

## Overview

Apex code is a strongly-typed programming language that executes on the Force.com platform. Using Apex code, you can add business logic to applications, write database triggers, and create Visualforce controllers. Apex code has a tight integration with the database and query language, web services, and email handling support. It also includes features such as asynchronous execution and support for testing.

### Important Reserved Words

Keyword	Description	Example
abstract	Declares a class that contains abstract methods that only have their signature and no body defined. Can also define methods.	<pre>public <b>abstract</b> class Foo {     protected void method1() { /*... */ }     <b>abstract</b> Integer abstractMethod(); }</pre>
break	Exits the entire loop	<pre>while(reader.hasNext()) {     if (reader.getEventType() == END) {         <b>break</b>;     };     // process     reader.next(); }</pre>
catch	Identifies a block of code that can handle a particular type of exception	<pre>try {     // Your code here } <b>catch</b> (ListException e) {     // List Exception handling code here }</pre>
class	Defines a class	<pre>private <b>class</b> Foo {     private Integer x;     public Integer getX() { return x; } }</pre>
continue	Skips to the next iteration of the loop	<pre>while (checkBoolean) {     if (condition)         <b>continue</b>;     // do some work }</pre>
do	Defines a do-while loop that executes repeatedly (at least once) while a Boolean condition remains true	<pre>Integer count = 1; <b>do</b> {     System.debug(count);     count++; } <b>while</b> (count &lt; 11);</pre>
else	Defines the else portion of an if-else statement, that executes if the initial evaluation is untrue	<pre>Integer x, sign; if (x==0) {     sign = 0; } <b>else</b> {     sign = 1; }</pre>

### Important Reserved Words *continued*

Keyword	Description	Example
enum	Defines an enumeration type on a finite set of values	<pre>public <b>enum</b> Season {WINTER,     SPRING, SUMMER, FALL}; Season e = Season.WINTER;</pre>
extends	Defines a class or interface that extends another class or interface	<pre>public class MyException <b>extends</b>     Exception {}  try {     Integer i;     if (i &lt; 5) throw new MyException(); } <b>catch</b> (MyException e) {     // Your MyException handling code }</pre>
false	Identifies an untrue value assigned to a Boolean	<pre>Boolean isNotTrue = <b>false</b>;</pre>
final	Defines constants	<pre>public class myCls {     static <b>final</b> Integer intConstant; }</pre>
finally	Identifies a block of code that is guaranteed to execute	<pre>try {     // Your code here } <b>catch</b> (ListException e) {     // List Exception handling code } <b>finally</b> {     // will execute with or without     // exception }</pre>
for	Defines a loop. The three types of for loops are: iteration using a variable, iteration over a list, and iteration over a query	<pre><b>for</b> (Integer i = 0, j = 0; i &lt; 10; i++) { System.debug(i+1); }  Integer[] myInts = new Integer[] {1, 8, 9}; <b>for</b> (Integer i : myInts) {     System.debug(i); }  String s = 'Acme'; <b>for</b> (Account a : [SELECT Id, Name, FROM account WHERE Name LIKE :(s+'%')]) {     // Your code }</pre>
global	Defines a class, method, or variable that can be used by any Apex that has access to the class, not just the Apex in the same application.	<pre><b>global</b> class myClass {     webservice static void     makeContact(String lastName) {         // do some work     } }</pre>

# Apex Code Cheatsheet

## Important Reserved Words *continued*

Keyword	Description	Example
if	Defines a condition, used to determine whether a code block should be executed	<pre>Integer i = 1; <b>if</b> (i &gt; 0) {     // do something; }</pre>
implements	Declares a class or interface that implements an interface	<pre>global class CreateTaskEmailExample <b>implements</b> Messaging. InboundEmailHandler {     global Messaging.InboundEmailResult handleInboundEmail(Messaging. inboundEmail email, Messaging.InboundEnvelope env){     // do some work, return value; } }</pre>
instanceOf	Verifies at runtime whether an object is actually an instance of a particular class	<pre><b>if</b> (reports.get(0) <b>instanceof</b> CustomReport) {     // Can safely cast     CustomReport c = (CustomReport) reports.get(0); } <b>else</b> {     // Do something with the non-custom-report. }</pre>
interface	Defines a data type with method signatures. Classes implement interfaces. An interface can extend another interface.	<pre><b>public interface</b> PO {     <b>public void</b> doWork(); } <b>public class</b> MyPO <b>implements</b> PO {     <b>public override</b> doWork() {         // actual implementation     } }</pre>
new	Creates a new object, sObject, or collection instance	<pre>Foo f = <b>new</b> Foo(); MyObject__c mo = <b>new</b> MyObject__c(Name= 'hello'); List&lt;Account&gt; la = <b>new</b> List&lt;Account&gt;();</pre>
null	Identifies a null constant that can be assigned to any variable	<pre>Boolean b = <b>null</b>;</pre>
override	Defines a method as overriding another method on a class that's being extended or implemented.	<pre><b>public virtual class</b> V {     <b>public virtual void</b> foo()     { /*Does nothing*/ } }  <b>public class</b> RealV <b>implements</b> V {     <b>public override</b> void foo() {         // Do something real     } }</pre>

