Oracle JDeveloper for Windows NT

JSP アプリケーションのための Apache 利用ガイド

リリース 3.1

2000 年 7 月

部品番号: J02143-01

目次

はじめに

このドキュメントでは、Oracle JDeveloper 3.1 のビジネス･コンポーネント JSP アプリケーション・ウィザードを使用して作成された JSP アプリケーションを、Apache 上で実行させるために必要な設定方法を説明します。

なお、このドキュメントを作成するために、次の環境で検証を行っています。

ソフトウェア	Windows 環境	UNIX/Linux 環境
OS	Windows NT 4.0（Service Pack 6a）	Solaris 2.6
Apache	バージョン 1.3.12	バージョン 1.3.9
JServ	バージョン 1.1	バージョン 1.1
Tomcat	バージョン 3.1	バージョン 3.1
JDK	Java2 SDK v 1.2.2 005	Java2 SDK v1.2.2 005
その他		gcc、GNU make を使用

ドキュメントの表記規則

このドキュメントで使用される表記規則は、次のとおりです。

規則	例	意味
太字	**sample.jsp**	ファイル名
	configure	ユーティリティ
	http://www.oracle.com/	URL
山カッコで囲まれたイタリック	*<file1>*	テキスト内の可変部分
	<JDev>	JDeveloper のインストール・ディレクトリ
	<Apache>	Apache のインストール・ディレクトリ
	<Tomcat>	Tomcat がインストール・ディレクトリ
クーリエ･フォント	`apachectl start`	表示どおりに入力するテキスト

第 1 章　Apache のインストールと設定

この章では、Apache を Web サーバーとして導入するためのインストール方法および必要最低限の設定方法について説明します。

Apache のインストール

Windows の場合

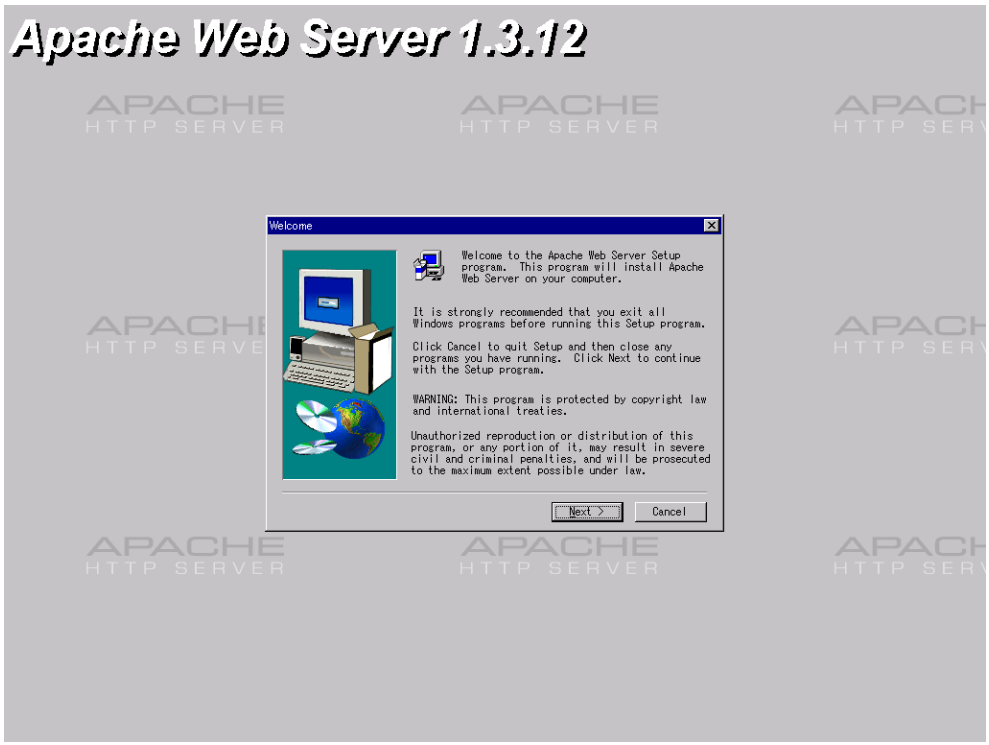
Windows 版は、インストーラがついた*.exe 形式で提供されています。Windows 版をインストールするための手順は次のとおりです。

1. 次の URL から **apache_1_3_12_win32.exe**（3.0MB）をダウンロードします。

   http://www.apache.org/dist/binaries/win32/

2. ダウンロードした **apache_1_3_12_win32.exe** をダブル・クリックします。インストーラが起動されるので、インストールするフォルダの選択など、インストーラの指示に従ってください。

図 1 - 1　Apache のインストーラ



UNIX/Linux の場合

UNIX/Linux 版には、ダウンロードしたソース・コードをユーザー自身がコンパイルして使用するタイプと、コンパイル済のタイプ（バイナリ）があります。ここでは、ソース・コードをコンパイルしてインストールする方法について説明します。

なお、コンパイル済のタイプを使用する場合は、次の URL からプラットフォームを選択し、ダウンロードしてください。

http://www.apache.org/dist/binaries/

Apache のソース・コードをコンパイルし、インストールする手順は次のとおりです。

1. 次の URL から **apache_1.3.9.tar.gz** をダウンロードします。

   http://www.apache.org/dist/old/

2. 次の 2 つにパスが通っていることを確認します。

   - ANSI 準拠の C コンパイラ（GCC など）
   - Perl 5

3. ダウンロードした **apache_1.3.9.tar.gz** を展開します。これにより、**apache_1.3.9** ディレクトリが生成されます。

```
% gzip -d apache_1.3.9.tar.gz
% tar xvf apache_1.3.9.tar
```

4. カレント・ディレクトリを **apache_1.3.9** に移動し、Apache をコンパイルするための環境設定を行うユーティリティ **configure** を実行します。

   Apache に DSO（Dynamic Shared Object）という機能を組み込むことで、後で JServ を追加する際に、Apache を再コンパイルする必要がなくなります。DSO を組み込むには、**configure** に次のオプションをつけて実行します。

```
--enable-rule=SHARD_CORE --enable-module=so
```

   次はその実行例です。

```
% ./configure --prefix=〈Apache をインストールするディレクトリ名〉 ¥
    --enable-rule=SHARD_CORE --enable-module=so
```

   備考： **configure** のその他のオプションは、**../apache_1.3.9/INSTALL** を参照してください。

5. 次のコマンドを実行して Apache をコンパイルし、インストールします。

```
% make install
```

設定ファイルの編集

Apache を実行するための設定情報は、***<Apache>*¥conf¥httpd.conf** に記述されています。初めて Apache を起動する前に、少なくとも次の設定について確認してください。

Port

HTTP プロトコルが使用するネットワークのポート番号を設定します。1～65535 の任意の番号を選択できます。デフォルト値は 80 です。他の Web サーバーがインストールされている場合は、設定したポート番号がすでに使用されていないかを確認する必要があります。

また、UNIX/Linux 環境で 1～1023 のポートを使用するためには、Apache を root 権限で起動する必要があります。

ServerName

Apache が自分自身を識別するためのホスト名または IP アドレスを入力します。ただし、ホスト名を設定する場合には、そのホスト名が DNS から参照可能である必要があります。

備考: 更に詳細な設定が必要な場合は、次の URL を参照してください。

- **http://<*ServerName*>:<*Port*>/manual/index.html**
- **http://www.apache.org/docs/**

Apache の起動と停止

Windows の場合

Windows 版をインストールすると、スタート・メニューに Apache を起動するためのショートカットが登録されます。

コマンド・プロンプトから起動、停止および再起動を行う場合は、次のコマンドを使用します。

操作	コマンド
起動	<*Apache*>¥apache
停止	<*Apache*>¥apache -k shutdown
再起動	<*Apache*>¥apache -k restart

Windows NT のサービスの作成

Windows 環境では、コマンド・ラインから Apache を起動した場合、Windows からログオフすると Apache は停止します。Apache を Windows NT サービスとして登録し、実行することで、Windows からログオフしても Apache を停止させないようにすることが可能です。

Apache 1.3.12 では、インストール時にスタート・メニューに登録される「Apache Install as Service」というショートカットを実行することにより、サービスが作成されます。

また、サービスをコマンド・ラインから作成する場合には、次のコマンドを実行します。

```
> <Apache>¥apache -i -n <サービス名>
```

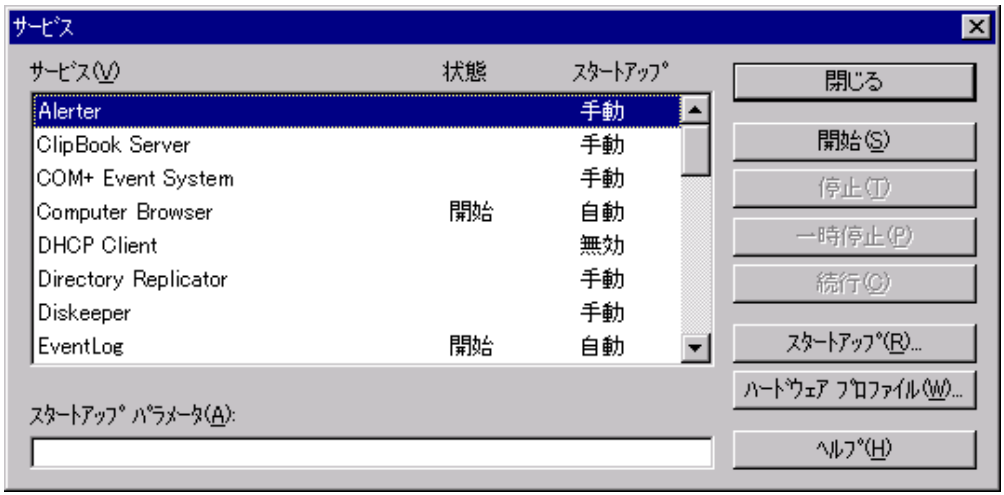
備考: 「-n <サービス名>」を省略した場合、サービス名は「Apache」となります。

