

# JSF 2 で RichFaces を使用する

## Web ページのコンポーネントを RichFaces 4 にマイグレーションする

Joe Sam Shirah

Developer and Principal  
conceptGO

2012年 2月 10日

JavaServer Faces (JSF) で使用するために設計された他のリッチ/Ajax コンポーネント・フレームワークと同じく、RichFaces も JSF 2 での大幅な変更に対応するように大々的に変更されました。この記事の著者である Joe Sam Shirah は、以前の記事「[An introduction to RichFaces](#)」で RichFaces バージョン 3.1 の機能を紹介しましたが、今回はそれと同じ機能を提供する、新しく変更された RichFaces 4.1 のコンポーネントについて説明します。また、JSF で RichFaces を使用するためのセットアップ要件に関する最新の情報も提供します。

2009年にリリースされた JavaServer Faces (JSF) 2 では、さまざまな部分で大幅な変更と追加が行われました。これには、システム・イベント、リソース、そして Facelet と Ajax 両方の処理の標準化も含まれます（「[参考文献](#)」を参照）。この進化は一般には歓迎されたものの、残念ながら重大な副作用を伴いました。それは、JSF 1.2 用に作成されたほぼすべてのリッチ・コンポーネント・フレームワークが、まったくとは言わないまでも、確実に機能しなくなったことです。RichFaces バージョン 3.x 以降もその例外ではありません。これに対応し、RichFaces チームはバージョン 4 に向けた大々的な作り直しに着手しました。この記事を読むとわかるように、新しいバージョンでは一部のコンポーネントの名前が変更されています。また、削除されたコンポーネントや新しく追加されたコンポーネントもあります。

### RichFaces 4.1

この記事でターゲットとする RichFaces は、RichFaces 4.1 です（この記事が執筆している時点では、マイルストーンおよび候補の段階にあります）。4.0 リリースでは、複数の項目を選択して順番に並べることができる「List Shuttle (移動式項目選択)」コンポーネントがなくなったことが大きな痛手でしたが、バージョン 4.1 ではこれを補うべく、「Pick List (項目選択)」コンポーネントに並び替えの機能が設けられています。

私が 2008年に執筆した記事「[An introduction to RichFaces](#)」では、RichFaces バージョン 3.1 のコンポーネントをいくつか取り上げ、RichFaces と JSF (JavaServer Faces) 1.2 のセットアップ要件について説明しました。その記事をフォローアップする今回の記事は、RichFaces を初めて導入する開発者のためのガイドにも、前のバージョンからバージョン 4.x へと移行するための参考にもなります。記載するデモ・コードは、記事に付属の WAR から入手することができます（「[ダウンロード](#)」を参照）。

読者が RichFaces を使用するのが初めてで、JSF 2 と一緒に使用したいのであれば、この記事を読めば十分です (もちろん、[前の記事](#)の内容を読み返していただいても構いません)。もしバージョン 4 より前の RichFaces を使用しているとしたら、2 つの記事を並べて表示することをお勧めします。バージョン間の違いを比較しやすいように、両方の記事のセクション見出しは揃えてあります。また、コンポーネントの外観および機能も同じになるようにしました。

ここから先は、RichFaces のバージョンを記載しない場合は、バージョン 4.x を表すこととします。まずは、RichFaces を使用した開発で必要となるインフラストラクチャーの要素を検討するところから始めます。

## 基本から始めます

RichFaces を使用した開発で必要となるインフラストラクチャーの最小要件は以下のとおりです。

- Java SE 6
- Servlet 2.5 コンテナ
- なるべく最近のブラウザ (Firefox 3.5 以降または Internet Explorer 7 以降など)

私が開発およびテストに使用したのは、JDK 7、Apache Tomcat 7.0.22、GlassFish 3.1.1 です。ブラウザには Firefox 7.0.1 と Internet Explorer 8 を使用しました。

私が説明するセットアップと上記の最小要件に従っていれば、通常は問題が発生することはないはずですが、[デモ・コード](#)に関しては以下の点に注意してください。

- Servlet 2.5 しかサポートしないコンテナを使用する場合には、web.xml で、`<web-app version="3.0" ...>` を `<web-app version="2.5" ...>` に変更する必要があります。
- GlassFish V3 および他の JEE 6/JSF 2 をサポートするコンテナでは、デモ・プロジェクトから `javax.faces.jar` を削除することができます。このデモで使用するバージョンは、`mojarra-2.1.4` です。
- メーリング・リストに、JSF 2 バージョン以前のコンテナでは、EL (Expression Language) でたまに問題が発生するという指摘がありました。おかしい結果になった場合には、最新の EL JAR をダウンロードしてみてください。

この記事では、読者に JSF 2 と、Tomcat または GlassFish に関する基本的な知識があることを前提とします。これらの技術の予備知識が必要な場合には、「[参考文献](#)」で該当するリンクを参照してください。

## RichFaces の要件である Facelets

その名前に反して、Facelets は JSF を小さくしたバージョンではありません。根本的に、Facelets は JSP (JavaServer Pages) に代わる JSF の `ViewHandler` 実装です。Facelets は JSF のすべての UI コンポーネントをサポートし、JSF アプリケーション向けのビューを反映した独自のコンポーネント・ツリーを組み立てます。JSF 2 仕様では、ビュー定義言語 (VDL) としての JSP のサポートを廃止して、優先される VDL として Facelets の標準バージョンを組み込んでいます。このことから、RichFaces では JSP のサポートを廃止し、Facelets を要件としています。

