



Visualforce Workbook: Spring '11

# Visualforce Workbook



Last updated: February 28, 2011



## 目次

このワークブックについて.....	3
チュートリアル 1: Visualforce ページの作成と確認.....	4
ステップ 1: Visualforce 開発モードを有効にする.....	4
ステップ 2: Visualforce ページを作成する.....	4
ステップ 3: Visualforce ページを編集する.....	5
ステップ 4: 作成済みの Visualforce ページを確認する.....	6
ステップ 5: Page Editor 以外の方法でページを作成する.....	6
まとめ.....	6
チュートリアル 2: Visualforce の属性、入力支援機能、コンポーネント.....	7
ステップ 1: 属性を追加し、入力支援機能を活用する.....	7
ステップ 2: 新しいコンポーネントを追加する.....	9
ステップ 3: コンポーネントをネストする.....	10
まとめ.....	10
チュートリアル 3: シンプルな変数と数式.....	12
ステップ 1: グローバル変数.....	12
ステップ 2: 基本数式.....	12
ステップ 3: 条件式.....	13
まとめ.....	14
チュートリアル 4: 標準コントローラ.....	15
ステップ 1: レコードの ID を取得する.....	15
ステップ 2: レコードのデータを取得して表示する.....	16
ステップ 3: 他の項目を表示する.....	17
ステップ 4: リレーションが設定されたレコードの項目を表示する.....	17
まとめ.....	18
チュートリアル 5: 標準のユーザインターフェースコンポーネント.....	19
ステップ 1: 特定のレコードに関する詳細と関連リストを表示する.....	19
ステップ 2: 表示する項目を切り替える.....	19
ステップ 3: テーブルを表示する.....	20
まとめ.....	21
チュートリアル 6: ページの上書きと参照.....	22
ステップ 1: ページのデフォルトの表示を上書きする.....	22
ステップ 2: 標準のページレイアウトに Visualforce ページを埋め込む.....	22
ステップ 3: Visualforce ページを表示するボタンを作成する.....	23
ステップ 4: Visualforce ページから他の Visualforce ページや外部 URL を参照するハイパーリンクを作成する.....	24
まとめ.....	24
チュートリアル 7: カスタムのユーザインターフェースコンポーネント.....	25
ステップ 1: ヘッダーとサイドバーを削除する.....	25
ステップ 2: CSS で HTML のスタイルを設定する.....	25

ステップ 3: リストとテーブルの反復処理を利用する .....	26
まとめ .....	27
チュートリアル 8: フォームを使ったデータの入力 .....	28
ステップ 1: 基本となるフォームを作成する.....	28
ステップ 2: フォームに項目の表示ラベルを表示する.....	28
ステップ 3: 警告やエラーを表示する .....	29
まとめ .....	29
チュートリアル 9: テンプレートを使用したページの再利用 .....	31
ステップ 1: テンプレートを作成する.....	31
ステップ 2: テンプレートを使用する.....	31
ステップ 3: Visualforce ページを他の Visualforce ページに挿入する.....	32
まとめ .....	32
チュートリアル 10: カスタムコンポーネント .....	33
ステップ 1: シンプルなカスタムコンポーネントを作成する .....	33
ステップ 2: カスタムコンポーネントを使用する .....	33
まとめ .....	34
チュートリアル 11: Ajax によるページの部分的な更新.....	35
ステップ 1: 動的に更新する領域を明示的に指定する.....	35
ステップ 2: 領域を動的に更新する .....	35
まとめ .....	36
チュートリアル 12: コントローラ拡張機能による機能の追加 .....	37
ステップ 1: コントローラ拡張機能を作成する .....	37
ステップ 2: Visualforce ページに拡張機能を追加する .....	37
まとめ .....	38
チュートリアル 13: カスタムコントローラ.....	39
ステップ 1: カスタムコントローラを含んだページを作成する .....	39
ステップ 2: 取引先レコードを取得するメソッドを追加する .....	39
ステップ 3: コントローラにアクションメソッドを追加する.....	40
まとめ .....	41
おわりに .....	42

## このワークブックについて

Visualforce は、Force.com プラットフォーム上でネイティブホストが可能な高度なカスタムユーザインターフェースを開発できるフレームワークです。このワークブックでは、Visualforce のさまざまな機能をわかりやすく解説します。

Force.com の標準ユーザインターフェースに似たユーザインターフェースを作成するほか、きめ細かく制御された独自のユーザインターフェースを作成します。合わせて、Visualforce ページで共通に利用するコンポーネントを作成する方法や、Visualforce とアプリケーションを連携させる方法なども学びます。さらに、Visualforce の基盤になっている MVC (Model-View-Controller: モデル-ビュー-コントローラ)、Apex コードなどについても解説します。

このワークブックの目的は、Visualforce の数多くの機能を紹介することであり、実際のアプリケーションを作成することではありません。学習を進めながら、コードに少し手を加えたり、別のコンポーネントを使ったりして、自由に学んでみてください。

### 学習を始める前に

一連のチュートリアルは、Force.com Developer Edition 環境を使って作業することを前提にしています。Developer Edition は、<http://developer.force.com/join> から無料で利用できます。これから紹介するすべてのテクニックは、開発をサポートする他の Force.com 環境にも適用可能です。たとえば、Force.com Free Edition 環境でも、同様のテクニックを簡単に試すことができます。

また、Force.com 自体についても学んでおくと役に立ちます。「[Force.com Workbook](#)」の前半のいくつかのチュートリアルが参考になります。



**メモ:** Force.com の開発環境では、Mozilla Firefox バージョン 3.6.x、Microsoft® Internet Explorer 6.0、7.0、8.0、および Apple® Safari バージョン 5.0.x の各ブラウザをサポートしています。詳細は、Salesforce のオンラインヘルプで「サポートされるブラウザ」のトピックを参照してください。

### 学習を終えた後は

チュートリアルを最後まで終えたら、以下の Web リソースを活用して Visualforce と Force.com を使った開発についてさらに深く学ぶことができます。

- Visualforce 早見表 ([http://developerforcejp.s3.amazonaws.com/books/cheatsheets/Visualforce-cheatsheet\\_Ja.pdf](http://developerforcejp.s3.amazonaws.com/books/cheatsheets/Visualforce-cheatsheet_Ja.pdf))
- Force.com Workbook (<http://wiki.developerforce.com/index.php/JP:Forcedotcomworkbook>): Force.com と Visualforce についてさらに詳しく学べるワークブックです。
- Developerforce (<http://developer.salesforce.co.jp/>): Force.com について調べたり、さまざまな情報や資料を閲覧したりできます。「[Visualforce Developer's Guide](#)」(英語) は必ずチェックしてください。

# チュートリアル 1: Visualforce ページの作成と確認

このチュートリアルでは、Visualforce ページを新規に作成して編集します。今回作成するページはとてもシンプルなものです。これをベースに、さまざまな機能を学んでいくことができます。チュートリアルを進めるにつれ、エディタとページの自動作成に関する知識を身に付けられるでしょう。

作業を開始する前に、上述の「学習を始める前に」で説明した無料の Force.com Developer Edition 環境を用意しておいてください。

## ステップ 1: Visualforce 開発モードを有効にする

開発モードを有効にすると、Visualforce 用のエディタである Page Editor を使って、Web ブラウザ内でコードを確認しながらページをプレビューできるようになります。開発モードでは、コントローラやコントローラ拡張機能を編集できる Apex 用のエディタも提供されます。

1. [**<あなたの名前>**] > [**設定**] > [**私の個人情報**] > [**個人情報**] の順にクリックします。
2. [**編集**] をクリックします。
3. [**開発モード**] チェックボックスをオンにして、[**保存**] をクリックします。

## ステップ 2: Visualforce ページを作成する

ステップ 1 の作業で Visualforce ページを作成する準備が整いました。では、ページを作成してみましょう。

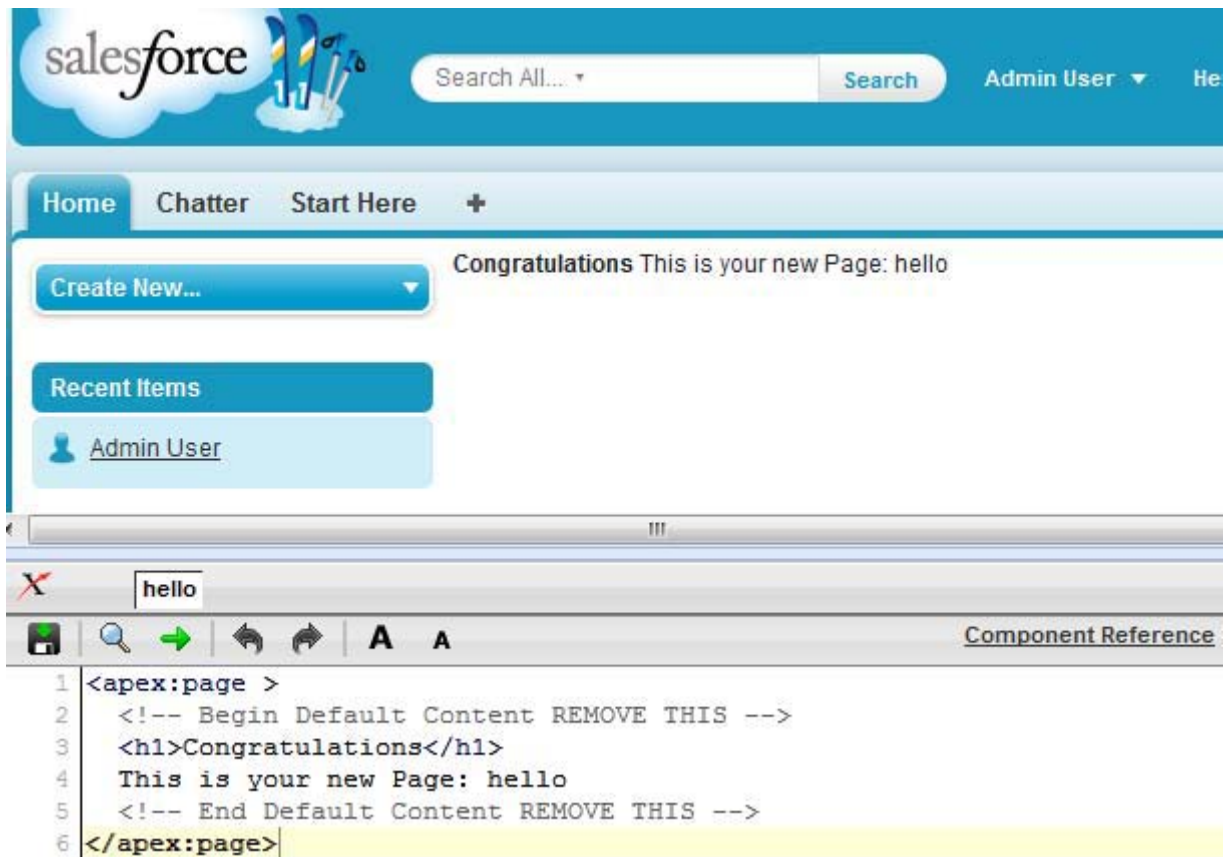
1. ブラウザのアドレスバーで、Salesforce インスタンスの URL に「/apex/hello」を追加します (たとえば、インスタンスが「https://na1.salesforce.com」の場合は、「https://na1.salesforce.com/apex/hello」となります)。すると、次のようなエラーメッセージが表示されます。

### Visualforce Error

**Page hello does not exist**

 [Create Page hello](#)

2. [**Create Page hello**] リンクをクリックして、ページを新規作成します。



これにより、デフォルトのテキストを含んだページが表示されます。また、ページの下部にはソースコードが記述された Page Editor が表示されます。以上が、一連のチュートリアルでページを作成するときの基本的な操作です。

## ステップ 3: Visualforce ページを編集する

Visualforce ページを作成した後、次に必要になるのは、必要に応じてページをカスタマイズすることです。このステップでは、ページを編集しながらリアルタイムに変更をプレビューしてみます。

