Java Web Start

クライアント・サイドの Java アプリケーションの開発と配布

Steven Kim

2001年 9月 01日

Java Web Start とは、クライアント・サイドの Java アプリケーションの開発を容易にするための新しいテクノロジーです。このテクノロジーのユニークな特色は、開発者がクライアント・アプリケーションの起動方法を気にしなくてもよいという点にあります。つまり、クライアント・アプリケーションを Web ブラウザーから起動するか、デスクトップから起動するかという問題を気にする必要がないのです。さらに、このテクノロジーでは、Web サーバーがクライアント・コードを自主的に配布したり更新したりするための包括的な配置方式が採用されています。今回の記事でこの画期的な新しいテクノロジーを紹介し、開発者とユーザーの両方の観点から実装内容を取り上げるのは、ソフトウェア・エンジニアの Steve Kim 氏です。なお、このテクノロジーは、Java 1.4 の最終リリースに組み込まれる予定です。

ソフトウェア業界のマーケットがどんどん拡大している昨今、ユーザーが求めているのは、デスクトップ環境からだけでなく、インターネットでも実行できるクライアント・アプリケーションです。しかも、ユーザーは、デスクトップ環境のアプリケーションと同じ程度の機能やサポートを Web ベースのアプリケーションにも期待しています。さらに欲を言えば、そのような Web ベースのアプリケーションは、シン・クライアント・インターフェースでパッケージ化されているのが望ましいでしょう。現時点では、そのような要望に対して、2 種類の Java アプリケーションを開発するというのが一般的な対応になっています。つまり、1 つはデスクトップ・アプリケーションであり、もう 1 つはデスクトップ・アプリケーションの機能をかなり模倣した Web ベース・アプリケーションというわけです。ところが、このような対応では、2 つの実行環境に合わせて、同じ機能を持ったアプリケーションを 2 種類開発しなければならないため、開発担当者にとっては大きな負担になります。さらに、2 種類のアプリケーションのサポートを実施しなければならないソフトウェア会社にとっても、財政的な負担となってきます。そこで、このような問題を解消するために、Sun Microsystems 社が投入したのが、Java Web Start というクライアント・サイドのテクノロジーなのです。このテクノロジーを活用すれば、Java アプリケーションをデスクトップからでも、Web ページからでも起動することができます。したがって、Web アプリケーション用に Java サポートを開発する必要がなくなると同時に、ごく一般的な Web サーバーのサポートによって、クライアント・サイドの Java アプリケーションの更新・保守・管理を効率的に実施できるようになります。

Java Web Start とは何か

Java Web Start とは、アプレットのような移植性と、サーブレットや JavaServer Pages (JSP) テクノロジーのような保守容易性と、マークアップ言語 (XML や HTML) のようなシンプルさを取り入れたソフトウェア・テクノロジーです。この Java ベースのアプリケーションでは、十分な機能を備えた Java 2 クライアント・アプリケーションを標準的な Web サーバーから起動したり、配置したり、更新したりすることが可能になります。Java Web Start を初めて起動したときに、ユーザーは、Web から新しいクライアント・アプリケーションをダウンロードできます。いったんダウンロードしたアプリケーションは、その後、Web ページのリンクから起動することも、(Windows であれば) デスクトップ・アイコンや「スタート」メニューから起動することもできるわけです。Java Web Start の下では、アプリケーションが短時間で初期化され、クライアント・マシンのキャッシュに入れられ、オフラインで起動することも可能になります。さらに、Java Web Start は Java 2 テクノロジーをベースにしているので、Java プラットフォームの優れたセキュリティー方式を継承しています。

Java Web Start は、それ自体が Java アプリケーションなので、プラットフォームを選びません。したがって、Java 2 プラットフォームをサポートしているクライアント・システムであれば、どんなシステムででも実行できます。Java Web Start は、クライアント・アプリケーションの起動時に、以前のキャッシュが存在すれば、そのキャッシュからアプリケーションを読み込むと同時に、Web から最新のコードをダウンロードするという方法で、自動的に更新を実行します。Java Web Start には、Java Application Manager というユーティリティーも組み込まれています。これは、エンド・ユーザーが各種の Java アプリケーションを管理するためのユーティリティーであり、ダウンロードしたアプリケーションのキャッシュを消去したり、複数の JRE の使用を指定したり、HTTP プロキシーを設定したりするためのいろいろなオプションが用意されています。

Java Web Start と Java Plug-in の比較

Java Web Start と Java Plug-in にはいずれにも共通する目的があります。つまり、どんなプラットフォームでも、どの場所からでも、Java プログラムを安全に実行するという目的です。このどちらの Java テクノロジーの場合も、「サンドボックス」と呼ばれる安全な環境が用意されており、その環境の中で Java プログラムを起動して実行できるようになっています。さらには、アプリケーションをキャッシュに入れたり、使用する JRE を指定したりするための機能も共通です。ところが、Java Web Start と Java Plug-in には根本的な違いがあります。それはつまり、Java Plug-in の機能が、Web ブラウザー内での Java アプレットの実行に限定されているという点です。Java Plug-in は、Web ブラウザーの JRE に大きく依存しているため、ブラウザーが稼働していない状態では実行できません。それに対し、Java Web Start の場合は、Web ページ内のリンクを 1 回クリックするか、Java Application Manager で所定のオプションを 1 回クリックするだけで、Java アプリケーションを起動できます。Windows オペレーティング・システムの環境では、「スタート」メニューから起動することも、デスクトップ・アイコンをダブルクリックして起動することもできるわけです。Web ブラウザーから起動したクライアント・アプリケーションは、ブラウザーのウィンドウを閉じたとしても影響を受けません。

