

**i モード対応 Java API 向け Java™ 2
Micro Edition Wireless SDK
ユーザーズガイド**

本製品または文書は著作権法により保護されており、その使用、複製、再頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。サン（その他に許諾者がある場合は当該許諾者も含めて）の書面による事前の許可なく、本製品および関連する文書のいかなる部分も、いかなる方法によっても複製することが禁じられます。本製品にはRSA Security からライセンスを受けたコードが含まれます。フォントを含む第三者のソフトウェアは、著作権法により保護されており、その提供者からライセンスを受けているものです。

Sun、Sun Microsystems、サンのロゴマーク、Java、J2SE、JVM、J2ME および JDK は、米国およびその他の国における米国 Sun Microsystems, Inc. の商標または登録商標です。

i モードは NTT ドコモの登録商標です。

その他、掲載されている会社名、製品名、サービス名は各社の商標または登録商標です。

Federal Acquisitions: Commercial Software--Government Users subject to standard license terms and conditions.

すべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標の付いた製品は、米国 Sun Microsystems, Inc. が開発したアーキテクチャに基づくものです。

文書は現状有姿で提供されており、その商品性、特定目的への適合性、第三者の権利の非侵害を含み、明示的または黙示のいかなる条件、表明および保証も行われず、また提供されません。但し、かかる保証の否定が、適用される法律の下で無効である場合は、この限りではありません。

Copyright 2000 NTTDoCoMo, Inc. 2-11-1 Nagata-cho, Chiyoda-ku, Tokyo, Japan and Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California, 94303, U.S.A. All rights reserved.

目 次

第 1 章 はじめに	5
1.1 J2ME Wireless SDK の概要	5
1.1.1 J2ME Wireless SDK のコンポーネント	7
1.2 J2ME Wireless SDK の開発環境	8
1.2.1 KToolBar 開発環境	8
1.2.2 Forte for Java 開発環境	9
1.3 エミュレータの概要	9
1.3.1 エミュレータの利点	9
1.3.2 エミュレータの設定	9
1.3.3 エミュレーションの範囲	10
1.4 J2ME Wireless SDK によるアプリケーションの開発	11
第 2 章 J2ME WIRELESS SDK のインストール	15
2.1 システム要件	15
2.2 J2ME Wireless SDK のインストール	16
2.2.1 J2ME Wireless SDK ディレクトリ	17
第 3 章 KTOOLBAR の使用	19
3.1 KToolBar の実行	19
3.2 エミュレータの設定	21
3.2.1 プロキシサーバの設定	24
3.2.2 PRNG アルゴリズムの選択	24
3.2.3 ログメッセージのトレース	25
3.3 KToolBar プロジェクト	29
3.3.1 KToolBar プロジェクトウィンドウ	29
3.3.2 既存プロジェクトのオープン	29
3.3.3 アプリケーション・ディスクリプタ・ファイルの作成/変更	30
3.3.4 新規プロジェクトの作成	34
3.3.5 プロジェクトのビルドと実行	35
3.4 サンプルプロジェクト	35
第 4 章 FORTE FOR JAVA COMMUNITY EDITION の使用	37
4.1 はじめに	37
4.1.1 Forte for Java	37
4.2 サンプルアプリケーションのビルドと実行	38
4.2.1 Forte for Java の起動	38
4.2.2 アプリケーションのビルドと実行	39
4.3 新規アプリケーションの作成	42
4.3.1 プロジェクトの作成	43
4.3.2 ファイルシステムの設定	44
4.3.3 アプリケーション・ディスクリプタ・ファイル(ADF)の作成	47
4.3.4 ソースファイルの作成	50
4.3.5 ADF の設定	53
4.3.6 プロジェクト jar ファイルの内容を設定	57
4.3.7 アプリケーションのビルドと実行	59
4.4 エミュレータの設定	61
4.4.1 プロキシサーバの設定	64
4.4.2 PRNG アルゴリズムの選択	65

