目次

1. Sun Java Studio Creator で試すJSF

Sun Java Studio Creator とは

Sun Java Studio Creator とは Sun が提供するWebアプリケーション構築に特化した JSF 対応の開発環境です。

2004 年 4 月 8 日から Early Access 版がダウンロードできます。

http://developers.sun.com/prodtech/javatools/jscreator/index.jsp

将来的には正式版が有償で提供される予定です。

JSF(JavaServer Faces)とは

JSF は今後 J2EE の世界で標準になると予想される Web アプリケーションのインターフェイスを構築するためのフレームワークです。

JSF を使用することにより、Web アプリケーションのビジネスロジックとプレゼンテーションデザインを分離することができます。またコンポーネントに対して、ステート情報の保持や入力値のチェック、型の変換、イベントの制御といった機能を与えられるため、従来のWebアプリケーションでは困難であったユーザインターフェイス作成の問題を解決できると言われています。

1.1. 目的

本資料では Sun Java Studio Creator のマニュアルに従い、Sun Java Studio Creator と JSF の使用方法を見ていきます。

細かい説明は省きますので、英語が苦にならなければ英語マニュアルをダウンロードしてそちらを読む事をお勧めします☺

http://developers.sun.com/prodtech/javatools/jscreator/learning/tutorials/index.jsp

2. 環境構築

2.1. Sun Java Studio Creator のインストール

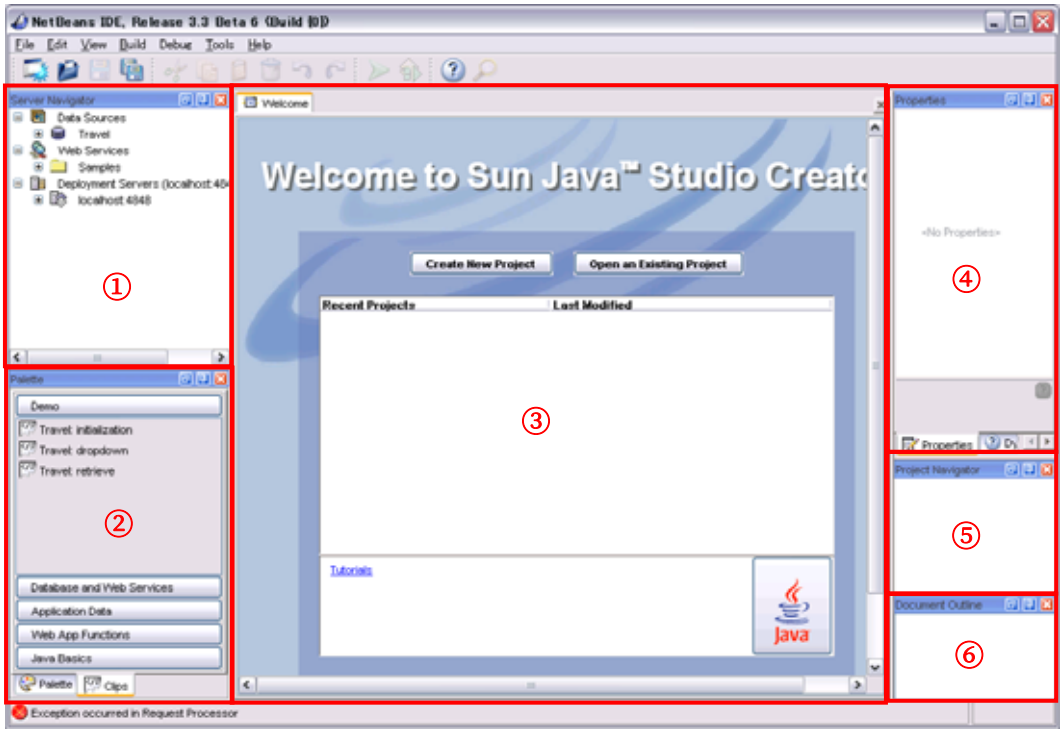
今回は Windows 上でサンプルアプリケーションを構築するので以下のサイトから Windows 版をダウンロードします。

http://developers.sun.com/prodtech/javatools/jscreator/index.jsp

ダウンロードした「creator-ea-windows-en.exe」を実行しインストールします。

インストール後起動すると以下のような初期画面が開きます。

説明のために以下の各エリアの名前を統一しておきます。



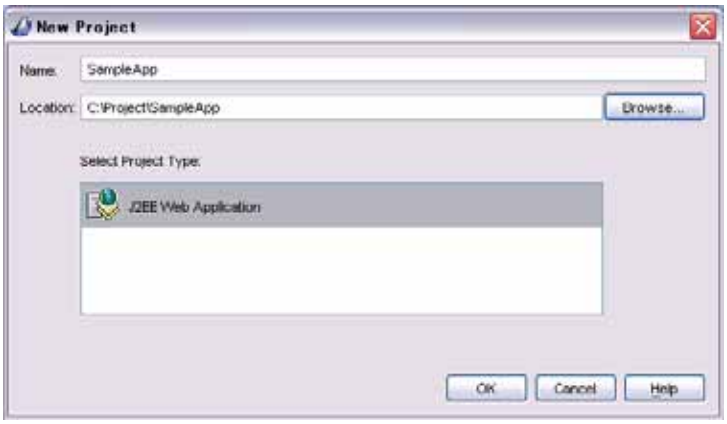
①	サーバ ナビゲーター
②	パレットエリア
③	編集エリア
④	プロパティエリア
⑤	プロジェクトナビゲーター
⑥	ドキュメントエリア