確かに、Java Web Start には、Java アプレットのサポートも Appletviewer も組み込まれていますが、このテクノロジーの主な目的は、あくまでも、Java クライアント・アプリケーションを起動したり、配置したりすることにあるのです。したがって、アプレット用のポリシー・ファイルなど、一部の制限事項はサポートされていません。

ユーザーの観点から見た Java Web Start

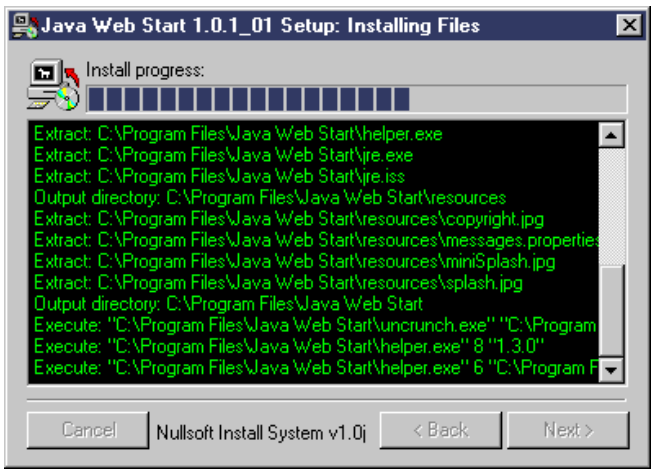
Java Web Start があれば、Web ページから起動して実行できるアプリケーション (アプレットだけではない) の開発が容易になるという開発者にとっての利点と共に、ユーザーにとっての利点も確かにあります。つまり、ユーザーが何か操作しなくても、既存のクライアント・コードを自動的に更新する配置方式にも重点が置かれているのです。この方式を理解し、Java Web Start に対応したアプリケーション開発を進めるべきかどうかを判断していただくために、ユーザーの観点からこの製品を見てみることにしましょう。まずは、ユーザーの側で実行する 2 つの基本的な操作について簡単に説明します。1 つは、Java Web Start をインストールするという操作、もう 1 つは、Java Web Start を使って Web からアプリケーションをダウンロードして起動するという操作です。

Java Web Start を初めてインストールする

Java Plug-In の場合は、ユーザーが何かのアプリケーション・プログラムをクライアント・マシンにインストールする必要はありません。しかし、Java Web Start は、まずすべてのクライアント・マシンにインストールする必要があります。インストールした Java Web Start によって、Web から Java アプリケーションが起動されるというわけです。ユーザーが Java Web Start を使った Web ベース・アプリケーションを起動しようとすると、Web ブラウザーが Java Web Start を起動して、該当するファイルのダウンロードを開始することになります。

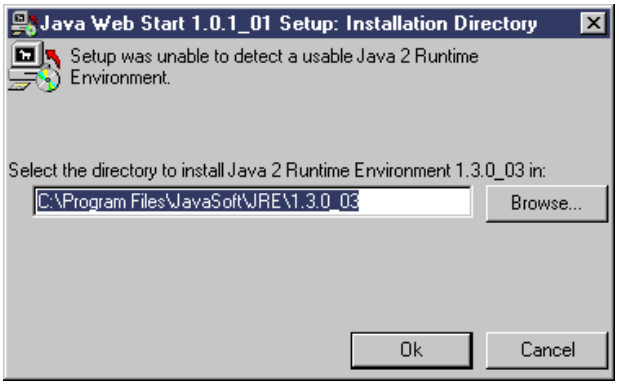
Java Web Start がローカル・マシンにインストールされていれば、目的のアプリケーションが起動して、正常な処理が始まります。しかし、Java Web Start がインストールされていない場合は、Java Web Start をダウンロードするための画面が表示されます。ユーザーが Java Web Start のダウンロードに同意して、そのファイルを実際にダウンロードしたら、次のそのプログラムを実行して、Java Web Start をインストールする必要があります。図 1 は、そのインストールの画面です。

図 1. Java Web Start のインストール



インストール中に、Java Web Start のセットアップ・プログラムは、Java 2 環境がクライアント・マシンにインストールされているかどうかを確認します。インストールされていない場合は、図 2 のようなダイアログが表示されます。

図 2. Java Web Start による Java 2 環境の検出



Java Web Start をインストールしたら、その時点で開いた状態になっている Web ブラウザーをすべて閉じてください。Web ブラウザーが自分自身と Web サーバーと Java Web Start とのやり取りを開始するための、拡張子 jnlp を持つ新しい MIME タグが Web ブラウザーに対して定義されます。(JNLP とは、Java Network Launching Protocol の略語です。JNLP ファイルによって、クライアント・サイドの Java アプリケーションに必要な JAR ファイルやリソースが決定されることになります。)Web ブラウザーをいったん閉じてから再起動した時点で、目的のアプリケーションを起動するためにもう一度同じリンクをクリックしてください。今度は、Web ブラウザーが Java Web Start を起動して、目的のアプリケーションの起動プロセスが始まることになります。

