



Microsoft Java 仮想マシン 移行ガイド

バージョン 1.3.2

このドキュメントに記載されている情報は、このドキュメントの発行時点におけるマイクロソフトの見解を反映したものです。変化する市場状況に対応する必要があるため、このドキュメントは、記載された内容の実現に関するマイクロソフトの確約とはみなされません。また、発行以降に発表される情報の正確性に関して、マイクロソフトはいかなる保証もいたしません。

このホワイト ペーパーに記載された内容は情報提供のみを目的としており、明示または黙示に関わらず、これらの情報についてマイクロソフトはいかなる責任も負わないものとします。

お客様ご自身の責任において、適用されるすべての著作権関連法規に従ったご使用を願います。このドキュメントのいかなる部分も、米国 Microsoft Corporation の書面による許諾を受けることなく、その目的を問わず、どのような形態であっても、複製または譲渡することは禁じられています。ここでいう形態とは、複写や記録など、電子的な、または物理的なすべての手段を含みます。ただしこれは、著作権法上のお客様の権利を制限するものではありません。

マイクロソフトは、このドキュメントに記載されている内容に関し、特許、特許申請、商標、著作権、またはその他の無体財産権を有する場合があります。別途マイクロソフトのライセンス契約上に明示の規定のない限り、このドキュメントはこれらの特許、商標、著作権、またはその他の無体財産権に関する権利をお客様に許諾するものではありません。

© 2003 Microsoft Corporation. All rights reserved.

Microsoft、Windows、Windows Server、Visual J#、Visual J++、Visual C#、ActiveX、JScript、Visual Basic、Visual Studio は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

記載されている会社名、製品名には、各社の商標のものもあります。

目次

1	5
概要	5
対象読者	6
必要な予備知識	6
このソリューション ガイドの使い方	6
移行が必要な理由	7
システムへの影響	7
移行オプション	8
早急な対応が求められる IT 技術者	8
2	9
現状評価	9
アプレットとアプリケーション	9
問題を手動で特定する	9
アプレットの特徴	9
アプリケーションの特徴	10
Web ページ上の問題を特定する	10
Java コンポーネントを特定する	11
Diagnostic Tool for the Microsoft VM	12
Diagnostic Tool for the Microsoft VM を実行する	12
結果を一覧する	14
3	15
評価結果の分析	15
アプリケーションの種類	15
リッチ アプリケーション	15
ユーザー インターフェイス アプレット	16
表示アプレットまたはプレゼンテーション アプレット	16
シン クライアント アプレット	16
ユーティリティ アプレット	16
ビジネス ニーズを確認する	16
移行オプション	17
アプレットを削除する	17
セキュリティをロックダウンする (警告アプリケーションなし)	17
アプリケーションを移行またはリライトする	18
採用するアプローチの立案	21
4	22
セキュリティのロックダウン	22
MSJVM の指定	22
5	24
.NET Framework への移行	24
J# Browser Control と .NET Framework への移行	24
アプレットの移行	24
JLCA を使用した Visual C# と .NET Framework への移行	25
サポート クラス	26
変換レポート	26
6	27
DHTML へのリライト	27
7	29
代替 JRE への切り替え	29
代替 JRE	29
付録	30
A	31

MSJVM 開発者向け FAQ.....	31
B.....	34
Diagnostic Tool for the Microsoft VM.....	34
C.....	35
Java Language Conversion Assistant.....	35
サポート クラス.....	35
変換レポート.....	36
Visual J++ または Java 言語プロジェクトを Visual C# に変換する.....	36
Visual J++ プロジェクト.....	36
Java 言語プロジェクト.....	36
変換されなかったコードを手動でアップグレードする.....	37
JLCA を使ってアプレットを変換する.....	38
HTML ページをアップグレードする.....	38
要約.....	39
D.....	40
Java アプレットの Microsoft J# Browser Control への移行.....	40
はじめに.....	40
Java アプレットをコンパイルして J# Browser Control を作成する.....	41
コマンド プロンプトから Java アプレットをコンパイルする.....	41
Visual Studio .NET を使って Java アプレットを移行する.....	42
HTML ページの <APPLET> タグを <OBJECT> タグに変更する.....	42
<APPLET> タグと J# Browser Control の <OBJECT> タグ間での属性マッピング.....	44
J# Browser Control を展開する.....	45
複数 DLL のコントロールを展開する.....	45
複数の J# Browser Control を 1 つのライブラリにパッケージする.....	45
ユーザーのコンピュータ上で J# Browser Control を実行する.....	45
ベータ版ではサポートされない機能.....	47
まとめ.....	48

1

概要

Microsoft Corporation (以下「マイクロソフト」) の Java 実装の配布に関する係争を解決するための 2001 年 1 月の合意で、Sun Microsystems (以下「Sun」) およびマイクロソフトはマイクロソフトが Sun のソースコードおよび互換性テストスイートを使用して MSJVM をサポートする期間を制限することに合意しました。最初に設定された期間内に MSJVM の依存関係を除去できるかどうかに関心を寄せていた開発者および企業があったため、Sun およびマイクロソフトは MSJVM からの円滑な移行を推進できるように期間を延長することに合意しました。Sun およびマイクロソフトは MSJVM の現在のご利用者を円滑に移行する必要性を認識し、Sun の Java ソースコードおよび互換性テストスイートの使用に関するマイクロソフトのライセンス期限を延長することに合意しました。これにより、マイクロソフトは 2004 年 9 月 30 日まで MSJVM をサポートして、潜在的なセキュリティ問題を解決することができます。

Sun の Java ソースコードおよび互換性テストスイートを使用するためのマイクロソフトのライセンスの終了に備えて、マイクロソフトでは、係争が決着して以来、各製品における MSJVM の段階的な使用打ち切りを進めてきました。その一環として、Windows Server™ 2003 や Microsoft® Windows® 2000 SP4 を始めとした今後のリリース製品には、MSJVM を搭載しない運びとなりました。

現在 MSJVM に依存しているサイトやアプリケーションに、MSJVM をインストールしていないシステムからアクセスした場合、それらのサイトやアプリケーションは正しく動作しません。インターネットサイトでは、すべてのユーザーに動作上の問題が生じる可能性があります。イントラネットサイトでは、MSJVM を搭載していないマイクロソフトの新製品をクリーンインストールした新システムを使用しているユーザーに対して問題が発生すると考えられます (Windows オペレーティングシステムのアップグレードでは MSJVM は削除されないため、影響はありません)。

マイクロソフトでは、Microsoft Visual J++® および Microsoft SDK for Java を使って、MSJVM を使用するアプリケーションや Web サイトを構築しているお客様に対して、それらのアプリケーションや Web サイトについて MSJVM への依存関係をご確認いただき、代替ソリューションへの移行をご検討いただくように、お勧めしています。このドキュメントには、マイクロソフトがサポートしている選択肢である Microsoft .NET Framework への移行に加え、マイクロソフトのサポート外の選択肢についての情報も記載されています。マイクロソフトがサポート対象とするソリューションを選択されるお客様には、Microsoft Visual J#® .NET および Microsoft Java Language Conversion Assistant に関する情報を提供します。これらは、既存ソースコードを .NET Framework に移行し、また使用可能な拡張機能をご理解いただけるように、開発者の皆様を支援する製品です。Java プログラミング言語について、マイクロソフトが提供する拡張機能を使用していない場合でも、プランに最適な推奨オプションを選択していただくことができます。

このドキュメントの付録 A に記載する「開発者向け FAQ」では、MSJVM の削除や、移行計画に関する一般的なご質問への回答を一覧しています。

対象読者

このドキュメントはテクニカル ガイドです。MSJVM を必要とするサイトやアプリケーションまたは Java 言語コードを含むサイトやアプリケーションを確認し、代替ソリューションへの移行を計画して実施する業務担当者である IT 技術者や開発者向けに作成されています。

IT 技術者の方にとっては、このドキュメントは MSJVM への依存関係を確認する作業に役立ちます。IT ビジネス上の意思決定や技術戦略構築を担当する方々には、MSJVM からの移行戦略を決定づけるのに有用な情報を提供します。テクニカル面での意思決定を行う方々には、戦略実装に向けてのガイドとなります。

また、開発者の方々には、必要となる変更作業の種類を判断し、適切な作業へのアプローチを決定づけるためのガイドとなります。

必要な予備知識

Java に関する知識を、ある程度備えている必要があります。採用する移行ソリューションによって必要な知識は異なりますが、Java 以外に ECMAScript、DHTML (Dynamic HTML)、.NET Framework や C# 言語などのマイクロソフトソリューションに関する知識が必要になります。

このソリューション ガイドの使い方

このドキュメントは、MSJVM 移行プロジェクトを支援するためのガイドです。このガイドでは、現状評価に必要な情報、オプションの説明、企業組織に適した戦略の選択に関するガイド、各オプションに関する詳細情報などを記載します。付録では、これから 2004 年 9 月 30 日にかけてリリース予定の、マイクロソフトが開発中の移行ツールに関する情報を提供しています。

プロジェクト構造は、企業組織で採用しているプロセス モデルによって異なります。正式なプロセス モデルがない場合は、Microsoft Solutions Framework プロセス モデルに関する情報が <http://www.microsoft.com/msf> (英語情報) に記載されているので、ご利用ください。MSF プロセス モデルは、プロジェクトでよく発生する問題を回避し、プロジェクトを成功に導く可能性を高めることを目的としています。

次に、ガイドの章と付録の構成を一覧します。

第 1 章 概要 ガイドの概要と、マイクロソフトが MSJVM のサポートを中止することになった理由を説明します。

第 2 章 現状評価 業務上、移行プロジェクトが必要となる状況について説明します。既存システムに MSJVM への依存関係があるかどうかを確認することから着手します。

第 3 章 評価結果の分析 一般的に見られるアプレットやアプリケーションの種類、考えられる移行オプションを説明し、移行プロジェクト計画における検討事項の概略を説明します。

第 4 章 セキュリティのロックダウン レガシ システム上で、継続して MSJVM を使用するための手法を説明します。1 台のコンピュータ上で 2 つの仮想マシン (VM) を使用するときのいくつかの問題についても記載します。

第 5 章 .NET Framework への移行 .NET Framework へのコード変換を行う 2 つの自動化ツール (J# Browser Control と、C# への移行に使用する JLCA (Java Language Conversion Assistant)) について説明します。

第 6 章 DHTML へのリライト アプレットの置き換えに、DHTML フォームを使用する場合の問題について説明します。サンプルを紹介しているサイトのリンク集も含まれています。

第 7 章 代替 JRE への切り替え 代替となる JRE (Java Runtime Environment) の選択、インストール、導入にかかわる問題について説明します。

付録 A MSJVM 開発者向け FAQ 一般的な疑問点对する回答をまとめ、簡単に参照できるようにしてあります。

付録 B Diagnostic Tool for the Microsoft VM Diagnostic Tool for the Microsoft VM の

入手、インストール、および使用について記載します。

付録 C Java Language Conversion Assistant JLCA (Java Language Conversion Assistant) の入手、インストール、および使用について説明します。プロジェクトやアプレットの変換処理についても触れます。

付録 D Java アプレットの Microsoft J# Browser Control への移行 J# Browser Control を紹介し、既存の Java アプレットを移行する手順について説明します。

移行が必要な理由

MSJVM の将来は、2001 年 1 月の Sun との法的な係争の決着により、明確に決定づけられました。2003 年 10 月の Sun およびマイクロソフトのライセンス期間の延長の合意により、マイクロソフトは 2004 年 9 月 30 日まで MSJVM をサポートすることができます。2004 年 9 月 30 日以降、MSJVM に対してマイクロソフトが提供できるサポート内容は限定されることになりました。セキュリティ上の重大な脆弱性や、お客様にご利用いただくうえでの不具合点も含め、マイクロソフトは MSJVM のアップデートを行うことができなくなります。マイクロソフトは、Sun による法的な制約条件に基づき、可能な範囲でお客様へのサポートは継続してまいりますが、その範囲は大幅に制限される見込みです。これらの制約条件を踏まえ、Windows Server 2003 および Windows 2000 SP4 を含めて今後リリースされるマイクロソフトの新製品には、MSJVM が搭載されなくなります。

この情報は、2001 年 1 月の決着以来、各種メディアで取り上げられ、マイクロソフトの Web サイトでも公開していましたが、2004 年 9 月 30 日という期限は急速に近づいています。多くの OEM 製品では、いまだに MSJVM が同梱されています。

期限である 2004 年 9 月 30 日以降、MSJVM には、マイクロソフトが修正することのできないセキュリティ上の脆弱性の問題が存在する可能性があることとなります。MSJVM に依存関係のある業務アプリケーションを、代替ソリューションに移行できない場合、制限付きの通知に基づき、MSJVM を無効にする必要がある状況に追い込まれる可能性があります。結果として、業務上重要な役割を果たすアプレットや J++ アプリケーションが動作しなくなり、これらの業務アプリケーションを使用している業務や顧客に影響を及ぼす可能性があります。

システムへの影響

今回の決定事項が、使用中のシステムに影響を及ぼすかどうかを確認することが重要です。

MSJVM の特定のビルドを対象にして記述された Java アプレットやアプリケーションを使用しているケースが多いと考えられますが、クライアント コンピュータ上に仮想マシン (VM) があることを想定して、アプリケーションの展開を行っているケースもあります。MSJVM の存在を前提としているアプリケーションは、今回の決定事項の影響を受ける可能性があります。

特に、以下の環境を構築している場合は、その可能性が高まります。

- MSJVM の存在を前提としたアプレット (<APPLET> または <OBJECT> タグを使用) を含むサイト。これらのアプレットは、DLL や Microsoft ActiveX[®] コントロールによって起動されることがあります。
- Microsoft SDK for Java または Microsoft Visual J++ (すべてのバージョン) を使用して構築したアプリケーションやコンポーネント。これらのアプリケーションは、通常、CD に収録されて顧客やエンド ユーザー向けに出荷されます。ユーザーのコンピュータにダウンロードされて実行される場合もあります。

また、影響を受けるように思われるテクノロジーでも、それに該当しないケースもあります。たとえば、以下の場合は、今回の決定事項の影響を受けません。

- サイト上での JavaScript、Microsoft JScript[®]、または ECMAScript の使用
- サーバーサイドの Java テクノロジーを使用する Web サイト (JSP (Java Server Pages) やサーブレットなど。ただし、前述の影響を受けるテクノロジーを使用している場合は除く。)

移行オプション

MSJVM からの移行を計画する最初のステップとして、MSJVM への依存関係を確認することが必要です。この作業を支援するため、Microsoft では、後日、開発者や IT 技術者向けに、システムをスキャンし、MSJVM や Java 言語コードに依存しているリソースに関するレポートを生成するためのツールを提供する予定です。ツールの詳細については、「第 2 章 現状評価」を参照してください。

