# アプリケーションロジック

# Apex コード 早見表

## 概要

Apex コードは、Force.com プラットフォームで実行される強い型付けのプログラミング言語であり、アプリケーションへのビジネスロジックの追加、データベースのトリガの作成、Visualforce で使用するコントローラのプログラミングなどに使用します。Apex コードは、データベースやクエリ言語と密接に連携し、Webサービスや電子メールの処理、非同期実行、テストなどをサポートします。

## 主な予約語

| 予約語 | 説明 | コード例 |
|---|---|---|
| abstract | シグネチャのみが含まれ、本文が定義されていない抽象メソッドを含むクラスを宣言します。メソッドも定義できます。 | `public abstract class Foo {`<br>`protected void method1() { /*… */ }`<br>`abstract Integer abstractMethod();`<br>`}` |
| break | ループ全体を終了します。 | `while(reader.hasNext()) {`<br>`if (reader.getEventType() == END) {`<br>`break;`<br>`};`<br>`// process`<br>`reader.next();`<br>`}` |
| catch | 特定の例外タイプを処理できるコードブロックを識別します。 | `try {`<br>`// Your code here`<br><br>`} catch (ListException e) {`<br>`// List Exception handling code here`<br><br>`}` |
| class | クラスを定義します。 | `private class Foo {`<br>`private Integer x;`<br>`public Integer getX() { return x; }`<br>`}` |
| continue | 後続の文をスキップし、ループの次の反復処理に移動します。 | `while (checkBoolean) {`<br>`if (condition)`<br>`{continue; }`<br>`// do some work`<br>`}` |
| do | 真偽式がtureのときに繰り返し実行する do-while ループを定義します。 | `Integer count = 1;`<br>`do {`<br>`System.debug(count);`<br>`count++;`<br>`} while (count < 11);` |
| else | if-else ステートメントの else の部分を定義します。最初の評価が true 以外の場合に実行されます。 | `Integer x, sign;`<br>`if (x==0) {`<br>`sign = 0;`<br>`} else {`<br>`sign = 1;`<br>`}` |
| enum | 有限個の値のセットに対する列挙型を定義します。 | `public enum Season {WINTER, SPRING, SUMMER, FALL};`<br>`Season e = Season.WINTER;` |
| extends | 他のクラス (またはインターフェース)を拡張するクラス (またはインターフェース)を定義します。 | `public class MyException extends Exception {}`<br><br>`try {`<br>`Integer i;`<br>`if (i < 5) throw new MyException();`<br>`} catch (MyException e) {`<br>`// Your MyException handling code`<br>`}` |
| false | 真偽式に割り当てられる true 以外の値を識別します。 | `Boolean isNotTrue = false;` |

