.
- isAccessible()

  Returns true if the current user can see this field, false otherwise.
- isAiPredictionField() (Beta)

  Returns true if the current field is enabled to display Einstein prediction data, false otherwise.
- isAutoNumber()

  Returns true if the field is an Auto Number field, false otherwise.
- isCalculated()

  Returns true if the field is a custom formula field, false otherwise. Note that custom formula fields are always read-only.

- isCascadeDelete()

    Returns true if the child object is deleted when the parent object is deleted, false otherwise.

- isCaseSensitive()

    Returns true if the field is case sensitive, false otherwise.

- isCreateable()

    Returns true if the field can be created by the current user, false otherwise.

- isCustom()

    Returns true if the field is a custom field, false if it is a standard field, such as Name.

- isDefaultedOnCreate()

    Returns true if the field receives a default value when created, false otherwise.

- isDependentPicklist()

    Returns true if the picklist is a dependent picklist, false otherwise.

- isDeprecatedAndHidden()

    Reserved for future use.

- isExternalID()

    Returns true if the field is used as an external ID, false otherwise.

- isFilterable()

    Returns true if the field can be used as part of the filter criteria of a WHERE statement, false otherwise.

- isFormulaTreatNullNumberAsZero()

    Returns true if null is treated as zero in a formula field, false otherwise.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- isGroupable()

  Returns true if the field can be included in the GROUP BY clause of a SOQL query, false otherwise. This method is only available for Apex classes and triggers saved using API version 18.0 and higher.

- isHtmlFormatted()

  Returns true if the field has been formatted for HTML and should be encoded for display in HTML, false otherwise. One example of a field that returns true for this method is a hyperlink custom formula field. Another example is a custom formula field that has an IMAGE text function.

- isIdLookup()

  Returns true if the field can be used to specify a record in an upsert method, false otherwise.

- isNameField()

  Returns true if the field is a name field, false otherwise.

- isNamePointing()

  Returns true if the field can have multiple types of objects as parents. For example, a task can have both the Contact/Lead ID (WhoId) field and the Opportunity/Account ID (WhatId) field return true for this method. because either of those objects can be the parent of a particular task record. This method returns false otherwise.

- isNillable()

  Returns true if the field is nillable, false otherwise. A nillable field can have empty content. A non-nillable field must have a value for the object to be created or saved.

- isPermissionable()

  Returns true if field permissions can be specified for the field, false otherwise.

- isRestrictedDelete()

  Returns true if the parent object can't be deleted because it is referenced by a child object, false otherwise.
- isRestrictedPicklist()

  Returns true if the field is a restricted picklist, false otherwise
- isSearchPrefilterable()

  Returns true if a foreign key can be included in prefiltering when used in a SOSL WHERE clause, false otherwise.
- isSortable()

  Returns true if a query can sort on the field, false otherwise
- isUnique()

  Returns true if the value for the field must be unique, false otherwise
- isUpdateable()

  Returns true if the field can be edited by the current user, or child records in a master-detail relationship field on a custom object can be reparented to different parent records; false otherwise.
- isWriteRequiresMasterRead()

  Returns true if writing to the detail object requires read sharing instead of read/write sharing of the parent.

- **DescribeSObjectResult**
  - **Contains methods for describing SObjects. None of the methods take an argument.**
    - getChildRelationships()
      - Returns a list of child relationships, which are the names of the sObjects that have a foreign key to the sObject being described

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- getName()
  - Returns the name of the object.
- getLocalName()
  - Returns the name of the object, similar to the getName method. However, if the object is part of the current namespace, the namespace portion of the name is omitted.
- getRecordTypeInfos()
  - Returns a list of the record types supported by this object. The current user is not required to have access to a record type to see it in this list
- **getRecordTypeInfosByDeveloperName**()
  - Returns a map that matches developer names to their associated record type.
  - The current user is not required to have access to a record type to see it in this map
- getRecordTypeInfosById()

  Returns a map that matches record IDs to their associated record types. The current user is not required to have access to a record type to see it in this map.
- getRecordTypeInfosByName()

  Returns a map that matches record labels to their associated record type. The current user is not required to have access to a record type to see it in this map
- isAccessible()

  Returns true if the current user can see this object, false otherwise.

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- isCreateable()

  Returns true if the object can be created by the current user, false otherwise.

- isCustom()

  Returns true if the object is a custom object, false if it is a standard object.

- isCustomSetting()

  Returns true if the object is a custom setting, false otherwise.

- isDeletable()

  Returns true if the object can be deleted by the current user, false otherwise.

- isDeprecatedAndHidden()

  Reserved for future use.

- isFeedEnabled()

  Returns true if Chatter feeds are enabled for the object, false otherwise. This method is only available for Apex classes and triggers saved using SalesforceAPI version 19.0 and later.

- isMergeable()

  Returns true if the object can be merged with other objects of its type by the current user, false otherwise. true is returned for leads, contacts, and accounts.

- isMruEnabled()

  Returns true if Most Recently Used (MRU) list functionality is enabled for the object, false otherwise.