MSJVM への依存関係を確認したら、以下に一覧する対策の中から、1 つまたは複数の適切な策を講ずることをお勧めします。

- セキュリティのロックダウン、アプレットへのアクセス制限
- 代替ソリューションへの移行
 - Visual J# .NET または J# Browser Control
 - .NET Framework 環境で C# による構築 (Java Language Conversion Assistant を使用)
 - その他の表示技術 (DHTML、Flash や Shockwave などのサードパーティ製表示技術、代替 JRE など)

究極の選択オプションは、何も対策を講じないことです。これは、変更を加えないことなので、移行オプションではありません。しかし、特定のアプリケーションに対しては選択可能なオプションです。ここでは、考えられるこれらすべての対策について説明します。

また、Microsoft では、クライアントシステム上の MSJVM を削除または無効にするためのツールを提供することも視野に入れていますが、Windows 2000 コンピュータの場合、MSJVM を削除するツールを使用するには、サービス パック (SP4) をインストールすることが必要になります。現在、Microsoft は、リリースの詳細を最終的に決定し、ツールを提供できるように、作業を進めています。このツール (アンインストール ユーティリティ) は、クライアントコンピュータから MSJVM を削除することを目的に IT 部門向けに提供されますが、暫定的な措置として Internet Explorer のセキュリティロックダウン機能を使用する、という選択肢もあります。

早急な対応が求められる IT 技術者

Microsoft は IT 技術者の皆様に対し、少なくとも、MSJVM への依存関係の確認を即座に行うことを、強くお勧めします。これは、状況対処のために最終的に採用するソリューションにかかわらず、必要な作業です。どのようなソリューションを採用する場合でも、テストを実施し、システムへの影響範囲を理解することが求められます。この作業が、今後の実装計画の効果を大きく高めることにつながります。

依存関係に関する分析を行うことで、今後必要となる作業がないことや、MSJVM を無効にしても安全であることが明確になることもあります。しかし、分析を行わなければ、このようなことを知る手段はありません。ここでは、判明した依存関係に対処するために必要な技術情報を提供します。

2

現状評価

アプレットとアプリケーション

移行計画を立てる前に、現状の依存関係を確認することが必要です。業務に使用している、さまざまな Web サイトやアプリケーションは、それぞれの理由で MSJVM との依存関係を持つ可能性があるため、MSJVM からの移行プロセスの第 1 ステップとして、それらの依存関係を確認することをお勧めします。

アプレット、Java 依存型のアプリケーションおよびアプリケーション コンポーネントについて、すべて確認する必要があります。

- アプレットは、<APPLET> および <OBJECT> タグで識別されます。識別は直接および間接的に (DLL や ActiveX コントロール経由で) 行われます。
- Java 依存型のアプリケーションおよびアプリケーション コンポーネントは、MSJVM を使用するアプリケーションであり、MSJVM に直接または間接的に (DLL や ActiveX コントロール経由で) アクセスします。J++ ソース ファイルやバイナリ ファイルなどがこれに含まれます。バイナリ ファイルの場合、使用される拡張子として考えられるのは .exe、.ocx、.dll です。

アプレットは、Web プレゼンテーション メカニズムとして有名で、初めて配布されたのは 1996 年のことでした。しかし、アプレットはユーザーからの支持を得ることができず、JSP、ASP.NET、VBScript など別の種類のプレゼンテーション コードに取って代わられました。DHTML などのテクノロジーが発展する前は、アプレットが、ドロップダウン メニューやテキスト スクロールなどのアクティブな機能を提供できる唯一の手段でした。

問題を手動で特定する

Diagnostic Tool for the Microsoft VM が提供される前に、手動での依存関係評価に着手することをお勧めします。診断ツールの検索オプションに相当する内容を調べることを推奨いたします。Web ページの <APPLET>、<OBJECT>、<EMBED> タグを確認し、Java 依存型のアプリケーションおよびアプリケーション コンポーネントを検出することが必要です。

アプレットの特徴

アプレットの参照は通常、元の <APPLET> タグ、<OBJECT> タグ、<EMBED> タグのいずれかのタグからの参照により、行われます。<APPLET> タグは、次のような形式で表されます。

```

<APPLET code=Applet1.class
  height=200 width=320
  name=Applet1
  VIEWASTEXT>
  <PARAM NAME="foreground" VALUE="FFFFFF">
  <PARAM NAME="background" VALUE="008080">
  <PARAM NAME="label" VALUE="This string was passed from the HTML host.">
  No applet tag handlers installed.
</APPLET>

```

<APPLET> タグは、従来、アプレットを指すための最も一般的な方法でしたが、W3C は、<APPLET> タグを <OBJECT> タグに置き換えるように推奨しています。<OBJECT> タグを使用すると、アプレット コンテンツだけでなく、何種類かのダイナミック コンテンツを指定できます。<OBJECT> タグの構造は、次のような形式で表されます。

```

<OBJECT CLASSID="clsid:08B0E5C0-4FCB-11CF-AAA5-00401C608501"
  height=200 width=320
  VIEWASTEXT>
  <PARAM NAME="code" VALUE="Applet1.class">
  <PARAM NAME="type" VALUE="application/x-java-applet">
  <PARAM NAME="scriptable" VALUE="true">
  <PARAM NAME="foreground" VALUE="FFFFFF">
  <PARAM NAME="background" VALUE="008080">
  <PARAM NAME="label" VALUE="This string was passed from the HTML host.">
  No MS VM installed
</OBJECT>

```

classid 属性は必須です。上記の例の値は、MSJVM を指しています。その他の JRE の場合は、指定する値が異なります。

Web 上でコンテンツにサービスを提供するほとんどのページには、これらの参照が含まれます。これには、HTML、JSP、ASP.NET ページなど、静的な Web ページと動的な Web ページの両方が含まれます。企業組織の中には、データベースや、他の形式から変換したデータを使用してダイナミック HTML ファイルを作成している場合もあります。ASP および ASP.NET の場合は、次のようなコードを含むスクリプトを探してください。

```
document.write(<APPLET>...)
```

このような <APPLET> の出現部分はすべて確認することが必要です。これらのタグを確認するには、HTML ページのテキスト (静的、動的を問わず、HTML ページまたはソース ファイル) の検索が必要になります。

アプリケーションの特徴

Java 依存型のアプリケーションおよびアプリケーション コンポーネントは、MSJVM を使用するアプリケーションであり、MSJVM に直接的または間接的に (DLL や ActiveX コントロール経由で) アクセスします。実行可能ファイル、DLL、および OCX ファイルの特定は困難ですが、MSJVM 対応の診断ツールは、そのようなファイルの検索に信頼性の高いツールとしてご利用いただけます。コンパイル済みの Java ファイル (.class および .jar 拡張子ファイル) は、16 進数の値 0xCAFEBABE で始まるマジック番号を検索することで、特定が可能です。ファイルの拡張子やマジック番号を使用した検索は、コンパイル済み Java ファイルの検出に役立ちます。16 進数の値を使用すると、Java バイナリファイルの検索には有効ですが、Java バイナリファイルを呼び出すファイルを特定することはできません。Java ソース ファイルの検索では、jvc.exe コンパイラに /x-フラグを指定して作成したビルド ファイルも検索対象となります。/x- フラグは、コンパイラに対して、Microsoft Java 拡張機能を使用するように明示的に指示するものです。

Web ページ上の問題を特定する

ローカル サーバー上での問題特定は簡単です。

単純なテキスト検索機能を使用して、静的な Web ページに対し、"applet" および "object" という用語のインスタンスを検索します。たとえば、HTML ファイルのディレクトリで <APPLET> タグのインスタンスを検索するには、次のように CMD.EXE コマンドを発行します。

```
Find /I "applet" c:\mydir\*.*
```

動的に生成されるページに対する検索を行うには、<APPLET> タグと <OBJECT> タグのインスタンスを検索することが必要です。さらに、最終的な表示コンテンツに <APPLET> タグと <OBJECT> タグを表示する可能性のある要素についても、検索が必要です。データベース エントリをベースに作成したページについては、<APPLET> および <OBJECT> タグのインスタンスについて、データベースを検索してください。他の入力ファイルから変換したページを使用している場合、変換後に <APPLET> タグになる "トリガ" が含まれていないかを確認する必要があります(変換エンジン変える方法もあります)。Web 上に発行しているその他のコンテンツの種類も調べて、現状評価に最適なアプローチを決定してください。

ローカル サーバー以外のサーバー上にあるファイルの問題を特定するには、各 URL にあるファイルを調べる必要があります。

Java コンポーネントを特定する

特定の必要な依存関係の種類は 3 つあります。まず、Java 言語コードの依存関係を確認することをお勧めします。次に、それらの中で、MSJVM を指定するコンポーネントとアプリケーションを特定し、MSJVM に固有な特徴に依存していないかどうかを調べます。最後に、MSJVM に固有な特徴に依存するコンポーネントとアプリケーションを特定し、MSJVM を指定していないかどうかを調べます。

MSJVM への依存関係があるコンポーネントやアプリケーションを事前に検出するには、実行可能ファイル、ライブラリ ファイル、コントロール ファイルの内部にあるバイナリ コードを調べる必要があります。この作業は、Diagnostic Tool for the Microsoft VM を使用すると容易に行うことができますが、手動でこれらの調査を行う場合は、以下に列挙する内容について調査してください。

- Java コンポーネントおよびアプリケーション用のバイト コードである、拡張子 .class のファイルが格納されていないか。拡張子が .class のファイルはすべて依存関係を持ちます。
- COM オブジェクトとして起動される Java アプリケーションが、レジストリに一覧されていないか。
- Java コンポーネントを検索するには、16 進数値 0xCAFEBABE で始まる実行可能ファイル、ライブラリ ファイル、コンポーネント ファイルの存在をコンピュータ上でチェックします。
- コンピュータによっては、レジストリ値に "msjava.dll" という値を持つ CLSID エントリを検索する必要があります。検索は reg.exe などのツールを使用して、バッチ スクリプトで実行できます (Windows XP および Windows 2000 リソース キットに同梱されています)。reg.exe ツールは、現在のコンピュータだけを検索対象とします。検索を実行する独自のアプリケーションを作成しなければならない場合もあります。レジストリ操作を処理することが意図されている MSF の Registry および Query クラス モジュールを使用することは、このようなツールを記述する際に便利です。

他のアプリケーションから Java オブジェクトへの呼び出しを特定する

Java クライアントを呼び出す ActiveX クライアントは、JNI (Java Native Interface) と RNI (Raw Native Interface) のいずれかを使用できます。Microsoft Visual Basic[®]、VBScript、または ASP における ActiveX クライアントは、Java アプレットを実行するために MSJVM を起動する必要があります。MSJVM の起動は、コード内で XJB.JClassFactory オブジェクトの XJBInit() メソッドを呼び出すことで実行されます。

Java オブジェクトは、J++ SDK の一部である javareg ツールを使用して、COM オブジェクトとして登録することもできます。レジストリには、COM オブジェクトの CLSID 値が格納されます。COM オブジェクトとして登録される Java オブジェクトの CLSID 値は常に msjava.dll です。

Diagnostic Tool for the Microsoft VM

MSJVM の段階的廃止に備え、MSJVM への依存関係があると思われるすべてのアプリケーションおよび Web ページを特定できるようにする必要があります。時間のかかる単調な作業ではありますが、これらのファイルを手作業で検索することをお勧めします。

これらのファイル検索処理を簡易化するため、マイクロソフトは、近日中に "Diagnostic Tool for the Microsoft VM" をリリースする予定です。リリース日の最新情報については、MSJVM Web サイト (<http://www.microsoft.com/japan/java/default.asp>) をご覧ください。このツールは、指定のコンピュータをスキャンし、MSJVM に依存するアプリケーションおよび Web ページを検索します。スキャン結果は、1 つのレポートにまとめて表示されます。このツールを今すぐ入手する必要がある場合は、マイクロソフト顧客担当者または、各地域のマイクロソフトにお問い合わせください。

実際に使用するときには、診断ツールを何回か実行することが必要になります。移行を試みる前の現状評価、プロジェクトの進行中 (現状の指針として)、新規展開中 (全アプレットに対処が施されたことを確認するため)、の 3 段階での実行が必要です。

ここでは、Diagnostic Tool for the Microsoft VM の概要について説明します。Diagnostic Tool for the Microsoft VM のインストールについては、付録 B を参照してください。ツール実行に関する手引きについては、ツールに付属のドキュメントである『Diagnostic Tool for the Microsoft VM User Manual』を参照してください。

Diagnostic Tool for the Microsoft VM を実行する

Diagnostic Tool for the Microsoft VM を実行するには、Diagnostic wizard を使用する方法と、コマンドライン ツールとして使用する方法の 2 とおりがあります。どちらの方法でも、同じタイプの依存関係が検出されます。

検索対象となる依存関係のタイプには、次のようなものがあります。

- MSJVM の存在 (レジストリ エントリも含む)
- MSJVM 依存型の実行可能アプリケーション (拡張子が .exe、.dll、.ocx のファイルを含む)
- コンパイル済み Java ファイル (拡張子が .class のファイル)
- Web ページ ファイル (拡張子が .html、.jsp、.php、.asp のファイルなど) - <APPLET> タグの検索、または、<OBJECT> や <EMBED> タグで MSJVM の起動が行われていないかどうかの検索

初期段階の現状評価では、すべてのコンピュータに対して検索処理を実行してください。プロジェクト進行に伴い、解決済みの問題と排除が必要な問題が明らかになり、検索対象を狭めていくことができます。

Diagnostic wizard を実行する

Diagnostic wizard により、現状評価を実行するコンピュータと検索対象とする依存関係の 2 つの制御情報を指定するように求められます。

ローカル ネットワーク上であれば、任意のコンピュータを指定できます。ただし、このツールを実行するには、管理者レベルの権限が必要です。

メモ 検索対象とするコンピュータは、既定では 1 台も選択されません。これは、既定オプションとしてはローカル コンピュータの検索は実行しないことを意味します。検索処理を実行するコンピュータを指定する必要があります。最初に表示されるダイアログ ボックスでは、ローカル ドメインにあるコンピュータを検索するようにメッセージが表示されます。最後に表示されるダイアログ ボックスでは、URL のセットを検索するようにメッセージが表示されます。

コマンドラインから診断ツールを実行する

使用法は、以下のとおりです。

```
JavaDiag [@argfile] [/?|h|help] [/v|version] [/vm] [/deps] [/java] [/vj]
[/webfile [/Webextfile <value>]] [/urlfile <value> [/crawldepth <value>]]
[/hostfile <value>] [/reportfile <value>]
```

