BEA WebLogic Server

WebLogic ZAC
ユーザーズ ガイド
（非推奨）

WebLogic Server バージョン 6.1
マニュアル第 1.0 版
2001 年 11 月 30 日

著作権

限定的権利条項

商標または登録商標

WebLogic ZAC ユーザーズ ガイド

マニュアルの版数	日付	ソフトウェアのバージョン
6.1	2001 年 11 月 30 日	BEA WebLogic Server 6.1

目次

1 WebLogic ZAC を使用したパブリッシュ

この節では、WebLogic Zero Administration Client（ZAC）の使い方の概要を説明します。内容は以下のとおりです。

はじめに

注意： WebLogic Zero Administration Client は、非推奨の製品です。BEA では、Sun Microsystems の Java Web Start 製品を使用することをお勧めしています。Java Web Start は Java 2 に準拠した製品で、これを使用すると、ユーザは Java アプリケーションをダウンロードできるようになります。

WebLogic ZAC（Zero Administration Client ユーティリティ）を使用すると、アプリケーション、アプレット、ライブラリをパブリッシュ / 再パブリッシュできるので、それらをエンド ユーザのクライアント マシン上で最新バージョンに透過的かつ自動的に更新することができます。そのため、アプリケーション、アプレット、またはライブラリを手動でクライアントに配布する必要がなくなり、配布作業を ZAC の自動サービスに任せることができます。

ZAC は非常に効率的です。ZAC アプリケーションを再パブリッシュすると、必要最小限のデータだけがネットワーク経由で各クライアントに転送され、クライアント上のアプリケーションが最新のバージョンに更新されます。変更がごく一部、またはまったくないような一般的な場合には、起動時に新しいファイルをチェックするためのオーバーヘッドはユーザが意識しないほど小さなものです。

ZAC では詳細なコンフィグレーションが可能なので、アプリケーションのパブリッシュ、インストール、および更新の方法を設計することができます。アプリケーションが更新されていないかどうかを、アプリケーションを起動または停止するたびにチェックしたり、スケジュールに基づいてチェックしたりすることができます。また、ZAC の更新チェックを無効にすることもできます。これは ZAC の最も便利な機能を無効にすることになりますが、パッケージを一括して配布する場合には適しています。

ZAC は、1997 年 8 月に W3C に提出された仕様、HTTP Distribution and Replication Protocol（DRP）というプロトコルを使用して、データを効率的にレプリケートします。

このマニュアルでは、ZAC Publish Wizard を使用して、ユーザに配布するためのアプリケーションを WebLogic Server 上でパブリッシュする方法について説明します。

ZAC デモの実行

インストール シールド（.exe）バージョンを使用して Windows 上にインストールした場合、WebLogic には、ZSimple と ZUpdate という準備なしで使える 2 つの ZAC パッケージが付属し、WebLogic ZAC がどのように動作するかをすぐに試してみることができます（このプリインストールされたデモは、WebLogic Server を .zip ファイルの配布キットからインストールした場合には正しく動作しません）。

1. WebLogic を起動します（パブリッシュするにはシステム パスワードが必要です）。
2. ZAC Publish Wizard を起動します。Win32 ユーザは、[スタート] メニューの WebLogic ディレクトリにある ZAC Publisher のショートカットを使用します。

   Windows ユーザ以外は、（次のコマンドを使用して CLASSPATH を設定した後に）コマンド ラインから Publish Wizard を起動します。

   ```
   $ java weblogic.PublishWizard
   ```

3. [Publish Wizard] ウィンドウで、ZSimple または ZUpdate を選択してダブルクリックします。順を追ってすべての手順を参照すると、パッケージをパブリッシュする方法がわかります。また、[Finish] ボタンをクリックすると、概要部分をスキップできます。
4. [Done] ボタンをクリックします。
5. アプリケーションをパブリッシュするには、リストで選択してから [Package] メニューの [Publish...] を選択します。適切なホストを選択して、[Publish] ボタンをクリックします。1 つのホストだけに接続している場合、そのホストが自動的に選択されます。[Publish...] ダイアログにパブリッシュ処理のステータスが表示され、正常に処理が完了すると、パッケージが Publish Wizard のウィンドウの左下の、選択した WebLogic Server ホストの下に表示されます。「+」記号を展開しないと、パブリッシュされたパッケージが見えない場合もあります。
6. パブリッシュされたパッケージをテストするには、パブリッシュ ウィンドウの左下で WebLogic Server ホストの下のパッケージを選択してから、[Server] メニューの [Test run] を選択します。

テストが始まった後も、パッケージを開いてパラメータを変更したり、アプリケーションを再パブリッシュしてクライアントの応答を確認したりすることができます。

ここからは、パッケージの作成およびパブリッシュする方法と、ブートストラップ実行可能ファイルの作成方法について説明します。ブートストラップ実行可能ファイルは、ZAC パッケージからアプリケーションをインストールおよび実行する標準的な方法です。

ZAC の動作

どのような Java アプリケーション、アプレット、またはライブラリも、ZAC パッケージとしてパブリッシュできます。ZAC を使用すると、Java ソース コードに何も追加することなく、プログラムをパブリッシュできます（ただし、ZAC には Java API が含まれており、ZAC 更新を行うタイミングやアプリケーションが ZAC 更新にすぐに応答するかどうかを対話方式で制御するコードをアプリケーションに追加する場合に使用できます。ZAC API を使用した開発については、「WebLogic ZAC を使用した開発」で説明しています）。

ZAC の仕組みは、ユーザにとって非常に簡単です。

1. 開発者が ZAC Publish Wizard を使用して、アプリケーションを WebLogic Server にパブリッシュします。
2. アプリケーションのユーザが小さなネイティブ OS インストーラをダウンロードして、ダブルクリックするだけでアプリケーションをインストールおよび起動できます。
3. インストーラは、ユーザのマシン上でアプリケーションを作成し、必要なライブラリをインストールするだけでなく、新しくパブリッシュされたバージョンのアプリケーションをモニタし、更新処理を実行する小さなブートストラップもインストールします。

サーバ上で ZAC パッケージをパブリッシュする方法

ZAC Publish Wizard を使用すると、アプリケーションを簡単にパブリッシュできます。ZAC Publish Wizard では、以下の処理を行うことができます。

- 新規 ZAC パッケージを作成します。
- ZAC パッケージを WebLogic Server 上でパブリッシュします。
- ZAC パッケージを他の WebLogic Server からインポートします。
- WebLogic Server 上でパブリッシュ済みの ZAC パッケージを更新します。

ZAC Publish Wizard の手順に従うと、ZAC を使用するパッケージを作成し、WebLogic Server 上でパブリッシュできます。パブリッシュ処理では、アプリケーションまたは ZAC パッケージをクライアント マシン上で実行するために必要な以下のパラメータを設定します。

- アプリケーションを構成するファイルが入っているディレクトリ
- アプリケーションが依存する他の ZAC パッケージ
- クライアント マシン上に必須の Java 実行環境
- アプリケーションが正しく動作するために必要なその他の情報

アプリケーションが WebLogic Server にパブリッシュされると、ユーザは HTTP 経由でそのアプリケーションをダウンロードできるようになります。アプリケーションまたはライブラリを変更するたびに、パッケージを再パブリッシュできます。

ZAC がパブリッシュされているアプリケーションをユーザのマシンにインストールする方法

パブリッシュ処理が完了すると、アプリケーションが WebLogic Server 上でパブリッシュされます。ZAC Publish Wizard は、ZAC がサポートしている各マシンのネイティブ形式で小さなインストール プログラムを作成します。ユーザがこのプログラムをダウンロードして実行すると、パブリッシュされているアプリケーションまたはパッケージがインストールされます。ここからは、このインストール プログラムを「インストーラ」と呼びます。インストーラはごく小さな実行可能ファイルなので、短時間でダウンロードできます。

パブリッシュされたアプリケーションをユーザが使用できるようにするには、インストーラを電子メールに添付するか、インストーラがある FTP へのリンクを HTML ページに埋め込みます。ユーザはインストーラをダウンロードして実行します。インストール プログラムはネイティブ形式の実行可能ファイルなので、ユーザは Java 実行環境（JRE または JDK のような Java 開発環境）をプリインストールしておく必要がありません。

インストーラは、ZAC Publish Wizard でのコンフィグレーション内容によって、以下の作業の一部またはすべてを実行します。

- （省略可能）パブリッシュされているアプリケーションが必要とする特定の JRE がクライアント マシンにプリインストールされているかどうかをチェックします。ZAC インストーラは、別の ZAC パッケージとして用意されている JRE を必要に応じて自動的にインストールできます。
- （省略可能）パブリッシュ側 WebLogic Server に接続するための HTTP プロキシ情報をユーザに問い合わせます。
- （必ず実行）ZAC アプリケーションをダウンロードしてインストールします。
- (省略可能）パブリッシュされているアプリケーションが必要とする他の ZAC パッケージをダウンロードしてインストールします。
- （必ず実行）ブートストラップ実行可能ファイルをインストールします。このファイルは、パブリッシュされているアプリケーションを起動し、新しいバージョンがパブリッシュされていないか WebLogic Server をモニタし、必要に応じて新しいバージョンをダウンロードします。
- （省略可能）ブートストラップにリンクしたデスクトップ アイコンまたは [スタート] メニュー項目を作成します。

■ （省略可能）ZAC アプリケーションを起動します。

初期インストールが完了したら、インストーラをクライアント マシンから削除してもかまいません。

パブリッシュされているアプリケーションがユーザのマシン上で動作する仕組み

インストーラはアプリケーションだけでなく、ネイティブ形式のブートストラップ ファイルもインストールします。アプリケーションがインストールされると、ユーザはネイティブ OS ブートスラップを使用してアプリケーションを起動します。ZAC Publish Wizard のプロセスでは、ブートストラップ プログラムも作成します。

クライアントがブートストラップを実行するたびに、ブートストラップはまず、WebLogic Server 上に新しいバージョンのアプリケーションがパブリッシュされていないかチェックします。新しいバージョンのアプリケーションまたは ZAC アプリケーションが必要とする他の ZAC パッケージがパブリッシュされていた場合、ブートストラップは、クライアント マシン上の ZAC パッケージを新しいバージョンに自動的に更新します。次に、ブートストラップは、新しいバージョンのアプリケーションを起動します。

ZAC によるパブリッシュに対応するための WebLogic の設定

ユーザが ZAC にアクセスできるようにするには、ZAC を Web アプリケーションとして WebLogic Server にデプロイしておく必要があります。ZAC Web アプリケーションに含まれているサーブレットがクライアントからのリクエストを処理します。ZAC は、WAR ファイル、`zac.war` としてサーバ上にデプロイされます。デプロイ用に ZAC パッケージを準備するには、以下の手順を実行します。

1. ファイル `web.xml`（ZAC Web アプリケーションの一部）を編集して、パブリッシュされたパッケージを見つけるためのパブリッシュ ルートを設定します。パブリッシュ ルートを相対パスに設定した場合、ディレクトリは WebLogic ホーム ディレクトリが基準となります（`myserver\` ディレクトリ

と同じレベル)。未設定の場合、パブリッシュ ルートは、デフォルトで WebLogic ホームの exports\ ディレクトリに設定されます。次の例は、パブリッシュ ルートを設定する web.xml のエントリです。

```
<context-param>
 <param-name>weblogic.zac.publishRoot</param-name>
 <param-value>C:/weblogic/publish</param-value>
</context-param>
```

2. 必要に応じて、パブリッシュするパッケージに ACL を設定して、アクセスを制限します。未設定の場合、書き込みパーミッションが system に、読み込みパーミッションが everyone にデフォルト設定されます。アクセス制御リストを設定する場合も、zac.war アプリケーションに含まれる web.xml ファイルを編集します。次の例は、myApp へのアクセスを Peter、Paul、Mary の 3 ユーザに制限する ACL を設定する web.xml ファイルのエントリの例です。

```
<context-param>
<param-name>weblogic.allow.read.weblogic.zac.myApp</param-name>
 <param-value>Peter,Paul,Mary</param-value>
</context-param>
```

ZAC Web アプリケーションをデプロイするには、次の手順を実行します。

1. WebLogic Server を起動します。ZAC パッケージをパブリッシュする場合には、WebLogic Server が稼働している必要があります。ただし、ZAC パッケージを作成する場合には稼働している必要がありません。
2. WebLogic Administration Console を起動します。
3. パブリッシュされたパッケージを使用可能にするには、ZAC Web アプリケーションのアーカイブ ファイル zac.war を Web アプリケーションとしてデプロイします。Web アプリケーションのデプロイの詳細については、『Web アプリケーションのアセンブルとコンフィグレーション』を参照してください。

2 Publish Wizard の使い方

この節では、Publish Wizard について説明します。内容は以下のとおりです。

Publish Wizard の起動

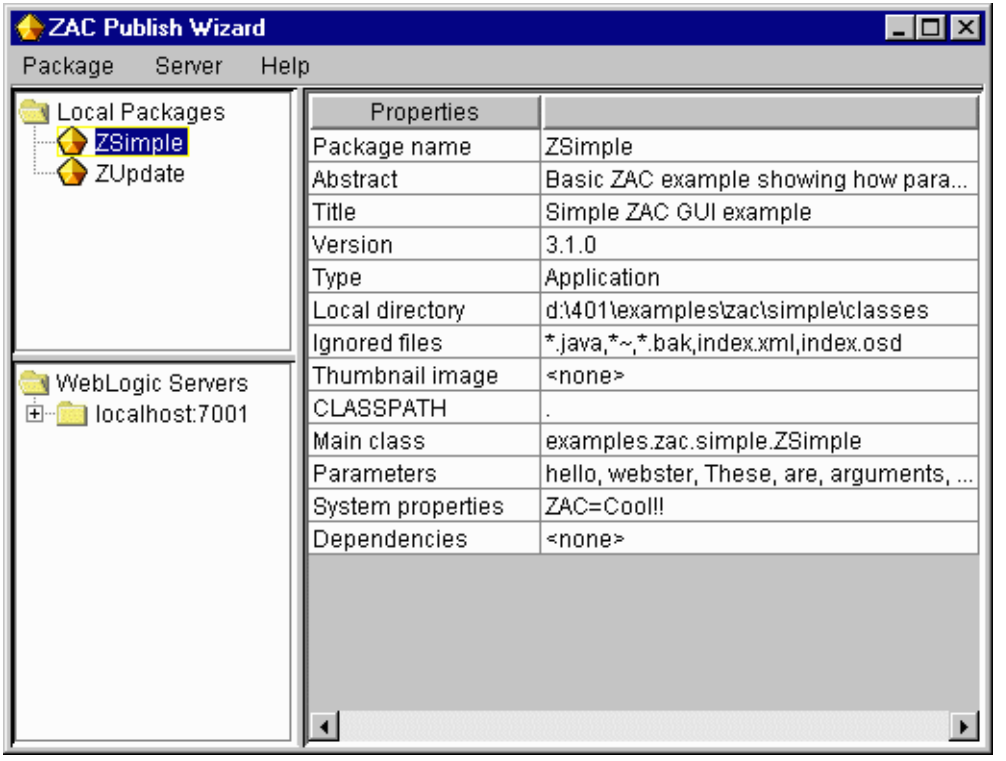
WebLogic Server にパブリッシュする場合、そのサーバが実行中でなければならず、パブリッシュするためのパーミッションを持つユーザとパスワードも必要になります。詳細については、「ZAC パッケージのパブリッシュ」を参照してください。ただし、ZAC パッケージを作成したり、既存の ZAC パッケージを調べたりする場合には、サーバが実行中である必要はありません。

Publish Wizard は、（次のコマンドを使用して CLASSPATH を設定した後に）コマンド ラインから起動します。

```
$ java weblogic.drp.admin.PublishWizard
```

