Mac OS Runtime for Java 2.1

MRJ と Mac OS での Java 開発

Apple は Mac OS でのソフトウエア実行環境として Java をサポートしています。Java をサポートする基本システムが MRJ（Mac OS Runtime for Java）です。この文書では、MRJ を中心にして、Mac OS 上で Java を使ったソフウエア開発を行うときに、どのような機能が提供されているのかをまとめてあります。まず、Java の一般的な利点などをまとめておきました。そして、Mac OS での Java 実行環境について、さらには開発のときに Mac OS 向けに留意しなければいけないことをまとめておきました。開発ツールを使った実際の開発手順の基本的なことも記載しました。

目次

第 2 版

1999 年 3 月

第 1 章　Java によるソフト開発と MRJ

1.1　Java 開発の意義

Java は現在のコンピュータの世界の中で、1 つの大きな枠組みとしてクローズアップされてきます。Java のこれまでの経緯となぜ Java が注目されているかについて、まず最初にまとめておきましょう。

●Java とアプレット

Java は 1995 年より開発者向けにはリリースされていましたが、1996 年の始めに正式に登場しました。Java はプログラミング言語の 1 つという見方ができます。実際、C++を改良した言語としての Java が存在します。しかしながら、その Java で記述し作られたソフトウエアが従来にない機能を提供し、さらにソフトウエアを作成するためのクラスライブラリでも豊富な機能を提供することで、現在の Java は OS 環境にも匹敵すると言ってもよいでしょう。

当初の Java は、Web ブラウザ上で機能する「アプレット」という形式のソフトウエアが中心的な存在でした。アプレットは Web ページとして記述した HTML 文書から呼び出す形式のソフトウエアで、ソフトウエア自体は Web サーバーからダウンロードされ、実際の実行はクライアント側で行います。Web サービスは特定の OS 向けではなく、クロスプラットフォーム対応になっているので、ダウンロードした先のクライアントの OS は一定ではありません。それでも、1 つのファイルに納められたアプレットというソフトウエアが、Mac OS でも Windows でも UNIX でも実行できるというのが Java アプレットの大きな特徴です。こうしたクロスプラットフォーム対応していることと、インターネットベースでのソフトウエア開発を行うという需要が結びつき、Java のリリース当時は非常に注目が集まりました。

現在でも、もちろん、Web ブラウザで Java アプレットを制作するということは行われていますが、Java はもっと広い範囲で適用されるようになってきています。また、Web ブラウザで動くソフトウエアも、Flash のようなオーサリング系ツールで事が足りることが多くなり、Java 以外の選択肢も充実してきたため、Java 一色ではなくなっているというのが実情です。

●システム開発とアプリケーション開発

現在の Java の世界で注目されているのは、サーバー上での実行を中心としたシス

テムやあるいは分散システムの開発のベースになるという点です。Web サーバーから呼び出されるソフトウエアとしての「サーブレット」や、あるいはソフトウエアの部品を組み合わせてシステム構築を行う場合に、ネットワークを経由して異なるマシンに存在するソフトウエア同士のやりとりを実現する Enterprise Java Beans などが代表的な手法です。Java の世界では、こうした最新のソフトウエアテクノロジーを機能としてサポートすることによって、新しい分野で積極的に利用できるような環境を整えています。また、そうした状況を実際の開発に取り入れられています。

Java のソフトウエアは、アプレットやサーバー向けのものだけでなく、一般的なアプリケーションも、作成できます。Java の機能が OS に匹敵するようになると、通常のアプリケーションも、OS 本来の機能を使うのではなく、Java で提供されたライブラリを用いて作成することができます。そうしたアプリケーションでも限られた機能ではなく、通常の OS 機能を使った場合とほとんど遜色ないものが作成できるまで発展してきています。Java で作ったプログラムの多くの部分あるいはすべての部分は、クロスプラットフォーム対応なので、アプリケーションであっても別の OS 向けへの移植は従来の開発手法に比べて非常に短期間で低コストで可能になります。

●クロスプラットフォームの実現

Java が提供するクロスプラットフォームの機能がどのようにして実現しているのでしょうか。そこが、Java の実行環境の重要な部分です。Java のソフトウエアを実行するための仮想的なコンピュータとして、Java Virtual Machine（Java VM）が定義されています。そして、OS 上で、Java VM のエミュレータを組み込むか、あるいは Web ブラウザが Java VM を構築するなどして、いずれにしても Java VM を何らかの形で実現します。そして、たとえばアプレットがあれば、Web ブラウザは Java VM を起動し、その Java VM がアプレットを起動して、ソフトウエアの実行を行います。

Java VM によって実行できるソフトウエアは、「バイトコード」という実行可能なバイナリです。バイトコードは、Java VM でしか実行できない一種の機械語と考えて良く、そのままでは Power PC や Pentium では実行できないのです。しかしながら、OS や Web ブラウザで Java VM を用意することで、CPU や OS が異なっても同一のバイトコードが実行できるわけです。アプリケーションやなど他の形式の Java ソフトウエアでも、同様に Java VM を起動してその上でバイトコードが実行されるのが基本です。

Java という言語でソースを作成し、コンパイルすることで Java VM に対応したバイトコードのオブジェクトが作成されます。Mac OS や Windows で Java VM が搭載されていれば、共通のバイトコードのアプレットが実行でき、つまりは 1 つのプログラムファイルがいろいろな OS 上で実行できるということになります。

●プログラミング環境としての Java

Java のもう 1 つの側面は、開発環境としての Java の優れた点です。C++をベースにして開発された言語ですが、オブジェクト指向を全面的に取り入れ、C++において複雑で間違いのおきやすい機能を省略するなど、より開発がスムーズに行われるような改良がなされています。

まず、いちばん大きなポイントは、メモリー管理をほぼシステムまかせにできるという点でしょう。メモリの確保はオブジェクトの生成で自動的に行われます。使わなくなったメモリ、つまりオブジェクトは、Java VM が自動的に解放します。これは「ガベージコレクション」という機能を使って、システム側で自動的に使わなくなったメモリを解放するのです。そのため、プログラムとしてメモリの解放を記述する必要は多くの場合必要なくなるため、不適切な解放やあるいは解放のし忘れなどによる不安定要素は言語自体に持っていないということになります。

また、オブジェクトを通じてメモリを利用するため、C/C++でのポインタに相当するデータは Java では使う必要がなくなっています。Java ではオブジェクトへの参照というデータは持ち得ますが、参照は計算の対象にはなりません。一方、C/C++のようにデータ管理で自由に計算処理ができるポインタがあるため、柔軟な処理はできるとは言え、関係ないメモリ領域でも平気で読み書きできてしまい、それがシステム全体の不安定要素にもなっています。Java ではそうしたメモリーリークの問題も、言語自体で発生しないように考えられています。