表 2.1 に、パラメータを一覧します。

表 2.1 診断ツールのコマンドライン パラメータ

パラメータ	定義
@argfile	指定の argfile から引数を読み取る。
/?	使用方法を表示する。
/v	バージョンを表示する。
/vm	バージョンを表示する。
/deps	Java VM に依存関係のあるプリケーションをスキャンする。
/java	Java クラス ファイルをスキャンする。
/vj	VJ++ 拡張機能を使用した Java クラス ファイルをスキャンする。
/webfile	Web ファイルのスキャン - Java への依存関係があるか、Web ファイルをスキャンする。
/webextfile <listfile>	Web ファイルのスキャン - スキャンを実行するファイル拡張子を一覧したテキスト ファイルの名前を指定する (1 行に 1 つの拡張子を、ピリオドは付けずに明記する)。
/urlfile <listfile>	URL スキャン - スキャンを実行する URL を一覧したテキスト ファイルの名前を指定する。
/crawldepth <value>	URL スキャン - スキャンを実行するページのリンクの深さの最大値を指定する (網羅的にすべてをスキャンする場合は、-1 を指定)。
/hostfile <listfile>	すべてをスキャン - スキャンを実行するコンピュータのホスト名を一覧したテキスト ファイルの名前を指定する (1 行に 1 つのホスト名を指定)。
/reportfile <outfile>	スキャン結果を報告する出力ファイル名を指定する (拡張子は .htm または .html。このオプションを使用するには、MSXML 4.0 が必要)。

以下に、すべてのオプションを指定したコマンドラインのサンプルを示します。このサンプルでは、ファイル Webext.txt に一覧されている拡張子を持つすべてのファイル、および、ファイル urlfile.txt に一覧されているすべての URL に対して、スキャンが実行されます。出力ファイルは、report.htm という名前で、現在のディレクトリに格納されます。

```
C:\> JavaDiag /hostfile hosts.txt /vm /deps /java /vj /Webfile /Webextfile  
Webext.txt /urlfile urlfile.txt /crawldepth 3 /reportfile report.htm
```

結果を一覧する

自動生成される HTML 形式のレポートには、診断結果が含まれています。ツールにより、対象コンピュータごとに結果の概要が報告され、指定したスキャン オプションごとに詳細が報告されます。結果レポートを入手したら、次章の説明を参考に、その内容を分析してください。

3

評価結果の分析

依存関係の評価結果リストを入手したら、ビジネス ニーズやタイプに応じて、その内容を分類します。依存関係の種類（アプリケーションまたはアプレット、ナビゲーション支援またはリッチ アプレットなど）によって、移行の方法が決まります。業務上のニーズ（ビジネス ニーズ）によって、採用するアプローチを決定します。

アプリケーションの種類

表 3.1 は、最も一般的なアプリケーションの種類を一覧したものです。サンプルごとに、推奨されるソリューションについても明記しています。対応方法が適切であれば、ほとんどの移行オプションは適切に動作しますが、オプションを組み合わせた対応を行う場合もあります。表に一覧したアプレットとアプリケーションについては、以降のセクションで詳しく説明します。

表 3.1 アプレット、アプリケーション、解決方法

アプレットまたはアプリケーション	例	解決方法
リッチ アプリケーション	ゲーム	.NET Framework への移行 (C#、J#) または Flash
ユーザー インターフェイス アプレット	テキストスクローラ、メニュー	DHTML、Windows フォームをベースにした C#、J# Browser Control
表示アプレットまたはプレゼンテーション アプレット	アニメーション	Flash、DHTML、Windows フォームをベースにした C#、J# Browser Control
シンクライアント アプレット	データベース照会用のフロントエンド	DHTML、C#、Windows フォームをベースにした C#、J# Browser Control
ユーティリティ アプレット	電卓、チャット クライアント	DHTML、Windows フォームをベースにした C#、J# Browser Control、.NET Framework への移行

これらのアプレットやアプリケーションに対するその他の解決方法として、代替 JRE が挙げられます。詳細については、「第 7 章 代替 JRE への切り替え」で説明します。

リッチ アプリケーション

機能が豊富なアプレットやアプリケーションには、応答時間、表示機能、データ転送、グラフィック

スの品質など、特有のニーズがあります。現在のシステム環境が、主として Windows をベースとしている場合、.NET Framework への移行を検討することが推奨されます。.NET Framework では、緊密性の高い高速な実行可能ファイルが作成されます。詳細については、「第 5 章 .NET Framework への移行」を参照してください。

クロス プラットフォーム対応のアプリケーションの場合、Flash や代替 JRE などを使用するソリューションが必要になると考えられます。いずれのソリューションを採用する場合でも、適切なランタイム環境をダウンロードする必要があります。

ユーザー インターフェイス アプレット

テキスト スクローラ、メニュー、ロールオーバー ボタンなどのユーザー インターフェイス アプレットは、Windows フォームをベースにした C#、J# Browser Control や DHTML を使用して、簡単に実装できます。その他の技術を使用することも可能ですが、今回の対処法としては、これらの手法が最適です。ロールオーバー ボタン、テキスト スクローラ、ポップアップ メニューのサンプルコードへのリンクについては、「第 6 章 DHTML へのリライト」で説明します。

表示アプレットまたはプレゼンテーション アプレット

アプレットは、グラフィックスを重視した動画イメージである傾向があります。DHTML コードを使用すると、イメージの単純な変換や表示と非表示の制御などを含め、何種類かの動画 (アニメーション) を実行できます。詳細については、「第 6 章 DHTML へのリライト」を参照してください。グラフィックスも、.NET Framework への移行により対処できます。詳細については、「第 5 章 .NET Framework への移行」を参照してください。クロス プラットフォーム上で、より複雑なグラフィックスを扱う必要がある場合は、サードパーティのレンダリング ツールの使用を検討することをお勧めします。

シンクライアント アプレット

Java アプレットは、データベース照会のフロント エンドとして動作するフォームとして使用されることがあります。このようなアプレットでは、データ再編や予備的な変換処理を行うことがあります。ほよんどの場合、サーバー アプリケーションなどに情報を渡しています。変換処理の複雑さに応じて、DHTML が複数のプラットフォームに及ぶ変換処理を実行したり、C# で記述された共通言語ランタイム (CLR : Common Language Runtime) のフォーム (JLCA を使って変換されます) もしくは J# Browser Control が変換処理を実行したりします。

ユーティリティ アプレット

"ユーティリティ アプレット" には、さまざまな小規模アプリケーションが含まれます。たとえば、電卓やチャット クライアントなどです。アプリケーションの複雑さによって異なりますが、電卓とチャット クライアントのいずれも DHTML による移行が可能です。別の言語を使ってアプリケーションを最初から作り直すのではなく、ネイティブ テクノロジを使用するアプローチが適している場合もあります。たとえば、チャット クライアントを新たに記述するのではなく、Windows Messenger や、エンドユーザーのコンピュータ上にある同等のツールを起動するスクリプトを記述することが考えられます。

ビジネス ニーズを確認する

アプレットやアプリケーションを確認する作業よりも、ビジネス ニーズを見極める作業の方が困難であることが考えられます。ビジネス ニーズにより、移行プロジェクトの優先順位や適用するリソースが決まります。以下に、移行計画を決定する上で調査が必要と思われる問題について、概要を列挙します。

- **顧客とユーザー** Web サイトやアプリケーションを使用するのはだれでしょうか。支払いを行うのはだれでしょうか。ユーザーが求める機能は何でしょうか。どの企業からソリューションのサポートを受けますか。顧客のシステム環境が Windows ユーザーと Macintosh ユー

ザーが半々の割合で構成されている場合、全員が Windows ユーザーである顧客ベースシステムと比較して、Web サイトに対するユーザーの要求には多様性が見られます。イントラネット サイトの移行処理では、インターネット サイトの移行処理よりも、ユーザーのベースシステムは均一的になります。エクストラネット (パートナー) サイトは一般に多様性があるといえます。顧客 (購入者) が実際の使用者 (ユーザー) でない場合、顧客側にその他のニーズがあるでしょうか。たとえば、IT 部門の顧客である場合、展開や管理の容易さなど、付加的な要望を抱いているでしょうか。

- **ライフ サイクル** 対象の Web サイトは現在、そのライフ サイクルのどの地点にあるでしょうか。使用されていない Web サイトや Web ページは、これ以上の作業は不要になることがあります。つまり、移行ではなく、「閉鎖」という戦略をとります。開発中の Web サイトやアプリケーションは、移行プロジェクトにより、その完成が遅れる可能性があります。堅固な顧客層に支えられている成熟した Web サイトでは、慎重な移行計画が必要です。
- **業務目標** Web サイトを使って達成しようとしているものは何でしょうか。現状は、その目標に見合ったものでしょうか。今後 5 年間で、達成しようとしているものは何でしょうか。行うべきことを行っていない Web サイトは、手直しが必要です。Windows の既存投資をベースにした対応を行うのが適している計画が企業組織内にあるでしょうか。このようなケースに該当する場合、.NET Framework への移行がよりよい選択であるといえます。
- **移行規模** 作業規模はどのくらいですか。変換が必要な Web ファイルはいくつありますか。アプレットは複雑ですか。それとも単純ですか。変換するファイルの数は、数百にも達しますか。数千ですか。数十万ですか。移行プロジェクトは、Web サーバーの構造全体に影響が及ぶ可能性があります。ファイルの所有権がどのようになっているかも考慮することが必要です。現在の所有者が移行作業を担当しますか。それとも、移行作業を担当するチームを設けますか。
- **リソース** 移行プロジェクトに、何名のリソースを配置できますか。

上記 5 つのカテゴリに一覧した質問への回答を導き出すことが、移行戦略立案の手引きとなります。

移行オプション

移行オプションには、それぞれ、長所と短所があります。以下に、これらのオプションについて説明します。オプションの詳細については、このガイドの以降の章で取り上げます。

アプレットを削除する

アプレットの中には、ページ上で有意な働きをしないものや、代替となるものを開発しても妥当性を十分に満たさないものがあります。ページに大きな影響を及ぼすことなく削除できるアプレットや、削除しても差し支えないページについては、削除を検討してください。<APPLET> </APPLET> テキストを削除またはコメント文にします。

メリット 実装およびテストが簡単です。

デメリット 一般に、ページ上の一部の機能が失われます。

展開上の検討事項 ファイルを展開するだけで、それ以外は特にありません。

セキュリティをロックダウンする (警告アプリケーションなし)

一部のアプリケーションについては、MSJVM の使用の継続を検討することが考えられます。Visual J++ 6.0 および Microsoft SDK for Java の使用許諾契約により、これらのアプリケーションのランタイム パージョンの一環として MSJVM の頒布は継続されます。これは、使用許諾契約 (EULA) に基づくもので、既にインストールされている MSJVM を削除しないように選択することも可能です。前に述べたとおり、MSJVM を現在の場所に残すことは、将来的にそのシステムをセキュリティリスクにさらすことになり、お勧めできません。

しかし、IE (Internet Explorer) には、MSJVM の適用範囲を無効化または制限できるセキュリティ設定機能があります。脆弱性を最低レベルに抑えるために、すべての Web サイトに対して

MSJVM を無効に設定することも可能です。別の方法として、MSJVM の使用は継続するけれども、特定サイトにおける Java の使用を制限することで、将来的なセキュリティ問題の発生を低減する、という選択肢もあります。インターネット ゾーンにおいて Java を無効にしてから、ユーザーまたは管理者 (サイト ポリシーによって異なります) が信頼済みサイトに対して Java を有効にして、[信頼済みサイト] の一覧に、アプレットをサポートするサイトを追加できます。

このアプローチでは、アプリケーションへの変更は不要ですが、エンド ユーザーの環境に MSJVM をインストールしておくことが必要になります。これは、インターネットで使用するアプリケーションには適したアプローチではありません。

このソリューションが適しているのは、使用頻度の低いアプリケーションや近い将来に使用期限が迫っているアプリケーションで、かつ、限定されたユーザーベースまたは将来起こり得るセキュリティ問題に関係のないユーザー ベースで使用されているアプリケーションです。たとえば、もうすぐ使用されなくなるけれども、将来的なサポートやメンテナンスを理由に継続が必要な古いアプリケーションなどは、セキュリティ ロックダウンによる隔離が考えられます。

メリット コストがかかりません。

デメリット クライアント コンピュータ向けであり、イントラネットには適しています。将来的なセキュリティ リスクを低減できますが、排除することはできません。

展開上の検討事項 セキュリティ ロックダウンは、各エンド ユーザーのコンピュータ上で行われます。ネットワーク構成によって異なりますが、IT によりグループ ポリシーとして設定することが可能です。また、ログイン スクリプトや .reg ファイルを使って設定内容を組織全体に反映することも可能です。

セキュリティ ロックダウンの詳細については、「第 4 章 セキュリティのロックダウン」を参照してください。

アプリケーションを移行またはリライトする

前に述べた 2 つの方法を利用できないサイトやアプリケーションでは、別のソリューションに移行するために、コードを移行またはリライトする必要があります。このオプションは、現在のアプレットやアプリケーションと同一の機能を提供しながら、より高いセキュリティを実現します。

.NET Framework への移行により、Window プラットフォームの強力な機能を利用できます。.NET Framework の本質的機能により、アプリケーションの拡張や統合を容易に行うことができます。マイクロソフトにより、サポート サービスも継続して提供されます。

ビジネス ニーズによっては、.NET Framework 戦略は適さない、クロスプラットフォーム ソリューションが必要になることがあります。このような場合は、少なくとも以下の 3 つのオプションがあります。

- DHTML を使用します。
- その他の表示技術 (Flash や Shockwave など) で使用できるように、アプリケーションとアプレットをリライトします。
- 代替 JRE をインストールします。

.NET Framework への移行と Visual J# (J# Browser Control)

Java アプレットもしくはアプリケーションは、Visual J# コンパイラ (vjc.exe) を使ってコンパイルすることで、J# にアップグレードできます。これにより、マネージ ライブラリに対応可能となります。バイトコード (.class ファイル) だけで構成される Java コンポーネントの場合は、Visual J# バイナリ変換ツール (Jblmp.exe) を使って、マネージ ライブラリに変換できます。アプレットを移行する場合は、J# コンパイラその他、J# Browser Control ランタイムも必要となるでしょう。

Java アプレットを J# Browser Control にコンパイルするとき、Java アプレットのソース コードを変更する必要はありません。J# Browser Control ランタイムは、(java.applet パッケージも含めた) JDK 1.1.4 レベル パッケージとほぼ同等の機能をサポートします。