4.4.3 ログメッセージのトレース	65
第 5 章 コマンド行からの操作	71
5.1 コンパイラの操作	71
5.2 バイトコード事前検証プログラムの実行	72
5.3 アプリケーション JAR ファイルの作成	72
5.4 アプリケーション・ディスクリプタ・ファイルの作成	72
5.5 エミュレータの操作	73
5.5.1 エミュレータの設定	73
5.5.2 エミュレータの起動	74
第 6 章 J2ME WIRELESS SDK ユーティリティ	75
6.1 証明書管理ツール	75
6.1.1 暗号化された証明書	75
6.1.2 CMT の起動	77
6.1.3 CMT メインウィンドウ	79
6.2 ScratchPad ビューア	82
6.2.1 ScratchPad ファイル	82
6.2.2 SPV の起動	83
6.2.3 Scratchpad 内容の表示	87
6.2.4 別の ScratchPad ファイルの表示	88
6.2.5 SPV の終了	88
6.2.6 アプリケーション ScratchPad ファイルの削除	89
6.3 SJIS エンコーダ	91

第 1 章

はじめに

このマニュアルでは、J2ME Wireless SDK の各コンポーネントをインストール、設定、および使用するための方法について説明しています。（SDK：Software Development Kit）

このマニュアルで述べる J2ME Wireless SDK は、Java 対応 i モード携帯電話で動作する Java アプリケーションの開発をサポートします。

このマニュアルを理解するには、Java プログラミング、CLDC、および NTT ドコモが提供する Java API の知識が必要です。

この章では、J2ME Wireless SDK の各コンポーネントについて概要を説明します。この章の内容は次のとおりです。

- J2ME Wireless SDK の概要
- J2ME Wireless SDK 開発環境
- エミュレータの概要
- J2ME Wireless SDK によるアプリケーションの開発

注 - 本書ならびにエミュレータ内部において、i モード対応 Java のことを“DoJa”として記載している場合があります。

注 - 本書内に記載されている URL は予告なく変更される場合があります。

1.1 J2ME Wireless SDK の概要

J2ME Wireless SDK では、バイトコード事前検証プログラムやエミュレータといった基本的なソフトウェア開発ツールの他に、これらのツールが一体となって動作する 2 つの開発環境が提供されます。これらの開発環境は、インストールすればすぐに使用できます。どちらでも好きな方を使用してください。

J2ME Wireless SDK

- KVM 用アプリケーションの開発の開始から配布可能なファイルを用意するまでをサポートします。（Java ソースファイルから最終的な jar ファイル及びアプリケーション・ディスクリプタ・ファイルの作成まで）

- ユーザーは開発ツールの使用方法として、GUI ベースの 2 つの開発環境からどちらかを選択するか、コマンド行による操作を選択できます。
- 複数の異なる対象デバイスで実行されるアプリケーションのエミュレーションを行うための強力なエミュレーター機能を提供します。

1.1.1 J2ME Wireless SDK のコンポーネント

J2ME Wireless SDK では、3 つのグループに分けられた次のコンポーネントが提供されます。

- 基本開発ツール
- ユーティリティ
- 開発環境のコンポーネント

表 1-1 J2ME Wireless SDK のコンポーネント

コンポーネント	説明
基本開発ツール:	
バイトコード事前検証プログラム	パッケージングの前にバイトコードを事前に検証します。
エミュレータ	ターゲットデバイスをエミュレートしてアプリケーションを実行します。
ユーティリティ:	
ScratchPad ビューア	アプリケーションの ScratchPad ファイルを 16 進数で表示します。
証明書管理ツール	安全な通信のために使用される HTTPS 証明書を管理します。
SJIS エンコーダ	i モード対応 Shift-JIS 文字エンコーダ
開発環境のコンポーネント:	
KToolBar	最小限の機能を備えた使い易い GUI ベースの開発環境
J2ME Wireless Module	開発ツールを Forte for Java 開発環境のもとで実行するためのプラグインモジュール

1.2 J2ME Wireless SDK の開発環境

開発サイクルを GUI ベースの統合環境で行う方法としては、KToolBar 開発環境と Forte for Java 開発環境があります。さらに、ツールをコマンド行から実行することもできます。