私はなるべく物事を単純明快にしておきたい性格です。このサンプル・プロジェクトでは、ページを構成するコードのほぼすべてが (マークアップとは対照的に) ゲッター、セッター、メソッド

ド・バインディングの EL 式で構成されています。これよりも複雑なプロジェクトでは、式も複雑になってくるはずですが、一般に、Facelets は Java コードを Web ページのマークアップとは切り離しやすくします。

JSP を使用した JSF 2 より前の JSF 開発と、JSF 2 の開発との主な違いとして、以下の点が挙げられます。

- web.xml および faces-config.xml にいくつかの表記を追加する必要があること。あるいは、追加の表記がデフォルト設定されること。
- Web ページが XHTML 文書であること。
- JSP taglib の代わりに XML 名前空間が使用されること。

フォーマットに関する限り、最初の部分 (リスト 1 を参照) を除けば、Web ページのコードはどれも今までと同じように見えるはずです。私はこの側面を、控えめな Facelets の特徴の 1 つだと思っています。この記事のプロジェクト (そして、主にビューを処理するために Facelets を使用する多くのプロジェクト) で知っておかなければならないのは、まさにこの Facelets の控えめな部分です。Facelets にはそれ以外にも、テンプレートを簡単に作成できる機能とか、Web ページ・デザイナーの作業を楽にしてくれる要素など、数多くの便利な機能があります (Facelets の詳細を学ぶには、「[参考文献](#)」を参照してください)。