メリット 迅速な実装、既存の Java コードの再使用が可能です。

デメリット Java データ型やクラス ライブラリに制限されます。自動化変換処理を行う場合でも、コードの一部リライトが必要です。

スキル要件 .NET Framework および Visual J#.NET の知識が必要となります。

展開上の検討事項 移行したアプレットは、Web サーバーにインストールされ、J# Browser Control を呼び出します。エンド ユーザーのコンピュータには、.NET Framework および J# Browser Control のランタイムをインストールすることが必要になります。

.NET Framework への移行と Visual C# (JLCA)

Java アプレットやアプリケーションは、JLCA (Java Language Conversion Assistant) などのツールを使用し、Visual C# を使用したアプリケーションにリライト可能です。コードの大部分 (ほとんどの場合、90% 以上) は、自動的に変換されます。残りの部分は、手動で変換することが必要です。

メリット ネイティブ .NET Framework API にアクセスするため、.NET Framework との完全統合が可能です。自動化ツールを使用することで、合理的かつ迅速な変換が達成されます。この移行戦略に基づいて変換されたアプリケーションは、変換前より速く動作し、サイズも小さくなります。.NET Framework 全体を使用する将来的な拡張も可能です。Windows フォームやインターフェイスの採用により、洗練されたユーザー インターフェイスを実現できます。

デメリット 自動化変換処理を行う場合でも、コードの一部リライトが必要です。

スキル要件 .NET Framework および Visual C# の知識が必要です。

展開上の検討事項 移行したアプリケーションは、Web サーバーにインストールします。エンド ユーザーのコンピュータには、.NET Framework ランタイム環境をインストールすることが必要です。

表 3.2: それぞれのオプションに関するメリットおよびデメリットのまとめ

オプション	メリット	デメリット
IE セキュリティゾーン	<ul style="list-style-type: none"> 容易に実施が可能。 アプリケーションを継続して実行することが可能。 	<ul style="list-style-type: none"> すべての JVM を無効にする。 短期的なソリューション。 対象としているそれぞれのサイトを知ることが必要。
.NET Framework への移行	<ul style="list-style-type: none"> マイクロソフトがサポートしているプラットフォームへの移行。 次世代の開発環境が提供されている。 移行を容易するためのツールが用意されている。 	<ul style="list-style-type: none"> ソースコードへのアクセスが必要。 .NET Framework 実行環境のユーザーのマシンへのインストールが必要。

DHTML へのリライト

ここでは、"DHTML" という用語は、ドキュメント オブジェクト モデル (DOM) における CSS、スクリプト、HTML の組み合わせを含めた広範な意味で使用しています。DHTML の技術は過去数年で成熟したもので、旧式の Web ページが配置されていた当時には実行可能なオプションではありませんでした。しかし、最も単純なナビゲーション アイテム (メニュー、ロールオーバー、グラフィック効果、簡単な電卓やチャット クライアント) であれば、DHTML で簡単に実装できるようになりました。ナビゲーションやユーザー インターフェイスに関するタスク専用の、複雑さが低～中程

度のアプレットを変換するには、このアプローチは適しています。一般的なアプレットを DHTML にリライトするサンプルについては、「第 6 章 DHTML へのリライト」を参照してください。

メリット アプレットと MSJVM を読み込む必要がないため、大幅な高速化が達成されます。クロスプラットフォームの互換性も維持されます。元のアプレットの再使用頻度は高く、共通に使用されているため、基本となるアプレットの種類のリライトが完了すると、一般に、実装は速やかに行われます。

デメリット リッチ アプレットや広範なデータ操作には適していません。アプレット コードを、完全に DHTML にリライトする必要があります。ブラウザとプラットフォームのそれぞれの組み合わせに対してテストを行うため、テスト作業には時間がかかります。Microsoft からのサポートはありません。

スキル要件 DHTML に関する知識が必要です。

展開上の検討事項 移行したファイルは、Web サーバーにインストールします。

代替 JRE の使用

Java アプレットやアプリケーションは異なる Java 仮想マシン上でも実行できます。Windows プラットフォームで利用できる代替の選択肢がいくつかあります。JRE の利用可能性については、慎重に検討してください。仮想環境が異なると、Java 標準の異なるバージョンを実装していることが考えられるためです。外部 (インターネット) に接続するサイトでは、アプレット参照に、ランタイムバージョンを取得する場所へのリンクが含まれていることが考えられます。<OBJECT> タグから参照先を見つける方法ではなく、明示的に JRE をダウンロードする方法を採用することも可能です。

メリット 一部のアプリケーションには変更が必要とされますが、実際には Java コードが再利用されます。

デメリット 全システムにおいて、指定の JRE をダウンロードすることが必要です。JRE が異なると、Java 標準の異なるバージョンを実装している可能性があります。このソリューションでは、サポートは複雑になります。他のベンダの製品が原因で問題が生じることがあるためです。何らかの理由により、1 台のコンピュータに複数の JRE (Java Runtime Environment) を搭載している場合、アプレットを適切に実行するため、正しい JRE とそのバージョンを明示的に指定することが必要になります。

スキル要件 Java の実装に関する知識が必要です。

展開上の検討事項 新しい JRE (Java Runtime Environment) を指すように、Web ファイルの更新が必要になる場合があります。エンド ユーザーは、JRE のアップグレードが必要です。

代替ソリューションへのリライト

多数のアニメーション処理を必要とするアプリケーションや、多数のグラフィックスを含むアプリケーションで、クロスプラットフォームの実装が必要になる場合があります。このような場合、Flash などのサードパーティの表示技術が適切であることが考えられます。

メリット クロスプラットフォームの実装、アプリケーションの優れた応答時間を実現します。アニメーションやグラフィックスに最適なツールもあります。

デメリット 完全なリライトが必要になりますが、組織内に専門知識を備えた人材がない場合があります。Flash などのツールは、開発プラットフォームというよりも、グラフィックアーティストのツールであると見なされることも多いようです。バージョンの問題もあります。

スキル要件 選択したソリューションに関する知識が必要です。

展開上の検討事項 依然としてユーザーのコンピュータにプラグインが必要です。Flash は広く普及していますが、その他のプラグインでは、その使用可能性 (可用性) が問題になることがあります。

メモ マイクロソフトとしては他社のソリューションについてセキュリティと信頼性を保証することはできませんが、マイクロソフト以外のソリューションを導入するという方法も可能です。サード

パーティのソリューションを使用する場合は、必ずテストを行ってから移行に着手してください。

何もしない

リソースを配置する必要がないため、最もコストのかからないアプローチであるといえます。マイクロソフトは、このアプローチをお勧めしません。少なくとも、影響のあるサイトでは、セキュリティのロックダウンを行ってください。この方法は、ビジネスを遂行するような重要なサイトでは採用しないようにしてください。一般にこのアプローチは、今後使用されなくなる、使用期限が迫っている、現在使用されていないサイトや、アプレットの使用目的があまり重要でないサイトに対する選択肢です。

メリット コストがかかりません。

デメリット 生産性の低下や、顧客またはユーザーの満足度の低下を生じる可能性があります。インストール先のシステムが、将来的に潜在的なセキュリティ リスクにさらされる可能性があります。

展開上の検討事項 特にありません。

採用するアプローチの立案

ガイドをここまで読み進めていただいた読者の方は、影響を受けるサイトやファイルのリスト、使用しているアプレットの種類、選択可能なアプローチ (解決方法)、各サイトにおける業務上の優先順位について、情報を入手されていることと思います。今こそ、どのように移行を進めていくかを決定するときです。

アプレットやアプレットの種類ごとに、採用する移行オプションを決める必要はありません。そのようなオプションは、まだ明らかになっていないこともあります。どのような種類のアプレットでも、これらのオプションによって移行することが可能です (何もしない場合やアプレットを削除する場合は除きます)。企業組織で決定すべき事項は、ビジネス ニーズです。

初期段階では、パイロット プロジェクトを設定して、技術的なテストを行うことも有用な方法です。パイロット プロジェクトには、対象となる各種類のアプレットのサンプルを含め、範囲を限定して実行することが必要です。パイロット プロジェクトの結果は、単一の Web サイト上で確認する必要があります。また、パイロット プロジェクトから、現実的なプロジェクト計画を引き出せるようにすることも必要です。

いくつかの変換処理をテストしてみてください。最小限のコードの変更で J# Browser Control を使って Java アプレットを .NET Framework にアップグレードできる場合は、このツールを採用してください。広範なコード変更が必要になる場合や、将来的に C# での開発を予定している場合は、JLCA を使用して C# に変換してください。 .NET Framework を使用すると、ソリューションを組み合わせることができます。

インターネットとイントラネットの両方に接続する表示アプレットを使用している場合は、インターフェイスの整合性が最も重要になると考えられます。このような場合は、DHTML 変換が最適なソリューションとなり得ます。

企業組織の立場で考えると、ソリューションは 1 つに絞って選択することに大きな利点があります。作業全体への目が届き、必要とされる新しいスキルの数も最小限に抑えることができるからです。一方、ソリューションを組み合わせると必要となる作業量は増えますが、2 つのソリューションがニーズに適していることが明らかである場合は、その手法を採用してください。ただし、計画と企業組織内の問題を認識したうえで、決定してください。

4

セキュリティのロックダウン

この戦略を採用する場合は、一部のコンピュータ上に MSJVM を継続して使用することができず、将来的に、セキュリティ リスクが発生する可能性があります。それらのリスクは、排除することはできませんが、低減することはできます。Internet Explorer (IE) には、MSJVM の適用範囲を無効化または制限するセキュリティ設定機能があります。

脆弱性を最小限に抑えるために、すべての Web サイト上の MSJVM を無効にすることもできます。IE で、[ツール] メニューの [インターネット オプション] を選択します。[セキュリティ] タブをクリックします。インターネット ゾーンの Java の設定を無効にするため、[レベルのカスタマイズ] を選択します。[MSJVM] セクションの [Java の許可] で、[Java を無効にする] を選択します。ここで設定した内容は、レジストリに格納されます。.reg ファイルとして保存して、.reg ファイルを読み込むログイン スクリプトを介して企業組織全体に反映することもできます。複数の人のアカウントが 1 台のマシン上にある場合は、それぞれのアカウントに対して別々の変更を行わなければなりません。Sun Microsystems 社の Java plug-In for Windows がインストールされているシステムでは、このオプションは、MSJVM を "エミュレート" する Sun JRE に対しても影響します。

MSJVM を継続して使用するお客様の選択肢として、Java を特定サイトに対して使用するよう制限することによりセキュリティ リスクを低減する方法が、次に考えられます。インターネット ゾーンで Java を無効に設定してから、すべての信頼済みサイトに対して Java を有効にします。次に、信頼済みサイトの一覧にアプレットを使用する特定サイトを追加して、アプレットのサポート機能を有効に戻します。

メモ IE のセキュリティ ゾーンの設定は、<APPLET> タグ処理のためにインストールされているすべての JRE に対して影響を及ぼします。

MSJVM の指定

ロックダウンするコンピュータに代替として JRE をインストールする予定がある場合は、別の事項を検討する必要があります。

システム上に複数の仮想マシン (VM) をインストールしているユーザーがいる場合には、移行に関して別の問題が発生します。インストールされているその他の JRE の機能によっては、既定の JRE がオブジェクトやアプレットを実行することがあります。オブジェクトを実行する JRE は、指定できません。JRE ベンダの中には、そのベンダの JRE が <OBJECT> タグを実行し、別の JRE が <APPLET> タグを実行するように指定できる構成アイテムを提供しているものがあります。

IE のセキュリティ ゾーンで設定した内容は、すべてのアプレットに適用されます。<APPLET> タグの代わりに <OBJECT> タグを使用するように、Web ファイルを編集することも可能です。<OBJECT> タグは、アプレットをはじめとする任意の埋め込みオブジェクトに対して、汎用性があります。

以下に示すように、2つのタグの構造は類似しています。

```
<APPLET code=Applet1.class
  height=200 width=320
  name=Applet1
  VIEWASTEXT>
  <PARAM NAME="foreground" VALUE="FFFFFF">
  <PARAM NAME="background" VALUE="008080">
  <PARAM NAME="label" VALUE="This string was passed from the HTML host.">
  No applet tag handlers installed.
</APPLET>
```

<OBJECT> タグでは、名前空間の競合を回避するために、いくつかの属性はパラメータになりません。また、<OBJECT> タグにさらに汎用性を持たせるために、追加のパラメータもあります。

```
<OBJECT CLASSID="clsid:08B0E5C0-4FCB-11CF-AAA5-00401C608501"
  height=200 width=320
  VIEWASTEXT>
  <PARAM NAME="code" VALUE="Applet1.class">
  <PARAM NAME="type" VALUE="application/x-java-applet">
  <PARAM NAME="scriptable" VALUE="true">
  <PARAM NAME="foreground" VALUE="FFFFFF">
  <PARAM NAME="background" VALUE="008080">
  <PARAM NAME="label" VALUE="This string was passed from the HTML host.">
  No MSJVM installed
</OBJECT>
```

classid 属性は、インライン データを指す URI です。Active X コントロール (JRE は、Active X コントロールと見なされます) の場合、URI は "clsid" で開始し、続けて、レジストリにある clsid 値を指定します。値 08B0E5C0-4FCB-11CF-AAA5-00401C608501 は、MSJVM を示します。サードパーティ製の JRE のバージョンによっては別の番号を使用することがありますが、それぞれ該当する番号が存在します。Java plug-in の中には、インストール時のレジストリ エントリとクラス ID を使用して MSJVM を "エミュレート" することで、Windows システムにインストールをプラグインするものがあります。このような、ベンダによる "エミュレーション" はマイクロソフトではサポートしていないため、マイクロソフト提供のユーティリティやアプリケーションが正しく機能しない原因になることがあります。最悪の場合、システムがセキュリティ リスクにさらされる危険があります。MSJVM を "エミュレート" する JRE のインストールには、細心の注意を払ってください。1 台のクライアントコンピュータで複数の JRE を使用すると、それらが競合することがあります。

ある状況下において特定の JRE を指定する設定をすると、別の問題が発生します。コードを記述する段階で、少なくとも 1 つの JRE により、MSJVM を指定するキーの下にサブキーがインストールされています。次に示すキーは、JRE の最新のバージョンで使用される CLSID の既定値です。そのため、MSJVM はこの値を読み込みません。