「開発者ガイド Java Web Start」(参考文献を参照) には、Java Web Start がローカル・マシンにインストールされているかどうかを確認するための JavaScript と Visual Basic のスクリプトも収録されています。

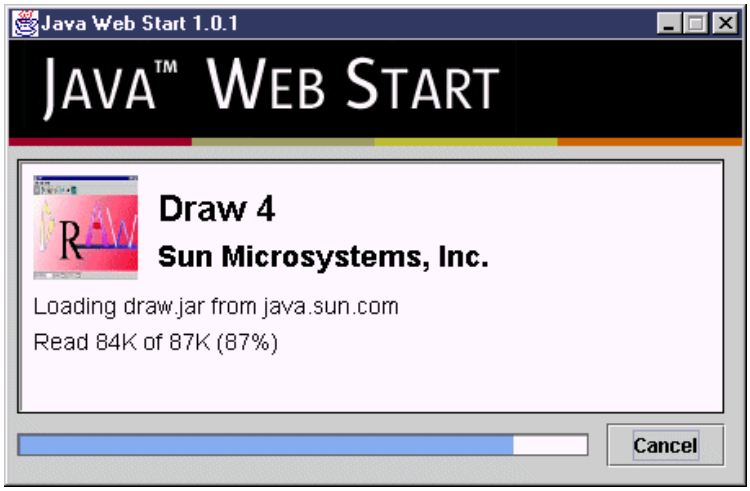
Java Web Start を使ってアプリケーションを起動する

ユーザーがクライアント・サイドの Java アプリケーションを起動するために初めてリンクをクリックすると、次のような流れで処理が進みます。

- Web ブラウザーに対して、Java Web Start を起動するように命令が送られます。
- スプラッシュ画面が表示されて、Java Web Start が起動します。
- Java Web Start が指定の Web サーバーに接続し、目的の Java アプリケーション用のどのファイルがすでにダウンロードされているかどうかを確認します。
- 必要に応じてファイルがローカル・マシンにダウンロードされます。
- Java Web Start が、ダウンロードされたアプリケーションを起動します。

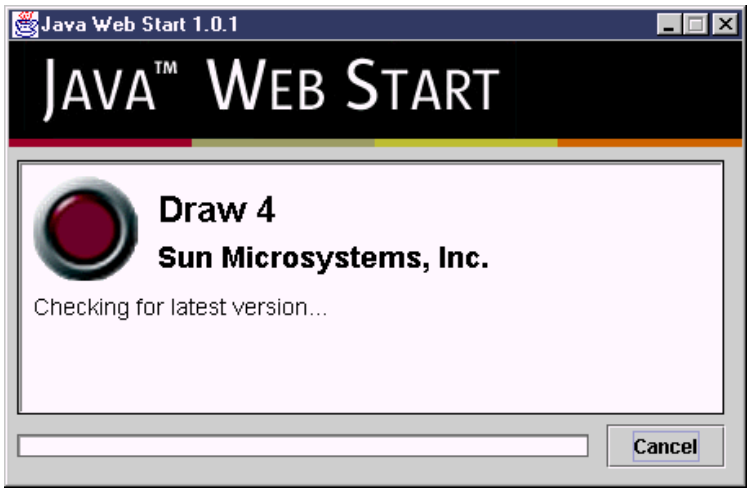
Java Web Start にも、Java Web Start Application Manager にも、外観に一貫性のあるインターフェースが用意されており、インストールと、起動プロセス中、ユーザーに対する通知画面が表示されます。図 3 は、Java Web Start が初めてクライアント Java アプリケーションを起動するときに表示される画面の一例です。

図 3. Java Web Start からクライアント JAR ファイルをダウンロードしているときの画面



Java Web Start がアプリケーションをローカル・マシンにダウンロードした後に、クライアント・プログラムが実行されます。将来、Java Web Start が再びそのプログラムを起動するときには、クライアントに最新バージョンのアプリケーションがあるかどうかの確認が行われます。その確認処理の最中では、図 4 のような画面がユーザーに表示されます。

図 4. Java Web Start がアプリケーションのバージョンを確認しているときの画面



ユーザーには、クライアント Java アプリケーションが起動中なのか、それとも更新中なのかが、画面によってはっきりと通知されます。さらに、進行状況バーや説明文によって、処理の所要時間だけでなく、更新や起動がいつごろ完了するのかをある程度つかめるようになります。これはこの製品のたいへん便利な機能であり、アプリケーションの起動処理のどの段階においても、ユーザーには処理の内容が通知されることになります。

