Solaris Java Plug-in ユーザーズガイド

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–3956–11
2003 年 4 月

原典: *Solaris Java Plug-in User's Guide*

Part No: 816-0378-11

Revision A

Adobe PostScript

030227@5533

目次

はじめに　**7**

1　概要 **— Java Plug-in** の紹介　**11**
Java Plug-in の紹介　11

2　従来の **APPLET** タグの使用　**13**
APPLET タグのサポート　13
OBJECT タグとの互換性　13
サポートするブラウザ　14
以前のクラスファイルの更新　14

3　**Java Plug-in** での **OBJECT**、**EMBED**、および **APPLET** タグの使用　**15**
はじめに　15
Microsoft Windows プラットフォームで稼動する Internet Explorer での Java Plug-in　16
Microsoft Windows や Linux プラットフォームまたは Solaris オペレーティング環境で稼働する Netscape Navigator での Java Plug-in　20
Internet Explorer および Netscape Navigator ブラウザでの Java Plug-in　23
環境に依存しない Java Plug-in　25
まとめ　33

4　**HTML Converter** を使用した **Java Plug-in** の **APPLET** タグの変換　**35**
はじめに　35
GUI バージョンの HTML Converter の実行　36
コマンド行からのコンバータの実行　42

5 プロキシ構成 **45**
はじめに 45
Java Plug-in がプロキシ情報を取得する方法 46
直接接続 47
手動プロキシ設定 47
自動プロキシ設定 48

6 プロトコルのサポート **51**
HTTP、FTP 、および Gopher 51
HTTPS 51
SOCKS 53

7 **cookie** のサポート **55**
はじめに 55
Java Plug-in による cookie のサポート方法 55
Java Plug-in での cookie ポリシーのサポート 57
cookie およびディスクキャッシュ 57
cookie およびセキュリティ 57

8 **Java Plug-in** でのアプレットキャッシュおよびインストール **59**
キャッシュオプション 59
キャッシュの更新アルゴリズム 61
セキュリティ 62
既知の制限 62

9 **Java Plug-in** コントロールパネルによる **Plug-in** の動作/オプションの設定 **63**
概要 63
Java Plug-in コントロールパネルの起動 63
保存オプション 64
コントロールパネルのオプション設定 64

10 **Netscape 6** **71**
`APPLET` タグ、`EMBED` タグ、および `OBJECT` タグのサポート 71
Java-JavaScript 間の双方向通信 71
RSA 署名付きアプレットの検証 71
Java コンソールの表示 72

Java プラットフォームの有効または無効　72
アプレットのライフサイクルの変更　72
プロキシおよび cookie のサポート　72
HTTPS　72
自動ダウンロード　72
下位互換性の問題　72

11　FAQ (よくある質問**)　73**

はじめに

本書は、Java™ Plug-in バージョン 1.4 の紹介、および使用方法の概略を示します。本書の内容は、http://java.sun.com/j2se/1.4/ja/docs/ja/guide/plugin から入手可能な Java Plug-in マニュアルでさらに詳しく解説されています。

対象読者

本書には、Web ページにアプレットを配備するアプレット開発者および Web サイト管理者にとって有用な情報が含まれています。

内容の紹介

第 1 章では、Java Plug-in 製品を紹介します。

第 2 章では、APPLET タグの使用について説明します。

第 3 章では、HTML ファイルで使用するタグについて説明します。

第 4 章では、HTML Converter について解説します。

第 5 章では、アプレットを配備する Web サイトの設定に関する情報を提供します。

第 6 章では、HTTP、FTP、および GOPHER プロトコルのサポートについて説明します。

第 7 章では、Java Plug-in での cookie のサポートについて解説します。cookie には、クライアントプラットフォームのデータが格納されます。

第 8 章には、アプレットのキャッシュに関係する詳細な情報が含まれます。

第 9 章では、Java Plug-in コントロールパネルの操作および使用方法を説明します。

第 10 章には、Netscape 6 および Open Java Interface (OJI) 関連の情報が含まれます。

第 11 章では、Java Plug-in のオンライン FAQ で扱われるさまざまなトピックへの参照情報を提供します。

関連マニュアル

以下に示すマニュアルには、Java 2 Platform バージョン 1.4 に関する情報が含まれています。

- *Java 2 SDK* 開発ガイド *(Solaris* 編*)*
- *Java 2 SDK, Standard Edition v. 1.4* リリースノート
  (http://java.sun.com/j2se/1.4 /ja/relnotes.html からオンラインで閲覧可能)
- *Java 2 SDK, Standard Edition, v. 1.4* ドキュメント
  (http://java.sun.com/j2se/1.4/ja/docs/ja/index.html からオンラインで閲覧可能)
- *Java 2 Platform, Standard Edition, v 1.4 API* 仕様
  (http://java.sun.com/j2se/1.4/ja/docs/ja/api/index.html からオンラインで閲覧可能)

Sun のオンラインマニュアル

docs.sun.com では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索を行うこともできます。URL は、`http://docs.sun.com` です。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 system%
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	system% **su** password:
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、rm *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	sun% **grep '^#define ** **XV_VERSION_STRING'**

コード例は次のように表示されます。

- C シェル

  machine_name% **command y|n** [*filename*]

- C シェルのスーパーユーザー

  machine_name# **command y|n** [*filename*]

- Bourne シェルおよび Korn シェル

  $ **command y|n** [*filename*]

- Bourne シェルおよび Korn シェルのスーパーユーザー

  # **command y|n** [*filename*]

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: `Shift` キーを押します)。ただし、キーボードによっては `Enter` キーが `Return` キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、`Ctrl-D` は `Control` キーを押したまま `D` キーを押すことを意味します。

一般規則

- このマニュアルでは、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が、適宜併記されています。
- このマニュアルでは、「x86」という用語は、Intel 32 ビット系列のマイクロプロセッサチップ、および AMD が提供する互換マイクロプロセッサチップを意味します。

第 1 章

概要 — Java Plug-in の紹介

Java Plug-in の紹介

Java™ Plug-in コンポーネント (以降、Java Plug-in) は、Web ブラウザの機能を拡張し、Web ブラウザに付属する Java 実行環境ではなく、Sun の Java 2 Runtime Environment でアプレットの実行を可能にします。Java Plug-in は、Java 2 Runtime Environment (JRE) の一部であるため、JRE をコンピュータにインストールする際、JRE といっしょにインストールされます。Microsoft Internet Explorer および Netscape™ の両方の Web ブラウザで動作します。

Java Plug-in の機能は、2 つの異なる方法で実現可能です。

1. Web ページで従来の APPLET タグを使用する。
2. Internet Explorer では、APPLET タグを OBJECT タグに置き換える。Netscape 4 では、APPLET タグを EMBED タグに置き換える。Netscape 6 では、APPLET タグを OBJECT タグまたは EMBED タグに置き換える。ただし、OBJECT および EMBED タグは、次の章に示す特別な形式に準拠している必要があります。

Web ページ内の `OBJECT` タグおよび `EMBED` タグは手動で更新することもできますが、Web ページの新規フォーマットへの更新作業を支援するため、HTML Converter が提供されています。HTML Converter については、第 4 章で説明します。

ここまでで、Java Plug-in の基本的な機能について説明しましたが、理解が必要な関連トピックはほかにも多くあります。たとえば、読者は Java Plug-in でプロキシ構成がどのように動作するのか、Java Plug-in がサポートするプロトコルは何か、あるいは cookie サポートやキャッシュについて、理解する必要を感じるかもしれません。本書は、これらのトピックについても説明します。

本書には、Netscape 6.1 や Open Java Interface (OJI) に関する情報、および FAQ も含まれています。

本書の情報の出典は、Java 2 Platform, Standard Edition, v1.4 マニュアルセットの一部である『*Java Plug-in* 開発者ガイド』です。完全な『*Java Plug-in* 開発者ガイド』は、http://java.sun.com/j2se/1.4/ja/docs/ja/guide/plugin/developer_guide/contents.html からオンラインで入手できます。オンラインの『*Java Plug-in* 開発者ガイド』には、本書の情報に加え、配備スキーマ、セキュリティ、デバッグサポート、および上級者向けトピックなどの章が含まれます。

第 2 章

従来の APPLET タグの使用

APPLET タグのサポート

Java Plug-in は、アプレットの起動用に HTML APPLET タグをサポートするようになりました。 ユーザは、ブラウザを設定することにより、APPLET タグを処理するために JRE 1.4.0 をデフォルトの実行環境にできます。

開発者は、Java Plug-in 拡張機能を使用することにより、HTML Converter や OBJECT タグを使用せずにアプレット Web ページを配備し、かつユーザが最新の JRE および Java Plug-in を使用してコードを実行することを保証できます。

現在、Java Plug-in は APPLET タグをサポートしますが、独自の技術を使用するアプレットはサポートしません。そのような技術には、以下が含まれます。

- CAB ファイル
- Authenticode 署名
- Java Moniker

OBJECT タグとの互換性

このリリースの Java Plug-in は、APPLET タグを使用したアプレットの起動をサポートします。ただし、以前の Java Plug-in リリースとの完全な下位互換性を保持するため、OBJECT タグを使用したアプレットの起動もサポートします。開発者は、HTML Converter を使用することにより、OBJECT タグを使用してアプレットの Web ページを従来通り設定できます。

サポートするブラウザ

Java Plug-in は、次のブラウザ上で APPLET タグをサポートします。

- Internet Explorer 4.0 (4.01 を推奨)、5.0 (5.01 を推奨)、5.5 (Service Pack 2 を推奨)、および 6.0
- Netscape 6.0、6.1 および 6.2

以前のクラスファイルの更新

この Java Plug-in リリースに関連した新規 Java Runtime Environment が、以前のコンパイラで生成されたクラスファイルを実行できないことがあります。この場合、この種のクラスファイルをロードしようとすると、仮想マシンが `java.lang.ClassFormatError` をスローするという現象がよく見られます。この障害は、このリリースの変更とは特に関係がありません。 原因は、以前のバイトコードコンパイラがクラスファイルを生成する際、ある状況下で適切なクラスファイル形式に厳密には準拠しないことです。最近の仮想マシン実装では、適切なクラスファイル形式が厳密に適用されるため、不適切な形式の以前のクラスファイルをロードしようとすると、エラーが発生することがあります。以前のクラスファイルでよく発生する問題の一部を、次に示します (この一覧にすべてが網羅されているわけではありません)。

- クラスファイルの末尾に余分なバイトが存在する
- クラスファイルに、文字以外で始まるメソッド名またはフィールド名が含まれる
- クラスが、別のクラスの private メンバにアクセスしようとする
- クラスファイルで、不正な定数プールインデックスおよび不正な UTF-8 文字列など、ほかの形式エラーが発生する
- 初期の (サードパーティ製) バイトコードオブファスケータには、適切なクラスファイル形式に違反するクラスファイルを生成するものがある

この種の問題は、最新の Java 2 SDK に付属の Javac バイトコードコンパイラでクラスを再コンパイルすることで回避できます。サードパーティ製のオブファスケータを使用する場合は、適切なクラスファイル形式のクラスファイルを生成するものを使用してください。

第 3 章

Java Plug-in での OBJECT、EMBED、および APPLET タグの使用

この章の内容は、次のとおりです。

- 15 ページの「はじめに」
- 16 ページの「Microsoft Windows プラットフォームで稼動する Internet Explorer での Java Plug-in」
- 20 ページの「Microsoft Windows や Linux プラットフォームまたは Solaris オペレーティング環境で稼働する Netscape Navigator での Java Plug-in」
- 23 ページの「Internet Explorer および Netscape Navigator ブラウザでの Java Plug-in」
- 25 ページの「環境に依存しない Java Plug-in」
- 33 ページの「まとめ」

はじめに

ここでは、OBJECT および EMBED タグなどの、Java Plug-in に必須のタグ構造について説明します。この章の内容は、HTML ページに手動で Java Plug-in タグを挿入する、Web ページの作成者を対象としています。

注 – Sun Microsystems が無料で提供している Java Plug-in HTML Converter を使用すると、HTML ファイルの変換を自動的に実行できます。 大半の Web 作成者にはこのツールの使用をお勧めします。

通常、アプレットは HTML ファイル内で次のように指定します。

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
        <PARAM name="model" value="models/HyaluronicAcid.xyz">
          No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
```

```
    </APPLET>
```

また、通常、APPLET タグにはアプレットに関する情報を指定し、<PARAM> タグにはインスタンスごとのアプレット情報を格納し、そのとき <PARAM> タグ は、<APPLET> と </APPLET> タグの間に配置します。

ただし、APPLET はブラウザにより描画されるため、ブラウザの動作を遮断して、アプレットの実行に Sun の Java Runtime Environment (JRE) を使用させる簡単な方法はありません。 次に示すように、HTML ページで通常の APPLET タグの代わりに、OBJECT タグと EMBED タグのいずれかまたは両方を使用して特別な Java Plug-in タグ構造を構築することにより、ブラウザにこの動作を実行させることが可能です。これにより、ブラウザが Java Plug-in を起動し、Java Plug-in が Sun の JRE を使用してアプレットを実行します。

以降の節では、特定のブラウザとプラットフォームの各組み合わせで必要な操作について説明します。

注 – 製品のバージョン番号は、次の形式になります。

```
n1.n2.n3_n4n5
```

ここで、n1.n2 はメジャーバージョン番号、n3 はマイナーバージョン番号 (保守バージョン番号とも呼ばれます)、および n4n5 は更新バージョン番号を表します。

以下に示す例では、バージョン番号として、メジャーリリース番号 1.4、マイナーリリース番号 0、および必要に応じて仮想的な更新番号を使用します。

1,4,0,nm や 14 など、バージョンの指定方法が多少異なる場合もあります (後者ではメジャーバージョン番号のみを指定します)。

Microsoft Windows プラットフォームで稼動する Internet Explorer での Java Plug-in

Microsoft Windows プラットフォームで稼動する Internet Explorer で Java Plug-in を使用するには、OBJECT タグを使用します。 以下に、APPLET タグを Java Plug-in タグにマッピングする方法を示します。

元の APPLET タグ

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
       <PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
          No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
       </APPLET>
```