●洗練されたライブラリ機能

実際のソフトウエアを組むとなると、ウインドウを表示したり、ボタンを配置したり、あるいはファイル処理やネットワーク処理などの機能を利用しなければなりません。そうした処理は、クラスライブラリとして提供されています（「パッケージ」とも呼ばれます）。GUI を構築する場合でも、シンプルな機能から豊富なコントロール類が使えるパッケージまで用意されており、現状では一般的な OS の機能に匹敵するかあるいは十分に超えていると言えるでしょう。基本的なウインドウなどの GUI コンポーネントは、AWT（Abstract Windowing Toolkit）としてパッケージとしてまとめられています。さらに、たくさんの種類のコントロールが利用できる Swing あるいは JFC（Java Foundation Class）についても、実用化されています。

また、ネットワーク処理が非常に手軽に行えることも、大きな特徴です。こうしたライブラリの機能が充実しており、そして洗練された設計になっていることも Java の特徴です。また、こうしたライブラリ機能も、Java VM と同じように Java の供給元で統括されているため、OS に寄らずに共通になっています。こうしたライブラリを備えていることも、クロスプラットフォームを実現する 1 つの重要な要素です。

他にも、マルチスレッド機能をサポートしていることや、ダウンロードして実行す

る場合にコードを検証してから実行できる安全性を備えていること、あるいはライブラリなどを実行時にリンクできるなどの特徴もあります。

以上のように、開発言語として見た時の Java には従来にない特徴がありますが、概してより効率的な開発が可能な環境を提供するものとして総括できるでしょう。

●Mac OS 向けのソフトウエアも Java で開発できる

OS には OS 独自の API があり、それを利用すると、もちろんそのソフトは別の OS では機能しません。ただし、Java は多くの範囲を共通のクラスライブラリを使ってプログラミング可能です。Mac OS 向けのソフトウエアを作る時に、もちろん Mac OS 独自の機能を利用することはできますが、たとえばウインドウ処理や GUI コンポーネント、ファイル処理やネットワーク処理など、多くの部分は共通のクラスライブラリを使って行えます。

つまり、Java という OS を超えた共通の枠組みを利用したソフトウエア開発を Mac OS でもでき、そうして作成したソフトウエアは Mac OS でも利用できるのです。続いて、Mac OS 上での Java 環境がどのように実現されているかを説明しましょう。

1.2 MRJ とは何か

Apple は、Java の実行環境として、MRJ（Mac OS Runtime for Java）をリリースしています。MRJ の位置付けなどについて説明をしましょう。

●Java のソフトウエアを実行する機能拡張

1.1 で説明したように、Java のソフトウエアを実行するには、Java VM が必要です。Java がリリースされた当初は、Java VM が Web ブラウザに組み込まれている場合もあったのですが、最近の傾向として、Java VM は OS に組み込まれています。アプリケーションは、その OS に組み込まれた Java VM を利用しますが、Web ブラウザでも同様な動きをするのが一般的になってきています。

MRJ（Mac OS Runtime for Java）は Apple が提供する Mac OS 向けの Java VM です。最新版として、MRJ Ver.2.1 が 1999 年 2 月にリリースされています。MRJ 自体は、機能拡張として働くファイルのですが､たとえば Mac OS で機能するように作られた Java アプリケーションは、MRJ を利用して実行を行います。

Web ブラウザについても、MRJ を使うのが一般化しつつあります。Internet Explorer は、従来から Microsoft 社が開発した Java VM と MRJ を切り替えるようにできていましたが、Internet Explorer Ver.4.5 からは、使えるのは MRJ のみになりました。

Netscape Navigator Ver.4.5 については、独自の Java VM を組み込んでおり、MRJ は基本的には利用できません。Ver.5 より Java Plug-In を利用することで、MRJ が利用できるようになる予定です。Ver.4.X 向けの Java Plug-In もリリースされていますが、開

発者向けのリリースになっています。

●MRJ は JRE に相当

Java のリリースは、Sun Microsystems が行っていますが、Sun からは、JDK（Java Development Kit）として、コンパイラなどの基本開発ツールや、ライブラリ、Java VM などのセットがリリースされています。この JDK が Java 実行環境の 1 つの基準になっています。また、開発者ではない一般の Java 利用者向けには、Java の実行環境だけを含めた JRE（Java Runtime Environment）がリリースされています。

MRJ は、Mac OS 環境における JRE に相当します。JDK に対応するものとして、MRJ SDK というものがやはり Apple から配付されています。JDK と JRE は、各 OS ごとに必要ですが、最初は Sun より配付されていたものの、現在 Sun によって配付されているのは Solaris 版と Windows 版のみで、その他の OS はライセンス先から実行環境や開発キットを配付するようになっています。MRJ と MRJ SDK は、Mac OS 環境での Java 環境のリリースとなっています。

Web ブラウザが MRJ を利用する方向にあり、また、Mac OS 自体に MRJ が付属するなど、Mac OS の Java 実行環境は、MRJ に集約されつつあります。

●入手方法

MRJ は Mac OS に付属しますが、Mac OS 8.5 では 1 つ前のバージョンである MRJ 2.0 が付属しています。最新版の MRJ は、以下の URL から入手できます。

http://www.apple.co.jp/java/

1.3 MRJ が提供する機能

Mac OS での標準 Java 環境である MRJ がどのような機能を提供しているかを説明しましょう。

●JDK1.1.7 に対応した VM とクラス

MRJ 2.1 は、JDK 1.1.7 に対応した Java VM とライブラリが含まれています。つまり、JDK 1.1.7 の Mac OS 版が MRJ 2.1 と言っても良いでしょう。まず、規格にのっとった Java VM が動作しており、JDK で示された標準のクラスライブラリ機能はすべて MRJ でも利用できます。つまり、Java の標準環境を Mac OS 上で実現するのが MRJ の 1 つの大きな役割です。

MRJ 2.1 は GUI コンポーネント群の Swing への対応も行われていますが、Swing 自体は MRJ 2.1 には含まれていません。しかしながら、Swing がスムーズに機能するよ

うに開発が行われており、Swing を使ったソフトウエア開発も安心して行えます。

●Mac OS の機能を使う独自拡張

MRJ が JDK で規定された標準機能をサポートするのはもちろんですが、Mac OS で必要とされる機能についてのサポートも行っています。これらの一部については、第2章で詳細を説明しますが、ここでは概要だけを説明しましょう。

まず、MRJToolkit として、Mac OS で機能するアプリケーションとして必要な処理を組み込めるようになっています。Mac OS では、Finder でアプリケーションのファイルに文書ファイルをドラッグ&ドロップして起動したり、あるいは文書ファイルをダブルクリックしてアプリケーションが起動するようなことを実現しなければなりません。そのため、Finder からの基本的なイベントに対応したり、あるいはファイルタイプやクリエイターの設定を行うような機能を、Java のアプリケーションで利用できるようになっています。

そして、JDirect として、Mac OS の機能を Java アプリケーションから利用できるようになっています。Mac OS の API コールを行うことができますが、それだけでなく、Mac OS の機能を Java のライブラリのようなオブジェクトとして定義された使い方もできるようになっています。

