Oracle9i JDeveloper を利用した Struts アプリケーション開発

Creation Date: 2002-08-26
Last Update: 2002-11-29
Version: 1.1

目次

はじめに

本書の目的は、Oracle9i JDeveloper を利用した Struts アプリケーションを作成するための一連の手順を説明することです。Oracle9i JDeveloper の導入から Struts の説明、そしてサンプルプログラムの作成・コンパイル・実行までを説明します。サンプルプログラムは入力した数値を計算して、その結果を表示するプログラムです。

なお、本書中のオラクル製品およびプロジェクトのプラットフォームはWindows2000を使用して動作確認を行いました。

構成

本書は、次の各章で構成されています。

『1. Struts とは?』

Jakarta プロジェクトの Struts について説明します。

『2. JDeveloper での Struts のための設定』

Oracle9i JDeveloper を Struts アプリケーション開発のために設定する方法について説明します。

『3. サンプルプログラムの概要』

Struts を使ったサンプルプログラムの概要について説明します。

『4. サンプルプログラムの作成』

Oracle9i JDeveloper で Struts を使用したサンプルプログラムを作成する手順を説明します。

『5. アプリケーション・サーバーへの配布（デプロイ）』

Oracle9i Application Server の J2EE コンテナである OC4J（Oracle9iAS Containers for J2EE）にサンプルプログラムを配布する方法とサンプルプログラムの実行方法について説明します。

1. Struts とは?

この章では、Jakarta プロジェクトの Struts について説明します。Struts とは何なのか、また、その Struts を使うメリット、そして、Struts アプリケーションの処理の流れを簡単に説明します。

1.1 Struts とは?

アプリケーションの基本的な機能をまとめて提供するものに、フレームワークというものがあります。Struts は、Apache Software Foundation がサポートする Jakarta プロジェクトの一部で、オープンソースの Web アプリケーション・フレームワークを提供しています。

海外では、この Struts を Web アプリケーションのフレームワーク製品の開発ベースとして、商用でも利用されているケースも少なくありません。

Struts では、MVC（Model-View-Controller）モデルを採用し、プレゼンテーション・ロジックとビジネス・ロジックを切り離しています。MVC モデルは、次の３つに区別されます。

1. Model
   Java Beans や EJB などでビジネス処理を行う。
2. View
   JSP などでプレゼンテーション処理を行う。
3. Controller
   Servlet などで Model と View の中間的な役割を行う。

この MVC モデルの採用によって、開発チームを明確に区別し効率よく開発を行うことができるようになります。

また、Struts は、100% Pure Java で構成されており、動作環境を選びません。

1.2 Struts を使うメリット

MVC モデルを使ったフレームワークは、Struts 以外にもあります。では、Struts を使うメリットは何なのでしょうか。ここでは、そのメリットについて述べます。

・画面遷移が容易

Struts は、画面遷移を容易に行えるように設計されています。アクション・マッピングを用いることによって、struts-config.xml ファイルを修正するだけで画面遷移先を変更することができます。これにより、例えば、各画面の処理結果を表示する成功画面への遷移先を簡単に修正することが可能になります。

・Struts タグ・ライブラリ

Struts では、タグ・ライブラリというものがあり、JSP の作成を容易にします。標準で、様々なタグ・ライブラリが提供されており、メッセージの表示や判定処理、JSP-Action 間のデータ連携などが簡単に行えます。また、このタグ・ライブラリは新たな機能を独自に作成することも可能です。

・シンプルなフレームワーク

Struts では、基本的な機能のみを提供するようにしてあり、必要に応じてモジュールを追加していくことが簡単に行えます。さらに、Java の世界で標準的にサポートされる国際化機能を利用するため、メッセージ・リソース・ファイルも利用できます（サンプルプログラムとして後述）。

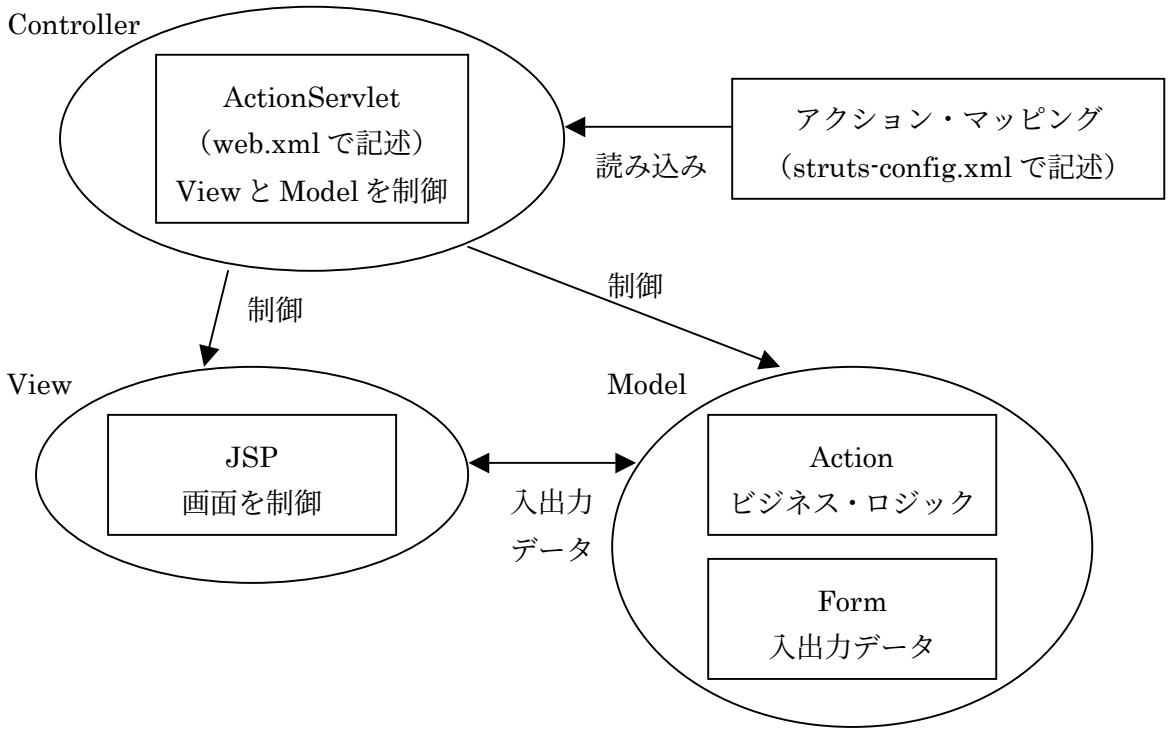
1.3 Struts アプリケーションにおける処理の流れ

では、Struts アプリケーションについて、簡単な処理の流れを紹介しましょう。

デフォルト環境では、ActionServlet と呼ばれるクラスがアクション・マッピングを読み込んでリクエストに応じた画面を呼び出します。ActionServlet は、MVC モデルの Controller にあたります。また、アクション・マッピングとは、リクエスト毎に割り当てられた情報を定義してあるものです。

通常、このアクション・マッピングは、struts-config.xml ファイルに定義されています。このファイルでは、リクエスト毎に、JSP と Action、そして Form が一括りで定義されます。JSP は、MVC モデルの View にあたり、Action と Form は、MVC モデルの Model にあたります。JSP が画面表示を行ない、Action が画面ごとのビジネス・ロジックを行います。また、JSP-Action とのデータ連携に Form を用います。

Struts アプリケーションの構造を図で示すと次のようになります。



処理の流れは、次のようになります。

① クライアントがブラウザから特定の URL にアクセスする。

② アクション・マッピングを参照して、対応される Action と Form オブジェクトを生成する。web.xml に記述される ActionServlet を起点として、View の JSP、Model の Action・Form オブジェクトを生成する。生成されるオブジェクトは、struts-config.xml で指定されたアクション・マッピングに対応する Action と Form になる。

③ 生成された Action と Form オブジェクトは、ビジネス・ロジック処理を行ない、JSP に表示する出力データを取得する。

④ ActionServlet から JSP を呼び出す。

⑤ JSP で、②で取得された出力データを画面に表示する。

⑥ クライアントが JSP でデータを入力して送信する。

⑦ Action と Form オブジェクトでクライアントが入力したデータを取得して、ビジネス・ロジック処理を行う。

ここで紹介した処理の流れは一例です。他にも作成方法によって、様々な形態を取ることが可能です。興味がある方は次の Jakarta プロジェクトのサイトを参考にしてください。

http://jakarta.apache.org/struts

2. JDeveloper での Struts のための設定

この章では、Struts アプリケーションを効率よく開発するために、Oracle9i JDeveloper 上で設定できる作業について説明します。

3.1 Struts のインストール

Oracle9i JDeveloper で Struts を使用するための設定を行いましょう。

まず、Struts を下記のサイトからダウンロードしてください。2002 年 9 月現在の安定版リリースは Struts 1.0.2 です。

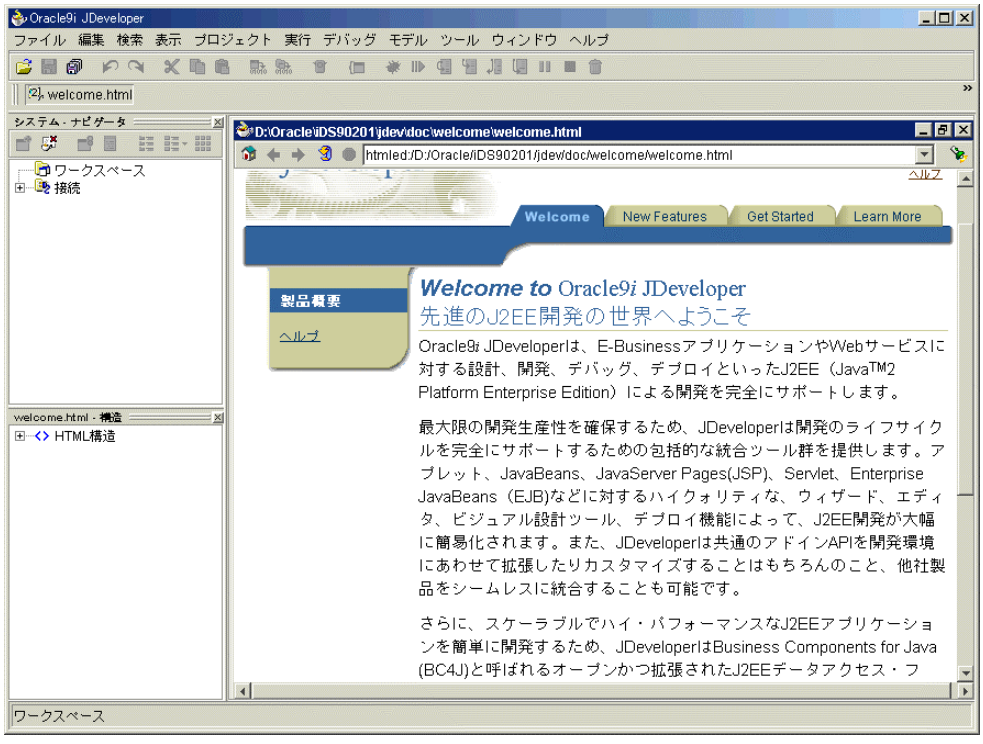
http://jakarta.apache.org/builds/jakarta-struts/release/v1.0.2/

上記サイトにある jakarta-struts-1.0.2.zip をダウンロードして解凍します。本書では、これを D:¥Users¥Struts に解凍しているものとします。この場合、ディレクトリ構成は次のようになります。

```
D:¥Users¥Struts
      |-- jakarta-struts-1.0.2
            |-- lib
            |-- webapps
```

3.2 JDeveloper の実行と設定

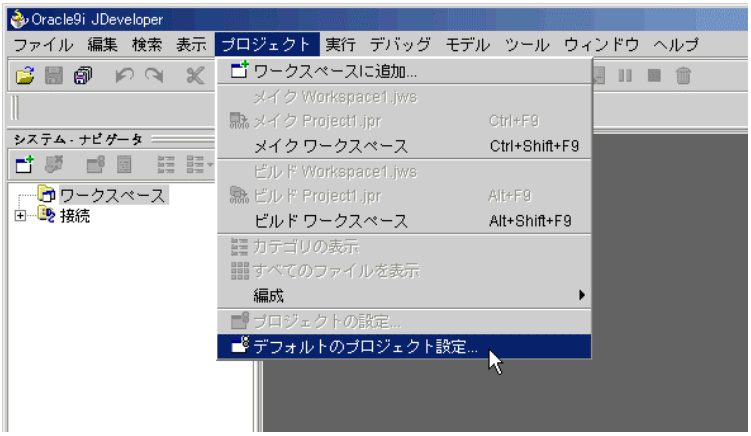
JDeveloper を起動します。以下のような画面が表示されます。



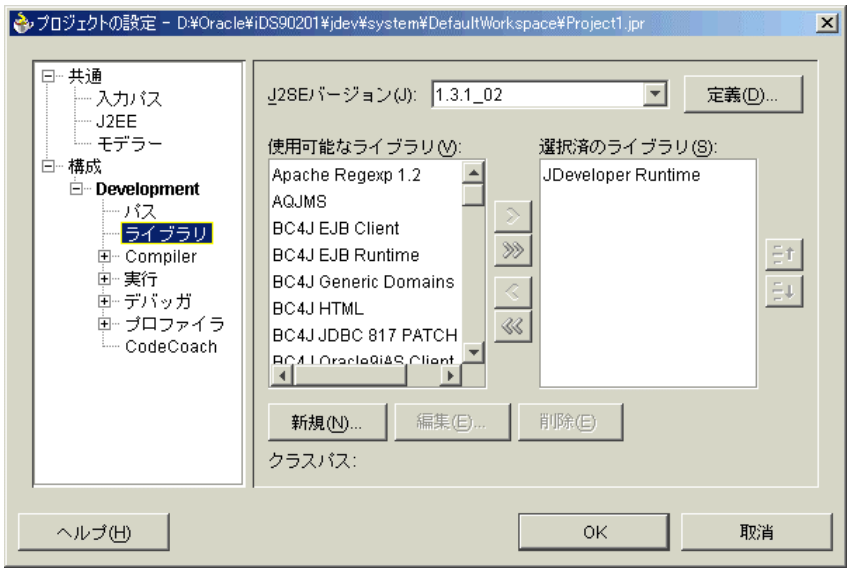
3.3 JDeveloper での Struts に関する設定

Struts ライブラリの登録

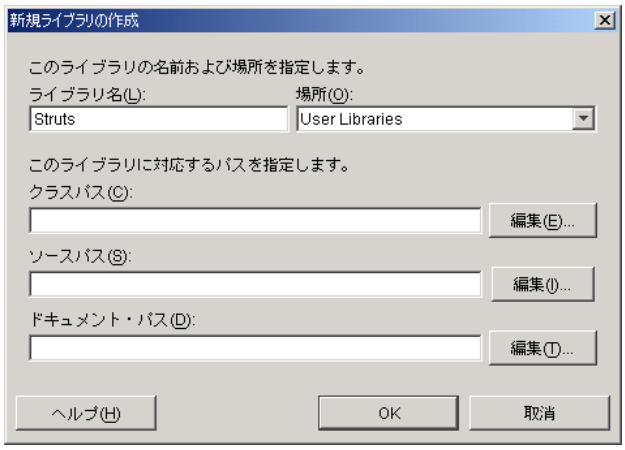
JDeveloper に Struts のライブラリを登録します。メニューから「プロジェクト→デフォルトのプロジェクト設定」を選択します。