また、作成したサービスを削除するには、次のコマンドを実行します。

```
> <Apache>¥apache -u -n <サービス名>
```

作成されたサービスは、コントロール・パネルの「サービス」で開始/停止できます。

図 1 - 2　コントロール・パネルの「サービス」



サービスをコマンド・ラインから開始/停止する場合は、次のコマンドを使用します。

操作	コマンド
サービスの開始	`net start apache`
サービスの停止	`net stop apache`

UNIX/Linux の場合

UNIX/Linux 環境で、Apache を起動、停止および再起動するためのコマンドは、次のとおりです。

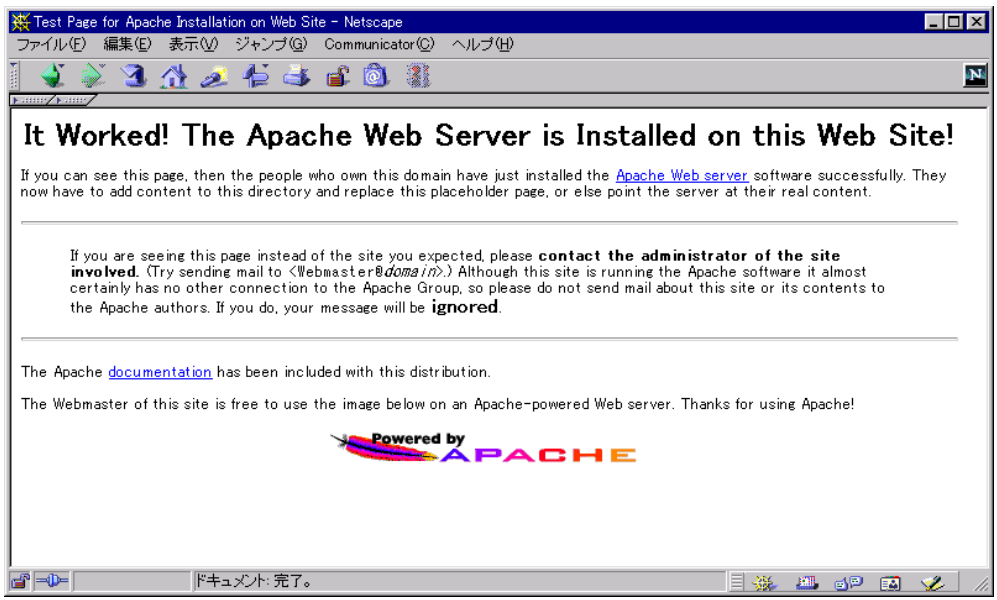
操作	コマンド
起動	<*Apache*>/bin/apachectl start
停止	<*Apache*>/bin/apachectl stop
再起動	<*Apache*>/bin/apachectl restart

テスト

Apache を起動後、ブラウザで次の URL にアクセスすると、次の図のように表示されます。

http://<*ServerName*>:<*Port*>/

図 1 - 3　Apache の起動確認



第 2 章　Apache JServ のインストールと設定

インストール前の設定

Apache JServ を使用するためには、次の 2 つがインストールされている必要があります。

- Java Development Kit 1.1.x または Java2 SDK 1.2 以降
- Java Servlet Development Kit 2.0

Java2 SDK のインストール

このドキュメントでは、Java2 SDK v1.2.2 を基に説明しています。Java2 SDK v1.2.2 のインストール方法の詳細は、次の URL を参照してください。

Windows 版

http://java.sun.com/products/jdk/1.2/ja/download-windows.html

Solaris 版

http://www.sun.com/software/solaris/java/download.html

Linux 版

http://java.sun.com/products/jdk/1.2/ja/download-linux.html

JSDK 2.0 のインストール

JSDK 2.0 は次の URL よりダウンロードが可能です。

http://java.sun.com/products/servlet/download.html

注意： JSDK ではすでに 2.1 というバージョンもリリースされていますが、Apache JServ が対応しているのは JSDK 2.0 のみです。

Windows 版

jsdk20-win32.exe をダウンロードし、ダブル・クリックします。インストーラが起動されるので、インストールするディレクトリの入力など、指示に従ってください。

UNIX/Linux 版

jsdk20-solaris2-sparc.tar.Z をダウンロードし、適当なディレクトリに展開してください。

備考: ファイル名に"solaris2-sparc"とありますが、Solaris 以外のオペレーティング・システムでも使用可能です。

JServ のインストール

Windows の場合

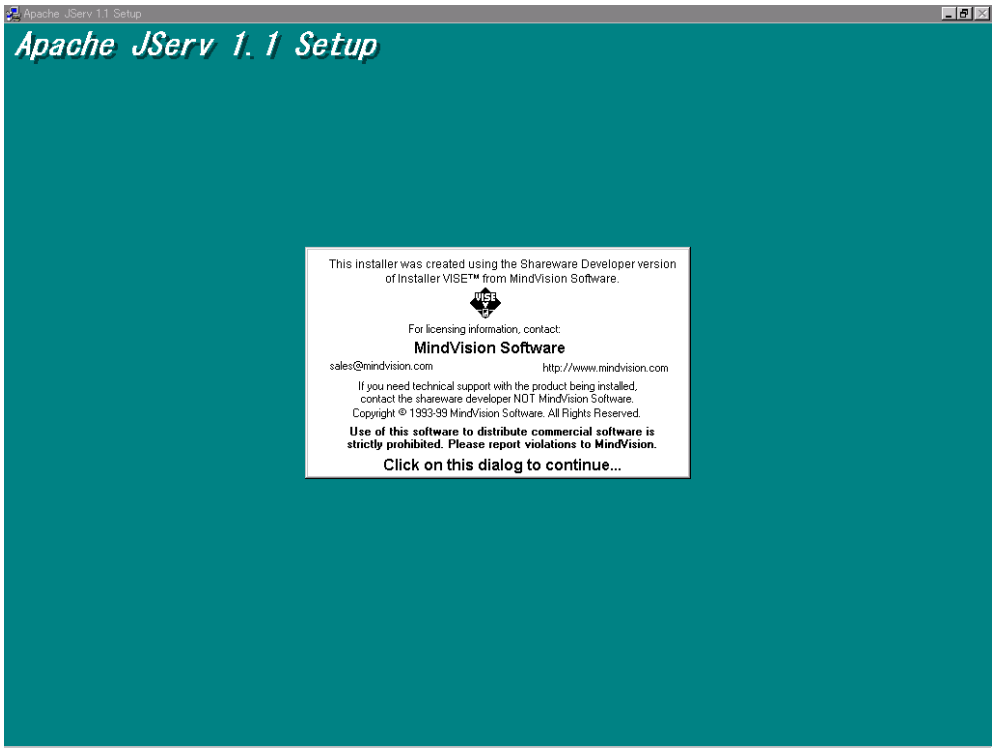
Apache と同様に、Windows 版の JServ もインストーラ付きの*.exe 形式で提供されています。Windows 版のインストール手順は、次のとおりです。

1. 次の URL から **ApacheJServ-1.1.exe** をダウンロードします。

   http://java.apache.org/jserv/dist/

2. ダウンロードした **ApacheJServ-1.1.exe** をダブル・クリックし、インストーラを起動します。

図 2 - 1　JServ のインストーラ



3. インストーラでは、次の情報を入力します。
   - JServ をインストールするディレクトリ名
   - **java.exe** がインストールされているディレクトリ
   - JSDK2.0 がインストールされているディレクトリ
   - Apache の設定情報が記述された **httpd.conf** が保存されているディレクトリ

UNIX/Linux の場合

UNIX/Linux 版の JServ にはインストーラが用意されていないため、ダウンロードしたソース・コードをコンパイルする必要があります。JServ のソース・コードをコンパイルし、インストールする手順は次のとおりです。

1. 次の URL から **ApacheJServ-1.1.tar.gz** をダウンロードします。

   http://java.apache.org/jserv/dist/

2. ダウンロードした **ApacheJServ-1.1.tar.gz** を次の例のように展開します。これにより、**ApacheJServ-1.1** ディレクトリが生成されます。

```
% gzip -d ApacheJServ-1.1.tar.gz
% tar xvf ApacheJServ-1.1.tar
```

3. カレント・ディレクトリを **ApacheJServ-1.1** に移動して、JServ をコンパイルするための環境設定を行うユーティリティ **configure** を実行します。次はその実行例です。

```
% ./configure ¥
    --prefix=<JServ をインストールするディレクトリ> ¥
    --with-jdk-home=<Java2 SDK がインストールされたディレクトリ> ¥
    --with-JSDK=<JSDK2.0 がインストールされたディレクトリ>/lib/jsdk.jar ¥
    --with-java-platform=2 ¥
    --with-apxs=<Apache>/bin/apxs
```

   備考: **configure** のその他のオプションの詳細は、**../ApacheJServ-1.1/INSTALL** を参照してください。

4. 次のコマンドを実行して JServ をコンパイルし、インストールします。

```
% make install
```

5. Apache のログが保存されている**<*Apache*>/logs** ディレクトリの書き込み権限を、**httpd.conf** の User/Group で定義されているオペレーティング・システムのユーザーおよびグループに対して与えます。

設定ファイルの編集

この項では、JServ を実行するための必要最低限の設定方法について説明します。**httpd.conf** に JServ を実行するために必要な情報を追加するため、次の手順が必要です。

1. **httpd.conf** の最終行に次の 1 行を追加します。なおこの行は、Windows 版 JServ のインストール時に「Do you want to modify the file “httpd.conf” in order to include Apache JServ」で「はい」を選択すると、自動的に追加されます。

```
# Windows 版の場合
include "<jserv>/conf/jserv.conf"
```

```
#UNIX/Linux の場合

include "<Apache>/conf/jserv/jserv.conf"
```

2. UNIX/Linux 版の場合、**httpd.conf** の 200 行目付近に次の 1 行を追加します。これにより、JServ のモジュールがロードされます。

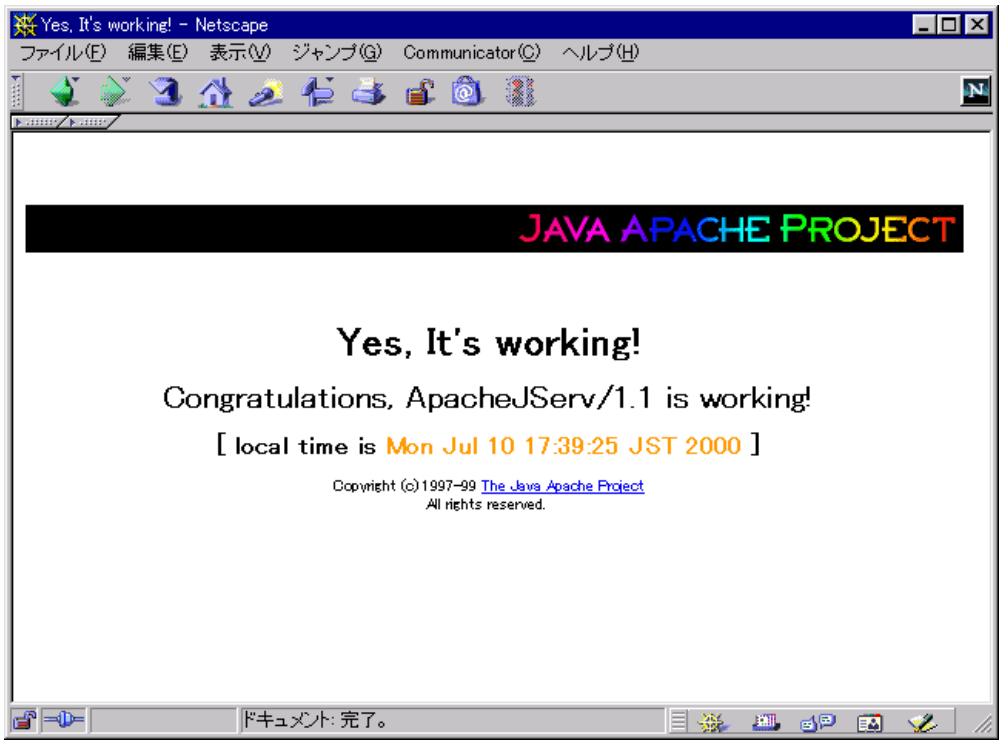
```
LoadModule jserv_module libexec/mod_jserv.so
```

