



# BEA WebLogic Server

WebLogic Server アプレット  
ユーザーズ ガイド

BEA WebLogic Server 6.1  
マニュアルの日付：2001 年 12 月 19 日

## 著作権

Copyright © 2001 BEA Systems, Inc. All Rights Reserved.

## 限定的権利条項

本ソフトウェアおよびマニュアルは、BEA Systems, Inc. 又は日本ビー・イー・エー・システムズ株式会社（以下、「BEA」といいます）の使用許諾契約に基づいて提供され、その内容に同意する場合にのみ使用することができ、同契約の条項通りにのみ使用またはコピーすることができます。同契約で明示的に許可されている以外の方法で同ソフトウェアをコピーすることは法律に違反します。このマニュアルの一部または全部を、BEA からの書面による事前の同意なしに、複写、複製、翻訳、あるいはいかなる電子媒体または機械可読形式への変換も行うことはできません。

米国政府による使用、複製もしくは開示は、BEA の使用許諾契約、および FAR 52.227-19 の「Commercial Computer Software-Restricted Rights」条項のサブパラグラフ (c)(1)、DFARS 252.227-7013 の「Rights in Technical Data and Computer Software」条項のサブパラグラフ (c)(1)(ii)、NASA FAR 補遺 16-52.227-86 の「Commercial Computer Software--Licensing」条項のサブパラグラフ (d)、もしくはそれらと同等の条項で定める制限の対象となります。

このマニュアルに記載されている内容は予告なく変更されることがあり、また BEA による責務を意味するものではありません。本ソフトウェアおよびマニュアルは「現状のまま」提供され、商品性や特定用途への適合性を始めとする（ただし、これらには限定されない）いかなる種類の保証も与えません。さらに、BEA は、正当性、正確さ、信頼性などについて、本ソフトウェアまたはマニュアルの使用もしくは使用結果に関していかなる確約、保証、あるいは表明も行いません。

## 商標または登録商標

BEA、Jolt、Tuxedo、および WebLogic は BEA Systems, Inc. の登録商標です。BEA Builder、BEA Campaign Manager for WebLogic、BEA eLink、BEA Manager、BEA WebLogic Collaborate、BEA WebLogic Commerce Server、BEA WebLogic E-Business Platform、BEA WebLogic Enterprise、BEA WebLogic Integration、BEA WebLogic Personalization Server、BEA WebLogic Process Integrator、BEA WebLogic Server、E-Business Control Center、How Business Becomes E-Business、Liquid Data、Operating System for the Internet、および Portal FrameWork は、BEA Systems, Inc. の商標です。

その他の商標はすべて、関係各社がその権利を有します。

## WebLogic Server アプレット ユーザーズ ガイド

パート番号	マニュアルの日付	ソフトウェアのバージョン
なし	2001 年 12 月 19 日	WebLogic Server バージョン 6.1

---

# 目次

## このマニュアルの内容

対象読者.....	v
e-docs Web サイト.....	v
このマニュアルの印刷方法.....	vi
サポート情報.....	vi
表記規則.....	vii

## 1. WebLogic Server でのアプレットの使用

はじめに.....	1-1
アプレットの動作と仕組み.....	1-1
アプレットをいつ使用するか.....	1-2
WebLogic Server の Web プレゼンテーション — ベストプラクティス.....	1-3
アプレットの使用 — 確認済みの制限.....	1-4
アプレット使用のデメリット.....	1-4
ClassNotFoundException.....	1-4
ClassCastException.....	1-5
Java Plug-in の使い方.....	1-5
CODEBASE 属性.....	1-7
CODE 属性.....	1-8
トラブルシューティングとパフォーマンス.....	1-8
アプレットのトラブルシューティング.....	1-9
アプレットがブラウザで動作しない.....	1-9
ClassFormatError.....	1-9
ローカル環境でのテスト.....	1-10
ローカル開発環境からの移動.....	1-11
アーカイブによるアプレットの高速化.....	1-11