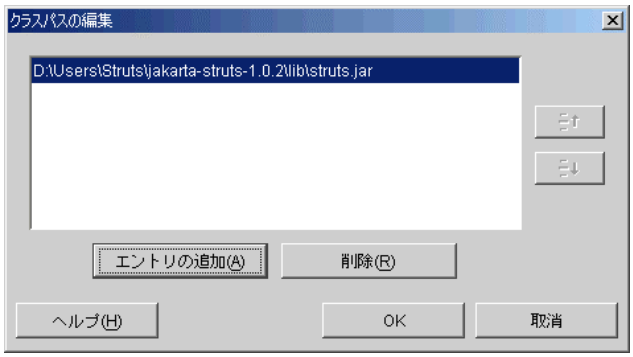
「プロジェクトの設定」ウィンドウが表示されます。左のツリーより「ライブラリ」を選択して「新規」ボタンをクリックします。



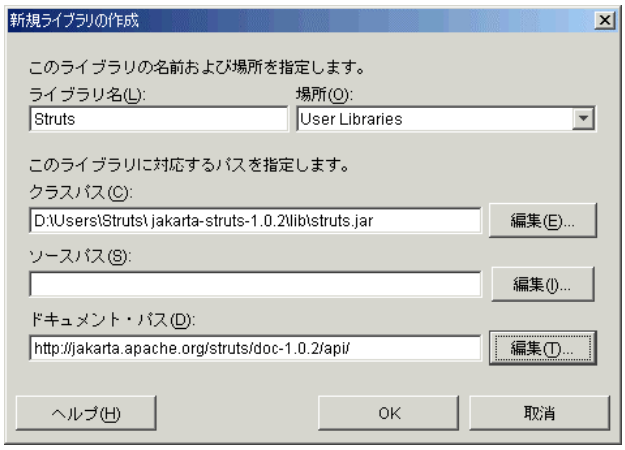
「新規ライブラリの作成」ウィンドウが表示されるので、ライブラリ名を入力します。ここでは、例として「Struts」とします。続いて「クラスパス」右側の「編集」ボタンをクリックします。



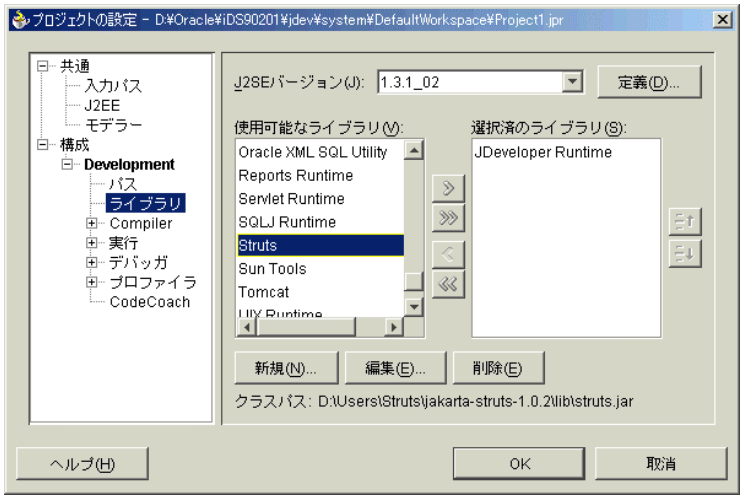
「クラスパスの編集」ウィンドウが表示されるので、「エントリの追加」ボタンをクリックして、先ほど解凍した「D:¥Users¥Struts¥ jakarta-struts-1.0.2¥lib¥struts.jar」ライブラリを追加します。以下の画面のように追加されましたら OK ボタンをクリックしてください。



「新規ライブラリの作成」ウィンドウに戻ったら、続いて「ドキュメント・パス」右側の「編集」ボタンをクリックします。「ドキュメント・パスの編集」ウィンドウが表示されるので、「URL の追加」ボタンをクリックして、Struts 1.0.2 の Javadoc の URL である「http://jakarta.apache.org/struts/doc-1.0.2/api/」を入力し、OK ボタンをクリックします。



「新規ライブラリの作成」ウィンドウに戻ったら、OK ボタンをクリックしてください。「プロジェクトの設定」のライブラリで「選択済のライブラリ」に Struts が追加されます。ここでは、これをいったん「使用可能なライブラリ」のほうへ戻します。



Struts タグ・ライブラリの登録

JDeveloper のコンポーネント・パレットに、Struts のタグ・ライブラリを登録することによって、Struts JSP アプリケーションの開発をより円滑に進められるようになります。

作業内容の詳細は、付録の「Struts 1.0.2 タグ・ライブラリの設定」を参照ください。

この作業は、必須ではありませんが、今後の開発効率を向上させるために実施することをお勧めします。

3. サンプルプログラムの概要

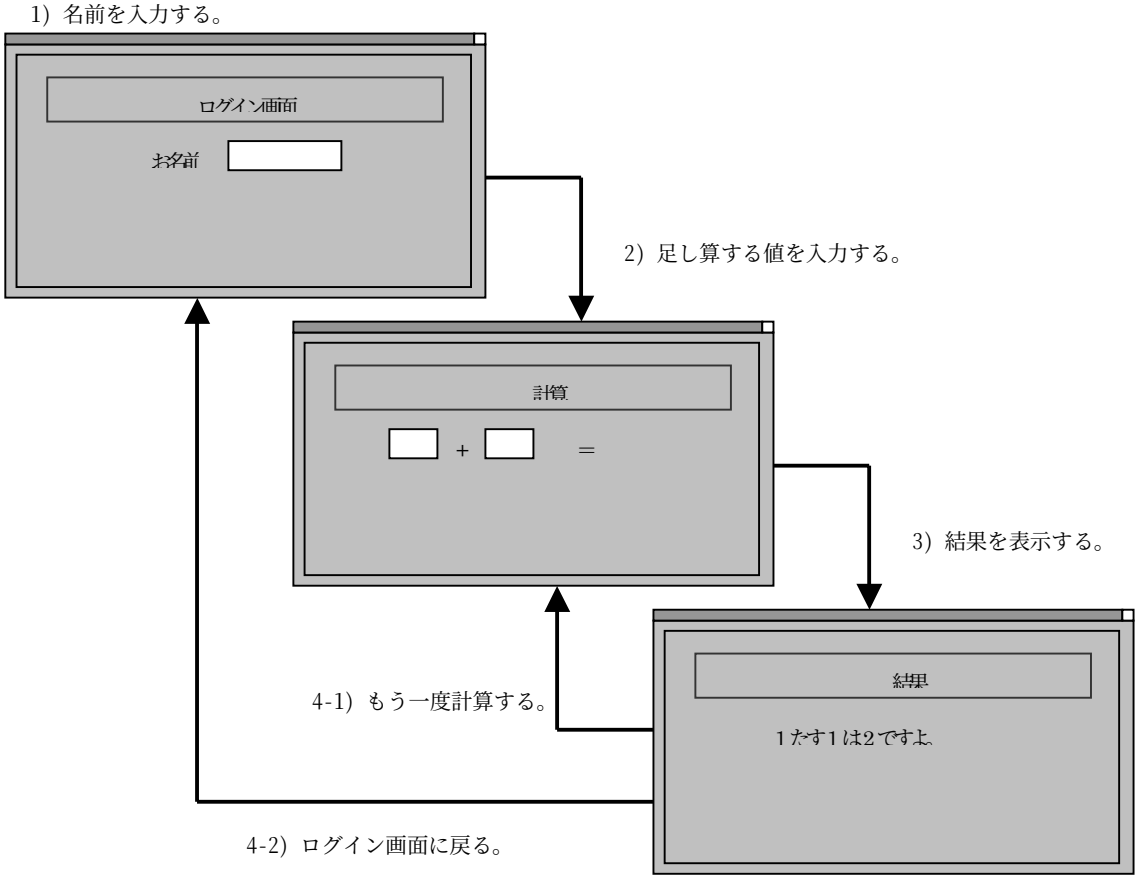
この章では、本書で使用する Struts アプリケーションのサンプルプログラムについて説明します。

3.1 概要

サンプルプログラムは、入力された値で足し算を行ない、その結果を表示するプログラムです。

3.2 画面遷移

サンプルプログラムの画面遷移は、次の図のようになります。



3.3 アプリケーションの構成

上記のような画面遷移および内部の処理を実現するには、どのようなコンポーネントを開発する必要があるのでしょう?

明らかにわかることは、次のコンポーネントが必要だろうということです。

View として

- ログインページ
- 計算入力ページ
- 計算結果ページ

Model として

- ログイン処理
- 計算処理

さらに、以下のことを考慮する必要があります：

- ログインしたユーザー情報を保持するオブジェクトが必要です。
- Struts では、View での入力に対応して、Form オブジェクトでその入力値をチェックし、Action オブジェクトで引き続く処理を行います。これに従うと、ログインの入力値に対する Form オブジェクト、計算入力値に対する Form オブジェクトが必要になります。
- 「計算入力ページ」と「計算結果ページ」は、Struts で提供される JSP カスタム・タグを使用することで、ひとつのページにまとめることができます。

このようなことを踏まえて、今回のサンプルプログラムでは、以下のソースを作成することにします。

JSP ファイル

- ログインページ
- 計算入力&計算結果ページ

java ファイル

- ログイン処理 Form
- ログイン処理 Action
- 計算処理 Form
- 計算処理 Action

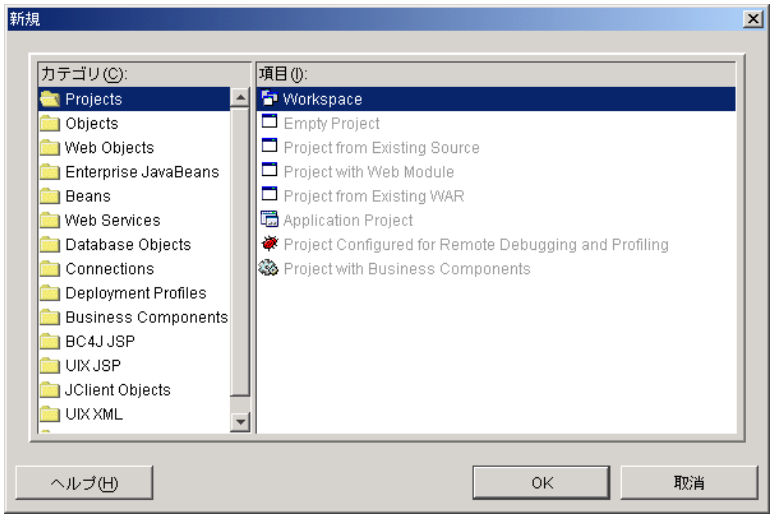
4. サンプルプログラムの作成

この章では、Struts を使ったサンプルプログラムを実際に作成する手順について説明します。

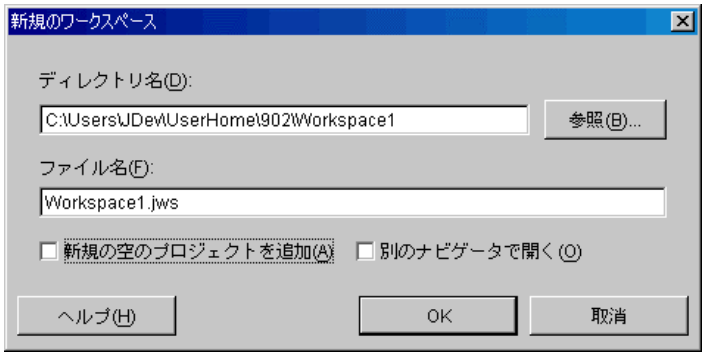
4.1 Struts 用にプロジェクトを作成

Oracle9i JDeveloper のメニューの「ファイル」から「新規」を選びます。

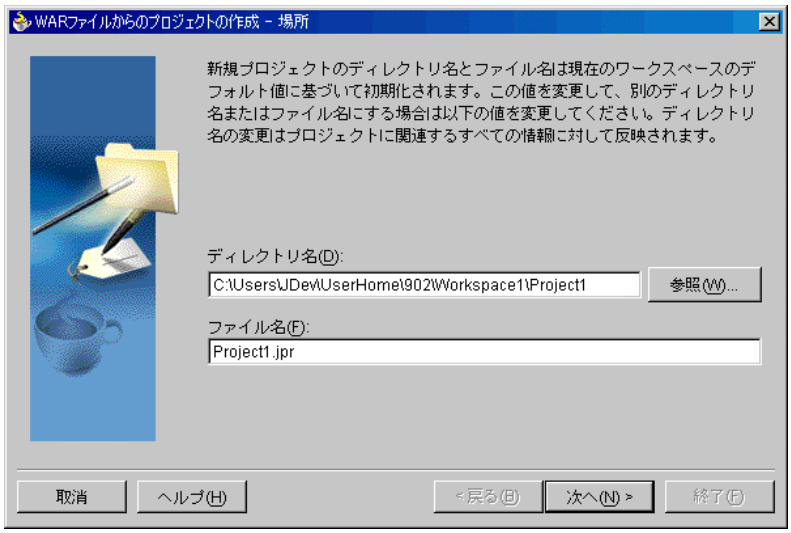
「新規」ウィンドウでカテゴリ「Projects」、項目「Workspace」を選択して OK ボタンをクリックします。



次に、「新規のワークスペース」ウィンドウで、ディレクトリ名、ファイル名を確認して OK ボタンをクリックします。「新規の空のプロジェクトを追加」のチェックは外しておきます。



次に、「新規」ウィンドウでカテゴリ「Projects」、項目「Project from Existing WAR」を選択して OK ボタンをクリックします。「WAR ファイルからのプロジェクトの作成」ウィザードが表示されます。