## リスト 1. Facelets XHTML 文書の最初の部分

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets">
```

## デモを行うコンポーネント

RichFaces は、一目見るだけでも非常に優れていることがわかります。RichFaces の平均的なコンポーネントには、設定可能な属性として、一般属性に加え、20 を超える固有の属性があります。ただし、ほとんどの属性には妥当なデフォルト値が設定されているので、標準的な使い方であれば、コンポーネントをセットアップするのは難しくありません。バージョン 4 で変更されたデフォルトおよび追加されたデフォルトについては、ドキュメントを確認するだけの価値があります。

以下に、この記事のデモで取り上げる主なコンポーネントを紹介します (バージョン 4 で置き換えられたコンポーネントについてはその旨を明記します)。

- **Calendar (カレンダー)**: 日付を選択できるポップアップ・コンポーネント。図 1 に一例を示します。「<」と「>」で月を移動し、「<<」と「>>」で年を移動します。一番下の「Today (今日)」をクリックすると、今日の日付が選択されます。「Clean (クリア)」は、日付の選択を解除します。左端の最初の列に示されている数字は、その週がその年の何週目であるかを示します。

## 図 1. RichFaces の「Calendar (カレンダー)」コンポーネント



- Pick List (項目選択): 選択および並び替えコンポーネント。バージョン 4 で「List Shuttle (移動式項目選択)」コンポーネントに代わって導入された「Pick List (項目選択)」コンポーネントは、選択可能な項目リストと選択済み項目リストとの間で項目を移動したり、選択済み項目リストの中で項目を上下に移動したりすることができます。図 2 に一例を記載します。

## 図 2. RichFaces の「Pick List (項目選択)」コンポーネント



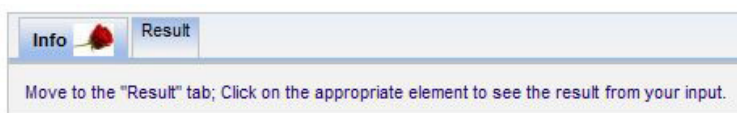
- AutoComplete (自動補完): バージョン 4 で「Suggestion Box (候補ボックス)」入力コンポーネントに代わって導入されたコンポーネント。このコンポーネントは入力を補完するための候補を表示し、その候補をクリックすることで項目に入力することができます。図 3 に一例を記載します。

## 図 3. RichFaces の「AutoComplete (自動補完)」コンポーネント



- Tab Panel (タブ・パネル): タブ付きページを作成する出力コンポーネント。図 4 に一例を記載します。

## 図 4. RichFaces の「Tab Panel (タブ・パネル)」コンポーネント



- Accordion (アコーディオン): バージョン 4 で「Panel Bar (パネル・バー)」入力コンポーネントに代わって導入されたコンポーネント。サンプル・プロジェクトでは、「Accordion (アコーディオン)」コンポーネントを使用して説明がされています。図 5 に一例を記載します。

図 5. RichFaces の「Accordion (アコーディオン)」コンポーネント



- Collapsible Panel (折り畳みパネル): バージョン 4 で「Simple Toggle Panel (シンプル・トグル・パネル)」コンポーネントに代わって導入されたコンポーネント。サンプル・プロジェクトでは、「Collapsible Panel (折り畳みパネル)」コンポーネントに結果が表示されます。図 6 に一例を記載します。

図 6. RichFaces の「Collapsible Panel (折り畳みパネル)」コンポーネント



RichFaces は Ajax4jsf (「[参考文献](#)」を参照) をベースに作成されています。そのおかげで、あらゆるコンポーネントをさまざまな方法で Ajax 対応にすることができます。バージョン 4 では多くの場合、Ajax 機能は自動的に適用されるか、デフォルトで有効になっています。サンプル・アプリケーションでは「AutoComplete (自動補完)」コンポーネントと「Collapsible Panel (折り畳みパネル)」コンポーネントに Ajax 機能を使用します。

## RichFaces サンプル・アプリケーションの見事な姿

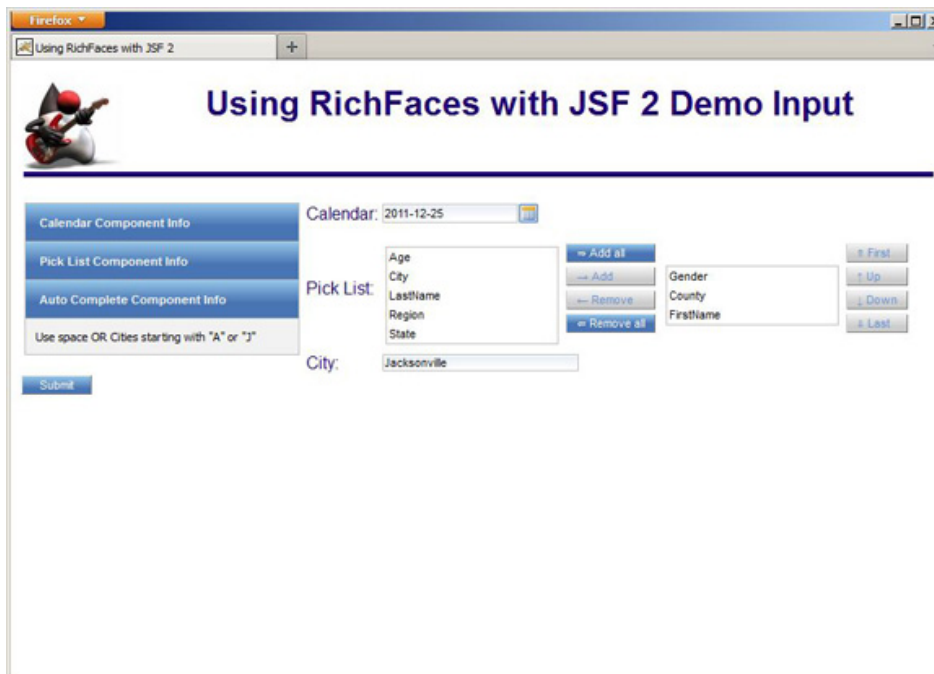
dwRichFaces4Demo1 サンプル・アプリケーションは必要最小限のアプリケーションです。その唯一の目的は、選択したコンポーネントのセットアップおよび使用方法を実例で説明することなので、このサンプル・アプリケーションが実行する内容は、入力データの収集とその表示のみ



です。本番アプリケーションでどのようにデータとコンポーネントを使用するかについては、皆さんの想像力にお任せします。必要な JAR、画像、リソース・バンドルをサポートするプロパティ・ファイル、および CSS (Cascading Style Sheets) ファイルの他、このサンプル・アプリケーションは 2 つの XHTML ページと 2 つの Java クラスで構成されています。

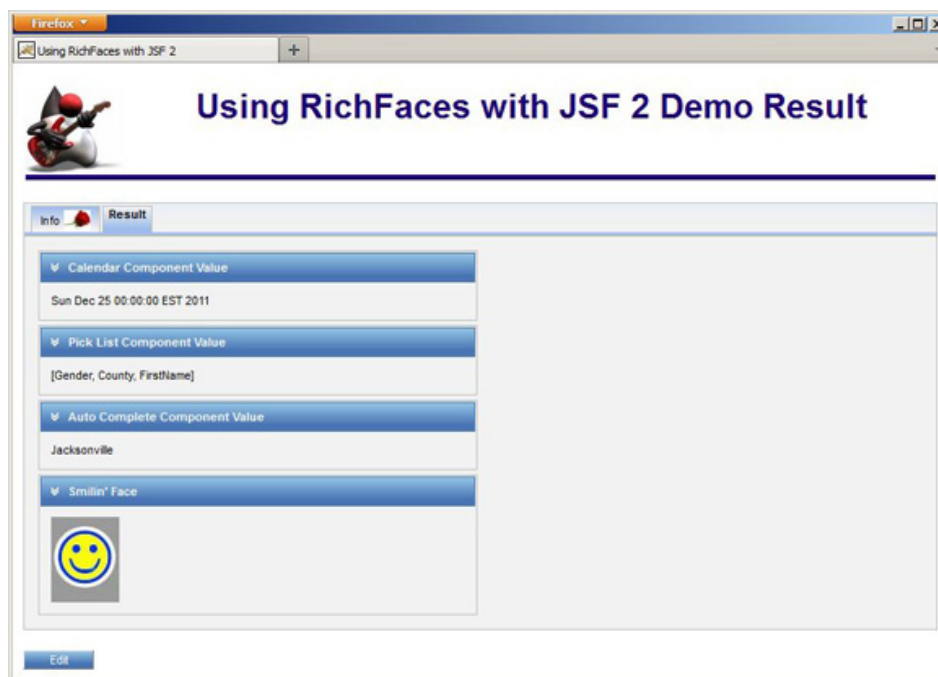
図 7 に表示されている入力ページの URL は、(Tomcat または GlassFish のデフォルト・セットアップを使用しているという前提で) `http://localhost:8080/dwRichFaces4Demo1` です。この入力ページでは、「Calendar (カレンダー)」コンポーネントを使用して日付を選択することができます。「Pick List (項目選択)」コンポーネントでは、選択可能な項目を移動して、並び替えることができます。「City (都市)」では、「AutoComplete (自動補完)」コンポーネントを使用して都市名を入力できるようになっています。「City (都市)」は Ajax 対応のフィールドで、スペース・バーを押すと選択可能なすべての都市が表示されます。例えば、都市名の最初の文字として「A」または「J」を入力すると、「A」または「J」で始まる都市のリストが表示されます。リストに表示される選択可能な都市は、文字を入力するにつれ、絞り込まれていきます。コンポーネントの基本的な使用手順を調べるには、左側の「Accordion (アコーディオン)」コンポーネントの項目をクリックします。

図 7. `dwRichFaces4Demo1` の入力ページ



項目の入力が完了したら、「Submit (送信)」ボタンをクリックします。このアプリケーションは必要最小限のアプリケーションなので、編集操作は一切行うことができません。「Calendar (カレンダー)」コンポーネントには手で入力することができないため、無効な日付を入力することもできません。「Submit (送信)」ボタンをクリックすると、結果が表示されます (図 8 を参照)。

## 図 8. dwRichFaces4Demo1 を実行した結果のページ



結果のページで「Result (結果)」タブをクリックして、該当する「Collapsible Panel (折り畳みパネル)」コンポーネントの項目をクリックすると、入力した値が表示されます。入力ページに戻るには、「Edit (編集)」ボタンをクリックします。

図 7 の「Submit (送信)」ボタンと図 8 の「Edit (編集)」ボタンは、RichFaces ではなく JSF の標準コンポーネントですが、これらのボタンの背景色は他の要素と同じである点に注目してください。その理由は、次のセクションで説明します。

## RichFaces サンプル・アプリケーションのためのセットアップ

サンプル・アプリケーションに何よりもまず必要なのは、JSF、Facelets、そして RichFaces のイネーブラーです。つまり、これらの機能を実装する JAR がなければ何も始まりません。マイグレーションする場合、バージョン 4.x では依存関係が一新されていることに注意してください。この記事から[ダウンロード](#)できる WAR (「ダウンロード」を参照) の lib ディレクトリーには、以下に示すバージョンの JAR が組み込まれています。以下のリストは、EL および Servlet API の最新バージョンをサポートする Web コンテナを使用していることを前提としています。デモを実行できない場合には、JSF、Facelets、および RichFaces の要件を確認してください(「[参考文献](#)」を参照)。また、「[ダウンロード](#)」の注記も一読してください。

あらゆる RichFaces プロジェクトに共通して必要な JAR は以下のとおりです。

- JSF 2 (GlassFish V3.x などの JEE 6 コンテナに付属)
  - javax.faces.jar
- Facelets
  - JSF 2 JAR に付属
- RichFaces

- richfaces-components-api-ver.jar
- richfaces-components-ui-ver.jar
- richfaces-core-api-ver.jar
- richfaces-core-impl-ver.jar
- 上記の他に必要な依存関係
  - cssparser-ver.jar
  - guava-ver.jar
  - sac-ver.jar

サンプル・プロジェクトと付属のダウンロード WAR ([ダウンロード・サイト](#)については「[参考文献](#)」を参照) では、以下のバージョンが使用されています。

- JSF 2
  - javax.faces.jar — mojarra-2.1.4
- Facelets
  - javax.faces.jar に付属
- RichFaces
  - richfaces-components-api-4.1.0.20111111-CR1.jar
  - richfaces-components-ui-4.1.0.20111111-CR1.jar
  - richfaces-core-api-4.1.0.20111111-CR1.jar
  - richfaces-core-impl-4.1.0.20111111-CR1.jar
- その他の必要な依存関係
  - cssparser-0.9.5.jar
  - guava-10.0.1.jar
  - sac-1.3.jar

JSF を有効にするために次に必要となるのは、リスト 2 に記載する web.xml のエントリーです。JSF 2 を Servlet 3.0 コンテナで使用するとしたら、これらのエントリーはオプションですが、私は明示するほうを選びます。省略した場合、`*.faces`、`*.jsf`、および `/faces/*` `<url-pattern />` は自動的にマッピングされます。

## リスト 2. web.xml に最小限必要な JSF エントリー

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Faces Servlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

JSF 2 より前のバージョンでは、Facelets がデフォルトの .jsp 接尾部を .xhtml に指定変更するために、`javax.faces.DEFAULT_SUFFIX` エントリーが必要でした。現在は Facelets がデフォルトの VDL となっているため、このエントリーは必要ありません。

RichFaces 4 では、web.xml に `org.ajax4jsf.VIEW_HANDLERS`、`filter`、`filter-mapping` といった要素を含める必要がなくなりました。JAR を放り込むだけで、JAR を使用することができます。



リスト 3 に、デモ・アプリケーションで使用する、web.xml 内の RichFaces 関連のエントリーを記載します。

### リスト 3. web.xml 内の RichFaces 関連エントリー