ZAC の起動中に ZAC Publish Wizard のスプラッシュ画面が表示されてから、メイン ダイアログボックスが表示されます。

図 2-1　ZAC Publish Wizard のメイン ウィンドウ



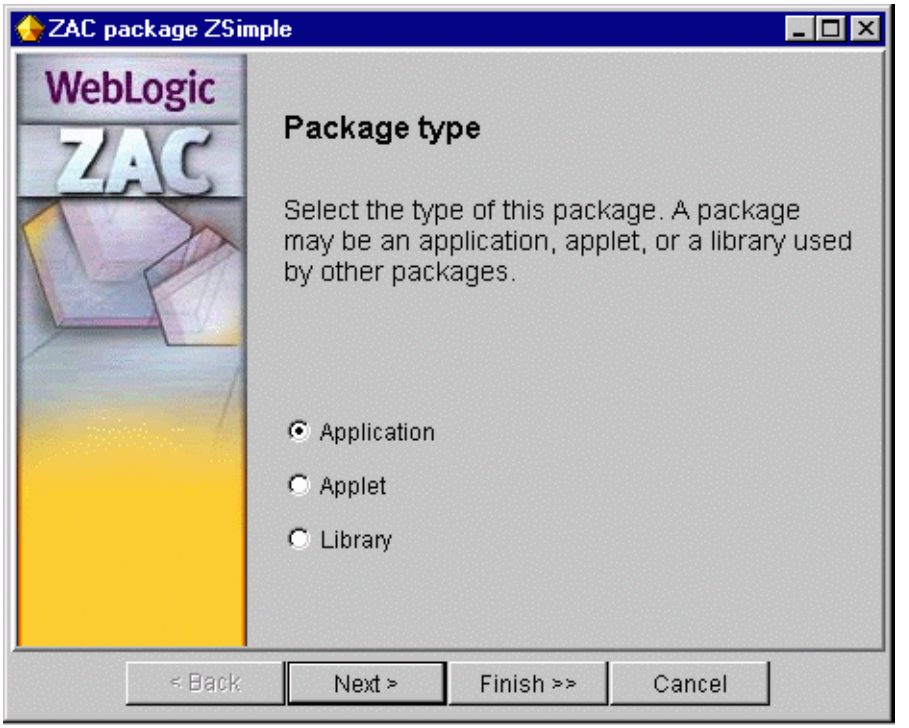
ZAC パッケージの作成

ZAC Publish Wizard を使用すると、新しい ZAC パッケージを作成したり、既存のパッケージを他のサーバからインポートしたり、以前にパブリッシュしたパッケージを更新したりすることができます。ZAC パッケージは、アプリケーション、アプレット、またはライブラリです。

Publish Wizard の指示に従うと、新しいパッケージを作成したり、既存のパッケージを再パブリッシュしたりすることができます。また、ZAC の [test run] オプションを使用すると、ZAC アプリケーションをテスト実行して、コンフィグレーションしたとおりに動作するかどうかをチェックできます。

1. [Package] メニューから、[New] を選択して新しいパッケージを作成するか、[Open] を選択して既存のパッケージを参照または編集します。既存のパッケージをダブルクリックしても、[Package] パネルが表示されます。
2. [Package] パネルで、パッケージ タイプを選択します。パブリッシュできるのは、アプレット、Java アプレット、またはクラス ライブラリです。一般に、ライブラリは、パブリッシュされる他のアプリケーションが依存するコンポーネントとしてパブリッシュします。

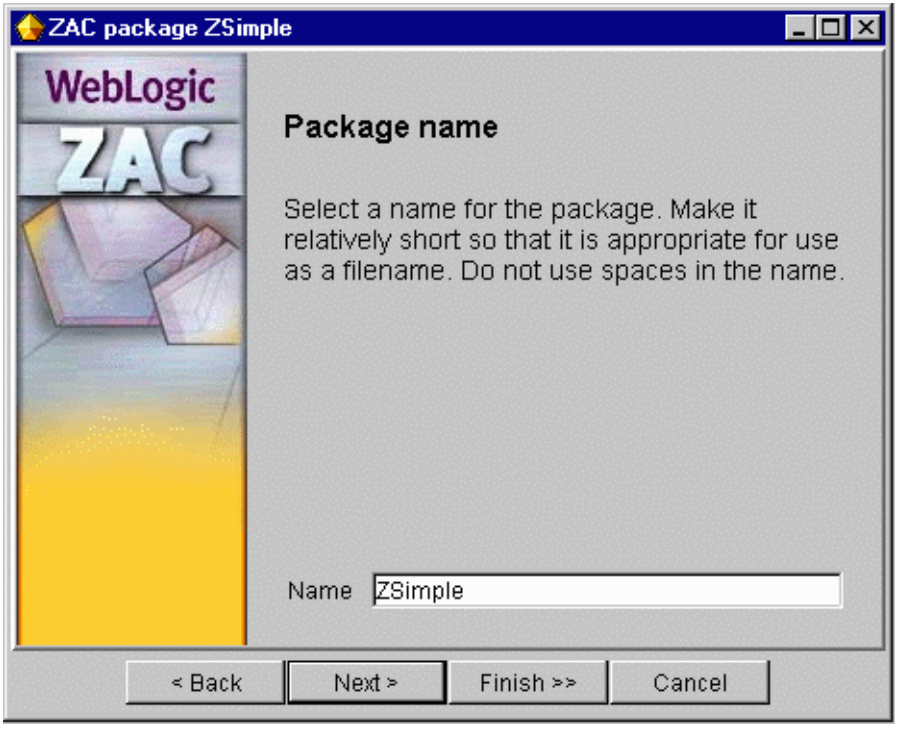
図 2-2 新規ライブラリ パッケージのパブリッシュ



3. パッケージに名前を付けます。これは表示用の名前ではなく、ファイル名として使用するだけなので、短い名前で簡単に識別できるものにします。

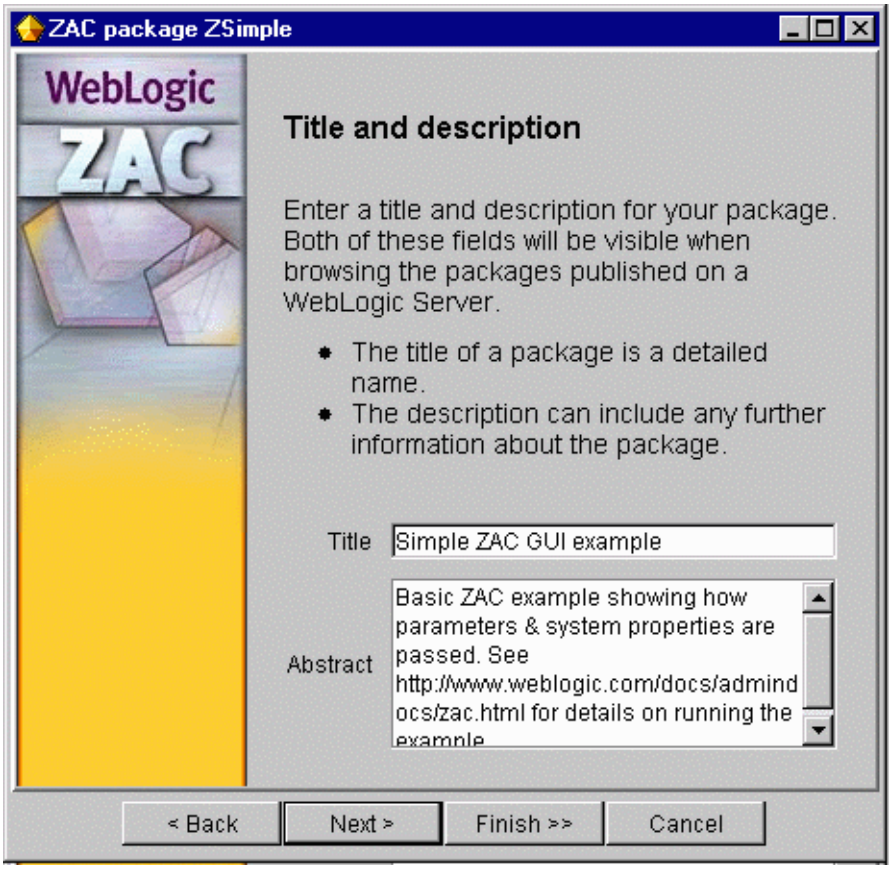
注意： パッケージ名にはスペースを使用できません。

図 2-3 パッケージに名前を付ける



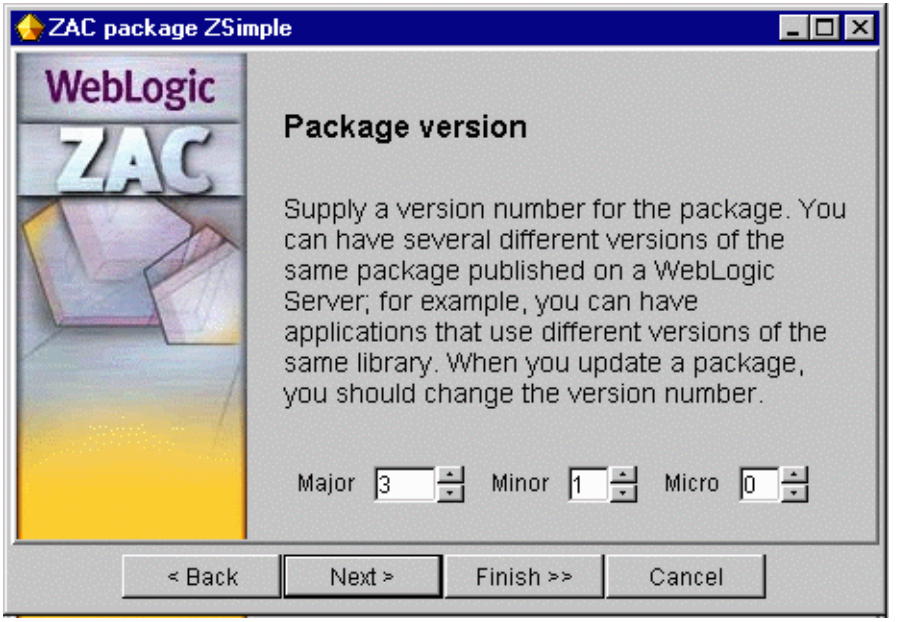
4. パッケージのタイトルと短い説明を入力します。これらはどちらも表示用として使用されます。

図 2-4　タイトルと説明の入力



5. バージョン番号を設定します。

図 2-5 バージョン番号の設定



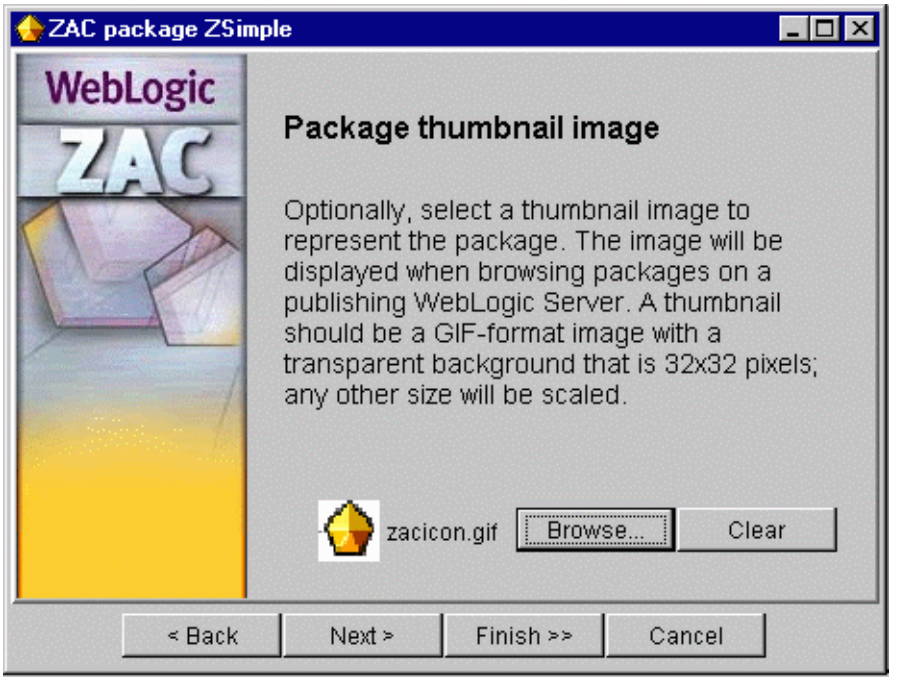
6. ZAC パッケージのファイルが入っているディレクトリを参照します。選択したディレクトリ内のすべてに加えて、そのサブディレクトリのすべての内容がパブリッシュされます。

   パブリッシュされるパッケージからファイルを除外するには、無視するファイル タイプのリストに追加します。このリストを使用すると、同じディレクトリ内のファイルのうち、パッケージの作成に使用して、パブリッシュの対象から除外するものを指定できます。各クライアントに固有のファイル index.xml と index.osd は無視するのが一般的です。パブリッシュするパッケージから除外するファイルのリストを指定した後も、リストにファイルの追加や削除を行うことができます。

図 2-6　パッケージの最上位ディレクトリの選択

7. パッケージのサムネイルとして使用する GIF 画像を見つけます。画像は、32x32 ピクセルで背景が透明のものが最適です。

図 2-7　サムネイル画像の指定



8. アプレットをパブリッシュする場合は、アプレットのメイン クラス、CODEBASE、および PARAM タグで任意にリストされるアプレットのパラメータ リストを指定するよう要求されます。アプレットがオンラインの場合は、ZAC がアプレットを見つけてパラメータ リストを自動的に入力するための URL を入力してもかまいません。アプレットを識別するには、以下の情報を指定します。

   [main classname]

   アプレットのオンラインでの場所を示す URL を（ドキュメント ベースと一緒に）指定した場合、ZAC はアプレットの場所を見つけて、パラメータ リストを自動的に入力します。

   [document base]

   アプレットのホストとなる Web サーバの URL です。ドキュメント ベースとクラス名を指定するだけで、ZAC はオンラインで実行するアプレットを見つけて、パラメータ リストを自動的にロードします。

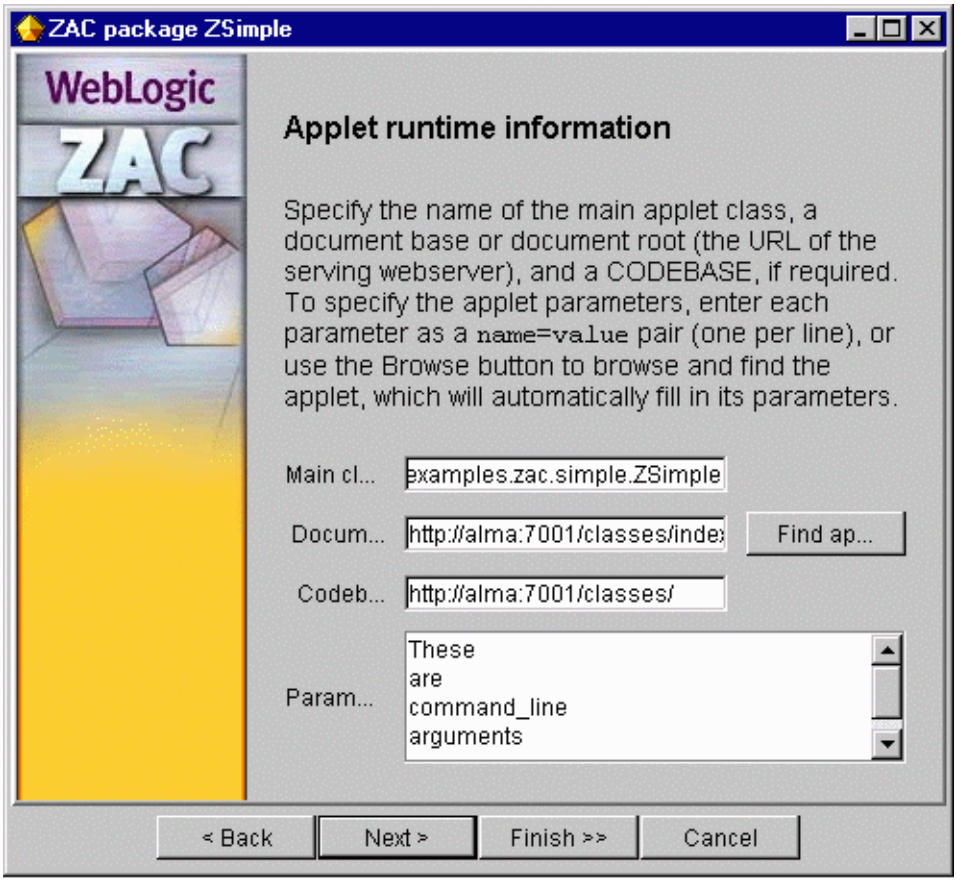
[CODEBASE attribute]