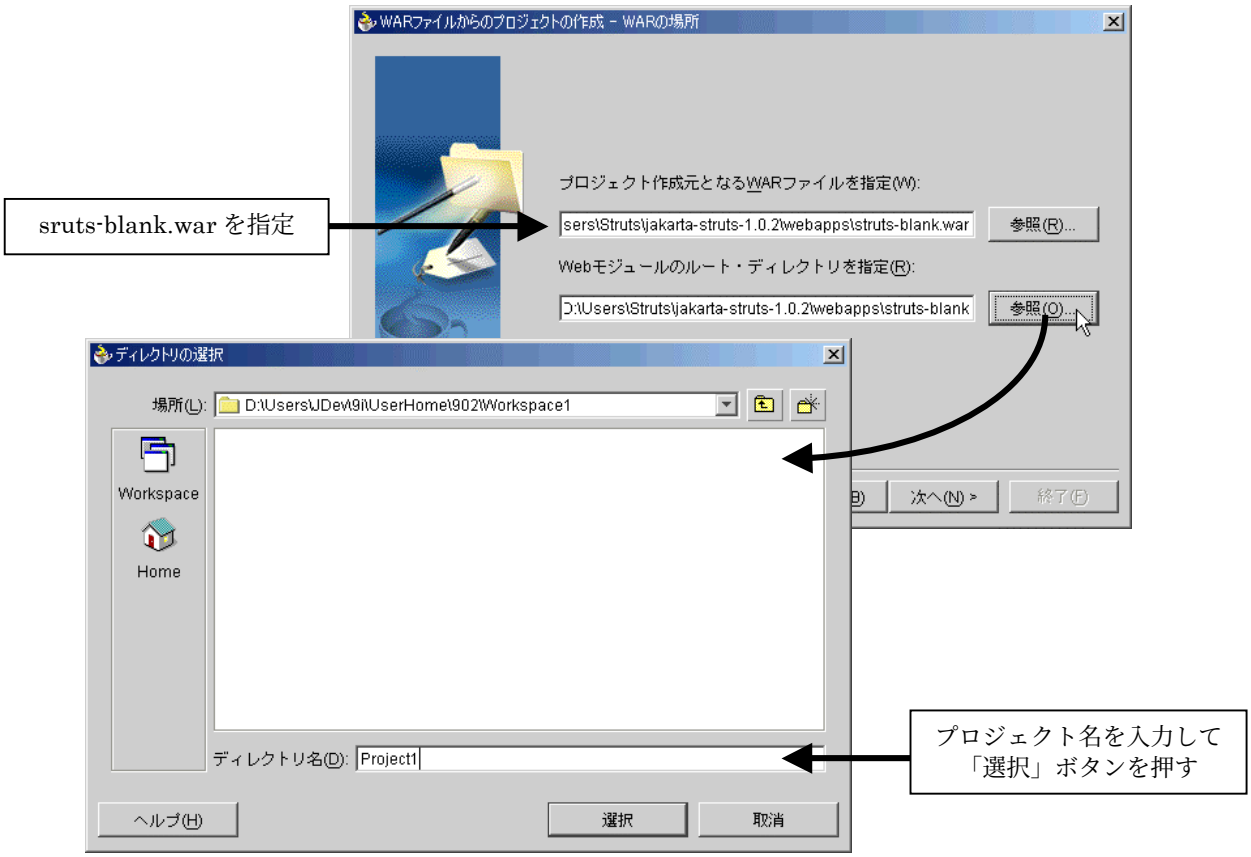


最初のページでは、プロジェクトに関する基本情報を設定します。「次へ」をクリックして次のページへ進みます。

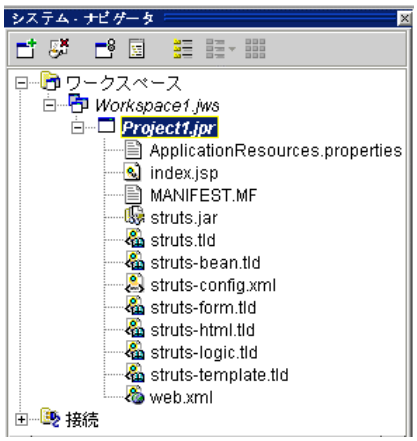
次のページでは、プロジェクトのベースにする WAR ファイルを選択できます。ここで、「プロジェクトの作成元となる WAR ファイルを指定」に Struts に付属の struts-blank.war を選択します（参照ボタンから Struts をインストールしたディレクトリ（D:¥Users¥Struts）配下の jakarta-struts-1.0.2¥webapps¥struts-blank.war を選択します）。

同ディレクトリには、Struts 独自の設定やリソース・ファイルのさまざまな例が用意してあり、有用です。もし、新規に Struts アプリケーションを作成する場合には、Jakarta のサイトを参考にしながら、この WAR ファイルに入っているファイルの中身も参考にすることをお勧めします。

その下の「Web モジュールのルート・ディレクトリ」が、自動的に選択された WAR ファイルのディレクトリ位置に設定されます。これをプロジェクトのディレクトリに変更します（参照ボタンを押すと、「ディレクトリの選択」ダイアログでワークスペースのディレクトリが表示されますので、そこでプロジェクト名を入力すると簡単です）。



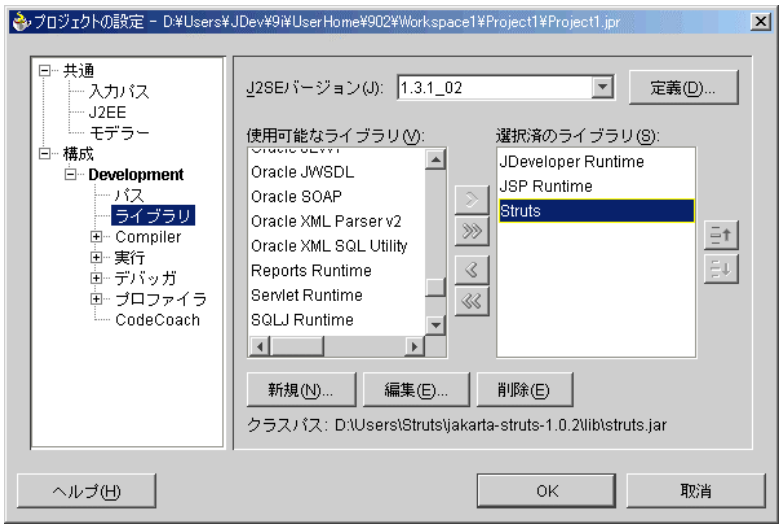
「次へ」を押すと、ウィザードの最終ページになります。「終了」をクリックします。これにより、Struts の基本設定が完了したプロジェクトが作成されます。



では、プロジェクトが作成されたので、プロジェクトの設定を行いましょう。Project1.jpr を右クリックして、「プロジェクトの設定」を選択します。

Struts アプリケーション開発用にライブラリを追加

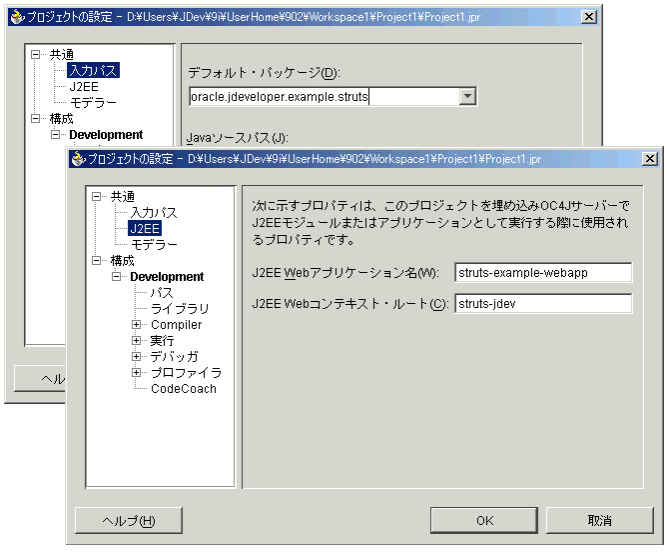
「Struts」ライブラリを選択して、「選択済のライブラリ」に移動させて、OK をクリックします。



その他の基本設定

「共通－入力パス」の「デフォルト・パッケージ」に、これから作成するパッケージ名を入力します。ここでは、例として「oracle.jdeveloper.example.struts」とします。

次に、「共通－J2EE」の「J2EE Web アプリケーション名」に「struts-example-webapp」、「J2EE Web コンテキスト・ルート」に「struts-jdev」と入力します（「J2EE Web コンテキスト・ルート」の値が、アプリケーション完成後にブラウザからアクセスするときのルート名になります）。入力したら OK ボタンをクリックしてください。



4.2 サンプルプログラムの作成 1

では、実際にサンプルプログラムを作ってみましょう。

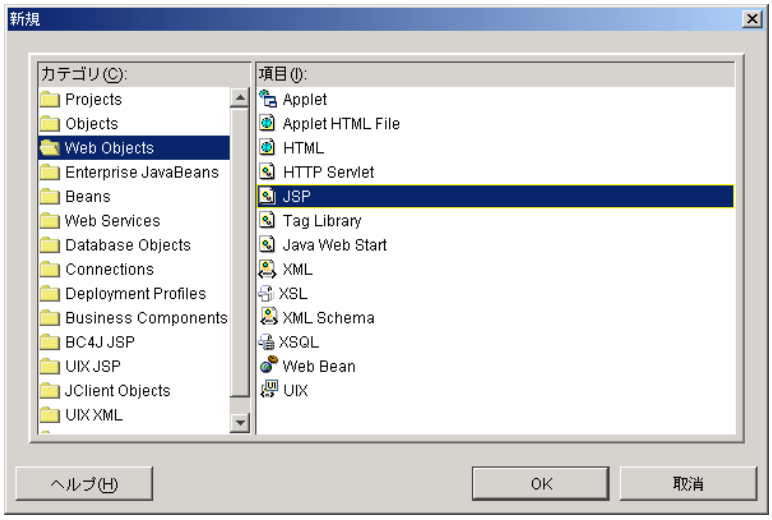
以下の順番で、各ソース・ファイルを作成します。

① ログオンページの作成
② ユーザー情報オブジェクトの作成
③ ログオン用 Form オブジェクトの作成
④ ログオン用 Action オブジェクトの作成
⑤ 計算入力ページの作成
⑥ 入力値 Form オブジェクトの作成
⑦ 計算処理 Action オブジェクトの作成

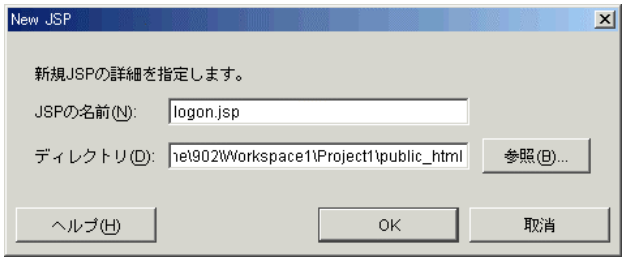
ログオンページの作成

まず JSP ファイルを作成しましょう。この JSP ファイルが、サンプルプログラムのログオン画面になります。

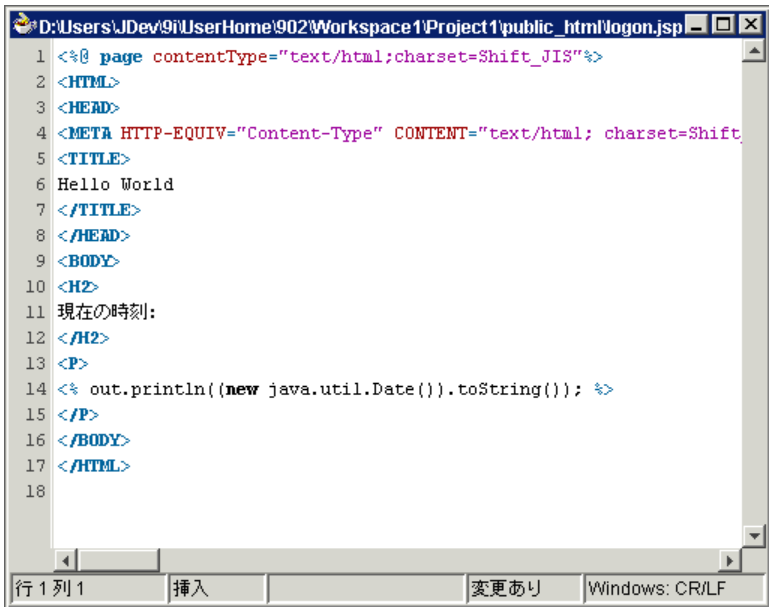
メニューの「ファイル」から「新規」を選択します。新規ウィンドウが表示されるので、カテゴリから「Web Objects」、項目から「JSP」を選択して OK ボタンをクリックします。



「New JSP」ウィンドウが表示されるので、「JSP の名前」を入力します。ここでは、「logon.jsp」と入力します。入力したら OK ボタンをクリックしてください。



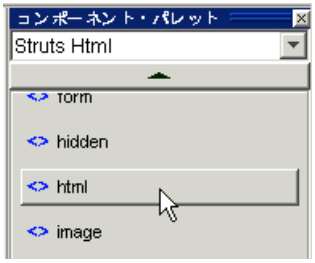
これで、logon.jsp ファイルの雛型が作成されました。



エディタで、logon.jsp ファイルを編集します。雛型として作成された内容のうち、<BODY>タグ内部を全て削除して、JSP ソースの編集を開始します。ここでは、最終的には、23 ページのようなソースを作成します。このソースをエディタ上で淡々とコーディングしてもよいですが、せっかくなので、JDeveloper のコーディング支援機能を利用してみましょう。次の手順を行ってください。

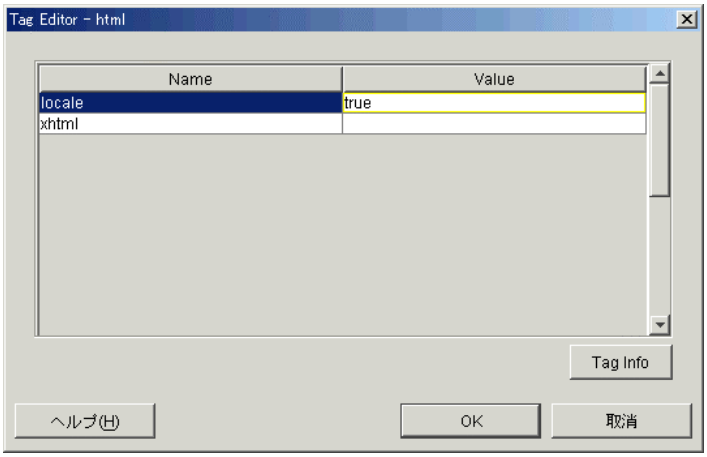
1. 2 行目の最初（<HTML>の前）にカーソルを配置します。

2. コンポーネント・パレットのリストで「Struts Html」を選択し、その中の「html」コンポーネント・ボタンを押します。



※ コンポーネント・パレット上の「Struts Html」は、11 ページ「Struts タグ・ライブラリの登録」での作業を行わないと表示されません。「Struts タグ・ライブラリの登録」の作業をしない場合は、すべて手動で、23 ページのようにコードを入力する必要があります。

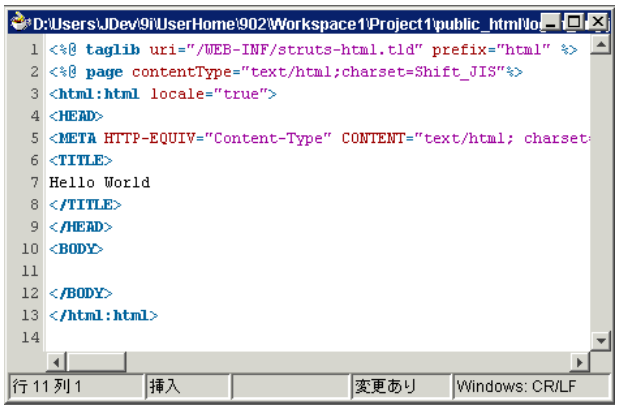
3. html タグに関する属性入力用のダイアログが表示されます。ここでは「locale」に「true」を設定し、OK ボタンをクリックしてください。



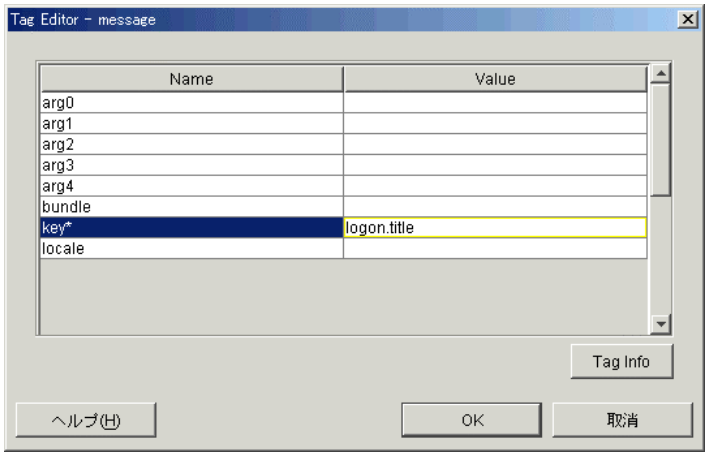
4. これにより、以下のようなコードになります。このように、taglib の指定タグとともに適切なタグの挿入を行うことができます。

```
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ page contentType="text/html;charset=shift_JIS" %>
<html:html locale="true"></html:html>
<HTML>
   :
```