注 – この例および以降の例の *codebase* 属性に示す URL は、単なるサンプルです。指定された URL には、cab ファイルは存在しません

新規 OBJECT タグ

```
<OBJECT classid="clsid:CAFEEFAC-0014-0000-0000-ABCDEFFEDCBA"
width="200" height="200" align="baseline"
codebase="http://java.sun.com/jpi/jinstall-14-win32.cab#Version=1,4,0,mn">
<PARAM NAME="code" VALUE="XYZApp.class">
<PARAM NAME="codebase" VALUE="html/">
<PARAM NAME="type" VALUE="application/x-java-applet;jpi-version=1.4">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="scriptable" VALUE="true">
        No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
       </OBJECT>
```

OBJECT タグには、APPLET タグと同様の情報が含まれることに注意してください。Internet Explorer で Java Plug-in を起動するには、これで十分です。 OBJECT タグ内の classid は、Java Plug-in 自体のクラス識別子です。このクラス識別子は、すべての HTML ページで同じである必要があります。Internet Explorer が OBJECT タグ内のこのクラス識別子を描画する場合、Java Plug-in をブラウザ内にロードしようとします。

OBJECT タグには、幅、高さ、位置合わせなどの属性が存在します。これらの属性は、APPLET タグの対応する属性から直接マッピングされます。Internet Explorer は、これらの属性に含まれる書式情報を使用して、Java Plug-in の位置を決定します。 この情報は、変更されることなく直接マッピングされるため、Java Plug-in を使用するアプレットと APPLET タグを使用するアプレットの位置および外観は同じはずです。

APPLET タグ内の属性をすべて、 OBJECT タグ属性にマッピングできるわけではありません。 たとえば、APPLET タグの *code* 属性および *codebase* 属性は、OBJECT タグ属性にはマッピングされません。 一方、*code* 属性は、 PARAM の *code* にマッピングされます。これは、HTML 仕様では、OBJECT タグに *code* 属性が存在しないためです。OBJECT タグ属性に対応しない属性は、ほかにも存在します。 これらの属性と 1 つの例外属性は、 PARAM タグにマッピングする必要があります。

注 – OBJECT タグでは、重複するパラメータ名を使用してはなりません。

ここで 1 つの例外となるのは、*codebase* 属性です。APPLET タグでは、*codebase* 属性は追加クラスおよび jar ファイルのダウンロード元の場所を表します。一方、OBJECT タグでは、*codebase* 属性は、ローカルマシンで Java Plug-in が検出されなかった場合の Java Plug-in のダウンロード元の場所を表します。 *codebase* 属性は、APPLET タグと OBJECT タグとで異なる意味を持つため、この属性を OBJECT タグの PARAM タグの *codebase* 属性にマッピングして、この競合を解決する必要があります。

上の例では、APPLET タグの *code* 属性および *codebase* 属性は、OBJECT タグのパラメータにマッピングされます。PARAM の *code* には、アプレットを指定し、この値は、APPLET タグ内の *code* 属性と同じにする必要があります。 PARAM の *codebase* には、アプレットの *codebase* を指定します。 Java Plug-in は、パラメータからアプレットのダウンロード先を読み取ることができます。 パラメータタイプは、APPLET タグからはマッピングされません。OBJECT タグにパラメータタイプを指定する必要があります。 Java 実行可能ファイルのタイプ (アプレットなど) を認識することにより、Java Plug-in は Java 実行可能ファイルの初期化方法を識別します。 上の例の 3 つの PARAM タグ (*code*、*codebase*、および *type*) は、Java Plug-in により指定されています。これらのタグは、元の APPLET タグの PARAM 内には存在しません。 OBJECT タグ内の model パラメータが、APPLET タグ内の model パラメータと同一であることに注意してください。 Java Plug-in で指定されている最初の 3 つのパラメータ (*code*、*codebase*、および *type*) を除く、残りのパラメータは、APPLET タグ内部のパラメータと同じです。

Java Plug-in 1.3 から、新たに PARAM *scriptable* タグが追加されました (Java Plug-in 1.4 でも有効)。 このタグの追加により、JavaScript や VBScript を必要としないアプレットのパフォーマンスが改善されます。 アプレットがスクリプト作成によるサポートを必要とする場合は値を *true* に、必要としない場合は *false* に設定します。 デフォルト値は、*false* です。

PARAM *scriptable* と PARAM *Mayscript* は同じではないことに注意してください。 *Mayscript* は Java アプレットから JavaScript への通信のみをサポートするのに対し、*scriptable* は Internet Explorer での JavaScript から Java アプレットへの通信のみをサポートします。

「No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!」というテキストは APPLET タグ内にあり、 <OBJECT> タグと </OBJECT> タグの間にマッピングされています。本来、これは、ブラウザが Java をサポートしない場合にのみ表示されるテキストです。これを OBJECT タグの内部にマッピングすると、ブラウザが OBJECT タグをサポートしない場合にこのテキストが表示されます。

APPLET タグと OBJECT タグの属性マッピングは次のようになります。

属性	**APPLET** タグのサポート	**OBJECT** タグのサポート	**OBJECT** タグの属性のマッピング先
ALIGN	O	O	ALIGN 属性
ALT	O		
ARCHIVE	O		archive パラメータ
CODE	O	O	code パラメータ
CODEBASE	O	O	codebase パラメータ
HEIGHT	O	O	HEIGHT 属性
HSPACE	O	O	HSPACE 属性

属性	**APPLET** タグのサポート	**OBJECT** タグのサポート	**OBJECT** タグの属性のマッピング先
NAME	O	O	NAME 属性、NAME パラメータ
OBJECT	O		object パラメータ
TITLE	O	O	TITLE 属性
VSPACE	O	O	VSPACE 属性
WIDTH	O	O	WIDTH 属性
MAYSCRIPT	O	O	MAYSCRIPT パラメータ

`OBJECT` タグに固有の属性もいくつかあります。`EMBED` タグに固有の属性は次のとおりです。

属性/パラメータ	**OBJECT** タグ内の意味
classid 属性	動的バージョンサポートの場合、常に同じ値 (clsid:8AD9C840-044E-11D1-B3E9-00805F499D93) を保持する。静的バージョンサポートの場合、バージョン固有の値 (たとえば、clsid:CAFEEFAC-0014-0000-0000-ABCDEFFEDCBA) を保持する 注:この節に示す例では、動的バージョンサポートを使用します。
codebase 属性	ネットワーク上の CAB ファイルを指す絶対 URL。デフォルトでは、Java Software Web サイトのバイナリを指す
type パラメータ	Java アプレットの場合、値は “application/x-java-applet;jpi-version=1.4” または “application/x-java-applet”。JavaBeans コンポーネントの場合、値は “application/x-java-bean;jpi-version=1.4” または “application/x-java-bean”
codebase パラメータ	アプレットのベース URL を指定する。 この属性は省略可能
code パラメータ	Java アプレットまたは JavaBeans コンポーネントの名前を指定する。同じ `OBJECT` タグの内部で `object` パラメータとともに使用することはできない
scriptable パラメータ	JavaScript または VBScript を使用して、HTML ページからアプレットをスクリプト処理できるかどうかを指定する。*true* または *false* を指定できる。この属性は Java Plug-in 1.4 で新たに導入された

属性/パラメータ	**OBJECT** タグ内の意味
object パラメータ	直列化された Java アプレットまたは JavaBeans コンポーネントの名前を指定する。同じ OBJECT タグの内部で code パラメータとともに使用することはできない。この属性は省略可能
archive パラメータ	Java アーカイブの名前を指定する。この属性は省略可能
mayscript パラメータ	アプレットに netscape.javascript.JSObject へのアクセスを許可するかどうかを指定する。*true* または *false* を指定できる。この属性は省略可能

元の APPLET タグが PARAM *type*、 *codebase*、*code*、*object*、または *archive* を保持する場合、パラメータ名が重複するため、OBJECT タグへのマッピングで問題が発生します。 これを回避するため、Java Plug-in は次のパラメータ名セットも別途サポートします。

元のパラメータ名	新しいパラメータ名
code	java_code
codebase	java_codebase
archive	java_archive
object	java_object
type	java_type

これらの新しいパラメータ名は、必要な場合にのみ使用してください。1 つの OBJECT タグに新しいパラメータ名と元のパラメータ名の両方が存在する場合、Java Plug-in は常に新しいパラメータ名に関連付けられた値を使用してアプレットまたは JavaBean をロードします。

Microsoft Windows や Linux プラットフォームまたは Solaris™ オペレーティング環境で稼働する Netscape Navigator での Java Plug-in

Microsoft Windows や Linux プラットフォーム、または Solaris™ オペレーティング環境で稼働する Netscape Navigator™ 4 ブラウザで Java Plug-in を使用する場合は、EMBED タグを使用します。次の例は、APPLET タグを Java Plug-in の EMBED タグにマッピングする方法を示します。

元の APPLET タグ

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
```

```
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
          No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</APPLET>
```

新規 *EMBED* タグ

```
<EMBED type="application/x-java-applet;jpi-version=1.4" width="200"
        height="200" align="baseline" code="XYZApp.class"
        codebase="html/" model="models/HyaluronicAcid.xyz"
        pluginspage="http://java.sun.com/jpi/plugin-install.html">
<NOEMBED>
No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</NOEMBED>
</EMBED>
```

EMBED タグには APPLET タグと同様の情報が含まれることに注目してください。Netscape Navigator ブラウザで Java Plug-in を起動するには、この情報で十分です。EMBED タグ内の type 属性は、Java プログラミング言語実行可能ファイルの種類 (アプレットや Bean など) の識別に使用されます。Netscape Navigator ブラウザが EMBED タグ内のこの属性を描画するときに、Java Plug-in をブラウザへロードしようとします。type 属性を指定することにより、Java Plug-in は Java プログラミング言語の実行可能ファイルの初期化方法を識別します。

上の例では、EMBED タグ内のいくつかの属性 (width、height、align など) が APPLET タグの対応する属性から直接マッピングされています。Netscape Navigator ブラウザは、これらの属性に含まれる書式情報を使用して、Java Plug-in の位置を決定します。この情報は、変更されることなく直接マッピングされるため、Java Plug-in を使用するアプレットと APPLET タグを使用するアプレットの位置および外観は同じはずです。

OBJECT タグとは異なり、PARAM を使用せずに、すべての情報を <EMBED> タグ内に格納する必要があります。このため、APPLET タグ内のすべての属性およびパラメータを、EMBED タグ内部で属性と値のペアとしてマッピングする必要があります。

上の例では、APPLET タグ内の *code* 属性および *codebase* 属性は、EMBED タグ属性にマッピングされています。*code* は、アプレットを識別する属性です。この値は、APPLET タグの *code* 属性と同じにする必要があります。*codebase* 属性は、アプレットのコードベースを識別します。Java Plug-in は、この属性から情報を読み取ることで、アプレットや JavaBeans コンポーネントのダウンロード先を識別します。EMBED タグ内部の model 属性が、 APPLET タグ内部の model パラメータからマッピングされている点にも注目してください。

OBJECT タグの *codebase* 属性と同様、Java Plug-in がインストールされていない場合、Netscape Navigator ブラウザは EMBED タグの *pluginspage* 属性を使用します。この属性からは、Java Software Web サイトの「Java Plug-in Download Page」を常に参照する必要があります。

「No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!」というテキストは APPLET タグ内にあり、 <NOEMBED> タグと </NOEMBED> タグの間にマッピングされています。 本来、これは、ブラウザが Java テクノロジをサポートし

ない場合にのみ表示されるテキストです。このテキストを NOEMBED タグ内にマッピングすると、ブラウザが EMBED タグをサポートしていないか、またはブラウザが Java Plug-in の起動に失敗した場合に、このテキストが表示されます。

APPLET タグと EMBED タグの属性のマッピングは次のようになります。

属性	**APPLET** タグのサポート	**EMBED** タグのサポート	**EMBED** タグの属性のマッピング先
ALIGN	O	O	ALIGN 属性
ALT	O	O	ALT 属性
ARCHIVE	O		archive 属性
CODE	O		code 属性
CODEBASE	O		codebase 属性
HEIGHT	O	O	HEIGHT 属性
HSPACE	O	O	HSPACE 属性
NAME	O	O	NAME 属性
OBJECT	O		object 属性
TITLE	O	O	TITLE 属性
VSPACE	O	O	VSPACE 属性
WIDTH	O	O	WIDTH 属性
MAYSCRIPT	O	O	MAYSCRIPT 属性

EMBED タグに固有の属性もいくつかあります。EMBED タグに固有の属性は次のとおりです。

属性	**EMBED** タグ内の意味
type 属性	アプレットの場合、値は "application/x-java-applet;jpi-version=1.4" または "application/x-java-applet" を指定する。Bean の場合、値は "application/x-java-bean;jpi-version=1.4" または "application/x-java-bean" を指定する
codebase 属性	アプレットのベース URL を指定する。 この属性は省略可能
code 属性	Java アプレットまたは JavaBeans コンポーネントの名前を指定する。同じ EMBED タグの内部で object パラメータとともに使用することはできない

属性	**EMBED** タグ内の意味
object 属性	直列化された Java アプレットまたは JavaBeans コンポーネントの名前を指定する。EMBED タグの内部で code パラメータとともに使用することはできまない。この属性は省略可能
archive 属性	Java アーカイブの名前を指定する。この属性は省略可能
pluginspage 属性	ネットワーク上の HTML ページを指す絶対 URL
mayscript 属性	アプレットに netscape.javascript.JSObject へのアクセスを許可するかどうかを指定する。true または false を指定できる。この属性は省略可能

OBJECT タグの場合と同様、元の APPLET タグに PARAM *type*、*codebase*、*code*、*object*、または *archive* が含まれる場合、これを EMBED タグ属性にマッピングすると問題が発生します。これを回避するため、Java Plug-in は次の属性名セットも新たにサポートしています。

元の属性名	新しい属性名
code	java_code
codebase	java_codebase
archive	java_archive
object	java_object
type	java_type

これらの新しい属性名は、必要な場合にのみ使用してください。1 つの EMBED タグに新しい属性名と元の属性名の両方が存在する場合、Java Plug-in は、常に新しい属性名に関連付けられた値を使用して、アプレットまたは Bean をロードします。

Internet Explorer および Netscape Navigator ブラウザでの Java Plug-in

Microsoft Windows プラットフォームまたは Solaris オペレーティング環境で HTML ページをブラウズする場合、Internet Explorer で OBJECT タグを、Netscape Navigator で EMBED タグを指定すると、HTML で Java Plug-in を使用できます。ただし、HTML ページがインターネットまたはイントラネット上に存在する場合、ページが Internet Explorer と Netscape Navigator ブラウザの両方でブラウズされる可能性が

あります。Netscape Navigator と Internet Explorer の両方で同じ HTML ページをブラウズする場合は、Java Plug-in を起動する必要があります。この操作を実行するには、Java Plug-in の OBJECT タグを次のように指定します。

元の *APPLET タグ*

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
No Java 2 SDK, Standard Editoin v 1.4 support for APPLET!!
</APPLET>
```