```
<context-param>
  <param-name>org.richfaces.skin</param-name>
  <param-value>classic</param-value>
</context-param>
<context-param>
  <param-name>org.richfaces.enableControlSkinning</param-name>
  <param-value>true</param-value>
</context-param>
```

リスト 3 には、以下の 2 つのコンテキスト・パラメーターが組み込まれています。

- `<param-name>org.richfaces.skin</param-name>`: アプリケーションのカラー・スキーム (スキン) を定義するパラメーターです。RichFaces には複数の組み込みスキンがあります。classic スキンは濃くも薄くもない青色です。この要素を省略すると、デフォルトで明るいグレーに設定されます。
- `<param-name>org.richfaces.enableControlSkinning</param-name>`: このパラメーターの値は、RichFaces の外観だけでなく、意外なことに標準 JSF コンポーネントの外観にも影響します。値が true の場合、標準コントロールにスキンが適用されます。図 7 の「Submit (送信)」ボタンと図 8 の「Edit (編集)」ボタンの背景色が RichFaces のテーマと同じ色になっているのは、これが理由です。この要素を省略すると、値は true にデフォルト設定されます。

嬉しいことに、リスト 2 とリスト 3 のエントリーは、実質的にすべてのアプリケーションで共通していて、基本的にボイラープレート・コードとなっています。デフォルトを喜んで受け入れ、明示的に指定することにこだわらない場合には、完全に手間を省くことができます。

すべてのアプリケーションで共通している部分はもう 1 つあります。それは、アプリケーションの XHTML ページで使用される RichFaces 名前空間です。リスト 4 に、RichFaces 名前空間を追加した後のリスト 1 を記載します。

### リスト 4. Facelets/RichFaces XHTML 文書の最初の部分

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:a4j="http://richfaces.org/a4j"
  xmlns:rich="http://richfaces.org/rich" >
```