テスト

Apache が起動されると、自動的に JServ も起動されます。JServ が起動されているかどうかは、次の URL にアクセスすることで確認できます。

http://<*ServerName*>:<*Port*>/servlets/IsItWorking

図 2 - 2　JServ の起動確認



備考: 前述で紹介した**/servlets/IsItWorking** 上部のイメージは、デフォルトの設定では Apache が稼働中のマシン以外からアクセスした場合、表示されません。また、使用する環境によっては、Apache が稼働中のマシンからのアクセスであっても表示されないことがあります。このイメージを表示させるためには、**<*JServ*>¥conf¥jserv.conf**（UNIX/Linux の場合は、**<*Apache*>/conf/jserv.conf**）を開き、136 行目を次の例のように変更してください。

```
<Location /jserv/>
  SetHandler jserv-status
  order deny,allow
  deny from all
  allow from <アクセスするマシンの IP アドレス>
</Location>
```

Oracle JSP Engine の設定

Apache と JServ だけでは JSP アプリケーションを実行することはできません。そこで JDeveloper には、Oracle JSP Engine という JSP1.0 に準拠した JSP エンジンがバンドルされています。Oracle JSP Engine は、JServ 上でサーブレットとして実行されます。Oracle JSP Engine を使用するためには、次の 3 つのファイルを編集する必要があります。

- **jserv.properties**
- **jserv.conf**
- **zone.properties**

jserv.properties の編集

まず、Oracle JSP Engine が JServ 上でサーブレットとして動作するために必要なクラス・パスの設定を行います。

1. JDeveloper がインストールされているマシンから、Apache を実行するマシンの任意のディレクトリに次の 2 つの JAR ファイルをコピーします。
   - **<*JDev*>¥lib¥ojsp.jar**
   - **<*JDev*>¥jswdk-1.0.1¥lib¥servlet.jar**
2. **jserv.properties** に次の 3 行を追加します。

```
# Windows 環境で、ojsp.jar および servlet.jar を C:¥Sample にコピーした場合
wrapper.classpath=C:¥Sample¥ojsp.jar
wrapper.classpath=C:¥Sample¥servlet.jar
wrapper.classpath=%Java2%¥lib¥tools.jar
```

備考: jserv.properties は次のディレクトリに保存されています。

- Windows 版: ***<JServ>*¥conf**
- UNIX/Solaris 版: ***<Apache>*/conf/jserv**

jserv.conf の編集

拡張子が .jsp であるファイルが Oracle JSP Engine によって処理されるよう設定するために、**jserv.conf** に次の 1 行を追加します。

```
ApJservAction .jsp /servlets/oracle.jsp.JspServlet
```

備考: **jserv.conf** は次のディレクトリに保存されています。

- Windows 版: ***<JServ>*¥conf**
- UNIX/Linux 版: ***<Apache>*/conf/jserv**

zone.properties の編集

Oracle JSP Engine を実行する際に必要な初期化パラメータの設定を、次のように行います。

1. **zone.properties** を次のように編集します。

```
# ブラウザから入力された文字列を正しく変換する（必須）
servlet.oracle.jsp.JspServlet.initArgs=translate_params=true
# JSP 実行時に生成される Java ソース/クラスファイルを保存するディレクトリ
# 指定されていない場合は<Apache>¥htdocs¥_pages
servlet.oracle.jsp.JspServlet.initArgs=page_repository_root=<ディレクトリ名>
# JServ 起動時にあらかじめ Oracle JSP Engine を起動させる
servlets.startup=oracle.jsp.JspServlet
```

2. UNIX/Linux の場合、page_repository_root で指定したディレクトリの書き込み権限を、**httpd.conf** の User/Group で指定されているユーザーおよびグループに対して与える必要があります。

   備考: **jserv.conf** は次のディレクトリに保存されています。

   - Windows 版: <*JServ*>¥servlets
   - UNIX/Linux 版: <*Apache*>/conf/jserv

テスト

Oracle JSP Engine の設定が正しいかどうかは、次の手順で確認できます。

1. 次のような簡単な JSP ファイルを作成します。

```
<%@ page language="java" contentType="text/html;charset=SHIFT_JIS"%>
<html>
<body>
<H1>Oracle JSP Engine</H1>
<% out.println("こんにちは！"); %>
</body>
</html>
```

2. 作成した JSP ファイルを、***<Apache>*¥htdocs** ディレクトリに保存します。
3. 次の URL にアクセスします。

   http://*<ServerName>*:*<Port>*/*<JSP ファイル名>*

JSP アプリケーションの配布

作成したビジネス・コンポーネント JSP アプリケーションを実行するためには、次のファイルを Apache が実行されているマシンへ配布する必要があります。

- ビジネス・コンポーネントを定義する Java クラスおよび XML ファイル（ローカル・モードで実行する場合のみ）
- ビジネス・コンポーネント JSP アプリケーション・ウィザードによって生成された*.jsp ファイルおよび*.properties ファイル
- 作成した Web Bean

- ビジネス・コンポーネント JSP アプリケーションを実行する際にクラス・パスに追加する必要のある Java ライブラリ
- ビジネス・コンポーネント JSP アプリケーションが使用するイメージ・ファイルおよびカスケーディング・スタイル・シート

ビジネス・コンポーネントの配布

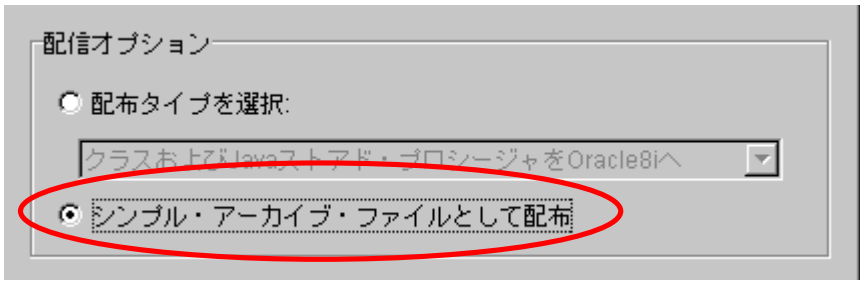
ビジネス・コンポーネント JSP アプリケーションをローカル・モードで実行する場合（ビジネス・コンポーネントが Oracle8*i* Jserver などに EJB や CORBA として配布されていない場合）は、次の手順でビジネス・コンポーネントを定義する Java クラスおよび XML ファイルを配布する必要があります。

1. JDeveloper で、ビジネス・コンポーネントが定義されたプロジェクトを開きます。
2. メニュー・バーから「プロジェクト」→「配布」→「新規プロファイル」を選択し、配布プロファイル・ウィザードを起動します。

図 2 - 3　配布プロファイル・ウィザード「初期画面」

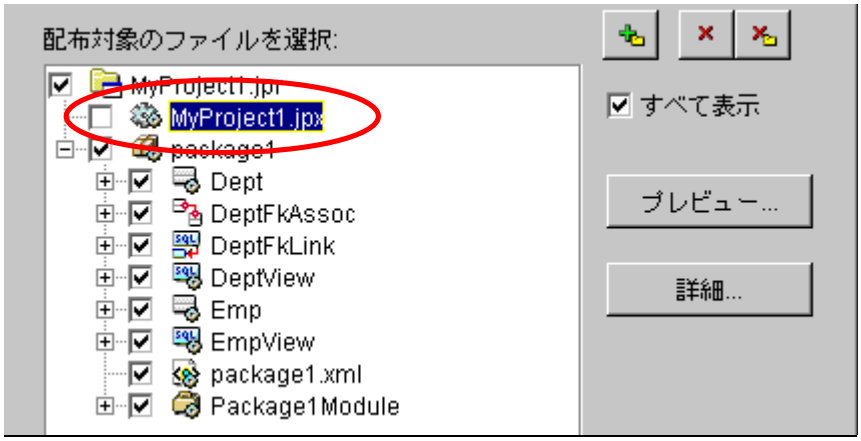
3. 配布プロファイル・ウィザード「ステップ 1」の「配信オプション」で「シンプル・アーカイブ・ファイルとして配布」を選択し、「次へ>」ボタンをクリックします。

図 2 - 4　配布プロファイル・ウィザード「ステップ 1」



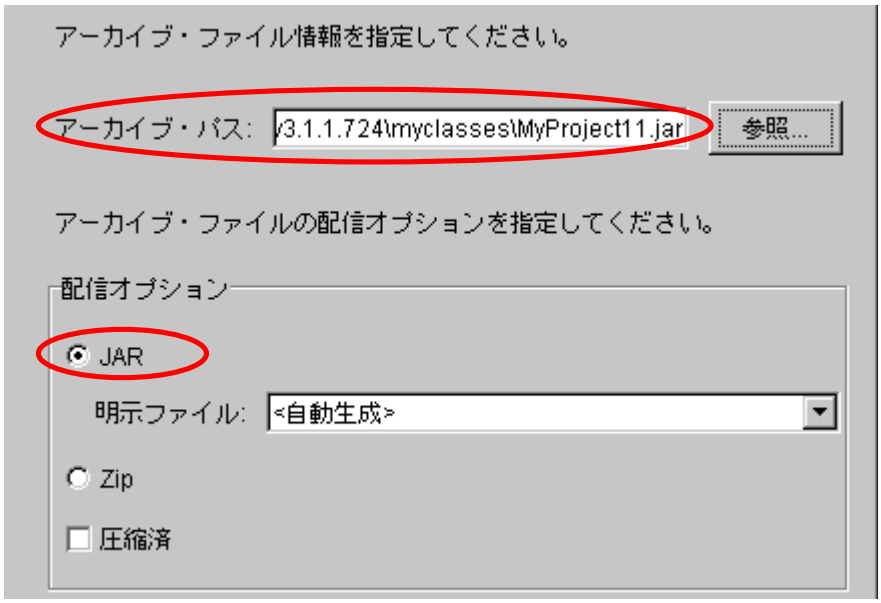
4. 配布プロファイル・ウィザード「ステップ 2」では、JAR ファイルに含めるファイルの選択を行います。「配布対象のファイルを選択」で<プロジェクト名>.jpx ファイルのチェックを外してから、「次へ>」ボタンをクリックします。

図 2 - 5　配布プロファイル・ウィザード「ステップ 2」



5. 配布プロファイル・ウィザード「ステップ 3」では、「アーカイブ・パス」フィールドに JAR ファイル名を絶対パスで入力し、「配信オプション」で「JAR」を選択してから、「完了」ボタンをクリックします。