新規 *OBJECT タグ*

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="200" height="200" align="baseline"
codebase="http://java.sun.com/jpi/jinstall-14-win32.cab#Version=1,4,0,mn">
<PARAM NAME="code" VALUE="XYZApp.class">
<PARAM NAME="codebase" VALUE="html/">
<PARAM NAME="type" VALUE="application/x-java-applet;jpi-version=1.4">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="scriptable" VALUE="true">
<COMMENT>
<EMBED type="application/x-java-applet;jpi-version=1.4" width="200"
    height="200" align="baseline" code="XYZApp.class"
    codebase="html/" model="models/HyaluronicAcid.xyz"
    pluginspage="http://java.sun.com/jpi/plugin-install.html">
<NOEMBED>
</COMMENT>
        No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</NOMEMBED></EMBED>
</OBJECT>
```

Internet Explorer は <OBJECT> タグを認識するため、Java Plug-in の起動を試みます。<COMMENT> タグは Internet Explorer のみが理解できる特殊な HTML タグです。Internet Explorer は、<COMMENT> タグと </COMMENT> タグの間にあるテキストを無視します。上記のタグは、実際には次のようになります。

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="200" height="200" align="baseline"
codebase="http://java.sun.com/jpi/jinstall-14-win32.cab#Version=1,4,0,mn">
<PARAM NAME="code" VALUE="XYZApp.class">
<PARAM NAME="codebase" VALUE="html/">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="type" VALUE="application/x-java-applet;jpi-version=1.4">
<PARAM NAME="scriptable" VALUE="true">
   No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</NOEMBED></EMBED>
</OBJECT>
```

これは、上述の OBJECT タグの例と同じになります。 </NOEMBED> タグおよび </EMBED> タグは、対応する <NOEMBED> タグおよび <EMBED> タグが存在しないため、OBJECT タグにより無視されます。

Netscape Navigator ブラウザは OBJECT および COMMENT タグを理解しないため、上記のタグは次のように認識されます。

```
<EMBED type="application/x-java-applet;jpi-version=1.4"
    width="200" height="200"
    align="baseline" code="XYZApp.class" codebase="html/"
    model="models/HyaluronicAcid.xyz"
    pluginspage="http://java.sun.com/jpi/plugin-install.html">
<NOEMBED>
    No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</NOEMBED>
</EMBED>
```

これは、上述の EMBED タグの例と同じになります。<OBJECT> タグおよび <COMMENT> タグは、Internet Explorer ブラウザのみで理解される HTML 拡張機能であるため、Netscape Navigator ブラウザはこれらのタグを無視します。

この例は、Internet Explorer と Netscape Navigator のどちらが使用される場合でも、OBJECT タグおよび EMBED タグを組み合わせて使用することにより、ブラウザ内で Java Plug-in をアクティブにできることを示しています。すべてのユーザが同機種環境で HTML ページをブラウズするのでない限り、この複合タグを使用することが強く推奨されています。Sun Microsystems の提供する Java Plug-in HTML Converter を使用すると、HTML ページをこのタグの書式に自動的に変換できます。

環境に依存しない Java Plug-in

インターネット/イントラネット環境では、HTML ページを表示するブラウザのプラットフォームが異なることが考えられます。Java Plug-in を起動するのは、ブラウザとプラットフォームの組み合わせが適切な場合だけに限定する必要があります。組み合わせが適切でない場合は、ブラウザのデフォルト JVM を使用してください。この操作の実行には、次の Java Plug-in タグを使用します。

元の *APPLET* タグ

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
          No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</APPLET>
```

新しい書式のタグ

次に、上記のタグと等価な Java Plug-in タグの例を示します。この例には、コメントも含まれています。

```
<!-- The following code is specified at the beginning of the <BODY> tag. -->
<SCRIPT LANGUAGE="JavaScript"><!--
 var _info = navigator.userAgent; var _ns = false;
      var _ie = (_info.indexOf("MSIE")> 0 && _info.indexOf("Win")> 0
          && _info.indexOf("Windows 3.1") < 0);
  //--></SCRIPT>
```

```
<COMMENT><SCRIPT LANGUAGE="JavaScript1.1"><!--
var _ns = (navigator.appName.indexOf("Netscape")>= 0
    && ((_info.indexOf("Win")> 0 && _info.indexOf("Win16") < 0
    && java.lang.System.getProperty("os.version").indexOf("3.5") < 0)
    || _info.indexOf("Sun")> 0));
  //--></SCRIPT></COMMENT>

<!-- The following code is repeated for each APPLET tag -->
<SCRIPT LANGUAGE="JavaScript"><!--
      if (_ie == true) document.writeln('
<OBJECT
classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="200" height="200" align="baseline"
codebase="http://java.sun.com/jpi/jinstall-14-win32.cab#Version=1,4,0,mn">
      <NOEMBED><XMP>');
      else if (_ns == true) document.writeln('
<EMBED
      type="application/x-java-applet;jpi-version=1.4"
      width="200" height="200"
      align="baseline" code="XYZApp.class" codebase="html/"
      model="models/HyaluronicAcid.xyz"
      pluginspage="http://java.sun.com/jpi/plugin-install.html">
      <NOEMBED><XMP>');
//--></SCRIPT>
      <APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
</XMP>
<PARAM NAME="java_code" VALUE="XYZApp.class">
<PARAM NAME="java_codebase" VALUE="html/">
<PARAM NAME="java_type" VALUE="application/x-java-applet;jpi-version=1.4">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="scriptable" VALUE="true">
      No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</APPLET></NOEMBED></EMBED>
</OBJECT>

<!--
      <APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
      <PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
      No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
      </APPLET>
-->
```

このタグは元の APPLET タグと比較して複雑に見えますが、実際にはそうではありません。Java Plug-in タグのコードの大部分は、使用されるアプレットに関係なく同じです。Web ページの作成者は、たいていの場合、Java Plug-in タグをコピー&ペーストできます。

スクリプトの最初のブロックで、ブラウザおよびプラットフォームを抽出します。アプレットを実行するブラウザおよびプラットフォームを判別する必要があります。 これは、JavaScript™ を使用して、まずブラウザ名、次にプラットフォームを抽出することで実行します。HTML ドキュメントごとに、この操作を 1 回実行します。

スクリプトの 2 番目のブロックで、APPLET タグの置換を実行します。各 APPLET タグを類似のコードブロックで置き換える必要があります。スクリプトは、ブラウザに応じて、APPLET タグを EMBED タグまたは OBJECT タグに置き換えます。Internet Explorer ブラウザの場合は OBJECT タグ、Netscape Navigator ブラウザの場合は EMBED タグに置き換えます。最後に、元の APPLET タグをコメントとして末尾に含めます。Java Plug-in の起動を解除することを考えて、元の APPLET タグを残しておくことをお勧めします。

最初の JavaScript で、ブラウザおよびそのブラウザが実行されているプラットフォームを確立します。現在のところ、Java Plug-in は Microsoft Windows プラットフォームおよび Solaris オペレーティング環境のみをサポートするため、この操作が必要になります。Java Plug-in がサポートしない Win32 プラットフォームは、Windows NT 3.51 だけです。Java Plug-in の起動は、サポートされているブラウザおよびプラットフォーム上でのみ実行する必要があります。ブラウザが Internet Explorer の場合、スクリプトにより変数 *_ie* が true に設定されます。Netscape ブラウザの場合、変数 *_ns* が true に設定されます。(JavaScript では、変数名はすべて "_" で始まることに注意してください。これは、同一ページのほかの JavaScript 変数との競合を避けるためです。)

適切なブラウザを検出するため、JavaScript は JavaScript の navigator オブジェクト内の次の 3 つの文字列を評価します。*userAgent* , *appVersion*, および *appName* 。これらの文字列には、ブラウザおよびプラットフォームに関する情報が含まれます。次に示す、文字列 *userAgent* のいくつかの例を参考にすることにより、*userAgent* を評価してブラウザを判別する方法を理解できます。以下に示すのは、Internet Explorer 内で使用されるさまざまなプラットフォーム用の *userAgent* 文字列の例です。この表では、IE は Internet Explorer を表します。

プラットフォームとブラウザ	**JavaScript** の *Navigator.userAgent* 文字列
Windows 2000 と IE 5.0	"Mozilla/4.0 (compatible; MSIE 5.01; Window NT5.0)
Windows NT 4.0 と IE 4.0	"Mozilla/4.0 (compatible; MSIE 4.0; Windows NT)" “
Windows 95 と IE 4.0	"Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)“
Windows NT 3.51 と IE 4.0	"Mozilla/4.0 (compatible; MSIE 4.0; Windows 3.1)“
Windows 3.1 と IE 4.0	"Mozilla/4.0 (compatible; MSIE 4.0; Windows 3.1)“
Windows NT 4.0 と Navigator 4.04	Mozilla/4.04 [en] (WinNT; I) I)“
Windows NT 3.51 と Navigator 4.04	Mozilla/4.04 [en] (WinNT; I) I)“
Windows 95 と Navigator 4.03	Mozilla/4.03 [en] (Win95; I) I)“

プラットフォームとブラウザ	**JavaScript** の *Navigator.userAgent* 文字列
Solaris 2.6 と Navigator 4.02	"Mozilla/4.02 [en] (X11; I; SunOS 5.6 sun4u)

どの場合でも、Internet Explorer 内の文字列 *userAgent* に、部分文字列 "MSIE" が常に含まれます。また、Windows 3.1 および Windows 3.51 で稼働する Internet Explorer は 16 ビットであるため、これらのプラットフォーム上の Internet Explorer の *userAgent* 文字列には、部分文字列 "Windows 3.1" が含まれます。Internet Explorer 4 は Macintosh および Solaris オペレーティング環境でも利用可能であるため、これらのバージョンでは *userAgent* 文字列に部分文字列 "Win" は含まれません。また、Windows CE 上の Internet Explorer は JavaScript をサポートしません。これらをまとめると、次のようになります。

userAgent 文字列/ブラウザ	**"MSIE"** を含む	**"Win"** を含む	**"Windows 3.1"** を含まない
Windows 3.1 と IE 4	O	O	
Windows NT 3.51 と IE 4	O	O	
Windows 95 と IE 4	O	O	O
Windows NT 4.0 と IE 4	O	O	O
Windows CE と IE			
Mac と IE	O		O
UNIX と IE	O		O
任意のプラットフォームのほかのブラウザ			O

上の表から、Windows 95 および Windows NT 4.0 で稼働する Internet Explorer だけが Java Plug-in のブラウザおよびプラットフォーム要件を満たすことがわかります。ただし、ここでは、今後の Internet Explorer のリリースや Internet Explorer を搭載する Microsoft Windows のリリースのことは考慮に入れていません。*userAgent* 文字列に "MSIE" および "Win" が含まれる限り、上記のコードはWin32 用 Internet Explorer の将来のリリースで動作するはずです。

このロジックをまとめると、次のようになります。

```
<SCRIPT LANGUAGE="JavaScript">
          var _info = navigator.userAgent;
          var _ie = (_info.indexOf("MSIE")> 0 && _info.indexOf("Win")> 0
                    && _info.indexOf("Windows 3.1") < 0);
//--></SCRIPT>
```

Navigator が動作しているプラットフォームが適切であることを検出するのは、もっと難しい問題です。JavaScript を使用するだけでは、ブラウザが稼働しているオペレーティングプラットフォームが、Windows NT 3.51 と Windows NT 4.0 のどちらかなのかを判別することはできません。(上記の表を参照して、文字列 *userAgent* を確認してください。Navigator ブラウザでは、Windows NT 3.51 と Windows NT 4.0 オペレーティングプラットフォームの文字列 *userAgent* が同じになります。)Java Plug-in は Windows NT 4.0 オペレーティングプラットフォームだけをサポートするため、この違いを識別することは重要です。適切なプラットフォームで Java Plug-in を実行するには、Netscape Navigator ブラウザで LiveConnect を使用して OS のバージョン番号を判別する必要があります。これらをまとめると、次のようになります。

ロジック検査/ブラウザ	**appName** に **"Netscape"** を含む	**userAgent** に **"Win"** あり	**userAgent** に **"Win16"** を含まない	**os.version** に **"3.5"** を含まない	**userAgent** に **"Sun"** を含む
Windows 3.1 と Netscape 4	O	O		O	
Windows NT 3.51 と Netscape 4	O	O	O		
Windows 95 と Netscape 4	O	O	O	O	
Windows NT 4.0 と NS 4	O	O	O	O	
Solaris と NS	O				O
Solaris と IE					O
ほかのプラットフォームの Netscape	O		O	OS に依存	
ほかのプラットフォームの IE		O	O		
任意のプラットフォームのほかのブラウザ					

上記のロジックは、次のコードで表されます。

```
<SCRIPT LANGUAGE="JavaScript">
         var _info = navigator.userAgent; var _ns = false;
//--></SCRIPT>
<COMMENT><SCRIPT LANGUAGE="JavaScript1.1">
    var _ns = (navigator.appName.indexOf("Netscape")>= 0
        && ((_info.indexOf("Win")> 0  && _info.indexOf("Win16") < 0
```

```
        && java.lang.System.getProperty("os.version").indexOf("3.5") < 0)
        || _info.indexOf("Sun")> 0));
//--></SCRIPT></COMMENT>
```

上記の表で、すべての条件を満たすのは Windows 95、Windows NT 4.0、および Solaris オペレーティング環境の Netscape Navigator ブラウザだけであることに注目してください。LiveConnect を使用して OS のバージョン番号を取得しますが、LiveConnect は Netscape Navigator ブラウザだけがサポートしているため、Internet Explorer では LiveConnect を使用する JavaScript は認識されません。このことで問題が発生するのを防ぐため、スクリプトのこの部分を Internet Explorer 固有のコメントタグである COMMENT タグで囲む必要があります。Internet Explorer は COMMENT タグ内のテキストを無視しますが、Netscape Navigator は無視しません。さらに、ブラウザが Netscape Navigator 2 の場合には、スクリプト言語を JavaScript1.1 と指定することにより、このテキストが無視されるようにする必要があります。

ここで、Internet Explorer と Netscape Navigator に対応した上記のロジックをまとめると、次のようなスクリプトになります。