---

# このマニュアルの内容

このマニュアルでは、WebLogic Server におけるアプレットの使い方について説明します。内容は以下のとおりです。

- 第 1 章「WebLogic Server でのアプレットの使用」

## 対象読者

このマニュアルは、Web アプリケーションの構築に関心のあるアプリケーション開発者を対象としています。アプレットおよび Java プログラミングに読者が精通していることを前提として書かれています。

## e-docs Web サイト

BEA 製品のドキュメントは、BEA の Web サイトで入手できます。BEA ホームページの [製品のドキュメント] をクリックするか、WebLogic Server 製品ドキュメント ページ (<http://edocs.beasys.co.jp/e-docs/wls61>) を直接表示してください。

---

# このマニュアルの印刷方法

Web ブラウザの [ ファイル | 印刷 ] オプションを使用すると、Web ブラウザからこのマニュアルを一度に 1 章ずつ印刷できます。

このマニュアルの PDF 版は、Web サイトで入手できます。WebLogic Server PDF を Adobe Acrobat Reader で開くと、マニュアルの全体（または一部分）を書籍の形式で印刷できます。PDF を表示するには、WebLogic Server ドキュメントのホームページを開き、[ ドキュメントのダウンロード ] をクリックして、印刷するマニュアルを選択します。

Adobe Acrobat Reader は Adobe の Web サイト (<http://www.adobe.co.jp>) で無料で入手できます。

## サポート情報

BEA のドキュメントに関するユーザからのフィードバックは弊社にとって非常に重要です。質問や意見などがあれば、電子メールで [docsupport-jp@beasys.com](mailto:docsupport-jp@beasys.com) までお送りください。寄せられた意見については、ドキュメントを作成および改訂する BEA の専門の担当者が直に目を通します。

電子メールのメッセージには、ご使用のソフトウェアの名前とバージョン、およびドキュメントのタイトルと日付をお書き添えください。本バージョンの BEA WebLogic Server について不明な点がある場合、または BEA WebLogic Server のインストールおよび動作に問題がある場合は、BEA WebSUPPORT ([www.beasys.com](http://www.beasys.com)) を通じて BEA カスタマ サポートまでお問い合わせください。カスタマ サポートへの連絡方法については、製品パッケージに同梱されているカスタマ サポート カードにも記載されています。

カスタマ サポートでは以下の情報をお尋ねしますので、お問い合わせの際はあらかじめご用意ください。

- お名前、電子メールアドレス、電話番号、ファクス番号
- 会社の名前と住所
- お使いの機種とコード番号

- 製品の名前とバージョン
- 問題の状況と表示されるエラー メッセージの内容

## 表記規則

このマニュアルでは、全体を通して以下の表記規則が使用されています。

表記法	適用
[Ctrl] + [Tab]	複数のキーを同時に押すことを示す。
<i>斜体</i>	強調または書籍のタイトルを示す。
等幅テキスト	コード サンプル、コマンドとそのオプション、データ構造体とそのメンバー、データ型、ディレクトリ、およびファイル名とその拡張子を示す。等幅テキストはキーボードから入力するテキストも示す。 例： <pre>import java.util.Enumeration; chmod u+w * config/examples/applications .java config.xml float</pre>
<i>斜体の等幅テキスト</i>	コード内の変数を示す。 例： <pre>String CustomerName;</pre>
すべて大文字のテキスト	デバイス名、環境変数、および論理演算子を示す。 例： <pre>LPT1 BEA_HOME OR</pre>

表記法	適用
{ }	構文の中で複数の選択肢を示す。
[ ]	構文の中で任意指定の項目を示す。 例： <pre>java utils.MulticastTest -n name -a address       [-p portnumber] [-t timeout] [-s send]</pre>
	構文の中で相互に排他的な選択肢を区切る。 例： <pre>java weblogic.deploy [list deploy undeploy update]       password {application} {source}</pre>
...	コマンドラインで以下のいずれかを示す。 <ul style="list-style-type: none"> <li>◆ 引数を複数回繰り返すことができる</li> <li>◆ 任意指定の引数が省略されている</li> <li>◆ パラメータや値などの情報を追加入力できる</li> </ul>
.	コードサンプルまたは構文で項目が省略されていることを示す。 . .



