



salesforce: Summer '13

# Force.com Apex コード開発者ガイド



最終更新日: 2013/6/5

© Copyright 2000–2013 salesforce.com, inc. All rights reserved. Salesforce.com およびその他の名称や商標は、salesforce.com, inc. の登録商標です。本ドキュメントに記載されたその他の商標は、各社に所有権があります。



# 目次

<b>第 1 章: Apex の概要.....</b>	<b>1</b>
Apex とは?.....	2
Apex はどのように機能しますか?.....	4
Apex 開発プロセスとは?.....	4
開発者組織または Sandbox 組織の使用.....	5
Apex に関する説明.....	7
Apex の記述.....	8
テストの記述.....	9
Sandbox 組織への Apex のリリース.....	10
Salesforce 本番組織への Apex のリリース.....	10
Force.com AppExchange アプリケーションへの Apex コードの追加.....	11
Apex を使用する必要がある状況は?.....	11
Apex の制限事項.....	12
最新情報.....	13
Apex クイックスタート.....	13
ドキュメント表記規則.....	13
Apex の基本概念について.....	14
最初の Apex コードおよびトリガの作成.....	20
カスタムオブジェクトの作成.....	20
Apex クラスの追加.....	21
Apex トリガの追加.....	22
テストクラスの追加.....	24
本番組織へのコンポーネントのリリース.....	27
<b>第 2 章: 言語構造.....</b>	<b>29</b>
データ型.....	30
プリミティブデータ型.....	30
sObject 型.....	33
sObject 項目へのアクセス.....	35
リレーションを使用した sObject 項目へのアクセス.....	37
sObjects と項目の検証 .....	39
Collections.....	39
List.....	39
Set.....	47
対応付け.....	49
パラメータ化された型.....	52
対応付けのキーとセットでのカスタムデータ型の使用.....	52
コレクションの繰り返し処理.....	56

enum.....	56
変換の規則について.....	58
変数.....	60
大文字と小文字の区別.....	60
定数.....	61
式.....	62
式について.....	62
式の演算子について.....	63
演算子の優先順位について.....	70
sObject 式およびリスト式の拡張.....	71
コメントの使用.....	71
代入ステートメント.....	72
条件 (If-Else) ステートメント.....	73
ループ.....	74
Do-While ループ.....	75
While ループ.....	75
For ループ.....	76
従来の For ループ.....	77
リスト反復またはセット反復の For ループ.....	77
SOQL For ループ.....	78
Exception ステートメント.....	81
throw ステートメント.....	81
Try-Catch-Finally ステートメント.....	81
SOQL および SOSL クエリ.....	83
SOQL および SOSL クエリ結果の処理.....	84
SOQL 集計関数の使用.....	85
非常に大きい SOQL クエリの処理.....	86
1つのレコードを返す SOQL クエリの使用.....	89
null 値検索の回避によるパフォーマンスの改善.....	90
外部キーおよび親 - 子リレーションの SOQL クエリについて.....	91
SOQL クエリの多態的なリレーションの処理.....	92
SOQL クエリおよび SOSL クエリでの Apex 変数の使用.....	93
SOQL ステートメントを使用したすべてのレコードのクエリ.....	96
Apex でのデータの使用.....	96
DML ステートメントと Database クラスメソッド.....	98
アトミックトランザクションとしての DML 操作.....	99
DML の仕組み.....	99
DML 操作.....	101
レコードの挿入と更新.....	101
レコードの更新/挿入.....	105
レコードのマージ.....	109

レコードの削除.....	111
削除したレコードの復元.....	112
取引の開始.....	113
DML 例外とエラー処理.....	115
例外処理.....	115
Database クラスメソッドの結果オブジェクト.....	115
返されるデータベースエラー.....	116
DML の詳細.....	117
DML オプションの設定.....	117
トランザクションの制御.....	120
DML 操作で同時に使用できない sObject.....	122
DML 操作をサポートしない sObject.....	126
一括 DML 例外処理.....	127
Apex のデータについて知っておくべきこと.....	128
レコードのロック.....	129
ロックステートメント.....	129
SOQL For ループのロック.....	130
デッドロックの回避.....	130
<b>第 3 章: Apex の呼び出し.....</b>	<b>131</b>
トリガ.....	132
一括トリガ.....	133
トリガ構文.....	133
トリガコンテキスト変数.....	134
コンテキスト変数の考慮事項.....	137
一般的な一括トリガイディオム.....	138
一括トリガでの対応付けおよびセットの使用.....	138
一括トリガのレコードとクエリ結果の関連付け.....	139
トリガを使用した、一意の項目を持つレコードの挿入または更新.....	140
トリガの定義.....	140
トリガと Merge ステートメント.....	142
トリガと復元レコード.....	143
トリガと実行の順序.....	144
トリガを呼び出さない操作.....	146
トリガのエンティティおよび項目の考慮事項.....	147
トリガの例外.....	149
トリガと一括要求に関するベストプラクティス.....	149
Apex スケジューラ.....	151
匿名ブロック.....	158
Apex in AJAX.....	160
<b>第 4 章: クラス、オブジェクトおよびインターフェース.....</b>	<b>162</b>

クラスを理解する.....	163
Apex クラスの定義.....	163
拡張クラスの例.....	165
クラス変数の宣言.....	170
クラスメソッドの定義.....	171
コンストラクタの使用.....	174
アクセス修飾子.....	176
静的およびインスタンスマソッド.....	178
静的メソッドと変数の使用.....	178
インスタンスマソッドと変数の使用.....	180
初期化コードの使用.....	181
Apex プロパティ.....	182
インターフェースおよび拡張クラス.....	186
カスタムイテレータ.....	187
キーワード.....	190
final キーワードの使用.....	191
instanceof キーワードの使用.....	191
super キーワードの使用.....	192
this キーワードの使用.....	193
transient キーワードの使用.....	194
with sharing または without sharing キーワードの使用.....	196
アノテーション.....	199
Deprecated アノテーション.....	200
Future アノテーション.....	200
IsTest アノテーション.....	202
ReadOnly アノテーション.....	207
RemoteAction アノテーション.....	207
TestVisible アノテーション.....	208
Apex REST アノテーション.....	211
RestResource アノテーション.....	212
HttpDelete アノテーション.....	212
HttpGet アノテーション.....	212
HttpPatch アノテーション.....	213
HttpPost アノテーション.....	213
HttpPut アノテーション.....	213
クラスとキャスト.....	213
クラスとコレクション.....	215
コレクションキャスト.....	216
Apex クラスと Java クラスの違い.....	216
クラス定義の作成.....	217
名前付け規則.....	219

名前のシャドウイング.....	219
クラスのセキュリティ.....	220
オブジェクト権限と項目権限の適用.....	221
名前空間プレフィックス.....	222
System 名前空間の使用.....	223
名前空間、クラス、変数名の優先順位.....	224
型の解決と型のシステム名前空間.....	225
バージョン設定.....	226
クラスおよびトリガへの Salesforce API バージョン設定.....	226
Apex クラスとトリガのパッケージバージョンの設定.....	228
<b>第 5 章: Apex のテスト.....</b>	<b>229</b>
Apex のテストについて.....	230
Apex テストを行う理由.....	230
Apex のテスト内容.....	230
Apex の単体テスト.....	231
単体テストの組織データとテストデータの分離.....	233
runAs メソッドの使用.....	234
Limits、startTest、および stopTest の使用.....	236
SOSL クエリの単体テストへの追加.....	237
単体テストメソッドの実行.....	238
ベストプラクティスのテスト.....	240
テストの例.....	242
<b>第 6 章: 動的 Apex.....</b>	<b>250</b>
Apex Describe Information について.....	251
動的 SOQL.....	265
動的 SOSL.....	266
動的 DML.....	267
<b>第 7 章: Apex の一括処理.....</b>	<b>272</b>
Apex の一括処理の使用.....	273
Apex による共有管理について.....	288
共有の理解.....	288
Apex を使用したレコードの共有.....	291
Apex による共有管理の再適用.....	299
<b>第 8 章: Chatter in Apex の使用.....</b>	<b>307</b>
ConnectApi 入力および出力クラスの使用.....	308
コミュニティでの ConnectApi データへのアクセス.....	308
ConnectApi クラスの制限について.....	309
ConnectApi オブジェクトの逐次化と並列化.....	310
ConnectApi バージョン設定と同等性チェック.....	311

ConnectApi オブジェクトのキャスト.....	312
ワイルドカードの使用.....	313
ConnectApi クラスのテスト.....	315
ConnectApi クラスとその他の Apex クラスの違い.....	317
<b>第 9 章: Apex のデバッグ.....</b>	<b>319</b>
デバッグログについて.....	320
開発者コンソールのログの操作.....	325
Apex API コールのデバッグ.....	335
検出されなかった例外の処理.....	337
実行ガバナと制限について.....	337
ガバナ制限のメール警告の使用.....	344
<b>第 10 章: 管理パッケージでの Apex の開発.....</b>	<b>345</b>
パッケージバージョン.....	346
Apex の廃止.....	347
パッケージバージョンの動作.....	347
Apex コードの動作のバージョン設定.....	347
バージョン設定されていない Apex コードの項目.....	348
パッケージバージョンの動作のテスト.....	349
<b>第 11 章: Apex メソッドを SOAP Web サービスとして公開.....</b>	<b>353</b>
WebService メソッド.....	354
WebService メソッドによるデータの公開.....	354
WebService キーワードの使用に関する考慮事項.....	355
Web サービスマソッドのオーバーロード.....	357
<b>第 12 章: Apex クラスを REST Web サービスとして公開.....</b>	<b>358</b>
Apex REST の概要.....	359
Apex REST アノテーション.....	359
Apex REST のメソッド.....	359
Apex REST Web サービスマソッドを使用したデータの公開.....	366
Apex REST のコードサンプル.....	367
Apex REST の基本コードサンプル.....	367
RestRequest を使用した Apex REST のコードサンプル.....	369
<b>第 13 章: Apex を使用したコールアウトの呼び出し.....</b>	<b>372</b>
リモートサイトの設定の追加.....	373
SOAP サービス: WSDL ドキュメントからのクラスの定義.....	373
外部サービスの呼び出し.....	374
HTTP ヘッダーのサポート.....	375
サポートされる WSDL の機能.....	376
生成されるコードについて.....	379

Web サービスコールアウトのテスト.....	384
DML 操作と擬似コールアウトの実行.....	387
WSDL 使用についての考慮事項.....	389
ヘッダーの対応付け.....	389
ランタイムイベントについて.....	389
変数名でサポートされていない文字について.....	390
WSDL ファイルから生成したクラスのデバッグ.....	390
HTTP コールアウトの呼び出し.....	390
証明書の使用.....	391
証明書の生成.....	391
SOAP サービスでの証明書の使用.....	392
HTTP 要求での証明書の使用.....	393
コールアウトの制限事項.....	394
<b>第 14 章: リファレンス.....</b>	<b>395</b>
DML 操作.....	396
DML ステートメント.....	396
Insert ステートメント.....	396
Update ステートメント.....	397
Upsert ステートメント.....	398
Delete ステートメント.....	399
Undelete ステートメント.....	400
Merge ステートメント.....	400
一括 DML 例外処理.....	401
データベースメソッド.....	402
Database クラス.....	402
Database.LeadConvert クラス.....	417
Database.LeadConvertResult クラス.....	420
Database.SaveResult クラス.....	421
Database.UpsertResult クラス.....	422
Database.DeleteResult クラス.....	423
Database.UndeleteResult クラス.....	425
Database.EmptyRecycleBinResult クラス.....	425
Database.Error クラス.....	426
Database.QueryLocator クラス.....	426
Database.QueryLocatorIterator クラス.....	427
Database.DMLOptions プロパティ.....	428
Apex 標準クラスおよび標準メソッド.....	433
Apex Primitive メソッド.....	434
Blob メソッド.....	434
Boolean メソッド.....	435
Date メソッド.....	436

dateTime メソッド.....	439
Decimal メソッド.....	446
Double メソッド.....	452
ID メソッド.....	454
Integer メソッド.....	460
Long メソッド.....	461
String メソッド.....	462
Time メソッド.....	494
Apex Collection メソッド.....	495
List メソッド.....	496
Map メソッド.....	505
Set メソッド.....	512
Enum メソッド.....	517
Apex sObject メソッド.....	518
Schema メソッド.....	518
sObject メソッド.....	523
sObject Describe Result メソッド.....	534
Describe Field Result メソッド.....	539
Schema.FieldSet メソッド.....	548
カスタム設定メソッド.....	552
Apex System メソッド.....	563
ApexPages メソッド.....	564
Approval メソッド.....	564
JSON サポート.....	566
Limits メソッド.....	591
Math メソッド.....	595
MultiStaticResourceCalloutMock メソッド.....	600
Apex REST.....	601
Search メソッド.....	608
StaticResourceCalloutMock メソッド.....	608
System メソッド.....	609
Test メソッド.....	624
TimeZone メソッド.....	631
型メソッド.....	633
URL メソッド.....	638
UserInfo メソッド.....	642
Version メソッド.....	644
例外メソッドの使用.....	646
Apex クラス.....	650
Apex 承認プロセスクラス.....	650
Apex 承認プロセスの例.....	651

ProcessRequest クラス.....	653
ProcessResult クラス.....	653
ProcessSubmitRequest クラス.....	654
ProcessWorkitemRequest クラス.....	655
Apex Community (Zone) クラス.....	656
Answers クラス.....	656
Ideas クラス.....	658
Apex メールクラス.....	662
送信メール.....	662
受信メール.....	676
Apex のサポートクラス.....	683
BusinessHours クラス.....	683
Cases クラス.....	685
Chatter in Apex クラス.....	686
ConnectApi.Chatter クラス.....	687
ConnectApi.ChatterFavorites クラス.....	688
ConnectApi.ChatterFeeds クラス.....	693
ConnectApi.ChatterGroups クラス.....	726
ConnectApi.ChatterUsers クラス.....	734
ConnectApi.Communities クラス.....	742
ConnectApi.Organization クラス.....	743
ConnectApi.Records クラス.....	743
ConnectApi 入力クラス.....	744
ConnectApi 出力クラス.....	749
ConnectApi Enum.....	780
ConnectApi 例外.....	785
例外クラス.....	785
例外の作成.....	786
例外変数の使用.....	788
Flow.Interview クラス.....	788
HTTP (RESTful) サービスクラス.....	790
HTTP クラス.....	790
Crypto クラス.....	807
EncodingUtil クラス.....	815
ナレッジ管理の公開サービスクラス.....	816
Network クラス.....	821
Pattern および Matcher クラス.....	822
Pattern と Matcher の使用.....	822
リージョンの使用.....	823
マッチ処理の使用.....	823
境界の使用.....	824

キャプチャグループについて.....	824
Pattern と Matcher の例.....	825
Pattern メソッド.....	827
Matcher メソッド.....	829
Publisher Action クラス.....	835
QuickAction クラス.....	836
QuickAction.QuickActionRequest クラス.....	838
QuickAction.QuickActionResult クラス.....	839
Site クラス.....	839
Cookie クラス.....	848
Visualforce クラス.....	851
Action クラス.....	851
動的コンポーネントメソッドとプロパティ.....	853
IdeaStandardController クラス.....	854
IdeaStandardSetController クラス.....	858
KnowledgeArticleVersionStandardController クラス.....	863
Message クラス.....	868
PageReference クラス.....	869
SelectOption クラス.....	876
StandardController クラス.....	879
StandardSetController クラス.....	882
XML クラス.....	885
XmlStream クラス.....	885
DOM クラス.....	895
Apex インターフェース.....	903
Auth.RegistrationHandler インターフェース.....	904
Comparable インターフェース.....	909
HttpCalloutMock インターフェース.....	912
InstallHandler インターフェース.....	912
Support.EmailTemplateSelector インターフェース.....	915
Site.UrlRewriter インターフェース.....	917
Process.Plugin インターフェースの使用.....	926
Process.Plugin インターフェース.....	927
Process.PluginRequest クラス.....	929
Process.PluginResult クラス.....	930
Process.PluginDescribeResult クラス.....	931
Process.Plugin データ型変換.....	934
リードの変換用の Process.Plugin の実装のサンプル.....	934
UninstallHandler インターフェース.....	946
WebServiceMock インターフェース.....	948
第 15 章: Apex のリリース.....	950

変更セットを使用した Apex のリリース.....	951
Apex をリリースするための Force.com IDE の使用.....	951
Force.com Migration Tool の使用.....	952
deploy について.....	954
retrieveCode について.....	955
runTests() について.....	957
SOAP API を使用した Apex のリリース.....	957
<b>付録.....</b>	<b>958</b>
<b>付録 A: 納入先請求書の例.....</b>	<b>958</b>
納入先請求書の例の模擬体験.....	958
納入先請求書のコード例.....	961
<b>付録 B: 予約キーワード.....</b>	<b>976</b>
<b>付録 C: Apex および Visualforce 開発のセキュリティのヒント.....</b>	<b>978</b>
クロスサイトスクリプト (XSS).....	978
Visualforce ページのエスケープされない出力と式.....	980
クロスサイトリクエストフォージェリ (CSRF).....	982
SOQL インジェクション.....	983
データアクセスコントロール.....	986
<b>付録 D: Apex の SOAP API および SOAP ヘッダー.....</b>	<b>988</b>
compileAndTest().....	989
CompileAndTestRequest.....	991
CompileAndTestResult.....	991
compileClasses().....	994
compileTriggers().....	995
executeanonymous().....	996
ExecuteAnonymousResult.....	997
runTests().....	997
RunTestsRequest.....	1000
RunTestsResult.....	1000
DebuggingHeader.....	1004
PackageVersionHeader.....	1005
<b>用語集.....</b>	<b>1008</b>



# 第1章

## Apex の概要

### トピック:

- [Apex とは?](#)
- [最新情報](#)
- [Apex クイックスタート](#)

salesforce.com は、伝統的なクライアント-サーバベースの企業アプリケーションをオンデマンド、マルチテナント方式の Web 環境 (Force.com プラットフォーム) へと移すことによって、ビジネスの方法を変えてきました。上記の環境により、組織が Salesforce Automation や Service & Support などのアプリケーションの実行とカスタマイズを行い、特定のビジネスニーズに基づいて新しいカスタムアプリケーションを構築できるようになりました。

Salesforce ユーザインターフェースでは、新規項目、オブジェクト、ワークフロー、および承認プロセスを定義する機能などの多くのカスタマイズオプションを使用できますが、開発者は、SOAP API を使用して、クライアント側のプログラムから `Apex` 、`Visualforce` 、`JavaScript` など のデータ操作コマンドを発行することもできます。

これらのクライアント側プログラムは通常 Java、JavaScript、.NET、またはその他のプログラミング言語で作成され、このプログラムによって組織はより柔軟にカスタマイズを行うことができます。ただし、これらのクライアント側プログラムの制御ロジックは Force.com プラットフォームサーバ上にないため、次のような制限があります。

- 一般的なビジネストランザクションを完了させるために salesforce.com サイトへの複数回の呼び出しを必要とするパフォーマンスコスト
- Java や .NET などのサーバコードを安全で安定した環境にホストするためのコストと複雑さ

これらの問題に対処し、開発者がオンデマンドアプリケーションを作成する方法を大幅に改革するために、salesforce.com は、次世代のビジネスアプリケーションの構築に関心を持つ開発者のための、初のマルチテナント、オンデマンドプログラミング言語である Force.com Apex コードを導入します。

- [Apex とは?](#)— Apex の用途、開発プロセス、および制限事項についての詳細
- [Apex リリースの新機能](#)
- [Apex クイックスタート](#)— コードを徹底的に調べ、初めて Apex クラスおよびトリガを作成する

## Apex とは?

Apex は、開発者が Force.com プラットフォームサーバでフローとトランザクションの制御ステートメントを Force.com API へのコールと組み合わせて実行できるようにした、強く型付けされたオブジェクト指向のプログラミング言語です。Java に似た構文を使い、データベースのストアドプロシージャのように動作する Apex を使用して、開発者は、ボタンクリック、関連レコードの更新、および Visualforce ページなどのほとんどのシステムでのイベントに対しビジネスロジックを追加できます。Apex コードは、Web サービス要求、およびオブジェクトのトリガから開始できます。



図 1: ほとんどのシステムの行動に Apex を追加できます。

Apex は、言語として次の特徴があります。

統合されている

Apex では、次の Force.com プラットフォームの共通のイディオムが組み込まれています。

- 、 、 および など、組み込み  
コール 处理を含むデータ操作言語 (DML)
- sObject レコードのリストを返す、インラインの Salesforce Object Query Language (SOQL) と Salesforce Object Search Language (SOSL) のクエリ
- 複数のレコードの一括処理を可能にするループ

- レコード更新の競合を回避するロック構文
  - 保存された Apex メソッドから構築できる、カスタムの公開 Force.com API コール
  - Apex が参照するカスタムオブジェクトまたはカスタム項目を編集または削除しようとすると発行される警告とエラー

使いやすい

Apex は、変数および式の構文、ブロックおよび条件ステートメントの構文、ループ構文、オブジェクトおよび配列の表記など、よく知られた Java のイディオムに基づいています。Apex が新しい要素を導入している場合には、理解しやすく、Force.com プラットフォームを効率的に使用できるようにする構文および意味を使用します。その結果、Apex は、簡潔で記述しやすいコードを作成します。

データ指向

Apex は、複数のクエリと DML ステートメントを Force.com プラットフォームサーバ上の 1 つの作業にまとめるように設計されています。開発者はデータベースのストアドプロシージャを使用して、複数のトランザクションステートメントをデータベースサーバにまとめます。他のデータベースのストアドプロシージャと同じく、Apex は 30 ユーティリティ 485-0259+ 及び 36 の要素の実行はサポートされません。0 0 103 0 1 0731G003wprce

正確である

Apex は、オブジェクト名や項目名などのスキーマオブジェクトを直接参照する、強力に定型化された言語です。参照が無効である場合は、コンパイル時にすぐにエラーが発生します。アクティブな Apex コードが要求しているときに削除されないように、メタデータのすべてのカスタム項目、オブジェクト、クラス連動関係を保存します。

ホストされている

Apex は、すべて Force.com の 請り喰 、 鮎湖 、 鳗鰐翠磧 に 姦遂 まヰ襯 - トナバ裸臣 言

## バージョン設定されている

Apex コードを異なるバージョンの Force.com API に保存できます。これにより、動作を維持できます。

Apex は Unlimited Edition、Developer Edition、Enterprise Edition、および Database.com に含まれています。

## Apex はどのように機能しますか?

すべての Apex は、次のアーキテクチャの図で示すように、Force.com プラットフォームでは完全にオンデマンドで実行されます。

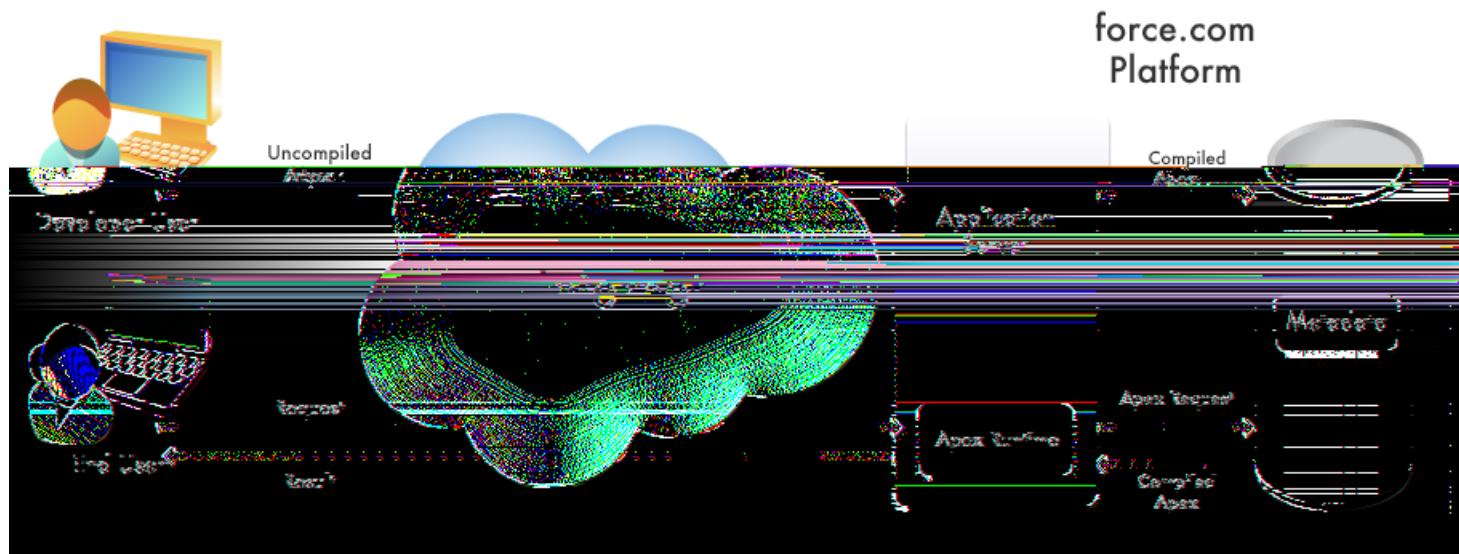


図 2: Apex は、すべて Force.com プラットフォームでコンパイル、保存、および実行されます。

開発者が Apex コードを記述してプラットフォームに保存するときに、プラットフォームのアプリケーションサーバはまず、Apex ランタイムインタプリタによって解釈される抽象的な命令セットにコードをコンパイルし、その後それらの命令をメタデータとして保存します。

エンドユーザがボタンをクリックするか Visualforce ページにアクセスするなどして Apex の実行をトリガすると、プラットフォームのアプリケーションサーバはコンパイルされた命令をメタデータから取得し、ランタイムインタプリタを通して送信してから、結果を返します。エンドユーザは、標準プラットフォーム要求との実行時間の違いに気付くことはありません。

## Apex 開発プロセスとは?

Apex の開発には、次の手順をお勧めします。

1. [Developer Edition アカウントを取得します。](#)
2. [Apex についての詳細を確認します。](#)
3. [Apex コードを記述します。](#)
4. Apex を記述するときに、[テストも記述する必要があります。](#)

5. 必要に応じて Apex を Sandbox 組織にリリースし、最終単体テストを行います。
6. Salesforce 本番組織に Apex をリリースします。

コードを作成およびテストしたら、Apex コードのリリースだけでなく、Force.com AppExchange アプリケーションパッケージにクラスおよびトリガを追加することもできるようになります。

## 開発者組織または Sandbox 組織の使用

Apex を実行可能な組織は次の 3 つのタイプの組織です。

- ・ **開発者組織:** Developer Edition アカウントで作成された組織。
- ・ **本番組織:** データにアクセスするライブユーザを持っている組織。
- ・ **Sandbox 組織:** 本番組織上に作成された組織であり、本番組織のコピー。



メモ: Apex トリガは Salesforce の Trial Edition で使用できますが、他のエディションに変換すると無効になります。新しくサインアップした組織に Apex が含まれる場合、いずれかのリリースメソッドを使用してコードを組織にリリースする必要があります。

Salesforce 本番組織では Apex を開発することはできません。実際にユーザが利用中のシステムで開発を行う場合、データが不安定になったり、アプリケーションが破損したりする可能性があります。Sandbox または Developer Edition 組織上で、すべての開発作業を行うことを推奨します。

まだ開発者コミュニティのメンバーでない場合、  
Edition アカウントのサインアップの説明に従ってください。Developer Edition アカウントによって、Developer Edition 組織に自由にアクセスできるようになります。Enterprise Edition または Unlimited Edition の組織、および Apex を作成するための Sandbox 組織がすでにある場合でも、開発者コミュニティのリソースを参照することを強くお勧めします。



メモ: Salesforce の本番組織では、Salesforce ユーザインターフェースを使用して Apex に変更を加えることはできません。

## Sandbox 組織の作成

Sandbox 組織を作成または更新する手順は、次のとおりです。

1. [設定] で、[Sandbox] をクリックします。
2. [新規 Sandbox] をクリックします。
3. Sandbox の名前と説明を入力します。Sandbox を作成または更新するときのみ、名前を変更できます。



ヒント: 次の条件を満たす名前を選択することをお勧めします。

- ・ 「QA」など、この Sandbox の目的を反映している。
- ・ 文字数の少ない名前。これは、Salesforce が Sandbox 環境のユーザレコードのユーザ名とアドレスに Sandbox 名を自動的に付加するためです。文字数の少ない名前であれば、Sandbox へのログイン時の入力も容易です。

4. 使用する Sandbox の種別を選択します。

## 開発者 Sandbox

開発者 Sandbox は、1人の開発者によるコーディングとテストを目的とした、特殊な設定用 Sandbox です。複数のユーザが1つの開発者 Sandbox にログインできますが、その主な目的は、共有可能な状態になるまで、開発中に行われた変更を分離できる環境を提供することにあります。設定のみの Sandbox の場合と同様に、開発者 Sandbox ではすべてのアプリケーションと設定情報が Sandbox にコピーされます。開発者 Sandbox では、テストデータまたはサンプルデータは 10 MB に制限されていますが、多くの開発およびテストのタスクにはこれで十分です。開発者 Sandbox は、1日に1回更新できます。

## 設定のみの Sandbox

設定のみの Sandbox は、[設定] にある本番組織のレポート、ダッシュボード、価格表、商品、アプリケーション、およびカスタマイズ設定をすべてコピーしますが、組織の標準およびカスタムオブジェクトトレコード、ドキュメント、および添付ファイルはすべて除外します。設定のみの Sandbox を作成することによって、Sandbox の作成または更新にかかる時間を数時間から数分にまで短縮できますが、含めることができるデータは 500 MB までです。設定のみの Sandbox は、1日に1回更新できます。

## フル Sandbox

フル Sandbox は、本番組織全体と、標準およびカスタムのオブジェクトトレコード、ドキュメント、添付ファイルなど、すべてのデータをコピーします。フル Sandbox は、29日ごとに更新できます。



メモ: Sandbox オプションが表示されない場合や、Sandbox の追加ライセンスが必要な場合は、salesforce.com に問い合わせて、組織の Sandbox を注文してください。

購入した Sandbox の数を減らしたにも関わらず、特定の種別の Sandbox を許可された数を超えて所有している場合は、所有する Sandbox の数と Sandbox の購入数が一致するように変更することを求められます。たとえば、2つのフル Sandbox を所有しているが1つしか購入していない場合、所有する1つのフル Sandbox はフル Sandbox として更新することはできません。代わりに、1つのフル Sandbox を選択し、使用可能な種別に応じて、設定のみまたは開発者 Sandbox など、より小さい Sandbox に変換する必要があります。

### 5. Sandbox に含めるデータを選択します (フル Sandbox の場合に、この選択を行います)。

フル Sandbox の場合、コピーするオブジェクト履歴、ケース履歴、および商談履歴の量と、Chatter データをコピーするかどうかを選択します。オブジェクト履歴はカスタムオブジェクトとほとんどの標準オブジェクトの項目履歴管理であり、ケース履歴および商談履歴は、ケースおよび商談について同じ機能を提供します。履歴は 0 ~ 180 日分のコピーが可能で、30 日単位で増加できます。デフォルト値は 0 日です。Chatter データには、フィード、メッセージ、および検出トピックが含まれます。コピーするデータの量を減らすと、Sandbox のコピー時間を大幅に短縮できます。

[テンプレートベース] データを含めるように選択できます。このオプションを使用するには、Sandbox テンプレートを事前に作成しておく必要があります。作成したテンプレートのリストから、テンプレートを選択できます。詳細は、「Sandbox テンプレートの作成」を参照してください。

### 6. [作成] をクリックします。

組織のサイズによって、このプロセスには数分、数時間、あるいは数日を要する可能性があります。



ヒント: Sandbox のコピー処理中は、本番組織への変更を制限するようにしてください。

## Apex に関する説明

開発者アカウントを作成すると、Apexについて学ぶための次のような多くのリソースを使用できるようになります。

### Force.com Workbook: クラウドでアプリケーション開発を始めよう

#### 初級プログラマ

さまざまな Force.com プラットフォーム機能を紹介する 30 分間の 10 個のチュートリアルのセットです。Force.com ワークブックチュートリアルは、ごく簡単な在庫管理システムの構築に焦点を当てます。アプリケーションの開発をボトムアップ方式で始めることができます。つまり、商品を追跡するためのデータベースモデルを最初に構築します。次に、ビジネスロジックを追加します。これらのビジネスロジックには、十分な在庫があることを確認する入力規則、商品が売れた場合に在庫を更新するワークフロー、大量の請求書の値のメール通知を送信する承認、未処理の請求書の価格を更新するトリガルールなどがあります。データベースおよびビジネスロジックが完了したら、製品在庫をスタッフに表示するユーザインターフェースおよび製品カタログを表示する公開 Web サイトを作成し、単純な店舗ページを作成します。オフラインでも利用できるアプリケーションの開発は、Adobe Flash Builder for Force.com を利用した最後のチュートリアルを参照してください。

Force.com ワークブック: [HTML](#) | [PDF](#)

### Apex ワークブック

#### 初級プログラマ

Apex ワークブックは、一連のチュートリアルを通じて Apex プログラム言語を紹介します。Apex の基本、および Force.com プラットフォームでトリガ、単体テスト、スケジュール済み Apex、Apex 一括処理、REST Web サービス、Visualforce コントローラを使用したカスタムビジネスロジックを追加する方法を学習できます。

Apex ワークブック: [HTML](#) | [PDF](#)

### Developer Force Apex ページ

#### 初級および上級プログラマ

Developer Force の Apex ページには、Apex プログラミング言語に関する記事を含むいくつかのリソースへのリンクがあります。これらのリソースには、Apex の簡単な概要と Apex 開発のベストプラクティスが記載されています。

### Force.com Cookbook

#### 初級および上級プログラマ

このコラボレーションサイトでは、Web サービス API の使用、Apex コードの開発、Visualforce ページの作成に関する多くの手順を提供しています。『Force.com Cookbook』は開発者が一般的な Force.com プログラミングの手法およびベストプラクティスに精通するよう支援します。では、既存の手順を参照したり、コメントしたり、自分の手順を提出したりできます。

### 開発ライフサイクル: Force.com プラットフォームでのエンタープライズ開発

#### アーキテクトおよび上級プログラマ

アーキテクト、システム管理者、開発者、またはマネージャであるかを問わず、*Development Life Cycle Guide* では、Force.com プラットフォームでの複雑なアプリケーションの開発とリリースを行う準備を整えることができます。

## トレーニングコース

Salesforce トレーニングや認定制度をご利用できます。トレーニング/認定制度のサイトを参照してください。

## 本書(『Force.com Apex コード開発者ガイド』)

初級プログラマは次を参照してください。

- [Apex の概要](#)。特に次を参照してください。
  - ◊ [ドキュメント表記規則](#)
  - ◊ [基本概念](#)
  - ◊ [クイックスタートチュートリアル](#)
- [クラス、オブジェクトおよびインターフェース](#)
- [Apex のテスト](#)
- [実行ガバナと制限について](#)

上記だけでなく、上級プログラマは次も参照してください。

- [トリガと一括要求に関するベストプラクティス](#)
- [高度な Apex プログラミングの例](#)
- [Apex Describe Information について](#)
- [非同期実行\(アノテーション\)](#)
- [Apex の一括処理および Apex スケジューラ](#)

## Apex の記述

次のような編集環境で Apex コードおよびテストを記述できます。

- [Force.com IDE](#) は Eclipse IDE のプラグインです。Force.com IDE には、Force.com アプリケーションを構築およびリリースする統合インターフェースがあります。開発者および開発チーム向けに設計された IDE には、ソースコードエディタ、テスト実行ツール、ウィザードおよび統合ヘルプなど、Force.com アプリケーション開発を促進するツールが用意されています。基本的なカラー表示エディタ、アウトラインビュー、統合された単体テスト、および保存時の自動コンパイルとエラーメッセージ表示を提供します。インストール方法および使用方法についての詳細は、Web サイトを参照してください。



メモ: Force.com IDE は salesforce.com により提供されるユーザとパートナーをサポートする無料のリソースですが、salesforce.com の主登録契約 (MSA) におけるサービスの一部とはみなされません。

- [Salesforce ユーザインターフェース](#)。すべてのクラスおよびトリガは保存時にコンパイルされ、構文エラーがある場合はフラグが表示されます。エラーがなくなるまで、コードを保存することはできません。Salesforce

ユーザインターフェースはコードの行にも番号を表示し、コメント、キーワード、リテラル文字列など、さまざまな要素を区別しやすいように色分けして表示します。

- ◊ 標準オブジェクトでのトリガの場合は、[設定] から [カスタマイズ] をクリックし、オブジェクトの名前をクリックして、[トリガ] をクリックします。[トリガ] 詳細ページで、[新規] をクリックし、内容 テキストボックスにコードを入力します。
- ◊ カスタムオブジェクトでのトリガの場合は、[設定] から [開発] > [オブジェクト] をクリックし、オブジェクトの名前をクリックします。[トリガ] 関連リストで、[新規] をクリックし、内容 テキストボックスにコードを入力します。
- ◊ クラスの場合は、[設定] から [開発] > [Apex クラス] をクリックします。[新規] をクリックし、内容 テキストボックスにコードを入力します。



メモ: Salesforce の本番組織では、Salesforce ユーザインターフェースを使用して Apex に変更を加えることはできません。

- メモ帳などのテキストエディタで、Apex コードを記述した後、コピーしてアプリケーションに貼り付けたり、API コールのいずれかを使用してリリースできます。



ヒント: Eclipse プラグインを拡張したり、Apex IDE を独自に開発したりする場合、SOAP API には、トリガやクラスをコンパイルし、テストメソッドを実行するためのメソッドが含まれています。一方、メタデータ API には、本番環境にコードをリリースするためのメソッドが含まれています。詳細は、「[Apex のリリース](#)」(ページ 950)および「[Apex の SOAP API および SOAP ヘッダー](#)」(ページ 988)を参照してください。

## テストの記述

テストは、長期間の開発を正常に行うための主要部分であり、開発プロセスの重要な部分を占めます。テストコードを開発時に同時に作成する、テスト駆動型の開発プロセスで開発することを強くお勧めします。

堅牢で、エラーのないコードの開発を促進するため、Apex は単体テストの作成と実行をサポートします。単体テストは、コード内の特定の部分が正しく機能していることを確認するクラスメソッドです。単体テストのメソッドは引数を取らず、データベースへのデータの確定やメールの送信を行うことなく、メソッド定義にキーワードまたはアノテーションでフラグが付けられています。また、テストメソッドは、テストクラス(アノテーションが付加されているクラス)で定義されている必要があります。

さらに、Apex をリリースまたは Force.com AppExchange 用にパッケージ化する前に、次の条件を満たす必要があります。

- Apex コードの少なくとも 75% が単体テストでカバーされており、かつすべてのテストが成功している。

次の点に注意してください。

- ◊ 本番組織にリリースするときに、組織の名前空間内のすべての単体テストが実行されます。
- ◊ へのコールは、Apex コードカバー率の対象とはみなされません。
- ◊ テストメソッドとテストクラスは、Apex コードカバー率の対象とはみなされません。
- ◊ Apex コードの 75% が単体テストでカバーされている必要がありますが、カバー率を上げることだけに集中すべきではありません。アプリケーションのすべてのユースケース(正・誤両方の場合や单一データだ

けでなく複数データの場合)の単体テストを作成するようにしてください。このような多様なユースケースのテストコードを実装することが 75% 以上のカバー率につながります。

- すべてのトリガについて何らかのテストを行う。
- すべてのクラスとトリガが正常にコンパイルされる。

テスト記述について詳細は、「[Apex のテスト](#)」(ページ 229)を参照してください。

## Sandbox 組織への Apex のリリース

Salesforce では、テストやトレーニングなどさまざまな目的のために、Salesforce 本番組織のデータとアプリケーションを損なうことなく、別個の環境で組織のコピーを複数作成できます。これらのコピーは Sandbox と呼ばれ、Salesforce の本番組織とほぼ同じです。Sandbox は Salesforce 本番組織から完全に独立しているため、Sandbox で実行する操作は Salesforce 本番組織に影響せず、逆に本番組織で実行する操作が Sandbox に影響することもありません。

Apex を Force.com IDE のローカルプロジェクトから Salesforce 組織にリリースするには、Force.com のコンポーネントリリースウィザードを使用します。Force.com IDE についての詳細は、  
を参照してください。

Metadata API コールを使用して、開発者組織から Sandbox 組織に Apex をリリースすることもできます。

便利な API コールは、  
です。開発組織または Sandbox 組織では、特定のクラス、クラスのリスト、または名前空間で単体テストを実行できます。

Salesforce には、これらのコマンドをコンソールウィンドウで発行できる Force.com 移行ツールがあります。また、独自のリリースコードを実装することもできます。



メモ: Force.com IDE および Force.com 移行ツールは、salesforce.com が提供するユーザおよびパートナーをサポートする無料のリソースですが、salesforce.com 主登録契約 (MSA) を趣旨とする当社サービスの一部とはみなされていません。

詳細は、「[Force.com Migration Tool の使用](#)」および「[Apex のリリース](#)」を参照してください。

## Salesforce 本番組織への Apex のリリース

すべての単体テストが完了し、Apex コードが適切に実行されていることを確認したら、最後のステップは Salesforce 本番組織に Apex をリリースすることです。

Apex を Force.com IDE のローカルプロジェクトから Salesforce 組織にリリースするには、Force.com のコンポーネントリリースウィザードを使用します。Force.com IDE についての詳細は、  
を参照してください。

また、Salesforce ユーザインターフェースの変更セットを使用して Apex をリリースできます。

詳細および追加のリリースオプションは、「[Apex のリリース](#)」(ページ 950)を参照してください。

## Force.com AppExchange アプリケーションへの Apex コードの追加

AppExchange 用に作成するアプリケーションの Apex クラスまたはトリガを含むこともできます。

パッケージの一部として含まれている Apex はいずれも、累積テストカバー率が少なくとも 75% である必要があります。各トリガについても何らかのテストを行う必要があります。パッケージを AppExchange にアップロードすると、すべてのテストが実行され、エラーがない状態で実行されていることが確認されます。また、インストーラの組織にパッケージがインストールされるときにも、  
アノテーションが付加されたテストが実行されます。テストに  
アノテーションを付加することで、パッケージインストール時にどのテストを実行するかを指定できます。パッケージを正常にインストールするには、このアノテーションが付加されたテストに合格する必要があります。

また、Apex を含む AppExchange パッケージは管理パッケージとすることをお勧めします。

詳細は、『[Force.com Quick Reference for Developing Packages](#)』を参照してください。管理パッケージの Apex についての詳細は、Salesforce オンラインヘルプの「管理パッケージでの Apex の開発」を参照してください。



メモ: 翻訳文のあるカスタム表示ラベルへの参照を含む Apex クラスのパッケージに翻訳を含めるには、トランスレーションワークベンチを有効にし、翻訳されたカスタム表示ラベルで使用されている個々の言語を明示的にパッケージ化します。Salesforce オンラインヘルプの「カスタム表示ラベルの概要」を参照してください。

## Apex を使用する必要がある状況は?

Salesforce では、強力な CRM 機能を提供するアプリケーションが組み込まれています。また、Salesforce では組織に応じて組み込みアプリケーションをカスタマイズする機能も用意されています。ただし、組織には、既存の機能ではサポートされていない複雑なビジネスプロセスがあります。この場合、Force.com プラットフォームには、高度な管理者や開発者がカスタム機能を実装できるさまざまな方法が搭載されています。そうした方法の中には、Apex、Visualforce、および SOAP API などがあります。

### Apex

次のような場合に Apex を使用します。

- ・ Web サービスを作成する
- ・ メールサービスを作成する
- ・ 複数のオブジェクトに複雑な検証を実行する
- ・ ワークフローでサポートされていない複雑なビジネスプロセスを作成する
- ・ カスタムトランザクションロジック (1 つのレコードやオブジェクトだけでなく、トランザクション全体で発生するロジック) を作成する
- ・ 操作がユーザインターフェース、Visualforce ページ、または SOAP API のどこから行われているかに関係なく、操作が実行されるといつでも行われるよう、レコードの保存などの別の操作にカスタムロジックを添付する

## Visualforce

Visualforce では、タグベースのマークアップ言語を使用して、開発者はより効果的にアプリケーションを開発したり、Salesforce のユーザインターフェースをカスタマイズしたりできます。Visualforce を使用して、次のことができます。

- ・ ウィザードやその他のマルチステッププロセスの構築
- ・ アプリケーションを介した独自のカスタムフローコントロールの作成
- ・ 最適かつ効果的なアプリケーションの相互作用を目的とした、ナビゲーションパターンやデータ固有ルールの定義

詳細は、『[Visualforce 開発者ガイド](#)』を参照してください。

## SOAP API

一度に1つのレコードタイプのみを処理し、トランザクション制御 (Savepoint の設定や変更のロールバックなど) を必要としない複合アプリケーションに機能を追加する場合、標準の SOAP API コールを使用します。

詳細は、『[SOAP API 開発者ガイド](#)』を参照してください。

## Apex の制限事項

Apex は、開発者がオンデマンドビジネスアプリケーションを作成する方法を根本的に変えました。しかし、これはプログラミング言語の一般的な目的ではありません。このリリースから、Apex では次の操作ができなくなりました。

- ・ エラーメッセージ以外のユーザインターフェースの要素の表示
- ・ 標準機能の変更。Apex では、機能の実行または機能の追加の回避のみが可能です。
- ・ 一時ファイルの作成
- ・ スレッドの実行



### ヒント:

すべての Apex コードは、他のすべての組織で使用される共有リソースである Force.com プラットフォーム上で実行されます。一貫したパフォーマンスと拡張性を確保するため、Apex の実行は、Apex 実行が Salesforce のサービス全体に一切影響を及ぼさないことを保証するガバナ制限によって制約されています。これは、すべての Apex コードは、1 回のプロセスで実行できる操作数 (DML、SOQL など) に限定されることを意味します。

すべての Apex 要求は、1 件から 50,000 件のレコードを含むコレクションを返します。コードが一度に1つのレコードでしか機能しないことは考えられません。そのため、一括処理を考慮するプログラミングパターンを実装する必要があります。実装しない場合、ガバナ制限による制約を受ける可能性があります。

## 関連リンク

[実行ガバナと制限について](#)

[トリガと一括要求に関するベストプラクティス](#)

## 最新情報

『Summer '13 リリースノート』で、Summer '13 での Apex の新機能と変更された機能を確認してください。

## Apex クイックスタート

Developer Edition または Sandbox 組織を入手したら、Apex の基本概念について学習する必要があります。Apex は Java によく似ているため、多くの機能がなじみ深いものです。

基本を確認したら、最初の Apex プログラムとして非常に単純なクラス、トリガおよび単体テストを作成することができます。

さらに、もう少し複雑な納入先請求書の例を確認することもできます。この例では、多数の言語機能を示します。



メモ: Hello World と納入先請求書のサンプルでは、カスタム項目およびオブジェクトが必要です。項目やオブジェクトを自分で作成したり、オブジェクト、項目および Apex コードを管理パッケージとして Force.com AppExchange からダウンロードできます。詳細は、

を参照してください。

## ドキュメント表記規則

Apex および Visualforce ドキュメントは、次の表記規則を使用しています。

規則	説明
	構文の記述では、等幅フォントは、角かっこを除いて表示されたとおりに入力する必要のある項目を示します。次に例を示します。
斜体	構文の記述では、斜体は変数を示します。実際の値を入力してください。次の例では、3つの値を入力する必要があります。 <code>datatype variable_name [= value];</code> 構文で太字かつ斜体のテキストは、クラス名や変数の値など、ユーザが指定する必要があるコード要素を表します。
<b>Bold Courier font</b>	コードサンプルと構文の記述では、太字の Courier フォントはコードまたは構文の部分を強調します。

*YourClassHere*

規則	説明
< >	構文の記述では、不等号 (< >) は表示されたとおりに入力します。
{ }	構文の説明では、中かっこ ({ }) は表示されたとおりに入力します。
[ ]	構文の記述では、角かっこで囲まれるものはすべて省略可能です。次の例では、 <code>value</code> の指定は省略可能です。
	構文の記述では、パイプ記号は「または」を意味します。次のいずれか(すべてではない)を実行できます。次の例では、2つの方法のいずれかを使用して未入力のセットを作成するか、次のようにセットを入力することができます。

## Apex の基本概念について

一般的に Apex コードには、他のプログラミング言語でなじみのある内容が多く含まれています。

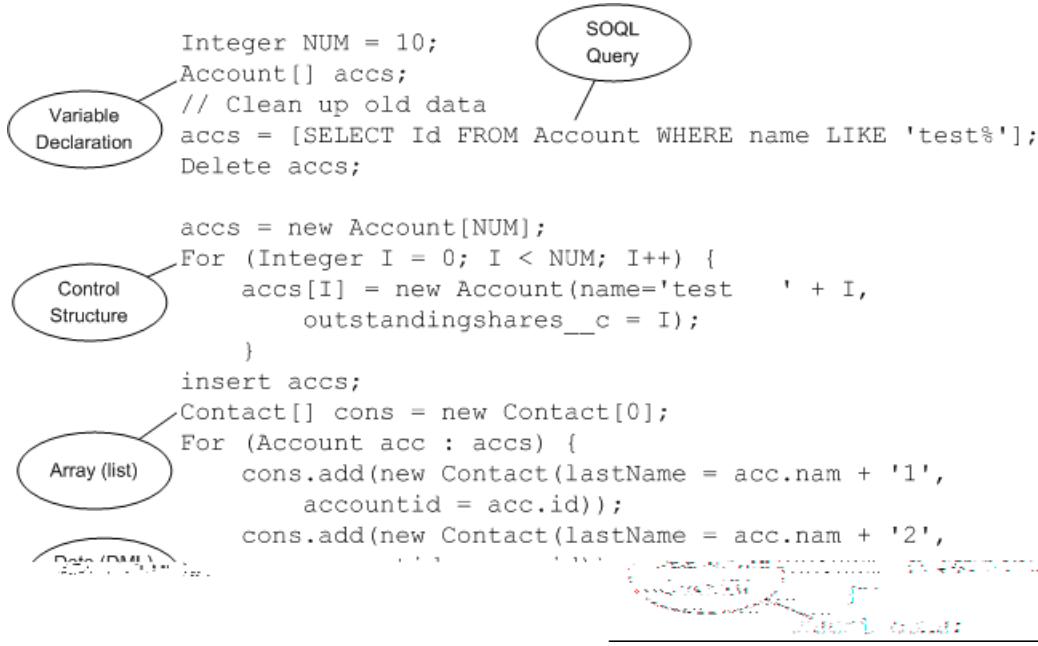


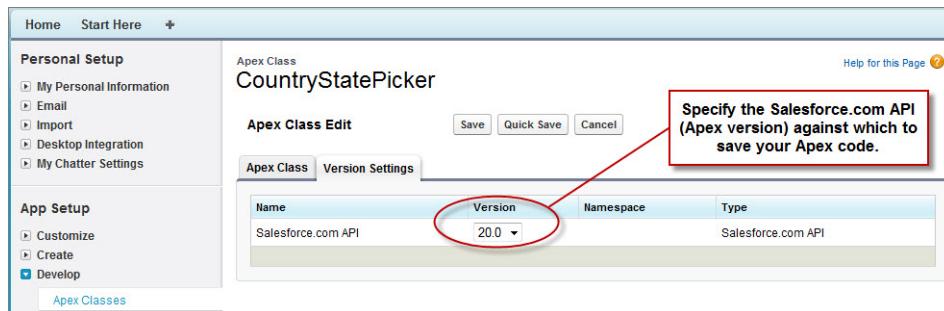
図3: Apexのプログラミング要素

このセクションは、Apex の基本的な機能および基本概念の一部について説明します。

## バージョン設定の使用

Salesforce ユーザインターフェースで、Apex クラスまたはトリガを保存する Salesforce.com API のバージョンを指定できます。この設定は、使用する SOAP API バージョンだけではなく、Apex のバージョンも示します。保存後、バージョンを変更できます。各クラス名またはトリガ名は一意である必要があります。異なるバージョンに同じクラスまたはトリガを保存することはできません。

バージョン設定を使用すると、AppExchange から組織にインストールした管理パッケージの特定のバージョンにクラスまたはトリガを関連付けることができます。管理パッケージのこのバージョンは、より新しいバージョンの管理パッケージがインストールされても、バージョン設定を手動で更新しない限り、クラスまたはトリガによって引き続き使用されます。インストール済み管理パッケージを設定リストに追加するには、使用可能なパッケージのリストからパッケージを選択します。リストは、クラスまたはトリガにまだ関連付けられていないインストール済み管理パッケージがある場合にのみ表示されます。



管理パッケージでバージョン設定を使用する詳細は、Salesforceオンラインヘルプの「パッケージバージョンについて」を参照してください。

## 変数、メソッド、およびクラスの命名

変数、メソッドまたはクラスを命名する場合、Apex の予約キーワードは使用できません。使用できない語には、**予約キーワード**のほかに、**、**、**、**または**、**などの Apex および Force.com プラットフォームの一部である語が含まれます。

## 変数と式の使用

Apex は、強く型付けされた言語です。つまり、最初に参照するときに変数のデータ型を宣言する必要があります。Apex データ型には、Integer、Date、Boolean などの基本のデータ型に加え、lists、maps、objects、sObjects など、高度なデータ型があります。

変数は名前とデータ型で宣言されます。宣言するときに、値を変数に割り当てるすることができます。後で値を割り当てるすることもできます。変数を宣言する場合、次の構文を使用します。

```
datatype variable_name value
```



ヒント: 上記の末尾にあるセミコロンは省略できません。ステートメントの末尾には、必ずセミコロンを使用します。

次の例は、変数の宣言を示します。

```
String myString = 'Hello, World!';
```

Apex では、Integer または String などのすべてのプリミティブデータ型引数は、値によってメソッドに渡されます。つまり、引数への変更はメソッドの範囲内でのみ存在することになります。メソッドが返ったときに、その引数への変更は失われます。

sObject などの非プリミティブデータ型引数も、値によってメソッドに渡されます。つまり、メソッドが返ったときに、渡された引数はメソッドをコールする前と同じオブジェクトをそのまま参照することになり、別のオブジェクトを参照するようには変更できません。ただし、オブジェクトの項目の値はメソッド内で変更できます。

## ステートメントの使用

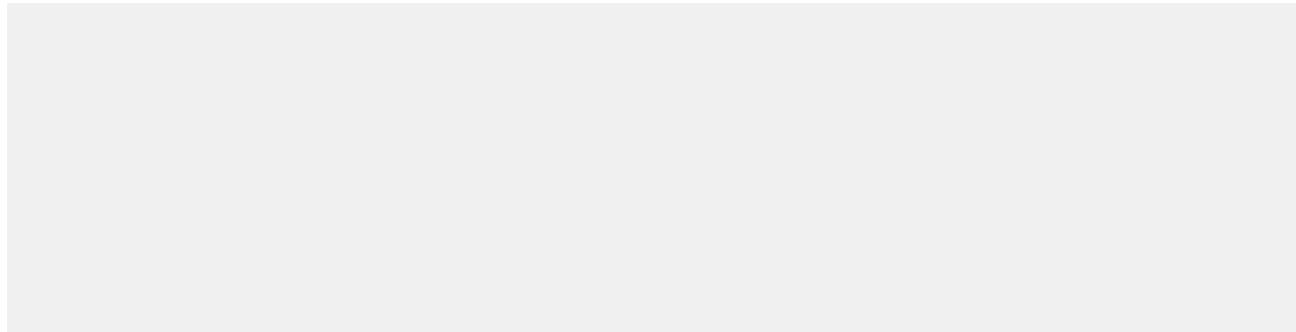
ステートメントは、操作を実行するコード化された指示です。

Apex では、ステートメントの末尾にセミコロンを使用し、次の種類のいずれかになります。

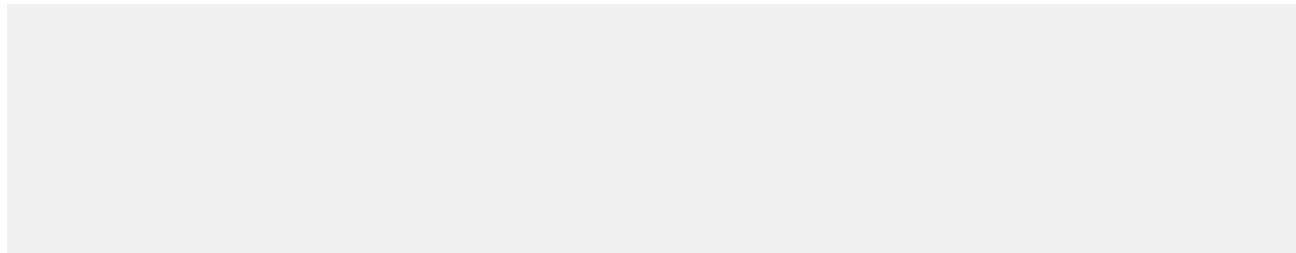
- ・ 割り当て (値の変数への割り当てなど)
- ・ 条件 (if-else)
- ・ ループ:

- ◊ Do-while
  - ◊ While
  - ◊ For
- ロック
  - データ操作言語 (DML)
  - トランザクションの制御
  - メソッド呼び出し
  - 例外処理

ブロックは、中かっこでまとめられる一連のステートメントです。単一のステートメントが使用できる場所であればどこでも使用できます。次に例を示します。



ブロックが1つのステートメントだけで構成される場合、中かっこを取ることができます。次に例を示します。



## コレクションの使用

Apex には、次の種類のコレクションがあります。

- リスト (配列)
- Map
- Set

リストは、Integer、String、オブジェクト、他のコレクションなどの要素のコレクションです 要素のシーケンスが重要な場合はリストを使用します。リスト内に重複する要素を含めることができます。

リスト内の最初のインデックスの位置は必ず 0 になります。

リストを作成する手順は、次のとおりです。

- キーワードを使用します。
- 文字で囲まれた要素の種類の前に キーワードを使用します。

次の構文を使用して、リストを作成します。

```
datatype list_name  
datatype  
datatype value    value2
```

次の例では Integer のリストを作成し、変数 `list_name` に割り当てます。Apex は強く型付けされているため、データ型を Integer のリストとして宣言する必要があります。

詳細は、[「List」](#) (ページ 39)を参照してください。

セットとは、一意の順不同の要素のコレクションです。セットには String、Integer、Date などのプリミティブデータ型を含めることができます。また、より複雑な sObject などのデータ型も含められます。

セットを作成する手順は、次のとおりです。

- キーワードを使用します。
- 文字で囲まれたプリミティブデータ型の前に キーワードを使用します。

次の構文を使用して、セットを作成します。

```
datatype set_name  
datatype  
datatype value    value2
```

次の例では、String のセットを作成します。セットの値は、中かっこ `{ }` を使用して渡されます。

詳細は、[「Set」](#) (ページ 47)を参照してください。

対応付けは、キー - 値のペアのコレクションです。キーには、任意のプリミティブデータ型を使用できます。値には、プリミティブデータ型およびオブジェクトその他のコレクションを含められます。キーによる検索が重要な場合は、対応付けを使用します。対応付けでは重複する値は存在できますが、各キーは一意である必要があります。

対応付けを作成する手順は、次のとおりです。

- キーワードを使用します。
- 文字で囲まれ、カンマで区切られたキー - 値の前に キーワードを使用します。

次の構文を使用して、対応付けを作成します。

```
key_datatype  value_datatype  map_name  
            key_datatype  value_datatype  
            key_datatype  value_datatype  
key1_value    value1_value  
key2_value    value2_value
```

次の例では、キーのデータ型が Integer、値が String である対応付けを作成します。この例では、対応付けが作成されると、中かっこ の間に対応付けの値が渡されます。

詳細は、「[対応付け](#)」(ページ 49)を参照してください。

## 条件分岐の使用

ステートメントは、アプリケーションが条件に基づいてさまざまなことを実行できるようにする true-false テストです。基本構文は次のとおりです。

```
Condition
```

詳細は、「[条件 \(If-Else\) ステートメント](#)」(ページ 73)を参照してください。

## ループの使用

ステートメントを使用すると、アプリケーションは条件に基づいて操作を実行できますが、ループはアプリケーションが条件に基づいて同じ操作を繰り返し実行するよう指示します。Apex では、次の種類のループを使用できます。

- Do-while
- While
- For

*Do-while* ループは、コードの実行後に条件をチェックします。

*While* ループは、コードの実行前の開始時に条件をチェックします。

*For* ループを使用すると、ループ内で使用される条件をより詳細に制御できます。また、Apex では、条件を設定する従来の For ループ、条件の一部としてリストおよび SOQL クエリを使用する For ループを使用できます。

詳細は、「[ループ](#)」(ページ 74)を参照してください。

## 最初の Apex コードおよびトリガの作成

このステップごとのチュートリアルでは、簡単な Apex クラスおよびトリガを作成する方法を説明します。また、これらのコンポーネントを本番組織にリリースする方法を示します。

このチュートリアルは、最初のステップで作成される Book というカスタムオブジェクトに基づいています。このカスタムオブジェクトはトリガを使用して更新されます。

### 関連リンク

[カスタムオブジェクトの作成](#)

[Apex クラスの追加](#)

[Apex トリガの追加](#)

[テストクラスの追加](#)

[本番組織へのコンポーネントのリリース](#)

## カスタムオブジェクトの作成

前提条件:

Sandbox **Unlimited** または Enterprise Edition 組織の Salesforce 取引先、または開発者組織の取引先。

Sandbox 組織の作成についての詳細は、Salesforce オンラインヘルプの「Sandbox の概要」を参照してください。無償の開発者組織にサインアップするには、[Developer Edition 環境のサインアップページ](#)を参照してください。

このステップでは、Price というカスタム項目を持つ Book というカスタムオブジェクトを作成します。

1. Sandbox または開発者組織にログインします。
2. [設定] から [作成] > [オブジェクト] をクリックし、[新規カスタムオブジェクト] をクリックします。
3. 表示ラベルに「」と入力します。
4. 複数形の表示ラベルには「」と入力します。
5. [保存] をクリックします。  
ご覧ください! 最初のカスタムオブジェクトを作成できました。次に、カスタム項目を作成します。
6. Book の詳細ページの [カスタム項目 & リレーション] セクションで、[新規] をクリックします。
7. データ型に [Number] を選択し、[次へ] をクリックします。
8. 項目の表示ラベルに「」と入力します。
9. 長さのテキストボックスに「16」と入力します。
10. 小数点の位置を指定するテキストボックスに「2」を入力し、[次へ] をクリックします。
11. 項目レベルのセキュリティのデフォルト値を受け入れるには、[次へ] をクリックします。
12. [保存] をクリックします。

Book というカスタムオブジェクトを作成し、それにカスタム項目を追加しました。カスタムオブジェクトには、Name や CreatedBy などの一部の標準の項目が含まれており、より実装に固有の項目をさらに追加することができます。

きます。このチュートリアルでは、Price 項目は Book オブジェクトの一部であり、アクセスするには次のステップで記述する Apex クラスを使用します。

## 関連リンク

[最初のApex コードおよびトリガの作成](#)

[Apex クラスの追加](#)

## Apex クラスの追加

前提条件:

- Sandbox **Unlimited** または Enterprise Edition 組織の Salesforce 取引先、または開発者組織の取引先。
- **Book カスタムオブジェクト**

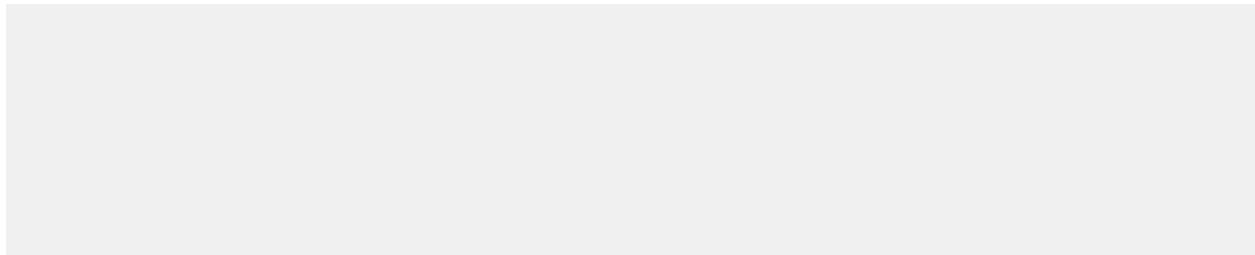
このステップでは、本の価格を更新するメソッドを含む Apex クラスを追加します。このメソッドは、次のステップで追加するトリガでコールされます。

1. [設定] から [開発] > [Apex クラス] をクリックし、[新規] をクリックします。
2. クラスエディタで、次のクラス定義を入力します。



前述のコードは、次のステップで 1 つのメソッドを追加するクラス定義です。Apex コードは、通常、クラスに含まれています。このクラスは \_\_\_\_\_ と定義されているため、他の Apex クラスおよびトリガで使用できます。詳細は、「[クラス、オブジェクトおよびインターフェース](#)」(ページ 162)を参照してください。

3. クラスの開きかっこおよび閉じかっこ間にこのメソッド定義を追加します。

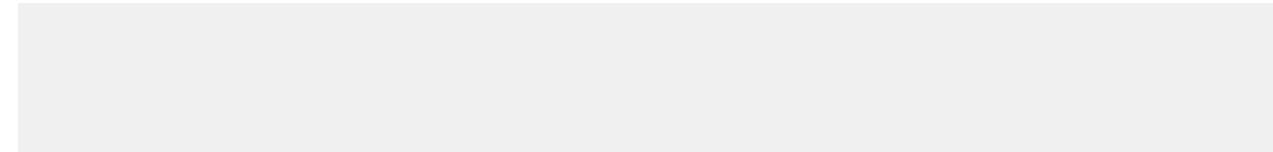


このメソッドは \_\_\_\_\_ と呼ばれ、公開かつ静的メソッドです。これは静的メソッドであるため、メソッドにアクセスするためにクラスのインスタンスを作成する必要はありません。このメソッドにアクセスするには、クラス名の後にカンマ (,)、メソッド名を指定します。詳細は、「[静的およびインスタンスマソッド](#)」(ページ 178)を参照してください。

このメソッドでは 1 つのパラメータ、Book レコードのリストを使用します。これは変数 \_\_\_\_\_ に割り当てられます。オブジェクト名の後に \_\_\_\_\_ を記述し、\_\_\_\_\_ とします。これは、この項目がカスタムオブジェク

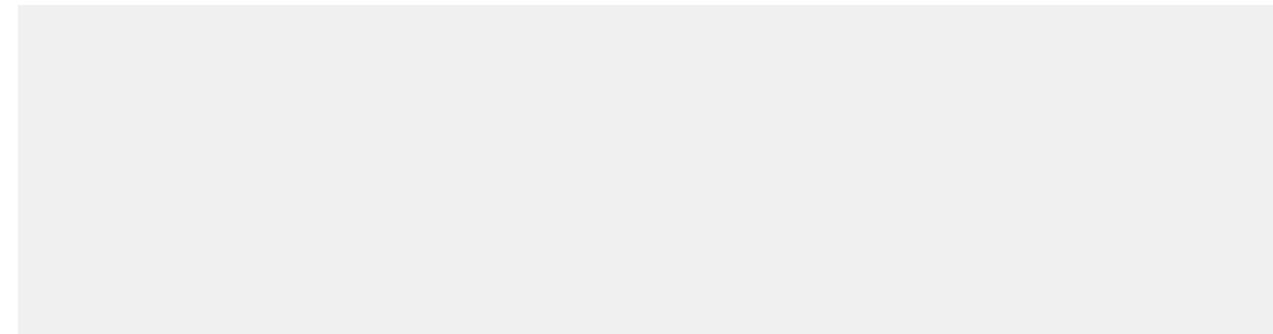
ト、つまり自分で作成した項目であることを示します。Account など Salesforce アプリケーションで提供される標準オブジェクトの末尾はこのポストフィックスではありません。

コードの次のセクションでは、メソッド定義の残りを記述します。



項目名の後に `__` を記述し、`Class` とします。これは、この項目がカスタム項目、つまり自分で作成した項目であることを示します。Salesforce のデフォルトで提供されている標準項目へのアクセスには同じ種類のドット表記が使用されますが、`Class` は使用されません。たとえば、`Account` の `getPrice()` 項目の末尾には付きません。ステートメント `String price = record.getPrice();` では `getPrice()` の古い値を取り、この値を 0.9 で乗算します。つまり、10% 割り引きされた値にします。次に、新しい値を `record.setPrice(record.getPrice() * 0.9)` 項目に保存します。演算子はショートカットです。このステートメントを作成する他の方法は、`record.setPrice(record.getPrice() * 0.9)` です。「式の演算子について」(ページ 63)を参照してください。

4. [保存] をクリックすると、新しいクラスが保存されます。これで、次の完全なクラス定義が設定されます。



これで、Book (本) リストを反復処理し、各本の価格項目を更新するコードを含むクラスを作成できました。このコードは、次のステップで作成するトリガによってコールされる `updateBooks()` 静的メソッドの一部です。

## 関連リンク

- [最初のApex コードおよびトリガの作成](#)
- [カスタムオブジェクトの作成](#)
- [Apex トリガの追加](#)

## Apex トリガの追加

前提条件:

- Sandbox **Unlimited** または Enterprise Edition 組織の Salesforce 取引先、または開発者組織の取引先。
- [MyHelloWorld Apex クラス](#)

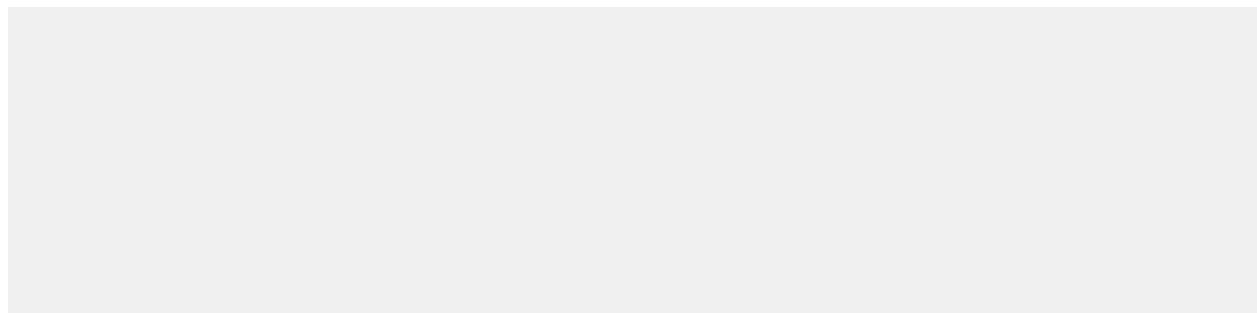
このステップでは、前のステップで作成した  
カスタムオブジェクトのトリガを作成します。

クラスの

メソッドをコールする

トリガは、Force.com プラットフォームデータベースで特定のタイプのレコードが挿入、更新、または削除される前または後に実行するコードの一部です。各トリガは、トリガが実行されるレコードへのアクセス権限を提供する一連のコンテキスト変数で実行します。すべてのトリガは一括で実行します。つまり、複数のレコードを一度に処理します。

1. [設定] から、[作成]>[オブジェクト] をクリックし、作成したオブジェクトの名前(Book) をクリックします。
2. [トリガ] セクションで、[新規] をクリックします。
3. トリガエディタで、デフォルトのテンプレートコードを削除し、このトリガの定義を入力します。



コードの最初の行はトリガを定義します。

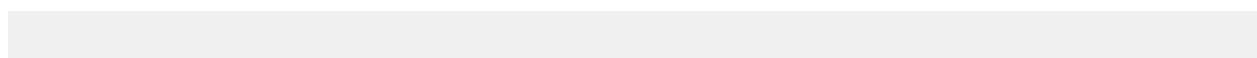
このコードはトリガに名前を付け、トリガを実行するオブジェクトを指定し、トリガを実行するイベントを定義します。たとえば、このトリガを HelloWorldTrigger とし、オブジェクトで動作し、新しいブックがデータベースに挿入される前に実行するようにします。

トリガの次の行は、という名前のブックレコードのリストを作成し、というトリガコンテキスト変数の内容を割り当てます。などのトリガコンテキスト変数は、すべてのトリガで暗黙的に定義され、トリガを実行するレコードにアクセスできるようにします。この場合、には、挿入される新しいブックがすべて含まれます。

コードの次の行は、  
に渡します。

クラスのメソッド

をコールします。新しいブックの配列



挿入されるすべてのブックの価格を更新するために必要なすべてのコードが揃いました。ただし、このパズルのピースが1つ不足しています。単体テストは、コードを記述する上で重要な部分であり、必須です。次のステップでは、これが重要である理由を確認し、テストクラスを追加できます。

## 関連リンク

[最初のApex コードおよびトリガの作成](#)

[Apex クラスの追加](#)

[テストクラスの追加](#)

## テストクラスの追加

前提条件:

- Sandbox **Unlimited** または Enterprise Edition 組織の Salesforce 取引先、または開発者組織の取引先。
- [HelloWorldTrigger Apex トリガ](#)

このステップでは、1つのテストメソッドを持つテストクラスを追加します。また、テストを実行して、コードカバー率を検証します。テストメソッドはトリガとクラスのコードを実行して検証します。また、トリガとクラスのコードカバー率が 100% に達するようにします。



メモ: テストは開発プロセスの重要な部分です。Apex をリリースまたは Force.com AppExchange 用にパッケージ化する前に、次の条件を満たす必要があります。

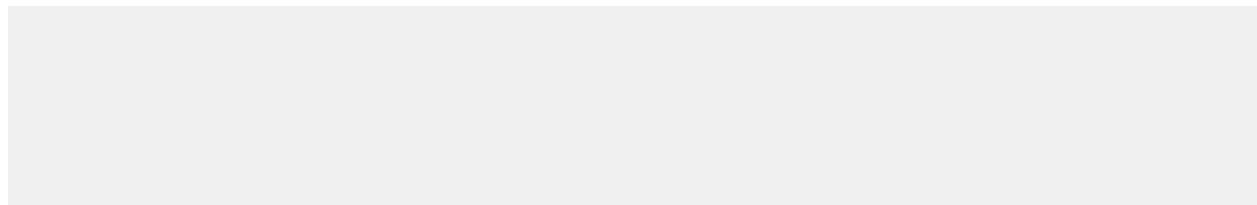
- Apex コードの少なくとも 75% が単体テストでカバーされており、かつすべてのテストが成功している。

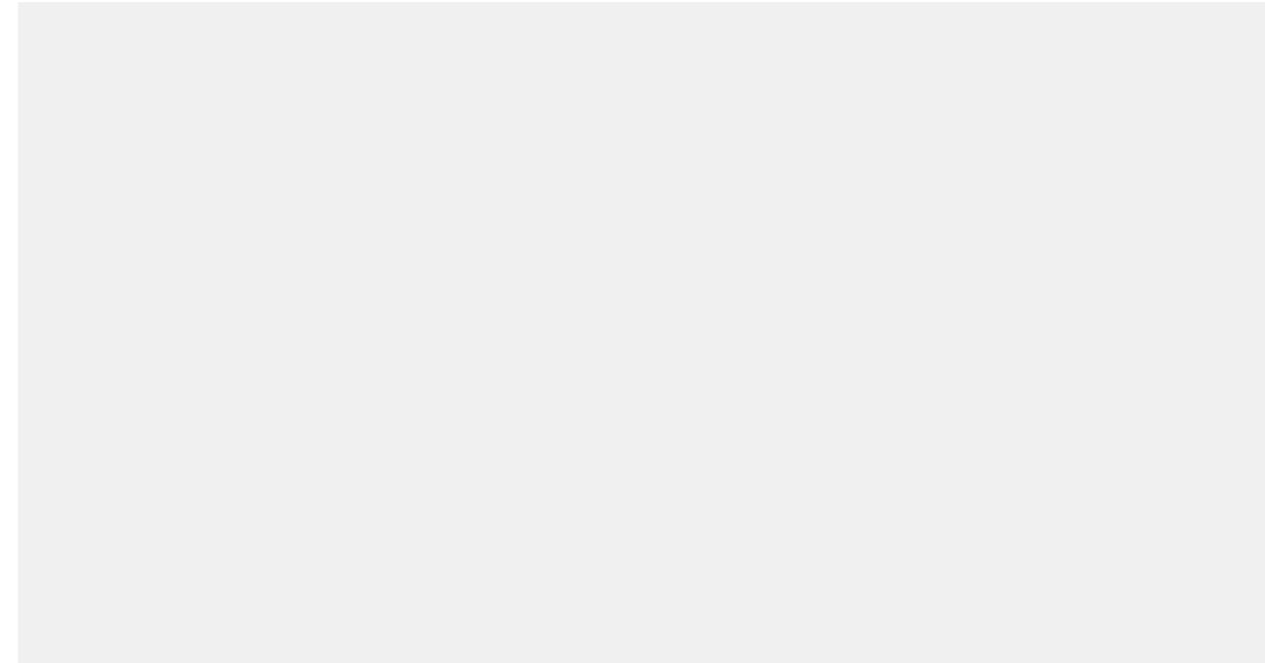
次の点に注意してください。

- ◊ 本番組織にリリースするときに、組織の名前空間内のすべての単体テストが実行されます。
- ◊ \_\_\_\_\_へのコールは、Apex コードカバー率の対象とはみなされません。
- ◊ テストメソッドとテストクラスは、Apex コードカバー率の対象とはみなされません。
- ◊ Apex コードの 75% が単体テストでカバーされている必要がありますが、カバー率を上げることだけに集中すべきではありません。アプリケーションのすべてのユースケース(正・誤両方の場合や单一データだけでなく複数データの場合)の単体テストを作成するようにしてください。このような多様なユースケースのテストコードを実装することが 75% 以上のカバー率につながります。

- すべてのトリガについて何らかのテストを行う。
- すべてのクラスとトリガが正常にコンパイルされる。

1. [設定] から [開発] > [Apex クラス] をクリックし、[新規] をクリックします。
2. クラスエディタで、このテストクラスの定義を追加し、[保存] をクリックします。

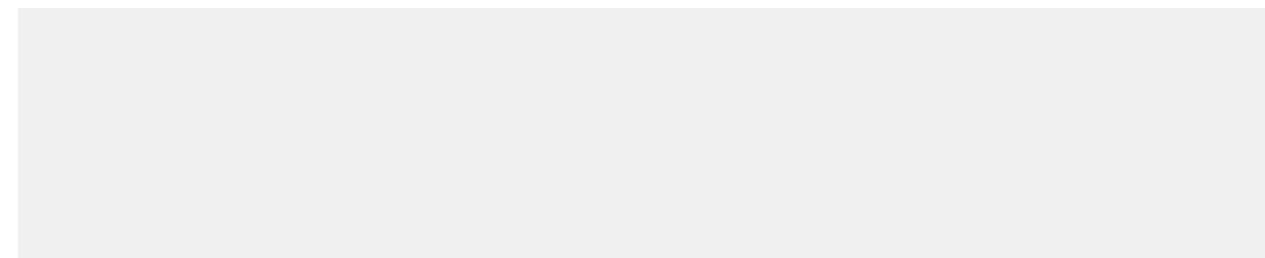




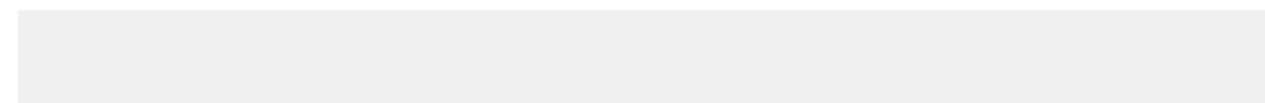
このクラスは、**アノテーション**を使用して定義されています。こうして定義されたクラスには、**テストメソッド**のみが含まれます。テストメソッドを既存のクラスに追加する場合に対して、テスト用に個別のクラスを作成することの利点の1つは、**アノテーション**で定義されたクラスは、すべての Apex コードに対して組織で設定された 3 MB の制限の対象としてカウントされないことです。**アノテーション**を個別のメソッドに追加することもできます。詳細は、「[アノテーション](#)」(ページ 202)および「[実行ガバナンスと制限について](#)」(ページ 337)を参照してください。

メソッドは**アノテーション**として定義されます。つまり、データベースに変更が行われると、実行の完了時に自動的にロールバックされ、テストメソッドで作成されたテストデータを削除する必要はありません。

まず、テストメソッドは新しいブックを作成し、データベースに一時的に挿入します。ステートメントによって、デバッグログに価格の値が書き込まれます。



ブックが挿入されると、コードは、挿入時にブックに割り当てられた ID を使用して新たに挿入されたブックを取得し、トリガによって変更された新しい価格を記録します。



クラスが実行されると、項目が更新され、値が10%減少します。次の行は実際のテストで、メソッドが実際に実行され、予想どおりの結果が得られたことを検証します。

- クラスページの [テストを実行] をクリックし、このクラスのすべてのテストメソッドを実行します。この場合、テストメソッドは1つしかありません。

Apex テスト結果ページは、テストの実行が終了した後に表示されます。このページには、テストに失敗した数、コードカバー率の情報、およびダウンロード可能なログファイルへのリンクなどのテストの結果に関する情報が含まれます。

- [ダウンロード] をクリックして、ログファイルを開きます。トリガイベント、クラスメソッドへのコール、およびトリガ前後の価格のデバッグ出力に関するログ情報が表示されます。  
また、Apex コードのデバッグに、開発者コンソールを使用することができます。Salesforce オンラインヘルプの「開発者コンソール」を参照してください。
- また、Apex テスト実行ページでテストを実行することもできます。ここでは、非同時的にテストが実行されるため、テ스트ランの終了と結果の出力を待たずに、テストが実行されている最中にユーザインターフェースで他のタスクを実行し、後でページに戻ってテストの状況を確認することができます。
  - [設定] で、[開発] > [Apex テスト実行] をクリックします。
  - [テストを実行] をクリックします。
  - クラスを選択して [実行] をクリックします。

テストの実行が終了すると、次のことができます。

- テストをクリックして、結果の詳細を確認します。テストが失敗すると、最初のエラーメッセージとスタック追跡が表示されます。
- ソース Apex コードを表示するには、[表示] をクリックします。

- テストが実行し終わったら、コードカバー率を確認します。
  - [設定] で、[開発] > [Apex クラス] をクリックします。
  - [組織のコードカバー率を計算する] をクリックし、単体テストによってカバーされる組織内のコードの量を表示します。
  - [コードカバー率] 列で 100% をクリックすると、単体テストがカバーするコードの行を表示できます。

[開発] > [Apex トリガ] をクリックして、[設定] でトリガのリストを確認します。記述したトリガにも 100% のコードカバー率が適用されていることを確認します。

ここまでで、検証済みで、開発環境で実行する Apex コードを得るために必要なすべてのステップを完了しました。実際に、コードを十分にテストして満足できる結果が得られたら、他の要件のコンポーネントと共にコード

を本番組織にリリースできます。次のステップでは、コードと作成したカスタムオブジェクトを本番組織にリリースする方法を説明します。

## 関連リンク

- [最初のApex コードおよびトリガの作成](#)
- [Apex トリガの追加](#)
- [本番組織へのコンポーネントのリリース](#)

## 本番組織へのコンポーネントのリリース

前提条件:

- Sandbox の Unlimited または Enterprise Edition 組織の Salesforce アカウント
- [HelloWorldTestClass Apex テストクラス](#)
- 受信変更セットを本番組織で受信できるようにする、Sandbox と本番組織間のリリース接続。Salesforce オンラインヘルプの「[変更セットの概要](#)」を参照してください。
- 送信変更セットを作成、編集、またはアップロードするための「[変更セットの作成とアップロード](#)」ユーザ権限

このステップでは、変更セットを使用して、以前に作成した Apex コードとカスタムオブジェクトを本番組織にリリースできます。

変更セットは Unlimited、Enterprise、または Database.com Edition 組織でのみ使用できるものであるため、この手順は、Developer 組織には適用されません。Developer Edition アカウントを使用している場合は、その他のリリースメソッドを使用できます。「[Apex のリリース](#)」を参照してください。

1. [設定] で、[リリース] > [送信変更セット] をクリックします。
2. スプラッシュページが表示される場合は、[次へ] をクリックします。
3. [変更セット] リストで、[新規] をクリックします。
4. など、変更セットの名前を入力し、必要に応じて説明を入力します。[保存] をクリックします。
5. 変更セットのコンポーネントのセクションで、[追加] をクリックします。
6. コンポーネントの種類のドロップダウンリストで [Apex] を選択してから、リストから MyHelloWorld クラスと HelloWorldTestClass クラスを選択し、[変更セットに追加] をクリックします。
7. [運動関係を参照/追加] をクリックし、運動コンポーネントを追加します。
8. すべてのコンポーネントを選択するには、上部のチェックボックスをオンにします。[変更セットに追加] をクリックします。
9. 変更セットページの変更セット詳細セクションで、[アップロード] をクリックします。
10. 対象組織（この場合は本番組織）を選択し、[アップロード] をクリックします。
11. 変更セットのアップロードが完了したら、本番組織にリリースできます。
  - a. 本番組織にログインします。
  - b. [設定] で、[リリース] > [受信変更セット] をクリックします。
  - c. スplash ページが表示される場合は、[次へ] をクリックします。
  - d. リリース待ちの変更セットのリストで、変更セットの名前をクリックします。

- e. [リリース] をクリックします。

このチュートリアルでは、カスタムオブジェクトの作成方法、Apex トリガ、クラス、およびテストクラスの追加方法、およびコードのテスト方法を学習しました。最後に、変更セットを使用して、コードとカスタムオブジェクトのアップロード方法も学習しました。

## 関連リンク

- [最初のApex コードおよびトリガの作成](#)
- [テストクラスの追加](#)

## 第2章

### 言語構造

トピック:

- データ型
- 変数
- 式
- 代入ステートメント
- 条件 (If-Else) ステートメント
- ループ
- Exception ステートメント
- SOQL および SOSL クエリ
- Apex でのデータの使用

Apex は、強く型付けされたオブジェクト指向の、大文字と小文字を区別しないプログラミング言語です。Apex 言語構造は、Apex でプログラムを記述できるようにするビルディングブロックとなっています。これらの言語構造を使用して、組み込まれたデータ型(プリミティブと sObject)の列挙の変数と定数、およびシステムとユーザが指定した Apex 型に基づくカスタムデータ型の変数と定数を宣言できます。Apex では、式、割り当て、および条件ステートメントが提供されます。他のプログラミング言語と同様に、Apex では例外処理とさまざまな種類のループを使用できます。他の言語とは異なり、Apex には SOQL For ループと呼ばれる特殊な種類のループがあり、クエリ結果をまとめることができます。Apex はデータベースに統合されているため、インラインクエリの記述、レコードロックの実行、およびトランザクションの制御が可能です。

次の言語構造は、Apex の基本部分を形成します。

- データ型
- 変数
- 式
- 代入ステートメント
- 条件 (If-Else) ステートメント
- ループ
- SOQL および SOSL クエリ
- ロックステートメント
- トランザクションの制御
- 例外ステートメント

Apex はトリガまたはクラスのいずれかに含まれます。詳細は、「[トリガ](#)」(ページ 132)および「[クラス、オブジェクトおよびインターフェース](#)」(ページ 162)を参照してください。

## データ型

Apex の場合、すべての変数および式は次のいずれかのデータ型です。

- Integer、Double、Long、Date、Datetime、String、ID、または Boolean (「[プリミティブデータ型](#)」(ページ 30) を参照) などのプリミティブデータ型
- 取引先、取引先責任者、または MyCustomObject\_\_c など、汎用 sObject または標準 sObject (「[sObject 型](#)」(ページ 33)を参照)
- 次のものを含むコレクション
  - ◊ プリミティブ、sObjects、ユーザ定義のオブジェクト、Apex クラスから作成されたオブジェクト、または コレクションのリスト (配列) (「[List](#)」(ページ 39)を参照)
  - ◊ プリミティブ型のセット (「[Set](#)」(ページ 47)を参照)
  - ◊ プリミティブからプリミティブ、sObject またはコレクションへの対応付け (「[対応付け](#)」(ページ 49)を参照)
- 列挙型と呼ばれる型付けされた値のリスト (「[enum](#)」(ページ 56)を参照)
- ユーザ定義の Apex クラスから作成されるオブジェクト (「[クラス、オブジェクトおよびインターフェース](#)」(ページ 162)を参照)
- システムが提供する Apex クラスから作成されるオブジェクト (「[Apex クラス](#)」(ページ 650)を参照)
- null (任意の変数に割り当てることができる 定数)

メソッドは、上記のいずれかのデータ型を返すか、または値を返さない Void 型となります。

データ型チェックはコンパイル時に厳密に行われます。たとえば、データ型 Integer のオブジェクト項目に String 型の値が割り当てられると、パーサーはエラーを生成します。ただし、すべてのコンパイル時の例外は、エラーの行番号および列を記載した特定の障害コードとして返されます。詳細は、「[Apex のデバッグ](#)」(ページ 319)を参照してください。

## プリミティブデータ型

Apex は、SOAP API と同じプリミティブデータ型を使用します。すべてのプリミティブデータ型は、値によって渡されます。

すべての Apex 変数は、クラスのメンバー変数であるかメソッド変数であるかに関係なく、 に初期化されます。変数を使用する前に、必ず適切な値に初期化してください。たとえば、Boolean 変数を に初期化します。

Apex のプリミティブデータ型は次のとおりです。

データ型	説明
Blob	単一のオブジェクトとして保存されるバイナリデータのコレクション。 メソッドを使用してこのデータ型を String に変換したり、 メソッドを使用して String からこのデータ型に変換したりできます。 Blobs は Web サービス引数として受け入れられ、ドキュメントに保存され (ドキュメントの本文は Blob 型)、添

データ型	説明
	付ファイルとして送信されます。 詳細は、「 <a href="#">クラス</a> 」(ページ 807)を参照してください。
Boolean	、 、 のみを割り当てる値。 次に例を示します。
Date	特定の日を示す値。 Datetime 値と異なり、 Date 値に時間に関する情報は含まれません。 Date は必ず、 システムの静的メソッドを使用して作成する必要があります。 この Date 値は、 単に Date 変数に数値を追加して日付を追加するなどの処理を行うことはできません。 <a href="#">Date</a> メソッドを使用する必要があります。
Datetime	タイムスタンプなど、 特定の日と時刻を示す値。 Datetime は必ず、 システムの静的メソッドを使用して作成する必要があります。 Datetime 値に対して、 単に Datetime 変数に数値を追加して分を追加するなどの処理を行うことはできません。 <a href="#">Datetime</a> メソッドを使用する必要があります。
Decimal	小数点を含む数値。 Decimal は、 任意の精度数です。 通貨項目には自動的に Decimal 型が割り当てられます。  <code>scale</code> つまり小数部分の桁数を明示的に、 メソッドを使って Decimal に設定しない場合、 scale は Decimal が作成された項目によって決まります。 <ul style="list-style-type: none"> <li>decimal がクエリの一部として作成される場合、 スケールはクエリから返される項目のスケールに基づきます。</li> <li>decimal が string から作成される場合、 スケールは string の小数点以下の桁の文字数となります。</li> <li>decimal が小数以外の数値から作成される場合、 数値を string に変換して小数点以下の桁の文字数を使用することで、 スケールを決定します。</li> </ul>
	小数点を含む 64 ビットの数値。 Doubles の最小値は - $2^{63}$ 希 啊口 ナ' 頭ズににズ増么最 拔二

データ型	説明
Integer	小数点を含まない 32 ビットの数値。Integer の最小値は -2,147,483,648、最大値は 2,147,483,647 です。次に例を示します。
Long	小数点を含まない 64 ビットの数値。Longs の最小値は $-2^{63}$ 、最大値は $2^{63}-1$ です。Integer が提供するよりも広範囲の値が必要な場合に、このデータ型を使用します。次に例を示します。
String	<p>单一引用符で囲まれた文字のセット。次に例を示します。</p> <p>文字列サイズ: String には、使用できる文字数の制限はありません。<a href="#">ヒープサイズの制限</a>を使用して、Apex プログラムが過度に大きくならないようにします。</p> <p>空の文字列と末尾の空白文字: sObject の String 項目値は SOAP API と同じ規則に従います。空白は使用できず（　　を除く）、先頭および末尾に空白文字を使用できません。データベースの保存はこれらの規則に従います。</p> <p>これに対し、Apex の String には　　または空白を使用できます。また、先頭と末尾に空白文字を使用できます(メッセージを構築するような場合に使用できます)。</p> <p>Solution sObject 項目の SolutionNote は、特別なデータ型 String として処理されます。HTML ソリューションを有効にした場合、この項目で使用される HTML タグは、オブジェクトが作成または更新される前に検証されます。無効な HTML が入力されると、エラーが発生します。この項目で使用される JavaScript は、オブジェクトが作成または更新される前に削除されます。次の例では、Solution が詳細ページに表示される場合、SolutionNote 項目には H1 HTML 書式が適用されます。</p> <p>次の例では、Solution が詳細ページに表示される場合、SolutionNote 項目には "こんにちはさようなら" のみが含まれます。</p>

データ型	説明
	<p>詳細は、Salesforce オンラインヘルプの「HTML ソリューションの概要」を参照してください。</p> <p>エスケープシーケンス : Apex のすべての String は SOQL 文字列と同じエスケープシーケンスを使用します ( (バックスペース)、 (タブ)、 (改行)、 (フォームフィード)、 (行頭復帰)、 (二重引用符)、 (一重引用符)、 (バックスラッシュ))</p>

SOAP API と同様、Apex では汎用の sObject 抽象型を使用してオブジェクトを表すことができます。sObject データ型は、さまざまな種類の sObjects を処理するコードで使用できます。

演算子は具体的な sObject 型を要求するため、すべてのインスタンスは特定の sObjects です。次に例を示します。

汎用 sObject 型と特定の sObject 型の間にキャストを使用することもできます。次に例を示します。

sObjects はオブジェクトと同様に機能するため、次のようにになります。

DML 操作は汎用 sObject データ型および正規の sObjects として宣言される変数を処理します。

sObject 変数は に初期設定されますが、 演算子を使用して有効なオブジェクト参照に割り当てることができます。次に例を示します。

新しい sObject をインスタンス化する場合、開発者はカンマで区切られた  
に指定することもできます。次に例を示します。

Force.com プラットフォームデータベースから既存の sObject へのアクセスについての詳細は、『*Force.com SOQL および SOSL リファレンス*』の「SOQL および SOSL クエリ」を参照してください。



メモ: sObject の ID は参照専用の値で、clone 操作でクリアされない限り、またはコンストラクタがアサインされない限り、Apex で明示的に変更できません。Force.com プラットフォームは、オブジェクトトレコードが初めてデータベースに挿入されると、ID 値を自動的に割り当てます。詳細は、[List](#) (ページ 39) を参照してください。

## カスタム表示ラベル

カスタム表示ラベルは標準の sObjects ではありません。カスタム表示ラベルの新規インスタンスを作成することはできません。カスタム表示ラベルの値にアクセスするには、必ず `label_name` を使用します。次に例を示します。

カスタム表示ラベルについての詳細は、Salesforce オンラインヘルプの「[カスタム表示ラベルの概要](#)」を参照してください。

## sObject 項目へのアクセス

Java の場合と同様、単純なドット表記を使用して sObject 項目にアクセスしたり、変更したりできます。次に例を示します。

作成者 または 最終更新日 など、システムによって生成された項目は変更できません。変更しようとすると、Apex ランタイムエンジンはエラーを生成します。また、数式項目値と、コンテキストユーザ参照専用の他の項目値も変更できません。

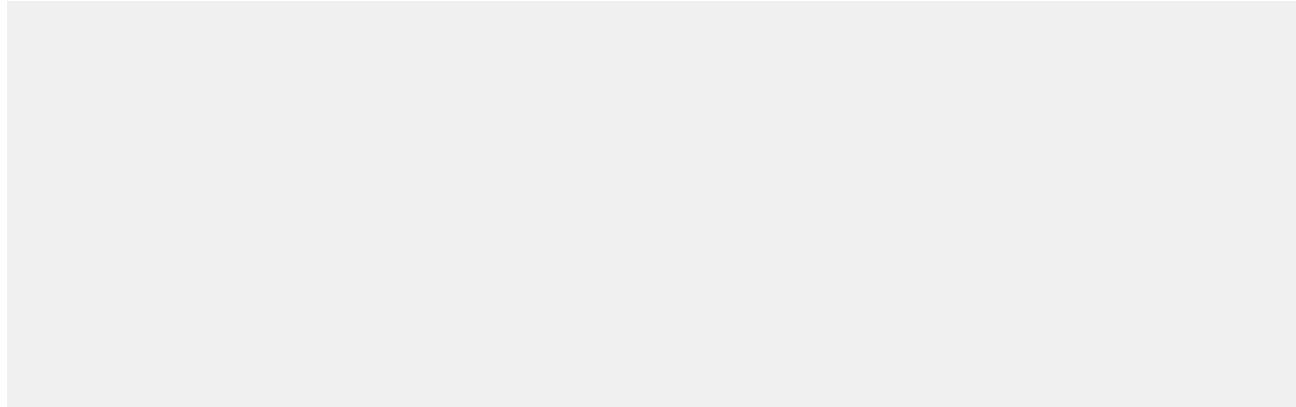
Account などの特定のオブジェクトではない汎用 sObject 型の場合、ドット表記を使用して 項目のみを取得できます。Salesforce.com API バージョン 27.0 以降を使用して保存された Apex コードの 項目を設定できます。また、汎用の sObject メソッドおよび メソッドも使用できます。「[sObject メソッド](#)」を参照してください。

この例では、 項目にアクセスする方法および汎用 sObject で許可されない操作を示します。

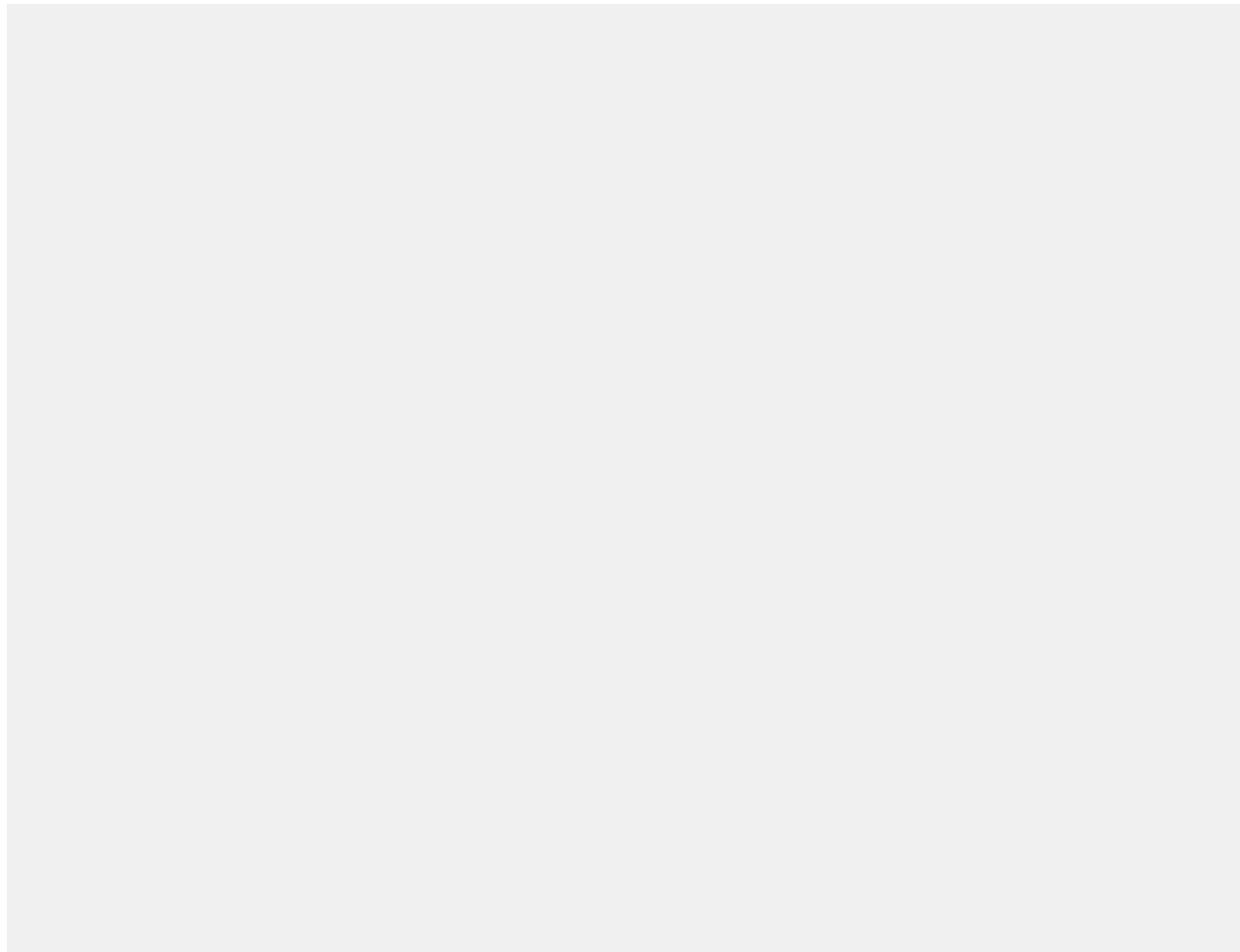


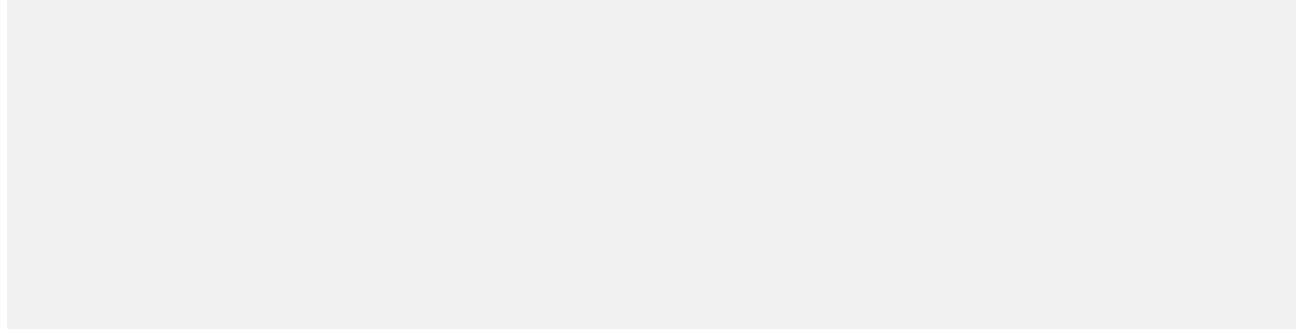
メモ: 組織で個人取引先が有効になっている場合は、法人取引先と個人取引先の 2 種類の取引先を使用できます。コードが を使用して新しい取引先を作成すると、法人取引先が作成されます。コードが を使用する場合、個人取引先が作成されます。

sObject で処理を実行する場合、最初にその sObject を特定のオブジェクトに変換することをお勧めします。次に例を示します。



次の例は、SOSL で取得したしたレコードのセットのオブジェクト型をどのように判定するかについて示しています。汎用 sObject レコードを取引先責任者、リード、または取引先に変換すると、項目をそれぞれ次のように変更できます。





## リレーションを使用した sObject 項目へのアクセス

sObject レコードは、ID と、関連付けられた sObject の表示を示すアドレスの 2 つの項目によって他のレコードとの関係を表します。たとえば、Contact sObject には種別が ID の \_\_\_\_\_ 項目と、関連付けられた sObject 自体を示す、種別が取引先の \_\_\_\_\_ 項目があります。

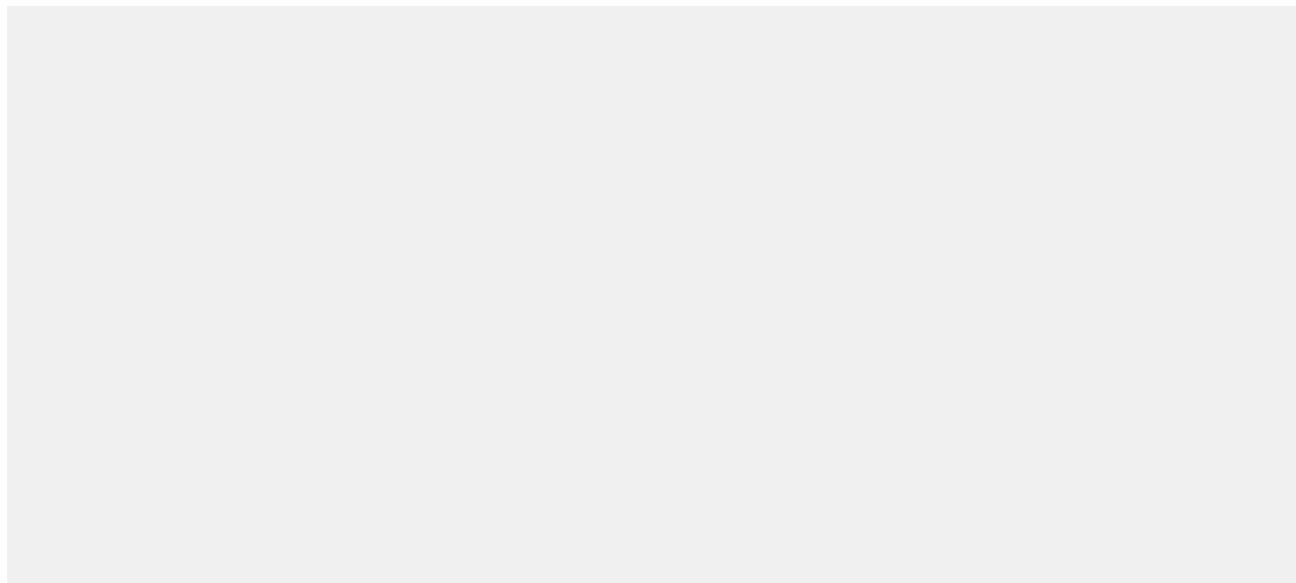
ID 項目を使用して取引先責任者と関連する取引先を変更したり、sObject 参照項目を使用して取引先のデータにアクセスしたりできます。参照項目は、SOQL クエリまたは SOSL クエリの結果としてのみ入力されます (下記参照)。

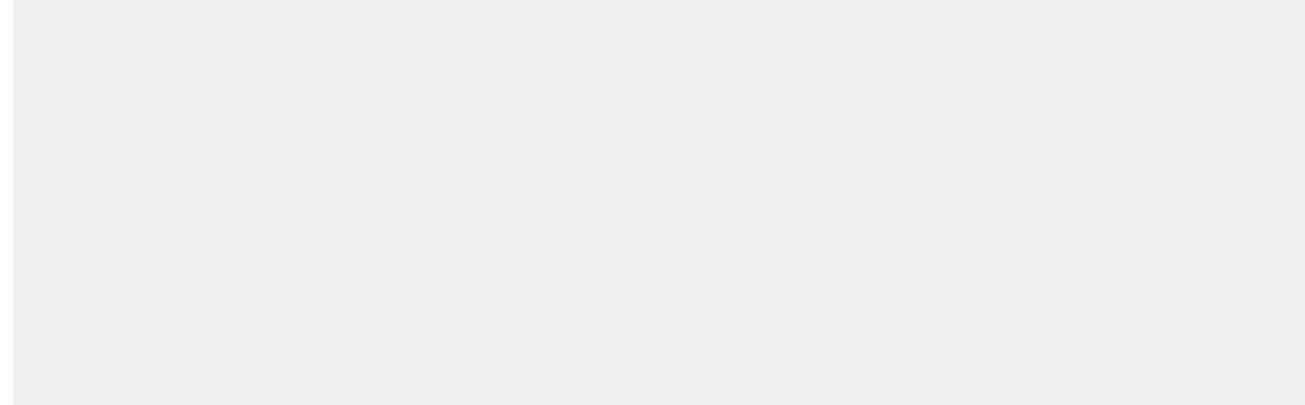
たとえば、次の Apex コードは、取引先と取引先責任者を相互に関連付ける方法と、取引先責任者を使用して取引先の項目を変更する方法を示します。



メモ: 次に示すのは最も複雑な例です。このコードで使用する一部の要素については、このガイドの後のセクションで説明します。

- \_\_\_\_\_ と \_\_\_\_\_ の詳細は、「[Insert ステートメント](#)」(ページ 396)および「[Update ステートメント](#)」(ページ 396)を参照してください。
- SOQL と SOSL の詳細は、「[SOQL および SOSL クエリ](#)」(ページ 83)を参照してください。

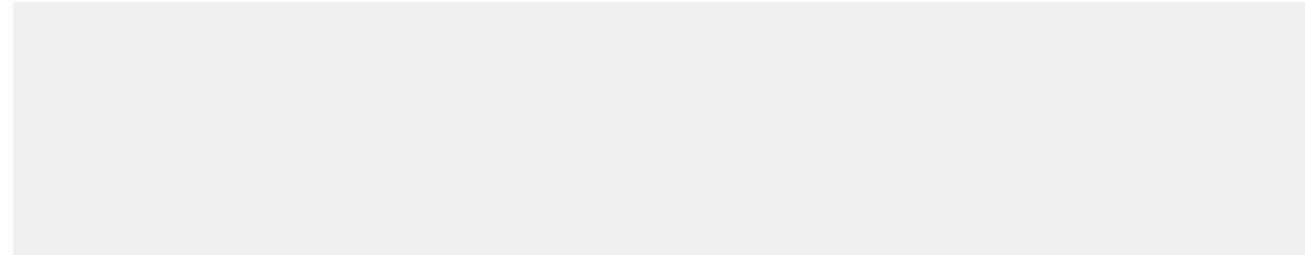




メモ: という式表現は、関係にまたがるその他の式と同様に、変更する場合と値として参照する場合では、若干異なる特徴があります。

- 値として参照する場合、 が null の場合 は と評価されますが、 は生成されません。これにより、開発者は null 値をチェックする必要なく多段の関係を参照できます。
- 変更するとき、 が null の場合、 を参照すると が生成されます。

また、sObject 項目キーは 、 または 時に、外部IDによる外部キー解決に使用されます。次に例を示します。

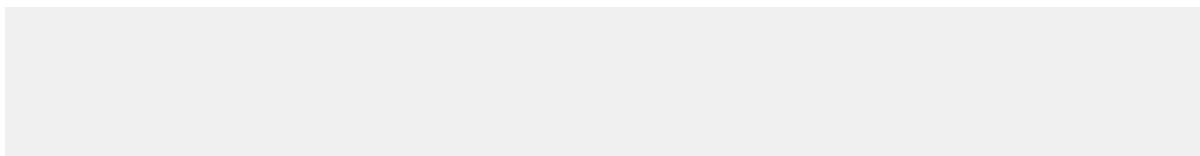


新しい取引先責任者に、 が「12345」である取引先と同じ を挿入します。そのような取引先がない場合、挿入は失敗します。



ヒント:

たとえば、次のコードは上記のコードと同一です。ただし、SOQL クエリを使用するため、上記のコードほど効率的ではありません。このコードが複数回コールされた場合、SOQL クエリ実行制限の最大数に達する場合があります。実行制限の詳細は、「[実行ガバナと制限について](#)」(ページ 337)を参照してください。



## sObjects と項目の検証

Apex コードの解析と検証を行うときにすべての sObject と項目参照が実際のオブジェクト名と項目名に照らして検証され、無効な名前が使用されている場合は、解析時の例外が発生します。

また、Apex パーサーは、埋め込み SOQL ステートメントや SOSL ステートメントおよびコードの構文で使用されるカスタムオブジェクトとカスタム項目を追跡します。これらの変更によって Apex コードが無効になる場合、プラットフォームは次のような変更をユーザが行えないようにします。

- ・ 項目名またはオブジェクト名の変更
- ・ あるデータ型から別のデータ型への変換
- ・ 項目またはオブジェクトの削除
- ・ 組織全体で行う、レコード共有、項目履歴管理、レコードタイプなどの変更

## Collections

Apex には、次の種類のコレクションがあります。

- ・ [Lists](#)
- ・ [Map](#)
- ・ [Set](#)



メモ: コレクションに保持できる項目の数に制限はありません。ただし、[ヒープサイズ](#)には制限があります。

### List

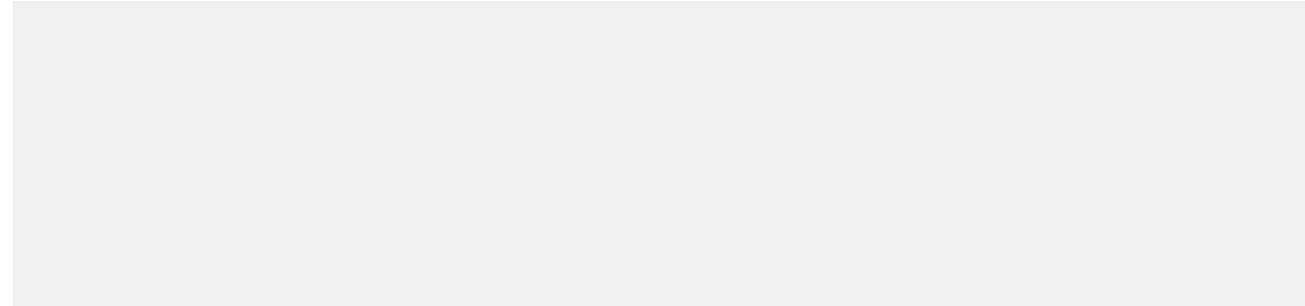
リストはプリミティブ型、sObjects、ユーザ定義のオブジェクト、Apex オブジェクトの順序付けられたコレクション、またはインデックスで識別される複数のコレクションです。たとえば、次の表は String のリストの視覚的な表示を示します。

インデックス 0	インデックス 1	インデックス 2	インデックス 3	インデックス 4	インデックス 5
'Red'	'Orange'	'Yellow'	'Green'	'Blue'	'Purple'

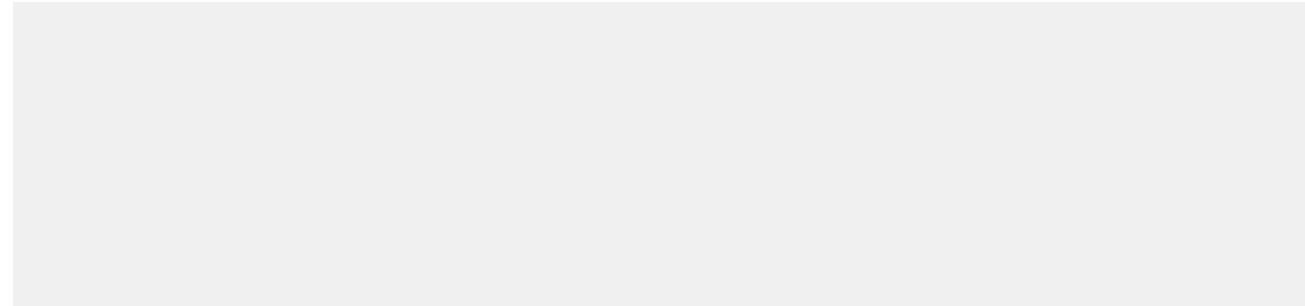
リスト内の最初の要素の位置は必ず 0 になります。

リストにはどのコレクションでも含めるられるため、相互にネストでき、多次元的に構築できます。たとえば、Integer セットのリストのリストを作成できます。リストでは、コレクションを最大 4 レベルまでネストできます。

リストを宣言するには、<> 文字で囲まれたプリミティブデータ型、sObject、ネストされたリスト、対応付け、または設定された要素の種類の前に `list` キーワードを使用します。次に例を示します。



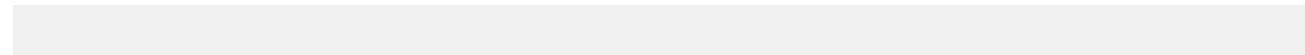
リストの要素にアクセスするには、Apex が提供するシステムメソッドを使用します。次に例を示します。



サポートされるすべてのメソッドなどの詳細は、「[List メソッド](#)」(ページ 496)を参照してください。

#### プリミティブまたは sObject の一次元リストの配列表記の使用

プリミティブまたは sObjects の一次元リストを使用する場合、従来の配列表記を使用してリスト要素を宣言および参照することもできます。たとえば、次のようにデータ名または sObject 型名の後に [] 文字を挿入して、プリミティブまたは sObject の一次元的リストを宣言することができます。



プリミティブデータ型または sObject の一次元リストの要素を参照するには、リスト名の後に要素のインデックス位置を大かっこで囲んで表記することもできます。次に例を示します。



すべてのリストは `list` に初期設定されます。リストには値を割り当て、リテラル表記を使用してメモリを割り当てるすることができます。次に例を示します。

例	説明
	要素のない Integer リストを定義します。

例	説明
	要素のない Account リストを定義します。
	メモリが 6 つの Integer に割り当られた Integer リストを定義します。
	3 つの取引先にメモリが割り当てられた取引先リストを定義します。最初の位置に新しい取引先オブジェクト、2 番目の位置に _____ 、3 番目の位置に別の新しい取引先オブジェクトが割り当てられます。
	新しいリストで取引先責任者リストを定義します。

### sObjects のリスト

Apex は、リストがデータ操作言語 (DML) ステートメントによってデータベースに正常に挿入または更新されると、sObject リストの各オブジェクトに ID を自動的に生成します。従って、sObjects のリストに同じ sObject が複数含まれる場合は、それが \_\_\_\_\_ ID を持つ場合であっても、sObject のリストの挿入や更新は行えません。この場合、2 つの ID がメモリ内の同じ構造に書き込まれなければならないことになり、これは不正処理となります。

たとえば、次のコードブロックの  
を挿入しようとするため、

ステートメントは、同じ sObject ( )への 2 つの参照が含まれるリスト  
を生成します。

DML ステートメントの詳細は、「[DML ステートメント](#)」(ページ 396)を参照してください。

汎用 sObject データ型にリストを使用することができます。リストの汎用インスタンスを作成することもできます。

### リストの並び替え

メソッドを使用して、プリミティブデータ型、  
データ型(Apex クラス)、SelectOption 要素、sObject(標準オブジェクトとカスタムオブジェクト)のリストを並び  
替えできます。

プリミティブデータ型の場合、並び替えは昇順です。

カスタムデータ型の場合、並び替えの条件と並び替え順は、  
インターフェースを実装するカスタム  
データ型によって異なります。独自のクラスの  
インターフェースの実装についての詳細は、  
[「インターフェース」](#)を参照してください。

sObject の場合、並び替えは昇順で、次のセクションで説明する一連の比較ステップを使用します。ただし、  
[「sObject のカスタム並び替え順」](#)に示されるように、sObject を Apex クラスでラップして  
インターフェースを実装することで、sObject のカスタム並び替え順を実装することもできます。

SelectOption の場合、並び替えは値項目と表示ラベル項目に基づく昇順です。SelectOption で使用する比較ステップの順序は、[「SelectOption のデフォルトの並び替え順」](#)を参照してください。

### sObject のデフォルトの並び替え順

メソッドは sObject を昇順で並び替え、一連の順序付けられたステップに従って sObject を比較します。これらのステップには比較に使用される表示ラベルまたは項目が規定されています。比較は最初のステップから開始され、規定の表示ラベルまたは項目を使用して 2 つの sObject が並び替えられたときに終了します。使用される比較の順序は次のようになります。

#### 1. sObject 型の表示ラベル。

たとえば、Account sObject は Contact の前になります。

#### 2. Name 項目(該当する場合)。

たとえば、リストに A と B という名前の 2 つの取引先がある場合、取引先 A が取引先 B よりも前になります。

#### 3. 標準項目。ID 項目と Name 項目を除き、アルファベット順で最初の項目から使用されます。

たとえば、2 つの取引先が同じ名前の場合、並び替えに使用される最初の標準項目は AccountNumber です。

#### 4. カスタム項目。アルファベット順で最初の項目から使用されます。

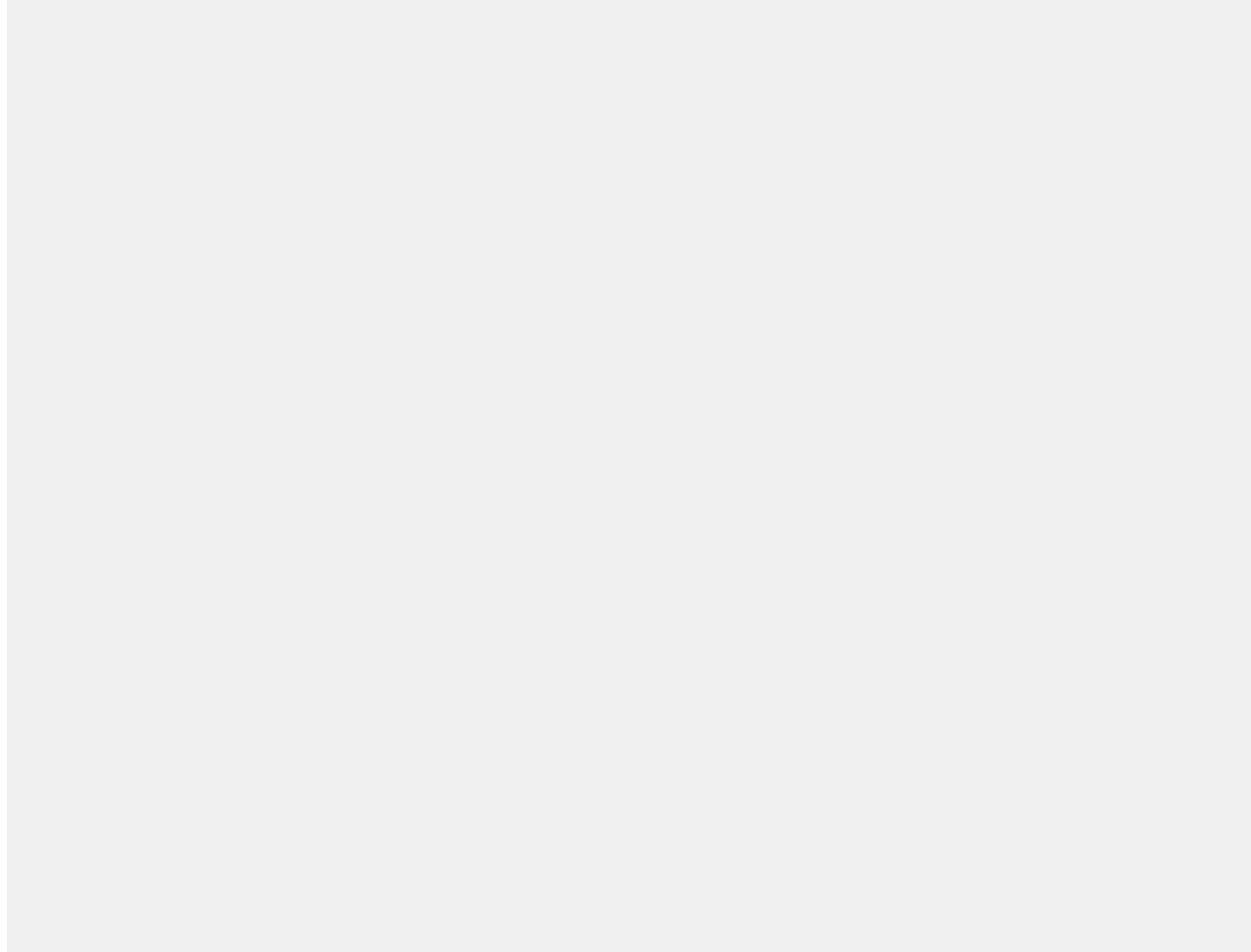
たとえば、2 つの取引先の名前と標準項目が同じで、FieldA と FieldB という 2 つのカスタム項目がある場合、  
並び替えでは FieldA の値が最初に使用されます。

この一連のすべてのステップが実行されると限ではありません。たとえば、リストに同じ種別で一意の Name 値がある  
2 つの sObject が含まれる場合、これらは Name 項目に基づいて並び替えられ、並び替えはステップ 2 で停止

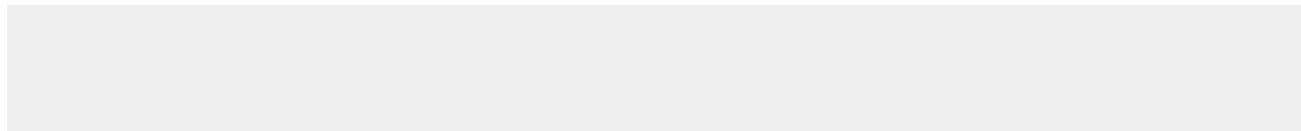
します。別の例で、名前が同じか sObject に Name 項目がない場合、並び替えはステップ 3 まで進み、標準項目を基準に並び替えられます。

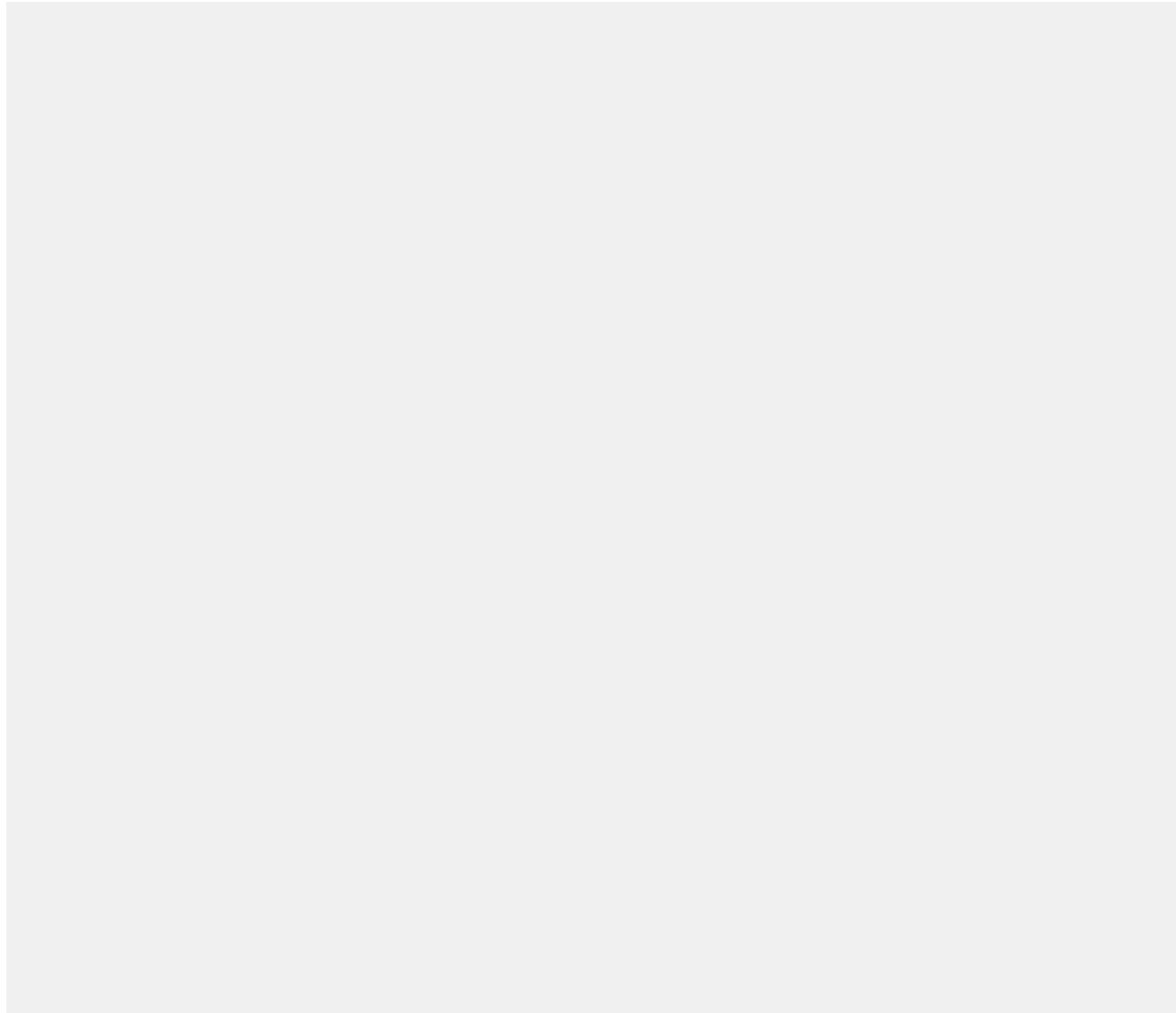
テキスト項目の場合、並び替えアルゴリズムは Unicode 並び替え順を使用します。また、空の項目は並び替え順で空でない項目より前にります。

次の例は、Account sObject のリストの並び替えです。この例では、Name 項目が使用されて、リスト内で Acme 取引先が 2 つの sForce 取引先より前に配置されることを示します。sForce という名前の取引先が 2 つあり、アルファベット順で Industry 項目は Site 項目より前になるため、Industry 項目が残りの取引先の並び替えに使用されます。



次の例は前の例と同様ですが、Merchandise\_\_c カスタムオブジェクトを使用する点が異なります。この例では、Name 項目が使用されて、リスト内で Notebooks 商品が Pens より前に配置されることを示します。Name 項目値が Pens の商品 sObject が 2 つあり、アルファベット順で Description 項目は Price および Total\_Inventory 項目より前になるため、Description 項目が残りの商品品目の並び替えに使用されます。

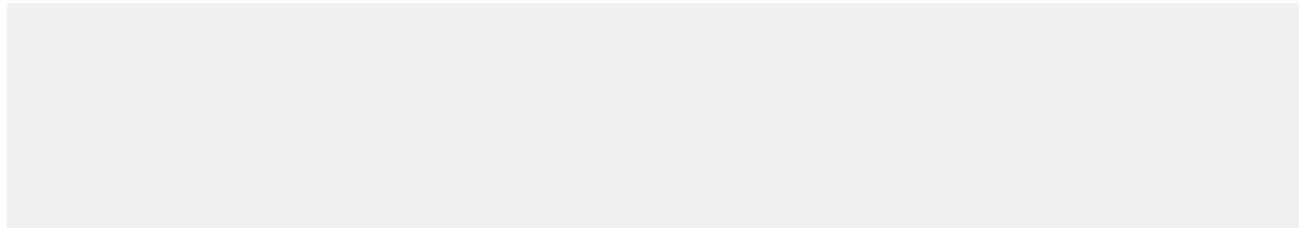


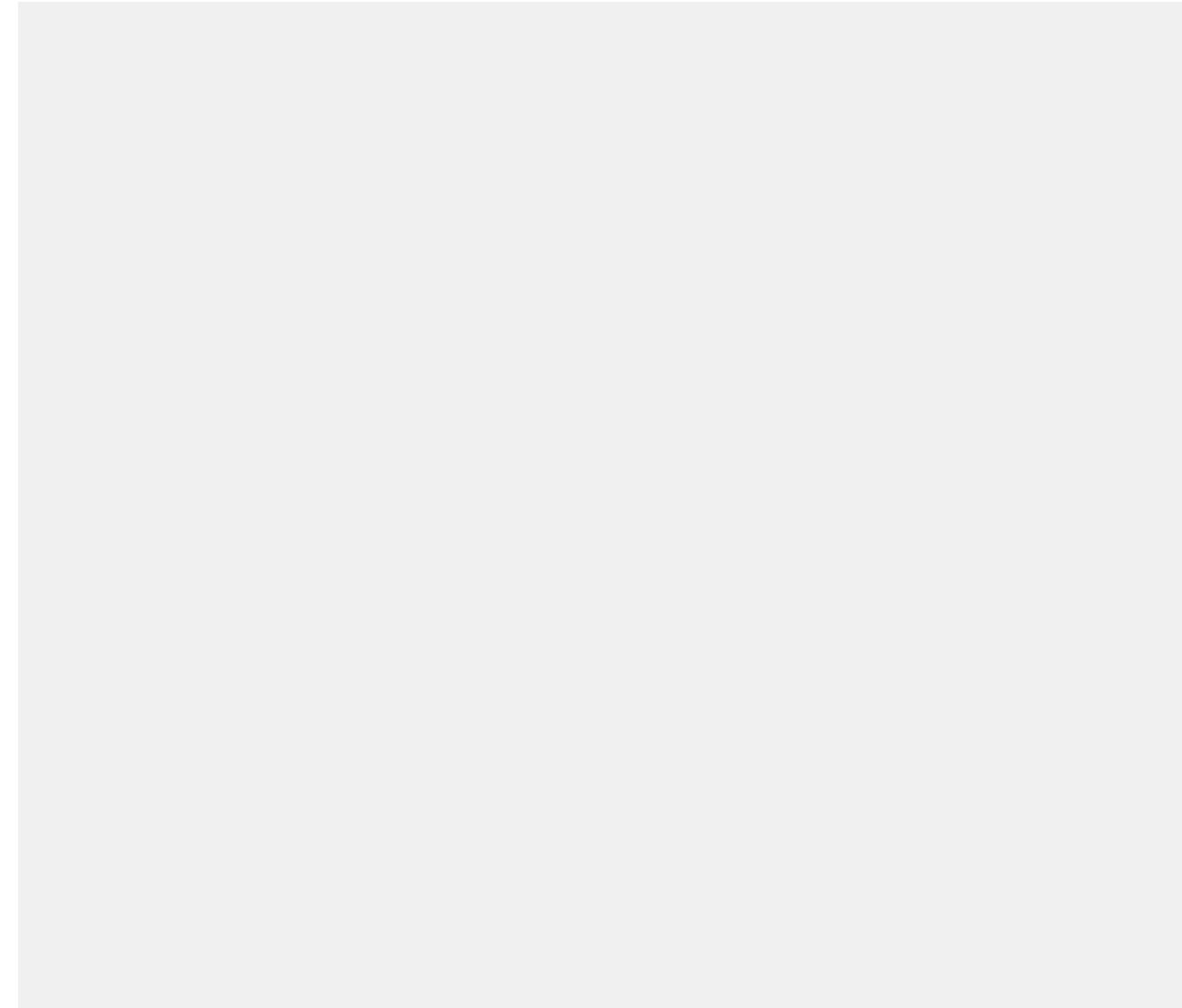


### sObject のカスタム並び替え順

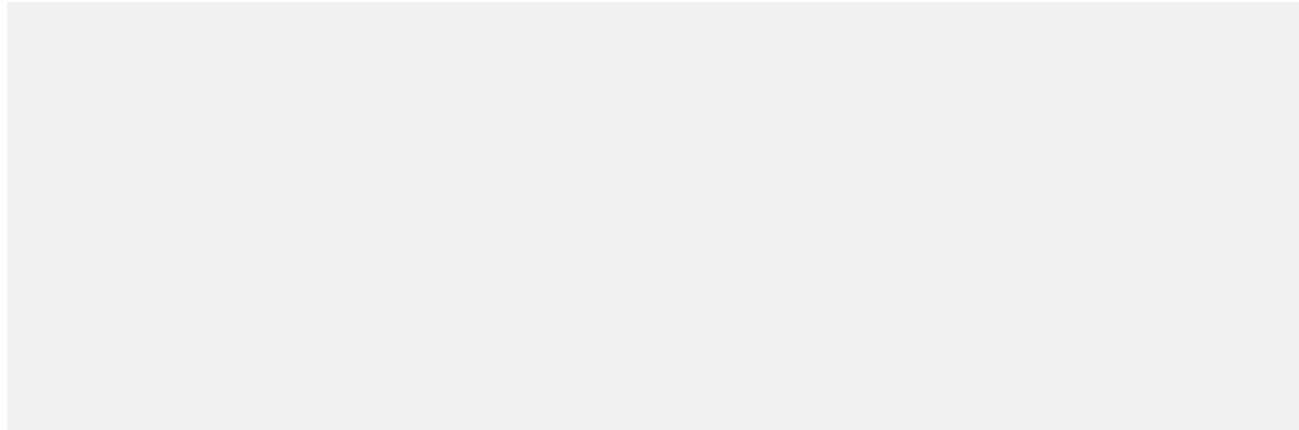
リストに sObject のカスタム並び替え順を実装するには、sObject のラッパークラスを作成し、  
インターフェースを実装します。ラッパークラスに対象の sObject を含め、並び替えロジックを指定する  
メソッドを実装します。

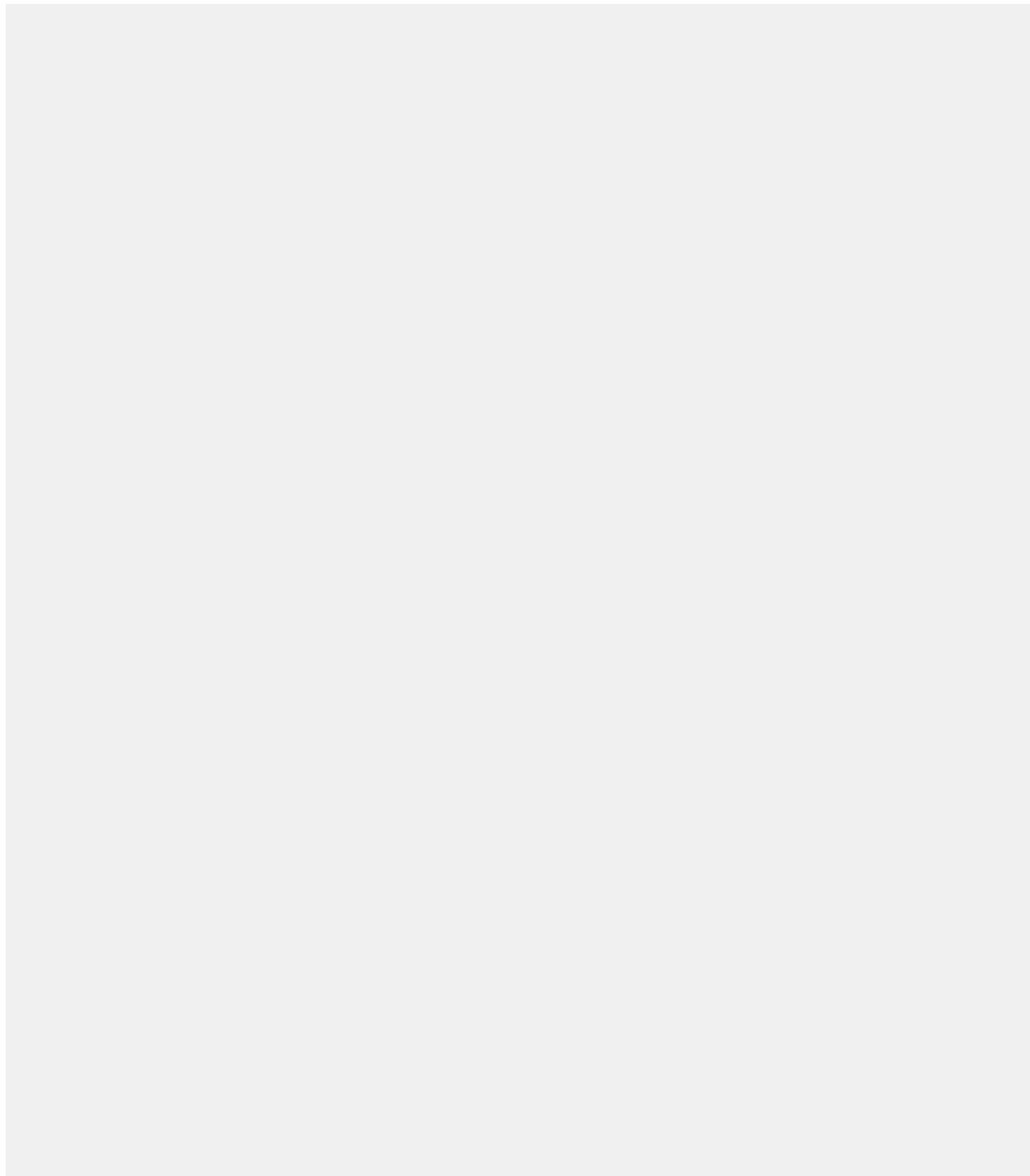
次の例では、Opportunity のラッパークラスを作成する方法を示します。このクラスの  
メソッドの実装では、Amount 項目 (このインスタンスに含まれるクラスメンバー変数)、およびメソッドに渡された商談オブ  
ジェクトに基づいて 2 つの商談を比較します。





次の例には、クラスのテストが含まれています。オブジェクトのリストを並び替え、リストの要素が商談金額を基準に並び替えられていることを確認します。





### SelectOption のデフォルトの並び替え順

メソッドは、この比較順序に基づいて、値項目と表示ラベル項目を使用して昇順で SelectOption 要素を並び替えます。

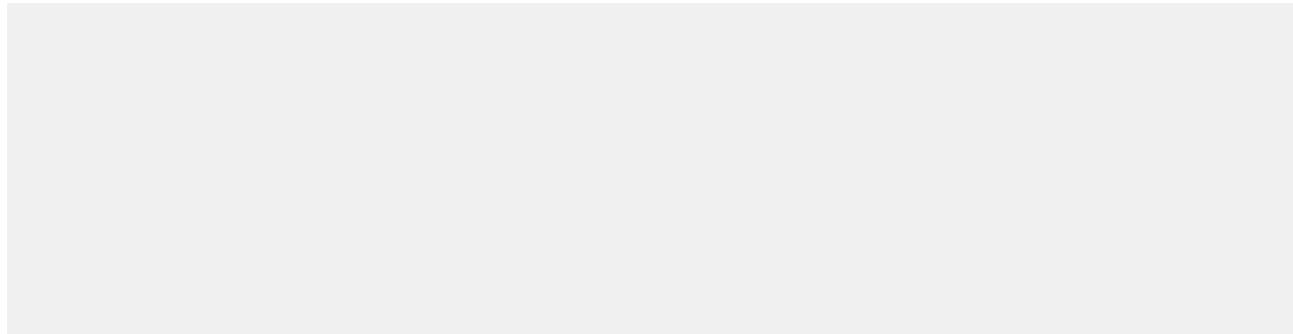
1. 並び替えには最初に値項目が使用されます。

2. 2つの値項目が同じ値の場合、または両方とも空の場合、表示ラベル項目が使用されます。

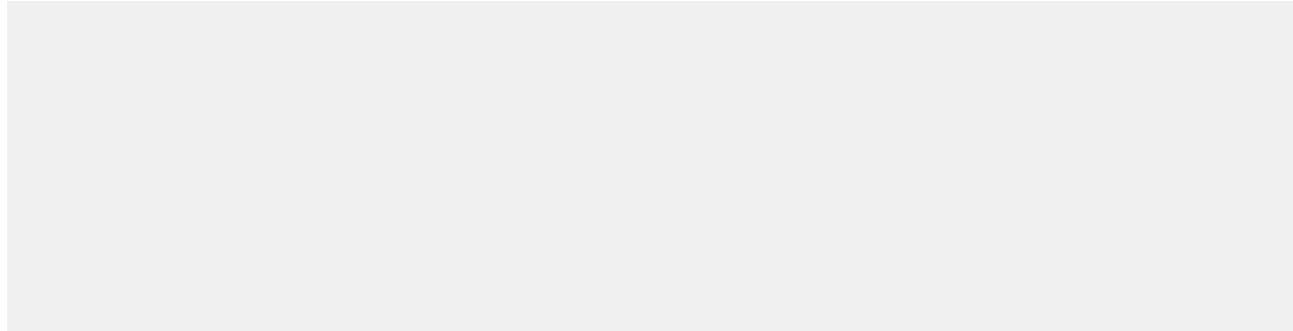
無効になっている項目は並び替えには使用されません。

テキスト項目の場合、並び替えアルゴリズムは Unicode 並び替え順を使用します。また、空の項目は並び替え順で空でない項目より前にります。

次の例では、リストに 3 つの SelectOption 要素が含まれています。United States と Mexico の 2 つの要素には同じ値項目 ('A') があります。メソッドは表示ラベル項目に基づいて 2 つの要素を並び替え、出力に示されるように Mexico を United States より前に配置します。並び替えられたリストの最後の要素は Canada で、その値項目 'C' は 'A' より後になります。



これは debug ステートメントの出力です。並び替え前後のリストの内容が示されています。



## Set

セットは、重複を含まない要素の順序付けされていないコレクションです。セットの要素には、プリミティブ型、コレクション型、sObject 型、ユーザ定義型、組み込み Apex 型のいずれかのデータ型を使用できます。たとえば、次の表は都市名を使用する文字列のセットを示します。

'San Francisco'	'New York'	'Paris'	'Tokyo'
-----------------	------------	---------	---------

セットには相互にネストすることが可能なコレクションを含めることができます。たとえば、Integer セットのリストのセットを作成できます。セットでは、コレクションを最大 4 レベルまでネストできます。

セットを宣言するには、<> 文字で囲まれたプリミティブデータ型名の前に キーワードを使用します。次に例を示します。

次のようにして、セットを宣言し、入力します。

セットの要素にアクセスするには、Apex が提供するシステムメソッドを使用します。次に例を示します。

ただし、取引先の 1 つに説明を追加すると、その取引先は一意であるとみなされます。

ユーザ定義型のオブジェクトの一意性は、クラスで提供する メソッドと メソッドによって判断されます。

サポートされるすべてのセットシステムメソッドなどの詳細は、「[Set メソッド](#)」(ページ 512)を参照してください。

セットについて、次の点に注意してください。

- Java と異なり、Apex 開発者は、宣言でセットを実装するために使用するアルゴリズム( または など)を参照する必要がありません。Apex は、すべてのセットにハッシュ構造を使用します。
- セットは、順序付けられていないコレクションです。セットの結果が返される順序に依存しないでください。セットによって返されるオブジェクトの順序は、警告なく変更される場合があります。

## 対応付け

対応付けは、単一の値に一意のキーを対応付ける、キー - 値のペアのコレクションです。キーと値には、プリミティブ型、コレクション型、sObject 型、ユーザ定義型、組み込み Apex 型のいずれかのデータ型を使用できます。たとえば、次の表は国名と通貨の対応付けを示します。

国(キー)	'United States'	'Japan'	'France'	'England'	'India'
通貨(値)	'Dollar'	'Yen'	'Euro'	'Pound'	'Rupee'

対応付けのキーと値にはどのコレクションでも含めることができます。コレクションをネストすることができます。たとえば、各種の対応付けに Integer を対応付けることができ、その対応付けによって String がリストに対応付けられます。対応付けのキーでは、コレクションを最大 4 レベルまでネストできます。

対応付けを宣言するには、<> 文字で囲まれたキーおよび値のデータ型の前に キーワードを使用します。次に例を示します。

汎用 sObject データ型に対応付けを使用できます。対応付けの汎用インスタンスを作成することもできます。

リスト同様、中かっこ( )構文を使用して対応付けを宣言する場合、対応付けのキー - 値のペアを入力できます。中かっこの中で、キーを最初に指定し、 を使用してそのキーの値を指定します。次に例を示します。

最初の例で、キー の値は 、キー の値は です。2番目の例で、キー にはリスト の値が含まれます。これは、sObject を対応付けのキーとして使用した例です。

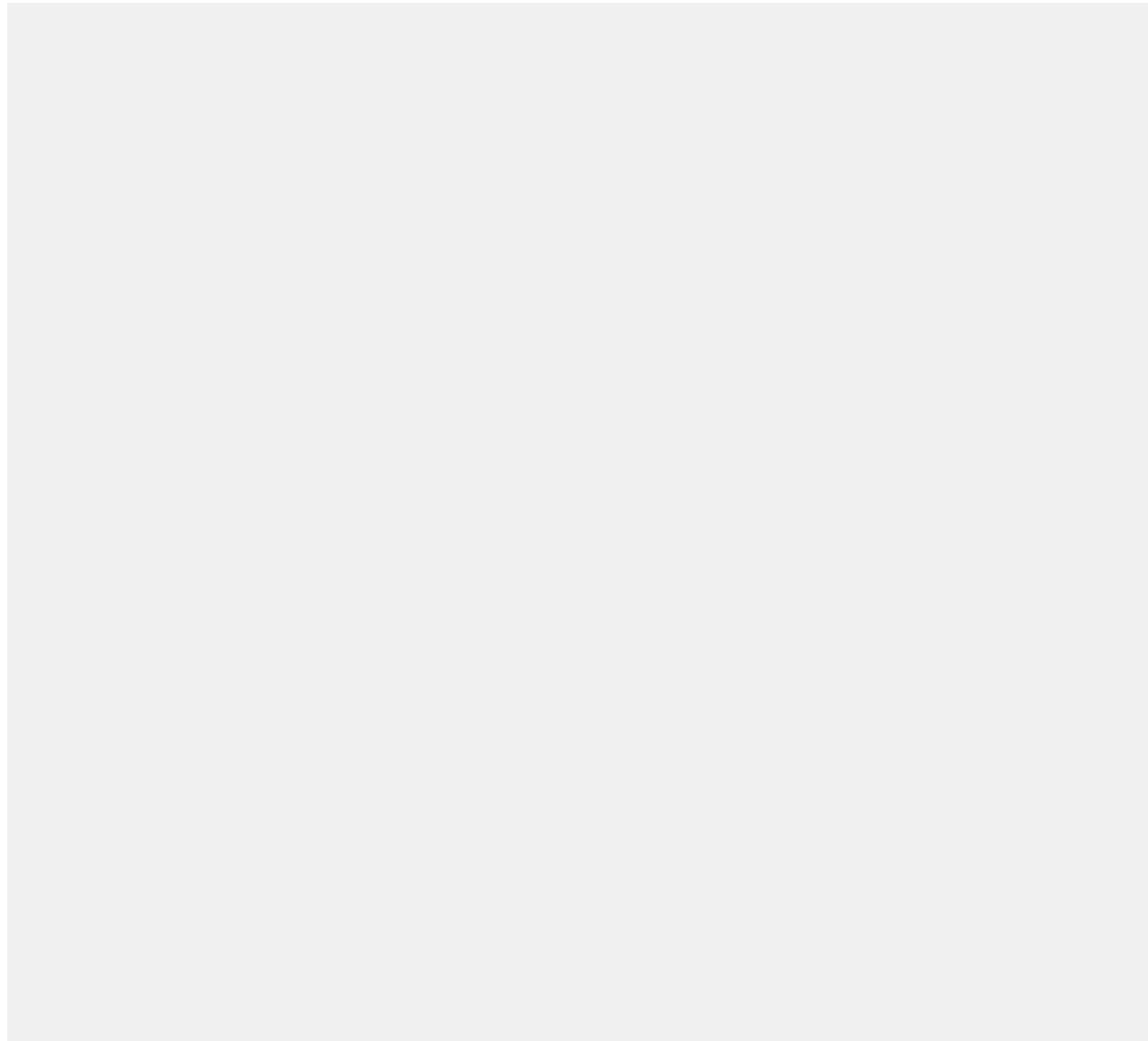
対応付けの要素にアクセスするには、Apex が提供する Map メソッドを使用します。たとえば、次のようになります。

サポートされるすべての Map メソッドなどの詳細は、「[Map メソッド](#)」(ページ 505)を参照してください。

#### 対応付けの考慮事項

- Java と異なり、Apex 開発者は、宣言に対応付けを実装するために使用するアルゴリズム( または )を参照する必要がありません。Apex は、すべての対応付けにハッシュ構造を使用します。
- 対応付けの結果が返される順序に依存しないでください。対応付けによって返されるオブジェクトの順序は、警告なく変更される場合があります。対応付け要素には、常にキーでアクセスします。
- 対応付けのキーは、 値を保持できます。
- 対応付けの既存のキーと一致するキーを含む対応付けエントリを追加すると、そのキーを含む既存のエントリが新しいエントリで上書きされます。
- String 型の対応付けキーでは、大文字と小文字が区別されます。大文字と小文字のみが異なる 2 つのキーは一意であるとみなされ、それぞれに別個の対応付けエントリがあります。したがって、 、 、 、 および などの Map メソッドでは、これらのキーが別個のものとして処理されます。
- 対応付けのユーザ定義型キーの一意性は、クラスで提供する **メソッド** と **メソッド** によって判断されます。sObject キーなど、他のすべての非プリミティブ型のキーの一意性は、オブジェクト項目の値の比較によって判断されます。
- sObject を対応付けのキーとして使用する場合には、注意が必要です。sObject のキーの照合は、すべての sObject 項目値の比較に基づいています。sObject を対応付けに追加した後で 1 つ以上の項目値が変更した場合に、この sObject を対応付けから取得しようとすると、 が返されます。これは、項目値が異なるので変更後の sObject が対応付けで見つからないためです。sObject で項目を明示的に変更するか、sObject の挿入後

に sObject 変数の ID 項目が自動入力されるなど、sObject 項目がシステムによって暗黙的に変更された場合に、この状況が発生します。次の例に示すように、操作の前に追加された対応付けからこのオブジェクトを取得しようとしても、対応付けエントリは生成されません。



たとえば、sObject の before insert および after insert トリガを使用する場合に、sObject 項目の自動入力がトリガに含まれるという別のシナリオもあります。これらのトリガではクラスで定義された静的対応付けを共有しており、の sObject が before トリガでこの対応付けに追加されると、自動入力される項目で 2 つのセットの sObject が異なるため、after トリガにある の sObject は検出されません。after トリガにある の sObject には、挿入後に入力されるシステム項目 (ID、CreatedDate、CreatedById、LastModifiedDate、LastModifiedById、および SystemModStamp) が含まれます。

## sObject 配列からの対応付け

ID または String データ型から sObject への対応付けは、sObjects のリストから初期設定できます。オブジェクトの ID (null 以外の重複しない値) は、キーとして使用されます。この種の対応付けは一般的に、メモリ内での 2 つのテーブルの「結合」に使用します。たとえば、この例では ID と取引先責任者の対応付けを読み込みます。

この例では、SOQL クエリは 項目と 項目を含む取引先責任者のリストを返します。演算子はリストを使用して対応付けを作成します。詳細は、「[SOQL および SOSL クエリ](#)」(ページ 83)を参照してください。

## パラメータ化された型

Apex は通常、静的なプログラム言語で、ユーザは変数を使用する前に変数のデータ型を指定する必要があります。たとえば、Apex では次の変数は適切です。

が始めに定義されていない場合、次の変数は正しくありません。

リスト、対応付けおよびセットは Apex でパラメータ化されます。Apex が引数としてサポートするデータ型を取ります。このデータ型は、リスト、対応付け、またはセットの構造時に実際のデータ型と置き換える必要があります。たとえば、次のようにになります。

## パラメータ化されたリストによる再分類

Apex では、型 `g` の下位型である場合、`g` は `g` の下位型となります。たとえば、次の例は有効です。

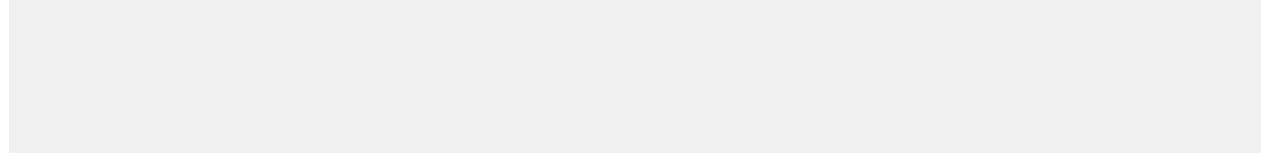
## 対応付けのキーとセットでのカスタムデータ型の使用

対応付けのキーまたはセット要素にカスタムデータ型 (Apex クラス) を使用する場合、クラスで メソッドと メソッドを提供します。Apex はこの 2 つのメソッドを使用して、オブジェクトのキーの等価と一意性を判断します。

### クラスへの `equals` メソッドと `hashCode` メソッドの追加

カスタムデータ型の対応付けのキーが適切に比較され、その一意性が一貫して認識されるようにするために、クラスで次の 2 つのメソッドの実装を提供します。

- 署名付きの メソッドを次に示します。

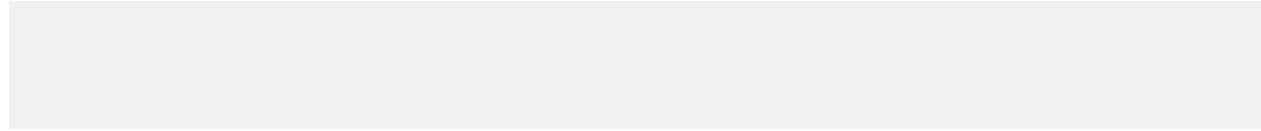


メソッドの実装時には、次の点に留意してください。クラスの x、y、z が null 以外のインスタンスである場合、メソッドは次の条件を満たす必要があります。

- 反射性:
- 対称性: は、 が を返す場合にのみ を返す
- 推移性: が を返し、かつ が を返す場合、 は を返す
- 整合性: の複数の呼び出しで常に を返すか常に を返す
- null 以外の参照値 x では、 は を返す

Apex の メソッドは、[Java の equals メソッド](#)に基づいています。

- 署名付きの メソッドを次に示します。

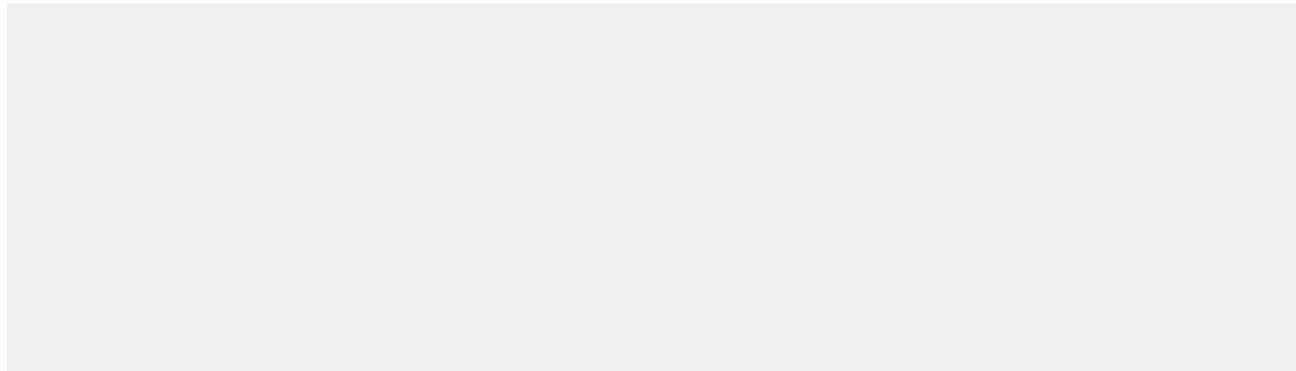


メソッドの実装時には、次の点に留意してください。

- メソッドが Apex 要求の実行中に同じオブジェクトで複数回呼び出された場合、同じ値を返す必要がある
- メソッドで 2 つのオブジェクトが等価とされた場合、 は同じ値を返す必要がある
- メソッドで 2 つのオブジェクトが等価でないとされた場合、 は異なる値を返す必要はない

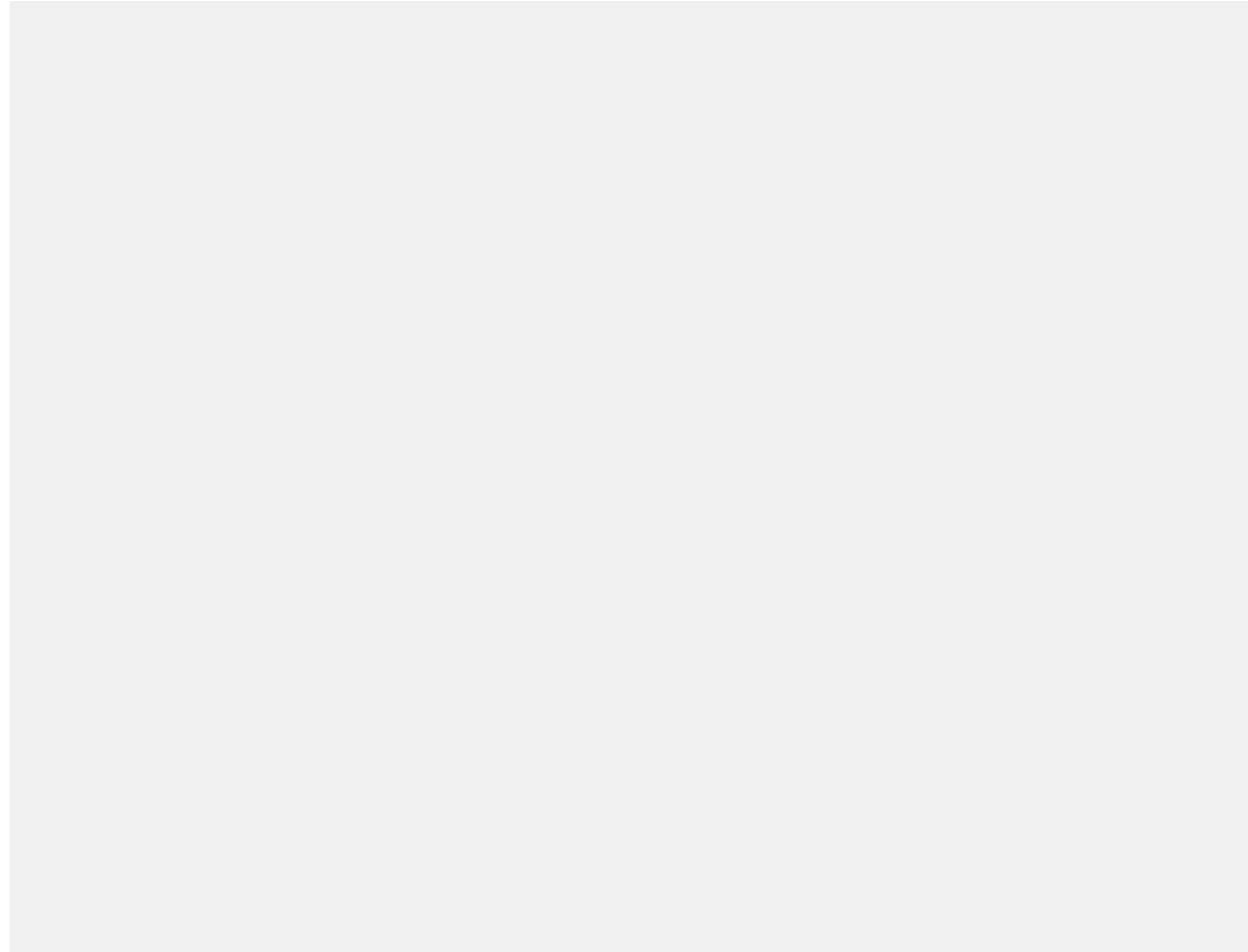
Apex の メソッドは、[Java の hashCode メソッド](#)に基づいています。

クラスで メソッドを提供することによる別の利点は、オブジェクトの比較が簡素化される点です。オブジェクトの比較に 演算子または メソッドを使用できます。たとえば、次のようにになります。

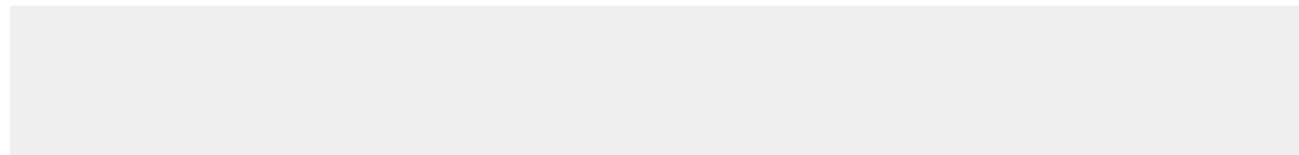


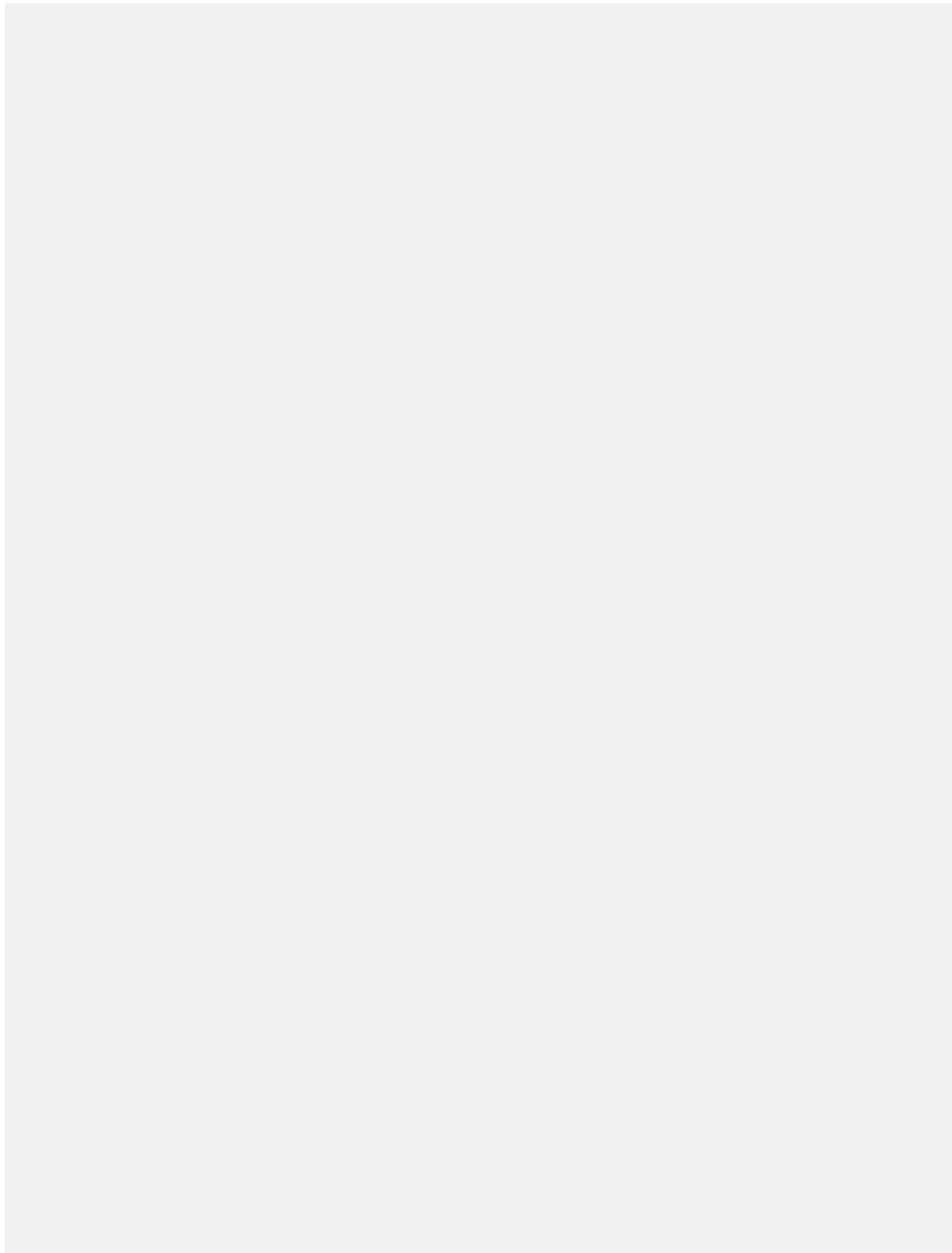
## サンプル

このサンプルでは、**メソッド**と**メソッド**の実装方法を示します。これらのメソッドを提供するクラスが最初に表示されています。2つの Integer を取るコンストラクタも含まれています。2番目のサンプルはコードスニペットで、このクラスの3つのオブジェクトを作成し、そのうち2つは値が同じです。次に、ペアオブジェクトをキーとして使用して対応付けエントリが追加されます。最後に追加されたエントリには最初のエントリと同じキーが含まれているため、最初のエントリが上書きされ、サンプルでは対応付けに2つのエントリのみが存在することが確認されます。次に、**演算子**を使用します。クラスは**を実装するため、この演算子は期待どおりに機能します。**また、対応付けに特定のキーが含まれているかどうかの確認、デバッグログへのすべてのキーと値の書き込みなど、他のいくつかの対応付け操作が実行されます。最後に、**セットを作成してそのセットに同じオブジェクトを追加します。**3つのオブジェクトのうち2つのみが一意であるため、**セットのサイズが2であることを確認します。**



このコードスニペットは **クラス**を使用します。







## コレクションの繰り返し処理

コレクションはリスト、セット、または対応付けで構成されます。コレクションの繰り返し処理中にコレクションの要素を変更することはできません。変更するとエラーが発生します。要素を含むコレクションの繰り返し処理中に、要素を直接追加したり、削除したりしないでください。

### 繰り返し処理中の要素の追加

リスト、セットまたは対応付けの繰り返し処理中に要素を追加するには、新しい要素を一時的なリスト、セット、または対応付けに保存し、コレクションの処理が完了した後で元のコレクションに追加します。

### 繰り返し処理中の要素の削除

リストの繰り返し処理中に要素を削除するには、新しいリストを作成し、保存する要素をコピーします。または、削除する要素を一時的なリストに追加して、コレクションの処理が完了した後で削除することもできます。



#### メモ:

メソッドは線形的に処理を実行します。このメソッドを使用して要素を削除する場合は、時間とリソースを要します。

対応付けまたはセットの繰り返し処理中に要素を削除するには、削除するキーを一時的なリストに保存し、コレクションの処理が完了した後で削除します。

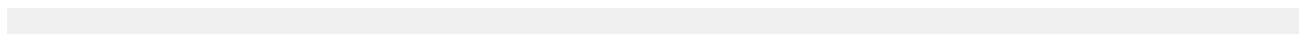
## enum

列挙型は、ユーザが指定した識別子の有限のセットのうちの1つだけを値に持つ抽象データ型です。列挙型は通常、一組のトランプや特定の季節など、番号付けされた順序を持たない使用可能な値のセットを定義します。列挙型の各値は一意の整数値に対応しますが、その整数値を演算処理の実行に使用するなど、ユーザが間違って使用するのを防ぐため、列挙型はこの実装を非表示にします。列挙型を作成した後、変数、メソッド引数、戻り値をこのデータ型として宣言できます。

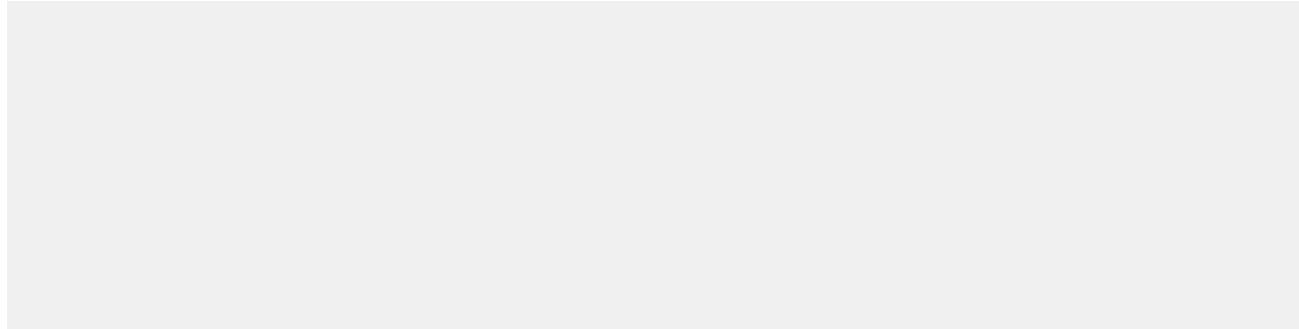


メモ: Java と異なり、列挙型自体にはコンストラクタ構文はありません。

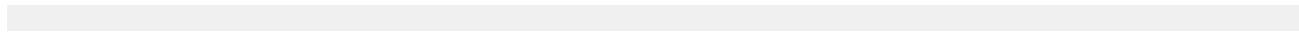
列挙型を定義するには、宣言で `enum` キーワードを使用し、値のリストを中かっこで区切れます。たとえば、次のコードは `Color` という列挙型を作成します。



列挙型を作成すると、という新しいデータ型も作成されます。この新しいデータ型は、他のデータ型と同じように使用できます。次に例を示します。



クラスを列挙型として定義することもできます。列挙型クラスを作成する場合、定義ではキーワードを使用しません。

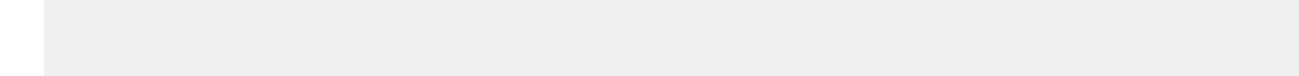


列挙型は、他のデータ型名を使用できるすべての場所で使用できます。列挙型の変数を定義する場合、その変数に割り当てるオブジェクトはその列挙型クラスのインスタンスである必要があります。

メソッドは列挙型を署名の一部として使用できます。この場合、関連付けられた WSDL ファイルには列挙型とその値の定義が含まれ、API クライアントはその定義を使用できます。

Apex には、次のシステム定義の列挙型があります。

- すべての API 演算子の WSDL ドキュメントに公開される API エラーコードに対応します。次に例を示します。



状況コードの完全なリストは、組織の WSDL ファイルから入手できます。組織の WSDL ファイルへのアクセスについての詳細は、Salesforce オンラインヘルプの「Salesforce WSDL およびクライアント認証証明書のダウンロード」を参照してください。

- : メソッドから返される結果 XML の解析に使用する XML タグのリストを返します。詳細は、「[クラス](#)」(ページ 885)を参照してください。
- : この列挙型は メソッドと共に使用して、すべての コールのログレベルを指定します。詳細は、「[System メソッド](#)」(ページ 609)を参照してください。
- : Decimal メソッドおよび Double メソッドなど、数学的演算を実行して演算の丸め動作を指定するメソッドで使用されます。詳細は、「[丸めモード](#)」(ページ 450)を参照してください。

- Field Describe Result のメソッドによって返されます。詳細は、「[Schema.SOAPType Enum 値](#)」(ページ 548)を参照してください。
- Field Describe Result のメソッドによって返されます。詳細は、「[Schema.DisplayType Enum 値](#)」(ページ 543)を参照してください。
- この enum は JSON コンテンツの解析に使用されます。詳細は、「[Enum](#)」(ページ 590)を参照してください。
- Visualforce メッセージの重要度を指定します。詳細は、「[ApexPages.Severity Enum](#)」(ページ 869)を参照してください。
- DOM ドキュメントのノードタイプを指定します。詳細は、「[ノードの種類](#)」(ページ 898)を参照してください。



メモ: システム定義の列挙型は Web サービスメソッドで使用できません。

システム列挙型を含むすべての列挙型の値には、共通メソッドが関連付けられています。詳細は、「[Enum メソッド](#)」(ページ 517)を参照してください。

ユーザ定義のメソッドは列挙型の値に追加できません。

## 変換の規則について

通常、Apex では、あるデータ型を別のデータ型に変換する場合、明示的に行う必要があります。たとえば、Integer データ型の変数を暗黙的に String に変換することはできません。メソッドを使用する必要があります。ただし、一部のデータ型はメソッドを使用せず暗黙的に変換できます。

Number はデータ型の階層です。下位の数値型の変数は常に、明示的に変換せずに、より高位のデータ型に割り当することができます。次に示すのは数値の階層です(下位から上位の順)。

1. Integer
2. Long
3. Double
4. Decimal



メモ: 値が下位のデータ型数値から上位のデータ型数値に渡されると、その値は数値の上位のデータ型に変換されます。

この階層と暗黙的な変換は、Java の数値階層とは異なります。Java の数値階層では基本のインターフェース数値が使用され、オブジェクトの暗黙的変換は行われません。

数値の他にも、暗黙的に変換できるデータ型があります。以下の規則が適用されます。

- ID は常に String に割り当てることができる。
- String を ID に割り当てることができる。ただし、実行時、値が正当な ID であることを確認します。正当でない場合、実行時例外が発生します。
- いつでも キーワードを使用して文字列が ID かどうかをテストできる。

## データ型に関するその他の考慮事項

### 数値のデータ型

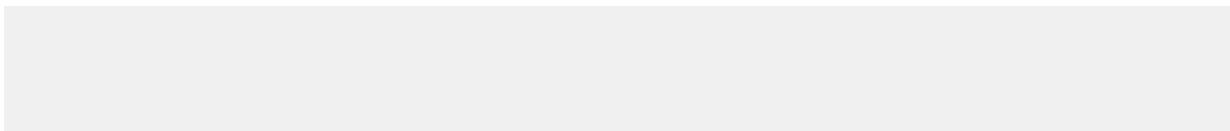
数値は、Long の L または Double または Decimal の .0 が追加されていない限り、Integer 値です。たとえば、式 `d = 123L` は、d という Long 型の変数を宣言し、Integer 型の数値(123)に割り当てて明示的に Long 型に変換されます。右側の Integer 型の値は、Integer の範囲内にあるため割り当てに成功しますが、右側の数値が Integer 型の最大値を超える場合、コンパイルエラーが発生します。この場合、数値に L を追加することによって、より範囲の広い Long 型の値にします。これは `d = 123L` のように表します。

### データ型の値のオーバーフロー

現在の型の最大値よりも大きな値を生成する演算をオーバーフローと言います。たとえば、式 `int result = 2147483647 + 1;` では、2147483647 が Integer の最大値であり、それに 1 を加算することで Integer の負の最小値 -2147483648 に戻されてしまうため、値 -2147483648 となります。

演算によって現在の型の最大値よりも大きな結果が生成される場合、最大値を超える計算値がオーバーフローしてしまうため、最終結果は不正な値となります。たとえば、式

は、右側の Integer の生成値が最大値を超えてオーバーフローするため、不正な結果となります。そのため、最終的な値は予想されたものと異なります。これは、演算に使用している数値または変数の型が結果を保持するのに十分な大きさであるように指定することで回避できます。この例では、数値に L を追加して Long 型にすることによって、中間の結果が Long 型でありオーバーフローが起こらないようにしています。次の例では、Long 型の数値を乗算することによって、1 年あたりのミリ秒を正しく計算する方法を示します。

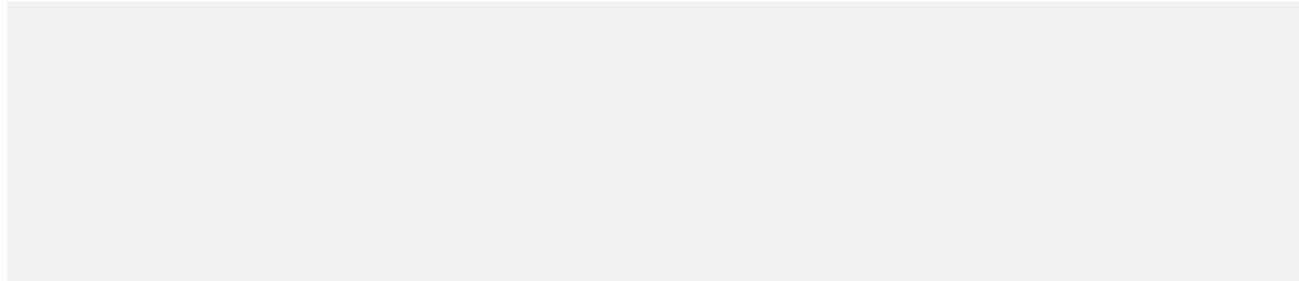


### 除算における端数の消失

Integer または Long 型の数値を除算するとき、結果の端数が発生した場合、それは Double 型や Decimal 型への暗黙的な変換を実行する前に除外されてしまいます。たとえば、式 `int result = 5 / 3;` は、実際の結果 (1.666...) が Integer 型であり、暗黙的に Double 型に変換される前に 1 に丸められるため、1.0 を返します。端数の値を維持するには、除算で Double 型または Decimal 型の数値を使用する必要があります。たとえば、式 `double result = 5.0 / 3.0;` は、5.0 と 3.0 が Double 型の値であるため 1.6666666666666667 を返します。指数が Double 型である結果を生み出すため、端数の値が除外されません。

## 変数

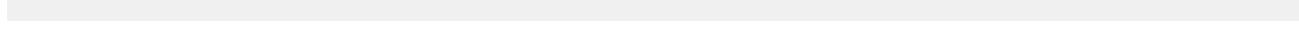
ローカル変数は、Java スタイルの構文で宣言されます。次に例を示します。



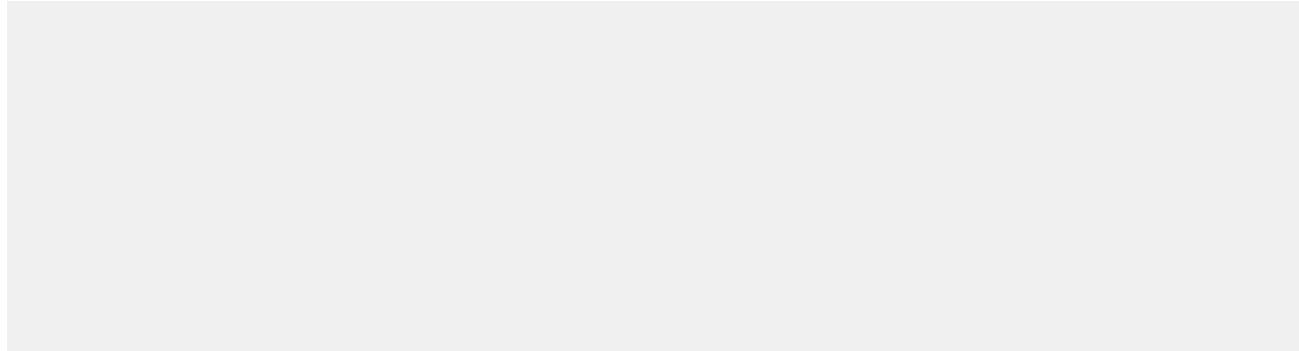
Java と同様、カンマ区切り形式を使用して、複数の変数を单一のステートメントで宣言および初期設定できます。次に例を示します。



すべての変数は `String` を値とすることができ、別の値が割り当てられていない場合は `String` に初期設定されます。たとえば、次の例では、`String a, b, c;` には値が割り当てられ、`String d;` には値が割り当てられていないため、`d` に設定されます。



変数はブロック内のどの場所でも定義でき、その地点から適用されます。サブブロックは、すでに親ブロックで使用されている変数名を再定義できませんが、並行ブロックでは変数名を再利用できます。次に例を示します。



## 大文字と小文字の区別

大文字と小文字を区別しない SOQL クエリおよび SOSL クエリとの混乱を避けるため、Apex も大文字と小文字の区別をしません。つまり、次のようにになります。

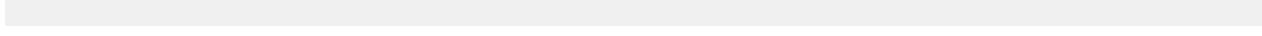
- 変数名とメソッド名では、大文字と小文字を区別しない。次に例を示します。



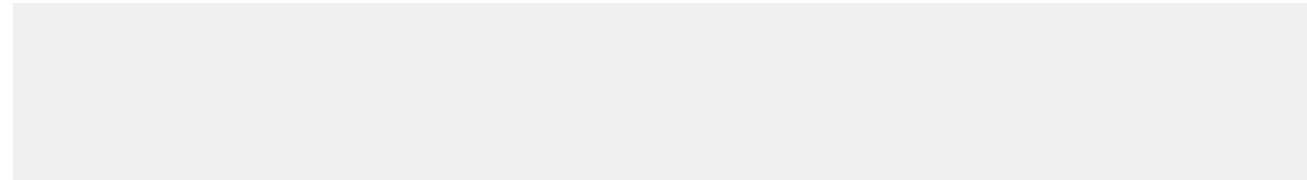
- オブジェクト名と項目名への参照では、大文字と小文字を区別しない。次に例を示します。



- SOQL および SOSL ステートメントは大文字と小文字を区別しない。次に例を示します。



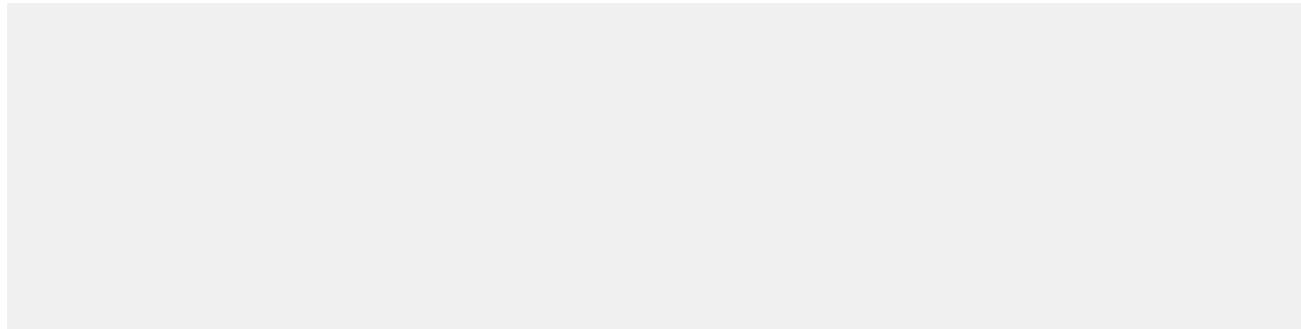
また、Apex は、SOQL と同じ条件セマンティックを使用します。これに基づいて、SOAP API および Salesforce ユーザインターフェースでの比較が行われます。これらのセマンティックを使用すると、興味深い動作が発生します。たとえば、エンドユーザが英字の「m」の前の値という条件(値 <'m'>)に基づいてレポートを生成すると、結果に null 項目が返されます。この動作は、一般にユーザは値を持たない項目を実際の「null」値ではなく、単なる「スペース」文字とみなすという根拠に基づきます。そのため、Apex では、次の表記はすべて true と評価します。

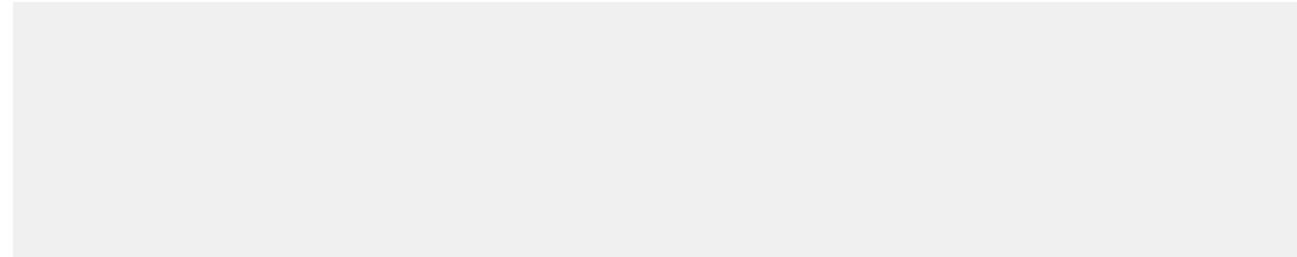


メモ: 上記の例では の評価は true になりますが、 は文字を null 値と比較しようとするとため、エラーを生成します。

## 定数

定数は キーワードを使用して定義できます。定数がクラスに定義されている場合、変数は宣言内で、または静的イニシャライザメソッドを使用して一回のみ割り当てるすることができます。次に例を示します。





詳細は、「[キーワードの使用](#)」(ページ 191)を参照してください。

## 式

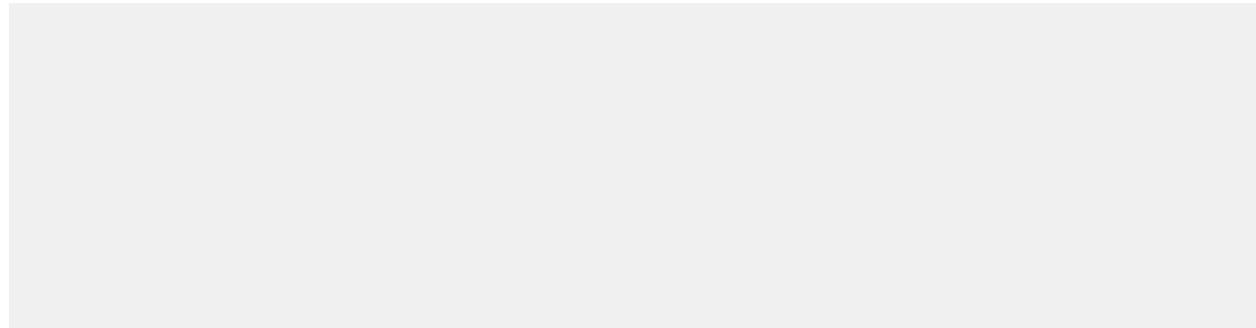
式は、変数、演算子、単一の値を評価するメソッドの呼び出しからなる構成体です。このセクションでは、Apex の式の概要と、次のトピックについて説明しています。

- ・ 式について
- ・ 式の演算子について
- ・ 演算子の優先順位について
- ・ sObject 式およびリスト式の拡張
- ・ コメントの使用

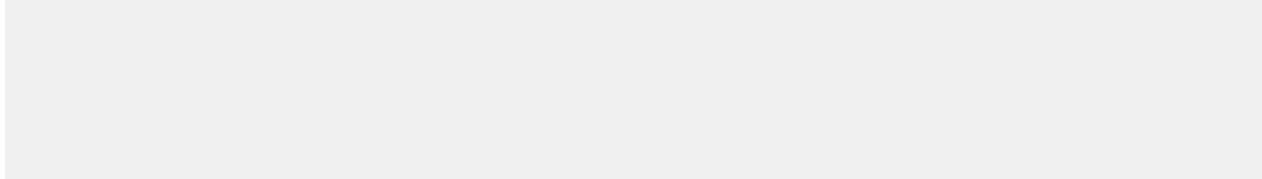
### 式について

式は、変数、演算子、単一の値を評価するメソッドの呼び出しからなる構成体です。Apex では、式は必ず次のいずれかの型になります。

- ・ リテラル式。次に例を示します。
- ・ 新しいsObject、Apex オブジェクト、リスト、セットまたは対応付け。次に例を示します。



- 代入演算子の左側で機能する値 ( $L$  値)。変数、一次元リストの位置、`sObject` または Apex オブジェクト項目参照のほとんど、などです。次に例を示します。



- $L$  値ではない `sObject` 項目参照。次のようなものがあります。
  - リスト内の `sObject` の ID ([「List」を参照](#))
  - `sObject` に関連付けられた子レコードのセット (特定の取引先と関連付けられた取引先責任者のセットなど)。この型の式は、SOQL クエリおよび SOSL クエリとよく似たクエリ結果を返します。
- 大かっこで囲まれた SOQL クエリまたは SOSL クエリ。Apex でその場で評価できます。次に例を示します。



詳細は、[「SOQL および SOSL クエリ」](#) (ページ 83) を参照してください。

- 静的メソッドまたはインスタンスマソッドの呼び出し。次に例を示します。



## 式の演算子について

演算子を使用して式を相互に結合し、複合式を作成することもできます。Apex では、次の演算子を使用できます。

演算子	構文	説明
		代入演算子 (右結合)。 $z$ の値を $L$ 値 $x$ に割り当てます。 $z$ のデータ型は $x$ のデータ型と一致する必要があります。 $z = x$ となることはできません。
		加算代入演算子 (右結合)。 $z$ の値を $x$ の元の値に追加し、 $z$ に新しい値を再代入します。詳細は、 <a href="#">「演算子」</a> を参照してください。 $z += x$ とすることはできません。

演算子	構文	説明
		乗算代入演算子(右結合)。 の値と の元の値を乗算し、 に新しい値を再代入します。 および は Integer または Double、または組み合わせである必要があります。 および を とすることはできません。
		減算代入演算子(右結合)。 の値を の元の値から減算し、 に新しい値を再代入します。 および は Integer または Double、または組み合わせである必要があります。 および を とすることはできません。
		除算代入演算子(右結合)。 元の値を の値で除算し、 に新しい値を再代入します。 および は Integer または Double、または組み合わせである必要があります。 および を とすることはできません。
		OR代入演算子(右結合)。 が Boolean、かつ が Boolean でいずれも false である場合、 は false のままとなります。 そうでない場合、 には true の値が代入されます。
		注意:
		<ul style="list-style-type: none"> <li>この演算子は「短絡」的に動作します。つまり、 は、 が false の場合にのみ評価されます。</li> <li>および を とすることはできません。</li> </ul>
		AND代入演算子(右結合)。 が Boolean、かつ が Boolean でいずれも true である場合、 は true のままとなります。 そうでない場合、 には false の値が代入されます。
		注意:
		<ul style="list-style-type: none"> <li>この演算子は「短絡」的に動作します。つまり、 は、 が true の場合にのみ評価されます。</li> <li>および を とすることはできません。</li> </ul>
		ビット単位の左シフト代入演算子。 の各ビットを ビット分左にシフトします。上位の順位のビットが失われ、新しい右側のビットが0に設定されます。この値は に再代入されます。
		ビット単位の右シフト符号付き代入演算子。 の各ビットを ビット分右にシフトします。下位の順位のビットが失われ、新しい左のビットが、 が正の値の場合は0に、 が負の値の場合は1に設定されます。この値は に再代入されます。
		ビット単位の右シフト符号なし代入演算子。 の各ビットを ビット分右にシフトします。下位の順位のビットが失われ、 のすべての値

演算子	構文	説明
		<p>で、新しい左側のビットが 0 に設定されます。この値は に再代入されます。</p>
		<p>3 項演算子 (右結合)。この演算子は、if-then-else ステートメントの短縮として機能します。 が Boolean で true の場合、 が結果となります。そうでない場合、 が結果となります。 を とすることはできません。</p>
		<p><b>AND</b> 論理演算子 (左結合)。 が Boolean、かつ が Boolean でいずれも true である場合、式の評価は true になります。そうでない場合、式の評価は false になります。</p> <p>注意:</p> <ul style="list-style-type: none"> <li>は より優先されます。</li> <li>この演算子は「短絡」的に動作します。つまり、 は、 が true の場合にのみ評価されます。</li> <li>および を とすることはできません。</li> </ul>
		<p><b>OR</b> 論理演算子 (左結合)。 が Boolean、かつ が Boolean でいずれも false である場合、式の評価は false になります。そうでない場合、式の評価は true になります。</p> <p>注意:</p> <ul style="list-style-type: none"> <li>は より優先されます。</li> <li>この演算子は「短絡」的に動作します。つまり、 は、 が false の場合にのみ評価されます。</li> <li>および を とすることはできません。</li> </ul>
		<p>等価演算子。 の値が の値に等しい場合、式の評価は true になります。そうでない場合、式の評価は false になります。</p> <p>注意:</p> <ul style="list-style-type: none"> <li>Java とは異なり、Apex の は参照の等式ではなく、オブジェクト値の等式を比較します。したがって、次のようになります。 <ul style="list-style-type: none"> <li>を使用した文字列の比較では、大文字と小文字を区別しない</li> <li>を使用した ID の比較では大文字と小文字を区別し、15 文字形式と 18 文字形式を区別しない</li> </ul> </li> <li>sObjects および sObject 配列に対し、 は結果を返す前にすべての sObject 項目の詳細なチェックを実行します。コレクション、組み込み Apex 型、ユーザ定義型に対しても同様です。</li> </ul>

演算子	構文	説明
		<ul style="list-style-type: none"> <li>レコードに対し、各項目には、true と評価する の値が含まれている必要があります。</li> <li>または をリテラルの とすることができます。</li> <li>2つの値の比較によって となることはありません。</li> <li>SOQL および SOSL では、等価演算子に ではなく、 を使用します。Apex と SOQL および SOSL は強くリンクしていますが、多くの現代語では代入に を、等式に を使用するため、構文の不一致が発生します。Apex のデザイナーは、開発者に新しい代入演算子を学ばせるよりも、このパラダイムを維持することが重要であると考えます。したがって、Apex 開発者は主要な Apex コードの本文で を等式テストに、 を SOQL クエリおよび SOSL クエリの等式に使用する必要があります。</li> </ul>
		<p>厳密な等価演算子。 および がメモリ内のまったく同じ場所を参照する場合、式の評価は true になります。そうでない場合、式の評価は false になります。この演算子は sObjects またはコレクション(対応付けまたはリストなど)のみを対象とします。Apex オブジェクト(例外またはクラスのインスタンス化など)の場合、厳密な等価演算子は等価演算子と同じです。</p>
		<p>小なり演算子。 が より小さい場合、式の評価は true になります。そうでない場合、式の評価は false になります。</p> <p>注意:</p> <ul style="list-style-type: none"> <li>他のデータベースストアドプロシージャと異なり、Apex ではトライステート Boolean 論理はサポートされず、2つの値の比較によって となることはありません。</li> <li>または が で Integer、Double、Date、または Datetime となる場合、式は false となります。</li> <li>以外の String または ID 値は常に 値より大きくなります。</li> <li>および が ID の場合、それらは同じデータ型のオブジェクトを参照する必要があります。そうでない場合、ランタイムエラーが発生します。</li> <li>または のいずれかが ID でもう一方の値が String の場合、String 値は ID として検証され、処理されます。</li> <li>および を Boolean とすることはできません。</li> <li>2つの文字列の比較は、コンテキストユーザのロケールにしたがって実行されます。</li> </ul>

演算子	構文	説明
		<p>大なり演算子。 <code>が</code> より大きい場合、式の評価は true になります。 そうでない場合、式の評価は false になります。</p> <p>注意:</p> <ul style="list-style-type: none"> <li>2つの値の比較によって <code>      </code> となることはありません。</li> <li>または <code>が</code> <code>      </code> で Integer、Double、Date、または Datetime となる場合、式は false となります。</li> <li>以外の String または ID 値は常に <code>      </code> 値より大きくなります。</li> <li>および <code>が</code> ID の場合、それらは同じデータ型のオブジェクトを参照する必要があります。そうでない場合、ランタイムエラーが発生します。</li> <li>または のいずれかが ID でもう一方の値が String の場合、String 値は ID として検証され、処理されます。</li> <li>および <code>を</code> Boolean とすることはできません。</li> <li>2つの文字列の比較は、コンテキストユーザのロケールにしたがって実行されます。</li> </ul>
		<p><code>&lt;=</code> 演算子。 <code>が</code> より小さいか等しい場合、式の評価は true になります。 そうでない場合、式の評価は false になります。</p> <p>注意:</p> <ul style="list-style-type: none"> <li>2つの値の比較によって <code>      </code> となることはありません。</li> <li>または <code>が</code> <code>      </code> で Integer、Double、Date、または Datetime となる場合、式は false となります。</li> <li>以外の String または ID 値は常に <code>      </code> 値より大きくなります。</li> <li>および <code>が</code> ID の場合、それらは同じデータ型のオブジェクトを参照する必要があります。そうでない場合、ランタイムエラーが発生します。</li> <li>または のいずれかが ID でもう一方の値が String の場合、String 値は ID として検証され、処理されます。</li> <li>および <code>を</code> Boolean とすることはできません。</li> <li>2つの文字列の比較は、コンテキストユーザのロケールにしたがって実行されます。</li> </ul>
		<p><code>&gt;=</code> 演算子。 <code>が</code> より大きいか等しい場合、式の評価は true になります。 そうでない場合、式の評価は false になります。</p> <p>注意:</p> <ul style="list-style-type: none"> <li>2つの値の比較によって <code>      </code> となることはありません。</li> </ul>

演算子	構文	説明
		<ul style="list-style-type: none"> <li>または が で Integer、Double、Date、または Datetime となる場合、式は false となります。</li> <li>以外の String または ID 値は常に 値より大きくなります。</li> <li>および が ID の場合、それらは同じデータ型のオブジェクトを参照する必要があります。そうでない場合、ランタイムエラーが発生します。</li> <li>または のいずれかが ID でもう一方の値が String の場合、String 値は ID として検証され、処理されます。</li> <li>および を Boolean とすることはできません。</li> <li>2つの文字列の比較は、コンテキストユーザのロケールにしたがって実行されます。</li> </ul>
		不等価演算子。 の値が の値と等しくない場合、式の評価は true になります。そうでない場合、式の評価は false になります。
		注意:
		<ul style="list-style-type: none"> <li>Java とは異なり、Apex の は参照の等式ではなく、オブジェクト値の等式を比較します。</li> <li>sObjects および sObject 配列に対し、 は結果を返す前にすべての sObject 項目の詳細なチェックを実行します。</li> <li>レコードに関して、項目のさまざまな値がレコードに存在する場合、 は true に評価します。</li> <li>または をリテラルの とすることができます。</li> <li>2つの値の比較によって となることはありません。</li> </ul>
		厳密な不等価演算子。 および がメモリ内のまったく同じ場所を参照しない場合、式の評価は true になります。そうでない場合、式の評価は false になります。この演算子は sObjects またはコレクション(対応付けまたはリストなど)、または Apex オブジェクト(クラスの例外またはインスタンス化)のみを対象とします。
		加算演算子。次のルールに従って、 の値を の値に加算します。
		<ul style="list-style-type: none"> <li>および が Integer または Double の場合、 の値を の値に加算します。Double が使用される場合、結果は Double となります。</li> <li>が Date で が Integer の場合、指定した日数を増分した新しい Date を返します。</li> <li>が Datetime で が Integer または Double の場合、指定した日数を増分した新しい Date を返します。小数点以下の値は1日未満の部分に相当します。</li> </ul>

演算子	構文	説明
		<ul style="list-style-type: none"> <li>が String で が String またはその他のデータ型の 以外の引数である場合、 を の末尾に連結します。</li> </ul>
		<p>減算演算子。次のルールに従って、 の値から の値を減算します。</p> <ul style="list-style-type: none"> <li>および が Integer または Double の場合、 の値を の値から 減算します。Double が使用される場合、結果は Double となります。</li> <li>が Date で が Integer の場合、指定した日数を減分した新しい Date を返します。</li> <li>が Datetime で が Integer または Double の場合、指定した日数を減分した新しい Date を返します。小数点以下の値は1日未満の部分に相当します。</li> </ul>
		<p>乗算演算子。Integer または Double の と、Integer または Double の を乗算します。Double が使用されると、結果は Double になります。</p>
		<p>除算演算子。Integer または Double の を、Integer または Double の で除算します。Double が使用されると、結果は Double になります。</p>
		<p>論理補数演算子。Boolean の値を反転し、true は false に、false を true にします。</p>
		<p>単項否定演算子。Integer または Double の を -1 で乗算します。正の等価 も構文的に有効ですが、数学的効果はありません。</p>
		<p>インクリメント演算子。1を 数値型の変数 の値に加算します。プレフィックスとして付けた場合 ( )、式の評価は増分後の x の値になります。ポストフィックスとして付けた場合 ( )、式の評価は増分前の x の値になります。</p>
		<p>デクリメント演算子。1を 数値型の変数 の値から減算します。プレフィックスとして付けた場合 ( )、式の評価は減分後の x の値になります。ポストフィックスとして付けた場合 ( )、式の評価は減分前の x の値になります。</p>
		<p>ビット単位の AND 演算子。 の各ビットと の対応するビットを AND 演算します。両方のビットが 1 に設定されると結果ビットは 1 に設定されます。この演算子は Long または Integer には適用されません。</p>
		<p>ビット単位の OR 演算子。 の各ビットと の対応するビットを OR 演算します。少なくとも1つのビットが1に設定されると結果ビットは 1 に設定されます。この演算子は Long または Integer には適用されません。</p>

演算子	構文	説明
		ビット単位の排他的OR演算子。 の各ビットと の対応するビットを排他的 OR 演算します。1つのみのビットが1に設定され、他のビットが0に設定されると、結果ビットは1に設定されます。
		ビット単位の排他的OR演算子。 の各ビットと の対応するビットを排他的 OR 演算します。1つのみのビットが1に設定され、他のビットが0に設定されると、結果ビットは1に設定されます。
		ビット単位の左シフト演算子。 の各ビットを ビット分左にシフトします。上位の順位のビットが失われ、新しい右側のビットが0に設定されます。
		ビット単位の右シフト符号付き演算子。 の各ビットを ビット分右にシフトします。下位の順位のビットが失われ、新しい左のビットが、 が正の値の場合は0に、 が負の値の場合は1に設定されます。
		ビット単位の右シフト符号なし演算子。 の各ビットを ビット分右にシフトします。下位の順位のビットが失われ、 のすべての値で、新しい左側のビットが0に設定されます。
		小かっこ。式 の優先順位を評価します。複合式で第一優先として評価します。

## 演算子の優先順位について

Apex では、次の演算子の優先順位の規則を使用します。

優先順位	演算子	説明
1		グループ化と、プレフィックスインクリメントおよびデクリメント
2		単項否定、型キャスト、およびオブジェクト作成
3		乗算および除算
4		加算および減算
5		大なり記号および小なり記号、参照テスト
6		比較: 等しい、等しくない
7		論理 AND
8		論理 OR

優先順位	演算子	説明
9		代入演算子

## sObject 式およびリスト式の拡張

Java の場合と同様、sObject 式とリスト式をそれぞれメソッド参照とリスト式で拡張して、新しい式を作成できます。

次の例では、新しい取引先名の長さを含む新しい変数が に割り当てられます。

上記の はリストを生成します。

このリストは、SOQL ステートメント によって入力されます。

Item 0、つまりリストの最初の項目が、文字列 の次の部分によってアクセスされます。

リストの sObject の名前がアクセスされた後、メソッドが長さ を返します。

次の例では、小文字に変更された名前が返されます。

## コメントの使用

Apex コードでは、単一のコメントと複数のコメントを使用できます。

- 1 行のコメントを作成するには、`//` を使用します。`//` の右側の同じ行にあるすべての文字は、パーサーで無視されます。次に例を示します。

- 複数のコメントを作成するには、コメントブロックの冒頭から末尾までを `/*` と `*/` で囲みます。次に例を示します。

## 代入ステートメント

代入ステートメントは、次の 2 つの形式のいずれかで値を変数に代入するステートメントです。

上記の形式で、\_\_\_\_\_ は、代入演算子の左側に入力できる式を表します。その具体的な内容は次のとおりです。

- 単純な変数。次に例を示します。

- 参照解決されたリスト要素。次に例を示します。

- コンテキストユーザが編集権限を持つ sObject 項目参照。次に例を示します。

代入は必ず参照によって行われます。次に例を示します。

同様に、2つのリストがメモリ内の同じ値を示すことができます。次に例を示します。

のほか、有効な割り当て演算子には = 、 += 、 -= 、 \*= 、 /= 、および %= があります。「式の演算子について」(ページ 63)を参照してください。

## 条件 (If-Else) ステートメント

Apex の条件ステートメントは、Java と同じように動作します。

の部分は常に省略可能で、最も近い `if` にグループ化されます。次に例を示します。

```
if           else
```

上記は、次のステートメントと同等です。

```
if  
  
else
```

繰り返しのステートメントも使用できます。次に例を示します。

## ループ

Apex では、次の 5 種類の手続き型ループをサポートしています。

- `statement Boolean_condition`
- `Boolean_condition statement`
- `initialization Boolean_exit_condition increment statement`
- `variable array_or_set statement`

- `variable    inline_soql_query    statement`

すべてのループは、次のループ制御構文を使用できます

- ループ全体を終了します。
- ループの次の反復にスキップします。

## Do-While ループ

Apex ループは、特定の Boolean 条件が true である限り、コードのブロックを繰り返し実行します。構文は次のとおりです。

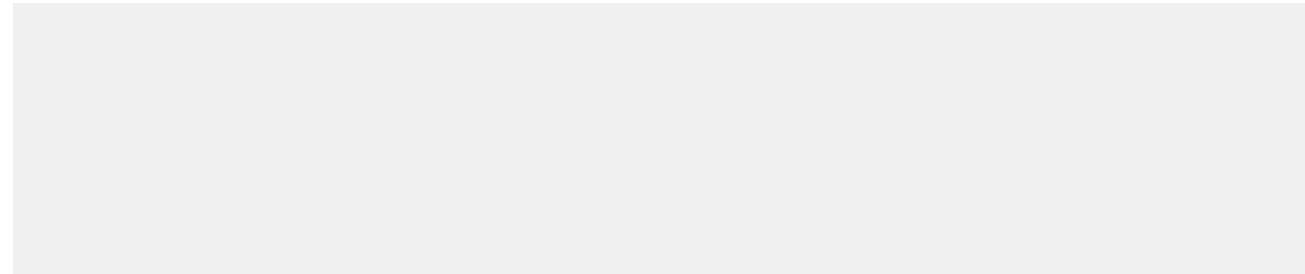


```
code_block
```

メモ: `code_block` は必ず中かっこ( )で囲まれている必要があります。

Java の場合と同様、Apex ループは、最初のループが実行されるまで、Boolean 条件ステートメントをチェックしません。そのため、コードブロックは必ず少なくとも 1 回は実行されます。

次のコード例は、1 から 10 の数値をデバッグログに出力します。

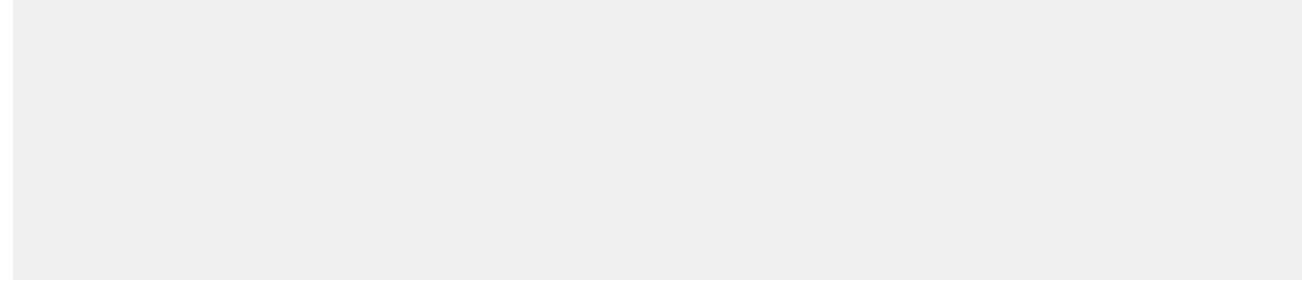


```
code_block
```

メモ: `code_block` に複数のステートメントが含まれる場合にのみ、このブロックを中かっこ( )で囲む必要があります。

と異なり、`FOR` ループは、最初のループが実行される前に Boolean 条件ステートメントをチェックします。その結果、コードブロックが実行されない場合もあります。

次のコード例は、1 から 10 の数値をデバッグログに出力します。



## For ループ

Apex では、`FOR` ループの次の 3 つのバリエーションを使用できます。

- 従来の `FOR` ループ:

```
init_stmt  exit_condition  increment_stmt
code_block
```

- リスト反復またはセット反復の `FOR` ループ:

```
variable  list_or_set
code_block
```

ここで、`variable` は、`list_or_set` と同じプリミティブデータ型または sObject 型である必要があります。

- SOQL ループ:

```
variable  soql_query
code_block
```

または

```
variable_list  soql_query
code_block
```

`variable` および `variable_list` は、`soql_query` で返される sObject と同じデータ型である必要があります。



メモ: `code_block` に複数のステートメントが含まれる場合にのみ、このブロックを中かっこ( )で囲む必要があります。

それぞれについて、後のセクションで詳細に説明します。

## 従来の For ループ

Apex の従来の ループは、Java その他の言語で使用される従来の構文に対応しています。構文は次のとおりです。

```
init_stmt  exit_condition  increment_stmt  
code_block
```

この種類の ループを実行すると、Apex ランタイムエンジンは、次の手順を順番に実行します。

1. ループの `init_stmt` コンポーネントを実行します。このステートメントで複数の変数の宣言、初期設定、またはその両方を行えます。
2. `exit_condition` チェックを実行します。true の場合、ループは続行します。false の場合、ループは終了します。
3. `code_block` を実行します。
4. `increment_stmt` ステートメントを実行します。
5. 手順 2 に戻ります。

次のコード例は、1 から 10 の数値をデバッグログに出力します。構文を実証するために、追加の初期設定変数が挿入されています。

```
variable      list_or_set  
code_block
```

## リスト反復またはセット反復の For ループ

リスト反復またはセット反復の ループは、リスト内またはセット内のすべての要素を反復します。構文は次のとおりです。

```
variable      list_or_set  
code_block
```

ここで、`variable` は、`list_or_set` と同じプリミティブデータ型または sObject 型である必要があります。

この種類の ループを実行すると、Apex ランタイムエンジンは `variable` を `list_or_set` の各要素に割り当て、各値で `code_block` を実行します。

たとえば、次のコードは、1 から 10 の数値をデバッグルогに出力します。

## SOQL For ループ

SOQL ループは SOQL クエリで返されたすべての sObject レコードを反復します。SOQL ループの構文は次のいずれかになります。

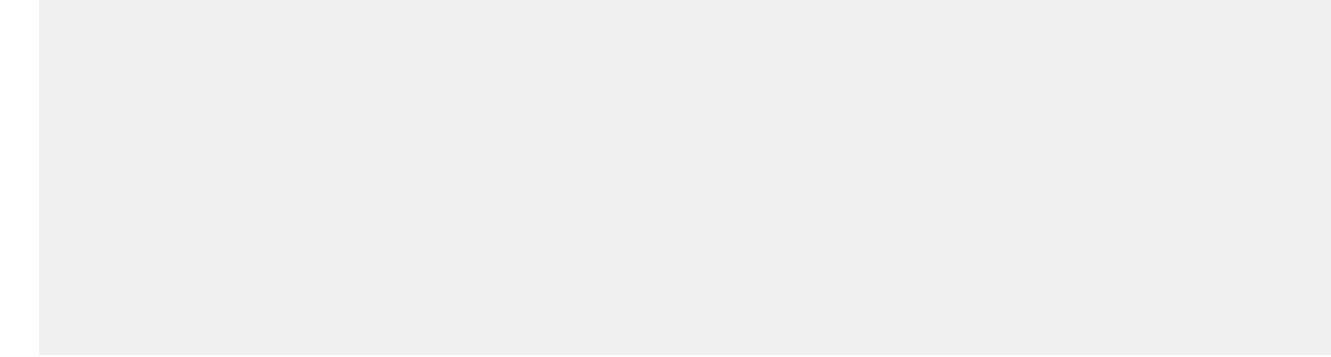
```
variable    soql_query  
code_block
```

または

```
variable_list    soql_query  
code_block
```

`variable` および `variable_list` は、`soql_query` で返される sObject と同じデータ型である必要があります。標準 SOQL クエリと同様、`soql_query` ステートメントは、構文を使用して句のコード式を参照することができます。次に例を示します。

次の例では、SOQL クエリからのリストの作成と DML メソッドを結合します。



### SOQL For ループと標準 SOQL クエリの比較

SOQL ループは、sObjects を取得するために使用するメソッドが、標準 SOQL ステートメントとは異なります。「[SOQL および SOSL クエリ](#)」で説明する標準クエリはクエリの または多数のオブジェクトレコードを取得できますが、SOQL ループは、SOAP API の メソッドと メソッドをコールする効率的なチャネルを使用して、すべての sObject を取得します。開発者は常に SOQL ループを使用して、多数のレコードを返すクエリ結果を処理し、[ヒープサイズ](#)の制限に達するのを回避します。

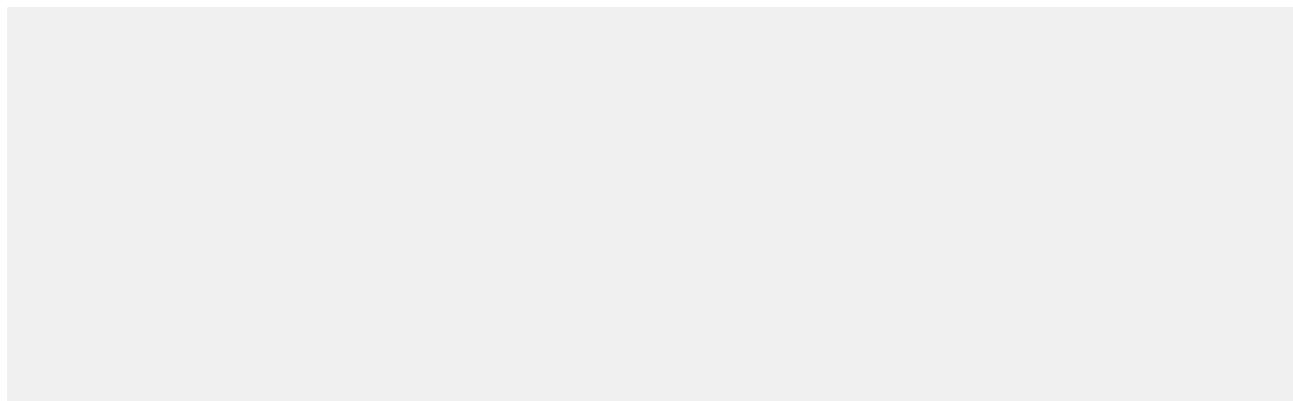
[集計関数](#)を含むクエリでは、 をサポートしません。 ループで 2000 を超える行を返す集計関数を含むクエリを使用すると、実行時例外が発生します。

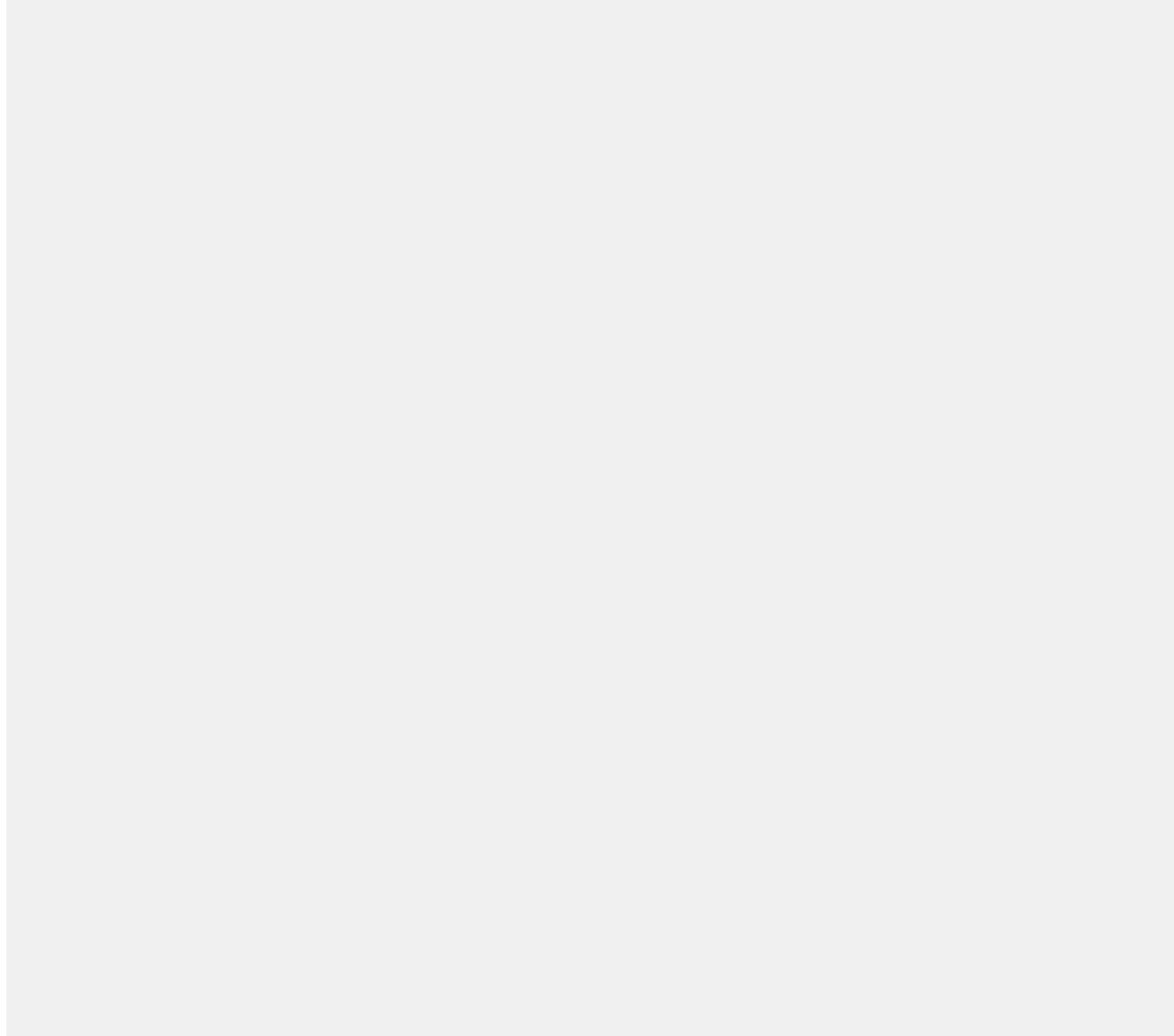
### SOQL For ループの形式

SOQL ループは、単一の sObject 変数を使用して一度に 1 件のレコードを処理するか、sObject リストを使用して一度に 200 個の sObject を一括処理できます。

- ・ 単一の sObject 形式は ループの を sObject レコードごとに 1 回実行します。そのため、理解しやすく、簡単に使用できますが、 ループの本文内でデータ操作言語 (DML) ステートメントを使用すると、効率性が大幅に低下します。DML ステートメントは、一度に 1 つの sObject の処理のみを完了します。
- ・ sObject リスト形式は ループの を 200 件の sObject のリストごとに 1 回実行します。そのため、多少理解しにくく、使用が難しくなりますが、 ループの本文内で DML ステートメントを使用する必要がある場合に最適です。DML ステートメントは、sObject のリストを一括処理します。

たとえば、次のコードは 2 種類の SOQL クエリ ループの差異を示します。

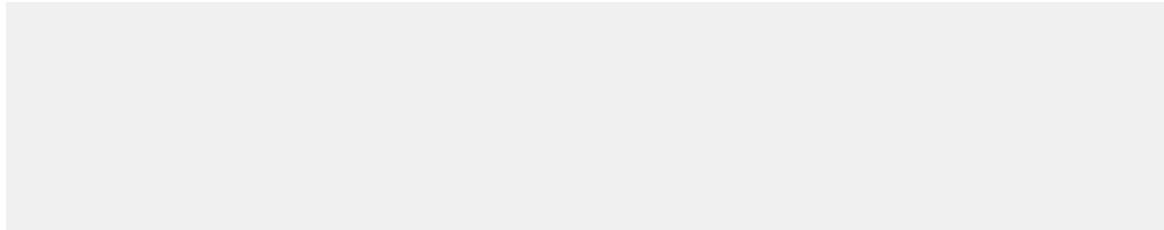




 メモ:

- キーワードと キーワードは、どちらのインラインクエリ ループ形式でも使用できます。sObject リスト形式を使用すると、 は、sObjects の次のリストにスキップします。
- DML ステートメントは一度に最大 10,000 件のレコードを処理でき、sObject リスト ループは 200 件のレコードを一括処理します。そのため、sObject リスト ループで返されたレコードごとに複数のレコードを挿入、更新、または削除する場合、制限のランタイムエラーが発生する可能性があります。「[実行ガバナと制限について](#)」(ページ 337)を参照してください。
- SOQL ループで「 」というメッセージが表示されて が発生する可能性があります。この例外は、取得された sObject の大量の子レコードにループ内でアクセスしたり、このようなレコードセットの

サイズを取得したりする場合に発生することがあります。この例外を回避するには、次のようにループを使用して、子レコードを反復処理します。



## Exception ステートメント

Apex では、*Exception* を使用して、コード実行の正常な流れを中断させるエラーその他のイベントを記録します。

ステートメントは例外の生成に使用でき、`throw`、`catch`、および `finally` は例外から適切に復旧するために使用できます。

*Exception* クラスを使用して、独自の例外を作成することもできます。詳細は、「[例外クラス](#)」(ページ 785)を参照してください。

### throw ステートメント

ステートメントを使用して、エラーが発生したことを通知できます。例外を発生させるには、`throw` ステートメントに例外オブジェクトを指定して、特定のエラーに関する情報を提供します。次に例を示します。

```
exceptionObject
```

### Try-Catch-Finally ステートメント

、`try`、`catch`、`finally` の各ステートメントを使用して、発生した例外から適切に復旧できます。

- `try` ステートメントは例外が発生する可能性のあるコードのブロックを識別します。
- `catch` ステートメントは、特定の種類の例外を処理できるコードのブロックを識別します。単一のブロックに対して、複数の `catch` ステートメントを関連付けられますが、各 `catch` ステートメントには一意の例外種別が必要です。
- 必要に応じて、`finally` ステートメントは実行が保証されているコードのブロックを識別し、`finally` ブロック内のコードの後でクリーンアップすることができます。1つ `try` ステートメントに `finally` ステートメントを1つのみ関連付けられます。

## 構文

これらのステートメントの構文は次のとおりです。

```
code_block
    exceptionType
code_block
```

```
code_block
```

```
code_block
```

## 例

次に例を示します。

 メモ: 実行ガバナにより発生した制限の例外は検出できません。「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

## キャッチできない例外

キャッチできない特殊なタイプの組み込み例外もあります。このような例外は、Force.com プラットフォームの重大な状況に関連付けられています。このような状況では、コードの実行を中止する必要があります。例外処理で実行を再開することはできません。このような例外の 1 つとして、ガバナ制限に達した場合 (SOQL クエリの最大発行数に達した場合など) に実行時に発生する制限の例外があります。他の例として、アサーションステートメント (`Method` を使用) に失敗した場合に発生する例外やライセンスの例外が挙げられます。

例外をキャッチできない場合、ブロックやブロック (ある場合) は実行されません。

## SOQL および SOSL クエリ

ステートメントを大かっこで囲むことによって、Apex の Salesforce オブジェクトクエリ言語 (SOQL) または Salesforce オブジェクト検索言語 (SOSL) ステートメントをその場で評価することができます。

### SOQL ステートメント

SOQL ステートメントは、`sObjects` のリスト、単一 `sObject`、または メソッドクエリの `Integer` を評価します。

たとえば、`Acme` という取引先のリストを取得したとします。

このリストから各要素にアクセスできます。

既存のオブジェクトの SOQL クエリから新しいオブジェクトを作成することもできます。次の例では、従業員数が 10 人を超える最初の取引先の新しい取引先責任者を作成します。

新規作成したオブジェクトのこの項目には `null` 値が入力されます。設定する必要はありません。

メソッドを使用して、クエリによって返される行数を返すことができます。次の例では、姓が `Weissman` の取引先責任者の合計数を返します。

次の標準的な演算を使用して、結果を処理することもできます。

SOQL クエリの構文の詳細は、『[Salesforce SOQL および SOSL リファレンス](#)』を参照してください。

## SOSL ステートメント

SOSL は、sObject リストの一覧に対して評価を行います。各リストには特定の sObject 型の検索結果が含まれます。結果リストは必ず、SOSL クエリで指定された順序で返されます。SOSL クエリが指定された sObject 型のレコードを返さない場合、検索結果には、その sObject の空のリストが返されます。

たとえば、次のように語句の対応付けで始まる取引先、取引先責任者、商談、およびリードのリストを返すことができます。

### メモ:

Apex の `SELECT` 句の構文は、SOAP API の `SELECT` 句の構文とは異なります。

- Apex の場合、`SELECT` 句の値は単一引用符で区画されます。次に例を示します。

- Force.com API の場合、`SELECT` 句の値は中かっこで区切られます。次に例を示します。

で、返された各オブジェクトの配列を作成できます。

SOSL クエリの構文の詳細は、『[Salesforce SOQL および SOSL リファレンス](#)』を参照してください。

## SOQL および SOSL クエリ結果の処理

SOQL クエリおよび SOSL クエリは、元のクエリで選択された sObject 項目のデータのみを返します。SOQL クエリまたは SOSL クエリで選択されていない項目 (ID 以外) にアクセスしようとすると、データベースのその項目に値が含まれている場合であっても、ランタイムエラーが発生します。次のコード例では、ランタイムエラーが発生します。

次のコード例は、ランタイムエラーが発生しないように上記のコードを書き換えたものです。      が      の後に、SELECT ステートメントの一部として追加されています。

選択された sObject 項目が 1 つのみの場合でも、SOQL クエリまたは SOSL クエリは必ずすべてのレコードとしてデータを返します。その結果、項目にアクセスするには、項目を参照解決する必要があります。たとえば、次のコードは、SOQL クエリでデータベースから sObject リストを取得し、リスト内の最初の取引先レコードにアクセスし、レコードの                    項目を参照解決します。

SOQL クエリ結果で sObject 項目を参照解決する必要がないのは、クエリが                    演算の結果として Integer を返す場合のみです。

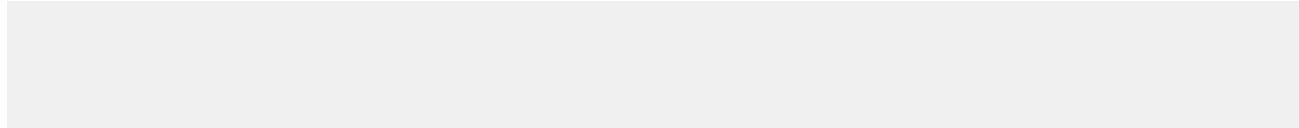
SOSL クエリで返されるレコードの項目は、必ず参照解決する必要があります。

数式を含む sObject 項目は、SOQL クエリまたは SOSL クエリが発行されたときに項目の値を返します。数式内で使用されている他の項目に対する変更は、レコードが Apex で保存され、再度クエリされるまでは、数式項目の値に反映されません。その他の参照専用 sObject 項目と同様、数式項目の値自体を Apex で変更することはできません。

## SOQL 集計関数の使用

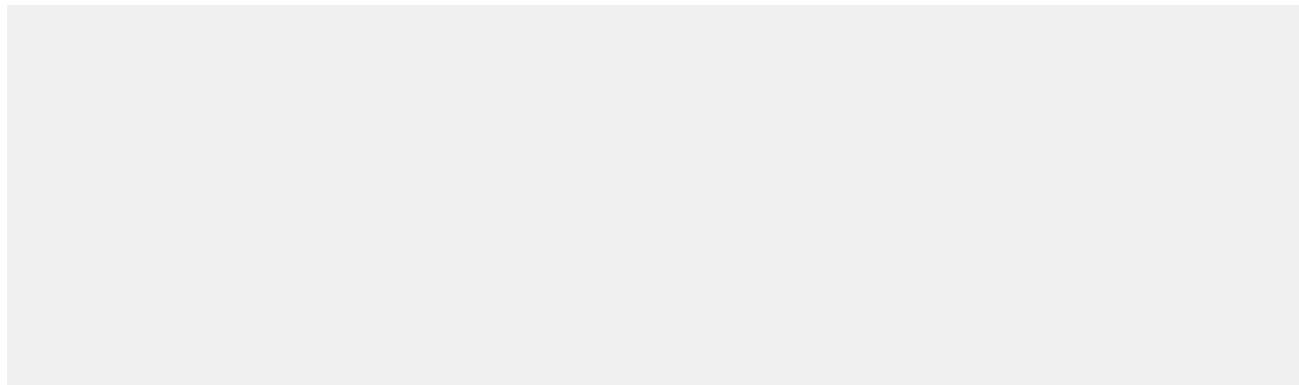
や      などの SOQL の集計関数を使用して、クエリでデータをロールアップおよび集計できます。集計関数の詳細は、[Salesforce SOQL and SOSL Reference Guide](#)の「Aggregate Functions」を参照してください。

集計関数は `sum` 句を使用せずに使用できます。たとえば、`sum` 集計関数を使用して、すべての商談の平均金額を調べることができます。



集計関数を含むクエリは、AggregateResult オブジェクトの配列で結果を返します。AggregateResult は参照専用 sObject で、クエリ結果にのみ使用されます。

集計関数は `group by` 句と共に使用すると、より強力にレポートを生成するツールとなります。たとえば、キャンペーンごとにすべての商談の平均金額を調べることができます。



別名のない `list` リストの集計項目は、形式が `i` の暗黙的別名を自動的に取得します。`i` は、明示的な別名のない集計項目の順序を示します。`i` の値は 0 から始まり、明示的な別名のない集計項目ごとに増えます。詳細は、[Salesforce SOQL and SOSL Reference Guide](#) の「Using Aliases with `list`」を参照してください。

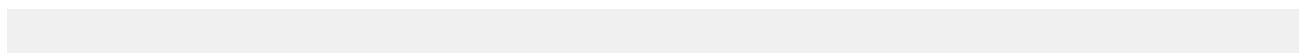


メモ: 集計関数を含むクエリには、返されるレコードの合計数に関する他の SOQL クエリと同じ [オーバーラン制限](#) が適用されます。制限対象には、クエリによって返される行数だけでなく、集計に含まれるレコード数も含まれます。この制限に達した場合は、`list` 句に条件を追加して、クエリが処理するレコード数を減らす必要があります。

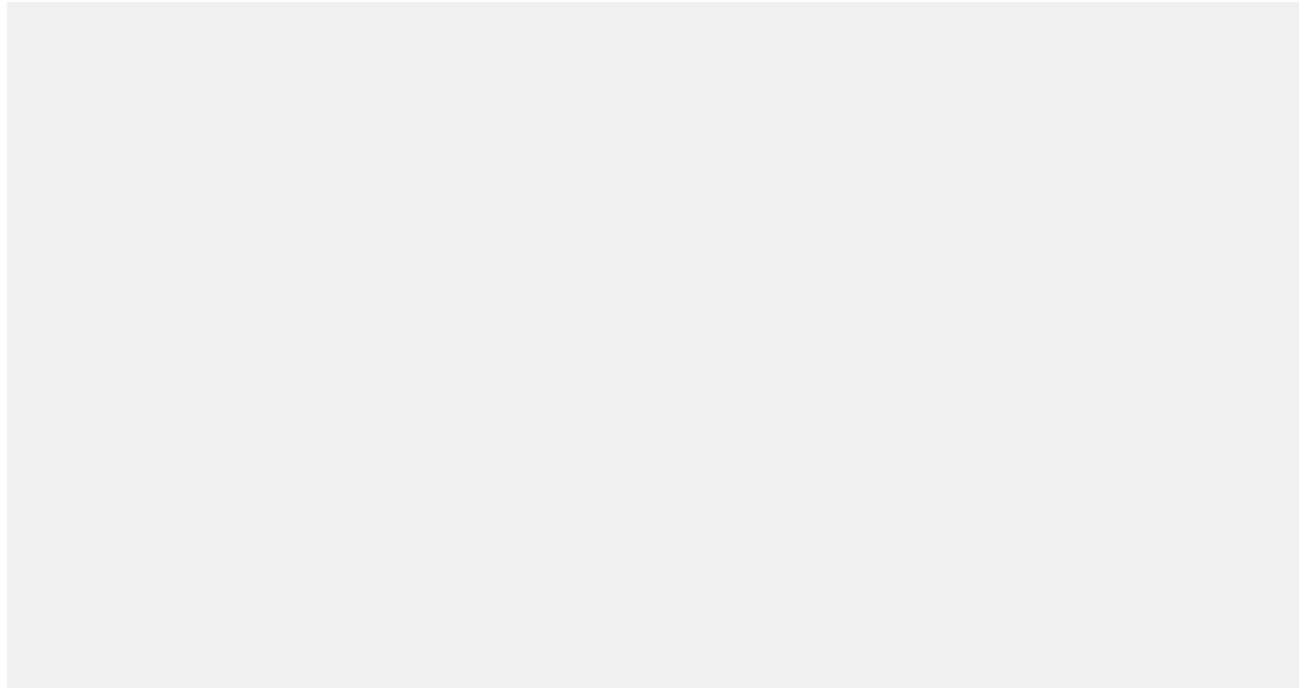
## 非常に大きい SOQL クエリの処理

SOQL クエリがヒープサイズの制限を超える多数の sObjects を返し、エラーを引き起こす場合があります。問題を解決するには、代わりに SOQL クエリ ループを使用します。 および への内部コールが使用されるため、レコードの複数の一括処理が可能になります。

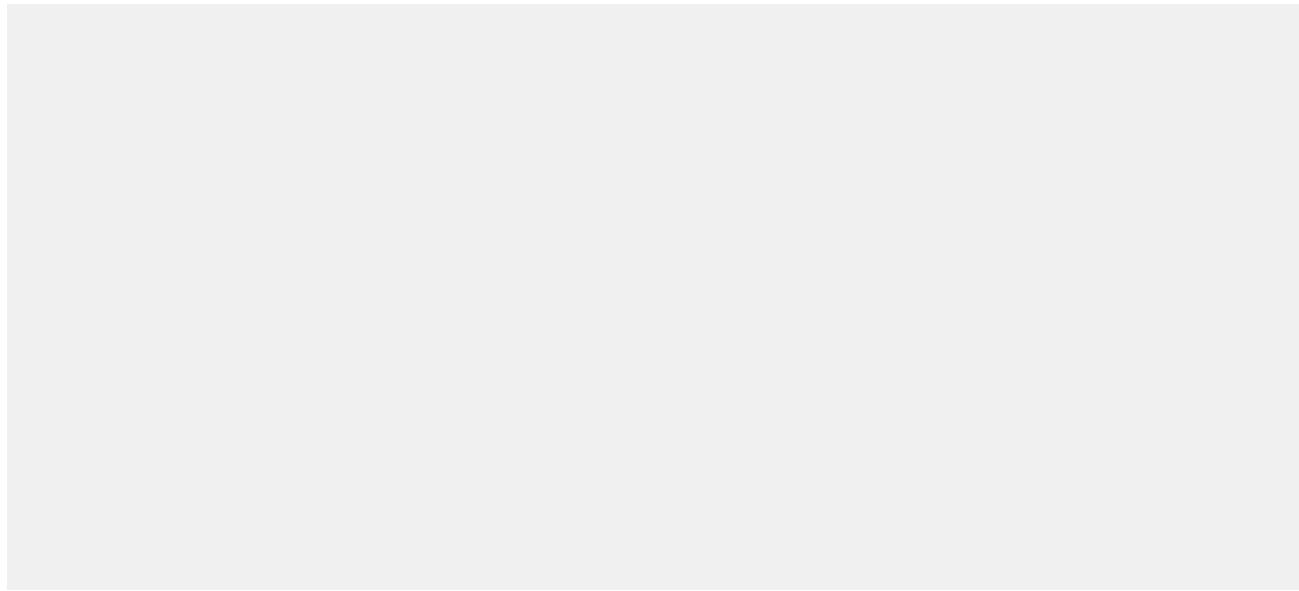
たとえば、結果が大きすぎる場合、次の構文で実行時例外が発生します。



代わりに、次の例のいずれかで SOQL クエリ ループを使用します。



次の例は、レコードの一括更新に使用する SOQL クエリ ループを示します。指定された条件に一致する姓名を持つ取引先責任者のすべてのレコードで、取引先責任者の姓を変更するとします。



ループで SOQL クエリを使用する代わりに、[Apex の一括処理](#)を使用してレコードを一括更新すると、ガバナ制限に達するリスクが最小限に抑えられます。

詳細は、「[SOQL For ループ](#)」(ページ 78)を参照してください。

## より効率的な SOQL クエリ

最高のパフォーマンスを得るために、特にトリガ内のクエリに対しては、セレクティブ SOQL クエリを使用する必要があります。実行時間が長時間に渡ることを回避するために、セレクティブ以外の SOQL クエリはシステムより終了される場合があります。100,000 件を超えるレコードを含むオブジェクトに対してトリガでセレクティブではないクエリを使用すると、エラーメッセージが表示されます。このエラーを回避するには、必ずセレクティブクエリを使用します。

### セレクティブ SOQL クエリ条件

- クエリ検索条件の 1 つがインデックス付き項目にあり、そのクエリ検索条件によって結果となる行数がシステム定義のしきい値より少なくなる場合、そのクエリはセレクティブです。SOQL クエリのパフォーマンスは、WHERE 句に使用される 2 つ以上の検索条件がその条件を満たす場合に改善されます。
- 選択度しきい値は、初めの 100 万レコードについてはレコードの 10% で、以降のレコードから最大 333,000 個のレコードについては 5% になります。インデックス付き標準項目のクエリ検索条件などの一部の状況では、しきい値が高くなる場合があります。また、選択度しきい値は変化します。

### セレクティブ SOQL クエリのカスタムインデックスに関する考慮事項

- 主キー (ID、名前、所有者項目)、外部キー (ロックアップまたは主従関係項目)、監査日付 (LastModifiedDate など)、および外部 ID または一意としてマークされたカスタム項目は、デフォルトでインデックスが付けられます。
- 頻繁に実行されるクエリのパフォーマンスがインデックスによって向上することが Salesforce オプティマイザによって確認された場合、デフォルトでインデックスが付けられない項目に、後から自動的にインデックスを付けることができます。
- Salesforce.com サポートは、お客様からの要求に応じてカスタムインデックスを追加できます。
- カスタムインデックスは、複数選択リスト、マルチ通貨組織の通貨項目、ロングテキスト項目、一部の数式項目、およびバイナリ項目 (blob 型の項目、ファイル、または暗号化されたテキスト項目) では作成できません。新しいデータ型 (一般的に複雑なデータ型) は Salesforce に追加できますが、これらの項目にはカスタムインデックスを使用できません。
- 通常、カスタムインデックスは次の場合に使用されません。
  - ◊ クエリされた値が前述のシステム定義のしきい値を超える
  - ◊ 検索条件の演算子が、 $($ または $)$ 、 $,$  および $\neq$  の負の演算子である
  - ◊ 検索条件に $\in$  演算子が使用されており、スキヤンされる行数が 333,000 を超える。これは、演算子にインデックスの完全スキヤンが必要であるためです。このしきい値は変化します。
  - ◊ 空の値と比較している ( $=$  )

ただし、カスタムインデックスが使用されない複雑なシナリオは他にもあります。ここに記載された条件以外のシナリオがある場合、またはセレクティブではないクエリに関するヘルプが必要な場合は、Salesforce.com カスタマーサポートにお問い合わせください。

### セレクティブ SOQL クエリの例

大きなオブジェクトでのクエリがセレクティブであるかどうかを理解するために、いくつかのクエリを解析することにします。これらのクエリについては、Account sObject に 100,000 件を超えるレコード (削除されたレコードがごみ箱に残っている、論理削除されたレコードを含む) があると仮定します。

クエリ 1:

句は、インデックス付き項目 (ID) に使用されています。

が選択度しきい値より少ないレコードを返す場合、ID へのインデックスが使用されます。これは、ID のリストに少ない数のレコードが含まれるため、一般的なケースです。

クエリ 2:

名前はインデックス付きですが(主キー) Account は大きなオブジェクトであるため、この検索条件はほとんどのレコードを返すことから、クエリは非セレクティブとなります。

クエリ 3:

ここでは、各検索条件が個別に考慮された場合にセレクティブであるかどうかを確認する必要があります。前の例で確認したように、最初の検索条件はセレクティブではありません。そのため、2つの検索条件を重点的に確認することにします。

が返すレコードの件数が選択度しきい値より少なく、かつ CustomField\_\_c がインデックス付きである場合、このクエリはセレクティブです。

## 1つのレコードを返す SOQL クエリの使用

結果リストに1つだけ要素が含まれている場合、SOQL クエリを使用して単一の sObject 値を割り当てることができます。式の L 値が单一の sObject 型である場合、Apex は自動的にクエリ結果リストの1つの sObject レコードに L 値を割り当てます。リスト内に sObjects がない場合、または複数の sObject がある場合、実行時例外が発生します。次に例を示します。

## null 値検索の回避によるパフォーマンスの改善

SOQL クエリおよびSOSL クエリで、null 値を含むレコードの検索を回避します。パフォーマンスを改善するために、まず null 値を除外します。次の例では、 の値が null であるすべてのレコードが返される値から除外されます。

## 外部キーおよび親 - 子リレーションの SOQL クエリについて

SOQL クエリのステートメントは、外部キーや親 - 子レコードの結合などの有効な SOQL ステートメントとして使用できます。外部キーの結合が含まれている場合、生成される sObjects は、通常の項目表記を使用して参照できます。次に例を示します。

また、sObjects での親 - 子リレーションは SOQL クエリとして動作します。次に例を示します。

## SOQL クエリの多態的なリレーションの処理

多態的なリレーションは、参照されるオブジェクトに複数の異なる種別を使用できるオブジェクト間のリレーションです。たとえば、Event の What リレーション項目には Account、Campaign、Opportunity のいずれかを使用できます。

Apex で多態的なリレーションの SOQL クエリを使用する方法についての説明を次に示します。多態的なリレーションについてのより一般的な情報は、『Force.com SOQL and SOSL Reference』の「[多態的なキーとリレーションについて](#)」を参照してください。

Apex で多態的な項目を参照する SOQL クエリを使用して、多態的な項目によって参照されるオブジェクト種別に依存する結果を取得できます。1つのアプローチとして、修飾子を使用して結果を絞り込むという方法があります。次の例では、What 項目を使用して Account または Opportunity に関連する Event をクエリします。

別のアプローチとして、SOQL のステートメントで句を使用する方法があります。この例でも、What 項目を使用して Account または Opportunity に関連する Event をクエリします。

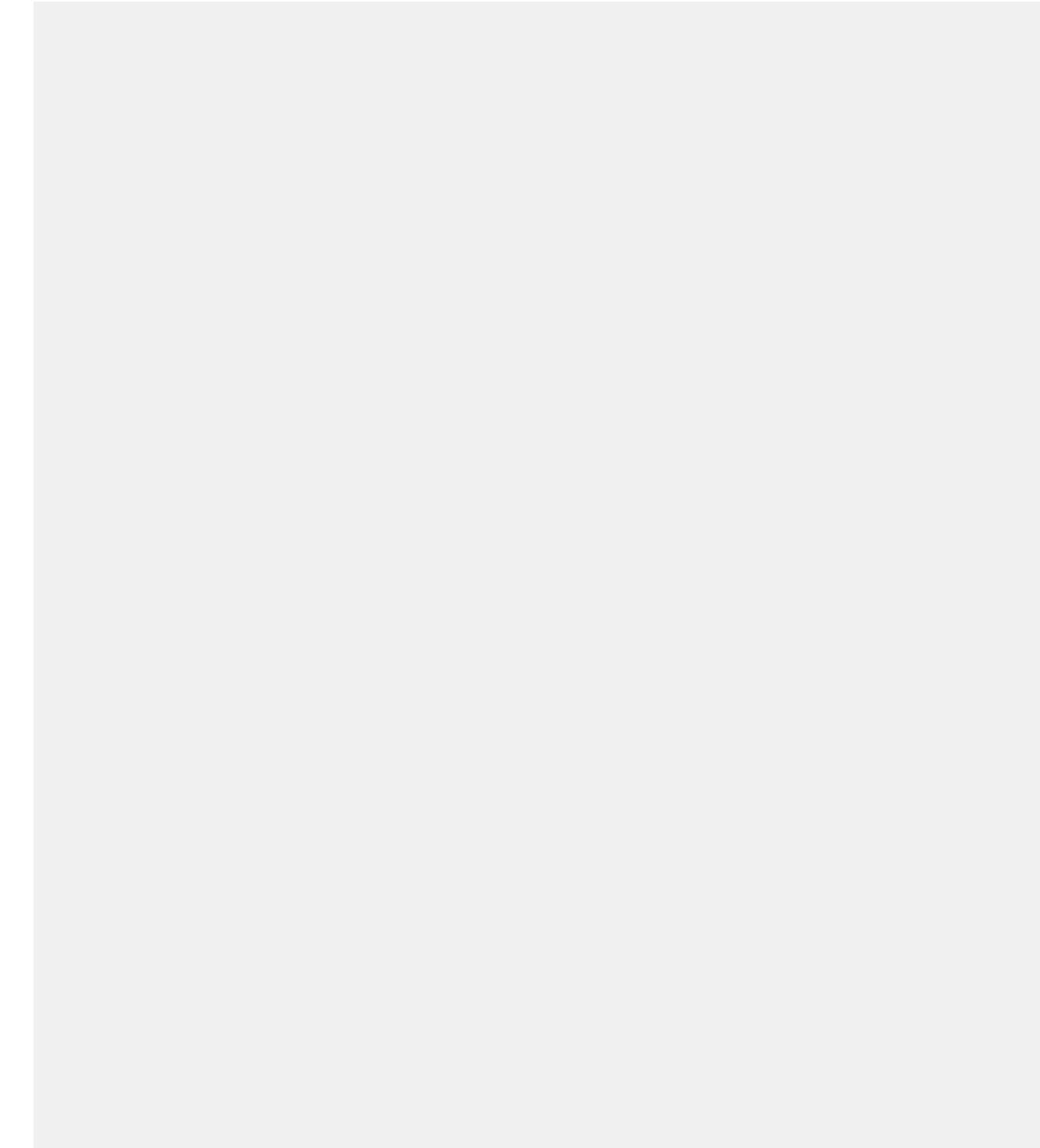


メモ: は、現在 SOQL 多態性機能の一部の開発者プレビューとして利用可能です。組織での有効化については、salesforce.com にお問い合わせください。

これらのクエリは、リレーション項目が目的のオブジェクト種別を参照する sObject のリストを返します。

多態的なリレーションで参照されるオブジェクトにアクセスする必要がある場合は、オブジェクト種別を判断するために instanceof キーワードを使用できます。次の例では、を使用して、Account または Opportunity が Event に関連しているかどうかを判断します。

別のメソッドに渡す前に、クエリが返す参照される sObject を適切な種別の変数に割り当てる必要があります。次の例では、句を含む SOQL クエリを使用して Merchandise\_c カスタムオブジェクトの User または Group 所有者をクエリし、を使用して所有者の種別を判断してから、ユーティリティメソッドに渡す前に所有者オブジェクトを User または Group の種別の変数に割り当てます。



## SOQL クエリおよびSOSL クエリでの Apex 変数の使用

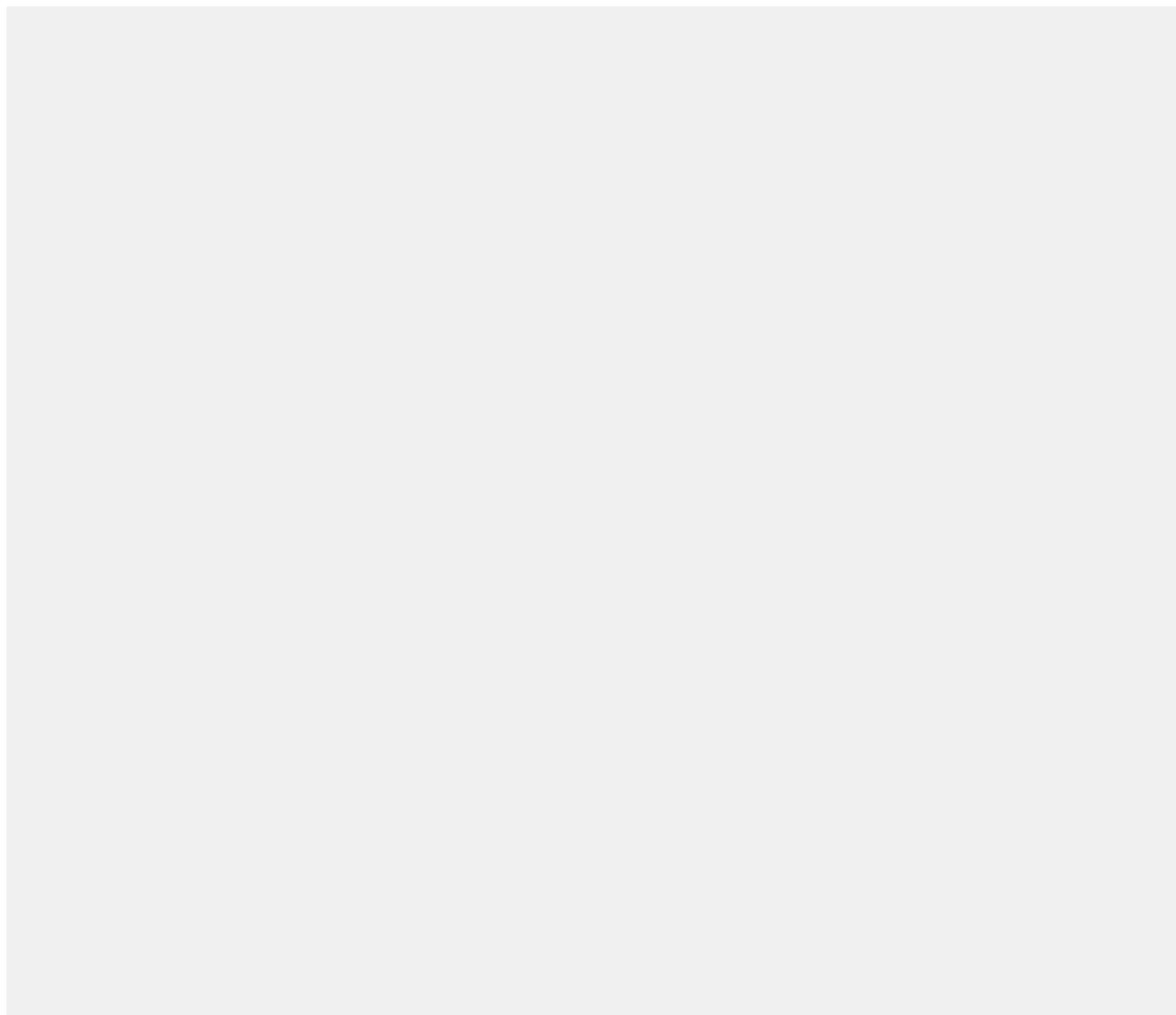
Apex の SOQL ステートメントと SOSL ステートメントは、前にコロン( )がある場合、Apex コード変数と式を参照できます。このように SOQL ステートメントまたは SOSL ステートメント内でローカルコード変数を使用

することを、バインドと呼びます。Apex パーサーは、SOQL ステートメントまたは SOSL ステートメントを実行する前に、最初にコードコンテキスト内のローカル変数を評価します。バインド式は、次のように使用できます。

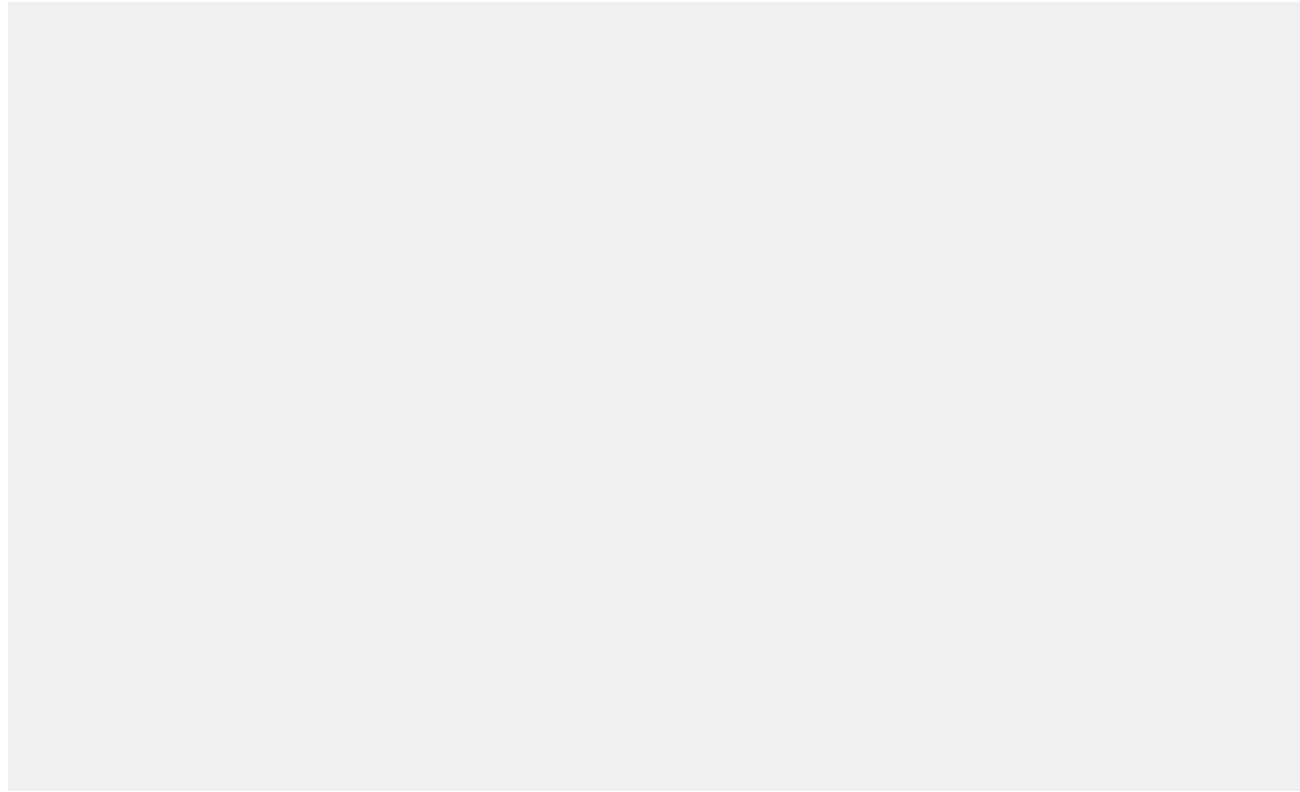
- ・ 句の検索文字列
- ・ 句の条件リテラル
- ・ 句の 演算子または 演算子の値。値の動的セットを絞り込むことができます。いずれのデータ型のリストでも機能しますが、特に ID または String のリストで使用されます。
- ・ 句のディビジョン名
- ・ 句の数値
- ・ 句の数値

バインド式は などの他の句と共に使用することはできません。

次に例を示します。







## SOQLステートメントを使用したすべてのレコードのクエリ

SOQLステートメントは、キーワードを使用して、削除されたレコードやアーカイブされた活動など、組織内のすべてのレコードをクエリできます。次に例を示します。

を使用して、組織のごみ箱の中のレコードをクエリできます。  
キーワードと共に使用することはできません。

## Apexでのデータの使用

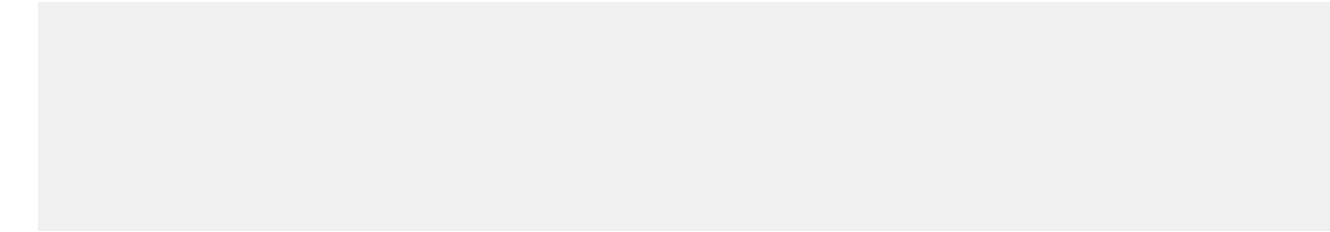
Apexは、Force.com プラットフォームの永続レイヤと緊密に統合されています。データベースのレコードは、Apexで単純なステートメントを使用して直接挿入および操作できます。管理者がデータベースのレコードを追加および管理できる Apex の言語を、データ操作言語 (DML) と呼びます。読み取り操作 (レコードのクエリ) に使用される SOQL 言語とは異なり、DML は書き込み操作に使用されます。

レコードの挿入または操作を行う前に、レコードデータが sObject としてメモリ内に作成されます。sObject データ型は汎用データ型で、レコードデータを保持する変数のデータ型に対応します。sObject データ型のサブタイプとなる特定のデータ型が存在します。これらは、標準オブジェクトレコード (Account や Contact など) やカスタムオブジェクト (Invoice\_Statement\_\_c など) のデータ型に対応します。通常、これらの特定の sObject データ型

を使用します。ただし、sObject のデータ型を事前に把握していない場合は、汎用 sObject データ型を使用できます。次に、新しい特定の Account sObject を作成して変数に割り当てる方法の例を示します。

前の例では、変数 `newAccount` で参照される取引先が必須の `Id` 項目によりメモリ内に存在しています。ただし、これは Force.com プラットフォームの永続レイヤにはまだ保持されていません。DML ステートメントをコールして、sObject をデータベースに保持する必要があります。次に、`insert` ステートメントを使用してこの取引先を作成および保持する例を示します。

また、すでに挿入されているレコードを DML を使用して変更することもできます。実行できる操作は、レコードの更新、レコードの削除、ごみ箱からのレコードの復元、レコードのマージ、リード取引の開始です。レコードをクエリすると、変更してその変更を保持できる sObject インスタンスを取得します。次に、以前に保持されている既存のレコードをクエリして、メモリ内でこのレコードの sObject の表示に関するいくつかの項目を更新し、その変更をデータベースに保持する例を示します。



## 関連リンク

[sObject 型](#)

## DML ステートメントと Database クラスメソッド

Apex には、DML 操作の実行方法として、DML ステートメントを使用する方法と Database クラスメソッドを使用する方法の 2 通りがあります。このため、データ操作の実行方法が柔軟になります。DML ステートメントは使いやすいため例外が発生しますが、コード内で処理することができます。Database クラスメソッドでは、エラー処理をより詳細に制御できるため、レコードの部分的な処理が可能です。

DML ステートメントと Database クラスメソッドのどちらを使用するかを決めるには、次の点を参考にしてください。

- DML 一括処理中に発生するエラーを、コントロールフローをその場で中断する Apex 例外として処理する場合、DML ステートメントを使用します。ここではプロックを使用します。この動作は、ほとんどのデータベース手続き型言語での例外の処理方法に似ています。
- DML 一括操作の部分的な完了を可能にする場合は、Database クラスメソッドを使用します。レコードが失敗した場合でも、DML 操作の残りは終了できます。アプリケーションは拒否されたレコードを確認でき、可能であれば操作を再試行します。この形式を使用すると、DML 例外エラーが発生することがないコードを書くことができます。エラーが発生しない代わりに、作成したコードでは、成功または失敗を判断するための適切な結果配列を使用できます。Database クラスメソッドには、DML ステートメントに類似する、発生した例外をサポートする構文も含まれます。



### メモ:

この 2 つの方法ではほとんどの操作が重複していますが、次の点は異なります。

- 操作は DML ステートメントでのみ使用でき、Database クラスメソッドでは使用できません。
- は Database クラスメソッドでのみ使用でき、DML ステートメントでは使用できません。
- Database クラスには、メソッドのトランザクションの制御とロールバック、ごみ箱を空にする、SOQL クエリに関連するメソッドなど、DML ステートメントでは使用できないメソッドも備えられています。

## アトミックトランザクションとしての DML 操作

DML 操作はトランザクション内で実行されます。トランザクションの実行には、すべての DML 操作が正常に完了することが求められます。いずれかの操作でエラーが発生した場合はトランザクション全体がロールバックされます。この場合、データは一切データベースにコミットされません。トランザクションの境界は、トリガ、クラスメソッド、匿名のコードブロック、Apex ページ、カスタム Web サービスメソッドのいずれかにすることができます。

トランザクション境界内部で発生するすべての操作は、操作の 1 つの単位に相当します。これは、トランザクション境界内で実行されたコードの結果として起動されたクラスやトリガなど、トランザクション境界から外部コードへのコールにも適用されます。たとえば、カスタム Apex Web サービスメソッドによってクラスのメソッドがコールされ、そのメソッドがいくつかの DML 操作を実行するという連続した操作があるとします。この場合、トランザクション内のすべての操作がエラーなしで実行を完了した後にのみ、すべての変更がデータベースにコミットされます。中間ステップのいずれかでエラーが発生した場合、すべてのデータベース変更はロールバックされ、トランザクションはコミットされません。

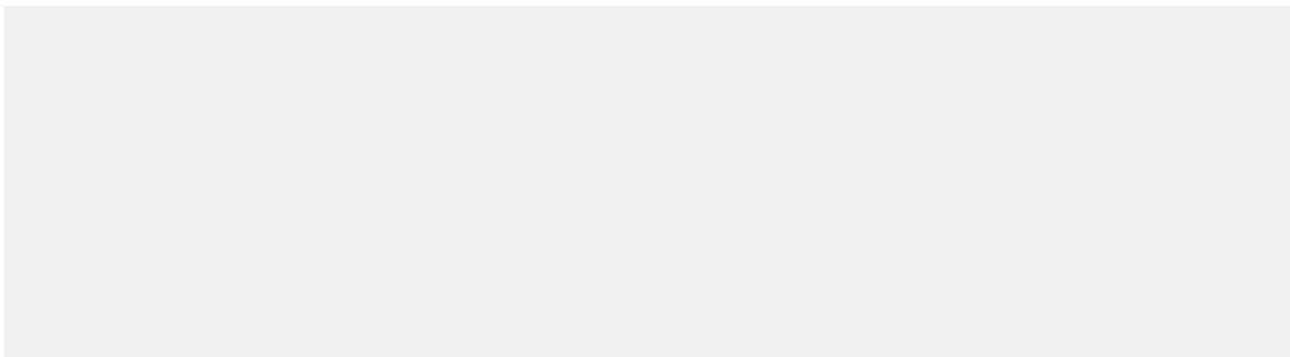
## DML の仕組み

### 単一 DML 操作と一括 DML 操作

DML 操作は、単一の sObject で行うことも、sObject のリストで一括で行うこともできます。一括 DML 操作ではガバナ制限 (Apex トランザクションごとのステートメント数を 150 件に制限する DML 制限など) に達することを防止できるため、この操作を実行することをお勧めします。この制限は、Force.com マルチテナントプラットフォームの共有リソースに公正にアクセスできるようにするために設定されています。sObject のリストで DML 操作を実行すると、sObject ごとに 1 つのステートメントとしてカウントされるのではなく、リストのすべての sObject が 1 つの DML ステートメントとしてカウントされます。

次に、単一の sObject で DML コールを実行する非効率な例を示します。

for ループで取引先責任者を 1 つずつ反復処理し、Department 項目が特定の値に一致した場合に Description\_\_c 項目に新しい値を設定しています。リストに 150 を超える品目が含まれる場合、151 回目の update コールは、DML ステートメント制限の 150 を超えるため、キャッチできない例外を返します。



次の例は、ガバナ制限に達しないように前の例を変更したものです。取引先責任者のリストで `update` をコールして DML 操作を一括処理します。これは 1 つの DML ステートメントとしてカウントされるため、150 の制限をはるかに下回ります。

```
updatedList.add(con);
```

```
update updatedList;
```

DML 操作に影響する他のガバナ制限は、1 つのトランザクションの DML 操作で処理できる合計行数 (10,000 行) です。同じトランザクションの全 DML コールで処理されるすべての行は、この制限に対して増分的にカウントされます。たとえば、同じトランザクションで 100 人の取引先責任者を挿入して 50 人の取引先責任者を更新すると、DML で処理された合計行数は 150 行となり、残りは 9,850 行になります (10,000 - 150)。

## システムコンテキストと共有ルール

ほとんどの DML 操作はシステムコンテキストで実行され、現在のユーザの権限、項目レベルセキュリティ、組織の共有設定、ロール階層内での位置付け、および共有ルールを無視します。DML 操作が `with sharing` キーワードによって定義されたクラス内でコールされた場合のみ例外となり、現在のユーザの共有ルールが考慮されます。

DML 操作を匿名ブロック内で実行する場合、現在のユーザのオブジェクトレベルの権限と項目レベルの権限で実行されます。

## 関連リンク

[with sharing または without sharing キーワードの使用](#)

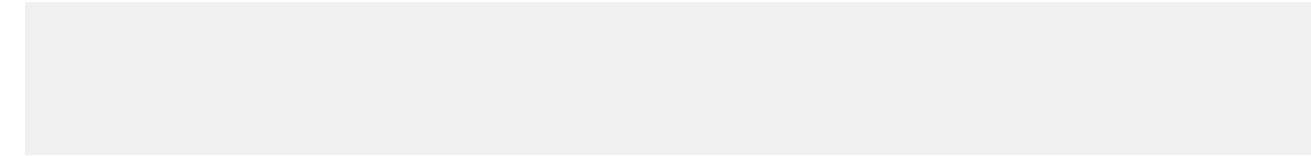
[匿名ブロック](#)

## DML 操作

### レコードの挿入と更新

DML を使用すると、新規レコードを挿入して、データベースにコミットできます。同様に、既存のレコードの項目値を更新することもできます。

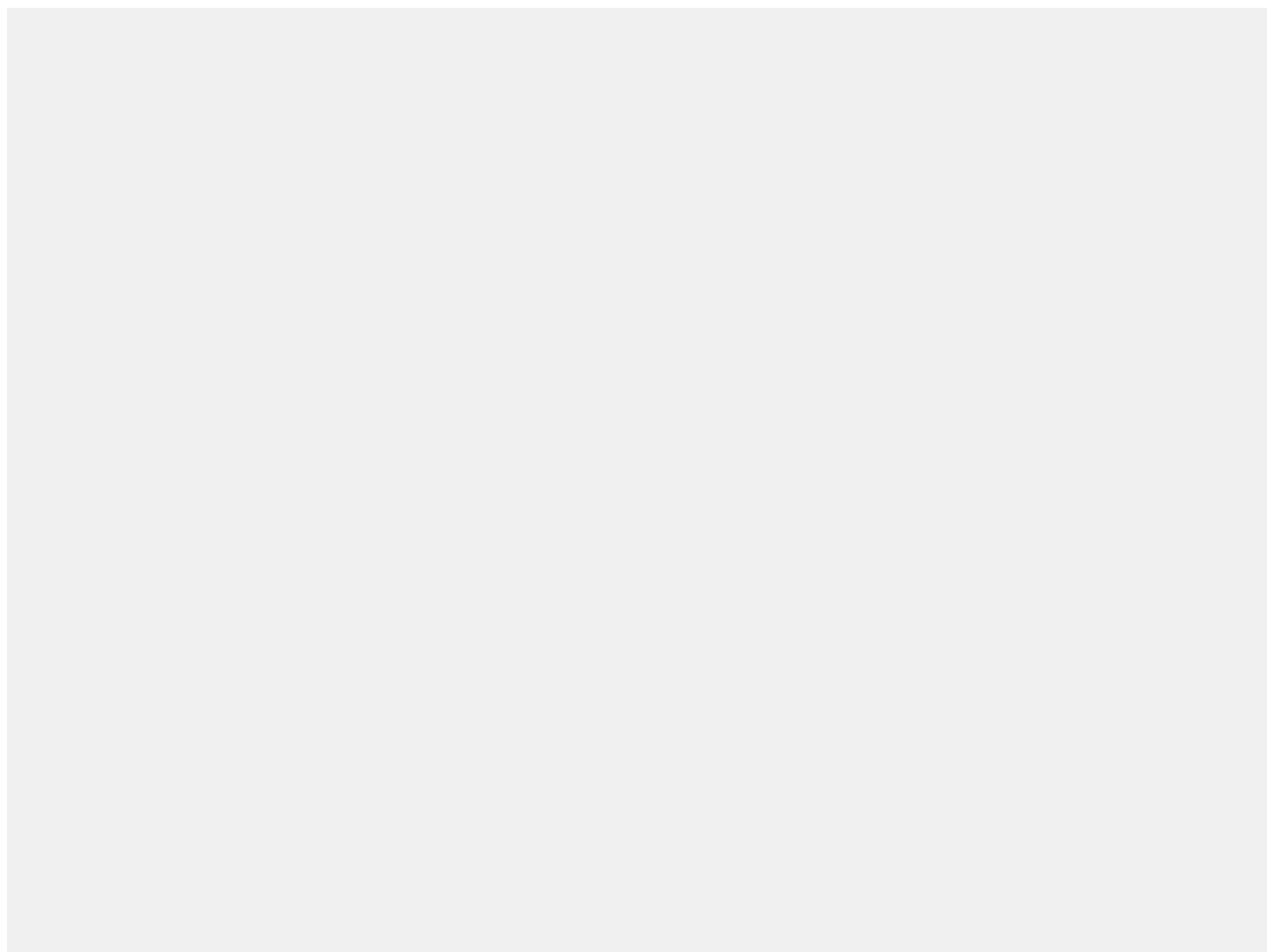
この例では、3つの取引先レコードを挿入して既存の取引先レコードを更新する方法を示します。まず、3つの Account sObject を作成してリストに追加します。次に、1つの `insert` ステートメントで取引先のリストを挿入して一括挿入を実行します。その後、2つ目の取引先レコードをクエリして請求先市区郡を更新し、`update` ステートメントをコールしてデータベースに変更を保持します。



### 関連レコードの挿入

2つのオブジェクト間のリレーション(参照関係や主従関係など)がすでに定義されている場合、既存のレコードに関連するレコードを挿入できます。レコードは、外部キー ID を使用して関連レコードに関連付けられます。この外部キー ID はマスタレコードにのみ設定できます。たとえば、新規取引先責任者を挿入する場合、項目の値を設定することで、取引先責任者の関連取引先レコードを指定できます。

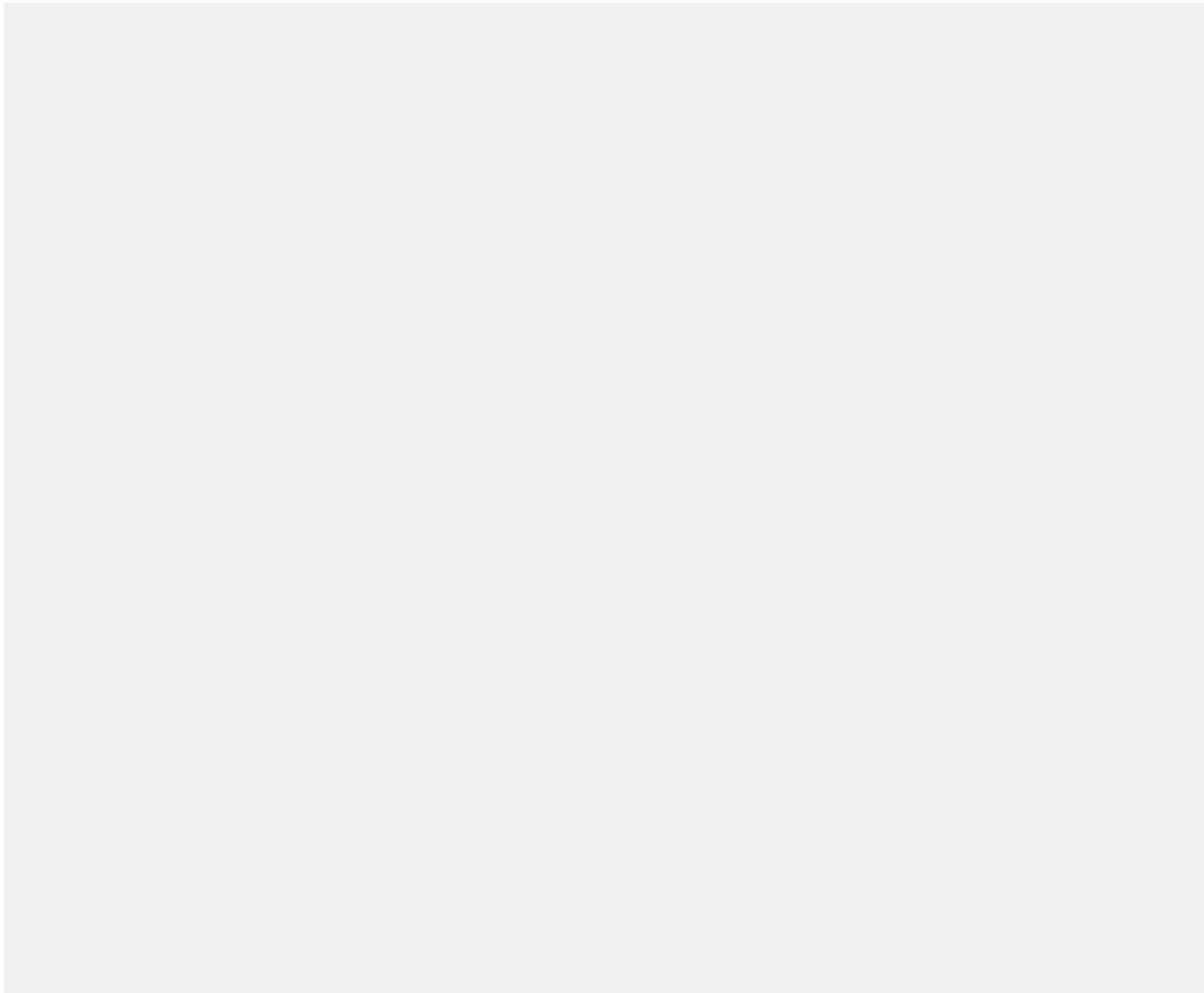
この例では、取引先責任者の `Account` 項目を設定して、取引先責任者を取引先(関連レコード)に追加する方法を示します。取引先責任者と取引先は参照関係でリンクされています。



### 関連レコードの更新

関連レコードの項目は、同じ DML 操作のコールでは更新できないため、別の DML コールが必要になります。たとえば、新規取引先責任者を挿入する場合、`Account` 項目の値を設定することで、取引先責任者の関連取引先レコードを指定できます。ただし、別の DML コールを使用して取引先自体を更新しない場合、取引先の名前

を変更することはできません。同様に、取引先責任者を更新するときに、取引先責任者の関連取引先も更新する場合は、2つの DML コールを作成する必要があります。次の例では、2つのステートメントを使用して取引先責任者とその関連取引先を更新しています。



### 外部キーを使用して1つのステートメントで親レコードと子レコードを作成する

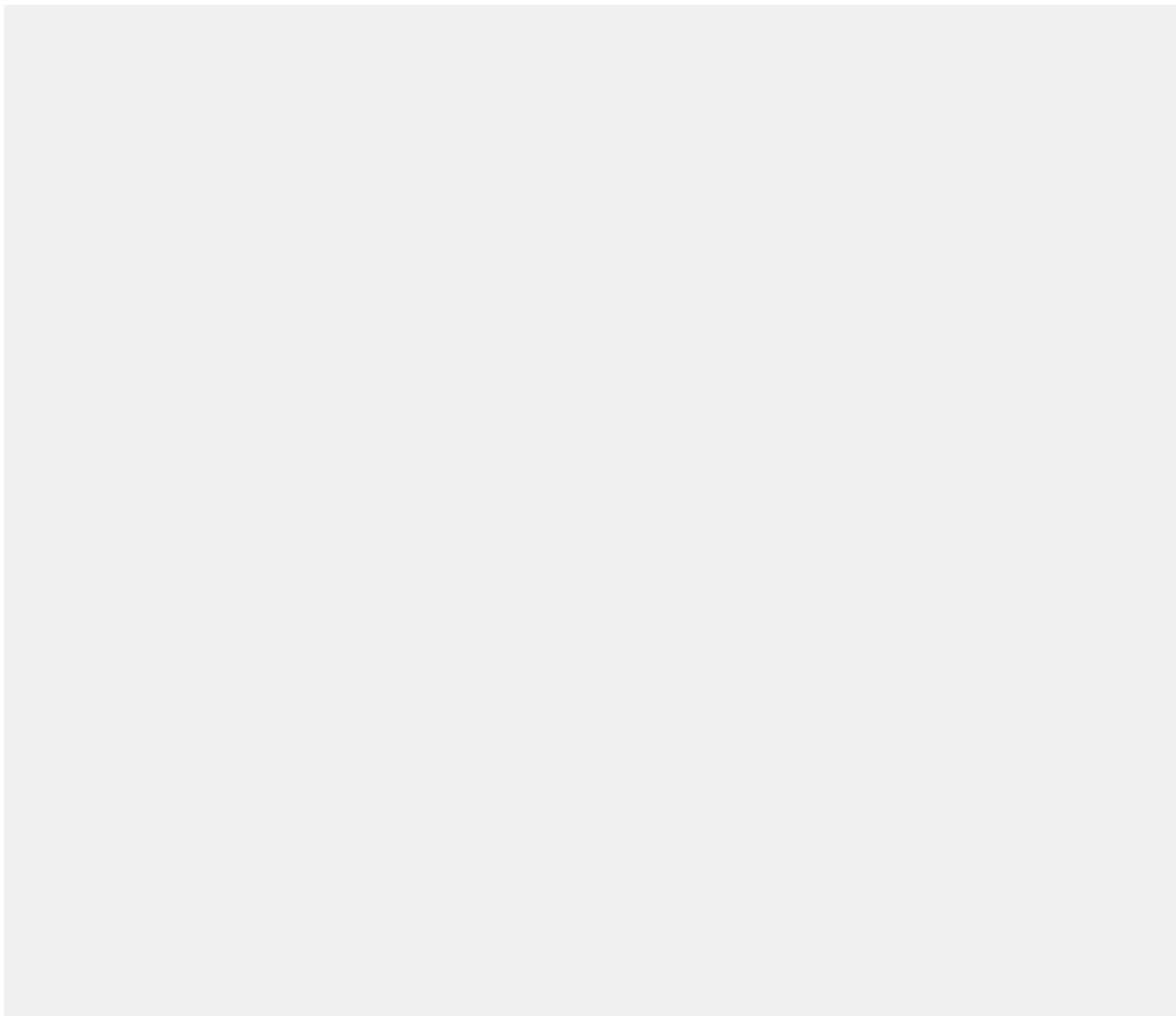
外部キーとして外部 ID 項目を使用することによって、最初に親レコードを作成して、その ID をクエリしてから子レコードを作成するのではなく、他の種別の sObject の親レコードおよび子レコードを1つの手順で作成することができます。手順は、次のとおりです。

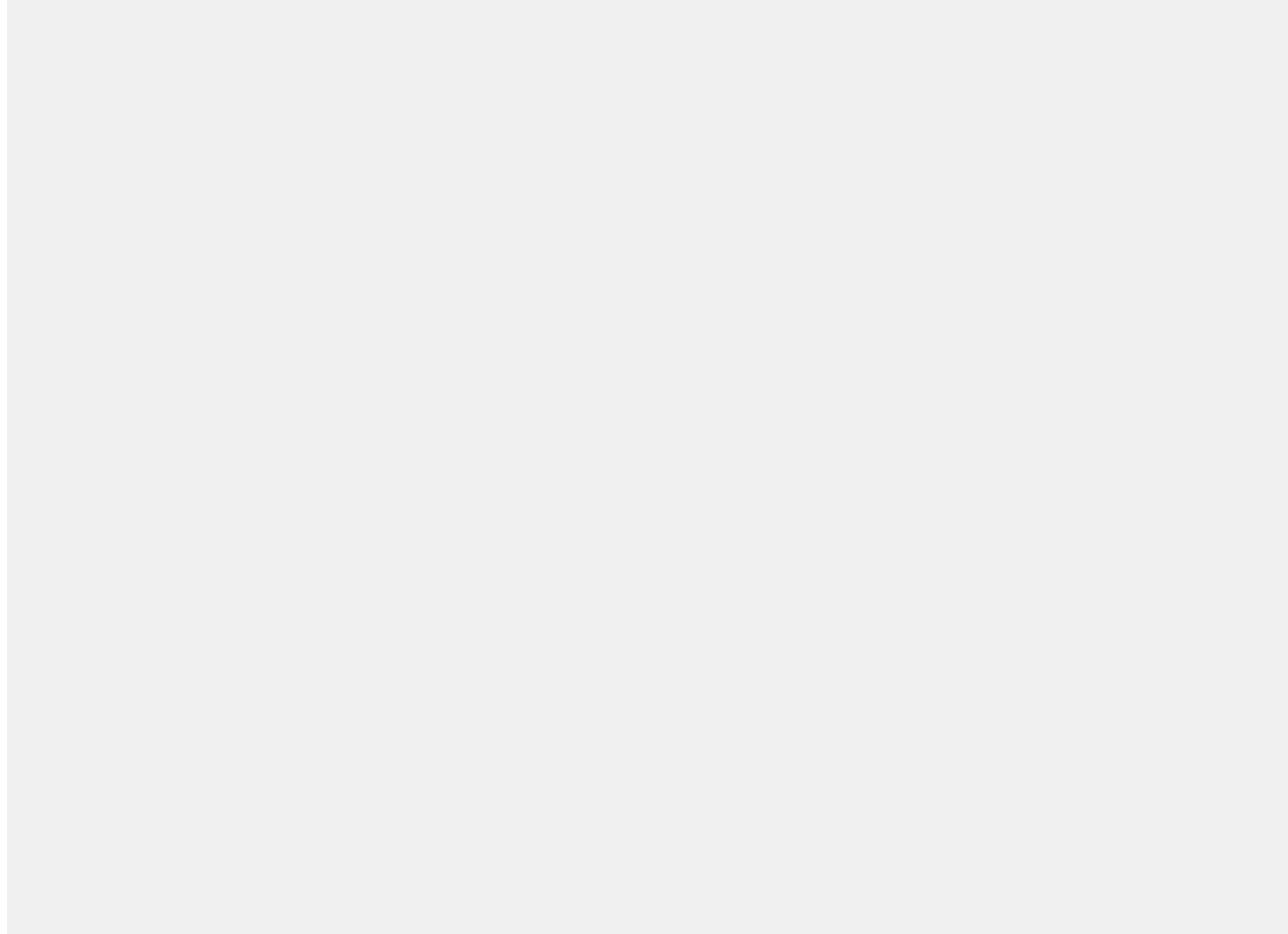
- ・ 子 sObject を作成し、必須項目 (必要に応じて、その他の項目) を入力します。
- ・ 子 sObject に対する親外部キー参照を設定するためにのみ使用される、親参照 sObject を作成します。この sObject には定義された外部 ID 項目のみがあり、その他の項目は設定されません。
- ・ 作成した親参照 sObject に子 sObject の外部キー項目を設定します。
- ・ ステートメントに渡すその他の親 sObject を作成します。この sObject には、外部 ID 項目に加えて、必須項目 (必要に応じて、その他の項目) を設定する必要があります。