GUI

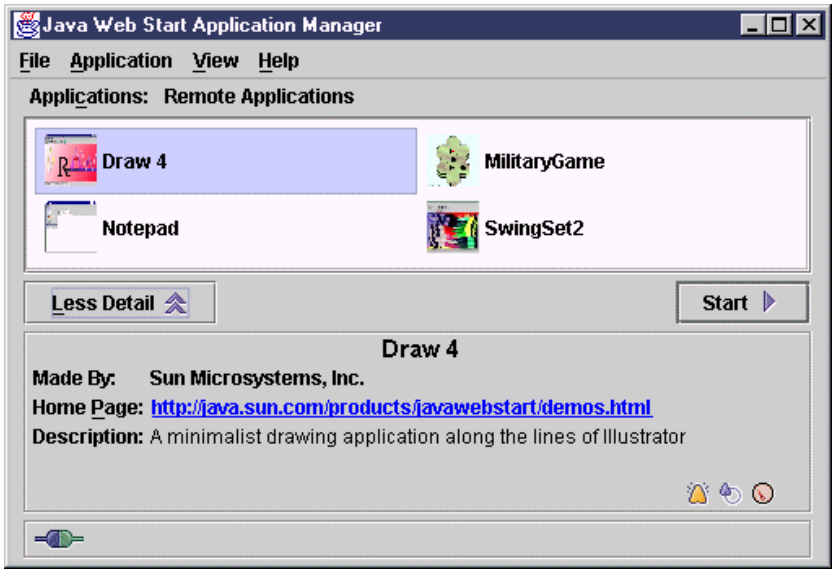
Application Manager の GUI (グラフィカル・ユーザー・インターフェース) は、シンプルでありながらも合理的なインターフェースであり、アプリケーション起動時のオプションをいろいろと指

定できるようになっています。Application Manager を起動すると、図 5 のようなスプラッシュ画面が表示されます。

図 5. Application Manager のスプラッシュ画面

このスプラッシュ画面が消えた時点で、Application Manager のメイン・ウィンドウが表示されます。その画面が図 6 です。

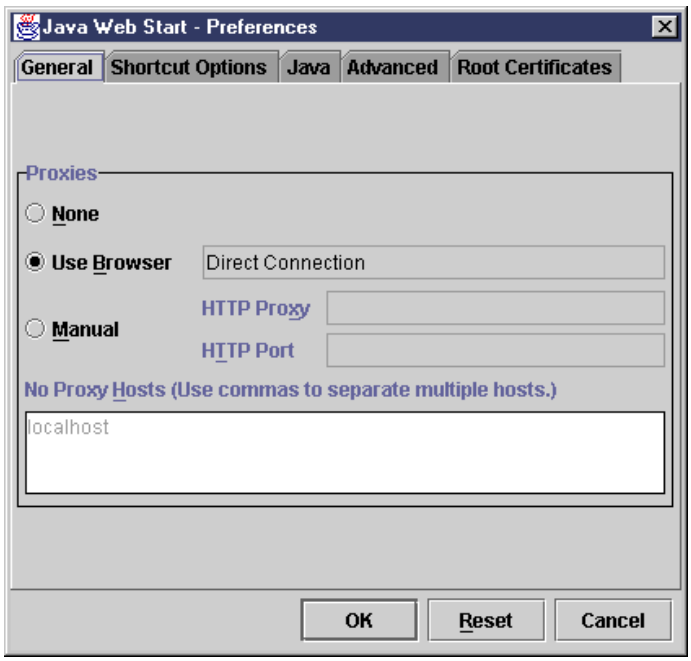
図 6. Application Manager のメイン・ウィンドウ



「Applications (アプリケーション)」ボックスには、Java Web Start から起動する全アプリケーションが表示されており、ここから実際に起動できます。その下のボックスには、選択した Java アプリケーション (図 6 の場合は Draw 4) に関する追加情報が表示されます。つまり、そのアプリケーションのベンダー名、そのアプリケーションの詳細が説明されているホーム・ページ、そのアプリケーションの簡単な説明といった情報です。このような情報は、そのアプリケーションの JNLP ファイルから直接読み込まれます (JNLP の詳細は後述)。

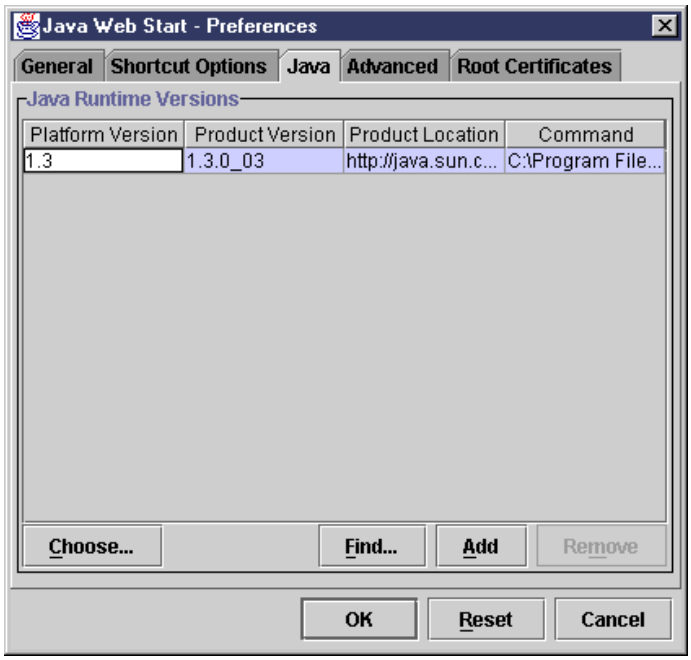
「File (ファイル)」—「Preferences (プリファレンス)」の画面では、ユーザーが各種の設定を調整できます。ほとんどの値は Java Web Start によって自動的に設定されますが、プロキシーの設定など、ユーザーが上書きできる設定もいろいろあります。図 7 は、プロキシーの設定の画面です。