3. サンプルアプリケーションの作成

3.1. プロジェクトの作成

Welcome画面上の[Create New Project]をクリックします。

プロジェクト生成画面が開くので、名称、保存場所を指定して[OK]を押します。



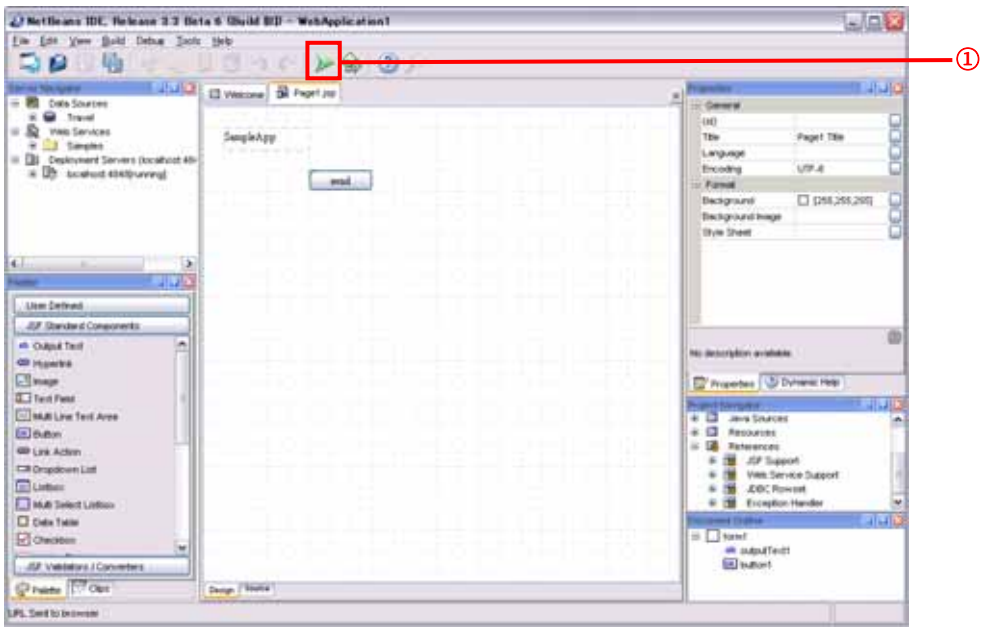
プロジェクトが生成されると Page1.jsp が開かれます。

早速試しに画面を表示してみましょう。

パレットエリアで[JSP Standard Components]を選択し、編集エリアに配置します。ここでは[Output Text]と[Button]を配置します。

配置したコンポーネントを選択すると、プロパティエリアに設定可能なプロパティが表示されます。Valueプロパティを変更すると表示文字列を変更できます。

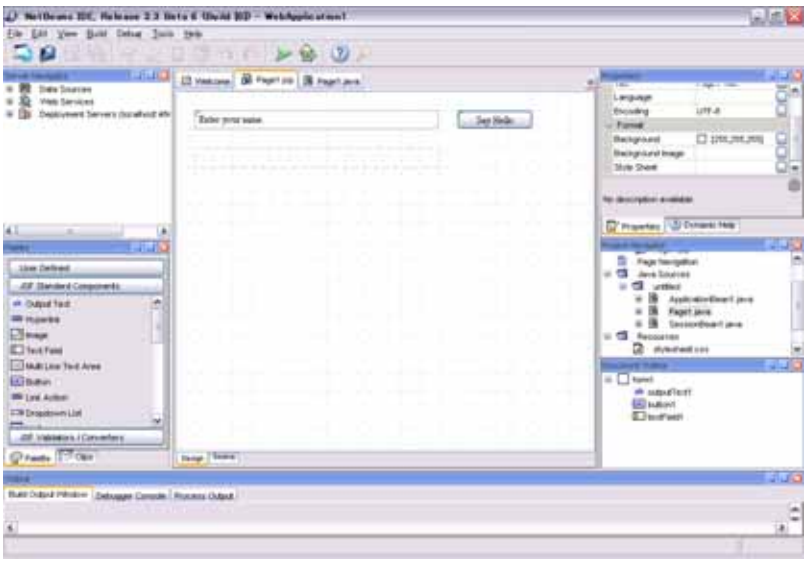
[Run Project(下図中①)]を押すと、付属のアプリケーションサーバが起動し、画面が表示されます。



3.2. 画面生成

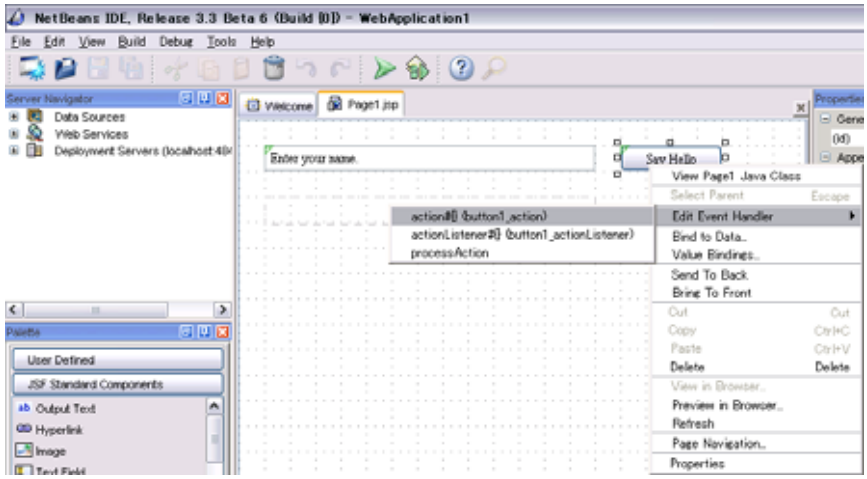
① 前項と同様に[JSP Standard Components]から以下のコンポーネントを配置します。

コンポーネント種別	コンポーネント名称	プロパティ
Text Field	textField1	value="Enter your name"
Output Text	outputText1	
Button	button1	value="Say Hello"



② ボタンを押下時のアクションを定義します。

[button1 を右クリックし、[Edit Event Handler]-[action#{} (button1_action)]を選択します。

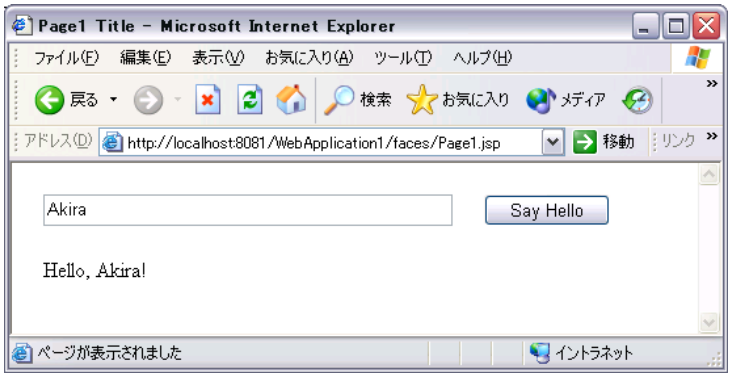


③ 画面に対して自動生成された Java Page Bean が表示されるので以下の記述を追加します。

```
/**
 * button1 押下時のアクション
 * 入力フィールドに入力された名称を使用して出力フィールドにメッセージを表示します。
 */
public String button1_action(){
    String name = (String)textField1.getValue();
    name = "Hello, " + name + "!";
    outputText1.setValue(name);
    return null;
}
```