- 作成する sObject の配列を渡して、**【関連レコードを作成】** をコールします。親 sObject は配列の子 sObject の前に付ける必要があります、つまり、親の配列インデックスは子のインデックスより小さい必要があります。

最大 10 レベルの深度で関連レコードを作成できます。また、1回のコールで作成される関連レコードには、他の種別の sObject が必要です。詳細は、『SOAP API 開発者ガイド』の「[オブジェクト種別が異なるレコードの作成](#)」を参照してください。

次の例では、1回の **【取引先関連】** ステートメントで親取引先に関連する商談を作成する方法を示します。この例では、商談 sObject を作成し、その項目のいくつかに入力してから、2 つの取引先オブジェクトを作成します。最初の取引先は外部キーリレーションのみであり、2 番目の取引先は取引先作成のためのものであり、取引先項目が設定されています。両方の取引先には外部 ID 項目 **【外部 ID】** が設定されています。次に、このサンプルでは、sObject の配列を渡して、**【関連レコードを作成】** をコールします。配列の最初の要素は親 sObject で、2 番目は商談 sObject です。**【取引先関連】** ステートメントでは、1 つの手順で商談とその親取引先を作成します。最後に、このサンプルでは、結果を確認し、作成されたレコードの ID をデバッグログに書き込むか、レコードの作成が失敗した場合は最初のエラーを書き込みます。このサンプルでは、MyExtID という取引先に外部 ID テキスト項目が必要です。





## レコードの更新/挿入

操作を使用すると、既存のレコードの挿入または更新を1つのコールで実行できます。ステートメントまたはデータベースメソッドでは、レコードがすでに存在しているかどうかを確認するために、レコードのIDまたはカスタム外部ID項目値(指定されている場合)をレコードを照合するキーとして使用します。

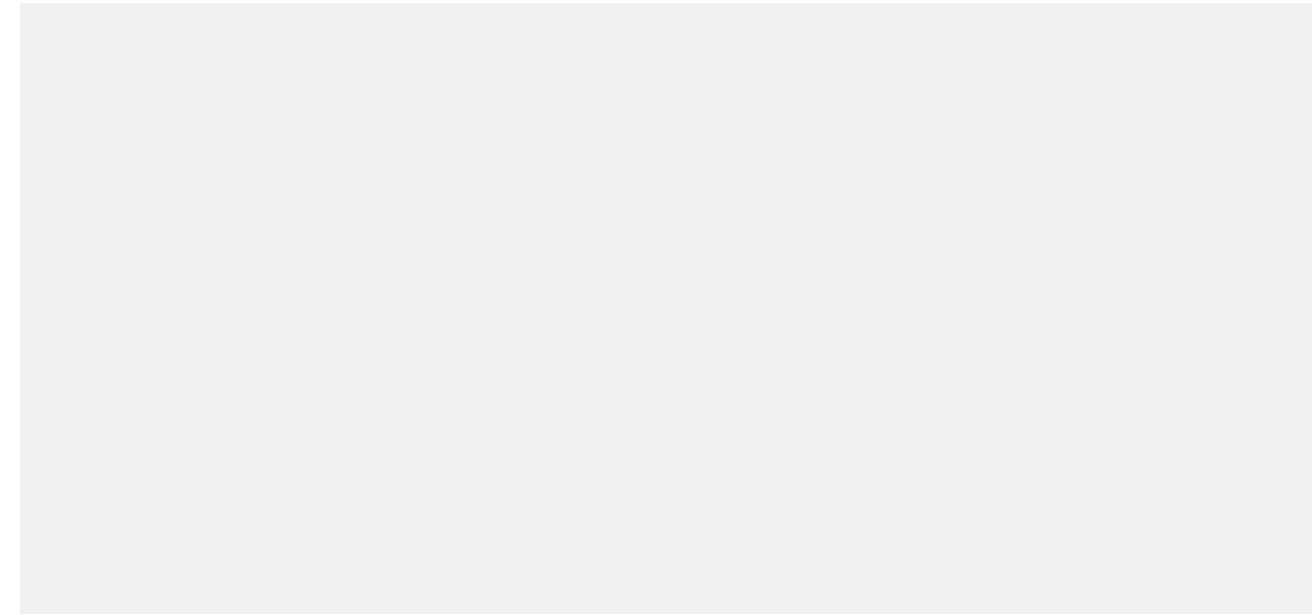
- キーが一致しない場合、新規オブジェクトレコードが作成されます。
- キーが一度だけ一致したら、既存のオブジェクトレコードが更新されます。
- キーが複数回一致する場合は、エラーが生成され、オブジェクトレコードは挿入も更新もされません。



メモ: カスタム項目に、項目定義の一部として[ユニーク]と[「ABC」と「abc」を値の重複として扱う(大文字と小文字を区別しない)]属性が選択されている場合のみ、カスタム項目による照合では大文字と小文字を区別しません。この場合、「ABC123」は「abc123」と一致します。詳細は、「カスタム項目の作成」を参照してください。

## 例

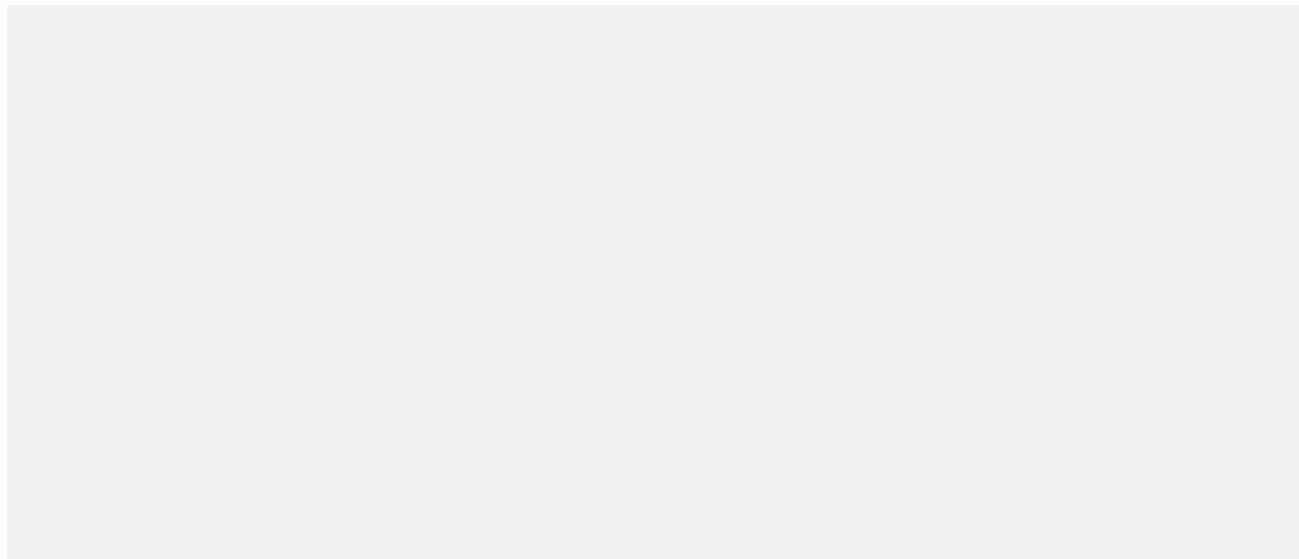
次の例では、以前 Bombay となっていた市に所在するすべての既存取引先の市の名前を更新し、さらに、San Francisco に所在していた新規取引先を挿入します。

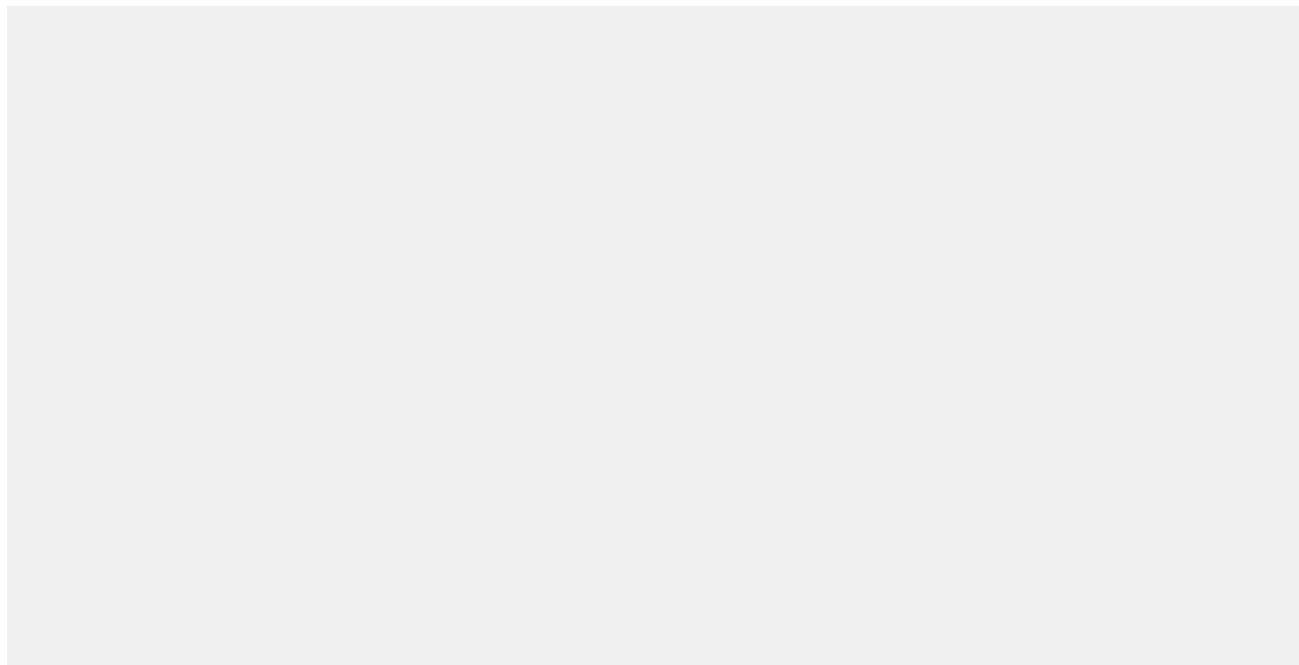
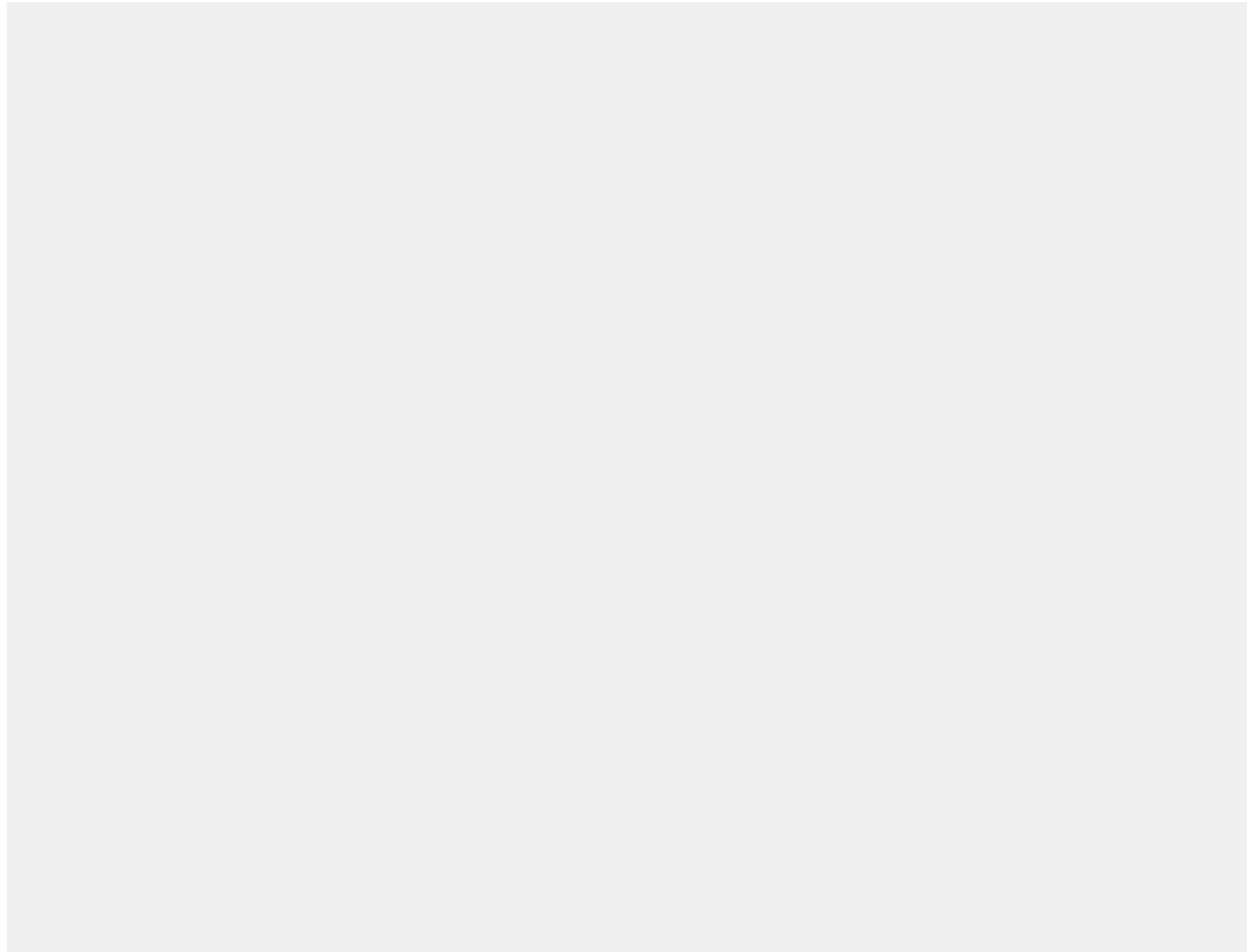


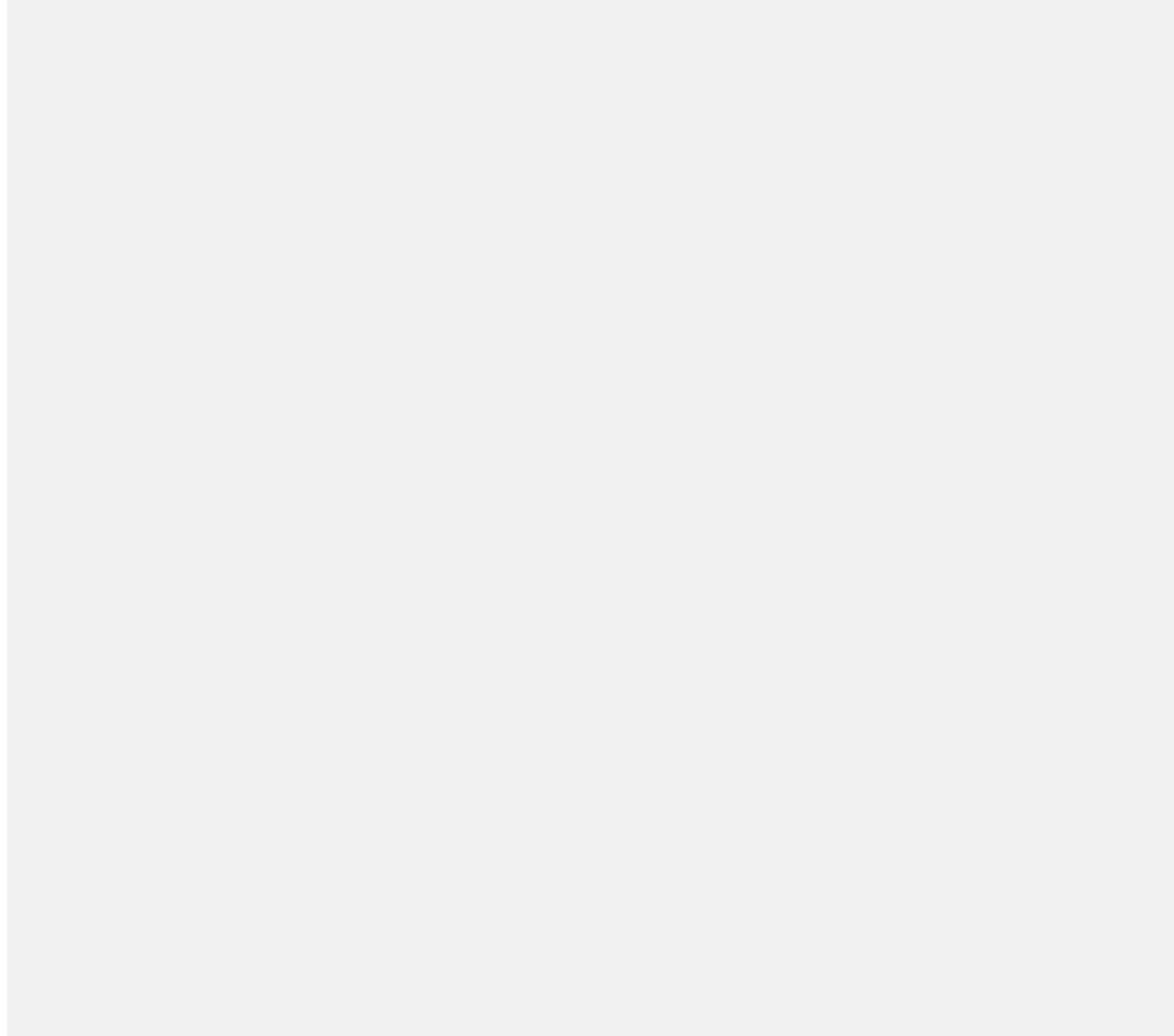
メモ:

の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

次の例では、メソッドを使用して、渡されるリードのコレクションを更新/挿入します。この例では、レコードの部分処理を許可しています。つまり、一部のレコードが処理に失敗した場合でも、残りのレコードは引き続き挿入または更新されます。また、結果を反復処理して、正常に処理された各レコードに新しいToDo を追加します。TodosObject は、リストに保存され、その後一括挿入されます。この例の後に、この例をテストするテストメソッドを含むテストクラスが続きます。



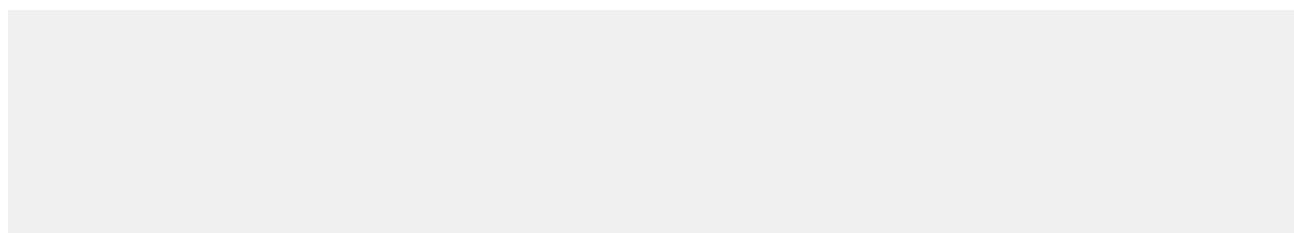


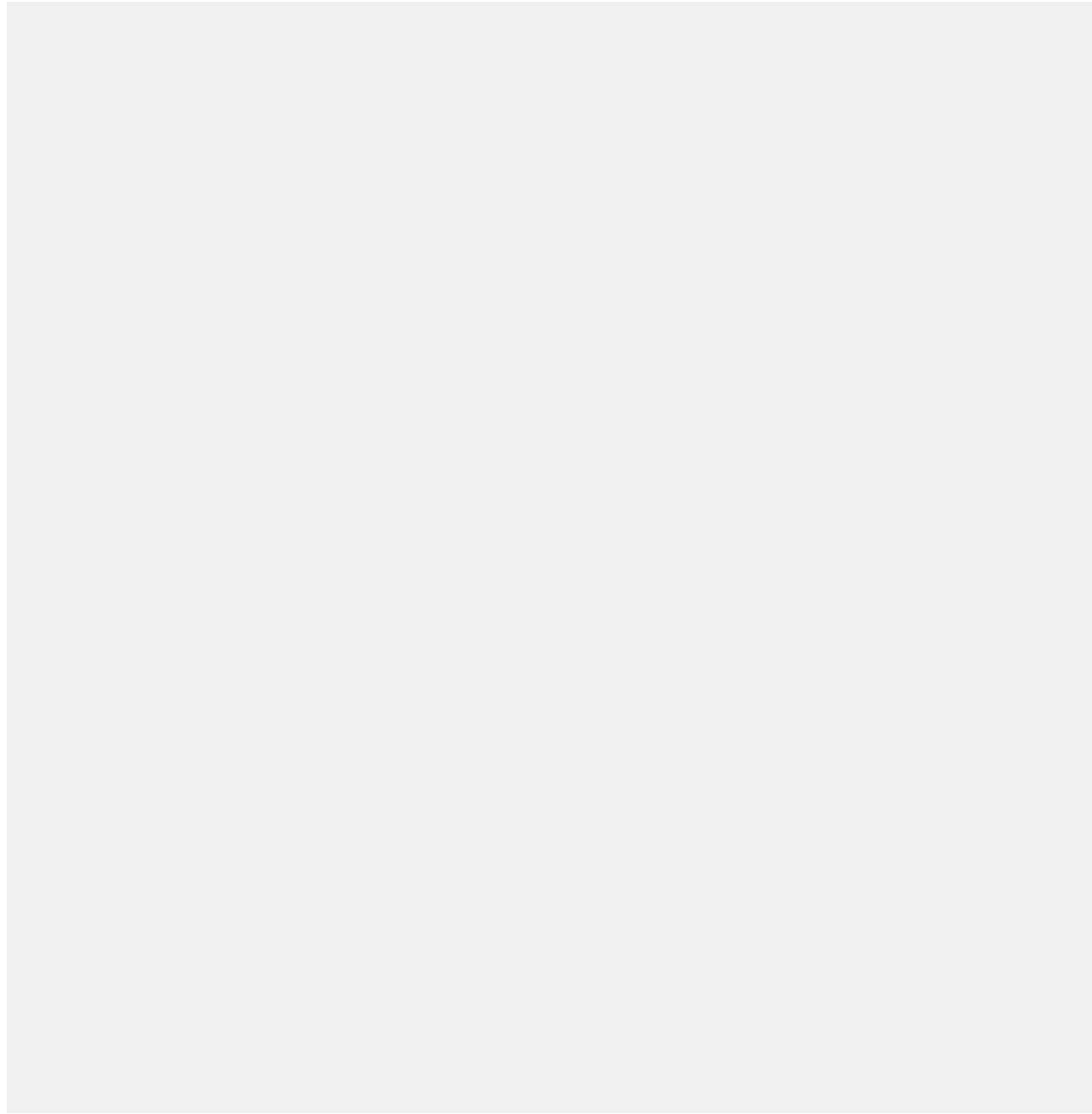


を外部 ID と一緒に使用すると、コード内の DML ステートメントの数が減少し、ガバナ制限に該当しないようにします（「[実行ガバナと制限について](#)」を参照）。この次の例では、納入商品と商談品目間の一対一の関係を維持するために、Asset オブジェクトの `ExternalId` と外部 ID 項目 `ExternalId` を使用します。



メモ: このサンプルを実行する前に、Asset オブジェクト上に `ExternalId` という名前でカスタム テキスト項目を作成し、外部 ID としてマークします。カスタム項目についての詳細は、Salesforce オン ラインヘルプを参照してください。





## レコードのマージ

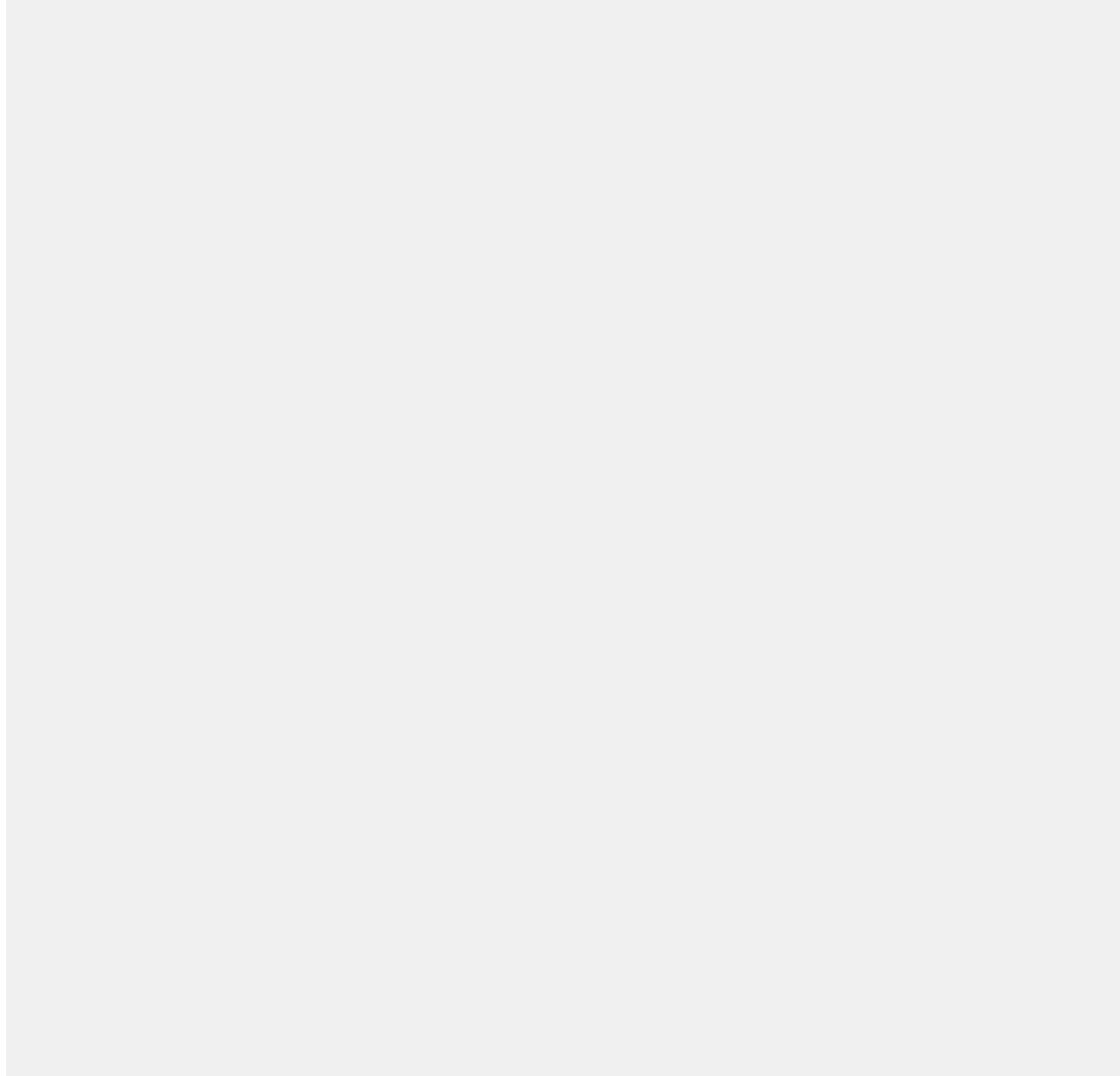
データベースのリード、取引先責任者、取引先レコードが重複している場合、データをクリーンアップしてレコードを統合することをお勧めします。同じ sObject 型のレコードを 3 つまでマージできます。操作は、最大 3 つのレコードを 1 つのレコードにマージし、他のレコードを削除してから、関連レコードを再ペアレント化します。

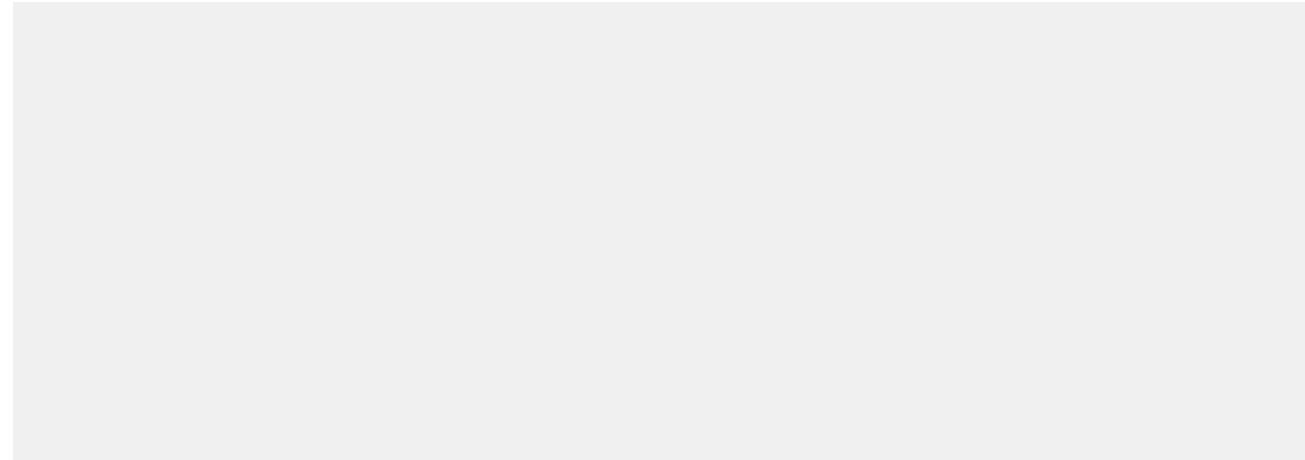


メモ: この DML 操作には、一致するデータベースシステムメソッドはありません。

### 例

次に、既存の取引先レコードを主取引先にマージする方法を示します。マージする取引先には、マージ操作後に主取引先レコードに移動される関連取引先責任者があります。また、マージするレコードはマージ後に削除され、1つのレコードのみがデータベースに残ります。この例では、2つの取引先のリストを作成してからリストを挿入します。次に、クエリを実行して新規取引先レコードをデータベースから取得し、マージする取引先に取引先責任者を追加します。その後、2つの取引先をマージします。最後に、取引先責任者が主取引先に移動していく、2つ目の取引先が削除されていることを確認します。





### マージに関する考慮事項

sObject レコードをマージする場合、次のルールとガイドラインを考慮する必要があります。

- リード、取引先責任者、および取引先のみがマージ可能です。「[DML 操作をサポートしない sObject](#)」(ページ 126)を参照してください。
- 1つのメソッドには、1つの主レコードと最大 2 つのその他の sObject レコードを渡すことができます。

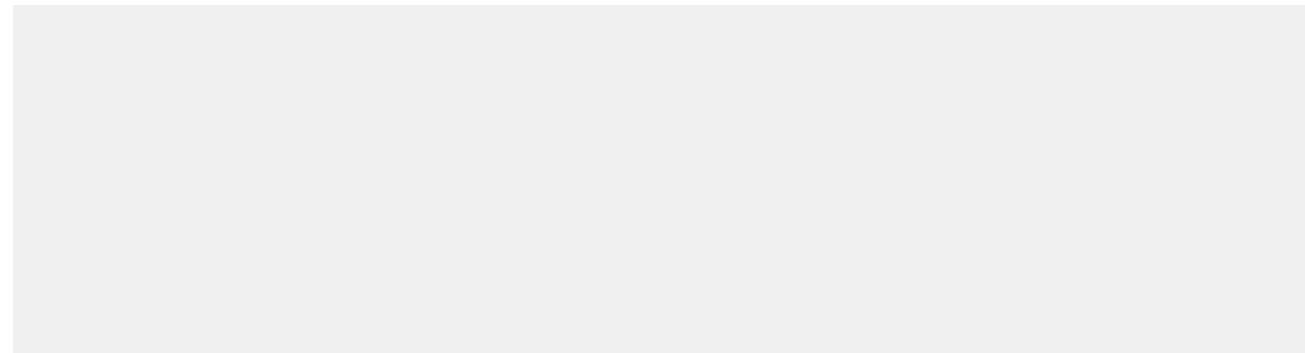
リード、取引先責任者、および取引先のマージについての詳細は、Salesforce オンラインヘルプを参照してください。

## レコードの削除

データベースにレコードを保持したら、操作を使用してそれらのレコードを削除できます。レコードを削除しても Force.com から完全に削除されるわけではなく、復元できるように 15 日間はごみ箱に置かれます。削除したレコードの復元については、後のセクションで説明します。

### 例

次の例では、「DotCom」という名前のすべての取引先を削除しています。





メモ: の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

### レコードを削除および復元するときの参照整合性

操作では、カスケード削除がサポートされています。親オブジェクトを削除すると、各子レコードが削除可能な場合は自動的に削除されます。

たとえば、ケースレコードを削除すると、Apex はケースに関連付けられたすべての CaseComment、CaseHistory、および CaseSolution レコードを自動的に削除します。ただし、特定の子レコードが削除可能でない場合、または現在使用中の場合、親ケースレコードの 操作は失敗します。

操作を行うと、次のリレーションの種類に関して、レコードの関連付けが復元されます。

- ・ 親取引先 (取引先の 親取引先 項目で指定)
- ・ 親ケース (ケースの 親ケース 項目で指定)
- ・ 翻訳ソリューションのマスタソリューション (ソリューションの マスタソリューション 項目で指定)
- ・ 取引先責任者のマネージャ (取引先責任者の 上司 項目で指定)
- ・ 納入商品に関連付けられている商品 (納入商品の 商品 項目で指定)
- ・ 見積に関連付けられている商談 (見積の 商談 項目で指定)
- ・ すべてのカスタム参照関係
- ・ 取引先およびリレーショングループのリレーショングループメンバー (一部例外あり)
- ・ タグ
- ・ 記事のカテゴリ、公開状態、割り当て



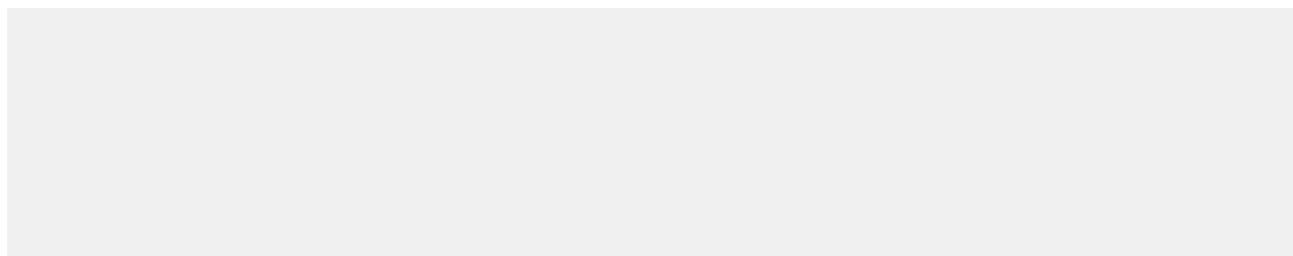
メモ: Salesforce は、置換されていない参照関係のみを復元します。たとえば、納入商品が、元の商品レコードが元に戻される前に別の商品と関連付けられている場合、その納入商品と商品のリレーションは復元されません。

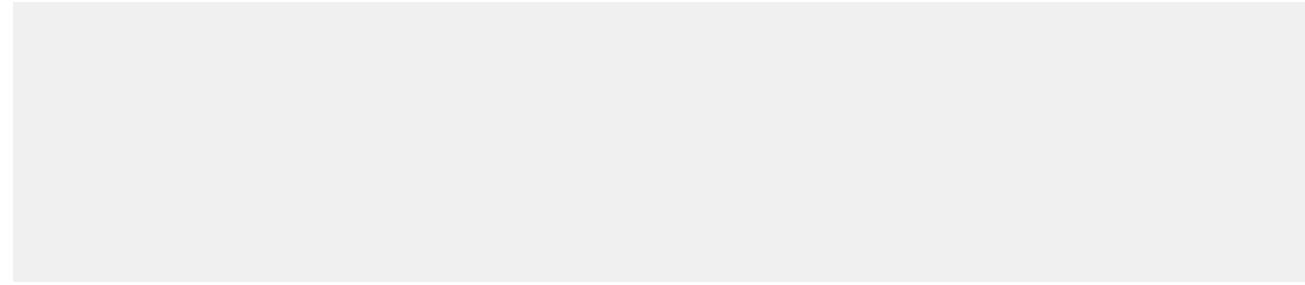
## 削除したレコードの復元

レコードを削除しても 15 日間はごみ箱に置かれ、その後で完全に削除されます。レコードがごみ箱にある間は、操作を使用して復元できます。これは、保持するレコードを誤って削除した場合などに便利です。

### 例

次の例では、「Trump」という名前の取引先を復元しています。キーワードは、削除されたレコードやアーカイブ済みの活動を含め、最上位リレーションと集計リレーションの両方にあるすべての行をクエリします。





メモ: の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

## 復元に関する考慮事項

ステートメントを使用するときは、次の点に注意してください。

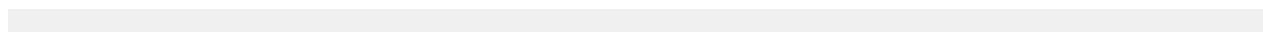
- マージの結果として削除されたレコードを復元できますが、子オブジェクトに再設定された親を元に戻すことはできません。
- マージの結果として削除されたレコードなど、削除されたレコードを識別するには、SOQL クエリでパラメータを使用します。「[SOQL ステートメントを使用したすべてのレコードのクエリ](#)」(ページ 96)を参照してください。
- [レコードを削除および復元するときの参照整合性](#)を参照してください。

## 取引の開始

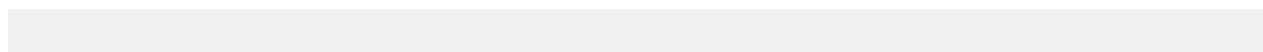
DML 操作は、リード取引開始によって取引先、取引先責任者、および(必要に応じて)商談を作成します。リードは、クラスのメソッドとしてのみ使用でき、DML ステートメントとしては使用できません。

リードの変換は、次の基本ステップに従います。

- アプリケーションは、変換されるリードの ID を確認します。
- 必要に応じて、アプリケーションはリードをマージする取引先の ID も確認します。アプリケーションは SOQL を使用し、リード名と一致する取引先を検索します。次に例を示します。



- 必要に応じて、アプリケーションはリードをマージする取引先責任者の ID も確認します。アプリケーションは SOQL を使用し、リードの取引先責任者名と一致する取引先責任者を検索します。次に例を示します。

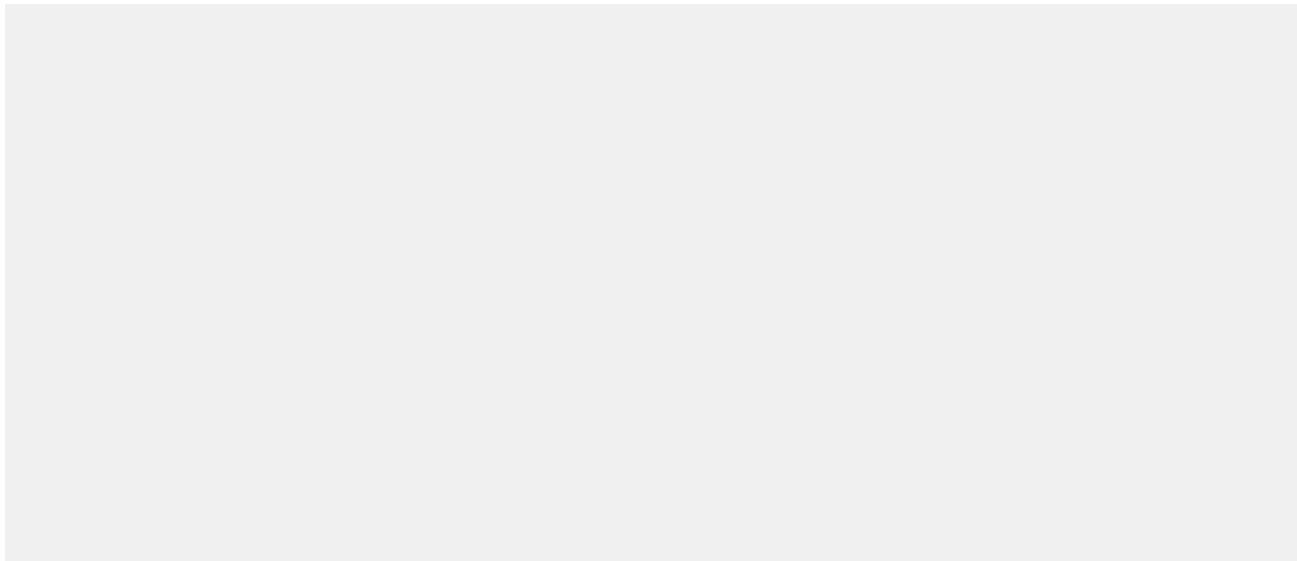


- 必要に応じて、アプリケーションはリードから商談を作成するかどうかを決定します。
- アプリケーションは LeadSource テーブルへのクエリを実行して、考えられるすべての取引開始後の状況オプション( )を取得し、取引開始後の状況の値を選択します。

6. アプリケーションは をコールします。
7. アプリケーションは返された結果の各 LeadConvertResult オブジェクトを繰り返し確認し、各リードの変換が成功したかどうかを確認します。
8. 必要に応じて、キューが所有するリードを変換する場合は所有者を指定する必要があります。これは、キューが取引先と取引先責任者を所有することができないためです。既存の取引先または取引先責任者を指定する場合も、所有者を指定する必要があります。

#### 例

この例では、 メソッドを使用してリード取引を開始する方法を示します。新規リードを挿入し、 オブジェクトを作成してその状況を取引開始済みに設定し、 メソッドに渡します。最後に、取引の開始が成功したことを確認します。



#### リード取引の開始に関する考慮事項

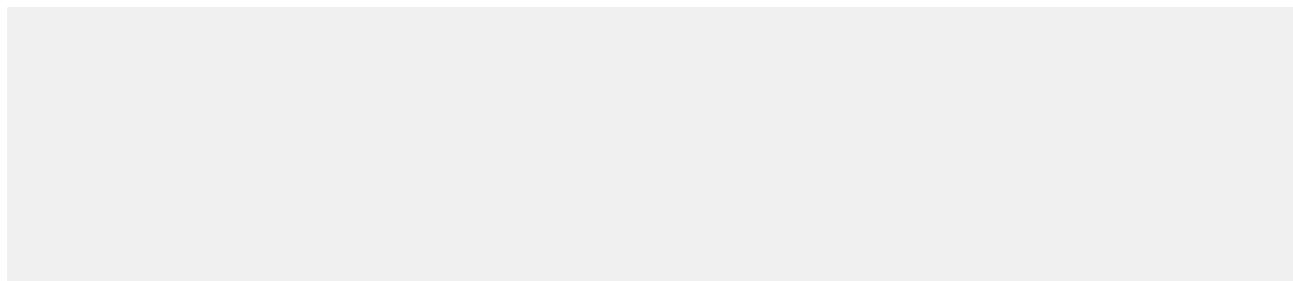
- **項目の対応付け:** システムは、リードの標準項目を取引先、取引先責任者、商談の標準項目に自動的に対応付けます。リードのカスタム項目に関しては、Salesforce システム管理者が、取引先、取引先責任者、商談のカスタム項目に対応づける方法を指定することができます。項目の対応付けについての詳細は、Salesforce オンラインヘルプを参照してください。
- **差し込み項目:** データが既存の取引先や取引先責任者オブジェクトにマージされる場合、変換先オブジェクトの空の項目のみが上書きされ、ID を含めた既存データは上書きされません。唯一の例外は、LeadConvert オブジェクトの `mergeLead` を True に設定している場合です。この場合、変換先の取引先責任者オブジェクトの `leadId` 項目が、変換元の LeadConvert オブジェクトの `leadId` 項目の内容で上書きされます。
- **レコードタイプ:** 組織でレコードタイプを使用している場合、新しい所有者のデフォルトのレコードタイプはリード変換時に作成されたレコードに割り当てられます。リードを変換するユーザのデフォルトのレコードタイプによって、変換時に使用できるリードの変換元値が決まります。必要なリードの変換元値を使用できない場合、リードを変換するユーザのデフォルトのレコードタイプに値を追加します。レコードタイプの詳細は、Salesforce オンラインヘルプを参照してください。

- 選択リスト値:** システムは、空の標準リード選択リスト項目を対応付けるときに、取引先、取引先責任者、商談のデフォルトの選択リストを割り当てます。組織でレコードタイプを使用している場合、空の値は新しいレコード所有者のデフォルトの選択リストの値で置き換えられます。
- 自動フィード登録:** リードを新規取引先、取引先責任者、および商談に変換すると、リード所有者はリードアカウントから登録解除されます。リード所有者、生成されたレコードの所有者、およびリードに登録されたユーザは、Chatter フィード設定で自動登録を有効化しない限り、生成されたレコードに自動登録されません。ニュースフィードで取引先、取引先責任者、および商談レコードへの変更を表示するには、自動登録を有効化する必要があります。作成するレコードを登録するには、ユーザは個人設定の 作成したレコードを自動的にフォローする オプションを有効にする必要があります。レコードへの変更がユーザのホームページのニュースフィードに表示されるように、レコードを登録できます。Salesforce でレコードに行われた変更の最新の状況を得る便利な方法です。

## DML 例外とエラー処理

### 例外処理

DML ステートメントは、DML 操作の実行中にデータベースで問題が発生すると実行時例外を返します。try-catch ブロック内に DML ステートメントを含めることでコードで例外を処理できます。次の例では、 DML ステートメントが try-catch ブロック内に含まれています。



### Database クラスメソッドの結果オブジェクト

Database クラスメソッドは、データ操作の結果を返します。これらの結果オブジェクトには、各レコードのデータ操作に関する有益な情報(操作が成功したのかどうかやエラー情報など)が含まれています。操作のタイプごとに特定の結果オブジェクト種別が返されます。以下にその概要を示します。

操作	Result クラス
insert、 update	Database.SaveResult クラス
upsert	Database.UpsertResult クラス
delete	Database.DeleteResult クラス
undelete	Database.UndeleteResult クラス

操作	Result クラス
convertLead	<a href="#">Database.LeadConvertResult クラス</a>
emptyRecycleBin	<a href="#">Database.EmptyRecycleBinResult クラス</a>

## 返されるデータベースエラー

DML ステートメントでは、処理されているいずれかのレコードの操作に失敗すると、必ず例外が返されてすべてのレコードの操作がロールバックされますが、Database クラスメソッドの場合、同様の動作を行うことも、レコード処理の一部の成功を許可することもできます。後者(

## DML の詳細

### DML オプションの設定

オブジェクトで目的のオプションを設定することにより、挿入操作や更新操作の DML オプションを指定できます。操作の を設定するには、`sObject` で メソッドをコールするか、これをパラメータとして および メソッドに渡します。

DML オプションを使用して、次のことを指定できます。

- ・ 項目の切り捨て動作。
- ・ 割り当てルール情報。
- ・ メールの自動送信を許可するかどうか。
- ・ 表示ラベルのユーザロケール。
- ・ 部分的な完了を操作で許可するかどうか。

### Database.DMLOptions プロパティ

クラスには次のプロパティがあります。

- ・ `allowFieldTruncation` プロパティ
- ・ `assignmentRuleHeader` プロパティ
- ・ `emailHeader` プロパティ
- ・ `localeOptions` プロパティ
- ・ `optAllOrNone` プロパティ

`DMLOptions` は、API バージョン 15.0 以降で保存された Apex にのみ使用できます。

#### `allowFieldTruncation` プロパティ

プロパティでは、文字列の切り捨て動作を指定します。バージョン 15.0 より前の API に対して保存された Apex では、文字列に値を指定し、その値が大きすぎる場合、値は切り捨てられます。API バージョン 15.0 以降では、大きすぎる値が指定されると、操作は失敗し、エラーメッセージが返されます。

プロパティを使用すると、API バージョン 15.0 以降に対して保存された Apex の新しい動作ではなく、以前の動作である切り捨てを使用するように指定できます。

プロパティは Boolean 値を使用します。 の場合、長すぎる文字列値を切り捨てます。これは API バージョン 14.0 以前の動作です。次に例を示します。

### assignmentRuleHeader プロパティ

プロパティは、ケース、またはリード作成時に使用する割り当てルールを指定します。



メモ: Database.DMLOptions オブジェクトは、ケースおよびリードの割り当てルールをサポートしますが、取引先またはテリトリー管理の割り当てルールはサポートしません。

プロパティを使用すると、次のオプションを設定できます。

- : ケースまたはリードの割り当てルールの ID。割り当てルールは有効または無効にできます。ID は、AssignmentRule sObject をクエリして取得することができます。assignmentRuleId が指定されている場合は、  
を指定しないでください。
- : ケースまたはリードにデフォルトの (有効な) 割り当てルールを使用するかどうかを示します。useDefaultRule が指定されている場合は、  
を指定しないでください。

次の例では、オプションを使用します。

次の例では、

オプションを使用します。

### emailHeader プロパティ

Salesforce ユーザインターフェースを使用して、次のようなイベントが発生した場合にメールを送信するかしないかを指定できます。

- ケースまたは ToDo の新規作成
- ケースコメントの作成
- ケースメールの取引先責任者への変換
- 新規ユーザのメール通知

- リードキューのメール通知
- パスワードのリセット

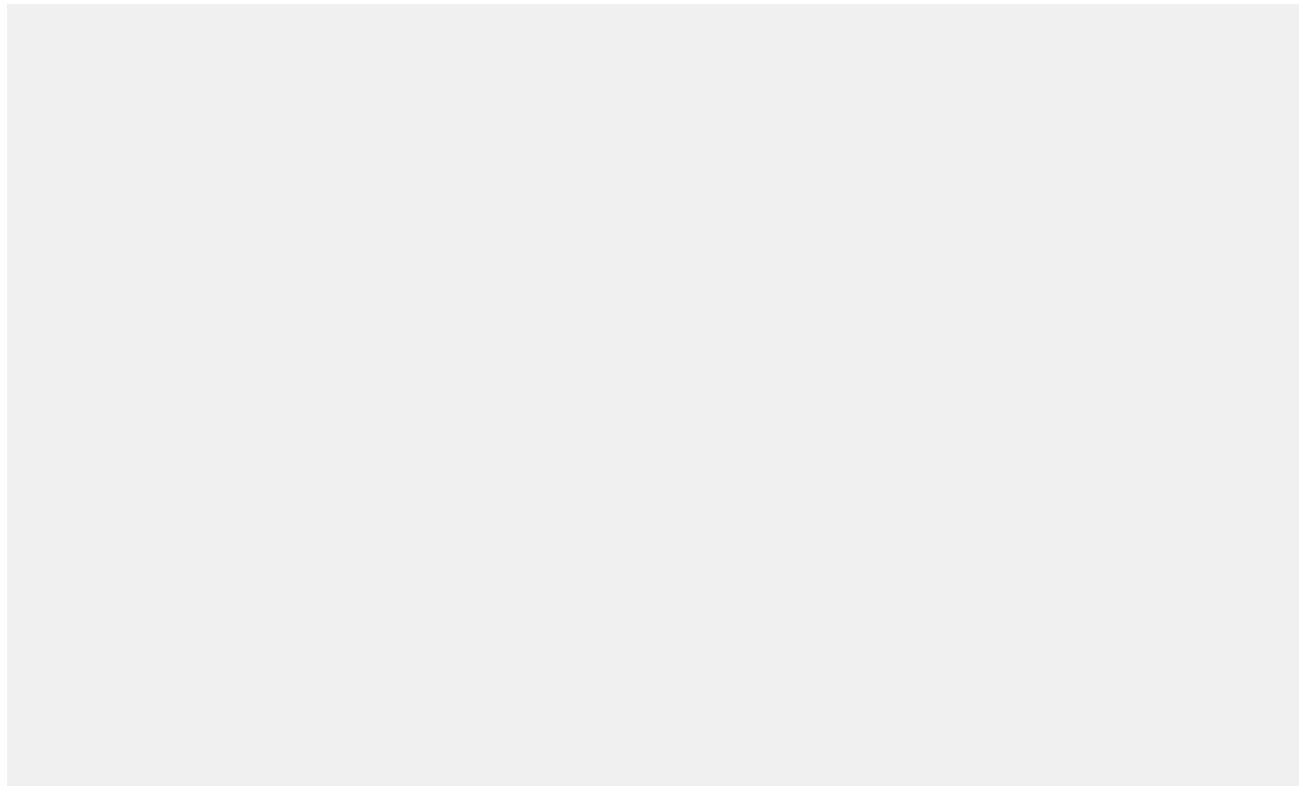
API バージョン 15.0 以降に対して保存された Apex で、`Database.DMLOptions` プロパティを使用すると、コードの実行によりイベントのいずれかが発生したときに送信されるメールに関する追加情報を指定できます。

プロパティを使用すると、次のオプションを設定できます。

- : リード、ケースに対して自動応答ルールをトリガするか( )、トリガしないか( )を示します。Salesforce ユーザインターフェースで、このメールは、ケースの作成やユーザパスワードのリセットなど、さまざまなイベントによって自動的にトリガされます。この値が に設定されている場合、ケースが作成されると、 に指定された取引先責任者のメールアドレスがあれば、メールは で指定されたアドレスに送信されます。アドレスがない場合、メールは で指定されたアドレスに送信されます。
- : 組織外のメールをトリガするか( )、トリガしないか( )を示します。Salesforce ユーザインターフェースで、このメールは、ケースの取引先責任者の作成、編集、削除によって自動的にトリガされます。
- : 組織内のユーザに送信されるメールをトリガするか( )、トリガしないか( )を示します。Salesforce ユーザインターフェースで、このメールはパスワードのリセット、ユーザの新規作成、コメントのケースへの追加、ToDo の作成または変更など、さまざまなイベントによって自動的にトリガされます。

次の例では、

オプションが指定されます。



グループイベントによって Apex で送信されるメールには、追加の動作が含まれます。グループイベントとは、`isTrue` であるイベントです。EventAttendee オブジェクトは、グループイベントに招待されているユーザ、リード、または取引先責任者を追跡します。Apex を使用して送信されるグループイベントメールでは、次のような動作に注意してください。

- ユーザに対するグループイベントの招待状の送信は、オプションの影響を受けます。
- リードまたは取引先責任者に対するグループイベントの招待状の送信は、オプションの影響を受けます。
- グループイベントの更新または削除時に送信されるメールも、送信対象に基づき や オプションの影響を受けます。

#### `localeOptions` プロパティ

プロパティでは、Apex で返される表示ラベルの言語を指定します。値は、`de_DE` や `en_GB` など、有効なユーザロケール(言語および国)である必要があります。値は文字列で、文字数は 2 から 5 文字です。最初の 2 文字は常に、「`fr`」や「`en`」などの ISO 言語コードです。値がさらに国別に評価される場合、文字列はアンダースコア( \_) に続き、「`US`」や「`UK`」などの ISO 国コードが続きます。たとえば、アメリカを示す文字列は「`en_US`」、カナダのフランス語圏を示す文字列は「`fr_CA`」です。

Salesforce がサポートする言語の一覧は、Salesforce オンラインヘルプの「Salesforce がサポートする言語は?」を参照してください。

#### `optAllOrNone` プロパティ

プロパティでは、部分的な完了を操作で許可するかどうかを指定します。 が  
に設定されている場合、レコードでエラーが発生すると、すべての変更はロールバックされます。このプロパ  
ティのデフォルトが である場合、レコードにエラーがない限り、正常に処理されたレコードがコミットさ  
れます。このプロパティは、Salesforce.com API バージョン 20.0 以降で保存された Apex で使用できます。

#### 関連リンク

[sObject メソッド](#)  
[Database クラス](#)

## トランザクションの制御

すべての要求は、Apex コードを実行するトリガ、クラスメソッド、Web サービス、Visualforce ページ、または匿名ブロックによって区切られます。要求全体が正常に完了した場合、すべての変更はデータベースに確定され  
ます。たとえば、Visualforce ページが Apex コントローラをコールしたことにより、さらに Apex クラスがコール  
されたとします。すべての Apex コードの実行が完了し、Visualforce ページの実行が完了したときに、変更がデータ  
ベースに確定されます。要求が正常に完了しなかった場合は、データベースへのすべての変更はロールバック  
されます。

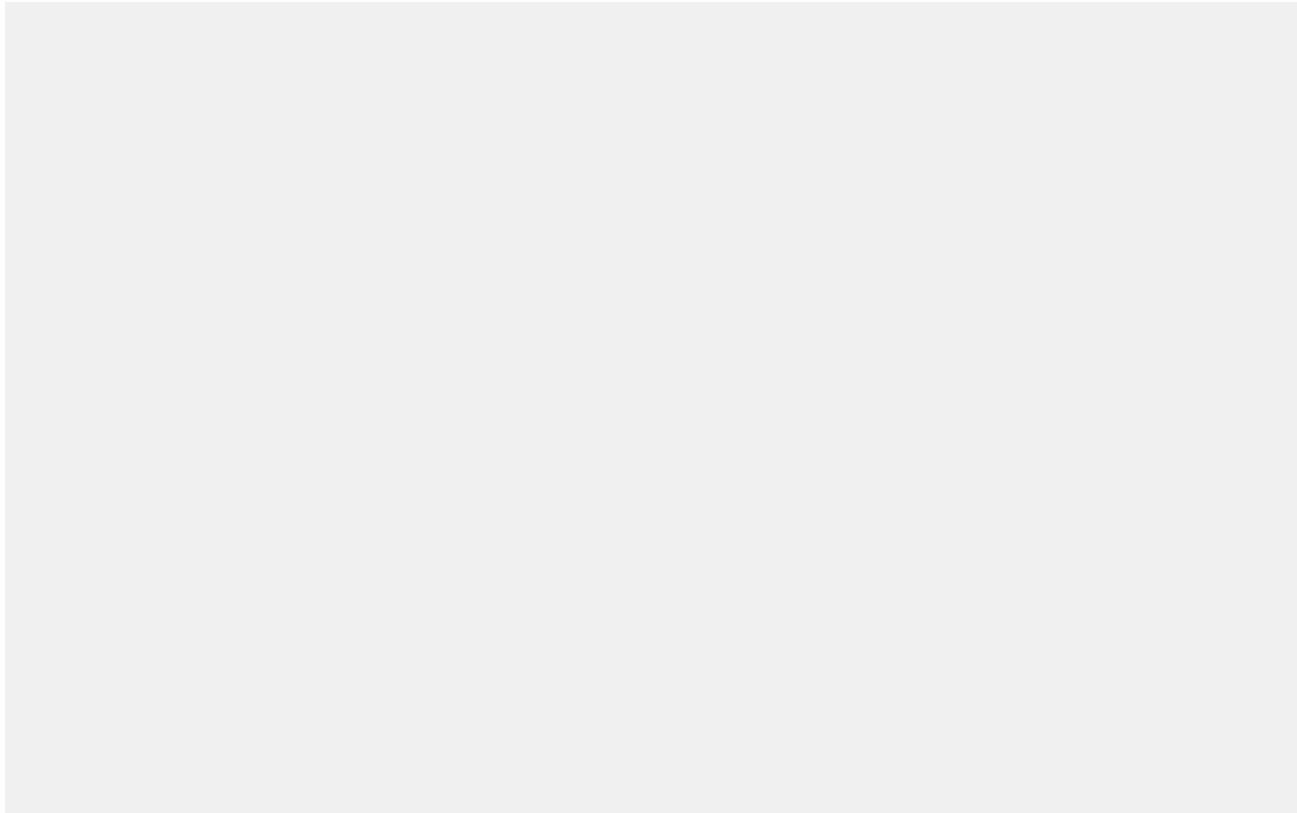
場合によっては、ビジネスルールによってレコードの処理中に作業の一部(すでに実行された DML ステートメント)を「ロールバック」してその処理を別の指示のもとで続行できるようにする必要があります。Apex では、`savepoint` を生成できます。これは要求中のある時点を示し、その時点でのデータベースの状態を指定します。

savepoint の後にある DML ステートメントを破棄して、savepoint の生成時点と同じ状況にデータベースを復元できます。

次の制限事項は、savepoint 変数の生成とデータベースのロールバックに適用されます。

- 複数の savepoint を設定し、生成した最新 savepoint ではない savepoint にロールバックすると、ロールバックされた savepoint 変数は無効になります。たとえば、最初に savepoint 1 を生成し、次に savepoint 2 を生成した場合、2 にロールバックすると、変数 1 は無効になります。その変数を使用しようとすると、ランタイムエラーが発生します。
- 各トリガ呼び出しが新しい実行コンテキストであるため、savepoints への参照は、トリガ呼び出しを通過することはできません。静的変数として savepoint を宣言し、トリガコンテキスト全体で使用しようとする、ランタイムエラーが発生します。
- 設定した各セーブポイントは、DML ステートメントのガバナ制限にカウントされます。
- ロールバック中、静的変数は戻されません。トリガの実行を再試行する場合、静的変数には最初の実行から得た値が維持されます。
- 各ロールバックは、DML ステートメントのガバナ制限にカウントされます。データベースをそれ以上の回数ロールバックしようとすると、ランタイムエラーが発生します。
- savepoint の設定後に挿入された sObject の ID は、ロールバック後にクリアされません。ロールバック後に挿入するには、新しい sObject を作成します。ロールバック前に作成した変数を使用して sObject を挿入しようとすると、その sObject 変数には ID があるため失敗します。同じ変数を使用して sObject を更新または更新/挿入しようとした場合も、sObject はデータベース内に存在せず、更新できないため失敗します。

と データベースメソッドの使用例を次に示します。



## DML 操作で同時に使用できない sObject

特定の sObject に対する DML 操作は、同じトランザクション内の他の sObject と混在させることができません。sObject には、組織のレコードへのユーザのアクセスに影響を与えるものがあるためです。こうした種別の sObject は、不正なアクセスレベル権限で操作が実行されないように、別のトランザクションで挿入または更新する必要があります。たとえば、1つのトランザクション内で取引先とユーザロールを更新することはできません。ただし、delete DML 操作の制限はありません。

DML 操作を実行するとき、同じトランザクション内で次の sObject を他の sObject と一緒に使用することはできません。

- FieldPermissions
- Group

他の sObject を含む 1 つのトランザクションでは、グループの挿入と更新のみを行うことができます。その他の DML 操作は使用できません。

- GroupMember

Salesforce.com API バージョン 14.0 以前を使用して保存された Apex コードの場合、他の sObject を含む 1 つのトランザクションでは、グループメンバーの挿入と更新のみを行うことができます。

- ObjectPermissions
- PermissionSet
- PermissionSetAssignment
- QueueSObject
- SetupEntityAccess
- User

Salesforce.com API バージョン 14.0 以前を使用して保存された Apex コードの場合、他の sObject を含む 1 つのトランザクションでは、ユーザの挿入を行うことができます。

Salesforce.com API バージョン 15.0 以降を使用して保存された Apex コードの場合、`OwnerId` が null に指定されていれば、他の sObject を含む 1 つのトランザクションでは、ユーザの挿入を行うことができます。

Salesforce.com API バージョン 14.0 以前を使用して保存された Apex コードの場合、他の sObject を含む 1 つのトランザクションでは、ユーザの更新を行うことができます。

Salesforce.com API バージョン 15.0 以降を使用して保存された Apex コードの場合、次の項目も更新されなければ、他の sObject を含む 1 つのトランザクションで、ユーザの更新を行うことができます。

◊  
◊  
◊  
◊  
◊  
◊

- UserRole
- UserTerritory
- Territory
- Salesforce.com API バージョン 17.0 以前を使用して保存された Apex コードのカスタム設定。

カスタムコントローラで Visualforce ページを使用している場合、1 つの要求またはアクションでは、DML 操作は1つのみのデータ型の Object に対して行うことができます。ただし、後続要求では、異なるデータ型の sObject で DML 操作を実行できます。たとえば、保存ボタンで取引先を作成した後に、送信ボタンでユーザを作成できます。

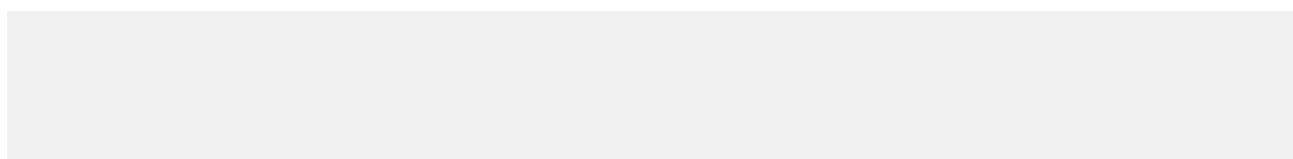
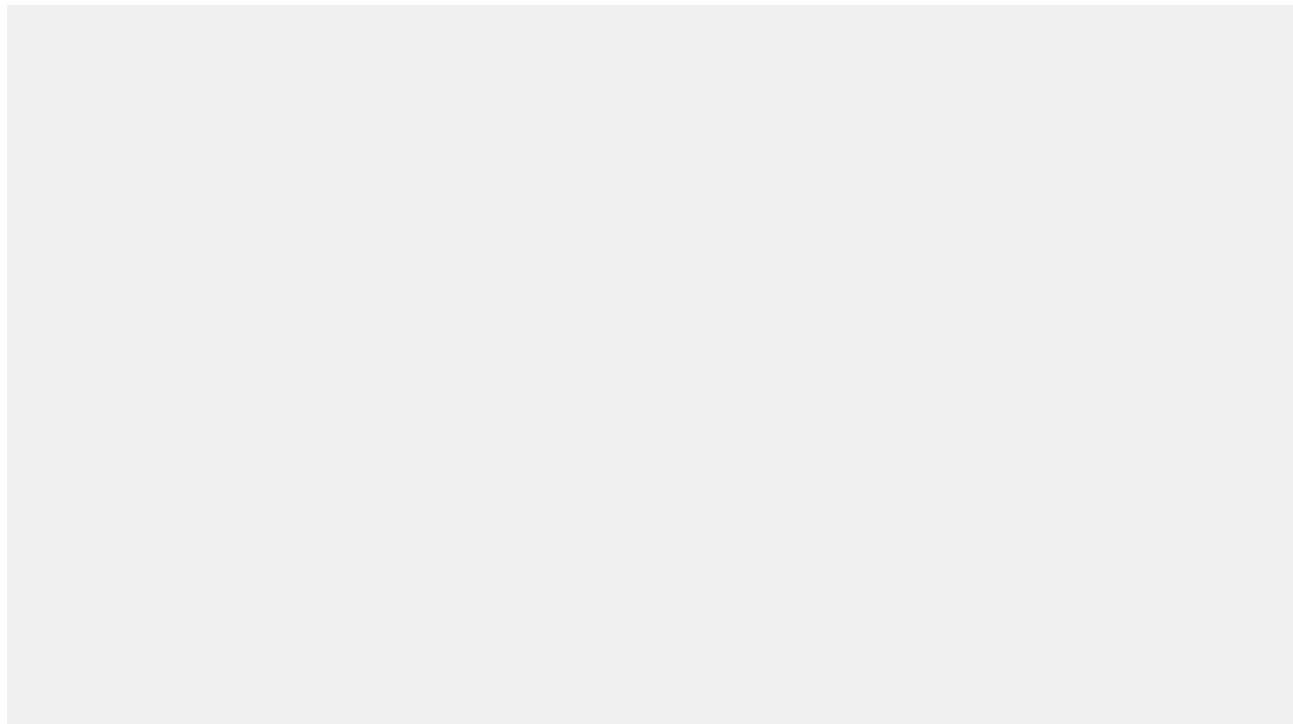
次のプロセスを使用して、1 つのクラスで複数のデータ型の sObject に対して DML 操作を実行できます。

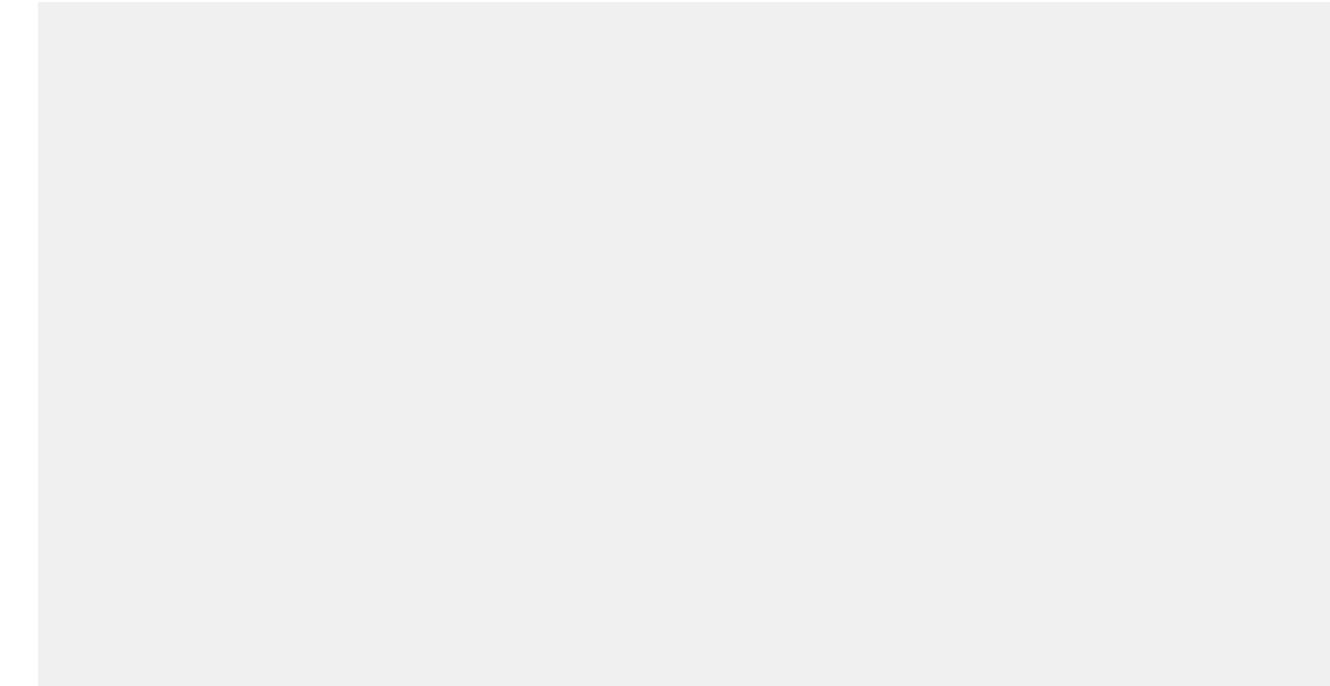
1. 1 つのデータ型の sObject で DML 操作を行うメソッドを作成します。
2. 2 番目の sObject データ型を操作するために アノテーションを使用する 2 番目のメソッドを作成します。

このプロセスは、次のセクションの例で説明します。

#### 例: future メソッドを使用した混合 DML 操作の実行

この例では、future メソッドを使用して User オブジェクトに対する DML 操作を実行することで、混合 DML 操作を実行する方法を示します。





## テストメソッドでの混合 DML 操作

テストメソッドでは、DML 操作を実行するコードが **「DML 操作で同時に使用できない sObject」** に記載された sObject とその他の sObject の間の混合 DML 操作の実行が許可されます。たとえば、これによりロールとその他の sObject を持つユーザを同じテスト内で作成できます。

### 例: `System.runAs` ブロックでの混合 DML 操作

この例では、混合 DML 操作を **ブロック** で囲み、混合 DML エラーを回避する方法を示します。ブロックは、現在のユーザのコンテキストで実行されます。このブロックは、ロールを持つテストユーザとテスト取引先を作成するという混合 DML 操作を実行します。

```
System.runAs (thisUser) {
```

```
}
```

#### テストメソッドで `Test.startTest` と `Test.stopTest` を使用して混合 DML エラーを回避する

混合 DML 操作を実行するコードブロックを ブロックで囲んでも、混合 DML 例外エラーが返されることがあります。これは、テストメソッドが、グループの削除など、他の操作とは混合できない DML 操作を実行する `future` メソッドをコールした場合に発生する可能性があります。この場合、混合 DML 例外が返されたら、`future` メソッドコールを行うコードブロックを、 および ステートメントで囲みます。

この例では、 および ステートメントで ステートメントを囲むことにより、テストで混合 DML 例外エラーを回避する方法を示します。 ステートメントにより、混合 DML 操作は `future` メソッドで実行されることになるため、ステートメントを および ステートメントで囲みます。 ステートメントはトリガを起動し、 クラスの `future` メソッドをコールして取引先挿入トリガですでに挿入されていたグループを削除します。

これは、混合 DML 操作を起動するテストクラスです。取引先挿入および削除がトリガを起動します。

```
System.RunAs(u1) {
```

```
Test.startTest()  
  
Test.stopTest()  
}
```

これは、グループを挿入する取引先挿入トリガです。

これは、future メソッドをコールしてグループを削除する取引先削除トリガです。

これは、グループを削除する future メソッドです。

## DML 操作をサポートしない sObject

組織には、Salesforce が提供する標準オブジェクトと、独自に作成したカスタムオブジェクトが含まれます。これらのオブジェクトは、Apex で sObject データ型のインスタンスとしてアクセスできます。これらのオブジェクト

に対し、クエリや DML 操作を実行できます。ただし、一部の標準オブジェクトはクエリで取得できますが、DML 操作をサポートしていません。これには次のようなオブジェクトがあります。

- AccountTerritoryAssignmentRule
- AccountTerritoryAssignmentRuleItem
- ApexComponent
- ApexPage
- BusinessHours
- BusinessProcess
- CategoryNode
- CurrencyType
- DatedConversionRate
- ProcessInstance
- Profile
- RecordType
- SelfServiceUser
- StaticResource
- UserAccountTeamMember
- UserTerritory
- WebLink



メモ: SOAP API を使用しても、すべての標準オブジェクトとカスタムオブジェクトにアクセスできます。例外は ProcessInstance です。SOAP API で ProcessInstance の作成、更新、削除はできません。

## 一括 DML 例外処理

一括 DML コールによって発生する例外（コールの直接的な結果によって実行されるトリガ内の再帰的 DML 操作を含む）は、コールの発生元ごとに異なる処理がされます。

- Apex DML ステートメントから直接発生した一括 DML コールが原因でエラーが発生した場合、または、データベース DML メソッドの `BatchSize` パラメータが `True` として指定されている場合、ランタイムエンジンは「オールオアナッシング」ルールに従います。つまり、1回の操作の間、すべてのレコードを正常に更新するか、または操作全体を DML ステートメントのすぐ前の時点にロールバックする必要があります。
- SOAP API から発生した一括 DML コールが原因でエラーが発生した場合、ランタイムエンジンは少なくとも部分的な保存を試みます。
  1. 最初の試行で、ランタイムエンジンはすべてのレコードを処理します。入力規則や独自のインデックス違反などの問題によるエラーを生成したレコードは、除外されます。
  2. 最初の試行にエラーがあった場合、ランタイムエンジンは、エラーを生成しなかったレコードのみを含む2回目の試行を行います。最初の試行でエラーを生成しなかったすべてのレコードが処理され、競合の条件などが理由でエラーを生成したレコードがあれば、それも除外されます。
  3. 2回目の試行中に追加工エラーがあった場合、ランタイムエンジンは、初回と2回目にエラーを生成しなかったレコードのみを含む3回目と最後の試行を行います。エラーを生成したレコードがある場合、操作全体

は失敗し、エラーメッセージ「Too many batch retries in the presence of Apex triggers and partial failures (Apex トリガと部分的な失敗がある場合にバッチ試行の回数が多すぎます)」が表示されます。



メモ: 2回目と3回目の試行中、ガバナ制限は、最初の試行前の元の状態にリセットされます。「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

## Apex のデータについて知っておくべきこと

### Null 以外の必須項目値と Null 項目

新規レコードの挿入または既存のレコードの必須項目の更新を行う場合、すべての必須項目に \_\_\_\_\_ 以外の値を指定する必要があります。

SOAP API とは異なり、Apex では、sObject レコードの \_\_\_\_\_ 配列を更新せずに、項目値をに変更できます。多くの SOAP プロバイダで \_\_\_\_\_ 値の処理が統一されていないため、API ではこの配列に更新する必要があります。Apex は Force.com プラットフォーム上のみで実行されるため、この回避策は不要です。

### DML は一部の sObject でサポートされていない

DML 操作は、特定の sObject ではサポートされていません。 「[DML 操作をサポートしない sObject](#)」を参照してください。

### 文字列項目の切り捨てと API バージョン

項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存(コンパイル)した Apex クラスとトリガにはランタイムエラーが発生します。

### DML 操作を有効にする sObject プロパティ

sObject レコードを挿入、更新、削除、復元できるようにするには、sObject の対応するプロパティ \_\_\_\_\_ ( \_\_\_\_\_ 、 \_\_\_\_\_ 、 \_\_\_\_\_ 、 \_\_\_\_\_ ) を \_\_\_\_\_ に設定する必要があります。

### ID 値

ステートメントは、すべての新規 sObject レコードの ID 値を自動的に設定します。すでに ID がある(つまり、すでに組織のデータに存在している)レコードを挿入すると、エラーが発生します。詳細は、「[List](#)」を参照してください。

および \_\_\_\_\_ ステートメントは、レコードの各バッチに重複 ID 値がないかどうか確認します。重複がある場合、最初の 5 つが処理されます。6 番目とその他すべての重複 ID については、これらのエントリの SaveResult が、次のようなエラーでマークされます。

*number\_of\_attempts*

更新された sObject レコードの ID は \_\_\_\_\_ ステートメントで変更できませんが、関連レコード ID は変更できます。

## 一意制約のある項目

一意制約のある項目を含む一部の sObject では、重複する sObject レコードを挿入するとエラーになります。たとえば、同じ名前の複数の CollaborationGroup sObject を挿入すると、CollaborationGroup レコードには一意の名前が必要なためエラーになります。

## 自動的に設定されるシステム項目

新規レコードを挿入すると、  
、  
、  
などのシステム項目が自動的に更新されます。これらの値を Apex で明示的に指定することはできません。同様に、レコードを更新すると、  
、  
、  
などのシステム項目が自動的に更新されます。

## DML ステートメントで処理される最大レコード数

1つの<sup>1</sup> 揿入、更新、削除、DELETE メソッドには、最大 10,000 個の sObject レコードを渡すことができます。

各 DML ステートメントは、レコードの挿入とレコードの更新という 2 つの操作で構成されます。これらの各操作は、INSERT と UPDATE のランタイム制限でそれぞれ制限されます。たとえば、10,000 を超えるレコードを更新/挿入し、すべてが更新中の場合、エラーが発生します（「[実行ガバナと制限について](#)」（ページ 337）を参照してください）。

## 更新/挿入と外部キー

sObject レコードが参照項目として設定されている場合、sObject レコードを更新/挿入するために外部キーを使用できます。詳細は、『*Object Reference for Salesforce and Force.com*』の「[データ型](#)」を参照してください。

# レコードのロック

## ロックステートメント

Apex では、レコードの更新中に sObject レコードをロックして、競合の条件やスレッドの安全性の問題の発生を回避できます。sObject レコードがロックされると、他のすべてのクライアントとユーザは、コードまたは Salesforce ユーザインターフェースを使用して更新を行えません。レコードをロックしているクライアントは、レコードに対してロジックを実行し、更新を行うことができます。ロック中は、ロックされたレコードが別のクライアントによって変更されることはありません。トランザクションが完了するとロックが解除されます。

Apex の一連の sObject レコードをロックするには、インライン SOQL ステートメントの後に LOCK キーワードを埋め込みます。たとえば、次のステートメントでは 2 つの取引先をクエリすると共に、返された取引先をロックします。



メモ: ロックを使用する SOQL クエリでは、

キーワードを使用できません。

### ロックに関する考慮事項

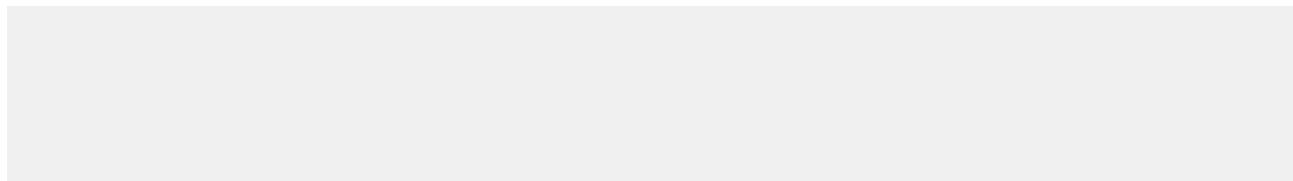
- クライアントがレコードをロックしている間、そのクライアントは同一トランザクションでデータベースの項目値を変更できます。他のクライアントが同じレコードを更新するには、トランザクションが完了してレコードのロックが解除されるまで待機する必要があります。ロックされている間も、他のクライアントは同じレコードをクエリできます。
- 別のクライアントが現在ロックしているレコードをロックしようとすると、  
が発生します。  
同様に、別のクライアントが現在ロックしているレコードを更新しようとすると、  
が発生します。
- ロックされているレコードをクライアントが変更しようとした場合、update コールが行われてから短時間でロックが解除されれば、更新操作は成功する可能性があります。この場合、2番目のクライアントがレコードの古いコピーを取得していると、ロックしていたクライアントが行った変更がこの更新によって上書きされる可能性があります。これを回避するには、2番目のクライアントが最初にレコードをロックする必要があります。ロックプロセスは、  
ステートメントを使用してデータベースのレコードの最新のコピーを返します。2番目のクライアントはこのコピーを使用して新しい更新を行うことができます。



警告: Apex コードにロックを設定する場合は、慎重に行ってください。 「デッドロックの回避」 を参照してください。

### SOQL For ループのロック

キーワードも SOQL ループ内で使用できます。次に例を示します。



「SOQL For ループ」で説明するように、上記の例は、SOAP API の  
メソッドおよび  
ソッドのコールに内部的に対応します。

ステートメントはありません。Apex トリガが正常に完了すると、自動的にデータベースの変更がコミットされます。Apex トリガが正常に完了しない場合、データベースへの変更はロールバックされます。

### デッドロックの回避

複数のデータベーステーブルや行の更新を行う他の手続き型ロジック言語と同様に、Apex はデッドロックが発生する可能性があります。デッドロックを回避するため、Apex ランタイムエンジンでは、次の処理が行われます。

- sObject の親レコードをロックしてから子レコードをロックします。
- 同じ型の複数のレコードを編集している場合は、ID 順に sObject レコードをロックします。

開発者はデッドロックが引き起こされないように行をロックする場合、慎重に行ってください。アプリケーション内のあらゆる場所から同じ順序でテーブルと行にアクセスして、標準のデッドロック回避手法が使用されていることを確認してください。

## 第3章

### Apex の呼び出し

---

トピック:

- [トリガ](#)
- [Apex スケジューラ](#)
- [匿名ブロック](#)
- [Apex in AJAX](#)

Apex コードを呼び出すには、複数あるメカニズムのいずれかを使用します。Apex トリガを記述し、指定された sObject 型に対する特定の操作の前または後に、トリガで指定されたイベントについてトリガコードを呼び出すことができます。また、Apex クラスを記述して指定された間隔で実行されるようにスケジュールしたり、匿名ブロック内でコードスニペットを実行したりすることもできます。さらに、AJAX Toolkit を使用して、Apex で実装されている Web サービスマソッドを呼び出すことができます。

この章では、次の内容を取り上げます。

- [トリガ](#)
- [Apex スケジューラ \(Apex クラスのみ\)](#)
- [匿名ブロック](#)
- [AJAX Toolkit](#)

## トリガ

Apex は、トリガを使用して呼び出すことができます。トリガは、次の操作の前または後に実行する Apex コードです。

- insert
- update
- delete
- merge
- upsert
- undelete

たとえば、オブジェクトのレコードがデータベースに挿入される前、レコードが削除された後、またはレコードがごみ箱から復元された後に実行されるトリガがあります。

Contact または Account、CaseComment などの一部の標準的な子オブジェクト、およびカスタムオブジェクトなど、トリガをサポートする最上位の標準オブジェクトのトリガを定義できます。

- ケースコメントの場合は、[設定] から [カスタマイズ] > [ケース] > [ケースコメント] > [トリガ] をクリックします。
- メールメッセージの場合は、[設定] から [カスタマイズ] > [ケース] > [メールメッセージ] > [トリガ] をクリックします。

トリガは、次の 2 つの種類に分けられます。

- before トリガは、レコードがデータベースに保存される前にレコード値を更新または検証するために使用できます。
- after トリガは、データベースで設定された項目値 (レコードの 項目または 項目など) にアクセスし、監査テーブルへのログインやキューを使用する非同期イベントの実行など、他のレコードの変更に影響を与える場合に使用できます。

トリガは、最初にトリガを実行したレコードと同じ種類の別のレコードを変更することもできます。たとえば、取引先責任者 A が更新された後でトリガを実行する場合、このトリガは取引先責任者 B、C、および D を変更することもできます。トリガを使用して他のレコードを変更でき、これらの変更によってさらに複数のトリガを実行できるために、Apex ランタイムエンジンはこうしたすべての操作を単一の作業単位とみなし、実行可能な操作数の制限を設定して無限に操作が反復されないようにします。「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

さらに、before トリガでレコードを更新または削除したり、after トリガでレコードを削除したりすると、ランタイムエラーが発生します。これは、直接操作または間接操作のいずれの場合にも該当します。たとえば、取引先 A を更新し、取引先 A の更新 before トリガが取引先責任者 B を挿入し、取引先責任者 B の挿入 after トリガが取引先 A をクエリして DML ステートメントまたはデータベースメソッドを使用してその取引先を更新する場合、before トリガで取引先 A を間接的に更新することになるため、ランタイムエラーが発生します。

### 実装に関する考慮事項

トリガを作成する前に、次の点に留意してください。

- トリガは、必要に応じて before および after のトリガを実行します。
- トリガは、削除されるレコードには before および after のトリガのみを実行します。 「[トリガと Merge ステートメント](#)」(ページ 142)を参照してください。
- レコードが復元された後に実行するトリガは、特定のオブジェクトでのみ機能します。 「[トリガと復元レコード](#)」(ページ 143)を参照してください。
- トリガが終了するまで、項目履歴は記録されません。 トリガで項目履歴をクエリしても、現在のトランザクションの履歴は表示されません。
- Salesforce.com API バージョン 20.0 以前を使用して保存された Apex の場合、API コールによってトリガが起動されると、200 レコードずつに分割されていたチャンクが、100 レコードずつのチャンクにさらに分割されます。 Salesforce.com API バージョン 21.0 以降を使用して保存された Apex の場合、API のチャンクがそれ以上分割されることはありません。 静的変数の値は、API バッチ間でリセットされますが、ガバナ制限はリセットされません。 API バッチ間の状態情報の追跡に静的変数を使用しないでください。

## 一括トリガ

デフォルトでは、すべてのトリガが一括トリガで、複数のレコードを一度に処理できます。常に一度に複数のレコードの処理を予定します。



メモ: 定期的と定義された行動オブジェクトは、[定期的](#)、[定義された行動](#)、[定期的](#)のトリガで一括処理されません。

一括トリガは、単一のレコード更新と次のような一括処理を行えます。

- データのインポート
- Force.com BulkAPI コール
- レコード所有者の変更や削除などの一括操作
- 再帰的 Apex メソッドや DML ステートメントの一括処理を呼び出すトリガ

## トリガ構文

トリガを定義するには、次の構文を使用します。

```
triggerName    ObjectName   trigger_events
code_block
```

*trigger\_events* には、次のイベントを 1 つ以上含むカンマ区切りのリストを指定できます。

- 
- 
- 
- 
-



### メモ:

- キーワードをトリガで使用できるのは、メソッドで非同期として定義されている場合、つまりメソッドがキーワードで定義されている場合のみです。
- 定期的な行動または定期的な ToDo の、またはによって呼び出されるトリガは、Force.com API からトリガが大量に呼び出されるとき、ランタイムエラーになります。

たとえば、次のコードは Account オブジェクトでトリガを定義します。

イベントおよび

イベントの

トリガのコードブロックに、キーワードを指定することはできません。トリガには、内部クラスに適用できるキーワードのみを含めることができます。また、トリガにより行われたデータベースへの変更は、手動で確定する必要はありません。Apex トリガが正常に完了すると、自動的にデータベースの変更がコミットされます。Apex トリガが正常に完了しない場合、データベースへの変更はロールバックされます。

## トリガコンテキスト変数

すべてのトリガは、開発者がランタイムコンテキストにアクセスできるようにする暗黙的な変数を定義します。これらの変数は、クラスに含まれています。

変数	使用方法
	Apex コードの現在のコンテキストが Visualforce ページ、Web サービス、または API コールではなく、トリガである場合、true を返します。
	Salesforce ユーザインターフェース、Apex、または API から挿入操作によりトリガが実行された場合、true を返します。
	Salesforce ユーザインターフェース、Apex、または API から更新操作によりトリガが実行された場合、true を返します。
	Salesforce ユーザインターフェース、Apex、または API から削除操作によりトリガが実行された場合、true を返します。
	レコードが保存される前にこのトリガが実行された場合、true を返します。

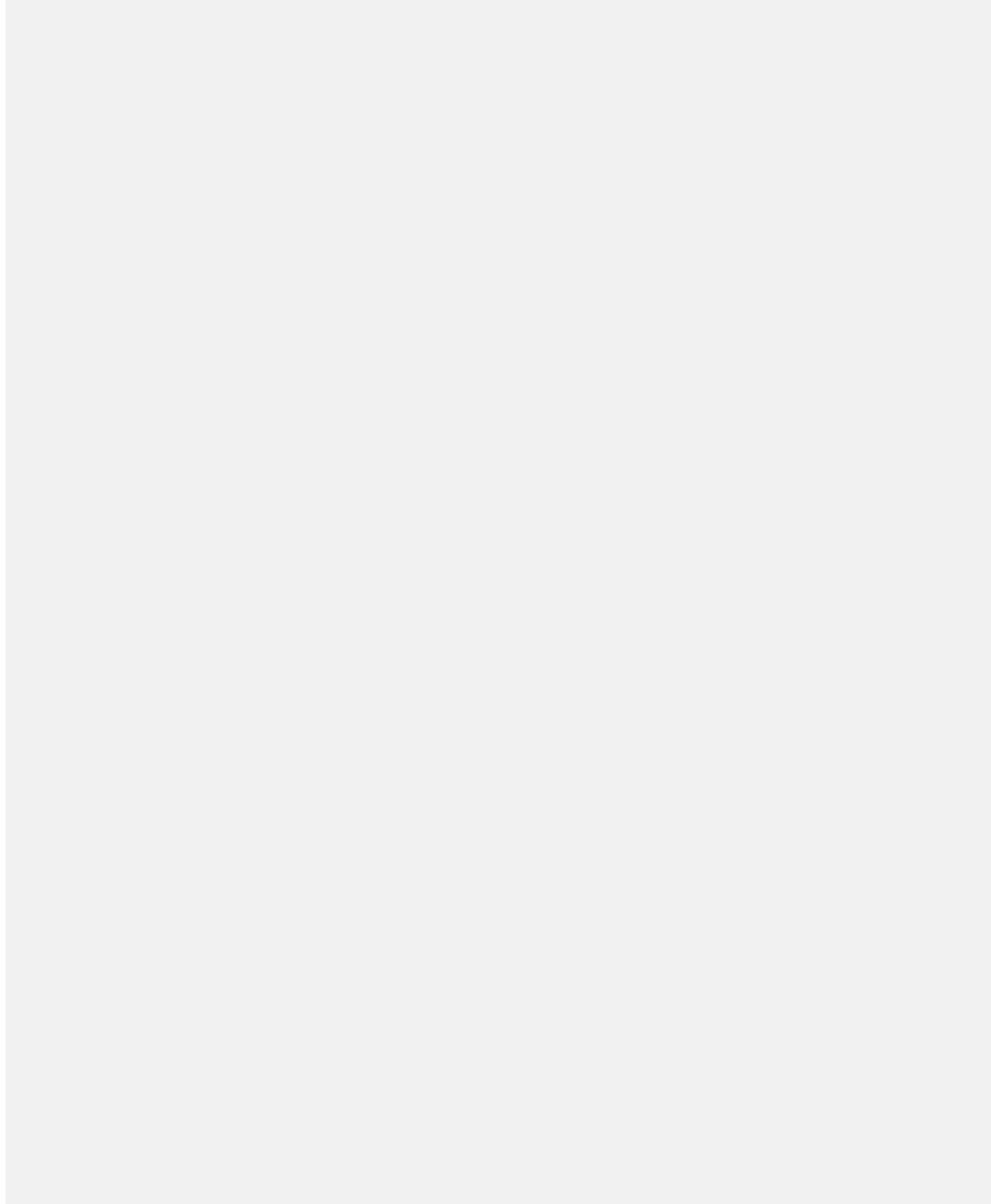
変数	使用方法
	新しいバージョンの sObject レコードのリストを返します。 この sObject リストは トリガと トリガでのみ使用でき、レコード は トリガでのみ更新できます。
	新しいバージョンの sObject レコードへの ID の対応付けです。 この対応付けは トリガ、 トリガ、 トリガでのみ使用できます。
	古いバージョンの sObject レコードのリストを返します。 この sObject リストは トリガと トリガでのみ使用できます。
	古いバージョンの sObject レコードへの ID の対応付けです。 この対応付けは トリガと トリガでのみ使用できます。
	古いバージョンと新しいバージョンの両方を含む、トリガ呼び出しのレコードの合 計数。

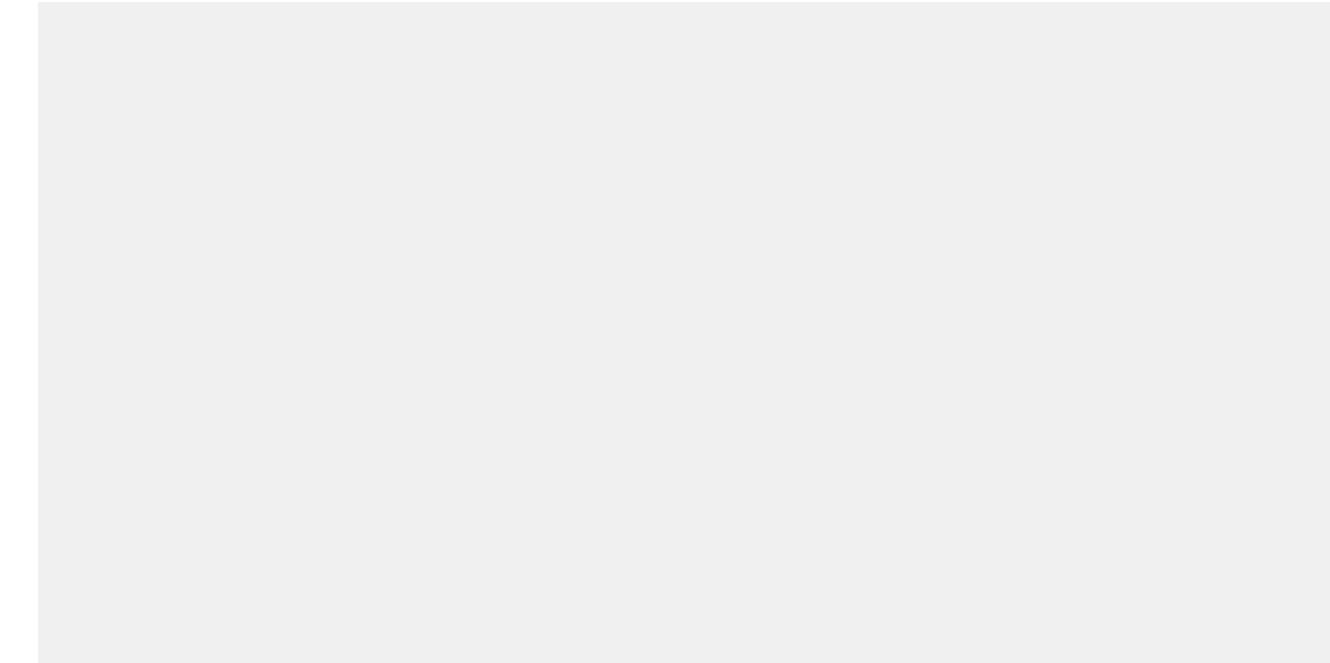


メモ: トリガを実行するレコードに無効な項目値がある場合(たとえば、0で割る式など)、値は 、  
、 および のトリガコンテキスト変数で に設定されます。

たとえば、この単純なトリガの場合、 は sObjects のリストで、 ループで繰り返し実行でき、  
SOQL クエリの 句でバインド変数として使用できます。

このトリガでは、や のような Boolean コンテキスト変数を使用して、特定のトリガ条件でのみ実行するコードを定義します。





## コンテキスト変数の考慮事項

トリガコンテキスト変数については、次の考慮事項について注意してください。

- および `trigger.old` を、Apex DML 操作で使用することはできません。
- `trigger.old` を使用してオブジェクトの項目値を変更できますが、before トリガでのみ行えます。すべての after トリガで、`trigger.old` は保存されず、実行時例外が発生します。
- `trigger.old` は常に読み取り専用です。
- `trigger.old` を削除することはできません。

次の表は、さまざまなトリガイベントの特定の操作についての考慮事項を示します。

トリガイベント	<code>trigger.new</code> を使用した項 目の変更	update DML 操作を使用し た元のオブジェクトの更新	delete DML 操作を使用し た元のオブジェクトの削除
	可。	該当なし。元のオブジェクトが作成されていません。 参照できるものがないため、更新できません。	該当なし。元のオブジェクトが作成されていません。 参照できるものがないため、更新できません。
	不可。 すでに保存されているため、 ランタイムエラーが発生します。	がす 可。	可能ですが、必須ではありません。挿入後すぐにオブジェクトが削除されます。
	可。	不可。ランタイムエラーが 発生します。	不可。ランタイムエラーが 発生します。

トリガイベント	<code>trigger.new</code> を使用した項 目の変更	update DML 操作を使用し た元のオブジェクトの更新	delete DML 操作を使用し た元のオブジェクトの削除
	不可。 すでに保存されているため、ランタイムエラーが発生します。	がす 可。不適切なコードにより無限に不適切な操作が反復される可能性があります。	可。オブジェクトが削除されると前に更新が保存されるため、オブジェクトが復元ガバナ制限によりエラーが発生する可能性があります。
	不可。ランタイムエラーが発生します。 は before 削除トリガで使用できません。	可。オブジェクトが削除される前に更新が保存されるため、オブジェクトが復元されません。	不可。ランタイムエラーが発生します。削除はすでに処理中です。
	不可。ランタイムエラーが発生します。 は after 削除トリガで使用できません。	該当なし。オブジェクトはすでに削除されています。	該当なし。オブジェクトはすでに削除されています。
	不可。ランタイムエラーが発生します。 は after 復元トリガで使用できません。	可。	可能ですが、必須ではありません。挿入後すぐにオブジェクトが削除されます。

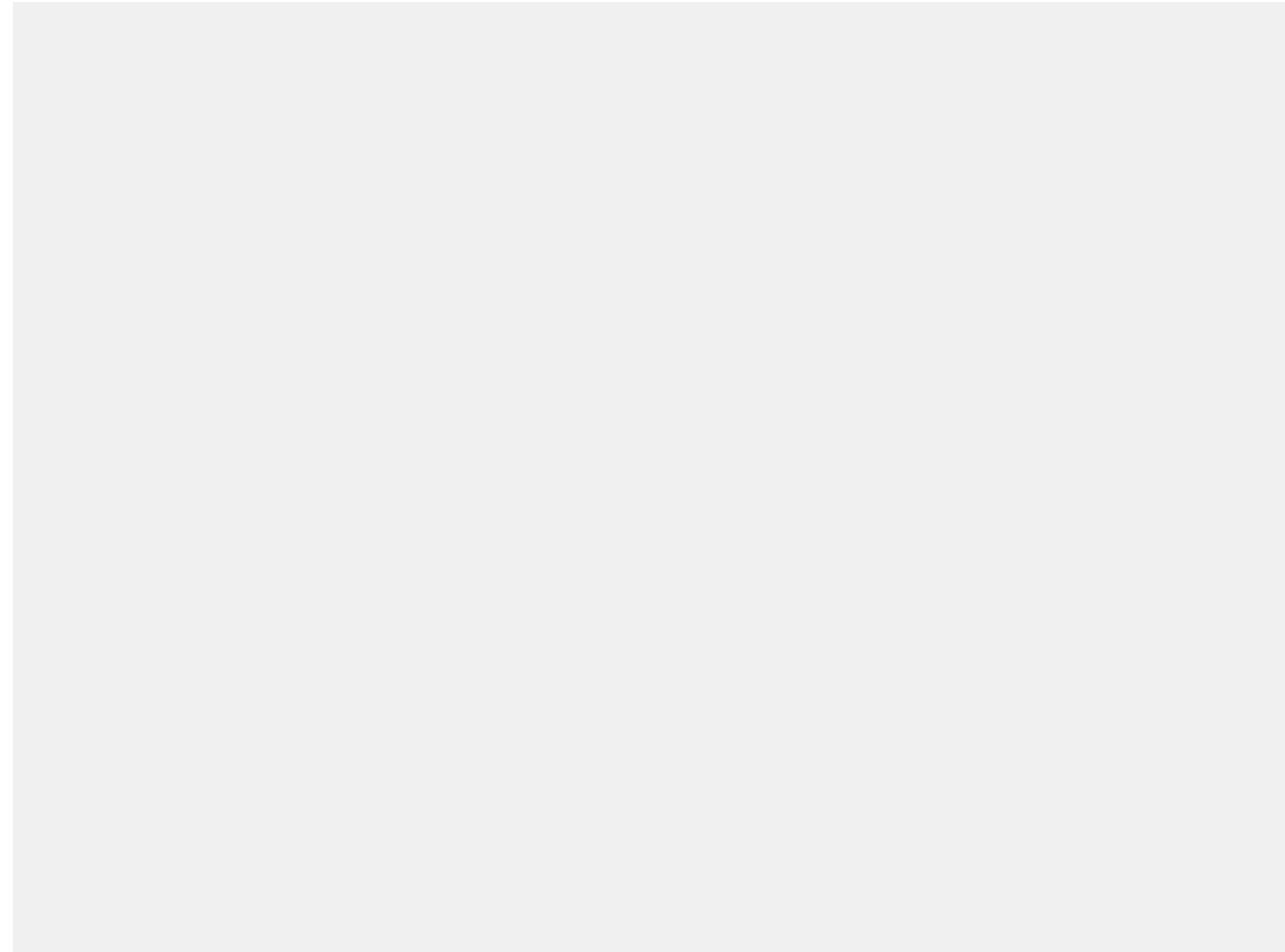
## 一般的な一括トリガイディオム

一括トリガを使用すると、開発者は実行ガバナ制限を超えることなくより多くのレコードを処理することができますが、一度に複数のレコードの一括処理を呼び出すため、理解しにくい場合やコード化が難しくなる場合があります。次のセクションでは、一括処理を記述する場合に頻繁に使用されるイディオムの例について説明します。

### 一括トリガでの対応付けおよびセットの使用

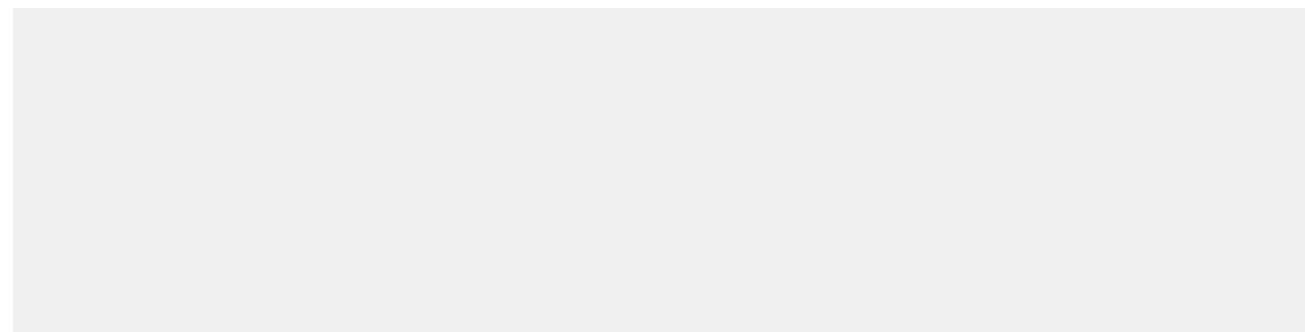
セットおよび対応付けのデータ構造は、一括トリガを適切にコード化する上で重要です。セットを使用して各レコードを分割し、対応付けを使用してクエリ結果をレコード ID で編成して保持することができます。

たとえば、サンプルの見積アプリケーションの一括トリガは、最初に OpportunityLineItem レコードに関連付けられた各価格表エントリをセットに追加し、セットに異なる要素のみが含まれるようにします。次に、関連付けられた製品の色の PricebookEntries をクエリして、結果を対応付けに配置します。対応付けが作成されると、トリガは OpportunityLineItems により繰り返し実行され、対応付けを使用して適切な色を割り当てます。



## 一括トリガのレコードとクエリ結果の関連付け

および の ID-to-sObject 対応付けを使用して、レコードをクエリ結果に関連付けます。たとえば、サンプル見積アプリケーションのこのトリガでは、 を使用して、一意の ID のセットを作成します( )。セットはクエリの一部として使用され、トリガで処理される商談に関連付けられる見積のリストを作成します。クエリによって返される各見積の場合、関連付けられた商談が から取得され、削除されないようにします。



## トリガを使用した、一意の項目を持つレコードの挿入または更新

イベントまたは イベントによってレコードがバッチ内の別の新しいレコードで一意の項目の値を複製する場合、重複したレコードについてのエラーメッセージには、最初のレコードの ID が記載されます。ただし、要求が完了するまではエラーメッセージが適切でない場合があります。

トリガが存在する場合、一括操作の再試行ロジックにより、ロールバック/再試行サイクルが発生します。その再試行サイクルは新しいキーを新しいレコードに割り当てます。たとえば、2つのレコードに一意の項目の同じ値が挿入され、 イベントがトリガに定義されている場合、2番目の重複レコードが失敗し、最初のレコードの ID が報告されます。ただし、変更がロールバックされ、最初のレコードが再挿入されると、レコードは新しい ID を受け取ります。つまり、2番目のレコードによって報告されたエラーメッセージは有効ではなくなります。

## トリガの定義

トリガコードは、トリガスクリプトが関連付けられたオブジェクトの下にメタデータとして保存されます。Salesforce でトリガを定義する手順は、次のとおりです。

- 標準オブジェクトの場合は、[設定] から [カスタマイズ] をクリックし、オブジェクトの名前をクリックしてから、[トリガ] をクリックします。

カスタムオブジェクトの場合は、[設定] から [作成]>[オブジェクト]をクリックし、オブジェクトの名前をクリックします。

キャンペーンメンバーの場合は、[設定] から [カスタマイズ]>[キャンペーン]>[キャンペーンメンバー]>[トリガ] をクリックします。

ケースコメントの場合は、[設定] から [カスタマイズ]>[ケース]>[ケースコメント]>[トリガ] をクリックします。

メールメッセージの場合は、[設定] から [カスタマイズ]>[ケース]>[メールメッセージ]>[トリガ] をクリックします。

アイデアに対するコメントの場合は、[設定] から [カスタマイズ]>[アイデア]>[アイデアのコメント]>[トリガ] をクリックします。

Attachment、ContentDocument、および Note 標準オブジェクトでは、Salesforce ユーザインターフェースでトリガを作成できません。これらのオブジェクトの場合、開発者コンソールや Force.com IDE などの開発ツールを使用してトリガを作成します。または、メタデータ API を使用することもできます。

- [トリガ] 関連リストで、[新規] をクリックします。
- [バージョン設定] をクリックして、このトリガで使用する Apex および API のバージョンを指定します。組織が AppExchange から管理パッケージをインストールした場合、このトリガで使用する各管理パッケージのバージョンも指定できます。すべてのバージョンでデフォルト値を使用します。デフォルト値では、Apex および API についても、各管理パッケージについても、トリガを最新バージョンに関連付けます。最新バージョンのパッケージのものとは異なるコンポーネントや機能にアクセスする場合は、管理パッケージの古いバージョンを指定することもできます。

4. トリガをコンパイルして有効にする必要があれば、[Apex トリガ] をクリックして 有効 チェックボックスをオンにします。組織のメタデータにコードを保存するだけならば、このチェックボックスはオフにしておきます。このチェックボックスは、デフォルトではオンです。
5. 内容 テキストボックスで、そのトリガの Apex を入力します。1 つのトリガは、最大 1,000,000 文字までです。

トリガを定義するには、次の構文を使用します。

```
triggerName    ObjectName  trigger_events  
code_block
```

*trigger\_events* には、次のイベントを 1 つ以上含むカンマ区切りのリストを指定できます。

- 
- 
- 
- 
- 
- 
- 



メモ:

- キーワードをトリガで使用できるのは、メソッドで非同期として定義されている場合、つまりメソッドが キーワードで定義されている場合のみです。
- 定期的な行動または定期的な ToDo の 、 、 または によって呼び出されるトリガは、Force.com API からトリガが大量に呼び出されるとき、ランタイムエラーになります。

6. [保存] をクリックします。



メモ: トリガは、最後にコンパイルされて以降、依存するメタデータに変更がない限り、 フラグを に設定して保存します。オブジェクトや項目の説明の編集などの表面的な変更も含めて、トリガで使用されているオブジェクト名や項目に変更があると、Apex コンパイラがコードを再処理するまで、 フラグは に設定されます。トリガが次に実行されるときか、ユーザがトリガをメタデータに再保存するときに、再コンパイルされます。

参照項目が削除されたレコードを参照している場合、デフォルトで参照項目の値がクリアされます。または、レコードが参照関係にある場合は削除されないように選択することもできます。

## Apex トリガエディタ

Visualforce または Apex を編集するとき、Visualforce 開発モードのフッターまたは設定のいずれかで、エディタを使用できます。エディタの機能は、次のとおりです。

## 構文の強調表示

エディタは、キーワードとすべての関数および演算子について、自動的に構文を強調表示します。

### 検索 (🔍)

検索により、現在のページ、クラス、またはトリガの中のテキストを検索できます。検索を使用するには、検索 テキストボックスに文字列を入力し、[次を検索] をクリックします。

- ・ 検出した検索文字列を他の文字列で置き換えるには、置換 テキストボックスに新しい文字列を入力し、そのインスタンスだけを置き換える場合は [replace] をクリックし、そのインスタンスと、それ以外にそのページ、クラス、またはトリガに出現する検索文字列のすべてのインスタンスを置き換える場合は、[すべて置換] をクリックします。
- ・ 検索操作で大文字と小文字を区別するには、[大文字と小文字を区別する] オプションをオンにします。
- ・ 検索文字列として正規表現を使用するには、[正規表現] オプションをオンにします。正規表現は、JavaScript の正規表現規則に従います。正規表現を使った検索では、折り返されて複数行になる文字列も検索できます。

正規表現で検出した文字列を置換操作で使用する場合、検出した検索文字列から得られる正規表現のグループ変数 ( ` など) をバインドすることもできます。たとえば、`<タグ>` を `<タグ>` で置き換え、元の `<タグ>` の属性はすべてそのままにするには、`<タグ>` を検索し、それを `<タグ>` で置き換えます。

### 指定行に移動 (➡)

このボタンにより、指定した行番号を強調表示できます。その行が現在表示されていない場合は、エディタがその行までスクロールします。

### 元に戻す (⬅) およびやり直し (➡)

[Undo (元に戻す)] を使用すると編集動作を取り消します。[Redo (やり直し)] を使用すると元に戻した編集動作をやり直します。

### フォントサイズ

ドロップダウンリストからフォントサイズを選択し、エディタに表示される文字のサイズを制御します。

### 行と列の位置

カーソルの行と列の位置は、エディタ下部のステータスバーに表示されます。これは、[Go To Line (指定行に移動)] (➡) と共に使用し、エディタ内をすばやく移動できます。

### 行と文字の計数

行と文字の合計数は、エディタ下部のステータスバーに表示されます。

## トリガと Merge ステートメント

マージイベントでは、独自のトリガイベントは実行されません。その代わりに、delete イベントと update イベントが実行されます。

## 無効となるレコードの削除

1回のマージ操作により、そのマージ削除されるすべてのレコードに対して1つの delete イベントが実行されます。マージ操作の結果として削除されたレコードを判別するには、  
項目を使用します。マージ操作によりレコード削除されると、そのレコードの  
保持されるレコードの ID が設定されます。  
項目は トリガイベントでのみ  
設定されます。アプリケーションで、マージの結果削除されたレコードに特別な処理が必要な場合、  
トリガイベントを使用する必要があります。

## 保持されるレコードの更新

1回のマージ操作により、保持されるレコードに対してのみ1つの更新イベントが実行されます。マージ操作の結果、親が変更される子レコードではトリガは実行されません。

たとえば、2人の取引先責任者がマージされる場合、取引先責任者の削除トリガと更新トリガのみが実行されます。取引先責任者に関連する取引先や商談などのレコードのトリガは実行されません。

マージが行われる場合、次の順にイベントが発生します。

1. トリガが実行されます。
2. マージによって無効となるレコードが削除され、新しい親レコードが子レコードに割り当てられ、削除されたレコードの 項目が設定されます。
3. トリガが実行されます。
4. マスタレコードに必要な特定の更新を実行します。通常の更新トリガが適用されます。

## トリガと復元レコード

トリガイベントは、復元レコード(削除された後 DMLステートメントによってごみ箱から復元したレコード)に対してのみ機能します。これらのレコードは元に戻したレコードとも呼ばれます。

トリガイベントは、最上位のオブジェクトでのみ実行します。たとえば、取引先を削除すると、商談も削除されます。取引先をごみ箱から復元すると、商談も復元されます。取引先と商談の両方に関連付けられた トリガイベントがある場合、取引先の トリガイベントのみが実行されます。

トリガイベントは、次のオブジェクトでのみ実行されます。

- Account
- Asset
- Campaign
- Case
- Contact
- ContentDocument
- Contract
- カスタムオブジェクト
- Event
- Lead
- Opportunity

- Product
- Solution
- Task

## トリガと実行の順序

レコードを **新規作成**、**変更**、または **削除**ステートメントを使用して保存すると、Salesforce は次のイベントを順番に実行します。



メモ: Salesforce がサーバでこれらのイベントを実行する前に、ブラウザは、レコードに連動選択リスト項目が含まれているかどうかを JavaScript で検証します。この検証では、各連動選択リスト項目を指定可能な値に制限します。クライアント側では他に検証は行われません。

サーバで、Salesforce により次の手順が実行されます。

1. 元のレコードがデータベースから読み込まれるか、**ステートメント用にレコードが初期設定されます。**
2. 要求から新しいレコード項目の値が読み込まれ、古い値を上書きします。

要求が標準 UI 編集ページから行われた場合、Salesforce はシステム検証を実行して、レコードについて次の点を確認します。

- レイアウト固有のルールへの準拠
- レイアウトレベルおよび項目定義レベルで必要な値
- 有効な項目形式
- 最大項目サイズ

Salesforce は、要求が Apex アプリケーションや SOAP API コールなどの他のソースから送信されている場合、このステップでシステム検証を実行しません。

3. すべての **トリガが実行されます。**
4. すべての必須項目に **以外の値が入力されていることの確認や、ユーザ定義の入力規則の実行など、システム検証のほとんどの手順がもう一度実行されます。** Salesforce が標準 UI + 編集ページから要求が行われた場合に再度実行しない唯一のシステム検証は、レイアウト固有のルールの適用です。
5. レコードはデータベースに保存されますが、まだ確定されません。
6. すべての **トリガが実行されます。**
7. 割り当てルールが実行されます。
8. 自動応答ルールが実行されます。
9. ワークフロールールが実行されます。
10. ワークフロー項目自動更新が存在する場合、レコードが再度更新されます。
11. レコードがワークフロー項目自動更新により更新された場合、標準検証のほかに、**トリガおよびトリガがもう一度だけ実行されます。** カスタム検証ルールは実行されません。



メモ: 更新の必要がある場合にのみ、**トリガおよびトリガがもう一度実行されます。** 項目にすでに値が設定されている場合、トリガは再実行されません。

12. エスカレーションルールが実行されます。
13. レコードに積み上げ集計項目が含まれる場合、またはレコードがクロスオブジェクトワークフローの一部である場合、計算が実行され、親レコードの積み上げ集計項目が更新されます。親レコードに対して保存手順が実行されます。
14. 親レコードが更新され、さらにその親レコードに積み上げ集計項目が含まれるか、その親レコードがクロスオブジェクトワークフローの一部である場合、計算が実行され、親レコードの積み上げ集計項目が更新されます。親レコードのさらに親レコードに対して保存手順が実行されます。
15. 条件に基づく共有の評価が実行されます。
16. すべての DML 操作がデータベースで確定されます。
17. メール送信など、確定後のロジックが実行されます。

 メモ: 再保存時に、手順 7 から手順 14 まではスキップされます。

## 他の考慮事項

トリガを使用する場合、次の点に注意してください。

- リードの取引開始による入力規則とワークフロートリガの実行 が選択されており、リード取引開始によって作成された商談に Apex before トリガが関連付けられている場合、そのトリガは商談が作成された直後、かつ商談の取引先責任者ロールが作成される前に実行されます。詳細は、Salesforce オンラインヘルプの「リード設定のカスタマイズ」を参照してください。
- トリガを使用して商談レコードの フェーズ および 売上予測分類 を設定する場合、次のように動作します。
  - ◊ フェーズ および 売上予測分類 を設定すると、商談レコードにはこれらの正確な値が含まれます。
  - ◊ フェーズ を設定して 売上予測分類 を設定しない場合、商談レコードの 売上予測分類 はデフォルトで フェーズ トリガに関連付けられた値に設定されます。
  - ◊ フェーズ を API コールで指定した値またはユーザインターフェースから受信した値にリセットすると、売上予測分類 値も API コールまたはユーザインターフェースによって入力されます。 売上予測分類 に値を指定せず、入力された フェーズ がトリガ フェーズ とは異なる場合、 売上予測分類 はデフォルトで フェーズ に関連付けられた値に設定されます。トリガ フェーズ と入力された フェーズ が同じ場合、 売上予測分類 はデフォルト値に設定されません。
- 商品に関連する商談をコピーする場合、次のイベントが順に発生します。
  1. 親商談が上記のイベントのリストに従って保存されます。
  2. 商談商品が上記のイベントのリストに従って保存されます。

 メモ: 商談商品でエラーが発生した場合は、商談に戻ってエラーを修正してからコピーを行う必要があります。

商談商品に固有のカスタム項目が含まれている場合は、それらをすべて null に設定してから商談をコピーする必要があります。

- には、トリガを起動した特定の更新より前のオブジェクトのバージョンが含まれます。ただし、これには例外があります。レコードが更新された後にワークフロールールの項目自動更新がトリガされた場

合、最後の更新トリガの [ ] にはワークフローの更新直前のオブジェクトのバージョンは含まれず、最初の更新が実行される前のオブジェクトのバージョンが含まれます。たとえば、既存のレコードに初期値が 1 の数値項目があるとします。ユーザがこの項目を 10 に更新し、ワークフロールールの項目自動更新が起動されて項目が 11 に増えます。ワークフローの項目自動更新後に起動される更新トリガでは、から取得されるオブジェクトの項目値は、通常の場合のように 10 ではなく元の値の 1 になります。

## トリガを呼び出さない操作

トリガは、Java アプリケーションサーバによって開始または処理されるデータ操作言語 (DML) 操作に対してのみ呼び出されます。そのため、システムによるいくつかの一括処理では、トリガを呼び出しません。例には、次のものが含まれます。

- ・ 削除操作のカスケード。 [ ] を開始しなかったレコードでは、トリガの評価は行なわれません。
- ・ マージ操作の結果として親が変更される子レコードの更新のカスケード
- ・ キャンペーン状況の一括変更
- ・ 一括ディビジョン移行
- ・ 住所の一括更新
- ・ 承認申請の一括移行
- ・ メールの一括送信
- ・ カスタム項目のデータ型の変更
- ・ 選択リストの名前変更または置換
- ・ 価格表の管理
- ・ 転送ディビジョンオプションがオンになっているユーザのデフォルトディビジョンの変更
- ・ 次のオブジェクトへの変更
  - ◊ BrandTemplate
  - ◊ MassEmailTemplate
  - ◊ Folder
- ・ 取引先の更新トリガは、法人取引先レコードタイプが個人取引先に変更される前後(または個人取引先レコードタイプが法人取引先に変更される前後)には発行されません。



メモ: 個人取引先の挿入、更新、削除を行うと取引先責任者トリガではなく、取引先トリガが実行されます。

リードの取引開始処理の場合、リードの取引開始時の入力規制およびトリガが有効化されている場合のみ、次の操作に関連付けられた before トリガが実行されます。

- ・ 取引先、取引先責任者、商談の
- ・ 取引先および取引先責任者の

商談トリガは、関連付けられた商談の所有者の変更によって取引先所有者が変更される場合には実行されません。

商談の商談商品を変更する場合、または商談商品のスケジュールで商談商品が変更される場合、商談商品によって商談が変更される場合でも、商談の トリガと トリガおよび入力規制は実行されません。ただし、積み上げ集計項目が更新され、商談に関連付けられたワークフロールールが実行されます。

および PageReference メソッドは、トリガ内で使用できません。

ContentVersion オブジェクトについては、次の点に注意してください。

- ・ スライドおよびスライドの自動修正など、ContentVersion オブジェクトを使用するコンテンツパック操作は、トリガを呼び出しません。



メモ: パック内のスライドが修正されると、コンテンツパックが修正されます。

- および 項目の値は、ContentVersion レコードの作成要求または更新要求が API から作成される場合にのみトリガで使用できます。
  - トリガまたは トリガを ContentVersion オブジェクトと併用することはできません。

#### トリガのエンティティおよび項目の考慮事項

QuestionDataCategorySelection エンティティを after insert トリガで使用できない

1件以上の  
付けられた  
では レコードを挿入すると起動する トリガには、挿入された  
レコードへのアクセス権がありません。たとえば、次のクエリ  
トリガで結果を返しません。

項目を before トリガで更新できない

一部の項目値は、トリガの起動後に行われるシステムの保存操作時に設定されます。結果として、これらの項目は変更できず、またトリガまたはトリガで正確に検出できません。

\* に がない場合、 は トリガによって変更できます。

\*\* および は トリガで検出できますが、変更はできません。

#### **update** トリガでサポートされないエンティティ

特定のオブジェクトは更新できないため、 トリガおよび トリガは使用できません。

- FeedItem
- FeedComment

#### **after undelete** トリガでサポートされないエンティティ

特定のオブジェクトは復元できないため、 トリガは使用できません。

## トリガの例外

トリガを使用して、レコードまたは項目にメソッドをコールして、DML 操作が行われないようにすることができます。 トリガおよび トリガの レコード、または トリガの レコードに使用すると、アプリケーションインターフェースおよびログにカスタムエラーメッセージが表示されます。



メモ: エラーが トリガに追加されると、応答時間の遅延がほとんど生じません。

処理されるレコードのサブセットは、 メソッドでマーク付けできます。

- トリガが Apex の DML ステートメントにより実行される場合、1 つのエラーはすべての処理のロールバックを引き起こします。ただし、ランタイムエンジンはすべてのレコードを処理して、完全なエラーリストをコンパイルします。
- トリガが Force.com API の DML コールの一括処理により実行される場合、ランタイムエンジンは不正なレコードを除外し、エラーのないレコードのみを保存します。「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

トリガで未処理の例外が発生した場合、すべてのレコードがエラーとしてマーク付けされ、それ以降の処理は行われません。



メモ: このメソッドは、指定されたエラーメッセージ内のすべての HTML マークアップをエスケープします。エスケープされる文字は、< > & & #39; &amp; #39; &amp; #39; &amp; #39; および &amp; #39; です。これらの文字がエスケープされるため、HTML マークアップはレンダリングされず、Salesforce ユーザインターフェースでテキストとして表示されるようになります。HTML エスケープが実行されるバージョンは、組織で重要な更新が有効になっているかどうかによって異なります。

- ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっている場合、このメソッドでは、Apex のすべてのバージョンに対して、指定されたエラーメッセージのすべての HTML マークアップがエスケープされます。
- ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっていない場合、このメソッドでは、Salesforce.com API バージョン 27.0 以降を使用して保存された Apex に対してのみ、エラーメッセージ内のすべての HTML マークアップがエスケープされます。

## トリガと一括要求に関するベストプラクティス

よくある開発の落とし穴は、トリガの呼び出しには複数のレコードが含まれないと想定することです。Apex トリガは、一括操作ができるように最適化されています。したがって、開発者は一括操作をサポートするロジックを記述する必要があります。

これは、弱点のあるプログラミングパターンの例です。トリガの呼び出し時に取り込まれるレコードは1つのみと想定します。この場合、ほとんどのユーザインターフェースイベントはサポートされますが、SOAP API または Visualforce を使用して呼び出される一括操作はサポートされません。

これは、弱点のあるプログラミングパターンの別の例です。トリガの呼び出し時に、取り込まれるレコードは100未満と想定します。要求に20を超えるレコードが取り込まれると、トリガは、100 SELECTステートメントのSOQL クエリの制限を超えます。

ガバナ制限についての詳細は、「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

この例では、ガバナ制限を重視し、トリガの一括処理をサポートする適切なパターンを示します。

このパターンは、コレクションをセットに渡し、単一のSOQL クエリでそのセットを使用して、トリガの一括処理を重視します。このパターンは、SOQL クエリ数を制限しますが、要求が受信するすべてのレコードを取り込みます。

## 一括プログラム設計のベストプラクティス

次は、設計パターンのベストプラクティスです。

- コレクションにレコードを追加し、それらのコレクションに対してデータ操作言語(DML)の操作を実行して、DMLの数を最小化します。

- レコードを事前処理してセットを生成することによって、一句を使用する1つのSOQLステートメントに配置できるSOQLステートメント数を最小化します。

## 関連リンク

[Apex の制限事項](#)

## Apex スケジューラ

特定の時間に実行されるようにApexクラスを呼び出すには、まずクラスに  
装し、Salesforce ユーザインターフェースの [Apex をスケジュール] ページまたは  
いずれかを使用してスケジュールを指定します。

インターフェースを実  
メソッドの

**!** 重要: Salesforce は、指定された時間に実行されるようにクラスをスケジュール設定します。実際の実行  
は、サービスの使用可能状態に応じて遅れる場合があります。

一度にスケジュールできる Apex ジョブの数は 100 です。Salesforce の [スケジュール済みジョブ] ページ  
を表示し、[スケジュール済み Apex] と同じデータ型の検索条件でカスタムビューを作成することで、現  
在の個数を確認できます。また、プログラムで CronTrigger オブジェクトをクエリして、スケジュール済  
みジョブの数を取得することができますが、この場合は Apex のスケジュール済みジョブだけでなく、全  
タイプのスケジュール済みジョブが含まれます。

クラスをトリガからスケジュールする場合は、細心の注意を払ってください。トリガで許可されている  
100件を超えるスケジュールクラスを追加しないようにする必要があります。特に、APIの一括更新、イ  
ンポートウィザード、ユーザインターフェースを使用したレコードの一括変更、および複数のレコード  
を一度に更新するすべての処理については十分に考慮してください。この 100 件の同時スケジュールク  
ラスの制限は、  
メソッドを使用して開始されるスケジュール済み一括処理ジョ  
ブには適用されません。

Apex クラスに 1 つ以上のアクティブなスケジュール済みジョブがある場合、そのクラスは更新できませ  
ん。

### Schedulable インターフェースの実装

一定の間隔で実行されるように Apex クラスのスケジュールを設定するには、最初に Salesforce が提供するイン  
ターフェース  
を実装する Apex クラスを記述します。

スケジューラは、システムとして実行されます。ユーザがそのクラスの実行権限を持っているかどうかにかかわ  
らず、すべてのクラスが実行されます。

Salesforce ユーザインターフェースを使用してスケジュールされた Apex ジョブの実行を監視および停止するに  
は、[設定] から [監視] > [スケジュール済みジョブ] または [ジョブ] > [Apex ジョブ] をクリックします。

インターフェースには、実装が必要な 1 つのメソッド  
が含まれています。

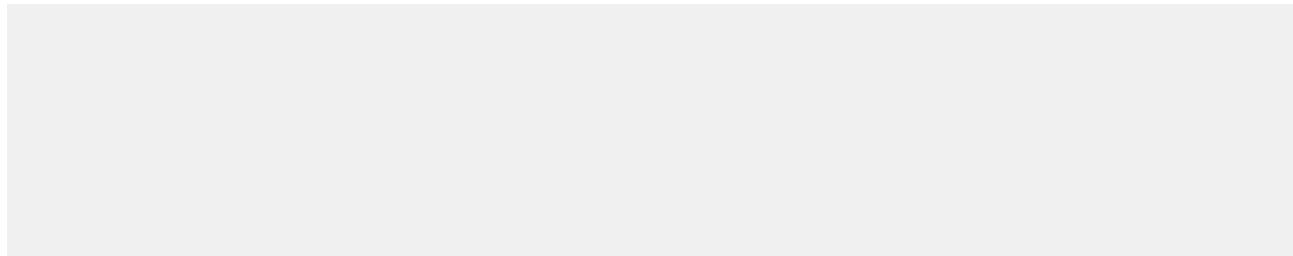
実装されたメソッドは      または      として宣言する必要があります。

このメソッドは、スケジュールを設定するクラスをインスタンス化するために使用します。



ヒント: メソッドで追加処理を行うことはできますが、すべての処理が個別のクラスで行われるようになりますことをお勧めします。

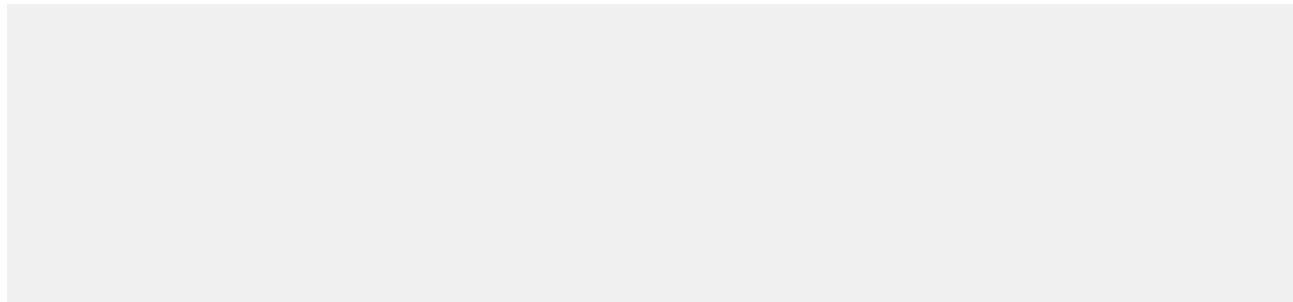
次の例では、  
と呼ばれるクラスの  
インターフェースを実装します。



次の例では、上記のクラスを実装するための  
メソッドを使用します。



Apex の一括処理クラスで  
と呼ばれる Apex の一括処理クラスの  
インターフェースを使用することもできます。次の例では、  
インターフェースを実装します。



スケジュール済みジョブを追跡するには、SchedulableContext オブジェクトを使用します。SchedulableContext メソッド  
は、このスケジュール済みジョブに関連付けられている [CronTrigger](#) オブジェクトの ID  
を文字列として返します。このメソッドは、スケジュール済みジョブの進行状況を追跡するために使用します。

スケジュール済みジョブの実行を停止するには、  
メソッドによって返された ID と共に  
メソッドを使用します。

## Apex スケジューラのテスト

次に、Apex スケジューラを使用したテスト方法の例を示します。

メソッドは、匿名プロセスを開始します。つまり、スケジュールされた Apex をテストするとき、結果に対してテストする前にスケジュール済みジョブが終了している必要があります。

メソッドを実行する前後で、テストメソッドとを使用して、テストを続行する前にスケジュール済みジョブが終了するようにします。

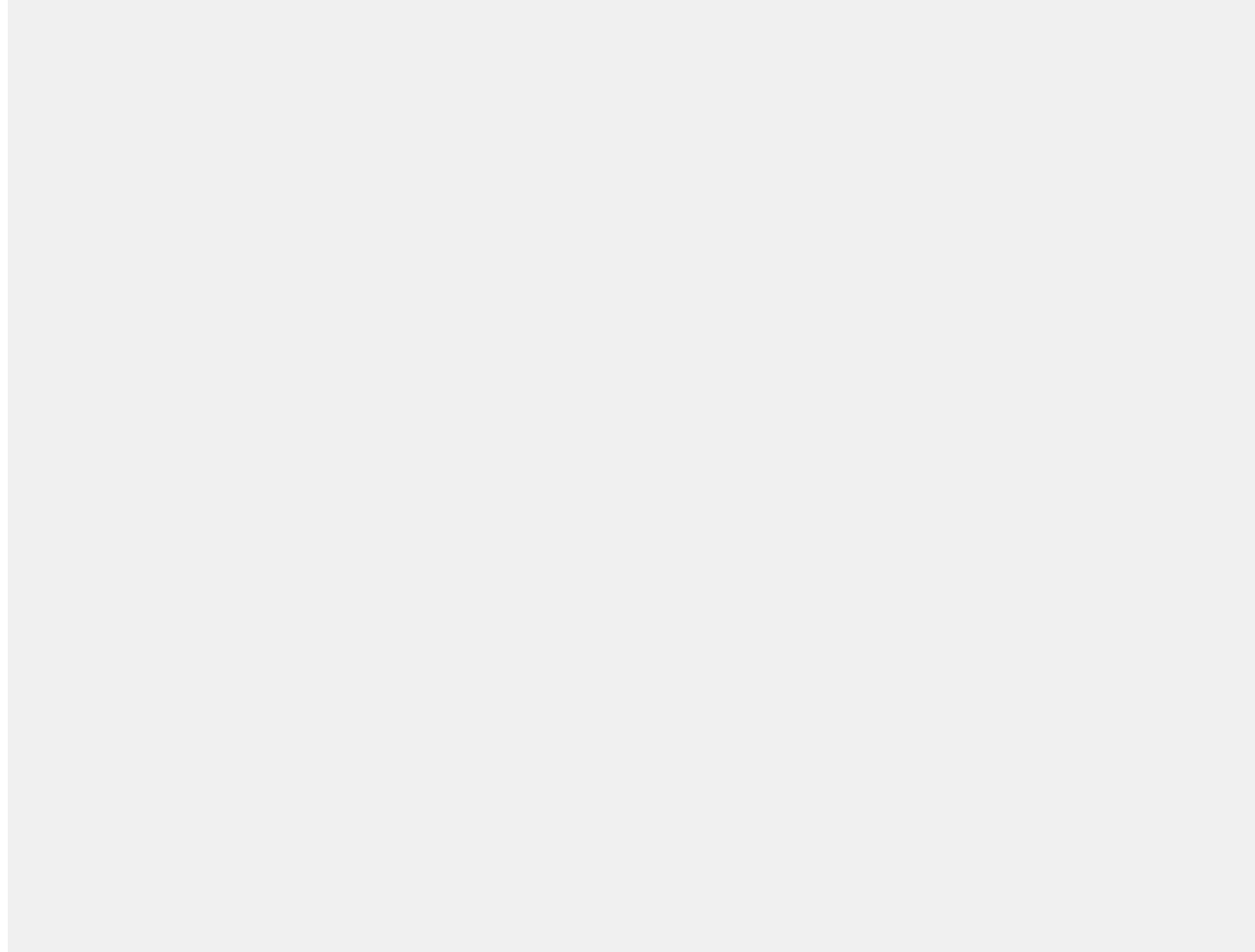
システムによって収集されます。メソッドの後に作成されたすべての非同期コールはシ

ステムによって収集されます。を実行する場合、すべての非同期プロセスが同期して実行されます。

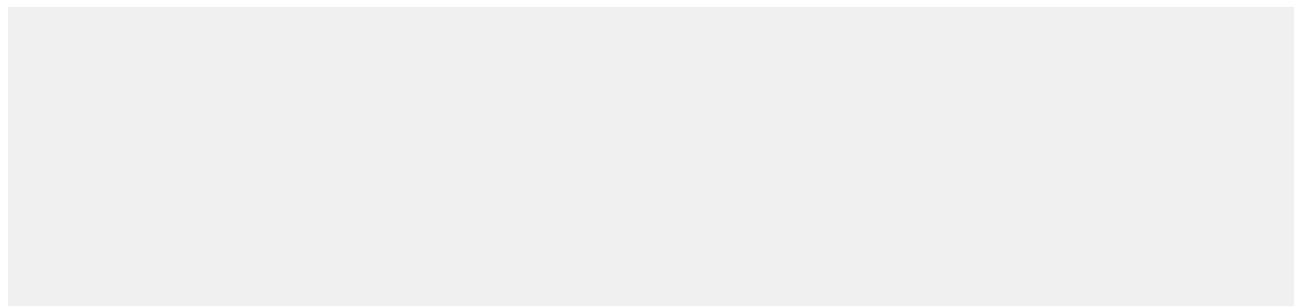
メソッドとメソッド内にメソッドを含めない場合、スケジュール済みジョブは Salesforce.com API バージョン 25.0 以降で保存された Apex のテストメソッドでは最後に実行されます

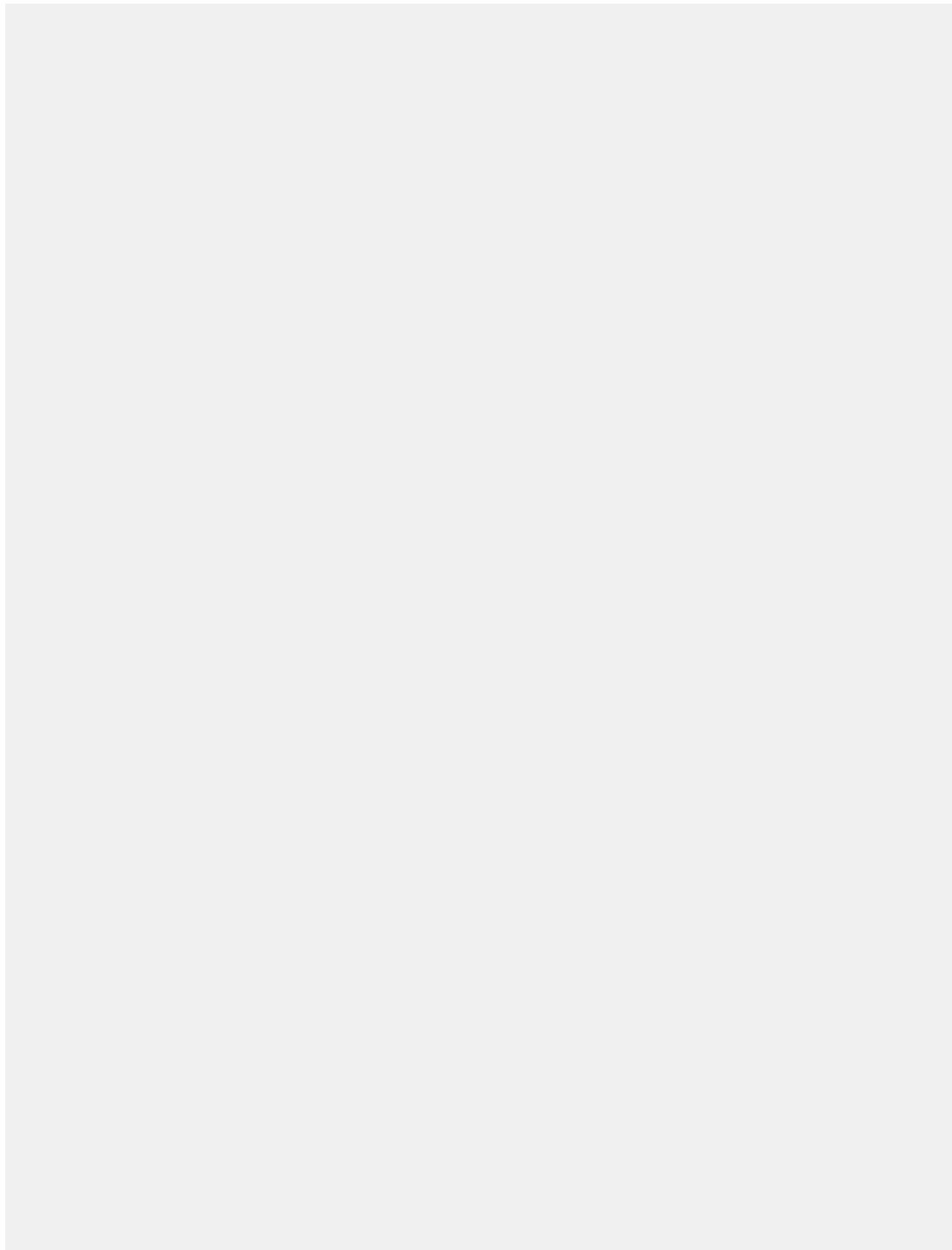
が、それよりも前のバージョンでは実行されません。

テストするクラスは次のとおりです。



次の例では、上記のクラスをテストします。





### **System.Schedule メソッドの使用**

インターフェースでクラスを実装したら、メソッドを使用してそれを実行します。スケジューラは、システムとして実行されます。ユーザがそのクラスの実行権限を持っているかどうかにかかわらず、すべてのクラスが実行されます。



メモ: クラスをトリガからスケジュールする場合は、細心の注意を払ってください。トリガで許可されている 100 件を超えるスケジュールクラスを追加しないようにする必要があります。特に、API の一括更新、インポートウィザード、ユーザインターフェースを使用したレコードの一括変更、および複数のレコードを一度に更新するすべての処理については十分に考慮してください。この 100 件の同時スケジュールクラスの制限は、  
ジョブには適用されません。  
メソッドを使用して開始されるスケジュール済み一括処理

メソッドは、ジョブの名前、ジョブの実行予定日時を表すために使用する式、クラスの名前という3つの引数を取ります。この式の構文は次のとおりです。

*Seconds Minutes Hours Day of month Month Day of week optional year*



メモ: Salesforce は、指定された時間に実行されるようにクラスをスケジュール設定します。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。

メソッドでは、すべてのスケジュールの基準としてユーザのタイムゾーンが使用されます。

式の値は次のとおりです。

名前	値	特殊文字
	• •	
<i>Day_of_week</i>	1 ~ 7、または次のとおりです。 • • • • • • •	
<i>optional_year</i>	Null または 1970 ~ 2099	

特殊文字の定義は次のとおりです。

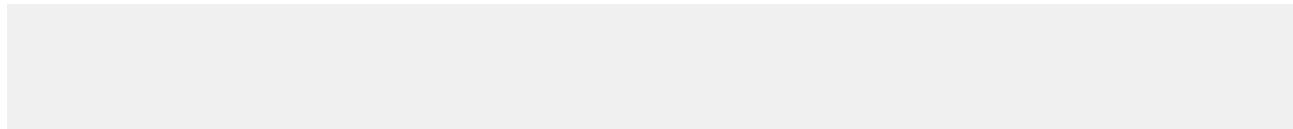
特殊文字	説明
	値を区切ります。たとえば、複数の月を指定する場合は を使用します。
	範囲を指定します。たとえば、複数の月を指定する場合は を使用します。
	すべての値を指定します。たとえば、 <i>Month</i> を と指定すると、ジョブ は毎月にスケジュールされます。
	特定の値を指定しません。 <i>Day_of_month</i> と <i>Day_of_week</i> のみで使用 でき、通常は、特定の値以外を指定しない場合に使用します。
	増分を指定します。スラッシュの前の数値は期間の開始を指定し、ス ラッシュの後の数値は期間の長さを指定します。たとえば、 <i>Day_of_month</i> に と指定した場合、Apex クラスは月の 1 日から始 まり、5 日おきに実行されます。
	範囲の終了を指定します。 <i>Day_of_month</i> と <i>Day_of_week</i> でのみ使用 できます。日で使用すると、1月の場合は 1月 31 日、うるう年の 2 月 の場合は 2 月 28 日など、 は常に月末日を意味します。 <i>Day_of_week</i> のみで使用すると、 または を意味します。 <i>Day_of_week</i> の値と 一緒に使用すると、その月で指定した曜日の最後を意味します。たとえ ば、 と指定すると、月の最終月曜日が指定されます。 と一緒に値 の範囲は使用しないでください。予期しない結果が生じる場合がありま す。
	特定の日に最も近い平日(月曜日～金曜日)を指定します。 <i>Day_of_month</i> でのみ使用できます。たとえば、 と指定し、20日が土曜日の場合、 クラスは 19 日に実行されます。 と指定すると、1 日が土曜日の場

特殊文字	説明
	<p>合、クラスはその前の月ではなく、次の月曜日である3日に実行されます。</p> <p> ヒント: 月の最後の平日を指定するには、とを一緒に使用します。</p> <p><code>weekday day_of_month</code>という形式で、月の第<code>nth</code>日目を指定します。  <code>Day_of_week</code>でのみ使用できます。の前の数値は、平日( )を指定します。の後の数値は、月の日付を指定します。たとえば、と指定すると、クラスは毎月第2月曜日に実行されます。</p>

次に、式の使用方法の例を示します。

式	説明
	クラスは毎日午後1時に実行されます。
	クラスは毎月最終金曜日の午後10時に実行されます。
	クラスは月曜日から金曜日の午前10時に実行されます。
	クラスは2010年の毎日午後8時に実行されます。

次の例では、クラスによってインターフェースが実装されます。このクラスは、2月13日の午前8時に実行するようにスケジュールされています。



### System.scheduleBatch メソッドを使用した一括処理ジョブの実行

メソッドをコールして、将来の指定された時刻に1回実行されるように一括処理ジョブをスケジュールできます。このメソッドは一括処理クラスにのみ使用できます。また、インターフェースを実装する必要はありません。これにより、一括処理ジョブが1回実行されるように簡単にスケジュール設定できます。メソッドの使用方法の詳細については、「[メソッドの使用](#)」を参照してください。

### Apex スケジューラの制限

- 一度にスケジュールできる Apex ジョブの数は 100 です。Salesforce の [スケジュール済みジョブ] ページを表示し、[スケジュール済み Apex] と同じデータ型の検索条件でカスタムビューを作成することで、現在の個数を確認できます。また、プログラムで CronTrigger オブジェクトをクエリして、スケジュール済みジョブの数を取得することもできますが、この場合は Apex のスケジュール済みジョブだけでなく、全タイプのスケジュール済みジョブが含まれます。

- 24 時間でのスケジュール済み Apex の最大実行数は、250,000 または組織のライセンス数の 200 倍の大きいほうです。これは組織全体の制限で、他のすべての非同期 Apex (Apex 一括処理メソッドおよび future メソッド) と共有されます。この制限のカウント対象となるライセンスは、Salesforce フルユーザライセンスまたは Force.com アプリケーションサブスクリプションのユーザライセンスです。Chatter 限定ユーザ、Chatter カスタマーカー、カスタマー・ポータルユーザ、およびパートナー・ポータルユーザライセンスは、含まれません。

## Apex スケジューラのベストプラクティス

- Salesforce は、指定された時間に実行されるようにクラスをスケジュール設定します。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。
- クラスをトリガからスケジュールする場合は、細心の注意を払ってください。トリガで許可されている 100 件を超えるスケジュールクラスを追加しないようにする必要があります。特に、API の一括更新、インポートウィザード、ユーザインターフェースを使用したレコードの一括変更、および複数のレコードを一度に更新するすべての処理については十分に考慮してください。この 100 件の同時スケジュールクラスの制限は、メソッドを使用して開始されるスケジュール済み一括処理ジョブには適用されません。
- メソッドで追加処理を行うことはできますが、すべての処理が個別のクラスで行われるようにすることをお勧めします。
- スケジュール済みの Apex で **メソッドは使用できません。**
- 同期 Web サービスコールアウトは、スケジュールされた Apex からは実行できません。コールアウトを実行するには、**のアノテーションを付加したメソッドにコールアウトを配置し、このメソッドをスケジュールされた Apex からコールすることで非同期コールアウトを実行します。**ただし、スケジュールされた Apex で一括処理ジョブを実行する場合、一括処理クラスからコールアウトを実行できます。  
[「Apex の一括処理の使用」を参照してください。](#)

## 匿名ブロック

匿名ブロックとは、メタデータには格納されないが、次のいずれかを使用してコンパイルおよび実行できる Apex コードです。

- 開発者コンソール
- Force.com IDE
- SOAP API コール:

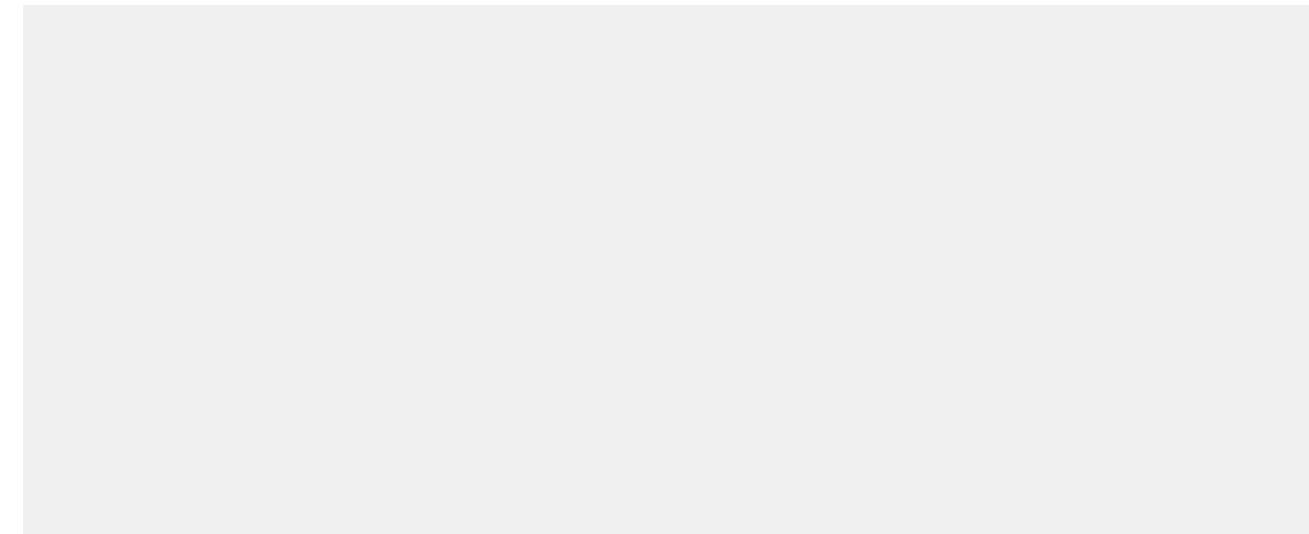
匿名ブロックは、開発者コンソールや Force.com IDE での Apex のすばやい評価や、実行時に動的に変化するコードの記述に使用できます。たとえば、名前や住所などのユーザ入力を取得して、Apex の匿名ブロックを使用し、その名前と住所の取引先責任者をデータベースに書き込むクライアント側の Web アプリケーションを記述できます。

匿名ブロックの内容については、次の点に注意してください( **、 文字列**)。

- ユーザ定義メソッドおよび例外を含めることができます。

- ユーザ定義メソッドにキーワードを含めることはできません。
- データベースの変更を手動でコミットする必要はありません。
- Apex トリガが正常に完了すると、自動的にデータベースの変更がコミットされます。Apex トリガが正常に完了しない場合、データベースへの変更はロールバックされます。
- クラスやトリガとは異なり、匿名ブロックは現在のユーザとして実行するため、コードがユーザオブジェクトの権限や項目レベルの権限に違反するとコンパイルが失敗する場合があります。
- ローカル以外の範囲を含めないでください。たとえば、アクセス修飾子を使用できますが機能しません。メソッドの範囲は、匿名ブロックに制限されています。
- 匿名ブロックにクラスまたはインターフェース (カスタムデータ型) を定義すると、匿名ブロックの実行時にそのクラスまたはインターフェースはデフォルトで仮想とみなされます。これは、カスタムデータ型が修飾子で定義されなかった場合でも同様です。これを避けるには、Salesforce にクラスまたはインターフェースを保存します。匿名ブロックに定義されたクラスやインターフェースは、組織には保存されません。

ユーザ定義メソッドは、事前に宣言せずにそのメソッド自体や後のメソッドで参照できますが、変数は宣言されるまで参照できません。次の例では、整数  は宣言する必要がありますが、 は宣言する必要はありません。



匿名ブロックで返される結果には次の情報が含まれます。

- 発生したすべてのエラーを含む、コールのコンパイルフェーズと実行フェーズの状況情報
- メソッドへのすべてのコールの出力を含むデバッグログの内容 (「[デバッグログについて](#)」(ページ 320)を参照)
- 各コールのスタック要素に対するクラス、メソッド、行番号を含む、検出されなかったすべてのコード実行例外の Apex のスタック追跡

についての詳細は、「[Apex の SOAP API および SOAP ヘッダー](#)」を参照してください。  
[「開発者コンソールのログの操作」](#) および [「Force.com IDE」](#) も参照してください。

## Apex in AJAX

AJAX Toolkit には、匿名ブロックや 公開メソッドを使用して Apex を起動するためのサポートが組み込まれています。これを行うには、AJAX コードに次の行を含めます。



メモ: AJAX ボタンの場合、これらを別の形式で使用します。

Apex を起動するには、次の 2 つのメソッドのいずれかを使用します。

- `script` を使用して匿名で実行します。このメソッドは API の結果型と似た結果を返しますが、JavaScript 構造として返します。
- WSDL クラスを使用します。たとえば、次の Apex クラスをコールします。

次の JavaScript コードを使用します。

メソッドはプリミティブデータ型、sObjects、プリミティブデータ型または sObjects のリストを使用します。

パラメータを指定せずに webService メソッドをコールするには、 の 3 つ目のパラメータに を使用します。たとえば、次の Apex クラスをコールするとします。

次の JavaScript コードを使用します。



メモ: 組織内で名前空間が定義されている場合、クラスを起動するときにその名前空間を JavaScript コードに含める必要があります。たとえば、上記のクラスをコールするには、JavaScript を次のように書き換えます。

どちらの例も、メソッドの戻り値を表すネイティブな JavaScript 値となります。

デバッグ情報を含むpopupwindowを表示するには、次の行を使用します。

## 第4章

# クラス、オブジェクトおよびインターフェース

トピック:

- [クラスを理解する](#)
- [インターフェースおよび拡張クラス](#)
- [キーワード](#)
- [アノテーション](#)
- [クラスとキャスト](#)
- [Apex クラスと Java クラスの違い](#)
- [クラス定義の作成](#)
- [クラスのセキュリティ](#)
- [オブジェクト権限と項目権限の適用](#)
- [名前空間プレフィックス](#)
- [バージョン設定](#)

クラスは、Apex オブジェクトを作成するためのテンプレート、つまり設計図で、他のクラス、ユーザ定義メソッド、変数、例外種別、および静的初期化コードで構成されます。クラスは、アプリケーションの [設定] の [開発] > [Apex クラス] に格納されます。

正常に保存されると、クラスメソッドや変数は他の Apex コードから、または、キーワードで指定されたメソッドの SOAP API (または AJAX Toolkit) を介して呼び出すことができます。

ほとんどの場合、ここで説明するクラスの概念は Java で使用される場合とほぼ同じであり、Java での経験があればすぐに理解できます。

- [クラスを理解する — Apex のクラス作成の詳細](#)
- [インターフェースおよび拡張クラス — インターフェースに関する情報](#)
- [キーワードおよびアノテーション — クラス、メソッド、または変数の追加の修飾子](#)
- [クラスとキャスト — 別のクラスへのデータ型の割り当て](#)
- [Apex クラスと Java クラスの違い — Apex と Java の相違点](#)
- [クラス定義の作成とクラスのセキュリティ — Salesforce ユーザインターフェースでのクラスの作成およびユーザによるクラスへのアクセスの有効化](#)
- [名前空間プレフィックスとバージョン設定 — 名前空間プレフィックスの使用および Apex クラスのバージョン設定](#)

## クラスを理解する

Java と同じように、Apex ではクラスを作成できます。クラスは、オブジェクトを作成するためのテンプレート、つまり設計図です。オブジェクトはクラスのインスタンスです。たとえば、クラスは、注文全体と 1 つの注文に対するすべての操作を示します。 クラスの 1 つのインスタンスが、送受信する特定の注文にあたります。

すべてのオブジェクトには、状態と動作、つまりオブジェクト自体に関する情報とオブジェクトが実行できる処理があります。PurchaseOrder オブジェクトの状態、つまりオブジェクト自体の情報には、送信元のユーザ、作成日時、重要性を表すフラグの有無などがあります。PurchaseOrder の動作、つまり実行できる処理には、在庫の確認、製品の出荷、または顧客への通知が含まれます。

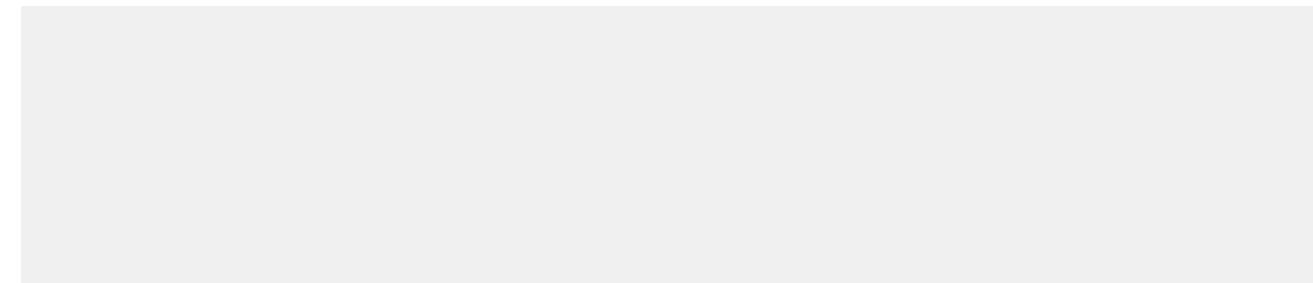
クラスには、変数とメソッドが含まれます。変数は、オブジェクトの や など、オブジェクトの状態を指定するために使用されます。これらの変数はクラスに関連付けられており、クラスのメンバーであるため、一般にメンバー変数と呼ばれます。メソッドは、や など、動作を制御するために使用されます。

インターフェースは、メソッドが実装されていないクラスのようなものです。メソッドの署名はありますが、各メソッドの本文は空です。インターフェースを使用するには、インターフェースに含まれるすべてのメソッドの本文を提供することによって、別のクラスがインターフェースを実装する必要があります。

クラス、オブジェクト、およびインターフェースに関する詳細については、  
<http://java.sun.com/docs/books/tutorial/java/concepts/index.html> を参照してください。

## Apex クラスの定義

Apex では、最上位クラス (外部クラスとも呼ぶ) と、クラス内に定義されている内部クラスの両方を定義できます。内部クラスは、1 つ下のレベルのみです。次に例を示します。



クラスを定義するには、次を指定します。

### 1. アクセス修飾子:

- 最上位クラスの宣言には、または などのアクセス修飾子の 1 つを使用する必要があります。
- 内部クラスの宣言にはアクセス修飾子を使用する必要はありません。

### 2. 省略可能な定義修飾子 ( や など)

3. 必須: クラス名の前に付ける      キーワード
4. 必要に応じて拡張および実装、またはそのいずれか

クラスを定義するには、次の構文を使用します。



- アクセス修飾子は、このクラスがローカルで表示される、つまり、コードのこのセクションのみで表示されることを宣言します。これが内部クラスのデフォルトアクセスです。つまり、内部クラスにアクセス修飾子を指定しない場合、**internal** とみなされます。このキーワードは内部クラスでのみ使用できます。
- アクセス修飾子は、このクラスがアプリケーションや名前空間で表示されることを宣言します。
- アクセス修飾子は、このクラスがすべてのApexコードで表示されることを宣言します。  
キーワードで定義されているメソッドを含むすべてのクラスは **global** として宣言する必要があります。メソッド、または内部クラスを **internal** として宣言した場合、最上位(外部)クラスも **global** として宣言する必要があります。
- **public** および **private** の各キーワードはこのクラスの共有モードを指定します。詳細は、「[キーワードの使用](#)」(ページ 196)を参照してください。
- 定義修飾子は、このクラスが拡張や上書きを許可することを宣言します。クラスが **global** として定義されていない場合、**global** キーワードを使用したメソッドの上書きはできません。
- 定義修飾子は、このクラスに抽象メソッド(署名のみが宣言され、本文が定義されていないメソッド)が含まれることを宣言します。



メモ:

- クラスが「管理-リリース済み」パッケージバージョンでアップロードされた後に、抽象メソッドを global クラスに追加することはできません。
- 「管理-リリース済み」パッケージのクラスが仮想の場合、そこに追加できるメソッドも仮想であり、実装があることが必要です。
- インストール済み管理パッケージのグローバルクラスの public または protected 仮想メソッドは上書きできません。

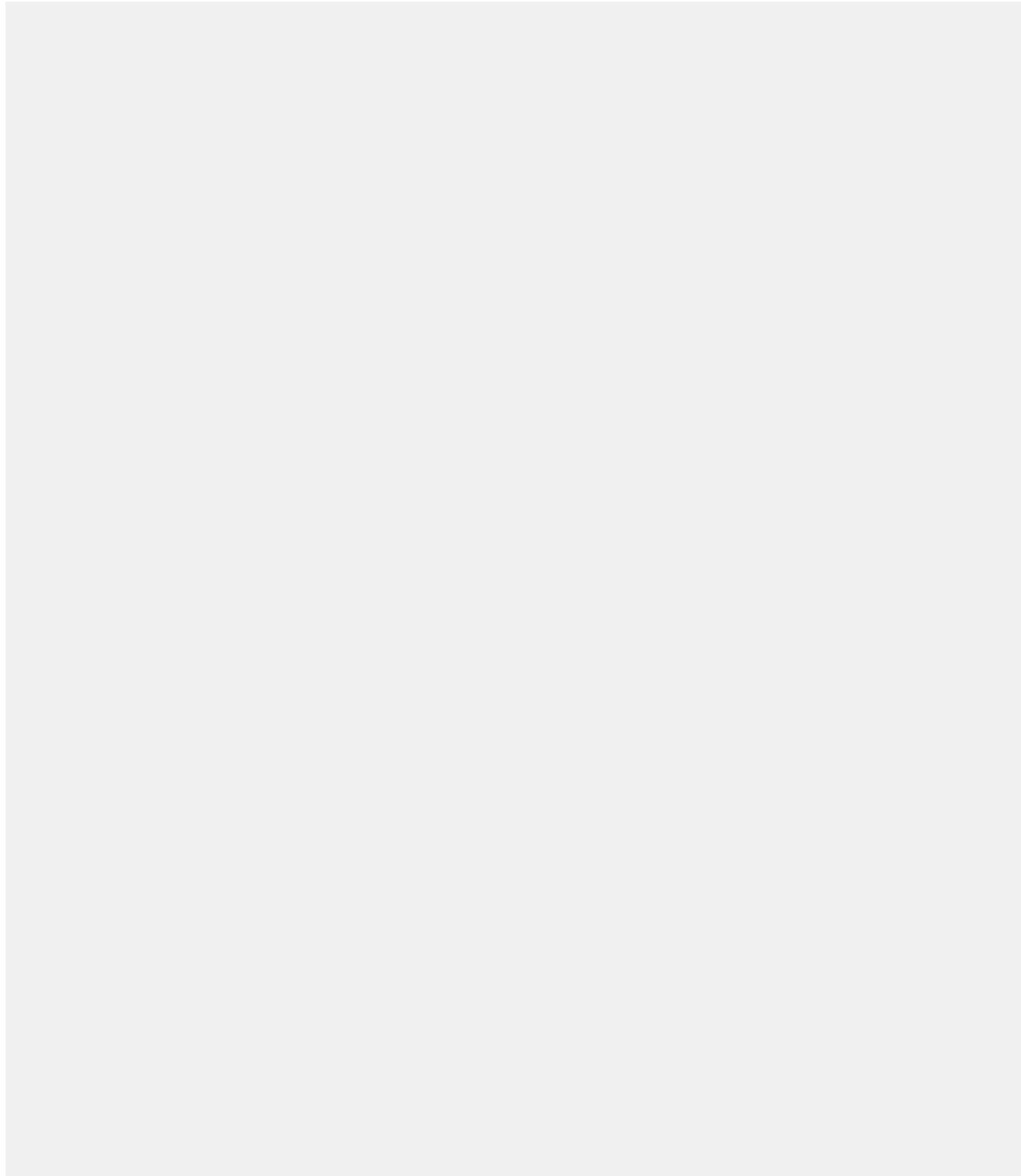
管理パッケージの詳細は、「[管理パッケージでの Apex の開発](#)」(ページ 345)を参照してください。

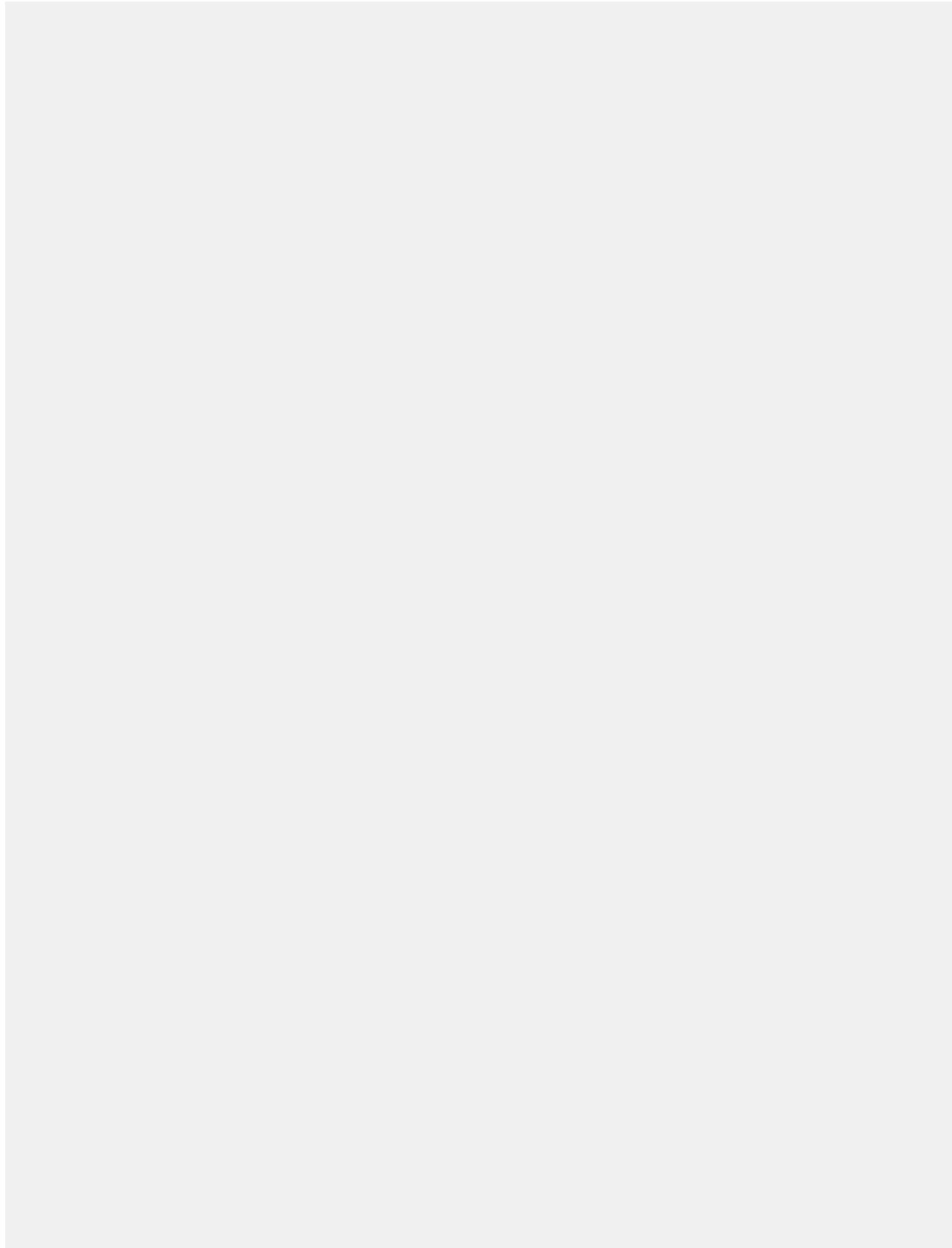
クラスは複数のインターフェースを実装できますが、既存のクラスを 1 つしか拡張できません。この制限は、Apex が複数の継承をサポートしていないことを意味しています。リストのインターフェース名はカンマで区切られています。インターフェースの詳細は、「[インターフェースおよび拡張クラス](#)」(ページ 186)を参照してください。

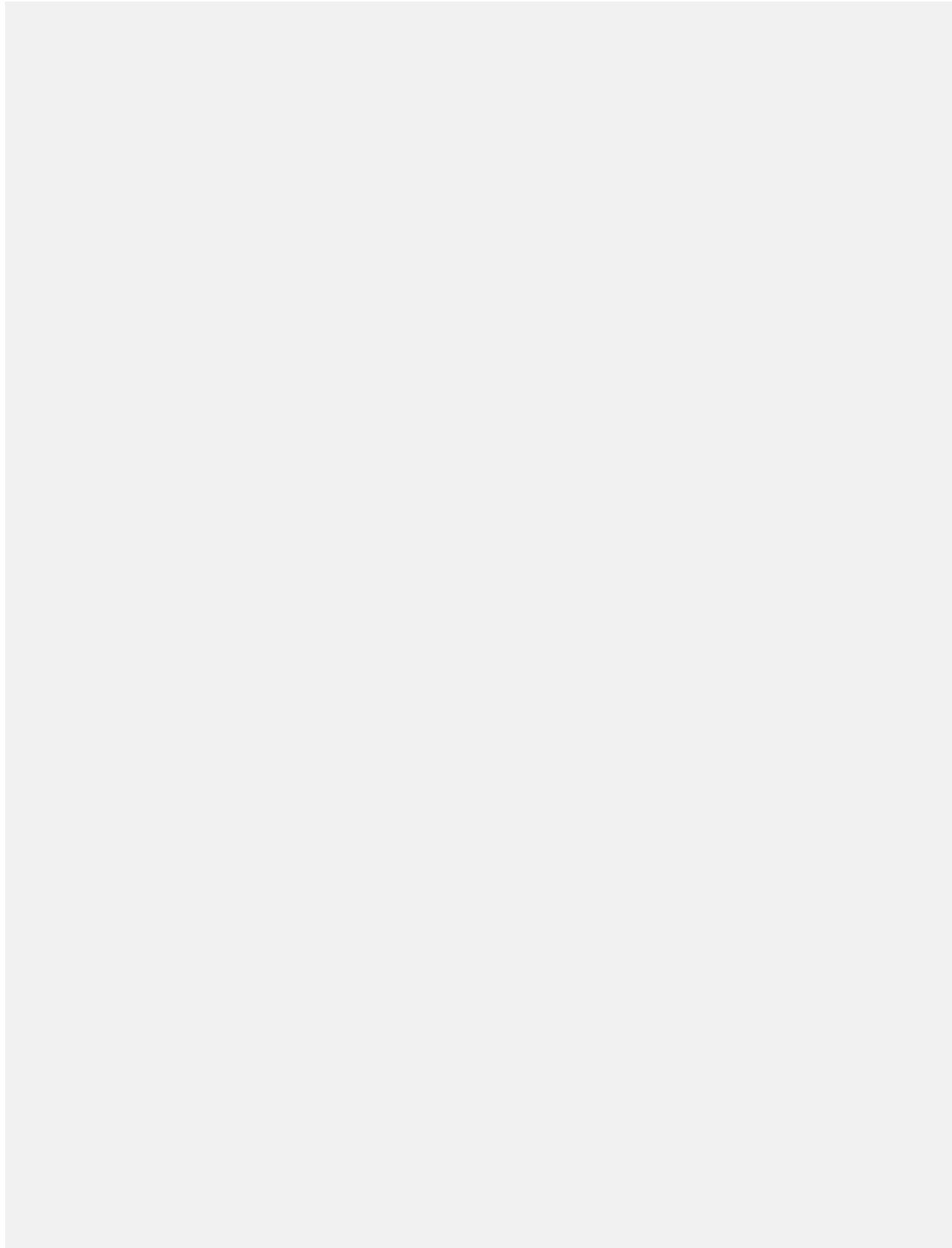
メソッドと変数のアクセス修飾子の詳細は、「[アクセス修飾子](#)」(ページ 176)を参照してください。

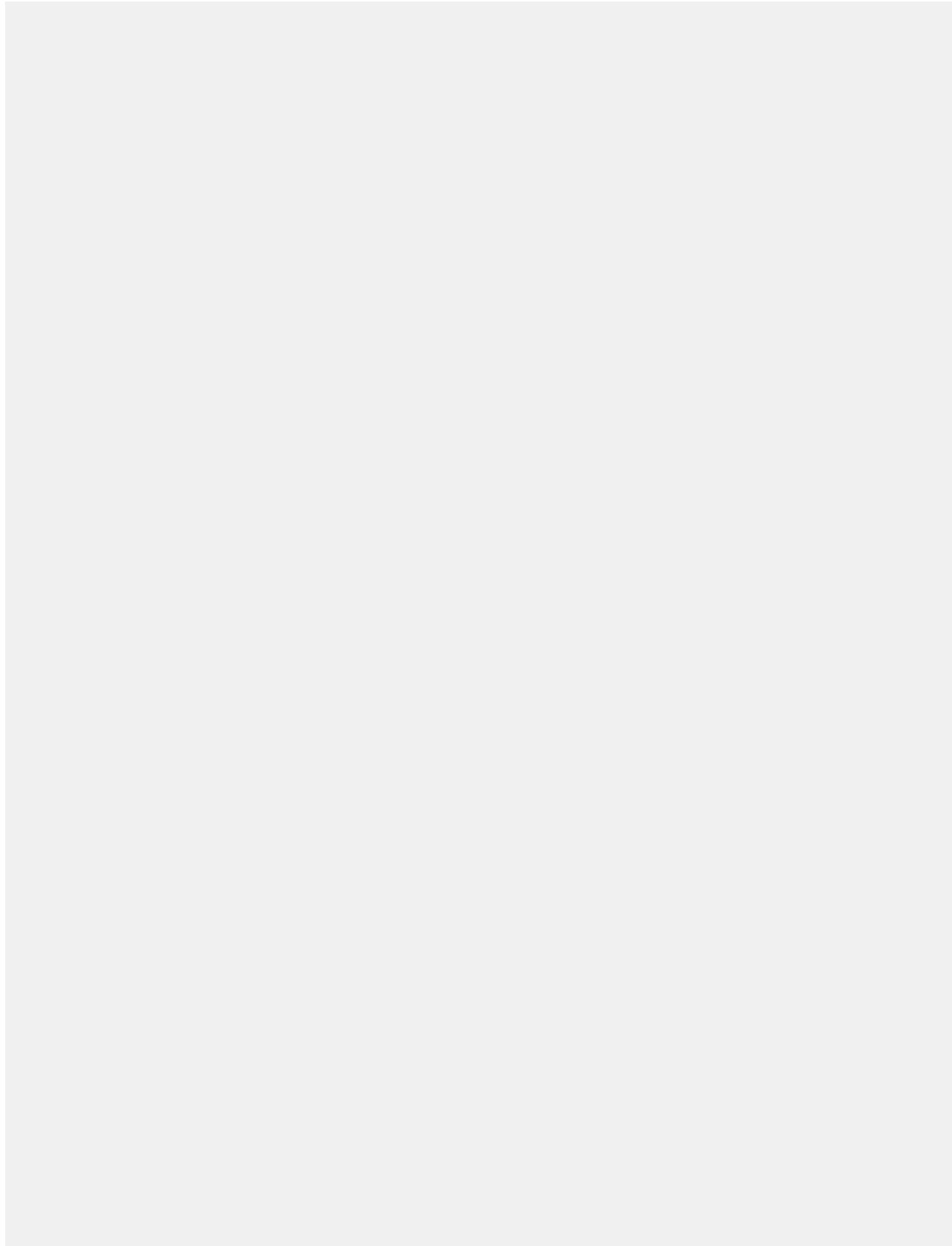
## 拡張クラスの例

クラスの拡張の例を次に示します。Apex クラスのすべての機能を示します。この例で使用されるキーワードや概念は、この章内で詳細に説明します。





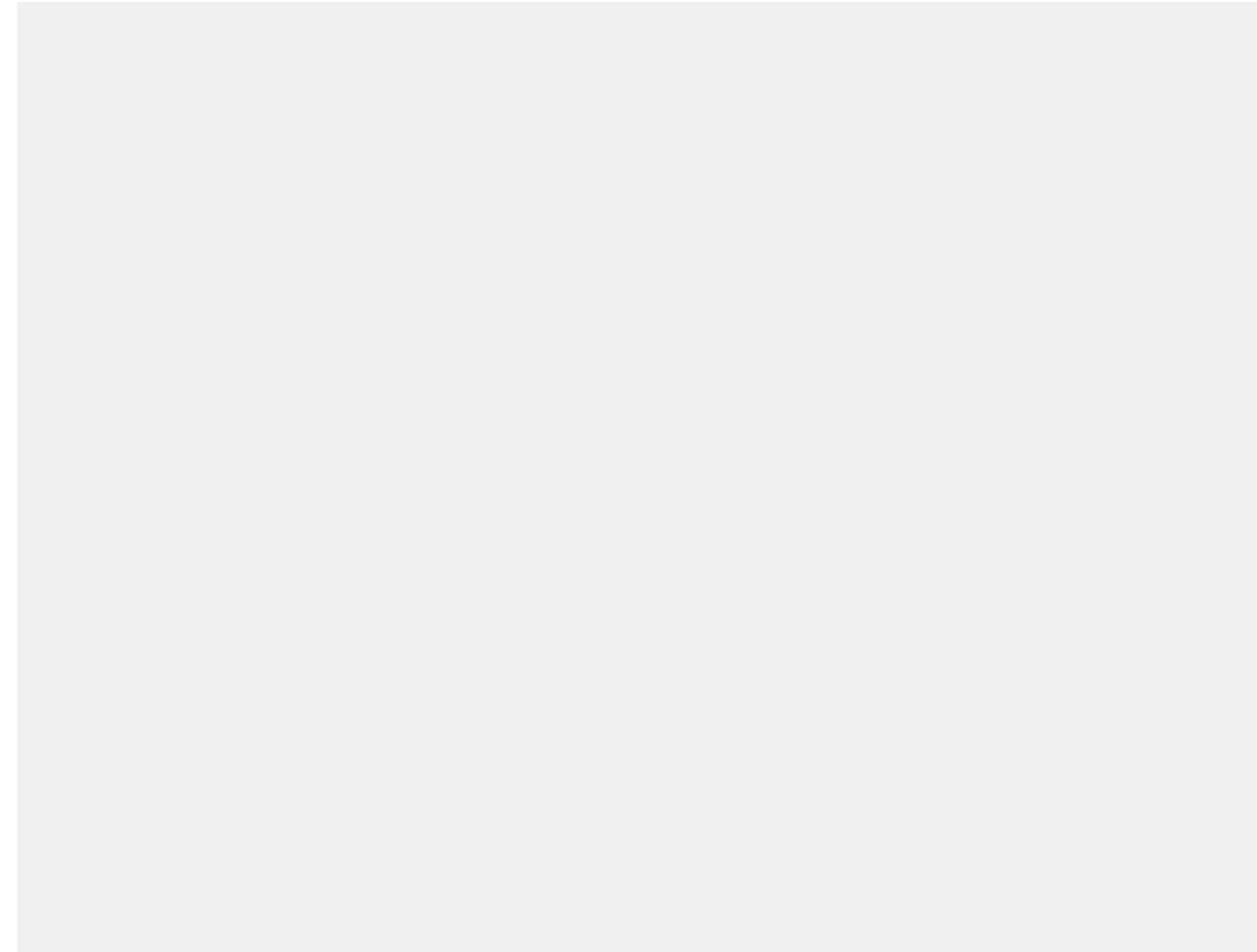




このコード例では次を示しています。

- 最上位クラスの定義 (外部クラスとも呼ぶ)
- 最上位クラスの静的変数および静的メソッド、および静的初期化コードブロック
- 最上位クラスのメンバー変数とメソッド
- ユーザ定義のコンストラクタが存在しないクラス。暗黙的に、引数をとらないコンストラクタを含む。
- 最上位クラスのインターフェース定義
- 別のインターフェースを拡張するインターフェース
- 最上位クラス内の内部クラス定義 (1つ下のレベル)
- メソッド署名の公開バージョンを実装することでインターフェース (つまり、関連付けられているサブインターフェース) を実装するクラス
- 内部クラスコンストラクタの定義と呼び出し
- 内部クラスのメンバー変数と、キーワード (引数なし) を使用したその変数の参照
- 別のコンストラクタの呼び出しにキーワード (引数なし) を使用する内部クラスコンストラクタ
- コンストラクタ外 (変数が定義されている箇所と、中かっこ ( ) で囲まれた匿名のブロックの両方) の初期化コード。これらのコードは、Java と同様にファイルに記述されている順序どおりにすべてのコンストラクションと共に実行されます。
- クラスの拡張と抽象クラス
- 基本のクラスメソッドを上書きするメソッド (として宣言する必要がある)
- サブクラスメソッドを上書きするメソッドのキーワード
- 抽象メソッドと具体的なサブクラスによる実装
- アクセス修飾子
- ファーストクラスオブジェクトとしての例外とそのメンバー、メソッド、コンストラクタ

この例では、上記のクラスを他の Apex コードからコールする方法を示します。



このコード例では次を示しています。

- 外部クラスの作成
- 内部クラスの作成と内部インターフェース型の宣言
- インターフェース型として宣言された変数を、インターフェースを実装するクラスのインスタンスに割り当て可能
- そのインターフェースを実装するクラス型にインターフェース変数をキャスト（演算子を使用した検証後）

## クラス変数の宣言

変数を宣言するには、次を指定します。

- 省略可能: `var`、`let`、`const` などの修飾子。
- 必須: `string`、`boolean` などの変数のデータ型。
- 必須: 変数の名前。
- 省略可能: 変数の値。

変数を定義するには、次の構文を使用します。

```
data_type variable_name  
value
```

次に例を示します。

```
int age = 20;
```

## クラスメソッドの定義

メソッドを定義するには、次を指定します。

- 省略可能: `や` などの修飾子。
- 必須: `string` や `integer` など、メソッドが返す値のデータ型。メソッドが値を返さない場合は、`void` を使用します。
- 必須: カンマで区切られたメソッドの入力パラメータのリスト。かっこ `( )` で囲まれます。各パラメータの前にデータ型を指定します。パラメータがない場合は、1組の空のかっこを使用します。メソッドに指定できるパラメータは 32 個までです。
- 必須: 中かっこ `{ }`  で囲まれたメソッドの本文。ローカル変数宣言を含めたメソッドのすべてのコードがここに含まれます。

メソッドを定義するには、次の構文を使用します。

```
data_type method_name  
input parameters
```



メモ: として定義されたクラスのメソッドの上書きに使用できるのは `new` のみです。

次に例を示します。

```
new int add(int a, int b) {  
    return a + b;  
}
```

Java の場合と同様に、結果が別の変数に割り当てられない場合、値を返すメソッドもステートメントとして実行できます。

ユーザ定義メソッドの次の点に注意してください。

- システムメソッドが使用されている任意の場所で使用できます。
- 再帰可能です。
- sObject ID を初期化する DML ステートメントなど、悪影響がある可能性があります。「[DML ステートメント](#)」(ページ 396)を参照してください。
- ユーザ定義メソッド自体または同じクラスまたは匿名ブロックで後で定義されたメソッドを参照できます。Apex は、2つのフェーズでメソッドを解析します。そのため、事前の宣言は必要ありません。
- 多相的な実装が可能です。たとえば、というメソッドは、1つの integer パラメータを使用する場合と、2つの integer パラメータを使用する場合の、2 通りの方法で実装できます。Apex のパーサーは、メソッドが1つの integers でコールされるか2つの interger でコールされるかによって適切な実装を選択して実行します。パーサーで完全一致を検出できない場合、データ型の強制規則を使用して、おおよその一致を検索します。データ変換の詳細は、「[変換の規則について](#)」(ページ 58)を参照してください。



メモ: パーサーがおおよその一致を複数検出した場合、解析時間の例外が生成されます。

- 副次的影響のある void メソッドを使用する場合、ユーザ定義メソッドは、通常、Apex コードのスタンドアロンの手順のステートメントとして実行されます。次に例を示します。
- 結果が別の変数に割り当たらない場合、戻り値をステートメントとして実行するステートメントを指定できます。これは Java と同じです。

## 値によってメソッド引数を渡す

Apex では、Integer または String などのすべてのプリミティブデータ型引数は、値によってメソッドに渡されます。つまり、引数への変更はメソッドの範囲内でのみ存在することになります。メソッドが返ったときに、その引数への変更は失われます。

sObject などの非プリミティブデータ型引数も、値によってメソッドに渡されます。つまり、メソッドが返ったときに、渡された引数はメソッドをコールする前と同じオブジェクトをそのまま参照することになり、別のオブジェクトを参照するようには変更できません。ただし、オブジェクトの項目の値はメソッド内で変更できます。

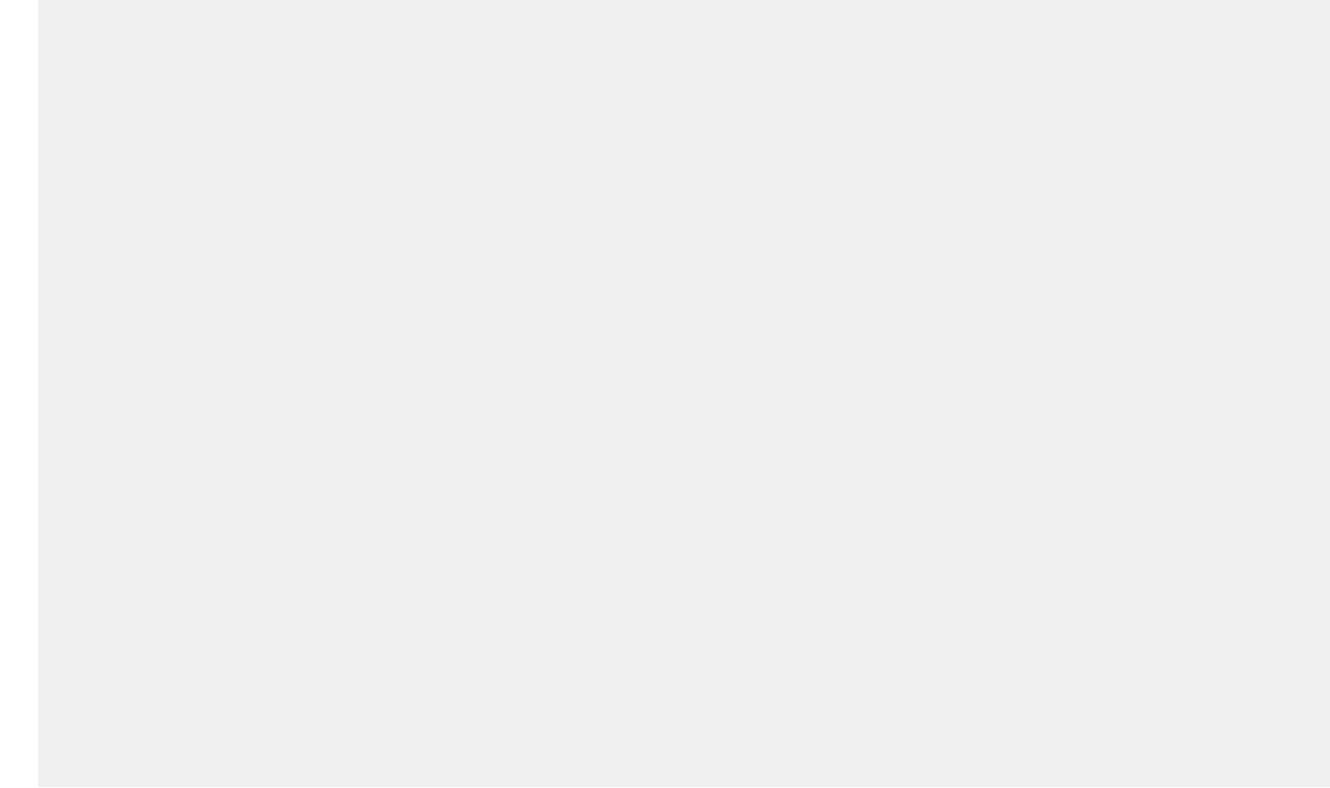
メソッドにプリミティブデータ型と非プリミティブデータ型を渡す例を次に示します。

### 例: プリミティブデータ型引数を渡す

この例では、String 型のプリミティブ引数が値によって別のメソッドに渡されることを示します。この例のメソッドは、String 変数 `msg` を作成して値を割り当てます。次に、この変数を引数として別のメソッドに渡し、この String の値を変更します。ただし、String はプリミティブ型のため、値によって渡され、メソッドが返ったときに、元の変数 `msg` の値は変更されていません。assert ステートメントは、`msg` の値が古い値のままであることを確認します。

### 例: 非プリミティブデータ型引数を渡す

この例では、List 引数を値によって別のメソッドに渡し、変更できる方法を示します。また、List 引数は別の List オブジェクトを参照するように変更できないことも示します。最初に、メソッドで変数 `fillMe` (Integer の List) を作成し、その変数を別のメソッドに渡します。コールされたメソッドは、丸められた温度値を表す Integer 値をこの List に入力します。メソッドが返ったときに、元の List 変数が変更されていて現在 5 つの値が含まれていることを assert で確認します。次に、2 番目の List 変数 `createMe` を作成し、別のメソッドに渡します。コールされたメソッドは、渡された引数を新しい Integer 値を含む新しく作成された List に割り当てます。メソッドが返ったときに、元の `createMe` 変数は新しい List は参照せず、元の空の List を参照します。assert で `createMe` に値が含まれないことを確認します。

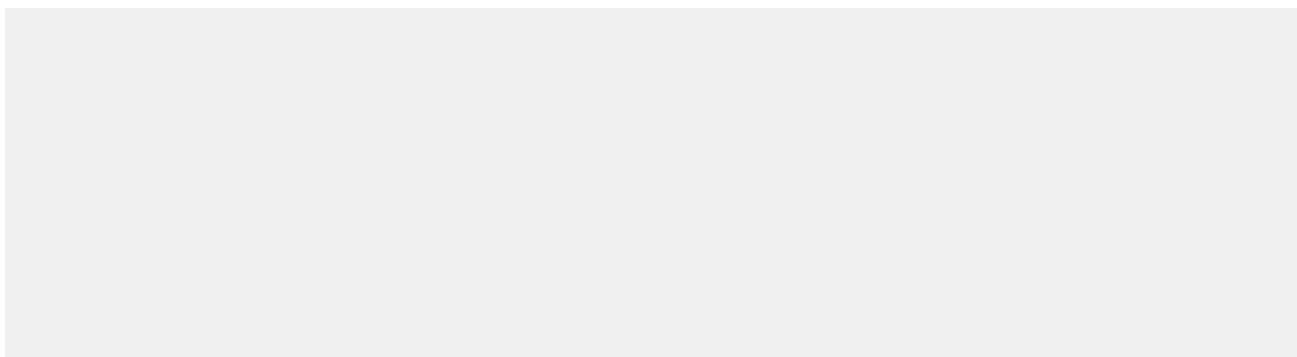


## コンストラクタの使用

コンストラクタとは、クラスの設計図からオブジェクトを作成するときに呼び出されるコードです。すべてのクラスにコンストラクタを記述する必要はありません。クラスにユーザ定義のコンストラクタが存在しない場合、引数をとらない暗黙的な公開コンストラクタが使用されます。

コンストラクタの構文はメソッドと似ていますが、コンストラクタには明示的な戻り値の型がないことと、作成元のオブジェクトから継承されないという点がメソッドとは異なります。

クラスのコンストラクタを記述した後に、コンストラクタを使用してそのクラスのオブジェクトをインスタンス化するには、**new** キーワードを使用する必要があります。たとえば、次のクラスを使用するとします。



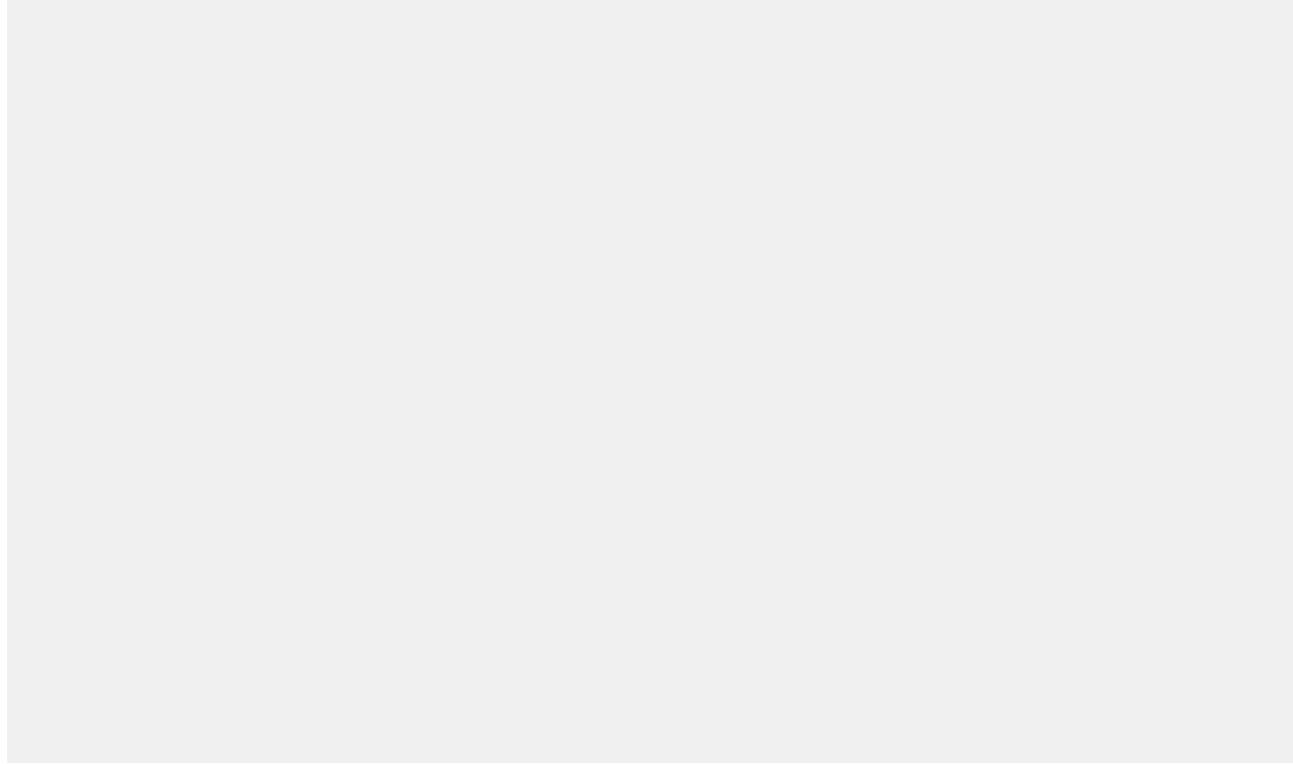
この型の新しいオブジェクトは、次のコードを使用してインスタンス化できます。

引数を取るコンストラクタを記述する場合、記述したコンストラクタを使用して、その引数を使用するオブジェクトを作成できます。引数を取るコンストラクタを作成するが、引数を取らないコンストラクタを引き続き使用する場合は、引数を取らないコンストラクタをコードに含める必要があります。いったんクラスのコンストラクタを作成すると、デフォルトの引数を取らない公開コンストラクタにアクセスすることはできません。新たに作成する必要があります。

Apex では、コンストラクタはオーバーロード、つまり、異なるパラメータを持つ複数のコンストラクタを持つことができます。次の例では、引数のないコンストラクタと、単純な整数の引数を取るコンストラクタの2つのコンストラクタを持つクラスを示します。また、コントラクタが 構文を使用して別のコンストラクタをコールする方法（コントラクタチーニングとも呼ばれる）を示します。

この型の新しいオブジェクトは、次のコードを使用してインスタンス化できます。

クラスに作成した各コンストラクタには、それぞれ個別の引数リストが必要です。適切なコンストラクタの例を次に示します。



新しいクラスを定義する場合、新しいデータ型を定義することになります。クラス名は、string、boolean、accountなど、他のデータ型の名前を使用できる場所であれば、どの場所でも使用できます。型がクラスである変数を定義する場合、それに割り当てるオブジェクトはそのクラスまたはサブクラスのインスタンスでなければなりません。

## アクセス修飾子

Apex では、メソッドや変数の定義で `private`、`protected`、`public` の各アクセス修飾子を使用できます。

トリガや匿名ブロックでもアクセス修飾子を使用できますが、Apex の狭い範囲では有用ではありません。たとえば、匿名ブロックでメソッドを `private` として宣言しても、メソッドをそのコードの外からコールすることはできません。

クラスアクセス修飾子の詳細は、「[Apex クラスの定義](#)」(ページ 163)を参照してください。



メモ: インターフェースメソッドにはアクセス修飾子はありません。常に `global` となります。詳細は、「[インターフェースおよび拡張クラス](#)」(ページ 186)を参照してください。

デフォルトでは、メソッドや変数は「定義されたクラス内でのみ」Apex コードに表示されます。メソッドや変数を同じアプリケーション名前空間の他のクラスで使用できるようにするには、明示的に `public` として指定する

必要があります(「[名前空間プレフィックス](#)」を参照)。次のアクセス修飾子を使用して表示のレベルを変更できます。

#### **private**

これはデフォルトです。メソッドや変数は定義された Apex クラス内でのみアクセスできます。アクセス修飾子を指定しない場合、メソッドや変数は **private** となります。

#### **protected**

メソッドや変数は、定義された Apex クラスのすべての内部クラスから参照できます。このアクセス修飾子は、インスタンスマソッドやメンバー変数でのみ利用できます。Java と同様にデフォルト(**private**)よりも厳密な権限付与が必要であることに注意してください。

#### **public**

メソッドや変数は、このアプリケーションや名前空間のすべての Apex クラスで使用できます。



メモ: Apex での **public** アクセス修飾子は Java の場合とは異なります。アプリケーションの結合を妨げ、各アプリケーションのコードを分離するための措置です。Java で行われるようにメソッドや変数を公開する場合、Apex では **global** アクセス修飾子を使用します。

#### **global**

メソッドや変数は、同じアプリケーションの Apex コードだけでなく、クラスへのアクセス権のあるすべての Apex コードで使用できます。アプリケーション外(SOAP API 内、または別の Apex コード)から参照されるすべてのメソッドはこのアクセス修飾子を使用する必要があります。メソッドまたは変数を **global** として宣言する場合、それを含むクラスも **global** として宣言する必要があります。

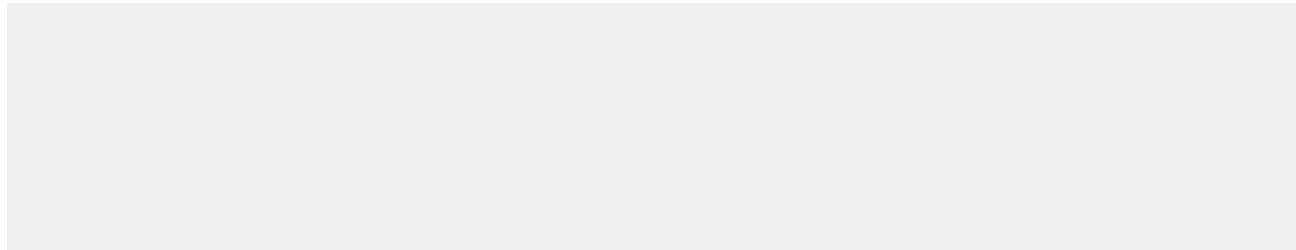


メモ: **global** アクセス修飾子は極力使用しないか、まったく使用しないことをお勧めしています。アプリケーション間の依存関係は維持が困難なためです。

、 **private** 、 **protected** 、 **public** 、 **global** アクセス修飾子を使用するには、次の構文に従います。

#### **declaration**

次に例を示します。



## 静的およびインスタンスマソッド

Apexでは、静的メソッド、変数、および初期化コードを持つことができます。Apexクラスは静的にできません。また、インスタンスマソッド、メンバー変数、および初期化コード(修飾子を含まない)とローカル変数も持つことができます。

- 静的メソッド、変数、または初期化コードは、クラスに関連付けられており、外部クラス内のみで許可されています。として、メソッドまたは変数を宣言した場合、クラスが読み込まれたときの一度だけ初期化されます。静的な変数はVisualforceページのピューステートの一部として転送されません。
- インスタンスマソッド、メンバー変数、初期化コードは、特定のオブジェクトに関連付けられており、定義修飾子はありません。インスタンスマソッド、メンバー変数、初期化コードを宣言すると、そのクラスからインスタンス化された各オブジェクトと一緒にその項目のインスタンスが作成されます。
- ローカル変数は、宣言されたコードのブロックと関連付けられます。すべてのローカル変数は、使用前に初期化する必要があります。

次は、範囲がコードブロックの持続時間であるローカル変数の例です。

### 静的メソッドと変数の使用

外部クラスの静的メソッドと静的変数のみを使用できます。内部クラスには静的なメソッドや変数はありません。静的メソッドや静的変数は、実行にクラスのインスタンスは必要ありません。

クラス内のすべての静的メンバー変数は、クラスのオブジェクトが作成される前に初期化されます。これには、静的初期化コードブロックが含まれます。これらのコードブロックは、クラス内で表示される順番に実行されます。

静的メソッドは、一般にユーティリティメソッドとして使用され、特定のインスタンスマネージャー変数の値には依存しません。静的メソッドは、1つのクラスのみに関連付けられているため、そのクラスのインスタンスマネージャー変数の値にはアクセスできません。

静的変数は、要求の範囲内のみで静的です。サーバ全体、または全体組織においては、静的ではありません。

静的変数は、クラスの範囲内で共有される情報を保存する場合に使用します。同じクラスのすべてのインスタンスは、静的変数の1つのコピーを共有します。たとえば、同じトランザクションによって生成されるすべてのトリガは、関連するクラス内の静的変数を確認したり更新したりすることで、互いに通信することができます。再帰的なトリガが、再帰を終了するタイミングを判断するためにクラス変数の値を使うこともできます。

次のクラスがあるとします。

このクラスを使うトリガは、次のように、選択的にトリガの最初の実行を失敗することができます。

トリガで定義された静的変数の値は、同じトランザクション内の異なるトリガコンテキスト間(たとえば、insertの呼び出し前と呼び出し後など)では保持されません。代わりにクラスに静的変数を定義し、トリガがこれらのクラスメンバー変数にアクセスして、静的値を確認できるようにします。

クラスの静的変数に、そのクラスのインスタンスを介してアクセスすることはできません。そのため、クラスに静的変数 `があり、` `が` のインスタンスの場合、`は不正な表現です。`

次のインスタンスマソッドに関しても同じです。`が静的メソッドの場合、` `は不正です。` 代わりに、作成するコードでは、クラス `および` `使用して、` それらの静的識別子を参照します。

ローカル変数がクラスネームと同じ名前の場合、これらのメソッドと変数は非表示になります。

内部クラスは、Javaの静的な内部クラスのように機能しますが、`キーワードを要求しません。` 内部クラスは、外部クラスのようにインスタンスマンバー変数を持つことができますが、(`キーワードを使った`) 外部クラスのインスタンスへの暗黙的ポインタはありません。



メモ: Salesforce.com API バージョン 20.0 以前を使用して保存された Apex の場合、API コールによってトリガが起動されると、200 レコードずつに分割されていたチャunkが、100 レコードずつのチャunkにさらに分割されます。Salesforce.com API バージョン 21.0 以降を使用して保存された Apex の場合、API のチャunkがそれ以上分割されることはありません。静的変数の値は、API バッチ間でリセットされますが、ガバナ制限はリセットされません。API バッチ間の状態情報の追跡に静的変数を使用しないでください。

## インスタンスメソッドと変数の使用

インスタンスメソッドとメンバー変数は、クラスのインスタンス、すなわちオブジェクトによって使用されます。インスタンスマンバー変数は、メソッド内ではなく、クラス内で宣言されます。インスタンスメソッドは通常、メソッドの動作に影響を与えるためにインスタンスマンバー変数を使用します。

二次元の点を集めるクラスを作成し、グラフ上に点をプロットするとします。次のスケルトンクラスでは、点のリストを保持するメンバー変数と、点の二次元リストを管理する内部クラスを使用します。

## 初期化コードの使用

インスタンス初期化コードは、クラス内で定義される、次の形式のコードブロックです。

クラス内のインスタンス初期化コードは、そのクラスからオブジェクトがインスタンス化されるたびに実行されます。これらのコードブロックは、コンストラクタの前に実行されます。

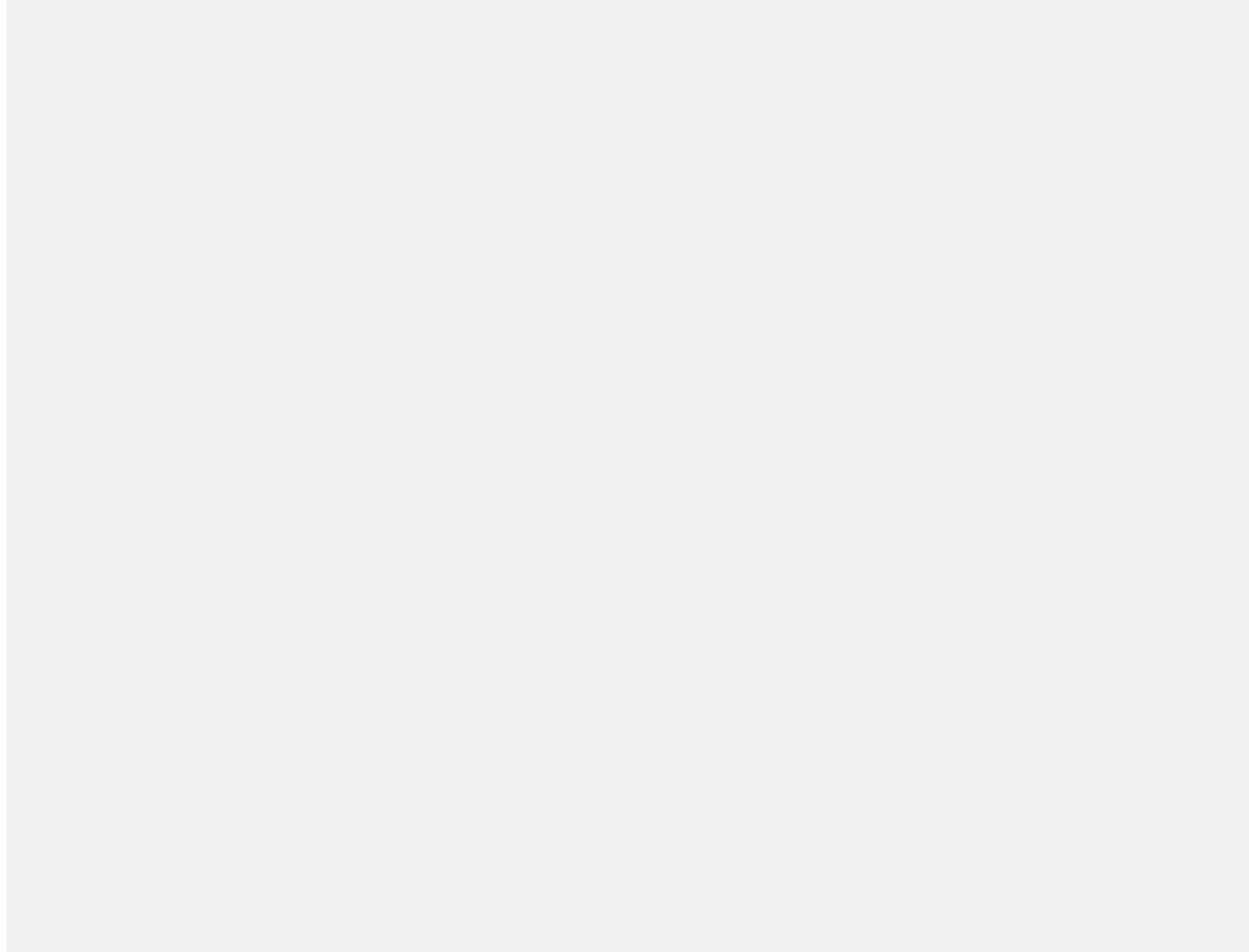
クラスに独自のコンストラクタを記述しない場合は、インスタンス初期化コードブロックを使用してインスタンス変数を初期化できます。ただし、たいていの場合はインスタンス初期化コードを使用せずに、変数にデフォルト値を設定するか、コンストラクタの本文を使用して初期化を行います。

静的初期化コードは、次のように、キーワード `static` の後に記述するコードブロックです。

他の静的コードと同様に、静的初期化コードブロックは、クラスの初回使用時に一度だけ初期化されます。

1つのクラスに、静的初期化コードブロックまたはインスタンス初期化コードブロックのいずれかを、任意の数含めることができます。コード本文のどこに記述しても構いません。Javaの場合と同様に、コードブロックはファイル内で表示される順番に実行されます。

静的なファイナル変数の初期化や、値の対応付けなどの静的な情報の宣言に静的初期化コードを使用できます。次に例を示します。



## Apex プロパティ

Apex プロパティは変数と似ていますが、アクセスまたは返される前に、プロパティ値にコードの内容を追加できます。プロパティには、さまざまな使い方があります。まず、変更の前にデータを検証できます。また、他のメンバー変数値の変更など、データが変更される前にアクションを要求できます。また、別のクラスなど、別のソースから取得したデータを表示することもできます。

プロパティの定義には、1つまたは2つのコードブロックが含まれ、*get* アクセス機構と *set* アクセス機構を表します。

- ・ プロパティが読み込まれると、*get* アクセス機構内のコードが実行されます。
- ・ プロパティが新しい値に割り当てられると、*set* アクセス機構内のコードが実行されます。

*get* アクセス機構だけを持つプロパティは、読み取り専用と考えられます。*set* アクセス機構だけを持つプロパティは、書き込み専用と考えられます。両方のアクセス機構を持つプロパティは読み書き用です。

プロパティを宣言するには、クラスの本文内で次の構文を使用します。

```
access_modifier return_type property_name
```

この場合、次のようにになります。

- *access\_modifier* はプロパティのアクセス修飾子です。プロパティに適用可能なアクセス修飾子として、  
      、      、      、      、      があります。さらに、定義修飾子      と      を適用で  
      きます。アクセス修飾子についての詳細は、「[アクセス修飾子](#)」(ページ 176)を参照してください。
- *integer*、*double*、*sObject*、など *return\_type* は、プロパティの型です。詳細は、「[データ型](#)」(ページ 30)を  
      参照してください。
- *property\_name* は、プロパティ名です。

たとえば、次のクラスは、      という名のプロパティを定義します。プロパティは *public* です。プロパティは  
*integer*x

次の点に注意してください。

- get アクセス機構の本文は、メソッドの本文に似ています。プロパティ型の値を返します。get アクセス機構を実行することは、変数の値を読み取るのと同じことです。
- get アクセス機構は、リターンステートメント内で終わる必要があります。
- get アクセス機構が定義されているオブジェクトの状態を、get アクセス機構で変更しないことをお勧めします。
- set アクセス機構は、戻り値が void のメソッドに似ています。
- プロパティに値を割り当てるとき、新しい値を渡す引数と共に、set アクセス機構が呼び出されます。
- set アクセス機構が呼び出されると、システムは、暗黙的な引数をプロパティと同じデータ型の  と呼ばれる setter に渡します。
- プロパティは、 上では定義できません。
- Apex プロパティは、C# のプロパティに基づいていますが、次の点が異なります。
  - ◊ プロパティは、値のストレージを直接提供します。ストレージ値のためにサポートイングメンバーを作成する必要はありません。
  - ◊ Apex 内に自動プロパティを作成できます。詳細は、「[自動プロパティを使用する](#)」(ページ 184)を参照してください。

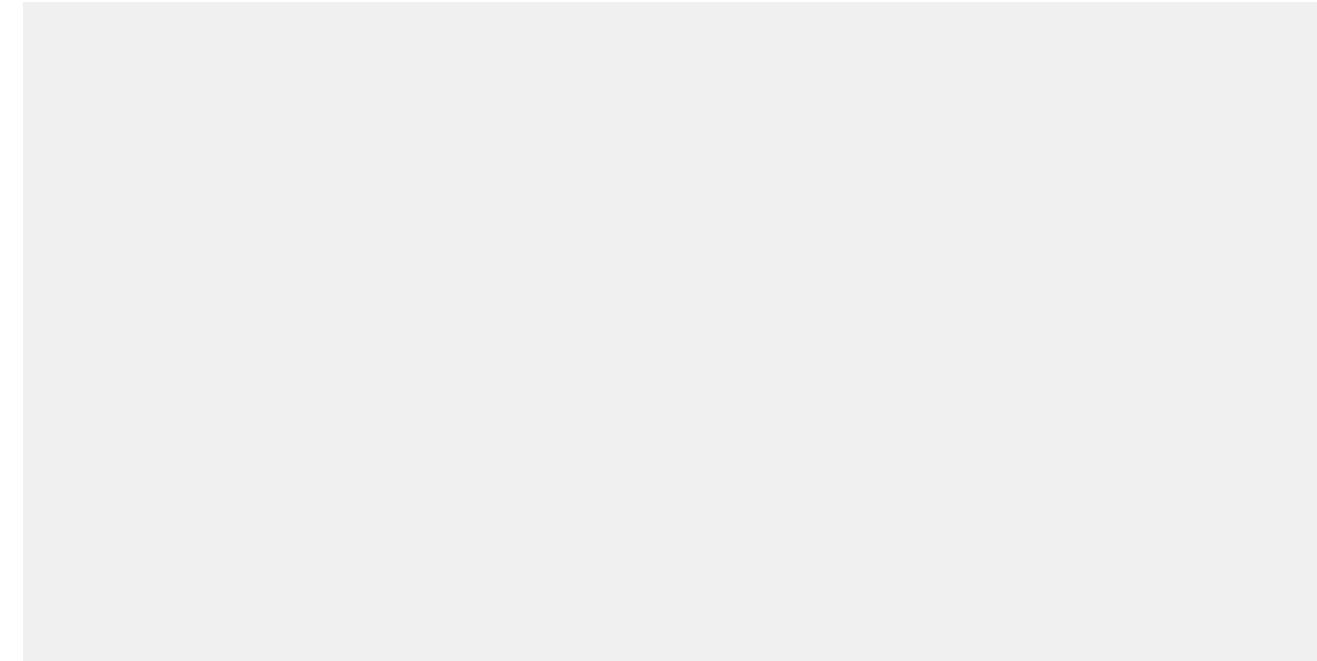
## 自動プロパティを使用する

プロパティでは、get または set アクセス機構のコードブロック内に追加コードは必要ありません。get および set アクセス機構のコードブロックを空白のままにして、自動プロパティを定義できます。自動プロパティによって、デバッグと保守が簡単な、簡潔なコードを記述できます。読み取り専用、読み書き用、書き込み専用として宣言可能です。次に、3 つの自動プロパティの例を示します。

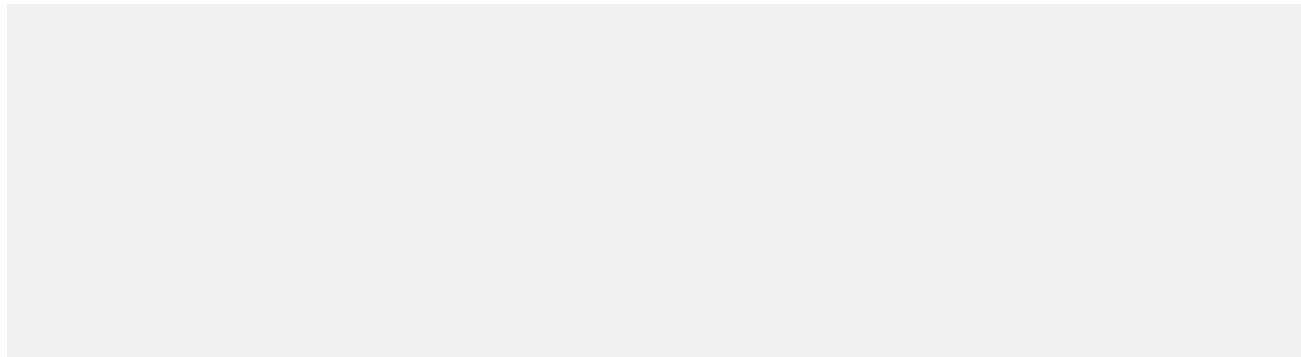
次のコードセグメントで、これらのプロパティを実行します。

## 静的プロパティの使用

プロパティを `static` として宣言すると、そのプロパティのアクセス機構方式は、静的コンテキストで実行されます。これは、そのアクセス機構に、クラスで定義されている非静的メンバー変数へのアクセス権がないことを意味します。次の例では、静的およびインスタンスプロパティの両方を持つクラスを作成します。

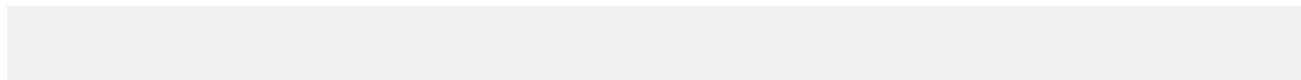


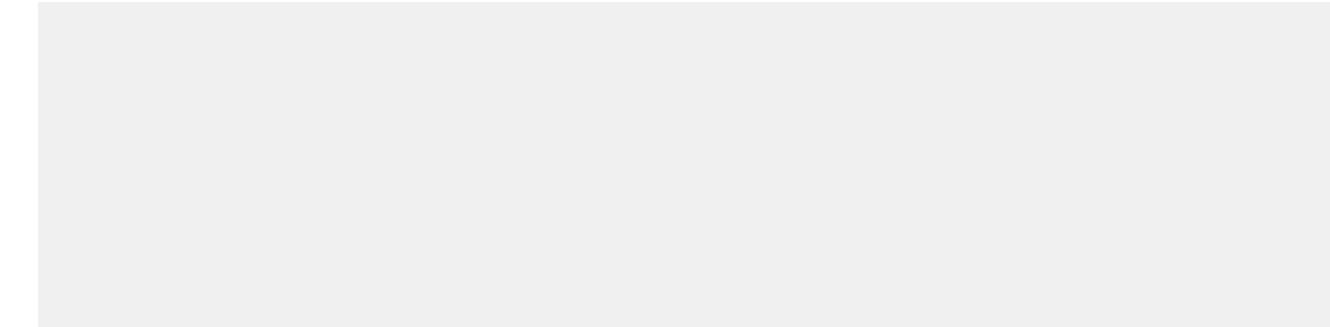
次のコードセグメントでは、静的およびインスタンスプロパティをコールします。



## プロパティアクセス機構でのアクセス修飾子の使用

プロパティアクセス機構は、自身のアクセス修飾子で定義できます。アクセス機構が自身のアクセス修飾子を含む場合、この修飾子は、プロパティのアクセス修飾子より優先されます。個別のアクセス機構のアクセス修飾子は、プロパティ自身のアクセス修飾子より限定的である必要があります。たとえば、プロパティが `private` として定義されている場合、個別のアクセス機構は、`public` としては定義できません。クラス定義の例を次に示します。





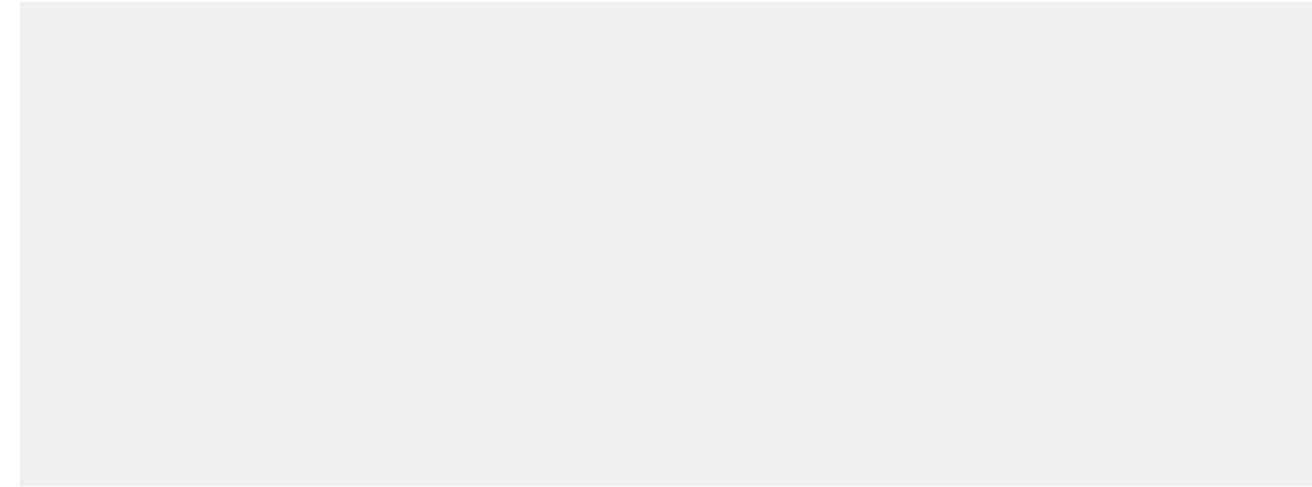
## インターフェースおよび拡張クラス

インターフェースは、メソッドが実装されていないクラスのようなものです。メソッドの署名はありますが、各メソッドの本文は空です。インターフェースを使用するには、インターフェースに含まれるすべてのメソッドの本文を提供することによって、別のクラスがインターフェースを実装する必要があります。

インターフェースにより、コードで抽象化レイヤを使用できます。インターフェースは、メソッドの特定の実装をメソッドの宣言から切り離します。これにより、1つのメソッドをアプリケーションに基づいて別々に実装できます。

インターフェースの定義は、新しいクラスの定義に似ています。たとえば、ある企業に2種類の注文があるとします。顧客からの注文と、従業員からの注文です。どちらも注文の1つのタイプです。割引をするメソッドが必要であるとします。割引額は、注文のタイプにより異なります。

注文の一般的な概念をインターフェースとしてモデリングし、顧客用および従業員用に実装します。次の例では、注文の割引についてのみ示します。



上記の例では、次の点にご注意ください。

- インターフェース は汎用的なプロトタイプとして定義されています。インターフェース内で定義されているメソッドにはアクセス修飾子はなく、その署名のみが含まれます。
- クラスはこのインターフェースを実装しているため、 メソッドの定義を提供する必要があります。Javaでは、インターフェースを実装するすべてのクラスで、インターフェースに含まれるすべてのメソッドを定義する必要があります。
- 従業員用の注文は、顧客用の注文を拡張します。1つのクラスから別のクラスに拡張するには、 キーワードを使用します。クラスは、別のクラスを1つまでしか拡張できませんが、複数のインターフェースを実装できます。

新しいインターフェースを定義する場合、新しいデータ型を定義することになります。インターフェース名は、他のデータ型の名前を使用できる場所であれば、どの場所でも使用できます。型がインターフェースである変数を定義する場合、それに割り当てるオブジェクトはインターフェースを実装するクラスのインスタンスまたはサブインターフェースデータ型でなければなりません。

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。クラスでは、インターフェースが別のインターフェースを拡張すると、拡張元のインターフェースのすべてのメソッドとプロパティが拡張先のインターフェースでも利用できます。

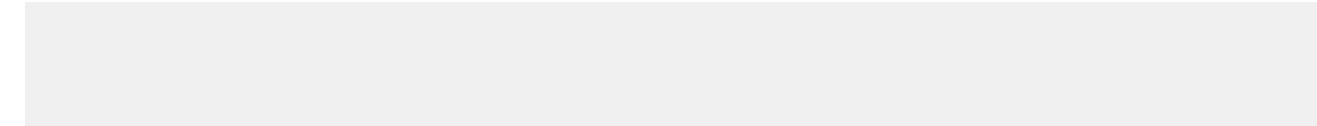
「[クラスとキャスト](#)」(ページ 213)も参照してください。



メモ: クラスが「管理-リリース済み」パッケージバージョンでアップロードされた後に、globalインターフェースにメソッドを追加することはできません。

## カスタムイテレータ

イテレータは、コレクション内のすべての項目を辿ります。たとえば、Apex の ループで、ループを終了する条件を定義し、コレクションを辿るいくつかの方法、つまりイテレータを提供する必要があります。次の例では、ループが実行されるごとに( )、 が1ずつ増加します。



インターフェースを使用して、ループ全体のリストを辿るためのカスタムの一連の指示を作成できます。通常、**ステートメント**を使用して範囲を定義する Salesforce 外のソースにあるデータに役立ちます。複数の **ステートメント**がある場合にもイテレータを使用できます。

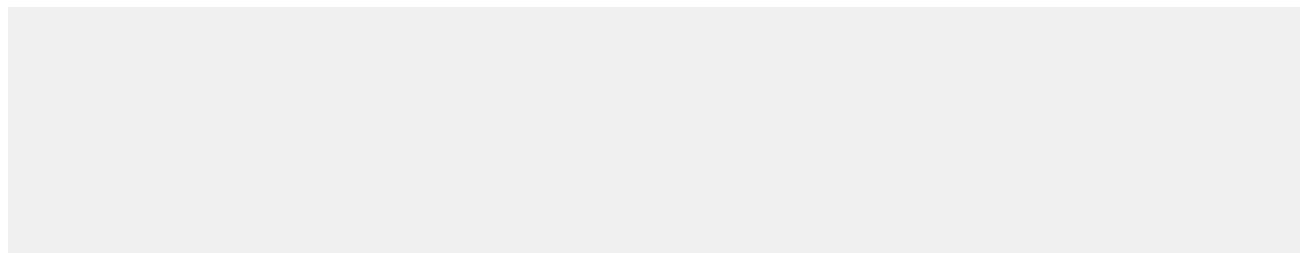
### カスタムイテレータの使用

カスタムイテレータを使用するには、インターフェースを実装する Apex クラスを作成する必要があります。

クラスには次のインスタンスマソッドがあります。

名前	引数	戻り値	説明
	Boolean		コレクション内の別の項目が辿られている場合は が返され、そうでない場合は が返されます。
	anyType		コレクション内の次の項目を返します。

インターフェース内のすべてのメソッドは または として宣言する必要があります。  
カスタムイテレータは ループでのみ使用できます。次に例を示します。



イテレータは現在、ループではサポートされていません。

### Iterable とカスタムイテレータの使用

リストでカスタムイテレータを使用せずに独自のデータ構造を作成する場合、インターフェースを使用してデータ構造を生成できます。

インターフェースには次のメソッドがあります。

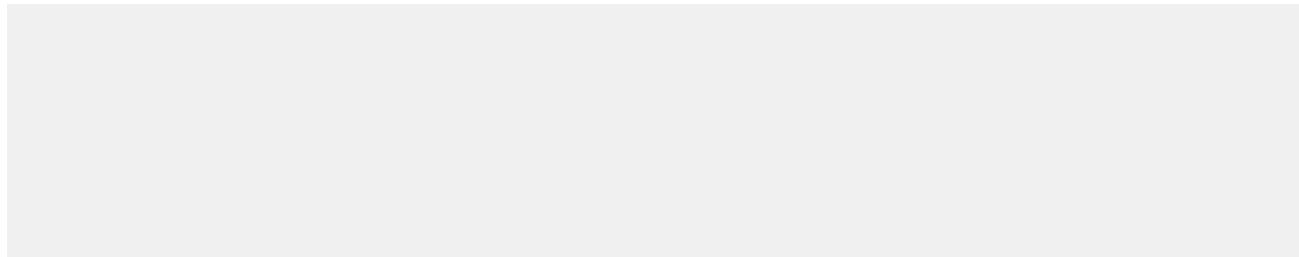
名前	引数	戻り値	説明
	Iterator クラス		このインターフェースのイテレータへの参照を返します。

メソッドは　　または　　として宣言する必要があります。データ構造の走査に使用できる  
イテレータへの参照を作成します。

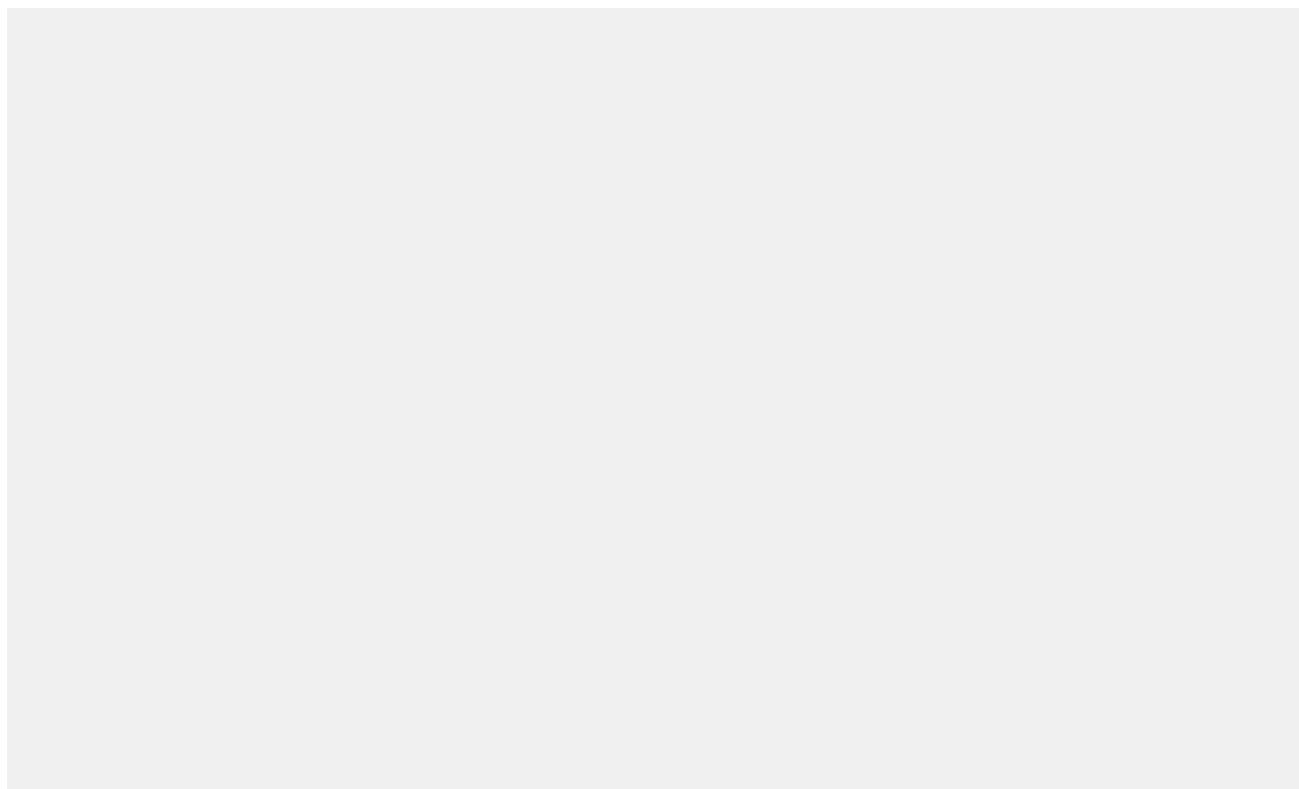
次の例では、コレクションのカスタムイテレータの例を示します。



次で、上記のコードをコールします。



次は、イテレータを使用する一括処理ジョブです。



## キーワード

Apex では、次のキーワードをサポートしています。

- 
-

- - 
  - 
  -
- と

## final キーワードの使用

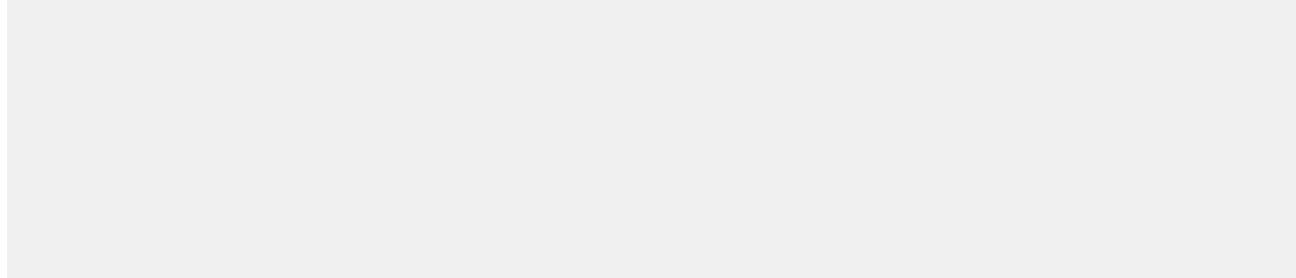
キーワードは、次のように使用できます。

- ファイナル変数には、変数の宣言時またはコードの初期化時のいずれか 1 回のみ値を割り当てることができます。このいずれかで値を割り当てる必要があります。
- 静的なファイナル変数は、静的初期化コードまたは定義時に変更できます。
- メンバーファイナル変数は、初期化コードブロック、コンストラクタ、または他の変数の宣言と共に変更できます。
- 定数を定義するには、変数を **final** および **const** の両方に定義します（「[定数](#)」（ページ 61）を参照してください）。
- ファイナルでない静的変数は、クラスレベルでの状態の通信（トリガ間の状態など）に使用します。ただし、要求間で共有されることはありません。
- メソッドおよびクラスはデフォルトで **final** です。**final** キーワードはクラスやメソッドの宣言では使用できません。つまり、上書きはできません。メソッドまたはクラスを上書きするには **override** キーワードを使用します。

## instanceof キーワードの使用

実行時に、オブジェクトが実際に特定のクラスのインスタンスであることを確認するには、**instanceof** キーワードを使用します。**instanceof** キーワードは、式中のキーワードの右にある対象の型を、キーワードの左で宣言される型の代替にできるかどうかを調べる場合のみに使用できます。

[クラスとキャスト](#) の例の **instanceof** クラスで、項目を **instanceof** オブジェクトに再度キャストする前に、次の確認を追加できます。

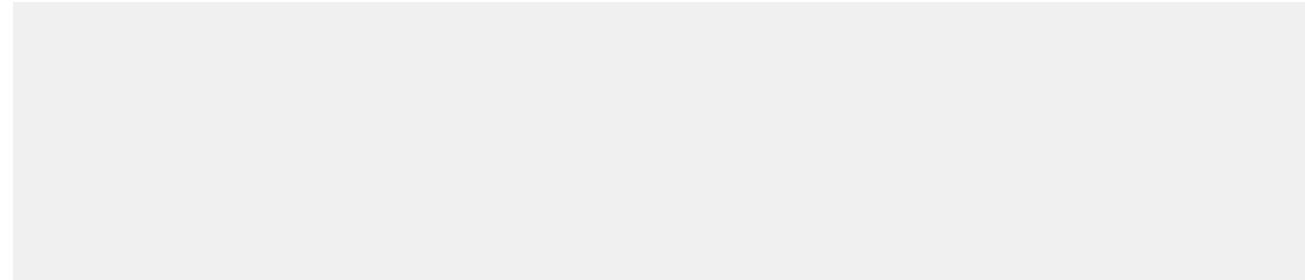


## super キーワードの使用

キーワードは、仮想クラスまたは抽象クラスから拡張されるクラスで使用できます。とによって、親クラスのコンストラクタおよびメソッドを上書きできます。

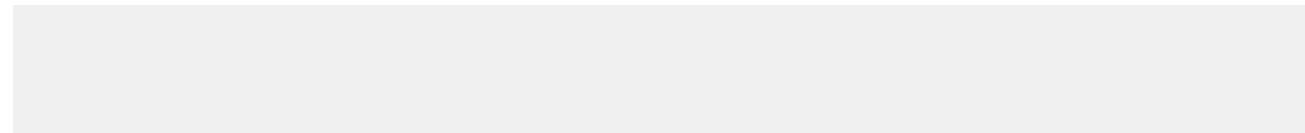
たとえば、次の仮想クラスがあるとします。

を拡張し、メソッドを上書きする次のクラスを作成できます。



をコールする場合に期待される出力は、次のとおりです。

を使用して、コンストラクタを呼び出すこともできます。次のコンストラクタをに追加します。



の期待される出力は次のとおりです。

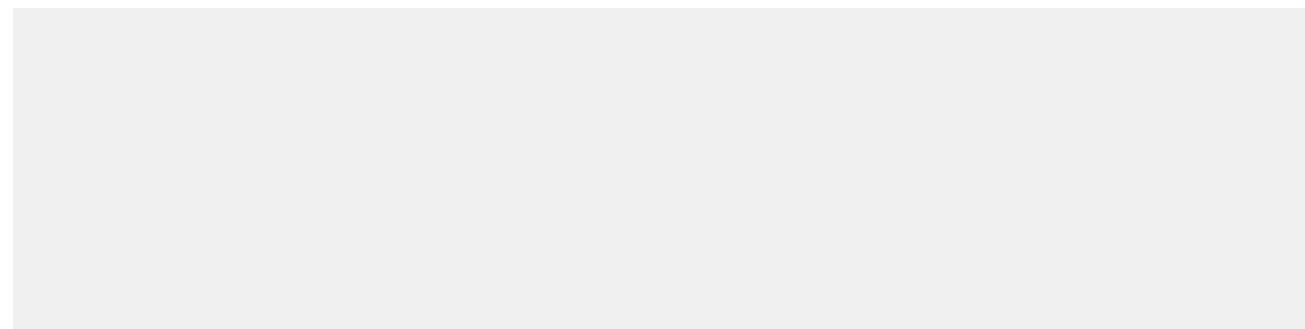
### **super** キーワード使用のベストプラクティス

- クラスまたはクラスから拡張されるクラスのみがを使用できます。
- キーワードで指定されているメソッドでのみを使用できます。

### **this** キーワードの使用

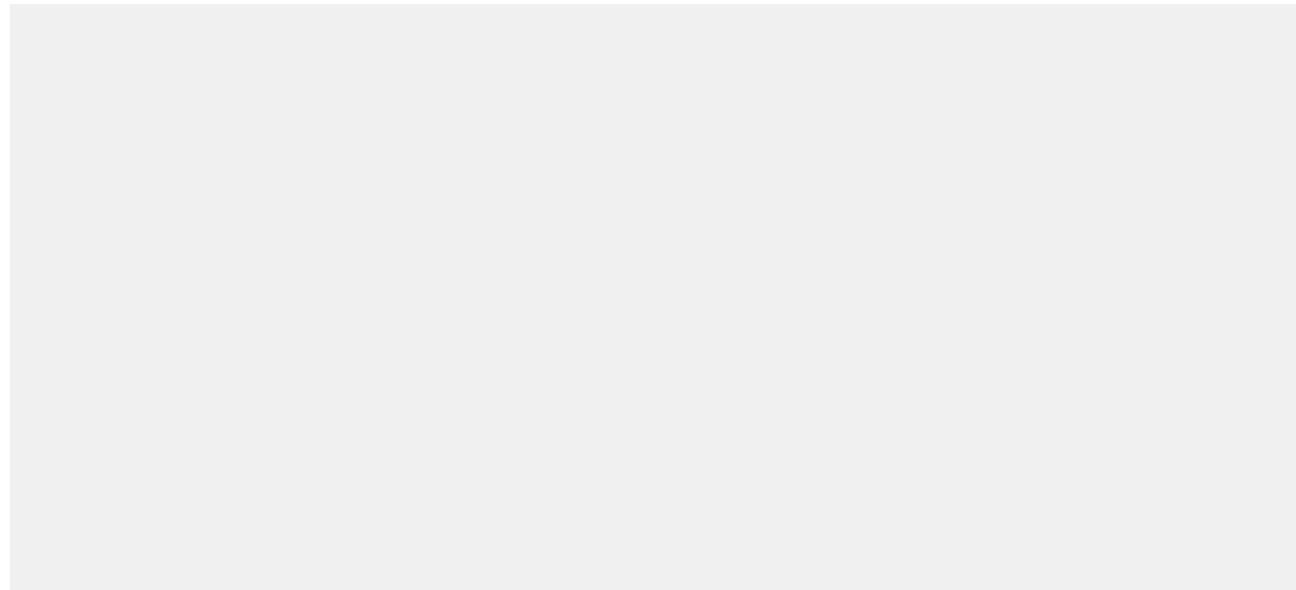
キーワードには、2つの使用方法があります。

をドット表記でかっこをつけずに使用し、表示されるクラスの現在のインスタンスを表すことができます。キーワードのこの形式は、インスタンス変数とメソッドへのアクセスに使用します。次に例を示します。



上記の例では、クラス はインスタンス変数 を宣言します。初期化コードで キーワードを使用して変数に値を設定します。

また、コンストラクタチェーニングの実行で キーワードを使用することもできます。コンストラクタチェーニングとは、1つのコンストラクタから別のコンストラクタをコールすることです。この形式では、 キーワードをかっこと共に使用します。次に例を示します。



コンストラクタでのコンストラクタチェーニングの実行で キーワードを使用する場合、コンストラクタの1つ目のステートメントに記述する必要があります。

## transient キーワードの使用

キーワードは、保存ができず、Visualforceページのビューステートの一部として送信することもできないインスタンス変数の宣言に使用します。次に例を示します。

また、逐次化可能な Apex クラス (つまり、コントローラ、コントローラ拡張、またはインターフェースを実装するクラス) で キーワードを使用できます。また、逐次化可能なクラスで宣言する項目の型を定義するクラスで キーワードを使用できます。

変数を として宣言すると、ビューステートのサイズが縮小されます。 キーワードは、Visualforce ページでページ要求の間のみ必要な項目でよく使用されます。この項目は、ページのビューステートには含まれず、要求中に何度も再計算するには非常に大きなシステムリソースを使用します。

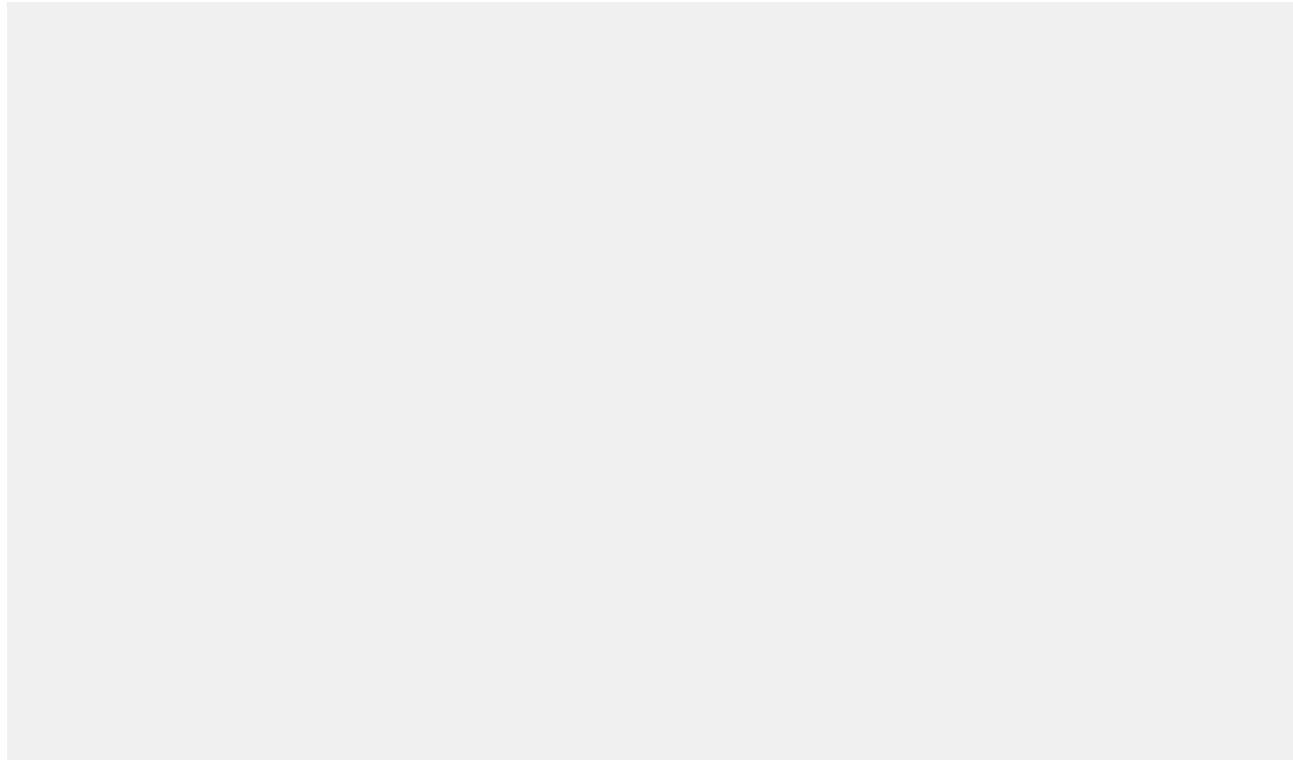
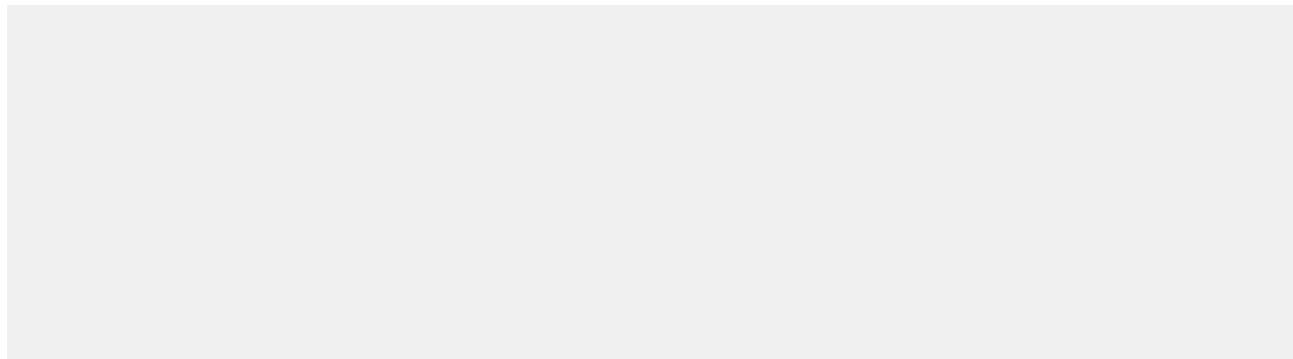
Apex オブジェクトの中には、自動的に transient と判断されるものもあります。つまり、その値はページのビューステートの一部として保存されません。例として次のようなオブジェクトがあります。

- PageReferences
- XmlStream クラス

- コレクションが自動的に transient とマーキングされるのは、Savepoints のコレクションなど、コレクションに含まれているオブジェクトが自動的に transient とマーキングされている場合だけです。
- などほとんどのオブジェクトがシステムメソッドにより自動的に生成されます。
- クラスインスタンス。詳細は、「[JSON サポート](#)」(ページ 566)を参照してください。

また、[静的な変数](#)はページのビューステートを使用して転送されません。

次の例には、Visualforce ページとカスタムコントローラの両方が含まれています。ページが更新されるごとに transient 日付は再作成されるため、[refresh] ボタンをクリックすると、日付が更新されます。非 transient 日付には、ビューステートから逐次化されなかった元の値が保持されるため、変わりません。



## with sharing または without sharing キーワードの使用

Apex は一般に、システムコンテキストで実行されます。つまり、コード実行時に、現在のユーザの権限、項目レベルセキュリティ、および共有ルールは考慮されません。



メモ: このルールの唯一の例外は、  
Apex コードです。  
コールおよび Chatter in Apex と共に実行される  
は常に、現在のユーザのフル権限を用いて実行されます。  
の詳細は、「[匿名ブロック](#)」(ページ 158)を参照してください。

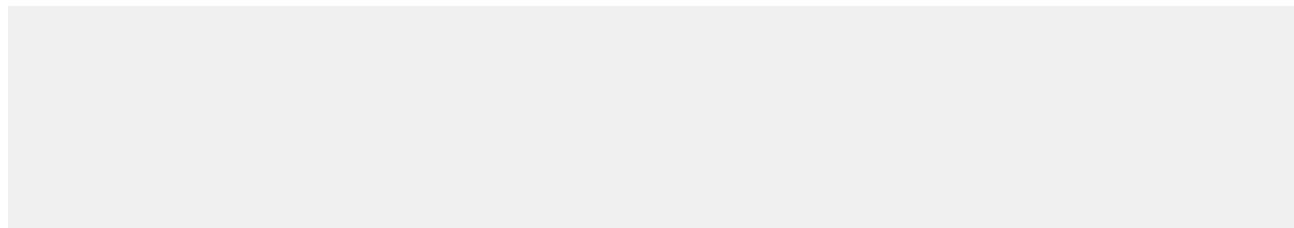
これらのルールは強制されないため、Apex を使用する開発者は、ユーザ権限、項目レベルのセキュリティ、または組織のデフォルト設定によって通常は非表示となる機密データが不注意で公開されないようにする必要があります。Web サービスについては特に注意が必要です。Web サービスは権限によって制限できますが、初期化された後はシステムコンテキストで実行されます。

多くの場合、システムコンテキストは、組織内のすべてのデータへのアクセスが必要なトリガや Web サービスなど、システムレベルの操作に対して、正しい動作を設定します。ただし、特定の Apex クラスが現在のユーザに適用されている共有ルールを強制実行するように指定することもできます(共有ルールの詳細は、Salesforce.com オンラインヘルプを参照してください)。

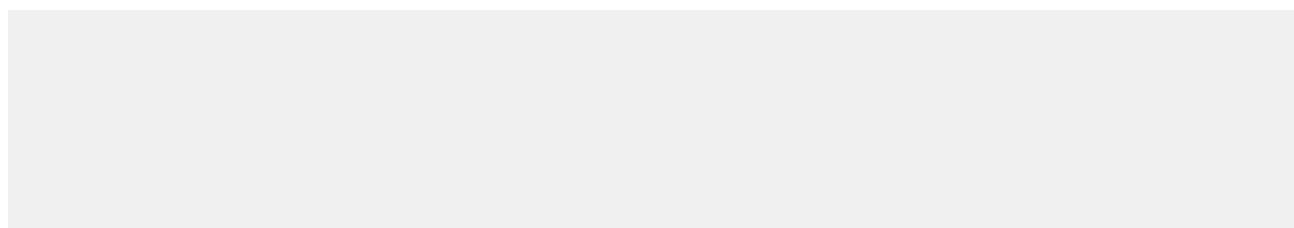


メモ: キーワードを使用した共有ルールを強制実行しても、ユーザの権限および項目レベルセキュリティは適用されません。Apex コードには、組織のすべての項目およびオブジェクトへのアクセス権が常にありますため、項目またはオブジェクトがユーザに対して非表示であるためにコードの実行が失敗することはありません。

現在のユーザに適用されている共有ルールを強制実行するには、クラスの宣言時に  
キーワードを使用します。次に例を示します。



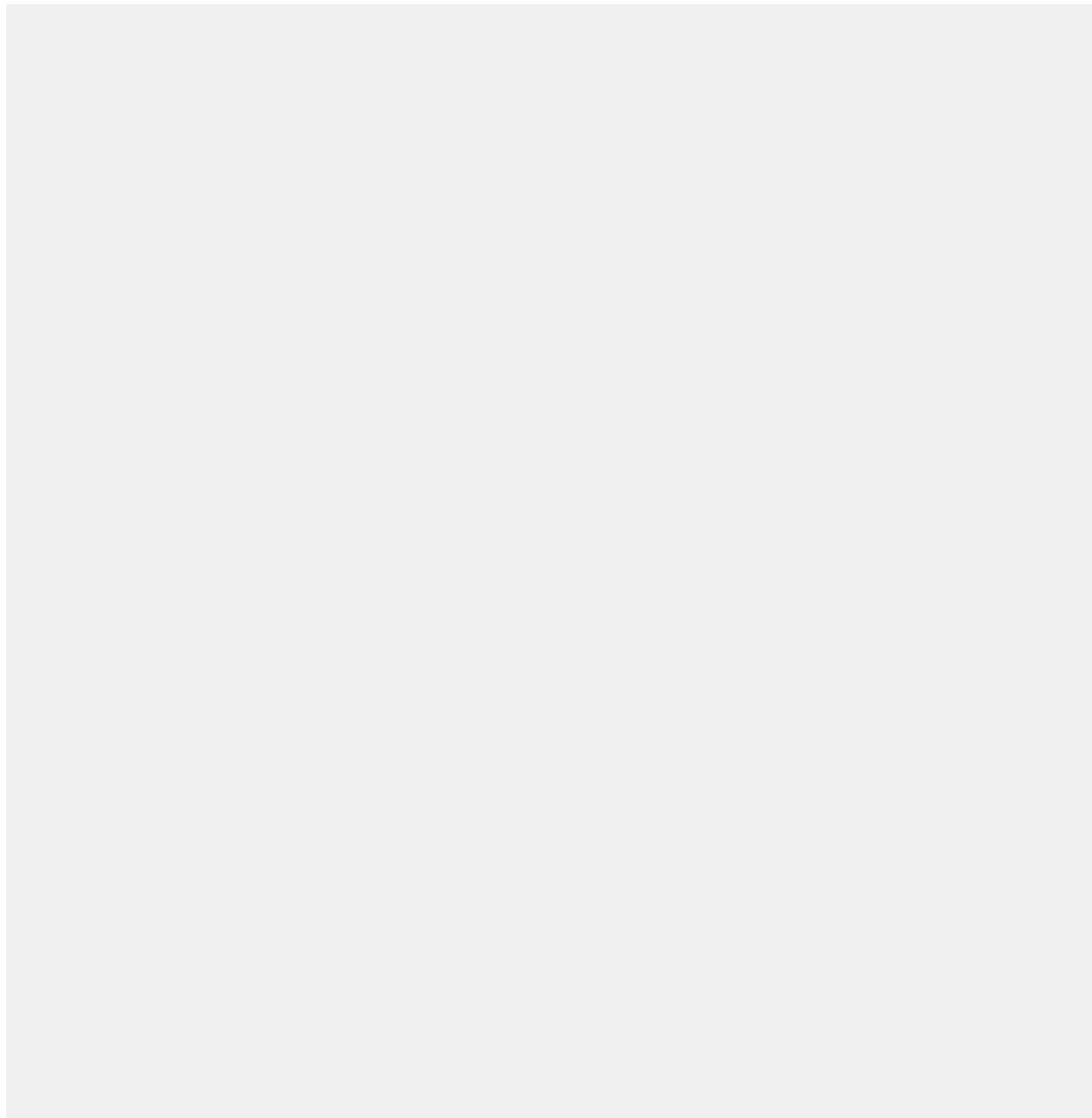
現在のユーザに適用されている共有ルールを強制実行されないようにするには、クラスの宣言時に  
キーワードを使用します。次に例を示します。

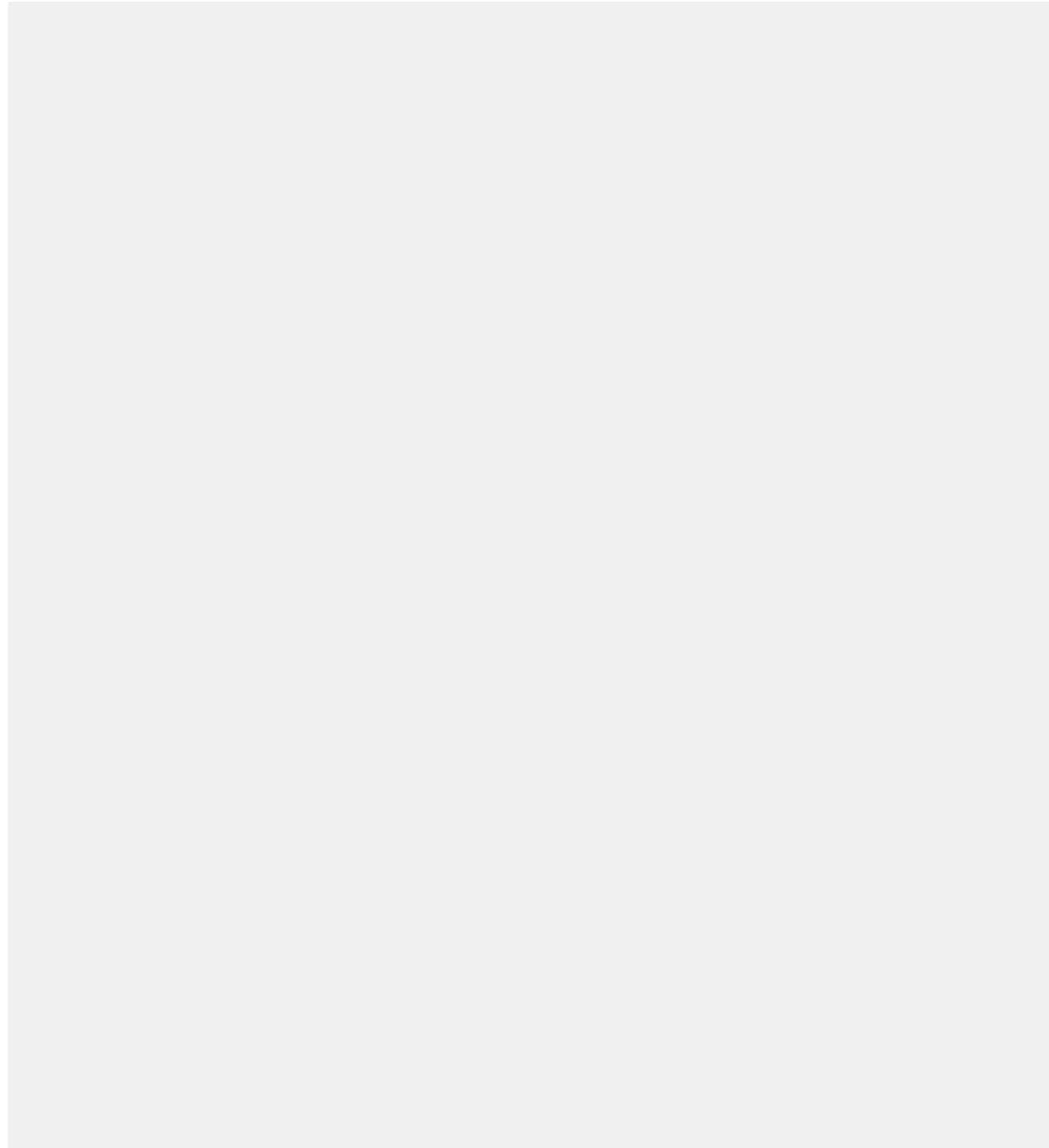


with sharing も without sharing もクラスで宣言されていない場合、現在の共有ルールが有効となります。つまり、そのクラスが、共有が強制実行されているクラスからコールされると、コールされたクラスでも共有が強制実行されます。

内部クラスと外部クラスは、どちらも `Object` として宣言できます。共有設定は、初期化コード、コンストラクタ、メソッドなどクラスに含まれているすべてのコードに適用されます。クラスは、1つのクラスを拡張または実装すると、この設定を親クラスから継承します。ただし、内部クラスはコンテナクラスから共有設定を継承しません。

次に例を示します。





警告: として宣言されたクラスが、 として動作するコードをコールしないという保証はありません。そのため、クラスレベルのセキュリティは常に必要となります。さらに、PriceBook2 を使用するすべての SOQL または SOSL クエリは、 キーワードを無視します。適用された共有ルールに関わらず、すべての PriceBook レコードが返されます。

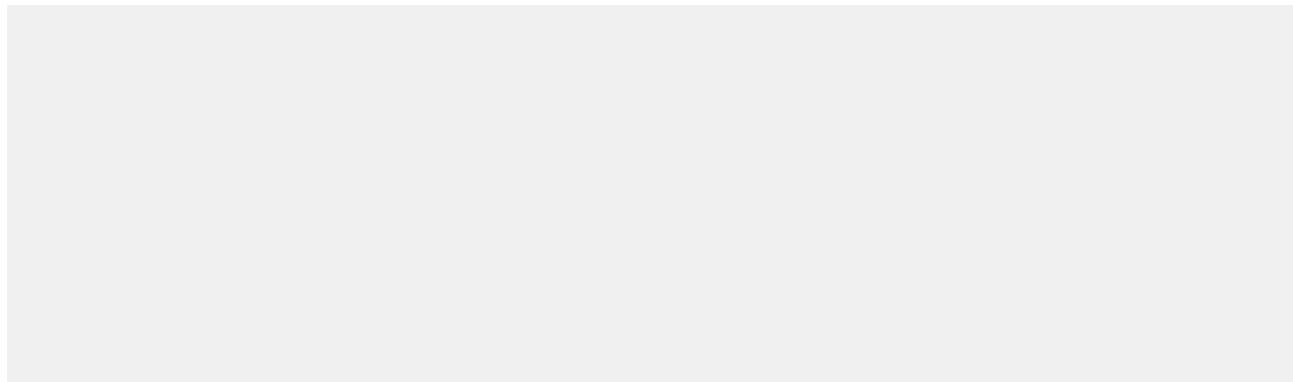
現在のユーザの共有ルールを強制実行すると次のような影響があります。

- SOQL および SOSL クエリ。クエリがシステムコンテキストで動作する場合より少ない行を返す場合があります。
- DML 操作。現在のユーザに正しい権限が付与されていない場合、操作が失敗する場合があります。たとえば、ユーザが組織内に存在する外部キー値を指定したものの、現在のユーザにはそのキー値へのアクセス権が付与されていない場合などです。

## アノテーション

Apex アノテーションは、メソッドまたはクラスの処理方法を変更するもので、Java のアノテーションと似ています。

アノテーションは先頭が 記号から始まり、適切なキーワードがそれに続きます。メソッドにアノテーションを追加するには、メソッド定義またはクラス定義の直前で指定します。次に例を示します。



Apex では、次のアノテーションをサポートしています。

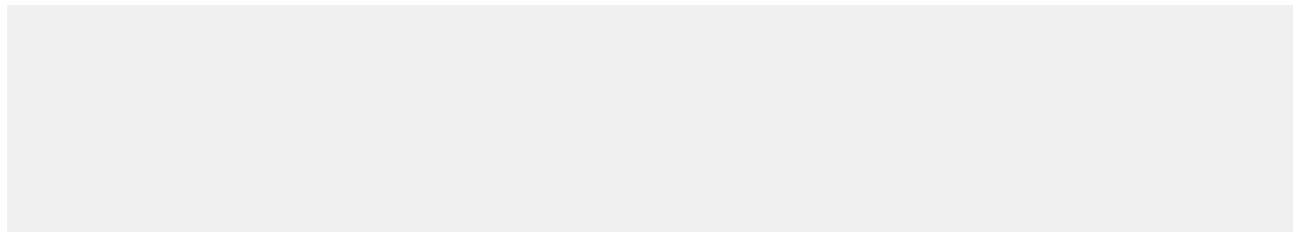
- 
- 
- 
- 
- 
- 
- Apex REST アノテーション:

```
◊           yourUrl  
◊  
◊  
◊  
◊  
◊
```

## Deprecated アノテーション

アノテーションを使用すると、今後のリリースの管理パッケージに含まれるが、参照されないメソッド、クラス、例外、列挙、インターフェース、変数を特定することができます。要件の変化にともなって、管理パッケージのコードをリファクタリングする場合に役立ちます。新しい登録者は廃止された要素を参照できませんが、既存の登録者や API 統合に対しては引き続き機能します。

次のコードスニペットは、廃止されたメソッドを示します。同じ構文を使用して、クラス、例外、列挙、インターフェースまたは変数を廃止できます。



Apex 識別子を廃止する場合、次のルールに注意してください。

- 非管理パッケージには、キーワードを使用するコードを含めることはできません。
- Apex 項目を廃止する場合、廃止する識別子を参照するすべてのアクセス修飾子も廃止する必要があります。署名や、入力引数またはメソッドの戻り値のいずれかで廃止する種類を使用する global メソッドも廃止する必要があります。パッケージ開発者は、メソッドまたはクラスなどの廃止された項目を、内部で引き続き参照できます。
- メソッドおよび変数は廃止できません。
- は廃止できますが、各 値は廃止できません。
- インターフェースは廃止できますが、インターフェースの各メソッドは廃止できません。
- 抽象クラスは廃止できますが、抽象クラスの各抽象メソッドは廃止できません。
- Apex の項目を廃止するパッケージをリリースした後は、アノテーションを削除しても Apex のその項目の廃止を取り消すことはできません。

パッケージバージョンの詳細は、「[管理パッケージでの Apex の開発](#)」(ページ 345)を参照してください。

## Future アノテーション

非同期で実行するメソッドを特定するにはアノテーションを使用します。を指定すると、Salesforce に使用可能リソースが存在するときにこのメソッドが実行されます。

たとえば、外部サービスへの非同期の Web サービスコールアウトを実行するときにアノテーションを使用できます。このアノテーションを使用しない場合、Web サービスコールアウトは Apex スクリプトを実行している同じスレッドから実行され、コールアウトが完了するまで他の処理は実行されません(同期処理)。

アノテーションのあるメソッドは静的メソッドである必要があり、void 型のみを返します。

クラスのメソッドを非同期に実行するには、

アノテーションのあるメソッドを定義します。次に例を示します。

次のスニペットでは、メソッドがコールアウトを実行するように指定する方法を示します。

メソッドがコールアウトを実行しないようにするには、

を指定します。

アノテーションのあるメソッドをテストするには、  
コードブロック内でメソッド  
を含むクラスをコールします。  
メソッドの後に作成されたすべての非同期コールはシステムによって  
収集されます。  
を実行する場合、すべての非同期プロセスが同期して実行されます。

アノテーションのあるメソッドには次のような制限事項があります。

- Apex呼び出しごとの、メソッドのコール数は 10 以下にする必要があります。  
 メモ: ブロックおよび ブロックでコールされた または などの非同期コールは、キュー内ジョブ数の制限に対してカウントされません。
- 24 時間あたりの メソッドの最大呼び出し数は、250,000 または組織のユーザライセンス数の 200 倍のいずれか大きいほうです。これは組織全体の制限で、他のすべての非同期 Apex (Apex の一括処理およびスケジュール済み Apex) と共有されます。この制限のカウント対象となるライセンスは、Salesforce フルユーザライセンスおよび Force.com アプリケーションサブスクリプションのユーザライセンスです。Chatter 限定ユーザ、Chatter カスタマーユーザ、カスタマー ポータル ユーザ、およびパートナー ポータル ユーザライセンスは、含まれません。
- 指定するパラメータはプリミティブデータ型、プリミティブデータ型の配列、プリミティブデータ型のコレクションである必要があります。
- アノテーションのあるメソッドは、sObject またはオブジェクトを引数として取ることはできません。
- アノテーションのあるメソッドは、Visualforce コントローラの *MethodName* または *MethodName* メソッド内でも、コンストラクタ内でも使用できません。

アノテーションを使用するすべてのメソッドは、メソッドがコールされた順番に実行されるとは限らないため、特別な考慮が必要です。

アノテーションのあるメソッドを、同じくアノテーションのあるメソッドからコールすることはできません。また、アノテーションのある別のメソッドをコールするアノテーションのあるメソッドからトリガをコールすることはできません。

アノテーションのあるメソッドでは、  
は使用できません。  
および  
メソッド

コールアウトの詳細は、「[Apex を使用したコールアウトの呼び出し](#)」(ページ 372)を参照してください。

## 関連リンク

[実行ガバナと制限について](#)

## IsTest アノテーション

アプリケーションのテストに使用するコードのみを含むクラスまたは個別のメソッドを定義するにはアノテーションを使用します。アノテーションは、として宣言するメソッドの作成と似ています。

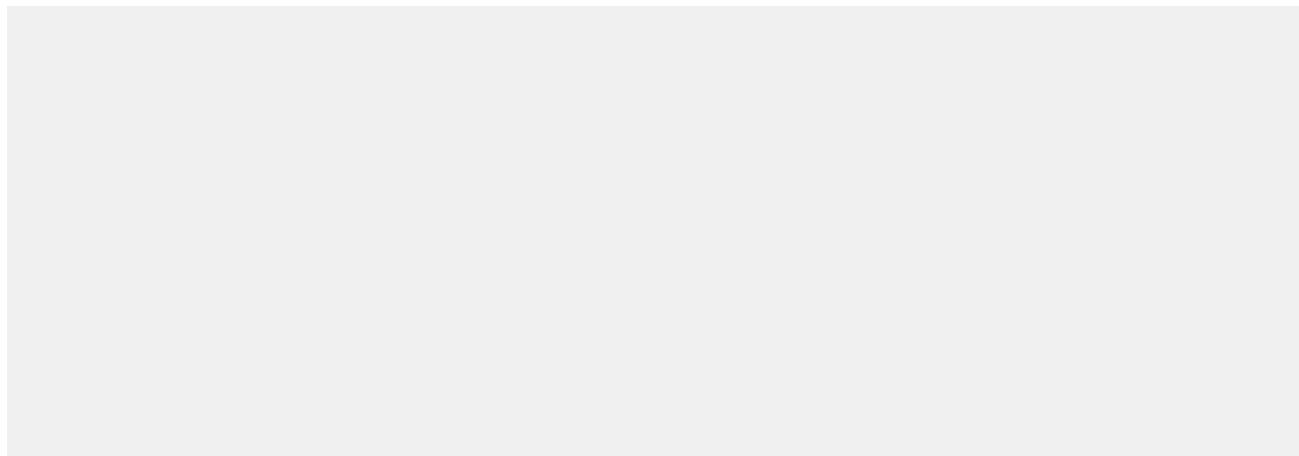


メモ: アノテーションで指定したクラスは、Apex コードの組織内の上限の 3 MB には含まれません。

Salesforce.com API バージョン 24.0 以降を使用して保存した Apex コードでは、デフォルトで、テストメソッドは組織の既存のデータにアクセスできません。ただし、Salesforce.com API バージョン 23.0 以前で保存されたテストコードは、引き続き、組織のすべてのデータにアクセスでき、そのデータアクセス権は変わりません。「[単体テストの組織データとテストデータの分離](#)」(ページ 233)を参照してください。

として定義されたクラスとメソッドはまたはのいずれかと宣言する必要があります。  
として定義したクラスは最上位クラスである必要があります。

次に、2 つのテストメソッドを含む非公開テストクラスの例を示します。



次に、テストデータ作成のユーティリティメソッドを含む公開テストクラスの例を示します。

として定義されたクラスは、インターフェースまたは enum 値とすることはできません。

公開テストクラスのメソッドは、実行中のテスト、つまり、テストメソッドまたはテストメソッドから呼び出されるコードからのみ呼び出すことができ、テスト以外の要求から呼び出すことはできません。また、Salesforce ユーザインターフェースまたは API を使用してテストクラスメソッドを呼び出すことができます。詳細は、「[单体テストメソッドの実行](#)」を参照してください。

`IsTest(SeeAllData=true)` アノテーション

Salesforce.com API バージョン 24.0 以降を使用して保存された Apex コードでは、テストクラスおよび個々のテストメソッドに対して、テストが作成していない既存のデータを含む組織のすべてのデータへのアクセス権を付与するには、**アノテーション**を使用します。Salesforce.com API バージョン 24.0 以降

を使用して保存した Apex コードでは、デフォルトで、テストメソッドは組織の既存のデータにアクセスできません。ただし、Salesforce.com API バージョン 23.0 以前で保存されたテストコードは、引き続き、組織のすべてのデータにアクセスでき、そのデータアクセス権は変わりません。「[単体テストの組織データとテストデータの分離](#)」(ページ 233)を参照してください。

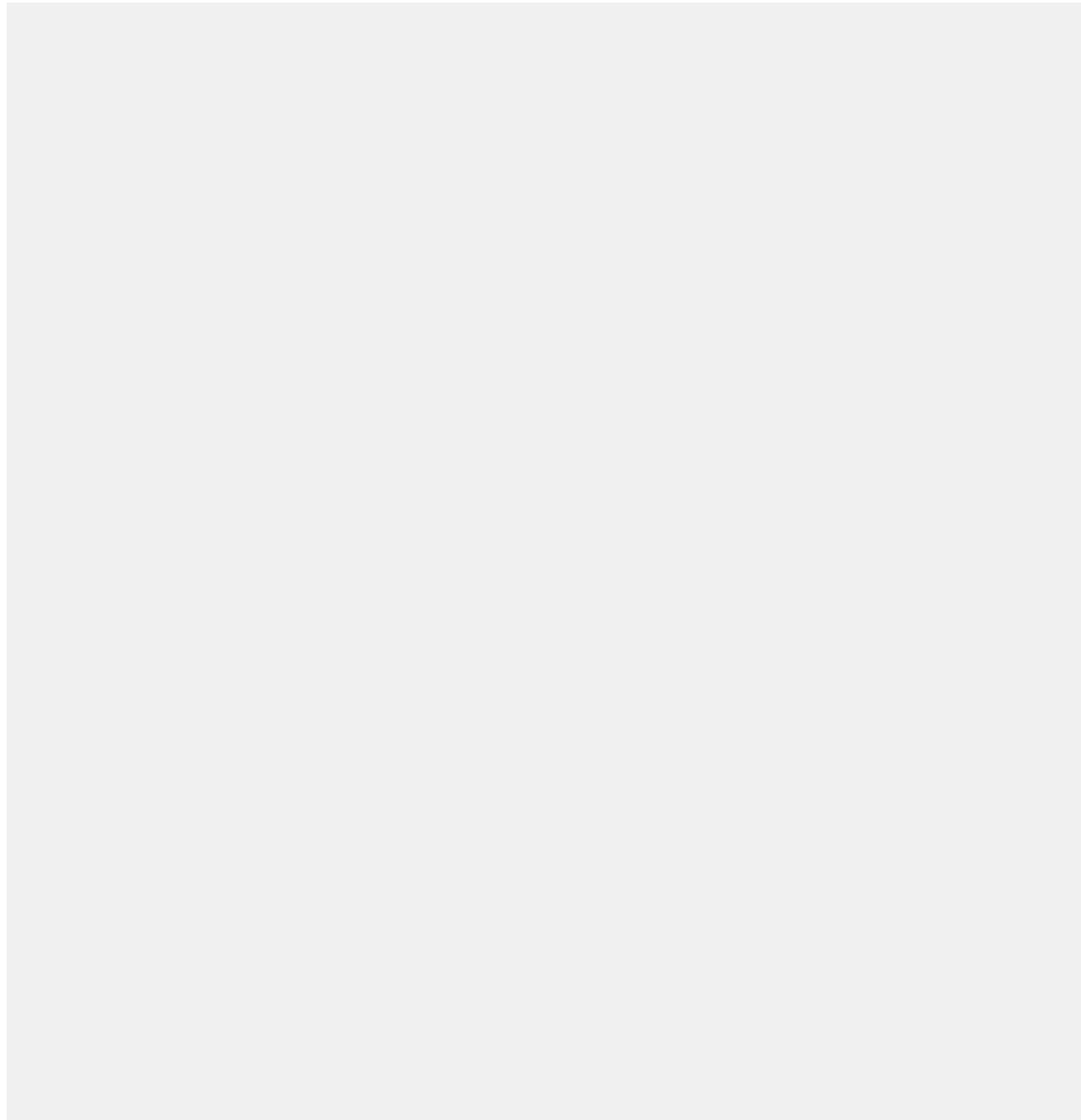
`IsTest(SeeAllData=true)` アノテーションの考慮事項

- テストクラスがアノテーションで定義されている場合、このアノテーションは、テストメソッドがアノテーションとキーワードのどちらを使用して定義されているかにかかわらず、すべてのテストメソッドに適用されます。

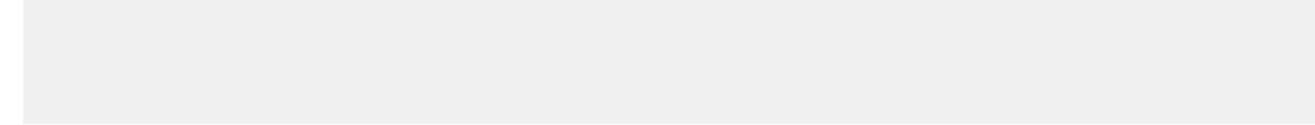
- アノテーションは、クラスまたはメソッドレベルで適用される場合にデータにアクセスできるようにするために使用します。ただし、含まれているクラスがすでにアノテーションで定義されている場合、メソッドでの使用によって、そのメソッドの組織データアクセスが制限されることはありません。この場合、メソッドは組織のすべてのデータにアクセスできます。

この例では、  
このクラスのすべてのテストメソッドは組織のすべてのデータにアクセスできます。

アノテーションを使用してテストクラスを定義する方法を示します。



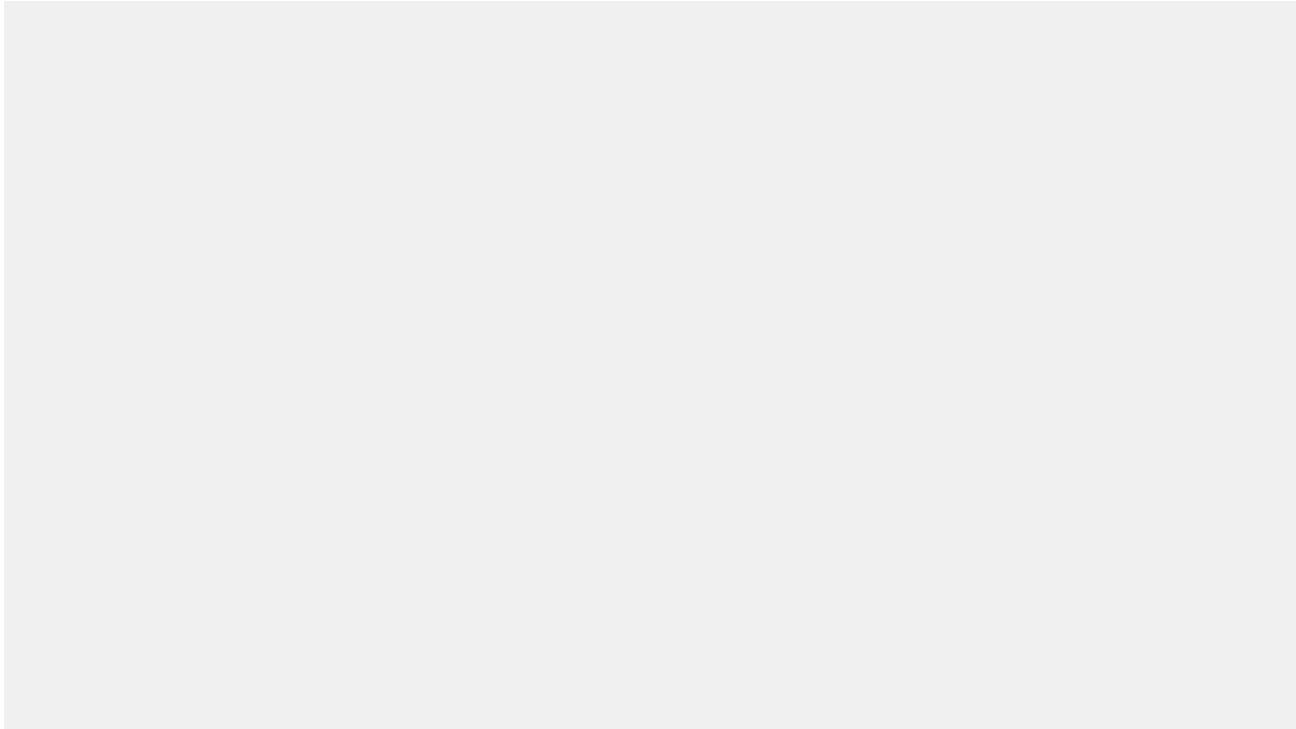
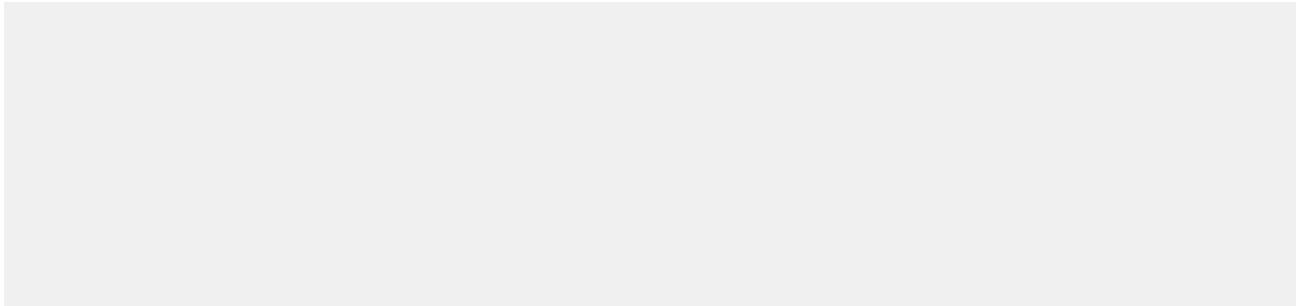
この 2 番目の例では、テストメソッドにアノテーションを適用する方法を示します。このクラスはテストメソッドに含まれているけれども、このアノテーションを使用して定義されていないため、テストメソッドがすべてのデータにアクセスできるようにするには、このアノテーションをテストメソッドに適用する必要があります。2 番目のテストメソッドにはこのアノテーションはありません。そのため、テストメソッドでアクセスできるデータはテストメソッドで作成されたデータに加え、組織の管理に使用するオブジェクトのデータになります。組織の管理に使用するオブジェクトの例としてユーザが挙げられます。

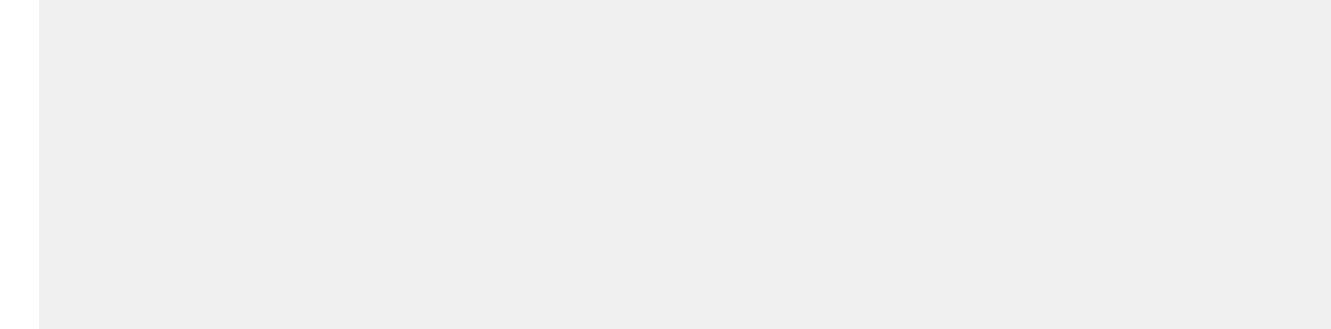


### **IsTest(OnInstall=true) アノテーション**

アノテーションを使用して、パッケージのインストール時に実行される Apex テストを指定します。このアノテーションは、管理パッケージまたは未管理パッケージのテストで使用されます。このアノテーションを付加したテストメソッドまたはこのアノテーションを含むテストクラスの一部であるメソッドのみが、パッケージのインストール時に実行されます。パッケージのインストールが成功するためには、パッケージのインストール時に実行というアノテーション指定されたテストが正常に終了する必要があります。パッケージのインストール時に失敗したテストをバイパスすることはできなくなりました。このアノテーションが付加されていないテストメソッドまたはクラス、または \_\_\_\_\_ または \_\_\_\_\_ でアノテーションされたテストメソッドまたはクラスは、インストール時に実行されません。

次に、パッケージのインストール時に実行されるテストメソッドにアノテーションを付加する方法の例を示します。この例の \_\_\_\_\_ は実行されますが、\_\_\_\_\_ と \_\_\_\_\_ は実行されません。





