Q.1 What is Asynchronous Apex?

While working on the Salesforce application, we store huge data in our org and it will keep growing year by year. If we have to run some job that will take some time then in the case of synchronous apex we will get the limit error, heap size error, or timeout error. To avoid such issues, we can do those long-running or time-consuming operations using Asynchronous apex which will run in a separate thread and will not impact the main thread. So Asynchronous Apex is used to run the process in a separate thread at a later time.

Q. 2 Where we can use Asynchronous Apex?

Asynchronous Apex can be used in long-running or time-consuming operations like

Sending Email to the user.

Creating complex reports using Apex code

Calling an external system for multiple records

Schedule a job for a specific time

Chaining apex code execution which might use API call

Sharing Re-calculation

Q3. What are the benefits of Asynchronous processing?

Asynchronous process runs when the system is available for operation. It is not blocking users from doing any work in the application. This will give the below benefit to application.

Increase User Efficiency

More Heap Size and Higher Governer limit

Better Scalability

Q.4 What are the types of Asynchronous Apex?

Currently, Salesforce supports four types of Asynchronous Apex.

Batch Apex

Future methods

Queueable Apex

Scheduled Apex

Q.5 Within what timeframe will an asynchronous request be processed after being enqueued?

Asynchronous processing has a lower priority than real-time interaction via the browser and API. So it will always run in the background when the resource is available to process asynchronous jobs. We can not determine when the job will run, it will be determined by the server. It can be immediate also.

Q. 6 Can you make a web service callout from a trigger?

We cannot call external web services synchronously from triggers, because calling a web service synchronously from triggers will hold up the database transaction until the callout is completed so we should use future methods to call external web services.

Q. 7 Can a future method call another non-future method to process tasks like callouts, and have those methods return data to the future method for further processing?

No, we can not call future methods from the future method.

Q. 8 What are advantages of Batch Apex?

Every batch transaction starts with a new set of governor limits.

The system itself divides the number of batches for records

If one batch fails, the other batches will continue to be executed, and successful batches will still be committed to the database. Successful batches are not rolled back when one batch fails.

Q. 9 Why use Batch Apex in Salesforce instead of the normal Apex?

There are various reasons why Batch Apex is better than normal Apex.

SOQL queries: Normal Apex uses 100 records per cycle to execute SOQL queries. Whereas, Batch Apex does the same in 200 records per cycle.

Retrieval of SOQL queries: Normal Apex can retrieve 50,000 SOQL queries but, in Batch Apex, 50,000,000 SOQL queries can be retrieved.

Heap size: Normal Apex has a heap size of 6 MB; whereas, Batch Apex has a heap size of 12 MB.

Errors: When executing bulk records, Normal Apex classes are more vulnerable to encountering errors as compared to Batch Apex.

Q. 10 What are some best practices when implementing batch apex?

Use normal Apex instead of Batch Apex when a small number of records need to run.

Use extreme care to invoke a batch job from a trigger. The trigger should not add more batch jobs than the limit

Methods declared as future aren’t allowed in classes that implement the Database.Batchable interface.

Methods declared as future can’t be called from a batch Apex class.

All methods in the class must be defined as global or public.

Minimize web service callout times.

Tune queries used in batch Apex code.

Minimize the number of asynchronous requests created to minimize the chance of delays.

Q. 11 How many Schedulable Apex jobs can you have at one time? In other words, what is the maximum number of Apex classes that can be scheduled concurrently?

100 Schedulable Apex jobs can be scheduled.

Q. 12 Are synchronous web service callouts supported by Scheduled Apex?

No. However, if Scheduled Apex calls a Batch Apex job which then makes a web service callout, the callout will be supported.

Q. 13 What are best practices for future methods?

every future method invocation adds one request to the asynchronous queue, avoid design patterns that add large numbers of future requests over a short period of time.

Ensure that future methods execute as fast as possible.

If using Web service callouts, try to bundle all callouts together from the same future method, rather than using a separate future method for each callout.

Consider using Batch Apex instead of future methods to process a large number of records asynchronously.

Q. 14 Why sObject parameters not supported in future methods?

The sObject might change between the time we call the method and the time it executes that’s why sObjects can’t be passed as arguments to future methods. In this case, the future method will get the old sObject values and might overwrite them.