アプレットが必要に応じてさらにクラスを取得する場所を表す URL です。一般的には、`http://yourserver:port/classes`（`yourserver` は WebLogic Server を表し、ClasspathServlet を仮想名 `classes` に対して登録）に設定します。詳細については、『管理者ガイド』の「サーブレットのコンフィグレーション」を参照してください。

[applets parameters]

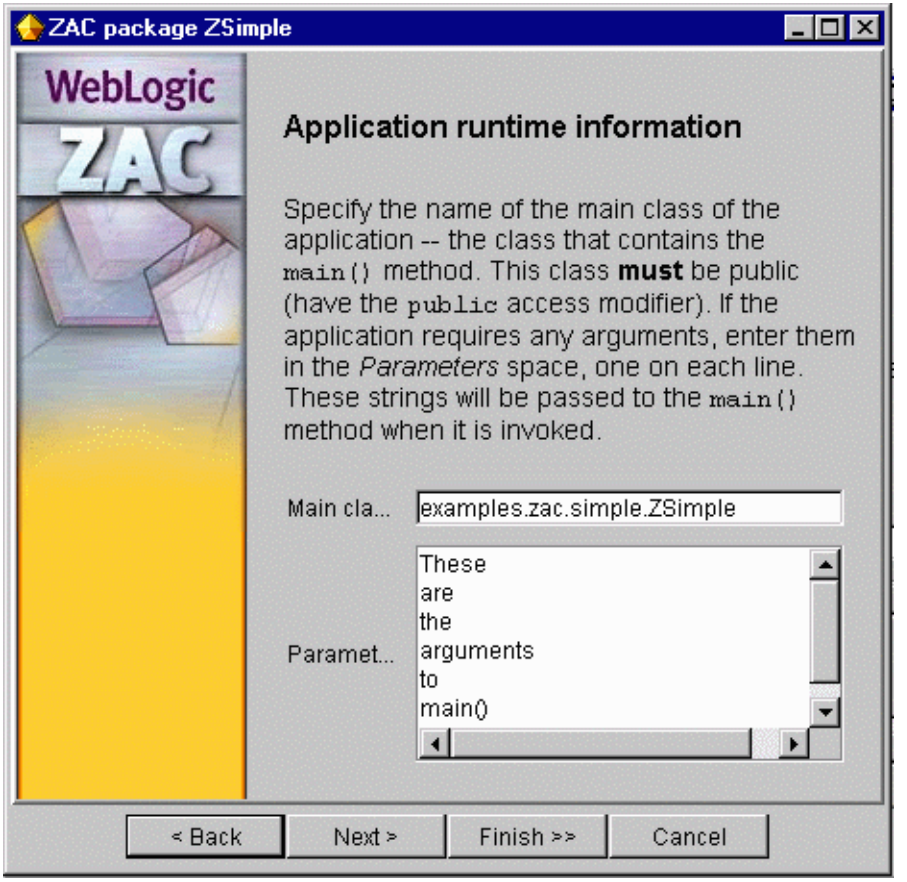
アプレットのパラメータは、初期化変数と実行時変数をアプレットに提供する name=value のペアからなるリストです。これらの変数は、HTML ファイルの <APPLET> タグで囲んで提供される変数と同様です。

図 2-8　アプレットのパラメータの指定



9. パブリッシュするパッケージのクラスのパスを入力します。アプリケーションをパブリッシュする場合は、アプリケーションを起動する `main()` メソッドが入った Java クラスの絶対パスを入力するよう要求されます。また、`main()` メソッドに必要な初期化引数もすべて、[Parameters] テキスト領域に入力します。

図 2-9　アプリケーション起動用のクラスの指定



パッケージの CLASSPATH を指定します（必須）。パブリッシュするパッケージ自体に必要なクラスがすべて入っている場合は、カレント ディレクトリにドット「.」を指定するだけです。CLASSPATH は、必ずアプリケーションの最上位ディレクトリを起点とします。アプリケーションをパブリッシュする場合は、アプリケーションを実行する `main()` メソッドが格納するクラスがパッケージの CLASSPATH に入っている必要があります。

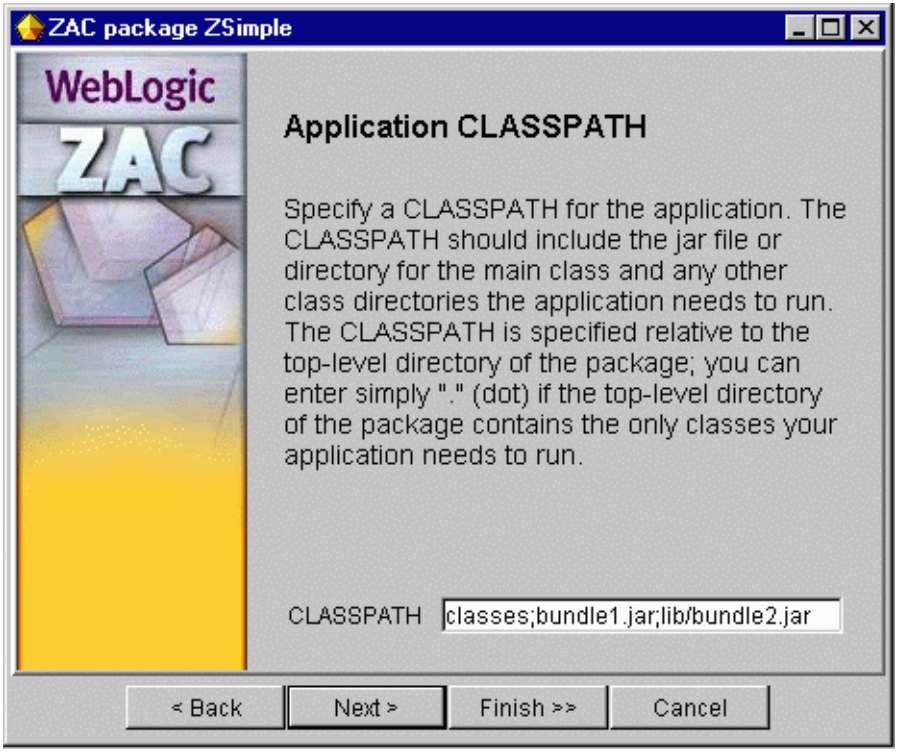
どのタイプのパッケージをパブリッシュする場合も、CLASSPATH のエントリは、必ず、パブリッシュするローカル ディレクトリの最上位を起点とします。たとえば、以下の内容のローカル ディレクトリ `c:\myapps`

```
c:\myapps\classes\foo\Main.class
c:\myapps\bundle1.jar
c:\myapps\lib\bundle2.jar
```

（foo.Main はパッケージ foo に格納されています）をパブリッシュする場合、パッケージの CLASSPATH を以下のように設定します。

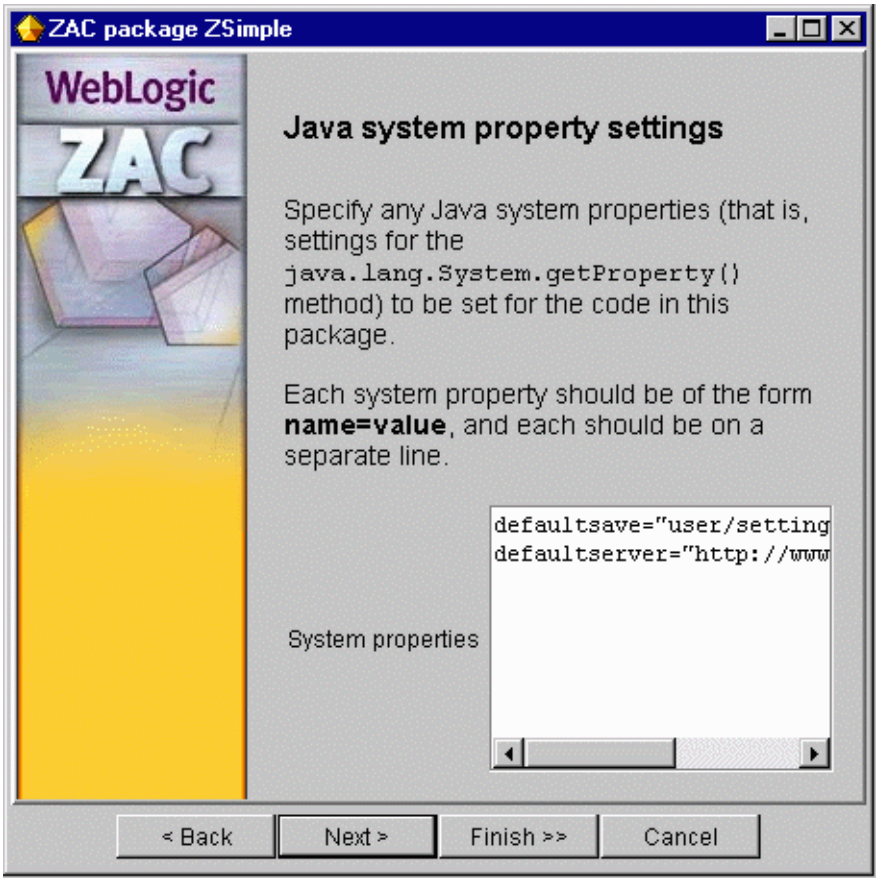
```
classes;bundle1.jar;lib\bundle2.jar
```

図 2-10　パッケージの CLASSPATH の指定



10. 次に、アプリケーションに必要な Java システム プロパティを指定します。1 行につき 1 つの **name=value** ペアを指定します。

図 2-11　Java システム プロパティの指定



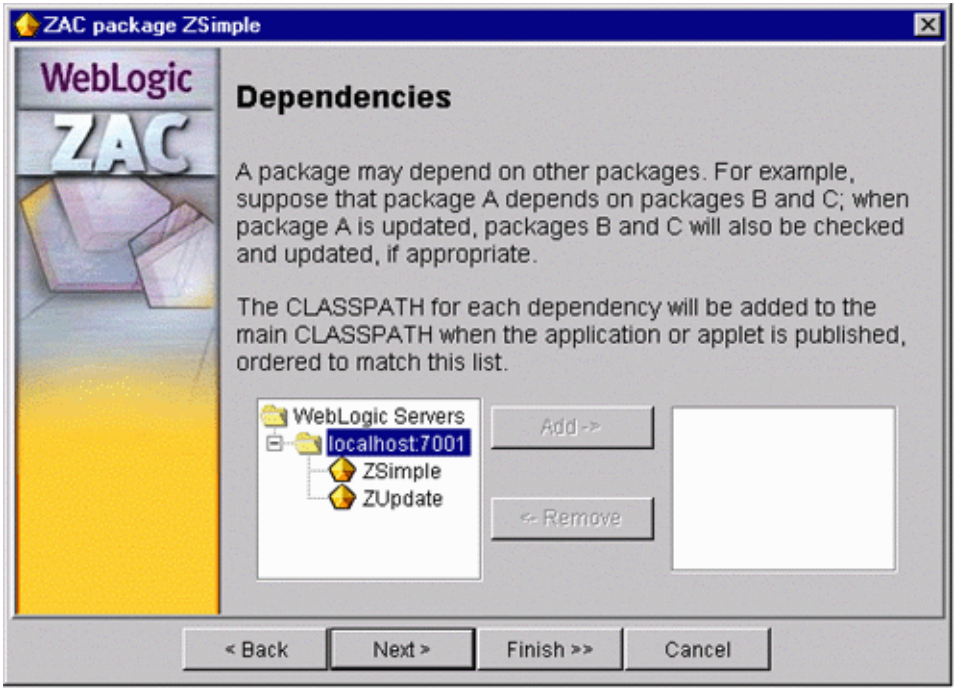
11. 次に、パッケージと他のパッケージの依存関係を設定します。ZAC を使用すると、これらのパッケージもクライアント マシンにインストールされ、常に最新の状態に保たれます。パッケージが依存する共有ライブラリまたは他のアプリケーションの ZAC パッケージを作成し、パッケージ名を使って依存関係を指定する必要があります。

    ZAC で依存関係を使用すると、別々のアプリケーションが共通のコードをクライアント マシン上で共有できます。共通のコードが更新されると、依存関係を持つすべての ZAC アプリケーションも更新されます。

    たとえば、WebLogic と、ライブラリとして Swing クラスに依存するアプリケーションを開発した場合、各アプリケーションの依存関係としてこれらを

リストできるので、クライアント マシンで必要なライブラリのコピーは 1 つだけです。

図 2-12　依存関係の設定



左側のウィンドウでパブリッシュされているパッケージを WebLogic Server から見つけて、[Add] をクリックします。依存するパッケージの詳細が右側のウィンドウに表示されます。依存するパッケージを削除するには、右側のウィンドウで選択してから [Remove] をクリックします。

ZAC パッケージのパブリッシュ

新しいパッケージを作成するか、すでにパブリッシュしたパッケージの詳細を更新したら、パッケージを WebLogic Server にパブリッシュできるようになります。

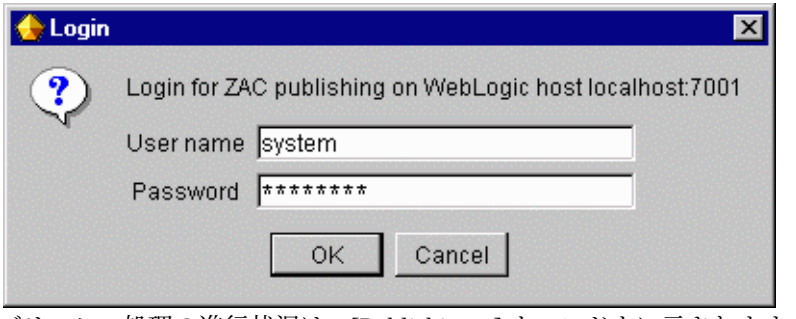
1. パブリッシュするパッケージを、ZAC Publish Wizard のメイン ウィンドウで選択します。[Package] メニューで **[Publish]** を選択します（有効になっている場合）。[WebLogic Servers] パネルに複数の WebLogic Server がリストされている場合、2 つ目のオプション **[Publish to]** が有効になります。このオプションを使用すると、どの WebLogic Server に対してパブリッシュするかを選択できます。

   [Publish] を選択した場合、WebLogic Server が自動的に選択状態になります。このサーバは、リストされている唯一のサーバ、またはパブリッシュ先または復帰の対象として前に選択したサーバです。

   [Publish to] を選択した場合、このパッケージをパブリッシュする WebLogic Server のアドレス / ポートを選択する必要があります。

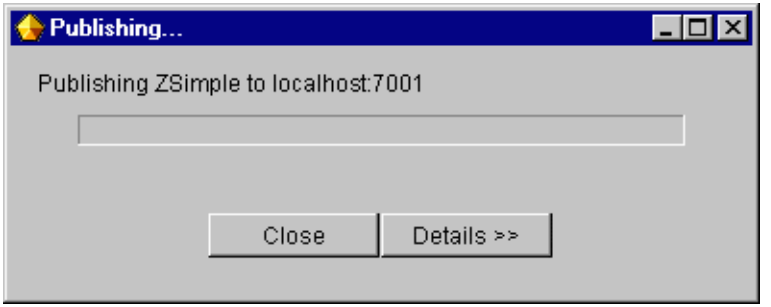
2. WebLogic Server に最初にパブリッシュするときと、ZAC を再起動した後にパブリッシュするたびに、名前とパスワードを入力するよう要求されます。パブリッシュするには、`weblogic.properties` ファイルの ACL `weblogic.zac` で「書き込み」パーミッションを持つ T3User のユーザ名およびパスワードを入力しなければなりません。未設定の場合、この ACL では、デフォルトで「system」ユーザにのみ書き込み（パブリッシュ）特権が与えられます。