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{08B0E5C0-4FCB-11CF-AAA5-00401C608501}\TreatAs.
```

サブキーを削除すると、問題が解決されます。

5

.NET Framework への移行

J++ や Java アプリケーションを .NET Framework に移行するための自動化オプションには、J# Browser Control と Java Language Conversion Assistant の 2 種類があります。両オプションとも、既存の Java アプレットやアプリケーションを移行するためのウィザードとコマンドラインユーティリティを備えています。オプションには、それぞれの長所があり、またいずれのオプションでも .NET Framework への統合を行います。このようなことから、この章では、これら 2 つのオプションについて説明します。これらの 2 つのオプション ツールに関する詳細情報については、「付録 C」と「付録 D」を参照してください。ドキュメントの最新版は、Web サイトの MSDN ライブラリでご利用いただけます。

J# Browser Control は、アプレットを移行するためのものです。対応する Java クラスは限られていますが、処理機能は優れています。コードの 5% 程度は手動でのアップグレードが必要になるため、アプレットの再コンパイルが必要なことがあります。エンド ユーザーのコンピュータに読み込むため、.NET Framework ランタイム環境が必要です。

C# 対応の Java Language Conversion Assistant (JLCA) は、アプレットとアプリケーションを C# および .NET Framework に変換する機能を提供します。JLCA は、プロジェクトと個々のファイルを変換し、JLCA 特有の <OBJECT> タグを使用して HTML ページを更新します。Visual J++ コードの変換処理は、90% 以上が自動的に行われます。

1 つの移行プロジェクトにおいて、両方のオプション ツールを使用することも可能です。

J# Browser Control と .NET Framework への移行

Microsoft J# Browser Control を使用すると、Java アプレットを .NET Framework に移行できます。J# Browser Control にアップグレードされた Java アプレットは、エンド ユーザーのコンピュータの IE 上で、.NET Framework のコンテキストの範囲内で実行されます。Web サイトによってホストされる J# Browser Control を実行するエンド ユーザーは、コンピュータに .NET Framework と J# Browser Control のランタイムをインストールしておく必要があります。同様に、Java アプレットの J# Browser Control へのアップグレードを行う開発者は、開発用コンピュータに J# Browser Control ランタイムをインストールしておく必要があります。

J# Browser Control にアップグレードされた Java アプレットは、移行後も機能を失うことなく、同様のランタイム機能を実現します。多くの場合、エンド ユーザーが Web サイトを表示しているときに、Java アプレットが J# Browser Control にアップグレードされていることに気付くことはありません。

アプレットの移行

Java アプレットを J# Browser Control に移行するには、以下の 3 段階のプロセスで行います。

1. Visual J# .NET コンパイラを使用して、Java アプレットをコンパイルし、J# Browser Control に変換する。

2. アプレットの <APPLET> タグを含む HTML ページをアップグレードする。
3. Web サーバーに、J# Browser Control と HTML ページを配置する。

これらの各ステップについては、付録 D で説明します。

J# Browser Control を展開するには、マネージ ライブラリをコピーして、Web サーバー上の適切なディレクトリに対して HTML ページを更新します。マネージ ライブラリは、HTML ページと同じディレクトリまたはサブディレクトリにコピーする必要があります。

IIS Web サーバーを使用している場合は、J# Browser Control のコピー先である仮想ディレクトリのアクセス権を正しく設定してください。J# Browser Control で使用されているリソース ファイル (イメージ、オーディオ、データ ファイルなど) はすべて、マネージ ライブラリをコピーしたのと同じ場所にコピーする必要があります。J# Browser Control でリソース ファイルを使用する方法の詳細については、付録 D の「Upgrading Visual J++ 6.0 Applications That Use Resources」を参照してください。

J# Browser Control は、複数のマネージ ライブラリに分割することができます。分割する場合は、HTML ページが、J# Browser Control のメイン クラスを含むライブラリを参照している必要があります。また、J# Browser Control に関連するすべてのファイルは、同じディレクトリにコピーする必要があります。ファイルは、CAB、ZIP、JAR ファイルなどにパッケージ化せず、独立したファイルとしてコピーする必要があります。

JLCA を使用した C# と .NET Framework への移行

Java Language Conversion Assistant (JLCA) は、Visual J++ 6.0 プロジェクトと Java 言語 ファイルを Visual C# に変換するツールです。これらのファイルを Visual C# に変換することで、既存のコード ベースを使用し、.NET Framework のメリットを活かすことができます。JLCA の適用対象は、MSJVM を対象にした Java コードに限定されません。Java2 ライブラリ環境での JSP (Java Server Page) やサーブレットなどの、サーバー側の Java プログラムの変換にも対応しています。今後、JLCA にはより多くの変換機能が追加される予定です。その中には、Swing や J2EE アプリケーションを変換する機能も含まれます。変換後の Visual C# プロジェクトには、既存の Visual J++ または Java 言語コードから自動生成された Visual C# コードが含まれます。JLCA を使用して、Visual J++ プロジェクトまたは Java 言語アプリケーションと、アプレットのプロジェクトを変換することもできます。変換は、以下のように行われます。

表 5.1 変換結果

変換前	変換後
アプリケーション	Windows フォーム アプリケーション
アプレット	Web ユーザー コントロール

変換後の Web ユーザー コントロールは、アプレットと同様にブラウザでホストできます。ホスト対象となるコントロールは、<APPLET> タグではなく <OBJECT> タグを使用して、HTML ページに宣言されます。classid 属性はコントロールの指定に使用するもので、コントロールへのパスと完全な修飾子を付加したコントロール名を明示します。パスと名前はシャープ記号 (#) で区切ります。その例を次に示します。

```
<OBJECT id="myControl"
classid="http:ControlLibrary1.dll#ControlLibrary1.myControl"
VIEWASTEXT></OBJECT>
```

コントロールが適切に表示されるようにするには、コントロールを定義する .dll ファイルを、表示する Web ページと同じ仮想ディレクトリに配置するか、グローバル アセンブリ キャッシュに配置する必要があります。

サポート クラス

元のプロジェクトにおいて Microsoft Visual C#[®] では利用できない機能を変換するため、JLCA は、元の機能を複製したサポート クラス ("マネージャ" と呼ばれます) を生成します。サポート クラスは、それらがエミュレートするクラスとは、本質的に構造が異なる場合があります。変換対象となるプロジェクトにおけるアプリケーションの元の構造を維持するようにあらゆる手段を尽くしますが、これらのサポート クラスの本来の目的は、元の機能を複製することにあります。

変換レポート

プロジェクトには、自動変換の不可能なコードが存在していることがあります。Java Language Conversion Assistant ウィザードの実行後、変換プロセス中に発生したすべてのエラー、警告、問題を詳細に報告する変換レポートを参照できます。変換されなかったコードは、UPGRADE_TODO というラベルが付いたコメントとして、新プロジェクトのコードに記述されます。変換に関するコメント (変換コメント) は、タスク一覧で確認できます。それぞれの変換コメントには [ヘルプ] のトピックへのリンクが含まれており、そのコードを手動で変換する方法を確認できます。

6

DHTML へのリライト

ここでは、"DHTML (ダイナミック HTML)" という用語は、ドキュメント オブジェクト モデル (DOM) における CSS、スクリプト、HTML の組み合わせを含めた広範な意味で使用しています。DHTML は、注意深く使用すれば、柔軟性をもってオーバーヘッドを低く抑え、マルチブラウザおよびマルチプラットフォーム ソリューションとして活用できることにその長所があります。プラグインやダウンロードにかかるオーバーヘッドはありません。つまり、仮想マシン (VM) やプラグインではなく、ブラウザにおいて、ナビゲーション支援の解釈やレンダリングが行われます。多くの Web サイトでは、ドロップダウン メニュー、テキスト スクローラ、時計、ロールオーバー ボタンに加え、電卓、チャットクライアント、エディタなどのより複雑なユーティリティなどのインターフェイス機能を実装するために、パブリックドメインや開発コストの安いコードを提供しています。

DHTML の短所として、ブラウザ間の互換性を確実にするために注意を払う必要があるため、適切なプラットフォームとブラウザを使用したテストが必要です。

メモ マイクロソフトとしては他社のソリューションについてセキュリティと信頼性を保証することはできませんが、マイクロソフト以外のソリューションを導入するという方法も可能です。サードパーティのソリューションを使用する場合は、必ずテストを行ってから移行に着手してください。

ブラウザのバージョンに関する問題

DHTML は、スクリプト言語やスタイル シートの実装に大きく依存するため、各ブラウザおよびプラットフォーム上でどの程度適切に動作するか、という問題があります。バージョン 6.0 のブラウザは、スタイル シートを適度に扱うことができ、ほとんどのブラウザが信頼を寄せることのできるスクリプト言語 (ECMAScript および拡張機能) による機能のコア セットを備えています。

アプレットの DHTML バージョンに対してどの程度広範に互換性を持たせるかは、ビジネス ニーズによって異なります。Web 上で使用できる小規模の DHTML スクリプトでは、ほとんどの場合は互換性があります。DHTML によるソリューションを選択する場合は、必要なブラウザに対応しているかどうかを十分にテストすることが必要になります。

実例とサンプル

このガイドの目的は DHTML を説明することではありませんが、以下に、DHTML 実装によるさまざまな効果を示すサンプルへのリンクの一覧を示します。ニーズに合ったサンプルを見つけたら、それを開始点として作業に着手してください。

IE における DHTML については、<http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/jpdnie60/htm/whatsnewpublicpreview.asp> を参照してください。

ロールオーバー効果のサンプルについては、

<http://msdn.microsoft.com/downloads/samples/internet/default.asp?url=/downloads/samples/internet/author/html/rollover/default.asp> (英語情報) を参照してください。

時計やチャット クライアントなど、さまざまな実用的サンプルについては、<http://msdn.microsoft.com/downloads/samples/internet/default.asp> (英語情報) を参照してください。

マイクロソフト以外の Web サイトにも、たくさんの役に立つ情報が提供されています。<http://www.codetoad.com/dhtml/>、<http://simplythebest.net>、<http://www.dynamicdrive.com/dynamicindex1> などのサイト (英語情報) では、短時間で多数のサンプルを検索できます。

マイクロソフトは、これらのサンプルが正しくまたは期待どおりに動作することを保証することはできません。十分なテストを行い、ニーズに見合うことを確認して、それぞれの責任の下で使用してください。

7

代替 JRE への切り替え

ビジネス ニーズに最適なソリューションが JRE (Java Runtime Environment) であるときは、代替 JRE への切り替えが必要になります。

代替 JRE

数社のコンピュータ ベンダが、各社のコンピュータ上で動作する代替 JRE (Java Runtime Environment) を出荷すると発表しています。このソリューションを採用するかどうかは、ビジネス ニーズを基準に決定してください。代替 JRE に移行する場合は、移植性、信頼性、バージョン、保守などの検討事項があります。

Windows 上で使用できる代替 JRE は、多数あります。各ベンダは、各社の JRE が関連する Java 仕様に準拠していると表明していますが、すべての JRE が必ずしもユーザーのニーズを満たすとは限りません。また、Java 仕様の実装がすべて同じであるわけではなく、これらの相違が問題の原因になる可能性をはらんでいます。

現在の Visual J++ コードを移植して、代替 JRE で動作することを確認する必要があると考えられます。

マイクロソフトが提供する拡張機能を使用しているかどうかは、現在の Visual J++ ソース ファイルを検索することで容易に見つけることができます。以下のフォーマットで CMD.EXE コマンドを実行して、@dll 命令が使用されているかどうかを Java ソース ファイルで検索します。