5. この<html:html>タグが、<HTML>の代わりの役割をします。したがって、<HTML>は不要ですので削除できます。同様に、</HTML>の代わりが、</html:html>になりますので、</html:html>を適切な場所に配置してください。



6. <TITLE>タグ内の一行（”Hello World”とかかれている行）すべてを選択した状態で、コンポーネント・パレットのリストで「Struts Bean」を選択し、その中の「message」コンポーネント・ボタンを押します。



ダイアログ上で、「key」に「logon.title」と入力し、OK ボタンをクリックします。この結果、次のようなソースになります。

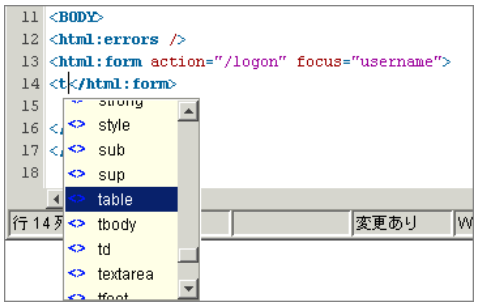
```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ page contentType="text/html;charset=shift_JIS" %>
<html:html locale="true">
<HEAD>
<META HTTP-EQUIV="Content-Type"
      CONTENT="text/html; charset=Shift_JIS">
<TITLE>
<bean:message key="logon.title"/>
</TITLE>
</HEAD>
<BODY>
   :
```

7. 同様の操作によって、<BODY>タグの次の行に、以下の行を挿入します。

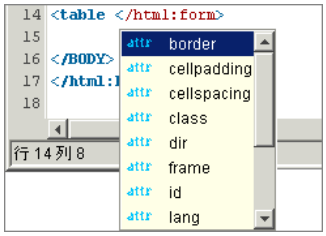
```
<html:errors/>
<html:form action="/logon" focus="username"></html:form>
```

これらは、ともに、コンポーネント・パレットの「Struts Html」のカテゴリ内のタグです。

8. </html:form> の前で改行し、”<” を入力します。入力候補のタグがリストで表示されますので「table」を選択します。



9. 引き続き、スペースを入力します。今度は、属性の入力候補がリストされます。ここでは「border」を選択し、続けて「”0”」を入力します。



10. “>” を入力してタグを閉じると、同時に終了タグである「</table>」も入力されます。

※ 終了タグが自動挿入されない場合は、「ツール→設定」から「エディタ-HTML/JSP」を選択して、「終了タグの自動補完」がチェックされていることを確認してください。

上記手順によって、以下のようなソースになります（見やすさのため、大文字/小文字の統一や改行位置の変更を行っています）。

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ page contentType="text/html;charset=shift_JIS" %>
<html:html locale="true">
  <head>
  <meta HTTP-EQUIV="Content-Type"
        CONTENT="text/html; charset=Shift_JIS">
  <title><bean:message key="logon.title"/></title>
  </head>

<body>
<html:errors/>
<html:form action="/logon" focus="username">
<table border="0"></table></html:form>

</body>
</html:html>
```

これらの手順を繰り返して、最終的には以下のようなソースにしてください。

logon.jsp

```
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ page contentType="text/html;charset=shift_JIS" %>
<html:html locale="true">
  <head>
  <meta HTTP-EQUIV="Content-Type"
        CONTENT="text/html; charset=Shift_JIS">
  <title><bean:message key="logon.title"/></title>
  </head>

<body>
<html:errors/>

<html:form action="/logon" focus="username">
<table border="0">

  <tr>
    <th align="right">
      <bean:message key="prompt.username"/>
    </th>
    <td align="left">
      <html:text property="username" size="16" maxlength="16"/>
       <html:submit property="submit" value="OK"/>
    </td>
  </tr>

</table>
</html:form>
</body>

</html:html>
```

<html:～>や<bean:～>などで始まるのが Struts タグ・ライブラリです。例えば、<html:text propert= "username" ～>はテキストボックスを表示するタグです。ここの property で定義されている username と後述の LogonForm にあるプロパティの username の値が、やり取りされます。

<bean:message>タグについては、37 ページ「メッセージ・リソースの作成」を参照してください。その他のタグ・ライブラリの詳細については、以下のサイトを参照してください。

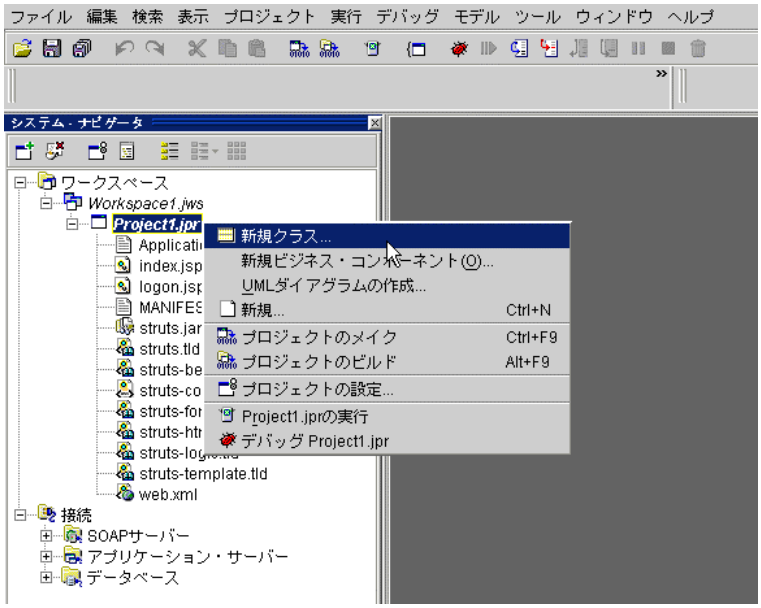
http://jakarta.apache.org/struts/doc-1.0.2/index.html

なお、「HTML デザインは一般の HTML エディタを用いて行いたい」というニーズのために、JDeveloper から外部エディタを起動するアドイン・プログラムが OTN-J（http://otn.oracle.co.jp/）で公開されています。それを組み合わせることで、より生産性の高い JSP ページ開発が可能になるでしょう。

ユーザー情報オブジェクト（User.java）の作成

次に、ログオンしたユーザー情報を管理するオブジェクト・クラスを作成します。このオブジェクトは、各画面や Action がセッションを通して共有するユーザー情報オブジェクトです。このサンプルでは、ログオン画面に入力されたユーザー名のみ保持します。

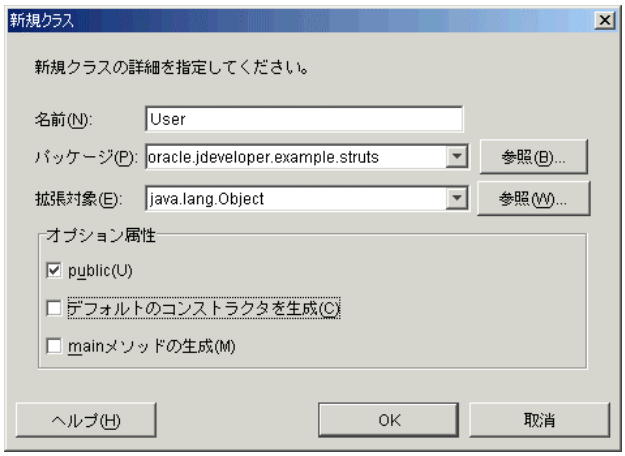
Project1.jpr を右クリックして「新規クラス」を選択します。



「新規クラス」ウィンドウが表示されます。

名前: User
オプション属性: 「public」のみチェック

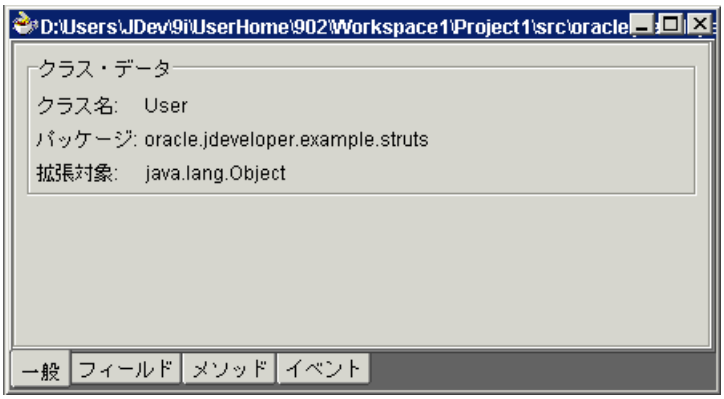
として OK ボタンをクリックします。



これによって作成されたファイルを編集して、ユーザー情報管理用のオブジェクト・クラスとします。最終的に目指すソースは、26 ページのようなものですが、

ここで、JDeveloper のクラス・エディタを利用した、より効率的な開発方法を紹介します。

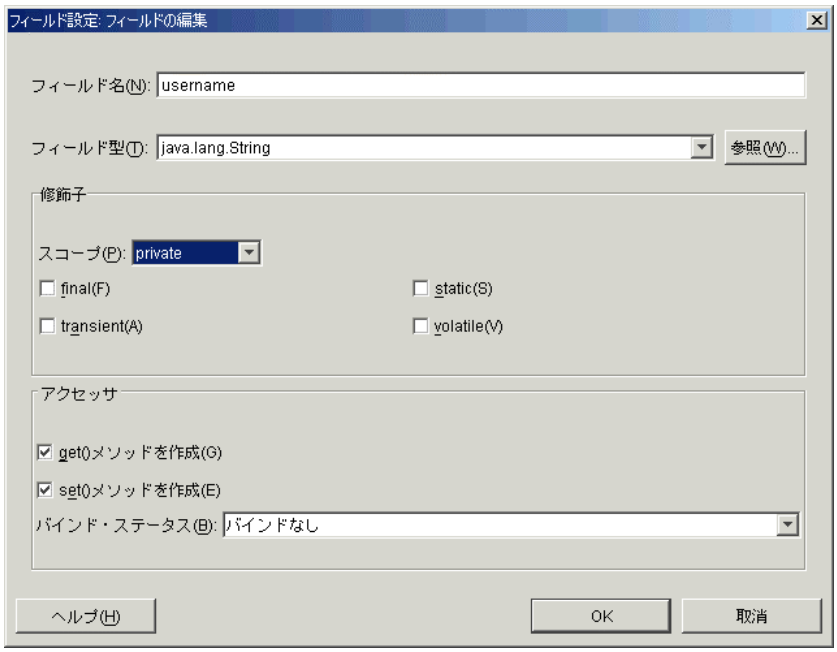
1. 開かれたソース・ファイル上で右クリックメニューから「クラス・エディタ」を選択します。クラス・エディタが表示されます。



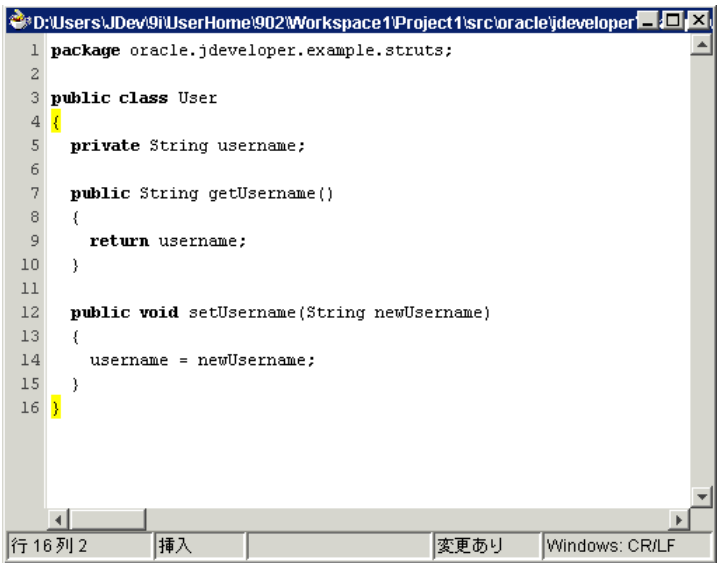
2. 「フィールド」タブの「追加」ボタンをクリックします。「フィールド設定」ウィンドウが表示されます。

   フィールド名: username
   スコープ: private

   として、OK ボタンをクリックします。



これにより、ソース上は以下のようなコードが記述されています。



このように、一連のクラス・メンバとそのアクセッサ・メソッド（get および set）を作成する際には、クラス・エディタが便利です。

User.java に必要なコードはこれだけです。後は、ドキュメント目的でのコメント部分を必要に応じて追加してください。最終的には、次のようなソースになります。

User.java

```
package oracle.jdeveloper.example.struts;

/**
 * ログインユーザー
情報
 *
 * @author
 * @version $Revision: 1.0 $ $Date: 2002/xx/xx $
 */
public class User {

    /**
     * ユーザー名
     */
    private String username;

    /**
     * ユーザー名を取得する
     */
    public String getUsername() {
        return (username);
    }
```

```
    /**
     * ユーザー名を設定する
     *
     * @param newUsername 新しいユーザー名
     */
    public void setUsername(String newUsername) {
       username = newUsername;
    }
}
```

ログイン処理の作成

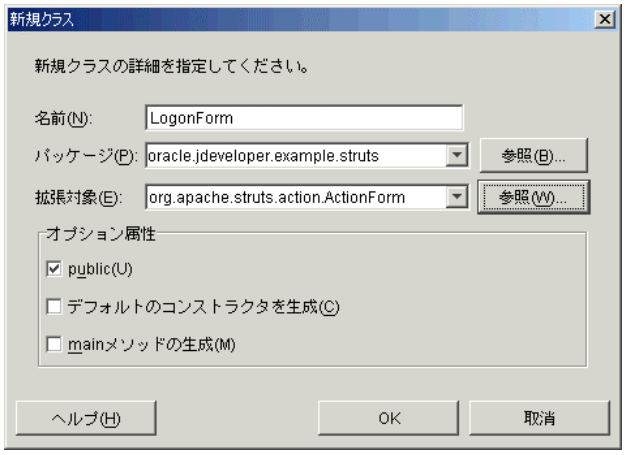
次に、ログオン画面を処理する Form と Action を作成します。Project1.jpr を右クリックして「新規クラス」を選択します。

「新規クラス」ウィンドウが表示されます。

名前:	LogonForm
拡張対象:	org.apache.struts.action.ActionForm
オプション属性:	「public」のみチェック

として OK ボタンをクリックします。

※　「拡張対象」は、その右側の「参照」ボタンから対象のクラスを選択してください。



これで、以下のような LogonForm.java ファイルの雛型が作成されます。

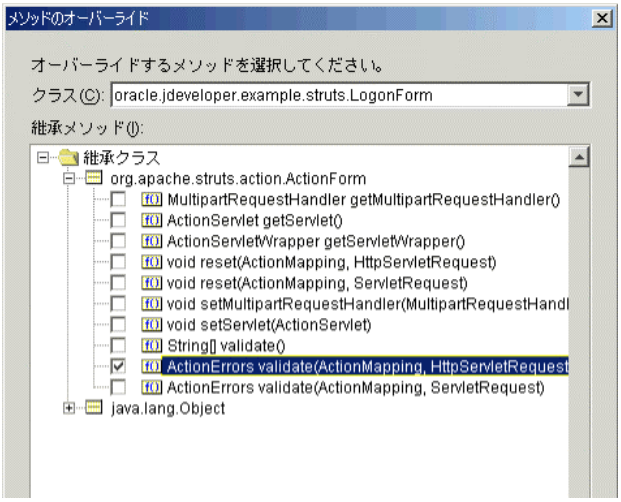
```
package oracle.jdeveloper.example.struts;
import org.apache.struts.action.ActionForm;

public class LogonForm extends ActionForm
{
}
```

この後、エディタで、LogonForm.java ファイルを編集して、29 ページのようなコードにします。まずは、先ほどの User.java のときと同じ手法（クラス・エディタの利用）で、次のように、username およびそのアクセッサ・メソッドを追加してください。