## Important Reserved Words *continued*

Keyword	Description	Example
private	Defines a class, method, or variable that is only known locally, within the section of code in which it is defined	<pre><b>public class</b> OuterClass {     // Only visible to methods and     // statements within OuterClass     <b>private static final</b> Integer     myInt; }</pre>
protected	Defines a method or variable that is visible to any inner classes in the defining Apex class, and to classes that extend the defining class	<pre><b>public class</b> Foo {     <b>public void</b> quiteVisible();     <b>protected void</b> lessVisible(); }</pre>
public	Defines a method or variable that can be used by any Apex in this application or namespace	<pre><b>public class</b> Foo {     <b>public void</b> quiteVisible();     <b>private void</b> almostInvisible(); }</pre>
return	Returns a value from a method	<pre><b>public Integer</b> meaningOfLife() {     <b>return</b> 42; }</pre>
static	Defines initialization code, or a method or variable that is initialized only once and is associated with a class	<pre><b>public class</b> OuterClass {     // Associated with instance     <b>public static final</b> Integer     myInt;      // Initialization code     <b>static</b> {         myInt = 10;     } }</pre>
super	Invokes a constructor or method on a parent class that's designated as virtual or abstract. Can be used in methods only if they're designated with the override keyword.	<pre><b>public class</b> AnotherChildClass <b>extends</b> InnerClass {     AnotherChildClass(String s) {         <b>super</b>();         // different constructor,     } }</pre>

## Important Reserved Words *continued*

Keyword	Description	Example
this	Represents the current instance of a class, or calls the previous constructor in a constructor chain	<pre>public class Foo {     public Foo(String s) { /* ... */ }     public Foo() {         <b>this</b>('memes repeat'); } }</pre>
throw	Throws an exception, signaling that an error has occurred	<pre>public class MyException extends Exception {} try {     Integer i;     if (i &lt; 5)         <b>throw</b> new MyException(); } catch (MyException e) {     // Your MyException handling     // code here }</pre>
transient	Declares instance variables that cannot be saved, and should not be transmitted as part of the view state, in Visualforce controllers and extensions	<pre><b>transient</b> integer currentValue;</pre>
trigger	Defines a trigger on an sObject	<pre><b>trigger</b> MyAccountTrigger on Account (before insert, before update) {     if (Trigger.isBefore) {         for (Account a : Trigger.old) {              if (a.Name != 'okToDelete') {                 a.addError(                     'You can\'t delete this                     record!');             }         }     } }</pre>
true	Identifies a true value assigned to a Boolean	<pre>Boolean mustIterate = <b>true</b>;</pre>
try	in which you can handle an exception if one occurs	<pre><b>try</b> {     // Your code here } catch (ListException e) {     // List Exception handling code     // here }</pre>

## Important Reserved Words *continued*

Keyword	Description	Example
webservice	Defines a static method that is exposed as a Web service method that can be called by external client applications. Web service methods must be defined in a global class.	<pre>global class MyWebService {     <b>webservice</b> static Id     makeContact(         String lastName, Account a) {          Contact c = new Contact(             LastName = 'Weissman',             AccountId = a.Id);         insert c;         return c.Id;     } }</pre>
while	Executes a block of code repeatedly as long as a particular Boolean condition remains true	<pre>Integer count=1; <b>while</b> (count &lt; 11) {     System.debug(count);     count++; }</pre>
with sharing	Enforces sharing rules that apply to the current user. If absent, code is run under the default system context.	<pre>public <b>with sharing</b> class SharingClass {     // Code will enforce current     user's // sharing rules }</pre>
without sharing	Ensures that the sharing rules of the current user are not enforced	<pre>public <b>without sharing</b> class noSharing {     // Code won't enforce the current     // user's sharing rules }</pre>
virtual	Defines a class or method that allows extension and overrides. You can't override a method with the override keyword unless the class or method has been defined as virtual.	<pre>public <b>virtual</b> class MyException extends Exception {     // Exception class member     // variable     public Double d;      // Exception class constructor     MyException(Double d) {         this.d = d;     }      // Exception class method     protected void doIt() {} }</pre>