```
find /I "@dll" c:\src\*.*
```

同じコマンドを使用して、@com 命令や @security 命令が使用されているかどうかなど、マイクロソフトが提供する他の拡張機能も検索できます。拡張機能の使用を確認したら、それらを除外することによる効果狙います。

命令文はコメントに埋め込まれるため、マイクロソフトが提供する拡張機能を使用または削除しても、Java コードに構文上の相違は生じません。javac コマンドにオプションを指定して拡張機能を無効化できるため、この機能を付加した場合と付加しない場合のアプリケーション実行効果の比較は容易に行うことができます。

サポートおよび保守面のニーズを確認してください。Java に関する問題は、マイクロソフトではサポートしません。JRE の提供者にお問い合わせください。

Sun Microsystems 社の Windows 版 JRE は、<http://java.sun.com/getjava/ja/index.html> で入手できます。IBM 社の Windows 版 JRE は (IBM Websphere SDK for Web Services の一環として) <http://www-106.ibm.com/developerworks/webservices/wsdsk/> (英語情報) で入手できます。その他にも、利用可能なパッケージがあります。

マイクロソフトは、これらの JRE が正しくまたは期待どおりに動作することを保証できません。十分なテストを行い、ニーズに見合うことを確認して、それぞれの責任の下でご使用ください。

付録

A

MSJVM 開発者向け FAQ

2001 年 1 月における Sun Microsystems 社との合意に基づき、マイクロソフトは、マイクロソフト製品からの Microsoft 仮想マシン (MSJVM) の段階的廃止に向けて作業を進めています。マイクロソフトは、MSJVM に対して依存関係のある開発作業は中止し、MSJVM に依存する既存コードの移行オプションについて検討することをお勧めします。

次の FAQ では、開発者の皆様が抱くと思われる MSJVM、および、Java 言語ベースの Microsoft Visual J++ 開発システムに関する質問事項を取り上げています。

よく寄せられる質問

質問 : MSJVM と Visual J++ を打ち切りにする理由を教えてください。

回答 : この決定は、2001 年 1 月、Sun Microsystems 社との法的な係争が決着したことによるものです。マイクロソフトにおいては、2004 年 9 月 30 日をもって Microsoft 仮想マシン (MSJVM) のサポートが認められなくなります。マイクロソフトは、セキュリティ上の重大な脆弱性や、お客様にご利用いただくうえで不具合点も含め、MSJVM のアップデートを行うことができなくなります。このことを理由として、マイクロソフトは、今後開発される環境において、お客様が MSJVM の使用を継続することをお勧めできません。マイクロソフトは、多くのお客様が、Java に対応したマイクロソフトテクノロジーを使用してアプリケーションや Web サイトを構築されていることを認識しています。このようなお客様に代替ソリューションに移行していただけるように、マイクロソフトは情報を提供していきたいと考えています。このガイドの目的もその点を満たすことにあります。

質問 : MSJVM および Visual J++ のユーザーが使用できる移行オプションにはどんなものがありますか。

回答 : 次のような、いくつかのオプションがあります。

- Visual J# .NET または JLCA (Java Language Conversion Assistant) を使用して、Java コードを .NET Framework に移行する - 開発者の皆様に支援するため、マイクロソフトは、これらの製品の機能を強化する予定です。このオプションでは、マイクロソフトはユーザーの皆様に継続したサポート サービスを提供します。
- 現在、Java コードが提供している機能を、他のテクノロジー (HTML (Microsoft JScript を使用)、Microsoft ActiveX コントロール、ASP.NET、Macromedia Shockwave、Macromedia Flash など) で置き換えます。
- MSJVM を、他社提供の JRE (Java Runtime Environment) で置き換えます。

質問 : マイクロソフトは、MSJVM のサポートを継続する予定ですか。

回答 : 2004 年 9 月 30 日まではサポートを行いません。2004 年 9 月 30 日以降、マイクロソフトは、セキュリティ上の脆弱性への取り組みや製品の機能強化を含め、MSJVM になんら修正を加えることはできなくなるため、MSJVM をサポート対象と位置付けることはできません。

質問 : マイクロソフトは、Visual J++ のサポートを継続する予定ですか。

回答：はい、継続します。ただし、サポート オプションは限定されます。詳細については、『Visual J++ 6.0 サポート ガイダンス』の Visual J# サイトに関するドキュメントを参照してください。

質問：マイクロソフトは、MSJVM の開発または機能強化を継続する予定ですか。

回答：いいえ。Sun Microsystems 社との合意に基づき、マイクロソフトは MSJVM の機能強化を行うことはできません。

質問：当面の間、MSJVM を入手するにはどうすればよいのでしょうか。

回答：マイクロソフトは、Windows Update を介して、既にインストールされている MSJVM に対するアップデートを提供します。これらのアップデートを入手するには、コンピュータ上に既に MSJVM がインストールされていることが必要です。今後リリースされる Windows およびその他のマイクロソフト新製品には、MSJVM が搭載されなくなります。また、Visual J++ の頒布も行いません。

Java アプリケーションの構築に マイクロソフトの開発者ツールを使用している場合、それらのツールの使用許諾契約に基づいて、MSJVM を再頒布することができます。使用許諾契約 (EULA) を参照して、再頒布権の行使範囲を確認してください。

質問：MSJVM の頒布を続けることはできますか。

回答：マイクロソフトによるサポートが制限されるため、MSJVM の頒布を続けることはお勧めできません。Visual J++ および Microsoft SDK for Java の使用許諾契約 (EULA) には、MSJVM のインストーラ (msjavx86.exe) の再頒布権を制限する条項があります。詳細については、「ライセンスの許諾: インストールおよび使用」のページを確認してください。

質問：Visual J# .NET をインストールすると、Visual J++ 6.0 プロジェクト全体がシームレスにアップグレードされるのですか。

回答：Visual J# .NET には、Visual J++ 6.0 プロジェクト ファイルを Visual J# .NET にアップグレードするためのアップグレード ウィザードがあります。Visual J# .NET に同等の機能がある場合は、Visual J++ 6.0 のソース コードはシームレスにアップグレードされます。Visual J# .NET でサポートされないクラス ライブラリまたは機能がある場合は、その問題に関するヘルプが表示されません。アップグレードが成功するかどうかは個々のプロジェクトによりますが、約 95% のコードを問題なくシームレスにアップグレードできたという実績があります。

- 詳細については、MSDN のドキュメント「Moving Java Applications to .NET (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/dotnet_movingjavaapps.asp)」(英語情報) をご覧ください。

質問：JLCA をインストールすると、Visual J++ 6.0 プロジェクト全体がシームレスにアップグレードされるのですか。

回答：JLCA は既存の Visual J++ 6.0 アプリケーションを .NET Framework 上で C# に変換するツールです。Visual J++ の主要なコンポーネントをすべて変換でき、変換できなかった部分や手動での修正が必要な部分を報告するレポートも生成されます。アップグレードが成功するかどうかは個々のプロジェクトによりますが、80 ~ 90% のコードを問題なくシームレスにアップグレードできたという実績があります。ただし、マイクロソフトが、すべてのケースにおいて同様にアップグレードできることを保証するものではありません。

- 詳細については、MSDN のドキュメント「Moving Java Applications to .NET (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/dotnet_movingjavaapps.asp)」(英語情報) をご覧ください。

質問：Sun Microsystems 社の Java Runtime Environment (Sun JRE) をインストールした場合でも、MSJVM を使用し続けることはできますか。

回答：インストールすると、Sun JRE が、Java アプレットや Java アプリケーションに対する Windows における既定の実行環境になります。ただし、MSJVM がインストールされているコンピュータでは、Sun JRE を使用するように設定されているアプリケーションごとに、コントロール パ

ネルを介して、使用する JRE を選択できます。

質問 : Windows XP Service Pack 1a (SP1a) をインストールした場合、現在使用している MSJVM は削除されるのですか。

回答 : いいえ。コンピュータに既存の MSJVM は削除されません。

質問 : Sun JRE をインストールしても、MSJVM を使用する .exe ファイルを実行できますか。

回答 : はい。そのようなシステム構成下では、Visual J++ または Microsoft SDK for Java を使って構築されたアプリケーション (.exe ファイル) は、継続して MSJVM を使用することができます。

B

Diagnostic Tool for the Microsoft VM

Diagnostic Tool for the Microsoft VM の入手

現在、マイクロソフトは、リリースの詳細を最終的に決定し、ツールを提供できるように、作業を進めています。リリース日の最新情報およびダウンロード先へのリンクについては、MSJVM Web サイト (<http://www.microsoft.com/japan/java/default.asp>) をご覧ください。

Diagnostic Tool for the Microsoft VM のインストール

ウィザードは、Windows インストーラ パッケージとして出荷されます。インストールは、以下の手順に従って実行してください。

重要 製品の新しいバージョンをインストールする場合は、あらかじめ、以前のバージョンをアンインストールする必要があります。

1. Microsoft Diagnostic Tool for the Microsoft VM に同梱されている Windows インストーラ パッケージ (.msi) ファイルまたは Setup.exe ファイルをダブルクリックします。
2. [よろこそ] 画面が表示され、一般情報および著作権情報を示します。[次へ] をクリックします。
3. 使用許諾契約 (EULA) を確認します。条項に同意する場合は、[同意する] をクリックし、続いて [次へ] をクリックします。
4. 既定のフォルダにインストールする場合は、[次へ] をクリックします。[参照] をクリックして、別のインストール場所を指定することもできます。
5. 診断ツールを、ローカル コンピュータ上のすべてのユーザーが使用できるようにインストールするには、[Everyone] をオンにします。現在のユーザーだけが使用できるようにインストールするには、[Just me] をオンにします。
6. [次へ] をクリックして、インストール プログラムを起動します。[戻る] をクリックして、インストールの設定内容を確認することもできます。
7. リリース情報を確認します。

メモ この情報はインストールの後、ツールの ReadMe ファイルで調べることができます。

C

Java Language Conversion Assistant

Java Language Conversion Assistant は、Visual J++ 6.0 プロジェクトと Java 言語ファイルを Visual C# に変換するツールです。これらのファイルを Visual C# に変換することで、既存のコードベースを活用し、.NET Framework のメリットを活かすことができます。

変換後の Visual C# プロジェクトには、既存の Visual J++ または Java 言語コードから自動生成された Visual C# コードが含まれます。詳細については、「Visual J++ または Java 言語から Visual C# へのプロジェクトの変換 (http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/dv_jlc_a/html/vbtskusingjavavalanguageconversionassistant.asp)」を参照してください。

JLCA (Java Language Conversion Assistant) を使用して、Visual J++ プロジェクト、または、Java 言語アプリケーションとアプレットのプロジェクトを変換することもできます。変換は、以下のように行われます。

変換前	変換後
アプリケーション	Windows フォーム アプリケーション
アプレット	Web ユーザー コントロール

変換後の Web ユーザー コントロールは、ブラウザで、アプレットと同様にホストすることができます。ホスト対象となるコントロールは、(<APPLET> タグではなく) <OBJECT> タグを使って、HTML ページに宣言されます。classid 属性はコントロールの指定に使用するもので、コントロールへのパスと完全な修飾子を付加したコントロール名を明示します。パスと名前はシャープ記号 (#) で区切ります。次に例を示します。

```
<OBJECT id="myControl"
classid="http:ControlLibrary1.dll#ControlLibrary1.myControl"
VIEWASTEXT></OBJECT>
```

コントロールが適切に表示されるようにするには、コントロールを定義する .dll ファイルを、表示する Web ページと同じ仮想ディレクトリに配置するか、グローバル アセンブリ キャッシュに配置する必要があります。

サポート クラス

元のプロジェクトにあって Microsoft Visual C#[®] では利用できない機能を変換するため、JLCA は、元の機能を複製したサポート クラス ("マネージャ" とも呼ばれます) を生成します。サポート クラスは、それらがエミュレートするクラスとは、本質的に構造が異なる場合があります。変換対象となるプロジェクトにおけるアプリケーションの元の構造を維持するようあらゆる手段は尽くされますが、これらのサポートクラスの本来の目的は、元の機能を複製することにあります。

変換レポート

プロジェクトには、自動変換の不可能なコードが存在していることがあります。Java Language Conversion Assistant ウィザードの実行後、変換プロセス中に発生したすべてのエラー、警告、問題を詳細に報告する変換レポートを参照できます。変換されなかったコードは、UPGRADE_TODO というラベルが付いたコメントとして、新プロジェクトのコードに記述されます。変換に関するコメント (変換コメント) は、タスク一覧で確認できます。それぞれの変換コメントには [ヘルプ] のトピックへのリンクが含まれており、そのコードを手動で変換する方法を確認できます。詳細については、「未変換コードの手動アップグレード (http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/dv_jlca/html/vbtskmanuallyupgradingnonconvertedcode.asp)」を参照してください。

Visual J++ または Java 言語プロジェクトを Visual C# に変換する

将来の開発に備え、JLCA (Java Language Conversion Assistant) を使用して、Visual J++ 6.0 プロジェクトや Java 言語ファイルを Visual C# に変換することができます。JLCA は、既存プロジェクトの変更は行わず、元のプロジェクトをベースにした新たな Visual C# プロジェクトを作成します。変換プロセス中に発生したエラー、警告、問題はすべて、新プロジェクト生成後に報告される変換レポートに表示されます。これらのエラー、警告、問題は、変換後のコードにもコメントとして表記されます。これは、プロジェクト内で自動変換できなかったコード部分を手動で変換するときに役立ちます。

メモ Visual C# の初級ユーザーの方は、変換プロセスを実行する前に、Visual C# を試用する時間を設けてください。詳細については、「C# 言語ツアー (http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/csccon/html/vclrfgettingstarted_pg.asp)」を参照してください。

Visual J++ プロジェクト

Visual J++ プロジェクトの変換プロセス手順は、以下のとおりです。

1. Visual Studio .NET を起動します。
2. [ファイル] メニューの [開く] をポイントし、[変換] をクリックします。
3. [Java Language Conversion Assistant] を選択して [OK] をクリックします。
4. [ソースファイル] ページで、[Visual J++ 6.0 プロジェクト] を選択します。
5. [プロジェクトファイルの選択] ページで、[参照] をクリックします。
6. 適切な .vjp ファイルを探して選択します。

メモ .vjp ファイルに .jar または .class ファイルを指す CLASSPATH 命令がある場合、それらの命令は無視されます。

7. [新しいプロジェクトを作成する場所] ページで、新プロジェクトを作成するディレクトリとプロジェクト名を指定します。
8. [変換の開始] ページで、[次へ] をクリックします。
9. 自動的に変換されなかったコードを修正します。詳細については、「未変換コードの手動アップグレード (http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/dv_jlca/html/vbtskmanuallyupgradingnonconvertedcode.asp)」を参照してください。

Java 言語プロジェクト

Java 言語プロジェクトの変換プロセス手順は、以下のとおりです。

1. Visual Studio .NET を起動します。

2. [ファイル]メニューの [開く] をポイントし、[変換] をクリックします。
3. [Java Language Conversion Assistant] を選択して [OK] をクリックします。
4. [ソースファイル] ページで、[プロジェクトファイルを含むディレクトリ] を選択します。
5. [ソースディレクトリの選択] ページで、[参照] をクリックします。
6. 適切なプロジェクトを探して選択します。

メモ 選択したディレクトリにあるファイルの一覧が表示されますが、変換対象となるのは .jav および .java ファイルです。ディレクトリ内のその他のファイルは、すべて無視されます。

[新しいプロジェクトの構成] ページで、以下の内容を指定します。

作成するプロジェクトの名前

作成するプロジェクトに関し、追加ファイルを配置するディレクトリ

メモ 選択したディレクトリにあるファイルの一覧が表示されますが、変換対象となるのは .jav および .java ファイルです。ディレクトリ内のその他のファイルは、すべて無視されます。

正しいプロジェクトの種類

[新しいプロジェクトを作成する場所] ページで、新プロジェクトを作成するディレクトリとプロジェクト名を指定します。

[変換の開始] ページで、[次へ] をクリックします。

自動的に変換されなかったコードを修正します。詳細については、「未変換コードの手動アップグレード」(http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/dv_jlca/html/vbtskmanuallyupgradingnonconvertedcode.asp) を参照してください。

変換されなかったコードを手動でアップグレードする

JLCA (Java Language Conversion Assistant) を使って Visual J++ プロジェクトや Java 言語ファイルを変換して作成される新たな Visual C# プロジェクトには、自動変換できなかったコードが含まれている可能性があります。新プロジェクトの該当するコードにはコメントが挿入されているため、Visual C# にコードを手動で変換する作業に役立ちます。

コメントの種類	説明
UPGRADE_TODO	自動変換できなかったコード
UPGRADE_WARNING	問題があると考えられるコード
UPGRADE_NOTE	元のコードとは異なる動作を実行すると考えられるコード

アプリケーションをコンパイルする前に修正が必要となる、コンパイラ エラーが含まれていることがあります。各コメントには、問題について簡単な説明が表示されています。また、コードの変換方法に関する [ヘルプ] トピックへのリンクも表記されています。

変換されなかったコードを手動でアップグレードするには

1. Java Language Conversion Assistant ウィザードの実行後、Visual Studio .NET でプロジェクトを開きます。
2. [表示]メニューの [タスクの表示] をポイントし、[コメント] または [すべて] を選択します。
[タスクの一覧] に、変換コメントが表示されます。
3. [タスクの一覧] で、最初の変換コメントをダブルクリックします。
画面表示が、コード内の該当コメントに移動します。
4. 変換コメントを確認し、関連するコードを調べます。エラーを手動で修正する手順の詳細については、変換コメントに示されているリンクをクリックして、該当する [ヘルプ] トピックを参照してください。

JSP ページの変換コードでの変換コメントには、リンク先が表示されていないことがあります。このようなときは、ヘルプの [検索] ボックスに、変換コメントの内容をコピーしてください。エラー、警告、問題の一覧については、「Java Language Errors, Warnings, and Issues (http://msdn.microsoft.com/library/en-us/dv_jlca/html/vberrjavalanguageerrorwarningsissues.asp)」(英語情報)を参照してください。自動変換が不可能な JSP のクラスやメソッドに関する [ヘルプ] トピックの一覧については、「Javax.servlet.jsp Error Messages (http://msdn.microsoft.com/library/en-us/dv_jlca/html/vberrjavaservletjsperrormessages.asp)」(英語情報)を参照してください。

5. コードを手動でアップグレードします。
6. ステップ 3 ~ 5 を繰り返して、すべての変換コメントに対処します。

JLCA を使ってアプレットを変換する

JLCA (Java Language Conversion Assistant) を使用してアプレットを変換するには、2 段階のプロセスが必要です。まず、実際のアプレットのコードを、JLCA を使って C# に変換します。次に、アプレットを参照する HTML ページを、.NET Framework の同等の機能に変換します。コードを C# に変換する手順は、前に述べたアプリケーション変換とまったく同じ手順で行います。アプレットコードは、クラス ライブラリ プロジェクトとして変換されます。

変換後のアプレットに存在する問題を解決すると、Windows フォーム コントロールが完成します。変換後の Web ユーザー コントロールは、Internet Explorer で、アプレットと同様にホストすることができます。

適切に表示されるように制御するには、コントロールを定義する .dll ファイルを、表示する Web ページと同じ仮想ディレクトリに配置するか、グローバル アセンブリ キャッシュに配置することが必要です。

HTML ページをアップグレードする

Windows フォーム コントロールは、Internet Explorer で Java アプレットを実行するために Microsoft 仮想マシン (MSJVM) やその他の仮想マシン (VM) で使用される、非推奨の <APPLET> タグや <OBJECT> タグ 文法をサポートしません。変換後のアプレットをブラウザで実行するためには、WFC の <OBJECT> タグ 文法を使用して Web ページを変更する必要があります。この操作は、自動または手動で行うことができます。

Web ページ参照をアップグレードする最も簡単な方法は、JLCA を使用することです。HTML プロジェクトに Visual J++ アプレットがあり、その Visual J++ ファイルを使って変換するように選択すると、JLCA により、自動的に、関連する HTML ファイルが解析され、(必要に応じて) すべての <applet> タグ参照が変更されます。このタグ テキストをコピーして、同じアプレットを参照する任意のページに貼り付けることができます。

アプレットを持つ Visual J++ プロジェクトがない場合、または、Visual J++ を使用したくない場合は、手動で HTML 参照を変換できます。今日の HTML ページで使用されている <APPLET> タグ 文法は、以下ようになります (MSJVM の拡張機能を含みます)。