1. ページの先頭の「Congratulations」というテキストを「Hello World」に変えたいとしましょう。この場合、<h1> タグで囲まれたコメントテキストを変更して上書きします。コードは次のようになります。

```

<apex:page>
  <h1>Hello World</h1>
</apex:page>

```

2. Page Editor の上部にある**保存**ボタンをクリックします。

ページが再度読み込まれ、変更内容が反映されます。「Hello World」の文字は大きく表示されます。これは HTML 標準の <h1> タグであるためです。Visualforce ページは通常 2 種類のタグで構築されています。1 つは Visualforce コンポーネントを表すタグ (<apex:page> など) で、もう 1 つは標準の HTML タグです。

ステップ 1 で開発モードを有効にしているため、開発をすばやく簡単に進められるようになっています。変更も容易で、保存ボタンをクリックするとただちにページに変更内容が反映されます。ショートカットキー **Control + S** を使ってこまめに保存することもできます。エディタの最小化ボタンをクリックすると、ページ部分の表示領域を最大化できます。

なお、作成したページを実際の運用環境に展開したり、開発モードを無効にしたりすると、Page Editor は利用できなくなります。

## ステップ 4: 作成済みの Visualforce ページを確認する

Visualforce ページを作成したら、そのページにどこからアクセスできるかを把握しておきましょう。

1. [**<あなたの名前>**] > [**設定**] > [**開発**] > [**ページ**] の順にクリックします。
2. [Visualforce ページ] が開きます。画面を下へスクロールすると、ステップ 2 で作成したページ「hello」がリストされています。

以上の手順で作成したページを確認できます。ここで、[編集] リンクをクリックしてページを編集することも可能です。ただし、このページの編集機能は前のステップで使った Page Editor よりも機能が限定的であるため、別のウィンドウでページを開いておかない限り、変更内容をすぐに確認することはできません。そのため、一連のチュートリアルでは、こちらの編集機能は使用しません。

## ステップ 5: Page Editor 以外の方法でページを作成する

この [Visualforce ページ] の画面からでも、ステップ 2 と同様に、新しいページを作成することが可能です。手順としては、ページを作成してからそのページの名前を使って URL を指定し、ページを表示します。

1. [**<あなたの名前>**] > [**設定**] > [**開発**] > [**ページ**] > [**新規**] の順にクリックします。
2. 「hello2」という名前で新しい Visualforce ページを作成します。
3. 作成したページを表示します。ステップ 2 のときと同様に、ブラウザのアドレスバーに URL「[https://<利用中のSalesforce インスタンス>/apex/hello2](https://<利用中のSalesforceインスタンス>/apex/hello2)」を指定します。

## まとめ

このチュートリアルでは、開発モードを有効化する方法、Visualforce ページを作成する方法、作成したページを確認する方法などを学びました。次のチュートリアルでは、Page Editor の機能についてももう少し学習し、Visualforce ページの構成要素であるコンポーネントの基本を学びます。

## チュートリアル 2: Visualforce の属性、入力支援機能、コンポーネント

チュートリアル 1 で作成したページには、すべての Visualforce ページに共通する要素が含まれています。開始時と終了時に `<apex:page>` というタグが使用されているのです。これは、Visualforce ページには不可欠なコンポーネントであり、ページ内では次のように記述されます。

```
<apex:page>
  Your Stuff Here
</apex:page>
```

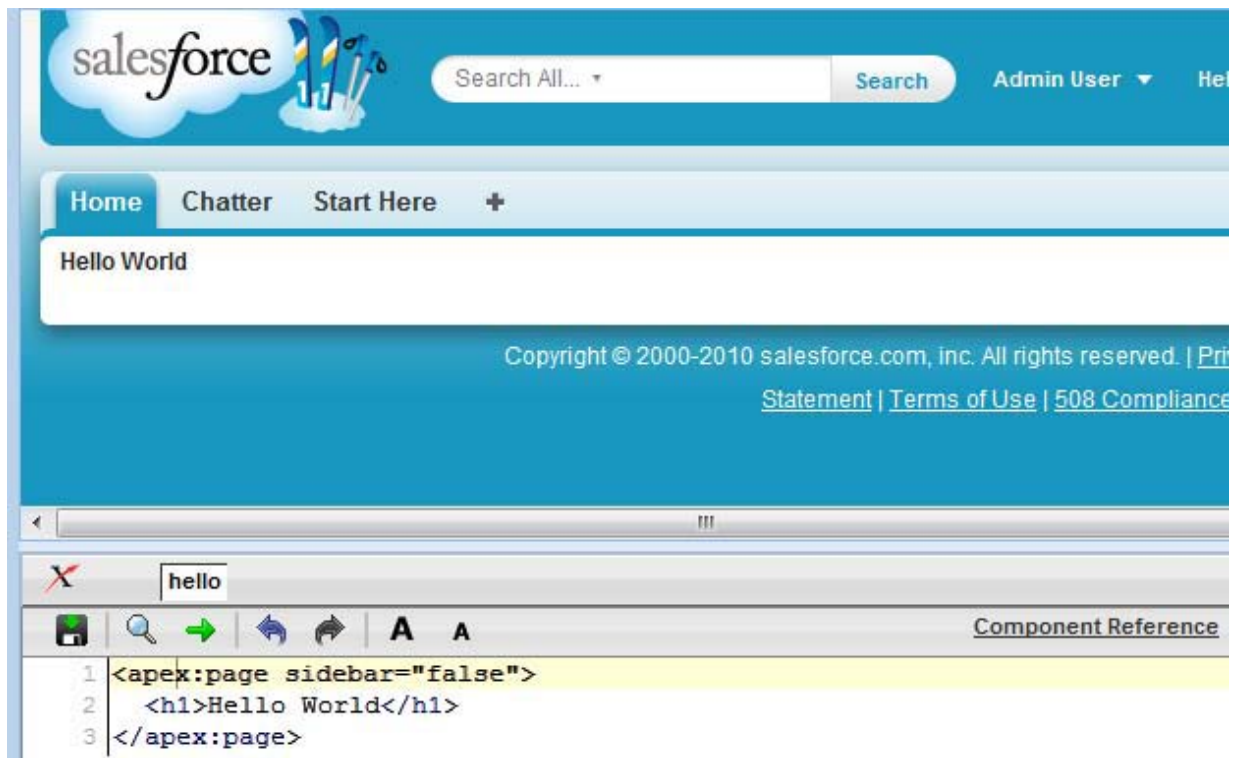
山かっこの使い方、コンポーネントの開始と終了の指定方法に注意してください。開始タグでは、`<apex:page>` のように、コンポーネント名を山かっこで囲みます。終了タグでは、`</apex:page>` のように、コンポーネント名の前にスラッシュを (/) を挿入して山かっこで囲みます。作成する Visualforce ページは整形形式の XML 文書である必要があるため、一部の例外を除き、常にこの規則に従います。一部のコンポーネントには、`<apex:detail />` の形式のように自己完結するものがあります (/ の位置に注意してください)。これは、1 つのタグ内に開始タグと終了タグが含まれているものと考えてください。

コンポーネントの動作や外観を変更するには、通常、属性を追加します。この属性は名前と値のペアで構成され、開始タグ内に含まれます。たとえば、`sidebar="false"` のような形で開始タグ内に記述されます。

### ステップ 1: 属性を追加し、入力支援機能を活用する

このステップでは、`<apex:page>` コンポーネントの有効な属性である `sidebar` 属性を追加します。入力候補を自動的に表示してくれる入力支援機能も使ってみましょう。

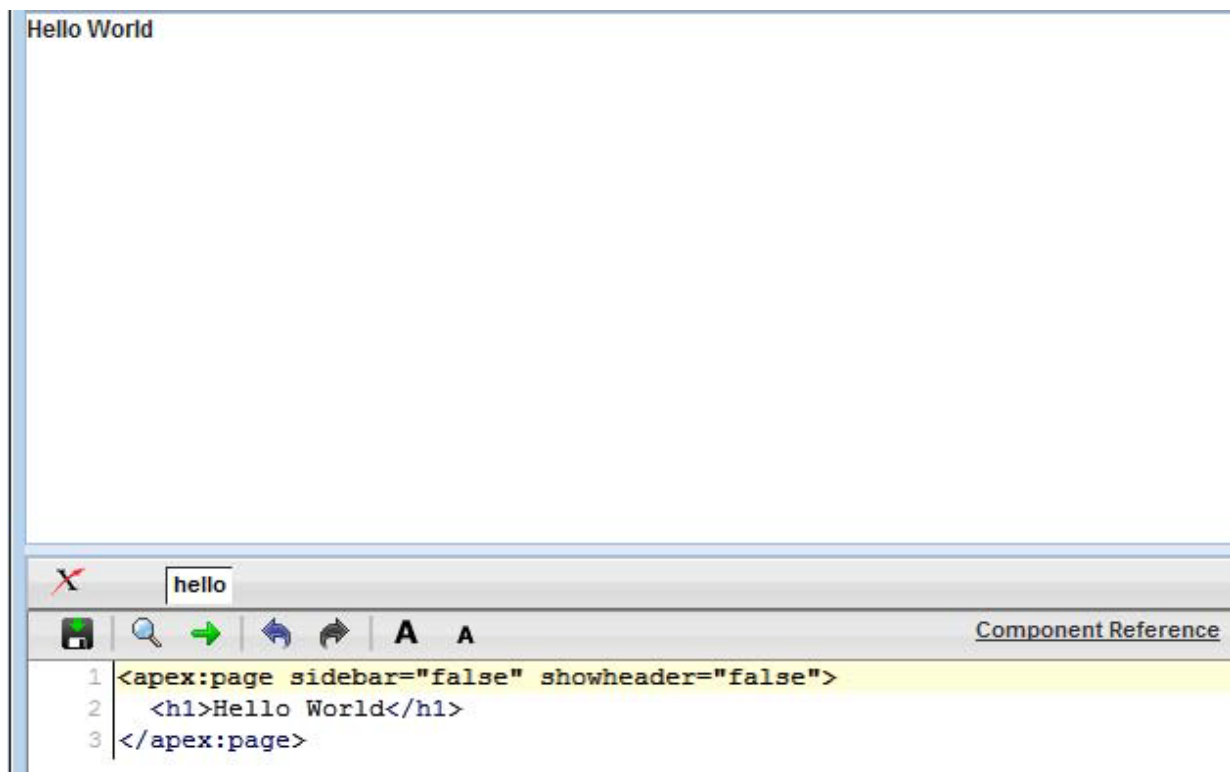
1. `<apex:page>` コンポーネントの開始タグ内に「`sidebar="false"`」を追加して、`<apex:page sidebar="false">` と記述します。
2. **保存ボタン**をクリックします。  
ページの左側の領域を見ると、サイドバーが削除されています。`sidebar` 属性により、`<apex:page>` コンポーネントの動作と外観を変更できました。



3. 末尾の引用符 (") の直後にカーソルを合わせ、スペースキーを押すと、`<apex:page>` コンポーネントに設定できる属性のリストがポップアップ表示されます。リストの中から `showHeader` 属性を選択します。
4. `showHeader` 属性を選択すると、入力支援機能により、値の指定を求められます。「false」を選択します。以上により、`<apex:page sidebar="false" showHeader="false">` という開始タグを記述できました。

```
<apex:page sidebar="false" showHeader="false">
```

5. **保存** ボタンをクリックします (ショートカットキー `Control + S` も使用できます)。ページの外観が大きく変わりました。`showHeader` 属性を `false` に設定したことで、上部のヘッダーだけでなく、ページのデフォルトのスタイルがすべて削除されています。



開発では、上部のヘッダーがあったほうが便利なので、元に戻しておきましょう。

6. showHeader 属性の値を「true」に変更します。
7. 保存ボタンをクリックします。

## ステップ 2: 新しいコンポーネントを追加する

これまでのステップでは、ページを作成し、<apex:page> コンポーネントを使用して Visualforce ページの動作を変更しました。続いて、別のコンポーネントを使用して機能を追加してみましょう。