## Accordion (アコーディオン) コンポーネントを使ってみる

RichFaces のコンポーネントの使い方を実際に確かめる準備ができました。まずは、「Accordion (アコーディオン)」コンポーネントと「Accordion Item (アコーディオン項目)」コンポーネントから始めます (図 5 を参照)。これらのコンポーネントは、RichFaces 4 より前の「Panel Bar (パネル・

バー)」および「Panel Bar Item (パネル・バー項目)」コンポーネントに代わって導入されました。度々使用することはないかもしれませんが、扱いやすいので、RichFaces 構文の最初の使用例としてはもってこいのコンポーネントです。

ここでの考え方は、「Accordion (アコーディオン)」コンポーネントは「Accordion Item (アコーディオン項目)」コンポーネントのコンテナであるというものです。「Accordion Item (アコーディオン項目)」コンポーネントにはヘッダーがあり、その下に他のあらゆるコンポーネントを含めることができます。「Accordion Item (アコーディオン項目)」コンポーネントは積み重ねられていて、項目のバーをクリックすると、その項目のコンテンツが表示されます。一度にひとつの項目のコンテンツしか表示されません。この例で使用しているコンテンツは、リスト 5 を見るとわかるようにテキストだけです。すべてのコンポーネントに `rich:` という接頭辞が付いていることに注目してください。この接頭辞は、[リスト 4](#) に記載した名前空間のエントリーを参照します。このデモでの「Accordion (アコーディオン)」コンポーネントは、JSF の標準 `PanelGrid` の中に収容されています。

## リスト 5. RichFaces の「Accordion (アコーディオン)」コンポーネント

```
<rich:accordion style="width: 256px;" styleClass="floatLeft">
  <rich:accordionItem header="#{dwRF4D1.calComp}" #{dwRF4D1.info}>
    <h:outputText value="#{dwRF4D1.calCompText}" style="font: menu;" />
  </rich:accordionItem>
  <rich:accordionItem header="#{dwRF4D1.plComp}" #{dwRF4D1.info}>
    <h:outputText value="#{dwRF4D1.plCompText}" style="font: menu;" />
  </rich:accordionItem>
  <rich:accordionItem header="#{dwRF4D1.acComp}" #{dwRF4D1.info}>
    <h:outputText value="#{dwRF4D1.acCompText}" style="font: menu;" />
  </rich:accordionItem>
</rich:accordion>
```

基本的な `<rich:accordionItem />` 要素に必要なのは、ヘッダーくらいのもので、この例では、EL 式を使ってリソース・バンドルからヘッダーを取り込みます。

デモ用「Accordion Item (アコーディオン項目)」コンポーネントに実際に含まれているコンテンツは、`<h:outputText />` 要素だけです。このテキストも、同じくリソース・バンドルから取り込まれます。font スタイル要素を使用しているのは、一貫して読みやすくするため、そして RichFaces では CSS を柔軟に使用できることを示すためです。

生成される JavaScript には、プログラマーの関与がまったく必要ないことに注目してください。ほんのわずかなステップで、見栄えが良く、複数のパネルを持った、クリック可能なコンポーネントを作成することができます。これが、RichFaces の素晴らしい利点の 1 つです。このデモで使っているような単純なコンポーネントでも、Ajax (`switchType` 属性を使用)、アクティブ/非アクティブ・スタイル、イベントなどを使用することができます。

## Calendar (カレンダー) コンポーネントを使ってみる

「Calendar (カレンダー)」コンポーネント ([図 1](#) を参照) についてはお馴染みのはずです。日付選択は、Web ページに加えられた JavaScript による機能強化のうち、おそらく最も初期の頃に行われたものの 1 つでしょう。RichFaces の「Calendar (カレンダー)」コンポーネントで使用する属性は 80 を超えていますが、リスト 6 に示されているとおり、たった数行で数多くの機能を有効にすることができます。