図 7. Application Manager の「Preferences (プリファレンス)」画面



ユーザーは、アプリケーションを起動するときに使う JRE のバージョンも指定できます。そのための画面が図 8 です。

図 8. JRE のバージョンを指定するための画面



要するに、Application Manager の目的は、Java Web Start のガイドラインとポリシーに準拠したすべてのアプリケーションを管理するための統合ユーティリティーを提供することにあります。さらに、上級ユーザーの場合は、一部のアプリケーション設定を調整することもできます。そのようにすれば、アプリケーションの起動や設定のために、スクリプトやバッチ・ファイルを作成するといった、通常であれば必要になるかもしれない作業を省けます。ただし、念のために強調しておきますが、Application Manager が威力を発揮するのは、Java 2 プラットフォームに対応し、なおかつ、Java Web Start との互換性があるアプリケーションを管理する場合に限られます。ローカル・マシンに存在するその他の Java アプリケーションは検出されません。

検討すべき問題点

Java Web Start に対応したアプリケーションを開発するかどうかを決めるときには、ユーザーの状況を全体的に検討するのが賢明です。

- Java Web Start は、Web または Java Application Manager からクライアント Java アプリケーションを起動するすべてのマシンにインストールする必要があります。したがって、クライアント Java アプリケーションを Web ページから起動できるとはいっても、まずローカル・マシンに Java Web Start をインストールすることが前提になります。つまり、Web ブラウザーは、アプリケーションを起動するための便利なリンクを提供するにすぎず、それ以上でもそれ以下でもないのです。
- Java Web Start は、Java Web Start のガイドラインに準拠して正しくパッケージ化されたアプリケーションについてのみ威力を発揮します。さらに、Web サーバーも必要であり、すべての JNLP タグと MIME タグを解釈できるように、その Web サーバーを正しく設定しておく必要があります。したがって、Web からダウンロードして起動するすべてのクライアント Java アプリケーションがそのような要件を満たしていないとすれば、Java Web Start から起動したり管理したりできないクライアント Java アプリケーションが存在するということになります。
- ユーザーの間には、Java Web Start に対応したアプリケーションと Java Plug-In に対応したアプレットの違いについて、いくらかの混乱があるかもしれません。クライアント Java アプリケーションも Java アプレットも Web ブラウザーから起動できるのですが、Java Web Start は、クライアント Java アプリケーションを管理したり起動したりすることはできても、Java Plug-in で実行する Java アプレットに対しては何もできません。この点を、ユーザーに正しく理解されない場合も生じ得ると思われます。
- たった 1 つのクライアント Java アプリケーションのために Java Web Start をダウンロードするとすれば、Java Web Start Application Manager の機能は大げさに感じられるかもしれませんし、Java Web Start のインストールに手間をかけるのも、無用な "大騒ぎ" のように思えるかもしれません。
- Java Web Start は、Java 2 プラットフォームに対応しているクライアント Java アプリケーションについてのみ威力を発揮します。
- Java Web Start が現時点で対応しているオペレーティング・システムは、Windows 95/98/NT/2000、Linux、Solaris です。

Java Web Start に対応した開発

開発者にとっては、Java Web Start に対応したコードを作成するのは、通常のクライアント・サイド Java アプリケーションの場合と変わりがありません。とはいえ、パッケージングの問題や、既存のクライアント・コードを更新するための連絡の問題については、あまり心配が要らないはず

です。Java Web Start に対応したクライアント・サイド Java アプリケーションを開発して配置する際の簡単なルールを次にいくつか記しておきます。

- この種のアプリケーションは、Java 2 プラットフォームに対応した単体のアプリケーションとして作成できます。
- この種のアプリケーションからローカル・システムにアクセスする必要がある場合は、JAR ファイルの各項目に署名が必要になります。
- イメージやサウンド・ファイルなどのリソースを含め、この種のアプリケーションに必要なすべてのファイルは、JAR ファイルのセットとしてまとめて格納しておく必要があります。
- イメージなどの読み込みリソースにアクセスするには、`getResource` メソッドとクラス・ローダーを併用する必要があります。`getResource` メソッドの使用例については、次のリスト 1 を参照してください。

リスト 1. ClassLoader オブジェクトを使って JAR ファイルからリソースを読み込む

```
//Obtain the current classloader
ClassLoader classLoader = this.getClass().getClassLoader();
//Load the company logo image
Image companyLogo = classLoader.getResource("images/companyLogo.gif");
```

アプリケーション・コードの更新

Web サーバーから最初にアプリケーションをダウンロードするときには、関連する JAR ファイルがすべてローカル・マシンにダウンロードされます。そのダウンロードの場所は、デフォルトのセキュリティー・プロトコル (ローカル・ディスクへのアクセスなど) の適用によって、Java Web Start が自動的に判別します。セキュリティーについては、後で詳しく説明します。

アプリケーションの開発と保守という観点からすれば、Java Web Start によって、アプリケーション・コードの更新が容易になります。アプリケーションに必要なすべてのファイルは、クライアント・マシン上にダウンロードして、JAR ファイルのセットとして格納しておく必要があるので、開発者としては、Web サーバー上の JAR ファイルのセットだけを更新すればよいということになります。アプリケーションの最初の起動時に、Java Web Start は、クライアント・マシンにどのファイルやリソースをダウンロードして更新する必要があるのかを判別します (その際に必要になるバージョン管理方式については、後で説明します)。ユーザーがアプリケーションを起動するたびに、Java Web Start は、JAR ファイルのセットが格納されている Web サーバーに接続し、必要なファイルをダウンロードします。念のために強調しておきますが、クライアントからダウンロードできるリソースは、JAR ファイル、イメージ、JNLP ファイルだけです。