```
<!-- The following code is specified at the beginning of the <BODY> tag. -->
<SCRIPT LANGUAGE="JavaScript"><!--
    var _info = navigator.userAgent; var _ns = false;
    var _ie = (_info.indexOf("MSIE")> 0 && _info.indexOf("Win")> 0
                && _info.indexOf("Windows 3.1") < 0);
//--></SCRIPT>
<COMMENT><SCRIPT LANGUAGE="JavaScript1.1"><!--
    var _ns = (navigator.appName.indexOf("Netscape")>= 0
        && ((_info.indexOf("Win")> 0 && _info.indexOf("Win16") < 0
        && java.lang.System.getProperty("os.version").indexOf("3.5") < 0
        || _info.indexOf("Sun")> 0));
//--></SCRIPT></COMMENT>
```

この JavaScript ブロックは、HTML ファイルの <BODY> 部の先頭に配置する必要があります。先頭に配置することにより、ほかの JavaScript が変数 *_ie* および *_ns* を参照できるようになります。この JavaScript はどの HTML ファイルでも同一であり、各 HTML 本体に一度だけ指定します。

HTML タグの第 2 ブロックは、APPLET タグ内のデータからマッピングされた、対応する OBJECT タグおよび EMBED タグです。ブラウザが Windows 95、Windows 98、または Windows NT 4.0 オペレーティング環境で稼働する Internet Explorer の場合、JavaScript は OBJECT タグを出力します。ブラウザが Windows 95、Windows 98、Windows NT 4.0、または Solaris オペレーティング環境で稼働する Netscape Navigator 4 ブラウザの場合、JavaScript は (構文は多少異なりますが) EMBED タグも出力します。ブラウザおよびプラットフォームの検出機構は、すでに説明したとおりです。(<!— タグおよび –> タグは HTML 内でコメントの記述に使用されます。)

元の *APPLET タグ*

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
          No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</APPLET>
```

新しい書式のタグ

```
<SCRIPT LANGUAGE="JavaScript"><!--
if (_ie == true) document.writeln('<OBJECT
classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="200" height="200" align="baseline"
codebase="http://java.sun.com/jpi/jinstall-14-win32.cab#Version=1,4,0,mn">
<NOEMBED><XMP> ');
    else if (_ns == true) document.writeln('<EMBED
    type="application/x-java-applet;version=1.4" width="200" height="200"
      align="baseline" code="XYZApp.class" codebase="html/"
      model="models/HyaluronicAcid.xyz"
      pluginspage="http://java.sun.com/jpi/plugin-install.html">
<NOEMBED><XMP>');
//--></SCRIPT>
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200"></XMP>
<PARAM NAME="java_code" VALUE="XYZApp.class">
<PARAM NAME="java_codebase" VALUE="html/">
<PARAM NAME="java_type" VALUE="application/x-java-applet;version=1.4">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
          LUE="models/HyaluronicAcid.xyz">
<PARAM NAME="scriptable" VALUE="true">
          No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</APPLET></NOEMBED></EMBED> </OBJECT>
```

新しい Java Plug-in タグでは、元の APPLET タグもマッピングされていることに注目してください。これは、サポートされるプラットフォームでのみ Java Plug-in を使用するためです。スクリプト内に APPLET タグを残しておくことにより、Java Plug-in をサポートしないブラウザや、JavaScript をサポートしないブラウザがデフォルトの JVM を使用するアプレットを適切に処理できます。Java Plug-in をサポートしないプラットフォームの HotJava Browser、Internet Explorer、および Netscape Navigator ブラウザ、または JavaScript をサポートしないブラウザは、上述のタグを次のように読み取ります。

```
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200"></XMP>
<PARAM NAME="java_code" VALUE="XYZApp.class">
<PARAM NAME="java_codebase" VALUE="html/">
<PPARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="scriptable" VALUE="true">
         No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
<PARAM NAME="java_type" VALUE="application/x-java-applet;jpi-version=1.4">
</APPLET></NOEMBED></EMBED></OBJECT>
```

これらのブラウザは、</XMP>、 </OBJECT>、</EMBED>、および </NOEMBED> タグも無視します。これは、対応する <XMP>、<OBJECT>、<EMBED>、および <NOEMBED> タグが存在しないためです。Java Plug-in は、Java 2 SDK, Standard Edition v 1.4 以降のリリースの機能を対象にして設計されています。そのため、Java 2 SDK 1.4 を完全にはサポートせず、Java Plug-in をサポートしないこれらのブラウザでは、「No Java 2 SDK, Standard Edition v 1.4 support for APPLET」というメッセージが表示されます。

これまで示した例とは異なり、マッピングされたパラメータ 名には、*code*、*codebase*、および *type* ではなく、 *java_code*、*java_codebase*、および *java_type* が含まれます。これが必要なのは、<APPLET> タグと </APPLET> タグの間の <PARAM> 内で code および codebase を指定すると、一部のブラウザで問題が発生するためです。

Windows 95、Windows 98、または Windows NT 4.0 で稼働する Internet Explorer は、タグを次のように読み取ります。

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="200" height="200" align="baseline"
codebase="http://java.sun.com/jpi/jinstall-14-win32.cab#Version=1,4,0,mn">
<NOEMBED><XMP>
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
          width="200" height="200"></XMP>
<PARAM NAME="java_code" VALUE="XYZApp.class">
<PARAM NAME="java_codebase" VALUE="html/">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="java_type" VALUE="application/x-java-applet;jpi-version=1.4">
<PARAM NAME="scriptable" VALUE="true">
          No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</APPLET></NOEMBED></EMBED>
</OBJECT>
```

<XMP> タグを使用する場合には注意が必要です。Internet Explorer は <OBJECT> タグを認識するため、<APPLET> タグを無効にする必要があります。無効にしない場合、2 つのアプレットがブラウザ内で同時に表示されます。1 つは Microsoft の JVM で動作し、もう 1 つは Java Plug-in を使用する Sun の JVM で動作します。<XMP> タグを使用することで、この問題を解決できます。<XMP> および </XMP> タグの基本動作は、その間に記述されたすべての HTML タグを静的テキストのストリームに変換することです。上の例では、<XMP> および </XMP> タグを指定することにより、ブラウザが <APPLET> タグを HTML タグとしてではなく静的テキストとして処理します。ブラウザは <OBJECT> タグと <PARAM> タグの間の静的テキストをすべて無視するため、上のタグは実質的に次のようになります。

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="200" height="200" align="baseline"
codebase="http://java.sun.com/jpi/jinstall-14-win32.cab#Version=1,4,0,mn">
<PARAM NAME="java_code" VALUE="XYZApp.class">
<PARAM NAME="java_codebase" VALUE="html/">
<PARAM NAME="java_type" VALUE="application/x-java-applet;version=1.4">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="scriptable" VALUE="true">
   No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</OBJECT>
```

これは、上に示した OBJECT タグの例と同じになります。<OBJECT> タグは <NOEMBED>、</NOEMBED>、および <EMBED> タグを無視することに注意してください。

Windows 95、Windows 98、または Windows NT 4.0 オペレーティング環境で動作する Netscape Navigator 4 は、タグを次のように読み取ります。

```
<EMBED type="application/x-java-applet;jpi-version=1.4"
         width="200" height="200"
         align="baseline" code="XYZApp.class" codebase="html/"
         model="models/HyaluronicAcid.xyz"
         pluginspage="http://java.sun.com/jpi/plugin-install.html">
<NOEMBED><XMP>
<APPLET code="XYZApp.class" codebase="html/" align="baseline"
         width="200" height="200"></XMP>
<PARAM NAME="java_code" VALUE="XYZApp.class">
<PARAM NAME="java_codebase" VALUE="html/">
<PARAM NAME="java_type" VALUE="application/x-java-applet;jpi-version=1.4">
<PARAM NAME="model" VALUE="models/HyaluronicAcid.xyz">
<PARAM NAME="scriptable" VALUE="true">
         No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</APPLET></NOEMBED></EMBED></OBJECT>
```

ここでも、<EMBED> タグの内部で <XMP> タグが使用されているため、<APPLET> タグが無効にされています。<EMBED> タグは <PARAM> タグおよび </OBJECT> タグも無視します。このため、上のタグは実質的に次のようになります。

```
<EMBED type="application/x-java-applet;jpi-version=1.4"
         width="200" height="200"
         align="baseline" code="XYZApp.class" codebase="html/"
         model="models/HyaluronicAcid.xyz"
         pluginspage="http://java.sun.com/jpi/plugin-install.html">
<NOEMBED>
        No Java 2 SDK, Standard Edition v 1.4 support for APPLET!!
</NOEMBED>
</EMBED>
```

これは、上述の EMBED タグの例と同じになります。

OBJECT-EMBED-JavaScript の複合タグを使用して、適切なプラットフォームの適切なブラウザで Java Plug-in を起動することができます。この複合タグは複雑なため、異機種混在環境で HTML ページを表示する場合にのみ使用してください。

まとめ

このマニュアルでは、Java Plug-in が利用するOBJECT タグおよび EMBED タグの指定方法について説明しています。マニュアル内では、特に APPLET タグから OBJECT タグおよび EMBED タグへの変換に焦点を合わせています。現在のところ、HTML 4.0 では、HTML ページに Java アプレットおよび JavaBeans コンポーネントを挿入する最善の方法として、OBJECT タグの使用が勧められています。OBJECT タグを Java Plug-in のタグ書式に変換する必要が生じた場合には、このマニュアルの内容は更新される予定です。このマニュアルで公開した情報は、HTML 移行ツールを作成する ISV、および Java Plug-in への移行を行う Web ページの作成者を支援するためのものです。このマニュアルで説明されているタグの書式は、将来変更される可能性があります。

Java Plug-in の使用は、このマニュアルに示したタグの書式だけに限定されるものではありません。実際のところ、Web ページの作成者には、必要に応じてタグの書式を変更したり、JavaScript とタグを連携させて使用することが勧められています。Internet Explorer で `OBJECT` タグが使用され、Netscape Navigator ブラウザで `EMBED` タグが使用されている限り、Java Plug-in の実行に問題はありません。現在、いくつかの変換テンプレートが Java Plug-in HTML Converter に添付されています。Web ページの作成者の皆さんには、ニーズに合ったテンプレートを選択し、必要に応じてテンプレートを変更して使用することをお勧めします。

第 4 章

HTML Converter を使用した Java Plug-in の APPLET タグの変換

この節では、次の内容について説明します。

- 35 ページの「はじめに」
- 36 ページの「GUI バージョンの HTML Converter の実行」
- 42 ページの「コマンド行からのコンバータの実行」

注 –

1. このツールを使用して変換を実行する前に、すべてのファイルのバックアップを作成してください。
2. 変換を途中で取り消しても、変更前の状態に戻すことはできません。
3. APPLET タグ内のコメントは無視されます。

はじめに

Java Plug-in HTML Converter は、アプレットを含む HTML ページ (ファイル) を Java Plug-in 形式に変換するユーティリティです。変換プロセスを次に示します。

最初に、アプレットの一部ではない HTML が、ソースファイルから一時ファイルに転送されます。次に、<APPLET> タグを検出すると、コンバータが最初の </APPLET> (引用符で囲まれていない) までアプレットを構文解析して、アプレットデータをテンプレートにマージします。エラーが発生することなくこの操作が完了すると、元の HTML ファイルがバックアップフォルダに移動し、一時ファイルが元の名前に戻されます。

コンバータは、所定位置のファイルを効果的に変換します。このため、コンバータを一度実行すると、ファイルが Java Plug-in 用に設定されます。

GUI バージョンの HTML Converter の実行

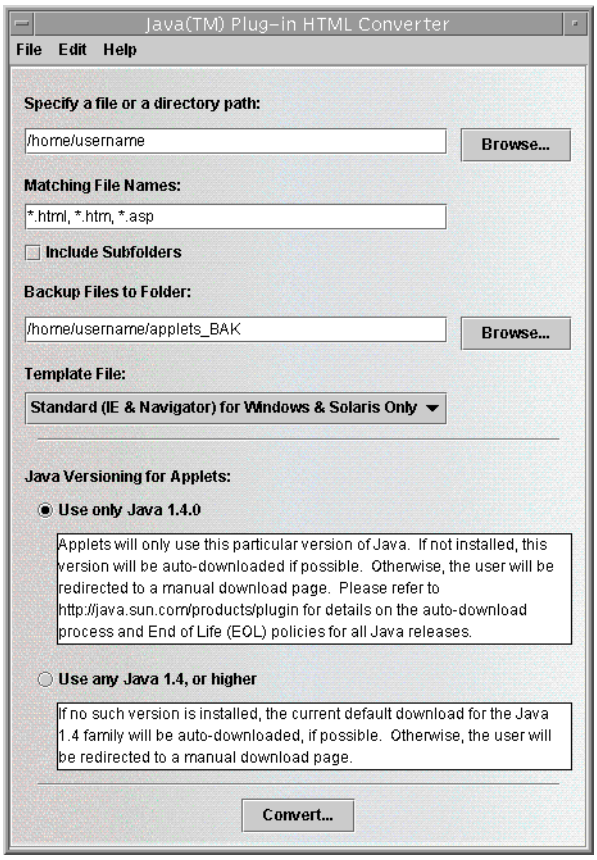
GUI バージョンの HTML Converter の実行

HTML Converter は、Java 2 SDK, Standard Edition, v1.4 インストールの `lib/htmlconverter.jar` ファイル内に存在します。デフォルトのパッケージインストールでは、`/usr/j2se/lib/htmlconverter.jar` になります。GUI バージョンの HTML Converter を起動する場合は、次の操作を実行します。

コマンド行で、<*HTML Converter Directory*> に移動し、次のように入力します。

```
java -jar htmlconverter.jar -gui
```

