App Logic

Apex Code Cheat Sheet

Overview

Force.com Apex code is a strongly-typed programming language that executes on the Force.com platform. Apex is used to add business logic to applications, write database triggers, and program controllers used with Visualforce. It has a tight integration with the database and query language, web service 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 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;`
		`}`

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;`

Important Reserved Words

Keyword	Description	Example

final	Defines constants and methods that cannot be overridden	`public class myCls {`
		`  static final Integer INT_CONST;`
		`}`

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`
		`}`

if	Defines a condition, used to determine if 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 handleInboundEm`
		`ail(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`
		`  }`
		`}`

Important Reserved Words

Keyword	Description	Example

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 or property as overriding another defined as virtual in a class 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`
		`  }`
		`}`

private	Defines a class, method or variable that is only known locally, within the section of code in which it is defined. This is the default scope for all methods and variables that do not have a scope defined	`public class OuterClass {`
		`// Only visible to methods and statements within OuterClass`
		`  private static final Integer MY_INT;`
		`}`

protected	Defines a method or variable that is visible to any inner classes in the defining Apex 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 a method or variable that is only initialized once, and is associated with an (outer) class, and initialization code	`public class OuterClass {`
		`  // associated with instance`
		`  public static final Integer MY_INT;`
		
		`  // initialization code`
		`  static {`
		`  MY_INT = 10;`
		`  }`
		`}`

super	Invokes a constructor on a superclass	`public class AnotherChildClass extends InnerClass {`
		`  AnotherChildClass(String s) {`
		`  super();`
		`  // different constructor, no args`
		`  }`
		`}`

testmethod	Defines a method as a unit test	`static testmethod testFoo() {`
		`  // some test logic`
		`}`

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

Important Reserved Words

Keyword	Description	Example

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	Identifies a block of code in which an exception can occur	`try {`
		`  // Your code here`
		
		`} catch (ListException e) {`
		`  // List Exception handling code here`
		`}`

webService	Defines a static method that can be used to access external servers. Web service methods can only 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 current user. If absent, code is run under 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 current user's sharing rules`
		`}`

virtual	Defines a class or method that allows extension and overrides. You cannot 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() {}`
		
		`}`

Annotations

Annotation	Description	Example
@future	Identifies 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, other long running code } }`
@isTest	Defines classes that only contain code used for testing your application. These classes do not count against the total amount of Apex used by your organization.	`@isTest private class MyTest { // Methods for testing }`
@deprecated	Identifies 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() { }`

Primitive Types

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 newd = 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 -263 -- maximum value of 263-1	`Double d=3.14159;`
ID	18-character Force.com record identifier	`ID id='00300000003T2PGAA0';`
Integer	32-bit number that does not 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 does not include a decimal point. Minimum value of -263 -- maximum value of 263-1.	`Long l = 2147483648L;`
String	Set of characters surrounded by single quotes	`String s = 'repeating memes';`
Time	Particular time	`Time myTime = Time.newInstance(18, 30, 2, 20); Integer myMinutes = myTime.minute();`

Collection Types

List	Ordered collection of typed primitives, sObjects, objects or collections that are distinguished by their indices	`// Create an empty list of String List<String> my_list = new List<String>(); My_list.add('hi'); String x = my_list.get(0);`
		
		`// Create list of records from a query List<Account> accs = [SELECT Id, Name FROM Account LIMIT 1000];`

Collection Types

Map	Collection of key-value pairs where each unique key maps to a single value. Keys can be any primitive data type, while values can be a primitive, sObject, collection type or an object.	`Map<String, String> mys = new Map<String, String>(); Map<String, String> mys = 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 of primitives that do not contain any duplicate elements.	`Set<Integer> s = new Set<Integer>(); s.add(12); s.add(12); System.assert(s.size()==1);`

Operator Precedence

Precedence	Operators	Description
1	`{} () ++ --`	Grouping and prefix increments and decrements
2	`! -x +x (type) new`	Unary negation, type cast and object creation
3	`* /`	Multiplication and division
4	`+ -`	Addition and subtraction
5	`< <= > >= instanceof`	Greater-than and less-than comparisons, reference tests
6	`== !=`	Comparisons: equal and not-equal
7	`&&`	Logical AND
8	`||`	Logical OR
9	`= += -= *= /= &=`	Assignment operators

Trigger Context Variables

Variable	Usage
`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 is recovered from the Recycle Bin.
`new`	Returns a list of the new versions of the sObject records. (Only in `insert` and `update` triggers, and the records can only be modified in `before` 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.

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){}
```

Database.Schedulable

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

Messaging.InboundEmailHandler

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

Apex Data Manipulation Language (DML) Operations

Keyword	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[] doomedAccts = [select id, name from account where name = 'DotCom'];`
		`try {`
		`  delete doomedAccts;`
		`} 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) {`
		`  // Process exception here`
		`}`

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) {`
		`  // Process exception here`
		`}`

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) {`
		`  // jump up and down`
		`}`

upsert	Creates new records and updates existing records	`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) {`
		`  // Process exception here`
		`}`

Standard Classes and Methods (Subset)

System

```
abortJob              assert                  assertEquals
assertNotEquals       currentPageReference    currentTimeMillis
debug                 now                     process
resetPassword         runAs                   schedule
setPassword           submit                  today

System.assertEquals(b.name, 'Acme');
```

Standard Classes and Methods (Subset)

Describe

```
fields              getChildRelationships    getKeyPrefix
getLabel            getLabelPlural           getLocalName
getName             getRecordTypeInfos       getRecordTypeInfosByID
getSobjectType      isAccessible             getRecordTypeInfosByName
isCreateable        isCustom                 isCustomSetting
isDeletable         isDeprecatedAndHidden    isMergeable
isQueryable         isSearchable             isUndeletable
isUpdateable

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

DescribeFieldResult

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

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

LoggingLevel

```
ERROR     WARN     INFO     DEBUG     FINE     FINER     FINEST

System.debug(logginglevel.INFO, 'MsgTxt');
```

Limits

```
getAggregateQueries                  getLimitAggregateQueries
getCallouts                          getLimitCallouts
getChildRelationshipsDescribes       getDMLRows
getLimitChildRelationshipsDescribes  getLimitDMLRows
getDMLStatements                     getLimitDMLStatements
getEmailInvocations                  getLimitEmailInvocations
getFieldsDescribes                   getLimitFieldsDescribes
getFindSimilarCalls                  getLimitFindSimilarCalls
getFutureCalls                       getLimitFutureCalls
getHeapSize                          getLimitHeapSize
getQueries                           getLimitQueries
getPicklistDescribes                 getLimitPicklistDescribes
getQueryLocatorRows                  getLimitQueryLocatorRows
getQueryRows                         getLimitQueryRows
getRecordTypesDescribes              getLimitRecordTypesDescribes
getRunAs                             getLimitRunAs
getSavepointRollbacks                getLimitSavepointRollbacks
getSavepoints                        getLimitSavepoints
getScriptStatements                  getLimitScriptStatements
getSoslQueries                       getLimitSoslQueries

Integer myDMLLimit = Limits.getDMLStatements();
```

Math

```
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

Decimal smaller = Math.min(12.3, 156.6);
```

UserInfo

```
getDefaultCurrency    getFirstName          getLanguage
getLastName           getLocale             getName
getOrganizationId     getOrganizationName   getProfileId
getSessionId          getUserId             getUserName
getUserRoleId         getUserType           isCurrentUserLicensed
IsMultiCurrencyOrganization

String result = UserInfo.getLocale();
System.assertEquals('en_US', result);
```