## ReadOnly アノテーション

アノテーションを使用して、Force.com データベースの無制限のクエリを実行できます。他のすべての制限は適用されます。要求に対して返される行数の制限がなくなる一方、要求内での DML 操作、  
へのコール、アノテーション が付加されたメソッドへのコール、およびメール送信の実行がブロックされるため、このアノテーションには注意する必要があります。

アノテーションは、Web サービスおよび インターフェースで使用可能です。  
アノテーションを使用するには、最上位レベルの要求がスケジュール実行または Web サービスの呼び出し内にある必要があります。たとえば、Visualforce ページが アノテーションを含めて Web サービスを呼び出す場合、Visualforce は Web サービスではなく、最上位レベルの要求であるため、要求は失敗します。

Visualforce ページでは アノテーションを使用してコントローラメソッドをコールすることができます。また、それらのメソッドは同じ制限が緩和された状態で実行されます。などの繰り返しコンポーネントで使用できるコレクションのサイズなどその他の Visualforce 固有の制限を増加するには、  
タグの 属性を に設定できます。詳細は、[Visualforce 開発者ガイド](#)の「大量のデータセットを使用した作業」を参照してください。

## RemoteAction アノテーション

アノテーションでは、Visualforce で使用する Apex メソッドの JavaScript を介したコールのサポートが提供されます。このプロセスは、多くの場合 JavaScript Remoting と呼ばれます。

 メモ: アノテーションのあるメソッドは、かつ または である必要があります。

Visualforce ページで JavaScript Remoting を使用するには、要求を次の形式の JavaScript 呼び出しとして追加します。

```
namespace controller method
```

```
callbackFunction
```

- はコントローラクラスの名前空間です。組織に名前空間が定義されている場合、またはクラスがインストール済みパッケージに基づく場合は必須です。
- は Apex コントローラの名前です。
- はコールする Apex メソッドの名前です。
- はメソッドが取るパラメータのカンマ区切りのリストです。
- はコントローラからの応答を処理する JavaScript 関数の名前です。匿名関数をインラインで宣言することもできます。ではメソッドコールの状況と結果をパラメータとして返します。
- は、リモートコールと応答の処理を設定します。Apex メソッドの応答をエスケープするかどうかを指定するなど、リモートコールの動作を変更する場合にこれを使用します。

コントローラでは、Apex のメソッド宣言は、次のように

アノテーションが先頭に付加されます。

メソッドでは、引数として、Apex プリミティブ、コレクション、型指定された sObject、汎用 sObject、ユーザ定義 Apex クラスおよびインターフェースを取ることができます。汎用 sObject では、実際の型を特定するために ID または sObjectType の値を指定する必要があります。インターフェースパラメータでは、実際の型を特定するために apexType を指定する必要があります。メソッドでは Apex プリミティブ、sObjects、コレクション、ユーザ定義 Apex クラスおよび列挙、  
、  
、  
、  
、または  
を返すことができます。

詳細は、『Visualforce 開発者ガイド』の「Apex コントローラの JavaScript Remoting」を参照してください。

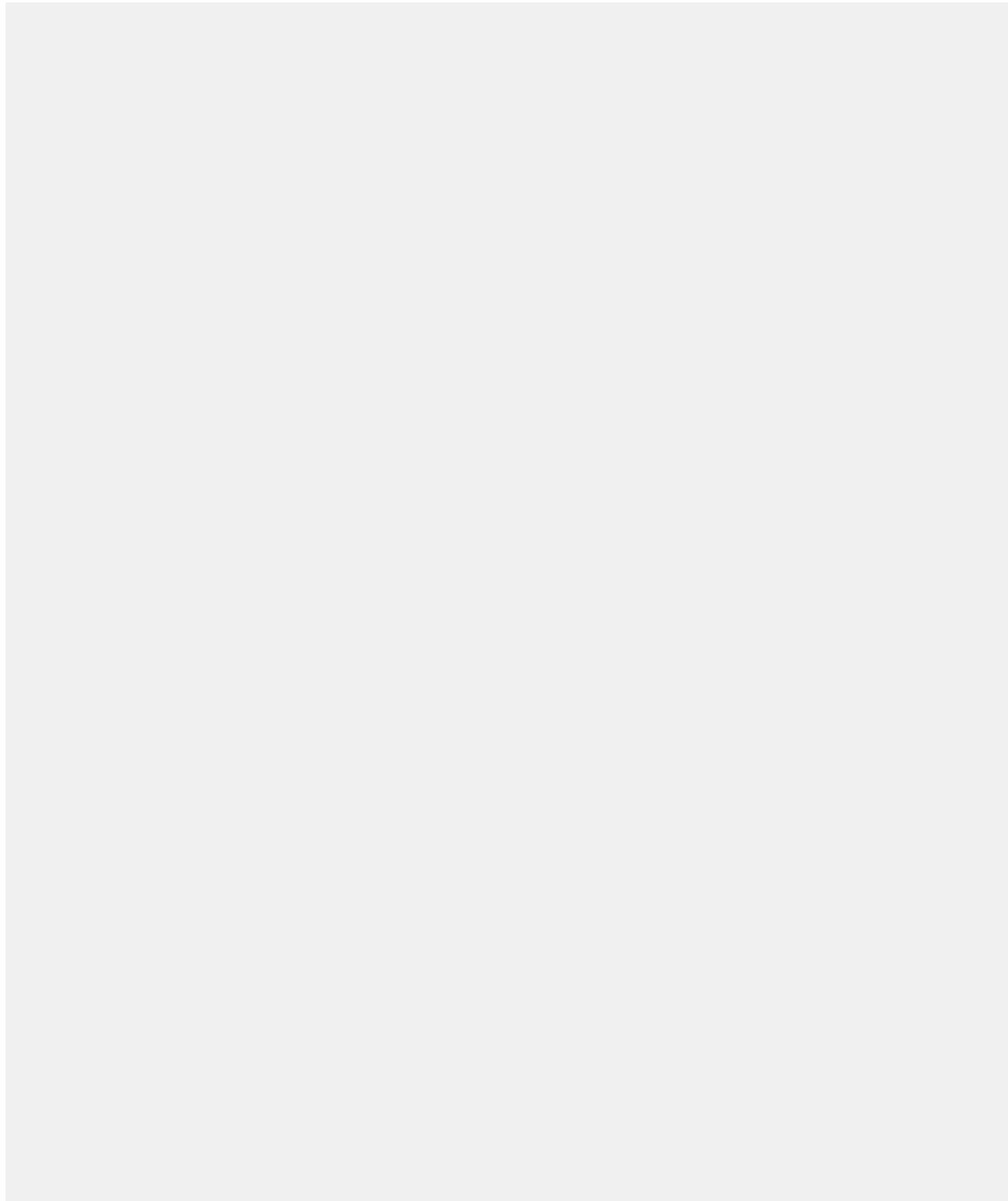
## TestVisible アノテーション

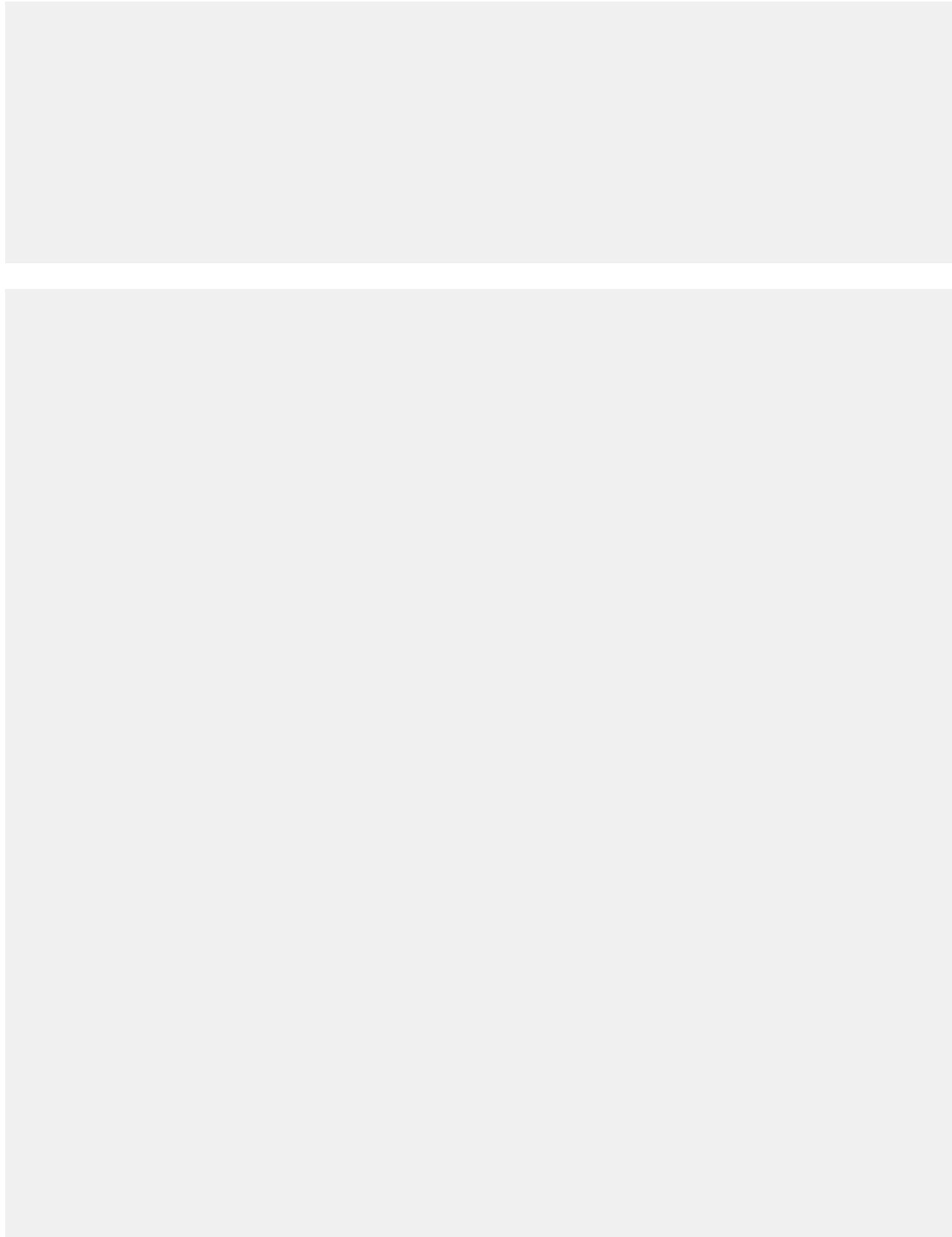
アノテーションを使用すると、テストクラス外にある別のクラスの非公開メンバーまたは保護メンバーにテストメソッドからアクセスできるようになります。これらのメンバーには、メソッド、メンバー変数、内部クラスが含まれます。このアノテーションは、テストを実行する目的でのみ、権限の高いアクセスレベルを有効にします。このアノテーションによって、非テストクラスからアクセスするメンバーの表示が変わることはありません。

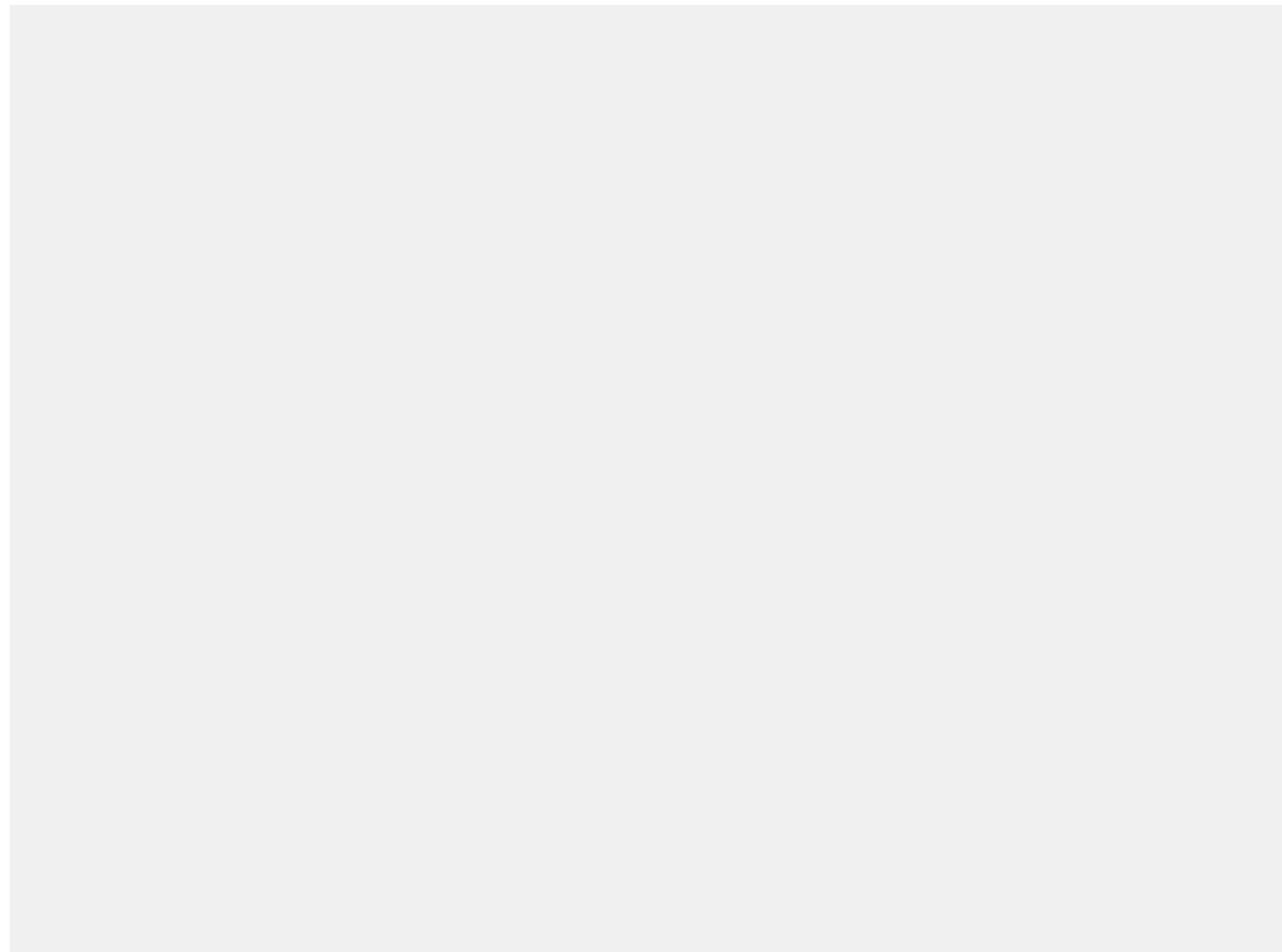
このアノテーションでは、メソッドのアクセス修飾子やメンバー変数にテストメソッドでアクセスする場合に、それらを public に変更する必要はありません。たとえば、外部クラスに対して非公開メンバー変数を表示せずに、テストメソッドからアクセスできるようにする場合は、アノテーションを変数定義に追加します。

また、このアノテーションは、テストコードと非テストコードが混在する既存のクラスの Salesforce.com API バージョンをアップグレードする場合にも便利です。API バージョン 28.0 以降ではテストメソッドが非テストクラスで使用できなくなるため、クラスの API バージョンをアップグレードする場合に、古いクラスから新しいクラス（アノテーションが付加されたクラス）にテストメソッドを移動する必要があります。元のクラスの非公開メソッドまたはメンバー変数にアクセスするときに、表示に関する問題が生じる場合があります。この場合は、これらの非公開メンバーにアノテーションを付加します。

次の例に、非公開メンバー変数、コンストラクタがある非公開内部クラス、非公開メソッド、および非公開カスタム例外での の使用法を示します。 アノテーションが付加されているため、これらはすべてテストクラスでアクセスできます。 クラスに続いて、テストメソッドを含むテストクラスを示します。







## Apex REST アノテーション

6 つの新しいアノテーションが追加され、Apex クラスを RESTful Web サービスとして公開できるようになります。

- *yourUrl*
- 
- 
- 
- 
- 

### 関連リンク

[Apex REST の基本コードサンプル](#)

## RestResource アノテーション

アノテーションはクラスレベルで使用され、Apex クラスを REST リソースとして公開できるようにします。

このアノテーションを使用する場合、次の点に留意してください。

- URL 対応付けは、*instance* と相対的です。
- ワイルドカード文字 (\*) を使用することができます。
- URL の対応付けでは、大文字と小文字は区別されます。 の URL の対応付けでは、ではなく を含む REST リソースのみが一致します。
- このアノテーションを使用するには、Apex クラスがグローバルとして定義されている必要があります。

### URL のガイドライン

URL パスの対応付けは次のようにになります。

- パスは '/' で開始する必要があります。
- '\*' が出現したら、その前に '/'、後に '/' を付ける必要があります。ただし、'\*' が末尾の文字である場合は後続の '/' は不要です。

URL 対応付けのルールは次のようにになります。

- 常に完全一致が優先されます。
- 完全一致がない場合、ワイルドカードを使用して一致するすべてのパターンを検索し、そのうち文字列の長さが最も長いものが選択されます。
- ワイルドカード一致が見つからない場合、HTTP 応答状況コード 404 が返されます。

名前空間にあるクラスの URL には名前空間が含まれます。たとえば、クラスが名前空間 *instance* 内にあり、クラスが *instance* に対応付けられている場合、API URL は

のように変更されます。URL が

競合する場合、名前空間にあるクラスが常に使用されます。

## HttpDelete アノテーション

アノテーションはメソッドレベルで使用され、Apex メソッドを REST リソースとして公開できるようにします。このメソッドは、HTTP DELETE 要求が送信されるとコールされ、指定されたリソースを削除します。

このアノテーションを使用するには、Apex メソッドがグローバルに静的として定義されている必要があります。

## HttpGet アノテーション

アノテーションはメソッドレベルで使用され、Apex メソッドを REST リソースとして公開できるようにします。このメソッドは、HTTP GET 要求が送信されるとコールされ、指定されたリソースを返します。

このアノテーションを使用する場合、次の点に留意してください。

- このアノテーションを使用するには、Apex メソッドがグローバルに静的として定義されている必要があります。

- HTTP 要求が 要求メソッドを使用する場合、 アノテーションが付加されたメソッドもコールされます。

### HttpPatch アノテーション

アノテーションはメソッドレベルで使用され、Apex メソッドを REST リソースとして公開できるようにします。このメソッドは、HTTP 要求が送信されるとコールされ、指定されたリソースを更新します。

このアノテーションを使用するには、Apex メソッドがグローバルに静的として定義されている必要があります。

### HttpPost アノテーション

アノテーションはメソッドレベルで使用され、Apex メソッドを REST リソースとして公開できるようにします。このメソッドは、HTTP 要求が送信されるとコールされ、新しいリソースを作成します。

このアノテーションを使用するには、Apex メソッドがグローバルに静的として定義されている必要があります。

### HttpPut アノテーション

アノテーションはメソッドレベルで使用され、Apex メソッドを REST リソースとして公開できるようにします。このメソッドは、HTTP 要求が送信されるとコールされ、指定されたリソースを作成または更新します。

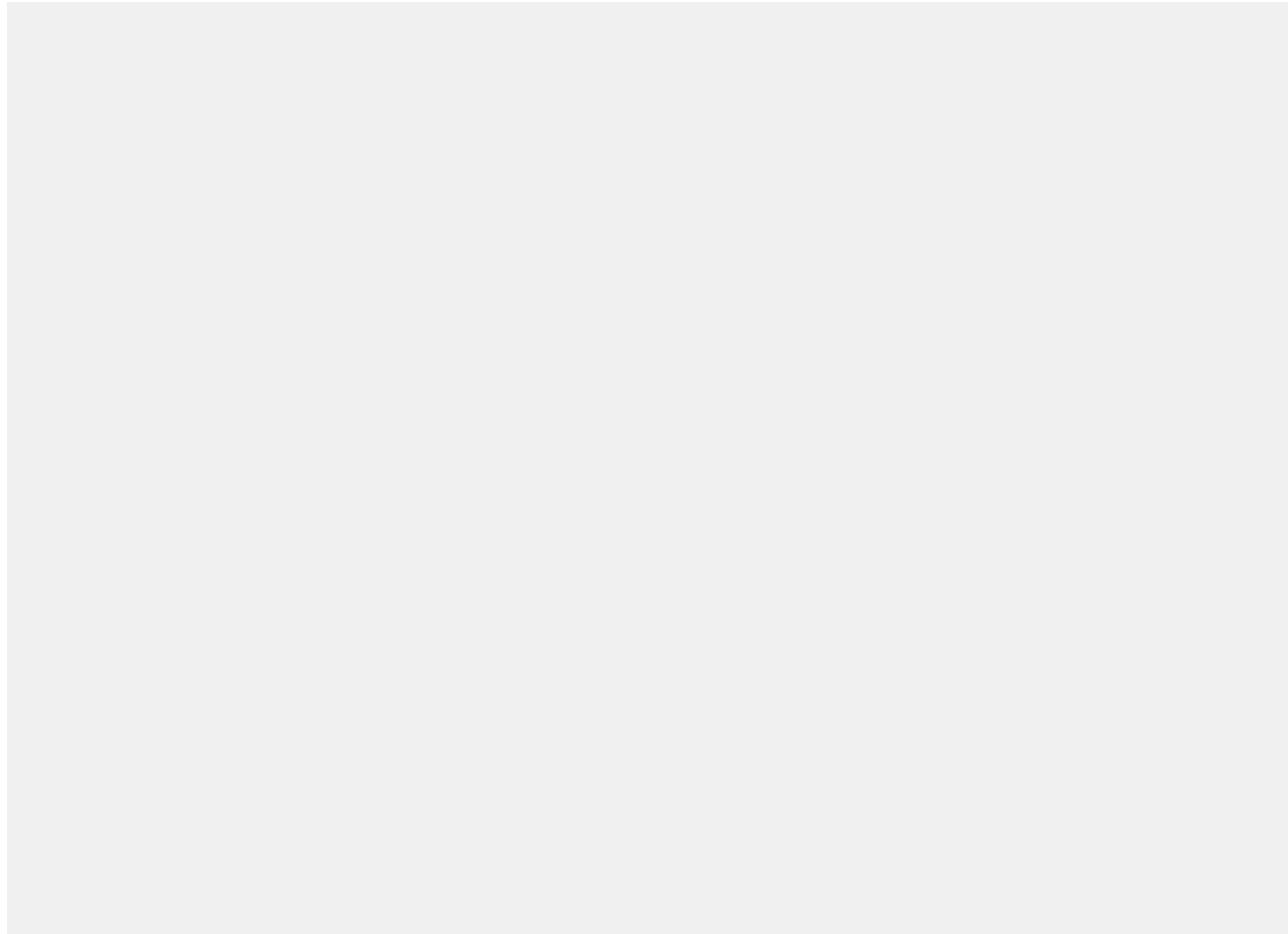
このアノテーションを使用するには、Apex メソッドがグローバルに静的として定義されている必要があります。

## クラスとキャスト

通常、すべての型情報は実行時に利用できます。つまり、Apex はキャストを許可しています。キャストとは、あるクラスのデータ型を別のクラスのデータ型として割り当てることです。ただし、割り当てるクラスが元のクラスの子である場合に限ります。あるデータ型のオブジェクトを別のデータ型に変換する場合にキャストを使用します。

次の例では、 が クラスを拡張しています。そのため、そのクラスの子となっています。つまり、親のデータ型( )のオブジェクトを、子のデータ型( )のオブジェクトにキャストできます。

次のコードブロックでは、まずレポートオブジェクトのリストにカスタムレポートオブジェクトが追加されます。その後、カスタムレポートオブジェクトがレポートオブジェクトとして返され、カスタムレポートオブジェクトとして再度キャストされます。



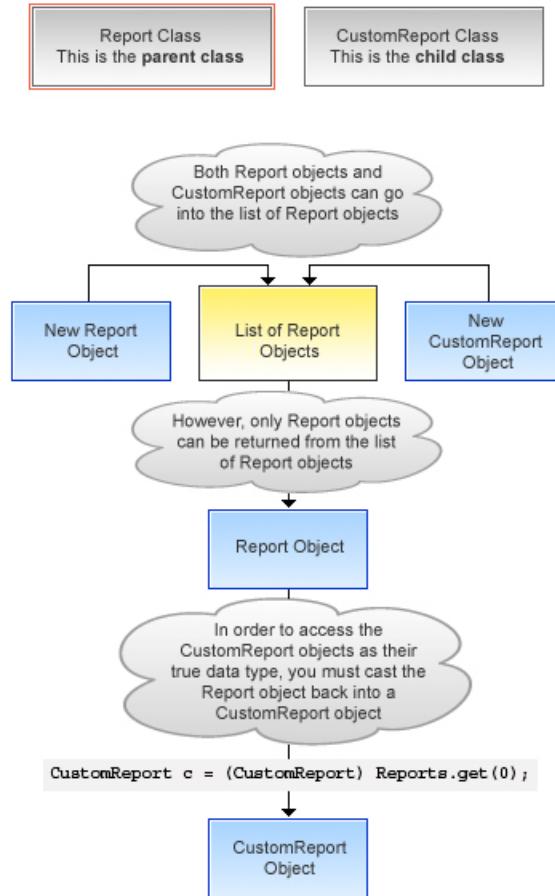


図 4: キャストの例

さらに、インターフェース型は、サブインターフェースまたはそのインターフェースを実装しているクラス型にキャストできます。



ヒント: あるクラスが特定の型のクラスであることを確認するには、キーワードを使用します。詳細は、「[キーワードの使用](#)」(ページ 191)を参照してください。

## クラスとコレクション

リストと対応付けは、`sObjects`で使用するのと同じように、クラスやインターフェースでも使用できます。つまり、ユーザ定義のデータ型は対応付けの値のみで使用でき、キーでは使用できません。同様に、ユーザ定義オブジェクトセットは作成できません。

インターフェースの対応付けやリストを作成する場合、インターフェースの子の型をそのコレクションに入れるすることができます。たとえば、リストにインターフェース `i1` が含まれており、`MyC` が `i1` を実装している場合、`MyC` をリストに含めることができます。

## コレクションキャスト

Apex のコレクションには実行時に宣言される型が存在するため、Apex ではコレクションキャストを許可しています。

コレクションは、Java で配列をキャストするのと似た方法でキャストされます。たとえば、  
    クラスが                          クラスの子である場合、CustomerPurchaseOrder オブジェクト  
    トのリストを PurchaseOrder オブジェクトのリストに割り当てることができます。

リストがリスト変数に割り当てられると、そのインスタンスが最初は CustomerPurchaseOrder のリストとしてインスタンス化されるため、CustomerPurchaseOrder オブジェクトのリストに再度キャストすることができます。このようにインスタンス化された PurchaseOrder オブジェクトのリストは、PurchaseOrder オブジェクトのリストに CustomerPurchaseOrder オブジェクトのみが含まれている場合でも、CustomerPurchaseOrder オブジェクトのリストにキャストできません。

CustomerPurchaseOrders オブジェクトのみを含む PurchaseOrder リストのユーザが の非 CustomerPurchaseOrder サブクラス ( など) を挿入しようとすると、実行時例外が発生します。これは、Apex のコレクションには実行時に宣言される型が存在するためです。



メモ: 対応付けは対応付けの値側に関してリストと同じ方法で動作します。対応付け A の値側を対応付け B の値側にキャストし、これらの対応付けが同じキータイプである場合、対応付け A を対応付け B にキャストできます。実行時に特定の対応付けでキャスティングが無効な場合は、ランタイムエラーとなります。

## Apex クラスと Java クラスの違い

次のリストに Apex クラスと Java クラスの主な違いを示します。

- 内部クラスとインターフェースは、外部クラスの 1 つ下のレベルでのみ宣言できます。
  - 静的メソッドと変数は、内部クラスではなく最上位クラスでのみ宣言できます。

- 内部クラスは、Java の静的な内部クラスのように機能しますが、**private** キーワードを要求しません。内部クラスは、外部クラスのようにインスタンスメンバー変数を持つことができますが、( **private** キーワードを使った) 外部クラスのインスタンスへの暗黙的ポインタはありません。
- デフォルトのアクセス修飾子は **public** です。つまり、メソッドまたは変数は、定義された Apex クラス内からのみアクセス可能です。アクセス修飾子を指定しない場合、メソッドや変数は **private** となります。
- メソッドまたは変数にアクセス修飾子を指定しない場合は、**private** アクセス修飾子を指定した場合と同じ意味となります。
- アクセス修飾子は、メソッドまたは変数がこのアプリケーションまたは名前空間内のすべての Apex で使用可能なことを意味します。
- アクセス修飾子は、メソッドまたは変数が、同じアプリケーション内の Apex コードだけではなく、クラスへのアクセス権を付与されたすべての Apex コードで使用可能なことを意味します。アプリケーション外 (SOAP API 内、または別の Apex コード) から参照されるすべてのメソッドはこのアクセス修飾子を使用する必要があります。メソッドまたは変数を **private** として宣言する場合、それを含むクラスも **private** として宣言する必要があります。
- メソッドおよびクラスはデフォルトで **final** です。
  - ◊ 定義修飾子は、拡張や上書きを許可します。
  - ◊ キーワードは、基本クラスメソッドを上書きするメソッドで明示的に使用する必要があります。
- インターフェースメソッドには修飾子はなく、常に **global** となります。
- 例外クラスは、例外または別のユーザ定義例外への拡張が必要です。
  - ◊ 例外クラス名の末尾には、**Exception** をつける必要があります。
  - ◊ 例外クラスは 4 つの暗黙的なコンストラクタが組み込まれていますが、追加することもできます。
- 詳細は、「[例外クラス](#)」(ページ 785)を参照してください。
- クラスとインターフェースはトリガや匿名ブロック内で定義できますが、ローカルとしてのみ定義できます。

## クラス定義の作成

Salesforce でクラスを作成する手順は、次のとおりです。

1. [設定] で、[開発] > [Apex クラス] をクリックします。
2. [新規] をクリックします。
3. [バージョン設定] をクリックして、このクラスで使用する Apex および API のバージョンを指定します。組織が AppExchange から管理パッケージをインストールした場合、このクラスで使用する各管理パッケージのバージョンも指定できます。すべてのバージョンでデフォルト値を使用します。デフォルト値では、Apex および API についても、各管理パッケージについても、クラスを最新バージョンに関連付けます。最新バージョンのパッケージのものとは異なるコンポーネントや機能にアクセスする場合は、管理パッケージの古いバージョンを指定することもできます。特定の動作を維持するには、Apex および API の古いバージョンを指定できます。
4. クラスエディタで、クラスの Apex コードを入力します。1 つのクラスの長さは、最大 1,000,000 文字です。を使用して定義したコメント、テストメソッド、またはクラスは含みません。

5. [保存] をクリックし、変更を保存してクラスの詳細画面に戻るか、[適用] をクリックし、変更を保存してクラスの編集を続行します。作成した Apex クラスは、クラスに保存する前に正しくコンパイルする必要があります。

[WSDL からの生成] をクリックして、WSDL から自動的にクラスを生成することもできます。『[SOAP サービス: WSDL ドキュメントからのクラスの定義](#)』(ページ 373)を参照してください。

いったん保存されると、クラスはトリガなど別の Apex コードからクラスメソッドや変数を介して呼び出すことができます。



メモ: 下位互換性を持たせるため、クラスは、Apex および API の特定のバージョンのバージョン設定と共に保存されます。Apex クラスが、インストール済みの管理パッケージ内で、カスタムオブジェクトなどのコンポーネントを参照する場合、クラスが参照する各管理パッケージのバージョン設定も同時に保存されます。また、クラスは、最後にコンパイルされて以降、依存するメタデータに変更がない限り、フラグを\_\_\_\_\_に設定して保存されます。オブジェクトや項目の説明の編集などの表面的な変更も含め、クラスで使用されているオブジェクト名や項目に変更があった場合、またはこのクラスを呼び出すクラスに変更があった場合には、\_\_\_\_\_ フラグが\_\_\_\_\_に設定されます。トリガまたは Web サービスコールによってクラスが呼び出されると、コードが再コンパイルされ、エラーが存在する場合にはユーザに通知されます。エラーがない場合は、\_\_\_\_\_ フラグが\_\_\_\_\_にリセットされます。

## Apex クラスエディタ

Visualforce または Apex を編集するとき、Visualforce 開発モードのフッターまたは設定のいずれかで、エディタを使用できます。エディタの機能は、次のとおりです。

### 構文の強調表示

エディタは、キーワードとすべての関数および演算子について、自動的に構文を強調表示します。

### 検索 (🔍)

検索により、現在のページ、クラス、またはトリガの中のテキストを検索できます。検索を使用するには、検索 テキストボックスに文字列を入力し、[次を検索] をクリックします。

- 検出した検索文字列を他の文字列で置き換えるには、置換 テキストボックスに新しい文字列を入力し、そのインスタンスだけを置き換える場合は [replace] をクリックし、そのインスタンスと、それ以外にそのページ、クラス、またはトリガに出現する検索文字列のすべてのインスタンスを置き換える場合は、[すべて置換] をクリックします。
- 検索操作で大文字と小文字を区別するには、[大文字と小文字を区別する] オプションをオンにします。
- 検索文字列として正規表現を使用するには、[正規表現] オプションをオンにします。正規表現は、JavaScript の正規表現規則に従います。正規表現を使った検索では、折り返されて複数行になる文字列も検索できます。

正規表現で検出した文字列を置換操作で使用する場合、検出した検索文字列から得られる正規表現のグループ変数 (\_\_\_\_\_ など) をバインドすることもできます。たとえば、\_\_\_\_\_ タグを\_\_\_\_\_ タグで置き換え、元の\_\_\_\_\_ の属性はすべてそのままにするには、\_\_\_\_\_ を検索し、それを\_\_\_\_\_ で置き換えます。

### 指定行に移動(→)

このボタンにより、指定した行番号を強調表示できます。その行が現在表示されていない場合は、エディタがその行までスクロールします。

### 元に戻す(⬅)およびやり直し(➡)

[Undo (元に戻す)] を使用すると編集動作を取り消します。[Redo (やり直し)] を使用すると元に戻した編集動作をやり直します。

### フォントサイズ

ドロップダウンリストからフォントサイズを選択し、エディタに表示される文字のサイズを制御します。

### 行と列の位置

カーソルの行と列の位置は、エディタ下部のステータスバーに表示されます。これは、[Go To Line(指定行に移動)] (→)と共に使用し、エディタ内をすばやく移動できます。

### 行と文字の計数

行と文字の合計数は、エディタ下部のステータスバーに表示されます。

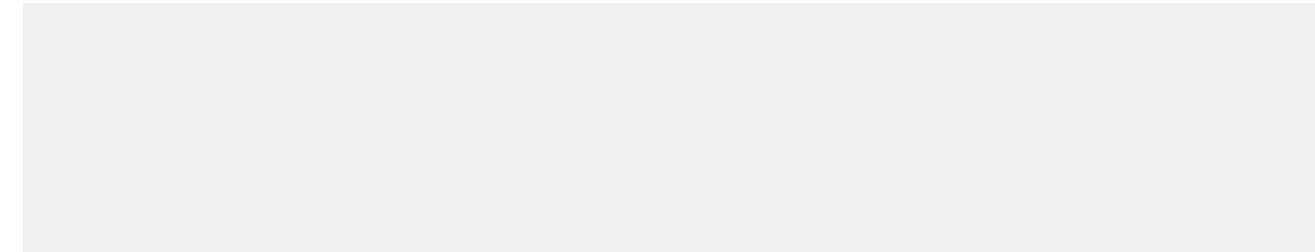
## 名前付け規則

名前付けに次のJava標準を推奨しています。クラス名は大文字から始め、メソッドは小文字の動詞から始め、変数名は意味のあるものにします。

同じクラス内で、クラスとインターフェースに同じ名前を付けることはできません。また、外部クラスと内部クラスに同じ名前を付けることはできません。ただし、メソッドと変数はクラス内に独自の名前空間があるため、この3種類の名前は競合しません。特に、クラス内の変数、メソッド、クラスに同じ名前を付けることは許されます。

## 名前のシャドウイング

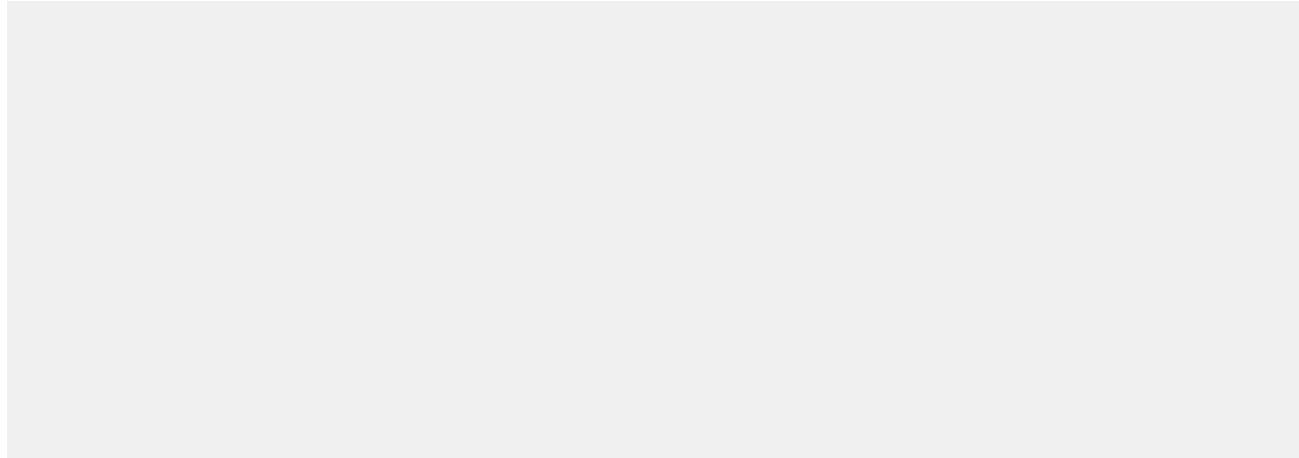
メンバー変数は、特に関数の引数でローカル変数によりシャドウイングできます。これにより、標準Java形式のメソッドやコンストラクタは次のように処理されます。



1つのクラスのメンバー変数は、親クラスの同じ名前のメンバー変数をシャドウイングできます。これは、2つのクラスが異なる最上位クラスにあり、異なるチームによって記述されている場合に有用です。たとえば、一方にはクラス C への参照が含まれており、親クラス P のメンバー変数 M (C のメンバー変数と同じ名前) ヘアクセスするとします。参照は、まず P への参照から割り当てます。

静的変数はクラス階層全体でシャドウイングできます。そのため、P では静的 S を定義し、サブクラス C は静的 S を宣言することもできます。C 内の S への参照は、その静的変数を参照します。P 内の S を参照するには、構文 P.S を使用する必要があります。

静的クラス変数は、クラスインスタンスを介して参照することはできません。本来の変数名自体(最上位クラスのファイル内)またはクラス名をつけたプレフィックスを使用して参照する必要があります。次に例を示します。



## クラスのセキュリティ

ユーザプロファイルまたは権限セットに基づいて、特定の最上位クラスでメソッドを実行できるユーザを指定できます。セキュリティを設定できるのは、Apex クラスのみです。トリガには設定できません。

クラス一覧ページから Apex クラスのセキュリティを設定する手順は、次のとおりです。

1. [設定] で、[開発] > [Apex クラス] をクリックします。
2. 制限するクラス名の横にある [セキュリティ] をクリックします。
3. [選択可能なプロファイル] リストから有効にするプロファイルを選択して [追加] をクリックするか、[有効にされたプロファイル] リストから無効にするプロファイルを選択して [削除] をクリックします。
4. [保存] をクリックします。

クラスの詳細ページから Apex クラスのセキュリティを設定する手順は、次のとおりです。

1. [設定] で、[開発] > [Apex クラス] をクリックします。
2. 制限するクラス名をクリックします。
3. [セキュリティ] をクリックします。
4. [選択可能なプロファイル] リストから有効にするプロファイルを選択して [追加] をクリックするか、[有効にされたプロファイル] リストから無効にするプロファイルを選択して [削除] をクリックします。
5. [保存] をクリックします。

Apex クラスのセキュリティを権限セットから設定する手順は、次のとおりです。

1. [設定] で、[ユーザの管理] > [権限セット] をクリックします。

2. 権限セットを選択します。
  3. [Apex クラスアクセス] をクリックします。
  4. [編集] をクリックします。
  5. [利用可能な Apex クラス] リストから有効にする Apex クラスを選択し、[追加] をクリックするか、[有効な Apex クラス] リストから無効にする Apex クラスを選択し、[削除] をクリックします。
  6. [保存] をクリックします。

Apex クラスのセキュリティをプロファイルから設定する手順は、次のとおりです。

1. [設定] で、[ユーザの管理] > [プロファイル] をクリックします。
  2. プロファイルを選択します。
  3. [Apex クラスのアクセス] ページまたは関連リストで、[編集] をクリックします。
  4. [利用可能な Apex クラス] リストから有効にする Apex クラスを選択し、[追加] をクリックするか、[有効な Apex クラス] リストから無効にする Apex クラスを選択し、[削除] をクリックします。
  5. [保存] をクリックします。

## オブジェクト権限と項目権限の適用

Apex は一般に、システムコンテキストで実行されます。つまり、コード実行時に、現在のユーザの権限、項目レベルセキュリティ、および共有ルールは考慮されません。このルールの唯一の例外は、

コールおよび Chatter in Apex と共に実行される Apex コードです。 は常に、現在のユーザの  
フル権限を用いて実行されます。 の詳細は、「匿名プロック」(ページ 158)を参照してくだ  
さい。

Apex は、デフォルトでは、オブジェクトレベルおよび項目レベルの権限を適用しませんが、現在のユーザのアクセス権限レベルを確認する ([Schema.DescribeSObjectResult](#)) `sObject describe result` メソッドおよび ([Schema.DescribeFieldResult](#)) `field describe result` メソッドを明示的にコールすることにより、コードでこれらの権限を適用できます。この方法では、現在のユーザに必要な権限があるかどうかを確認し、ユーザに十分な権限がある場合に限り、特定の DML 操作またはクエリを実行できます。

たとえば、メソッドまたは  
メソッドを呼び出すことにより、現在のユーザに sObject に対する参照、作成または更新のアクセス権があるかどうかをそれぞれ確認できます。同様に、では、現在のユーザの項目に対する  
参照、作成または更新アクセス権を確認するためにコールできるこれらのアクセス制御メソッドを公開します。  
また、メソッドをコールすることにより、現在の  
ユーザに特定の sObject を削除する権限があるかどうかを確認できます。

次に、アクセス制御メソッドをコールする方法の例を示します。

取引先責任者のメール項目を更新する前にこの項目の項目レベルの更新権限を確認する

取引先責任者を新規作成する前に取引先責任者のメール項目の項目レベルの作成権限を確認する



取引先責任者のメール項目をクエリする前に、この項目の項目レベルの参照権限を確認する



取引先責任者を削除する前に、取引先責任者のオブジェクトレベルの権限を確認する



共有ルールはオブジェクトレベルの権限および項目レベルの権限とは異なります。これら両方を設定することができます。共有ルールが Salesforce で定義されている場合、[キーワードを使用してクラスを宣言すること](#)により、クラスレベルで共有ルールを適用できます。詳細は、[「キーワードの使用」](#)を参照してください。sObject describe result および field describe result アクセス制御メソッドをコールする場合、オブジェクトおよび項目レベルの権限の確認は、有効な共有ルールに追加して実行されます。共有ルールにより付与されるアクセスレベルがオブジェクトレベルの権限または項目レベルの権限と競合する場合があります。

## 名前空間プレフィックス

Salesforce アプリケーションは、名前空間プレフィックスの使用をサポートしています。名前空間プレフィックスは管理対象の Force.com AppExchange パッケージで、カスタムオブジェクトと項目名を他の組織で使用されているものと区別するために使用します。開発者がグローバルで一意な名前空間プレフィックスを登録し、AppExchange レジストリに登録すると、開発者の管理パッケージのカスタムオブジェクトおよび項目名への外部参照は次のような長い形式となります。

```
namespace_prefix obj_or_field_name
```

この完全修飾名は、SOQL ステートメント、SOSL ステートメント、Apex でクラスが「管理済み」に設定されると更新が煩雑であるため、Apex はスキーマ名のデフォルトの名前空間をサポートしています。パーサーは ID を確認して、現在のオブジェクトの名前空間を考慮し、特に指定されていない限り、他のすべてのオブジェクトと項目の名前空間であると判断します。その結果、格納されているクラスは、同じアプリケーション名前空間で定義されているオブジェクトに対して、`obj_or_field_name` を使用してカスタムオブジェクトと項目名を直接参照します。



ヒント: AppExchange から組織にインストールされた管理パッケージのカスタムオブジェクトと項目を参照する場合のみ、名前空間プレフィックスを使用します。

## パッケージメソッドの起動での名前空間の使用

管理パッケージで定義されたメソッドを起動するため、Apex では次の形式の完全修飾識別子が許可されています。

```
namespace_prefix class method args
```

## System 名前空間の使用

名前空間は、Apex のデフォルトの名前空間です。つまり、システムクラスの新しいインスタンスを作成するときやシステムメソッドをコールするときに、この名前空間を除外できます。たとえば、組み込みの URL クラスが名前空間にあるため、クラスのインスタンスを作成する次の 2 つのステートメントは同等です。

および:

同様に、次のどちらを記述しても クラスの静的メソッドをコールできます。

または

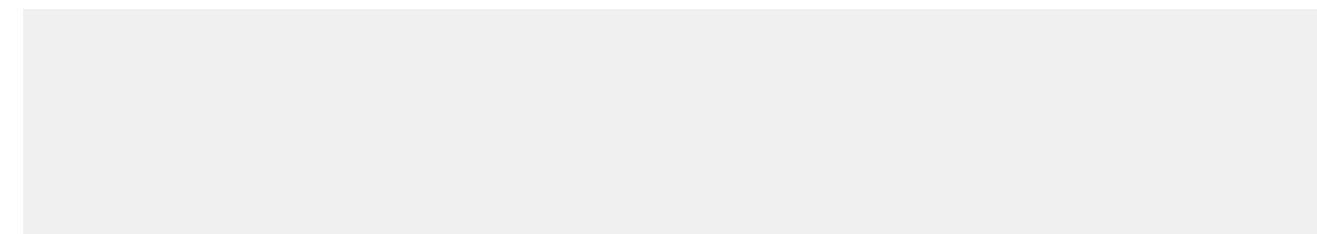


メモ: 名前空間に加え、名前空間には組み込みのクラスがあり、  
や のようなメソッドを提供します。この場合、名前空間とクラスが同じ名前なので混同しないで  
ください。  
トメントは同等になります。

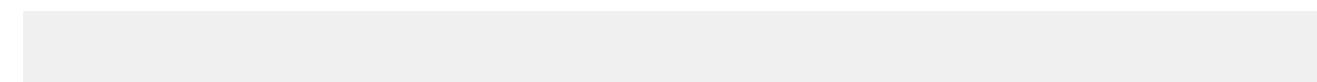
## 曖昧さ回避のための System 名前空間の使用

システムクラスの静的メソッドをコールするときは、名前空間を含めない方が簡単ですが、場合によつては、組み込みの Apex クラスと同じ名前を持つカスタムの Apex クラスと区別するために名前空間を含める必要があります。組み込みのクラスと同じ名前で定義した Apex クラスが組織に含まれる場合、Apex ランタイムでは、デフォルトのカスタムクラスを使用し、カスタムクラス内のメソッドをコールします。次の例を見てみましょう。

次のカスタム Apex クラスを作成します。



開発者コンソールでこのステートメントを実行します。



ステートメントが実行されると、Apex はまずカスタム クラスのクエリメソッドを検索します。ただし、このクラスのクエリメソッドはパラメータを取らず、一致するものが見つからないため、エラーが返されます。名前空間では、カスタム クラスが組み込みの クラスより優先されます。この問題を解決するには、名前空間プレフィックスをクラス名に追加して、Apex ランタイムに 名前空間の組み込み Database クラスのクエリメソッドをコールするように明示的に指示します。

```
System.
```

- 式に識別子が 3 つ以上含まれている場合、パーサーは \_\_\_\_\_ がクラス名、\_\_\_\_\_ が \_\_\_\_\_ から \_\_\_\_\_ を項目参照として持つ静的変数名、\_\_\_\_\_ がメソッド呼び出しであると仮定します。

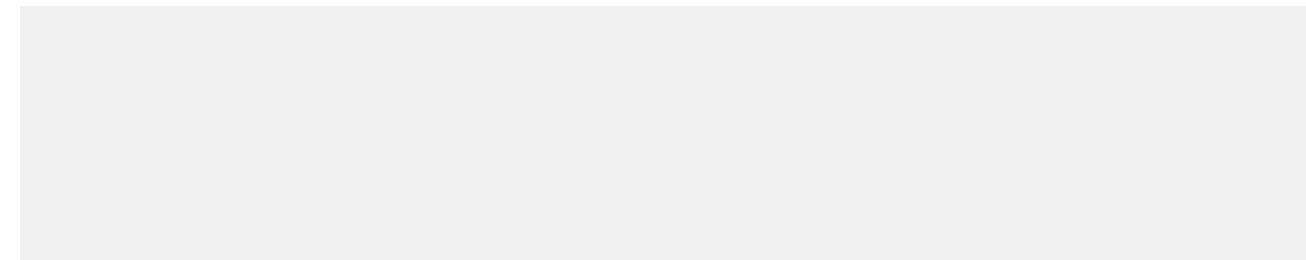
3. 2 つ目の仮定が true でない場合、パーサーは \_\_\_\_\_ が名前空間名、\_\_\_\_\_ がクラス名、\_\_\_\_\_ が静的変数名であり、\_\_\_\_\_ から \_\_\_\_\_ が項目参照、\_\_\_\_\_ がメソッド呼び出しであると仮定します。

4. 3 つ目の仮定も true でない場合は、パーサーはエラーを返します。

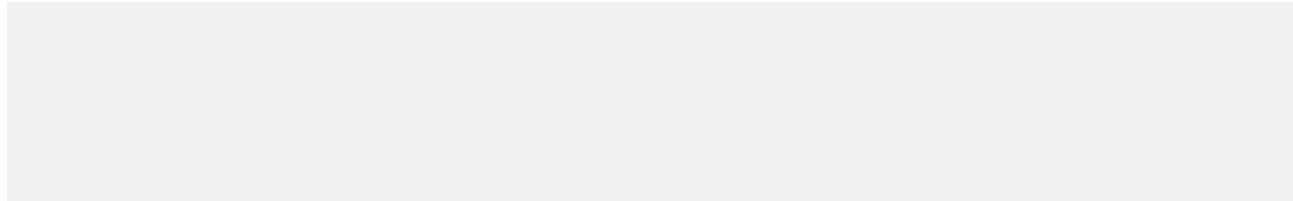
ただし、クラス変数については Apex はメンバー変数の参照にドット表記を使う場合もあります。それらのメンバー変数は他のクラスインスタンスを参照することも、また、項目名への参照(外部キーのアクセスのためなど)に独自のドット表記ルールを持つ sObject を参照することもあります。

式に sObject 項目を入力すると、式の残りは sObject ドメインにとどまります。つまり、sObject 項目は Apex 式を再度参照することはできません。

たとえば、次のクラスがあるとします。



その場合、次の式はすべて有効です。



## 型の解決と型のシステム名前空間

システム型はローカルまたは他のクラスで定義されたユーザ定義型を解決しなければならないため、Apex パーサーは次のように型を評価します。

- 型参照 \_\_\_\_\_ では、パーサーはまずその型をスカラー型として参照します。
- \_\_\_\_\_ が見つからない場合、パーサーはローカルで定義された型を参照します。
- そこでも \_\_\_\_\_ が見つからない場合、パーサーはその名前のクラスを参照します。
- そこでも \_\_\_\_\_ が見つからない場合、パーサーは sObjects などのシステム型を参照します。

型 \_\_\_\_\_ は、最上位クラス \_\_\_\_\_ の内部型 \_\_\_\_\_ 、または名前空間 \_\_\_\_\_ の最上位クラス \_\_\_\_\_ のいずれかを意味します(優先順位はこの順序のとおり)。

## バージョン設定

下位互換性を持たせるため、クラスおよびトリガは、特定の Salesforce.com API バージョンのバージョン設定と共に保存されます。Apex クラスまたはトリガが、インストール済みの管理パッケージ内で、カスタムオブジェクトなどのコンポーネントを参照する場合、クラスが参照する各管理パッケージのバージョン設定も同時に保存されます。Apex、API、および管理パッケージのコンポーネントが次のリリースバージョンにアップグレードされた場合でも、クラスまたはトリガは特定の、既知の動作のバージョンにバインドされたままになります。

インストール済みパッケージのバージョン設定を行うと、インストール済みパッケージの Apex コードの公開されるインターフェースおよび動作が決まります。これにより、コードが廃止される前のバージョンのパッケージをインストールした場合、最新バージョンのインストールパッケージで廃止される場合がある Apex を継続して参照できます。

通常は、最新の Salesforce.com API バージョンおよび各インストール済みパッケージのバージョンを参照します。Salesforce.com API バージョンを指定せずに Apex クラスまたはトリガを保存すると、クラスまたはトリガはデフォルトで最新のインストール済みバージョンと関連付けられます。管理パッケージのバージョンを指定せずに、管理パッケージを参照する Apex クラスまたはトリガを保存する場合、クラスまたはトリガは、デフォルトで、管理パッケージの最新のインストールバージョンに関連付けられます。

## クラスおよびトリガへの Salesforce API バージョン設定

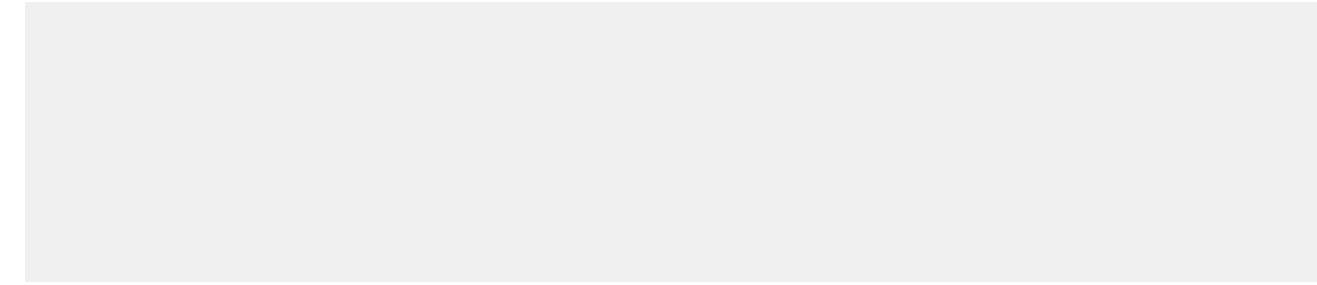
クラスまたはトリガに Salesforce.com API および Apex のバージョンを設定する手順は、次のとおりです。

1. クラスまたはトリガのいずれかを編集して、[バージョン設定] をクリックします。
2. Salesforce.com API のバージョンを選択します。このバージョンは、クラスまたはトリガに関連付けられている Apex のバージョンでもあります。
3. [保存] をクリックします。

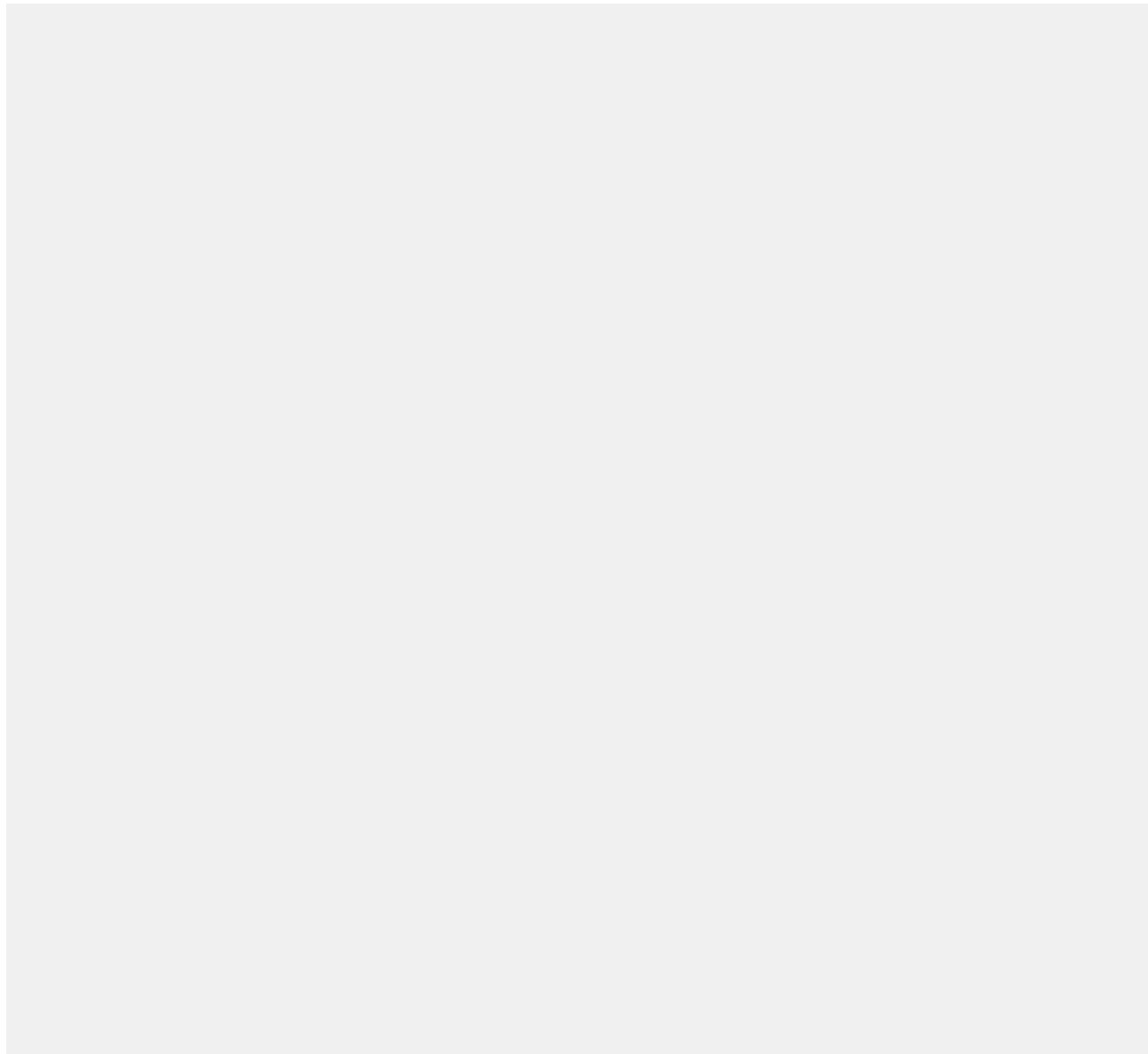
オブジェクトをメソッドコールのパラメータとして Apex クラス C1 から他のクラス C2 に渡し、C2 で Salesforce.com API のバージョン設定により異なる項目が公開されている場合、オブジェクトの項目は C2 のバージョン設定によって制御されます。

次の例では、  
最初のクラスは、Salesforce.com API バージョン 13.0 を使用して保存されています。

項目はバージョン 13.0 の API では使用できないため、テストクラス C1 のメソッドからクラス C2 の  
メソッドをコールした後に 項目が に設定されます。



次のクラスは、Salesforce.com API バージョン 16.0 を使用して保存されています。



## Apex クラスとトリガのパッケージバージョンの設定

クラスまたはトリガのパッケージバージョン設定を定義する手順は、次のとおりです。

1. クラスまたはトリガのいずれかを編集して、[バージョン設定] をクリックします。
2. クラスまたはトリガによって参照される各管理パッケージのバージョンを選択します。管理パッケージのこのバージョンは、より新しいバージョンの管理パッケージがインストールされても、バージョン設定を手動で更新しない限り、クラスまたはトリガによって引き続き使用されます。インストール済み管理パッケージを設定リストに追加するには、使用可能なパッケージのリストからパッケージを選択します。リストは、クラスまたはトリガにまだ関連付けられていないインストール済み管理パッケージがある場合にのみ表示されます。
3. [保存] をクリックします。

パッケージバージョン設定を使用する場合は、次のことに注意してください。

- 管理パッケージのバージョンを指定せずに、管理パッケージを参照する Apex クラスまたはトリガを保存する場合、Apex クラスまたはトリガは、デフォルトで、管理パッケージの最新のインストールバージョンに関連付けられます。
- パッケージをクラスまたはトリガで参照している場合は、管理パッケージのバージョン設定は [削除] できません。[連動関係の表示] を使用して、クラスまたはトリガから参照されている管理パッケージがどこにあるか検索できます。

## 第5章

### Apex のテスト

---

トピック:

- [Apex のテストについて](#)
- [Apex の単体テスト](#)
- [単体テストメソッドの実行](#)
- [ベストプラクティスのテスト](#)
- [テストの例](#)

Apex は、単体テストの記述、テストの実行、テスト結果の確認、コードカバー率の結果の取得を可能にする、テストフレームワークを提供します。

この章では、Apex のテストに Force.com プラットフォームで使用できるツールのほか、単体テストの概要、テストのデータ表示について説明します。

- [Apex のテストについて](#)
- [Apex の単体テスト](#)
- [単体テストメソッドの実行](#)
- [ベストプラクティスのテスト](#)
- [テストの例](#)

## Apex のテストについて

テストは、長期間の開発を正常に行うための主要部分であり、開発プロセスの重要な部分を占めます。テストコードを開発時に同時に作成する、テスト駆動型の開発プロセスで開発することを強くお勧めします。

## Apex テストを行う理由

テストは、アプリケーションをお客様にリリースするものである場合は特に、アプリケーションの成功の鍵となります。アプリケーションが予測どおりに機能すること、また、予期せぬ動作がないことを検証することで、お客様からの信頼が高まります。

アプリケーションのテストには2種類あります。1つは Salesforce ユーザインターフェースによる重要なテストですが、ユーザインターフェースを使用するテストでは、アプリケーションの使用ケースのすべてを把握できるわけではありません。もう1つは一括機能のテストで、SOAP API を使用して、または Visualforce 標準セットコントローラによってコードが呼び出され場合、そのコードを通じて最大 200 件のレコードを渡すことができます。

アプリケーションが完成することはほとんどありません。機能を変更または拡張する追加リリースがあります。包括的なテストを作成すれば、新しい機能のすべてについて機能の後退が存在しないこと確認することができます。

作成したコードをリリースしたり Force.com AppExchange 用にパッケージ化したりする前に、次の条件を満たす必要があります。

- Apex コードの少なくとも 75% が単体テストでカバーされており、かつすべてのテストが成功している。

次の点に注意してください。

- 本番組織にリリースするときに、組織の名前空間内のすべての単体テストが実行されます。
- へのコールは、Apex コードカバー率の対象とはみなされません。
- テストメソッドとテストクラスは、Apex コードカバー率の対象とはみなされません。
- Apex コードの 75% が単体テストでカバーされている必要がありますが、カバー率を上げることだけに集中すべきではありません。アプリケーションのすべてのユースケース（正・誤両方の場合や单一データだけでなく複数データの場合）の単体テストを作成するようにしてください。このような多様なユースケースのテストコードを実装することが 75% 以上のカバー率につながります。

- すべてのトリガについて何らかのテストを行う。
- すべてのクラスとトリガが正常にコンパイルされる。

Salesforce は Apex コードを使用するすべての組織ですべてのテストを実行し、サービスのアップグレードによって動作が変更されていないことを検証します。

## Apex のテスト内容

Salesforce.com は次の事項のテストを作成することをお勧めします。

### 単一操作

単一のレコードが適切かつ予測どおりの結果を生成することを確認するテスト。

## 一括操作

トリガ、クラス、拡張にかかわらず、すべての Apex コードが 1 件から 200 件のレコードについて呼び出されます。単一レコードのケースだけでなく、一括ケースについてもテストする必要があります。

## ポジティブ動作

予測される動作がすべての予測される順列で行われること、つまりユーザがすべてを正しく入力し、制限を超えないことを確認するテスト。

## ネガティブ動作

将来の日付を追加できない、負の数量を指定できないなどの制限がアプリケーションに存在する場合があります。ネガティブケースについてテストし、制限内のポジティブケースと同様、エラーメッセージが適切に生成されることを確認する必要があります。

## 制限ユーザ

コード内で使用する sObjectsへのアクセス権限が制限されているユーザが予測どおりの動作を行えるかどうかを確認するテスト。つまり、コードを実行できるかどうか、エラーメッセージを受信するかどうかを確認します。



メモ: 条件演算子および 3 項演算子は、ポジティブブランチとネガティブブランチの両方が実行されない限り、実行されたとはみなされません。

これらの種類のテストの例は、「[テストの例](#)」(ページ 242)を参照してください。

# Apex の単体テスト

堅牢で、エラーのないコードの開発を促進するため、Apex は単体テストの作成と実行をサポートします。単体テストは、コード内の特定の部分が正しく機能していることを確認するクラスメソッドです。単体テストのメソッドは引数を取らず、データベースへのデータの確定やメールの送信を行うこともなく、メソッド定義にキーワードまたはアノテーションでフラグが付けられています。また、テストメソッドは、テストクラス(アノテーションが付加されているクラス)で定義されている必要があります。

次に例を示します。

*code\_block*

これは前の例と同じテストクラスですが、代わりに、

アノテーションを使用してテストメソッドを定義

します。

*code\_block*

アプリケーションのテストに使用するコードのみを含むクラスまたは個別のメソッドを定義するには アノテーションを使用します。 アノテーションは、 として宣言するメソッドの作成と似ています。



メモ: アノテーションで指定したクラスは、Apex コードの組織内の上限の 3 MB には含まれません。

次に、2 つのテストメソッドを含むテストクラスの例を示します。



メモ: Salesforce.com API 28.0 からは、テストメソッドは非テストクラスに含めることができなくなります。また、 アノテーションが付加されているクラスの一部である必要があります。テストクラスから private クラスのメンバーにアクセスする方法については、 アノテーションを参照してください。

## 単体テストの考慮事項

単体テストでは、次の点に留意してください。

- ・ テストメソッドは、Web サービスコールアウトのテストには使用できません。代わりに、疑似コールアウトを使用します。「[Web サービスコールアウトのテスト](#)」および「[HTTP コールアウトのテスト](#)」を参照してください。
- ・ テストメソッドからメールメッセージを送信することはできません。
- ・ テストメソッドはテストで作成したデータをコミットしないため、完了時にテストデータを削除する必要はありません。
- ・ 一意制約のある項目を含む一部の sObject では、重複する sObject レコードを挿入するとエラーになります。たとえば、同じ名前の複数の CollaborationGroup sObject を挿入すると、CollaborationGroup レコードには一意の名前が必要なためエラーになります。
- ・ Chatter フィードのレコードの追跡変更 (FeedTrackedChange レコード) は、テストメソッドが関連レコードを変更すると、使用できません。FeedTrackedChange レコードでは、作成される前に、関連付けられている親レコードへの変更がデータベースにコミットされている必要があります。テストメソッドではデータをコミットしないため、FeedTrackedChange レコードの作成はできません。同様に、項目履歴管理レコード (AccountHistory など) は、他の sObject レコード (Account など) を最初にコミットする必要があるため、テストメソッドでは作成できません。

## 関連リンク

[IsTest アノテーション](#)

## 単体テストの組織データとテストデータの分離

Salesforce.com API バージョン 24.0 以降で保存された Apex コードより、テストメソッドは、標準オブジェクト、カスタムオブジェクト、およびカスタム設定データなどの組織の既存のデータにアクセスできません。アクセスできるのは、テストメソッドが作成するデータのみです。ただし、組織またはメタデータオブジェクトの管理に使用する次のオブジェクトは、そのままテストでアクセスできます。

- ・ User
- ・ Profile
- ・ Organization
- ・ RecordType
- ・ ApexClass
- ・ ApexTrigger
- ・ ApexComponent
- ・ ApexPage

可能な場合は常に、テストごとにテストデータを作成する必要があります。この制限は、

アノテーションで、テストクラスまたはテストメソッドにアノテーションを付加することによって無効にできます。詳細は、「[アノテーション](#)」を参照してください。

Salesforce.com API バージョン 23.0 以前を使用して保存されたテストコードは、引き続き、組織のすべてのデータにアクセスすることができ、そのデータアクセス権は変わりません。

## データアクセスに関する考慮事項

- Salesforce API バージョン 24.0 以降を使用して保存された新しいテストメソッドが、バージョン 23.0 以前を使用して保存された別のクラスのメソッドをコールする場合、コール元のデータアクセス制限がコールされるメソッドに適用されます。つまり、コールされるメソッドは、以前のバージョンで保存されていても、コール元にアクセス権がないため組織データにアクセスできません。
- テストデータへのこのアクセス制限は、テストコンテキストで実行されるすべてのコードに適用されます。たとえば、テストメソッドによりトリガが実行されても、テストが組織データにアクセスできない場合は、トリガも実行できません。
- テストで Visualforce 要求を行う場合、実行中のテストはテストのコンテキストに留まりますが、別のスレッドで実行するため、テストデータの分離が行われません。この場合、Visualforce 要求を開始した後で、組織内のすべてのデータにアクセスできるようになります。ただし、Visualforce 要求が JavaScript Remoting コールなどのコールバックを実行する場合、コールバックで挿入されたデータはテストからは認識できません。
- 特定の制限により、テストメソッドから特定のデータ型を作成できない場合があります。この制限の例として、次のようなものがあります。
  - ◊ 実行中のテストでは標準価格表にアクセスできず、作成もできないため、テストから商品の価格表エントリを挿入することはできません。また、カスタム価格表の価格表エントリの挿入は、標準価格表の定義が必要になるため、サポートされていません。このような場合は、テストメソッドにアノテーションを付加して、テストが組織データにアクセスできるようにします。
  - ◊ 標準オブジェクトの中には、作成できないものがあります。これらのオブジェクトについての詳細は、『[Salesforce および Force.com のオブジェクトリファレンス](#)』を参照してください。
  - ◊ 一意制約のある項目を含む一部の sObject では、重複する sObject レコードを挿入するとエラーになります。たとえば、同じ名前の複数の CollaborationGroup sObject を挿入すると、CollaborationGroup レコードには一意の名前が必要なためエラーになります。これは、テストでのアノテーションが付加されているかどうかに関わらず発生します。
  - ◊ Chatter の追跡変更のように、関連レコードがデータベースにコミットされた後にのみ作成されるレコードがあります。Chatter フィードのレコードの追跡変更 (FeedTrackedChange レコード) は、テストメソッドが関連レコードを変更すると、使用できません。FeedTrackedChange レコードでは、作成される前に、関連付けられている親レコードへの変更がデータベースにコミットされている必要があります。テストメソッドではデータをコミットしないため、FeedTrackedChange レコードの作成はできません。同様に、項目履歴管理レコード (AccountHistory など) は、他の sObject レコード (Account など) を最初にコミットする必要があるため、テストメソッドでは作成できません。

## runAs メソッドの使用

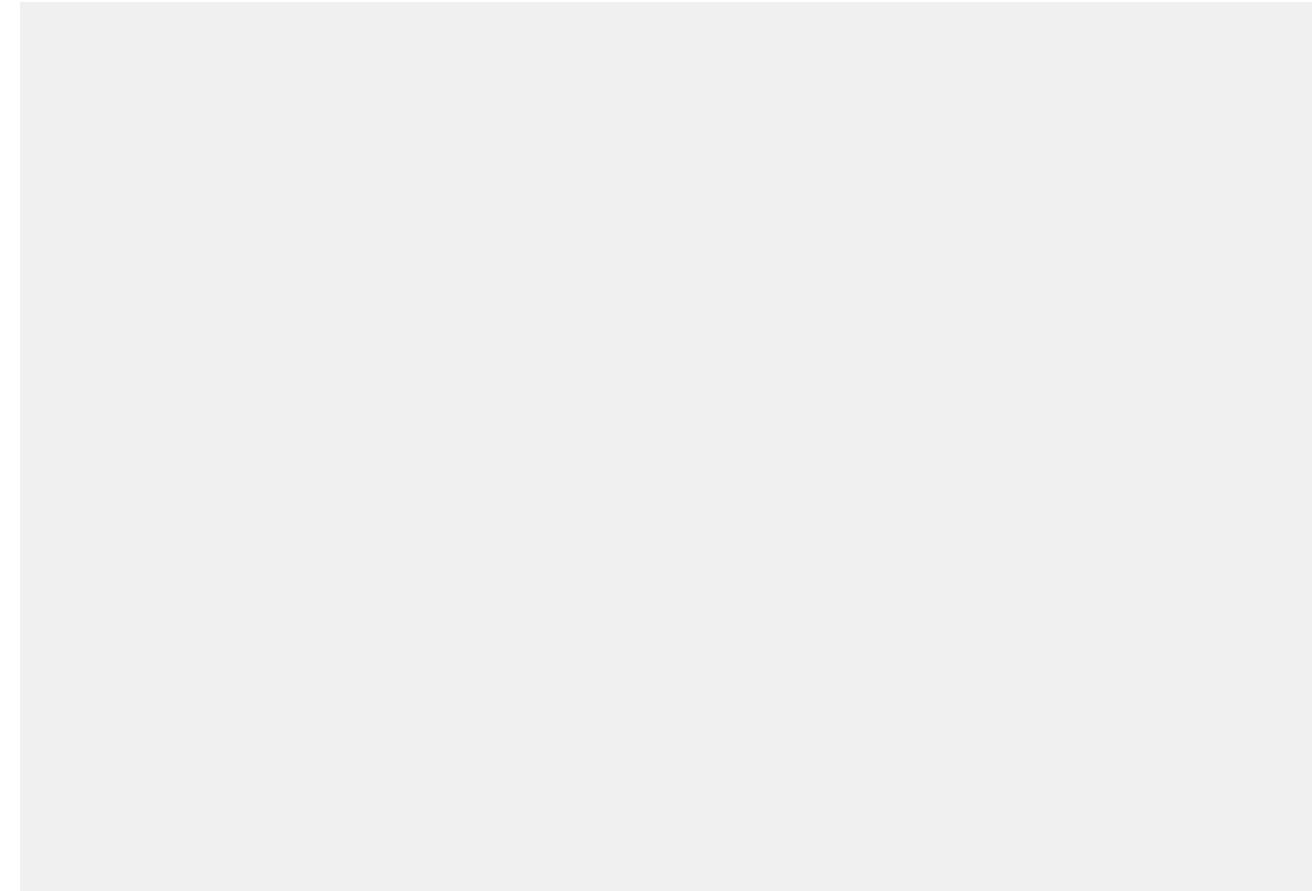
一般に、Apex コードはすべてシステムモードで実行され、現在のユーザの権限やレコード共有は考慮されません。システムメソッドを使用すると、ユーザコンテキストを既存のユーザまたは新規ユーザに変更するテストメソッドを作成したり、特定バージョンの管理パッケージのコードを使用して実行したりできます。ユーザとして実行する場合、ユーザのレコード共有のすべてが適用されます。テストメソッドではのみ使用できます。元のシステムコンテキストは、すべての テストメソッドが完了した後で再開されます。 メ

ソッドの使用とパッケージバージョンコンテキストの指定についての詳細は、[「パッケージバージョンの動作のテスト」](#)（ページ 349）を参照してください。

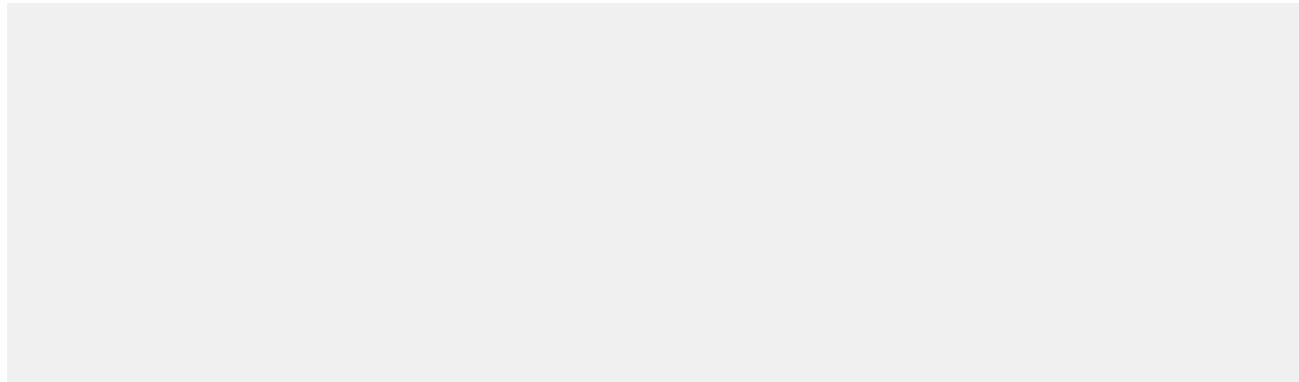


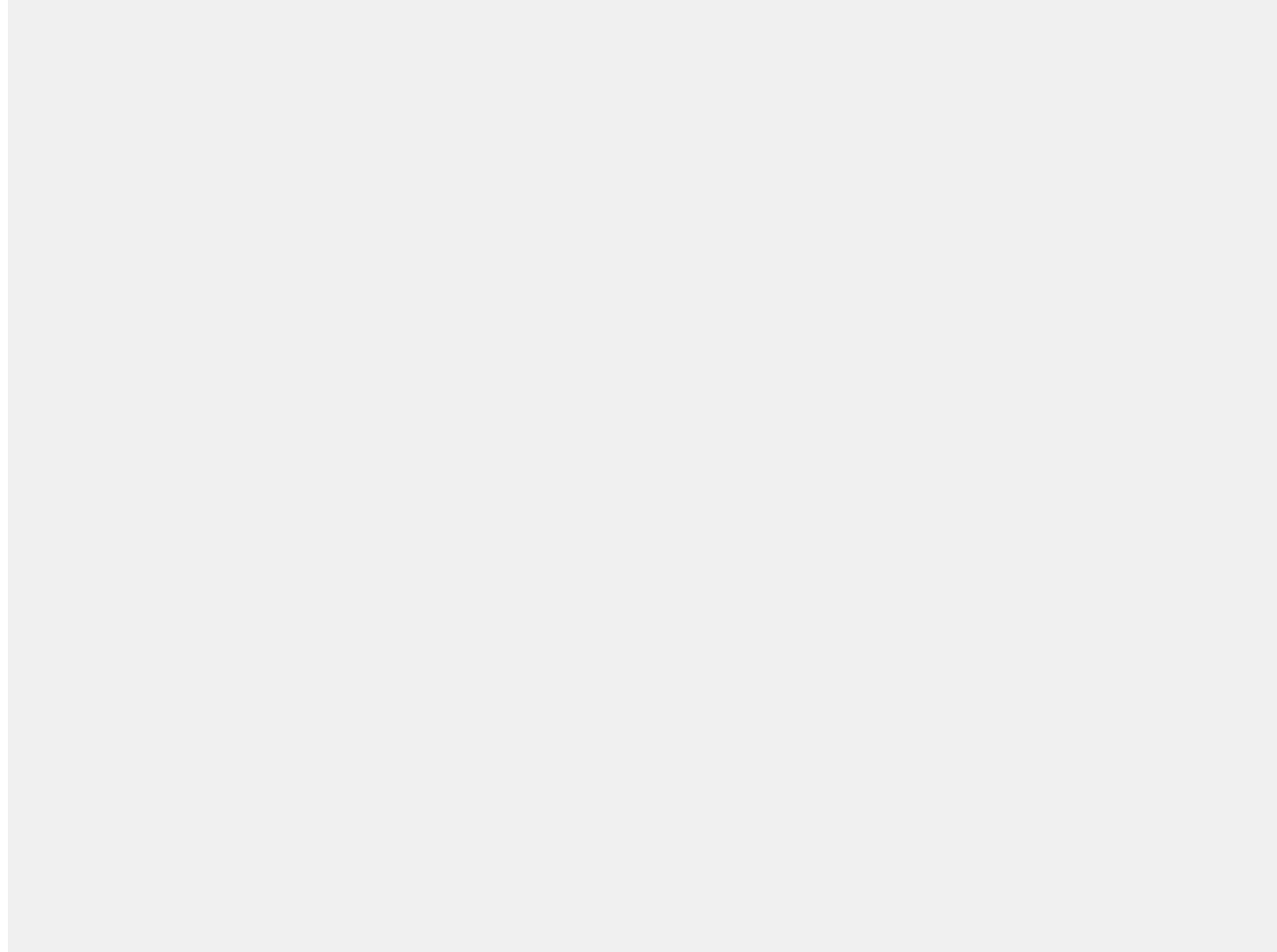
メモ: の各コールは、プロセスで発行される DML ステートメントの合計数にカウントされます。

次の例では、新しいテストユーザが作成され、コードはそのユーザとして、ユーザの権限およびレコードアクセス権限で実行されます。



複数の メソッドをネストできます。次に例を示します。





## runAs を使用するベストプラクティス

次の項目では、特定ユーザとして実行する `runAs` で指定されたユーザが割り当てた権限を使用します。

- ・ 動的 Apex
- ・ `System.runAs()` または `User.setCurrentUser()` を使用するメソッド
- ・ 共有レコード

元の権限は、`runAs` の完了後にリセットされます。

メソッドは、ユーザライセンスの制限を無視します。組織に追加ユーザライセンスがない場合でも、`runAs` で新しいユーザを作成できます。

## Limits、startTest、およびstopTest の使用

`Limits` メソッドは、メソッドのコール数やヒープサイズの残りの量など、特定のガバナの具体的な制限を返します。

各メソッドには2つのバージョンがあります。一方のバージョンのメソッドは現在のコンテキストで使用されているリソースの数を返し、もう一方のバージョンは `limit` という語を使用し、該当するコンテキストに使用できるリソースの合計数を返します。たとえば、`System.Limits.getHeapSize()` は現在のコンテキストで処理済みの外部サービスヘ

のコールアウト数を返し、  
返します。

Limits メソッドのほかに、  
メソッドと  
メソッドを使用して、コードがガバナ制限にどれく  
らい近づいているかを確認します。

メソッドは、テストコード内のテストが実際に開始するポイントをマークします。各テストメソッド  
は、このメソッドを1回のみコールできます。このメソッドの前のすべてのコードを、変数の初期化、データ構  
造の入力などのために使用する必要があります。これにより、テストを実行するために必要なすべてを設定でき  
ます。  
へのコールの後および  
の前に実行するコードはすべて、新しいガバナ制限セットが  
割り当てられます。

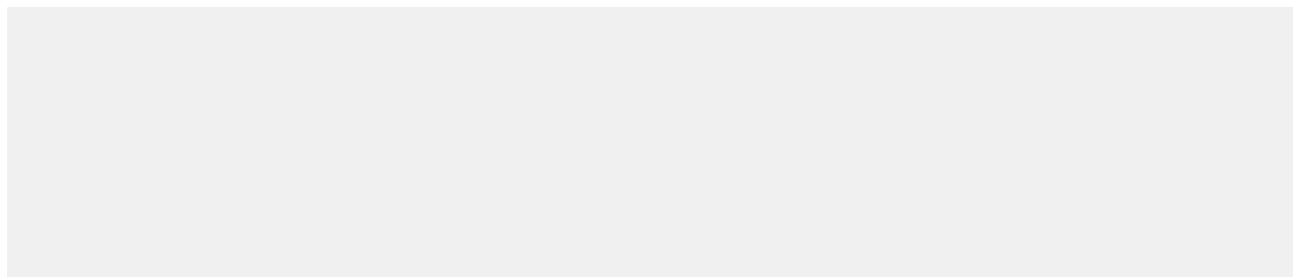
メソッドはテストのコンテキストを更新せず、コンテキストをテストに追加します。たとえば、クラ  
スが  
をコールする前に98件のSOQL クエリを作成し、  
後の最初の有意なステートメントがDMLステートメントである場合、プログラムはさらに100件のクエリを作成できます。ただし、  
がコールされると、プログラムは元のコンテキストに戻り、100件の制限に達するまで追加できるSOQL クエリ  
は2件だけになります。

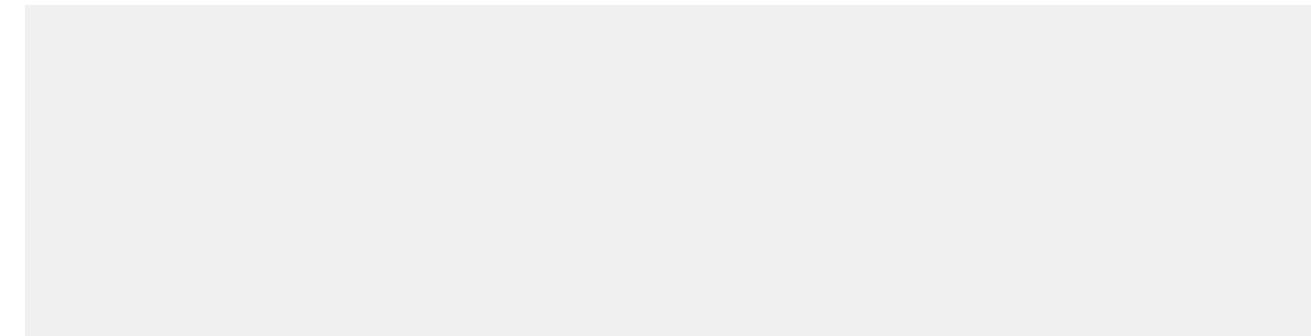
メソッドは、テストコード内のテストが終了するポイントをマークします。このメソッドは  
メソッドと組み合わせて使用します。各テストメソッドは、このメソッドを1回のみコールできます。  
メソッドの後に実行するコードはすべて、  
がコールされる前に有効だった元の制限が割り当てられま  
す。  
メソッドの後に作成されたすべての非同期コールはシステムによって収集されます。  
を実行する場合、すべての非同期プロセスが同期して実行されます。

## SOSL クエリの単体テストへの追加

テストメソッドが必ず予測されたとおりに動作するように、Apex テストメソッドに追加される Salesforce オブ  
ジェクト検索言語 (SOSL) クエリは、テストメソッドが実行された場合に検索結果の空のセットを返します。ク  
エリが結果の空のリストを返さないようにする場合は、  
システムメソッドを使  
用して、検索で返されるレコード ID のリストを定義できます。テストメソッドの後半で実行される SOSL クエ  
リは、  
メソッドで指定されたレコード ID のリストを返します。また、テス  
トメソッドは  
を複数回コールして、さまざまな SOSL クエリのさまざまな結果  
セットを定義できます。テストメソッドで  
メソッドをコールしない場合、または  
はレコード ID のリストを指定しないでこのメソッドをコールする場合、テストメソッドの後半で実行される  
SOSL クエリは、結果の空のリストを返します。

メソッドで指定されたレコード ID のリストは、  
句または  
句が適  
用されない場合に通常 SOSL クエリで返される結果を置き換えます。これらの句が SOSL クエリに存在する場合  
は、固定された検索結果のリストに適用されます。次に例を示します。





ID が である取引先レコードが FIND 句のクエリ文字列 ( ) に一致しない場合がありますが、レコードは SOSL ステートメントの 句に渡されます。ID を持つレコードが 句の検索条件に一致する場合、レコードが返されます。 句に一致しない場合、レコードは返されません。

## 単体テストメソッドの実行

単体テストは次を対象に実行できます。

- ・ 特定のクラス
- ・ クラスのサブセット
- ・ 組織内のすべての単体テスト

テストを実行するには、次のいずれかを使用します。

- ・ [Salesforce ユーザインターフェース](#)
- ・ [Force.com IDE](#)
- ・ [API](#)

### Salesforce ユーザインターフェースによるテストの実行

Apex テスト実行ページで単体テストを実行できます。このページで開始したテストは非同期的に実行するため、テストクラスの実行が完了するのを待つ必要はありません。Apex テスト実行ページは、テストが完了すると、テストの状況を更新して結果を表示します。

The screenshot shows the 'Apex Test Execution' page. At the top, there's a message: 'Click Select Tests to choose one or more Apex unit tests and run them. To see the current code coverage for an individual class or your organization, go to the [Apex Classes page](#)'. Below this are buttons for 'Select Tests...', 'Options...', and 'View Test History'. A large 'Abort' button is prominently displayed. A table header row has columns for 'Status', 'Class', and 'Result'. Under the 'Result' column, it says '(2/2) Test Methods Passed'. Below this, a section titled 'Test Run: 2012-03-16 10:16:28, jsmith@acme.org (2 Classes)' lists two entries: 'TestClass1' with status 'Pass' and 'TestClass2' with status 'Pass'. At the bottom, a detailed table shows three rows of test results with columns for 'Detail', 'Duration', 'Class', 'Method', 'Pass/Fail', 'Error Message', and 'Stack Trace'. The first two rows show 'Pass' for both methods. The third row shows 'Fail' for 'test2' with the error message 'System.AssertException: Assertion Failed' and stack trace 'Class.TestClass1.test2: line 20, column External entry point'.

Apex テスト実行ページを使用する手順は、次のとおりです。

1. [設定] で、[開発] > [Apex テスト実行] をクリックします。
2. [テストを選択...] をクリックします。



メモ: 管理パッケージからインストールされた Apex クラスがある場合、最初に [Apex クラス] ページの [すべてのクラスをコンパイル] をクリックしてこれらのクラスをコンパイルして、リストに表示されるようにする必要があります。 「Apex クラスの管理」を参照してください。

3. 実行するテストを選択します。テストのリストには、テストメソッドが含まれるクラスが表示されます。

- インストール済み管理パッケージからテストを選択するには、ドロップダウンリストから対応する名前空間を選択します。リストには、選択した名前空間の管理パッケージのクラスのみ表示されます。
- 組織にローカルに存在するテストを選択するには、ドロップダウンリストから [私の名前空間] を選択します。管理パッケージからインストールされたクラスを除くローカルクラスのみリストに表示されます。
- テストを選択するには、ドロップダウンリストから [すべての名前空間] を選択します。管理パッケージからインストールされたかどうかに関わらず、組織内のすべてのクラスが表示されます。



メモ: テストがまだ実行中のクラスは、リストに表示されません。

4. [実行] をクリックします。

Apex テスト実行ページを使用してテストを実行した後、そのテストによるコードカバー率を Apex クラス一覧に表示できます。[設定] から、[開発] > [Apex クラス] をクリックした後、[組織のコードカバー率を見積る] をクリックします。



メモ: [組織のコードカバー率を見積る] で計算したコードカバー率の値が、[すべてのテストを実行] を使用してすべての単体テストを実行した後に計算したコードカバー率の値と異なる場合があります。これは、[組織のコードカバー率を見積る] では、インストールされた管理パッケージに含まれるクラスを除外するのに対し、[すべてのテストを実行] では除外されないためです。

個々のクラスのテストで、コードのどの行がテストされるかを確認することもできます。[設定] から、[開発] > [Apex クラス] をクリックした後、クラスの [コードカバー率] 列にあるパーセント値をクリックします。

[設定] から、[開発] > [Apex テスト実行] > [テスト履歴を表示] をクリックし、自分で実行したテストだけでなく、組織のすべてのテスト結果を表示します。テスト結果は、消去されない限り、テストの完了後 30 日間保持されます。

## Force.com IDE を使用したテストの実行

また、Force.com IDE を使用してテストを実行することもできます  
([を参照](#))。

## API を使用したテストの実行

コードを SOAP API から使用して、テストを同期して実行できます。

このコードでは、すべてのクラスのすべてのテスト、特定の名前空間のすべてのテスト、または特定の名前空間のクラスのサブセットにあるすべてのテストを、RunTestsRequest オブジェクトに指定されるとおりに実行できます。次の値が返されます。

- 実行されたテストの合計数
- コードカバー率の統計 (下記で説明)
- 失敗したテストごとのエラー情報
- 成功したテストごとの情報
- テストの実行に要した時間

の詳細については、WSDL を参照してください。これは、  
*your\_salesforce\_server* にあります。*your\_salesforce\_server* は、  
など、組織が置かれているサーバに相当します。

Salesforce 本番組織の管理者は Salesforce ユーザインターフェースを使用して Apex コードに変更を加えることはできませんが、既存項目への固有の制約の追加など、何らかの変更を行った後に、既存の単体テストが確実に実行されて完了されるように、  
を使用することは重要です。Salesforce 本番組織で Apex コードを変更するには  
SOAP API コールを使用する必要があります。詳細は、「[Apex のリリース](#)」(ページ 950)を参照してください。

の詳細は、「[Apex の SOAP API および SOAP ヘッダー](#)」(ページ 988)を参照してください。

## ベストプラクティスのテスト

効果的なテストでは、次のことを行います。

- 可能な限り多くのコード行をカバーする。Apex をリリースまたは Force.com AppExchange 用にパッケージ化する前に、次の条件を満たす必要があります。



重要:

- Apex コードの少なくとも 75% が単体テストでカバーされており、かつすべてのテストが成功している。

次の点に注意してください。

- 本番組織にリリースするときに、組織の名前空間内のすべての単体テストが実行されます。
- ヘルプのコードは、Apex コードカバー率の対象とはみなされません。
- テストメソッドとテストクラスは、Apex コードカバー率の対象とはみなされません。
- Apex コードの 75% が単体テストでカバーされている必要がありますが、カバー率を上げることだけに集中すべきではありません。アプリケーションのすべてのユースケース（正・誤両方の場合や單一データだけでなく複数データの場合）の単体テストを作成するようにしてください。このような多様なユースケースのテストコードを実装することが 75% 以上のカバー率につながります。
  - ◊ すべてのトリガについて何らかのテストを行う。
  - ◊ すべてのクラスとトリガが正常にコンパイルされる。

- 条件ロジックの場合（3 項演算子など）、コードロジックの各ブランチを実行する。
- 有効な入力および無効な入力を使用してメソッドへのコールを行う。
- エラーが予期され、プロックで捕捉されない限り、例外が発生することなく正常に完了する。
- 例外を捕捉するだけでなく、捕捉されたすべての例外を処理する。
- メソッドを使用して、コードが適切に動作することを検証する。
- メソッドを使用して、さまざまなユーザコンテキストでアプリケーションをテストする。
- 一括トリガ機能を実行する。テストで最低 20 件のレコードを使用する。
- キーワードを使用し、レコードが予期された順序で返されるようにする。
- レコード ID が順序立っていることを想定しない。

複数のレコードを同じ要求で挿入しない限り、レコード ID は昇順で作成されません。たとえば、取引先 A を作成し、ID を受信した後で取引先 B を作成した場合、取引先 B の ID が次に大きい順序になる場合とならない場合があります。

- Apex クラスのリストには、[コードカバー率] 列がある。カバー率の数値をクリックするとページが表示され、テストの対象となるクラスまたはトリガのコードの行が青で、テストの対象外であるコードの行が赤で強調表示されます。また、クラスまたはトリガの特定の行がテストで実行された回数も表示されます。
- テストデータを次のように設定する。
  - ◊ テストクラスで必要なデータを作成して、テストが特定の組織のデータに依存する必要がないようにする。
  - ◊ メソッドをコールする前にすべてのテストデータを作成する。
  - ◊ テストでは何も確定しないため、データを削除する必要がない。

- コメントの記述では、テストの内容だけでなく、テスト実施者によるデータに関する想定事項やテスト結果予測などを明記する。
- アプリケーションで個別にクラスをテストする。1 回のテストでアプリケーション全体をテストしない。

多数のテストを実行する場合は、次の点を考慮します。

- Force.com IDE では、Apex プロジェクトの 参照タイムアウト 値を大きくする必要がある。詳細は、[を参照してください。](#)

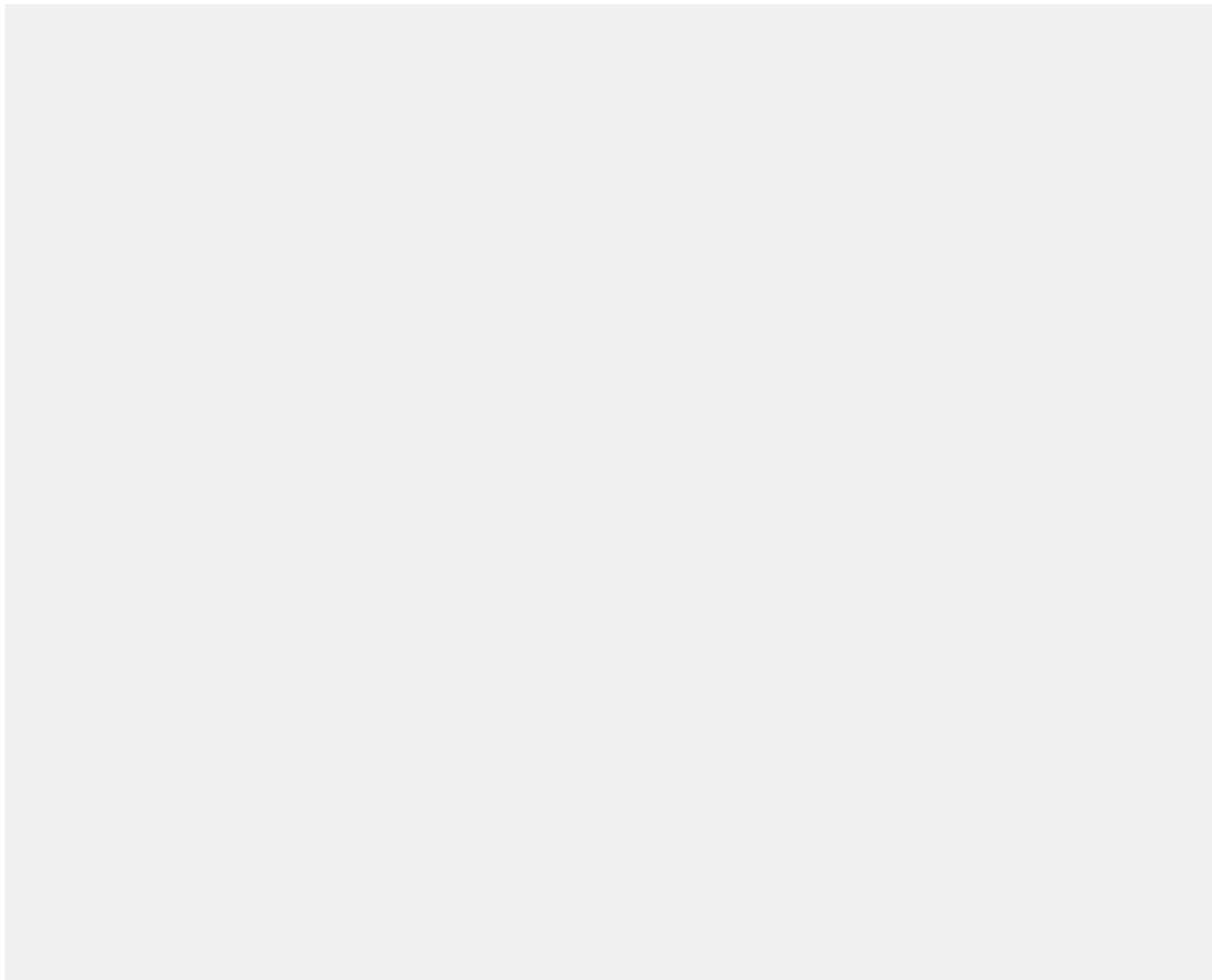
- Salesforce ユーザインターフェースで、[すべてのテストを実行] ボタンを使用してすべてのテストを同時に実行するのではなく、組織内のクラスで個別にテストを実行する必要がある場合がある。

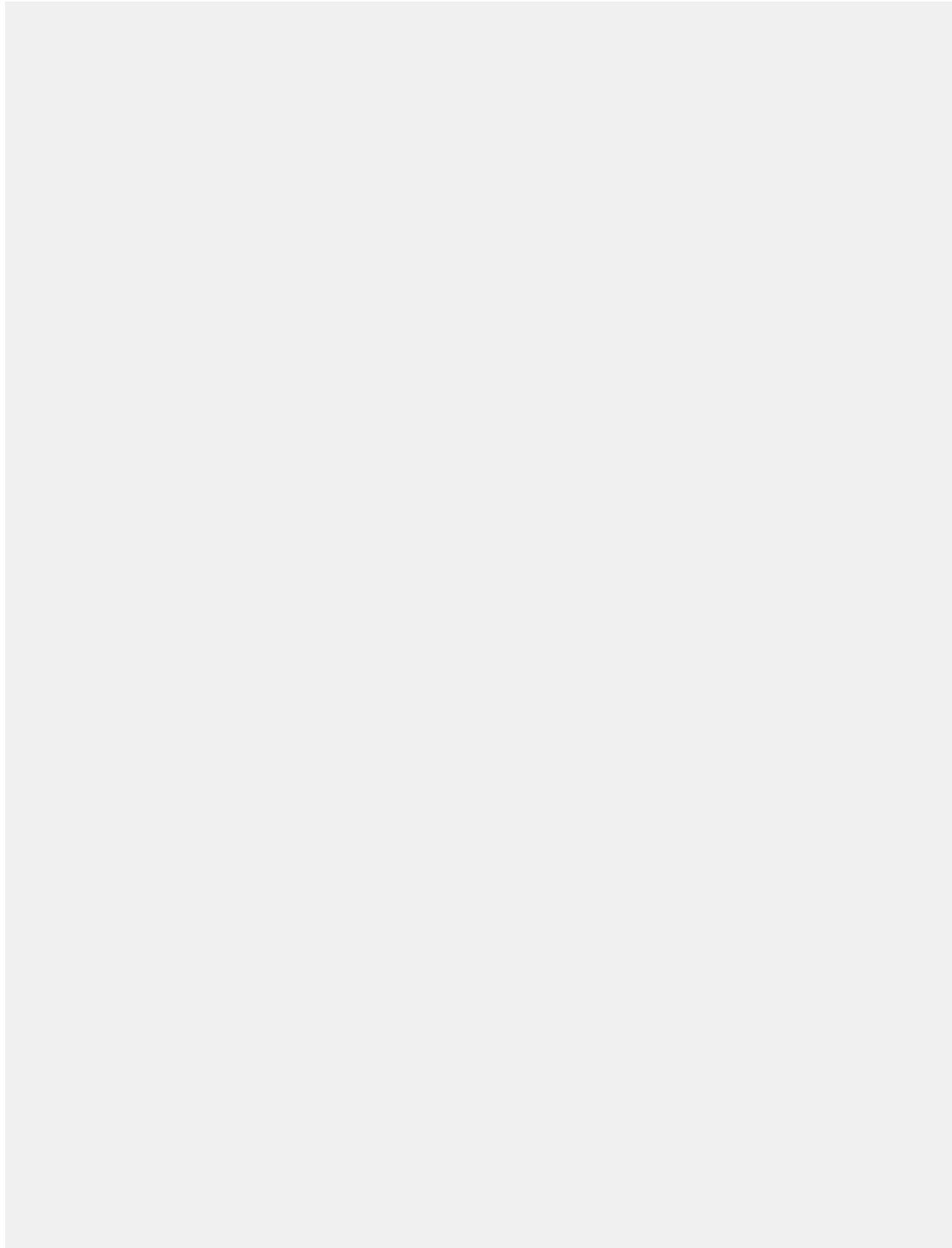
## テストの例

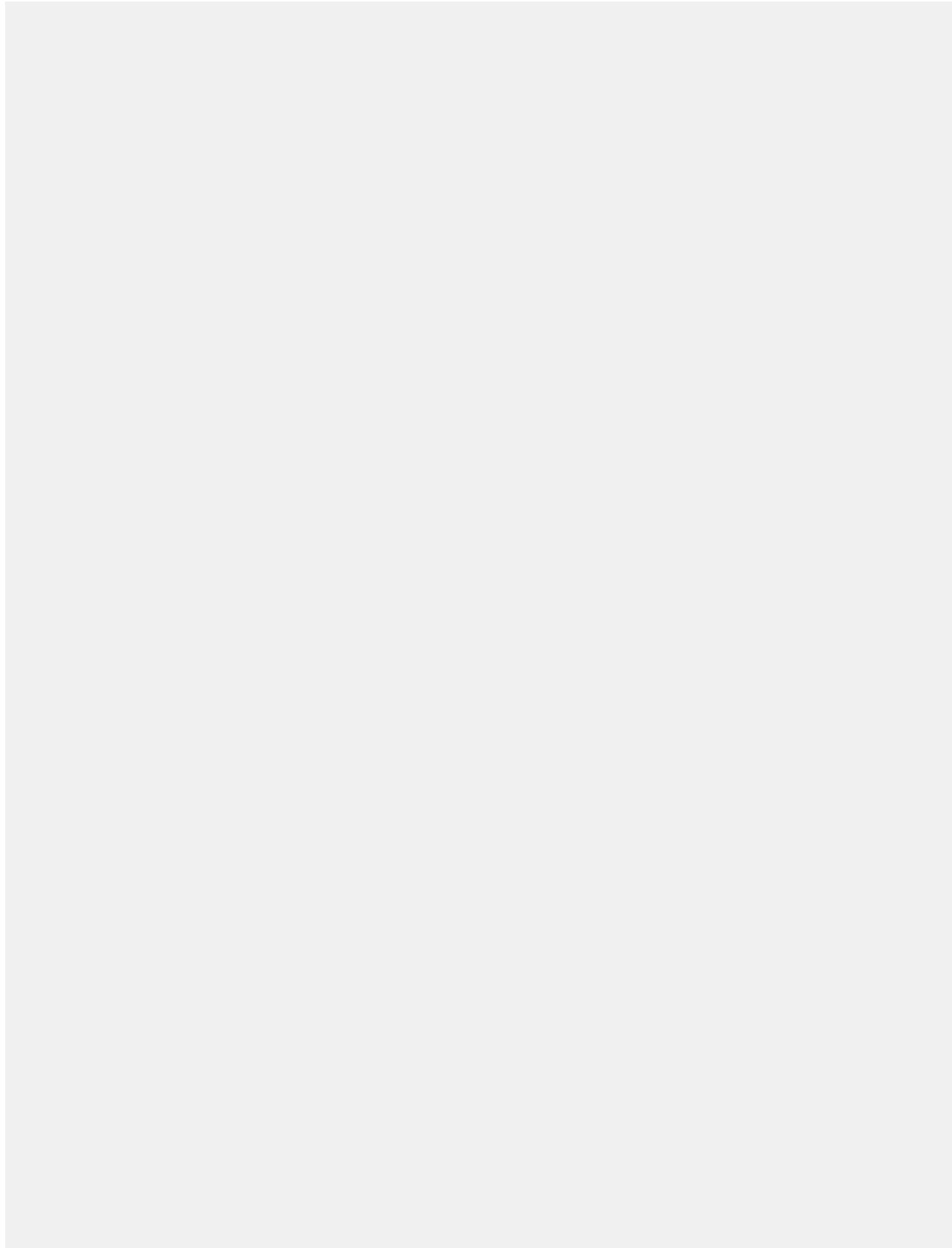
次の例では、下記の種類のテストのケースについて示します。

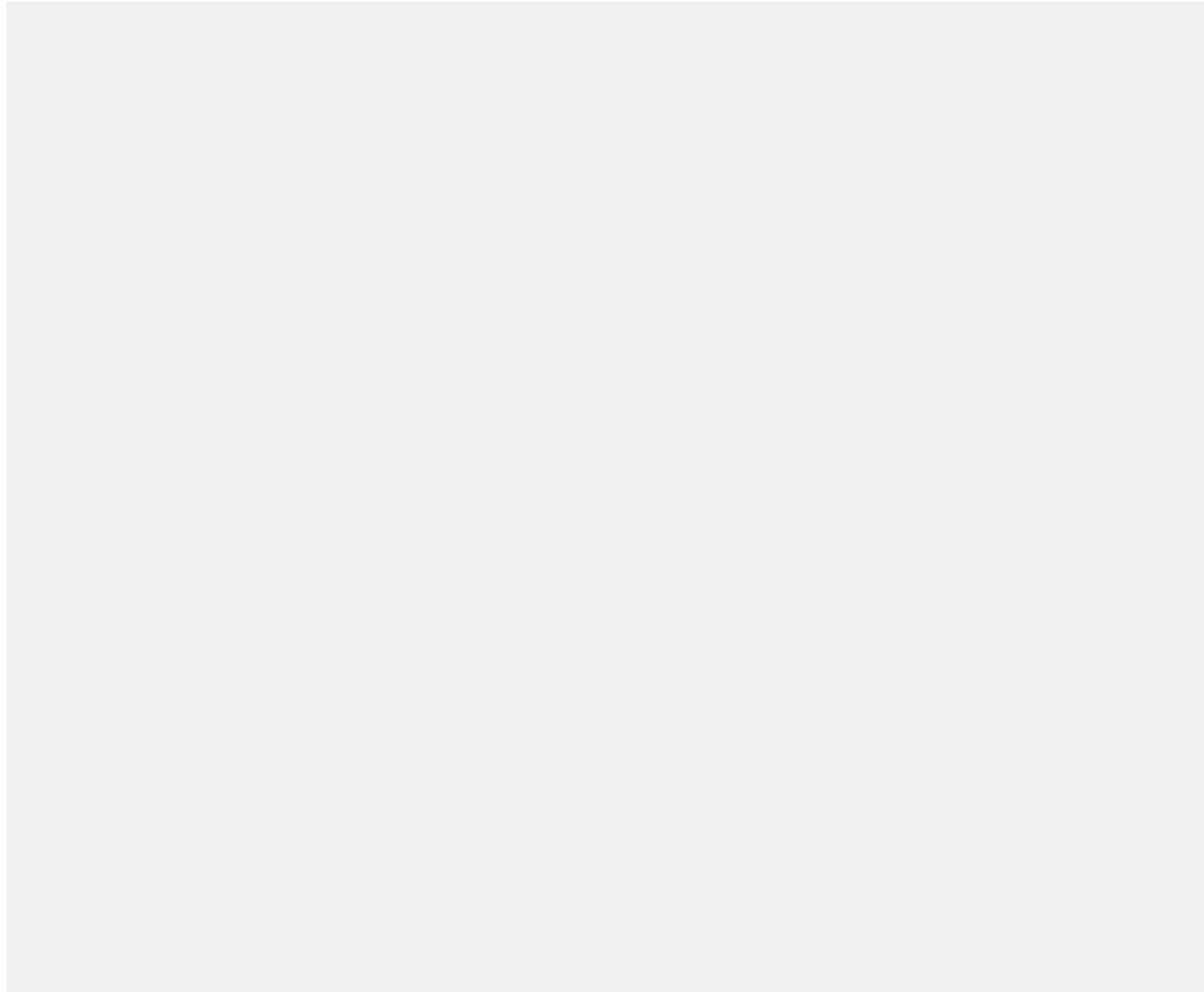
- 単一レコードおよび複数のレコードを含むポジティブケース
- 単一レコードおよび複数のレコードを含むネガティブケース
- 他のユーザによるテスト

単純なマイル追跡アプリケーションでテストを実行します。アプリケーションの既存のコードは、1日に入力されるマイル数が500マイルを超えないことを確認します。主オブジェクトは Mileage\_\_c というカスタムオブジェクトです。全体のテストクラスを次に示します。次のセクションでは、コードの特定の部分の手順を説明します。





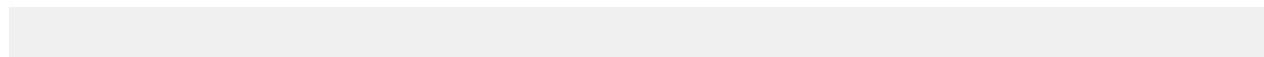




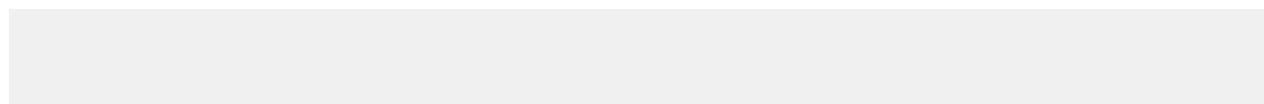
## ポジティブテストケース

上記のコードの、単一レコードおよび複数レコードのポジティブテストケースの手順は、次のとおりです。

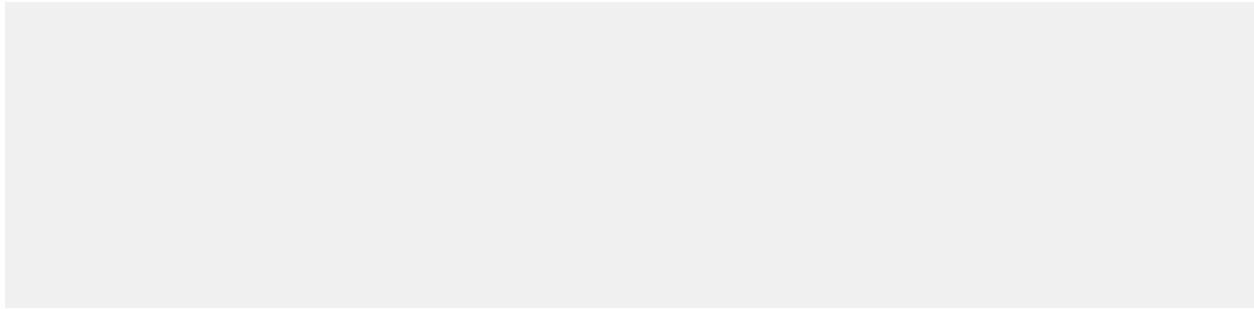
1. デバッグログにテキストを追加して、コードの次のステップを示します。



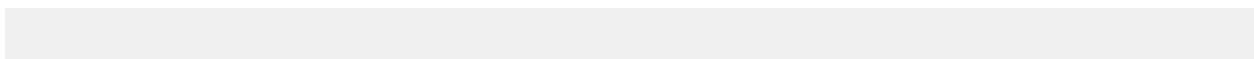
2. Mileage\_\_c オブジェクトを作成し、データベースに挿入します。



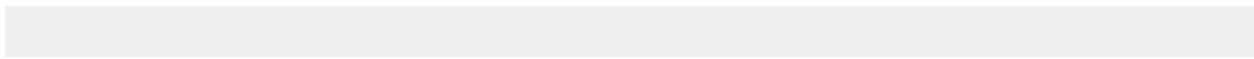
3. 挿入されたレコードを返してコードを検証します。



4. メソッドを使用して、期待どおりの結果が返されたことを確認します。



5. 次のテストに移る前に、合計マイル数を 0 に再設定します。

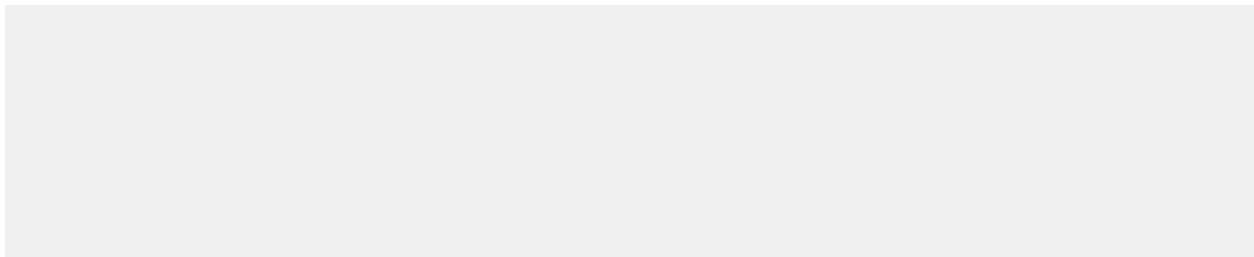


6. 200 レコードの一括挿入を作成して、コードを検証します。

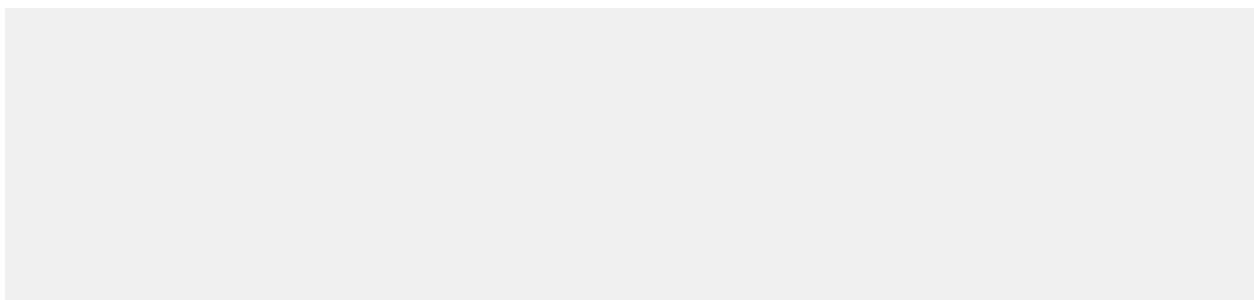
まず、デバッグログにテキストを追加し、コードの次のステップを示します。



7. 次に 200 件の Mileage\_c レコードを挿入します。



8. を使用して、期待どおりの結果が返されたことを確認します。



## ネガティブテストケース

上記のコードのネガティブテストケースの手順は、次のとおりです。

1. という静的テストメソッドを作成します。

2. デバッグルゴにテキストを追加して、コードの次のステップを示します。

3. 501 マイルの Mileage\_\_c レコードを作成します。

4. フレームワークを / ブロック内に配置します。これで、検証の例外を捕捉し、生成されたエラーメッセージを表示できます。

5. および  
た ブロックに追加します。  
を使用してテストを実行します。次のコードを、前に作成し

## セカンドユーザとしてのテスト

上記のコードを、セカンドユーザとして実行する手順は次のとおりです。

1. 次のテストに移る前に、合計マイル数を 0 に再設定します。

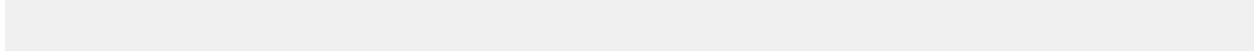
2. 次のユーザを設定します。

3. デバッグルogsにテキストを追加して、コードの次のステップを示します。

4. 次に 1 件の Mileage\_c レコードを挿入します。

- 5.挿入されたレコードを返してコードを検証します。

6. メソッドを使用して、期待どおりの結果が返されたことを確認します。



# 第6章

## 動的 Apex

### トピック:

- [Apex Describe Information について](#)
- [動的 SOQL](#)
- [動的 SOSL](#)
- [動的 DML](#)

動的 Apex を使用すると次の機能が提供されるため、開発者は、より柔軟性の高いアプリケーションを作成できます。

- [sObject と項目の Describe Information へのアクセス](#)

*Describe Information*は、*sObject* と項目プロパティについての情報を提供します。たとえば、*sObject* の *Describe Information* には、作成や復元などの操作をサポートする *sObject* のデータ型、*sObject* の名前と表示ラベル、*sObject* の項目と子オブジェクトなどの情報が含まれます。項目の *Describe Information* には、その項目にデフォルト値があるか、計算項目であるかどうか、項目のデータ型などの情報が含まれます。

*Describe Information* は、個別のレコードではなく、組織のオブジェクトについての情報を提供します。

- [動的 SOQL クエリ、動的 SOSL クエリ、および動的 DML の記述](#)

*動的 SOQL* および *SOSL* クエリにより、SOQL または SOSL を実行時に文字列として実行できます。一方、*動的 DML* では、レコードを動的に作成し、DML を使用してデータベースに挿入できます。動的 SOQL、SOSL、および DML を使用してユーザ権限をカスタマイズできるだけでなく、アプリケーションを組織に合わせて適切にカスタマイズすることもできます。これは、Force.com AppExchange からインストールされたアプリケーションに便利です。

## Apex Describe Information について

Apex は、`sObject` と項目の Describe Information に関する次の 2 つのデータ構造を提供します。

- ・ トーカン—軽量で逐次化可能な `sObject` への参照、またはコンパイル時に検証される項目。
- ・ `DescribeResult`—`sObject` または項目の Describe プロパティすべてを含むオブジェクト。 `DescribeResult` オブジェクトは、逐次化できず、ランタイムで検証されます。

トーカンからその `DescribeResult` まで、または `DescribeResult` からトーカンまでの移動は簡単です。`sObject` と項目トーカンには両方とも、トーカンの `DescribeResult` を返すメソッドがあります。`DescribeResult` で、`getSObject()` メソッドと `getRecordType()` メソッドは、`sObject` と項目にそれぞれのトーカンを返します。

トーカンは軽量であるため、それを使用すると、コードはより高速で効率的になります。たとえば、コードで使用する必要がある `sObject` のデータ型または項目を決定するときに、`sObject` または項目のトーカンバージョンを使用します。たとえば、`sObject` が `Account` オブジェクトであるかどうか、または項目が「項目とカスタム計算項目のどちらであるか」を決定するには、等価演算子(=)を使用してトーカンを比較できます。

次のコードは、`sObject` プロパティと項目プロパティに関する情報にアクセスするための、トーカンと `DescribeResult` の使い方の一般的な例を示しています。

次のアルゴリズムは、Apex で Describe Information を使用できる方法を示しています。

- 組織の sObject のトークンのリストまたは対応付けを作成します（「すべての sObject へのアクセス」（ページ 255）を参照）。
- アクセスする必要がある sObject を決定します。
- sObject の Describe Result を生成します。
- 必要に応じて、sObject の項目トークンの対応付けを作成します（「sObject のすべての Field Describe Result へのアクセス」（ページ 256）を参照）。
- コードがアクセスする必要がある Field Describe Result を作成します。

## Describe Information 権限について

Apex は通常、システムモードで実行されます。パッケージに含まれないすべてのクラスとトリガ、つまり組織に元々あるものは、動的に検索できるため sObject では制限されません。これは、ネイティブコードでは、現在のユーザ権限に関わらず、組織のためにすべての sObject の対応付けを作成できるという意味です。

Force.com AppExchange からインストールされる salesforce.com ISV パートナーによって作成された管理パッケージに含まれる動的 Apex は、管理パッケージ外の sObject へのアクセスが制限されています。パートナーは、管理パッケージの一部として含まれていない標準 sObject へのアクセスを許可するために、パッケージ内の `allowAccess` 値を設定できます。パートナーは標準オブジェクトへのアクセスを要求できますが、カスタムオブジェクトは管理パッケージの一部として含まれないため、パッケージ化された動的 Apex からカスタムオブジェクトを参照またはアクセスすることはできません。

詳細は、Salesforce オンラインヘルプの「パッケージの API および動的 Apex アクセスについて」を参照してください。

## sObject トークンの使用

Account や MyCustomObject\_\_c などの sObject は、トークンと Describe Result にアクセスするための特別な静的メソッドとメンバー変数を持った静的クラスとして機能します。Describe Result へのアクセス権を得るには、コンパイル時に sObject と項目名を明示的に参照する必要があります。

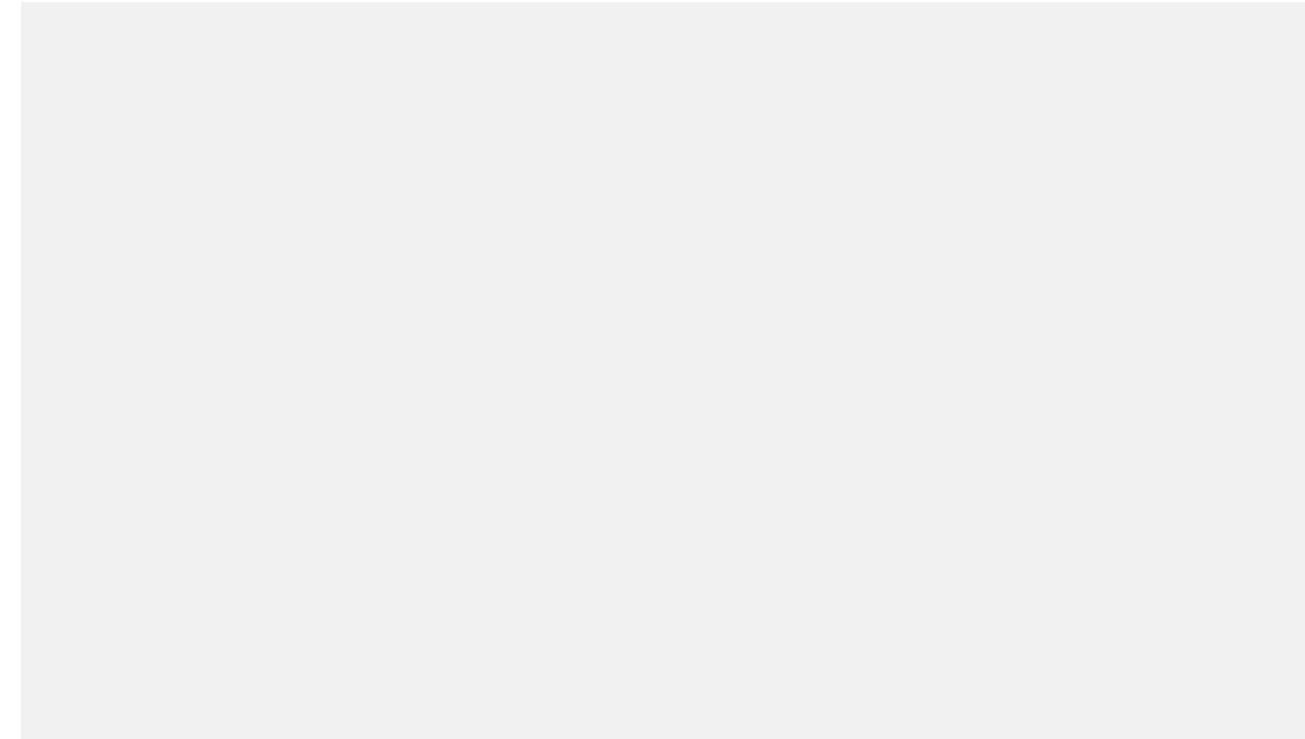
sObject のトークンにアクセスするには、次のいずれかのメソッドを使用します。

- Account などの sObject データ型の `get[Field]` メンバー変数にアクセスします。
- sObject Describe Result、sObject 変数、リスト、または対応付けの `get[Field]` メソッドをコールします。  
は sObject トークンのデータ型です。

次の例では、Account sObject のトークンが返されます。

次の例でも Account sObject のトークンが返されます。

この例は、sObject または sObject リストが特定のデータ型かどうか判断するために使用されます。



一部の標準 sObject には、**isData** と呼ばれる項目があります。たとえば、AssignmentRule、QueueSObject、および RecordType があります。これらのデータ型の sObject は、トークンの取得する場合は常にメソッドを使用します。プロパティを使用する場合(たとえば `isData`)、項目が返されます。

### sObject Describe Result の使用

sObject の Describe Result にアクセスするには、次のいずれかのメソッドを使用します。

- sObject トークンの `getDescribe()` メソッドをコールします。
- sObject の名前が付いている Schema の `getDescribe()` 静的変数を使用します。たとえば、`Account.getDescribe()` です。

は sObject Describe Result のデータ型です。

次の例では、sObject トークンで `getDescribe()` メソッドを使用します。

次の例では、Schema の `getDescribe()` 静的メンバー変数を使用します。

sObject Describe Result で使用可能なメソッドについての詳細は、「[sObject Describe Result メソッド](#)」(ページ 534) を参照してください。

## 項目トークンの使用

項目のトークンにアクセスするには、次のいずれかのメソッドを使用します。

- sObject 静的データ型の静的メンバー変数名、たとえば \_\_\_\_\_ にアクセスします。
- Field Describe Result の \_\_\_\_\_ メソッドをコールします。

項目トークンは、データ型 \_\_\_\_\_ を使用します。

次の例では、項目トークンは Account オブジェクトの \_\_\_\_\_ 項目に返されます。

次の例では、項目トークンは Field Describe Result から返されます。

## Field Describe Result の使用

Field Describe Result にアクセスするには、次のいずれかのメソッドを使用します。

- 項目トークンの \_\_\_\_\_ メソッドをコールします。
- sObject トークンの \_\_\_\_\_ メンバー変数に、項目メンバー変数( \_\_\_\_\_ など)を使用してアクセスします。

Field Describe Result は、データ型 \_\_\_\_\_ を使用します。

次の例では、\_\_\_\_\_ メソッドを使用します。

次の例では、次の \_\_\_\_\_ メンバー変数メソッドを使用します。

上記の例では、システムは、コンパイル時に最終メンバー変数( \_\_\_\_\_ )が指定の sObject に対して有効であることを検証する特殊な解析を使用します。パーサーが \_\_\_\_\_ メンバー変数を見つけたら、sObject( \_\_\_\_\_ )の名前を逆方向に検索して、\_\_\_\_\_ メンバー変数の後の項目名が正当であるかどうかを検証します。\_\_\_\_\_ メンバー変数は、この方式が使用された場合のみ機能します。

1 つの Apex クラスまたはトリガに保持できる \_\_\_\_\_ メンバー変数ステートメントの数は 100 個のみです。



メモ: 項目メンバー変数名またはメソッドのいずれかを使用しない場合、メンバー変数は使用しないでください。『sObject のすべての Field Describe Resultへのアクセス』(ページ 256)の詳細は、を参照してください。

Field Describe Result で使用できるメソッドについての詳細は、『Describe Field Result メソッド』(ページ 539)を参照してください。

## すべての sObject へのアクセス

Schema メソッドを使用して、すべての sObject 名(キー)と sObject トークン(値)間のリレーションを表す対応付けを返します。次に例を示します。

対応付けには次の特性があります。

- 動的である。つまり、権限に基づいて、現在組織に使用できる sObject で実行時に生成されます。
- sObject の名前は大文字と小文字を区別しない。
- キーには、名前空間(ある場合)のプレフィックスが付加される。<sup>\*</sup>
- キーは、sObject がカスタムオブジェクトかどうかを反映する。

\* Salesforce.com API バージョン 28.0 以降を使用して保存された Apex では、が返す対応付け内のキーには、実行されるコードの名前空間(ある場合)が常にプレフィックスとして付加されます。たとえば、コールを行うコードブロックが名前空間 NS1 内にあり、MyObject\_\_c という名前のカスタムオブジェクトが同じ名前空間内にある場合、返されるキーはです。28.0 よりも前の API バージョンを使用して保存された Apex では、コードブロックの名前空間と sObject の名前空間が異なる場合のみ、キーに名前空間が含まれます。たとえば、対応付けを生成するコードブロックが名前空間 N1 にあり、sObject も N1 にある場合、関連付け内のキーはとして表されます。ただし、コードブロックが名前空間 N1 にあり、sObject が名前空間 N2 にある場合、キーはです。

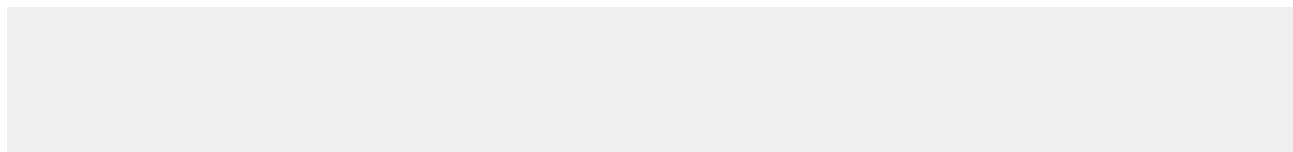
標準 sObject には名前空間プレフィックスがありません。

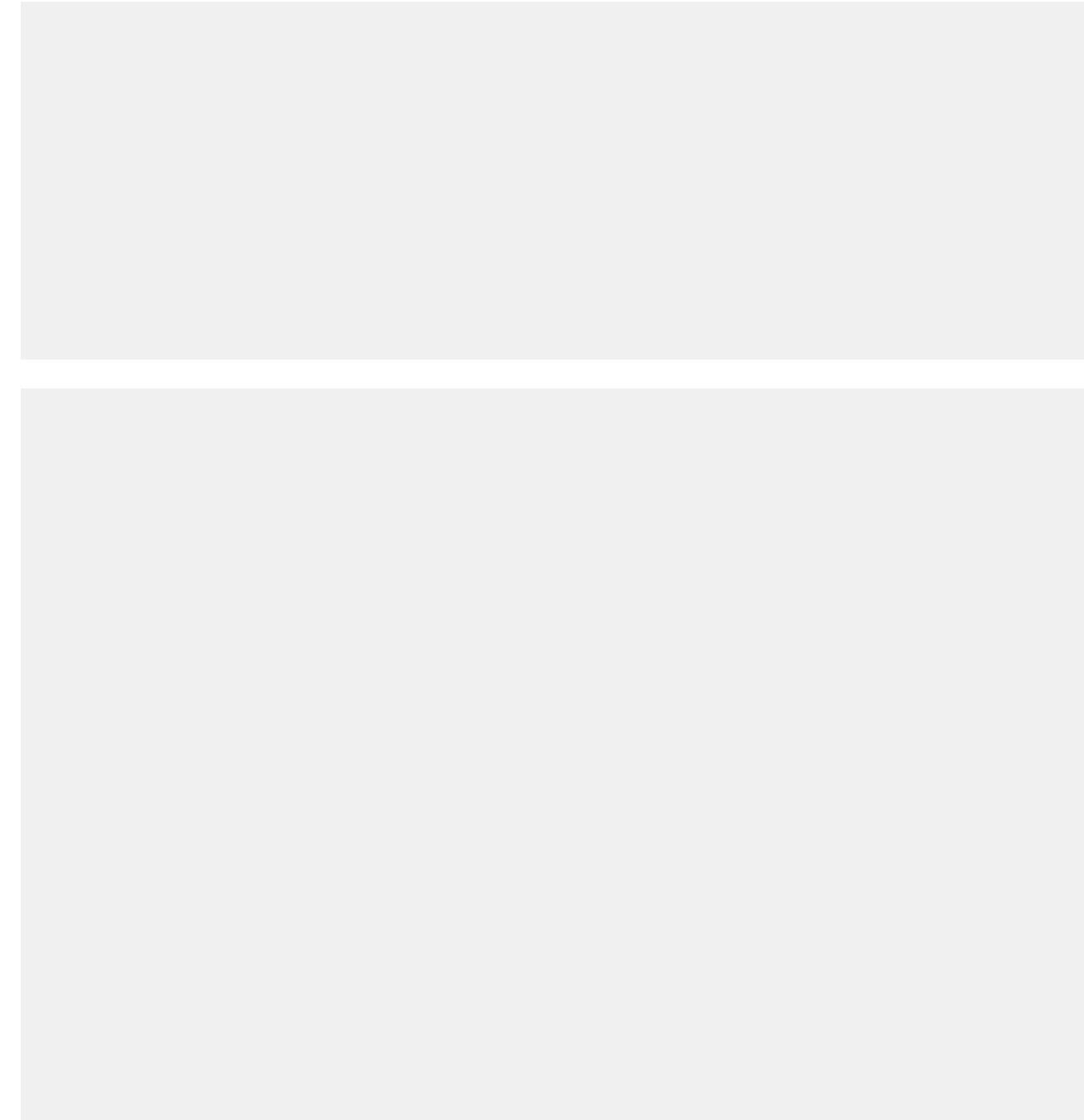


メモ: インストールされた管理パッケージからメソッドをコールした場合は、インストール先の組織で Chatter が有効になっていなくても、NewsFeed や UserProfileFeed など、Chatter sObject の sObject 名およびトークンが返されます。インストールされた管理パッケージ内にないクラスからメソッドをコールした場合は、この限りではありません。

## sObject の動的作成

sObject トークンクラスのメソッドをコールすることによって、実行時に型を決定する sObject を作成できます。次の例では、メソッドを使用して sObject 型名に対応する sObject トークンを取得する方法を示します。その後で、メソッドによって sObject のインスタンスが作成されます。この例にも、取引先の動的作成を検証するテストメソッドが含まれます。





## sObject のすべての Field Describe Result へのアクセス

Field Describe Result の `getDescribe()` メソッドを使用して、sObject のすべての項目名 (キー) と項目トーカン (値) 間のリレーションを表す対応付けを返します。

次の例では、項目に名前でアクセスするときに使用できる対応付けを生成します。



メモ: この対応付けの値のデータ型は、Field Describe Result ではありません。Describe Result を使用すると、システムリソースが過剰に使用されます。代わりに、該当する項目の検索に使用できるトークンの対応付けを使用します。項目を決定したら、その項目の Describe Result を生成します。

対応付けには次の特性があります。

- ・ 動的である。つまり、sObject の項目で実行時に生成されます。
- ・ すべての項目名は大文字と小文字を区別しない。
- ・ キーは、必要に応じて名前空間を使用する。
- ・ キーは、項目がカスタムオブジェクトかどうかを反映する。

たとえば、対応付けを生成するコードブロックが名前空間 N1 にあり、項目も N1 にある場合、関連付け内のキーは \_\_\_\_\_ として表されます。ただし、コードブロックが名前空間 N1 にあり、項目が名前空間 N2 にある場合、キーは \_\_\_\_\_ です。

また、標準項目には名前空間プレフィックスがありません。



メモ: インストールされた管理パッケージ内で Field Describe を実行した場合、インストール先の組織で Chatter が有効になっていなくても、Chatter 項目が返されます。インストールされた管理パッケージ内にないクラスから Field Describe を実行した場合は、Chatter 項目は返されません。

## sObject に関連付けられたすべてのデータカテゴリへのアクセス

メソッドおよび メソッドを使用して、特定のオブジェクトに関連付けられたカテゴリを返します。

1. 選択したオブジェクトに関連付けられたすべてのカテゴリグループを返します（「[describeDataCategory Groups](#)」（ページ 518）を参照）。
2. 返された対応付けから、詳細に検索するカテゴリグループ名と sObject 名を取得します（「[Schema.Describe DataCategoryGroupResult](#)」（ページ 520）を参照）。
3. カテゴリグループおよび関連付けられたオブジェクトを指定し、このオブジェクトに使用できるカテゴリを取得します（「[describeDataCategory GroupStructures](#)」（ページ 519）を参照）。

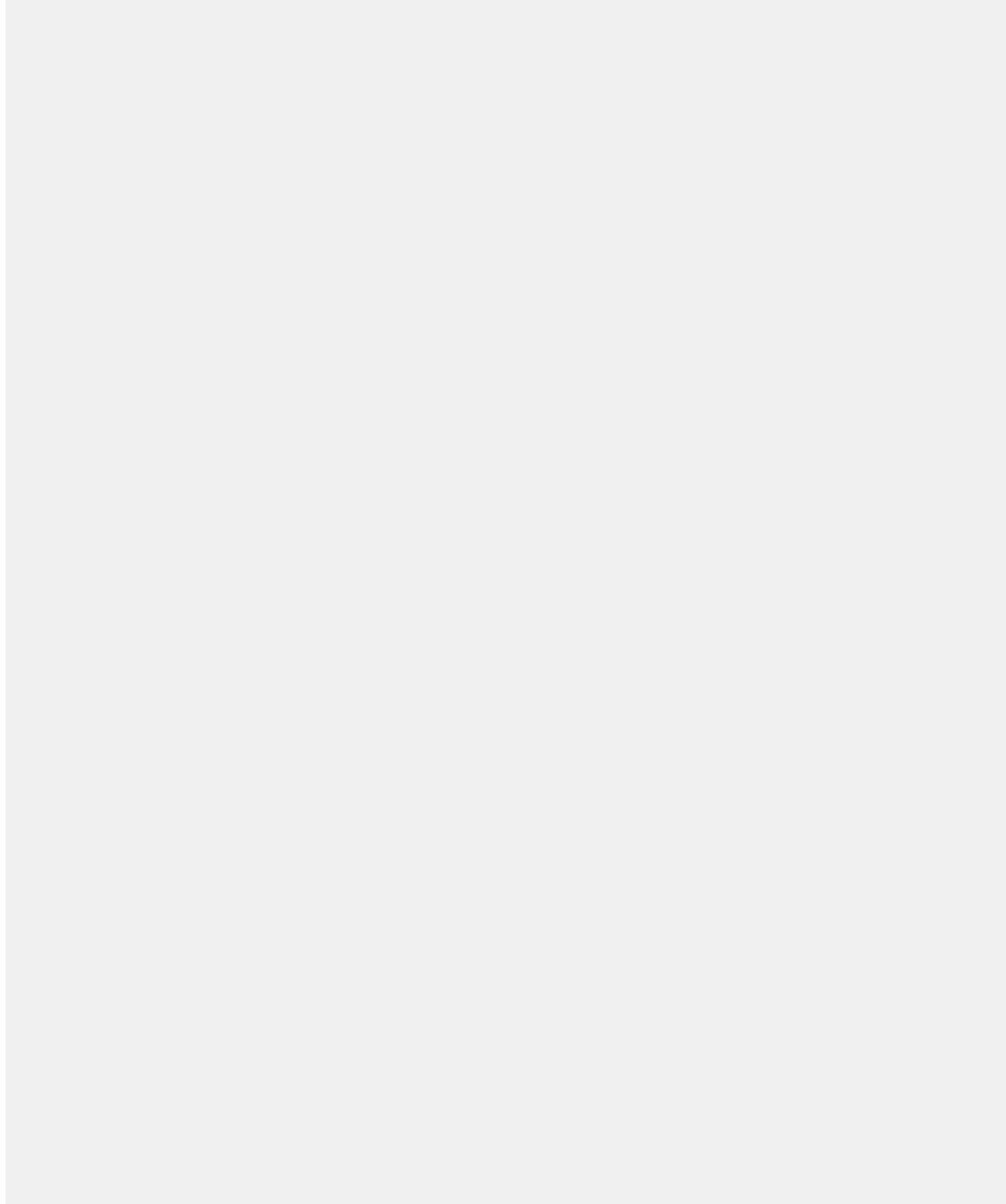
メソッドは、指定したカテゴリグループのオブジェクトに使用できるカテゴリを返します。データカテゴリについての詳細は、Salesforce オンラインヘルプの「データカテゴリとは？」を参照してください。

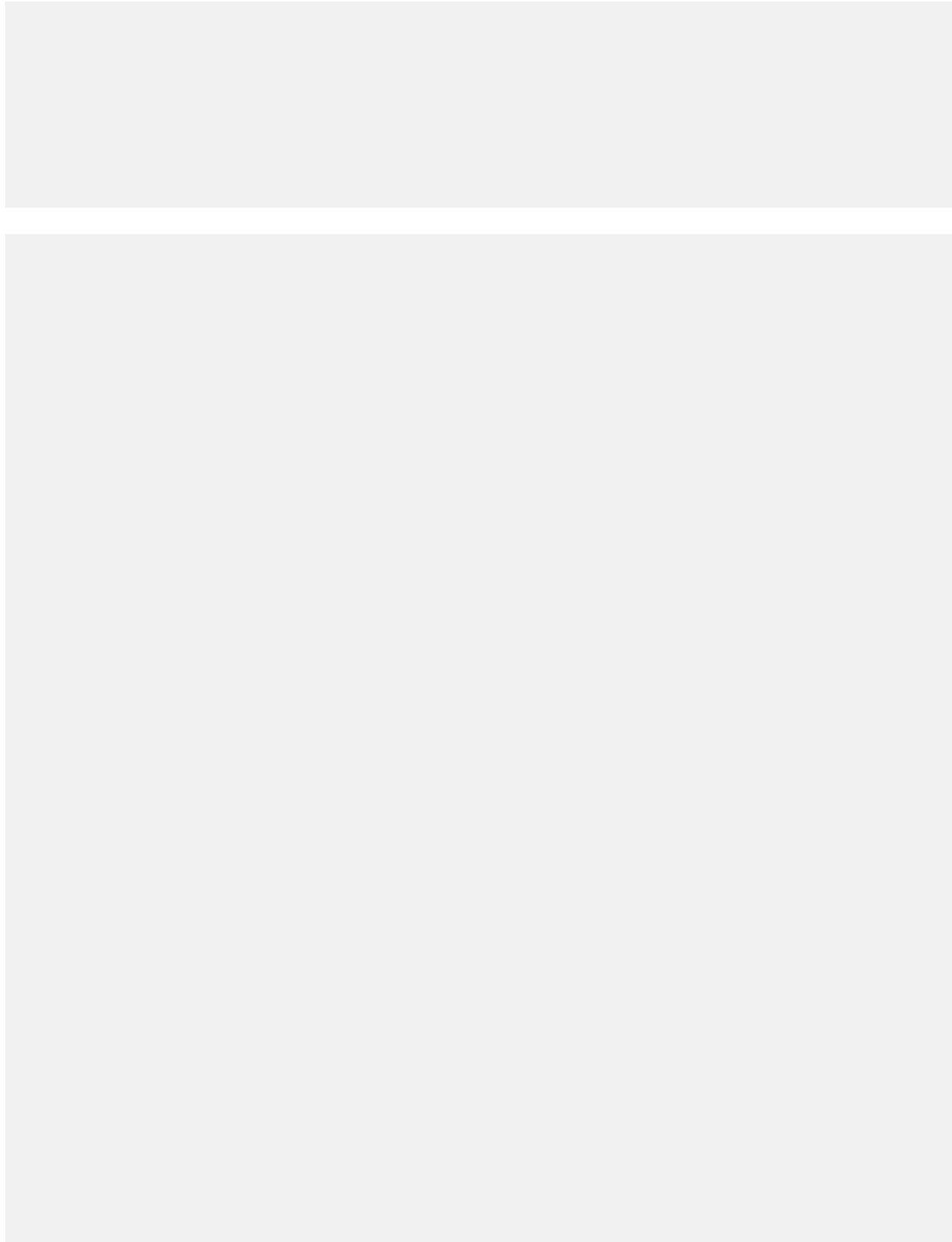
次の例では、  
メソッドは、Article オブジェクトおよび Question オブジェクトに関連付けられたすべてのカテゴリグループを返します。  
メソッドは、領域カテゴリグループの記事および質問に使用できるすべてのカテゴリを返します。記事および質問についての詳細は、Salesforce オンラインヘルプの「記事と翻訳の管理」および「アンサーの概要」を参照してください。

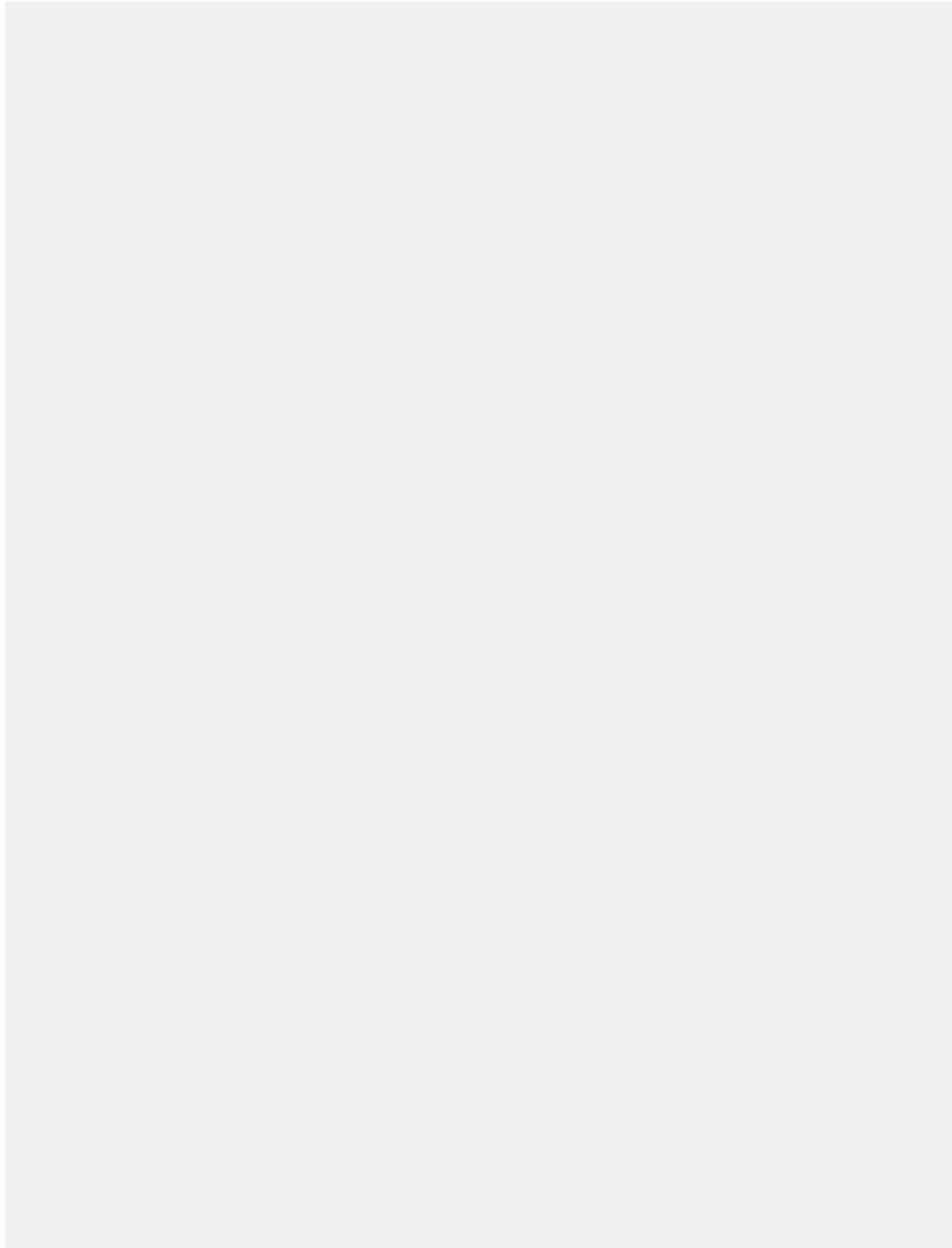
次の例を使用するには、次を行う必要があります。

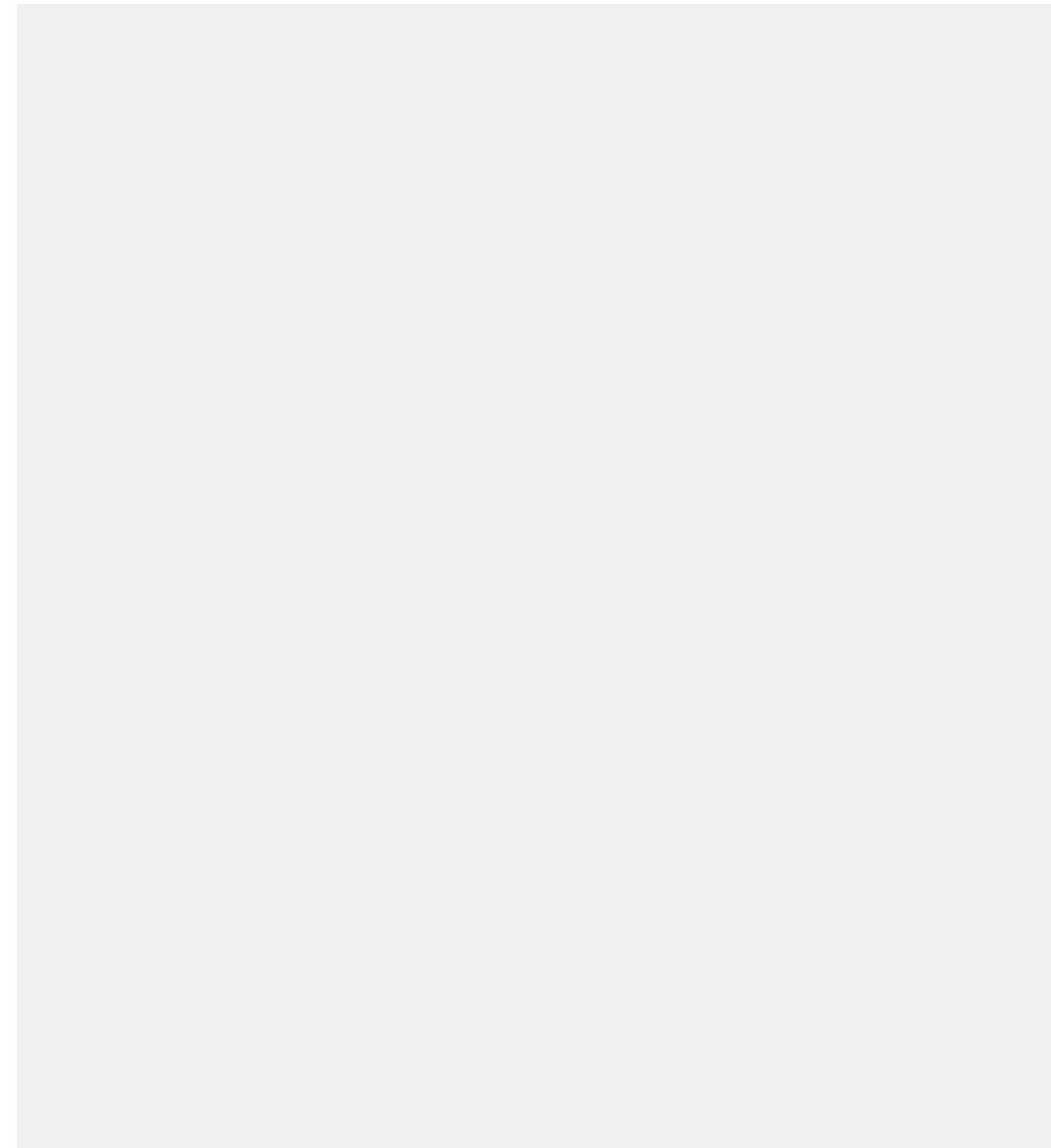
- ・ Salesforce ナレッジを有効化する。
- ・ アンサー機能を有効化する。
- ・ 領域というデータカテゴリグループを作成する。
- ・ 領域をアンサーで使用するデータカテゴリグループとして割り当てる。
- ・ 領域データカテゴリグループが Salesforce ナレッジに割り当てられていることを確認する。

データカテゴリグループの作成についての詳細は、Salesforce オンラインヘルプの「カテゴリグループの作成と変更」を参照してください。アンサーについての詳細は、Salesforce オンラインヘルプの「アンサーの概要」を参照してください。



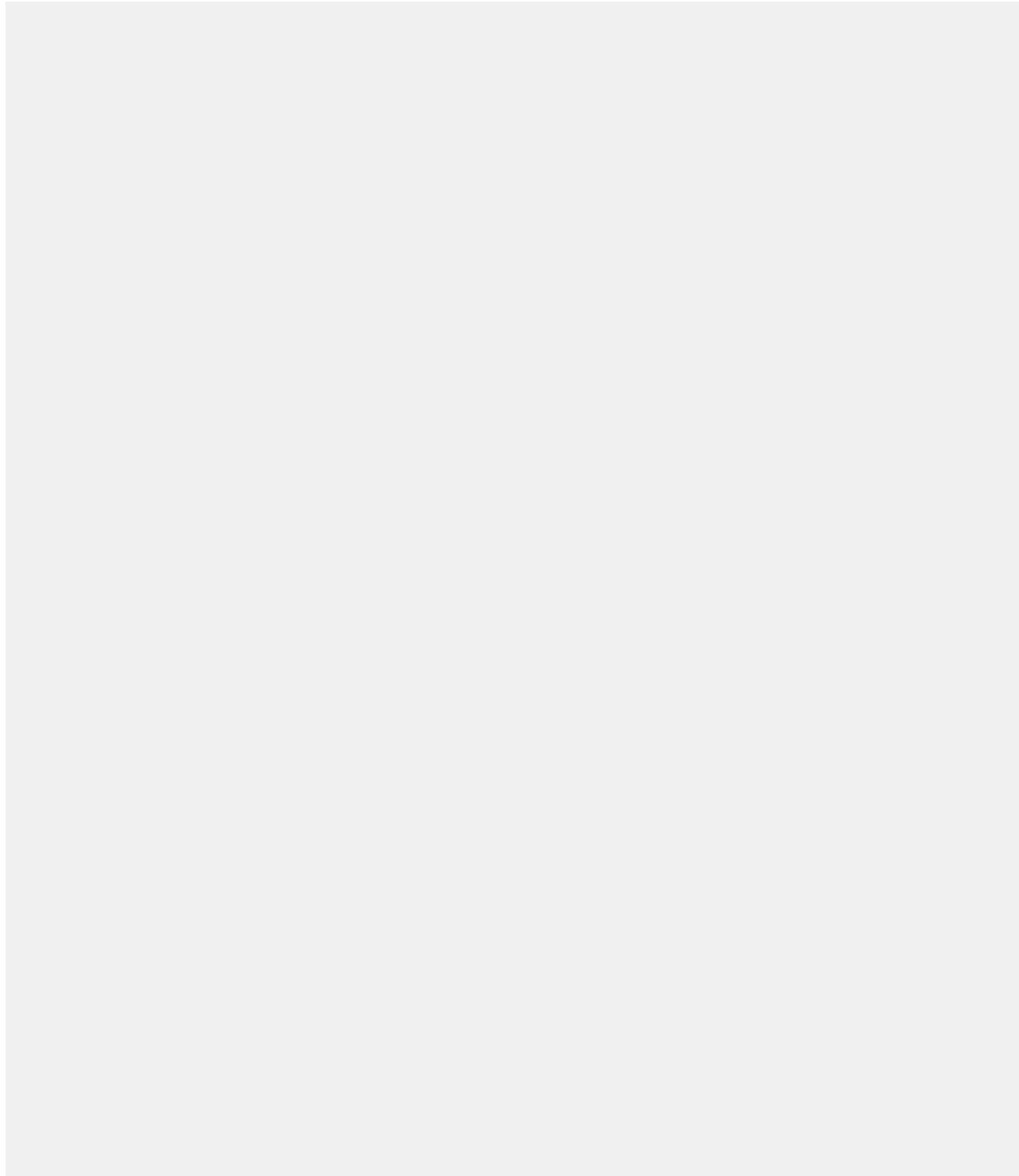




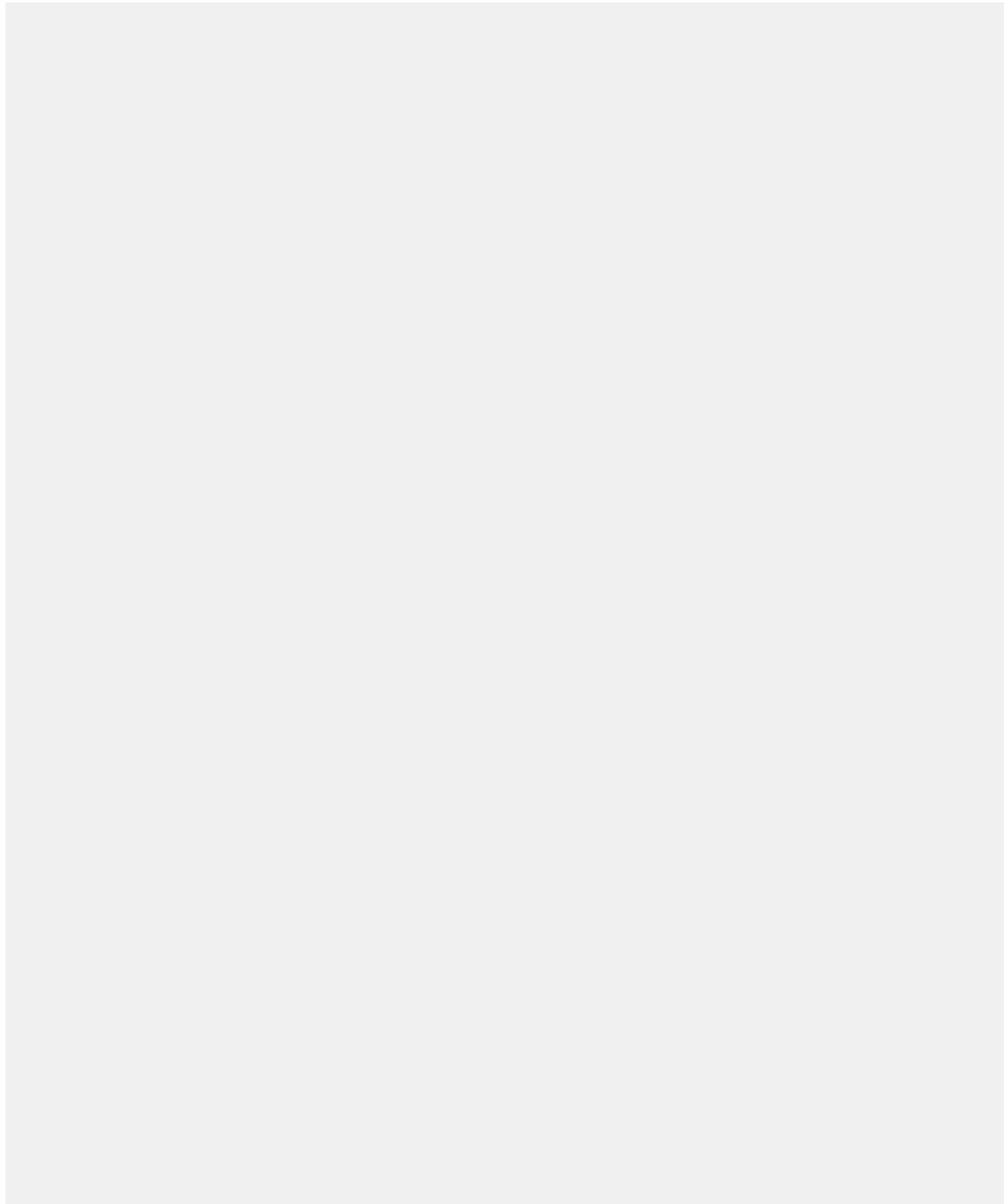


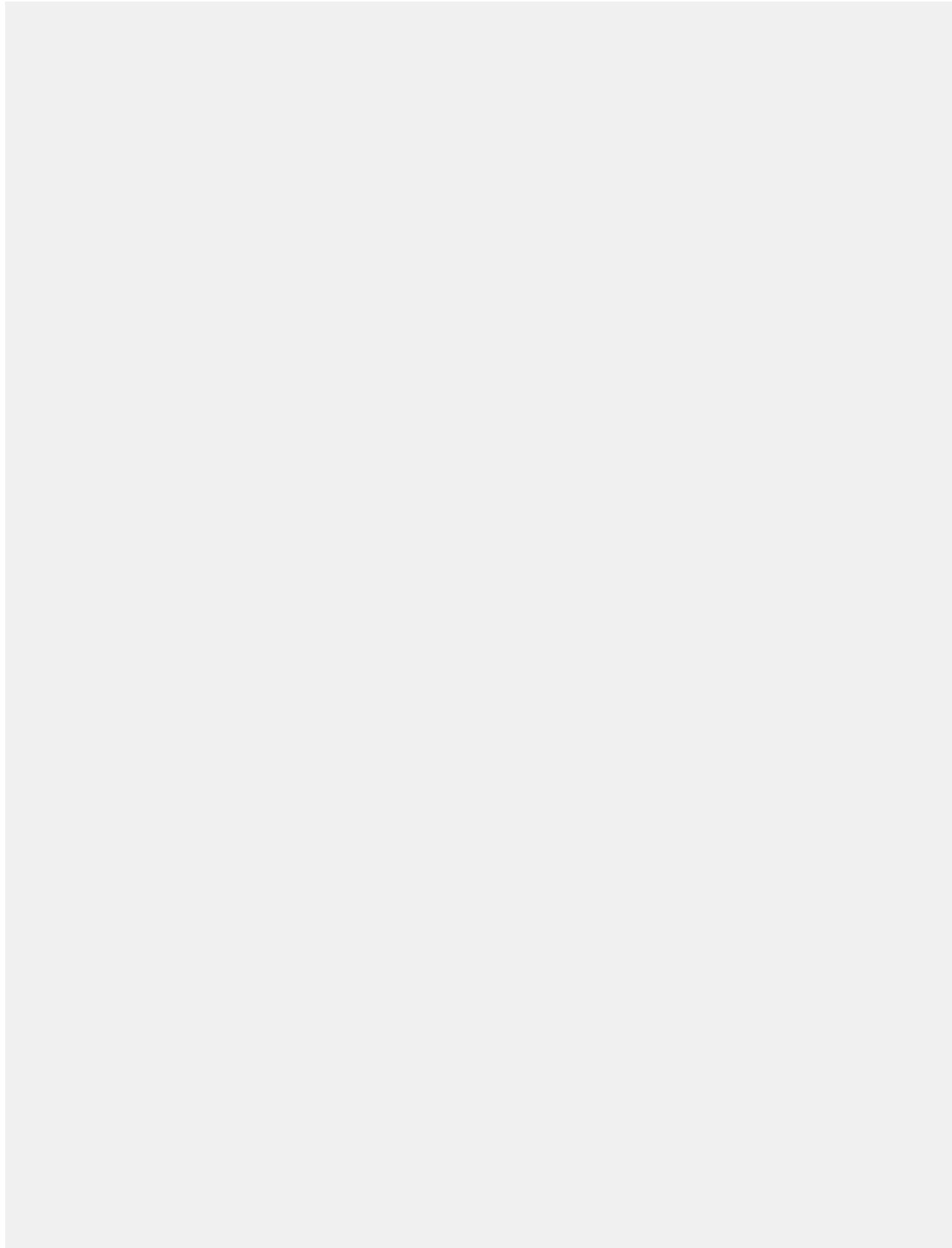
### sObject に関連付けられたすべてのデータカテゴリへのアクセスのテスト

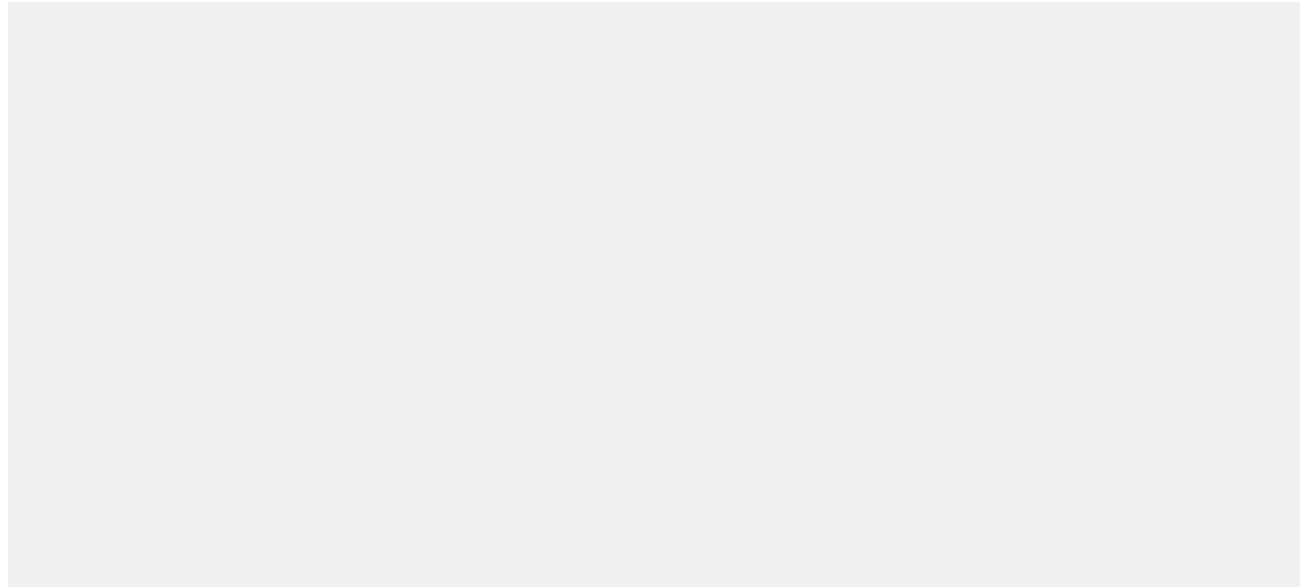
次の例では、「[sObject に関連付けられたすべてのデータカテゴリへのアクセス](#)」で示すメソッドをテストします。返されたカテゴリグループおよび関連付けられたオブジェクトが正しいことを確認できます。



この例では、「`sObject` に関連付けられたすべてのデータカテゴリへのアクセス」で示すメソッドをテストします。返されたカテゴリグループ、カテゴリ、および関連付けられたオブジェクトが正しいことを確認できます。







## 動的 SOQL

動的 SOQL は、Apex コードを使用して、実行時に SOQL 文字列の作成を参照します。動的 SOQL によって、さらに柔軟なアプリケーションの作成が可能になります。たとえば、エンドユーザーの入力に基づいた検索を作成したり、さまざまな項目名のレコードを更新したりできます。

実行時に動的 SOQL クエリを作成するには、次のいずれかの方法で `database` メソッドを使用します。

- クエリが 1 つのレコードを返すときに、1 つの `sObject` を返します。

`string_limit_1`

- クエリが複数のレコードを返すときに、`sObject` のリストを返します。

`string`

通常の割り当てステートメントやループなど、インライン SOQL クエリが使用可能な場合はいつでも、`database` メソッドを使用できます。結果は、静的 SOQL クエリの処理とほぼ同様の方法で処理されます。

動的 SOQL 結果は、`Account` や `MyCustomObject__c`、または汎用 `sObject` データ型などのように、具体的な `sObject` として指定できます。実行時に、システムは、クエリのタイプが宣言された変数の型と一致しているかどうか検証します。クエリが正しい `sObject` データ型を返さない場合、ランタイムエラーが発生します。これは、汎用 `sObject` から具体的な `sObject` を割り当てる必要がないことを意味します。

動的 SOQL クエリには、静的クエリと同じガバナ制限があります。ガバナ制限についての詳細は、「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

SOQL クエリの構文の詳細は、『*Force.com SOQL and SOSL Reference*』の「Salesforce Object Query Language (SOQL)」を参照してください。

### 動的 SOQL に関する考慮事項

動的 SOQL クエリ文字列で単純なバインド変数を使用できます。次の例は許可されます。

ただし、オンライン SOQL とは異なり、動的 SOQL は、バインド変数項目をクエリ文字列で使用できません。次の例はサポートされず、「」というエラーになります。

代わりに、次のように変数項目を文字列に解決し、その文字列を動的 SOQL クエリで使用できます。

### SOQL インジェクション

SOQL インジェクションとは、ユーザが SOQL ステートメントをあなたのコードに渡すことで、あなたのアプリケーションで意図していなかったデータベースメソッドを実行する手法です。動的 SOQL ステートメントを構築するためにアプリケーションがエンドユーザ入力に依存し、入力が適切に処理されなかった場合、常に Apex コードで発生する可能性があります。

SOQL インジェクションを防ぐには、メソッドを使用します。このメソッドは、ユーザから渡される文字列のすべての单一引用符にエスケープ文字 (\) を追加します。このメソッドにより、すべての单一引用符を、データベースコマンドではなく、囲まれた文字列として処理します。

### 動的 SOSL

動的 SOSL は、Apex コードを使用して、実行時に SOSL 文字列の作成を参照します。動的 SOSL によって、さらに柔軟なアプリケーションの作成が可能になります。たとえば、エンドユーザの入力に基づいた検索を作成したり、さまざまな項目名のレコードを更新したりできます。

実行時に動的 SOSL クエリを作成するには、search メソッドを使用します。次に例を示します。

`SOSL_search_string`

次の例では、単純な SOSL クエリ文字列を実行しています。

動的SOSLステートメントは、sObjectのリストを評価します。ここでは、各リストは特定の sObject データ型の検索結果を含みます。結果リストは常に、動的SOSL クエリで指定された順序と同じ順序で返されます。上記の例では、Account の結果が最初、次に Contact、Lead と続きます。

通常の割り当てステートメントや ループなど、インライン SOSL クエリが使用可能な場合はいつでも、search メソッドを使用できます。結果は、静的 SOSL クエリの処理とほぼ同様の方法で処理されます。

動的SOSL クエリには、静的クエリと同じガバナ制限があります。ガバナ制限についての詳細は、「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

SOSL クエリの構文の詳細は、『*Force.com SOQL and SOSL Reference*』の「[Salesforce Object Search Language \(SOSL\)](#)」を参照してください。

## SOSL インジェクション

SOSL インジェクションとは、ユーザが SOSL ステートメントをあなたのコードに渡すことで、あなたのアプリケーションで意図していなかったデータベースメソッドを実行する手法です。動的SOSLステートメントを構築するためにアプリケーションがエンドユーザ入力に依存し、入力が適切に処理されなかった場合、常に Apex コードで発生する可能性があります。

SOSL インジェクションを防ぐには、  
メソッドを使用します。このメソッドは、ユーザから渡される文字列のすべての单一引用符にエスケープ文字 (\) を追加します。このメソッドにより、すべての单一引用符を、データベースコマンドではなく、囲まれた文字列として処理します。

## 動的 DML

Describe Information をクエリし、実行時に SOQL クエリの構築を行うことができます。さらに sObject を動的に作成し、DML を使用してそれらをデータベースに挿入することもできます。

指定されたデータ型の新規 sObject を作成するには、sObject トークンで  
メソッドを使用します。  
トークンには、具体的な sObject のデータ型 (Account など) にキャストする必要があります。次に例を示します。

sObject トークン は Account のトークンですが、別々にアクセスされるため sObject とみなされます。

メソッドを使用するには、具体的な sObject のデータ型 Account にこのトークンを再度キャストする必要があります。割り当てについての詳細は、「[クラスとキャスト](#)」(ページ 213)を参照してください。

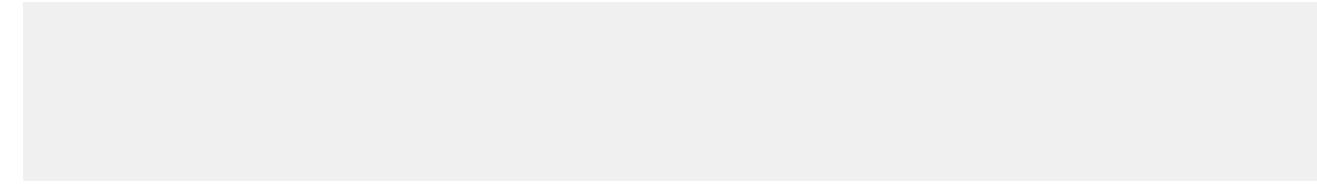
このもう 1 つの例では、

メソッドを介して sObject トークンを取得する方法を示

し、その後、トークンに対して

メソッドを使用して新しい sObject を作成します。この例にも、取

引先の動的作成を検証するテストメソッドが含まれます。



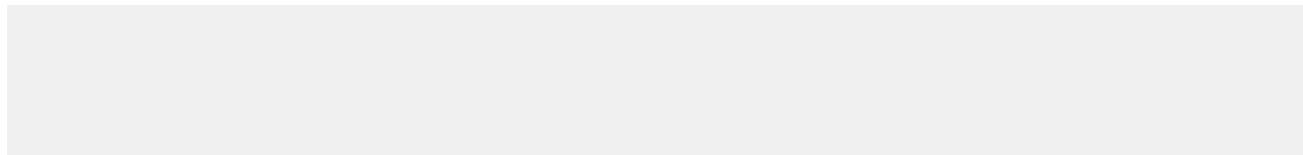
で ID を指定して、既存のレコードを参照する sObject を作成し、後でそのレコードを更新することもできます。次に例を示します。



「[Schema.sObjectType](#)」 (ページ 545) を参照してください。

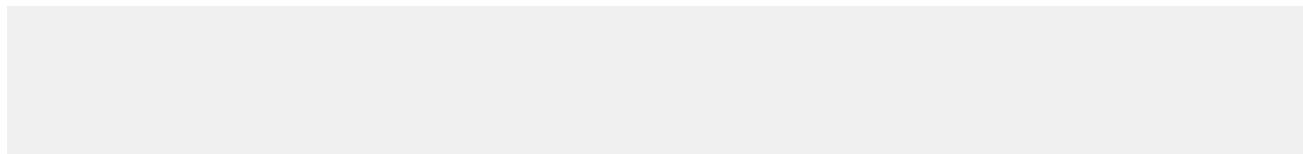
## 項目値の設定と取得

String として表される API 名、または項目トークンのいずれかを使用している項目値を設定または取得するには、オブジェクトの `set` メソッドおよび `get` メソッドを使用します。次の例では、項目 `Account` の API 名が使用されます。



次の例では、代わりに

項目のトークンを使用します。



Object スカラーデータ型は、sObject の項目値を設定または取得するために、汎用データ型として使用できます。これは、[anyType](#) 項目のデータ型と同等です。Object データ型は、sObject の汎用型として使用可能な sObject データ型とは異なります。

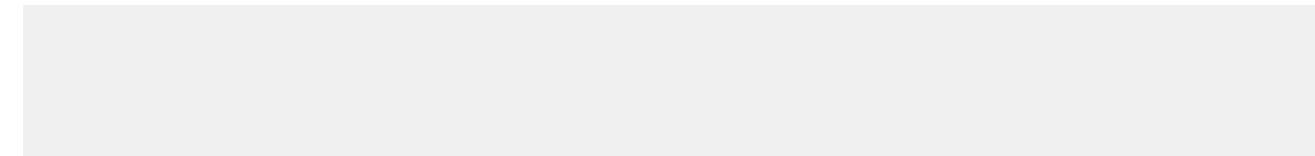


メモ: 項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存 (コンパイル) した Apex クラスとトリガにはランタイムエラーが発生します。

## 外部キーの設定と取得

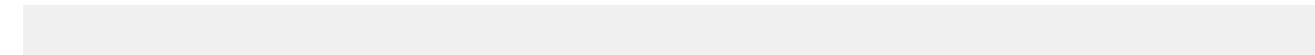
Apex は、API と同じ方法で、名前 (または外部 ID) による外部キーの入力をサポートします。外部キーのスカラ ID 値を設定または取得するには、`set` メソッドまたは `get` メソッドを使用します。

外部キーに関連付けられたレコードを設定または取得するには、**setExternalId** メソッドと **getExternalId** メソッドを使用します。これらのメソッドは、Object データ型ではなく sObject データ型で使用される必要があります。次に例を示します。



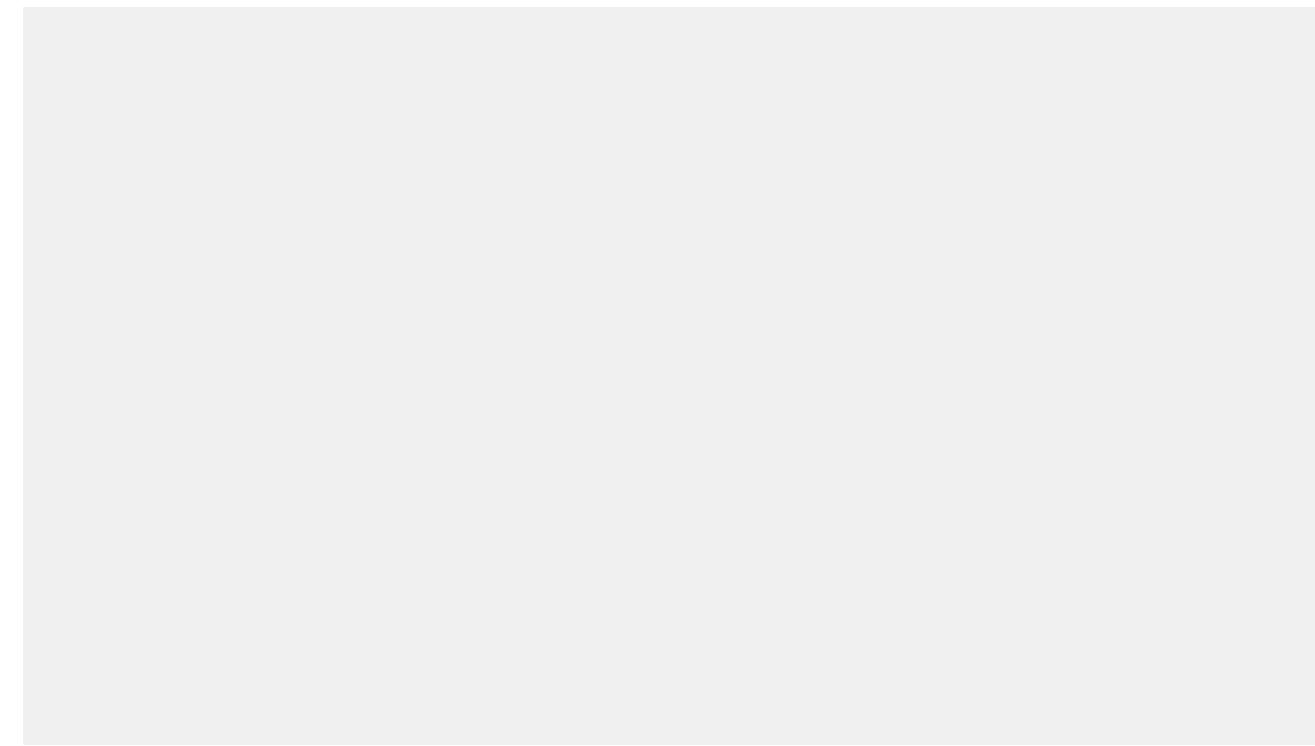
子 sObject を使用しているときに親 sObject 値の外部 ID を指定する必要はありません。親 sObject に ID を提供した場合、DML 操作によって無視されます。Apex は、常に入力した ID で親オブジェクトを返すリレーション SOQL クエリを介して、外部キーが入力されることを前提としています。ID がない場合、子オブジェクトを使用します。

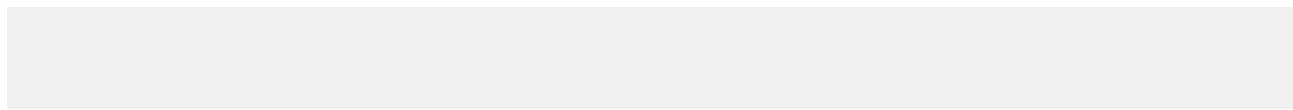
たとえば、カスタムオブジェクト C1 に、子カスタムオブジェクト C2 にリンクする外部キー **Parent\_C1** があるとします。C1 オブジェクトを作成し、値 **Parent\_C1** が割り当てられた「xxx」という名前の C2 レコードに関連付けるとします。親から子へのリレーションを介して入力されるため、「xxx」レコードの ID は不要です。次に例を示します。



の ID に値を割り当てる場合、その値は無視されます。ID がない場合、レコードではなくオブジェクト( )に割り当てます。

動的 Apex を使用して外部キーにアクセスすることもできます。次の例は、動的 Apex を使用して、親-子リレーションのサブクエリから値を取得する方法を示しています。





## 第7章

### Apex の一括処理

#### トピック:

- [Apex の一括処理の使用](#)
- [Apex による共有管理について](#)

開発者は Apex の一括処理を使用して、Force.com プラットフォームで長時間にわたり実行される複雑なプロセスを構築できるようになります。たとえば、特定の日付を過ぎたレコードを検索してアーカイブに追加する、夜間に実行されるアーカイブソリューションを構築できます。または、毎晩すべての取引先と商談を探索し、カスタム条件に基づいて必要に応じて更新するデータの整理処理を構築できます。

Apex の一括処理は、インターフェースとして公開され、開発者によって実行される必要があります。一括処理ジョブは実行時に Apex を使用してプログラムで起動できます。

一度に実行できるキュー内または有効な一括処理ジョブは 5 件のみです。Salesforce の [スケジュール済みジョブ] ページを表示するか、プログラムで SOAP API を使用してオブジェクトをクエリすることで、現在のジョブ件数を確認できます。



警告: 一括処理ジョブをトリガから開始する場合は、細心の注意を払ってください。制限の 5 件を超える一括処理ジョブがトリガで追加されないようにする必要があります。特に、API の一括更新、インポートウィザード、ユーザインターフェースを使用したレコードの一括変更、および複数のレコードを一度に更新するすべての処理については十分に考慮してください。

また、一括処理ジョブは [Apex スケジューラ](#)を使用して、プログラムで特定の時間に実行されるようにスケジュールしたり、Salesforce ユーザインターフェースの [Apex をスケジュール] ページを使用してスケジュールしたりすることもできます。[Apex をスケジュール] ページについての詳細は、Salesforce オンラインヘルプの「Apex のスケジュール」を参照してください。

Apex の一括処理インターフェースは、[Apex による共有管理の再適用](#)にも使用されます。

一括処理ジョブの詳細は、[Apex の一括処理の使用 \(ページ 273\)](#)を参照してください。

Apex による共有管理についての詳細は、「[Apex による共有管理について](#)」(ページ 288)を参照してください。

## Apex の一括処理の使用

Apex の一括処理を使用するには、Salesforce が提供するインターフェースクラスを記述し、次にプログラムでクラスを呼び出す必要があります。

を実装する Apex

Apex の一括処理ジョブの実行を監視または停止するには、[設定] から [監視] > [Apex ジョブ] または [ジョブ] > [Apex ジョブ] をクリックします。

### Database.Batchable インターフェースの実装

インターフェースには、実装が必要な次の 3 つのメソッドが含まれています。

- メソッド

bc

メソッドは、Apex の一括処理ジョブの開始時にコールされます。メソッドは、インターフェースメソッドに渡すレコードまたはオブジェクトを収集するために使用します。このメソッドは、オブジェクトまたは、ジョブに渡すレコードやオブジェクトを含む Iterable オブジェクトを返します。

オブジェクトは、一括処理ジョブで使用するオブジェクトの範囲を単純なクエリ( )で作成する場合に使用します。queryLocator オブジェクトを使用する場合、SOQL クエリによって取得されるレコード合計数に対するガバナ制限は無視されます。たとえば、Account オブジェクトに対する Apex の一括処理ジョブでは、組織内のすべての取引先レコード(最大 5000 万件のレコード)のを返すことができます。また、Contact オブジェクトに対して共有再適用を行うと、組織内のすべての取引先レコードのが返されます。

Iterable オブジェクトは、一括処理ジョブに複雑な範囲を作成する必要がある場合に使用します。また、リスト全体を反復する独自のカスタムプロセスを作成するために Iterable オブジェクトを使用することもできます。

**!** 重要: Iterable オブジェクトを使用する場合、SOQL クエリによって取得されるレコード合計数に対するガバナ制限はそのまま適用されます。

- メソッド

BC

メソッドは、メソッドに渡す一括レコードごとにコールされます。データの処理単位ごとに必要な処理をすべて実行する場合に、このメソッドを使用します。

このメソッドは次を取得します。

- オブジェクトへの参照。
- などの sObjects のリストまたはパラメータ化された型のリスト。  
を使用している場合は、返されたリストが使用されます。

レコードの一括処理は、メソッドから受け取る順序で実行されることを保証されません。

- メソッド

BC

メソッドは、すべての一括処理が行われた後にコールされます。確認メールの送信や後処理操作を行う場合に、このメソッドを使用します。

Apex の一括処理ジョブの各実行は、個別のトランザクションとみなされます。たとえば、1,000 件のレコードを含む Apex の一括処理ジョブが、から任意の *scope* パラメータを指定せずに実行されると、このジョブはそれぞれ 200 件のレコードを含む 5 つのトランザクションとみなされます。Apex のガバナ制限は、各トランザクションでリセットされます。最初のトランザクションが成功し、2 番目が失敗した場合、最初のトランザクションで行われたデータベースの更新はロールバックされません。

## Database.BatchableContext の使用

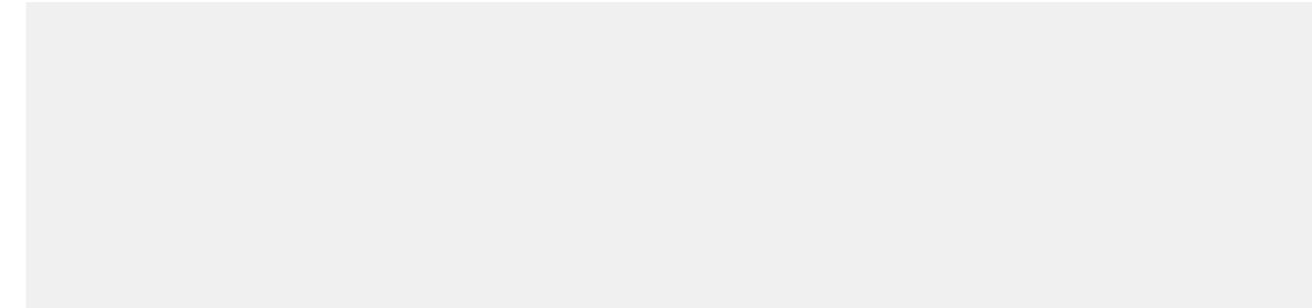
インターフェースのすべてのメソッドは  
オブジェクトへの参照を必要とします。このオブジェクトは、一括処理ジョブの進行状況を追跡するために使用します。

オブジェクトのインスタンスマソッドを次に示します。

名前	引数	戻り値	説明
	ID		この一括処理ジョブに関連付けられている <a href="#">AsyncApexJob</a> オブジェクトの ID を文字列として返します。このメソッドは、一括処理ジョブのレコードの進行状況を追跡するために使用します。 メソッドでもこの ID を使用できます。

次の例では、  
をクエリします。

を使用して、一括処理ジョブに関連付けられている



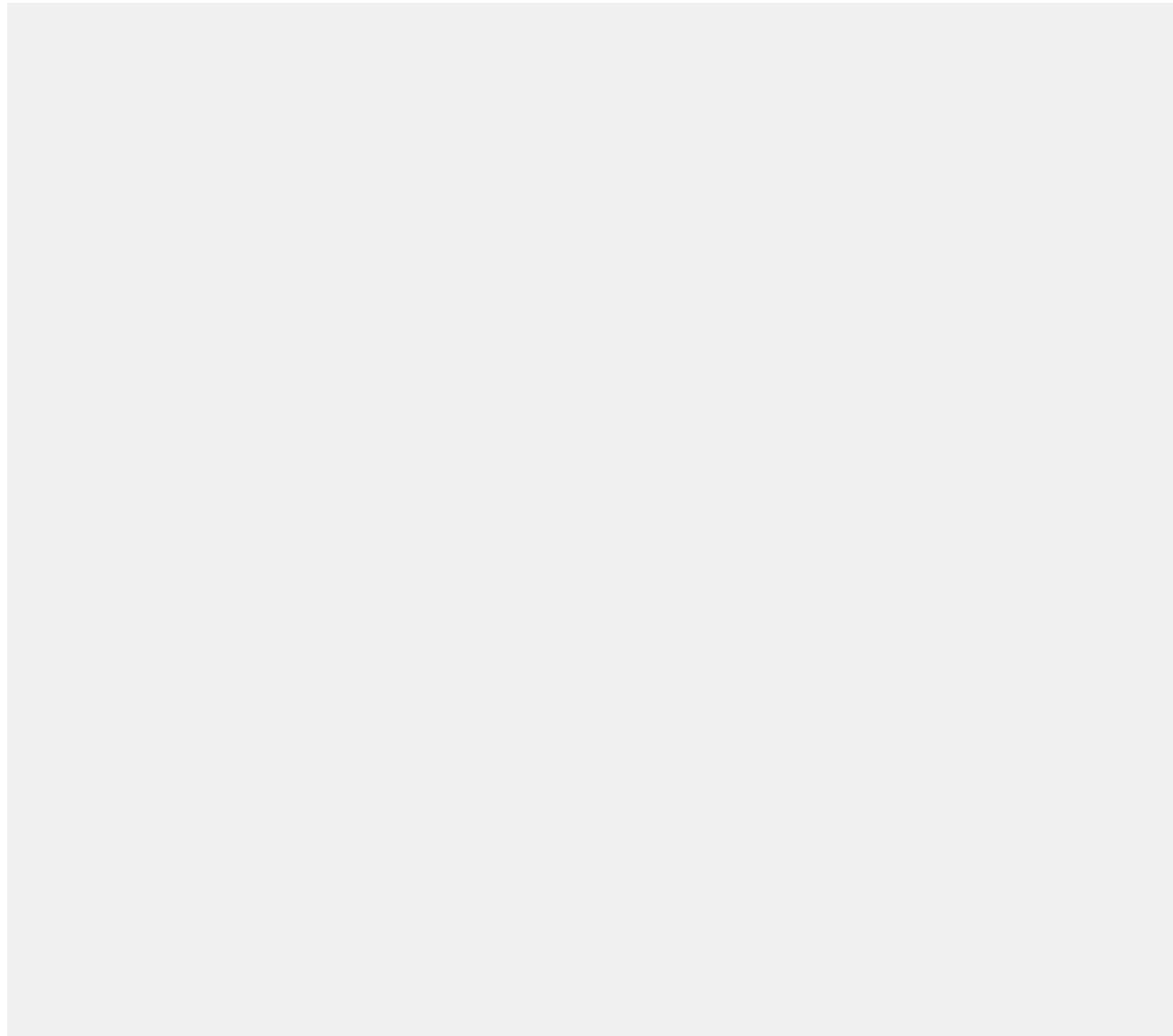
### Database.QueryLocator を使用した範囲の定義

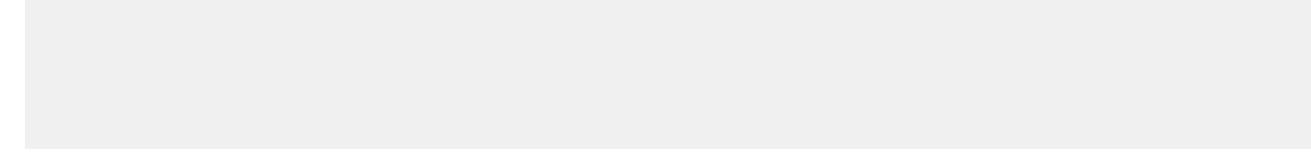
メソッドは、一括処理ジョブで使用するレコードを含む Iterable オブジェクトを返します。

次の例では、

メソッドを使用します。

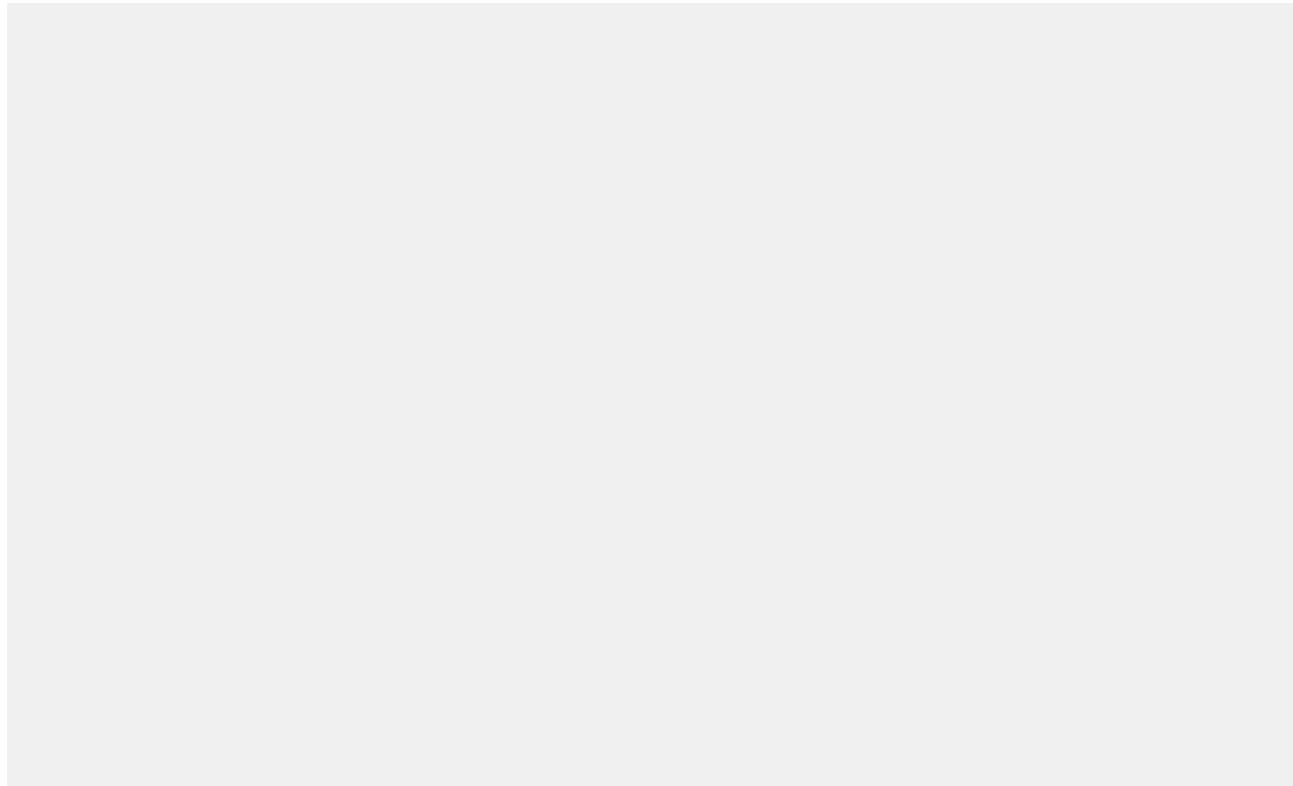
オブジェクトまたは





## Apex の一括処理に Iterable オブジェクトを使用した範囲の定義

メソッドは、一括処理ジョブで使用するレコードを含む Iterable オブジェクトを返します。Iterable オブジェクトを使用すると、より簡単に項目を返すことができます。



## Database.executeBatch メソッドの使用

一括処理ジョブをプログラムで開始するには、メソッドを使用します。



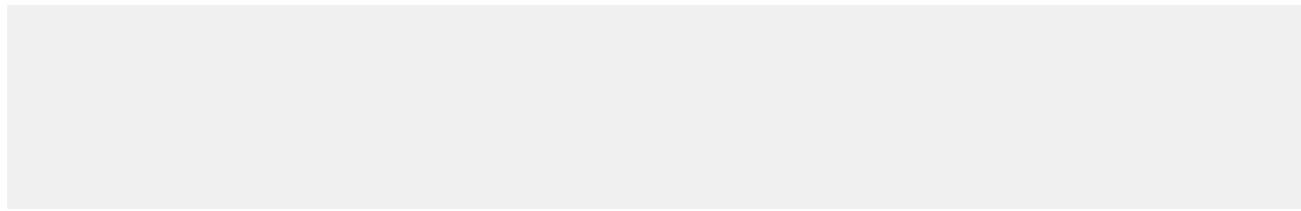
**重要:** をコールしたときに Salesforce が行うのは、そのプロセスをキューに追加することのみです。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。

メソッドは次の 2 つのパラメータを取ります。

- インターフェースを実装するクラスのインスタンス。
- メソッドは省略可能なパラメータ *scope* を取ります。このパラメータは、メソッドに渡すレコードの数を指定します。このパラメータは、メソッドに渡す各レコードに対して多数の処理があり、ガバナ制限に達する場合に使用します。レコード数を制限することによって、トランザクションあたりの処理が制限されます。この値は 0 より大きくする必要があります。メソッドが QueryLocator を返す場合、  
の省略可能な範囲パラメータには最大値 2,000 を指定できます。これ

より大きい値に設定すると、Salesforce では、QueryLocator が返すレコードを、最大 2,000 レコードまでの、より小さいバッチに分割します。メソッドが Iterable オブジェクトを返す場合、範囲パラメータ値に上限はありませんが、非常に大きい値を使用すると、他の制限が発生する場合があります。

メソッドは、ジョブの進捗状況の追跡に使用できる AsyncApexJob オブジェクトの ID を返します。次に例を示します。



AsyncApexJob オブジェクトについての詳細は、『*Object Reference for Salesforce and Force.com*』の「[AsyncApexJob](#)」を参照してください。

メソッドでもこの ID を使用できます。

### `System.scheduleBatch` メソッドの使用

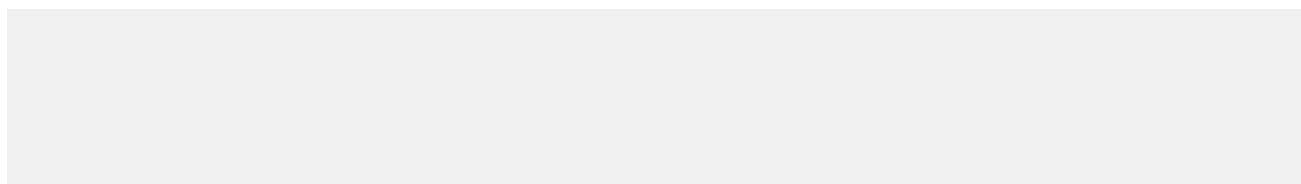
メソッドを使用して、一括処理ジョブを将来のある時点で一度実行するようにスケジュールできます。

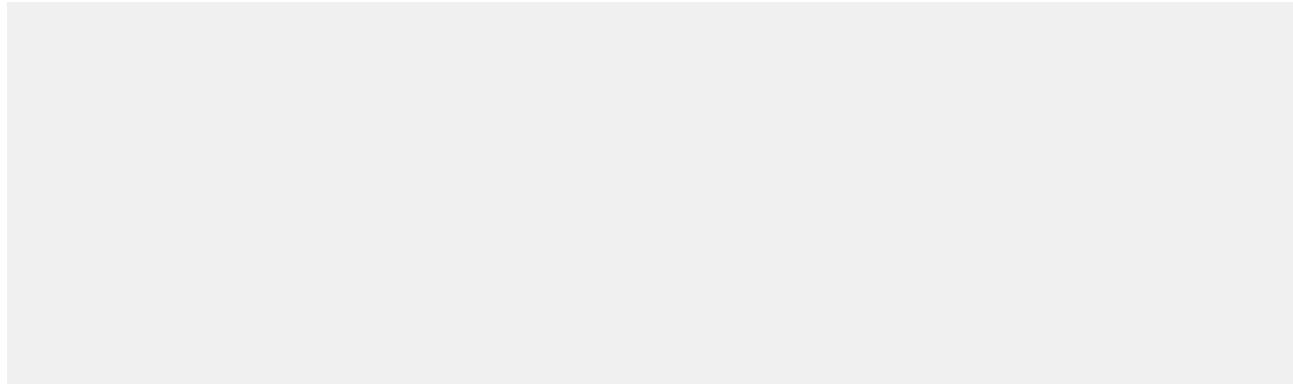
メソッドに、次のパラメータを指定します。

- インターフェースを実装するクラスのインスタンス。
- ジョブ名。
- ジョブを実行開始するまでの分単位の期間。
- 範囲の値(省略可能)。このパラメータは、メソッドに渡すレコードの数を指定します。このパラメータは、メソッドに渡す各レコードに対して多数の処理があり、ガバナ制限に達する場合に使用します。レコード数を制限することによって、トランザクションあたりの処理が制限されます。この値は 0 より大きくする必要があります。メソッドが QueryLocator を返す場合、の省略可能な範囲パラメータには最大値 2,000 を指定できます。これより大きい値に設定すると、Salesforce では、QueryLocator が返すレコードを、最大 2,000 レコードまでの、より小さいバッチに分割します。メソッドが Iterable オブジェクトを返す場合、scope パラメータ値に上限はありませんが、非常に大きい値を使用すると、他の制限が発生する場合があります。

メソッドは、スケジュール済みジョブ ID (CronTrigger ID) を返します。

次の例では、をコールして、今から 1 分後に一括処理ジョブを実行するようにスケジュールします。この例では、一括処理クラスのインスタンス (変数)、ジョブ名、期間 (1 分) をこのメソッドに渡します。省略可能な scope パラメータは指定していません。メソッドのコールから、スケジュール済みジョブ ID が返されます。この ID は、CronTrigger をクエリして、対応するスケジュール済みジョブの状況を取得するために使用されます。





CronTriggerについての詳細は、『*Salesforce および Force.com のオブジェクトリファレンス*』の「CronTrigger」を参照してください。



メモ:

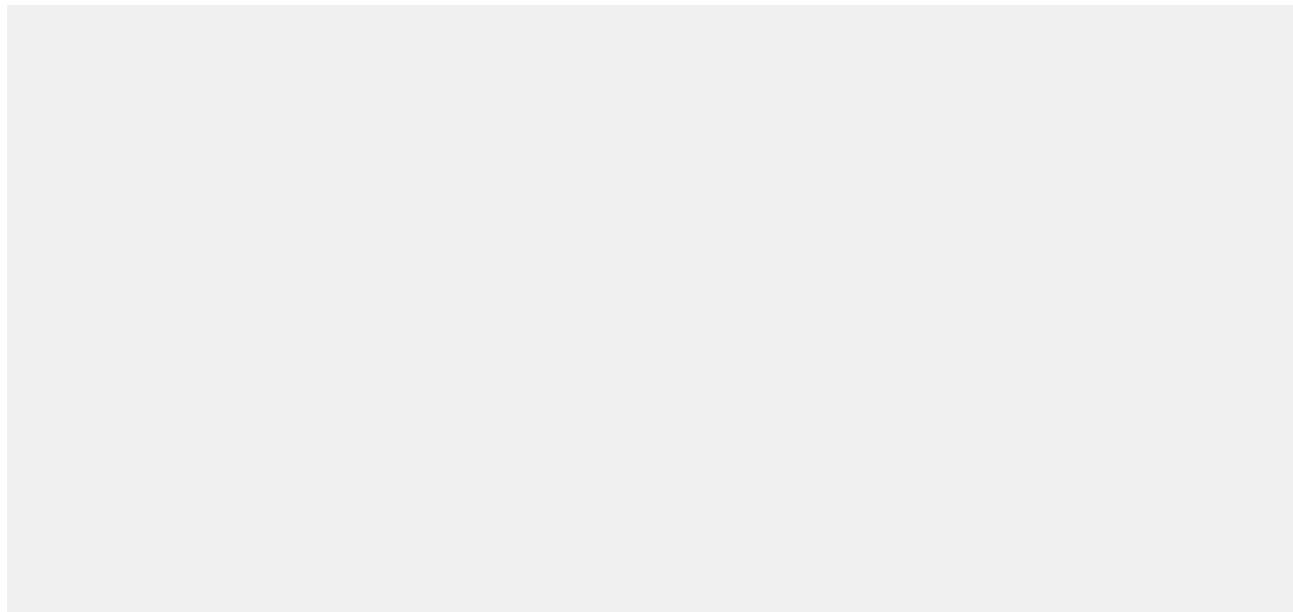
では、次の点に留意してください。

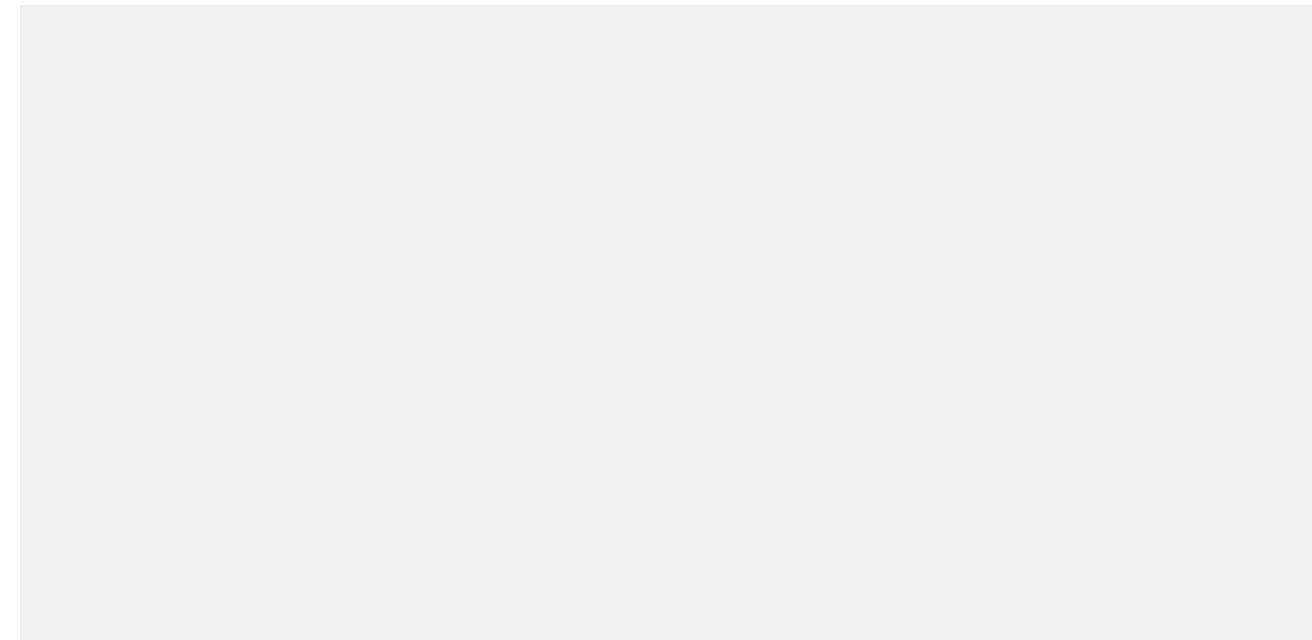
- ・ をコールすると、Salesforceにより指定の時間にジョブを実行するようにスケジュールされます。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。
- ・ スケジューラは、システムとして実行されます。ユーザがそのクラスの実行権限を持っているかどうかにかかわらず、すべてのクラスが実行されます。
- ・ スケジュールされたすべての Apex 制限は、 を使用してスケジュールされた一括処理ジョブに適用されます。一括処理ジョブの実行が開始すると、すべての一括処理ジョブ制限が適用され、ジョブはスケジュール済みの Apex 制限としてカウントされなくなります。
- ・ このメソッドをコールしてから一括処理ジョブが開始するまでは、返されたスケジュール済みジョブ ID を使用して、 メソッドを使用したスケジュール済みジョブを中止できます。

## Apex の一括処理の例

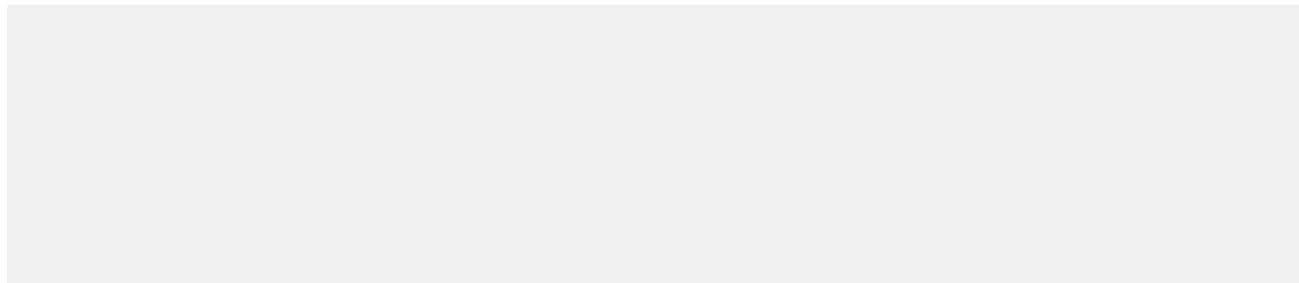
次の例では、

メソッドを使用します。

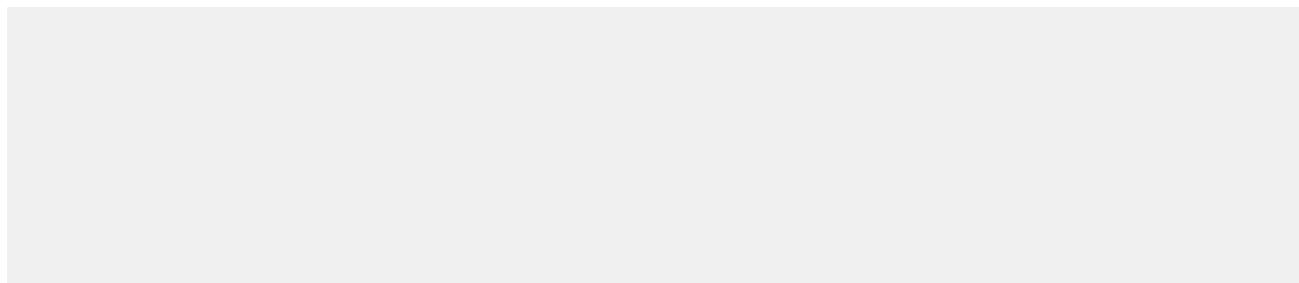




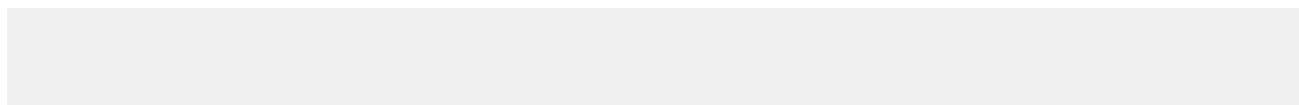
次のコードを使用して、上記のクラスをコールできます。

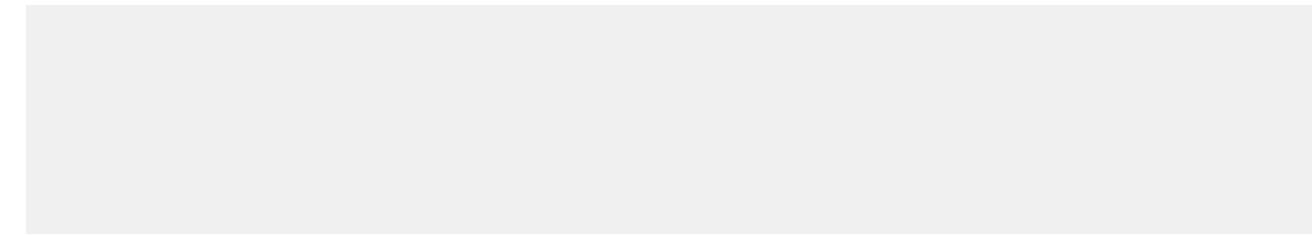


削除されてまだごみ箱に入っている取引先を除外するには、この変更したサンプルのクエリのように、SOQL クエリの WHERE 句に  
を付加します。

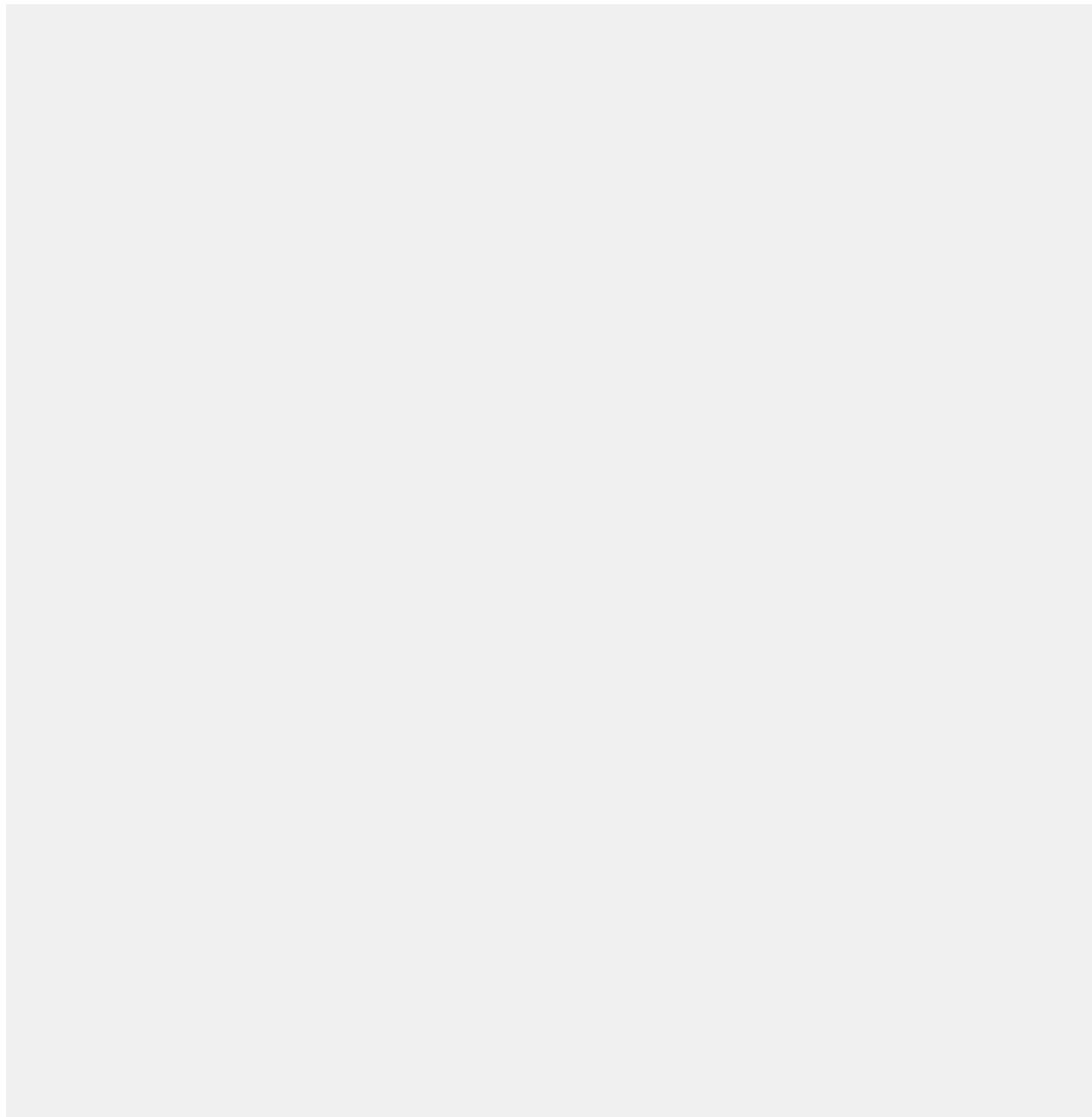


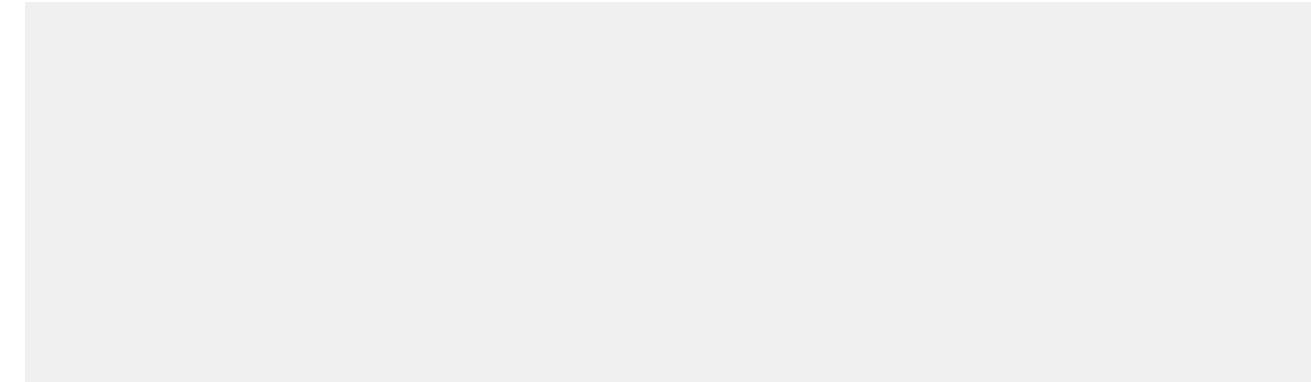
削除されてまだごみ箱に入っている請求書を除外するには、この変更したサンプルのクエリのように、SOQL クエリの WHERE 句に  
を付加します。





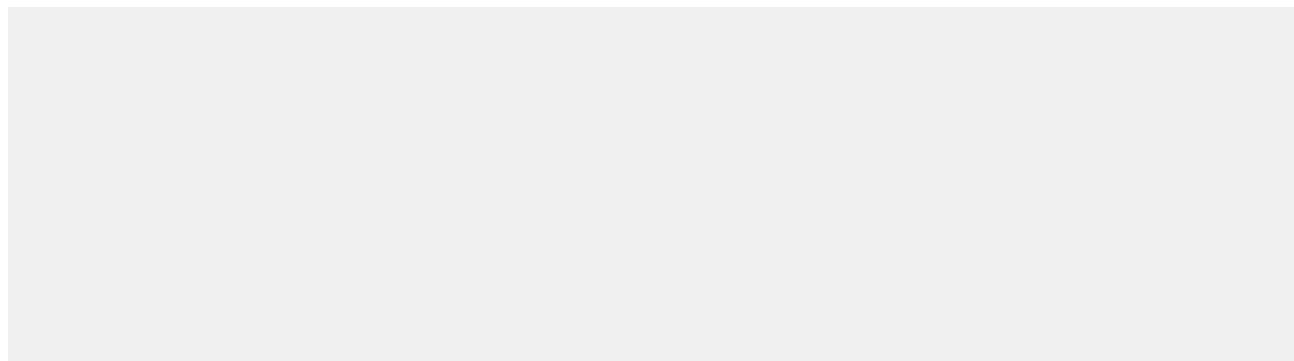
次のクラスでは、Apex の一括処理を使用して、特定のユーザが所有するすべての取引先を、異なるユーザに割り当てます。



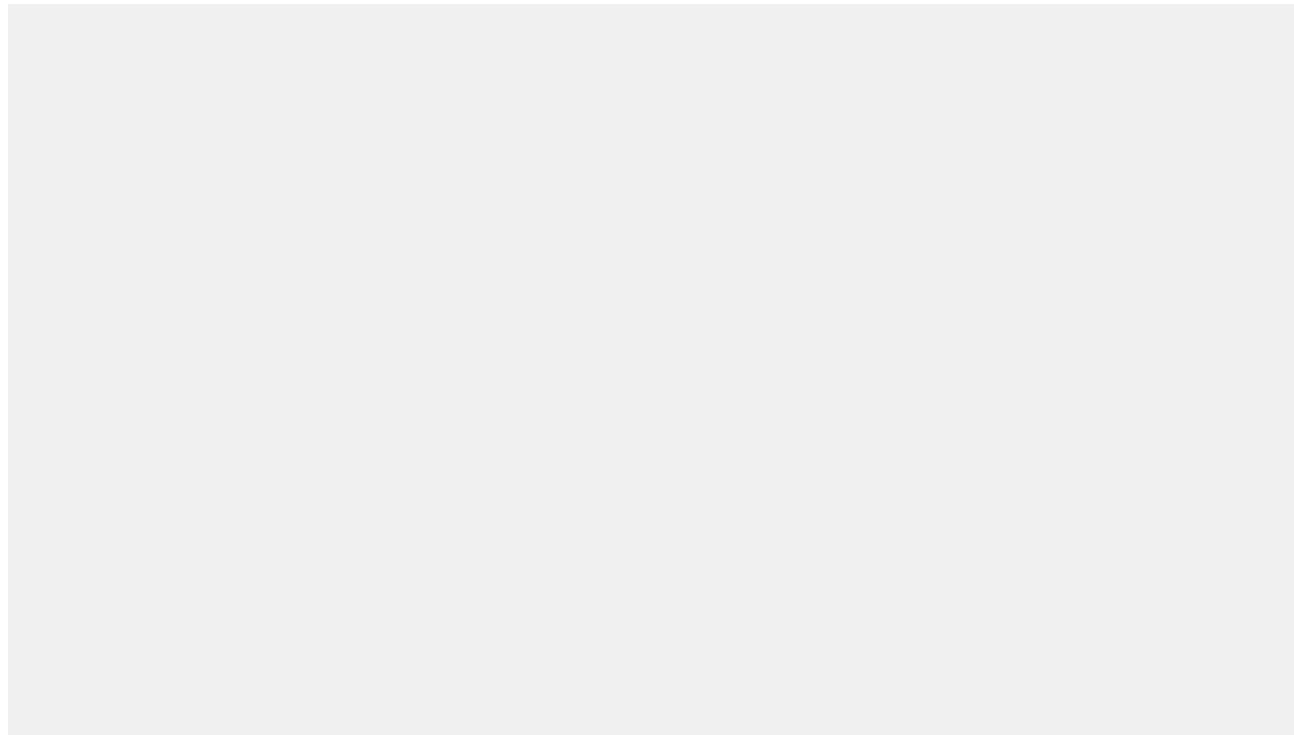


前の例の

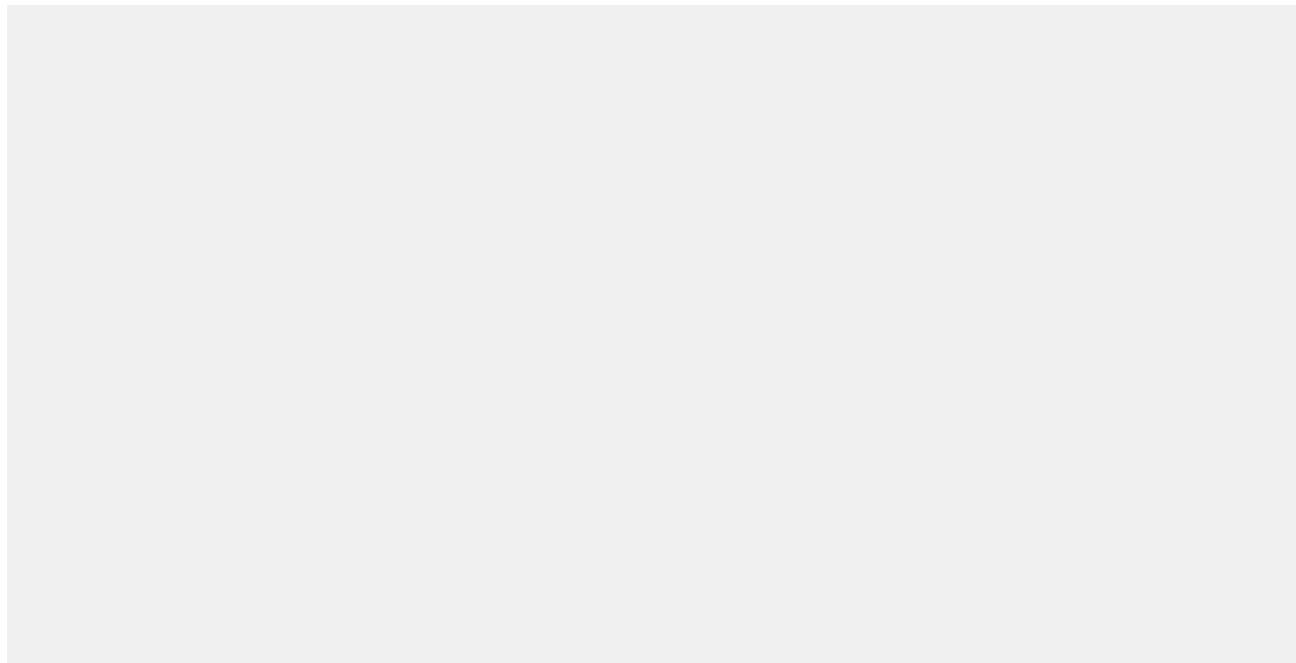
クラスを実行するには次を使用します。



次は、レコードを削除する Apex の一括処理クラスの例です。



このコードは、古いドキュメントを削除する、Apex の一括処理クラスをコールします。ここに示すクエリは、指定したフォルダ内の特定の日付より古いドキュメントをすべて削除するために、それらのドキュメントを選択し、その後で、一括処理ジョブを呼び出します。

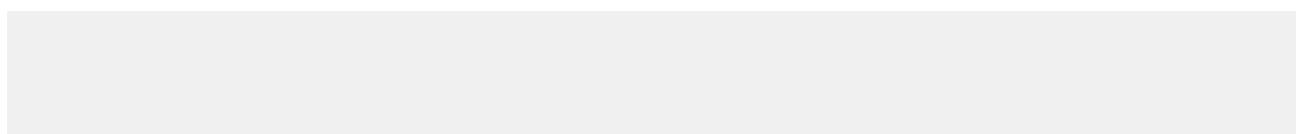


## Apex の一括処理でのコールアウトの使用

Apex の一括処理で [コールアウト](#) を使用するには、このクラス定義で

を指定する必

要があります。次に例を示します。



コールアウトには、HTTP 要求および

キーワードで定義されたメソッドが含まれています。

## Apex の一括処理での状態の使用

Apex の一括処理ジョブの各実行は、個別のトランザクションとみなされます。たとえば、1,000 件のレコードを含む Apex の一括処理ジョブが、任意の `scope` パラメータを指定せずに実行されると、このジョブはそれぞれ 200 件のレコードを含む 5 つのトランザクションとみなされます。

クラス定義で

を指定すると、これらのトランザクション間で状態を保持できます。

を使用するとき、インスタンスマンバー変数のみがトランザクション間で値を保持します。

静的メンバー変数は、トランザクション間で値を保持せず、リセットされます。状態を保持すると、処理されているレコードをカウントまたは集計する場合に役立ちます。たとえば、ジョブで商談レコードが処理されたとします。

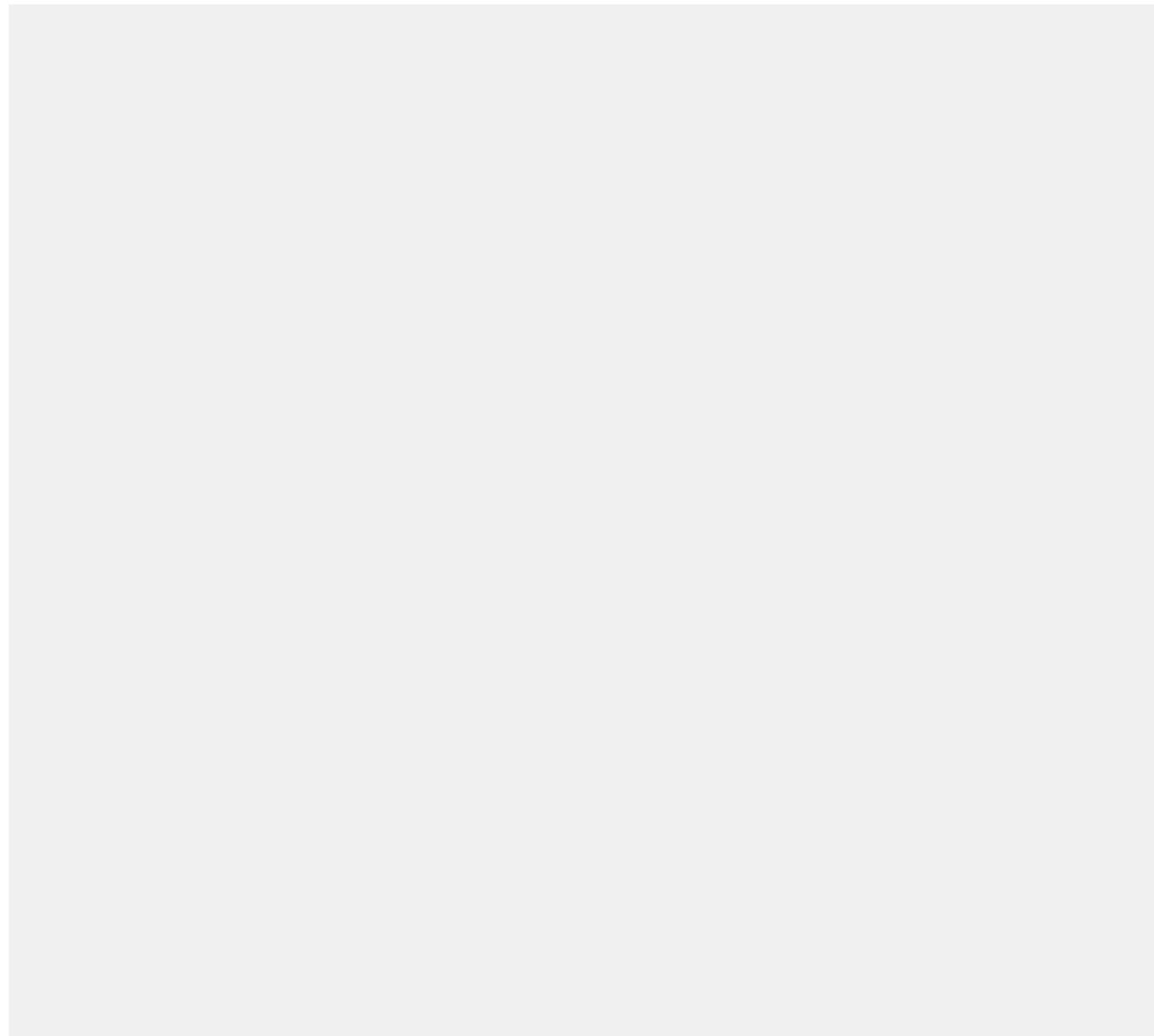
でメソッドを定義し、処理された商談数の合計を集計できます。

を指定しない場合、すべての静的メンバー変数とインスタンスマンバー変数が元の値に戻されます。

次の例では、レコードが処理されるとカスタム項目  
が集計されます。

また、変数を指定してクラスの最初の状態にアクセスできます。この変数を使用して、  
ソッドのすべてのインスタンスと最初の状態を共有できます。次に例を示します。

メ



はクラスの「最初の」状態です。これを使用して、一括処理ジョブの実行時にクラスのインスタンス間で情報を受け渡すことはできません。たとえば、`で` の値を変更した場合、2番目に処理されたレコード群は、新しい値にアクセスできません。最初の値のみにアクセスできます。

## Apex の一括処理のテスト

Apex の一括処理をテストするとき、`メソッドの 1 つの実行だけをテストできます。` メソッドの `scope` パラメータを使用して、`メソッドに渡されるレコード数を制限し、ガバナ制限に達しないようにすることができます。`

メソッドは、匿名プロセスを開始します。これは、Apex の一括処理をテストする場合、結果に対してテストする前に一括処理ジョブを終了する必要があることを意味します。`メソッドを実行する前後で、テストメソッド` `と` `を使用して、テストを続行する前に一括処理ジョブが終了するようにします。` メソッドの後に作成されたすべての非同期コールはシステムによって収集されま

す。 を実行する場合、すべての非同期プロセスが同期して実行されます。 メソッドおよび  
メソッド内に メソッドを含めない場合、一括処理ジョブは Salesforce.com API バージョン 25.0 以降を使用して保存された Apex のテストメソッドでは最後に実行されますが、それよりも前のバージョンで保存された Apex の場合は実行されません。

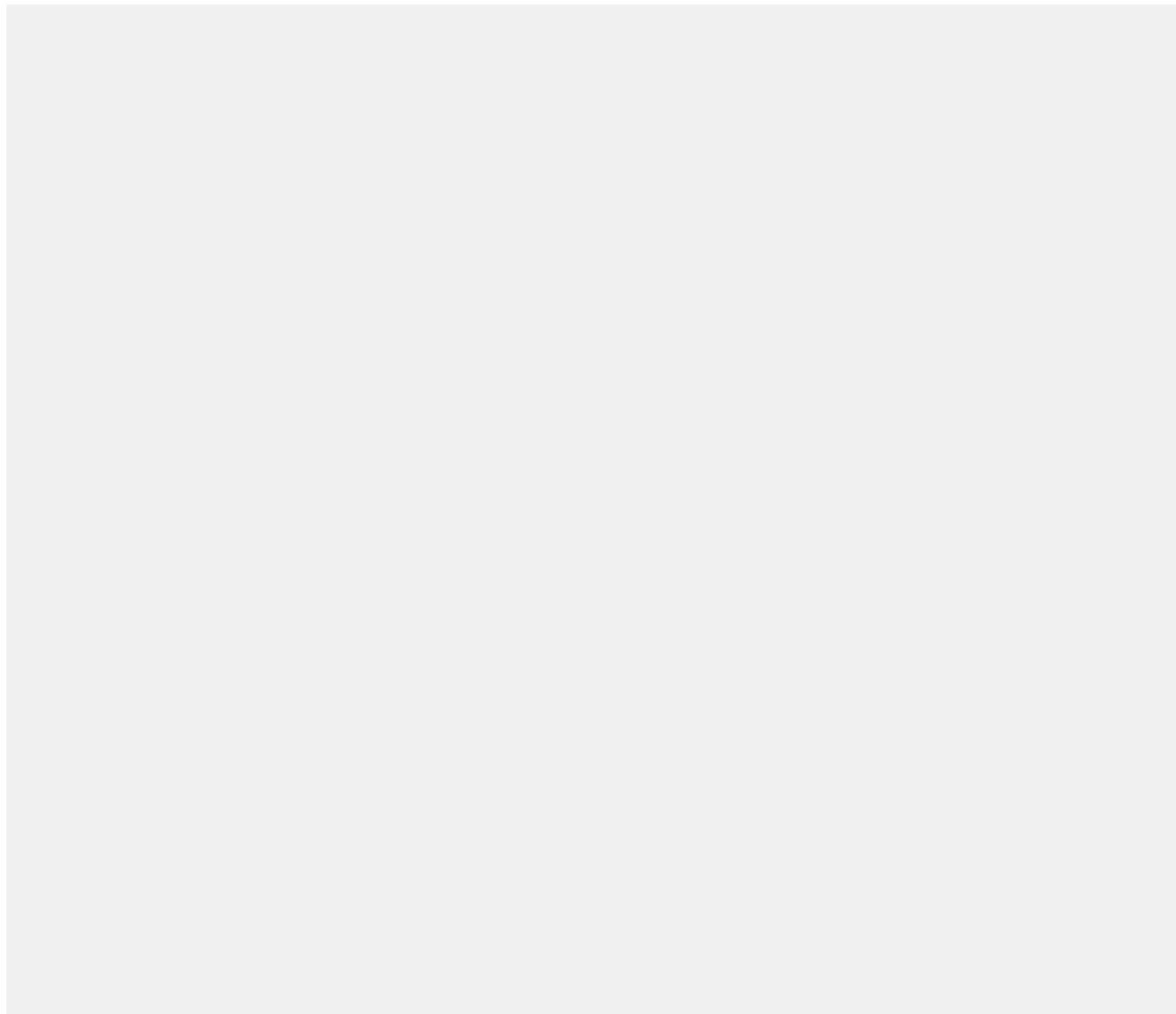
Salesforce.com API バージョン 22.0 を使用して保存した Apex 以降、テストメソッドによって呼び出される Apex の一括処理ジョブの実行中に発生する例外は、コード元のテストメソッドに渡されるようになり、その結果としてテストメソッドが失敗します。テストメソッドで例外を処理する必要がある場合は、コードを `try` ブロックと `catch` ブロックで囲みます。ただし、Salesforce.com API バージョン 21.0 以前を使用して保存した Apex では、該当する例外はテストメソッドに渡されないため、テストメソッドは失敗しません。

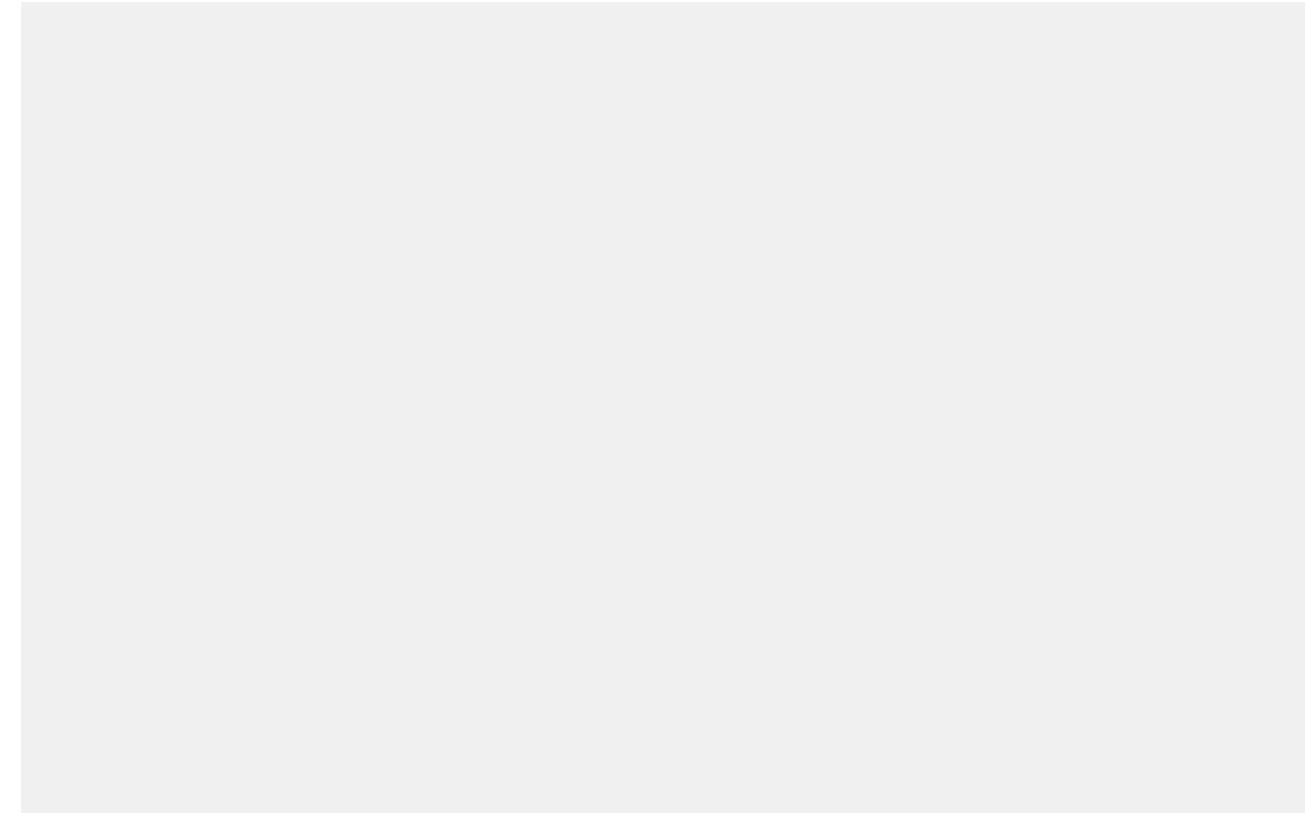


メモ: ブロックおよび ブロックでコールされた または などの非同期コールは、キュー内ジョブ数の制限に対してカウントされません。

以下の例では、

[クラス](#) をテストします。





## Apex の一括処理のガバナ制限

Apex の一括処理について、次のガバナ制限に注意してください。

- Apex では最大 5 件のクュー内または有効な一括処理ジョブを実行できます。
- 24 時間での Apex 一括処理メソッドの最大実行数は、250,000 または組織のライセンス数の 200 倍の大きいほうです。メソッドの実行数には、**Apex**、**Future**、および **Batch** メソッドの実行が含まれます。これは組織全体の制限で、他のすべての非同期 Apex(スケジュール済み Apex および future メソッド)と共有されます。この制限のカウント対象となるライセンスは、Salesforce フルユーザライセンスまたは Force.com アプリケーションサブスクリプションのユーザライセンスです。Chatter 限定ユーザ、Chatter カスタマーユーザ、カスタマー・ポートアルユーザ、およびパートナーポータルユーザライセンスは、含まれません。
- ユーザは一度に最大 50 個のクエリーカーソルを開くことができます。たとえば、50 個のカーソルが開いていて、同じユーザとしてログインしたままのクライアントアプリケーションが新しいカーソルを開こうすると、50 個のカーソルのうち最も古いカーソルが解放されます。この制限は、一度に 1 ユーザあたり最大 5 個のクエリーカーソルを開くことができる、Apex 一括処理メソッドでは異なります。それ以外の Apex 一括処理メソッドには、高い方のカーソル数制限である 50 が適用されます。

異なる Force.com 機能のカーソル制限は個別に追跡されます。たとえば、50 個の Apex クエリーカーソル、50 個の一括処理カーソル、および 50 個の Visualforce カーソルを同時に開くことができます。

- オブジェクトでは最大 5,000 万件のレコードが返されます。5,000 万件以上のレコードが返された場合、一括処理ジョブは即座に終了し「失敗」とマークされます。
- メソッドが QueryLocator を返す場合、**BatchSize** の省略可能な範囲パラメータには最大値 2,000 を指定できます。これより大きい値に設定すると、Salesforce では、QueryLocator が返すレコードを、最大 2,000 レコードまでの、より小さいバッチに分割します。メソッドが Iterable オブジェクトを

返す場合、範囲パラメータ値に上限はありませんが、非常に大きい値を使用すると、他の制限が発生する場合があります。

- の *scope* パラメータ (省略可能) でサイズが指定されない場合、Salesforce ではメソッドによって返されるレコードを 200 個ずつのバッチに分割し、各バッチをメソッドに渡します。Apex ガバナ制限は、の各実行でリセットされます。
- 、および メソッドは、それぞれ最大 10 回のコールアウトを実装できます。
- 一括処理の実行は、メソッド実行 1 回あたり 10 コールアウトに制限されます。
- Apex の一括処理ジョブの メソッドは、組織内で一度に 1 つのみ実行できます。キュー内のまだ開始されていない一括処理ジョブは、開始されるまで保持されます。なお、この制限により一括処理ジョブが失敗することはありません。また、複数のジョブが実行されている場合は、Apex の一括処理ジョブの メソッドが並行して実行されます。

## Apex の一括処理のベストプラクティス

- 一括処理ジョブをトリガから開始する場合は、細心の注意を払ってください。制限の 5 件を超える一括処理ジョブがトリガで追加されないようにする必要があります。特に、API の一括更新、インポートウィザード、ユーザインターフェースを使用したレコードの一括変更、および複数のレコードを一度に更新するすべての処理については十分に考慮してください。
- をコールしたときに Salesforce が行うのは、そのジョブをキューに入れることのみです。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。
- Apex の一括処理をテストするとき、 メソッドの 1 つの実行だけをテストできます。 メソッドの *scope* パラメータを使用して、 メソッドに渡されるレコード数を制限し、ガバナ制限に達しないようにすることができます。
- メソッドは、匿名プロセスを開始します。これは、Apex の一括処理をテストする場合、結果に対してテストする前に一括処理ジョブを終了する必要があることを意味します。 メソッドを実行する前後で、テストメソッド と を使用して、テストを続行する前に一括処理ジョブが終了するようにします。
- ジョブトランザクション全体でインスタンスマンバー変数またはデータを共有する場合は、クラス定義でを使用します。これを使用しない場合、各トランザクションの開始時にすべてのメンバー変数が初期状態にリセットされます。
- として宣言されたメソッドは、インターフェースを実装するクラスでは使用できません。
- として宣言されたメソッドは、Apex の一括処理クラスからはコールできません。
- Salesforce.com API バージョン 26.0 以降を使用して保存された Apex では、 または を メソッドからコールできます。これにより、現在の一括処理ジョブが終了したら新しい一括処理ジョブを開始またはスケジュールできます。以前のバージョンでは、Apex の一括処理メソッドから と はコールできませんでした。使用バージョンは他の一括処理ジョブを開始またはスケジュールするバッチを実行しているクラスのバージョンです。バッチを実行しているクラスの メソッドで一括処理ジョブを開始するヘルパークラスのメソッドをコールする場合、ヘルパークラスの Salesforce.com API バージョンは関係ありません。
- 一括処理ジョブでは、 および PageReference メソッドは使用できません。
- サービスの機能停止などの重大な障害が発生した場合、進行中の処理は「失敗」とマークされます。エラーを修正するために、一括処理ジョブをもう一度実行する必要があります。
- Apex の一括処理ジョブが実行されると、一括処理ジョブを送信したユーザにメール通知が送信されます。または、管理パッケージにコードが含まれ、登録組織が一括処理ジョブを実行している場合、[Apex の例外の通知受信者] 項目に表示された受信者にメールが送信されます。

- 各メソッドの実行では、標準のガバナ制限が、匿名ブロック、Visualforce コントローラ、または WSDL メソッド同様に適用されます。
- Apex の一括処理が呼び出されるたびに レコードが作成されます。このレコードの ID を使用して、ジョブの状況、エラーの数、進行状況、申請者を取得する SOQL クエリを構成します。  
オブジェクトについての詳細は、『Object Reference for Salesforce and Force.com』の「[AsyncApexJob](#)」を参照してください。
- 10,000 件ごとの タイプの追加の 項目を使用して レコードに対して、Apex では、内部で使用するための レコードを作成します。すべての レコードをクエリする場合は、  
場合、クエリにより 10,000 件の タイプのレコードを除外することをお勧めします。除外しない  
オブジェクトについての詳細は、『Object Reference for Salesforce and Force.com』の「[AsyncApexJob](#)」を参照してください。
- クラス内のすべてのメソッドは または として定義する必要があります。
- 共有再適用の場合、一括処理内のレコードに対する Apex による共有管理を、すべて メソッドで削除してから再作成することをお勧めします。そうすることで、正確で完全な共有が適用されます。

## 関連リンク

- [Exception ステートメント](#)
- [実行ガバナと制限について](#)
- [共有の理解](#)

## Apex による共有管理について

共有とは、レコードに対してアクションを実行する許可をユーザまたはユーザグループに付与する行為のことです。共有アクセス権は、Salesforce ユーザインターフェースおよび Force.com を使用して付与することも、Apex を使用してプログラムで付与することもできます。

この項では、Apex を使用した共有の概要について説明します。

- [共有の理解](#)
- [Apex を使用したレコードの共有](#)
- [Apex による共有管理の再適用](#)

共有についての詳細は、Salesforce オンラインヘルプの「組織のデフォルトの共有設定の設定」を参照してください。

## 共有の理解

共有は、すべてのカスタムオブジェクトと、Account、Contact、Opportunity、Caseなどの多くの標準オブジェクトのレコードレベルのアクセス制御を実現します。管理者は最初にオブジェクトの組織の共有アクセスレベルを設定し、レコード所有者、ロール階層、共有ルール、共有の直接設定などに基づいてその他のアクセス権を付与します。開発者は Apex 共有管理を使用できるようになり、Apex を使用したプログラムからのアクセス権の付与

が可能になります。レコードに対するほとんどの共有は、関連する共有オブジェクトで保持されます。これは、他のプラットフォームのアクセス制御リスト (ACL) と似た機能です。

## 共有のタイプ

Salesforce には、次のタイプの共有があります。

### Force.com による共有管理

Force.com による共有管理では、レコードの所有者、ロール階層、および共有ルールに基づいて Force.com によって付与される共有アクセス権を使用します。

#### レコードの所有者

各レコードは、ユーザまたは場合によっては、カスタムオブジェクト、ケース、およびリードのキューが所有します。レコードの所有者にはフルアクセスが自動的に付与され、レコードを参照、編集、移行、共有、および削除できます。

#### ロール階層

ロール階層により、その階層内の別のユーザよりも上位のユーザが、下位ユーザの所有レコードまたは下位ユーザに共有されているレコードに対して、同じレベルのアクセス権を持つことができます。

そのため、ロール階層内のレコード所有者より上位のユーザにも、そのレコードに対するフルアクセスが暗黙的に付与されます。ロール階層は共有するレコードと共に維持されません。代わりに、ロール階層アクセス権が実行時に取得されます。詳細は、Salesforce オンラインヘルプの「階層を使用したアクセス権の制御」を参照してください。

#### 共有ルール

共有ルールは、システム管理者が、特定のユーザグループが所有するレコードへのアクセス権を特定のグループまたはロール内のユーザに自動的に付与する場合に使用します。共有ルールは、Force.com AppExchange からインストールしたアプリケーションのパッケージに追加したり、共有ロジックをサポートする目的で使用したりすることはできません。

共有ルールは、レコード所有者または他の条件に基づいて作成できます。条件に基づく共有ルールの作成に Apex は使用できません。また、条件に基づく共有は Apex を使用してテストできません。

Force.com による共有管理で追加された暗黙的な共有はすべて、Salesforce ユーザインターフェース、SOAP API、または Apex を使用して直接変更することはできません。

### ユーザによる共有管理(共有の直接設定)

ユーザによる共有管理により、レコードの所有者や、レコードに対するフルアクセスを持つユーザは、ユーザまたはユーザグループとレコードを共有できます。一般にこの処理は、エンドユーザが単一レコードに対して実行します。レコードの所有者とロール階層内でその所有者の上位にあるユーザにのみ、レコードに対するフルアクセスが付与されます。他のユーザにフルアクセスを付与することはできません。特定のオブジェクトに対するオブジェクトレベルの「すべての編集」権限を持つユーザは、レコードを手動で共有することもできます。レコードの所有者が変更された場合や、共有で付与されたアクセス権でオブジェクトの組織の共有デフォルトアクセスレベルを超える追加アクセス権が許可されない場合に、ユーザによる共有管理が削除されます。

### Apex による共有管理

開発者は、Apex による共有管理を使用すると、アプリケーションの特定の共有要件を Apex または SOAP API によるプログラムでサポートできるようになります。この種類の共有は、Force.com による共有管理に

類似しています。「すべてのデータの編集」権限を持つユーザのみが、レコードへの Apex による共有管理を追加または変更できます。Apex による共有管理は、レコードの所有者を変更しても維持されます。



メモ: Apex 共有の理由と Apex による共有管理の再適用は、カスタムオブジェクトでのみ使用できます。

## 共有の理由項目

Salesforce ユーザインターフェースでカスタムオブジェクトの 理由 項目は、レコードで使用される共有の種類を指定します。この項目は、Apex または Force.com API では となります。

次の各リスト項目は、レコードに使用される共有の種類です。表は、 理由 項目値と関連する 値を示します。

- Force.com による共有管理

[理由] 項目値	rowCause 値 (Apex または Force.com API で使用)
取引先の共有	
関連するレコードの所有者または共有	
所有者	
商談チーム	
共有ルール	
テリトリー割り当てルール	

- ユーザによる共有管理

[理由] 項目値	rowCause 値 (Apex または Force.com API で使用)
共有の直接設定	
テリトリー直接設定	

- Apex による共有管理

[理由] 項目値	rowCause 値 (Apex または Force.com API で使用)
開発者による定義	Defined by developer

Apex による共有管理で表示されている理由は開発者が定義します。

## アクセスレベル

レコードへのユーザのアクセス権を決定する場合、最も権限の高いアクセスレベルを使用します。ほとんどの共有オブジェクトは、次のアクセスレベルをサポートしています。

アクセスレベル	API 名	説明
非公開	None	レコードの所有者とロール階層内でその所有者の上位にあるユーザのみがレコードを参照または編集できます。このアクセスレベルは AccountShare オブジェクトにのみ適用されます。
参照のみ	Read	指定されたユーザまたはグループがレコードの参照のみを実行できます。
参照・更新	Edit	指定されたユーザまたはグループがレコードを参照および編集できます。
フルアクセス	All	指定されたユーザまたはグループがレコードを参照、編集、移行、共有、削除できます。
		 メモ: このアクセスレベルは、Force.com 共有管理でのみ付与できます。

## Apex を使用したレコードの共有

プログラムから共有にアクセスするには、共有する標準オブジェクトまたはカスタムオブジェクトに関連付けられている共有オブジェクトを使用する必要があります。たとえば、AccountShare は Account オブジェクトの共有オブジェクト、ContactShare は Contact オブジェクトの共有オブジェクトです。他のオブジェクトについても同様です。さらに、すべてのカスタムオブジェクトの共有オブジェクトには次のように名前が付けられています。*MyCustomObject* はカスタムオブジェクトの名前です。

*MyCustomObject*

主従関係の従側にあるオブジェクトには、関連付けられた共有オブジェクトはありません。従レコードへのアクセスは、主の共有オブジェクトと関係の共有設定により定義されます。詳細は、Salesforce オンラインヘルプの「カスタムオブジェクトのセキュリティ」を参照してください。

共有オブジェクトには、Force.com 共有管理、ユーザ共有管理、Apex 共有管理の 3 種類の共有すべてをサポートするレコードが含まれています。組織の共有設定、ロールの階層、および特定オブジェクトの「すべての参照」や「すべての編集」などの権限、「すべてのデータの参照」および「すべてのデータの編集」を使用してユーザに暗黙的に付与された共有は、このオブジェクトでは追跡されません。

各共有オブジェクトには、次のプロパティがあります。

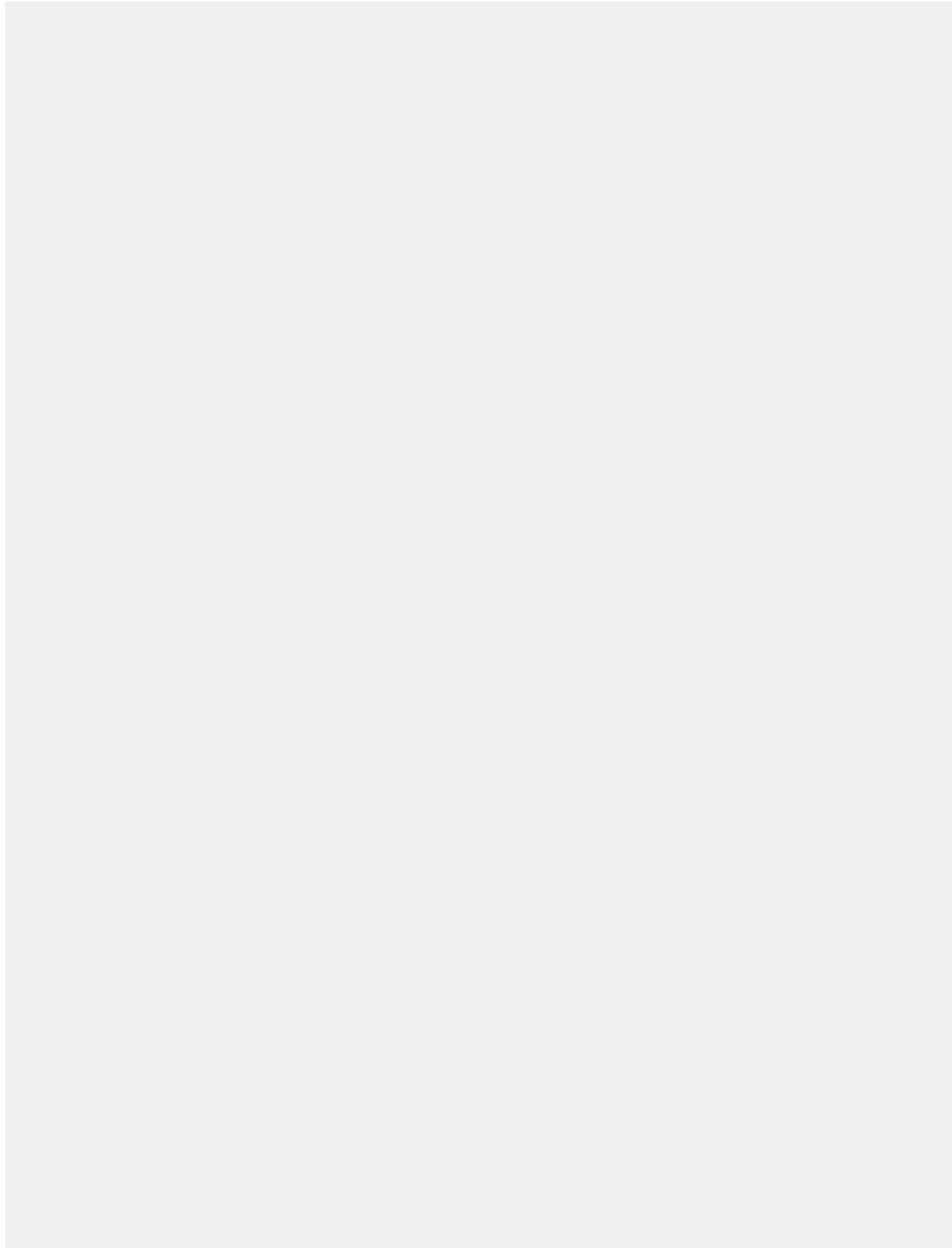
プロパティ名	説明
<i>objectName</i>	共有 sObject に対し、指定されたユーザまたはグループが権限を与えられたアクセスレベル。プロパティ名は、オブジェクト名に <i>Share</i> が追加したものです。

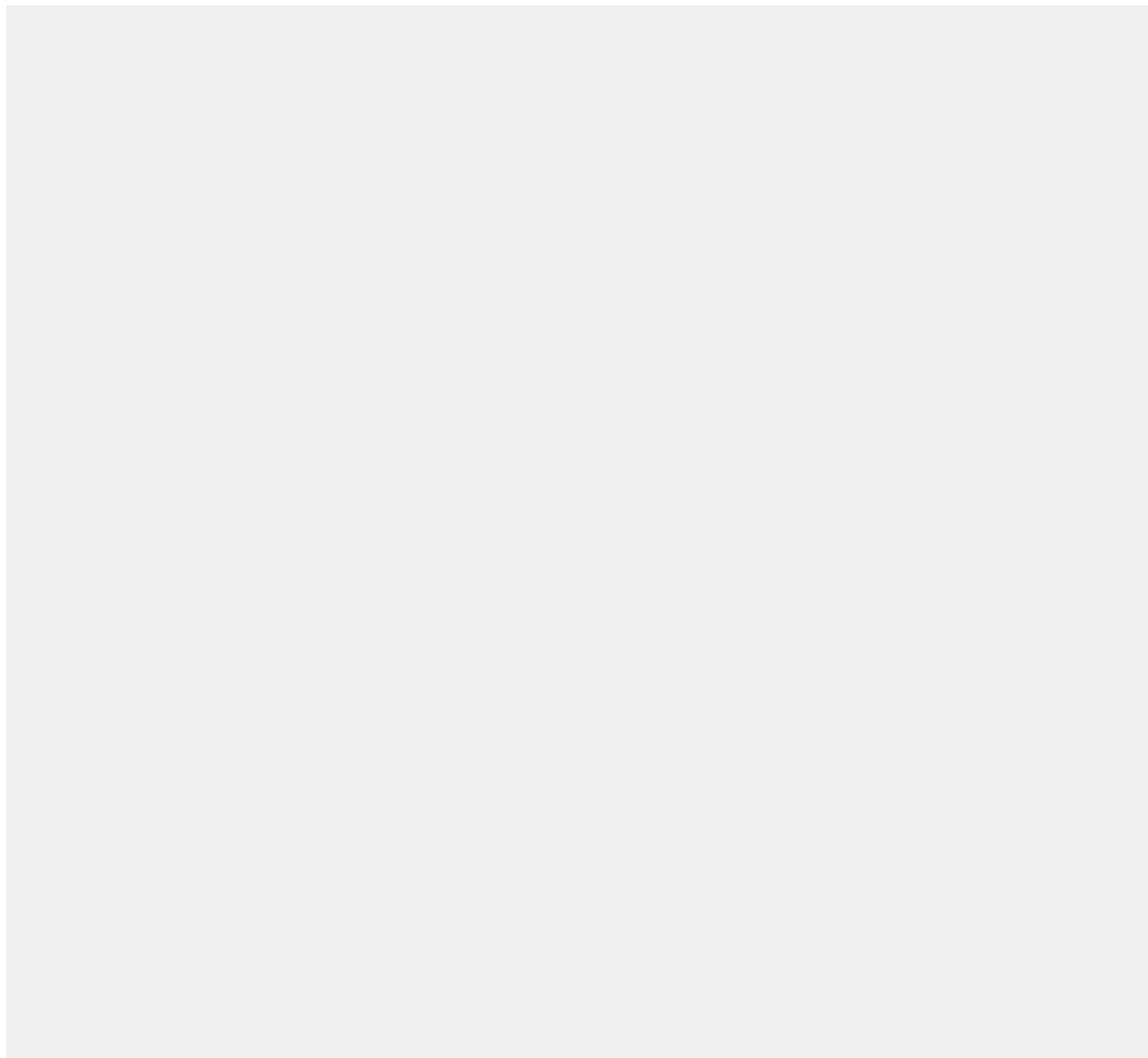
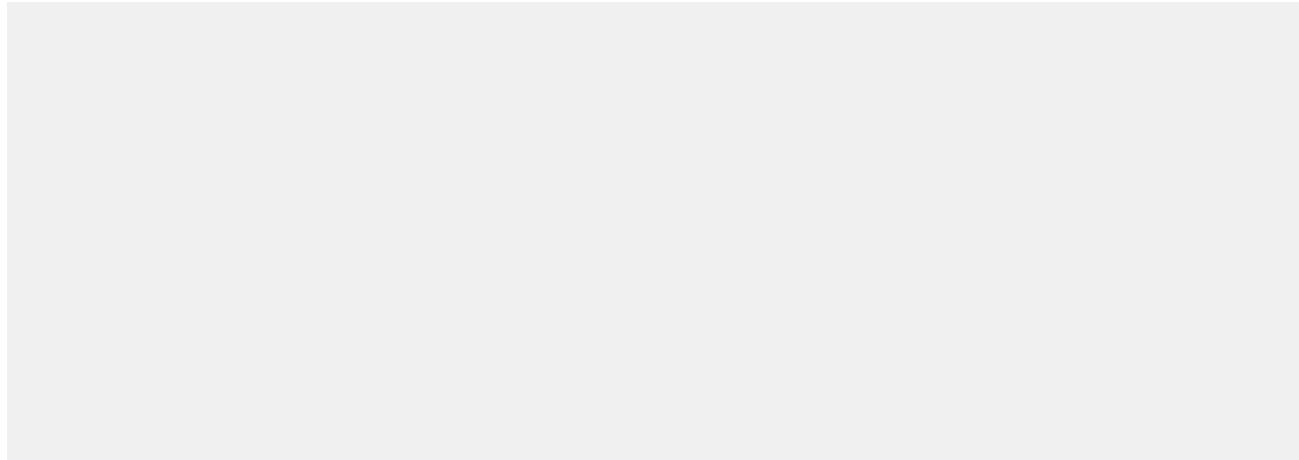
プロパティ名	説明
	<p>たとえば、LeadShare オブジェクトのプロパティ名は す。有効な値は、次のとおりです。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> </ul>
	 メモ: アクセスレベルは、Force.com 共有管理でのみ使用できます。
	<p>この項目には、その親オブジェクトに割り当てられた組織のデフォルトアクセスレベルよりも高いアクセスレベルを割り当てる必要があります。詳細は、「<a href="#">アクセスレベル</a>」(ページ 291)を参照してください。</p>
	<p>オブジェクトの ID。この項目は更新できません。</p>
	<p>ユーザまたはグループにアクセス権が付与される理由。この理由によって、共有のタイプが決まります。共有のタイプは、共有レコードの変更権限を制御します。この項目は更新できません。</p>
	<p>アクセス権を付与するユーザまたはグループの ID。グループには公開グループ、ロール、テリトリーを指定できます。この項目は更新できません。</p>

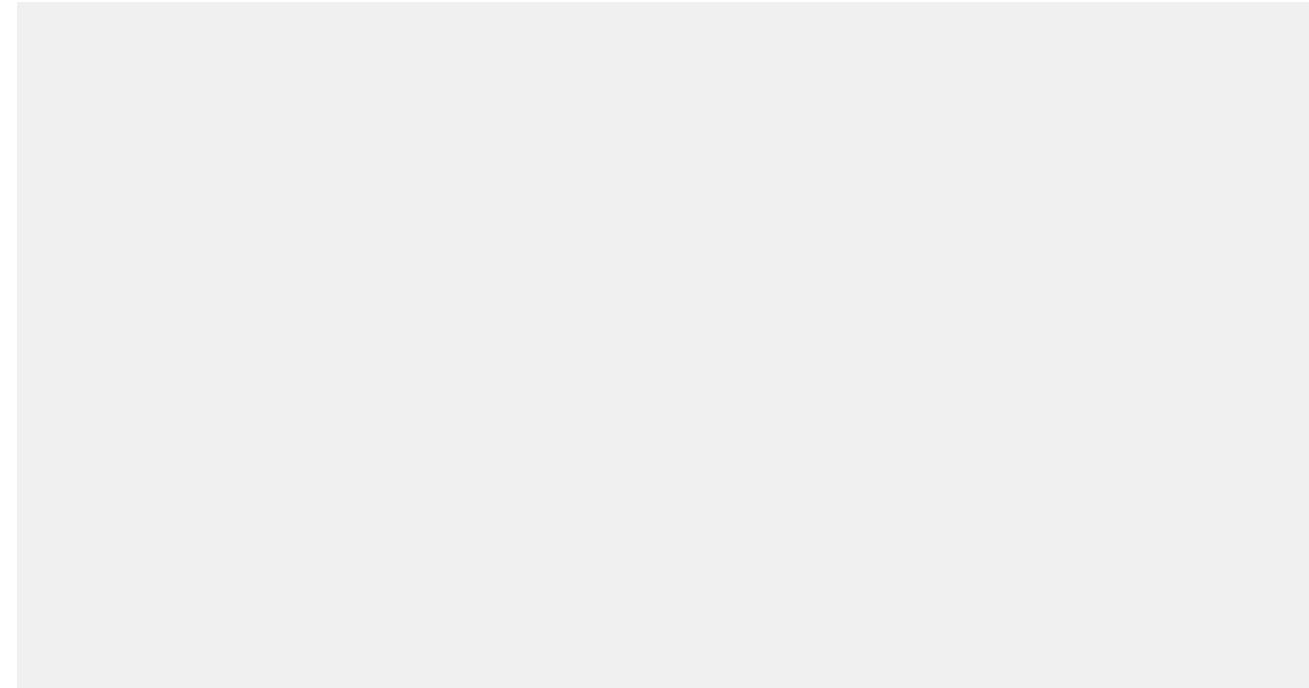
ユーザまたはグループでは標準オブジェクトまたはカスタムオブジェクトは共有できません。オブジェクトを共有できるユーザおよびグループの種別についての詳細は、『[Object Reference for Salesforce and Force.com](#)』の「User」と「Group」を参照してください。

### Apex を使用したユーザ共有管理の作成

Apex または SOAP API を使用して、1つのユーザまたはグループに対してレコードの共有を直接設定できます。レコードの所有者が変更されると、共有は自動的に削除されます。次の例のクラスには、参照アクセスのある特定のユーザまたはグループ ID を伴うジョブ ID によって指定されたジョブを共有するメソッドが含まれます。また、このメソッドを検証するテストメソッドも含まれます。このクラス例を保存する前に、Job というカスタムオブジェクトを作成します。







**重要:** 組織のデフォルトのアクセスレベルは、最も権限の大きいアクセスレベルに設定することはできません。カスタムオブジェクトの場合は「公開/参照・更新可能」です。詳細は、「[アクセスレベル](#)」(ページ 291)を参照してください。

## Apex による共有管理の作成

開発者は Apex による共有管理を使用すると、Apex または SOAP API を通じて、アプリケーションの動作をサポートする共有をプログラムで操作できるようになります。この種類の共有は、Force.com による共有管理に類似しています。「すべてのデータの編集」権限を持つユーザのみが、レコードへの Apex による共有管理を追加または変更できます。Apex による共有管理は、レコードの所有者を変更しても維持されます。

Apex による共有管理には、*Apex 共有の理由*を使用する必要があります。Apex 共有の理由は、ユーザやユーザグループでレコードを共有した理由を開発者が追跡するための 1 つの方法です。複数の Apex 共有理由を使用することで、共有レコードの更新や削除に必要なコーディングを簡略化することができます。また、開発者は、同じユーザやグループに対して異なる共有理由を設定して複数の共有を設定できます。

Apex 共有の理由は、オブジェクトの詳細ページとして定義されます。Apex 共有の理由には、それぞれラベルと名前が付けられます。

- ユーザインターフェースでレコードの共有を参照すると、理由 列に表示ラベルが表示されます。これにより、ユーザとシステム管理者が共有の目的を理解できます。表示ラベルは、トランスレーションワークベンチを使用する翻訳についても有効化されます。
- この名前は、API および Apex で理由を参照するときに使用します。

Apex 共有の理由の名前の形式は次のとおりです。

Apex 共有の理由は、次のようにプログラムで参照できます。

*CustomObject\_\_Share      SharingReason\_\_c*

たとえば、Job というオブジェクトの Apex 共有の理由である Recruiter は、次のように参照できます。

詳細は、「[Schema メソッド](#)」(ページ 518)を参照してください。

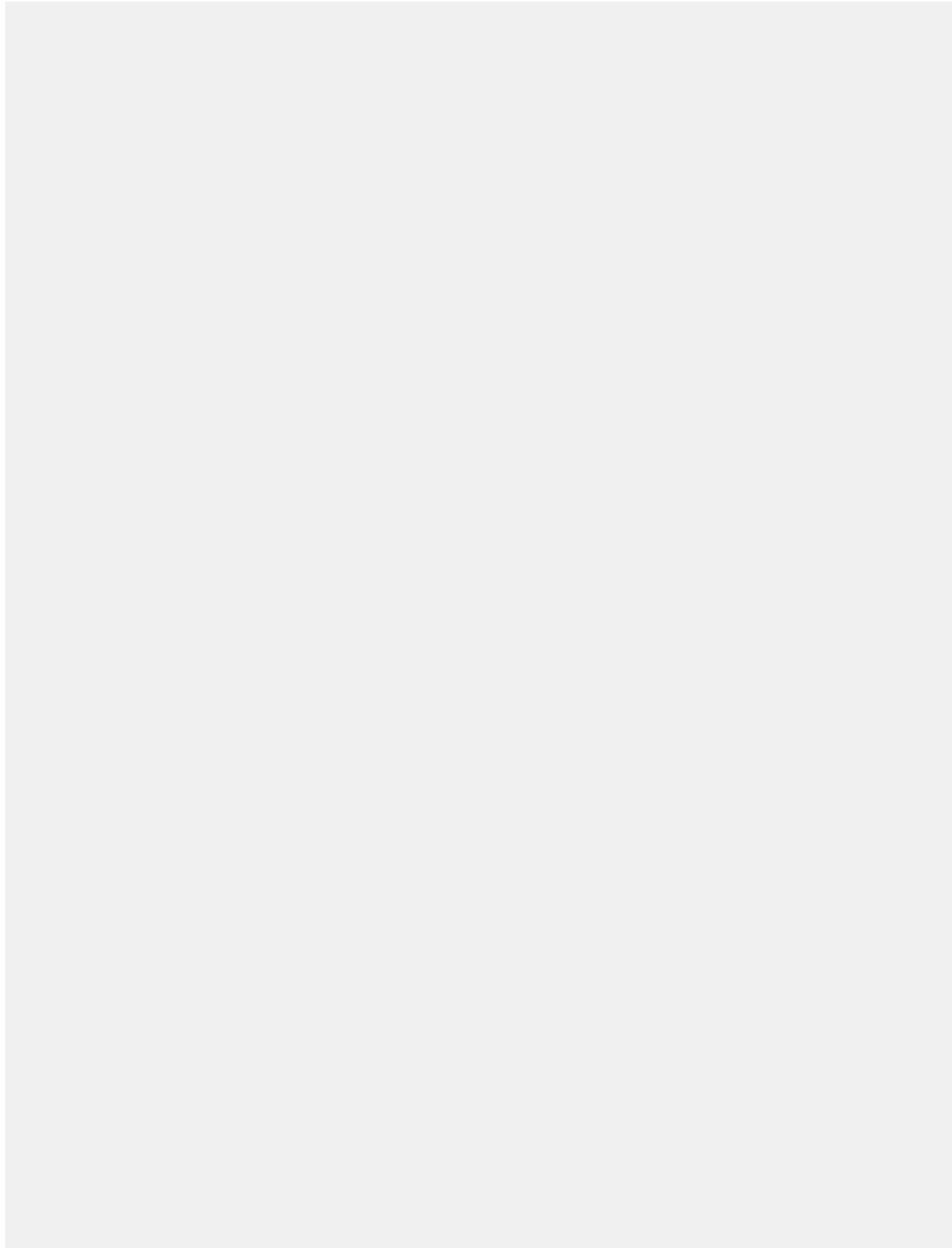
Apex 共有の理由を作成する手順は、次のとおりです。

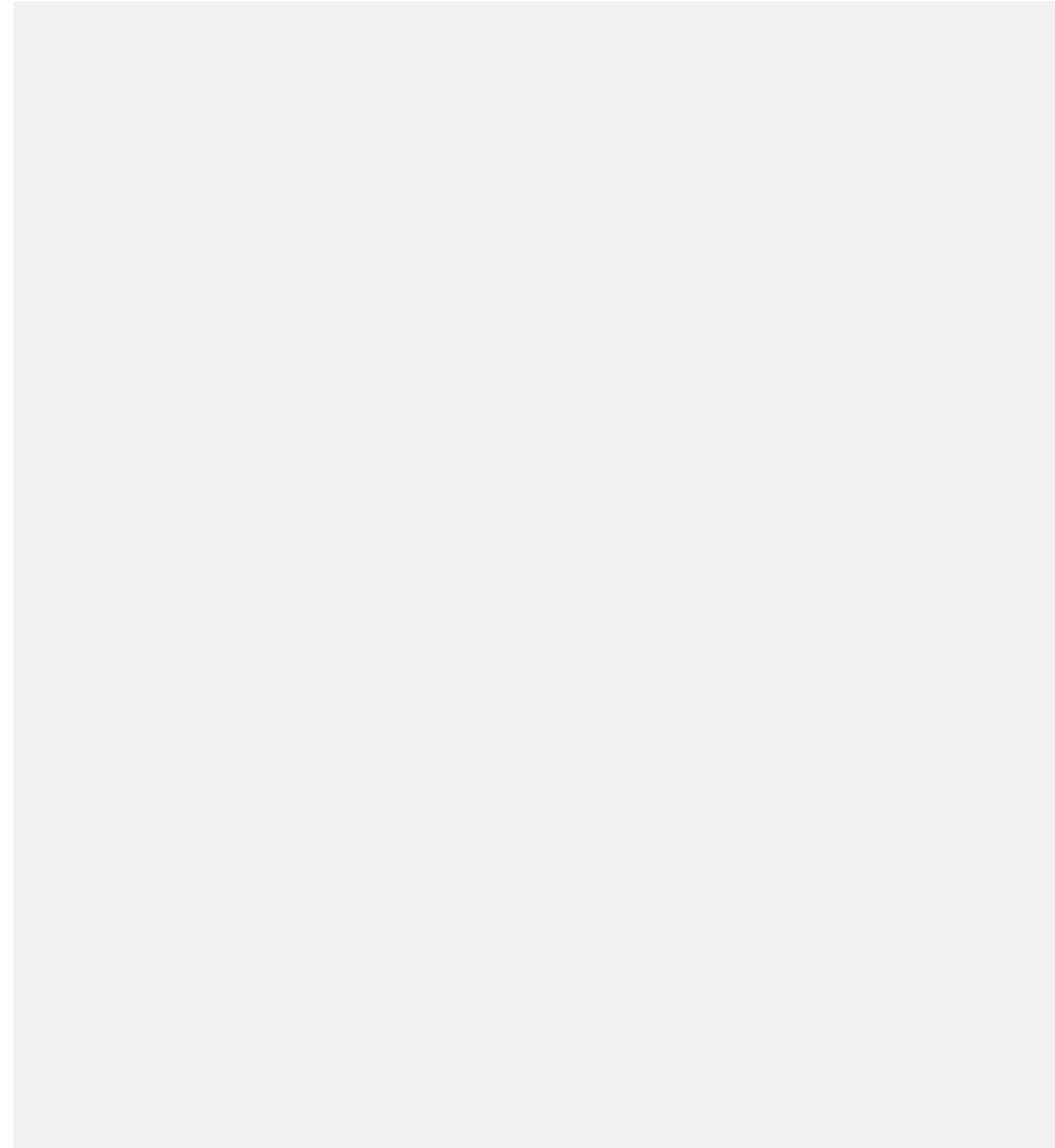
1. [設定] で、[作成] > [オブジェクト] をクリックします。
2. カスタムオブジェクトを選択します。
3. [Apex 共有の理由] 関連リストで [新規] をクリックします。
4. Apex 共有の理由の表示ラベルを入力します。ユーザインターフェースでレコードの共有を参照すると、理由列に表示ラベルが表示されます。表示ラベルは、トランスレーションワークベンチを使用する翻訳についても有効化されます。
5. Apex 共有の理由の名前を入力します。この名前は、API および Apex で理由を参照するときに使用します。この名前は、アンダースコアと英数字のみを含み、組織内で一意の名前にする必要があります。最初は文字であること、スペースは使用しない、最後にアンダースコアを使用しない、2つ続けてアンダースコアを使用しないという制約があります。
6. [保存] をクリックします。

 メモ: Apex 共有の理由と Apex による共有管理の再適用は、カスタムオブジェクトでのみ使用できます。

## Apex による共有管理の例

この例では、人事採用アプリケーションの構築中で、Job というオブジェクトが存在すると仮定しています。ジョブにリストされた採用担当者および採用担当マネージャにレコードへのアクセス権が付与されていることを確認したいと考えています。次のトリガは、ジョブレコード作成時に採用担当者および採用担当マネージャにアクセス権を付与します。この例では、User レコードと関連付けられた、Hiring\_Manager および Recruiter という 2 つの参照項目を持つ Job というカスタムオブジェクトが必要です。また、Job カスタムオブジェクトには、Hiring\_Manager と Recruiter という 2 つの共有の理由を追加する必要があります。





特定の状況下では、共有行を挿入すると、既存の共有行が更新されます。次の例を参考にしてください。

- ・ 共有の直接設定アクセスレベルが「参照」に設定されている場合、「更新」に設定された新しい共有行を挿入すると、元の共有行はより高いアクセスレベルを示す「更新」に更新されます。
- ・ ユーザが子レコード(取引先責任者、ケース、商談など)にアクセスできるため取引先にアクセスでき、取引先共有ルールが作成されている場合、親の暗黙的共有の共有理由は、共有ルールの共有理由で置き換えられ、高い方のアクセスレベルを示します。



**重要:** 組織のデフォルトのアクセスレベルは、最も権限の大きいアクセスレベルに設定することはできません。カスタムオブジェクトの場合は「公開/参照・更新可能」です。詳細は、「[アクセスレベル](#)」(ページ 291)を参照してください。

## Apex による共有管理の再適用

組織のデフォルトアクセスレベルが変更されると、オブジェクトの全レコードの共有が自動的に再適用されます。再適用により、適切な場合は Force.com 共有管理が追加されます。また、付与されたアクセス権が冗長である場合は、すべてのタイプの共有が削除されます。たとえば、オブジェクトの共有モデルが「非公開」から「公開/参照のみ」に変更されると、ユーザに「参照のみ」アクセス権を付与する共有の直接設定が削除されます。

Apex 共有管理を再適用するには、Salesforce が提供する再適用を行うインターフェースを実装する、Apex クラスを記述する必要があります。その後、[Apex 共有の再適用] 関連リストのカスタムオブジェクトの詳細ページで、クラスとカスタムオブジェクトを関連付ける必要があります。

メモ: Apex 共有の理由と Apex による共有管理の再適用は、カスタムオブジェクトでのみ使用できます。

Apex 共有の理由を指定するカスタムオブジェクトの詳細ページからこのクラスを実行します。ロックの問題により、アプリケーションのロジックに定義されたユーザへのアクセス権限の付与が Apex コードで実行されない場合、管理者はオブジェクトの Apex 共有管理を再適用する必要があることがあります。

**メソッド**を使用して、Apex 共有管理の再適用をプログラムで呼び出すこともできます。

メモ: カスタムオブジェクトの組織のデフォルト共有アクセスレベルが更新される度に、関連付けられたカスタムオブジェクトに定義された Apex 再適用クラスも実行されます。

Apex の再適用の実行を監視または停止するには、[設定] から [監視] > [Apex ジョブ] または [ジョブ] > [Apex ジョブ] をクリックします。

### 共有の再適用のための Apex クラスの作成

Apex 共有管理を再適用するには、再適用を行う Apex クラスを記述する必要があります。このクラスは、Salesforce が提供するインターフェースを実装している必要があります。

インターフェースは、Apex 共有管理の再適用など、すべての Apex の一括処理プロセスに使用されます。このインターフェースは、組織で複数回実装できます。実装する必要があるメソッドの詳細は、「[Apex の一括処理の使用](#)」(ページ 273)を参照してください。