HTML Converter のウィンドウが表示されます。



フォルダ内の変換対象ファイルの選択 -

フォルダ内のファイルをすべて変換する場合、フォルダへのパスを直接入力するか、「参照 (Browse)」ボタンを選択してダイアログからフォルダを選択します。パスの選択後に、「ファイル名 (Matching File Names)」に任意の数のファイル指定子を指定できます。各指定子を、カンマで区切る必要があります。ワイルドカードとして、「*」を使用できます。最後に、指定したファイル名に一致する、下位のフォルダ内のすべてのファイルを変換する場合は、「サブフォルダを含める (Include Subfolders)」を選択します。

バックアップフォルダの選択 -

デフォルトのバックアップフォルダパスは、ソースパス名に「_BAK」を付けたものになります。たとえば、ソースパスが `/home/user1/html` の場合、バックアップパスは `/home/user1/html_BAK` になります。バックアップフォルダパスは、「バックアップファイル用のフォルダ (Backup Files to Folder)」フィールドにパスを入力するか、フォルダをブラウズして変更できます。

ログファイルの生成 -

ログファイルを生成する場合、「詳細設定 (Advanced Options)」画面 (「編集 (Edit)」→「オプション (Options)」) を表示して、「ログファイルを生成 (Generate Log File)」をチェックします。テキストフィールドにパスを入力するか、ブラウズしてフォルダを選択します。ログファイルには、変換プロセスに関連した基本情報が書き込まれます。

変換テンプレートの選択 -

何も指定しない場合、デフォルトのテンプレートが使用されます。デフォルトのテンプートを使用すると、Internet Explorer ブラウザおよび Netscape ブラウザで動作するように、変換された HTML ファイルが生成されます。別のテンプレートを使用する場合は、メイン画面のメニューから選択できます。「その他のテンプレート (Other Template)」を選択すると、テンプレートとして使用可能なファイルを選択できます。ファイルを選択する場合は、そのファイルがテンプレートであることを確認してください。

バージョンスキーマの選択 -

以下に示す特定バージョンは、コンバータの起動に使用する JRE のバージョン (例、1.4.0) を表します。バージョンの最初の 2 つの数字は、ファミリを表します。たとえば、1.4.3_02 のファミリは 1.4 です。製品のバージョン番号の詳細は、「Java Plug-in での `OBJECT`、`EMBED`、および `APPLET` タグの使用」の注記を参照してください。"

次の 2 つの選択肢があります。

1. 特定バージョンの Java Plug-in を必要とする変換タグを使用して、アプレットを実行できます。選択したバージョンがインストールされていない場合、インストールおよびダウンロードを実行するかどうかの質問がクライアントに対して表示されます。
2. ファミリ内でインストールされている任意のバージョン (特定バージョン以降) の Java Plug-in でアプレットを実行できるように変換タグを指定できます。インストール済みの Java Plug-in が存在しないか、ファミリ内の特定バージョン以降の Java Plug-in がインストールされていない場合、クライアントは、ファミリ内の最新バージョンの Java Plug-in をインストールして実行するかどうかを尋ねられます。

変換 -

「変換 (Convert)」ボタンをクリックして、変換処理を開始します。処理中のファイル、処理済みのファイル数、検出されたアプレットの数、およびエラーの数を示すダイアログが表示されます。

終了またはほかのファイルの変換 -

変換が完了すると、プロセスダイアログのボタンが「キャンセル (Cancel)」から「完了 (Done)」に変化します。ダイアログを閉じる場合は、「完了 (Done)」を選択します。次に Java Plug-in HTML Converter プログラムを終了するか、変換する別のファイルセットを選択します。

テンプレートの詳細 -

テンプレートファイルは、アプレット変換の基盤となるファイルです。これはタグを含むテキストファイルで、各タグは元のアプレットの部分を表します。テンプレートファイル内でタグを追加、削除、または移動することにより、変換後のファイル出力を変更できます。

サポートされるタグ -

$OriginalApplet$	このタグは、元のアプレットのテキスト全体の代わりとして使用される
$AppletAttributes$	このタグは、すべてのアプレット属性 (code、codebase、width、height など) の代わりとして使用される
$ObjectAttributes$	このタグは、`OBJECT` タグに必要なすべての属性の代わりとして使用される
$EmbedAttributes$	このタグは、`EMBED` タグに必要なすべての属性の代わりとして使用される
$AppletParams$	このタグは、アプレットのすべての <param ...> タグの代わりとして使用される
$ObjectParams$	このタグは、`OBJECT` タグに必要なすべての <param ...> タグの代わりとして使用される
$EmbedParams$	このタグは、`EMBED` タグ (「名前 = 値」形式) に必要なすべての <param ...> タグの代わりとして使用される
$AlternateHTML$	このタグは、元のアプレット中で、アプレットをサポートしていない領域内のテキストの代わりとして使用される
$CabFileLocation$	このタグは、IE をターゲットとする各テンプレートで使用される cab ファイルの URL を表す
$NSFileLocation$	このタグは、Netscape をターゲットとする各テンプレートで使用される Netscape プラグインの URL を表す
$SmartUpdate$	これは、Netscape Navigator 4.0 以降をターゲットとする各テンプレートで使用される Netscape SmartUpdate の URL を表す
$MimeType$	これは、Java オブジェクトの MIME タイプを表す

以下に、HTML Converter に同梱される 4 つのテンプレートを示します。テンプレートを新たに作成した場合は、`template` フォルダ内に格納することにより、そのテンプレートを使用できるようになります。

default.tpl - コンバータのデフォルトテンプレート。 変換されたページは、Microsoft Windows プラットフォームで稼働する Internet Explorer ブラウザおよび Netscape Navigator ブラウザで、Java Plug-in の呼び出しに使用できます。このテンプレートは、Solaris オペレーティング環境で稼働する Netscape ブラウザでも使用できます。

```
<!-- HTML CONVERTER -->
<OBJECT classid="clsid:E19F9330-3110-11d4-991C-005004D3B3DB"
$ObjectAttributes$ codebase="$CabFileLocation$">
$ObjectParams$
<PARAM NAME="type" VALUE="$MimeType$">
<PARAM NAME="scriptable" VALUE="false">
$AppletParams$
<COMMENT>
<EMBED type="$MimeType$" $EmbedAttributes$
$EmbedParams$ scriptable=false pluginspage="$NSFileLocation$"><NOEMBED>
</COMMENT>
$AlternateHTML$
</NOEMBED></EMBED>
</OBJECT>

<!--
$ORIGINALAPPLET$
-->
```

ieonly.tpl - 変換済みのページは、Microsoft Windows で稼動する Internet Explorer で Java Plug-in を呼び出す場合にのみ使用できます。

```
<!-- HTML CONVERTER -->
<OBJECT classid="clsid:E19F9330-3110-11d4-991C-005004D3B3DB"
$ObjectAttributes$ codebase="$CabFileLocation$">
$ObjectParams$
<PARAM NAME="type" VALUE="$MimeType$">
<PARAM NAME="scriptable" VALUE="false">
$AppletParams$
$AlternateHTML$
</OBJECT>

<!--
$ORIGINALAPPLET$
-->
```

nsonly.tpl - 変換済みのページは、Microsoft Windows および Solaris オペレーティング環境で稼動する Netscape Navigator ブラウザで Java Plug-in を呼び出す場合に使用できます。

```
<!-- HTML CONVERTER -->
<EMBED type="$MimeType$" $EmbedAttributes$
$EmbedParams$ scriptable=false pluginspage="$NSFileLocation$"><NOEMBED>
$AlternateHTML$
</NOEMBED></EMBED>
```

```
<!--
$ORIGINALAPPLET$
-->
```

extend.tpl - 変換済みのページは、すべてのプラットフォームの任意のブラウザで使用できます。ブラウザが Microsoft Windows 上の Internet Explorer ブラウザまたは Netscape Navigator ブラウザの場合、または Solaris オペレーティング環境の Netscape Navigator ブラウザの場合、 Java™ Plug-in の呼び出しが実行されます。それ以外の場合、ブラウザのデフォルト仮想マシンが使用されます。

```
<!-- HTML CONVERTER -->
<SCRIPT LANGUAGE="JavaScript"><!--
var _info = navigator.userAgent; var _ns = false; var _ns6 = false;
var _ie = (_info.indexOf("MSIE")> 0 && _info.indexOf("Win")> 0 &&
_info.indexOf("Windows 3.1") < 0);
//--></SCRIPT>
<COMMENT><SCRIPT LANGUAGE="JavaScript1.1"><!--
var _ns = (navigator.appName.indexOf("Netscape")>= 0 &&
((_info.indexOf("Win")> 0 && _info.indexOf("Win16") < 0 &&
java.lang.System.getProperty("os.version").indexOf("3.5") < 0) ||
_info.indexOf("Sun")> 0));
var _ns6 = ((_ns == true) && (_info.indexOf("Mozilla/5")>= 0));
//--></SCRIPT></COMMENT>

<SCRIPT LANGUAGE="JavaScript"><!--
if (_ie == true) document.writeln('<OBJECT
classid="clsid:E19F9330-3110-11d4-991C-005004D3B3DB" $ObjectAttributes$
codebase="$CabFileLocation$"><NOEMBED><XMP>');
else if (_ns == true && _ns6 == false) document.writeln('<EMBED
type="$MimeType$" $EmbedAttributes$
$EmbedParams$ scriptable=false
pluginspage="$NSFileLocation$"><NOEMBED><XMP>');
//--></SCRIPT>
<APPLET $AppletAttributes$></XMP>
$ObjectParams$
<PARAM NAME="type" VALUE="$MimeType$">
<PARAM NAME="scriptable" VALUE="false">
$AppletParams$
$AlternateHTML$
</APPLET>
</NOEMBED></EMBED></OBJECT>

<!--
$ORIGINALAPPLET$
-->
```

コマンド行からのコンバータの実行

コマンド行からのコンバータの実行

書式*:* -

```
java -jar htmlconverter.jar [filespecs]  [-simulate] [-options1 value1
[-option2 value2 [...] ]
```

「**java -jar htmlconverter.jar**」だけが指定されている (ファイル指定子やオプションが指定されていない) 場合、GUI バージョンのコンバータが起動します。それ以外の場合、GUI の表示は抑制されます。

[filespecs]:空白で区切られたファイル指定子のリスト。ワイルドカード「*」(たとえば、*.html*、*file*.html*) を含めることができます。

[-simulate]:変換を実行せずに、変換結果をプレビューする場合に設定します。変換結果が不明確な場合に、このオプションを使用してください。変換が完了した場合は、その変換に関する詳細な情報が表示されます。

オプション:	説明
source	ファイルへのパス (たとえば、Windows の場合、 c:\htmldocs、Unix の場合、/home/user1/htmldocs)。デフォルト: <*userdir*> 相対パスの場合、HTML Converter が起動したディレクトリからの相対パスになる
dest	コンバータファイル位置へのパス。デフォルト: <*usrdir*>
backup	バックアップファイルを保存するディレクトリへのパス。デフォルト: <*userdir*>/<*source*>*_bak* 相対パスの場合、HTML Converter が起動したディレクトリからの相対パスになる
f	バックアップファイルを強制的に上書きする
subdirs	サブディレクトリのファイルを処理するかどうかを設定する。デフォルト: *false*

オプション:	説明
template	使用するテンプレートファイルの名前。デフォルト:Windows および Solaris の標準 (IE および Navigator のみ) 注 – 不明な場合は、デフォルトを選択してください。
log	ログのパスおよびファイル名。デフォルト: *<userdir>/convert.log*
progress	変換中に進行状況を標準出力に表示する。デフォルト: *false*
latest	MIME タイプをサポートする最新の JRE を使用する
gui	コンバータのグラフィカルユーザインタフェースを表示する

第 5 章

プロキシ構成

この章の内容は、次のとおりです。

- 45 ページの「はじめに」
- 46 ページの「Java Plug-in がプロキシ情報を取得する方法」
- 47 ページの「直接接続」
- 47 ページの「手動プロキシ設定」
- 48 ページの「自動プロキシ設定」

はじめに

企業カスタマにとって重要なことは、企業内に安全なコンピュータ環境を構築することであり、その上でプロキシ構成は欠かすことはできません。プロキシ構成は、セキュリティバリアとして機能し、プロキシサーバがインターネットとイントラネット間のすべてのトラフィックを確実に監視するようにします。これは通常、イントラネット内の企業ファイアウォールでセキュリティを強化する上で不可欠な要素です。Java Plug-in を使用してイントラネット Web ページ上にアプレットを配備する企業カスタマは、プロキシサポートを設定することもできます。イントラネット環境で Java Plug-in を実行する場合、このサポートが必要になります。これは、Java Plug-in コントロールパネルで設定できます。

コントロールパネルでは、次の 3 つのプロキシオプションの指定が可能です。

- 「ブラウザの設定を使用 (Use browser settings)」
- 「プロトコル - アドレス - ポート (Protocol-Address-Port)」テーブルを使用した手動構成
- 「自動プロキシ構成 URL (Automatic proxy configuration URL)」

「ブラウザの設定を使用 (Use browser settings)」を選択すると、プロキシ情報がブラウザ全体を通じて設定されます。Internet Explorer の場合、「ツール」→「インターネット オプション」を選択してから、「接続」タブの「ローカル エリア ネットワー

ク (LAN) の設定」を選択します。Netscape の場合、「編集」→「設定」を選択してから、「カテゴリ」の「詳細」を選択し、「プロキシ」を選択します。この動作方法およびブラウザから設定可能な 3 つの接続タイプ (「直接」、「手動」、および「自動」) については、次の節で説明します。

コントロールパネルで手動構成を選択した場合、プロトコルごとにプロキシサーバのアドレスおよびポートをテーブルに入力する必要があります。「プロキシホストなし (No proxy host)」というラベルのフィールドにホスト名を記述することにより、プロキシサーバを必要とするホストからそのホストを除外できます。

「自動プロキシ構成 URL (Automatic proxy configuration URL)」を選択した場合、URL で使用されるプロキシサーバを返す、`FindProxyForURL(URL url)` という JavaScript の位置を示す URL を入力する必要があります。このスクリプトは、「自動プロキシ設定」で説明するのと同じ方法でサポートされます。