---

# 1 WebLogic Server でのアプレットの の使用

## はじめに

BEA は、限られたケースでのみアプレットの使用をサポートしています。このマニュアルでは、アプレットの使用を検討する上で役立つその他のオプションも示します。また、BEA が推奨するベストプラクティス以外の方法で WebLogic Server とアプレットを使用するユーザのために、Sun のサイトへのリンクを示してあります。

- 1-1 ページの「アプレットの動作と仕組み」
- 1-2 ページの「アプレットをいつ使用するか」
- 1-5 ページの「Java Plug-in の使い方」
- 1-8 ページの「トラブルシューティングとパフォーマンス」

## アプレットの動作と仕組み

この節では、アプレットの機能について簡単に説明します。詳細については、Sun の Java Web サイトの Applets を参照してください。

アプレットは、次のように、<APPLET> タグを使用して HTML ページに埋め込まれます。

```
<APPLET CODE="HelloWorld.class"  
        CODEBASE="/bea_wls_internal/classes/" WIDTH=150 HEIGHT=25>  
</APPLET>
```

Web ブラウザは、<APPLET> タグを含む HTML ページを要求する場合、CODE 属性によって指定されているメイン アプレット クラスの検索を試みます。Web ブラウザは CODEBASE 属性によって指定された URL からそのクラスを要求します。アプレットが使用する他のクラスは、CODEBASE によって指定された URL から要求されます。

アプレットをテストするときには、Web ブラウザのクラスパスにアプレット クラスが指定されていないことに注意する必要があります。ブラウザが要求したクラスを HTTP サーバから取得できない場合、そのローカル パスを検索します。このため、アプレットのデプロイメントを適切にコンフィグレーションしたかのように感じられます。これは、アプレットがローカル ホスト マシン上で動作するからです。しかし、要求したアプレット クラスを Web サーバにすべてデプロイしていない場合、誰かがリモート クライアントからアプレットを使おうとしても、そのアプレットは実行できません。

## アプレットをいつ使用するか

BEA は、J2EE プラットフォームの一部としてサーバサイドアプリケーションを HTTP サブレットと JavaServer Pages (JSP) と一緒に使用することをサポートしています。新しいアプリケーションを開発する前に、サブレットまたは JSP を使用することをお勧めします。一般に、サブレットと JSP を使用する一連の対話型 Web ページを適切に作成すると、Web サイトの速度と信頼性が向上します。現在アプレットを使用している場合、Java Web Start を使用してそのほとんどを Java アプリケーションに変換し、引き続き WebLogic Server を使用できます。詳細については、Sun の Java Web Start サイトを参照してください。

アプレットは、WebLogic で実行されている分散アプリケーションの一部として、Web ブラウザのクライアントサイドインタフェースの対話性を高めるために使用できます。グラフィックの情報を時間の経過と共に更新する必要がある場合、アプレットは最良の方法です。アプレットには、ソフトウェアを配布することなく安全なクライアントサイド コードを実行できるというメリットがあります。

アプレットを使用して、ナビゲーション バーやコンソールなどのステートレスな、クリック / レスpons型アプリケーションや、株価表示などのポーリングアプリケーションで、ページの機能を拡張してもよいでしょう。

# WebLogic Server の Web プレゼンテーション – ベスト プラクティス

次の表に、WebLogic Server を使用するとき推奨される、Web プレゼンテーションの BEA ベスト プラクティスを示します。

目的とする作業	アプレットのメリット/デメリット	推奨される方法
ある期間にわたってデータを表示	グラフ、表、およびチャートを更新して現在の情報を表示できる。 Web ページのフレームを更新する。たとえば、Administration Console の階層情報が含まれる左フレームなど。	アプレット
データベースへの接続	Web ブラウザ JVM によって課せられる制限、および互換性を保持するためのコスト	サーブレットと JSP
スレッドの管理	スレッド ContextClassLoader の制限が、アプレット内で例外となることがある。	サーブレットと JSP
GUI の強化	アプレットは、GUI 強化の最適オプションであるが、ブラウザによってアプレットの処理が異なるので注意が必要。	アプリケーションと Java WebStart