④ [Run Project]を押すと画面が表示されます。

入力フィールドに名前を入れて、[Say Hello ボタン]を押すと、メッセージが表示されます。



3.3. コンバータの利用

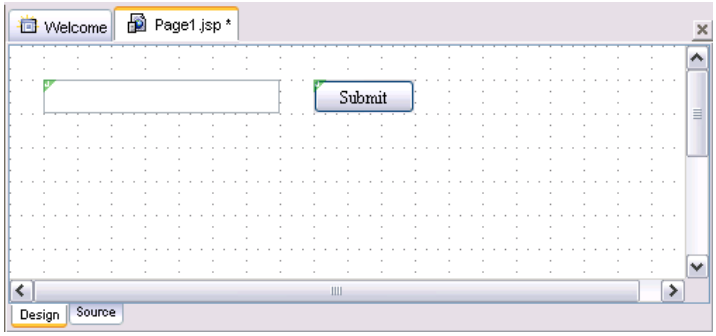
コンバータはユーザの入力フォーマットと、サーバーサイドのJavaで扱うデータフォーマットとの変換を行います。

現在標準で利用できるコンバータは以下になります。

- BooleanConverter
- ByteConverter
- CharacterConverter
- DateTimeConverter
- DoubleConverter
- FloatConverter
- IntegerConverter
- LongConverter
- NumberConverter
- ShortConverter

それではコンバータの動作をみてみましょう。

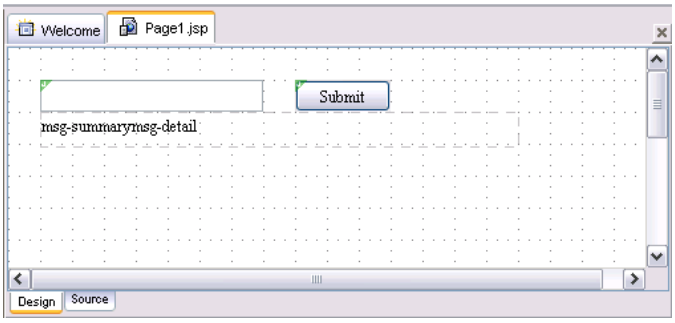
① 以下のようにコンポーネントを配置します。



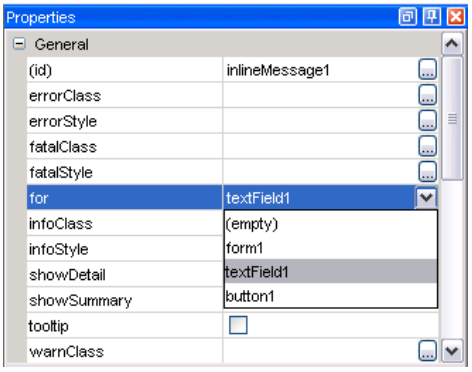
② パレットエリアで[JSF Validators / Converters]を選択し、[Integer Converter]を入力フィールド上にドラッグ＆ドロップします。

続いて、変換に失敗した場合にエラーメッセージを表示するための設定を行います。

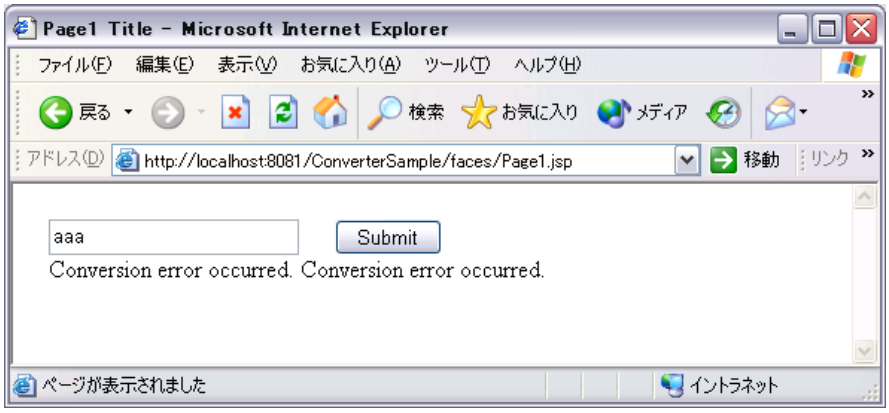
③ パレットエリアで[JSF Standard Components]を選択し、[Inline Message]を配置します。



④ 配置した[Inline Message]を選択し、[for]プロパティで入力フィールドを選択します。



⑤ 実行し、数値以外を入力してボタンを押すとエラーメッセージが表示されます。



3.4. バリデータの利用

バリデータは入力値のチェックという点においてはコンバータと機能的に似ていますが、データ型の変換を行わないという点で異なります。

もし入力値のLengthチェックとデータ型の変換を行いたい場合にはコンバータとバリデータを同一のコンポーネントに関連付けることができます。

現在標準で利用できるバリデータは以下になります。

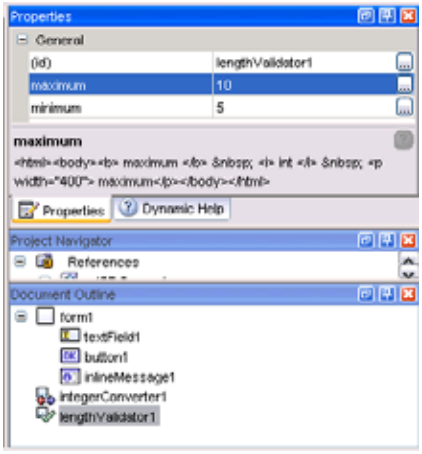
- DoubleRangeValidator
- LengthValidator
- LongRangeValidator

DoubleRangeValidator および LongRangeValidator は Double もしくは Long として入力値が指定された範囲内で有効かどうかをチェックし、LengthValidator は入力値の長さをチェックします。

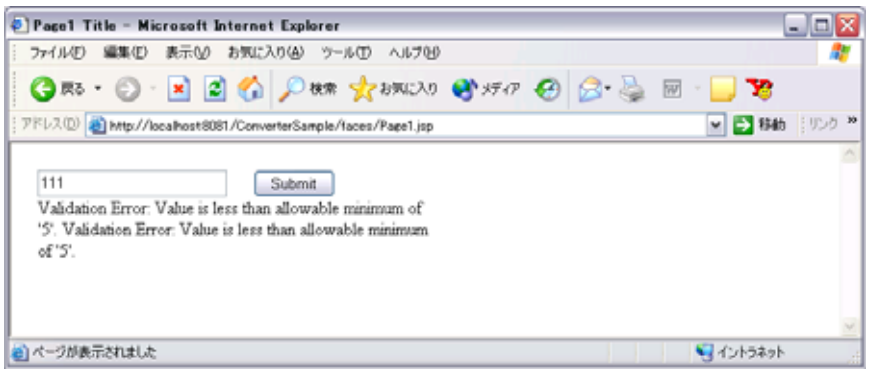
では、バリデータについても動作を確認してみましょう。

① パレットエリアで[JSF Validators / Converters]を選択し、[Length Validator]を入力フィールド上にドラッグ＆ドロップします。