Visualforce には、多数のコンポーネントがあらかじめ組み込まれていますが、コンポーネントを追加したり独自に作成したりすることで、コンポーネントセットを拡張できます。このステップでは、追加コンポーネントの探し方と利用方法を学びます。

1. Page Editor の **[Component Reference]** (コンポーネントリファレンス) リンクをクリックします。使用可能なコンポーネントをリストしたポップアップウィンドウが表示されます。
2. 例として <apex:pageBlock> をクリックします。[Component Details] (コンポーネントの詳細) タブに、コンポーネントの機能と、動作を変更するために追加できる属性についての説明が表示されます。
3. [Usage] (使用法) タブをクリックすると、コンポーネントの使用例が表示されます。<apex:pageBlock> コンポーネントとともに使われるコンポーネントとしては <apex:pageBlockSection> などがあるようです。左のリストでそのコンポーネント名をクリックすると、<apex:pageBlockSection> コンポーネントの詳細な情報を確認できます。

基本的に、必要なコンポーネントはコンポーネントリファレンスから見つけることができます。このリファレンスを使えば、主要なコンポーネントの機能を容易に確認できます。多数の属性をとるコンポーネントもありますが、実際に使用する属性の数はそれほど多くありません。

では、この 2 つのコンポーネントをページに追加してみましょう。手順を簡単に説明します。コード例は最後に示していますが、その例を見ないようにして記述してみてください。

4. <apex:page> コンポーネントの内側に <apex:pageBlock> コンポーネントを追加し、title 属性の値に「A Block Title」を指定します。

- さらに、`<apex:pageBlock>` コンポーネントの内側に `<apex:pageBlockSection>` コンポーネントを追加し、`title` 属性の値に「A Section Title」を指定します。
- `<apex:pageBlockSection>` 内に、「I'm three components deep」というテキストを追加します（「I'm three components deep」以外の任意のテキストを追加して構いません）。
- 保存** ボタンをクリックします。コードは次のようになります。

```
<apex:page sidebar="false">
  <apex:pageBlock title="A Block Title">
    <apex:pageBlockSection title="A Section Title">
      I'm three components deep!
    </apex:pageBlockSection>
  </apex:pageBlock >
</apex:page>
```

ページの表示は次のようになります。



「A Section Title」の横にある小さな三角のボタンをクリックすると、その下のセクションが折りたたまれます。

## ステップ 3: コンポーネントをネストする

このステップでは、コンポーネントをさらに追加する方法を紹介しましょう。手順は簡単です。

- `<apex:pageBlockSection>` コンポーネントの末尾にカーソルを移動し、`<apex:pageBlockSection>` コンポーネントをもう 1 つ追加して別のタイトルを入力します。2 つの `<apex:pageBlockSection>` コンポーネントは、同じ `<apex:pageBlock>` コンポーネントの内側に挿入される必要があります。
- あとは**保存** ボタンをクリックするだけです。じつに簡単な作業です。

ここで、いくつかのコンポーネントのネストの状態を確認してみましょう。2 つの `<apex:pageBlockSection>` の開始タグと終了タグは、`<apex:pageBlock>` コンポーネントの開始タグと終了タグの内側にあります。最初の `<apex:pageBlockSection>` は、2 つ目の `<apex:pageBlockSection>` が開始する前に終了しています（終了タグ `</apex:pageBlockSection>` が次の `<apex:pageBlockSection>` の開始タグの前にあります）。Visualforce ページでは、すべてのコンポーネントがこのような方法でネストされます。タグの指定にエラー（終了タグが欠けているなど）があると、エディタに警告メッセージが表示されます。

## まとめ

このチュートリアルでは、属性を追加して Visualforce コンポーネントの動作と外観を変更する方法、入力支援機能を使用する方法、コンポーネントレファレンスを表示して追加コンポーネントを参照する方法、Visualforce のコンポーネントでよく使用されるネストの構造などについて学びました。

## 補足

`<apex:pageBlock>` に関連するその他の Visualforce コンポーネントを、いくつか紹介します。

- `<apex:pageBlockButtons>` は、標準のユーザインターフェースのボタンに似たスタイルのボタンのセットを提供します。
- `<apex:pageBlockSectionItem>` は、`<apex:pageBlockSection>` 内の 1 件のデータを表すオプションとして使用できます。
- `<apex:tabPanel>`、`<apex:toolbar>`、`<apex:panelGrid>` は、ページに表示する情報のグループ化に使用できます。

## チュートリアル 3: シンプルな変数と数式

これまで作成した Visualforce ページは静的なものでしたが、Visualforce ページは通常、データベースから取得したデータを反映する、ログインするユーザーに応じて異なるデータを表示するなど、動的な性質を備えています。動的な Visualforce ページを作成するには、変数と数式を使用します。

このチュートリアルでは、Visualforce で使用する変数、数式、および数式言語の構文について解説します。変数には通常、Force.com データベース内のオブジェクトから取得した情報 (Force.com プラットフォームでユーザーが利用できる情報) が格納されます。例としては、ログインユーザー名を取得する変数などがあります。Visualforce には、ページに機能を追加するための組み込みの数式が豊富に用意されており、このチュートリアルでも、複数の基本的な数式を使用しています。

### ステップ 1: グローバル変数

Force.com は、ログインしたユーザーの情報を `User` という変数に保持しています。この `User` 変数の項目にアクセスして値を取得するには、`{!$<グローバル変数>.<項目名>}` という専用の数式言語構文を使用します (`User` 変数以外の変数でも同様)。

1. 作成した Visualforce ページに `{! $User.FirstName}` という行を追加します。なお、ページに表示するコンテンツは必ず `<apex:page>` コンポーネント内 (開始タグと終了タグの間) に含めるように注意してください。
2. 保存ボタンをクリックします。

Page Editor のコードは次のようになります。

```
<apex:page sidebar="false">
  {! $User.FirstName}
</apex:page>
```

皆さんは今後、`<apex:page>` タグ内であらゆる Visualforce マークアップを使って自由にページを作成できるようになるでしょう。保存ボタンをクリックすると編集結果を確認できる Visualforce の機能は、便利にお使いいただけると思います。

`{! ... }` は、中かっこの内側がすべて動的な要素であり、数式言語で記述されることを示しています。値は、ユーザーがページを表示する実行時に計算され、送信される必要があります。なお、Visualforce では大文字小文字は区別されず、`{! ... }` 構文内のスペースも無視されます。たとえば、`{! $User.firstname }` のようにスペースが混じっていても影響はありません。

ログインユーザーの姓と名を表示する場合は、`{! $User.LastName} {! $User.FirstName}` と入力します。

### ステップ 2: 基本数式

Visualforce では、変数以外の要素も数式言語に含めることができます。また、ユーザーが値を操作するための数式もサポートされます。では、文字列を連結する数式言語の演算子である「&」を使ってみましょう。

1. Page Editor の編集画面で「`{! $User.firstname & ' ' & $User.lastname}`」という行を追加してみてください。

この構文は、グローバル変数である `User` オブジェクトから `Firstname` 項目と `Lastname` 項目を取得し、両者を空白文字で結合することを示します。ページ上では、「Joe Blogs」のように表示されます。

通常は、数式ではもう少し高度な機能を使いますが、常に関数名、かっこのペア、オプション指定のパラメータから成る単純な構文をとるという点に変わりはありません。

2. 続いて、次の構文を追加してみてください。

```
<p> Today's Date is {! TODAY() } </p>
<p> Next week it will be {! TODAY() + 7 } </p>
```

Visualforce ページ上の表示は次のようになります。

```
Today's Date is Mon Aug 09 00:00:00 GMT 2010
Next week it will be Mon Aug 16 00:00:00 GMT 2010
```

<p> タグは段落の区切りを示す標準の HTML タグです。上記の 2 つの文は、このタグによって 1 行ではなく 2 つの独立した行として表されます。TODAY() 関数は、現在の日付を日付のデータ型で返します。時間の値がすべて 0 として返されている点に注意してください。なお、2 つ目の構文では日付の関数に + 演算子が追加されています。これは所定の日数の加算として解釈され、現在の日付に 7 日を加えた値が返されます。

3. 関数を別の関数のパラメータとして使用することや、関数に複数のパラメータをとることなども可能です。さらに、次のようなコードを追加してみてください。

```
<p>The year today is {! YEAR(TODAY())}</p>
<p>Tomorrow will be day number {! DAY(TODAY() + 1)}</p>
<p>Let's find a maximum: {! MAX(1,2,3,4,5,6,5,4,3,2,1)} </p>
<p>The square root of 49 is {! Sqrt(49)}</p>
<p>Is it true? {! CONTAINS('salesforce.com', 'force.com')}</p>
```

Visualforce ページ上の表示は次のようになります。

```
The year today is 2010
Tomorrow will be day number 10
Let's find a maximum: 6
The square root of 49 is 7.0
Is it true? true
```

CONTAINS() 関数はブール値 (true または false のいずれか) を返します。2 つのテキストの引数を比較し、最初の引数に 2 番目の引数が含まれている場合は true を返し、含まれていない場合は false を返します。上記の例では、「force.com」の文字列が「salesforce.com」に含まれているため、true が返されています。

## ステップ 3: 条件式

ここで、数式の値に応じて情報の表示を動的に変更することが必要になるシナリオを考えてみましょう。例としては、請求書の項目がないときに、空の明細行を表示する代わりに「なし」と表示したい場合や、支払期限が過ぎたときに、期日を示す代わりに「遅延」と表示したい場合などが考えられます。

Visualforce ページ上では、IF() をはじめとする条件式を使用することで、こうしたシナリオに対応できます。IF() 式は次の 3 つの引数をとります。

- 1 つ目は、ステップ 2 の CONTAINS() 関数の例で説明したブール値 (true または false) です。
- 2 つ目は、ブール値が true のときに返される値です。
- 3 つ目は、ブール値が false のときに返される値です。

では、Page Editor の編集画面に次のような条件式を追加してみましょう。保存を実行する前に、Visualforce ページ上でどのような表示になるか予想してみてください。

```
{! IF ( CONTAINS('salesforce.com','force.com'), 'Yep', 'Nah') }
{! IF ( DAY(TODAY()) > 14, 'After the 14th', 'Before the 14th') }
```

実際のページの表示は次のようになります。

```
Yep
Before the 14th
```

もちろん、コードが実行される日付によってこの表示は変わります。その月の 14 日を過ぎた場合、これとは異なるテキストが表示されます。

## まとめ

Visualforce では、専用の数式言語の構文である `{! expression}` を使用して、実行時に値を評価する処理を埋め込むことができます。グローバル変数には、`{! $<グローバル変数>.<項目名>}` という構文を使用してアクセスします。数式言語を使用すると、文字列、数値、テキスト、日付などの表示を操作したり、条件に応じて異なる処理を実行したりできます。

## 補足

- 「[数式早見表](#)」は、Visualforce で利用できるさまざまな数式をコンパクトにまとめたガイドです。
- 「[Visualforce Developer's Guide](#)」(英語) では、数式の詳細な説明を確認できます。

## チュートリアル 4: 標準コントローラ

Visualforce の MVC (Model-View-Controller: モデル-ビュー-コントローラ) フレームワークでは、表示やスタイルを、基盤となるデータベースとロジックから簡単に分離することができます。MVC では、ビュー (Visualforce ページ) でコントローラを操作します。Visualforce の場合、コントローラは通常 Apex クラスであり、このクラスがページに機能を追加する役割を果たします。たとえば、コントローラにはボタンをクリックしたときに実行されるロジックが実装されています。また、多くの場合、コントローラはビューに表示するデータを提供するモデル (データベース) とのやり取りも行います。