アプレットおよび WebLogic Server でテストされたブラウザとプラグインの詳細については、プラットフォーム サポート ページの「WebLogic Server でのアプレットのブラウザ サポート」を参照してください。

## アプレットの使用 — 確認済みの制限

この節では、アプレットを使用する場合のデメリットと、アプレットを使用する場合の確認済みの制限 2 点について説明します。

### アプレット使用のデメリット

アプレットを使用する場合のデメリットを挙げると、以下のようになります。

- 1つのアプレットでは、大規模なアプリケーションの機能を扱えない。
- アプレットは一般にキャッシュすることができないので、アプレットを実行するたびにクラスを毎回ダウンロードするということになる。
- Java Plug-in を使用して有効になるキャッシングはバージョン管理を行わないので、アプレットに対する更新があっても無視される。
- ブラウザとブラウザのバージョンによって、さまざまに実装されるため、アプレットのパフォーマンスは良くない。
- 2つのウィンドウを開いてその両方でアプレットを実行し、しかもどちらも同じページ上にあると、パフォーマンスは良くない。

### ClassNotFoundException

イベント処理スレッドの `ContextClassloaders` は、ライフサイクルメソッドを実行しているスレッドの `ContextClassLoader` とは異なります。ライフサイクルメソッドを実行しているスレッドの `ContextClassLoader` にのみ、コードベースからロードするクラスに関する情報があります。

アプレット ライフサイクル メソッド (`init`、`start`、`stop`、`destroy`) を実行するスレッド以外のスレッドで `initialContext` を取得しようとする、以下のような状況で `ClassNotFoundException` が送出されることがあります。

- (`ActionListener` を実装する) アプレットの `actionPerformed()` メソッド内で、`initialContext` を取得しようとした場合
- アプレット内に `getInitialContext` メソッドがあり、このメソッドが `document.AppletName.getInitialContext()` などの JSP から呼び出されている場合

## ClassCastException

アプレット内の **Weblogic Server** クライアントがクラスローダからなんらかのリソース情報を取得しようとして、キャッシュ タグと `codebase=/bea_wls_internal/classes` タグを併用すると、`ClassCastException` が送出されることがあります。

次のようにして、この問題を回避します。

- コードベースとしてクラスパス サーブレットを使用している間は、`cache_option` や `cache_archive` などのキャッシュ オプションは使用しない。
- キャッシュ オプションの使用中は、コードベースとして、`/classes` (`ClasspathServlet`) のままにしない。これを行うには、アーカイブユーティリティを使って、クライアントサイドの **JAR** ファイルを初めにパッケージ化する。

この制限の詳細については、<http://developer.java.sun.com/developer/bugParade/bugs/4648591.html> を参照してください。

## Java Plug-in の使い方

**BEA** は、常にアプレット用の **Java Plug-in** を使用することをお勧めします。

**Sun** は、アプレットがブラウザのデフォルト仮想マシンではなく標準 **Java** 実行時環境内で動作するためのブラウザ プラグインを提供しています。このため、プラグインをサポートするブラウザで一貫性が保証されます。つまり、アプレットの互換性と信頼性が保証されます。また、プラグインを使用すると、クライアント マシンでどの **JRE** が使用されているのかを簡単に調べることができます。

**Java Plug-in** は、**WebLogic Server** と通信する必要があるアプレットに不可欠の互換性を実現します。ほとんどの場合、クライアントの **Java** 仮想マシン (**JVM**) のバージョンはサーバの **JVM** と一致する必要があります。このため、サーバで **Java 1.3** が実行されている場合、**Java 1.3 Plug-in** を使用する必要があります。

詳細については、Sun の Java Plug-in ホームページを参照してください。Java Plug-in は、Internet Explorer または Netscape ブラウザのネイティブ プラグインです。プラグインを必要とするページに最初にアクセスすると、メッセージが表示されて Sun の Web サイトに移動し、そこからプラグインがダウンロードされます。プラグインは、一度ダウンロードするだけで済みます。プラグインは Sun の特定の JRE の安定したリリース上でアプレットを実行しますが、それでもアプレットは通常のアプレットのようにブラウザ内で実行できます。