図 2 - 6　配布プロファイル・ウィザード「ステップ 3」



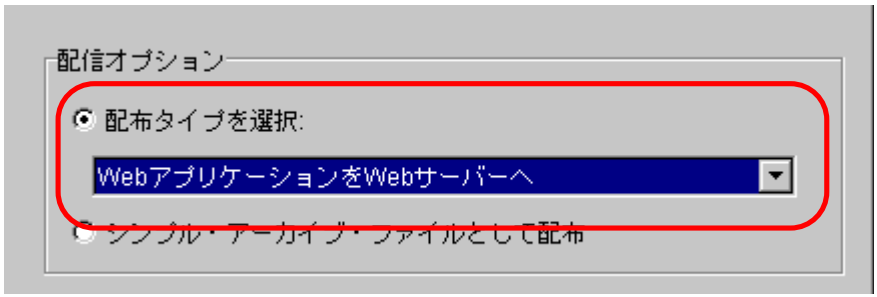
7. 作成された JAR ファイルを、Apache がインストールされているマシンの適当なディレクトリへコピーします。

配布プロファイル・ウィザードを使用した JSP アプリケーションの配布

JDeveloper と Apache が同一マシンにインストールされている場合や、Apache が稼働中のマシンに対してネットワーク・ドライブが設定されている場合は、配布プロファイル・ウィザードを使用して、次の手順で JSP アプリケーションを配布できます。

1. 配布するビジネス･コンポーネント JSP アプリケーションのプロジェクトを JDeveloper で開きます。
2. 配布プロファイル・ウィザードを起動します。
3. 配布プロファイル・ウィザード「ステップ 1」の「配信オプション」で、「配布タイプを選択」が選択されていることを確認し、ドロップダウン・リストから「Web アプリケーションを Web サーバーへ」を選択します。

図 2 - 7　配布プロファイル・ウィザード「ステップ 1」

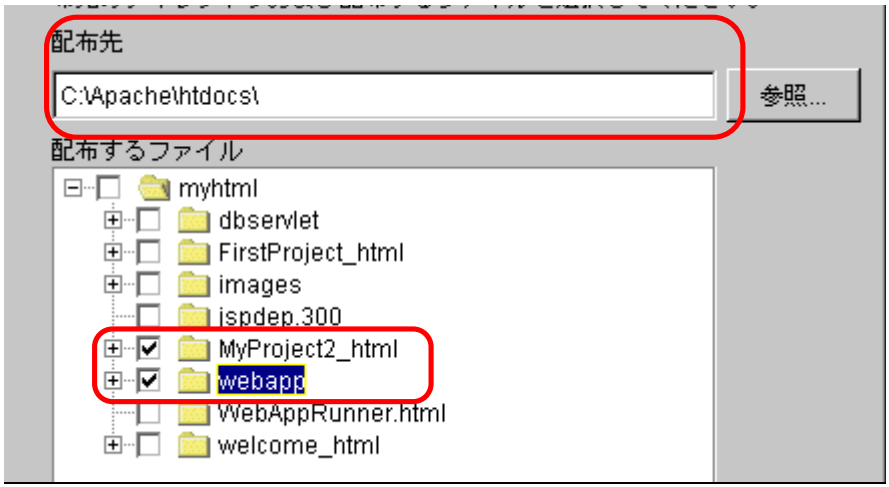


4. 配布プロファイル・ウィザード「ステップ 2」では、配布先のディレクトリおよび配布するファイルを選択します。「配布先」フィールドには、Apache がインストールされているマシン上の適当なディレクトリを絶対パスで入力し、「配布するファイル」では、次の 2 つをチェックします。

   - 配布するビジネス･コンポーネント JSP アプリケーションのプロジェクト･プロパティで「HTML ソース・ディレクトリ」に設定されているディレクトリ
   - webapp ディレクトリ

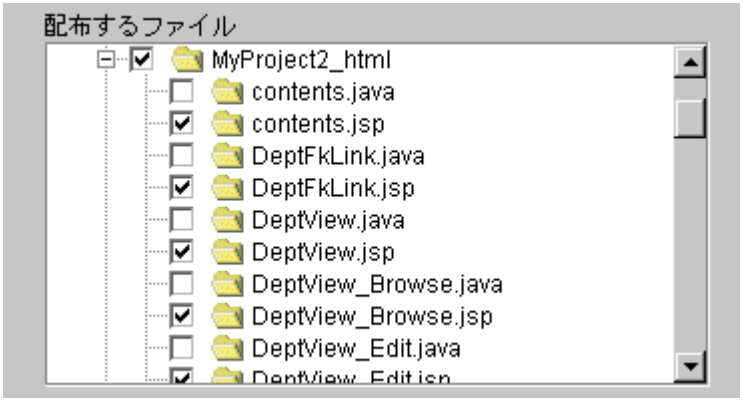
   また、webapp ディレクトリ以下のファイルがすべてチェックされていることを確認してください。

図 2 - 8　配布プロファイル・ウィザード「ステップ 2」



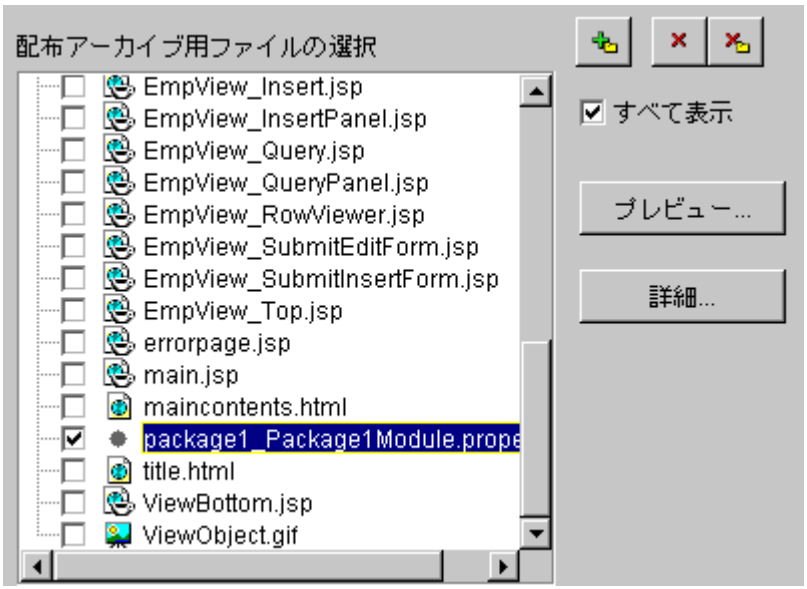
5. 次に、HTML ソース・ディレクトリ名の左にある「+」をクリックして展開し、保存されているファイルの一覧を表示させます。*.java ファイルのチェックをすべて外してから、「次へ>」ボタンをクリックします。

図 2 - 9　配布プロファイル・ウィザード「ステップ 2」



6. 配布プロファイル・ウィザード「ステップ 3」では、JAR ファイルに含めるファイルについて設定します。「配布アーカイブ用ファイルの選択」で、*.properties ファイルおよび作成した WebBean の*.java ファイルをチェックし、その他のファイルはチェックを外した状態で「詳細」ボタンをクリックします。

図 2 - 10　配布プロファイル・ウィザード「ステップ 3」

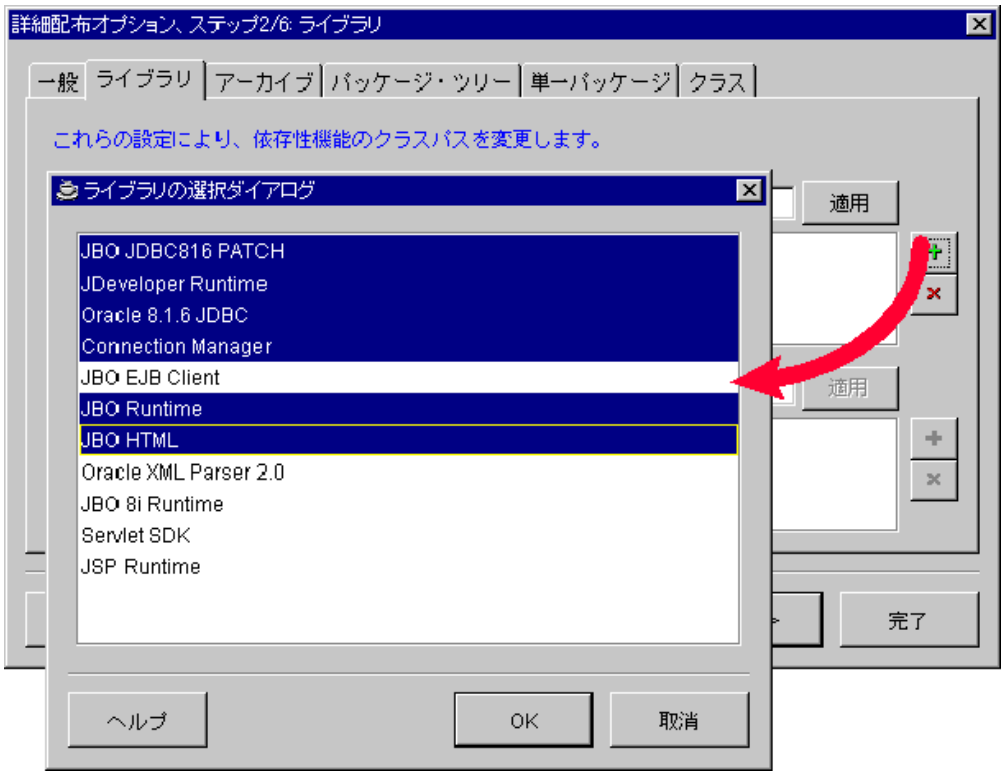


7. 「詳細配布オプション・ウィザード」で「ライブラリ」ページを表示し、ボタンをクリックして「ライブラリの選択ダイアログ」を起動します。「ライブラリの選択ダイアログ」には、プロジェクトに設定されているライブラリの一覧が表示されます。次のライブラリを選択して、「OK」ボタンをクリックします。

- JBO JDBC816 PATCH
- JBO OSQL Domains（ローカル・モードの場合は不要）
- JDeveloper Runtime
- Connection Manager
- Oracle 8.1.6 JDBC（ローカル・モードの場合のみ）
- JBO Runtime（ローカル・モードの場合のみ）
- JBO 8i Client（ビジネス・コンポーネントを JServer に CORBA として配布した場合のみ）
- JBO EJB Client（ビジネス・コンポーネントを JServer に EJB として配布した場合のみ）

- BC4J を EJB として配布した際に生成された<*Application Module* 名>EJBClient.jar（ビジネス・コンポーネントを JServer に EJB として配布した場合のみ）
- JBO HTML

図 2 - 11　詳細配布オプション「ライブラリ」ページと「ライブラリの選択ダイアログ」



8. 配布プロファイル・ウィザード「ステップ 4」では、「アーカイブ・パス」のフィールドに JAR ファイル名のみを入力し、「配信オプション」で「JAR」を選択してから「完了」ボタンをクリックします。これで JSP ファイル、イメージ・ファイル、カスケーディング・スタイル・シートおよび JAR ファイルが、配布プロファイル・ウィザード「ステップ 2」の「配布先」で指定したディレクトリにすべてコピーされます。