図 2-13　パブリッシュする許可の指定



3. パブリッシュ処理の進行状況は、[Publishing...] ウィンドウに示されます。処理が完了したら、**[Close]**、またはパブリッシュされたパッケージの詳細を参照するための **[Details>>]** をクリックします。

図 2-14 パブリッシュの進行状況ダイアログ



ZAC パッケージを元に戻す

パブリッシュする前のローカルの ZAC パッケージを変更した場合、そのパッケージを、WebLogic Server 上に以前にパブリッシュしたバージョンに戻すことができます。ローカル パッケージを選択し、**[Package]** メニューの **[Revert]** または **[Revert from]** を選択します。複数のサーバを実行している場合、どのサーバから元に戻すかを選択する必要があります。**[Revert]** を選択した場合、自動的に選択状態になるサーバは、リストされている唯一のサーバ、または前回パブリッシュ先または復帰の対象として選択したサーバです。

コマンド ライン Publish Utility の使い方

ZAC Publish Wizard 以外の方法として、Publish Utility を使用すると、ZAC パッケージを WebLogic Server 上でパブリッシュできます。Publish Utility はスタンドアロンの Java アプリケーションであり、コマンド ラインまたはシェル スクリプトから実行できます。Publish Utility のアクションをコンフィグレーションするには、コマンド ライン オプションを指定します。これらのオプションは、ZAC Publish Wizard で定義するパラメータにほぼ従っています。Publish Utility はコマンド ラインで次のように使用します。

```
$ java weblogic.drp.admin.Publish [options]
```

-name *zacPackage*

（必須）パブリッシュする ZAC パッケージの名前。WebLogic Server 上で表示されます。名前にはスペースを使えません。

-dir *packageDir*

（必須）作成およびパブリッシュする ZAC パッケージのすべての内容が入っている最上位のローカル ディレクトリのパス名。ディレクトリの内容と、すべてのサブディレクトリの内容が新しいパッケージに含まれます。

-host *hostname*

（省略可能）パブリッシュする WebLogic Server のホスト名。デフォルトでは 「localhost」です。

-port *portnumber*

（省略可能）パブリッシュする WebLogic Server のポート番号。デフォルトでは「7001」です。

-login *username* -password *passwd*

（省略可能）パブリッシュ認証に関してセキュリティを制御している WebLogic Server 上でパッケージをパブリッシュするためのユーザ名とパスワードを指定する必要があります。デフォルトでは、system ユーザとパスワードに設定されます。パブリッシュ（書き込み）特権をユーザまたはグループに与えるには、`weblogic.properties` ファイルで ACL を指定します。詳細については、「ZAC によるパブリッシュに対応するための WebLogic の設定」（手順 2）を参照してください。

デフォルトでは、Publish Utility はユーザ名とパスワードなしでパッケージをパブリッシュしようとします。ZAC パブリッシュ `write` 特権が `everyone` に与えられていないかぎり、この試みは失敗します。

-verbose | -v

Publish Utility が処理に関する冗長メッセージを出力するようにします。

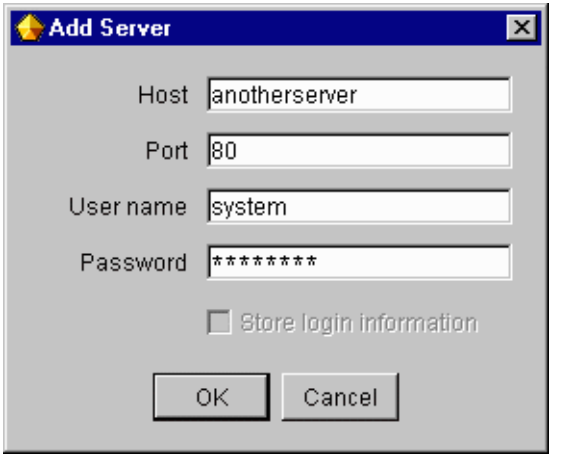
-help

Publish Utility の使い方に関する簡単な説明を出力します。

Publish Wizard と他のサーバとの接続

Publish Wizard を起動すると、デフォルトの場所で実行中の WebLogic Server に自動的に接続し、下側のウィンドウ ペインにその WebLogic Server が表示されます。他の WebLogic Server を検出するには、**[server]** メニューから **[Add]** を選択して次のダイアログにアクセスします。

図 2-15 [Add Server] ダイアログ



パブリッシュ済み ZAC パッケージの更新

すでにパブリッシュされているアプリケーションは簡単に更新できます。

1. Publish Wizard を起動します。
2. 更新するパッケージを選択します。
3. **[Package]** メニューから **[Open]** を選択します。
4. 新規パッケージを作成およびパブリッシュする場合と同じ手順を実行します。

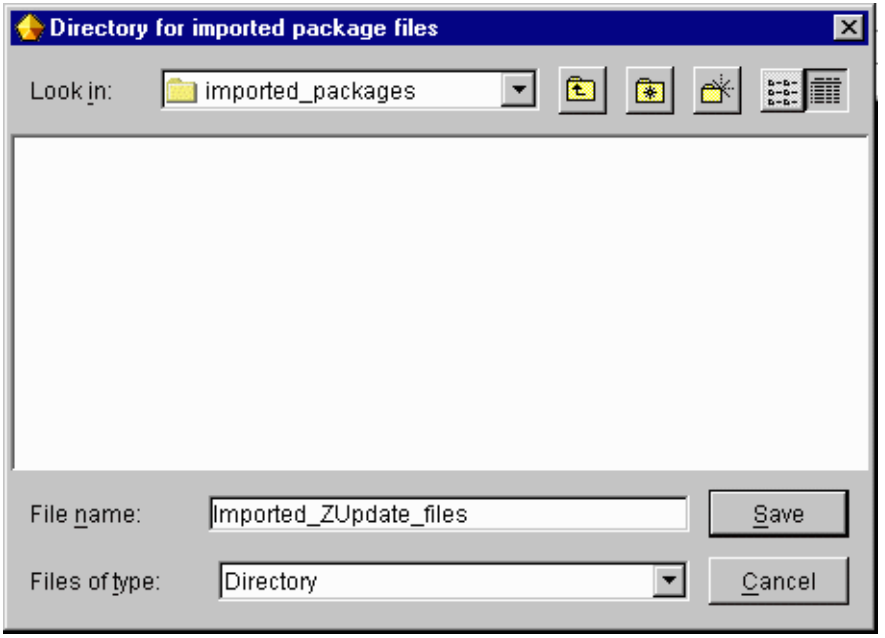
別のサーバからのパブリッシュ済み ZAC パッケージのインポート

ある WebLogic Server からパブリッシュ済みパッケージをインポートして、別の WebLogic Server 上でパブリッシュすることができます。パブリッシュした時点で、パッケージは別のものとなるので、元のパッケージが更新されてもそのパッケージは更新されません。

パブリッシュ済みパッケージをインポートするには、次の手順を実行します。

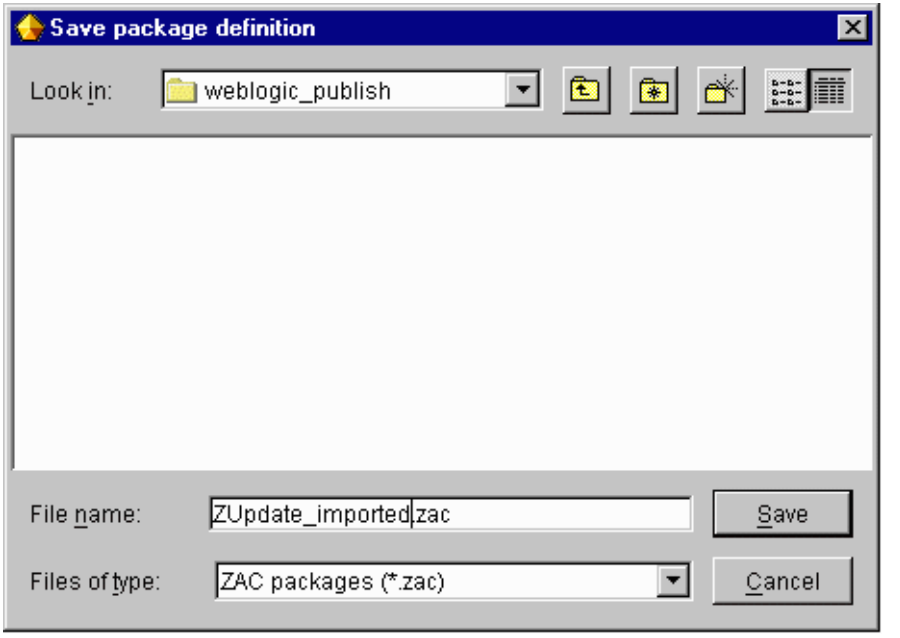
1. インポート元の WebLogic Server ホストを Publish Wizard のホスト リストに追加します。2-18 ページの「Publish Wizard と他のサーバとの接続」を参照してください。
2. 他の WebLogic Server ホストでパブリッシュ済みパッケージを選択します。
3. **[Package]** メニューから **[Import]** を選択します。
4. インポートする ZAC パッケージ ファイルの保存先を指定するよう要求されます。適切なディレクトリを選択して、テキスト ボックスにディレクトリ名を入力します。指定した名前の新しいディレクトリがカレント ディレクトリに作成されます。ディレクトリ名を指定しなかった場合、処理は正しく実行されません。

図 2-16 ZAC パッケージのファイルの保存



5. ZAC パッケージの定義を保存する名前を指定するよう要求されます。これにより、ZAC パッケージの定義がデフォルト ディレクトリに保存されます。通常は、ディレクトリ `\weblogic_publish` に保存されます。

図 2-17　ZAC パッケージの定義の保存



6. インポートしたパッケージは、「ローカル パッケージ」リストに、インポート元のサーバ上でパブリッシュされた名前で表示されます。
7. これで、「ZAC パッケージのパブリッシュ」で説明したように、パッケージをパブリッシュすることができます。

パブリッシュ済み ZAC パッケージの削除

削除処理では、ZAC パブリッシュ ルートのパッケージ ディレクトリに保存されているパッケージ用ディレクトリを含むパブリッシュ済みアプリケーション ファイルだけが削除されます。

パッケージを削除しても、WebLogic Server やパブリッシュされたパッケージの元のファイルは影響を受けず、ローカル ファイルも ZAC クライアントから削除されません。

1. [Server] メニューから、[Remove package] を選択します。
2. パッケージを削除する WebLogic Server を参照します。
3. パッケージを選択して、[Choose] ボタンをクリックします。

インストーラ / ブートストラップ アプリケーションの作成

ZAC Publish Wizard を使用すると、パブリッシュする Java アプリケーションの一部となる一連のネイティブ プログラム（インストーラとブートストラップ）をさまざまなオペレーティング システムに合わせて作成できます。

インストーラ プログラムとは、パブリッシュされた Java プログラムをローカル マシン上にインストールするネイティブ実行可能ファイルのことです。JRE もインストーラがインストールします。インストーラ自体は Java 環境を必要としないので、ネイティブ OS ですぐに実行できます。いわば、Java 用の InstallShield のようなものです。

ブートストラップもネイティブ プログラムで、ユーザはブートストラップを実行して、パブリッシュされているアプリケーションを起動します。ブートストラップは、ユーザのアプリケーションが更新されていないかモニタしたり、更新されている場合にはダウンロードして更新したりする他に、ZAC の管理機能も処理します。

インストーラ / ブートストラップは、以下の OS に合わせて作成できます。

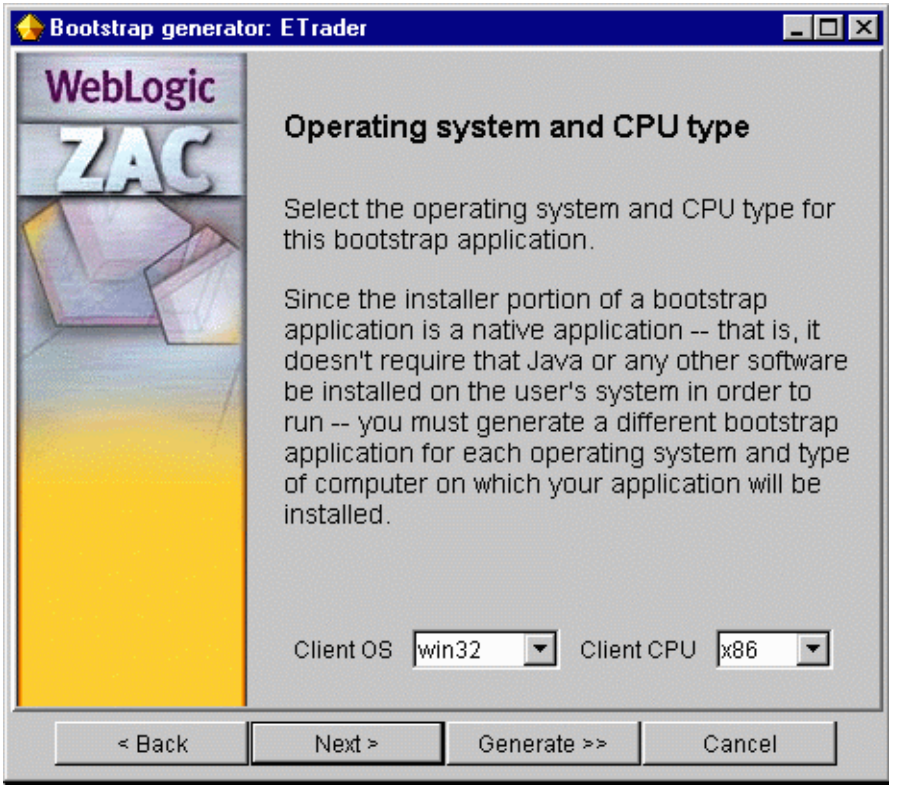
- Win32（Windows95/98 および Windows NT）
- Solaris/SPARC
- Linux/x86
- DECUnix/Alpha
- HPUX（HPUX 11）

インストーラもブートストラップも小さなネイティブ アプリケーションです。これらは、クライアントが使用していると考えられる各オペレーティング システムと CPU のタイプに合わせて作成する必要があります。