## リスト 6. RichFaces の「Calendar (カレンダー)」コンポーネント

```
<label for="CalendarID" >#{dwRF4D1.calendar}</label>
<rich:calendar id="CalendarID"
    value="#{dwRF4D1Handler.selectedDate}"
    timeZone="#{dwRF4D1Handler.timeZone}"
    datePattern="#{dwRF4D1.datePattern}" >
</rich:calendar>
```

`datePattern` 属性には、`java.text.SimpleDateFormat` で定義された標準日付パターンが値として必要です。この例では、この属性に対しても、同じくリソース・バンドルを使って `datePattern` キーを `yyyy-MM-dd` として定義する値を取得します。`value` 属性と `timeZone` 属性は、`faces-config.xml` に定義された管理対象 Bean のメソッドを使用してロードします (リスト 7 を参照)。

## リスト 7. `dwRF4D1Handler` の管理対象 Bean の定義

```
<managed-bean>
  <managed-bean-name>dwRF4D1Handler</managed-bean-name>
  <managed-bean-class>com.dw.dwRF4D1Handler</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

`com.dw.dwRF4D1Handler` クラスは、主にイニシャライザー、ゲッター、セッターからなる基本的なクラスなので、それほど詳しく説明する必要はないでしょう。簡単な説明として、`getSelectedDate()` および `setSelectedDate()` メソッドは `java.util.Date` を想定しています。`getTimeZone()` メソッドは単純に `java.util.TimeZone.getDefault()` を返すだけですが、これが本番にふさわしいかどうかは、それぞれのアプリケーションの要件次第です。

日付の値をユーザーが入力できるようにするには、`enableManualInput` 属性を `true` に設定します。「Calendar (カレンダー)」コンポーネントを表示するだけにしたい場合は、`popup` 属性を `false` に設定します。もちろん、使用できる設定は他にもたくさんありますが、基本的な「Calendar (カレンダー)」コンポーネントの機能に必要なのは、ここで説明した設定だけです。

RichFaces の前のバージョンから移行している場合、デモ・コードの機能向けに「Calendar (カレンダー)」コンポーネントをセットアップする手順は、実質的にバージョン 4 での手順と変わりません。

## Pick List (項目選択) コンポーネントを使ってみる

`orderable` 属性を持つ「Pick List (項目選択)」コンポーネントは、バージョン 4 で「List Shuttle (移動式項目選択)」コンポーネントに代わって導入されました (図 2 および囲み記事「[RichFaces 4.1](#)」を参照)。これは、初期状態のグループから項目を選択して順番に並べるにはうってつけのコンポーネントです。リスト 8 に、このコンポーネントは当初思ったよりも簡単に使用できることを実証します。

## リスト 8. RichFaces の「Pick List (項目選択)」コンポーネント

```
<label for="plID">#{dwRF4D1.pl}</label>
<rich:pickList id="plID"
    value="#{dwRF4D1Handler.orderByTarget}"
    orderable="true" listWidth="10em"
    sourceCaption="#{dwRF4D1.available}"
    targetCaption="#{dwRF4D1.selected}" >
    <f:selectItems value="#{dwRF4D1Handler.orderBySource}" />
</rich:pickList>
```

サンプル・コードの XHTML マークアップには、ラベルをデフォルトのラベルから変更するための属性が追加で組み込まれていますが、一般的な用途で必要となるのは `value` 属性と `<f:selectItems />` だけであり、大抵はそれに加えて `orderable` が必要となります。

選択済みの値を表す `value` と、選択可能な項目を表す `<f:selectItems />` は、任意の型のオブジェクトを収容する `java.util.List` クラスです。JSF 2 では、`<f:selectItems />` の値を、任意の型のオブジェクトからなるコレクションまたは配列にすることができます。ラベルを指定するには `toString()` を呼び出します。このデモ・コードと同じく、大抵は `String` からなるリストを使用することになるでしょう。`<f:selectItems />` 属性値の `List` 初期化は別として、リスト 9 にサンプル「Pick List (項目選択)」コンポーネントに関連するすべての Java ソースを記載します。この Java コードは、「[An introduction to RichFaces](#)」で使用したリストと基本的に変わりません。

## リスト 9. dwRF4D1Handler の「Pick List (項目選択)」コンポーネント関連のコード

```
private List<String> lOrderBySource = new ArrayList<String>(),
    lOrderByTarget = new ArrayList<String>();
...
public List getOrderBySource()
{
    return lOrderBySource;
}

public List getOrderByTarget()
{
    return lOrderByTarget;
}
...
public void setOrderBySource( List<String> orderBySource )
{
    lOrderBySource = orderBySource;
}

public void setOrderByTarget( List<String> orderByTarget )
{
    lOrderByTarget = orderByTarget;
}
```

ユーザーが必要な選択を行って送信した時点で、ハンドラーは選択項目が含まれるリストを受信します。

## AutoComplete (自動補完) コンポーネントを使ってみる

フォーラムやメーリング・リストに頻繁に参加しているうちに、遅かれ早かれ、数千や数万のエントリーをドロップダウン・リストにダウンロードする場合の処理方法についての質問を目にするはずです。「Suggestion Box (候補ボックス)」入力コンポーネントに代わって導入された「AutoComplete (自動補完)」コンポーネント ([図 3](#) を参照) は、このような実行不可能な大量エン

トリーのダウンロードを行わなくても、有効な入力の選択肢を表示する手段となります。実際、私が RichFaces や同様のスイートを調べ出したのは、この問題が大きな理由でした。実行中のアプリケーションで入力する項目のなかには、実際にドロップダウン・リストに表示するには選択肢の数が多すぎるもの（例えば、都市名など）があります。「AutoComplete (自動補完)」コンポーネントの機能は、多くのデスクトップ・アプリケーションで使用されているお馴染みの自動補完コンポーネントに似ています。Web アプリケーションで効率良く作業できるようにするためには、この種の機能に対して Ajax 送信を使用する必要があるのは明らかです。

リスト 10 に、「AutoComplete (自動補完)」コンポーネントのマークアップを記載します。フィールドへの入力は、キーからの入力か、ドロップダウン・リストからの選択のいずれかです。

## リスト 10. RichFaces の「AutoComplete (自動補完)」コンポーネント