このプラグインを HTML ページに埋め込むのは、複雑な作業です。Internet Explorer と Netscape は異なる構文を使用するからです。Sun の Web サイトには、両方の構文フォーマットを同じ HTML ファイルに変換する方法が公開されています。また、既存の <APPLET> タグを自動的に変換する HTML コンバータをダウンロードできます。この解決策は非常に巧妙ですが、不自然で管理が困難です。このため、より優れた解決策として JavaServer Pages の使用を検討することをお勧めします。

JSP では、<jsp:plugin> タグを使用して、JSP によって生成された Web ページにアプレットを組み込みます。生成されたサーブレットは、クライアントの Web ブラウザのタイプを検出し、適切なプラグインタグを応答として送信します。詳細については、『WebLogic JSP プログラマーズガイド』を参照してください。

アプレット JVM の要件は、スタンドアロンのクライアント JVM の要件と同じです。WebLogic 6.1 サーバに対して 1.3 JVM 上でスタンドアロンのクライアントを実行する必要がある場合は、アプレットクライアントも 1.3 プラグイン上で実行する必要があります。

プラグイン対応に変換した後のアプレットは、次のようになります。

```
<HTML>
<HEAD><TITLE>Title of Applet page</TITLE></HEAD>
<BODY>
<OBJECT
CLASSID="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = 600
HEIGHT = 350
CODEBASE="http://java.sun.com/products/plugin/1.3/jinstall-13-win
32.cab#Version=1,3,0,0">
<PARAM NAME = CODE VALUE = "Applet1.class">
<PARAM NAME = CODEBASE VALUE =
"/bea_wls_internal/classes/DefaultWebApp@DefaultWebApp/">
<PARAM NAME = ARCHIVE VALUE = "weblogic.jar">
<PARAM NAME="type" VALUE="application/x-java-applet;version=1.3">
<PARAM NAME="scriptable" VALUE="false">
<COMMENT>
<EMBED type="application/x-java-applet;version=1.3"
```

```
CODE = "Applet1.class"
CODEBASE =
"/bea_wls_internal/classes/DefaultWebApp@DefaultWebApp/"
ARCHIVE = "weblogic.jar"
WIDTH = 600
HEIGHT = 350
scriptable=false
pluginspage="http://java.sun.com/products/plugin/1.3/plugin-install.html";>
<NOEMBED>
</COMMENT>
alt="Your browser understands the <APPLET> tag but isn't
running the applet, for some reason."
Your browser is completely ignoring the <APPLET> tag!
</NOEMBED>
</EMBED>
</OBJECT>
</BODY>
</HTML>
```

## CODEBASE 属性

<APPLET> タグで CODEBASE 属性を使用すると、アプレットの Java クラス ファイルの検索先となる URL を指定できます。CODEBASE タグがない場合、Web ブラウザは <APPLET> タグが埋め込まれている HTML ファイルと同じディレクトリ内で必要なクラスを検索します。CODEBASE を使用すると、サイトの HTML コンテンツとは別個に 1 つのディレクトリを作成し、そのディレクトリにクラス ファイルを格納できるようになります。

**注意：** コードベースは、WebLogic Server 6.1 SP3 の場合、/classes/ ではなく /bea\_wls\_internal/classes/ でなければなりません。

多くの場合、WebLogic Server と一緒に動作するアプレットでは WebLogic クラスが必要となります。このため、CODEBASE 属性を使用して、ブラウザが WebLogic から必要なクラスをロードできるようにすると便利です。WebLogic は、/classes にマップされる特別なサーブレットを自動的に提供します。このサーブレットは、WebLogic Server のクラスパスからクラスを提供します。このサーブレットは、仮想サーブレット名「classes」としてデフォルトで登録されています。CODEBASE を以下のような URL に設定したとします。

```
CODEBASE="http://www.weblogic.com/bea_wls_internal/classes/"
```