Q. 15 Can I chain a job that has implemented allowsCallouts from a Job that doesn’t have?

Yes, callouts are also allowed in chained queueable jobs.

Q. 16 Can I call Queueable from a batch?

Yes, But it is limited to just one System.enqueueJob call per execute in the Database.Batchable class. Salesforce has imposed this limitation to prevent explosive execution.

Q.17 In which scenario, we can’t call a future method from a batch Job?

Calling a future method is not allowed in the Execute method, But a web service can be called. A web service can also call an @future method. So, we can define a web service having a future method invocation and call the web service from the execute method of Batch Job.

Q.18 How Future method helps in avoiding Mixed DML errors?

There are 2 kinds of sObjects in salesforce.

Non-Setup: Account, opportunity, etc

Setup: User, groups,Queue etc

If we are performing DML on both kinds of sObject in a single transaction, the system doesn’t allow it and throws an exception called Mixed DML exception, stating that a transaction can not have a Mixture of DML operation(Setup and Non-Setup).

To resolve this error, we can put DML operations of a particular kind in the Future scope. Since both the DML operations are isolated from each other, the transaction doesn’t fail.

Q.19 Let’s say, we have run an apex batch to process 2000 records, and It is running with batch size 200. Now, while doing DML on 298th record, an error occurred, What will happen in that case?

In batches, If the first transaction succeeds but the second fails, the database updates made in the first transaction are not rolled back.

Since the batch size is 200, so the first batch will be processed completely and all data will be committed to DB. In seconds batch, if we are committing records using normal DML statements like insert, update then the whole batch will be rollbacked. So records 201 to 400 will not be processed.

If we use the Database DML operations like Database.insert with AllOrNone as false, then partial commit can happen and only 298th record will not be processed in that batch, and a total of 97 records will be processed. Also, the other batch execution will not stop.

Q.20 What is Database.QueryLocator & Iterable<sObject>?

In Database.QueryLocator, we use a simple query (SELECT) to generate the scope of objects. The governor limit for the total number of records retrieved by SOQL queries is bypassed, i.e. it can return up to 50 million records.

In Iterable<sObject>, we can create a custom scope for processing, which would be not possible to create using SOQL where clauses. the governor limit for the total number of records retrieved by SOQL queries is still enforced.

…………………………………………………………………………………………………

1.Can we call the batch in to another batch apex?

Yes, we can call from finish method.

2.Can we call batch apex in to another batch in excute method?

Only in batch class finish method, We can call another batch class. If you will call another batch class from batch class execute and start method, then Salesforce will throw below runtime error.

System.AsyncException: Database.executeBatch cannot be called from a batch start, batch execute, or future method.

3.Can we call the batch apex from triggers in salesforce?

Yes, it is possible. We can call a batch apex from trigger but we should always keep in mind that we should not call batch apex from trigger each time as this will exceeds the governor limit this is because of the reason that we can only have 5 apex jobs queued or executing at a time.

4.Can we call webservice callout from batch apex?

To make a Webservice callout in batch Apex, we have to implement Database.AllowsCallouts interface.

5.How many times start,execute,finish methods will excute in batch apex?

Start method,finish method one time, execute method it depends on requirement. Based on the batch size and data retrieved in Start method.

6.What is the Batch executions limit per day?

The maximum number of batch executions is 250,000 per 24 hours.

7.Can we call the future method in batch class?

No,we can’t call.

8. Is Future method support primitive data types? Why sobject parameters not supported?

The reason why sObjects can’t be passed as arguments to future methods is that the sObject might change between the time you call the method and the time it executes. In this case, the future method will get the old sObject values and might overwrite them.

9.How can I perform Callouts from Future methods?

We need to add a parameter callout=true in @future.

10.Can I write a future call in Trigger?

Yes.

11.What are the limitations of future methods?

It is not a good option to process large numbers of records.

Only primitive data types supported.

Tracing a future job is also typical.

Can’t call future from batch and future contexts, 1 call from queueable context is allowed.

12.What is future method?