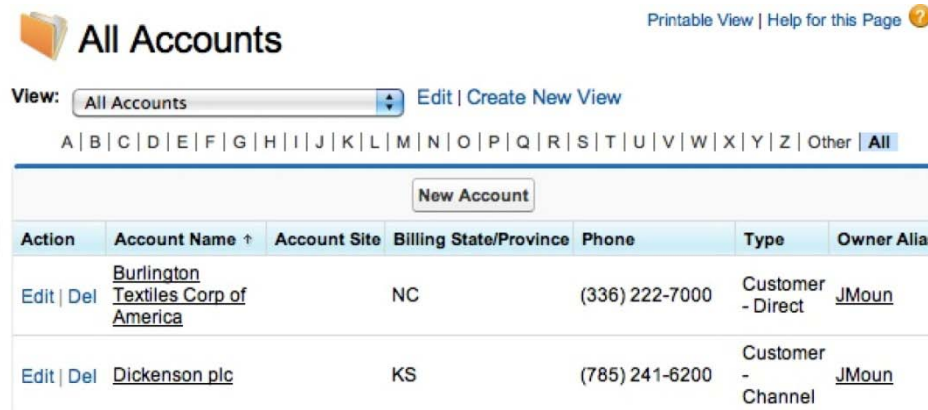
大部分の Force.com オブジェクトには、デフォルトで、オブジェクトに関連付けられたデータを操作するための標準コントローラが用意されており、ほとんどの場合、コントローラのコードを自分で記述しなくても済むようになっています。もちろん、必要な場合には、標準コントローラを拡張して新たな機能を追加したり、カスタムコントローラを作成したりすることができます。このチュートリアルでは、標準コントローラについて学びます。

### ステップ 1: レコードの ID を取得する

Visualforce ページをアプリケーションの他のページに連携させる場合、レコードの ID が自動的に渡されるようにすることで、Visualforce ページにそのページのデータを表示することができます。これまでのチュートリアルで作成したページは単体で実行されているため、データベースのレコードのデータをページに表示するには、レコードの ID 情報が必要になります。

Developer Edition 環境では、そのまま使用できる、データ格納済みのオブジェクトが多数用意されています。ここでは、取引先オブジェクトを例に、レコードの ID を取得してみましょう。


1. 右上のドロップダウンリストから **[セールス]** を選択して、セールスアプリケーションに切り替えます。
2. **[取引先]** タブを選択します。ビュー名が **[すべての取引先]** になっていることを確認し、**[Go!]** をクリックして取引先レコードの一覧を表示します。



The screenshot shows the 'All Accounts' page in Salesforce. At the top, there is a navigation bar with 'Printable View | Help for this Page'. Below that, the page title 'All Accounts' is displayed. A 'View:' dropdown menu is set to 'All Accounts', with 'Edit | Create New View' options. Below the view menu is a navigation bar with letters A through Z and 'Other | All'. A 'New Account' button is located above the table. The table has the following columns: Action, Account Name, Account Site, Billing State/Province, Phone, Type, and Owner Alias. Two rows of data are visible:

Action	Account Name ↑	Account Site	Billing State/Province	Phone	Type	Owner Alias
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">Burlington Textiles Corp of America</a>		NC	(336) 222-7000	Customer - Direct	<a href="#">JMoun</a>
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">Dickenson plc</a>		KS	(785) 241-6200	Customer - Channel	<a href="#">JMoun</a>

3. 取引先名 **[Burlington Textiles]** (別の名前を選択してもかまいません) のリンクをクリックすると、その取引先の詳細情報が次のような画面に表示されます。


Account  
 **Burlington Textiles Corp of America**  
 Customize Page | Edit Layout | Printable View | Help for this Page

Show Chatter New! Unfollow

« Back to List: Accounts

[Contacts \[1\]](#) | [Opportunities \[1\]](#) | [Cases \[2\]](#) | [Open Activities \[0\]](#) | [Activity History \[0\]](#) | [Notes & Attachments \[0\]](#) | [Partners \[0\]](#)

**Account Detail** Edit Delete Include Offline

Account Owner	 <a href="#">Jon Mountjoy [Change]</a>	Rating	Warm
Account Name	Burlington Textiles Corp of America <a href="#">[View Hierarchy]</a>	Phone	(336) 222-7000
Parent Account		Fax	(336) 222-8000
Account Number	CD656092	Website	<a href="http://www.burlington.com">http://www.burlington.com</a>
Account Site		Ticker Symbol	<a href="#">BTXT</a>
Type	Customer - Direct	Ownership	Public
Industry	Apparel	Employees	9,000
Annual Revenue	€350,000,000	SIC Code	546732
% Closed	1	% Pipe + Lost	0.00

ここで、ブラウザのアドレスバーの URL が、<https://<利用中の Salesforce インスタンス>.salesforce.com/0018000000MDfn1> のように変わっていることを確認してください。

この URL の末尾にある「0018000000MDfn1」がレコードの ID です。Force.com データベース内のすべてのレコードは一意の ID を持ちます。あるレコードの ID がわかっているユーザがレコードに対するアクセス権限を持っている場合は、そのレコードの ID を URL に追加することでレコードにアクセスできます。

<https://<利用中の Salesforce インスタンス>.salesforce.com/0018000000MDfn1> へアクセスすると、Force.com は自動的に 0018000000MDfn1 の ID を持つレコードをデータベースから取得し、そのレコードのユーザインターフェースを作成します。なお、手動でユーザインターフェースを作成する方法については、後続のチュートリアルで学んでいきます。

## ステップ 2: レコードのデータを取得して表示する

「accountDisplay」という名前で作成した新しい Visualforce ページを作成して、次のようなコードを記述します。

```
<apex:page standardController="Account">
  Hello {! $User.FirstName}!
  <p>You are viewing the {! account.name} account.</p>
</apex:page>
```

前のチュートリアルで学習した数式構文 `{! }` が使われています。`$User.FirstName` は、グローバル変数 `User` の `Firstname` 項目を参照します。今回のコードでは、そのほかに、次のような新しい要素が使用されています。

- standardController="Account" 属性:** この属性は、取引先 (Account) オブジェクトに対して自動作成済みの標準コントローラを使用するよう Visualforce に指示します。これにより、作成する Visualforce ページでこのコントローラを利用できるようになります。
- {! account.name} 式:** この式では、`account` 変数の `name` 項目の値を取得します。`account` 変数は標準コントローラによって自動的に使用可能になります (変数の値は標準コントローラの名前に準じます)。

通常、コントローラにはボタンのクリックに応じてデータベースを操作するロジックが実装されています。`standardController` 属性を指定することで、自動的に提供される高機能なコントローラを Visualforce ページから使用できるようになります。

取引先オブジェクトの標準コントローラは、ID がページで使用されるタイミングを特定します。ID がページで使用されると、データベースにクエリを行い、その ID に関連付けられたレコードを取得して `account` 変数に割り当てます。このようにして、Visualforce ページで必要なデータを利用できるようになります。

ここで**保存**ボタンをクリックしてみます。自分の名前が表示されますが、取引先名の部分は空になっています。これはどの取引先レコードを取得するかを指定していないためです。URL を編集して、ステップ 1 で確認したレコード ID を追加しましょう。現在の URL は以下のようになっています。

```
https://na3.salesforce.com/apex/AccountDisplay
```

これを変更して、次のようにレコードの ID を追加します。

```
https://na3.salesforce.com/apex/AccountDisplay?id=0018000000MDfn1
```

「0018000000MDfn1」の部分は、ステップ 1 で確認した ID に置き換えてください。また、「na3」の部分には、現在利用中の Salesforce インスタンス名が入ります。

保存ボタンをクリックすると、次のように取引先名が表示されます。

```
Hello Jon!
You are viewing the Burlington Textiles Corp of America account.
```

## ステップ 3: 他の項目を表示する

現在、`accountDisplay` ページには、取引先オブジェクトの名前 (Name) 項目しか表示されていません。その他の項目も表示してみましょう。[<あなたの名前>] > [設定] > [カスタマイズ] > [取引先] > [項目] の順にクリックして、オブジェクトの項目のリストを表示します。[株式コード] という項目をクリックして詳細を確認します。[項目名] の値が「TickerSymbol」となっています。Visualforce ページでは、この [項目名] の値を使用して、項目を表示します。

`accountDisplay` ページの編集画面に戻り、現在のコードに次の行を追加します。これにより、[株式コード] 項目の値も表示されるようになります。

```
<p>Here's the Ticker Symbol field: {! account.TickerSymbol}</p>
```

## ステップ 4: リレーションが設定されたレコードの項目を表示する

続いて、リレーションが設定されたレコードのデータをページに表示できるようにしてみましょう。取引先レコードの詳細ページに [取引先 所有者] という項目があります。ステップ 3 で参照した取引先オブジェクトの項目のリストを確認すると、この項目のデータ型は [参照関係 (ユーザ)] となっています。これは、この項目がユーザレコードとリレーションを持つことを示しています。続いて、項目の名前の上でクリックして [項目名] を確認します。値は [Owner] です。さらに、[カスタマイズ] > [ユーザ] > [項目] の順にクリックして、ユーザオブジェクトの項目のリストを表示します。[名前] という項目があり、[項目名] の値が [Name] であることも確認します。

1. 以上の情報を使って `accountDisplay` ページを編集します。現在のコードに次の行を追加します。

```
<p>Here's the Owner of this account: {! account.owner.name}</p>
```

「`account.owner.name`」のドット (.) 表記は、各レコードが参照関係にあることを示しています。「`account.owner`」は、取引先レコードの [取引先 所有者] 項目を示します。「`owner`」の後に「`name`」が続いているのは、Owner 項目が単純に文字列を格納する項目ではなく、別のレコードを参照して値を取得する項目であるためです (これは、先ほど確認し

たデータ型 [(参照関係(ユーザ))] に対応します)。accountDisplay ページが [取引先 所有者] 項目の値を表示するには、その項目の参照先となるユーザレコードの [名前] 項目の値を参照する必要があります。



**メモ:** 取引先オブジェクトなどの標準オブジェクトではなく、独自に作成したカスタムオブジェクトの項目を参照したい場合は、上記とは異なる手順を実行する必要があります。[<あなたの名前>] > [設定] > [作成] > [オブジェクト] の順にクリックして対象となるカスタムオブジェクトを選択し、項目の詳細を確認します。カスタムオブジェクトの場合、Visualforce ページでの設定で使用する項目の名前は、[API 参照名] 項目に表示されます。たとえば、「Foo」という項目の API 参照名が「Foo\_\_c」である場合、コード内で「{! <カスタムオブジェクト名>.foo\_\_c}」のように指定を行えば、項目を参照できます。

## まとめ

標準コントローラでは、特別な設定を行うことなく、レコードの自動表示をはじめとするコントローラの基本機能を使用できます。このチュートリアルでは、レコードの ID の確認方法、標準コントローラを使用したレコードの表示方法などを学びました。なお、標準コントローラは、レコードの保存、更新といったさまざまな機能を備えています。これらについては、後続のチュートリアルで説明します。

## 補足

Visualforce は、[標準リストコントローラ](#)もサポートしています。このコントローラでは、ページ送り機能を含む Visualforce ページによってレコードのリストを扱うことができます。

## チュートリアル 5: 標準のユーザインターフェースコンポーネント

「チュートリアル 2: Visualforce の属性、入力支援機能、コンポーネント」では、<apex:pageBlockSection> コンポーネントについて学び、チュートリアル 4 では、数式言語を使用して取引先レコードのデータを表示する方法を学習しました。このチュートリアルでは、新たな Visualforce コンポーネントを追加して、外観や動作がデフォルトのユーザインターフェースに類似したインターフェースを作成します。

### ステップ 1: 特定のレコードに関する詳細と関連リストを表示する

特定のレコードに関する詳細は、1 つのコンポーネントを記述するだけで簡単に取得できます。

1. accountDisplay ページを編集して、次のコードを記述します。

```
<apex:page standardController="Account">
  <apex:detail/>
</apex:page>
```

前のチュートリアルで説明した手順にもとづいて、有効なレコード ID をパラメータに追加した URL を指定します (例: <https://na3.salesforce.com/apex/AccountDisplay?id=0018000000MDfn1>)。すると、ページにレコードの詳細情報が表示されます。