手動による JSP アプリケーションの配布

Apache が他のプラットフォーム上で実行されている場合など、JSP アプリケーションの配布に配布プロファイル・ウィザードが使用できない場合は、次のものを手動で配布する必要があります。

- *.jsp ファイルおよび*.properties ファイル
- ビジネス・コンポーネント JSP アプリケーションが使用する Java ライブラリと作成した WebBean
- イメージ・ファイルやカスケーディング・スタイル・シート

.jsp ファイルおよび.properties ファイル

.jsp ファイルおよび.properties ファイルは、次の手順で配布プロファイル・ウィザードを使用して JAR ファイルを作成し、Apache を実行するマシン上で解凍します。

1. 配布するビジネス・コンポーネント JSP アプリケーションのプロジェクトを開きます。
2. 配布プロファイル・ウィザードを起動します。
3. 配布プロファイル・ウィザード「ステップ 1」の「配信オプション」で「シンプル・アーカイブ・ファイルとして配布」を選択し、「次へ＞」ボタンをクリックします。
4. 配布プロファイル・ウィザード「ステップ 2」では、すべてのファイルがチェックされていることを確認し、「次へ＞」ボタンをクリックします。
5. 配布プロファイル・ウィザード「ステップ 3」では、「アーカイブ・パス」のフィールドに JAR ファイル名を絶対パスで入力し、「配信オプション」で「JAR」を選択してから「完了」ボタンをクリックします。
6. 作成された JAR ファイルを Apache を実行するマシンの**<*Apache*>¥htdocs** ディレクトリにコピーします。
7. 次の例のようにコマンドを実行し、JAR ファイルを展開します。これにより、プロジェクト・プロパティで「HTML ソース・ディレクトリ」に定義されたディレクトリ名と同名のディレクトリが生成され、その下にすべての*.jsp および*.properties ファイルが保存されます。

```
> cd <Apache>¥htdocs
> jar xf <JAR ファイル名>
```

Java ライブラリを含む JAR ファイルと WebBean

ビジネス・ロジック JSP アプリケーションが使用する Java ライブラリを Apache を実行するマシン上に配布するには、次の手順で JAR ファイルを作成し、コピーします。

1. JDeveloper で配布するビジネス・コンポーネント JSP アプリケーションのプロジェクトを開きます。
2. 配布プロファイル・ウィザードを起動します。
3. 配布プロファイル・ウィザード「ステップ 1」の「配信オプション」で「シンプル・アーカイブ・ファイルとして配布」を選択し、「次へ＞」ボタンをクリックします。
4. 配布プロファイル・ウィザード「ステップ 2」で、JAR ファイルに含めるファイルの選択を行います。「配布対象のファイルを選択」で、*.jsp および*.properties のチェックを外し、WebBean を定義した*.java ファイルのみが選択されている状態で、「詳細」ボタンをクリックします。
5. 起動された詳細配布オプション・ウィザードの「ライブラリ」ページで、  をクリックして「ライブラリの選択ダイアログ」を起動します。次のライブラリを選択して、「OK」ボタンをクリックします。
   - JBO JDBC816 PATCH
   - JBO OSQL Domains（ローカル・モードの場合は不要）
   - JDeveloper Runtime
   - Connection Manager
   - Oracle 8.1.6 JDBC（ローカル・モードの場合のみ）
   - JBO Runtime（ローカル・モードの場合のみ）
   - JBO 8i Client（ビジネス・コンポーネントを JServer に CORBA として配布した場合のみ）
   - JBO EJB Client（ビジネス・コンポーネントを JServer に EJB として配布した場合のみ）
   - BC4J を EJB として配布した際に生成された<*Application Module* 名>EJBClient.jar（ビジネス・コンポーネントを JServer に EJB として配布した場合のみ）
   - JBO HTML

6. 詳細配布オプション・ウィザードの「完了」ボタンをクリックしてから、配布プロファイル・ウィザードの「次へ>」ボタンをクリックします。

7. 配布プロファイル・ウィザード「ステップ 4」では、「アーカイブ・パス」のフィールドに JAR ファイル名を絶対パスで入力し、「配信オプション」で「JAR」を選択してから「完了」ボタンをクリックします。

8. 作成された JAR ファイルを、Apache がインストールされているマシンの適当なディレクトリにコピーします。

イメージ・ファイルおよびカスケーディング・スタイル・シート

イメージ・ファイルおよびカスケーディング・スタイル・シートは、次の手順で配布します。

1. **<*JDev*>¥redist¥webapp.zip** を Apache を実行するマシンにコピーします。

2. 次のように **webapp.zip** を**<*Apache*>¥htdocs** ディレクトリの下で解凍します。

```
% jar -xft webapp.zip
```

設定ファイルの編集

クラス・パスの設定

配布したビジネス・コンポーネント JSP アプリケーションを実行するためには、これまでに作成した JAR ファイルをクラス・パスに追加する必要があります。

クラス・パスの設定は、**jserv.properties** に次のように追加します。

```
# C:¥sample に保存されている sample1.jar と sample2.jar をクラス・パスに追加
wrapper.classpath=C:¥sample¥sample1.jar
wrapper.classpath=C:¥sample¥sample2.jar
```

また、*.properties ファイルを手動で配布した場合は、*.properties ファイルが保存されているディレクトリも同様にクラス・パスに追加します。

```
# *.properties が C:¥sample に保存されている場合
wrapper.classpath=C:¥sample
```

仮想ディレクトリの設定

JSP ファイルが**<Apache>¥htdocs** ディレクトリ以下（サブディレクトリも含む）に保存されていない場合は、仮想ディレクトリを設定します。

仮想ディレクトリの設定は、**httpd.conf** に次のように記述します。

```
# C:¥sample に対し、仮想ディレクトリを「sample」に設定
Alias /sample/ "C:¥sample"
```

JDBC OCI ドライバのための設定

JDBC OCI ドライバを使用する場合、**jserv.properties** に次のように環境変数を定義する必要があります。

```
# Windows 環境の場合
wrapper.path=<ORACLE_HOME>¥bin
# Solaris の場合: 環境変数 ORACLE_HOME と LD_LIBRARY_PATH を追加
wrapper.env=ORACLE_HOME=<ORACLE_HOME>
wrapper.env=LD_LIBRARY_PATH=<ORACLE_HOME>/lib
```

備考: 環境変数の追加のための書式

```
wrapper.env=<環境変数名>=<設定する値>
```

実行

配布したビジネス･コンポーネント JSP アプリケーションを実行するには、Web ブラウザで次の URL にアクセスします。

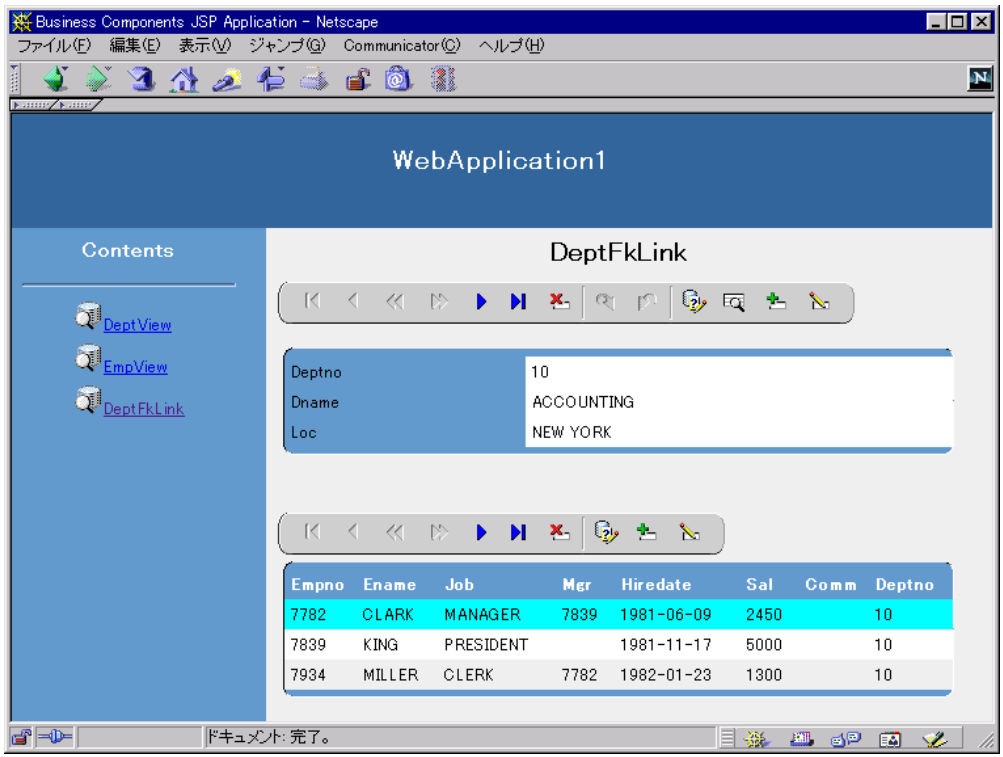
JSP ファイルが、<*Apache*>¥htdocs のサブディレクトリに保存されている場合:

http://<*ServerName*>:<*Port*>/<*サブディレクトリ名*>/main.jsp

JSP ファイルが保存されているディレクトリに仮想ディレクトリを設定した場合:

http://<*ServerName*>:<*Port*>/<*仮想ディレクトリ*>/main.jsp

図 2 - 12　実行例



リモート・デバッグのための JServ の設定

JDeveloper 2.0 には HTTP サーバーが組み込まれていたため、サーブレットのデバッグが可能でしたが、Enterprise JavaBeans や CORBA サーバー・オブジェクトをデバッグするには、ローカルのプログラムの main メソッドからそれぞれのメソッドをコールするなどの方法をとる必要がありました。

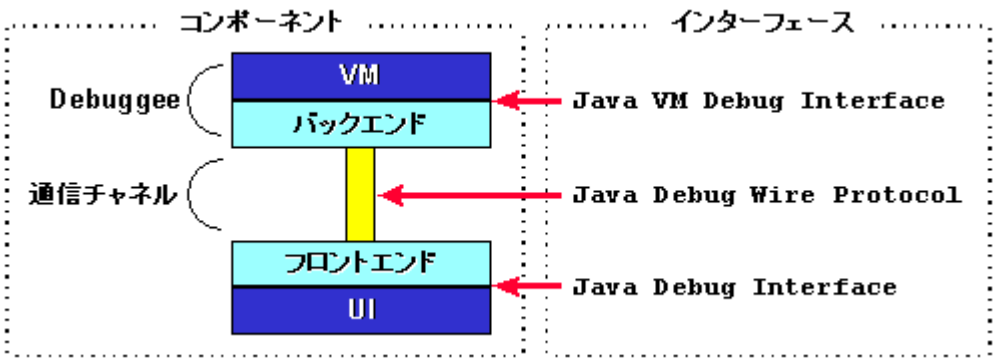
JDeveloper 3.1 では、Java Platform Debugger Architecture をサポートしたことにより、アプリケーション・サーバーに配布されている Java コンポーネントのデバッグ、つまりリモート・デバッグが可能になりました。この項では、JServ 上に配布されたサーブレットおよび JSP アプリケーションをリモート・デバッグするために必要な設定方法について説明します。