Apex 共有管理の再適用を作成する前に、[ベストプラクティス](#)についても検討してください。

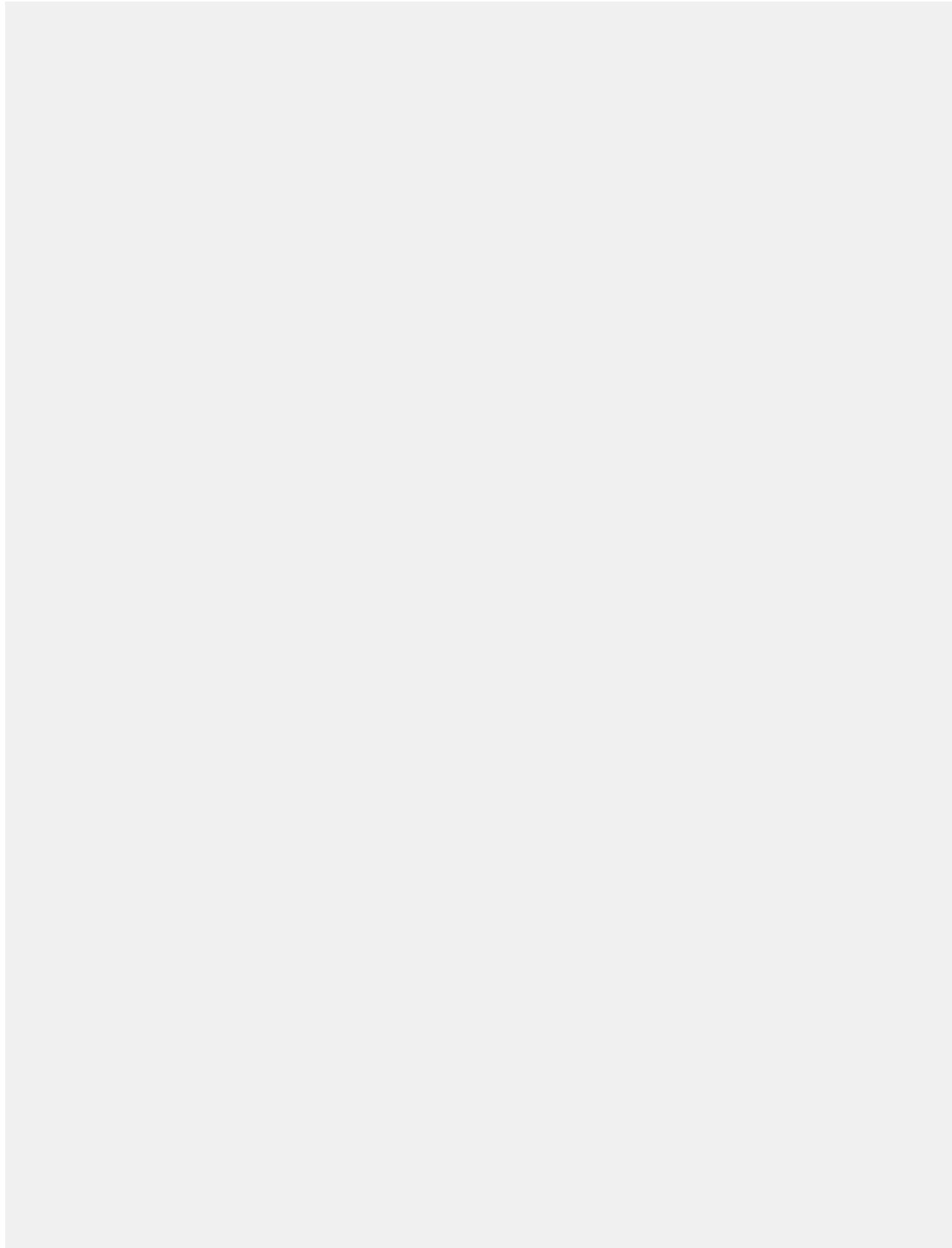


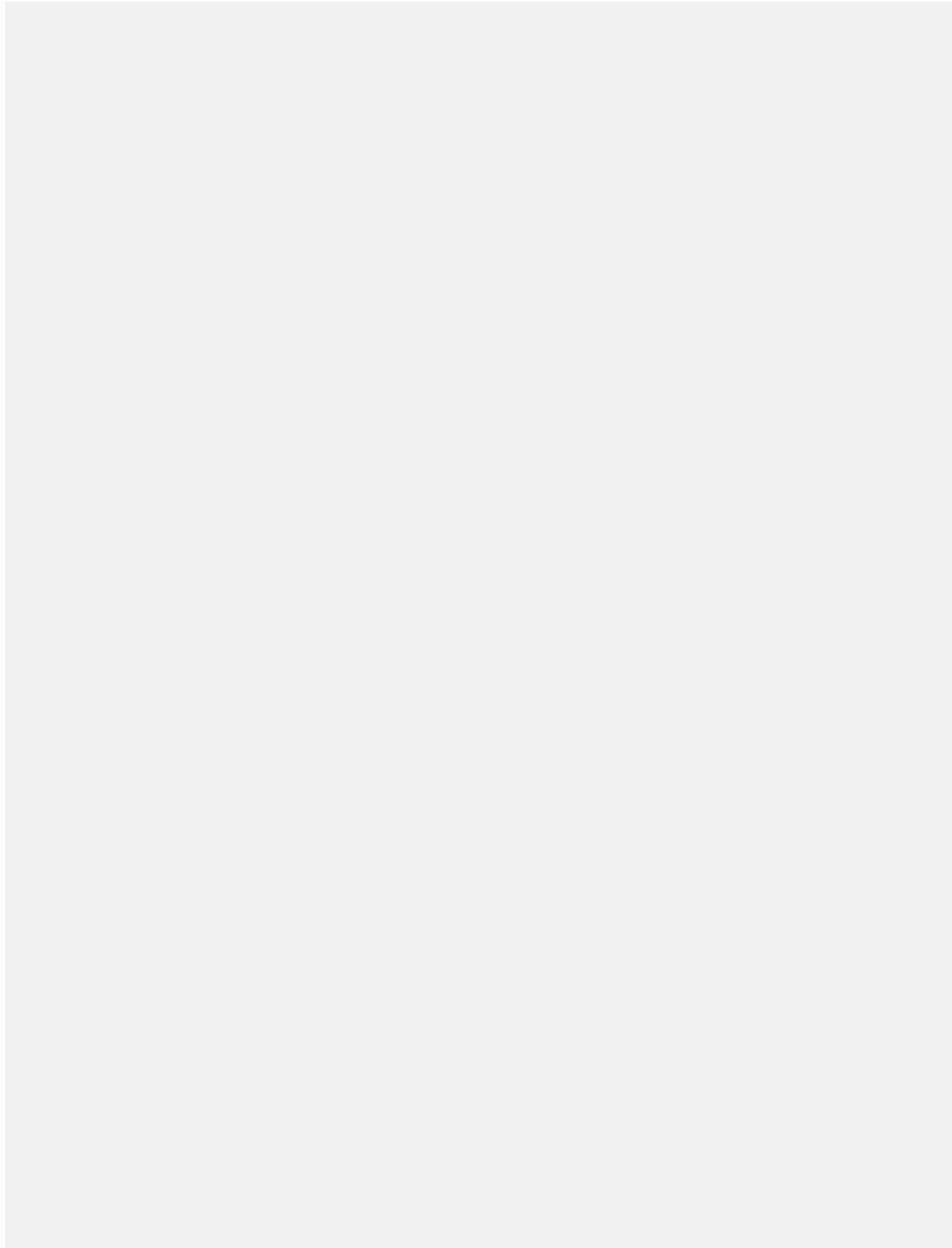
**重要:** 組織のデフォルトのアクセスレベルは、最も権限の大きいアクセスレベルに設定することはできません。カスタムオブジェクトの場合は「公開/参照・更新可能」です。詳細は、「[アクセスレベル](#)」(ページ 291)を参照してください。

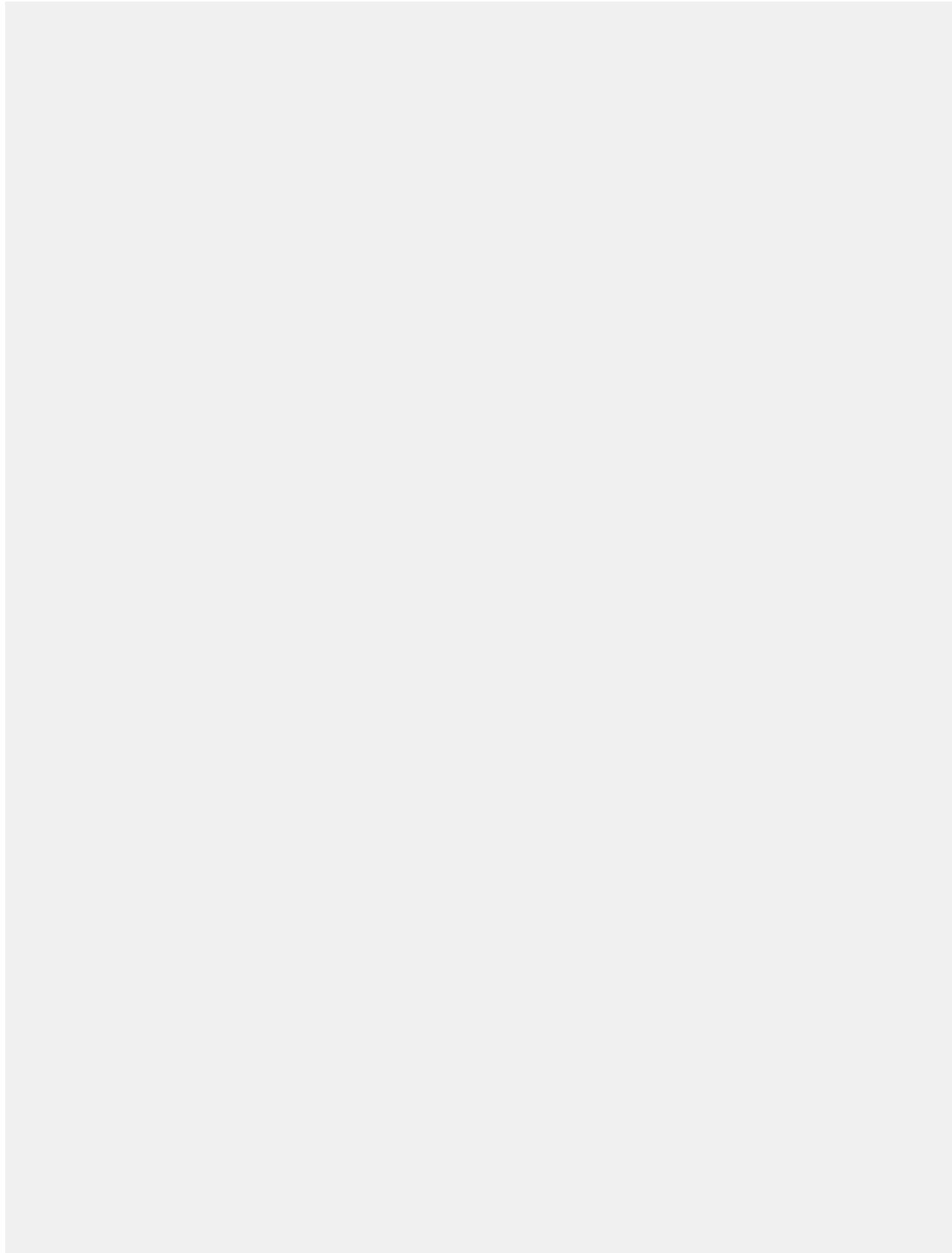
## Apex による共有管理の再適用の例

この例では、人事採用アプリケーションの構築中で、Job というオブジェクトが存在すると仮定しています。ジョブにリストされた採用担当者および採用担当マネージャにレコードへのアクセス権が付与されていることを確認

したいと考えています。次の Apex クラスでこの検証を実行できます。この例では、User レコードと関連付けられた、Hiring\_Manager および Recruiter という 2 つの参照項目を持つ Job というカスタムオブジェクトが必要です。また、Job カスタムオブジェクトには、Hiring\_Manager と Recruiter という 2 つの共有の理由を追加する必要があります。このサンプルを実行する前に、メールアドレスを、エラー通知とジョブ完了通知を送信する有効なメールアドレスに置き換えます。

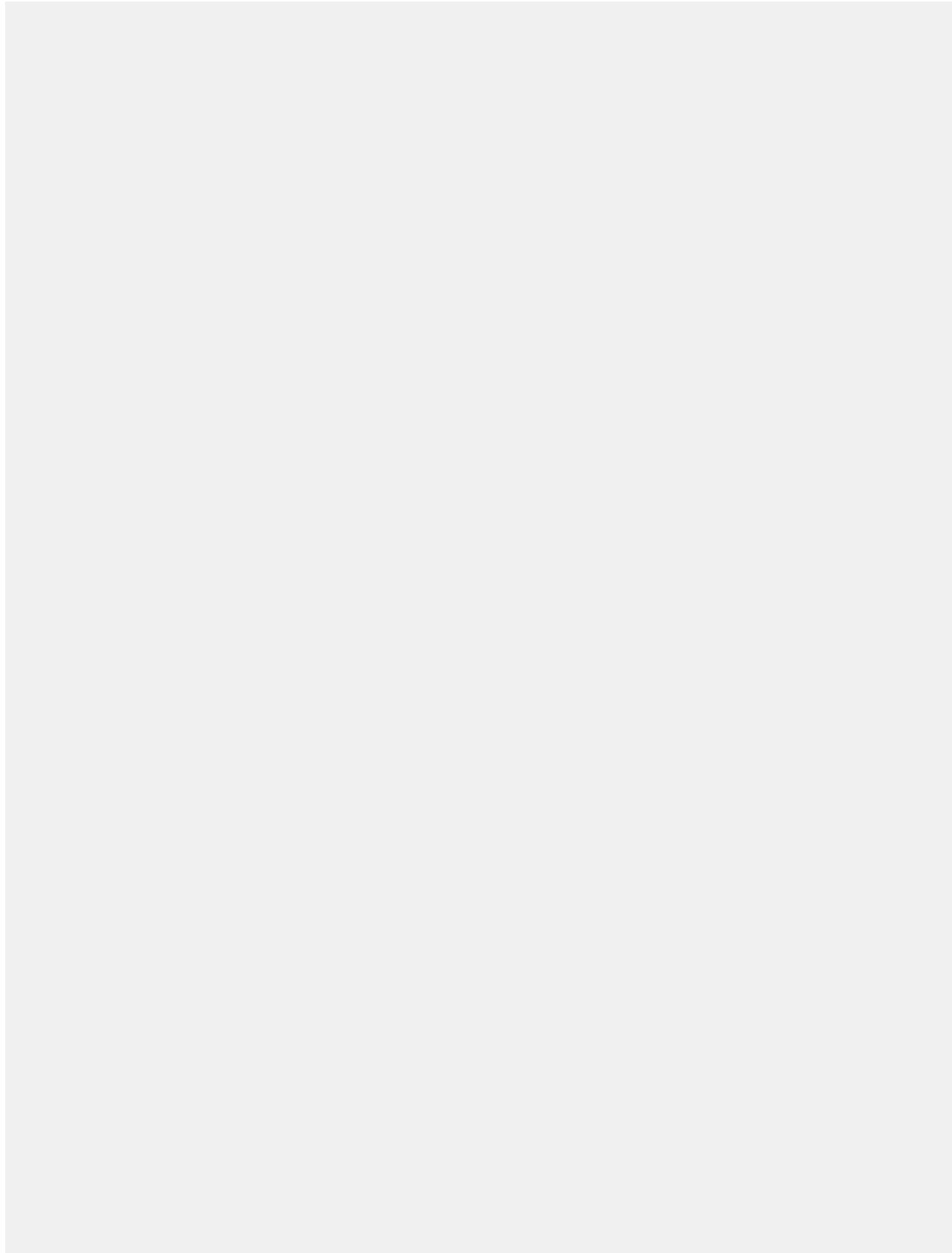


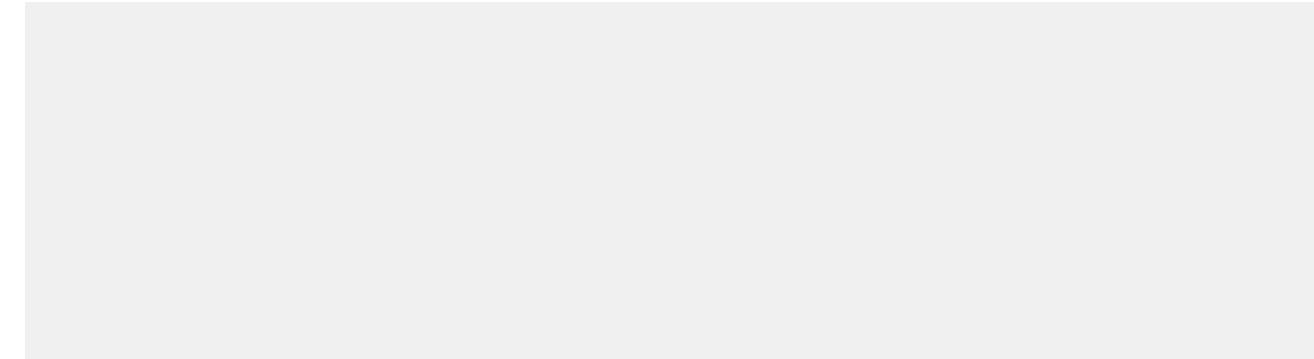




## Apex による共有管理の再適用のテスト

この例では、5つのJobレコードを挿入し、前の例で使用した一括処理クラスに実装される一括処理ジョブを呼び出します。この例では、Userレコードと関連付けられた、Hiring\_ManagerおよびRecruiterという2つの参照項目を持つJobというカスタムオブジェクトが必要です。また、Jobカスタムオブジェクトには、Hiring\_ManagerとRecruiterという2つの共有の理由を追加する必要があります。このテストを実行する前に、組織全体のJobのデフォルト共有設定を[非公開]に設定します。テストからはメールメッセージは送信されないため、また、一括処理クラスはテストメソッドによって呼び出されるため、この場合、メール通知は送信されません。





## 再適用に使用される Apex クラスの関連付け

再適用に使用される Apex クラスはカスタムオブジェクトと関連付けられている必要があります。

Apex による共有管理の再適用クラスをカスタムオブジェクトと関連付ける手順は、次のとおりです。

1. [設定] で、[作成] > [オブジェクト] をクリックします。
2. カスタムオブジェクトを選択します。
3. [Apex 共有の再適用] 関連リストで [新規] をクリックします。
4. このオブジェクトの Apex 共有を再適用する Apex クラスを選択します。選択するクラスは、インターフェースを実装している必要があります。同じ Apex クラスを、同じカスタムオブジェクトと複数関連付けることはできません。
5. [保存] をクリックします。

## 第8章

### Chatter in Apex の使用

#### トピック:

- ConnectApi 入力および出力クラスの使用
- コミュニティでの ConnectApi データへのアクセス
- ConnectApi クラスの制限について
- ConnectApi オブジェクトの逐次化と並列化
- ConnectApi バージョン設定と同等性チェック
- ConnectApi オブジェクトのキャスト
- ワイルドカードの使用
- ConnectApi クラスのテスト
- ConnectApi クラスとその他の Apex クラスの違い

名前空間の Apex クラスでは多くの Chatter REST API リソースアクションが静的メソッドとして公開されています。これらのメソッドでは、情報を入力したり返したりするために他のクラスが使用されます。名前空間は、*Chatter in Apex*とも呼ばれます。

ネイティブなソーシャル Force.com アプリケーションの開発には、Chatter in Apex を使用します。フィードを表示する Visualforce ページを作成し、メンションおよびトピックを含むフィード項目を投稿し、ユーザおよびグループの写真を更新します。Chatter フィードを更新するトリガを作成します。クラスを使用すると、Chatter Web ユーザインターフェースで可能なほとんどすべての操作を行うことができます。

Apex では、SOQL クエリとオブジェクトを使用して、一部の Chatter データにアクセスできます。ただし、クラスでは Chatter データがより単純な方法で公開されます。データは、表示用にローカライズされ、構成されます。たとえば、フィードへのアクセスや作成を、多くのコールではなく 1 回のコールで行うことができます。

## ConnectApi 入力および出力クラスの使用

名前空間のクラスには、Chatter REST API データにアクセスする静的メソッドが含まれるものがあります。名前空間には、パラメータを渡す入力クラスと、静的メソッドへのコールによって返される出力クラスも含まれます。

メソッドは、単純なデータ型または複雑なデータ型のどちらかを取ります。単純なデータ型とは、整数や文字列などのプリミティブ Apex データです。複雑なデータ型とは、入力オブジェクトです。

メソッドの実行が成功すると、名前空間から出力オブジェクトが返される場合があります。出力オブジェクトは、他の出力オブジェクトで構成されることがあります。たとえば、には、やなどの単純なプロパティと、サブオブジェクトも含まれます。

「[入力クラス](#)」(ページ 744)および「[出力クラス](#)」(ページ 749)を参照してください。

### 関連リンク

[Chatter in Apex の使用](#)

[コミュニティでの ConnectApi データへのアクセス](#)

[ConnectApi クラスの制限について](#)

[ConnectApi オブジェクトの逐次化と並列化](#)

[ConnectApi バージョン設定と同等性チェック](#)

[ConnectApi オブジェクトのキャスト](#)

[ワイルドカードの使用](#)

[ConnectApi クラスのテスト](#)

[ConnectApi クラスとその他の Apex クラスの違い](#)

## コミュニティでの ConnectApi データへのアクセス

すべてのメソッドは、1 つのコミュニティのコンテキスト内で機能します。

多くのメソッドは、最初の引数として `communityId` を取ります。コミュニティを有効化していない場合、このパラメータにはまたはを使用します。

コミュニティを有効化している場合、`communityId` 引数には、メソッドをデフォルトのコミュニティのコンテキストで実行するか(またはを指定)、または特定のコミュニティのコンテキストで実行するか(コミュニティ ID を指定)を指定します。コメント、フィード項目など、メソッド内の他の引数で参照されるエンティティは、指定されたコミュニティ内に存在する必要があります。指定されたコミュニティ ID は、出力で返されるすべての URL で使用されます。

コミュニティ ID を指定した場合、出力で返される URL では次の形式が使用されます。

`communityId resource`

を指定した場合、出力で返される URL では同じ形式が使用されます。

`resource`

を指定した場合、出力で返される URL では次のいずれかの形式が使用されます。

`resource`

`resource`

これらの URL は、Chatter REST API エンドポイントです。

## 関連リンク

[Chatter in Apex の使用](#)

[ConnectApi 入力および出力クラスの使用](#)

[ConnectApi クラスの制限について](#)

[ConnectApi オブジェクトの逐次化と並列化](#)

[ConnectApi バージョン設定と同等性チェック](#)

[ConnectApi オブジェクトのキャスト](#)

[ワイルドカードの使用](#)

[ConnectApi クラスのテスト](#)

[ConnectApi クラスとその他のApex クラスの違い](#)

## ConnectApi クラスの制限について

名前空間内のクラスの制限は、他の Apex クラスの制限とは異なります。

名前空間内のクラスの場合、各書き込み操作が Apex ガバナ制限で 1 回の DML 操作としてカウントされます。

メソッドコールも、レート制限の対象となります。 レート制限は、Chatter REST API レート制限と同じです。どちらにも、ユーザごと、名前空間ごと、時間ごとのレート制限があります。

レート制限を超えると、  
処理する必要があります。

が発生します。Apex で、この例外をキャッチして

コードのテスト時、Apex メソッドのコールにより、レート制限が新しく開始します。  
メソッドのコールにより、レート制限は をコールする前の値に設定されます。

## 関連リンク

- [Chatter in Apex の使用](#)
- [ConnectApi 入力および出力クラスの使用](#)
- [コミュニティでの ConnectApi データへのアクセス](#)
- [ConnectApi オブジェクトの逐次化と並列化](#)
- [ConnectApi バージョン設定と同等性チェック](#)
- [ConnectApi オブジェクトのキャスト](#)
- [ワイルドカードの使用](#)
- [ConnectApi クラスのテスト](#)
- [ConnectApi クラスとその他の Apex クラスの違い](#)

## ConnectApi オブジェクトの逐次化と並列化

出力オブジェクトを JSON に逐次化すると、Chatter REST API から返される JSON と類似した構造になります。 入力オブジェクトを JSON から並列化した場合も、Chatter REST API と類似した構造になります。

Chatter in Apex は、次の Apex コンテキストで逐次化と並列化をサポートします。

- および クラス — Chatter in Apex 出力を JSON に逐次化、および Chatter in Apex 入力を JSON から並列化。
- を含む Apex REST — Chatter in Apex 出力を JSON を戻り値として逐次化、および JSON をパラメータとして Chatter in Apex 入力に並列化。
- を含む JavaScript Remoting — Chatter in Apex 出力を JSON を戻り値として逐次化、および JSON をパラメータとして Chatter in Apex 入力に並列化。

Chatter in Apex は、次の逐次化および並列化ルールに従います。

- 逐次化できるのは出力オブジェクトのみです。
- 並列化できるのは最上位の入力オブジェクトのみです。現在、並列化できる最上位オブジェクトは、のみです。

- Enum 値および例外は、逐次化も並列化もできません。

## 関連リンク

- [Chatter in Apex の使用](#)
- [ConnectApi 入力および出力クラスの使用](#)
- [コミュニティでの ConnectApi データへのアクセス](#)
- [ConnectApi クラスの制限について](#)
- [ConnectApi バージョン設定と同等性チェック](#)
- [ConnectApi オブジェクトのキャスト](#)
- [ワイルドカードの使用](#)
- [ConnectApi クラスのテスト](#)
- [ConnectApi クラスとその他の Apex クラスの違い](#)

## ConnectApi バージョン設定と同等性チェック

クラスのバージョン設定は、他の Apex クラスとは大きく異なる特定のルールに従います。

クラスのバージョン設定は次のルールに従います。

- メソッドコールは、そのメソッドコールが含まれるクラスのバージョンのコンテキスト内で実行されます。バージョンの使用は、Chatter API URL の `xx.x` セクションと似ています。
- 各 出力オブジェクトは メソッドを公開します。このメソッドは、出力オブジェクトを作成したメソッドが呼び出されたバージョンを返します。
- 入力オブジェクトを操作するとき、Apex でアクセスできるのは、それを囲む Apex クラスのバージョンでサポートされるプロパティのみです。
- メソッドに渡される入力オブジェクトには、そのメソッドを実行する Apex クラスのバージョンでサポートされる、null 以外のプロパティのみが含まれる場合があります。入力オブジェクトにバージョン不適合のプロパティが含まれていると、例外が発生します。
- メソッドは、オブジェクトを操作するコードのバージョンでサポートされているプロパティのみを返します。出力オブジェクトの場合、返されるプロパティもビルトのバージョンでサポートされている必要があります。
- Apex REST、 および の逐次化には、バージョンに適合するプロパティのみが含まれます。
- Apex REST、 および の並列化は、バージョン不適合のプロパティを拒否します。

クラスの同等性チェックは次のルールに従います。

- 入力オブジェクト — プロパティが比較されます。

- 出力オブジェクト — プロパティとビルドのバージョンが比較されます。たとえば、2つのオブジェクトの同じプロパティが同じ値で、ビルドのバージョンが異なる場合、オブジェクトは同等ではありません。ビルドのバージョンを取得するには、  
をコールします。

## 関連リンク

[Chatter in Apex の使用](#)

[ConnectApi 入力および出力クラスの使用](#)

[コミュニティでの ConnectApi データへのアクセス](#)

[ConnectApi クラスの制限について](#)

[ConnectApi オブジェクトの逐次化と並列化](#)

[ConnectApi オブジェクトのキャスト](#)

[ワイルドカードの使用](#)

[ConnectApi クラスのテスト](#)

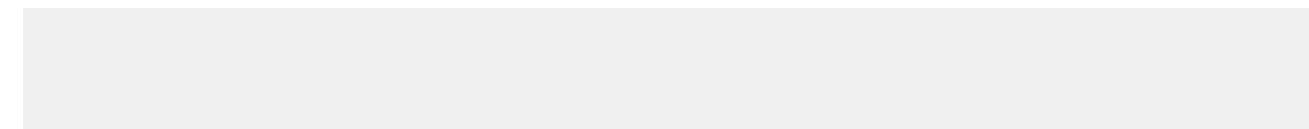
[ConnectApi クラスとその他の Apex クラスの違い](#)

## ConnectApi オブジェクトのキャスト

出力オブジェクトをより特定の型にダウンキャストすると便利な場合があります。

この方法は、メッセージセグメントとフィード項目添付ファイルで特に便利です。フィード項目のメッセージセグメントは、型です。フィード項目添付ファイルは型です。これらはどちらも抽象クラスで、複数の具象サブクラスがあります。実行時、を使用すると、これらのオブジェクトの具象型をチェックして、対応するダウンキャストを確実に実行することができます。ダウンキャスト時には、不明なサブクラスを処理するデフォルトの case が必要です。

次の例では、をにダウンキャストします。



重要: フィードの構成は、リリースによって異なる場合があります。コードでは、常にオブジェクトとオブジェクトの両方の不明なサブクラスを処理できるように準備しておく必要があります。

「[ConnectApi.MessageSegment クラス](#)」および「[ConnectApi.FeedItemAttachment クラス](#)」を参照してください。

## 関連リンク

- [Chatter in Apex の使用](#)
- [ConnectApi 入力および出力クラスの使用](#)
- [コミュニティでの ConnectApi データへのアクセス](#)
- [ConnectApi クラスの制限について](#)
- [ConnectApi オブジェクトの逐次化と並列化](#)
- [ConnectApi バージョン設定と同等性チェック](#)
- [ワイルドカードの使用](#)
- [ConnectApi クラスのテスト](#)
- [ConnectApi クラスとその他の Apex クラスの違い](#)

## ワイルドカードの使用

Chatter REST API と Chatter in Apex の検索でテキストパターンを一致させるには、ワイルドカード文字を使用します。

ワイルドカードが一般的に使用されるのはフィードを検索するときです。 パラメータで検索文字列とワイルドカードを渡します。次の例は、Chatter REST API 要求です。

次の例は、Chatter in Apex メソッドコールです。

検索内のテキストパターンと一致させるために、次のワイルドカード文字を指定できます。

ワイルドカード	説明
*	検索語の途中または末尾で、0個以上の文字の代わりにアスタリスクを使用できます。標準ルックアップ検索を実行する場合以外は、検索語の先頭にアスタリスクを使用しないでください。たとえば、「太」を検索すると、「太一」、「太郎」、「太次郎」などの「太」で始まるデータが表示されます。ただし、中国語、日本語、韓国語、またはタイ語で検索する場合は、検索語の中間にアスタリスクまたは疑問符のワイルドカードは使用できません。単語または語句内のリテラルアスタリスクを検索する場合、アスタリスクをエスケープします（文字をその前に付けます）。
?	疑問符は、検索語の途中または末尾（先頭ではない）にある1つのみの文字の代わりに使用できます。たとえば、「？」を検索すると、「john」や「joan」を含むデータが表示されます。ただし、中国語、日本語、韓国語、またはタイ語で検索する場合は、検索語の中間にアスター

## ワイルドカード 説明

スクまたは疑問符のワイルドカードは使用できません。また、検索語の先頭にワイルドカードの疑問符を使用しても機能しません。

ワイルドカードを使用する場合には、以下の点に注意してください。

- ワイルドカードは先行する文字の種類を表します。たとえば、「\_\_\_\_\_」は「aaaaa」と「aabccda」に一致しますが、「aa2a」や「aa.//a」には一致せず、「\_\_\_\_\_」は「pin」と「pan」には一致しますが、「p!n」や「p/n」には一致しません。同様に、「\_\_\_\_\_」は「123」と「143」には一致しますが、「1a3」や「1b3」には一致しません。
- ワイルドカード(\*)は、中国語、日本語、韓国語、タイ語 (CJKT) での検索で、完全に一致する語句の検索以外では、検索文字の最後に追加します。
- ワイルドカード検索の条件を絞り込むほど、検索結果はより速く返され、期待する結果が返される可能性が高まります。たとえば、単語 \_\_\_\_\_ (または複数形 \_\_\_\_\_) のすべての発生を検索するには、無関係の一致( \_\_\_\_\_ など)を返す可能性のある制限のより少ないワイルドカード検索( \_\_\_\_\_ など)を指定するよりも、検索文字列内で \_\_\_\_\_ を指定する方がより効率的です。
- 単語のすべてのバリエーションを見つけるために、検索を調整します。たとえば、\_\_\_\_\_と \_\_\_\_\_を見つけるには、\_\_\_\_\_を指定します。
- 句読点にはインデックスを付けます。語句内で \_\_\_\_\_ または \_\_\_\_\_ を見つけるためには、検索文字列を引用符で囲む必要があります、特殊文字をエスケープする必要があります。たとえば、\_\_\_\_\_は、語句 \_\_\_\_\_を見つけます。エスケープ文字( )は、この検索が正しく機能するために必要です。

## 関連リンク

[Chatter in Apex の使用](#)

[ConnectAPI 入力および出力クラスの使用](#)

[コミュニティでの ConnectAPI データへのアクセス](#)

[ConnectAPI クラスの制限について](#)

[ConnectAPI オブジェクトの逐次化と並列化](#)

[ConnectAPI バージョン設定と同等性チェック](#)

[ConnectAPI オブジェクトのキャスト](#)

[ConnectAPI クラスのテスト](#)

[ConnectAPI クラスとその他の Apex クラスの違い](#)

## ConnectApi クラスのテスト

Apex コードと同様に、Chatter in Apex コードにはテストカバー率が必要です。

Chatter in Apex コードは、システムモードでは実行されず、現在のユーザのコンテキストで実行されます。コードは、現在のユーザがアクセス権を持つものすべてにアクセスできます。Chatter in Apex は、システムメソッドをサポートしていません。

大部分の Chatter in Apex メソッドコールには、実際の組織データへのアクセス権が必要であり、  
のマークが付けられていないテストメソッドで使用されると失敗します。

など、一部の Chatter in Apex メソッドでは、テストで組織データにアクセスすることはできません。テキストコンテキストで返されるように出力を登録する特別なテストメソッドとともに使用する必要があります。テストメソッド名は、通常のメソッド名の先頭に  が付けられた名前になります。テストメソッドには、通常のメソッドのすべての署名と一致する署名(引数の組み合わせ)があります。通常のメソッドに 3 つのオーバーロードがある場合、テストメソッドには 3 つのオーバーロードがあります。

Chatter in Apex テストメソッドの使用方法は、Apex での Web サービスのテスト方法と類似しています。まず、メソッドで返すデータを作成します。データを作成するには、出力オブジェクトを作成し、それらのプロパティを設定します。すべての非抽象出力クラスに非引数コンストラクタを使用して、オブジェクトを作成できます。

データを作成したら、テストメソッドをコールして、データを登録します。テストする通常のメソッドと同じ署名を持つテストメソッドをコールします。

テストデータを登録したら、通常のメソッドを実行します。通常のメソッドを実行すると、一致する引数に登録された値が返されます。

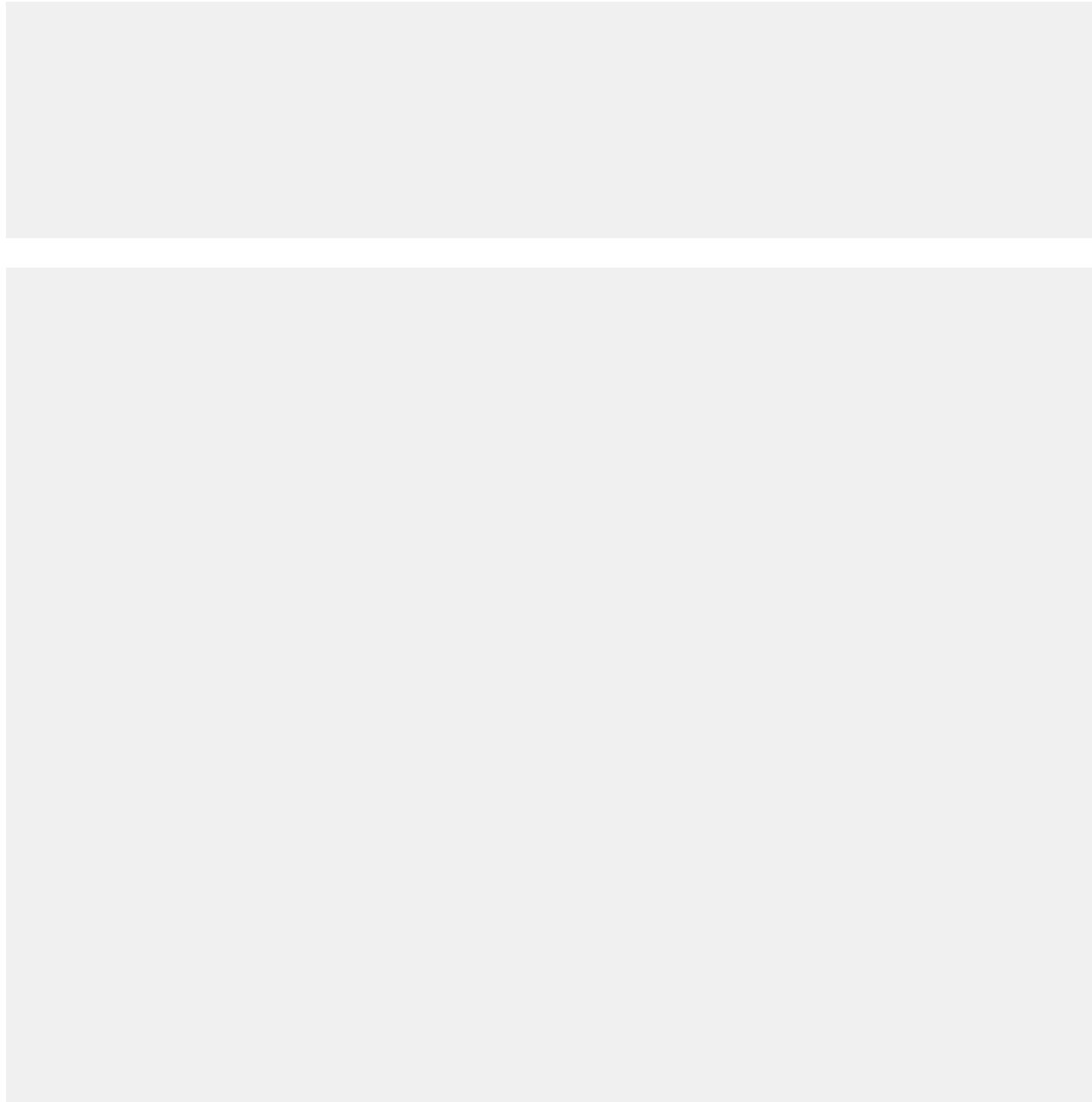
 **重要:** テストメソッドの署名には、通常のメソッドの署名と一致するものを使用する必要があります。通常のメソッドをコールするときに、一致する引数セットを使用してデータが登録されていなかった場合、例外を受け取ります。

次に表に、テストメソッドに関連づけられているメソッドを示します。

Get メソッド	Test メソッド

次の例は、

を構成し、特定の組み合わせのパラメータを指定して  
がコールされたときにそれが返されるように登録するテストを示しています。



## 関連リンク

- [Chatter in Apex の使用](#)
- [ConnectApi 入力および出力クラスの使用](#)
- [コミュニティでの ConnectApi データへのアクセス](#)
- [ConnectApi クラスの制限について](#)
- [ConnectApi オブジェクトの逐次化と並列化](#)
- [ConnectApi バージョン設定と同等性チェック](#)
- [ConnectApi オブジェクトのキャスト](#)
- [ワイルドカードの使用](#)
- [ConnectApi クラスとその他の Apex クラスの違い](#)

## ConnectApi クラスとその他の Apex クラスの違い

クラスとその他の Apex クラスには、さらに次のような違いがあります。

### システムモードとコンテキストユーザ

Chatter in Apex コードは、システムモードでは実行されず、現在のユーザのコンテキストで実行されます。コードは、現在のユーザがアクセス権を持つものすべてにアクセスできます。Chatter in Apex は、システムメソッドをサポートしていません。`subjectId` 引数を取るメソッドの多くで、その対象はコンテキストユーザである必要があります。これらの場合、ID の代わりに文字列 `_` を使用してコンテキストユーザを指定できます。

### `with sharing` と `without sharing`

Chatter in Apex は、`with sharing` および `without sharing` キーワードを無視します。代わりに、すべてのセキュリティ、項目レベルの共有、および表示設定は、コンテキストユーザによって制御されます。たとえば、コンテキストユーザが非公開グループのメンバーである場合、`Post` クラスは、そのグループに投稿できます。コンテキストユーザが非公開グループのメンバーではない場合、コードはそのグループのフィード項目を参照できず、グループへの投稿はできません。

### 非同期操作

一部の Chatter in Apex 操作は非同期、つまりすぐには行われません。たとえば、コードでユーザに対するフィード項目を追加しても、ニュースフィードですぐには使用できません。また、写真を追加しても、すぐには使用できません。テストの場合は、写真を追加してもすぐには取得できることになります。

### Apex REST では XML がサポートされない

Apex REST では、Chatter in Apex オブジェクトの XML 逐次化および並列化はサポートされません。Apex REST では、Chatter in Apex オブジェクトの JSON 逐次化および並列化はサポートされません。

### 空のログエントリ

Chatter in Apex 変数は、ログイベントには表示されません。

### Apex SOAP Web サービスがサポートされない

Chatter in Apex オブジェクトは、キーワードで指示された Apex SOAP Web サービス内では使用できません。

## 関連リンク

[Chatter in Apex の使用](#)

[ConnectApi 入力および出力クラスの使用](#)

[コミュニティでの ConnectApi データへのアクセス](#)

[ConnectApi クラスの制限について](#)

[ConnectApi オブジェクトの逐次化と並列化](#)

[ConnectApi バージョン設定と同等性チェック](#)

[ConnectApi オブジェクトのキャスト](#)

[ワイルドカードの使用](#)

[ConnectApi クラスのテスト](#)

## 第9章

### Apex のデバッグ

トピック:

- ・ [デバッグログについて](#)
- ・ [検出されなかった例外の処理](#)
- ・ [実行ガバナと制限について](#)
- ・ [ガバナ制限のメール警告の使用](#)

Apex では、デバッグサポートが提供されます。開発者コンソールとデバッグログを使用して Apex コードをデバッグできます。デバッグをさらに支援するために、未対応の例外について Apex から開発者にメールが送信されます。また、Apex では、実行コードに特定のガバナ制限のセットを適用し、共有リソースがマルチテナント環境で独占されないようにします。ガバナ制限の一定割合を超えるコードを実行しているエンドユーザにメールを送信するように選択することもできます。

この章では、次の内容を取り上げます。

- ・ [デバッグログについて](#)
- ・ [検出されなかった例外の処理](#)
- ・ [実行ガバナと制限について](#)
- ・ [ガバナ制限のメール警告の使用](#)

## デバッグログについて

デバッグログには、データベースの操作、システムプロセス、トランザクションの実行時または単体テストの実行中に発生したエラーを記録できます。デバッグログには、次の情報を記載できます。

- データベースの変更
- HTTP コールアウト
- Apex エラー
- Apex によって使用されるリソース
- 次のような自動化されたワークフロー処理:
  - ◊ ワークフロールール
  - ◊ 割り当てルール
  - ◊ 承認プロセス
  - ◊ 入力規則

特定ユーザのデバッグログを保持および管理できます。

保存されたデバッグログを参照するには、[設定] から [監視] > [デバッグログ] または [ログ] > [デバッグログ] をクリックします。

デバッグログに関する制限は次のとおりです。

- ユーザを追加すると、そのユーザは最大 20 個のデバッグログを記録できます。ユーザがこの制限値に達すると、そのユーザについてはデバッグログの記録を停止します。[デバッグログの監視] ページで [リセット] をクリックすると、該当するユーザのログ数を 20 にリセットします。既存のログは上書きされません。
- 各デバッグログは最大 2 MB です。デバックログのサイズが 2 MB を超えると、ステートメントの始めの方のログの行など、古いログの行が削除されてサイズが縮小されます。ログの行は、デバッグログの最初からだけでなく、どの位置からでも削除できます。
- 各組織は最大 50 MB のデバッグログを保持できます。組織のデバッグログが 50 MB に達すると、最も古いデバッグログから上書きされます。

## デバッグログセクションの調査

デバッグログを生成した後、表示される情報の種類や量は、ユーザに設定した検索値によって異なります。ただし、デバッグログの形式は常に同じです。

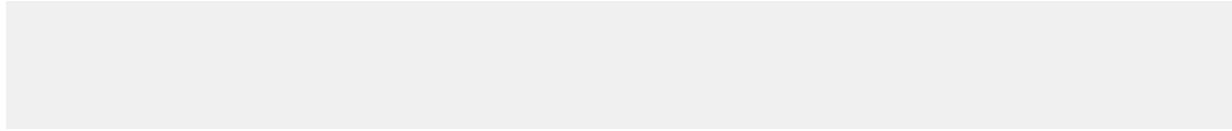
デバッグログには、次のセクションがあります。

### ヘッダー

ヘッダーには、次の情報が含まれます。

- トランザクションで使用される API のバージョン。
- ログの生成に使用されるログのカテゴリとレベル。次に例を示します。

次に、ヘッダーの例を示します。



この例では、API バージョンは 25.0 で、次のデバッグログカテゴリおよびレベルが設定されています。

Apex コード	DEBUG
Apex プロファイリング	INFO
コールアウト	INFO
データベース	INFO
システム	DEBUG
入力規則	INFO
Visualforce	INFO
ワークフロー	INFO

## 実行ユニット

実行ユニットは、トランザクションと同じです。実行ユニットには、トランザクション内で発生したすべての情報が含まれています。実行は、  
と  
に区切られます。

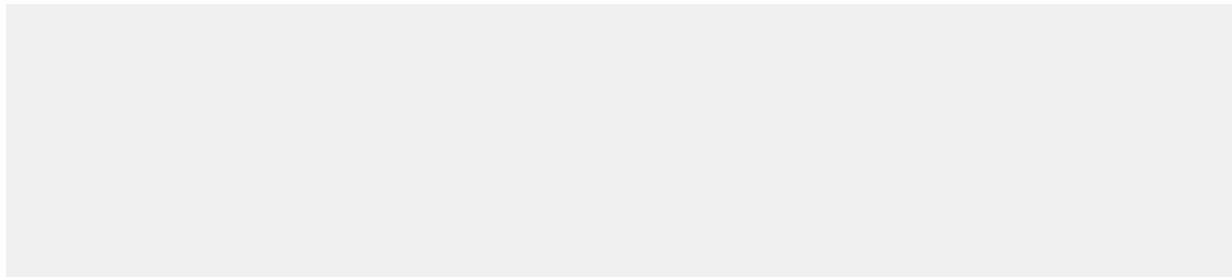
## コードユニット

コードユニットは、トランザクション内の作業単位です。たとえば、  
メソッド、入力規則であるトリガはコードの単位の 1 つです。



メモ: クラスはコードの単位ではありません。

コードの単位は、  
や  
の単位を組み込むことができます。次に例を示します。

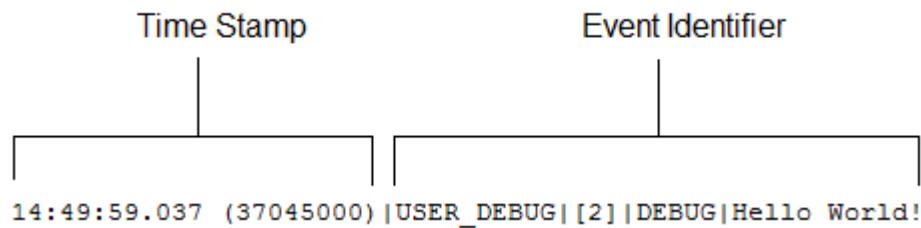


コードの単位には、次が含まれますが、これらには限定されません。

- ・ トリガ
- ・ ワークフローの呼び出しおよび時間ベースのワークフロー
- ・ 入力規則
- ・ 承認プロセス
- ・ Apex リードの変換
- ・ メソッドの呼び出し
- ・ Web サービスの呼び出し
- ・ コール
- ・ Apex コントローラの Visualforce プロパティアクセス
- ・ Apex コントローラの Visualforce アクション
- ・ メソッドの各実行のほか、Apex メソッドや メソッドの一括実行
- ・ Apex メソッドの実行
- ・ 受信メールの処理

### ログの行

ログの行は、コードユニット内に含まれ、実行されたコードやルールを示します。ログの行は、デバッグログ専用に書き込まれたメッセージである場合もあります。次に例を示します。



ログの行は、一連の項目で構成され、項目はパイプ( )で区切られます。形式は次のとおりです。

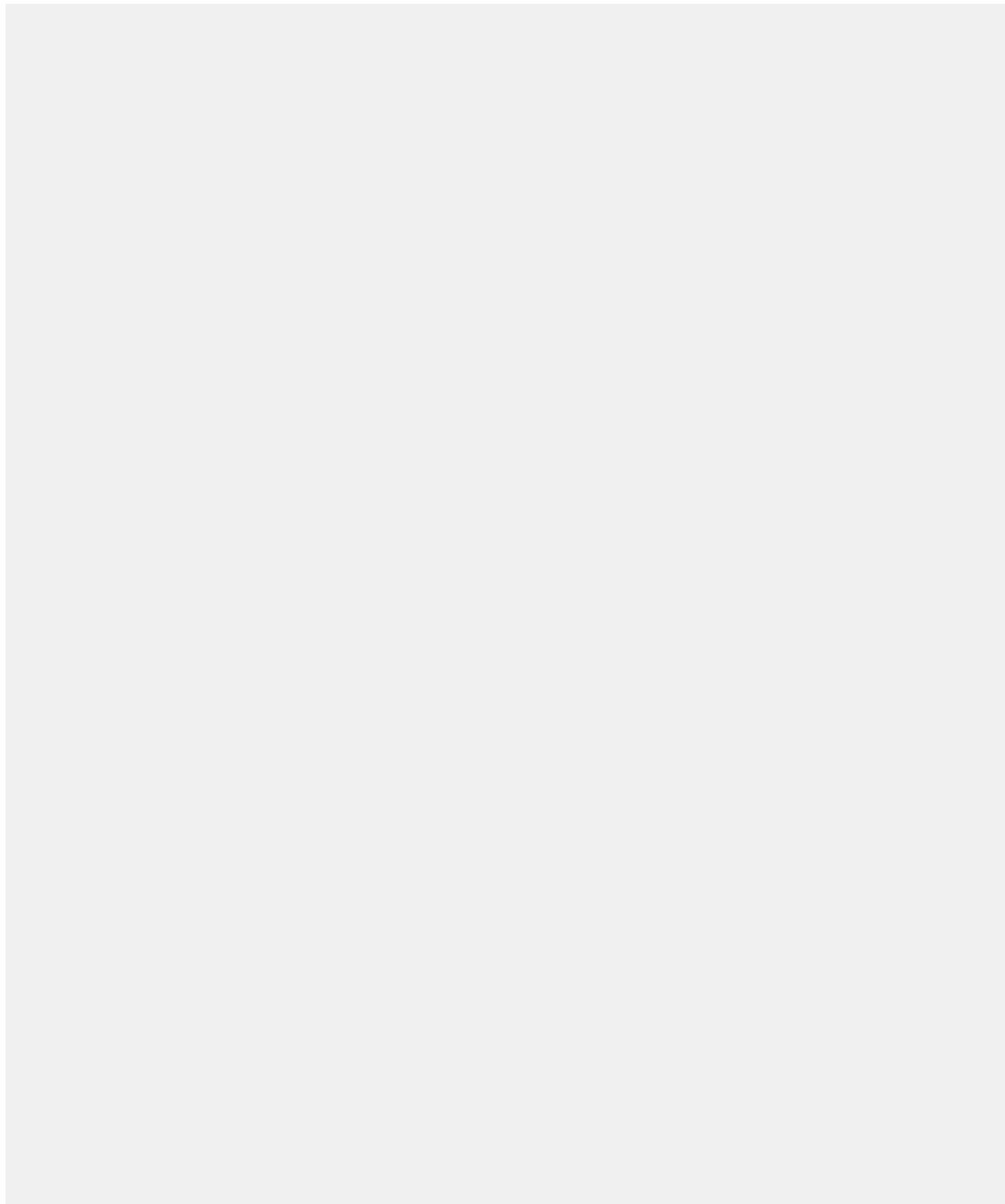
- ・ タイムスタンプ: イベント発生時の時刻とかっこで囲まれた値で構成されます。時刻はユーザのタイムゾーンで、形式は *HH:mm:ss.SSS* となります。値は、要求が開始されてからの経過時間をナノ秒単位で表します。経過時間の値は、開発者コンソールで確認したログには含まれません。
- ・ イベント識別子: や など、書き込まれるデバッグログをトリガした特定のイベント、およびメソッド名またはコードが実行された行番号および文字番号など、そのイベントと共に記録された追加情報で構成されます。

### 追加ログデータ

さらに、ログには次の情報が含まれます。

- ・ 累積リソース使用状況は、トリガ、 Apex のメッセージの一括処理、 メソッド、 Apex テストメソッド、 Apex Web サービスマソッド、 Apex リードの取引開始など、多くのコードユニットの終わりに記録されます。
- ・ 累積プロファイル情報は、トランザクションの終わりに 1 回記録され、最もコストのかかるクエリ (最も多くのリソースを使用)、 DML の呼び出しなどの情報が含まれます。

次に、デバッグログの例を示します。



## Apex クラスおよびトリガ用デバッグルогの検索条件の設定

デバッグルогの検索条件により、ログの冗長性をトリガおよびクラスレベルで微調整できます。これは Apex ロジックのデバッグルог時に特に便利です。たとえば、複雑なプロセスの出力を評価する場合、1 つの要求内で特定のクラスのログの冗長性を上げ、他のクラスまたはトリガのログをオフにすることができます。

クラスまたはトリガのデバッグルогレベルを上書きすると、これらのデバッグルогレベルは、クラスまたはトリガが呼び出すクラスメソッドと、結果として実行されるトリガにも適用されます。実行パス内のすべてのクラスメソッドとトリガは、これらのデバッグルог設定を上書きする場合を除き、コード元から設定を継承します。

次の図は、クラスおよびトリガレベルで上書きされるデバッグルогレベルを示しています。このシナリオでは、**Trigger1** が何らかの問題を起こし、詳しい調査が必要であるとします。このために、**Trigger1** のデバッグルогレベルが最も詳細なレベルに引き上げられます。**Class1** ではこのログレベルが上書きされないため、**Class1** の詳細なログレベルが継承されます。ただし、**Class1** はすでにテストされ、正しく動作することがわかっているため、ログの検索条件はオフになっています。同様に、**Trigger2** は問題の原因であるコードパスに含まれていないため、ログは最小限に抑えられ、Apex コードカテゴリのエラーのみを記録します。**UtilityClass** は、からこれらのログ設定を継承します。

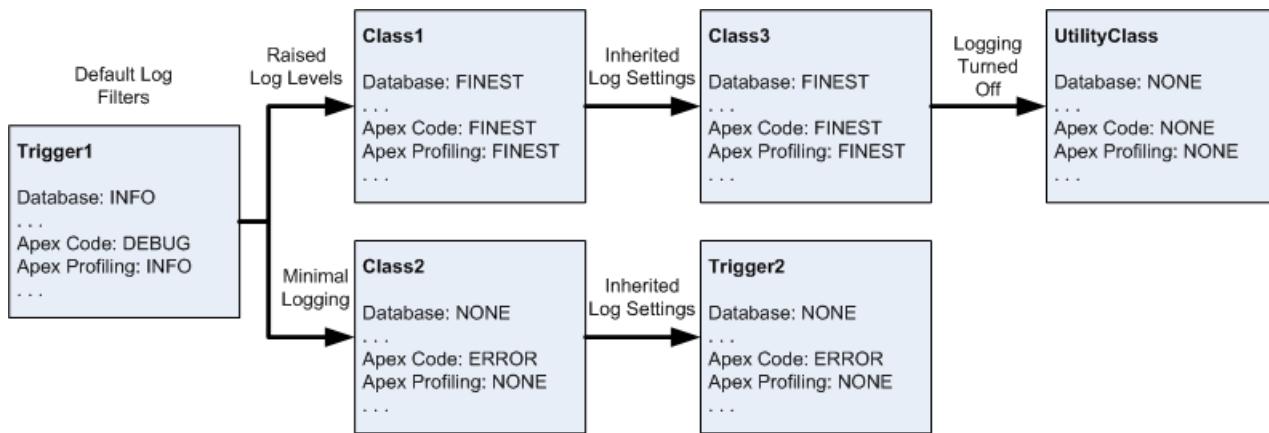
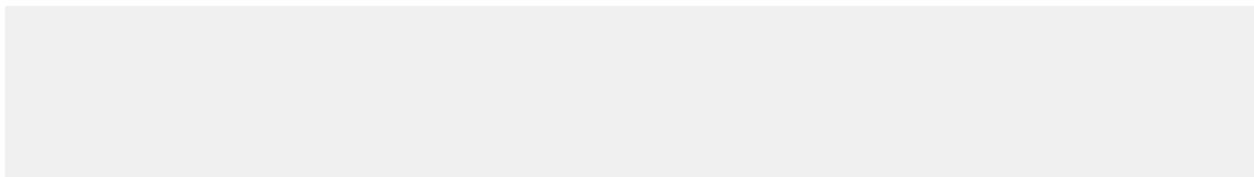


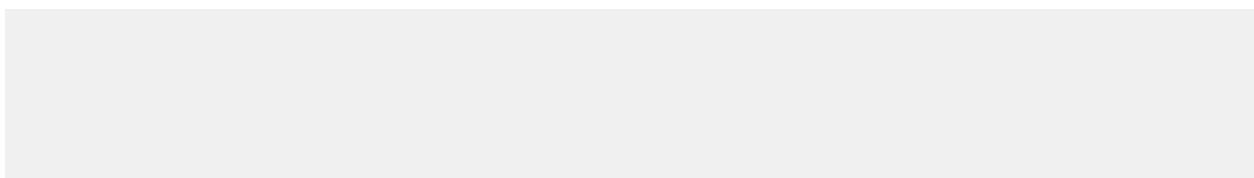
図 5: クラスおよびトリガ用デバッグルогの微調整

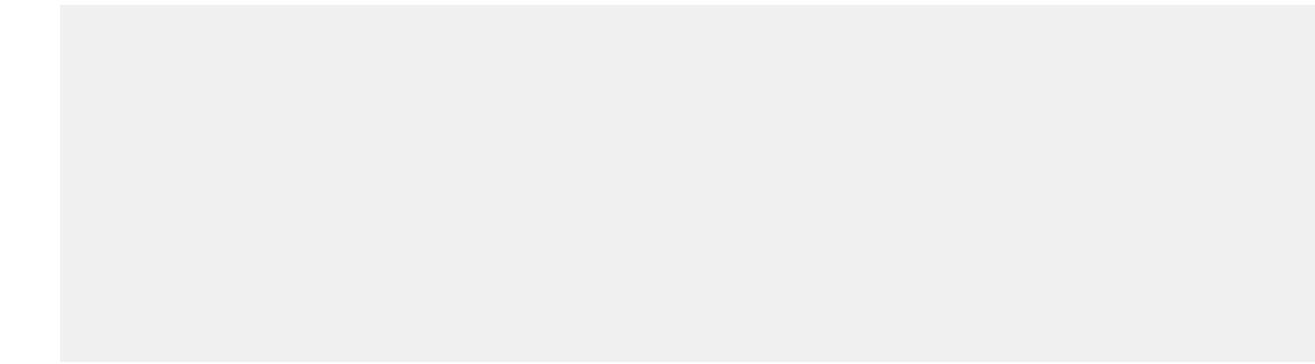
この図のもとになった擬似コード例を次に示します。

1. **Trigger1** が **UtilityClass** のメソッドと **UtilityClass** の別のメソッドをコールします。次に例を示します。

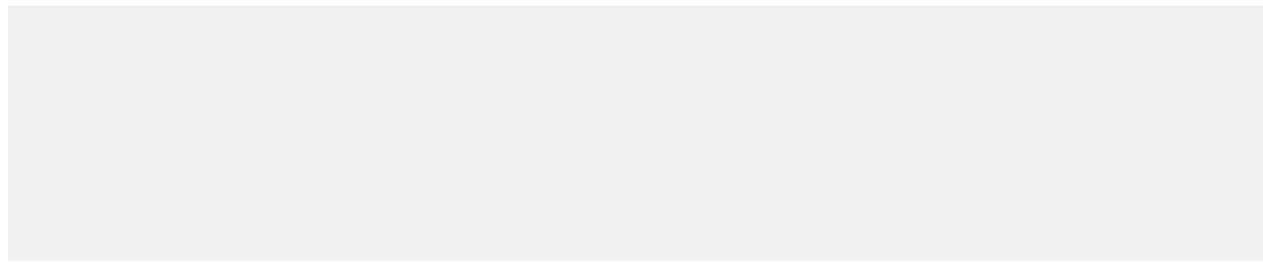


2. **Trigger1** は **UtilityClass** のメソッドをコールし、このメソッドが次にユーティリティクラスのメソッドをコールします。次に例を示します。





3. によってトリガ の実行が起動されます。次に例を示します。



ログの検索条件を設定する手順は、次のとおりです。

1. クラスまたはトリガの詳細ページから、[ログの検索条件] をクリックします。
2. [ログの検索条件の上書き] をクリックします。

ログの検索条件は、デフォルトのログレベルに設定されます。

3. 各ログカテゴリに適したログレベルを選択します。

デバッグルогカテゴリ、デバッグルогレベル、およびデバッグルогイベントについての詳細は、「デバッグルогの検索条件の設定」を参照してください。

## 関連リンク

[開発者コンソールのログの操作](#)

[Apex API コールのデバッグ](#)

## 開発者コンソールのログの操作

デバッグルогを開くには、開発者コンソールの [log (ログ)] タブを使用します。

User	Application	Operation	Time	Status	Read	Size
JS	Browser	/ui/common/apex/debu...	04/09 13:38:46	Success		39132

Filter [Click here to filter the log list](#)

ログはシステムログビューで開きます。ログインスペクタは、操作のソース、その操作のトリガ、その後の状況を表示する、コンテキスト依存の実行ビューアです。このツールを使用して、データベースイベント、Apex 处理、ワークフロー、および入力規則ロジックを含むデバッグログを検査できます。

開発者コンソールのシステムログビューでのログの操作についての詳細は、Salesforce オンラインヘルプの「ログインスペクタ」を参照してください。

開発者コンソールを使用またはデバッグログを監視している場合、ログに含まれる情報のレベルを指定できます。

#### ログカテゴリ

Apex またはワークフロールールの情報など、ログに記録する情報の種類。

#### ログレベル

ログに記録する情報量。

#### イベントの種別

カテゴリおよびレベルの組み合わせによって、記録するイベントが指定されます。イベントが開始した行番号や文字番号、イベントに関連する項目、イベントの期間(ミリ秒)など、各イベントは追加情報をログに記録できます。

### デバッグログカテゴリ

次のログカテゴリを指定できます。各カテゴリにログ記録される情報の量はログレベルによって異なります。

ログカテゴリ	説明
データベース	すべてのデータ操作言語(DML)ステートメント、オンライン SOQL または SOSL クエリなど、データベースアクティビティに関する情報を記録します。
ワークフロー	ルール名や実行されるアクションなど、ワークフロールールの情報が含まれます。
入力規則	ルール名、規則が true または false のどちらに評価したのかなど、入力規則に関する情報を記録します。
コールアウト	サーバが外部 Web サービスから送受信している要求応答 XML を記録します。これは、Force.com Web サービス API コールの使用に関する問題をデバッグするときに便利です。
コード	Apex コードに関する情報と、DML ステートメントによって生成されたログメッセージ、オンライン SOQL または SOSL クエリ、トリガの開始と完了、およびテストメソッドの開始と完了などの情報を記録します。
プロファイリング	名前空間の制限、送信されるメール数など、累積プロファイリング情報が含まれます。
	ビューステートの逐次化および並列化、Visualforce ページの数式項目の評価など、Visualforce のイベントに関する情報を記録します。

ログカテゴリ	説明
システム	メソッドなど、すべてのシステムメソッドへのコールに関する情報を記録します。

## デバッグログレベル

次のログレベルを指定できます。レベルは、低いものから順に並べてあります。特定のイベントはカテゴリおよびレベルの組み合わせに基づいてログ記録されます。多くのイベントはINFO レベルでログ記録を開始します。レベルは累積です。つまり、FINE を選択すると、ログには DEBUG、INFO、WARN および ERROR レベルでログ記録されたすべてのイベントも含まれます。



メモ: すべてのレベルをすべてのカテゴリで利用できるわけではありません。1つ以上のイベントに対応するレベルのみを利用できます。

- ERROR
- WARN
- INFO
- DEBUG
- FINE
- FINER
- FINEST



重要: リリースを実行する前に、Apex コードログレベルが FINEST に設定されていないことを確認します。Apex コードログレベルが FINEST に設定されている場合、リリースにかかる時間は予想よりも長くなる可能性があります。開発者コンソールが開いている場合、リリース中に作成されたログを含むすべてのログに開発者コンソールのログレベルが影響します。

## デバッグのイベントの種別

次は、デバッグログに記録される内容の例です。イベントは です。形式は `timestamp | イベント識別子` です。

- タイムスタンプ: イベント発生時の時刻と括弧で囲まれた値で構成されます。時刻はユーザのタイムゾーンで、形式は `HH:mm:ss.SSS` となります。値は、要求が開始されてからの経過時間をナノ秒単位で表します。経過時間の値は、開発者コンソールで確認したログには含まれません。
- イベント識別子: `や` など、書き込まれるデバッグログをトリガした特定のイベント、およびメソッド名またはコードが実行された行番号および文字番号など、そのイベントと共に記録された追加情報で構成されます。

次に、デバッグログ行の例を示します。

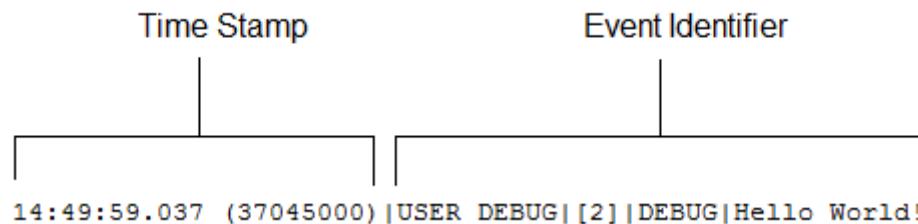


図 6: デバッグログの行の例

この例では、イベント識別子は次のもので構成されます。

- ・ イベントの名前:

[REDACTED]

- ・ コードのイベントの行番号:

[REDACTED]

- ・ メソッドが設定されたログレベル:

[REDACTED]

- ・ メソッドのユーザ入力の文字列:

[REDACTED]

ログの行の次の例は、次のコードスニペットによってトリガれます。

```
1 | @isTest
2 | private class TestHandleProductPriceChange {
3 | static testMethod void testPriceChange() {
4 | Invoice_Statement__c invoice = new Invoice_Statement__c(status__c = 'Negotiating');
5 | insert invoice;
6 | }
```

図 7: デバッグログの行のコードスニペット

次のログ行は、テストがコードの行 5 に達した場合に記録されます。

この例では、イベント識別子は次のもので構成されます。

- ・ イベントの名前:

[REDACTED]

- コードのイベントの行番号:
- [REDACTED]

- DML 操作種別:
- [REDACTED]

- オブジェクト名:
- [REDACTED]

- DML 操作に渡される行数:
- [REDACTED]

次の表には、記録されるイベントの種類、各イベントと記録される項目または情報、イベントをログ記録するログレベルおよびカテゴリの組み合わせを示しています。

イベントの名前	イベントと記録される項目または情報	記録されるカテゴリ	記録されるレベル
	割り当てられたバイト数	Apex コード	FINEST
	行番号、要求ヘッダー	コールアウト	INFO 以上
	行番号、リクエストボディ	コールアウト	INFO 以上
	なし	Apex コード	ERROR 以上
	行番号、	Apex コード	ERROR 以上
などのコードユニット名			
	行番号、Apex クラス ID、文字列 (パラメータがある場合は括弧 内にパラメータの種別)	Apex コード	DEBUG 以上
	行番号、文字列 (パラメータ がある場合はかっこ内にパラメータの種 別)	Apex コード	DEBUG 以上
	なし	Apex プロファイリング	INFO 以上
	なし	Apex プロファイリング	INFO 以上
	なし	Apex プロファイリング	FINE 以上
	なし	Apex プロファイリング	FINE 以上

イベントの名前	イベントと記録される項目または情報	記録されるカテゴリ	記録されるレベル
なし		Apex プロファイリング	FINE 以上
行番号、操作(　　、　　など)、レコード名またはタイプ、DML 操作に渡される行数		Apex コード	INFO 以上
行番号		Apex コード	INFO 以上
行番号		Apex コード	INFO 以上
パッケージ名前空間		Apex コード	INFO 以上
行番号、例外種別、メッセージ		Apex コード	INFO 以上
なし		Apex コード	ERROR 以上
なし		Apex コード	ERROR 以上
例外種別、メッセージ、スタック追跡		Apex コード	ERROR 以上
行番号、バイト数		Apex コード	FINE 以上
行番号、割り当て解除されたバイト数		Apex コード	FINE 以上
行番号		DB	FINEST
次の制限に従った、名前空間:		Apex プロファイリング	FINEST

イベントの名前	イベントと記録される項目または情報	記録されるカテゴリ	記録されるレベル
	行番号、クラスの Force.com ID、メソッドの署名	DEBUG 以上	
	行番号、クラスの Force.com ID、メソッドの署名。	DEBUG 以上	
	コンストラクタの場合、行番号、クラス名が記録されます。		
	行番号、ログ検索条件が設定されているか、範囲に含まれるクラスまたはトリガの Force.com ID、このクラスまたはトリガの名前、この範囲から出た後に有効になったログ検索条件設定	INFO 以上	
	行番号、ログ検索条件が設定されているか、範囲から除外されるクラスまたはトリガの Force.com ID、このクラスまたは	INFO 以上	

イベントの名前	イベントと記録される項目または情報	記録されるカテゴリ	記録されるレベル
	トリガの名前、この範囲に入った後に有効になったログ検索条件設定		
行番号、	反復数	DB	INFO 以上
行番号、Savepoint 名		DB	INFO 以上
行番号、Savepoint 名		DB	INFO 以上
ケース数、読み込み時間、処理時間、挿入/更新/削除するケースのマイルストンの数、新しいトリガ		ワークフロー	INFO 以上
マイルストン ID		ワークフロー	INFO 以上
なし		ワークフロー	INFO 以上
ケース ID		ワークフロー	INFO 以上
行番号、集計数、クエリソース		DB	INFO 以上
行番号、行数、期間(ミリ秒)		DB	INFO 以上
行番号、クエリソース		DB	INFO 以上
行番号、行数、期間(ミリ秒)		DB	INFO 以上
フレーム番号、次の形式の変数リスト: 变数番号   値。次に例を示します。		Apex プロファイリング	FINE 以上
行番号		Apex コード	FINER 以上
次の形式の変数リスト: 变数番号   値。次に例を示します。		Apex プロファイリング	FINE 以上
行番号、文字列 がある場合はかっこ内にパラメータの種別)	(パラメータ システム		DEBUG

イベントの名前	イベントと記録される項目または情報	記録されるカテゴリ	記録されるレベル
	行番号、文字列 がある場合はかっこ内にパラメータの種別)	(パラメータ システム	DEBUG
	行番号、メソッドの署名	システム	DEBUG
	行番号、メソッドの署名	システム	DEBUG
モード名		システム	INFO 以上
モード名		システム	INFO 以上
なし		Apex プロファイリング	INFO 以上
送信メール数		Apex プロファイリング	FINE 以上
行番号、ログレベル、ユーザ入力の文字列	Apex コード	デフォルトでは DEBUG 以上。 ユーザが メソッドのログレベルを設定すると、 イベントはこのレベルでログ記録さ れます。	
エラーメッセージ	入力規則	INFO 以上	
なし	入力規則	INFO 以上	
数式ソース、値	入力規則	INFO 以上	
なし	入力規則	INFO 以上	
ルール名	入力規則	INFO 以上	
行番号、変数名、変数の値の文字列表現、変数のアドレス	Apex コード	FINEST	
行番号、変数名、型、変数が参照可能かどうかを示す値、変数が静的かどうかを示す値	Apex コード	FINEST	
なし	Apex コード	FINEST	
要素名、メソッド名、戻り値	Apex コード	INFO 以上	
ビューステート ID	Visualforce	INFO 以上	
なし	Visualforce	INFO 以上	
ビューステート ID、数式	Visualforce	FINER 以上	

イベントの名前	イベントと記録される項目または情報	記録されるカテゴリ	記録されるレベル
なし		Visualforce	FINER 以上
メッセージテキスト		Apex コード	INFO 以上
ビューステート ID		Visualforce	INFO 以上
なし		Visualforce	INFO 以上
アクションの説明		ワークフロー	INFO 以上
ToDo件名、アクションID、ルール、所有者、期日		ワークフロー	INFO 以上
実行されたアクションの概要		ワークフロー	INFO 以上
トランザクションタイプ、プロセスノード名		ワークフロー	INFO 以上
		ワークフロー	INFO 以上
		ワークフロー	INFO 以上
所有者、割り当て先テンプレート ID		ワークフロー	INFO 以上
、ルール名、ルールID、トリガの種類(ルールがトリガの種類を重視する場合)		ワークフロー	INFO 以上
成功を示す boolean 値 (true または false)		ワークフロー	INFO 以上
アクション ID、ルール		ワークフロー	INFO 以上
メールテンプレート ID、受信者、CC メール		ワークフロー	INFO 以上
エンキューティされたアクションの概要		ワークフロー	INFO 以上
ケース ID、営業時間		ワークフロー	INFO 以上
なし		ワークフロー	INFO 以上
プロセス名、メールテンプレート ID、結果を示す Boolean 値 (true または false)		ワークフロー	INFO 以上
、オブジェクトまたは項目名		ワークフロー	INFO 以上
数式ソース、値		ワークフロー	INFO 以上
なし		ワークフロー	INFO 以上
所有者、次の所有者の種類、項目		ワークフロー	INFO 以上
なし		ワークフロー	INFO 以上
、アクション ID、ルール		ワークフロー	INFO 以上

イベントの名前	イベントと記録される項目または情報	記録されるカテゴリ	記録されるレベル
プロセス名		ワークフロー	INFO 以上
	、所有者	ワークフロー	INFO 以上
通知者名、通知者のメール、通知者テンプレート ID		ワークフロー	INFO 以上
順序を示す整数		ワークフロー	INFO 以上
ルールタイプ		ワークフロー	INFO 以上
なし		ワークフロー	INFO 以上
値		ワークフロー	INFO 以上
検索条件		ワークフロー	INFO 以上
		ワークフロー	INFO 以上
なし		ワークフロー	INFO 以上
プロセス名		ワークフロー	INFO 以上
ノードタイプ		ワークフロー	INFO 以上
	、タイム アクション、タイムアクションコンテナ、評価の日時	ワークフロー	INFO 以上
なし		ワークフロー	INFO 以上

## 関連リンク

[デバッグルогについて](#)  
[Apex API コールのデバッグ](#)

## Apex API コールのデバッグ

Apex を呼び出すすべての API コールは、  
 へのコールを含む、コードの実行に関する詳細情報  
 へのアクセスが可能なデバッグ機能をサポートしています。開発者コンソールに加えて、  
 と呼ばれる SOAP インプットヘッダーによって、次の表で概説されたレベルに応じたログ精度の設定が可能です。

要素名	型	説明
		デバッグルогに返される情報の種類を指定します。有効な値は、次のとおりです。 <ul style="list-style-type: none"> <li>•</li> <li>•</li> </ul>

要素名	型	説明
	<ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> </ul>	<p>デバッグログに返される情報の量を指定します。 のみで、ログカテゴリのレベルを使用します。</p> <p>有効なログレベルは次のとおりです(低いものから順に並べてあります)。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> </ul>

下位互換性のため、次のログレベルは の一部として引き続きサポートされます。

ログレベル	説明
	ログメッセージを含みません。
	低いレベルのメッセージと、 メソッドへのコールによって生成されたメッセージを記録します。
R 象	すべての各データ操作言語 (DML) ステートメントまたはインライン SOQL ま ‘ 腹縫

ログレベル	説明
	<p>レベルで生成されたすべてのメッセージと、次を含みます。</p> <ul style="list-style-type: none"> <li>変数宣言ステートメント</li> <li>ループ実行の開始</li> <li>break や continue など、すべてのループ制御</li> <li>例外発生 *</li> <li>静的リソースとクラスの初期化コード *</li> <li>with sharing コンテキストでの変更</li> </ul>

関連出力ヘッダー、[は、結果生成されるデバッグログを含みます。詳細は、「DebuggingHeader」\(ページ 1004\)を参照してください。](#)

## 関連リンク

- [デバッグルогеについて](#)
- [開発者コンソールのログの操作](#)

## 検出されなかった例外の処理

Apex コードにバグがある場合、またはコードレベルの例外を検出しない場合、次の手順が実行されます。

- エンドユーザに対して、問題についての簡単な説明がアプリケーションインターフェースに表示されます。このエラーメッセージには、Apex スタック追跡が含まれています。
- 項目で指定された開発者に対して、Apex スタック追跡と顧客の組織およびユーザ ID を記載した、エラーを通知するメールが送信されます。他の顧客データはレポートに記載されません。同期実行される Apex コードの場合、重複する例外エラーについてはエラーメールが抑制される場合があります。非同期実行される Apex コードの場合、つまり Apex 一括処理、スケジュールされた Apex、または future メソッド（のアノテーション付きメソッド）については、重複する例外についてのエラーメールは抑制されません。

## 実行ガバナと制限について

Apex はマルチテナント環境で実行するため、Apex ランタイムエンジンは、回避 Apex が共有リソースを独占しないようさまざまな制限事項を強制します。これらの制限、つまりガバナは、次の表で示される統計情報を追跡し、強制的に適用します。一部の Apex コードが制限を超える場合、関連付けられたガバナは、処理できない実行時例外を発行します。

ガバナ制限は、特定の名前空間または組織全体に適用されます。たとえば、salesforce.com ISV パートナーが作成する管理パッケージを Force.com AppExchange からインストールする場合、パッケージ内のコンポーネントは、

組織内の他のコンポーネントとは異なる一意の名前空間に所属します。そのため、そのパッケージ内の Apex コードは、実行中に最大 150 の DML ステートメントを発行できます。また、組織のネイティブの Apex コードも、最大 150 の DML ステートメントを発行できます。これは、管理パッケージとネイティブの組織の両方のコードが実行される場合、150 を超える DML ステートメントが 1 つの要求中に実行される可能性があることを意味しています。それに対し、salesforce.com ISV パートナーが作成していない AppExchange からパッケージをインストールする場合、そのパッケージのコードには、個別に独自のガバナ制限数はありません。使用するリソースは、組織の合計数に含まれます。累積リソースメッセージと警告メールも、管理パッケージの名前空間に基づいて生成されます。salesforce.com ISV パートナーパッケージについての詳細は、「[salesforce.com Partner Programs](#)」を参照してください。

説明	制限
発行される SOQL クエリの合計数 <sup>1</sup>	100
Apex 一括処理メソッドおよび future メソッド用に発行される SOQL クエリの合計数 <sup>1</sup>	200
SOQL クエリによって取得されるレコードの合計数	50,000
によって取得されるレコードの合計数	10,000
発行される SOSL クエリの合計数	20
1 つの SOSL クエリによって取得されるレコードの合計数	2000
発行される DML ステートメントの合計数 <sup>2</sup>	150
DML ステートメント、 、または の結果として処理されるレコードの合計数	10,000
実行されるコードステートメントの合計数	200,000
Apex 一括処理メソッドおよび future メソッド用に実行されるコードステートメントの合計数	1,000,000
ヒープの合計サイズ <sup>3</sup>	6 MB
Apex 一括処理メソッドおよび future メソッドのヒープの合計サイズ	12 MB
、 、または Apex 呼び出しのスタックの深さの合計 <sup>4</sup>	16
ループリストのバッチサイズ用	200
要求内のコールアウト (HTTP 要求または Web サービスコール) の合計数	10
要求内のすべてのコールアウト (HTTP 要求または Web サービスコール) の最大タイムアウト値	120 秒
要求内のコールアウト (HTTP 要求または Web サービスコール) のデフォルトのタイムアウト値	10 秒
Apex の各呼び出しで許可されるアノテーションを持つメソッドの合計数 <sup>5</sup>	10

説明	制限
コールアウト要求または応答 (HTTP 要求または Web サービスコール) の最大サイズ <sup>6</sup>	3 MB
許可される メソッドの合計数	10
使用できる定義の合計数 <sup>7</sup>	100
同時にスケジュールできるクラスの合計数	100
24 時間でキューできるテストクラスの合計数 <sup>8</sup>	500 または組織のテストクラス数の 10 倍の大きいほう

<sup>1</sup> 親-子リレーションのサブクエリを使用する SOQL クエリでは、各親-子リレーションは追加クエリとしてカウントされます。これらのクエリタイプには、最上位クエリ数の 3 倍に制限されています。これらのリレーションクエリの行数は、全体のコード実行の行数に加算されます。静的 SOQL ステートメントの他、次のメソッドへのコールは、要求内で発行された SOQL ステートメント数としてカウントされます。

- 
- 
- 

<sup>2</sup> 次のメソッドへのコールは、要求内で発行された DML クエリ数としてカウントされます。

- 
- 
- 
- 
- と
- と
- 
- と
- と
- と
- 

<sup>3</sup> メールサービスのヒープサイズは 36 MB です。

<sup>4</sup> 、 、 または 、ステートメントによってトリガを実行しない繰り返し Apex 処理は、1 つのスタックを使用する 1 つの呼び出し内に存在します。それに対し、トリガを実行した繰り返し Apex では、コードを実行した呼び出しとは別の新しい Apex 呼び出してトリガが発生します。Apex の新しい呼び出しの実行は、1 つの呼び出での繰り返しコールよりも手間のかかる操作であるため、これらの種類の繰り返しコールのスタックの深さには、より厳しいトリガ制限があります。

<sup>5</sup> 24 時間でのメソッドの最大呼び出し数は、250,000 または組織のライセンス数の 200 倍の大きいほうです。これは組織全体の制限で、他のすべての非同期 Apex (Apex の一括処理およびスケジュール済み Apex) と共有されます。この制限のカウント対象となるライセンスは、Salesforce フルユーザライセンスおよび Force.com アプリケーションサブスクリプションのユーザライセンスです。Chatter 限定ユーザ、Chatter カスタマーユーザ、カスタマー ポータルユーザ、およびパートナー ポータルユーザライセンスは、含まれません。

<sup>6</sup> HTTP 要求のサイズおよび応答のサイズは、ヒープサイズの合計にカウントされます。

<sup>7</sup> 定義には、次のメソッドおよびオブジェクトが含まれます。

- ChildRelationship オブジェクト
- RecordTypeInfo オブジェクト
- PicklistEntry オブジェクト
- コール
- コール

<sup>8</sup> この制限は、テストの非同期実行に適用されます。これには、開発者コンソールを含め、Salesforce ユーザインターフェースから開始するテストが含まれます。

制限は、各に対して個別に適用されます。

実行中にコードのコード実行制限を決定するには、Limits メソッドを使用します。たとえば、プログラムによってすでにコールされた DML ステートメント数を決定するには、メソッドを使用できます。または、コードに使用できる DML ステートメントの合計数を決定するには、メソッドを使用できます。

最高のパフォーマンスを得るためにには、特にトリガ内のクエリに対しては、セレクティブ SOQL クエリを使用する必要があります。実行時間が長時間に渡ることを回避するために、セレクティブ以外の SOQL クエリはシステムより終了される場合があります。100,000 件を超えるレコードを含むオブジェクトに対してトリガでセレクティブではないクエリを使用すると、エラーメッセージが表示されます。このエラーを回避するには、必ずセレクティブクエリを使用します。「[より効率的な SOQL クエリ](#)」を参照してください。

Salesforce.com API バージョン 20.0 以前を使用して保存された Apex の場合、API コールによってトリガが起動されると、200 レコードずつに分割されていたチャンクが、100 レコードずつのチャンクにさらに分割されます。Salesforce.com API バージョン 21.0 以降を使用して保存された Apex の場合、API のチャンクがそれ以上分割されることはありません。静的変数の値は、API バッチ間でリセットされますが、ガバナ制限はリセットされません。API バッチ間の状態情報の追跡に静的変数を使用しないでください。

実行ガバナ制限だけでなく、Apex には次の制限もあります。

- クラスの最大文字数は、1,000,000 です。
- トリガの最大文字数は、1,000,000 です。
- 組織内のすべての Apex コードで使用されるコードの最大量は、3 MB です。



メモ: この制限は、AppExchange からインストールされた認証管理パッケージ (AppExchange Certified) とマークされたアプリケーション) には適用されません。これらのパッケージタイプのコードは、組織のコードとは異なる独自の名前空間に属しています。AppExchange Certified パッケージについての詳細は、Force.com AppExchange オンラインヘルプを参照してください。

この制限は、[アノテーション](#)で定義されたクラスに含まれるコードにも適用されません。

- メソッドのサイズには制限があります。制限を超える大規模なメソッドはコードの実行中に例外が発生する場合があります。Java の場合と同様に、Apex でのメソッドサイズの制限はコンパイル形式で 65,535 のバイトコード命令です。
- SOQL クエリの実行が 120 秒を超える場合、Salesforce で要求をキャンセルできます。
- 各 Apex 要求は、10 分間の実行に制限されます。
- 同じホストの URL へのコールアウト要求は、最大 20 の同時要求に制限されます。ホストは URL の一意のサブドメインで定義されます。たとえば、[www.salesforce.com](#) と [www.force.com](#) は 2 つの異なるホストです。この制限は、同じホストにアクセスするすべての組織で計算されます。この制限を超えると、が発生します。
- システム管理者以外のユーザの場合、行動レポートが返すレコードの最大数は 20,000 件です。システム管理者の場合、100,000 件です。
- 各組織では、5 秒以上かかる長時間の要求に対して 10 個の同期同時要求が許可されています。10 個の長時間の要求が実行されている間に追加要求を作成すると、要求は拒否されます。
- ユーザは一度に最大 50 個のクエリカーソルを開くことができます。たとえば、50 個のカーソルが開いていて、同じユーザとしてログインしたままのクライアントアプリケーションで新しいカーソルを開こうとすると、50 個のカーソルのうち最も古いカーソルが解放されます。この制限は、一度に 1 ユーザあたり最大 5 個のクエリカーソルを開くことができる、Apex 一括処理メソッドでは異なります。それ以外の Apex 一括処理メソッドには、高い方のカーソル数制限である 50 が適用されます。

異なる Force.com 機能のカーソル制限は個別に追跡されます。たとえば、50 個の Apex クエリカーソル、50 個の一括処理カーソル、および 50 個の Visualforce カーソルを同時に開くことができます。

- 1 つのトランザクションでは、10 個の一意の名前空間のみを参照できます。たとえば、オブジェクトを更新するときに、管理パッケージでクラスを実行するオブジェクトがあるとします。その後、クラスは 2 番目のオブジェクトを更新します。つまり、他のパッケージの他のクラスを実行します。最初に 2 番目のパッケージに直接アクセスしない場合でも、同じトランザクション内で発生するため、1 つのトランザクションでアクセスする名前空間の数に含まれます。
- すべての Apex リリースのクラスとトリガの最大コードユニット数は、5,000 個です。
- Data.comクリーンアップ製品とその自動ジョブを使用していて、取引先、取引先責任者、またはリードレコードで実行する SOQL クエリの Apex トリガを設定している場合、それらのオブジェクトでクエリがクリーンアップジョブに干渉する可能性があります。Apex トリガ(合計)は、バッチあたり 200 個以下の SOQL クエリにしてください。この個数を超えた場合、そのオブジェクトのクリーンアップジョブは失敗します。また、トリガが メソッドをコールする場合は、バッチあたり 10 個の コールに制限されます。

## Apex の一括処理のガバナ制限

Apex の一括処理について、次のガバナ制限に注意してください。

- Apex では最大 5 件のキュー内または有効な一括処理ジョブを実行できます。
- 24 時間での Apex 一括処理メソッドの最大実行数は、250,000 または組織のライセンス数の 200 倍の大きいほうです。メソッドの実行数には、[Apex 一括処理](#)、[Apex トリガ](#)、および [Apex イベントハンドラー](#) メソッドの実行が含まれます。これは組織全体の制限で、他のすべての非同期 Apex(スケジュール済み Apex および future メソッド)と共有されます。この制限のカウント対象となるライセンスは、Salesforce フルユーザライセンスまたは Force.com アプリケーションサブスクリプションのユーザライセンスです。Chatter 限定ユーザ、Chatter カスタマーユーザ、カスタマー・ポータルユーザ、およびパートナー・ポータルユーザライセンスは、含まれません。

- ユーザは一度に最大 50 個のクエリカーソルを開くことができます。たとえば、50 個のカーソルが開いていて、同じユーザとしてログインしたままのクライアントアプリケーションで新しいカーソルを開こうすると、50 個のカーソルのうち最も古いカーソルが解放されます。なお、この制限は、Apex の一括処理メソッドである `execute` では異なっており、1 ユーザが同時に開けるクエリカーソルは最大 5 個です。それ以外の Apex 一括処理メソッドには、高い方のカーソル数制限である 50 が適用されます。
- 異なる Force.com 機能のカーソル制限は個別に追跡されます。たとえば、50 個の Apex クエリカーソル、50 個の一括処理カーソル、および 50 個の Visualforce カーソルを同時に開くことができます。
- オブジェクトでは最大 5,000 万件のレコードが返されます。5,000 万件以上のレコードが返された場合、一括処理ジョブは即座に終了し「失敗」とマークされます。
- メソッドが `QueryLocator` を返す場合、`size` の省略可能な範囲パラメータには最大 2,000 を指定できます。これより大きい値に設定すると、Salesforce では、`QueryLocator` が返すレコードを、最大 2,000 レコードまでの、より小さいバッチに分割します。メソッドが `Iterable` オブジェクトを返す場合、`scope` パラメータ値に上限はありませんが、非常に大きい値を使用すると、他の制限が発生する場合があります。
- メソッドによって返されるレコードを 200 個ずつのバッチに分割し、各バッチをメソッドに渡します。Apex ガバナ制限は、各実行でリセットされます。
- `execute`、`future`、および `batch` メソッドは、それぞれ最大 10 回のコールアウトを実装できます。
- 一括処理の実行は、メソッド実行 1 回あたり 10 コールアウトに制限されます。
- Apex の一括処理ジョブの `start` メソッドは、組織内で一度に 1 つのみ実行できます。キュー内のまだ開始されていない一括処理ジョブは、開始されるまで保持されます。なお、この制限により一括処理ジョブが失敗することはありません。また、複数のジョブが実行されている場合は、Apex の一括処理ジョブの `join` メソッドが並行して実行されます。

## Apex スケジューラの制限

- 一度にスケジュールできる Apex ジョブの数は 100 です。Salesforce の [スケジュール済みジョブ] ページを表示し、[スケジュール済み Apex] と同じデータ型の検索条件でカスタムビューを作成することで、現在の個数を確認できます。また、プログラムで CronTrigger オブジェクトをクエリして、スケジュール済みジョブの数を取得することができますが、この場合は Apex のスケジュール済みジョブだけでなく、全タイプのスケジュール済みジョブが含まれます。
- 24 時間でのスケジュール済み Apex の最大実行数は、250,000 または組織のライセンス数の 200 倍の大きいほうです。これは組織全体の制限で、他のすべての非同期 Apex (Apex 一括処理メソッドおよび `future` メソッド) と共有されます。この制限のカウント対象となるライセンスは、Salesforce フルユーザライセンスまたは Force.com アプリケーションサブスクリプションのユーザライセンスです。Chatter 限定ユーザ、Chatter カスタマーアカウント、カスタマーポータルユーザ、およびパートナーポータルユーザライセンスは、含まれません。

## メール制限

### 受信メール制限

メールサービス: 処理メールメッセージの最大数  
(オンデマンドメール-to-ケースの制限を含む)

ユーザライセンス数 × ユーザライセンス数 × 1,000、1 日あたりの最大数 1,000,000

メールサービス: メールメッセージの最大サイズ (本文および添付ファイル)	10 MB <sup>1</sup>
オンデマンドメール-to-ケース: メールの添付ファイルの最大サイズ	10 MB
オンデマンドメール-to-ケース: 処理するメールメッセージの最大数 (メールサービスの制限に対してカウントする)	ユーザライセンス数 × 1,000、1 日あたりの最大数 1,000,000

<sup>1</sup> メールサービスのメールメッセージの最大数は、言語および文字セットによって異なります。

メールサービスを定義するときには、次の点に注意してください。

- メールサービスは、そのアドレスの 1 つが受信したメッセージを処理するだけです。
- Salesforce は、[オンデマンドメール-to-ケース] など、すべてのメールサービスを合計した 1 日に処理できるメッセージの総数を制限します。この制限を超えたメッセージは、各メールサービスの失敗時の応答設定に基づいて、戻す、破棄する、あるいは翌日処理するためのキューに入れます。Salesforce は、ユーザライセンス数 × 1,000 で制限値を算出します。1 日の最大は 1,000,000 件です。たとえば、ライセンス数が 10 の場合、1 日最大 10,000 件のメールメッセージを処理できます。
- sandbox 内に作成したメールサービスアドレスは、本番組織にコピーできません。
- メールサービスごとに Salesforce に通知して、送信者のメールアドレスではなく、特定のアドレスにエラーメールメッセージを送信できます。
- メール (結合された本文テキスト、本文 HTML および添付ファイル) が約 10 MB を超える場合 (言語や文字セットに応じて異なる)、メールサービスはメールメッセージを拒否し、送信者に通知します。

#### 送信メール: Apex を使用して送信する單一メールおよび一括メールの制限

API または Apex を使用して、グリニッジ標準時間 (GMT) に基づいて、1 日に最大 1,000 個の外部メールアドレスに単一メール送信できます。Salesforce アプリケーションを使用して送信する單一メールはこの制限にカウントされません。取引先、取引先責任者、リード、商談、ケース、キャンペーン、カスタムオブジェクトの各ページから、組織の取引先責任者、リード、個人取引先、ユーザに個別のメールを送信する場合は、制限はありません。

單一メールを送信する場合は、次の点に注意してください。

- ごとに 100 個までのメールを送信できます。
- を使用して組織の内部ユーザにメールを送信するときに  
ユーザ ID を指定すると、メールが 1 日あたりの制限値にカウントされません。ただし、  
で内部ユーザのメールアドレスを指定すると、制限値にカウントされます。

グリニッジ標準時間 (GMT) に基づいて、1 組織あたり 1 日に合計 1,000 個の外部メールアドレスに一括メール送信できます。各一括メール送信に含むことのできる外部メールアドレスの最大数は、次のようにエディションに応じて異なります。

エディション	一括メール送信あたりの外部アドレス制限
Personal Edition、Contact Manager Edition、および Group Edition	一括メール送信は使用できません
Professional Edition	250

エディション	一括メール送信あたりの外部アドレス制限
Enterprise Edition	500
Unlimited Edition	1,000



メモ: 次のメール制限に注意してください。

- 単一メールおよび一括メールの制限では、アドレスが一意であるかどうかは考慮されません。たとえば、メールに  
が10回含まれている場合、制限に対して10とカウントされます。
- ポータルユーザを含め、組織の内部ユーザに送信できるメールには制限はありません。
- トライアル期間中の Developer Edition 組織と Salesforce を評価する組織では、1日あたり10個を超える外部メールアドレスに一括メール送信できません。この低い制限は、組織が Winter '12 リリースより前に作成されており、一括メール送信が高い制限すでに有効になっている場合は適用されません。

## 関連リンク

[Apex の制限事項](#)

[Future アノテーション](#)

## ガバナ制限のメール警告の使用

エンドユーザがガバナ制限の 50% を超える Apex コードを呼び出す場合、追加詳細がある行動についてのメール通知を受け取る組織ユーザを指定できます。メール警告を有効にする手順は、次のとおりです。

- 管理者ユーザとして Salesforce にログインします。
- [設定] で、[ユーザの管理] > [ユーザ] をクリックします。
- メール通知を受け取るユーザ名の横にある [編集] をクリックします。
- 警告メールの送信 オプションを選択します。
- [保存] をクリックします。

## 第 10 章

### 管理パッケージでの Apex の開発

---

トピック:

- ・ [パッケージバージョン](#)
- ・ [Apex の廃止](#)
- ・ [パッケージバージョンの動作](#)

パッケージとは、個々のコンポーネントなどの小さいものや関連アプリケーションのセットなどの大きいものを格納するコンテナです。パッケージの作成後、他の Salesforce ユーザおよび組織(社外のユーザ、組織も含む)にそのパッケージを配布できます。組織は、他の多くの組織でダウンロードおよびインストールできる単一の管理パッケージを作成できます。管理パッケージは、未管理パッケージとは異なり、コンポーネントの一部がロックされていて、後でアップグレードできます。未管理パッケージには、ロックされたコンポーネントは含まれておらず、アップグレードはできません。

ここでは、管理パッケージの Apex の開発に関連する次のトピックについて説明します。

- ・ [パッケージバージョン](#)
- ・ [Apex の廃止](#)
- ・ [パッケージバージョンの動作](#)

## パッケージバージョン

パッケージバージョンは、パッケージでアップロードされる一連のコンポーネントを特定する番号です。バージョン番号の形式は `majorNumber.minorNumber.patchNumber` (例: 2.1.3) です。メジャー番号とマイナー番号は、メジャーリリース時に指定した値に増えます。`patchNumber` は、パッチリリースにのみ生成および更新されます。

未管理パッケージはアップグレードできないため、各パッケージバージョンは単に配布用コンポーネントのセットです。パッケージバージョンは管理パッケージでより大きな意味を持ちます。パッケージは異なるバージョンで異なる動作をします。公開者は、パッケージバージョンを使用して、パッケージを使用する既存のインテグレーションに影響を与えることなく後続のパッケージバージョンをリリースすることにより、管理パッケージのコンポーネントを強化することができます。

既存の登録ユーザが新しいパッケージをインストールした場合、パッケージ内の各コンポーネントのインスタンスは1つだけですが、コンポーネントは古いバージョンをエミュレートできます。たとえば、登録ユーザが Apex クラスを含む管理パッケージを使用すると想定します。公開者が Apex クラスのメソッドを廃止し、新しいパッケージバージョンをリリースする場合でも、新しいバージョンをインストールした後、登録者は Apex クラスのインスタンスを1つのみ使用できます。ただし、この Apex クラスは、古いバージョンの廃止されたメソッドを参照するコードの以前のバージョンをエミュレートできます。

管理パッケージで Apex を開発する場合、次の点に注意が必要です。

- 管理パッケージの一部である Apex クラスまたはトリガに含まれるコードは、自動的に隠され、インストール先の組織では見ることができません。唯一の例外には、グローバルとして宣言されているメソッドがあります。それらのメソッド署名はインストールを行う組織でも参照できます。
- 管理パッケージは、一意の名前空間を受け取ります。この名前空間は、インストール先の組織で名前の重複を防ぐために、クラス名、メソッド、変数などの先頭に自動的に追加されます。
- 1つのトランザクションでは、10 個の一意の名前空間のみを参照できます。たとえば、オブジェクトを更新するときに、管理パッケージでクラスを実行するオブジェクトがあるとします。その後、クラスは2番目のオブジェクトを更新します。つまり、他のパッケージの他のクラスを実行します。最初に2番目のパッケージに直接アクセスしない場合でも、同じトランザクション内で発生するため、1つのトランザクションでアクセスする名前空間の数に含まれます。
- パッケージ開発者は、**アノテーション**を使用して、今後のリリースの管理パッケージでは参照できないメソッド、クラス、例外、列挙、インターフェース、変数を指定します。要件の変化にともなって、管理パッケージのコードをリファクタリングする場合に役立ちます。
- システムメソッド **use** を使用して、パッケージバージョンコンテキストを異なるパッケージバージョンに変更するテストメソッドを記述できます。
- インターフェースまたはクラスが「管理-リリース済み」パッケージバージョンでアップロードされた後は、global インターフェースにメソッドを追加することも、抽象メソッドをクラスに追加することもできません。「管理-リリース済み」パッケージのクラスが仮想の場合、そこに追加できるメソッドも仮想であり、実装が必要です。
- 明示的に名前空間を参照する未管理パッケージに含まれる Apex コードは、アップロードできません。

## Apex の廃止

パッケージ開発者は、アノテーションを使用して、今後のリリースの管理パッケージでは参照できないメソッド、クラス、例外、列挙、インターフェース、変数を指定します。要件の変化にともなって、管理パッケージのコードをリファクタリングする場合に役立ちます。別のパッケージバージョンを「管理-リリース済み」としてアップロードすると、最新のパッケージバージョンをインストールする新しい登録者に非推奨の要素が表示されることはありませんが、その要素は既存の登録者およびAPIインテグレーションでは機能し続けます。パッケージ開発者は、メソッドまたはクラスなどの廃止された項目を、内部で引き続き参照できます。



メモ: 未管理パッケージの Apex クラスまたはトリガの

アノテーションは使用できません。

パッケージ開発者は、異なる Salesforce 組織のユーザのパイロット版による評価およびフィードバックに、「管理-ベータ」パッケージバージョンを使用できます。開発者が Apex 識別子を廃止し、パッケージのバージョンを「管理-ベータ」としてアップロードしても、パッケージバージョンをインストールした登録者はパッケージバージョンの廃止された識別子を参照できます。パッケージ開発者がその後「管理-リリース済み」パッケージバージョンをアップロードした場合、インストールした登録者にはパッケージバージョンの廃止された識別子は表示されません。

## パッケージバージョンの動作

パッケージコンポーネントは、異なるパッケージバージョンで異なる動作をします。動作のバージョン設定を使用して、新しいコンポーネントをパッケージに追加し、既存のコンポーネントを調整できます。コードは既存の登録者にもシームレスに機能します。パッケージ開発者が新しいコンポーネントをパッケージに追加し、新しいパッケージバージョンをアップロードした場合、新しいパッケージバージョンをインストールした登録者は新しいコンポーネントを使用できます。

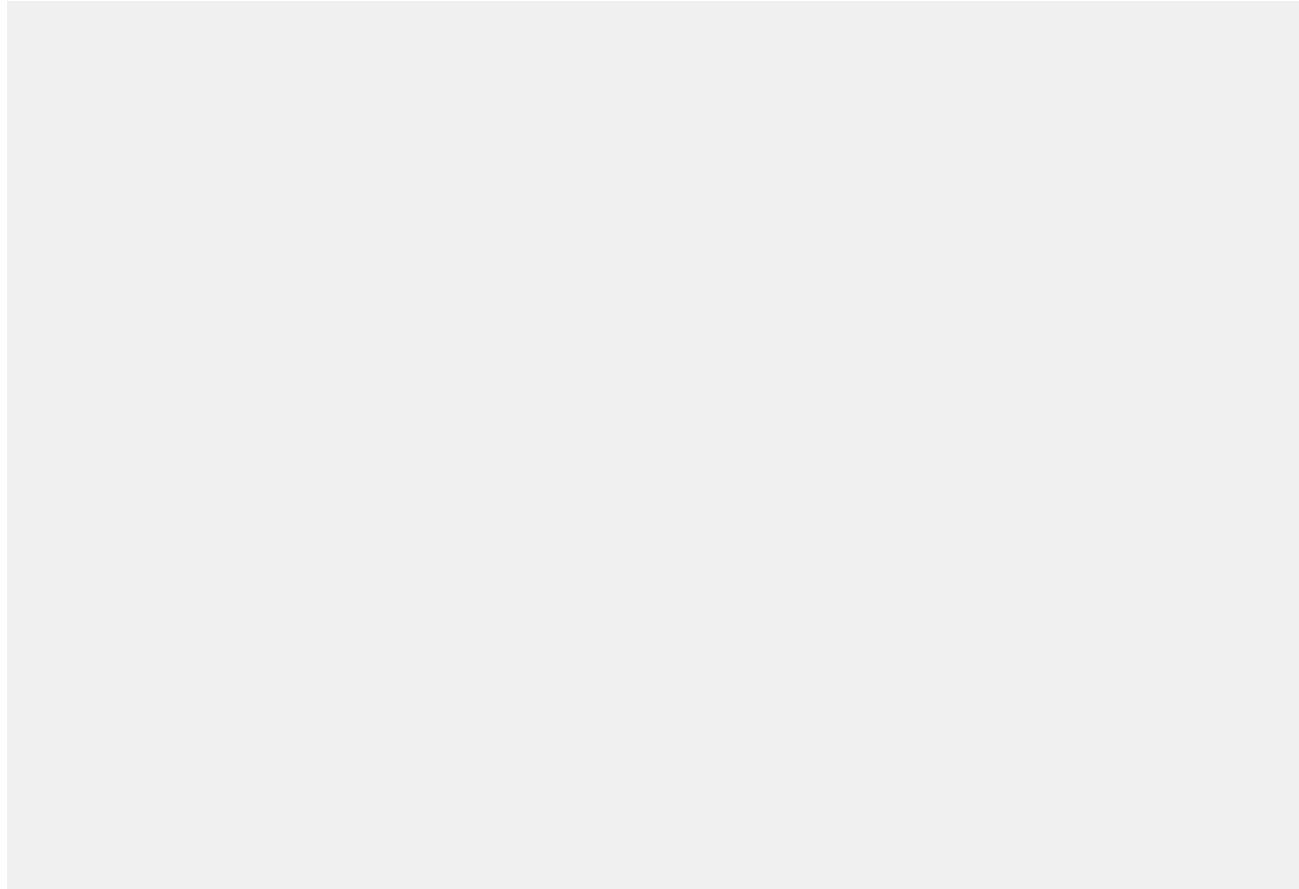
### Apex コードの動作のバージョン設定

パッケージ開発者は条件付きロジックを Apex クラスとトリガで使用し、異なるバージョンに異なる動作をさせることができます。こうすることで、パッケージ開発者は、コード開発を続けながら以前のパッケージバージョンのクラスとトリガでの既存の動作をサポートし続けることができます。

登録者が、複数のバージョンのパッケージをインストールし、パッケージ内の Apex クラスまたはトリガを参照するコードを記述する場合、参照しているバージョンを選択する必要があります。パッケージ内で参照している Apex コード内で、参照を作成する Apex コードのコールのバージョン設定に基づき、異なるコードパスを条件付で実行できます。コール元のコードのパッケージバージョン設定は、パッケージコード内で

メソッドをコールすることによって判断できます。こうすることで、パッケージ開発者は、要求コンテキストを決定し、さまざまなバージョンのパッケージに異なる動作を指定することができます。

次のサンプルでは、  
メソッドを使用して  
し、異なるバージョンのパッケージに対して、Apex トリガにさまざまな動作を定義します。  
クラスをインスタンス化



パッケージバージョンを処理するメソッドの完全な一覧は、[System メソッドの「Version メソッド」](#)および  
「[メソッド](#)」を参照してください。

インストール済みパッケージ内のあるクラスによってパッケージの別のクラスのメソッドが呼び出される場合、  
要求コンテキストは維持されます。たとえば、登録者が CountryUtil クラスおよび ContinentUtil Apex クラスを含む  
GeoReports パッケージをインストールしたとします。登録者は GeoReportsEx クラスを新規作成し、バージョン設定を使用して、GeoReports パッケージのバージョン 2.3 にバインドします。GeoReportsEx が、CountryUtil のメソッドを内部的に呼び出す ContinentUtil のメソッドを呼び出すと、要求コンテキストは ContinentUtil から CountryUtil に反映され、CountryUtil 内の  
メソッドは、GeoReports パッケージのバージョン 2.3 を返します。

## バージョン設定されていない Apex コードの項目

複数のパッケージバージョンに渡るいくつかの Apex 項目の動作を変更できます。たとえば、新しい登録者が後続のバージョンのパッケージを参照できないように、メソッドを廃止できます。

ただし、次のリストの修飾子、キーワード、アノテーションはバージョン設定できません。パッケージ開発者が次の修飾子、キーワード、またはアノテーションのいずれかに変更を加えると、変更はすべてのパッケージバージョンに反映されます。

一部の項目が管理パッケージの Apex コードで使用される場合、項目に行うことができる変更には制限があります。

パッケージ開発者は、次の項目を追加または削除できます。

- 
- 
- 
- 
- 

パッケージ開発者は、次の項目を制限付きで変更できます。

- : に変更できます。
- : に変更できます。
- : に変更できます。
- : に変更できますが削除はできません。
- : 削除できますが追加はできません。

パッケージ開発者は、次の項目の追加または変更ができません。

- 
- 

パッケージ開発者は キーワードを追加できますが、いったん追加すると削除することはできません。



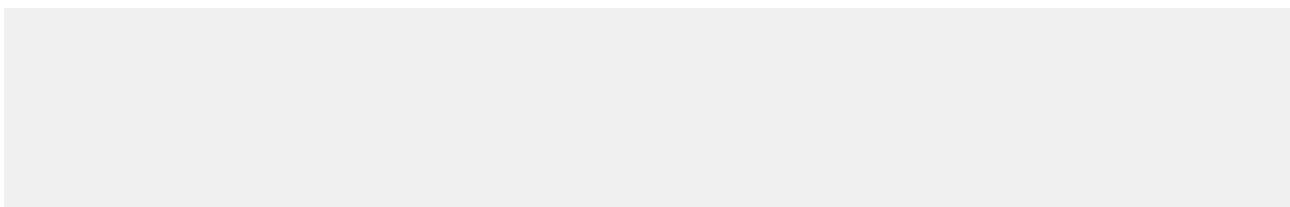
メモ: 管理パッケージコードの

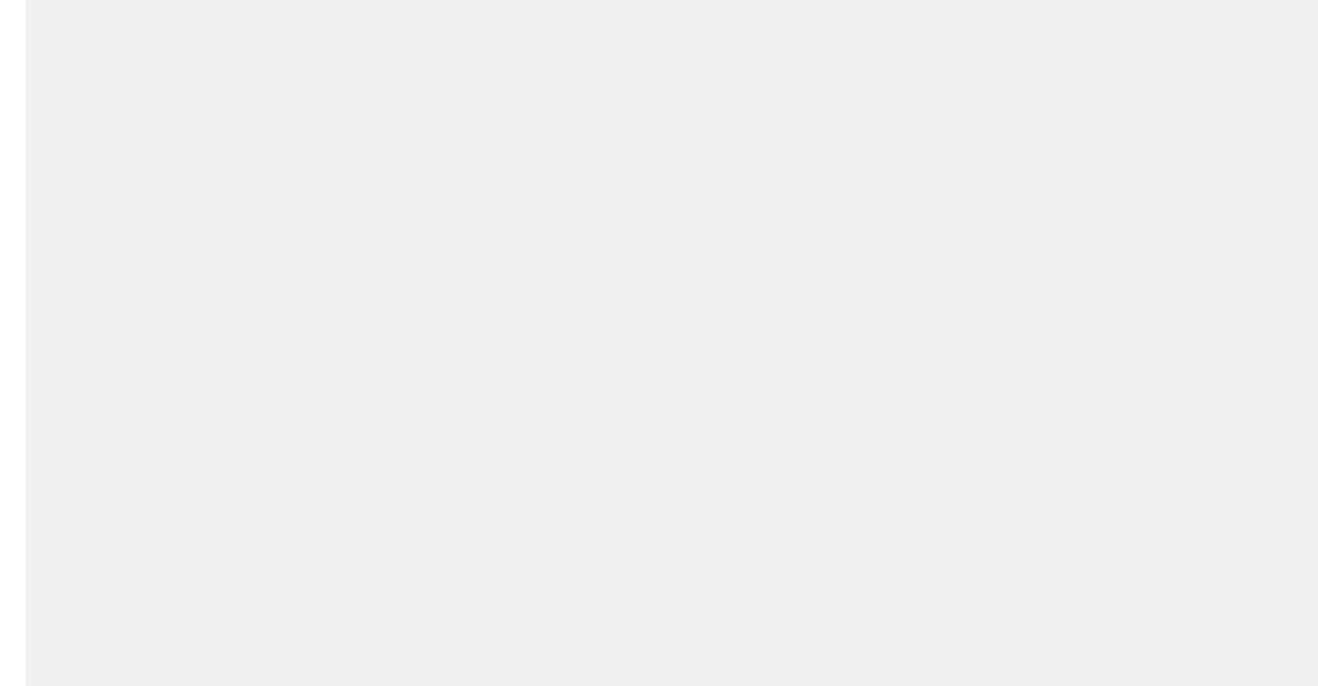
メソッドまたは変数を廃止することはできません。

## パッケージバージョンの動作のテスト

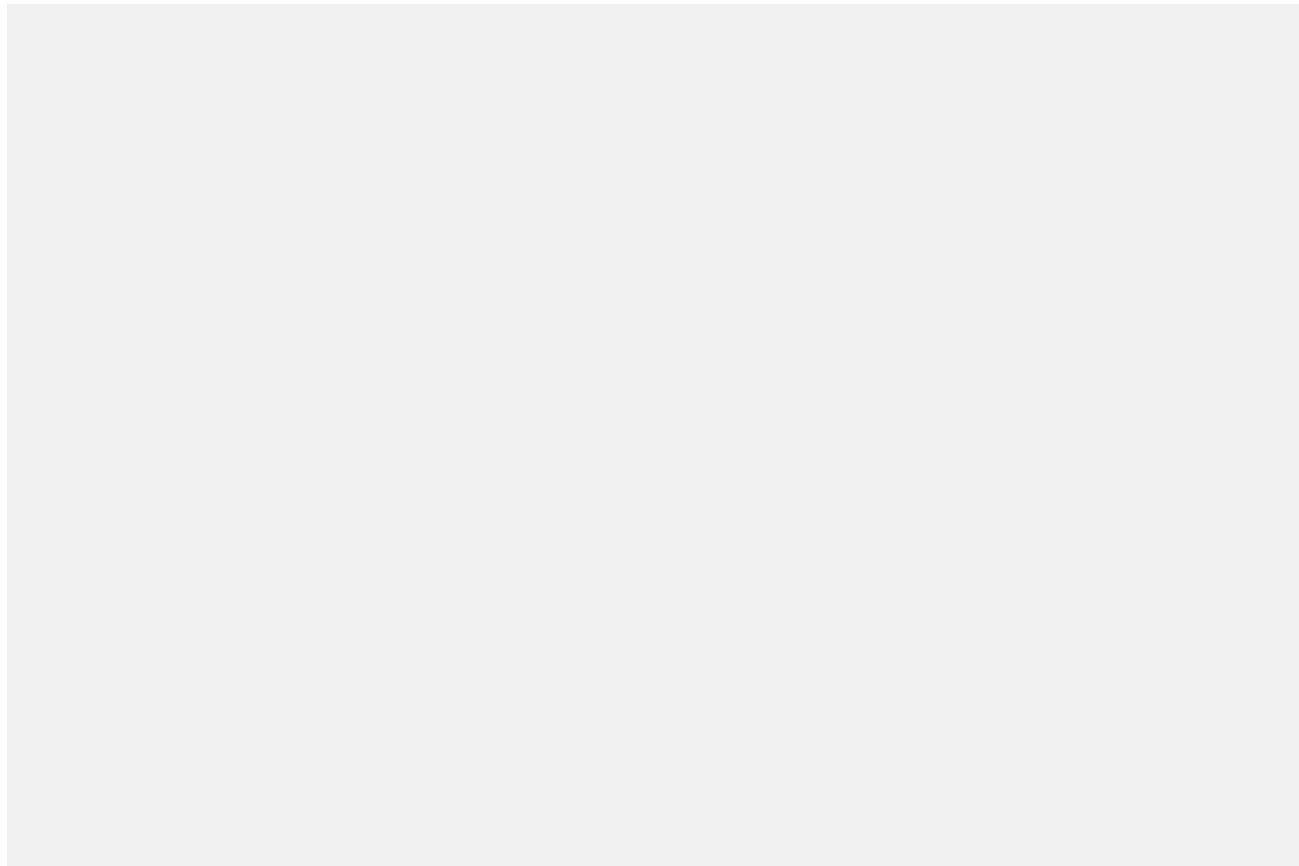
異なるパッケージバージョンの Apex クラスまたはトリガの動作を変更する場合、異なるパッケージバージョンでコードが期待通り実行されるようにテストすることが重要です。システムメソッド を使用して、パッケージバージョンコンテキストを異なるパッケージバージョンに変更するテストメソッドを記述できます。テストメソッドでは のみ使用できます。

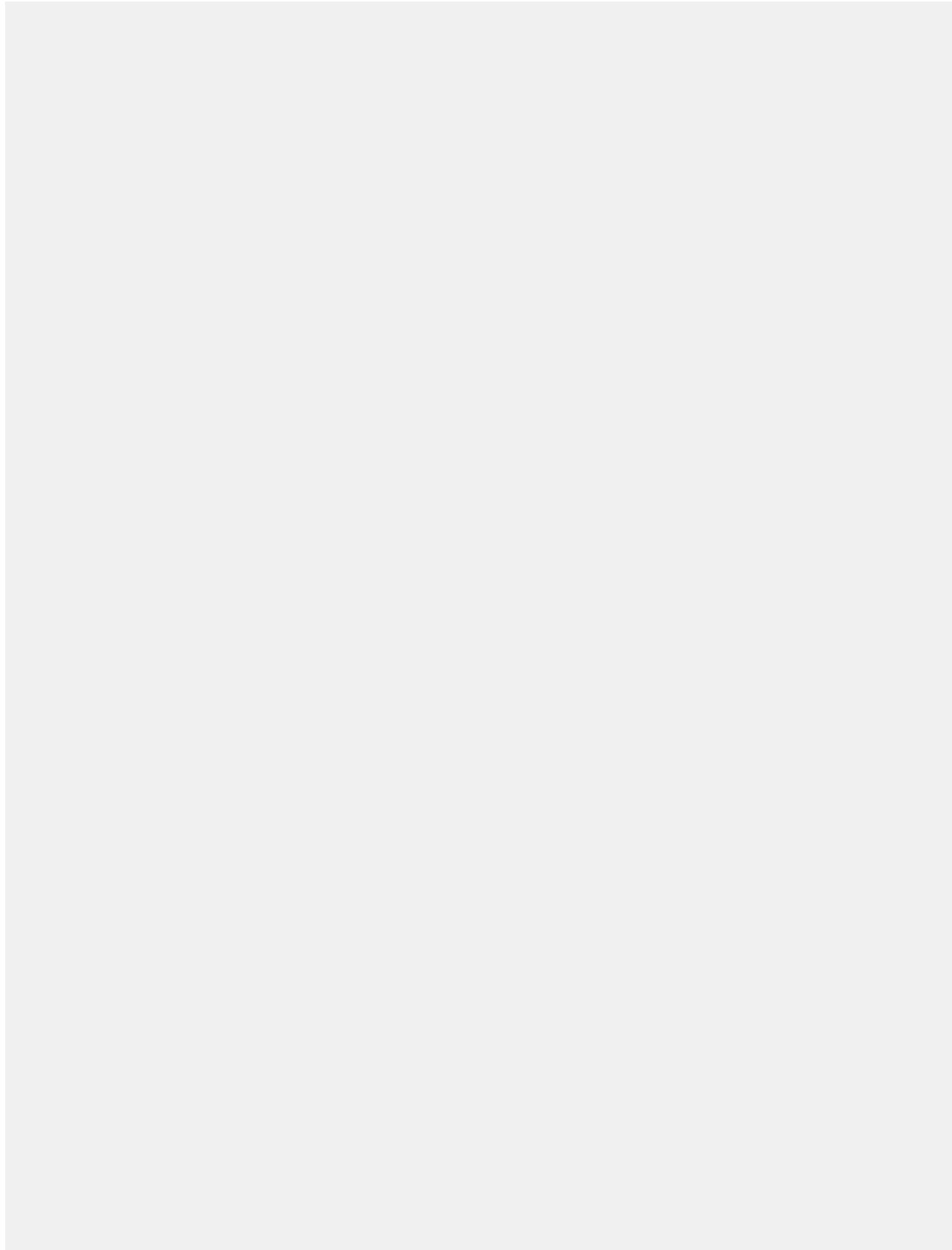
次の例は、異なるパッケージバージョンのトリガのさまざまな動作を示しています。

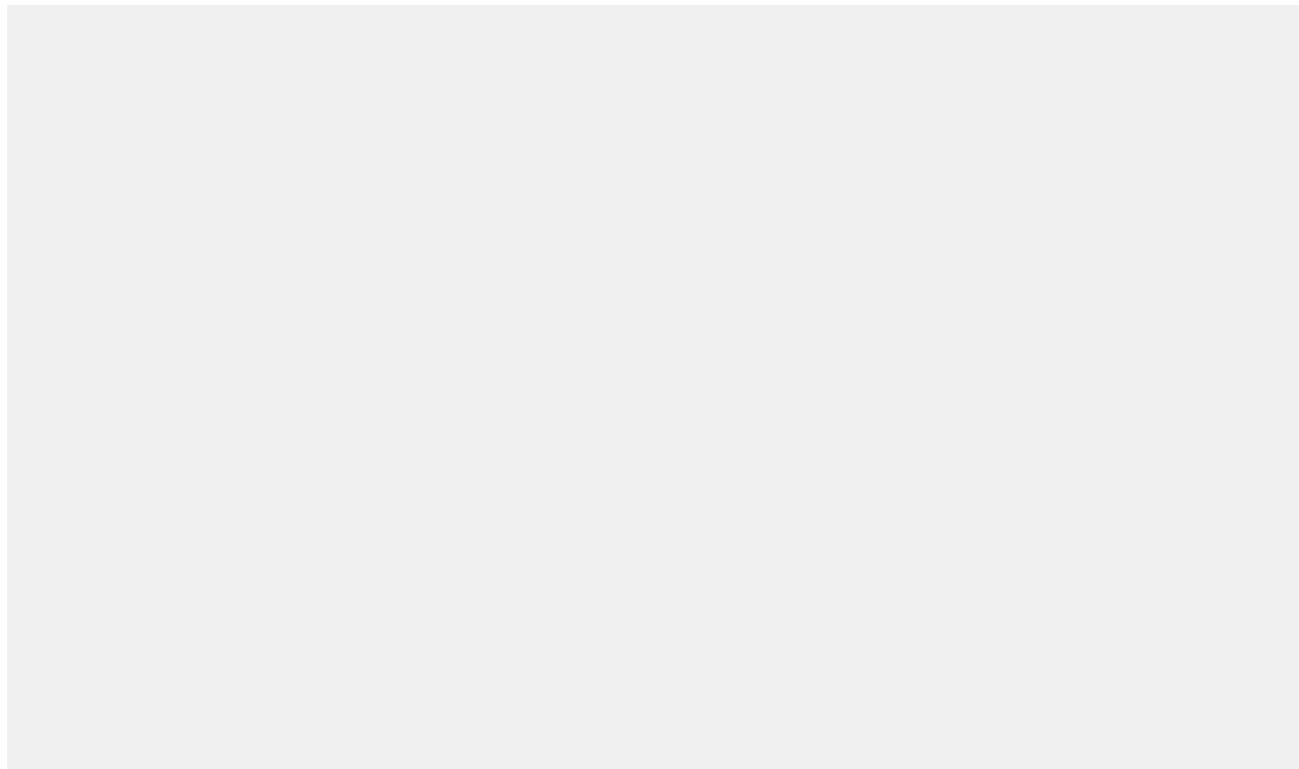




次のテストクラスでは、メソッドを使用して、特定のバージョンの有無におけるトリガの動作を検証します。







# 第 11 章

## Apex メソッドを SOAP Web サービスとして公開

トピック:

- [WebService メソッド](#)

外部アプリケーションがコードおよびアプリケーションにアクセスできるように、Apex メソッドを SOAP Web サービスとして公開できます。Apex メソッドを公開するには、[WebService メソッド](#)を使用します。

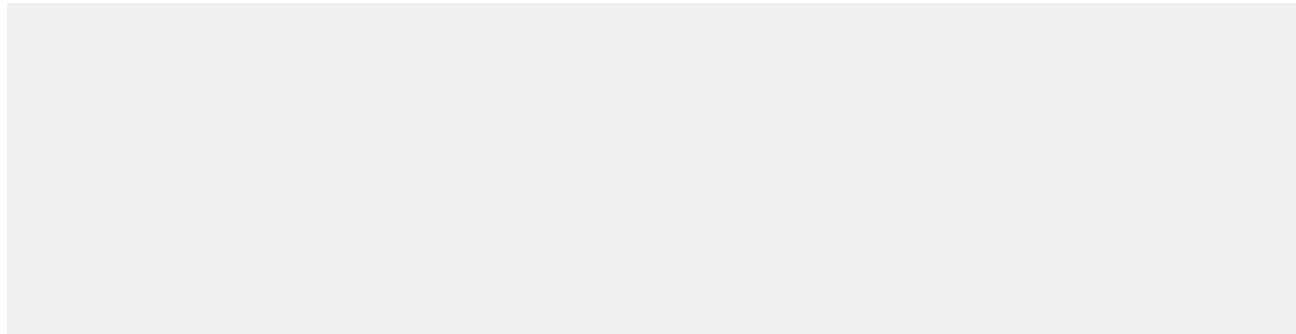


### ヒント:

- Apex SOAP Web サービスを使用すると、外部アプリケーションは SOAP Web サービスを使用して Apex メソッドを呼び出すことができます。[Apex コールアウト](#)を使用すると、Apex は外部の Web サービスまたは HTTP サービスを呼び出すことができます。
- Apex REST API は、Apex クラスおよびメソッドを REST Web サービスとして公開します。[「Apex クラスを REST Web サービスとして公開」](#)を参照してください。

## WebService メソッド

Apex クラスマソッドは、カスタムの SOAP Web サービスコールとして公開できます。これにより、外部アプリケーションが Apex Web サービスを呼び出して、Salesforce のアクションを実行できます。これらのメソッドの定義には **キーワード**を使用します。次に例を示します。



外部アプリケーションの開発者は、クラスの WSDL を生成して、**メソッド**を含む Apex クラスに統合できます。Apex クラス詳細ページから WSDL を生成する手順は、次のとおりです。

1. アプリケーションで、[設定] から [開発] > [Apex クラス] をクリックします。
2. **メソッド**を含むクラス名をクリックします。
3. [WSDL の作成] をクリックします。

## WebService メソッドによるデータの公開

カスタム **メソッド**の呼び出しには必ずシステムコンテキストを使用します。その結果、現在のユーザの証明書は使用されず、これらのメソッドにアクセスできるすべてのユーザが、権限、項目レベルのセキュリティ、共有ルールに関係なく、全機能を使用できます。そのため、**キーワード**でメソッドを公開する開発者は、ユーザが機密情報データを不注意に公開しないよう注意する必要があります。



警告: **キーワード**を使用する API によって公開されている Apex クラスマソッドは、オブジェクト権限と項目レベルのセキュリティがデフォルトで適用されていません。WebService メソッドがアクセスしようとしているオブジェクトと項目に対する現在のユーザのアクセスレベルをチェックするには、適切な Object Describe Result メソッドまたは Field Describe Result メソッドを使用することをお勧めします。『[Schema.DescribeSObjectResult](#)』および『[Schema.DescribeFieldResult](#)』を参照してください。

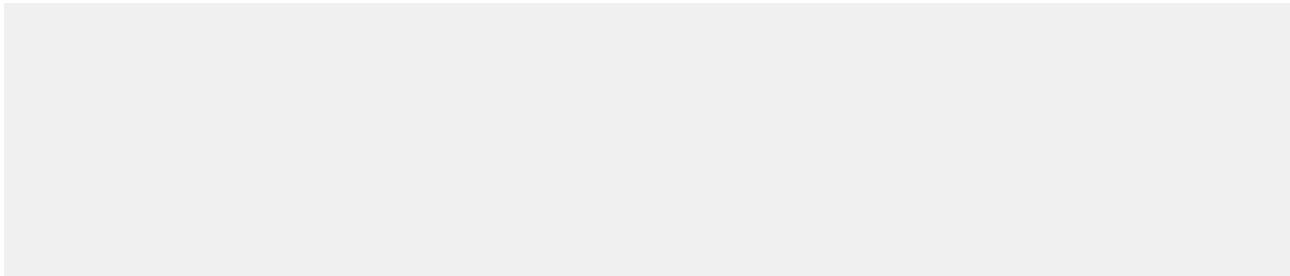
また、共有ルール（レコードレベルアクセス）は、**キーワード**でクラスを宣言するときのみに適用されます。この要件は、WebService メソッドを含むクラスを含むすべての Apex クラスに適用されます。WebService メソッドの共有ルールを強制するには、これらのメソッドを含むクラスを **キーワード**で宣言します。『[または](#)』または『[キーワードの使用](#)』を参照してください。

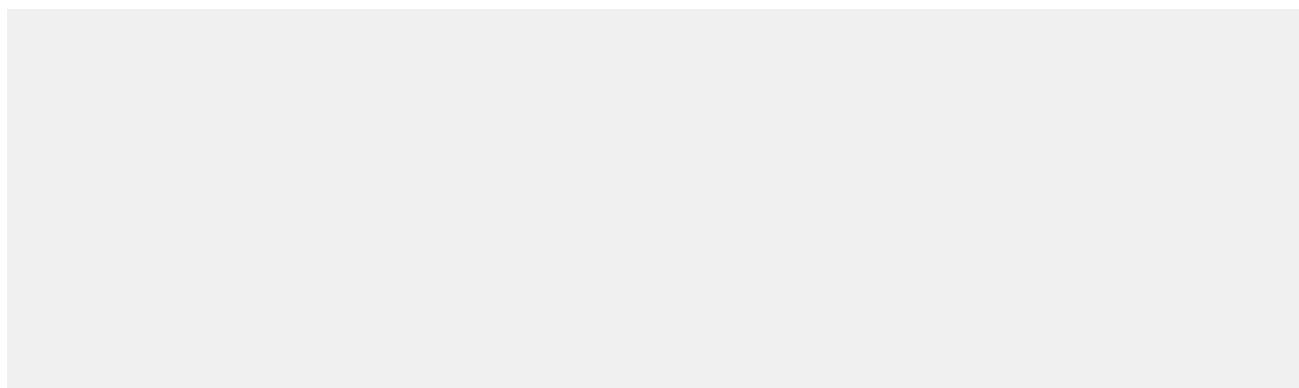
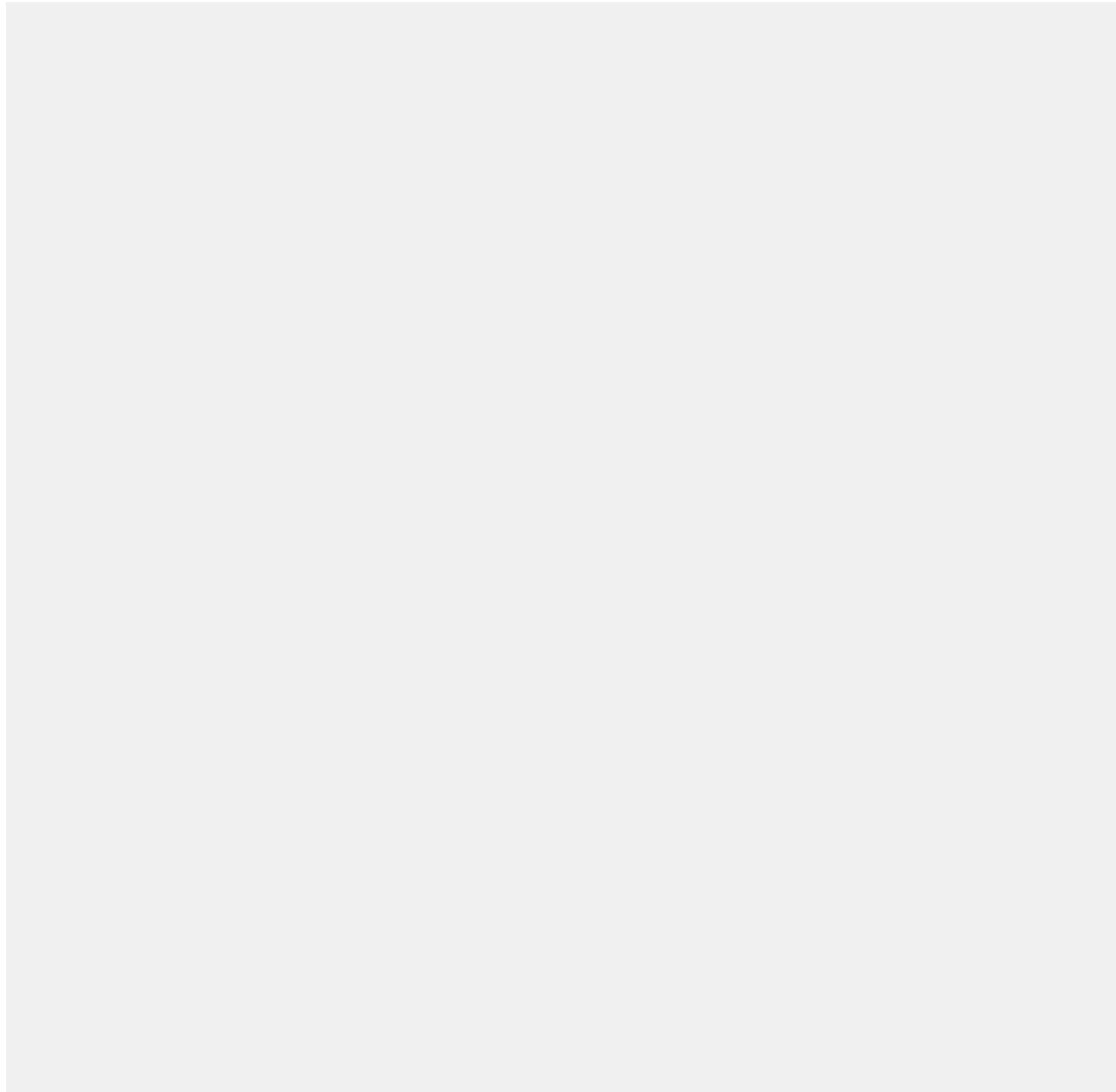
## WebService キーワードの使用に関する考慮事項

キーワードを使用する場合、次の考慮事項に留意してください。

- クラスの定義には キーワードを使用できません。ただし、最上位の外部クラスメソッドと内部クラスメソッドの定義に使用できます。
- キーワードを使用して、インターフェースや、インターフェースのメソッドと変数を定義することはできません。
- システム定義の列挙型は Web サービスマソッドで使用できません。
- トリガにはメソッドを定義できないため、トリガに キーワードを使用できません。
- キーワードで定義されているメソッドを含むすべてのクラスは として宣言する必要があります。メソッド、または内部クラスを として宣言した場合、最上位(外部) クラスも として宣言する必要があります。
- キーワードで定義されるメソッドは本質的にグローバルです。これらのメソッドを、クラスにアクセスできる Apex コードで使用できます。 キーワードは、 より多くのアクセスを可能にするアクセス修飾子の一種と考えることができます。
- キーワードを使用するメソッドを として定義する必要があります。
- 管理パッケージコードの メソッドまたは変数を廃止することはできません。
- 特定の Apex 要素に SOAP アナログがないため、 キーワードで定義されたメソッドは、次の要素をパラメータとして使用できません。これらの要素はメソッド内で使用できますが、戻り値としてマークすることはできません。
  - ◊ Map
  - ◊ Set
  - ◊ Pattern オブジェクト
  - ◊ Matcher オブジェクト
  - ◊ Exception オブジェクト
- キーワードは、Web サービスの一部として公開するメンバー変数と共に使用する必要があります。これらのメンバー変数は としてマークしません。
- Salesforce は、アクセスが 制限あり になっている AppExchange パッケージからの Web サービスへのアクセスと 要求を拒否します。
- 項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存(コンパイル)した Apex クラスとトリガにはランタイムエラーが発生します。

次の例は、Web サービス変数と Web サービスマソッドを持つクラスを示します。







この Web サービスは AJAX を使用して呼び出すことができます。詳細は、[「Apex in AJAX」](#) (ページ 160)を参照してください。

## Web サービスマソッドのオーバーロード

SOAPおよびWSDLでは、メソッドのオーバーロードはサポートされません。そのため、Apexでは、キーワードでマークされた 2 つのメソッドに同じ名前を付けることはできません。1 つのクラスで同じ名前を持つ複数の Web サービスマソッドを使用すると、コンパイル時エラーが発生します。

## 第 12 章

### Apex クラスを REST Web サービスとして公開

---

トピック:

- [Apex REST の概要](#)
- [Apex REST アノテーション](#)
- [Apex REST のメソッド](#)
- [Apex REST Web サービスマソッドを使用したデータの公開](#)
- [Apex REST のコードサンプル](#)

外部アプリケーションが REST アーキテクチャによってコードとアプリケーションにアクセスできるように、Apex クラスとメソッドを公開することができます。このセクションでは、Apex クラスを REST Web サービスとして公開する方法について説明します。クラスとメソッドのアノテーションについて学習し、この機能を実装する方法を示すコードサンプルを紹介します。

## Apex REST の概要

外部アプリケーションが REST アーキテクチャによってコードとアプリケーションにアクセスできるように、Apex クラスとメソッドを公開することができます。これは、REST リソースとして公開する Apex クラスをアノテーションで定義して行います。同様に、他のアノテーションもメソッドに追加して、REST を通じて公開します。詳細は、「[Apex REST アノテーション](#)」(ページ 211)を参照してください。

### ガバナ制限

Apex REST クラスへのコールは、組織の API ガバナ制限のカウントの対象です。すべての標準の Apex ガバナ制限は、Apex REST クラスに適用されます。たとえば、要求または応答の最大サイズは 3 MB です。詳細は、「[実行ガバナと制限について](#)」を参照してください。

### 認証

Apex REST は、次の認証メカニズムをサポートしています。

- OAuth 2.0
- セッション ID

『[REST API Developer's Guide](#)』の「[ステップ 2: 認証を設定する](#)」を参照してください。

## Apex REST アノテーション

6 つの新しいアノテーションが追加され、Apex クラスを RESTful Web サービスとして公開できるようにするようになりました。

- `yourUrl`
- 
- 
- 
- 

## Apex REST のメソッド

Apex REST は、リソースを表現するために、JSON と XML の 2 つの形式をサポートしています。JSON による表現は、要求またはレスポンスボディにデフォルトで渡され、形式は、HTTP ヘッダーのプロパティで示されます。本文は、Apex メソッドへのパラメータがない場合、HttpRequest オブジェクトから Blob として取得できます。パラメータが Apex メソッドに定義されている場合、リクエストボディをそれらのパラメータに並列化する試行が行われます。Apex メソッドの戻り値が non-void である場合、リソースの表現はレスポンスボディに逐次化されます。

次の戻り型とパラメータ型を使用できます。

- Apex プリミティブ (sObject と Blob を除く)
- sObjects
- Apex プリミティブまたは sObject のリストまたは対応付け (String キーの対応付けのみをサポート)
- 上記の型のメンバー変数を含む **ユーザ定義型**



**メモ:** Apex REST では、Chatter in Apex オブジェクトの XML 逐次化および並列化はサポートされません。Apex REST では、Chatter in Apex オブジェクトの JSON 逐次化および並列化はサポートされません。

または アノテーションのあるメソッドには、パラメータがありません。これは、GET 要求と DELETE 要求にはリクエストボディがなく、並列化するものがいためです。

1 つの Apex クラスにアノテーション が付加されている場合、同一の HTTP 要求メソッドでアノテーションが付加されたメソッドを複数含めることはできません。たとえば、同じクラスにアノテーション が付加されたメソッドを 2 つ含めることはできません。



**メモ:** Apex REST は現在、Content-Type の要求をサポートしていません。

## Apex REST メソッドに関する考慮事項

Apex REST メソッドを定義する場合、次の事項を考慮してください。

- オブジェクトおよび オブジェクトは、静的 オブジェクトによってこれらのオブジェクト でデフォルトで Apex メソッドで利用できます。この例では、 によってこれらをオブジェクト にアクセスする方法を示します。

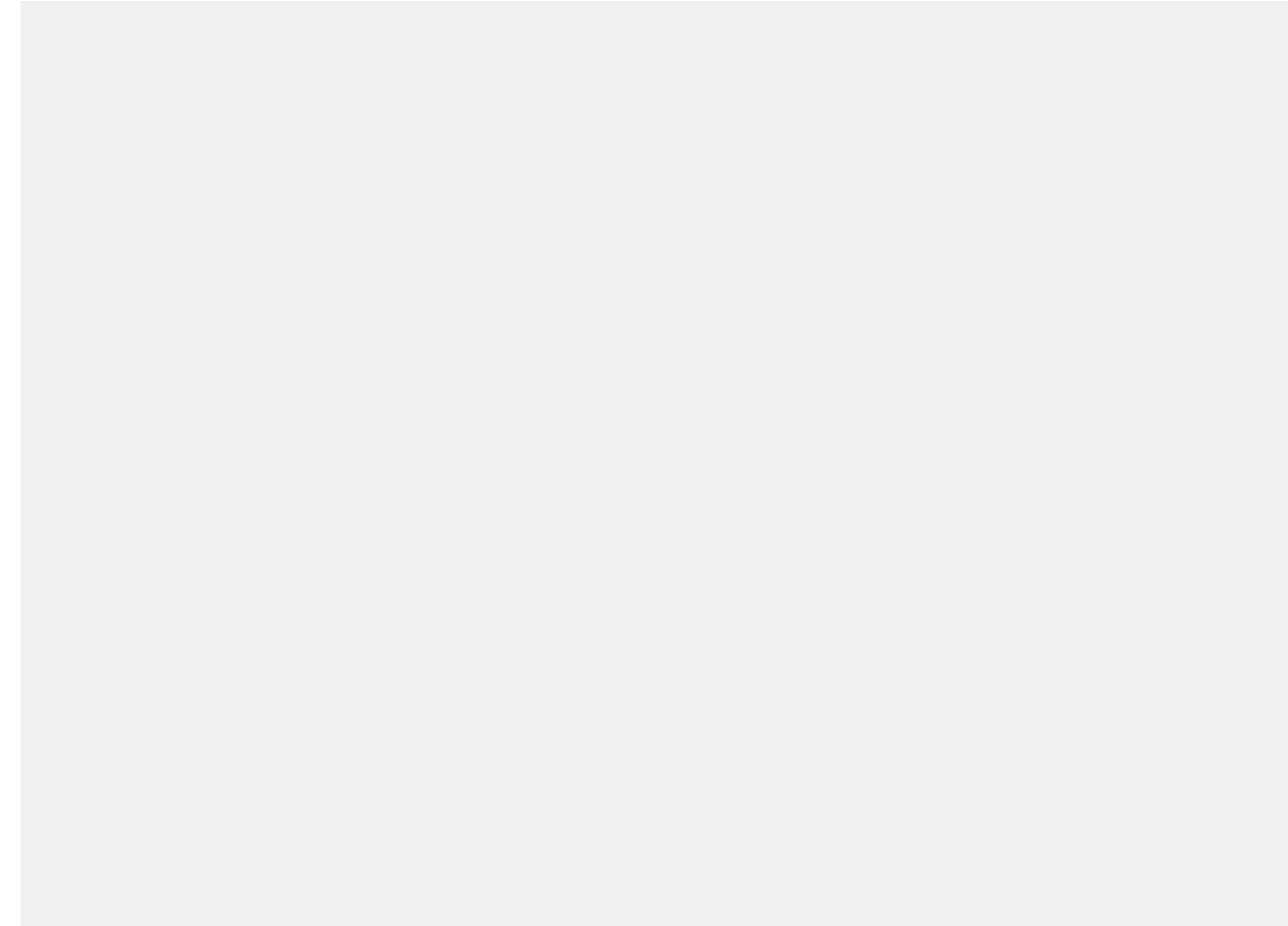


- Apex メソッドにパラメータがない場合、Apex REST は HTTP リクエストボディをプロパティにコピーします。メソッドにパラメータがある場合、Apex REST はデータをそれらのパラメータに並列化しようとします。ただし、データは プロパティには並列化されません。
- Apex REST は、応答に同様の逐次化ロジックを使用します。Apex メソッドの戻り値の型が non-void である場合、そのメソッドの戻り値は、 に逐次化されます。
- Apex REST メソッドは、管理パッケージと未管理パッケージで使用できます。管理パッケージに含まれる Apex REST メソッドをコールする場合、REST のコール URL に管理パッケージの名前空間を含める必要があります。たとえば、 という管理パッケージ名前空間にクラスが含まれてあり、Apex REST メソッドが という URL 対応付けを使用している場合、これらのメソッドをコールするために REST によって使用される URL は、 という形式になります。管理パッケージの詳細は、「[管理パッケージでの Apex の開発](#)」を参照してください。

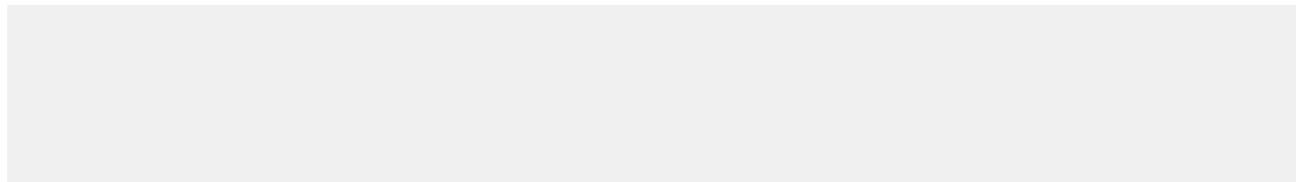
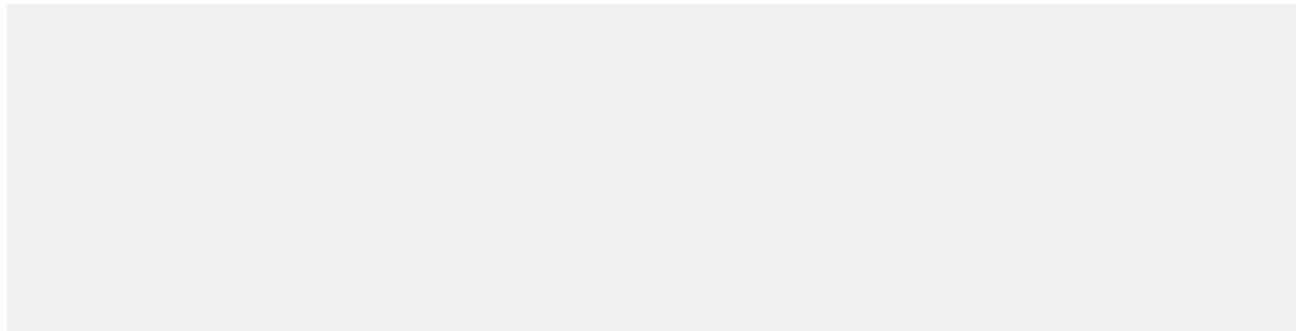
## ユーザ定義型

ユーザ定義型を Apex REST メソッドのパラメータとして使用できます。Apex REST は、ユーザ定義型の 、 または クラスメンバー変数に要求データを並列化します。ただし、変数が または

として宣言されている場合は、並列化されません。たとえば、ユーザ定義型パラメータを含む Apex REST メソッドは次のように記述されます。



このメソッドの有効な JSON および XML 要求データは、次のように記述されます。

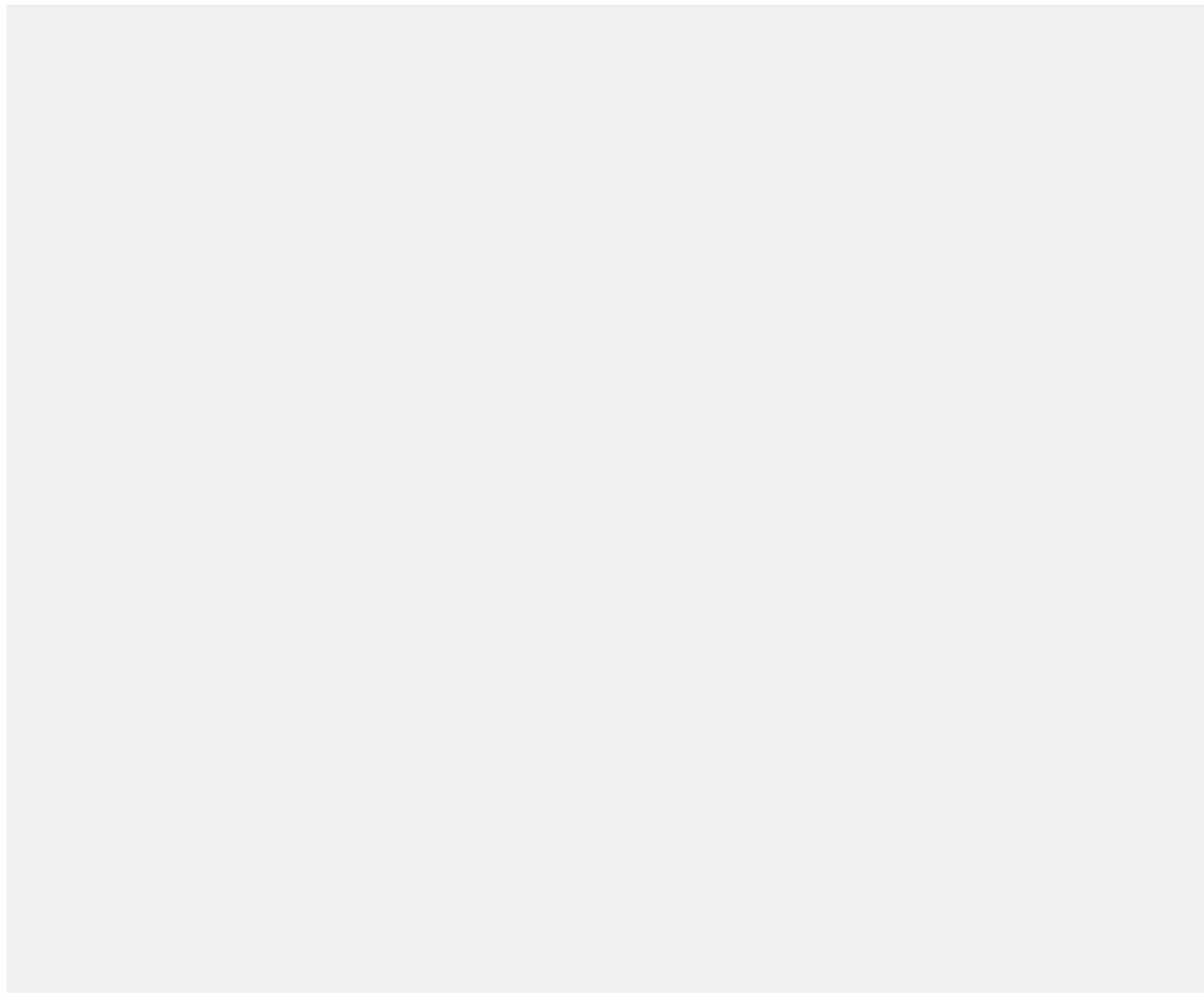




または の値が上記例の要求データに提供されている場合、HTTP 400 状況コード応答が生成されます。 または クラスメンバー変数は、次の Apex REST が許可する型である必要があります。

- Apex プリミティブ (sObject と Blob を除く)
- sObject
- Apex プリミティブまたは sObject のリストまたは対応付け (String キーの対応付けのみをサポート)

Apex REST メソッドのパラメータとして使用されるユーザ定義型を作成する場合、実行時に、ユーザ定義型でサイクルとなるクラスメンバー変数の定義(相互に依存する定義)を挿入しないようにしてください。次に、簡単な例を示します。

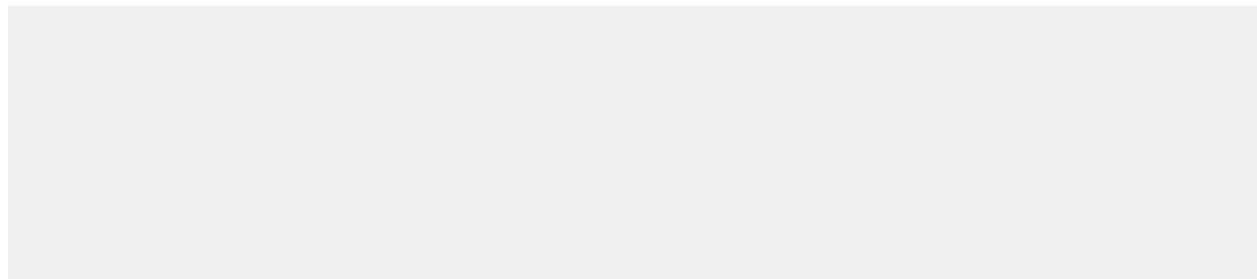
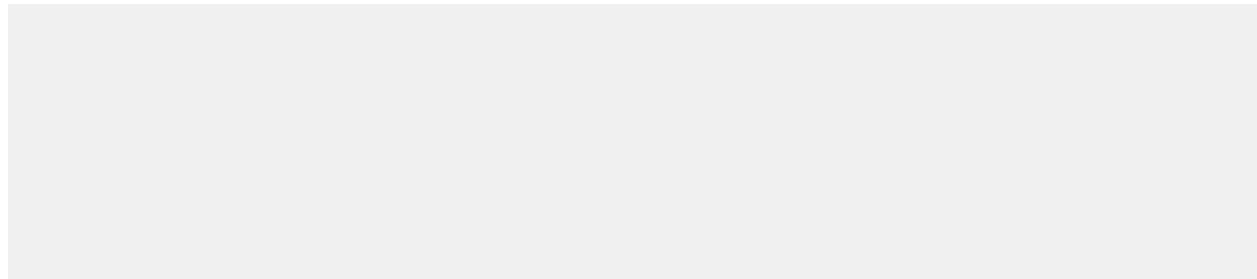
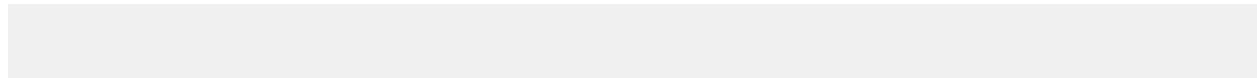


前の例で示すコードはコンパイルされますが、実行時に要求が発行されると、Apex REST は インスタンスと インスタンス間のサイクルを検出し、HTTP 400 状況コードエラー応答を生成します。

## 要求データの考慮事項

Apex REST メソッドにおける要求データの考慮事項をいくつか示します。

- Apex パラメータの名前は重要です。ただし、順番は考慮されません。たとえば、XML と JSON の有効な要求は次のようにになります。



- 一部のパラメータと戻り値の型は、要求の Content-Type として、または応答の許容形式として XML で使用することはできないため、これらのパラメータまたは戻り値の型を使用したメソッドを XML で使用することはできません。  
など、コレクションの対応付けまたはコレクションはサポートされていません。ただし、これらの型を JSON で使用することはできます。パラメータリストに XML では無効な型が含まれており、その XML が送信されると、HTTP 415 の状況コードが返されます。戻り値の型が XML で無効な型であり、XML が要求された応答形式である場合、HTTP 406 の状況コードが返されます。
- JSON または XML の要求データについては、Boolean パラメータの有効な値は、  
（これらは大文字と小文字を区別しません）、  
および  
（文字列「1」または「0」ではなく数値）です。Boolean パラメータに他の値が渡されると、エラーになります。
- JSON または XML 要求データに同じ名前のパラメータが複数含まれる場合は、HTTP 400 状況コードエラー応答が返されます。たとえば、メソッドが  
という入力パラメータを指定した場合、次の JSON 要求データはエラーになります。



同様に、ユーザ定義型についても、要求データに同一のユーザ定義型メンバー変数が複数含まれる場合、エラーになります。たとえば、次の Apex REST メソッドとユーザ定義型があるとします。

次の JSON 要求データもエラーになります。

- 要求データのパラメータの1つに null 値を指定する必要がある場合、すべてのパラメータを省略するか、null 値を指定できます。JSON では、`null` を値として指定できます。XML では、`nil` 値と共に名前空間を使用する必要があります。
- XML 要求データについては、メソッドが使用する Apex 名前空間を参照する XML 名前空間を指定する必要があります。たとえば、Apex REST メソッドを次のように定義するとします。

次の XML 要求データを使用できます。

XML 名前空間と Apex についての詳細は、[「XML 名前空間」](#) を参照してください。

## 応答の状況コード

応答の状況コードは、自動的に設定されます。この表では、一部の HTTP 状況コードと HTTP 要求メソッドでの意味を説明します。応答状況コードの完全なリストは、[「RestResponse メソッド」](#) を参照してください。

要求メソッド	応答の状況コード	説明
GET	200	要求は成功しました。
PATCH	200	要求は成功しました。戻り値の型は non-void です。
PATCH	204	要求は成功しました。戻り値の型は void です。
DELETE、GET、PATCH、POST、PUT	400	未処理のユーザ例外が発生しました。
DELETE、GET、PATCH、POST、PUT	403	指定された Apex クラスにアクセスできません。
DELETE、GET、PATCH、POST、PUT	404	URL は既存の アノテーションで対 応付けられていません
DELETE、GET、PATCH、POST、PUT	404	URL 拡張はサポートされていません。
DELETE、GET、PATCH、POST、PUT	404	指定された名前空間を使用する Apex クラスは見つかりませんでした。

要求メソッド	応答の状況コード	説明
DELETE、GET、PATCH、POST、PUT	405	要求メソッドには対応する Apex メソッドがありません。
DELETE、GET、PATCH、POST、PUT	406	ヘッダーの Content-Type プロパティは JSON または XML 以外の値に設定されました。
DELETE、GET、PATCH、POST、PUT	406	HTTP 要求に指定されたヘッダーはサポートされていません。
GET、PATCH、POST、PUT	406	形式に指定された XML の戻り値の型はサポートされていません。
DELETE、GET、PATCH、POST、PUT	415	XML パラメータの型はサポートされていません。
DELETE、GET、PATCH、POST、PUT	415	HTTP 要求のヘッダーに指定された Content-Header 型はサポートされていません。
DELETE、GET、PATCH、POST、PUT	500	未処理の Apex 例外が発生しました。

## Apex REST Web サービスマソッドを使用したデータの公開

カスタム Apex REST Web サービスマソッドの呼び出しには、必ずシステムコンテキストを使用します。その結果、現在のユーザの証明書は使用されず、これらのメソッドにアクセスできるすべてのユーザが、権限、項目レベルのセキュリティ、共有ルールに関係なく、全機能を使用できます。そのため、Apex REST アノテーションを使用してメソッドを公開する開発者は、ユーザが機密情報データを不用意に公開しないよう注意する必要があります。



警告: Apex REST API を使用して公開されている Apex クラスマソッドは、デフォルトではオブジェクト権限と項目レベルのセキュリティを適用しません。Apex REST API メソッドがアクセスしようとしているオブジェクトと項目に対する現在のユーザのアクセスレベルをチェックするには、適切な Object Describe Result メソッドまたは Field Describe Result メソッドを使用することをお勧めします。『[sObject Describe Result メソッド](#)』および『[Describe Field Result メソッド](#)』を参照してください。

また、共有ルール(レコードレベルアクセス)は、キーワードでクラスを宣言するときのみに適用されます。この要件は、Apex REST API によって公開されるクラスを含むすべての Apex クラスに適用されます。Apex REST API メソッドに共有ルールを適用するには、これらのメソッドを含むクラスをキーワードで宣言します。『[または](#)』または『[の使用](#)』を参照してください。

## Apex REST のコードサンプル

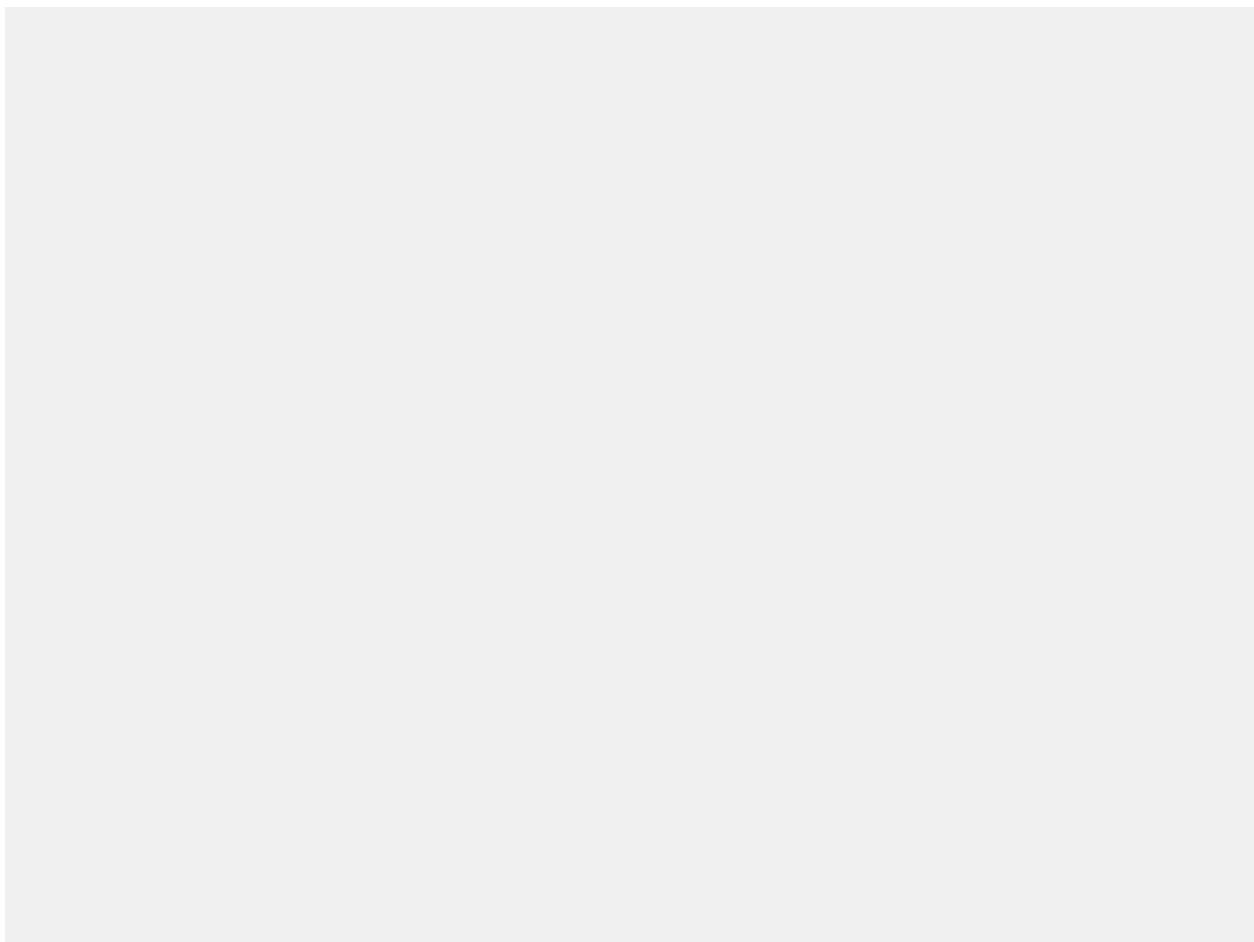
これらのコードサンプルでは、REST アーキテクチャによる Apex クラスとメソッドの公開方法とクライアントのリソースのコール方法を説明します。

- [Apex REST の基本コードサンプル](#): レコードを削除、取得、および更新するためにコールできる 3 つのメソッドを使用した、Apex REST クラスの例を示します。
- [RestRequest を使用した Apex REST のコードサンプル](#): RestRequest オブジェクトを使用して添付ファイルをレコードに追加する Apex REST クラスの例を示します。

## Apex REST の基本コードサンプル

このサンプルでは、3 つの異なる HTTP 要求メソッドを処理する簡単な REST API を Apex に実装する方法を示します。を使った認証についての詳細は、『*REST API Developer's Guide*』の「[クイックスタート](#)」のセクションを参照してください。

1. [設定] から [開発] > [Apex クラス] > [新規] をクリックし、このコードを新しいクラスに追加して、インスタンスに Apex クラスを作成します。



2. クライアントから `getAccount` メソッドをコールするには、コマンドラインウィンドウを開き、次のコマンドを実行して ID で取引先を取得します。

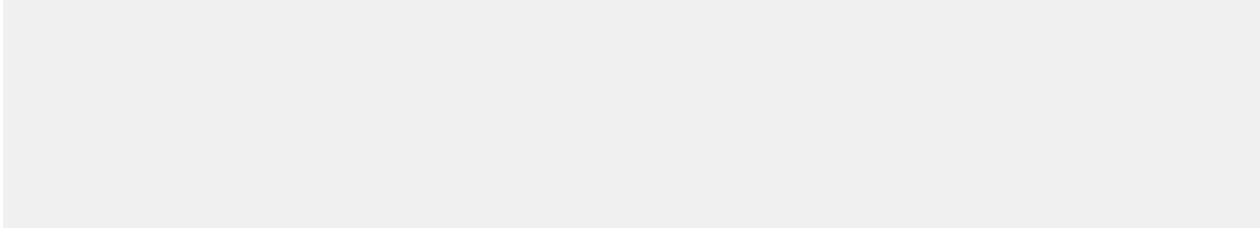
```
sessionId  
instance          accountId  
• sessionId を、ログイン応答でメモした要素に置き換えます。  
• instance を要素に置き換えます。  
• accountId を、組織に存在する取引先の ID に置き換えます。  
メソッドをコールすると、Salesforce は、次のようなデータを伴う JSON 応答を返します。
```

```
accountId  
accountId
```



メモ: このセクションの例では、名前空間による Apex クラスを使用していないため、URL に名前空間は含まれません。

3. 次のステップで作成する取引先のデータを含めるための `instance` というファイルを作成します。



4. コマンドラインウィンドウを使用して、次の `instance` コマンドを実行し、新しい取引先を作成します。

```
sessionId  
instance
```

メソッドをコールすると、Salesforce は、次のようなデータを伴う応答を返します。

```
accountId
```

`accountId` は、POST 要求で作成した取引先の ID です。

5. コマンドラインウィンドウを使用して、次の `instance` コマンドを実行し、ID の指定によって取引先を削除します。

```
sessionId  
instance  
accountId
```

## 関連リンク

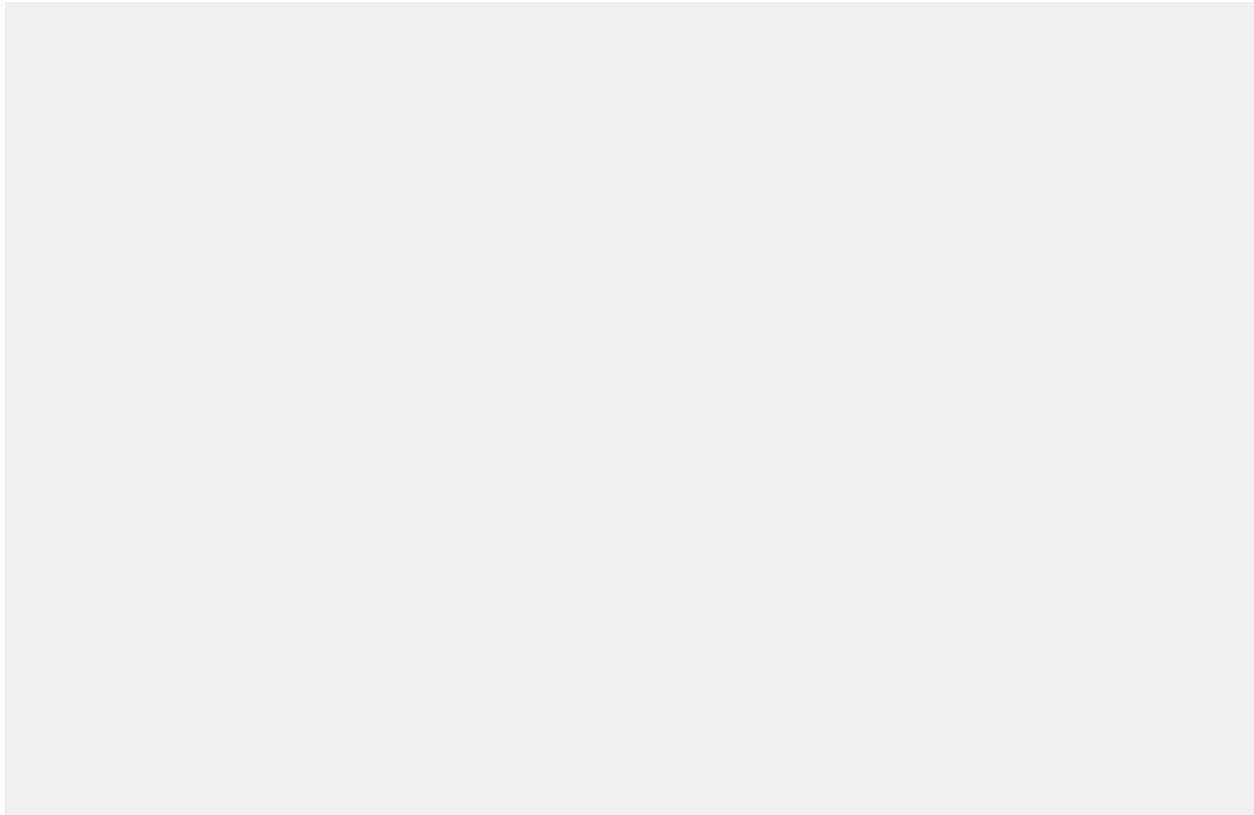
[Apex REST アノテーション](#)

## RestRequest を使用した Apex REST のコードサンプル

次のサンプルでは、`RestRequest` オブジェクトを使用して、ケースに添付ファイルを追加する方法を示します。使用した認証についての詳細は、『REST API Developer's Guide』の「[クイックスタート](#)」のセクションを参照してください。このコードでは、バイナリファイルのデータは `RestRequest` オブジェクトに保存され、Apex サービスクラスはその `RestRequest` オブジェクトのバイナリデータにアクセスします。

1. [設定] から [開発] > [Apex クラス] をクリックして、インスタンスに Apex クラスを作成します。[新規] をクリックして、次のコードを新しいクラスに追加します。



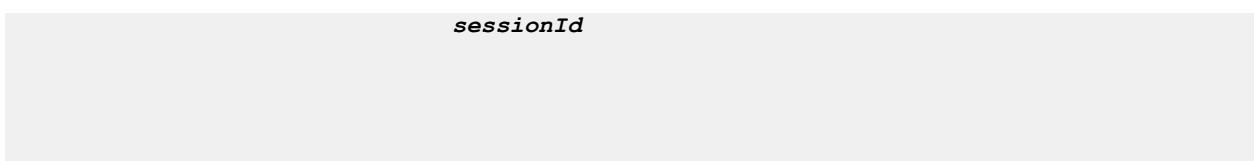


2. コマンドラインウィンドウを開き、次のコマンドを実行して、ケースに添付ファイルをアップロードします。

```
sessionId  
file  
instance  
caseId
```

- *sessionId* を、ログイン応答でメモした要素に置き換えます。
- *instance* を要素に置き替えます。
- *caseId* を、添付ファイルを追加するケースの ID に置き換えます。
- *file* を、添付するファイルのパスとファイル名に置き換えます。

コマンドは次のようにになります (*sessionId* は、実際のセッション ID です)。



メモ: このセクションの例では、名前空間による Apex クラスを使用していないため、URL に名前空間は含まれません。

Apex クラスは、添付ファイル ID を含む次のような JSON 応答を返します。

3. 添付ファイルと画像がケースに追加されたことを確認するには、[ケース] に移動し、[すべての進行中ケース] ビューを選択します。ケースをクリックし、添付ファイルの関連リストまでスクロールダウンします。作成した添付ファイルが表示されます。

## 第 13 章

### Apex を使用したコールアウトの呼び出し

トピック:

- リモートサイトの設定の追加
- SOAP サービス: WSDL ドキュメントからのクラスの定義
- HTTP コールアウトの呼び出し
- 証明書の使用
- コールアウトの制限事項

Apex コールアウトを使用して、外部 Web サービスへのコールを作成、または Apex コードから HTTP 要求を送信して応答を受信することによって、Apex を外部サービスと密接に統合することができます。Apex は、SOAP および WSDL、または HTTP サービス (RESTful サービス) を使用する Web サービスと統合できます。



メモ: Apex コールアウトが外部サイトを呼び出す前に、そのサイトを [リモートサイトの設定] ページで登録する必要があります。登録しない場合、コールアウトが失敗します。Salesforce では未承認のネットワークアドレスへのコールが行われないようになります。

2 種類のコールアウトの詳細は、次の項を参照してください。

- SOAP サービス: WSDL ドキュメントからのクラスの定義 (ページ 373)
- HTTP コールアウトの呼び出し (ページ 390)



ヒント: コールアウトによって、Apex は外部 Web または HTTP サービスを呼び出すことができます。Apex Web サービスを使用すると、外部アプリケーションは Web サービスを使用して Apex メソッドを呼び出すことができます。

## リモートサイトの設定の追加

Apex コールアウトが外部サイトを呼び出す前に、そのサイトを [リモートサイトの設定] ページで登録する必要があります。登録しない場合、コールアウトが失敗します。Salesforce では未承認のネットワークアドレスへのコールが行われないようにします。

リモートサイトの設定を追加する手順は、次のとおりです。

1. [設定] で、[セキュリティのコントロール] > [リモートサイトの設定] をクリックします。
2. [新規リモートサイト] をクリックします。
3. リモートサイト名 には、分かりやすい名前を入力してください。
4. リモートサイトの URL を入力します。
5. 必要に応じて、サイトの説明を入力します。
6. [保存] をクリックします。

## SOAP サービス: WSDL ドキュメントからのクラスの定義

クラスは、ローカルハードドライブまたはネットワークに保管されている WSDL ドキュメントから自動的に生成できます。WSDL ドキュメントを使ってクラスを作成すると、開発者は Apex コードの中で外部 Web サービスへのコールアウトすることができます。



メモ: 可能な場合には、アウトバウンドメッセージを使用して、インテグレーションソリューションを処理します。必要な場合に限り、サードパーティの Web サービスのコールアウトを使用します。

WSDL から Apex クラスを作成する手順は、次のとおりです。

1. アプリケーションで、[設定] から [開発] > [Apex クラス] をクリックします。
2. [WSDL からの生成] をクリックします。
3. [参照] をクリックして、ローカルハードドライブまたはネットワーク上の WSDL ドキュメントを選択するか、フルパスを入力します。この WSDL ドキュメントが、作成する Apex クラスの基礎となります。



メモ:

指定した WSDL ドキュメントに、送信ポートを参照する SOAP エンドポイントの場所が記載されている場合があります。

セキュリティ上の理由から、Salesforce では、指定できる送信ポートを、次のいずれかに制限します。

- 80: このポートは、HTTP 接続のみを受け付けます。
- 443: このポートは、HTTPS 接続のみを受け付けます。
- 1024–66535 (1024 と 66535 も含む): これらのポートは、HTTP 接続または HTTPS 接続を受け付けます。

4. [WSDL を解析] をクリックして、WSDL ドキュメントの内容を確認します。アプリケーションが、WSDL ドキュメント内の各名前空間のデフォルトクラス名を生成し、エラーがあれば報告します。WSDL に Apex クラスがサポートしていないスキーマ種別またはスキーマ構造が含まれているか、結果生成されるクラス名が 100 万文字という Apex クラスの制限を超える場合には、解析は失敗します。たとえば、Salesforce SOAP API WSDL は解析できません。
5. 必要に応じて、そのクラス名を変更します。それぞれの名前空間に対して同じクラス名を使用することにより、1 つのクラスに複数の WSDL 名前空間を保存できますが、Apex クラスは、合計 100 万文字以内にしてください。
6. [Apex コードの生成] をクリックします。ウィザードの最終ページには、正常に生成されたクラスと、他のクラスのエラーが表示されます。また、正常に生成されたコードを表示するためのリンクも示されます。

正常に生成された Apex クラスには、WSDL ドキュメントで示されるサードパーティ Web サービスをコールするスタブと種別クラスが含まれています。これらのクラスにより、Apex から外部の Web サービスをコールすることができます。

生成された Apex に関して次の点に注意してください。

- WSDL ドキュメントに Apex の予約語が含まれている場合は、Apex クラスが生成されるときに、その語の後ろに「\_」が付きます。たとえば、WSDL ドキュメントに「\_」があると、生成される Apex クラスでは「\_」になります。「[予約キーワード](#)」を参照してください。Apex 変数名でサポートされていない WSDL の要素名の文字の処理の詳細については、「[WSDL 使用についての考慮事項](#)」を参照してください。
- WSDL の操作に複数の要素を含む出力メッセージがある場合、生成された Apex は内部クラスの要素をラップします。WSDL の操作を示す Apex メソッドは、各要素ではなく内部クラスを返します。
- Apex クラス名にピリオド(.)は使用できないため、Apex クラスの生成に使用される WSDL 名に含まれるすべてのピリオドは、生成される Apex コードではアンダースコア(\_)で置き換えられます。

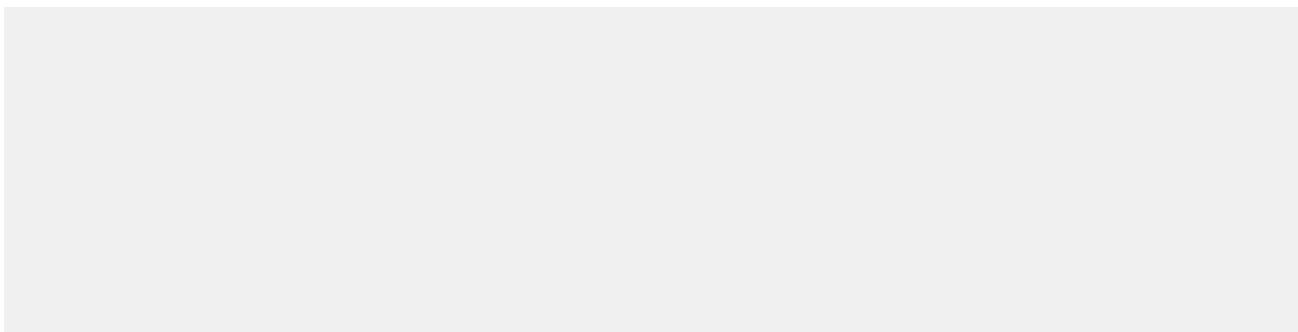
WSDL からクラスを生成した後、WSDL で参照される外部サービスを呼び出すことができます。

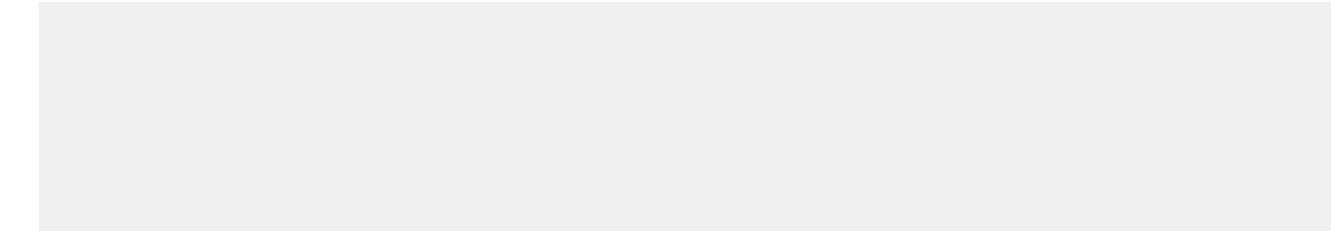


メモ: このトピックの残りの部分にあるサンプルを使用する前に、[生成されるコードについて](#)にある Apex クラスをコピーして組織に追加する必要があります。

## 外部サービスの呼び出し

WSDL ドキュメントを使用して Apex クラスを生成した後、外部サービスを呼び出すには、Apex コードにスタブのインスタンスを作成して、そこでメソッドをコールします。たとえば、Apex から [Strikelron IP アドレス検索サービス](#) を呼び出すために、次のようなコードを作成できます。





## HTTP ヘッダーのサポート

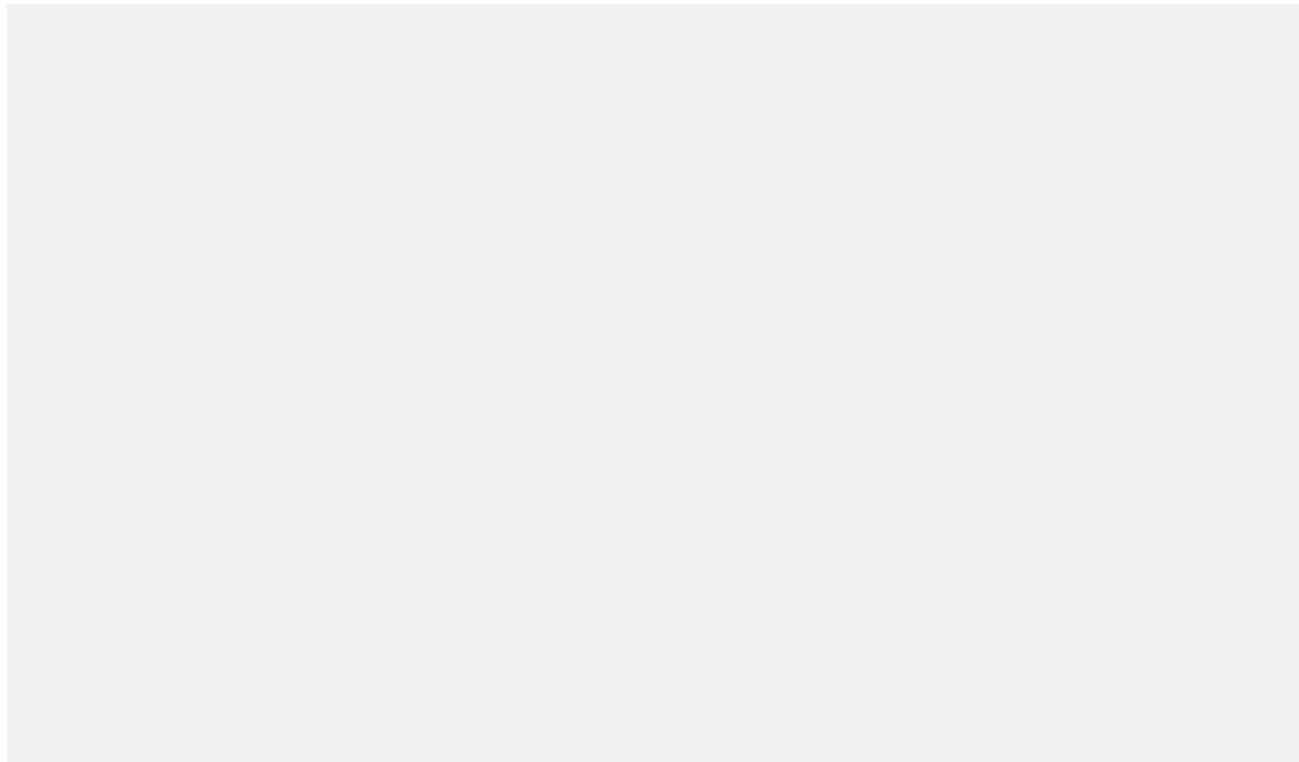
Web サービスコールアウトに HTTP ヘッダーを設定できます。たとえば、この機能を使用して認証ヘッダーに Cookie の値を設定できます。HTTP ヘッダーを設定するには、  
および  
をスタブに追加します。



メモ: API バージョン 16.0 以前では、コールアウトの HTTP 応答は、コンテンツタイプのヘッダーに関係なく UTF-8 を使用して復号化されます。API バージョン 17.0 以降では、HTTP 応答はコンテンツタイプのヘッダーで指定された符号化方式を使用して復号化されます。

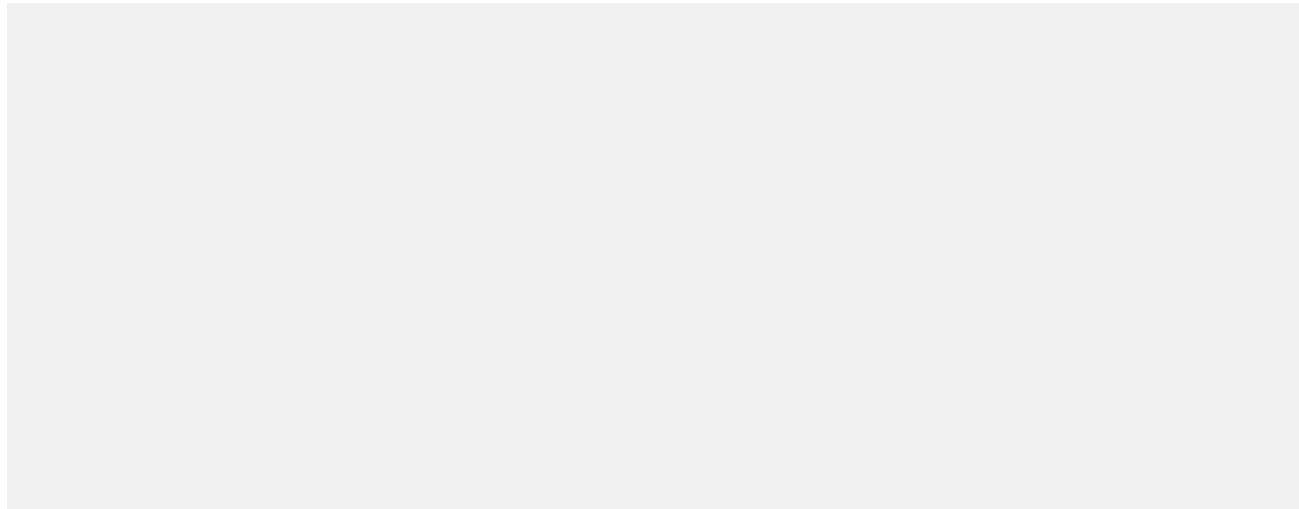
次のサンプルでは、[生成されるコードについて](#) (ページ 379)のサンプル WSDL ファイルを使用しています。

### Web サービスコールアウトでの HTTP ヘッダーの送信



の値を指定すると、標準ヘッダーセットがその値で上書きされます。

## Web サービスコールアウト応答からの HTTP 応答ヘッダーへのアクセス



のデフォルト値は `null` です。応答のヘッダーの内容にアクセスするには、  
を設定する必要があります。

## サポートされる WSDL の機能

Apex では、ドキュメントリテラルでラップした WSDL スタイルと次のプリミティブデータ型と組み込みデータ型のみをサポートしています。

スキーマの型	Apex の型
	<code>String</code>
	<code>Boolean</code>
	<code>date</code>
	<code>dateTime</code>
	<code>double</code>
	<code>double</code>
	<code>integer</code>
	<code>integer</code>
	<code>String</code>
	<code>long</code>
	<code>String</code>
	<code>String</code>
	<code>integer</code>
	<code>String</code>

スキーマの型	Apex の型
	String
	String
	String
	integer
	String
	integer
	String
	dateTime
	String
	integer
	long
	integer



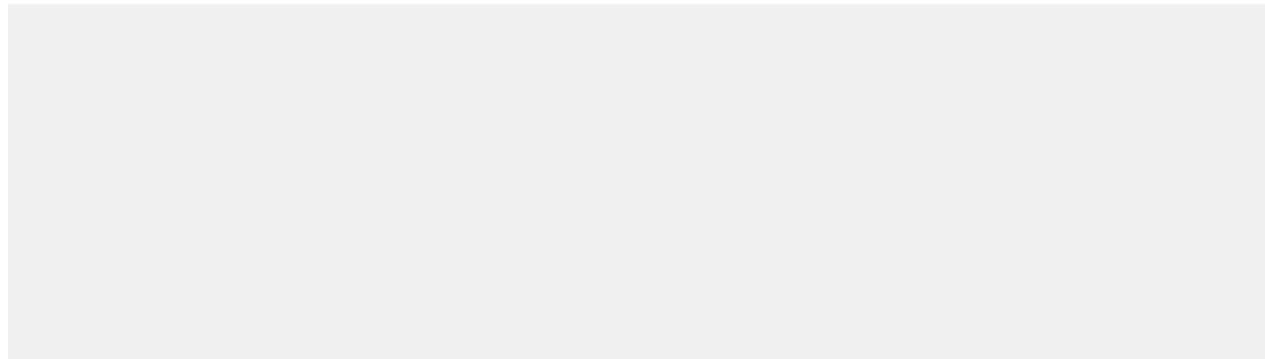
メモ: Salesforce データ型 anyType は、API バージョン 15.0 以降を使用して保存される Apex コードを生成するときに使用する WSDL ではサポートされません。API バージョン 14.0 以前を使用して保存されるコードでは、anyType は string に対応付けされます。

Apex では、次のスキーマ構造をサポートしています。

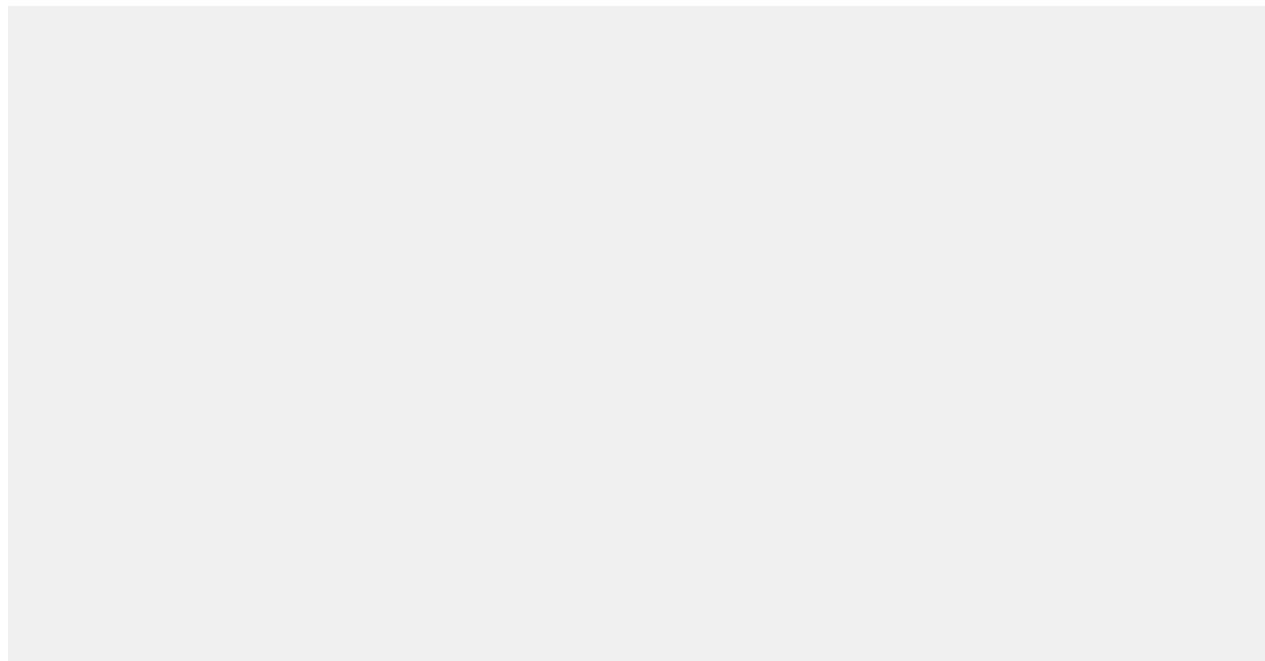
- 、API バージョン 15.0 以降を使用して保存した Apex コードにおいて
- 、API バージョン 15.0 以降を使用して保存した Apex コードにおいて、属性が次の制限付きでサポートされます。
  - ◊ 他の名前空間の はコールできません。
  - ◊ グローバル要素では を使用できません。
  - ◊ また、要素に が含まれる場合も、 または

Apex は次のようなその他の WSDL コンストラクタ、データ型、サービスをサポートしていません。

- RPC/符号化サービス
- 複数の \_\_\_\_\_ 、複数のサービス、または複数のバインドを含む WSDL サービス
- 外部スキーマをインポートする WSDL ファイル。たとえば、外部スキーマをインポートしている次の WSDL フラグメントは、サポートされていません。

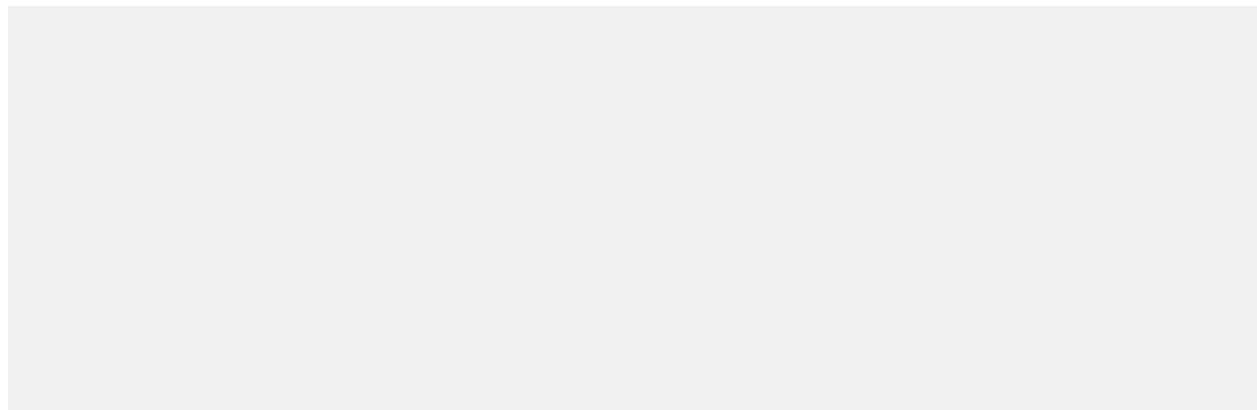


ただし、同じスキーマ内のインポートはサポートされています。次の例では、外部 WSDL は変換する WSDL に貼り付けられます。

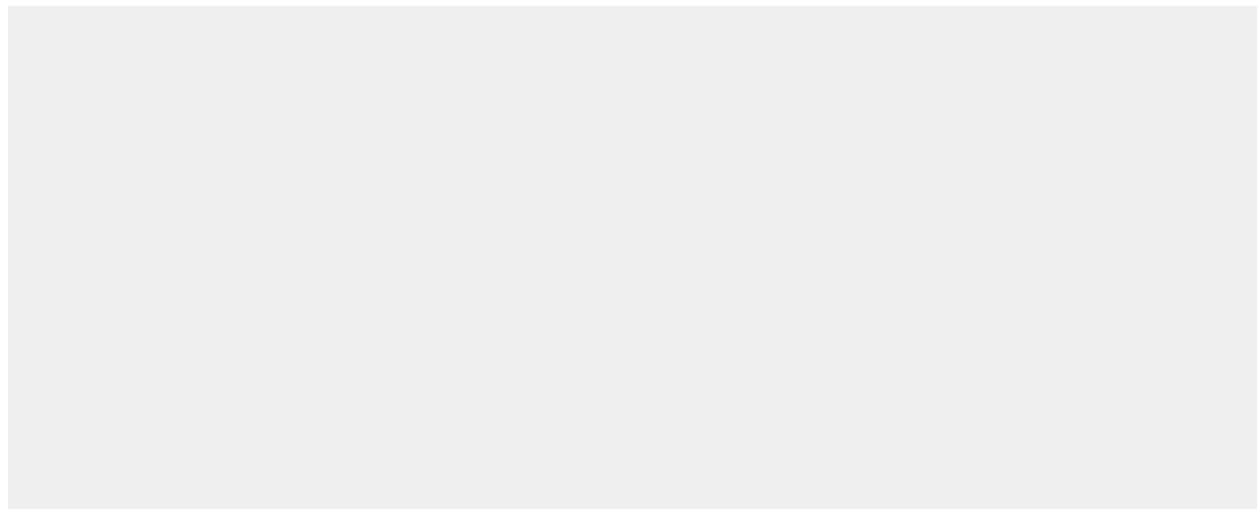


- 前の表に記載されていないスキーマの型
- Salesforce WSDL を含めた、サイズ制限を超えた WSDL

- ドキュメントリテラルでラップしたスタイルを使用しない WSDL。次の WSDL スニペットはドキュメントリテラルでラップしたスタイルを使用しないため、インポートしたときに「Unable to find complexType (complexType が見つかりません)」というエラーが発生します。

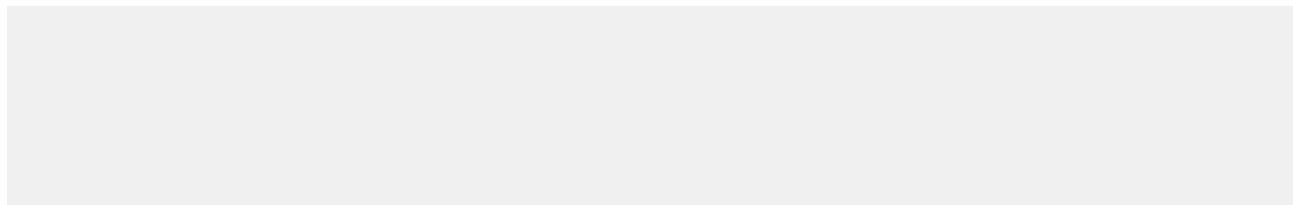


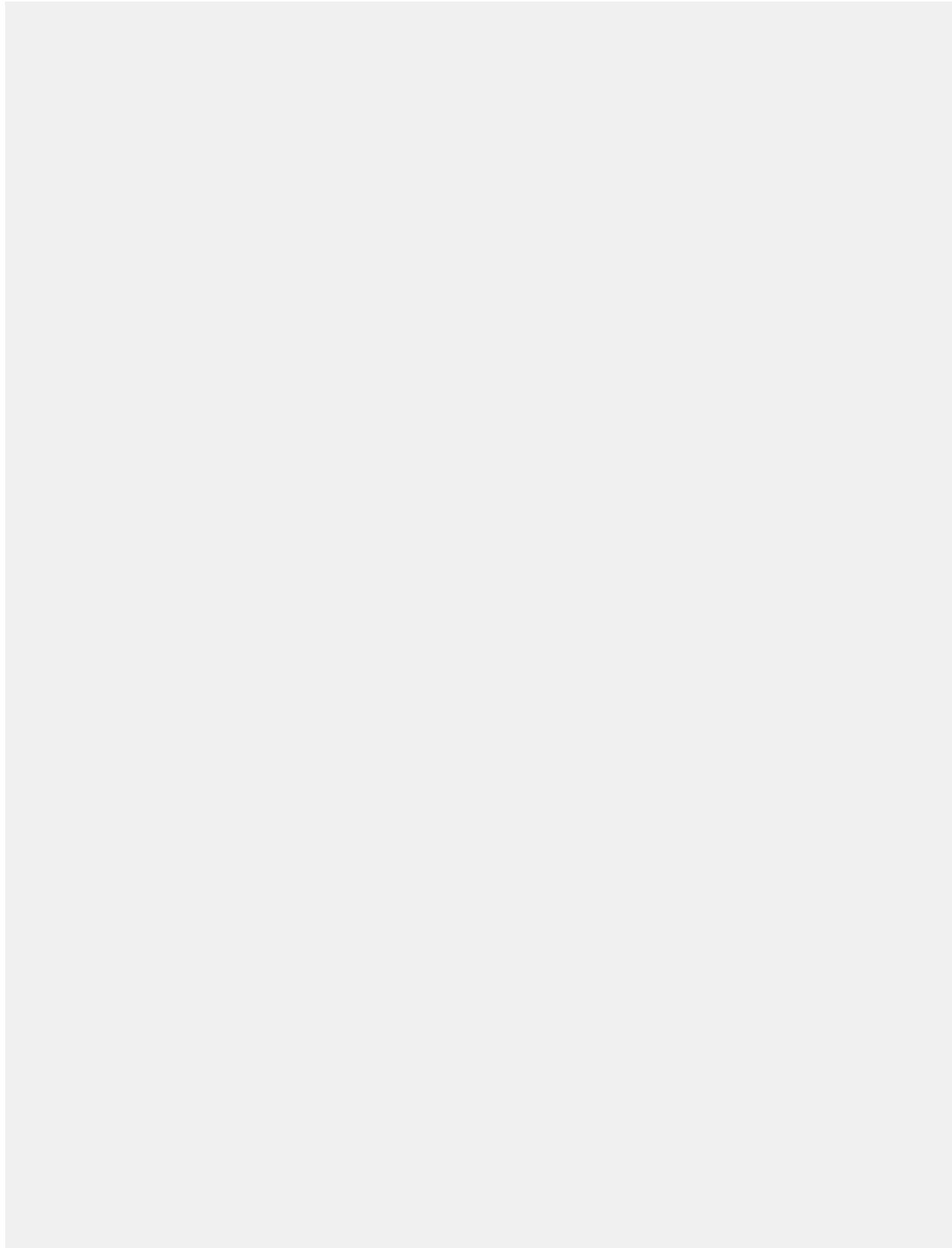
次のスニペットは、要素の順序を含む `<ns0:complexType>` として `<ns0:sequence>` 要素をラップするよう変更したものです。この場合はドキュメントリテラルのスタイルに従っているため、サポートされています。



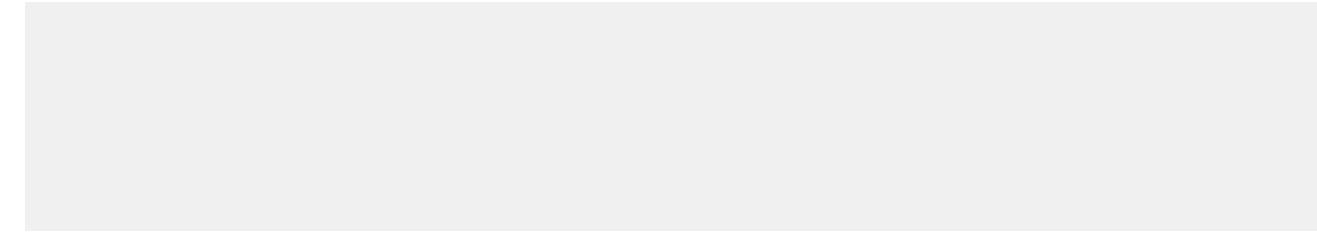
## 生成されるコードについて

次の例では、WSDL ドキュメントから Apex クラスがどのように作成されるかを示します。Apex クラスは、WSDL をインポートすると自動生成されます。次のコードは、サンプル WSDL ドキュメントを示します。

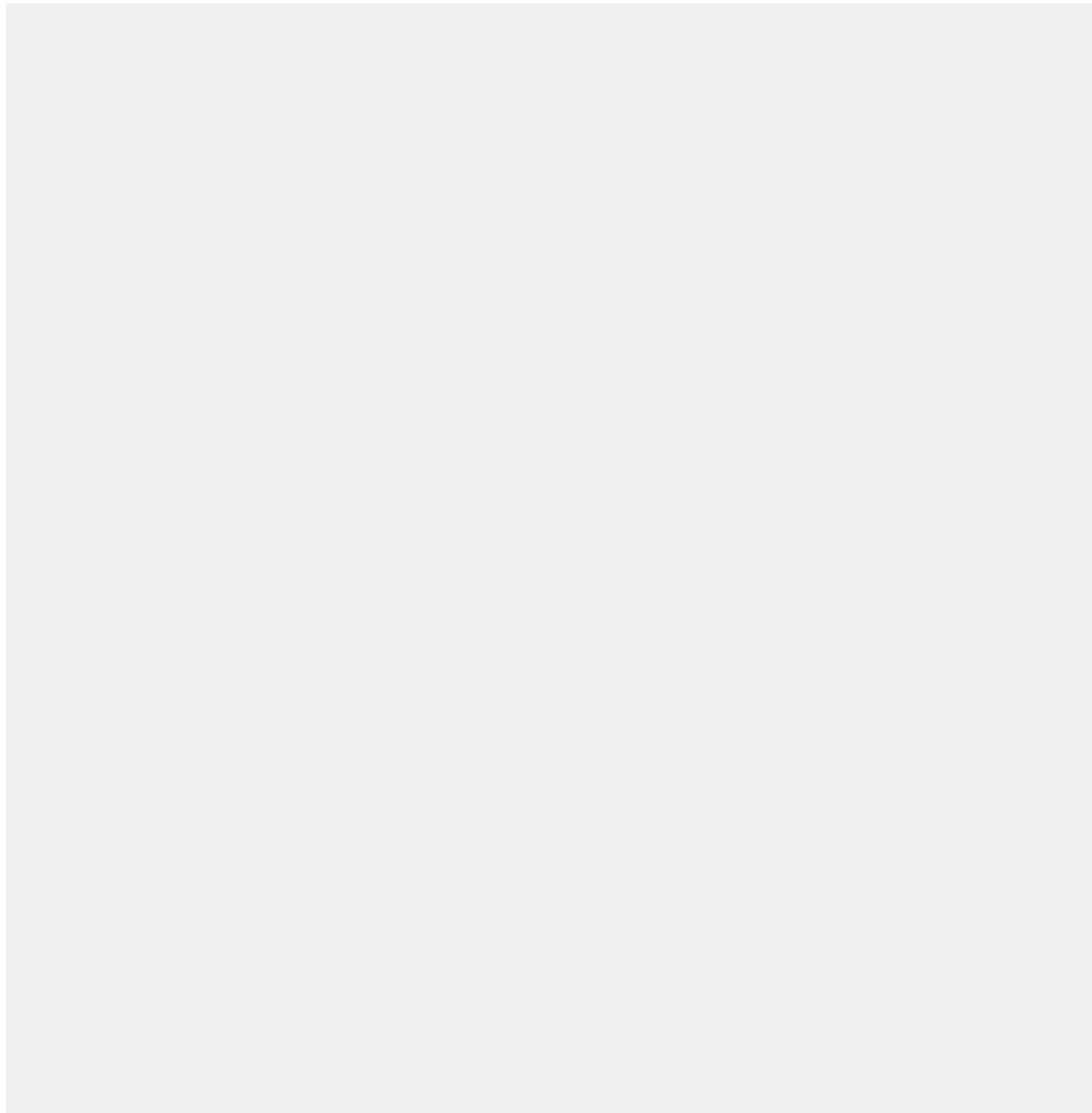


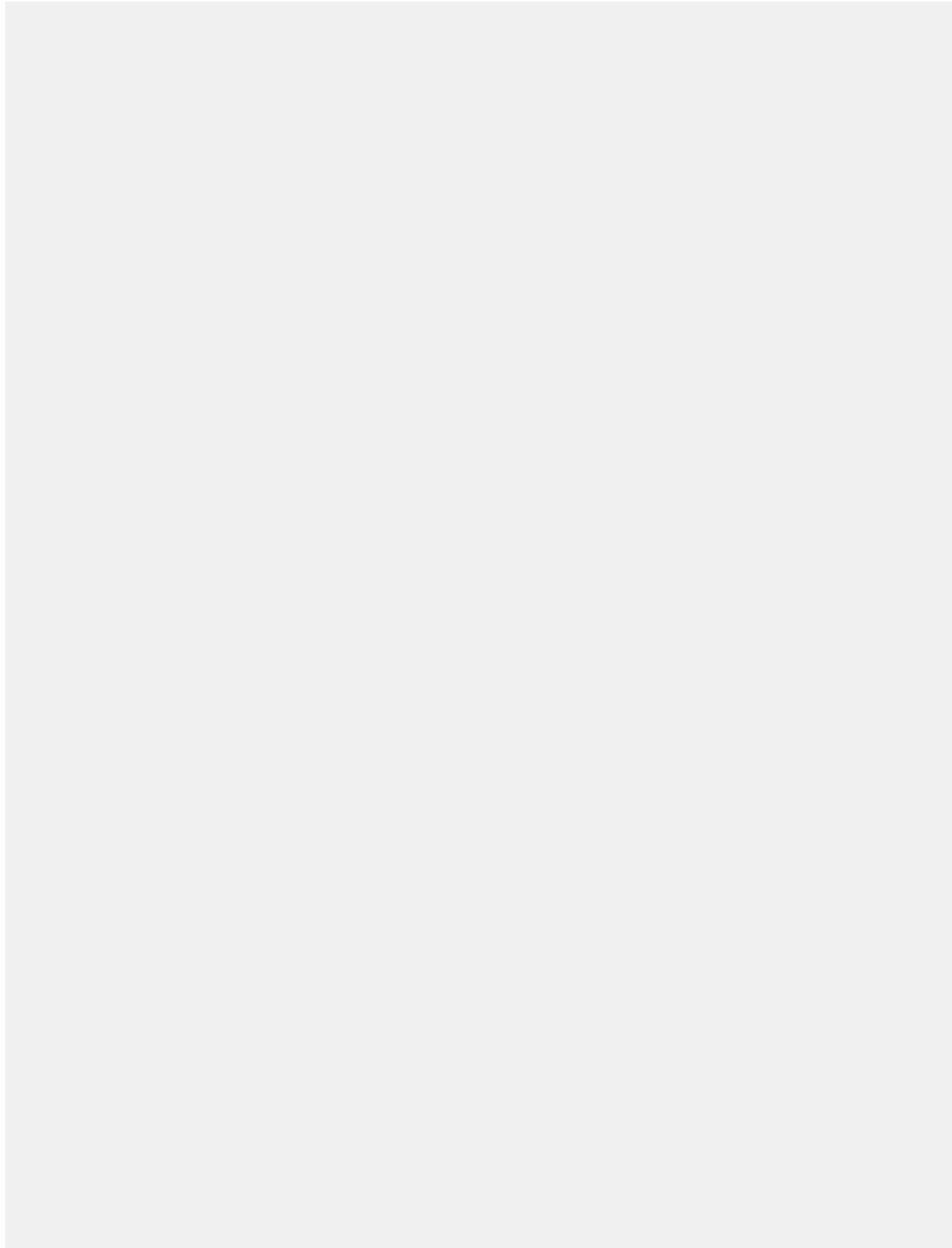


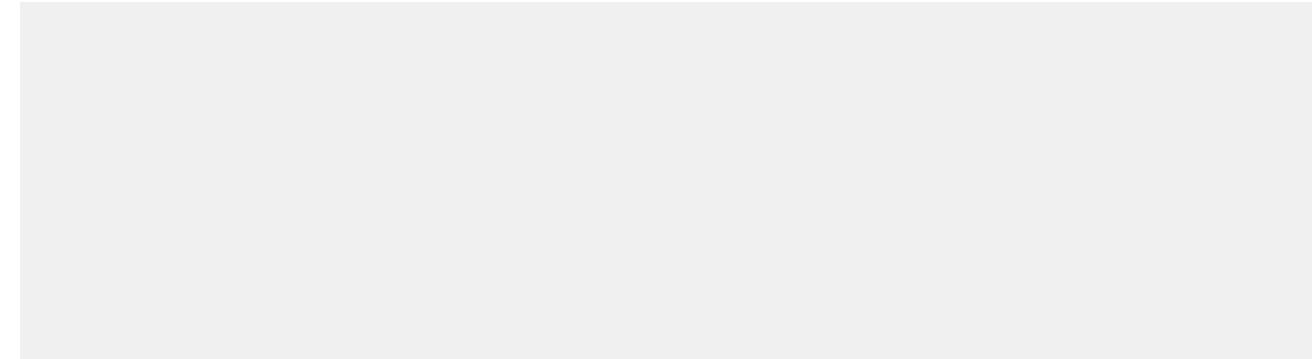




この WSDL ドキュメントから、次の Apex クラスが自動生成されます。クラス名は、WSDL をインポートしたときに指定した名前です。







元の WSDL ドキュメントからの次の対応付けに注意してください。

- WSDL 対象名前空間は Apex クラス名に対応付けされます。
- 複雑なデータ型はクラスになります。データ型の各要素は、クラスの公開項目です。
- WSDL ポート名はスタブクラスに対応付けます。
- WSDL の各操作は、公開メソッドに対応付けます。

上記で生成されたクラスを使用して、外部 Web サービスを呼び出すことができます。次のコードは、外部サーバ上の `WebServiceMock` メソッドをコールする方法を示します。



## Web サービスコールアウトのテスト

生成されたコードは、Web サービスをコールするために呼び出せるメソッドが含まれる Apex クラスとして保存されます。この Apex クラスと付随する他のコードをリリースまたはパッケージ化するには、生成されたクラス内のメソッドを含め、コードのテストカバー率が 75% に達している必要があります。デフォルトでは、テストメソッドは Web サービスコールアウトをサポートせず、Web サービスコールアウトを実行するテストはスキップされます。テストのスキップを回避し、コードカバー率を高めるために、Apex では組み込みのインターフェースと `WebServiceMock` メソッドを使用してテストメソッドで擬似応答を受信できます。

### Web サービスコールアウトをテストするための擬似応答の指定

WSDL から Apex クラスを作成した場合、自動生成されたクラスのメソッドが `WebServiceMock` をコールし、そこから外部サービスへのコールアウトが実行されます。これらのメソッドをテストする場合、`WebServiceMock` がコールされたときには常に擬似応答を生成するように Apex ランタイムに指示できます。そのためには、インターフェースを実装し、Apex ランタイムが送信する擬似応答を指定します。手順の詳細は次のとおりです。

最初に、インターフェースを実装し、`WebServiceMock` メソッドに擬似応答を指定します。

```
YourWebServiceMockImpl implements WebServiceMock
```

```
Map<String, Object> response,  
  
response.put('response_x', responseElement);
```



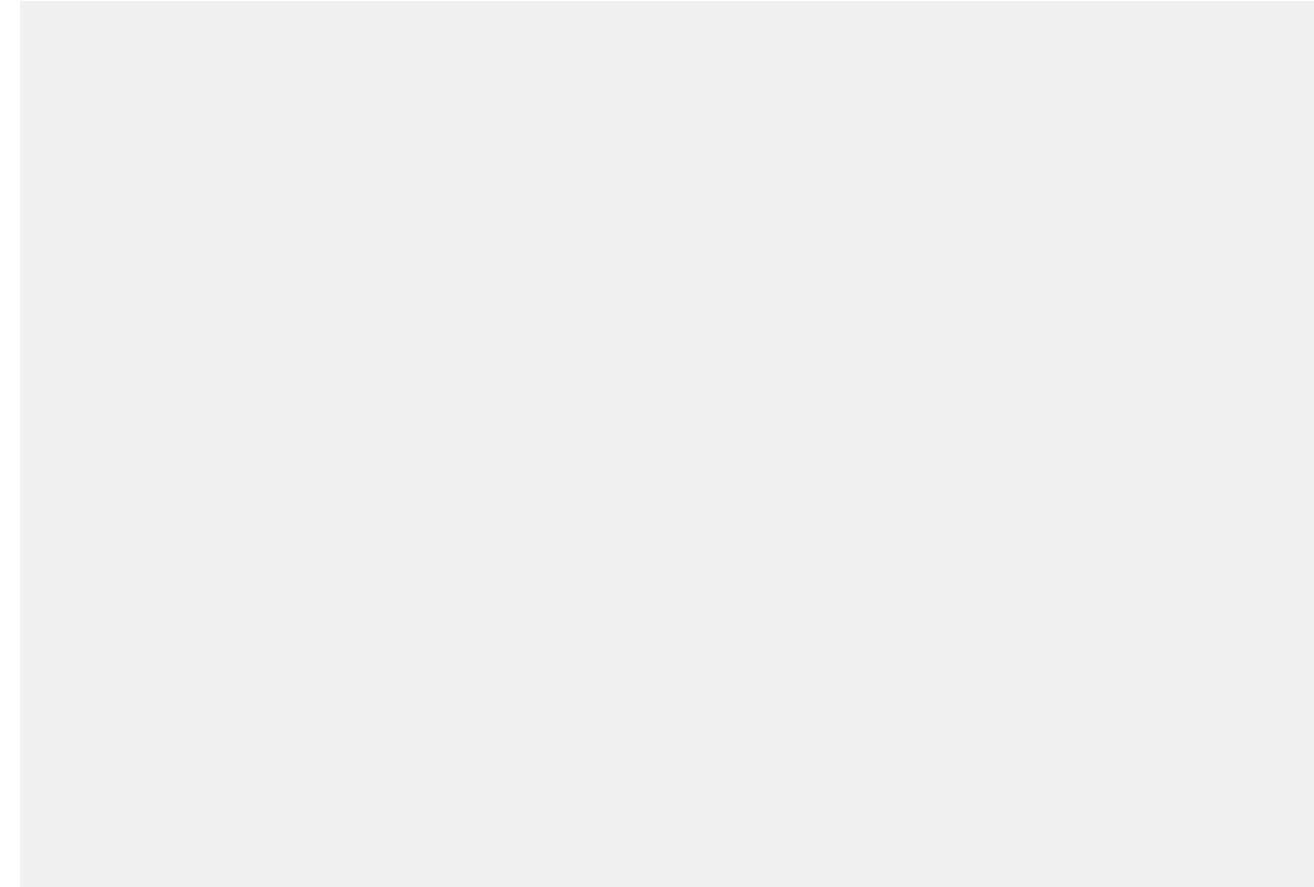
メモ:

- インターフェースを実装するクラスは、global または public にできます。
- このクラスはテストコンテキストでのみ使用されるため、`@Test` のアノテーションを付加できます。この方法で、3 MB の組織コードサイズ制限からクラスを除外できます。

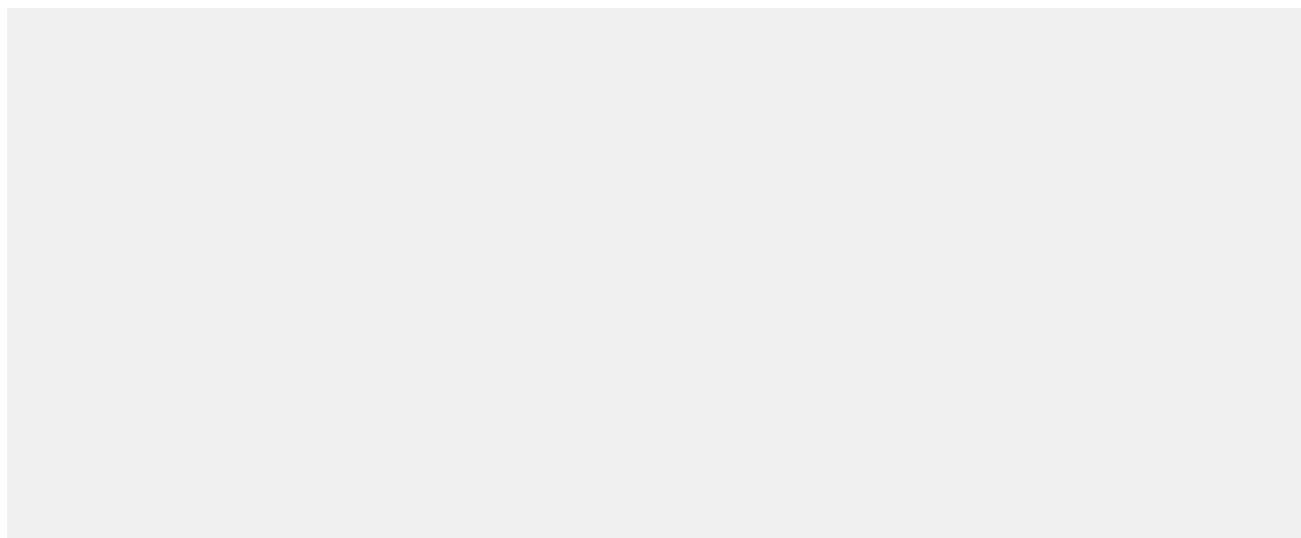
擬似応答の値を指定したら、テストメソッドで

をコールし、この擬似応答を送信するように

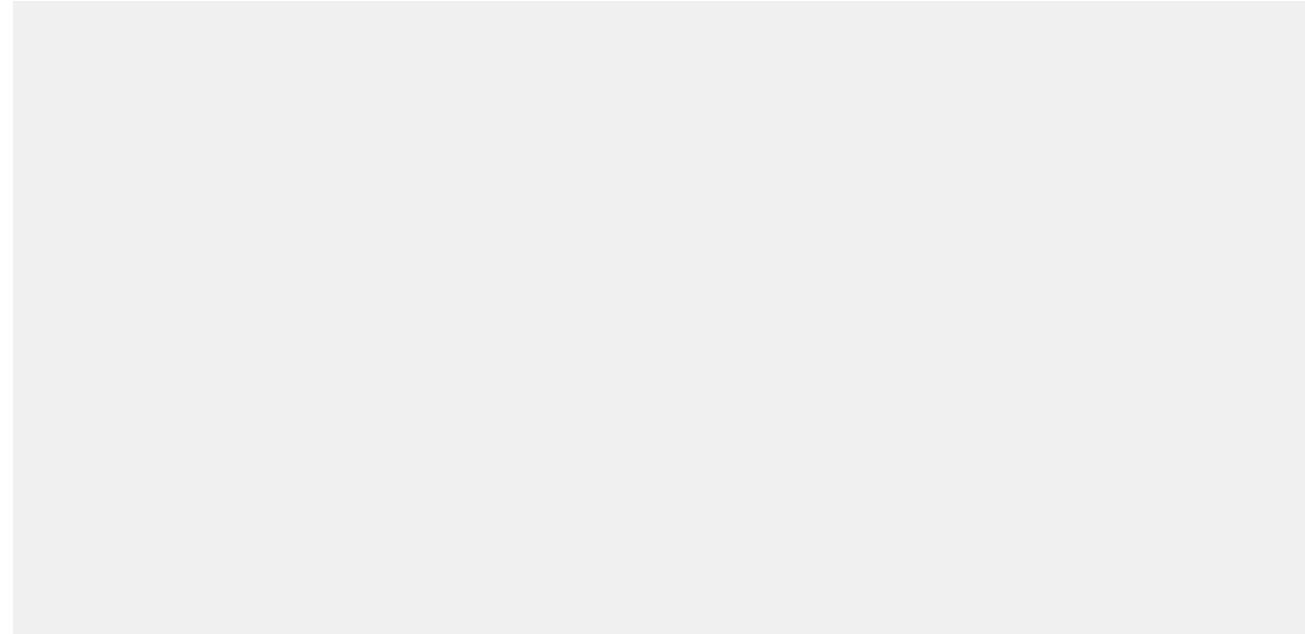
WSDL に基づいています。このクラスを保存する前に、この WSDL をインポートし、  
というクラスを生成しておきます。



次のメソッドでは、Web サービスコールアウトを行います。



次のテストクラスには、擬似コールアウトモードを設定するテストメソッドが含まれます。これは前述のクラスに含まれる メソッドをコールし、擬似応答が受信されたことを確認します。



## 関連リンク

[WebServiceMock インターフェース](#)

## DML 操作と擬似コールアウトの実行

デフォルトでは、同じトランザクション内で DML 操作の後にコールアウトを実行することは許可されません。これは DML 操作によって、コミットされていない待機中の作業が発生してコールアウトの実行が妨げられるためです。場合によっては、コールアウトを行う前に、DML を使用してテストメソッドにテストデータを挿入する必要が生じことがあります。これを行うには、コールアウトを実行するコード部分を

ステートメントの間に配置します。ステートメントは、ステートメントとステートメントの間に配置します。

トメントの前に配置する必要があります。また、DML 操作のコールは、/ ブロックの一部にすることはできません。

擬似コールアウト後の DML 操作は許可されており、テストメソッドでの変更は必要ありません。

### 擬似コールアウト前の DML の実行

この例は、前の例に基づいています。この例では、およびステートメントを使用して、テストメソッドで擬似コールアウトの前に DML 操作を実行できるようにします。テストメソッド( )は最初にテスト取引先を挿入し、をコールします。次に、

を使用して疑似コールアウトモードを設定して、コールアウトを実行するメソッドをコールし、疑似応答値を確認します。最後に、  
をコールします。

```
// Perform some DML to insert test data
Account testAcct = new Account('Test Account');
insert testAcct;

// Call Test.startTest before performing callout
// but after setting test data.

Test.startTest();

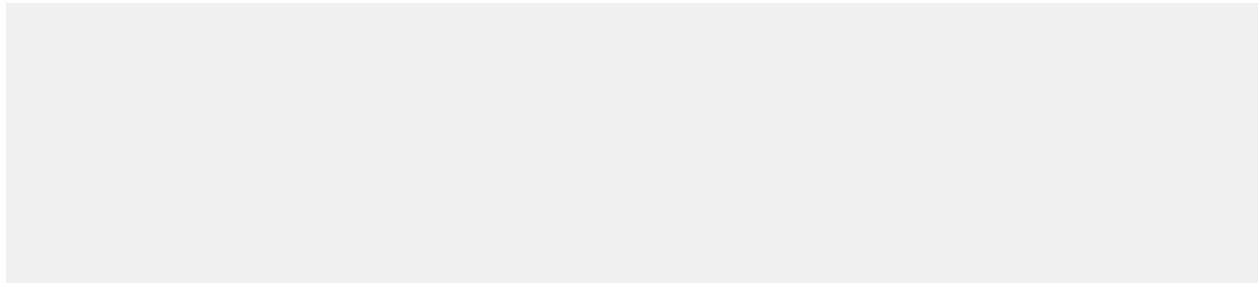
// Set mock callout class
Test.setMock(WebServiceMock.class, new WebServiceMockImpl());

Test.stopTest();
```

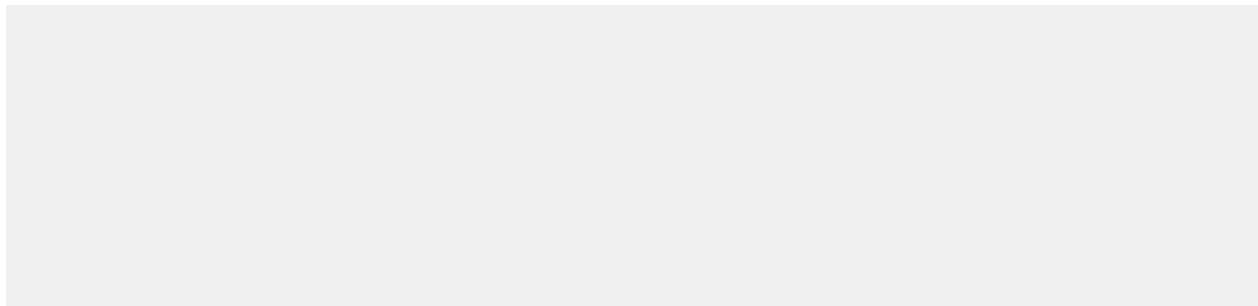
## 非同期 Apex と擬似コールアウト

DML と同様に、非同期 Apex 操作では、コミットされていない待機中の作業によって、同じトランザクションの後の方でコールアウトの実行が妨げられる結果になります。非同期 Apex 操作の例としては、future メソッド、Apex 一括処理、スケジュール済み Apex のコールがあります。通常、これらの非同期コールは、  
の後で実行されるようになります。テストメソッドで **Test.startTest()** と **Test.stopTest()** をステートメント間に配置します。この場合、擬似コールアウトは非同期コールの後で実行できるため、変更は不要です。ただし、非同期コールが **Test.startTest()** と **Test.stopTest()** の間で実行されると、ステートメント間に配置されていない場合は、コミットされない待機中の作業のため例外が発生します。この例外を回避するには、次のいずれかを行います。

- 非同期コールを **と** ステートメント間に配置する。



- DML コールと同じルールに従う。つまり、コールアウトを実行するコード部分を **と** **ス** **ブ** **ス** **テートメント**間に配置します。**ステートメントは、/ ブ** **ロック**の一部にすることはできません。



擬似コールアウト後の非同期コールは許可されており、テストメソッドでの変更は必要ありません。

## WSDL 使用についての考慮事項

WSDL から Apex クラスを生成する場合、次の点に注意してください。

### ヘッダーの対応付け

WSDL ドキュメントで定義されているヘッダーは、生成されたクラスのスタブの公開項目となります。これは、AJAX Toolkit および .NET の動作と似ています。

### ランタイムイベントについて

Apex コードで外部サービスへのコールアウトを作成している場合、次のことを確認します。

- HTTP 要求または Web サービスコールを作成する場合のタイムアウト制限の詳細は、「[コールアウトの制限事項](#)」(ページ 394)を参照してください。
- Apex クラス内の循環参照は許可されていません。
- Salesforce ドメインへの複数のループバック接続は許可されていません。

- エンドポイントにアクセスできるようにするには、エンドポイントを [設定] の [セキュリティ] > [リモートサイトの設定] で登録する必要があります。
- データベース接続を独占しないよう、トランザクションが開始されないようにします。

## 変数名でサポートされていない文字について

WSDL ファイルには、Apex 変数名で使用できない要素名を使用できます。WSDL ファイルから Apex 変数名を生成する場合、次のルールが適用されます。

- 要素名の最初の文字が英字でない場合、生成された Apex 変数名の先頭に文字 \_ が追加されます。
- 要素名の最後の文字が Apex 変数名で使用できない場合、生成された Apex 変数名の最後に文字 \_ が追加されます。
- 要素名に Apex 変数名で使用できない文字が含まれている場合、その文字は ( ) に置き換えられます。
- 要素名に Apex 変数名で使用できない文字が 2 文字続けて含まれている場合、最初の文字はアンダースコア ( ) に置き換えられ、2 番目の文字は \_ に置き換えられます。これにより、Apex で許可されていない連続した 2 つのアンダースコアが変数名に含まれないようにします。
- 2 つのパラメータ、\_ と \_ を使用する操作があるとします。生成される Apex には 2 つの変数があり、いずれも \_ という名前です。クラスはコンパイルしません。手動で Apex を編集し、いずれかの変数名を変更する必要があります。

## WSDL ファイルから生成したクラスのデバッグ

Salesforce では、SOAP API、.NET、および Axis でコードをテストします。別のツールを使用すると問題が発生する場合があります。

デバッグヘッダーを使用して、要求の XML を返し、SOAP メッセージに応答して問題の検出を行います。詳細は、「[Apex の SOAP API および SOAP ヘッダー](#)」(ページ 988) を参照してください。

## HTTP コールアウトの呼び出し

Apex では、HTTP サービスを使用して、GET、POST、PUT、DELETE などの HTTP 要求を作成する組み込みクラスが提供されます。

これらの HTTP クラスを使用して、REST ベースのサービスに統合できます。また、HTTP クラスを SOAP ベースの Web サービスに統合して、WSDL から Apex コードを生成することもできます。WSDL で開始する代わりに HTTP クラスを使用することで、要求と応答の SOAP メッセージの構造をより明確に把握することができます。

詳細およびサンプルは、「[HTTP \(RESTful\) サービスクラス](#)」を参照してください。また、[Force.com Toolkit for Google Data APIs](#) を使用すると、HTTP コールアウトの用途を拡張できます。

## 証明書の使用

Salesforce で生成された、または証明機関 (CA) によって署名された証明書をコールアウトと共に送信することによって、双方向の SSL 認証を使用できます。これにより、コールアウトの送信先が証明書を受信し、キーストアに対する要求の認証にこの証明書を使用できるので、セキュリティが向上します。

コールアウトの双方向 SSL 認証を有効にする手順は、次のとおりです。

1. [証明書を生成します。](#)
2. 証明書とコードを統合します。 「[SOAP サービスでの証明書の使用](#)」および「[HTTP 要求での証明書の使用](#)」を参照してください。
3. サードパーティに接続しており、自己署名証明書を使用している場合は、Salesforce 証明書をそれらと共に共有し、キーストアに証明書を追加できるようにします。組織内で使用される別のアプリケーションに接続している場合、Web サーバまたはアプリケーションサーバでクライアント証明書を要求するように設定します。このプロセスは、使用的 Web サーバまたはアプリケーションサーバの種類によって異なります。Apache Tomcat での双方向の SSL の設定方法の例は、  
[を参照してください。](#)
4. コールアウトの[リモートサイト設定](#)を設定します。Apex コールアウトが外部サイトを呼び出す前に、そのサイトを [リモートサイトの設定] ページで登録する必要があります。登録しない場合、コールアウトが失敗します。

## 証明書の生成

Salesforce で生成された自己署名の証明書、または証明機関 (CA) によって署名された証明書を使用できます。  
コールアウトの証明書を生成する手順は、次のとおりです。

1. [設定] で、[セキュリティのコントロール] > [証明書と鍵の管理] をクリックします。
2. 外部の Web サイトで承認する証明書の種類に基づいて、[自己署名証明書の作成] または [認証機関署名証明書の作成] を選択します。署名つき証明書が作成されたら、証明書の種類は変更できません。
3. Salesforce 証明書の表示ラベルを入力します。この名前は、証明書の表示時、主に管理者によって使用されます。
4. 一意の名前 を入力します。入力した証明書ラベルに基づいて、この名前が自動的に入力されます。この名前は、アンダースコアと英数字のみを含み、組織内で一意の名前にする必要があります。最初は文字であること、スペースは使用しない、最後にアンダースコアを使用しない、2つ続けてアンダースコアを使用しないという制約があります。Force.com Web サービス API または Apex を使用して証明書を参照する場合、一意の名前 を使用します。
5. 付与された証明書と鍵の 鍵サイズ を選択します。セキュリティ上の理由により、デフォルトの鍵のサイズに を指定することをお勧めします。 を選択すると、2048 ビットの鍵を使用した証明書が生成されます。証明書は 2 年間有効です。 を選択すると、1024 ビットの鍵を使用した証明書が生成されます。証明書は 1 年間有効です。



メモ: 正常に Salesforce 証明書を保存した後、キーサイズを変更することはできません。

- CA署名の証明書を作成する場合、次の情報も入力する必要があります。これらの項目が結合され、一意の証明書を生成します。

項目	説明
一般名	署名付き証明書を要求する会社の完全修飾ドメイン名。通常の形式はです。
メールアドレス	証明書に関連付けられているメールアドレス。
会社名	会社の正式名称または登記名。
部署	マーケティングや経理など、証明書を使用する会社の部署。
市区郡	会社のある市区郡。
都道府県	会社のある都道府県。
国コード	会社が所在する国を示す2文字のコード。アメリカの場合、値はです。

- [保存] をクリックします。

正常に Salesforce 証明書を保存した後、証明書と該当する鍵が自動的に生成されます。

認証機関署名証明書を作成した後、使用する前に署名付き証明書をアップロードする必要があります。Salesforce オンラインヘルプの「認証機関 (CA) 署名付き証明書のアップロード」を参照してください。

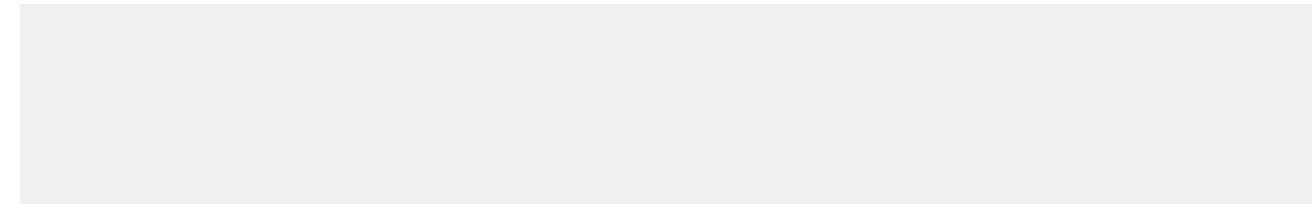
## SOAP サービスでの証明書の使用

Salesforce で証明書を生成した後、証明書を使用して SOAP Web サービスへのコールアウトの双方向認証をサポートできます。

証明書と Apex を統合する手順は、次のとおりです。

- サードパーティから Web サービスの WSDL を受け取るか、接続するアプリケーションから生成します。
- Web サービスの WSDL から Apex クラスを生成します。[「SOAP サービス: WSDL ドキュメントからのクラスの定義」](#)を参照してください。
- 生成される Apex クラスには、WSDL ドキュメントで示される、サードパーティの Web サービスをコールするスタブが含まれています。Apex クラスを編集し、スタブクラスのインスタンスの変数に値を割り当てます。この値は、[設定] の [セキュリティのコントロール] > [証明書と鍵の管理] で生成した証明書の一意の名前と一致する必要があります。

次の例では、[生成されるコードについて](#)のサンプル WSDL ファイルを使用して前の手順の最後のステップを説明します。この例では、[一意の名前](#) で証明書を生成したと想定しています。

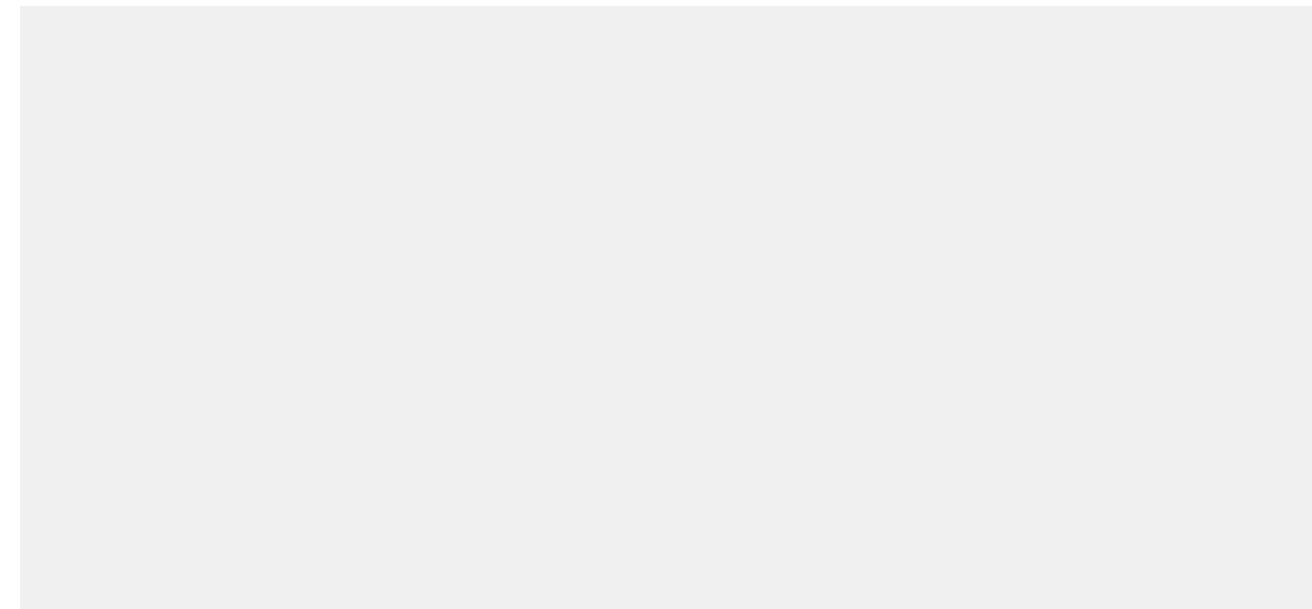


サードパーティから取得した組織用の証明書を使用する従来のプロセスがあります。base64でクライアント証明書の鍵を符号化し、スタブの変数に割り当てます。これは、非公開鍵を保護するセキュリティのベストプラクティスには従わないため、本質的に、Salesforce 証明書を使用する場合よりセキュリティは低くなります。Salesforce 証明書を使用する場合、非公開鍵は Salesforce 外では共有されません。



メモ: [設定] の [開発] > [API] > [クライアント証明書の生成] で生成したクライアント証明書は使用しないでください。従来のプロセスを使用する場合、サードパーティから取得した組織用の証明書を使用する必要があります。

次の例では、[生成されるコードについて](#)(ページ 379)のサンプル WSDL ファイルを使用して、従来のプロセスを説明しています。



## HTTP 要求での証明書の使用

Salesforce で証明書を生成した後、証明書を使用して HTTP 要求へのコールアウトの双向認証をサポートできます。

証明書を Apex と統合する手順は、次のとおりです。

1. 証明書を生成します。証明書の 一意の名前 を確認します。
2. Apex で、 クラスの メソッドを使用します。このメソッドの引数に使用する値は、前のステップで生成された証明書の 一意の名前 に一致する必要があります。

次の例は、前の手順の最後のステップを示します。この例では、 の 一意の名前 で証明書を生成したと想定しています。

## コールアウトの制限事項

Apex コードで、HTTP 要求または Web サービスコールに対するコールアウトを実行する場合に次の制限が適用されます。Web サービスコールには、SOAP API コールまたは外部の Web サービスコールを使用できます。

- 1つの Apex トランザクションで、HTTP 要求または API コールに対するコールアウトを最大 10 回実行できます。
- デフォルトのタイムアウトは 10 秒です。カスタムタイムアウトはコールアウトごとに定義できます。最小値は 1 ミリ秒、最大値は 120 秒です。Web サービスまたは HTTP コールアウトのカスタムタイムアウトの設定方法については、次の例を参照してください。
- 1つの Apex トランザクションによる各コールアウトのタイムアウトの累積値は、最大 120 秒です。累積値とは、特定の Apex トランザクションによって呼び出されたすべてのコールアウトのタイムアウトを合計した値です。
- 同じトランザクション内に待機中の操作が存在する場合はコールアウトを実行できません。操作が待機中となるものには、DML ステートメント、非同期 Apex (future メソッドや Apex 一括処理ジョブなど)、スケジュール済み Apex、メールの送信があります。このような操作を行う前に、コールアウトを実行するようにします。
- 同じトランザクション内で疑似コールアウトよりも前に待機中の操作が発生する可能性があります。「[DML 操作と擬似コールアウトの実行](#)」または「[DML 操作と擬似コールアウトの実行](#)」を参照してください。

### コールアウトタイムアウトの設定

次の例では、Web サービスコールアウトのカスタムタイムアウトを設定します。この例では、サンプルの WSDL ファイルと生成された クラス (「[生成されるコードについて](#)」 (ページ 379) を参照) を使用します。スタブの 変数に値を割り当てることにより、ミリ秒単位でタイムアウト値を設定します。

次に、HTTP コールアウトのカスタムタイムアウトの設定の例を示します。

# 第 14 章

## リファレンス

### トピック:

- DML 操作
- Apex 標準クラスおよび標準メソッド
- Apex クラス
- Apex インターフェース

Apex リファレンスには、Apex 言語についての情報が含まれます。

- [データ操作言語 \(DML\) の操作](#) — データベース内のデータの操作に使用する
- [標準クラスおよびメソッド](#) — プリミティブデータ型、コレクション、sObjects、および Apex の他の部分に使用できる
- [Apex クラス](#) — 使用可能な組み込みクラス
- [Apex インターフェース](#) — ユーザが実装できるインターフェース

さらに、SOAP API のメソッドとオブジェクトを Apex で利用できます。付録セクションの [「Apex の SOAP API および SOAP ヘッダー」](#) (ページ 988) を参照してください。

## DML 操作

DML 操作は、DML ステートメントまたは

リード取引開始の場合、

クラスの

ません。

レコードをマージするには、

DML ステートメントを使用します。

クラスには、merge に相当す

るメソッドはありません。

クラスのメソッドを使用して実行できます。

メソッドを使用します。これに相当する DML はあり

## DML ステートメント

データベース内のデータを挿入、更新、マージ、削除、および復元するには、データ操作言語 (DML) 操作を使用します。

次の Apex DML 操作を使用できます。

- 
- 
- 
- 
- 
- 

### Insert ステートメント

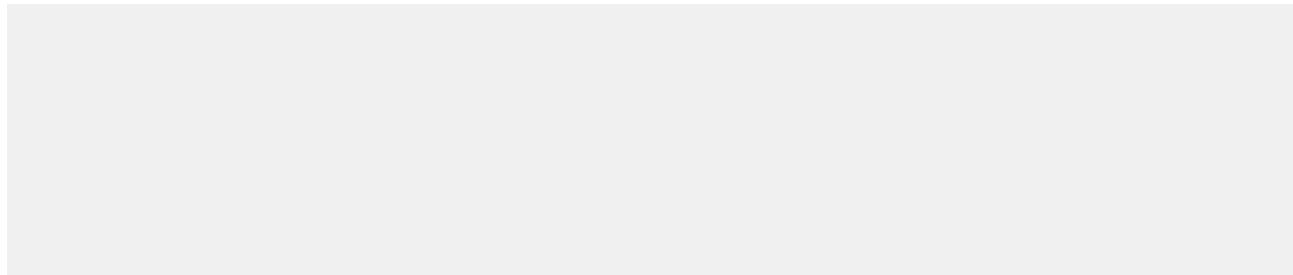
DML 操作は、個別の取引先、取引先責任者など、1つ以上の sObject を組織のデータに追加します。は SQL の INSERT ステートメントに類似しています。

#### 構文

```
sObject  
sObject[]
```

#### 例

次の例では、「Acme」という名前の取引先を挿入しています。





メモ:  
の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

## 関連リンク

[レコードの挿入と更新](#)

## Update ステートメント

DML 操作は、個別の取引先、取引先責任者、請求書の明細などの、組織のデータ内にある 1 つ以上の既存の sObject レコードを更新します。　　は SQL の UPDATE ステートメントに類似しています。

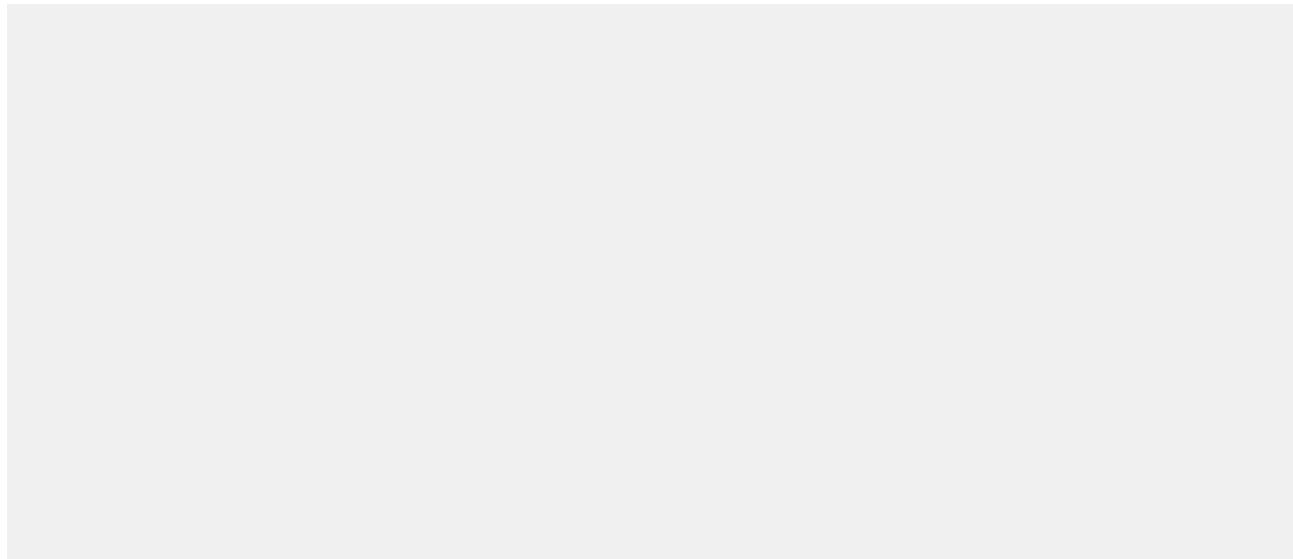
### 構文

*sObject*

*sObject[]*

### 例

次の例では、「Acme」という名前の 1 つの取引先の　　項目を更新しています。



メモ:  
の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

## 関連リンク

[レコードの挿入と更新](#)

## Upsert ステートメント

DML 操作は、既存オブジェクトの有無を確認するために任意のカスタム項目を使用して、1つのステートメント内で新規 sObject レコードを作成したり、既存の sObject レコードを更新したりします。

### 構文

```
sObject opt_external_id  
sObject[] opt_external_id
```

`opt_external_id` は、組織のデータにすでに存在するレコードを照合するために使用されるカスタム項目を指定する、任意の変数です。カスタム項目は、属性が選択された状態で作成する必要があります。また、項目に 属性が選択されていない場合、によって誤って重複レコードが挿入されないように、コンテキストユーザは対象オブジェクトに対するオブジェクトレベルの「すべての参照」権限、または「すべてのデータの参照」権限が必要です。

`opt_external_id` が指定されていない場合、デフォルトで sObject レコードの ID 項目が使用されます。



メモ: カスタム項目に、項目定義の一部として [ユニーク] と 「ABC」と「abc」を値の重複として扱う(大文字と小文字を区別しない) 属性が選択されている場合のみ、カスタム項目による照合では大文字と小文字を区別しません。この場合、「ABC123」は「abc123」と一致します。詳細は、Salesforce オンラインヘルプの「カスタム項目の作成」を参照してください。

### Upsert が Insert と Update を判別する方法

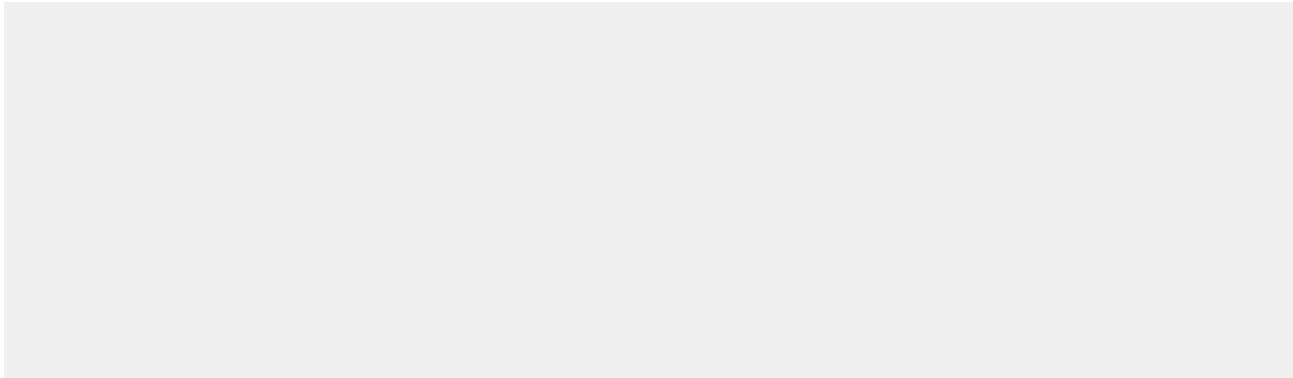
Upsert では、新規オブジェクトレコードを作成するか既存のレコードを更新するか判別するために、sObject レコードの主キー(指定されている場合は外部 ID)を使用します。

- キーが一致しない場合、新規オブジェクトレコードが作成されます。
- キーが一度だけ一致したら、既存のオブジェクトレコードが更新されます。
- キーが複数回一致する場合は、エラーが生成され、オブジェクトレコードは挿入も更新もされません。

sObject レコードが参照項目として設定されている場合、sObject レコードを更新/挿入するために外部キーを使用できます。詳細は、『Object Reference for Salesforce and Force.com』の「データ型」を参照してください。

### 例

この例は、取引先のリストの更新/挿入を実行します。



次の例では、既存の一一致するレコード（ある場合）の外部キーを使用して取引先のリストの更新/挿入を実行します。

## 関連リンク

[レコードの更新/挿入](#)

## Delete ステートメント

DML 操作は、個別の取引先や取引先責任者など、1つ以上の既存の sObject レコードを組織のデータから削除します。 は、SOAP API の に類似しています。

### 構文

*sObject* *ID*

### 例

次の例では、「DotCom」という名前のすべての取引先を削除しています。



メモ:  
い。

の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

## 関連リンク

[レコードの削除](#)

## Undelete ステートメント

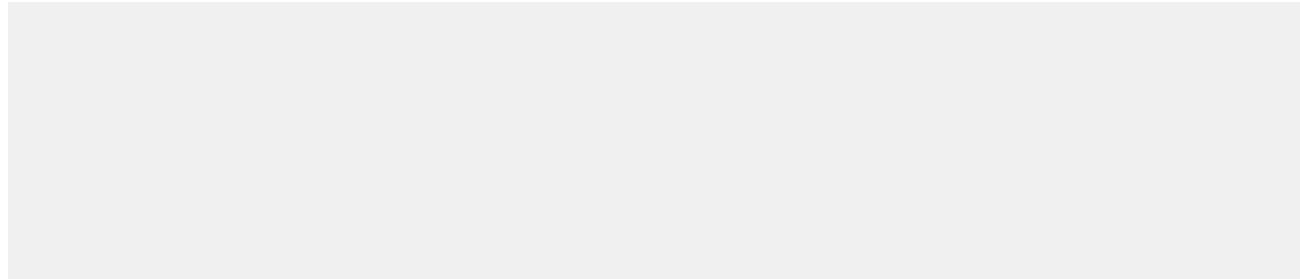
DML 操作は、個別の取引先や取引先責任者など、1 つ以上の既存の sObject レコードを組織のごみ箱から復元します。      は SQL の UNDELETE ステートメントに類似しています。

### 構文

```
sObject      Record ID  
sObject[]     ID[]
```

### 例

次の例では、「Trump」という名前の取引先を復元しています。      キーワードは、削除されたレコードやアーカイブ済みの活動を含め、最上位リレーションと集計リレーションの両方にあるすべての行をクエリします。



メモ:  
い。

の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

## 関連リンク

[削除したレコードの復元](#)

## Merge ステートメント

ステートメントは、同じ sObject データ型の最大 3 つのレコードを 1 つのレコードにマージし、他のレコードを削除してから、関連レコードを再ペアレンタ化します。



メモ: この DML 操作には、一致するデータベースシステムメソッドはありません。

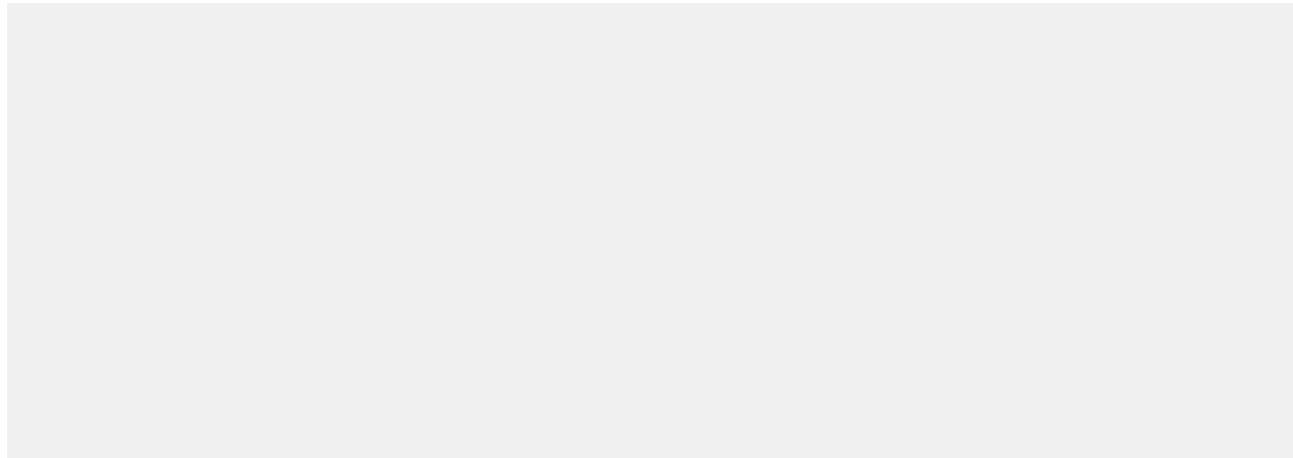
## 構文

```
sObject sObject  
sObject sObject[]  
sObject ID  
sObject ID[]
```

最初のパラメータは、他のレコードがマージされる主レコードを表します。2番目のパラメータは、マージされてから削除される1つ以上の他のレコードを表します。これらのその他のレコードは、1つの sObject レコードまたは ID、または2つの sObject レコードまたは ID のリストとして、ステートメントに渡すことができます。

## 例

次の例では、「Acme Inc.」と「Acme」という名前の2つの取引先を1つのレコードにマージしています。



メモ: の処理についての詳細は、「[一括 DML 例外処理](#)」(ページ 401)を参照してください。

## 関連リンク

[レコードのマージ](#)

## 一括 DML 例外処理

一括 DML コールによって発生する例外(コールの直接的な結果によって実行されるトリガ内の再帰的 DML 操作を含む)は、コールの発生元ごとに異なる処理がされます。

- Apex DML ステートメントから直接発生した一括 DML コールが原因でエラーが発生した場合、または、データベース DML メソッドのパラメータが `True` として指定されている場合、ランタイムエンジンは「オールオアナッシング」ルールに従います。つまり、1回の操作の間、すべてのレコードを正常に更新するか、または操作全体を DML ステートメントのすぐ前の時点にロールバックする必要があります。
- SOAP API から発生した一括 DML コールが原因でエラーが発生した場合、ランタイムエンジンは少なくとも部分的な保存を試みます。
  - 最初の試行で、ランタイムエンジンはすべてのレコードを処理します。入力規則や独自のインデックス違反などの問題によるエラーを生成したレコードは、除外されます。
  - 最初の試行にエラーがあった場合、ランタイムエンジンは、エラーを生成しなかったレコードのみを含む2回目の試行を行います。最初の試行でエラーを生成しなかったすべてのレコードが処理され、競合の条件などが理由でエラーを生成したレコードがあれば、それも除外されます。
  - 2回目の試行中に追加工エラーがあった場合、ランタイムエンジンは、初回と2回目にエラーを生成しなかったレコードのみを含む3回目と最後の試行を行います。エラーを生成したレコードがある場合、操作全体は失敗し、エラーメッセージ「Too many batch retries in the presence of Apex triggers and partial failures (Apex トリガと部分的な失敗がある場合にバッチ試行の回数が多すぎます)」が表示されます。



メモ: 2回目と3回目の試行中、ガバナ制限は、最初の試行前の元の状態にリセットされます。「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

## データベースメソッド

クラスには、DML 操作を実行するメソッドが含まれており、それらの大部分は DML ステートメントに対応するメソッドです。このクラスには、SOQL クエリと Apex の一括処理に関連するメソッドも含まれています。

### Database クラス

次に、データベースのシステムの静的メソッドを示します。

名前	引数	戻り値	説明
<code>LeadConvert</code>	<code>LeadConvert leadToConvert,</code> <code>Boolean opt_allOrNone</code>	<code>Database. LeadConvertResult</code>	リード取引開始によって取引先、取引先責任者、および(必要に応じて)商談を作成します。 (省略可能) <code>opt_allOrNone</code> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <code>True</code> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。

名前	引数	戻り値	説明
			メソッドは、最大 100 の オブジェクトを受け入れます。
		実行された各 ステートメントのガバナ制限にカウントされ ます。	メソッドは、DML
			LeadConvert オブジェクトのリストを、取引 先、取引先責任者、および(必要に応じて)商 勧遂

名前	引数	戻り値	説明
			<p>(省略可能) <code>opt_allOrNone</code> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <code>True</code> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>実行された各 <code>delete</code> メソッドは、DML ステートメントのガバナ制限にカウントされます。</p>
	<code>SObject[] recordsToDelete</code> <code>Boolean opt_allOrNone</code>	<code>Database.DeleteResult[]</code>	<p>個別の取引先や取引先責任者など、既存の <code>sObject</code> レコードのリストを組織のデータから削除します。 <code>delete</code> は、SOAP API のステートメントに類似しています。</p> <p>(省略可能) <code>opt_allOrNone</code> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <code>True</code> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>実行された各 <code>delete</code> メソッドは、DML ステートメントのガバナ制限にカウントされます。</p> <p>例:</p> <p>次の例では、「DotCom」という名前の取引先を削除しています。</p>

名前	引数	戻り値	説明
	RecordID <i>ID</i> Boolean <i>opt_allOrNone</i>	Database. DeleteResult	個別の取引先や取引先責任者など、既存の sObject レコードを組織のデータから削除します。 は、SOAP API のステートメントに類似しています。  (省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。  実行された各 メソッドは、DML ステートメントのガバナ制限にカウントされます。
	RecordIDs [] <i>ID</i> Boolean <i>opt_allOrNone</i>	Database. DeleteResult[]	個別の取引先や取引先責任者など、既存の sObject レコードのリストを組織のデータから削除します。 は、SOAP API のステートメントに類似しています。  (省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。  実行された各 メソッドは、DML ステートメントのガバナ制限にカウントされます。
	RecordIds [] <i>ID</i>	Database. EmptyRecycleBin Result[]	指定したレコードがごみ箱から完全に削除されます。次の点に注意してください。 <ul style="list-style-type: none"><li>このメソッドを使用してレコードが削除されると、復元することはできません。</li><li>削除対象として指定できるのは 10,000 件のレコードのみです。</li><li>ログインユーザは、自身のごみ箱にあるレコード、または、下位のごみ箱にあるレコードの中でクエリ可能なものはすべて削除されます。</li></ul>

名前	引数	戻り値	説明
			<p>除できます。ログインユーザが「すべてのデータの編集」権限を持っている場合、組織内のすべてのごみ箱のレコードに対するクエリと削除を実行できます。</p> <ul style="list-style-type: none"> <li>カスケード削除レコード ID は ID のリストに含めないでください。リストに含めるとエラーが発生します。たとえば、取引先レコードが削除されると、関連するすべての取引先責任者、商談、契約なども削除されます。最上位の取引先の ID のみを含めるようにしてください。関連するレコードはすべて自動的に削除されます。</li> <li>DML ステートメントによって処理された項目の数に、削除された項目が追加され、発行された DML ステートメントの合計数にメソッドコールが追加されます。実行された各 メソッドは、DML ステートメントのガバナ制限にカウントされます。</li> </ul>
<code>sObject sObject</code>	<code>Database.EmptyRecycleBinResult</code>	<code>Database.EmptyRecycleBinResult</code>	<p>指定した sObject がごみ箱から完全に削除されます。次の点に注意してください。</p> <ul style="list-style-type: none"> <li>このメソッドを使用して sObject が削除されると、復元することはできません。</li> <li>削除対象として指定できるのは 10,000 件の sObject のみです。</li> <li>ログインユーザは、自身のごみ箱または下位のごみ箱にある sObject の中でクエリ可能なものはすべて削除できます。ログインユーザが「すべてのデータの編集」権限を持っている場合、組織内のすべてのごみ箱の sObject に対するクエリと削除を実行できます。</li> <li>カスケード削除によって削除された sObject は含めないでください。含めるとエラーが発生します。たとえば、取引先が削除されると、関連するすべての取引先責任者、商談、契約なども削除されます。最上位の取引先の sObject のみを含めるようにしてください。関連する sObject はすべて自動的に削除されます。</li> </ul>

名前	引数	戻り値	説明
	sObjects [] list<SObject> Database. EmptyRecycleBin Result[]		<ul style="list-style-type: none"><li>DML ステートメントによって処理された項目の数に、削除された項目が追加され、発行された DML ステートメントの合計数にメソッドコールが追加されます。実行された各メソッドは、DML ステートメントのガバナ制限にカウントされます。</li><li>指定した sObject がごみ箱から完全に削除されます。次の点に注意してください。<ul style="list-style-type: none"><li>このメソッドを使用して sObject が削除されると、復元することはできません。</li><li>削除対象として指定できるのは 10,000 件の sObject のみです。</li><li>ログインユーザは、自身のごみ箱または下位のごみ箱にある sObject の中でクエリ可能なものはすべて削除できます。ログインユーザが「すべてのデータの編集」権限を持っている場合、組織内のすべてのごみ箱の sObject に対するクエリと削除を実行できます。</li><li>カスケード削除によって削除された sObject は含めないでください。含めるとエラーが発生します。たとえば、取引先が削除されると、関連するすべての取引先責任者、商談、契約なども削除されます。最上位の取引先の sObject のみを含めるようにしてください。関連する sObject はすべて自動的に削除されます。</li><li>DML ステートメントによって処理された項目の数に、削除された項目が追加され、発行された DML ステートメントの合計数にメソッドコールが追加されます。実行された各メソッドは、DML ステートメントのガバナ制限にカウントされます。</li></ul></li></ul>

名前	引数	戻り値	説明
	sObject <i>className</i>	ID	<p>指定したクラスを Apex の一括処理ジョブとして実行します。 詳細は、「<a href="#">Apexの一括処理の使用</a>」(ページ 273)を参照してください。</p> <p> メモ: メソッドによってコールされたクラスはメソッドを実装します。</p>
	sObject <i>className</i> , integer <i>scope</i>	ID	<p>指定したクラスを Apex の一括処理ジョブとして実行します。 <i>scope</i> の値は 0 より大きくなります。 詳細は、「<a href="#">Apexの一括処理の使用</a>」(ページ 273)を参照してください。</p> <p> メモ: メソッドによってコールされたクラスはメソッドを実装します。</p>
	sObject [] <i>listOfQueries</i>	Database.QueryLocator	<p>Apex または Visualforce の一括処理で使用される QueryLocator オブジェクトを作成します。 詳細は、「<a href="#">Database.QueryLocator クラス</a>」、「<a href="#">Apex による共有管理について</a>」、および「<a href="#">クラス</a>」を参照してください。</p> <p><a href="#">集計関数</a>が含まれるクエリでを使用することはできません。</p> <p>実行された各メソッドは、取得されるレコードの合計数と発行される SOQL クエリの合計数である 10,000 のガバナ制限にカウントされます。</p>
	String <i>query</i>	Database.QueryLocator	<p>Apex または Visualforce の一括処理で使用される QueryLocator オブジェクトを作成します。 詳細は、「<a href="#">Database.QueryLocator クラス</a>」、「<a href="#">Apex による共有管理について</a>」、および「<a href="#">クラス</a>」を参照してください。</p> <p><a href="#">集計関数</a>が含まれるクエリでを使用することはできません。</p>

名前	引数	戻り値	説明
			実行された各 メソッドは、取得されるレコードの合計数と発行される SOQL クエリの合計数である 10,000 のガバナ制限にカウントされます。
	sObject <i>recordToInsert</i> boolean <i>opt_allOrNone</i>   database.DMLOptions <i>opt_DMLOptions</i>	Database.SaveResult	個別の取引先や取引先責任者など、sObject を組織のデータに追加します。 は SQL の INSERT ステートメントに類似しています。  (省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。  (省略可能) <i>opt_DMLOptions</i> パラメータは、レコード挿入時にエラーが発生した場合の割り当てルールの情報やロールバック動作など、トランザクションの追加データを指定します。  項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存 (コンパイル) した Apex クラスとトリガにはランタイムエラーが発生します。  実行された各 メソッドは、DML ステートメントのガバナ制限にカウントされます。

名前	引数	戻り値	説明
			<p>(省略可能) <code>opt_DMLOptions</code> パラメータは、レコード挿入時にエラーが発生した場合の割り当てルールの情報やロールバック動作など、トランザクションの追加データを指定します。</p> <p>項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存 (コンパイル) した Apex クラスとトリガにはランタイムエラーが発生します。</p> <p>実行された各 メソッドは、DML ステートメントのガバナ制限にカウントされます。</p> <p>例:</p> <p>次の例では 2 つの取引先が挿入されます。</p>
	<code>String query</code>	<code>sObject[]</code>	<p>実行時に動的 SOQL クエリを作成します。このメソッドは、通常の割り当てステートメントや ループなど、静的 SOQL クエリが使用可能な場合に使用できます。インライン SOQL 項目の場合とは異なり、バインド変数はサポートされていません。</p> <p>詳細は、「<a href="#">動的 SOQL</a>」(ページ 265)を参照してください。</p> <p>実行された各 メソッドは、SOQL クエリのガバナ制限にカウントされます。</p>
	<code>System.Savepoint sp</code>	<code>Void</code>	データベースを、 <code>savepoint</code> 変数で指定された状態に復元します。最後の <code>savepoint</code> 後に送信されたメールもロールバックされ、送信されません。

名前	引数	戻り値	説明
			<p>次の点に注意してください。</p> <ul style="list-style-type: none"> <li>ロールバック中、静的変数は戻されません。トリガの実行を再試行する場合、静的変数には最初の実行から得た値が維持されます。</li> <li>各ロールバックは、DML ステートメントのガバナ制限にカウントされます。データベースをそれ以上の回数ロールバックしようとすると、ランタイムエラーが発生します。</li> <li>savepoint の設定後に挿入された sObject の ID は、ロールバック後にクリアされません。ロールバック後に挿入するには、新しい sObject を作成します。ロールバック前に作成した変数を使用して sObject を挿入しようとすると、その sObject 変数には ID があるため失敗します。同じ変数を使用して sObject を更新または更新/挿入しようとした場合も、sObject はデータベース内に存在せず、更新できないため失敗します。</li> </ul> <p>例は、「<a href="#">トランザクションの制御</a>」を参照してください。</p>
	System.Savepoint		<p>ローカル変数として保存でき、メソッドで使用してデータベースをその時点に復元できる savepoint 変数を返します。</p> <p>次の点に注意してください。</p> <ul style="list-style-type: none"> <li>複数の savepoint を設定し、生成した最新 savepoint ではない savepoint にロールバックすると、ロールバックされた savepoint 変数は無効になります。たとえば、最初に savepoint を生成し、次に savepoint を生成した場合、にロールバックすると、変数 は無効になります。その変数を使用しようとすると、ランタイムエラーが発生します。</li> <li>各トリガ呼び出しが新しい実行コンテキストであるため、savepoints への参照は、トリガ呼び出しを通過することはできません。静的変数として savepoint を宣言し、</li> </ul>

名前	引数	戻り値	説明
			<p>トリガコンテキスト全体で使用しようとすると、ランタイムエラーが発生します。</p> <ul style="list-style-type: none"> <li>設定した各セーブポイントは、DML ステートメントのガバナ制限にカウントされます。</li> </ul> <p><a href="#">「トランザクションの制御」</a> の例を参照してください。</p>
	sObject <i>recordToUndelete</i> Boolean <i>opt_allOrNone</i>	Database. UndeleteResult	<p>個別の取引先や取引先責任者など、既存の sObject レコードを組織のごみ箱から復元します。 <a href="#">UNDELETE</a> は SQL の UNDELETE ステートメントに類似しています。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <a href="#">true</a> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>実行された各 <a href="#">Undelete</a> メソッドは、DML ステートメントのガバナ制限にカウントされます。</p>
	sObject [] <i>recordsToUndelete</i> Boolean <i>opt_allOrNone</i>	Database. UndeleteResult[]	<p>個別の取引先や取引先責任者など、1つ以上の既存の sObject レコードを組織のごみ箱から復元します。 <a href="#">UNDELETE</a> は SQL の UNDELETE ステートメントに類似しています。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <a href="#">true</a> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>実行された各 <a href="#">Undelete</a> メソッドは、DML ステートメントのガバナ制限にカウントされます。</p>

名前	引数	戻り値	説明
			<p>例:</p> <p>次の例では、「Trump」という名前のすべての取引先が復元されます。キーワードは、削除されたレコードやアーカイブ済みの活動を含め、最上位リレーションと集計リレーションの両方にあるすべての行をクエリします。</p>
	RecordID <i>ID</i> Boolean <i>opt_allOrNone</i>	Database. <a href="#">UndeleteResult</a>	<p>個別の取引先や取引先責任者など、既存の sObject レコードを組織のごみ箱から復元します。は SQL の UNDELETE ステートメントに類似しています。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>実行された各 メソッドは、DML ステートメントのガバナ制限にカウントされます。</p>
	RecordIDs [] <i>ID</i> Boolean <i>opt_allOrNone</i>	Database. <a href="#">UndeleteResult[]</a>	<p>個別の取引先や取引先責任者など、1つ以上の既存の sObject レコードを組織のごみ箱から復元します。は SQL の UNDELETE ステートメントに類似しています。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定</p>

名前	引数	戻り値	説明
	sObject <i>recordToUpdate</i> boolean <i>opt_allOrNone</i>   Database.DMLOptions <i>opt_DMLOptions</i>	Database.SaveResult	<p>します。このパラメータを <code>True</code> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>実行された各 <code>Update</code> メソッドは、DML ステートメントのガバナ制限にカウントされます。</p> <p><code>sObject recordToUpdate</code> パラメータは、個別の取引先または取引先責任者など、組織のデータ内の既存の <code>sObject</code> レコードを更新します。これは SQL の UPDATE ステートメントに類似しています。</p> <p>(省略可能) <code>opt_allOrNone</code> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <code>True</code> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>(省略可能) <code>opt_DMLOptions</code> パラメータは、レコード挿入時にエラーが発生した場合の割り当てルールの情報やロールバック動作など、トランザクションの追加データを指定します。</p> <p>項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存 (コンパイル) した Apex クラスとトリガにはランタイムエラーが発生します。</p> <p>実行された各 <code>Update</code> メソッドは、DML ステートメントのガバナ制限にカウントされます。</p> <p>例:</p> <p>次の例では、1つの取引先の <code>Account</code> 項目が更新されます。</p>

名前	引数	戻り値	説明
	sObject [] recordsToUpdate Boolean opt_allOrNone   database.DMLOptions <i>opt_DMLOptions</i>	Database.SaveResult []	<p>個別の取引先や取引先責任者、請求書の明細など、組織のデータ内の 1 つ以上の既存の sObject レコードを更新します。 は SQL の UPDATE ステートメントに類似しています。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>(省略可能) <i>opt_DMLOptions</i> パラメータは、レコード挿入時にエラーが発生した場合の割り当てルールの情報やロールバック動作など、トランザクションの追加データを指定します。</p> <p>項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存 (コンパイル) した Apex クラスとトリガにはランタイムエラーが発生します。</p> <p>実行された各 メソッドは、DML ステートメントのガバナ制限にカウントされます。</p>

名前	引数	戻り値	説明
	sObject <i>recordToUpsert</i> Schema.SObjectField <i>External_ID_Field</i> Boolean <i>opt_allOrNone</i>	Database.UpsertResult	<p>既存オブジェクトの有無を確認するために任意のカスタム項目を使用して、1つのステートメント内で新規 sObject レコードを作成したり、既存の sObject レコードを更新したりします。</p> <p><i>External_ID_Field</i> は Schema.SObjectField のデータ型、つまり項目トークンです。</p> <p>特殊メソッドを使用して、項目のトークンを検索します。たとえば、</p> <p style="padding-left: 2em;">です。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <code>True</code> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。</p> <p>項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存 (コンパイル) した Apex クラスとトリガにはランタイムエラーが発生します。</p> <p>実行された各 <code>Upsert</code> メソッドは、DML ステートメントのガバナ制限にカウントされます。</p>
	sObject [] <i>recordsToUpsert</i> Schema.SObjectField <i>External_ID_Field</i> Boolean <i>opt_allOrNone</i>	Database.UpsertResult []	<p>既存オブジェクトの有無を確認する任意のカスタム項目を使用します。</p> <p><i>External_ID_Field</i> は Schema.SObjectField のデータ型、つまり項目トークンです。</p> <p>特殊メソッドを使用して、項目のトークンを検索します。たとえば、</p> <p style="padding-left: 2em;">です。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータは、操作で部分的な完了を許可するかどうかを指定します。このパラメータを <code>True</code> に設定した場合、レコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、</p>

名前	引数	戻り値	説明
			どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。
			項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存(コンパイル)した Apex クラスとトリガにはランタイムエラーが発生します。
			実行された各メソッドは、DML ステートメントのガバナ制限にカウントされます。

## 関連リンク

[DML ステートメント](#)  
[実行ガバナと制限について](#)

## Database.LeadConvert クラス

取引の開始に使用する情報が含まれます。

### 使用方法

データベースメソッドは、リード取引開始によって取引先と取引先責任者、および(必要に応じて)商談を作成します。 は、 クラスのインスタンスをパラメータとして取ります。このクラスのインスタンスを作成し、リードとその変換先の取引先と取引先責任者の設定など、取引の開始に必要な情報を設定します。

### メソッド

次に、 クラスのインスタンスマソッドを示します。

名前	引数	戻り値	説明
	ID		リードをマージする取引先の ID を取得します。
	ID		リードをマージする取引先責任者の ID を取得します。
	String		取引開始済みのリードのリード状況の値を取得します。
	ID		取引を開始するリードの ID を取得します。
	String		作成する商談の名前を取得します。
	ID		新しく作成する取引先、取引先責任者、商談の所有者となるユーザの ID を取得します。

名前	引数	戻り値	説明
	Boolean		リード変換時に商談を作成するかどうかを指定します (デフォルトは <code>true</code> で商談を作成し、 <code>false</code> では作成しない)。
	Boolean		変換先の取引先責任者オブジェクトの <code>OpportunityLineItem</code> 項目で、変換元のリードオブジェクトの <code>Opportunity</code> 項目の値を上書きするかどうかを指定します (上書きする場合は <code>true</code> 、上書きしない場合は <code>false</code> で、デフォルトでは上書きしません)。
	Boolean		で指定された所有者に通知メールを送るかどうかを示します (送信する場合は <code>true</code> 、送信しない場合は <code>false</code> )。デフォルトは <code>false</code> 。
<code>ID leadID</code>	Void		リードをマージする取引先の ID を設定します。この値は、個人取引先を含めた既存の取引先を更新する場合のみ必要です。それ以外の場合は、 <code>Opportunity</code> が指定された場合、新しい取引先が作成されます。
<code>ID opportunityOwnerID</code>	Void		リードがマージされる取引先責任者の ID を設定します (この取引先責任者は、 <code>Opportunity</code> で指定された取引先と関連付けられている必要があります)。この値は、既存の取引先責任者を更新する場合のみ必要です。
		 重要: リードを個人取引先に変換する場合、 <code>Opportunity</code> を指定しないでください。指定するとエラーが発生します。個人取引先ののみを指定してください。	
			が指定された場合、アプリケーションは、取引先と暗黙的に関連付けられる新しい取引先責任者を作成します。取引先責任者および他の既存のデータは上書きされません (ただし、 <code>Opportunity</code> が <code>true</code> に設定されている場合は <code>Opportunity</code> 項目のみが上書きされます)。
<code>String status</code>	Void		変換されたリードのリード状況の値を設定します。この項目は必須です。
<code>Boolean CreateOpportunity</code>	Void		リード変換時に商談を作成するかどうかを指定します。デフォルト値は <code>false</code> です。デフォルトでは、商談が作成されます。リードの商談を作成したくない場合のみ、このフラグを <code>true</code> に設定します。
<code>ID leadID</code>	Void		変換するリードの ID を設定します。この項目は必須です。
<code>String oppName</code>	Void		作成する商談の名前を設定します。名前が指定されない場合、デフォルト値はリードの会社名となります。この

名前	引数	戻り値	説明
			項目の文字数は 80 文字までです。 が true の場合、商談は作成されません。また、この項目は空のままにする必要があります。空でない場合はエラーが発生します。
	Boolean <i>OverwriteLeadSource</i>	Void	変換先の取引先責任者オブジェクトの に、変換元のリードオブジェクトの 値を上書きするかどうかを指定します。デフォルト値は false で、項目の値を上書きしません。True として指定し た場合は、変換先取引先責任者の も指定 する必要があります。
ID <i>ID</i>		Void	新しく作成する取引先、取引先責任者、商談の所有者と なるユーザの ID を指定します。アプリケーションでこ の値を指定しない場合、リードの所有者が新しいオブ ジェクトの所有者となります。このメソッドは、既存の オブジェクトにマージする場合は適用されません。 が指定された場合、既存の取引先または取 引先責任者の 項目は上書きされません。
	Boolean <i>SendEmail</i>	Void	で指定された所有者に通知メールを送るか どうかを指定します。デフォルト値は false で、メールを 送信しません。

## 例

この例では、  
を挿入し、  
メソッドを使用してリード取引を開始する方法を示します。新規リード  
オブジェクトを作成してその状況を取引開始済みに設定し、  
メソッドに渡します。最後に、取引の開始が成功したことを確認します。

## 関連リンク

[取引の開始](#)

## Database.LeadConvertResult クラス

リード取引開始の結果。

### 使用方法

LeadConvertResult オブジェクトの配列は、データベースメソッドで返されます。LeadConvertResult 配列の各要素は、データベースメソッドの `sObject[]` パラメータとして渡された `sObject` 配列に対応します。つまり、LeadConvertResult 配列の最初の要素は、`sObject` 配列の最初の要素と一致します。また、LeadConvertResult 配列の 2 番目の要素は `sObject` 配列の 2 番目の要素と一致し、3 番目以降も同様です。`sObject` が 1 つのみ渡される場合、LeadConvertResult 配列には 1 つの要素が含まれます。

### メソッド

次に、クラスのインスタンスマソッドを示します。

メソッド	戻り値	説明
	ID	新しい取引先の ID (新しい取引先が指定されている場合)。が呼び出された場合は、指定された取引先の ID。
	ID	新しい取引先責任者の ID (新しい取引先責任者が指定されている場合)。が呼び出された場合は、指定された取引先責任者の ID。
	<code>Database.Error[]</code>	エラーが発生した場合、エラーコードと説明を示す 1 つ以上のデータベースエラー オブジェクトからなる配列。詳細は、「 <a href="#">Database.Error クラス</a> 」(ページ 426)を参照してください。
	ID	変換されたリードの ID。
	ID	新しい商談の ID (が呼び出されたときに新規に作成された場合)。
	Boolean	このオブジェクトに対する DML 操作が成功した場合、Boolean 値は <code>true</code> に設定されます。それ以外の場合は <code>false</code> です。

## Database.SaveResult クラス

Database メソッドによって返される insert または update DML 操作の結果。

### 使用方法

SaveResult オブジェクトの配列は、データベースメソッドとデータベースメソッドで返されます。SaveResult 配列の各要素は、データベースメソッドの `sObject[]` パラメータとして渡された `sObject` 配列に対応します。つまり、SaveResult 配列の最初の要素は、`sObject` 配列の最初の要素と一致します。また、SaveResult 配列の 2 番目の要素は `sObject` 配列の 2 番目の要素と一致し、3 番目以降も同様です。`sObject` が 1 つのみ渡される場合、SaveResult 配列には 1 つの要素が含まれます。

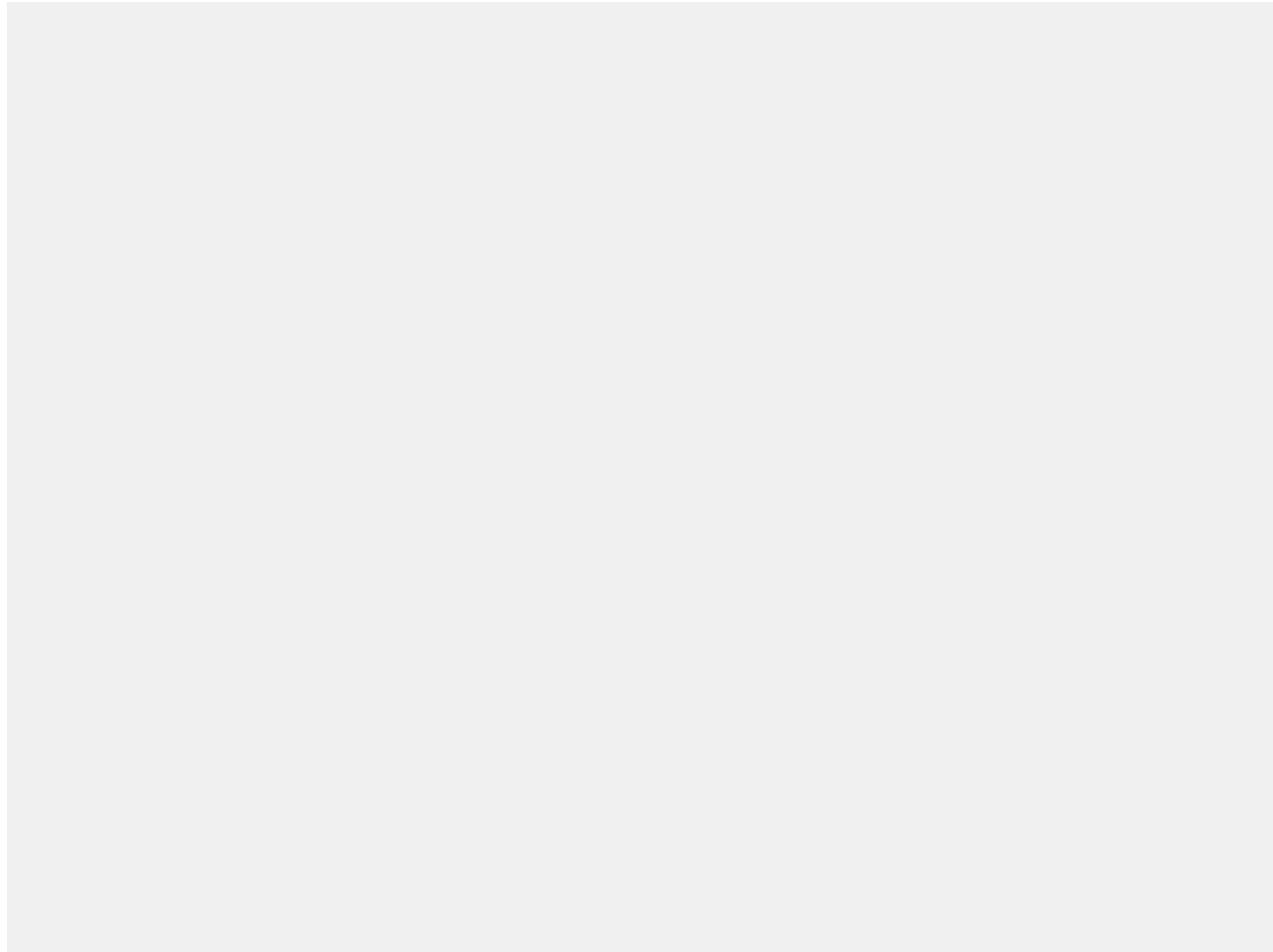
### メソッド

次に、クラスのインスタンスマソッドを示します。

名前	型	説明
	<code>Database.Error[]</code>	エラーが発生した場合、エラーコードと説明を示す 1 つ以上のデータベースエラー オブジェクトからなる配列。詳細は、「 <a href="#">Database.Error クラス</a> 」(ページ 426)を参照してください。
	ID	挿入または更新しようとしている <code>sObject</code> の ID。この項目に値が入力されている場合、オブジェクトは正常に挿入または更新されています。この項目が空白の場合、そのオブジェクトに対する操作は失敗しています。
	Boolean	このオブジェクトに対する DML 操作が成功した場合、Boolean 値は <code>true</code> に設定されます。それ以外の場合は <code>false</code> です。