Java Web Start は、HTTP 要求を使って Web サーバーからリソース・ファイルを取得し、クライアント・サイド Java アプリケーションをプロキシー・サーバーやファイアウォールの背後から実行できるようにします。そのために、Java Web Start はまず、ローカル・マシンのデフォルトの Web ブラウザーに定義されているプロキシー設定を検出し、標準的なプロキシー自動設定スクリプトを使おうとします。追加情報が必要な場合は、Application Manager の「Preferences (プリファレンス)」画面を使って、ポートやプロキシーを指定できます。

Java Web Start のコア: JNLP

Java Web Start を使って Java アプリケーション を実行するには、そのアプリケーションの Java Networking Language Protocol (JNLP) ファイルを作成する必要があります。JNLP ファイルとは、

アプリケーションの基本的な要素や説明を記述した XML ファイルです。JNLP は Java Community Process (JCP) の JSR 000056 仕様に準拠しています (参考文献を参照)。

JNLP ファイルには、次のような目的があります。

- Web サーバーからダウンロードする JAR ファイルを指定すること
- その他のパッケージ要件を指定すること
- システム・プロパティーを指定すること
- 必要な実行時パラメーターを指定すること
- 使用する Java 2 プラットフォームのバージョンを指定すること

JNLP ファイルのサンプル

JNLP ファイルは、Web ブラウザーと同じ要領で、ファイルを探し出して取得します。ただし、使用するのは URL だけで、特定のファイル名は使いません。リスト 2 は JNLP ファイルのサンプルです。

リスト 2. JNLP ファイルのサンプル: clientApp.jnlp

```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp codebase=http://www.companySite.com/javaApp>
<href="clientApp.jnlp">
<information>
     <title>This is my company's Java client application</title>
     <vendor>Company name</vendor>
     <icon href="companyLogo.gif"/>
     <homepage ref="reference/tips.html">
     <offline-allowed/>
</information>
<resources>
     <j2se version=1.3/>
     <jar href="companySong.jar" part="music" download="lazy"/>
</resources>
<resources os="Windows"/>
<nativelib="windowIconsForWindowOS.jar" part="windowIcons"
download="eager"/>
<application-desc main-class="com.company.ui.Client"/>
<security>
     <all-permissions/>
</security>
</jnlp>
```

リスト 2 に関する注意点

この JNLP ファイルのサンプルについて、いくつかの注意点を記しておきます。これを参考にして、独自のファイルを作成してみてください。

- 最初の行では、ファイルのエンコード方式として、UTF-8 文字エンコード方式が指定されています。したがって、JNLP ファイルを編集するときには、ファイルの正しいエンコード方式を確実に指定するために、JDK に用意されている native2ascii ツールを使って、仕上がったファイルを変換するとよいでしょう。
- `jnlp codebase` 属性には、JNLP ファイルで使用する相対 URL をすべて指定します。`href` 属性は、アプリケーションを Java Web Start Application Manager に組み込むために設定します (必須)。
- `information` タグには、`title` や`vendor` などがあり、Java Web Start Application Manager にあるアプリケーションに関する追加情報を指定します。特に、`homepage ref` 属性には、アプリ

ケーションの詳細が説明されている Web ページの URL を指定します。`information` タグの中でも、一番面白いのは`offline-allowed` 属性でしょう。この属性では、クライアント Java アプリケーションをオフラインで起動できるかどうかを指定します。クライアント Java アプリケーションをオフラインで実行する場合でも、Java Web Start は、Web サーバー上の最新のファイルを判別しようとします。しかし、その場合はオフラインでの起動が指定されているので、すぐにタイムアウトになり、ローカル・キャッシュからアプリケーションが起動されることになります。

- `resources` タグでは、アプリケーションが使用する JAR ファイルや、JAR ファイルのダウンロード方法を指定できます。ダウンロードの方法としては、"がんばり屋モード" (eager) と "怠け者モード" (lazy) があります。クライアントに大量のファイルをダウンロードしなければならないとはいえ、クライアントの実行にすべてのファイルが必要なわけではないという場合に、この機能はたいへん役立ちます。デフォルトでは、ほとんどのリソースが がんばり屋モード でダウンロードされます。つまり、アプリケーションを実際に起動する前に、JAR ファイルとリソースをダウンロードしてしまうという意味です。それに対して、怠け者モード でダウンロードするリソースは、Java 仮想マシン (JVM) が、アプリケーションからリソースやファイルを読み込む処理を開始した時点ではじめてダウンロードされることになります。怠け者モード でダウンロードするリソースの例としては、クライアントのヘルプ・ファイルがあります。ユーザーが実際にクライアントからヘルプ・ファイルを要求するまでは、ヘルプ・ファイルがダウンロードされることはありません。したがって、クライアントの起動と実行にかかる時間が短縮されます。ところが、ユーザーがヘルプ・ファイルを要求すると、その要求の処理中に、該当する JAR ファイルがダウンロードされ、おおよそのダウンロード時間を示した画面がユーザーに表示されます。
- JNLP ファイル内の最後の面白いタグは、`security` でしょう。デフォルトでは、Java Web Start から起動されるクライアント Java アプリケーションは、安全な限定的環境で実行されることになります。その環境の中では、ローカル・ファイルへのアクセスや、他のコンピューターへのネットワーク接続などが禁止されています。しかし、クライアント・マシンやネットワークへのアクセスを必要とする高機能なクライアント・アプリケーションの場合は、`all-permissions` という値を使って、完全なアクセス権を与えることができます。ローカル・マシンに対する完全アクセスを認める場合は、Java アプリケーションが読み込んで使用するすべての JAR ファイルとリソースにデジタル署名を付ける必要があります。セキュリティーについては、後で詳しく説明します。