Java Plug-in がプロキシ情報を取得する方法

プラットフォームが異なると、ブラウザがプロキシ情報を格納する方法も異なるため、プロキシ情報を取得する汎用機構は存在しません。以下に、3 つの異なるブラウザとプラットフォームの組み合わせで、Java Plug-in がプロキシ情報を取得する方法を示します。

Win32 で稼働する *Internet Explorer:* Internet Explorer は、Windows レジストリの同じキーセット内にプロキシ情報を格納します。Java Plug-in は、この情報を直接取得します。

Win32 で稼働する *Netscape Navigator* ブラウザ*:* Navigator 4 は、ローカルマシンのユーザ設定ファイルにプロキシ情報を格納します。Java Plug-in は、ユーザ設定ファイルの読み取りおよび構文解析を実行して、Navigator 4 のプロキシ情報を取得します。Netscape 6 は、プロキシ情報の取得用 API を保持します。`findProxyForURL(URL)` は、指定した URL のプロキシ構成情報を返します。

Solaris オペレーティング環境および *Linux* で稼働する *Netscape Navigator* ブラウザ*:* Navigator は、プロキシ情報をローカルマシンの 1 つのファイル内に格納します。Java Plug-in はファイルの読み取りおよび構文解析を実行して、プロキシ情報を取得します。Netscape 6 は、前の節で説明したのと同じ方法で処理を実行します。

Java Plug-in は、起動時にプロキシ情報を取得します。Java Plug-in の起動後にプロキシ設定を変更する場合、Java コンソールで `p` オプションを使用して、Java Plug-in がブラウザからプロキシ情報を強制的に再ロードするように設定する必要があります。

直接接続

直接接続では、プロキシは使用されません。モバイルユーザがモデム経由で会社に接続するような特定の状況では、イントラネット環境への直接接続が必要であり、この場合は、プロキシを使用しないでください。

手動プロキシ設定

Internet Explorer と Netscape Navigator は、どちらも手動プロキシ設定をサポートします。ユーザは、プロトコルごとに、プロキシサーバおよびポートを指定できます。ユーザは、すべてのプロトコルに対し、1 つのプロキシサーバおよびポートを指定することもできます。プロキシサーバの負荷を最小限に抑えるため、あるサイトでは、マシンがイントラネット環境内の別のマシンに接続している場合、プロキシサーバを完全にバイパスする場合があります。そのためには、ネットワーク管理者およびユーザは、ブラウザ設定でプロキシサーバのバイパスリストを指定します。

Internet Explorer: Java Plug-in は、プロトコルに関連付けられたプロキシサーバおよびポート設定を認識およびサポートします。Internet Explorer は、次に示すプロキシサーババイパスリストのさまざまな構文をサポートします。

- IP アドレス/ホスト名のみ
- IP アドレス/ホスト名 (ワイルドカード指定)
- IP アドレス/ホスト名 (プロトコル指定)

たとえば、プロキシサーババイパスリストで「**`121.141.23.5; *.eng;http://*.com`** 」を指定すると、次のいずれかが発生した場合、ブラウザは常にプロキシをバイパスします。

- 「**`121.141.23.5`**」が要求される
- URL ホスト名の末尾が「**`.eng`**」である
- URL プロトコルが `http` であり、URL ホスト名の末尾が「**`.com`**」である

現在のところ、Java Plug-in は、Internet Explorer のプロキシサーババイパスリスト内で、最初の 2 つの構文をサポートします。Internet Explorer では、ローカル (イントラネット) アドレスの場合、プロキシサーババイパスリストを使用しなくてもプロキシサーバをバイパスします。プレーンな URL ホスト名の場合、Java Plug-in は、プロキシサーバをバイパスすることによりこのオプションをサポートします（例えば、ホスト名にドット(.)が含まれない場合） 。

Netscape Navigator: Java Plug-in は、プロトコルに関連付けられたプロキシサーバおよびポート設定を認識およびサポートします。たとえば、Netscape Navigator ブラウザのプロキシサーババイパスリストで「**.eng,.sun.com**」を指定した場合、URL ホスト名の末尾が「**.eng**」または「 **.sun.com**」の場合は常にプロキシがバイパスされます。Java Plug-in は、Navigator のプロキシサーババイパスリストで、この構文を完全にサポートします。

ブラウザごとの手動プロキシ設定の詳細は、ブラウザのユーザーズガイドを参照してください。

自動プロキシ設定

Internet Explorer と Netscape Navigator は、どちらも自動プロキシ設定をサポートします。ブラウザの自動プロキシ設定は、拡張子 `.pac` または `.js` の JavaScript ファイルを含む特定の URL に設定されます。このファイルには、関数 `FindProxyForURL()` が含まれており、この関数には、ブラウザが接続要求を受信した場合に、使用するプロキシサーバを決定するロジックが格納されています。システム管理者は、特定のイントラネット環境に合わせてこの関数を記述します。ブラウザは、起動時に JavaScript ファイルの URL を認識し、直接接続を使用してローカルマシンにファイルをダウンロードします。新規接続が必要になるたびに、ブラウザはファイル内の JavaScript 関数 `FindProxyForURL()` を実行してプロキシ情報を取得し、接続を確立します。

Internet Explorer: Java Plug-in は、起動時に直接接続を使用して JavaScript ファイルをローカルマシンにダウンロードします。次に、新規接続が必要になるたびに `FindProxyForURL()` 関数を実行し、Internet Explorer の JavaScript エンジンを使用してプロキシ情報を取得します。

Netscape Navigator ブラウザ*:* Java Plug-in は、起動時に直接接続を使用して JavaScript ファイルをローカルマシンにダウンロードします。次に、新規接続が必要になるたびに `FindProxyForURL()` 関数を実行し、Netscape Navigator ブラウザの JavaScript エンジンを使用してプロキシ情報を取得します。

JavaScript 関数 `FindProxyForURL()` から呼び出し可能な、定義済みの JavaScript 関数が多数存在します。 Java Plug-in は、これらの関数を独自に実装することにより、自動プロキシ設定を完全にエミュレートします。この実装に関する注意点を次に示します。

- ホストが IP アドレスではない場合、関数 `dnsResolve()` は常に空の文字列を返します。
- ホストが IP アドレスではない場合、関数 `isResolvable()` は常に false を返します。
- ホストが IP アドレスではない場合、関数 `isInNet()` は常に false を返します。

関数 `FindProxyForURL()` を実行すると、プロキシ情報は常に文字列として返されることに注意してください。Java Plug-in は、次の方法で設定を抽出します。

- 文字列内に「DIRECT」が含まれている場合、Java Plug-in は直接接続と判断します。
- 文字列内に「PROXY」が含まれている場合、最初のプロキシ設定を接続に使用します。
- 文字列内に「SOCKS」が含まれている場合、SOCKS v4 を接続に使用します。
- それ以外の場合、文字列内のプロキシ情報は不正です。この場合、Java Plug-in は直接接続と判断します。

ブラウザごとの自動プロキシ構成の詳細は、ブラウザのユーザーズガイドを参照してください。

第 6 章

プロトコルのサポート

HTTP、FTP 、および Gopher

Java Plug-in は、組み込みのプロキシ設定サポートを含む、HTTP、FTP、および GOPHER プロトコルをサポートします。

HTTPS

概要

Java 2 Platform, Standard Edition のバージョン 1.4 までは、Java Plug-in はブラウザ固有のネイティブ API を介して HTTPS をサポートしていました。Java™ Secure Socket Extension (JSSE) は、1.4 で新たに導入された Java 拡張機能であり、Java プラットフォーム用 SSL および HTTPS の Java 実装を提供します。1.4 では、Java Plug-in はブラウザに依存するのではなく、JSSE を利用して HTTPS サポートを提供します。

これには、ブラウザ固有のネイティブ API を使用する場合に比べて、次の利点があります。

- ネイティブコードを使用しないため、各プラットフォームのブラウザごとに個別の HTTPS をサポートする必要はありません。このため、コードの保守および移植が簡単です。
- `java.net.HttpsURLConnection` の実装が JSSE で提供されるため、開発者は、トンネリングを含む HTTPS の全機能を利用できます。
- サポートがマルチスレッド対応です。Java の内部に実装が存在するため、相互排他ロック (mutex) を使用して接続をロックする必要がなく、このため、同時 HTTPS 接続における Java のパフォーマンスが向上します。

Java Plug-in は、Win32、Linux、および Solaris オペレーティング環境で、JSSE を介した HTTPS をサポートします。

プロキシおよび *cookie* のサポート

HTTPS 接続ごとに異なるプロキシ設定を使用できます。Java Plug-in は、HTTPS でのプロキシ設定をフルサポートします。プロキシ設定は、ブラウザのユーザ環境設定、および Java Plug-in コントロールパネルで設定できます。直接、手動、および自動のプロキシ設定がサポートされます。

すべての HTTPS 接続で、cookie の送受信が可能です。Java Plug-in は cookie をフルサポートするため、ブラウザの cookie ストアを使用して cookie を自動的に取得および更新できます。

エラー処理のサポート

HTTPS サーバへのアクセス時にエラーが発生する場合があります。Java Plug-in は JSSE 内でフックを実行して、次の種類のエラー処理を行います。

- ホスト名の不一致: HTTPS サーバのホスト名がサーバ証明書の名前と一致していない場合、警告ダイアログが表示されます。
- 信頼されないサーバ証明書: SSL ハンドシェイクの実行時にサーバ証明書を検証できない場合、警告ダイアログが表示されます。
- 信頼されないクライアント証明書: サーバがクライアント認証を必要とするが、クライアント証明書を検証できない場合、警告ダイアログが表示されます。
- サーバ認証: クライアントが HTTPS サーバ上の保護されたディレクトリにアクセスすると、ユーザ名とパスワードの入力がユーザに求められます。注: 現時点で、基本認証およびダイジェスト認証のみがサポートされています。

JSSE を使用した *HTTPS* の潜在的な問題点

JSSE を使用した HTTPS のサポートにより、ブラウザ固有の問題の多くを回避できますが、開発者が注意する必要のあるいくつかの問題点が存在します。

- 信頼されないサーバ証明書: HTTPS 接続の確立時に SSL ハンドシェークが実行されると、J2SE に格納されているルート CA に対してサーバ証明書が検証されます。ただし、J2SE がサポートするルート CA 証明書の数は、ブラウザがサポートするルート CA 証明書の数よりも少なくなります。 このため、信頼されないサーバ証明書の問題が発生する場合があります。
- クライアント認証: HTTPS サーバがクライアント認証を必要とする場合、ローカルのクライアント証明書が認証のためサーバに送信されます。JSSE では、クライアント認証は、別個のファイルに格納され、ブラウザから独立しています。クライアント証明書を機能させるため、開発者は keytool を使用してクライアント証明書を JSSE 内にインポートする必要があります。詳細は、http://java.sun.com/j2se/1.4/ja/docs/ja/guide/security/jsse/JSSERefGuide.html の JSSE オンラインマニュアルを参照してください。
- エラー処理のレベル: 現在、Java Plug-in は前の節に示した種類のエラーを処理できます。ただし、Java Plug-in が認識しない、ほかの種類のエラーが存在する場合、Java アプレットコードがブレークする可能性があります。

- 起動時の遅延: HTTPS を使用する際、安全なランダムジェネレータが作成されます。この処理には、数秒から 1 分かかります (クライアントマシンの速度に応じて異なります)。安全なランダムジェネレータの作成時に、Java Plug-in がハングしたように見える場合があります。この問題は、Java Plug-in への HTTPS コードのロードを遅らせること、およびプラットフォームで利用可能な場合にはネイティブ OS の安全なシードジェネレータを利用することで解決されています。このため、HTTPS を使用しない場合でも、Java Plug-in の起動時間は変化しません。ただし、Java Plug-in の起動コードをロードする順序によっては、起動時の遅れが見られる場合もあります。

SOCKS

Java Plug-in は、現在 SOCKS バージョン 4 をサポートします。

注 – HTTP/HTTPS の場合、Web プロキシサーバに対して SOCKS プロキシサーバを使用してキャッシュを追加できます。ただし、Java Plug-in のないブラウザで同様の構成を実行しても、同じ動作にならない場合があります。

第 7 章

cookie のサポート

ここでは、次の内容について説明します。

- 55 ページの「はじめに」
- 55 ページの「Java Plug-in による cookie のサポート方法」
- 57 ページの「Java Plug-in での cookie ポリシーのサポート」
- 57 ページの「cookie およびディスクキャッシュ」
- 57 ページの「cookie およびセキュリティ」

はじめに

cookie は、クライアント側にデータを格納する手段の 1 つです。cookie は、ポータルサイトの個人向けカスタマイズ、ユーザ作業環境の追跡、および Web サイトのロギングなど、広範に使用されています。Web サイトで cookie を使用する企業カスタマの場合、Java Plug-in の cookie サポートは、Java アプレットや JavaBeans™ コンポーネントの配備を簡単にする上で重要な役割を果たします。

cookie サポートを使用することにより、アプレットまたは JavaBeans コンポーネントは、Web サーバから送信された cookie を Web サーバに返すことが可能になります。これにより、サーバにクライアントの状態に関する情報を提供できます。Java Plug-in は、バージョン 1.4 以降、双方向の cookie サポートを提供するようになりました。本書では、さまざまなブラウザ環境での cookie の動作について説明します。

Java Plug-in による cookie のサポート方法

Java Plug-in は、さまざまな Win32 プラットフォームおよび Solaris オペレーティング環境の Internet Explorer と Netscape Navigator ブラウザ、および Linux プラットフォームの Netscape Navigator ブラウザをサポートします。Java Plug-in は、ブラウザ API を介して cookie をサポートします。プラットフォームが異なるとブラウザ

API の実装方法も異なるため、Java Plug-in の cookie サポートはプラットフォームごとに異なります。このため、各ブラウザが cookie をサポートする方法、および Java Plug-in が cookie 情報にアクセスおよび更新する方法を知っておくことは重要です。

URL 接続を使用して HTTP/HTTPS 要求を実行する場合、通常ブラウザは、cookie キャッシュおよびポリシーを検査して、HTTP/HTTPS 要求ヘッダと共に cookie を送信するかどうかを決定します。cookie を送信する場合、ブラウザはキャッシュから cookie を読み取り、HTTP/HTTPS 要求ヘッダの一部に追加します。