```
package oracle.jdeveloper.example.struts;

public class LogonForm extends ActionForm
{
  private String username;

  public String getUsername()
  {
    return username;
  }

  public void setUsername(String newUsername)
  {
    username = newUsername;
  }
}
```

ここで、もうひとつ便利な機能をひとつ紹介します。

メニューから「ツール→メソッドのオーバーライド」を選択します。「メソッドのオーバーライド」ダイアログで「org.apache.struts.action.ActionForm」のツリーを開き、「ActionErrors validate(ActionMapping, HttpServletRequest)」 メソッドをチェックします。



OK ボタンをクリックすると、validate メソッドの雛型が追加記述されます。

ここまでで、次のようなコードになっているはずです。

```
package oracle.jdeveloper.example.struts;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionMapping;
import javax.servlet.http.HttpServletRequest;

public class LogonForm extends ActionForm
{
  private String username;

  public String getUsername()
```

```
  {
    return username;
  }

  public void setUsername(String newUsername)
  {
    username = newUsername;
  }

  public ActionErrors validate(ActionMapping p0, HttpServletRequest p1)
  {
    // TODO: （略）
    return super.validate(p0, p1);
  }
}
```

ここからはエディタ上での編集作業になります。以下の変更が必要です。

- username の初期値を null に設定
- validate メソッドの中身を記述

※ このメソッド内で org.apache.struts.action.ActionErrors クラスを宣言していますので、それをインポートする文も追加する必要があります。

- （必要に応じて）ドキュメント目的のコメント部分を追加
- （必要に応じて）validate メソッドの引数名をわかりやすく変更

最終的には次のようにします。

LogonForm.java

```
package oracle.jdeveloper.example.struts;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionErrors;

/**
 * ログオンフォーム:
 * <ul>
 * <li><b>username</b> - 入力されたユーザー名
 * </ul>
 *
 * @author
 * @version $Revision: 1.0 $ $Date: 2002/xx/xx $
 */
public class LogonForm extends ActionForm {

    /**
     * ユーザー名
     */
    private String username = null;

    /**
```

```
     * ユーザー名の取得
     */
    public String getUsername() {
       return (username);
    }

    /**
     * ユーザー名の設定
     *
     * @param newUsername 新しいユーザー名
     */
    public void setUsername(String newUsername) {
       username = newUsername;
    }

    /**
     * エラーチェック
     *
     * @param mapping インスタンス・マッピング
     * @param request サーブレット・リクエスト
     */
    public ActionErrors validate(ActionMapping mapping,
                                 HttpServletRequest request) {

       ActionErrors errors = new ActionErrors();
       if ((username == null) || (username.length() < 1))
          errors.add("username",
          new ActionError("error.username.required"));

       return errors;
    }
}
```

private String username = null; で定義されている username と logon.jsp テキストボックスの property である username が結び付けられています。この値が違うと JSP の表示でエラーになります。

同様にして、LogonAction.java を作成します。手順は以下のとおりです。

- LogonAction は、org.apache.struts.action.Action を拡張してクラス定義します。
- ActionForward perform(ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse) メソッドをオーバーライドします。
- Perform メソッド内のコードを追加します。

最終的に以下のようなコードにします。

LogonAction.java

```
package oracle.jdeveloper.example.struts;

import java.io.IOException;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

/**
 * ログオン処理
 *
 * @author
 * @version $Revision: 1.0 $ $Date: 2002/xx/xx $
 */
public final class LogonAction extends Action {

   public ActionForward perform(ActionMapping mapping,
             ActionForm form,
             HttpServletRequest request,
             HttpServletResponse response)
   throws IOException, ServletException {

     // 初期処理
     User user = new User();
     user.setUsername(((LogonForm) form).getUsername());

     // ユーザー情報をセッションに設定
     HttpSession session = request.getSession();
     session.setAttribute("USER", user);

     // 成功画面に遷移
     return (mapping.findForward("success"));
   }
}
```

perform メソッドが、実行時に ActionServlet クラスから呼ばれます。

※ perform メソッドは、現在開発段階にある Struts1.1 以降では excute メソッドに置き換わる予定です。

((LogonForm) form).getUsername() によって、logon.jsp で submit（「OK」ボタンをクリック）したときの「お名前」の文字列を取得できます（Struts が form に対して自動で取得しています)。

最後の return(mapping.findForward("success")) は、39 ページ「struts-config.xml の編集」の struts-config.xml で定義されている

<forward name="success" path="/calc.do?action=Init"/>

によって、/calc.do?action=Init に遷移します。

計算画面と処理の作成

では次に、計算画面を作成しましょう。作成方法はログオン画面と同様です。以下に、各ファイル作成の主要ポイントだけをリストしますので、それを参考にコードを作成してみてください。

- calc.jsp
  - 新規 JSP ページを作成後、コンポーネント・パレットやタグの記述支援機能を利用して内容を記述します。
  - <logic:…>のタグは、コンポーネント・パレットの「Struts Logic」に対応します。
  - ソース内の Java ブロックで User オブジェクトを利用しているため、それをインポートする必要があります。
- CalculationForm.java
  - org.apache.struts.action.ActionForm の拡張クラスとして作成します。
  - 特にメソッドのオーバーライドなどを行いません。クラス・エディタを利用するだけで基本ソースは完成します。
- CaluculationAction.java
  - org.apache.struts.action.Action の拡張クラスとして作成します。
  - ActionForward perform(ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse) メソッドをオーバーライドして、必要なコードを記述します。
  - 記述するコード内で利用しているため、javax.servlet.http.HttpSession、org.apache.struts.action.ActionError、org.apache.struts.action.ActionErrors をインポートする必要があります。

calc.jsp

```
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ page contentType="text/html;charset=Shift_JIS"%>
<%@ page import="oracle.jdeveloper.example.struts.User" %>

<html:html locale="true">
<%  User user = (User)session.getAttribute("USER"); %>

<head>
  <meta HTTP-EQUIV="Content-Type"
        CONTENT="text/html; charset=Shift_JIS">
  <title><bean:message key="calc.title" /></title>
</head>

<body>
<html:errors />

<table border="0" width="100%">
  <logic:equal name="calcForm" property="action"
               scope="request" value="Init">
    <tr>
      <td><%=user.getUsername()%>さん、いらっしゃいませ。<br>
          計算したい値をいれて「計算」ボタンをクリックしてください。</td>
    </tr>
```

```
    <tr>
      <td align="center">
        <html:form action="/calc">
          <html:text maxlength="5" property="input1" size="5" /> +
          <html:text maxlength="5" property="input2" size="5" /> =
          <html:submit property="submit" value="計算" />
        </html:form>
      </td>
    </tr>
  </logic:equal>

  <logic:notEqual name="calcForm" property="action"
                  scope="request" value="Init">
    <tr>
      <td align="left">
        <%=user.getUsername()%>さん
        <bean:write name="calcForm" property="input1" /> たす
        <bean:write name="calcForm" property="input2" /> は
        <bean:write name="calcForm" property="result" /> ですよ。
      </td>
    </tr>
    <tr>
      <td><ul>
        <li><html:link page="/calc.do?action=Init">
            <bean:message key="calc.recalc" /></html:link></li>
        <li><html:link page="/logon.jsp">
            <bean:message key="calc.logon" /></html:link></li>
      </ul></td>
    </tr>
  </logic:notEqual>

</table>
</body>
</html:html>
```

<logic:equal>タグは、指定された name="calcForm"で識別される Form の action と value="Init"の値が等しいときに出力されます。name="calcForm"は、39 ページ「struts-config.xml の編集」の struts-config.xml ファイルで定義されています。

また、<logic:notEqual>タグは、equal の逆で action と value=”Init”の値が等しくないときに出力されます。

CalculationForm.java

```
package oracle.jdeveloper.example.struts;

import org.apache.struts.action.ActionForm;

/**
 * ログオンフォーム:
```

```
 * <ul>
 * <li><b>input1</b> - 入力された値１
 * <li><b>input2</b> - 入力された値２
 * <li><b>result</b> - 計算結果
 * </ul>
 *
 * @author
 * @version $Revision: 1.0 $ $Date: 2002/xx/xx $
 */
public class CalculationForm extends ActionForm {

    /**
     * アクション
     */
    private String action = "Init";

    /**
     * 入力された値１
     */
    private String input1 = null;

    /**
     * 入力された値２
     */
    private String input2 = null;


    /**
     * 計算結果
     */
    private String result = null;

    /**
     * アクションの取得
     */
    public String getAction() {
        return (action);
    }

    /**
     * アクションの設定
     *
     * @param newAction 新しいアクション
     */
    public void setAction(String newAction) {
        action = newAction;
    }

    /**
     * 入力された値１の取得
     */
    public String getInput1() {
        return (input1);
    }
```

```
    /**
     * 入力された値１の設定
     *
     * @param newInput1 新しい入力された値１
     */
    public void setInput1(String newInput1) {
        input1 = newInput1;
    }

    /**
     * 入力された値２の取得
     */
    public String getInput2() {
        return (input2);
    }

    /**
     * 入力された値２の設定
     *
     * @param newInput2 新しい入力された値２
     */
    public void setInput2(String newInput2) {
        input2 = newInput2;
    }

    /**
     * 計算結果の取得
     */
    public String getResult() {
        return (result);
    }

    /**
     * 計算結果の設定
     *
     * @param newResult 新しい計算結果
     */
    public void setResult(String newResult) {
        result = newResult;
    }
}
```

CalculationAction.java

```
package oracle.jdeveloper.example.struts;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
```

```
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

/**
 * 計算処理
 *
 * @author
 * @version $Revision: 1.0 $ $Date: 2002/xx/xx $
 */
public final class CalculationAction extends Action {

    public ActionForward perform(
                ActionMapping mapping,
                ActionForm form,
                HttpServletRequest request,
                HttpServletResponse response )
    throws IOException, ServletException {

    ActionErrors errors = new ActionErrors();

    // ユーザー情報をセッションから取得
    User user = new User();
    HttpSession session = request.getSession();
    user = (User)session.getAttribute("USER");

    if (user == null) {
        errors.add(ActionErrors.GLOBAL_ERROR,
                  new ActionError("error.session.timeout"));
        saveErrors(request, errors);
        return (servlet.findForward("logon"));
    }

    // 初期処理
    String action = request.getParameter("action");
    if (action == null) {
        action = "";
    }
    ((CalculationForm) form).setAction(action);

    // 初期表示は何もしない
    if ("Init".equals(action)) {
        return (new ActionForward(mapping.getInput()));
    }

    // エラーチェック
    String input1 = ((CalculationForm) form).getInput1();
    String input2 = ((CalculationForm) form).getInput2();
    int nResult = 0;
    try {
        nResult = Integer.parseInt(input1) + Integer.parseInt(input2);
    } catch (NumberFormatException e) {
        errors.add(ActionErrors.GLOBAL_ERROR,
                  new ActionError("error.calc.numeric"));
        ((CalculationForm) form).setAction("Init");
    }
```

```
    // エラーがあったら入力画面に戻る
    if (!errors.empty()) {
        saveErrors(request, errors);
        return (new ActionForward(mapping.getInput()));
    }

    // 計算結果を設定
    ((CalculationForm) form).setResult(Integer.toString(nResult));

    // 成功画面に遷移
    return (new ActionForward(mapping.getInput()));

    }
}
```

4.3 サンプルプログラムの作成 2

ここでは、Struts サンプルプログラムの画面遷移の振る舞いを定義している struts-config.xml およびメッセージやプロパティを定義している ApplicationResources.properties ファイルを編集します。

メッセージ・リソースの作成

メッセージやプロパティなどを定義してあるメッセージ・リソース・ファイルの編集をします。Struts では、Struts タグ・ライブラリを使用して、このファイルからメッセージやプロパティなどを自動的に取得します。

メッセージ・リソース・ファイルは、Struts のデフォルトの設定では、ApplicationResources.properties という名前です。このファイルは既にプロジェクト内に初期ファイルとして用意されています。これを開き、次のように入力してください。

※ ApplicationResources.properties にはじめから記述されている内容は、そのまま残しても、消してしまってもどちらでもかまいません。

ApplicationResources.properties

```
logon.title=ログオン画面
prompt.username=お名前:
calc.title=計算
calc.recalc=Back
calc.logon=Logout
errors.header=<h3><font color="red">エラー</font></h3><ul>
errors.footer=</ul><hr>
error.username.required=<li>お名前を入れてください。</li>
error.calc.numeric=計算する値には数値を入力してください。
error.session.timeout=セッションがタイムアウトになりました。<br>再度ログオンしてください。
```

たとえば、「ログオンページの作成」で作成した logon.jsp を例に取ると、

```
<title><bean:message key="logon.title"/></title>
```

のようにして、<bean:message> タグを使用しています。ここの key の値とリソース・ファイルの値が結び付けられていて、実行時にリソース・ファイルから適切な値を取得します。なお、実行時に、対応する key が取得できないとエラーとなってしまうので注意してください。

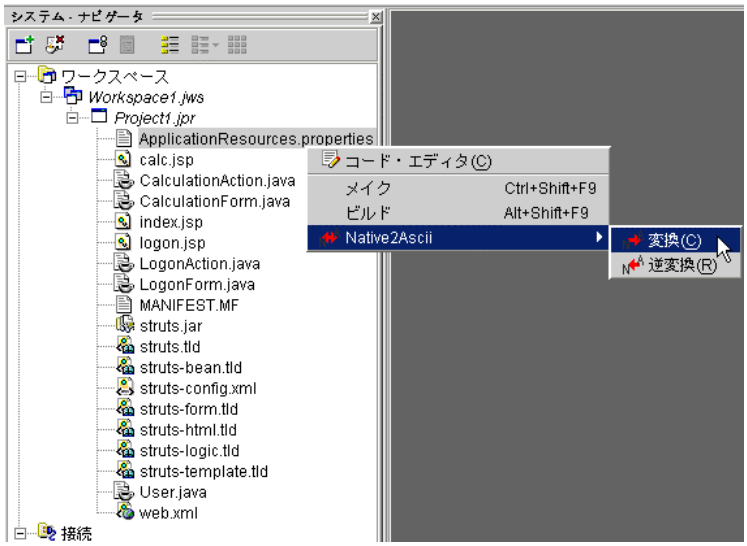
properties ファイルを Unicode エスケープ変換する

properties ファイル内の文字をこのままにしておくと、このファイルから取得された文字部分が文字化けを起こします。これを避けるためには、ファイル内の文字を Unicode エスケープ変換しておく必要があります。

通常、この作業は、Java2 SDK に標準で付属している「Navite2Ascii」ユーティリティを使用します。このユーティリティはコマンドラインから使用します。