または

```
CODEBASE="/bea_wls_internal/classes/"
```

この場合、WebLogic Server はサーブレットを起動します。このサーブレットは、WebLogic Server のクラスパスから必要なクラスを検索します。

ClassPath Servlet を使って CLASSPATH からリソースを提供する方法については、『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』の「Web アプリケーション コンポーネントのコンフィグレーション」を参照してください。

CODEBASE=/bea\_wls\_internal/classes/ の場合には、アプレットに必要なクラスは、システム クラスパス内に存在していなければなりません。

CODEBASE=/bea\_wls\_internal/classes/DefaultWebApp@DefaultWebApp の場合には、アプレットに必要なクラスは、applications/DefaultWebApp/WEB-INF/classes ディレクトリまたはシステム クラスパスに存在していなければなりません。

## CODE 属性

<APPLET> タグには、メインアプレットクラス ファイルの完全なパッケージ名を指定する CODE 属性が含まれていなければなりません。CODE の最後の拡張子「.class」は省略可能です。たとえば、GraphApplet を使用する場合、<APPLET> タグは次のようになります。

```
<APPLET CODE="GraphApplet"
        CODEBASE="/bea_wls_internal/classes/appName@componentName"
>
```

ここで appName はアプリケーションの名前、componentName は Web アプリケーションの名前です。

<APPLET> タグと CODEBASE の詳細については、JavaSoft の Java チュートリアル「Overview of Applets」を参照してください。

## トラブルシューティングとパフォーマンス

以下のトピックでは、トラブルシューティングとパフォーマンスの問題について説明します。



## アプレットのトラブルシューティング

ここでは、アプレットを使用するときに直面するいくつかのシナリオを示します。

### アプレットがブラウザで動作しない

**WebLogic JDBC** をアプレットで使用して、**DBMS** からデータを取得しています。ローカルマシンで **Sun Appletviewer** を使用してクラスを実行する場合は、何の問題もありません。しかし、**Netscape** ブラウザでアプレットを実行しようとすると、アプレットに接続できません。

アプレットが **Appletviewer** で動作するのにブラウザでは動作しない場合、**Netscape** セキュリティ制限に違反している可能性があります。このような場合、アプレットはそのロード元以外のマシンに対するソケットを開くことができません。この問題を解決するには、**DBMS** のホストとなるマシンからアプレットコードが提供されるようにする必要があります。

**注意：** アプレットの **CODEBASE** で使用する IP 名フォーマットと **WebLogic Server** に接続するために使用する URL は正確に一致する必要があります。一方でドット表記を使用し、他方でドメイン名を使用することはできません。

### ClassFormatError

**ClassFormatError** を取得した場合、**HTTP** サーバのコンフィグレーションに問題がある場合があります。**WebLogic** またはアプレットクラスを **HTTP** サーバの適切なディレクトリに配置していないか、または **APPLET** タグ内の **CODEBASE** または **CODE** を間違えて指定している可能性があります。次に、例を 2 つ示します。

アプレットが **Web** アプリケーション **MyWar** に格納されている可能性があります。この **Web** アプリケーションがアプリケーション **MyEar** の一部である場合、**CODEBASE** は次のいずれかでなければなりません。

```
CODEBASE=http://host:port/bean_wls_internal/classes/MyEar@MyWar/
```

または

```
CODEBASE=/bea_wls_internal/classes/MyEar@MyWar/
```

これにより、*MyWar* Web アプリケーションからすべてのクラスファイルとリソースファイルがダウンロードされます。すべてのリソースファイル（JPG ファイルや JAR ファイルなど）を、特定の Web アプリケーションの `WebApplicationRoot`（この場合は *MyWar* のルート ディレクトリ）に保持する必要があります。

`CODE=com.myapp.MyApplet` であるアプレットで `CODEBASE` をテストする場合は、`http://server:host/CODEBASEvalue/com/myapp/MyApplet.class` のような URL を指定して、ブラウザ ウィンドウからアクセスしてみます。このクラスに対するダウンロード ウィンドウが表示されるはずですが、表示されない場合は、サーバで Web アプリケーションのコンフィグレーションを修正する必要があります。