Publish Wizard でインストーラ / ブートストラップ実行可能ファイルを作成するには、次の手順を実行します。

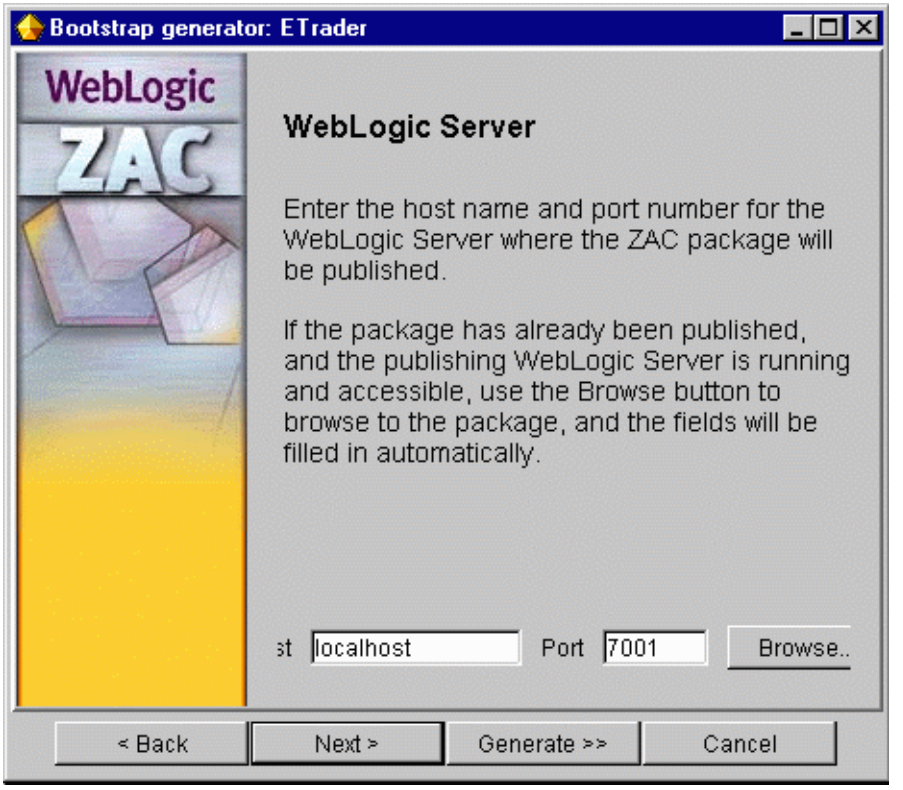
1. Publish Wizard を起動して、インストーラ / ブートストラップを作成する ZAC パッケージを強調表示します。事前に、アプリケーションの ZAC パッケージを作成し、WebLogic Server 上でパブリッシュしておく必要があります。
2. [Package] メニューから [Create bootstrap app...] を選択します。
3. 適切なオペレーティング システムおよび CPU タイプをアプリケーションに設定します。OS と CPU を選択すると、いくつかの値がデフォルトで設定されます。デフォルト値は必要に応じて調整できます。

図 2-18 ブートストラップ用の OS の詳細



4. パブリッシュ側の WebLogic Server のホストとポートを入力します。別のサーバにインストールされているパッケージ用のブートストラップ アプリケーションを準備する場合は、[Browse] ボタンをクリックして、サーバを見つけます。

図 2-19　パブリッシュ側の WebLogic Server の指定

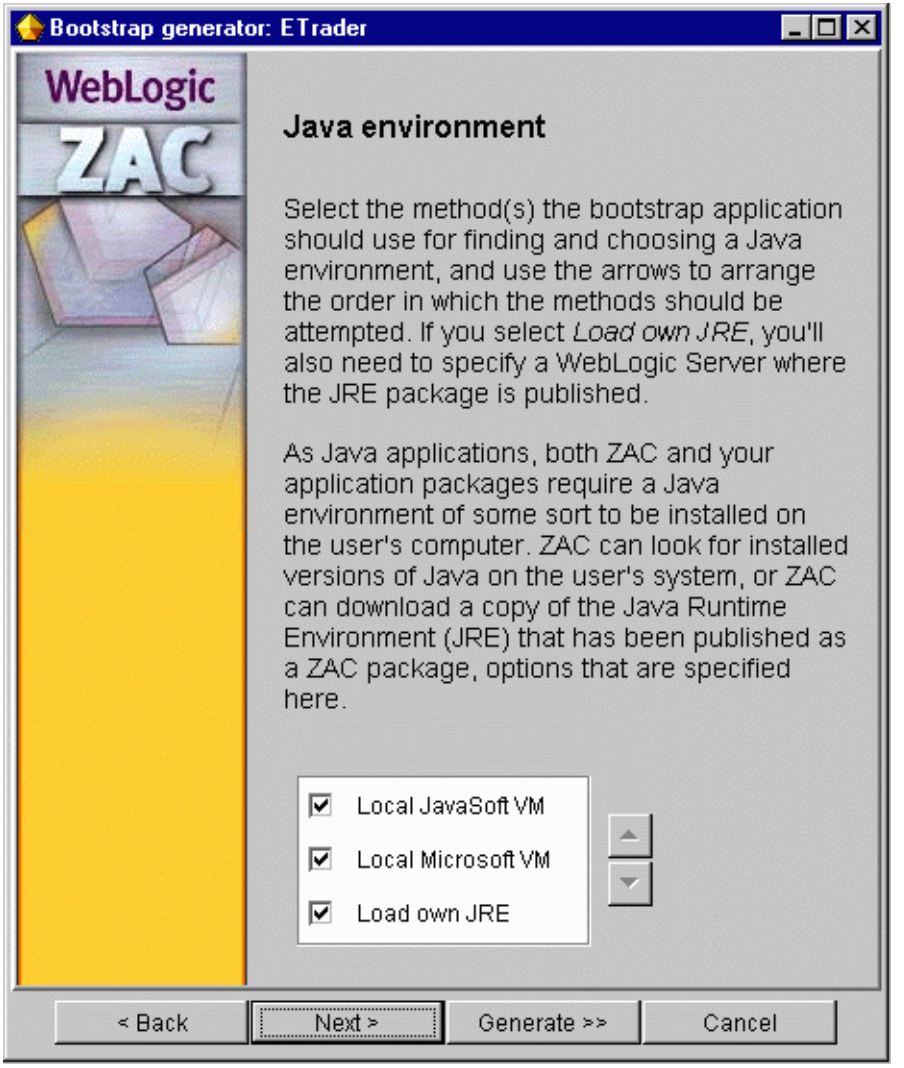


5. ネイティブ ブートストラップ実行可能ファイルの名前を指定します。ブートストラップ作成手順を実行すると、クライアントがダウンロードして最初に実行するインストーラ パッケージ（一般にサイズはごく小さい）と、ユーザがパブリッシュされているアプリケーションを起動するために使用されるブートストラップという 2 つのアプリケーションが作成されます。この手順で名前を付けるのはブートストラップ実行可能ファイルです。

図 2-20 名前の指定

6. 検索して選択する Java 環境を選択します。ローカルで使用可能な Microsoft または JavaSoft の JVM を使用することも、ZAC パッケージとしてパブリッシュされている JRE（Java 実行環境）をロードすることもできます。クライアントで使用可能にするオプションをチェックしてから、右側にある上下の矢印を使用して、それらのオプションを処理する方法を指定します。

図 2-21 クライアント アプリケーションで使用可能にする JRE の指定



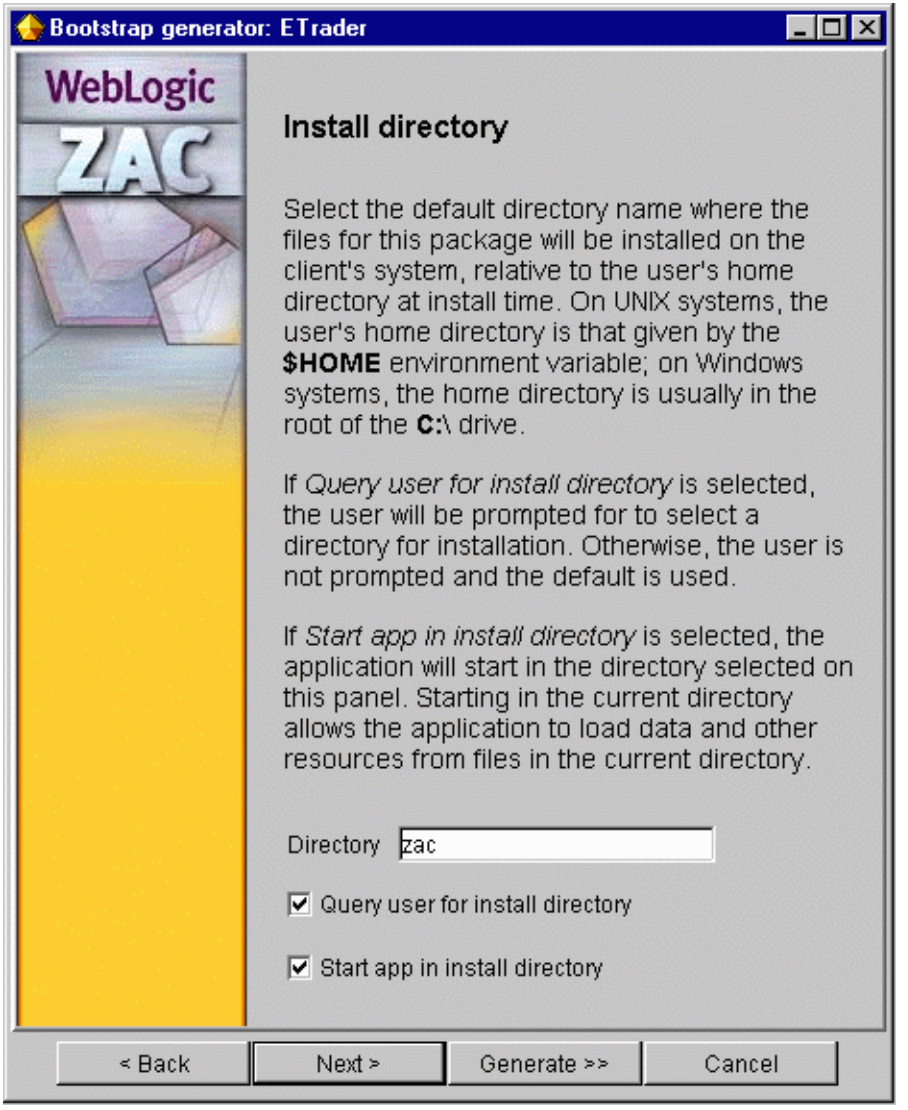
7. [Load own JRE] を選択した場合、パブリッシュされている JRE パッケージをインストーラが探す場所となるパブリッシュ側 WebLogic Server を指定する必要があります。ZAC を使用した独自の JRE のパッケージングの詳細については、このマニュアルの「JRE のパッケージ化」を参照してください。

8. ブートストラップ アプリケーションが Sun VM メモリを初期化する場合のメモリ フラグ オプションを指定します。すでに指定されている値がデフォルトです。アプリケーションが特別に必要とする場合は、それに応じて設定を変更します。

図 2-22 Sun VM のメモリ オプションの設定

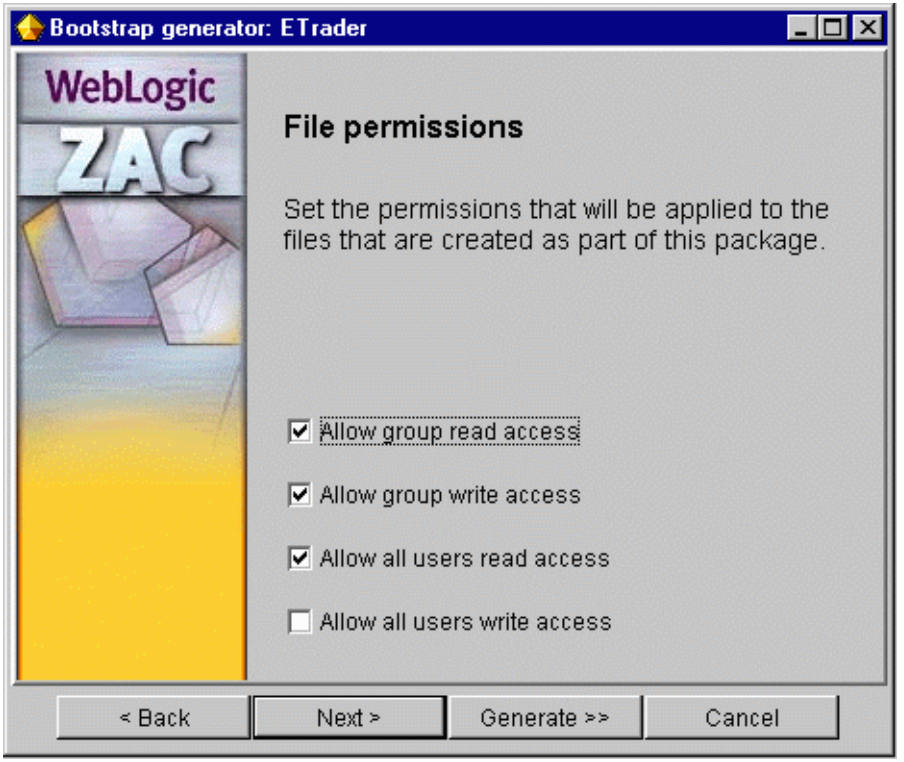
9. ユーザが .exe ファイルをダブルクリックしたときにブートストラップ アプリケーションがインストールされるクライアント マシン上のローカル ディレクトリを入力します。先頭にスラッシュ（スラッシュが前向きか後ろ向きかは、ブートストラップをパブリッシュするオペレーティング システムによって決まります）を使ってパスを指定した場合、ディレクトリは絶対パスになります。[Start app in install directory] オプションを選択した場合、ここで選択したディレクトリがアプリケーションを起動するディレクトリになります。

図 2-23　ローカル クライアント ディレクトリの設定



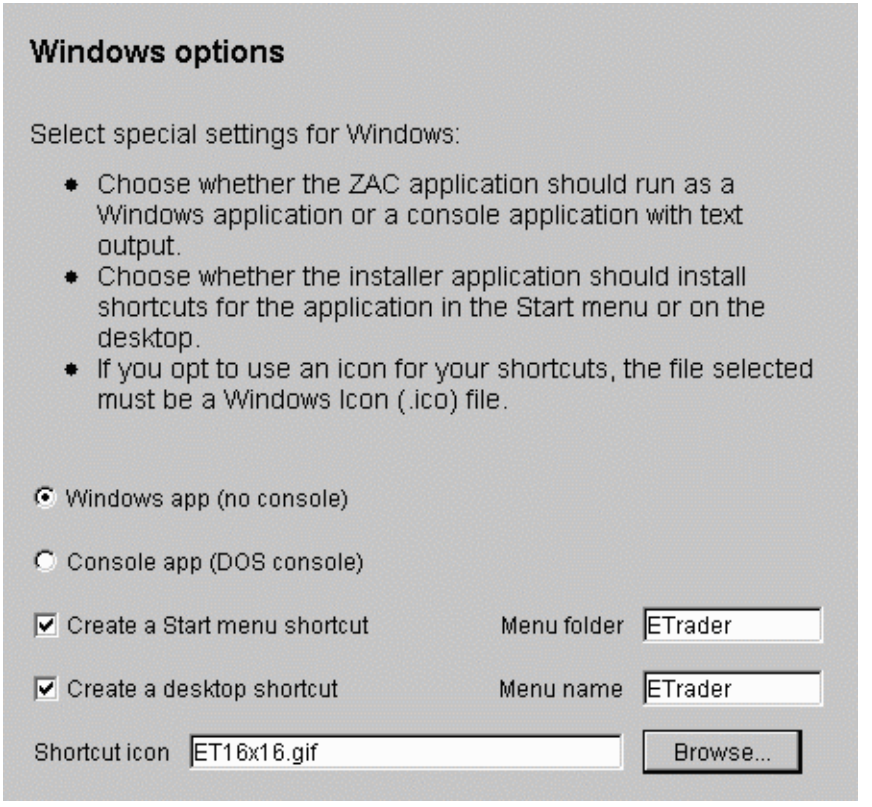
10. このパッケージに関連付けられているファイルに対するクライアントのパーミッションを設定します。

図 2-24 クライアント ファイルに対するアクセス パーミッションの設定



11. Windows に対してパブリッシュする場合、Windows 用の特別な設定を選択する必要があります。Windows の [スタート] メニューまたはデスクトップのショートカットをインストールしたり、ブートストラップ パッケージのアイコンを設定したりすることができます。

図 2-25　Windows 固有のオプションの設定



12. ZAC インストールブートストラップとポストインストールブートストラップの動作を指定します。オプションの定義は以下のとおりです。

    [Update application]

    このオプションをチェックした場合、ブートストラップ実行可能ファイルは、ZAC アプリケーションが新しいバージョンがパブリッシュされていないかチェックして、必要に応じて更新します。以下のいずれかの条件が当てはまる場合は、この機能を無効にします。