## 主な予約語

| 予約語 | 説明 | コード例 |
|---|---|---|
| final | オーバーライドを禁止する定数とメソッドを定義します。 | `public class myCls {`<br>`static final Integer INT_CONST;`<br>`}` |
| finally | 必ず実行されるコードブロックを識別します。 | `try {`<br>`// Your code here`<br>`} catch (ListException e) {`<br>`// List Exception handling code`<br>`} finally {`<br>`// will execute with or without exception`<br>`}` |
| for | ループを定義します。変数を使用した反復、リストに対する反復、クエリに対する反復という 3 種類のループを定義できます。 | `for (Integer i = 0, j = 0; i < 10; i++) { System.debug(i+1);`<br>`}`<br><br>`Integer[] myInts = new Integer[]{1, 8, 9};`<br>`for (Integer i : myInts) {`<br>`System.debug(i);`<br>`}`<br><br>`String s = 'Acme';`<br>`for (Account a : [select id, name from account`<br>`where name like :(s+'%')]) {`<br>`// Your code`<br>`}` |
| global | 同じアプリケーション内の Apex だけでなく、すべての Apex で利用できるクラス、メソッド、変数を定義します。 | `global class myClass {`<br>`webService static void`<br>`makeContact(String lastName) {`<br>`// do some work`<br>`}` |
| if | コードブロックを実行するかどうかを評価する条件を定義します。 | `Integer i = 1;`<br>`if (i > 0) {`<br>`// do something;`<br>`}` |
| implements | インターフェースを実装するクラスまたはインターフェースを宣言します。 | `global class CreateTaskEmailExample implements Messaging.InboundEmailHandler {`<br>`global Messaging.InboundEmailResult handleInboundEmail(Messaging.inboundEmail email,`<br>`Messaging.InboundEnvelope env){`<br>`// do some work, return value;`<br>`}`<br><br>`}` |
| instanceOf | オブジェクトが特定のクラスのインスタンスになっているかどうかを実行時に検証します。 | `if (reports.get(0) instanceof CustomReport) {`<br>`// Can safely cast`<br>`CustomReport c = (CustomReport) reports.get(0);`<br>`} else {`<br>`// Do something with the non-custom-report.`<br>`}` |
| interface | メソッドのシグネチャを含むデータ型を定義します。インターフェースはクラスによって実装されます。また、インターフェースは別のインターフェースを拡張できます。 | `public interface PO {`<br>`public void doWork();`<br>`}`<br>`public class MyPO implements PO {`<br>`public override doWork() {`<br>`// actual implementation`<br>`}`<br>`}` |

## 主な予約語

| 予約語 | 説明 | コード例 |
|---|---|---|
| new | オブジェクト、sObject、コレクションのインスタンスを新規に作成します。 | `Foo f = new Foo();`<br>`MyObject__c mo = new MyObject__c(name= 'hello');`<br>`List<Account> la = new List<Account>();` |
| null | 任意の変数に代入できる null 定数を識別します。 | `Boolean b = null;` |
| override | 拡張対象、実装対象のクラスで virtual として定義される他のメソッド(またはプロパティ)をオーバーライドするメソッド(またはプロパティ)を定義します。 | `public virtual class V {`<br>`  public virtual void foo() {/*does nothing*/}`<br>`}`<br><br>`public class RealV implements V {`<br>`  public override void foo() {`<br>`  // do something real`<br>`  }`<br>`}` |
| private | クラス、メソッド、変数を定義して、ローカルのみ(それが含まれるコードセクション内のみ) で認識されるようにします。スコープが指定されていないメソッドや変数では、これがデフォルトのスコープになります。 | `public class OuterClass {`<br>`// Only visible to methods and statements within OuterClass`<br>`  private static final Integer MY_INT;`<br>`}` |
| protected | メソッドや変数を定義して、それが含まれる Apex クラス内のすべての内部クラスで参照できるようにします。 | `public class Foo {`<br>`  public void quiteVisible();`<br>`  protected void lessVisible();`<br>`}` |
| public | メソッドや変数を定義して、それが含まれるアプリケーションや名前空間内のすべての Apex クラスで使用できるようにします。 | `public class Foo {`<br>`  public void quiteVisible();`<br>`  private void almostInvisible();`<br>`}` |
| return | メソッドから値を返します。 | `public Integer meaningOfLife() {`<br>`  return 42;`<br>`}` |
| static | 1 回のみ初期化され、特定のクラス (外部クラス) や初期化コードに割り当てられるメソッドや変数を定義します。 | `public class OuterClass {`<br>`  // associated with instance`<br>`  public static final Integer MY_INT;`<br><br>`  // initialization code`<br>`  static {`<br>`  MY_INT = 10;`<br>`  }`<br>`}` |
| super | スーパークラスのコンストラクタを起動します。 | `public class AnotherChildClass extends InnerClass {`<br>`  AnotherChildClass(String s) {`<br>`  super();`<br>`  // different constructor, no args`<br>`  }`<br>`}` |
| testmethod | メソッドを単体テストとして定義します。 | `static testmethod testFoo() {`<br>`  // some test logic`<br>`}` |
| this | クラスの現在のインスタンスを示します。コンストラクタチェーンで使用します。 | `public class Foo {`<br>`  public Foo(String s) { /* … */}`<br>`  public foo() { this('memes repeat'); }`<br>`}` |