# Apex Code Cheatsheet

## Annotations

Keyword	Description	Example
@future	Denotes methods that are executed asynchronously	<pre>global class MyFutureClass {     @future     static void myMethod(         String a, Integer i) {         System.debug(             'Method called with: ' + a +             ' and ' + i);         // do callout, or execute         // other long-running code     } }</pre>
@isTest	Denotes classes that only contain code used for testing your application. These classes don't count against the total amount of Apex used by your organization.	<pre>@isTest private class MyTest {     // Methods for testing }</pre>
@isTest(OnInstall=true)	Denotes a test class or test method that executes on package installation	<pre>@isTest (OnInstall=true) private class TestClass { }</pre>
@isTest(SeeAllData=true)	Denotes a test class or test method that has access to all data in the organization, including pre-existing data that the test didn't create. The default is false.	<pre>@isTest (SeeAllData=true) private class TestClass { }</pre>
@deprecated	Denotes methods, classes, exceptions, enums, interfaces, or variables that can no longer be referenced in subsequent releases of the managed package in which they reside	<pre>@deprecated public void limitedShelfLife() { }</pre>

## Annotations *continued*

Annotation	Description	Example
@readOnly	Denotes methods that can perform queries unrestricted by the number of returned rows limit for a request	<pre>@readOnly private void doQuery() { }</pre>
@remoteAction	Denotes Apex controller methods that JavaScript code can call from a Visualforce page via JavaScript remoting. The method must be static and either public or global.	<pre>@remoteAction global static String getId(     String s) { }</pre>
@restResource	Denotes a class that is available as a REST resource. The class must be global. The urlMapping parameter is your resource's name and is relative to https://instance.salesforce.com/services/apexrest/.	<pre>@restResource (urlMapping=     '/Widget/*') global with sharing class MyResource() { }</pre>
@HttpGet, @HttpPost, @HttpPatch, @HttpPut, @HttpDelete	Denotes a REST method in a class annotated with @restResource that the runtime invokes when a client sends an HTTP GET, POST, PATCH, PUT, or DELETE respectively.  The methods defined with any of these annotations must be global and static.	<pre>@HttpGet global static MyWidget__c doGet() { }  @HttpPost global static void doPost() { }  @HttpDelete global static void doDelete() { }</pre>

## Primitive Types

Annotation	Description	Example
Blob	Binary data stored as a single object	<pre><b>Blob</b> myBlob = <b>Blob</b>.valueOf('idea');</pre>
Boolean	Value that can only be assigned true, false, or null	<pre><b>Boolean</b> isWinner = true;</pre>
Date	Particular day	<pre><b>Date</b> myDate = Date.today(); <b>Date</b> weekStart = myDate.toStartOfWeek();</pre>
Datetime	Particular day and time	<pre><b>Datetime</b> myDateTime = Datetime.now(); <b>Datetime</b> newDateTime = myDateTime.addMonths(2);</pre>
Decimal	Number that includes a decimal point. Decimal is an arbitrary precision number.	<pre><b>Decimal</b> myDecimal = 12.4567; Decimal divDec = myDecimal.divide (7, 2, System.RoundingMode.UP); system.assertEquals(divDec, 1.78);</pre>
Double	64-bit number that includes a decimal point. Minimum value $-2^{63}$ . Maximum value of $2^{63}-1$	<pre><b>Double</b> d=3.14159;</pre>
ID	18-character Force.com record identifier	<pre><b>ID</b> id='003000000003T2PGAA0';</pre>
Integer	32-bit number that doesn't include a decimal point. Minimum value $-2,147,483,648$ – maximum value of $2,147,483,647$	<pre><b>Integer</b> i = 1;</pre>
Long	64-bit number that doesn't include a decimal point. Minimum value of $-2^{63}$ – maximum value of $2^{63}-1$ .	<pre><b>Long</b> l = 2147483648L;</pre>
Object	Any data type that is supported in Ape	<pre><b>Object</b> obj = 10; // Cast the object to an integer. Integer i = (Integer)obj; System.assertEquals(10, i);  <b>Object</b> obj = new MyApexClass(); // Cast the object to the // MyApexClass custom type. MyApexClass mc = (MyApexClass) obj; // Access a method on the // user-defined class. mc.someClassMethod();</pre>