### 例

次の例では、返されたオブジェクトを介して取得および反復処理する方法を示します。これは、`Object` の 2 番目のパラメータに `false` を指定して使用し、2 つの取引先を挿入して、失敗時にレコードの部分的な処理を行えるようにしています。取引先の 1 つで必須の [名称] 項目が欠落しており、これによって失敗が発生します。次に、結果を反復処理して、レコードごとに操作が成功したかどうかを判別します。正常に処理された各レコードの ID をデバッグログに書き込むか、失敗したレコードのエラーメッセージと項目を書き込みます。この例では、1 つの成功した操作と 1 つの失敗が生成されます。



## Database.UpsertResult クラス

メソッドによって返される、upsert DML 操作の結果。

### 使用方法

Database.UpsertResult オブジェクトの配列は、データベースメソッドで返されます。UpsertResult 配列の各要素は、データベースメソッドの `sObject[]` パラメータとして渡された sObject 配列に対応します。つまり、UpsertResult 配列の最初の要素は、sObject 配列の最初の要素と一致します。また、UpsertResult 配列の 2 番目の要素は sObject 配列の 2 番目の要素と一致し、3 番目以降も同様です。sObject が 1 つのみ渡される場合、UpsertResults 配列には 1 つの要素が含まれます。

### メソッド

UpsertResult オブジェクトには、次のメソッドがあります。

名前	型	説明
	<code>Database.Error[]</code>	エラーが発生した場合、エラーコードと説明を示す 1 つ以上のデータ

名前	型	説明
		ベースエラーオブジェクトからなる配列。詳細は、「 <a href="#">Database.Error クラス</a> 」(ページ 426)を参照してください。
ID		更新または挿入しようとしている sObject の ID。この項目に値が入力されている場合、オブジェクトは正常に更新または挿入されています。この項目が空白の場合、そのオブジェクトに対する操作は失敗しています。
Boolean		レコードが作成された場合、Boolean 値は true に設定されます。レコードが更新された場合は false です。
Boolean		このオブジェクトに対する DML 操作が成功した場合、Boolean 値は true に設定されます。それ以外の場合は false です。

## Database.DeleteResult クラス

メソッドによって返される、delete DML 操作の結果。

### 使用方法

オブジェクトの配列は、データベースメソッドで返されます。DeleteResult 配列の各要素は、データベースメソッドの `sObject[]` パラメータとして渡された sObject 配列に対応します。つまり、DeleteResult 配列の最初の要素は、sObject 配列の最初の要素と一致します。また、DeleteResult 配列の 2 番目の要素は sObject 配列の 2 番目の要素と一致し、3 番目以降も同様です。sObject が 1 つのみ渡される場合、DeleteResults 配列には 1 つの要素が含まれます。

### メソッド

次に、クラスのインスタンスマソッドを示します。

名前	型	説明
<code>Database.Error[]</code>		エラーが発生した場合、エラーコードと説明を示す 1 つ以上のデータベースエラーオブジェクトからなる配列。詳細は、「 <a href="#">Database.Error クラス</a> 」(ページ 426)を参照してください。
ID		削除しようとしている sObject の ID。この項目に値が入力されている場合、オブジェクトは正常に削除されています。

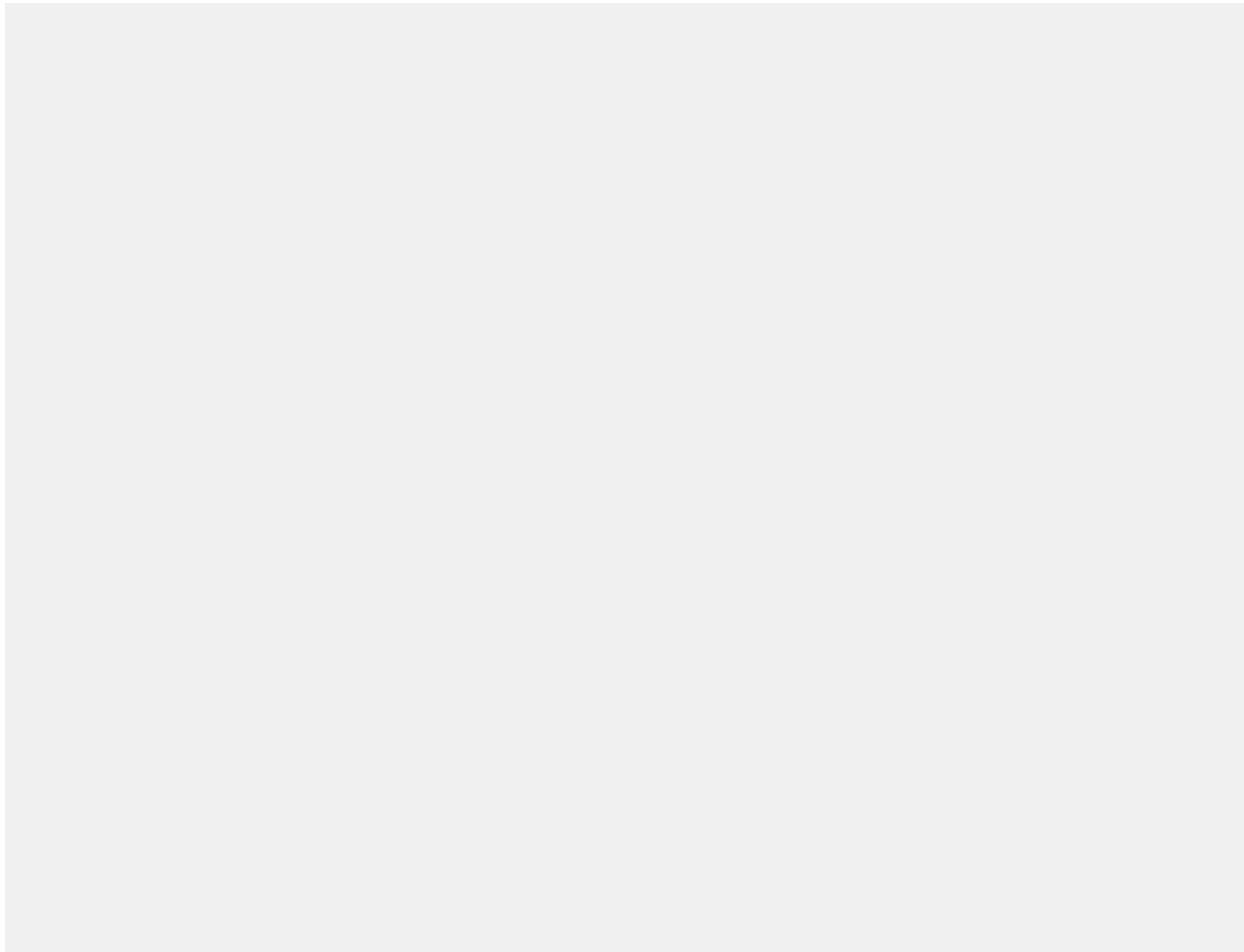
名前	型	説明
		ます。この項目が空白の場合、そのオブジェクトに対する操作は失敗しています。
	Boolean	このオブジェクトに対する DML 操作が成功した場合、Boolean 値は true に設定されます。それ以外の場合は false です。

## 例

次の例では、返された

オブジェクトを介して取得および反復処理する方法を示しま

す。  
の 2 番目のパラメータに false を指定して使用し、一部のクエリ済み取引先を削除して、失敗時にレコードの部分的な処理を行えるようにしています。次に、結果を反復処理して、レコードごとに操作が成功したかどうかを判別します。正常に処理された各レコードの ID をデバッグログに書き込むか、失敗したレコードのエラーメッセージと項目を書き込みます。



## **Database.UndeleteResult クラス**

## メソッド

すべてのインスタンスマソッドを次に示します。インスタンスマソッドは EmptyRecycleBinResult オブジェクトの特定のインスタンスで使用されます。引数を取るメソッドはありません。

名前	戻り値	説明
	Database.Errors []	このレコードまたは sObject を削除するときにエラーが発生した場合、1つ以上の Database.Error オブジェクトのリストが返されます。エラーが発生しない場合、このリストは空白です。
	ID	削除するレコードまたは sObject の ID を返します。
	Boolean	レコードまたは sObject がごみ箱から正常に削除された場合は TRUE 、正常に削除されない場合は FALSE を返します。

## Database.Error クラス

Database メソッドの使用時に DML 操作で発生したエラーに関する情報が含まれます。

## メソッド

次に、 Database.Error クラスのインスタンスマソッドを示します。

名前	戻り値	説明
getFields	String[]	1つ以上の項目名の配列を返します。オブジェクト内の項目でエラー条件に影響を与えるものが存在する場合、その項目を示します。
getMessage	string	エラーメッセージのテキストを返します。
getStatusCode	StatusCodes	エラーを特徴付けるコードを返します。状況コードの完全な一覧は、組織の WSDL ファイルで参照できます (Salesforce オンラインヘルプの「 Salesforce WSDL およびクライアント認証証明書のダウンロード」を参照してください)。

## Database.QueryLocator クラス

次の表に、 Database.QueryLocator クラスのメソッドのリストを示します。

名前	戻り値	説明
	string	Database.QueryLocator オブジェクトのインスタンス化に使用するクエリを返します。これは、 <a href="#">メソッド</a> をテストする場合に役立ちます。次に例を示します。
	Database.QueryLocatorIterator	<p>getQueryLocator クエリで <a href="#">キーワード</a>を使用してレコードのセットをロックすることはできません。メソッドは、バッチにあるレコードのセットを自動的にロックします。</p> <p>警告: クエリロケータを反復処理するには、このメソッドによって変数で返されるイテレータインスタンスを保存し、その変数を使用してコレクションを反復処理します。反復を実行するたびに <a href="#">コレクション</a> をコールすると、各コールで新しいイテレータインスタンスが返されるため、不適切な動作を生じる可能性があります。</p> <p><a href="#">「Database.QueryLocatorIterator クラス」</a> の例を参照してください。</p>

## Database.QueryLocatorIterator クラス

クエリロケータレコードセットに対するイテレータを表します。

### メソッド

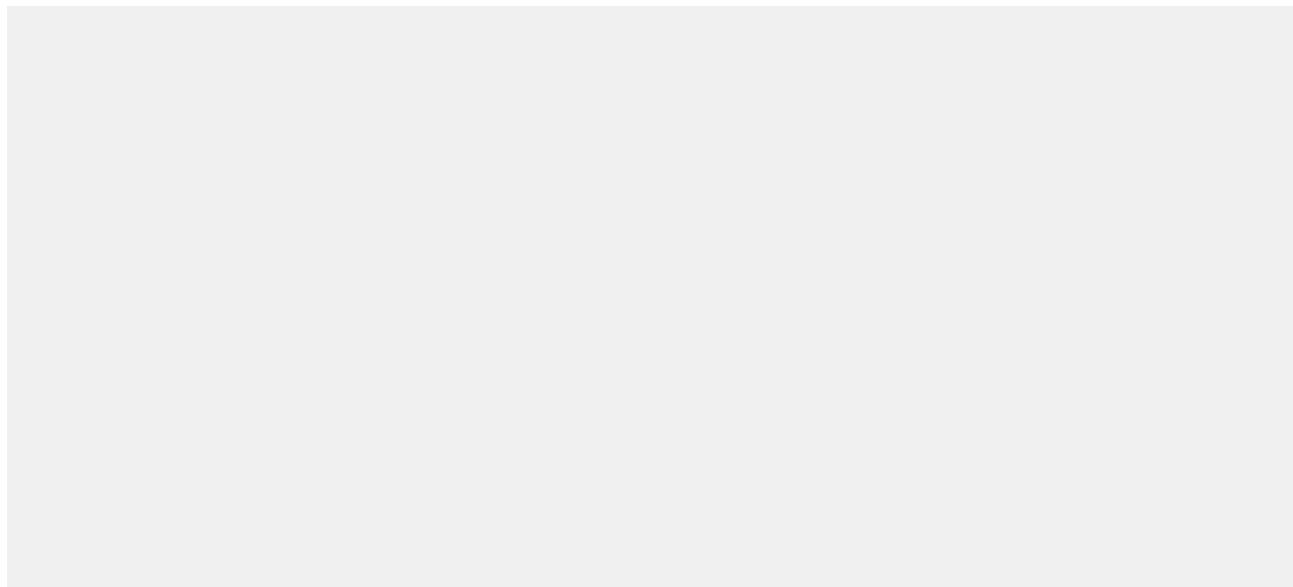
次に、クラスのインスタンスマソッドを示します。

メソッド	戻り値	説明
	Boolean	コレクション内に 1 つ以上のレコードが残っている場合は、それ以外の場合は <a href="#">true</a> を返します。

メソッド	戻り値	説明
	sObject	イテレータを次の sObject レコードに進め、sObject を返します。 戻り値は汎用 sObject 型であるため、具体的なデータ型を使用する場合は、キャストする必要があります。次に例を示します。

## サンプル

このサンプルでは、5 つの取引先が含まれるクエリロケータのイテレータを取得する方法を示します。このサンプルでは、`queryOptions` および `get` をコールして、コレクション内の各レコードを取得します。



## Database.DMLOptions プロパティ

DML 操作に関連するオプションを設定できるようにします。

### 使用方法

は、API バージョン 15.0 以降で保存された Apex にのみ使用できます。

### メソッド

クラスには、次のプロパティがあります。

名前	型	説明
	Boolean	<p>長い文字列の切り捨て動作を指定します。</p> <p>バージョン 15.0 より前の API に対して保存された Apex では、文字列に値を指定し、その値が大きすぎる場合、値は切り捨てられます。API バージョン 15.0 以降では、大きすぎる値が指定されると、操作は失敗し、エラーメッセージが返されます。</p> <p>プロパティを使用すると、API バージョン 15.0 以降に対して保存された Apex の新しい動作ではなく、以前の動作である切り捨てを使用するように指定できます。</p>
	Database. DmlOptions. Assignmentruleheader	<p>ケースまたはリードの作成時に使用する割り当てルールを指定します。</p> <p> メモ: database.DMLOptions オブジェクトは、ケースおよびリードの割り当てルールをサポートしますが、取引先またはテリトリー管理の割り当てルールはサポートしません。</p>
	Database. DmlOptions. EmailHeader	<p>イベントが発生した場合に送信される自動メールに関する追加情報を指定します。</p> <p>Salesforce ユーザインターフェースを使用して、次のようなイベントが発生した場合にメールを送信するかどうかを指定できます。</p> <ul style="list-style-type: none"> <li>・ ケースまたは ToDo の新規作成</li> <li>・ ケースコメントの作成</li> <li>・ ケースメールの取引先責任者への変換</li> <li>・ 新規ユーザのメール通知</li> <li>・ リードキューのメール通知</li> <li>・ パスワードのリセット</li> </ul> <p>API バージョン 15.0 以降に対して保存された Apex で、Database.DMLOptions プロパティを使用すると、コードの実行によりイベントのいずれかが発生したときに送信されるメールに関する追加情報を指定できます。</p>
	Database. DmlOptions. LocaleOptions	<p>Apex によって返される表示ラベルの言語を指定します。</p> <p>値は、de_DE や en_GB など、有効なユーザロケール（言語および国）である必要があります。値は文字列で、文字数は 2 から 5 文字です。最初の 2 文字は常に、「fr」や「en」などの ISO 言語コードです。値がさらに国別に評価される場合、文字列はアンダースコア（_）に続き、「US」や「UK」などの ISO 国コードが続きます。たと</p>

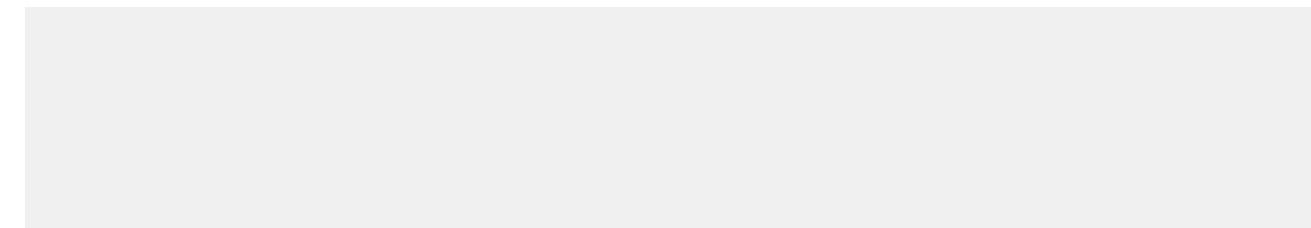
名前	型	説明
	Boolean	<p>えば、アメリカを示す文字列は「en_US」、カナダのフランス語圏を示す文字列は「fr_CA」です。</p> <p>Salesforce がサポートする言語の一覧は、Salesforce オンラインヘルプの「Salesforce がサポートする言語は?」を参照してください。</p> <p>部分的な完了を操作で許可するかどうかを指定します。</p> <p>が に設定されている場合、レコードでエラーが発生すると、すべての変更はロールバックされます。このプロパティのデフォルトが である場合、レコードにエラーがない限り、正常に処理されたレコードがコミットされます。</p> <p>このプロパティは、Salesforce.com API バージョン 20.0 以降で保存された Apex で使用できます。</p>

### Database.DmlOptions.AssignmentRuleHeader プロパティ

次に、  
で設定できるオプションを示します。

名前	型	説明
	ID	<p>ケースまたはリードに対して実行する特定の割り当てルールの ID を指定します。割り当てルールは有効または無効にできます。ID は、AssignmentRule sObject をクエリして取得することができます。assignmentRuleId が指定されている場合は、 を指定しないでください。</p> <p>値が適切な ID 形式 (15 文字または 18 文字の Salesforce ID) でない場合、コールは失敗し、例外が返されます。</p>
	Boolean	<p>ケースまたはリードに を指定した場合、システムはケースまたはリードのデフォルトの(有効な)割り当てルールを使用します。useDefaultRule が指定されている場合は、 を指定しないでください。</p>

次の例では、  
オプションを使用します。



次の例では、

オプションを使用します。

### Database.DmlOptions.EmailHeader プロパティ

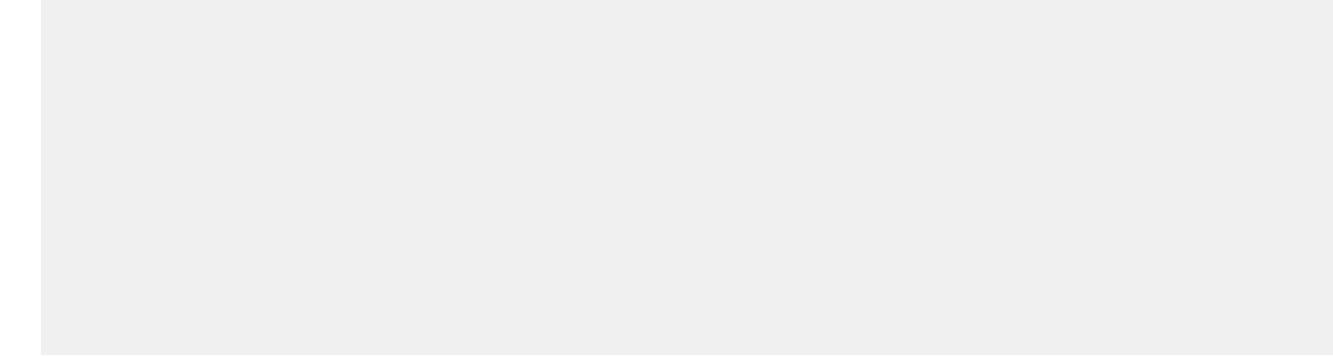
次に、プロパティで設定できるオプションを示します。

名前	型	説明
	Boolean	リード、ケースに対して自動応答ルールをトリガするか( )、トリガしないか( )を示します。Salesforce ユーザインターフェースで、このメールは、ケースの作成やユーザパスワードのリセットなど、さまざまなイベントによって自動的にトリガされます。この値がに設定されている場合、ケースが作成されると、に指定された取引先責任者のメールアドレスがあれば、メールはそのアドレスに送信されます。アドレスがない場合、メールはで指定されたアドレスに送信されます。
	Boolean	組織外のメールをトリガするか( )、トリガしないか( )を示します。Salesforce ユーザインターフェースで、このメールは、ケースの取引先責任者の作成、編集、削除によって自動的にトリガされます。   メモ: グループイベントによって Apex で送信されるメールには、追加の動作が含まれます。グループイベントとは、が true であるイベントです。オブジェクトは、グループイベントに招待されているユーザ、リード、または取引先責任者を追跡します。Apex を使用して送信されるグループイベントメールでは、次のような動作に注意してください。 <ul style="list-style-type: none"><li>リードまたは取引先責任者に対するグループイベントの招待状の送信は、オプションの影響を受けます。</li></ul>

名前	型	説明
	Boolean	<ul style="list-style-type: none"><li>グループイベントの更新または削除時に送信されるメールも、送信対象に基づき や オプションの影響を受けます。</li></ul> <p>組織内のユーザに送信されるメールをトリガするか ( )、トリガしないか ( ) を示します。Salesforce ユーザインターフェースで、このメールはパスワードのリセット、ユーザの新規作成、コメントのケースへの追加、ToDo の作成または変更など、さまざまなイベントによって自動的にトリガされます。</p> <p> メモ: グループイベントによって Apex で送信されるメールには、追加の動作が含まれます。グループイベントとは、 が true であるイベントです。EventAttendee オブジェクトは、グループイベントに招待されているユーザ、リード、または取引先責任者を追跡します。Apex を使用して送信されるグループイベントメールでは、次のような動作に注意してください。</p> <ul style="list-style-type: none"><li>ユーザに対するグループイベントの招待状の送信は、 オプションの影響を受けます。</li><li>グループイベントの更新または削除時に送信されるメールも、送信対象に基づき や オプションの影響を受けます。</li></ul>

次の例では、

オプションが指定されます。



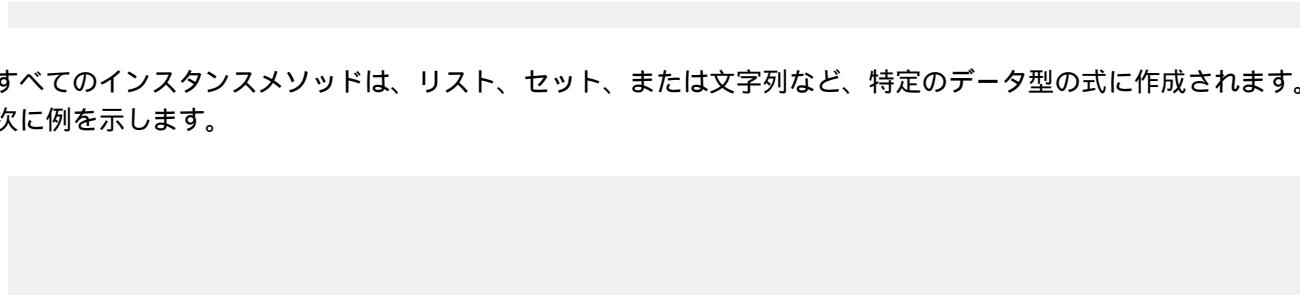
## Apex 標準クラスおよび標準メソッド

Apex には、プリミティブデータ型の式の、より複雑なオブジェクト向けの、静的メソッドとインスタンスマソッドの両方を含む標準クラスがあります。

標準静的メソッドは Java に類似しており、常に次のような形式となります。

`Class args`

プリミティブデータ型の標準静的メソッドには暗黙的なパラメータではなく、オブジェクトコンテキストなしで呼び出されます。たとえば、次の式は 1.75 の値を、他の値を使用せずに最も近い整数に丸めます。



メモ: に対して評価するオブジェクト式でメソッドがコールされる場合、Apex ランタイムエンジンは、Null ポインタの例外を発生させます。

一部のクラスでは、それらのメソッドのグループ化メカニズムとして名前空間を使用します。たとえば、クラスでは ApexPages 名前空間を使用します。

Apex 標準クラスは、次のカテゴリに分類されます。

- Primitives
- Collections
- enum

- [sObjects](#)
- [System](#)
- [Exceptions](#)

## Apex Primitive メソッド

Apex の多くのプリミティブデータ型には、データの追加処理に使用できるメソッドがあります。メソッドのあるプリミティブは次のとおりです。

- [Blob](#)
- [Boolean](#)
- [Date](#)
- [Datetime](#)
- [decimal](#)
- [Double](#)
- [ID](#)
- [Long](#)
- [String](#)
- [time](#)

### Blob メソッド

次に、Blob のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>s</i>	Blob	指定された文字列からバイナリオブジェクトを作成し、PDF ファイルとして符号化します。
	String <i>s</i>	Blob	指定した String <i>s</i> を Blob に割り当てます。次に例を示します。

次に、Blob のインスタンスマソッドを示します。

名前	引数	戻り値	説明
		Integer	Blob の文字数を返します。次に例を示します。

名前	引数	戻り値	説明
		string	Blob を string に割り当てます。

Blob についての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

## Boolean メソッド

次に、Boolean の静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>s</i>	Boolean	<p>指定した文字列を boolean 値に変換し、指定した文字列の値が「true」の場合に <code>true</code> を返します。ない場合は <code>false</code> を返します。</p> <p>指定した引数が null の場合は、例外が発生します。</p> <p>例:</p> <pre>Boolean b = Boolean.valueOf('true');</pre>
	Object <i>fieldValue</i>	Boolean	<p>指定した履歴管理項目の値を boolean 値に変換します。</p> <p>チェックボックス項目のように項目のデータ型が boolean 型に対応する場合は、<code>true</code> など、履歴 sObject の <code>Boolean</code> 項目または <code>Boolean</code> 項目でこのメソッドを使用します。</p> <p>例:</p> <pre>Boolean b = Boolean.valueOf(true);</pre>

名前	引数	戻り値	説明

Booleanについての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

## Date メソッド

次に、Date のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	Integer <i>year</i> Integer <i>month</i>	Integer	指定された <i>year</i> と <i>month</i> の月の日数を返します (1 = 1 月)。次の例では、1960 年の 2 月の日数を示しています。
	Integer <i>year</i>	boolean	指定した <i>year</i> がうるう年の場合、true を返します。
	Integer <i>year</i> Integer <i>month</i> Integer <i>date</i>	Date	<i>year</i> 、 <i>month</i> (1=1 月)、 <i>day</i> の integer 表現から date を構築します。次の例では、1960 年 2 月 17 日を作成します。
	String <i>Date</i>	Date	string から date を構築します。string の形式は、ローカルの日付形式によって異なります。次の例は、いくつかのロケールで機能します。
		Date	現在の日付を現在のユーザのタイムゾーンで返します。
	String <i>s</i>	Date	指定した string の値を含む date を返します。string は、ローカルタイムゾーンの標準の日付形式「yyyy-MM-dd HH:mm:ss」を使用する必要があります。次に例を示します。

名前	引数	戻り値	説明
	Object <i>fieldValue</i> Date		<p>指定した履歴管理項目の値を date に変換します。</p> <p>項目が date 項目の場合は、 など、履歴 sObject の <b>な</b> 項目または 項目でこのメソッドを使用します。</p> <p>例:</p>

次に、date のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	Integer <i>addDays</i>	Date	指定した <i>addDays</i> の数を date に追加します。次に例を示します。
	Integer <i>addMonths</i>	Date	指定した <i>addMonths</i> の数を date に追加します。
	Integer <i>addYears</i>	Date	指定した <i>addYears</i> の数を date に追加します。
		Integer	date の day-of-month コンポーネントを返します。たとえば、1999 年 2 月 5 日は、day 5 です。
		Integer	date の day-of-year コンポーネントを返します。たとえば、1999 年 2 月 36 日は、day 5 です。
	date <i>compDate</i>	Integer	メソッドをコールした date と <i>compDate</i> の間の日数を返します。メソッドをコールする date が <i>compDate</i> の後に発生する場合、戻り値は負になります。次に例を示します。
		string	コンテキストユーザのロケールを使用して、date を文字列として返します。
	date <i>compDate</i>	boolean	メソッドをコールした date が <i>compDate</i> と同じ場合、true を返します。次に例を示します。
		Integer	date の month コンポーネントを返します(1=1月)。

名前	引数	戻り値	説明
	date <i>compDate</i>	Integer	メソッドをコールした date と <i>compDate</i> の間の月数を返します。日付の差異は無視されます。たとえば、同じ年の 3月 1日と 3月 30日の場合、その間の月数は 0 です。
		Date	メソッドをコールした date の月の最初の日を返します。たとえば、1999年 7月 14日の場合は、1999年 7月 1日を返します。
		Date	コンテキストユーザのロケールに応じて、メソッドをコールした date の週の開始日を返します。たとえば、アメリカのロケールでは週は日曜日に始まり、ヨーロッパでは月曜日に始まります。次に例を示します。
		Integer	date の year コンポーネントを返します。

Date についての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

## dateTime メソッド

次に、dateTime のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	Long <i>i</i>	Datetime	dateTime を構築し、1970 年 1月 1日 00:00:00 (GMT) 以降の指定したミリ秒数を表すように初期化します。日付は GMT タイムゾーンで返されます。
	date <i>Date</i> time <i>Time</i>	Datetime	ローカルタイムゾーンの指定された <i>date</i> および <i>time</i> から dateTime を構築します。日付は GMT タイムゾーンで返されます。
	Integer <i>year</i> Integer <i>month</i>	Datetime	ローカルタイムゾーンの午前 0 時に、 <i>year</i> 、 <i>month</i> (1 = 1月)、 <i>day</i> の Integer 表現から dateTime を構築します。日付は GMT タイムゾーンで返されます。

名前	引数	戻り値	説明
	Integer <i>day</i>		たとえば、次のようにになります。
	Integer <i>year</i> Integer <i>month</i> Integer <i>day</i> Integer <i>hour</i> Integer <i>minute</i> Integer <i>second</i>	Datetime	ローカルタイムゾーンで、 <i>year</i> 、 <i>month</i> (1=1月)、 <i>day</i> 、 <i>hour</i> 、 <i>minute</i> 、および <i>second</i> の integer 表現から <i>dateTime</i> を構築します。日付は GMT タイムゾーンで返されます。 たとえば、次のようにになります。
	date <i>date</i> time <i>time</i>	Datetime	GMT タイムゾーンの指定された <i>date</i> および <i>time</i> から <i>dateTime</i> を構築します。
	Integer <i>year</i> Integer <i>month</i> Integer <i>date</i>	Datetime	GMT タイムゾーンの午前 0 時に、 <i>year</i> 、 <i>month</i> (1=1月)、 <i>day</i> の integer 表現から <i>dateTime</i> を構築します。
	Integer <i>year</i> Integer <i>month</i> Integer <i>date</i> Integer <i>hour</i> Integer <i>minute</i> Integer <i>second</i>	Datetime	GMT タイムゾーンで、 <i>year</i> 、 <i>month</i> (1=1月)、 <i>day</i> 、 <i>hour</i> 、 <i>minute</i> 、および <i>second</i> の integer 表現から <i>dateTime</i> を構築します。
		Datetime	GMT カレンダーに基づいて、現在の <i>dateTime</i> を返します。 たとえば、次のようにになります。
			返される <i>dateTime</i> の形式は です。

名前	引数	戻り値	説明
	String <i>datetime</i>	Datetime	<p>ユーザロケールのローカルタイムゾーンおよび形式の文字列 <i>datetime</i> から <code>dateTime</code> を構築します。日付は GMT タイムゾーンで返されます。</p> <p>次の例では _____ を使用して、英語(アメリカ)ロケール形式の文字列として渡される日付から <code>dateTime</code> を作成します。使用しているロケールによっては、日付文字列の形式を変更する必要があります。</p>
	String <i>s</i>	Datetime	<p>指定した string の値を含む <code>dateTime</code> を返します。string は、ローカルタイムゾーンの標準の日付形式「<code>yyyy-MM-dd HH:mm:ss</code>」を使用する必要があります。日付は GMT タイムゾーンで返されます。</p> <p>たとえば、次のようにになります。</p>

名前	引数	戻り値	説明
	Object <i>fieldValue</i>	Datetime	<p>指定した履歴管理項目の値を <code>dateTime</code> に変換します。</p> <p>項目が <code>dateTime</code> 項目の場合は、 など、履歴 <code>sObject</code> の _____ 項目または 項目でこのメソッドを使用します。</p> <p>例:</p>
	String <i>s</i>	Datetime	<p>指定した <code>string</code> の値を含む <code>dateTime</code> を返します。 <code>string</code> は、GMT タイムゾーンの標準の日付形式 「<code>yyyy-MM-dd HH:mm:ss</code>」を使用する必要があります。</p>

次に、`dateTime` のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	Integer <i>add1Days</i>	Datetime	<p>指定した <code>add1Days</code> の数を <code>datetime</code> に追加します。次に例を示します。</p>

名前	引数	戻り値	説明
	Integer <i>addHours</i>	Datetime	指定した <i>addHours</i> の数を datetime に追加します。
	Integer <i>addMinutes</i>	Datetime	指定した <i>addMinutes</i> の数を datetime に追加します。
	Integer <i>addMonths</i>	Datetime	指定した <i>addMonths</i> の数を datetime に追加します。
	Integer <i>addSeconds</i>	Datetime	指定した <i>addSeconds</i> の数を datetime に追加します。
	Integer <i>addYears</i>	Datetime	指定した <i>addYears</i> の数を datetime に追加します。
		Date	コンテキストユーザのローカルタイムゾーンで dateTIme の date コンポーネントを返します。
		Date	GMT タイムゾーンで dateTIme の date コンポーネントを返します。
		Integer	コンテキストユーザのローカルタイムゾーンで dateTIme の day-of-month コンポーネントを返します。たとえば、1999 年 2 月 5 日 午前 8 時 30 分 12 秒は、day 5 です。
		Integer	GMT タイムゾーンで dateTIme の day-of-month コンポーネントを返します。たとえば、1999 年 2 月 5 日 午前 8 時 30 分 12 秒は、day 5 です。
		Integer	コンテキストユーザのローカルタイムゾーンで dateTIme の day-of-year コンポーネントを返します。たとえば、2008 年 2 月 5 日 午前 8 時 30 分 12 秒は、day 36 です。
		Integer	GMT タイムゾーンで dateTIme の day-of-year コンポーネントを返します。たとえば、1999 年 2 月 5 日 午前 8 時 30 分 12 秒は、day 36 です。
		string	日付をローカルタイムゾーンに変換し、変換した日付をコンテキストユーザのロケールを使用して形式設定された文字列として返します。タイムゾーンを指定できない場合は、GMT が使用されます。
	String <i>dateFormat</i>	string	日付をローカルタイムゾーンに変換し、変換した日付を提供された Java の SimpleDateFormat を使用して文字列として返します。タイムゾーンを指定できない場合は、GMT が使用されます。次に例を示します。

名前	引数	戻り値	説明
			<p>Java の SimpleDateFormat についての詳細は、<a href="#">「Java SimpleDateFormat」</a> を参照してください。</p>
	<code>String dateFormat</code> <code>String timezone</code>	<code>string</code>	<p>日付を指定されたタイムゾーンに変換し、変換した日付を提供された Java の SimpleDateFormat を使用して文字列として返します。提供されたタイムゾーンが適切でない場合、GMT が使用されます。</p> <p>Java の SimpleDateFormat についての詳細は、<a href="#">「Java SimpleDateFormat」</a> を参照してください。</p> <p><code>timezone</code> 引数の有効なタイムゾーン値は、Java の <code>TimeZone.getAvailableIDs</code> メソッドから返されるタイムゾーンに対応する Java TimeZone クラスのタイムゾーンです。3 文字の省略名ではなく、タイムゾーンの完全名を使用することをお勧めします。</p> <p>次の例では <code>          </code> を使用して、GMT 日付を America/New_York タイムゾーンに変換し、指定された日付形式を使用してフォーマットします。</p>
	<code>String dateFormat</code> <code>String timeZone</code>	<code>string</code>	<p>提供された Java の SimpleDateFormat と GMT タイムゾーンを使用して、<code>dateTime</code> を文字列として返します。</p> <p>Java の SimpleDateFormat についての詳細は、<a href="#">「Java SimpleDateFormat」</a> を参照してください。</p>

名前	引数	戻り値	説明
		string	日付をローカルタイムゾーンに変換し、変換した日付を長い日付形式で返します。 たとえば、次のようにになります。
		Long	この <code>dateTime</code> オブジェクトで表された 1970 年 1 月 1 日 0 時 0 分 0 秒 (GMT) 以降のミリ秒数を返します。
		Integer	コンテキストユーザのローカルタイムゾーンで <code>dateTime</code> の hour コンポーネントを返します。
		Integer	GMT タイムゾーンで <code>dateTime</code> の hour コンポーネントを返します。
Datetime <i>compDt</i>	boolean		コンテキストユーザのローカルタイムゾーンで、メソッドをコールした <code>datetime</code> と <i>compDt</i> が同じ場合、true を返します。次に例を示します。
		Integer	コンテキストユーザのローカルタイムゾーンで <code>dateTime</code> の millisecond コンポーネントを返します。
		Integer	GMT タイムゾーンで <code>dateTime</code> の millisecond コンポーネントを返します。
		Integer	コンテキストユーザのローカルタイムゾーンで <code>dateTime</code> の minute コンポーネントを返します。
		Integer	GMT タイムゾーンで <code>dateTime</code> の minute コンポーネントを返します。

名前	引数	戻り値	説明
		Integer	コンテキストユーザのローカルタイムゾーンでdateTime の month コンポーネントを返します (1 = 1月)。
		Integer	GMT タイムゾーンで dateTime の month コンポーネントを返します (1 = 1月)。
		Integer	コンテキストユーザのローカルタイムゾーンでdateTime の second コンポーネントを返します。
		Integer	GMT タイムゾーンで dateTime の second コンポーネントを返します。
		Time	コンテキストユーザのローカルタイムゾーンでdateTime の time コンポーネントを返します。
		Time	GMT タイムゾーンで dateTime の time コンポーネントを返します。
		Integer	コンテキストユーザのローカルタイムゾーンでdateTime の year コンポーネントを返します。
		Integer	GMT タイムゾーンで dateTime の year コンポーネントを返します。

dateTime についての詳細は、[「プリミティブデータ型」](#) (ページ 30)を参照してください。

## Decimal メソッド

次に、Decimal のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	double d	Decimal	指定した double の値を含む decimal を返します。
	Long l	Decimal	指定した long の値を含む decimal を返します。
	String s	Decimal	指定した string の値を含む decimal を返します。 Java と同様、文字列は署名された decimal を表すものとして解釈されます。次に例を示します。

次に、decimal のインスタンスマソッドを示します。

名前	引数	戻り値	説明
		Decimal	decimal の絶対値を返します。
	decimal <i>divisor</i> , integer <i>scale</i>	Decimal	この decimal を <i>divisor</i> で除算し、スケール ( <i>scale</i> を使用した結果の小数点以下の数) を設定します。次の例で、D には 0.190 の値があります。
	decimal <i>divisor</i> , integer <i>scale</i> , object <i>roundingMode</i>	Decimal	この decimal を <i>divisor</i> で除算し、スケール ( <i>scale</i> を使用した結果の小数点以下の数) を設定し、必要に応じて <i>roundingMode</i> を使用して値を丸めます。 <i>roundingMode</i> の有効な値についての詳細は、「 <a href="#">丸めモード</a> 」を参照してください。次に例を示します。
		Double	decimal の double 値を返します。
		string	コンテキストユーザのロケールを使用して、decimal の string 値を返します。 指数が必要な場合、科学的記数法が使用されます。
		Integer	decimal の integer 値を返します。
		Long	decimal の long 値を返します。
	Integer <i>exponent</i>	Decimal	<i>exponent</i> の累乗まで乗算したこの decimal 値を返します。 <i>exponent</i> の値は、0 ~ 32,767 です。次に例を示します。

名前	引数	戻り値	説明
			<p>を使用する場合、1が返されます。</p> <p><a href="#">Math メソッドの</a> では負の値を使用できます。</p>
	Integer		<p>decimal の桁数の合計を返します。たとえば、decimal 値が 123.45 の場合、<code>Math.floor()</code> は 5 を返します。decimal 値が 123.123 の場合、<code>Math.floor()</code> は 6 を返します。次に例を示します。</p>
	Long		<p>decimal の丸められた近似値を返します。数値は、均等丸めモードを使用して、「最も近い近似値」である整数に丸められます。ただし、2つの近似値が等距離にある場合は、このモードでは偶数の近似値に丸められます。この丸めモードは、連続する計算に対して繰り返し適用される場合、統計的に累積エラーを最小化します。均等丸めモードについての詳細は、「<a href="#">丸めモード</a>」を参照してください。次に例を示します。</p>

名前	引数	戻り値	説明
System.RoundingMode Long <i>roundingMode</i>			decimal の丸められた近似値を返します。数値は、 <i>roundingMode</i> で指定された丸めモードを使用して、0 の小数点以下の桁数に丸められます。 <i>roundingMode</i> の有効な値についての詳細は、「 <a href="#">丸めモード</a> 」を参照してください。
Integer <i>scale</i>	Integer Decimal		decimal のスケール、つまり小数点以下の桁数を返します。 必要に応じて均等丸めモードを使用して、decimal のスケールを指定の小数点以下の桁数に設定します。均等丸めモードでは、「最も近い近似値」に丸められます。ただし、2 つの近似値が等距離にある場合は、このモードでは偶数の近似値に丸められます。均等丸めモードについての詳細は、「 <a href="#">丸めモード</a> 」を参照してください。 <i>scale</i> の値は、-33 ~ 33 です。 decimal のスケールを明示的に設定しない場合、スケールは decimal が作成された項目によって指定されます。

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>decimal が string から作成される場合、スケールは string の小数点以下の桁の文字数となります。</li> <li>decimal が小数以外の数値から作成される場合、数値を string に変換して小数点以下の桁の文字数を使用することで、スケールを決定します。</li> </ul>
	integer <i>scale</i> , System.RoundingMode <i>roundingMode</i>	Decimal	<p>必要に応じて <i>roundingMode</i> で指定された丸めモードを使用して、decimal のスケールを指定の小数点以下の桁数に設定します。 <i>roundingMode</i> の有効な値についての詳細は、「<a href="#">丸めモード</a>」を参照してください。<i>scale</i> の値は、-32,768 ~ 32,767 です。</p> <p>decimal のスケールを明示的に設定しない場合、スケールは decimal が作成された項目によって指定されます。</p> <ul style="list-style-type: none"> <li>decimal がクエリの一部として作成される場合、スケールはクエリから返される項目のスケールに基づきます。</li> <li>decimal が string から作成される場合、スケールは string の小数点以下の桁の文字数となります。</li> <li>decimal が小数以外の数値から作成される場合、数値を string に変換して小数点以下の桁の文字数を使用することで、スケールを決定します。</li> </ul>
		Decimal	末尾の 0 が削除された decimal を返します。
		string	科学的記数法を使用せずに、decimal の string 値を返します。

Decimal についての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

### 丸めモード

丸めモードでは、精度を破棄する数値操作の丸め動作を指定します。各丸めモードでは、丸められた結果の返される再下位の桁を計算する方法を示します。次に、*roundingMode* の有効な値を示します。

名前	説明
	<p>正の無限大に丸めます。つまり、結果が正の場合、このモードは、丸めモードと同じ動作をします。結果が負の場合、丸めモードと同じ動作をします。この丸めモードでは計算値は小さくなりません。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: 6</li> <li>• 入力値 1.1: 丸めモードの結果: 2</li> <li>• 入力値 -1.1: 丸めモードの結果: -1</li> <li>• 入力値 -2.7: 丸めモードの結果: -2</li> </ul>
	<p>0に丸めます。この丸めモードは常に、小数点以下の桁数部分を実行前に破棄します。この丸めモードでは計算値が大きくなることはありません。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: 5</li> <li>• 入力値 1.1: 丸めモードの結果: 1</li> <li>• 入力値 -1.1: 丸めモードの結果: -1</li> <li>• 入力値 -2.7: 丸めモードの結果: -2</li> </ul>
	<p>負の無限大に丸めます。つまり、結果が正の場合、このモードは、丸めモードと同じ動作をします。結果が負の場合、丸めモードと同じ動作をします。この丸めモードでは計算値は大きくなりません。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: 5</li> <li>• 入力値 1.1: 丸めモードの結果: 1</li> <li>• 入力値 -1.1: 丸めモードの結果: -2</li> <li>• 入力値 -2.7: 丸めモードの結果: -3</li> </ul>
	<p>「最も近い近似値」に丸められます。ただし、2つの近似値が等距離にある場合は、このモードでは切り捨てられます。破棄される小数点以下の値が 0.5 より大きい場合、この丸めモードは、丸めモードと同じ動作をします。0.5 以下の場合、丸めモードと同じ動作をします。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: 5</li> <li>• 入力値 1.1: 丸めモードの結果: 1</li> <li>• 入力値 -1.1: 丸めモードの結果: -1</li> <li>• 入力値 -2.7: 丸めモードの結果: -2</li> </ul>
	<p>「最も近い近似値」に丸められます。ただし、2つの近似値が等距離にある場合は、このモードでは偶数の近似値に丸められます。破棄される小数点以下の値の左側が奇数の場合、この丸めモードは、丸めモードと同じ動作をします。偶数の場合、丸めメソッドと同じ動作をします。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: 6</li> <li>• 入力値 1.1: 丸めモードの結果: 1</li> <li>• 入力値 -1.1: 丸めモードの結果: -1</li> <li>• 入力値 -2.7: 丸めモードの結果: -3</li> </ul>

名前	説明
	<p>この丸めモードは、連続する計算に対して繰り返し適用される場合、統計的に累積エラーを最小化します。</p> <p>「最も近い近似値」に丸められます。ただし、2つの近似値が等距離にある場合は、このモードでは切り上げられます。破棄される小数点以下の値が 0.5 以上の場合、この丸めメソッドは、丸めメソッドと同じ動作をします。0.5 より小さい場合、丸めメソッドと同じ動作をします。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: 6</li> <li>• 入力値 1.1: 丸めモードの結果: 1</li> <li>• 入力値 -1.1: 丸めモードの結果: -1</li> <li>• 入力値 -2.7: 丸めモードの結果: -3</li> </ul>
	<p>要求された操作の結果が正確であることが確認されました。つまり、丸める必要がありません。正確でない結果を生成する操作でこの丸めモードが指定される場合、exception が発生します。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: exception</li> <li>• 入力値 1.0: 丸めモードの結果: 1</li> </ul>
	<p>0から遠い方向に丸めます。この丸めモードは常に、小数点以下の桁数部分を実行前に切り捨てます。この丸めモードでは計算値が小さくなることはありません。次に例を示します。</p> <ul style="list-style-type: none"> <li>• 入力値 5.5: 丸めモードの結果: 6</li> <li>• 入力値 1.1: 丸めモードの結果: 2</li> <li>• 入力値 -1.1: 丸めモードの結果: -2</li> <li>• 入力値 -2.7: 丸めモードの結果: -3</li> </ul>

## Double メソッド

次に、Double のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>s</i>	Double	指定した string の値を含む double を返します。Java と同様、string は署名された decimal を表すものとして解釈されます。次に例を示します。
	Object <i>fieldValue</i>	Double	指定した履歴管理項目の値を double 値に変換します。 数値項目のように項目のデータ型が double 型に対応する場合は、など、履歴 sObject の

名前	引数	戻り値	説明
			項目または 使用します。 例:

次に、double のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	string		コンテキストユーザのロケールを使用して、double の string 値を返します。
	Integer		double の integer 値を integer に割り当てて返します。次に例を示します。
	Long		double の long 値を返します。
	Long		この double の値に最も近い long 値を返します。 例:

名前	引数	戻り値	説明
Double			

Doubleについての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

## ID メソッド

ID の静的メソッドを次に示します。

メソッド	引数	戻り値	説明
	String <i>s</i>	ID	指定した string を ID に変換してその ID を返します。

ID のインスタンスマソッドを次に示します。

メソッド	引数	戻り値	説明
	String <i>errorMsg</i>	Void	<p>カスタムエラーメッセージでレコードをマークし、DML 操作が行われないようにします。</p> <p><i>errorMsg</i>引数は、レコードにマークを付けるエラーメッセージです。</p> <p>このメソッドは、sObject メソッドと類似しています。</p> <p>例:</p> <p> メモ: このメソッドは、指定されたエラーメッセージ内のすべての HTML マークアップをエスケープします。エスケープさ</p>

メソッド	引数	戻り値	説明
			<p>れる文字は、 、 、 、 、 、 、 、 、 および です。これらの文字がエスケープされるため、HTMLマークアップはレンダリングされず、Salesforce ユーザインター フェースでテキストとして表示されるようになります。HTML エスケープが実行されるバージョンは、組織で重要な更新が有効になっているかどうかによって異なります。</p> <ul style="list-style-type: none"> <li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっている場合、このメソッドでは、Apex のすべてのバージョンに対して、指定されたエラーメッセージのすべての HTML マークアップがエスケープされます。</li> <li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっていない場合、このメソッドでは、Salesforce.com API バージョン 27.0 以降を使用して保存された Apex に対してのみ、エラーメッセージ内のすべての HTML マークアップがエスケープされます。</li> </ul>
	String <i>errorMsg</i> Boolean <i>escape</i>	Void	<p>カスタムエラーメッセージを使用してレコードにマークを付け、エラーメッセージをエスケープする必要があるかどうかを指定して、DML 操作が行われないようにします。</p> <p><i>errorMsg</i>引数は、レコードにマークを付けるエラーメッセージです。</p> <p><i>escape</i>引数は、カスタムエラーメッセージ内の HTML マークアップがエ</p>



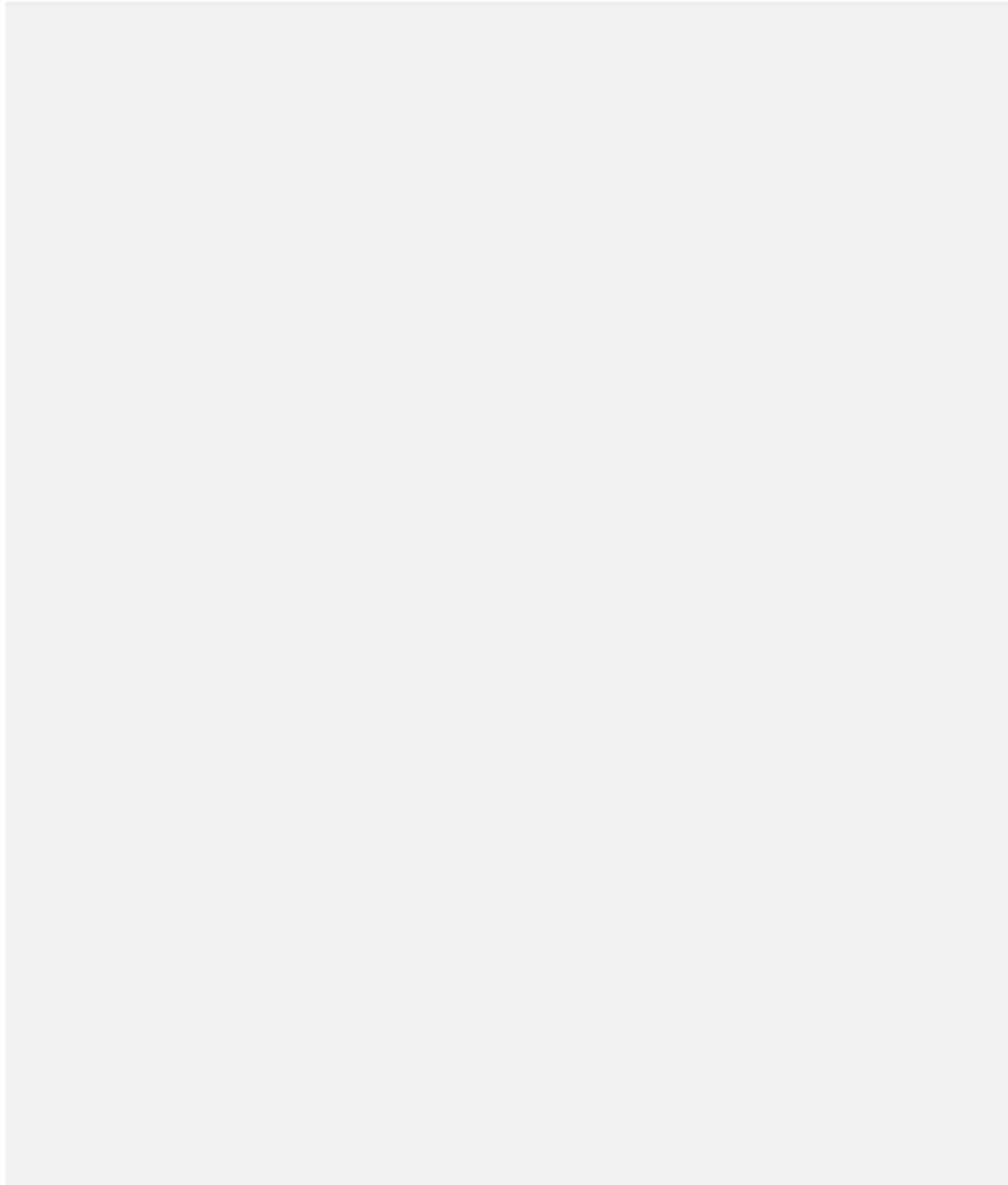
メソッド	引数	戻り値	説明
			<p>、 、 、 、 および です。これらの文字がエスケープされるため、HTML マークアップはレンダリングされず、Salesforce ユーザインター フェースでテキストとして表示されるようになります。HTML エスケープが実行されるバージョンは、組織で重要な更新が有効になっているかどうかによって異なります。</p> <ul style="list-style-type: none"> <li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっている場合、このメソッドでは、Apex のすべてのバージョンに対して、指定されたエラーメッセージのすべての HTML マークアップがエスケープされます。</li> <li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっていない場合、このメソッドでは、Salesforce.com API バージョン 27.0 以降を使用して保存された Apex に対してのみ、エラーメッセージ内のすべての HTML マークアップがエスケープされます。</li> </ul> <p>Exception <i>e</i> Boolean <i>escape</i></p> <p>カスタムエラーメッセージでレコードをマークし、DML 操作が行われないようにします。</p> <p><i>exception</i> 引数は、レコードにマークを付けるエラーメッセージを含む Exception オブジェクトまたはカスタム例外オブジェクトです。</p> <p><i>escape</i> 引数は、カスタムエラーメッセージ内の HTML マークアップがエ</p>

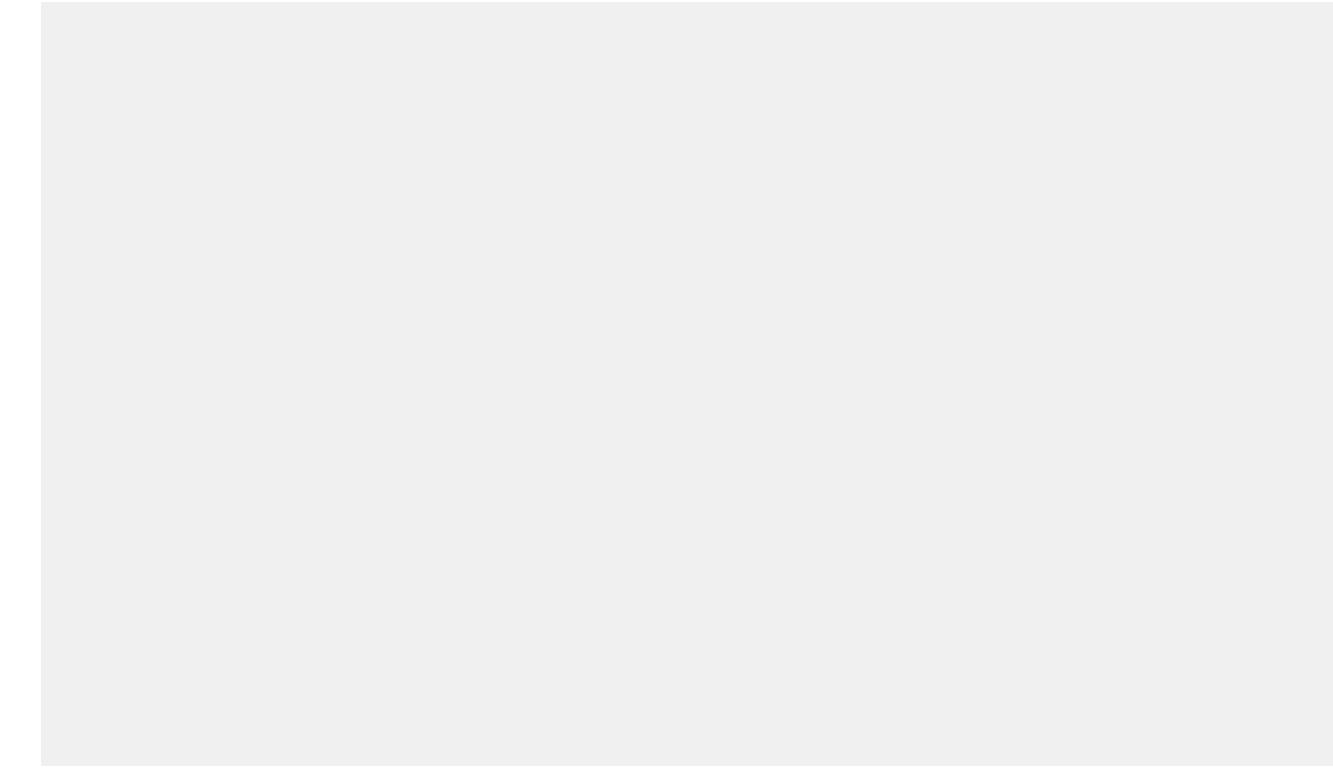
メソッド	引数	戻り値	説明
			<p>スケープされるか( )、否か( )を示します。</p> <p>エスケープされる文字は、 、 、 、 、 、 、 、 、 、 および です。これらの文字がエスケープされるため、HTMLマークアップはレンダリングされず、Salesforce ユーザインターフェースでテキストとして表示されるようになります。</p> <p> 警告: <code>escape</code> 引数に を指定するときは、慎重に行ってください。Salesforce ユーザインターフェースに表示されるエスケープ解除された文字列が、システムの脆弱性を示す場合があります。それらの文字列に有害なコードが含まれている可能性があるためです。エラーメッセージに HTML マークアップを含める場合は、 <code>escape</code> 引数を使用してこのメソッドをコールし、入力項目値などのすべての動的コンテンツをエスケープします。それ以外の場合は、 <code>escape</code> 引数に を指定するか、 <code>e</code> をコールします。</p>
	Schema.SObjectType		<p>この ID に対応する sObject のトークンを返します。このメソッドは Describe Information で使用されます。</p> <p>Describe についての詳細は、「<a href="#">Apex Describe Information について</a>」を参照してください。</p> <p>「<a href="#">サンプル: ID からの sObject トークンの取得</a>」の例を参照してください。</p>

### サンプル: ID からの sObject トークンの取得

このサンプルでは、 メソッドを使用して ID から sObject トークンを取得する方法を示します。このサンプルの メソッドは、 sObject の ID のリストを受け取って ownerId 項目を更新します。このリストには、同じデータ型の sObject の ID が含まれます。2 番目のパラメータは、新しい所有者 ID です。こ

これはfuture メソッドであるため、sObject のデータ型をパラメータとして受け取れません。そのため sObject の ID を受け取ります。このメソッドは、リストの1番目の ID から sObject トークンを取得し、オブジェクト名を取得する describe を実行して動的にクエリを構築します。次に、すべての sObject をクエリし、所有者 ID 項目を新しい所有者 ID に更新します。





## Integer メソッド

次に、Integer のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>s</i>	Integer	指定した string の値を含む integer を返します。Java と同様、string は署名された 10 進数を表すものとして解釈されます。次に例を示します。
	Object <i>fieldValue</i>	Integer	指定した履歴管理項目の値を integer 値に変換します。 数値項目のように項目のデータ型が integer 型に対応する場合は、など、履歴 sObject の項目または項目でこのメソッドを使用します。 例:

名前	引数	戻り値	説明

次に、integer のインスタンスマソッドを示します。

名前	引数	戻り値	説明
		string	コンテキストユーザのロケールを使用して、integer を文字列として返します。

Integer についての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

## Long メソッド

long のシステム静的メソッドを次に示します。

名前	引数	戻り値	説明
	String s	Long	指定した string の値を含む long を返します。Java と同様、文字列は署名された小数 long を表すものとして解釈されます。次に例を示します。

long のインスタンスマソッドを次に示します。

名前	引数	戻り値	説明
		string	コンテキストユーザのロケールを使用して、long の文字列形式を返します
		Integer	long の integer 値を返します

Long についての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

## String メソッド

次に、String のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	<code>String s</code>	<code>string</code>	<code>String s</code> の単一引用符の前にエスケープ文字 (\) を追加した <code>String</code> を返します。このメソッドは動的 SOQL ステートメントの作成時に役に立ち、SOQL インジェクションを回避します。動的 SOQL についての詳細は、「 <a href="#">動的 SOQL</a> 」を参照してください。「 <a href="#">文字列の分割例</a> 」も参照してください。
	<code>String s</code> <code>List&lt;String&gt; arguments</code>	<code>string</code>	<code>apex:outputText</code> 同じ方法で、現在の文字列を置換に使用するパターンとして扱います。
	<code>List&lt;Integer&gt; charArray</code>	<code>string</code>	整数のリストの値から <code>string</code> を返します。
	<code>List&lt;String&gt; strings</code>	<code>string</code>	指定したすべての <code>string</code> に共通する最初の文字シーケンスを <code>string</code> として返します。 例:  <pre>String str = 'Hello, World!'; String firstChar = String.valueOf(str.substring(0, 1)); System.out.println(firstChar); // Prints "H"</pre>
	<code>String s</code>	<code>boolean</code>	指定した <code>string</code> が空白、空 ("")、または <code>null</code> の場合は <code>false</code> 、それ以外の場合は <code>true</code> を返します。
	<code>String s</code>	<code>Boolean</code>	指定した <code>string</code> が空 ("") または <code>null</code> の場合は <code>false</code> 、それ以外の場合は <code>true</code> を返します。
	<code>String s</code>	<code>Boolean</code>	指定した <code>string</code> が空白でない、空 ("") でない、および <code>null</code> でない場合は <code>true</code> 、それ以外の場合は <code>false</code> を返します。

名前	引数	戻り値	説明
	String <i>s</i>	Boolean	指定した string が空 ("") でない、および null でない場合は <code>true</code> 、それ以外の場合は <code>false</code> を返します。
	Object <i>iterableObj</i>	String	指定した List などの Iterable オブジェクトの要素を、指定した区切り文字で区切られた 1 つの string に結合します。
	String <i>separator</i>		例:  例: <code>String result = iterableObj.join(' ');</code>
	date <i>d</i>	string	指定した date を表す string を、標準の「yyyy-MM-dd」形式で返します。次に例を示します。
	Datetime <i>dt</i>	string	指定した dateTime を表す string を、ローカルタイムゾーンの標準「yyyy-MM-dd HH:mm:ss」形式で返します。
	decimal <i>d</i>	string	指定された decimal を表す string を返します。
	double <i>d</i>	string	指定された double を表す string を返します。
			例:  例: <code>String result = Double.toString(123.45);</code>
	Integer <i>I</i>	string	指定された integer を表す string を返します。

名前	引数	戻り値	説明
	Long <i>l</i>	string	指定された long を表す string を返します。
	Object <i>x</i>	String	指定したオブジェクトの引数の文字列表現を返します。 例:
			引数が string でない場合、 <b>toString</b> メソッドはその引数に対する <b>toString</b> メソッド(利用可能な場合)、または引数がユーザ定義型の場合は上書きされた <b>toString</b> メソッドをコールすることで、引数を string に変換します。それ以外の、 <b>toString</b> メソッドが利用できない場合は引数の文字列表現を返します。
	Datetime <i>dt</i>	string	指定した dateTime を表す string を、GMT タイムゾーンの標準「yyyy-MM-dd HH:mm:ss」形式で返します。

次に、string のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	Integer <i>maxLength</i>	String	現在の string が指定した長さよりも長い場合、指定した長さに省略して省略記号を追加した string を返します。それ以外の場合、省略記号を付けずに元の string を返します。 <i>maxLength</i> が 4 未満の場合、このメソッドは実行時例外を発生させます。

名前	引数	戻り値	説明
例:			
	<code>Integer <i>maxWidth</i></code> <code>Integer <i>offset</i></code>	<code>String</code>	<p>指定した文字オフセットで開始する、指定した長さに省略した string を返します。返される string では、先頭と末尾の文字が削除されている場合はこれらの場所に省略記号が追加されます。</p> <p>オフセットは、返される string の左端の文字または省略記号に続く最初の文字であるとは限りませんが、結果のどこかに表示されます。これらに関係なく、<code>offset</code> は <code>maxWidth</code> を超える長さの string は返しません。</p> <p><code>maxWidth</code> が小さすぎる場合、このメソッドは実行時例外を発生させます。</p> <p>このメソッドは、Apache Commons Lang StringUtils ライブライリの <code>String.substring(int, int)</code> メソッドに基づいています。</p> <p>例:</p>

名前	引数	戻り値	説明
		string	Java メソッドに より最初の文字がタイトルの大文字または小文字 に変更された現在の string を返します。 例:
	Integer <i>size</i>	String	現在の string が指定したサイズで中央に表示され るように、左右に空白を埋め込んで返します。指 定したサイズが現在の string サイズよりも小さ い場合、string 全体が空白を追加せずに返されます。 例:
	Integer <i>size</i> String <i>padStr</i>	String	現在の string が指定したサイズで中央に表示され るように、左右に指定した string を埋め込んで返 します。指定したサイズが現在の string サイズよ りも小さい場合、string 全体が埋め込みなしで返 されます。 例:

名前	引数	戻り値	説明
			<p>String <i>compString</i> Integer</p> <p>string の各文字の unicode 値に基づいて、2 つの文字列を辞書編集的に比較します。結果は次のとおりです。</p> <ul style="list-style-type: none"><li>メソッドをコールした string が辞書編集的に <i>compString</i> の前に来る場合は負の Integer</li><li>メソッドをコールした string が辞書編集的に <i>compString</i> の後に来る場合は正の Integer</li><li>string が等しい場合は 0</li></ul> <p>string が異なるインデックス位置がない場合、辞書編集的に短い string が長い string の後になります。次に例を示します。</p>

名前	引数	戻り値	説明
			<p>String <i>compString</i> boolean</p> <p>現在の string に指定した string 内のいずれかの文字が含まれる場合は <code>true</code>、それ以外の場合は <code>false</code> を返します。</p> <p>例:</p>
			<p>String <i>compString</i> boolean</p> <p>現在の string に指定した文字シーケンス(大文字と小文字を区別しない)が含まれる場合は <code>true</code>、それ以外の場合は <code>false</code> を返します。</p> <p>例:</p>
			<p>String <i>compString</i> Boolean</p> <p>現在の string に指定した文字シーケンスが含まれない場合は <code>false</code>、それ以外の場合は <code>true</code> を返します。</p> <p><i>compString</i> が空の文字列の場合または現在の string が空の場合、このメソッドは <code>false</code> を返します。</p>

名前	引数	戻り値	説明
			<p><i>compString</i> が null の場合、このメソッドは実行時例外を返します。</p>
	String <i>compString</i>	Boolean	<p>現在の string に指定した文字シーケンス内の文字のみが含まれ、その他の文字は含まれない場合は true、それ以外の場合は false を返します。</p>
			<p>例:</p>
		Boolean	<p>現在の string に空白文字が含まれる場合は true、それ以外の場合は false を返します。</p>
	String <i>compString</i>	Integer	<p>現在の string 内で指定したサブ文字列が発生する回数を返します。</p>
		string	<p>現在の string のすべての空白文字を削除して返します。</p>
	String <i>compString</i>	String	<p>現在の string と指定した string 間の差異を返します。</p>
			<p><i>compString</i> が空の文字列の場合、このメソッドは空の文字列を返します。</p>
			<p><i>compString</i> が null の場合、このメソッドは実行時例外を発生させます。</p>
			<p>例:</p>



名前	引数	戻り値	説明
			<p>します。大文字と小文字は区別されません。たとえば、次のようにになります。</p>
	string		<p>必要に応じて、CSV 列の string を二重引用符で囲んで返します。</p> <p>string にカンマ、改行、または二重引用符が含まれる場合、返される string は二重引用符で囲まれます。また、string 内のすべての二重引用符はさらにもう1つの二重引用符でエスケープされます。</p> <p>string にカンマ、改行、二重引用符が含まれない場合、string が変更されずに返されます。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリの <a href="#">メソッド</a>に基づいています。</p>
	String		<p>EcmaScript string ルールを使用して string 内の文字をエスケープします。</p> <p>Apex string と EcmaScript string の唯一の違いは、EcmaScript では单一引用符とスラッシュ (/) がエスケープされる点です。</p>

名前	引数	戻り値	説明
			<p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリのメソッドに基づいています。</p> <p>例:</p>
		String	<p>HTML 3.0 エンティティを使用して string 内の文字をエスケープします。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリのメソッドに基づいています。</p> <p>例:</p>
		String	<p>HTML 4.0 エンティティを使用して string 内の文字をエスケープします。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリのメソッドに基づいています。</p>

名前	引数	戻り値	説明
			例:     
	String		<p>XML エンティティを使用して string 内の文字をエスケープします。</p> <p>5 つの基本 XML エンティティ (gt、lt、quot、amp、apos) のみをサポートします。DTD または外部エンティティはサポートしていません。0x7f より大きい Unicode 文字はエスケープされません。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリの <code>unescapeXml</code> メソッドに基づいています。</p>
	String s	Integer	<p>現在の string と指定した string 間のレーベンシュタイン距離を返します。</p> <p>レーベンシュタイン距離は、ある string から別の string に変更するために必要な変更の回数です。1</p>

名前	引数	戻り値	説明
			<p>文字の変更 (削除、挿入、または置換) が1つの変更としてカウントされます。</p> <p>例:</p>
	<code>String s</code> <code>Integer threshold</code>	<code>Integer</code>	<p>現在の <code>string</code> と指定した <code>string</code> 間のレーベンシュタイン距離が指定したしきい値以下の場合はその距離を返します。それ以外の場合は -1 を返します。</p> <p>レーベンシュタイン距離は、ある <code>string</code> から別の <code>string</code> に変更するために必要な変更の回数です。1 文字の変更 (削除、挿入、または置換) が1つの変更としてカウントされます。</p> <p>例:</p> <p>この例では、レーベンシュタイン距離は3ですが、しきい値の引数が2でこの距離よりも小さいため、このメソッドは -1 を返します。</p>
		<code>Integer</code>	<p>この文字列のハッシュコード値を返します。</p> <p>この値は、Java の同等メソッドによって計算されるハッシュコードに基づきます。</p> <p>このメソッドを使用して、<code>String</code> メンバー変数を含むカスタム型に対してハッシュコードの計算を</p>

名前	引数	戻り値	説明
			簡単にすることができます。各 String 変数のハッシュコードに基づいて、使用しているデータ型のハッシュコードを計算できます。次に例を示します。
			カスタム型を持つハッシュコードメソッドの使用の詳細については、「 <a href="#">対応付けのキーとセットでのカスタムデータ型の使用</a> 」を参照してください。
	String <i>subString</i>	Integer	指定したサブ文字列が最初に発生したインデックスを返します。サブ文字列がない場合、このメソッドは -1 を返します。
	String <i>substring</i>	Integer Integer <i>i</i>	インデックス <i>i</i> の位置から指定したサブ文字列が最初に出現した位置のインデックス (開始値 0) を返します。サブ文字列がない場合、このメソッドは -1 を返します。次に例を示します。
	String <i>substring</i>	Integer	サブ文字列で指定したいずれかの文字が最初に発生した位置の開始値 0 のインデックスを返します。指定したすべての文字が 1 つもない場合、-1 が返されます。

名前	引数	戻り値	説明
			例:   
			<p>String <i>substring</i> Integer</p> <p>指定したサブ文字列内に存在しない文字が最初に発生した位置の開始値 0 のインデックスを返します。サブ文字列内の文字のみで構成されている場合、-1 を返します。</p>
			例:   
			<p>String <i>s</i> Integer</p> <p>指定した string と異なる文字が最初に出現した位置のインデックス (開始値 0) を返します。</p>
			例:   
			<p>String <i>substring</i> Integer</p> <p>指定したサブ文字列 (大文字と小文字を区別しない) が最初に発生した位置の開始値 0 のインデックスを返します。サブ文字列がない場合、このメ</p>



名前	引数	戻り値	説明
		Boolean	現在の string 内のすべての文字が Unicode 文字または空白のみの場合は <code>false</code> 、それ以外の場合は <code>true</code> を返します。
		Boolean	現在の string 内のすべての文字が Unicode 文字または数字のみの場合は <code>false</code> 、それ以外の場合は <code>true</code> を返します。
			例:
		Boolean	現在の string 内のすべての文字が Unicode 文字、数字、または空白のみの場合は <code>false</code> 、それ以外の場合は <code>true</code> を返します。
		Boolean	現在の string に印字可能な ASCII 文字のみが含まれる場合は <code>false</code> 、それ以外の場合は <code>true</code> を返します。
		Boolean	現在の string に Unicode 数字のみが含まれる場合は <code>false</code> 、それ以外の場合は <code>true</code> を返します。 小数点 (1.2) は Unicode 数字ではありません。

名前	引数	戻り値	説明
		Boolean	現在の string に Unicode 数字または空白のみが含まれる場合は <code>true</code> 、それ以外の場合は <code>false</code> を返します。 小数点 (1.2) は Unicode 数字ではありません。
		Boolean	現在の string に空白文字のみが含まれる場合は <code>true</code> 、それ以外の場合は <code>false</code> を返します。
	<code>String substring</code>	<code>Integer</code>	指定したサブ文字列が最後に発生したインデックスを返します。サブ文字列がない場合、このメソッドは -1 を返します。
	<code>String substring</code> <code>Integer</code> <code>endPosition</code>	<code>Integer</code>	インデックス 0 の文字から始まり、指定したインデックスで終わる範囲で、指定したサブ文字列が最後に発生した位置のインデックスを返します。 サブ文字列がない場合または <code>endPosition</code> が負の場合、このメソッドは -1 を返します。 <code>endPosition</code> が現在の string の最後のインデックスよりも大きい場合、string 全体が検索されます。 例:  <code>String str = 'Hello World'; Integer index = str.substring(6).lastIndexOf('o');</code>
	<code>String substring</code>	<code>Integer</code>	指定したサブ文字列 (大文字と小文字を区別しない) が最後に発生した位置のインデックスを返します。 サブ文字列がない場合、このメソッドは -1 を返します。

名前	引数	戻り値	説明
			例:   
	String <i>substring</i> Integer <i>endPosition</i>	Integer	インデックス 0 の文字から始まり、指定したインデックスで終わる範囲で、指定したサブ文字列(大文字と小文字を区別しない)が最後に発生した位置のインデックスを返します。  サブ文字列がない場合または <i>endPosition</i> が負の場合、このメソッドは -1 を返します。 <i>endPosition</i> が現在の string の最後のインデックスよりも大きい場合、string 全体が検索されます。
	Integer <i>length</i>	String	現在の string の左端から指定した長さ分の文字を返します。  <i>length</i> が string のサイズよりも大きい場合、string 全体が返されます。

名前	引数	戻り値	説明
	Integer <i>length</i>	String	<p>指定した長さになるまで現在の string の左側に空白を埋め込んで返します。</p> <p><i>length</i> が現在の string サイズ以下の場合、string 全体が空白の埋め込みなしで返されます。</p> <p>例:</p>
	Integer		<p>string に含まれる 16 ビット Unicode 文字の数を返します。次に例を示します。</p>
	Integer <i>startIndex</i> String Integer <i>length</i>		<p>指定した開始値0の <i>startIndex</i> の文字で始まる、<i>length</i> によって指定された文字数の新しい string を返します。</p> <p><i>startIndex</i> が負の場合は、ゼロとみなされます。</p> <p><i>length</i> が負またはゼロの場合、空の string が返されます。<i>length</i> が残りの文字数よりも大きい場合、string の残りが返されます。</p> <p>このメソッドは、 ドと ドに類似していますが、2 番目の引数が返される 文字数である点が異なります。</p> <p>例:</p>

名前	引数	戻り値	説明
		String	<p>現在の string の先頭、末尾、繰り返しの空白文字を削除して返します。</p> <p>このメソッドは、Apache Commons Lang StringUtils ライブラリの メソッドに基づいています。</p> <p>例:</p>
		String <i>substring</i>	<p>発生したすべての指定したサブ文字列を削除して、結果の文字列を返します。</p> <p>例:</p>
		String <i>substring</i>	<p>指定したサブ文字列が string の末尾に発生した場合にのみサブ文字列を削除します。</p> <p>例:</p>

名前	引数	戻り値	説明
	String <i>substring</i>	String	指定したサブ文字列 (大文字と小文字を区別しない) が string の末尾に発生した場合にのみ、そのサブ文字列を削除します。 例:   
	String <i>substring</i>	String	指定したサブ文字列が string の先頭に発生した場合にのみ、そのサブ文字列を削除します。 例:   
	String <i>substring</i>	String	指定したサブ文字列 (大文字と小文字を区別しない) が string の先頭に発生した場合にのみ、そのサブ文字列を削除します。 例:   

名前	引数	戻り値	説明
	Integer <i>numTimes</i>	String	現在の string を指定した回数だけ繰り返して返します。 例:
	String <i>separator</i> Integer <i>numTimes</i>	String	現在の string を指定した回数だけ繰り返し、指定した区切り文字を使用して、繰り返される string を区切って返します。 例:
	String <i>target</i> String <i>replacement</i>	string	リテラルターゲットシーケンス <i>target</i> に一致する文字列の各サブ文字列を、指定したリテラル置換シーケンス <i>replacement</i> と置き換えます。
	String <i>regExp</i> String <i>replacement</i>	string	正規表現 <i>regExp</i> に一致する文字列の各サブ文字列を、置換シーケンス <i>replacement</i> と置き換えます。正規表現についての詳細は、Java クラスを参照してください。
	String <i>regExp</i> String <i>replacement</i>	string	正規表現 <i>regExp</i> に一致する文字列の最初のサブ文字列を、置換シーケンス <i>replacement</i> と置き換えます。正規表現についての詳細は、Java クラスを参照してください。
		String	すべての文字を逆順にした string を返します。

名前	引数	戻り値	説明
	Integer <i>length</i>	String	<p>現在の string の右端から指定した長さ分の文字を返します。</p> <p><i>length</i> が string のサイズよりも大きい場合、string 全体が返されます。</p> <p>例:</p>
	Integer <i>length</i>	String	<p>指定した長さになるまで現在の string の右側に空白を埋め込んで返します。</p> <p><i>length</i> が現在の string サイズ以下の場合は、string 全体が空白の埋め込みなしで返されます。</p> <p>例:</p>
	String <i>regExp</i> Integer <i>limit</i>	String []	<p>文字列の各サブ文字列を含むリストを返します。このサブ文字列は、正規表現 <i>regExp</i>、または文字列の末尾に達することで終了します。正規表現についての詳細は、Java <a href="#">クラス</a>を参照してください。</p> <p>このサブ文字列は、文字列の中で発生した順序でリストに記述されます。<i>regExp</i> が string のどの部分にも一致しない場合、結果リストには元の文字列を含む要素が 1 つだけ含まれます。</p> <p>(省略可能) <i>limit</i> パラメータは、パターンが適用された回数を制御するため、リストの長さに影響を与えます。</p>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>limit</i> が 0 より大きい場合、パターンは最大 <i>limit</i>-1 回適用されたことになります。また、リストの長さは最大 <i>limit</i> となり、リストの最後のエントリには最後に一致した区切り文字移行のすべての入力が含まれます。</li> <li>• <i>limit</i> が正の値でない場合、パターンを何度でも適用することが可能となり、リストの長さも任意となります。</li> <li>• <i>limit</i> が 0 の場合、パターンは何度でも適用することが可能となり、リストの長さも任意となりますですが、残りの続く空の文字列は破棄されます。</li> </ul> <p>たとえば、 次のようにになります。</p> <ul style="list-style-type: none"> <li>• _____ は _____ を生成します。</li> </ul> <p>「<a href="#">文字列の分割例</a>」も参照してください。</p> <p>List&lt;String&gt;</p> <p>現在の string を文字の種別ごとに分割し、同じ種別の連続文字グループのリストを完全なトークンとして返します。</p> <p>使用される文字の種別についての詳細は、  <a href="#">「java.lang.Character.getType(char)」</a> を参照してください。</p> <p>例:</p>

名前	引数	戻り値	説明	
		List<String>	<p>現在の string を文字の種別ごとに分割し、同じ種別の連続文字グループのリストを完全なトークンとして返します。ただし、小文字トークンの直前に大文字がある場合、その大文字は直前の小文字トークンではなく後続の小文字トークンに属します。</p> <p>使用される文字の種別についての詳細は、<a href="#">「java.lang.Character.getType(char)」</a> を参照してください。</p> <p>例:</p>	
		String <i>prefix</i>	boolean	メソッドをコールした string が <i>prefix</i> で始まる場合、 <i>true</i> を返します。
		String <i>prefix</i>	Boolean	現在の string が指定したプレフィックス(大文字と小文字を区別しない)で始まる場合は <i>true</i> を返します。
		String <i>htmlInput</i>	String	HTMLマークアップを入力文字列から削除し、ブレーンテキストを返します。

名前	引数	戻り値	説明
	Integer <i>startIndex</i> String		指定した開始値 0 の <i>startIndex</i> の文字で始まり string の末尾まで続く新しい string を返します。
	Integer <i>startIndex</i> String Integer <i>endIndex</i>		指定した開始値 0 の <i>startIndex</i> の文字で始まり <i>endIndex</i> - 1 の文字まで続く新しい string を返します。たとえば、次のようにになります。
	String <i>separator</i>	String	指定した区切り文字が最初に出現した位置より後にあるサブ文字列を返します。 例:
	String <i>separator</i>	String	指定した区切り文字が最後に出現した位置より後にあるサブ文字列を返します。 例:

名前	引数	戻り値	説明
	String <i>separator</i>	String	指定した区切り文字が最初に出現した位置より前にあるサブ文字列を返します。 例:
	String <i>separator</i>	String	指定した区切り文字が最後に出現した位置より前にあるサブ文字列を返します。 例:
	String <i>tag</i>	String	指定した string で囲まれたサブ文字列を返します。 例:
	String <i>open</i> String <i>close</i>	String	2 つの指定した string 間に出現したサブ文字列を返します。 例:

名前	引数	戻り値	説明
	String <i>open</i> String <i>close</i>	String	string のすべての文字の大文字と小文字を入れ替えて返します。  大文字およびタイトルの文字を小文字に変換し、小文字を大文字に変換します。  例:
		String	string のすべての文字を、デフォルトのロケールのルールを使用して、小文字に変換します。
	String <i>locale</i>	string	string のすべての文字を、指定したロケールのルールを使用して、小文字に変換します。
		String	string のすべての文字を、デフォルトのロケールのルールを使用して、大文字に変換します。たとえば、次のようになります。
	String <i>locale</i>	string	string のすべての文字を、指定したロケールのルールを使用して、大文字に変換します。

名前	引数	戻り値	説明
		String	<p>文字列のコピーを返します。このとき先頭と末尾の空白文字は含まれません。</p> <p>タブや改行文字など、先頭と末尾の ASCII 制御文字も削除されます。文の開始と終了にない空白文字と制御文字は削除されません。</p>
		String	<p>現在の string の最初の文字を小文字にして返します。</p> <p>例:</p>
		String	<p>エスケープ解除された CSV 列を表す string を返します。</p> <p>string が二重引用符で囲まれていてカンマ、改行、または二重引用符が含まれる場合、二重引用符は削除されます。また、二重引用符でエスケープされた文字(二重引用符のペア)は、エスケープが解除されて 1 つの二重引用符になります。</p> <p>string が二重引用符で囲まれていない場合、または二重引用符で囲まれていてカンマ、改行、二重引用符が含まれない場合、string が変更されずに返されます。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリのメソッドに基づいています。</p> <p>例:</p>

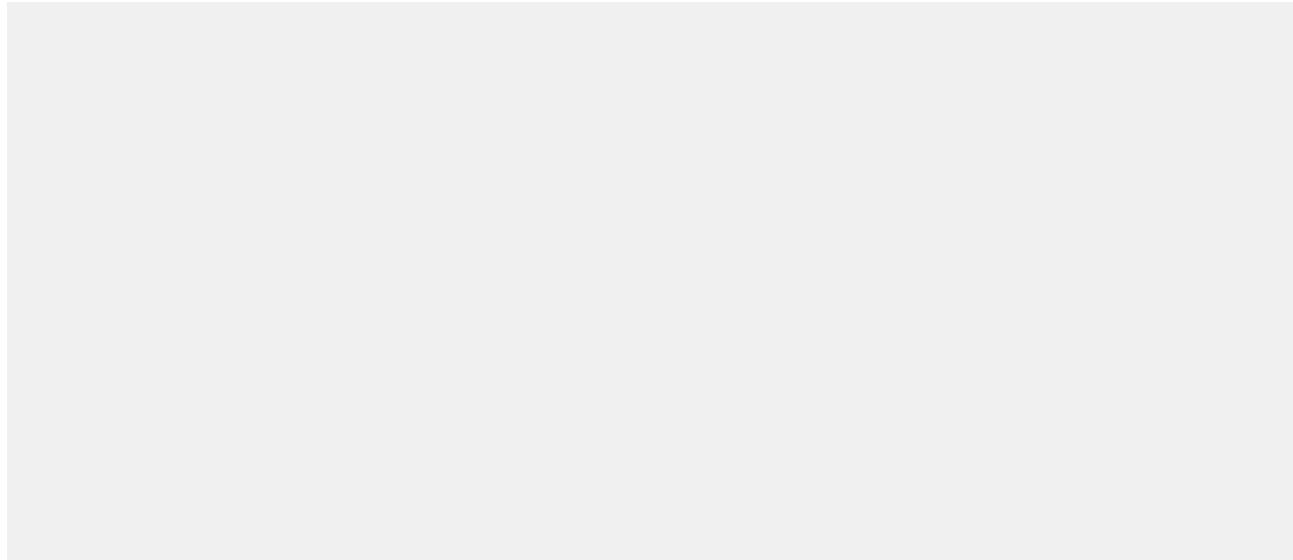
名前	引数	戻り値	説明
	String		<p>string 内にある EcmaScript リテラルのエスケープを解除します。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリの メソッドに基づいています。</p> <p>例:</p>
	String		<p>HTML 3.0 エンティティを使用して string 内の文字のエスケープを解除します。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリの メソッドに基づいています。</p> <p>例:</p>

名前	引数	戻り値	説明
		String	<p>HTML 4.0 エンティティを使用して string 内の文字のエスケープを解除します。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリのメソッドに基づいています。</p> <p>例:</p>
		String	<p>XML エンティティを使用して string 内の文字のエスケープを解除します。</p> <p>5 つの基本 XML エンティティ (gt、lt、quot、amp、apos) のみをサポートします。DTD または外部エンティティはサポートしていません。</p> <p>このメソッドは、Apache Commons Lang StringEscapeUtils ライブラリのメソッドに基づいています。</p> <p>例:</p>

String についての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

### 文字列の分割例

次の例では、バックスラッシュを区切り文字として使用して文字列を分割しています。



## Time メソッド

次に、Time のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	Integer <i>hour</i>	Time	
	Integer <i>minutes</i>		
	Integer <i>seconds</i>		
	Integer <i>milliseconds</i>		

次に、Time のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	Integer <i>addHours</i>	Time	指定した <i>addHours</i> の数を <i>time</i> に追加します。
	Integer <i>addMilliseconds</i>	Time	指定した <i>addMilliseconds</i> の数を <i>time</i> に追加します。

名前	引数	戻り値	説明
	Integer <i>addlMinutes</i>	Time	指定した <i>addlMinutes</i> の数を time に追加します。 次に例を示します。
	Integer <i>addlSeconds</i>	Time	指定した <i>addlSeconds</i> の数を time に追加します。
		Integer	time の hour コンポーネントを返します。次に例を示します。
		Integer	time の millisecond コンポーネントを返します。
		Integer	time の minute コンポーネントを返します。
		Integer	time の second コンポーネントを返します。

Time についての詳細は、「[プリミティブデータ型](#)」(ページ 30)を参照してください。

## Apex Collection メソッド

Apex のすべてのコレクションには、データを割り当て、取得、および処理するための、関連付けられたメソッドがあります。Collection メソッドは次のとおりです。

- [list](#)
- [map](#)
- [set](#)



メモ: コレクションに保持できる項目の数に制限はありません。ただし、[ヒープサイズ](#)には制限があります。

## List メソッド

List メソッドはすべてインスタンスマソッドで、リストの特定のインスタンスで動作します。たとえば、次のコードは `removeAll()` からすべての要素を削除します。

メソッドにはパラメータは含まれませんが、それをコールするリスト自身が暗黙的なパラメータです。

次に、List のインスタンスマソッドを示します。



メモ: 次の表では、`List__elem` はリストと同じデータ型の 1 つの要素を表します。

名前	引数	戻り値	説明
	任意の型 <code>e</code>	Void	リストの最後に要素 <code>e</code> を追加します。次に例を示します。
	<code>Integer i</code> 任意の型 <code>e</code>	Void	要素 <code>e</code> をリストのインデックス <code>i</code> の位置に挿入します。次の例では 6 つの要素を持つリストが作成され、最初と 2 番目のインデックス位置に整数が追加されます。
	List <code>l</code>	Void	リスト <code>l</code> のすべての要素を、メソッドをコールするリストに追加します。両方のリストは同じデータ型である必要があります。

名前	引数	戻り値	説明
	Set <i>s</i>	Void	セット <i>s</i> のすべての要素を、メソッドをコールするリストに追加します。セットとリストのデータ型は同じである必要があります。
		Void	すべての要素をリストから削除し、続いてリストの長さを 0 に設定します。
		list (同じデータ型)	リストの重複コピーを作成します。 これが sObject レコードのリストである場合、重複リストはリストの浅いコピーとなります。つまり、重複には各オブジェクトへの参照がありますが、sObject レコード自体は重複しません。次に例を示します。

sObject レコードもコピーするには、  
メソッドを使用する必要があります。

名前	引数	戻り値	説明
	Boolean <i>opt_preserve_id</i>	list (同じオブジェクトデータ型)	sObject レコード自体を含む、sObject レコードのリストの重複コピーを作成します。次に例を示します。
	boolean		
	<i>opt_preserve_readonly_timestamps</i>		
	boolean		
	<i>opt_preserve_autonumber</i>		

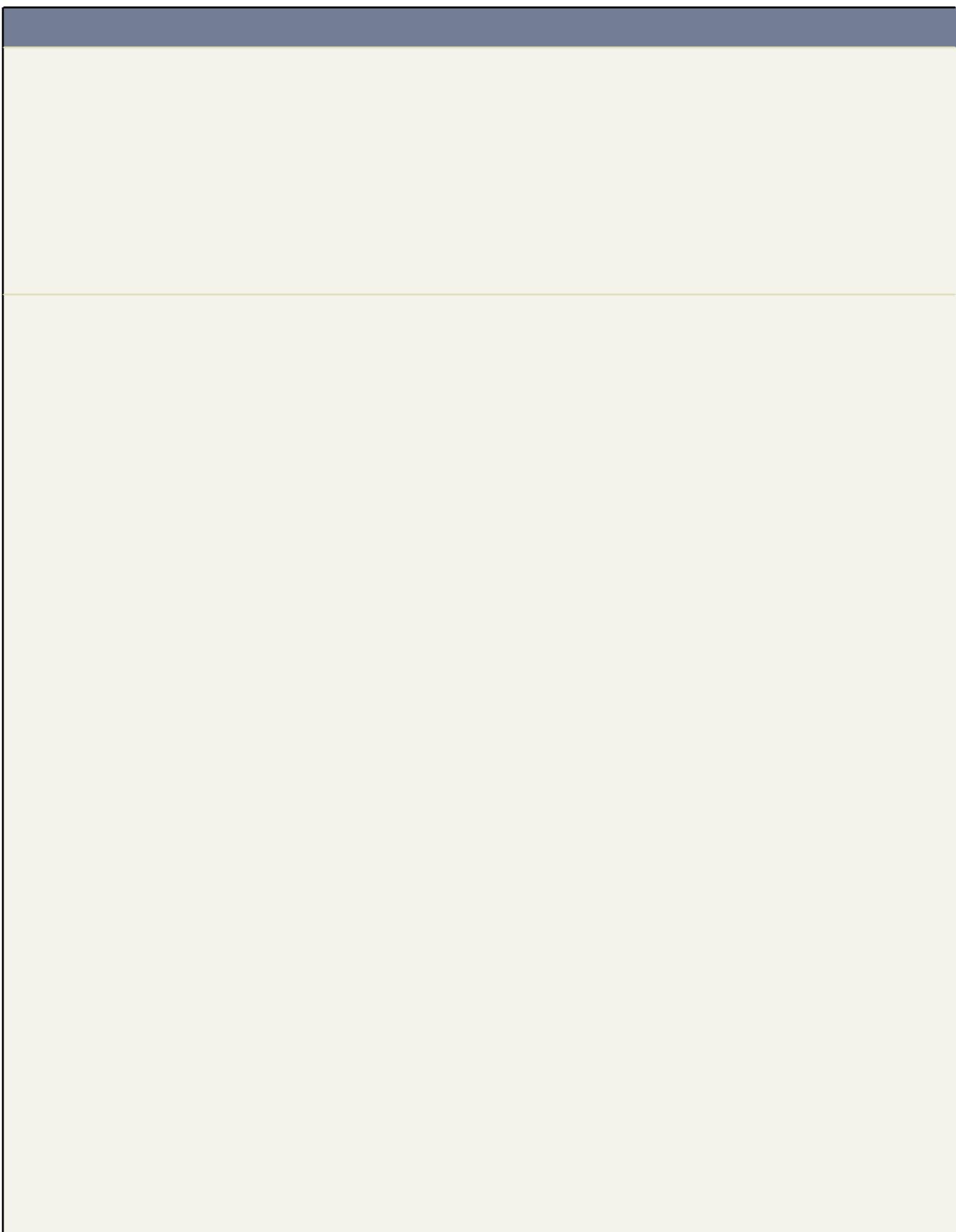
 メモ: *opt\_preserve\_id* はプリミティブデータ型のリストではなく、sObject のリストにのみ動作します。

(省略可能) *opt\_preserve\_id* 引数は、元のオブジェクトの ID を重複で保持するか削除するかを指定します。に設定すると、ID はコピーされたオブジェクトにコピーされます。デフォルトはであるため、ID はクリアされます。

名前	引数	戻り値	説明
			<p> メモ: Salesforce.com API バージョン 22.0 以前を使用して保存された Apex の場合、<code>opt_preserve_id</code> 引数のデフォルト値は のため、ID は保持されます。</p> <p>(省略可能)</p> <p><code>opt_preserve_readonly_timestamps</code> 引数は、参照のみのタイムスタンプとユーザ ID 項目を重複で保持するか削除するかを指定します。 に設定すると、参照のみの項目 、 、 および は、コピーされたオブジェクトにコピーされます。デフォルトは であるため、値はクリアされます。</p> <p>(省略可能) <code>opt_preserve_autonumber</code> 引数は、元のオブジェクトの自動採番項目を重複で保持するか削除するかを指定します。 に設定すると、自動採番項目はコピーされたオブジェクトにコピーされます。デフォルトは であるため、自動採番項目はクリアされます。</p> <p>この例は上記の例に基づいて、保持された参照のみのタイムスタンプとユーザ ID 項目を使用してリストをコピーする方法を示します。</p>

名前	引数	戻り値	説明
			含まれる sObject レコードが重複しないようにしてリストの浅いコピーを作成するには、メソッドを使用します。
	Integer <i>i</i>	配列要素	インデックス位置 <i>i</i> に保存されたリスト要素を返します。次に例を示します。

プリミティブデータ型または sObject の一次元リストの要素を参照するには、リスト名の後に要素のインデックス位置を大かっこで囲



名前	引数	戻り値	説明
			<p>このメソッドは sObject で構成されているリストでのみ使用できます。</p> <p>詳細は、「<a href="#">Apex Describe Information について</a>」(ページ 251)を参照してください。</p>
	boolean		<p>リストの要素が 0 の場合、true を返します。</p>
	Iterator		<p><a href="#">イテレータ</a>のインスタンスを返します。イテレータから反復可能なメソッド および を使用して、リスト内で反復できます。次に例を示します。</p>

名前	引数	戻り値	説明
Integer $i$	配列要素		<p>リストの <math>i</math> 番目のインデックス位置に保存された要素を削除し、削除された要素を返します。次に例を示します。</p> <p> メモ: リストでメソッドを使用するためにインターフェースを実装する必要はありません。</p>

名前	引数	戻り値	説明
	Integer $i$ 任意の型 $e$	Void	$e$ をリストのインデックス $i$ の位置に割り当てます。次に例を示します。
			プリミティブデータ型または sObject の一次元リストの要素を設定するには、リスト名の後に要素のインデックス位置を大括弧で囲んで表記することもできます。次に例を示します。
		Integer	リストの要素の数を返します。次に例を示します。
		Void	リスト内の項目を昇順で並び替えます。 次の例で、リストには3つの要素があります。 リストを並び替えると、最初の要素には値が

名前	引数	戻り値	説明
			割り当てられていないため null、2番目の要素の値は 5 になります。

 メモ: このメソッドを使用して、プリミティブ型、SelectOption 要素、sObject (標準オブジェクトとカスタムオブジェクト) を並び替えできます。sObject に使用される並び替え順についての詳細は、「[リストの並び替え](#)」を参照してください。カスタムデータ型 (Apex クラス) がインターフェースを実装している場合は、カスタムデータ型も並び替えできます。

List についての詳細は、「[List](#)」(ページ 39)を参照してください。

## Map メソッド

Map メソッドはすべてインスタンスマソッドで、対応付けの特定のインスタンスで動作します。次に、Map のインスタンスマソッドを示します。



メモ:

- 次の表では、*Key\_type* と *Value\_type* はそれぞれ対応付けのキーと値のデータ型を表し、データ型にはプリミティブ型、コレクション、sObjects、ユーザ定義型、組み込み Apex 型のいずれかを使用できます。

- 対応付けのユーザ定義型キーの一意性は、クラスで提供するメソッドによって判断されます。sObject キーなど、その他のすべての非プリミティブ型のキーの一意性は、オブジェクト項目の値の比較によって判断されます。
- String 型の対応付けキーでは、大文字と小文字が区別されます。大文字と小文字のみが異なる 2 つのキーは一意であるとみなされ、それぞれに別個の対応付けエントリがあります。したがって、`remove()`、`put()`、`putAll()` および `get()` などの Map メソッドでは、これらのキーが別個のものとして処理されます。

名前	引数	戻り値	説明
		Void	対応付けからすべてのキーと値の対応付けを削除します。
	map (同じデータ型)	対応付けの重複コピーを作成します。 これが sObject レコード値の対応付けである場合、重複対応付けは対応付けの浅いコピーとなります。つまり、重複には各 sObject レコードへの参照がありますが、レコード自体は重複しません。次に例を示します。	

名前	引数	戻り値	説明
			sObject レコードもコピーするには、メソッドを使用する必要があります。
	key type <i>key</i>	boolean	<p>指定された <i>key</i> の対応付けが含まれている場合は true を返します。</p> <p>キーが String の場合、キーの値では大文字と小文字が区別されます。</p> <p>次に例を示します。</p>
	map (同じデータ型)	sObject	sObject レコード値との対応付けの場合、sObject レコードを含む、対応付けの重複コピーを作成します。次に例を示します。

名前	引数	戻り値	説明
	key type <i>key</i>		<p>含まれる sObject レコードが重複しないようにして、対応付けの浅いコピーを作成するには、 <code>new</code> メソッドを使用します。</p> <p>対応付けにこのキーの値が含まれていない場合、指定した <i>key</i> が対応付けられている値または <code>null</code> を返します。</p> <p>キーが String の場合、キーの値では大文字と小文字が区別されます。</p> <p>次に例を示します。</p>

名前	引数	戻り値	説明
		Schema.SObjectType	Map 値を構成する sObject データ型のトークンを返します。Describe Information と共に使用して、対応付けに特定のデータ型の sObject を含めるかどうかを指定します。次に例を示します。

名前	引数	戻り値	説明
			<p>このメソッドはsObject 値を持つ対応付けでのみ使用できます。</p> <p>詳細は、「<a href="#">Apex Describe Informationについて</a>」(ページ 251)を参照してください。</p>
	boolean		<p>対応付けのキーと値のペアが 0 の場合、true を返します。次に例を示します。</p>
		<i>Key_type</i> のセット	<p>対応付けのすべてのキーを含むセットを返します。次に例を示します。</p>
	Key <i>key</i> , Value <i>value</i>	<i>Value_type</i>	<p>指定した <i>value</i> を対応付けで指定した <i>key</i> に関連付けています。対応付けにこのキーの対応付けが以前含まれていた場合、以前の値はメソッドで返され、その後置き換えられます。</p> <p>キーが String の場合、キーの値では大文字と小文字が区別されます。</p> <p>次に例を示します。</p>



名前	引数	戻り値	説明
			<p><i>Value_type</i> のリ　対応付けのすべての値を含むリストを順不同で返します。次に例を示します。</p>

Mapについての詳細は、「[対応付け](#)」(ページ 49)を参照してください。

## Set メソッド

Set メソッドは、キーワードを使用して初期化された要素の順序なしのコレクション、つまりセットで機能します。セットには一意の要素のみが含まれ、重複する値は含まれません。セットの要素には、プリミティブ型、コレクション型、sObject 型、ユーザ定義型、組み込み Apex 型のいずれかのデータ型を使用できます。Set メソッドはすべてインスタンスマソッドです。つまり、Set の特定のインスタンスで動作します。次に、Set のインスタンスマソッドを示します。



### メモ:

- 次の表では、*Set\_elem* はセットで 1 つの要素を表します。
- セットの要素の一意性は、クラスで提供する [メソッド](#)と [メソッド](#)によって判断されます。その他すべての非プリミティブ型の一意性は、オブジェクトの項目の比較によって判断されます。
- セットに String 要素が含まれる場合、要素では大文字と小文字が区別されます。大文字と小文字のみが異なる 2 つの要素は、別個のものとみなされます。

名前	引数	戻り値	説明
	Set element $e$	boolean	<p>要素がセットに追加されていない場合は、追加します。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、true を返します。次に例を示します。</p>
	List $l$	boolean	<p>指定されたリストのすべての要素がセットに追加されていない場合は、追加します。このメソッドは、リストとセットの結合を生成します。リストは、メソッドをコールするセットと同じデータ型である必要があります。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、true を返します。</p>
	Set $s$	boolean	<p>指定されたセットのすべての要素が、メソッドをコールするセットに追加されていない場合は、追加します。このメソッドは、2つのセットの結合を生成します。指定されたセットは、メソッドをコールする元のセットと同じデータ型である必要があります。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、true を返します。次に例を示します。</p>

名前	引数	戻り値	説明
		Void	セットからすべての要素を削除します。
		set (同じデータ型)	セットの重複コピーを作成します。
	Set element $e$	boolean	セットに指定した要素が存在する場合、 $\text{true}$ を返します。次に例を示します。
	List $l$	boolean	セットに指定したリストのすべての要素がある場合、 $\text{true}$ を返します。リストは、メソッドをコールするセットと同じデータ型である必要があります。
	Set $s$	boolean	セットに指定したセットのすべての要素が含まれる場合、 $\text{true}$ を返します。指定されたセットは、メソッドをコールする元のセットと同じデータ型である必要があります。次に例を示します。

名前	引数	戻り値	説明
		boolean	セットの要素が0の場合、 <code>false</code> を返します。次に例を示します。
<code>Set Element&lt;e&gt;</code>		boolean	<p>指定した要素がセットにある場合は、セットから削除します。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、<code>false</code> を返します。</p>
<code>List&lt;l&gt;</code>		boolean	<p>指定したリストの要素がセットにある場合は、セットから削除します。このメソッドは、2つのセットの相対補数を生成します。リストは、メソッドをコールするセットと同じデータ型である必要があります。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、<code>false</code> を返します。次に例を示します。</p>
<code>Set&lt;s&gt;</code>		boolean	<p>指定したセットの要素が元のセットにある場合は、削除します。このメソッドは、2つのセットの相対補数を生成します。指定されたセットは、メソッドをコールするセットと同じデータ型である必要があります。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、<code>false</code> を返します。次に例を示します。</p>

名前	引数	戻り値	説明
			<p>ドをコールする元のセットと同じデータ型である必要があります。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、<code>Set</code> を返します。</p>
	List <i>l</i>	boolean	<p>指定したリストに含まれるこのセットの要素のみを保持します。このメソッドは、リストとセットの交差を生成します。リストは、メソッドをコールするセットと同じデータ型である必要があります。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、<code>Set</code> を返します。次に例を示します。</p>
	Set <i>s</i>	boolean	<p>指定したセットに含まれる元のセットの要素のみを保持します。このメソッドは、2つのセットの交差を生成します。指定されたセットは、メソッドをコールする元のセットと同じデータ型である必要があります。</p> <p>このメソッドは、元のセットがコールの結果として変更された場合、<code>Set</code> を返します。</p>
		Integer	セットの要素の数(基数)を返します。次に例を示します。

名前	引数	戻り値	説明

Setについての詳細は、「[Set](#)」(ページ 47)を参照してください。

## Enum メソッド

enum 値にはユーザ定義のメソッドを追加できませんが、システム enum 値を含むすべての enum 値では、Apex に次のメソッドが定義されています。

名前	戻り値	説明
	string	enum 項目の名前を string として返します。
	Integer	enum 値のリスト内の項目の位置を返します。0 から始めます。

また、enum には次のメソッドがあります。

名前	戻り値	説明
	list<Enum type>	enum の値を、同じ enum 型のリストとして返します。

次に例を示します。


Enumについての詳細は、「[Enum](#)」(ページ 56)を参照してください。

## Apex sObject メソッド

用語 *sObject* は、Salesforce プラットフォームデータベースに保存できるオブジェクトを意味します。次の Apex sObject メソッドには、すべての sObject で使用できるメソッド、および sObject 構造の説明に使用する一般的なクラスが含まれます。

- [Schema](#)
- [sObject](#)
- [sObject Describe Result](#)
- [Field Describe Results](#)
- [Schema.FieldSet メソッド](#)
- [Custom Settings](#)

### Schema メソッド

次の表に、Schema のシステムメソッドの一覧を示します。

名前	引数	戻り値	説明
		Map<String, Schema.SObjectType>	すべての sObject 名(キー)の対応付けを、組織で定義された標準オブジェクトおよびカスタムオブジェクトの sObject トークン(値)に返します。次に例を示します。
			詳細は、「 <a href="#">すべての sObject へのアクセス</a> 」(ページ 255) を参照してください。
		List<Schema.Describe List<sObjectName> DataCategoryGroupResult>	指定したオブジェクトに関するカテゴリーグループのリストを返します。次の sObjectName のいずれかを指定できます。 <ul style="list-style-type: none"> <li>• <a href="#">記事タイプ</a>に関するカテゴリーグループを取得します。</li> </ul>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li><code>getCategoryGroups(sObject)</code>: 質問タイプに関連するカテゴリグループを取得します。</li> </ul> <p><code>describeDataCategoryGroups(sObject)</code> の使用についての詳細およびコード例は、「<a href="#">sObject に関連付けられたすべてのデータカテゴリへのアクセス</a>」を参照してください。</p> <p>記事および質問についての詳細は、Salesforce オンラインヘルプの「<a href="#">記事と翻訳の管理</a>」および「<a href="#">アンサーの概要</a>」を参照してください。</p>
	<code>pairs, topCategoriesOnly</code>	<code>List&lt;Schema.DescribeDataCategoryGroupStructureResult&gt;</code>	<p>要求で指定されたオブジェクトのデータカテゴリ構造と共に使用できるカテゴリグループを返します。</p> <p><code>describeDataCategoryGroupStructures(sObject)</code> の使用についての詳細およびコード例は、「<a href="#">sObject に関連付けられたすべてのデータカテゴリへのアクセス</a>」を参照してください。</p>

### Describe Data Category Group Structure 引数

`describeDataCategoryGroupStructures` メソッドは、データカテゴリ構造と共に使用できるカテゴリグループを返します。次に、このメソッドの引数を示します。

名前	戻り値	説明
<code>pairs</code>	<code>List&lt;Schema.DataCategoryGroupSObjectTypePair&gt;</code>	<p><code>Schema.DataCategoryGroupSObjectTypePair</code> をクエリする1つ以上のカテゴリグループおよびオブジェクトを指定します。指定されたオブジェクトの表示可能なデータカテゴリが取得されます。</p> <p>データカテゴリグループ表示設定についての詳細は、Salesforce オンラインヘルプの「<a href="#">カテゴリグループ表示設定について</a>」を参照してください。</p>

名前	戻り値	説明
topCategoriesOnly	boolean	を指定して、オブジェクトを分類する上位表示カテゴリのみを返します。 を指定して、表示できるすべての親カテゴリおよび子カテゴリを返します。両方の値は、ユーザのデータカテゴリグループ表示設定に応じて異なります。データカテゴリグループ表示設定についての詳細は、Salesforce オンラインヘルプの「カテゴリグループ表示設定について」を参照してください。

### Schema.DataCategoryGroupSobjectTypePair オブジェクト

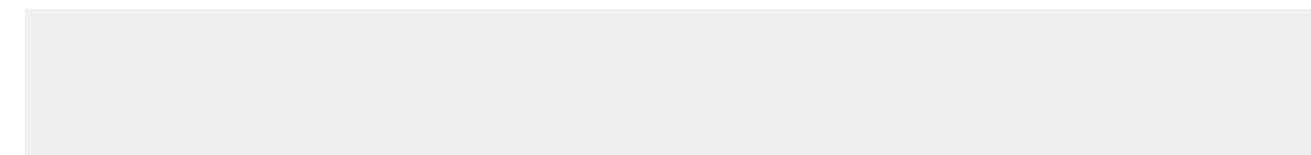
Schema.DataCategoryGroupSobjectTypePair は、カテゴリグループと関連オブジェクトを指定します。 describeDataCategoryGroupStructures メソッドを使用して、このオブジェクトに使用できるカテゴリを返します。次の表に、 Schema.DataCategoryGroupSobjectTypePair のすべてのメソッドの一覧を示します。

名前	引数	戻り値	説明
		string	データカテゴリグループにアクセスするために API で使用される一意の名前を返します。
		string	データカテゴリグループに関連付けられたオブジェクト名を返します。
		string	データカテゴリグループにアクセスするために API で使用される一意の名前を指定します。
	<i>String sObjectName</i>	Void	<i>sObjectName</i> はデータカテゴリグループに関連付けられたオブジェクト名です。有効な値は、次のとおりです。 <ul style="list-style-type: none"> <li>: 記事タイプの場合。</li> <li>: アンサーの質問の場合。</li> </ul>

### Schema.DescribeDataCategoryGroupResult オブジェクト

メソッドは、指定したオブジェクトに関連付けられたカテゴリグループのリストを含む Schema.DescribeDataCategoryGroupResult オブジェクトを返します。

次に、 Data Category Group Describe Result オブジェクトをインスタンス化する方法の例を示します。





の使用についての詳細およびコード例は、[「sObject に関連付けられたすべてのデータカテゴリへのアクセス」](#)を参照してください。

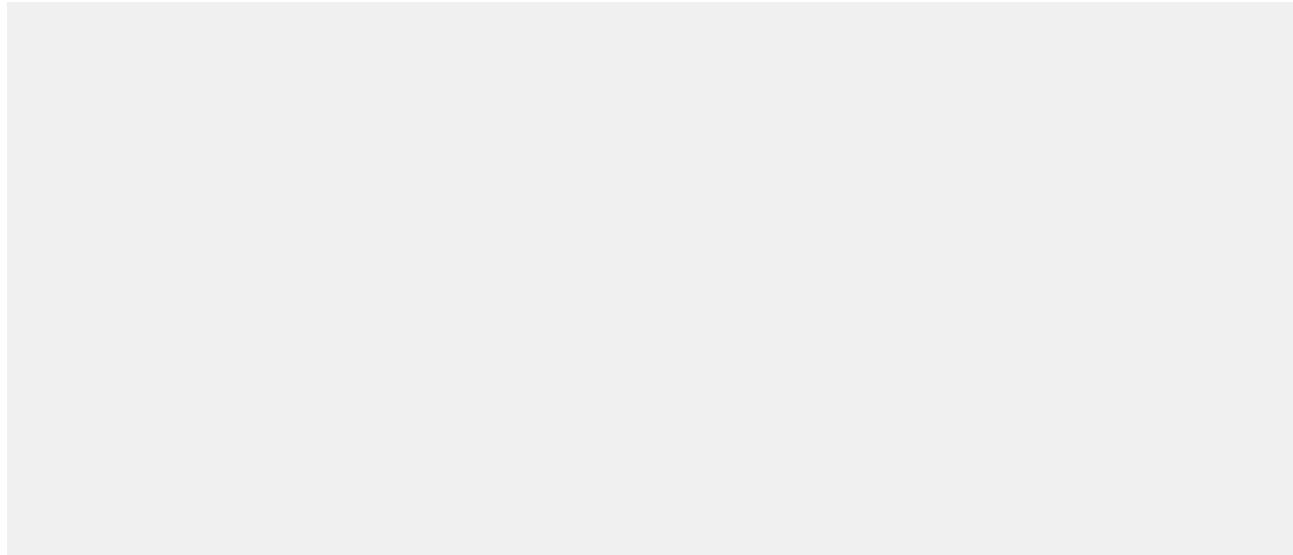
次の表に、Data Category Group Describe Result の一部として使用可能なメソッドの一覧を示します。引数を取るメソッドはありません。

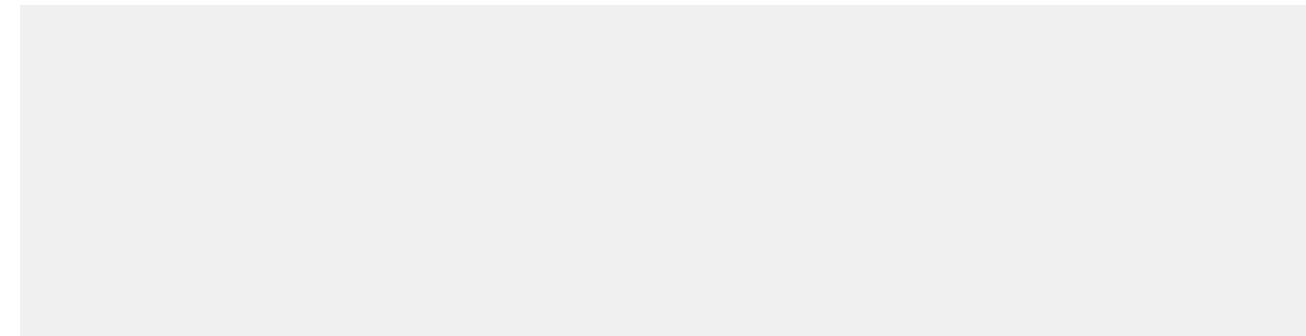
名前	戻り値	説明
	Integer	データカテゴリグループの表示可能なデータカテゴリ数を返します。
	string	データカテゴリグループの説明を返します。
	string	Salesforce ユーザインターフェースで使用されるデータカテゴリグループの表示ラベルを返します。
	string	データカテゴリグループにアクセスするために API で使用される一意の名前を返します。
	string	データカテゴリグループに関連付けられたオブジェクト名を返します。

#### Schema.DescribeDataCategoryGroupStructureResult オブジェクト

メソッドは、指定したオブジェクトに関連付けられたカテゴリグループおよびカテゴリを含む Schema.DescribeDataCategoryGroupStructureResult オブジェクトのリストを返します。

次に、Data Category Group Structure Describe Result オブジェクトをインスタンス化する方法の例を示します。





の使用についての詳細およびコード例は、[「sObject に関連付けられたすべてのデータカテゴリへのアクセス」](#)を参照してください。

次の表に、Data Category Group Structure Describe Result の一部として使用可能なメソッドの一覧を示します。引数を取るメソッドはありません。

名前	戻り値	説明
	string	データカテゴリグループの説明を返します。
	string	Salesforce ユーザインターフェースで使用されるデータカテゴリグループの表示ラベルを返します。
	string	データカテゴリグループにアクセスするために API で使用される一意の名前を返します。
	string	データカテゴリグループに関連付けられたオブジェクト名を返します。
	List<Schema.DataCategory>	ユーザのデータカテゴリグループ表示設定に基づいて表示可能な上位カテゴリを含む Schema.DataCategory オブジェクトを返します。データカテゴリグループ表示設定についての詳細は、Salesforce オンラインヘルプの「カテゴリグループ表示設定について」を参照してください。

### Schema.DataCategory オブジェクト

Schema.DataCategory オブジェクトはカテゴリグループ内のカテゴリを表します。Schema.DataCategory オブジェクトは [メソッド](#)によって返されます。次の表に、Schema.DataCategory オブジェクトのすべてのメソッドの一覧を示します。引数を取るメソッドはありません。

名前	戻り値	説明
	List<Schema.DataCategory>	データカテゴリで表示可能なサブカテゴリを含む再帰的オブジェクトを返します。
	string	Salesforce ユーザインターフェースで使用されるデータカテゴリの表示ラベルを返します。

名前	戻り値	説明
	string	データカテゴリにアクセスするために API で使用される一意の名前を返します。

## sObject メソッド

`sObject` メソッドはすべてインスタンスマソッドです。つまり、取引先または取引先責任者など、`sObject` の特定のインスタンスでコールされ、動作します。次に、`sObject` のインスタンスマソッドを示します。

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっている場合、このメソッドでは、Apex のすべてのバージョンに対して、指定されたエラーメッセージのすべての HTML マークアップがエスケープされます。</li> <li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっていない場合、このメソッドでは、Salesforce.com API バージョン 27.0 以降を使用して保存された Apex に対してのみ、エラーメッセージ内のすべての HTML マークアップがエスケープされます。</li> </ul>
			Visualforce コントローラで使用すると、生成されたメッセージが、そのページのエラーコレクションに追加されます。詳細は、『Visualforce 開発者ガイド』の「 <a href="#">入力規則と標準コントローラ</a> 」を参照してください。
	String <i>errorMsg</i> Boolean <i>escape</i>	Void	<p>カスタムエラーメッセージを使用してコードにマークを付け、エラーメッセージをエスケープする必要があるかどうかを指定して、DML 操作が行われないようにします。</p> <p><i>errorMsg</i> 引数は、レコードにマークを付けるエラーメッセージです。</p> <p><i>escape</i> 引数は、カスタムエラーメッセージ内の HTML マークアップがエスケープされるか（ ）、否か（ ）を示します。</p> <p>エスケープされる文字は、`&lt;`、`&gt;`、`&amp;lt;`、`&amp;gt;`、`&amp;#39;`、`&amp;#34;`、および`&amp;nbsp;`です。これらの文字がエスケープ</p>

名前	引数	戻り値	説明
			<p>されるため、HTML マークアップはレンダリングされず、Salesforce ユーザインター フェースでテキストとして表示されるようになります。</p> <p> 警告: <code>escape</code> 引数に _____ を指 定するときは、慎重に行ってください。Salesforce ユーザインター フェースに表示されるエスケープ 解除された文字列が、システムの 脆弱性を示す場合があります。そ れらの文字列に有害なコードが含 まれている可能性があるためで す。エラーメッセージに HTML マークアップを含める場合は、 <code>escape</code> 引数を使用してこの メソッドをコールし、入力項目値 などのすべての動的コンテンツを エスケープします。それ以外の場 合は、<code>escape</code> 引数に _____ を指定 するか、  <code>errorMsg</code> をコールします。</p>
Exception e		Void	<p>カスタムエラーメッセージでレコードを マークし、DML 操作が行われないように します。</p> <p><code>exception</code> 引数は、レコードにマークを 付けるエラーメッセージを含む <code>Exception</code> オブジェクトまたはカスタム例外オブジェ クトです。</p> <p>トリガおよび      トリガの _____ 、および      トリガの _____ で使用すると、アプリケーションインター フェースにエラーメッセージが表示され ます。</p> <p>「<a href="#">トリガ</a>」および「<a href="#">トリガの例外</a>」を参 照してください。</p> <p> メモ: このメソッドは、指定され たエラーメッセージ内のすべての</p>

名前	引数	戻り値	説明
			<p>HTML マークアップをエスケープします。エスケープされる文字は、&lt; &gt; &amp; &amp;#39; &amp;amp; #39; および &amp;lt; &amp;gt; です。これらの文字がエスケープされるため、HTML マークアップはレンダリングされず、Salesforce ユーザインターフェースでテキストとして表示されるようになります。HTML エスケープが実行されるバージョンは、組織で重要な更新が有効になっているかどうかによって異なります。</p> <ul style="list-style-type: none"><li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっている場合、このメソッドでは、Apex のすべてのバージョンに対して、指定されたエラーメッセージのすべての HTML マークアップがエスケープされます。</li><li>ApexaddError メソッドのデフォルト動作変更に関する重要な更新が組織で有効になっていない場合、このメソッドでは、Salesforce.com API バージョン 27.0 以降を使用して保存された Apex に対してのみ、エラーメッセージ内のすべての HTML マークアップがエスケープされます。</li></ul> <p>Visualforce コントローラで使用すると、生成されたメッセージが、そのページのエラーコレクションに追加されます。詳細は、『Visualforce 開発者ガイド』の「<a href="#">入力規則と標準コントローラ</a>」を参照してください。</p>



名前	引数	戻り値	説明
			<p>DML操作が行われないようにします。次に例を示します。</p> <p>注意:</p> <ul style="list-style-type: none"><li>トリガおよびトリガの、およびトリガで使用すると、アプリケーションインターフェースにエラーが表示されます。</li><li>Visualforce コントローラで使用すると、コンポーネントが項目に結合されている場合、コンポーネントにメッセージが添付されます。詳細は、『Visualforce 開発者ガイド』の「入力規則と標準コントローラ」を参照してください。</li><li>項目識別子は実際には呼び出しオブジェクトではなく、sObjectが呼び出し元であるため、このメソッドは専門分野に特化されます。項目を使用して、エラーの表示に使用する必要がある項目を識別します。</li><li>このメソッドは、Apex の今後のバージョンで変更される可能性があります。</li></ul> <p>「トリガ」および「トリガの例外」を参照してください。</p> <p> メモ: このメソッドは、指定されたエラーメッセージ内のすべてのHTMLマークアップをエスケープします。エスケープされる文字は、&lt; &gt; &amp; &amp; # &amp; # および、&amp; # です。これらの文字がエスケープされるため、HTMLマークアップはレンダリングされず、Salesforce ユーザインターフェースでテキストとして表示されるよう</p>



名前	引数	戻り値	説明
			<p>ダーリングされず、Salesforce ユーザインター フェースでテキストとして表示されるよ うになります。</p> <p> 警告: <code>escape</code> 引数に _____ を指 定するときは、慎重に行ってくだ さい。Salesforce ユーザインター フェースに表示されるエスケープ 解除された文字列が、システムの 脆弱性を示す場合があります。そ れらの文字列に有害なコードが含 まれている可能性があるためで す。エラーメッセージに HTML マークアップを含める場合は、 <code>escape</code> 引数を使用してこの メソッドをコールし、入力項目値 などのすべての動的コンテンツを エスケープします。それ以外の場 合は、<code>escape</code> 引数に _____ を指定 するか、<code>field</code> <code>errorMsg</code> をコールします。</p>
		Void	すべての項目値をクリアします。
	Boolean <code>opt_preserve_id</code> boolean <code>opt_IsDeepClone</code> boolean <code>opt_preserve_readonly_timestamps</code> boolean <code>opt_preserve_autonumber</code>	sObject (同じ データ型)	<p>sObject レコードのコピーを作成します。 (省略可能) <code>opt_preserve_id</code> 引数は、元 のオブジェクトの ID を重複で保持するか 削除するかを指定します。_____ に設定す ると、ID は重複する ID にコピーされま す。デフォルトは _____ であるため、ID はクリアされます。</p> <p> メモ: Salesforce.com API バージョ ン 22.0 以前を使用して保存された Apex の場合、<code>opt_preserve_id</code> 引数のデフォルト値は _____ のた め、ID は保持されます。</p> <p>(省略可能) <code>opt_IsDeepClone</code> 引数は、メ ソッドが sObject 項目の完全なコピーを作 成するか、参照を作成するかを決定しま す。</p>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>に設定すると、メソッドは sObject の完全版を作成します。リレーション項目など、sObject のすべての項目はメモリで重複します。その結果、コピーした sObject の項目に変更を行っても、元の sObject は影響されません。</li> <li>に設定すると、メソッドは sObject 項目の浅いコピーを作成します。コピーされたすべてのリレーション項目は元の sObject を使用します。その結果、コピーされた sObject でリレーション項目を変更すると、元の sObject の対応する項目も変更され、元の sObject で変更するとコピーされた sObject も変更されます。デフォルトは、です。</li> </ul> <p>(省略可能)  <code>opt_preserve_readonly_timestamps</code>引数は、参照のみのタイムスタンプ項目を重複で保持するか削除するかを指定します。に設定すると、参照のみの項目、およびは重複項目にコピーされます。デフォルトはであるため、値はクリアされます。</p> <p>(省略可能)  <code>opt_preserve_autonumber</code>引数は、元のオブジェクトの自動採番項目を複製で保持するか削除するかを指定します。に設定すると、自動採番項目はコピーされたオブジェクトにコピーされます。デフォルトはであるため、自動採番項目はクリアされます。</p> <p>など、<code>fieldName</code> で指定された項目の値を返します。          詳細は、「<a href="#">動的 SOQL</a>」を参照してください。</p>

名前	引数	戻り値	説明
	Schema.sObjectField <i>Field</i>	Object	項目トークン <i>sObjectField</i> (など) で指定された項目の値を返します。 詳細は、「 <a href="#">動的 SOQL</a> 」を参照してください。
	<a href="#">Database.DMLOptions</a>	sObject の database.DMLOptions オブジェクトを返します。 詳細は、「 <a href="#">Database.DMLOptions プロパティ</a> 」を参照してください。	
	String <i>fieldName</i>	sObject	<i>fieldName</i> で指定された項目の値を返します。このメソッドは主に、外部IDの値にアクセスするために動的 DML と共に使用します。 詳細は、「 <a href="#">動的 DML</a> 」を参照してください。
	Schema.SObjectField <i>fieldName</i> sObject	項目トークン <i>fieldName</i> (など) で指定された項目の値を返します。このメソッドは主に、外部IDの値にアクセスするために動的 DML と共に使用します。 詳細は、「 <a href="#">動的 DML</a> 」を参照してください。	
	String <i>fieldName</i>	sObject[]	<i>fieldName</i> で指定された項目の値を返します。このメソッドは主に、子リレーションなど、関連オブジェクトの値にアクセスするために動的 DML と共に使用します。 詳細は、「 <a href="#">動的 DML</a> 」を参照してください。
	Schema.SObjectType <i>fieldName</i> sObject[]	項目トークン <i>fieldName</i> (など) で指定された項目の値を返します。このメソッドは主に、子リレーションなど、関連オブジェクトの値にアクセスするために動的 DML と共に使用します。	

名前	引数	戻り値	説明
			詳細は、「 <a href="#">動的 DML</a> 」を参照してください。
		Schema.SObjectType	この sObject の token を返します。このメソッドは Describe Information で使用されます。
			詳細は、「 <a href="#">Apex Describe Information について</a> 」を参照してください。
		string	この sObject に関連付けられたパブリックアクションの名前を取得します。多くの場合、トリガで使用されます。
	String <i>fieldName</i> Object <i>value</i>	Object	<i>fieldName</i> で指定された項目の値を設定し、項目の以前の値を返します。
			詳細は、「 <a href="#">動的 SOQL</a> 」を参照してください。
	Schema.SObjectField <i>fieldName</i> Object <i>value</i>	項目 token ( など)	<i>fieldName</i> で指定された項目の値を設定し、項目の以前の値を返します。
			詳細は、「 <a href="#">動的 SOQL</a> 」を参照してください。
	String <i>fieldName</i> sObject <i>value</i>	sObject	<i>fieldName</i> で指定された項目の値を設定します。このメソッドは主に、外部 ID の値に設定するために動的 DML で使用します。メソッドは項目の以前の値を返します。
			詳細は、「 <a href="#">動的 SOQL</a> 」を参照してください。
	Schema.sObjectType <i>fieldName</i> sObject <i>value</i>	トークン で指定される項目の値を設定します。このメソッドは主に、外部 ID の値に設定するために動的 DML で使用します。メソッドは項目の以前の値を返します。	
			詳細は、「 <a href="#">動的 SOQL</a> 」を参照してください。

名前	引数	戻り値	説明
	database.DMLOptions <i>DMLOptions</i>	Void	sObject の DMLOptions オブジェクトを設定します。 詳細は、「 <a href="#">Database.DMLOptions プロパティ</a> 」を参照してください。

sObject についての詳細は、「[sObject 型](#)」(ページ 33)を参照してください。

## sObject Describe Result メソッド

次の表は、sObject Describe Result に使用可能なメソッドである DescribeSObjectResult オブジェクトを示しています。引数を取るメソッドはありません。

名前	データ型	説明
	special	単独で使用してはいけない特殊なデータ型を返します。には、項目メンバー変数名または メソッドのいずれかが常に必要です。次に例を示します。   詳細は、「 <a href="#">Apex Describe Information について</a> 」を参照してください。
	special	単独で使用してはいけない特殊なデータ型を返します。の後には常に、項目セット名または メソッドを続ける必要があります。次に例を示します。
	List<Schema.ChildRelationship>	定義される sObject の外部キーがある sObject の名前である、子リレーションのリストを返します。たとえば、Account オブジェクトには、子

名前	データ型	説明
		リレーションとして _____ と が含まれます。
	String	オブジェクトの3文字のプレフィックスコード を返します。レコード ID は、プレフィックス としてオブジェクトデータ型を指定する3文字 コードが付きます(たとえば、取引先には のプレフィックス、商談には _____ のプレフィッ クスが付きます)。  DescribeSobjectResult オブジェクトは、安定し たプレフィックスを持つオブジェクトに値を返 します。安定したプレフィックスまたは予測可 能なプレフィックスを持たないオブジェクト データ型については、項目は空白です。これら のコードに依存するクライアントアプリケー ションは、前方互換性を確保するために、この オブジェクトデータ型を決定する方法を使 用できます。
	String	オブジェクト名と一致または一致しないオブ ジェクトの表示ラベルを返します。たとえば、 医療分野の組織では、Account の表示ラベルを Patient に変更する可能性があります。この表示 ラベルは、その後 Salesforce ユーザインター フェースで使用されます。詳細は、Salesforce オンラインヘルプを参照してください。
	String	オブジェクト名と一致または一致しないオブ ジェクトの複数の表示ラベルを返します。たと えば、医療分野の組織では、Account の複数の 表示ラベルを Patient に変更する可能性あり ます。この表示ラベルは、その後 Salesforce ユー ザインターフェースで使用されます。詳細は、 Salesforce オンラインヘルプを参照してく ださい。
	String	メソッドと同様、オブジェクト名を返 します。ただし、オブジェクトが現在の名前空 間の一部である場合、名前の名前空間部分は削 除されます。
	String	オブジェクトの名前を返します。
	List<Schema.RecordTypeInfo>	このオブジェクトがサポートするレコードタ イプのリストを返します。このリスト内で参照す

名前	データ型	説明
		るには、現在のユーザにレコードタイプへのアクセス権は必要ありません。
	Map<ID, Schema.RecordTypeInfo>	関連付けられたレコードタイプにレコード ID を照合する対応付けを返します。この対応付け内で参照するには、現在のユーザにレコードタイプへのアクセス権は必要ありません。
	Map<String, Schema.RecordTypeInfo>	関連付けられたレコードタイプにレコードラベルを照合する対応付けを返します。この対応付け内で参照するには、現在のユーザにレコードタイプへのアクセス権は必要ありません。
	Schema.SObjectType	sObject の Schema.SObjectType オブジェクトを返します。類似した sObject の作成に使用できます。詳細は、「 <a href="#">Schema.sObjectType</a> 」を参照してください。
	Boolean	現在のユーザがこの項目を参照できる場合は <code>true</code> 、できない場合は <code>false</code> を返します。
	Boolean	現在のユーザがオブジェクトを作成できる場合は <code>true</code> 、できない場合は <code>false</code> を返します。
	Boolean	項目がカスタムオブジェクトの場合は <code>true</code> 、標準オブジェクトの場合は <code>false</code> を返します。
	Boolean	オブジェクトがカスタム設定の場合は <code>true</code> 、そうでない場合は <code>false</code> を返します。
	Boolean	現在のユーザがオブジェクトを削除できる場合は <code>true</code> 、できない場合は <code>false</code> を返します。
	Boolean	将来の使用のために予約されています。
	Boolean	そのオブジェクトで Chatter フィードが有効になっている場合は <code>true</code> 、有効でない場合は <code>false</code> を返します。このメソッドは、Salesforce.com API バージョン 19.0 以降を使用して保存された Apex クラスおよびトリガにのみ使用できます。
	Boolean	現在のユーザがオブジェクトを同じデータ型の他のオブジェクトとマージできる場合は <code>true</code> 、できない場合は <code>false</code> を返します。これは、リード、取引先責任者、および取引先に対して返されます。

名前	データ型	説明
	Boolean	現在のユーザがオブジェクトをクエリできる場合は <code>true</code> 、クエリできない場合は <code>false</code> を返します。
	Boolean	現在のユーザがオブジェクトを検索できる場合は <code>true</code> 、できない場合は <code>false</code> を返します。
	Boolean	現在のユーザがオブジェクトを復元できる場合は <code>true</code> 、できない場合は <code>false</code> を返します。
	Boolean	現在のユーザがオブジェクトを更新できる場合は <code>true</code> 、できない場合は <code>false</code> を返します。

### ChildRelationship メソッド

sObject が親オブジェクトの場合、ChildRelationship オブジェクトメソッドを使用した子 sObject と同様に、子リレーションにアクセスできます。

ChildRelationship オブジェクトは、  
メソッドを使用して sObject describe result から返されます。次に例を示します。

Apex 要求ごとに 100 件の  
メソッドコールの使用のみできます。ガバナ制限について  
の詳細は、「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

次の表に、ChildRelationship オブジェクトの一部として使用可能なメソッドを示します。引数を取るメソッドはありません。

名前	データ型	説明
	Schema.SObjectType	親 sObject への外部キーを持つ子 sObject のトークンを返します。
	Schema.SObjectField	親 sObject への外部キーを持つ項目のトークンを返します。
	String	リレーションの名前を返します。
	Boolean	親オブジェクトの削除時に子オブジェクトが削除される場合は <code>true</code> 、削除されない場合は <code>false</code> を返します。
	Boolean	将来の使用のために予約されています。
	Boolean	子オブジェクトから参照されるため親オブジェクトを削除できない場合は <code>false</code> を返し、削除できる場合は <code>true</code> を返します。

## RecordTypeInfo メソッド

sObject に自身に関連付けられたレコードタイプがある場合は、RecordTypeInfo オブジェクトメソッドを使用して、レコードタイプについての情報にアクセスできます。

RecordTypeInfo オブジェクトは、メソッドを使用して sObject Describe Result から返されます。次に例を示します。

メソッドだけでなく、

メソッドと

メソッドも使用できます。これらのメソッドはそれぞれ、RecordTypeInfo をレコード ID とレコードラベルに関連付ける対応付けを返します。

Apex 要求ごとに 100 件の RecordTypeInfo オブジェクトを返すことのみできます。ガバナ制限についての詳細は、「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

次の例では、少なくとも 1 つのレコードタイプが Account オブジェクト用に作成されています。

次の表に、RecordTypeInfo オブジェクトの一部として使用可能なメソッドを示します。引数を取るメソッドはありません。

名前	データ型	説明
	String	このレコードタイプの名前を返します。
	ID	このレコードタイプの ID を返します。
	Boolean	現在のユーザがレコードタイプを使用できる場合は <code>true</code> 、使用できない場合は <code>false</code> を返します。このメソッドは、新しいレコードの作成時に使用可能なレコードタイプの一覧をユーザに表示するために使用します。
	Boolean	これがデフォルトのレコードタイプ対応付けの場合は <code>true</code> 、そうでない場合は <code>false</code> を返します。

## Describe Field Result メソッド

次の表は、Field Describe Result の一部として使用可能なメソッドを示しています。次に、Field Describe Result オブジェクトをインスタンス化する方法の例を示します。

引数を取るメソッドはありません。

名前	データ型	説明
	Integer	可変長項目(バイナリ項目も含む)の最大サイズをバイトで返します。
	String	この項目に指定された数式を返します。
	Schema.sObjectField	制御項目のトークンを返します。
	Object	この項目のデフォルト値を返します。
	String	数式が使用されていない場合、この項目に指定されたデフォルト値を返します。
	Integer	項目に指定された桁の最大数を返します。このメソッドは、integer 項目でのみ有効です。
	String	項目レベルのヘルプの内容を返します。詳細は、Salesforce オンラインヘルプの「項目レベルのヘルプの定義」を参照してください。
	String	Salesforce ユーザインターフェースの項目の隣に表示されるテキストラベルを返します。このラベルはローカライズが可能です。
 <b>メモ:</b> 標準オブジェクトの [種別] 項目の場合、_____ はデフォルトのラベルとは異なるラベルを返します。これは、Object _____ という形式のラベルを返します。Object は標準オブジェクトラベルです。たとえば、[取引先] の [種別] 項目の場合、_____ はデフォルトラベルの _____ ではなく、_____ を返します。[種別] ラベルの名前が変更されると、_____ によって新しいラベルが返されます。[カスタマイズ] > [タブ名と表示ラベル] > [タブと表示ラベルの名称変更] を選択して、すべての標準オブジェクト項目のラベルを確認または変更できます。		

名前	データ型	説明
	Integer	文字列項目には、(バイトではなく)ユニコード文字での最大サイズを返します。
	String	メソッドと同様、項目名を返します。ただし、項目が現在の名前空間の一部である場合、名前の名前空間部分は削除されます。
	String	Apex に使用される項目名を返します。
	list <Schema.PicklistEntry>	PicklistEntry オブジェクトのリストを返します。項目が選択リストでない場合、ランタイムエラーを返します。
	Integer	データ型 double の項目には、小数点の右側と左側の両方をあわせた(ただし小数点自体は含まない)、格納可能な最大桁数を返します。
	list <Schema.sObjectType>	この項目の親オブジェクトの Schema.sObjectType オブジェクトのリストを返します。メソッドが を返す場合、リストには複数のエントリがあります。返さない場合、エントリは1つのみです。
	String	リレーションの名前を返します。リレーションとリレーション名についての詳細は、『Force.com SOQL and SOSL Reference』の「リレーション名について」を参照してください。
	Integer	項目が子の場合は 1、子でない場合は 0 を返します。リレーションとリレーション名についての詳細は、『Force.com SOQL and SOSL Reference』の「リレーション名について」を参照してください。
	Integer	データ型 double の項目には、小数点の右側の桁数を返します。小数点の右側に余分な桁がある場合は、切り捨てられます。小数点の左側の桁数が多すぎる場合は、このメソッドはエラー応答を返します。
	Schema.SOAPType	項目のデータ型に応じて、SoapType enum 値の 1 つを返します。詳細は、「Schema.SOAPType Enum 値」(ページ 548)を参照してください。
	Schema.sObjectField	この項目のトーカンを返します。
	Schema.DisplayType	項目のデータ型に応じて、DisplayType enum 値の 1 つを返します。詳細は、

名前	データ型	説明
	Boolean	「 <a href="#">Schema.DisplayType Enum 値</a> 」(ページ 543)を参照してください。
	Boolean	現在のユーザがこの項目を参照できる場合は、参照できない場合は を返します。
	Boolean	項目が Auto Number 項目の場合は 、そうでない場合は を返します。
	Boolean	SQL IDENTITY 型に似て、Auto Number 項目は参照のみ可能で、作成できない項目であり、最長 30 文字です。Auto Number 項目は、内部オブジェクト ID に依存しない一意な ID を提供します(購入注文番号や請求書番号など)。Auto Number 項目は、全体的に Salesforce ユーザインターフェースで構成されています。
	Boolean	項目がカスタム数式項目の場合は 、そうでない場合は を返します。カスタム数式項目は常に参照のみです。
	Boolean	親オブジェクトの削除時に子オブジェクトが削除される場合は 、削除されない場合は を返します。
	Boolean	項目が大文字と小文字を区別する場合は 、区別しない場合は を返します。
	Boolean	現在のユーザが項目が作成できる場合は 、できない場合は を返します。
	Boolean	項目がカスタム項目の場合は 、標準オブジェクトの場合は を返します。
	Boolean	作成時に項目がデフォルト値を受け取る場合は 、受け取らない場合は を返します。の場合、この項目の値が作成コールで渡されなくても、Salesforce はオブジェクト作成時にこの項目の値を暗黙的に割り当てます。たとえば、Opportunity オブジェクトでは値が Stage 項目から取得されているため、Probability 項目にはこの属性が指定されています。同様に、ほとんどのオブジェクトの Owner にはこの属性が設定されています。Owner 項目が特に指定されない限り、値は現在のユーザから取得されます。

名前	データ型	説明
	Boolean	選択リストが連動選択リストの場合は _____、 そうでない場合は _____ を返します。
	Boolean	将来の使用のために予約されています。
	Boolean	項目が外部 ID として使用されている場合は _____、 そうでない場合は _____ を返します。
	Boolean	項目を _____ ステートメントの検索条件の一部として使用できる場合は _____、 そうでない場合は _____ を返します。
	Boolean	項目が SOQL クエリの _____ 句に含まれる場合は _____、 含まれない場合は _____ を返します。このメソッドは、API バージョン 18.0 以降を使用して保存された Apex クラスおよびトリガにのみ使用できます。
	Boolean	項目が HTML のために形式化されており、 HTML として表示されるように符号化する必要がある場合は _____、 そうでない場合は _____ を返します。このメソッドに対して _____ を返す項目の例の1つは、ハイパーリンクのカスタム数式項目です。もう1つの例は、_____ テキスト関数があるカスタム数式項目です。
	Boolean	メソッドでレコードを指定するために 項目を使用できる場合は _____、 使用できない場合は _____ を返します。
	Boolean	項目が名前項目の場合は _____、 そうでない場合は _____ を返します。このメソッドは、標準オブジェクトの名前項目 (Account オブジェクトの _____ など) やカスタムオブジェクトの名前項目を識別するために使用します。 Contact オブジェクトのように _____ と 項目が代わりに使用される場合を除き、オブジェクトは名前項目を1つのみ持つことができます。
		個人取引先の _____ 項目などのように複合名が存在する場合、そのレコードの _____ は _____ に設定されます。
	Boolean	項目が複数のデータ型のオブジェクトを親として持つことが可能な場合、 _____ を返します。たとえば、1つの ToDo に 取引先責任者 リー

名前	データ型	説明
		ド ( ) 項目と 商談 取引先 ( ) 項目の両方が含まれ、このメソッドに対して を返すことができます。これは、どちらのオブジェクトも特定の ToDo レコードの親になることができるからです。それ以外の場合、このメソッドは を返します。
	Boolean	項目を空白にできる場合は 、できない場合は を返します。 null 値が許可される項目は、中身を空にすることができます。空白にできない項目では、オブジェクトを作成して保存するには必ず値を設定する必要があります。
	Boolean	項目に項目の権限を指定できる場合は 、そうでない場合は を返します。
	Boolean	子オブジェクトから参照されるため親オブジェクトを削除できない場合は を返し、削除できる場合は を返します。
	Boolean	項目が制限つき選択リストの場合は 、そうでない場合は を返します。
	Boolean	項目上でクエリをソートできる場合は 、できない場合は を返します。
	Boolean	項目の値を一意にする必要がある場合は 、そうでない場合は を返します。
	Boolean	次のいずれかの場合は を返します。 <ul style="list-style-type: none"><li>・ 現在のユーザが項目を編集できる</li><li>・ カスタムオブジェクトでの主従関係項目の子レコードの親を他の親レコードに変更できる</li></ul> 上記以外の場合は を返します。
	Boolean	詳細オブジェクトへの書き込みに親の参照・更新共有ではなく参照共有が必要な場合は、 を返します。

### Schema.DisplayType Enum 値

Schema.DisplayType Enum 値は、Field Describe Result の メソッドによって返されます。 詳細は、『Object Reference for Salesforce and Force.com』の「データ型」を参照してください。すべての enum で共有されるメソッドの詳細は、「Enum メソッド」(ページ 517)を参照してください。

データ型の項目値	Field オブジェクトに含まれる内容
	値のデータ型は、 、 、 、 、 、 、 、 、 、 、 、 、 、 または です。
	Base64 で符号化された任意のバイナリデータ (型は base64Binary)
	boolean の ( または ) の値
	列挙のセットを提供するコンボボックスで、ユーザはリストにない値も指定できます。
	通貨の値
	データカテゴリーグループまたはカテゴリの一意名への参照。
	日付の値
	日時値
	倍精度浮動小数点値
	メールアドレス
	暗号化された文字列
	オブジェクトの主キー項目
	整数値
	複数の値を選択可能な列挙のセットを提供する複数選択の選択リスト
	パーセント値
	電話番号。値には英字を含めることもできます。電話番号の書式は、クライアントアプリケーションが指定します。
	単一の値を選択可能な列挙のセットを含む、1つの値のみを選択できる選択リスト
	外部キー項目に似ている、別のオブジェクトへの相互参照
	文字列の値
	複数行のテキスト項目として表示される文字列値
	時間の値
	ハイパーリンクとして表示される URL 値

## Schema.PicklistEntry メソッド

picklist 項目には、ユーザが単一のデータを選択可能な 1 つ以上のデータのリストが含まれます。Salesforce ユーザインターフェースのドロップダウンリストとして表示されます。データの 1 つをデフォルトデータに設定できます。

Schema.PicklistEntry オブジェクトは、  
ます。次に例を示します。

メソッドを使用して Field Describe Result から返され

Apex 要求ごとに 100 件の メソッドコールの使用のみできます。ガバナ制限についての詳細  
は、「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

次の表は、PicklistEntry オブジェクトの一部として使用可能なメソッドを示しています。引数を取るメソッドはありません。

名前	データ型	説明
	String	選択リスト内のこのアイテムの表示名を返します。
	String	選択リスト内のこの項目の値を返します。
	Boolean	ユーザインターフェースの選択リスト項目のドロップダウンリストに項目を表示する必要がある場合は TRUE 、必要がない場合は FALSE を返します。
	Boolean	この項目が選択リスト用のデフォルト値の場合は TRUE 、そうでない場合は FALSE を返します。選択リスト内の 1 つのアイテムのみをデフォルトに設定できます。

### Schema.sObjectField

Schema.sObjectField オブジェクトは、 メソッドおよび メソッドを使用して  
Field Describe Result から返されます。次に例を示します。

次の表は、sObjectField オブジェクトの一部として使用可能なメソッドを示しています。このメソッドは引数を採用しません。

名前	データ型	説明
	Schema.DescribeFieldResult	この項目の Describe Field Result を返します。

### Schema.sObjectType

Schema.sObjectType オブジェクトは、 メソッドを使用して Field Describe Result から、  
メソッドを使用して sObject Describe Result から返されます。次に例を示します。

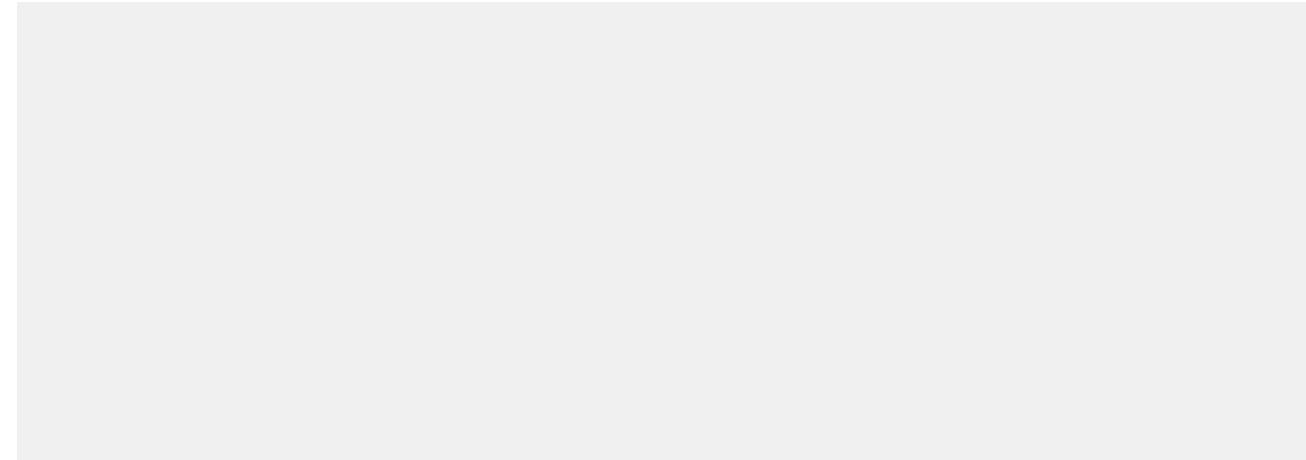
次の表は、sObjectType オブジェクトの一部として使用可能なメソッドを示しています。

名前	引数	データ型	説明
		Schema.DescribeSObjectResult	この項目の Describe SObject Result を返します。
	sObject		このデータ型の新しい sObject を構築します。 例は、「 <a href="#">sObject の動的作成</a> 」を参照してください。
	ID <i>Id</i>	sObject	指定された ID でこの型の新しい sObject を作成します。 引数として、データベースにある既存のレコードの ID を渡します。 新しい sObject を作成すると、返された sObject のすべての項目は _____ に設定されています。更新可能な項目を目的の値に設定し、データベースのレコードを更新できます。新しい値を設定した項目のみが更新され、それ以外のシステム項目ではないすべての項目は保持されます。
	ID <i>recordTypeId</i>	sObject	このデータ型の新しい sObject を構築します。必要に応じて、指定するレコードタイプ ID の sObject や、デフォルトのカスタム項目値を持つ sObject を作成できます。
	Boolean <i>loadDefaults</i>		<i>recordTypeId</i> 引数には、作成する sObject のレコードタイプ ID を指定します。この sObject にレコードタイプが存在しない場合、_____ を使用します。この sObject にレコードタイプが存在し、を指定した場合、デフォルトのレコードタイプが使用されます。 <i>loadDefaults</i> 引数には、定義済みのデフォルト値をカスタム項目に入力するか( )、しないか( )を指定します。
			備考: <ul style="list-style-type: none"><li>デフォルト値が存在しない必須項目には、新しい sObject を挿入する前に値を指定してください。値が指定されていない場合、挿入でエラーが発生します。たとえば、Account Name 項目や主従関係項目がこれに該当します。</li></ul>

名前	引数	データ型	説明
			<ul style="list-style-type: none"> <li>選択リストと複数選択リストではレコードタイプごとに特定のデフォルト値を使用できるため、このメソッドは指定されたレコードタイプに対応するデフォルト値を入力します。</li> <li>項目に定義済みのデフォルト値が存在せず、<code>loadDefaults</code> 引数が <code>      </code> の場合、このメソッドは項目値が <code>      </code> の sObject を作成します。</li> <li><code>loadDefaults</code> 引数が <code>      </code> の場合、このメソッドは項目値が <code>      </code> の sObject を作成します。</li> <li>このメソッドは、新しい sObject の参照のみのカスタム項目にデフォルト値を入力します。その後、これらの参照のみの項目を含む新しい sObject を挿入できますが、これらの項目は挿入後も編集できません。</li> <li>カスタム項目が一意としてマークされていてデフォルト値を指定した場合、複数の新しい sObject を挿入すると項目値が重複するため、実行時例外が発生します。</li> </ul> <p>デフォルト項目値についての詳細は、Salesforce オンラインヘルプの「デフォルト項目値について」を参照してください。</p> <p><a href="#">「サンプル: デフォルト値での新しい sObject の作成」</a> の例を参照してください。</p>

### サンプル: デフォルト値での新しい sObject の作成

このサンプルでは、 メソッドを使用して、カスタム項目にデフォルト値(ある場合)を入力した取引先を作成します。また、指定するレコードタイプで 2 つ目の取引先を作成します。新しい取引先を挿入する前に、両方の取引先に対して、デフォルト値のない必須項目である Name 項目を設定します。



### Schema.SOAPType Enum 値

schema.SOAPType enum 値は、Field Describe Result の メソッドによって返されます。

詳細は、『SOAP API Developer's Guide』の「SOAPTypes」を参照してください。すべての enum で共有されるメソッドの詳細は、Enum メソッド (ページ 517)を参照してください。

データ型の項目値	Field オブジェクトに含まれる内容
	値のデータ型は、 、 、 、 、 、 または です。
	Base64 で符号化された任意のバイナリデータ (型は base6Binary)
	boolean の ( または ) の値
	日付の値
	日時値
	倍精度浮動小数点値
	オブジェクトの主キー項目
	整数値
	文字列の値
	時間の値

### Schema.FieldSet メソッド

sObject に作成された項目セットの詳細を検出および取得するためのメソッドが含まれます。



メモ: このリリースには、項目セットのベータバージョンが含まれています。本番品質ではありますが、既知の制限があります。

## 使用方法

項目セットに含まれる項目を検出し、項目セット自体の詳細(名前、名前空間、表示ラベルなど)を取得するには、  
クラスのメソッドを使用します。次の例では、sObjectについて項目セットの Describe  
Result オブジェクトのコレクションを取得する方法を示します。返される Map のキーは項目セット名で、値は対  
応する項目セットの Describe Result です。

項目セットは sObject Describe Result からも使用できます。次のコード行は、前述のサンプルと同じです。

個々の項目セットを使用するには、sObject 上の項目セットのマップ経由でアクセスするか、事前に項目セット  
の名前がわかる場合は、項目セットへの明示的参照を使用してアクセスします。次の 2 行のコードは、同じ項目  
セットを取得します。

## メソッド

次に、  
クラスのインスタンスマソッドを示します。引数を取るメソッドはありません。

メソッド	戻り値	説明
		項目セットの説明を返します。 説明は項目セットの必須項目で、項目セットのコンテキストと コンテンツを説明するためのものです。多くの場合、エンド ユーザではなく、管理パッケージに定義された項目セットを設 定するシステム管理者を対象とします。
		項目セットを構成する項目の オブ ジェクトのリストを返します。
		Salesforce ユーザインターフェースの項目の隣に表示されるテキ ストラベルを返します。
		項目セットの名前を返します。
		項目セットの名前空間を返します。 組織で名前空間を設定しておらず、項目セットが組織で定義さ れている場合、返される名前空間は空の文字列です。それ以外

メソッド	戻り値	説明
		の場合、これは組織の名前空間か、項目セットが含まれる管理パッケージの名前空間です。

### Schema.FieldSetMember メソッド

項目セットのメンバー項目のメタデータにアクセスするためのメソッドが含まれます。

#### 使用方法

項目セットに含まれる項目の詳細 (項目表示ラベル、型、動的 SOQL 対応項目パスなど) を取得するには、クラスのメソッドを使用します。次の例では、sObject の特定の項目セットについて、項目セットメンバーの Describe Result オブジェクトのコレクションを取得する方法を示します。

項目セットの名前が事前にわかる場合、項目セットへの明示的な参照を使用して、より直接的に項目にアクセスできます。

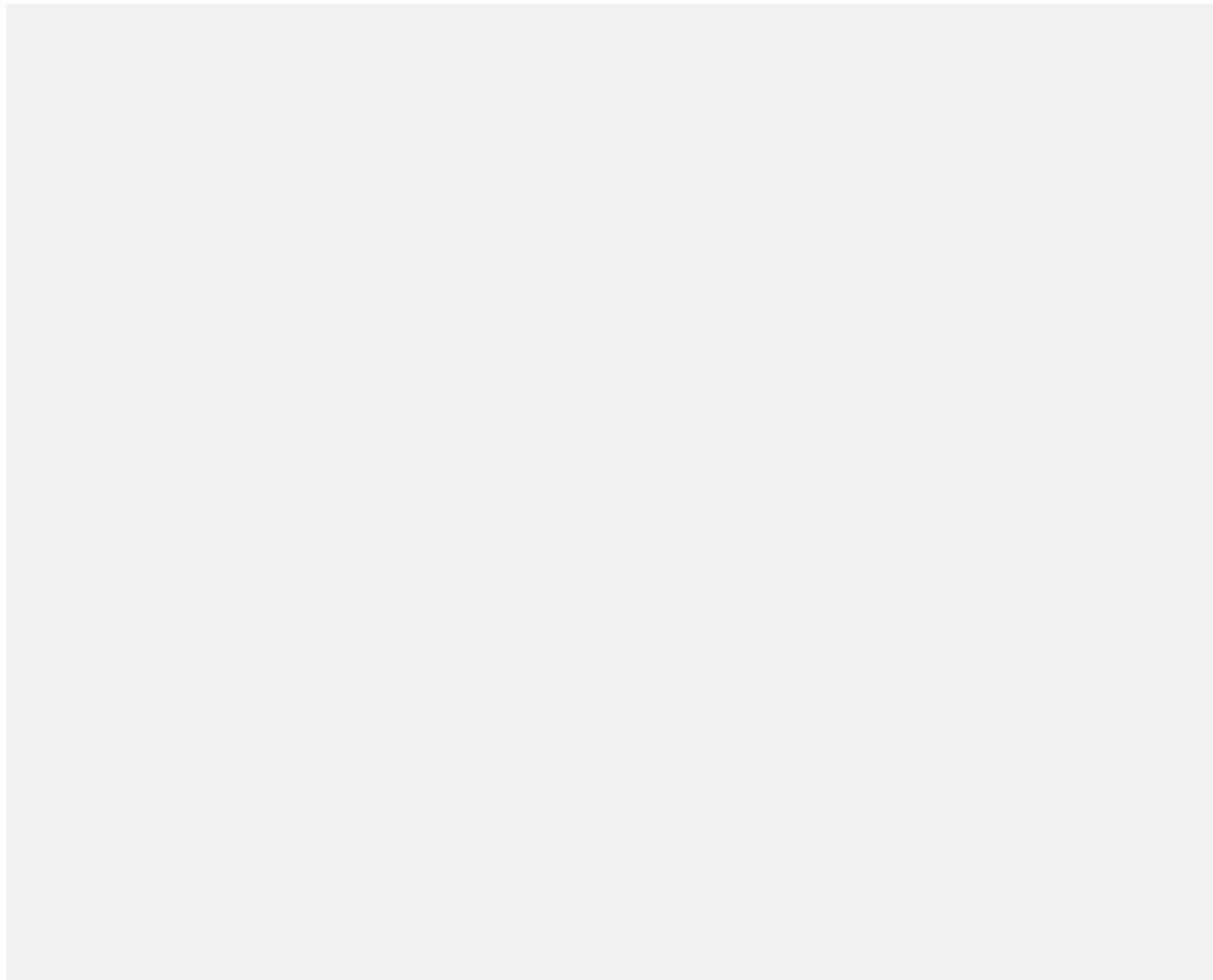
#### メソッド

次に、クラスのインスタンスマソッドを示します。引数を取るメソッドはありません。

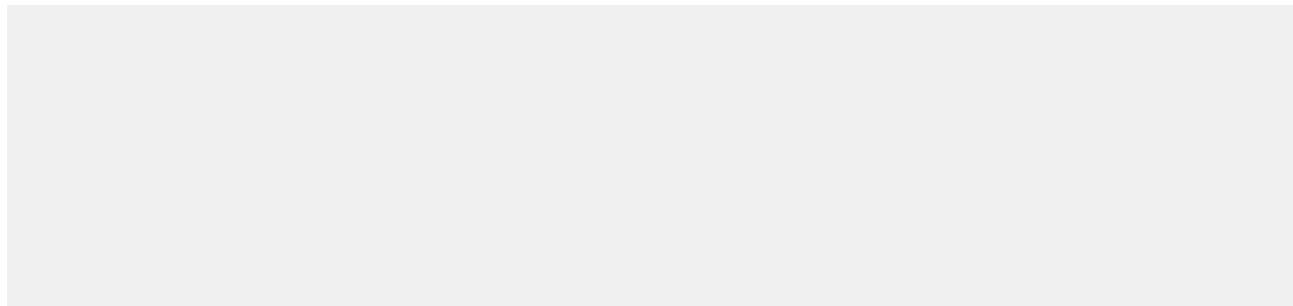
メソッド	戻り値	説明
		項目が、sObject の項目の定義で必須の場合は 、それ以外の場合は を返します。
		動的 SOQL クエリでそのまま使用できる形式で項目パス文字列を返します。
		このメソッドの使用例は、「 <a href="#">Visualforce ページへの項目セットの表示</a> 」を参照してください。
		Salesforce ユーザインターフェースの項目の隣に表示されるテキストラベルを返します。
		項目が項目セットで必須の場合は 、そうでない場合は を返します。
		項目の Apex データ型を返します。

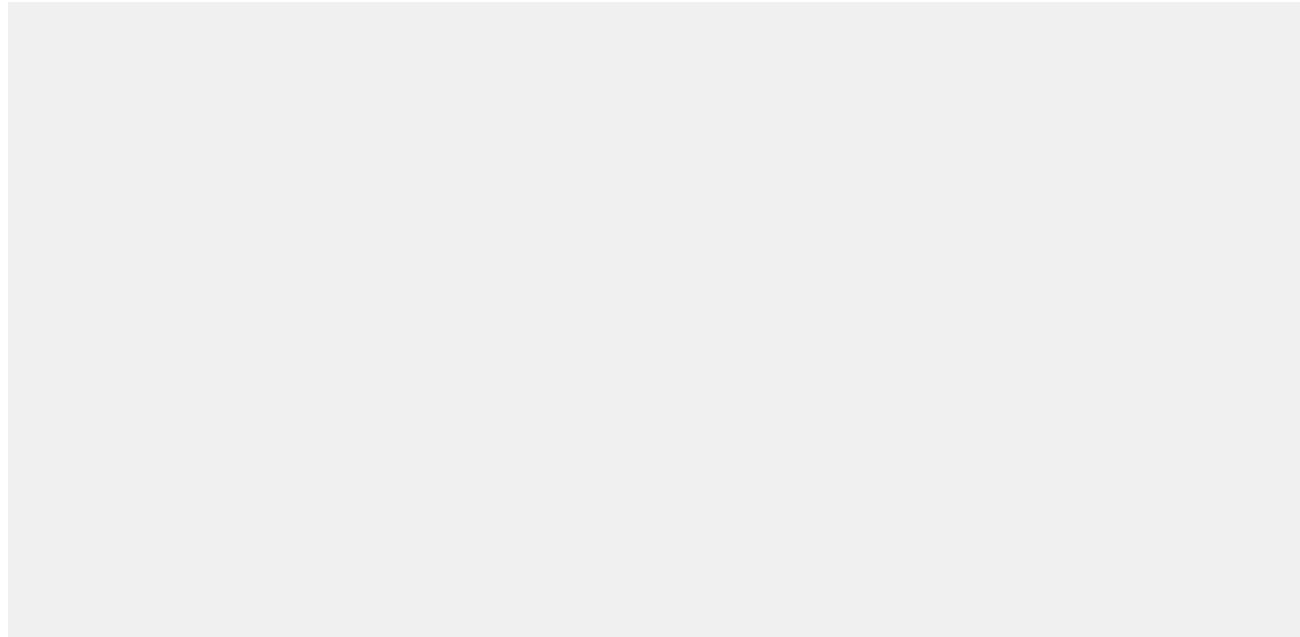
### サンプル: Visualforce ページへの項目セットの表示

このサンプルでは、および メソッドを使用して、Merchandise カスタムオブジェクトの Dimensions 項目セットに含まれるすべての項目を動的に取得します。取得した項目のリストを使用して、これらの項目を表示に使用できるようにする SOQL クエリを作成します。Visualforce ページは、クラスをコントローラとして使用します。



上記のコントローラを使用する Visualforce ページは単純です。





上記のマークアップは、フォーム上の項目を必須項目として示す必要があるかどうかを判定するために使用する式です。項目セット内の項目は、項目セット定義または項目自体の定義によって必須にすることができます。この式では両方を処理します。

## カスタム設定メソッド

カスタム設定メソッドはすべてインスタンスマソッドです。つまり、カスタム設定の特定のインスタンスでコードされ、動作します。カスタム設定には、階層とリストの2種類があります。メソッドは、リストカスタム設定を処理するメソッド、および階層カスタム設定を処理するメソッドに分類されます。

次に、リストカスタム設定のインスタンスマソッドを示します。

表1: リストカスタム設定メソッド

名前	引数	戻り値	説明
	Map<String <i>Data_Set_name</i> , CustomSetting__c>	カスタム設定用に定義されたデータセットの対応付けを返します。  データセットが定義されていない場合、このメソッドは空の対応付けを返します。	 メモ: Salesforce.com API バージョン 20.0 以前を使用して保存された Apex の場合、返される対応付けのキーであるデータセット名は小文字に変換されます。 Salesforce.com API バージョン 21.0 以降を使用して保存された Apex の場合、返される対応付けキーのデータセット名の大文字

名前	引数	戻り値	説明
			と小文字は変更されず、元の文字が保持されます。
	String <i>dataset_name</i>	CustomSetting__c	指定された <i>dataset_name</i> のカスタム設定データセットレコードを返します。このメソッドは、 <i>dataset_name</i> と同一のオブジェクトを返します。 指定されたデータセットにデータが定義されていない場合は、このメソッドは null を返します。
	String <i>dataset_name</i>	CustomSetting__c	指定された <i>dataset_name</i> のカスタム設定データセットレコードを返します。このメソッドは、 <i>dataset_name</i> と同一のオブジェクトを返します。 指定されたデータセットにデータが定義されていない場合は、このメソッドは null を返します。

次に、階層カスタム設定のインスタンスマソッドを示します。

表2: 階層カスタム設定メソッド

名前	引数	戻り値	説明
		CustomSetting__c	現在のユーザのカスタム設定データセットレコードを返します。カスタム設定レコードに返される項目は、階層内で定義された最下位レベル項目に基づいてマージされます。 ユーザにカスタム設定データが定義されていない場合、このメソッドは新しいカスタム設定オブジェクトを返します。新しいカスタム設定オブジェクトには、 ID に設定された ID と、より上位の階層からマージされた項目が含まれます。この新しいカスタム設定レコードをユーザに追加するには、 または を使用します。階層にカスタム設定データが定義されていない場合、ユーザ ID が含まれる 項目を除き、返されるカスタム設定の項目は空です。   メモ: Salesforce.com API バージョン 21.0 以前を使用して保存された Apex の場合、このメソッドは、階層内の最下位レベルに定義されている項目値から差し込まれた項

名前	引数	戻り値	説明
			<p>目を持つ、カスタム設定データセットレコードを返します。差し込みはユーザから開始します。また、階層にカスタム設定データが定義されていない場合、このメソッドは _____ を返します。</p> <p>例:</p> <ul style="list-style-type: none"> <li>ユーザに定義されているカスタム設定データセット: 「山田太郎」というユーザ、「システム管理者」というプロファイル、および組織全体に定義されたカスタム設定データセットがあり、コードを実行しているユーザが山田太郎である場合、このメソッドは、山田太郎に定義されているカスタム設定レコードを返します。</li> <li>差し込み項目: 「山田太郎」というユーザと「システム管理者」というプロファイルに項目 A および項目 B を持つカスタム設定データセットがあるとします。項目 A は山田太郎に定義されており、項目 B は _____ であるがシステム管理者プロファイルに定義されている場合、このメソッドは、山田太郎には、山田太郎の項目 A とシステム管理者プロファイルから得た項目 B を持つカスタム設定レコードを返します。</li> <li>ユーザにカスタム設定データセットレコードが定義されていない: 現在のユーザが「井上花子」であり、「システム管理者」プロファイルを共有しているが、ユーザとしての井上花子に定義されたデータがない場合、このメソッドは、ID が _____ で、階層内で最下位レベルに定義された項目に基づいて差し込まれた項目を持つ、新しいカスタム設定レコードを返します。</li> </ul> <p>このメソッドは現在のユーザの _____ へのメソッドコールと同じです。</p>
ID <i>User_Id</i>	CustomSetting__c		指定された <i>User_Id</i> のカスタム設定データセットレコードを返します。最下位レベルのカスタム設定レコードと項目が返されます。ユーザレベルのカスタム設定のデータを明示的に取得する場合に使用します。

名前	引数	戻り値	説明
			<p>ユーザにカスタム設定データが定義されていない場合、このメソッドは新しいカスタム設定オブジェクトを返します。新しいカスタム設定オブジェクトには、<code>Profile.Id</code> に設定された ID と、より上位の階層からマージされた項目が含まれます。この新しいカスタム設定レコードをユーザに追加するには、<code>CustomSetting__c</code> または <code>CustomSetting</code> を使用します。階層にカスタム設定データが定義されていない場合、ユーザ ID が含まれる <code>CustomSetting</code> 項目を除き、返されるカスタム設定の項目は空です。</p> <p> メモ: Salesforce.com API バージョン 21.0 以前を使用して保存された Apex の場合、このメソッドは、階層内の最下位レベルに定義されている項目値から差し込まれた項目を持つ、カスタム設定データセットトレコードを返します。差し込みはユーザから開始します。また、階層にカスタム設定データが定義されていない場合、このメソッドは <code>CustomSetting</code> を返します。</p>
	<code>ID Profile_Id</code>	<code>CustomSetting__c</code>	<p>指定された <code>Profile_Id</code> のカスタム設定データセットトレコードを返します。最下位レベルのカスタム設定レコードと項目が返されます。プロファイルレベルのカスタム設定のデータを明示的に取得する場合に使用します。</p> <p>プロファイルにカスタム設定データが定義されていない場合、このメソッドは新しいカスタム設定レコードを返します。新しいカスタム設定オブジェクトには、<code>Profile.Id</code> に設定された ID と、組織のデフォルト値からマージされた項目が含まれます。この新しいカスタム設定をプロファイルに追加するには、<code>CustomSetting__c</code> または <code>CustomSetting</code> を使用します。階層にカスタム設定データが定義されていない場合、プロファイル ID が含まれる <code>CustomSetting</code> 項目を除き、返されるカスタム設定の項目は空です。</p> <p> メモ: Salesforce.com API バージョン 21.0 以前を使用して保存された Apex の場合、このメソッドは、階層の最下位レベルに定義されている項目値から差し込まれた項目を持つ、カスタム設定データセットトレコードを返します。差し込みはユーザから開始します。また、階層にカスタム設定データが定義されていない場合、プロファイル ID が含まれる <code>CustomSetting</code> 項目を除き、返されるカスタム設定の項目は空です。</p>

名前	引数	戻り値	説明
			ドを返します。差し込みはプロファイルから開始します。また、階層にカスタム設定データが定義されていない場合、このメソッドは を返します。
	CustomSetting__c	組織のカスタム設定データセットレコードを返します。  組織にカスタム設定データが定義されていない場合、このメソッドは空のカスタム設定オブジェクトを返します。	 メモ: Salesforce.com API バージョン 21.0 以前を使用して保存された Apex の場合、このメソッドは、組織にカスタム設定データが定義されていなければ、 を返します。
ID <i>User_Id</i>	CustomSetting__c	指定された <i>User_Id</i> のカスタム設定データセットレコードを返します。ユーザレベルで定義されているカスタム設定データのサブセットが必要な場合にのみ使用します。たとえば、組織レベルで「foo」の値を割り当てられているカスタム設定項目があり、ユーザレベルまたはプロファイルレベルで値が割り当てられていないとします。  <i>User_Id</i> は、このカスタム設定項目に を返します。	
ID <i>Profile_Id</i>	CustomSetting__c	指定された <i>Profile_Id</i> のカスタム設定データセットを返します。プロファイルレベルで定義されているカスタム設定データのサブセットが必要な場合にのみ使用します。たとえば、組織レベルで「foo」の値を割り当てられているカスタム設定項目があり、ユーザレベルまたはプロファイルレベルで値が割り当てられていないとします。  を使用すると、このカスタム設定項目に を返します。	

カスタム設定についての詳細は、Salesforce オンラインヘルプの「カスタム設定の概要」を参照してください。

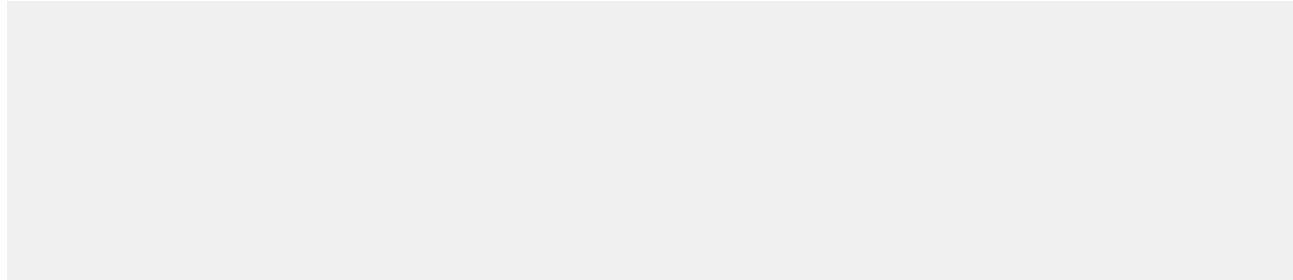


メモ: すべてのカスタム設定データはアプリケーションキャッシュで公開されます。これにより、データベースへのクエリを繰り返し行うコストをかけずに、効率的なアクセスを実現します。ただし、Standard Object Query Language (SOQL) を使ってカスタム設定データをクエリするとアプリケーションキャッシュ

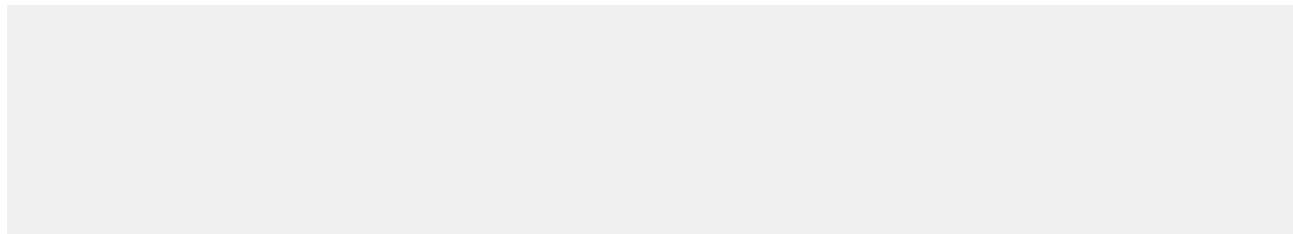
を活用しません。そのため、カスタムオブジェクトのクエリと似ています。キャッシュのメリットを得るには、Apex カスタム設定メソッドなどのカスタム設定データにアクセスする他のメソッドを使用します。

### カスタム設定の例

次の例では、Games というリストカスタム設定を使用します。Games には という項目があります。この例では、最初のデータセットの値が文字列 がどうかを決定します。



次の例では、[国コードと州コードのカスタム設定例](#)のカスタム設定を使用します。 と メソッドのリストカスタム設定が同一の値を返すことを示します。



### 階層カスタム設定の例

次の例では、階層カスタム設定 GamesSupport に という項目があります。コードは で指定されたプロファイルの値を返します。



メソッドを使用した場合、例は同一になります。

次の例では、階層カスタム設定メソッドの使用方法を示します。この例では、 には、階層の下から 2 番目のレベルで定義されている項目から返される特定のユーザまたはプロファイルに設定されない項目値が返されます。また、 の使用方法を例で示します。

最後に、 が特定のユーザまたはプロファイルのみのカスタム設定レコードの項目を返し、階層の他のレベルの値をマージしない方法を例示します。代わりに、 では設定されていない項目では を返します。次の例では、Hierarchy という階層カスタム設定を使用します。Hierarchy には、 および という 2 つの項目があります。また、ユーザ Robert にはシステム管理者プロファイルがあります。この例では、組織、プロファイル、ユーザ設定は次のようにになります。

## 組織の設定

```
: Hello  
: World
```

## プロファイルの設定

```
: Goodbye  
は設定されません。
```

## ユーザの設定

```
: Fluffy  
は設定されません。
```

次の例は、Robert が組織で

メソッドをコールした場合の結果を示します。

Robert が \_\_\_\_\_ で指定したユーザ ID を \_\_\_\_\_ に渡すと、同一の結果になります。これは、カスタム設定のデータの最下位レベルがユーザレベルで指定されるためです。

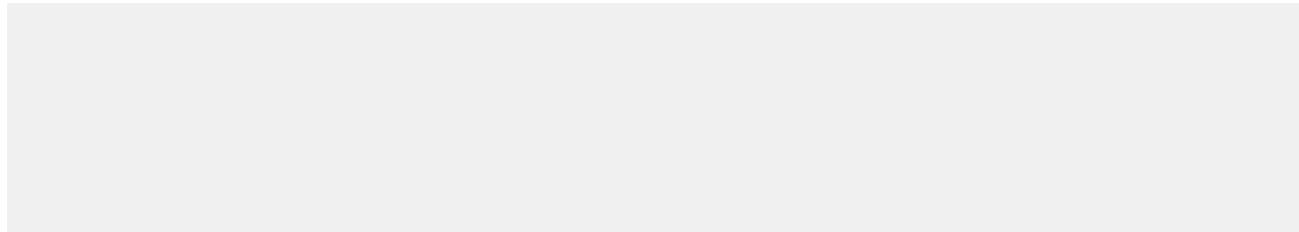
Robert が \_\_\_\_\_ で指定されたシステム管理者プロファイル ID を \_\_\_\_\_ に渡すと、結果は異なります。プロファイルに指定されたデータが返されます。

Robert が \_\_\_\_\_ を使用して組織のデータセットを返そうとする場合、結果は次のようになります。

メソッドを使用して、Robert はユーザ設定およびプロファイル設定固有の階層カスタム設定値を取得できます。たとえば、Robert がユーザ ID \_\_\_\_\_ を \_\_\_\_\_ に渡す場合、結果は次のようになります。



Robert がシステム管理者プロファイル ID を渡すと、結果は次のようにになります。

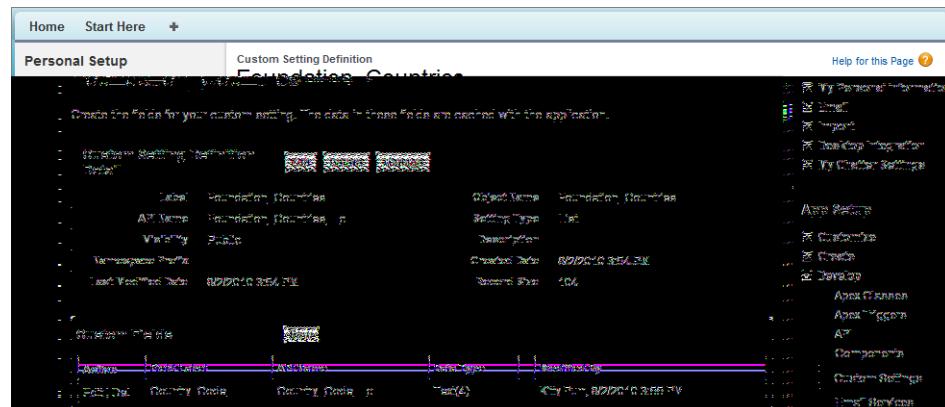


### 国コードと州コードのカスタム設定例

この例では、関連する情報を保存するために2つのカスタム設定オブジェクトを使用する方法と、関連する選択リストの集合のデータを表示する Visualforce ページについて示します。

次の例では、国コードと州コードは、Foundation\_Countries と Foundation\_States という2つの異なるカスタム設定に保存されます。

Foundation\_Countries カスタム設定はリストタイプのカスタム設定であり、という1つの項目が含まれます。



Foundation\_States カスタム設定もリストタイプのカスタム設定であり、次の項目が含まれます。

- 国コード
- 都道府県コード
- 都道府県名

The screenshot shows the Salesforce Setup interface. On the left, there's a sidebar with sections like Personal Setup, App Setup, and Custom Settings. In the main area, a new custom setting is being created with the name 'Foundation\_States'. The details page shows the following information:

Label	Foundation_States	Object Name	Foundation_States
API Name	Foundation_States_c	Setting Type	List
Visibility	Public	Description	
Namespace Prefix		Created Date	8/2/2010 3:55 PM
Last Modified Date	8/2/2010 3:55 PM	Record Size	149

Below this, the 'Custom Fields' section lists three fields:

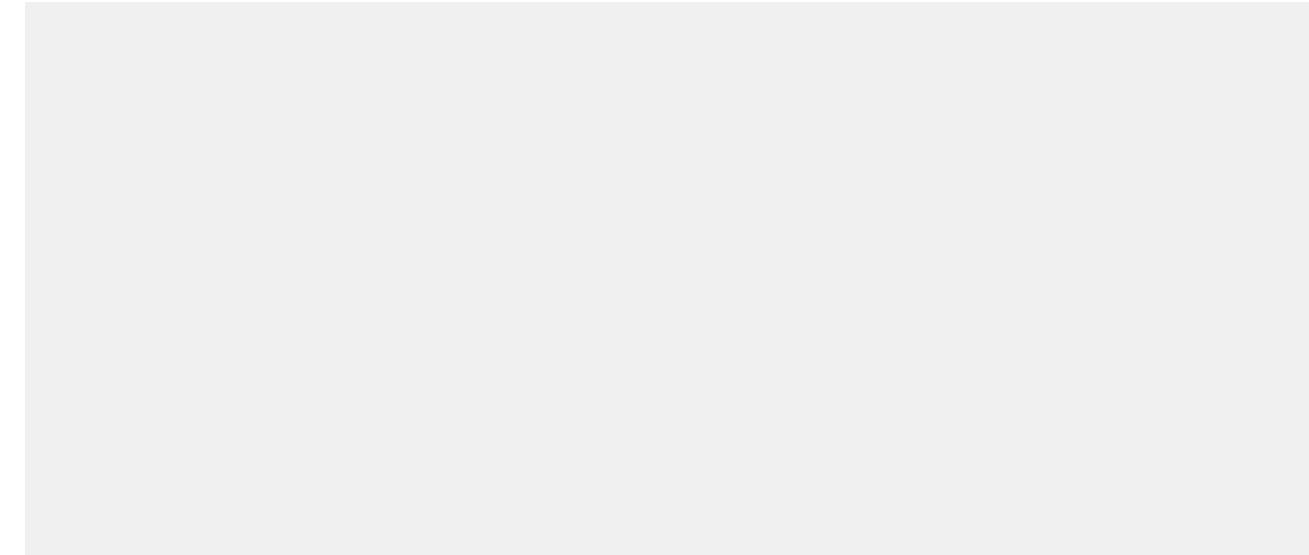
Action	Field Label	API Name	Data Type	Modified By
Edit   Del	Country Code	Country_Code_c	Text(4)	Kitty Purr 8/3/2010 3:46 PM
Edit   Del	State Code	State_Code_c	Text(5)	Kitty Purr 8/2/2010 3:57 PM
Edit   Del	State Name	State_Name_c	Text(40)	Kitty Purr 8/2/2010 3:58 PM

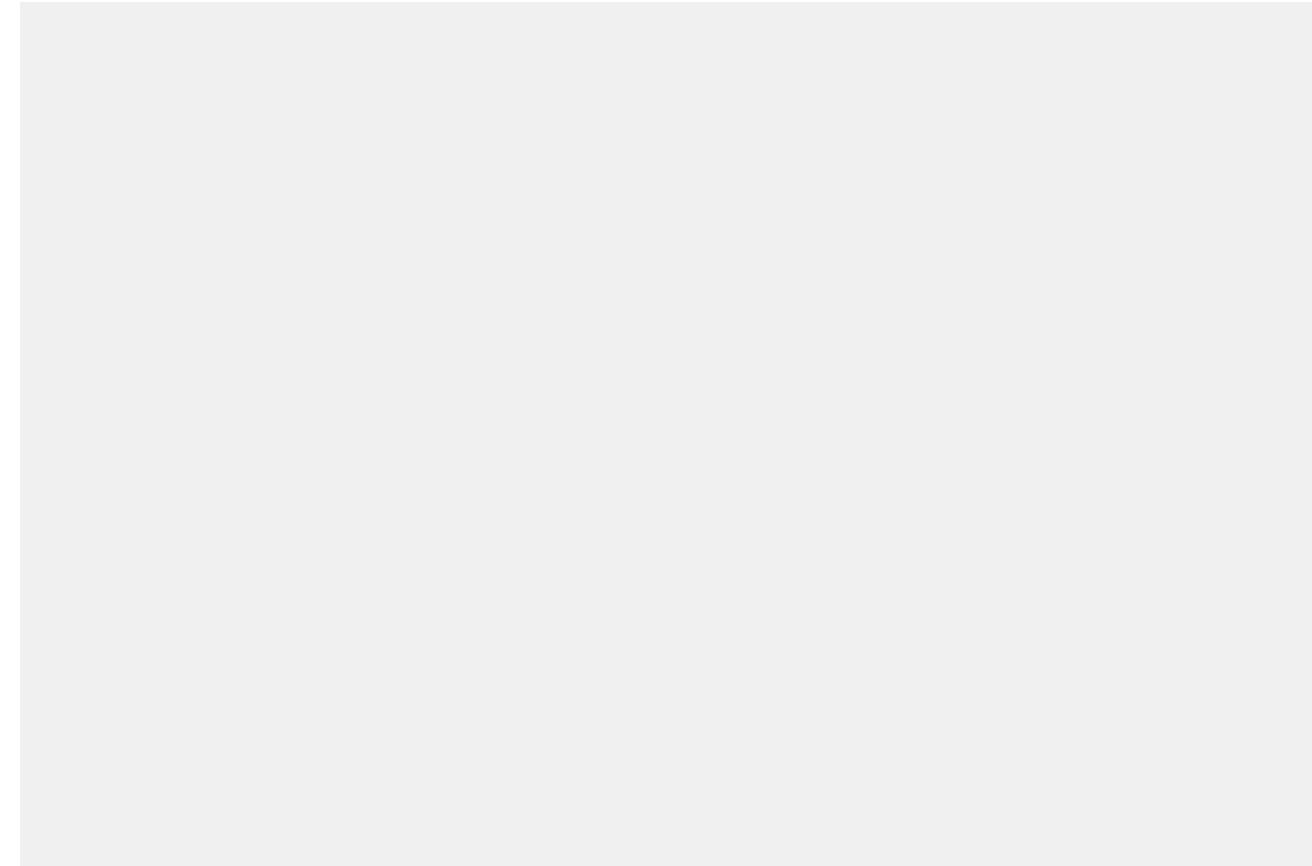
Visualforce ページには、国用と都道府県用の 2 つの選択リストが表示されます。

The screenshot shows the Visualforce page editor. A dropdown menu is open, showing a list of Canadian provinces and territories: Alberta, British Columbia, Manitoba, New Brunswick, Newfoundland and Labrador, and Nova Scotia. The page code below the editor shows the markup for this component:

```

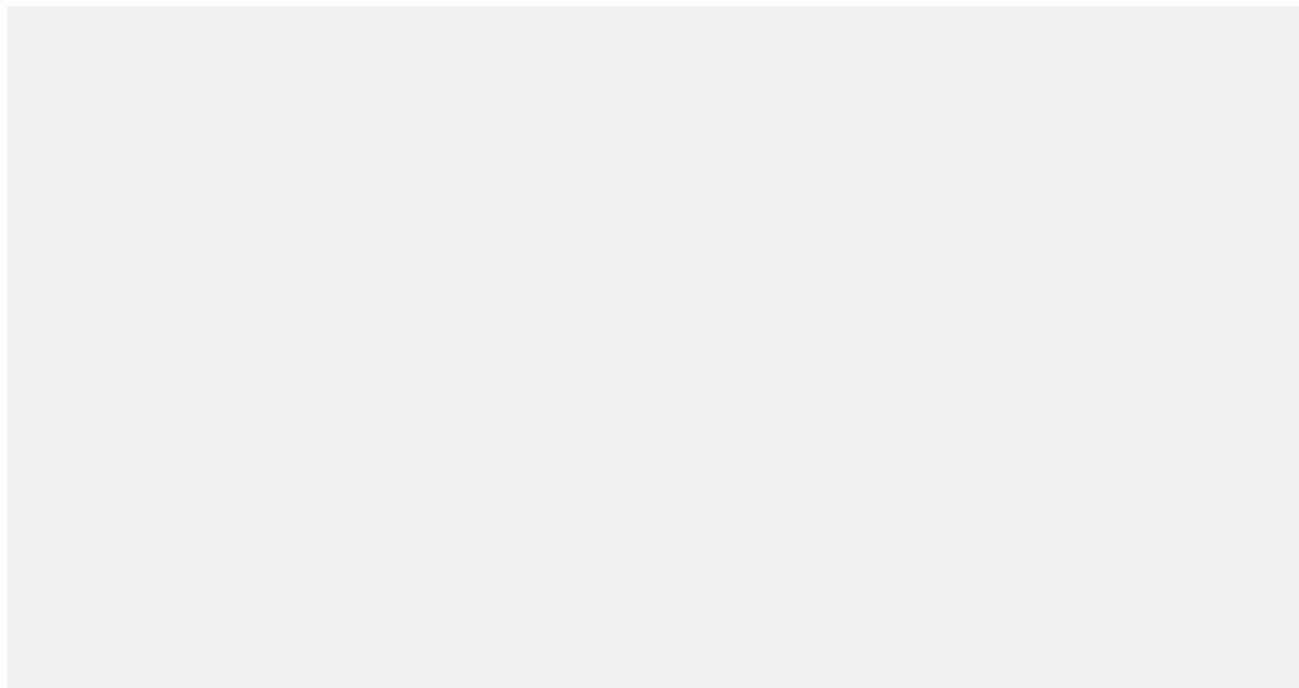
1 <apex:page controller="CountryStatePicker">
2   <apex:form>
3     <apex:actionFunction name="rerenderStates" rerender="states SelectList" >
4       <apex:param name="firstParam" assignTo="{!country}" value="" />
5     </apex:actionFunction>
6
7   <table><tbody>
8     <tr>
9       <th>Country</th>
10      <td>
11        <apex:selectList id="country" styleclass="std" size="1"
12          value="{!country}" onChange="rerenderStates(this.value)">
```

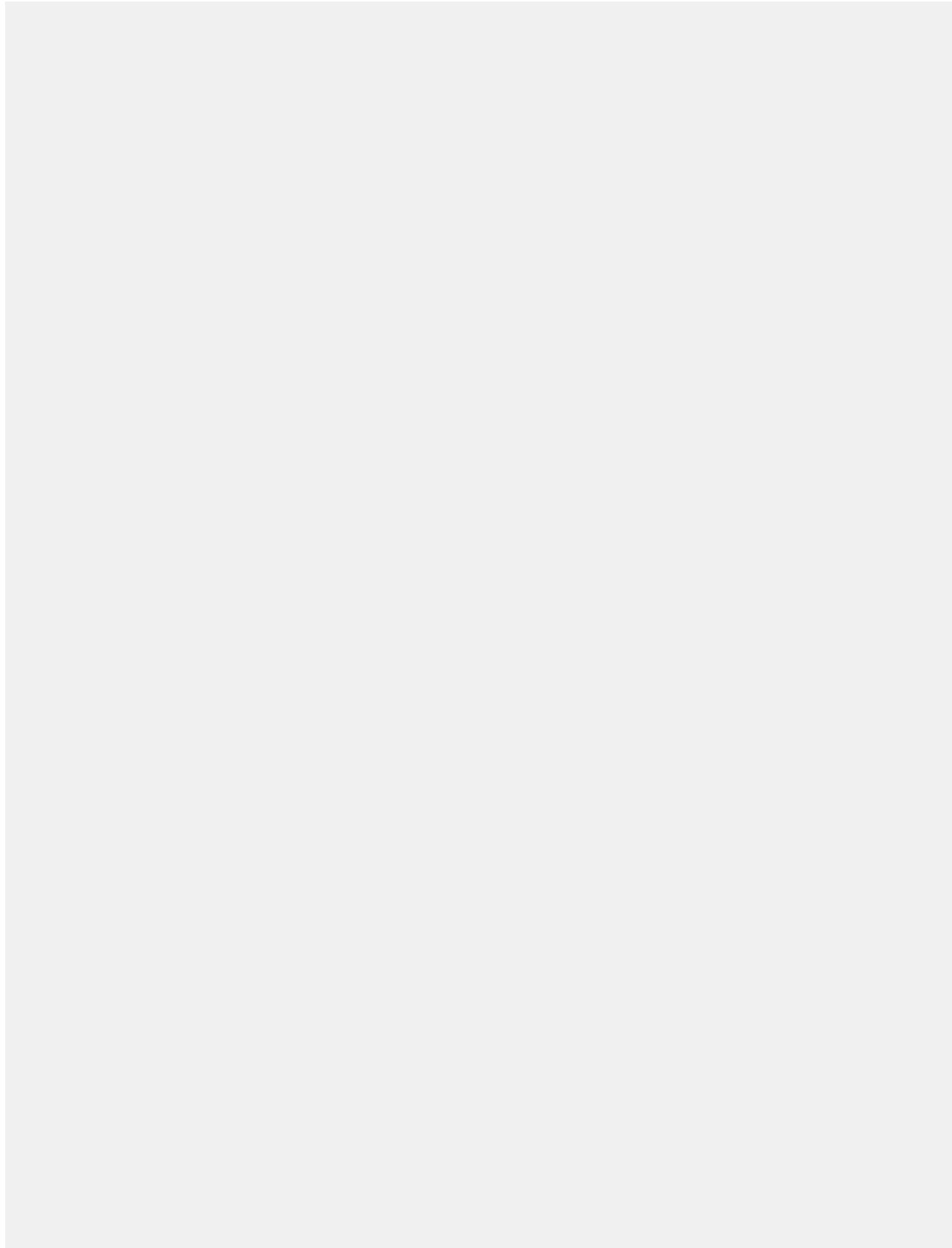


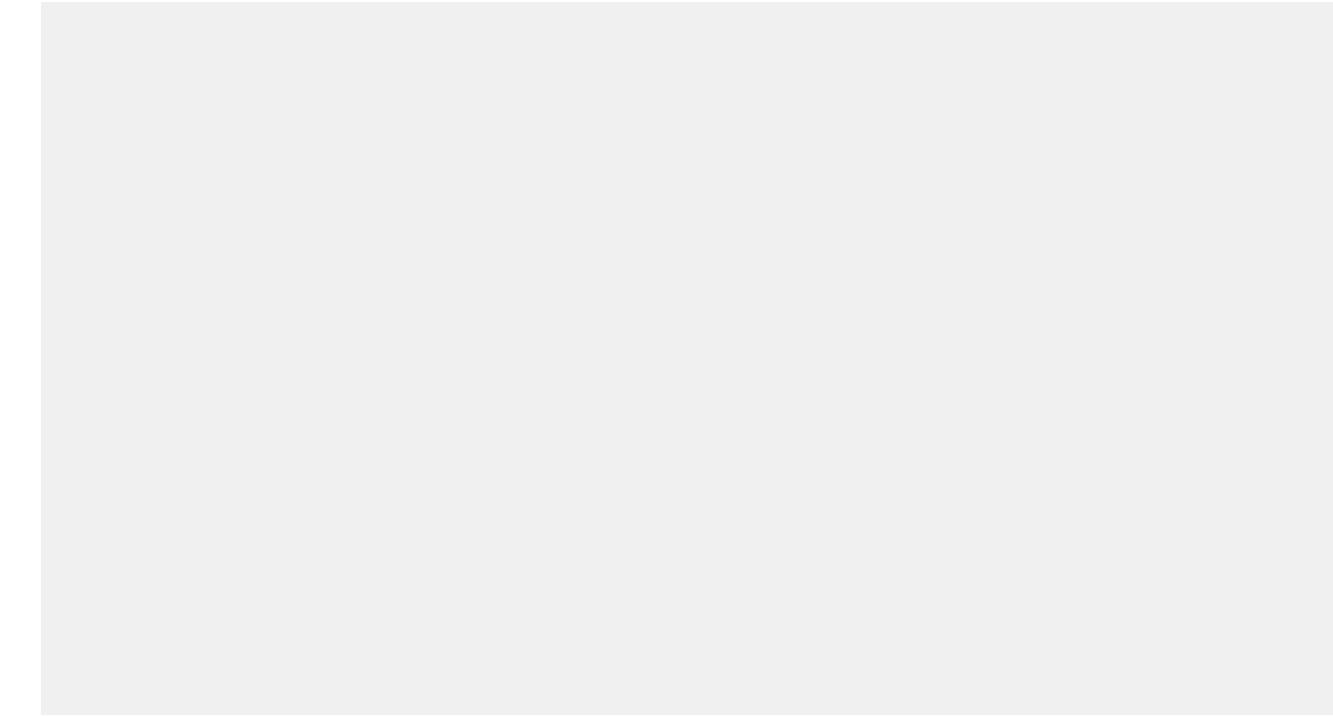


Apex コントローラ  
を返します。

は、カスタム設定に入力された値を探し、Visualforce ページにその値







## Apex System メソッド

次の Apex システムクラスがあります。

- [ApexPages](#)
- [Approval](#)
- [Database](#)
  - ◊ [Database Batch](#)
  - ◊ [Database DMLOptions](#)
  - ◊ [Database EmptyRecycleBinResult](#)
  - ◊ [Database Error](#)
- [JSON サポート](#)
  - ◊ [JSON メソッド](#)
  - ◊ [JSONGenerator メソッド](#)
  - ◊ [JSONParser メソッド](#)
- [Limits](#)
- [Math](#)
- [MultiStaticResourceCalloutMock](#)
- [Apex REST](#)
  - ◊ [RestContext メソッド](#)
  - ◊ [RestRequest メソッド](#)

### ◊ RestResponse メソッド

- Search
- StaticResourceCalloutMock
- System
- Test
- TimeZone
- Type
- URL
- UserInfo
- Version

## ApexPages メソッド

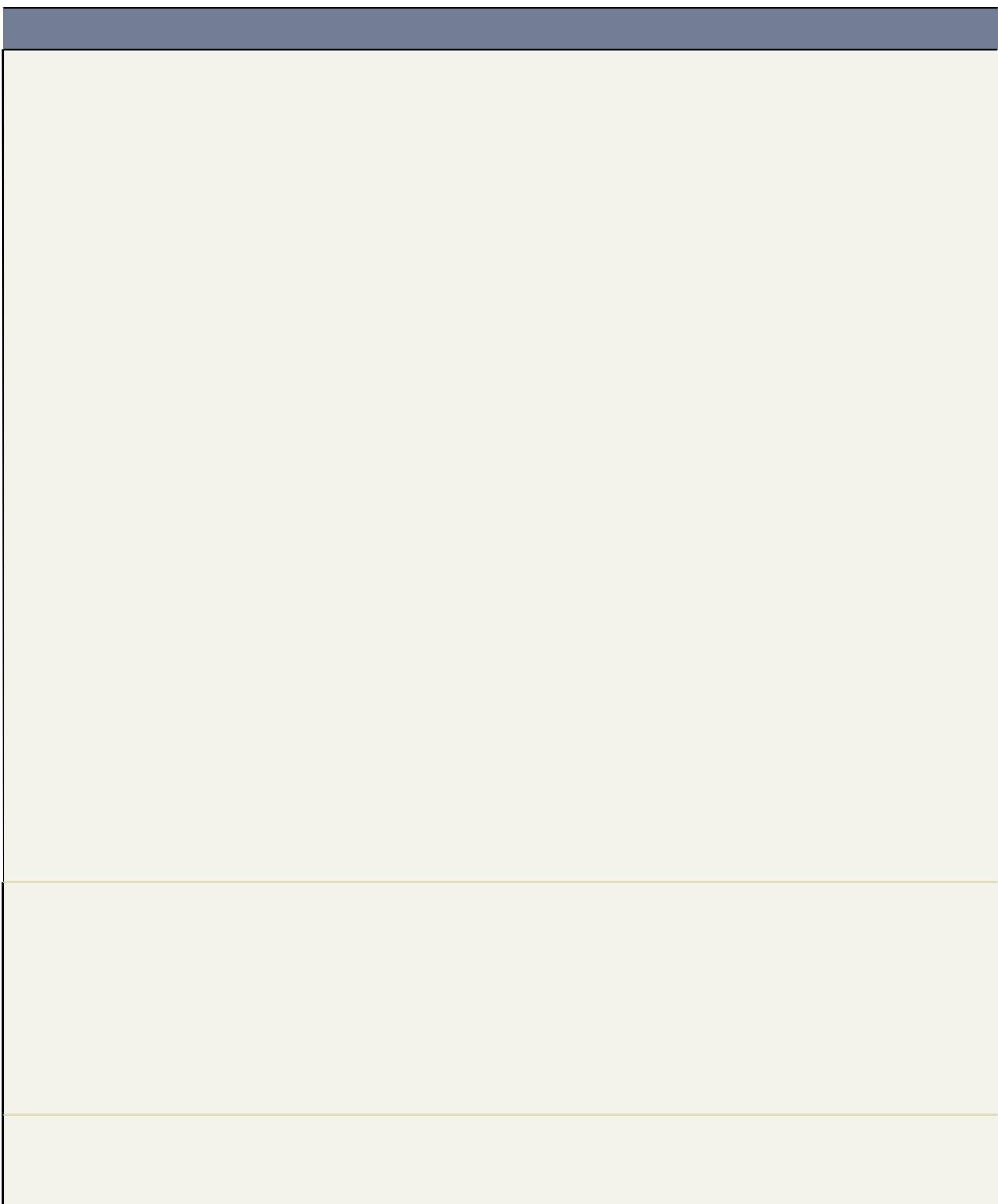
現在のページの参照、および現在のページに関連付けられたメッセージの追加や確認をするために、ApexPages を使用します。また、ApexPages は クラスおよび クラスの名前空間として使用されます。

次の表に、ApexPages メソッドの一覧を示します。

名前	引数	戻り値	説明
	sObject <i>ApexPages.Message</i>	Void	現在のページのコンテキストにメッセージを追加します。
	Exception <i>ex</i>	Void	発生した例外に基づいて、現在のページのコンテキストにメッセージのリストを追加します。
		ApexPages.Message[]	現在のコンテキストに関連付けられたメッセージのリストを返します。
		boolean	現在のコンテキストに関連付けられたメッセージが存在する場合は 、存在しない場合は を返します。
	ApexPages.Severity	boolean	指定された重要度のメッセージが存在する場合は 、存在しない場合は を返します。

## Approval メソッド

次の表に、静的 Approval メソッドの一覧を示します。Approval は、 クラスおよび クラスの名前空間としても使用されます。



名前	引数	戻り値	説明
	Approval.ProcessRequest [] <i>ProcessRequests</i>	Approval.ProcessResult []	新しい承認のリストを申請し、既存の承認申請を承認または却下します。
	Boolean <i>opt_allOrNone</i>		(省略可能) <i>opt_allOrNone</i> パラメータは、部分的な完了を操作で許可するかどうかを指定します。このパラメータを <b>True</b> に設定した場合、承認が失敗しても、残りの承認プロセスを正常に完了できます。

Apex 承認プロセスについての詳細は、「[Apex 承認プロセスクラス](#)」(ページ 650)を参照してください。

## JSON サポート

Apex では JavaScript Object Notation (JSON) がサポートされ、Apex オブジェクトの JSON 形式への逐次化、逐次化された JSON コンテンツの並列化を実行できます。Apex では、JSON 逐次化と並列化のメソッドを公開するクラスセットを提供します。次の表は、使用可能なクラスを示しています。

クラス	説明
	Apex オブジェクトを JSON 形式で逐次化するメソッドと、このクラスの <code>JSON.serialize()</code> メソッドを使用して、逐次化された JSON コンテンツを並列化するメソッドがあります。
	標準 JSON 符号化方式を使用して Apex オブジェクトを JSON コンテンツに逐次化する場合に使用されるメソッドを含みます。
	JSON 符号化されたコンテンツのパーサーを表します。

は、JSON 解析に使用されるトークンを列挙します。

これらのクラスのメソッドは、実行中に問題が発生した場合 `JSONException` を生成します。

### JSON サポートの考慮事項

- JSON の逐次化と並列化のサポートは、`sObject` (標準オブジェクトとカスタムオブジェクト)、Apex プリミティブ型とコレクション型、データベースメソッドの戻り値のデータ型 (`SaveResult`、`DeleteResult` など)、Apex クラスのインスタンスで利用できます。
- 管理パッケージの `CustomObject` 型のカスタムオブジェクトのみ、管理パッケージ外のコードから逐次化できます。管理パッケージに定義される Apex クラスのインスタンスであるオブジェクトは、逐次化できません。
- 文字列ではないキーが含まれる `Map<String, Object>` オブジェクトを並列化すると、逐次化前の対応する `Map<Object, sObject>` オブジェクトとは一致しません。キー値は、逐次化時に文字列に変換されるため、並列化すると型が変更されます。たとえば、`Map<Object, sObject>` は、`Map<String, sObject>` になります。

- オブジェクトが上位型として宣言され、下位型のインスタンスに設定されている場合、一部のデータが失われる可能性があります。オブジェクトを逐次化して並列化すると、上位型および下位型に固有の項目は失われます。
- オブジェクトに、そのオブジェクト自体への参照が設定されている場合は逐次化されず、が生成されます。
- 同じオブジェクトを 2 回参照する参照グラフが並列化されると、参照されるオブジェクトのコピーが複数生成されます。
- データ型は、逐次化できません。Visualforce コントローラなど、型のメンバー変数を持つ逐次化可能なクラスがあり、このオブジェクトを作成しようとすると、例外が発生します。逐次化可能なクラスで を使用するには、メソッド内でローカル変数を使用します。

## JSON メソッド

Apex オブジェクトを JSON 形式で逐次化するメソッドと、このクラスの逐次化された JSON コンテンツを並列化するメソッドがあります。

メソッドを使用して、逐次化された JSON コンテンツを並列化するメソッドがあります。

### 使用方法

クラスのメソッドを使用して、Apex オブジェクトの JSON の逐次化と並列化の往復処理を実行します。

### メソッド

次に、クラスの静的メソッドを示します。

メソッド	引数	戻り値	説明
	Boolean <i>pretty</i>		新しい JSON ジェネレータを返します。  <i>pretty</i> 引数は、JSON ジェネレータが、JSON コンテンツをインデントされた見栄えのよい印刷形式で作成するかどうかを指定します。インデントされたコンテンツを作成するには、 を設定します。
	String <i>jsonString</i>		新しい JSON パーサーを返します。  <i>jsonString</i> 引数は、解析する JSON コンテンツです。
	String <i>jsonString</i>  String <i>apexType</i>	anyType	指定された JSON 文字列を Apex オブジェクトの指定された型に並列化します。  <i>jsonString</i> 引数は、並列化する JSON コンテンツです。  <i>apexType</i> 引数は、このメソッドが JSON コンテンツの並列化後に作成するオブジェクトの Apex 型です。解析する JSON コンテンツに、引数で指定された Apex 型に存在しない属性 (存在しない項目やオブジェクト

メソッド	引数	戻り値	説明
			<p>など) が含まれている場合、このメソッドはそれらの属性を無視して、残りの JSON コンテンツを解析します。ただし、Salesforce.com API バージョン 24.0 以前を使用して保存された Apex の場合、このメソッドは存在しない属性に対して実行時例外を発生させます。</p> <p>次の例では、 値を並列化します。</p>
			<p>String <i>jsonString</i></p> <p><i>apexType</i></p> <p>指定された JSON 文字列を Apex オブジェクトの指定された型に並列化します。JSON 文字列内のすべての属性は、指定された型に存在する必要があります。</p> <p><i>jsonString</i> 引数は、並列化する JSON コンテンツです。</p> <p><i>apexType</i> 引数は、このメソッドが JSON コンテンツの並列化後に作成するオブジェクトの Apex 型です。</p> <p>解析する JSON コンテンツに、引数で指定された Apex 型に存在しない属性(存在しない項目やオブジェクトなど)が含まれている場合、このメソッドは実行時例外を発生させます。</p> <p>次の例は、JSON 文字列を クラスで表されるユーザ定義型のオブジェクト(この例で定義)に並列化します。</p>

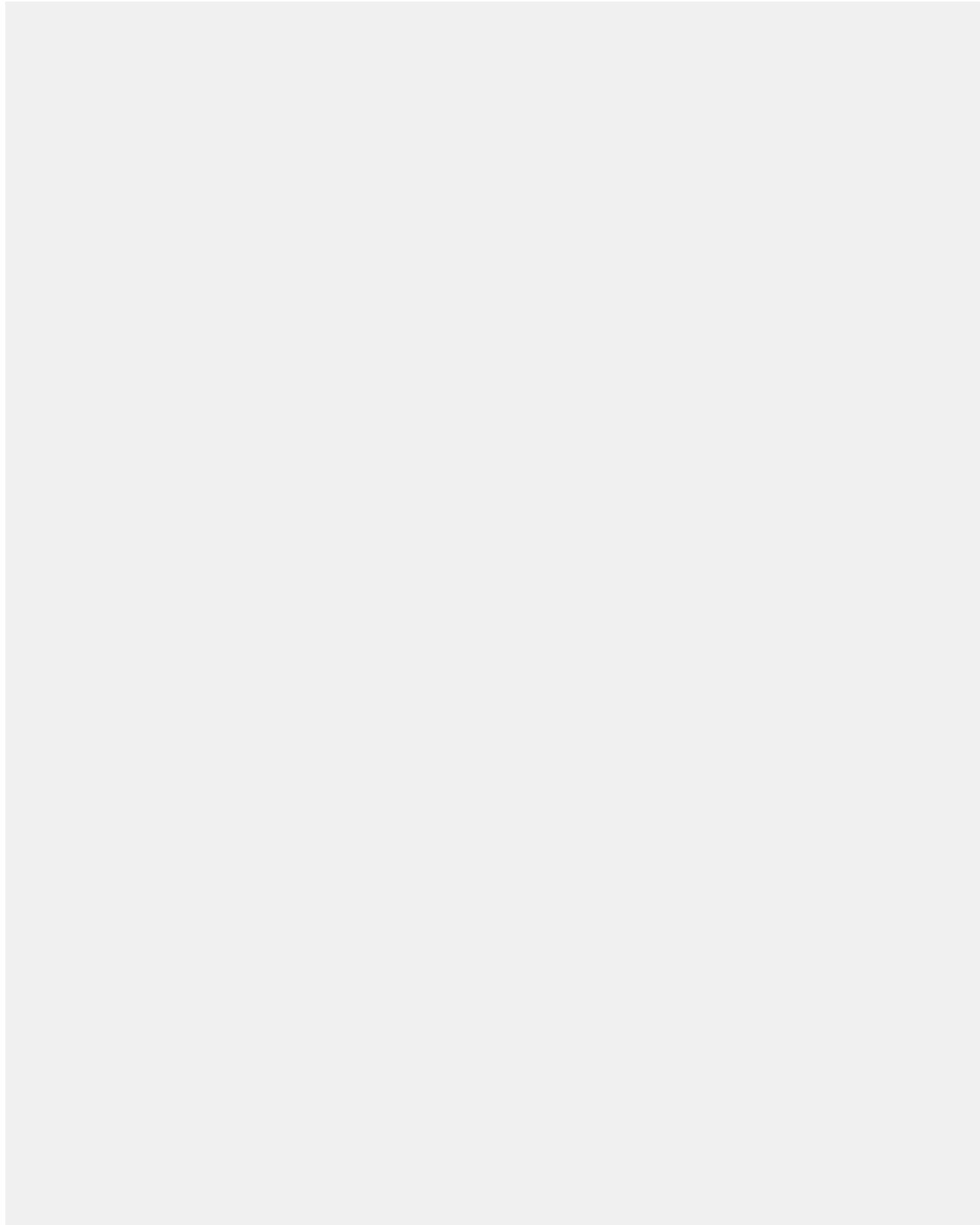
メソッド	引数	戻り値	説明
	String <i>jsonString</i>	anyType	<p>指定された JSON 文字列をプリミティブデータ型のコレクションに並列化します。</p> <p><i>jsonString</i> 引数は、並列化する JSON コンテンツです。</p> <p>次の例では、アプライアンスオブジェクトの JSON 表現を、プリミティブデータ型が含まれるマップと、さらにプリミティブ型のコレクションに並列化します。その後、並列化された値を検証します。</p>

メソッド	引数	戻り値	説明
	任意の型 <i>object</i>	String	Apex オブジェクトを JSON コンテンツに逐次化します。  <i>object</i> 引数は、逐次化する Apex オブジェクトです。

メソッド	引数	戻り値	説明
			次の例では、新しい 値を逐次化します。
任意の型 <i>object</i>	String		Apex オブジェクトを JSON コンテンツに逐次化し、見 栄えのよい印刷形式を使用してインデントされたコン テンツを生成します。  <i>object</i> 引数は、逐次化する Apex オブジェクトです。

#### サンプル: 請求書リストの逐次化と並列化

次のサンプルは、オブジェクトのリストを作成して、リストを逐次化します。次に、逐次化された JSON 文字列を使用してリストを並列化し、元のリストに表示された請求書と同じ請求書が新しいリストに含まれることをサンプルで検証します。



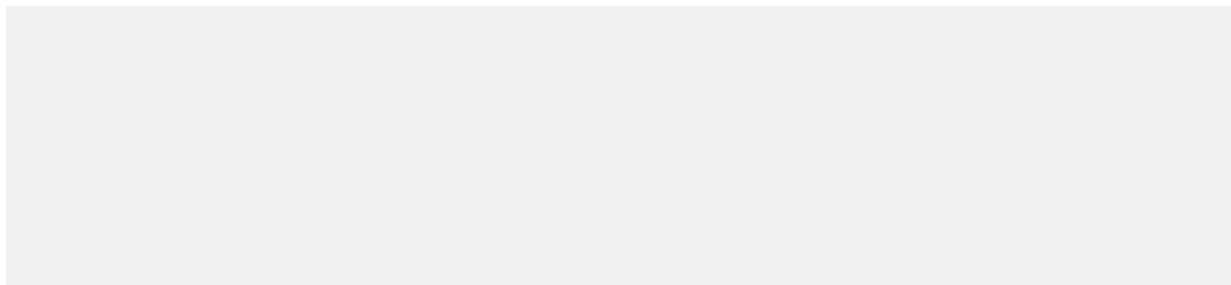
## JSON 逐次化の考慮事項

次に、**メソッド**の動作の相違を説明します。これらの相違は、保存された Apex コードの Salesforce.com API バージョンによって異なります。

### 追加項目セットのあるクエリ対象の sObject の逐次化

Salesforce.com API バージョン 27.0 以前を使用して保存された Apex の場合、クエリ対象の sObject に追加項目セットがある場合、これらの項目は **メソッド**によって返される逐次化された JSON 文字列に含まれません。Salesforce.com API バージョン 28.0 以降を使用して保存された Apex で開始する場合は、逐次化される JSON 文字列に追加項目が含まれます。

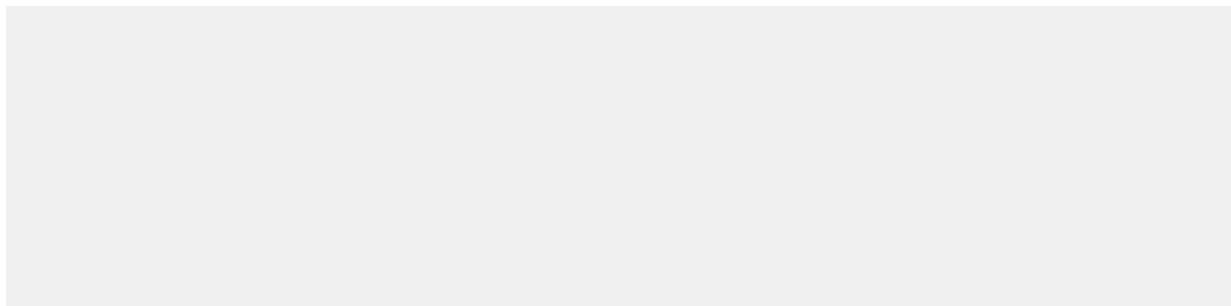
次の例では、クエリされた後に項目が取引先責任者に追加され、その後取引先責任者が逐次化されます。アサーションステートメントは、JSON 文字列に追加項目が含まれていることを検証します。このアサーションは、Salesforce.com API バージョン 28.0 以降を使用して保存された Apex に対して有効です。



### 集計クエリの結果項目の逐次化

Salesforce.com API バージョン 27.0 を使用して保存された Apex の場合、**メソッド**を使用して逐次化すると、集計クエリの結果に SELECT ステートメントの項目は含まれません。API の以前のバージョン、または API バージョン 28.0 以降の場合、逐次化された集計クエリの結果に SELECT ステートメントのすべての項目が含まれます。

次に、2つの項目 (ID 項目の数と取引先名) を返す集計クエリの例を示します。



## 関連リンク

[型メソッド](#)

## JSONGenerator メソッド

標準 JSON 符号化方式を使用して Apex オブジェクトを JSON コンテンツに逐次化する場合に使用されるメソッドを含みます。

### 使用方法

クラスの逐次化メソッドを使用して Apex により生成される JSON 符号化方式は、標準 JSON 符号化方式と異なる場合があるため、標準 JSON で符号化されたコンテンツを生成できるクラスが提供されています。

### メソッド

次に、クラスのインスタンスマソッドを示します。

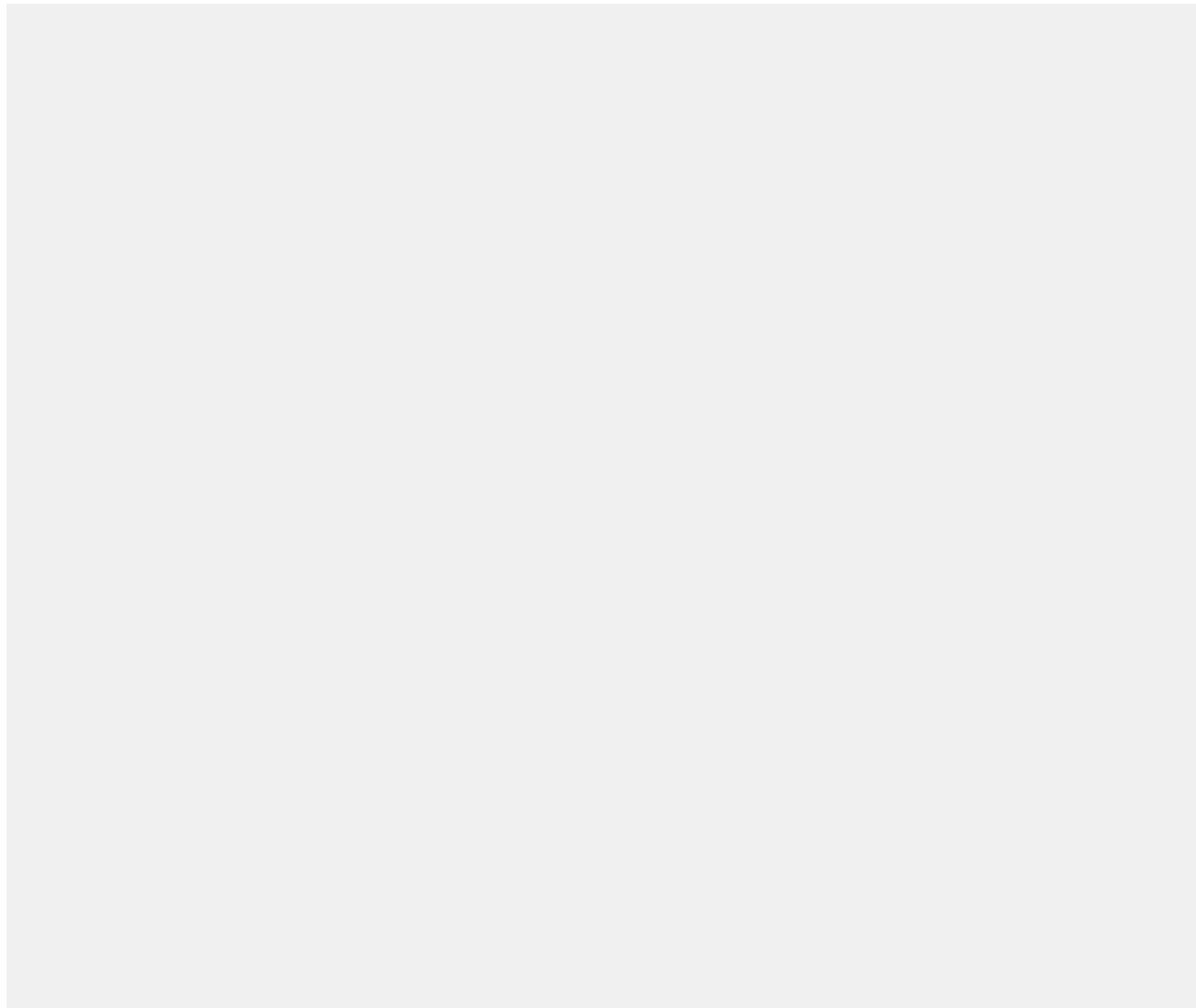
メソッド	引数	戻り値	説明
		Void	JSON ジェネレータを終了します。 JSON ジェネレータが終了すると、コンテンツを出力することはできません。
		String	生成された JSON コンテンツを返します。 また、JSON ジェネレータが終了していない場合は終了します。
		Boolean	JSON ジェネレータが終了している場合は <code>true</code> を返します。終了していない場合は <code>false</code> を返します。
BLOB <i>blobValue</i>		Void	指定された <code>blobValue</code> 値を Base64 で符号化された文字列として出力します。
String <i>fieldName</i> Blob <i>blobValue</i>		Void	指定された項目名と BLOB 値を使用して、項目名と値のペアを出力します。
Boolean <i>blobValue</i>		Void	指定された boolean 値を出力します。
String <i>fieldName</i> boolean <i>booleanValue</i>		Void	指定された項目名と boolean 値を使用して、項目名と値のペアを出力します。
date <i>dateValue</i>		Void	指定された date 値を ISO-8601 形式で出力します。
String <i>fieldName</i> date <i>dateValue</i>		Void	指定された項目名と date 値を使用して、項目名と値のペアを出力します。date 値は、ISO-8601 形式で出力されます。
dateTime <i>datetimestampValue</i>		Void	指定された dateTime 値を ISO-8601 形式で出力します。

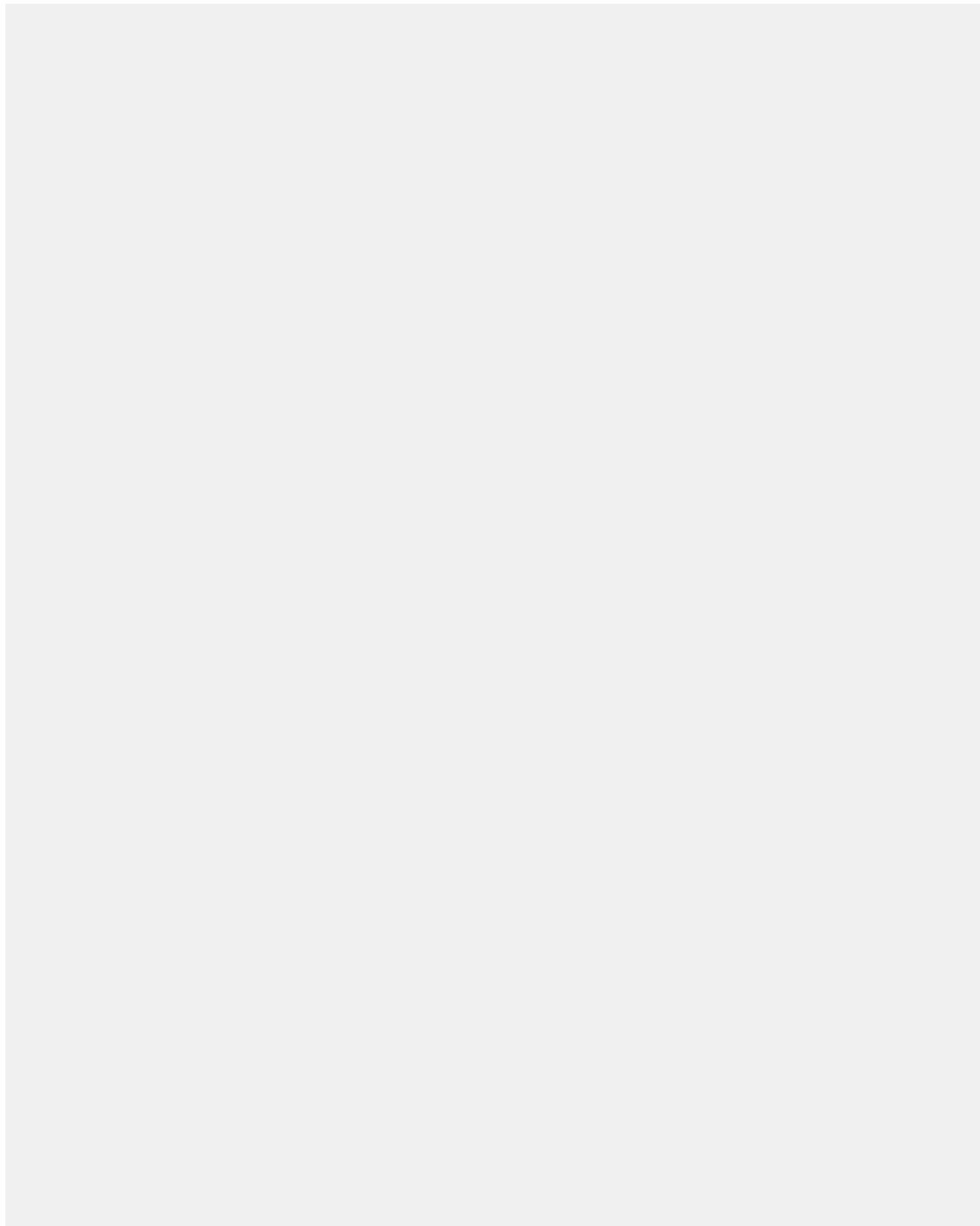
メソッド	引数	戻り値	説明
	<code>String <i>fieldName</i> Void dateTime <i>datetimetypeValue</i></code>		指定された項目名と <code>dateTime</code> 値を使用して、項目名と値のペアを出力します。 <code>dateTime</code> 値は、ISO-8601 形式で出力されます。
		<code>Void</code>	JSON 配列の終了マーカー (「 <code>]</code> 」) を出力します。
		<code>Void</code>	JSON オブジェクトの終了マーカー (「 <code>}</code> 」) を出力します。
	<code>String <i>fieldName</i> Void</code>		項目名を出力します。
	<code>ID <i>identifier</i> Void</code>		指定された ID 値を出力します。
	<code>String <i>fieldName</i> Void Id <i>identifier</i></code>		指定された項目名と ID 値を使用して、項目名と値のペアを出力します。
		<code>Void</code>	JSON <code>null</code> リテラル値を出力します。
	<code>String <i>fieldName</i> Void</code>		指定された項目名と JSON <code>null</code> リテラル値を使用して、項目名と値のペアを出力します。
	<code>decimal <i>number</i> Void</code>		指定された <code>decimal</code> 値を出力します。
	<code>double <i>number</i> Void</code>		指定された <code>double</code> 値を出力します。
	<code>integer <i>number</i> Void</code>		指定された <code>integer</code> 値を出力します。
	<code>long <i>number</i> Void</code>		指定された <code>long</code> 値を出力します。
	<code>String <i>fieldName</i> Void Decimal <i>number</i></code>		指定された項目名と <code>decimal</code> 値を使用して、項目名と値のペアを出力します。
	<code>String <i>fieldName</i> Void Double <i>number</i></code>		指定された項目名と <code>double</code> 値を使用して、項目名と値のペアを出力します。
	<code>String <i>fieldName</i> Void Integer <i>number</i></code>		指定された項目名と <code>integer</code> 値を使用して、項目名と値のペアを出力します。
	<code>String <i>fieldName</i> Void long <i>number</i></code>		指定された項目名と <code>long</code> 値を使用して、項目名と値のペアを出力します。
	<code>任意の型 <i>object</i> Void</code>		指定された Apex オブジェクトを JSON 形式で出力します。
	<code>String <i>fieldName</i> Void 任意の型 <i>object</i></code>		指定された項目名と Apex オブジェクトを使用して、項目名と値のペアを出力します。
		<code>Void</code>	JSON 配列の開始マーカー (「 <code>[</code> 」) を出力します。
		<code>Void</code>	JSON オブジェクトの開始マーカー (「 <code>{</code> 」) を出力します。

メソッド	引数	戻り値	説明
	String <i>stringValue</i>	Void	指定された文字列値を出力します。
	String <i>fieldName</i>	Void	指定された項目名と string 値を使用して、項目名と値のペアを出力します。
	String <i>stringValue</i>		
	time <i>timeValue</i>	Void	指定された time 値を ISO-8601 形式で出力します。
	String <i>fieldName</i>	Void	指定された項目名と ISO-8601 形式の time 値を使用して、項目名と値のペアを出力します。
	time <i>timeValue</i>		

### JSONGenerator のサンプル

次の例は、**JSONGenerator** のメソッドを使用して JSON 文字列を生成します。





## JSONParser メソッド

JSON 符号化されたコンテンツのパーサーを表します。

### 使用方法

メソッドを使用して、Web サービスコールアウトの JSON 符号化方式の応答など、コールから外部サービスに返される JSON 形式の応答を解析します。

### メソッド

次に、  
クラスのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
	Void	現在のトークンを削除します。  このメソッドがコールされた後は、 へのコールは を返し、 へのコ ルは を返します。 をコ ルして、クリアされたトークンを取得できます。	
	Blob	現在のトークンを BLOB 値として返します。  現在のトークンは 型で Base64 で符号化されている必要があります。	
	Boolean	現在のトークンを boolean 値として返します。  現在のトークンは 型または 型である必要があります。  次の例では、サンプルの JSON 文字列を解析して boolean 値を取得します。	

メソッド	引数	戻り値	説明
		String	<p>現在のトークンに関連付けられた名前を返します。</p> <p>現在のトークンが _____ 型の場合、 同じ値を返します。現在のトークンが値の場合、このトークンより前にある項目名を返します。配列値やルートレベル値など他の値の場合は _____ を返します。</p> <p>次の例では、サンプルの JSON 文字列を解析します。項目値まで処理を進めて、対応する項目名を取得します。</p>
			<p>パーサーが現在指し示しているトークンを返します。現在のトークンがない場合は _____ を返します。</p> <p>次の例では、サンプルの JSON 文字列内のすべてのトークンに対して繰り返し処理を行います。</p>

メソッド	引数	戻り値	説明
	dateTime		<p>現在のトークンを日時値として返します。</p> <p>現在のトークンは 型である 必要があり、ISO-8601 形式の 値を表す必要があります。</p> <p>次の例では、サンプルの JSON 文字列を解析して dateTime 値を取得します。</p>
	date		<p>現在のトークンを日付値として返します。</p> <p>現在のトークンは 型である 必要があり、ISO-8601 形式の 値を表す必要があります。</p> <p>次の例では、サンプルの JSON 文字列を解析して date 値を取得します。</p>

メソッド	引数	戻り値	説明
	decimal		<p>現在のトークンを小数値として返します。</p> <p>現在のトークンは 型 または 型である必要が あり、 型の値に変換可能な数値です。</p> <p>次の例では、サンプルの JSON 文字列を解析して decimal 値を取得します。</p>
	double		<p>現在のトークンを double 値として返します。</p> <p>現在のトークンは 型 である必要があり、 型の値に変換可能な数値で す。</p> <p>次の例では、サンプルの JSON 文字列を解析して double 値を取得します。</p>

メソッド	引数	戻り値	説明
	ID		<p>現在のトークンを ID 値として返します。</p> <p>現在のトークンは 型で、有効な である必要があります。</p> <p>次の例では、サンプルの JSON 文字列を解析して ID 値を取得します。</p>
	integer		<p>現在のトークンを整数値として返します。</p> <p>現在のトークンは 型で、 を表す必要があります。</p> <p>次の例では、サンプルの JSON 文字列を解析して integer 値を取得します。</p>

メソッド	引数	戻り値	説明
			<p>メソッドによりクリアされた最後のトークンを返します。</p>
	long	現在のトークンを long 値として返します。 現在のトークンは <b>型</b> である必要があり、 <b>型</b> の値に変換可能な数値です。 次の例では、サンプルの JSON 文字列を解析して long 値を取得します。	
	String	現在のトークンのテキスト表現を返します。現在のトークンがない場合は <b> </b> を返します。 が一度もコールされていない場合、または パーサーが入力ストリームの終了に達した場合は、現在のトークンは存在しないため <b> </b> を返します。 (ページ 579) の例を参照してください。	
	time	現在のトークンを time 値として返します。 現在のトークンは <b>型</b> である必要があり、ISO-8601 形式の <b> </b> 値を表す必要があります。	



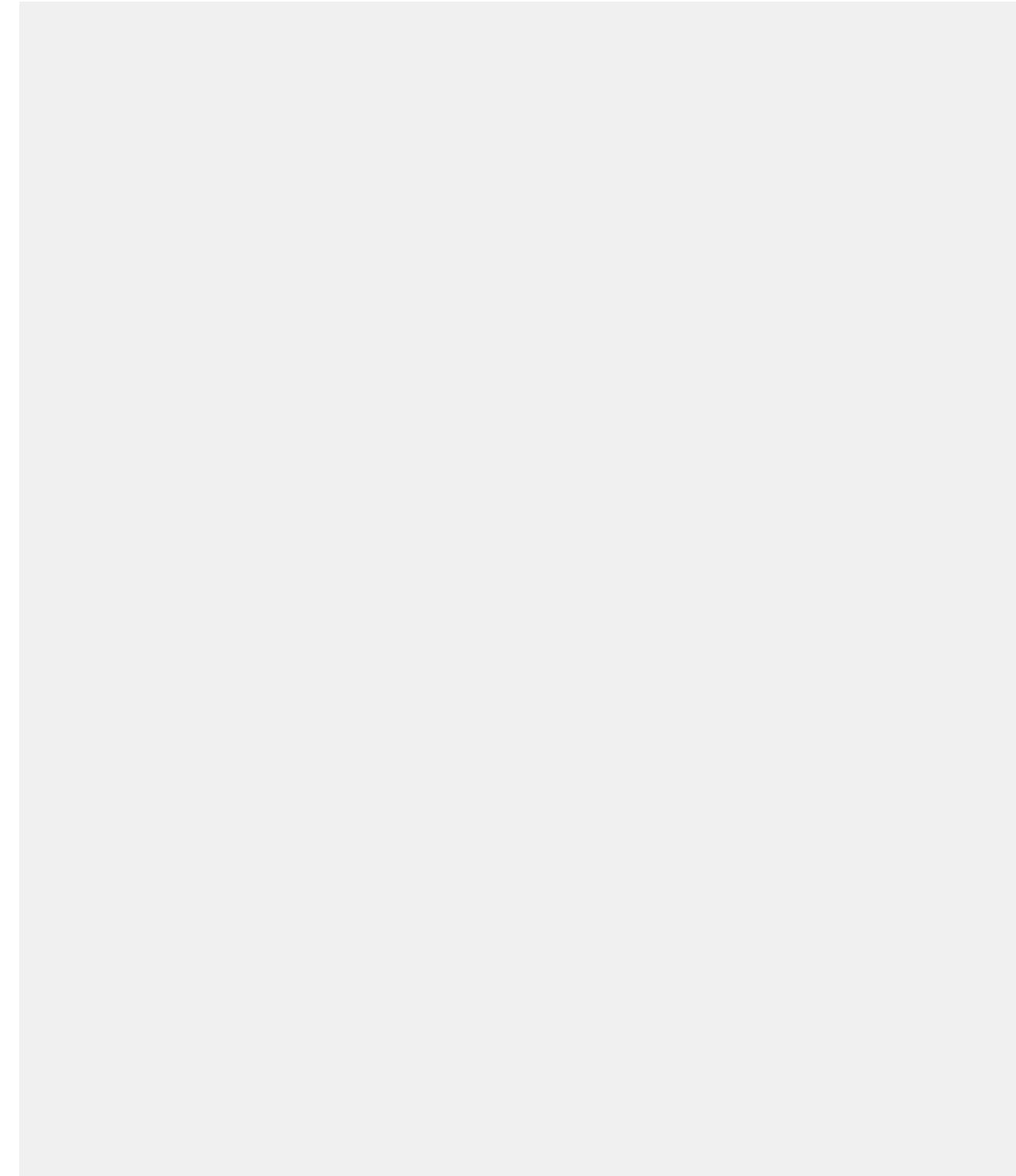
メソッド	引数	戻り値	説明
			<p>だし、Salesforce.com API バージョン 24.0 以前を使用して保存された Apex の場合、このメソッドは存在しない属性に対して実行時例外を発生させます。</p> <p>次の例では、サンプルの JSON 文字列を解析して <code>dateTime</code> 値を取得します。このサンプルを実行するには、次のように Apex クラスを新規作成する必要があります。</p> <p>その後で、クラスメソッドに次のサンプルを挿入します。</p>

メソッド	引数	戻り値	説明
	<i>apexType</i>	anyType	<p>JSON コンテンツを指定された Apex 型のオブジェクトに並列化して、並列化されたオブジェクトを返します。 JSON コンテンツ内のすべての属性は、指定された型に存在する必要があります。<i>apexType</i> 引数は、現在の値を逐次化した後にこのメソッドが返すオブジェクトの型を指定します。</p> <p>解析する JSON コンテンツに、引数で指定された Apex 型に存在しない属性(存在しない項目やオブジェクトなど)が含まれている場合、このメソッドは実行時例外を発生させます。</p> <p>次の例では、サンプルの JSON 文字列を解析して <code>dateTime</code> 値を取得します。このサンプルを実行するには、次のように Apex クラスを新規作成する必要があります。</p> <p>その後で、クラスメソッドに次のサンプルを挿入します。</p>

メソッド	引数	戻り値	説明
	Void		パーサーが現在指示している型と 型のすべての子トークンをスキップします。

#### サンプル: Web サービスコールアウトからの JSON 応答の解析

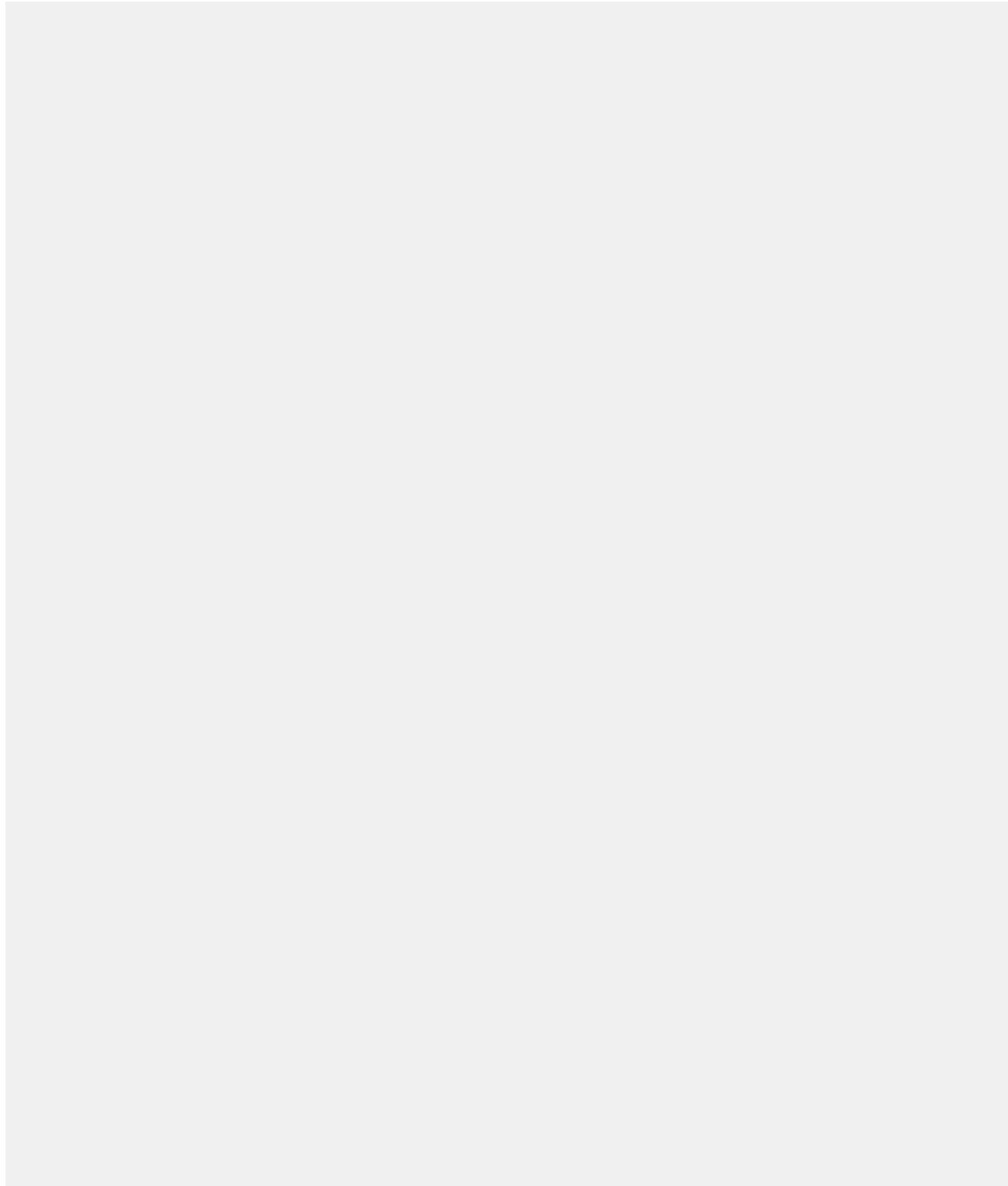
次の例は、メソッドを使用して JSON 形式の応答を解析する方法を示します。この例では、JSON 形式の応答を返す Web サービスへのコールアウトを作成します。次に、応答を解析して、すべての totalPrice 項目値を取得し、価格の総計を計算します。このサンプルを実行するには、Salesforce ユーザインターフェースで Web サービスエンドポイント URL を認証済みリモートサイトとして追加する必要があります。そのためには、Salesforce にログインし、[設定] から [セキュリティのコントロール] > [リモートサイトの設定] をクリックします。

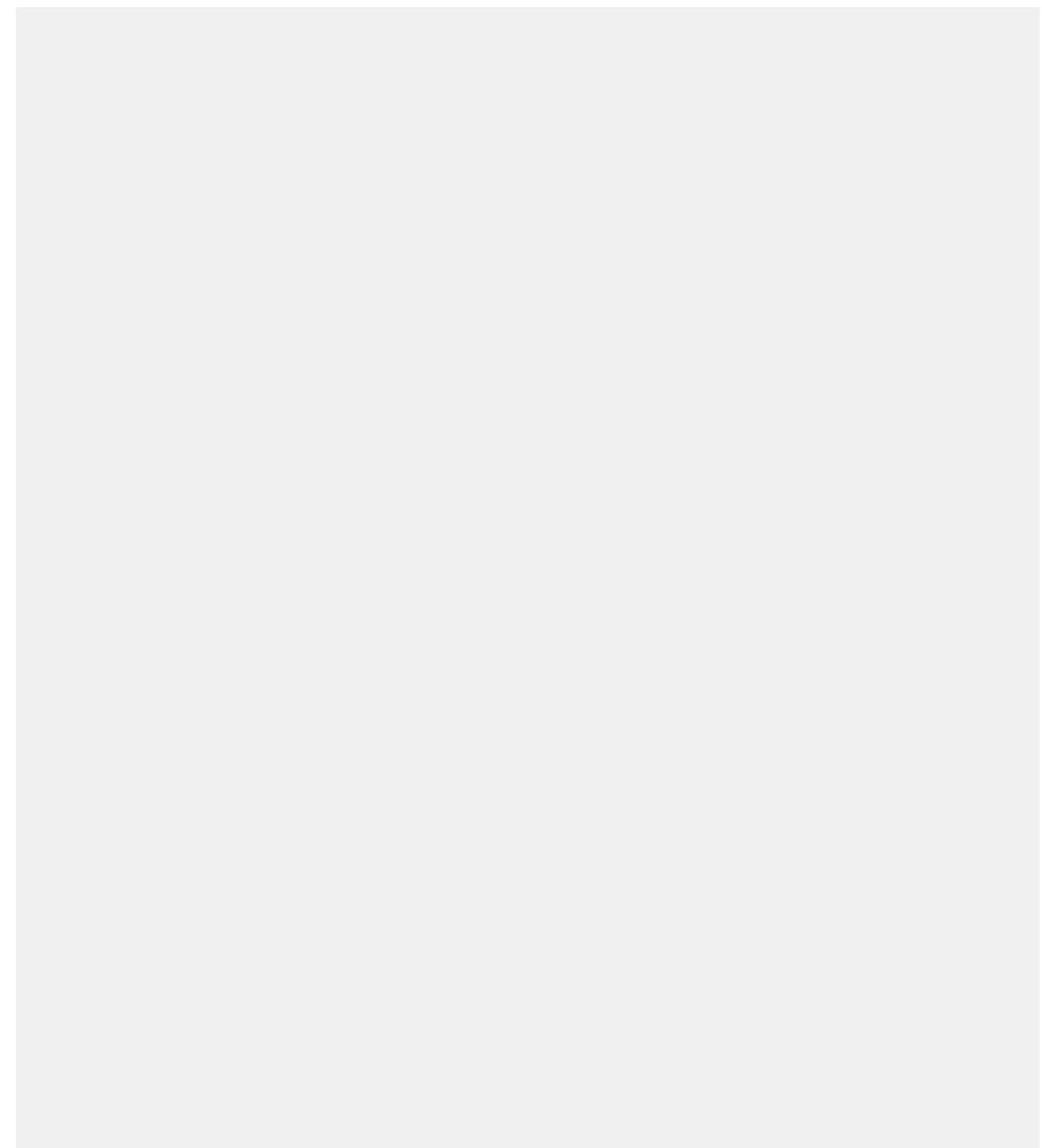


#### サンプル: JSON 文字列の解析とオブジェクトへの並列化

この例では、ハードコードされた JSON 文字列を使用します。これは、前の例のコールアウトで返された JSON 文字列と同じです。この例では、文字列全体が `JSON.deserializeUntyped()` メソッドを使用して `Object` オブジェクトに解析

されます。また、**parse** メソッドを使用して子配列と子オブジェクトをスキップし、リストに含まれる次の同階層の請求書を解析できるようにします。解析されたオブジェクトは、内部クラスとして定義されているクラスのインスタンスです。各請求書には品目が含まれるため、対応する品目型を表すクラスであるクラスも内部クラスとして定義されます。このサンプルコードをクラスに追加して使用します。





## System.JSONToken Enum



enum 値	説明
END_OBJECT	オブジェクト値の終了。このトークンは、「}」が見つかった場合に返されます。
FIELD_NAME	項目名である文字列トークン。
NOT_AVAILABLE	要求されたトークンは使用できません。
START_ARRAY	配列値の開始。このトークンは、「[」が見つかった場合に返されます。
START_OBJECT	オブジェクト値の開始。このトークンは、「{」が見つかった場合に返されます。
VALUE_EMBEDDED_OBJECT	開始オブジェクトトークン START_OBJECT と終了オブジェクトトークン END_OBJECT を含む一般的なオブジェクト構造としてアクセスできないが、生オブジェクトとして表される埋め込みオブジェクト。
VALUE_FALSE	リテラル値「false」。
VALUE_NULL	リテラル値「null」。
VALUE_NUMBER_FLOAT	float 値。
VALUE_NUMBER_INT	integer 値。
VALUE_STRING	string 値。
VALUE_TRUE	「true」文字列リテラルに対応する値。

## 関連リンク

[型メソッド](#)

## Limits メソッド

Apex はマルチテナント環境で実行するため、Apex ランタイムエンジンは、回避 Apex が共有リソースを独占しないようさまざまな制限事項を強制します。

Limits メソッドは、メソッドのコール数やヒープサイズの残りの量など、特定のガバナの具体的な制限を返します。

Limits メソッドは引数を必要としません。Limits メソッドの形式は次のとおりです。

各メソッドには 2 つのバージョンがあります。1 つのバージョンのメソッドは、使用されているリソースの量を返します。もう一方のバージョンは名前に limit が含まれ、使用できるリソースの合計を返します。

[「実行ガバナと制限について」](#) (ページ 337)を参照してください。

名前	戻り値	説明
	Integer	SOQL クエリステートメントで処理された集計クエリの数を返します。
	Integer	SOQL クエリステートメントで処理できる集計クエリの合計数を返します。
	Integer	処理された Web サービスステートメントの数を返します。
	Integer	処理できる Web サービスステートメントの合計数を返します。
	Integer	返された子リレーションオブジェクトの数を返します。
	Integer	返すことができる子リレーションオブジェクトの合計数を返します。
	Integer	現在のトランザクションの Salesforce サーバの累積 CPU 時間 (ミリ秒単位) を返します。
		現在のトランザクションの CPU 使用の時間制限 (ミリ秒単位) を返します。
		テストメソッドなど CPU 時間制限がないコンテキスト内でコールされた場合、 -1 を返します。
	Integer	DML ステートメント、メソッド、および他のメソッドなど、DML 制限にカウントされるすべてのステートメントを使用して処理されたレコードの数を返します。
	Integer	DML ステートメント、メソッド、および他のメソッドなど、DML 制限にカウントされるすべてのステートメントを使用して処理できるレコードの合計数を返します。
	Integer	コールされた DML ステートメント (INSERT、UPDATE、DELETE メソッドなど) の数を返します。
	Integer	コールできる DML ステートメントまたはメソッドの合計数を返します。
	Integer	コールされたメール呼び出し (EmailMessage など) の数を返します。
	Integer	コールできるメール呼び出し (EmailMessage など) の合計数を返します。

名前	戻り値	説明
	Integer	作成された、項目の記述用の API コール (describe) の数を返します。
	Integer	作成できる、項目の記述用の API コール (describe) の合計数を返します。
	Integer	作成された、項目セットの記述用の API コール (describe) の数を返します。
	Integer	作成できる、項目セットの記述用の API コール (describe) の合計数を返します。
	Integer	このメソッドは非推奨です。と同じ値を返します。メソッドの数は、個別の制限ではなく、発行された SOSL クエリの数として追跡されるようになりました。
	Integer	このメソッドは非推奨です。と同じ値を返します。メソッドの数は、個別の制限ではなく、発行された SOSL クエリの数として追跡されるようになりました。
	Integer	実行された (必ずしも完了しない) アノテーションがあるメソッドの数を返します。
	Integer	実行できる (必ずしも完了しない) アノテーションがあるメソッドの合計数を返します。
	Integer	ヒープに使用されたメモリのおおよその容量 (バイト単位) を返します。
	Integer	ヒープに使用できるメモリの合計容量 (バイト単位) を返します。
	Integer	実行されたやり取りの総数を返します。
	Integer	実行できるやり取りの総数を返します。
	Integer	発行された SOQL クエリの数を返します。
	Integer	発行できる SOQL クエリの合計数を返します。
	Integer	返された PicklistEntry オブジェクトの数を返します。
	Integer	返すことができる PicklistEntry オブジェクトの合計数を返します。
	Integer	メソッドで返されたレコードの数を返します。

名前	戻り値	説明
	Integer	メソッドで返されたレコードの合計数を返します。
	Integer	SOQL クエリの発行で返されたレコード数を返します。
	Integer	SOQL クエリの発行で返すことができるレコードの合計数を返します。
	Integer	返された RecordTypeInfo オブジェクトの数を返します。
	Integer	返すことができる RecordTypeInfo オブジェクトの合計数を返します。
	Integer	このメソッドは非推奨です。 同じ値を返します。メソッドの数は、個別の制限ではなく、発行された DML ステートメントの数として追跡されるようになりました。
	Integer	実行された Apex ステートメントの数を返します。
	Integer	実行できる Apex ステートメントの合計数を返します。
	Integer	発行された SOSL クエリの数を返します。
	Integer	発行できる SOSL クエリの合計数を返します。

## Math メソッド

次に、Math のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	decimal <i>d</i>	Decimal	指定された decimal の絶対値を返します。
	double <i>d</i>	Double	指定された double の絶対値を返します。
	Integer <i>i</i>	Integer	指定された integer の絶対値を返します。次に例を示します。
	Long <i>l</i>	Long	指定された long の絶対値を返します。
	decimal <i>d</i>	Decimal	角のアークコサインを $0.0 \sim \pi/2$ の範囲で返します。
	double <i>d</i>	Double	角のアークコサインを $0.0 \sim \pi/2$ の範囲で返します。
	decimal <i>d</i>	Decimal	角のアークサインを $-\pi/2 \sim \pi/2$ の範囲で返します。
	double <i>d</i>	Double	角のアークサインを $-\pi/2 \sim \pi/2$ の範囲で返します。
	decimal <i>d</i>	Decimal	角のアークタンジェントを $-\pi/2 \sim \pi/2$ の範囲で返します。
	double <i>d</i>	Double	角のアークタンジェントを $-\pi/2 \sim \pi/2$ の範囲で返します。
	decimal <i>x</i> Decimal <i>y</i>	Decimal	直交座標 ( <i>x</i> および <i>y</i> ) を極 ( <i>r</i> および <i>theta</i> ) に変換します。このメソッドは、 <i>x/y</i> のアークタンジェントを $-\pi \sim \pi$ の範囲で計算することで、相 <i>theta</i> を計算します。
	double <i>x</i> Double <i>y</i>	Double	直交座標 ( <i>x</i> および <i>y</i> ) を極 ( <i>r</i> および <i>theta</i> ) に変換します。このメソッドは、 <i>x/y</i> のアークタンジェントを $-\pi \sim \pi$ の範囲で計算することで、相 <i>theta</i> を計算します。
	decimal <i>d</i>	Decimal	指定された decimal の立方根を返します。負の値の立方根は、値の大きさの平方根を負にしたものです。

名前	引数	戻り値	説明
	double $d$	Double	指定された double の立方根を返します。負の値の立方根は、値の大きさの平方根を負にしたもので
	decimal $d$	Decimal	す。 最も小さい(負の無限大に最も近い) decimal を返します。引数より大きく、数学的整数と等しい値になります。
	double $d$	Double	最も小さい(負の無限大に最も近い) double を返します。引数より大きく、数学的整数と等しい値になります。
	decimal $d$	Decimal	$d$ で指定された角の三角関数のコサインを返します。
	double $d$	Double	$d$ で指定された角の三角関数のコサインを返します。
	decimal $d$	Decimal	$d$ の双曲線コサインを返します。 $d$ の双曲線コサインは、 $(e^d + e^{-d})/2$ となるように定義します。ここで $e$ はオイラーの数値です。
	double $d$	Double	$d$ の双曲線コサインを返します。 $d$ の双曲線コサインは、 $(e^d + e^{-d})/2$ となるように定義します。ここで $e$ はオイラーの数値です。
	decimal $d$	Decimal	指定した decimal の累乗まで乗算したオイラーの数値 $e$ を返します。
	double $d$	Double	指定した double の累乗まで乗算したオイラーの数値 $e$ を返します。
	decimal $d$	Decimal	最も大きい(正の無限大に最も近い) decimal を返します。引数より小さく、数学的整数と等しい値になります。
	double $d$	Double	最も大きい(正の無限大に最も近い) double を返します。引数より小さく、数学的整数と等しい値になります。
	decimal $d$	Decimal	指定された decimal の自然対数 (base $e$ ) を返します。
	double $d$	Double	指定された double の自然対数 (base $e$ ) を返します。
	decimal $d$	Decimal	指定された decimal の対数 (base 10) を返します。
	double $d$	Double	指定された double の対数 (base 10) を返します。

名前	引数	戻り値	説明
	decimal <i>d1</i> Decimal <i>d2</i>	Decimal	指定された 2 つの decimal の大きい方を返します。 次に例を示します。
	double <i>d1</i> Double <i>d2</i>	Double	指定された 2 つの double の大きい方を返します。
	Integer <i>i1</i> Integer <i>i2</i>	Integer	指定された 2 つの integer の大きい方を返します。
	Long <i>l1</i> long <i>l2</i>	Long	指定された 2 つの long の大きい方を返します。
	decimal <i>d1</i> Decimal <i>d2</i>	Decimal	指定された 2 つの decimal の小さい方を返します。 次に例を示します。
	double <i>d1</i> Double <i>d2</i>	Double	指定された 2 つの double の小さい方を返します。
	Integer <i>i1</i> Integer <i>i2</i>	Integer	指定された 2 つの integer の小さい方を返します。
	Long <i>l1</i> long <i>l2</i>	Long	指定された 2 つの long の小さい方を返します。
	Integer <i>i1</i> Integer <i>i2</i>	Integer	<i>i1</i> を <i>i2</i> で除算した余りを返します。次に例を示します。
	Long <i>l1</i> long <i>l2</i>	Long	<i>l1</i> を <i>l2</i> で除算した余りを返します。

名前	引数	戻り値	説明
	double <i>d</i> Double <i>exp</i>	Double	<i>exp</i> の累乗まで乗算した最初の double 値を返します。
		Double	0.0 以上 1.0 未満の正の double を返します。
	decimal <i>d</i>	Decimal	<i>d</i> に最も近く、数学的整数と等しい値を返します。
	double <i>d</i>	Double	<i>d</i> に最も近く、数学的整数と等しい値を返します。
	double <i>d</i>	Integer	使用しません。このメソッドは、Winter '08 リリースの時点で廃止されています。代わりに、 を使用してください。指定された double に最も近い integer を返します。結果が -2,147,483,648 未満または 2,147,483,647 より大きい場合、Apex はエラーを生成します。
	decimal <i>d</i>	Integer	decimal の丸められた近似値を返します。数値は、均等丸めモードを使用して、「最も近い近似値」である整数に丸められます。ただし、2つの近似値が等距離にある場合は、このモードでは偶数の近似値に丸められます。  この丸めモードは、連続する計算に対して繰り返し適用される場合、統計的に累積エラーを最小化します。均等丸めモードについての詳細は、「 <a href="#">丸めモード</a> 」を参照してください。
			例:     
	decimal <i>d</i>	Long	decimal の丸められた近似値を返します。数値は、均等丸めモードを使用して、「最も近い近似値」である整数に丸められます。ただし、2つの近似値が等距離にある場合は、このモードでは偶数の近似値に丸められます。



名前	引数	戻り値	説明
	decimal <i>d</i>	Decimal	<i>d</i> の双曲線タンジェントを返します。 <i>d</i> の双曲線タンジェントは $(e^x - e^{-x})/(e^x + e^{-x})$ となるように定義します。ここで $e$ はオイラーの数値です。つまり、 と等しくなります。正確な の絶対値は常に 1 より小さい値です。
	double <i>d</i>	Double	<i>d</i> の双曲線タンジェントを返します。 <i>d</i> の双曲線タンジェントは $(e^x - e^{-x})/(e^x + e^{-x})$ となるように定義します。ここで $e$ はオイラーの数値です。つまり、 と等しくなります。正確な の絶対値は常に 1 より小さい値です。

## MultiStaticResourceCalloutMock メソッド

HTTP コールアウトのテストで複数のリソースを使用して、擬似応答を指定するために使用されるユーティリティクラスです。

### 使用方法

このクラスのメソッドを使用して、HTTP コールアウトのテストでの応答のプロパティを設定します。各エンドポイントのリソースを指定できます。 「[使用した HTTP コールアウトのテスト](#)」を参照してください。

### メソッド

次に、  
クラスのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
	String <i>headerName</i> String <i>headerValue</i>	Void	擬似応答に指定されたヘッダー名と値 を設定します。
	String <i>endpoint</i> String <i>resourceName</i>	Void	エンドポイントに対応する指定された 静的リソースを設定します。静的リソースにはレスポンスボディが含まれます。
	String <i>httpStatus</i>	Void	応答に指定された HTTP 状況を設定し ます。

メソッド	引数	戻り値	説明
	Integer <i>httpStatusCode</i>	Void	応答に指定された HTTP 状況コードを設定します。

## 関連リンク

[StaticResourceCalloutMock メソッド](#)

## Apex REST

Apex REST では、Apex のカスタム Web サービスを実装し REST アーキテクチャを通じて公開できます。Apex クラスを REST サービスとして公開するには、最初にアノテーションを付加したクラスを定義して REST リソースとして公開します。同様に、他のアノテーションもクラスメソッドに追加して、REST を通じて公開します。たとえば、アノテーションをメソッドに追加して、HTTP 要求から呼び出すことができる REST リソースとして公開できます。

クラス	説明
	オブジェクトとオブジェクトを含みます。
	HTTP 要求から Apex RESTful Web サービスマソッドにデータを渡す場合に使用されるオブジェクトを表します。
	Apex RESTful Web サービスマソッドから HTTP 応答にデータを渡す場合に使用されるオブジェクトを表します。

## RestContext メソッド

オブジェクトとオブジェクトを含みます。

## 使用方法

クラスを使用して、Apex REST メソッドのオブジェクトにアクセスします。

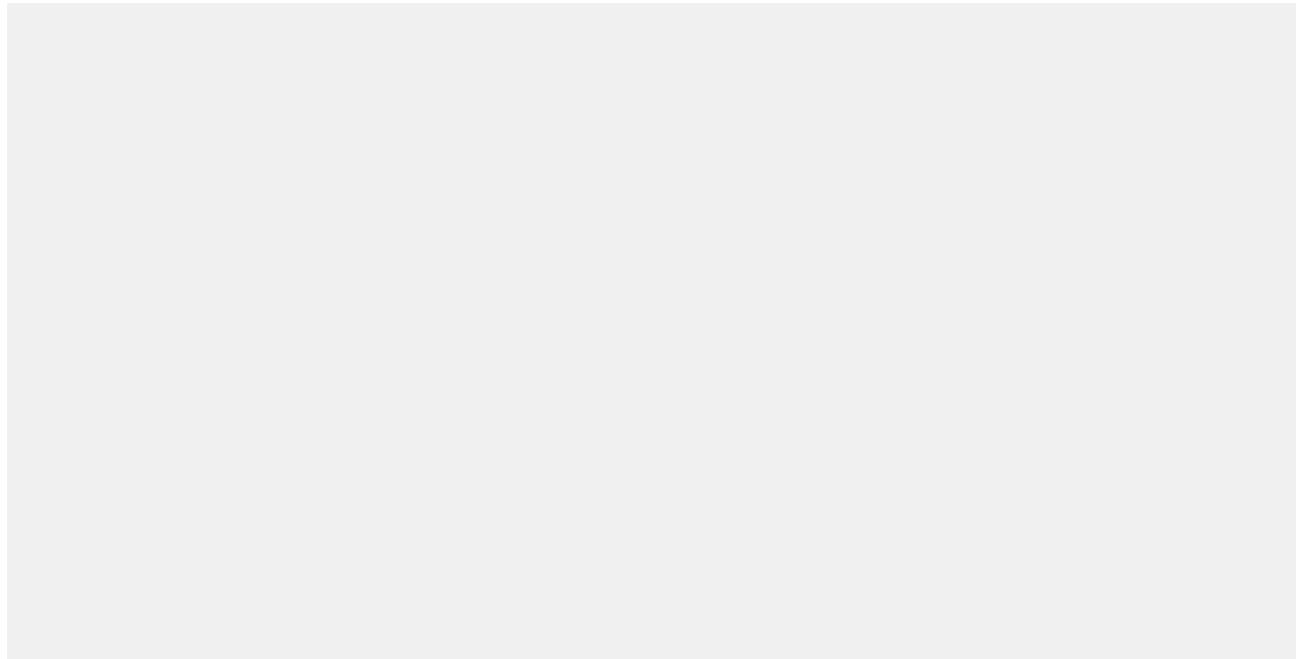
## プロパティ

次に、クラスのプロパティを示します。

名前	戻り値	説明
		Apex REST メソッドに対し ます。
		Apex REST メソッドに対し ます。

## サンプル

次の例では、 を使用して、Apex REST メソッドの オブジェクトと オブジェクトにアクセスする方法を示します。



## 関連リンク

[Apex REST の概要](#)

## RestRequest メソッド

HTTP 要求から Apex RESTful Web サービスマソッドにデータを渡す場合に使用されるオブジェクトを表します。

## 使用方法

クラスを使用して、REST アノテーションの 1つを使用して定義される Apex RESTful Web サービスマソッドに要求データを渡します。

## メソッド

次に、

クラスのインスタンスマソッドを示します。



**メモ:** 実行時に、ヘッダーまたはパラメータは、対応するプロパティに自動的に並列化されるため、通常オブジェクトに追加する必要はありません。次のメソッドは、Apex REST クラスをテストするユニットを対象とします。これらのメソッドを使用して、ヘッダーまたはパラメータ値をオブジェクトに追加でき、REST メソッドコールを再作成する必要はありません。

メソッド	引数	戻り値	説明
	String <i>name</i> 、 String <i>value</i>	Void	<p>要求ヘッダー対応付けにヘッダーを追加します。このメソッドは、Apex REST クラスのテストユニットを対象としています。</p> <p>次のヘッダーは許可されていません。</p> <ul style="list-style-type: none"> <li>Cookie</li> <li>set-cookie</li> <li>set-cookie2</li> <li>content-length</li> <li>認証</li> </ul> <p>いずれかが使用された場合は Apex 例外が返されます。</p>
	String <i>name</i> 、 String <i>value</i>	Void	要求パラメータ対応付けにパラメータを追加します。このメソッドは、Apex REST クラスのテストユニットを対象としています。

## プロパティ

次に、

クラスのプロパティを示します。



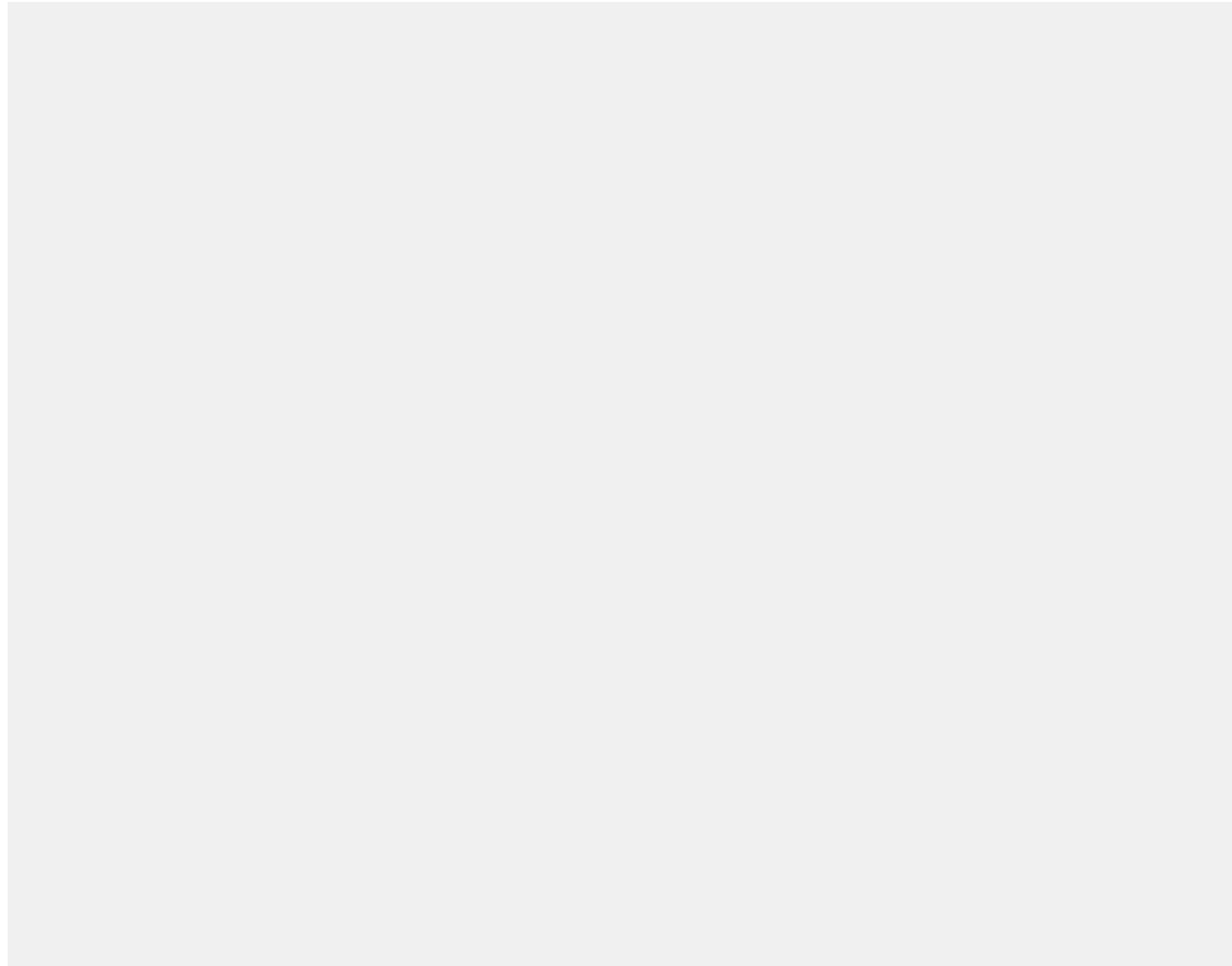
**メモ:** List プロパティと Map プロパティは参照専用ですが、内容は参照・更新が可能です。変更するには、Collection メソッドを直接コールするか、前の表に示した、関連付けられたメソッドを使用できます。

名前	戻り値	説明
	string	<p>要求が受け取るヘッダーを返します。</p> <p>サポートされる次の HTTP 要求メソッドの 1 つを返します。</p> <ul style="list-style-type: none"> <li>Delete</li> <li>GET</li> <li>HEAD</li> <li>PATCH</li> <li>POST</li> <li>PUT</li> </ul>

名前	戻り値	説明
		要求が受け取るパラメータを返します。
	string	要求を行うクライアントの IP アドレスを返します。
	Blob	リクエストボディを返すか、設定します。 Apex メソッドにパラメータがない場合、Apex REST は HTTP リクエストボディをプロパティにコピーします。パラメータがある場合、Apex REST はデータをそれらのパラメータに並列化しようとします。ただし、データはプロパティには並列化されません。
	string	HTTP 要求文字列内のホスト名の後の文字列をすべて返すか、設定します。たとえば、要求文字列が <code>https://instance.salesforce.com/services/apexrest/Account/</code> の場合、 <code>services/apexrest/Account/</code> です。
	String	要求の REST リソースパスを返します。たとえば、Apex REST クラスが <code>Account</code> を定義している場合、 <code>Account</code> のプロパティはを返します。

### サンプル: REST アノテーションが付加されたメソッドを含む Apex クラス

次の例では、Apex の Apex REST API の実装方法を示します。このクラスが公開する GET、DELETE、および POST の 3 つのメソッドはそれぞれ、異なる HTTP 要求を処理します。HTTP 要求を発行して、クライアントからアノテーションが付加されたこれらのメソッドをコールできます。



## 関連リンク

[Apex REST の概要](#)

## RestResponse メソッド

Apex RESTful Web サービスマソッドから HTTP 応答にデータを渡す場合に使用されるオブジェクトを表します。

### 使用方法

クラスを使用して、[REST アノテーション](#)(ページ 211)の 1 つを使用して定義される Apex RESTful Web サービスマソッドの応答データを渡します。

### メソッド

次に、

クラスのインスタンスマソッドを示します。



メモ: 実行時に、ヘッダーは、対応するプロパティに自動的に並列化されるため、通常オブジェクトに追加する必要はありません。

メソッド	引数	戻り値	説明
	String <i>name</i> , String <i>value</i>	Void	<p>応答ヘッダー対応付けにヘッダーを追加します。</p> <p>次のヘッダーは許可されていません。</p> <ul style="list-style-type: none"> <li>• Cookie</li> <li>• set-cookie</li> <li>• set-cookie2</li> <li>• content-length</li> <li>• 認証</li> </ul> <p>いずれかが使用された場合は Apex 例外が返されます。</p>

## プロパティ

次に、クラスのプロパティを示します。



メモ: List プロパティと Map プロパティは参照専用ですが、内容は参照・更新が可能です。変更するには、Collection メソッドを直接コールするか、前の表に示した、関連付けられたメソッドを使用できます。

名前	戻り値	説明
		応答に送信されるヘッダーを返します。
	Blob	<p>レスポンスボディを返すか、設定します。</p> <p>応答は、メソッドの戻り値の逐次化された形式、または、次のルールに基づいた responseBody プロパティの値です。</p> <ul style="list-style-type: none"> <li>• メソッドが void を返す場合、Apex REST は、プロパティの応答を返します。</li> <li>• メソッドが値を返す場合、Apex REST は、戻り値を応答として逐次化します。</li> </ul>
	Integer	応答状況コードを返すか、設定します。次の表に、サポートされる状況コードを示します。HTTP 仕様で定義されている状況コードのサブセットです。

## 状況コード

次に、有効な応答状況コードを示します。状況コードは、

プロパティから返されます。



メモ: プロパティを表に示されていない値に設定した場合、HTTP 状況 500 とエラーメッセージ「Invalid status code for HTTP response: nnn」が返されます。nnn は無効な状況コード値です。

状況コード	説明
200	OK
201	CREATED
202	ACCEPTED
204	NO_CONTENT
206	PARTIAL_CONTENT
300	MULTIPLE_CHOICES
301	MOVED_PERMANENTLY
302	FOUND
304	NOT_MODIFIED
400	BAD_REQUEST
401	UNAUTHORIZED
403	FORBIDDEN
404	NOT_FOUND
405	METHOD_NOT_ALLOWED
406	NOT_ACCEPTABLE
409	CONFLICT
410	GONE
412	PRECONDITION_FAILED
413	REQUEST_ENTITY_TOO_LARGE
414	REQUEST_URL_TOO_LARGE
415	UNSUPPORTED_MEDIA_TYPE
417	EXPECTATION_FAILED
500	INTERNAL_SERVER_ERROR
503	SERVER_UNAVAILABLE

## 関連リンク

[Apex REST の概要](#)

## Search メソッド

次に、Search のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>query</i>	sObject[sObject[]]	実行時に動的 SOSL クエリを作成します。このメソッドは、通常の割り当てステートメントやループなど、静的 SOSL クエリが使用可能な場合に使用できます。 詳細は、「 <a href="#">動的 SOQL</a> 」を参照してください。

## StaticResourceCalloutMock メソッド

HTTP コールアウトのテストで擬似応答を指定するために使用するユーティリティクラスです。

### 使用方法

このクラスのメソッドを使用して、HTTP コールアウトのテストでの応答のプロパティを設定します。  
[「  
を使用した HTTP コールアウトのテスト」](#)を参照してください。

### メソッド

次に、クラスのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
	String <i>headerName</i> String <i>HeaderValue</i>	Void	擬似応答に指定されたヘッダー名と値を設定します。
	String <i>resourceName</i>	Void	レスポンスボディを含む、指定された静的リソースを設定します。
	String <i>httpStatus</i>	Void	応答に指定された HTTP 状況を設定します。
	Integer <i>httpStatusCode</i>	Void	応答に指定された HTTP 状況コードを設定します。

## 関連リンク

[MultiStaticResourceCalloutMock メソッド](#)

## System メソッド

次に、System の静的メソッドを示します。



**メモ:** *AnyDataType* は、プリミティブ、オブジェクトレコード、配列、対応付け、セット、または特殊な値 を表します。

名前	引数	戻り値	説明

String *Job\_ID*  
Void

*condition* が true であることを確認します。true ではない場合、致命的なエラーが返され、コードの実行が停止します。返されるエラーには、必要に応じて、最後の引数に指定するカスタムメッセージを含めることもできます。

任意のデータ型  
*opt\_msg*

アサーションの失敗は、例外としてログに記録されても、try/catch ブロックを使用して捕捉することはできません。

名前	引数	戻り値	説明
	任意のデータ型 <i>x</i> 任意のデータ型 <i>y</i> 任意のデータ型 <i>opt_msg</i>	Void	最初の 2 つの引数 <i>x</i> と <i>y</i> が同じであることを確認します。同じではない場合、致命的なエラーが返され、コードの実行が停止します。返されるエラーには、必要に応じて、最後の引数に指定するカスタムメッセージを含めることもできます。  <i>x</i> 引数には期待値を指定します。  <i>y</i> 引数には実際の値を指定します。  アーサーションの失敗は、例外としてログに記録されても、try/catch ブロックを使用して捕捉することはできません。
	任意のデータ型 <i>x</i> 任意のデータ型 <i>y</i> 任意のデータ型 <i>opt_msg</i>	Void	最初の 2 つの引数 <i>x</i> と <i>y</i> が異なることを確認します。同じ場合、致命的なエラーが返され、コードの実行が停止します。返されるエラーには、必要に応じて、最後の引数に指定するカスタムメッセージを含めることができます。  <i>x</i> 引数には期待値を指定します。  <i>y</i> 引数には実際の値を指定します。  アーサーションの失敗は、例外としてログに記録されても、try/catch ブロックを使用して捕捉することはできません。
		System.PageReference	現在のページへの参照を返します。 Visualforce ページで使用します。詳細は、「 <a href="#">クラス</a> 」(ページ 869)を参照してください。
		Long	現在の時間をミリ秒単位で返します。現在の時刻と 1970 年 1 月 1 日 午前 0 時 (UTC)との差異で表されます。
	任意のデータ型 <i>msg</i>	Void	引数 <i>msg</i> を実行デバッグログに string 形式で書き込みます。 ログレベルが使用されます。  Apex コードのログレベルが 以上に設定されていると、この debug ステートメントのメッセージはデバッグログに書き込まれます。

名前	引数	戻り値	説明
			<p><i>msg</i> 引数が string でない場合、メソッドは <code>String.valueOf()</code> をコールして引数を string に変換します。</p> <p>メソッドは、その引数に対するメソッド (利用可能な場合)、または引数がユーザ定義型の場合は上書きされた <code>toString()</code> をコールします。それ以外のメソッドが利用できない場合は引数の文字列表現を返します。</p> <p>対応付けまたはセットが印刷されると、出力はキー順に並び替えられ、大かっこ ( ) で囲されます。配列またはリストが印刷されると、出力は丸かっこ ( ) で囲されます。</p> <p> メモ: このコードは、Apex コードカバー率の対象とはみなされません。</p> <p>ログレベルについての詳細は、Salesforce オンラインヘルプの「デバッグログの検索条件の設定」を参照してください。</p>
	<p>Enum <code>logLevel</code> Void 任意のデータ型 <i>msg</i></p>		<p>指定されたログレベルで、引数 <i>msg</i> を実行デバッグログに string 形式で書き込みます。</p> <p> メモ: このコードは、Apex コードカバー率の対象とはみなされません。</p> <p><code>logLevel</code> 引数は、このメソッドに設定するログレベルです。</p> <p><i>msg</i> 引数は、実行デバッグログに string 形式で書き込むメッセージまたはオブジェクトです。</p> <p>有効なログレベルは次のとおりです (低いものから順に並べてあります)。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> </ul>

名前	引数	戻り値	説明
			<p>•</p> <p>•</p> <p>•</p> <p>•</p> <p>ログレベルは累積です。たとえば、Apex コードに最も低いレベル <code>System.debug</code> が指定されている場合、ログレベルが <code>System.debug</code> メソッドのみが記録されます。次に低いレベル <code>System.info</code> が指定されている場合、デバッグログには <code>ERROR</code> または <code>INFO</code> として指定されている <code>System.debug</code> メソッドのみが含まれます。</p> <p><code>msg</code> 引数が <code>string</code> でない場合、<code>String.valueOf</code> メソッドは <code>String.valueOf</code> をコールして引数を <code>string</code> に変換します。</p> <p><code>Method</code> メソッドは、その引数に対する <code>Method</code> メソッド (利用可能な場合)、または引数がユーザ定義型の場合は上書きされた <code>Method</code> をコールします。それ以外の、<code>Object</code> メソッドが利用できない場合は引数の文字列表現を返します。</p> <p>ログレベルについての詳細は、Salesforce オンラインヘルプの「デバッグログの検索条件の設定」を参照してください。</p>
<code>System.ApplicationReadWriteMode</code>		<code>Salesforce.com</code>	<p><code>Salesforce.com</code> アップグレードとダウンタイム中に、組織の参照・更新モードセットを返します。このメソッドは、enum <code>System.ApplicationReadWriteMode</code> を返します。有効な値は、次のとおりです。</p> <p>•</p> <p>•</p> <p>•</p> <p>•</p> <p>は、5 分アップグレードの一部として提供されます。</p>
	<code>boolean</code>		現在実行中のコードが Apex の一括処理ジョブから呼び出された場合は <code>true</code> を

名前	引数	戻り値	説明
			返します。呼び出されていない場合は を返します。
		boolean	future メソッドは、Apex の一括処理ジョブから呼び出せません。future メソッドの呼び出し前にこのメソッドを使用して、現在実行中のコードが Apex 一括処理ジョブであるかどうかを確認します。
		boolean	現在実行中のコードがアノテーション が付加されたメソッドに含まれる コードから呼び出された場合は を返します。呼び出されていない場合は を返します。
		Datetime	future メソッドは、別の future メソッド から呼び出せないため、future メソッド の呼び出し前にこのメソッドを使用して、 現在のコードが future メソッドのコ ンテキスト内で実行されているかどうか を確認します。
		boolean	現在実行中のコードが Apex のスケ ジュール済みジョブから呼び出された場 合は を返します。呼び出され ていない場合は を返します。
	List<WorkItemIDs> <i>WorkItemIDs</i> String <i>Action</i> String <i>Comments</i> String <i>NextApprover</i>	Datetime	現在の日付と時刻を GMT タイムゾーン で返します。
	date <i>dt</i>	Integer	作業項目 ID のリストを処理します。詳 細は、「 <a href="#">Apex 承認プロセスクラス</a> 」 (ページ 650)を参照してください。
			指定日より前に完了、中止、または失敗 状況で実行を終了したジョブの非同期 Apex ジョブレコード ( <a href="#">AsyncApexJob</a> の レコード) を削除し、削除したレコード 数を返します。  <i>dt</i> 引数には、どの日付以前の古いレコード を削除するのか、日付を指定します。

名前	引数	戻り値	説明
			<p>日付の比較は、<code>AsyncApexJob</code> の項目 (GMT タイムゾーン)に基づいて比較されます。</p> <p>システムは、実行を終了した、8日以上前のジョブについて非同期ジョブレコードをクリーンアップします。このメソッドを使用し、より多くのレコードをクリーンアップすることで、<code>AsyncApexJob</code> のサイズをさらに小さくすることができます。</p> <p>このメソッドの各実行は、DML ステートメントのガバナ制限に單一行としてカウントされます。詳細は、「<a href="#">実行ガバナと制限について</a>」を参照してください。</p> <p>次の例では、前日以前に終了したジョブのすべてのジョブレコードを削除する方法を示します。</p>
		<code>System.Version</code>	<p>パッケージのメジャーバージョン番号とマイナーバージョン番号の2つの番号で構成されるバージョンを返します。</p> <p>このメソッドでは、コール元のコードがパッケージを参照するパッケージのインストール済みインスタンスのバージョンを特定できます。コール元のコードが保持するバージョンに基づいて、パッケージコードの動作をカスタマイズできます。</p> <p>未管理パッケージの場合、 メソッドはサポートされません。未管理パッケージからコールする場合、例外が発生します。</p>

