Apex Code Cheatsheet

salesforce force.com

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.	public **abstract** class Foo {
		 protected void method1() { /*… */ }
		 abstract Integer abstractMethod();
		}

break	Exits the entire loop	while(reader.hasNext()) {
		 if (reader.getEventType() == END) {
		   break;
		 };
		 // process
		 reader.next();
		}

catch	Identifies a block of code that can handle a particular type of exception	try {
		 // Your code here
		
		} **catch** (ListException e) {
		 // List Exception handling code here
		}

class	Defines a class	private **class** Foo {
		 private Integer x;
		 public Integer getX() { return x; }
		}

continue	Skips to the next iteration of the loop	while (checkBoolean) {
		 if (condition)
		   {**continue**; }
		 // do some work
		}

do	Defines a do-while loop that executes repeatedly (at least once) while a Boolean condition remains true	Integer count = 1;
		do {
		  System.debug(count);
		  count++;
		} while (count < 11);

else	Defines the else portion of an if-else statement, that executes if the initial evaluation is untrue	Integer x, sign;
		if (x==0) {
		 sign = 0;
		} **else** {
		 sign = 1;
		}

Important Reserved Words *continued*

Keyword	Description	Example

enum	Defines an enumeration type on a finite set of values	public **enum** Season {WINTER,
		SPRING, SUMMER, FALL};
		Season e = Season.WINTER;

extends	Defines a class or interface that extends another class or interface	public class MyException **extends**
		  Exception {}
		
		try {
		  Integer i;
		  if (i < 5) throw new MyException();
		} catch (MyException e) {
		  // Your MyException handling code
		}

false	Identifies an untrue value assigned to a Boolean	Boolean isNotTrue = **false**;

final	Defines constants	public class myCls {
		  static **final** Integer intConstant;
		}

finally	Identifies a block of code that is guaranteed to execute	try {
		 // Your code here
		} catch (ListException e) {
		 // List Exception handling code
		} **finally** {
		 // will execute with or without
		 // exception
		}

for	Defines a loop. The three types of for loops are: iteration using a variable, iteration over a list, and iteration over a query	**for** (Integer i = 0, j = 0; i < 10;
		i++) { System.debug(i+1);
		}
		
		Integer[] myInts = new Integer[]
		{1, 8, 9};
		for (Integer i : myInts) {
		  System.debug(i);
		}
		
		String s = 'Acme';
		for (Account a : [SELECT Id, Name,
		FROM account
		WHERE Name LIKE :(s+'%')]) {
		  // Your code
		}

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.	**global** class myClass {
		 webService static void
		  makeContact(String lastName) {
		  // do some work
		}

Apex Code Cheatsheet

Important Reserved Words *continued*

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

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	`public class OuterClass { // Only visible to methods and // statements within OuterClass private static final Integer myInt; }`
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	`public class Foo { public void quiteVisible(); protected void lessVisible(); }`
public	Defines a method or variable that can be used by any Apex in this application or namespace	`public class Foo { public void quiteVisible(); private void almostInvisible(); }`
return	Returns a value from a method	`public Integer meaningOfLife() { return 42; }`
static	Defines initialization code, or a method or variable that is initialized only once and is associated with a class	`public class OuterClass { // Associated with instance public static final Integer myInt; // Initialization code static { myInt = 10; } }`
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.	`public class AnotherChildClass extends InnerClass { AnotherChildClass(String s) { super(); // different constructor, } }`

Important Reserved Words *continued*

Keyword	Description	Example

this	Represents the current instance of a class, or calls the previous constructor in a constructor chain	`public class Foo {`
		` public Foo(String s) { /* … */}`
		` public Foo() {`
		`   this('memes repeat'); }`
		`}`

throw	Throws an exception, signaling that an error has occurred	`public class MyException extends`
		`  Exception {}`
		`try {`
		` Integer i;`
		` if (i < 5)`
		`     throw new MyException();`
		`} catch (MyException e) {`
		`  // Your MyException handling`
		`  // code here`
		`}`

transient	Declares instance variables that cannot be saved, and should not be transmitted as part of the view state, in Visualforce controllers and extensions	`transient integer currentValue;`

trigger	Defines a trigger on an sObject	`trigger 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!');`
		`   }`
		`  }`
		`}`

true	Identifies a true value assigned to a Boolean	`Boolean mustIterate = true;`

try	in which you can handle an exception if one occurs	`try {`
		`  // Your code here`
		
		`} catch (ListException e) {`
		`  // List Exception handling code`
		`  // here`
		`}`

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.	`global class MyWebService {`
		` webService static Id`
		`makeContact(`
		`   String lastName, Account a) {`
		
		`  Contact c = new Contact(`
		`    LastName = 'Weissman',`
		`    AccountId = a.Id);`
		`    insert c;`
		`    return c.Id;`
		`  }`
		`}`

while	Executes a block of code repeatedly as long as a particular Boolean condition remains true	`Integer count=1;`
		`while (count < 11) {`
		`  System.debug(count);`
		`  count++;`
		`}`

with sharing	Enforces sharing rules that apply to the current user. If absent, code is run under the default system context.	`public with sharing class`
		`SharingClass {`
		`// Code will enforce current`
		`user's // sharing rules`
		`}`

without sharing	Ensures that the sharing rules of the current user are not enforced	`public without sharing class`
		`noSharing {`
		`// Code won't enforce the current`
		`// user's sharing rules`
		`}`

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.	`public virtual 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() {}`
		`}`

Apex Code Cheatsheet

Annotations

Keyword	Description	Example

@future	Denotes methods that are executed asynchronously	`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`
		`  }`
		`}`

@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.	`@isTest private class MyTest {`
		`  // Methods for testing`
		`}`