- クライアントがアプリケーションをオフラインで実行するようにしたい
- ZAC API を使って ZAC 更新機能をアプリケーションに直接組み込んだ

- ZAC アプリケーションを更新するための別のブートストラップを作成する予定である（[Launch application] オプションを参照）

[Check all dependencies]

このオプションをチェックした場合、ブートストラップ実行可能ファイルは、このアプリケーションが依存する ZAC パッケージの新しいバージョンがパブリッシュされていないかチェックします。上記と同様の理由があれば、この機能を無効にします。

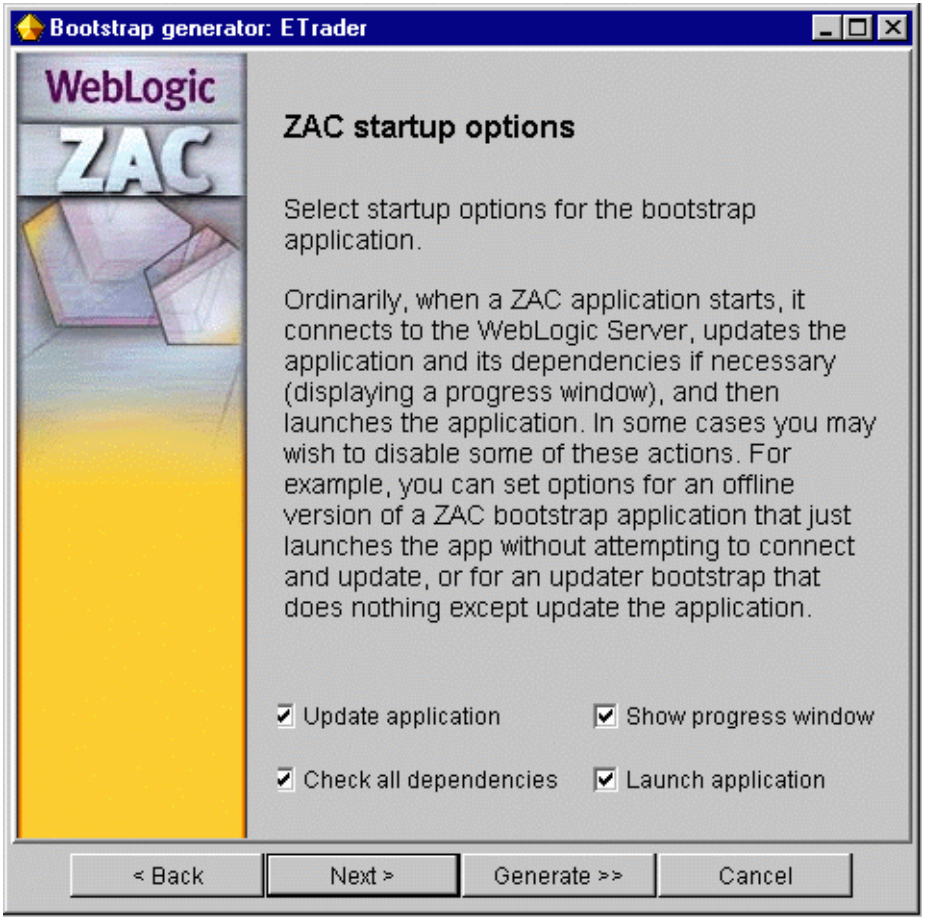
[Show progress window]

このオプションをチェックした場合、ブートストラップ実行可能ファイルは、アプリケーションを更新するときのダウンロードの進行状況を示すメーターを表示します。更新をサイレントで処理する場合、このオプションのチェックをはずします。

[Launch application]

このオプションはチェックするのが一般的です。このオプションをチェックしなかった場合、ブートストラップ実行可能ファイルは、アプリケーションを起動しません。クライアント マシン上でアプリケーションの更新だけを行うブートストラップを作成する場合は、このオプションのチェックをはずします。そのようなブートストラップは、アプリケーションの起動だけを行い、更新を行わないブートストラップと組み合わせて使用します。

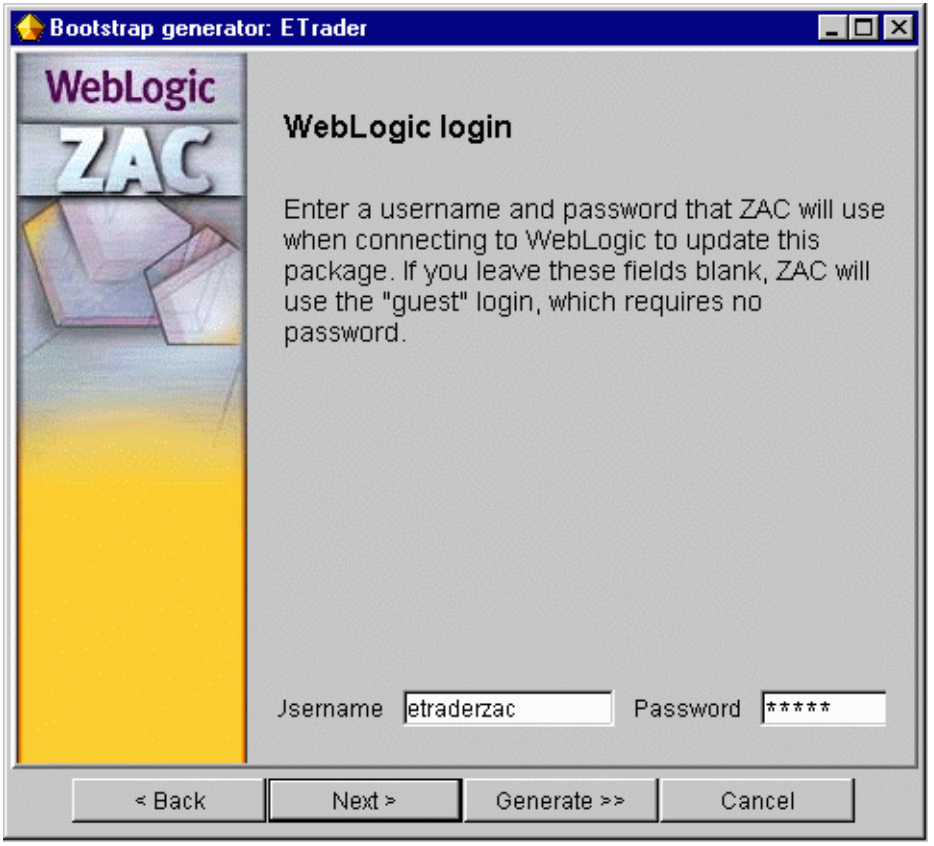
図 2-26　ブートストラップ オプションの設定



13. このパッケージにアクセスするための WebLogic ユーザ名とパスワードを設定します。これは、パブリッシュ側 WebLogic Server 上のアプリケーションに対するアクセス制御リスト（ACL）で読み込みパーミッションを持つ WebLogic ユーザに対応している必要があります。ただし、パブリッシュされているパッケージに ACL が設定されていない場合、読み込みパーミッションは特別なグループ everyone に、書き込み（パブリッシュ）パーミッションは特別な管理ユーザ system にそれぞれデフォルト設定されます。つま

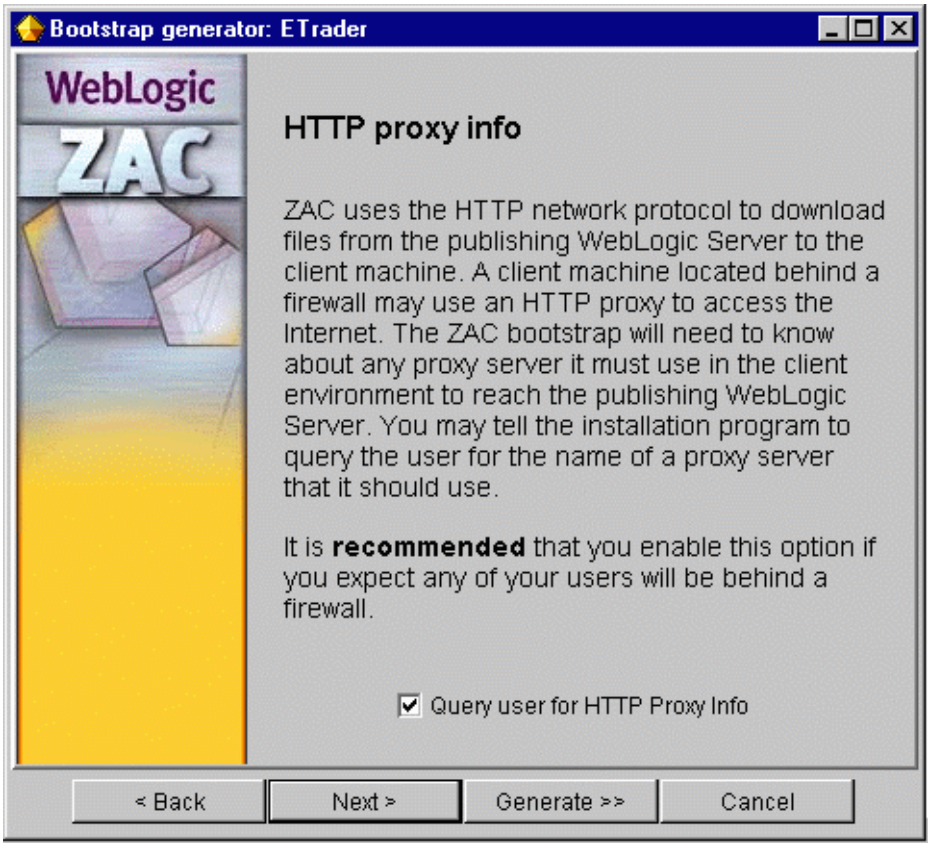
り、どのユーザもパブリッシュされたパッケージをダウンロードすることができますが、パブリッシュまたは再パブリッシュすることができるのはシステムレベルのユーザだけです。

図 2-27 セキュア ZAC パッケージのユーザおよびパスワードの設定



14. ユーザがファイアウォール越しにインターネットにアクセスすると考えられる場合、WebLogic Server にアクセスするときに経由する HTTP プロキシサーバの情報を要求するようブートストラップをコンフィグレーションする必要があります。クライアントがファイアウォールを利用していないことがわかっている場合を除き、このオプションを常にチェックしておくことをお勧めします。

図 2-28　HTTP プロキシの詳細を要求するためのブートストラップのコンフィグレーション

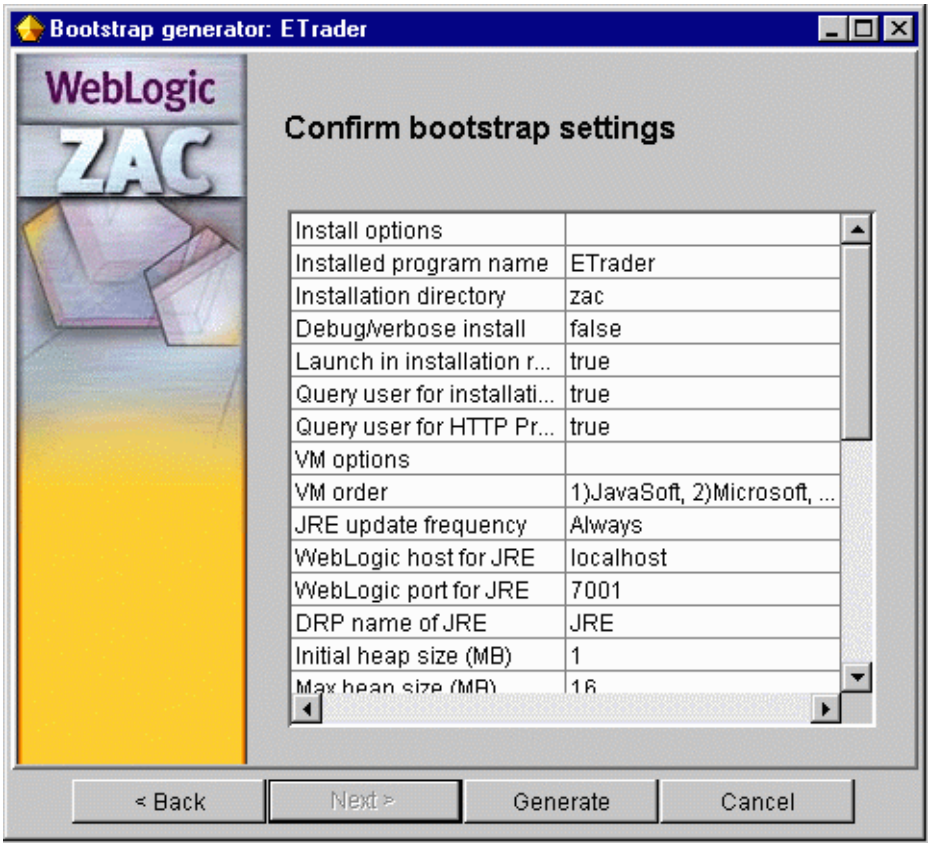


15. デバッグ モードとアプリケーションの冗長性を設定します。クライアント マシンから ZAC アプリケーションのデプロイメントをテストする場合に便利なオプションです。

図 2-29 デバッグ モードの設定

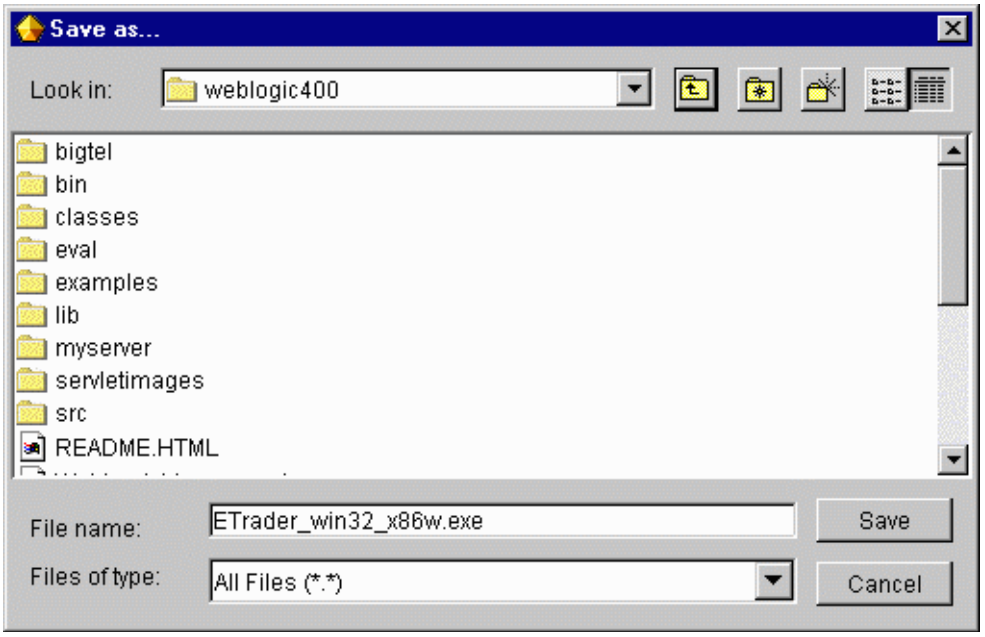
16. ブートストラップの設定を見直します。設定に問題がなければ [Generate] ボタンをクリックします。

図 2-30　設定の確認



17. [Save As...] ウィンドウで、インストーラ実行可能ファイルの保存先と名前を指定します。ZAC パッケージの名前と対象 OS に基づいてデフォルト名が示されていますが、任意に名前を付けることができます。アプリケーションをインストールするクライアントにこの実行可能ファイルをデプロイすることになります。

図 2-31 生成された ZAC インストール プログラムの保存



JRE のパッケージ化

JRE 全体を ZAC パッケージにラップして、ZAC アプリケーションに含めることができます。JRE を含めると、エンド ユーザのマシンに要件を設定する必要がなくなり、正しいバージョンの Java を始めとして、アプリケーションに必要なものすべてが提供されることになります。ただし、最初にダウンロードするパッケージが大きくなる上に、すでに別の JRE が存在する場合でも JRE をインストールすることになる可能性があります。この問題が生じても、ユーザ側でインストールする手間を省けるので、パッケージに含める方がよい場合もあります。