```
<Java2 SDK パス>¥bin¥native2ascii <変換元のファイル名> <変換後のファイル名>
```

なお、この Native2Ascii を JDeveloper から実行可能にするアドイン・プログラムが OTN-J（http://otn.oracle.co.jp/）で公開されています。それを利用すれば、コマンドラインを使用せずに変換作業を行うことができます。



コード変換後のファイル内容は次のようになります。

ApplicationResources.properties（変換後）

```
logon.title=¥u30ed¥u30b0¥u30aa¥u30f3¥u753b¥u9762
prompt.username=¥u304a¥u540d¥u524d:
calc.title=¥u8a08¥u7b97
calc.recalc=Back
calc.logon=Logout
errors.header=<h3><font
color="red">¥u30a8¥u30e9¥u30fc</font></h3><ul>
errors.footer=</ul><hr>
error.username.required=<li>¥u304a¥u540d¥u524d¥u3092¥u5165¥u308c¥u30
66¥u304f¥u3060¥u3055¥u3044¥u3002</li>
error.calc.numeric=¥u8a08¥u7b97¥u3059¥u308b¥u5024¥u306b¥u306f¥u6570¥
u5024¥u3092¥u5165¥u529b¥u3057¥u3066¥u304f¥u3060¥u3055¥u3044¥u3002
error.session.timeout=¥u30bb¥u30c3¥u30b7¥u30e7¥u30f3¥u304c¥u30bf¥u30
a4¥u30e0¥u30a2¥u30a6¥u30c8¥u306b¥u306a¥u308a¥u307e¥u3057¥u305f¥u3002
<br>¥u518d¥u5ea6¥u30ed¥u30b0¥u30aa¥u30f3¥u3057¥u3066¥u304f¥u3060¥u30
55¥u3044¥u3002
```

struts-config.xml の編集

続いて、struts-config.xml ファイルを編集します。struts-config.xml をエディタで開いて、次のように修正してください。

struts-config.xml

```
<?xml version="1.0" encoding="shift_JIS" ?>

<!DOCTYPE struts-config PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
   "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">

<struts-config>

  <!-- ========== Form Bean Definitions ======================== -->
  <form-beans>

    <!-- Logon form bean -->
    <form-bean name="logonForm"
          type="oracle.jdeveloper.example.struts.LogonForm"/>

    <!-- Calculation form bean -->
    <form-bean name="calcForm"
          type="oracle.jdeveloper.example.struts.CalculationForm"/>

  </form-beans>


  <!-- ========== Global Forward Definitions ================== -->
  <global-forwards>
    <forward   name="logoff"  path="/logoff.do"/>
```

```
    <forward   name="logon"   path="/logon.jsp"/>
  </global-forwards>


  <!-- ========== Action Mapping Definitions ================== -->
  <action-mappings>

    <!-- Process a user logoff -->
    <action   path="/logoff"
         type="oracle.jdeveloper.example.struts.LogoffAction">
      <forward name="success" path="/logoff.jsp"/>
    </action>

    <!-- Process a user logon -->
    <action   path="/logon"
         type="oracle.jdeveloper.example.struts.LogonAction"
         name="logonForm"
         scope="request"
         input="/logon.jsp">
      <forward name="success" path="/calc.do?action=Init"/>
    </action>

    <!-- Process a calculation -->
    <action   path="/calc"
         type="oracle.jdeveloper.example.struts.CalculationAction"
         name="calcForm"
         scope="request"
         input="/calc.jsp">
      <forward name="success" path="/calc.do?action=Result"/>
    </action>

  </action-mappings>

</struts-config>
```

（参考情報）struts-config.xml の編集を行うための専用エディタ・アドインも存在します。

http://www.jamesholmes.com/struts/console/

これはオラクルで出しているものではありませんが、これを使うと XML のコーディング・ミスがなくて便利です。必要に応じてご利用ください。

4.4 開発環境での実行

システム・ナビゲータ上で「logon.jsp」を選択して、メニューから「実行→logon.jspの実行」を選択します。JDeveloper 内のローカル環境で内蔵のアプリケーション・サーバーが起動して、テスト実行することができます。

実行イメージは、後に紹介する「5.4 サンプルプログラムの実行」のとおりです。ただし、今のままでは、最初のログイン時に日本語のユーザー名を入力した場合、2 ページ目以降でそのユーザー名が文字化けします。次節では、この日本語入力に対応するようにコードを追加します。

4.5 日本語入力対応にするために

前節でのテスト実行でもわかるように、今のままではブラウザから入力された文字が日本語の場合に、正しく文字コードが判別されません。これは Struts の仕様です。これを日本語入力対応にするためには、HTTP リクエストを受けるサーブレットに対して、送信された文字のキャラクタ・セットを指示する必要があります。これを行うために、Struts の ActionServlet を拡張します。

拡張クラスの作成

Project1.jpr を右クリックして「新規クラス」を選択します。

「新規クラス」ウィンドウが表示されます。

名前: ExtendedActionServlet
拡張対象: org.apache.struts.action.ActionServlet
オプション属性: 「public」のみチェック

として OK ボタンをクリックします。さらに、メニューから「ツール→メソッドのオーバーライド」を選択して、次のメソッドをオーバーライドします。

- `org.apache.struts.action.ActionServlet の`
  `void doPost(HttpServletRequest , HttpServletResponse)`

このメソッド内に、次の一行を追加します。

```
p0.setCharacterEncoding("shift_JIS");
```

※ doPost の第一引数名が p0 の場合のコード例です。

上記で指定する文字コードは、前述の JSP ページ（logon.jsp および calc.jsp）の <meta>タグ内で指定した文字コードと一致させてください。最終的には次のようなコードになります。

```
package oracle.jdeveloper.example.struts;

import org.apache.struts.action.ActionServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import javax.servlet.ServletException;

public class ExtendedActionServlet extends ActionServlet {
  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
  throws IOException, ServletException {
    request.setCharacterEncoding("shift_JIS");
    super.doPost(request, response);
  }
}
```

web.xml の編集

web.xml をコード・エディタで開きます。

<servlet-class>タグ部分として「org.apache.struts.action.ActionServlet」と設定されている個所がありますので、それを次のように書き換えてください。

```
<servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
```

↓

```
<servlet-class>
oracle.jdeveloper.example.struts.ExtendedActionServlet
</servlet-class>
```

これで正しく日本語処理が行われるようになります。

また、上記方法のほかに、Struts の ActionServlet にサーブレット・フィルタを適用して、そこでキャラクタ・セットの指定を行うという方法もあります。これについては、本書では説明しません。興味のある方は、他の Web サイトなどの技術資料を参照ください。

5. アプリケーション・サーバーへの配布（デプロイ）

この章では、サンプルプログラムのアプリケーション・サーバーへのデプロイ方法と実行方法について説明します。

5.1 OC4J の起動

まず、Oracle9i Application Server の J2EE コンテナである OC4J（Oracle9iAS Containers for J2EE）をインストールしましょう。

カレント・ディレクトリを<JDeveloper_HOME>¥j2ee¥home とし、以下のコマンドを実行します。

```
java -jar oc4j.jar -install
```

ここで管理用のパスワードの入力が求められます。ここで入力したパスワードは、OC4J を再起動/停止する場合など OC4J 付属の管理ツールを使用する際に必要です。

では、OC4J を起動します。以下のコマンドを実行してください。