② ドキュメントエリアで[lengthValidator1]をクリックし、プロパティの[maximum]に”10”、[minimum]に”5”を設定します。



③ 実行し、入力フィールドに5～10桁以外の数値を入力してボタンを押した場合にエラーメッセージが表示されます。



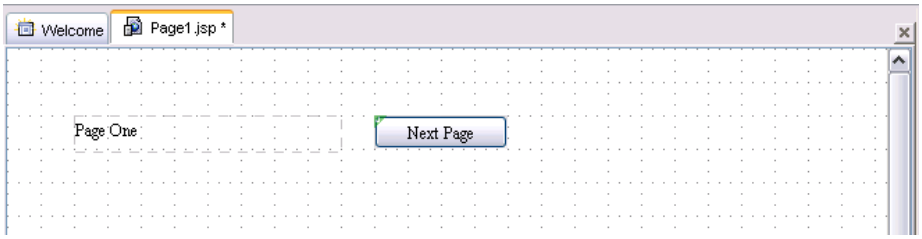
3.5. 静的画面遷移

続いて画面遷移の設定方法について説明します。

なお、本項で説明するのは静的な画面遷移設定です。動的な画面繊維については『3.6 動的画面遷移』を参照してください。

(A) 遷移元画面の作成

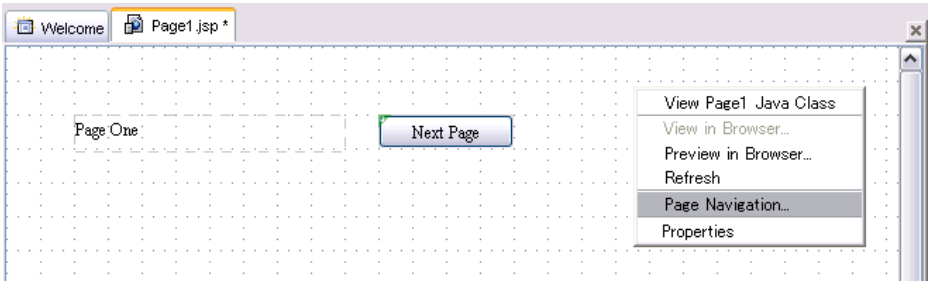
① 新たにSimpleNavigationというプロジェクトを作成し、以下のようにラベルとボタンを配置します。



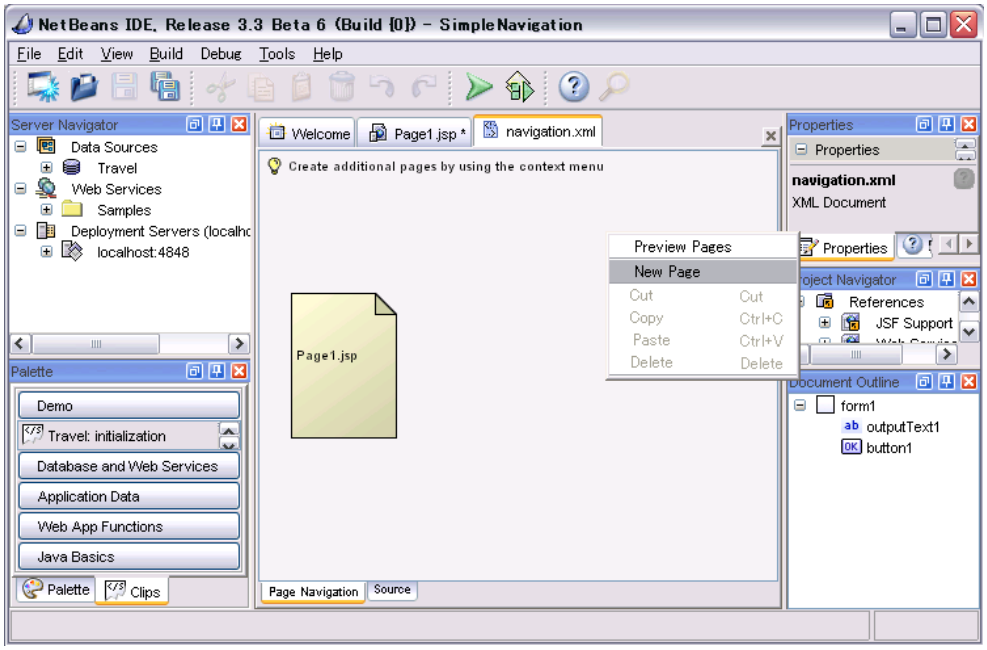
② ボタン押下時のアクションに以下を記述します。

```
/**
 * button1 押下時のアクション
 * @return 次画面遷移のための文字列
 */
public String button1_action() {
    return "nextAction";
}
```

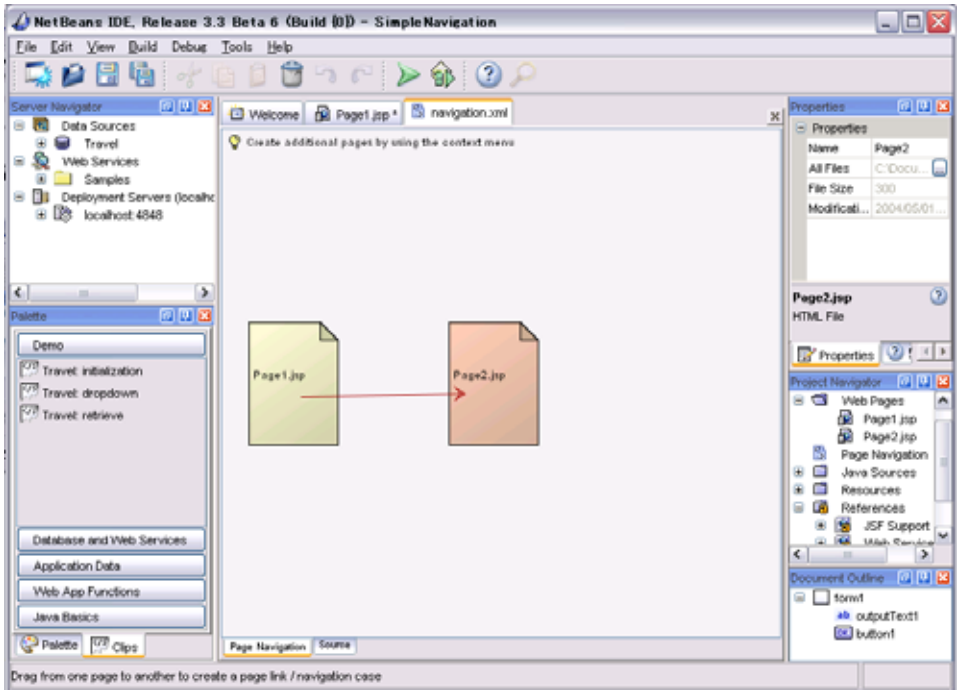
③ 編集エリアを右クリックし、メニューから[Page Navigation...]を選択すると編集エリアに画面遷移定義の編集エリアが表示されます。