Use of future methods to isolate DML operations on different sObject types to prevent the mixed DML error. Each future method is queued and executes when system resources become available. That way, the execution of your code doesn’t have to wait for the completion of a long-running operation. A benefit of using future methods is that some governor limits are higher, such as SOQL query limits and heap size limits

NOTE :-

1) Methods with the future annotation must be static methods

2) can only return a void type

3) The specified parameters must be primitive data types, arrays of primitive data types, or collections of primitive data types

4) Methods with the future annotation cannot take sObjects or objects as arguments.

5) You can invoke future methods the same way you invoke any other method. However, a future method can’t invoke another future method

6) No more than 50 method calls per Apex invocation

7) Asynchronous calls, such as @future or executeBatch, called in a startTest, stopTest block, do not count against your limits for the number of queued jobs

8) The maximum number of future method invocations per a 24-hour period is 250,000 or the number of user licenses in your organization multiplied by 200, whichever is greater

9) To test methods defined with the future annotation, call the class containing the method in a startTest(), stopTest() code block. All asynchronous calls made after the startTest method are collected by the system. When stopTest is executed, all asynchronous processes are run synchronousl

13.How does Queueable Apex differ from Future methods?

Queueable Apex is similar to future methods in that they're both queued for execution, but they provide us these additional benefits.

When you queue a Queueable Apex, you get a job ID, that can be used to trace it easily, which is not possible in case of future methods.

You can use non-primitive datatypes in Queueable Apex, like objects and sObjects, which is not possible in case of future methods, because it supports only primitive data types as params.

You can chain jobs, by calling another starting a second job from a running job, which is not possible in case of future methods, because we can't call another future method from a future context.

14.Can you write a sample Queueable Job?

Create a class, implement the Queueable interface, and override the execute method.

```
public class QueueableApexExample implements Queueable {
public void execute(QueueableContext context) {
//some process
}
}
```

15.How can I use this Job Id to trace the Job?

Just perform a SOQL query on AsyncApexJob by filtering on the job ID.

AsyncApexJob jobInfo = [SELECT Status,NumberOfErrors FROM AsyncApexJob WHERE Id=:jobID];

16.Can I do callouts from a Queueable Job?

Yes, you have to implement the Database.AllowsCallouts interface to do callouts from Queueable Jobs.

17.How many numbers of jobs, I can queue using System.enqueueJob() at a time?

You can add up to 50 jobs to the queue with System.enqueueJob in a single transaction in Synchronous apex. In asynchronous transactions, you can add only one job to the queue.

18.Can I call Queueable from a batch?

Yes, But you're limited to just one System.enqueueJob call per execute in the Database.Batchable class. Salesforce has imposed this limitation to prevent explosive execution.

19.If I have written more than one System.enqueueJob call, what will happen?

System will throw LimitException stating "Too many queueable jobs added to the queue: N"

20.I have a use case to call more than one Queueable Jobs from a Batch apex, how can I achieve it?

Since we can’t call more than one Queueable Job from each execution Context, We can go for scheduling the Queueable Jobs.

The approach is we need to first check how many queueable jobs are added in the queue in the current transaction by making use of Limits class. If the number has reached the limit, then call a schedulable class and enqueue the queueable class from the execute method of a schedulable class.

…………………………………………………………………………………………………..

t is a key benefit of asynchronous processing?

I)Higher governor and execution limits

II)Unlimited number of external callouts

III)Override organization-wide sharing defaults

IV)Enabling turbo mode for transactional processing

Show Answers

Q2)Batch Apex is typically the best type of asynchronous processing when you want to:

I)Make a callout to a web service when a user updates a record.

II)Schedule a task to run on a weekly basis.

III)Update all records in your org.

IV)Send a rickroll email to a contact.

Show Answers

#2.Use Future Methods

Hands-on Challenge

Earn+500 points

YOUR CHALLENGE

Create an Apex class that uses the @future annotation to update Account records.

Create an Apex class with a future method that accepts a List of Account IDs and updates a custom field on the Account object with the number of contacts associated to the Account. Write unit tests that achieve 100% code coverage for the class. Every hands-on challenge in this module asks you to create a test class.

#Create a field on the Account object:

#Label: Number Of Contacts

#Name: Number_Of_Contacts

#Type: Number

#This field will hold the total number of Contacts for the Account

#Create an Apex class:

#Name: AccountProcessor

#Method name: countContacts

#The method must accept a List of Account IDs

#The method must use the @future annotation

#The method counts the number of Contact records associated to each Account ID passed to the method and updates the 'Number_Of_Contacts__c' field with this value

#Create an Apex test class:

#Name: AccountProcessorTest

#The unit tests must cover all lines of code included in the AccountProcessor class, resulting in 100% code coverage.