JManager という機能もありますが、これは、C や C++などの言語から、Java で作られたコンポーネントを利用するための機能を提供するものです。

さらに、MRJScripting として、Java で作られたアプリケーションやアプレットのコンポーネントを、AppleScript でコントロールできるようにする機能も提供します。

1.4 MRJ SDK について

Mac OS での開発環境として提供されている MRJ SDK について、内容を説明しておきましょう。開発ソフトは市販の製品などがありますが、Java のソフト開発を Mac OS で行う人は、MRJ SDK は必ず入手しておくべきでしょう。

●MRJ SDK に含まれるもの

MRJ SDK には、MRJ 独自の機能である MRJToolkit、JDirect、JManager、そして MRJScripting の各機能を使うためのドキュメントが含まれています。また、C や C++、Pascal で使うヘッダーファイルなども含まれています。

そして、開発用のツールとして、Java コンパイラの javac や、JavaDoc ジェネレータの javadoc、アーカイブファイル作成などの jar などが含まれています。JDK のツールに相当しますが、デバッガについては含まれません。また、アプリケーション実行環境やアプレット実行環境は、以下に説明するように別のアプリケーションで提供されています。

javac などは独立したアプリケーションとして利用できますが、MPW からこれらの Java 関連ツールを利用できるようにもなっています。

●Applet Runner について

アプレットの実行は開発後はブラウザで行いますが、開発中のテスト実行などのために、Apple Applet Runner が MRJ SDK には付属しています。アプレットを含む HTML ファイルをドラッグ&ドロップすることで、その HTML ファイルに記述が含まれたアプレットの実行を行うことができます。ローカルにあるアプレットやあるいはネットワーク上にあるアプレットを実行できます。

Applet Runner で実行中のアプレットは、AppleScript でコントロールすることもできます。また、Applet Runner 自体も AppleScript でさまざまにコントロールすることができます。

●JBindery について

MRJ SDK でのアプリケーション実行環境として、JBindery が付属します。これは、コンパイルした結果の Java ソフトウエアのうち、アプリケーションとして機能するように作られたものを実行させるのが 1 つの機能です。また、そうして作られたファイルから、Mac OS 上でダブルクリックして起動可能なアプリケーションを作るのも JBindery で行えます。

実行のために必要なパラメータの設定などを行えると同時に、作成したアプリケーションにリソースの組み込みができるなど、Mac OS 独自の機能も備えています。

●リファレンスとしての MRJ SDK

MRJ SDK は Mac OS で Java 開発するための唯一のツールということではありません。市販の開発ツールとしていくつかのソフトが利用できます。たとえば、CodeWarrior を使えば、Java アプリケーションを作成し、Mac OS 上でダブルクリックして起動可能なアプリケーションまで生成できます。MRJ SDK はその意味では、開発ツールに組み込むべき機能のリファレンスとしての性格も持っているものです。

1.5 Apple の Java への取り組み

Apple の Java への取り組みは MRJ だけではありません。MRJ とは別の Java 環境についてまとめておきましょう。

●WebObjects と Java サポート

Mac OS X Server に標準で付属する WebObjects 4 は、Web ベースのアプリケーション開発ツールです。クライアントとして Web ブラウザを使い、サーバー上でデータベース利用を含むさまざまな機能を組み込んだシステムを開発するのに利用できるも

のです。さまざまなツールのサポートにより、短期間で実用的なシステム開発が可能な点が高く評価されており、Web サービスを利用したオンラインショッピングなどのサイト構築では広く利用されています。

WebObjects 4 では、Web ブラウザの機能をクライアントで利用するために、1 つの方法として、結果を Web ページとして表示する手法を取ることができます。これとは別に、データベース利用などを行う Java アプレットを生成するという機能もあります。つまり、クライアントで機能する Java アプレットを生成し、それによって Web サーバー経由でデータベース利用などができるわけです。こうしたアプレットを、ビジュアル開発ツールを使って手軽に作成できると同時に、必要なら Java のソースを独自に追加するなどクライアント向けのソフトウエアを Java によるアプレットで作成できます。

●Mac OS X 独自の機能を使うプログラムを Java で作成

Mac OS X および Mac OS X Server では、オブジェクト指向ベースの新しい機能が提供されています。この Mac OS X 独自の機能を使うプログラムを作るための基本的な言語は Objective-C ですが、Java のソフトウエアからも Mac OS X 独自の機能は使えるようになっています。Mac OS X 独自の機能を使うクラスを利用することにより、Java のプログラムでその機能を利用することが可能になります。

第 2 章　Mac OS 独自の機能を使う

2.1　MRJToolkit の概要

MRJToolkit は、Java で Mac OS のアプリケーションを開発するときには必ず使うと言ってもよい機能です。MRJToolkit の概要についてまず説明しましょう。

●Java で作ったアプリケーションを Mac OS 対応に

Java の開発言語として優れていることや、あるいはクロスプラットフォームに注目するなど、いずれにしても Java を使ったソフト開発は実用的な視野に入っています。しかしながら、Java の標準的な機能だけでは、Mac OS で機能するソフトウエアとして不足します。Java のライブラリ機能は、やはり Windows や Solaris がメインになっているという見方もできますが、むしろ Mac OS には Mac OS 独自の機能があるという見方もできるでしょう。

まず、Mac OS で重要な概念として、ファイルタイプとクリエイターというファイル情報があります。あるアプリケーションで作成した文書ファイルをダブルクリックして開くには、まず、文書ファイルから作成したアプリケーションを対応付ける必要があり、Mac OS ではそのための情報としてファイルタイプとクリエイターを使います。Java の標準のライブラリでは、作成したファイルのクリエイタやファイルタイプを変更する機能がないので、その機能を MRJ は提供しています。

また、Finder で文書ファイルをダブルクリックしたり、あるいは文書ファイルをアプリケーションにドラッグ&ドロップすれば、まずアプリケーションが起動します。その後、起動したアプリケーションに対して、ファイルを開くように指示する AppleEvent が送付されます。Mac OS 上のアプリケーションは、こうした AppleEvent に応答するような作りになっていなければなりません。やはり AppleEvent 処理についても、Java の標準機能には含まれていないので、MRJ として AppleEvent を処理する機能をサポートしています。

なお、あるアプリケーションがサポートするファイルタイプの情報は、アプリケーションファイル内の BNDL リソースに含まれます。リソースについては別途用意して、作成したアプリケーションにマージする必要があります。JBindery を始め、開発ツールではそうした機能を一般には利用できるようになっているので、必要なリソースを ResEdit などで作成しておいてマージします。

●MRJ にクラスとして組み込まれている MRJToolkit

Mac OS 独自の機能のうち、アプリケーションで必須の機能を MRJToolkit で提供しています。MRJToolkit はいくつかのクラスを含んだクラスライブラリです。MRJToolkit 自体は、MRJ をインストールすることによって自動的にシステムに組み込まれます。つまり、MRJ が機能していれば、MRJToolkit は利用できる状態になっているというわけです。