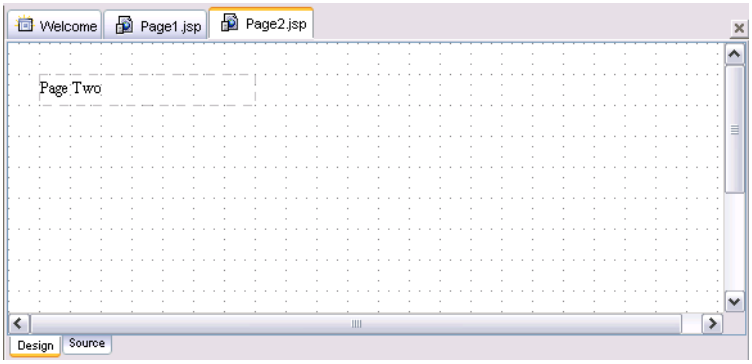
④ 編集エリアを右クリックし、メニューから[New Page]を選択し、ページ名称に[Page2]を指定して[OK]を押します。



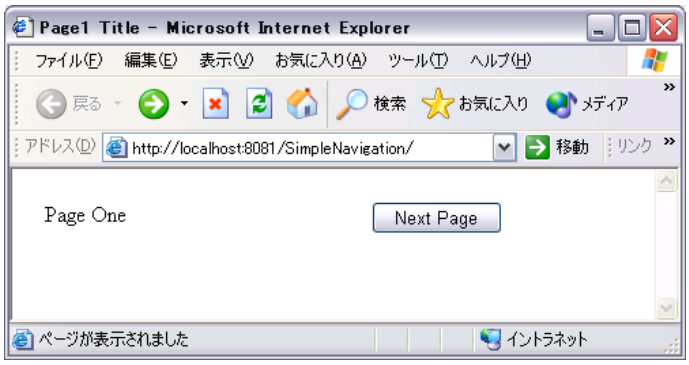
⑤ 編集エリアに[Page2.jsp]が表示されたら、[Page1.jsp]をクリックして[Page2.jsp]まで関連線を引きます。表示された関連線の名称を"nextAction"と変更します。



⑥ [Page2.jsp]をダブルクリックして編集画面を開き、以下のようにコンポーネントを配置します。



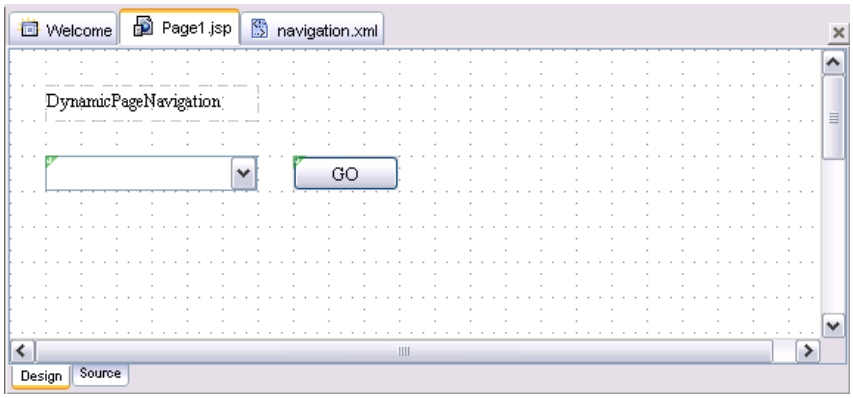
⑦ 実行すると[Page1.jsp]が表示されます。[Next Page]を押して[Page2.jsp]が表示される事を確認します。



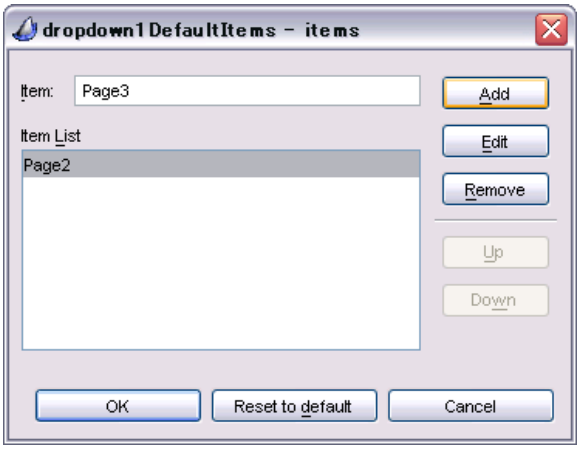
3.6. 動的画面遷移

本項ではユーザ操作によって動的に遷移先画面を変更する際の設定方法について説明します。

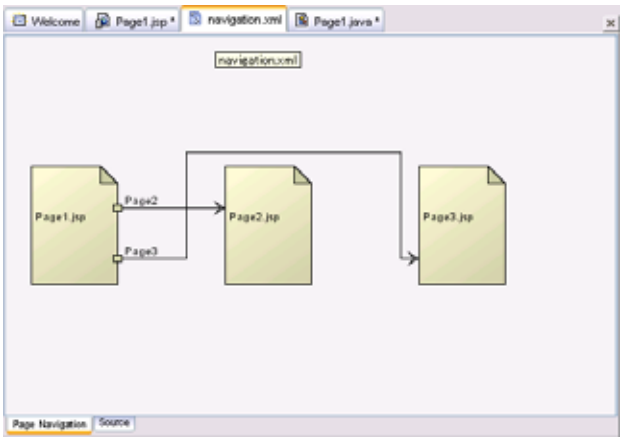
① 新たに DynamicNavigationというプロジェクトを作成し、以下のようにコンポーネントを配置します。