名前	引数	戻り値	説明
	ID <i>userID</i> boolean <i>send_user_email</i>	System.ResetPasswordResult	<p>指定されたユーザのパスワードをリセットします。ユーザが新しいパスワードでログインすると、新しいパスワードを入力してセキュリティに関する質問および回答を選択するように求められます。</p> <p><i>send_user_email</i> に 指定すると、パスワードがリセットされたことをユーザーに通知するメールが送信されます。このメールには、新しいパスワードを使用して Salesforce にサインオンするためのリンクが含まれます。ユーザに対して、ログイン時に新しいパスワードを入力するように求めない場合、を使用します。</p> <p> 警告: このメソッドを使用する場合は注意が必要です。また、この機能をエンドユーザに公開しないでください。</p>
	System.Version <i>version</i>	Void	<p>現在のパッケージバージョンを、引数で指定されたパッケージバージョンに変更します。</p> <p>パッケージ開発者は、コードをアップグレードしながら、以前のパッケージバージョンのクラスおよびトリガの既存の動作を引き続きサポートするために、<a href="#">Version メソッド</a>を使用できます。Apex クラスおよびトリガは、クラスまたはトリガが参照するインストール済みの各管理パッケージのバージョン設定で保存されます。</p> <p>このメソッドを使用して、AppExchange にアップロードする異なるパッケージバージョンのコンポーネントの動作をテストします。このメソッドは、異なるパッケージバージョンの動作をテストできるように、テストメソッドのメジャー バージョン番号とマイナーバージョン番号の2つで構成されるバージョン番号を効率的に設定します。</p>

名前	引数	戻り値	説明
			<p>テストメソッドでは　　のみ使用できます。トランザクションで、このメソッドに対するコール数の制限はありません。このメソッドの使用例は、「<a href="#">パックエージバージョンの動作のテスト</a>」を参照してください。</p>
	User <i>user_var</i>	Void	<p>現在のユーザを指定されたユーザに変更します。指定されたユーザの権限とコード共有のすべてが、　　の実行時に強制されます。テストメソッドではのみ使用できます。</p> <p> メモ: メソッドは、ユーザライセンスの制限を無視します。組織に追加ユーザライセンスがない場合でも、　　で新しいユーザを作成できます。</p>
			<p>メソッドは、パラメータとして渡されたユーザがインスタンス化済みでまだ挿入されていない場合は、暗黙的に挿入します。</p> <p>詳細は、「<a href="#">メソッドの使用</a>」(ページ 234)を参照してください。</p> <p>DML 操作を　　ブロックで囲んで、　　を使用して混合 DML 操作をテストで実行することもできます。この方法では、設定オブジェクトを他の sObject と一緒に挿入または更新しようとすると返される混在 DML エラーを回避できます。「<a href="#">DML 操作で同時に使用できない sObject</a>」を参照してください。</p> <p> メモ: の各コールは、プロセスで発行される DML ステートメントの合計数にカウントされます。</p>
	String <i>JobName</i>	string	
	String <i>CronExpression</i>		<p>インターフェースを実装する Apex クラスで　　を使用して、<i>CronExpression</i> によって指定された時間に実行されるようにクラスをスケ</p>



名前	引数	戻り値	説明
			<p><i>minutesFromNow</i> 引数は、ジョブの実行が開始されるまでの分単位の期間です。この引数は0よりも大きい値にする必要があります。</p> <p> メモ: Salesforce は、指定された時間に実行されるようにクラスをスケジュール設定します。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。</p> <p><a href="#">メソッドを使用して、スケジュールされた後にジョブを停止します。</a></p> <p>「<a href="#">メソッドの使用</a>」の例を参照してください。</p>
Database.Batchable<T>	<p>string <i>batchable</i></p> <p>String <i>jobName</i></p> <p>Integer <i>minutesFromNow</i></p> <p>Integer <i>scopeSize</i></p>		<p>指定された時間が経過した後に、指定されたジョブ名と範囲サイズを使用して一括処理ジョブが1回実行されるようにスケジュール設定します。スケジュール済みジョブ ID (CronTrigger ID) を返します。</p> <p><i>batchable</i> 引数は、インターフェースを実装する一括処理クラスです。</p> <p><i>jobName</i> 引数は、このメソッドで開始されるジョブの名前です。</p> <p><i>minutesFromNow</i> 引数は、ジョブの実行が開始されるまでの分単位の期間です。</p> <p><i>scopeSize</i> 引数は、一括処理メソッドに渡すレコードの数です。</p> <p> メモ: Salesforce は、指定された時間に実行されるようにクラスをスケジュール設定します。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。</p>

名前	引数	戻り値	説明
			<p>スケジュールされたすべての Apex 制限は、 を使用してスケジュールされた一括処理ジョブに適用されます。一括処理ジョブの実行が開始すると、すべての一括処理ジョブ制限が適用され、ジョブはスケジュール済みの Apex 制限としてカウントされなくなります。</p> <p><a href="#">メソッドを使用して、スケジュールされた後にジョブを停止します。</a></p> <p>「<a href="#">メソッドの使用</a>」の例を参照してください。</p>
	ID <i>userID</i> String <i>password</i>	Void	<p>指定されたユーザのパスワードを設定します。このパスワードでユーザがログインすると、新しいパスワードを作成するよう求められません。ユーザがリセットプロセスを行い、独自のパスワードを作成するようにする場合、 を使用します。</p> <p> 警告: このメソッドを使用する場合は注意が必要です。また、この機能をエンドユーザーに公開しないでください。</p>
	List<WorkItemIDs> <i>WorkItemIDs</i> String <i>Comments</i> String <i>NextApprover</i>	List<ID>	<p>処理された承認を送信します。詳細は、「<a href="#">Apex 承認プロセスクラス</a>」(ページ 650)を参照してください。</p>
		Date	現在の日付を現在のユーザのタイムゾーンで返します。

### システムの Logging Level

enum を使用して、すべての メソッドのログレベルを指定します。

有効なログレベルは次のとおりです (低いものから順に並べてあります)。

-

- 
- 
- 
- 
- 

ログレベルは累積です。たとえば、Apex コードに最も低いレベル `DEBUG` が指定されている場合、ログレベルが `DEBUG` である debug メソッドのみが記録されます。次に低いレベル `INFO` が指定されている場合、デバッガログには `ERROR` または `WARNING` として指定されている debug メソッドのみが含まれます。

次の例では、ログレベルが `DEBUG` で `System.debug` メソッドのレベルが `INFO` であるため、文字列 `INFO` はデバッガログには書き込まれません。



ログレベルについての詳細は、Salesforce オンラインヘルプの「デバッガログの検索条件の設定」を参照してください。

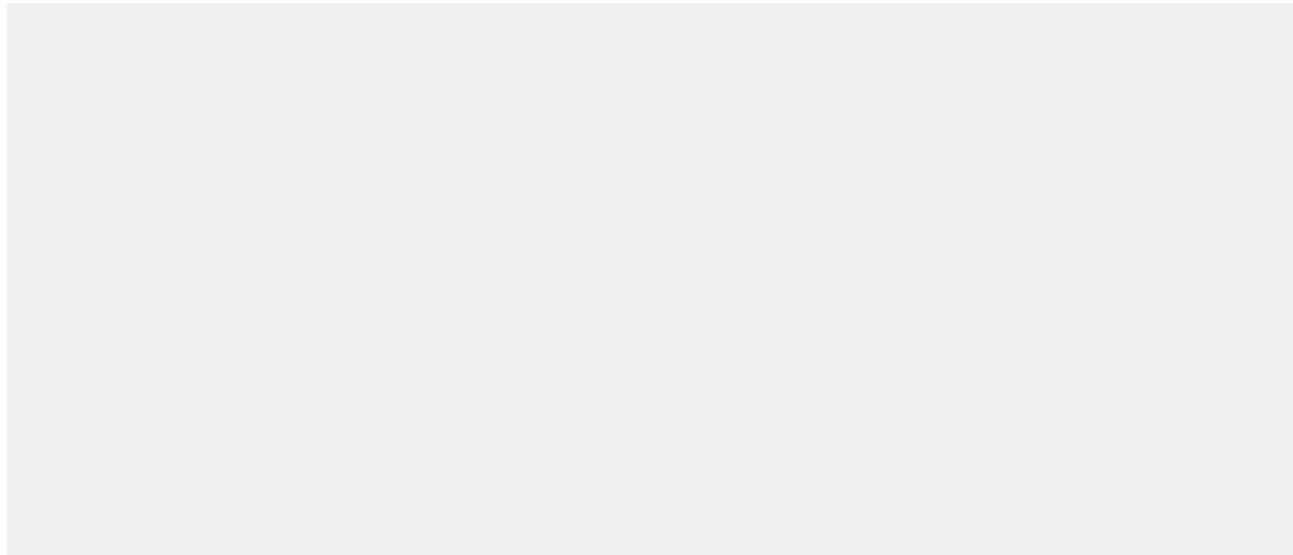
#### **System.ApplicationReadWriteMode enum の使用**

Salesforce アップグレードおよびダウンタイム中にアプリケーションが参照のみモードであるかどうかをプログラムで判断するには、`System.ApplicationReadWriteMode` enum を使用します。

enum の有効な値は、次のとおりです。

- 
- 

例:





### System.Schedule メソッドの使用

インターフェースでクラスを実装したら、  
メソッドを使用してそれを実行します。スケジューラは、システムとして実行されます。ユーザがそのクラスの実行権限を持っているかどうかにかかわらず、すべてのクラスが実行されます。



メモ: クラスをトリガからスケジュールする場合は、細心の注意を払ってください。トリガで許可されている 100 件を超えるスケジュールクラスを追加しないようにする必要があります。特に、API の一括更新、インポートウィザード、ユーザインターフェースを使用したレコードの一括変更、および複数のレコードを一度に更新するすべての処理については十分に考慮してください。この 100 件の同時スケジュールクラスの制限は、メソッドを使用して開始されるスケジュール済み一括処理ジョブには適用されません。

メソッドは、ジョブの名前、ジョブの実行予定日時を表すために使用する式、クラスの名前という 3 つの引数を取ります。この式の構文は次のとおりです。

`Seconds Minutes Hours Day_of_month Month Day_of_week optional_year`



メモ: Salesforce は、指定された時間に実行されるようにクラスをスケジュール設定します。実際の実行は、サービスの使用可能状態に応じて遅れる場合があります。

メソッドでは、すべてのスケジュールの基準としてユーザのタイムゾーンが使用されます。

式の値は次のとおりです。

名前	値	特殊文字
	0 ~ 59	なし
	0 ~ 59	なし
	0 ~ 23	
	1 ~ 31	
	1 ~ 12、または次のとおりです。 • • • • • • • • •	

名前	値	特殊文字
	• • 1 ~ 7、または次のとおりです。	
	• • • • • • •	
optional_year	Null または 1970 ~ 2099	

特殊文字の定義は次のとおりです。

特殊文字	説明
	値を区切れます。たとえば、複数の月を指定する場合は を使用します。
	範囲を指定します。たとえば、複数の月を指定する場合は を使用します。
	すべての値を指定します。たとえば、Monthを と指定すると、ジョブ は毎月にスケジュールされます。
	特定の値を指定しません。Day_of_month と Day_of_week のみで使用 でき、通常は、特定の値以外を指定しない場合に使用します。
	増分を指定します。スラッシュの前の数値は期間の開始を指定し、ス ラッシュの後の数値は期間の長さを指定します。たとえば、 Day_of_month に と指定した場合、Apex クラスは月の 1 日から始 まり、5 日おきに実行されます。
	範囲の終了を指定します。Day_of_month と Day_of_week でのみ使用 できます。日で使用すると、1月の場合は 1 月 31 日、うるう年の 2 月 の場合は 2 月 28 日など、 は常に月末日を意味します。Day_of_week のみで使用すると、 または を意味します。Day_of_week の値と 一緒に使用すると、その月で指定した曜日の最後を意味します。たとえ ば、 と指定すると、月の最終月曜日が指定されます。 と一緒に値 の範囲は使用しないでください。予期しない結果が生じる場合がありま す。
	特定の日に最も近い平日(月曜日～金曜日)を指定します。Day_of_month でのみ使用できます。たとえば、 と指定し、20 日が土曜日の場合、 クラスは 19 日に実行されます。 と指定すると、1 日が土曜日の場

特殊文字	説明
	<p>合、クラスはその前の月ではなく、次の月曜日である3日に実行されます。</p> <p> ヒント: 月の最後の平日を指定するには、とと一緒に使用します。</p> <p><code>weekday day_of_month</code> という形式で、月の第 <i>nth</i> 日目を指定します。  <code>Day_of_week</code> でのみ使用できます。 の前の数値は、平日 ( ) を指定します。 の後の数値は、月の日付を指定します。たとえば、と指定すると、クラスは毎月第2月曜日に実行されます。</p>

次に、式の使用方法の例を示します。

式	説明
	クラスは毎日午後1時に実行されます。
	クラスは毎月最終金曜日の午後10時に実行されます。
	クラスは月曜日から金曜日の午前10時に実行されます。
	クラスは2010年の毎日午後8時に実行されます。

次の例では、クラスによってインターフェースが実装されます。このクラスは、2月13日の午前8時に実行するようにスケジュールされています。

#### System.ResetPasswordResult オブジェクト

System.ResetPasswordResult オブジェクトはパワードのアクセスに使用できます。

メソッドによって返されます。生成された

次の表に、System.ResetPasswordResult オブジェクトのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
		String	System.ResetPasswordResult オブジェクトをインスタンス化した

メソッドの結果として生成

メソッド	引数	戻り値	説明
			されたパスワードを返します。

## 関連リンク

[Apex の一括処理](#)

[Future アノテーション](#)

[Apex スケジューラ](#)

## Test メソッド

次に、Test のシステム静的メソッドを示します。

名前	引数	戻り値	説明
		boolean	現在実行中のコードが、テストメソッドに含まれているコードによってコールされた場合、 <code>true</code> を返します。その他の場合は、 <code>false</code> を返します。テストからコールされたかどうかに応じて異なるコードを実行する必要がある場合に、このメソッドを使用します。
	<code>Schema.SObjectType sObjectType</code> <code>String resourceName</code>	List<sObject>	<p><code>sObjectType</code> 引数は <code>sObject</code> 型であり、この型でテストレコードを挿入します。</p> <p><code>resourceName</code> 引数は、読み込むテストレコードを含む .CSV ファイルに対応する静的リソースの名前です。この名前は大文字と小文字を区別しません。</p> <p>このメソッドをコールする前に静的リソースを作成する必要があります。静的リソースは、拡張子が .CSV のカンマ区切りファイルです。このファイルにはテストレコードの項目名と値が含まれます。ファイルの最初の行に項目名を含め、2 行目以降に値を含める必要があります。</p>

名前	引数	戻り値	説明
			<p>静的リソースについての詳細は、Salesforce オンラインヘルプの「静的リソースの定義」を参照してください。</p> <p>.CSV ファイルの静的リソースを作成したら、その静的リソースに MIME タイプが割り当てられます。次の MIME タイプがサポートされています。</p> <ul style="list-style-type: none"> <li>• text/csv</li> <li>• application/vnd.ms-excel</li> <li>• application/octet-stream</li> <li>• text/plain</li> </ul> <p><a href="#">「SOSL クエリの例」</a> の例を参照してください。</p>
	PageReference <i>page</i>	Void	コントローラの現在の PageReference を設定する Visualforce のテストメソッド。
	PageReference <i>page</i>	Void	コントローラの現在の PageReference を設定する Visualforce のテストメソッド。
	ID[] <i>opt_set_search_results</i>	Void	<p>固定された検索結果のリストを、テストメソッドで後続のすべての SOSL ステートメントに返されるよう定義します。<i>opt_set_search_results</i> が指定されていない場合、後続のすべての SOSL クエリは結果を返しません。</p> <p><i>opt_set_search_results</i> で指定されたレコード ID のリストは、一句または一句に指定されていない場合、通常は SOSL クエリで返される結果を置き換えます。これらの句が SOSL クエリにある場合、固定された検索結果のリストに適用されます。</p> <p>詳細は、「<a href="#">SOSL クエリの単体テストへの追加</a>」(ページ 237)を参照してください。</p>
	Type <i>interfaceType</i> Object <i>instance</i>	Void	擬似応答モードを設定し、HTTP クラスまたは WSDL から自動生成されたコードを使用してコールアウトが実行されるたびに擬似応答を送信するように、Apex ランタイムに指示します。

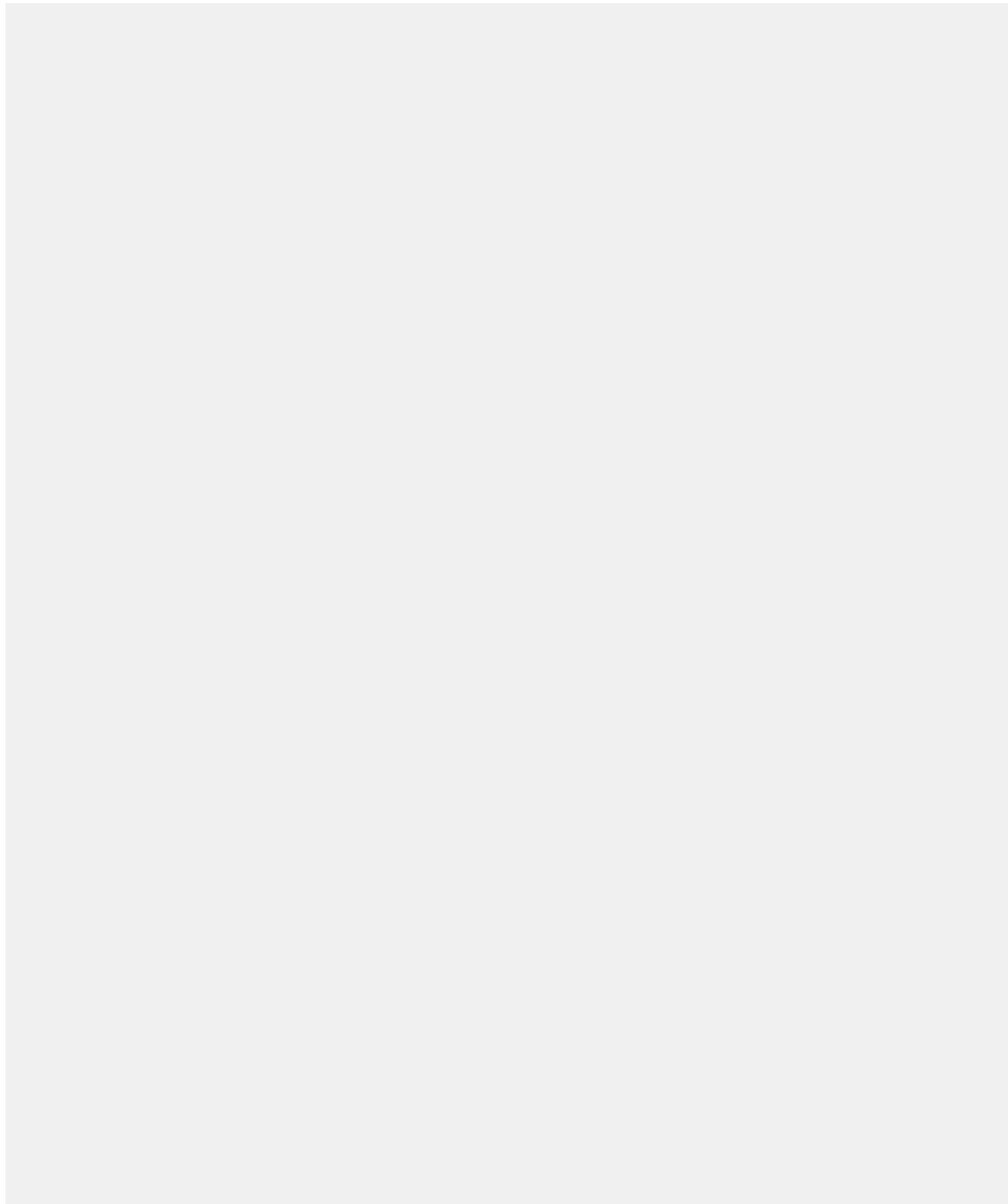
名前	引数	戻り値	説明
			 メモ: コールアウトを実行するコードが管理パッケージに含まれる場合、同じパッケージ内のテストメソッドから同じ名前空間を使用して <code>callout</code> をコールし、擬似コールアウトを行う必要があります。  この例については、「HTTP コールアウトのテスト」および「Web サービスコールアウトのテスト」を参照してください。
	<code>Boolean application_mode Void</code>		Salesforce アップグレードおよびダウンタイム中に参照のみモードをシミュレートするには、Apex テストにおける組織のアプリケーションモードを参照のみに設定します。アプリケーションモードは、Apex テストの各実行が終了するとデフォルトのモードにリセットされます。  は、5 分アップグレードの一部として提供されます。 <a href="#">System メソッド</a> も参照してください。
	<code>Void</code>		テストが実際に開始されるときに、テストコードのポイントをマークします。ガバナ制限をテストする場合にこのメソッドを使用します。と共にこのメソッドを使用して、メソッドの後のすべての非同期コールが、アサーションまたはテストを実行する前に実行されるようにすることができます。各テストメソッドは、このメソッドを1回のみコールできます。このメソッドの前のすべてのコードを、変数の初期化、データ構造の入力などのために使用する必要があります。これにより、テストを実行するために必要なすべてを設定できます。へのコールの後およびの前に実行するコードはすべ

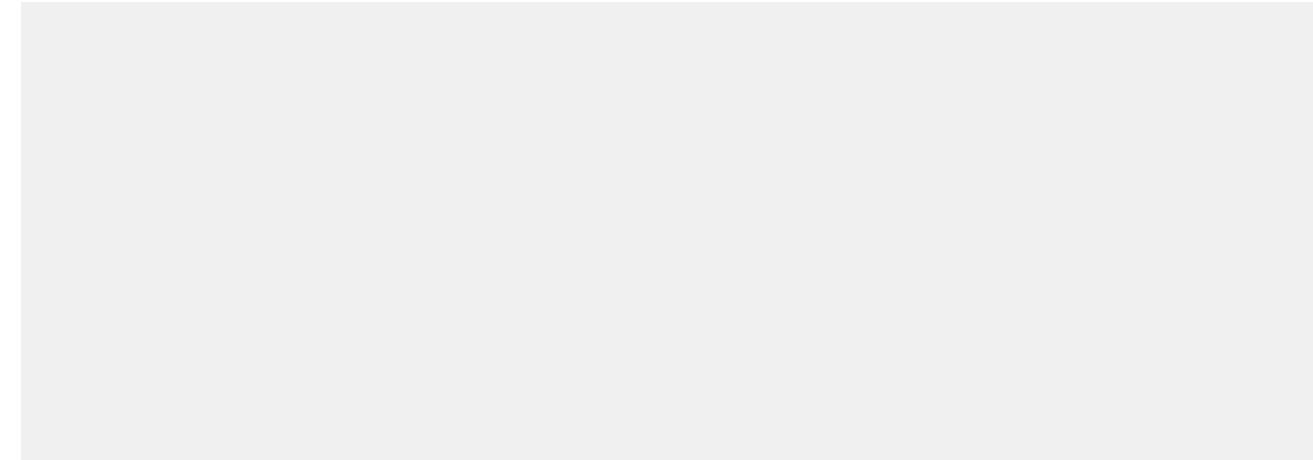
名前	引数	戻り値	説明
			<p>て、新しいガバナ制限セットが割り当てられます。</p> <p>Void テストが終了するときに、テストコードのポイントをマークします。このメソッドは メソッドと組み合わせて使用します。各テストメソッドは、このメソッドを1回のみコールできます。</p> <p>メソッドの後に実行するコードはすべて、 がコールされる前に有効だった元の制限が割り当てられます。 メソッドの後に作成されたすべての非同期コールはシステムによって収集されます。 を実行する場合、すべての非同期プロセスが同期して実行されます。</p> <p> メモ: ブロックおよびブロックでコールされたまたはなどの非同期コールは、キュー内ジョブ数の制限に 対してカウントされません。</p>
	<i>installImp</i> <i>ver</i> Boolean <i>isPush</i>	Void	<p>パッケージでのインストール後スクリプトの指定に使用される、インターフェースの実装をテストします。テストは、開発環境のテストインターフェータとして実行されます。</p> <p><i>installImp</i> 引数は、インターフェースを実装するクラスです。</p> <p><i>ver</i> 引数では、登録者組織にインストールされた既存パッケージのバージョン番号を指定します。</p> <p><i>isPush</i> 引数は省略可能で、アップグレードが転送かどうかを指定します。デフォルト値は、 です。</p>

名前	引数	戻り値	説明
	<i>uninstImp</i> Void		<p>このメソッドでは、テストのインストールが失敗すると実行時例外が発生します。</p> <p>パッケージでのアンインストールスクリプトの指定に使用される、インターフェースの実装をテストします。テストは、開発環境のテストイニシエータとして実行されます。</p> <p><i>uninstImp</i> 引数は、インターフェースを実装するクラスです。</p> <p>このメソッドでは、テストのアンインストールが失敗すると実行時例外が発生します。</p>

**setReadOnlyApplicationMode の例**

次の例では、アプリケーションモードを参照のみに設定し、新しい取引先レコードを挿入しようとしています。結果は例外となります。その後、アプリケーションモードはリセットされ、正しく挿入されます。



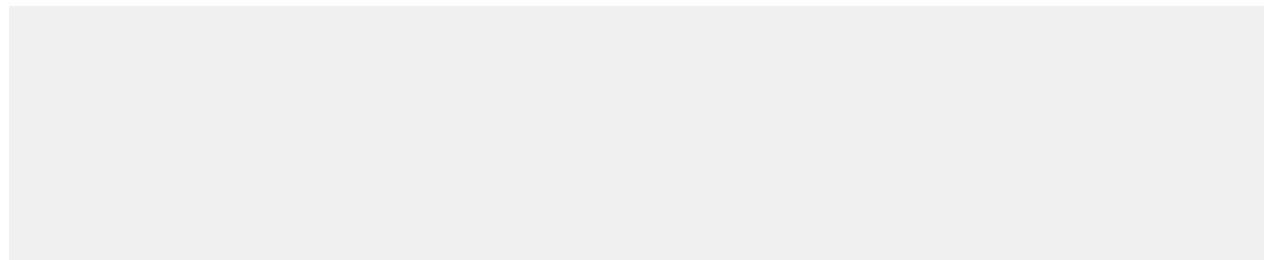


#### Test.loadData の例

サンプル .CSV ファイルと静的リソースを作成し、  
は、次のとおりです。

をコールしてテストレコードを挿入する手順

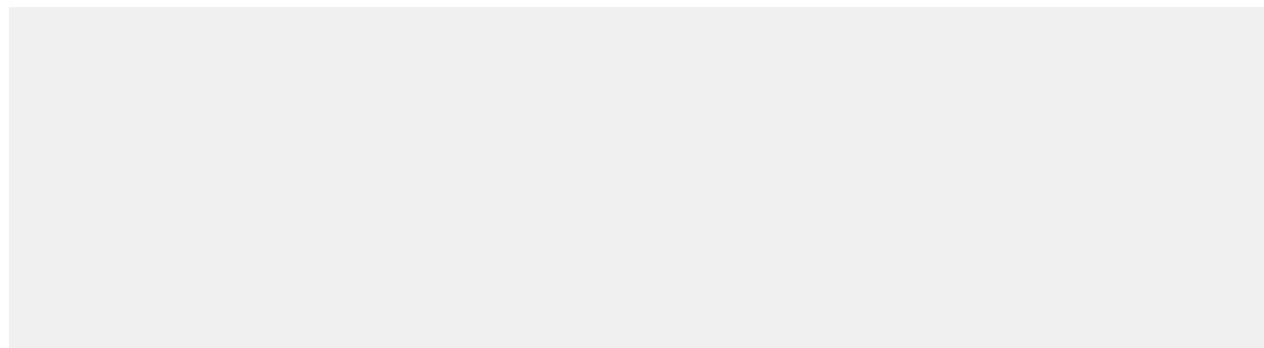
1. テストレコードのデータを含む .CSV ファイルを作成します。このサンプル .CSV ファイルには 3 つの取引先レコードが含まれています。このサンプルの内容を使用して .CSV ファイルを作成できます。



2. .CSV ファイル用の静的リソースを作成します。

- a. [開発] > [静的リソース] をクリックして、[新規静的リソース] をクリックします。
- b. 静的リソースに という名前を付けます。
- c. 作成したファイルを選択します。
- d. [保存] をクリックします。

3. テストメソッドで をコールしてテスト取引先を入力します。



## TimeZone メソッド

タイムゾーンを表します。新しいタイムゾーンを作成し、タイムゾーン ID、オフセット、表示名などのタイムゾーンプロパティを取得するためのメソッドを含みます。

### 使用方法

このクラスのメソッドを使用して、  
から返されるタイムゾーンまたはこのクラスの  
から返されるタイムゾーンのプロパティなど、タイムゾーンのプロパティを取得できます。

### メソッド

クラスの静的メソッドを次に示します。

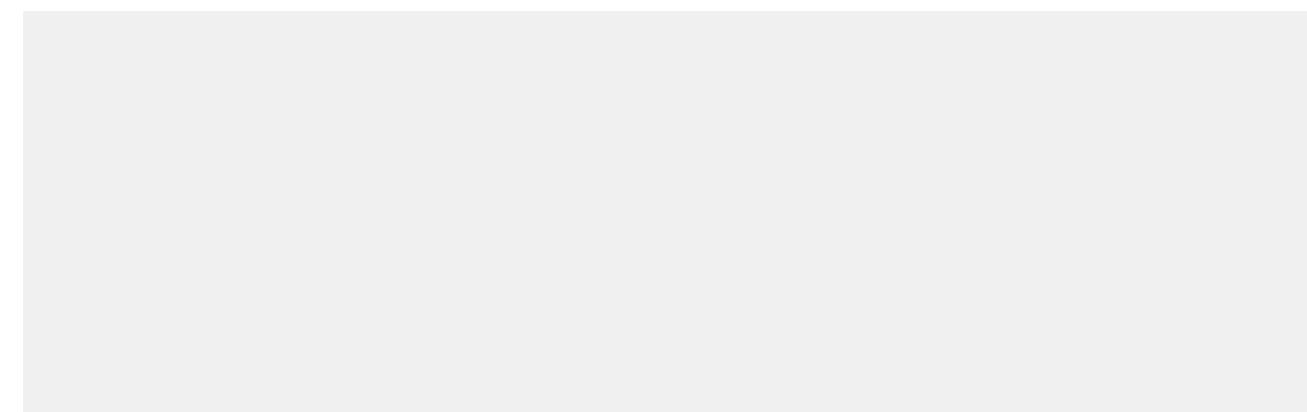
メソッド	引数	戻り値	説明
	string <i>ID</i>	TimeZone	<p>指定されたタイムゾーン ID に対応するタイムゾーンを返します。</p> <p><i>ID</i> 引数に使用できるタイムゾーン値は、<a href="#">Java TimeZone クラス</a>でサポートされる有効なタイムゾーン値です。</p> <p>例:</p>

次に、クラスのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
		string	このタイムゾーンの表示名を返します。
		string	このタイムゾーンの ID を返します。
	dateTime <i>date</i>	integer	指定された日付の GMT タイムゾーンに対するタイムゾーンオフセットをミリ秒単位で返します。 <i>date</i> 引数は、評価する日時です。
			 メモ: <i>date</i> 引数がこのタイムゾーンの夏時間の場合、返されたオフセットは夏時間に調整されます。
		string	このタイムゾーンを文字列表現で返します。

### サンプル

この例では、現在のユーザのタイムゾーンのプロパティを取得し、それをデバッグログに表示する方法を示します。



このサンプルの出力は、ユーザのタイムゾーンによって異なります。ユーザのタイムゾーンが America/Los\_Angeles の場合の出力例を次に示します。このタイムゾーンの場合、夏時間は GMT から -7 時間 (-25200000 ミリ秒) であり、標準時間は GMT から -8 時間 (-28800000 ミリ秒) です。

## 型メソッド

Apex クラスに対応する Apex のデータ型を取得し、新しい型をインスタンス化するためのメソッドを含みます。

### 使用方法

メソッドを使用して、Apex クラス(組み込みクラスまたはユーザ定義クラス)のデータ型を取得します。また、メソッドは、インターフェースを実装する型をインスタンス化し、そのメソッドをコールすると同時に、パッケージの登録者など他のユーザがメソッドの実装を提供できるようにする場合に使用します。

### メソッド

次に、クラスの静的メソッドを示します。

メソッド	引数	戻り値	説明
	String <i>fullyQualifiedName</i>		<p>指定された完全修飾クラス名に対応するデータ型を返します。</p> <p><i>fullyQualifiedName</i>引数は、データ型を取得するクラスの完全修飾名です。完全修飾クラス名には、などの名前空間名が含まれます。</p> <p> メモ: 名前空間が定義されていない組織のローカル型の名前を取得するためにインストールされた管理パッケージから</p> <p>メソッドをコールすると、が返されます。</p> <p>メソッドを代わりに使用し、<i>namespace</i>引数に空の文字列またはを指定します。</p>
	String <i>namespace</i> String <i>name</i>		<p>指定された名前空間およびクラス名に対応するデータ型を返します。</p> <p><i>namespace</i>引数はクラスの名前空間です。</p> <p><i>name</i>引数はクラスの名前です。</p> <p>クラスに名前空間がない場合、<i>namespace</i>引数をまたは空の文字列に設定します。</p> <p> メモ: 名前空間が定義されていない組織にインストールされた管理パッケージからコールする場合には、</p>

メソッド	引数	戻り値	説明
			<p>の代わりにこのメソッドを使用します。ローカル型の名前を取得するには、namespace 引数を空の文字列または <code>System.Type</code> に設定します。たとえば、</p> <p>です。</p> <p>この例では、<code>System.Type</code> クラスおよび名前空間に対応するデータ型を取得する方法を示します。</p>

次に、クラスのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
	Object <code>toCompare</code>	Boolean	<p>指定されたデータ型が現在のデータ型と同じ場合は <code>true</code> を返し、そうでない場合は <code>false</code> を返します。</p> <p><code>toCompare</code> 引数は、現在のデータ型と比較するデータ型です。</p> <p>例:</p>
		String	<p>現在の型の名前を返します。</p> <p>この例では、<code>Type</code> の名前を取得する方法を示します。最初に <code>Type</code> を取得し、次にその <code>Type</code> オブジェクトに対して <code>getName</code> をコールします。</p>

メソッド	引数	戻り値	説明
		Integer	<p>現在のデータ型のハッシュコード値を返します。</p> <p>返されたハッシュコード値は、が返す型名のハッシュコードに対応します。</p>
		Object	<p>現在の型のインスタンスを作成し、この新しいインスタンスを返します。</p> <p>は汎用オブジェクト型を返すため、この値を保持する変数の型に戻り値を変換する必要があります。</p> <p>このメソッドを使用すると、インターフェースを実装する Type をインスタンス化し、そのメソッドをコールできると同時に、他のユーザがメソッドの実装を提供できるようになります。たとえば、パッケージ開発者がインターフェースを提供し、登録者がそのインターフェースを実装してパッケージをインストールできます。パッケージのコードは、登録者の Type をインスタンス化することで、登録者のインターフェースメソッドの実装をコールします。</p> <p>次の例では、Type のインスタンスを作成する方法を示します。最初に をクラスの名前( )でコールして Type を取得し、次にこの Type オブジェクトに対して をコールします。</p> <p>インスタンスは、 クラスが実装するインターフェース型( )を使用して宣言されます。 メソッドの戻り値は、 型に変換されます。</p>

メソッド	引数	戻り値	説明
		String	<p>現在のデータ型 (データ型名) を文字列表現で返します。</p> <p>このメソッドは、 と同じ値を返します。 および はこのメソッドを使用して、 Type 引数を string に変換します。</p> <p>この例では、Integer のリストに対応する Type に対して をコールします。</p>

サンプル: 名前に基づいた Type のインスタンス化

次のサンプルは、Type メソッドを使用して、Type をその名前に基づいてインスタンス化する方法を示します。このシナリオの典型的な応用として、パッケージの登録者が、インストールされたパッケージの一部としてインターフェースのカスタム実装を提供する場合があります。パッケージは、登録者の組織のカスタム設定を介してインターフェースを実装するクラスの名前を取得できます。パッケージは、このクラス名に対応する型をインスタンス化して、登録者が実装したメソッドを呼び出すことができます。

このサンプルでは、  
クラスには、  
が  
クラスによって実装されるインターフェースを表します。最後の  
に実装されたメソッドを呼び出すコードサンプルが含まれます。

これがインターフェースです。

これがインターフェースの実装です。

このクラスのメソッドは、**インターフェースを実装するクラスの名前をカスタム設定値を介して取得します。**その後、このクラスをインスタンス化するために、対応する型を取得し、**メソッドをコードします。**次に、**に実装されたメソッドを呼び出します。**このサンプルでは、**という名前のテキスト項目を持つ**という名前の公開リストカスタム設定を作成する必要があります。このカスタム設定のレコードを、**というデータセット名とクラス名値**で1つ作成します。

## クラスのプロパティ

プロパティは、コールされたデータ型の**を返します。**これは、プリミティブデータ型とコレクション、sObject型、ユーザ定義クラスを含むすべてのApex組み込みデータ型で公開されます。**メソッドの代わりにこのプロパティを使用できます。**

データ型名に対してこのプロパティをコールします。たとえば、次のようにになります。

メソッドの2番目の引数にこのプロパティを使用して、並列化するオブジェクトのデータ型を取得できます。たとえば、次のようにになります。

## URL メソッド

URL (Uniform Resource Locator) を表し、URL の一部へのアクセスを提供します。Salesforce インスタンス URL へのアクセスを有効にします。

使用方法

組織内のオブジェクトへのリンクを作成するには、クラスのメソッドを使用します。これらのオブジェクトは、外部メール、活動、または Chatter 投稿に組み込むファイル、画像、ロゴ、レコードがあります。たとえば、次の例のように、Salesforce の基本 URL にファイル ID を連結することによって、Chatter 投稿への添付ファイルとしてアップロードされたファイルへのリンクを作成できます。

次の例では、Salesforce レコードへのリンクを作成します。Salesforce の基本 URL とレコード ID が連結されて完全な URL が作成されます。

## コンストラクタ

引数	説明
<code>String protocol</code>	指定したプロトコル、ホスト、ポート、およびそのホストのファ
<code>String host</code>	イルを使用して、
<code>Integer port</code>	クラスの新しいインスタンスを作成します。

引数	説明
String <i>file</i>	
String <i>protocol</i>	指定したプロトコル、ホスト、およびそのホストのファイルを使用して、クラスの新しいインスタンスを作成します。
String <i>host</i>	
String <i>file</i>	指定したプロトコルのデフォルトのポートが使用されます。
URL <i>context</i>	指定したコンテキスト内で指定した spec を解析して、
String <i>spec</i>	クラスの新しいインスタンスを作成します。 このコンストラクタの引数についての詳細は、Java のそれぞれの URL(java.net.URL, java.lang.String) コンストラクタを参照してください。
String <i>spec</i>	URL の指定した文字列表現を使用して、 クラスの新しいインスタンスを作成します。

## メソッド

次に、クラスの静的メソッドを示します。

メソッド	引数	戻り値	説明
			Salesforce インスタンスでの要求全体の URL を返します。 たとえば、 です。
	String <i>entityId</i> String <i>fieldName</i>	String	添付ファイルのダウンロード URL を返します。 <i>entityId</i> 引数は、ファイルデータを保持するエンティティの ID を指定します。 <i>fieldName</i> 引数は、などのファイル項目コンポーネントの API 名を指定します。 例:

メソッド	引数	戻り値	説明
			Salesforce インスタンスの URL を返します。 たとえば、 です。

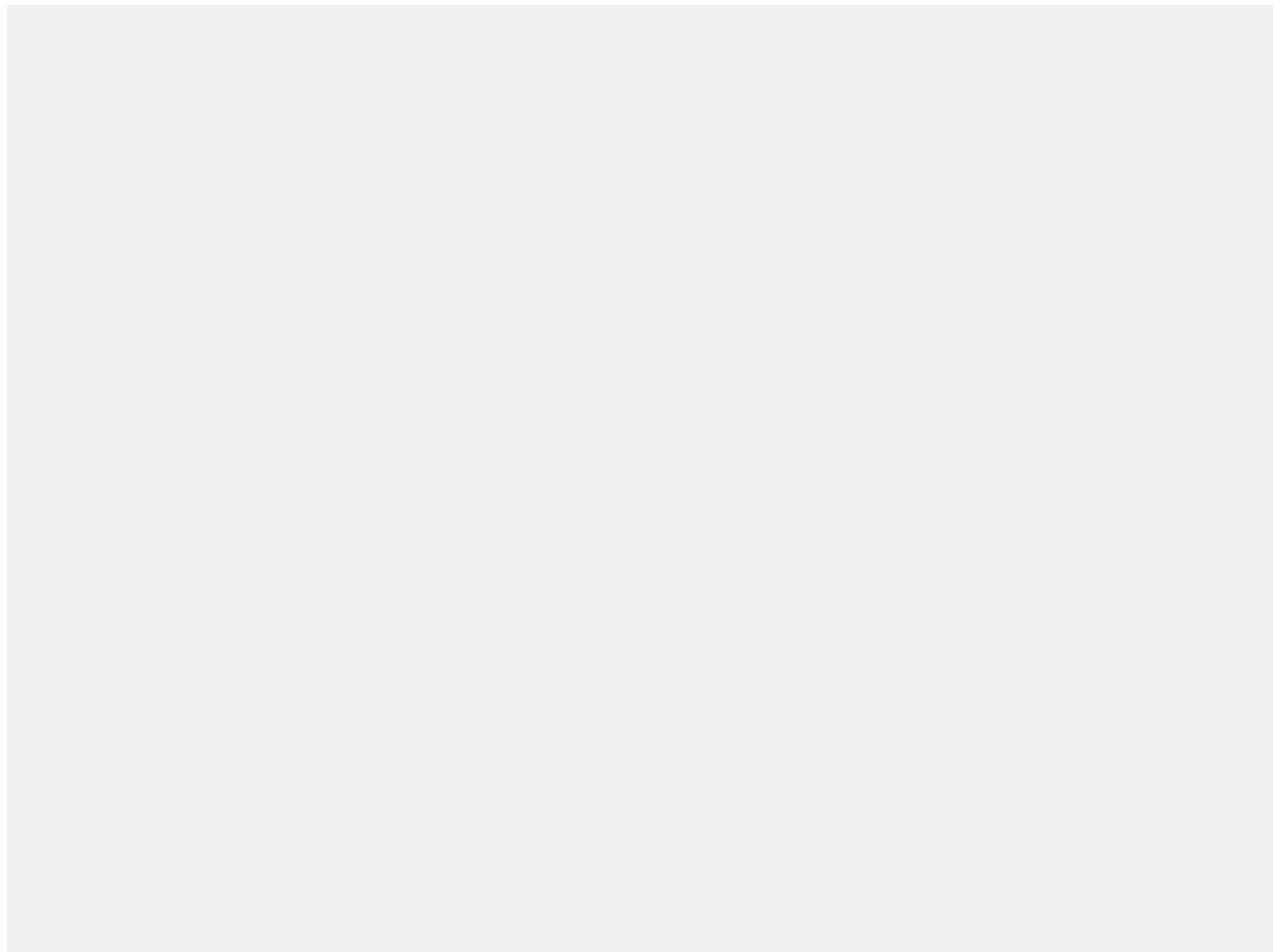
次に、 クラスのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
		String	現在の URL の権限部分を返します。
		Integer	現在の URL に関連付けられたプロトコルのデフォルトのポート番号を返します。
			URL の URL スキームまたはストリームプロトコルハンドラにデフォルトのポート番号が定義されていない場合、-1 を返します。
		String	現在の URL のファイル名を返します。
		String	現在の URL のホスト名を返します。
		String	現在の URL のパス部分を返します。
		Integer	現在の URL のポートを返します。
		String	現在の URL のプロトコル名を返します。たとえば、 です。
		String	現在の URL のクエリ部分を返します。 クエリ部分が存在しない場合は、 を返します。
		String	現在の URL のアンカーを返します。 クエリ部分が存在しない場合は、 を返します。
		String	現在の URL の UserInfo 部分を取得します。 UserInfo 部分が存在しない場合は を返します。

メソッド	引数	戻り値	説明
	<code>URLToCompare</code>	Boolean	フラグメントコンポーネントを除き、現在の URL と指定した URL オブジェクトを比較します。 両方の URL オブジェクトが同じリモートソースを参照する場合は <code>true</code> 、そうでない場合は <code>false</code> を返します。
		String	URI とフラグメントコンポーネントの構文についての詳細は、「 <a href="#">RFC3986</a> 」を参照してください。

### URL のサンプル

この例では、現在の Salesforce サーバインスタンスの基本 URL と完全要求 URL が取得されます。次に、特定の取引先オブジェクトを指定する URL が作成されます。最後に、基本 URL と完全 URL のコンポーネントが取得されます。この例では、すべての結果がデバッグログに出力されます。



## Userinfo メソッド

次に、UserInfo のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	string	マルチ通貨組織のコンテキストユーザのデフォルト通貨コードまたは単一通貨の組織の組織の通貨コードを返します。	 メモ: Salesforce.com API バージョン 22.0 以前を使用して保存された Apex の場合、 <code>getCurrencyIsoCode()</code> は、単一通貨の組織に <code>getCurrencyCode()</code> を返します。
	string	コンテキストユーザの名前を返します。	
	string	コンテキストユーザの言語を返します。	
	string	コンテキストユーザの姓を返します。	
	string	コンテキストユーザのロケールを返します。 次に例を示します。	
	string	コンテキストユーザの氏名を返します。名前の形式は、組織に指定された言語設定に応じて異なります。形式は次のいずれかになります。	<ul style="list-style-type: none"><li>• FirstName LastName</li><li>• LastName, FirstName</li></ul>
	string	コンテキスト組織の ID を返します。	
	string	コンテキスト組織の会社名を返します。	
	string	コンテキストユーザのプロファイル ID を返します。	

名前	引数	戻り値	説明
		string	現在のセッションのセッション ID を返します。 メソッド、Apex の一括処理ジョブ、または Apex スケジュール済みジョブなど、非同期で実行される Apex コードでは、 は を返します。 ベストプラクティスとして、自分のコードが、セッション ID が使用可能な場合と使用できない場合の両方のケースに対応できるようにしてください。
		System.TimeZone	現在のユーザのローカルタイムゾーンを返します。 例:
		string	デフォルトの組織テーマを返します。 を使用して、現在のユーザに実際に表示されるテーマを決定します。 有効な値は、次のとおりです。 • • • •
		string	現在のユーザに表示されるテーマを返します。 有効な値は、次のとおりです。 •

名前	引数	戻り値	説明
			• 現在のユーザのメールアドレスを返します。 例:
		string	コンテキストユーザの ID を返します。
		string	コンテキストユーザのログイン名を返します。
		string	コンテキストユーザのロールIDを返します。
		string	コンテキストユーザのデータ型を返します。
	String <i>namespace</i>	boolean	コンテキストユーザに <i>namespace</i> で示された管理パッケージに対するライセンスがある場合は <code>true</code> を返します。ない場合は <code>false</code> を返します。 <i>namespace</i> が無効なパラメータの場合、 <code>false</code> が返されます。
		boolean	組織がマルチ通貨を使用するかどうかを指定します。

## Version メソッド

Version メソッドを使用して、登録者の管理パッケージのバージョンを取得して、パッケージのバージョンを比較します。

### 使用方法

パッケージバージョンは、パッケージでアップロードされる一連のコンポーネントを特定する番号です。バージョン番号の形式は `majorNumber.minorNumber.patchNumber` (例: 2.1.3) です。メジャー番号とマイナー番号は、毎回のメジャーリリース時に指定した値に増えます。`patchNumber` は、パッチリリースにのみ生成および更新されます。

コールされたコンポーネントは、メソッドを使ってコンパイルされたコード元のバージョンを確認し、コード元が想定した動作に応じて動作を変化できます。このため、コードを更新しながらも、クラスとトリガの既存の動作を以前のパッケージバージョンでサポートし続けることができます。

メソッドが返す値は、メジャー番号とマイナー番号の 2 つの番号で構成されたバージョン番号が付加された、このクラスのインスタンスです。メソッドはパッチ番号を返さないため、返される Version オブジェクトのパッチ番号は null です。

クラスは、パッチ番号を含む 3 つの番号で構成されるバージョン番号も保持できます。

## コンストラクタ

引数	説明
Integer <i>major</i>	指定されたメジャー・バージョン番号とマイナー・バージョン番号を
Integer <i>minor</i>	使って、2 つの番号で構成されたパッケージ・バージョンを作成します。
Integer <i>major</i>	指定されたメジャー・バージョン番号とマイナー・バージョン番号、
Integer <i>minor</i>	さらにパッチ・バージョン番号を使って、3 つの番号で構成された
Integer <i>patch</i>	パッケージ・バージョンを作成します。

## メソッド

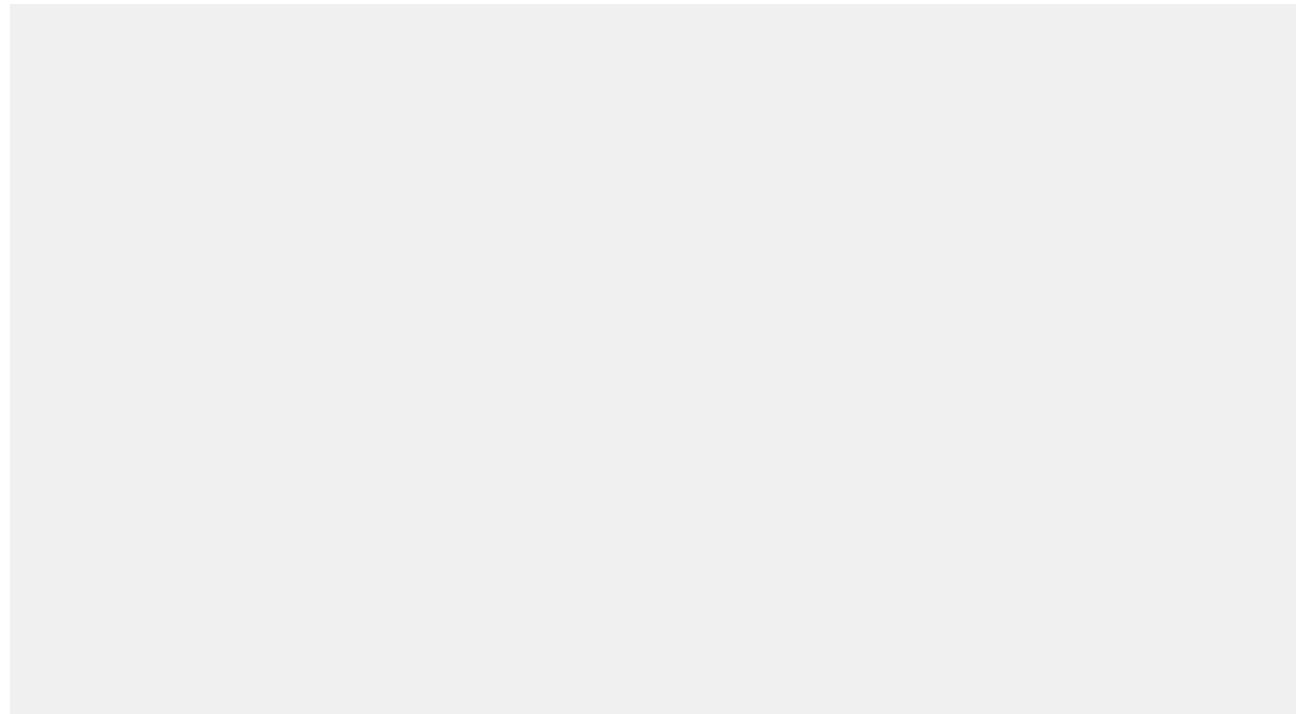
次に、クラスのインスタンスマソッドを示します。

メソッド	引数	戻り値	説明
	System.Version <i>version</i> Integer	現在のバージョンと指定されたバージョンを比較し、次のいずれかの値を返します。 <ul style="list-style-type: none"> <li>ゼロ。現在のパッケージ・バージョンが指定されたパッケージ・バージョンと同じである場合。</li> <li>ゼロより大きい整数値。現在のパッケージ・バージョンが指定されたパッケージ・バージョンより大きい場合。</li> <li>ゼロより小さい整数値。現在のパッケージ・バージョンが指定されたパッケージ・バージョンより小さい場合。</li> </ul> 2 つの番号で構成されたバージョンが 3 つの番号で構成されたバージョンと比較される場合、パッチ番号は無視されます。したがって、比較は、メジャー番号とマイナー番号のみに基づいて行われます。	

メソッド	引数	戻り値	説明
		Integer	コードのメジャーパッケージバージョンを返します。
		Integer	コードのマイナーパッケージバージョンを返します。
		Integer	コードのパッチパッケージバージョンを返します。パッチバージョンがない場合は、 <code>0</code> を返します。

### Version の例

この例では、このクラスのメソッドおよび `System` メソッドを使用して、パッケージをコールするコードの管理パッケージバージョンを判定する方法を示します。



### 関連リンク

[System メソッド](#)

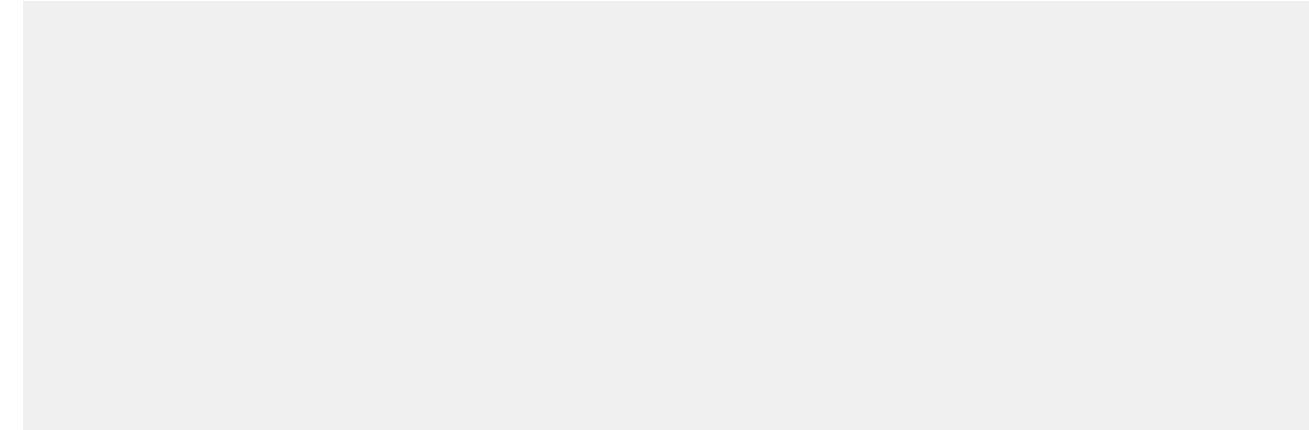
## 例外メソッドの使用

すべての例外は、エラーメッセージや例外型を返す組込みメソッドをサポートしています。標準のクラスに加え、例外にはさまざまな型があります。

例外	説明
	非同期コールのエンキューの失敗など、非同期処理に関する問題を示す例外。
	外部システムへのコールの失敗など、Web サービス処理に関する問題を示す例外。
	ステートメントでレコードの必要な項目が欠落している場合など、DML ステートメントに関する問題を示す例外。
	送信の失敗など、メールに関する問題を示す例外。詳細は、「 <a href="#">Apex メールクラス</a> 」(ページ 662)を参照してください。
	URL に関する問題を示す例外。通常は Visualforce ページで使用します。Visualforce についての詳細は、「 <a href="#">Visualforce 開発者ガイド</a> 」を参照してください。
	JSON の逐次化処理および並列化処理に関する問題を示す例外。詳細は、「 <a href="#">JSON の逐次化処理</a> 」、「 <a href="#">並列化処理</a> 」、および「 <a href="#">メソッドを参照してください。</a>
	範囲外のインデックスへのアクセスなど、リストに関する問題を示す例外。
	0 による除算など、算術演算に関する問題を示す例外。
	現在のユーザがアクセス権を付与されていない sObject へのアクセスなど、承認されないアクセスに関する問題を示す例外。通常は Visualforce ページで使用します。Visualforce についての詳細は、「 <a href="#">Visualforce 開発者ガイド</a> 」を参照してください。
	削除された sObject へのアクセスなど、存在しないデータに関する問題を示す例外。通常は Visualforce ページで使用します。Visualforce についての詳細は、「 <a href="#">Visualforce 開発者ガイド</a> 」を参照してください。
	特に <a href="#">Iterator</a> メソッドで使用されます。リストの終わり以降の項目にアクセスしようとすると、この例外が発生します。たとえば、 で <a href="#">next()</a> をコールすると、この例外が発生します。
	次のコードで示す例のような、null 値の逆参照に関する問題を示す例外。
	sObject の単一変数に対する、レコードを返さない、または複数のレコードを返すクエリの割り当てなど、SOQL クエリに関する問題を示す例外。
	Chatter が有効でない組織にリリースされているコードに Chatter 機能が要求されている。

例外	説明
SOAP API	コールで実行される SOSL クエリの問題を示す例外。たとえば、パラメータに含まれる文字数が 2 文字未満。詳細は、『 <a href="#">SOAP API 開発者ガイド</a> 』を参照してください。
Crypto ユーティリティクラスの静的メソッドに関する問題を示す例外。詳細は、「 <a href="#">クラス</a> 」(ページ 807)を参照してください。	
	データの逐次化に関する問題を示す例外。通常は Visualforce ページで使用します。Visualforce についての詳細は、『 <a href="#">Visualforce 開発者ガイド</a> 』を参照してください。
	の間のみ変更可能な sObject レコードに関する問題を示す例外。
	ヒープサイズを超える string など、string に関する問題を示す例外。
	メソッドを使用した string 型「a」の integer 型への変換など、型の変換に関する問題を示す例外。
	Visualforce ページに関する問題を示す例外。Visualforce についての詳細は、『 <a href="#">Visualforce 開発者ガイド</a> 』を参照してください。
	XML の読み取り、書き込みの失敗など、XmlStream クラスに関する問題を示す例外。詳細は、「 <a href="#">XmlStream クラス</a> 」を参照してください。

DmlException 例外の使用例を次に示します。



## 共通例外メソッド

例外メソッドは、例外の特定のインスタンスからコールされ、処理されます。次の表にすべてのインスタンス例外メソッドを示します。すべての例外型に共通で次のメソッドが含まれます。

名前	引数	戻り値	説明
		Exception	例外オブジェクトとして例外の原因を返します。

名前	引数	戻り値	説明
		Integer	例外が発生した箇所の行番号を返します。
		string	ユーザに表示されるエラーメッセージを返します。
		string	文字列としてスタック追跡を返します。
		string	DmlException、ListException、MathException などの例外型を返します。
	Exception <i>cause</i>	Void	この例外の原因が設定されていない場合は設定します。
	String <i>s</i>	Void	ユーザに表示されるエラーメッセージを設定します。

## DMLEception および EmailException メソッド

共通例外メソッドに加え、DMLEceptions および EmailExceptions には次のメソッドもあります。

名前	引数	戻り値	説明
			System.StatusCode についての詳細は、「enum」を参照してください。
	Integer		DML 例外で失敗した行数を返します。

## Apex クラス

Apex を使用して独自のクラスを作成できますが、アプリケーション構築用のシステム提供のクラスも使用できます。

- [Apex 承認プロセスクラス](#)
- [Apex Community \(Zone\) クラス](#)
- [Apex メールクラス](#)
  - [クラス](#)
- [Cases クラス](#)
- [Chatter in Apex クラス](#)
- [Cookie クラス](#)
- [Exception クラス](#)
  - [クラス](#)
- [HTTP \(RESTful\) サービスクラス](#)
- [ナレッジ管理の公開サービスクラス](#)
- [Network クラス](#)
  - [および `Publisher Action` クラス](#)
- [クラス](#)
- [Visualforce クラス](#)
- [XML クラス](#)

## Apex 承認プロセスクラス

承認プロセスは、Salesforce でレコードを承認する場合に、組織で使用できる自動化されたプロセスです。承認プロセスでは、承認するレコードの条件と各承認ステップの承認者を指定します。各承認ステップは、その承認プロセスの対象レコードすべてに適用することも、システム管理者が定義した特定の条件を満たすレコードのみに適用することもできます。承認プロセスでは、レコードの承認、却下、取り消しまたは最初の承認申請時に実施するアクションも指定します。

Apex を使用し、次のようにプログラムで実行する承認プロセスを作成して、既存の承認プロセスを拡張することができます。

- Apex プロセスクラス: 承認申請を作成し、これらの要求の結果を処理します。詳細は、以下を参照してください。

- ◊ [クラス \(ページ 653\)](#)
- ◊ [クラス \(ページ 653\)](#)
- ◊ [クラス \(ページ 654\)](#)
- ◊ [クラス \(ページ 655\)](#)

- Approval 名前空間の [メソッド](#): 承認申請を送信し、既存の承認申請を承認または却下します。詳細は、「[Approval メソッド](#)」(ページ 564)を参照してください。

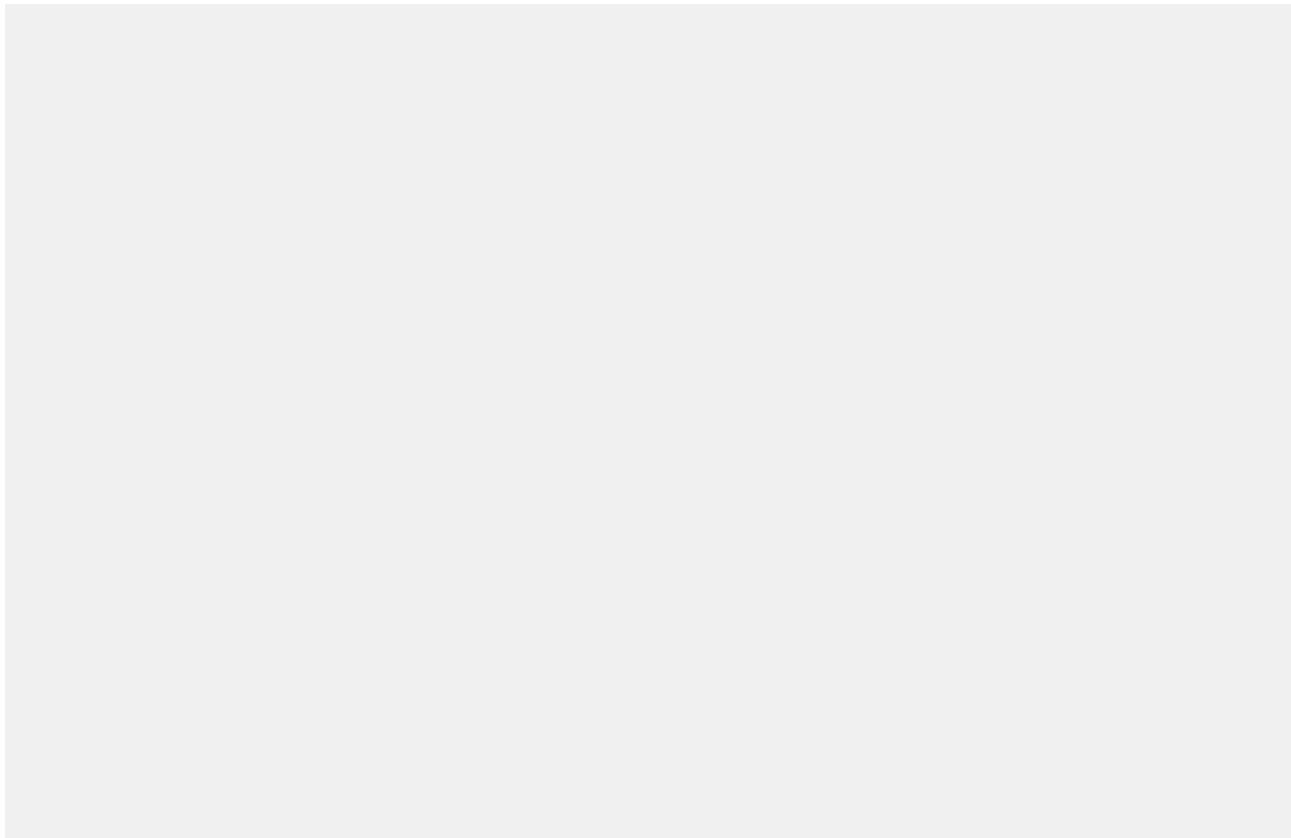


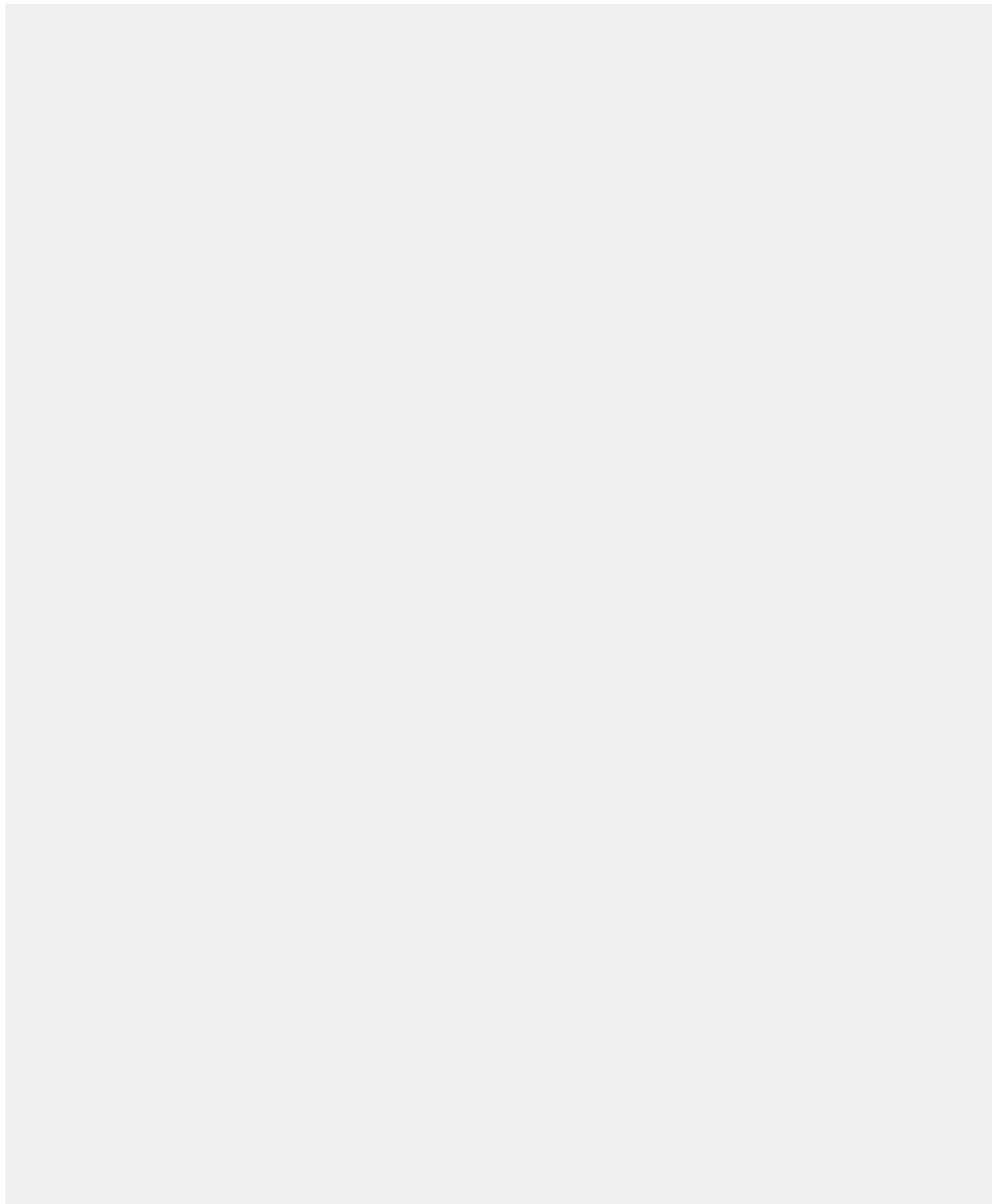
メモ: メソッドは、組織の DML 制限にカウントされます。「[実行ガバナと制限について](#)」(ページ 337)を参照してください。

承認プロセスの詳細については、Salesforce オンラインヘルプの「[承認プロセスの開始](#)」を参照してください。

## Apex 承認プロセスの例

次のサンプルコードでは、最初に承認のレコードを送信し、その後要求を承認します。この例では、承認プロセスを取引先に設定する必要があります。





## ProcessRequest クラス

クラスは クラスおよび クラスの親クラスです。いずれかのクラスからオブジェクトを処理できる汎用の Apex を記述するには、クラスを使用します。

このクラスのインスタンスを作成するとき、Approval 名前空間を指定する必要があります。このクラスのコンストラクタは、引数をとりません。次に例を示します。

クラスには次のメソッドがあります。

名前	引数	戻り値	説明
	string		以前承認申請に追加されたコメントを返します。
	ID[]		承認者として指定されたユーザのユーザ ID のリストを返します。
	String	Void	承認要求に追加されるコメント。
	ID[]	Void	承認プロセスの次のステップが別の Apex 承認プロセスである場合、次の承認者として 1 つのユーザ ID を指定します。そうでない場合、ユーザ ID を指定できず、このメソッドは である必要があります。

## ProcessResult クラス

承認を求めてレコードを送信した後、 クラスを使用して、承認プロセスの結果を処理します。

ProcessResult オブジェクトは メソッドによって返されます。このクラスのインスタンスを作成するとき、Approval 名前空間を指定する必要があります。次に例を示します。

クラスには次のメソッドがあります。このメソッドは引数を取りません。

名前	戻り値	説明
	string	処理されるレコードの ID。
	Database.Error[]	エラーが発生した場合、エラーコードや記述子など、1 つまたは複数のデータベースエラー オブジェクト

名前	戻り値	説明
		の配列を返します。詳細は、「 <a href="#">Database.Error クラス</a> 」(ページ 426)を参照してください。
	string	承認を得るために送信される承認プロセスの ID。
	string	現在の承認プロセスの状況。有効な値は、Approved、Rejected、Removed または Pending です。
	ID[]	承認プロセスに送信された新しい項目の ID です。0 件または 1 件の承認プロセスがあります。
	boolean	boolean 値です。承認プロセスが正常に完了した場合は TRUE、そうでない場合は FALSE に設定されます。

### ProcessSubmitRequest クラス

クラスを使用し、承認を要求してレコードを送信します。

このクラスのインスタンスを作成するとき、Approval 名前空間を指定する必要があります。このクラスのコンストラクタは、引数をとりません。次に例を示します。

次のメソッドは、  
クラス独自のものです。これらのメソッドに加え、  
クラスは、親クラス のすべてのメソッドにアクセスできます。

名前	引数	戻り値	説明
		string	承認を得るために送信されるレコードの ID を返します。たとえば、取引先、取引先責任者、カスタムオブジェクトレコードを返します。
	String Id	Void	承認を得るために送信されるレコードの ID を設定します。たとえば、取引先、取引先責任者、カスタムオブジェクトレコードを指定します。

クラスは、 と次のメソッドを共有します。

名前	引数	戻り値	説明
		string	以前承認申請に追加されたコメントを返します。

名前	引数	戻り値	説明
		ID[]	承認者として指定されたユーザのユーザ ID のリストを返します。
	String	Void	承認要求に追加されるコメント。
	ID[]	Void	承認プロセスの次のステップが別の Apex 承認プロセスである場合、次の承認者として 1 つのユーザ ID を指定します。そうでない場合、ユーザ ID を指定できず、このメソッドは <code>void</code> である必要があります。

## ProcessWorkitemRequest クラス

クラスを使用して、送信後に承認要求を処理します。

このクラスのインスタンスを作成するとき、Approval 名前空間を指定する必要があります。このクラスのコンストラクタは、引数をとりません。次に例を示します。

次のメソッドは、  
クラスは、親クラス  
クラス独自のものです。これらのメソッドに加え、  
のすべてのメソッドにアクセスできます。

名前	引数	戻り値	説明
		string	すでに承認要求と関連するアクションの種類を返します。有効な値は、Approve、Reject、または Removed です。
		string	承認、却下、または削除されるプロセスの承認要求の ID を返します。
	String <i>s</i>	Void	承認要求を処理するために実行するアクションの種類を設定します。有効な値は、Approve、Reject、または Removed です。Removed を指定できるのは、システム管理者だけです。
	String <i>Id</i>	Void	承認、却下、または削除される承認要求の ID を設定します。

クラスは、  
と次のメソッドを共有します。

名前	引数	戻り値	説明
	string		以前承認申請に追加されたコメントを返します。
	ID[]		承認者として指定されたユーザのユーザ ID のリストを返します。
	String	Void	承認要求に追加されるコメント。
	ID[]	Void	承認プロセスの次のステップが別の Apex 承認プロセスである場合、次の承認者として 1 つのユーザ ID を指定します。そうでない場合、ユーザ ID を指定できず、このメソッドは <code>void</code> である必要があります。

## Apex Community (Zone) クラス



メモ: Summer '13 リリースから、「コミュニティ」は「ゾーン」という名前に変更されました。

ゾーンは、アイデアとアンサーを論理グループに整理するときに役立ちます。ゾーンには、それぞれ独自のテーマ、固有のアイデアやアンサーのトピックがあります。Apex には、ゾーンに関連した次のクラスがあります。

- [Answers クラス](#)
- [Ideas クラス](#)

### 関連リンク

[Answers クラス](#)

[Ideas クラス](#)

### Answers クラス

アンサーは、コミュニティアプリケーションの機能の1つです。この機能により、ユーザが質問したり、コミュニティメンバーが返信を投稿したりすることができます。コミュニティメンバーは、それぞれの返信の役立ち度について投票し、また質問したユーザは最良の回答として返信をマークできます。

次に、アンサーの静的メソッドを示します。

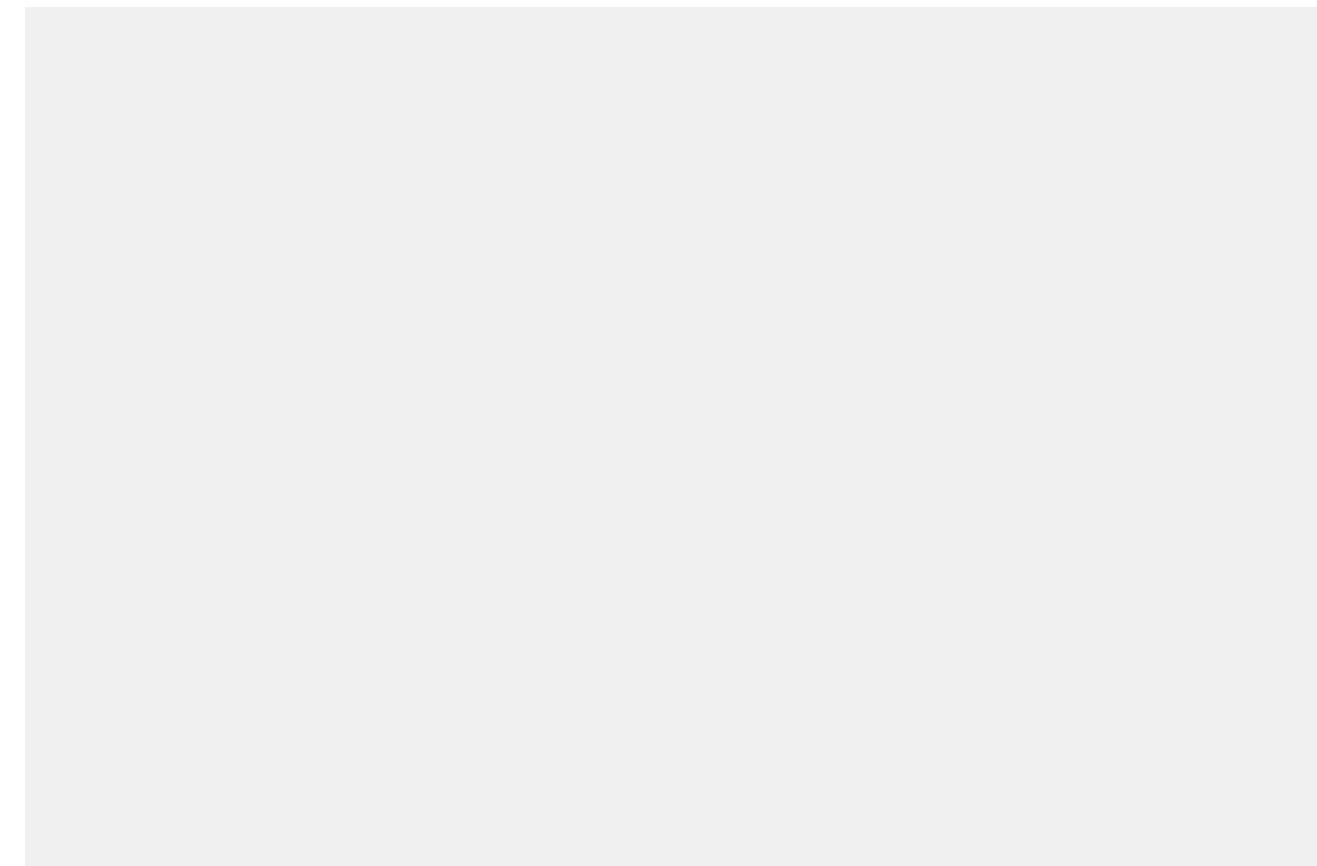
名前	引数	戻り値	説明
	Question <i>question</i>	ID[]	<i>question</i> のタイトルに基づいた類似質問のリストが返されます。各 コールは、プ

名前	引数	戻り値	説明
			口セスで使用できる SOSL ステートメントガバナーの制限にカウントされます。
	String <i>questionId</i> Void		指定した質問の指定した返信を最良の返信に設定します。質問には複数の返信があるため、最良の返信を設定しておくことで最も役立つ情報を含む返信をユーザが迅速に特定できます。
	String <i>replyId</i>		

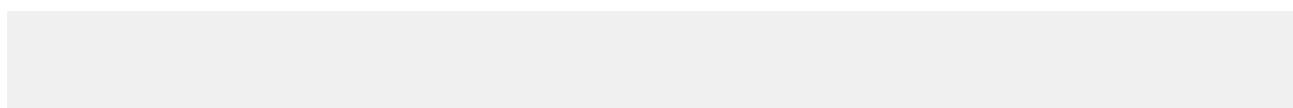
アンサーについての詳細は、Salesforce オンラインヘルプの「アンサーの概要」を参照してください。

#### アンサーの例

内部ゾーンの中で新しい質問に似たタイトルの質問を検索する例を次に示します。



返信を最良の返信に選択する例を次に示します。



## 関連リンク

[Apex Community \(Zone\) クラス](#)

[Ideas クラス](#)

## Ideas クラス

アイデアは、アイデアとアイデアに対する投票およびコメントを投稿するユーザのコミュニティです。アイデアコミュニティは、オンラインのわかりやすい方法で、革新的なアイデアを訴求、管理、および紹介できます。

最近のコメントセット(メソッドにより返されます。下記を参照)には、ユーザが投稿したコメントや、別のユーザが投稿したコメントに対するコメントなどのアイデアが含まれます。返されたアイデアは、別のユーザが行った最後のコメント投稿時間に基づいてリストされ、最新のアイデアが先頭となります。

*userID*引数は必須です。結果を絞り込んで、指定されたユーザが投稿またはコメントしたアイデアのみを返します。

*communityID*引数は、結果を絞り込んで、指定されたゾーン内のアイデアのみを返します。この引数が空の文字列である場合、指定されたユーザの最近のコメントすべてが、ゾーンに関わらず返されます。

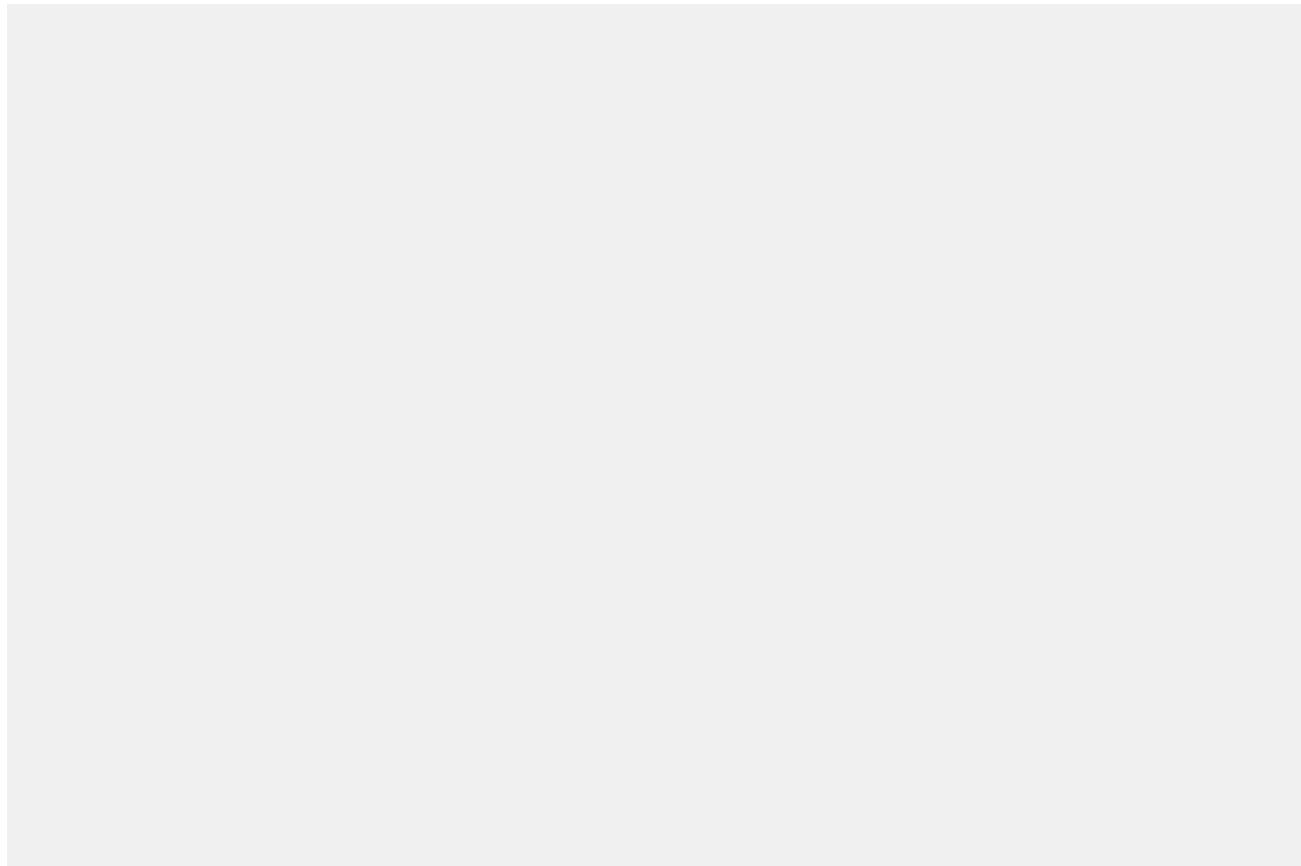
アイデアの静的メソッドを次に示します。

名前	引数	戻り値	説明
	Idea <i>idea</i>	ID[]	<i>idea</i> の件名に基づき、類似アイデアのリストを返します。各コールは、プロセスで使用できる SOSL ステートメントガバナーの制限にカウントされます。
	String <i>userID</i> String <i>communityID</i>	ID[]	指定されたユーザまたはゾーンで最近コメントが投稿されたアイデアを返します。既読および未読のすべてのコメントが含まれます。
	String <i>userID</i> String <i>communityID</i>	ID[]	既読とマークされた、最近コメントが投稿されたアイデアを返します。
	String <i>userID</i> String <i>communityID</i>	ID[]	未読とマークされた、最近コメントが投稿されたアイデアを返します。
	String <i>ideaID</i>	Void	現在ログインしているユーザのすべてのコメントを既読に設定します。

アイデアについての詳細は、Salesforce オンラインヘルプの「アイデアの使用」を参照してください。

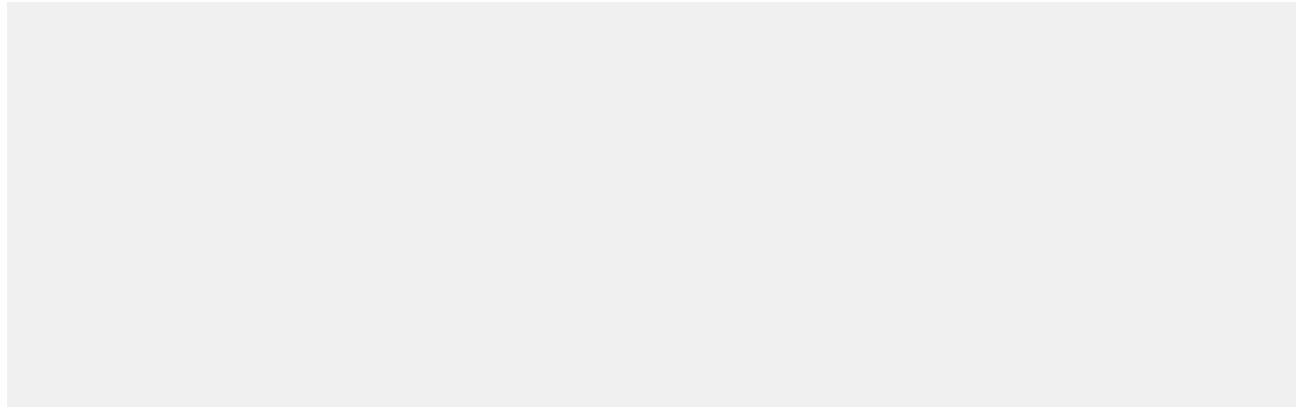
#### アイデアの例

次に、特定のゾーン内で、新しいアイデアと似た件名のアイデアを検索する例を示します。

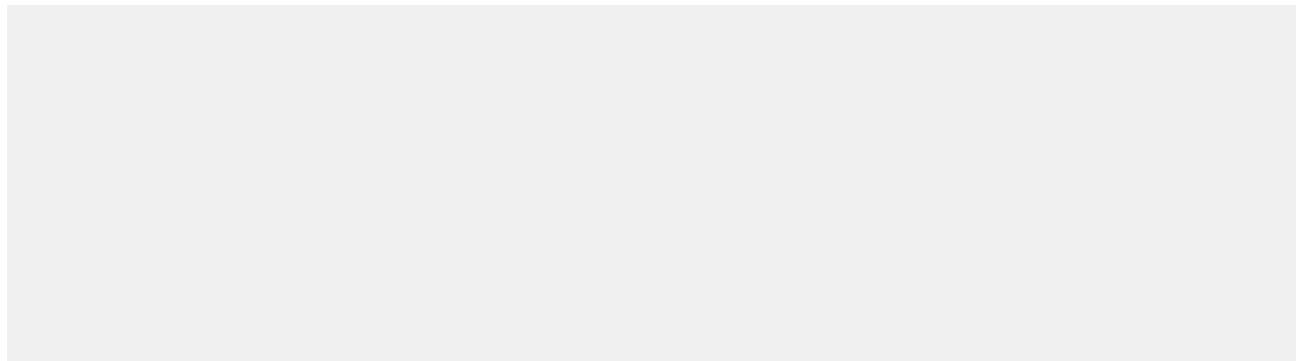


次に、Visualforce ページと、特別な Apex クラスであるカスタムコントローラの両方を使用する例を示します。Visualforce についての詳細は、『[Visualforce 開発者ガイド](#)』を参照してください。

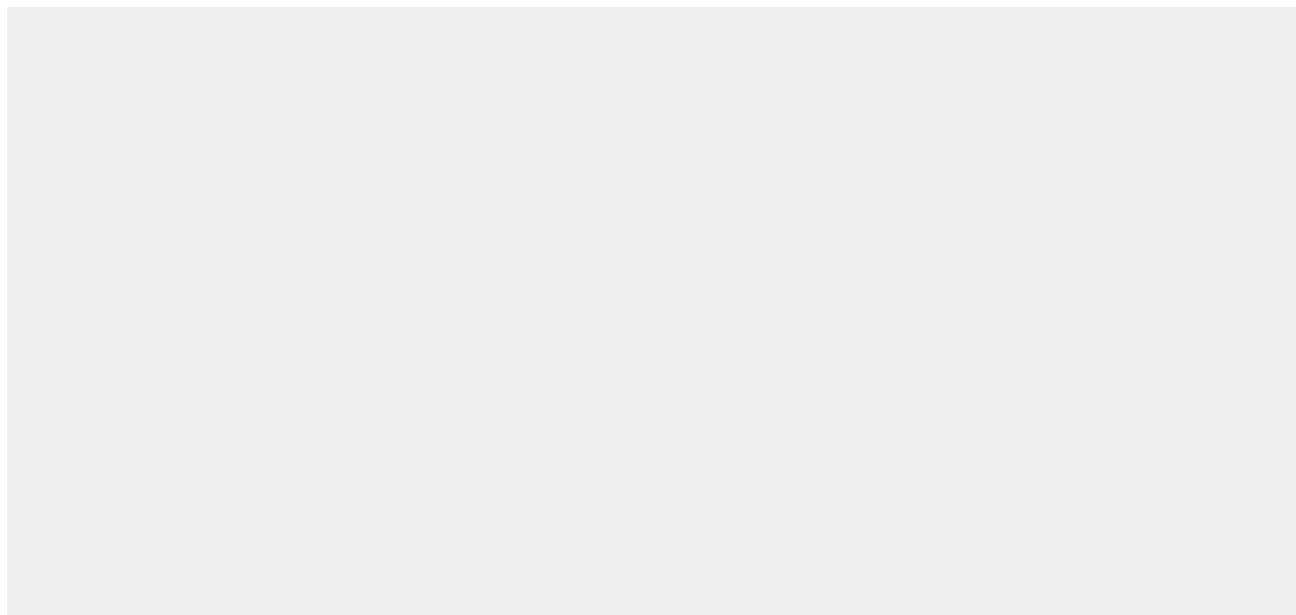
この例では、未読の最近のコメントを返すコントローラに Apex メソッドを作成します。この例は、  
メソッドおよび メソッドでも活用できます。この例が動作するには、ゾーンに投稿されているアイデアが必要となります。さらに、最低1人のゾーンメンバーが別のゾーンのメンバーのアイデアやコメントにコメントを投稿していかなければなりません。 照してく水 草び状體



次に、上記のカスタムコントローラを使用して、未読の最近のコメントリストを作成する Visualforce ページのマークアップを示します。

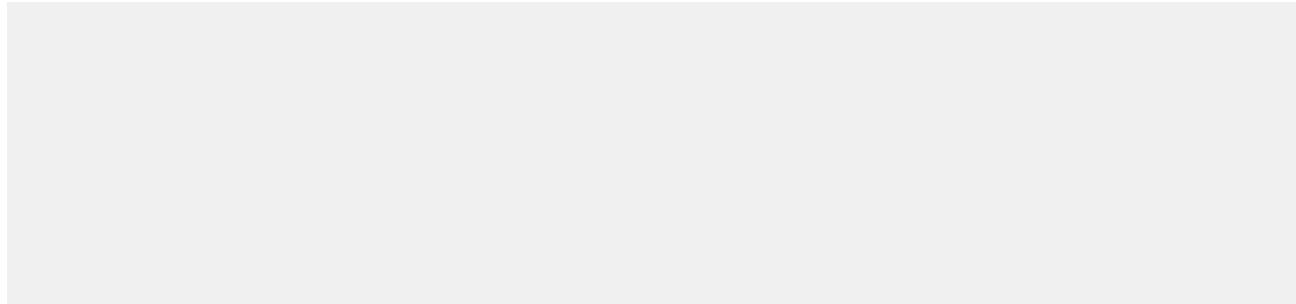


次に、データのリストに Visualforce ページとカスタムコントローラを使用する例を示します。次に、2 つ目の Visualforce ページとカスタムコントローラを使用して特定のアイデアを表示し、既読に設定する方法を示します。この例が動作するには、ゾーンに投稿されているアイデアが必要となります。





次に、上記のカスタムコントローラを使用してアイデアを既読として表示する Visualforce ページのマークアップを示します。



## 関連リンク

- [Apex Community \(Zone\) クラス](#)
- [Answers クラス](#)
- [IdeaStandardController クラス](#)
- [IdeaStandardSetController クラス](#)

## Apex メールクラス

Apex には、Salesforce のメールの送信、受信機能へのアクセスに使用するいくつかのクラスとオブジェクトが含まれます。

詳細は、以下を参照してください。

- [受信メール](#)
- [送信メール](#)

## 送信メール

個別メール送信または一括メール送信に Apex を使用することができます。メールには、件名、BCC アドレスなど標準的なメールの属性をすべて含めることができます。Salesforce メールテンプレートを使用することが可能で、平文テキスト、HTML 形式または Visualforce で生成されたテンプレートのいずれかを選択できます。



メモ: Visualforce メールテンプレートは一括メール送信には使用できません。

Salesforce を使用し、メールが送られた日付、最初に開かれた日付と最後に開かれた日付、開かれた合計回数など HTML 形式のメールの状況を追跡できます (詳細は、Salesforce オンラインヘルプの「HTML メールの追跡」を参照してください)。

個別メール送信または一括メール送信に Apex を使用するには、次のクラスを使用します。

## SingleEmailMessage

単一のメールメッセージの送信に使用されるメールオブジェクトをインスタンス化します。構文は次のとおりです。

## MassEmailMessage

メールメッセージの一括メール送信に使用されるメールオブジェクトをインスタンス化します。構文は次のとおりです。

## Messaging

静的な メソッドを含みます。このメソッドでは、 クラスまたは  
クラスのいずれかを使用してインスタンス化するメールオブジェクトを送信し、  
SendEmailResult オブジェクトを返します。

メールを送信する構文は次のとおりです。

は または のいずれかとなります。

(省略可能) `opt_allOrNone` パラメータでは、任意のメッセージがエラーで失敗した場合、  
その他すべてのメッセージの配信を行わない( )か、エラーのないメッセージの配信を行う( )かを  
指定します。デフォルトは、 です。

静的な メソッドおよび メソッドを含みます。これらのメソッドはメールを送信する前にコールし、トランザクションをコミットしてメールを送信するときに、送信する組織の1日あたりのメール送信量の制限を超えていないことを確認します。構文は次のとあります。

*count*

あよび

count

ここで、`count` はメールの送信先アドレスの総数を示します。

次の点に注意してください。

- Apex トランザクションがコミットされるまでメールは送信されません。

- メソッドをコールしているユーザのメールアドレスは、メールヘッダーの 送信元アドレス 項目に挿入されます。返信されたメール、不達メールおよび外出中の自動返信メールは、このメソッドをコールしているユーザに送信されます。
- トランザクションあたり最大 10 個の メソッド。Limits メソッドを使用して、トランザクション内の メソッドの数を確認します。
- メソッドで送信される単一のメールメッセージは、送信する組織の 1 日の単一メール制限にカウントされます。個の制限値に達すると、 を使用する メソッドへのコールは拒否され、ユーザは エラーコードを受信します。ただし、Salesforce アプリケーションを通して送られた単一メールは許可されます。
- メソッドで送信される一括メールメッセージは、送信する組織の 1 日の一括メール制限にカウントされます。個の制限値に達すると、 を使用する メソッドへのコールは拒否され、ユーザは エラーコードを受信します。
- SendEmailResult オブジェクトで返されるすべてのエラーは、メールが送信されなかったことを表します。

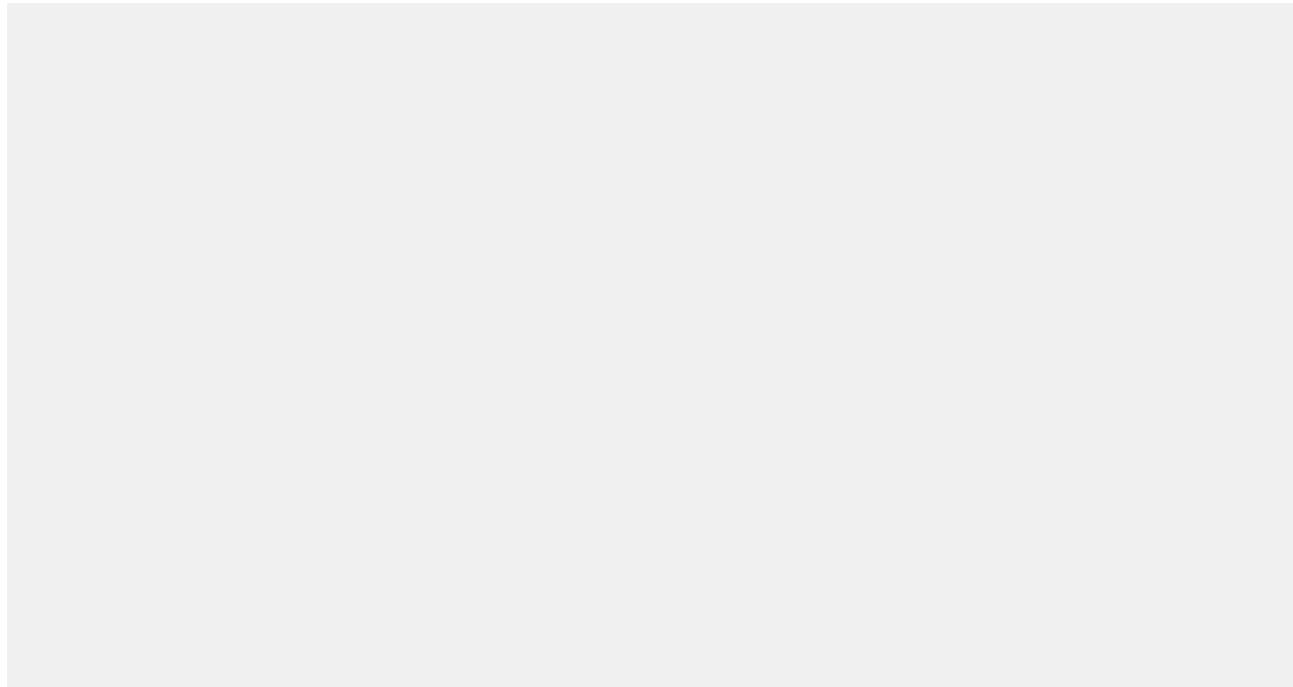
には というメソッドがあります。  
オブジェクトのオブジェクト ID を受け取ります。 に有効な ID が渡されると、 項目が、ログインユーザの 表示名 ではなく、メールヘッダーに使用されます。ヘッダーの送信メールアドレスも、 で定義された項目に設定されます。

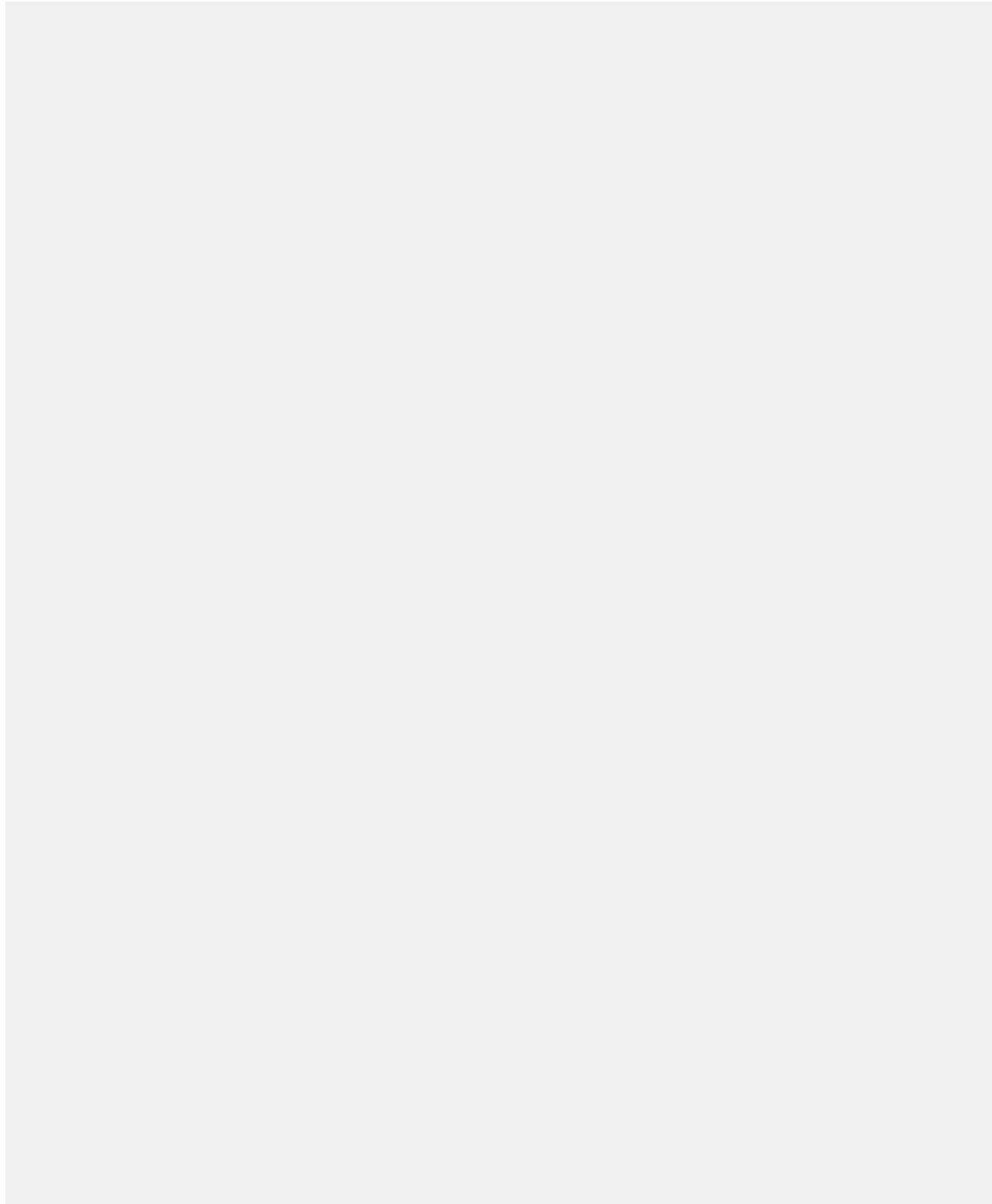


メモ: および の両方が定義されると、  
エラーが発生します。

詳細は、Salesforce オンラインヘルプの「組織の共有アドレス」を参照してください。

## 例





詳細は、以下を参照してください。

- [基本メールメソッド](#)

- SingleEmailMessage メソッド
- MassEmailMessage メソッド
- EmailFileAttachment メソッド
- メッセージメソッド
- Messaging.SendEmailResult オブジェクトメソッド
- SendEmailError Object メソッド

### 基本メールメソッド

次の表に、單一メール送信および一括メール送信のどちらにも使用されるメールオブジェクトメソッドを示します。



**メモ:** テンプレートが使用されていない場合、すべてのメールコンテンツは平文テキスト、HTML、またはその両方で書かれている必要があります。Visualforce メールテンプレートは一括メール送信には使用できません。

名前	引数の型	戻り値	説明
	Boolean	Void	<p>送られるメールのコピーを、メール送信者が受け取るかどうかを示します。一括メール送信では、送信者は最初に送られるメールにのみコピーされます。</p> <p> メモ: BCC コンプライアンスオプションが組織レベルで設定されている場合、ユーザは BCC アドレスを標準のメッセージに追加することができません。</p> <p>「」というエラーコードが返されます。BCC コンプライアンスについては、salesforce.com の担当者にお問い合わせください。</p>
	String	Void	省略可能。受信者が返信した場合に、メッセージを受け取るメールアドレス。
	ID	Void	<p>このメールを作成するためにマージされるテンプレートの ID。、またはの値を指定する必要があります。または、およびの両方を定義できます。</p>

名前	引数の型	戻り値	説明
			 メモ: と は、一括メール送信メソッドではなく、単一メール送信メソッドにのみ適用されます。
	Boolean	Void	省略可能。デフォルト値は で、メールが活動として保存されます。この引数は、受信者リストが または に基づいている場合のみ適用されます。HTMLメールの追跡が組織で有効になっている場合は、メールが開かれた確率を追跡することができます。
	String	Void	省略可能。メールの From 行に表示される名前。SingleEmailMessage の に関するオブジェクトが 項目を定義している場合、設定できません。
	Boolean	Void	そのユーザが設定された署名を持っている場合、メールがメール署名を含むかどうか示します。デフォルトは、 で、 を指定しない限り、ユーザはメールに署名が含まれます。

### SingleEmailMessage メソッド

次の表に、單一メール送信に使用されるメールオブジェクトメソッドを示します。基本メールメソッドへの追加メソッドです。

名前	引数の型	戻り値	説明
	String[]	Void	省略可能。ブラインドカードボンコピー(BCC) アドレスのリスト。最大値は 25 です。テンプレートを使用していない場合のみこの引数を使用できます。次のいずれかの項目の少なくとも 1 つの値を指定しなければなりません。 、 、 、 。

名前	引数の型	戻り値	説明
			<p>BCC コンプライアンスオプションが組織レベルで設定されている場合、ユーザは BCC アドレスを標準のメッセージに追加することができません。次のエラーコードが返されます:</p> <p>。BCC コンプライアンスについては、salesforce.com の担当者にお問い合わせください。</p>
	String[]	Void	<p>省略可能。カーボンコピー (CC) アドレスのリスト。最大値は 25 です。テンプレートを使用していない場合のみこの引数を使用できます。</p> <p>すべてのメールは次の受信者の値を少なくとも 1 つ含んでいなければなりません。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> </ul>
	String	Void	省略可能。メール用の文字セット。この値が null の場合、ユーザのデフォルト値が使われます。
	ID[]	Void	省略可能。メールに添付する各ドキュメントオブジェクトの ID を含むリスト。添付文書の合計が 10 MB を超えない限り、いくつでも文書を追加できます。
	EmailFileAttachment[]	Void	省略可能。メールに添付するバイナリファイルとテキストファイルのファイル名を含むリスト。添付ファイルの合計が 10 MB を超えない限り、いくつでもファイルを追加できます。
	String	Void	<p>省略可能。メールの HTML 版(送信者による指定)。組織に関連付けられた仕様に従って、値は符号化されます。</p> <p>、または の値を指定する必要が あります。または、および</p>

名前	引数の型	戻り値	説明
			の両方を定義できます。
	String	Void	省略可能。送信メールの In-Reply-To 項目。この返信メールの相手となるメール(親メール)を示します。親メールまたはメールのメッセージ ID が含まれます。
	String	Void	省略可能。メールのテキスト版(送信者による指定)、 、または の値を指定する必要があります。または、 および の両方を定義できます。
	ID	Void	省略可能。送信メールに関連する組織の共有アドレスの ID。 項目がすでに設定されている場合、 項目は設定できません。
	String	Void	省略可能。送信メールの References 項目。メールスレッドを示します。親メールの References 項目およびメッセージ ID、In-Reply-To 項目のリストが含まれます。
	String	Void	省略可能。メールの件名行。メールテンプレートを使用している場合、この値はテンプレートの件名で上書きされます。
	ID	Void	テンプレートを使用している場合は必須ですが、使用していない場合は省略可能です。メールを送信する取引先責任者、リード、ユーザの ID。指定する ID によりコンテキストが設定され、テンプレートの差し込み項目に正しいデータが含まれていることを保証します。  メール送信除外 オプションが選択されている ID やレコードを指定しないでください。  すべてのメールは次の受信者の値を少なくとも 1 つ含んでいなければなりません。  • • •

名前	引数の型	戻り値	説明
			<ul style="list-style-type: none"> <li>•</li> <li>•</li> </ul>
	String[]	Void	<p>省略可能。メールの送信先のメールアドレスのリスト。メールアドレス数の最大値は 100 です。テンプレートを使用していない場合のみこの引数を使用できます。</p> <p>すべてのメールは次の受信者の値を少なくとも 1 つ含んでいなければなりません。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> </ul>
	ID	Void	<p>省略可能。項目に取引先責任者を指定する場合、も指定することができます。これにより、テンプレート内の差し込み項目が適切なデータを含んでいることが確実に保証されるようになります。値は、次の型のいずれかです。</p> <ul style="list-style-type: none"> <li>• Account</li> <li>• Asset</li> <li>• Campaign</li> <li>• Case</li> <li>• Contract</li> <li>• Opportunity</li> <li>• Order</li> <li>• Product</li> <li>• Solution</li> <li>• Custom</li> </ul>

### MassEmailMessage メソッド

次の表に、一括メール送信に使用される一意のメールオブジェクトメソッドを示します。基本メールメソッドへの追加メソッドです。

名前	引数の型	戻り値	説明
	String	Void	メールの説明。

名前	引数の型	戻り値	説明
	ID[]	Void	<p>メールを送信する取引先責任者、リード、ユーザの ID のリスト。指定する ID によりコンテキストが設定され、テンプレートの差し込み項目に正しいデータが含まれていることを保証します。オブジェクトはすべて同じ型でなければなりません(すべての取引先責任者、リード、ユーザ)。1回のメール送信で最大 250 まで ID をリストすることができます。</p> <p>項目の値を指定した場合、必要に応じて、メールのコンテキストを規定する <code>to</code>、ユーザ、取引先責任者、またはリードに設定することができます。これにより、テンプレート内の差し込み項目が適切なデータを含んでいることが保証されます。メール送信除外 オプションが選択されている ID やレコードを指定しないでください。</p> <p>すべてのメールは次の受信者の値を少なくとも 1 つ含んでいなければなりません。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> </ul>
	ID[]	Void	<p>省略可能。</p> <p>項目に取引先責任者のリストを指定した場合、そのリストも同様に指定できます。これにより、テンプレート内の差し込み項目が適切なデータを含んでいることが確実に保証されるようになります。値は、次の型のいずれかです。</p> <ul style="list-style-type: none"> <li>• Contract</li> <li>• Case</li> <li>• Opportunity</li> <li>• Product</li> </ul> <p> メモ: を指定する場合 は、ごとに 1 つ</p>

名前	引数の型	戻り値	説明
			指定します。それ以外の場合、エラーが発生します。

また、

クラスには、基本メールメッセージメソッドへのアクセス権があります。

名前	引数の型	戻り値	説明
	Boolean	Void	送られるメールのコピーを、メール送信者が受け取るかどうかを示します。一括メール送信では、送信者は最初に送られるメールにのみコピーされます。
			 メモ: BCC コンプライアンスオプションが組織レベルで設定されている場合、ユーザは BCC アドレスを標準のメッセージに追加することができません。次のエラーコードが返されます:  。BCC コンプライアンスについては、salesforce.com の担当者にお問い合わせください。
	String	Void	省略可能。受信者が返信した場合に、メッセージを受け取るメールアドレス。
	ID	Void	このメールを作成するためにマージされるテンプレートの ID。 、または の値を指定する必要があります。または、 および の両方を定義できます。
			 メモ: と は、一括メール送信メソッドではなく、単一メール送信メソッドにのみ適用されます。

名前	引数の型	戻り値	説明
	Boolean	Void	省略可能。デフォルト値は <code>True</code> で、メールが活動として保存されます。この引数は、受信者リストが <code>From</code> または <code>To</code> に基づいている場合のみ適用されます。HTML メールの追跡が組織で有効になっている場合は、メールが開かれた確率を追跡することができます。
	String	Void	省略可能。メールの <code>From</code> 行に表示される名前。SingleEmailMessage の <code>FromName</code> に関連するオブジェクトが <code>From</code> 項目を定義している場合、設定できません。
	Boolean	Void	そのユーザが設定された署名を持っている場合、メールがメール署名を含むかどうか示します。デフォルトは、 <code>False</code> で、を指定しない限り、ユーザはメールに署名が含まれます。

### EmailFileAttachment メソッド

EmailFileAttachment オブジェクトは、Salesforce の既存のドキュメントに対し、SingleEmailMessage オブジェクト内で要求の一部として渡される添付ファイルを指定するのに使用します。

名前	引数の型	戻り値	説明
<code>blob attachment</code>	Void		添付ファイル自体。
<code>String content_type</code>	Void		添付ファイルのコンテンツタイプ。
<code>String file_name</code>	Void		添付するファイルの名前。
<code>boolean Content-Disposition</code>	Void		Content-Dispositions がインラインか ( <code>inline</code> ) 添付ファイルか ( <code>attachment</code> ) を示します。多くの場合、インラインコンテンツは、メッセージ表示時にユーザに表示されます。添付ファイルコンテンツは、ユーザのアクションに基づいて表示されます。

### メッセージメソッド

次の表に、単一メール送信または一括メール送信に使用されるメッセージメソッドを示します。

名前	引数の型	戻り値	説明
	integer <i>AmountReserved</i>	Void	<p>現在のトランザクションがコミットされた後に、指定数のメールアドレスに一括メール送信するためのメール容量を確保します。トランザクションの結果として送信するメールの数を事前に把握している場合は、このメソッドをコールできます。このトランザクションで組織の1日あたりのメール送信量の制限を超える場合、このメソッドを使用すると、</p> <p style="text-align: right;">というエラーになります。</p>
	integer <i>AmountReserved</i>	Void	<p>現在のトランザクションがコミットされた後に、指定数のメールアドレスに単一メール送信するためのメール容量を確保します。トランザクションの結果として送信するメールの数を事前に把握している場合は、このメソッドをコールできます。このトランザクションで組織の1日あたりのメール送信量の制限を超える場合、このメソッドを使用すると、</p> <p style="text-align: right;">というエラーになります。</p>
Messaging.Email[]	<a href="#">Messaging.SendEmailResult[]</a>		または
Boolean <i>allOrNothing</i>			<p>のいずれかを使用してインスタンス化するメールオブジェクトのリストを送信し、<a href="#">SendEmailResult</a> オブジェクトのリストを返します。</p> <p>(省略可能) <i>opt_allOrNone</i> パラメータでは、任意のメッセージがエラーで失敗した場合、でその他すべてのメッセージの配信を行わない( )か、エラーのないメッセージの配信を行う( )かを指定します。デフォルトは、です。</p>
List <ID> <i>emailMessageIds</i>	<a href="#">Messaging.SendEmailResult[]</a>		<p>指定したメールメッセージ ID で定義されているドラフトメールメッセージを最大 10 件送信し、<a href="#">SendEmailResult</a> オブジェクトのリストを返します。</p> <p>メソッドは、<i>opt_allOrNone</i> パラメータ(省略可能)は常に であるとみなされ、設定した値を無視します。この省略可能なパラメータでは、任意のメッセージがエラーで失敗した場合、でその他すべてのメッセージの配信を行わない( )か、エラーの</p>

名前	引数の型	戻り値	説明
			<p>ないメッセージの配信を行う( )かを指定します。</p> <p>例:</p> <p>この例では、ドラフトメールメッセージを送信する方法を示します。ケースとそのケースに関連付けられた新しいメールメッセージを作成します。次に、ドラフトメールメッセージを送信し、結果を確認します。この例を実行する前に、メールアドレスを有効なアドレスに置き換えてください。</p>

名前	引数の型	戻り値	説明

### Messaging.SendEmailResult オブジェクトメソッド

メソッドは、SendEmailResult オブジェクトのリストを返します。各 SendEmailResult オブジェクトには、次のメソッドがあります。このメソッドは引数を取りません。

名前	戻り値	説明
	SendEmailError[]	メソッドはの実行時にエラーが発生した場合、SendEmailError オブジェクトが返されます。
	Boolean	メールが正しく送信されたか (true)、されなかったか (false) を示します。 true であっても、受信者がメールを受信したとは限りません。メールアドレスの問題、不達、スパムブロックによるブロックなどが発生する場合があるためです。

### SendEmailError Object メソッド

SendEmailResult オブジェクトには SendEmailError が含まれている場合があります。SendEmailError には次のメソッドがあります。このメソッドは引数を取りません。

名前	戻り値	説明
	String[]	1つ以上の項目名のリスト。オブジェクト内の項目でエラー条件に影響を与えるものが存在する場合、その項目を示します。
	String	エラーメッセージのテキスト。
	System.StatusCode	エラーを特徴付けるコード。状況コードの完全なリストは、組織の WSDL ファイルから入手できます。組織の WSDL ファイルへのアクセスについての詳細は、Salesforce オンラインヘルプの「Salesforce WSDL およびクライアント認証証明書のダウンロード」を参照してください。
	String	エラーが発生した対象レコードの ID。

### 受信メール

Apex を使用してメールと添付ファイルの受信および処理を行うことができます。メールは Apex メールサービスで受信し、InboundEmail オブジェクトを使用する Apex クラスで処理します。



メモ: Apex メールサービスは、Developer、Enterprise、Unlimited Edition 組織でのみ利用可能です。

この項で説明する内容は次のとおりです。

- [Apex メールサービスとは](#)
- [InboundEmail オブジェクトの使用](#)
- [InboundEmail オブジェクト](#)
- [InboundEmail.Header オブジェクト](#)
- [InboundEmail.BinaryAttachment オブジェクト](#)
- [InboundEmail.TextAttachment オブジェクト](#)
- [InboundEmailResult オブジェクト](#)
- [InboundEnvelope オブジェクト](#)

### Apex メールサービスとは

メールサービスは、Apex クラスを使用して、受信メールの内容、ヘッダーおよび添付ファイルを処理する自動化プロセスです。たとえば、メッセージに含まれる取引先責任者情報に基づいて、取引先責任者レコードを作成できます。

各メールサービスには、Salesforce が生成したメールアドレスを 1 つ以上関連付けることができ、ユーザはそのアドレス宛てに処理を求めるメッセージを送信できます。複数ユーザに 1 つのメールサービスへのアクセス権を与える手順は、次のとおりです。

- Salesforce が生成した複数のメールアドレスをメールサービスに関連付け、これらのアドレスをユーザに割り当てます。
- Salesforce が生成した単一のメールアドレスをメールサービスに関連付け、メールサービスにアクセスするユーザに従って実行する Apex クラスを記述します。たとえば、ユーザのメールアドレスに基づいてユーザを識別し、そのユーザのレコードを作成する Apex クラスを記述します。

メールサービスを使用するには、[設定] から [開発] > [メールサービス] をクリックします。

- 新しいメールサービスを定義するには、[新規メールサービス] をクリックします。
- 既存のメールサービスを選択して、その設定の表示、有効化または無効化、およびそのメールサービス用のアドレスの表示または指定を行います。
- 既存のメールサービスを変更するには、[編集] をクリックします。
- メールサービスを削除するには、[削除] をクリックします。



メモ: メールサービスを削除する前に、関連するメールサービスアドレスをすべて削除する必要があります。

メールサービスを定義するときには、次の点に注意してください。

- メールサービスは、そのアドレスの 1 つが受信したメッセージを処理するだけです。
- Salesforce は、[オンデマンドメール-to-ケース] など、すべてのメールサービスを合計した 1 日に処理できるメッセージの総数を制限します。この制限を超えたメッセージは、各メールサービスの失敗時の応答設定に

基づいて、戻す、破棄する、あるいは翌日処理するためのキューに入れます。Salesforce は、ユーザライセンス数 × 1,000 で制限値を算出します。1 日の最大は 1,000,000 件です。たとえば、ライセンス数が 10 の場合、1 日最大 10,000 件のメールメッセージを処理できます。

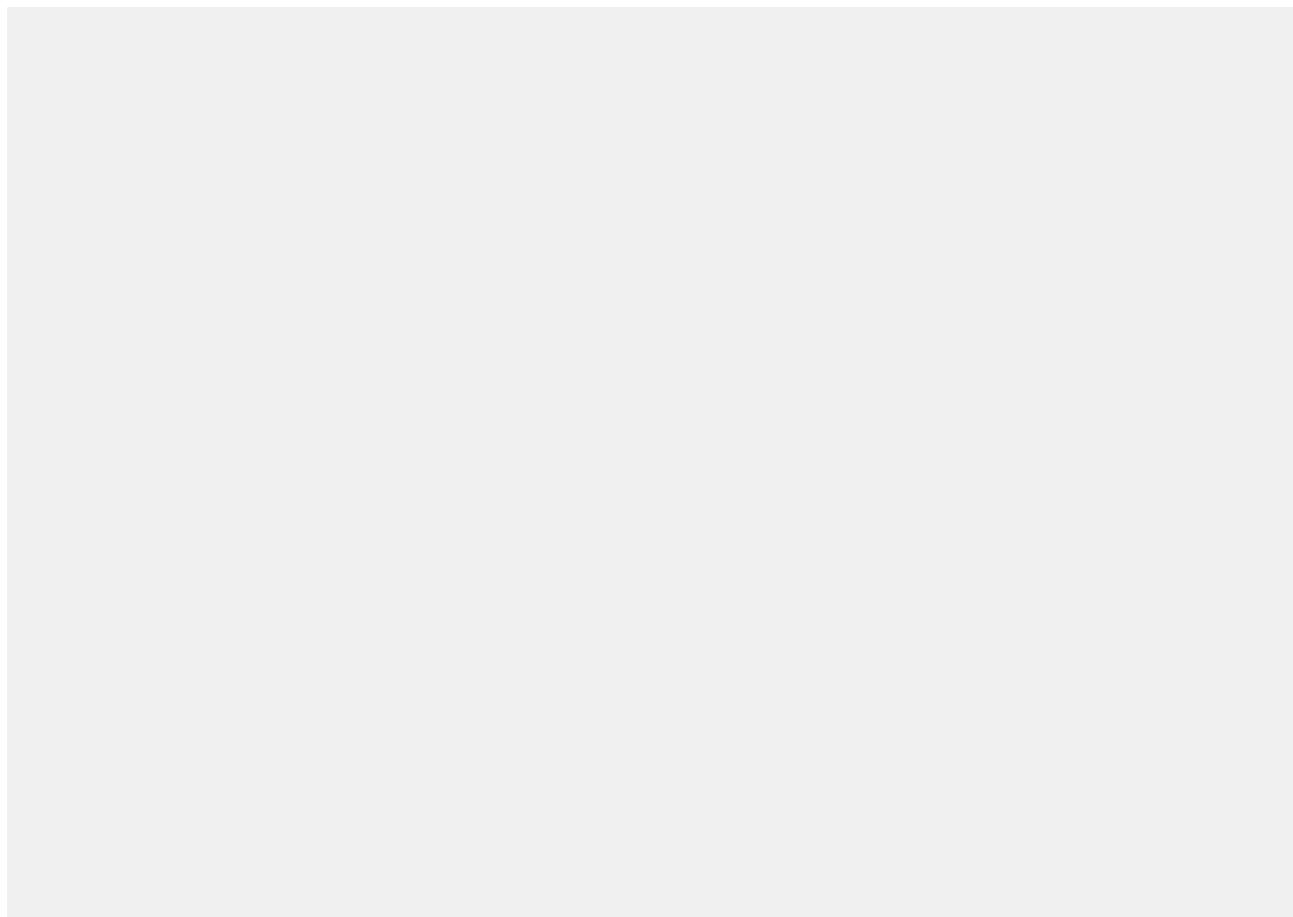
- sandbox 内に作成したメールサービスアドレスは、本番組織にコピーできません。
- メールサービスごとに Salesforce に通知して、送信者のメールアドレスではなく、特定のアドレスにエラー メールメッセージを送信できます。
- メール (結合された本文テキスト、本文 HTML および添付ファイル) が約 10 MB を超える場合 (言語や文字 セットに応じて異なる)、メールサービスはメールメッセージを拒否し、送信者に通知します。

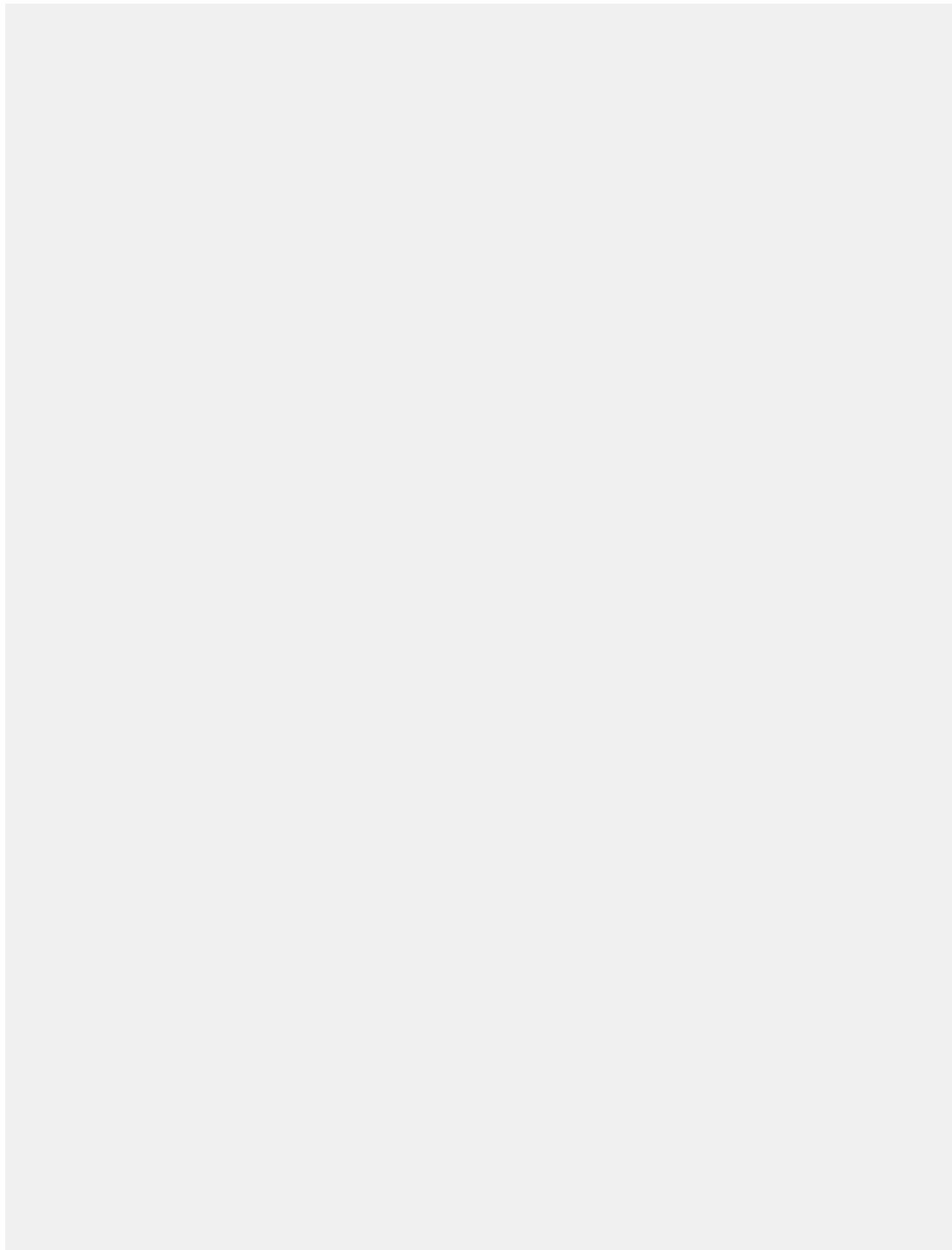
### InboundEmail オブジェクトの使用

Apex メールサービスドメインが受信するすべてのメールについて、Salesforce は、そのメールの内容と添付ファイルを含む個別の InboundEmail オブジェクトを作成します。  
インターフェースを実装する Apex クラスを使用して、受信メールメッセージを処理できます。そのクラスで  
メソッドを使用して、InboundEmail オブジェクトにアクセスし、受信メールメッセージの内容、ヘッダー、およ  
び添付ファイルの取得と、その他多数の機能を実行することができます。

#### 例 1: 取引先責任者の ToDo の作成

受信メールアドレスに基づいて取引先責任者を検索し、新規 ToDo を作成する方法の例は次のとおりです。





## InboundEmail オブジェクト

InboundEmail オブジェクトには、次の項目があります。

名前	型	説明
	InboundEmail.BinaryAttachment[]	そのメールで受信したバイナリ添付ファイルのリスト(存在する場合)。 バイナリ添付ファイルの例としては、画像、音声、アプリケーション、映像ファイルなどがあります。
	String[]	カーボンコピー(CC)アドレスのリスト(存在する場合)。
	String	[送信者] 項目に表示されるメールアドレス。
	String	[送信者] 項目に表示される名前(存在する場合)。
	InboundEmail.Header[]	メールの RFC 2822 ヘッダーのリスト。具体的なヘッダーは次のとおりです。 <ul style="list-style-type: none"><li>• Recieved from</li><li>• Custom headers</li><li>• Message-ID</li><li>• Date</li></ul>
	String	メールの HTML 版(送信者が指定した場合)。
	Boolean	HTML 本文テキストを切り捨てる( )か、そのままにする( )かを示します。
	String	受信メールの In-Reply-To 項目。この返信メールの相手となるメール(親メール)を示します。親メールまたはメールのメッセージ ID が含まれます。
	String	メッセージ ID — 受信メールの一意の ID。
	String	メールの平文テキスト版(送信者が指定した場合)。
	Boolean	本文平文テキストを切り捨てる( )か、そのままにする( )かを示します。

名前	型	説明
	String[]	受信メールの References 項目。メールスレッドを示します。親メールの References 項目とメッセージ ID 項目、場合によっては In-Reply-To 項目が含まれます。
	String	reply-to ヘッダーに表示されるメールアドレス。reply-to ヘッダーが存在しない場合、項目と同じになります。
	String	メールの件名 (存在する場合)。
	InboundEmail.TextAttachment[]	そのメールで受信したテキスト添付ファイルのリスト (存在する場合)。 テキスト添付ファイルは次のいずれかです。 <ul style="list-style-type: none"> <li>の Multipurpose Internet Mail Extension (MIME) タイプの添付ファイル</li> <li>の MIME タイプの添付ファイルと、または 拡張子で終わるファイル名の添付ファイル。これらはそれぞれ、および MIME タイプとして保存されます。</li> </ul>
	String[]	項目に表示されるメールアドレス。

### InboundEmail.Header オブジェクト

InboundEmail オブジェクトは、InboundEmail.Header オブジェクトに RFC 2822 メールヘッダー情報と次の項目を格納します。

名前	型	説明
	String	や など、ヘッダーパラメータの名前。
	String	ヘッダーの値。

### InboundEmail.BinaryAttachment オブジェクト

InboundEmail オブジェクトは、InboundEmail.BinaryAttachment オブジェクトにバイナリ添付ファイルを格納します。

バイナリ添付ファイルの例としては、画像、音声、アプリケーション、映像ファイルなどがあります。

`InboundEmail.BinaryAttachment` オブジェクトには、次の項目があります。

名前	型	説明
	Blob	添付ファイルの本文。
	String	添付ファイルの名前。
	String	プライマリおよびサブ MIME タイプ。

### InboundEmail.TextAttachment オブジェクト

`InboundEmail` オブジェクトは、`InboundEmail.TextAttachment` オブジェクトにテキスト添付ファイルを格納します。

テキスト添付ファイルは次のいずれかです。

- の Multipurpose Internet Mail Extension (MIME) タイプの添付ファイル
- の MIME タイプの添付ファイルと、または 拡張子で終わるファイル名の添付ファイル。これらはそれぞれ、および MIME タイプとして保存されます。

`InboundEmail.TextAttachment` オブジェクトには、次の項目があります。

名前	型	説明
	String	添付ファイルの本文。
	Boolean	添付ファイルの本文テキストを切り捨てる( )か、そのままにする( )かを示します。
	String	body 項目の元の文字セット。body は、Apex メソッドに入力されるときに UTF-8 で符号化し直されます。
	String	添付ファイルの名前。
	String	プライマリおよびサブ MIME タイプ。

### InboundEmailResult オブジェクト

`InboundEmailResult` オブジェクトにより、メールサービスの結果が返されます。このオブジェクトが `null` であれば、結果は正常とみなされます。`InboundEmailResult` オブジェクトには、次の項目があります。

名前	型	説明
	Boolean	メールが正常に処理されたかどうかを示す値。 この場合は、Salesforce が受信メールを拒否し、項目で指定されているメッセージを含む返信メールを元の送信者に送信します。
	String	Salesforce が返信メールの本文で返すメッセージ。この項目には、項目で返された値に関係なく、テキストを取り込むことができます。

### InboundEnvelope オブジェクト

InboundEnvelope オブジェクトには、受信メールに関連するエンvelope 情報が保管されます。次の項目があります。

名前	型	説明
	String	項目に表示される名前(存在する場合)。
	String	項目に表示される名前(存在する場合)。

## Apex のサポートクラス

サポートクラスを使用すると、営業時間やケースなど、サポートセンターで一般的に使用されるレコードを操作できます。Apex には、次のサポートクラスがあります。

- [Cases クラス](#)
- [Cases クラス](#)

### BusinessHours クラス

BusinessHours では、複数のタイムゾーンなど、カスタマーサポートチームが活動するさまざまな営業時間を指定することができます。

BusinessHours メソッドはすべて営業時間の特定のインスタンスによってコールされ、動作します。のインスタンスマソッドを次に示します。

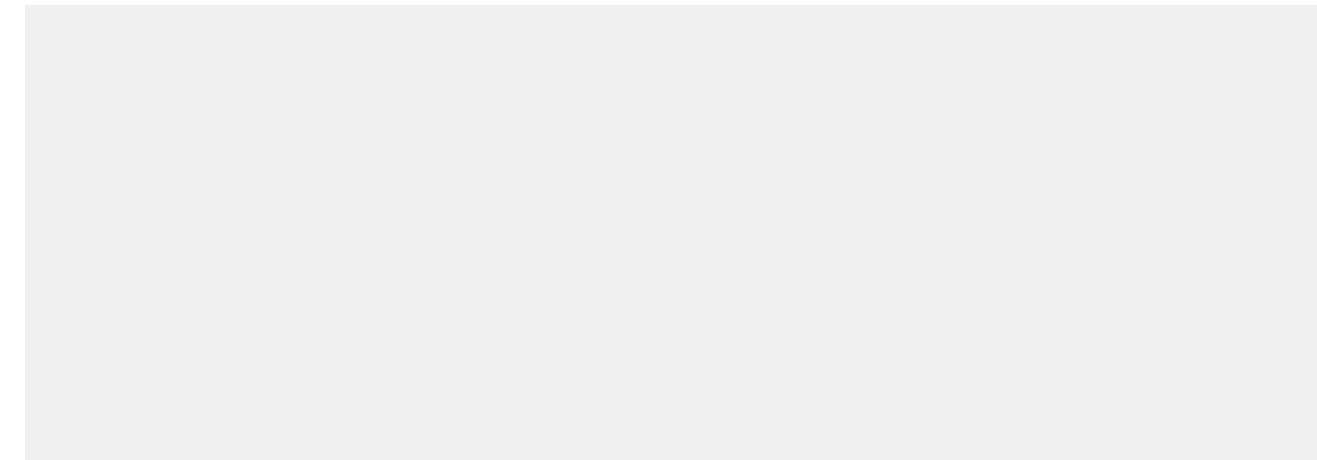
名前	引数	戻り値	説明
	String <i>businessHoursId</i>	Datetime <i>dateTime</i>	開始時の <i>dateTime</i> からの間隔をミリ秒単位で追加して、営業時間のみを辿ります。結果の <i>dateTime</i> をローカルタイムゾーンで返します。

名前	引数	戻り値	説明
	long <i>interval</i>		「BusinessHours の例」(ページ 684)の例を参照してください。
	String <i>businessHoursId</i>	Datetime	開始時のdateTimeからの間隔をミリ秒単位で追加して、営業時間のみを辿ります。結果のdateTimeをGMTで返します。「BusinessHours の例」(ページ 684)の例を参照してください。
	dateTime <i>startDate</i>		
	long <i>interval</i>		
	String <i>businessHoursId</i>	Long	特定の営業時間のセットの開始と終了のdateTimeの差異を返します。「BusinessHours の例」(ページ 684)の例を参照してください。
	dateTime <i>startDate</i>		
	dateTime <i>endDate</i>		

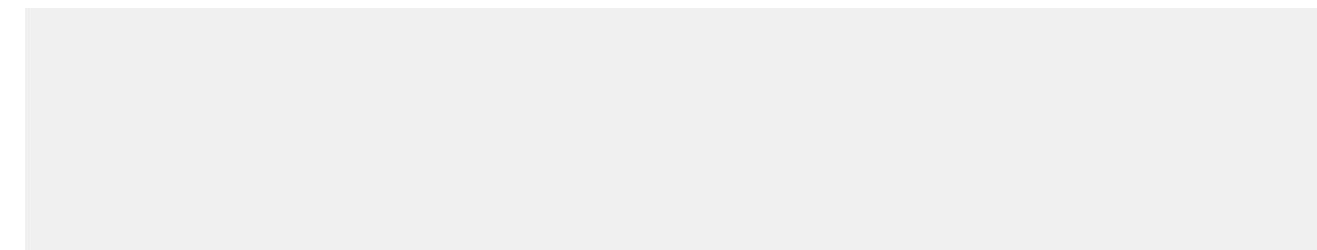
営業時間についての詳細は、Salesforce オンラインヘルプの「営業時間の設定」を参照してください。

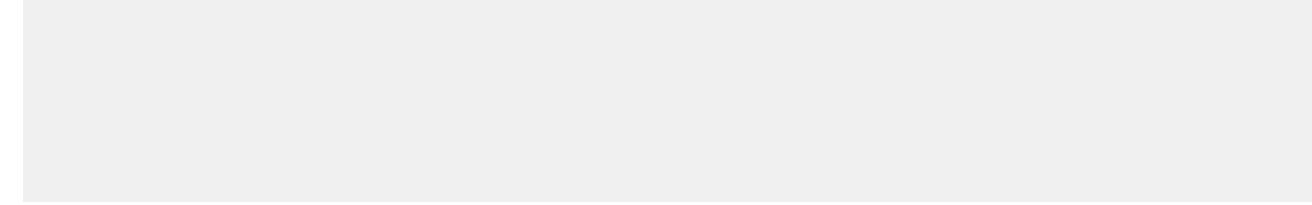
#### BusinessHours の例

次の例では、startTime から 1 営業時間後の時間を求め、ローカルタイムゾーンで dateTime を返します。

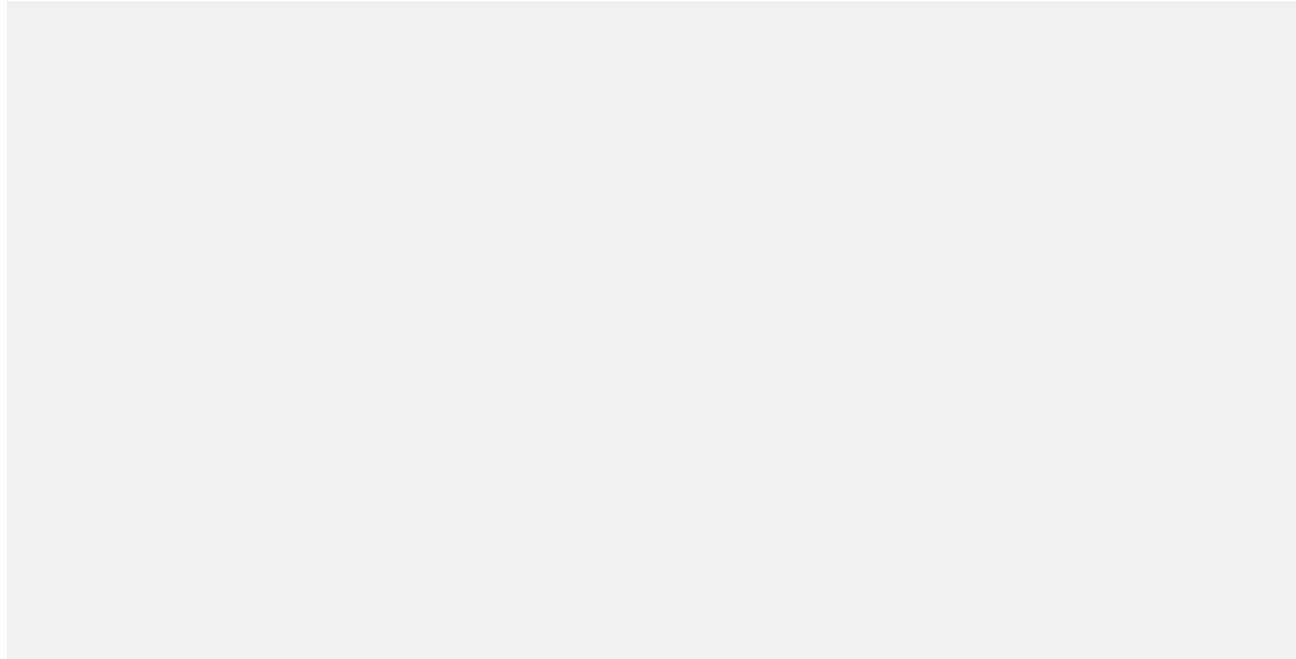


次の例では、startTime から 1 営業時間後の時間を求め、dateTime を GMT で返します。





次の例では、`startTime` と `nextTime` の差異を求めます。



## Cases クラス

### 使用方法

クラスを使用し、ケースレコードを操作します。

### メソッド

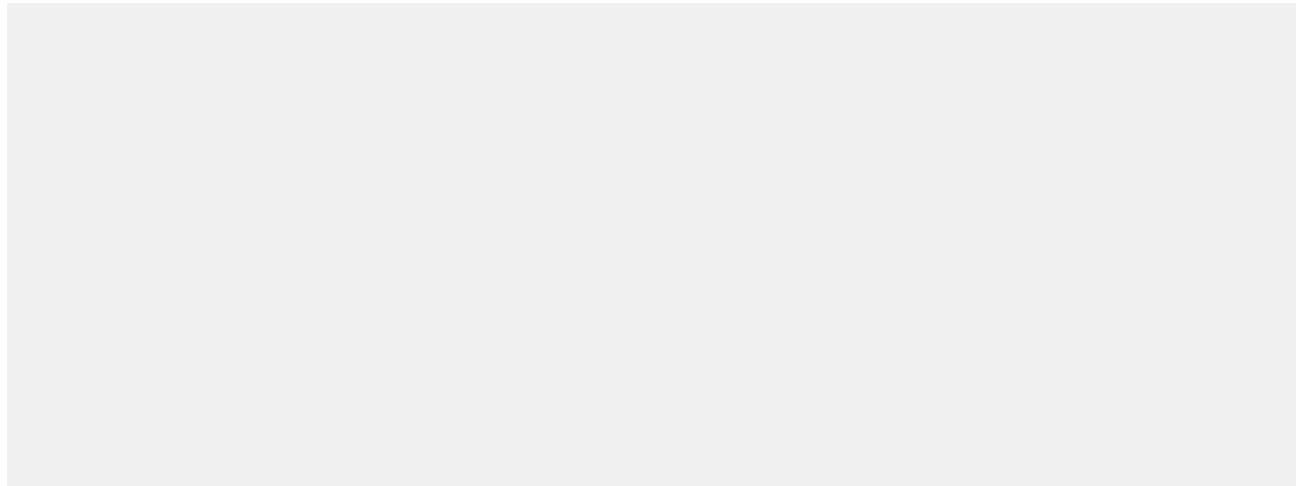
クラスの静的メソッドを次に示します。

メソッド	引数	戻り値	説明
	string <code>emailThreadId</code> ID		指定されたメールスレッド ID に対応するケース ID を返します。 <code>emailThreadId</code> 引数の形式は、 とします。 や

メソッド	引数	戻り値	説明
などの他の形式は無効です。			

### Cases の例

次の例では、メールスレッド ID を使用して、関連するケース ID を取得します。



## Chatter in Apex クラス

Chatter API データにアクセスするには、

名前空間でこのクラスを使用します。

クラスの使用についての詳細は、「[Chatter in Apex の使用](#)」(ページ 307)を参照してください。

### 関連リンク

- [ConnectApi.Chatter クラス](#)
- [ConnectApi.ChatterFavorites クラス](#)
- [ConnectApi.ChatterFeeds クラス](#)
- [ConnectApi.ChatterGroups クラス](#)
- [ConnectApi.ChatterUsers クラス](#)
- [ConnectApi.Communities クラス](#)
- [ConnectApi.Organization クラス](#)
- [ConnectApi.Records クラス](#)
- [ConnectApi 入力クラス](#)
- [ConnectApi 出力クラス](#)
- [ConnectApi Enum](#)
- [ConnectApi 例外](#)

**ConnectApi.Chatter クラス**

レコードのフォロワーと登録に関するアクセス情報

**メソッド**

API バージョン 28.0 では、すべてのメソッドを使用できます。

次に、**クラスの静的メソッドを示します。**

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>subscriptionId</i>	void	<p>指定された登録を削除します。このメソッドを使用して、レコード、ユーザ、またはファイルのフォロー解除を行います。グループを脱退するには、 をコールします。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subscriptionId</i>—登録の ID。</li> </ul>
	String <i>communityId</i> , String <i>recordId</i>	ラス	<p>ク 指定されたコミュニティの指定されたレコードのフォロワーの最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>recordId</i>—レコードまたはキーワード の ID。</li> </ul>
	String <i>communityId</i> , String <i>recordId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	ラス	<p>ク 指定されたレコードのフォロワーの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>recordId</i>—レコードまたはキーワード の ID。</li> </ul>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>pageParam</i>—返すページの数を指定します。0 から開始します。または 0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i>—ページあたりの項目数を指定します。有効な値は 1 ~ 1000 です。を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>subscriptionId</i>	ラス	<p>ク 指定された登録に関する情報を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>subscriptionId</i>—登録の ID。</li> </ul>

## 関連リンク

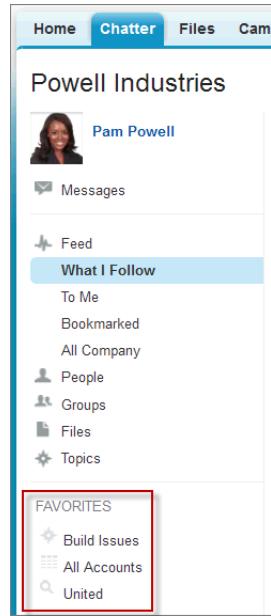
[Chatter in Apex クラス](#)

## ConnectApi.ChatterFavorites クラス

Chatterのお気に入りを使用すると、トピック、リストビュー、およびフィード検索に簡単にアクセスできます。

Chatter in Apex を使用して、お気に入りとして追加されたトピック、リストビュー、およびフィード検索を取得および削除します。トピックとフィード検索をお気に入りとして追加し、フィード検索またはリストビューフィードの最終参照日付を現在のシステム時間に更新します。

salesforce.com の次の画像では、トピックが「Build Issues」で、リストビューが「All Accounts」であり、フィード検索が「United」です。



## 静的メソッド

次に、クラスの静的メソッドを示します。

API バージョン 28.0 では、すべてのメソッドを使用できます。

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>subjectId</i> , String <i>searchText</i>		<p>指定されたコミュニティの指定されたユーザのお気に入りを追加します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>subjectId</i>—コンテキストユーザまたはキーワードの ID を指定します。</li> <li>• <i>searchText</i>—お気に入りとして保存する検索のテキストを指定します。このメソッドでは、リストビューのお気に入りまたはトピックではなく、フィード検索のお気に入りのみを作成できます。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>subjectId</i> , String <i>targetId</i>		<p>トピックをお気に入りとして追加します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subjectId</i>—コンテキストユーザまたはキーワードの ID を指定します。</li> <li>• <i>targetId</i>—お気に入りとして追加するトピックの ID。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> , String <i>favoriteId</i>	Void	<p>指定されたお気に入りを削除します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subjectId</i>—コンテキストユーザまたはキーワードの ID を指定します。</li> <li>• <i>favoriteId</i>—お気に入りの ID。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> , String <i>favoriteId</i>		<p>お気に入りの説明を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subjectId</i>—コンテキストユーザまたはキーワードの ID を指定します。</li> <li>• <i>favoriteId</i>—お気に入りの ID。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> ,		<p>指定されたコミュニティの指定されたユーザのすべてのお気に入りリストを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、 、または のいずれかを使用します。</li> </ul>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>subjectId</i> — コンテキストユーザまたはキーワードの ID を指定します。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> , String <i>favoriteId</i>		<p>指定されたコミュニティの指定されたお気に入りのフィード項目の最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subjectId</i> — コンテキストユーザまたはキーワードの ID を指定します。</li> <li>• <i>favoriteId</i> — お気に入りの ID。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> , String <i>favoriteId</i> , String <i>pageParam</i> , integer <i>pageSize</i> , FeedSortOrder enum <i>sortParam</i>		<p>指定された順番で、指定されたコミュニティ内の指定されたお気に入りのフィード項目の指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subjectId</i> — コンテキストユーザまたはキーワードの ID を指定します。</li> <li>• <i>favoriteId</i> — お気に入りの ID。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 と、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を</li> </ul>

名前	引数	戻り値	説明
			<p>渡すと、デフォルトサイズの 25 に設定されます。</p> <ul style="list-style-type: none"> <li>• <i>sortParam</i> — 値は次のとおりです。           <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p>
	String <i>communityId</i> , String <i>subjectId</i> , String <i>favoriteId</i> , Boolean <i>updateLastViewDate</i>		<p><i>updateLastViewDate</i> に を指定すると、保存された検索またはリストビューフィードの最終参照日付が現在のシステム時間に更新されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subjectId</i> — コンテキストユーザまたはキーワードの ID を指定します。</li> <li>• <i>favoriteId</i> — お気に入りの ID。</li> <li>• <i>updateLastViewDate</i> — 指定されたお気に入りの最終参照日付を現在のシステム時間に更新する( )か、更新しない( )かを指定します。</li> </ul>

## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi.ChatterFeeds クラス

フィードおよびフィードデータにアクセスし、変更します。この操作には、フィード項目、いいね!、コメント、ブックマークのアクセス、追加、および削除が含まれます。また、このクラスを使用してフィード項目を検索することもできます。

プログラムで返されるフィードのコンテンツが、Chatter ユーザインターフェースで返されるコンテンツと照合されます。

フィード項目のメッセージセグメントは、  
型です。フィード項目添付ファイルは  
型です。これらはどちらも抽象クラスで、複数の具象サブクラスがあります。実行時、  
オブジェクトと  
を使用すると、これらのオブジェクトの具象型をチェックして、対応するダウンキャストを確実に実行することができます。ダウンキャスト時には、不明なサブクラスを処理するデフォルトの case が必要です。

 重要: フィードの構成は、リリースによって異なる場合があります。コードでは、常に  
オブジェクトと  
オブジェクトの両方の不明なサブクラスを処理できるように準備しておく必要があります。

### 静的メソッド

API バージョン 28.0 では、すべてのメソッドを使用できます。

次に、クラスの静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>commentId</i>	Void	<p>指定されたコメントを削除します。ニュースフィードやレコードフィードなど、任意のフィードでコメント ID を検索できます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>commentId</i> — コメントの ID。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i>	Void	<p>指定されたフィード項目を削除します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> </ul>
	String <i>communityId</i> , String <i>likeId</i>	Void	<p>指定されたいいね!を削除します。コメントまたはフィード項目のいいね!を指定できます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>likeId</i> — いいね! の ID。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>commentId</i>	クラス	<p>指定されたコメントを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>commentId</i> — コメントの ID。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i>	クラス	<p>指定されたフィード項目へのコメントの最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i> , String <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>指定されたフィード項目へのコメントの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> ,  Enum <i>feedType</i>	クラス	<p>ク 指定されたフィード種別のフィードに関する情報を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は のみです。</li> </ul>
	String <i>communityId</i> ,  Enum <i>feedType</i> ,	クラス	<p>ク 指定されたフィード種別のフィードを、指定された順序で返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> </ul>

名前	引数	戻り値	説明
	Enum <i>sortParam</i>		<ul style="list-style-type: none"> <li>• <i>feedType</i> — フィードの種別。有効な値はのみです。</li> <li>• <i>sortParam</i> — 値は次のとおりです。           <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i>	ラス	<p>ク 指定されたユーザの指定されたフィード種別のフィードを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は、を除くすべてのです。</li> <li>• <i>subjectId</i> — <i>feedType</i> がである場合、<i>subjectId</i>には任意のユーザ ID を使用できます。<i>feedType</i> がである場合、<i>subjectId</i>はトピック ID である必要があります。<i>feedType</i> がその他の値の場合、<i>subjectId</i>はコンテキストユーザの ID またはキーワードである必要があります。</li> </ul>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i> , Enum <i>sortParam</i>	ラス	<p>ク 指定されたユーザの指定されたフィード種別のフィードを、指定された順序で返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は、を除くすべてのです。</li> <li>• <i>subjectId</i> — <i>feedType</i> がである場合、<i>subjectId</i>には任意のユーザ ID を使用できます。<i>feedType</i> がである場合、<i>subjectId</i>はトピック ID である必要があります。<i>feedType</i> がその他の値の場合、<i>subjectId</i>はコン</li> </ul>

名前	引数	戻り値	説明
			<p>テキストユーザの ID またはキーワードである必要があります。</p> <ul style="list-style-type: none"> <li>• <i>sortParam</i> — 値は次のとおりです。           <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。渡すと、デフォルトのが使用されます。</p>
	String <i>communityId</i> , String <i>feedItemId</i>	クラス	<p>指定されたフィード項目の詳細な説明を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> </ul>
	String <i>communityId</i> , Enum <i>feedType</i>	クラス	<p>指定されたフィード種別のフィード項目の最初のページを返します。一致するメソッドを使用したテスト時に呼び出された場合は、そのメソッドで渡されたフィード項目ページを返します。test メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。ページには、デフォルトの項目数が含まれます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値はのみです。</li> </ul>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>pageParam</i> , integer <i>pageSize</i>	クラス	<p>指定されたフィード種別の指定されたページのフィード項目を、指定された順序で返します。一致するメソッドを使用したテスト時に呼び出された場合は、そのメソッドで渡されたフィード項目ページを返します。test メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p>

名前	引数	戻り値	説明
		Enum <i>sortParam</i>	<ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は のみです。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i> — 値は次のとおりです。           <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。を渡すと、デフォルトのが使用されます。</p>
	String <i>communityId</i> ,  <b>クラス</b>  Enum <i>feedType</i> , String <i>subjectId</i>		<p>指定されたフィード種別のフィード項目の最初のページを、指定された順序で返します。一致するメソッドを使用したテスト時に呼び出された場合は、そのメソッドで渡されたフィード項目ページを返します。test メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。ページには、デフォルトの項目数が含まれます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は、 を除くすべてのです。</li> <li>• <i>subjectId</i> — <i>feedType</i> がである場合、<i>subjectId</i> には任意のユーザ ID を使用できます。<i>feedType</i> が</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i> , String <i>pageParam</i> , integer <i>pageSize</i> ,  Enum <i>sortParam</i>	クラス	<p>である場合、<i>subjectId</i>はトピック ID である必要があります。<i>feedType</i> がその他の値の場合、<i>subjectId</i>はコンテキストユーザの ID またはキーワードである必要があります。</p> <p>指定されたユーザの指定されたページの、指定されたフィード種別のフィード項目を、指定された順序で返します。一致するメソッドを使用したテスト時に呼び出された場合は、そのメソッドで渡されたフィード項目ページを返します。test メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は、 を除くすべてのです。</li> <li>• <i>subjectId</i> — <i>feedType</i> がである場合、<i>subjectId</i>には任意のユーザ ID を使用できます。<i>feedType</i> がである場合、<i>subjectId</i>はトピック ID である必要があります。<i>feedType</i> がその他の値の場合、<i>subjectId</i>はコンテキストユーザの ID またはキーワードである必要があります。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i> — 値は次のとおりです。 <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul>

名前	引数	戻り値	説明
			<p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。渡すと、デフォルトのが使用されます。</p>
	String <i>communityId</i> , String <i>subjectId</i> String <i>keyPrefix</i>	クラス	<p>指定されたユーザおよび指定されたキープレフィックスのフィード項目の最初のページを返します。一致する set test メソッドを使用したテスト時に呼び出された場合は、そのメソッドで渡され、キープレフィックスで絞り込まれたフィード項目ページを返します。指定された種別に関連付けられたフィード項目のみが返されます。test メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>subjectId</i> — コンテキストユーザの ID またはキーワード。</li> <li>• <i>keyPrefix</i> — 目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> String <i>keyPrefix</i> String <i>pageParam</i> , integer <i>pageSize</i> ,  Enum <i>sortParam</i>	クラス	<p>指定されたユーザおよび指定されたキープレフィックスのフィード項目の指定されたページを、指定された順序で返します。一致する set test メソッドを使用したテスト時に呼び出された場合は、そのメソッドで渡され、キープレフィックスで絞り込まれたフィード項目ページを返します。指定された種別に関連付けられたフィード項目のみが返されます。test メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>subjectId</i> — コンテキストユーザの ID またはキーワード。</li> <li>• <i>keyPrefix</i> — 目的のレコードタイプの キープレフィックスを指定します。キープレフィックスは、オブジェクト ID に 含まれる 3 文字のプレフィックスコード です。オブジェクト ID はオブジェクト 型を示す 3 文字のコードが先頭に付けら れます。たとえば、User オブジェクトの プレフィックスは 005、Group オブジエ クトのプレフィックスは 0F9 です。</li> <li>• <i>pageParam</i> — ページの表示に使用される ページトークンを指定します。ページ トークンは、 のように、応答クラス の一部として返されます。 を渡す と、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指 定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i> — 値は次のとおりです。 <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> <p>最近作成されたフィード項目、または最 近変更されたフィード項目ごとに、返さ れたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p> </li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i>	クラス	<p>指定されたフィード項目に関連付けられた アンケートを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>subjectId</i> String <i>keyPrefix</i>	ラス	<p>ク 指定されたユーザおよび特定のキープレフィックスのフィードの最初のページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>subjectId</i>— コンテキストユーザの ID またはキーワード。</li> <li>• <i>keyPrefix</i>— 目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> String <i>keyPrefix</i>  Enum <i>sortParam</i>	ラス	<p>ク 指定されたユーザおよび特定のキープレフィックスのフィードの最初のページを指定された順序で返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>subjectId</i>— コンテキストユーザの ID またはキーワード。</li> <li>• <i>keyPrefix</i>— 目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> <li>• <i>sortParam</i>— 値は次のとおりです。 <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p> </li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>likeId</i>		<p>指定されたいいね! を返します。</p> <p><b>クラス</b></p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>likeId</i> — いいね! の ID。</li> </ul>
	String <i>communityId</i> , String <i>commentId</i>	ラス	<p>指定されたコメントへのいいね! の最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>commentId</i> — コメントの ID。</li> </ul>
	String <i>communityId</i> , String <i>commentId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	ラス	<p>指定されたコメントへのいいね! の指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>commentId</i> — コメントの ID。</li> <li>• <i>pageParam</i> — 返すページの数を指定します。0 から開始します。または 0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i>	ラス	<p>指定されたフィード項目へのいいね! の最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	ラス	<p>指定されたフィード項目へのいいね! の指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> </ul>



名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>communityId — コミュニティの ID、 、または のいずれかを使用します。</li> <li>feedItemId — フィード項目の ID。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i> , String <i>text</i>	クラス	コンテキストユーザの指定されたフィード項目へのコメントとして、指定されたテキストを追加します。 <i>text</i> でハッシュタグまたはリンクが検出された場合は、ハッシュタグセグメントまたはリンクセグメントとしてコメントに含まれます。メンションは、 <i>text</i> では検出されず、テキストから分離されることもありません。 <ul style="list-style-type: none"> <li>communityId — コミュニティの ID、 、または のいずれかを使用します。</li> <li>feedItemId — フィード項目の ID。</li> <li><i>text</i> — 投稿のテキストを指定します。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i> ,  <b>クラス</b> <i>comment</i> ,  <b>クラス</b> <i>feedItemFileUpload</i>	クラス	コンテキストユーザの指定されたフィード項目にコメントを追加します。このメソッドは、@メンションなどのリッチテキストを使用してコメントにファイルを添付するために使用します。 <a href="#">サンプル: ファイルを添付したコメントの投稿</a> を参照してください。 <ul style="list-style-type: none"> <li>communityId — コミュニティの ID、 、または のいずれかを使用します。</li> <li>feedItemId — フィード項目の ID。</li> <li><i>comment</i> — オブジェクトで、@メンションなどのリッチテキストを指定します。必要に応じて、 プロパティで、既存または新規のファイルを指定します。</li> <li><i>feedItemFileUpload</i> — プロパティで オブジェクトを指定する場合は、添付する新規のバイナリファイルをこの引数で指定します。それ以外の場合は、値を指定しません。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i> , String <i>text</i>	クラス	<p>コンテキストユーザの指定されたフィードにフィード項目を追加します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — 次のいずれかになります。</li> </ul> <p>◊</p> <p>◊</p> <p>◊</p> <p>グループに投稿するには、 を使用します。</p> <ul style="list-style-type: none"> <li>• <i>subjectId</i> — 値は、 <i>feedType</i> によって異なります。 <ul style="list-style-type: none"> <li>◊ — <i>subjectId</i> は、コンテキストユーザの ID またはキーワード である必要があります。</li> <li>◊ — グループを含むフィードの任意のレコードの ID。</li> <li>◊ — 任意のユーザの ID。</li> </ul> </li> <li>• <i>text</i> — コメントのテキスト。メンションとハッシュタグは削除されます。</li> </ul>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i> ,  クラス <i>feedItemInput</i> ,  クラス <i>feedItemFileUpload</i>	クラス	<p>コンテキストユーザの指定されたフィードにフィード項目を追加します。このメソッドは、@メンションやハッシュタグなどのリッチテキストを投稿してフィード項目にファイルを添付するために使用します。また、このメソッドを使用して、フィード項目の共有およびコメントの追加を行うこともできます。サンプル: <a href="#">フィード項目の共有とコメントの追加</a>を参照してください。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — 次のいずれかになります。</li> </ul> <p>◊</p> <p>◊</p> <p>◊</p> <p>フィード項目をグループに投稿するには、 を使用し、 <i>subjectId</i> にグループ ID を使用します。</p>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>subjectId</i> — <i>feedType</i> が である場合、<i>subjectId</i>には任意のユー ザ ID を使用できます。<i>feedType</i> が である場合、<i>subjectId</i>はトピッ ク ID である必要があります。<i>feedType</i> がその他の値の場合、<i>subjectId</i>はコン テキストユーザの ID またはキーワード である必要があります。</li> <li>• <i>feedItemInput</i> — オブ ジェクトで、リッチテキストを指定しま す。必要に応じて、 プロパティ で、リンク、アンケート、既存または新 規のファイルを指定します。</li> <li>• <i>feedItemFileUpload</i> — プロパティ で オブジェ クトを指定する場合は、添付する新規の バイナリファイルをこの引数で指定しま す。それ以外の場合は、値を指定しませ ん。</li> </ul>
	String <i>communityId</i> , String <i>q</i>	クラス	<p>指定された検索条件と一致するすべての フィード項目の最初のページを返します。 このページにはデフォルトの項目数が含ま れています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使 用します。</li> <li>• <i>q</i> — 必須項目であり、 は無効です。 検索する文字列を指定します。検索文字 列にはワイルドカード文字を除いて 2 文 字以上が含まれている必要があります。 <a href="#">「ワイルドカードの使用」</a>を参照してく ださい。</li> </ul>
	String <i>communityId</i> , String <i>q</i> , Enum <i>sortParam</i>	クラス	<p>指定された検索条件と一致するすべての フィード項目の最初のページを返します。 このページにはデフォルトの項目数が含ま れています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使 用します。</li> </ul>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>q</i>—必須項目であり、　　は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>• <i>sortParam</i>— 値は次のとおりです。           <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。を渡すと、デフォルトのが使用されます。</p>
	String <i>communityId</i> , String <i>q</i> , String <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>指定された検索条件と一致し、コンテキストユーザが参照できるすべてのフィード項目のリストを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、　　、または　　のいずれかを使用します。</li> <li>• <i>q</i>—必須項目であり、　　は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>• <i>pageParam</i>—ページの表示に使用されるページトークンを指定します。ページトークンは、　　または　　のように、応答クラスの一部として返されます。　　を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i>—ページあたりの項目数を指定します。有効な値は1～100です。　　を渡すと、デフォルツサイズの25に設定されます。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>q</i> , String <i>pageParam</i> , integer <i>pageSize</i> ,  Enum <i>sortParam</i>	クラス	<p>指定された検索条件と一致し、コンテキストユーザーが参照できるすべてのフィード項目のリストを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>q</i>—必須項目であり、 は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>• <i>pageParam</i>—ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i>—ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i>—値は次のとおりです。</li> </ul> <p style="text-align: center;">◊ ◊</p> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>q</i>	クラス	<p>指定されたフィード種別のフィード項目を検索します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i>—フィードの種別。有効な値は のみです。</li> <li>• <i>q</i>—必須項目であり、 は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。</li> </ul>

名前	引数	戻り値	説明
			<p>「ワイルドカードの使用」を参照してください。</p>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>pageParam</i> , integer <i>pageSize</i> ,  Enum <i>sortParam</i> , String <i>q</i>	クラス	<p>指定されたフィード種別のフィード項目を検索し、指定されたページおよびページサイズを指定された並び替え順で返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は のみです。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i> — 値は次のとおりです。 <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p> <ul style="list-style-type: none"> <li>• <i>q</i> — 必須項目であり、 は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。 「ワイルドカードの使用」を参照してください。</li> </ul>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i>	クラス	<p>指定されたフィード種別およびコンテキストユーザーのフィード項目を検索します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> </ul>

名前	引数	戻り値	説明
	String <i>q</i>  String <i>communityId</i> ,  Enum <i>feedType</i> ,  String <i>subjectId</i> ,  String <i>pageParam</i> ,  integer <i>pageSize</i> ,   Enum <i>sortParam</i> ,  String <i>q</i>	クラス	<ul style="list-style-type: none"> <li>• <i>feedType</i> — フィードの種別。有効な値は、<b>と</b> を除くすべてのです。</li> <li>• <i>subjectId</i> — <i>feedType</i> がである場合、<i>subjectId</i> には任意のユーザ ID を使用できます。<i>feedType</i> がその他の値の場合、<i>subjectId</i> はコンテキストユーザの ID またはキーワード である必要があります。</li> <li>• <i>q</i> — 必須項目であり、<b>は無効です。</b> 検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。 <a href="#">「ワイルドカードの使用」</a> を参照してください。</li> </ul> <p>指定されたフィード種別およびコンテキストユーザのフィード項目を検索し、指定されたページおよびページサイズを指定された並び替え順で返します。</p> <ul style="list-style-type: none"> <li>• <i>feedType</i> — フィードの種別。有効な値は、<b>と</b> を除くすべてのです。</li> <li>• <i>subjectId</i> — <i>feedType</i> がである場合、<i>subjectId</i> には任意のユーザ ID を使用できます。<i>feedType</i> がその他の値の場合、<i>subjectId</i> はコンテキストユーザの ID またはキーワード である必要があります。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、<b>または</b> のように、応答クラスの一部として返されます。<b>を渡すと、最初のページが返されます。</b></li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。<b>を渡すと、デフォルトサイズの 25 に設定されます。</b></li> <li>• <i>sortParam</i> — 値は次のとおりです。 <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul>

名前	引数	戻り値	説明
			<p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。渡すと、デフォルトのが使用されます。</p> <ul style="list-style-type: none"> <li>—必須項目であり、は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> , String <i>keyPrefix</i> , String <i>q</i>	クラス	<p>キープレフィックスで絞り込まれたフィードのフィード項目を検索します。</p> <ul style="list-style-type: none"> <li><i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li><i>subjectId</i> — コンテキストユーザの ID またはキーワード。</li> <li><i>keyPrefix</i> — 目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる3文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す3文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> <li>—必須項目であり、は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> </ul>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i> , String <i>keyPrefix</i> , String <i>pageParam</i>	クラス	<p>キープレフィックスで絞り込まれたフィードのフィード項目を検索し、指定されたページおよびページサイズを指定された並び替え順で返します。</p> <ul style="list-style-type: none"> <li><i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li><i>subjectId</i> — コンテキストユーザの ID またはキーワード。</li> </ul>

名前	引数	戻り値	説明
	integer <i>pageSize</i> , Enum <i>s</i>	der ©ñ36.303f30303e3ñDUGHU	<ul style="list-style-type: none"><li>• <i>keyPrefix</i> — 目的のレコードタイプのキー・プレフィックスを指定します。キー・プレフィックスは、目的のレコードタイプの名前を表す DER 形式の文字列です。</li></ul>

名前	引数	戻り値	説明
	<pre>String communityId, Void Enum feedType, String pageParam, integer pageSize, Enum sortParam クラス result</pre>		<ul style="list-style-type: none"> <li>communityId — コミュニティの ID、または のいずれかを使用します。</li> <li>feedType — フィードの種別。有効な値は のみです。</li> <li>result — テストフィード項目ページを指定します。</li> </ul> <p>一致するパラメータを使用してテスト中にがコールされたときに返される、テストフィード項目ページを登録します。getFeed メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>communityId — コミュニティの ID、または のいずれかを使用します。</li> <li>feedType — フィードの種別。有効な値は のみです。</li> <li>pageParam — ページの表示に使用されるページトークンを指定します。ページトークンは、または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>pageSize — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>sortParam — 値は次のとおりです。 <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p> </li> <li>result — テストフィード項目ページを指定します。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , Void  Enum <i>feedType</i> ,  String <i>subjectId</i> ,  <span style="color: blue;">クラス</span> <i>result</i>		<p>一致するパラメータを使用してテスト中に がコールされたとき に返される、テストフィード項目ページ を登録します。get feed メソッドでは、必ず 同じパラメータを使用する必要があります。 パラメータが同じでないと、例外が発生し ます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使 用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値 は、 を除くすべての です。</li> <li>• <i>subjectId</i> — <i>feedType</i> が である場合、<i>subjectId</i> には任意のユー ザ ID を使用できます。<i>feedType</i> が である場合、<i>subjectId</i> はトピッ ク ID である必要があります。<i>feedType</i> がその他の値の場合、<i>subjectId</i> はコン テキストユーザの ID またはキーワード である必要があります。</li> <li>• <i>result</i> — テストフィード項目ページを 指定します。</li> </ul>
	String <i>communityId</i> , Void  Enum <i>feedType</i> ,  String <i>subjectId</i> String <i>pageParam</i> , integer <i>pageSize</i> ,  Enum <i>sortParam</i>  <span style="color: blue;">クラス</span> <i>result</i>		<p>一致するパラメータを使用してテスト中に がコールされたとき に返される、テストフィード項目ページ を登録します。get feed メソッドでは、必ず 同じパラメータを使用する必要があります。 パラメータが同じでないと、例外が発生し ます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使 用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値 は、 を除くすべての です。</li> <li>• <i>subjectId</i> — <i>feedType</i> が である場合、<i>subjectId</i> には任意のユー ザ ID を使用できます。<i>feedType</i> が である場合、<i>subjectId</i> はトピッ ク ID である必要があります。<i>feedType</i> がその他の値の場合、<i>subjectId</i> はコン</li> </ul>

名前	引数	戻り値	説明
			<p>テキストユーザの ID またはキーワードである必要があります。</p> <ul style="list-style-type: none"> <li>• <i>pageParam</i>—ページの表示に使用されるページトークンを指定します。ページトークンは、またはのように、応答クラスの一部として返されます。を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i>—ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i>—値は次のとおりです。           <ul style="list-style-type: none"> <li>◊</li> <li>◊</li> </ul> </li> </ul> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。を渡すと、デフォルトのが使用されます。</p> <ul style="list-style-type: none"> <li>• <i>result</i>—テストフィード項目ページを指定します。</li> </ul>
	String <i>communityId</i> , Void String <i>subjectId</i> String <i>keyPrefix</i>  <b>クラス</b> <i>result</i>		<p>テストに使用するフィード項目ページを作成します。ページの作成後、一致するメソッドを使用してテストページにアクセスし、テストを実行します。get feed filter メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>subjectId</i>—コンテキストユーザの ID またはキーワード。</li> <li>• <i>keyPrefix</i>—目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトの</li> </ul>

名前	引数	戻り値	説明
			<p>プレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</p> <ul style="list-style-type: none"> <li>• <i>result</i> — テストフィード項目ページを指定します。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> String <i>keyPrefix</i> String <i>pageParam</i> , integer <i>pageSize</i> ,  Enum <i>sortParam</i>		<p>テストに使用するフィード項目ページを作成します。ページの作成後、一致するメソッドを使用してテストページにアクセスし、テストを実行します。get feed filter メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>subjectId</i> — コンテキストユーザの ID またはキーワード。</li> <li>• <i>keyPrefix</i> — 目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i> — 値は次のとおりです。 ◊ ◊</li> </ul>
			最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。

名前	引数	戻り値	説明
			<p>を渡すと、デフォルトの が使用されます。</p> <ul style="list-style-type: none"> <li>• <i>result</i> — テストフィード項目ページを 指定します。</li> </ul>
	String <i>communityId</i> , Void  Enum <i>feedType</i> ,  String <i>q</i> ,  <b>クラス</b> <i>result</i>		<p>テストに使用するフィード項目ページを作成します。ページの作成後、一致する メソッドを使用してテストページにアクセスし、テストを実行します。メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は のみです。</li> <li>• <i>q</i> — 必須項目であり、 は無効です。 検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。 <a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>• <i>result</i> — テストフィード項目ページを指定します。</li> </ul>
	String <i>communityId</i> , Void  Enum <i>feedType</i> ,  String <i>pageParam</i> , integer <i>pageSize</i> ,  <i>sortParam</i> , String <i>q</i> ,  <b>クラス</b> <i>result</i>		<p>テストに使用するフィード項目ページを作成します。ページの作成後、一致する メソッドを使用してテストページにアクセスし、テストを実行します。メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は、 を除くすべての です。</li> <li>• <i>pageParam</i> — ページの表示に使用される ページトークンを指定します。ページトークンは、 または</li> </ul>

名前	引数	戻り値	説明
			<p>のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</p> <ul style="list-style-type: none"> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i> — 値は次のとおりです。</li> </ul> <p>◊ ◊</p> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。 を渡すと、デフォルトの が使用されます。</p> <ul style="list-style-type: none"> <li>• <i>q</i> — 必須項目であり、 は無効です。 検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。 「ワイルドカードの使用」を参照してください。</li> <li>• <i>result</i> — テストフィード項目ページを指定します。</li> </ul>
	String <i>communityId</i> , Void  Enum <i>feedType</i> , String <i>subjectId</i> , String <i>q</i> ,  <span style="color: blue;">クラス</span> <i>result</i>		<p>テストに使用するフィード項目ページを作成します。ページの作成後、一致する メソッドを使用してテストページにアクセスし、テストを実行します。メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。コンテキストユーザの指定されたフィード種別で、フィード項目を検索します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、 、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は、 と を除くすべての です。</li> <li>• <i>subjectId</i> — <i>feedType</i> が である場合、<i>subjectId</i> には任意のユーザ ID を使用できます。<i>feedType</i> がその</li> </ul>

名前	引数	戻り値	説明
			<p>他の値の場合、<i>subjectId</i>はコンテキストユーザのIDまたはキーワードである必要があります。</p> <ul style="list-style-type: none"> <li>• <i>q</i>—必須項目であり、<i>q</i>は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>• <i>result</i>—テストフィード項目ページを指定します。</li> </ul> <p>String <i>communityId</i>, Void</p> <p>Enum <i>feedType</i>,</p> <p>String <i>subjectId</i>,</p> <p>String <i>pageParam</i>,</p> <p>integer <i>pageSize</i>,</p> <p><i>sortParam</i>,</p> <p>String <i>q</i>,</p> <p>クラス <i>result</i></p> <p>テストに使用するフィード項目ページを作成します。ページの作成後、一致するメソッドを使用してテストページにアクセスし、テストを実行します。メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティのID、<i>communityId</i>、または<i>communityId</i>のいずれかを使用します。</li> <li>• <i>feedType</i>—フィードの種別。有効な値は、<i>feedType</i>と<i>feedType</i>を除くすべての値です。</li> <li>• <i>subjectId</i>—<i>feedType</i>が<i>feedType</i>である場合、<i>subjectId</i>には任意のユーザIDを使用できます。<i>feedType</i>がその他の値の場合、<i>subjectId</i>はコンテキストユーザのIDまたはキーワードである必要があります。</li> <li>• <i>pageParam</i>—ページの表示に使用されるページトークンを指定します。ページトークンは、<i>pageParam</i>または<i>pageParam</i>のように、応答クラスの一部として返されます。<i>pageParam</i>を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i>—ページあたりの項目数を指定します。有効な値は1～100です。<i>pageSize</i>を渡すと、デフォルトサイズの25に設定されます。</li> <li>• <i>sortParam</i>—値は次のとおりです。</li> </ul>

名前	引数	戻り値	説明
		◊	<p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返されたフィードがソートされます。を渡すと、デフォルトのが使用されます。</p> <ul style="list-style-type: none"> <li>—必須項目であり、は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>—テストフィード項目ページを指定します。</li> </ul>
	String <i>communityId</i> , String <i>subjectId</i> , String <i>keyPrefix</i> , String <i>q</i> ,  <span style="color: blue;">クラス</span> <i>result</i>	Void	<p>テストに使用するフィード項目ページを作成します。ページの作成後、一致するメソッドを使用してテストページにアクセスし、テストを実行します。メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>—コミュニティのID、または のいずれかを使用します。</li> <li>—コンテキストユーザのID またはキーワード。</li> <li>—目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクトIDに含まれる3文字のプレフィックスコードです。オブジェクトIDはオブジェクト型を示す3文字のコードが先頭に付けられます。たとえば、Userオブジェクトのプレフィックスは005、Groupオブジェクトのプレフィックスは0F9です。</li> <li>—必須項目であり、は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> </ul>

名前	引数	戻り値	説明
	<p>String <i>communityId</i>, Void</p> <p>Enum <i>feedType</i>,</p> <p>String <i>subjectId</i>,</p> <p>String <i>keyPrefix</i>,</p> <p>String <i>pageParam</i>,</p> <p>integer <i>pageSize</i>,</p> <p>Enum <i>sortParam</i>,</p> <p>String <i>q</i>,</p> <p>クラス <i>result</i></p>		<ul style="list-style-type: none"> <li>• <i>result</i> — テストフィード項目ページを指定します。</li> </ul> <p>テストに使用するフィード項目ページを作成します。ページの作成後、一致するメソッドを使用してテストページにアクセスし、テストを実行します。メソッドでは、必ず同じパラメータを使用する必要があります。パラメータが同じでないと、例外が発生します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedType</i> — フィードの種別。有効な値は、 を除くすべてのです。</li> <li>• <i>subjectId</i> — コンテキストユーザの ID またはキーワード。</li> <li>• <i>keyPrefix</i> — 目的のレコードタイプのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> <li>• <i>pageParam</i> — ページの表示に使用されるページトークンを指定します。ページトークンは、 または のように、応答クラスの一部として返されます。 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> <li>• <i>sortParam</i> — 値は次のとおりです。</li> </ul> <p style="text-align: center;">◊ ◊</p> <p>最近作成されたフィード項目、または最近変更されたフィード項目ごとに、返さ</p>

名前	引数	戻り値	説明
			<p>れたフィードがソートされます。を渡すと、デフォルトのが使用されます。</p> <ul style="list-style-type: none"> <li>—必須項目であり、は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて2文字以上が含まれている必要があります。<a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>—テストフィード項目ページを指定します。</li> </ul>
	String <i>communityId</i> , Enum <i>feedType</i> , String <i>subjectId</i> , String <i>originalFeedItemId</i>	クラス	<p><i>feedType</i> で指定されたフィードと <i>originalFeedItemId</i> を共有します。</p> <p><a href="#">サンプル: グループとのフィード項目の共有 (ページ 723)</a>を参照してください。</p> <ul style="list-style-type: none"> <li>—コミュニティの ID、または のいずれかを使用します。</li> <li>—次のいずれかになります。 <ul style="list-style-type: none"> <li>—<i>subjectId</i> は、コンテキストユーザの ID またはキーワード である必要があります。</li> <li>—<i>subjectId</i> には、グループ ID またはコンテキストユーザの ID (または ) を使用できます。</li> <li>—<i>subjectId</i> には任意のユーザ ID を使用できます。</li> </ul> </li> <li>—<i>originalFeedItemId</i>—共有するフィード項目の ID。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>feedItemId</i> , Boolean <i>isBookmarkedByCurrentUser</i>	クラス	<p>指定されたフィード項目にブックマークを付けるか、指定されたフィード項目からブックマークを削除します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — フィード項目の ID。</li> <li>• <i>isBookmarkedByCurrentUser</i> — を指定すると、現在のユーザのブックマークのリストにフィード項目が追加されます。ブックマークを削除するには、を指定します。</li> </ul>
	String <i>communityId</i> , String <i>feedItemId</i> , String <i>myChoiceId</i>	クラス	既存のフィードのアンケートに投票するか、投票を変更するために使用します。 <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>feedItemId</i> — アンケートに関連付けられているフィード項目を指定します。</li> <li>• <i>myChoiceId</i> — 投票するアンケートの項目の ID を指定します。</li> </ul>

### サンプル: グループとのフィード項目の共有

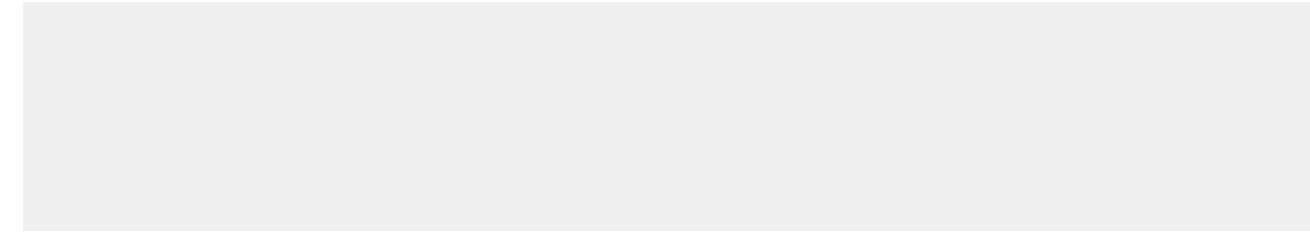
フィード項目をグループと共有するには、  
コミュニティ ID (または null)、フィード種別、グループ ID、共有するフィード項目の ID を渡します。

### サンプル: フィード項目の共有とコメントの追加

フィード項目を共有しコメントを追加するには、コメントおよび共有するフィード項目を含む  
オブジェクトを作成し、*feedItemInput* 引数で  
にそのオブジェクトを渡します。メッセージ本文に入力されたメッセージセグメントは、コメントとして使用されます。

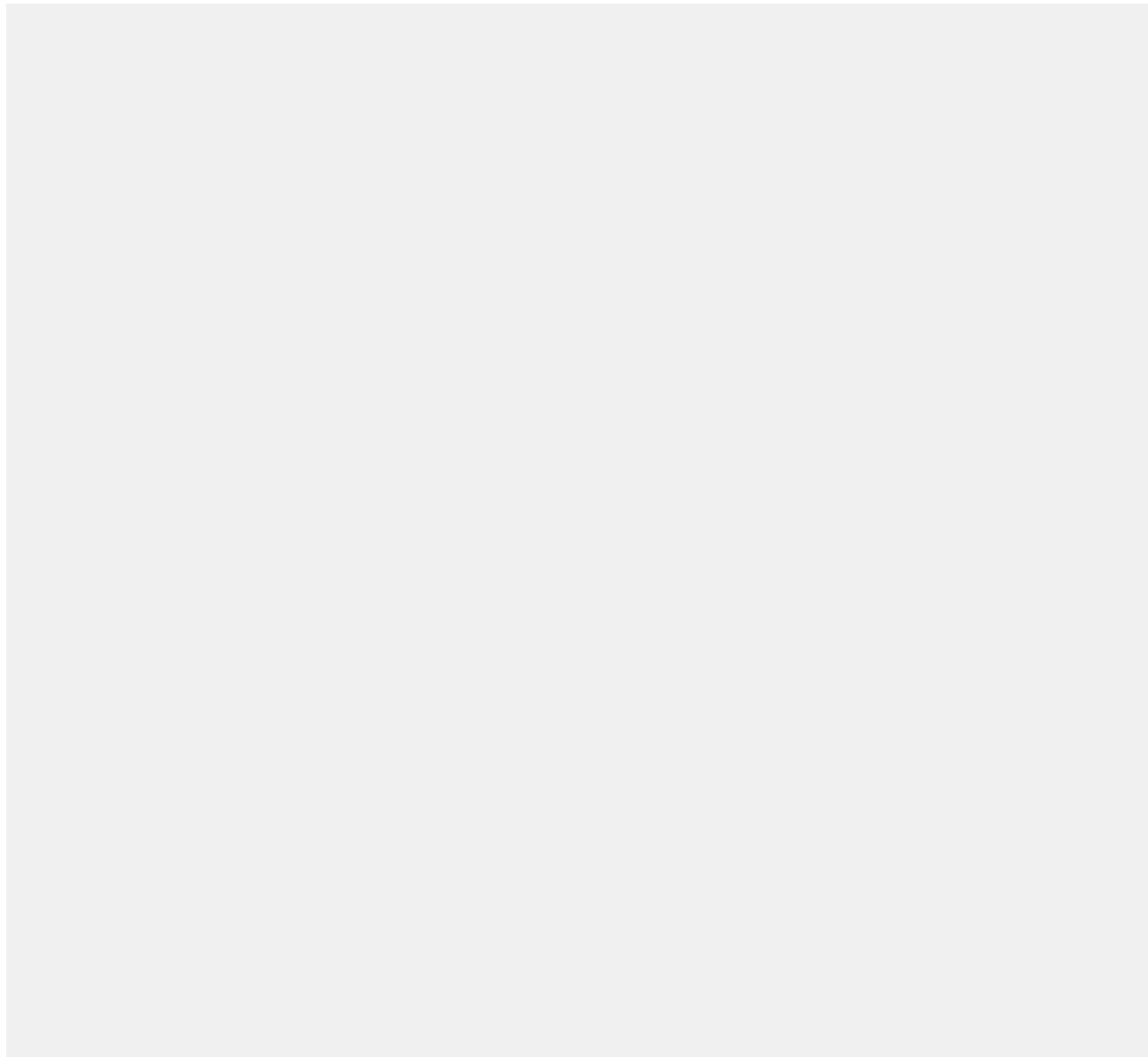
サンプル: ファイルを添付したコメントの投稿

コメントを投稿し、新しいファイルをアップロードしてコメントに添付するには、  
オブジェクトと オブジェクトを作成して  
メソッドに渡します。



サンプル: ユーザプロファイルフィードへの @メンションの投稿

ユーザプロファイルフィードに投稿し、@メンションを含めるには、  
メソッドをコールします。



## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi.ChatterGroups クラス

グループのメンバー、写真、および指定されたユーザがフォローしているグループなど、グループに関するアクセス情報。グループへのメンバーの追加やメンバーの削除、グループの写真の変更に使用します。

### メソッド

API バージョン 28.0 では、すべてのメソッドを使用できます。

次に、

クラスの静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>groupId</i> , String <i>userId</i>		<p>指定されたコミュニティ内の指定されたグループに、指定されたユーザを追加します。 このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"><li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li><li>• <i>groupId</i> — グループの ID。</li><li>• <i>userId</i> — ユーザの ID。</li></ul>
	String <i>communityId</i> , String <i>membershipId</i>	Void	<p>指定されたメンバーシップを削除します。 このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"><li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li><li>• <i>membershipId</i> — メンバーシップの ID。</li></ul>
	String <i>communityId</i> , String <i>groupId</i>	Void	<p>指定されたグループの写真を削除します。 このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「す</p>

名前	引数	戻り値	説明
			<p>「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>groupId</i> — グループの ID。</li> </ul>
	String <i>communityId</i> , String <i>groupId</i>	クラス	<p>指定されたグループに関する情報を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>groupId</i> — グループの ID。</li> </ul>
	String <i>communityId</i> , String <i>requestId</i>	クラス	<p>private グループへの指定された参加要求に関する情報を返します。</p> <p>このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>requestId</i> — private グループへの参加要求の ID。</li> </ul>
	String <i>communityId</i> , String <i>groupId</i>	クラス	<p>指定された private グループへのすべての参加要求に関する情報を返します。</p> <p>このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>groupId</i> — グループの ID。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>groupId</i> , Enum <i>status</i>	クラス	<p>指定された状況の、指定された private グループへのすべての参加要求に関する情報を返します。</p> <p>このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>groupId</i> — グループの ID。</li> <li>• <i>status</i> — private グループへの参加要求の状況。</li> </ul> <p style="text-align: center;">◊ ◊ ◊</p>
	String <i>communityId</i>	クラス	<p>すべてのグループの最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> </ul>
	String <i>communityId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>すべてのグループに関する情報の指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>pageParam</i> — 返すページの数。0 から開始します。または を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>membershipId</i>	クラス	<p>指定されたグループメンバーに関する情報を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>membershipId</i> — メンバーシップの ID。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>groupId</i>	<b>クラス</b> <b>Z</b>	<p>指定されたグループのすべてのメンバーに関する情報の最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または QXO のいずれかを使用します。</li> <li>• <i>groupId</i> — グループの ID。</li> </ul>
	String <i>communityId</i> , String <i>groupId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	<b>クラス</b> <b>Z</b>	<p>指定されたグループのすべてのメンバーに関する情報の指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または QXO のいずれかを使用します。</li> <li>• <i>groupId</i> — グループの ID。</li> <li>• <i>pageParam</i> — 返すページの数。0 から開始します。レゾルベーションにべの値</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>q</i>	クラス	<p>指定された検索条件と一致するグループの最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>q</i> — 検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。「ワイルドカードの使用」を参照してください。 を指定できます。</li> </ul>
	String <i>communityId</i> , String <i>q</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>指定された検索条件と一致するグループの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>q</i> — 検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。「ワイルドカードの使用」を参照してください。 を指定できます。</li> <li>• <i>pageParam</i> — 返すページの数を指定します。0 から開始します。 または 0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>groupId</i> , String <i>fileId</i> , Integer <i>versionNumber</i>	クラス	<p>グループの写真を、指定された、すでにアップロードされたファイルに設定します。キー プレフィックスは 069 に、ファイルサイズは 2 MB 未満にする必要があります。<a href="#">サンプル:既存ファイルを使用したグループ写真の更新</a> (ページ 733) を参照してください。</p> <p>このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>groupId</i> — グループの ID。</li> </ul>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>fileId</i>—すでにアップロードされたファイルの ID。ファイルは 2 MB 未満の画像にする必要があります。</li> <li>• <i>versionNumber</i>—既存ファイルのバージョン番号。既存のバージョン番号を指定するか、<a href="#">最新バージョン</a>を指定して最新バージョンを取得します。</li> </ul>
	String <i>communityId</i> , String <i>groupId</i> ,  <i>fileUpload</i>	クラス	<p>指定されたグループの写真として、提供された blob を設定します。<a href="#">サンプル: 新しいファイルをアップロードしてグループ写真として使用する</a> (ページ 734)。</p> <p>このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、<a href="#">コミュニティ</a>、または <a href="#">コミュニティ</a> のいずれかを使用します。</li> <li>• <i>groupId</i>—グループの ID。</li> <li>• <i>fileUpload</i>—グループの写真として使用する画像。画像として使用できるコンテンツタイプである必要があります。</li> </ul>
	String <i>communityId</i> , String <i>groupId</i> ,  <i>groupInput</i>	クラス	<p>グループの「情報」セクションを更新します。グループが非公開の場合は、このセクションはメンバーにのみ表示されます。</p> <p>このメソッドは、コンテキストユーザがグループ管理者または所有者であるか、「すべてのデータの編集」権限を持っている場合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、<a href="#">コミュニティ</a>、または <a href="#">コミュニティ</a> のいずれかを使用します。</li> <li>• <i>groupId</i>—グループの ID。</li> <li>• <i>groupParam</i>—「情報」セクションのテキストとタイトルを含むオブジェクト。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>groupId</i> , Enum <i>emailFrequency</i>	クラス	<p>指定されたグループのコンテキストユーザ ク の Chatter 設定を更新します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、 、または のいずれかを使 用します。</li> <li>• <i>groupId</i>— グループの ID。</li> <li>• <i>emailFrequency</i>— グループからユーザ がメールを受信する頻度を指定します。</li> </ul> <p>◊ ◊ ◊ ◊ ◊</p> <p>の値は、 へのコールの値セットを使用します。</p>
	String <i>communityId</i> , String <i>requestId</i> , Enum <i>status</i>	クラス	private グループへの参加要求を更新します。 サンプル: <a href="#">private グループへの参加要求の受 諾または拒否 (ページ 733)</a> を参照してください。 <p>このメソッドは、コンテキストユーザがグ ループ管理者または所有者であるか、「す べてのデータの編集」権限を持っている場 合にのみ正常に実行されます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、 、または のいずれかを使 用します。</li> <li>• <i>requestId</i>— private グループへの参加要 求の ID。</li> <li>• <i>status</i>— 要求の状況。</li> </ul> <p>◊ ◊</p> <p>このメソッドでは、 値に Enum は使用できません。</p>

### サンプル: private グループへの参加要求

このサンプルコードは  
への参加要求を行います。

をコールして private グループ

### サンプル: private グループへの参加要求の受諾または拒否

このサンプルコードは  
プ要求 ID と

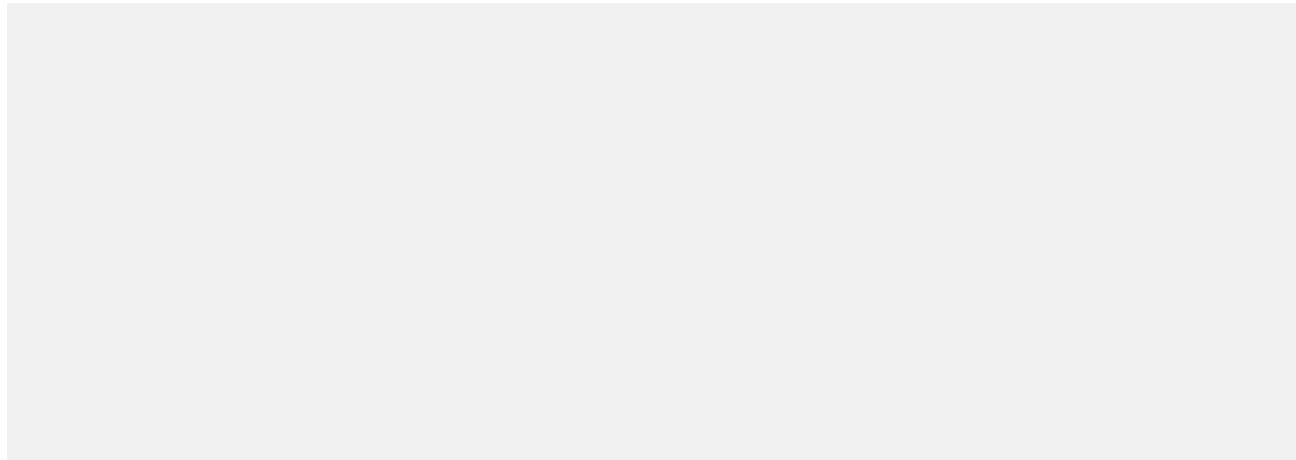
をコールし、それをメンバーシップ  
状況を渡します。また、  
を渡すこともできます。

### サンプル: 既存ファイルを使用したグループ写真の更新

グループを作成した時点では、グループ写真は含まれていません。Salesforce にすでにアップロードされている既存の写真をグループ写真として設定できます。キープレフィックスは 069 に、ファイルサイズは 2 MB 未満にする必要があります。

サンプル: 新しいファイルをアップロードしてグループ写真として使用する

グループを作成した時点では、グループ写真は含まれていません。写真をアップロードし、グループ写真としてそれを設定できます。



## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi.ChatterUsers クラス

ユーザをフォローしている人、ユーザの登録、ファイル、グループなど、ユーザに関する情報にアクセスします。

### メソッド

API バージョン 28.0 では、すべてのメソッドを使用できます。

次に、クラスの静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>communityId</i>	Void	指定されたユーザの写真を削除します。
	String <i>userId</i>		<ul style="list-style-type: none"><li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li><li>• <i>userId</i> — コンテキストユーザの ID またはキーワード 。</li></ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>userId</i> , String <i>subjectId</i>	ス	<p><b>クラ</b> 指定された <i>userId</i> をフォロワーとして、指定された <i>subjectId</i> に追加します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>userId</i> — コンテキストユーザの ID またはキーワード。</li> <li>• <i>subjectId</i> — フォローする件名の ID。</li> </ul>
	String <i>communityId</i> , String <i>userId</i>	ラス	<p><b>クラ</b> 指定されたユーザのデフォルトの Chatter 設定に関する情報を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>userId</i> — コンテキストユーザの ID またはキーワード。</li> </ul>
	String <i>communityId</i> , String <i>userId</i>	ス	<p><b>クラ</b> 指定されたユーザID のフォロワーの最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>userId</i> — ユーザの ID。</li> </ul>
	String <i>communityId</i> , String <i>userId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	ス	<p><b>クラ</b> 指定されたユーザID のフォロワーの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、またはのいずれかを使用します。</li> <li>• <i>userId</i> — ユーザの ID。</li> <li>• <i>pageParam</i> — 返すページの数を指定します。0 から開始します。または0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>userId</i>	ス	<p><b>クラ</b> 指定されたユーザのフォロワーに関する情報の最初のページを返します。このページにはデフォルトの項目数が含まれています。これは、指定されたユーザをフォローして</p>

名前	引数	戻り値	説明
			いるユーザを返す とは異なります。 <ul style="list-style-type: none"> <li>communityId — コミュニティの ID、 、または のいずれかを使用します。</li> <li>userId — ユーザの ID。</li> </ul>
	String <i>communityId</i> , String <i>userId</i> , Integer <i>pageParam</i>	ス	指定されたユーザのフォロワーに関する情報の指定されたページを返します。このページにはデフォルトの項目数が含まれています。これは、指定されたユーザをフォローしているユーザを返す とは異なります。 <ul style="list-style-type: none"> <li>communityId — コミュニティの ID、 、または のいずれかを使用します。</li> <li>userId — ユーザの ID。</li> <li>pageParam — 返すページの数を指定します。0 から開始します。 または0 を渡すと、最初のページが返されます。</li> </ul>
	String <i>communityId</i> , String <i>userId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	ス	指定されたユーザのフォロワーに関する情報の特定のページを返します。これは、指定されたユーザをフォローしているユーザを返す とは異なります。 <ul style="list-style-type: none"> <li>communityId — コミュニティの ID、 、または のいずれかを使用します。</li> <li>userId — ユーザの ID。</li> <li>pageParam — 返すページの数を指定します。0 から開始します。 または0 を渡すと、最初のページが返されます。</li> <li>pageSize — ページあたりの項目数を指定します。有効な値は 1 ~ 1000 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>userId</i> , String <i>filterType</i>	ス	指定されたユーザの指定された種別のフォロワーに関する情報の最初のページを返します。このページにはデフォルトの項目数が含まれています。これは、指定されたユーザをフォローしているユーザを返す とは異なります。

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>userId</i> — ユーザの ID。</li> <li>• <i>filterType</i> — 返されるオブジェクトの種別を絞り込むためのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> </ul>
	String <i>communityId</i> , String <i>userId</i> , String <i>filterType</i> , Integer <i>pageParam</i>	ス	<p><b>クラス</b> 指定されたユーザの指定された種別のフォローに関する情報の指定されたページを返します。このページにはデフォルトの項目数が含まれています。これは、指定されたユーザをフォローしているユーザを返すとは異なります。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>userId</i> — ユーザの ID。</li> <li>• <i>filterType</i> — 返されるオブジェクトの種別を絞り込むためのキープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> <li>• <i>pageParam</i> — 返すページの数を指定します。0 から開始します。または 0 を渡すと、最初のページが返されます。</li> </ul>
	String <i>communityId</i> , String <i>userId</i> , String <i>filterType</i> ,	ス	<p><b>クラス</b> 指定されたユーザの指定された種別のフォローに関する情報の指定されたページを返します。これは、指定されたユーザをフォローしているユーザを返すとは異なります。</p>

名前	引数	戻り値	説明
	integer <i>pageParam</i> , Integer <i>pageSize</i>		<ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>userId</i> — ユーザの ID。</li> <li>• <i>filterType</i> — 返されるオブジェクトの種別を絞り込むためのキープレフィックスを指定します。キープレフィックスは、オブジェクト ID に含まれる 3 文字のプレフィックスコードです。オブジェクト ID はオブジェクト型を示す 3 文字のコードが先頭に付けられます。たとえば、User オブジェクトのプレフィックスは 005、Group オブジェクトのプレフィックスは 0F9 です。</li> <li>• <i>pageParam</i> — 返すページの数を指定します。0 から開始します。または 0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 1000 です。を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>userId</i>	クラス	<p>指定されたユーザがメンバーであるグループの最初のページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>userId</i> — ユーザの ID。</li> </ul>
	String <i>communityId</i> , String <i>userId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>指定されたユーザがメンバーであるグループの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>userId</i> — ユーザの ID。</li> <li>• <i>pageParam</i> — 返すページの数を指定します。0 から開始します。または 0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>

名前	引数	戻り値	説明
	String <i>communityId</i> , String <i>userId</i>	クラス	<p>指定されたユーザの写真に関する情報を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>userId</i>— ユーザの ID。</li> </ul>
	String <i>communityId</i> , String <i>userId</i>	クラス	<p>指定されたユーザに関する情報を返します。コンテキストユーザが Chatter 顧客でない場合は、返されたオブジェクトを <a href="#">UserDetail</a> オブジェクトにダウンキャストできます。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>userId</i>— ユーザの ID。</li> </ul>
	String <i>communityId</i>	クラス	<p>ユーザの最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、または のいずれかを使用します。</li> </ul>
	String <i>communityId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>ユーザの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>pageParam</i>— 返すページの数を指定します。0 から開始します。または 0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i>— ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>q</i>	クラス	<p>指定された検索条件と一致するユーザの最初のページを返します。このページにはデフォルトの項目数が含まれています。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>— コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>q</i>— 必須項目であり、 は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文</li> </ul>

名前	引数	戻り値	説明
			<p>字以上が含まれている必要があります。  <a href="#">「ワイルドカードの使用」</a>を参照してください。</p>
	String <i>communityId</i> , String <i>q</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>指定された検索条件と一致するユーザの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>q</i>—必須項目であり、 は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。  <a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>• <i>pageParam</i>—返すページの数を指定します。0 から開始します。 または 0 を渡すと、最初のページが返されます。</li> <li>• <i>pageSize</i>—ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。          を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <i>communityId</i> , String <i>q</i> , String <i>searchContextId</i> , integer <i>pageParam</i> , Integer <i>pageSize</i>	クラス	<p>指定された検索条件と一致するユーザの指定されたページを返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>q</i>—必須項目であり、 は無効です。検索する文字列を指定します。検索文字列にはワイルドカード文字を除いて 2 文字以上が含まれている必要があります。  <a href="#">「ワイルドカードの使用」</a>を参照してください。</li> <li>• <i>searchContextId</i>—フィード@メンションの検索結果を絞り込むフィード項目 ID。最も役に立つ結果が最初に表示されます。この引数を指定する場合は、500 件を超える結果をクエリできず、検索語にワイルドカードも使用できません。</li> <li>• <i>pageParam</i>—返すページの数を指定します。0 から開始します。 または 0 を渡すと、最初のページが返されます。</li> </ul>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>• <code>pageSize</code> — ページあたりの項目数を指定します。有効な値は 1 ~ 100 です。 を渡すと、デフォルトサイズの 25 に設定されます。</li> </ul>
	String <code>communityId</code> , String <code>userId</code> , String <code>fileId</code> , Integer <code>versionNumber</code>	クラス	<p>指定された、すでにアップロードされているファイルにユーザの写真を設定します。</p> <ul style="list-style-type: none"> <li>• <code>communityId</code> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <code>userId</code> — コンテキストユーザの ID またはキーワード。</li> <li>• <code>fileId</code> — すでにアップロードされたファイルの ID。ファイルは画像であり、2 MB 未満である必要があります。</li> <li>• <code>versionNumber</code> — 既存ファイルのバージョン番号。既存のバージョン番号を指定するか、 を指定して最新バージョンを取得します。</li> </ul>
	String <code>communityId</code> , String <code>userId</code> ,  <span style="color: blue;">クラス</span> <code>fileUpload</code>	クラス	<p>指定されたユーザーの写真として、提供された blob を設定します。画像として使用できるコンテンツタイプである必要があります。</p> <ul style="list-style-type: none"> <li>• <code>communityId</code> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <code>userId</code> — コンテキストユーザの ID またはキーワード。</li> </ul>
	String <code>communityId</code> , String <code>userId</code> , Enum  <code>defaultGroupEmailFrequency</code>	クラス	<p>指定されたユーザのデフォルトの Chatter 設定を更新します。</p> <ul style="list-style-type: none"> <li>• <code>communityId</code> — コミュニティの ID、または のいずれかを使用します。</li> <li>• <code>userId</code> — コンテキストユーザの ID またはキーワード。</li> <li>• <code>defaultGroupEmailFrequency</code> — グループからユーザがメールを受信する頻度を指定します。値は次のとおりです。</li> </ul> <p style="text-align: center;">◊ ◊ ◊</p>

名前	引数	戻り値	説明
		◊ ◊	をコールする とデフォルト値が設定されるため、 <i>defaultGroupEmailFrequency</i> パラメー タに 値を渡さないでください。

関連リンク

## Chatter in Apex クラス

## ConnectApi.Communities クラス

組織内のコミュニティに関する一般情報にアクセスします。

## メソッド

API バージョン 28.0 では、すべてのメソッドを使用できます。

次に、クラスの静的メソッドを示します。

名前	引数	戻り値	説明
		クラス	コンテキストユーザがアクセスできるコミュニティのリストを返します。
Enum	<i>communityStatus</i>	クラス	コンテキストユーザがアクセスできる、指定された状況のコミュニティのリストを返します。 <ul style="list-style-type: none"><li>• <i>communityStatus</i> — コミュニティの現在の状況を指定します。 値は次のとおりです。<ul style="list-style-type: none"><li>◊</li><li>◊</li><li>◊</li></ul></li></ul>

名前	引数	戻り値	説明
	String <i>communityId</i>	クラス	<p>特定のコミュニティに関する情報を返します。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—<i>communityId</i> の ID を指定する必要があります。 またはは指定できません。</li> </ul>

## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi.Organization クラス

組織に関するアクセス情報。

### メソッド

次に、 クラスの静的メソッドを示します。

メソッド	引数	戻り値	説明	バージョン
	なし		有効化されている機能など、組織に関する情報を返します。	28.0

## サンプル

```
public static void main() {
    ConnectApi.Organization org = ConnectApi.Organization.get();
}
```

## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi.Records クラス

motif に関する情報を含む、組織内のレコードに関するアクセス情報。motif は、salesforce.com UI でレコードタイプを区別するために使用される小さいアイコンです。

### 静的メソッド

API バージョン 28.0 では、すべてのメソッドを使用できます。

次に、 クラスの静的メソッドを示します。

名前	引数	戻り値のデータ型	説明
	String <i>communityId</i> , String <i>idOrPrefix</i>		<p>小、中、大の一連の motif アイコンの URL を含む オブジェクトを返します。各 Salesforce レコードタイプには、独自の motif アイコンのセットがあります。</p> <p>を参照してください。</p> <ul style="list-style-type: none"> <li>• <i>communityId</i>—コミュニティの ID、または のいずれかを使用します。</li> <li>• <i>idOrPrefix</i>—ID またはキープレフィックス。</li> </ul>

## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi 入力クラス

一部の メソッドは引数を取ります。これらの引数は、 入力クラスのインスタンスです。

すべてのプロパティは、API バージョン 28.0 で使用できます。

入力クラスは、このドキュメントで抽象クラスと明記されていない限り具象クラスです。具象入力クラスには、引数を取らない公開コンストラクタがあります。

一部のメソッドは、抽象入力オブジェクトを取ります。具象子クラスのインスタンスを作成して、これらのメソッドに渡します。

入力クラスのほとんどのプロパティは、設定可能です。参照のみプロパティについては、このドキュメントで明記されています。

### ConnectApi.BinaryInput クラス

オブジェクトを作成して、フィード項目およびドキュメントにファイルを添付します。

コンストラクタは次のとおりです。

コンストラクタは、次の引数を取ります。

引数	型	説明
	Blob	入力に使用するファイルのコンテンツ

引数	型	説明
	String	など、コンテンツの MIME タイプの説明
	String	UserPhoto.jpg など、ファイル拡張子を含むファイル名

#### ConnectApi.ChatterGroupInput クラス

プロパティ	型	説明
	クラス	グループの「情報」セクション。Web UI では、このセクションは [説明] セクションの上にあります。グループが非公開の場合は、このセクションはメンバーにのみ表示されます。

### ConnectApi.CommentInput クラス

@メンションまたは添付ファイルを含むコメントなど、リッチコメントを追加するために使用します。

プロパティ	型	説明
		省略可能。コメントの添付ファイルを指定します。有効な値は、次のとおりです。 •
	クラス	• はコメントには無効です。

## ConnectApi.ContentAttachmentInput クラス

コメントまたはフィード項目に既存のコンテンツを添付するために使用します。

[ConnectApi.FeedItemAttachmentInput クラス](#) の子クラス

プロパティ	型	説明
	String	既存のコンテンツの ID。

#### **ConnectApi.FeedItemAttachmentInput クラス**

フィード項目にファイルを添付するために使用します。

これは抽象クラスであり、公開コンストラクタはありません。子クラスのインスタンスのみを作成できます。

次のクラスのスーパークラスです。

- ConnectApi.ContentAttachmentInput クラス

- [ConnectApi.LinkAttachmentInput クラス](#)
- [ConnectApi.NewFileAttachmentInput クラス](#)
- [ConnectApi.PollAttachmentInput クラス](#)

#### ConnectApi.FeedItemInput クラス

@メンションまたはファイルを含むフィード項目など、リッチフィード項目を追加するために使用します。また、フィード項目のブックマークにも使用します。

プロパティ	型	説明
		フィード項目の添付ファイルを指定します。フィード項目の種別は、指定された添付ファイルに基づいて推定されます。
	<a href="#">クラス</a>	
		メッセージ本文
	<a href="#">クラス</a>	
Boolean		新しいフィード項目をユーザのためにブックマークする必要があるか( )、否か( )を指定します。
string		共有するフィード項目の 18 文字の ID。
		コンテンツ投稿、テキスト投稿など、フィード項目の種別を指定します。
Enum		• •

#### ConnectApi.GroupInformationInput クラス

プロパティ	型	説明
	String	グループの「情報」セクションのテキスト。
	String	グループの「情報」セクションのタイトル。

#### ConnectApi.HashtagSegmentInput クラス

フィード項目またはコメントにハッシュタグを含めるために使用します。

#### ConnectApi.MessageSegmentInput クラス の子クラス

プロパティ	型	説明
	String	#(ハッシュタグ) プレフィックスを除いたハッシュタグのテキスト

**ConnectApi.LinkAttachmentInput クラス**

リンクを追加するためにフィード項目の添付ファイルの一部として使用します。

**ConnectApi.FeedItemAttachmentInput クラス の子クラス**

プロパティ	型	説明
	String	リンクに使用する URL
	String	リンクのタイトル

**ConnectApi.LinkSegmentInput クラス**

フィード項目またはコメントにリンクセグメントを含めるために使用します。

**ConnectApi.MessageSegmentInput クラス の子クラス**

プロパティ	型	説明
	String	リンクに使用する URL

**ConnectApi.MentionSegmentInput クラス**

フィード項目またはコメントにユーザの @メンションを含めるために使用します。

**ConnectApi.MessageSegmentInput クラス の子クラス**

プロパティ	型	説明
	String	メンションするユーザの ID

**ConnectApi.MessageBodyInput クラス**

フィード項目およびコメントにリッチメッセージを追加するために使用します。

プロパティ	型	説明
	List< クラス>	本文に含まれるメッセージセグメントのリスト

**ConnectApi.MessageSegmentInput クラス**

フィード項目およびコメントにリッチメッセージセグメントを追加するために使用します。

これは抽象クラスであり、公開コンストラクタはありません。子クラスのインスタンスのみを作成できます。

次のクラスのスーパークラスです。

- [ConnectApi.HashtagSegmentInput クラス](#)
- [ConnectApi.LinkSegmentInput クラス](#)

- [ConnectApi.MentionSegmentInput クラス](#)
- [ConnectApi.TextSegmentInput クラス](#)

#### [ConnectApi.NewFileAttachmentInput クラス](#)

フィード項目に添付する新しいファイルを説明します。添付ファイルとなる実際のバイナリファイルは、  
や など、この添付ファイル入力を取るメソッドの [BinaryInput](#) の一部として指定さ  
れます。

#### [ConnectApi.FeedItemAttachmentInput クラス](#) の子クラス

プロパティ	型	説明
	String	アップロードするファイルの説明。
	String	ファイルのタイトル

#### [ConnectApi.PollAttachmentInput クラス](#)

フィード項目にアンケートを添付するために使用します。

#### [ConnectApi.FeedItemAttachmentInput クラス](#) の子クラス

プロパティ	型	説明
	List<String>	アンケート項目のテキストラベル。アンケートには 2 ~ 10 個 の選択肢を含める必要があります。

#### [ConnectApi.TextSegmentInput クラス](#)

フィード項目またはコメントにテキストセグメントを含めるために使用します。

#### [ConnectApi.MessageSegmentInput クラス](#) の子クラス

プロパティ	型	説明
	String	このセグメントのplainテキスト。 <code>text</code> でハッシュタグま たはリンクが検出された場合は、ハッシュタグセグメントまた はリンクセグメントとしてコメントに含まれます。メンション は、 <code>text</code> では検出されず、テキストから分離されることもあ りません。メンションには が必要で す。

## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi 出力クラス

大部分の メソッドは、出力クラスのインスタンスを返します。

すべてのプロパティは、API バージョン 28.0 で使用できます。

テストコード内で作成された出力クラスのインスタンスを除くすべてのプロパティは参照のみです。

このドキュメントの出力クラスは、抽象クラスとして明記されていない限りすべて具象クラスになります。

すべての具象出力クラスには、テストコードからのみ呼び出すことができる、引数をとらないコンストラクタがあります。 「[クラスのテスト](#)」(ページ 315)を参照してください。

### ConnectApi.AbstractMessageBody クラス

このクラスは抽象クラスです。

次のクラスのスーパークラスです。

- [ConnectApi.FeedBody クラス](#)
- [ConnectApi.MessageBody クラス](#)

名前	型	説明
	クラス	メッセージセグメントのリスト
	String	表示可能なテキスト。メッセージセグメントを処理しない場合は、このテキストを使用します。

### ConnectApi.Actor クラス

次のクラスのスーパークラスです。

- [ConnectApi.ActorWithId クラス](#)
- [ConnectApi.UnauthenticatedUser クラス](#)

名前	型	説明
	String	グループ名などのアクター名
	String	次のいずれかになります。 • • • • • レコードタイプ名 — などのレコードタイプ名 または取引先

**ConnectApi.ActorWithId クラス**

次のクラスのスーパークラスです。

- [ConnectApi.ChatterGroup クラス](#)
- [ConnectApi.FileSummary クラス](#)
- [ConnectApi.RecordSummary クラス](#)
- [ConnectApi.User クラス](#)

[ConnectApi.Actor クラス の子クラス](#)

名前	型	説明
	String	アクターの 18 文字の ID
	クラス	アクターをユーザ、グループ、ファイル、カスタムオブジェクトとして識別するアイコン。アイコンは、ユーザまたはグループの写真でも、ファイルのプレビューでもありません。
	String	コンテキストユーザがこの項目をフォローしている場合、登録に関する情報が含まれます。それ以外の場合、を返します。
	String	リソースの Chatter REST API URL

**ConnectApi.Address クラス**

名前	型	説明
	String	市区郡の名前
	String	国の名前
	String	コンテキストユーザのロケールごとに書式設定された住所
	String	都道府県などの名前
	String	町名・番地
	String	郵便番号

**ConnectApi.ApprovalAttachment クラス**

[ConnectApi.FeedItemAttachment クラス](#) の子クラス。

名前	型	説明
	string	作業項目 ID 承認投稿テンプレート項目のコレクション

#### **ConnectApi.ApprovalPostTemplateField クラス**

名前	型	説明
	String	項目名
	String	項目が _____ に設定されている場合、項目値または _____ 。
<b>クラス</b>	レコード ID	レコードが存在しない場合、または参照が _____ の場合、この値は _____ になります。

## ConnectApi.BasicTemplateAttachment クラス

ConnectApi.FeedItemAttachment クラス の子クラス。

このタイプのオブジェクトは、タイプが「フィード」のフィード項目の添付ファイルで返されます。

プロパティ	型	説明
	String	最大 500 文字の説明 (省略可能)
	ラス	ク アイコン (省略可能)
	String	が Salesforce レコードを参照する場合、 にはそのレコードの ID が含まれます。
	String	インライン表示できない追加コンテンツがある場合の詳細 ページへの URL (省略可能)。 を指定していない場合 は、 を指定しないでください。

プロパティ	型	説明
	String	詳細ページのタイトル(省略可能)。が指定される と、タイトルがにリンクします。

**ConnectApi.CaseComment クラス****ConnectApi.FeedItemAttachment クラス の子クラス**

このタイプのオブジェクトは、タイプが のフィード項目の添付ファイルで返されます。

名前	型	説明
	Enum	コメントを行ったユーザの種別を示します。 • — Chatter 顧客がコメントを行った場合 • — サービス担当者がコメントを行った場合
	Datetime	コメントの作成者
	string	ISO8601 の日付文字列(例: 2011-02-25T18:24:31.000Z)
	Boolean	コメントの 18 文字の ID
	string	コメントが公開されたかどうかを示します。
	string	コメントのテキスト

**ConnectApi.ChatterActivity クラス**

名前	型	説明
	Integer	ユーザが行った組織またはコミュニティ内のコメントの合計数
	Integer	ユーザが受け取った組織またはコミュニティ内のコメントの合計数
	Integer	ユーザが受け取った組織またはコミュニティ内の投稿とコメントに対するいいね! の合計数
	Integer	ユーザが行った組織またはコミュニティ内の投稿の合計数

**ConnectApi.ChatterGroup クラス**

このクラスは抽象クラスです。

次のクラスのスーパークラスです。

- [ConnectApi.ChatterGroupDetail クラス](#)
- [ConnectApi.ChatterGroupSummary クラス](#)

[ConnectApi.ActorWithId クラス](#) の子クラス。

名前	型	説明
	Boolean	このグループで Chatter を許可している場合は true、許可していない場合は false。
	string	グループの説明
	string	将来の使用のために予約されています。
	Datetime	グループに投稿された最新のフィード項目の ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	Integer	グループメンバーの合計数
	Enum	グループ所有者、マネージャ、メンバーなど、グループでのユーザのメンバー種別を指定します。 • Member • Manager • Owner
	string	コンテキストユーザがこのグループのメンバーである場合、登録に関する情報が含まれます。それ以外の場合、を返します。
	string	グループの名前。
	string	グループの所有者に関する情報。
	string	グループの写真に関する情報
	Enum	グループが private または public のどちらであるかを指定します。有効な値は、次のとおりです。 • Private • Public

[ConnectApi.ChatterGroupDetail クラス](#)

[ConnectApi.ChatterGroup クラス](#) の子クラス。

名前	型	説明
	Integer	グループに投稿されたファイルの数
	string	グループの「情報」セクションの内容。Web UI では、このセクションは [説明] セクションの上にあります。グル

名前	型	説明
		が非公開の場合は、このセクションはメンバーにのみ表示されます。コンテキストユーザがグループのメンバーでない場合や、コンテキストユーザに「すべてのデータの編集」権限または「すべてのデータの参照」権限がない場合、この値はになります。

**ConnectApi.ChatterGroupPage クラス**

名前	型	説明
	String	グループのこのページへの Chatter REST API URL
		グループの詳細のリスト
		クラス
	String	グループの次のページを識別する Chatter REST API URL
	String	グループの前のページを識別する Chatter REST API URL

**ConnectApi.ChatterGroupSummary クラス**

ConnectApi.ChatterGroup クラス の子クラス。

**ConnectApi.ChatterLike クラス**

名前	型	説明
	String	いいね! の 18 文字の ID
		いいね! と言われたコメントまたはフィード項目への参照
		クラス
	String	いいね! の Chatter REST API URL
		いいね! の作成者
		クラス

**ConnectApi.ChatterLikePage クラス**

名前	型	説明
	Integer	いいね! の現在のページを識別するトークン
	String	いいね! の現在のページを識別する Chatter REST API URL
		いいね! のリスト
		クラス

名前	型	説明
	Integer	いいね! の次のページを識別するトークン
	String	いいね! の次のページを識別する Chatter REST API URL
	Integer	いいね! の前のページを識別するトークン
	String	いいね! の前のページを識別する Chatter REST API URL
	Integer	全ページのいいね! の合計数

**ConnectApi.ClientInfo クラス**

名前	型	説明
	String	認証に使用される接続アプリケーションの名前。
	String	認証に使用される接続アプリケーションの 情報 項 目の値

**ConnectApi.Comment クラス**

名前	型	説明
添付ファイル	クラス	コメントに添付ファイルが含まれる場合、プロパティ値は になります。コメントに添付ファイルが含まれない場合、 になります。
本文	クラス	コメントの本文
接続情報	クラス	接続の認証に使用される接続アプリケーションに関する情報
日付	Datetime	ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
フィード項目	クラス	フィード項目に関する情報
ID	string	コメントの 18 文字の ID
削除可否	Boolean	このプロパティ値が <code>false</code> の場合、コンテキストユーザは コメントを削除できません。 <code>true</code> の場合、コンテキスト ユーザはコメントを削除できる可能性はありますが、必ず 削除できるわけではありません。
最初のページ	クラス	コメントのいいね! の最初のページ
本文	クラス	コメントにいいね! と言ったユーザを記述するメッセージ

名前	型	説明
	クラス	コンテキストユーザがコメントにいいね! と言った場合はその特定のいいね!への参照、それ以外の場合はこのコメントの親フィード項目に関する情報
	クラス	
	string	相対的なローカライズされた文字列として書式設定された作成日（「17 分前」、「昨日」など）。
	Enum	コメントの種別を指定します。 <ul style="list-style-type: none"><li>• コメントに添付ファイルが含まれる</li><li>• コメントにテキストのみが含まれる</li></ul>
	string	このコメントへの Chatter REST API URL
	クラス	コメント作成者に関する情報

**ConnectApi.CommentPage クラス**

名前	型	説明
	クラス	コメントのコレクション
	String	コメントの最初のページを識別するトークン
	String	コメントの最初のページ、つまり最新コメントを識別する Chatter REST API URL
	String	コメントの次のページを識別するトークン
	String	コメントの次のページ、つまり以前のコメントを識別する Chatter REST API URL
	Integer	親フィード項目のコメント合計数

**ConnectApi.Community クラス**

名前	型	説明
	string	コミュニティの説明
	string	コミュニティ ID
	Boolean	ユーザは他の外部ユーザをコミュニティに招待できます。
	string	コミュニティ名
	Boolean	参加時にすべての新規ユーザにメールを送信します。

名前	型	説明
	Enum	コミュニティの現在の状況を指定します。
	string	コミュニティへのフル URL
	string	コミュニティに固有の URL プレフィックス

**ConnectApi.CommunityPage クラス**

名前	型	説明
	クラス	コンテキストユーザがアクセスできるコミュニティのリスト
	Integer	コミュニティの合計数

**ConnectApi.ComplexSegment クラス**

ComplexSegments は、項目変更の一部にすぎません。

[ConnectApi.FieldChangeSegment クラス](#) のスーパークラス

[ConnectApi.MessageSegment クラス](#) の子クラス

これは抽象クラスです。

名前	型	説明
	クラス	メッセージセグメントのリスト

**ConnectApi.ContentAttachment クラス**

[ConnectApi.FeedItemAttachment クラス](#) の子クラス

このタイプのオブジェクトは、タイプが のフィード項目の添付ファイルで返されます。

名前	型	説明
	String	ファイルの MD5 チェックサム
	String	添付ファイルの説明
	String	ファイルの URL
	String	ファイルの拡張子

名前	型	説明
	String	ファイルのサイズ(バイト)。サイズを判定できない場合は、 を返します。
	String	ファイルの種類
	Boolean	ファイルでプレビュー画像を使用できる場合は　、そ れ以外の場合は
	Boolean	ファイルで PDF プレビューを使用できる場合は　、そ れ以外の場合は
	String	コンテンツの 18 文字の ID
	Boolean	将来の使用のために予約されています。
	String	ファイルの MIME タイプ
	String	ファイルの変換リソースへの URL
	String	ファイルのタイトル
	String	ファイルのバージョン番号

**ConnectApi.DashboardComponentAttachment クラス****ConnectApi.FeedItemAttachment クラス** の子クラス

このタイプのオブジェクトは、タイプが のフィード項目の添付ファイルとして返されま  
す。

名前	型	説明
	String	コンポーネントの 18 文字の ID
	String	コンポーネントの名前。コンポーネントと一緒に名前が保 存されていない場合、ローカライズされた文字列 "タイト ル未定のコンポーネント" を返します。
	String	フィード項目の本文でアクターの横に表示するテキスト。 これは、デフォルトの本文テキストの代わりに使用されま す。テキストが指定されておらず、デフォルトの本文テキ ストもない場合、null を返します。
	String	ダッシュボードの 18 文字の ID
	String	ダッシュボードの名前。
	String	実寸大のダッシュボード画像の URL
	Datetime	このダッシュボードの最終更新日がいつかを示す ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	String	最終更新日の表示テキスト ("最終更新 2011 年 10 月 31 日" など)。

名前	型	説明
	クラス	ダッシュボードを実行しているユーザ。
	String	サムネイルサイズのダッシュボード画像の URL。

**ConnectApi.EntityLinkSegment クラス****ConnectApi.MessageSegment クラス の子クラス**

名前	型	説明
	クラス	エンティティ種別(ファイル、グループ、レコード、ユーザ)を指定する小、中、大の一連のアイコン
	クラス	該当する場合は Link オブジェクトへの参照、それ以外の場合は

**ConnectApi.Features クラス**

プロパティ	型	説明
	Boolean	組織で Chatter が有効になっているかどうかを示します。
	Boolean	ユーザの詳細に Chatter 活動に関する情報が含まれるかどうかを示します。
	Boolean	ユーザの詳細にグローバル Chatter 活動が含まれるかどうかを示します。
	Boolean	Chatter メッセージが組織で有効になっているかどうかを示します。
	Boolean	Chatter トピックが有効になっているかどうかを示します。
	Boolean	ユーザがダッシュボードコンポーネントのスナップショットを投稿できるかどうかを示します。
	String	デフォルト通貨の ISO コード。 false に設定されている場合のみ有効です。
	Boolean	is-modified リソースが Chatter API で有効になっているかどうかを示します。
	Boolean	ファイルが Chatter API のリソースとして機能できるかどうかを示します。
	Boolean	ファイルをコメントに添付できるかどうかを示します。
	Boolean	将来の使用のために予約されています。
	Boolean	ユーザの組織がマルチ通貨を使用するか( )、否か( )を示します。 の場合、

プロパティ	型	説明
		はデフォルト通貨の ISO コードを示します。
Boolean		パブリッシャーアクションが有効かどうかを示します
Boolean		将来の使用のために予約されています。
Boolean		トピックのトレンドが有効になっているかどうかを示します。
Boolean		既存の Chatter ユーザが同僚を Chatter に招待できるかどうかを示します。

**ConnectApi.Feed クラス**

名前	型	説明
	String	フィード項目の Chatter REST API URL
	String	フィードがいつ最終更新されたのかが記述された不透明 トークンを含む 要求パラメータがある Chatter REST API URL。フィードがニュースフィードではない場合は を返します。この URL は、ニュースフィードをポー リングして更新する場合に使用します。

**ConnectApi.FeedBody クラス**

ConnectApi.AbstractMessageBody クラス の子クラス。

他のプロパティはありません。

**ConnectApi.FeedFavorite クラス**

名前	型	説明
		お気に入りを含むコミュニティに関する情報 クラス
		お気に入りの作成者 クラス
	string	このお気に入りのフィード項目を識別する Chatter REST API URL
	string	お気に入りの 18 文字の ID
	Datetime	ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	string	お気に入りの名前

名前	型	説明
	string	お気に入りが検索に基づく場合は検索テキストが含まれ、それ以外の場合は空の文字列
	Enum	空の文字列か、次のいずれかの値 • • •
	string	このお気に入りへの Chatter REST API URL
		このお気に入りを保存したユーザに関する情報
		クラス

**ConnectApi.FeedFavorites クラス**

名前	型	説明
	クラス	お気に入りの完全なリスト
	Integer	お気に入りの合計数

**ConnectApi.FeedItem クラス**

名前	型	説明
	クラス	フィード項目を作成したエンティティ。
	クラス	添付ファイルに関する情報。添付ファイルがない場合、を返します。
	クラス	フィード項目の本文
	Boolean	フィード項目を共有できる場合は 、それ以外の場合は
	クラス	接続の認証に使用される接続アプリケーションに関する情報
	クラス	このフィード項目へのコメントの最初のページ
	Datetime	ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	string	フィード項目の 18 文字の ID

名前	型	説明
	Boolean	現在のユーザがこのフィード項目をブックマークした場合は TRUE、それ以外の場合は FALSE。
	Boolean	このプロパティ値が TRUE の場合、コンテキストユーザはコメントを削除できません。 FALSE の場合、コンテキストユーザはコメントを削除できる可能性はありますが、必ず削除できるわけではありません。
	Boolean	フィード項目が行動の変更によって作成された場合は TRUE、それ以外の場合は FALSE。
	Boolean	現在のユーザがこのフィード項目にいいね! と言った場合は TRUE、それ以外の場合は FALSE。
	クラス	このフィード項目へのいいね! の最初のページ
	クラス	フィード項目にいいね! と言ったユーザを記述するメッセージ
	Datetime	ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	クラス	コンテキストユーザがフィード項目にいいね! と言った場合はその特定のいいね! への参照、それ以外の場合は NULL。
	クラス	このフィード項目が共有フィード項目の場合は、元のフィード項目への参照、それ以外の場合は NULL。
	クラス	このフィード項目が共有フィード項目の場合はフィード項目の元の投稿者に関する情報を返し、それ以外の場合は NULL を返します。
	クラス	フィード項目の親
	string	フィード項目に関連付けられた写真の URL
	クラス	フィード項目のタイトルとして使用できるメッセージの書式設定されていないテキストを含む、メッセージセグメントのコレクション。メッセージセグメントには、フィード項目を作成したアクターの名前、リンク、motif アイコン情報が含まれます。
	Enum	コンテンツ投稿、テキスト投稿など、フィード項目の種別を指定します。 <ul style="list-style-type: none"> <li>FEED_TYPE_NEWS - ケースフィードがオンであり、フィードが有効になっている親レコードに関連付けられた行動またはToDo が作成または更新されるときに生成されるフィード項目。</li> </ul>

名前	型	説明
		<ul style="list-style-type: none"> <li>— 承認時のアクションが添付されたフィード項目。承認者は、フィード項目の親で操作を実行できます。</li> </ul>
		<ul style="list-style-type: none"> <li>— ケースフィードのケースに記事が添付されているときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— 画像、リンク、タイトルが含まれた添付ファイルを含む標準表示を行うフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ケースフィードのケースに活動ログが保存されたときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ケースフィードにケースコメントが保存されたときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ケースの状況がケースフィードで変更されたときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— Live Agent チャットのトランスクriptがケースに保存されたときにケースフィードで生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— 新しいグループが作成されたときに生成されるフィード項目。新しいグループへのリンクが含まれます。</li> </ul>
		<ul style="list-style-type: none"> <li>— アーカイブされたグループがアーカイブ解除されたときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ファイルが添付されたフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— パブリッシャーで作成されたコードを説明するフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ダッシュボードアラートが添付されたフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ダッシュボードスナップショットが添付されたフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ケースフィードのケースからメールが送信されたときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ケースフィードのケースから Facebook 投稿が作成されたときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— ハイパーリンクが添付されたフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— アクション可能なアンケートが添付されたフィード項目。フィード項目の閲覧者は、アンケートのオプションで投票できます。</li> </ul>
		<ul style="list-style-type: none"> <li>— 感謝バッジが作成されたときに生成されるフィード項目。</li> </ul>
		<ul style="list-style-type: none"> <li>— 添付ファイルのないフィード項目。</li> </ul>

名前	型	説明
		<ul style="list-style-type: none"> <li>— レコードの 1 つ以上の項目が変更されたときに作成されるフィード項目。</li> <li>— ケースフィードのケースから Twitter 投稿が作成されたときに生成されるフィード項目。</li> <li>—</li> <li>非推奨。ユーザ自身のプロファイルへの投稿。</li> </ul>
	string	<p>このフィード項目への Chatter REST API URL</p> <p>フィード項目を表示できるユーザの種別を指定します。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> </ul>
	Enum	

#### ConnectApi.FeedItemAttachment クラス

子クラス:

- [ConnectApi.ApprovalAttachment クラス](#)
- [ConnectApi.BasicTemplateAttachment クラス](#)
- [ConnectApi.CaseComment クラス](#)
- [ConnectApi.ContentAttachment クラス](#)
- [ConnectApi.DashboardComponentAttachment クラス](#)
- [ConnectApi.FeedPoll クラス](#)
- [ConnectApi.LinkAttachment クラス](#)

他のプロパティはありません。このクラスのすべてのインスタンスは、実際にはいずれかの子クラスです。

これは抽象クラスです。

フィード項目のメッセージセグメントは、  
型です。フィード項目添付ファイルは  
型です。これらはどちらも抽象クラスで、複数の具象サブクラスがあります。実行時、  
を使用すると、これらのオブジェクトの具象型をチェックして、対応するダウンキャストを確実に実行することができます。ダウンキャスト時には、不明なサブクラスを処理するデフォルトの case が必要です。

 重要: フィードの構成は、リリースによって異なる場合があります。コードでは、常に  
オブジェクトと  
オブジェクトの両方  
の不明なサブクラスを処理できるように準備しておく必要があります。

#### ConnectApi.FeedItemPage クラス

名前	型	説明
	String	フィード項目の最初のページを識別するトークン

名前	型	説明
	String	フィード項目の最初のページを識別する Chatter REST API URL
	String	メソッドの <i>since</i> パラメータで使用する不透明ポーリングトークン。このトークンには、フィードがいつ最終更新されたのかが記述されています。
	String	フィードがいつ最終更新されたのかが記述された不透明トークンを含む 要求パラメータがある Chatter REST API URL。フィードがニュースフィードではない場合は を返します。この URL は、ニュースフィードをポーリングして更新する場合に使用します。
クラス		フィード項目のリスト
	String	フィード項目の次のページを識別するトークン
	String	フィード項目の次のページを識別する Chatter REST API URL

#### ConnectApi.FeedModifiedInfo クラス

名前	型	説明
	Boolean	ニュースフィードが最後にポーリングされた後に変更されている場合は 、それ以外の場合は 。フィードがニュースフィードではない場合は を返します。
	String	メソッドの <i>since</i> パラメータで使用する不透明ポーリングトークン。このトークンには、フィードがいつ最終更新されたのかが記述されています。
	String	フィードがいつ最終更新されたのかが記述された不透明トークンを含む 要求パラメータがある Chatter REST API URL。フィードがニュースフィードではない場合は を返します。この URL は、ニュースフィードをポーリングして更新する場合に使用します。 または プロパティで URL を要求した後で、この URL に要求します。

#### ConnectApi.FeedPoll クラス

ConnectApi.FeedItemAttachment クラス の子クラス。

このオブジェクトは、タイプが PollPost の FeedItem の添付ファイルとして返されます。

名前	型	説明
ク	アンケート選択肢のリスト	
ラス	String	このアンケートにおいて現在のユーザが投票したアンケート選択肢の ID。現在のユーザが投票しなかった場合は、が返されます。
	Integer	フィードアンケート項目に投じられた投票の合計数。

**ConnectApi.FeedPollChoice クラス**

名前	型	説明
	String	アンケート選択肢 ID
	Integer	このアンケート選択肢があるアンケート内の場所。最初のアンケート選択肢は 1 から開始します。
	String	アンケート選択肢に関連付けられた表示ラベルテキスト。
	Integer	このアンケート選択肢の投票合計数。
	Double	このアンケートに投じられたすべての投票数に対するこのアンケート選択肢への合計投票数の割合。この割合を 100 で乗算して、このアンケート選択肢の投票数のパーセントを出します。

**ConnectApi.FieldChangeSegment クラス**[ConnectApi.ComplexSegment クラス の子クラス](#)

他のプロパティはありません。

**ConnectApi.FieldChangeNameSegment クラス**[ConnectApi.MessageSegment クラス の子クラス](#)

他のプロパティはありません。

**ConnectApi.FieldChangeValueSegment クラス**[ConnectApi.MessageSegment クラス の子クラス](#)

名前	型	説明
	String	項目変更が URL 項目 (Web アドレスなど) に対するものである場合、URL 値

**ConnectApi.FileSummary クラス**

ConnectApi.ActorWithId クラス の子クラス

名前	型	説明
	String	ファイルの MD5 チェックサム
	Integer	ファイルのサイズ (バイト)
	String	ファイルがリンクの場合は URL を返し、それ以外の場合は文字列 "null" を返します。
	String	ファイルの説明
	String	ファイルのダウンロードに使用できる、ファイルへの URL
	String	ファイルの拡張子
	String	ファイルの種類 (PDF、PowerPoint など)
	String	ファイルの Flash プレビュー バージョンが表示されたかどうかを示します。
	Boolean	将来の使用のために予約されています。
	String	ファイルの MIME タイプ
	Datetime	ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	String	ファイルソースを示します。有効な値は、次のとおりです。 <ul style="list-style-type: none"> <li>• — ファイルソースが Chatter の場合</li> <li>• — ファイルソースがコンテンツの場合</li> </ul>
ファイルの所有者		
クラス		
	String	ファイルの PDF プレビュー バージョンが表示されたかどうかを示します。
	String	ファイルの変換への URL
	String	ファイルの 120x90 ピクセルサイズのプレビュー画像が表示されたかどうかを示します。
	String	ファイルの 240x180 ピクセルサイズのプレビュー画像が表示されたかどうかを示します。
	String	ファイルの 720x480 ピクセルサイズのプレビュー画像が表示されたかどうかを示します。
	String	ファイルのタイトル
	String	ファイルのバージョン番号

**ConnectApi.FollowerPage クラス**

名前	型	説明
	String	現在のページを識別する Chatter REST API URL
リスト クラス		登録のリスト
	String	次のページを識別する Chatter REST API URL
	String	前のページを識別する Chatter REST API URL
	Integer	全ページのフォロワーの合計数

**ConnectApi.FollowingCounts クラス**

名前	型	説明
	Integer	ユーザがフォローしている人の数
	Integer	ユーザがフォローしているレコードの数
	Integer	ユーザがフォローしている項目の合計数

**ConnectApi.FollowingPage クラス**

名前	型	説明
	String	現在のページを識別する Chatter REST API URL
リスト クラス		登録のリスト
	String	次のページの Chatter REST API URL
	String	前のページの Chatter REST API URL
	Integer	全ページのフォローされているレコードの合計数

**ConnectApi.GlobalInfluence クラス**

名前	型	説明
	String	組織またはコミュニティ内でのユーザの影響度ランクを示すパーセント値
	Integer	組織またはコミュニティ内の他の全ユーザに対するユーザの相対的な影響度ランクを示す数値

**ConnectApi.GroupChatterSettings クラス**

特定のグループのユーザの Chatter 設定です。

名前	型	説明
	Enum	グループメンバーがグループからメールを受信する頻度。

**ConnectApi.GroupInformation クラス**

グループの「情報」セクションの内容。Web UI では、このセクションは [説明] セクションの上にあります。グループが非公開の場合は、このセクションはメンバーにのみ表示されます。

名前	型	説明
	String	グループの「情報」セクションのテキスト。
	String	グループの「情報」セクションのタイトル。

**ConnectApi.GroupMember クラス**

名前	型	説明
	string	ユーザの 18 文字の ID
	Enum	グループ所有者、マネージャ、メンバーなど、グループでのユーザのメンバー種別を指定します。 • • • • •
	string	このメンバーシップへの Chatter REST API URL
	クラス	このグループに登録しているユーザに関する情報

**ConnectApi.GroupMemberPage クラス**

名前	型	説明
	string	メンバーのこのページへの Chatter REST API URL
	クラス	グループメンバーのリスト >

名前	型	説明
	クラス	コンテキストユーザがこのグループのメンバーである場合はそのメンバーシップに関する情報を返し、それ以外の場合を返します。
	string	メンバーの次のページを識別する Chatter REST API URL
	string	メンバーの前のページを識別する Chatter REST API URL
	Integer	全ページのグループメンバーの合計数

**ConnectApi.GroupMembershipRequest クラス**

名前	型	説明
	Datetime	ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	string	グループメンバー要求オブジェクトの ID
	Datetime	ISO8601 の日付文字列 (例: 2011-02-25T18:24:31.000Z)
	クラス	コンテキストユーザが参加要求しているグループに関する情報
	string	メンバーシップ要求が却下された場合にユーザに表示するメッセージ。このプロパティの値は、プロパティの値が の場合にのみ使用されます。 最大文字数は 756 文字です。
	Enum	private グループへの参加要求の状況。値は次のとおりです。 • • •
	string	グループメンバーシップ要求オブジェクトの URL
	クラス	グループのメンバーシップを要求しているユーザに関する情報

**ConnectApi.GroupMembershipRequests クラス**

名前	型	説明
	クラス	グループのメンバーシップ要求に関する情報
	Integer	合計要求数

**ConnectApi.HashtagSegment クラス**

ConnectApi.MessageSegment クラス の子クラス

名前	型	説明
	String	ハッシュ記号 (#) を除いたトピックのテキスト
	String	トピックを検索する Chatter REST API Topics リソース:  <i>topic</i>
	String	組織のすべてのフィード項目のトピックを検索する Chatter REST API Feed Items リソースの URL:  <i>topic</i>

**ConnectApi.Icon クラス**

プロパティ	型	説明
	Integer	アイコンの高さ (ピクセル単位)
	Integer	アイコンの幅 (ピクセル単位)
	String	アイコンの URL。この URL は、認証されていないユーザが使用できます。この URL の有効期限はありません。

**ConnectApi.LinkAttachment クラス**

ConnectApi.FeedItemAttachment クラス の子クラス

名前	型	説明
	String	リンクに付けられるタイトル (使用可能な場合)、それ以外の場合は
	String	リンクの URL

**ConnectApi.LinkSegment クラス**

ConnectApi.MessageSegment クラス の子クラス

名前	型	説明
	String	リンクの URL

**ConnectApi.MentionSegment クラス**

ConnectApi.MessageSegment クラス の子クラス

名前	型	説明
	Boolean	メンションされたユーザがメンションされた投稿を表示できるかどうかを示し、メンションされたユーザが投稿を表示できる場合は <code>true</code> 、それ以外の場合は <code>false</code>
	String	メンションされたユーザの名前
	クラス	メンションされたユーザに関する情報

**ConnectApi.MessageBody クラス**

ConnectApi.AbstractMessageBody クラス の子クラス。

他のプロパティはありません。

**ConnectApi.MessageSegment クラス**

次のクラスのスーパークラス:

- [ConnectApi.ComplexSegment クラス](#)
- [ConnectApi.EntityLinkSegment クラス](#)
- [ConnectApi.FieldChangeSegment クラス](#)
- [ConnectApi.FieldChangeNameSegment クラス](#)
- [ConnectApi.FieldChangeValueSegment クラス](#)
- [ConnectApi.HashtagSegment クラス](#)
- [ConnectApi.LinkSegment クラス](#)
- [ConnectApi.MentionSegment クラス](#)
- [ConnectApi.MoreChangesSegment クラス](#)
- [ConnectApi.ResourceLinkSegment クラス](#)
- [ConnectApi.TextSegment クラス](#)

これは抽象クラスです。

フィード項目のメッセージセグメントは、  
型です。フィード項目添付ファイルは  
型です。これらはどちらも抽象クラスで、複数の具象サブクラスがあります。  
実行時、  
を使用すると、これらのオブジェクトの具象型をチェックして、対応するダウンキャストを確実に実行することができます。ダウンキャスト時には、不明なサブクラスを処理するデフォルトの case が必要です。



重要: フィードの構成は、リリースによって異なる場合があります。コードでは、常に  
オブジェクトと  
オブジェクトの両方  
の不明なサブクラスを処理できるように準備しておく必要があります。

#### ConnectApi.MoreChangesSegment クラス

## ConnectApi.MessageSegment クラス の子クラス

大量の追跡変更を含むフィード項目では、メッセージは "changed A, B, and made X more changes" という形式に設定されます。この場合、その他の変更 ("X more changes") に関する情報は に含まれます。

名前	型	説明
	Integer	追加の変更の数

## ConnectApi.Motif

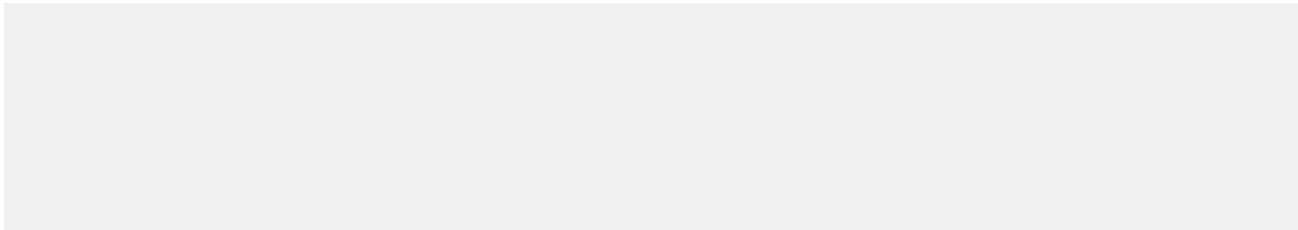
motif クラスの各プロパティには、Salesforce レコードタイプを示す小、中、大のアイコンへの URL があります。Chatter の一般的なレコードタイプは、ファイル、ユーザ、グループですが、すべてのレコードタイプに一連の motif アイコンがあります。カスタムオブジェクトレコードでは、タブスタイルアイコンが使用されます。認証されていないユーザでもすべてのアイコンを使用できるため、たとえば、motif アイコンをメールで表示することができます。



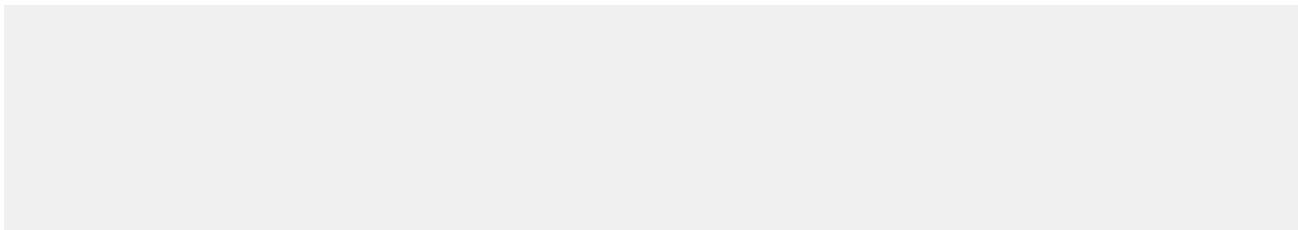
メモ: motif 画像はアイコンであり、ユーザがアップロードした画像または写真ではありません。たとえば、すべてのユーザは同じセットの motif アイコンを使用できます。

カスタムオブジェクトレコードでは、タブスタイルアイコンが使用されます。たとえば、次のカスタムオブジェクトでは、「boat」タブスタイルが使用されます。

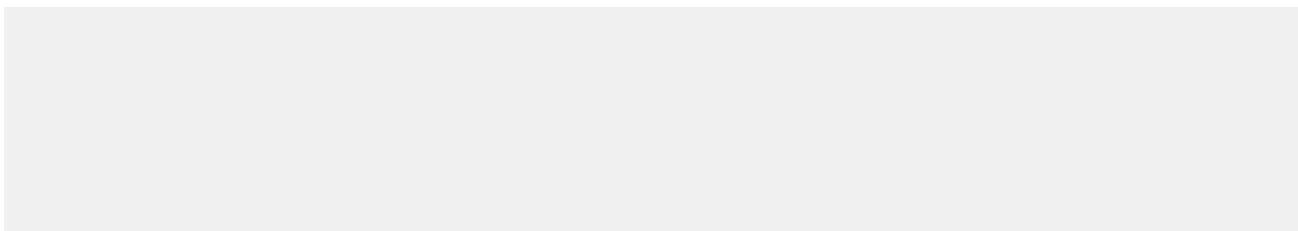
ユーザは、次のアイコンを使用します。



グループは、次のアイコンを使用します。



ファイルは、次のアイコンを使用します。



メモ: 前の例のアイコンを表示するには、URL を  
*instance\_name*

*instance\_name* で置き換えます。たとえば、  
に保存されます。

名前	型	説明
	String	レコードタイプを示す大アイコン
	String	レコードタイプを示す中アイコン
	String	レコードタイプを示す小アイコン

**ConnectApi.OrganizationSettings クラス**

名前	型	説明
	Integer	この時間を過ぎると、何も操作を行っていないユーザに対し、ログアウトするか操作を続行するかを選択させるポップアップウィンドウが表示されます。
	クラス	組織で使用可能な機能に関する情報
	String	組織名
	String	組織の 18 文字の ID
	クラス	ユーザの組織権限に関する情報

**ConnectApi.PhoneNumber クラス**

名前	型	説明
	String	電話番号
	String	値は次のとおりです。 • • •

**ConnectApi.Photo クラス**

名前	型	説明
	String	大きなプロファイル写真への一時的な URL。この URL は認証されていないユーザでも利用でき、有効期限が 30 日後に切れます。
	String	大きなプロファイル写真への URL。デフォルトの幅は 200 ピクセルです。高さは、元の画像の比率が維持されるように設定されます。
	String	そのバージョンの写真の 18 文字の ID
	String	小さいプロファイル写真への URL。デフォルトのサイズは 64x64 ピクセルです。
	String	小さいプロファイルへの一時的な URL。この URL は認証されていないユーザでも利用でき、有効期限が 30 日後に切れます。
	String	などの写真オブジェクトを返すリソース

**ConnectApi.RecordSummary クラス****ConnectApi.ActorWithId クラス** の子クラス

他のプロパティはありません。これは、特殊な子クラスのない汎用レコードです。

**ConnectApi.Reference クラス****ConnectApi.MessageSegment クラス** の子クラス

名前	型	説明
	String	参照されるオブジェクトの 18 文字の ID
	String	リソースへの URL

**ConnectApi.ResourceLinkSegment クラス**

名前	型	説明
	String	他の方法では ID 項目 (ユーザのリストへのリンクなど) で識別されることのないリソースへの URL

**ConnectApi.Subscription クラス**

名前	型	説明
		登録が存在するコミュニティに関する情報 クラス
	String	登録の 18 文字の ID
		親、つまりフォロー対象のものまたは人に関する情報 クラス
		登録者、つまりこの項目をフォローしている人に関する情 報
	String	この特定の登録への Chatter REST API URL

**ConnectApi.TextSegment クラス****ConnectApi.MessageSegment クラス** の子クラス

他のプロパティはありません。

**ConnectApi.UnauthenticatedUser クラス****ConnectApi.Actor クラス** の子クラス

他のプロパティはありません。

このクラスのインスタンスは、Chatter 顧客が投稿したフィード項目およびコメントのアクターとして使用されます。

#### `ConnectApi.User`

次のクラスのスーパークラス:

- [ConnectApi.UserDetail クラス](#)
- [ConnectApi.UserSummary クラス](#)

[ConnectApi.ActorWithId クラス](#) の子クラス

名前	型	説明
	string	会社の名前
	string	ユーザの名
	Boolean	ユーザが Chatter 顧客の場合は 、それ以外の場合は
	Boolean	ユーザがコンテキストユーザと同じコミュニティに存在する場合は 、それ以外の場合は
	string	ユーザの姓
ユーザの写真に関する情報		
クラス		
	string	ユーザの役職
	Enum	ユーザの種別を指定します。 •                   — 外部グループの Chatter 顧客 •                   — Chatter Free 顧客 •                   — 認証されていないユーザ •                   — 標準組織メンバー •                   — カスタマーポータルユーザ、コミュニティ ユーザなど •                   — Chatter Expert またはシステムユーザ •                   — カスタムオブジェクトのユーザ種別

#### `ConnectApi.UserChatterSettings` クラス

ユーザのグローバル Chatter 設定。

名前	型	説明
	Enum	ユーザが参加するグループからメールを受信するデフォルトの頻度。

**ConnectApi.UserDetail クラス**

ConnectApi.User の子クラス

プロパティを表示する権限がコンテキストユーザがない場合、この値は \_\_\_\_\_ に設定されます。

名前	型	説明
	String	ユーザのプロファイルから取得したテキスト
		ユーザの住所
クラス		
		Chatter 活動統計
クラス		ク
		ユーザの影響度ランク
クラス		
	String	ユーザのメールアドレス
	Integer	このユーザをフォローしているユーザの数
		ユーザがフォローしている項目に関する情報
クラス		ク
	Integer	ユーザがフォローしているグループの数
	Boolean	ユーザが有効な場合は _____ 、それ以外の場合は
	String	ユーザのマネージャの 18 文字の ID
	String	ロケールに基づいて連結されたマネージャの姓と名
		ユーザの電話番号のコレクション
クラス		
	String	ユーザのユーザ名 ( <i>Admin@mycompany.com</i> など)

**ConnectApi.UserGroupPage クラス**

名前	型	説明
	String	現在のページを識別する Chatter REST API URL
		グループのリスト
クラス		
	String	次のページの Chatter REST API URL
	String	前のページの Chatter REST API URL
	Integer	全ページのグループの合計数

**ConnectApi.UserPage クラス**

名前	型	説明
	Integer	現在のページを識別するトークン
	String	現在のページを識別する Chatter REST API URL
	Integer	次のページを識別するトークン
	String	次のページを識別する Chatter REST API URL
	Integer	前のページを識別するトークン
	String	前のページを識別する Chatter REST API URL
クラス		ユーザ詳細情報のリスト。プロパティを表示する権限がコンテキストユーザにない場合、プロパティは null に設定されます。

**ConnectApi.UserSettings クラス**

プロパティ	型	説明
	Boolean	ユーザは、Chatter 投稿からワークフローを承認できます。
	Boolean	ユーザがユーザやレコードをフォローできるかどうか。
	Boolean	ユーザに「すべてのデータの編集」権限があるかどうか。
	Boolean	ユーザはグループを所有できます。
	Boolean	ユーザに「すべてのデータの参照」権限があるかどうか。
	Boolean	ユーザに「すべてのグループの参照」権限があるかどうか。
	Boolean	ユーザに「すべてのユーザの参照」権限があるかどうか。
	Boolean	ユーザが他のユーザの Chatter プロファイルを表示できるかどうか。
	Boolean	ユーザが公開とマークされたすべてのファイルを表示できるかどうか。
	String	通貨の値を表示するために使用する通貨記号。 プロパティが null に設定されている場合にのみ有効です。
	Boolean	ユーザが Chatter 顧客であるかどうか。
	Boolean	ユーザが、社内組織のメンバーであるかどうか。
	Boolean	将来の使用のために予約されています。

プロパティ	型	説明
	String	デフォルト通貨の ISO コード。 プロパティが に設定されている場合にのみ有効です。
	String	ユーザの 18 文字の ID
	String	ユーザのロケール

**ConnectApi.UserSummary クラス**

ConnectApi.User の子クラス。

名前	型	説明
	Boolean	ユーザが有効な場合は 、それ以外の場合は

**関連リンク**

[Chatter in Apex クラス](#)

**ConnectApi.Enum**

Enum は、値が定数であるデータ型です。

Enum は、Apex Enum のプロパティとメソッドをすべて継承します。

には、次の Enum があります。

Enum	説明
	コメントを行ったユーザの種別を示します。 <ul style="list-style-type: none"> <li>• — Chatter 顧客がコメントを行った場合</li> <li>• — サービス担当者がコメントを行った場合</li> </ul>
	コメントの種別を指定します。 <ul style="list-style-type: none"> <li>• — コメントに添付ファイルが含まれる</li> <li>• — コメントにテキストのみが含まれる</li> </ul>
	コミュニティの現在の状況を指定します。 <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> </ul>

Enum	説明
	<p>検索語またはリストビューなど、フィードのお気に入りの発生元を指定します。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> </ul>
	<p>フィード項目の出力オブジェクトに使用する添付ファイルの種別を指定します。</p> <ul style="list-style-type: none"> <li>• — 承認を必要とするフィード項目。</li> <li>• — 画像、リンク、タイトルの汎用表示を行うフィード項目。</li> <li>• — ケースレコードへのコメントから作成されるフィード項目。</li> <li>• — ファイルが添付されたフィード項目。</li> <li>• — ダッシュボードが添付されたフィード項目。</li> <li>• — URL が添付されたフィード項目。</li> <li>• — アンケートが添付されたフィード項目。</li> </ul>
	<p>フィード項目の入力オブジェクトに使用する添付ファイルの種別を指定します。</p> <ul style="list-style-type: none"> <li>• — 以前にアップロードされたファイルが添付されたフィード項目。</li> <li>• — URL が添付されたフィード項目。</li> <li>• — 投稿とともにアップロードされたファイルが添付されたフィード項目。</li> <li>• — アンケートが添付されたフィード項目。</li> </ul>
	<p>コンテンツ投稿、テキスト投稿など、フィード項目の種別を指定します。</p> <ul style="list-style-type: none"> <li>• — ケースフィードがオンであり、フィードが有効になっている親レコードに関連付けられた行動またはToDo が作成または更新されるときに生成されるフィード項目。</li> <li>• — 承認時のアクションが添付されたフィード項目。承認者は、フィード項目の親で操作を実行できます。</li> <li>• — ケースフィードのケースに記事が添付されているときに生成されるフィード項目。</li> <li>• — 画像、リンク、タイトルが含まれた添付ファイルを含む標準表示を行うフィード項目。</li> </ul>

Enum	説明
	<ul style="list-style-type: none"><li>— ケースフィードのケースに活動ログが保存されたときに生成されるフィード項目。</li><li>— ケースフィードにケースコメントが保存されたときに生成されるフィード項目。</li><li>— ケースの状況がケースフィードで変更されたときに生成されるフィード項目。</li><li>— Live Agent チャットのトランスクriptがケースに保存されたときにケースフィードで生成されるフィード項目。</li><li>— 新しいグループが作成されたときに生成されるフィード項目。新しいグループへのリンクが含まれます。</li><li>— アーカイブされたグループがアーカイブ解除されたときに生成されるフィード項目。</li><li>— ファイルが添付されたフィード項目。</li><li>— パブリッシャーで作成されたレコードを説明するフィード項目。</li><li>— ダッシュボードアートが添付されたフィード項目。</li><li>— ダッシュボードスナップショットが添付されたフィード項目。</li><li>— ケースフィードのケースからメールが送信されたときに生成されるフィード項目。</li><li>— ケースフィードのケースから Facebook 投稿が作成されたときに生成されるフィード項目。</li><li>— ハイパーリンクが添付されたフィード項目。</li><li>— アクション可能なアンケートが添付されたフィード項目。フィード項目の閲覧者は、アンケートのオプションで投票できます。</li><li>— 感謝バッジが作成されたときに生成されるフィード項目。</li><li>— 添付ファイルのないフィード項目。</li><li>— レコードの 1 つ以上の項目が変更されたときに作成されるフィード項目。</li><li>— ケースフィードのケースから Twitter 投稿が作成されたときに生成されるフィード項目。</li><li>—</li></ul>

Enum	説明
	非推奨。ユーザ自身のプロファイルへの投稿。
	フィード項目を表示できるユーザの種別を指定します。 <ul style="list-style-type: none"><li>•</li><li>•</li></ul>
	作成日や最終更新日などで並び替えて返される順序を指定します。 <ul style="list-style-type: none"><li>•</li><li>•</li></ul>
	ブックマーク、グループ、レコードで構成されるフィードなど、フィードの種別を指定します。 <ul style="list-style-type: none"><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li></ul>
	ユーザがグループからメールを受信する頻度を指定します。 <ul style="list-style-type: none"><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li></ul>
	メソッドではデフォルト値が設定されるため、このメソッドで値を使用しないでください。
	グループ所有者、マネージャ、メンバーなど、グループでのユーザのメンバー種別を指定します。 <ul style="list-style-type: none"><li>•</li><li>•</li><li>•</li><li>•</li><li>•</li></ul>

Enum	説明
	private グループへの参加要求の状況。 • • •
	グループが private または public のどちらであるかを指定します。 • •
	テキスト、リンク、項目名の変更、項目値の変更など、メッセージセグメントの種別を指定します。 • • • • • • • • • •
	ユーザの種別を指定します。 •       — 外部グループの Chatter 顧客 •       — Chatter Free 顧客 •       — 認証されていないユーザ •       — 標準組織メンバー •       — カスタマーポータルユーザ、コミュニティユーザなど •       — Chatter Expert またはシステムユーザ •       — カスタムオブジェクトのユーザ種別
	ワークフロープロセスの状況を指定します。 • • • • • • •

Enum	説明
•	

## 関連リンク

[Chatter in Apex クラス](#)

## ConnectApi 例外

名前空間には、例外クラスが含まれています。

すべての例外は、エラーメッセージや例外型を返す組込みメソッドをサポートしています。  
名前空間には、次の例外があります。

例外	説明
•	アプリケーションでコードを利用する方法に論理エラーがあります。これは、REST の Chatter API で発生する 400 エラーと同じです。
•	指定された検索中のリソースに問題があります。これは、REST の Chatter API で発生する 404 エラーと同じです。
•	レート制限を超えたときに発生します。これは、REST の Chatter API で発生する 503 Service Unavailable エラーと同じです。

## 関連リンク

[Chatter in Apex クラス](#)

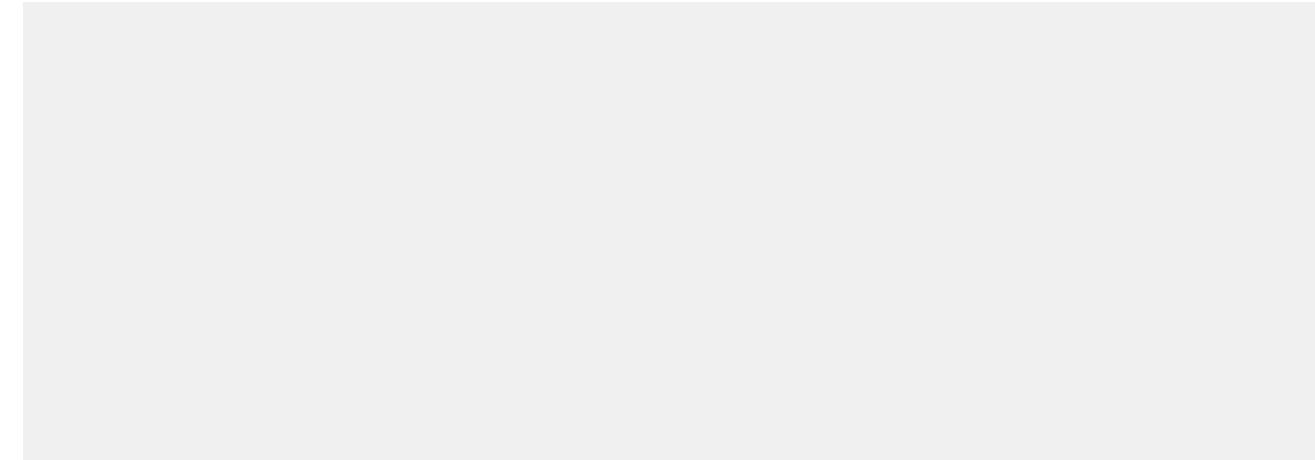
## 例外クラス

Apex で独自の例外クラスを作成できます。例外は最上位クラスにできます。つまり、メンバー変数、メソッド、コンストラクタを持ち、インターフェースの実装などが可能です。

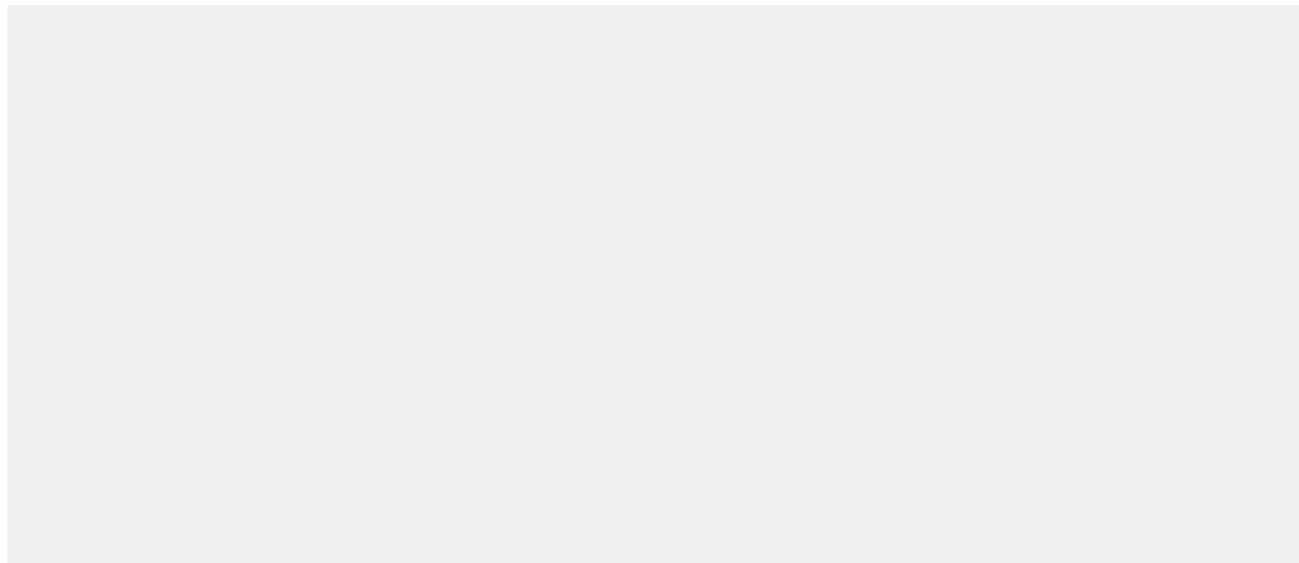
作成した例外は、他の標準の例外型と同様に、例外を発生させたり、発生した例外を捕えたりできます。

ユーザ定義の例外クラスの名前は、末尾に `Exception` を付ける必要があります。たとえば、「`MyException`」や「`PurchaseException`」などです。すべての例外クラスは、自動的にシステム定義の基本クラス `Exception` を拡張します。

たとえば、次のコードでは匿名ブロック内の例外型を定義します。



Java クラスと同様に、ユーザ定義の例外型は継承ツリーを構成し、キャッチブロックにより任意の部分を受け取ることができます。次に例を示します。



この項で説明する内容は次のとおりです。

- [例外の作成](#)
- [例外変数の使用](#)

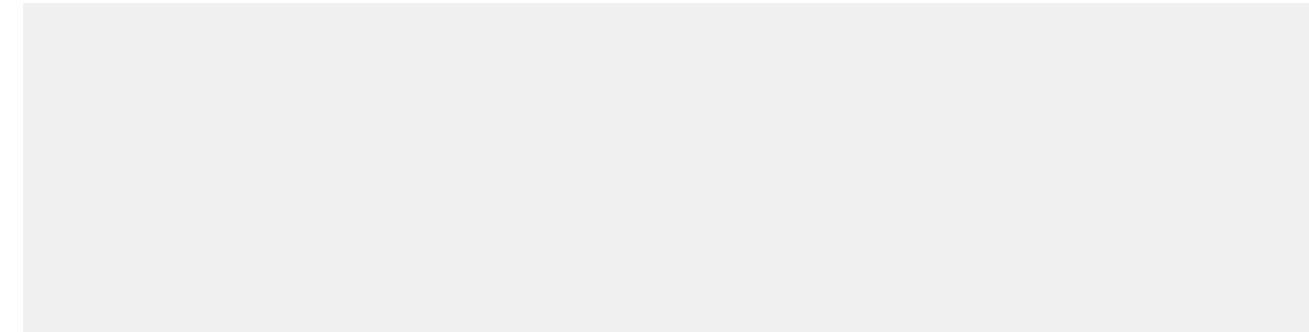
「[例外メソッドの使用](#)」も参照してください。

## 例外の作成

次のような例外を作成できます。

- 引数のない例外
- エラーメッセージを指定する 1 つの string 型の引数を取る例外
- 1 つの Exception 型の引数を取るもの。これは原因を特定でき、任意にスタック追跡できます
- string 型のエラーメッセージと、任意のスタック追跡に表示される例外チェーンの両方を取る例外

たとえば、次のコードは と の両方の情報を含むスタック追跡を生成します。



次の図は、上記のコードを実行した結果のスタック追跡を示します。



```
Log View
19.0 DB_INFO;WORKFLOW_INFO;VALIDATION_INFO;CALLOUT_INFO;APEX_CODE;DEBUG;APEX_PROFILING;INFO
Execute Anonymous: public class My1Exception extends Exception {}
Execute Anonymous: public class My2Exception extends Exception {}
Execute Anonymous: try {
Execute Anonymous:     throw new My1Exception();
Execute Anonymous: } catch (My1Exception e) {
Execute Anonymous:     throw new My2Exception("This is bad", e);
Execute Anonymous: }
13:54:02.084|EXECUTION_STARTED
13:54:02.084|CODE_UNIT_STARTED|[EXTERNAL]|execute_anonymous_apex
13:54:02.222|EXCEPTION_THROWN|[4.5]My1Exception: Script-thrown exception
13:54:02.223|EXCEPTION_THROWN|[6.5]My2Exception: This is bad
13:54:02.227|FATAL_ERROR|My2Exception: This is bad

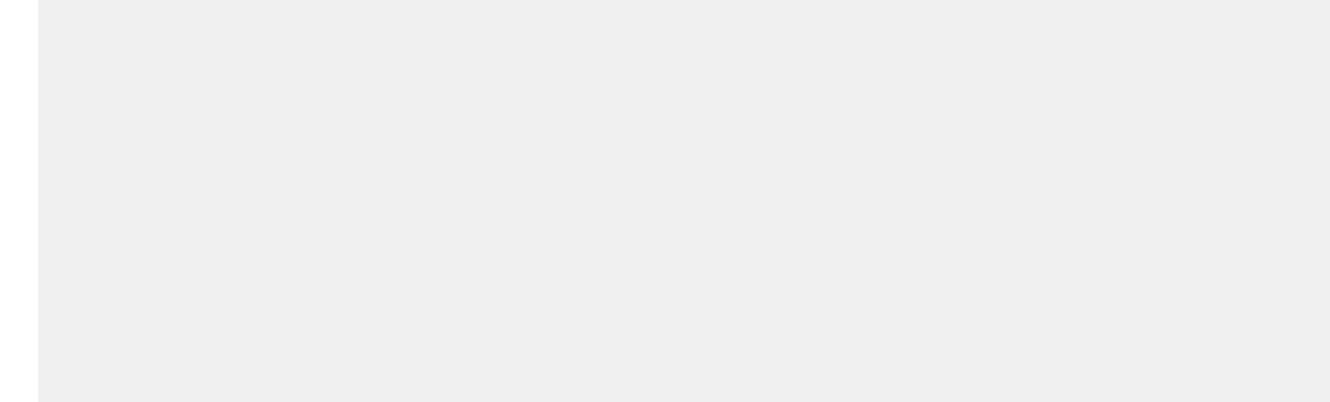
AnonymousBlock: line 6, column 11
Caused by:
AnonymousBlock: line 4, column 11
13:54:02.227|CUMULATIVE_LIMIT_USAGE
13:54:02.227|LIMIT_USAGE_FOR_NS|(default)
Number of SOQL queries: 0 out of 1000
Number of query rows: 0 out of 10000
Number of SOSL queries: 0 out of 20
Number of DML statements: 0 out of 100
Number of DML rows: 0 out of 10000
Number of script statements: 2 out of 200000
Maximum heap size: 0 out of 2000000
Number of callouts: 0 out of 10
Number of Email Invocations: 0 out of 10
Number of fields describes: 0 out of 10
Number of record type describes: 0 out of 10
Number of child relationships describes: 0 out of 10
Number of picklist describes: 0 out of 10
Number of future calls: 0 out of 10
Number of find similar calls: 0 out of 10
Number of System.runAs() invocations: 0 out of 20

13:54:02.227|CUMULATIVE_LIMIT_USAGE_END
13:54:02.228|CODE_UNIT_FINISHED
13:54:02.228|EXECUTION_FINISHED
```

図 8:例外のスタック追跡(デバッグログより)

## 例外変数の使用

Java と同様に、Exception型は変数、引数、戻り値としても使用できます。Exception型は Apex ではシステム定義基本クラスです。次に例を示します。



## Flow.Interview クラス

クラスは、フローへの高度な Visualforce コントローラアクセスを提供します。

## 使用方法

Visual Workflow で使用する

クラスには、Visualforce ページ、またはサブフロー要素によってコールされる個別フローに埋め込まれたフロー変数の値に Visualforce コントローラからアクセスできる 1 つのメソッドがあります。

## メソッド

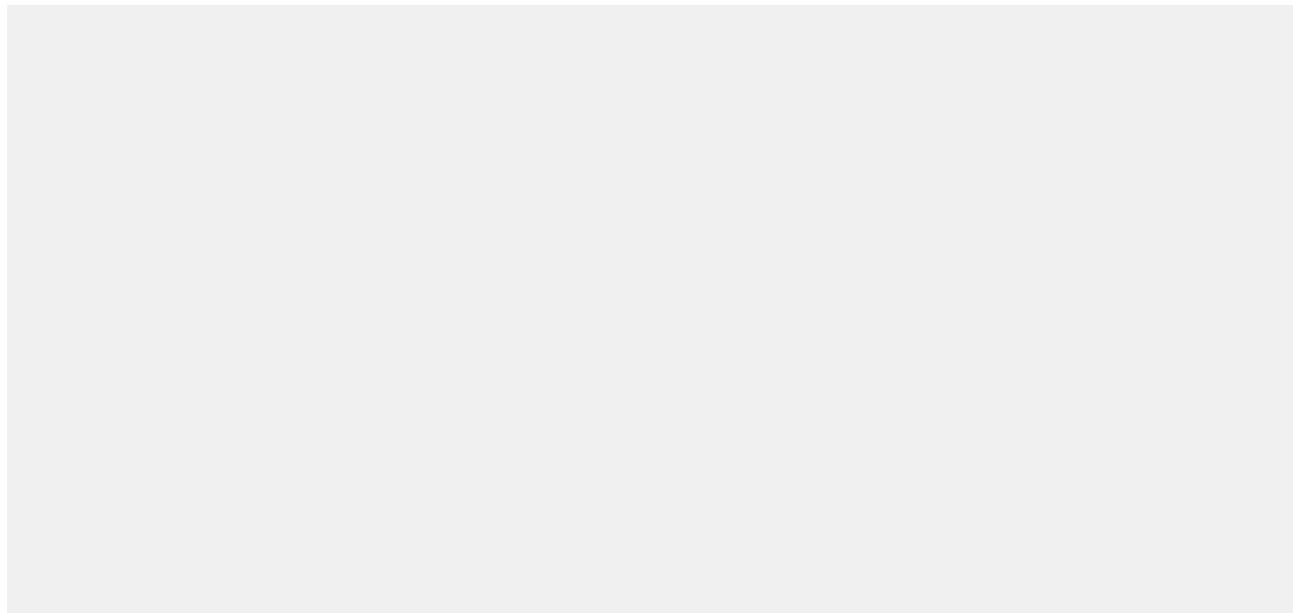
クラスのインスタンスマソッドを次に示します。

メソッド	引数	戻り値	説明
	String <i>variableName</i>	Object	<p>指定されたフロー変数の値を返します。フロー変数は、Visualforce ページに埋め込まれたフロー内、またはサブフロー要素によってコールされる個別のフロー内にあります。</p> <p>変数値は、これらのうちインタビューが現在実行されているフローから返されます。指定された変数がフロー内に見つからない場合、メソッドは <code>null</code> を返します。</p> <p><i>variableName</i> 引数にはフロー変数の一意の名前を指定します。</p> <p>このメソッドは、コンパイル時ではなく実行時にのみ変数の存在を確認します。</p>

## サンプル

次のサンプルでは、

メソッドを使用して Visualforce ページに埋め込まれたフローからプレッドクラム(ナビゲーション)情報を取得します。そのフローにサブフロー要素が含まれ、参照される各フローにも `vaBreadcrumb` 変数が含まれる場合、どのフローでインタビューが実行されているかに関わらず、すべてのフローのプレッドクラムを Visualforce ページから取得できます。



## HTTP (RESTful) サービスクラス

次のクラスを使用して HTTP サービス (RESTful サービスとも呼ばれる) にアクセスできます。

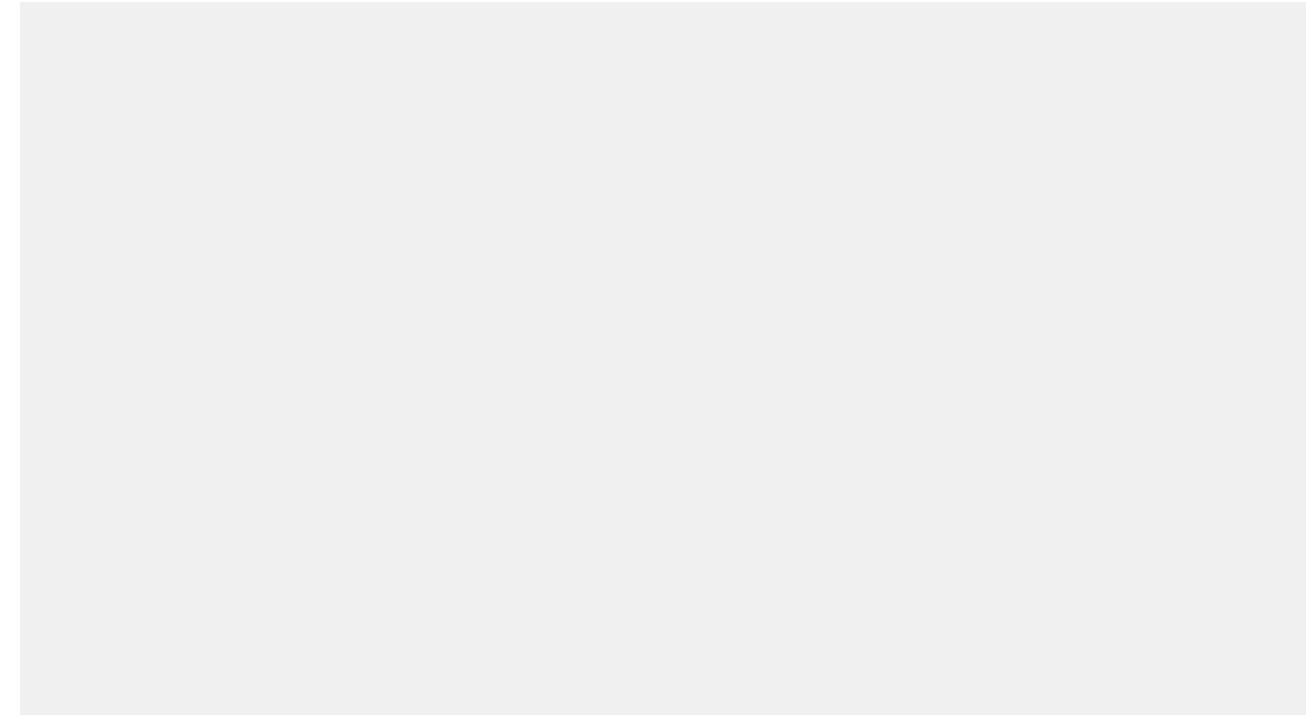
- [HTTP クラス](#)
- [クラス](#)
- [クラス](#)

### HTTP クラス

これらのクラスは一般的な HTTP 要求/応答の機能を表示します。

- [クラス](#): HTTP 要求と応答を開始するにはこのクラスを使用します。
- [クラス](#): プログラムに基づいて GET、POST、PUT、および DELETE のような HTTP 要求を作成するには、このクラスを使用します。
- [クラス](#): で返された HTTP の応答を処理するには、このクラスを使用します。
  - クラスと クラスは、次の要素をサポートします。
    - :
    - ◊ GET、POST、PUT、DELETE、TRACE、CONNECT、HEAD、および OPTIONS などの HTTP 要求型
    - ◊ 要求ヘッダー (必要な場合)
    - ◊ 読み取りおよび接続タイムアウト
    - ◊ リダイレクト (必要な場合)
    - ◊ メッセージ本文の内容
  - :
  - ◊ HTTP 状況コード
  - ◊ 要求ヘッダー (必要な場合)
  - ◊ レスポンスボディの内容

次の例は、メソッドに送られた `url` の値によって指定された外部サーバへの HTTP GET 要求を示します。この例では、返されたレスポンスボディへのアクセスについても示します。



前の例は、同期して実行されます。つまり、外部 Web サービスが応答を返すまで、それ以上の処理は発生しません。または、[@future アノテーション](#)を使用してコールアウトを非同期に実行することもできます。

Apex またはその他の機能を使用してエンドポイントまたはリダイレクトエンドポイントから外部サーバにアクセスするには、Salesforce ユーザインターフェース内の認証されたリモートサイトのリストにリモートサイトを追加する必要があります。そのためには、Salesforce にログインし、[設定] から [セキュリティのコントロール] > [リモートサイトの設定] をクリックします。



メモ: AJAX プロキシは、リダイレクトと認証チャレンジ (401/407 応答) を自動的に処理します。AJAX プロキシの詳細は、[AJAX Toolkit のマニュアル](#)を参照してください。

で作成されたリクエストボディ内、または でアクセスされたレスポンスボディ内の XML または JSON コンテンツを解析するには、[DOM クラス](#)または[JSON クラス](#)を使用します。

## Http クラス

HTTP 要求と応答を開始するには `HttpRequest` クラスを使用します。このクラスには、次の公開メソッドが含まれます。

名前	引数	戻り値	説明
	<code>HttpRequest request</code>	<code>System.HttpResponse</code>	<code>HttpRequest</code> を送信して、応答を返します。
		<code>string</code>	オブジェクトのプロパティを表示、特定する文字列を返します。