MRJ SDK では、MRJToolkit の詳細なドキュメントに加えて、MRJToolkit を使うアプリケーションをコンパイルするときなどに使う MRJToolkitStubs.zip というファイルも含まれています。

2.2 ファイル処理の機能

MRJToolkit がサポートするファイル処理の機能についての概要を説明をしましょう。詳細な点については、MRJ SDK に付属するドキュメントを御覧下さい。

●Java の標準機能としてのファイル処理

Java の標準クラスライブラリには、ファイル処理を行うパッケージが組み込まれており、アプリケーションなどで利用できます。テキストやあるいはバイナリデータをファイルに書き込んだり、あるいは読み込んだりができます。Mac OS で機能するアプリケーションでも、基本的なファイル処理は、こうした Java の標準的なファイル機能を利用します。その標準のライブラリで提供されていない機能のうち、Mac OS でどうしても必要になる機能を MRJToolkit で提供しています。

●ファイルタイプ、クリエイターの処理

MRJToolkit を利用すれば、既存のファイルのファイルタイプやクリエイターを書き換えることができます。また、新しく作るファイルのタイプやクリエイターを設定することもできます。既存のファイルのタイプやクリエイターを読み取ることもできます。つまり、ファイルタイプやクリエイターに関する処理は、すべて提供されていることになります。

●フォルダ検索

MRJToolkit の機能で、システムフォルダやあるいは初期設定フォルダなど、システムが管理するフォルダを検索することができます。たとえば、初期設定フォルダに、何らかの初期設定ファイルを作ることや、あるいはデスクトップのファイルを調べるなどという場合、この機能を利用して、まずフォルダへの参照を得ます。そして、そのフォルダにファイルを作成するなど、必要な処理を組み込むことになります。

●アプリケーション検索

Java で作ったアプリケーションから、別のアプリケーションを起動したいこともあります。その時に便利なのが、アプリケーションのファイルを検索する機能です。クリエイターを与えると、そのクリエイターを持つファイルへの参照が得られます。これを利用して、そのアプリケーションを起動することができます。アプリケーションの起動は Java の標準的な方法を使って行うことができます。

2.3 必須イベントの処理

MRJToolkit の機能として、アプリケーションが対応すべき必須イベントの処理に対応させる機能があります。これらについて説明しましょう。詳細な利用方法については、MRJ SDK のドキュメントなどを参照してください。

●Open、Print イベントへの対応

Finder で文書ファイルをダブルクリックしたり、あるいはアプリケーションにドラッグ&ドロップしたときには、Open イベントがシステムからアプリケーションに送付されます。また、Finder 上でファイルを選択して印刷を行うと、アプリケーションには Print イベントが送られます。これらのイベントには、どのファイルを開くあるいは印刷するのかという情報が含まれています。

これらのイベントを Java で作ったアプリケーションでも対応しないと、Mac OS でスムーズに利用できないかもしれません。そのための機能が MRJToolkit には含まれています。

こうしたイベントへの対応は、ちょうど、Java の標準ライブラリで規定されたイベント処理であるデリゲーションイベントの手法によく似た方法で行えるようになっています。これらのイベントが発生したときに、指定したクラスの規定されたメソッドを呼び出すようにします。どのクラスを呼び出すのかということをシステムに対して登録しておくと、後はイベントが発生すればそのクラスの規定された名称のメソッドが呼び出され、引数には開くないしは印刷すべきファイルへの参照が得られるようになっています。後は、そのメソッドで実際にファイルを開くなどの作業を組み込みます。

●Quit イベントへの対応

もう 1 つの重要なイベントとして Quit イベントがあります。たとえば、Mac OS をシャットダウンするときに、起動しているアプリケーションを強制的に終了させる場合、システムはアプリケーションに Quit イベントを送付します。このイベントを受けると、アプリケーションは終了しなければなりません。

ただし、MRJ 環境下のアプリケーションでは、特にプログラムを組み込まなくても、

終了処理は正しく行われます。MRJ のアプリケーションは、アップルメニューに自動的に「終了」という項目が付加され、それを選択すると終了できるようにもなっています。

Quit イベントの処理についても、Open などと同じように、システムに登録することによって規定のメソッドが呼ばれることになります。アプリケーション側ではファイルを保存させるなど通常は何らかの終了時の処理が必要になるので、結果的には Quit イベントの処理を組み込むことになるでしょう。

なお、Finder の必須イベントとしては、他に OpenApplication があり、アプリケーションをダブルクリックして起動したときに、新規のドキュメントを用意するなどの機能を組み込みます。ただし、これについては、「起動時」というタイミングで発生するので、わざわざイベント処理として組み込む必要はないとも言えるでしょう。MRJToolkit ではこのイベントには対応していません。

●アップルメニューの「情報」に対応する

アップルメニューのいちばん上の項目は、そのアプリケーションに関する情報を表示することになっています。MRJ 環境下のアプリケーションは、自動的に「情報」という項目が付加されます。しかしながら、アップルメニューというのは Mac OS 独自の機能です。つまり、Java のアプリケーションからは、アップルメニューというものにはアクセスできないことになります。

そこで、MRJToolkit の機能を利用すれば、アップルメニューの「情報」を選択すると、イベントが発生するというメカニズムを使えるようになります。やはりこれもデリゲーションイベントのような使用方法になります。イベントが発生したときに呼び出すクラスを、システムに対して登録します。メニューが選択されるとイベントが発生して、そのクラスの規定のメソッドが呼び出されるという仕組みです。そのメソッドで例えばダイアログボックスなどを表示すれば良いでしょう。

●MRJToolkit を利用したプログラム例

実際に Open イベントなどに対応する簡単なアプリケーションのサンプルは次のようなものです。たとえば、Open イベントに対応させるポイントとしては、次のようにまとめられます。

- com.apple.mrj.*をインポートしておく
- Open イベントに反応するクラスには、MRJOpenDocumentHandler のインタフェースをインプリメントしておく
- アプリケーションを実行する最初の段階で、registerOpenDocumentHandler メソッドを使って、Open イベントに反応するクラスをシステムに登録してお