2. <apex:detail> コンポーネントは、レコードの標準の参照用ページを表示します。ここには、取引先責任者などの関連リストも含まれます。こうした関連リストを非表示にするには、<apex:detail> に `relatedList="false"` 属性を追加します。追加後、**保存** ボタンをクリックして、ページの表示がどのように変わるか確認してみてください。
3. 特定の関連リストだけを表示することもできます。たとえば、表示中の取引先レコードに関連付けられたケースレコードの関連リストのみを表示する場合は、次のようなタグを追加します。

```
<apex:relatedList list="Cases" />
```

Action	Case	Contact Name	Subject
Edit   Cls	00001019	Jack Rogers	Structural failure of generator base
Edit   Cls	00001020	Jack Rogers	Power generation below stated level

これらのタグは非常にシンプルなものですが、標準コントローラを使用してデータにアクセスしたり、データを取得したりする場合に、さまざまな処理を実行できます。ただし、ページレイアウトの細かい調整が追加で必要になることもあります。

### ステップ 2: 表示する項目を切り替える

<apex:outputField> コンポーネントを使用すると、レコードでどの項目の値を表示するかを個別に指定できます。このコンポーネントを <apex:pageBlock> コンポーネントにネストすると、項目の値と一緒に項目名も表示されるようになります。

1. 次のコードを追加して、ページ上の表示を確認します。

```
<apex:pageBlock title="Custom Output">
  <apex:pageBlockSection title="Custom Section Title" >
```

```
<apex:outputfield value="{!account.name}"/>
<apex:outputfield value="{!account.owner.name}"/>
</apex:pageBlockSection>
</apex:pageBlock>
```

account.name では、現在の取引先レコードの [取引先名] 項目の値を取得します。一方、account.owner.name では、取引先レコードの [取引先 所有者] 項目を参照し、そのレコードのユーザ名を取得します。

Custom Output	
▼ Custom Section Title	
Account Name	Burlington Textiles Corp of America
Name	Jon Mountjoy

[Custom Section Title] の横にある三角ボタンをクリックすると、セクションを折りたたんだり、展開したりできます。

## ステップ 3: テーブルを表示する

これまでのステップでは、特定の項目の 1 件のレコードの情報をそのまま取得していました。では、多数のレコードにまたがって複数の項目の情報を取得したい場合にはどうすればよいでしょう。今回のステップでは、「取引先に関連付けられた取引先責任者のリストを表示する」というシナリオを使って、その手順を紹介합니다。取引先責任者のリストを反復処理してレコードを個別に表示するというやり方で、1 件の取引先レコードに関連付けられた複数の取引先責任者レコードのリストを作成します。複数のタグがネストされるためわかりにくく感じるかもしれませんが、すべてのステップを完了すれば、記述内容の意味を理解できるでしょう。各コンポーネントの詳細を確認したいときは、[Component Reference] (コンポーネントリファレンス) リンクをクリックしてみてください。

1. Page Editor で <apex:pageBlock> コンポーネントを追加します。

```
<apex:pageBlock title="My Account Contacts">
</apex:pageBlock>
```

2. この時点で結果を確認したい場合は、保存ボタンをクリックしてください。続いて、このコンポーネントの内側に <apex:pageBlockTable> コンポーネントをネストします。

```
<apex:pageBlockTable value="{! account.contacts}" var="item">
</apex:pageBlockTable>
```

このコンポーネントは、value 属性をチェックし、指定されているレコードのリストを取得します。今回は、取引先レコードと、その取引先レコードに関連付けられた取引先責任者レコードのリストを表す contacts 項目を取得します。次に、個々の取引先責任者レコードを、item という変数に割り当てます。この処理が、リストの最後のレコードに到達するまで反復されます。ここでは、反復処理の結果が出力されるコンポーネントの本文が重要になります。その出力結果により、各取引先レコードに対して任意の処理を効率よく行うことが可能になります。

3. では、<apex:pageBlockTable> コンポーネントの内側に、取得したアイテムに対する処理を行うコンポーネントを挿入してみましょう。次のコードを追加します。

```
<apex:column value="{! item.name}"/>
```

<apex:column> は、テーブルに新しい列を追加するコンポーネントです。項目の名前にもとづくヘッダーを自動的に作成した後、項目の値を列に挿入します。ここでは、{! item.name} を指定することにより、処理中のアイテム (反復処理されている取引先責任者レコード) の Name (名前) 項目を表示します。

My Account Contacts	
Name	Jack Rogers

最終的なコードは次のようになります。

```
<apex:relatedList list="Cases" />
  <apex:pageBlock title="My Account Contacts">
    <apex:pageBlockTable value="{! account.contacts}" var="item">
      <apex:column value="{! item.name}"/>
    </apex:pageBlockTable>
  </apex:pageBlock>
```

取引先責任者レコードには、[電話] (Phone) という項目もあります。電話番号を表示する列も追加してみてください。なお、指定した取引先レコードに関連付けられた取引先責任者レコードが存在しない場合や、取引先レコードの ID を指定していない場合は、何も表示されません。

## まとめ

`<apex:detail>` コンポーネントと `<apex:relatedList>` コンポーネントを使用すると、標準コントローラを使って特定のレコードのデータを自動で取得し、そのレコードと関連リストを簡単に表示することができます。`<apex:pageBlockTable>` では、レコードのリストに対して反復処理を実行することで、リストに含まれるレコードの出力を生成できます。

## 補足

- `<apex:facet>` コンポーネントを使用すると、テーブルのキャプション、ヘッダー、フッターをカスタマイズできます。
- `<apex:enhancedList>` コンポーネントと `<apex:listViews>` コンポーネントを使用すると、オブジェクトのレコードの標準のリストビューを埋め込むことが可能になります。

## チュートリアル 6: ページの上書きと参照

Visualforce では、ボタン、タブ、リンクなどのさまざまなユーザインターフェースを上書きすることができます。

このチュートリアルでは、作成した Visualforce ページで標準の Salesforce ユーザインターフェースの動作を変更する方法を説明します。

### ステップ 1: ページのデフォルトの表示を上書きする

「チュートリアル 4: 標準コントローラ」で作成した Visualforce ページを使って、取引先レコードの標準の詳細ページを上書きしてみます。Salesforce のデフォルトのページの代わりに、今回作成した Visualforce ページが表示されるようにしてみましょう。

1. [<あなたの名前>] > [設定] > [カスタマイズ] > [取引先] > [ボタンとリンク] の順にクリックします。
2. [標準ボタンと標準リンク] リストで、[参照] の横にある [編集] リンクをクリックし、[プロパティの上書き] ページを開きます。
3. [上書き手段] で、[Visualforce ページ] を選択します。
4. [Visualforce ページ] のドロップダウンリストから、[AccountDisplay] を選択します。
5. [保存] ボタンをクリックします。

変更結果を確認するには、[取引先] タブをクリックし、任意の取引先レコードを選択します。これまでのステップで、取引先レコードの ID を自動的に Visualforce ページに渡すような設定が適切に行えていれば、標準のページの代わりに作成した Visualforce ページが表示されます。

6. このあとのステップのために、[プロパティの上書き] ページに戻り、上書きしたページをデフォルトに戻しておきます。

### ステップ 2: 標準のページレイアウトに Visualforce ページを埋め込む

作成した Visualforce ページを表示するもう 1 つの方法は、他のページの標準のページレイアウト内に Visualforce ページを埋め込むことです。たとえば、取引先に関する興味深いデータが含まれている accountDisplay ページを取引先レコードの詳細ページに埋め込んで表示したい場合には、以下の手順を実行します。

1. [<あなたの名前>] > [設定] > [カスタマイズ] > [取引先] > [ページレイアウト] の順にクリックします。
2. [Account Layout] の横にある [編集] リンクをクリックします。
3. ページ左上で [Visualforce ページ] を選択します。
4. 右側に、[accountDisplay] ページが表示されます (取引先の標準コントローラを用いてページが表示されます)。
5. [accountDisplay] を [取引先情報] パネルヘドラッグ & ドロップします。

6. [保存] をクリックします。
7. 結果を確認します。[取引先] タブに移動して任意の取引先レコードをクリックします。デフォルトのページ内に Visualforce ページが埋め込まれています。単に追加しただけなのでレイアウト的には改善が必要かもしれませんが、今回は、Visualforce ページにデフォルトのページのレコードに対応するデータが自動的に表示されていること、ID パラメータが渡されていることを確認できれば問題ありません。

## ステップ 3: Visualforce ページを表示するボタンを作成する

これまでのステップで取り上げた取引先レコードの詳細ページのような標準のページには、[編集]、[削除]などのボタンが組み込まれています。このステップでは、作成した Visualforce ページを表示する新しいボタンを標準のページに追加してみましょう。

1. [<あなたの名前>] > [設定] > [カスタマイズ] > [取引先] > [ボタンとリンク] の順にクリックします。
2. [カスタムボタンとカスタムリンク] で、[新規] をクリックします。
3. [表示ラベル] と [名前] に値を入力します。
4. [表示の種類] で [詳細ページボタン] を選択します。
5. [内容のソース] 選択リストで [Visualforce ページ] を選択します。
6. [コンテンツ] 選択リストが表示されます。accountDisplay ページを選択して、[保存] をクリックします。

Account Custom Button or Link

### New Button or Link

7. 新しいカスタムボタンを表示するにはページレイアウトに追加を行う必要があるというメッセージが表示されたら、[OK] をクリックして確定し、ページレイアウトへの追加を実行します。ステップ 2 の 1 ~ 2 の手順にしたがってページレイアウト [Account Layout] の編集画面を開き、ページ左上で [ボタン] を選択した後、右側のリストから先ほど作成したボタンをドラッグ & ドロップして [カスタムボタン] エリアに追加します。なお、リンクを作成する場合も、ボタンと同様の手順で実行できます。

## ステップ 4: Visualforce ページから他の Visualforce ページや外部 URL を参照するハイパーリンクを作成する

Visualforce ページから他の Visualforce ページや外部 URL を参照するハイパーリンクを作成するには、次の手順を実行します。

1. ハイパーリンクを作成するには、Visualforce ページの Page Editor で `<apex:outputlink>` コンポーネントを追加します。外部 URL を参照する場合は、次のように指定します。

```
<apex:outputLink value="http://developer.force.com/">Click me!</apex:outputLink>
```

2. ページを参照する場合は、ページの URL を特定する式 `{! $Page.<ページ名>}` を設定します。
3. たとえば、`accountDisplay` ページを参照する場合は、次のように指定します。

```
<apex:outputLink value="{! $Page.AccountDisplay}">I am me!</apex:outputLink>
```

`$Page` は、作成したあらゆるページの任意の項目を格納するグローバルオブジェクトとみなすことができます。

## まとめ

作成した Visualforce ページは、さまざまな方法で表示できます。単にその URL を指定して表示する以外に、標準のページと置き換える、標準のページのレイアウト内に埋め込む、Visualforce ページを表示するボタンやリンクを埋め込むといった方法があります。

## 補足

- サイト機能 (Force.com Sites) を利用して Visualforce ページを公開 Web サイトに表示することもできます。詳細は、「[Force.com Workbook](#)」で関連チュートリアルを参照してください。
- 取引先レコードの新規作成といったデフォルトの動作にリンクを埋め込むには、次の例のように、`<apex:outputLink>` コンポーネント、`URLFOR()` 関数、グローバル変数 `$Action` を使用します。

```
<apex:outputlink value="{! URLFOR($Action.Account.new)}">Create</apex:outputlink>
```

# チュートリアル 7: カスタムのユーザインターフェースコンポーネント

これまでのチュートリアルでは、Force.com が標準で提供するデフォルトのページと同様の外観、動作を備えた Visualforce ページを取り上げてきましたが、Visualforce には、スタイルが設定されていないページを作成するコンポーネントもあります。こうしたコンポーネントでは、HTML や CSS の標準的なテクニックを活用してスタイルを設定できます。このチュートリアルでは、ページのスタイルを柔軟にコントロールできるコンポーネントの使用方法を学びます。

## ステップ 1: ヘッダーとサイドバーを削除する

「styled」という名前で新しい Visualforce ページを作成し、次のコードを記述します。

```
<apex:page standardController="Account">
  <h1>My Styled Page</h1>
  Great!
</apex:page>
```

<apex:page> コンポーネントに以下の属性を追加します。なお、属性を 1 つ追加したら、その都度**保存**ボタンをクリックして、結果を確認するようにしてください。

1. sidebar="false"
2. showHeader="false"

以上により、ヘッダーとサイドバーが削除された、よりシンプルなページが作成されます。Force.com のデフォルトのスタイル設定をすべて削除するために、さらに次の属性を追加します。

3. standardStylesheets="false"

これにより、Force.com のデフォルトのスタイル設定がすべて削除されました。

## ステップ 2: CSS で HTML のスタイルを設定する

標準の Web テクニックを使用し、カスタムの CSS を追加してページのスタイルを設定します。

1. <apex:page> タグの後に、次のコードを追加します。

```
<style>
  body {font-family: Arial Unicode MS;}
  h1 {color:red;}
</style>
```

# My Styled Page

Great!