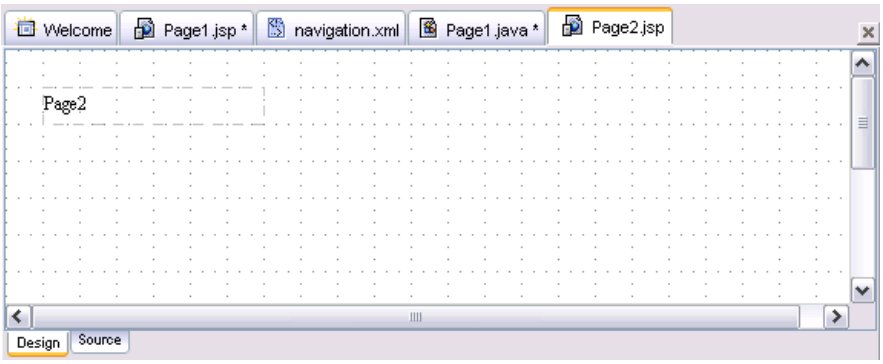
② ドキュメントエリアで[dropdown1DefaultItems]を選択し、[items]プロパティの元の内容をクリアし、”Page2”,”Page3”を追加します。

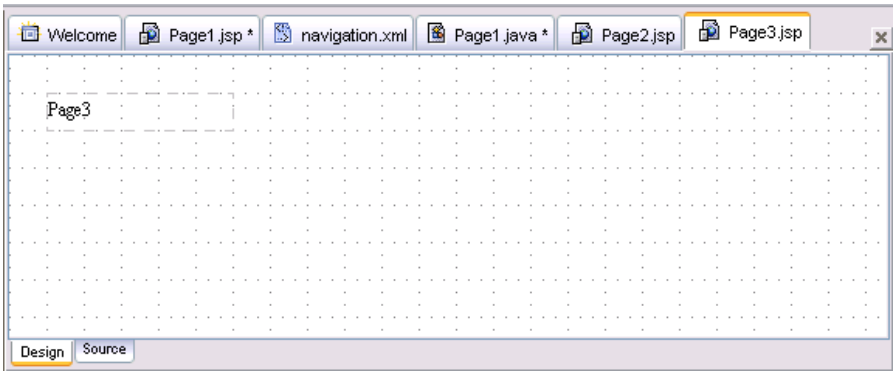


③ 先ほどと同様にページナビゲーションを開き、以下のように定義します。



④ [Page2.jsp]、[Page3.jsp]を作成しそれぞれ以下のように作成します。

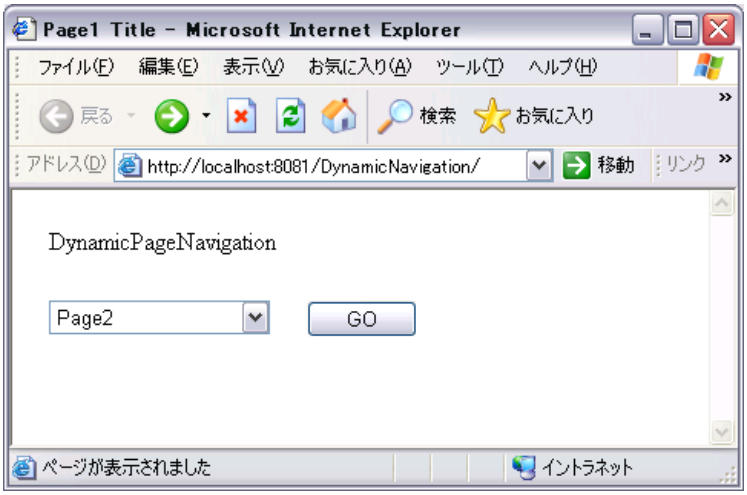




⑤ ボタン押下時のアクションに以下を記述します。

```
/**
 * button1 押下時のアクション
 * @return  次画面遷移のための文字列
 */
public String button1_action() {
    return (String)dropdown1.getValue();
}
```

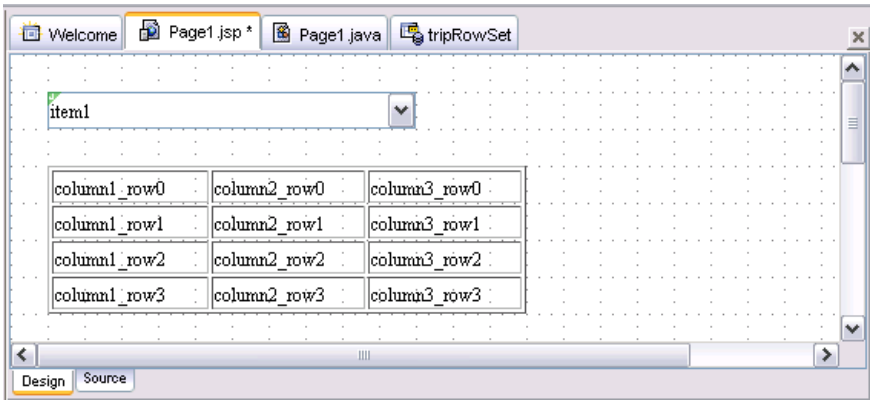
⑥ 実行して、選択内容によって遷移先画面が変わっている事を確認します。



3.7. データアクセス

本項ではJSFコンポーネントからデータベースアクセスを行う方法について説明します。

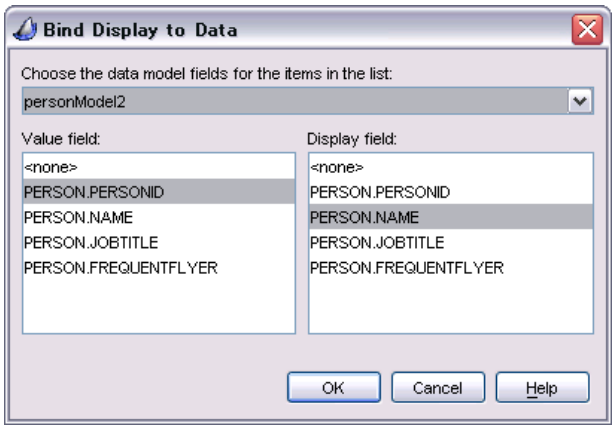
① 新たに"DataAccess"というプロジェクトを作成し、以下のように[Dropdown List]と[Data Table]を配置します。