## 主な予約語

| 予約語 | 説明 | コード例 |
|---|---|---|
| throw | 例外をスローし、エラーの発生を通知します。 | `public class MyException extends Exception {}`<br><br>`try {`<br>`  Integer i;`<br>`  if (i < 5) throw new MyException();`<br>`} catch (MyException e) {`<br>`  // Your MyException handling code here`<br>`}` |
| transient | 保存対象から除外するインスタンス変数を宣言します。この変数は、Visualforce コントローラおよびコントローラ拡張機能において、表示状態の要素として送信されることはありません。 | `transient integer currentValue;` |
| trigger | sObject のトリガを定義します。 | `trigger myAccountTrigger on Account (before insert, before update) {`<br>`  if (Trigger.isBefore) {`<br>`  for (Account a : Trigger.old) {`<br>`  if (a.name != 'okToDelete') {`<br>`  a.addError('You can\'t delete this record!');`<br>`  }`<br>`  }`<br>`}` |
| true | 真偽式に代入された true 値を識別します。 | `Boolean mustIterate = true;` |
| try | 例外が発生する可能性のあるコードブロックを識別します。 | `try {`<br>`  // Your code here`<br><br>`} catch (ListException e) {`<br>`  // List Exception handling code here`<br>`}` |
| webService | 外部サーバへのアクセスに使用できる静的メソッドを定義します。このメソッドは global クラスのみで定義できます。 | `global class MyWebService {`<br>`  webService static Id makeContact(String lastName, Account a) {`<br><br>`  Contact c = new Contact(lastName = 'Weissman',  AccountId = a.Id);`<br>`insert c; return c.id;`<br>`  }`<br>`}` |
| while | 特定のブール式が true である間、コードブロックを繰り返し実行します。 | `Integer count=1;`<br>`while (count < 11) {`<br>`  System.debug(count); count++;`<br>`}` |
| with sharing | 現在のユーザの共有ルールを適用します。共有ルールが存在しない場合、コードはデフォルトのシステムコンテキストにもとづいて実行されます。 | `public with sharing class sharingClass {`<br>`// Code will enforce current user's sharing rules`<br>`}` |
| without sharing | 現在のユーザの共有ルールが適用されないようにします。 | `public without sharing class noSharing {`<br>`// Code won't enforce current user's sharing rules`<br>`}` |
| virtual | 拡張とオーバーライドが可能なクラスやメソッドを定義します。そのクラスやメソッドが virtual として定義されていない状態で override キーワードを指定しても、オーバーライドは可能になりません。 | `public virtual class MyException extends Exception {`<br>`  // Exception class member variable`<br><br>`  public Double d;`<br>`  // Exception class constructor`<br>`  MyException(Double d) {`<br>`  this.d = d;`<br>`  }`<br><br>`  // Exception class method`<br>`  protected void doIt() {}`<br><br>`}` |

## アノテーション

| アノテーション | 説明 | コード例 |
|---|---|---|
| @future | 非同期で実行されているメソッドを識別します。 | `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 | アプリケーションテストに使用するコードのみを格納するクラスを定義します。これらのクラスは、通常の Apex クラスとしてはカウントされないため、組織当たりの Apex クラスの上限に関係なく定義できます。 | `@isTest private class MyTest { // Methods for testing }` |
| @deprecated | 管理パッケージに含まれるメソッド、クラス、例外、列挙型、インターフェース、変数について、後続のリリースでは参照不可となるものを識別します。 | `@deprecated public void limitedShelfLife() { }` |