多くの Web デザイナーは、ページ内に CSS を埋め込まず、外部の CSS ファイルを参照させるようにページを設計します。Visualforce でこれを行うには、<apex:stylesheet> コンポーネントを使用します。

2. 先ほど追加した <style> セクション全体を、次のコードで置き換えます。

```
<apex:stylesheet value="http://developer.force.com/workbooks/vfdemo.css"/>
```

なお、スタイルに関する CSS ファイルや関連画像を Force.com 上でホストするには、次の手順を実行します。

3. カスタムの CSS ファイルを zip ファイルに圧縮します (または[サンプルの zip ファイルをダウンロード](#)します)。
4. [[あなたの名前](#)] > [[設定](#)] > [[開発](#)] > [[静的リソース](#)] の順にクリックします。
5. [[新規](#)] をクリックします。
6. [名前] に「Style」と入力します。
7. [ファイル] の [[参照](#)] をクリックして、自分が zip したファイル (または[サンプルの zip ファイル](#)) をアップロードします。
8. Visualforce ページに戻り、CSS ファイルを読み込む `<apex:stylesheet>` コンポーネントの属性を次のように変更します。

```
<apex:stylesheet value="{! URLFOR($Resource.Style, 'demo.css')}" />
```

グローバル変数 `$Resource` は、アップロード対象の静的リソースの項目を格納します。上の例では、この変数は zip ファイルに割り当てた「Style」という名前の静的リソースを参照します。URLFOR() は、URL を生成する数式です。上の例では、URL は静的リソースとそのリソース内のアイテムの名前から生成されます。このように、CSS ファイルを静的リソースとしてアップロードすることにより、Web ページとは切り離れた形でスタイルシートを編集できます。

## ステップ 3: リストとテーブルの反復処理を利用する

「[チュートリアル 5: 標準ユーザインターフェースのコンポーネント](#)」では、`<apex:pageBlockTable>` コンポーネントを使って反復処理を行い、Salesforce の標準の外観と動作に準じた形式でデータを表示しました。Visualforce ではそれとは反対に、シンプルな標準 HTML 形式でデータを表示して、スタイルを自由にカスタマイズできるような反復処理も可能です。このステップではそうした処理を可能にするコンポーネントについて学びます。

1. accountDisplay ページを開き、次のコードを追加します。

```
<apex:dataTable value="{!account.contacts}" var="item">
  <apex:column value="{!item.name}" />
  <apex:column value="{!item.phone}" />
</apex:dataTable>
```

2. ページに何も表示されない場合は、取引先レコードの ID を URL パラメータとして渡していることを確認してください。スタイルが設定されていないため、外観はプレーンなものになりますが、これがポイントです。つまり、標準の CSS を使用して自由に書式を設定できるということです。また、以下のように `<apex:dataList>` コンポーネントを使用すれば、テーブルの代わりに、HTML 標準のリスト項目を作成できます。

```
<apex:dataList value="{!account.contacts}" var="item">
  <apex:outputText value="{!item.name}" />
</apex:dataList>
```

3. 出力を完全にコントロールするには、反復処理のみを実行する `<apex:repeat>` コンポーネントを使用します。ここでは、スタイルをすべて自分で記述することになります。たとえば、`<apex:dataList>` コンポーネントと同様のスタイルを適用するには、次のように記述します。

```
<ul>
  <apex:repeat value="{!account.contacts}" var="item">
    <li><apex:outputText value="{!item.name}" /></li>
  </apex:repeat>
</ul>
```

## まとめ

`showHeader`、`sidebar`、`standardStyleSheets` の各属性を使用すると、Visualforce ページの標準のスタイルを簡単に編集したり削除したりできます。標準のスタイルは、独自の CSS によって置き換えることが可能です。これは、コードに直接埋め込めるほか、静的リソースとして参照させたりすることもできます。`<apex:outputText>` コンポーネントではスタイルの適用されていないテキストを表示できます。`<apex:dataTable>`、`<apex:dataList>`、`<apex:repeat>` の各コンポーネントでは、反復処理によって取得されたデータの出力形式を自由にカスタマイズすることができます。

## 補足

`<apex:dataTable>` などの多くのコンポーネントでは、コンポーネントのスタイルやスタイルクラスを簡単にコントロールできる属性をサポートしています。参考に、「`styleClass`」などの属性を確認してみてください。

## チュートリアル 8: フォームを使ったデータの入力

このチュートリアルでは、標準コントローラを使って、レコードの保存や更新が可能な入力画面をデフォルトで提供する方法を学びます。また、主な入力機能とそのコンテナである `<apex:form>` コンポーネントについても解説します。なお、「チュートリアル 13: カスタムコントローラ」では、ここで取り上げる内容をさらに発展させた、カスタムのコントローラによるフォームの作成方法について説明しています。

### ステップ 1: 基本となるフォームを作成する

フォームは、データ入力にとって重要なポイントです。このステップでは、もっとも基本的なフォームを作成します。

1. 「MyForm」という名前で新しい Visualforce ページを作成します。以下のように、取引先オブジェクトの標準コントローラを使用します。

```
<apex:page standardController="Account">
</apex:page>
```

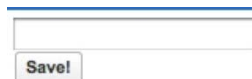
2. `<apex:form>` コンポーネントを挿入します。入力用の項目はすべてこのコンポーネント内に記述します。

```
<apex:form>
</apex:form>
```

3. このフォーム内に、取引先の名前を入力する項目と、クリック操作にもとづいてフォームの保存を実行するボタンを追加します。

```
<apex:inputField value="{! account.name}" />
<apex:commandButton action="{! save}" value="Save!" />
```

このフォームはきわめて基本的なものです。フォーム、入力項目、フォームを操作するボタンという重要な要素をすべて含んでいます。たとえば、`<apex:commandButton>` コンポーネントによってボタンが生成されています。



アクション要素は、`{! save}` にバインドされています。この式言語による構文は、レコード内の項目を指定する際に使った構文に似ていますが、この利用方法では、「save」というコードで表されるメソッドを参照します。すべての標準コントローラは、デフォルトで `save()` メソッドと `update()` メソッドを提供します。上のコードでは、ボタンのクリックにより標準コントローラの `save()` メソッドが呼び出されます。

値を入力して **[Save!]** ボタンをクリックすると、入力した項目の値が、新規レコード内の対応する名前を持つ項目の値にバインドされます。つまり、`<apex:inputField>` コンポーネントにより、新しい取引先レコードの [取引先名] 項目に対応する入力用の項目が作成され、**[Save!]** ボタンのクリックによって、入力した値がそのレコードの [取引先名] 項目に自動的に保存されます。

ボタンをクリックすると、新規作成されたレコードが表示されます。MyForm ページの URL (<https://na6.salesforce.com/apex/MyForm> など) を指定して、先ほどのページに戻ります。

### ステップ 2: フォームに項目の表示ラベルを表示する

Visualforce は、入力項目を新規レコードの項目にバインドしながら、バックグラウンドでさまざまな処理を行います。項目の表示ラベルを自動で表示する（「チュートリアル 5: 標準のユーザインターフェースコンポーネント」で説明する `<apex:outputfield>` と同様の動作）、データ型の一致を確保するために入力形式を自動で変更する（たとえば、テキストボックスの代わりに選択リストを表示するなど）といった処理が可能です。

<apex:form> の内容を、次のコードに差し替えます。

```
<apex:pageBlock>
  <apex:pageBlockSection>
    <apex:inputField value="{!account.name}"/>
    <apex:inputField value="{!account.industry}"/>
    <apex:commandButton action="{!save}" value="Save!" />
  </apex:pageBlockSection>
</apex:pageBlock>
```

<apex:pageBlock> コンポーネントと <apex:pageBlockSection> コンポーネントの内部に入力項目を組み込むと、Visualforce では、その入力項目の表示ラベル ([取引先名]) と、値の入力が必須であることを示すマーク (赤い縦線) を自動的にページに追加します。

## ステップ 3: 警告やエラーを表示する

<apex:pageMessages> コンポーネントを使用すると、そのページを構成するすべてのコンポーネントについて、通知、警告、エラーなどのメッセージを表示できます。前のステップで作成したフォームでは、[取引先名] 項目は必須項目でした。ユーザが取引先名を指定せずにフォームを保存しようとした場合に標準的なエラーメッセージを表示するには、以下の手順を実行します。

1. <apex:pageBlock> タグの後に以下の行を挿入して、ページを更新します。

```
<apex:pageMessages />
```

2. ここで、フォームの [Save!] ボタンをクリックすると、次のようなエラー画面が表示されます。

## まとめ

Visualforce の標準コントローラには、レコードの保存や更新を簡単に実行できるメソッドが用意されています。<apex:form> コンポーネントと <apex:inputField> コンポーネントを組み合わせて使用すると、標準コントローラを使って入力項目を新しいレコードにバインドできます。ユーザインターフェースでは、項目のデータ型に対応する入力形式が自動的に適用されます (たとえば、日付データ型の項目では、カレンダーを使って日付を指定します)。ページ内のすべてのコンポーネントを対象に、通知、警告、エラーなどのメッセージを表示するには <apex:pageMessages> コンポーネントを使用します。

## 補足

- フォーム内で <apex:commandButton> の代わりに、<apex:commandLink> コンポーネントを指定すると、フォームの処理に使用するリンクを作成することができます。

- `save()` メソッドの代わりに `quicksave()` メソッドを使用すると、新規レコードのページを表示せずにレコードを挿入したり、既存のレコードを更新したりすることができます。
- `<apex:pageBlock>` コンポーネントとともに `<apex:pageBlockButtons>` コンポーネントを使用すると、ボタンをページに追加できます。
- `<apex:pageMessage>` コンポーネントでは、カスタムメッセージを作成することができます (複数形の `<apex:pageMessages>` コンポーネントとは異なりますので注意してください)。

## チュートリアル 9: テンプレートを使用したページの再利用

多くの Web サイトには、バナーやサイドバーなど、すべてのページに共通して表示されるデザイン要素があります。Visualforce では、基盤となるテンプレートを作成することで、Web ページと同じように、多様なコンテンツを含む Visualforce ページに標準の構成を適用することが可能です。各ページでは、テンプレート内のプレースホルダをさまざまなコンテンツで置き換えることができます。

### ステップ 1: テンプレートを作成する

テンプレートは、プレースホルダのテキストの挿入位置を指定する特殊なタグを含む Visualforce ページです。このステップでは、シンプルなテンプレートを作成します。挿入位置を指定するタグとして、`<apex:insert>` コンポーネントを使用します。

1. 「BasicTemplate」という名前で新しい Visualforce ページを作成します。
2. コードを以下のように変更します。

```
<apex:page>
  <apex:stylesheet value="http://developer.force.com/workbooks/vfdemo2.css" />
  <h1>My Fancy Site</h1>
  <apex:insert name="body" />
</apex:page>
```