詳細については、『WebLogic HTTP サーブレット プログラマーズ ガイド』を参照してください。

## ローカル環境でのテスト

WebLogic Server と Netscape Communicator 4.x を同じホスト上で実行する場合、Communicator を実行するシェルの環境から `CLASSPATH` を削除する必要があります。セキュリティ上の理由により、Netscape Communicator は標準クラスの悪意ある変更を避けるためにローカル `CLASSPATH` からクラスをロードしません。ブラウザの実行時にローカル `CLASSPATH` を削除すると、Netscape は WebLogic Server の `CLASSPATH` からクラスをロードします。

その場合でも、WebLogic を起動するシェルに `CLASSPATH` を設定する必要があります。WebLogic では、使用する環境に `CLASSPATH` を設定せず、WebLogic を実行するシェルに `CLASSPATH` を適切に設定することをお勧めします。

アプリケーション全体を開発する前に、アプレット上のプロトタイプのアプリケーションをテストすることをお勧めします。WebLogic Server 側では解決できない問題は、アプレットのプラグインへの依存性に起因するものなので、そのような問題についてテストすることをお勧めします。このテストは、次のような場合にお勧めします。

- アプレット内部でなんらかのセキュア プロトコルの使用を必要とするアプリケーション

- RMI コールバック オブジェクトを使用する特別な設計のアプレット
- 内部で JMS を使用するアプレット

## ローカル開発環境からの移動

アプレットをローカル環境から移動する場合、WebLogic クラスとアプレット クラスを Web サーバ上の適切な場所にインストールしたかどうかを確認する必要があります。

WebLogic 配布キットをインストールしたマシン上でアプレットを実行する場合、これによって CODEBASE に関する問題が隠されてしまう場合があります。アプレットは、最初にローカル CLASSPATH の WebLogic クラスを検索します。クラスを適切にインストールしなかったため、HTTP サーバからアプレットが提供されない場合でも、アプレットはデフォルトでローカル CLASSPATH を検索して動作するので、この問題は表面化しません。HTTP コンフィギュレーションを適切にテストするには、ローカル CLASSPATH の WebLogic クラス名を一時的に変更するか、アプレットを別のマシンからロードするようにします。

## アーカイブによるアプレットの高速化

WebLogic には、HTML サーバ ログをスキャンし、アプレットのクラスの zip ファイルを作成してファイルのダウンロードを高速化するためのユーティリティが用意されています。さらに高速な手段は、可能な限りアプレットで JDBC を使用せず、DBMS データをサブレットから HTML 形式で取得することです。サブレットは、アプレットに代わってクエリを実行し、ワークスペースからデータを取得して HTML として提供します。このデータを非同期に維持する WebLogic プロセスと連携することにより、アプリケーションのパフォーマンスが向上します。

アプレットが実行前に数多くのファイルをダウンロードしなければならない場合、HTML ページの APPLET タグの中で ARCHIVE パラメータを使用することで、これを高速化できます。複数のアプレットに関する典型的な問題は、ブラウザがアプレット内で使用されているファイルごとに別個の HTTP 接続を確立しなければならないことです。接続を確立するのに数秒かかる場合もあり、ファイル自体のダウンロード時間より長くなることもあります。ARCHIVE パラメータを使用

## 1 WebLogic Server でのアプレットの使用

---

すると、これらのクラスを 1 個の .jar ファイル (Microsoft Internet Explorer の場合は .cab ファイル) にまとめることができます。このファイルは、単一の HTTP 接続でダウンロードできます。JAR ファイルは圧縮可能なので (CAB ファイルは常に圧縮される)、ダウンロード時間がさらに短縮します。

**注意:** Appletviewer、Netscape Navigator (3.0 以降のみ)、および HotJava ブラウザを使用するときの手順は、Microsoft Internet Explorer (4.0 以降のみ) で使用する手順とは異なります。完全な互換性を実現するために、両方の方法を組み合わせることができます。