- isQueryable()

  Returns true if the object can be queried by the current user, false otherwise

- ■ isSearchable()

  Returns true if the object can be searched by the current user, false otherwise.
- ■ isUndeletable()

  Returns true if the object can be undeleted by the current user, false otherwise.
- ■ isUpdateable()

  Returns true if the object can be updated by the current user, false otherwise.
- ■ getLabelPlural()
  - ● Returns the object's plural label, which may or may not match the object name.
- ■ The getLabel()
  - ● method of the DescribeSObjectResult class returns the object's label, which may or may not match the object name.
- ● **Schema Class**
  - ○ *getGlobalDescribe()*
    - ■ Returns a map of all sObject names (keys) to sObject tokens (values) for the standard and custom objects defined in your organization.
    - ■ Map<String, Schema.SObjectType> gd = Schema.getGlobalDescribe();
  - ○ **describeTabs()**
    - ■ Returns information about the standard and custom apps available to the running user.

```
1  Schema.DescribeTabSetResult[] tabSetDesc = Schema.describeTabs();
```

- ○ **describeSObjects(sObjectTypes)**
  - ■ Describes metadata (field list and object properties) for the specified sObject or array of sObjects.

```
1  Schema.DescribeSObjectResult[] descResult = Schema.describeSObjects(
2                                               new String[]{'Account','Contact'});
```

- ● **Exceptions**
  - ○ **ListException**
    - ■ Any problem with a list, such as attempting to access an index that is out of bounds.
  - ○ **FinalException**
    - ■ You cannot update fields records when you are in AN AFTER TRIGGER
  - ○ **NullPointerException**
    - ■ Any problem with dereferencing a null variable.
  - ○ **SObjectException**
    - ■ Any problem with sObject records, such as attempting to change a field in an update statement that can only be changed during insert.
    - ■ When Apex attempts to access a field that has not been included in the SOQL select statement, an **SObjectException** will be thrown
  - ○ **DmlException**
    - ■ If a DML operation fails, it returns an exception of type DmlException.
    - ■ You can catch exceptions in your code to handle error conditions.
  - ○ **QueryException**

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- - Any problem with SOQL queries, such as assigning a query that returns no records or more than one record to a singleton sObject variable.

● Write SOSL Queries

- ○ SOSL is similar to Apache Lucene.

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

- ● When building efficient SOSL queries, create filters that are selective.
- ● **SOSL queries scan all entities.**
- ● The search engine looks for matches to the search term across
  - ○ **a maximum of 2,000 records.**
- ● You can search **text**, **email**, and phone fields for multiple objects, including custom objects, that you have access to in a single query in the following environments.
  - ○ SOAP API search() calls.
  - ○ REST API Search calls.
  - ○ Apex statements.
  - ○ Visualforce controllers and getter methods.

- Schema Explorer of the Eclipse Toolkit.
- **Performance**
  - **IN:**
    - Limits the types of fields to search, including email, name, or phone.
  - **LIMIT**:
    - Specifies the maximum number of rows to return.
  - **OFFSET:**
    - Displays the search results on multiple pages.
  - **RETURNING:**
    - Limits the objects and fields to return.
  - **WITH DATA CATEGORY:**
    - Specifies the data categories to return.
  - **WITH DivisionFilter:**
    - Specifies the division field to return.
  - **WITH NETWORK:**
    - Specifies the Experience Cloud site ID to return.
  - **WITH PricebookId:**
    - Specifies the price book ID to return.
- **Results**
  - Results are **grouped by objec**t and returned in the **order they were defined** in the SOSL query

Author: Olavo Alexandrino
https://www.linkedin.com/in/oalexandrino/
https://www.oalexandrino.com.br/

- Always return a **List of List of sObjects**
- **FIND**
  - SOSL query begins with the required **FIND** clause.
  - Use the required FIND clause of a SOSL query to specify the word or phrase to search for

Type of Search	Examples
Single term examples	`FIND {MyProspect}`
	`FIND {mylogin@mycompany.com}`
	`FIND {FIND}`
	`FIND {IN}`
	`FIND {RETURNING}`
	`FIND {LIMIT}`
Single phrase	`FIND {John Smith}`
Term OR Term	`FIND {MyProspect OR MyCompany}`
Term AND Term	`FIND {MyProspect AND MyCompany}`
Term AND Phrase	`FIND {MyProspect AND "John Smith"}`
Term OR Phrase	`FIND {MyProspect OR "John Smith"}`
Complex query using AND/OR	`FIND {MyProspect AND "John Smith" OR MyCompany}`
	`FIND {MyProspect AND ("John Smith" OR MyCompany)}`

- **OFFSET**
  - Optional clause used to specify the **starting row** offset into the result set returned by your query.
  - **OFFSET** can be used only when querying a **single object.**
  - **OFFSET** must be the last clause specified in a **query**.
  - If you want to specify an **OFFSET** clause, you must include a *FieldList* **with at least one specified field.**
  - RETURNING Account (Name, Industry OFFSET 25)
- **RETURNING**
  - **RETURNING is an optional** clause that you can add to a SOSL query to specify the information to be returned in the text search result.
  - If you don't specify this clause, the default behavior is to return the IDs of all searchable objects in advanced search up to the maximum limit
  - The maximum limit is specified in the **LIMIT *n* clause or 2,000** (API version 28.0 and later), whichever is smaller.