く

- MRJOpenDocumentHandler の定義に従って、Open イベントによって呼び出される handleOpenFile メソッドを定義する

以上のような対応を、4 つのイベントに対して行っています。ただし、イベントの処理は、イベントが発生したことをコンソールに出力しているだけです。通常はその部分に実際にファイルを開くなどの処理を組み込む必要があります。

```
import com.apple.mrj.*;   //MRJToolkit のパッケージを取り込む
import java.io.*;         //File クラスを利用する

public class TrivialApplication
    implements    MRJOpenDocumentHandler,  //Open イベントのハンドラインタフェース
                  MRJPrintDocumentHandler,   //Print イベントのハンドラインタフェース
                  MRJQuitHandler,    //Quit イベントのハンドラインタフェース
                  MRJAboutHandler   //About イベントのハンドラインタフェース
{

    public static void main(String args[]) {          //アプリケーションの起動で呼び出される
        System.out.println( "Hello World!" );      //コンソールに文字列を出力
        new TrivialApplication();                  //自分自身のインスタンスを生成

    }
    public TrivialApplication()      //コンストラクタ。インスタンス生成時に呼び出される
    {
        MRJApplicationUtils.registerOpenDocumentHandler(this);
            //Open イベントでこのオブジェクトが呼び出されるように設定する
        MRJApplicationUtils.registerPrintDocumentHandler(this);
            //Print イベントでこのオブジェクトが呼び出されるように設定する
        MRJApplicationUtils.registerQuitHandler(this);
            //Quit イベントでこのオブジェクトが呼び出されるように設定する
        MRJApplicationUtils.registerAboutHandler(this);
            //About イベントでこのオブジェクトが呼び出されるように設定する
    }

    //Open イベントが発生したとき、呼び出されるメソッド
    public void handleOpenFile(File filename)
    {
        System.out.println("Dragged "+filename.toString());
            //コンソールにドラッグしてきたファイルのパスを表示
    }

    //Print イベントが発生したとき、呼び出されるメソッド
    public void handlePrintFile(File filename)
    {
        System.out.println("Print Event "+filename.toString());
            //コンソールに印刷するファイルのパスを表示
    }

    //Quit イベントが発生したとき、呼び出されるメソッド
    public void handleQuit()
    {
        System.out.println("Quit Event Received");    //コンソールに文字列を表示
        System.exit(0);              //アプリケーションを終了する
    }
```

```
        //About イベントが発生したとき、呼び出されるメソッド
        public void handleAbout()
        {
            System.out.println("Select About Menu");      //コンソールに文字列を表示
        }
    }
```

2.4 JDirect の概要

Mac OS の機能を直接利用する JDirect についての概要を説明しておきましょう。JDirect についてはドキュメント化されていない部分もあり、MRJ SDK でも必ずしもすべての情報は含められているわけではありません。

●Mac OS 機能へ直接アクセス

Java の標準機能として、JNI（Java Native Interface）というものがあり、たとえば、C で作ったライブラリを Java から呼び出すということができます。これと似た物として JDirect という機能が MRJ には組み込まれています。JNI は汎用的な意味で、C などで作られたライブラリを呼び出す機能として、MRJ でも利用できます。これに対して、JDirect は、Mac OS の API つまり Toolbox の機能を利用するための機能を提供する一種のライブラリです。Toolbox の API を利用するためには JNI の機能を使えばできなくはないのですが、JDirect では Java で必要な定義が行われているので、それをライブラリとしてすぐに利用できるようになっているものと考えれば良いでしょう。

単に Toolbox の API をコールするだけの機能もありますが、たとえば、Toolbox でのファイル参照として FSSpec という構造体があります。その構造体の Java 版とでも言うべきクラスの定義があるなど、Toolbox の機能を Java 的に使えるような機能を提供するのが JDirect の特徴です。他には、AppleEvent の送付なども JDirect によって提供されるクラスで可能になっています。

●機能の一部はドキュメント化されている

ただし、JDirect についてはすべてがドキュメント化されているわけではありません。一部のクラスについて、そのオリジナルの Java ソースファイルが MRJ SDK で提供されています。また、一部のクラスについても、MRJ SDK でドキュメントが公開されています。

実際に利用される範囲などを含めて今後の状況を見て、より広い範囲でドキュメントは公開されることになるものと思われます。

第 3 章　CodeWarrior を使った開発

3.1　アプレットの開発

Mac OS でソフト開発を行っている人の多くは CodeWarrior を使っていることと思います。C や C++での開発だけでなく、Java にも対応しています。CodeWarrior を使ってアプレットを開発する基本的な手順を説明しましょう。C/C++である程度 CodeWarrior を使った経験がある方を想定しています。

●プロジェクトの用意

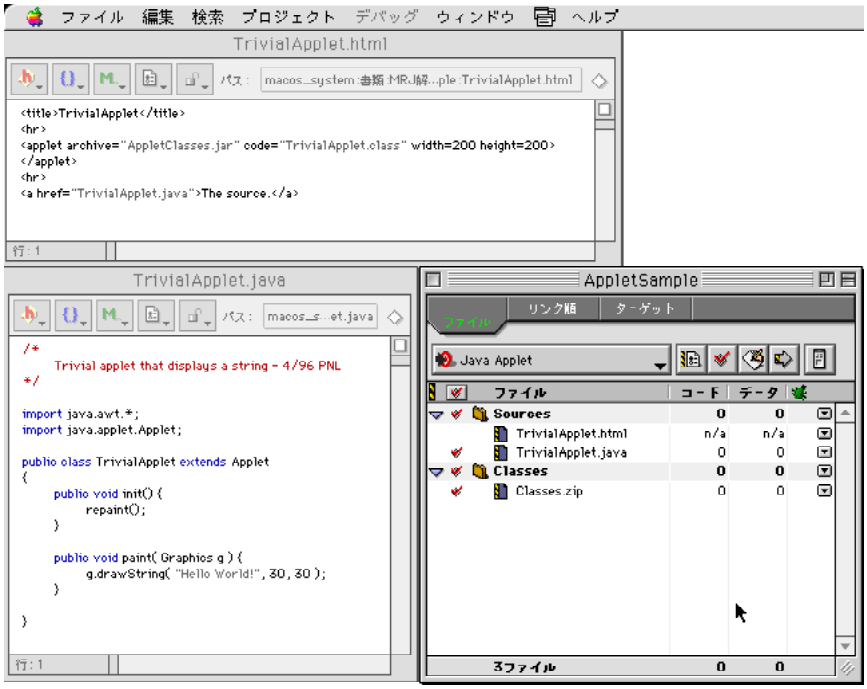
まず最初は、プロジェクトの用意です。「ファイル」メニューから「新規プロジェクト」を選択すると、次の図のようなダイアログボックスが表示されます。まず、ここで、Java の分類の左にある三角形の部分をクリックして下位項目を表示します。アプレットを作成するときには「Java Applet」を選択すれば良いでしょう。

新規プロジェクトで Java Applet を選択する

この後、プロジェクトファイルの指定を行いますので、適当なフォルダを指定します。「フォルダを作成」のチェックボックスが入っているので、指定した名前のフォルダを作り、その中に同じ名前のプロジェクトファイルを作成します。

●ソースの作成

プロジェクトの初期状態は次のようになっています。最初から、アプレットのひな形となるソースの TrivialApplet.java が含まれています。また、アプレットを呼び出す HTML ファイルとして、TrivialApple.html も作成されています。