② [サーバ ナビゲータ]で[PERSON] を選択し、[dropdown1]へドラッグ＆ドロップします。

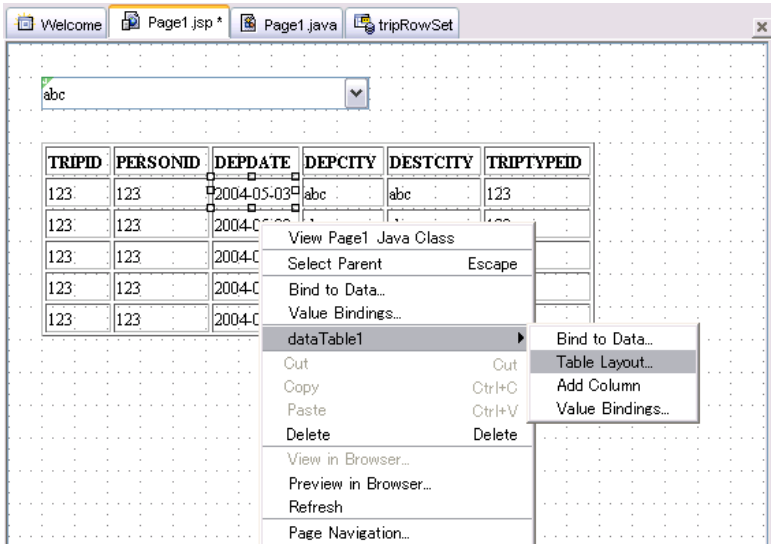
③ ダイアログが開きますので"Fill the List"を選択すると、[dropdown1]の表示が"abc"と変わります。

④ [dropdown1]を右クリックし、[Bind Display to Data...]を選択すると、ダイアログボックスが表示されますので、[Value field:]に"PERSON.PERSONID"、[Display field:]に"PERSON.NAME"を選択し、[OK]を押します。

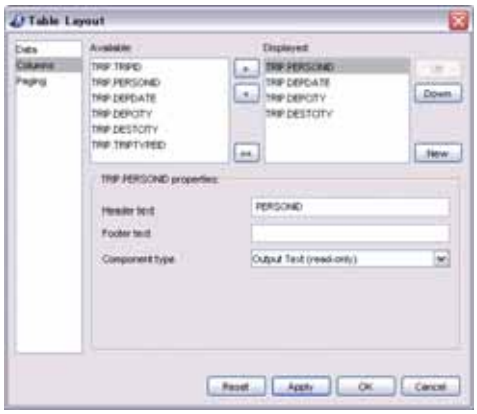


⑤ 続いて[サーバ ナビゲータ]で[TRIP] を選択し、[dataTable1]へドラッグ＆ドロップします。

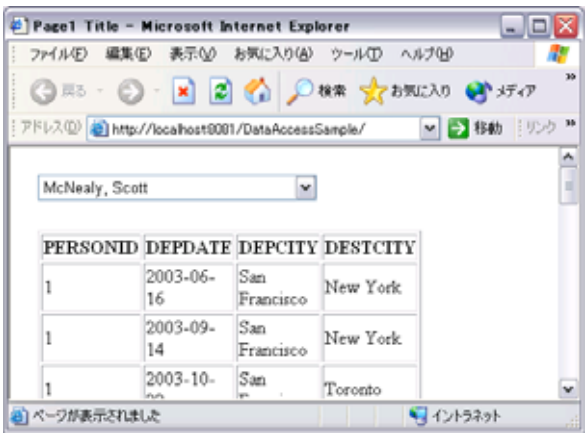
⑥ [dataTable1]を右クリックし、[Table Layout...]を選択します。



⑦ 表示項目と各カラムの[Header text][Footer text]が変更できますので、"TRIP.TRIPID"と"TRIP.TRIPTYPEID"を[Displayed]から削除します。

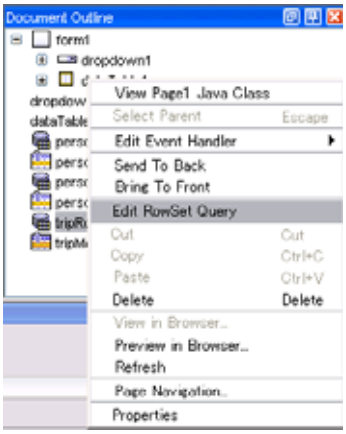


⑧ ここまでで実行すると、以下の様にテーブルの内容を反映した画面が表示されます。

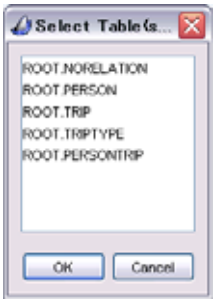


続いてクエリのカスタマイズを行います。

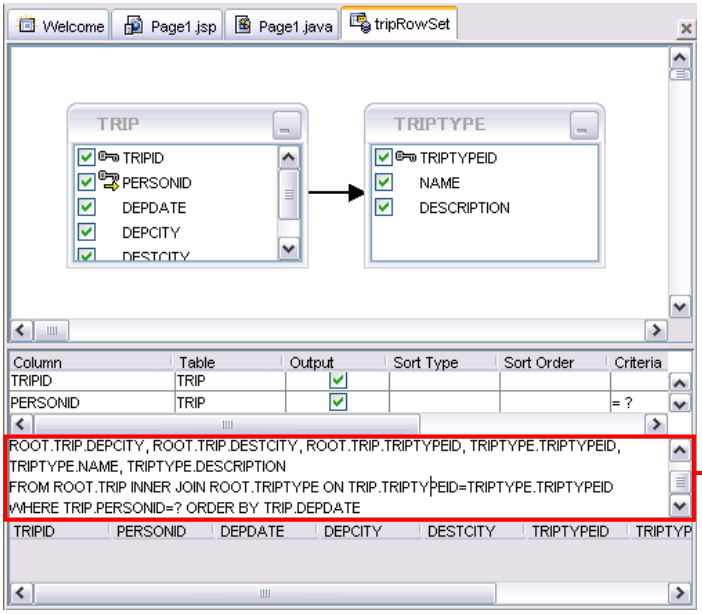
[ドキュメントエリア]で[tripRowSet]を右クリックし[Edit RowSet Query]を選択します。