```
<h:form>
  <a4j:queue ignoreDupResponses="true"/>

  ...

<label for="CityID">#{dwRF4D1.city}</label>
<rich:autocomplete id="CityID" mode="ajax"
  value="#{dwRF4D1Handler.city}"
  autocompleteMethod="#{dwRF4D1City.suggest}"
  autofill="false"
  selectedItemClass="selCtl" >
</rich:autocomplete>
```

「Suggestion Box (候補ボックス)」入力コンポーネントを使用した経験があれば、「AutoComplete (自動補完)」コンポーネントの単純さがありがたく感じられるはずです。最小限必要な属性は、mode、value、そして autocompleteMethod だけです。

- mode には通常 ajax を指定することで、それぞれのキー入力に対して Ajax リクエストが送信されるようにします。これは非常に便利で基本的に自動で行われます。mode の値としては、ajax の他に Cached Ajax、Client、Lazy Client があります。
- value は、他のほとんどのコンポーネントと同じで、通常は管理対象 Bean からのゲッターおよびセッターです。
- autocompleteMethod メソッドは、デモ・コードの 2 番目の Java クラスをベースとした別の管理対象 Bean である com.dw.City の suggest() です。メソッド・シグニチャーは、public List suggest(Object anObject) です。

デモ・コードでは、初期化の時点で「A」および「J」で始まる都市のリストが作成されます。ユーザーが「City (都市)」テキスト・フィールドでスペース・バーを押すと、リスト全体が返されます。「A」が押されると、「A」で始まる都市が含まれるサブリストが返されます。「J」で始まる都市も、同じように処理されます。結果は、キー入力ごとに絞り込まれます。皆さんが作成する独自のコードでは、入力に対して適切な処理内容であれば、どのような処理にすることもできます。私の本番アプリケーションでは、限られたデータベース・セットから結果が返されるようになっていますが、他のアプリケーションでもおそらくこの仕組みが最も一般的なはずです。データを十分に理解していれば、通常はメソッドに手を加えて、結果を最適化することができます。例えば、特定のキー入力は無視できるので、そのキーが押された場合には、メソッドがデータベースにアクセスせずにそのままリターンすることができます。



バージョン 4.x の RichFaces でベースとしているキューは、JSF の標準キューです。<a4j:queue /> 属性は必須ではありませんが、おそらく頻繁に使用したくなることと思います。この例では、ignoreDupResponses でこの属性を使用しています。他の用途についてはドキュメントを調べてください。注意する点として、<a4j:queue /> は RichFaces のコンポーネント・イベントに専用の属性で、ビューまたはフォーム・レベルで 사용할 ことができます。グローバル・アプリケーション・キューを有効にするには、web.xml で<param-name>org.richfaces.queue.global.enabled</param-name> コンテキスト・パラメーターを使用します。

名前付きキューを作成して、コンポーネントが <a4j:attachQueue name="myRequestQueue" /> を使用して名前付きキューを参照するようにすることもできます。

selectedItemClass は、コンポーネントに適用される CSS クラスです。RichFaces が使用するスキンの概念は、コンポーネントでの選択時の背景にも引き継がれます。私独自のアプリケーションには、背景にシステム・カラーを使用する JSF の標準ドロップダウン・リストを使用しました。

「AutoComplete (自動補完)」コンポーネントでは classic スキンの色を使用したもので、一貫した外観となっています。select クラスは、代わりに lightgrey 背景色を使用するようにコンポーネントに指示します。

## Tab Panel (タブ・パネル) コンポーネントを使ってみる

「Tab Panel (タブ・パネル)」コンポーネント (図 4 を参照) はごく簡単に使用できるものの、前のバージョンから移行する場合には、いくつか注意しなければならない変更点があります。このコンポーネントでは、典型的な RichFaces の属性を使用することができますが、主要な属性は switchType です。この属性の値は、デフォルトでは server に設定されます。情報が不変の場合、おそらく最も頻繁に使用することになる値は client でしょう。また、ajax を指定することもできます。リスト 11 を見るとわかるように、タブ付きページのセットを作成する場合、典型的な RichFaces では <rich:tabPanel /> 要素の「サンドイッチ」と、その中に積み重ねた <rich:tab /> 要素の「サンドイッチ」を使用します。

### リスト 11. RichFaces の「Tab Panel (タブ・パネル)」コンポーネント

```
<rich:tabPanel switchType="client" >
  <rich:tab >
    <f:facet name="header">
      <h:panelGrid columns="2">
        <h:outputText value="#{dwRF4D1.info}" />
        <h:graphicImage value="/images/roseTab.jpg" height="20" width="25" />
      </h:panelGrid>
    </f:facet>
    #{dwRF4D1.resultText}
  </rich:tab>