J2ME Wireless SDK の開発環境には、次の利点があります。

- 開発サイクルを統合環境で進めることができます。
- 2 つの開発環境のいずれかによる操作か、コマンド行による操作を選択できます。

1.2.1 KToolBar 開発環境

KToolBar 開発環境の画面は、図 1-1 のように表示されます。

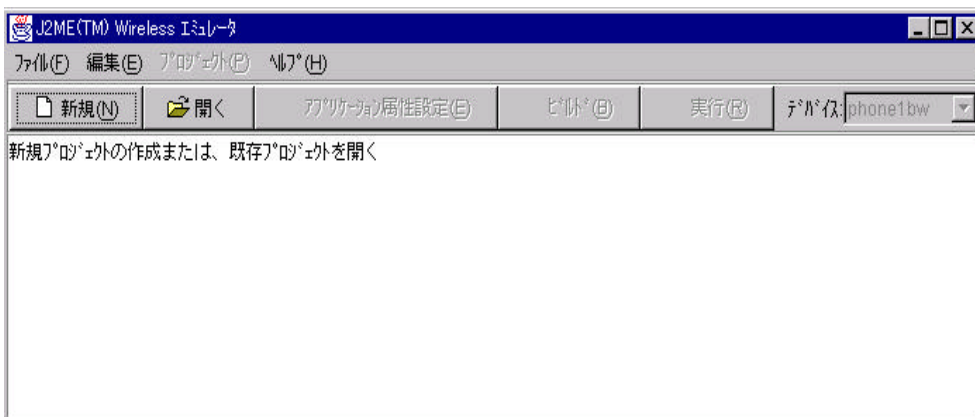


図 1-1 KToolBar 画面

KToolBar は、次の機能を備えた最小限の開発環境です。

- ビルド
 - コンパイル
 - バイトコードの検証
 - パッケージング
- アプリケーション・ディスクリプタ・ファイルの作成
- テスト(エミュレーション)

アプリケーションファイルは、エミュレータでのテストが完了したら、Web サイトにアップロードして、実際のデバイスでテストします。

開発サイクルの詳細については、本書の「1.4 J2ME Wireless SDK によるアプリケーションの開発」または『i モード対応 Java コンテンツ開発ガイド ～詳細編～』の「第9章 アプリケーションのビルド、テスト、およびパッケージ化」を参照してください。

1.2.2 Forte for Java 開発環境

Forte for Java は、Java アプリケーション向けの包括的な統合開発環境(IDE)です(Forte for Java の詳細については、<http://www.sun.com/forte/ffj/index.html> を参照して下さい)。J2ME Wireless SDK では、Forte for Java 開発環境のもとでアプリケーション開発サイクルを行うために必要なオプションモジュールが提供されます。Forte for Java 環境では、Java アプリケーションの編集、コンパイル、ビルド、およびエミュレータによる実行をすべて行うことができます。

1.3 エミュレータの概要

エミュレータコンポーネントは、ターゲットデバイスの多様なエミュレーションを PC 上で行うことによってアプリケーション開発をサポートします。エミュレータを使用すれば、ターゲットデバイスで行われるアプリケーションの動作を PC で検証できます。

1.3.1 エミュレータの利点

エミュレータの利点には、次のものがあります。

- 開発に使用したのと同じ PC プラットフォームでアプリケーションをテストできます。
- 実際のターゲットデバイスは、最終的なテストのときしか必要ありません。
- 多様なターゲットデバイスでのアプリケーションの実行を同じ PC からエミュレートできます。
- 実行時に、メソッドのトレース、ガベージコレクション、クラスのロード、スローされた例外など、さまざまなイベントのロギングを行うことができ、開発者はアプリケーションの実行をトレースできます。
- この開発環境には、新しいデバイスプロパティファイルや、特定のデバイス向けに新しく開発されたデバイスエミュレータなど、新しいデバイスを使用する上での柔軟性が備わっています。

ソフトウェア開発サイクルにおけるエミュレータの位置付けについては、「1.4 J2ME Wireless SDK によるアプリケーションの開発」を参照してください。

1.3.2 エミュレータの設定

エミュレータは、次のユーザーインターフェースを使って設定できます。

- コマンド行(詳細は、「第5章 コマンド行からの操作」を参照)
- KToolBar(詳細は、「第3章 KToolBar の使用」を参照)
- Forte for Java(詳細は、「第4章 Forte for Java Community Edition の使用」を参照)

1.3.3 エミュレーションの範囲

エミュレータをテストの目的で使用するには、エミュレータがエミュレーションできる範囲を理解しておくことが大切です。

- アプリケーションマネージャの処理

エミュレータは Java アプリケーションの実行環境を提供していますが、実際の電話機上で実行されるアプリケーションマネージャの動作をエミュレートする環境は提供していません。例えばアプリケーションの更新はエミュレータが実行しない処理の一つです。これは、エミュレータがアプリケーションを保存しないためです。