Java Platform Debugger Architecture

Java Platform Debugger Architecture（以下、JPDA）は、Java2 プラットフォームにおける、デバッグのための新しいアーキテクチャで、Java2 SDK 1.3（JDK1.3）からは標準で提供されている機能です。

JPDA は次の図のような構成になっています。

図 2 - 13　Java Platform Debugger Architecture の構成



コンポーネント名	説明
Debuggee	デバッグされるアプリケーションが実行されているプロセス。 VM とバックエンドによって構成されています。
VM	デバッグされるアプリケーションが動作している Java Virtual Machine。 Java Virtual Machine Debug Interface を実装しています。
バックエンド	フロントエンドからのリクエストを VM に渡し、VM のレスポンスをフロントエンドに返します。
通信チャネル	フロントエンドとバックエンドの通信が行われます。 プロトコルとして Java Debug Wire Protocol を使用します。
Debugger	開発者がプログラムをデバッグするための機能を提供します。
フロントエンド	Java Debug Interface の実装部。
UI	デバッグするためのユーザー・インタフェース。

JDeveloper を使用してリモート・デバッグを行う場合、使用する VM として次の 2 つから選択が可能です。

a. Oracle Java Virtual Machine（Oracle JVM）
リモート・デバッグの対象となるアプリケーション・サーバーのマシンに JDeveloper がインストールされている場合に使用可能です。

b. Java プラットフォーム・デバッグ・アーキテクチャ
JPDA によって提供されている VM。あらゆるプラットフォームで使用可能です。

JPDA のダウンロードおよびインストール方法は、次の URL を参照してください。

JPDA のダウンロード

http://java.sun.com/products/jpda/

JPDA のインストール方法

http://java.sun.com/products/jpda/installinst.html

備考: JPDA の詳細は、次の URL を参照してください。

http://java.sun.com/j2se/1.3/ja/docs/ja/guide/jpda/index.html

JServ の設定

Oracle JVM を使用する場合

Oracle Java Virtual Machine を使用する場合は、**jserv.properties** を次のように編集します。

```
# Apache JServ を起動する java.exe のパスを指定
# Oracle Java VM を使用する場合は必ず"<JDev>¥java1.2¥bin¥java.exe"を指定
wrapper.bin=<JDev>¥java1.2¥bin¥java.exe
# java.exe に渡すリモート・デバッグに必要なパラメータを指定
# 次は JDeveloper と JServ 間の通信のポート番号として 4000 を使用する場合
wrapper.bin.parameters=-ojvm -Xxdebugondemandquietport4000
# クラス・パスに<JDev>¥jswdk-1.0¥lib¥servlet.jar を追加
wrapper.classpath=<JDev>¥jswdk-1.0¥lib¥servlet.jar
```

```
# <JPDA のインストール・ディレクトリ>¥bin ディレクトリを環境変数 PATH に追加
wrapper.path=<JDev>¥jpda-1.0¥bin
```

JPDA の JVM を使用する場合

JPDA の Java Virtual Machine を使用する場合は、**jserv.properties** を次のように編集します。次の例は、Windows 環境の場合です。

```
# Apache JServ を起動する java.exe のパスを指定
wrapper.bin=<Java2 SDK のインストール・ディレクトリ>¥bin¥java.exe
# java.exe に渡すリモート・デバッグに必要なパラメータを指定
# 次は JDeveloper と JServ 間の通信のポート番号として 4000 を使用する場合
# 実際には、途中で改行が入らないことに注意
wrapper.bin.parameters=-classic -Xrunjdwp:transport=dt_socket,
server=y,suspend=n,address=4000 -Xdebug -Xnoagent -Djava.compiler=NONE
# <JDev>¥jswdk-1.0¥lib¥servlet.jar を任意のディレクトリにコピーし、
# クラス・パスに追加
# 次は、C:¥Tomcat¥JDev にコピーした場合
wrapper.classpath=C:¥Tomcat¥JDev¥servlet.jar
# <JPDA のインストール・ディレクトリ>¥bin ディレクトリを環境変数 PATH に追加
wrapper.path=<JPDA のインストール・ディレクトリ>¥bin
```

リモート・デバッグ

JDeveloper を使用して、サーブレットまたは JSP のリモート・デバッグを行う手順は次のとおりです。

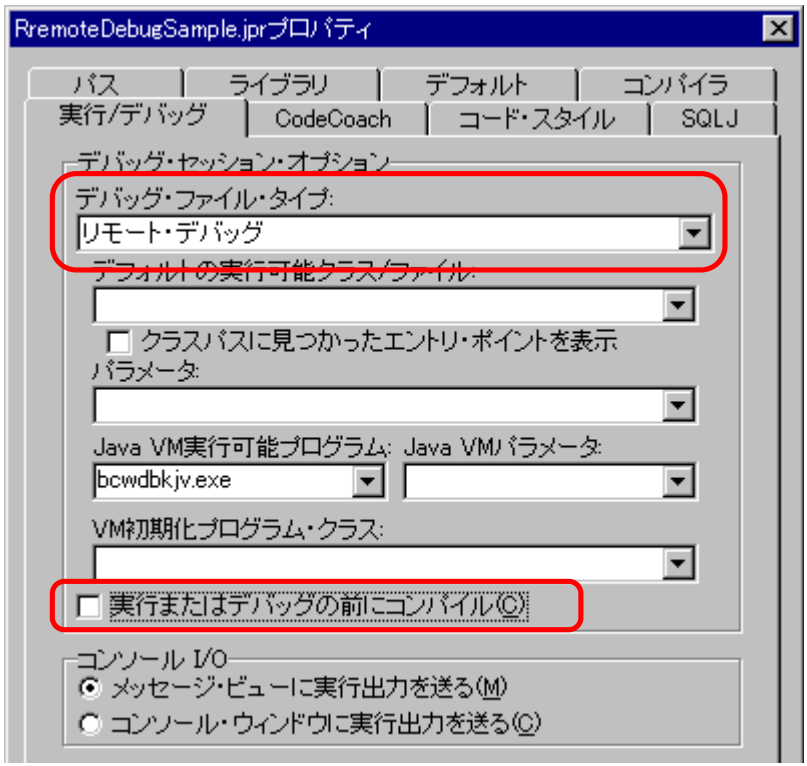
1. デバッグを行うサーブレットまたは JSP が含まれたプロジェクトを開きます。
2. 次のいずれかの方法で、プロジェクトのプロパティを変更するダイアログを起動します。
   a. ナビゲーション・ペインで、プロジェクト・ファイルをダブル・クリック

b. ナビゲーション・ペインで、プロジェクト・ファイルを右クリックし、ポップアップ・メニューから「プロパティ」を選択

c. プロジェクトがアクティブ（ナビゲーション・ペインでプロジェクト・ファイルが太字で表示）な状態で、メニュー・バーから「プロジェクト」→「プロジェクト・プロパティ」を選択

3. 「実行/デバッグ」ページを開き、「デバッグ・ファイル・タイプ」のドロップダウン・リストから「リモート・デバッグ」を選択し、「実行またはデバッグの前にコンパイル」のチェックを外します。

図 2 - 13　プロジェクトのプロパティを変更



4. 次のいずれかの方法で、サーブレットや JSP のソース・コードの適当な行にブレークポイントを設定します。

a. ソース・ビューアで、ブレークポイントを設定する行の左マージンをクリック

b. ブレークポイントを設定する行にカーソルを移動させ、ソース・ビューア上で右クリックし、ポップアップ・メニューから「ブレークポイントの設定」を選択

c. ブレークポイントを設定する行にカーソルを移動させ、メニュー・バーから「実行」→「ブレークポイントの追加」を選択

d. ブレークポイントを設定する行にカーソルを移動させ、メニュー・バーから「表示」→「ブレークポイント」を選択して、表示されたブレークポイント画面を右クリックし、ポップアップ・メニューから「ブレークポイントの追加」を選択

e. メニュー・バーから「実行」→「ブレークポイントの追加」を選択し、起動された「ブレークポイント・オプション」ダイアログにブレークポイントを設定する行の行番号を入力

図 2 - 14　ブレークポイントが設定された状態

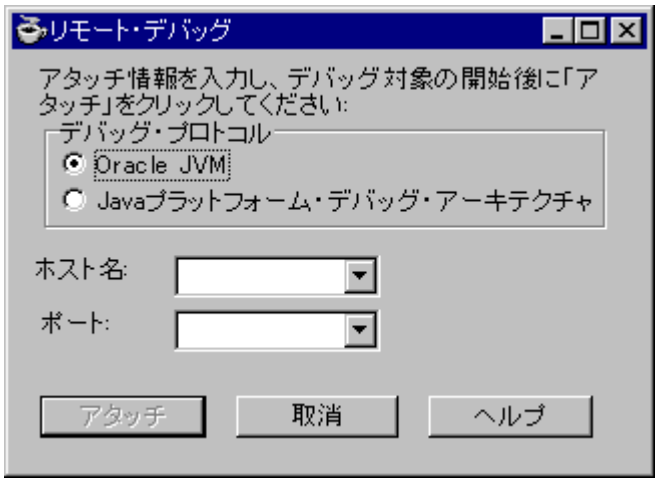
備考:　ブレークポイント・オプションの設定方法の詳細は、オンライン・ヘルプの次の項目を参照してください。

『ユーザー・ガイド』
→「JDeveloper の操作」
→「JDeveloper によるプログラムの開発」
→「Java プログラムのデバッグ」
→「ブレークポイントの使用方法」

5.  アイコンをクリックするか、メニュー・バーから「実行」→「デバッグ」を選択してデバッグを開始すると、「リモート・デバッグ」ダイアログが起動されます。次の項目の値を選択または入力して、「アタッチ」ボタンをクリックします。

設定項目	説明
デバッグ・プロトコル	使用する Java VM
ホスト名	アプリケーション・サーバーを実行しているホスト名
ポート	リモート・デバッグ時に使用するポート番号

図 2 - 15　「リモート・デバッグ」ダイアログ



6. デバッグ実行が開始された後、サーバー上のサーブレットまたは JSP に Web ブラウザからアクセスします。

7. 4 で設定したブレークポイントで、プログラムの実行が一時停止されます。ブレークポイントの設定の他、ステップ実行やトレース実行が可能です。

第 3 章　Tomcat のインストールと設定

インストール前の設定

Tomcat を使用するには、Java2 SDK が必要です。Java2 SDK をインストールする方法の詳細は、12ページの「第 2 章　Apache JServ のインストールと設定」、「インストール前の設定」を参照してください。

