

**Developers** 



~

Queueable Interface



Apex Reference Guide / System Namespace / OrgLimits Class

# **Queueable Interface**

Enables the asynchronous execution of Apex jobs that can be monitored.

## **Namespace**

System

## Usage

To execute Apex as an asynchronous job, implement the Queueable interface and add the processing logic in your implementation of the execute method.

To implement the Queueable interface, you must first declare a class with the implements keyword as follows:

```
public class MyQueueableClass implements Queueable {
```

Next, your class must provide an implementation for the following method:

```
public void execute(QueueableContext context) {
    // Your code here
}
```

Your class and method implementation must be declared as public or global.

To submit your class for asynchronous execution, call the System.enqueueJob by passing it an instance of your class implementation of the Queueable interface as follows:

```
□ | □
ID jobID = System.enqueueJob(new MyQueueableClass());
```

- Queueable Methods
- Queueable Example Implementation

### See Also

• Apex Developer Guide: Queueable Apex

## **Queueable Methods**

The following are methods for Queueable.

execute(context)
 Executes the queueable job.

## execute(context)



\_

#### **Parameters**

#### context

Type: QueueableContext

Contains the job ID.

#### **Return Value**

Type: Void

## **Queueable Example Implementation**

This example is an implementation of the Queueable interface. The execute method in this example inserts a new account.

```
public class AsyncExecutionExample implements Queueable {
   public void execute(QueueableContext context) {
        Account a = new Account(Name='Acme',Phone='(415) 555-1212');
        insert a;
   }
}
```

To add this class as a job on the queue, call this method:

After you submit your queueable class for execution, the job is added to the queue and will be processed when system resources become available. You can monitor the status of your job programmatically by querying AsyncApexJob or through the user interface in Setup by entering Apex Jobs in the Quick Find box, then selecting **Apex Jobs**.

To query information about your submitted job, perform a SOQL query on AsyncApexJob by filtering on the job ID that the System.enqueueJob method returns. This example uses the jobID variable that was obtained in the previous example.

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

Similar to future jobs, queueable jobs don't process batches, and so the number of processed batches and the number of total batches are always zero.

## **Testing Queueable Jobs**

This example shows how to test the execution of a queueable job in a test method. A queueable job is an asynchronous process. To ensure that this process runs within the test method, the job is submitted to the queue between the Test.startTest and Test.stopTest block. The system executes all asynchronous processes started in a test method synchronously after the Test.stopTest statement. Next, the test method verifies the results of the queueable job by querying the account that the job created.

```
@isTest

public class AsyncExecutionExampleTest {

    static testmethod void test1() {

    // startTest/stopTest block to force async processes
```



```
// by verifying that the record was created.
        \ensuremath{//} This query returns only the account created in test context by the
        // Queueable class method.
        Account acct = [SELECT Name, Phone FROM Account WHERE Name='Acme' LIMIT 1];
        System.assertNotEquals(null, acct);
        System.assertEquals('(415) 555-1212', acct.Phone);
    }
}
```



#### Note

The ID of a queueable Apex job isn't returned in test context-System.enqueueJob returns null in a running test.

#### **DID THIS ARTICLE SOLVE YOUR ISSUE?**

Let us know so we can improve!

Share your feedback











### **DEVELOPER CENTERS**

Heroku MuleSoft Tableau

**Commerce Cloud** Lightning Design System

Einstein

Quip

### POPULAR RESOURCES

Documentation Component Library

APIs

Trailhead Sample Apps

Podcasts

AppExchange

### COMMUNITY

Trailblazer Community **Events and Calendar** 

**Partner Community** 

Blog

Salesforce Admins Salesforce Architects

© Copyright 2025 Salesforce, Inc. All rights reserved. Various trademarks held by their respective owners. Salesforce, Inc. Salesforce Tower, 415 Mission Street, 3rd Floor, San Francisco, CA 94105, United States

Privacy Information Terms of Service Legal Use of Cookies Cookie Preferences



Your Privacy Choices

Responsible Disclosure

Contact