URL 接続を使用して HTTP/HTTPS 応答ヘッダを処理する場合、ブラウザはヘッダを検査して、cookie の設定が必要かどうかを判断します。ブラウザは、cookie ポリシーも検査して、アクションが許可されているかどうかを確認します。アクションが許可されている場合、ブラウザは HTTP/HTTPS 応答ヘッダから cookie を抽出して、cookie キャッシュに書き込みます。

Java Plug-in を使用して HTTP/HTTPS 要求を実行する場合、Java Plug-in は cookie も送信する必要があるかどうかをブラウザに問い合わせます。cookie を送信する場合、HTTP/HTTPS 要求には、ヘッダの一部として cookie が含まれます。cookie を送信しない場合、送信される HTTP/HTTPS 要求には cookie が添付されません。

HTTP/HTTPS 応答ヘッダから cookie を設定する場合、Java Plug-in はブラウザ API を使用して処理を実行します (Netscape Navigator 4 ブラウザを除く)。Netscape Navigator 4 ブラウザの場合、Java Plug-in がこの処理に利用できる API が存在しません。

Netscape Navigator 4 ブラウザには、別の制限も存在します。Netscape Navigator 4 ブラウザで Java Plug-in を使用する場合、コードベースがドキュメントベースと同じかそのサブディレクトリである場合にだけ、cookie サポートが有効になります。次の表に示す例を参照してください。

ドキュメントベース	コードベース	**cookie** サポート
http://host.com/my/	http://host.com/my/	有効
http://host.com/my/	http://host.com/my/page	有効
http://host.com/my/page	http://host.com/my/	無効

現時点で、HTTP/HTTPS 接続の確立が必要になると、Java Plug-in の cookie サポートが自動的に起動します。

Java Plug-in の cookie サポートを常に期待通りに動作させるために、以下の項目が推奨されています。

- Internet Explorer ブラウザまたは Netscape 6 ブラウザを使用する。
- Netscape 4 ブラウザを使用しなければならない場合、アプレットとの HTTP/HTTPS 接続で Web サーバが cookie を設定しないようにする。

(上記の推奨事項は、ブラウザおよび Web サーバ配備について制御が可能なイントラネット環境に当てはまります。)

cookie の動作に関する一般的な情報については、ブラウザのユーザーズガイドを参照してください。

Java Plug-in での cookie ポリシーのサポート

Java Plug-in は、Internet Explorer ブラウザと Netscape Navigator ブラウザの両方でサポートされる cookie ポリシーをすべてサポートします。どちらのブラウザでも、cookie ポリシーの設定が可能です (詳細はブラウザのユーザーズガイドを参照)。 さまざまなオプションを設定できます。次にその一部を示します。

- 常に cookie を受け入れる
- cookie の使用をすべて無効にする
- cookie を受け入れる前にプロンプトまたは警告を表示する
- 送信元のサーバに送り返される cookie だけを受け入れる

ブラウザで cookie ポリシーが変更されると、Java Plug-in を使用した次回の HTTP/HTTPS 接続時に変更が有効になります。

Java Plug-in は、cookie キャッシュをサポートしません。その代わり、HTTP/HTTPS 接続が確立されるたびに、ブラウザに問い合わせます。このため、cookie の格納先はブラウザだけです。新しく HTTP/HTTPS 接続が確立されると、ブラウザ内の cookie に対する変更はすべて Java Plug-in に即座に反映されます。

cookie およびディスクキャッシュ

Java Plug-in は、ブラウザ API を使用してディスクキャッシュをサポートします。HTTP/HTTPS 接続経由で `.jar` ファイルまたは `.class` ファイルがダウンロードされるたびに、ディスクキャッシュが起動します。ディスクキャッシュサポートを起動すると、ブラウザがファイル全体をダウンロードし、cookie を自動的に処理します。

cookie およびセキュリティ

Java Plug-in で HTTP/HTTPS 接続が確立されると cookie が送信されますが、たとえ信頼できるコードであったとしても、アプレットや Bean はこの情報にはアクセスしません。 さらに、cookie は、送信元のホストおよびドメインに送り返されるだけです。

第 8 章

Java Plug-in でのアプレット キャッシュおよびインストール

この節の内容は、次のとおりです。

- 59 ページの「キャッシュオプション」
- 62 ページの「セキュリティ」
- 62 ページの「既知の制限」

注 – 以下の説明は、JavaBeans コンポーネントおよびアプレットに適用されます。

キャッシュオプション

バージョン 1.4 リリースでは、新しい形式のアプレットキャッシュが導入されました。アプレットを配備する際、アプレットを「固定」に (ブラウザが上書きできない二次ディスクキャッシュに配置) するかどうかを決定できます。 キャッシュ後に固定アプレットがダウンロードされるのは、サーバ上でアプレットが更新された場合だけです。 それ以外の場合、常にアプレットの高速なロードが可能です。コアビジネスアプリケーションを提供するアプレットは固定にして、起動時のパフォーマンスを改善する必要があります。

いったんアプレットがキャッシュされると、アプレットを再度参照する際、ダウンロードする必要がなくなります。このため、パフォーマンスが改善されます。

この新機能は、次に示すように、Java Plug-in の使用を指定する `OBJECT` タグまたは `EMBED` タグに新しく *cache_archive*、*cache_version*、および *cache_archive_ex* 値を含めることでアクティブにできます。

cache_archive

cache_archive 属性には、キャッシュするファイルのリストを含めます。

```
<PARAM NAME="cache_archive" VALUE="a.jar,b.jar,c.jar">
```

APPLET タグのアーカイブ属性と同様、*cache_archive* 属性内の jar ファイルのリストには絶対 URL は含まれず、EMBED タグまたは OBJECT タグに指定されたコードベースから常にダウンロードされます。

cache_archive 属性内および *archive* 属性内の .jar ファイルのリストは同じように見えますが、同じ .jar ファイルを両方に含めることはできません。

cache_version

cache_version は、オプション属性です。 使用する場合は、キャッシュするファイルバージョンのリストを含めます。

```
<PARAM NAME="cache_version" VALUE="1.2.0.1, 2.1.1.2, 1.1.2.7">
```

各バージョン番号は、*X.X.X.X* の形式で指定します。ここで、*X* は 16 進数です。各バージョン番号は、*cache_archive* 内の各 .jar ファイルに対応します。

cache_archive

jar ファイルのプリロードを可能にするため、HTML パラメータ *cache_archive_ex* を使用できます。このパラメータを使用して、jar ファイルをプリロードする必要があるかどうかを指定できます。また、jar ファイルのバージョンの指定も可能です。*cache_archive_ex* の値は、次の形式で指定します。

```
cache_archive_ex =
"<jar_file_name>;<preload(optional)>;<jar_file_version>,<jar_file_name>;
<preload(optional>;<jar_file_version(optional>,...."
```

preload や *jar_file_version* のようなオプションタグは、 *jar_file_name* の後に区切り文字「;」で区切って任意の順序で指定できます。 以下に、これらのタグを HTML ページ内で使用する方法を示します。

```
<OBJECT ....>
<PARAM NAME="archive" VALUE="a.jar">
<PARAM NAME="cache_archive" VALUE="b.jar, c.jar, d.jar">
<PARAM NAME="cache_version" VALUE="0.0.0.1, 0.0.A.B, 0.A.B.C">
<PARAM NAME="cache_archive_ex" VALUE="applet.jar;preload,
util.jar;preload;0.9.0.abc, tools.jar;0.9.8.7">
```

上の例では、a.jar は archive 内で指定され、b.jar、c.jar、および d.jar は cache_archive 内で指定されています。 b.jar、c.jar、および d.jar には、それぞれバージョン 0.0.0.1、0.0.A.B、および 0.A.B.C も指定されています。*cache_archive_ex* では、 applet.jar のプリロードが指定されています。 util.jar のプリロードも指定されていますが、バージョンも同時に指定されています。tools.jar は、バージョンだけが指定されています。

HTML パラメータ *cache_archive* 内に指定されたすべての jar ファイルにバージョンが指定されていない限り、Java Plug-in は、バージョンを比較しません。 *cache_archive* を *cache_version* なしで使用する場合、*cache_archive* で指定された jar ファイルは、

HTML パラメータ archive で指定された jar ファイルとまったく同様に扱われます。プリロードおよびバージョンオプションが指定されていない場合、同様の扱いが *cache_archive_ex* で指定された jar ファイルに対しても適用されます。

HTML パラメータで指定された .jar ファイルから、次の順序でクラスファイルおよびリソースの検索が行われます。

1. cache_archive_ex
2. cache_archive
3. archive

キャッシュの更新アルゴリズム

デフォルトでは、*cache_version* 属性が存在しないと、次のいずれかの場合にアプレットキャッシュが更新されます。

- *cache_archive* がキャッシュされたことがない
- Web サーバ上の *cache_archive* の「Last-Modified」の値が、ローカルのアプレットキャッシュに格納された値よりも新しい
- Web サーバ上の *cache_archive* の 「Content-Length」の値が、ローカルのアプレットキャッシュに格納された値とは異なる

ただし、状況によっては、Web サーバから HTTP/HTTPS 経由で返される「Last-Modified」値が、アプレットの実際のバージョンを反映していない場合があります。たとえば、Web サーバがクラッシュしてすべてのファイルが復元された場合、サーバ上の *cache_archive* は異なる修正日を保持します。 *cache_archive* が更新されていない場合でも、すべての Java Plug-in クライアントに *cache_archive* の再ダウンロードが要求されます。

バージョンの更新を強制的に実行する場合は、アプレットの配備時に *cache_version* 属性を使用することをお勧めします。

cache_version を使用する場合、`EMBED` タグおよび `OBJECT` タグ内の *cache_archive* の *cache_version* がローカルのアプレットキャッシュに格納されたものよりも番号が大きいと、アプレットキャッシュが更新されます。 バージョン番号が、更新の起動に使用されることに注意してください。Web サーバ上の `.jar` ファイルに実際のバージョン番号が付けられているのではありません。 実際のところ、更新の起動にバージョンが使用されていない場合、*cache_archive* 内のアプレットなしで Web サーバ上のアプレットを更新することが可能です。

cache_version を使用することにより、Web サーバに接続して *cache_archive* の「Last-Modified」および「Content-Length」を取得する必要がなくなります。 たいていの場合、これによりパフォーマンスが向上します。

セキュリティ

固定アプレットは、ローカルにキャッシュされても、元のコードベースおよび署名者により定義されたセキュリティポリシーに準拠します。

既知の制限

- Java Plug-in のキャッシュを使用して、マニフェストの *Class-Path* 変数で指定された JAR ファイルをキャッシュする機能は、現在サポートされていません。
- *cache_archive* 内に指定するパスは、アプレットのコードベースに対する相対 URL でなければなりません。 絶対 URL は、*cache_archive* ではサポートされていません。

第 9 章

Java Plug-in コントロールパネルによる Plug-in の動作/オプションの設定

この節の内容は、次のとおりです。

- 63 ページの「概要」
- 63 ページの「Java Plug-in コントロールパネルの起動」
- 64 ページの「保存オプション」
- 64 ページの「コントロールパネルのオプション設定」

概要

Java™ Plug-in コントロールパネルを使用すると、Java Plug-in が起動時に使用するデフォルト設定を変更できます。 実行中の Java Plug-in インスタンス内部で稼働するアプレットはすべてこの設定を使用します。

Java Plug-in コントロールパネルの起動

コントロールパネルは、ControlPanel 実行可能ファイルを起動することで実行できます。Java 2 SDK では、このファイルは次の位置に存在します。

```
<SDK installation directory>/jre/bin/ControlPanel
```

たとえば、Java 2 SDK が `/usr/j2se` にインストールされている場合、次のコマンドでコントロールパネルを起動します。

```
/usr/j2se/jre/bin/ControlPanel
```

Java 2 Runtime Environment のインストールでは、このファイルは次の位置に格納されます。

```
<JRE installation directory>/bin/ControlPanel
```

Netscape を使用してコントロールパネルのアプレットページを表示することも可能です。このページは、ControlPanel.html というファイル名で JRE ディレクトリにインストールされています。Java 2 SDK では、このファイルは次の位置に存在します。

```
<SDK installation directory>/jre/ControlPanel.html
```

JRE では、次の位置に存在します。

```
<JRE installation directory>/ControlPanel.html
```

保存オプション

コントロールパネルのオプションの変更が完了したら、「適用 (Apply)」をクリックして変更を保存します。変更を取り消して、構成ファイルからコントロールパネルの値を再ロードする場合は、「リセット (Reset)」をクリックします。

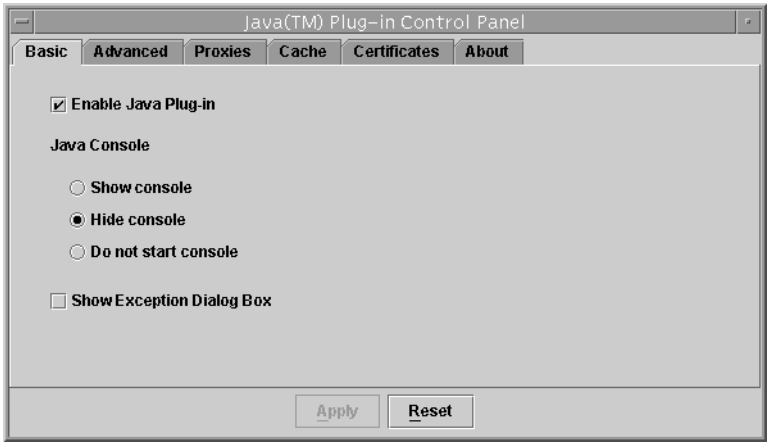
コントロールパネルのオプション設定

Java Plug-in コントロールパネルには 5 つのパネルがあり、さまざまなオプションを設定できます。各パネルのラベルを次に示します。

- 基本
- 拡張機能
- プロキシ (Proxies)
- キャッシュ (Cache)
- 証明書 (Certificates)