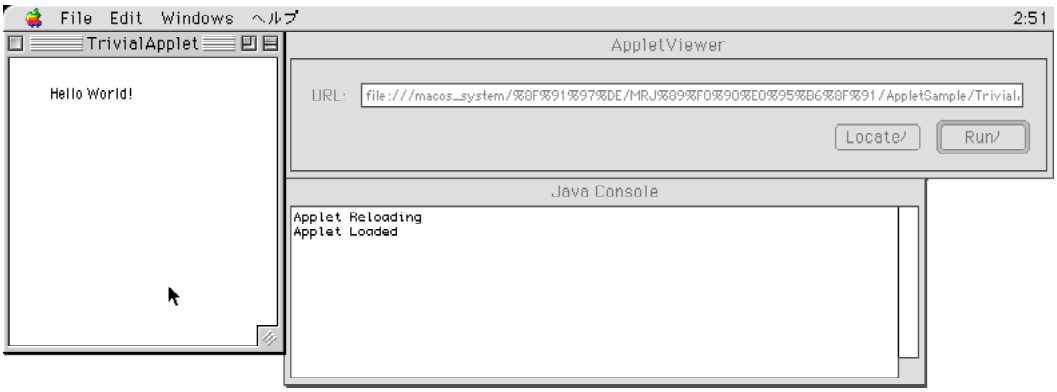
プロジェクトを作成した最初の状態

Classes.zip は、Java の標準ライブラリを利用するためのファイルですので、通常は必要になります。削除などはする必要はありません。

TrivialApplet ではない名前でアプレットを作成したい場合、まず、プロジェクトから、TrivialApple.java を取り除きます。そして、そのファイルの名前を＜作成したいアプレットのクラス名＞.java に変更し、そのファイルを再度プロジェクトに登録します。そして、ソース中の TrivialApplet という記述を、作成したアプレットのクラス名に変更すればよいでしょう。

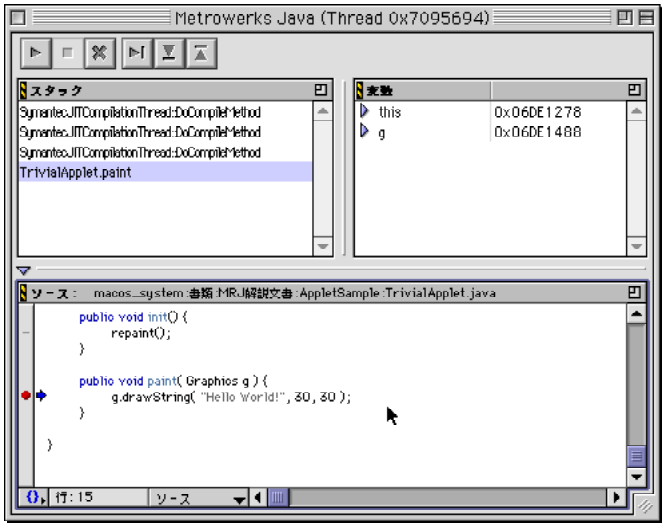
●コンパイルとテスト

CodeWarrior で「プロジェクト」メニューから「実行」あるいは「デバッグ」（Command+R）を選択すると、アプレットを実行します。Metrowerks Java という独自の Java 実行環境が起動しますが、既定値では Java VM は MRJ を利用するようになっています。「実行」の場合は、しばらく待つと、ウインドウでアプレットが実行されます。



Metrowerks Java 上で、アプレットが起動した

「デバッグ」の場合はデバッガが起動します。デバッガは、C や C++でのプログラミングと同様に行えます。



デバッグを有効にしておくとデバッガが起動する

アプレットのプログラム作成は、このように、比較的簡単にできます。Java VM として MRJ 以外に Metrowerks 社の独自の VM も利用できますが、一般には MRJ を使って実行するので、Metrowerks 社の VM を利用することは特別な場合を除いてはないと思われます。

アプレットの実行アプリケーションで、Metrowerks Java が利用されていますが、MRJ SDK に付属する Apple Applet Runner を利用することもできます。ただし、その場合はデバッガは利用できなくなります。「編集」メニューの「JavaApplet の設定」を選び、ダイアログボックスの左側で「Java ターゲット」の項目を選択すると、アプレットを実行するアプリケーションの設定が行えます。

3.2　アプリケーションの開発

CodeWarrior を使って、Mac OS 向けのアプリケーションを作成してみましょう。そのとき、Open イベントなどに対応するなど MRJToolkit の機能を使ったアプリケーションの作成方法を説明します。第 2 章で紹介したソースプログラムを実行させてみます。ここでは、CodeWarrior Release4 日本語版を使った場合の方法を説明します。

●プロジェクトの用意

アプリケーションを作る場合も、やはり最初はプロジェクトを用意します。「ファイル」メニューの「新規プロジェクト」を選び、Java の分類にある「Java Application」を選択して、ひな形のプロジェクトを作成します。

Java Application を選択する

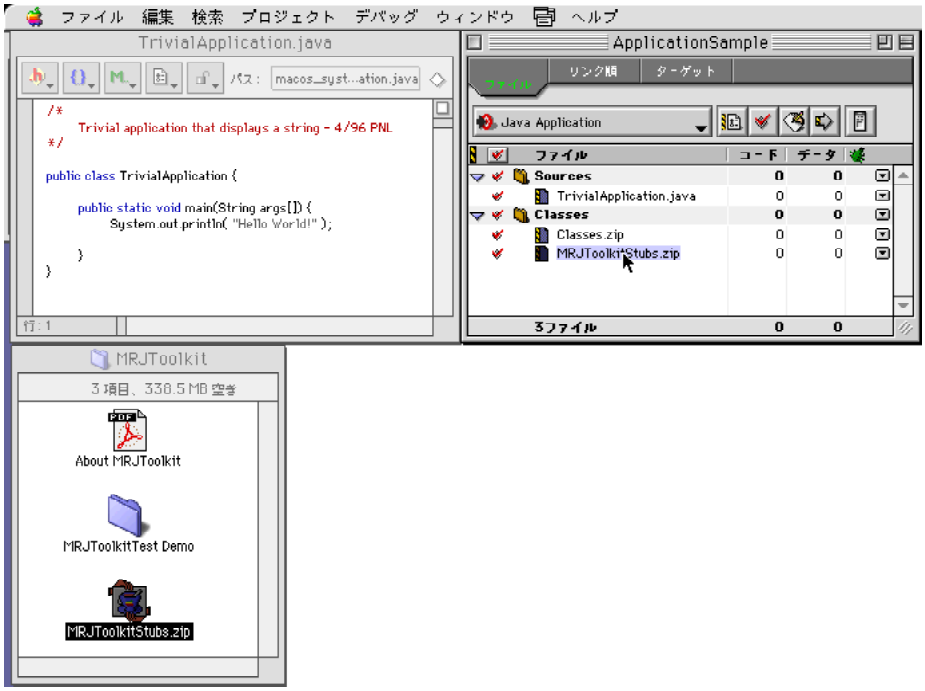
アプリケーションの場合は、TrivialApplication.java というアプリケーションのひな形になるファイルが作成されます。プロジェクトにはそのファイルと、Java の基本ライブラリを使うための Classes.zip ファイルが登録されています。