**HttpRequest クラス**

GET、POST、PUT、およびDELETEのようなHTTP要求をプログラムで作成するには、  
クラスを使用します。

クラ

で作成されたリクエストボディ内のXMLまたはJSONコンテンツを解析するには、[DOM クラス](#)  
または[JSON クラス](#)を使用します。

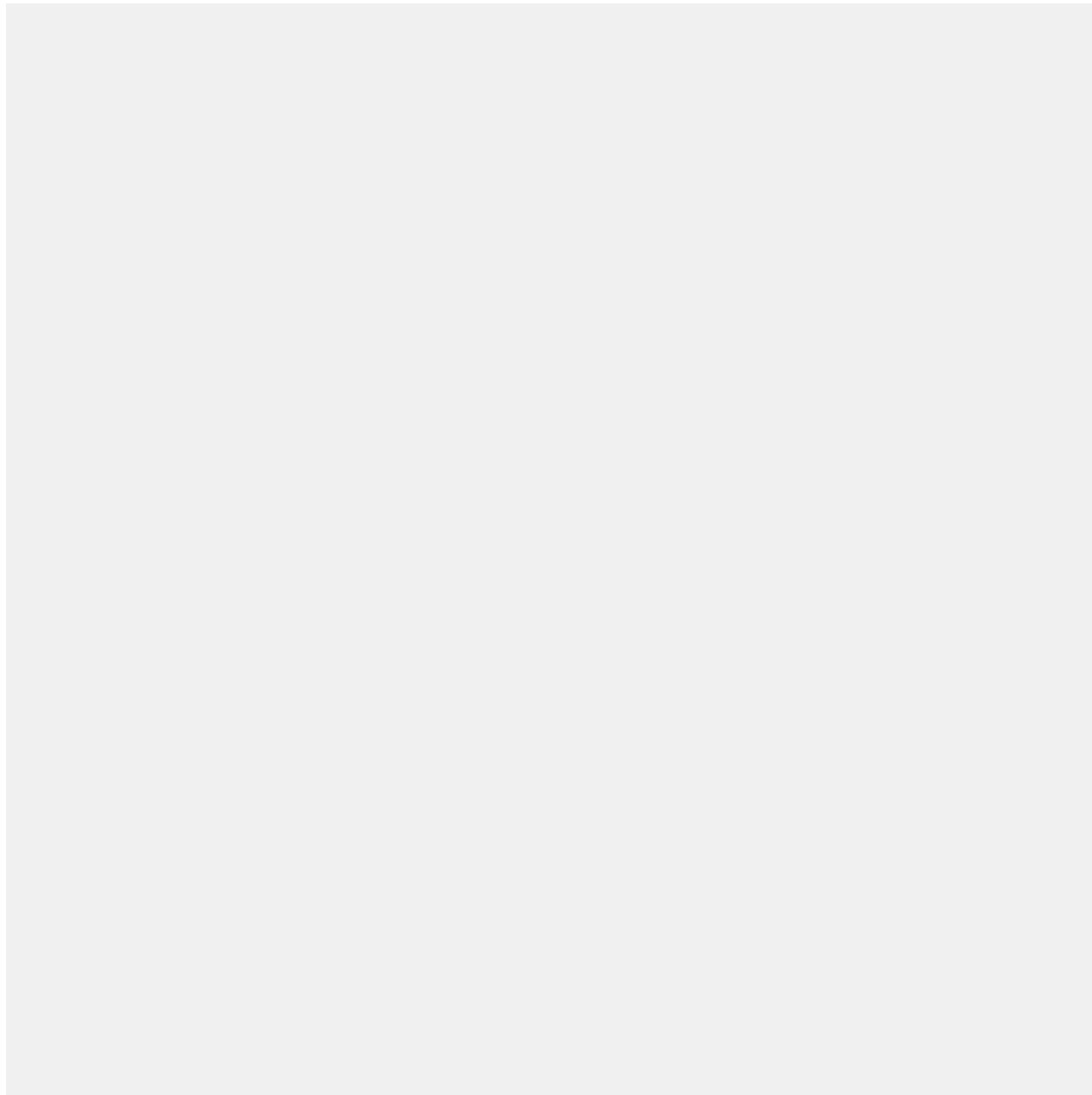
クラスには、次の公開メソッドが含まれます。

名前	引数	戻り値	説明
		string	このリクエストボディを取得します。
		Blob	このリクエストボディを Blob として取得します。
			このリクエストボディを DOM ドキュメントとして取得します。次のショートカットとして使用できます。
		boolean	の場合、リクエストボディは圧縮され、 の場合は圧縮されません。
		string	この要求の外部サーバのエンドポイントのURL を取得します。
<i>String key</i>	string		要求ヘッダーの内容を取得します。
		string	によって使用されるメソッドの 種別を返します。次に例を示します。 <ul style="list-style-type: none"><li>• Delete</li><li>• GET</li><li>• HEAD</li><li>• POST</li><li>• PUT</li><li>• TRACE</li></ul>
<i>String body</i>	Void		このリクエストボディの内容を設定します。最大 3 MB です。  HTTP 要求のサイズおよび応答のサイズは、 ヒープサイズの合計にカウントされます。

名前	引数	戻り値	説明
	BLOB <i>body</i>	Void	Blob を使用して、このリクエストボディの内容を設定します。最大 3 MB です。 HTTP 要求のサイズおよび応答のサイズは、ヒープサイズの合計にカウントされます。
	Dom.Document <i>document</i>	Void	このリクエストボディの内容を設定します。内容は DOM ドキュメントを表します。最大 3 MB です。
	String <i>clientCert</i> String <i>password</i>	Void	このメソッドは非推奨です。代わりに、 を使用してください。 サーバが認証用のクライアント証明書を要求する場合、クライアント証明書 PKCS12 キーストアとパスワードを設定します。
	String <i>certDevName</i>	Void	外部サービスに認証用のクライアント証明書が必要な場合、証明書の名前を設定します。 「 <a href="#">HTTP 要求での証明書の使用</a> 」を参照してください。
	Boolean <i>flag</i>	Void	の場合、本文内のデータは gzip 圧縮形式でエンドポイントに配信されます。 の場合、非圧縮形式が使用されます。
	String <i>endpoint</i>	Void	この要求の外部サーバのエンドポイントのURLを設定します。
	String <i>key</i> String <i>value</i>	Void	要求ヘッダーの内容を設定します。制限 100 KB
	String <i>method</i>		HTTP 要求によって使用されるメソッドの種別を設定します。次に例を示します。 <ul style="list-style-type: none"><li>• Delete</li><li>• GET</li><li>• HEAD</li><li>• POST</li><li>• PUT</li><li>• TRACE</li></ul> このメソッドは要求オプションの設定にも使用できます。
	Integer <i>timeout</i>	Void	要求のタイムアウトをミリ秒単位で設定します。 値は 1 ~ 60,000 ミリ秒の間で設定します。

名前	引数	戻り値	説明
		string	この要求の外部サーバのエンドポイントのURLと、使用されるメソッドが含まれる文字列を返します。次に例を示します。

次の例は、要求を含む認証ヘッダーの使用方法と応答の処理を示しています。



## 圧縮

送信するデータを圧縮する必要がある場合は、次のサンプルで説明するように  
を使用してください。

```
my_endpoint
```

```
some post body
```

応答が圧縮形式で返されると、  
は、自動的に形式を認識して展開し、展開された値を返します。

## HttpResponse クラス

クラスによって返された HTTP 応答を処理するには、  
クラスを使用します。

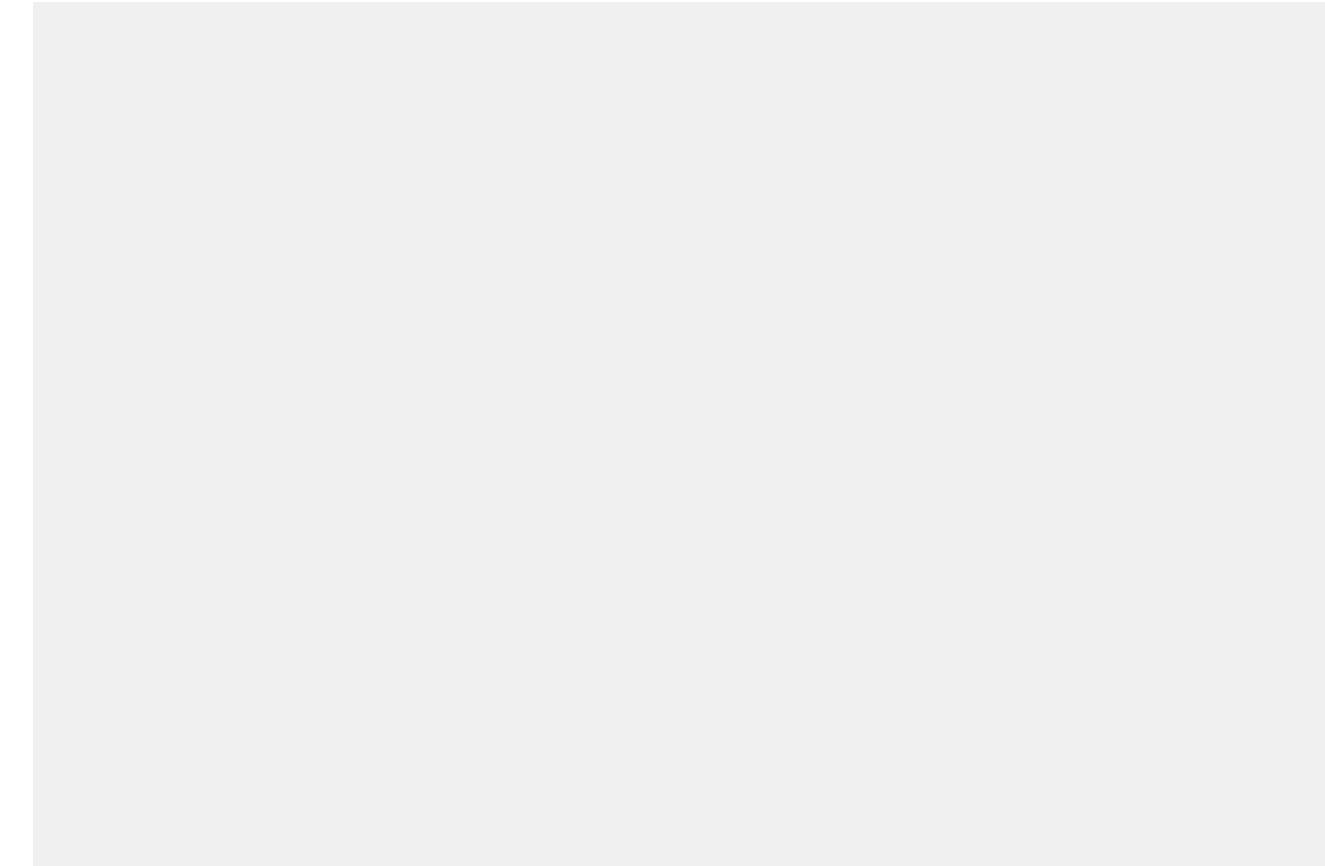
でアクセスされたレスポンスボディ内の XML または JSON コンテンツを解析するには、[DOM  
クラス](#)または[JSON クラス](#)を使用します。

クラスには、次の公開メソッドが含まれます。

名前	引数	戻り値	説明
	string		応答で返された本文を取得します。最大 3 MB です。 HTTP 要求のサイズおよび応答のサイズは、ヒープサ イズの合計にカウントされます。
	Blob		応答で返された本文を Blob として取得します。最大 3 MB です。 HTTP 要求のサイズおよび応答のサイズは、ヒープサ イズの合計にカウントされます。
			応答で返された本文を DOM ドキュメントとして取得 します。次のショートカットとして使用できます。
	String key	String	応答ヘッダーの内容を取得します。
	String []		応答内に返されたヘッダーキーの配列を取得します。
	string		応答に返された状況メッセージを取得します。
	Integer		応答内に返された状況コードの値を取得します。

名前	引数	戻り値	説明
		XmlStreamReader	コーラルアウトレスポンスボディを解析する XmlStreamReader ( <a href="#">クラス</a> ) を返します。次のショートカットとして使用できます。
			完全な例は、 <a href="#">「 の例 」</a> を参照してください。
string	Void		応答で返された本文を指定します。
Blob	Void		Blob を使用して、応答で返された本文を指定します。
String <i>key</i> , String <i>value</i>	Void		応答ヘッダーの内容を指定します。
String	Void		応答で返された状況メッセージを指定します。
Integer	Void		応答で返された状況コードの値を指定します。
	string		次のような応答内に返された状況メッセージと状況コードを返します。

次の の例では、内容は外部 Web サーバから取得され、XML は を使  
用して解析されます。



## HTTP コールアウトのテスト

Apex をリリースまたはパッケージ化するには、コードのテストカバー率が 75% に達している必要があります。デフォルトでは、テストメソッドでは HTTP コールアウトはサポートされないため、コールアウトを実行するテストはスキップされます。ただし、[HTTPCalloutMock](#) をコールして次のいずれかの方法で擬似応答を指定し、テストで擬似応答を生成するように Apex に指示することで、HTTP コールアウトのテストを有効にできます。

- ・ [インターフェースの実装による方法](#)
- ・ [または](#) [の静的リソースを使用する方法](#)

テストメソッドで擬似コールアウトの前に DML 操作を実行できるようにするには、「[DML 操作と擬似コールアウトの実行](#)」を参照してください。

### `HttpCalloutMock` インターフェースの実装による HTTP コールアウトのテスト

インターフェースを実装して `call` メソッドで送信される応答を指定できるようにします。Apex ランタイムでこのメソッドをコールしてコールアウトへの応答を送信します。

```
YourHttpCalloutMockImpl implements HttpCalloutMock
```

 メモ:

- インターフェースを実装するクラスには、global と public のいずれかを使用できます。
- このクラスはテストコンテキストでのみ使用されるため、のアノテーションを付加できます。この方法で、3 MB の組織コードサイズ制限からクラスを除外できます。

擬似応答の値を指定したら、テストメソッドで をコールし、この擬似応答を送信するように Apex ランタイムに指示できます。次のように、第 1 引数では を渡し、第 2 引数では のインターフェース実装の新しいインスタンスを渡します。

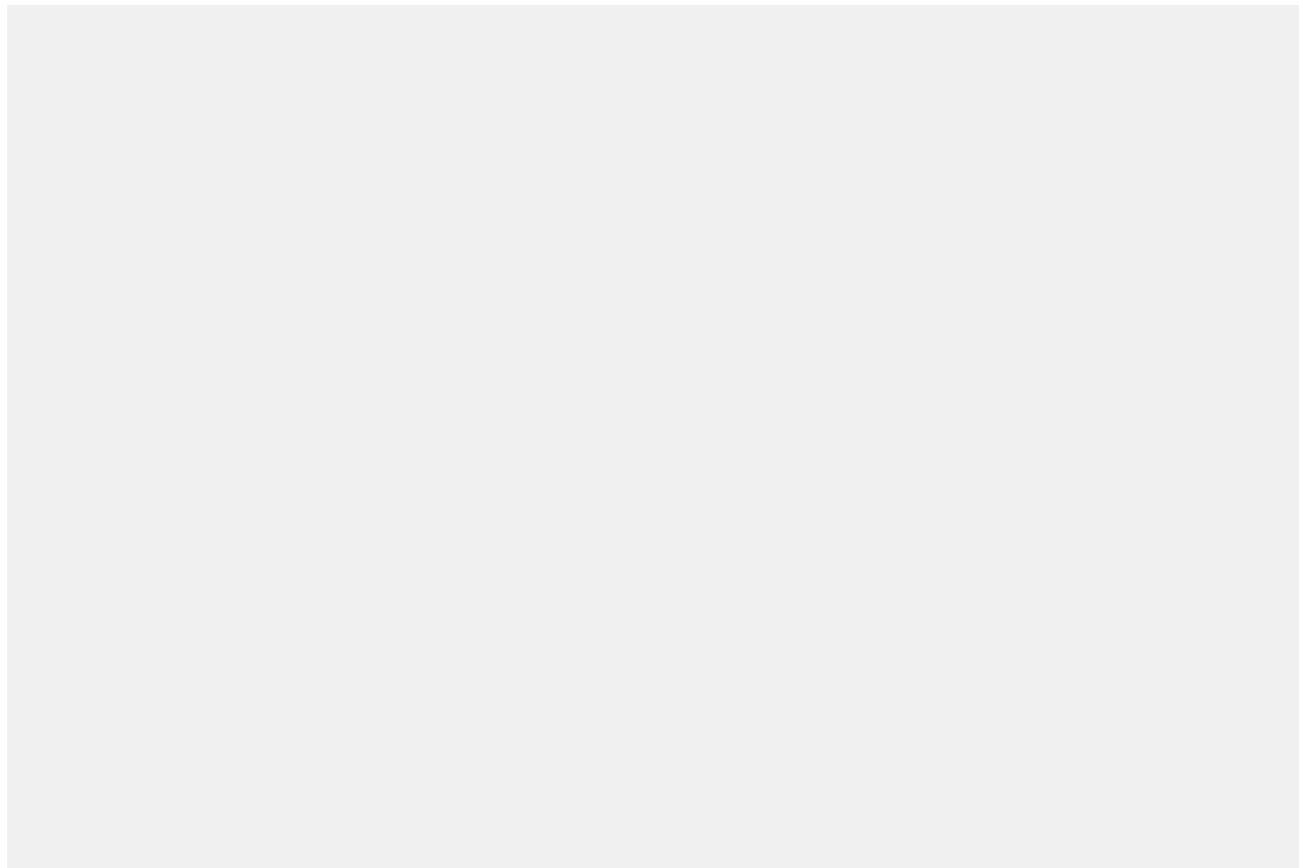
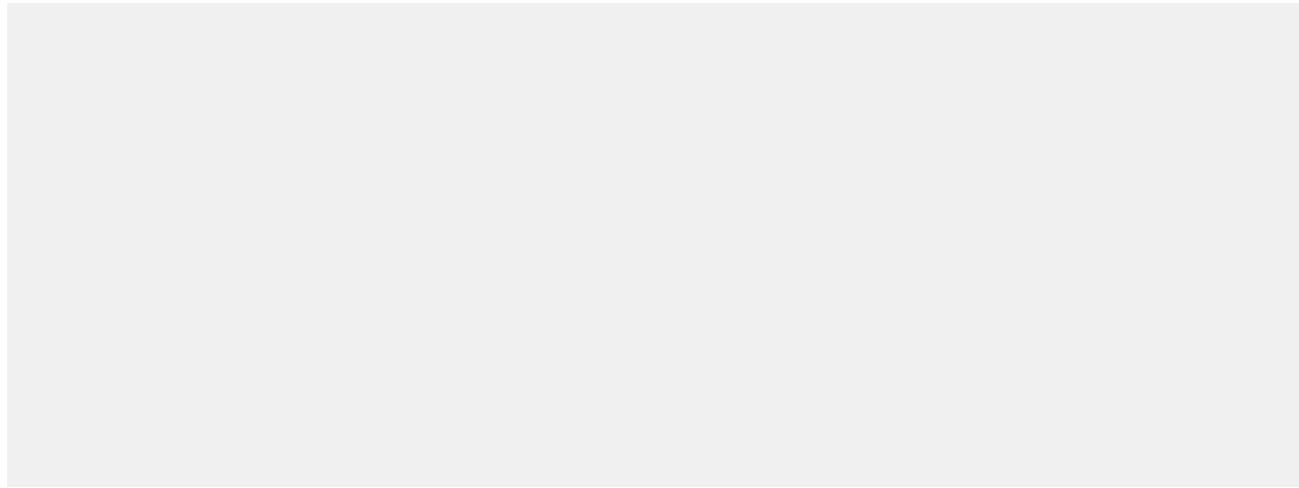
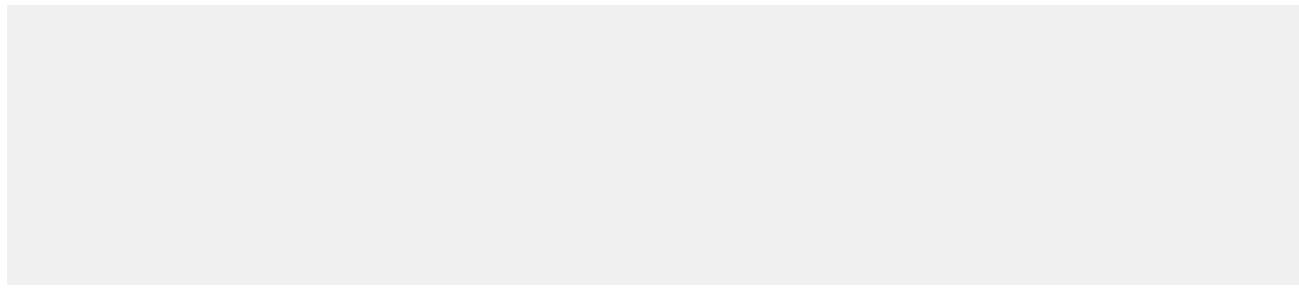
`YourHttpCalloutMockImpl`

これ以降で HTTP コールアウトがテストコンテキストで呼び出された場合、コールアウトは実行されず、`respond` メソッド実装で指定した擬似応答が受信されます。



メモ: コールアウトを実行するコードが管理パッケージに含まれる場合、同じパッケージ内のテストメソッドから同じ名前空間を使用して をコールし、擬似コールアウトを行う必要があります。

次の詳細な例は、HTTP コールアウトのテスト方法を示しています。インターフェースの実装 ( ) が最初に記述されています。その後に、テストメソッドを含むクラスと、テストでコールするメソッドを含む別のクラスが続きます。 テストメソッドは、 をコールする前に を設定します。次に、返された応答が、実装された メソッドで送信した内容と同じであることを確認します。各クラスを個別に保存して、 でテストを実行します。



## 関連リンク

[静的リソースを使用した HTTP コールアウトのテスト](#)

[HttpCalloutMock インターフェース](#)

### 静的リソースを使用した HTTP コールアウトのテスト

受信するレスポンスボディを静的リソース内に指定し、2つの組み込みクラス（[StaticResourceCalloutMock](#)）のいずれかを使用することで、HTTP コールアウトをテストできます。

### StaticResourceCalloutMock を使用した HTTP コールアウトのテスト

Apex には、静的リソースでレスポンスボディを指定することでコールアウトのテストに使用できる、組み込みクラスが用意されています。このクラスを使用する場合、インターフェースを独自に実装する必要ありません。代わりに、単にインスタンスを作成し、レスポンスボディに使用する静的リソースを応答の他のプロパティ（状況コードやコンテンツタイプなど）と共に設定します。

最初に、レスポンスボディを含めるテキストファイルから静的リソースを作成する必要があります。

1. 返すレスポンスボディを含むテキストファイルを作成します。レスポンスボディには任意の文字列を使用できますが、コンテンツタイプを指定した場合はそのタイプと一致する必要があります。たとえば、応答にコンテンツタイプを指定しない場合、ファイルには任意の文字列 を含めることができます。応答に application/json のコンテンツタイプを指定した場合、ファイルコンテンツは {"hah":"fooled you"} などの JSON 文字列にする必要があります。
2. このテキストファイル用の静的リソースを作成します。
  - a. [開発] > [静的リソース] をクリックして、[新規静的リソース] をクリックします。
  - b. 静的リソースに名前を付けます。
  - c. アップロードするファイルを選択します。
  - d. [保存] をクリックします。

静的リソースについての詳細は、Salesforce オンラインヘルプの「静的リソースの定義」を参照してください。

次に、 のインスタンスを作成し、静的リソースとその他のプロパティを設定します。

テストメソッドで、をコールして擬似コールアウトモードを設定し、最初の引数として  
、2番目の引数として用に作成した変数名を渡します。

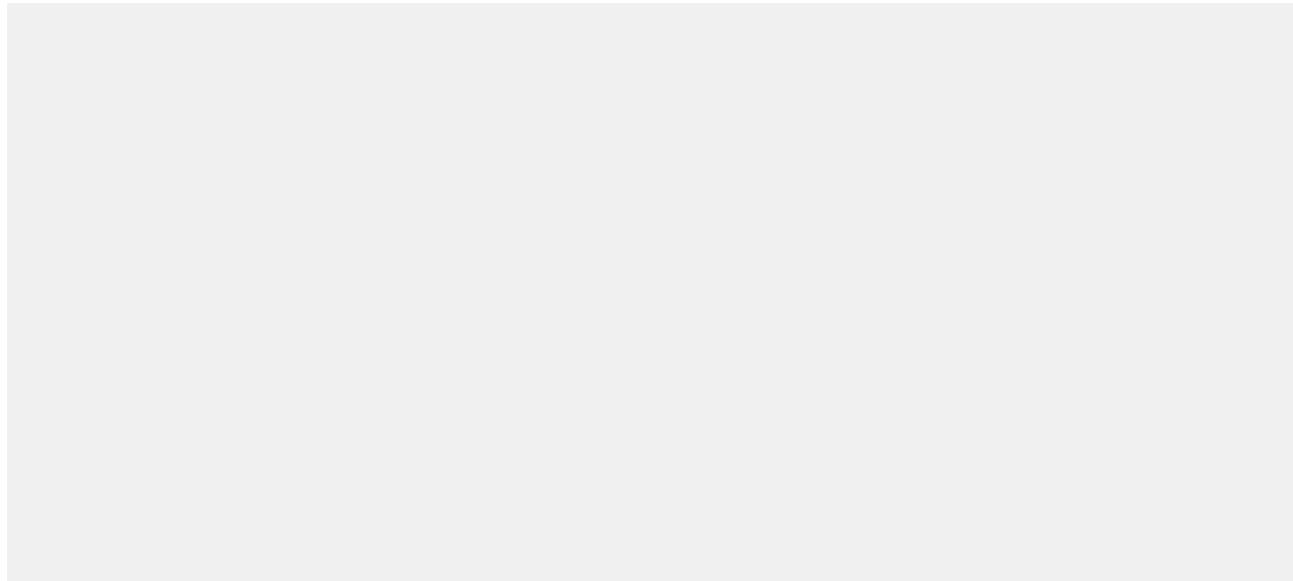
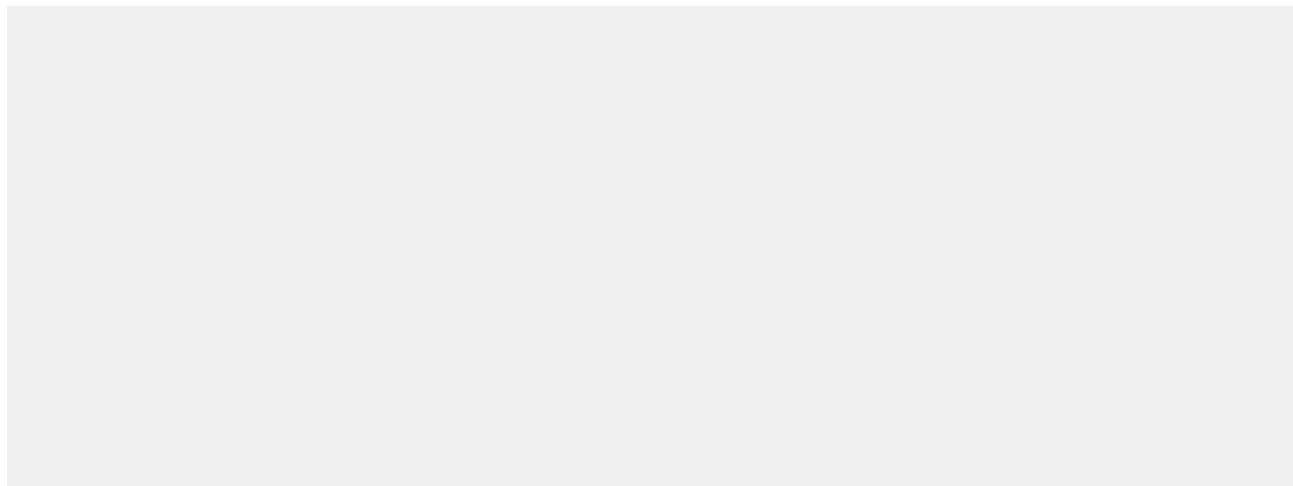
*mock*

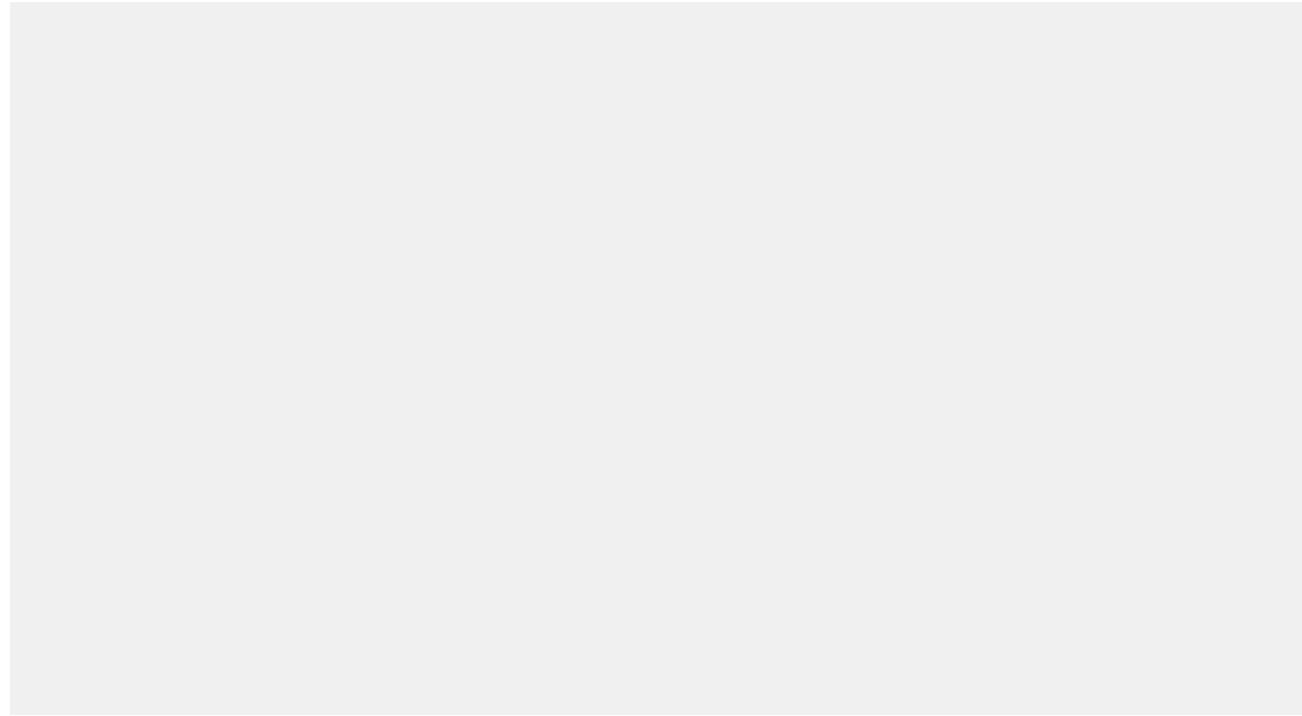
これ以降テストメソッドでコールアウトを実行しようとすると、コールアウトは実行されず、Apex ランタイムがのインスタンスで指定した擬似応答を送信します。



メモ: コールアウトを実行するコードが管理パッケージに含まれる場合、同じパッケージ内のテストメソッドから同じ名前空間を使用してをコールし、擬似コールアウトを行う必要があります。

次の詳細な例には、テストメソッド()が含まれ、このメソッドで、コールアウトを実行するをテストします。この例を実行する前に、コンテンツを含むテキストファイルに基づいたという名前の静的リソースを作成します。各クラスを個別に保存して、でテストを実行します。



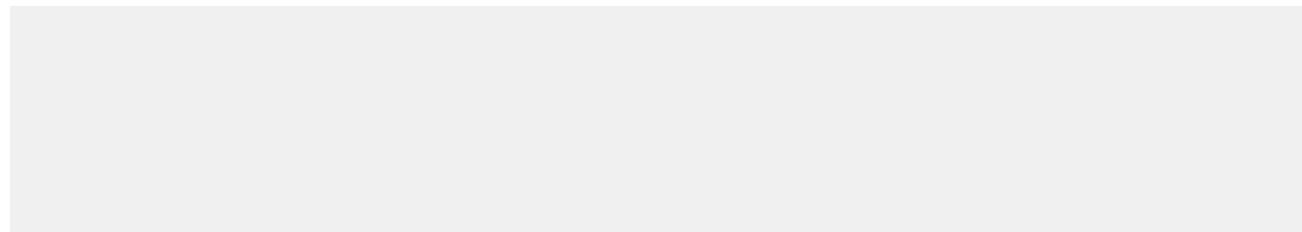


#### **MultiStaticResourceCalloutMock を使用した HTTP コールアウトのテスト**

Apex には、各エンドポイントの静的リソースでレスポンスボディを指定することでコールアウトのテストに使用できる、組み込み  
クラスが用意されています。このクラスは、複数のレスポンスボディを指定できること以外は  
と似ています。このクラスを使用する場合、インターフェースを独自に実装する必要ありません。代わりに、単に  
のインスタンスを作成し、エンドポイントごとに使用する静的リソースを設定します。状況コードやコンテンツタイプなどの応答の他のプロパティも設定できます。

最初に、レスponsBody を含めるテキストファイルから静的リソースを作成する必要があります。  
[「\*\*使用した HTTP コールアウトのテスト\*\*」](#)に示された手順を参照してください。

次に、  
のインスタンスを作成し、静的リソースとその他のプロパティを設定します。

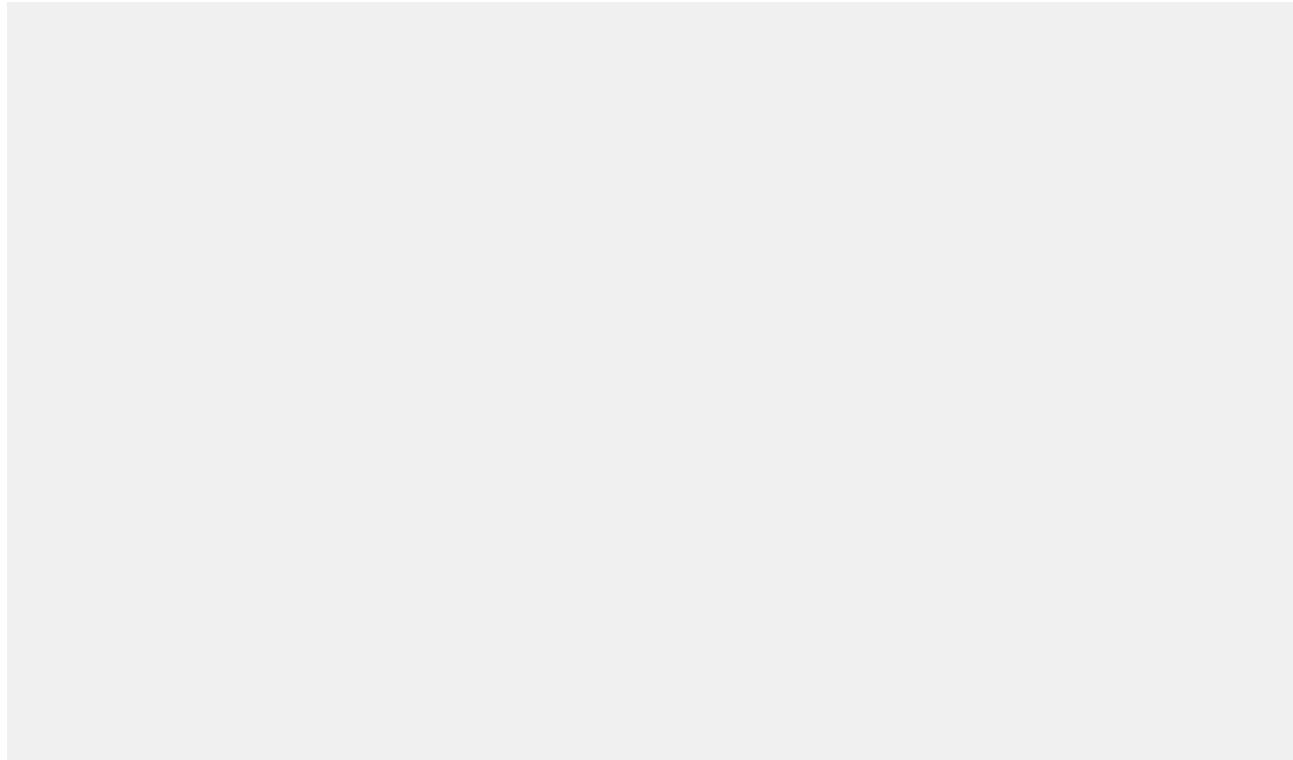
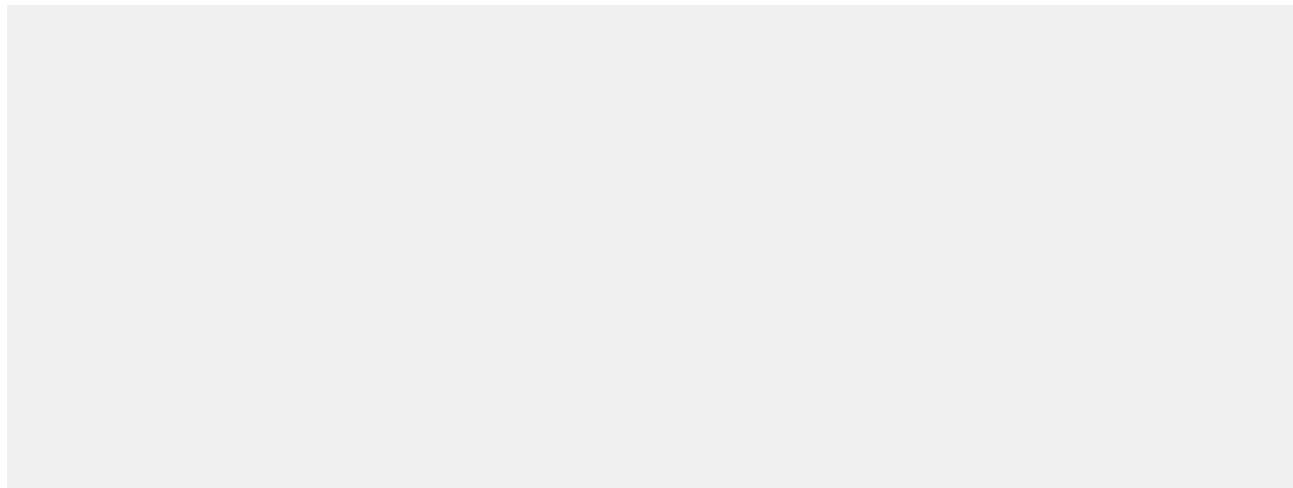


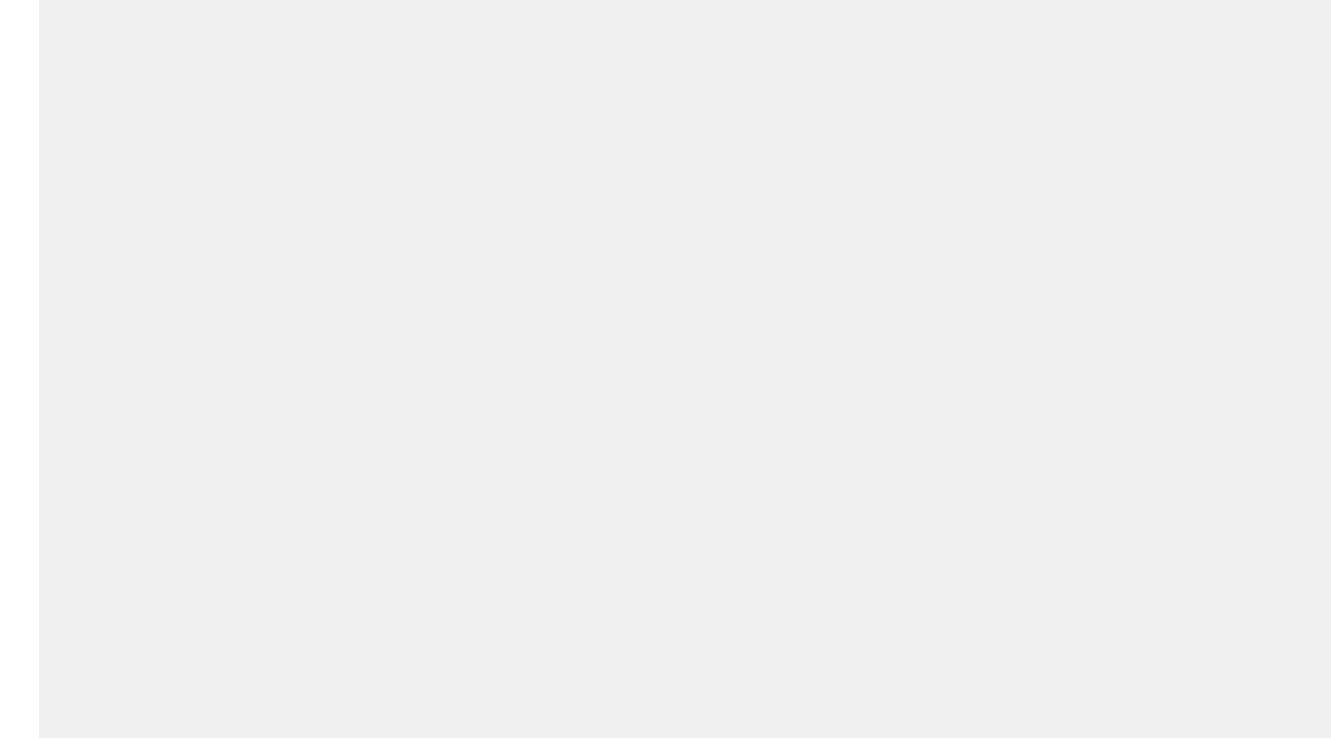
テストメソッドで、  
をコールして擬似コールアウトモードを設定し、最初の引数として  
、2番目の引数として  
用に作成した変数名を渡します。

*multimock*

これ以降テストメソッドで  
のいずれかのエンドポイントへの HTTP コールアウトを実行しようと  
すると、コールアウトは実行されず、Apex ランタイムが  
のインスタンスで指定した対応する擬似応答を送信します。

次の詳細な例には、テストメソッド()  
で、コールアウトを実行する  
)が含まれ、このメソッド  
を含むテキストファイルに基づいた  
をテストします。この例を実行する前に、コンテ  
ンツ  
と、コンテンツ  
という名前の静的リソース  
名前の静的リソースを作成します。各クラスを個別に保存して、  
でテストを実行します。





## 関連リンク

[HttpCalloutMock インターフェースの実装による HTTP コールアウトのテスト](#)

[MultiStaticResourceCalloutMock メソッド](#)

[StaticResourceCalloutMock メソッド](#)

## DML 操作と擬似コールアウトの実行

デフォルトでは、同じトランザクション内で DML 操作の後にコールアウトを実行することは許可されません。これは DML 操作によって、コミットされていない待機中の作業が発生してコールアウトの実行が妨げられるためです。場合によっては、コールアウトを行う前に、DML を使用してテストメソッドにテストデータを挿入する必要が生じことがあります。これを行うには、コールアウトを実行するコード部分を

と

ステートメントの間に配置します。ステートメントは、ステートメントは、  
トメントの前に配置する必要があります。また、DML 操作のコールは、/ ブ  
ロックの一部にすることはできません。

擬似コールアウト後の DML 操作は許可されており、テストメソッドでの変更は必要ありません。

DML 操作のサポートは、

インターフェースおよび静的リソース(

または

)を使用することで、疑似コールアウトのすべての実装で動作しま

す。次の例では、実装された  
使用するときにも適用できます。

インターフェースを使用しますが、同じ方法を静的リソースを

## 擬似コールアウト前の DML の実行

この例は、前の [HttpCalloutMock](#) の例に基づいています。この例では、  
ステートメントを使用して、テストメソッドで疑似コールアウトの前に DML 操作を実行できるようにします。  
テストメソッド( )は最初にテスト取引先を挿入し、  
を使用して疑似コールアウトモードを設定して、コールアウトを実行するメソッドをコールし、  
疑似応答値を確認します。最後に、  
をコールします。

```
// Perform some DML to insert test data
Account testAcct = new Account('Test Account');
insert testAcct;

// Call Test.startTest before performing callout
// but after setting test data.

Test.startTest();

// Set mock callout class
Test.setMock(HttpCalloutMock.class, new MockHttpResponseGenerator());

Test.stopTest();
```

## 非同期 Apex と擬似コールアウト

DML と同様に、非同期 Apex 操作では、コミットされていない待機中の作業によって、同じトランザクションの後の方でコールアウトの実行が妨げられる結果になります。非同期 Apex 操作の例としては、future メソッド、Apex 一括処理、スケジュール済み Apex のコールがあります。通常、これらの非同期コールは、  
の後で実行されるようにするため、テストメソッドで **と** **ステートメント間に** 配置します。この場合、擬似コールアウトは非同期コールの後で実行できるため、変更は不要です。ただし、非同期コールが **と** **ステートメント間に** 配置されていない場合は、コミットされていない待機中の作業のため例外が発生します。この例外を回避するには、次のいずれかを行います。

- 非同期コールを **と** **ステートメント間に** 配置する。

- DML コールと同じルールに従う。つまり、コールアウトを実行するコード部分を **と** **ステートメント間に** 配置します。  
ステートメントは、  
テストメントの前に配置する必要があります。また、非同期コールは、  
ロックの一部にすることはできません。

擬似コールアウト後の非同期コールは許可されており、テストメソッドでの変更は必要ありません。

## 関連リンク

[コールアウトの制限事項](#)

## Crypto クラス

クラスのメソッドは、ダイジェスト、メッセージ認証コード、署名の作成、および情報の暗号化び復号化を行うための標準アルゴリズムを提供します。これらは、Force.com のコンテンツのセキュリティを確保したり、Google、Amazon WebServices (AWS) などの外部サービスと統合するために使用できます。

名前	引数	戻り値	説明
	String <i>algorithmName</i> Blob <i>privateKey</i> Blob <i>initializationVector</i> Blob <i>cipherText</i>	Blob	<p>指定アルゴリズム、非公開鍵、および初期化ベクトルを使用して blob <i>cipherText</i> を復号化します。このメソッドを使用して、サードパーティアプリケーションまたは メソッドにより暗号化された blob を復号化します。</p> <p><i>algorithmName</i> の有効値は次のとおりです。</p> <ul style="list-style-type: none"> <li>• AES128</li> <li>• AES192</li> <li>• AES256</li> </ul> <p>さまざまなサイズの鍵を使用するすべての業界標準の Advanced Encryption Standard (AES) アルゴリズムがあります。これらのアルゴリズムでは、暗号解読ブロックチェーン (CBC) および PKCS5 パディングを使用します。</p> <p><i>privateKey</i> の長さは指定のアルゴリズム (128 ビット、192 ビット、または 256 ビット) に一致する必要があります。長さは、それぞれ、16、24、32 バイトです。サードパーティアプリケーションを使用するか、 メソッドを使用して、自分用にこの鍵を生成します。</p> <p>初期化ベクトルは 128 ビット (16 バイト) である必要があります。</p> <p>「<a href="#">暗号化および復号化の例</a>」を参照してください。</p> <p>実行中に発生する可能性のある例外についての詳細は、「<a href="#">暗号化および復号化の例外</a>」を参照してください。</p>
	String <i>algorithmName</i> Blob <i>privateKey</i>	Blob	<p>指定アルゴリズム、非公開鍵を使用して blob <i>IVAndCipherText</i> を復号化します。このメソッドを使用して、サードパーティアプリケーションまたは メソッドにより暗号化された blob を復号化します。</p> <p><i>algorithmName</i> の有効値は次のとおりです。</p>

名前	引数	戻り値	説明
	Blob <i>IVAndCipherText</i>	<ul style="list-style-type: none"> <li>• AES128</li> <li>• AES192</li> <li>• AES256</li> </ul>	<p>さまざまなサイズの鍵を使用するすべての業界標準の Advanced Encryption Standard (AES) アルゴリズムがあります。これらのアルゴリズムでは、暗号解読ブロックチェーン (CBC) および PKCS5 パディングを使用します。</p> <p><i>privateKey</i> の長さは指定のアルゴリズム (128 ビット、192 ビット、または 256 ビット) に一致する必要があります。長さは、それぞれ、16、24、32 バイトです。サードパーティアプリケーションを使用するか、メソッドを使用して、自分用にこの鍵を生成します。</p> <p><i>IVAndCipherText</i> の 128 ビット (16 バイト) には初期化ベクトルが含まれている必要があります。</p> <p>「<a href="#">暗号化および復号化の例</a>」を参照してください。</p> <p>実行中に発生する可能性のある例外についての詳細は、「<a href="#">暗号化および復号化の例外</a>」を参照してください。</p>
	String <i>algorithmName</i> Blob <i>privateKey</i> Blob <i>initializationVector</i> Blob <i>clearText</i>	Blob	<p>指定アルゴリズム、非公開鍵、および初期化ベクトルを使用して blob <i>clearText</i> を暗号化します。独自の初期化ベクトルを指定する場合、このメソッドを使用します。初期化ベクトルは 128 ビット (16 バイト) である必要があります。サードパーティアプリケーションまたは メソッドのいずれかを使用して、このメソッドにより暗号化された blob を復号化します。Salesforce で初期化ベクトルが生成されるようにする場合は、メソッドを使用します。</p> <p>暗号化 blob の最初の 128 ビット (16 バイト) として格納されます。</p> <p><i>algorithmName</i> の有効値は次のとおりです。</p> <ul style="list-style-type: none"> <li>• AES128</li> <li>• AES192</li> <li>• AES256</li> </ul> <p>さまざまなサイズの鍵を使用するすべての業界標準の Advanced Encryption Standard (AES) アルゴリ</p>

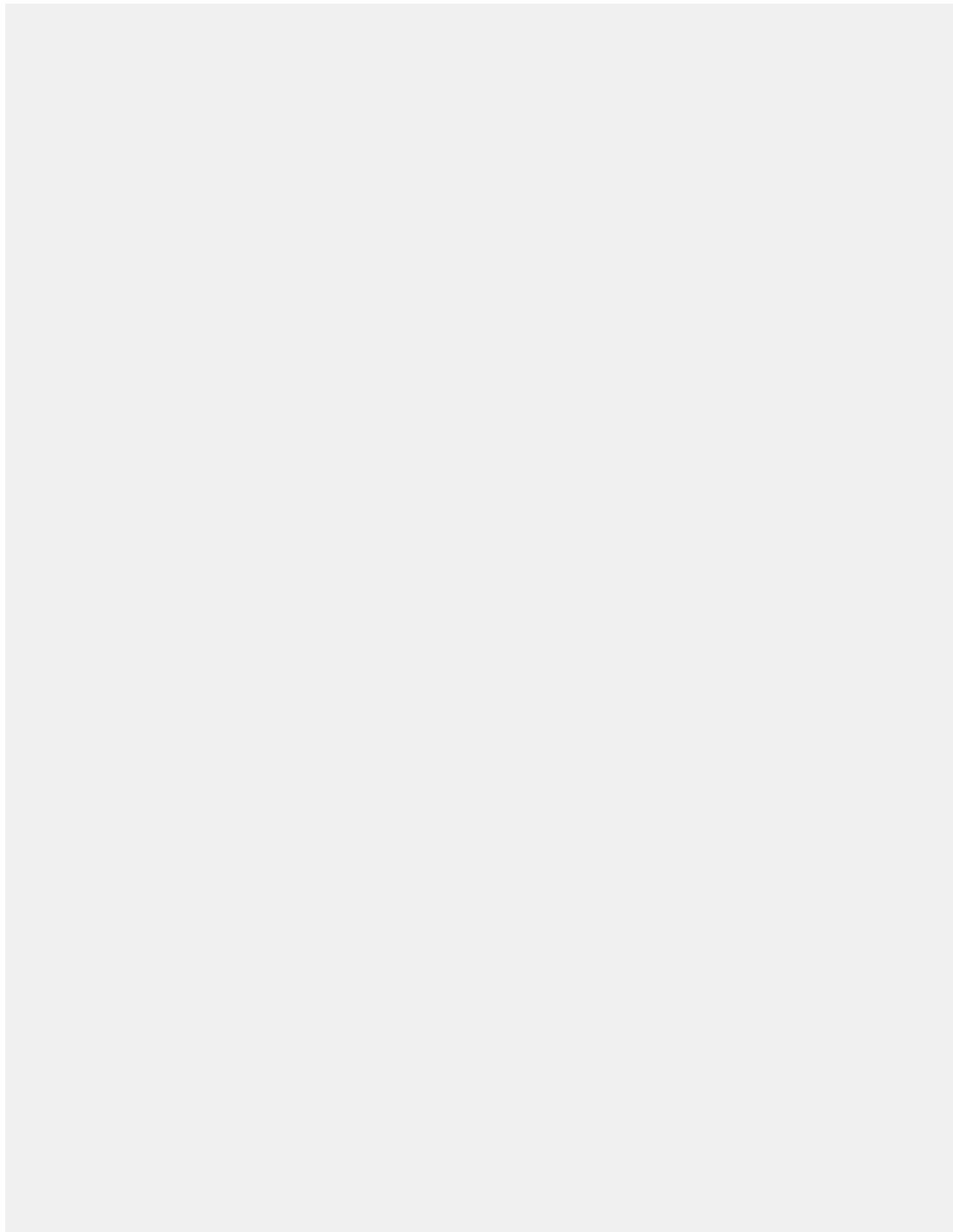
名前	引数	戻り値	説明
			<p>ズムがあります。これらのアルゴリズムでは、暗号解読ブロックチェーン (CBC) および PKCS5 パディングを使用します。</p> <p><i>privateKey</i> の長さは指定のアルゴリズム (128 ビット、192 ビット、または 256 ビット) に一致する必要があります。長さは、それぞれ、16、24、32 バイトです。サードパーティアプリケーションを使用するか、<a href="#">メソッド</a>を使用して、自分用にこの鍵を生成します。</p> <p>「<a href="#">暗号化および復号化の例</a>」を参照してください。</p> <p>実行中に発生する可能性のある例外についての詳細は、「<a href="#">暗号化および復号化の例外</a>」を参照してください。</p>
	String <i>algorithmName</i> Blob <i>privateKey</i> Blob <i>clearText</i>	Blob	<p>指定アルゴリズム、非公開鍵を使用して blob <i>clearText</i> を暗号化します。Salesforce で初期化ベクトルが生成されるようにする場合は、このメソッドを使用します。暗号化 blob の最初の 128 ビット (16 バイト) として格納されます。サードパーティアプリケーションまたはメソッドのいずれかを使用して、このメソッドにより暗号化された blob を復号化します。独自の初期化ベクトルを生成する場合は、<a href="#">メソッド</a>を使用します。</p> <p><i>algorithmName</i> の有効値は次のとおりです。</p> <ul style="list-style-type: none"> <li>• AES128</li> <li>• AES192</li> <li>• AES256</li> </ul> <p>さまざまなサイズの鍵を使用するすべての業界標準の Advanced Encryption Standard (AES) アルゴリズムがあります。これらのアルゴリズムでは、暗号解読ブロックチェーン (CBC) および PKCS5 パディングを使用します。</p> <p><i>privateKey</i> の長さは指定のアルゴリズム (128 ビット、192 ビット、または 256 ビット) に一致する必要があります。長さは、それぞれ、16、24、32 バイトです。サードパーティアプリケーションを使用するか、<a href="#">メソッド</a>を使用して、自分用にこの鍵を生成します。</p>

名前	引数	戻り値	説明
			<p>「暗号化および復号化の例」を参照してください。</p> <p>実行中に発生する可能性のある例外についての詳細は、「暗号化および復号化の例外」を参照してください。</p>
	Integer <i>size</i>	Blob	<p>Advanced Encryption Standard (AES) 鍵を生成します。<i>size</i>を使用してビット単位のキーのサイズを指定します。有効な値は、次のとおりです。</p> <ul style="list-style-type: none"> <li>• 128</li> <li>• 192</li> <li>• 256</li> </ul>
	String <i>algorithmName</i>	Blob	<p>供給された入力文字列とアルゴリズム名に基づいた安定した一方向のハッシュダイジェストを計算します。<i>algorithmName</i>の有効値は次のとおりです。</p> <ul style="list-style-type: none"> <li>• MD5</li> <li>• SHA1</li> <li>• SHA-256</li> <li>• SHA-512</li> </ul>
	String <i>algorithmName</i>	Blob	<p>秘密鍵と指定アルゴリズムを使用して、入力文字列用のメッセージ認証コード(MAC)を計算します。<i>algorithmName</i>の有効値は次のとおりです。</p> <ul style="list-style-type: none"> <li>• hmacMD5</li> <li>• hmacSHA1</li> <li>• hmacSHA256</li> <li>• hmacSHA512</li> </ul>
	Blob <i>input</i>	Blob	<p><i>privateKey</i>の値は復号化形式である必要はありません。値は4 KB を超えることはできません。</p>
		Integer	ランダムな integer を返します。
		Long	ランダムな long を返します。
	String <i>algorithmName</i>	Blob	<p>供給された秘密鍵と指定アルゴリズムを使用して、入力文字列用の固有のデジタル署名を計算します。<i>algorithmName</i>用の有効値は、または です。両方の値とも、同じアルゴリズムを表します。</p>

名前	引数	戻り値	説明
	Blob <i>privateKey</i>		<p><i>privateKey</i> の値は メソッドを使用して復号化し、RSA の PKCS #8 (1.2) Private-Key Information Syntax Standard 形式で なければなりません。値は4KBを超えることはで きません。</p> <p>次のスニペットは、宣言と初期化の例を示します。</p> <pre>pkcs8 format private key</pre>

### Amazon WebService のインテグレーションの例

次の例は、Salesforce を伴った Amazon WebServices の統合を示しています。



### 暗号化および復号化の例

次の例では、  
メソッドを使用します。

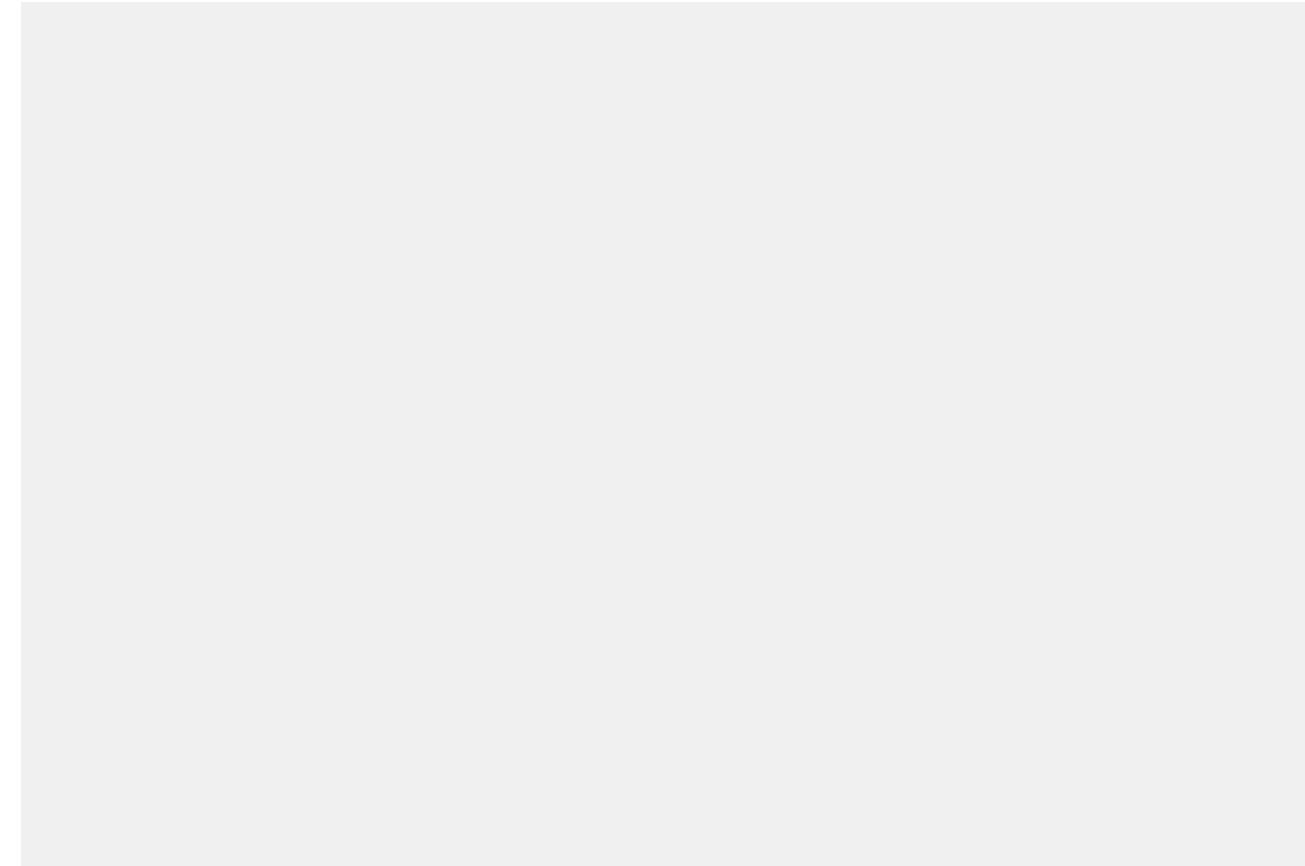
メソッドおよび

メソッド、および

次は、

メソッドおよび

メソッドの単体テストの作成例です。



### 暗号化および復号化の例外

次のメソッドで次の例外が発生する可能性があります。

- 
- 
- 
- 

例外	メッセージ	説明
InvalidParameterValue	暗号化データの初期化ベクトルは解析できません。	管理初期化ベクトルを使用している場合で、暗号解読テキストが 16 バイト未満の場合に発生します。
InvalidParameterValue	無効なアルゴリズム <code>algoName</code> 。AES128、AES192、または AES256 である必要があります。	アルゴリズム名が有効な値の 1 つでない場合に発生します。
InvalidParameterValue	無効な非公開鍵。 <code>size</code> バイトである必要があります。	非公開鍵のサイズが指定のアルゴリズムに一致しない場合に発生します。
InvalidParameterValue	無効な初期化ベクトル。16 バイトである必要があります。	初期化ベクトルが 16 バイトでない場合に発生します。

例外	メッセージ	説明
InvalidParameterValue	無効なデータ。入力データは <code>size</code> バイトです。1048576 バイトの制限を超えています。	データが 1 MB を超える場合に発生します。復号化の場合、初期化ベクトルヘッダーでは 1048608 バイトが許容されます。さらに、ブロックサイズに合わせて暗号にパディングを追加できます。
NullPointerException	引数は null (空白) にできません。	必要なメソッドの引数の 1 つが null である場合に発生します。
SecurityException	所定の最終ブロックが適切にパディングされていません。	暗号化または復号化でデータが適切にブロック整列されていないか、同様の問題が発生している場合に発生します。
SecurityException	さまざまなメッセージ	暗号化か復号化時に問題が発生している場合に発生します。

## EncodingUtil クラス

URL 文字列を符号化し、復号化し、文字列を 16 進法の形式に変換するには、  
使用します。

クラスのメソッド

名前	引数	戻り値	説明
	<code>String inputString</code>	<code>Blob</code>	Base64 の符号化された string をその標準フォームを表している blob に変換します。
	<code>Blob inputBlob</code>	<code>string</code>	blob をその標準フォームを表している符号化されていない string に変換します。
	<code>Blob inputString</code>	<code>string</code>	<code>inputString</code> の 16 進法 (16 進数) 表現を返します。このメソッドは、HTTP ダイジェスト認証(RFC2617)のためにクライアント応答(たとえば HA1 または HA2)を計算するために使用可能です。
	<code>String inputString</code>	<code>String encodingScheme</code>	特定の符号化方式を使用している形式、たとえば "UTF-8" を復号化します。どの文字が連続シーケンスによって表されているかを決定するために、このメソッドは供給された符号化方式を使用します。形式についての詳細は、 <i>Hypertext Markup Language - 2.0</i> 内の「The form-urlencoded Media Type」を参照してください。
	<code>String inputString</code>	<code>String encodingScheme</code>	特定の符号化方式を使用している形式、たとえば

名前	引数	戻り値	説明
			<p>"UTF-8" に符号化します。不確かな文字用のバイトを得るために、このメソッドは供給された符号化方式を使用します。形式についての詳細は、<i>Hypertext Markup Language - 2.0</i> 内の「<a href="#">The form-urlencoded Media Type</a>」を参照してください。</p> <p>例:</p> <pre>url</pre>



メモ: 非 ASCII 文字を含む文書を Salesforce に移動するために EncodingUtil メソッドを使用することはできません。ただし、Salesforce から文書をダウンロードできます。その場合、API 呼び出しを使用して文書の ID をクエリし、ID によって文書を要求してください。

HTTP ダイジェスト認証 (RFC2617) 用のクライアント応答を計算するための

の使用方法を、次

に例示します。

```
myData
```

## ナレッジ管理の公開サービスクラス

### 使用方法

記事とその翻訳のライフサイクルで次の部分を管理するには、ソッドを使用します。

クラスのメ

- ・ 公開
- ・ 更新
- ・ 取得
- ・ 削除
- ・ 翻訳の申請

- 翻訳を完了または未完了の状況に設定
- アーカイブ
- ドラフト記事または翻訳のレビュータスクの割り当て



メモ: 日付値は、GMT に基づきます。

このクラスのメソッドを使用するには、Salesforce ナレッジを有効にする必要があります。Salesforce ナレッジの設定についての詳細は、『[Salesforce Knowledge Implementation Guide](#)』を参照してください。

## メソッド

次に、

クラスの静的メソッドを示します。

メソッド	引数	戻り値	説明
	String <i>articleId</i> Datetime <i>scheduledDate</i>	Void	記事のオンラインバージョンをアーカイブします。 <i>scheduledDate</i> が null の場合、記事は即時にアーカイブされます。それ以外の場合、記事は予定日にアーカイブされます。
			<i>Insert article ID</i>
	String <i>articleId</i> String <i>assigneeID</i> String <i>instructions</i> Datetime <i>dueDate</i> Boolean <i>sendEmailNotification</i>	Void	ドラフト記事に関連するレビューtaskを割り当てます。
			<i>Insert article ID</i>
			<i>Insert assignee</i>



メソッド	引数	戻り値	説明
	String <i>articleId</i> Integer <i>versionNumber</i>	Void	アーカイブされた記事の特定のバージョンを削除します。  <i>Insert article ID</i>
	String <i>articleId</i>	Void	ドラフト記事を削除します。  <i>Insert article ID</i>
	String <i>articleVersionId</i>	Void	ドラフト翻訳を削除します。  <i>Insert article version ID</i>
	String <i>articleId</i>	String	アーカイブされたマスタバージョンからドラフト記事を作成し、記事の新しいドラフトマスタバージョン ID を返します。  <i>Insert article ID</i>
	String <i>articleId</i> Boolean <i>unpublish</i>	String	オンラインバージョンからドラフト記事を作成し、記事の新しいドラフトマスタバージョン ID を返します。さらに、 <i>unpublish</i> がに設定されている場合は、オンライン記事の公開を解除します。  <i>Insert article ID</i>

メソッド	引数	戻り値	説明
	String <i>articleId</i> String <i>language</i> Boolean <i>unpublish</i>	String	特定の言語のオンライン翻訳のドラフトバージョンを作成し、記事の新しいドラフトマスター バージョン ID を返します。さらに、に設定されている場合は、記事の公開を解除します。
			<i>Insert article</i>
			<i>ID</i>
	String <i>articleId</i> Boolean <i>flagAsNew</i>	Void	記事を公開します。 <i>flagAsNew</i> が に設定されている場合は、記事をメジャーバージョンとして公開します。
			<i>Insert article</i>
			<i>ID</i>
	String <i>articleId</i> Integer <i>versionNumber</i>	String	既存のオンライン記事の指定されたアーカイブバージョンに基づいて、その記事からドラフト記事を作成し、記事のバージョン ID を返します。
			<i>Insert article</i>
			<i>ID</i>
	String <i>articleId</i> Datetime <i>scheduledDate</i>	Void	メジャーバージョンとして記事の公開をスケジュールします。 <i>scheduledDate</i> が null の場合、記事は即時に公開されます。
			<i>Insert article</i>
			<i>ID</i>

メソッド	引数	戻り値	説明
	String <i>articleVersionId</i>	Void	公開準備完了のドラフト翻訳を「処理中」状況に戻します。  <i>Insert article version ID</i>
	String <i>articleId</i> String <i>language</i> String <i>assigneeID</i> Datetime <i>dueDate</i>	String	指定された言語への記事の翻訳を申請します。 さらに、指定されたユーザと期日も申請に割り当て、ドラフト翻訳の新しいIDを返します。  <i>Insert article ID</i>  <i>Insert assignee ID</i>

## Network クラス

コミュニティを表します。

### 使用方法

ユーザが現在ログインしているコミュニティを判断するには、クラスのメソッドを使用します。

 メモ: Salesforce コミュニティは、パイロットプログラムで利用可能です。組織がコミュニティパイロットを利用する条件を満たしているかどうかは、Salesforce.com の担当者にお問い合わせください。

### メソッド

クラスには、次の静的メソッドが含まれます。このメソッドは引数を取りません。

メソッド	戻り値	説明
	String	ユーザの現在のコミュニティを返します。コミュニティがユーザの組織で有効になっていないか、

メソッド	戻り値	説明
		ユーザが現在内部組織に含まれる場合は、 <code>True</code> を返します。

## Pattern および Matcher クラス

正規表現とは、特定の構文を使用して他の文字列との一致を探すために使用する文字列です。Apex では、*Pattern* および *Matcher* クラスでの正規表現の使用をサポートしています。



メモ: Apex では、正規表現と同様に、Pattern と Matcher も Java での動作に基づいています。

を参

照してください。

## Pattern と Matcher の使用

Pattern とは正規表現をコンパイルしたものです。Pattern は、Matcher が文字列に対してマッチ処理を実行するのに使用します。次の図に示すように、多くの Matcher オブジェクトは同じ Pattern オブジェクトを共有します。

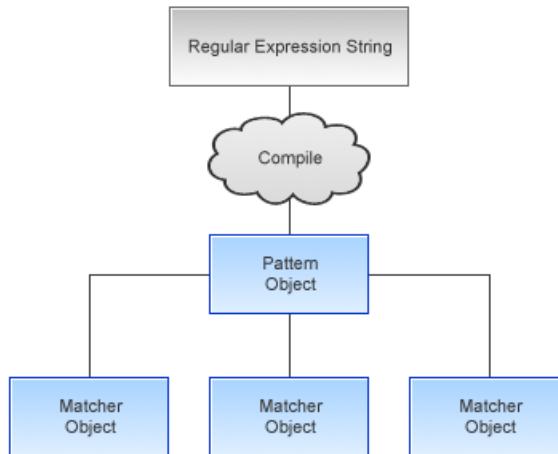


図 9: 多くの Matcher は同じ Pattern オブジェクトから作成します。

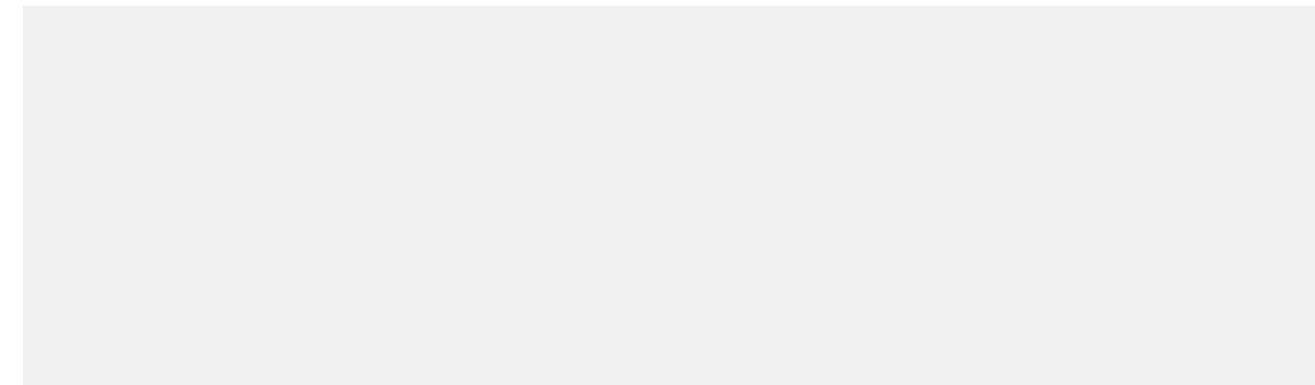
Apex の正規表現は、Java で使用される正規表現の標準に従っています。Java ベースのすべての正規表現文字列を簡単に Apex コードにインポートできます。



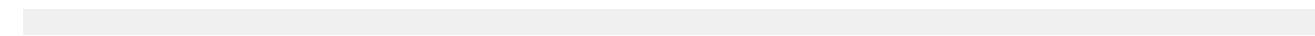
メモ: Salesforce では、正規表現の入力シーケンスにアクセスできる回数を 1,000,000 回に制限しています。その制限に達すると、ランタイムエラーが発生します。

すべての正規表現は文字列として指定されます。ほとんどの正規表現は、まず Pattern オブジェクトにコンパイルされます。String メソッドのみがコンパイルされていない正規表現を扱うことができます。

一般的に、正規表現を Pattern オブジェクトにコンパイルすると、Pattern オブジェクトが使用されるのは Matcher オブジェクト作成時の 1 回のみです。その他の処理は Matcher オブジェクトを使用して実行されます。次に例を示します。



正規表現を 1 回のみ使用する場合は、**Matcher** クラスの **find()** メソッドを使用すると、表現のコンパイルと文字列に対するマッチ処理を 1 回の呼び出して実行できます。たとえば、次のコードは上記のコードと同一です。



## リージョンの使用

Matcher オブジェクトは、リージョンという入力文字列のサブセットで一致を探します。Matcher オブジェクトのデフォルトリージョンは常に入力文字列全体です。ただし、**region()** メソッドを使用してリージョンの開始点と終了点を変更できます。リージョンの終了点は **endRegion()** および **end()** メソッドを使用してクエリを実行し取得できます。

メソッドには、start 値と end 値の両方が必要です。次の表は、一方の値のみを設定し、もう一方の値を設定しない例について示します。

リージョンの開始	リージョンの終了	コード例
明示的に指定	変更しない	
変更しない	明示的に指定	
デフォルト値にリセット	明示的に指定	

## マッチ処理の使用

Matcher オブジェクトは、Pattern を解釈し文字シーケンスに対するマッチ処理を実行します。

Matcher オブジェクトは、Pattern のメソッドにより Pattern 内でインスタンス化されます。一度作成すると、Matcher オブジェクトは次のタイプのマッチ処理の実行に使用できます。

- メソッドを使用した、パターンに対する Matcher オブジェクトの入力文字列全体の一致。
- メソッドを使用した、パターンに対する Matcher オブジェクトの入力文字列の一致。先頭から開始しますが、リージョン全体のマッチングは行いません。
- メソッドを使用した、パターンに一致する次のサブ文字列を検索するための Matcher オブジェクトの入力文字列のスキャン。

各メソッドは、成功または失敗を表す boolean を返します。

これらのメソッドのいずれかを使用した後に、次の Matcher クラスメソッドを使用して、前回の一致に関する詳細情報(検索されたものなど)を取得できます。

- : 一致があると、このメソッドは、一致文字列の中で最後の文字が一致した後ろの位置を返します。
- : 一致があると、このメソッドは一致文字列の中の最初の文字が一致した位置を返します。
- : 一致があると、このメソッドは一致したサブシーケンスを返します。

## 境界の使用

デフォルトでは、リージョンはアンカー付き境界で区切られています。つまり、リージョンの境界が入力文字列の先頭から末尾まで移動したとしても、ラインアンカー( または など)がリージョンの境界に一致します。

リージョンがアンカー付き境界を使うかどうかは メソッドで指定できます。デフォルトでは、リージョンは常にアンカー付き境界を使用します。 を 設定する場合、ラインアンカーは入力文字列の実際の末尾のみと一致します。

デフォルトでは、リージョンの外にあるすべてのテキストは検索されません。つまり、リージョンには不透明な境界があるということになります。ただし、透明な境界を使用すると、リージョン外にあるテキストを検索できます。透明な境界は、リージョン内に入力文字列全体が含まれていない場合のみ使用します。

メソッドを使用し、リージョンの境界のタイプを指定できます。

次の文字列の検索で、リージョンには「STRING」という単語しか含まれていないとします。

「cat」という単語の検索では、透明な境界が設定されていない限り一致しません。

## キャプチャグループについて

マッチ処理中、パターンと一致する入力文字列の各サブ文字列が保存されます。一致するサブ文字列のことをキャプチャグループと呼びます。

キャプチャグループは、左から右へ左かっここの数を数えて番号付けされます。たとえば、正規表現文字列では、キャプチャグループは 4 つあります。

- 1.
- 2.
- 3.
- 4.

グループ 0 は常に表現全体を表します。

グループに関連付けられたキャプチャされた入力は常に、最も最近一致したグループのサブ文字列です。このサブ文字列は、Matcher クラスのマッチ処理の 1 つが返した文字列です。

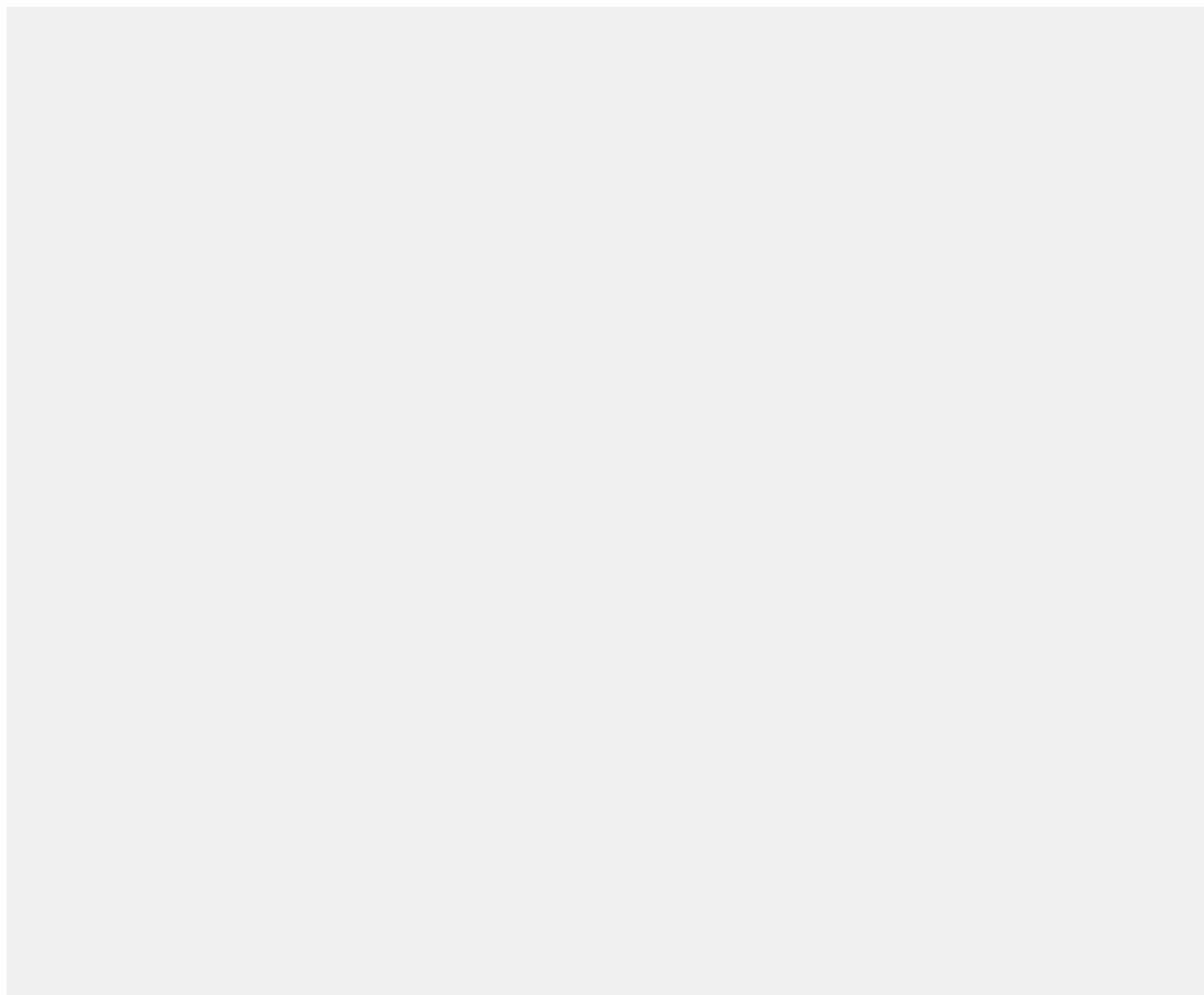
マッチ処理の 1 つを使用してグループを再度評価する場合、2 回目の評価が失敗すると、前に取得した値がある場合はその値が保持されます。

## Pattern と Matcher の例

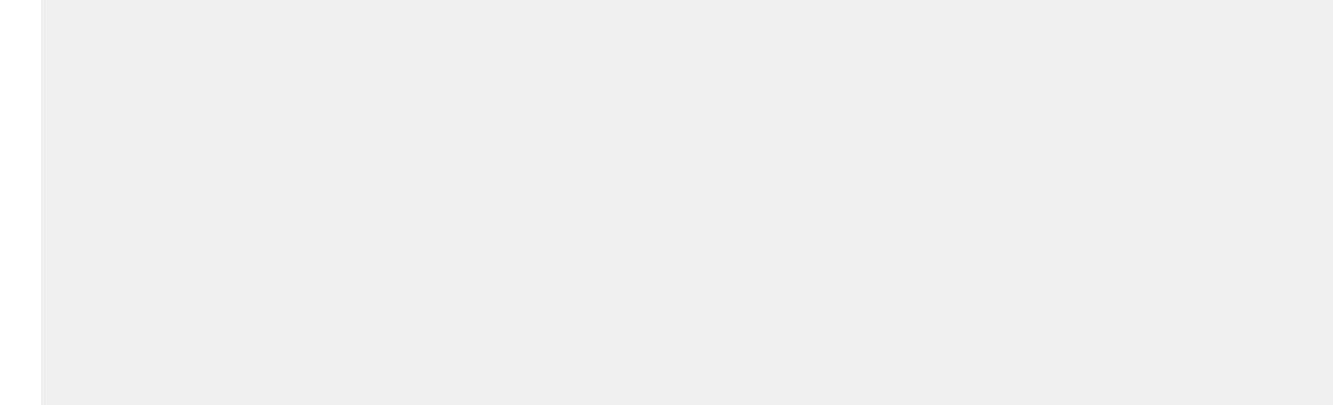
Matcher クラスの `endOfInput()` メソッドは、最後の文字が一致した後の一致文字列の位置を返します。これは、文字列を解析中に一致する部分が見つかった後、次の一致を見つけるなど別の処理を行う場合に使用します。

正規表現構文では、`\z` は 1 つ一致、または一致がないことを示し、`\Z` は 1 つ以上一致することを示します。

次の例では、Matcher オブジェクトと共に渡された文字列がパターンに一致します。これは、`\z` が、文字列 `"( )"` ( `" "` の後に `" "` が 1 回) に一致するためです。次に、最後の `\Z` ( `" "` の後に `\Z` が 1 つもない) に一致します。



次の例では、メールアドレスが正規化され、類似するメールアドレスに対して異なる最上位のメイン名やサブドメインがある場合、重複が報告されます。たとえば、  
は  
に正規化され  
ます。



## Pattern メソッド

次に、Pattern のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>regExp</i>	Pattern オブジェクト	正規表現を Pattern オブジェクトにコンパイルします。
	String <i>regExp</i> String <i>s</i>	boolean	正規表現 <i>regExp</i> をコンパイルして、 <i>s</i> に対するマッチ処理を実行します。文字列 <i>s</i> が正規表現に一致する場合は <code>true</code> を、それ以外は <code>false</code> を返します。  パターンを複数回使用する場合、一度コンパイルしてそれを再利用すれば、このメソッドを毎回起動するよりも効率的に処理できます。  次にコード例を示します。
	String <i>s</i>	string	リテラルパターンのように、文字列 <i>s</i> に一致するパターンを作成するのに使用する文字列を返します。入力文字列の <code>\\$</code> や <code>\n</code> などのメタキャラクタやエスケープシー

このコードは、次のコード例と同じ結果を生成します。

名前	引数	戻り値	説明
			ケンスは、特に意味のないリテラル文字として扱われます。

次に、Pattern のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	String <i>regExp</i>	Matcher オブジェクト	この Pattern オブジェクトに対し、入力文字列 <i>regExp</i> に一致する Matcher オブジェクトを作成します。
		string	この Pattern オブジェクトがコンパイルされた正規表現を返します。
	String <i>s</i>	String []	このパターンに一致する文字列のサブ文字列を含むリストを返します。 このサブ文字列は、文字列の中で発生した順序でリストに記述されます。 <i>s</i> がパターンに一致しない場合、結果リストには元の文字列を含む要素が 1 つだけ含まれます。
	String <i>regExp</i> Integer <i>limit</i>	String []	文字列の各サブ文字列を含むリストを返します。このサブ文字列は、このパターンに一致する正規表現 <i>regExp</i> 、または文字列の末尾に達したことのいずれかにより終了します。(省略可能) <i>limit</i> パラメータは、パターンが適用された回数を制御するため、リストの長さに影響を与えます。 <ul style="list-style-type: none"> <li>• <i>limit</i> が 0 より大きい場合、パターンは最大 <i>limit</i> - 1 回適用されたことになります。また、リストの長さは最大 <i>limit</i> となり、リストの最後のエントリには最後に一致した区切り文字移行のすべての入力が含まれます。</li> <li>• <i>limit</i> が正の値でない場合、パターンを何度も適用することが可能となり、リストの長さも任意となります。</li> <li>• <i>limit</i> が 0 の場合、パターンは何度でも適用することが可能となり、リストの長さも任意となりますですが、残りの続く空の文字列は破棄されます。</li> </ul>

## Matcher メソッド

次に、Matcher のシステム静的メソッドを示します。

名前	引数	戻り値	説明
	String <i>s</i>	string	指定された文字列 <i>s</i> をリテラルに置き換える文字列を返します。返された文字列の文字は、 <i>s</i> の文字シーケンスに一致します。入力文字列の \ やなどのメタキャラクタやエスケープシーケンスは、特に意味のないリテラル文字として扱われます。

次に、Matcher のインスタンスマソッドを示します。

名前	引数	戻り値	説明
		Integer	最後に一致した文字の後の位置を返します。
	Integer <i>groupIndex</i>	Integer	前のマッチ処理で、グループ <i>groupIndex</i> が取得したサブシーケンスの最後の文字の後の位置を返します。一致は成功したが、グループ自体に一致がない場合は、メソッドの戻り値は -1 となります。
			取得されたグループは、左から右へ、1 から順にインデックス付けされます。グループ 0 はパターン全体を表します。つまり、 <i>groupIndex</i> は同じ意味となります。  <a href="#">「キャプチャグループについて」</a> を参照してください。
		Boolean	パターンに一致する入力シーケンスの次のサブシーケンスを検索します。入力シーケンスのサブシーケンスが Matcher オブジェクトのパターンに一致する場合、このメソッドは true を返します。  このメソッドは Matcher オブジェクトのリージョンの最初から開始します。または、前のメソッド呼び出しが成功し、Matcher オブジェクトがそれ以降リセットされていない場合、前のマッチ処理で一致しなかった最初の文字から開始します。

名前	引数	戻り値	説明
			<p>マッチ処理が成功した場合、<code>group</code>、<code>groupIndex</code>、<code>groupValue</code> メソッドを使用してさらに多くの情報を取得できます。</p> <p>詳細は、「<a href="#">リージョンの使用</a>」を参照してください。</p>
	<code>Integer group</code>	<code>Boolean</code>	<p><code>Matcher</code> オブジェクトをリセットし、パターンに一致する入力シーケンスの次のサブシーケンスを検索します。入力シーケンスのサブシーケンスが <code>Matcher</code> オブジェクトのパターンに一致する場合、このメソッドは <code>true</code> を返します。</p> <p>マッチ処理が成功した場合、<code>group</code>、<code>groupIndex</code>、<code>groupValue</code> メソッドを使用してさらに多くの情報を取得できます。</p>
		<code>String</code>	<p>前のマッチ処理で返された入力サブシーケンスを返します。</p> <p>など、グループによっては空の文字列にも一致します。入力された空の文字列とそのようなグループが一致した場合、このメソッドは空の文字列を返します。</p>
	<code>Integer groupIndex</code>	<code>String</code>	<p>前のマッチ処理の中で、指定したグループ <code>groupIndex</code> が取得した入力サブシーケンスを返します。一致は成功したが、指定されたグループが入力サブシーケンスのどの部分にも一致しない場合は、<code>null</code> 値を返します。</p> <p>取得されたグループは、左から右へ、1から順にインデックス付けされます。グループ0はパターン全体を表します。つまり、<code>group(0)</code> は <code>groupValue()</code> と同じ意味となります。</p> <p>など、グループによっては空の文字列にも一致します。入力された空の文字列とそのようなグループが一致した場合、このメソッドは空の文字列を返します。</p> <p>「<a href="#">キャプチャグループについて</a>」を参照してください。</p>

名前	引数	戻り値	説明
		Integer	一致するオブジェクトのパターン内のキャプチャグループ数を返します。グループ 0 はパターン全体を表し、この数には含まれません。  <a href="#">「キャプチャグループについて」</a> を参照してください。
		Boolean	Matcher オブジェクトにアンカー付き境界がある場合は true、それ以外は false を返します。デフォルトでは、Matcher オブジェクトはアンカー付き境界リージョンを使用します。  Matcher オブジェクトがアンカー付き境界を使用している場合、Matcher オブジェクトのリージョンの境界は ^ や \$ などのラインアンカー行の開始と終了に一致します。  詳細は、 <a href="#">「境界の使用」</a> を参照してください。
		Boolean	Matcher オブジェクトに透明な境界がある場合は true、不透明な境界を使用している場合は false を返します。デフォルトでは、Matcher オブジェクトは不透明なリージョン境界を使用します。  詳細は、 <a href="#">「境界の使用」</a> を参照してください。
		Boolean	この Matcher オブジェクトが最後に実行したマッチ処理で、検索エンジンにより入力の最後が見つかった場合に true を返します。このメソッドが true を返す場合、入力がさらに多ければ最後の検索の結果が異なっていた可能性があります。
		Boolean	パターンに対し、リージョンの先頭から入力シーケンスの一一致を確認します。  メソッドと同様に、このメソッドは必ずリージョンの先頭から開始します。異なる点は、リージョン全体が一致する必要がないことです。  マッチ処理が成功した場合、  メソッドを使用してさらに多くの情報を取得できます。  <a href="#">「リージョンの使用」</a> を参照してください。

名前	引数	戻り値	説明
		Boolean	パターンに対してリージョン全体が一致するかどうかを確認します。 マッチ処理が成功した場合、 <a href="#">next()</a> 、 <a href="#">start()</a> 、 <a href="#">end()</a> メソッドを使用してさらに多くの情報を取得できます。 <a href="#">「リージョンの使用」</a> を参照してください。
		Pattern オブジェクト	この Matcher オブジェクトが作成された Pattern オブジェクトを返します。
	Integer <i>start</i> Integer <i>end</i>	Matcher オブジェクト	この Matcher オブジェクトのリージョンの制限を設定します。リージョンは一致を検索する入力シーケンスの一部です。このメソッドは最初に Matcher オブジェクトをリセットし、 <a href="#">start()</a> で指定されたインデックスで開始し、 <a href="#">end()</a> で指定されたインデックスで終了するよう設定します。 使用されている透明な境界により、アンカーのような特定の構造が、リージョンの境界またはその周りで異なる動作をする可能性があります。 <a href="#">「リージョンの使用」</a> および <a href="#">「境界の使用」</a> を参照してください。
		Integer	この Matcher オブジェクトのリージョンの終了インデックス (含まない) を返します。 <a href="#">「リージョンの使用」</a> を参照してください。
		Integer	この Matcher オブジェクトのリージョンの開始インデックス (含む) を返します。 <a href="#">「リージョンの使用」</a> を参照してください。
	String <i>s</i>	String	入力シーケンスのすべてのサブシーケンスを、置き換え文字列 <i>s</i> で置き換えます。 このメソッドは最初に Matcher オブジェクトをリセットし、パターンの一一致を探しながら入力シーケンスをスキャンします。一致のどの部分にも含まれない文字は、結果の文字列に直接追加されます。各一致は、置き換え文字列により結果の文字列が置き換えられます。置き換え文

名前	引数	戻り値	説明
			<p>字列には、取得されたサブシーケンスへの参照が含まれている場合があります。</p> <p>置き換え文字列のバックスラッシュ (\) とドル記号 (\$) は、文字列がリテラル置き換え文字列として処理された場合は異なる結果となる場合があります。ドル記号は取得されたサブシーケンスへの参照、バックスラッシュは置き換え文字列の中でリテラル文字をエスケープするために使用されます。</p> <p>このメソッドを起動すると、Matcher オブジェクトの状態が変わります。Matcher オブジェクトをさらにマッチ処理で使用する場合、まずリセットする必要があります。</p> <p>正規表現 、 入力文字列 、 置き換え文字列 を指定する場合、その表現の Matcher オブジェクトでこのメソッドを呼び出すと、文字列 が生成されます。</p>
	String <i>s</i>	String	<p>置き換え文字列 <i>s</i> のパターンに一致する入力シーケンスの最初のサブシーケンスを置き換えます。</p> <p>置き換え文字列のバックスラッシュ (\) とドル記号 (\$) は、文字列がリテラル置き換え文字列として処理された場合は異なる結果となる場合があります。ドル記号は取得されたサブシーケンスへの参照、バックスラッシュは置き換え文字列の中でリテラル文字をエスケープするために使用されます。</p> <p>このメソッドを起動すると、Matcher オブジェクトの状態が変わります。Matcher オブジェクトをさらにマッチ処理で使用する場合、まずリセットする必要があります。</p> <p>正規表現 、 入力文字列 、 置き換え文字列 を指定する場合、その表現の Matcher オブジェクトでこのメソッドを呼び出すと、文字列 が生成されます。</p>

名前	引数	戻り値	説明
		Boolean	<p>入力が増えると、正の一致が負の一致になる可能性がある場合、true を返します。</p> <p>このメソッドが true を返し、かつ一致が見つかった場合、入力が増えると一致しなくなる場合があります。</p> <p>このメソッドが false を返し、かつ一致が見つかった場合、入力が増えると一致結果が変わることがありますが、一致しなくなることはありません。</p> <p>一致が見つからなかった場合、は意味を持ちません。</p>
		Matcher オブジェクト	<p>この Matcher オブジェクトをリセットします。Matcher オブジェクトをリセットすると、明示的な状態情報はすべて破棄されます。</p> <p>Matcher オブジェクトがアンカー付き境界を使用しているかどうかに関わらず、メソッドは変わりません。アンカー付き境界を変更するには、メソッドを明示的に使用する必要があります。</p> <p>詳細は、「<a href="#">境界の使用</a>」を参照してください。</p>
	String <i>s</i>	Matcher	この Matcher オブジェクトを新しい入力シーケンス <i>s</i> でリセットします。Matcher オブジェクトをリセットすると、明示的な状態情報はすべて破棄されます。
		Integer	前のマッチ処理の最初の文字の開始インデックスを返します。
	Integer <i>groupIndex</i>	Integer	<p>前のマッチ処理の中で、<i>groupIndex</i> が指定したグループが取得したサブシーケンスの開始インデックスを返します。取得されたグループは、左から右へ、1 から順にインデックス付けされます。グループ 0 はパターン全体を表します。つまり、は同じ意味となります。</p> <p><a href="#">「キャプチャグループについて」</a> (ページ 824) を参照してください。</p>
	Boolean <i>b</i>	Matcher オブジェクト	この Matcher オブジェクトのリージョンのアンカー付き境界を設定します。デフォルトでは、

名前	引数	戻り値	説明
			<p>Matcher オブジェクトはアンカー付き境界リージョンを使用します。</p> <p>このメソッドに <code>Pattern</code> を指定すると、Matcher オブジェクトはアンカー付き境界を使用します。 <code>Boolean</code> を指定すると、非アンカー付き境界を使用します。</p> <p>Matcher オブジェクトがアンカー付き境界を使用している場合、Matcher オブジェクトのリージョンの境界は ^ や \$ などのラインアンカー行の開始と終了に一致します。</p> <p>詳細は、「<a href="#">境界の使用</a>」(ページ 824)を参照してください。</p>
<code>Pattern pattern</code>	<code>Matcher</code> オブジェクト	<code>Matcher</code> オブジェクト	<p><code>Matcher</code> オブジェクトが一致を探すのに使用する <code>Pattern</code> オブジェクトを変更します。このメソッドにより、<code>Matcher</code> オブジェクトは最後に一致したグループについての情報を失います。入力での <code>Matcher</code> オブジェクトの位置は保持されます。</p>
<code>Boolean b</code>	<code>Matcher</code> オブジェクト	<code>Matcher</code> オブジェクト	<p>この <code>Matcher</code> オブジェクトの透明な境界を設定します。デフォルトでは、<code>Matcher</code> オブジェクトはアンカー付き境界リージョンを使用します。</p> <p>このメソッドに <code>Boolean</code> を指定すると、<code>Matcher</code> オブジェクトは透明な境界を使用します。</p> <p><code>Boolean</code> を指定すると、不透明な境界を使用します。</p> <p>詳細は、「<a href="#">境界の使用</a>」を参照してください。</p>

## Publisher Action クラス



メモ: アプリケーションでは、QuickAction はアクションと呼ばれます。

パブリッシャーアクション機能を使用すると、アクションを作成して、ホームページ、Chatter タブ、およびレコード詳細ページの Chatter パブリッシャーにそれらを追加できます。また、投稿、ファイル、リンク、およびアンケートなど、標準的な Chatter アクションが表示される順番をカスタマイズすることもできます。

作成アクションとカスタムアクションという 2 つのタイプのアクションがあります。

- 作成アクションではレコードを作成できます。これらは、オブジェクトタブの[新規]ボタンや、ホームページの[簡易作成]機能および[新規作成]機能とは異なります。作成アクションでは入力規則や項目の必須性が遵守され、各アクションの項目を選択できます。
- カスタムアクションは、定義された機能を備えた Visualforce ページです。たとえば、ユーザが 1000 文字よりも長いコメントを作成できるように、カスタムアクションを作成するとします。

両方のタイプのアクションで、オブジェクト固有アクションまたはグローバルアクションのいずれかを作成できます。

このセクションには、Apex で *QuickAction* と呼ばれるこれらのアクションを作成するための Apex クラスに関する情報が含まれています。

パブリッシャーアクションについての詳細は、オンラインヘルプを参照してください。

## 関連リンク

[QuickAction クラス](#)

[QuickAction.QuickActionRequest クラス](#)

[QuickAction.QuickActionResult クラス](#)

## QuickAction クラス

Apex を使用して、カスタム項目が許可されるオブジェクトや Chatter フィードに表示されるオブジェクト、またはグローバルに使用可能なオブジェクトについて、パブリッシャーアクションを要求したり処理したりできます。

### メソッド

次に、クラスの静的メソッドを示します。

メソッド	引数	戻り値	説明
	QuickAction. QuickActionRequest <i>performQuickAction</i>	QuickAction.QuickActionResult	クリックアクション要求で指定されたクリックアクションを実行し、アクションの結果を返します。
	QuickAction. QuickActionRequest <i>performQuickAction</i> Boolean <i>allOrNothing</i>	QuickAction.QuickActionResult	部分的な完了オプションを設定し、クリックアクション要求で指定されたクリックアクションを実行して、結果を返します。  <i>allOrNothing</i> 引数では、この操作で部分的な完了を許可するかどうかを指定します。この引数を <b>True</b> に設定した場合、1つのレコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理

メソッド	引数	戻り値	説明
			由の確認に使用できる結果オブジェクトを返します。
	LIST<QuickAction.QuickActionRequest> <i>performQuickActions</i>	LIST<QuickAction.QuickActionResult>	クイックアクション要求リストで指定された各クイックアクションを実行し、アクションの結果を返します。
	LIST<QuickAction.QuickActionRequest> <i>performQuickActions</i> Boolean <i>allOrNothing</i>	LIST<QuickAction.QuickActionResult>	部分的な完了オプションを設定し、クイックアクション要求リストで指定された各クイックアクションを実行して、アクションの結果を返します。  <i>allOrNothing</i> 引数では、この操作で部分的な完了を許可するかどうかを指定します。この引数を <b>True</b> に設定した場合、1つのレコードが失敗しても、残りの DML 操作を正常に完了できます。このメソッドは、どのレコードが成功または失敗したか、およびその理由の確認に使用できる結果オブジェクトを返します。

## サンプル

このサンプルでは、挿入する新しい取引先責任者をクイックアクションで作成するかどうかをトリガで判定します。作成する場合、**Account** カスタム項目に、クイックアクションが取引先責任者にとってグローバルかローカルかに応じた値が設定されます。または、挿入する取引先責任者をクイックアクションで作成しない場合は、**Account** 項目が **False** に設定されます。

このサンプルでは、渡された取引先責任者オブジェクトに対してグローバルアクション( )を実行します。

## 関連リンク

## Publisher Action クラス

#### `QuickAction.QuickActionRequest` クラス

クラスを使用して、アクション情報を提供し、  
クラスメソッドでクリックアクションを実行できるようにします。アクション情報には、アクション名、親レコード ID、レコードが含まれます。このクラスのコンストラクタは、引数を取りません。

## メソッド

次に、クラスのインスタンスマетодを示します。

メソッド	引数	戻り値	説明
		Id	この QuickAction の親レコード ID を返します。
		String	この QuickAction の名前を返します。
		SObject	QuickAction に関連付けられたレコードを返します。
setParentId	Id	Void	この QuickAction の親 ID を設定します。 によって返されます。
setName	String	Void	この QuickAction の名前を設定します。 によって返されます。

メソッド	引数	戻り値	説明
	SObject	Void	この QuickAction のレコードを設定します。によって返されます。

## 関連リンク

[Publisher Action クラス](#)

## QuickAction.QuickActionResult クラス

クラスを使用してパブリッシャーアクションを開始した後、クラスを使用してアクションの結果を処理します。

### メソッド

次に、

クラスのインスタンスマソッドを示します。

メソッド	戻り値	説明
	List<Database.Error>	エラーが発生した場合、1つ以上のデータベースエラーオブジェクトからなる配列、エラーコード、および説明を返します。詳細は、「 <a href="#">Database.Error クラス</a> 」(ページ 426)を参照してください。
	List<Id>	処理される QuickActions の ID。
	Boolean	アクションが作成される場合は <code>true</code> 、それ以外の場合は <code>false</code> を返します。
	Boolean	アクションが正常に完了した場合は <code>true</code> 、それ以外の場合は <code>false</code> を返します。

## 関連リンク

[Publisher Action クラス](#)

## Site クラス

次は、Force.com Sites の一部である `Site` クラスの静的メソッドです。

名前	引数	戻り値	説明
	String <i>newpassword</i> String <i>verifynewpassword</i> String <i>opt_oldpassword</i>	System.PageReference	現在のユーザのパスワードを変更します。
	sObject <i>user</i> String <i>ownerId</i> String <i>password</i>	ID	ゲストユーザのプロファイルに定義されているデフォルトのレコードタイプを使用して個人取引先を作成し、サイトのポータルでその個人取引先を有効化します。
			 メモ: この方法は、サイトがカスタマーポータルに関連付けられている場合、およびデフォルトの新しいユーザプロファイルのユーザライセンスが大規模ポータルユーザである場合にのみ有効です。
	sObject <i>user</i> String <i>ownerId</i> String <i>recordTypeID</i> String <i>password</i>	ID	指定された <i>recordTypeID</i> を使用して個人取引先を作成し、サイトのポータルでその個人取引先を有効化します。
			 メモ: この方法は、サイトがカスタマーポータルに関連付けられている場合、およびデフォルトの新しいユーザプロファイルのユーザライセンスが大規模ポータルユーザである場合にのみ有効です。
	sObject <i>user</i> String <i>accountId</i> String <i>opt_password</i> boolean <i>opt_sendEmailConfirmation</i>	ID	指定された取引先のポータルユーザを作成し、サイトのポータルと関連付けます。 (省略可能) <i>opt_password</i> 引数は、ポータルユーザのパスワー

名前	引数	戻り値	説明
			<p>ドです。指定しない場合、または　または空の文字列が設定されている場合、このメソッドは新しいパスワードメールをポータルユーザに送信します。</p> <p>(省略可能)</p> <p><i>opt_sendEmailConfirmation</i> 引数によって、新しいユーザメールがポータルユーザに送信されるかどうかが決定します。新しいユーザメールをポータルユーザに送信するには、に設定します。デフォルトはで、新しいユーザメールは送信されません。</p> <p>項目は、 メソッドを使用する場合にユーザの sObject に必要です。</p> <p> メモ: このメソッドは、サイトがカスタマーportalに関連付けられている場合のみ有効です。</p>
	String <i>username</i>	boolean	ユーザのパスワードをリセットし、新しいパスワードを記載したメールをユーザに送信します。パスワードのリセットが正常に行われたかどうかを示す値を返します。
	string		サイト管理者のメールアドレスを返します。
	ID		サイト管理者のユーザ ID を返します。
	string		サイトに関連付けられている追跡コード。このコードは、サイトに対するページリクエストデータを追跡するために、

名前	引数	戻り値	説明
			Google Analytics などのサービスで使用されます。
	string		現在の要求に対するサイト URL の値 (例: やなど) を返します。
	string		現在のサイトの カスタム アドレス 項目の値を返します。
	string		組織の Force.com ドメイン名を返します。
	string		現在のページがサイトに指定されたエラーページであり、エラーがある場合は、現在のページのエラーの説明を返し、そうでない場合は空の文字列を返します。
	string		現在のページがサイトに指定されたエラーページで、エラーがある場合は、現在のページのエラーメッセージを返し、そうでない場合は空の文字列を返します。
	string		現在のサイトの API 名を返します。
	string		このページがサイトに指定されたエラーページである場合は、元の URL を返し、そうでない場合は null を返します。
	string		現在のサイトの URL パスのプレフィックスを返します。たとえば、サイト URL が である場合、 がパスのプレフィックスです。プレフィックスが定義されていない場合、または、ページにカスタ

名前	引数	戻り値	説明
			ム Web アドレスを使用してア クセスした場合には、Null を返 します。
		System.PageReference	現在のサイトに関連付けられた テンプレートを返します。テン プレートが指定されていない場 合、デフォルトテンプレートを 返します。
		boolean	現在のサイトがログインが有効 なポータルと関連付けられてい る場合は <code>true</code> を返し、そうで ない場合は <code>false</code> を返します。
		boolean	認証ユーザの場合、現在ログイ ンしているユーザのパスワード の有効期限が切れている場合、 <code>false</code> を返します。認証され ていないユーザの場合は、 <code>true</code> を返します。
		boolean	現在のサイトがセルフ登録対応 のカスタマーポータルと関連付 けられている場合は <code>true</code> を返 し、そうでない場合は <code>false</code> を返します。
	String <i>username</i> String <i>password</i> String <i>startUrl</i>	System.PageReference	ユーザは指定されたユーザ名お よびパスワードで現在のサイト にログインでき、ユーザを に誘導します。 が相対パスでない場 合、デフォルトはサイトの指定 されたインデックスページにな ります。
			 メモ: <code>startUrl</code> に または を指定しな いでください。

名前	引数	戻り値	説明
	sObject <i>user</i> String <i>contactId</i>	Void	<p>サイトのポータル内の指定されたユーザ情報を認証プロバイダ経由で設定します。</p> <ul style="list-style-type: none"><li>このメソッドは、サイトがカスタマーportalに関連付けられている場合にのみ有効です。</li><li>認証プロバイダについての詳細は、 を参照してください。</li></ul>

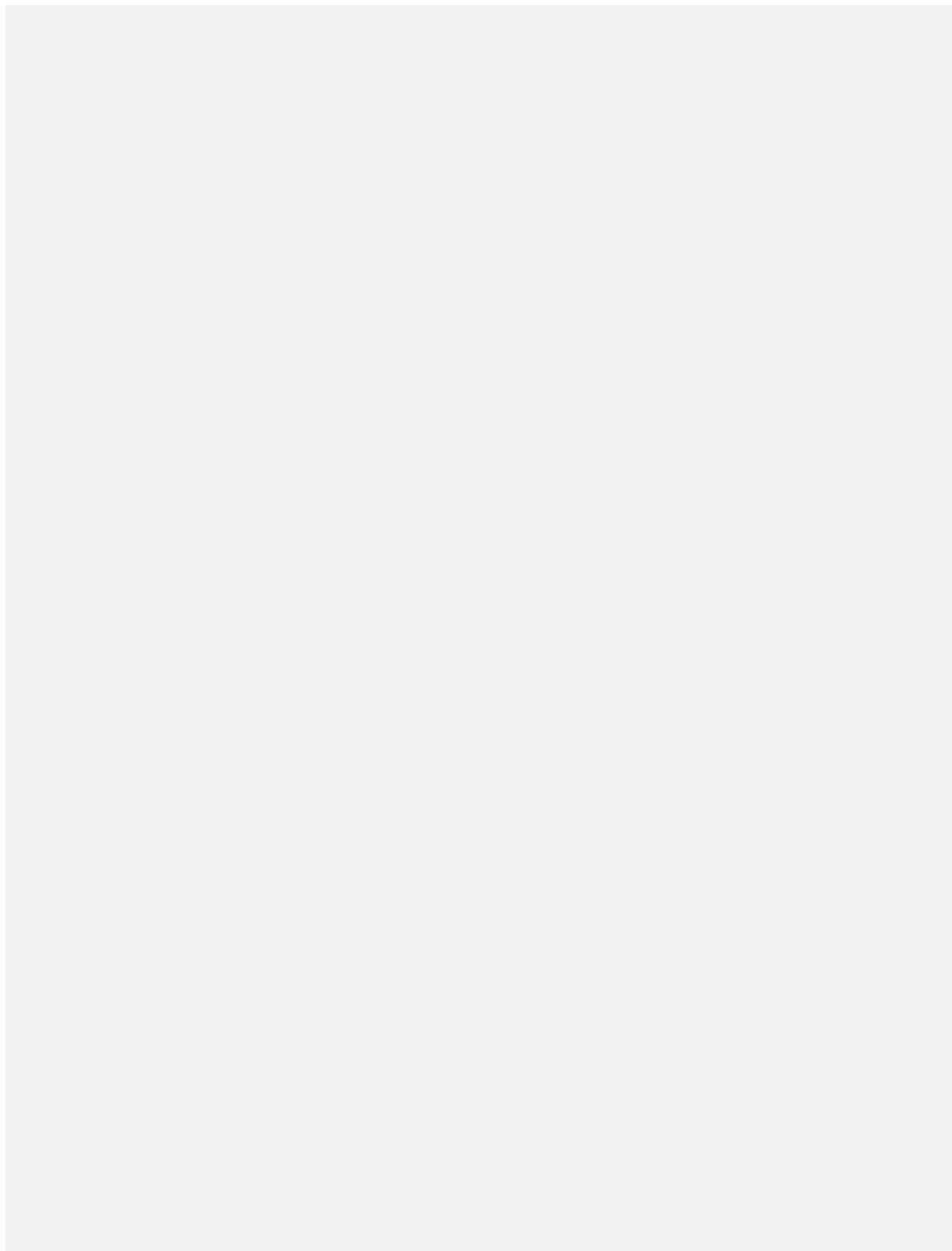
サイトの詳細は、Salesforce オンラインヘルプの「Force.com Sites の概要」を参照してください。

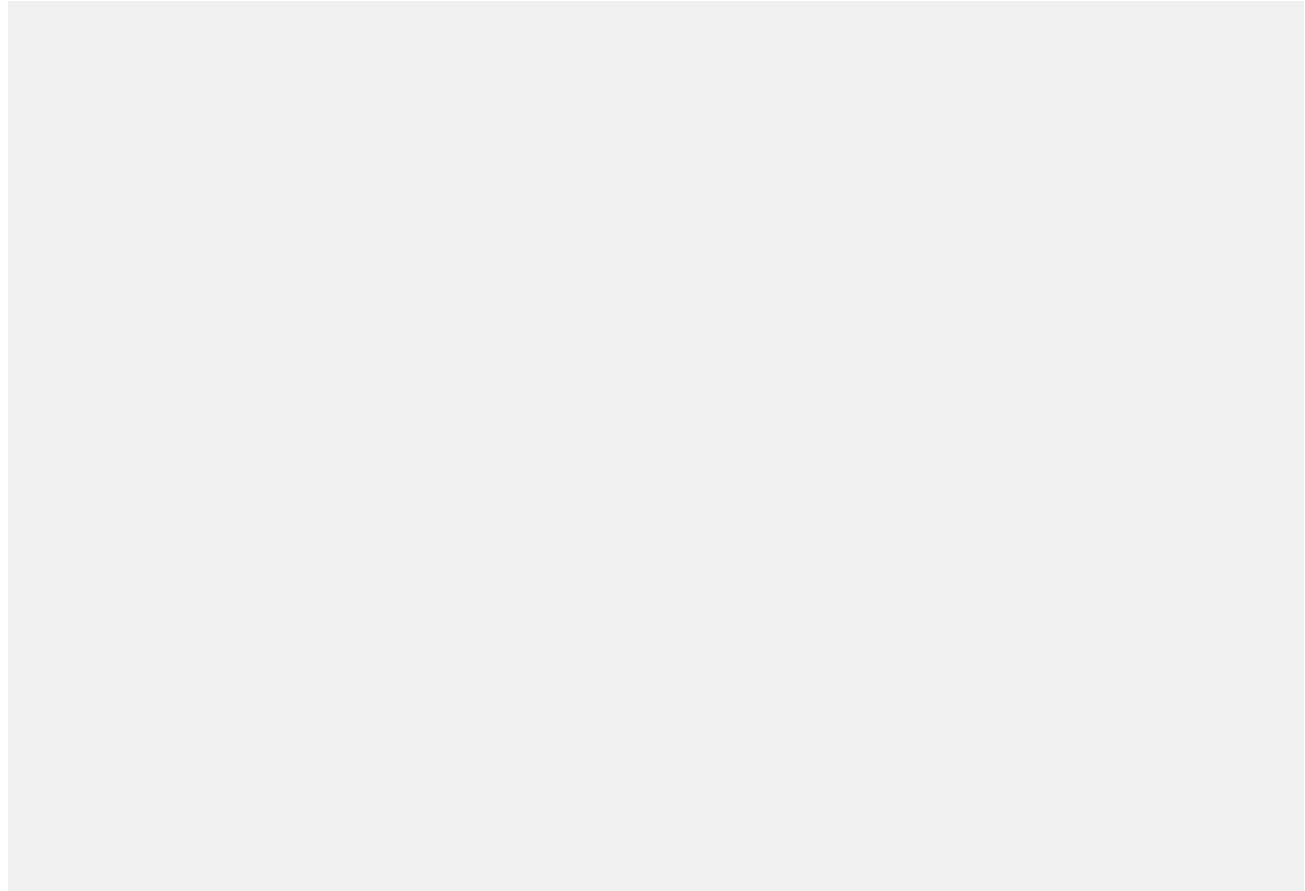
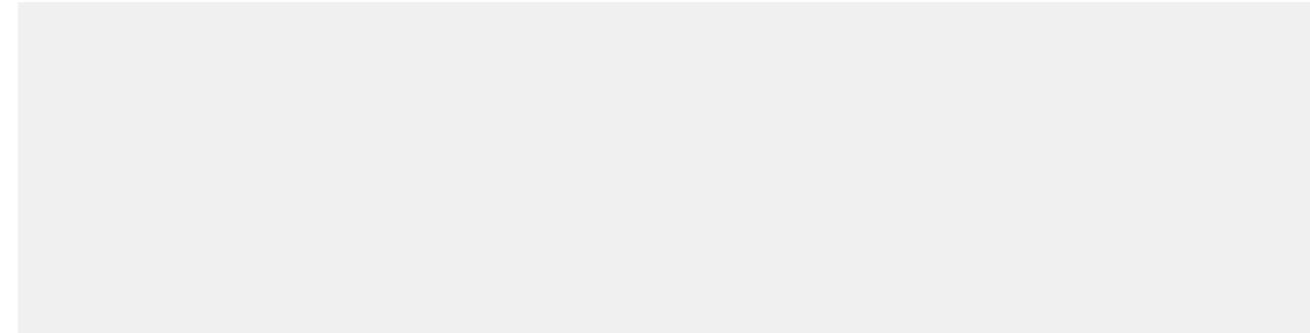
### Force.com Sites の例

次の例では、クラスを作成します。このクラスは Visualforce ページ(下記マークアップを参照)を使用して、新規カスタマーportalユーザを登録します。



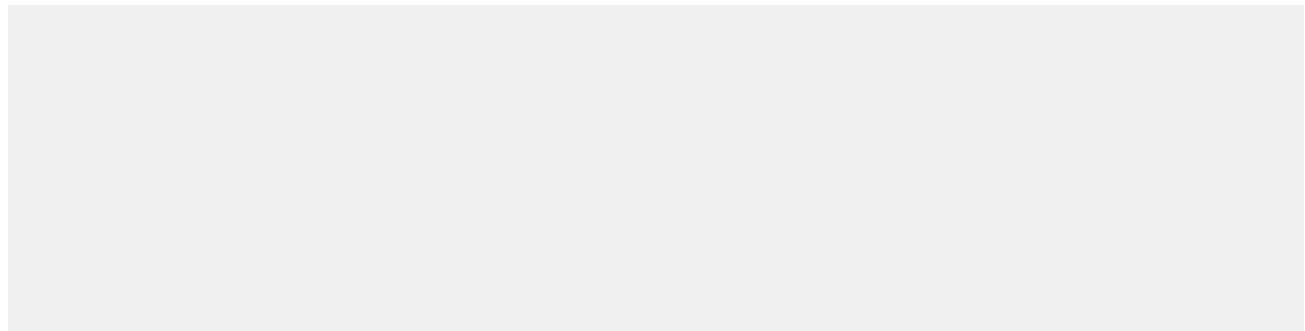
メモ: 次の例では、新しいportalユーザと関連付ける取引先の取引先 ID を入力する必要があります。取引先所有者をこのコード例が機能するロール階層に追加する必要があります。詳細は、Salesforce オンラインヘルプの「カスタマーportalの設定」を参照してください。

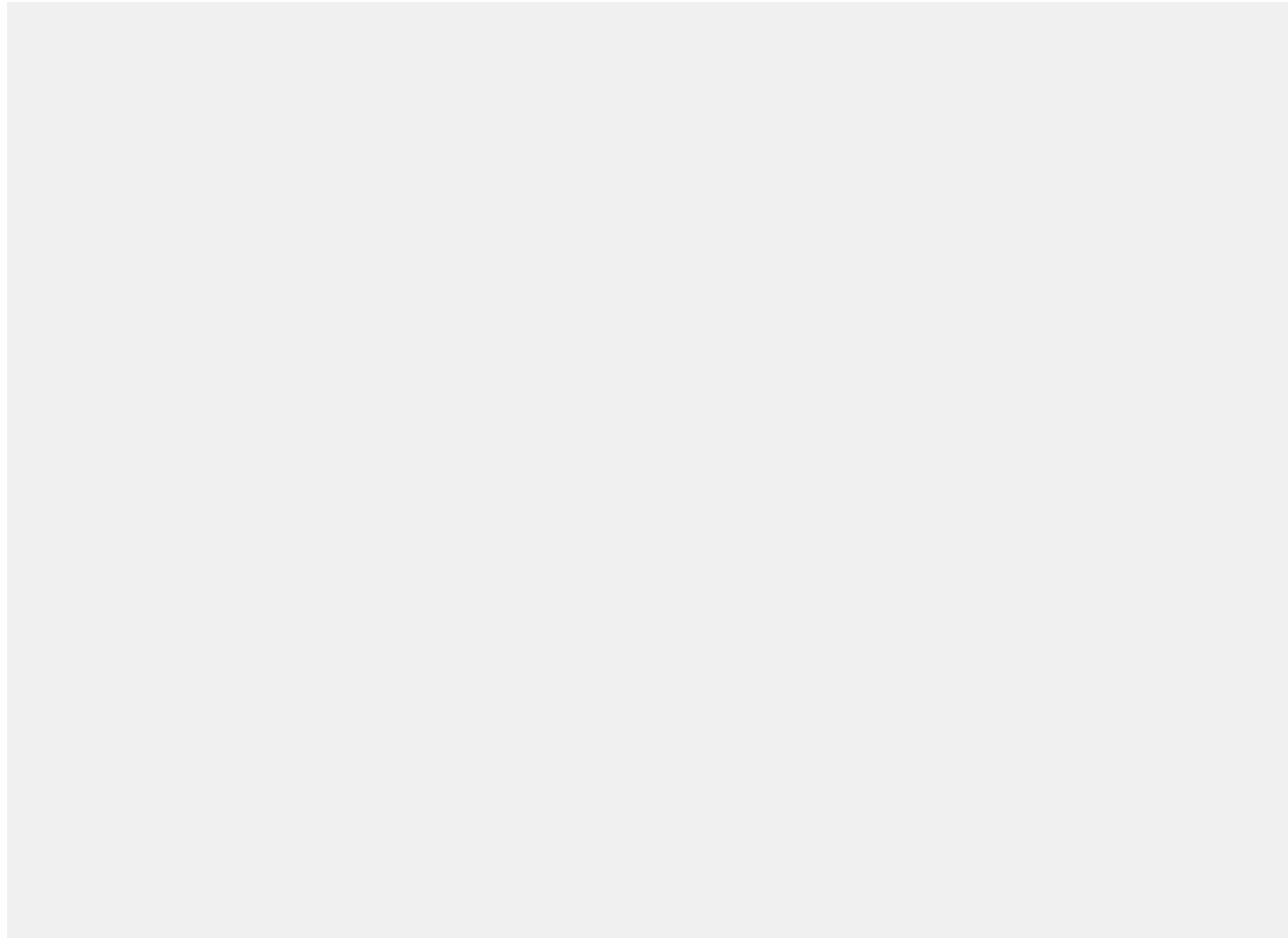




次は、上記の

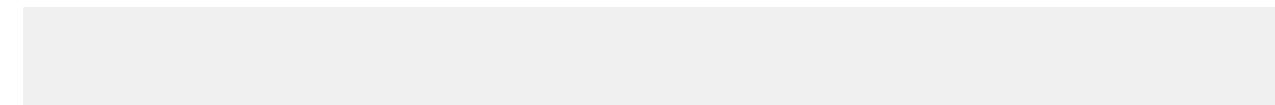
Apex コントローラを使用する Visualforce 登録ページです。





メソッドのサンプルコードは、上記のサンプルコードとほぼ同じですが、次の点が変更されています。

- のすべてのインスタンスを に置き換えています。
- accountIDではなくownerIDを決定し、次のコードブロックに置き換えることにより、  
ソッドではなく メソッドを使用しています。 メ



置換後



## Cookie クラス

クラスにより、Apex を使用して Force.com サイトの Cookie にアクセスできます。

クラスの メソッドを使用して、ページに Cookie を添付します。

**重要:**

- Apex の Cookie 名と値セットは URL 符号化されています。つまり、@などの文字は % 記号および16進数表現に置き換えられます。
- メソッドは Cookie 名にプレフィックス「」を追加します。
- Cookie の値を に設定すると、期限切れの属性の設定ではなく、空の文字列値の Cookie を送信します。
- Cookie の作成後は、Cookie のプロパティを変更することはできません。
- 機密情報を Cookie に格納する場合は注意してください。Cookie の値に関係なくページはキャッシュされます。動的なコンテンツを生成するために Cookie の値を使用する場合は、ページキャッシュを無効にする必要があります。詳細は、Salesforce オンラインヘルプの「Force.com サイトページのキャッシュ」を参照してください。

クラスを使用する場合は、次の制限に留意してください。

- クラスには、Salesforce.com API バージョン 19 以降を使用して保存されている Apex を使用することでのみアクセスできます。
- Force.com ドメインごとに設定できる Cookie の最大数はブラウザにより異なります。新しいブラウザは古いブラウザより高い制限が設定されています。
- Cookie は名前および属性を含め 4K 未満である必要があります。

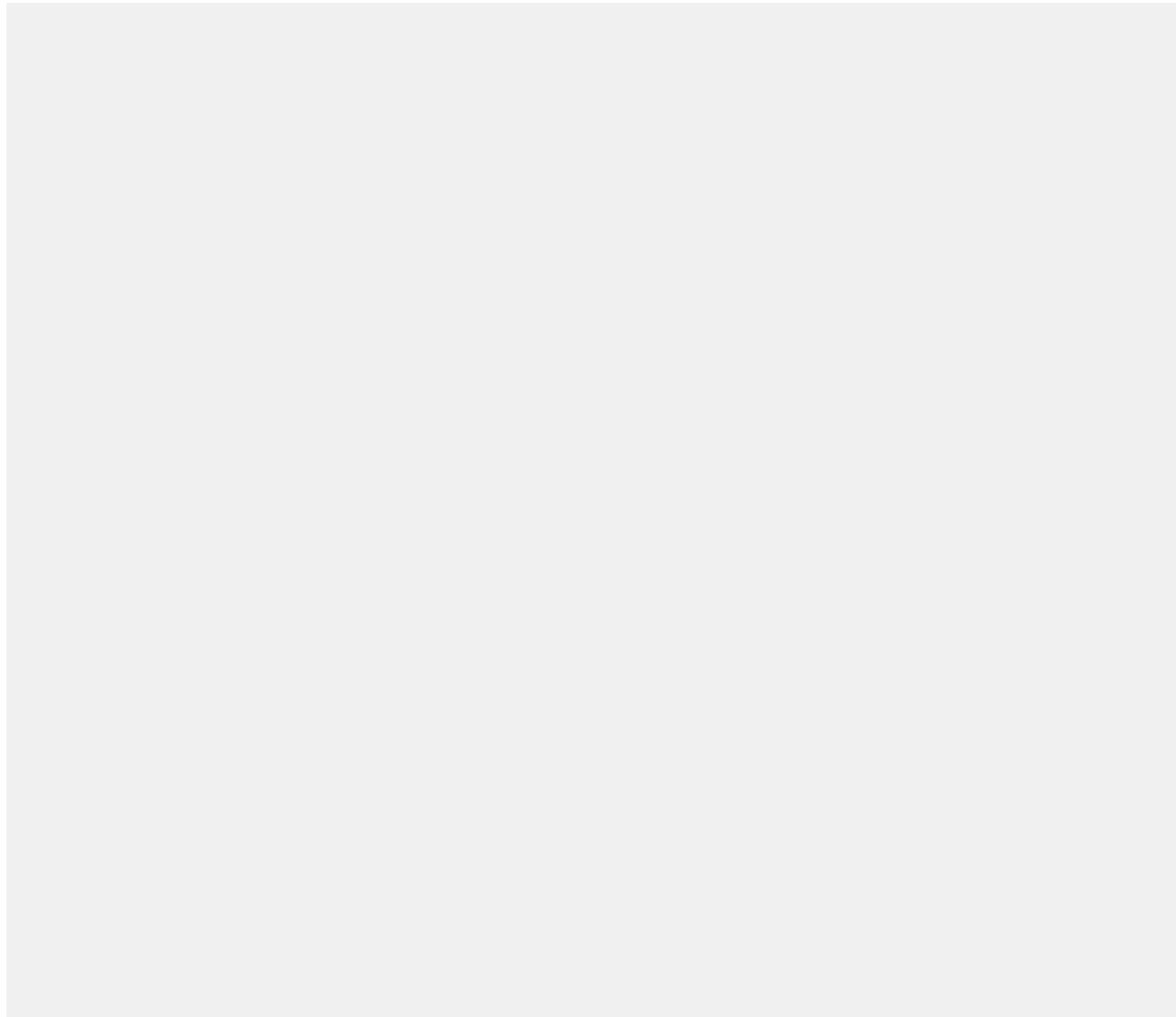
次に、Force.com Sites の一部である クラスのインスタンスマетодを示します。

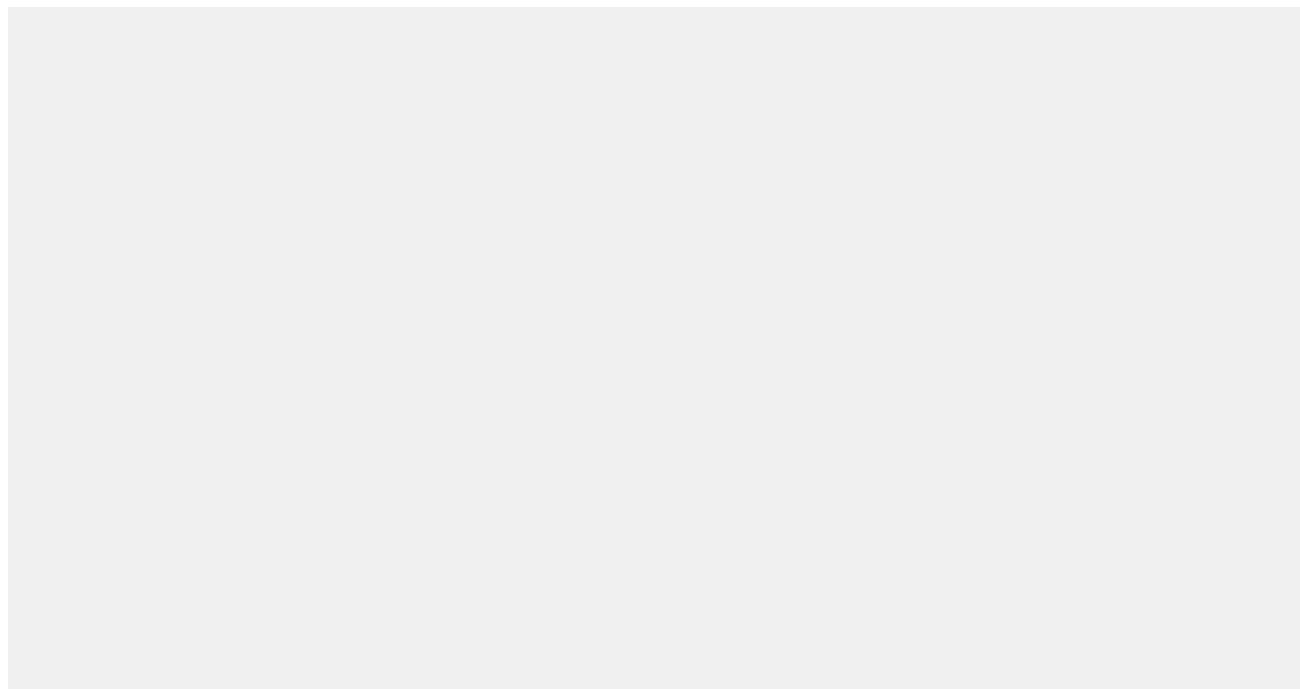
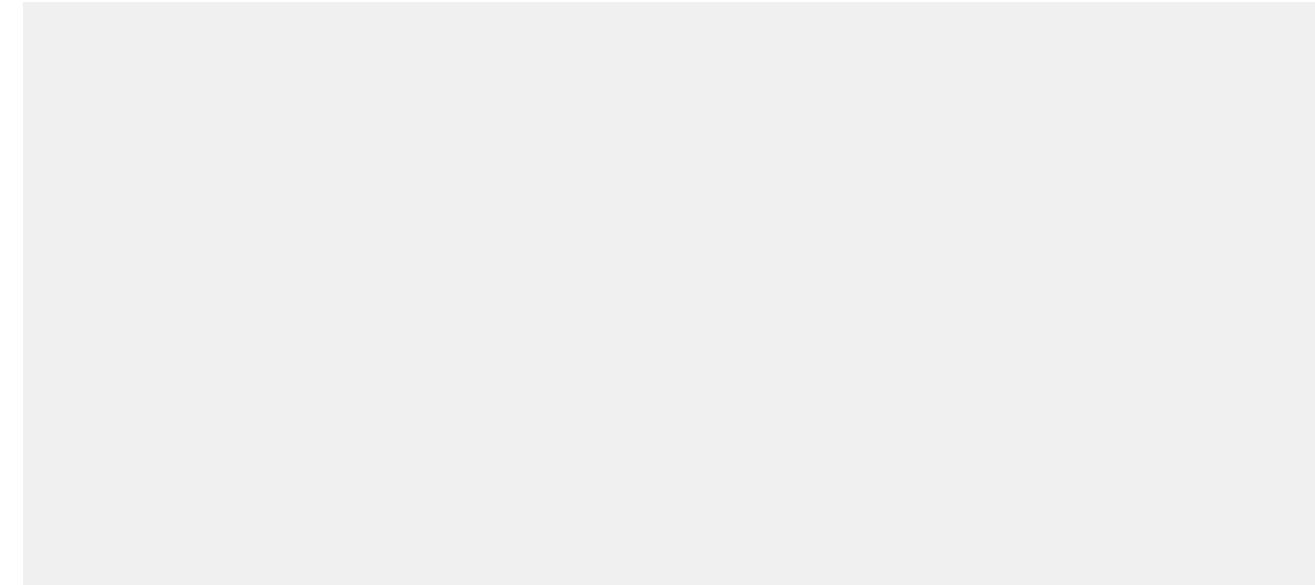
名前	引数	戻り値	説明
		string	要求を行うサーバの名前を返します。
		Integer	Cookie の有効期間を示す秒単位の数字が返されます。 を設定すると、セッション Cookie が発行されます。 を設定すると、Cookie は削除されます。
		string	Cookie の名前を返します。にはできません。
		string	Cookie の取得元のパスを返します。 または空白にすると、場所はルートまたは「/」に設定されます。

名前	引数	戻り値	説明
		string	セッション ID など、Cookie で取得されるデータを返します。
		boolean	Cookie が HTTPS でのみアクセス可能な場合、 <code>true</code> を返します。 それ以外の場合は、 <code>false</code> を返します。

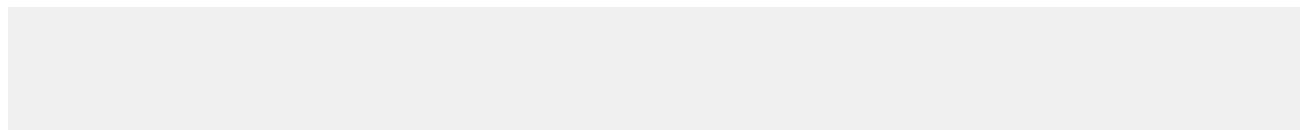
サイトの詳細は、Salesforce オンラインヘルプの「Force.com サイトの概要」を参照してください。

次の例では、`CountAccess` クラスを作成します。このクラスは Visualforce ページ（下記マークアップを参照）を使用して、ユーザにページが表示されるたびにカウンタが更新されます。ページへのアクセス回数が Cookie に保存されます。





次は、上記の Apex コントローラを使用する Visualforce ページです。アクションでは、上記のコントローラでメソッドをコールします。



## Visualforce クラス

Apex を使用すれば、開発者は、ボタンのクリック、関連レコードの更新などの Salesforce システムでのイベントに対しビジネスロジックを追加できるほか、次のカスタム Visualforce コントローラとコントローラ拡張を使用して Visualforce ページにカスタムのロジックを適用することもできます。

- カスタムコントローラは Apex で記述されるクラスで、標準コントローラを使用せずにすべてのページのロジックを実装します。カスタムコントローラを使用する場合、新しいナビゲーション要素または動作を定義できますが、標準コントローラにすでに定義された機能も再実装する必要があります。

その他の Apex クラスと同様に、カスタムコントローラ全体はシステムモードで実行されます。このモードでは現在のユーザのオブジェクトと項目レベルの権限は無視されます。カスタムコントローラ内で、ユーザプロファイルを用いてアクセスするか否かを独自に決定することができます。

- コントローラ拡張は、Apex で記述されるクラスで、標準コントローラまたはカスタムコントローラの動作を追加するか、動作を上書きします。拡張を使用すれば、独自のカスタムロジックを追加する一方で、別のコントローラの機能も使用できます。

標準コントローラはユーザモードで実行し、現在のユーザの権限、項目レベルのセキュリティ、共有ルールが強制されるため、標準コントローラを拡張すると、ユーザ権限を重視する Visualforce ページを構築できます。拡張クラスはシステムモードで実行しますが、標準コントローラはユーザモードで実行します。カスタムコントローラと同様、ユーザプロファイルを参照してプログラムでアクセスさせるか否かを指定できます。

ここでは、カスタム Visualforce コントローラおよびコントローラ拡張を構築する場合に使用できる、システムが提供する Apex クラスの情報について説明します。これらのクラスに加え、コントローラおよびコントローラ拡張でメソッドを宣言する場合に [キーワード](#) を使用できます。詳細は、「[キーワードの使用](#)」(ページ 194)を参照してください。

Visualforce についての詳細は、『[Visualforce 開発者ガイド](#)』を参照してください。

## Action クラス

を使用して、Visualforce カスタムコントローラまたはコントローラ拡張で使用できる action メソッドを作成できます。たとえば、カスタム保存を実行するコントローラ拡張に [メソッド](#) を作成できます。

### インスタンス化

次のコードのスニペットは、save アクションを使用する新しいオブジェクトをインスタンス化する方法について説明しています。

#### メソッド

action メソッドはすべて、[の特定のインスタンス](#) でコールされ、実行されます。

次の表は、[のインスタンスマソッド](#) を示します。



## 動的コンポーネントメソッドとプロパティ

Apex で表されるすべての動的 Visualforce コンポーネントは、次のプロパティにアクセスします。

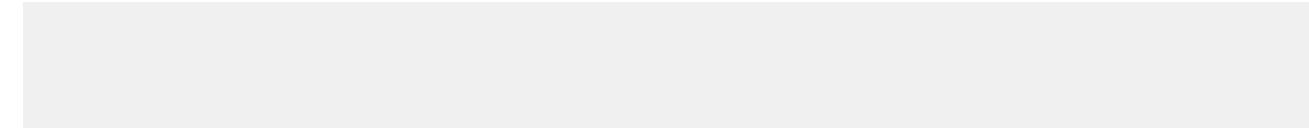


名前	引数	戻り値	説明
	List<String>	Void	<p>Visualforce ページが読み込まれると、Visualforce マークアップで参照される項目に基づいて、ページにアクセスできる項目が表示されます。このメソッドは、コントローラがそれらの項目にも明示的にアクセスできるように、に指定された各項目に参照を追加します。</p> <p>このメソッドは、レコードが読み込まれる前にコールする必要があります。通常、コントローラのコンストラクタによってコールされます。このメソッドがコンストラクタ外でコールされる場合、をコールする前にメソッドを使用する必要があります。</p> <p>の文字列には、AccountIdなどのAPI項目名か、foo__r.myField__cなどの項目への明示的なリレーションを使用できます。</p> <p>このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。</p>
		System.PageReference	キャンセルページの PageReference を返します。
		System.PageReference	レコードを削除し、削除ページの PageReference を返します。
		System.PageReference	標準編集ページの PageReference を返します。
	string		Visualforce ページ URL の クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードの ID を返します。
	SObject		Visualforce ページ URL の クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードを返します。
			関連付けられた Visualforce マークアップで参照される項目のみを、この SObject でクエリすることができます。関連するオブジェクトの項目など、他のすべての項目については、SOQL 表現を使用してクエリする必要があります。
			 <b>ヒント:</b> クエリする任意の追加項目を参照する非表示コンポーネントを使用すれば、この制約を回避できます。コンポーネントの属性をに設定して、コンポーネントを非表示にします。次に例を示します。

名前	引数	戻り値	説明
	Void		<p>新たに参照された項目へのアクセス権限を再取得するようにコントローラを強制します。このメソッドがコールされる前にレコードに加えられた変更是、すべて破棄されます。</p> <p>これは、<b>StandardController</b> がコンストラクタ外でコールされる場合にのみ使用するメソッドで、<b>StandardController</b> がコールされる直前にコールする必要があります。</p> <p>このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。</p>
	System.PageReference	System.PageReference	変更を保存し、更新された PageReference を返します。
	System.PageReference	System.PageReference	標準詳細ページの PageReference オブジェクトを返します。

## 例

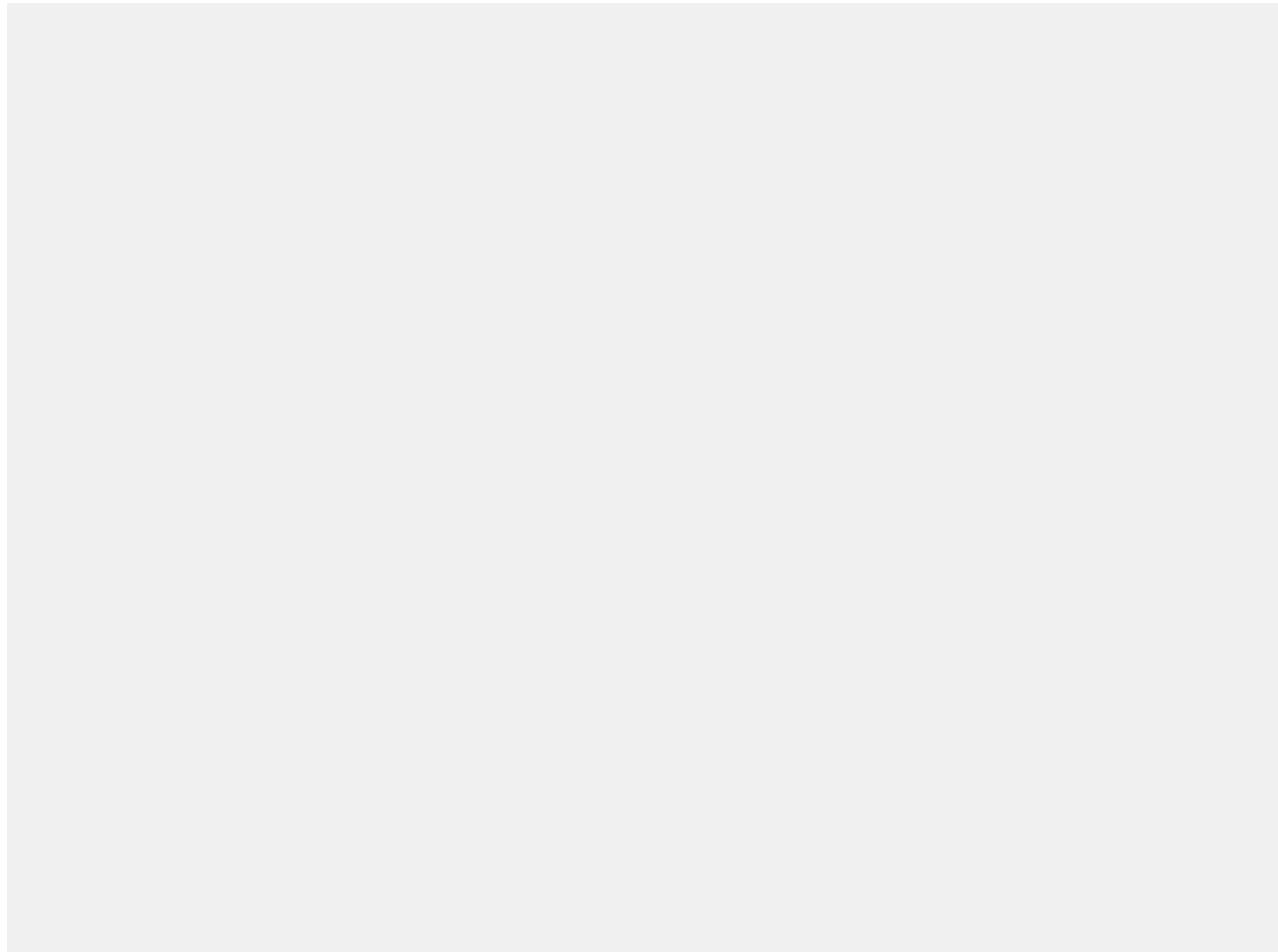
次の例では、`IdeaStandardController` オブジェクトをカスタムリストコントローラのコンストラクタで使用する方法を示します。この例では、コメントリストデータを Visualforce ページに表示する前に操作するためのフレームワークを示します。



次の Visualforce マークアップは、上記の IdeaStandardController の例をページ内で使用する方法を示します。この例が機能するためには、ページ名を `ideas` にする必要があります。



メモ: Visualforce ページにアイデアとコメントを表示するには、次の例でコメントを表示する特定のアイデアの ID (例: `Idea.getComments('00A100000000000')`) を指定する必要があります。



## 関連リンク

[Ideas クラス](#)

### IdeaStandardSetController クラス

IdeaStandardSetController オブジェクトは、固有の機能を提供します。

クラスで提供される機能のほか、アイデア



メモ: クラスおよび クラスは、現在限定リ  
リースプログラムでのみ使用できます。組織でのこれらのクラスの有効化についての詳細は、株式会社  
セールスフォース・ドットコムの担当者までお問い合わせください。

## インスタンス化

`IdeaStandardSetController` オブジェクトはインスタンス化できません。アイデアの標準リストコントローラを使用する場合は、カスタム拡張コントローラのコンストラクタを介してインスタンスを取得できます。

## メソッド

IdeaStandardSetController オブジェクトのメソッドは、IdeaStandardSetController の特定のインスタンスでコールされ、実行されます。

次の表は、`IdeaStandardSetController` のインスタンスマソッドを示します。

上記のメソッドのほか、  
他のメソッドを継承します。



メモ: **クラス**から継承したメソッドを使用して、  
返されたアイデアのリストを変更することはできません。

クラスは

**クラス**に関連付けら

メソッドによつ

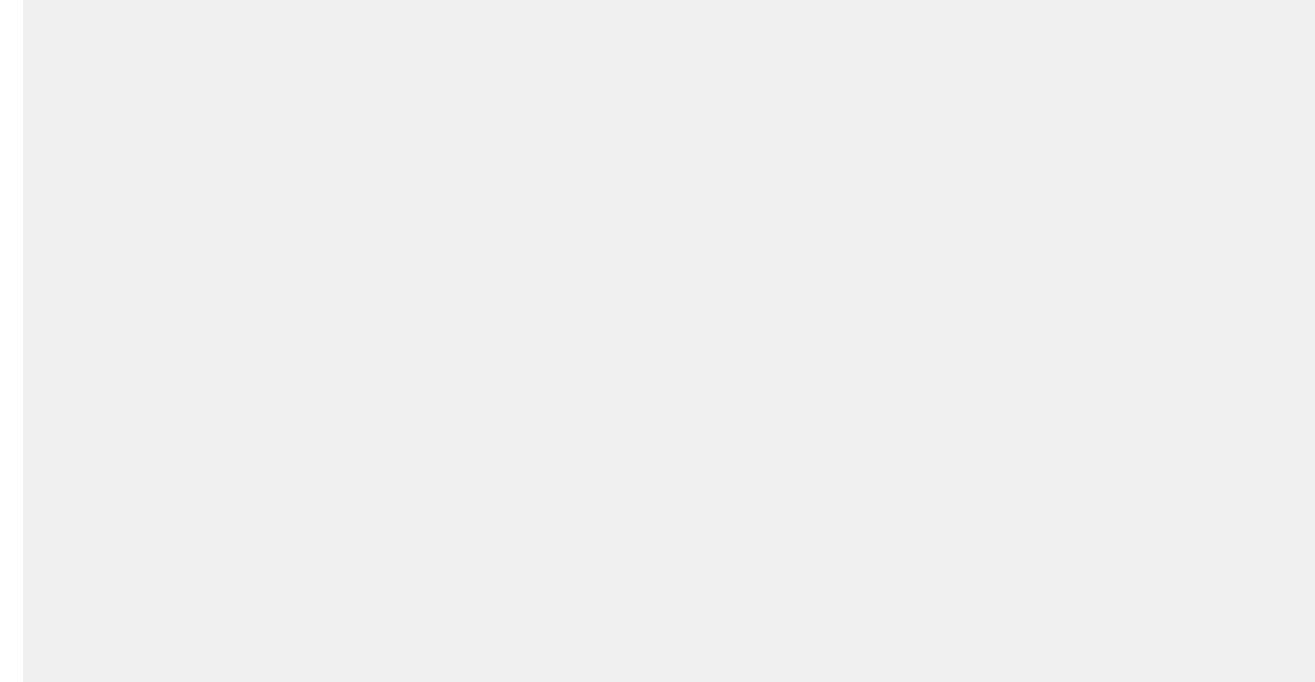
次の表に、継承されるメソッドの一覧を示します。

名前	引数	戻り値	説明
	System.PageReference	元のページ(わかっている場合)、またはホームページの PageReference を返します。	
	Void	レコードの最初のページを返します。	
	boolean	セット内に存在するレコード数がレコード数の上限を超えてるかどうかを示します。false の場合、レコード数がリストコントローラを使用して処理できる数を超えています。レコード数の上限は 10,000 レコードです。	
	string	現在のコンテキストでの検索条件の ID を返します。	
	boolean	現在のページセットの後に、より多くのレコードがあるかどうかを示します。	
	boolean	現在のページセットの前に、より多くのレコードがあるかどうかを示します。	
	System.SelectOption[]	現在のユーザが使用できるリストビューのリストを返します。	
	Integer	現在のページセットのページ番号を返します。最初のページは 1 を返します。	
	Integer	各ページセットに存在するレコード数を返します。	
	sObject	選択したレコードへの変更を示す sObject を返します。クラス内に含まれるプロトタイプオブジェクトを取得し、一括更新の実行に使用されます。	
	sObject[]	現在のページセットにある sObject のリストを返します。このリストは不变であるため、 <code>()</code> をコールできません。	
	Integer	セットに存在するレコード数を返します。	
	sObject[]	選択されている sObject のリストを返します。	
	Void	レコードの最後のページを返します。	
	Void	レコードの次のページを返します。	
	Void	レコードの前のページを返します。	

名前	引数	戻り値	説明
		System.PageReference	新しいレコードを挿入するか、変更された既存のレコードを更新します。この操作が完了した後、元のページ(わかっている場合)、またはホームページの PageReference を返します。
	String <i>filterId</i>	Void	コントローラの検索条件 ID を設定します。
	Integer <i>pageNumber</i>	Void	ページ番号を設定します。
	Integer <i>pageSize</i>	Void	各ページセット内のレコード数を設定します。
	sObjects [] <i>selectedRecords</i>	Void	選択したレコードを設定します。

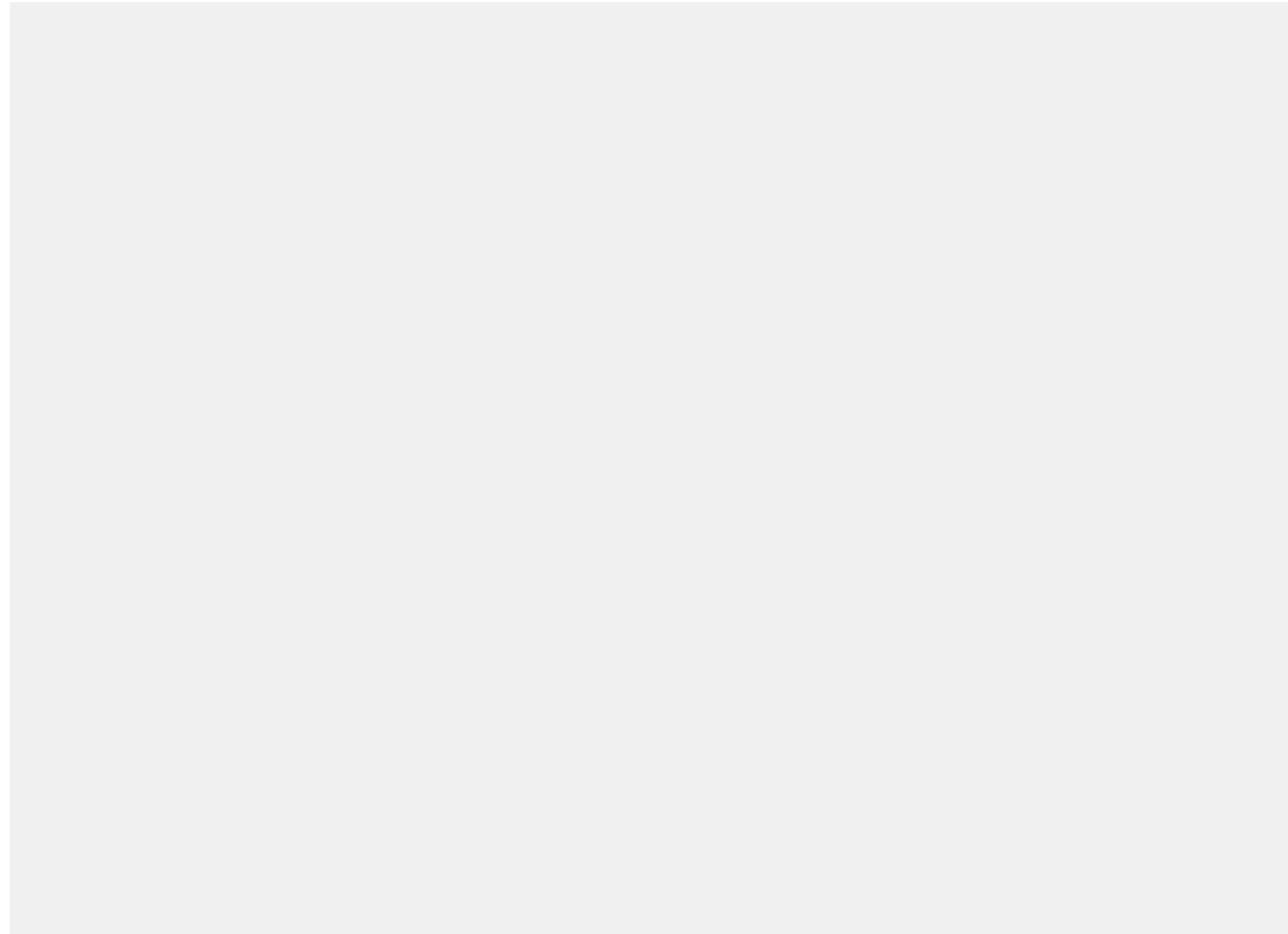
#### 例: プロファイルページの表示

次の例では、IdeaStandardSetController オブジェクトのカスタムリストコントローラのコンストラクタでの使用方法を示します。

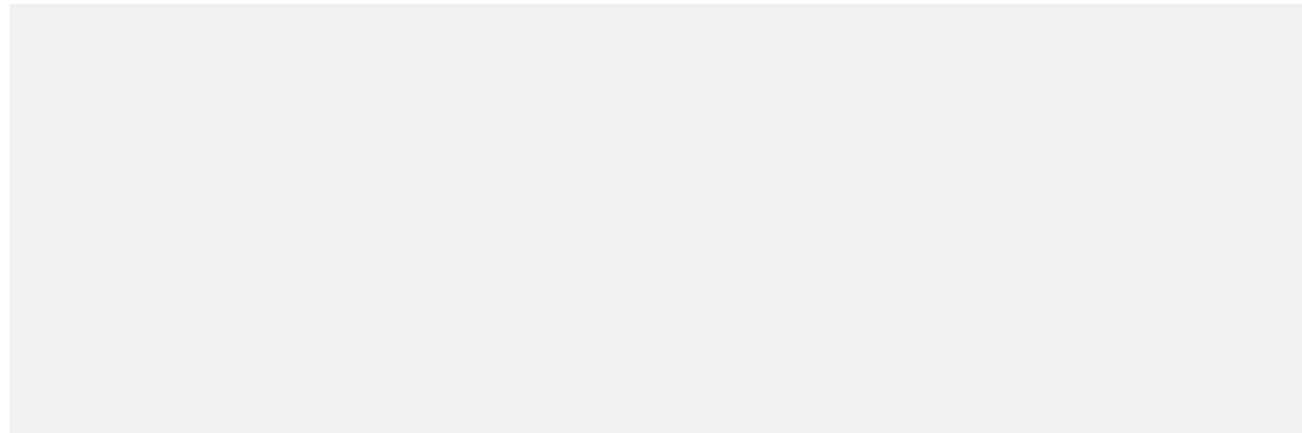


次の Visualforce マークアップは、上記の IdeaStandardSetController の例と  
コンポーネントによって、最新の回答、登録されたアイデア、ユーザに関連する投票の一覧を表示するプロファイルページがどのように表示されるかを示します。この例では特定のユーザ ID を識別しないため、ページには現在ログインしているユーザのプロファイルページが自動的に表示されます。この例が機能するためには、ページ名を [ ] にする必要があります。





前の例では、コンポーネントは、特定のアイデアの詳細ページを表示する次の Visualforce マークアップにリンクします。この例が機能するためには、ページ名を にすること必要があります。

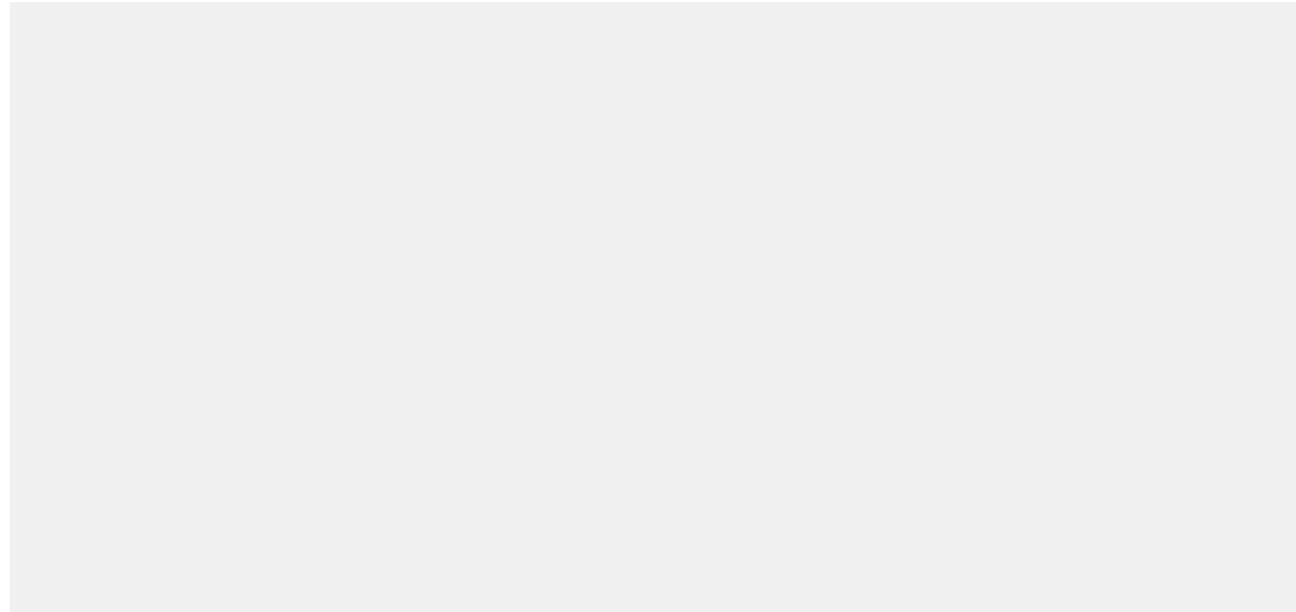


例: 上位のアイデアとコメント、最近のアイデアとコメント、最も人気のあるアイデアとコメントのリストを表示

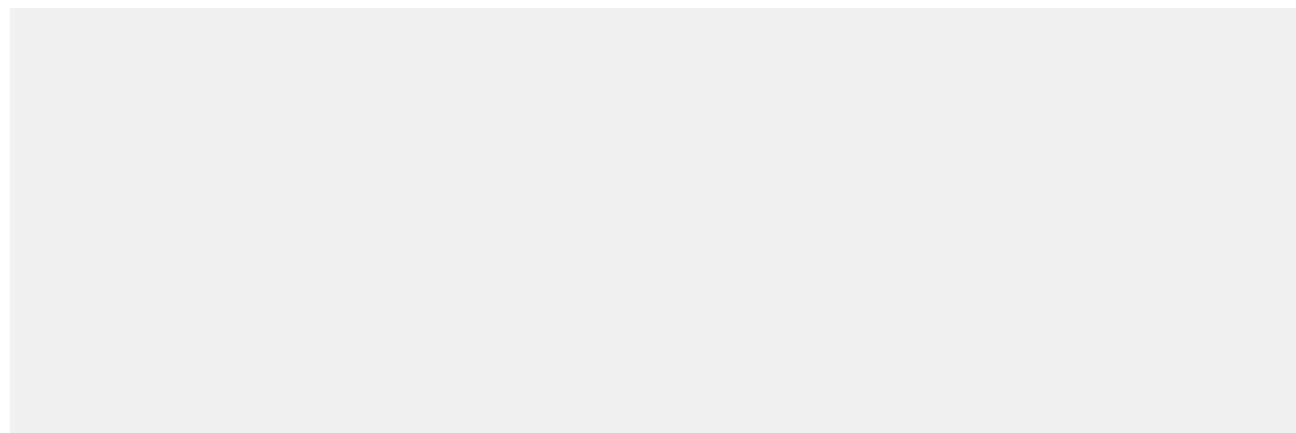
次の例では、`IdeaStandardSetController` オブジェクトのカスタムリストコントローラのコンストラクタでの使用方法を示します。

 メモ: この例でアイデアが返されるためには、少なくとも 1 つのアイデアを作成する必要があります。

次の Visualforce マークアップは、上記の `IdeaStandardSetController` 例を  
トと共に使用して、最近、上位、最も人気あるアイデアとコメントをどのように表示するかを示します。この例  
が機能するためには、ページ名を コンポーネン  
にする必要があります。



前の例では、コンポーネントは、特定のアイデアの詳細ページを表示する次の Visualforce マークアップにリンクします。このページの名前は にする必要があります。



## 関連リンク

[Ideas クラス](#)

## KnowledgeArticleVersionStandardController クラス

KnowledgeArticleVersionStandardController オブジェクトは、[クラス](#)で提供される機能のか、記事固有の機能を提供します。

### メソッド

KnowledgeArticleVersionStandardController オブジェクトには、次の特殊なインスタンスマソッドがあります。

名前	引数	戻り値	説明
		string	別のオブジェクトから新しい記事を作成するときに、ソースオブジェクトレコードの ID を返します。
	String <i>categoryGroup</i>	Void	新しい記事を作成するときに、指定したデータカテゴリーグループのデフォルトのデータカテゴリを指定します。
	String <i>category</i>		

上記のメソッドのほか、KnowledgeArticleVersionStandardController クラスは [クラス](#) に関連付けられたすべてのメソッドを継承します。次の表に、継承されるメソッドの一覧を示します。



メモ: ただし、、、および  メソッドは、継承されても KnowledgeArticleVersionStandardController クラスには使用できません。

名前	引数	戻り値	説明
	List<String>	Void	Visualforce ページが読み込まれると、Visualforce マークアップで参照される項目に基づいて、ページにアクセスできる項目が表示されます。このメソッドは、コントローラがそれらの項目にも明示的にアクセスできるように、 <a href="#"> </a> に指定された各項目に参照を追加します。  このメソッドは、レコードが読み込まれる前にコールする必要があります。通常、コントローラのコンストラクタによってコールされます。このメソッドがコンストラクタ外でコールされる場合、 <a href="#"> </a> をコールする前に <a href="#"> </a> メソッドを使用する必要があります。  の文字列には、AccountId などの API 項目名か、foo__r.myField__c などの項目への明示的なリレーションを使用できます。
			このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。
		System.PageReference	キャンセルページの PageReference を返します。
		System.PageReference	レコードを削除し、削除ページの PageReference を返します。
		System.PageReference	標準編集ページの PageReference を返します。
	string		Visualforce ページ URL の クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードの ID を返します。

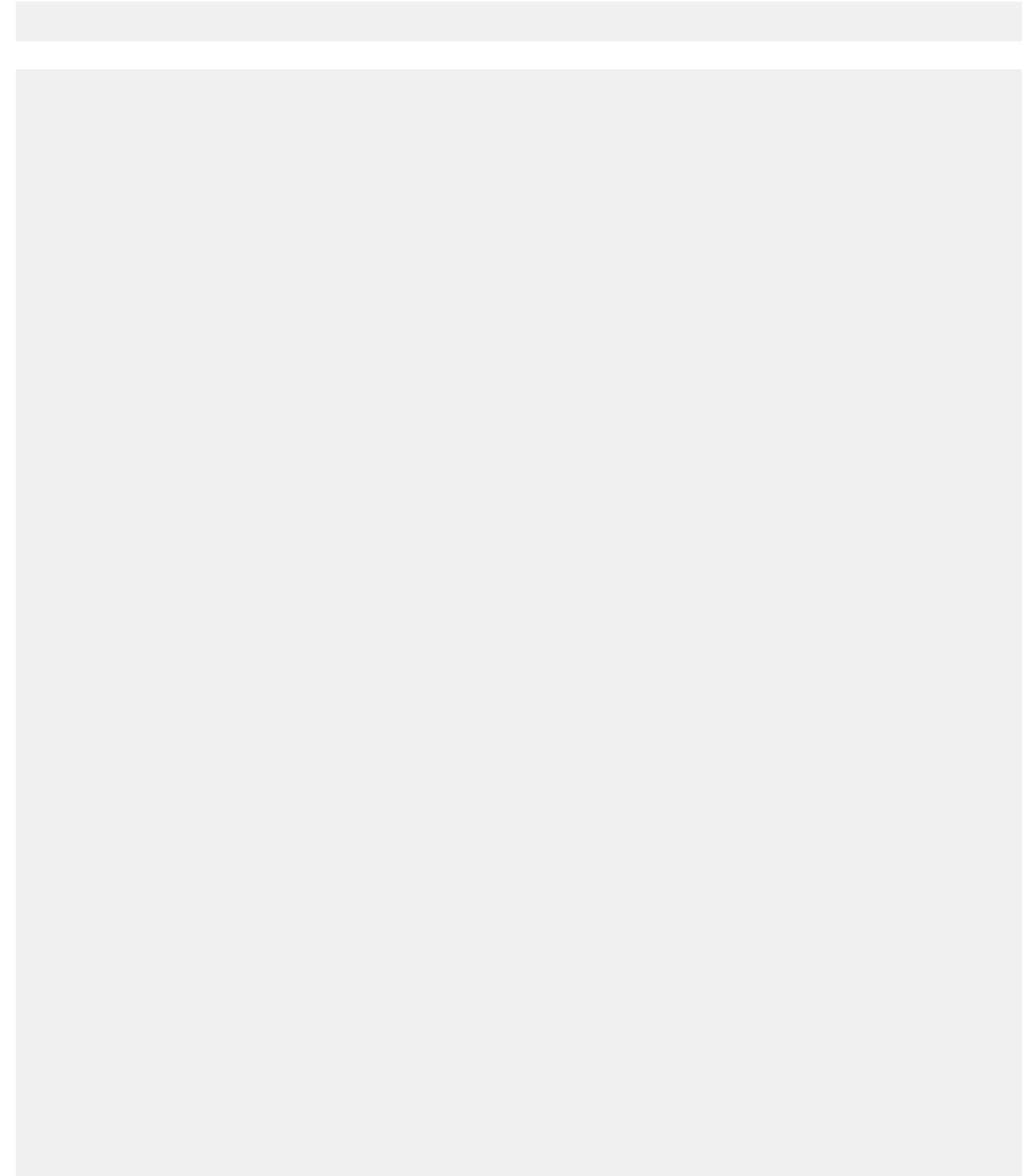
名前	引数	戻り値	説明
	SObject		<p>Visualforce ページ URL の クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードを返します。</p> <p>関連付けられた Visualforce マークアップで参照される項目のみを、この SObject でクエリすることができます。関連するオブジェクトの項目など、その他のすべての項目については、SOQL 表現を使用してクエリする必要があります。</p> <p> ヒント: クエリする任意の追加項目を参照する非表示コンポーネントを使用すれば、この制約を回避できます。コンポーネントの <code>IsVisible</code> 属性を <code>false</code> に設定して、コンポーネントを非表示にします。次に例を示します。</p>
	Void		<p>新たに参照された項目へのアクセス権限を再取得するようにコントローラを強制します。このメソッドがコールされる前にレコードに加えられた変更是、すべて破棄されます。</p> <p>これは、<code>IsAccessible</code> がコンストラクタ外でコールされる場合にのみ使用するメソッドで、<code>IsAccessible</code> がコールされる直前にコールする必要があります。</p> <p>このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。</p>
	System.PageReference	System.PageReference	変更を保存し、更新された PageReference を返します。
	System.PageReference	System.PageReference	標準詳細ページの PageReference オブジェクトを返します。

## 例

次の例では、KnowledgeArticleVersionStandardController オブジェクトを使用してカスタム拡張コントローラを作成する方法を示します。この例では、ケースのクローズ中にカスタマーサポートエージェントを使用して作成するドラフト記事に項目を自動入力できる AgentContributionArticleController というクラスを作成します。

**前提条件:**

1. 「」という記事タイプを作成します。手順は、Salesforce オンラインヘルプの「記事タイプの定義」を参照してください。
2. **詳細** というテキストカスタム項目を作成します。手順は、Salesforce オンラインヘルプの「カスタム項目の記事タイプへの追加」を参照してください。
3. 「**場所**」というカテゴリグループを作成して、「」というカテゴリに割り当てます。手順は、Salesforce オンラインヘルプの「カテゴリグループの作成と編集」および「カテゴリグループへのデータカテゴリの追加」を参照してください。
4. 「**トピック**」というカテゴリグループを作成して、「**メンテナンス**」というカテゴリに割り当てます。



前の例で説明した目的で(ケースで登録された記事の変更)カスタム拡張コントローラを作成した場合、クラスを作成した後に次の手順を実行します。

1. Salesforce 組織にログインし、[設定] で [カスタマイズ] > [ナレッジ] > [設定] をクリックします。

2. [編集] をクリックします。
3. カスタマイズを使用 項目にクラスを割り当てます。この操作により、新しいクラスに指定された記事タイプは、クローズケースに割り当てられた記事タイプに関連付けられます。
4. [保存] をクリックします。

## Message クラス

標準コントローラを使用している場合、エンドユーザがページを保存したときに発生するすべての入力規則エラー（標準およびカスタム）が自動的にページのエラーコレクションに追加されます。コンポーネントがバインドされた項目にエラーが発生すると、そのコンポーネントのエラーコレクションにメッセージが追加されます。そのページのエラーコレクションにすべてのメッセージが追加されます。詳細は、『Visualforce開発者ガイド』の「[入力規則と標準コントローラ](#)」を参照してください。

アプリケーションでカスタムコントローラや拡張を使用する場合は、エラーを収集するためのクラスを使用する必要があります。

### インスタンス化

カスタムコントローラまたはコントローラ拡張では、次のいずれかの方法でメッセージをインスタンス化できます。

- `severity summary`
- ここで、`severity` はメッセージの重要度を指定する enum で、`summary` はメッセージを要約するために使用する String です。次に例を示します。

- `severity summary detail`
- ここで、`severity` はメッセージの重要度を指定する enum、`summary` はメッセージを要約するために使用する String、`detail` はエラーに関する詳細情報を示す String です。

### メソッド

Message メソッドは、Message の特定のインスタンスからコールされ、実行されます。

次の表は、Message のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	string	関連する	コンポーネントのラベルを返します。表示ラベルが定義されていない場合、メソッドは を返します。
	string	メッセージの作成に使用する詳細パラメータの値を返します。詳細 string が指定されていない場合、このメソッドは を返します。	

名前	引数	戻り値	説明
		ApexPages.Severity	メッセージの作成に使用する重要度の enum を返します。
		string	メッセージの作成に使用する要約の String を返します。

### ApexPages.Severity Enum

enum 値を使用して、メッセージの重要度を指定します。有効な値は次のとおりです。

- 
- 
- 
- 
- 

すべての enum は、`getLabel()` や `getShortLabel()` などの標準メソッドにアクセスできます。

### PageReference クラス

PageReference は、ページのインスタンス化への参照です。多数の属性の 1 つである PageReferences は URL、一連のクエリパラメータ名および値で構成されます。

PageReference オブジェクトは次の目的で使用します。

- ・ ページのクエリ文字列パラメータおよび値を表示または設定する
- ・ ユーザを action メソッドの結果として異なるページにナビゲートする

#### インスタンス化

カスタムコントローラまたはコントローラ拡張では、次のいずれかの方法で、PageReference を参照またはインスタンス化できます。

- existingPageName***

組織すでに保存している Visualforce ページの PageReference を参照します。このプラットフォームはこのようにページを参照することで、コントローラまたはコントローラ拡張が指定されたページの有無に依存することを認識し、コントローラまたは拡張が存在する間はページが削除されないようにします。

- partialURL***

Force.com プラットフォームでホストされる任意のページに PageReference を作成します。たとえば、'partialURL' を *mySalesforceInstance* に設定すると、*recordID* にある Visualforce ページを参照します。同様に、*partialURL* を *recordID* に設定すると、指定したレコードの詳細ページを参照します。

この構文は、PageReference はコンパイル時ではなく、実行時に構成されるため、*existingPageName* のページ以外の Visualforce ページの参照にはお推めしません。実行時の参照は、参照整合性システムには使用できません。したがって、プラットフォームはこのコントローラまたはコントローラ拡張機能が指定されたページの有無に依存することを認識しないため、ユーザによるページの削除を防ぐためにエラーメッセージを表示しません。

- fullURL***

外部 URL の PageReference を作成します。次に例を示します。

ApexPages メソッドを使用して、現在のページの PageReference オブジェクトをインスタンス化することができます。次に例を示します。

## メソッド

PageReference メソッドはすべて、PageReference の特定のインスタンスからコールされ、実行されます。

次の表は、PageReference のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	string		ページの URL で参照されるアンカーの名前を返します。これは、URL のハッシュタグ (#) より後の部分です。
	Blob		Web ブラウザでユーザに表示されるページの出力を返します。返される Blob の内容は、ページの表示方法によって異なります。ページを PDF で表示すると、PDF が返されます。ページを PDF で表示しない場合、HTML が返されます。文字列として返される HTML の内容にアクセスするには、Blob メソッドを使用します。

名前	引数	戻り値	説明
			 メモ: テストメソッドで使用すると、PDF として表示される Visualforce ページを併用して空白の PDF が生成されます。
			このメソッドは、次のものには使用できません。 <ul style="list-style-type: none"> <li>• トリガ</li> <li>• スケジュール済みの Apex</li> <li>• 一括処理ジョブ</li> <li>• Test メソッド</li> <li>• Apex メールサービス</li> </ul> Visualforce ページにエラーがある場合、が発生します。
	Blob		コンポーネントの属性に関係なくページを PDF として返します。
			このメソッドは、次のものには使用できません。 <ul style="list-style-type: none"> <li>• トリガ</li> <li>• スケジュール済みの Apex</li> <li>• 一括処理ジョブ</li> <li>• Test メソッド</li> <li>• Apex メールサービス</li> </ul>
	Map<String, System.Cookie[]>		Cookie 名と Cookie オブジェクトの対応付けを返します。キーはCookie名の String で、値にはその名前を持つCookieオブジェクトのリストが含まれます。クラスと組み合わせて使用します。メソッドによって設定された「」プレフィックス付きのCookieのみを返します。
	Map<String, String>		要求ヘッダーの対応付けを返します。キー文字列にはヘッダー名が含まれ、値文字列にはヘッダーの値が含まれます。この対応付けを変更して、PageReference オブジェクトの範囲内に保持できます。たとえば、次のように指定できます。
			要求ヘッダーの説明については、「 <a href="#">要求ヘッダー</a> 」(ページ 873)を参照してください。
	Map<String, String>		ページ URL に含まれるクエリ文字列パラメータの対応付けを返します。キー文字列にはパラメー

名前	引数	戻り値	説明
			タの名前が含まれ、値文字列にはパラメータの値が含まれます。この対応付けを変更して、 <code>PageReference</code> オブジェクトの範囲内に保持できます。たとえば、次のように指定できます。
			パラメータキーでは、大文字と小文字は区別されません。次に例を示します。
		boolean	<code>PageReference</code> オブジェクトの属性の現在の値を返します。
			<code>PageReference</code> オブジェクトの URL がドメイン外の Web サイトに設定されている場合、属性がまたはどちらに設定されているかに関係なく、常にリダイレクトされます。
		string	URL が本来定義されている場合は、クエリ文字列パラメータやアンカーをすべて含む <code>PageReference</code> に関連付けられた相対 URL を返します。
<code>String Anchor</code>		<code>System.PageReference</code>	URL のアンカー参照を指定された文字列に設定します。たとえば、 <code>Salesforce_instance</code> です。
<code>Cookie[] cookies</code>	<code>Void</code>		<code>Cookie</code> オブジェクトのリストを作成します。クラスと組み合わせて使用します。
			<p><b>重要:</b></p> <ul style="list-style-type: none"> <li>Apex の <code>Cookie</code> 名と値セットは URL 符号化されています。つまり、@などの文字は % 記号および 16 進数表現に置き換えられます。</li> <li>メソッドは <code>Cookie</code> 名にプレフィックス「」を追加します。</li> </ul>

名前	引数	戻り値	説明
			<ul style="list-style-type: none"> <li>Cookie の値を _____ に設定すると、期限切れの属性の設定ではなく、空の文字列値の Cookie を送信します。</li> <li>Cookie の作成後は、Cookie のプロパティを変更することはできません。</li> <li>機密情報を Cookie に格納する場合は注意してください。Cookie の値に関係なくページはキャッシュされます。動的なコンテンツを生成するために Cookie の値を使用する場合は、ページキャッシュを無効にする必要があります。詳細は、Salesforce オンラインヘルプの「Force.com サイトページのキャッシュ」を参照してください。</li> </ul> <p>Boolean <i>redirect</i> System.PageReference PageReference オブジェクトの _____ 属性の値を設定します。 _____ に設定した場合、クライアント側のリダイレクトでリダイレクトが実行されます。この種類のリダイレクトは HTTP GET 要求を実行し、POST を使用してビューステートを更新します。 _____ に設定した場合、リダイレクトはサーバ側の転送で実行されます。これは参照先ページが同じコントローラを使用し、参照元ページで使用される拡張の適切なサブセットを含む場合にのみビューステートを維持します。</p> <p>PageReference オブジェクトの URL が _____ ドメイン外の Web サイト、または別のコントローラまたはコントローラ拡張を使用するページに設定されている場合、属性が _____ または _____ のどちらに設定されているかに関係なく、常にリダイレクトされます。</p>

## 要求ヘッダー

次の表に、要求時に設定される一部のヘッダーを示します。

ヘッダー	説明
Host	要求 URL で要求されるホスト名です。このヘッダーは、常に Force.com サイト要求および「私のドメイン」要求に設定されます。また、HTTP/1.1 ではなく、HTTP/1.0 を使用する場合、その他の要求ではこのヘッダーは省略可能です。
Referer	現在の要求の URL に含まれるか、リンクされた URL です。このヘッダーは省略可能です。

ヘッダー	説明
User-Agent	この要求を開始したプログラム (Web ブラウザなど) の名前、バージョン、拡張子のサポートです。このヘッダーは省略可能で、ほとんどのブラウザで別の値に上書きできます。そのため、信頼できるヘッダーではありません。
CipherSuite	このヘッダーが存在し、空白以外の値である場合、要求には HTTPS が使用されています。それ以外の場合、要求には HTTP が使用されています。空白以外の値の内容はこの API で定義するものではなく、予告なく変更される場合があります。
X-Salesforce-SIP	要求の要求元 IP アドレスです。このヘッダーは、Salesforce のデータセンター外で開始された HTTP 要求と HTTPS 要求に常に設定されます。   メモ: 要求が Content Delivery Network (CDN) またはプロキシサーバを通過する場合、要求元 IP アドレスは変更されて元のクライアント IP アドレスとは同じではなくになっている可能性があります。
X-Salesforce-Forwarded-To	この要求を処理している Salesforce インスタンスの完全修飾ドメイン名です。このヘッダーは、Salesforce のデータセンター外で開始された HTTP 要求と HTTPS 要求に常に設定されます。

#### 例: クエリ文字列パラメータの取得

次の例では、PageReference オブジェクトを使用して、現在の URL のクエリ文字列パラメータを取得する方法を示します。この例では、メソッドは `QueryParameters` を参照します。

次のページマークアップは、上記のコントローラからメソッドをコールします。



## メモ:

この例が正しく機能するためには、Visualforce ページを URL の有効な取引先レコードに関連付ける必要があります。たとえば、  
が取引先 ID の場合、次の URL を使用します。

**Salesforce\_instance**

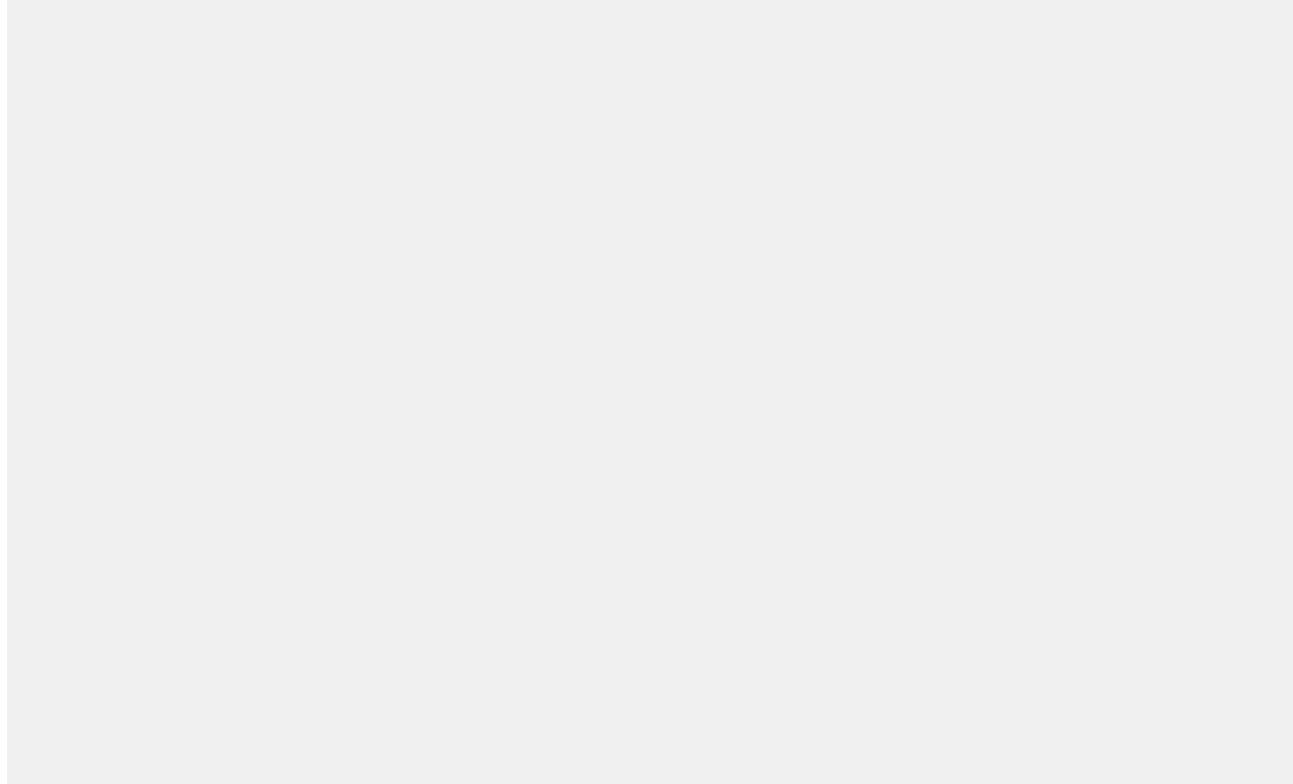
メソッドは、埋め込み SOQL クエリを使用して、ページの URL の パラメータで指定した取引先を返します。 にアクセスするために、 メソッドは次のように 名前空間を使用します。

- まず、 メソッドが現在のページの インスタンスを返します。  
は、クエリ文字列パラメータなど、Visualforce ページへの参照を返します。
- ページ参照に基づいて、 メソッドを使用して、指定されたクエリ文字列パラメータの名前と値の対応付けを返します。
- 次に、 を指定する メソッドのコールにより、 パラメータ自体の値を返します。

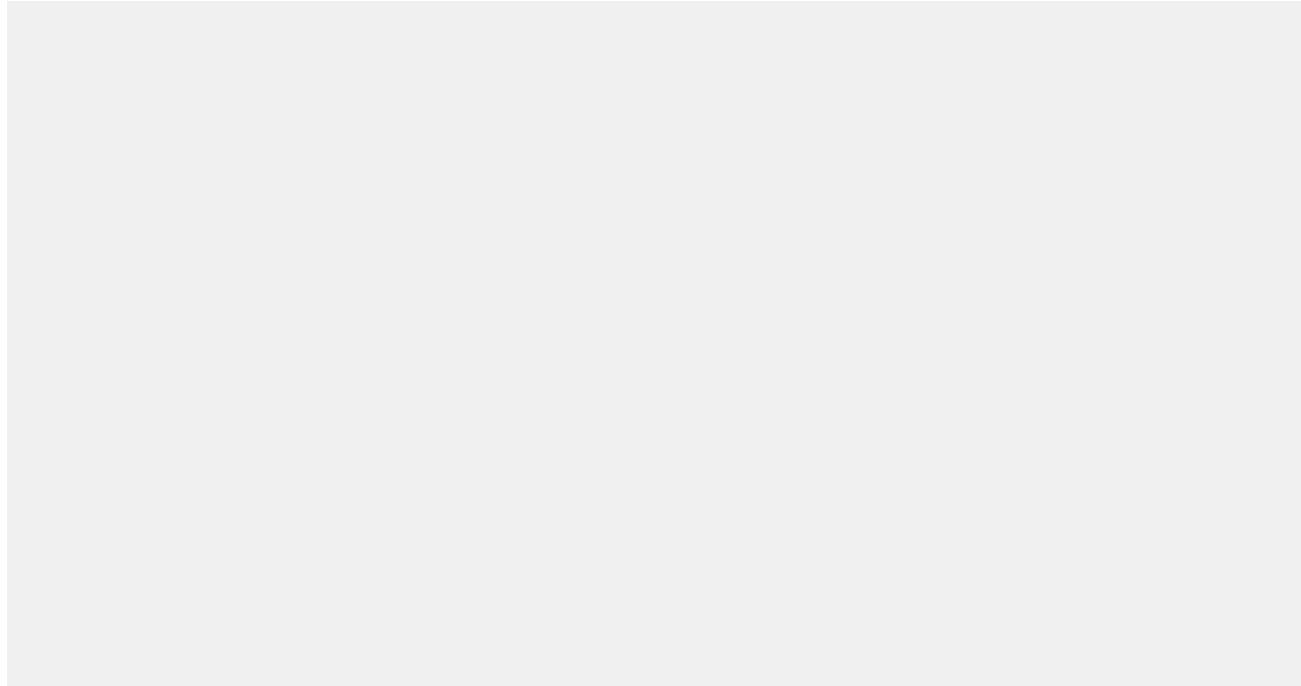
例: `action` メソッドの結果として新しいページに移動

カスタムコントローラまたはコントローラ拡張の `action` メソッドはいずれも、メソッドの結果として `PageReference` オブジェクトを返すことができます。 `PageReference` の 属性を に設定すると、`PageReference` が指定した URL に移動します。

次の例では、 メソッドでこの移動を実装する方法を示します。この例では、 メソッドで返された `PageReference` によって、ユーザは新たに保存した取引先レコードの詳細ページに移動させます。



次のページマークアップは、上記のコントローラから \_\_\_\_\_ メソッドをコールします。ユーザが [保存] をクリックすると、新たに作成した取引先の詳細ページに移動します。



## SelectOption クラス

SelectOption オブジェクトは Visualforce \_\_\_\_\_ 、 \_\_\_\_\_ 、または \_\_\_\_\_ コンポーネントに指定可能な値のいずれかを指定します。SelectOption オブジェクトは、エンドユーザーに表示されるラベルと、オプションが選択された場合にコントローラに返される値で構成されます。SelectOption は無効な状態で表示することもできます。そのため、ユーザはオプションとして選択することはできませんが、表示することはできます。

### インスタンス化

カスタムコントローラまたはコントローラ拡張では、次のいずれかの方法で、SelectOption をインスタンス化できます。

- value label isDisabled**

`value` は、ユーザがオプションを選択した場合にコントローラに返される String です。`label` は、オプション選択肢としてユーザに表示される String です。`isDisabled` は Boolean で、これを `true` に設定すると、ユーザはオプションを選択できませんが、表示することができます。

- value label**

`value` は、ユーザがオプションを選択した場合にコントローラに返される String です。`label` は、オプションの選択肢としてユーザに表示される String です。`isDisabled` の値は指定されないため、ユーザはオプションの表示と選択を行えます。

## メソッド

SelectOption メソッドはすべて、SelectOption の特定のインスタンスからコールされ、実行されます。

次の表は、SelectOption のインスタンスマソッドを示します。

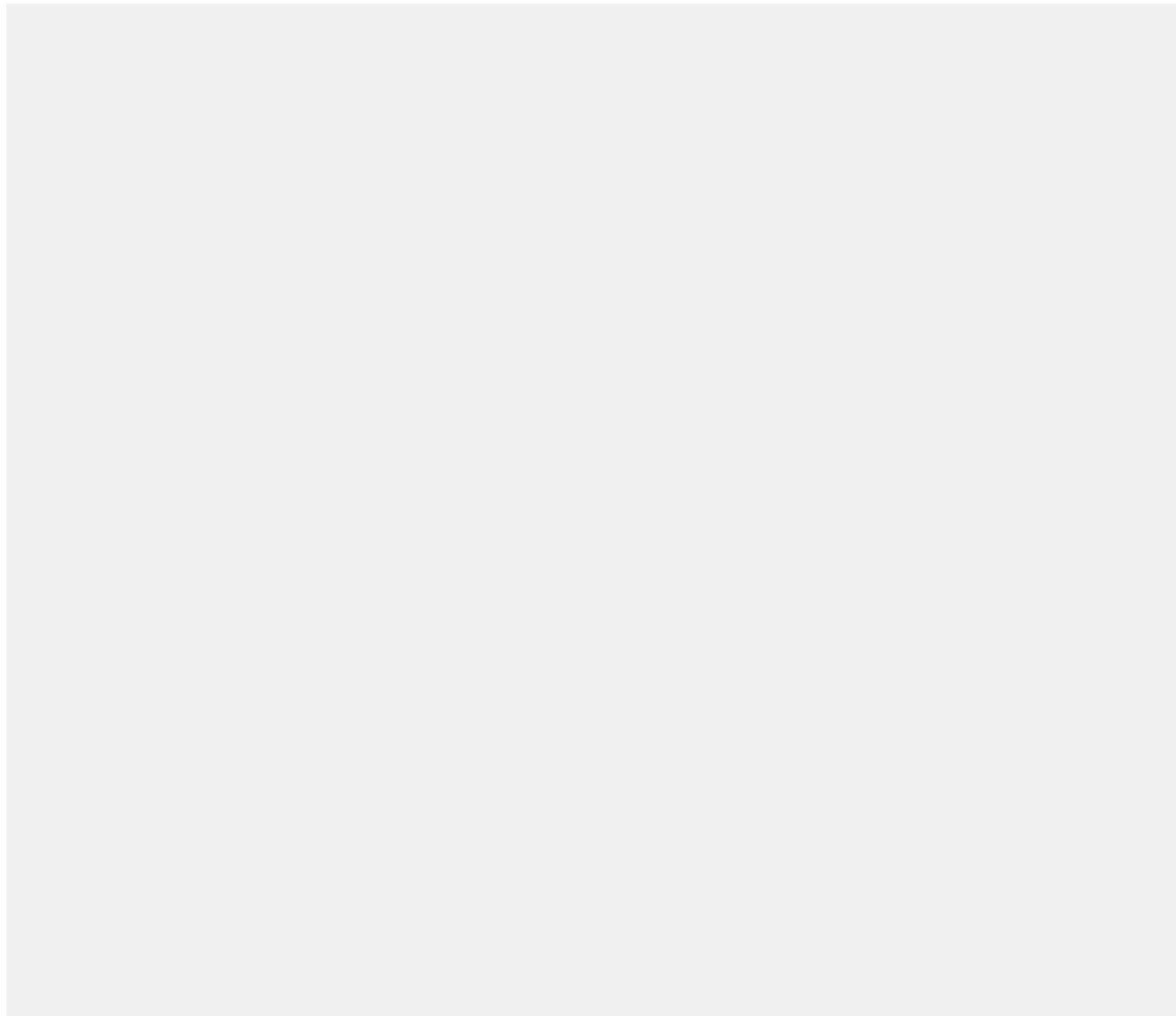
名前	引数	戻り値	説明
	boolean	SelectOption オブジェクトの属性の現在の値を返します。 を に設定した場合、オプションは表示されますが、選択できません。 を に設定した場合、オプションは表示され、選択できます。	
	boolean	SelectOption オブジェクトの属性の現在の値を返します。 を に設定した場合、重要な HTML および XML 文字はこのコンポーネントによって生成された HTML 出力でエスケープされます。 が に設定されている場合、項目は書き込まれたとおりに表示されます。	
	string	ユーザに表示されるオプションのラベルを返します。	
	string	ユーザがオプションを選択した場合にコントローラに返されるオプション値を返します。	
Boolean <i>isDisabled</i>	Void	SelectOption オブジェクトの値を設定します。 を に設定した場合、オプションは表示されますが、選択できません。 を に設定した場合、オプションは表示され、選択できます。	
Boolean <i>itemsEscaped</i>	Void	SelectOption オブジェクトの値を設定します。 を に設定した場合、重要な HTML および XML 文字はこのコンポーネントによって生成された HTML 出力でエスケープされます。 が	

名前	引数	戻り値	説明
			に設定されている場合、項目は書き込まれたとおりに表示されます。
	String <i>l</i>	Void	ユーザに表示されるオプションラベルの値を設定します。
	String <i>v</i>	Void	ユーザがオプションを選択した場合にコントローラに返されるオプション値の値を設定します。

## 例

次の例では、SelectOptions オブジェクトのリストを使用して、Visualforce ページのネントに指定可能な値を提供する方法を示します。次のカスタムコントローラでは、

コンポーネットは使用可能な SelectOption オブジェクトのリストを定義して返します。



次のページマークアップで、  
して、使用可能な値のリストを取得します。  
子であるため、オプションはチェックボックスとして表示されます。

タグは上記のコントローラの  
は、  
メソッドを使用

タグの

## StandardController クラス

StandardController オブジェクトは、salesforce.com が提供する、プリビルドされた Visualforce コントローラを参照します。StandardController オブジェクトを参照する必要があるのは、標準コントローラの拡張を定義する場合のみです。StandardController は、拡張クラスコンストラクタの单一引数のデータ型です。

### インスタンス化

次の方法で、StandardController をインスタンス化することができます。

## メソッド

StandardController メソッドはすべて、StandardController の特定のインスタンスからコールされ、実行されます。

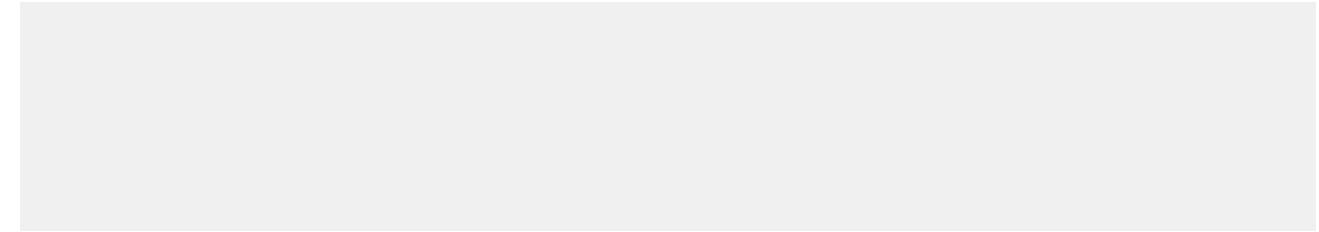
次の表は、StandardController のインスタンスマソッドを示します。

名前	引数	戻り値	説明
	List<String>	Void	<p>Visualforce ページが読み込まれると、Visualforce マークアップで参照される項目に基づいて、ページにアクセスできる項目が表示されます。このメソッドは、コントローラがそれらの項目にも明示的にアクセスできるように、に指定された各項目に参照を追加します。</p> <p>このメソッドは、レコードが読み込まれる前にコールする必要があります。通常、コントローラのコンストラクタによってコールされます。このメソッドがコンストラクタ外でコールされる場合、をコールする前にメソッドを使用する必要があります。</p> <p>の文字列には、AccountId などの API 項目名か、foo__r.myField__c などの項目への明示的なリレーションを使用できます。</p> <p>このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。</p>
	System.PageReference	System.PageReference	キャンセルページの PageReference を返します。
	System.PageReference	System.PageReference	レコードを削除し、削除ページの PageReference を返します。
	System.PageReference	System.PageReference	標準編集ページの PageReference を返します。
	string	Visualforce ページ URL の クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードの ID を返します。	
	SObject	Visualforce ページ URL の クエリ文字列パラメータの値に基づいて、現在コンテキストにあるレコードを返します。	<p>関連付けられた Visualforce マークアップで参照される項目のみを、この SObject でクエリすることができます。関連するオブジェクトの項目など、その他のすべての項目については、SOQL 表現を使用してクエリする必要があります。</p>
			 <p>ヒント: クエリする任意の追加項目を参照する非表示コンポーネントを使用すれば、この制約を回避できます。コンポーネントの属性をに設定して、</p>

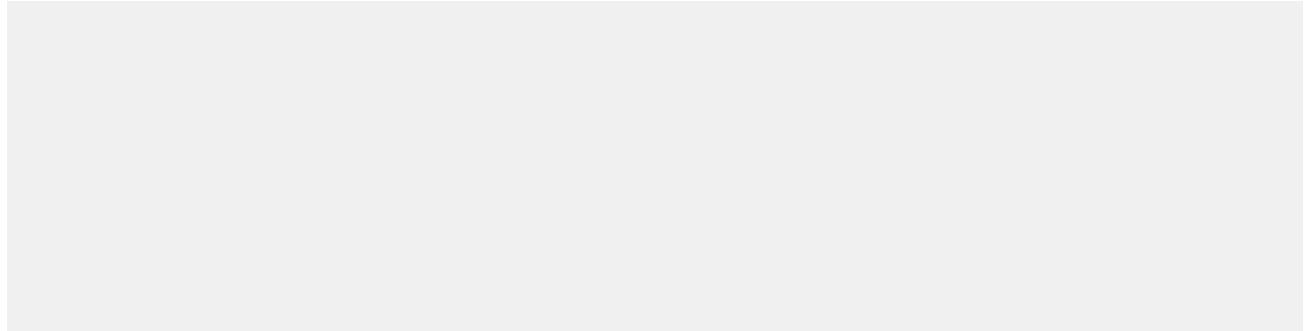
名前	引数	戻り値	説明
			コンポーネントを非表示にします。次に例を示します。
		Void	<p>新たに参照された項目へのアクセス権限を再取得するようにコントローラを強制します。このメソッドがコールされる前にレコードに加えられた変更是、すべて破棄されます。</p> <p>これは、<code>get</code> がコンストラクタ外でコールされる場合にのみ使用するメソッドで、<code>set</code> がコールされる直前にコールする必要があります。</p> <p>このメソッドは、動的な Visualforce バインドで使用されるコントローラのみに使用できます。</p>
		System.PageReference	変更を保存し、更新された PageReference を返します。
		System.PageReference	標準詳細ページの PageReference オブジェクトを返します。

## 例

次の例では、StandardController オブジェクトの標準コントローラ拡張のコンストラクタでの使用方法を示します。



次の Visualforce マークアップは、上記のコントローラ拡張をページ内で使用する方法を示します。



### **StandardSetController クラス**

StandardSetController オブジェクトを使用して、Salesforce が提供する、プリビルドされた Visualforce リストコントローラと同様のリストコントローラ、またはその拡張としてリストコントローラを作成することができます。

クラスには、プロトタイプオブジェクトも含まれます。これは、Visualforce の StandardSetController クラスに含まれる単一の `standardSetController` です。プロトタイプオブジェクトの項目が設定されている場合、それらの値は、保存操作中に使用されます。つまり、値は設定されたコントローラコレクションのすべてのレコードに適用されます。これは、一括更新(オブジェクトのコレクション内の項目に同一の変更を適用)を実行するページを記述するときに役立ちます。



メモ: 他の Salesforce オブジェクトに必要な項目は、プロトタイプオブジェクトに使用される場合にも必要です。

y ばニ k q p ?要で蛻翠碘 火ヨ睡ベ闇ホ 蛭悦 世セにナ' 顱クレズにニズ嘆築や パベみ 要リ韜悔ツイ; 黥韓 蛮翠



**メモ:** StandardSetController のレコード数の上限は 10,000 件です。10,000 件を超えるレコードを返すクエリロケータを使用して StandardSetController をインスタンス化すると、LimitException が発生します。ただし、10,000 件を超えるレコードのリストを使用して StandardSetController をインスタンス化すると、例外が発生する代わりに、レコードが上限まで切り捨てられます。

## メソッド

StandardSetController メソッドはすべて、StandardSetController の特定のインスタンスからコールされ、実行されます。

次の表は、StandardSetController のインスタンスマソッドを示します。

名前	引数	戻り値	説明
		System.PageReference	元のページ(わかっている場合)、またはホームページの PageReference を返します。
		Void	レコードの最初のページを返します。
		boolean	セット内に存在するレコード数がレコード数の上限を超えてるかどうかを示します。false の場合、レコード数がリストコントローラを使用して処理できる数を超えています。レコード数の上限は 10,000 レコードです。
		string	現在のコンテキストでの検索条件の ID を返します。
		boolean	現在のページセットの後に、より多くのレコードがあるかどうかを示します。
		boolean	現在のページセットの前に、より多くのレコードがあるかどうかを示します。
		System.SelectOption[]	現在のユーザが使用できるリストビューのリストを返します。
		Integer	現在のページセットのページ番号を返します。最初のページは 1 を返します。
		Integer	各ページセットに存在するレコード数を返します。
		sObject	選択したレコードへの変更を示す sObject を返します。クラス内に含まれるプロトタイプオブジェクトを取得し、一括更新の実行に使用されます。
		sObject[]	現在のページセットにある sObject のリストを返します。このリストは不变であるため、() をコールできません。
		Integer	セットに存在するレコード数を返します。
		sObject[]	選択されている sObject のリストを返します。
		Void	レコードの最後のページを返します。

名前	引数	戻り値	説明
		Void	レコードの次のページを返します。
		Void	レコードの前のページを返します。
		System.PageReference	新しいレコードを挿入するか、変更された既存のレコードを更新します。この操作が完了した後、元のページ(わかっている場合)、またはホームページの PageReference を返します。
	String <i>filterId</i>	Void	コントローラの検索条件 ID を設定します。
	Integer <i>pageNumber</i>	Void	ページ番号を設定します。
	Integer <i>pageSize</i>	Void	各ページセット内のレコード数を設定します。
	sObjects [] <i>selectedRecords</i>	Void	選択したレコードを設定します。

## 例

次の例では、StandardSetController オブジェクトのカスタムリストコントローラのコンストラクタでの使用方法を示します。

次の Visualforce マークアップは、上記のコントローラをページ内で使用する方法を示します。

## XML クラス

次のクラスを使用して、XML コンテンツの読み取りおよび書き込みを行います。

- [XmlStream クラス](#)
- [DOM クラス](#)

### XmlStream クラス

XmlStream メソッドを使用して、XML 文字列の読み書きを行います。

- [クラス](#)
- [クラス](#)

### XmlStreamReader クラス

StAX の XMLStreamReader ユーティリティクラスと同様に、[XmlStreamReader クラス](#)は、XML データの転送と読み込み専用アクセスを可能にします。データを XML からプルし、余分なイベントをスキップします。

次のコードスニペットは、新しい XmlStreamReader オブジェクトのインスタンス化の方法を示しています。

これらのメソッドは、次の XML イベント上で動作します。

- 属性イベントは、特定の要素のために指定されます。たとえば、要素 `<parent>` には、属性 `id="123"` があります。
- 要素開始イベントは、要素用の開始タグです。例: `<parent>`
- 要素終了イベントは、要素用の終了タグです。例: `</parent>`

- ドキュメント開始イベントは、ドキュメント用の開始タグです。
- ドキュメント終了イベントは、ドキュメント用の終了タグです。
- エンティティ参照は、コード内のエンティティ参照です。例:
- 文字イベントは、テキスト文字です。
- コメントイベントは、XML ファイル内のコメントです。

XML データを繰り返し処理するには、メソッドとメソッドを使用します。メソッドなどのメソッドを使用して XML 内のデータにアクセスします。



メモ: Apex 内の

クラスは、Java で相当するクラスに基づいています。  
を参照してください。

次のメソッドは、XML ファイルの読み取りをサポートしています。

名前	引数	戻り値	説明
		Integer	開始要素上の属性の番号を返します。このメソッドは、開始要素または属性 XML イベント上でのみ有効です。この値は、名前空間の定義を除外します。属性 XML イベント用の属性の番号は、0 で始まります。
	Integer <i>index</i>	string	指定したインデックスで属性のローカル名を返します。名前がない場合、空白の文字列が返されます。このメソッドは、開始要素または属性 XML イベントでのみ有効です。
	Integer <i>index</i>	string	指定したインデックスで属性の名前空間 URI を返します。名前空間がない場合、null が返されます。このメソッドは、開始要素または属性 XML イベントでのみ有効です。
	Integer <i>index</i>	string	指定したインデックスでこの属性のプレフィックスを返します。プレフィックスがない場合、null が返されます。このメソッドは、開始要素または属性 XML イベントでのみ有効です。
	Integer <i>index</i>	string	指定したインデックスで属性の XML の型を返します。たとえば、は属性型です。このメソッドは、開始要素または属性 XML イベントでのみ有効です。
	String <i>namespaceURI</i>	string	特定の URI で指定された <i>localName</i> 内の属性の値を返します。値が見つからない場合 null を返します。 <i>localName</i> の値を指定する必要があります。このメソッドは、開始要素または属性 XML イベントでのみ有効です。
	String <i>localName</i>		



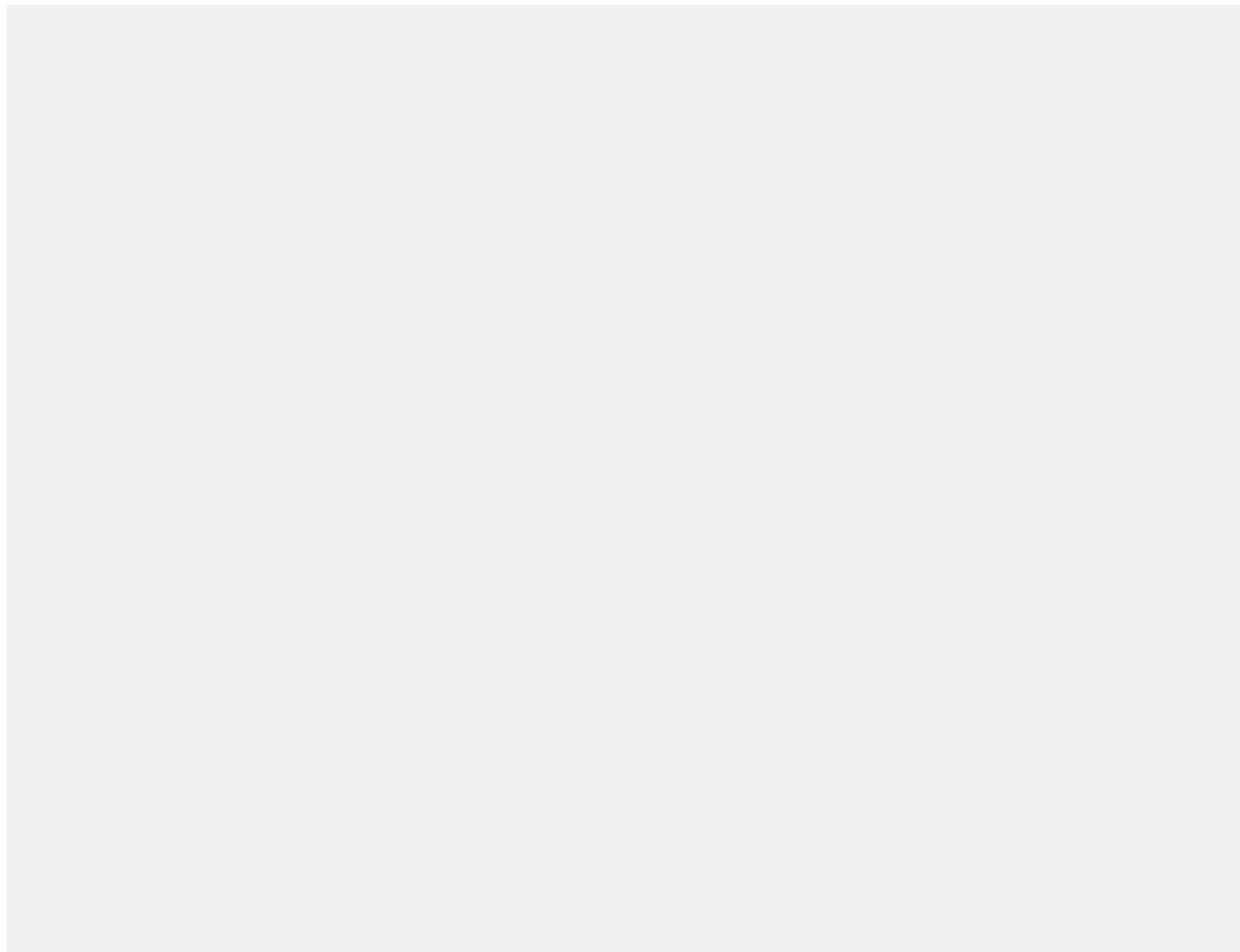
名前	引数	戻り値	説明
	Integer <i>index</i>	string	インデックスで宣言された名前空間のプレフィックスを返します。これがデフォルトの名前空間宣言の場合、null を返します。このメソッドは、開始要素、終了要素、または名前空間 XML イベント上でのみ有効です。
	String <i>Prefix</i>	string	特定のプレフィックス用の URI を返します。返される URI は、プロセッサの状態によって異なります。
	Integer <i>Index</i>	string	インデックスで宣言された名前空間の URI を返します。このメソッドは、開始要素、終了要素、または名前空間 XML イベント上でのみ有効です。
		string	処理方法のデータセクションを返します。
		string	処理方法の対象セクションを返します。
		string	イベントにプレフィックスがない場合、現在の XML イベントのプレフィックスまたは null が返されます。
		string	文字列として XML イベントの現在の値が返されます。異なるイベントに対する有効値は次のとおりです。 <ul style="list-style-type: none"> <li>文字 XML イベントの文字列値</li> <li>コメントの文字列値</li> <li>エンティティ参照のための置換値 たとえば、 が次の XML スニペットを読むとします。</li> </ul>
			メソッドは、 ではなく、 を返します。 <ul style="list-style-type: none"> <li>CDATA セクションの文字列値</li> <li>空白 XML イベントの文字列値</li> <li>DTD の内部サブセットの文字列値</li> </ul>

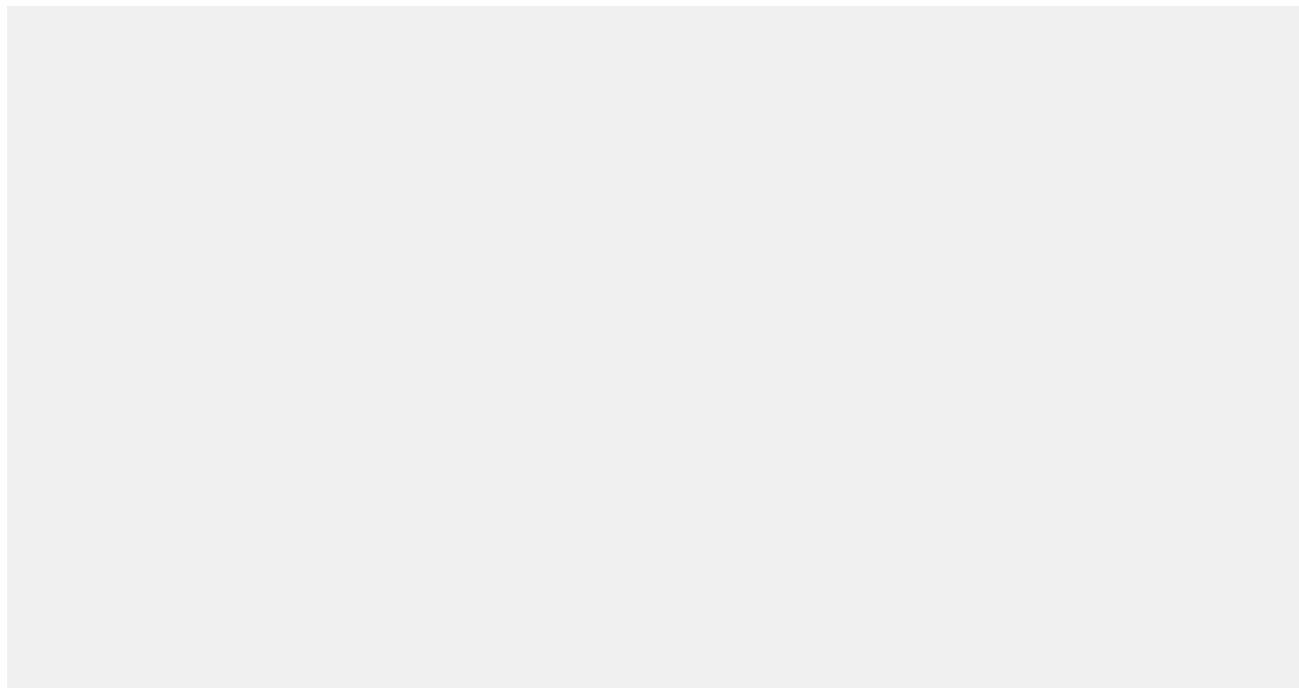
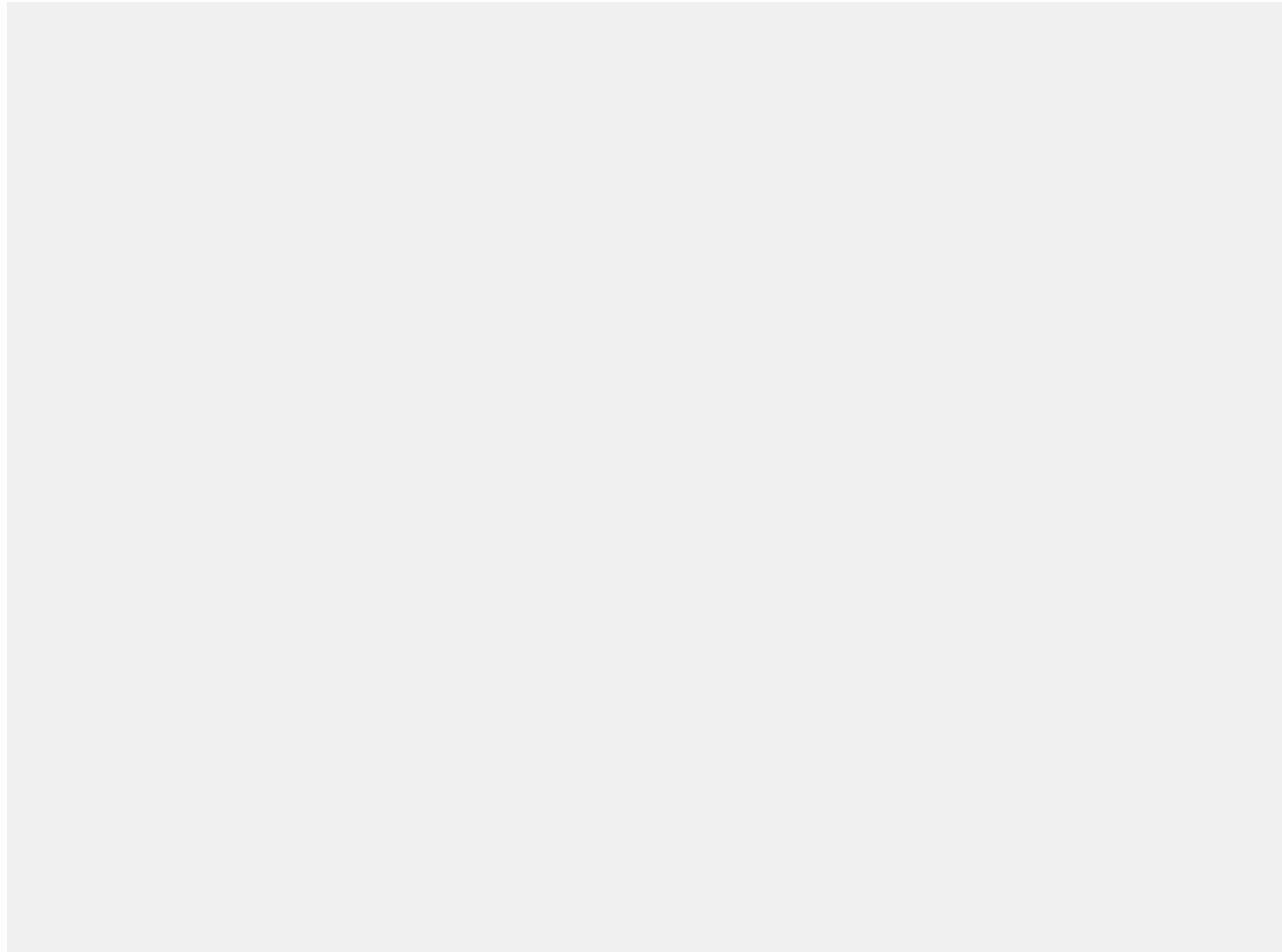
名前	引数	戻り値	説明
		string	XML 宣言で指定された XML バージョンを返します。何も宣言されていない場合、null を返します。
		boolean	現在の XML イベントに名前がある場合、を返します。ない場合は を返します。このメソッドは、開始要素または終了要素の XML イベントでのみ有効です。
		boolean	さらに XML イベントがある場合は 、それ以上 XML イベントがない場合は を返します。現在の XML イベントが終了ドキュメントの場合、このメソッドは、を返します。
		boolean	現在のイベントにテキストがある場合 を返し、その他の場合、 を返します。文字、エンティティ参照、コメントおよび空白の XML イベントにはテキストがあります。
		boolean	カーソルが文字データ XML イベントを指し示している場合、 を返します。ない場合は を返します。
		boolean	カーソルが終了タグを指し示している場合、 を返します。ない場合は を返します。
		boolean	カーソルが開始タグを指し示している場合、 を返します。ない場合は を返します。
		boolean	カーソルが、すべての空白を含む文字データ XML イベントを指し示している場合、 を返します。ない場合は を返します。
		Integer	次の XML イベントを読み取ります。プロセッサは、単一ブロック内のすべての連続文字データを返すか、いくつかのチャンクに分割します。イベントのタイプを示す整数を返します。
		Integer	( メソッドが を返す) 空白、コメント、または処理命令 XML イベントを、開始要素または終了要素に到達するまでスキップします。XML イベント用のインデックスを返します。空白、コメント、処理命令、開始要素または終了要素以外の要素があると、このメソッドでエラーが生成されます。
Boolean <i>returnAsSingleBlock</i>		Void	<i>returnAsSingleBlock</i> に対して を指定した場合、開始要素から最初の終了要素または次の

名前	引数	戻り値	説明
			開始要素のいずれか先にくる方に、テキストが单一ブロックで返されます。 <i>isNamespaceAware</i> と指定した場合は、パーサーは、複数のブロック内でテキストを返します。
Boolean <i>isNamespaceAware</i>	Void		<i>isNamespaceAware</i> に対して <i>true</i> を指定した場合、パーサーは、名前空間を認識します。 として指定した場合、パーサーは認識しません。 デフォルト値は、 <i>false</i> です。
	string		に指定された入力 XML の長さと入力 XML の最初の 50 文字を含む文字列を返します。

### XmIStreamReader の例

次の例のように XML 文字列は処理されます。





## XmlStreamWriter クラス

StAX の XMLStreamReader ユーティリティクラスと同様に、  
みを可能にします。たとえば、  
クラスを使用して、XML ドキュメントをプログラムで作成で  
きます。次に、[HTTP クラス](#)を使用して、ドキュメントを外部サーバに送ります。

次のコードスニペットは、新しい XmlStreamWriter のインスタンス化の方法を示しています。



メモ: Apex 内の

クラスは、Java で相当するクラスに基づいています。

を参照してください。

次のメソッドは、XML ファイルの書き込みのサポートに利用可能です。

名前	引数	戻り値	説明
		Void	XmlStreamWriter のインスタンスを閉じ、それに 関連付けられたリソースを解放します。
		string	XmlStreamWriter インスタンスで書き込まれた XML を返します。
	String <i>URI</i>	Void	指定された URI をデフォルトの名前空間にバインドします。この URI は、現在の START_ELEMENT – END_ELEMENT ペアの 範囲でバインドします。
	String <i>prefix</i> String <i>namespaceURI</i> String <i>localName</i> String <i>value</i>	Void	属性を出力ストリームに書き込みます。 <i>localName</i> は、属性名を指定します。
	String <i>data</i>	Void	指定された CData を出力ストリームに書き込み ます。

名前	引数	戻り値	説明
	String <i>text</i>	Void	指定されたテキストを出力ストリームに書き込みます。
	String <i>data</i>	Void	指定されたコメントを出力ストリームに書き込みます。
	String <i>namespaceURI</i>	Void	指定された名前空間を出力ストリームに書き込みます。
	String <i>prefix</i> String <i>localName</i> String <i>namespaceURI</i>	Void	空白要素のタグを出力ストリームに書き込みます。 <i>localName</i> は、書き込むタグの名前を指定します。
		Void	開始タグを閉じて、対応する終了タグを出力ストリームに書き込みます。
		Void	終了タグを出力ストリームに書き込みます。プレフィックスとローカル名を決定する writer の内部状態に依存します。
	String <i>prefix</i> String <i>namespaceURI</i>	Void	指定された名前空間を出力ストリームに書き込みます。
	String <i>target</i> String <i>data</i>	Void	指定された処理命令を書き込みます。
	String <i>encoding</i> String <i>version</i>	Void	指定された XML エンコーディングとバージョンを使用して、XML 宣言を書き込みます。
	String <i>prefix</i> String <i>localName</i> String <i>namespaceURI</i>	Void	<i>localName</i> によって指定された開始タグを出力ストリームに書き込みます。

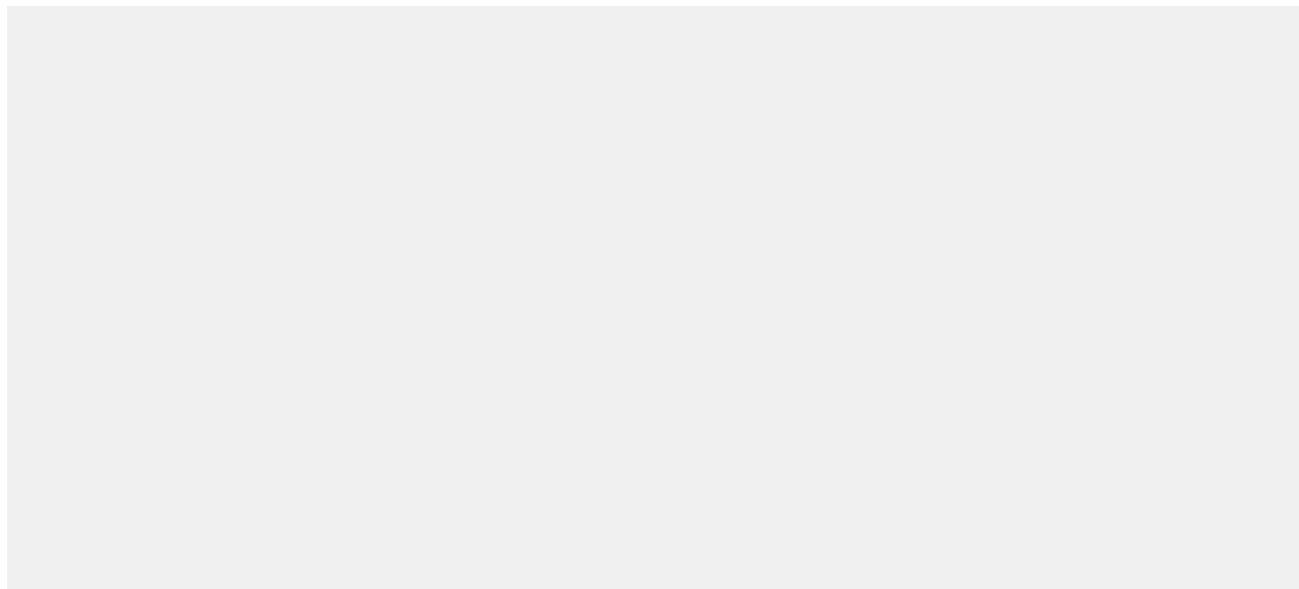
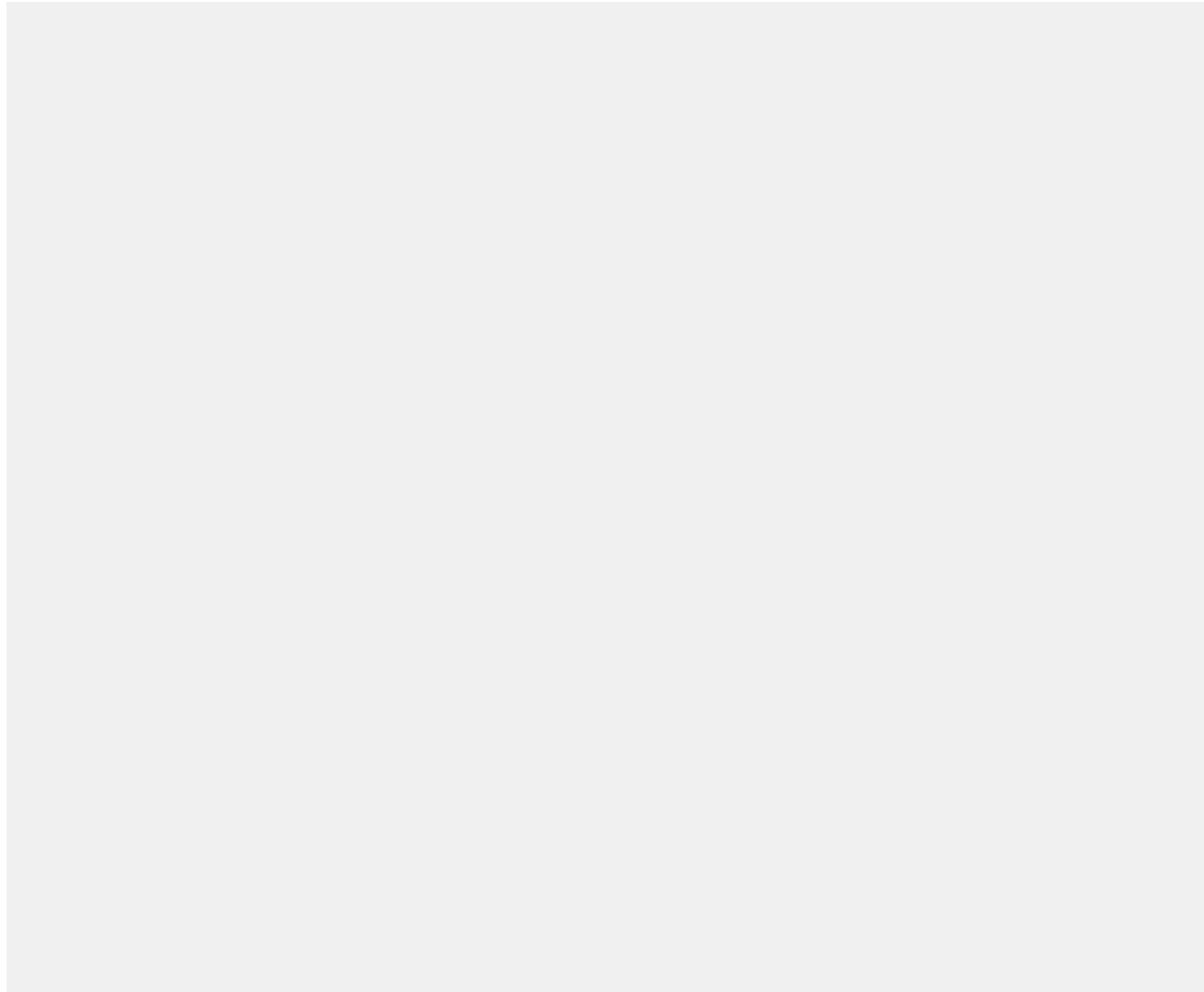
## XML ライターメソッド例

次の例では、XML ドキュメントを書き込み、その妥当性をテストします。



メモ: Hello World と納入先請求書のサンプルでは、カスタム項目およびオブジェクトが必要です。項目やオブジェクトを自分で作成したり、オブジェクト、項目および Apex コードを管理パッケージとして Force.com AppExchange からダウンロードできます。詳細は、

を参照してください。



## DOM クラス

DOM (ドキュメントオブジェクトモデル) クラスを使用して、XML コンテンツを解析または生成できます。これらのクラスを使用して、XML コンテンツを処理できます。ある一般的なアプリケーションでは、このクラスを使用して **Document** クラスを使用して XML ドキュメントの本文の内容を処理します。ある一般的なアプリケーションでは、このクラスを使用して **Element** クラスを使用して XML ドキュメントのノードを処理します。このクラスを使用して XML ドキュメントをノードの階層として示します。分岐ノードで子ノードがあるノードもあれば、葉ノードで子ノードがないものもあります。

DOM クラスは **Document** クラスと **Element** クラスを使用して XML ドキュメントを処理します。

**Document** クラスを使用して XML ドキュメントの本文の内容を処理します。

**Element** クラスを使用して XML ドキュメントのノードを処理します。

### Document クラス

Document クラスを使用して XML コンテンツを処理します。ある一般的なアプリケーションでは、このクラスを使用して **Document** クラスを使用して XML ドキュメントの本文の内容を処理します。ある一般的なアプリケーションでは、このクラスを使用して **Element** クラスを使用して XML ドキュメントのノードを処理します。このクラスを使用して XML ドキュメントをノードの階層として示します。分岐ノードで子ノードがあるノードもあれば、葉ノードで子ノードがないものもあります。

### XML 名前空間

XML 名前空間は、URI 参照で識別される名前のコレクションで XML ドキュメントで使用され、要素の種類や属性名を一意に特定します。XML 名前空間の名前は修飾名として示される場合があり、コロンを使用して、名前を名前空間プレフィックスとローカルの部分に分割します。URI 参照に対応付けられたプレフィックスは、名前空間を選択します。管理された URI 名前空間とドキュメント独自の名前空間を組み合わせて、一意の識別子を作成します。

次の XML 要素には、**namespace** 属性と **prefix** 属性があります。

次の例では、XML 要素に 2 つの属性があります。

- 最初の属性には、**namespace** 属性があります。値は **http://www.w3.org/2000/svg** です。
- 2 番目の属性には、**prefix** 属性があります。値は **svg** です。

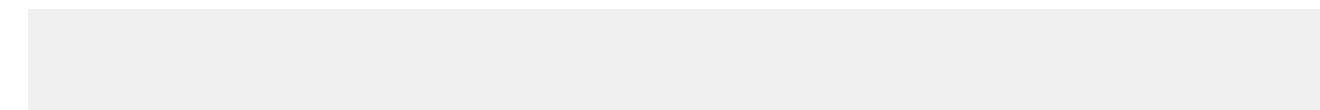
### メソッド

Document クラスには次のメソッドがあります。

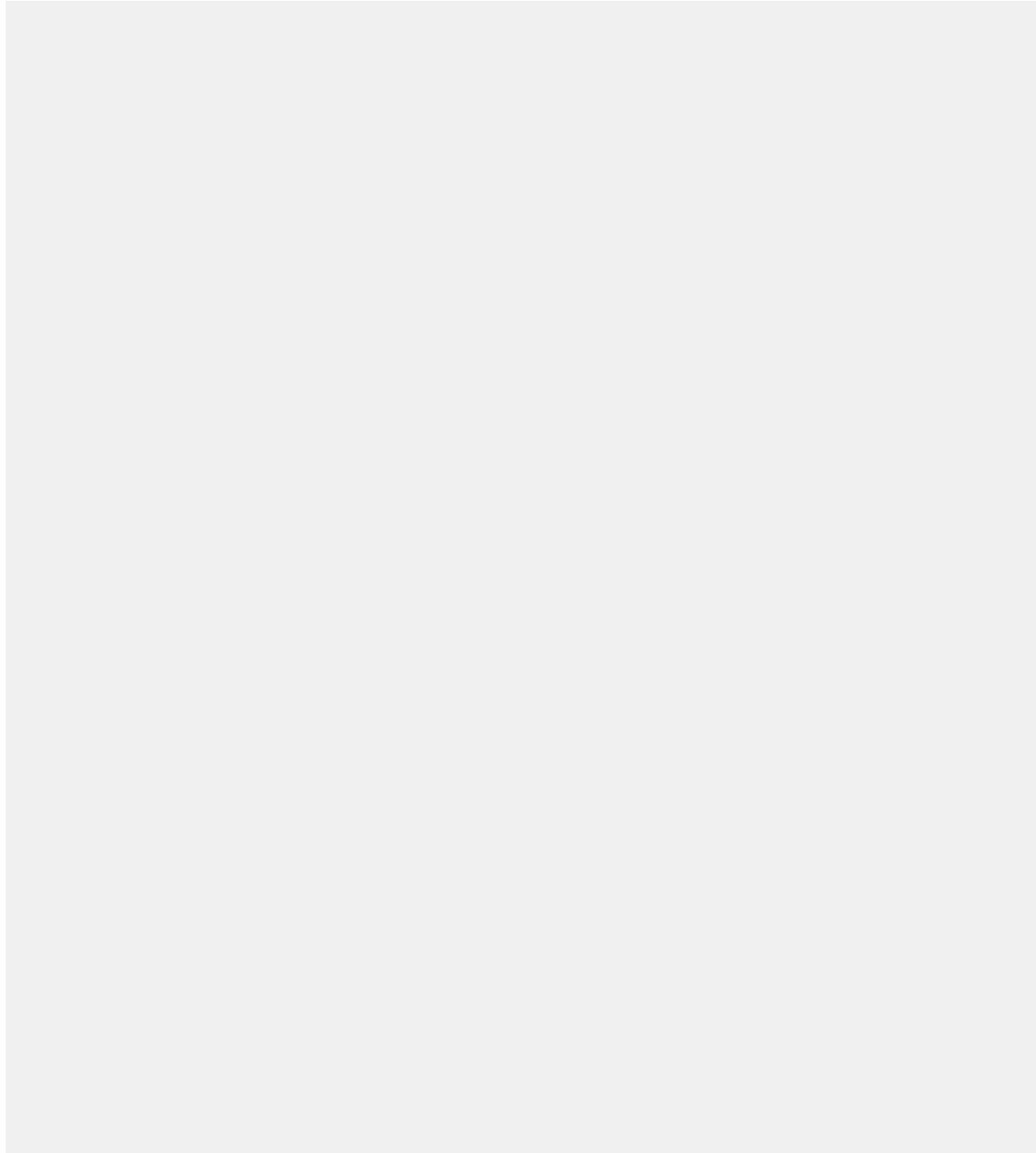
名前	引数	戻り値	説明
	String <i>name</i>		ドキュメントの最上位のルート要素を作成します。
	String <i>namespace</i>		<i>name</i> 引数には 値を設定できません。
	String <i>prefix</i>		<i>namespace</i> 引数に 以外の値があり、 <i>prefix</i> 引数が である場合、名前空間はデフォルトの名前空間として設定されます。 <i>prefix</i> 引数が である場合、要素に自動的にプレフィックスが割り当てられます。自動プレフィックスの形式は <i>i</i> で、 <i>i</i> は番号を示します。 <i>prefix</i> 引数が である場合、名前空間はデフォルトの名前空間として設定されます。 名前空間についての詳細は、「XML 名前空間」(ページ 895)を参照してください。 ドキュメントでこのメソッドを複数回コールすると、ドキュメントに指定できるルート要素は 1 つだけであるため、エラーが発生します。
			ドキュメントの最上位のルート要素を返します。 このメソッドが を返す場合、ルート要素はまだ作成されていません。
	String <i>xml</i>	Void	<i>xml</i> 引数で指定されたドキュメントの XML の表示を解析し、ドキュメントに読み込みます。次に例を示します。
		string	ドキュメントの XML 表示を文字列として返します。

### Document の例

この例では、 に渡される 引数が次の XML 応答を返すと想定します。



次の例では、DOM クラスを使用して リクエストボディで返される XML 応答をどのように解析するかを示しています。



## XmlNode クラス

クラスを使用して XML ドキュメントのノードを処理します。DOM は、XML ドキュメントをノードの階層として示します。分岐ノードで子ノードがあるノードもあれば、葉ノードで子ノードがないものもあります。

### ノードの種類

Apex で使用できるさまざまな種類の DOM ノードがあります。

は、これらの種類の列挙です。値

は次のとおりです。

- COMMENT
- ELEMENT
- TEXT

XML ドキュメントでは、要素とノードを区別することが重要です。次に、XML の簡単な例を示します。

この例には、`<parent>`、`<parent>child</parent>`、`<parent>child</parent><parent>child</parent>` の 3 つの XML 要素が含まれています。`<parent>`、`<parent>child</parent>`、`<parent>child</parent><parent>child</parent>` の 3 つの要素ノード、`<parent>`、`<parent>child</parent>` の 2 つのテキストノード、合計 5 つのノードが含まれています。要素ノード内のテキストは、個別のテキストノードとみなされます。

すべての enum で共有されるメソッドの詳細は、「[Enum メソッド](#)」(ページ 517)を参照してください。

### メソッド

クラスには次のメソッドがあります。

名前	引数	戻り値	説明
	<code>String name</code>	<code>Dom.XmlNode</code>	このノードの子要素ノードを作成します。
	<code>String namespace</code>		<code>name</code> 引数には 値を設定できません。
	<code>String prefix</code>		<code>namespace</code> 引数に 以外の値があり、 <code>prefix</code> 引数が である場合、名前空間はデフォルトの名前空間として設定されます。
			<code>prefix</code> 引数が である場合、要素に自動的にプレフィックスが割り当てられます。自動プレフィックスの形式は <code>i</code> で、 <code>i</code> は番号を示します。

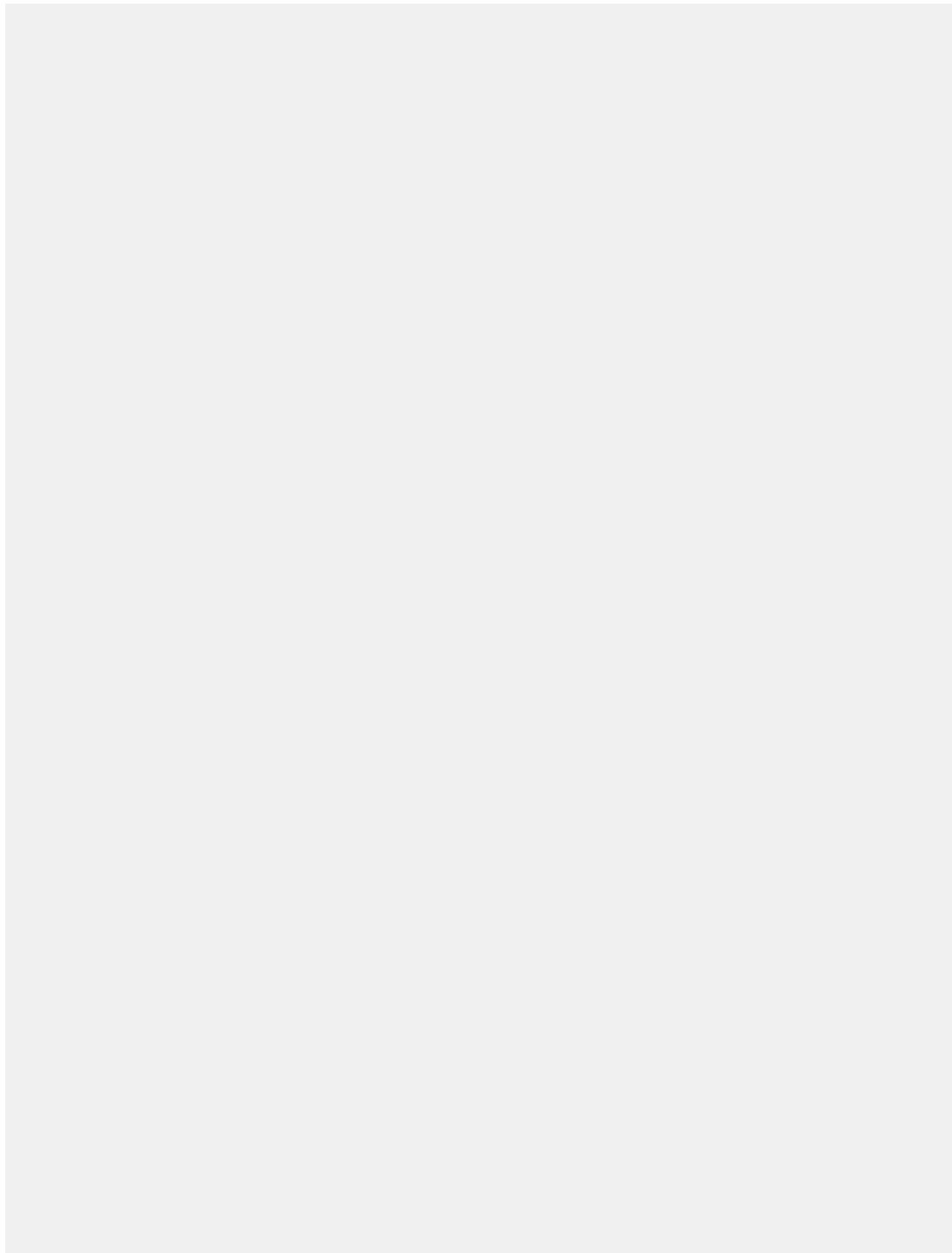
名前	引数	戻り値	説明
			<i>prefix</i> 引数が　である場合、名前空間はデフォルトの名前空間として設定されます。
	String <i>text</i>	Dom.XmlNode	このノードの子コメントノードを作成します。 <i>text</i> 引数には　値を設定できません。
	String <i>text</i>	Dom.XmlNode	このノードの子テキストノードを作成します。 <i>text</i> 引数には　値を設定できません。
	String <i>key</i> String <i>keyNamespace</i>	string	指定された <i>key</i> と <i>keyNamespace</i> の <i>namespacePrefix:attributeValue</i> を返します。 たとえば、要素では、次の ようになります。 ・　　は　を返す ・　　は　を返す
		Integer	このノードの属性の数を返します。
	Integer <i>index</i>	string	指定された <i>index</i> の属性キーを返します。インデックス値は 0 から始まります。
	Integer <i>index</i>	string	指定された <i>index</i> の属性キーの名前空間を返します。詳細は、「 <a href="#">XML 名前空間</a> 」(ページ 895)を参照してください。
	String <i>key</i> String <i>keyNamespace</i>	string	指定された <i>key</i> と <i>keyNamespace</i> の属性値を返します。 たとえば、要素では、次の ようになります。 ・　　は　を返す ・　　は　を返す
	String <i>key</i> String <i>keyNamespace</i>	string	指定された <i>key</i> と <i>keyNamespace</i> の属性値の名前空間を返します。詳細は、「 <a href="#">XML 名前空間</a> 」を参照してください。
	String <i>name</i> String <i>namespace</i>	Dom.XmlNode	指定された <i>name</i> と <i>namespace</i> を含むノードの子要素ノードを返します。
		Dom.XmlNode[]	このノードの子要素ノードを返します。これには子テキストまたはコメントノードは含まれません。詳細は、「 <a href="#">ノードの種類</a> 」を参照してください。
		Dom.XmlNode[]	このノードの子ノードを返します。これにはすべてのノードの種別が含まれます。詳細は、「 <a href="#">ノードの種類</a> 」を参照してください。

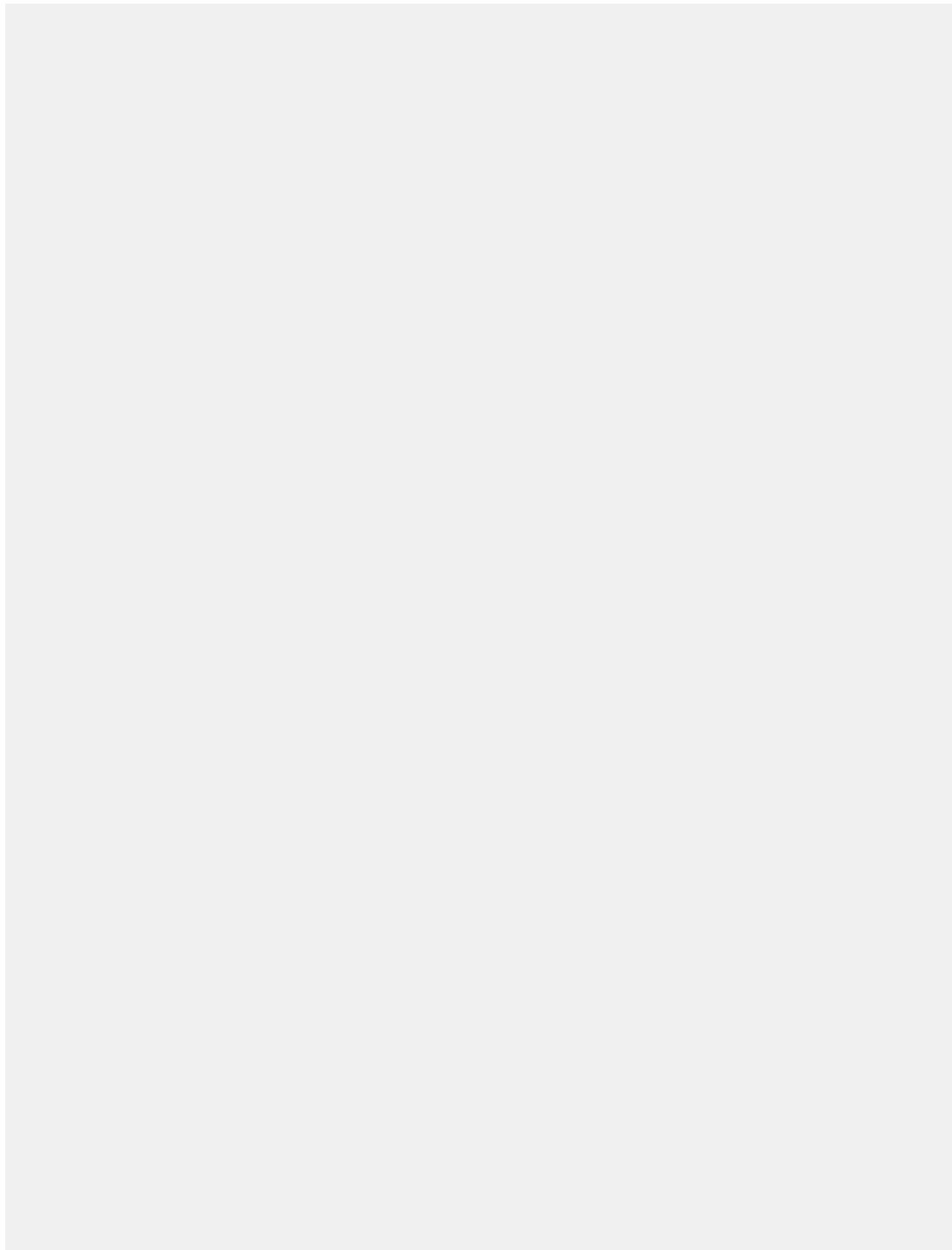
名前	引数	戻り値	説明
		string	要素の名前を返します。
		string	要素の名前空間を返します。詳細は、「 <a href="#">XML 名前空間</a> 」を参照してください。
	String <i>prefix</i>	string	指定された <i>prefix</i> の要素の名前空間を返します。 詳細は、「 <a href="#">XML 名前空間</a> 」を参照してください。
		Dom.XmlNodeType	<a href="#">ノードの種類</a> を返します。
		Dom.XmlNode	要素の親を返します。
	String <i>namespace</i>	string	指定された <i>namespace</i> のプレフィックスを返します。 <i>namespace</i> 引数には 値を設定できません。 詳細は、「 <a href="#">XML 名前空間</a> 」を参照してください。
		string	このノードのテキストを返します。
	String <i>key</i>	boolean	指定された <i>key</i> と <i>keyNamespace</i> の属性を削除します。 成功した場合は 、失敗した場合は を返します。 詳細は、「 <a href="#">XML 名前空間</a> 」を参照してください。
	String <i>keyNamespace</i>		
	Dom.XmlNode <i>childNode</i>	boolean	指定された <i>childNode</i> を削除します。
	String <i>key</i>	Void	<i>key</i> 属性値を設定します。
	String <i>value</i>		
	String <i>key</i>	Void	<i>key</i> 属性値を設定します。 詳細は、「 <a href="#">XML 名前空間</a> 」を参照してください。
	String <i>value</i>		
	String <i>keyNamespace</i>		
	String <i>valueNamespace</i>		
	String <i>prefix</i>	Void	指定された <i>prefix</i> の <i>namespace</i> を設定します。 詳細は、「 <a href="#">XML 名前空間</a> 」を参照してください。
	String <i>namespace</i>		

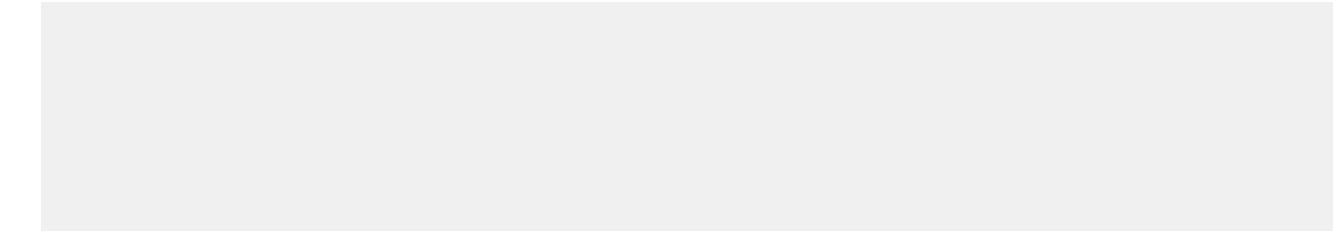
### XmlNode の例

この例では、 メソッドおよび名前空間を使用して XML 要求を作成する方法を示します。

メソッドを使用した基本的な例は、「 [クラス](#)」(ページ 895)を参照してください。







## Apex インターフェース

Apex には、次のシステム定義インターフェースがあります。

- Salesforce では、Salesforce へのシングルサインオンに Facebook<sup>®</sup>、Janrain<sup>®</sup> などの認証プロバイダを使用する機能を提供します。シングルサインオンを設定するには、[インターフェースを実装するクラスを作成する必要があります。](#) 認証プロバイダ定義に `登録ハンドラ` として指定され、Facebook などのサードパーティのサービスから Salesforce ポータルと組織へのシングルサインオンを有効にします。
- インターフェースは、非プリミティブ型を含むリスト、つまりユーザ定義型のリストの並び替えのサポートを追加します。Apex クラスのリスト並び替えのサポートを追加するには、[インターフェースを、そのメソッドと共にクラスに実装する必要があります。](#)
- Apex の一括処理は、インターフェースとして公開され、開発者によって実行される必要があります。一括処理ジョブは実行時に Apex を使用してプログラムで起動できます。
  - [インターフェース](#) インターフェースでは、HTTP コールアウトをテストするときに擬似応答を送信できます。
  - [ループ](#) イテレータは、コレクション内のすべての項目を辿ります。たとえば、Apex のループで、ループを終了する条件を定義し、コレクションを辿るいくつかの方法、つまりイテレータを提供する必要があります。
  - アプリケーション開発者は、このインターフェースを実装して、登録者が管理パッケージをインストールまたはアップグレードした後に自動的に実行される Apex コードを指定できます。これにより、登録者の組織の詳細に基づいてパッケージのインストールまたはアップグレードをカスタマイズできます。たとえば、スクリプトを使用して、カスタム設定の入力、サンプルデータの作成、インストーラへのメール送信、外部システムへの通知、または大きなデータセットに新しい項目を入力するための一括処理操作の起動などができます。

Apex メールサービスドメインが受信するすべてのメールについて、Salesforce は、そのメールの内容と添付ファイルを含む個別の InboundEmail オブジェクトを作成します。

イン

インターフェースを実装する Apex クラスを使用して、受信メールメッセージを処理できます。そのクラスでメソッドを使用して、InboundEmail オブジェクトにアクセスし、受信メールメッセージの内容、ヘッダー、および添付ファイルの取得と、その他多数の機能を実行することができます。

- は組み込みインターフェースで、組織内のデータを処理し、指定のフローにデータを渡すことができます。

- 特定の時間に実行されるように Apex クラスを呼び出すには、まずクラスを実装し、Salesforce ユーザインターフェースの [Apex をスケジュール] ページまたはソッドのいずれかを使用してスケジュールを指定します。

#### インターフェース

インターフェースでは、ケースフィードのデフォルトのメールテンプレートを指定できます。デフォルトのメールテンプレートを使用すると、ケース発生源や件名などの条件に基づいて、指定したメールテンプレートがケースに事前に読み込まれます。

- アドレスバーに入力したり、ブックマークから起動したり、または外部 Web サイトからリンクする URL 要求を再記述するルールを作成します。サイトページ内のリンクの URL を再記述するルールも作成できます。URL を再記述すると、URL がわかりやすくなるだけでなく、ユーザが直感的に理解できるようになるため、検索エンジンによるサイトページのインデックス作成がさらに容易になります。

- アプリケーション開発者は、このインターフェースを実装して、登録者が管理パッケージをアンインストールした後に自動的に実行される Apex コードを指定できます。これにより、登録者の組織の詳細に基づいてクリーンアップおよび通知タスクを実行できます。

#### インターフェース

インターフェースでは、WSDL から自動生成されたクラスの Web サービスコールアウトをテストするときに擬似応答を送信できます。

## Auth.RegistrationHandler インターフェース

Salesforce では、Salesforce へのシングルサインオンに Facebook<sup>®</sup>、Janrain<sup>®</sup> などの認証プロバイダを使用する機能を提供します。シングルサインオンを設定するには、を実装するクラスを作成する必要があります。

インターフェースを実装するクラスは、認証プロバイダ定義

に 登録ハンドラ として指定され、Facebook などのサードパーティのサービスから Salesforce ポータルと組織へのシングルサインオンを有効にします。クラスは、認証プロバイダの情報を使用して、関連付けられた取引先レコードと取引先責任者レコードも含めて、ユーザデータの作成と更新のロジックを必要に応じて実行する必要があります。

名前	引数	戻り値	説明

```
locale  
provider  
siteLoginUrl  
attributeMap
```

のプロパティは次のとあります。

プロバイダについての詳細は、Salesforce オンラインヘルプの「外部認証プロバイダについて」を参照してください。

認証プロバイダとして Janrain を使用する場合、Janrain 辞書の値を使用してアクセストークンまたは同等の項目を取得する必要があります。Janrain でサポートされている一部のプロバイダのみがアクセストークンを提供し、その他のプロバイダは他の項目を使用します。Janrain の項目は、  
クラスの 变数で返されます。  
」のドキュメントを参照してください。

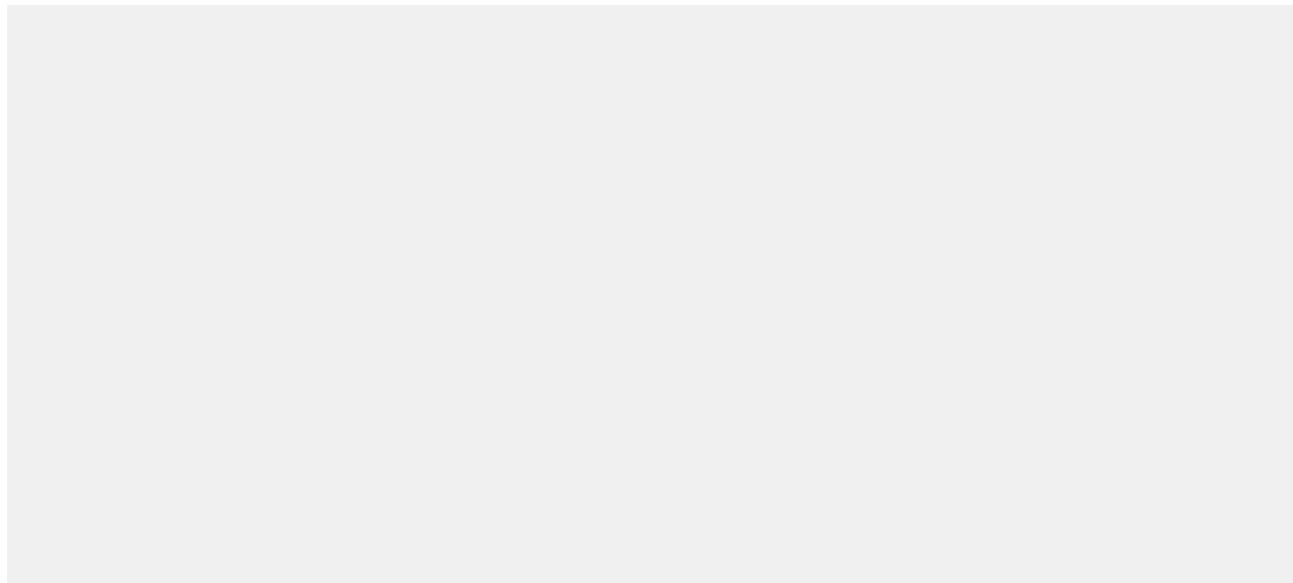


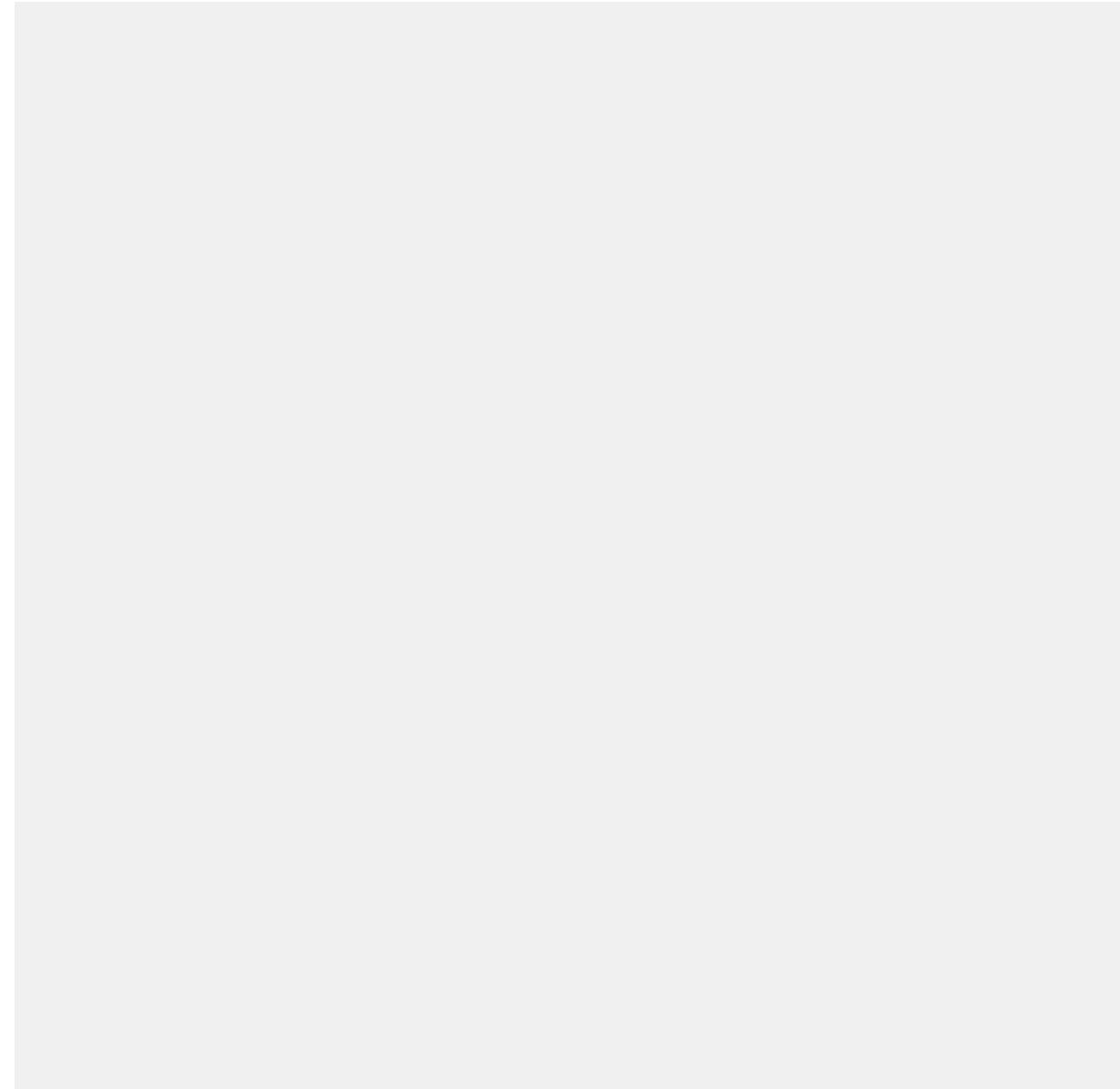
メモ: すべての Janrain のアカウントの種類が を返すとは限りません。情報を受信するのに、アカウントの種類の変更が必要な場合があります。

名前	引数	戻り値	説明
	String <i>authProviderId</i> String <i>providerName</i>	String	組織の認証プロバイダ定義に指定された 18 文字の ID を使用している現在のユーザのアクセストークンと、Salesforce、Facebook などのプロバイダの名前を返します。
	String <i>authProviderId</i> String <i>providerName</i>	Map<String, String>	現在ログインしている Salesforce ユーザサードパーティ識別子からアクセストークンへの対応付けを返します。識別子の値はサードパーティにより異なります。たとえば、Salesforce ではユーザ ID ですが、Facebook ではユーザ番号です。

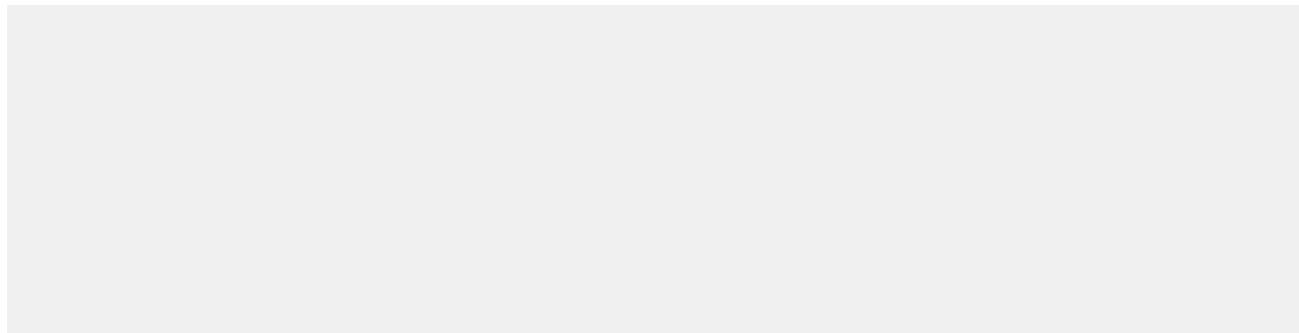
## 実装例

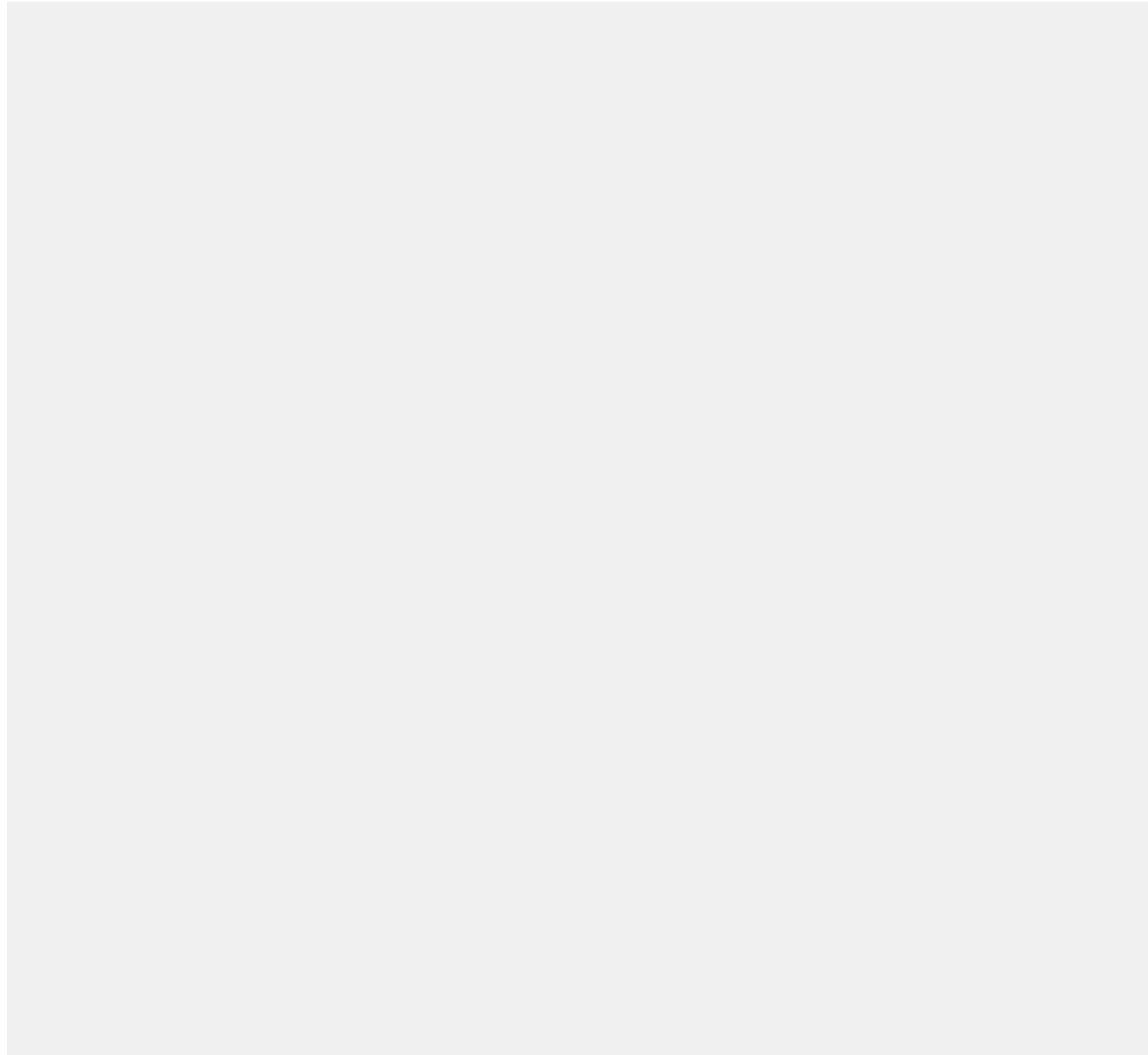
この例では、認証プロバイダが提供するデータに基づいて標準ユーザを作成および更新するインターフェースを実装します。ここでは、例を単純化するためにエラーチェックを省略しています。





次の例では、上記のコードをテストします。





## Comparable インターフェース

インターフェースは、非プリミティブ型を含むリスト、つまりユーザ定義型のリストの並び替えのサポートを追加します。

Apex クラスのリスト並び替えのサポートを追加するには、メソッドと共にクラスに実装する必要があります。

インターフェースを、その

インターフェースには次のメソッドが含まれます。

名前	引数	戻り値	説明
	Object <i>objectToCompareTo</i>	Integer	<p>比較の結果である integer 値を返します。このメソッドの実装では、次の値を返す必要があります。</p> <ul style="list-style-type: none"><li>このインスタンスと <i>objectToCompareTo</i> が等しい場合は 0</li><li>このインスタンスが <i>objectToCompareTo</i> より大きい場合は 1 以上</li><li>このインスタンスが <i>objectToCompareTo</i> より小さい場合は 0 未満</li></ul>

インターフェースを実装するには、最初に必要があります。

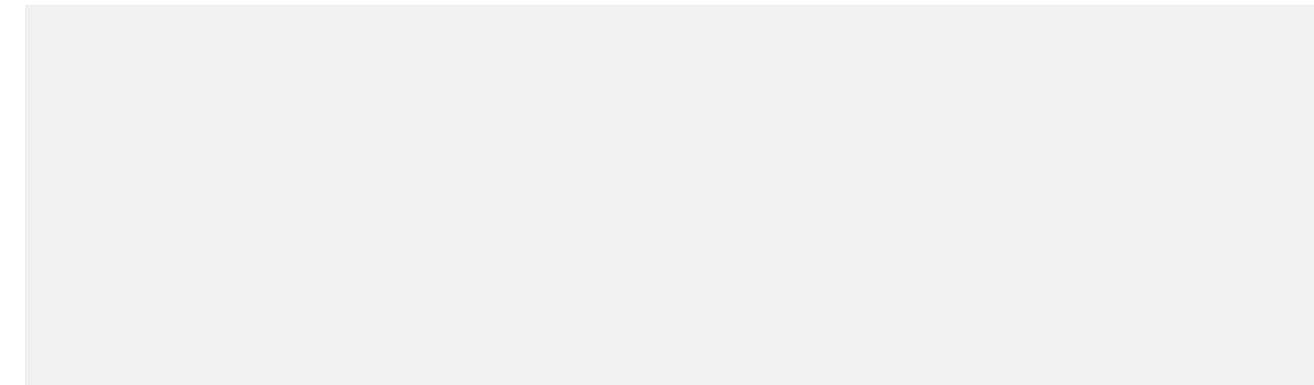
キーワードでクラスを次のように宣言する

```
implements Comparable
```

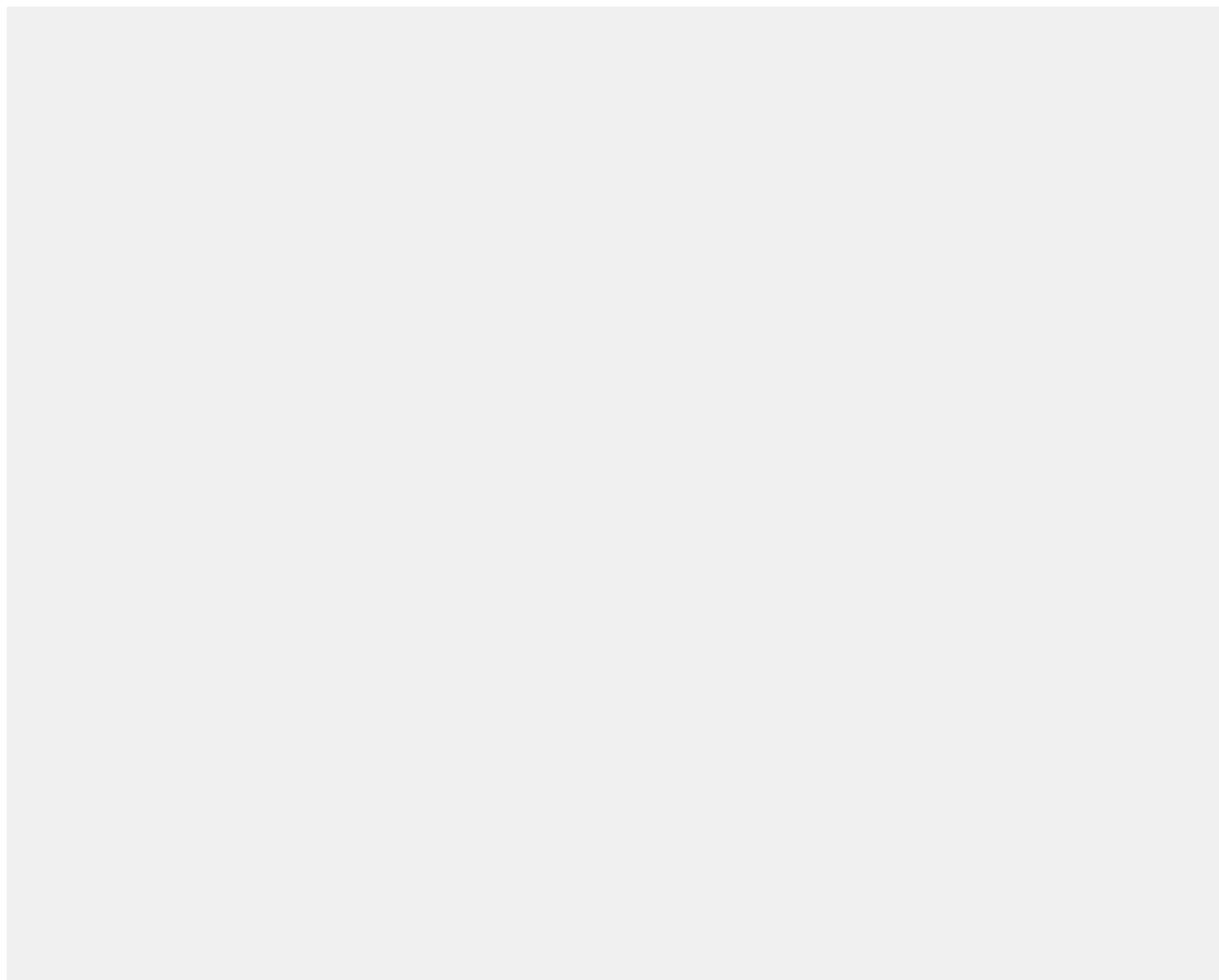
次に、クラスで次のメソッドの実装を提供する必要があります。

実装されたメソッドは      または      として宣言する必要があります。

これは、      インターフェースの実装例です。この例の      メソッドは、このクラスインスタンスの従業員を引数で渡された従業員と比較します。メソッドは、従業員 ID の比較に基づいて integer 値を返します。



この例では、オブジェクトのリストの並び替え順をテストします。



## 関連リンク

[List メソッド](#)

## HttpCalloutMock インターフェース

インターフェースでは、HTTP コールアウトをテストするときに擬似応答を送信できます。このインターフェースには次のメソッドが含まれます。

名前	引数	戻り値	説明
	HttpRequest <i>req</i>	HttpResponse	指定された要求の HTTP 応答を返します。このメソッドの実装は Apex ランタイムによってコールされ、がコールされた後に HTTP コールアウトが実行されたときに擬似応答を送信します。

実装例は、「[インターフェースの実装による HTTP コールアウトのテスト](#)」を参照してください。

## InstallHandler インターフェース

アプリケーション開発者は、このインターフェースを実装して、登録者が管理パッケージをインストールまたはアップグレードした後に自動的に実行される Apex コードを指定できます。これにより、登録者の組織の詳細に基づいてパッケージのインストールまたはアップグレードをカスタマイズできます。たとえば、スクリプトを使用して、カスタム設定の入力、サンプルデータの作成、インストーラへのメール送信、外部システムへの通知、または大きなデータセットに新しい項目を入力するための一括処理操作の起動などができます。

インストール後スクリプトは、テストを実行した後に呼び出され、デフォルトのガバナ制限が適用されます。パッケージを象徴する特殊なシステムユーザとして実行するため、スクリプトによって実行されるすべての操作は、パッケージによって行われているように見えます。このユーザには、UserInfo を使用してアクセスできます。このユーザは実行時にのみ確認でき、テストの実行中には確認できません。

スクリプトが失敗すると、インストール/アップグレードは中止されます。スクリプト内のエラーは、パッケージの [Apex エラーを通知] 項目に指定されたユーザにメールされます。ユーザが指定されていない場合、インストール/アップグレードの詳細は利用できません。

インストール後スクリプトには、他に次のような特性があります。

- 一括処理ジョブ、スケジュール済みジョブ、および今後のジョブを開始できます。
- セッション ID にアクセスできません。
- async 操作を使用してコールアウトのみを実行できます。コールアウトは、スクリプトが実行され、インストールが完了およびコミットされた後に実行されます。

インターフェースには、  
アクションを指定する単一のメソッドがあります。

という、インストール/アップグレード時に実行されるア

メソッドは、次の情報を提供するコンテキストオブジェクトを引数として取ります。

- ・ インストールが実施される組織の組織 ID
- ・ インストールを開始したユーザのユーザ ID
- ・ 以前にインストールされたパッケージのバージョン番号 (　　クラスを使用して指定)。これは、1.2.0 の  
ように、常に 3 つの番号で構成されています。
- ・ インストールがアップグレードかどうか
- ・ インストールがプッシュかどうか

コンテキスト引数は、データ型が  
インターフェースは、システムによって自動的に実装されます。  
コンテキスト引数にコールできるメソッドを示しています。

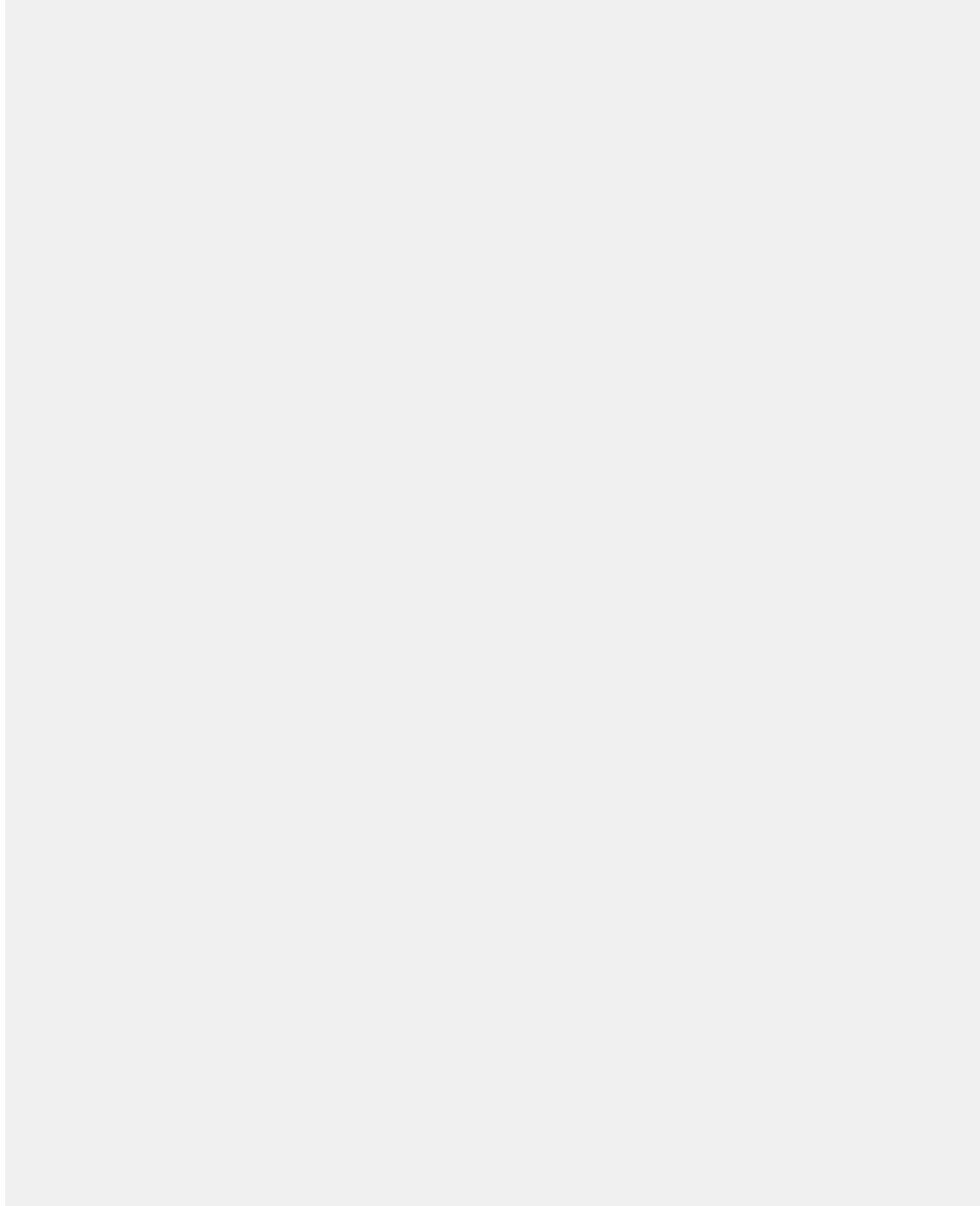
インターフェースであるオブジェクトです。このインター  
インターフェースの次の定義では、コ

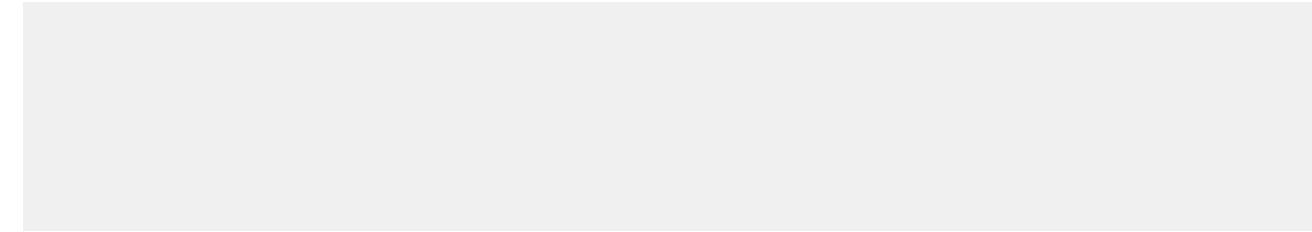
## インストール後スクリプトの例

次のインストール後スクリプトのサンプルは、パッケージのインストール/アップグレード時に次のアクション  
を実行します。

- ・ 以前のバージョンが null である場合、つまりパッケージが初めてインストールされている場合、スクリプト  
は次を行う
  - ◊ 「Newco」という新しいアカウントを作成し、作成されたことを検証する。
  - ◊ 「Client Satisfaction Survey」というカスタムオブジェクト Survey の新しいインスタンスを作成する。
  - ◊ 登録者に、パッケージのインストールを確認するメールメッセージを送信する。
- ・ 以前のバージョンが 1.0 である場合、「Upgrading from Version 1.0」という Survey の新しいインスタンスを作  
成する。
- ・ パッケージがアップグレードである場合、「Sample Survey during Upgrade」という Survey の新しいインスタン  
スを作成する。

- ・ アップグレードがプッシュで実行されている場合、「Sample Survey during Push」という Survey の新しいインスタンスを作成する。

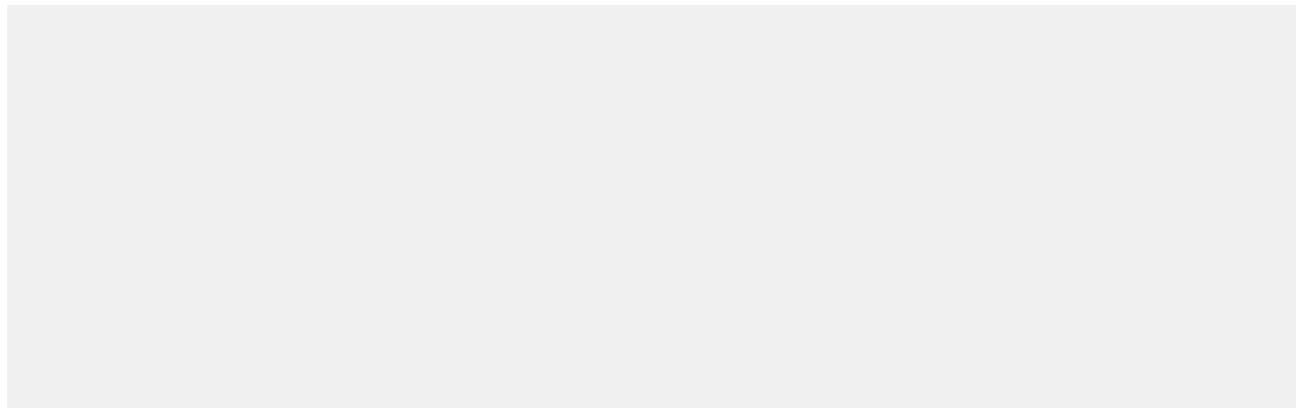




インストール後スクリプトは、**Support.EmailTemplateSelector** クラスの新しい  
ソッドが取る引数は、次のとあります。

- 既存のパッケージのバージョン番号を指定する **version** メソッド
- インストールがブッシュである場合は **isInstallable** である省略可能な Boolean 値。デフォルトは **false** です。

このサンプルでは、**getTemplateId** Apex クラスに実装されたインストール後スクリプトのテスト方法を説明しています。



## Support.EmailTemplateSelector インターフェース

インターフェースでは、ケースフィールドのデフォルトのメールテンプレートを指定できます。デフォルトのメールテンプレートを使用すると、ケース発生源や件名などの条件に基づいて、指定したメールテンプレートがケースに事前に読み込まれます。

デフォルトのテンプレートを指定するには、**getTemplateId** を実装するクラスを作成する必要があります。インターフェースには次のメソッドが含まれます。

名前	引数	戻り値	説明
	ID <b>caseId</b>	ID	指定したケース ID を使用して、ケースフィールドで現在表示されているケースに事前に読み込まれるメールテンプレートの ID を返します。