Tomcat のインストール

Tomcat をインストールする方法として、次の 2 種類が用意されています。

a. ソース・コードをダウンロードし、コンパイルする方法

b. バイナリ・ファイルをダウンロードし、展開する方法

この項では、バイナリ・ファイルを使用して Tomcat をインストールする方法について説明します。

バイナリ・ファイルを使用したインストール

1. Tomcat のバイナリ・ファイル **jakarta-tomcat** を次の URL からダウンロードします。

http://jakarta.apache.org/builds/tomcat/release/v3.1/bin/

圧縮方法は、*.zip、*.tar.Z、*.tar.gz の 3 種類が用意されているので、使用可能な解凍ツールに合わせて選択してください。

2. ダウンロードしたファイルを任意のディレクトリで展開すると、次のディレクトリが生成されます。

ディレクトリ名	説明
bin	Tomcat を起動/停止させるためのスクリプトを保存
conf	Tomcat を実行する際に必要な設定ファイルおよびそのサンプルを保存
doc	Readme、FAQ などのドキュメントを保存
lib	Tomcat を実行する際に必要な JAR ファイルを保存

ディレクトリ名	説明
logs	Tomcat の実行時のログ・ファイルを保存
src	Tomcat のソース・ファイルを保存
webapps	サンプル・アプリケーションを保存

備考: この他に **classes** ディレクトリを作成することが可能です。**classes** ディレクトリは、Tomcat 起動時に自動的にクラス・パスに含まれます。

設定ファイルの編集

前述したように、Tomcat を実行するために必要な設定ファイルは、***<Tomcat>*¥conf** ディレクトリに保存されています。このディレクトリに保存されているファイルは、更に次の 2 つに分類できます。

拡張子	説明
.xml、.dtd	*.xml ファイルで Tomcat そのものに関する設定情報を記述。*.dtd ファイルはその中で使用されているタグを定義
.conf、.properties	Tomcat を Apache など他の Web サーバーに組み込んで使用する際に必要な情報を記述

Tomcat を起動するためには、少なくとも使用するポート番号が他のアプリケーションですでに使用されていないかを確認する必要があります。Tomcat では HTTP サーバーが使用するポート（デフォルトは 8080）と、Tomcat のプロセス間が通信するためのポート（デフォルトは 8007）の 2 種類があります。

使用するポート番号を変更する場合は、**server.xml** の次の部分を編集します。

```
<!-- 52～55 行目：HTTP サーバーが使用するポート番号の設定 -->
<Connector className="org.apache.tomcat.service.SimpleTcpConnector">
  <Parameter name="handler"
    value="org.apache.tomcat.service.http.HttpConnectionHandler"/>
  <Parameter name="port" value="8080"/>
</Connector>

<!-- 57～60 行目：プロセス間の通信に使用するポート番号の設定 -->
```

```
<Connector className="org.apache.tomcat.service.SimpleTcpConnector">
  <Parameter name="handler"
    value="org.apache.tomcat.service.connector.Ajp12ConnectionHandler"/>
  <Parameter name="port" value="8007"/>
</Connector>
```

Tomcat の起動と停止

Tomcat を単体で起動/停止する際には、あらかじめ次の環境変数を設定しておく必要があります。

環境変数名	設定する値	例
TOMCAT_HOME	Tomcat がインストールされているディレクトリ	C:¥jakarta-tomcat
JAVA_HOME	JDK がインストールされているディレクトリ	C:¥jdk1.2.2
PATH	%JAVA_HOME%¥bin を含める	
TOMCAT_OPTS	SHIFT_JIS（UNIX/Linux のみ）	
LANG	設定しない（UNIX/Linux のみ）	

Tomcat を起動/停止するには、**<*Tomcat*>¥bin** に保存されている次のスクリプトを実行します。

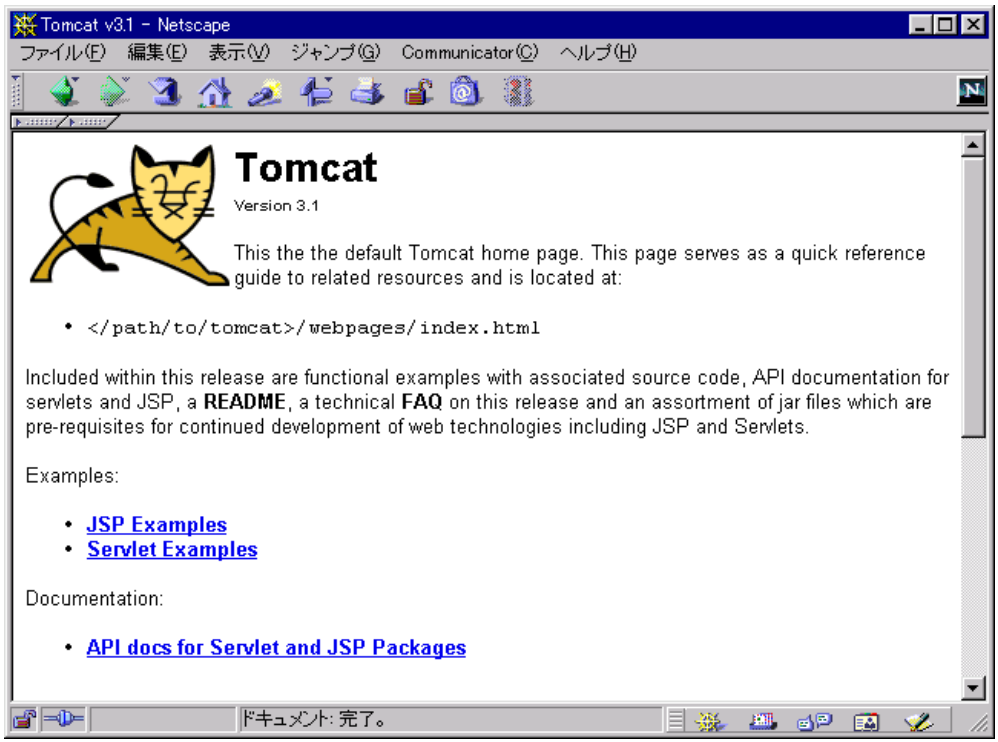
環境	起動用スクリプト	停止用スクリプト
Windows	startup.bat	shutdown.bat
Solaris	startup.sh	shutdown.sh

テスト

Tomcat の起動を確認するには、Web ブラウザから次の URL にアクセスします。

http://<*ServerName*>:<*Port*>/

図 3 - 1　Tomcat の起動確認



Tomcat と Apache の並行稼動

Tomcat は、他の Web サーバーに組み込んで使用することで、より多くのリクエストを効率よく処理することが可能になります。Tomcat 3.1 においてサポートされている Web サーバーは次のとおりです。

- Apache 1.3 以降
- Microsoft Internet Information Server 4.0 以降
- Microsoft Personal Web Server 4.0 以降
- Netscape Enterprise Server 3.0 以降

このうち、JDeveloper で作成したサーブレットおよび JSP の実行環境としてサポートされるのは、 Apache 1.3.9 以降のみです。

ここでは、Windows 環境において Tomcat を Apache に組み込む手順を説明します。その他 UNIX/Linux 環境での手順は、**<*Tomcat*>¥doc¥appdev¥index.html** の『Setting Tomcat to Cooperate with the Apache Web Server』を参照してください。

なお、次の手順は、Apache および Tomcat がそれぞれ単体で正常に動作することを前提としています。

1. 次の URL から、Apache JServ のモジュール ApacheModuleJserv.dll（64KB）をダウンロードします。

   http://jakarta.apache.org/builds/tomcat/release/v3.1/bin/win32/i386/

2. ダウンロードした **ApacheModuleJserv.dll** を**<*Apache*>¥modules** ディレクトリにコピーします。

3. Tomcat を実行すると、**<*Tomcat*>¥tomcat-apache.conf** ファイルが生成され、Tomcat と Apache を並行稼動させるために必要な設定情報が自動的に書き込まれます。これらの情報を Apache の起動時に読み込ませるために、**<*Apach*>¥conf¥http.conf** の最終行に次の 1 行を追加します。

   ```
   Include <Tomcat>¥tomcat-apache.conf
   ```

4. すべての設定が終了したら、Apache を起動し、その後 Tomcat を起動します。また、Apache を停止する場合は、Tomcat を停止してから、Apache を停止してください。

Tomcat への Web アプリケーションの追加

この項では、Tomcat にサーブレットや JSP などを追加するための基本的な方法について説明します。

Tomcat では、Servlet API 2.2 で導入された「Web アプリケーション」という概念が使用されています。Web アプリケーションは、サーブレットや JSP、HTML、イメージ・ファイルなど、アプリケーションを構成するファイルが、共通の仮想パスにマッピングされます。

例えば、作成したアプリケーションを、”/sample”という仮想パスにマッピングするためには、**<*Tomcat*>¥server.xml** に次のように記述します。

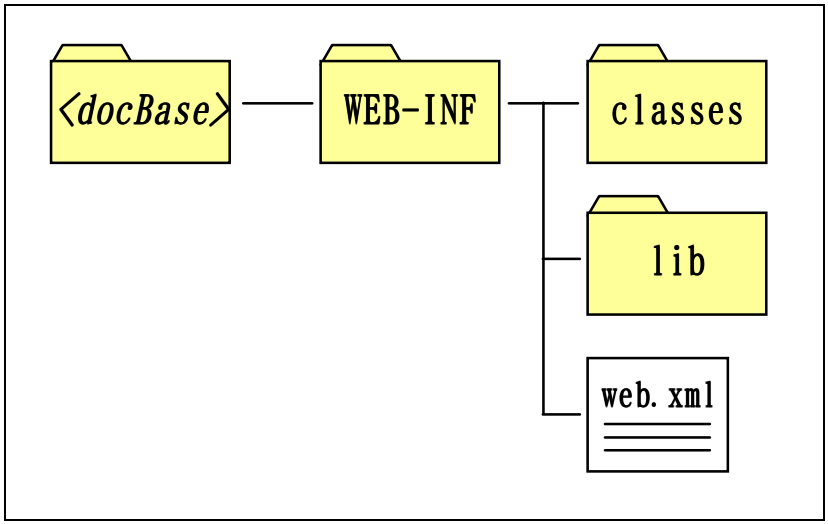
```
<ContextManager debug="0" workDir="works">
...
<Context path="/sample" docBase="sampleBase" debug="0" reloadable="true">
</Context>
</ContextManager>
```

この場合、**<*Tomcat*>¥sampleBase** ディレクトリ以下に、関連するファイルすべてが保存され、次の URL によりアクセスが可能になります。

http://<*ServerName*>:<*Port*>/sample/<*ファイル名*>

docBase で指定されたディレクトリには、任意のサブディレクトリを作成し、JSP や HTML、イメージ・ファイルを自由に保存することが可能です。ただし、*.class ファイルや*.jar ファイルは、WEB-INF という特別なディレクトリ以下に保存する必要があります。WEB-INF ディレクトリは次の図のような構造を持っています。

図 3 - 2　WEB-INF ディレクトリの構造



ディレクトリ/ファイル名	説明
classes	サーブレットや Web Bean などの*.class ファイルを保存
lib	JAR ファイルを保存
web.xml	サーブレットの名前、クラス名、初期化パラメータを定義

classes ディレクトリおよび lib ディレクトリ内の JAR ファイルは、自動的にアプリケーション実行時のクラス・パスに含まれます。また、JSP や HTML などのファイルを WEB-INF ディレクトリ以下に保存することはできません。