Java Application によって作られたプロジェクト

●アプリケーション化に必要な設定

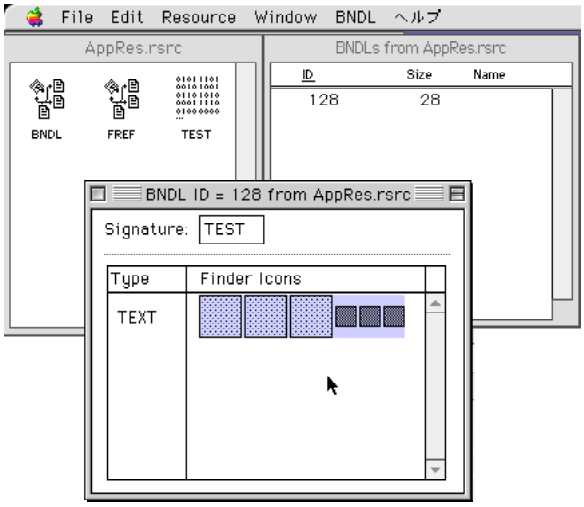
MRJToolkit の機能を Java のアプリケーションで使いたい場合には、MRJ SDK の MRJToolkit フォルダにある MRJToolkitStubs.zip というファイルをプロジェクトに登録しておきます。Finder からプロジェクトのウインドウにドラッグ&ドロップすると登

録されます。



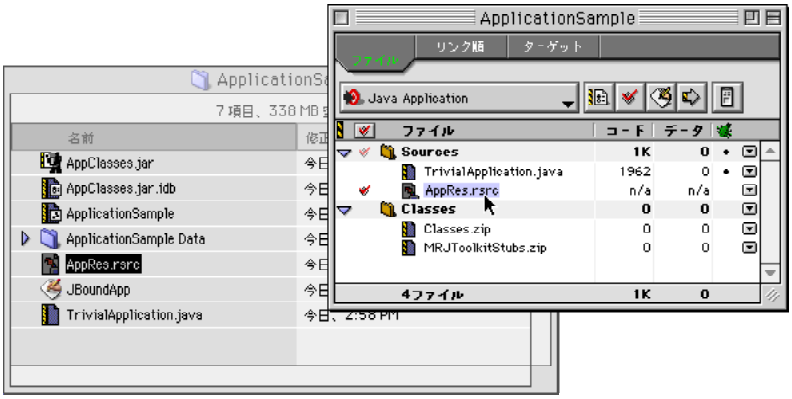
MRJToolkitStubs.zip をプロジェクトに登録する

次に最低限必要な BNDL リソースを作成しておきます。ここでは作成するアプリケーションのクリエイターは TEST とし、ドラッグ&ドロップ可能なファイルタイプを TEXT とします。ResEdit で新しくリソースファイルを作成します(ここでは AppRes.rsrc というファイル名にしました)。そして、BNDL リソースを新しく定義し、Signature には TEST を指定します。そして、新たに Type を追加して、TEXT と指定します。アイコンは適当に設定しておくと良いでしょう。アプリケーションのアイコンが必要なら、さらに APPL という Type を加えておき、アイコンをデザインします。



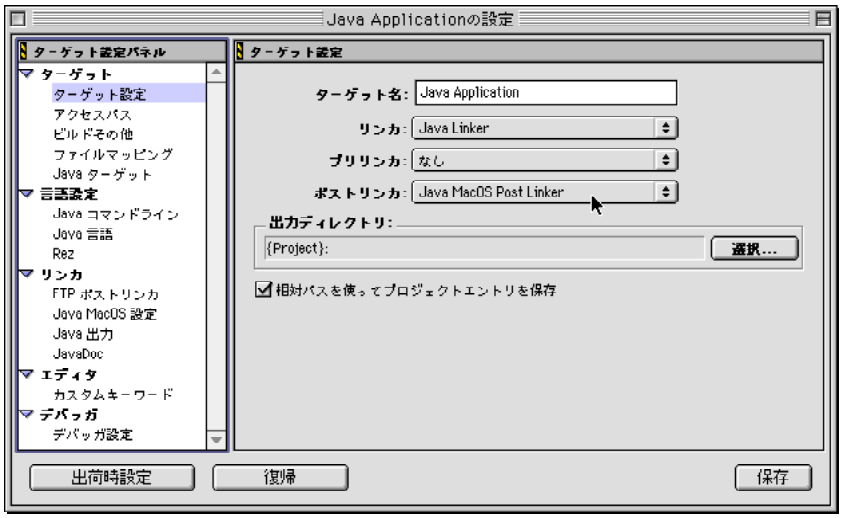
BNDL リソースを新たに用意する

こうしてリソースファイルを用意して、そのリソースファイルをプロジェクトにドラッグ&ドロップして、ファイルを追加しておきます。



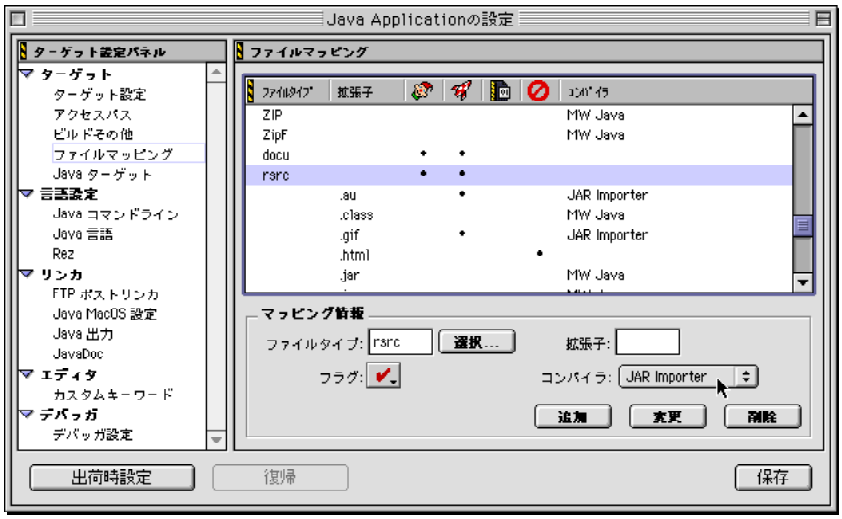
リソースファイルをプロジェクトに追加する

次に、「編集」メニューから「Java Application の設定」を選択して、プロジェクトの設定を変更します。まず、左側で「ターゲット設定」を選択します。そして、ポストリンカとして「Java Mac OS Post Linker」を選択します。このポストリンカは、MRJ SDK の JBindery と同様な働きをするものと考えればよいでしょう。コンパイルした結果から、Mac OS 上でダブルクリックして起動できるアプリケーションを作成します。