始める前に

JRE の ZAC パッケージを作成する手順は以下のとおりです。この ZAC パッケージをアプリケーションの依存関係として組み入れます。Publish Wizard を起動する前に、以下の作業を行う必要があります。

1. JRE インストールシールドをダウンロードします。JavaSoft の JRE は、JavaSoft からダウンロードできます。
2. JRE を開発マシンにインストールします。この手順では、インストール先のディレクトリを `c:\jre117` とします。

これで、Publish Wizard を起動して次の手順に進むことができます。

ZAC JRE パッケージを作成してパブリッシュする

1. パッケージ タイプを「Library」で作成し、CLASSPATH「.」に置きます（CLASSPATH はこの場合、重要ではありません）。依存関係は設定しません。
2. JRE 用の新しいパッケージを作成します。「JRE_117_win32_x86」のように、OS と CPU タイプに基づいた名前を付けることをお勧めします。
3. JRE パッケージのタイトルと説明を入力します。タイトルは任意に入力できます。
4. できるかぎり JRE バージョンに似たパッケージ バージョンを入力します。次に例を示します。

   JRE 1.1.7 の場合には「1」「1」「7」を入力します。

   JRE 1.2.0 の場合には「1」「2」「0」を入力します。

   JRE をローカルにインストールする場合のデフォルト ディレクトリはバージョン文字列に基づいて決定されるので、JRE のバージョンは正確に入力してください。JRE バージョンと同じバージョン番号を使用すると、2 つの異なる ZAC パッケージが互いの JRE を互換性のないバージョンで上書きする可能性を最小限に抑えることができます。もちろん、このバージョン番号はデフォルトのインストール ディレクトリを決めるために使われるだけで、実際に JRE のインストール先を決めるのはユーザです。

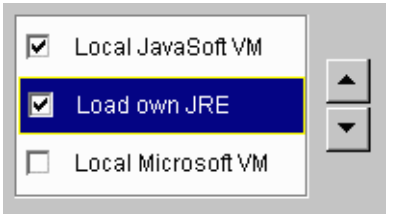
5. オプションとして、パッケージのサムネイル画像を選択することもできます。
6. 以前に JRE をインストールした最上位ディレクトリを指定します（この例では `c:\jre117`）。
7. ライブラリの CLASSPATH としてドット（.）だけを指定します。
8. 他のダイアログ画面をスキップして、パッケージをパブリッシュします。パブリッシュする前に、パブリッシュ側サーバが実行中であることを確認します。

パブリッシュ手順が終了したら、新しい ZAC JRE パッケージが [ZAC Publish Wizard] ウィンドウに表示されます。

パブリッシュされた JRE パッケージをアプリケーション用として指定する

独自の JRE ZAC パッケージ（前節を参照）を作成したので、そのパッケージと ZAC アプリケーションとの依存関係を追加する必要があります。方法は以下のとおりです。

1. Publish Wizard で、アプリケーション用のインストーラ / ブートストラップ実行可能ファイルを作成します。手順については、「インストーラ / ブートストラップ アプリケーションの作成」を参照してください。



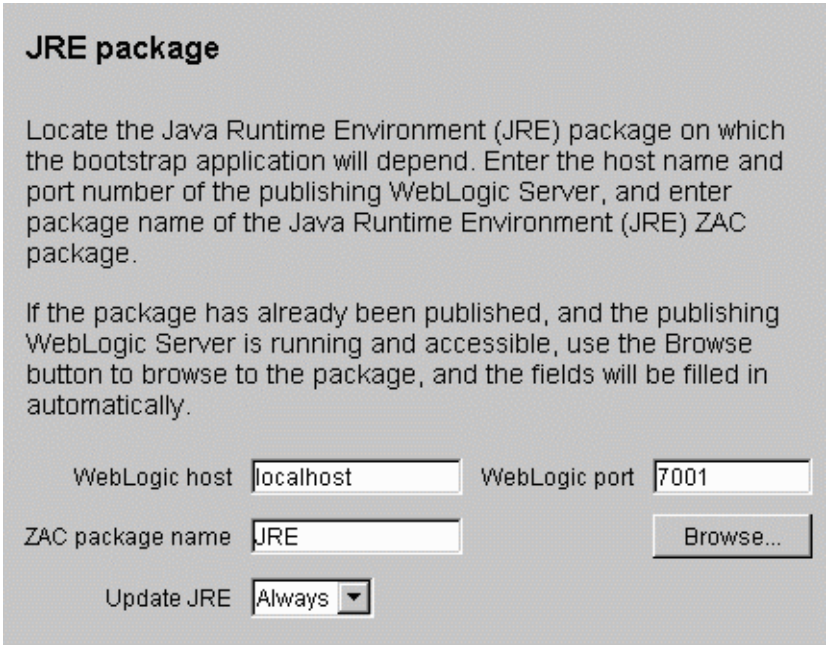
2. 作成手順の中で Java 環境のコンフィグレーション方法を選択する場合、クライアント ユーザのホスト上の JRE を選択する 3 つのオプションが表示されます。右側に示されているように、3 つのオプションを順序付けしたり、選

択 / 選択解除したりすることができます。VM のオプションの 1 つとして [Load own JRE] を選択し、矢印ボタンを使って設定リスト内での位置を調整してから、[Next] ボタンをクリックします。

3. ブートストラップ ウィザードで JRE がパブリッシュされている場所を指定する場合は、以下の情報を提供します。
   - ZAC JRE をパブリッシュした WebLogic Server のホスト名とポート
   - JRE 配布キットをパブリッシュするときに使用した ZAC 名（「JRE_117_win32_x86」など）

   パスを直接入力するか、サーバ リストの該当するサーバ上でパブリッシュされているパッケージを参照します。

図 2-32 パブリッシュされている JRE の指定



4. [Generate] ボタンをクリックするか、ブートストラップ ウィザードの最後に進みます。

5. 最後に、ファイル名とパスを指定して、パブリッシュ側サーバ上でインストーラ実行可能ファイルを保存します。たとえば、`\weblogic\public_html\MyAppInstall.exe` と指定すると、インストーラ / ブートストラップは、HTML ページからのアクセス先となる WebLogic Server ホストのデフォルトのドキュメント ルートに保存されます。

   ネイティブ インストーラのサイズは小さく、Windows アーキテクチャの場合で約 250 KB です。インストーラは、Web ページにリンクを張る、電子メールに添付するなどの方法でクライアントに配布できます。

パブリッシュされたアプリケーションをデバッグおよびテストする

WebLogic 配布キットには、Win32、Solaris、Linux、および DECUnix のコマンド ラインから ZAC ブートストラップを実行するためのネイティブ実行可能ファイルが入っています。Windows バージョン（`zac.exe`）は `bin\` ディレクトリに、非 Windows バージョンは `lib\{arch}\zac_{arch}\` ディレクトリにあります。各実行可能ファイルには、冗長デバッグ用の `_g` バージョンもあります。

適切なバージョンのコマンド ライン ブートストラップを使用して、ZAC パッケージをインストールまたは実行します。たとえば、Windows バージョンの実行方法は次のとおりです。

```
$ zac.exe -name zacPackage options
```

オプションの定義は以下のとおりです。

-name *zacPackage*

必須です。起動またはダウンロードするアプリケーションの名前です。次に例を示します。

```
$ zac.exe -name ETrader
```

-host *hostname*

パブリッシュ側 WebLogic Server のホスト名です。デフォルトでは `localhost` です。次に例を示します。

```
$ zac.exe -name ETrader -host zac.weblogic.com
```

-port *port*

パブリッシュ側 WebLogic Server がログイン リクエストをリスンしているポートです。デフォルトでは 7001 です。次に例を示します。

```
$ zac.exe -name ETrader -host zac.weblogic.com -port 80
```

-proxy

パッケージをパブリッシュするときに ZAC ブートストラップ ウィザードで設定したコンフィグレーションに従って、プロキシ情報を要求します。

-dir *localDir*

ZAC アプリケーション ファイルがあるローカル（クライアント）ディレクトリです。デフォルトでは現在の作業ディレクトリになります。次の例は、*ETrader* という ZAC アプリケーションを起動します。

```
$ zac.exe -name ETrader -dir /usr/local/zac/
```

この例の場合、ETrader アプリケーションのファイルとその依存関係はすべて、/usr/local/zac/ の下のサブディレクトリに入っています。実際の ZAC アプリケーション ETrader は、このディレクトリの下にあるパッケージと同じ名前の別のディレクトリ、つまり /usr/local/zac/ETrader/ に格納されています。ZAC ブートストラップは、このディレクトリ内で OSD アプリケーション マニフェスト（index.osd）を探します。

-root

ZAC ルート ディレクトリの Java ZAC アプリケーションを起動します。デフォルトでは、現在の作業ディレクトリの ZAC アプリケーションを起動します。

-vm *JVM type*

クライアント マシン上で JVM を見つける順序を指定します。文字「S」、「M」、または「O」を任意に組み合わせて指定します。左にある文字が最も優先順位が高く、それぞれの文字が表す JVM は以下のとおりです。

- S（Sun Java VM）
- M（Microsoft VM）
- O（Own VM。独自の JVM を ZAC パッケージとしてパブリッシュします。）

この引数に「O」を使用した場合、-jre フラグで JRE を指定する必要があります。

-jre *zacPackage*

-vm フラグのオプションとして「O」を使用する場合には必須です。ZAC によってパブリッシュされる JRE を指定します。

-D*name=value*

ブートストラップによって呼び出されるときの Java VM に対する Java システム プロパティを指定します。

-option

フラグ「-option」（ハイフンを 1 つ付ける）を、起動時に Java アプリケーションに渡します。

-msnumber

クライアント アプリケーションの JVM 初期ヒープ サイズを設定します（メガバイト単位）。

-mxnumber

クライアント アプリケーションの JVM 最大ヒープ サイズを設定します（メガバイト単位）。

-nolaunch

ZAC パッケージを起動しないで更新します。

-noquery

ブートストラップを実行したときに代替インストール ディレクトリを指定するようユーザに要求するダイアログを無効にします。デフォルトでは、このオプションは有効です。

-noprogress

ダウンロードの進行状況を示すメーターが表示されないようにします。

-noupdate

ZAC アプリケーションの起動だけを行い、更新を試みません。以前にダウンロードが正常に終了している必要があります。

-verbose

Sun JVM でクラスロード時の冗長性を有効化します。このオプションは、クライアント上でクラスの依存関係が見つからないという問題を解決する際に役立ちます。

-help

使用可能なオプションのリストを表示します。各オプションにも簡略化したリストがあります。これらは、-help コマンドの出力として表示されます。

3　WebLogic ZAC を使用した開発

この節では、ZAC（Zero Administration Client）用の API（Application Programmatic Interface）について説明します。内容は以下のとおりです。

はじめに

このマニュアルでは、ZAC API を使用して Java アプリケーションを ZAC 対応にする方法について説明します。また、WebLogic クライアント アプリケーションの作成についても説明します。この説明では、WebLogic 環境で使用できるすべてのサービスおよび機能を取り上げます。

WebLogic ZAC（Zero Administration Client）を使用すると、アプリケーション ソフトウェアの配布とメンテナンスを自動化できます。ソフトウェアの配布、インストール、再インストール、アップグレード、バグ修正パッチの配布、およびデータの配布を手動で行う手間を省くことができます。ZAC は、インターネットまたはイントラネットを経由して、クライアント マシン上のアプリケーション ソフトウェアを常に最新の状態に保ちます。ZAC のサービスは自動的に実行できる上に、エンドユーザに対して透過的です。ZAC は各クライアントを最新の状態にするために必要最小限の変更データだけを転送するので、更新は短時間で効率的に行われます。

ZAC は、W3C 仕様、HTTP Distribution and Replication Protocol の実装です。GUI ウィザードの ZAC Publish Wizard を使用すると、ZAC 対応の WebLogic Server 上でソフトウェアを簡単にパブリッシュできます。また、ZAC には API も用意されており、同じ機能を直接 Java アプリケーションに組み込むことができます。ただし、Java HotSpot TM の有無にかかわらず、ZAC で 1.3.0 JDK を使用することはサポートされていません。

ZAC API をいつ使用するか

ZAC は、既存の Java アプレットおよびアプリケーションのラッパーとして使用したり、ZAC API を使用してアプリケーションに組み込んだりすることができます。ZAC を使用して Java ソフトウェアを配布して自動的に更新する場合、ZAC API を実際に使用する必要はありません。ZAC Publish Wizard を使用すると、アプリケーションをパブリッシュするために必要なすべての情報を指定し、クライアント マシンが使用できるようにして、自動更新ポリシーを設定できます。ZAC を使用したアプリケーションのパブリッシュの詳細については、「WebLogic ZAC を使用したパブリッシュ」を参照してください。

アプリケーションで ZAC の更新サービスを厳密に制御する場合は、ZAC を Java コードに組み込むことを考慮する必要があります。アプリケーションでは、更新がパブリッシュされたときに自動的に応答することも、新しいソフトウェアを更新するタイミングをユーザが制御できるようにすることも可能です。ZAC API のもう 1 つの用途は、他のアプリケーションまたはクライアント マシン上の非実行可能データの更新を管理するアプリケーションを作成することです。たとえば、現場の各営業担当者のラップトップ上で大規模な社内イントラネットの情報を最新の状態に保つ場合のように、各クライアント マシン上で大きなデータセットを管理するとします。変更のたびにデータセット全体をネットワーク経由でダウンロードすると時間がかかりますが、ZAC 対応アプリケーションでは、新しくパブリッシュされたバージョンを見つけると、変更された部分だけがダウンロードされます。

ZAC がアプリケーションをデプロイする方法

ZAC アプリケーションを WebLogic クライアントが使用できるようにするには、ZAC 対応 WebLogic Server 上でパブリッシュする必要があります。ZAC でアプリケーションをパブリッシュするためにアプリケーション コード内に ZAC Java API を使用する必要はありません。パブリッシュ手順の中で、ZAC はクライアントがダウンロード可能な小さなブートストラップ実行可能ファイルを提供し、そのブートストラップが初期ダウンロードと、その後の更新処理（コンフィグレーション可能）、およびパブリッシュされたアプリケーションの起動を行います。このブートストラップ実行可能ファイルは、対象オペレーティング システムごとにコンパイルされます。そのため、Java をクライアント上にプリインストールしておく必要はありません。

ブートストラップをパブリッシュするには、Web ページからその実行可能ファイルへのリンクを張ります。ユーザがブートストラップをダウンロードして実行したら、デスクトップにアイコンがインストールされます。このアイコンをダブルクリックすると、アプリケーションの初期インストールが開始され、次いでアプリケーションが起動されます。パブリッシュされたアプリケーションを起動する前に、ブートストラップは、サーバ上に更新がないかチェックして、必要に応じてアプリケーションを更新します。変更分だけがダウンロードされるので、更新のチェックに続いて更新処理も行う方が効率的です。

ZAC API を使用して コードを Java クライアントに追加すると、アプリケーションを更新するタイミングをさらに細かく制御することができます。ブートストラップを ZAC Publish Wizard で生成する場合（「Publish Wizard の使い方」を参照）、オプションのチェックをはずせるので、クライアントは新しくパブリッシュされた ZAC パッケージを自動的には更新しません。代わりに、ZAC 更新機能をアプリケーション自体に追加すると、ユーザがアプリケーションのユーザインタフェースを利用していつアプリケーションを更新するかを選択できるようになります。

WebLogic ZAC API

Package-weblogic.zac

```
Class java.lang.Object
  Class weblogic.zac.ZAC
    (implements weblogic.drp.events.ProgressListener,
    weblogic.drp.common.DRPConstants)
  Class weblogic.zac.ZACLog
```