## プリミティブ型

| | | |
|---|---|---|
| Blob | 単一のオブジェクトとして格納されるバイナリデータ。 | `Blob myBlob = Blob.valueof('idea');` |
| Boolean | true、false、null のいずれかが割り当てられる値。 | `Boolean isWinner = true;` |
| Date | 特定の日付。 | `Date myDate = date.today(); Date weekStart = myDate.toStartofWeek();` |
| Datetime | 特定の日付および時刻。 | `Datetime myDateTime = datetime.now(); datetime newd = myDateTime.addMonths(2);` |
| Decimal | 小数点を含む数値。任意精度数。 | `Decimal myDecimal = 12.4567; Decimal divDec = myDecimal.divide (7, 2, System.RoundingMode.UP); system.assertEquals(divDec, 1.78);` |
| Double | 小数点を含む 64 ビットの数値。最小値は $-2^{63}$、最大値は $2^{63}-1$ です。 | `Double d=3.14159;` |
| ID | 18 文字から成る Force.com レコードの識別子。 | `ID id='00300000003T2PGAA0';` |
| Integer | 小数点を含まない32ビッドの数値。最小値は -2,147,483,648、最大値は 2,147,483,647 です。 | `Integer i = 1;` |
| Long | 小数点を含まない64ビットの数値。最小値は $-2^{63}$、最大値は $2^{63}-1$ です。 | `Long l = 2147483648L;` |
| String | 単一引用符で囲まれた文字列。 | `String s = 'repeating memes';` |
| Time | 特定の時刻。 | `Time myTime = Time.newInstance(18, 30, 2, 20); Integer myMinutes = myTime.minute();` |

## コレクション型

| | | |
|---|---|---|
| List | 型付けされたプリミティブデータ、sObject、オブジェクトが一定の順序で配列されたコレクション。またはインデックスで区切られたコレクション。 | `// Create an empty list of String List<String> my_list = new List<String>(); My_list.add('hi'); String x = my_list.get(0);`<br><br>`// Create list of records from a query List<Account> accs = [SELECT Id, Name FROM Account LIMIT 1000];` |

## コレクション型

| | | |
|---|---|---|
| Map | 一意のキーが単一の値に対応付けられる、キー・値のペアから成るコレクション。キーはプリミティブデータ型となるのに対し、値はプリミティブデータ型、sObject、コレクション型、オブジェクトのいずれかとなります。 | `Map<String, String> mys = new Map<String, String>(); Map<String, String> mys = new Map<String, String>{'a' => 'b', 'c' => 'd'.toUpperCase()};`<br><br>`Account myAcct = new Account(); Map<Integer, Account> m = new Map<Integer, Account>(); m.put(1, myAcct);` |
| Set | プリミティブデータ型の順不同の配列で、重複する要素は含みません。 | `Set<Integer> s = new Set<Integer>(); s.add(12); s.add(12); System.assert(s.size()==1);` |

## 演算子の優先度

| 優先度 | 演算子 | 説明 |
|---|---|---|
| 1 | `{} () ++ --` | グループ化、前置インクリメント、前置デクリメント |
| 2 | `! -x +x (type) new` | 単項否定、型キャスト、オブジェクト作成 |
| 3 | `* /` | 乗算、除算 |
| 4 | `+ -` | 加算、減算 |
| 5 | `< <= > >= instanceof` | 大なり記号、小なり記号、参照テスト |
| 6 | `== !=` | 比較子 (等価、非等価) |
| 7 | `&&` | 論理 AND |
| 8 | `\|\|` | 論理 OR |
| 9 | `= += -= *= /= &=` | 代入演算子 |