```
<APPLET
CODE = appletFile
CODEBASE = codebaseURL
WIDTH = pixels
HEIGHT = pixels
VSPACE = pixels
HSPACE = pixels
ALIGN = alignment
ID = appletInstanceName
ARCHIVE = archiveList
ALT = alternateText
>
```

```

<PARAM NAME = FireScriptEvents VALUE = True>
<PARAM NAME = cabbase VALUE = cabFileName>
<PARAM NAME = cabinets VALUE = cabFileNames>
<PARAM NAME = useslibrary VALUE = DUFriendlyName>
<PARAM NAME = useslibrarycodebase VALUE = DUFileName>
<PARAM NAME = useslibraryversion VALUE= DUVersionNumber>
<PARAM NAME = namespace VALUE = applicationNamespace>
<PARAM NAME = appletAttribute1 VALUE = value>
<PARAM NAME = appletAttribute2 VALUE = value>
. . .
alternateHTML
</APPLET>

```

Windows フォーム コントロールでは、次のような文法がサポートされています。

```

<OBJECT
CLASSID="http:<OutputFileName>#<ClassName>"
WIDTH= pixels
HEIGHT= pixels
ID=browserControlInstanceName
ALIGN= alignment
HSPACE= pixels
VSPACE= pixels
>
<PARAM NAME = attribute1 VALUE = value>
<PARAM NAME = attribute2 VALUE = value>
. . .
alternateHTML
</OBJECT>

```

各要素の意味は次のとおりです。

"CLASSID" は CLASSID 文字列で、出力ファイル名と Windows フォーム コントロールのクラス自身の名前で構成されています。これらの設定内容は、[プロジェクトのプロパティ] ダイアログボックスで確認できます。[プロジェクトのプロパティ] ダイアログ ボックスを開くには、ソリューションエクスプローラで変換後のアプレット プロジェクトを右クリックし、表示されるメニューで [プロパティ] を選択します。

要約

アプリケーションを .NET Framework に移行する作業は、比較的直接的なプロセスを踏むもので、変換には JLCA (Java Language Conversion Assistant) を使用します。アプレットの移行作業には、アプレットを参照する HTML ページへの追加の変更作業が必要になりますが、単純な作業で済みます。

D

Java アプレットの Microsoft J# Browser Control への移行

ここに記載する内容は、J# Browser Control パッケージに関するホワイトペーパーを転載したものです。このホワイトペーパーの最新版は、http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/vjtskMigratingJavaAppletsToMicrosoftJBrowserControls.asp (英語情報) から入手できます。

注意 この章に登場する J# Browser Control に関する記載は、開発中の英語版の製品に基づいています。今後、日本語版が開発される予定です。

概要 Microsoft J# Browser Control を使用すると、開発者は、Java 仮想マシン (VM) 上で実行するように記述された Java アプレットを、.NET Framework に移行することができます。J# Browser Control は、現在、ベータ版がリリースされています。最終版は、本年後半にリリースされる予定です。

この付録は、開発者の方を対象として、J# Browser Control を紹介し、既存の Java アプレットを J# Browser Control に移行する手順について説明します。また、J# Browser Control に移行する Java アプレットについてのセキュリティ セマンティクスや、ベータ版ではサポートされていない機能についても説明します。

このホワイトペーパーは、開発者の方が .NET Framework および Visual J# .NET に精通していることを前提に記載しています。詳細については、それぞれ、<http://msdn.microsoft.com/netframework> (英語情報) と http://msdn.microsoft.com/library/en-us/dv_vjsharp/html/vjoriMicrosoftVisualJ.asp (英語情報) を参照してください。J# Browser Control をダウンロードする方法については、<http://www.microsoft.com/japan/msdn/vjsharp/downloads/browsercontrols/> (英語情報) を参照してください。

はじめに

Visual J# .NET を使って .NET Framework に移行した Java アプレットを、J# Browser Control といいます。J# Browser Control は、.NET Framework を使って、Internet Explorer 上で動作します。

J# Browser Control は、移行前の Java アプレットと同様のランタイム機能を保有し、また、Java 言語セマンティクスを維持しています。多くの場合、エンドユーザーが Web サイトを表示しているときに、Java アプレットが J# Browser Control にアップグレードされていることに気付くことはありません。

Java アプレットを J# Browser Control に移行するには、J# Browser Control ランタイム バージョンと、.NET Framework SDK または Visual Studio .NET 2003 をインストールする必要があります。同様に、Web サイトによってホストされる J# Browser Control を実行するエンド ユーザーは、コンピュータに .NET Framework と J# Browser Control ランタイム バージョンをインストールしておく必要があります。J# Browser Control は、Java 仮想マシン (VM) 上での実行は意図されていません。J# Browser Control は、Internet Explorer 上でのみ実行可能です。

Java アプレットを J# Browser Control に移行するには、以下のような 3 段階のプロセスが必要です。

1. Visual J# コンパイラを使用して、Java アプレットをコンパイルして、J# Browser Control に変換する。
2. ツールを使って、(<APPLET> タグを含む HTML ページが) <OBJECT> タグを使用するように、HTML ページをアップグレードする。
3. Web サーバー上に、J# Browser Control と HTML ページを展開する。

Java アプレットをコンパイルして J# Browser Control を作成する

Java アプレットは、"Visual J# コンパイラ" (vjc.exe) (http://msdn.microsoft.com/library/en-us/dv_vjsharp/html/vjgrfVisualJCompilerOptions.asp) (英語情報) を使ってコンパイルすることで、J# Browser Control にアップグレードできます。これにより、マネージ ライブラリに対応できます。Java アプレットをマネージ ライブラリ向けにコンパイルする作業は、Visual J# でライブラリをコンパイルする作業と同様です。Java アプレットにバイトコード (.class ファイル) がある場合は、"Visual J# バイナリ変換ツール" (JbImp.exe) (http://msdn.microsoft.com/library/en-us/dv_vjsharp/html/vjgrfJavaBinaryConverter.asp) (英語情報) を使います。このツールは、Java アプレットのマネージ ライブラリへの変換にも利用できます。

Java アプレットを J# Browser Control にコンパイルするとき、Java アプレットのソース コードを変更する必要はありません。これは、J# Browser Control ランタイム バージョンが、(java.applet パッケージも含めた) JDK 1.1.4 レベル パッケージとほぼ同等の機能をサポートするためです。

J# Browser Control を含むマネージ ライブラリと .NET Framework におけるマネージ ライブラリに、相違点はありません。ユーザーが、J# Browser Control をホストする Web サイトを訪れると、J# Browser Control ランタイム バージョンによりマネージ ライブラリがダウンロードされ、Internet Explorer で java.applet.Applet を展開するクラスが実行されます。

Java アプレットから J# Browser Control へのコンパイルは、コマンド プロンプトまたは Visual Studio .NET を使用して行います。

コマンド プロンプトから Java アプレットをコンパイルする

コマンドラインを使って、Java アプレットをコンパイルできます。その例を次に示します。

```
C:\AppletSources>vjc.exe /target:library /out:MyApplet.dll *.java
```

Visual J# バイナリ変換ツールを使用する方法もあります。

```
C:\AppletSources>jbimp.exe /target:library /out:MyApplet.dll *.class
```

Visual Studio .NET をインストールしている環境では、Visual Studio .NET のコマンド プロンプトから、Visual J# コンパイラ (vjc.exe) や Visual J# バイナリ変換ツール (JbImp.exe) にアクセスできます。Visual Studio .NET をインストールしていない環境では、コマンド プロンプトに、これらのツールへのパスを入力する必要があります。

リソースを使用する Java アプレットを移行する場合は、「Upgrading Visual J++ 6.0 Applications That Use Resources」(<http://msdn.microsoft.com/library/>

en-us/dv_vjsharp/html/vjgrfUpgradingVisualJ60ApplicationsThatUseResources.asp)」(英語情報)の説明に従ってください。

Visual Studio .NET を使って Java アプレットを移行する

Java アプレットが HTML プロジェクトの Visual J++ 6.0 アプレットであるときは、Visual Studio .NET を使用して、J# Browser Control にコンパイルすることができます。

Visual Studio .NET を使用して、HTML プロジェクトの Visual J++ 6.0 アプレットをアップグレードするには

1. Visual Studio .NET で、Visual J++ 6.0 プロジェクトを開きます。Visual J# .NET プロジェクトアップグレードウィザードが起動します。
2. [次へ] をクリックして、アップグレードウィザードのすべてのステップを実行します。アップグレードウィザードにより、Visual J++ プロジェクトが Visual J# .NET クラスライブラリプロジェクトに変換されます。
3. アップグレードの処理中、何らかの問題が検出された場合には、ウィザードにより、アップグレードレポートが提示されます。

メモ このレポートで、「アプレットプロジェクトはサポートされていません」と報告されることがありますが、このエラーは無視してもかまいません。アップグレードレポートに一覧表示されている、これ以外の問題に対処してから、プロジェクトの構築に進んでください。

1. プロジェクトを構築します。この段階で、Java アプレットがマネージライブラリにコンパイルされます。

[デバッグ]メニューの[開始]をクリックする操作や、F5 キーを押す操作では、Visual Studio を使って、Internet Explorer で J# Browser Control を起動することはできません。J# Browser Control を起動するには、Web サーバーの仮想ディレクトリに、J# Browser Control をコピーする必要があります。詳細については、「Deploying J# Browser Control (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/vjtskMigratingJavaAppletsToMicrosoftJBrowserControls.asp)」(英語情報)および「How to: Debug J# Browser Control (http://msdn.microsoft.com/library/en-us/dv_vstechart/html/vjtskhowtodebugjbrowsercontrols.asp)」(英語情報)を参照してください。

Visual J++ 6.0 プロジェクトを Visual J# .NET にアップグレードする方法の詳細については、「Upgrading from Visual J++ 6.0 (http://msdn.microsoft.com/library/en-us/dv_vjsharp/html/vjsamUpgradingFromVisualJ60.asp)」(英語情報)を参照してください。

HTML ページの <APPLET> タグを <OBJECT> タグに変更する

Java アプレットをマネージライブラリ向けにコンパイルしたら、次のステップとして、Java アプレットをホストする HTML ページをアップグレードします。HTML ページに記述されている <APPLET> タグや Java アプレットの <OBJECT> タグは、J# Browser Control がサポートする <OBJECT> 構文に変換する必要があります。

J# Browser Control ランタイムバージョンには、HTML Applet to Object Tag Converter (TagConvert.exe) というツールが実装されています。このツールは、Java アプレットをホストする HTML ページの自動アップグレードを実行します。TagConvert.exe は、J# Browser Control のインストールディレクトリに格納されています。ツールは、以下のように使用します。

```
TagConvert [options] <source files>
```

その例を次に示します。

```
TagConvert.exe MyAppletPage.html
```

ツールへの入力に、テキストファイルを使用できます。.html、.htm、.asp、.aspx の拡張子を持つ

ファイルも使用可能です。

このツールを使用すると、<APPLET> タグや Java アプレットの <OBJECT> タグが、次のような J# Browser Control の <OBJECT> タグに変換されます。

```
<OBJECT
CLASSID="clsid:a399591c-0fd0-41f8-9d25-bd76f632415f"
WIDTH= pixels
HEIGHT= pixels
ID=browserControlName
ALIGN= alignment
HSPACE= pixels
VSPACE= pixels
VJSCODEBASE = codebaseURL
>
<PARAM NAME = attribute1 VALUE = value>
<PARAM NAME = attribute2 VALUE = value>
. . .
alternateHTML
</OBJECT>
```

パラメータは次のとおりです。

CLASSID

ActiveX コントロールの CLASSID。J# Browser Control をダウンロードし、実行します。この CLASSID は、J# Browser Control の <OBJECT> タグに指定されることが必要です。

VJSCODEBASE

J# Browser Control クラスの URL、および、このクラスを含むマネージ ライブラリ。"#" トークンは、マネージ ライブラリ ファイル名と、J# Browser Control のクラス名の区切り文字として使用されません。ファイル名には、拡張子も含める必要があります。その例を次に示します。

