**Q:** **Can a workflow rule be triggered from a field update event happening in an approval process?**

**A:** You have to select the "Re-evaluate Workflow Rules after Field Change" box in order for the Approval process to trigger the workflow.

An approval process can specify a field update action that re-evaluates workflow rules for the updated object. If, however, the re-evaluated workflow rules include a cross-object field update, those cross-object field updates will be ignored.

**Q:** **A process builder is set to re-evaluate the record, which make it iterate multiple times. Does the process builder iterate indefinitely or stop after a few iterations?**

A: The process will iterate for a stack depth of 5. Which means 5 times the process will iterate and then it will halt. So, the chain will continue for 5 times. If infinite loop will happen then the org will hit governor Limits. And in order to avoid such conditions, Salesforce has kept the stack depth of 5 and post that process will halt.

**Q: We have an approval process on a custom object, when the approver rejects the record, that should be automatically deleted. How to achieve this?**

**A:** Create a custom checkbox on that custom object and then Create a Approval Process, as soon as the approver rejects the record then do a field(checkbox) update. After that create a **Autolauched (No trigger) flow.** Now use this flow to delete records based on that checkbox value.

So, in this way as soon as that record is rejected by approver, field update will happen. Once the field is updated, that field will be picked by Flow and then that record will be deleted automatically.

**Q:** **Why do we prefer custom object over custom settings?**

**A:** No triggers on Custom Settings but can have triggers on custom Objects.

Custom Settings are stored in the Cache Memory whereas Custom Objects are stored in Force.com Database.

Custom Settings will have only limited Data types, but custom Objects have large variety of data types.

Custom Settings cannot have Relationships like MD or Lookup but custom objects can have.

Custom Settings cannot have its own custom tab in app but a custom object can have its custom tab in app.

No formula fields or RollUp summary in Custom Settings. But in custom Objects we have.

**Q: can I call a future method from batch class?**

A: Future method and Batch apex both are Asynchronous apex. So, we cannot call a async method from another async methods. So, we cannot directly call a future method from batch class But there can be a indirect or substitute way.

1st: Call the future method from finish() of batch class. Since, finish() is used for post processing logics. We can call a future method from finish().

2nd: Call a webservice method from Batch class, which in return will call a future method.

**Q: write a test class for batch class?**

**A:** The only catch here is to use **@testSetup** to create test records and  **Test.startTest() and Test.stopTest()**. This is used to make all the async calls between this block of code as sync in test context.

@isTest

Public class batchTest {

@testSetup

Public static void setup(){

/\*Create Records here\*/

}

Private static testMethod void(){

/\*Some logic\*/

Create a user and use System.runAs

Test.startTest()

Call the Batch Apex class

Assert statements

Test.stopTest()

}

}

**Q: OOPs Concepts in Salesforce?**

**A:** Oops concepts that we have in Apex are:

1. Objects
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

**Objects:** It is a real life entity. Like Car, Bike, book, Customer. It is an instance of class. A class can have multiple instances. Objects has methods and member variables or instance variables.

Eg: Account, Contact, Opportunity, Lead, etc.

**Class:** A class is just like a collection of real life entity. Like a showroom of cars, A compant having employees, etc. It is a collection of similar entities.

Eg: public class AccountHandler{

}

**Inheritance:** When the object acquires the behaviour/properties of its parent’s class then it is called Inheritance. Inheritance is possible when a child class extends the Parent class. And the Child class inherits the properties of parent or super class.

**Polymorphism:** Polymorphism is a concept in oops. Polymorphism means Many forms/shapes. Polymorphism can be of 2 types static or dynamic.

Static Polymorphism happens at compile time only. Meaning that the compiler knows at compile time only that which method will be called.

Suppose there are two methods in a class and they are having different signatures then it will be a static polymorphism. This is also called method Overloading.

Eg: public class CalculateArea {

Integer side;

Integer length;

Integer breadth;

Public void **Area**(Integer side) {

Double area = side\*side;

System.debug(area);

}

Public vid **Area**(Integer length, Integer breadth) {

Double area = length \* breadth;

System.debug(area);

}

}

In the above example the method area is Overoaded. And this method has different method signatures.