`ZAC` クラスを使用して、アプリケーション内からクライアント マシン上の ZAC パッケージを管理します。一般には、`update()` メソッドなど、このクラスのメソッドのごく小さなセットを使用するだけで、パブリッシュ側 WebLogic Server からクライアント マシン上の ZAC パッケージを更新できます。

`ZACLog` クラスは、クライアントが使用しているパブリッシュ済み ZAC パッケージと、各パッケージの最新の更新についての情報を提供します。

WebLogic ZAC を使用した実装

この節では以下の内容について説明します。

パッケージをインポートする

ZAC クラスを Java アプリケーションまたはアプレットで使用するには、WebLogic ZAC パッケージだけでなく、すべての WebLogic クライアントをサポートするパッケージもインポートする必要があります。次に例を示します。

```
import weblogic.zac.*;
import weblogic.common.*;
```

ZAC アプリケーションを更新する

アプリケーションでは、ZAC パッケージを参照して更新するために、ZAC オブジェクトを作成および使用します。ZAC コンストラクタには、管理する ZAC パッケージについて以下の詳細を指定する必要があります。

- ZAC パッケージをパブリッシュする場所となる WebLogic Server のアドレス
- WebLogic Server 上でパブリッシュされる ZAC パッケージの名前
- クライアント マシン上で ZAC パッケージをインストールするローカル ディレクトリ

ZACLog クラスを使用すると、クライアント アプリケーションで使用している各 ZAC パッケージについての情報を取得してから、その情報を ZAC コンストラクタに渡すことができます。この処理を行う方法を以下のコードで説明します。

```
// パブリッシュする各サーバのアドレスを見つけて
// それぞれに順に接続する
ZACLog zl;
ZAC zac;
// このアプリケーションが依存しているパブリッシュ済み
// パッケージごとの ZACLog の列挙値を取得する
Enumeration enum = ZACLog.getUpdateLogs();
while (enum.hasMoreElements()) {
  zl = (ZACLog)enum.nextElement();
  // 各 ZACLog を基にサーバ アドレスの URL を作成する
  t3url = "t3://" + zl.getZACHost() + ":" + zl.getZACPort() + '/';
  // 新しい ZAC オブジェクトを作成して、サーバに接続する
  zac = new ZAC(zl.getZACHost(),
                zl.getZACPort(),
                zl.getZACName(),
                zl.getLocalDirectory());
  // ZAC パッケージの更新を実行する
  zac.update();
}
```

上記の例では、列挙された ZACLog インスタンスを調べるために、ZACLog.getUpdateLogs() メソッドを使用しています。各 ZACLog インスタンスは、このアプリケーションが依存している別々の ZAC パッケージを参照しています。各 ZACLog のプロパティを使用して、各 ZAC パッケージの ZAC オブジェクトを作成します。次に、zac.update() メソッドを使用して、各 ZAC パッケージを更新します。

アプリケーションが依存している以外の別の ZAC パッケージを管理する場合、ZACLog.getUpdateLogs() メソッドは、現在の ZAC アプリケーションが使用している各パッケージの ZACLog だけを返すので、効果がありません。この場合には、アプリケーションで、パブリッシュ側 WebLogic Server に関する明示的な詳細とパッケージの名前をコンストラクタに渡す必要があります。

ZAC オブジェクトを作成したら、その update() メソッドを呼び出すことができます。このメソッドは ZAC パッケージが新しくパブリッシュされていないかチェックして、必要に応じてクライアントのパッケージを更新します。

ZACLog を使用して最新の更新を問い合わせる

各 ZAC 更新の詳細は、一連の ZACLog オブジェクトに記録されます。更新が行われるのは、アプリケーションが起動するときやアプリケーションが更新自体を初期化するときです（上記参照）。静的メソッドである `ZACLog.getUpdateLogs()` を使用して最新の更新から ZACLog オブジェクトの列挙値を取得します。以前の ZACLog レコードはすべて破棄されます。アプリケーションは、必要に応じて更新レコードを保守しなければなりません。

`getUpdateLogs()` メソッドを呼び出して返された列挙値には、アプリケーションが依存している各パッケージにつき 1 つの ZACLog が入っています。この列挙値を以下のコードで処理します。

```
  // ZACLog の列挙値を取得する
  Enumeration enum = ZACLog.getUpdateLogs();
  ZACLog zl;
  // 各 ZACLog を処理する
  while(enum.hasMoreElements()) {
    zl = (ZACLog)enum.nextElement();
    // ZAC パッケージ名を出力する
    System.out.println("ZAC log for package" + zl.getZACName());

    // ZAC の更新ステータスを出力する
    switch(zl.getUpdateStatus()) {
    case ZACLog.UPDATE_NONE:
      System.out.println("ZAC update status: No update was
necessary.");
      break;
    case ZACLog.UPDATE_FAILURE:
      System.out.println("ZAC update status: FAILED!");
     System.out.println("Details: " + zl.getUpdateFailureString());
      break;

    // ここから実際の処理が開始される
    case ZACLog.UPDATE_SUCCESS:
      System.out.println("ZAC update status: Completed
successfully");

      // 更新についての情報を取得する
      int fileCnt = zl.getUpdateFileCount();
      long binarySize = zl.getUpdateByteCount();

      String details;

      if (fileCnt == 0) {
        details = "Update Success, 0 files updated, 0 bytes
transferred.";
      }
      else {
```

```
          if (fileCnt == 1) {
            details = "Update Success, 1 file updated, ";
          }
          else {
          details = "Update Success, " + fileCnt + " files updated, ";
          }

          if (binarySize > 1000) {
            details += (binarySize / 1000) + " KBytes transferred.";
          }
          else {
            details += (binarySize) + " bytes transferred.";
          }
        }
        System.out.println("Details: " + details);

        // 更新されたファイルが報告される
        System.out.println("The following files were updated:");
        Enumeration fl = zl.getUpdateFileList();
        File zacroot = zl.getLocalDirectory();
        while (fl.hasMoreElements()) {
          String path = (String)fl.nextElement();
          File updated = new File(zacroot, path);
         System.out.println("Updated: " + updated.getAbsolutePath());
        }
      }
    }
```

この例は、ZACLog から最新の ZAC 更新についての情報を取得する方法を示しています。まず、ZACLog が参照している ZAC パッケージの名前を取得して、コンソールに出力します。次に、ZAC 更新のステータスを問い合わせ、その結果に基づいて `switch` 文で処理を実行します。`getUpdateStatus()` メソッドは以下のいずれかの定数を返します。

`ZACLog.UPDATE_NONE`
: 更新の必要はありません。パッケージは最新です。

`ZACLog.UPDATE_FAILURE`
: 更新に失敗しました。失敗した理由を調べるには、`getUpdateFailureString()` メソッドおよび `getUpdateFailure()` メソッドを呼び出します。これらのメソッドは、スタック トレースを格納する文字列と発生した `Throwable` 例外をそれぞれ返します。更新が失敗する原因としては、サーバ エラー、通信の中断、クライアント マシンのディスク容量不足などが考えられます。

`ZACLog.UPDATE_SUCCESS`
: 更新が成功したことを示します。更新が成功した場合には、`ZACLog` について、転送データのサイズや更新されたファイルなどの情報も取得できます。

getUpdateFileList() メソッドによって返される更新済みファイルの名前は、ZAC パッケージのインストール ディレクトリを起点としています。上記の例の場合、FILE オブジェクトは相対パス名で構築されているので、getLocalDirectory() メソッドによって返されるパスに追加すると、各ファイルの絶対パス名になります。

更新によってアプリケーションに適用された変更は、アプリケーションを再起動するまで有効になりません。

ZAC クライアント アプリケーションを再起動する

クライアント アプリケーションが ZAC ブートストラップによって起動された場合、ZAC.ZAC_EXIT_RESTART フラグを System.exit() メソッドに送信することで、アプリケーションの再起動を要求できます。次に例を示します。

```
System.exit(ZAC.ZAC_EXIT_RESTART);
```

ZAC ブートストラップを使用してアプリケーションを起動したので、ブートストラップで終了ステータスを取得することもできます。ZAC.ZAC_EXIT_RESTART ステータスを取得すると、ブートストラップはアプリケーションを再起動します。ブートストラップは、コンフィグレーションによって、再起動する前にアプリケーションとアプリケーションが依存する他の ZAC パッケージの更新処理も実行します。このコンフィグレーションは、ZAC Publish Wizard を使用してアプリケーションをパブリッシュするときに行います。

ZAC で WebLogic Event を使用する

WebLogic Event をクライアント アプリケーションに組み込むと、新しいバージョンの ZAC パッケージが WebLogic Server 上でパブリッシュされた場合に、クライアント アプリケーションがすぐに応答できるようにすることができます。

注意： WebLogic Event は、WebLogic Server リリース 6.0 より非推奨になりました。

WebLogic Server は、myPackage という新しいパッケージの更新が WebLogic Server 上でパブリッシュされると、WEBLOGIC.ZAC.UPDATE.myPackage トピックの新しいイベントを生成します。アプリケーションでイベント トピック

WEBLOGIC.ZAC.UPDATE に関係を登録して、sink フラグを true に設定すると、新しいバージョンのパッケージがパブリッシュされたときにアプリケーションに通知されます。イベントの詳細については、『WebLogic Event ユーザーズ ガイド』を参照してください。

次のコードは、クライアントが WebLogic Server に接続して、ZAC パッケージのパブリッシュに関係を登録する方法を示しています。この例では、クライアントは "WEBLOGIC.ZAC.UPDATE.myPackage" イベント トピックに関係を登録します。

```
ZACLog zl;
String t3url = null;
while (enum.hasMoreElements()) {
  ZACLog zl = (ZACLog)enum.nextElement();
  // パッケージ名と一致するかどうかをテストする
  if (zl.getZACName().equals("myPackage")) {
    t3url = "t3://" + zl.getZACHost() +
            ":" + zl.getZACPort() + '/';
    break;
  }
}

if (t3url != null) {
  // ここでサーバに接続する
  T3ServicesDef t3services = getT3Services(t3url);

  // 更新イベントへの関心を登録する
  // 登録で使用される Evaluate オブジェクトを作成する
  Evaluate eval =
    new Evaluate("weblogic.event.evaluators.EvaluateTrue");

  // 登録用の Action オブジェクト パラメータを作成する。
  // クライアントに通知が返されるようにするには、クライアントが
  // ActionDef を実装し、Action が「this」などの
  // オブジェクトを使用してインスタンス化される必要がある
  Action act = new Action(this);
  // EventRegistrationDef オブジェクトを作成する
  EventRegistrationDef erd =
    t3services.events()
      .getEventRegistration("WEBLOGIC.ZAC.UPDATE.myPackage",
                            eval, act, true, true, 1);

  // 最後に、イベントへの関心を登録する
  int regid = erd.register();
}
```

クライアントには、ActionDef インタフェースの一部である action() メソッドを実装する必要があります。action() メソッドでは、クライアントは、更新がパブリッシュされたことを示す通知に基づいて処理を行い、自身を更新することができます。次に例を示します。

```
public synchronized void action(EventMessageDef ev) {
  System.out.println("Notification of an " +
                     ev.getTopic() +
                     " Event received.");
  zacUpdate = true;
  notifyAll();
  }
```

通常、action() メソッドは同期され、複数のスレッドを同時に実行することを防ぎます。この実装では、メッセージをコンソールに出力して、private 変数 zacUpdate を true に設定し、イベントを処理するために、notifyAll() メソッドで他のアプリケーション スレッドに通知しています。zacUpdate 変数は、ZAC 更新を実行する停止中のアプリケーション スレッドに通知されます。

注意： イベントの通知は別のスレッドから呼び出されるので、この例のように、コールバック メソッドで時間のかかる処理を実行することは避けてください。すぐに復帰する小さな処理は実行してもかまいませんが、この例では、比較的時間のかかる処理である ZAC 更新をクライアント アプリケーション スレッドの 1 つで処理するのが最適であると判断しています。

ZAC アプリケーションと一緒にライブラリをパッケージ化する

ZAC アプリケーションをパブリッシュする場合は、必要なライブラリもすべてパブリッシュしなければなりません。Weblogic クラスの ZAC サブセットは、ZAC アプリケーションと一緒にデフォルトで配布され、アプリケーションが ZAC API を使用しているかどうかにかかわりなく、アプリケーションの操作に必要となります。これらのクラスは zac.jar ファイルとしてクライアントにダウンロードされ、ZAC アプリケーションのインストール ディレクトリの下の lib ディレクトリに格納されます。ZAC クラスはデフォルトで提供されるので、ZAC API を使用するアプリケーションはこれらのクラスをデプロイする必要がありません。

ただし、zac.jar ファイルに入っていない WebLogic リソース、またはデフォルト以外のライブラリやパッケージを使用するアプリケーションの場合は、それらのリソースもクライアント マシン上にインストールされるようにコンフィグレーションする必要があります。この処理を行う方法は以下の 2 通りです。

ZAC パッケージにライブラリを含める

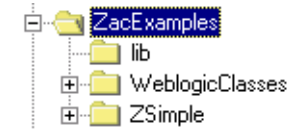
必要なクラス、jar、ライブラリなど、ZAC アプリケーションのパブリッシュ ディレクトリの下にあるすべてのデータを含めた場合、それらはパッケージと一緒に自動的にデプロイされ、クライアント マシン上で同じ相対パッケージ ディレクトリの下にインストールされます。

アプリケーションがアプリケーション パッケージに入っていないクラスに依存する場合、それらのクラスを ZAC アプリケーションの CLASSPATH に指定する必要があります。ZAC Publish Wizard でアプリケーションの実行時 CLASSPATH を指定する（「Publish Wizard の使い方」参照）ことが可能です。CLASSPATH は、ルート インストール ディレクトリを起点に指定します。

ZAC パッケージと別の ZAC パッケージとの間に依存関係を設定する

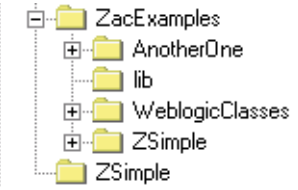
同じライブラリまたはクラスを使用する複数の ZAC アプリケーションをパブリッシュする場合は、共有コンポーネントを別のパッケージとしてパブリッシュすると、クライアント マシン上のディスク領域を節約できます。その上で、各パッケージと共有パッケージとの依存関係を構築します。アプリケーションが依存するパッケージは、アプリケーション自体も含めて、クライアント アプリケーションが更新されるたびに一緒に更新されます。

ZAC がクライアント上にパッケージをインストールすると、すべてのものがルート ディレクトリの下にインストールされます。最初にパブリッシュした ZAC アプリケーションは、このルート ディレクトリの下のサブディレクトリに同じ名前でインストールされます。その他のパッケージもすべて、このルート ディレクトリの下のパッケージと同じ名前のサブディレクトリに格納されます。ZAC は、ルート ディレクトリにインストールされたという情報だけを持ちます。インストールルート ディレクトリの場所は、パッケージをパブリッシュするときに定義されるか、またはパッケージのコンフィグレーションによって可能な場合にユーザがインストール時に指定します。



この例では、ZAC ルート ディレクトリは ZacExamples で、2 つのパッケージ、ZSimple と WeblogicClasses を格納しています。この場合、ZSimple パッケージが WeblogicClasses に依存している可能性があります。

2 つの ZAC パッケージが別のパッケージに依存している場合、どちらの ZAC パッケージも同じルート ディレクトリの下にインストールする必要があります。



この図は、AnotherOne という別のパッケージが同じルート ディレクトリの下にインストールされたときのディレクトリ構造を示しています。どちらのパッケージも同じ WeblogicClasses パッケージに依存しており、必要であればどちらも更新されます。

注意： 同じルートにインストールされていない場合は、それぞれが同じパッケージの別々のコピーを、対応するルート ディレクトリの下に保持します。これは、別のパッケージが依存するパッケージを ZAC パッケージが更新してしまうことがないようにするためのデフォルト動作です。