  <rich:tab header="#{dwRF4D1.result}">
...
  </rich:tab>
...
</rich:tabPanel>
```

header 属性は、タブの表示名ごとに使用します。デモ・コードでは例の如く、ヘッダーの値をリソース・バンドルから取り込みます。「Accordion Item (アコーディオン項目)」コンポーネン

トと同じように、タブにはあらゆる種類のコンポーネントを含めることができます。デモでは、「Info (情報)」タブのコンテンツにリソース・バンドルのテキストを使用し、「Result (結果)」タブのコンテンツに「Collapsible Panel (折り畳みパネル)」コンポーネントを使用しています。

header 属性の代わりに、<f:facet /> で指定した「ヘッダー」を使用して、追加のコンテンツを組み込むこともできます。デモ・コードでは、テキストを格納した <h:panelGrid /> と、「Info (情報)」タブのヘッダーとして画像を使用しています。

## Collapsible Panel (折り畳みパネル) コンポーネントを使ってみる

バージョン 4.x では、「Simple Toggle Panel (シンプル・トグル・パネル)」コンポーネントに代わって「Collapsible Panel (折り畳みパネル)」コンポーネントが導入されました。「Collapsible Panel (折り畳みパネル)」コンポーネントは、バーとコンテンツの表示で構成されます。コンテンツには、あらゆるコンポーネントを使用することができます (図 6 を参照)。タイトルには header 属性を使用します。バーをクリックすると、1 つの項目しかない「Accordion (アコーディオン)」コンポーネントの如く、コンテンツの表示/非表示が切り替わります。リスト 12 に、デモ・プロジェクトの「Collapsible Panel (折り畳みパネル)」コンポーネントのマークアップを記載します。

### リスト 12. RichFaces の「Collapsible Panel (折り畳みパネル)」コンポーネント

```
<h:panelGrid columns="1" width="50%">
  <rich:collapsiblePanel switchType="ajax"
    header="{dwRF4D1.calComp} #{dwRF4D1.value}"
    expanded="false" >
    #{dwRF4D1Handler.selectedDate}
  </rich:collapsiblePanel>
  <rich:collapsiblePanel switchType="ajax"
    header="{dwRF4D1.plComp} #{dwRF4D1.value}"
    expanded="false" >
    #{dwRF4D1Handler.orderByTarget}
  </rich:collapsiblePanel>
  <rich:collapsiblePanel switchType="ajax"
    header="{dwRF4D1.acComp} #{dwRF4D1.value}"
    expanded="false" >
    #{dwRF4D1Handler.city}
  </rich:collapsiblePanel>
  <rich:collapsiblePanel switchType="ajax"
    header="{dwRF4D1.face}"
    expanded="false" >
    
  </rich:collapsiblePanel>
</h:panelGrid>
```

上記でも switchType 属性が使用されていることに注目してください。この属性の値には、このコードで使用されている ajax の他に、client と server を使用することができます。expanded 属性は、初期表示でコンテンツを表示するかどうかを決定します。デモでは、複数の「Collapsible Panel (折り畳みパネル)」コンポーネントを組み込んだ JSF <h:panelGrid /> をタブに配置し、そのタブを「Tab Panel (タブ・パネル)」コンポーネントに配置しています。

## まとめ

RichFaces には、RIA および Ajax 対応のアプリケーションを構築するために利用できる数多くの JSF コンポーネントが用意されています。この記事でデモとして取り上げたコンポーネントはその

なかのほんのわずかですが、RichFaces を適用する場合にはどのようにすればよいかを把握できたはずです。また、多くのアプリケーションで役立つ可能性のあるコンポーネントも見つかったはずです。バージョン 4 の導入によって JSF 2 に対応するようになった Richfaces には、評価に値する単純化と効率性が盛り込まれました。このスイートに含まれているコンポーネントすべてのオンライン・デモ、ドキュメント、その他のリソースは、RichFaces プロジェクト・ページ(「[参考文献](#)」を参照)で調べることができます。

RichFaces を使うことにした場合、RichFaces のドキュメントで説明されている `a4j:` コンポーネントおよび処理について詳しく調べ、これらの概念を他の RichFaces コンポーネントに適用する方法を理解するようお勧めします。この調査のために費やす時間の報いは、開発プロセスと実行時のパフォーマンスに何倍にもなって返ってくるはずです。

---

## ダウンロード

内容	ファイル名	サイズ
Demonstration WAR for this article <sup>1</sup>	<a href="#">j-richfaces4.zip</a>	8.9MB

### 注

1. The Java sources are included in the WAR file. For instruction on using the WAR file, see the readme.txt file included with the download.

## 著者について

Joe Sam Shirah



Joe Sam Shirah は、conceptGO の代表兼開発者です。クライアントの要求を満足させるかたわら、developerWorks および Sun 開発者向けサイトに多数のチュートリアルを発表しています。Java Community Award の受賞者である彼は、developerWorks の Java Filter Forum のモデレーターでもあり、jGuru の JDBC、i18n、そして Java400 FAQs を管理しています。

© Copyright IBM Corporation 2012

([www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml))

商標

([www.ibm.com/developerworks/jp/ibm/trademarks/](http://www.ibm.com/developerworks/jp/ibm/trademarks/))