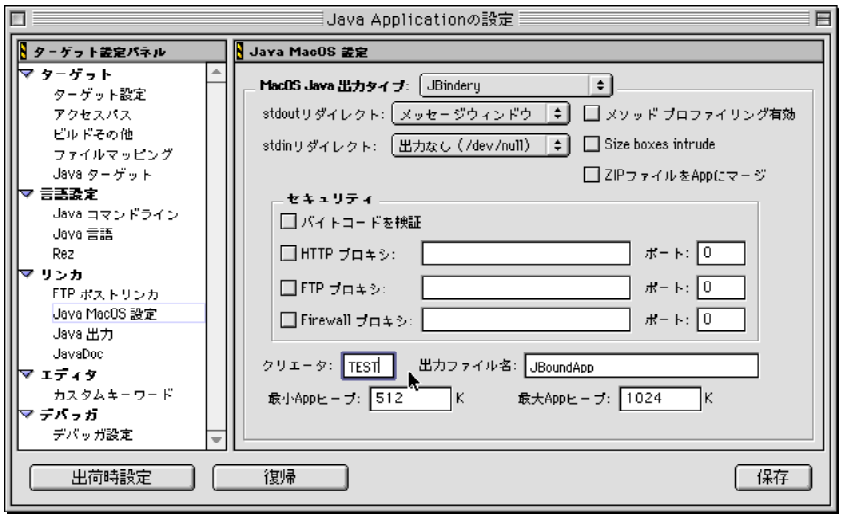
ＭacOS向けファイルを作るポストリンカを利用する

続けて、「Java Application の設定」ウインドウの左側で「ファイルマッピング」を選択します。「ファイルタイプ」として rsrc（つまり ResEdit で作成したファイル）が最初からあるので、その項目を選択します。そして「コンパイラ」のポップアップメニューから「JAR Importer」を選択し、「変更」ボタンをクリックしておきます。これにより、リソースファイルが最終的なアプリケーションに取り込むことが可能になります。



リソースファイルを取り込むように指定する

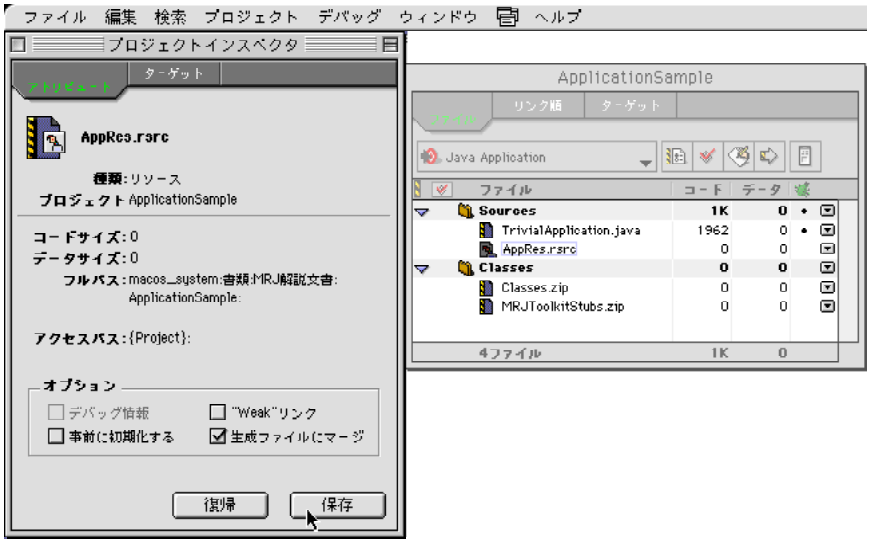
さらに、「Java Application の設定」ウインドウの左側で「Java Mac OS 設定」を選択します。ここでは必要応じて設定しますが、最低限、「クリエータ」と「出力ファイル名」の設定は必要です。クリエータとしては、ここでは TEST を指定しました。出力ファイル名は、既定値のままでもかまいません。



作成するアプリケーションファイルの設定を行う

以上の設定を行うと、「保存」ボタンをクリックして、設定を保存しておきます。

さらに、プロジェクトのウインドウに戻り、登録しているリソースファイルを選択して、「ウインドウ」メニューから「プロジェクトインスペクタ」を選択します。そのウインドウにある「生成ファイルにマージ」のチェックボックスを入れて、「保存」ボタンをクリックします。



リソースがマージされるように設定する

以上のように、Java のソースを組む以外にいろいろな設定が必要になります。ここまでに紹介した内容は最低限のことなので、必要に応じて追加のリソースを定義することなどを行うことになるでしょう。

●ソースの作成

Java のソースファイルは必要に応じて作れば良いでしょう。最初は TrivialApplication.java だけになっていますが、別の名前を付けたいのであれば、その

名前のソースファイルを用意すれば済みます。アプリケーションなので、どれかのクラスに static void main メソッドがあるはずです。それがどのクラスに存在するかは、「編集」メニューから「Java Application の設定」を選択して表示されるウインドウの左側で「Java ターゲット」を選択したときに設定できる「Main クラス」で指定を行います。つまり、TrivialApplication ではなく別の名前のクラスを用意したときには、この「Main クラス」の設定を書き換えてやらないといけません。

●コンパイルとテスト

実行は、「プロジェクト」メニューの「実行」あるいは「デバッグ」（Command+R）によって行うことができます。アプリケーションも基本的には、Metrowerks Java というアプリケーションによって起動されます。その場合には、デバッグも行えます。

ポストリンカで、Java Mac OS Post Linker を指定した状態でも実行すれば、デバッガは利用できます。このときも Metroweks Java アプリケーション内で起動します。通常の処理はデバッグできますが、イベントの発生処理をデバッグできません。生成されたアプリケーション自体を Finder から起動した場合はデバッガは利用できません。

以上の設定をしていれば、作成したアプリケーションにテキストファイルをドラッグ&ドロップすることができるはずです。すると、アプリケーションに Open イベントが発生し、この場合は、コンソールにドラッグしてきたファイルのパスが表示されるはずです。ドラッグ&ドロップができない場合には、デスクトップの再構築などをしてみて下さい。

第 4 章　MRJ についてのまとめ

- MRJ 2.1 は、Mac OS に Java 実行環境を構築します。
- JDK 1.1.7 に対応した、標準の Java 実行環境を提供します。AWT などの標準の Java ライブラリ機能は、もちろん、MRJ に組み込まれています。
- Web ブラウザで機能するアプレットは、MRJ をベースに機能します。あるいはその方向性にあります。
- Mac OS 向けのアプリケーション開発を Java で行うことは現実的になってきています。
- MRJ は独自の機能も提供しており、それを利用することで、Mac OS で機能するアプリケーションに必要な処理も組み込むことができます。
- Swing 対応ソフトウエアの実行やあるいは開発にも MRJ は対応しています。
- MRJ SDK として、開発者向けの素材も提供します。
- 市販の開発ツールを用いれば、ソースコードデバッグが可能になるなど、より効率的に Java でのソフトウエア開発が行えます。

修正履歴

- 第 2 版(1999/3/22)　p24「コンパイラとテスト」の 2 つ目の段落を書き直しました。Java Mac OS Post Linker を使っていてもデバッグは行えます。初版ではできないと記載していました。訂正の上、おわびします。

アップル コンピュータ株式会社
東京都新宿区西新宿3丁目20番2号
東京オペラシティ タワー 〒163-1480