Dynamic Polymorphism happens at runtime. Meaning that at runtime it will be decided that which method will be called. Dynamic Polymorphism is also called Method Overriding.

Eg: Public class car {

Public void color() {

System.debug(‘Generic Red car’);

}

}

Public class TataCar extends Car {

Public void color() {

System.debug(‘Tata Red car’);

}

}

Now when this color() will be called/invoked, at runtime it will be decided that which class’s method will be invoked.

**Abstraction:** Hiding the data and showing/exposing only the relevant/necessary part is called Abstraction. Hiding internal details and showing functionality is abstraction. We use inheritance for implementing abstraction. Access Specifiers(public, private, protected) are also used to achieve abstraction. Hiding internal complexity and showing functionality is known as abstraction.

**Encapsulation:** Capsulating/binding the data into a single unit is called Encapsulation. We use classes and Interfaces to implement encapsulation. Encapsulation is used to provide a better security to data. Moreover, we can manipulate the data using encapsulation.

Global class BatchClass implements Database.Batchable {

}

**Q: Constructor Overloading in Apex?**

**A:** public class TestClass {

Public TestClass(){

//Default constructor or ParameterLess Constructor

}

Public TestClass(Boolean isFalse){

//Paramaterized Constructor

}

Public TestClass(Boolean isTrue, String Name) {

//Paramaterized Constructor

}

}

**Q: Assert statements in Apex**

**A:** System.assert()

This statement has 2 parameter. 1. Is the condition. 2nd(Optional) is the message.

System.assert() is used to assert that the specified condition is true. If the condition is false then a error is thrown.

System.AssertEquals()

This statement takes 3 parameters 1st The actual result, 2nd the expected result, 3rd is the message(optional).

System.AssertEquals() is used to compare whether the actual result and the expected result are equal or not. If not equal then it will throw an Error.

System.AssertNotEquals()

This statement takes 3 parameters 1st The actual result, 2nd the expected result, 3rd is the message(optional).

System.AssertNotEquals() is used to compare whether the actual result and the expected result are equal or not. If equal then it will throw an Error.

**Q: How can we maintain the chain of batch classes?**

**A:** In order to maintain the chain of batch classes. We first need to understand the 3 methods inside Database.batchable.

1. Start() this method is executed once to send the list of sobjects to execute().
2. Execute() this method is executed multiple times depending upon the batch size, taking the sObjects on chunks.
3. Finish() is executed once and is used to do post processing implementations like sending email alerts etc.

So, the batch apex chaining can happen through the finish(). We can call another batch class from finish() method. Important point to remember here is that we can have 5 jobs running concurrently. So the only 5 jobs can be chained. **Ex:**

Public void finish(Database.BatchableContext bc) {

Id firstBatchId = bc.getJobId();

Id job2Id = Database.executeBatch(new otherBatchJob(), 200);

}

**Q: How can I restrict access of RECORDS from higher hierarchy level?**

**A:**  To tackle this scenario we need to take help of OWD. Since, we need to prevent the record access so set the OWD to private for the object. Post that, uncheck the Grant Access Using Hierarchies checkbox. By unchecking this checkbox, the higher roles in the hierarchy won’t be able to access the records. **Important point here is that this checkbox is Enabled for Custom Objects only. For standard objects we cannot uncheck this checkbox. It is by default checked and we cannot edit it**.

**Q: How can I stop the execution of trigger, workflow and validation rules while data uploader?**

**A:** Custom settings are available from both Workflows and Triggers. If you create a "Hierarchy" custom setting object with a checkbox you can reference that checkbox to enable/disable the validation or code. The hierarchy design would allow you to set system wide defaults and then override individual users as needed. It’s been pretty effective in a number of occasions.