ここでのポイントは、`<apex:insert>` コンポーネントです。このテンプレートは直接アクセスして使用するものではなく（そのように設定することも可能ですが）、複数の Visualforce ページにテンプレートを埋め込み、各 Visualforce ページで `<apex:insert>` コンポーネントにさまざまな値を挿入する、という形で運用します。なお、`<apex:insert>` コンポーネントを使用する場合は、必ず `name` 属性を指定します。上のコードでは、`body` という 1 つの挿入位置が指定されていますが、挿入位置を複数指定することも可能です。

### ステップ 2: テンプレートを使用する

このステップでは、新しいページにテンプレートを埋め込み、1 つずつ値を入力していきます。

1. 「MainPage」という名前で新しい Visualforce ページを作成します。
2. Page Editor で、コードを以下のように変更します。

```
<apex:page sidebar="false" showHeader="false">
  <apex:composition template="BasicTemplate">
    <apex:define name="body">
      <p>This is a simple page demonstrating that
        this text is substituted, and that a banner is created.</p>
    </apex:define>
  </apex:composition>
</apex:page>
```

`<apex:composition>` コンポーネントは、前のステップで作成した Visualforce テンプレートを取得し、`<apex:define>` コンポーネントは、テンプレートのプレースホルダに値を入力します。同様の手順で複数のページを作成できますが、その際は、コンポーネントは同じものを使用し、プレースホルダ内のテキストを都度変更するようにします。

## ステップ 3: Visualforce ページを他の Visualforce ページに挿入する

あるページのコンテンツを他のページに適用するもう 1 つの方法として、`<apex:include>` コンポーネントを使用する方法があります。テンプレートを使用する場合とは異なり、元のページのコンテンツはそのまま複製され、変更を加えることはできません。

1. 「EmbedsAnother」という名前の新しい Visualforce ページを作成します。
2. Page Editor で、コードを以下のように変更します。

```
<apex:page sidebar="false" showHeader="false">
  <p>Test Before</p>
  <apex:include pageName="MainPage"/>
  <p>Test After</p>
</apex:page>
```

元の MainPage ページがそのまま挿入されます。

## まとめ

テンプレートは、複数の Visualforce ページで共通に使用したいページ要素をカプセル化するのに適した方法です。Visualforce ページで記述する必要があるのは、テンプレートを埋め込み、テンプレート内のプレースホルダのコンテンツを定義する処理だけです。`<apex:include>` コンポーネントを使用して他のページにページを挿入する方法はさらに簡単です。Visualforce でコンテンツを再利用するその他の方法については、「[チュートリアル 10: カスタムコンポーネント](#)」で取り上げます。

## チュートリアル 10: カスタムコンポーネント

これまでのチュートリアルでは、<apex:dataTable> などの標準の Visualforce コンポーネントを使用してきました。Visualforce にはこのようなコンポーネントが豊富に用意されていますが、独自のカスタムコンポーネントを作成する必要がある出てくることもあります。たとえば、マークアップと動作をカプセル化して、複数の Visualforce ページで再利用できるようにしたい場合などです。

「チュートリアル 9: テンプレートを使用したページの再利用」で説明したテンプレートとは異なり、こうしたカスタムコンポーネントでは、独自の属性を使用してページの外観を変更することが可能です。また、コンポーネントの特定のインスタンスに対して実行する、コントローラベースの複雑なロジックを実装することもできます。なお、作成したカスタムコンポーネントは [Component Reference] (コンポーネントリファレンス) ヘルプのリストに自動的に登録されます。カスタムコンポーネントを利用すれば、Visualforce をあらゆる用途に合わせて拡張することが可能になります。

### ステップ 1: シンプルなカスタムコンポーネントを作成する

Visualforce のカスタムコンポーネントは、すべて <apex:component> コンポーネントの内側にネストされます。こうしたカスタムコンポーネントは、通常、名前付きの属性を使用し、その属性の値を編集することで動作や外観を指定します。このステップでは、新しいカスタムコンポーネントを作成し、赤い枠内に含めるコンテンツと枠線の幅を指定する属性を定義します。

1. [あなたの名前] > [設定] > [開発] > [コンポーネント] の順にクリックします。
2. [新規] をクリックします。
3. [表示ラベル] と [名前] に、「boxedText」と入力します。
4. [Visualforce Markup] タブで、次のコードを入力して保存します。

```
<apex:component>
  <apex:attribute name="text"
    description="The contents of the box."
    type="String" required="true"/>
  <apex:attribute name="borderWidth"
    description="The width of the border."
    type="Integer" required="true"/>

  <div style="border-color:red; border-style:solid; border-width:{!
borderWidth}px">
    <apex:outputText value="{! text}"/>
  </div>
</apex:component>
```

このコードでは、データ型の異なる 2 つの属性が定義されています。Visualforce ではさまざまな型がサポートされており、属性に型を設定することで、カスタムコンポーネントを使用する際に使用する値の型を強制できます。

コンポーネントのコード本文には、上に示したように他のコンポーネントや単純な HTML を含めることもできます。また、動的な属性を参照することも可能です。たとえば、{! text} は、コンポーネントの使用時に text 属性に入力された値で置き換えられます。

### ステップ 2: カスタムコンポーネントを使用する

前のステップで作成したコンポーネントは、Visualforce の標準コンポーネントと同じように使用したり、参照したりすることが可能ですが、標準コンポーネントと異なり、コンポーネントの名前には「apex:」ではなく「c:」が使用されます。

1. 「custom」という名前で新しい Visualforce ページを作成します。
2. ページに次のコードを入力します。

```
<apex:page>
  <c:boxedText borderWidth="1" text="Example 1"/>
  <c:boxedText borderWidth="20" text="Example 2"/>
</apex:page>
```

このページは、前のステップで作成したコンポーネントを 2 回参照し、borderWidth 属性にそれぞれ異なる値を指定しています。ページの表示は次のようになります。



カスタムコンポーネントはヘルプにも登録されます。[Component Reference] (コンポーネントリファレンス) をクリックし、下へスクロールすると `<c:boxedText>` が見つかります。

## まとめ

Visualforce のカスタムコンポーネント機能を使用すると、標準の Visualforce コンポーネントと同等のコンポーネントを独自に作成できます。カスタムコンポーネントは [Component Reference] (コンポーネントリファレンス) に登録され、ページの動作や外観を変更するための属性を備え、複数の Visualforce ページで再利用が可能です。また、コントローラを組み込んで Apex の機能を最大限に活用することもできます (「[チュートリアル 13: カスタムコントローラ](#)」を参照)。

## チュートリアル 11: Ajax によるページの部分的な更新

Visualforce では、複雑な JavaScript ロジックを実装せずに Ajax 機能を使用して、ページの部分的な更新などを実行できます。これを行うには、まず動的に更新する必要のある領域を明示的に指定し、その後、`render` 属性を使用してページ内で該当領域の表示を動的に更新します。

### ステップ 1: 動的に更新する領域を明示的に指定する

Visualforce で Ajax を使用する際の標準的なテクニックとして、動的に更新する領域をグループ化して明示的に指定するという方法があります。この際よく使われるのが `<apex:outputPanel>` コンポーネントと、領域を識別する `id` 属性です。

1. 「Dynamic」という名前で作成した新しい Visualforce ページを作成し、次のコードを記述します。

```
<apex:page standardController="Account">
  <apex:pageBlock title="{!account.name}"/>
    <apex:outputPanel id="detail">
      <apex:detail subject="{!$CurrentPage.parameters.cid}"
        relatedList="false" title="false"/>
    </apex:outputPanel>
</apex:page>
```

2. このページの URL に、有効な取引先レコードの ID を渡します。

ページには、取引先の名前以外には何も表示されません。コードを確認してみましょう。`<apex:outputPanel>` には、`id` 属性として `detail` という値が指定されています。`<apex:detail>` には、詳細を表示したいレコードの ID を取得する `subject` 属性に続いて `{!$CurrentPage.parameters.cid}` という式が指定されています。この式はページに渡される CID パラメータを返しますが、今回はそうしたパラメータをまだ渡していないため、何も表示されません。

### ステップ 2: 領域を動的に更新する

このステップでは、パラメータを設定する要素をページに追加し、`detail` という名前を付けた領域を動的に表示します。

1. 前のステップで作成したコードの `<pageBlock>` コンポーネントの下に、以下の `<pageBlock>` コンポーネントを追加します。

```
<apex:pageBlock title="Contacts" >
  <apex:form >
    <apex:dataList value="{! account.Contacts}" var="contact">
      {! contact.Name}
    </apex:dataList>
  </apex:form>
</apex:pageBlock>
```

このコードは、この取引先レコードに関連付けられた取引先責任者レコードのリストを反復処理し、各取引先責任者の名前のリストを作成します。

2. 保存ボタンをクリックします。

ページには、取引先責任者のリストが表示されます。次に、各取引先責任者の名前をクリックできるようにします。

3. `{! contact.Name}` 式を編集します。次のようにして、`<apex:commandLink>` コンポーネントで式を囲みます。

```
<apex:commandLink rerender="detail">
  {! contact.Name}
  <apex:param name="cid" value="{! contact.id}"/>
</apex:commandLink>
```

この `<apex:commandLink>` コンポーネントには、2 つの重要な役割があります。1 つは、`rerender="detail"` 属性を使用して、前のステップの `<apex:outputPanel>` コンポーネントで指定した領域を参照することです。これにより、取引先責任者の名前がクリックされたときに、該当の領域に対する部分的なページ更新を行うよう Visualforce に指示することができます。もう 1 つは、`<apex:param>` コンポーネントを使用してパラメータ (ここでは、取引先責任者レコードの ID) を渡すことです。

ページ上の任意の取引先責任者の名前の上でクリックしてみましょう。ページ全体が更新される代わりに、クリックした領域だけが動的に更新され、取引先責任者レコードの詳細が表示されます。

## まとめ

Visualforce は、ページを部分的に更新する Ajax の機能をネイティブでサポートしています。ポイントは領域を明示的に指定することと、`rerender` 属性を使用してその領域の動的な更新を指示することです。

## 補足

Visualforce では、上記のほかにも、Ajax や JavaScript をサポートするコンポーネントを複数提供しています。

- `<apex:actionStatus>` では、Ajax リクエストの状況を表示できます。進行中であるか完了しているかに応じて、異なる値が表示されます。
- `<apex:actionSupport>` では、任意のコンポーネントに対して Ajax アクションをトリガするユーザアクションを指定できます。たとえば、`<apex:commandLink>` などのコンポーネントを使用することなく、ラベルにマウスを合わせたときのシンプルなロールオーバーなどを有効化できます。
- `<apex:actionPoller>` では、指定された間隔で、Force.com に対して Ajax の更新リクエストを送信するタイマーを設定できます。
- `<apex:actionFunction>` では、Ajax リクエストを使用して、JavaScript コードからコントローラのアクションメソッドを直接呼び出せるようにします。
- `<apex:actionRegion>` では、Ajax リクエストの生成時に Force.com が処理するコンポーネントの領域を明示的に指定することができます。

# チュートリアル 12: コントローラ拡張機能による機能の追加

これまでのチュートリアルでは、レコードの自動取得、保存、更新などの一連の機能を実行する際に標準コントローラを使用してきました。しかし、デフォルト以外の方法でレコードを取得したい場合や、異なる処理を実行したい場合には、標準コントローラを超える機能が必要になることがあります。このような場合には、コントローラ拡張機能を追加します。この拡張機能は、Visualforce ページからアクセス可能な機能を含むカスタムの Apex クラスとして実装します。