## Primitive Types *continued*

Annotation	Description	Example
String	Set of characters surrounded by single quotes	<pre><b>String</b> s = 'repeating memes';</pre>
Object	Any data type that is supported in Apex	<pre><b>String</b> s = 'repeating memes';</pre>
Time	Particular time	<pre><b>Object</b> obj = 10; // Cast the object to an integer. Integer i = (Integer)obj; System.assertEquals(10, i);  <b>Object</b> obj = new MyApexClass(); // Cast the object to the // MyApexClass custom type. MyApexClass mc = (MyApexClass)obj; // Access a method on the // user-defined class. mc.someClassMethod();</pre>

## Trigger Context Variables

Variable	Operators
isExecuting	Returns true if the current context for the Apex code is a trigger only
isInsert	Returns true if this trigger was fired due to an insert operation
isUpdate	Returns true if this trigger was fired due to an update operation
isDelete	Returns true if this trigger was fired due to a delete operation
isBefore	Returns true if this trigger was fired before any record was saved
isAfter	Returns true if this trigger was fired after all records were saved
isUndelete	Returns true if this trigger was fired after a record was recovered from the Recycle Bin
new	Returns a list of the new versions of the sObject records. (This sObject list is available only in <code>insert</code> and <code>update</code> triggers. The included records can be modified only in <code>before</code> triggers.)
newMap	A map of IDs to the new versions of the sObject records. (Only available in <code>before update</code> , <code>after insert</code> , and <code>after update</code> triggers.)
old	Returns a list of the old versions of the sObject records. (Only available in <code>update</code> and <code>delete</code> triggers.)
oldMap	A map of IDs to the old versions of the sObject records. (Only available in <code>update</code> and <code>delete</code> triggers.)
size	The total number of records in a trigger invocation, both old and new.

# Apex Code Cheatsheet

## Collection Types

Annotation	Description	Example
List	Ordered collection of typed primitives, sObjects, objects, or collections that are distinguished by their indices	<pre>// Create an empty list of String List&lt;String&gt; myList = new List&lt;String&gt;(); myList.add('hi'); String x = myList.get(0);  // Create list of records from a query List&lt;Account&gt; accs = [SELECT Id, Name FROM Account LIMIT 1000];</pre>
Map	Collection of key-value pairs where each unique key maps to a single value. A key can be any primitive data type, while a value can be a primitive, an sObject, a collection type, or an object.	<pre>Map&lt;String, String&gt; MyStrings = new Map&lt;String, String&gt;{     'a' =&gt; 'b', 'c' =&gt; 'd'.toUpperCase();      Account myAcct = new Account();     Map&lt;Integer, Account&gt; m = new Map&lt;Integer, Account&gt;();     m.put(1, myAcct);</pre>
Set	Unordered collection that doesn't contain any duplicate elements.	<pre>Set&lt;Integer&gt; s = new Set&lt;Integer&gt;(); s.add(12); s.add(12); System.assert(s.size()==1);</pre>

## Standard Interfaces (Subset)

### Database.Batchable

```
global (Database.QueryLocator | Iterable<sObject>)
    start(Database.BatchableContext bc) {}
global void execute(Database.BatchableContext BC,
    list<P>){}
global void finish(Database.BatchableContext BC){}
```

### Schedulable

```
global void execute(ScheduleableContext SC) {}
```

### Messaging.InboundEmailHandler

```
global Messaging.InboundEmailResult handleInboundEmail(Messaging.InboundEmail email, Messaging.InboundEnvelope env){}
```