@isTest(OnInstall=true)	Denotes a test class or test method that executes on package installation	`@isTest(OnInstall=true)`
		`private class TestClass {`
		`}`

@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.	`@isTest(SeeAllData=true)`
		`private class TestClass {`
		`}`

@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	`@deprecated`
		`public void limitedShelfLife() {`
		`}`

Annotations *continued*

Annotation	Description	Example

@readOnly	Denotes methods that can perform queries unrestricted by the number of returned rows limit for a request	`@readOnly`
		`private void doQuery() {`
		`}`

@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.	`@remoteAction`
		`global static String getId(`
		`    String s) {`
		`}`

@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/.	`@restResource(urlMapping=`
		`'/Widget/*')`
		`global with sharing class`
		` MyResource() {`
		`}`

@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.
		`@httpGet`
		`global static MyWidget__c doGet()`
		`{`
		`}`
		
		`@httpPost`
		`global static void doPost() {`
		`}`
		
		`@httpDelete`
		`global static void doDelete() {`
		`}`

salesforce force.com

Primitive Types

Annotation	Description	Example

Blob	Binary data stored as a single object	`Blob myBlob =`
		`  Blob.valueof('idea');`

Boolean	Value that can only be assigned true, false, or null	`Boolean isWinner = true;`

Date	Particular day	`Date myDate = Date.today();`
		`Date weekStart =`
		`  myDate.toStartofWeek();`

Datetime	Particular day and time	`Datetime myDateTime =`
		`  Datetime.now();`
		`Datetime newDateTime =`
		`   myDateTime.addMonths(2);`

Decimal	Number that includes a decimal point. Decimal is an arbitrary precision number.	`Decimal myDecimal = 12.4567;`
		`Decimal divDec = myDecimal.divide`
		`(7, 2, System.RoundingMode.UP);`
		`system.assertEquals(divDec, 1.78);`

Double	64-bit number that includes a decimal point. Minimum value -2^{63}. Maximum value of 2^{63}-1	`Double d=3.14159;`
ID	18-character Force.com record identifier	`ID id='00300000003T2PGAA0';`
Integer	32-bit number that doesn't include a decimal point. Minimum value -2,147,483,648 – maximum value of 2,147,483,647	`Integer i = 1;`
Long	64-bit number that doesn't include a decimal point. Minimum value of -2^{63} – maximum value of 2^{63}-1.	`Long l = 2147483648L;`

Object	Any data type that is supported in Ape	`Object obj = 10;`
		`// Cast the object to an integer.`
		`Integer i = (Integer)obj;`
		`System.assertEquals(10, i);`
		
		`Object 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();`

Primitive Types *continued*

Annotation	Description	Example
String	Set of characters surrounded by single quotes	`String s = 'repeating memes';`
Object	Any data type that is supported i n Apex	`String s = 'repeating memes';`

Time	Particular time	`Object obj = 10;`
		`// Cast the object to an integer.`
		`Integer i = (Integer)obj;`
		`System.assertEquals(10, i);`
		
		`Object 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();`

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 `insert` and `update` triggers. The included records can be modified only in `before` triggers.) And in undelete triggers
newMap	A map of IDs to the new versions of the sObject records. (Only available in `before update`, `after insert`, and `after update` triggers.)
old	Returns a list of the old versions of the sObject records. (Only available in `update` and `delete` triggers.)
`oldMap`	A map of IDs to the old versions of the sObject records. (Only available in `update` and `delete` 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	`// Create an empty list of String`
		`List``<String> myList = new`
		`List``<String>();`
		`myList.add('hi');`
		`String x = myList.get(0);`
		`// Create list of records from a query`
		`List``<Account> accs =`
		`[SELECT Id, Name FROM Account LIMIT 1000];`

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.	**`Map`**`<String, String> MyStrings =`
		`    new Map<String, String>{`
		`        'a' => 'b', 'c' =>`
		`        'd'.toUpperCase()};`
		
		`Account`` myAcct = new Account();`
		`Map``<Integer, Account> m =`
		`  new Map<Integer, Account>();`
		`m.put(1, myAcct);`

Set	Unordered collection that doesn't contain any duplicate elements.	**`Set`**`<Integer> s = new`
		`Set``<Integer>();`
		`s.add(12);`
		`s.add(12);`
		`System.assert(s.size()==1);`

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 handleIn-
boundEmail(Messaging.inboundEmail email, Messag-
ing.InboundEnvelope env){}
```

Comparable

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

Apex Data Manipulation Language (DML) Operations

Annotation	Description	Example

insert	Adds one or more records	`Lead l = new Lead(Company='ABC',`
		`  LastName='Smith');`
		`insert`` l;`

delete	Deletes one or more records	`Account[] webAccts = [SELECT Id, Name`
		`  FROM Account WHERE Name = 'DotCom'];`
		`try {`
		`  `**`delete`**` webAccts;`
		`} catch (DmlException e) {`
		`  // Process exception here`
		`}`

merge	Merges up to three records of the same type into one of the records, deleting the others, and re-parenting any related records	`List<Account> ls = new List<Account>{`
		`  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) {`
		`}`

undelete	Restores one or more records from the Recycle Bin	`Account[] savedAccts = [SELECT Id, Name`
		`  FROM Account WHERE Name = 'Trump'`
		`  ALL ROWS];`
		`try { `**`undelete`**` savedAccts;`
		`} catch (DmlException e) {`
		`}`

update	Modifies one or more existing records	`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) {`
		`}`

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.	`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) {`
		`}`

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	cbrt
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
getRecordTypeInfos	getRecordTypeInfosByID
getSobjectType	isAccessible
getRecordTypeInfosByName	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

10042016