```
VJSCODEBASE =
http://www.MyWebSite.com/MyApplet/MyAppletClass.dll#MyAppletClass
```

変換プロセス

HTML ページの変換プロセスにおいて、ツールにより、元の <APPLET> タグや Java アプレットの <OBJECT> タグが、J# Browser Control の <OBJECT> タグに置換されます。タグの変換処理の前に、オリジナル ファイルのバックアップが作成されます。ファイルのバックアップ コピーには、拡張子 .vjsbak が付加されます。たとえば、index.htm は、index.htm.vjsbak としてバックアップされます。バックアップ ファイルは、オリジナル ファイルと同じディレクトリに作成されます。

ツールは、元の <APPLET> タグ (または Java アプレットの <OBJECT> タグ) の CODE および CODEBASE 属性に指定されている値を使用して、VJSCODEBASE 属性の値を生成します。その例を次に示します。

```
CODE = "MyAppletClass"
CODEBASE = http://www.MyWebSite.com/MyApplet
```

このコマンドは、次のように変更されます。

```
VJSCODEBASE =
http://www.MyWebSite.com/MyApplet/MyAppletClass.dll#MyAppletClass
```

このツールを使用する場合、Java アプレットをマネージ ライブラリ向けにコンパイルするときは、アプレットクラスと同じ名前を使用することをお勧めします。その例を次に示します。

```
C:\MyAppletClassSources>vcj /target:library /out:MyAppletClass.dll *.java
```

J# Browser Control ベータ版でサポートされていない <APPLET> タグおよび Java アプレットの <OBJECT> タグに定義されている属性は、変換プロセスで削除されます。詳細については、「<APPLET> タグと J# Browser Control の <OBJECT> タグ間での属性マッピング」を参照してください。

メモ J# Browser Control ベータ版の既知の問題として、<APPLET> タグおよび Java アプレットの <OBJECT> タグ内のコメントは変換プロセスで保持されない、という問題があります。

コマンドライン オプション

サポートされるコマンドライン オプションは、以下のとおりです。

`/recurse:<wildcard>`
 ファイルの現在のディレクトリとすべてのサブディレクトリを検索し、ワイルドカード仕様に従って変換します。その例を次に示します。

```
TagConvert /recurse *.htm *.html
```

現在のディレクトリとすべてのサブディレクトリにある、拡張子が .htm および .html であるすべてのファイルがアップグレードされます。

`/verbose`
 変換プロセス中に変更されたファイルの名前をプリントします。ファイル名には、ファイルへの完全な修飾子を付加したパスが含まれます。解析したファイルの総数と、変換したファイルの総数もプリントします。その例を次に示します。

```
TagConvert /verbose \AppletSources\Pages\*.htm > changedfiles.txt
```

指定ディレクトリにある、拡張子が .htm のファイルがすべてアップグレードされます。また、変更されたファイルの一覧が、changedfiles.txt ファイルにダンプされます。

`/nologo`
 著作権情報の表示を抑制します。その例を次に示します。

```
TagConvert /nologo \AppletSources\Pages\*.htm
```

指定ディレクトリにある、拡張子が .htm のファイルがすべてアップグレードされます。著作権情報は表示されません。

<APPLET> タグと J# Browser Control の <OBJECT> タグ間での属性マッピング

次の表は、<APPLET> タグの属性と J# Browser Control の <OBJECT> タグの属性におけるマッピングです。

<APPLET> タグ シンタックス (Internet Explorer 拡張機能を含む)	J# Browser Control の <OBJECT> タグ シンタックス
CODEBASE	VJSCODEBASE
CODE	VJSCODEBASE
WIDTH	WIDTH
HEIGHT	HEIGHT
ID	ID
ALIGN	ALIGN
VSPACE	VSPACE
HSPACE	HSPACE
ARCHIVE	(ベータ版ではサポートされていません。)
ALT	ALT
<PARAM>	<PARAM>
alternateHTML	alternateHTML
<PARAM NAME = FireScriptEvents VALUE = True>	(ベータ版ではサポートされていません。)
<PARAM NAME = cabbase VALUE = cabFileName>	(ベータ版ではサポートされていません。)
<PARAM NAME = cabinets VALUE = cabFileNames>	(ベータ版ではサポートされていません。)
<PARAM NAME = useslibrary VALUE = DUFriendlyName>	(ベータ版ではサポートされていません。)
<PARAM NAME = useslibrarycodebase VALUE = DUFileName>	(ベータ版ではサポートされていません。)
<PARAM NAME = useslibraryversion VALUE = DUVersionNumber>	(ベータ版ではサポートされていません。)
<PARAM NAME = namespace VALUE = applicationNamespace>	(ベータ版ではサポートされていません。)

ベータ版では、<APPLET> タグの次の属性はサポートされておらず、J# Browser Control の <OBJECT> タグと同等の属性は存在しません。

archive、cabbase、cabinets 属性 J# Browser Control を .cab、.zip、または .jar ファイルにパッケージする機能は、ベータ版ではサポートされていません。J# Browser Control は、スタンドアロンの .dll ファイルとして、Web サーバーに展開する必要があります。

FireScriptEvents 属性 ベータ版では、HTML ページのスク립トで J# Browser Control によって起動されたイベントシンクはサポートしていません。

useslibrary、useslibrarycodebase、useslibraryversion、namespace 属性 J# Browser Control ランタイム バージョンは、MSJVM の Java Package Manager セマンティクスをサポートしません。

<OBJECT> タグと同等の機能を持つ <APPLET> タグの属性は、そのままコピーされます。J# Browser Control の <OBJECT> タグは、元の <APPLET> タグと同じセマンティクスを維持します。

J# Browser Control を展開する

J# Browser Control の展開は、マネージ ライブラリをコピーして、Web サーバー上の適切なディレクトリに対して HTML ページを更新する、という簡単な作業で行うことができます。マネージ ライブラリは、HTML ページと同じディレクトリまたはサブディレクトリにコピーします。

IIS Web サーバーを使用している場合、仮想ディレクトリの [実行アクセス権] フィールドは、IIS における仮想ディレクトリの既定のアクセス権レベルである [スクリプトのみ] に設定されている必要があります。

複数 DLL のコントロールを展開する

J# Browser Control は、複数の DLL ファイルに分割することができます。分割する場合、HTML ページは、J# Browser Control のメイン クラスを含む DLL を参照している必要があります。その他の DLL については、J# Browser Control ランタイム バージョンにより、実行時の要求に応じてダウンロードされます。

1 つの Web サーバーに複数の DLL のコントロールを展開するときは、その同じ J# Browser Control に関連するすべてのファイルは、同じディレクトリにコピーしてください。ファイルは、CAB、ZIP、JAR ファイルなどにパッケージせず、独立したファイルとしてコピーする必要があります。

複数の J# Browser Control を 1 つのライブラリにパッケージする

複数の J# Browser Control を、1 つのマネージ ライブラリにパッケージすることができます。その場合、各 Browser Control の <OBJECT> タグは、同じマネージ ライブラリの異なるクラス名を指す必要があります。たとえば、MyApplets.dll に MyApplet1 と MyApplet2 という名前の 2 つの J# Browser Control がある場合、次のように、これらの Browser Control を参照します。

```
VJSCODEBASE = http://www.microsoft.com/MyApplet/MyApplets.dll#MyApplet1
```

```
VJSCODEBASE = http://www.microsoft.com/MyApplet/MyApplets.dll#MyApplet2
```

J# Browser Control は、HTML ページと同じ位置またはそのいずれかのサブディレクトリからしかダウンロードできないため、HTML ページのディレクトリが異なる場合は、複数の場所に、マネージ ライブラリをコピーする必要があります。

ユーザーのコンピュータ上で J# Browser Control を実行する

エンド ユーザーの環境で、Internet Explorer 上で J# Browser Control を実行するには、コンピュータ上に J# Browser Control ランタイム バージョンをインストールしておく必要があります。J# Browser Control ベータ版はテスト目的に提供されているため、ユーザーのコンピュータに展開することはお勧めできません。

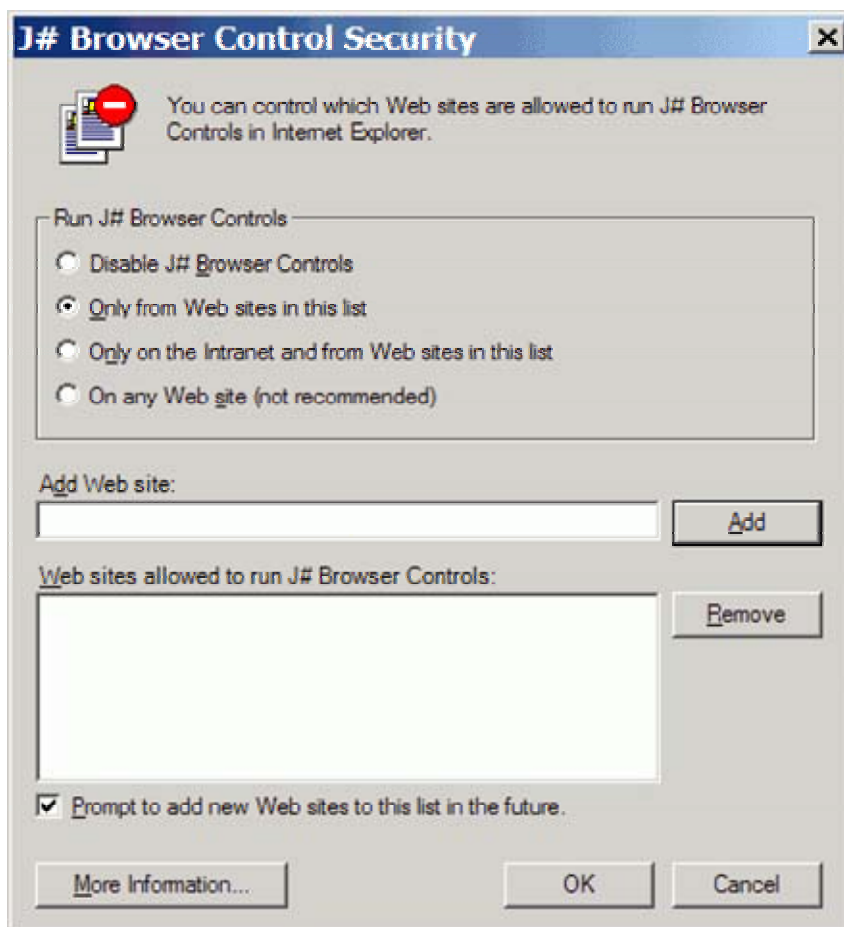
既定では、J# Browser Control ランタイム バージョンは Web ページでホストされるコントロールを実行する前に、エンド ユーザーに対してプロンプトを表示します。ユーザーが J# Browser Control を含む Web サイトをブラウズしようとすると、次のダイアログ ボックスが表示されます。



ユーザーが [Yes] をクリックすると、J# Browser Control がダウンロードされ、実行されます。[No] をクリックすると、J# Browser Control は実行されません。

[Add this site to the list of sites allowed to run J# Browser Control and don't ask me again] をオンにして [Yes] をクリックすると、J# Browser Control の実行が許可されたサイトの一覧に、その Web サイトが追加されます。ユーザーに対して、Web サイトのページに関するプロンプトは表示されなくなります。このチェック ボックスは、既定でオンになっています。

ユーザーは、[J# Browser Controls security] を使用して、J# Browser Control の実行を許可するサイトの一覧を管理することもできます。このダイアログ ボックスは、[コントロール パネル] の [管理ツール] からアクセスできます。Windows XP の場合は、[コントロール パネル] の [パフォーマンスとメンテナンス] からアクセスします。[J# Browser Control Security] アイコンをダブルクリックすると、次のダイアログ ボックスが開きます。



各オプションの機能は次のとおりです。

Disable J# Browser Controls

コンピュータ上で、すべての Web サイトから J# Browser Control が実行されることを防ぎます。

Only from Web sites in this list

ユーザーがリストに追加した Web サイトからのみ、J# Browser Control の実行を許可します。これが既定の設定です。

Only on the Intranet and from Web sites in this list

イントラネットまたはユーザーがリストに追加した Web サイトからのみ、コンピュータ上での J# Browser Control の実行を許可します。

On any Web site

どの Web サイトからの J# Browser Control でも、コンピュータ上での実行を許可します。

[Add Web site] ボックスに Web サイトのアドレスを入力して [Add] をクリックすると、J# Browser Control の実行を許可するサイトの一覧に、その Web サイトが追加されます。同様に、[Web sites allowed to run J# Browser Controls] から Web サイトを選択して [Remove] をクリックすると、リストからサイトが削除されます。J# Browser Control の実行を許可するサイトの一覧に追加された Web サイトについては、そのサイト上で J# Browser Control を実行してもプロンプトは表示されません。

既定では、[Prompt to add new Web sites to this list in the future] はオンに設定されています。これは、J# Browser Control ランタイム バージョンに対して、リストにない Web サイトの J# Browser Control を実行する前に、ユーザーに対してプロンプトを表示するように指示するものです。このチェック ボックスをオフにすると、リストにない Web サイトの J# Browser Control を実行しても、ユーザーに対してプロンプトは表示されません。

ベータ版ではサポートされない機能

以下の機能は、J# Browser Control ベータ版ではサポートされていません。

HTML ページのスクリプトから J# Browser Control へのアクセス

HTML ページのスクリプトから、J# Browser Control のパブリック メソッドおよびパブリック変数へのアクセスはサポートされていません。HTML ページのスクリプトで、<param name=FireScriptEvents value=True> 属性を使用して J# Browser Control が起動したイベントシンクもサポートされていません。

信頼ベースのセキュリティ

MSJVM でサポートされている信頼ベースのセキュリティ セマンティクスは、サポートされていません。

Java Package Manager

MSJVM でサポートされている Java Package Manager の機能は、J# Browser Control ではサポートされていません。J# Browser Control は、クラスをローカルにインストールし、署名付きアクセス権に基づいてそれらのクラスを実行するときに使用する、オブジェクト キャッシュはサポートしません。したがって、Distribution Units のインストールおよび実行もサポートされません。

アーカイブ ファイル

アーカイブ ファイルは、サポートされていません。J# Browser Control は、複数のマネージ ライブラリに分割することができます。ただし、Web サーバーにマルチ ライブラリ コントロールを展開するときには、ファイルは CAB、ZIP、JAR ファイルなどにパッケージせず、独立したファイルとしてコピーする必要があります。

デザイナの非サポート

J# Browser Control に対応したデザイナーサポートは、Visual Studio .NET にはありません。

まとめ

J# Browser Control は、開発者に対して、Java アプレットを .NET Framework 上で実行するための移行手段を提供します。J# Browser Control は、移行前の Java アプレットと同様のランタイム機能を保有し、また、Java 言語セマンティクスを維持しています。また、J# Browser Control は、.NET Framework に対してフル アクセスが可能です。XML Web サービスに対応したネイティブ サポートにアクセスする機能も備えています。