JNLP ファイルと WAR ファイル

Java Web Start では、アプリケーションの JNLP ファイルと JAR ファイルをパッケージ化するためのさらに便利で効率的な方式として、Web Archive (WAR) ファイルによる配布方式が採用されています。WAR ファイルはディレクトリー構造になっていて、その中に、一つの JNLP ファイルと複数の JAR ファイルとともにパッケージ化されたサーブレットが入っているため、バンドル全体を Web サーバー上に簡単に配置できるようになっています。そのサーブレット自体は、jnlp-servlet.jar ファイルとしてパッケージ化されており、メインの実行 Java クラスとして、`JnlpDownloadServlet` クラスがさまざまなタスクの実行を担当します。たとえば、次のようなタスクです。

- JARDiff ファイルの生成
- WAR ファイル内のファイルごと、またはディレクトリーごとのバージョン管理
- JNLP ファイルに定義されているダウンロード・プロトコルのサポート

- JNLP ファイルに対する URL の自動挿入 (したがって、開発者が URL をハードコーディングする必要はない)

WAR ファイルでアプリケーションをバンドルしなくても、Java Web Start からアプリケーションを起動することはできます。しかし、WAR ファイルの形にしておけば、Web サーバー上のファイルの管理や更新が容易になります。WAR ファイルの詳細については、今回の記事の守備範囲を超えているので、参考文献に挙げられているリンクを参照してください。

Java Web Start におけるバージョン管理

これまで、Java Web Start を実装する開発者とユーザーの双方にとって必要な配置・管理テクニックをざっと取り上げてきました。さらに説明を進める前に、ここで配置手順の流れをまとめておきましょう。

- アプリケーションで使用するファイルを JAR ファイルに格納します。
- JNLP ファイルを作成します。Java Web Start はこのファイルを読み込んで、アプリケーションが使用する JAR ファイルや、必要なファイルのダウンロード方法などを判別することになります。
- 標準的な Web サーバーの設定を調整して、Java Web Start が解釈する MIME タイプを使えるようにします。
- JAR ファイルを Web サーバー上に置きます。
- ユーザーが Java Web Start をダウンロードしてインストールします (1 度だけ)。
- ユーザーがアプリケーションを初めて実行します。そのときに、JAR ファイルがダウンロードされ、実行されます。
- ユーザーがアプリケーションを再度実行します。Java Web Start Application Manager からでも、Web ページからでも起動できます。そのときに、Java Web Start が、更新の必要なファイルをダウンロードしてから、アプリケーションを起動することになります。

このような要約を読むと、技術的な観点から 1 つの質問が出てくるかもしれません。つまり、Java Web Start は更新の必要なファイルをどのようにして判別するのか、という質問です。その答えは JNLP ファイルにあります。JNLP ファイルで定義する各リソースには、バージョン ID の付いた固有の URL が関連付けられています。その例がリスト 3 です。

リスト 3. JNLP ファイル内でバージョン情報の付いた JAR ファイルを定義する

```
<jar href=http://www.companySite.com/javaApp/imageFiles.jar
version="1.1+">
```

アプリケーションの起動時に、Java Web Start は JNLP ファイルを検査し、その中の URL とバージョン ID を使って HTTP の`GET` 要求を作成します。`version` 属性が指定されていない場合は、単に JAR ファイルに対する`GET` 要求が作成されることになります。その JAR ファイルに最新のバージョンが存在するかどうかは、Web サーバーからの応答の状況コードと MIME タイプによって判別されます。

JNLP ファイルでは、JAR ファイルの差分更新も指定できます。そのためには、`JARDiff` ユーティリティーを使います。この`JARDiff` ユーティリティーを使えば、JAR ファイル内の特定のファイルだけ (JAR ファイル全体ではない) をダウンロードできるので、ダウンロード時間が短縮されます。JNLP ファイルの設定の詳細については、参考文献を参照してください。

セキュリティー

Java Web Start に対応したアプリケーションを開発するかどうかを決める際に、最も重要な考慮点になるのは、おそらくセキュリティーでしょう。Java Web Start では、クライアント・サイド Java アプリケーションが、ローカル・マシンのさまざまなリソース (ファイルやクリップボードなど) にアクセスするような設定も可能です。しかし、この製品は Java 2 プラットフォームに対応しているので、Java 2 プラットフォームのセキュリティー方式を継承しています。すでに述べたとおり、JNLP ファイルの`security`属性では、Java Web Start から起動するアプリケーションのセキュリティー・レベルを指定できます。デフォルトでは、限定的な環境が使われることになり、アプリケーションからネットワークやローカル・マシンに対するアクセスは制限されます。そのような環境では、アプレットのサンドボックス環境と同様に、悪意のあるアプリケーションから被害を受けることはありません。