```
java -jar oc4j.jar
```

ここで、次のメッセージが表示されれば OC4J の起動は成功です。

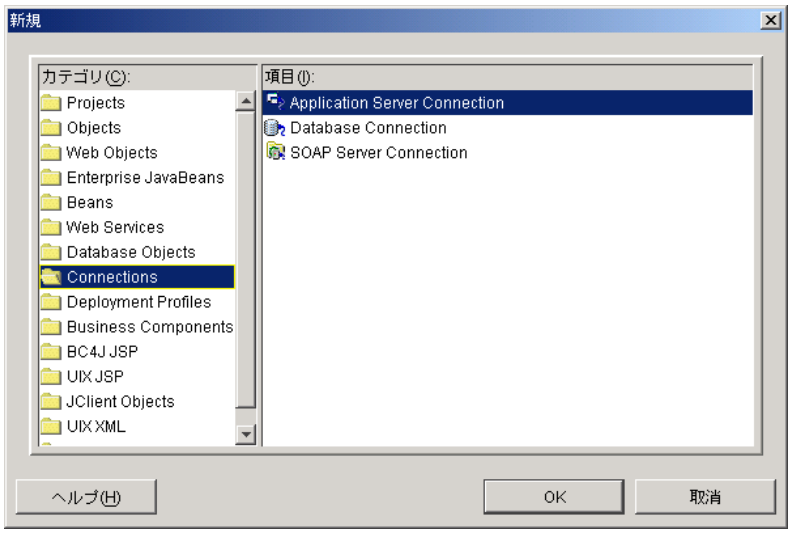
```
Oracle9iAS (9.0.2.0.0) Containers for J2EE initialized
```

5.2 Oracle9i JDeveloper から OC4J へ接続

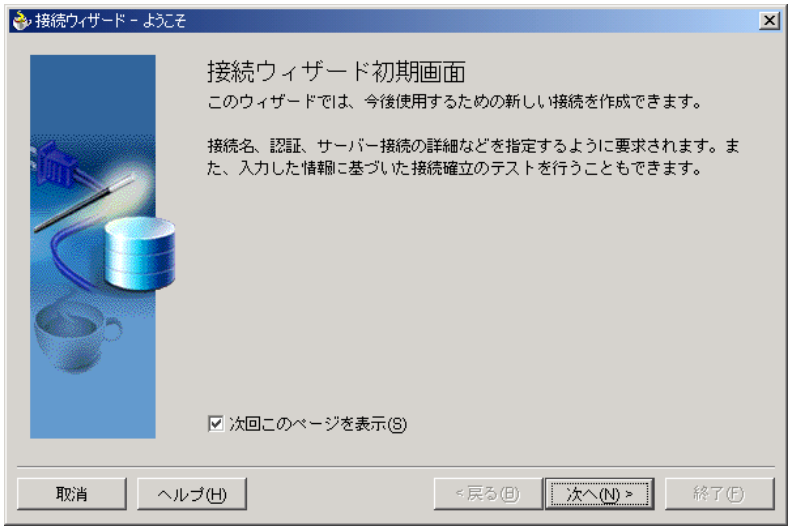
ここでは、JDeveloper から OC4J に接続するための設定をします。

メニューの「ファイル→新規」を選びます。

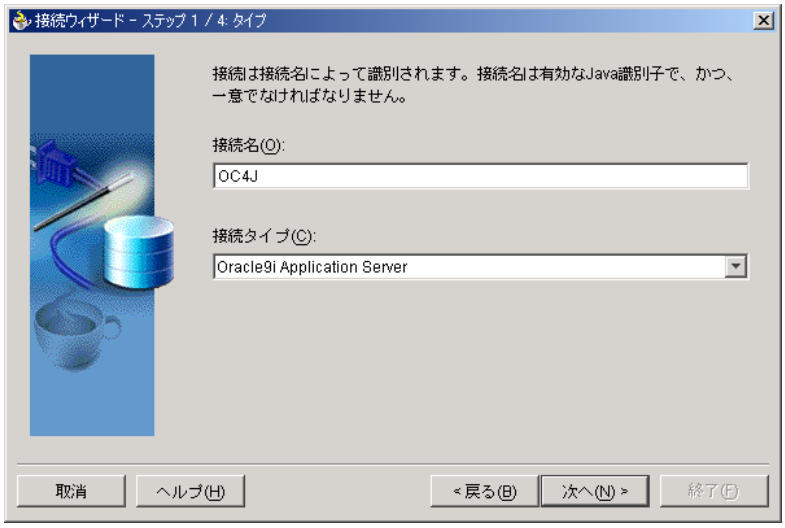
「新規」ウィンドウでカテゴリ「Connections」、項目「Application Server Connection」を選択して OK ボタンをクリックします。



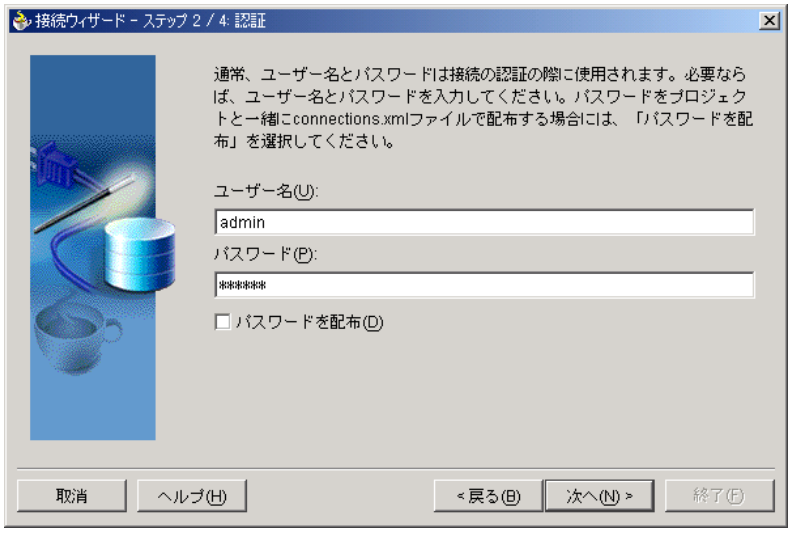
接続ウィザードが表示されます。「次へ」をクリックしてください。



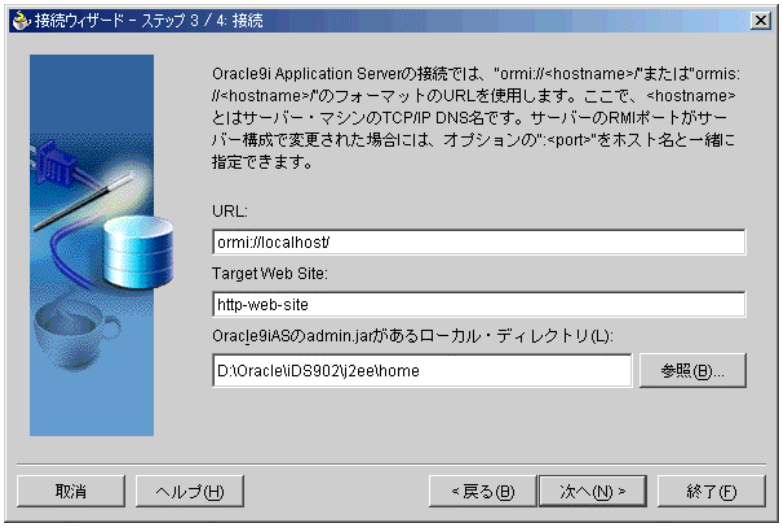
次に、「接続名」と「接続タイプ」を入力します。ここでは、「接続名」に例として「OC4J」、「接続タイプ」で「Oracle9i Application Server」を選択します。



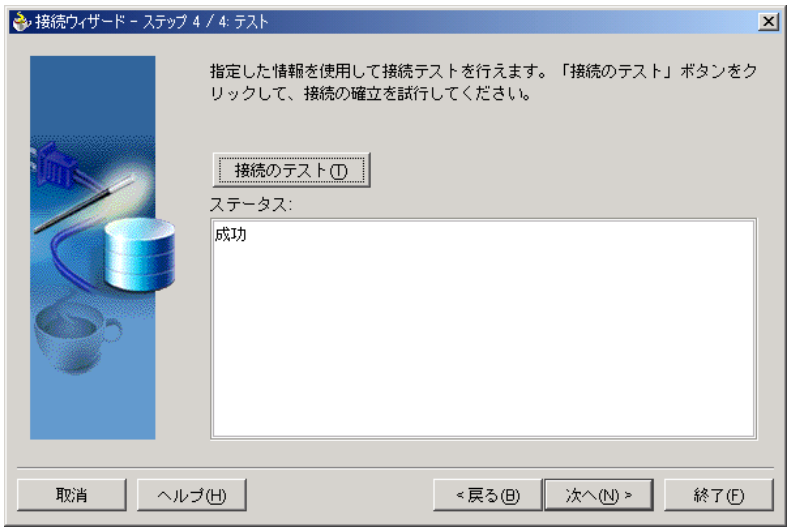
次に、OC4J の管理用のユーザー名とパスワードを入力します。ユーザー名はデフォルトの「admin」、パスワードは OC4J をインストールしたときの管理用のパスワードを入力して「次へ」をクリックします。



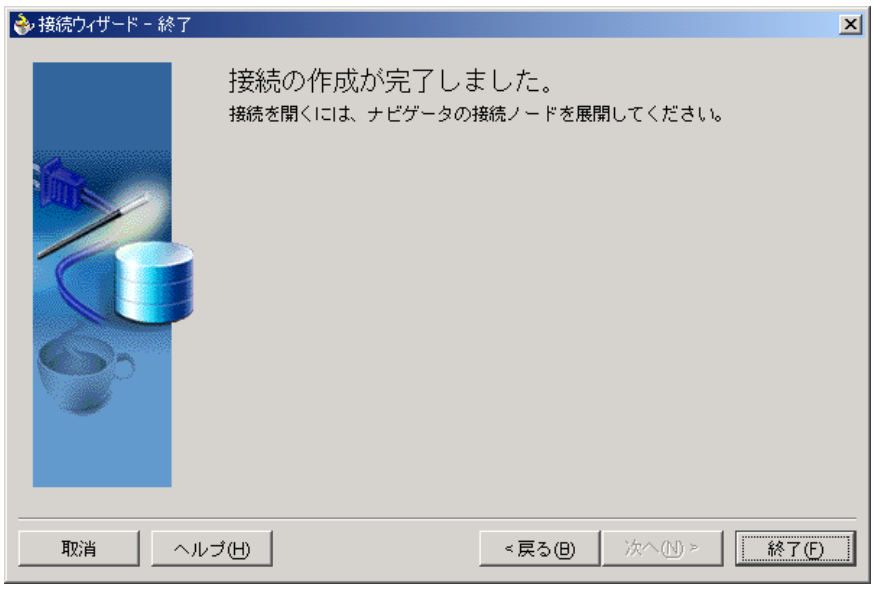
次に、サーバー構成情報を入力します。ここでは、デフォルトのまま「次へ」をクリックしましょう。



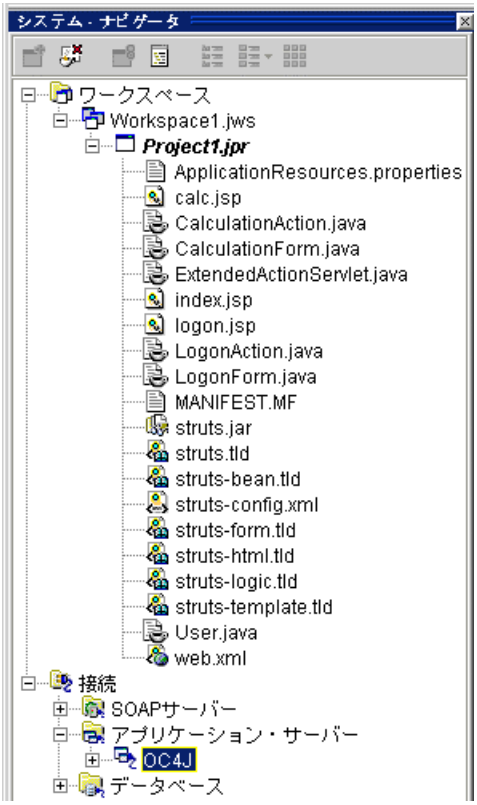
最後に、OC4J との接続テストを行います。「接続のテスト」ボタンをクリックしてください。下記画面のように「ステータス」に「成功」と表示されれば OC4J との接続が成功しました。「成功」したら「次へ」をクリックしましょう。



これで、OC4J への接続設定が完了しました。「終了」ボタンをクリックしてください。



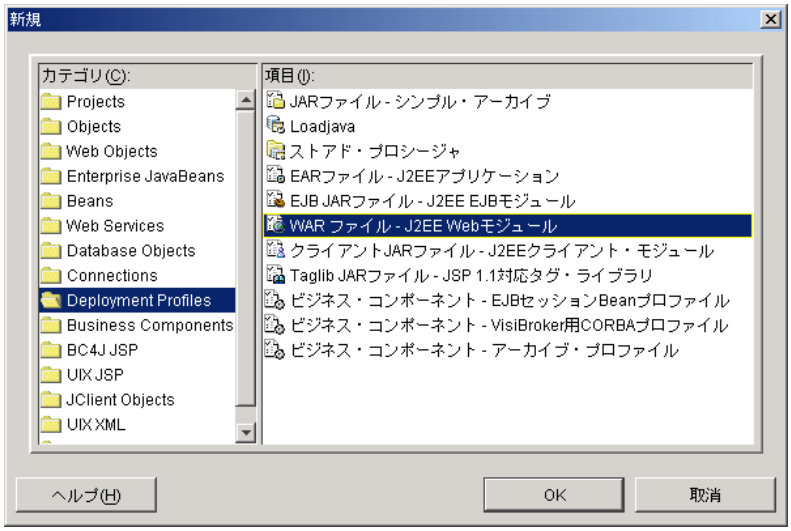
JDeveloper の「システム・ナビゲータ」の「接続」-「アプリケーション・サーバー」に「OC4J」が追加されていることを確認してください。



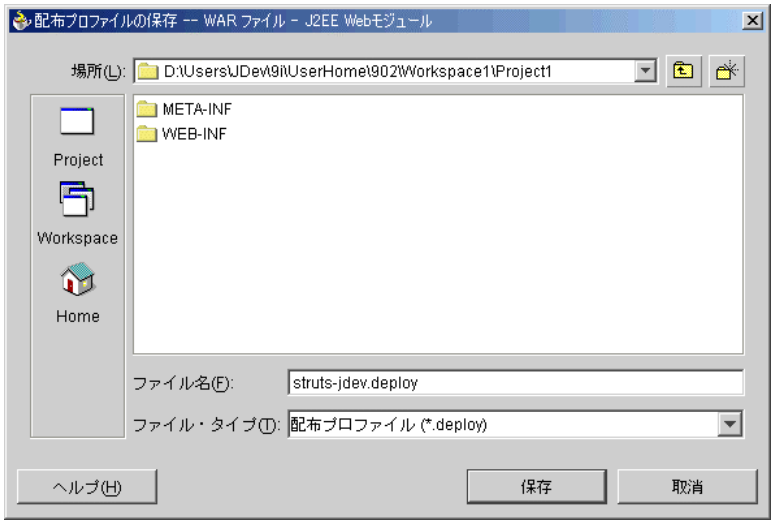
5.3 OC4J に配布（デプロイ）

配布プロファイルの作成

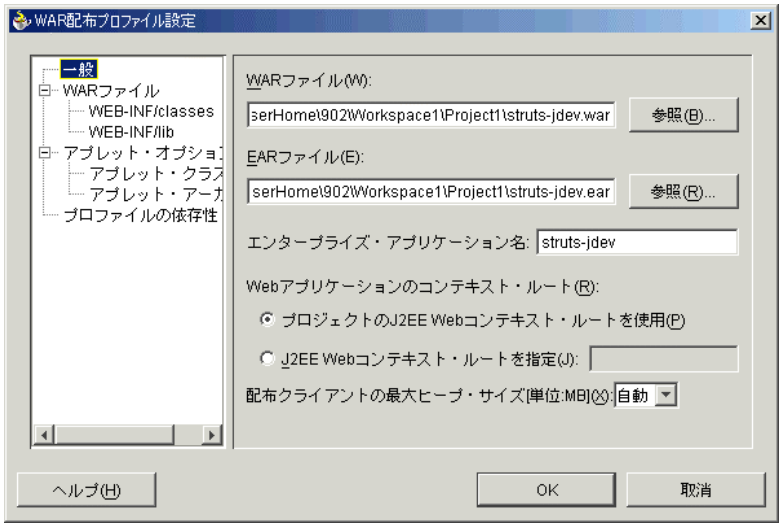
配布プロファイルを作成します。JDeveloper で、Project1.jpr を右クリックして「新規」を選択してください。「新規」ウィンドウのカテゴリで「Deployment Profiles」、項目で「WAR ファイル-J2EE Web モジュール」を選択して OK ボタンをクリックします。



「配布プロファイルの保存」ウィンドウが表示されるので、「ファイル名」を入力します。ここでは例として「struts-jdev.deploy」を入力して保存ボタンをクリックします。

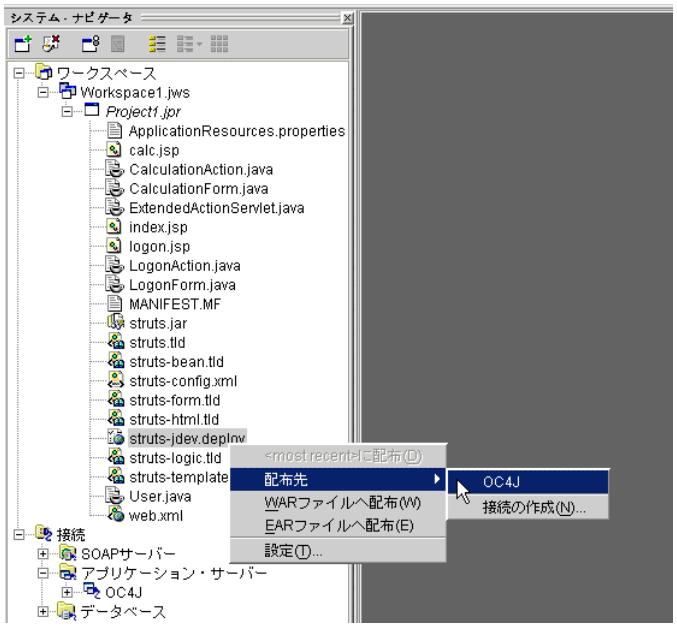


続けて「WAR 配布プロファイルの設定」ウィンドウが表示されますので、このまま OK ボタンをクリックします。



配布（デプロイ）

サンプルプログラムをデプロイしてみましょう。「システム・ナビゲータ」で struts-jdev.deploy ファイルを右クリックして「配布先」から「OC4J」を選択します。



次のようなメッセージが表示されればデプロイは成功です。