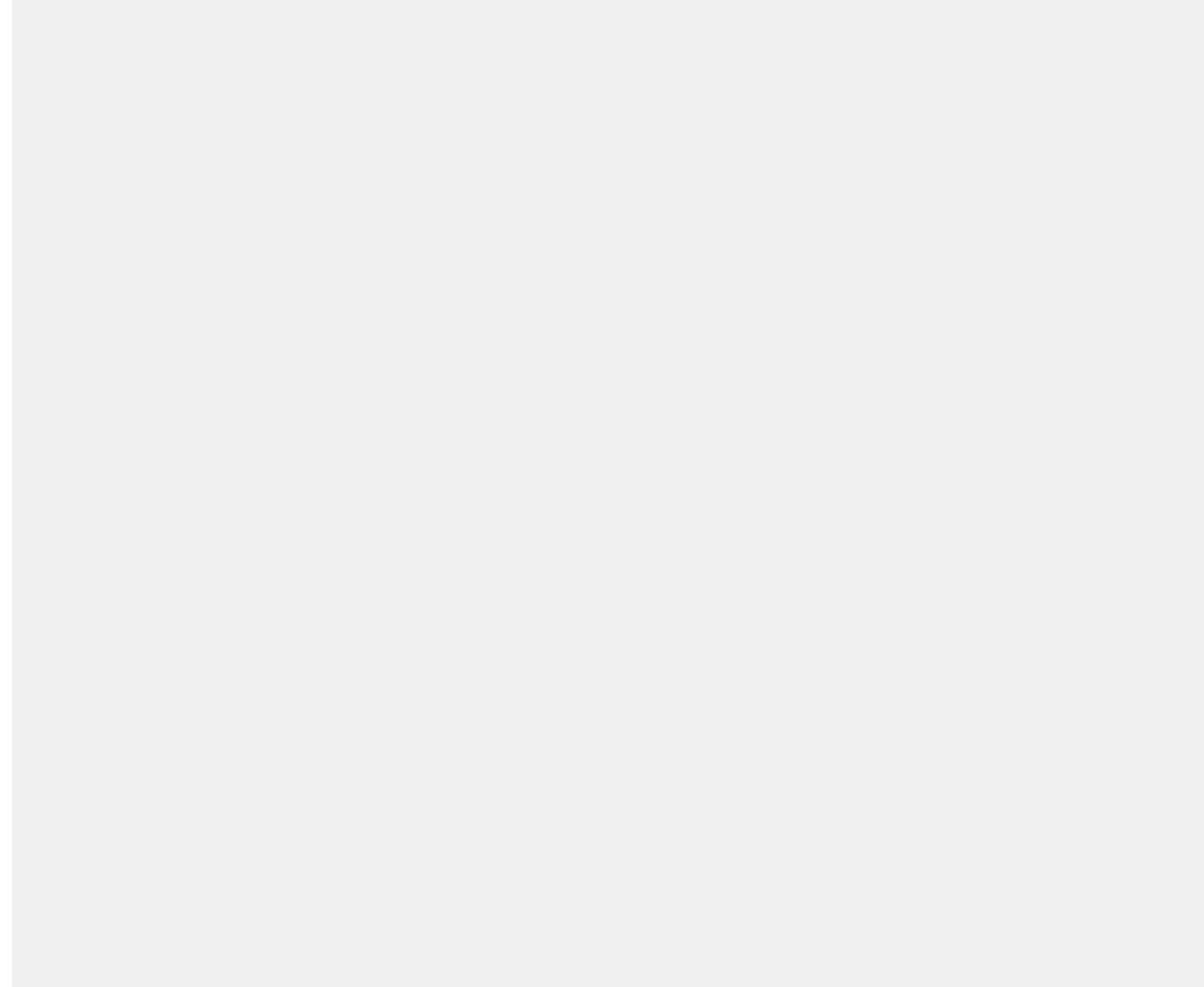
このインターフェースを実装する場合は、パラメータのない空のコンストラクタを用意します。

## 実装例

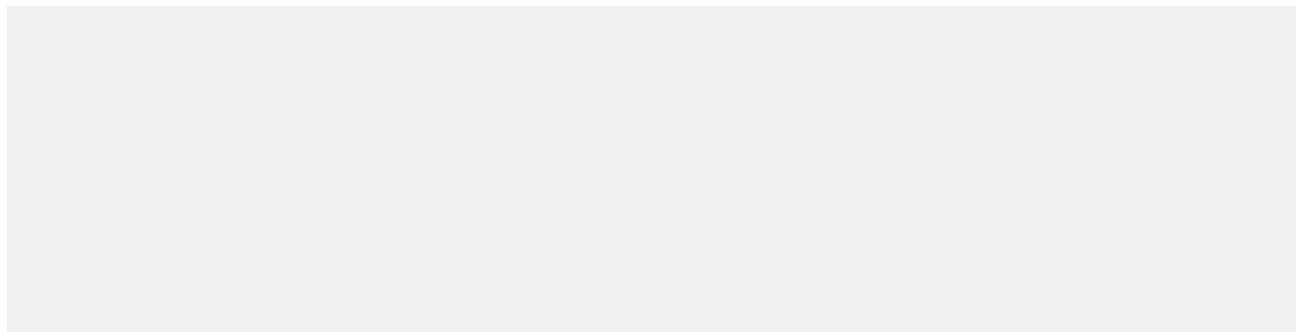
これは、

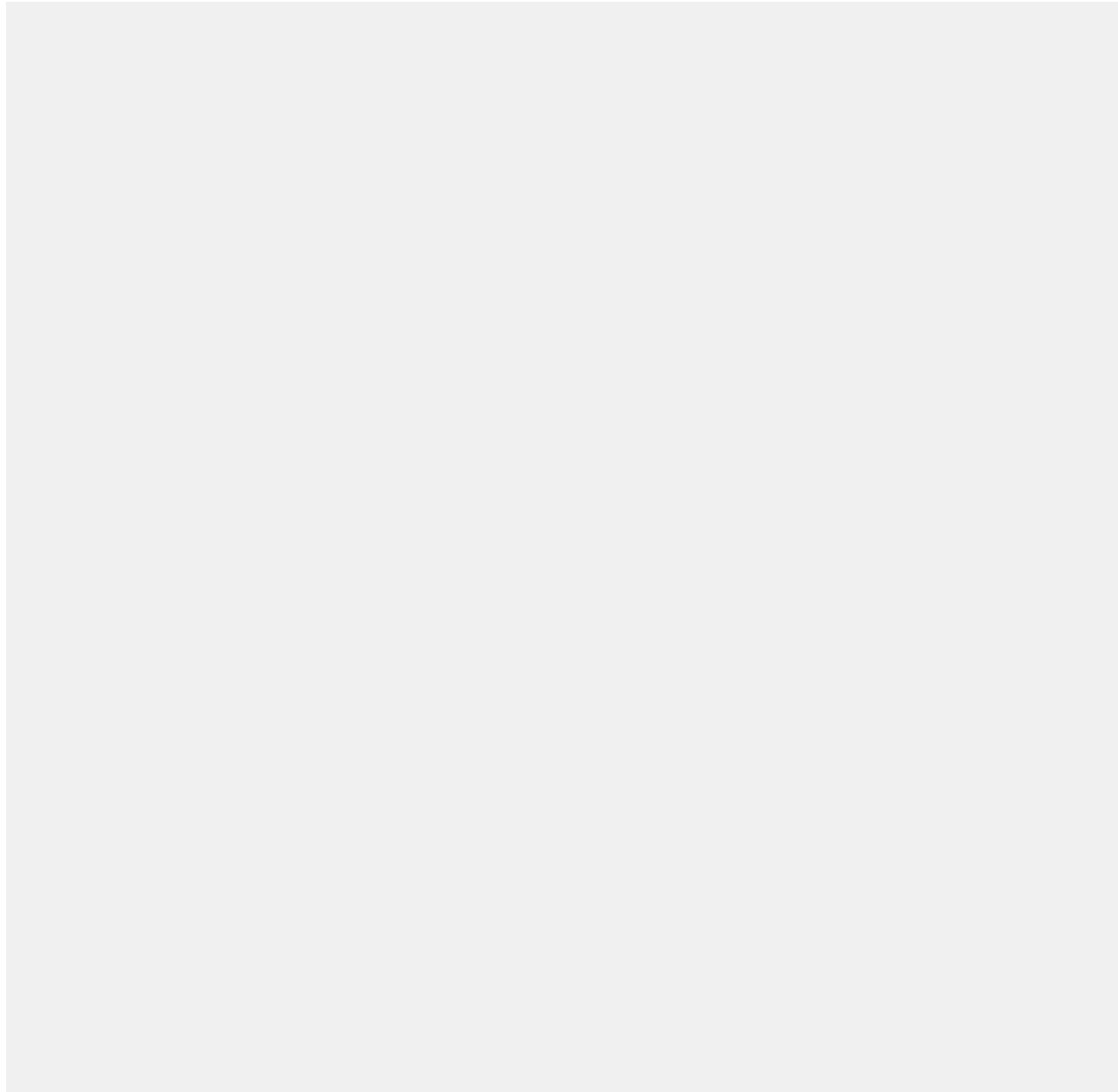
インターフェースの実装例です。

メソッドの実装で、指定したケース ID に対応するケースの件名と説明を取得します。次に、ケースの件名に基づいてメールテンプレートを選択し、メールテンプレート ID を返します。



次の例では、上記のコードをテストします。





## Site.UrlRewriter インターフェース

サイトは、サイト訪問者にわかりやすい URL とリンクを表示する組み込みロジックを備えています。アドレスバーに入力したり、ブックマークから起動したり、または外部 Web サイトからリンクする URL 要求を再記述するルールを作成します。サイトページ内のリンクの URL を再記述するルールも作成できます。URL を再記述すると、URL がわかりやすくなるだけでなく、ユーザが直感的に理解できるようになるため、検索エンジンによるサイトページのインデックス作成がさらに容易になります。

たとえば、自分のブログサイトを持っているとします。URL を書き換えない場合、ブログのエントリの URL は次のようになります。

URL を書き換えると、ユーザはレコード ID ではなく日付やタイトルでブログの投稿にアクセスできます。大晦日の投稿の URL は次のようにになります。

また、サイトページ内に表示されるリンクの URL を書き換えることもできます。大晦日の投稿にバレンタインデーの投稿へのリンクが含まれる場合、リンク URL は次のように表示されます。

サイトの URL を書き換えるには、元の URL をわかりやすい URL に対応付ける Apex クラスを作成して、Apex クラスをサイトに追加します。

次は、Force.com Sites の一部である

インターフェースのインスタンスマソッドです。

名前	引数	戻り値	説明
	System.PageReference[]	System.PageReference[]	<p>Salesforce URL のリストをわかりやすい URL のリストに対応付けます。必要に応じて、ではなく、を使用できます。</p> <p> <b>重要:</b> Salesforce URL の入力リストのサイズと順序は、わかりやすい URL の生成されたリストのサイズと順序に厳密に対応している必要があります。メソッドは、リストの順序に基づいて入力 URL を出力 URL に対応付けます。</p>
	System.PageReference	System.PageReference	わかりやすい URL を Salesforce の URL に対応付けます。

## Apex クラスの作成

作成する Apex クラスでは、Force.com 提供のインターフェース通常、次の形式を使用する必要があります。

を実装する必要があります。

Apex クラスを作成するときは、次の制限と推奨事項に留意してください。

クラスおよびメソッドはグローバルである必要がある

Apex クラスおよびメソッドはすべて である必要があります。

クラスに両方のメソッドを実装する必要がある

Apex クラスには メソッドおよび メソッドの両方を実装する必要があり  
ます。いずれのメソッドも使用しない場合は、そのメソッドが を返すようにします。

Visualforce サイトページでのみ機能する書き換え

受信 URL 要求は、サイトに関連付けられている Visualforce ページのみに対応付けできます。標準ページ、画像、その他のエンティティに対応付けることはできません。

サイトページのリンクの URL を書き換えるには、マージ変数を含む関数を使用します。たとえば、次のコードでは、myPage という名前の Visualforce ページにリンクします。



メモ:  
されません。

を使用する Visualforce

要素は、

## によって影響

*Visualforce 開発者ガイド*の付録「関数」を参照してください。

## 符号化された URL

インターフェースを使用して取得する URL は符号化されています。符号化されていない URL の値にアクセスする必要がある場合は、クラスのメソッドを使用します。

## 文字の制限

わかりやすい URL は Salesforce の URL とは異なる必要があります。3 文字のエンティティのプレフィックスまたは 15 文字または 18 文字の ID を持つ URL は書き換えられません。

書き換えられた後の URL ではピリオドは使用できません。

## 文字列の制限

書き換えられた後の URL パスの一部として、次の予約文字列を使用することはできません。

- • • • • • • • •

- • • • • • • • • • • • • • • • • •

相対パスのみ

ドメインとサイトのプレフィックスは書き換えできません。

一意のパスのみ

サイトのプレフィックスと同じ名前を持つディレクトリには、URL を対応付けできません。たとえば、サイト URL が `http://example.com/help` で、サイトプレフィックスが「help」である場合、への URL をポイントすることはできません。結果として、返されるパスは ではなく、 になります。

## 一括クエリ

ページ作成でのパフォーマンスを向上させるには、  
メソッドでタスクを一度に1つずつではなく、一括で実行します。

## 項目の一意性の適用

URL を書き換えるために選択した項目が一意であることを確認します。クエリに SOQL の一意の項目またはインデックス付き項目を使用すると、パフォーマンスが向上する可能性があります。

また、**メソッド**を使用して、一意の項目名と値でレコードを検索できます。このメソッドでは、指定の項目に一意の ID または外部 ID が含まれていることを確認します。含まれていない場合は、エラーを返します。

次はその一例です。ここで、`Blog` は名前空間、`Blog` はカスタムオブジェクト名、`title` はカスタム項目名、`myBlog` は検索対象の値です。

## サイトへの URL 書き換えの追加

URL を書き換える Apex クラスを作成したら、次のステップに従ってそのクラスをサイトに追加します。

1. [設定] で、[開発] > [サイト] をクリックします。
2. [新規] をクリックします。既存のサイトを変更する場合は [編集] をクリックします。
3. [サイトの編集] ページで、 書き換えクラス の [Apex クラス] を選択します。
4. [保存] をクリックします。



メモ: サイトで URL の書き換えが有効になっている場合、すべての PageReferences はこの URL 書き換えクラスを通過して渡されます。

## コード例

この例では、mycontact と myaccount という 2 つの Visualforce ページで構成される単純なサイトが存在します。このサンプルを試してみる前に、両方のページで「参照」権限が有効になっていることを確認します。各ページでそのオブジェクト種別の標準コントローラが使用されます。取引先責任者ページには親取引先ページへのリンクと取引先責任者の詳細が含まれます。

書き換えを実装する前は、[図 10: 書き換え前のサイト URL](#) に示されるように、アドレスバーとリンク URL はレコード ID (ランダムな 15 衝の文字列) を表示しました。書き換えを有効にした後は、[図 11: 書き換え後のサイト URL](#) に示されるように、アドレスバーとリンクがわかりやすく書き換えられた URL を表示します。

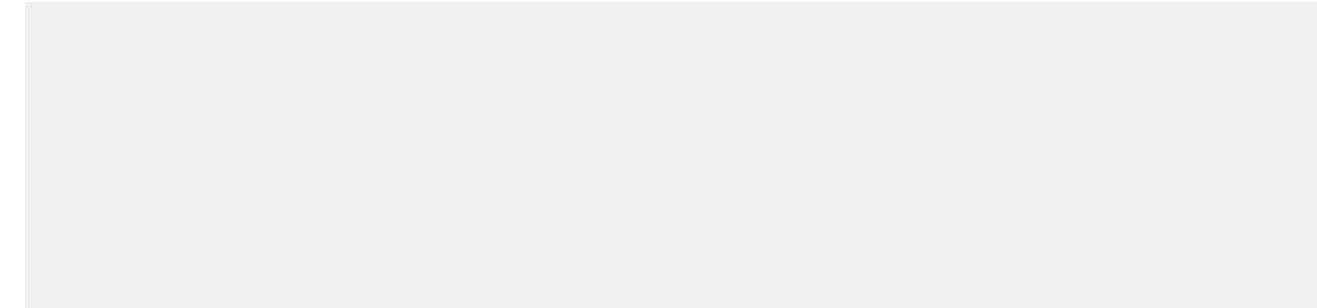
これらのページの URL の書き換えに使用する Apex クラスについては、詳しい説明と共に「[URL を書き換える Apex クラスの例](#)」に示しています。

## サイトページの例

このセクションでは、この例で使用する取引先ページおよび取引先責任者ページの Visualforce を示します。

取引先ページは取引先の標準コントローラを使用する、標準的な詳細ページです。このページは myaccount という名前になります。

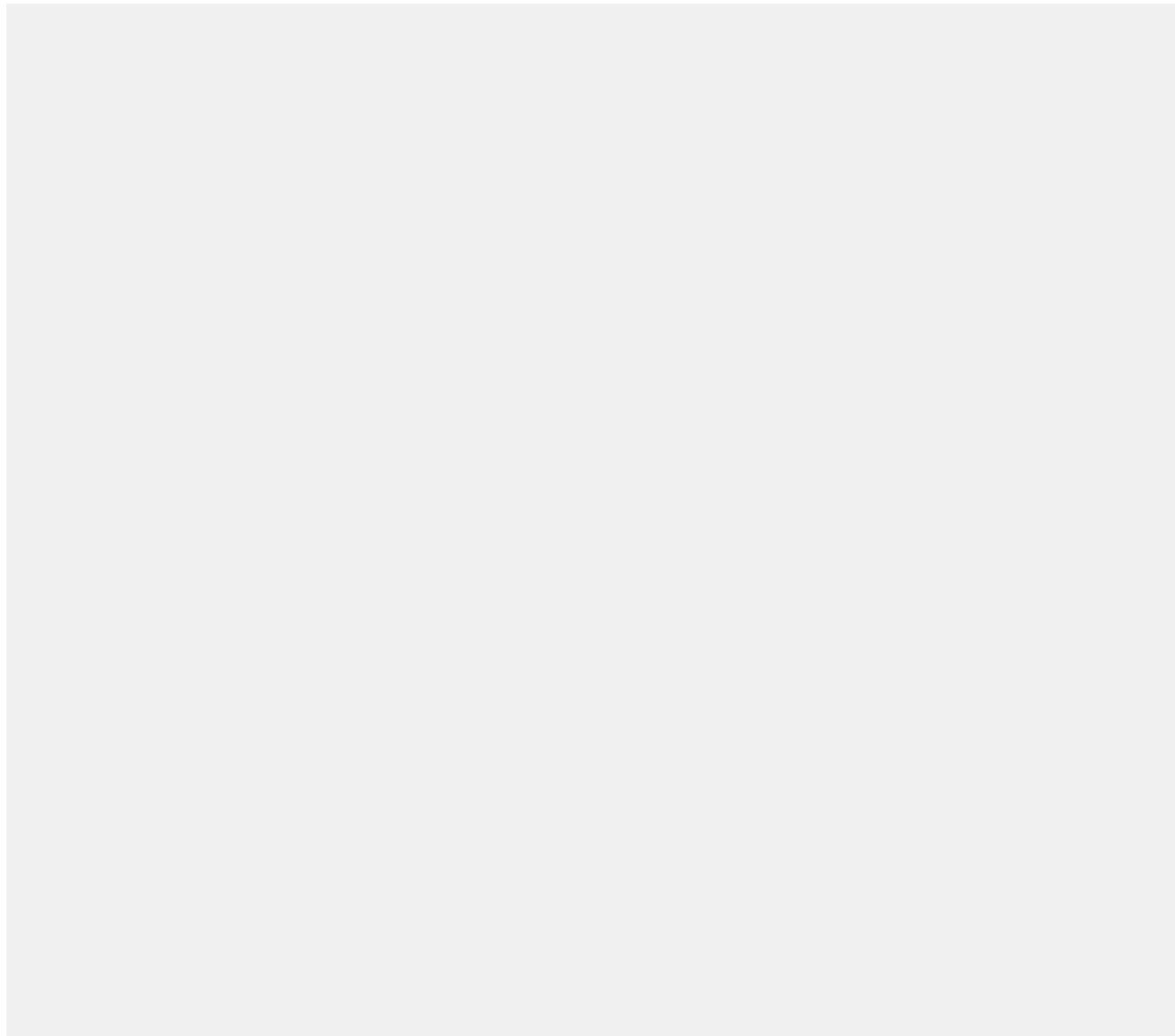
取引先責任者ページで取引先責任者の標準コントローラを使用し、2 つの部分で構成されます。最初の部分は関数と  マージ変数を使用して親取引先にリンクし、後の部分は単純に取引先の詳細を示します。Visualforce ページでは  以外に書き換えロジックを備えていません。このページは mycontact という名前になります。

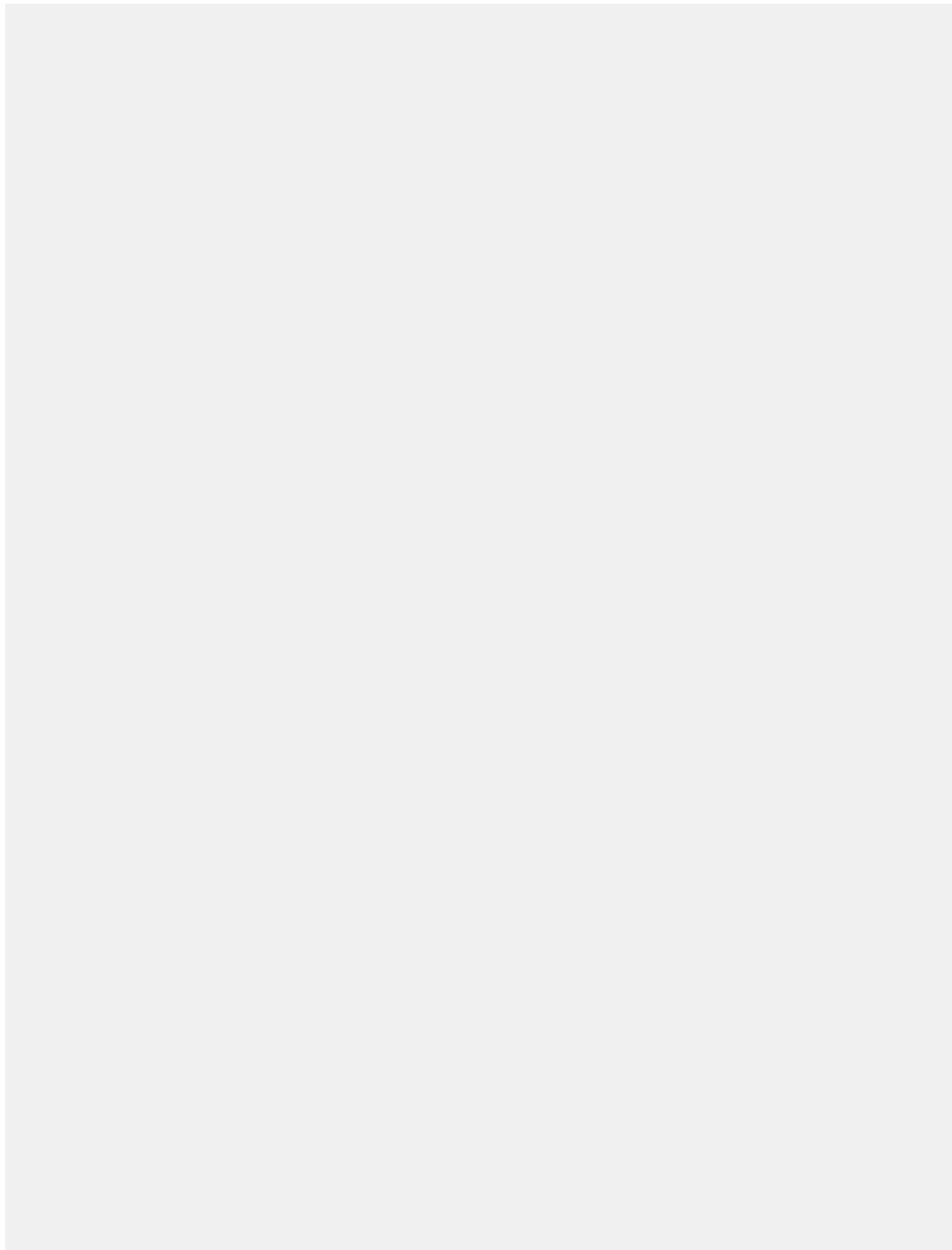


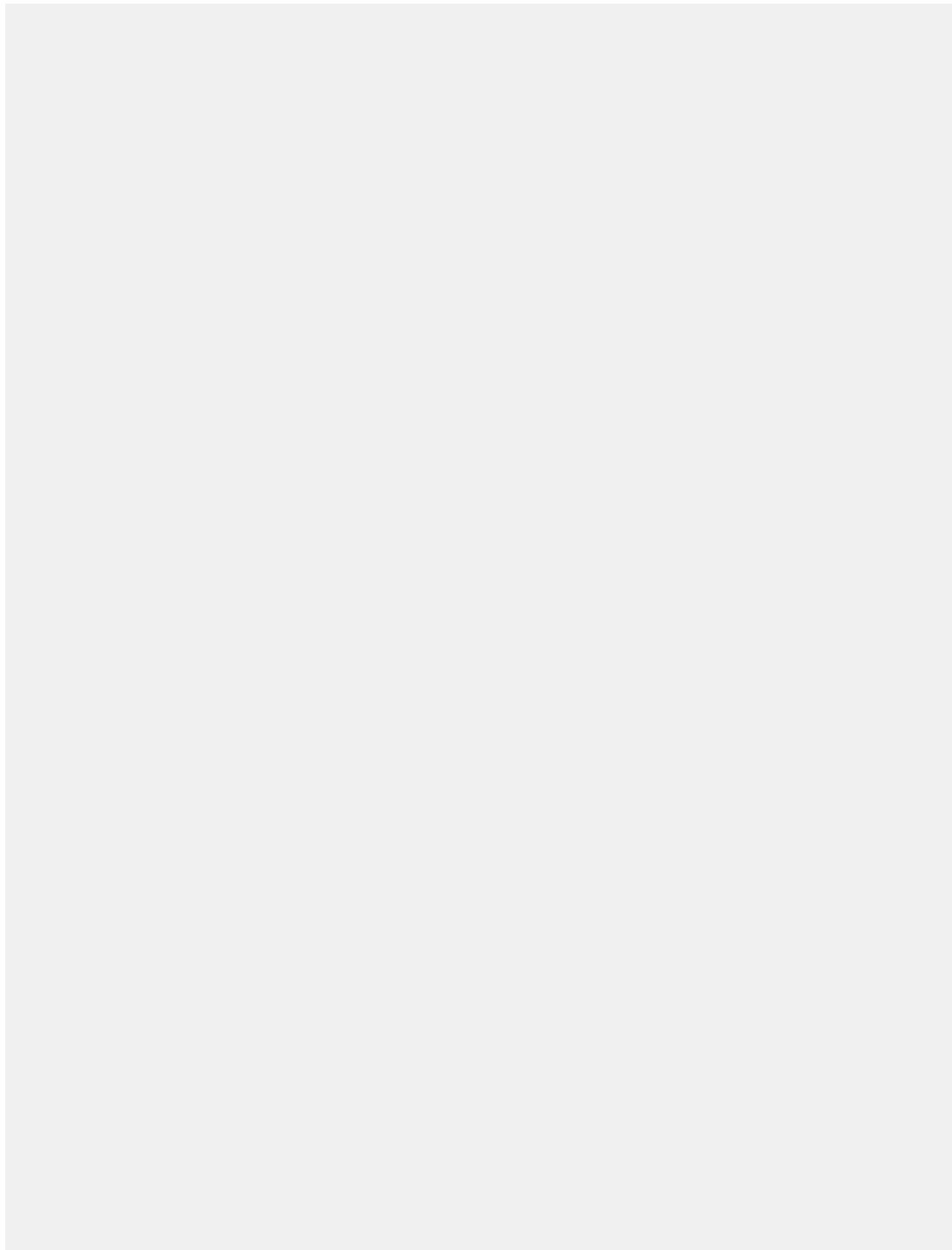
## URL を書き換える Apex クラスの例

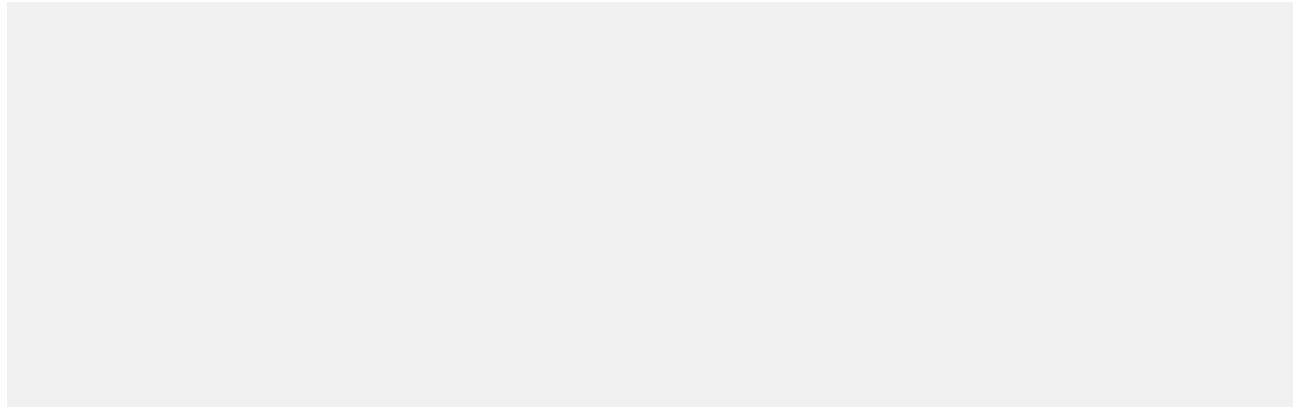
サイトで URL の書き換えに使用される Apex クラスでは、適切な Salesforce レコードに対応付けます。また、取引先ページへのリンクの URL を書き換えます。

メソッドを使用して、受信 URL 要求  
メソッドを使用して、わかりやすい形式









## 書き換え前と書き換え後

ここでは、元のサイト URL を書き換える Apex クラスを実装した結果の表示例を示します。最初の図では ID ベースの URL、2 番目の図ではわかりやすい URL が表示されています。

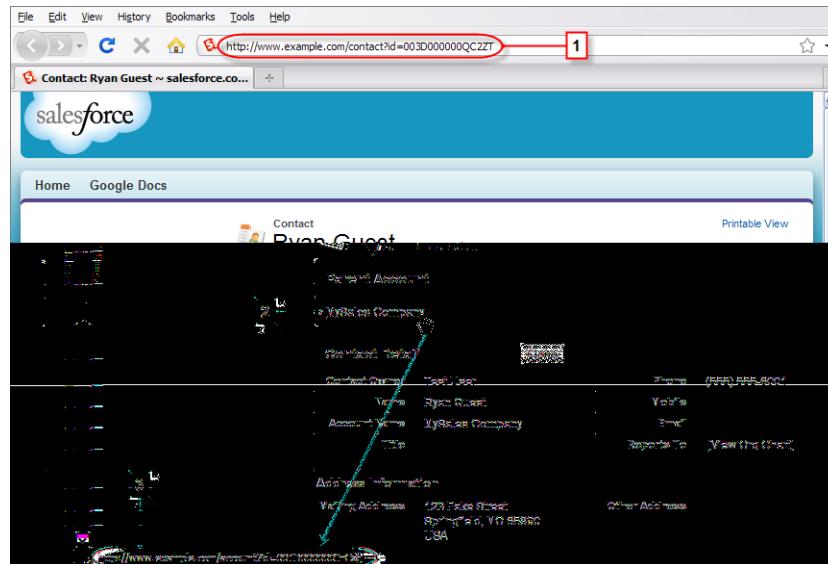


図 10: 書き換え前のサイト URL

図中の番号を付した要素の内容は、次のとおりです。

1. 取引先責任者ページの書き換え前の元の URL
2. 取引先責任者ページから親取引先ページへのリンク
3. 取引先ページへのリンクの書き換え前の元の URL (ブラウザのステータスバーに表示される)

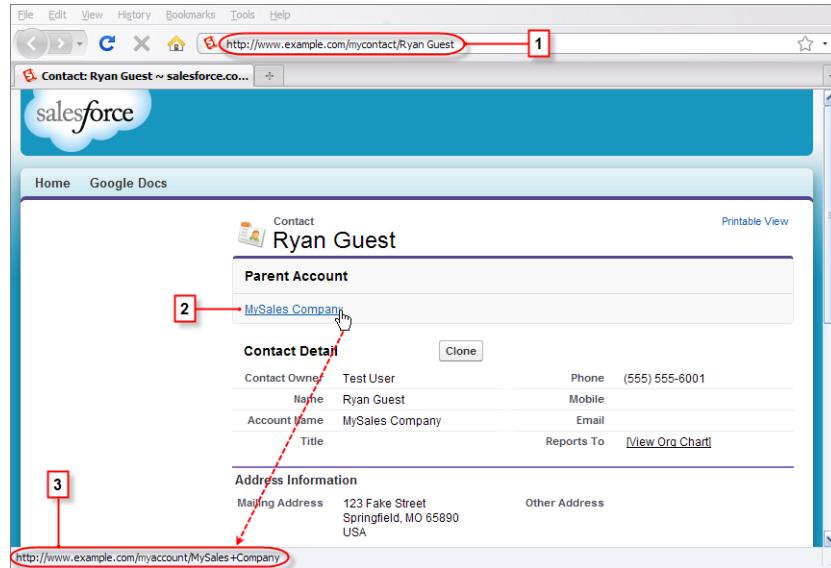


図 11: 書き換えた後のサイト URL

図中の番号を付した要素の内容は、次のとおりです。

1. 取引先責任者ページの書き換えられた URL
2. 取引先責任者ページから親取引先ページへのリンク
3. 取引先ページへのリンクの書き換え後の URL (ブラウザのステータスバーに表示される)

## Process.Plugin インターフェースの使用

は組み込みインターフェースで、組織内のデータを処理し、指定のフローにデータを渡すことができます。インターフェースは Apex をサービスとして公開し、サービスは入力値を受け付け、出力をフローに戻します。

組織にインターフェースを実装する Apex クラスを定義すると、Cloud Flow Designer のパレットにその Apex クラスが表示されます。

には、次の最上位クラスがあります。

- 
- 
- 

クラスは、インターフェースを実装するクラスからフローに入力パラメータを渡します。

クラスは、インターフェースを実装するクラスからフローに出力パラメータを返します。

クラスは、フローからインターフェースを実装するクラスに入力パラメータを渡します。

Apex 単体テストを記述する場合は、クラスをインスタンス化してインターフェースのメソッドに渡す必要があります。また、対応付けを作成してコンストラクタに使用し、システムが必要とするパラメータに渡す必要があります。

詳細は、「[クラス](#)」を参照してください。

クラスは、プラグインで必要な入力パラメータと出力パラメータを判断するために使用されます。

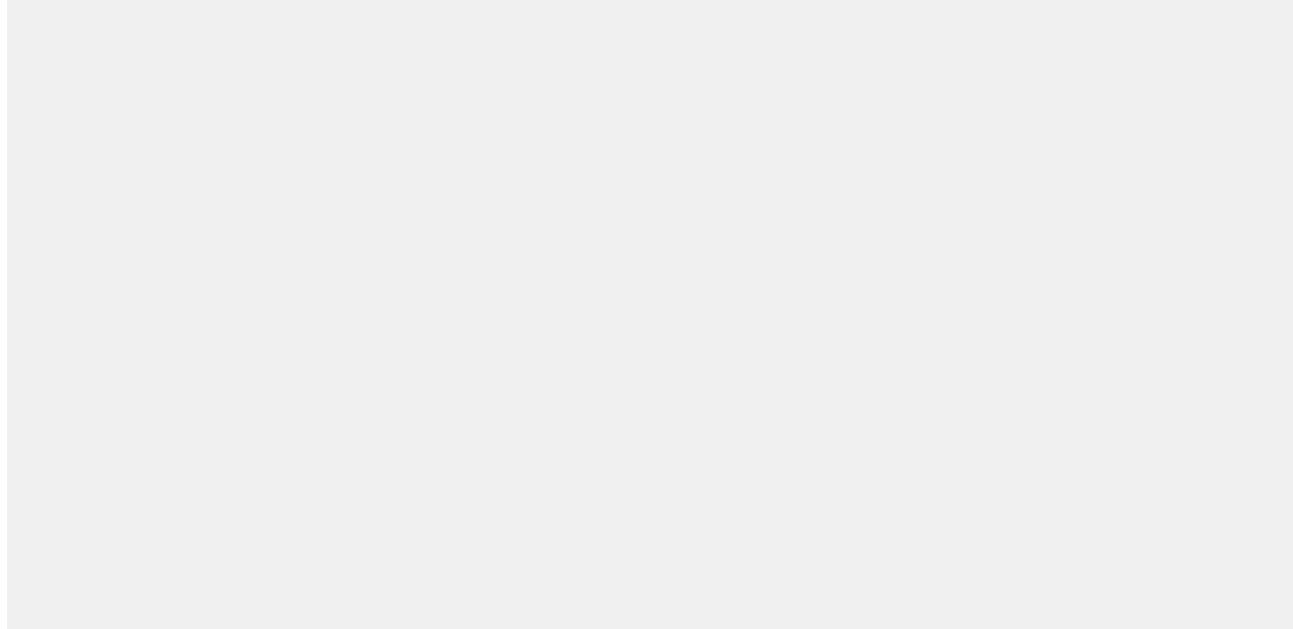
## Process.Plugin インターフェース

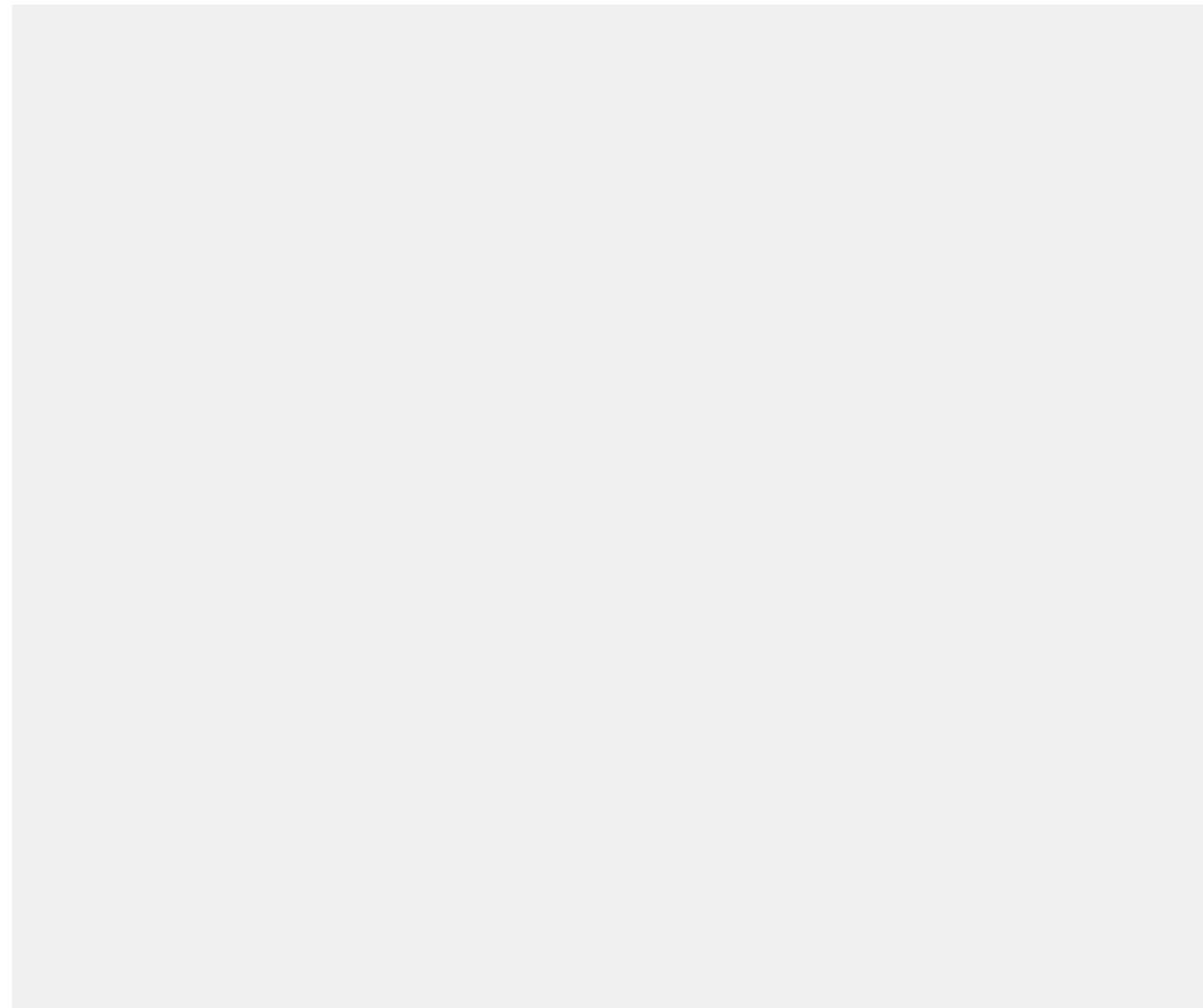
は、組織と指定したフローの間でデータを渡すための組み込みインターフェースです。

インターフェースを実装するクラスを使用してコールする必要のあるメソッドを次に示します。

名前	引数	戻り値	説明
			このメソッドのコールを記述する
			オブジェクトを返します。
			インターフェースを実装するクラスがインスタンス化されるときにシステムが呼び出す主なメソッドです。

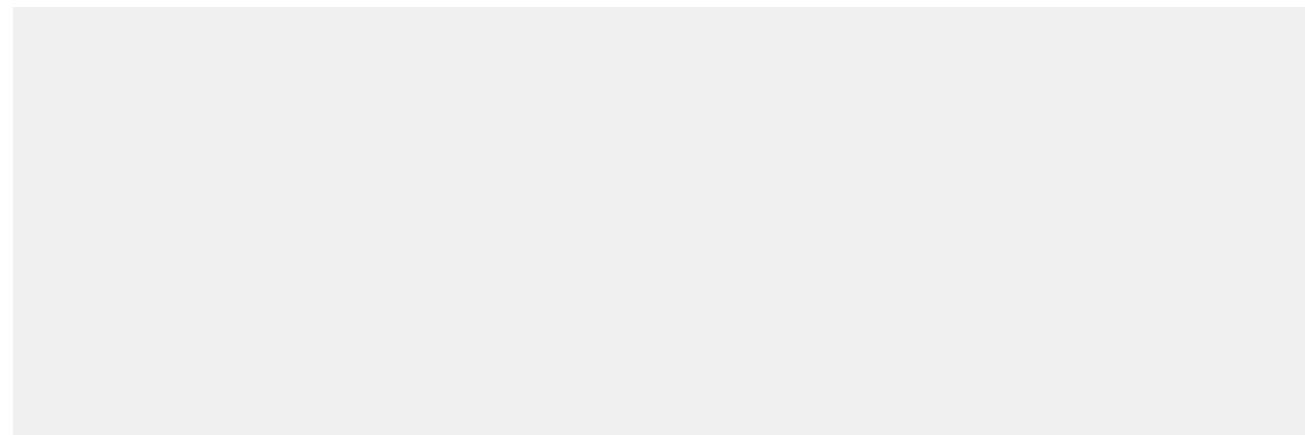
## 実装例

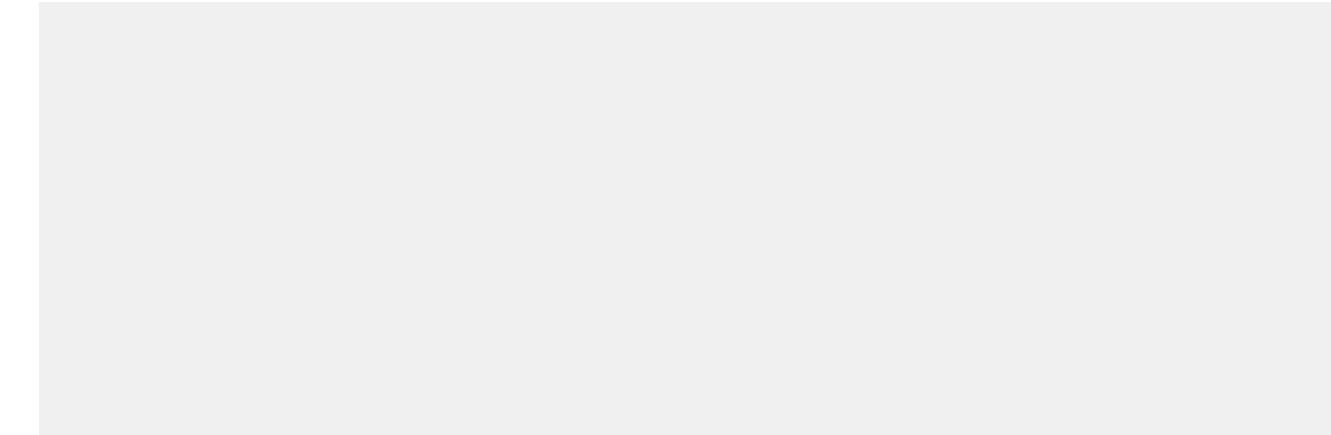




## テストクラス

上記のクラスに使用するテストクラスは次のとおりです。





### **Process.PluginRequest クラス**

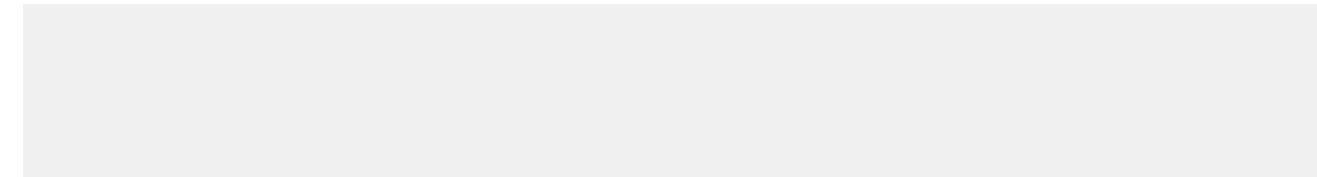
クラスは、インターフェースを実装するクラスからフローに入力パラメータを渡します。

このクラスにはメソッドはありません。

コンストラクタの署名:

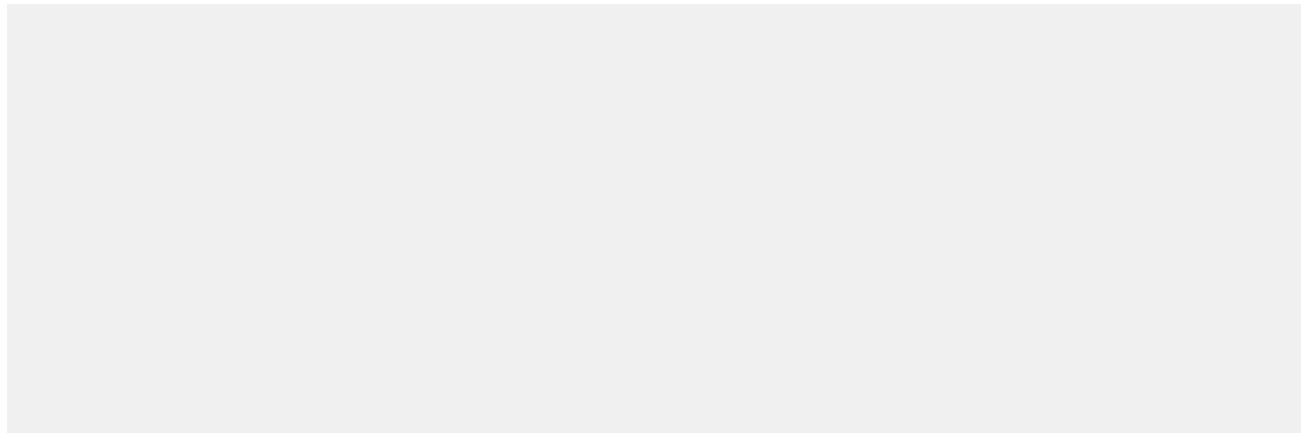
次に、

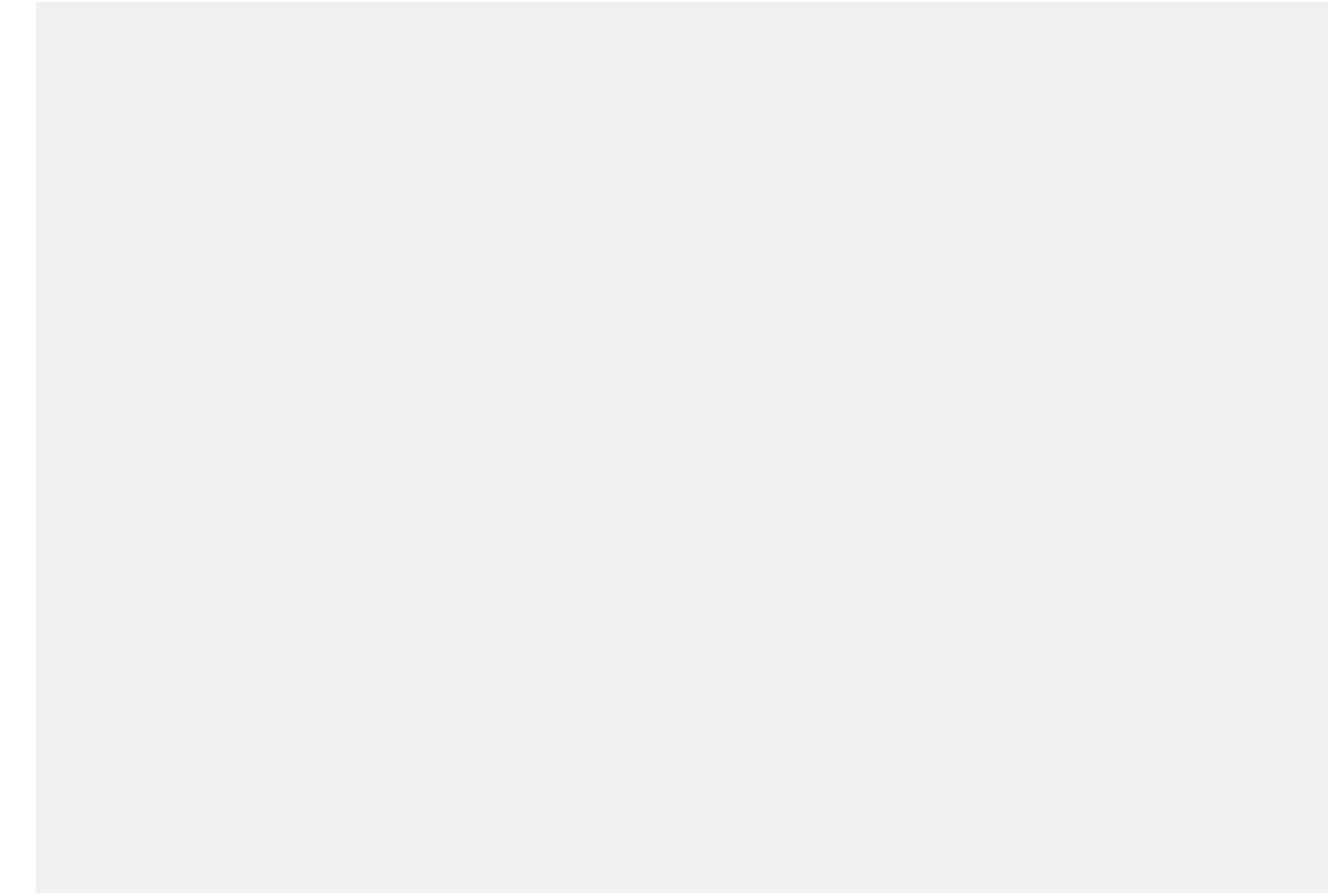
クラスを 1 つの入力パラメータでインスタンス化する例を示します。



#### **コード例**

この例では、コードはフローからの Chatter 投稿の件名を返し、現在のユーザのフィードに投稿します。





## **Process.PluginResult クラス**

クラスは、インターフェースを実装するクラスからフローに出力パラメータを返します。

次のいずれかの形式を使用して、

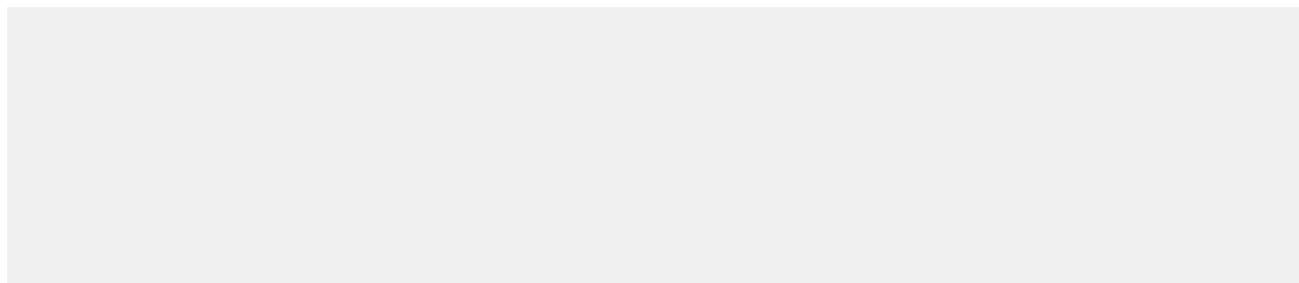
クラスをインスタンス化できます。

- 
- 

複数の結果が返される場合、または返される結果の件数が不明な場合は、対応付けを使用します。

次に、

クラスのインスタンス化の例を示します。



**Process.PluginDescribeResult クラス**

クラスは、  
ラメータを判断するために使用されます。

フローの入力パラメータと出力パラメータの両方を動的に検出するには、  
メソッドを使用します。このメソッドは、

クラスで必要な入力パラメータと出力パ

インターフェースの  
クラスを返します。

クラスは、次の機能の実行には使用できません。

- クエリ
- データの変更
- メール
- Apex のネストされたコールアウト

**Process.PluginDescribeResult クラスおよびサブクラスのプロパティ**

表3: Process.PluginDescribeResult プロパティ

名前	型	説明	サイズ制限
	String	この省略可能な項目では、プラグインの 255 文字目的を説明します。	
	List< >	入力パラメータは、 クラスによつ て、フローから インターフェースを実装するクラスに渡され ます。	イン
	String	プラグインの一意の名前。	40 文字
	List< >	出力パラメータは、 クラスによつ て、インターフェース を実装するクラスからフローに渡されま す。	イン
	String	この省略可能な項目では、プラグインが 40 文字 Flow Designer 内のパレットの Apex プラ グインセクションに一緒に表示されるよ うにタグ名でプラグインをグループ化で きます。これは、フローに複数のプラグ インがある場合に役立ちます。	

クラスのコンストラクタは次のとおりです。

*classname*

表4: `Process.PluginDescribeResult.InputParameter` プロパティ

名前	型	説明	サイズ制限
	String	この省略可能な項目では、プラグインの目的を説明します。	255 文字
	String	プラグインの一意の名前。	40 文字
		入力パラメータのデータ型。	
	Boolean	必須である場合は、 <code>True</code> に、他の場合は、 <code>False</code> に設定します。	

クラスのコンストラクタは次のとおりです。

```
ip
  Name Optional_description_string
  Enum Boolean_required
```

表5: `Process.PluginDescribeResult.OutputParameter` プロパティ

名前	型	説明	サイズ制限
	String	この省略可能な項目では、プラグインの目的を説明します。	255 文字
	String	プラグインの一意の名前。	40 文字
		入力パラメータのデータ型。	

クラスのコンストラクタは次のとおりです。

```
op
  Name Optional_description_string
  Enum
```

クラスを使用するために、次の追加サブクラスのインスタンスを作成します。

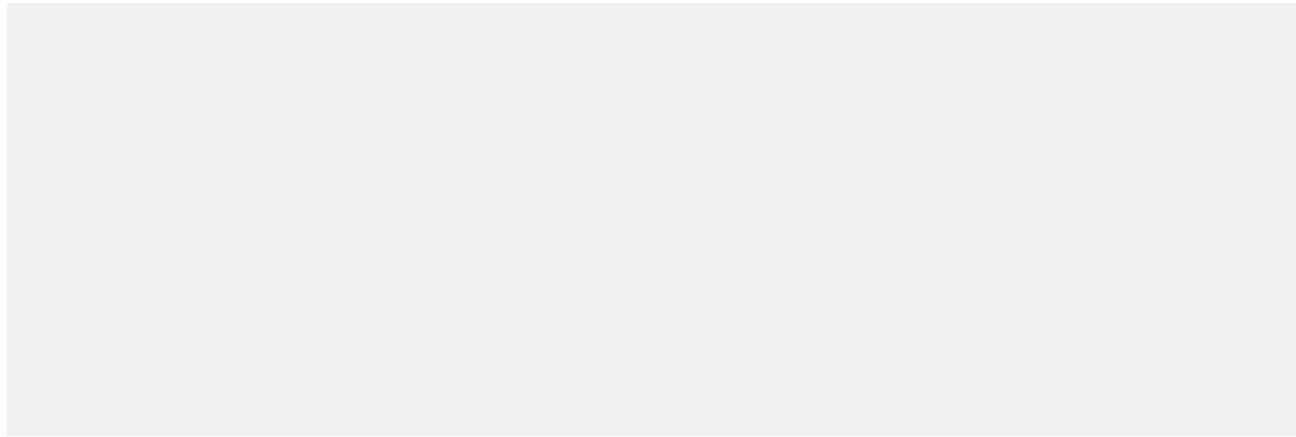
- 
- 

は、入力パラメータのリストで、次の形式になります。


*Name* *Optional\_description\_string*

*Enum* *Boolean\_required*

次に例を示します。

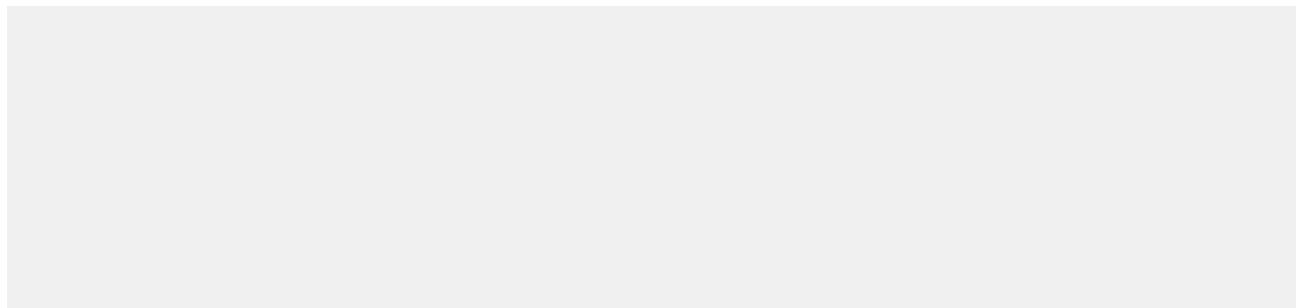


は、出力パラメータのリストで、次の形式になります。

*Name* *Optional\_description\_string*

*Enum*

次に例を示します。



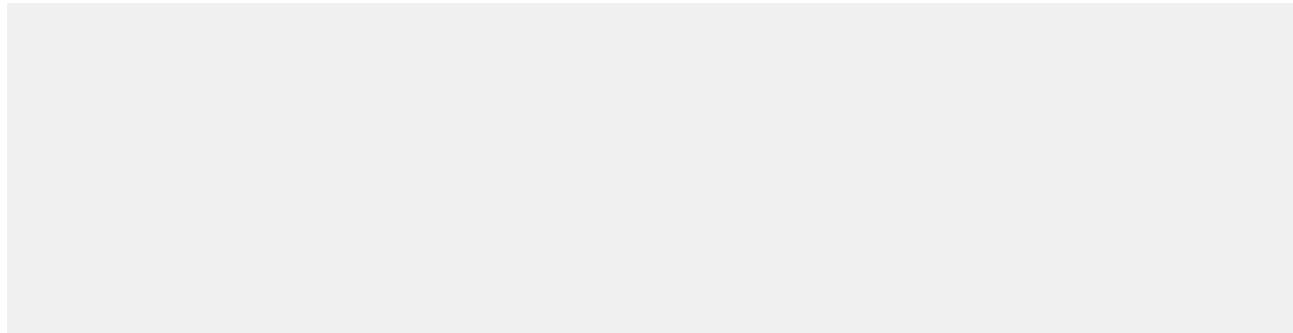
両方のクラスは

列挙型で、次の値を持ちます。

- BOOLEAN
- DATE
- DATETIME
- DECIMAL
- DOUBLE
- FLOAT
- ID

- INTEGER
- LONG
- STRING

次に例を示します。



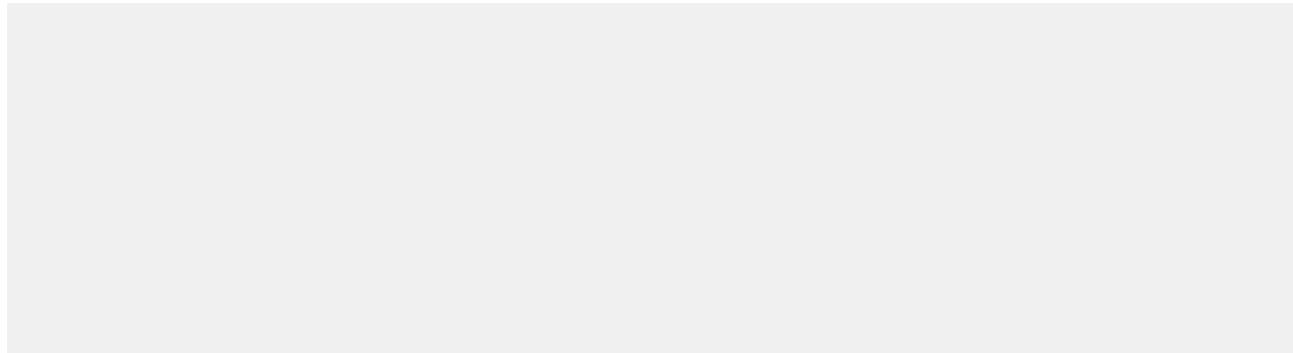
### Process.Plugin データ型変換

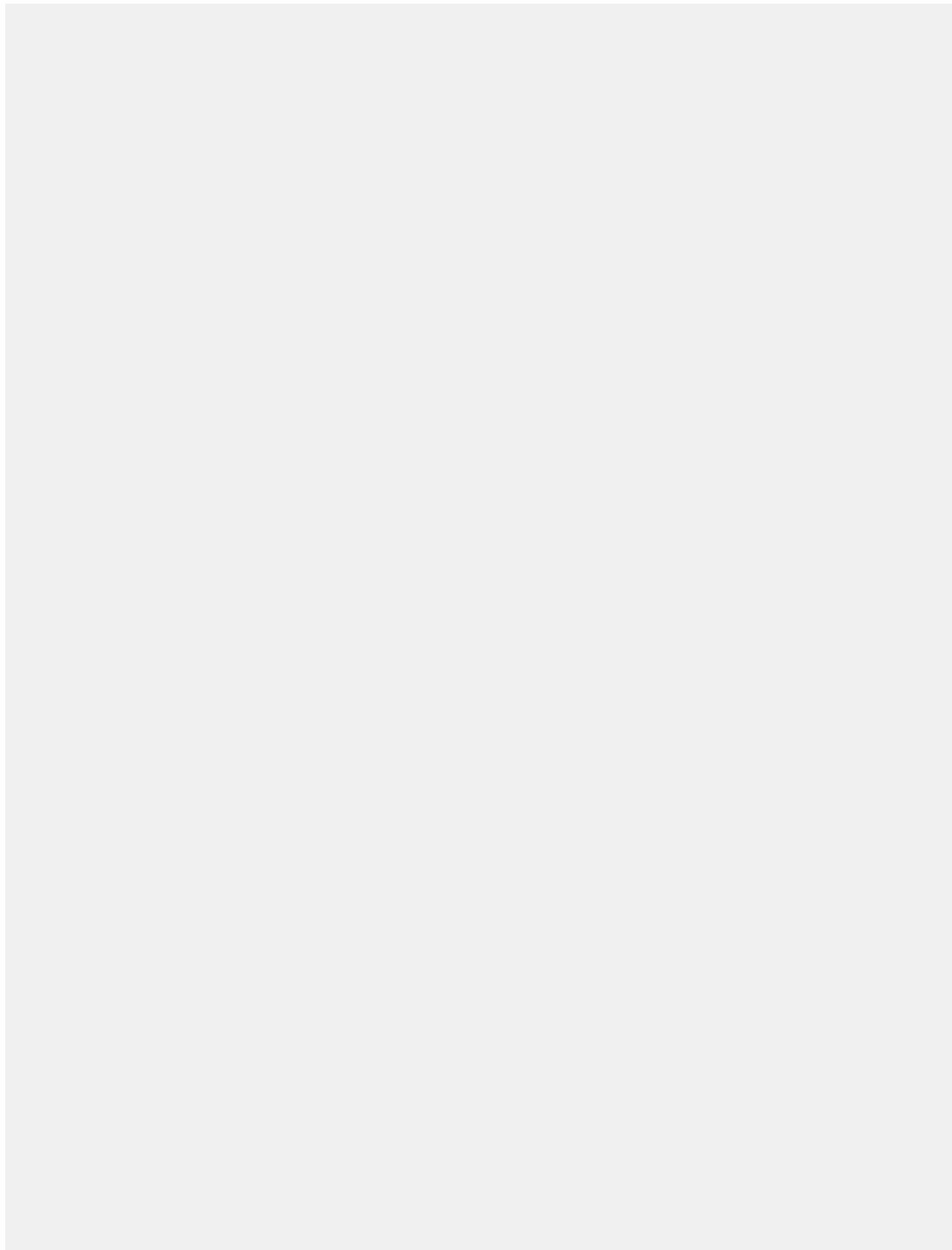
次の表に、Apex と **Process.Plugin** に返された値の間のデータ型変換を示します。たとえば、フローのテキストデータを Apex の文字列データに変換します。

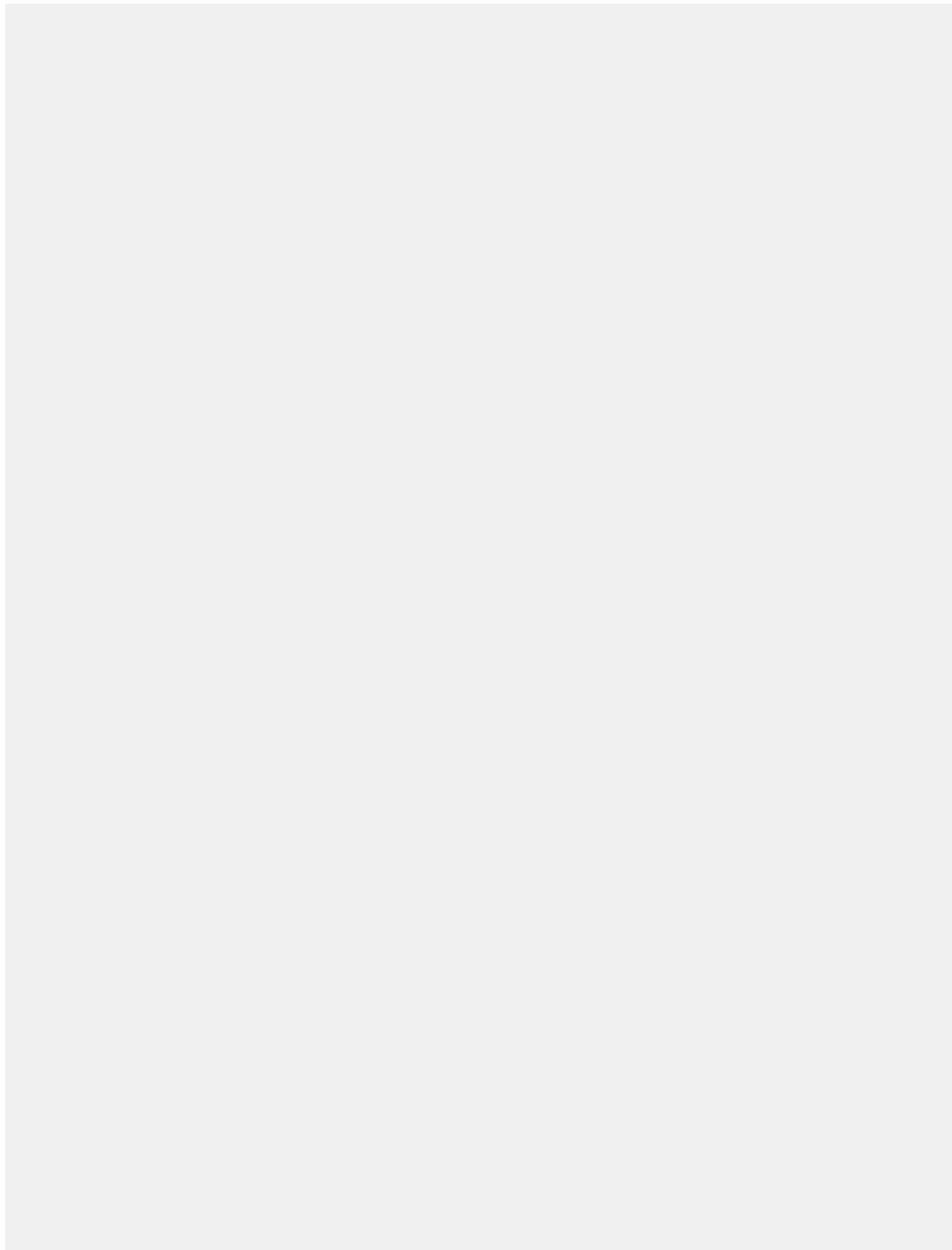
フローデータ型	データ型
Number	Decimal
Date	datetime/date
Boolean	Boolean および numeric (値が 1 または 0 のみ)
text	String

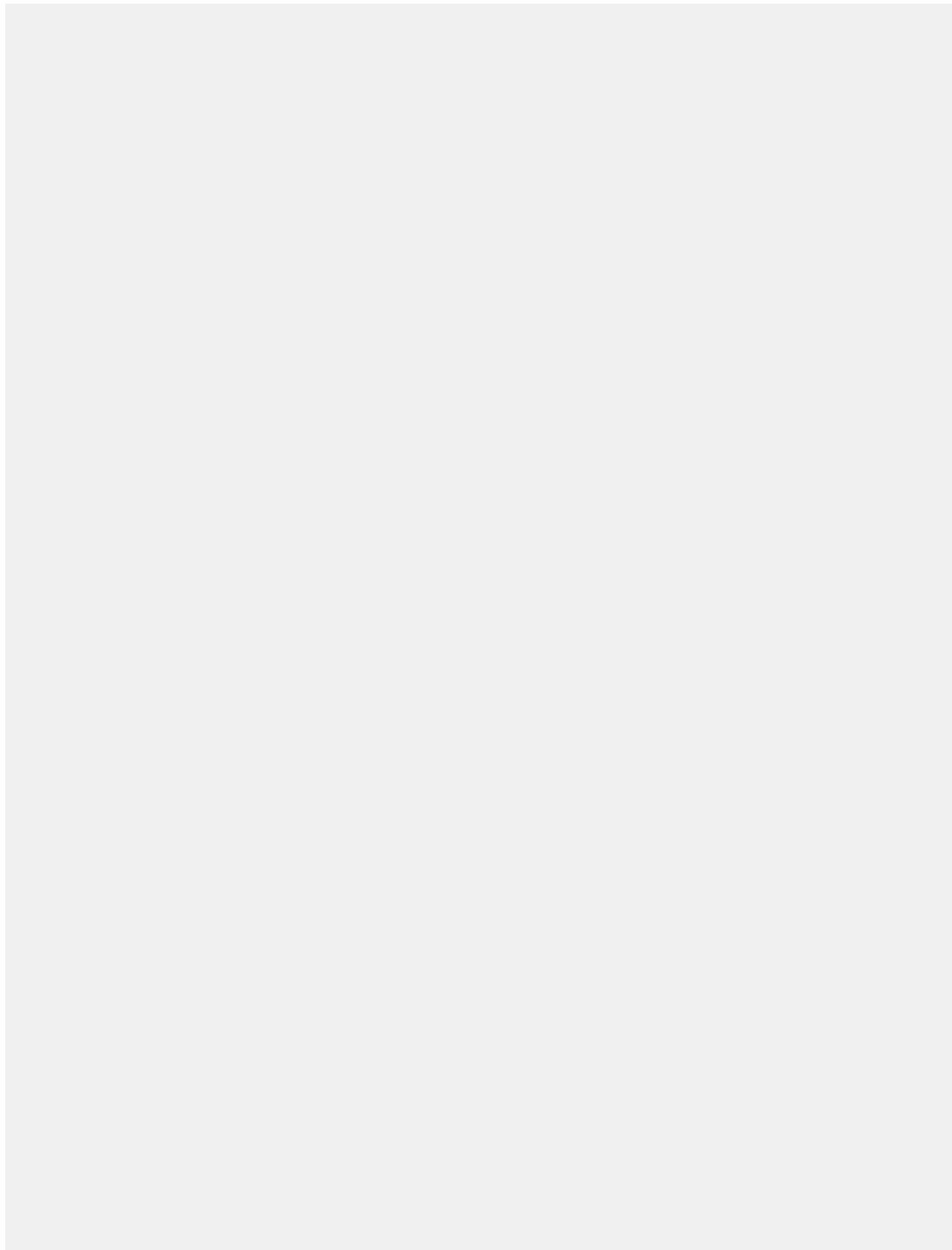
### リードの変換用の Process.Plugin の実装のサンプル

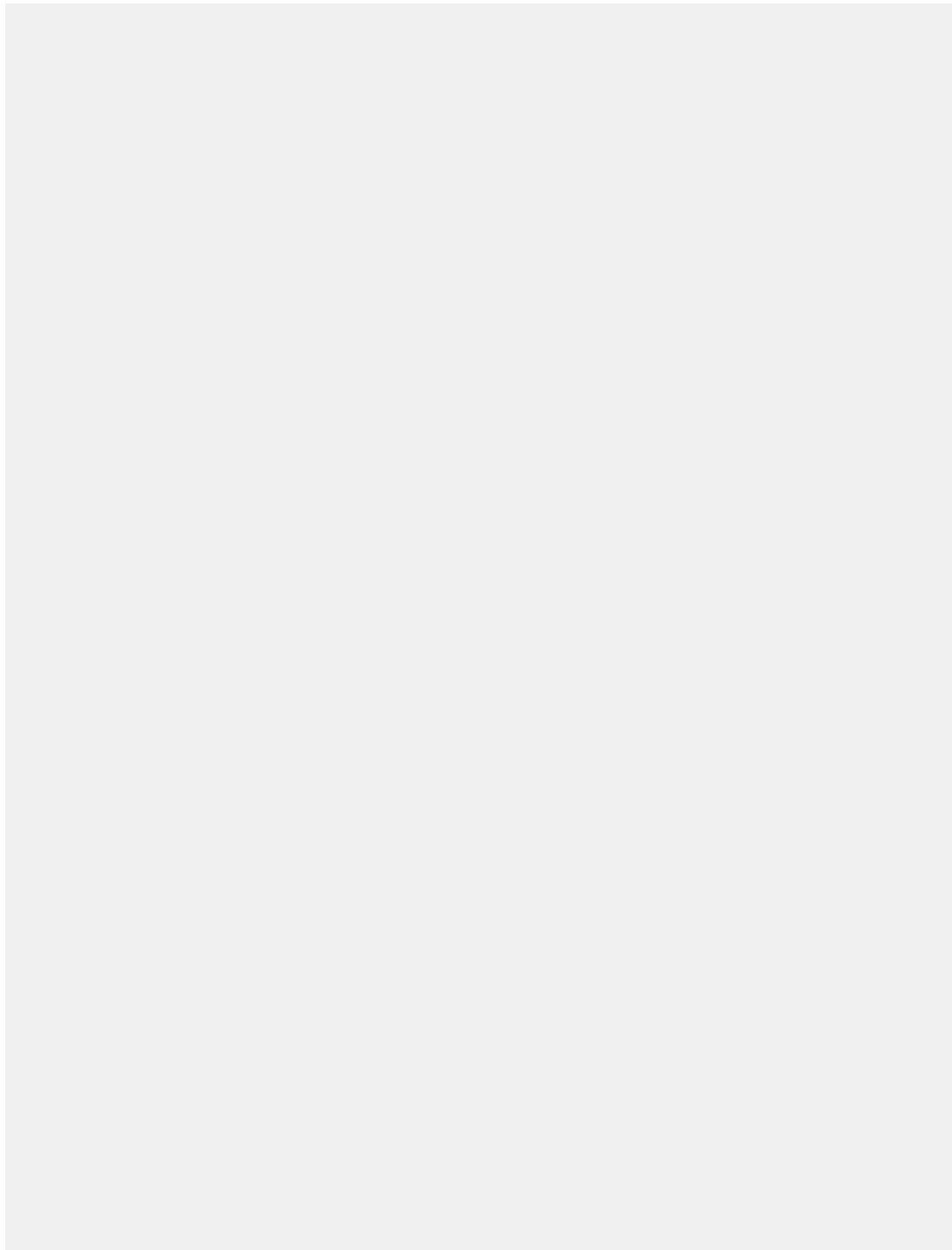
この例では、Apex クラスで Process.Plugin インターフェースを実装し、リードを取り引先、取引先責任者、および必要に応じて商談に変換します。プラグインのテストメソッドも含まれています。この実装は、Apex プラグイン要素を使用してフローからコールできます。

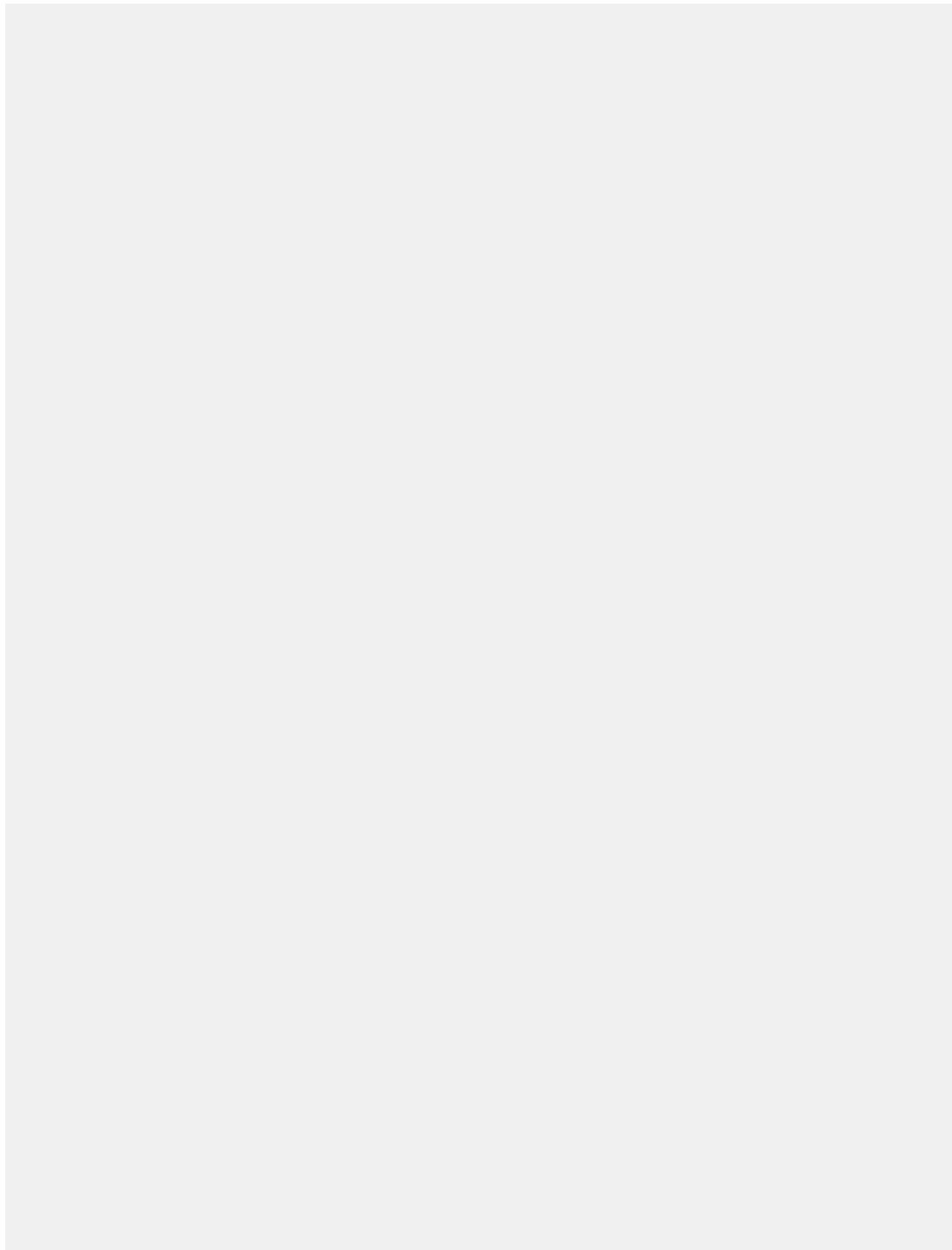


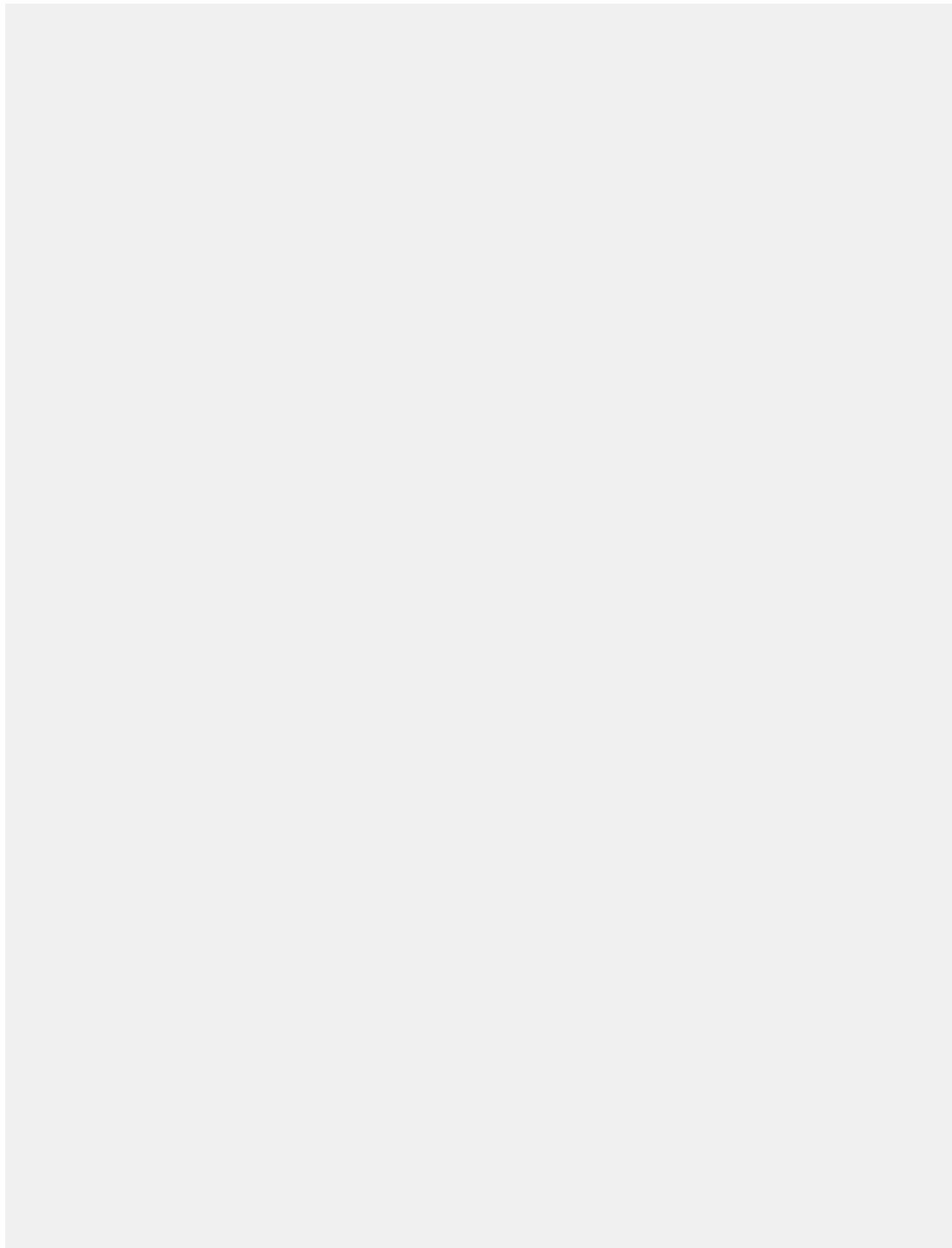


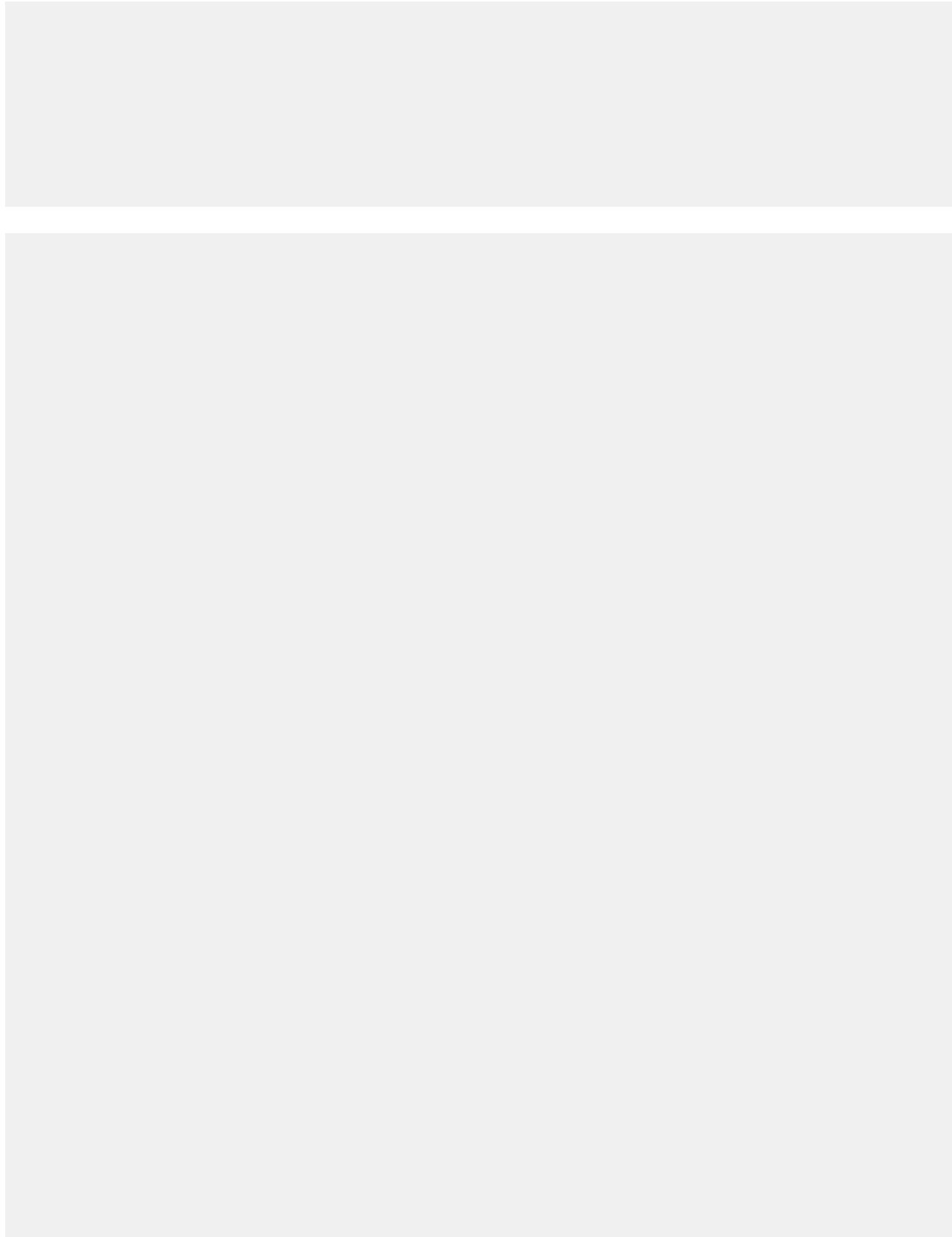


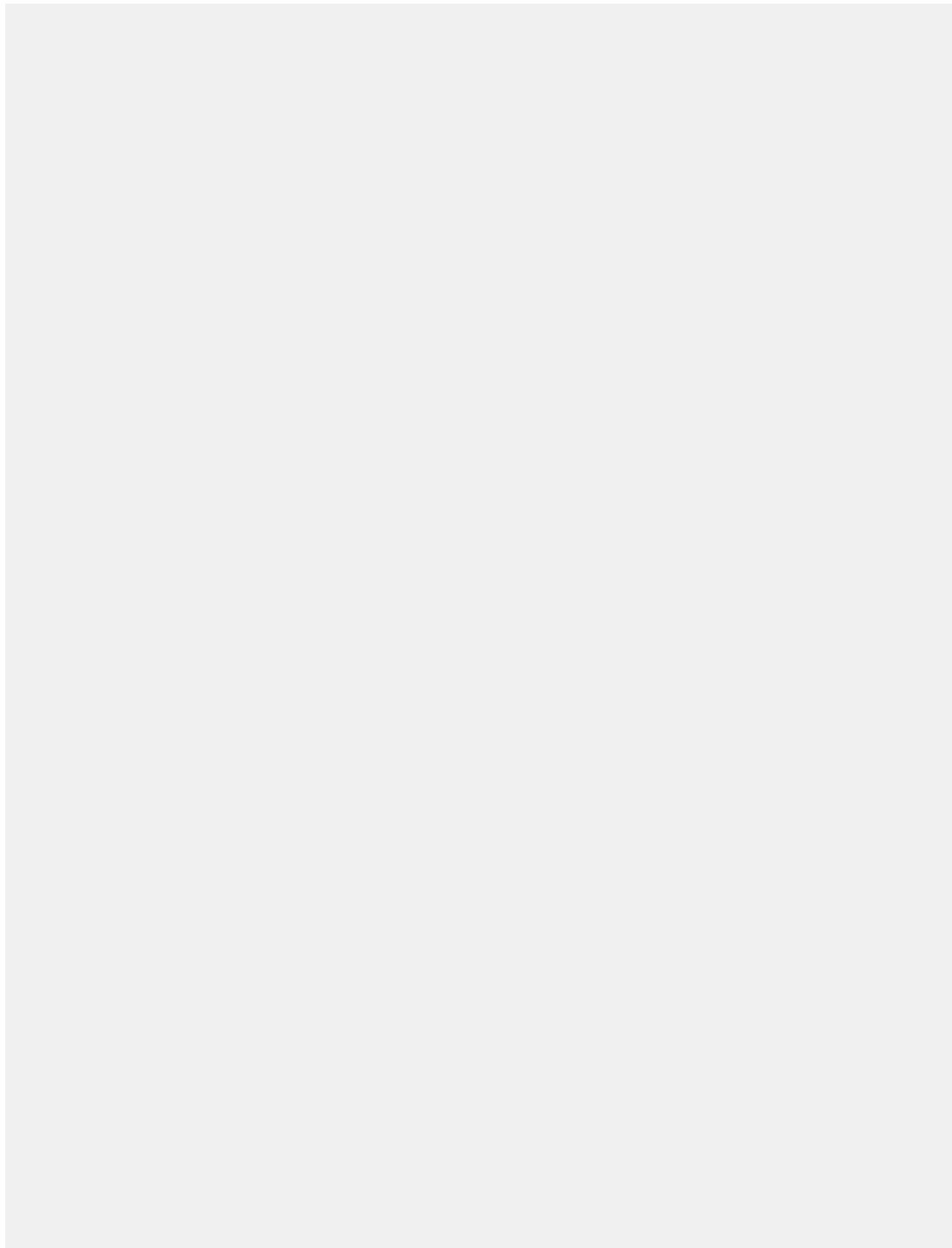


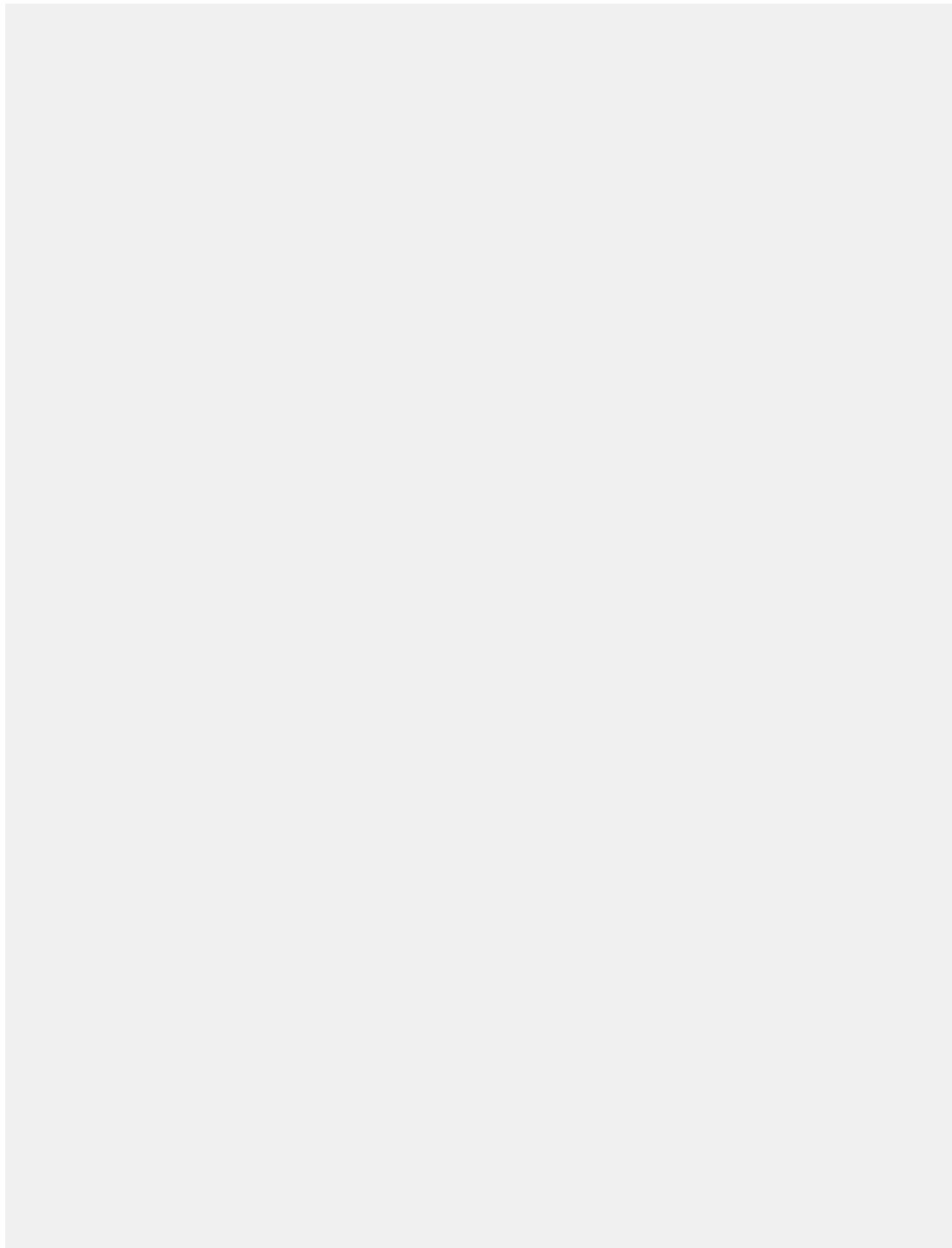


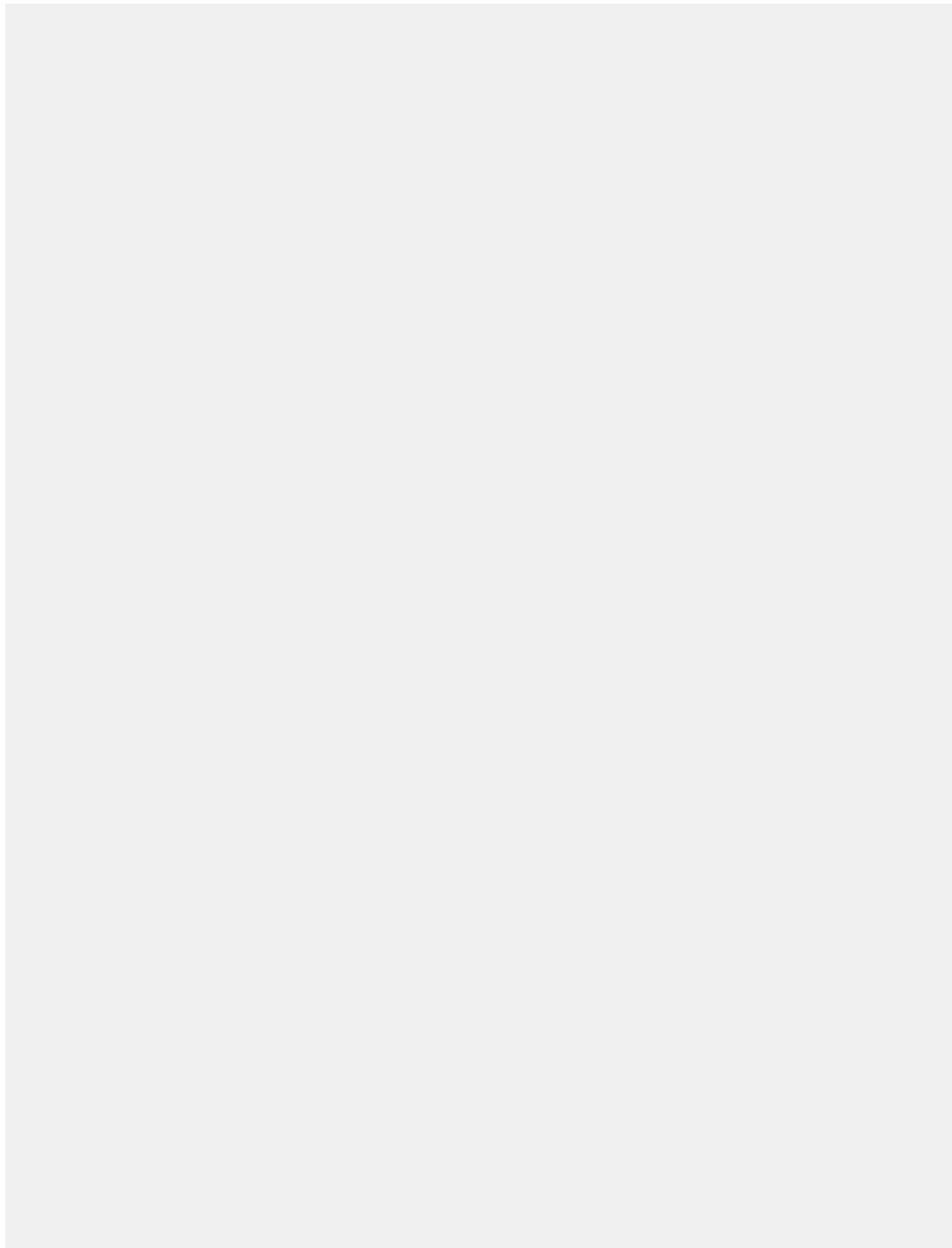


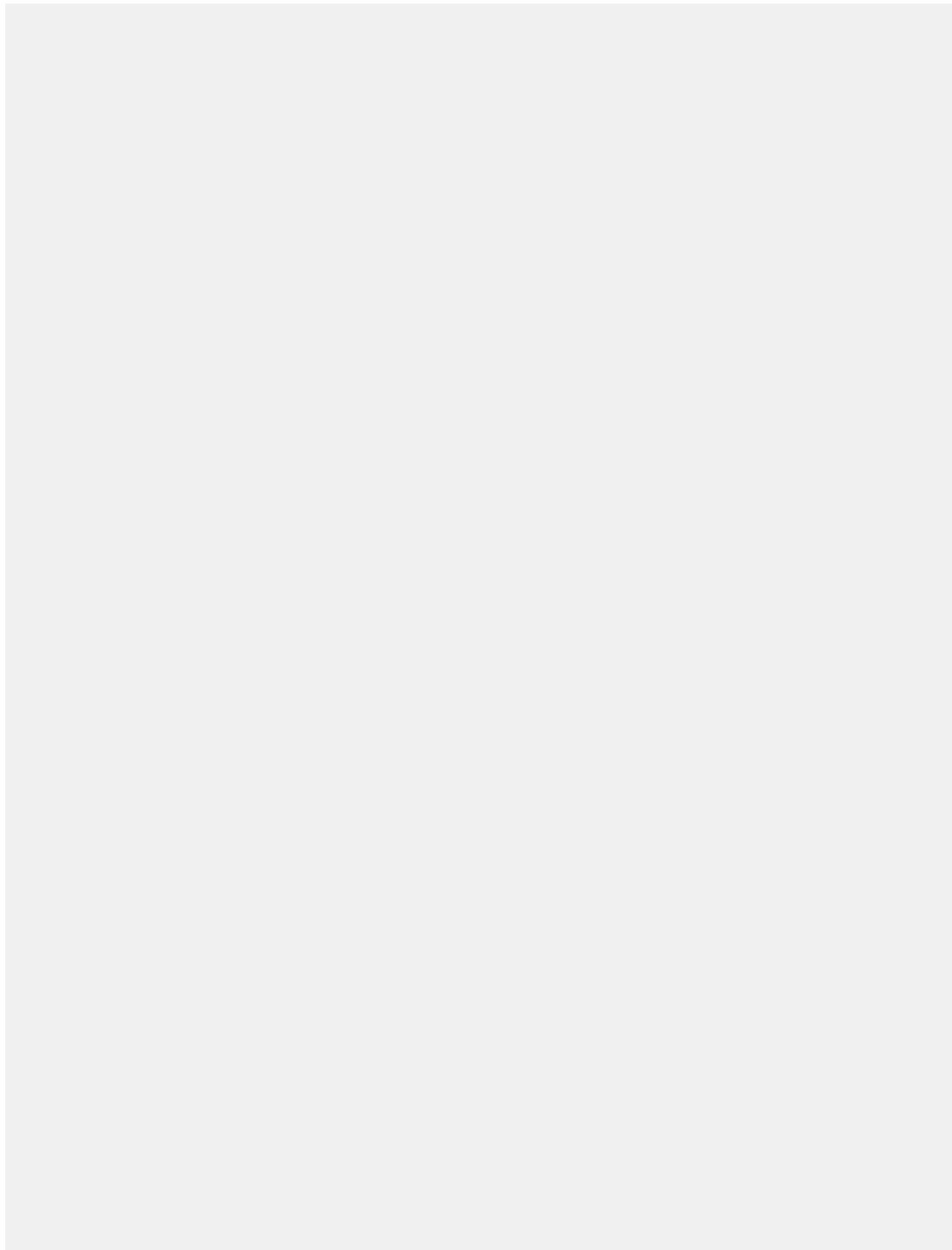












## 関連リンク

[Process.Plugin インターフェースの使用](#)  
[Process.Plugin インターフェース](#)  
[Process.PluginRequest クラス](#)  
[Process.PluginResult クラス](#)  
[Process.PluginDescribeResult クラス](#)

## UninstallHandler インターフェース

アプリケーション開発者は、このインターフェースを実装して、登録者が管理パッケージをアンインストールした後に自動的に実行される Apex コードを指定できます。これにより、登録者の組織の詳細に基づいてクリーンアップおよび通知タスクを実行できます。

アンインストールスクリプトには、デフォルトのガバナ制限が適用されます。パッケージを表す特別なシステムユーザとして実行するため、スクリプトによって実行されるすべての操作は、パッケージによって行われているように見えます。このユーザには、UserInfo を使用してアクセスできます。このユーザは実行時にのみ確認でき、テストの実行中には確認できません。

スクリプトが失敗すると、アンインストールは続行しますが、スクリプトによる変更はコミットされません。スクリプト内のエラーは、パッケージの [Apex エラーを通知] 項目に指定されたユーザにメールされます。ユーザが指定されていない場合、アンインストールの詳細は利用できません。

アンインストールスクリプトには、次の制限があります。バッチジョブ、スケジュールされたジョブ、および今後のジョブの開始、セッション ID へのアクセス、またはコールアウトの実行に使用することはできません。

インターフェースには、という、アンインストール時に実行されるアクションを指定する単一のメソッドがあります。

メソッドは、次の情報を提供するコンテキストオブジェクトを引数として取ります。

- アンインストールが実施される組織の組織 ID。
- アンインストールを開始したユーザのユーザ ID。

コンテキスト引数は、データ型が  
インターフェースは、システムによって自動的に実装されます。  
コンテキスト引数にコールできるメソッドを示しています。

インターフェースであるオブジェクトです。このインター<sup>1</sup>  
インターフェースの次の定義では、

### アンインストールスクリプトの例

以下のアンインストールスクリプトのサンプルは、パッケージのアンインストール時に次のアクションを実行します。

- ・ アンインストールを行ったユーザと組織を示すエントリをフィードに挿入する
- ・ そのユーザにアンインストールを確認するメール通知を作成して送信する

クラスの  
ドは、

メソッドを使って、アンインストールスクリプトをテストできます。このメソッド  
インターフェースを実装するクラスを引数に取ります。

このサンプルでは、  
します。

Apex クラスに実装されたアンインストールスクリプトのテスト方法を示

## WebServiceMock インターフェース

インターフェースでは、WSDL から自動生成されたクラスの Web サービスコールアウトをテ  
ストするときに擬似応答を送信できます。このインターフェースには次のメソッドが含まれます。

名前	引数	戻り値	説明
	Object <i>stub</i> Object <i>request</i> Map<String, Object> <i>response</i> String <i>endpoint</i> String <i>soapAction</i> String <i>requestName</i> String <i>responseNS</i> String <i>responseName</i> String <i>responseType</i>	Void	<p>このメソッドの実装は Apex ランタイムによってコールさ れ、 がコールされた後に Web サービスコ ールアウトが実行されたときに擬似応答を送信します。</p> <p>このインターフェースを実装する場合、<i>response</i> 引数を 目的の応答を表すキー/値ペアに設定します。</p> <p>引数:</p> <ul style="list-style-type: none"><li>• <i>stub</i>: 自動生成されたクラスのインスタンス。</li><li>• <i>request</i>: 呼び出される SOAP Web サービス要求。</li><li>• <i>response</i>: 要求に対して送信する応答を表すキー/値ペ アのコレクション。</li><li>• <i>endpoint</i>: 要求のエンドポイント URL。</li><li>• <i>soapAction</i>: 要求された SOAP 操作。</li><li>• <i>requestName</i>: 要求された SOAP 操作名。</li><li>• <i>responseNS</i>: 応答の名前空間。</li><li>• <i>responseName</i>: WSDL で定義された応答要素の名前。</li><li>• <i>responseType</i>: 自動生成されたクラスで定義された応答 のクラス。</li></ul>

実装例は、「[Web サービスコールアウトのテスト](#)」(ページ 384)を参照してください。

## 第 15 章

### Apex のリリース

---

トピック:

- [変更セットを使用した Apex のリリース](#)
- [Apex をリリースするための Force.com IDE の使用](#)
- [Force.com Migration Tool の使用](#)
- [SOAP API を使用した Apex のリリース](#)

Salesforce 本番組織では Apex を開発することはできません。実際にユーザが利用中のシステムで開発を行う場合、データが不安定になったり、アプリケーションが破損したりする可能性があります。Sandbox または Developer Edition 組織上で、すべての開発作業を行うことを推奨します。

Apex は、次を使用してリリースできます。

- [変更セット](#)
- [Force.com IDE](#)
- [Force.com 移行ツール](#)
- [SOAP API](#)

すべての Apex リリースのクラスとトリガの最大コードユニット数は、5,000 個です。

## 変更セットを使用した Apex のリリース

使用可能なエディション: Enterprise Edition、Unlimited Edition、および Database.com Edition

Sandbox 組織と本番組織間など、接続している組織の間で Apex クラスおよびトリガをリリースできます。Salesforce ユーザインターフェースの送信変更セットを作成し、リリース先組織にアップロードおよびリリースする Apex コンポーネントを追加できます。変更セットについての詳細は、Salesforce オンラインヘルプの「変更セット」を参照してください。

## Apex をリリースするための Force.com IDE の使用

Force.com IDE は Eclipse IDE のプラグインです。Force.com IDE には、Force.com アプリケーションを構築およびリリースする統合インターフェースがあります。開発者および開発チーム向けに設計された IDE には、ソースコードエディタ、テスト実行ツール、ウィザードおよび統合ヘルプなど、Force.com アプリケーション開発を促進するツールが用意されています。基本的なカラー表示エディタ、アウトラインビュー、統合された単体テスト、および保存時の自動コンパイルとエラーメッセージ表示を提供します。



メモ: Force.com IDE は salesforce.com により提供されるユーザとパートナーをサポートする無料のリソースですが、salesforce.com の主登録契約 (MSA) におけるサービスの一部とはみなされません。

Apex を Force.com IDE のローカルプロジェクトから Salesforce 組織にリリースするには、サーバへのリリースウィザードを使用します。



メモ: 本番組織にリリースする場合、次の点に注意します。

- Apex コードの少なくとも 75% が単体テストでカバーされており、かつすべてのテストが成功している。

次の点に注意してください。

- 本番組織にリリースするときに、組織の名前空間内のすべての単体テストが実行されます。
- へのコールは、Apex コードカバー率の対象とはみなされません。
- テストメソッドとテストクラスは、Apex コードカバー率の対象とはみなされません。
- Apex コードの 75% が単体テストでカバーされている必要がありますが、カバー率を上げることだけに集中すべきではありません。アプリケーションのすべてのユースケース(正・誤両方の場合や単一データだけでなく複数データの場合)の単体テストを作成するようにしてください。このような多様なユースケースのテストコードを実装することが 75% 以上のカバー率につながります。

- すべてのトリガについて何らかのテストを行う。
- すべてのクラスとトリガが正常にコンパイルされる。

サーバへのリリースウィザードの使用方法についての詳細は、Eclipse で入手できる Force.com IDE ドキュメントの「Deploying Code with the Force.com IDE (Force.com IDE によるコードのリリース)」を参照してください。

## Force.com Migration Tool の使用

Force.com IDE に加えて、Apex のリリースにスクリプトを使用することもできます。

Apache の Ant 開発ツールを使用して Developer Edition または Sandbox を使用している組織から Database.com 本番組織に Apex リリース用のスクリプトを使用する場合は、Force.com 移行ツールをダウンロードします。



メモ: Force.com 移行ツールは salesforce.com により提供されるユーザとパートナーをサポートする無料のリソースですが、salesforce.com の主登録契約 (MSA) におけるサービスの一部とはみなされません。

Force.com 移行ツールを使うには、次を行います。

1. にアクセスし、Java JDK のバージョン 6.1 以上をリリースマシンにインストールします。

2. にアクセスし、Apache Ant のバージョン 1.6 以上をリリースマシンにインストールします。

3. 環境変数 ( 、 、 など) を、 の『Ant Installation Guide』で指定されたように設定します。

4. コマンドプロンプトを開き、 を入力して、JDK と Ant が正しくインストールされているか確認してください。出力は次のようになります。

5. リリースマシン上で Salesforce にログインします。[設定] から、[開発] > [ツール] をクリックして、次に、[Force.com 移行ツール] をクリックします。

6. ダウンロードしたファイルを、任意のディレクトリに展開します。Zip ファイルには次が含まれます。

- ツールの使用方法を説明した ファイル
- Ant タスクを含む Jar ファイル:
- 次の内容を含むサンプルフォルダ:

◊	と	を含む	フォルダ
◊	を含む	フォルダ	
◊	例で使用するカスタムオブジェクトを含む	フォルダ	
◊	組織から例を削除するための XML ファイルを含む	フォルダ	
◊	の Ant タスクを実行するための認証情報を指定するサンプル	ファイル	
◊	および API コールを実行するサンプル	ファイル	

7. 展開したファイルから、 を ant lib ディレクトリにコピーしてください。ant lib ディレクトリは、Ant インストール先のルートフォルダにあります。

8. 展開したファイル内のサンプルサブディレクトリを開きます。

9. ファイルを編集します。

a. Salesforce 本番組織ユーザ名およびパスワードを、と  
項目にそれぞれ入力します。



メモ:

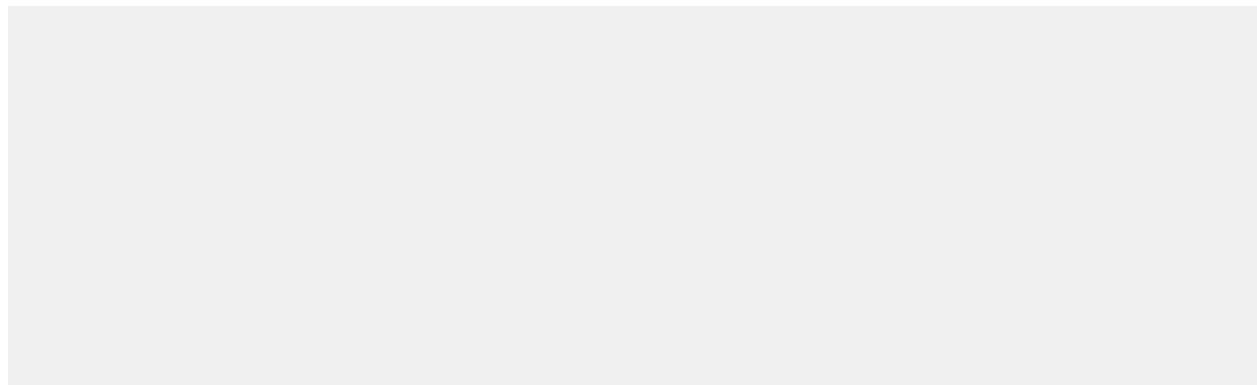
- Apex を編集するための権限を持っているユーザ名を指定する必要があります。
- 信頼されないネットワークから Force.com Migration Tool を使用する場合、パスワードにセキュリティトークンを追加します。セキュリティトークンについての詳細は、Salesforce オンラインヘルプの「セキュリティトークンのリセット」を参照してください。

b. sandbox 組織にリリースする場合、項目を  
に変更してください。

10. サンプルディレクトリのコマンドウィンドウを開きます。

11. を入力します。これは、Force.com 移行ツールで提供されたサンプルクラスと Account トリガを使用して、API コールを実行します。

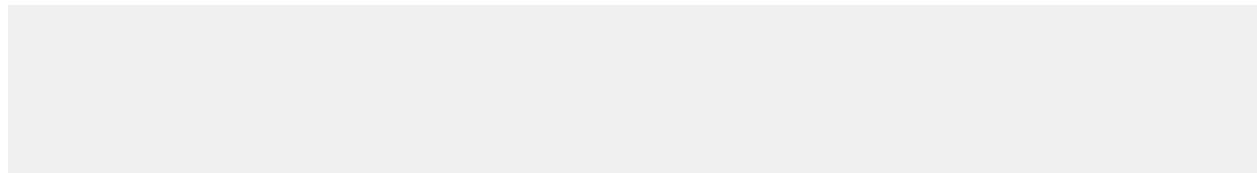
は、ファイルの  
という名前の Ant ターゲットをコールします。



の詳細は、「[\[ \]について](#)」(ページ 954)を参照してください。

12. の実行の一部として追加されたテストクラスとトリガを削除するには、コマンドウィンドウ  
内で次を入力します。

は、ファイル内で  
という Ant ターゲットをコールします。



## deploy について

コールは、次のすべてが一致する必要がある場合にのみ正常に完了します。

- Apex コードの少なくとも 75% が単体テストでカバーされており、かつすべてのテストが成功している。

次の点に注意してください。

- 本番組織にリリースするときに、組織の名前空間内のすべての単体テストが実行されます。
- へのコールは、Apex コードカバー率の対象とはみなされません。
- テストメソッドとテストクラスは、Apex コードカバー率の対象とはみなされません。
- Apex コードの 75% が単体テストでカバーされている必要がありますが、カバー率を上げることだけに集中すべきではありません。アプリケーションのすべてのユースケース（正・誤両方の場合や单一データだけでなく複数データの場合）の単体テストを作成するようにしてください。このような多様なユースケースのテストコードを実装することが 75% 以上のカバー率につながります。

- すべてのトリガについて何らかのテストを行う。
- すべてのクラスとトリガが正常にコンパイルされる。

一度に複数の Metadata API コールを実行することはできません。

Force.com 移行ツールは、リリーススクリプトに組み込み可能なタスク を提供します。組織のクラスとトリガが含まれるように サンプルを変更できます。 タスクのプロパティは次のとおりです。

### username

Salesforce 本番組織にログインするためのユーザ名。

### password

Salesforce 本番組織にログインするための関連パスワード。

### serverURL

ログインする Salesforce サーバの URL。値を指定しない場合、デフォルトは です。

### deployRoot

リリースする他のメタデータと同様に、Apex クラスおよびトリガを含むローカルディレクトリ。必要なファイル構造を作成する最適な方法は、組織または Sandbox から取得する方法です。詳細は、「[について](#)」(ページ 955)を参照してください。

- Apex クラスファイルは、classes という名前のサブディレクトリ内にある必要があります。次の名前の 2 つのファイルが各クラスにある必要があります。

- classname.cls
- classname.cls-meta.xml

たとえば、 と です。 -meta.xml ファイルにはクラスの API バージョンと状況 (有効/無効) が含まれます。

- Apex トリガファイルは、triggers という名前のサブディレクトリ内にある必要があります。次の名前の 2 つのファイルが各トリガにある必要があります。

- triggername.trigger

#### ◊ triggername.trigger-meta.xml

たとえば、です。-meta.xml ファイルにはトリ  
ガの API バージョンと状況(有効/無効)が含まれます。

- ルートディレクトリには、リリースするすべてのクラス、トリガ、およびその他のオブジェクトをリストした XML ファイルが含まれます。
- ルートディレクトリには、組織から削除するすべてのクラス、トリガ、およびその他のオブジェクトをリストした XML ファイルが必要に応じて含まれます。

#### checkOnly

クラスとトリガがリリース先環境にリリースされるかどうかを指定します。このプロパティは boolean 値を持ち、組織にクラスとトリガを保存しない場合は、保存する場合はを設定します。値を指定しない場合、デフォルトはです。

#### runTests

実行する単体テストを含むクラスの名前。



メモ: このパラメータは、Salesforce 本番組織にリリースされるときには無視されます。組織の名前空間内のすべての単体テストが実行されます。

#### runAllTests

このプロパティは Boolean 値を持ち、組織内のすべてのテストを実行する場合は、実行しない場合はを設定します。にを指定する場合、の値を指定する必要はありません。

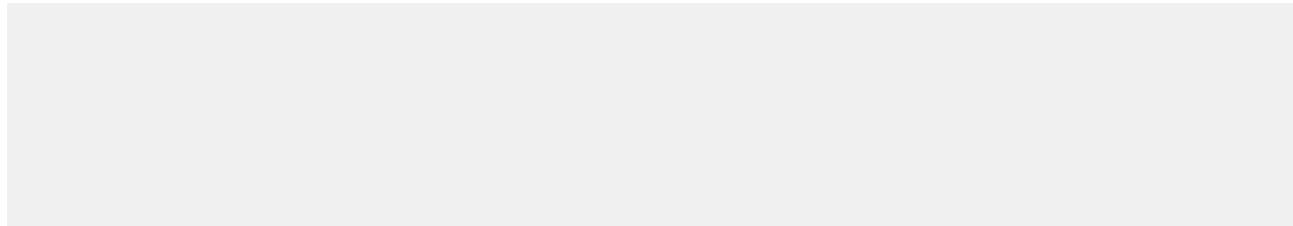


メモ: このパラメータは、Salesforce 本番組織にリリースされるときには無視されます。組織の名前空間内のすべての単体テストが実行されます。

## retrieveCode について

Sandbox または本番組織からクラスとトリガを取得するには、コールを使用します。通常のリリースサイクル中は、新しいクラスとトリガ用の正しいディレクトリ構造を取得するために、の前にを実行します。ただし、次の例では、取得するものがあることを確認するためにが最初に使用されます。

既存の組織からクラスとトリガを取得するには、次の構築ターゲットの例に示すように retrieve ant タスクを使用します。



## 取得タスクのプロパティ

**ファイル** は、取得されるメタデータコンポーネントをリストします。この例では、2つのクラスと1つのトリガを取得します。取得されたファイルはディレクトリに配置され、ディレクトリ内に存在するものがすべて上書きされます。

取得タスクのプロパティは次のとおりです。

### **username**

Salesforce 本番組織にログインするためのユーザ名。

### **password**

Salesforce 本番組織にログインするための関連パスワード。

### **serverURL**

ログインする Salesforce サーバの URL。値を指定しない場合、デフォルトは <https://login.salesforce.com> です。

### **apiversion**

ファイルが取得される先の Metadata API のバージョン。

### **retrieveTarget**

ファイルのコピー先のディレクトリ。

### **unpackaged**

取得する必要があるファイルのリストを含むファイルの名前。このパラメータまたは [unpackagedjar](#) のどちらかを指定します。

### **packageName**

取得されるパッケージの名前。

表 6 : build.xml 取得先項目設定

項目	説明
	必須。 ログイン用の Salesforce ユーザ名。
	必須。 このプロジェクトに関連付けられた組織にログインするために使用するユーザ名 セキュリティトークンを使用している場合は、パスワードの最後に 25 衔のトークン値を貼り付けます。この接続に関連付けられるユーザ名は、「すべてのデータの編集」権限を持っている必要があります。通常、これはシステム管理者のみに有効です。
	省略可能。 Salesforce サーバ URL (空白の場合、デフォルトは <a href="https://login.salesforce.com">https://login.salesforce.com</a> )。 Sandbox では、 <a href="#">unpackagedjar</a> を使用します。

項目	説明
	省略可能。デフォルトは 5000 です。salesforce.com で結果を取得するまで待機する各ポーリング間隔の時間(ミリ秒単位)。
	省略可能。デフォルトは 10 です。レポートの結果に対して salesforce.com をポーリングする回数。
	必須。メタデータファイルを取得する先のディレクトリ構造のルート。
	省略可能。取得するコンポーネントを指定するファイルマニフェストの名前。
	省略可能。デフォルトは False です。取得される内容が单一のパッケージかどうかを指定します。
	省略可能。取得するパッケージ名のリスト。
	省略可能。取得するファイル名のリスト。

## runTests() について

Force.com 移行ツールでは `runTests()` を使用するだけでなく、`runTests()` API コールも使用できます。このコードには次のプロパティがあります。

### `class`

単体テストを含むクラスの名前。このプロパティは複数回指定できます。

### `alltests`

すべてのテストを実行するかどうか指定します。このプロパティは boolean 値を持ち、すべてのテストを実行する場合は `true`、そうでない場合は `false` を設定します。

### `namespace`

テストを実行する名前空間。名前空間を指定する場合、その名前空間内のすべてのテストが実行されます。

## SOAP API を使用した Apex のリリース

Force.com IDE、変更セット、または Force.com 移行ツールを使用して Apex をリリースしない場合、次の SOAP API コールを使用して Apex を開発組織または Sandbox 組織にリリースできます。

設定不要な項目値と同様に、これらすべてのコールは、クラスまたはトリガを含む Apex コードを実施します。

# 付録

## 付録 A

### 納入先請求書の例

この付録では、Apex アプリケーションの例を示します。この例は Hello World 例よりも複雑です。

- ・ [納入先請求書の例の模擬体験 \(ページ 958\)](#)
- ・ [納入先請求書のコード例 \(ページ 961\)](#)

#### 納入先請求書の例の模擬体験

このセクションで示すサンプルアプリケーションには、Apex と組み合わされた従来の Salesforce 機能が含まれています。一般的なイディオムと共に、Apex の構文およびセマンティック上の機能の多くが、このアプリケーションに例示されます。



メモ: Hello World と納入先請求書のサンプルでは、カスタム項目およびオブジェクトが必要です。項目やオブジェクトを自分で作成したり、オブジェクト、項目および Apex コードを管理パッケージとして Force.com AppExchange からダウンロードできます。詳細は、  
を参照してください。

#### シナリオ

このサンプルアプリケーションでは、納入先請求書、または注文を新規作成し、品目を請求書に追加します。納入費用を含む注文金額合計は、請求書に追加または削除された品目に基づいて自動的に計算され、更新されます。

#### データおよびコードモデル

このサンプルアプリケーションでは、Item と Shipping\_invoice の新しい 2 つのオブジェクトを使用します。次のように想定します。

- Item A は shipping\_invoice1 および shipping\_invoice2 のいずれの注文にも含めることができません。2人の顧客は同じ(物理的)商品を取得できません。
- 消費税率は 9.25% です。
- 輸送料は 1 ポンドあたり 75 セントです。
- 注文が \$100 を超えた場合、輸送料の割引が適用されます(輸送量は無料)。

Item カスタムオブジェクトの項目には、次のものがあります。

名前	型	説明
Name	String	品目の名前
Price	Currency	品目の価格
Quantity	Number	注文に含まれる品目数
Weight	Number	品目の重量。輸送費用の計算に使用します。
Shipping_invoice	Master-Detail (shipping_invoice)	この品目が関連付けられた注文

Shipping\_invoice カスタムオブジェクトの項目は次のとおりです。

名前	型	説明
Name	String	納入先請求書/注文の名前
Subtotal	Currency	小計
GrandTotal	Currency	消費税、輸送料を含む合計金額
Shipping	Currency	輸送料として請求する金額(1 ポンドあたり \$0.75 と想定)
ShippingDiscount	Currency	小計金額が \$100 に達した場合に 1 回のみ適用
Tax	Currency	消費税金額(9.25% と想定)
TotalWeight	Number	すべての品目の総重量

このアプリケーションのすべての Apex がトリガに含まれます。このアプリケーションには、次のトリガがあります。

オブジェクト	トリガ名	実行タイミング	説明
Item	Calculate	挿入後、更新後、削除後	納入先請求書を更新し、合計および輸送料を計算します。
Shipping_invoice	ShippingDiscount	更新後	納入先請求書を更新し、送料割引があるかどうかを計算します。

次に、ユーザアクションとトリガが実行されるタイミングの一般的な流れを示します。

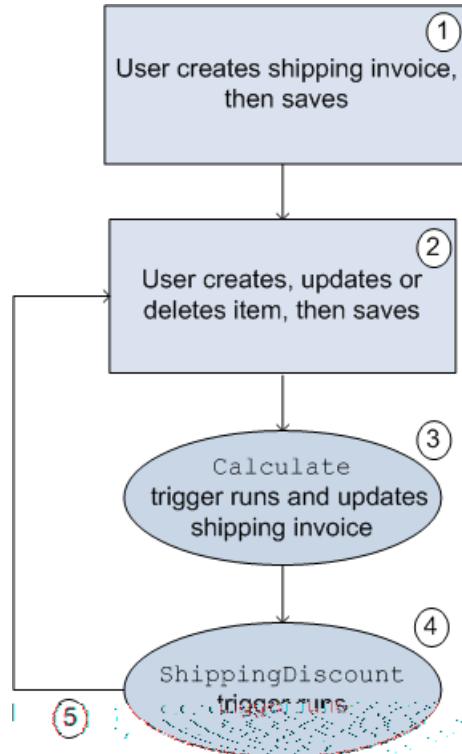


図 12: ショッピングカートアプリケーションのユーザアクションおよびトリガの流れ

1. ユーザが [注文] > [新規] をクリックして納入先請求書の名前を付け、[保存] をクリックします。
2. ユーザが [新規項目] をクリックし、情報を入力して [保存] をクリックします。
3. Calculate トリガが実行されます。Calculate トリガの一部として、納入先請求書が更新されます。
4. ShippingDiscount トリガが実行されます。
5. 請求書の品目を追加、削除、変更できます。

納入先請求書のコード例にトリガおよびテストクラスが表示されます。このコードのコメントは、機能について説明します。

### 納入先請求書アプリケーションのテスト

パッケージの一部としてアプリケーションを追加するには、単体テストでコードの 75% をカバーする必要があります。そのため、納入先請求書アプリケーションの一部は、トリガのテストに使用するクラスとなります。

テストクラスは、次のアクションが正常に行われたことを確認します。

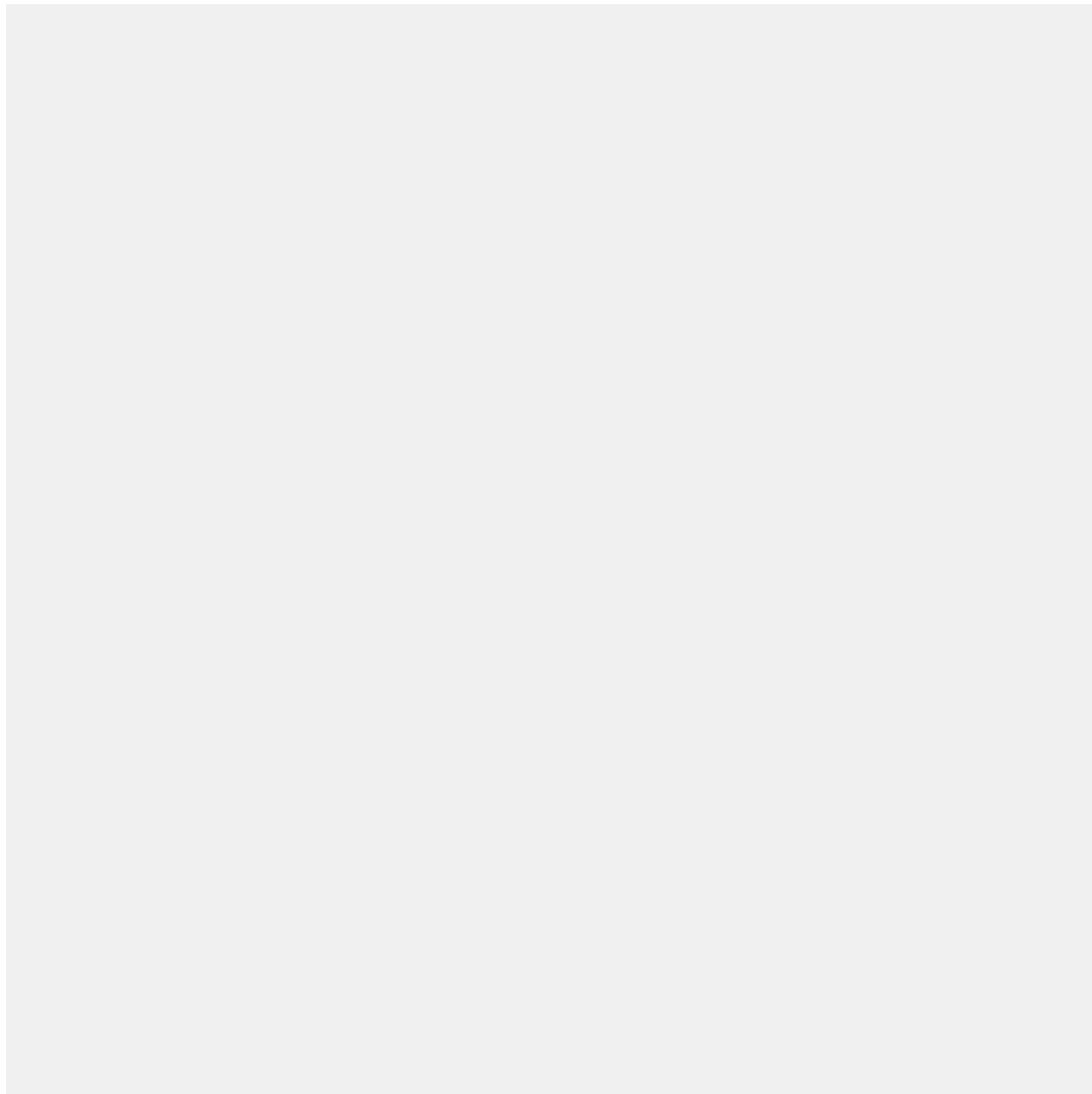
- ・ 品目の挿入
- ・ 品目の更新
- ・ 品目の削除
- ・ 送料割引の適用
- ・ 不正入力のネガティブテスト

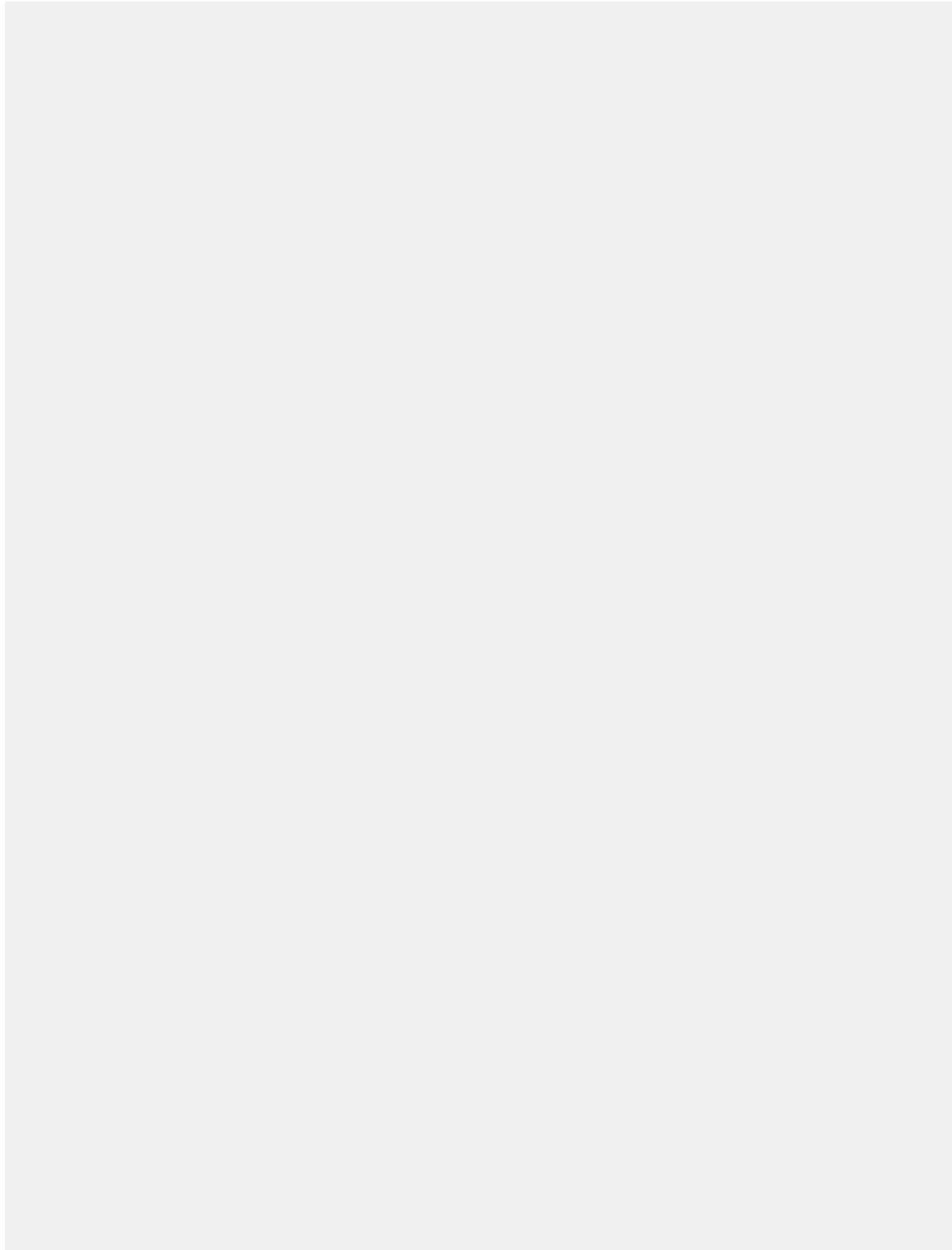
## 納入先請求書のコード例

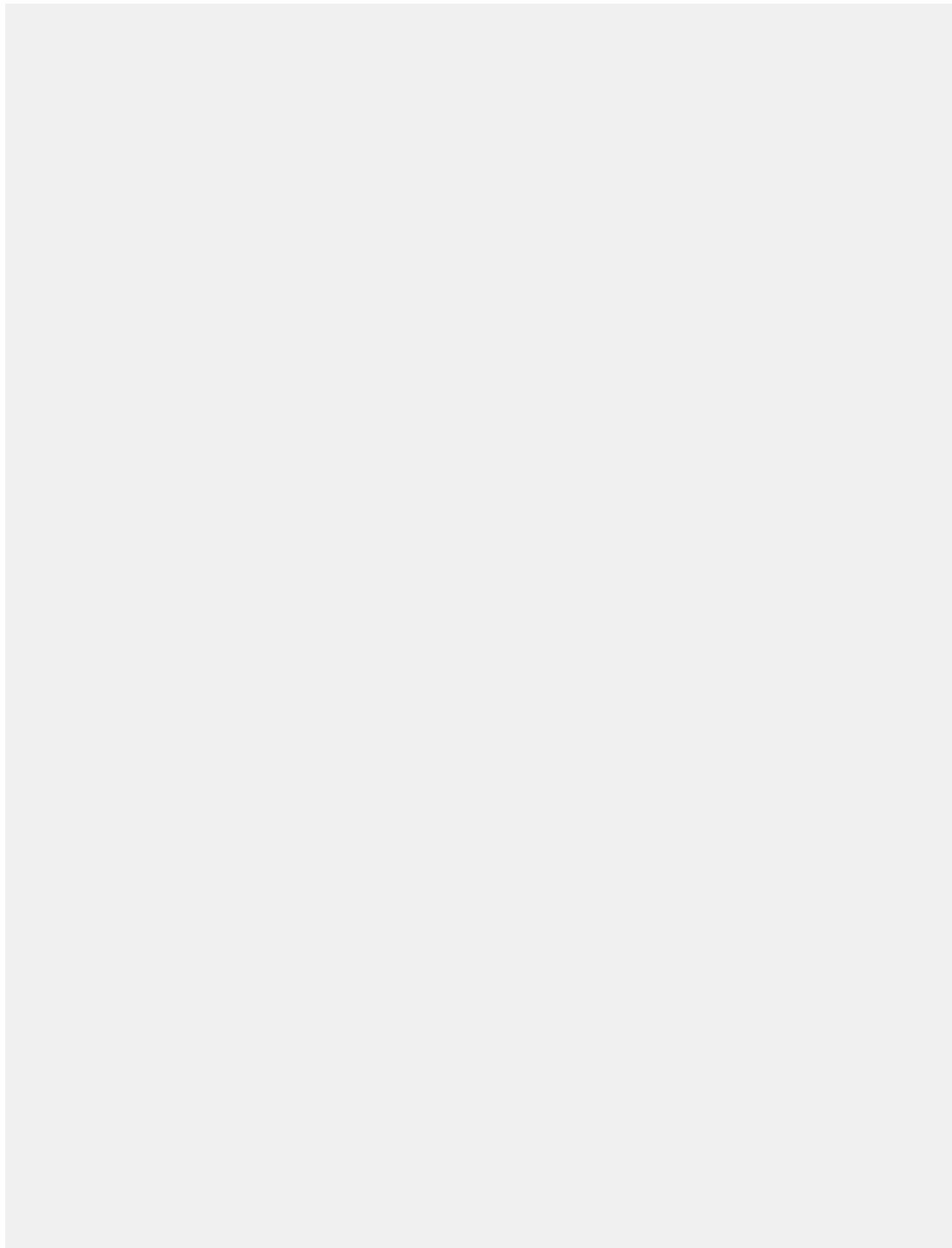
次のトリガおよびテストクラスは、納入先請求書のアプリケーション例を構成します。

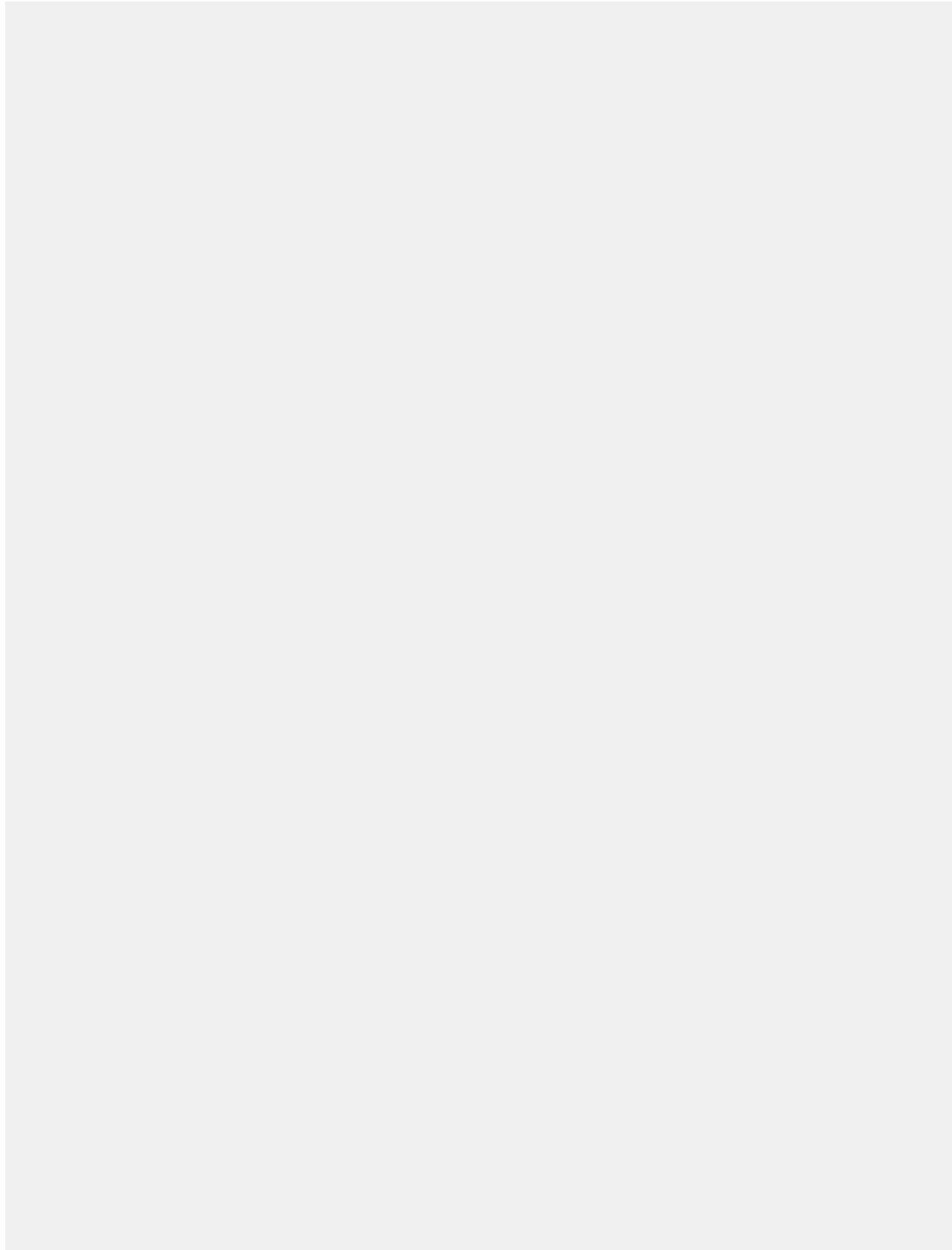
- [Calculate トリガ](#)
- [ShippingDiscount トリガ](#)
- [テストクラス](#)

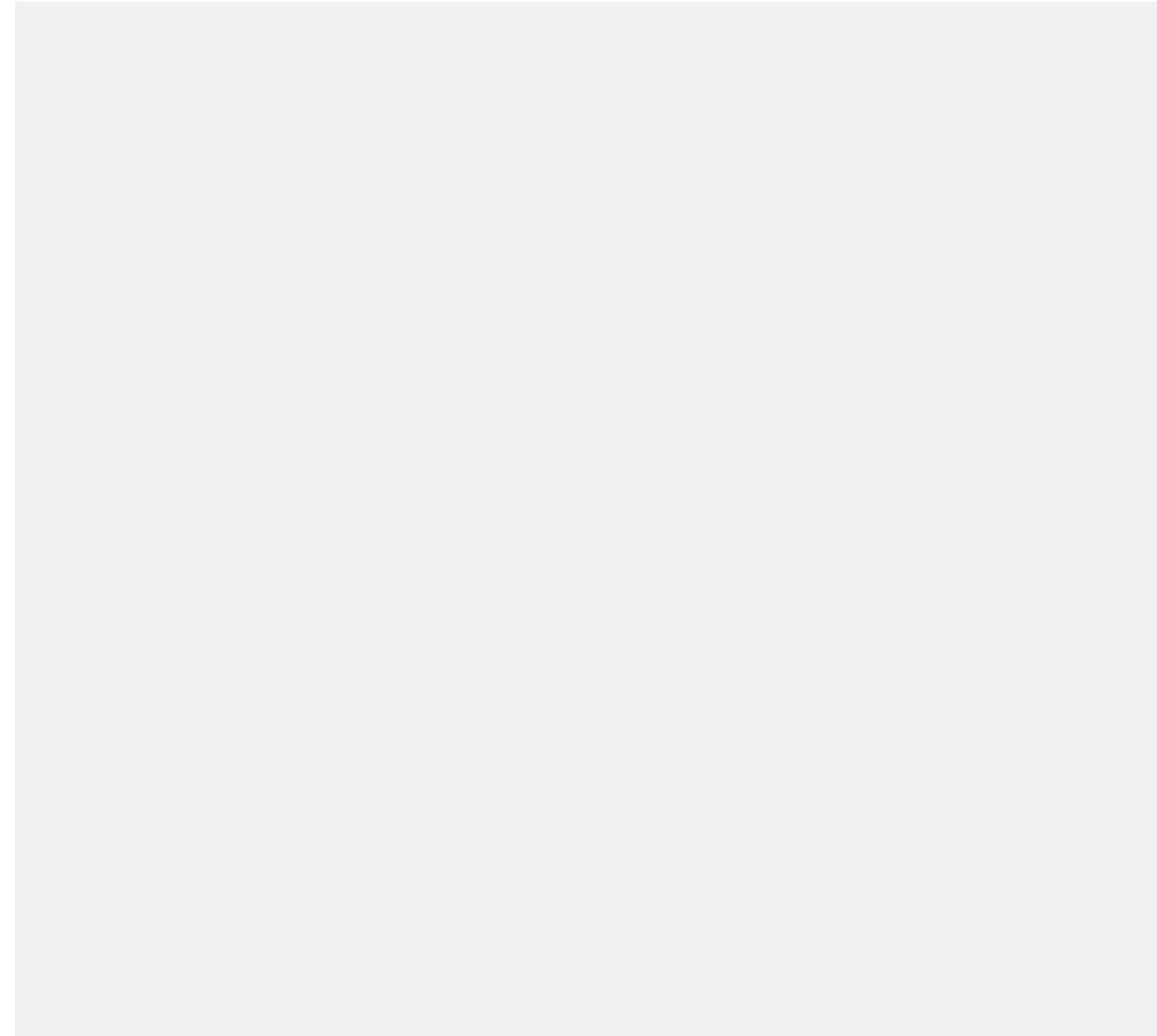
### Calculate トリガ



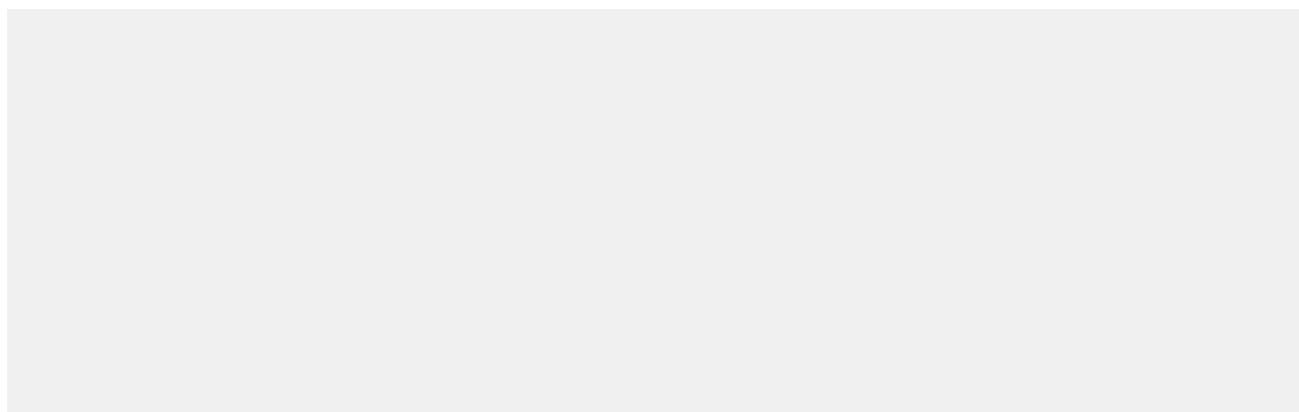


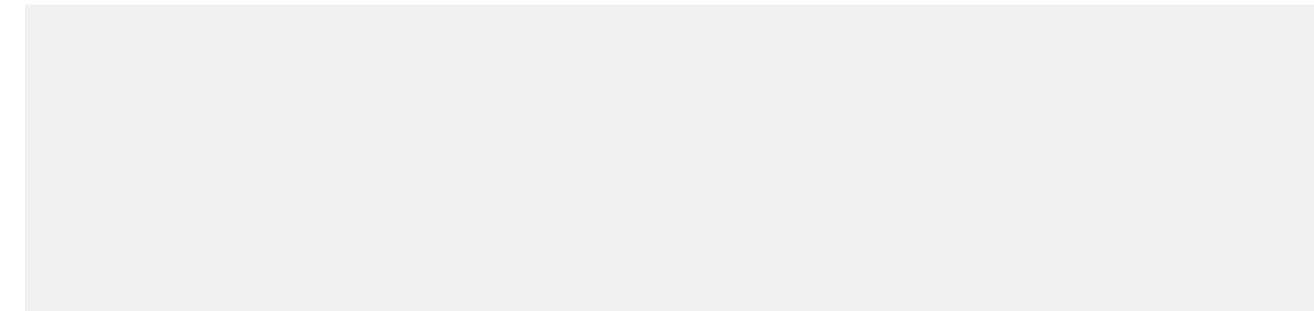




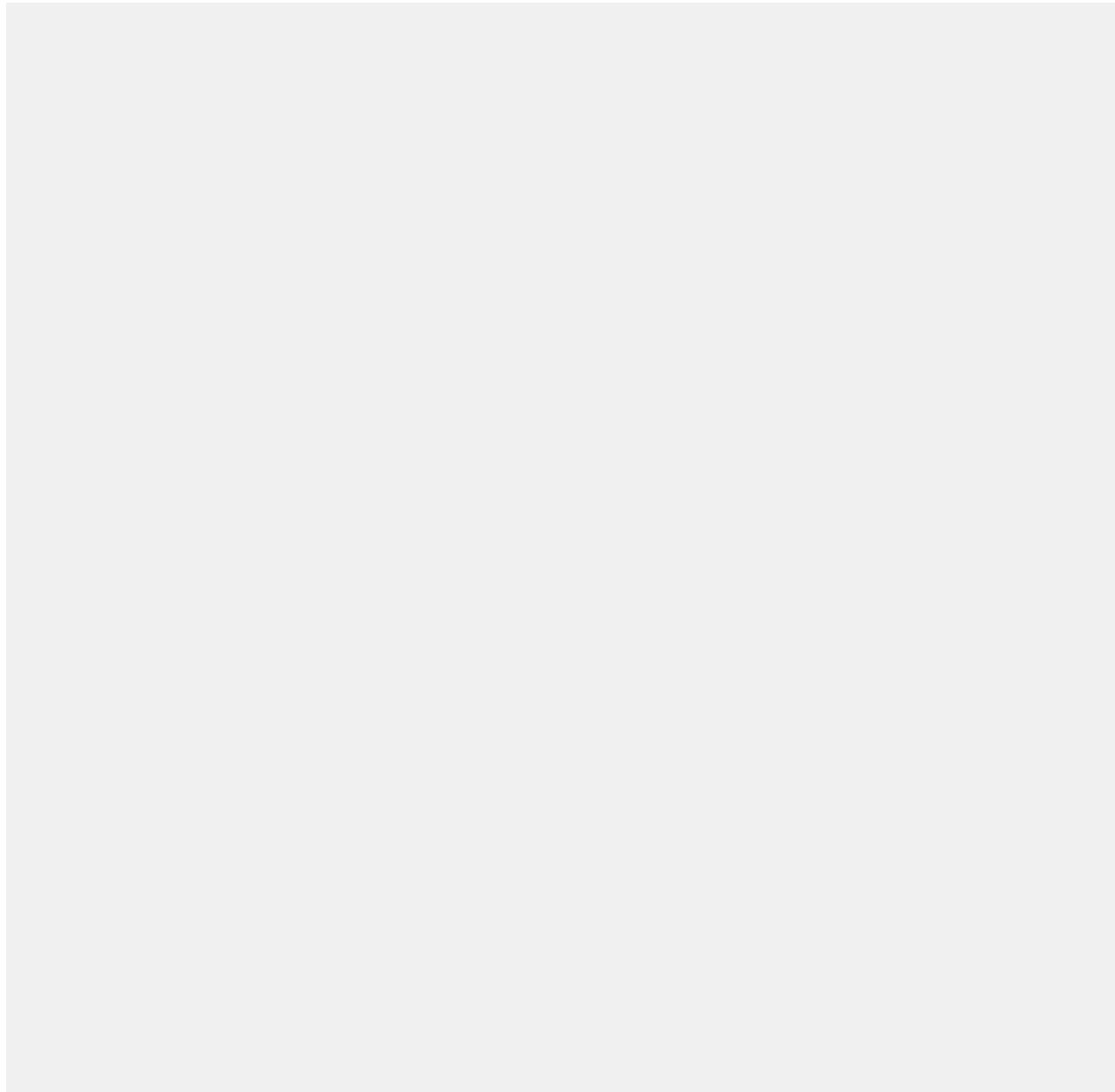


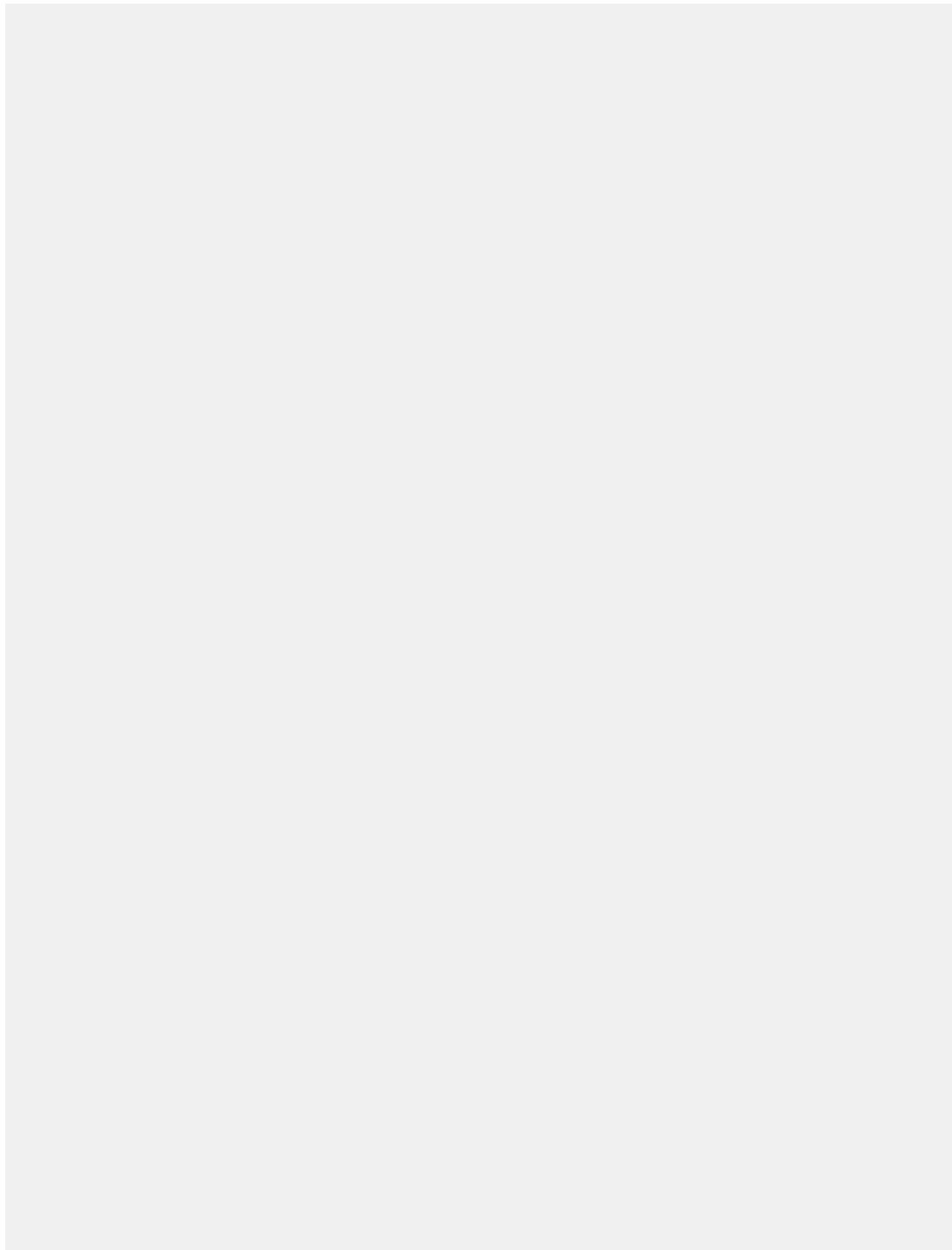
ShippingDiscount トリガ

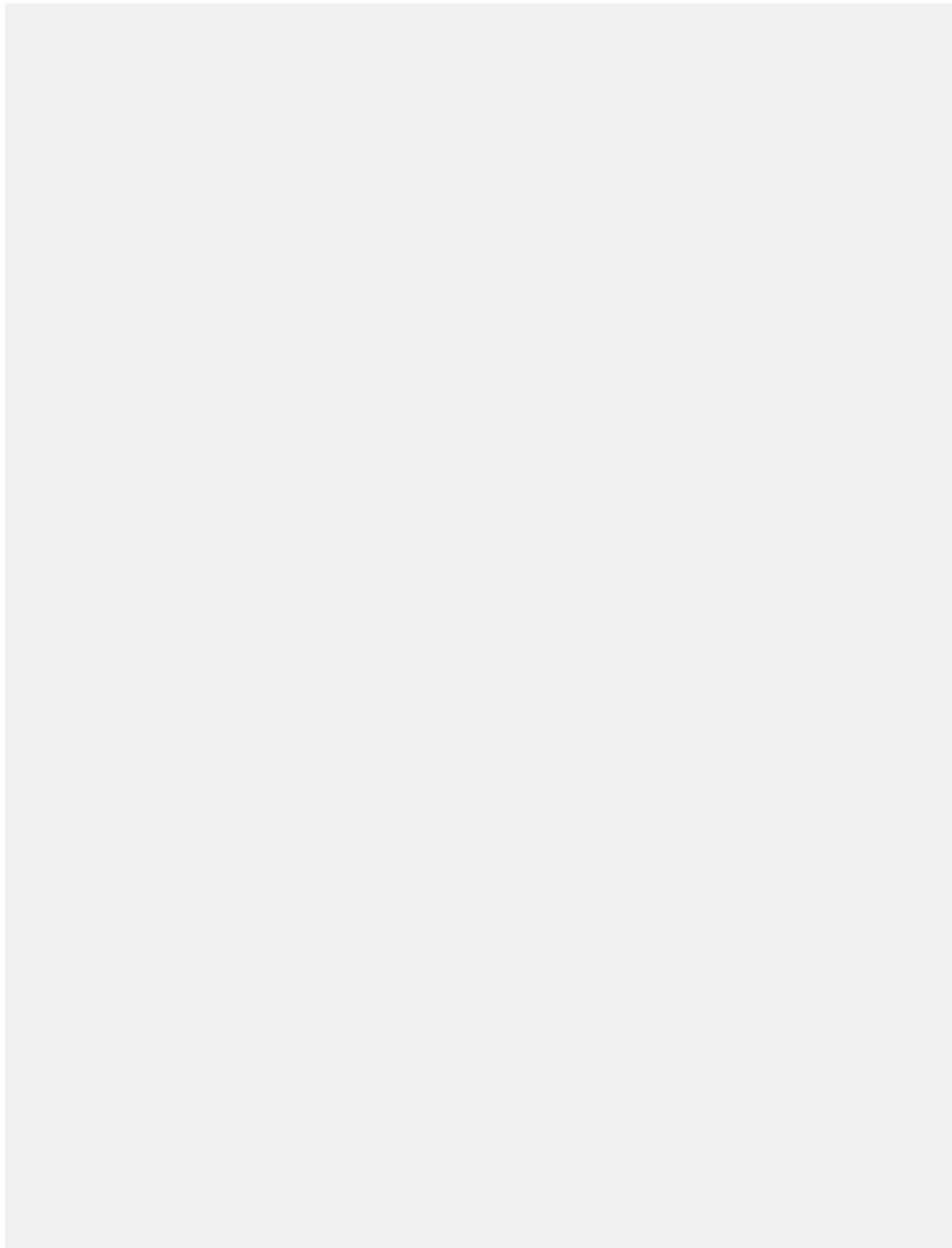


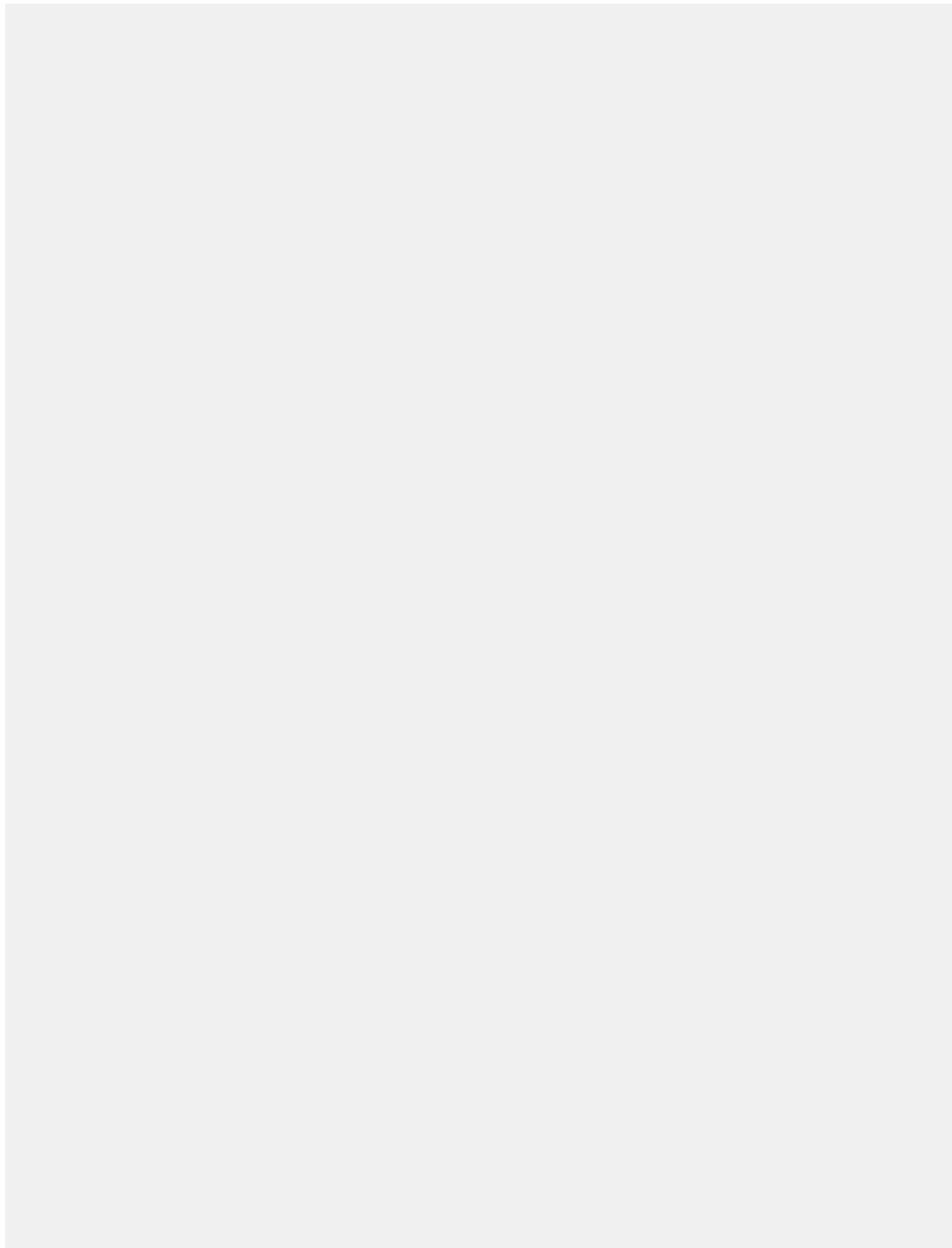


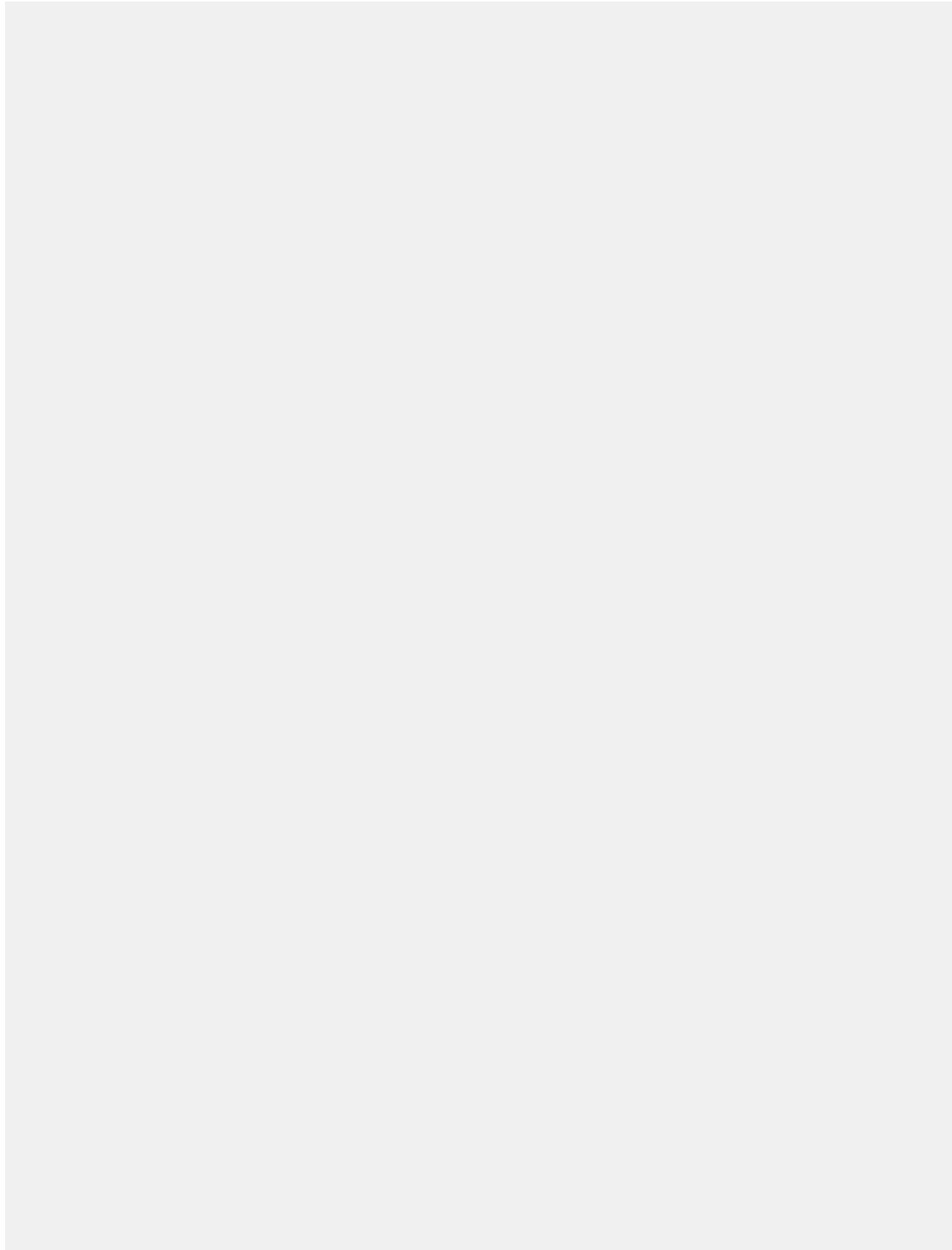
納入先請求書のテスト

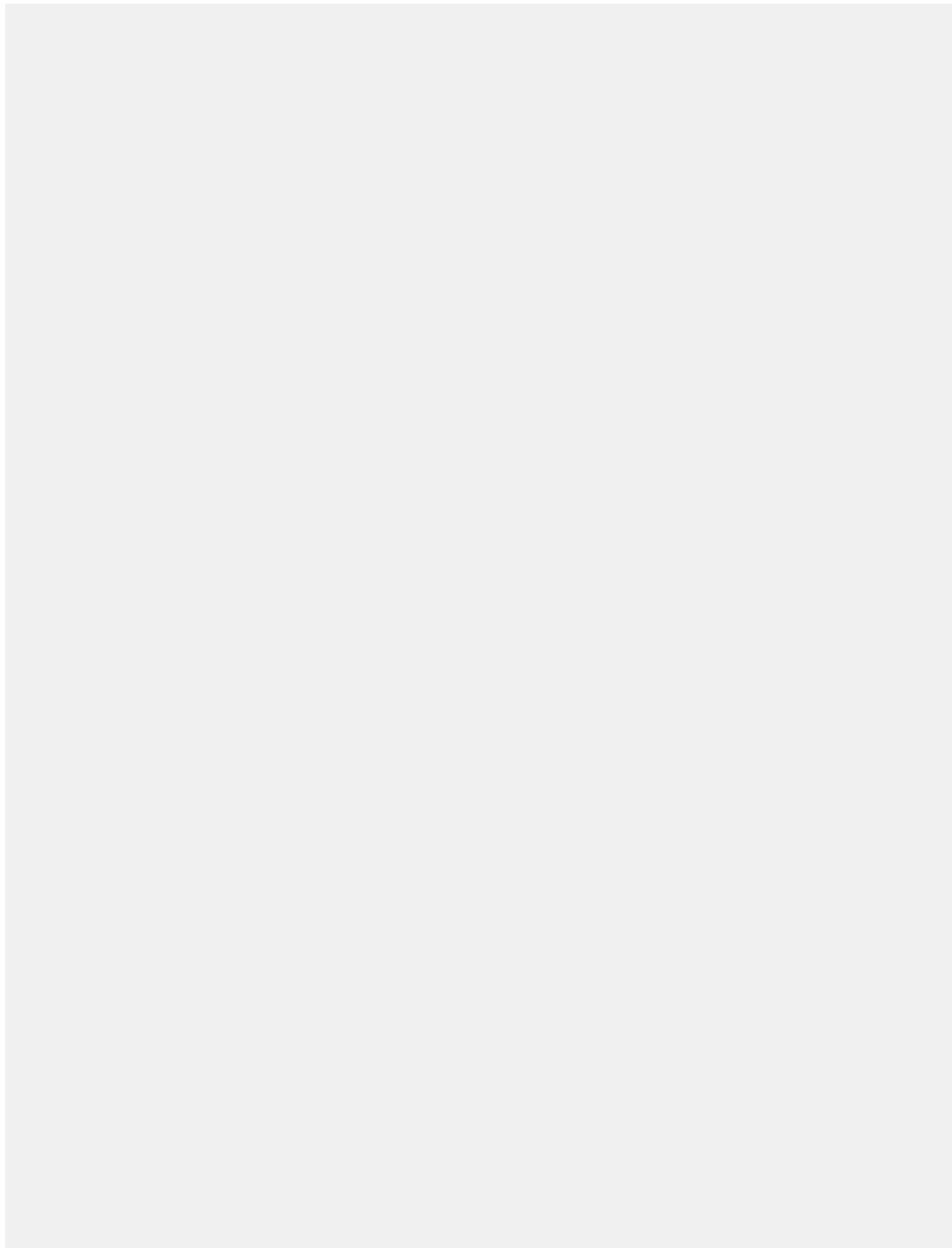


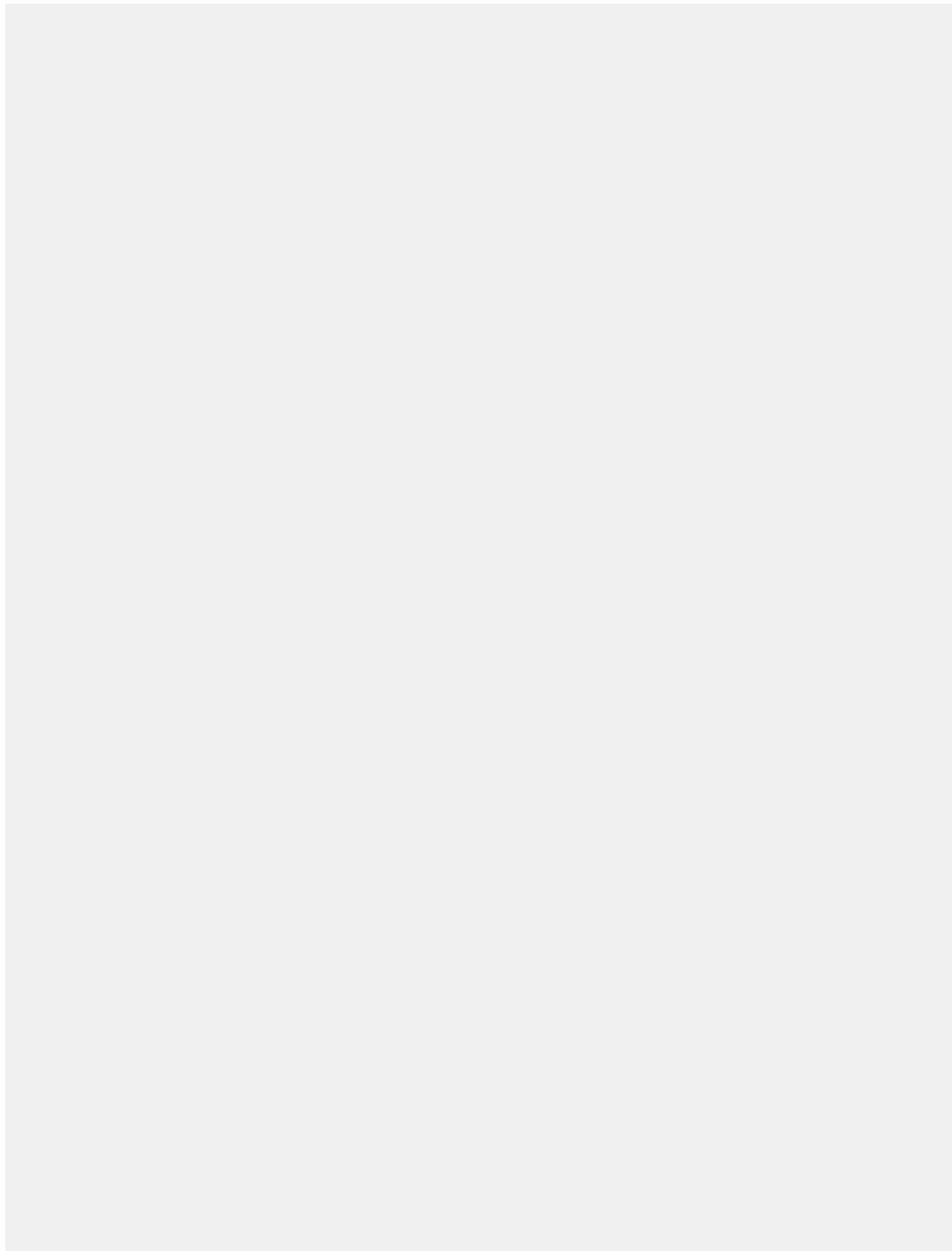


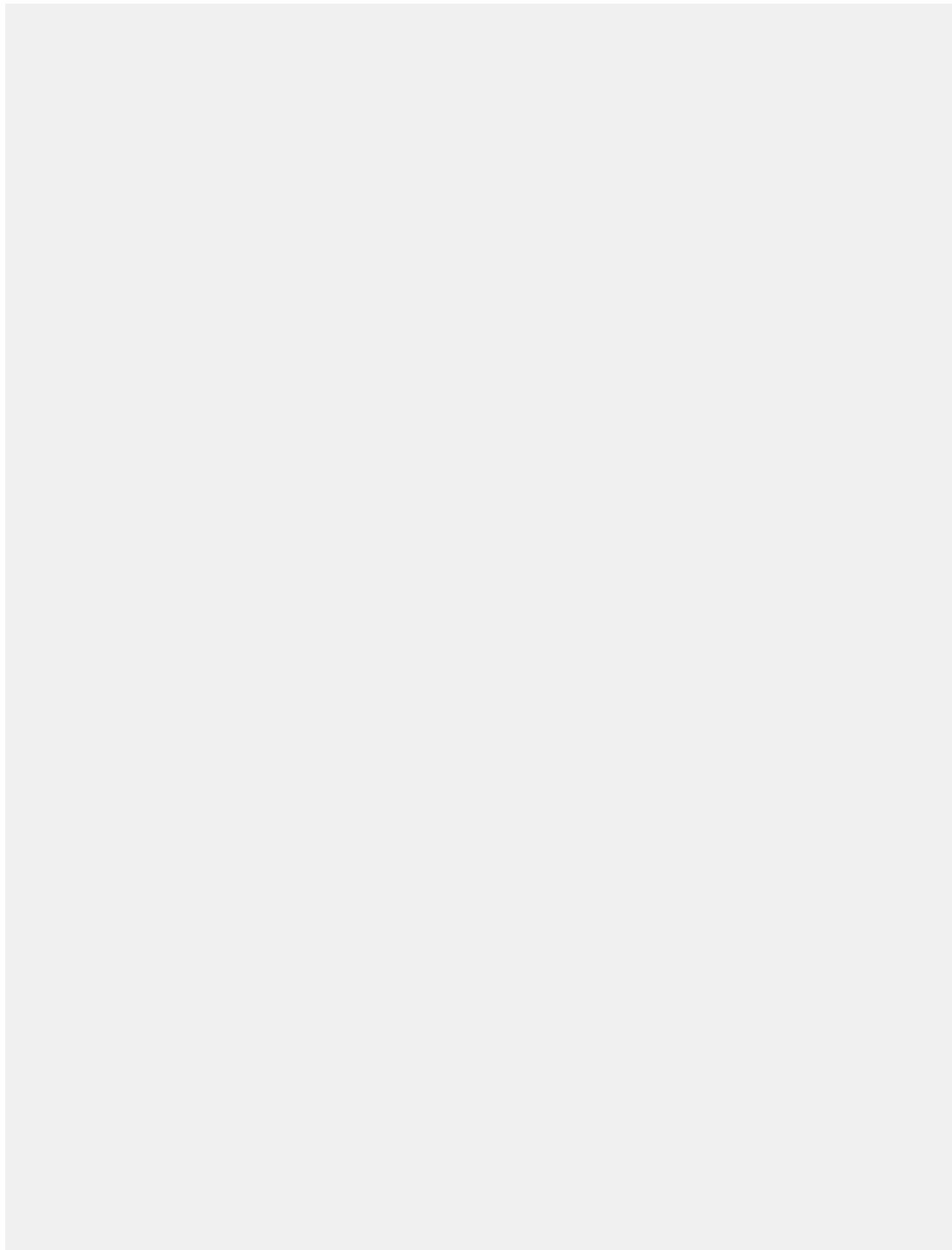


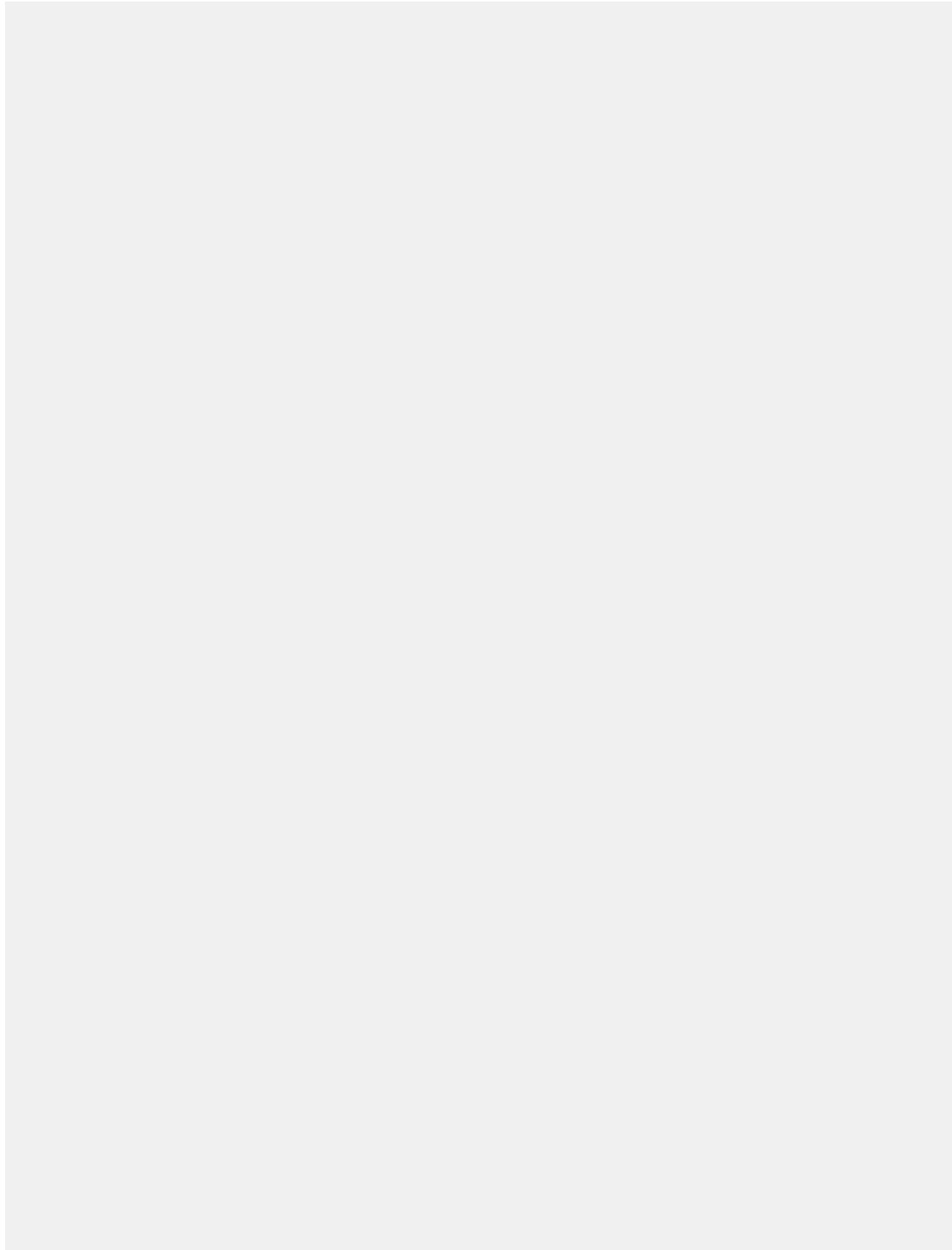


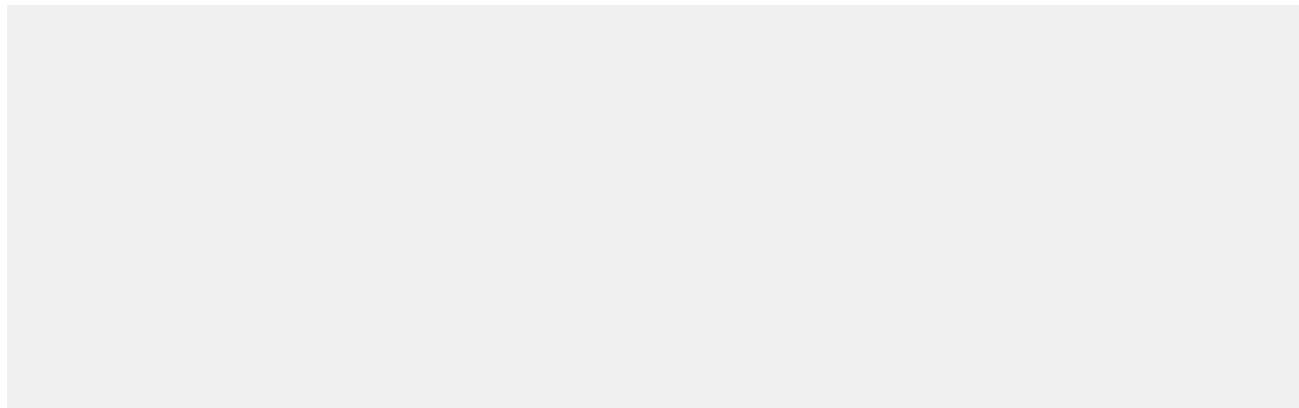












# 付録 B

## 予約キーワード

次の語はキーワードとしてのみ使用できます。



メモ: アスタリスク (\*) が付いたキーワードは今後使用するために予約されています。

表7: 予約キーワード

abstract	having*	retrieve*
activate*	hint*	return
and	if	returning*
any*	implements	rollback
array	import*	savepoint
as	inner*	search*
asc	insert	select
autonomous*	instanceof	set
begin*	interface	short*
bigdecimal*	into*	sort
blob	int	stat*
break	join*	super
bulk	last_90_days	switch*
by	last_month	synchronized*
byte*	last_n_days	system
case*	last_week	testmethod
cast*	like	then*
catch	limit	this
char*	list	this_month*
class	Long	this_week
collect*	loop*	throw
commit	map	today
const*	merge	tolabel
continue	new	tomorrow
convertcurrency	next_90_days	transaction*

decimal	next_month	trigger
default*	next_n_days	true
delete	next_week	try
desc	not	type*
do	null	undelete
else	nulls	update
end*	number*	upsert
enum	object*	using
exception	of*	virtual
exit*	on	webservice
export*	or	when*
extends	outer*	where
false	override	while
final	package	yesterday
finally	parallel*	
float*	pragma*	
for	private	
from	protected	
future	public	
global		
goto*		
group*		

次の単語は、予約語ではない特殊なキーワードで、識別子として使用できます。

- after
- before
- count
- excludes
- first
- includes
- last
- order
- sharing
- with

# 付録 C

## Apex および Visualforce 開発のセキュリティのヒント

### セキュリティについて

Apex および Visualforce ページの強力な組み合わせにより、Force.com 開発者は、Salesforce にカスタム機能およびビジネスロジックを提供したり、Force.com プラットフォーム内部で実行するまったく新しいスタンダロン製品を作成することができます。ただし、プログラミング言語と同様、開発者はセキュリティ関連の不備について認識する必要があります。

Salesforce.com は、複数のセキュリティ防御を Force.com プラットフォーム自体に統合しました。ただし、不注意な開発者は多くの場合に組み込み防御を回避し、アプリケーションと顧客をセキュリティ上のリスクにさらしている場合があります。開発者が Force.com プラットフォーム上で犯す多くのコーディングエラーは、一般的な Web アプリケーションのセキュリティ脆弱性と類似しています。一部のコーディングエラーは Apex 固有のものです。

AppExchange のアプリケーションを認証するには、開発者はここで説明するセキュリティ上の弱点について学習および理解する必要があります。詳細は、<http://wiki.developerforce.com/page/Security> にある Developer Force の Force.com セキュリティリソースのページを参照してください。

### クロスサイトスクリプト (XSS)

クロスサイトスクリプト (XSS) の攻撃は、悪意のある HTML またはクライアント側のスクリプトが Web アプリケーションに提供される、幅広い範囲の攻撃となります。Web アプリケーションには、Web アプリケーションのユーザに対する悪意のあるスクリプトが含まれています。ユーザは、知らぬ間に攻撃の被害者となります。攻撃者は、Web アプリケーションに対する被害者の信頼を利用し、攻撃の媒体として Web アプリケーションを使用しています。データを適切に検証することなく動的 Web ページを表示する多くのアプリケーションは攻撃されやすいといえます。Web サイトに対する攻撃は、あるユーザからの入力が別のユーザに表示されることを目的としている場合は特に単純です。可能性として、掲示板、ユーザコメントスタイルの Web サイト、ニュース、またはメールアーカイブなどがあります。

たとえば、次のスクリプトがスクリプトコンポーネント、行動、または Visualforce ページを使用する Force.com ページに使用されているとします。

このスクリプトブロックは、ユーザが入力した  
に次の値を入力することができます。

の値をページに挿入します。これで攻撃者は

この場合、現在のページのすべての Cookie が  
に送信されます。この時点で、攻撃者は被害者のセッション Cookie を持つてあり、彼らが  
被害者になりますまで Web アプリケーションに接続することができます。

攻撃者は、Web サイトまたはメールを使用して、悪意のあるスクリプトを送信できます。Web アプリケーションユーザにより攻撃者の入力が表示されるだけでなく、ブラウザによって信頼されたコンテキストで攻撃者のスクリプトを実行することもできます。こうした機能により、攻撃者はさまざまな攻撃を被害者に対して行うことができます。攻撃の範囲はウィンドウを開いたり閉じたりする単純なアクションから、データまたはセッションの Cookie を盗むなど、被害者のセッションに攻撃者が完全にアクセスできるようになる悪意に満ちた攻撃にまでわたります。

こうした攻撃についての一般的な詳細は、次の記事を参照してください。

- [http://www.owasp.org/index.php/Cross\\_Site\\_Scripting](http://www.owasp.org/index.php/Cross_Site_Scripting)
- <http://www.cgisecurity.com/articles/xss-faq.shtml>
- [http://www.owasp.org/index.php/Testing\\_for\\_Cross\\_site\\_scripting](http://www.owasp.org/index.php/Testing_for_Cross_site_scripting)
- <http://www.google.com/search?q=cross-site+scripting>

Force.com プラットフォーム内では、複数の対 XSS 防御策が実行されています。たとえば、多くの出力方法の有害な特性を除外するフィルタが実装されています。標準クラスおよび出力方法を使用する開発者に対する XSS の脆弱性の脅威は、大幅に緩和されています。ただし、クリエイティブな開発者によって、デフォルトのコントロールをわざとまたは偶然エスケープする方法がいまだに見つかっています。次のセクションでは、保護されている場所、保護されていない場所について説明しています。

## 既存の保護

で始まるすべての標準 Visualforce コンポーネントでは、対 XSS フィルタが設定されています。たとえば、ユーザに直接返されるユーザ指定の入力および出力を採用するため、次のコードは通常 XSS の攻撃に対して脆弱ですが、  
タグは XSS に対して安全です。HTML タグとされるすべての文字は、リテラル形式に変換されます。たとえば、< 文字は < に変換され、ユーザの画面上ではリテラル < が表示されます。

## Visualforce タグのエスケープの無効化

デフォルトでは、ほぼすべての Visualforce タグは XSS に対して脆弱な文字をエスケープします。省略可能な属性 を設定することによって、この動作を無効化することができます。たとえば、次の出力は、XSS の攻撃に対して脆弱です。

## XSS から保護されていないプログラミング項目

次の項目には XSS 保護を組み込んでいないため、これらのタグおよびオブジェクトを使用する場合は特別な保護を行なう必要があります。これは、これらの項目により、開発者がスクリプトコマンドを挿入してページをカスタマイズできるようになっているためです。意図的にページに追加されるコマンドに対 XSS フィルタを指定しても意味はありません。

### カスタム JavaScript

独自の JavaScript を作成した場合、Force.com プラットフォームにはユーザを保護する方法がありません。たとえば JavaScript で使用している場合、次のコードは XSS の攻撃に対して脆弱です。

```
<apex:includeScript>
```

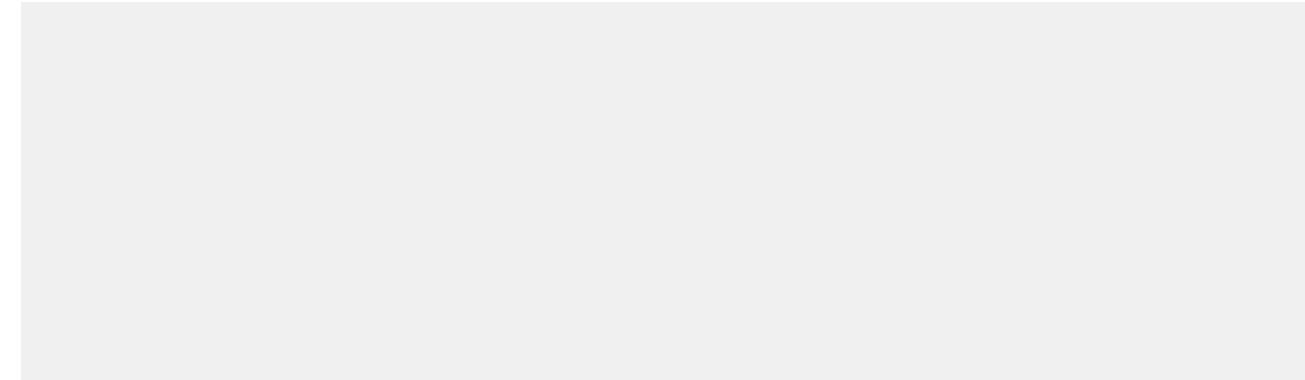
Visualforce コンポーネントを使用して、ページにカスタムスクリプトを追加できます。こうした場合、内容が安全で、ユーザが提供したデータが含まれていないことを慎重に確認してください。たとえば、次のスニペットはスクリプトの値としてユーザ提供の入力が含まれているため、特に脆弱です。タグによって指定された値は、使用する JavaScript への URL です。攻撃者がパラメータに任意のデータを入力できる場合(下記の例参照)、被害者に別の Web サイトの JavaScript ファイルを使用するよう指示することができる可能性があります。

## Visualforce ページのエスケープされない出力と式

属性を `false` に設定するコンポーネントを使用する場合、または Visualforce コンポーネント外の式を含める場合は、出力がフィルタ処理されないため、セキュリティの検証が必要です。これは、数式を使用する場合は特に重要です。

数式は関数コールとして使用したり、プラットフォームオブジェクト、ユーザの環境、システム環境、要求の環境に関する情報を含めることができます。式が生成する出力が表示されるときにエスケープされないことを認識することが重要です。数式はサーバに表示されるため、JavaScript またはその他のクライアント側の技術を使用してクライアントの表示データをエスケープすることはできません。これにより、数式が非システムデータ(悪意のあるまたは編集可能なデータ)を参照し、式自体が関数にラップされていない場合、表示中に出力をエスケープするという危険な状況を誘発する場合があります。

一般的な脆弱性は、ユーザ入力をページに表示する場合に発生します。次に例を示します。



エスケープされない  
ば、

によっても、クロスサイトスクリプトの脆弱性が誘発されます。たとえ

を入力し、[更新] をクリックすると、JavaScript が実行されます。この場合、アラートダイアログが表示されますが、悪意のある使用が設定されている場合があります。

安全でないと考えられる文字列をエスケープするために使用できる関数があります。

#### HTMLENCODE

HTMLENCODE 関数は、大なり記号 (>) などの HTML で予約されている文字を、< などの HTML エンティティ文字に置き換え、HTML で使用するテキスト文字列や差し込み項目値を符号化します。

#### JSENCODE

新しいJSENCODE 関数は、バックスラッシュ (\) のようなエスケープ文字を、アポストロフィー ('') のような安全でない JavaScript 文字の前に挿入して、JavaScript で使用するテキスト文字列や差し込み項目値を符号化します。

#### JSINHTMLENCODE

新しいJSINHTMLENCODE 関数は、エスケープ文字を、安全でない JavaScript 文字の前に挿入し、HTML で予約されている文字を HTML エンティティ文字に置き換えて、HTML タグ内の JavaScript で使用するテキスト文字列や差し込み項目値を符号化します。

#### URLENCODE

新しいURLENCODE 関数は、空白などの URL では不正な文字を *RFC 3986, Uniform Resource Identifier (URI): Generic Syntax* で定義されているようにこうした文字を表すコードに置き換え、URL で使用するテキスト文字列や差し込み項目を符号化します。たとえば、感嘆符は ! に置き換えられます。

前述の例を保護するために

を使用するには、

を次のように変更します。

ユーザが を入力し、[更新] をクリックしても、JavaScript は実行されません。代わりに  
文字列が符号化され、ページには と表示されます。

タグの代入およびデータの使用によって、エスケープされた文字およびエスケープが必要な文字が異なります。たとえば、次のような文の場合

リンクで使用されるため、URL では HTML エスケープ文字の の代わりに を使用して二重引用符をエスケープする必要があります。そうでない場合、次のような要求

では、次のようにになります。

JavaScript が実行し、アラートが表示されます。

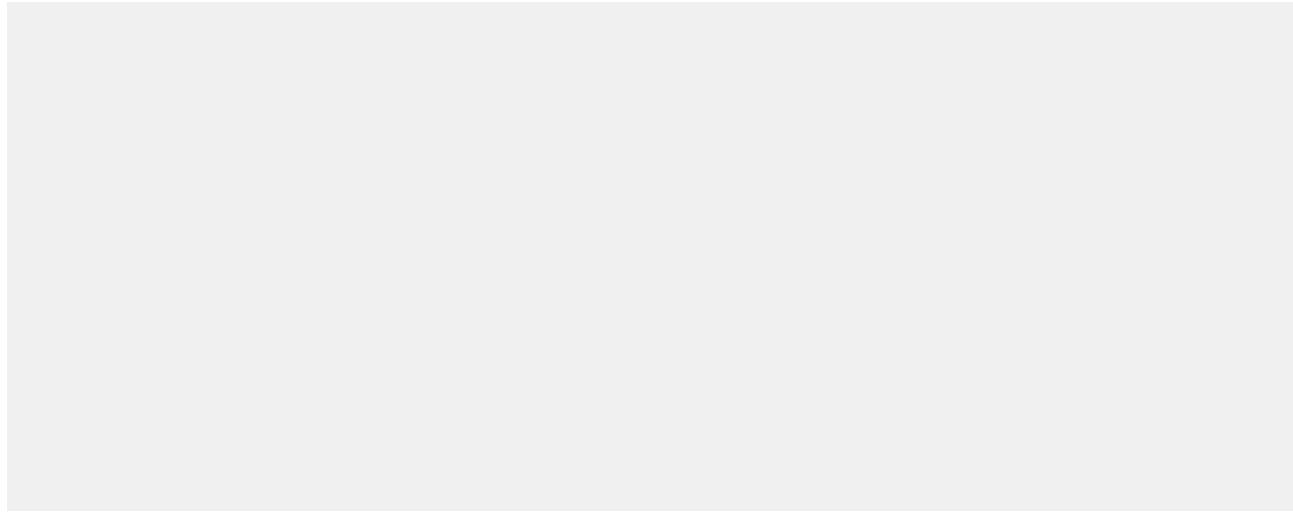
この場合、JavaScript が実行されないようにするために、 関数を使用します。例

また、数式タグを使用して、プラットフォームオブジェクトデータを追加することもできます。データがユーザの組織から直接取得されますが、データをエスケープしてユーザが他のユーザ(権限レベルがより高いユーザ)のコンテキストでコードを実行できなくなります。これらの種類の攻撃は同じ組織内のユーザによって実行され、

- <http://shiflett.org/articles/cross-site-request-forgeries>

Force.com プラットフォーム内では、この攻撃を回避する対 CSRF トークンが実装されています。すべてのページにランダムな文字列が非表示項目として含まれています。次のページが読み込まれると、アプリケーションはこの文字列の正当性を確認し、値が予測値と一致しない限り、コマンドを実行しません。この機能によって、すべての標準コントローラおよびメソッドの使用時に攻撃から保護されます。

開発者は、リスクを意識せずに組み込み防御策をスキップしてしまう場合があります。たとえば、オブジェクト ID を入力パラメータとして SOQL コールで使用するカスタムコントローラがあるとします。次のコードスニペットについて考えます。



この場合、開発者は独自の action メソッドを作成して、意識せずに CSRF 対策コントロールをスキップしています。パラメータはコードで読み込まれ、使用されます。CSRF 対策トークンが読み込まれたり、検証されたりすることはありません。攻撃者の Web ページでは、CSRF 攻撃を使用してユーザをこのページに移動させ、パラメータとして攻撃者が望む値を指定する可能性があります。

このような状況に対する組み込み防御策がないため、開発者は前例の変数のようなユーザ指定のパラメータに基づいてアクションを実行するページの書き込みに対し、注意する必要があります。回避策の 1 つは、アクションを実行する前に中間の確認ページを挿入して、ユーザが本当にそのページをコールしようとしているのかどうかを確認することです。その他の対策として、組織のアイドルセッションのタイムアウトを短くすること、ユーザがあるサイトで認証されたままブラウザを使用して別のサイトに移動しないように、アクティブなセッションからログアウトすることを推奨すること、などが考えられます。

## SOQL インジェクション

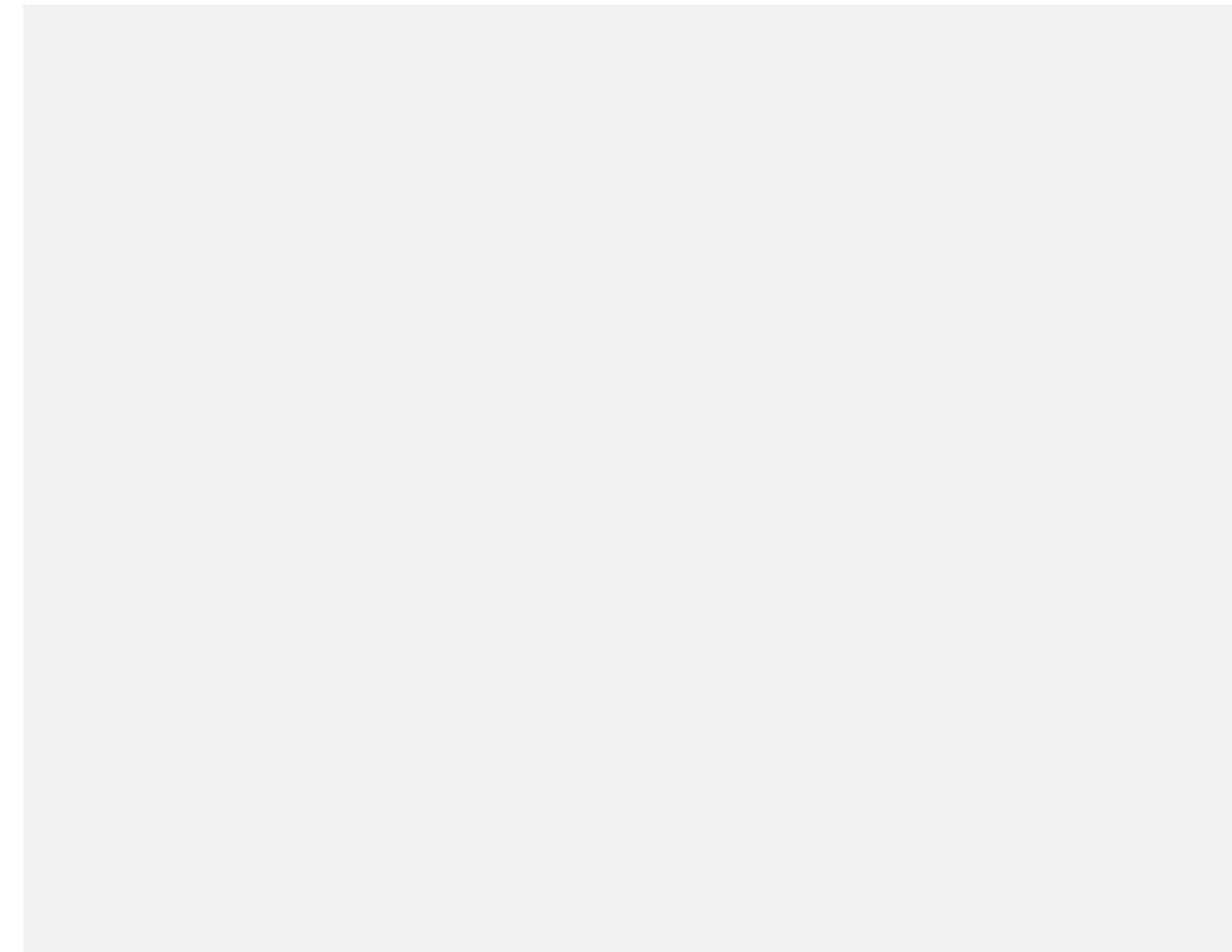
他のプログラミング言語では、上記の弱点を SQL インジェクションといいます。Apex では SQL を使用しませんが、独自のデータベースクエリ言語 SOQL を使用します。SOQL は、SQL より単純で、機能が制限されています。そのため、SOQL インジェクションのリスクは SQL と比較して大幅に低くなりますが、攻撃は従来の SQL インジェクションとほぼ同じです。集計時は、SQL/SOQL インジェクションではユーザが提供した入力を取得し、これらの値を動的 SOQL クエリに使用します。入力が検証されない場合、SOQL ステートメントを事实上変更する SOQL コマンドを指定し、アプリケーションにトリックを仕掛けて意図しないコマンドを実行するようになります。

SQL インジェクション攻撃の詳細は、以下を参照してください。

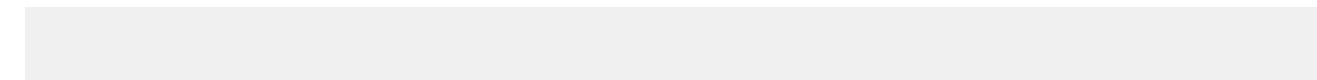
- [http://www.owasp.org/index.php/SQL\\_injection](http://www.owasp.org/index.php/SQL_injection)
- [http://www.owasp.org/index.php/Blind\\_SQL\\_Injection](http://www.owasp.org/index.php/Blind_SQL_Injection)
- [http://www.owasp.org/index.php/Guide\\_to\\_SQL\\_Injection](http://www.owasp.org/index.php/Guide_to_SQL_Injection)
- <http://www.google.com/search?q=sql+injection>

### Apex での SOQL インジェクションの脆弱性

以下に SOQL に対して脆弱な Apex コードおよび Visualforce の単純な例を示します。



これは単純な例ですが、ロジックについて説明しています。コードは、削除されていない取引先責任者の検索を行うためのものです。ユーザは \_\_\_\_\_ という入力値を指定します。値はユーザが指定する任意の値で、検証されません。SOQL クエリは動的に構築され、\_\_\_\_\_ メソッドで実行されます。ユーザが正当な値を指定すると、ステートメントは次のように期待どおり実行されます。



ただし、次のようにユーザが予期しない値を入力したかのようになります。

この場合、クエリ文字列は次のようになります。

結果には削除されていない取引先責任者だけでなく、すべての取引先責任者が表示されます。SOQL インジェクションにより、脆弱なクエリの対象となるロジックを変更することができます。

### SOQL インジェクションの防御策

SOQL インジェクションの攻撃を回避するには、動的 SOQL クエリを使用しないようにします。代わりに、静的クエリとバインド変数を使用します。上記の脆弱な例は、静的 SOQL を使用して次のように書き直すことができます。

動的 SOQL を使用する必要がある場合、

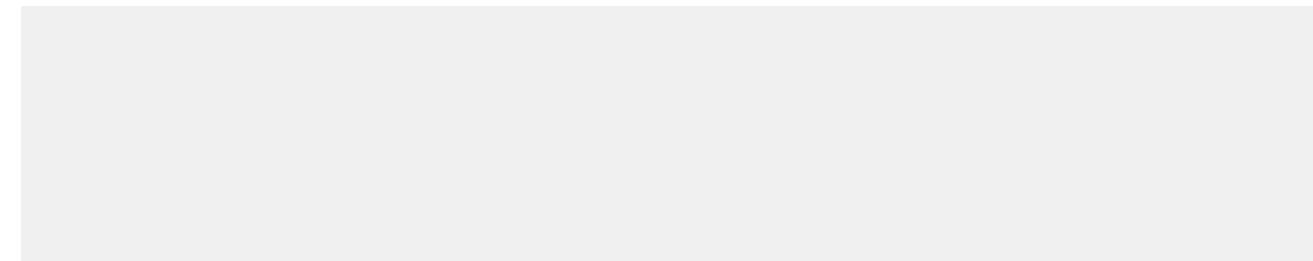
します。このメソッドは、ユーザから渡される文字列のすべての単一引用符にエスケープ文字(\)を追加します。このメソッドにより、すべての単一引用符を、データベースコマンドではなく、囲まれた文字列として処理します。

メソッドを使用して、ユーザ指定の入力を削除

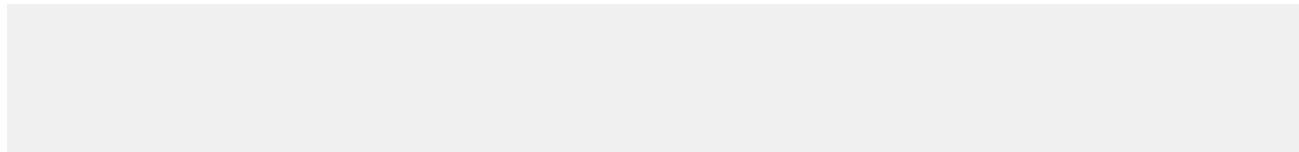
## データアクセスコントロール

Force.com プラットフォームは、データ共有ルールを広範囲に使用します。各オブジェクトには権限があり、ユーザが読み取り、作成、編集、削除できる共有設定がある場合があります。これらの設定は、すべての標準コントローラを使用する場合に強制されます。

Apex クラスを使用する場合、組み込みユーザ権限、および項目レベルのセキュリティ制限は実行時に重視されません。デフォルトの動作として、Apex クラスに組織内のすべてのデータを読み込み更新する機能があります。これらのルールは強制されないため、Apex を使用する開発者は、ユーザ権限、項目レベルのセキュリティ、または組織のデフォルト設定によって通常は非表示となる機密データが不注意で公開されないようにする必要があります。これは特に、Visualforce ページで当てはまります。たとえば、次の Apex 擬似コードについて考えます。



この場合、現在ログインしているユーザにこれらのレコードを表示する権限がない場合でも、すべての取引先責任者レコードが検索されます。解決策として、クラスを宣言する場合、次のように修飾キーワードのを使用します。



キーワードを使用すると、プラットフォームはすべてのレコードに完全アクセス権限を付与するのではなく、現在ログインしているユーザのセキュリティ共有権限を使用します。



# 付録 D

## Apex の SOAP API および SOAP ヘッダー

この付録では、Apex でデフォルトで使用できる SOAP API コールおよびオブジェクトの詳細について説明します。



メモ: Apex クラスメソッドは、カスタムの SOAP Web サービスコールとして公開できます。これにより、外部アプリケーションが Apex Web サービスを呼び出して、Salesforce のアクションを実行できます。これらのメソッドの定義には [キーワードを使用します](#)。詳細は、「[キーワードの使用に関する考慮事項](#)」(ページ 355)を参照してください。

SOAP API コールを使用して保存されたすべての Apex コードは、要求のエンドポイントと同じバージョンの SOAP API を使用します。たとえば、SOAP API バージョン 28.0 を使用する場合、次のようにエンドポイント 28.0 を使用します。

既存の Apex IDE の拡張または実装に使用できる SOAP API コールを含むその他のすべての SOAP API コールについての詳細は、salesforce.com の担当者までお問い合わせください。

次の SOAP API コールがあります。

- 
- 
- 
- 
- 

次の SOAP ヘッダーを Apex の SOAP API コールで使用できます。

- [DebuggingHeader](#)
- [PackageVersionHeader](#)

また、次の 2 つのコールについては、『*Metadata API 開発者ガイド*』を参照してください。

- 
-

## compileAndTest()

単一のコールで Apex をコンパイルおよびテストします。

### 構文

#### 使用方法

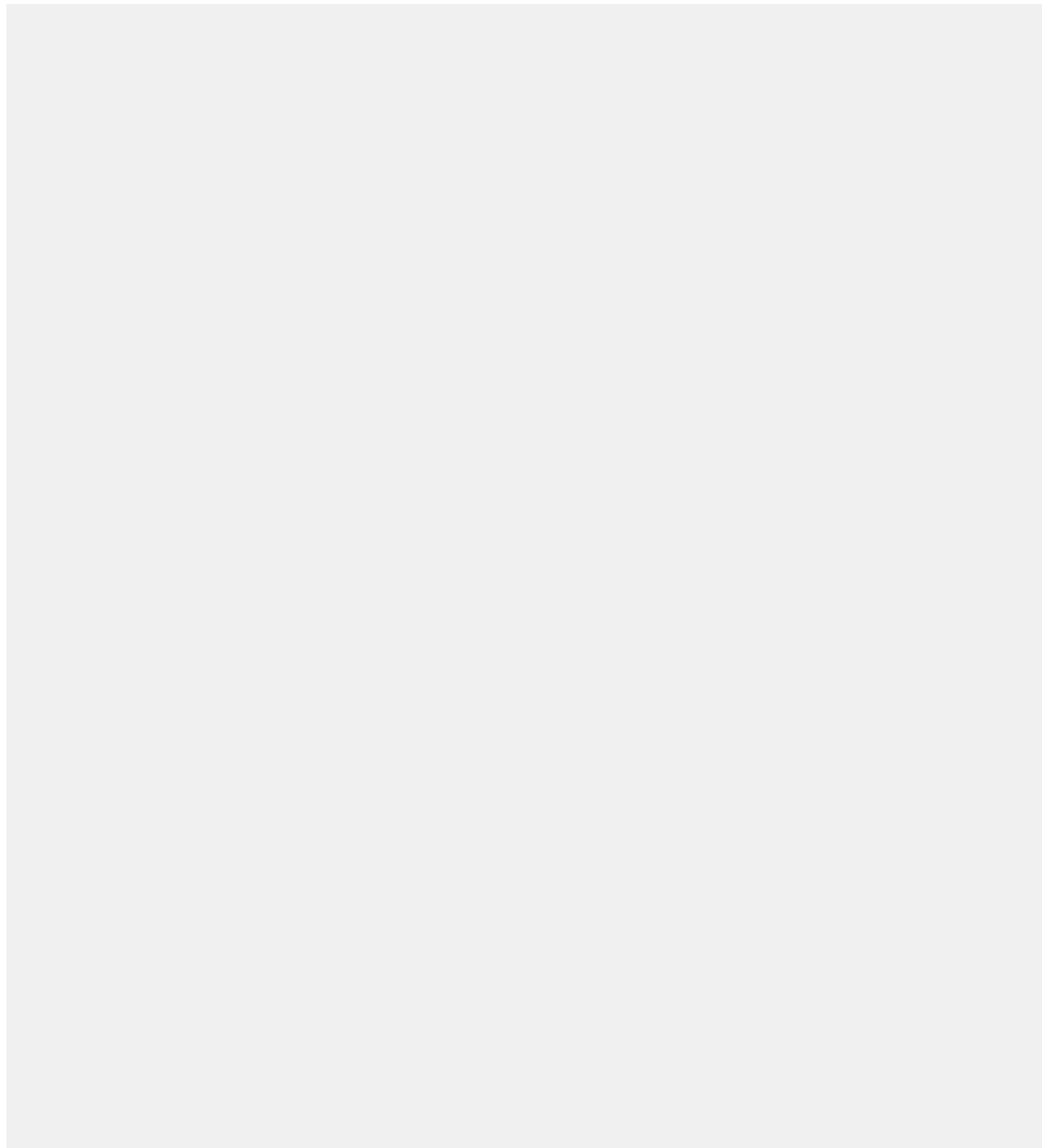
このコールを使用して、1つのコールで指定した Apex にコンパイルとテストの両方を実行します。本番組織 (Developer Edition または Sandbox Edition ではない) は、または の代わりにこのコールを使用する必要があります。

このコールは、DebuggingHeader と SessionHeader をサポートしています。API の SOAP ヘッダーの詳細は、『[SOAP API 開発者ガイド](#)』を参照してください。

指定されたすべてのテストに合格する必要があります。合格しない場合、データはデータベースに保存されません。このコールが本番組織で呼び出されると、CompileAndTestRequest の RunTestsRequest プロパティは無視され、組織で定義されたすべての単体テストが実行されます。これらのテストに合格する必要があります。

#### サンプルコード—Java

次の例では、を に設定して、このクラスのコンパイルおよびテストを実行するが、クラスがデータベースに保存されないようにします。



## 引数

名前	型	説明
	CompileAndTestRequest	Apex およびこの要求に設定する必要のある項目の値を含む要求。

## 応答

### [CompileAndTestResult](#)

## CompileAndTestRequest

コールにはこのオブジェクト (コンパイル対象の Apex に関する情報をもつ要求) が含まれます。

CompileAndTestRequest オブジェクトには、次のプロパティがあります。

名前	型	説明
	boolean	に設定されている場合、コードが正常にコンパイルされているかどうか、単体テストに合格しているかどうかに関係なく、送信された Apex クラスおよびトリガは組織に保存されません。
	string	コンパイルされるクラスの内容。
	string	削除されるクラスの名前。
	string	削除されるトリガの名前。
	RunTestsRequest	テストする Apex の情報を指定します。要求が本番組織に送信されると、このプロパティは無視され、組織全体ですべての単体テストが実行されます。
	string	コンパイルされるトリガの内容。

このオブジェクトについて、次の点に注意してください。

- このオブジェクトには、プロパティが含まれています。要求が本番組織で実行されると、このプロパティは無視されすべてのテストが実行されます。
- コンパイル、削除、テスト時にエラーが発生した場合、または75%のコードカバー率の目標が達成されなかった場合、クラスもトリガは組織に保存されません。これは、Force.com AppExchange パッケージテストと同じ要件です。
- すべてのトリガには、コードカバー率が設定されている必要があります。トリガにコードカバー率がない場合、クラスもトリガも組織には保存されません。

## CompileAndTestResult

コールは、成功または失敗など、指定された Apex のコンパイルおよび単体テストの実行に関する情報を返します。

CompileAndTestResult オブジェクトには、次のプロパティがあります。

名前	型	説明	
	<a href="#">CompileClassResult</a>	クラスがコンパイルされていた場合、コールの成功または失敗の情報。	
	<a href="#">DeleteApexResult</a>	クラスが削除されていた場合、成功または失敗の情報。	コールの
	<a href="#">DeleteApexResult</a>	トリガが削除されていた場合、成功または失敗の情報。	コールの
	<a href="#">RunTestsResult</a>	単体テストが指定された場合、Apex 単体テストの成功または失敗の情報。	
	<a href="#">boolean*</a>	の場合、指定されたすべてのクラス、トリガ、単体テストが正常に実行されています。クラス、トリガまたは単体テストが失敗した場合、値は で、詳細は次のような対応する結果オブジェクトで報告されます。 <ul style="list-style-type: none"><li>• <a href="#">CompileClassResult</a></li><li>• <a href="#">CompileTriggerResult</a></li><li>• <a href="#">DeleteApexResult</a></li><li>• <a href="#">RunTestsResult</a></li></ul>	
	<a href="#">CompileTriggerResult</a>	トリガがコンパイルされていた場合、コールの成功または失敗の情報。	

\* リンクから『SOAP API Developer's Guide』にアクセスできます。

## CompileClassResult

このオブジェクトは、または コールの一部として返されます。指定された Apex のコンパイルと実行が正常に行われたかどうかの情報が含まれています。

CompileClassResult オブジェクトには、次のプロパティがあります。

名前	型	説明
	int*	クラスファイルまたはトリガファイルの CRC(周期的冗長チェック)。
	int*	エラーが発生した場合、発生した列の番号。
	ID*	コンパイルされた各クラスの ID が作成されます。ID は組織内で一意です。
	int*	エラーが発生した場合、発生した行の番号。
	string*	クラスの名前です。
	string*	エラーが発生した場合、その問題の説明。

名前	型	説明
	boolean*	の場合、クラスは正常にコンパイルされています。 の場合、問題はこのオブジェクトのその他のプロパティで指定されています。

\* リンクから『SOAP API Developer's Guide』にアクセスできます。

### CompileTriggerResult

このオブジェクトは、または コールの一部として返されます。指定された Apex のコンパイルと実行が正常に行われたかどうかの情報が含まれています。

CompileTriggerResult オブジェクトには、次のプロパティがあります。

名前	型	説明
	int*	トリガファイルの CRC (周期的冗長検査)。
	int*	エラーが発生した場合、発生した列。
	ID*	コンパイルされた各トリガの ID が作成されます。ID は組織内で一意です。
	int*	エラーが発生した場合、発生した行の番号。
	string*	トリガの名前。
	string*	エラーが発生した場合、その問題の説明。
	boolean*	の場合、指定されたトリガは正常にコンパイルされ、実行されています。トリガのコンパイルまたは実行が失敗した場合、値はです。

\* リンクから『SOAP API Developer's Guide』にアクセスできます。

### DeleteApexResult

このオブジェクトは、 コールがクラスまたはトリガの削除に関する情報を返すときに、返されます。

DeleteApexResult オブジェクトには、次のプロパティがあります。

名前	型	説明
	ID*	削除されたトリガまたはクラスの ID。ID は組織内で一意です。
	string*	エラーが発生した場合、その問題の説明。

名前	型	説明
	boolean*	の場合、指定されたクラスまたはトリガはすべて正常に削除されています。クラスまたはトリガが削除されていない場合、値はです。

\* リンクから『SOAP API Developer's Guide』にアクセスできます。

## compileClasses()

Developer Edition または Sandbox を使用している組織の Apex をコンパイルします。

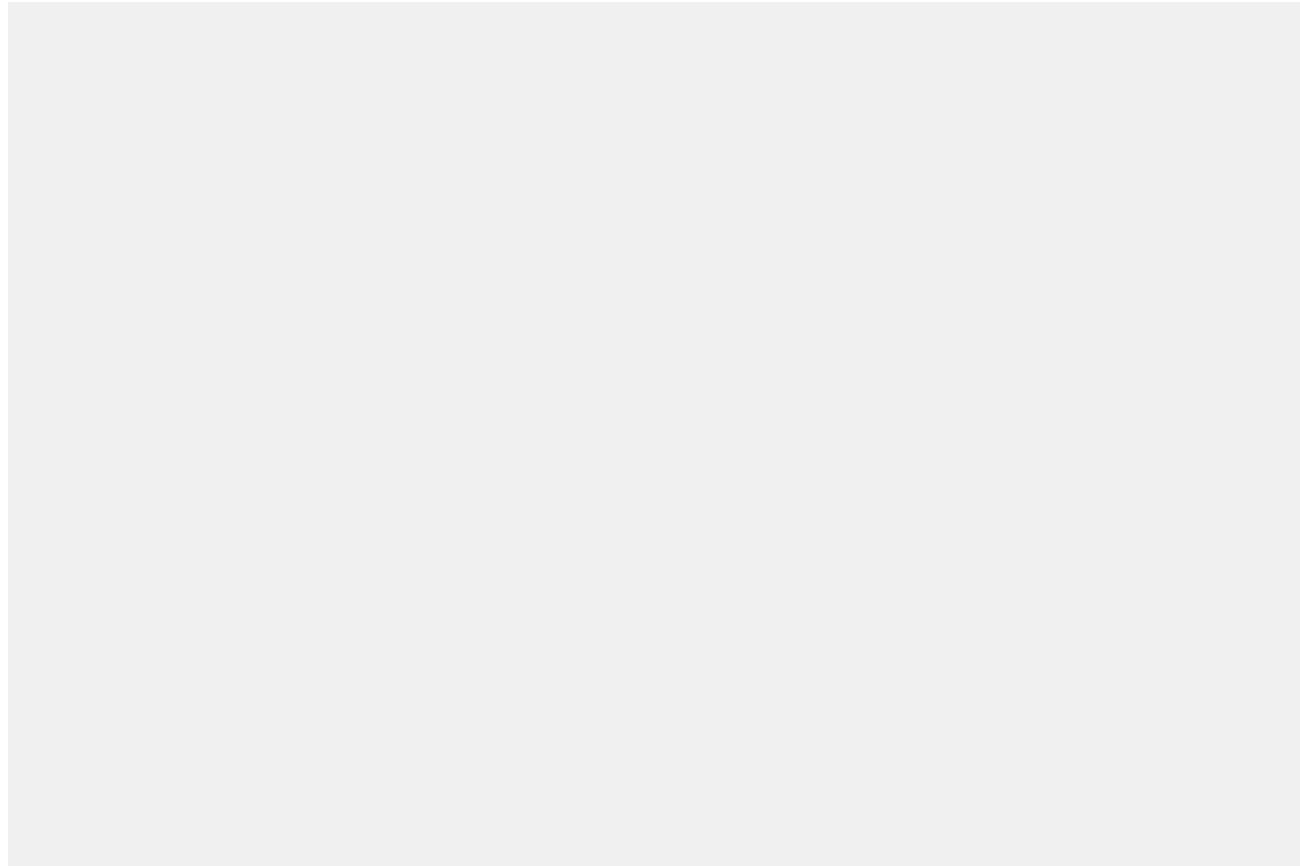
### 構文

#### 使用方法

このコールを使用して、Developer Edition または Sandbox を使用している組織の Apex クラスをコンパイルします。本番組織では、  
を使用する必要があります。

このコールは、DebuggingHeader と SessionHeader をサポートしています。API の SOAP ヘッダーの詳細は、  
『SOAP API 開発者ガイド』を参照してください。

#### サンプルコード—Java



## 引数

名前	型	説明
	string*	Apex クラスおよびこの要求に設定する必要のある項目の値を含む要求。

\* リンクから [『SOAP API Developer's Guide』](#) にアクセスできます。

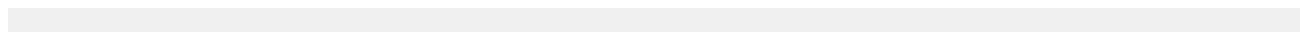
## 応答

[CompileClassResult](#)

## compileTriggers()

Developer Edition または Sandbox を使用している組織の Apex トリガをコンパイルします。

## 構文



## 使用方法

このコールを使用して、Developer Edition または Sandbox を使用している組織の指定された Apex トリガをコンパイルします。本番組織では、  
を使用する必要があります。

このコールは、DebuggingHeader と SessionHeader をサポートしています。API の SOAP ヘッダーの詳細は、  
[『SOAP API 開発者ガイド』](#) を参照してください。

## 引数

名前	型	説明
	string*	Apex トリガおよびこの要求に設定する必要のある項目の値を含む要求。

\* リンクから [『SOAP API Developer's Guide』](#) にアクセスできます。

## 応答

[CompileTriggerResult](#)

## executeanonymous ()

Apex のブロックを実行します。

## 構文

## 使用方法

このコールを使用して、Apex の匿名ブロックを実行します。このコールは AJAX から実行できます。

このコールは、API DebuggingHeader と SessionHeader をサポートしています。

制限された API アクセスを含むパッケージのコンポーネントがこのコールを発行する場合、要求はブロックされます。

項目に割り当てた文字列値が長すぎる場合、API バージョン 15.0 以降を使用して保存 (コンパイル) した Apex クラスとトリガにはランタイムエラーが発生します。

## 引数

名前	型	説明
	string*	Apex のブロック。

\* リンクから [『SOAP API Developer's Guide』](#) にアクセスできます。

[SOAP API Developer's Guide](#) には、セキュリティ、アクセス、SOAP ヘッダーに関する情報が記載されています。

## 応答

[ExecuteAnonymousResult\[\]](#)

### ExecuteAnonymousResult

コールは、コードのコンパイルと実行が正常に行われたかどうかの情報を返します。

オブジェクトには次のプロパティがあります。

名前	型	説明
	<code>int*</code>	が <code>false</code> である場合、この項目にはコンパイルが失敗したポイントの列番号が含まれています。
	<code>string*</code>	が <code>false</code> である場合、この項目にはコンパイルの失敗を引き起こした問題の説明が含まれています。
	<code>boolean*</code>	<code>true</code> の場合、コードは正常にコンパイルされています。 <code>false</code> の場合、項目は <code>null</code> ではありません。
	<code>string*</code>	が <code>false</code> である場合、この項目には失敗の例外メッセージが含まれています。
	<code>string*</code>	が <code>false</code> である場合、この項目には失敗のスタック追跡が含まれています。
	<code>int*</code>	が <code>false</code> である場合、この項目にはコンパイルが失敗したポイントの行番号が含まれています。
	<code>boolean*</code>	<code>true</code> の場合、コードは正常に実行されています。 <code>false</code> の場合、および の値は <code>null</code> ではありません。

\* リンクから [『SOAP API Developer's Guide』](#) にアクセスできます。

### runTests()

Apex 単体テストを実行します。

## 構文

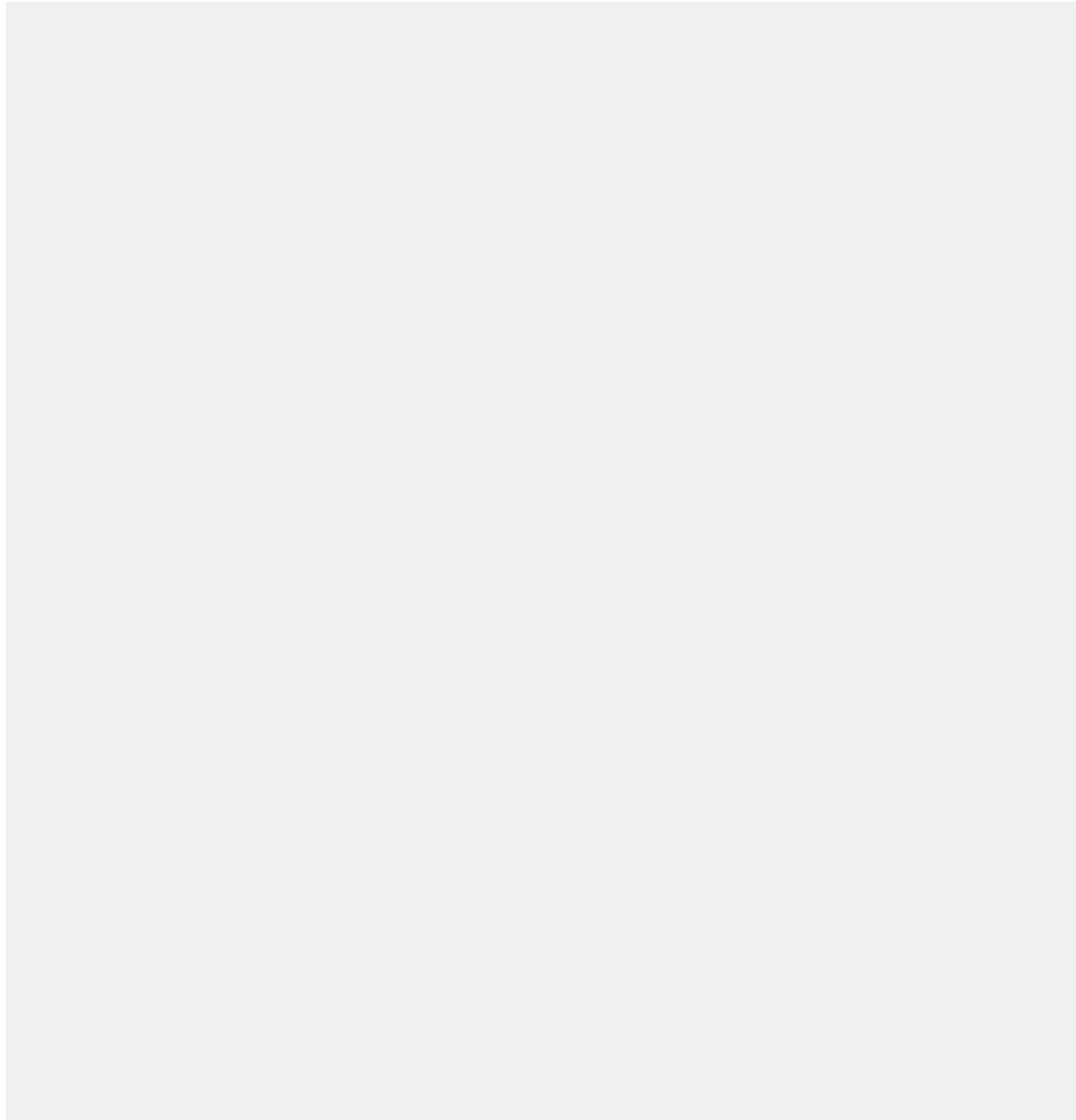
### 使用方法

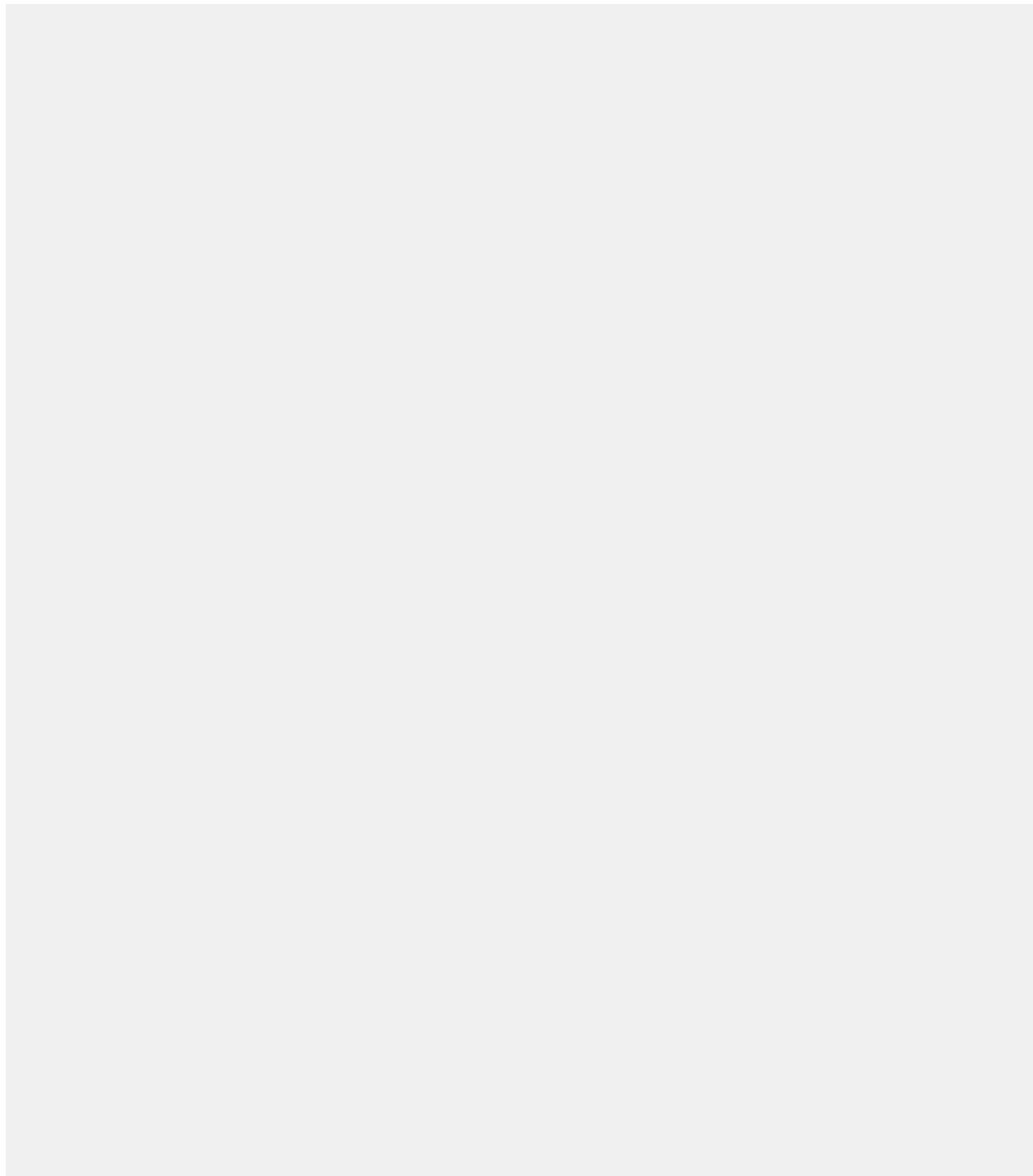
堅牢で、エラーのないコードの開発を促進するため、Apex は単体テストの作成と実行をサポートします。単体テストは、コード内の特定の部分が正しく機能していることを確認するクラスメソッドです。単体テストのメ

ソッドは引数を取らず、データベースへのデータの確定やメールの送信を行うこともなく、メソッド定義にキーワードまたはアノテーションでフラグが付けられています。また、テストメソッドは、テストクラス（アノテーションが付加されているクラス）で定義されている必要があります。このコールを使用して、Apex 単体テストを実行します。

このコールは、DebuggingHeader と SessionHeader をサポートしています。API の SOAP ヘッダーの詳細は、『[SOAP API 開発者ガイド](#)』を参照してください。

### サンプルコード—Java





## 引数

名前	型	説明
	RunTestsRequest	Apex 単体テストおよびこの要求に設定する必要のある項目の値を含む要求。

## 応答

RunTestsResult

## RunTestsRequest

コールには、コンパイルされる Apex に関する情報を伴う [CompileAndTestRequest](#) の要求が含まれます。要求には、テストする Apex の情報を指定するこのオブジェクトも含まれています。テストする同じクラスまたは異なるクラスをコンパイルされた状態に指定します。トリガを直接テストできないため、このオブジェクトに含めることはできません。代わりに、トリガをコールするクラスを指定する必要があります。

要求が本番組織に送信されると、この要求は無視され、組織に定義されたすべての単体テストが実行されます。 [CompileAndTestRequest](#) オブジェクトには、次のプロパティがあります。

名前	型	説明
	boolean*	が true の場合、組織に定義されたすべての単体テストが実行されます。
	string*	1 つ以上のオブジェクトの配列。
	string	指定されている場合、実行する単体テストを含む名前空間。 <code>allTests</code> を <code>true</code> に指定する場合、このプロパティを使用しないでください。また、本番組織で を実行する場合、このプロパティは無視され、組織に定義されたすべての単体テストが実行されます。
	string*	バージョン 10.0 以降は使用しないでください。サポートされていない古いリリースでは、パッケージの内容がテストされます。

\* リンクから [『SOAP API Developer's Guide』](#) にアクセスできます。

## RunTestsResult

コールは、指定された Apex のコンパイルが成功したか否か、単体テストが正常に完了したか否かについての情報を返します。

RunTestsResult オブジェクトには、次のプロパティがあります。

名前	型	説明
	CodeCoverageResult[]	単体テストのコードカバー率の詳細を含む 1 つ以上の CodeCoverageResult オブジェクトの配列。
	CodeCoverageWarning[]	テストの実行について警告する 1 つ以上のコード範囲の配列。結果には、実行された行の合計数、実行されなかったコードの数、行、列の位置が含まれています。
	RunTestFailure[]	単体テストの失敗があれば、それについての情報を含む 1 つ以上の RunTestFailure オブジェクトの配列。
	int	単体テストの失敗数。
	int	実行された単体テストの数。
	RunTestSuccess[]	成功についての情報を含む 1 つ以上の RunTestSuccesses オブジェクトの配列。
	double	テストの実行に費やした累積時間の合計。パフォーマンスの監視に役立つ場合があります。

## CodeCoverageResult

このオブジェクトを含む [RunTestsResult](#) オブジェクト。指定された Apex のコンパイルと単体テストの実行が正常に行われたかどうかの情報を含まれています。

CodeCoverageResult オブジェクトには、次のプロパティがあります。

名前	型	説明
	CodeLocation[]	このプロパティには、テストされた各クラスまたはトリガについて、また、テストされたコードの各部分について、DML ステートメントの場所、コードが実行された回数、これらのコールに費やした累積時間の合計が含まれています。パフォーマンスの監視に役立つ場合があります。
	ID	CodeLocation の ID。ID は組織内で一意です。
	CodeLocation[]	テストされた各クラスまたはトリガについて、コードが一切カバーされていない場合、テストされていないコードの行および列、コードが実行された回数。

名前	型	説明
	CodeLocation[]	テストされた各クラスまたはトリガについて、メソッド呼び出しの場所、コードが実行された回数、これらのコールに費やした累積時間の合計。パフォーマンスの監視に役立つ場合があります。
	string	カバーされているクラスまたはトリガの名前。
	string	指定されている場合、単体テストを含む名前空間。
	int	コードの場所の合計数。
	CodeLocation[]	テストされた各クラスまたはトリガについて、コードの SOQL ステートメントの場所、コードが実行された回数、これらのコールに費やした累積時間の合計。パフォーマンスの監視に役立つ場合があります。
	CodeLocation[]	テストされた各クラスについて、コードの SOSL ステートメントの場所、コードが実行された回数、これらのコールに費やした累積時間の合計。パフォーマンスの監視に役立つ場合があります。
	string	使用しません。以前のサポートされていないリリースでは、クラスまたはパッケージを指定していました。

## CodeCoverageWarning

このオブジェクトを含む [RunTestsResult](#) オブジェクト。警告を生成した Apex クラスに関する情報が含まれています。

このオブジェクトには次のプロパティがあります。

名前	型	説明
	ID	警告を生成したクラスの ID。
	string	生成された警告のメッセージ。
	string	警告を生成したクラスの名前。警告がコードカバー率全体に適用された場合、この値は null になります。
	string	指定されている場合、クラスを含む名前空間。

## RunTestFailure

[RunTestsResult](#) オブジェクトは、単体テスト実行時の失敗に関する情報を返します。

このオブジェクトには次のプロパティがあります。

名前	型	説明
	ID	失敗を生成したクラスの ID。
	string	失敗のメッセージ。
	string	失敗したメソッドの名前。
	string	失敗したクラスの名前。
	string	指定されている場合、クラスを含む名前空間。
	string	失敗についてのスタック追跡。
	double	失敗した処理についてテストの実行に費やした時間。パフォーマンスの監視に役立つ場合があります。
	string	使用しません。以前のサポートされていないリリースでは、クラスまたはパッケージを指定していました。

\* リンクから『SOAP API Developer's Guide』にアクセスできます。

## RunTestSuccess

[RunTestsResult](#) オブジェクトは、単体テスト実行時の成功に関する情報を返します。

このオブジェクトには次のプロパティがあります。

名前	型	説明
	ID	成功を生成したクラスの ID。
	string	成功したメソッドの名前。
	string	成功したクラスの名前。
	string	指定されている場合、クラスを含む名前空間。

名前	型	説明
	double	この操作についてテストの実行に費やした時間。パフォーマンスの監視に役立つ場合があります。

## CodeLocation

[RunTestsResult](#) オブジェクトは、多数の項目にこのオブジェクトを含みます。

このオブジェクトには次のプロパティがあります。

名前	型	説明
	int	テストされた Apex の列の場所。
	int	テストされた Apex の行の場所。
	int	テスト実行時に Apex が実行された回数。
	double	この場所で費やした累積時間の合計。パフォーマンスの監視に役立つ場合があります。

## DebuggingHeader

応答がリターンヘッダーのデバッグログを含むよう指定し、デバッグのヘッダーの詳細のレベルを指定します。

### API コール

#### 項目

要素名	型	説明
	logtype	この項目は廃止され、後方互換性にのみ提供されています。デバッグログに返される情報の種類を指定します。値は、返される情報が最も少ないものから最も多くの順に表示されます。使用できる値は次のとおりです。 • • •

要素名	型	説明
LogInfo[]		デバッグログに返される情報の量や種類を指定します。

## LogInfo

デバッグログに返される情報の量や種類を指定します。

項目は、これらのオブジェクトのリストを取ります。

### 項目

要素名	型	説明
		<p>デバッグログに返される情報の種類を指定します。有効な値は、次のとおりです。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> </ul> <p>デバッグログに返される情報の量を指定します。 のみで、ログカテゴリのレベルを使用します。</p> <p>有効なログレベルは次のとおりです(低いものから順に並べてあります)。</p> <ul style="list-style-type: none"> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> <li>•</li> </ul>

## PackageVersionHeader

インストールされた管理パッケージのパッケージバージョンを指定します。パッケージバージョンは、パッケージでアップロードされる一連のコンポーネントを特定する番号です。バージョン番号の形式は

*majorNumber.minorNumber.patchNumber* (例: 2.1.3) です。メジャー番号とマイナー番号は、メジャーリリース時に指定した値に増えます。*patchNumber* は、パッチリリースにのみ生成および更新されます。一連のコンポーネントのほか、パッケージバージョンには特定の動作が含まれています。公開者は、パッケージバージョンを使用して、パッケージを使用する既存の統合に影響を与えることなく後続のパッケージバージョンをリリースすることにより、管理パッケージのコンポーネントを強化することができます。

管理パッケージには、異なる内容および動作のさまざまなバージョンを指定できます。このヘッダーを使用して、API クライアントに参照される各パッケージに使用されるバージョンを指定できます。パッケージのバージョンが指定されていない場合、API クライアントは [設定] の [開発] > [API] の [バージョン設定] セクションで選択されているパッケージのバージョンを使用します。このヘッダーは、API バージョン 16.0 以降で使用できます。

## API コール

、  
、  
、

### 項目

要素名	型	説明
	PackageVersion[]	この API クライアントによって参照される、インストールされた管理パッケージバージョンのリスト。

### PackageVersion

インストールされた管理パッケージのバージョンを指定します。次の項目があります。

項目	型	説明
	int	パッケージバージョンのメジャー番号。パッケージバージョンは、「」のように、 <i>majorNumber.minorNumber</i> と表されます。
	int	パッケージバージョンのマイナー番号。パッケージバージョンは、「」のように、 <i>majorNumber.minorNumber</i> と表されます。
	string	管理パッケージの一意の名前空間。



# 用語集

---

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [Q](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#) | [X](#) | [Y](#) | [Z](#)

## A

### 管理者 (システム管理者)

アプリケーションの設定およびカスタマイズができる組織内の 1 人以上のユーザ。システム管理者プロファイルに割り当てられているユーザは、管理者権限があります。

### AJAX Toolkit

API 周辺の JavaScript ラッパーで、API コールを実行し、JavaScript コードで表示する権限を持つオブジェクトにアクセスできます。詳細については、『*AJAX Toolkit Developer's Guide*』を参照してください。

### 反結合

反結合は、SOQL クエリの `IN` 句の別のオブジェクトのサブクエリです。反結合を使用して、進行中の商談がないすべての取引先を取得するなど、高度なクエリを作成できます。「準結合」も参照してください。

### 匿名プロック、Apex

Salesforce に保存できないが、API コールまたは AJAX Toolkit の同等のコールを使用してコンパイルおよび実行できる Apex コードです。

### Apex

Apex は、開発者が Force.com プラットフォームサーバでフローとトランザクションの制御ステートメントを Force.com API へのコールと組み合わせて実行できるようにした、強く型付けされたオブジェクト指向のプログラミング言語です。Java に似た構文を使い、データベースのストアドプロシージャのように動作する Apex を使用して、開発者は、ボタンクリック、関連レコードの更新、および Visualforce ページなどのほとんどのシステムでのイベントに対しびジネスロジックを追加できます。Apex コードは、Web サービス要求、およびオブジェクトのトリガから開始できます。

### Apex による共有管理

開発者は、アプリケーションの動作をサポートする共有をプログラムで操作できるようになります。Apex による共有管理は、カスタムオブジェクトでのみ有効です。

### Apex ページ

「Visualforce ページ」を参照してください。

### アプリケーション

「App」と表記されることもあります。特定のビジネス要件を扱うタブ、レポート、ダッシュボードおよび Visualforce ページなどのコンポーネントの集合です。Salesforce では、セールスおよびコールセンターなどの

標準アプリケーションを提供しています。お客様のニーズに合わせてこれらの標準アプリケーションをカスタマイズできます。また、アプリケーションをパッケージ化して、カスタム項目、カスタムタブ、カスタムオブジェクトなどの関連コンポーネントと共に AppExchange にアップロードできます。そのアプリケーションを AppExchange から他の Salesforce ユーザが利用できるようにすることもできます。

## AppExchange

AppExchange は salesforce.com の共有インターフェースであり、Force.com プラットフォームのアプリケーションやサービスを参照および共有できます。

## アプリケーションプログラミンインターフェース (API)

コンピュータシステム、ライブラリ、またはアプリケーションが、その他のコンピュータプログラムがサービスを要求したりデータを交換したりできる機能を提供するインターフェースです。

## 承認プロセス

承認プロセスは、Salesforce でレコードを承認する場合に、組織で使用できる自動化されたプロセスです。承認プロセスでは、承認するレコードの条件と各承認ステップの承認者を指定します。各承認ステップは、その承認プロセスの対象レコードすべてに適用することも、システム管理者が定義した特定の条件を満たすレコードのみに適用することもできます。承認プロセスでは、レコードの承認、却下、取り消しまたは最初の承認申請時に実施するアクションも指定します。

## 非同期コール

操作に長い時間がかかるため、直ちに結果を返さないコールです。Metadata API と Bulk API のコールは非同期です。

## B

### Apex の一括処理

Apex を使用して多数のレコードに対して長く複雑な処理をスケジュールされた時間に実行する機能。

### ベータ、管理パッケージ

管理パッケージのベータ管理パッケージは、パッケージをテストするために対象者のサンプリングに貢献する旧バージョンの管理パッケージです。

### Bulk API

REST ベースの Bulk API は、大規模データセットの処理用に最適化されています。Salesforce によりバックグラウンドで処理される複数のバッチを送信することにより、多数のレコードを非同期でクエリ、挿入、更新、更新/挿入または削除できます。「SOAP API」も参照してください。

## C

### コールアウト、Apex

Apex コールアウトを使用して、外部 Web サービスへのコールを作成、または Apex コードから HTTP 要求を送信して応答を受信することによって、Apex を外部サービスと密接に統合することができます。

### 子リレーション

別の sObject を一对多リレーションの片方として参照する sObject に定義されたリレーション。たとえば、取引先責任者、商談および行動は取引先との子リレーションがあります。

「sObject」も参照してください。

## クラス、Apex

Apex オブジェクトの作成でベースとして使用する一種のテンプレート。他のクラス、ユーザ定義メソッド、変数、例外型、および static 初期設定化コードで構成されます。多くの場合、Apex クラスは、Java 内のその対応物に基づいています。

## クライアントアプリケーション

Salesforce ユーザインターフェースの外部で実行し、Force.com API または Bulk API のみを使用するアプリケーションです。通常、デスクトップまたはモバイルデバイス上で稼動します。これらのアプリケーションは、プラットフォームをデータソースとして扱い、設計されたツールおよびプラットフォームの開発モデルを使用します。

## コードカバー率

一連の単体テストが検証する、または検証しないコードの行を識別する手法。テストがまったく実行されないため、バグが含まれるリスクや将来のリリースで逆行する機能が導入される可能性が最も高いコードのセクションを特定するのに役立ちます。

## コンポーネント、メタデータ

コンポーネントは、Metadata API のメタデータ型のインスタンスです。たとえば、CustomObject はカスタムオブジェクトのメタデータ型で、コンポーネントはカスタムオブジェクトのインスタンスです。コンポーネントは XML ファイルに記述され、Metadata API を使用するか、Force.com IDE や Force.com 移行ツールなど、API で構築されたツールを使用してリリースしたり、取得したりできます。

## コンポーネント、Visualforce

などの一連のタグを使用して Visualforce ページに追加できます。Visualforce には、多くの標準コンポーネントが含まれていますが、独自のカスタムコンポーネントを作成することもできます。

## コンポーネントの参照、Visualforce

組織で使用できる Visualforce の標準コンポーネントおよびカスタムコンポーネントの説明。Visualforce ページの開発フッターまたは [『Visualforce 開発者ガイド』](#) からコンポーネントライブラリにアクセスできます。

## 複合アプリケーション

Yahoo! 地図など 1 つ以上の外部 Web サービスとネイティブのプラットフォーム機能を組み合わせるアプリケーション。複合アプリケーションを使用すれば、柔軟性が高まり、他のサービスとのインテグレーションが可能になります、外部コードの実行と管理が必要になる場合があります。「クライアントアプリケーション」と「ネイティブアプリケーション」も参照してください。

## コントローラ、Visualforce

Visualforce ページに実行する必要のあるデータおよびビジネスロジックを提供する Apex クラス。Visualforce ページは、デフォルトですべての標準オブジェクトまたはカスタムオブジェクトに付属する標準コントローラを使用、またはカスタムコントローラを使用できます。

## コントローラ拡張

コントローラ拡張は、標準コントローラまたはカスタムコントローラの機能を拡張する Apex クラスです。

## カスタムアプリケーション

「App」を参照してください。

## カスタムコントローラ

カスタムコントローラは、標準コントローラを使用せずにページのすべてのロジックを実装する Apex クラスです。Visualforce ページを完全にシステムモードで実行する場合に、カスタムコントローラを使用します。システムモードでは現在のユーザの権限と項目レベルのセキュリティが適用されません。

## カスタムリンク

カスタムリンクとは管理者によって定義された URL。これを使用して、Salesforce データを外部 Web サイトとバックエンドのオフィスシステムと統合します。以前は Web リンクと呼ばれていました。

## カスタムオブジェクト

組織固有の情報を保存することが可能なカスタムレコード。

## カスタム設定

カスタム設定はカスタムオブジェクトと類似しており、アプリケーション開発者は、カスタムデータセットの作成の他に、組織、プロファイル、または特定のユーザに対しカスタムデータを作成して関連付けることができます。すべてのカスタム設定データはアプリケーションキャッシュで公開されます。これにより、データベースへのクエリを繰り返し行うコストをかけずに、効率的なアクセスを実現します。さらに、このデータは、数式項目、入力規則、Apex、SOAP API で使用できます。

「階層カスタム設定」と「リストカスタム設定」も参照してください。

## D

### データベース

情報の編成されたコレクション。Force.com プラットフォームの基底となるアーキテクチャには、データが格納されているデータベースが含まれています。

### データベースステーブル

追跡する必要のある人物、物事、またはコンセプトに関する情報のリストで、行および列で表示されます。Object も参照してください。

### Salesforce 証明書と鍵のペア

Salesforce の証明書および鍵のペアは、要求がユーザの組織からのものであることを確認する署名として使用されます。証明書と鍵は、外部 Web サイトとの認証済み SSL 通信で使用されるか、組織を ID プロバイダとして使用するときに使用されます。要求が Salesforce 組織から行われていることの確認が必要な外部 Web サイトを使用している場合必要なのは、Salesforce 証明書と鍵のペアの生成のみです。

### データローダ

Salesforce 組織からデータをインポートおよびエクスポートするために使用する Force.com プラットフォームのツール。

### データ操作言語 (DML)

Force.com プラットフォームデータベースからレコードを挿入、更新、削除する Apex のメソッドまたは操作。

### データの状態

特定の時点でのオブジェクトに含まれるデータの構造。

## 日付リテラル

または など、時間の相対的範囲を示す SOQL クエリまたは SOSL クエリのキーワード。

## 小数点の位置

数値、通貨、パーセント項目で、小数点の右に入力できる桁数合計。たとえば、4.98 の場合は 2 となります。これ以上の桁の数値を入力した場合は、四捨五入されます。たとえば、小数点の位置が 2 の場合に 4.986 と入力すると、その数値は 4.99 となります。Salesforce では、切り上げアルゴリズムを使用します。中間値は常に切り上げられます。たとえば、1.45 は 1.5 に切り上げられます。-1.45 は -1.5 に切り上げられます。

## 連動関係

1つのオブジェクトの存在が別のオブジェクトの存在に依存する関係。必須項目、連動オブジェクト(親子)、ファイル含有(参照画像など)、および順序の依存性(あるオブジェクトをリリースする前に別のオブジェクトをリリースする必要がある場合)など、連動関係にはさまざまな種類があります。

## 連動項目

対応する制御項目で選択された値に基づいて、使用可能な値が表示される、カスタムの選択リストまたは複数選択の選択リストの項目。

## リリース

無効な状態の機能を有効な状態にします。たとえば、Salesforce ユーザインターフェースの新機能をリリースする場合、その機能を他のユーザが表示できるように「リリース済み」オプションを選択する必要があります。

アプリケーションまたは他の機能を開発から本番に移行するプロセスです。

ローカルファイルシステムから Salesforce 組織にメタデータコンポーネントを移動します。

インストールされたアプリケーションをリリースすると、アプリケーション内に、組織内のユーザが使用できるカスタムオブジェクトが作成されます。カスタムオブジェクトをリリース前に使用できるのは、「アプリケーションのカスタマイズ」権限を持つシステム管理者およびユーザのみです。

## 非推奨のコンポーネント

要件が時間と共に変化するにつれて、開発者は管理パッケージ内の機能を改良する場合があります。このとき、管理パッケージ内的一部のコンポーネントを再設計することが必要になる場合があります。「管理-リリース済み」パッケージのコンポーネントには開発者が削除できないものもありますが、後のパッケージバージョンでコンポーネントを非推奨にして、新しい登録者がそのコンポーネントを受け取らないようにすることができます。一方、そのコンポーネントは既存の登録者および API インテグレーションでは引き続き機能します。

## 詳細

単一のオブジェクトレコードに関する情報を表示するページ。レコードの詳細ページでは情報を表示できますが、編集ページでは変更が可能です。

レポートで、概要情報とレポートにあるすべての情報のすべての列データを含むものとを区別するための用語。[詳細の表示]/[詳細を非表示] を使用して、レポートの詳細の表示/非表示を切り替えることができます。

## Developer Force

Developer Force Web サイト ([developer.force.com](http://developer.force.com)) では、サンプルコード、ツールキット、オンライン開発者コミュニティなど、プラットフォーム開発者向けの幅広いリソースを提供しています。開発向けのForce.com プラットフォーム環境も、ここから入手できます。

## 開発環境

本番組織のユーザに影響を与えることなく設定変更を行える Salesforce 組織。 Sandbox 組織と Developer Edition 組織の 2 つの開発環境があります。

## E

### メールアラート

メールアラートは、メールテンプレートを使用してワークフロールールまたは承認プロセスによって生成され、Salesforce ユーザなど、指定された受信者に送信されるワークフローおよび承認アクションです。

### Enterprise WSDL

顧客が Salesforce 組織のみでインテグレーションを構築する場合や、パートナーが Tibco、webMethods などのツールを使って強い型付けが必要なインテグレーションを構築する場合に使用する強く型付けされた WSDL。Enterprise WSDL の欠点は、組織のデータモデルに存在するすべての一意のオブジェクトおよび項目にバインドされているため、1 つの Salesforce 組織のスキーマだけを扱うという点です。

### エンティティ関係図 (ERD)

データをエンティティ (または Force.com プラットフォームではオブジェクト) に整理し、それらのリレーションを定義することができるデータモデリングツール。主要な Salesforce オブジェクトの ERD ダイアグラムについては、『[SOAP API 開発者ガイド](#)』を参照してください。

## F

### Facet

表示された親領域を facet の内容で上書きできるようにする、別の Visualforce コンポーネントの子です。

### Field

テキストまたは通貨の値など、情報の特定の部分を保持するオブジェクトの一部。

### 項目の連動関係

別の項目の値に基づいて、選択リストの内容を変更できるフィルタ。

### 項目レベルセキュリティ

項目が、ユーザに非表示、表示、参照のみ、または編集可能であるかどうかを決定する設定。使用可能なエディションは、Enterprise Edition、Unlimited Edition、および Developer Edition です。

### Force.com

アプリケーションを構築する salesforce.com プラットフォーム。Force.com は、強力なユーザインターフェース、オペレーティングシステムおよびデータベースを結合して、企業全体でアプリケーションをカスタマイズおよび展開できます。

**Force.com IDE**

開発者が Eclipse 開発環境で Force.com アプリケーションを管理、作成、デバッグおよびリリースできる Eclipse プラグイン。

**Force.com 移行ツール**

ローカルファイルシステムと Salesforce 組織との間で Force.com コンポーネントを移行する Apache Ant 開発スクリプトを作成するためのツールキット。

**外部キー**

値が別のテーブルの主キーと同じ項目です。外部キーは、別のテーブルの主キーのコピーとしてみなすことができます。2つのテーブルのリレーションは、あるテーブルの外部キーの値と、別のテーブルの主キーの値が一致することによって成り立ちます。

**G****getter メソッド**

開発者がページのマークアップにデータベースその他の計算値を表示するためのメソッド。

値を返すメソッドです。「Setter メソッド」を参照してください。

**グローバル変数**

組織データの参照に使用できる特別な差し込み項目。

アプリケーション外 (SOAP API 内、または別の Apex コード) から参照する必要があるメソッドに使用するメソッドアクセス修飾子。

**ガバナ制限**

効率性の低いコードを作成する開発者が他の Salesforce ユーザのリソースを独占しないようにする Apex 実行の制限。

**グレゴリオ暦**

世界中で使用されている、12か月構造に基づいたカレンダーです。

**H****階層カスタム設定**

特定のプロファイルまたはユーザの設定を「カスタマイズ」できる組み込みの階層ロジックを使用するカスタム設定の種類。階層ロジックでは、現在のユーザの組織、プロファイル、およびユーザ設定を確認し、最も限定的な(つまり「最下位」)値が返されます。階層では、組織の設定はプロファイル設定によって上書きされ、プロファイル設定はユーザ設定によって上書きされます。

**HTTP デバッガ**

AJAX Toolkit から送信される SOAP 要求を識別し、調査するために使用できるアプリケーション。ローカルコンピュータで稼動するプロキシサーバとして動作し、各要求を調査および認証できます。

**I****ID**

「Salesforce レコード ID」を参照してください。

**IdeaExchange**

salesforce.com ユーザが新しい商品のコンセプトを提案したり、お気に入りの拡張機能を勧めたり、製品マネージャや他のユーザと対話したり、今後のリリースが予定される salesforce.com 製品のレビューを行ったりすることができるフォーラム。 IdeaExchange [ideas.salesforce.com](https://ideas.salesforce.com) を参照してください。

**インポートウィザード**

Salesforce 組織にデータをインポートするツール。[設定] からアクセスできます。

**インスタンス**

組織のデータをホストし、アプリケーションを実行する単一の論理サーバとして示されるソフトウェアおよびハードウェアのクラスタ。 Force.com プラットフォームは複数のインスタンスで稼動しますが、1つの組織のデータは常に1つのインスタンスに一元管理されています。

**インテグレーション開発環境 (IDE)**

ソースコードエディタ、テストツールおよびデバッグツール、ソースコード管理システムとの統合など、ソフトウェア開発者に包括的な機能を提供するソフトウェアアプリケーション。

**インテグレーションユーザ**

クライアントアプリケーションまたはインテグレーションのみに定義された Salesforce ユーザ。また、SOAP API コンテキストではログインユーザとも呼ばれます。

**ISO コード**

国際標準化機構が定める国コードで、各国を2文字で表します。

**J****連結オブジェクト**

2つの主従関係を持つカスタムオブジェクトです。カスタム連結オブジェクトを使用して、2つのオブジェクト間の「多対多」リレーションをモデル化できます。たとえば、「バグ」という名前のカスタムオブジェクトを作成し、1つのバグを複数のケースに、また1つのケースを複数のバグに関連付けることができます。

**K****鍵のペア**

「Salesforce 証明書と鍵のペア」を参照してください。

**L****文字数/桁数**

テキスト項目の場合、カスタム項目に入力できる最大文字数(255文字まで)を指定するパラメータ。

数値、通貨、パーセント項目の場合、整数部として入力できる桁数を指定するパラメータ。たとえば、123.98の場合は3と指定します。

**リストカスタム設定**

組織全体からアクセスできる再使用可能な静的データセットを提供するカスタム設定の種類。アプリケーション内で特定のデータセットを頻繁に使用する場合は、そのデータをリストカスタム設定に含めることにより、アクセスが簡素化されます。リスト設定に含まれるデータが、プロファイルやユーザごとに異なるということではなく、組織全体で利用できます。リストデータの例には、2文字の州の省略名、国際電話の発信番号、

製品のカタログ番号などがあります。データはキャッシュされるため、アクセスのコストが低く、効率的です。ガバナ制限の対象となる SOQL クエリを使用する必要はありません。

## リストビュー

特定の条件による項目(リード、取引先、または商談など)のリスト表示。Salesforce には、事前に定義されたビューがあります。

[コンソール] タブでは、リストビューが、具体的な条件に基づいてレコードのリストビューを表示する最上位のフレームです。[コンソール] タブに表示して選択できるリストビューは、各オブジェクトのタブで定義されたリストビューと同じです。コンソール内でリストビューを作成することはできません。

## ローカルネーム

ジェザまた遼取遼先の言謔蟠保懲垢（コ<sup>1</sup>でパ<sup>2</sup>）構（y<sup>3</sup>（ほ<sup>4</sup>8<sup>5</sup>ーザXやま蠶爻云ズににズ揆<sup>6</sup>ヰリ<sup>7</sup>ルカ侖請ナ門<sup>8</sup>）

## メタデータベースの開発

アプリケーションを宣言的な「設計図」として定義できるアプリケーション開発モデル。コードは必要ありません。データモデル、オブジェクト、フォーム、ワークフローなど、プラットフォームに構築されたアプリケーションはメタデータで定義されます。

## メタデータ WSDL

Force.com Metadata API コールを使用するユーザの WSDL。

## マルチテナンシー

すべてのユーザおよびアプリケーションが單一で共通のインフラストラクチャおよびコードベースを共有するアプリケーションモデル。

## MVC (Model-View-Controller)

アプリケーションをデータを示すコンポーネントに分割する設計パラダイム(モデル)、ユーザインターフェースでデータを表示する手段(ビュー)、およびビジネスロジックデータを使用してデータを処理する手段(コントローラ)。

## N

### 名前空間

パッケージコンテキストでは、ドメイン名と同様、AppExchange にある自社パッケージとその内容を他の開発者のパッケージと区別するための 1 ~ 15 文字の英数字で構成される識別子。Salesforce では、Salesforce 組織のすべての一意のコンポーネント名に自動的に名前空間プレフィックスとそれに続く 2 つのアンダースコア (\_) を追加します。

## ネイティブアプリケーション

Force.com の設定(メタデータ) 定義で排他的に開発されたアプリケーションです。ネイティブアプリケーションには、外部サービスまたは外部インフラストラクチャは必要ありません。

## O

### オブジェクト

Salesforce 組織に情報を保存するために使用するオブジェクト。オブジェクトは、保存する情報の種類の全体的な定義です。たとえば、ケースオブジェクトを使用して、顧客からの問い合わせに関する情報を保存できます。各オブジェクトについて、組織は、そのデータ型の具体的なインスタンスに関する情報を保存する複数のレコードを保有します。たとえば、佐藤次郎さんから寄せられたトレーニングに関する問い合わせに関する情報を保存するケースレコードと、山田花子さんから寄せられたコンフィグレーションの問題に関する情報を保存するケースレコードなどです。

### オブジェクトレベルのヘルプ

カスタムオブジェクトに提供できるカスタムヘルプのテキスト。カスタムオブジェクトレコードのホーム(概要)、詳細、編集ページ、リストビューや関連リストに表示されます。

### オブジェクトレベルセキュリティ

特定のユーザに対してオブジェクト全体を非表示にできる設定。ユーザはそうしたデータの存在を知ることもできません。オブジェクトレベルセキュリティはオブジェクト権限で指定されます。

## 一对多リレーション

1つのオブジェクトが多数のオブジェクトに関連するリレーション。たとえば、取引先に1つまたは複数の関連取引先責任者がある場合があります。

## 組織

ライセンスユーザセットが定義された Salesforce のリリース。組織は、salesforce.com の各お客様に提供される仮想スペースです。組織には、すべてのデータおよびアプリケーションが含まれており、他のすべての組織から独立しています。

## 組織の共有設定

ユーザが組織で持つデータアクセスのベースラインレベルを指定できる設定。たとえば、オブジェクト権限によって有効化されている特定のオブジェクトの任意のレコードを参照できますが、編集するには別の権限が必要となるよう、組織の共有設定を設定できます。

## アウトバウンドコール

Salesforce CRM Call Center のコールセンターの外部にユーザから発信するコール。

## アウトバウンドメッセージ

アウトバウンドメッセージは、外部サービスなどの指定したエンドポイントに指定の情報を送信するワークフロー、承認、およびマイルストン活動です。アウトバウンドメッセージは、エンドポイントに対し、特定の項目内のデータを SOAP メッセージとして送信します。アウトバウンドメッセージは、Salesforce の設定メニューで設定します。その後で、外部エンドポイントを設定する必要があります。SOAP API を使用して、メッセージのリスナーを作成できます。

## 所有者

レコード（取引先責任者またはケースなど）が割り当てられる個別ユーザ。

## P

### PaaS

「サービスとしてのプラットフォーム」を参照してください。

### パッケージ

AppExchange を介して他の組織で使用可能な Force.com のコンポーネントおよびアプリケーションのグループです。AppExchange にまとめてアップロードできるように、パッケージを使用してアプリケーションおよび関連するコンポーネントをバンドルします。

### パッケージの運動関係

これは、1つのコンポーネントが、そのコンポーネントが有効であるために必要な他のコンポーネント、権限、または設定を参照する場合に作成されます。コンポーネントに含めることができるのは、次のとおりです（ただし、それに限定するものではありません）。

- 標準項目またはカスタム項目
- 標準オブジェクトまたはカスタムオブジェクト
- Visualforce ページ
- Apex コード

権限と設定に含めることができるのは、次のとおりです（ただし、それに限定するものではありません）。

- Division

- ・ マルチ通貨
- ・ レコードタイプ

#### パッケージバージョン

パッケージバージョンは、パッケージでアップロードされる一連のコンポーネントを特定する番号です。バージョン番号の形式は *majorNumber.minorNumber.patchNumber* (例

関連 IT リソースを使用してアプリケーションをインストール、構成、保守する必要性を緩和します。PaaS 環境を使用して、あらゆる市場区分にサービスを配信することができます。

### Platform Edition

セールスやサービス & サポートなどの標準 Salesforce CRM アプリケーションを含まない Enterprise Edition または Unlimited Edition に基づいた Salesforce エディション。

### 主キー

リレーションナルデータベースのコンセプト。リレーションナルデータベースの各テーブルには、データ値が一意にレコードを識別する項目があります。この項目を、主キーと呼びます。2つのテーブルのリレーションは、あるテーブルの外部キーの値と、別のテーブルの主キーの値が一致することによって成り立ちます。

### 本番組織

実際の本番データとそれらにアクセスするライブユーザを持っている Salesforce 組織。

### プロトタイプ

他の Apex コードに使用できるクラス、メソッド、および変数。

## Q

### クエリロケータ

返された最後の結果レコードのインデックスを指定する、または  
されるパラメータ。

### クエリ文字列パラメータ

通常 URL の「?」文字の後に指定されている名前 - 値のペア。次に例を示します。

```
name=value
```

## R

### レコード

Salesforce オブジェクトの单一インスタンス。たとえば、「John Jones」は取引先責任者レコードの名前となります。

### レコード ID

「Salesforce レコード ID」を参照してください。

### レコードレベルセキュリティ

データを制御するメソッドで、特定のユーザがオブジェクトを参照および編集でき、ユーザが編集できるレコードを制限できます。

### レコードのロック

レコードのロックは、項目レベルのセキュリティや共有設定に関係なく、ユーザがレコードを編集できないようにします。未承認のレコードは、Salesforce によって自動的にロックされます。ロックされたレコードを編集するには、ユーザは特定のオブジェクトに対するオブジェクトレベルの「すべての編集」権限、または「すべてのデータの編集」権限が割り当てられている必要があります。[申請時のアクション] 関連リスト、[最終承認時のアクション] 関連リスト、[最終却下時のアクション] 関連リスト、および [取り消し時のアク

ション] 関連リストには、デフォルトで「レコードのロック」アクションが含まれています。申請時のアクションおよび取り消し時のアクションでは、このデフォルトのアクションを編集することはできません。

## レコード名

すべての Salesforce オブジェクトの標準項目。レコード名が Force.com アプリケーションに表示されると、値はレコードの詳細ビューへのリンクとして表示されます。レコード名は自由形式のテキストまたは自動採番項目です。レコード名には、必ずしも一意の値を割り当てる必要はありません。

## ごみ箱

削除した情報を表示し、復元できるページ。ごみ箱には、サイドバー内のリンクからアクセスします。

## リレーション

ページレイアウト内の関連リストおよびレポート内の詳細レベルを作成するために使われる、2つのオブジェクトの間の接続。両方のオブジェクトの特定の項目において一致する値を使用して、関連するデータにリンクします。たとえば、あるオブジェクトには会社に関連するデータが保存されていて、別のオブジェクトには人に関連するデータが保存されている場合、リレーションを使用すると、その会社で働いている人を検索できます。

## リレーションクエリ

SOQL コンテキストで、オブジェクト間のリレーションを辿り、結果を識別および返すクエリ。親対子および子対親の構文は、SOQL クエリでは異なります。

## ロール階層

レコードレベルのセキュリティで使用される設定。ロール階層によって特定のレベルのロールを割り当てられたユーザは、組織の共有モデルとは関係なく、階層において自分よりも下位のユーザが所有しているデータ、および該当のユーザと共有しているデータに対する参照、編集権限を持つことになります。

## 積み上げ集計項目

主従関係の子レコードの値の集計値を自動的に提供する項目の種別。

## 実行ユーザ

各ダッシュボードには実行ユーザが指定され、そのユーザのセキュリティ設定によってダッシュボードに表示されるデータが決まります。実行ユーザが特定の1ユーザである場合、すべてのダッシュボード閲覧者には、閲覧者個々人のセキュリティ設定に関係なく、実行ユーザのセキュリティ設定に基づいてデータが表示されます。動的ダッシュボードの場合、実行ユーザをログインユーザに設定するため、各ユーザには独自のアクセスレベルに従ってダッシュボードが表示されます。

## S

### SaaS

「サービスとしてのソフトウェア (SaaS)」を参照してください。

### Sコントロール



メモ: Sコントロールは、Visualforce ページに置き換えられました。2010 年以降、新しい組織同様、S コントロールを作成したことのない組織は、Sコントロールを作成できなくなります。既存の Sコントロールに影響はありません。今後も編集できます。

カスタムリンクで使用するカスタム Web コンテンツ。カスタム S コントロールには、Java アプレット、Active-X コントロール、Excel ファイル、カスタム HTML web フォームなど、ブラウザに表示できるあらゆる種類のコンテンツを入れることができます。

#### Salesforce 証明書と鍵のペア

Salesforce の証明書および鍵のペアは、要求がユーザの組織からのものであることを確認する署名として使用されます。証明書と鍵は、外部 Web サイトとの認証済み SSL 通信で使用されるか、組織を ID プロバイダとして使用するときに使用されます。要求が Salesforce 組織から行われていることの確認が必要な外部 Web サイトを使用している場合必要なのは、Salesforce 証明書と鍵のペアの生成のみです。

#### Salesforce レコード ID

Salesforce の 1 つのレコードを識別する 15 文字または 18 文字の一意の英数字文字列。

#### Salesforce SOA (サービス指向アーキテクチャ)

Apex 内から外部 Web サービスへのコールを作成できる Force.com の強力な機能。

#### Sandbox 組織

Salesforce 本番組織のほぼ同一コピー。テストやトレーニングなどさまざまな目的のために、本番組織のデータとアプリケーションに影響を与えることなく、複数の Sandbox をそれぞれの環境に作成できます。

#### 準結合

準結合は、SOQL クエリの一句の別のオブジェクトのサブクエリです。準結合を使用して、特定のレコードタイプの商談がある取引先のすべての取引先責任者を取得するなど、高度なクエリを作成できます。「[反結合](#)」も参照してください。

#### セッション ID

ユーザが Salesforce に正常にログインした場合に返される認証トークン。セッション ID を使用すると、ユーザが Salesforce で別のアクションを実行するときに毎回ログインする必要がなくなります。レコード ID または Salesforce ID と異なり、Salesforce レコードの一意の ID を示す用語です。

#### セッションタイムアウト

ログインしてからユーザが自動的にログアウトするまでの時間。セッションは、前もって決定された非活動状態の期間の後、自動的に終了します。非活動状態の期間の長さは、[セキュリティのコントロール] をクリックすることによって [設定] の Salesforce で設定できます。デフォルト値は 120 分 (2 時間) です。ユーザが Web インターフェースでアクションを実行または API コールを作成すると、非活動状態タイマーが 0 にリセットされます。

#### Setter メソッド

値を割り当てるメソッド。Getter メソッドも参照してください。

#### 設定

システム管理者が組織の設定および Force.com アプリケーションをカスタマイズおよび定義できるメニューです。組織のユーザインターフェース設定に応じて、[設定] はユーザインターフェースのヘッダーでリンクになっている場合もあれば、ユーザ名の下でドロップダウンリストになっている場合もあります。

#### Sites

Force.com Sites では、公開 Web サイトとアプリケーションを作成できます。それらは Salesforce 組織と直接統合されるため、ユーザがログインする場合にユーザ名やパスワードは必要ありません。

**SOAP (Simple Object Access Protocol)**

XML 符号化データを渡す一定の方法を定義するプロトコル。

**SOAP API**

Salesforce 組織の情報へのアクセスを提供する SOAP ベースの Web サービスアプリケーションのプログラミングインターフェース。

**sObject**

Force.com プラットフォームに保存できるオブジェクト。

**サービスとしてのソフトウェア (SaaS)**

ソフトウェアアプリケーションがサービスとしてホストされ、顧客にインターネットを経由して提供される配信モデル。SaaS ベンダは、アプリケーションおよび各顧客データの日常メンテナンス、操作およびサポートを行う責任があります。このサービスで、顧客が独自のハードウェア、ソフトウェア、そして関連 IT リソースを使用してアプリケーションをインストール、構成、保守する必要性を緩和します。SaaS モデルを使用して、あらゆる市場区分にサービスを配信することができます。

**SOQL (Salesforce オブジェクトクエリ言語)**

Force.com データベースからデータを選択するために使用する必要のある単純で強力なクエリ文字列を構築し、基準を指定できるクエリ言語。

**SOSL (Salesforce オブジェクト検索言語)**

Force.com API を使用して、テキストベースの検索を実行できるクエリ言語。

**標準オブジェクト**

Force.com プラットフォームに含まれる組み込みオブジェクト。アプリケーション独自の情報を格納するカスタムオブジェクトを作成することもできます。

**システムログ**

開発者コンソールの一部。コードスニペットのデバッグに使用できる独立したウィンドウ。ウィンドウの下部にテストするコードを入力して、[実行] をクリックします。システムログの本文には、実行する行の長さや、作成されたデータベースコール数などのシステムリソース情報が表示されます。コードが完了しなかった場合は、コンソールにデバッグ情報が表示されます。

**T****タグ**

Salesforce でデータを独自の方法で記述および整理するために使用され、ほとんどのレコードに関連付けることができる単語または短い語句。システム管理者がタグを有効化できるのは、取引先、活動、納入商品、キャンペーン、ケース、取引先責任者、契約、ダッシュボード、ドキュメント、行動、リード、メモ、商談、レポート、ソリューション、ToDo、およびカスタムオブジェクト(リレーションシップグループメンバーを除く)です。タグには、SOAP API からもアクセスできます。

Salesforce CRM Content では、タグは、ライブラリ全体の内容を分類および整理するための説明ラベルのこと を意味します。ユーザは、特定のタグに属すすべてのファイルや Web リンクのリストを表示したり、タグに基づいて検索結果を絞り込んだりすることができます。

## テストケースカバー率

テストケースは、コードを使用した、予測される実際のシナリオです。テストケースは実際の単体テストではありませんが、単体テストが実施する内容を指定するドキュメントです。テストケースのカバー率が高い場合、特定されたすべてまたはほとんどの実際のシナリオが単体テストとして実装されます。「コードカバー率」と「単体テスト」も参照してください。

## Test メソッド

特定のコードが適切に動作しているかを確認する Apex クラスメソッド。Test メソッドは引数を採用せず、データをデータベースにコミットしません。また、コマンドラインまたは Force.com IDE のような Apex IDE でシステムメソッドによって実行できます。

## テスト組織

テスト目的でのみ使用される Salesforce 組織。Sandbox 組織も参照してください。

## トランザクション、Apex

Apex トランザクションは、1 つの単位として実行される一連の操作を表します。トランザクションの実行には、すべての DML 操作が正常に完了することが求められます。いずれかの操作でエラーが発生した場合はトランザクション全体がロールバックされます。この場合、データは一切データベースにコミットされません。トランザクションの境界は、トリガ、クラスメソッド、匿名のコードブロック、Visualforce ページ、カスタム Web サービスマソッドのいずれかにすることができます。

## トリガ

データベースの特定の種類のレコードが挿入、更新、または削除される前後で実行する Apex の一部です。各トリガは、トリガが実行されるレコードへのアクセス権限を提供する一連のコンテキスト変数で実行し、すべてのトリガは一括モードで実行します。つまり、一度に 1 つずつレコードと処理するのではなく、複数のレコードを一度に処理します。

## トリガコンテキスト変数

トリガおよびトリガが起動するレコードに関する情報へのアクセス権限を提供するデフォルト値。

## U

## V

## 入力規則

指定される基準に一致しない場合、レコードを保存しない規則。

## バージョン

項目のリリースを示す数値。バージョンを表示できる項目は、API オブジェクト、項目およびコール、Apex クラスおよびトリガ、Visualforce ページおよびコンポーネントです。

## ビュー

Visualforce で定義された Model-View-Controller モデルのユーザインターフェース。

## ビューステート

要求間のデータベース状態を維持するために必要なすべての情報が、ビューステートに保存されます。

## Visualforce

開発者が、プラットフォームに作成されたアプリケーションのカスタムページおよびコンポーネントを容易に定義できる、単純で、タグベースのマークアップ言語。各タグが、ページのセクション、関連リスト、または項目など、大まかなコンポーネントと細かいコンポーネントのどちらにも対応しています。コンポーネントは、標準の Salesforce ページと同じロジックを使用して制御することができます。また、開発者が独自のロジックを Apex で記述されたコントローラと関連付けることもできます。

## Visualforce コントローラ

コントローラ、Visualforce を参照してください。

## Visualforce ライフサイクル

ユーザセッションでページがどのように作成されて破棄されるかを示す Visualforce ページの各実行フェーズ。

## Visualforce ページ

Visualforce を使用して作成された Web ページ。通常、Visualforce ページには組織に関連する情報が表示されますが、データの変更や取得も可能です。PDF ドキュメントやメールの添付ファイルなど、さまざまな方法で表示できます。また CSS スタイルに関連付けることもできます。

## W

### Web サービス

さまざまなプラットフォームで稼動、さまざまな言語で作成、またはお互い地理的に離れている場合であっても、2つのアプリケーションがインターネットを経由してデータを容易に交換できるメカニズム。

### WebService メソッド

サードパーティのアプリケーションのマッシュアップなど、外部システムによって使用できる Apex クラス メソッドまたは変数。Web サービスマソッドは、グローバルクラスで定義する必要があります。

### Web サービス API

Salesforce 組織の情報へのアクセスを提供する Web サービスマネージャー。『SOAP API』および『Bulk API』も参照してください。

### ワークフローと承認時のアクション

ワークフローと承認時のアクションは、ワークフロールールまたは承認プロセスで起動できるメールアラート、ToDo、項目自動更新、アウトバウンドメッセージで構成されています。

### ラッパークラス

ログイン、セッションの管理、レコードのクエリおよびバッチなど、一般的な機能を抽象化するクラス。ラッパークラスを使用すると、インテグレーションでより簡単にプログラムロジックを開発、保持、および一か所に保存でき、コンポーネント間で容易に再利用できるようになります。Salesforce のラッパークラスの例として、Salesforce SOAP API の JavaScript ラッパーである AJAX Toolkit、Salesforce CRM Call Center の CTI Adapter で使用される などのラッパークラス、または SOAP API を使用して Salesforce にアクセスするクライアントインテグレーションアプリケーションの一部として作成されたラッパークラスなどがあります。

### WSDL (Web Services Description Language) ファイル

Web サービスと送受信するメッセージの形式を説明する XML ファイル。開発環境の SOAP クライアントは、Salesforce Enterprise WSDL または Partner WSDL を使用して、SOAP API で Salesforce と通信します。

X

**XML (拡張可能マークアップ言語)**

構造化データの共有と移動を可能にするマークアップ言語。Metadata API を使用して取得またはリリースされるすべての Force.com コンポーネントは、XML 定義に従って表されます。

Y

該当用語はありません。

Z

該当用語はありません。