### Comparable

```
global Integer compareTo(Object compareTo) {}
```

## Apex Data Manipulation Language (DML) Operations

Annotation	Description	Example
insert	Adds one or more records	<pre>Lead l = new Lead(Company='ABC', LastName='Smith'); insert l;</pre>
delete	Deletes one or more records	<pre>Account[] webAccts = [SELECT Id, Name FROM Account WHERE Name = 'DotCom']; try {     delete webAccts; } catch (DmlException e) {     // Process exception here }</pre>
merge	Merges up to three records of the same type into one of the records, deleting the others, and re-parenting any related records	<pre>List&lt;Account&gt; ls = new List&lt;Account&gt;{     new Account(Name='Acme Inc.'),     new Account(Name='Acme')}; insert ls; Account masterAcct = [SELECT Id, Name FROM Account WHERE Name = 'Acme Inc.' LIMIT 1]; Account mergeAcct = [SELECT Id, Name FROM Account WHERE Name = 'Acme' LIMIT 1]; try { merge masterAcct mergeAcct; } catch (DmlException e) { }</pre>
undelete	Restores one or more records from the Recycle Bin	<pre>Account[] savedAccts = [SELECT Id, Name FROM Account WHERE Name = 'Trump' ALL ROWS]; try { undelete savedAccts; } catch (DmlException e) { }</pre>
update	Modifies one or more existing records	<pre>Account a = new Account(Name='Acme2'); insert(a); Account myAcct = [SELECT Id, Name, BillingCity FROM Account WHERE Name = 'Acme2' LIMIT 1]; myAcct.BillingCity = 'San Francisco'; try {     update myAcct; } catch (DmlException e) { }</pre>
upsert	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.	<pre>Account[] acctsList = [SELECT Id, Name, BillingCity FROM Account WHERE BillingCity = 'Bombay']; for (Account a : acctsList) {     {a.BillingCity = 'Mumbai';} Account newAcct = new Account(     Name = 'Acme', BillingCity = 'San Francisco'); acctsList.add(newAcct); try { upsert acctsList; } catch (DmlException e) { }</pre>

## Commonly Used Methods

### System Class

abortJob	assert
assertEquals	assertNotEquals
currentPageReference	currentTimeMillis
debug	enqueueJob
equals	getApplication
ReadWriteMode	hashCode
isBatch	isFuture
isScheduled	nowprocess
purgeOldAsyncJobs	requestVersion
resetPassword	runAs
schedule	scheduleBatch
setPassword	submit
today	

### Math Class

abs	acos	asin	atan	atan2	cbt
ceil	cos	cosh	exp	floor	log
log10	max	min	mod	pow	random
rint	round	roundToLong	signum	sin	
sinh	sqrt	tan	tanh		

### DescribeSObjectResult Class

fields	fieldSets
getChildRelationships	getKeyPrefix
getLabel	getLabelPlural
getLocalName	getName
getRecordTypeInfo	getRecordTypeInfoByID
getSObjectType	isAccessible
getRecordTypeInfoByName	isCreateable
isCustom	isCustomSetting
isDeletable	isDeprecatedAndHidden
isFeedEnabled	isMergeable
isQueryable	isSearchable
isUndeletable	isUpdateable

```

Schema.RecordTypeInfo rtByName =
rtMapByName.get(rt.name);
Schema.DescribeSObjectResult d =
Schema.SObjectType.Account;
  
```

## Commonly Used Methods *continued*

### DescribeFieldResult Class

getByteLength	getCalculatedFormula
getController	getDefaultValue
getDefaultValueFormula	getDigits
getInlineHelpText	getLabel
getLengthgetLocalName	getName
getPicklistValues	getPrecision
getReferenceTargetField	getReferenceTo
getRelationshipName	getRelationshipOrder
getScale	getSOAPType
getSObjectField	getType
isAccessible	isAutoNumber
isCalculated	isCascadeDelete
isCaseSensitive	isCreateable
isCustom	isDefaultedOnCreate
isDependantPicklist	isDeprecatedAndHidden
isExternalID	isFilterable
isGroupable	isHtmlFormatted
isIdLookup	isNameField
isNamePointing	isNillable
isPermissionable	isRestrictedDelete
isRestrictedPicklist	isSortable
isUnique	isUpdateable
isWriteRequiresMasterRead	

```

Schema.DescribeFieldResult f =
Schema.SObjectType.Account.fields.Name;
  
```

### Limits Class

getAggregateQueries	getLimitAggregateQueries
getAsyncCalls	getLimitAsyncCalls
getCallouts	getLimitCallouts
getCpuTime	getLimitCpuTime
getDMLRows	getLimitDMLRows
getDMLStatements	getLimitDMLStatements
getEmailInvocations	getLimitEmailInvocations
getFutureCalls	getLimitFutureCalls
getHeapSize	getLimitHeapSize
getMobilePushApexCalls	getLimitMobilePushApexCalls
getQueries	getLimitQueries
getQueryLocatorRows	getLimitQueryLocatorRows
getQueryRows	getLimitQueryRows
getQueueableJobs	getLimitQueueableJobs
getSoslQueries	getLimitSoslQueries

### UserInfo Class

getDefaultCurrency	getFirstName
getLanguage	getLastName
getLocale	getName
getOrganizationId	getOrganizationName
getProfileId	getSessionId
getTimeZone	getUiTheme
getUiThemeDisplayed	getUserEmail
getUserId	getUserName
getUserRoleId	getUserType
isCurrentUserLicensedq	isMultiCurrencyOrganization



[developer.salesforce.com](http://developer.salesforce.com)