⑨ クエリ編集画面が開きますので、右クリックし[Add Table...]から"ROOT.TRIPTYPE"を追加します。



⑩ [クエリ編集部(下図中①)]で任意の記述を追加できます。本例では" ORDER BY TRIP.DEPDATE"を追加してあります。

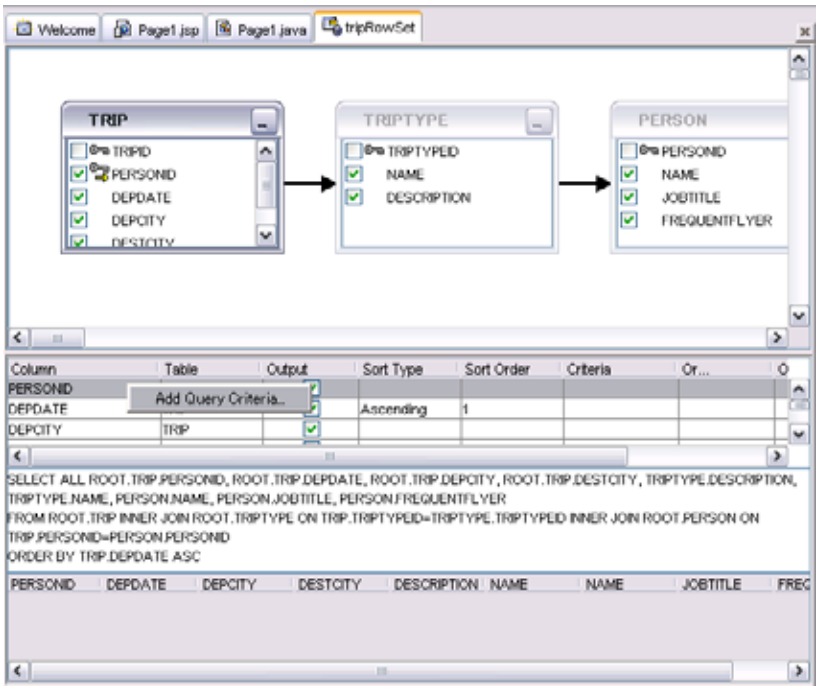


また、編集エリアを右クリックしメニューから[Run Query]を選択すると、現在編集中のクエリを実行する事ができます。

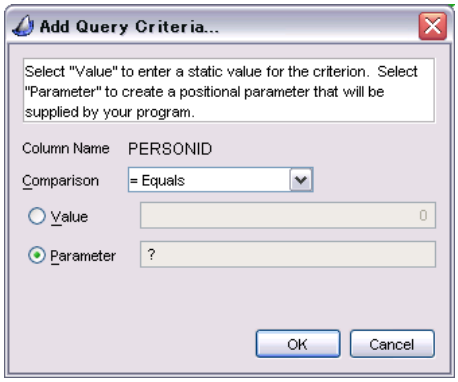
⑪ 再び[Table Layout...]を開き表示内容を変更することで出力内容を変えることができます。

続いて、[dropdown1]で選択された人物のみを[dataTable1]で表示するようにコンポーネントを関連付けます。

⑫ 以下の様に[PERSONID]を右クリックし、[Add Query Criteria...]を選択します。



⑬ ダイアログが開きますので、[Parameter]を選択し、[OK]を押すと、クエリに" WHERE TRIP.PERSONID=?"が追加されます。



⑭ [Page1.jsp]に戻り、[dropdown1]をダブルクリックし、以下の内容を追加します。

[パレットエリア]の[demo]から[Travel:dropdown]をドラッグ＆ドロップしても同様のコードが追加されます。

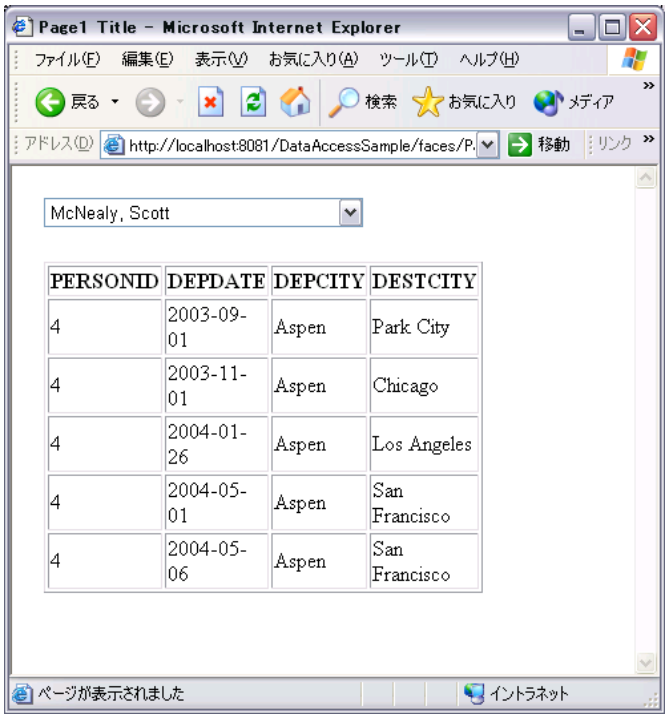
```
/*
 * dropdown1 の値が変更されたときに呼ばれるメソッド
 *
 * 選択された内容で tripRowSet を更新する。
 */
public void dropdown1_valueChangeListener(javax.faces.event.ValueChangeEvent vce) {
    try {
        tripRowSet.setObject(1, dropdown1.getValue());
        tripRowSet.execute();
    } catch (Exception ex) {
        throw new FacesException(ex);
    }
}
```

⑮ 同様に[Page1.java]のコンストラクタに以下の記述を追加します。

[パレットエリア]の[demo]から[Travel:initialization]をドラッグ＆ドロップしても同様のコードが追加されます。

```
    try {
        personRowSet.execute();
        personRowSet.next();
        tripRowSet.setInt(1, personRowSet.getInt("PERSONID"));
        tripRowSet.execute();
    } catch (Exception ex) {
        throw new FacesException(ex);
    }
```

⑯ 実行すると[dropdown1]で選択した内容のみ表示されます。



4. TIPS

(1) ログの使用方法

ログを使用する場合は、以下のように記述します。

```
log("Page1 Initialization Failure");
```

ログは#INSTALL_DIR#¥Creator¥SunAppServer8¥domains¥domain1¥logs¥server.log に出力されます。

5. 最後に

Sun Java Studio Creator のマニュアルに従いざっと基本機能の使用方法を追ってみましたがいかがでしょうか。

JSFは確かに現在の WEB アプリケーションが抱える問題点を解決できる技術要素ですが、エディタで編集するのはかなり面倒なので、このようなGUIツールがあると導入の敷居も下がると思います。

しかし、まだ不具合も残っているようで、開きなおすと画面レイアウトが崩れてしまったりしますが、WEB アプリケーションに特化しているだけあって、正式版の価格次第では今後流行るかもしれません。

時間と要望があればデバッグ、独自コンポーネントの追加方法や出力メッセージのカスタマイズ方法などなど今後記述を追加していこうと思います。

記述内容に何かお気づきの点、質問等ありましたら下記までご連絡ください。

開発部
横井 朗
yokoi@bbreak.co.jp