**Q: Different types of Workflow Actions?**

**A:** There are 5 types of Workflow actions available:

1. New Task: Create a new Task when record meets criteria
2. New Email Alert: Create a New Email Alert
3. Field Update: Performs a specified field update
4. New Outbound Message: Sends a new Outbound message
5. Select Existing Action: Performs an existing action

**Q: Different criteria of workflow?**

**A:** 2 types

1. Evaluation Criteria
2. Created
3. Created and every time edited
4. Created and any time it's edited to subsequently meet criteria
5. Rule Criteria
6. When criteria are met (field and values are present here)
7. When formula field is set to true (we can write our own formula here)



**Q: Write Validation Rule formula such that LeadSource\_\_c should be either Web or Email or Walk-In.**

**A:** NOT( OR(

ISPICKVAL(LeadSource\_\_c, ‘Web’),

ISPICKVAL(LeadSource\_\_c, ‘Email’),

ISPICKVAL(LeadSource\_\_c, ‘Walk-In’)

) )

**Q: Write a Validation Rule such that Ownership field in Account should not be blank and should be other than ‘Public’ or ‘Private’.**

**A:** OR(

ISPICKVAL(Ownership, ‘Public’),

ISPICKVAL(Ownership, ‘Private’),

ISPICKVAL(Ownership, ‘’)

)

**Q: What is a WSDL? Difference between Enterprise WSDL and Partner WSDL?**

**A:** WSDL(Web Service Description Language) file is provided by Salesforce. It is a XML document which contains a standardized description on how to communicate using a web service.

WSDL is used by developers to aid in the creation of Salesforce Integration.

Enterprise WSDL

1. It is strongly typed.
2. This file is bound/tied to a specific configuration of Salesforce. Meaning every org has its own Enterprise WSDL.
3. The Enterprise WSDL changes if modifications are made in the org’s Configuration.

Partner WSDL

1. It is loosely typed.
2. Partner WSDL can be used to reflect against any Salesforce’s org configuration.
3. It is static. Meaning it does not change if the modifications are made to an salesforce’s org configuration.

**Q: Can we perform DML operation in Constructor?**

**A:** DML operation is not allowed in the constructor of Apex class. A constructor is mainly used to initialization of variables. Salesforce has blocked this due to security issues. Whenever we go for a DML operation in the constructor it restricts us in Salesforce. Whenever we trying to perform any DML operations at the time of initial page load we will be ending up with an error called System.LimitException: DML currently not allowed.

First, using DML in constructor will slow down initialization of your object. Second, it may fail and object just doesn't create at all.

**Q: How restrict record access using record types?**

**A:** Create different record Types as per your requirement.

Now to control the access. First of all, make the object OWD to private.

After that, create Sharing rule and use the criteria based rule on Record type values and then finally add public groups, users, roles to whom you need to provide the Access.

Reference: <https://www.youtube.com/watch?v=n0sYCKhZ-e0>

**Q: Suppose I have 2 users A and B; both have same profile. Now I want A to have access to a field and B should not have access to that field. How to achieve this?**

**A:** The best way to tackle this scenario is to create a permission set which has the access to that specific field. And assign this permission set to User A so that it will have access to that field and User B won’t have access to that field.

**Q: Order of Execution in Salesforce**

**A:** When we save a record using Insert, update or upsert statement. Salesforce performs the following events in order.

1. Loads the original record from the database or initializes the record for an upsert statement.
2. Performs all the system validations like Mobile or email format etc.
3. Executes all the record triggered flows that are configured before the record is saved.
4. All the before triggers are executed.
5. Runs most system validations again, runs all Custom Validation rules.
6. Executes duplicate rules. If the duplicate rules identify the record as duplicate and uses the block action. Record is not saved and no further steps such as Workflow rule after triggers are taken.
7. Executes all after triggers.
8. Executes all assignment rules.
9. Executes auto response rules.
10. Executes workflow rules. If there is field update then
11. Updates the record again.
12. Runs system validations again
13. Custom Validation rules, duplicate rules, processes and escalations are not run again.
14. Runs all before and after triggers again regardless of the dml operation one more time and only one more time.
15. Executes escalation rules.
16. Executes the following automation processes but not in a guaranteed order: -
17. Processes (Process builder)
18. Flows launched by Processes
19. Flows launched by a workflow rule
20. Executes entitlement rules.
21. Executes record triggered flows that are configured to run after the record is saved.
22. Executes Roll Up summaries.
23. Executes criteria-based sharing evaluation.
24. Commits all DML operations to Database.
25. After changes are committed to database, executes post commit logic like sending email alerts, enqueued Async Apex Jobs, Future, Queueable methods.