#Before verifying this challenge, run your test class at least once using the Developer Console Run All feature

My Trailhead Playground 1

Last used on 11/26/2021

Complete the Challenge to **Earn 500Points.**

#3.Use Batch Apex

Hands-on Challenge

+500 points

YOUR CHALLENGE

Create an Apex class that uses Batch Apex to update Lead records.

Create an Apex class that implements the Database.Batchable interface to update all Lead records in the org with a specific LeadSource.

#Create an Apex class:

#Name: LeadProcessor

#Interface: Database.Batchable

#Use a QueryLocator in the start method to collect all Lead records in the org

#The execute method must update all Lead records in the org with the LeadSource value of Dreamforce

#Create an Apex test class:

#Name: LeadProcessorTest

#In the test class, insert 200 Lead records, execute the LeadProcessor Batch class and test that all Lead records were updated correctly

#The unit tests must cover all lines of code included in the LeadProcessor class, resulting in 100% code coverage

#Before verifying this challenge, run your test class at least once using the Developer Console Run All feature

My Trailhead Playground 1

Last used on 11/26/2021

Complete the Challenge to **Earn 500Points.**

#4.Control Processes With Queueable Apex

[ERP Production Planning Module Details]()

Hands-on Challenge

Earn +500 points

YOUR CHALLENGE

Create an Queueable Apex class that inserts Contacts for Accounts.

Create a Queueable Apex class that inserts the same Contact for each Account for a specific state.

#Create an Apex class:

#Name: AddPrimaryContact

#Interface: Queueable

#Create a constructor for the class that accepts as its first argument a Contact sObject and a second argument as a string for the State abbreviation

#The execute method must query for a maximum of 200 Accounts with the BillingName specified by the State abbreviation passed into the constructor and insert the Contact sObject record associated to each Account. Look at the sObject clone() method.

#Create an Apex test class:

#Name: AddPrimaryContactTest

#In the test class, insert 50 Account records for BillingName NY and 50 Account records for BillingName CA

#Create an instance of the AddPrimaryContact class, enqueue the job, and assert that a Contact record was inserted for each of the 50 Accounts with the BillingName of CA

#The unit tests must cover all lines of code included in the AddPrimaryContact class, resulting in 100% code coverage

#Before verifying this challenge, run your test class at least once using the Developer Console Run All feature

My Trailhead Playground 1

Last used on 11/26/2021

Complete the Challenge to Earn 500Points

#5.Schedule Jobs Using The Apex Scheduler

Hands-on Challenge

Earn +500 points

YOUR CHALLENGE

Create an Apex class that uses Scheduled Apex to update Lead records.

Create an Apex class that implements the Schedulable interface to update Lead records with a specific LeadSource. (This is very similar to what you did for Batch Apex.)

#Create an Apex class:

#Name: DailyLeadProcessor

#Interface: Schedulable

#The execute method must find the first 200 Lead records with a blank LeadSource field and update them with the LeadSource value of Dreamforce

#Create an Apex test class:

#Name: DailyLeadProcessorTest

#In the test class, insert 200 Lead records, schedule the DailyLeadProcessor class to run and test that all Lead records were updated correctly

#The unit tests must cover all lines of code included in the DailyLeadProcessor class, resulting in 100% code coverage.

#Before verifying this challenge, run your test class at least once using the Developer Console Run All feature

My Trailhead Playground 1

Last used on 11/26/2021

Complete the Challenge to Earn 500Points.

#6.Monitor Asynchronous Apex

Q1)What type of jobs do not show up in the Apex Flex Queue.

I)Future Method Jobs

II)Batch Apex Jobs

III)Queueable Apex Jobs

IV)Scheduled Apex Jobs

Show Answers

Q2.Batch Apex is typically the best type of asynchronous processing when you want to:

I)Make a callout to a web service when a user updates a record.

II)Schedule a task to run on a weekly basis.

III)Update all records in your org.

IV)Send a rickroll email to a contact.