## ステップ 1: コントローラ拡張機能を作成する

コントローラ拡張機能は Apex クラスです。Apex の詳細は、「[Force.com Workbook](#)」を参照してください。

1. [**<あなたの名前>**] > [**設定**] > [**開発**] > [**Apex クラス**] の順にクリックします。
2. [**新規**] をクリックします。
3. Apex クラスとして以下のコードを入力し、保存します。

```
public class MyExtension {
    private final Account acct;

    public MyExtension(ApexPages.StandardController controller) {
        this.acct = (Account)controller.getRecord();
    }
    public String getTitle() {
        return 'Account: ' + acct.name + ' (' + acct.id + ')';
    }
}
```

このコードは、MyExtension という名前の Apex クラスを表し、acct という名前のインスタンス変数を使用します。この Apex クラスには、コントローラのパラメータからレコードを取得し、acct 変数に保存するコンストラクタが含まれています。さらに、取引先名と ID を含む文字列を返す getTitle() という名前のメソッドも含まれています。

ほとんどの拡張機能ではこのフォームを使用します。このフォームではコンストラクタが重要な役割を果たします。拡張機能が使用される際、Visualforce により自動的にこのコンストラクタが呼び出され、表示対象のページのコントローラに渡されます。このコントローラによって、表示対象のレコードへのアクセスが提供されます。

## ステップ 2: Visualforce ページに拡張機能を追加する

「[チュートリアル 8: フォームを使ったデータの入力](#)」(28 ページ) では、非常にシンプルなフォームを作成しました。このステップでは、そのフォームを複製して、拡張機能を追加します。

1. 「MyAccountWithExtension」という名前で新しい Visualforce ページを作成します。
2. 次のようなコードを入力します。

```
<apex:page standardController="Account" extensions="MyExtension">
    <apex:form>
        <apex:inputField value="{!account.name}" />
        <apex:commandButton action="{!save}" value="Save!" />
    </apex:form>
    <p>{!title}</p>
</apex:page>
```

3. 有効な取引先レコードの ID をパラメータとして指定し、作成したページにアクセスします。ページの URL は次のようになります。  
<https://na6.visual.force.com/apex/MyAccountWithExtension?id=0018000000MDfn1>

フォームの入力項目には、取引先の名前が自動的に取得されています。また、取引先名と ID から構成されるタイトルも表示されます。



The screenshot shows a form with the following content:

Account: Burlington Textiles Corp of America (0018000000MDfn1AAD)

Burlington Textiles Corp of

Save!

ページを保存すると、[Save!] ボタンの下に新しい行が表示されます。ここでは、以下のような処理が行われています。

- 属性 `extensions="MyExtension"` により、Apex クラス `MyExtension` のコピーがインスタンス化され、現在のコントローラにも参照が渡されます (今回は取引先の標準コントローラに渡されています)。
- 構文 `{! title}` が、Visualforce に対して、`getTitle()` という名前のメソッドを検出してそのメソッドの呼び出し結果を挿入するように指示します。拡張機能内でそのメソッドが検出されたため、Visualforce はメソッドを実行し、実行結果でステートメントを置き換えます。

## まとめ

コントローラの機能を拡張するには、コントローラ拡張機能を Apex クラスとして実装します。この拡張機能により、Visualforce ページから呼び出せるメソッドを追加することができます。Visualforce ページには、複数の拡張機能を実装することができ、同じ拡張機能を複数の Visualforce ページで使用することが可能です。拡張機能は、多数のコントローラ間で追加機能を共有するためのコンテナとしても使用できます。

## 補足

「[Force.com Apex Code Developer's Guide](#)」では、`StandardController` クラスで使用可能なメソッドの詳細を確認できます。ステップ 1 で取り上げた `getRecord()` の他には、`cancel()`、`delete()`、`edit()`、`getId()`、`save()`、`view()` などを使用できます。

## チュートリアル 13: カスタムコントローラ

「チュートリアル 4: 標準コントローラ」では、Visualforce がサポートしている MVC (Model-View-Controller: モデル-ビュー-コントローラ) フレームワークにもとづくユーザーインターフェースの作成について説明しました。一般的に、コントローラは、Visualforce ページ内に表示されるデータを取得するためのものであり、ページアクション (コマンドボタンのクリックなど) への応答として実行するコードが含まれています。これまでは、Force.com が提供する標準コントローラ (設定なしで利用できるロジックのセット) を使用していましたが、このチュートリアルでは Apex コードを記述して独自のカスタムコントローラを作成します。

### ステップ 1: カスタムコントローラを含んだページを作成する

カスタムコントローラは、コントローラ機能拡張を追加した前のチュートリアルと同じように、[<あなたの名前>] > [設定] > [開発] > [Apex クラス] の順にクリックして作成できるほか、Visualforce の Page Editor を使って自動的に作成されるように設定することも可能です。今回は、Page Editor を使った手順を実行します。

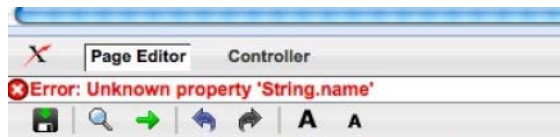
1. 「AccountWithContacts」という名前で、新しい Visualforce ページを作成します。
2. 次のようなコードを入力します。

```
<apex:page controller="MyController">
  <apex:form>
    <apex:dataList value="{! myaccounts}" var="acct">
      <apex:outputText value="{! acct.name}" />
    </apex:dataList>
  </apex:form>
</apex:page>
```

こうしたコードのパターンにはもう慣れたかと思いますが、ここでは、コンポーネントによって取引先レコードのリストを反復処理して取引先名を表示するという処理を記述しています。ただし、myaccounts は現時点ではまだ定義されておらず、今回指定した MyController 内に記述していきます。

3. 保存ボタンをクリックします。
4. 「MyController」という Apex クラスが存在しないことを通知するエラーメッセージが表示されます。右側のリンクをクリックして Apex クラス「**public class MyController**」を作成します。
5. 「MyController.myaccounts」というプロパティが不明であることを通知するエラーメッセージが表示されます。右側のリンクをクリックして Apex メソッド「**MyController.getMyaccounts**」を作成します。

以上により、Page Editor で 2 つの点が変わります。1 つ目は、[不明なプロパティ「String.name」] というエラーメッセージが表示されることです。このメッセージは、getMyAccounts() メソッドの定義が完了しておらず、Visualforce でこのメソッドの戻り値の型を識別できないために表示されます。2 つ目は、[MyController] というタブの追加です。このタブを使って、コントローラの Apex クラスを編集できます。



### ステップ 2: 取引先レコードを取得するメソッドを追加する

このステップでは、getMyAccounts() メソッドを定義していきます。最近変更された取引先 10 社のレコードを返すようにメソッドを編集します。

1. [MyController] タブをクリックします。
2. getMyAccounts() メソッドを以下のように修正します。

```
public List<Account> getMyAccounts() {
    return [SELECT Id, Name, Accountnumber from Account ORDER BY
            LastModifiedDate DESC LIMIT 10];
}
```

### 3. 保存ボタンをクリックします。

最近編集した取引先 10 社の名前がページに表示されます。ここでは、Visualforce ステートメント `{! myaccounts}` により `getMyAccounts()` メソッドが実行され、そのメソッドにより取引先レコードのリストが返され、そのリストが `<apex:dataList>` コンポーネントによって反復処理されます。

## ステップ 3: コントローラにアクションメソッドを追加する

このステップでは、Visualforce ページ内のボタンやリンクをユーザがクリックしたときに呼び出すアクションメソッドを追加します。前のチュートリアルでは、標準コントローラの `save()` などのメソッドを使用していましたが、今回は独自のカスタムコントローラを記述します。Visualforce ページを編集し、選択した取引先レコードに応じて各レコードに関連付けられた取引先責任者レコードのリストが動的に表示されるようにします。

1. [MyController] タブをクリックし、前のステップで作成したコントローラの編集画面を開きます。
2. 「`public class MyController {`」の行の下に、以下の 2 行を追加します。

```
public Id selectedAccount {get;set;}
public List<Contact> contactsInformation{get;set;}
```

これにより、2 つのプロパティが作成されます。1 つは ID を示す `selectedAccount`、もう 1 つは取引先責任者レコードのリストを示す `contactsInformation` です。

3. Page Editor のタブを切り替えて Visualforce ページを編集します。 `</apex:form>` 行のすぐ下に、以下のスニペットを追加します。

```
<apex:outputPanel id="ContactDetail">
  <apex:repeat value="{! contactsInformation}" var="contact">
    <p>{! contact.FirstName & ' ' & contact.LastName}</p>
  </apex:repeat>
</apex:outputPanel>
```

これにより、取引先責任者レコードのリストが反復処理され (これは、`contactsInformation` が `List<Contact>` を返すように定義されているためです)、取引先責任者の氏名が表示されます。 `&` は連結演算子です。なお、この時点では、ページを保存しても何も変更されません。後続の手順で、取引先名をクリックしたときにそのレコードに関連付けられた取引先責任者レコードを `contactsInformation` プロパティが取得するように設定します。

4. [MyController] タブに切り替えてコントローラを編集します。最後のかっこの前に以下のコードを追加します。

```
public void accountClicked(){
    contactsInformation = [SELECT FirstName, LastName From Contact
                          WHERE AccountID=:selectedAccount];
}
```

5. Visualforce ページのコードで「`<apex:outputText value="{! acct.name}" />`」の行を次の内容で置き換えます。

```
<apex:commandlink action="{! accountClicked}" re-render="ContactDetail">
  <apex:outputText value="{! acct.name}" />
  <apex:param name="id" value="{! acct.Id}"
    assignTo="{!selectedAccount}" />
</apex:commandLink>
```

このコードでは、`render` 属性によって、Ajax の更新機能を使用して出力パネルが更新されるように指定しています。

#### 6. 保存ボタンをクリックします。

これにより、取引先責任者レコードが関連付けられた取引先をクリックした場合に、取引先リストの下に該当する取引先責任者のレコードを動的に表示できるようになります。

なお、取引先名をクリックすると、次のようなさまざまな処理が行われます。

- `<apex:param>` コンポーネントにより、現在の取引先レコードの ID がコントローラのプロパティ `selectedAccount` に最初に割り当てられます。これにより、ユーザが選択した取引先をコントローラ側で特定します。
- 対象の取引先名は `<apex:commandLink>` コンポーネントで囲まれているため、アクション属性 `accountClicked()` に指定されたメソッドが呼び出されます。
- `accountClicked()` のメソッドが呼び出されると、メソッドは `selectedAccount` の ID を使用してシンプルなクエリを実行し、その結果を `contactsInformation` プロパティに割り当てます。
- `<apex:commandLink>` コンポーネントには `render` 属性が指定されているため、出力パネルが動的に表示され、その後 `contactsInformation` に対して反復処理が実行され、取引先責任者名が取得されます。

以上、駆け足で説明しました。このステップを試される際は、属性の削除、メソッドの編集などを、変更結果に注意しながら時間をかけて実行してみてください。また、「[チュートリアル 11: Ajax によるページの部分的な更新](#)」をもう一度参照し、Ajax の機能について確認してください。

## まとめ

カスタムコントローラでは、カスタムのロジックやデータ操作を実装して Visualforce ページで利用することができます。表示するアイテムのリストの取得、外部 Web サービスの呼び出し、データの検証や挿入をはじめとするさまざまな操作を記述できます。該当の操作は、作成したコントローラを Visualforce ページに含めることで実行されます。

## 補足

- Apex におけるプログラミングの基本については、「[Force.com Workbook](#)」が参考になります。
- Apex に関する詳細は、「[Force.com Apex Code Developer's Guide](#)」を参照してください。

## おわりに

以上ですべてのチュートリアルが完了しました。おめでとうございます。これで、Visualforce を構成する主要素について十分に理解できたと思います。Visualforce では、今回学んだこと以外にも多くのことを実現できます。たとえば、ページを PDF として表示したり、クエリパラメータを使ってページを操作したり、高機能なダッシュボードコンポーネントを作成したり、電子メールテンプレートを定義したり、ページ送りを調整したり、Force.com サイト内にページを作成したりすることが可能です。これらの機能はどれも決して難しいものではありません。皆さんは基本をすでに習得済みなので、すなわち。