Java Web Start のセキュリティーにおいて、もう 1 つの重要な要素は、コードに対するデジタル署名です。アプリケーションの起動時または更新時には、アプリケーションの JAR ファイルがクライアント・マシンにダウンロードされます。そのときに、Java Web Start は、JAR ファイルのデジタル・コードを使って、最初の署名時から JAR ファイルが改変されていないかどうかを検証します。改変が検出されたり、ファイルに署名が付いていなかったりする場合は、Java Web Start からそのアプリケーションを起動することはできません。したがって、正しい署名が付いているアプリケーション以外は、ローカル・マシンにアクセスできないというわけです。

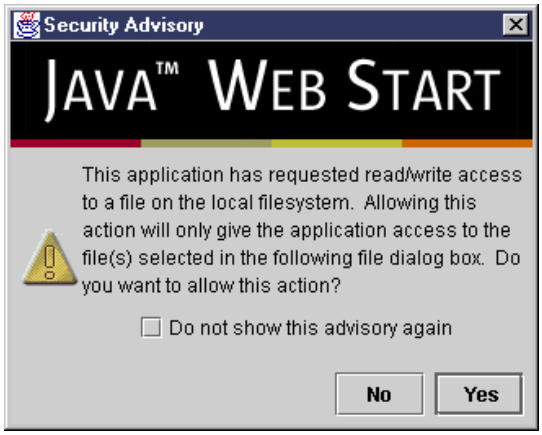
ユーザーがセキュリティーを設定する場合

ローカル・マシンに対する無制限のアクセスを必要とするようなアプリケーションを実行する場合は、ユーザーに対して、アプリケーションやベンダーの由来を示したダイアログがまず表示されます。そのダイアログで、ユーザーはアプリケーションに追加の権限を与えることができます。さらに、ローカル・マシンのリソースを使う必要があるアプリケーションに、署名付きの JAR ファイルがない場合は、Java Web Start の「Security Advisory (セキュリティー勧告)」ダイアログで、アクセス権を明示的に与えることもできます。たとえば、Windows OS のクリップボードに情報を貼り付ける必要があるクライアント Java アプリケーションの場合は、図 9 のようなダイアログ・ボックスが表示されます。

図 9. クリップボードにアクセスしようとするアプリケーションの場合に表示される「Security Advisory (セキュリティー勧告)」ダイアログ

クライアント・マシンのローカル・リソースにアクセスしようとするアプリケーションの場合は、さらにダイアログ・ウィンドウが表示されます。たとえば、ファイル・システムにアクセスする場合は、図 10 のような画面になります。

図 10. ファイル・システムにアクセスしようとするアプリケーションの場合に表示される「Security Advisory (セキュリティー勧告)」ダイアログ



ユーザーは、ローカル・マシンに対する次のようなアクセス権をアプリケーションに与えることができます。

- アプリケーションの現在の状態をローカル・マシンに格納する権限
- ローカル・ファイルの内容を表示する権限
- ローカル・マシンにファイルを保存する権限
- ローカル・マシンでファイルを開く権限
- ローカル・マシン上のランダム・アクセス・ファイルに対する読み取り / 書き込み権限
- ローカル・マシンから印刷する権限

JNLP API ライブラリーのさまざまなクラスは、開発者が信頼性の低い環境でシステム・リソースを使うことを可能にするので、各種操作を認めるかどうかを決める責任はユーザーにゆだねられることになります。

結論

Java Web Start は、クライアント・サイドの Java アプリケーションを開発したり配置したりするための独創的なソリューションを提供しています。ユーザーには、Java アプリケーションの起動・更新・管理のために設定できる多彩なオプションが用意されています。Java 開発者は、コードの修正・更新・配布のプロセスをあまり気にせずに、ソフトウェアの設計に専念できます。Java Web Start のテクノロジーはまだ新しいとはいえ、業界標準の確立を目指した試みでもあります。すでに定評のある Java 2 のセキュリティー方式をベースにしているので、Web とデスクトップの両方から Java アプリケーションを起動したり配置したりするための統一的な標準プロトコルとなる可能性を秘めています。

ダウンロード可能なリソース

内容	ファイル名	サイズ
Sample code	j-webstart.zip	1 KB

関連トピック

- Java Web Start をダウンロードできます。また、使い方を示したデモを見ることができます。
- Sun の「開発者ガイド Java Web Start」には、Java Web Start の詳しい技術情報が載っています。また、FAQ には、多くの一般的な質問に対する答えが記されています。
- Sun のプレス・リリースには、Java Web Start が及ぼす現在または将来の影しく説明されています。
- Rene Schmidt 氏のJNLP 仕様のガイドには、セキュリティー・プロトコルや API など、この仕様の技術的な情報が詳しく載っています。
- JNLP は、Java Community Process で開発されています。
- Joseph Sinclair 氏の「再訪: クライアント側の Java テクノロジー」という記事には、クライアント・サイドでの Java 言語の使用に関する別の視点が取り上げられています (developerWorks、2001年 3月)。