```
Oracle9iAS 管理ツールの終了ステータス(-deploy)： 0
D:¥Oracle¥iDS902¥jre¥bin¥javaw.exe -jar D:¥Oracle¥iDS902¥j2ee…
Oracle9iAS 管理ツールの終了ステータス(-bindWebApp)： 0
---- 配布が完了 ----    2002/XX/XX  xx:xx:xx
```

5.4 サンプルプログラムの実行

ここでは、実際にサンプルプログラムを実行してみます。

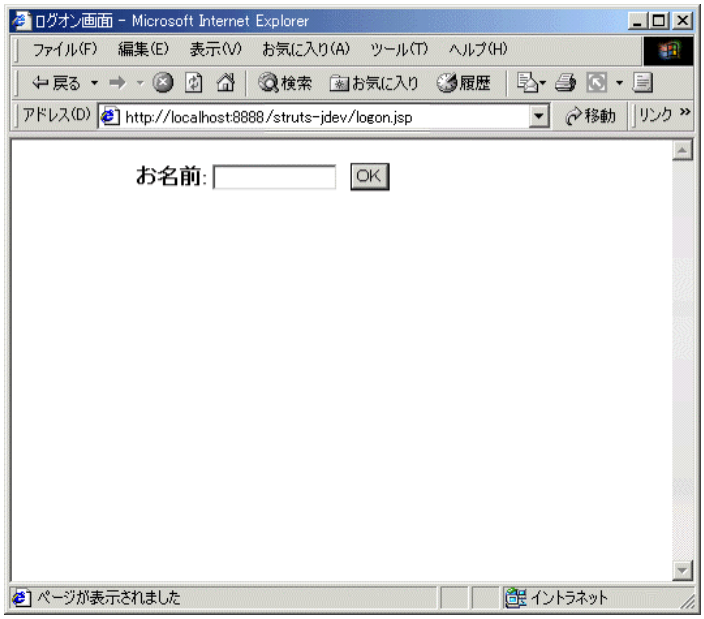
ログオン

では、画面を表示してみましょう。下記アドレスに Web ブラウザからアクセスしましょう。

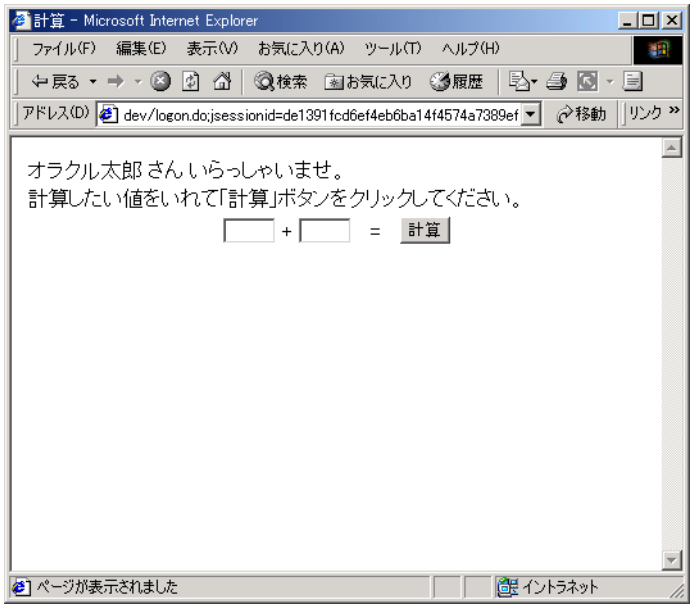
`http://<`**マシン名**`>:8888/struts-jdev/logon.jsp`

例） `http://localhost:8888/struts-jdev/logon.jsp`

もしここでログオン画面が正しく表示されない場合は、第 2 章からもう一度チェックして下さい。

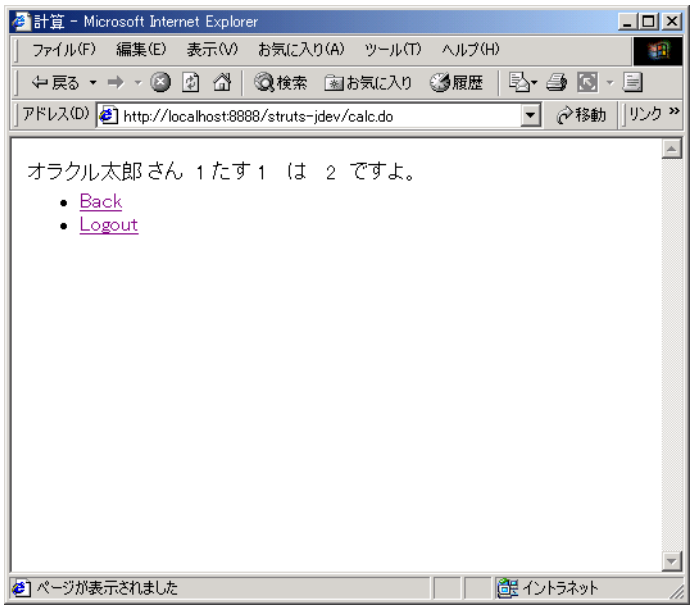


では、ログオンしてみましょう。「お名前」を入力して OK ボタンをクリックしてください。ここでは、例として「オラクル太郎」と入力します。ログオンが成功すると次の画面が表示されます。



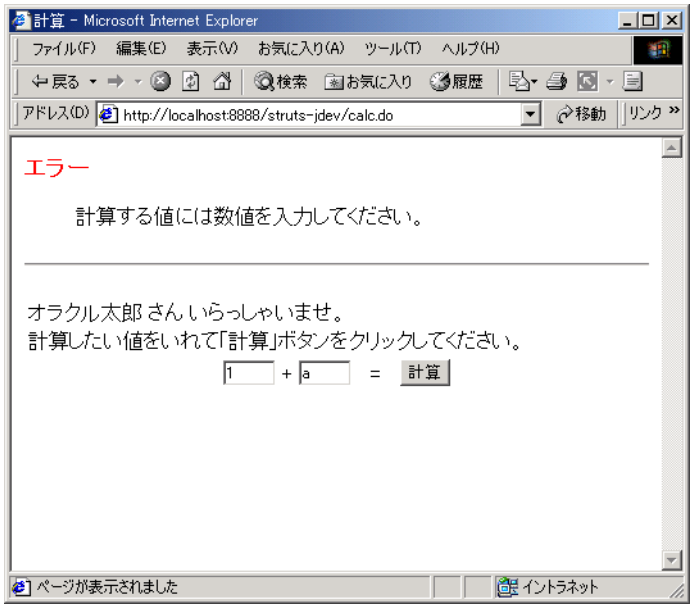
計算

次に、計算をしてみましょう。適当な数値を入力して「計算」ボタンをクリックしてください。ここでは、例として、両方とも「1」を入力しています。計算が成功すると次の画面が表示されます。



再計算

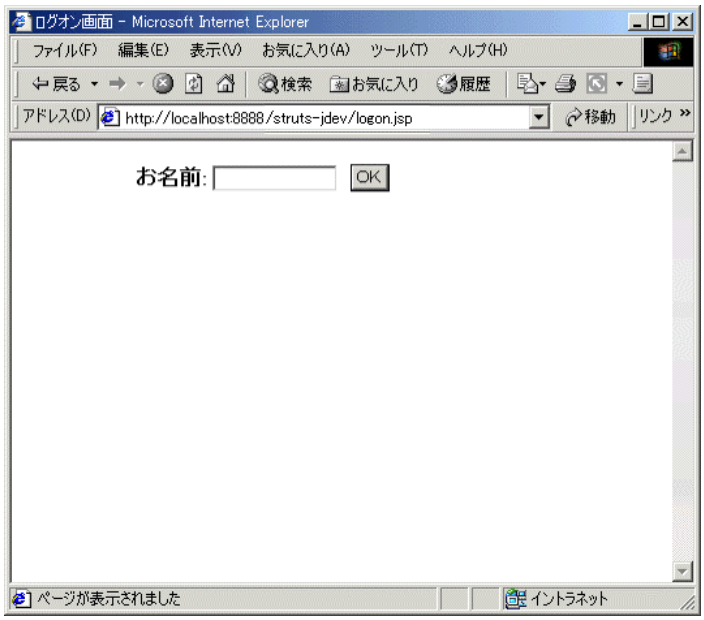
「Back」リンクをクリックしてください。また、計算画面に戻るので、計算したい値を入力します。ここでは、例として、計算できない値「1」と「a」を入力してみます。



計算できないのでエラーメッセージが表示されます。

ログアウト

「Logout」リンクをクリックしてください。ログオン画面に戻ります。



6. 最後に

Oracle9i JDeveloper による Struts のアプリケーションの構築を通して、JDeveloper にオープンソースのフレームワークを組み込むことが簡単で、より生産的な開発がおこなえることがわかります。

Java による実装、XML をベースとしたメタデータの管理、そしてタグ・ライブラリによるプレゼンテーション層に対する部品の提供など J2EE に準拠した JDeveloper および Struts の親和性がこれを実現しています。

JDeveloper は今回のようにオープンソースのフレームワークを利用することも、内蔵されている 2 つのフレームワーク ～データアクセスのための Business Components for Java（BC4J）と Web アプリケーションのための User Interface XML（UIX）～ を利用して、より生産性の高い開発をおこなうことも可能です。

どちらを選択するかは開発プロジェクトの方針によります。最低限の開発ルールを設定し、開発者の技量に任せた開発をするのであれば Struts のようなオープンソースのフレームワークを、最適化されたコンポーネントを用いて開発生産性や保守性を優先した開発をおこなうのであれば、BC4J と UIX を組み合わせて利用することをお勧めします。

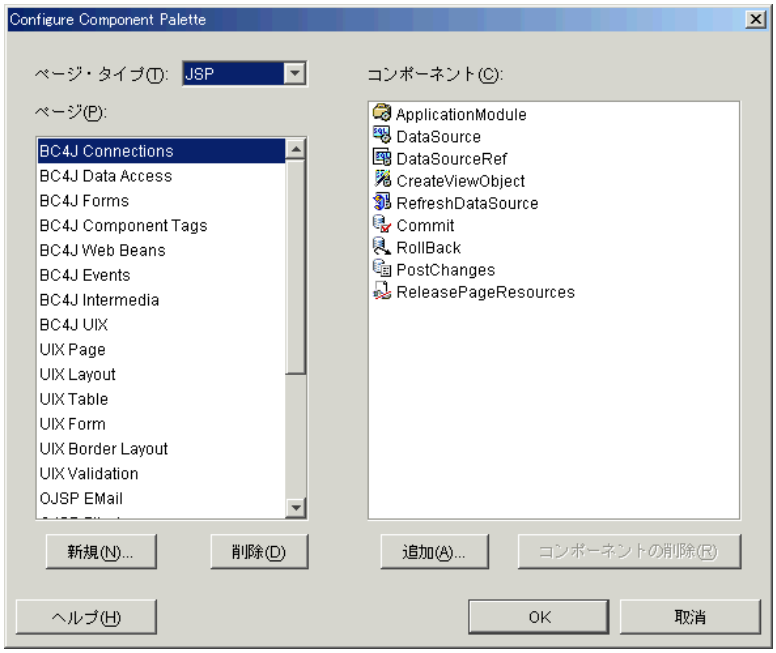
なお、JDeveloper の今後のリリースでは、Struts アプリケーション開発のより高度な機能が追加されます。本書で行ったいくつかの追加の設定作業が事前に行われた形でリリースされ、さまざまな開発支援機能も盛り込まれる予定です。

付録

ここでは、Oracle9i JDeveloper で Struts アプリケーション開発を行う上で設定しておくと便利な事柄について説明しています。

Struts 1.0.2 タグ・ライブラリの設定

JDeveloper には、JSP へのカスタム・タグの記述を支援するため、コンポーネント・パレットからのカスタム・タグの挿入機能があります。このコンポーネント・パレットには、いくつもの事前登録済みカスタム・タグが用意されていますが、ユーザーがここに新たにタグを追加することもできます。このタグの追加は、通常、メニューから「ツール→パレットの設定」を選択することで表示される「Configure Component Palette」ウィンドウから行います。



「新規」ボタンをクリックして、「新規パレット・ページ」を作成し、そこにタグを登録する、という作業手順でコンポーネントを追加できます。

本来は上記のような手順で行うのですが、ここでは、手順を簡略化するために、次のように、直接設定ファイルに書き込む方法で Struts カスタム・タグをコンポーネント・パレットに追加します。

1. JDeveloper をいったん終了します。
2. <JDeveloper_Home>¥jdev¥system¥palette.xml をテキスト・エディタで開きます。
3. 57 ページに記載されているタグ・コードを palette.xml の最後「</palette>」の直前に挿入します。

palette.xml

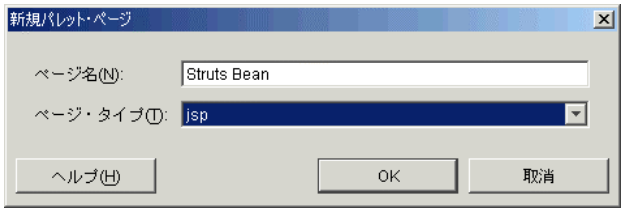
```
<?xml version = '1.0' encoding = 'Shift_JIS'?>
<palette xmlns="...">
      :
     <type>jsp</type>
     <view>list</view>
   </page>
   <!-- この位置に挿入する -->
</palette>
```

4. 追加したコード部分の最初にある「D:¥Users¥Struts¥jakarta-struts-1.0.2¥…」の部分を、自分の環境に合わせて書き直します（4 箇所あります）。
5. <JDeveloper_Home>¥jdev¥system¥palette.xml を保存し、JDeveloper を再起動します。
6. JDeveloper のメニューから「ツール→パレットの設定」を選択し「Configure Component Palette」ウィンドウを開きます。
7. 「新規」ボタンを押して、「新規パレット・ページ」ダイアログを表示します。ここで

   ページ名: Struts Bean

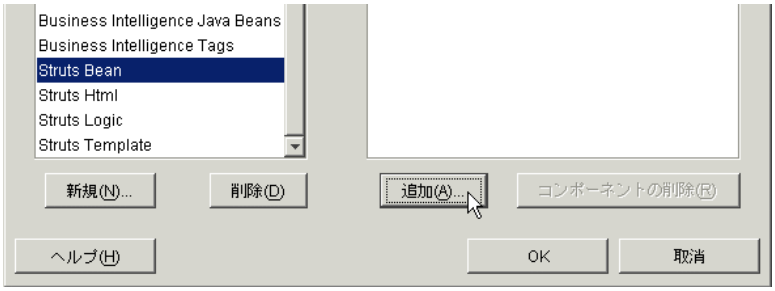
   ページ・タイプ: jsp

   として OK ボタンをクリックします。

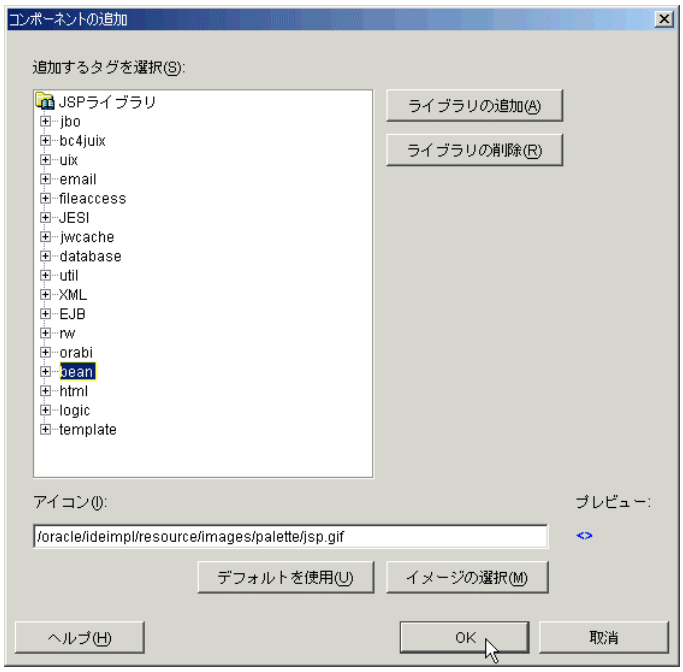


   同様にして、「Struts Html」、「Struts Logic」、「Struts Template」のページも作成します。

8. 「Configure Component Palette」ウィンドウで、「Struts Bean」ページを選択した状態で、コンポーネントの「追加」ボタンを押します。



9. 「コンポーネントの追加」ダイアログが開きます。ここで、追加するタグのツリーから「bean」を選択して「OK」をクリックします。



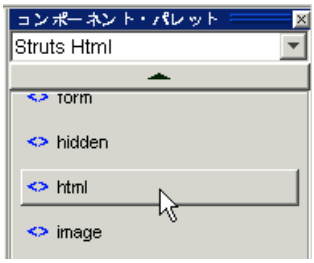
10. タグのインストールを確認するダイアログが表示されますので、「はい」を選択します。

これで、「Struts Bean」に対応したカスタム・タグが登録されます。

11. 同様の操作を「Struts Html」、「Struts Logic」、「Struts Template」のページに対しても行います。

以上の操作で、コンポーネント・パレットに、以下のパレットが追加されます。

- Struts Bean
- Struts Html
- Struts Logic
- Struts Template



palette.xml に追加するタグ・コード

見やすさのため、次のようなコード部分に改行を入れています（4 箇所）。

```
<location>D:¥Users¥Struts¥ /
        jakarta-struts-1.0.2¥lib¥struts.jar</location>
```

これは、実際には以下のようにする必要があります。

```
<location>D:¥Users¥Struts¥ jakarta-struts-1.0.2¥lib¥struts.jar</location>
```

また、このディレクトリの指定も、自分の環境に合わせて書き換える必要があります。

```
<jsplibrary>
   <displayName>/WEB-INF/struts-bean.tld</displayName>
   <location>D:¥Users¥Struts¥ /
             jakarta-struts-1.0.2¥lib¥struts.jar</location>
   <name>META-INF/tlds/struts-bean.tld</name>
   <prefix>bean</prefix>
</jsplibrary>
<jsplibrary>
   <displayName>/WEB-INF/struts-html.tld</displayName>
   <location>D:¥Users¥Struts¥ /
             jakarta-struts-1.0.2¥lib¥struts.jar</location>
   <name>META-INF/tlds/struts-html.tld</name>
   <prefix>html</prefix>
</jsplibrary>
<jsplibrary>
   <displayName>/WEB-INF/struts-logic.tld</displayName>
   <location>D:¥Users¥Struts¥ /
             jakarta-struts-1.0.2¥lib¥struts.jar</location>
   <name>META-INF/tlds/struts-logic.tld</name>
   <prefix>logic</prefix>
</jsplibrary>
<jsplibrary>
   <displayName>/WEB-INF/struts-template.tld</displayName>
      <location>D:¥Users¥Struts¥ /
             jakarta-struts-1.0.2¥lib¥struts.jar</location>
   <name>META-INF/tlds/struts-template.tld</name>
   <prefix>template</prefix>
</jsplibrary>
```

サンプルプログラムおよびコンポーネント・パレット追加用のコードについて

本書中で作成したサンプルプログラムは、OTN-J より、サンプルコードとして公開されています。

OTN-J トップページから、「サンプルコード→Oracle9i JDeveloper」と進んで、ダウンロードしてください。