Oracle JSP Engine の設定

Tomcat にはサーブレットとして実装されている JSP エンジンがすでに用意されています。しかしこのエンジンでは、日本語の文字列をブラウザから入力した際に、自動的に正しい文字コードに変換することができません。一方、JDeveloper にバンドルされている Oracle JSP Engine は、初期化パラメータを設定するだけで、自動的に入力された文字列の文字コードを変換することが可能です。

Tomcat に Oracle JSP Engine を組み込む手順は次のとおりです。

1. Tomcat を実行するマシンの任意のディレクトリに、***<JDev>*¥lib¥ojsp.jar** をコピーします。
2. ***<Tomcat>*¥conf¥web.xml** を次のように編集します。

```
（29 行目まで省略）
<servlet>
  <servlet-name>
    jsp
  </servlet-name>
  <servlet-class>
    <!-- 次の 1 行をコメントアウト
    org.apache.jasper.runtime.JspServlet
    -->
    <!-- 次の 1 行を追加 -->
    oracle.jsp.JspServlet
  </servlet-class>
<!-- uncomment the following to use Jikes for JSP compilation
  <init-param>
    <param-name>jspCompilerPlugin</param-name>
    <param-value>org.apache.jasper.compiler.JikesJavaCompiler
    </param-value>
```

```
  </init-param>
-->
  <!-- 次の 4 行を追加 -->
  <init-param>
    <param-name>translate_params</param-name>
    <param-value>true</param-value>
  </init-param>
（以下省略）
```

3. Tomcat を起動するスクリプトを次の例のように編集します。

Windows の場合: ***<Tomcat>*¥bin¥tomcat.bat** の 27 行目を次のように編集します。

```
# 変更前: set CLASSPATH=.
# ojsp.jar を<Tomcat>¥lib にコピーした場合、次のように編集
set CLASSPATH=.;%TOMCAT_HOME%¥ojsp.jar
```

UNIX/Linux の場合: ***<Tomcat>*/bin/tomcat.sh** の 93 行目をを次のように編集します。

```
# 変更前: CLASSPATH=.
# ojsp.jar を<Tomcat>/lib にコピーした場合
CLASSPATH=.:<Tomcat>/lib/ojsp.jar
```

JSP アプリケーションの配布

作成したビジネス・コンポーネント JSP アプリケーションを実行するためには、次のものを Tomcat が実行されているマシンへ配布する必要があります。

- ビジネス・コンポーネントを定義する Java クラスおよび XML ファイル（ローカル・モードで実行する場合）
- ビジネス・コンポーネント JSP アプリケーション・ウィザードによって生成された*.jsp ファイルおよび*.properties ファイル
- 作成した WebBean
- ビジネス・コンポーネント JSP アプリケーションが実行する際にクラス・パスに追加する必要のある Java ライブラリ

- ビジネス･コンポーネント JSP アプリケーションが使用するイメージ・ファイルおよびカスケーディング・スタイル・シート

ビジネス・コンポーネントの配布

ビジネス･コンポーネント JSP アプリケーションをローカル・モードで実行する場合（ビジネス・コンポーネントが Oracle8*i* JServer 等に EJB や CORBA として配布されていない場合）は、次の手順でビジネス・ロジックを定義している Java クラスや XML ファイルを配布する必要があります。

手順の詳細は、20ページの「第 2 章　Apache JServ のインストールと設定」、「ビジネス・コンポーネントの配布」を参照してください。

配布プロファイル・ウィザードを使用した JSP アプリケーションの配布

JDeveloper と Apache が同一マシンにインストールされている場合や、Tomcat が稼働中のマシンに対してネットワーク・ドライブが設定されている場合は、配布ウィザードを使用して JSP アプリケーションを配布することができます。

手順の詳細は、22ページの「第 2 章　Apache JServ のインストールと設定」、「配布プロファイル・ウィザードを使用した JSP アプリケーションの配布」を参照してください。

手動による JSP アプリケーションの配布

Tomcat が他のプラットフォーム上で実行されている場合など、配布プロファイル・ウィザードが使用できない場合は、次のものを手動で配布する必要があります。

- *.jsp ファイルおよび*.properties ファイル
- ビジネス・コンポーネント JSP アプリケーションが使用する Java ライブラリおよび作成した WebBean
- イメージ・ファイルおよびカスケーディング・スタイル・シート

手順の詳細は、27ページの「第 2 章　Apache JServ のインストールと設定」、「手動による JSP アプリケーションの配布」を参照してください。

設定ファイルの編集例

Tomcat に Web アプリケーションを追加するための基本的な方法は、42ページの「Tomcat への Web アプリケーションの追加」で説明しました。ここでは、Tomcat にビジネス・コンポーネント JSP アプリケーションを追加する例を紹介します。

server.xml

```
<ContextManager debug="0" workDir="works">
...
<!-- JSP アプリケーションの仮想パス: /sample -->
<Context path="/sample" docBase="bc4j/sample" debug="0"
reloadable="true">
</Context>
<!-- JSP アプリケーションが使用するイメージ・ファイルの仮想パス: /webapp -->
<Context path="/webapp" docBase="bc4j/webapp" debug="0"
reloadable="true">
</Context>
</ContextManager>
```

ディレクトリ構造

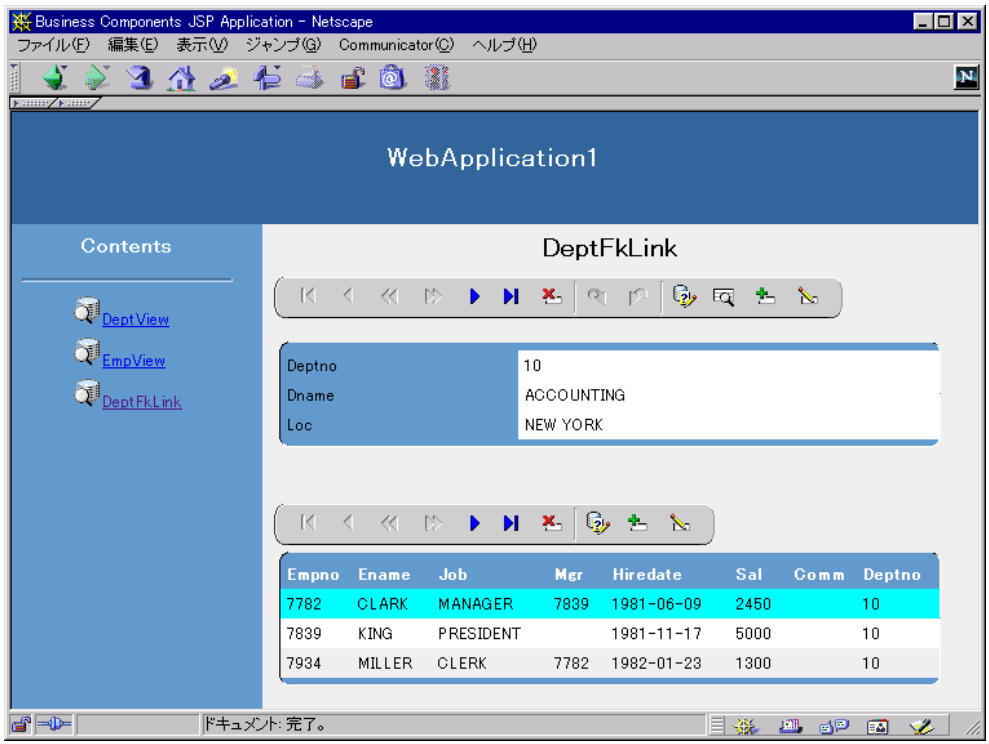
ディレクトリ名	説明
<*Tomcat*>¥bc4j¥sample	*.jsp/*.html/ViewObject.gif を保存
<*Tomcat*>¥bc4j¥sample¥WEB-INF¥lib	作成した BC4J や Java ライブラリを含む JAR ファイルを保存
<*Tomcat*>¥bc4j¥webapp	実行時に必要なイメージ・ファイルおよびカスケーディング・スタイル・シートをディレクトリ構造を保った状態で保存
<*Tomcat*>¥classes	<*ApplicationModule* 名>.properties を保存

実行

配布したビジネス･コンポーネント JSP アプリケーションを実行するには、Web ブラウザで次の URL にアクセスします。

http://<*ServerName*>:<*Port*>/sample/main.jsp

図 3 - 3　実行例



リモート・デバッグのための Tomcat の設定

31ページの「第 2 章　Apache JServ のインストールと設定」、「リモート・デバッグのための JServ の設定」で説明したように、JDeveloper 3.1 はサーブレットや JSP アプリケーション、Enterprise JavaBeans などのリモート・デバッグが可能です。

ここでは、Tomcat 上に配布されたサーブレットや JSP アプリケーションをリモートでデバッグするために必要な環境変数を紹介します。

なお、リモート・デバッグを行う際に必要な Java Platform Debugger Architecture（JPDA）の設定方法は、32ページの「第 2 章　Apache JServ のインストールと設定」、「Java Platform Debugger Architecture」を参照してください。リモート・デバッグの手順の詳細は、34ページの「第 2 章　Apache JServ のインストールと設定」、「リモート・デバッグ」を参照してください。

Oracle JVM を使用する場合

環境変数名	設定する値
PATH	<*JDev*>¥java1.2¥bin および<*JPDA のインストール・ディレクトリ*>¥bin を含める
JAVA_HOME	<*JDev*>¥java1.2
TOMCAT_HOME	Tomcat をインストールしたディレクトリ
TOMCAT_OPTS	-ojvm -Xxdebugondemandquietport<*デバッグに使用するポート番号*>
CLASSPATH	<*JDev*>¥jswdk-1.0¥lib¥servlet.jar を含める

JPDA の JVM を使用する場合

環境変数名	設定する値
PATH	<*Java2 SDK のインストール・ディレクトリ*>¥bin および<*JPDA のインストール・ディレクトリ*>¥bin を含める
JAVA_HOME	Java2 SDK のインストール・ディレクトリ
TOMCAT_HOME	Tomcat をインストールしたディレクトリ
TOMCAT_OPTS	Windows の場合: -classic -Xrunjdwp:transport=dt_socket,server=y,suspend=n, address=<*デバッグに使用するポート番号*> -Xdebug -Xnoagent -Djava.compiler=NONE Solaris の場合: -classic -Xrunjdwp:transport=dt_socket,server=y,suspend=n, address=<*デバッグに使用するポート番号*>,stdall=y -Xdebug -Xnoagent -Djava.compiler=NONE -Dfile.encoding=SHIFT_JIS

環境変数名	設定する値
CLASSPATH	<*JDev*>¥jswdk-1.0¥lib¥servlet.jar を任意のディレクトリにコピーし、含める
LD_LIBRARY_PATH	<*JPDA のインストール・ディレクトリ*>¥bin を含める（Solaris の場合のみ）