## トリガコンテキスト変数

| 変数 | 用途 |
|---|---|
| `isExecuting` | Apex コードの現在のコンテキストがトリガのみである場合に true を返します。 |
| `isInsert` | insert 操作によりトリガが起動した場合に true を返します。 |
| `isUpdate` | update 操作によりトリガが起動した場合に true を返します。 |
| `isDelete` | delete 操作によりトリガが起動した場合に true を返します。 |
| `isBefore` | レコードの保存前にトリガが起動した場合に true を返します。 |
| `isAfter` | すべてのレコードの保存後にトリガが起動した場合に true を返します。 |
| `isUndelete` | ごみ箱からレコードが復元された後にトリガが起動した場合に true を返します。 |
| `new` | sObject レコードの新しいバージョンのリストを返します (`insert` トリガと `update` トリガのみでサポート。レコードの更新は `before insert` トリガ、`before update` トリガの場合のみ可能)。 |
| `newMap` | 新しいバージョンの sObject レコードのリストに ID を対応付けます (`before update` トリガ、`after insert` トリガ、`after update` トリガのみでサポート)。 |
| `old` | sObject レコードの古いバージョンのリストを返します (`update` トリガおよび `delete` トリガのみでサポート)。 |
| `oldMap` | 古いバージョンの sObject レコードのリストに ID を対応付けます (`update` トリガおよび `delete` トリガのみでサポート)。 |
| `size` | トリガ呼び出しに含まれるレコードの総計 (古いバージョン、新しいバージョンの両方を含む)。 |

## 標準のインターフェース (サブセット)

**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 のデータ操作言語 (DML)

| 予約語 | 説明 | コード例 |
|---|---|---|
| insert | 1 つまたは複数のレコードを追加します。 | `Lead l = new Lead(company='ABC', lastname='Smith');`<br>`insert l;` |
| delete | 1 つまたは複数のレコードを削除します。 | `Account[] doomedAccts = [select id, name from account where name = 'DotCom'];`<br>`try {`<br>`  delete doomedAccts;`<br>`} catch (DmlException e) {`<br>`  // Process exception here`<br>`}` |
| merge | 同じタイプの複数のレコードを 1 つにマージします (一度に 3 つまでマージ可能)。マージでは、不要なレコードは削除され、関連付けられたレコードがある場合には親子関係が再設定されます。 | `List<Account> ls = new List<Account>{new Account(name='Acme Inc.'),new Account(name='Acme')};`<br>`insert ls;`<br>`Account masterAcct = [select id, name from account where name = 'Acme Inc.' limit 1];`<br>`Account mergeAcct = [select id, name from account where name = 'Acme' limit 1];`<br>`try {merge masterAcct mergeAcct;`<br>`} catch (DmlException e) {`<br>`  // Process exception here`<br>`}` |
| undelete | 1 つまたは複数のレコードをごみ箱から復元します。 | `Account[] savedAccts = [select id, name from account where name = 'Trump' ALL ROWS];`<br>`try {undelete savedAccts;`<br>`} catch (DmlException e) {`<br>`  // Process exception here`<br>`}` |
| update | 1 つまたは複数の既存レコードを更新します。 | `Account a = new Account(name='Acme2');`<br>`insert(a);`<br>`Account myAcct = [select id, name, billingcity from account where name = 'Acme2' limit 1];`<br>`myAcct.billingcity = 'San Francisco';`<br>`try {`<br>`  update myAcct;`<br>`} catch (DmlException e) {`<br>`  // jump up and down`<br>`}` |
| upsert | 新規レコードの作成と既存レコードの更新を実行します。 | `Account[] acctsList = [select id, name, billingcity from account where billingcity = 'Bombay'];`<br>`for (Account a : acctsList) {a.billingcity = 'Mumbai';`<br>`}Account newAcct = new Account(name = 'Acme', billingcity = 'San Francisco');`<br>`acctsList.add(newAcct);`<br>`try {upsert acctsList;`<br>`} catch (DmlException e) {`<br>`  // Process exception here`<br>`}` |

## 標準のクラス、メソッド (サブセット)

### System

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

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

## 標準のクラス、メソッド (サブセット)

### 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            isCaseSensitiveis
Createable            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);
```