「基本 *(Basic)*」パネル

「基本 (Basic)」パネルの外観を次に示します。

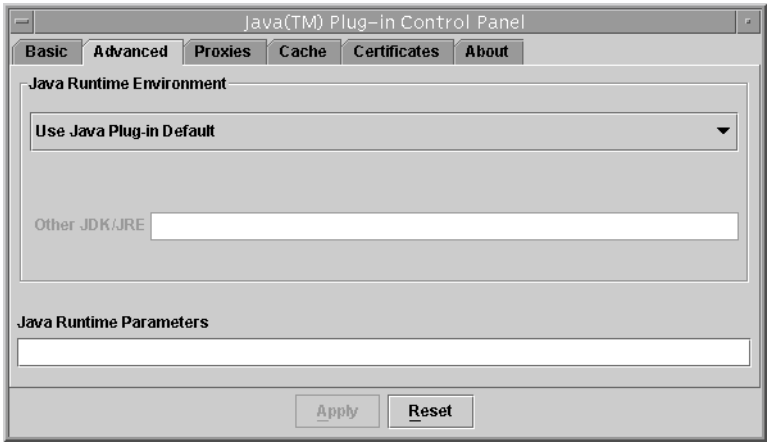


「基本 (Basic)」パネルを使用して、次のオプションを設定できます。

- *Java Plug-in* の有効化 *(Enable Java Plug-in):* Java Plug-in がアプレットや JavaBeans コンポーネントを実行できるようにします。このオプションをチェックしない場合、Java Plug-in はアプレットを実行しません。Plug-in は、デフォルトで有効に設定 (チェック) されています。
- コンソールを表示 *(Show Console):* アプレットの実行時に Java コンソールを表示します。コンソールには、`System.out` および `System.err` オブジェクトから出力されるメッセージが表示されます。この機能はデバッグ時に役立ちます。 デフォルトでは、Java コンソールは非表示 (チェックなし) に設定されています。
- コンソールを非表示 *(Hide console):* コンソールは実行されているが、表示されません。
- コンソールを開始しない *(Do not start console):* コンソールは起動しません。
- 例外ダイアログボックスの表示 *(Show Exception Dialog Box):* 例外の発生時に「例外」ダイアログを表示します。

「詳細 *(Advanced)*」パネル

「詳細 (Advanced)」パネルの外観を次に示します。



「詳細 (Advanced)」パネルを使用して、次のオプションを設定できます。

- *Java Runtime Environment:* マシンにインストール済みのいずれかの JRE または Java 2 SDK, Standard Edition v 1.3 または 1.4 で Java Plug-in が動作するようにします。 Java Plug-in 1.3/1.4 は、デフォルトの JRE に同梱されています。 ただし、デフォルトの JRE を上書きして、古いまたは新しいバージョンの JRE を使用できます。 コントロールパネルは、マシンにインストールされている Java 2 SDK および JRE のすべてのバージョンを自動的に検出します。 リストボックスに、インストール済みで使用可能な Java 2 SDK および JRE のバージョンがすべて表示されます。 リストの最初の項目は常に「Java Plug-in をデフォルトで使用 (Use Java Plug-in Default)」になり、最後の項目は常に「その他 (Other)」です。 「その他 (Other)」を選択した場合、JRE または Java 2 SDK, Standard Edition v 1.3/1.4 へのパスを指定する必要があります。 このオプションの変更は、上級ユーザだけが行うようにしてください。
- *Java* 実行時のパラメータ *(Java Runtime Parameters):* カスタムオプションを指定して、Java Plug-in のデフォルト起動パラメータをオーバーライドします。 この構文は、`java` コマンド行の呼び出しパラメータで使用されるのと同じ構文です。

  アサーションサポートを有効にするには、「Java 実行時のパラメータ (Java Runtime Parameters)」で次のシステムプロパティを指定する必要があります。

  ```
  -D[ enableassertions | ea ][: <package name>"..." | : <class name>]
  ```

  Java Plug-in でアサーションを無効にするには、「Java 実行時のパラメータ (Java Runtime Parameters)」で以下のように指定します。

  ```
  -D[ disableassertions | da ][:<package name>"..." | : <class name>]
  ```

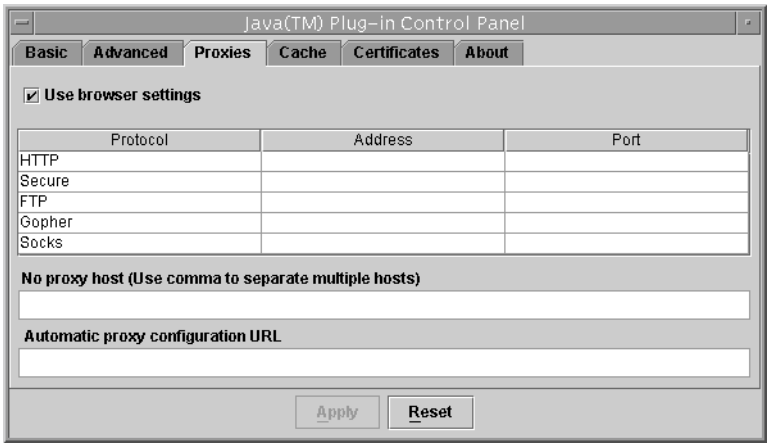
  Java Plug-in のコードでは、アサーションはデフォルトで無効に設定されています。アサーションの効果は Java Plug-in の起動時に決定されるため、Java Plug-in コントロールパネルでアサーション設定を変更した場合、ブラウザを再起動して新規設定を有効にする必要があります。

Java Plug-in 内の Java コードには、アサーションが組み込まれているため、次の方法で Java Plug-in コードのアサーションを有効に設定できます。

```
-D[ enableassertions | ea ]:sun.plugin
```

「プロキシ *(Proxies)*」パネル

「プロキシ (Proxies)」パネルの外観を次に示します。

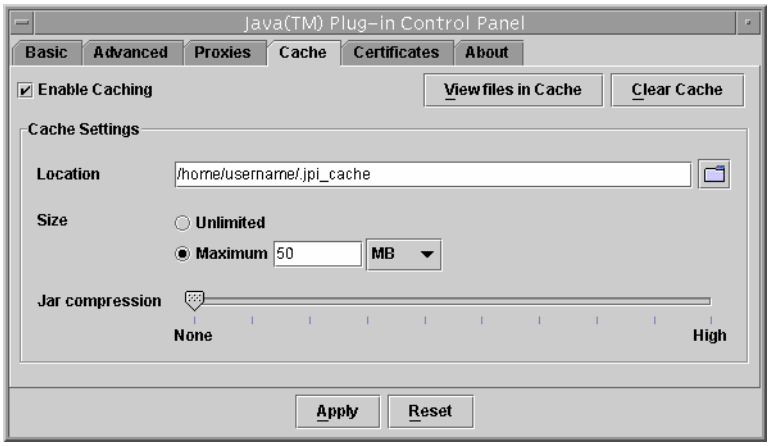


「プロキシ (Proxies)」パネルを使用して、ブラウザのデフォルト設定を使用したり、プロキシアドレスおよびポートを別のプロトコルでオーバーライドできます。

- ブラウザの設定を使用 *(Use browser settings):* ブラウザのデフォルトプロキシ設定を使用します。
- プロキシ情報テーブル*:*「ブラウザの設定を使用 (Use browser settings)」チェックボックスの選択を解除し、下のプロキシ情報テーブルに必要な情報を指定することにより、デフォルトの設定をオーバーライドできます。 サポートされるプロトコル (HTTP、Secure (HTTPS)、FTP、Gopher、および Socks) ごとにプロキシアドレスおよびポートを入力できます。
- プロキシホストなし *(No proxy host):* プロキシを使用しないホストまたはホストのリストを示します。 通常、「プロキシホストなし (No proxy host)」は、イントラネット環境の内部ホストに使用します。
- 自動プロキシ構成 *URL (Automatic proxy configuration URL):* `FindProxyForURL()` 関数を含む JavaScript ファイル (拡張子は .js または .pac) の URL を示します。`FindProxyForURL()` は、接続要求で使用するプロキシサーバを判別するロジックを保持しています。

「キャッシュ *(Cache)*」パネル

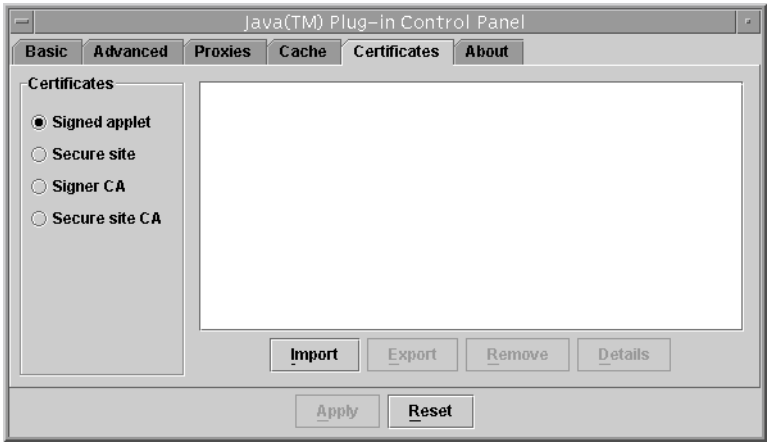
「キャッシュ (Cache)」パネルの外観を次に示します。



- キャッシュを有効 *(Enable Caching):* これをチェックすると、JAR キャッシュが有効になります。
- キャッシュ内のファイルを表示 *(View files in Cache):* これを押すと、JAR キャッシュが表示されます。
- キャッシュをクリア *(Clear Cache):* これを押すと、アプレット JAR ファイルの格納に使用する JAR キャッシュがクリアされます。
- ディレクトリ *(Location):* これを使用して、JAR キャッシュの位置を指定できます。
- サイズ *(Size):* JAR キャッシュのサイズを無制限に設定する場合は、「無限 (Unlimited)」をチェックします。また、JAR キャッシュのサイズを「最大 (Maximum)」に設定することもできます。
- *Jar* 圧縮 *(Jar compression):* JAR キャッシュファイルの圧縮を「None」と「High」の間に設定できます。 圧縮率を高くしてメモリを節約すると、パフォーマンスが低下します。圧縮を行わないときに、最高のパフォーマンスが得られます。

「証明書 *(Certificates)*」パネル

「証明書 (Certificates)」パネルの外観を次に示します。



次の 4 種類の証明書から選択できます。

- 署名済みアプレット *(Signed applet):* 署名付きアプレット用の証明書。
- セキュリティ保護されたサイト *(Secure site):* セキュリティ保護されたサイト用の証明書。
- 署名者の *CA (Signer CA):* 署名付きアプレット用の証明書発行局 (CA) の証明書。証明書発行局は、署名付きアプレットの署名者に証明書を発行する機関です。
- セキュリティ保護されたサイトの *CA (Secure site CA):* 証明書発行局 (CA) が発行した、セキュリティ保護されたサイト用の証明書。証明書発行局は、セキュリティ保護されたサイト用の証明書を発行する機関です。署名付きアプレットおよびセキュリティ保護されたサイト証明書では、「インポート (Import)」、「エクスポート (Export)」、「削除 (Remove)」、および「詳細 (Details)」の 4 つのオプションがあります。ユーザは、証明書のインポート、エクスポート、削除、および詳細表示を実行できます。 「署名者の CA (Signer CA)」および「セキュリティ保護されたサイトの CA (Secure site CA)」では、「詳細 (Details)」オプションのみを指定できます。ユーザは、証明書の詳細のみを表示できます。

第 10 章

Netscape 6

ここでは、Netscape 6 での Java Runtime の主な機能について説明します。これらの機能は、Java Plug-in から有効に設定できます。

APPLET タグ、EMBED タグ、および OBJECT タグのサポート

これらのタグが使用されている場合、アプレットは Java Plug-in によりロードされます。OBJECT タグまたは EMBED タグを使用してアプレットを起動する方法については、第 3 章を参照してください。

Java-JavaScript 間の双方向通信

JavaScript はアプレットのメソッドにアクセスでき、アプレットは JavaScript を使用して Document Object Model (DOM) にアクセスできます。このため、HTML 作成者はアプレットのメソッドにアクセスでき、アプレット開発者は DOM にアクセスできます。

詳細は、 http://java.sun.com/j2se/1.4/ja/docs/ja/guide/plugin/developer_guide/java_js.html のオンラインマニュアル『*Java* から *JavaScript* への通信』および http://java.sun.com/j2se/1.4/ja/docs/ja/guide/plugin/developer_guide/js_java.html のオンラインマニュアル『*JavaScript* から *Java* への通信』を参照してください。セキュリティに関する解説を必ずお読みください。

RSA 署名付きアプレットの検証

RSA 署名付きアプレットの検証が、サポートされています。

Java コンソールの表示

Netscape 6 ブラウザのメニューで、以下のように選択して Java コンソールを表示できます。「タスク (Tasks)」 -> 「ツール (Tools)」 -> 「Java コンソール (Java Console)」

Java プラットフォームの有効または無効

Netscape 6 ブラウザのメニューで、以下のように選択して Java プラットフォームを有効または無効にできます。「編集 (Edit)」 -> 「設定 (Preferences)」 -> 「詳細 (Advanced)」 設定は、ブラウザを再起動するまで有効になりません。

アプレットのライフサイクルの変更

ページが表示されるたびに、アプレットの `init()` メソッドおよび `start()` メソッドが呼び出され、 ページが閉じられるたびに、 `stop()` メソッドおよび `destroy()` メソッドが呼び出されます。

プロキシおよび cookie のサポート

これまでは、Java Plug-in が単独でプロキシおよび cookie サポートを処理していました。Netscape 6 では、ブラウザがこのサポート機能を担当することになりました。

HTTPS

HTTPS は、Java 2 Platform, Standard Edition の Java Secure Socket Extension (JSSE) を使用してサポートされます。

自動ダウンロード

Netscape 6 は、Java Plug-in が存在しない場合、XPInstall 機構を利用して Java Plug-in (JRE) を自動ダウンロードします。

下位互換性の問題

Java 2 Platform と Netscape VM との間の下位互換性を維持することが目標ですが、互換性は 100 % ではありません。現状どおり動作するアプレット、再コンパイルするだけで動作するアプレット、 また、Java 2 プラットフォームへの移植が必要なアプレットがあります。

第 11 章

FAQ (よくある質問)

Java Plug-in の FAQ
は、http://java.sun.com/j2se/1.4/ja/docs/ja/guide/plugin/developer_guide/faq/
で参照できます。

FAQ は、Java Plug-in の初心者向けの基本 FAQ、システム管理者および開発者を対象とした開発者向け FAQ、トラブルシューティングの FAQ、および APPLET タグサポートに関する FAQ で構成されています。