- アプリケーション・ディスクリプタ・ファイル

エミュレータでは、アプリケーション・ディスクリプタ・ファイルにおける次のキーが使用されます。

AppName
AppClass
AppSize (ビルド時に、エミュレータにより自動的に設定されます)
AppParam
KvmVer
SPsize
UseNetwork

エミュレータはこれらのキーの値を読み込んで、実際の Java 対応 i モード携帯電話と同様の動作を行います。その他のキーは処理されません。エミュレータは上記以外のキーが指定されていても、値を検査したり、使用することはありません。

AppVer

AppVer は、アプリケーション開発者が開発中のプログラムのバージョン管理を行えるようにするために設けられているものです。i モード Java アプリケーション環境では AppVer の値は使用しません。

LastModified

エミュレータは、JAM のアプリケーション更新機能をサポートしていません。従ってエミュレータでは、LastModified の値は検査されません。ただし、エミュレータのビルド機能はこのキーの自動更新をサポートしています。

PackageURL

エミュレータでは、アプリケーションの JAR ファイルは開発コンピュータのハードディスクからロードされます。また、HTTP 通信を行う際の接続先サーバーの制約の実現には、エミュレータ設定の「読み込み先 URL」の値を使用します。このため、PackageURL の値をエミュレータが使用することはありません。

(このキーに値を設定していても問題はありませんので、エミュレータの生成したアプリケーションディスクリプタファイルを実環境でも流用できるよう、本キーには正しい URL を指定しておくことをお勧めします。)

LaunchAt

実際の携帯電話でサポートされる起動時間間隔指定機能はエミュレータではサポートされないため、エミュレータが LaunchAt キーを使用することはありません。

TargetDevice

TargetDevice キーは実際の携帯電話上でアプリケーションの互換性をチェックするためのものであり、エミュレータでは検査されません。アプリケーションにメーカー拡張機能を使用している場合は、アプリケーションをそのメーカーの適切なエミュレータおよび携帯電話でテストする必要があります。

1.4 J2ME Wireless SDK によるアプリケーションの開発

この節では、上述の KToolBar と Forte for Java 開発環境でサポートされる開発サイクルについて説明します。ただし、この他にも、開発サイクルをコマンド行から行うことができます。

J2ME Wireless SDK でサポートされる開発サイクルには、次のフェーズがあります。

- テキストエディタを使って java ファイルを編集する(J2ME Wireless SDK の範囲には含まれません)。

- ビルド

- コンパイル

Java 2 Standard Edition SDK コンパイラを使って、ソースファイルをクラスファイルにコンパイルします。コンパイラは、CLDC や DoJa API に含まれていないパッケージやクラス、メソッドがアプリケーションで使用されていないか検証します。

- バイトコードの検証

事前検証ツールを使って、Java クラスの事前検証を行うとともに、Java クラスの中で浮動小数点演算やファイナライズメソッドが使用されていないか検査します。

- パッケージング

Java 2 Standard Edition SDK の jar ツールを使って、事前検証されたすべてのクラスファイルとアプリケーションリソースファイルを Jar ファイルとしてパッケージングします。

- アプリケーション・ディスクリプタ・ファイル

アプリケーション・ディスクリプタ・ファイルを作成します。これは、アプリケーション情報を記述するテキストファイルです。

- テスト(エミュレーション)

異なるターゲットデバイスに対しエミュレータを使ってアプリケーションを実行します。

同じアプリケーションを 2 つ以上のデバイスで同時に実行し、それぞれを比較することもできます(つまり、J2ME Wireless SDK では複数のエミュレータを同時に実行することが可能です)。

あるアプリケーションに対してエミュレータを最初に実行したときには、そのアプリケーションからスクラッチパッドファイルに対して、読み書きすることができます。しかし、その後で同じアプリケーションに対してさらにエミュレータを起動し、同時に実行した場合には、そのアプリケーションはスクラッチパッドを読み取り専用としてしか使用できません。(これは、スクラッチパッドの排他的制御をエミュレーターが行うための制約事項です)

エミュレータでのテストが完了したら、アプリケーションファイルを Web サイトにアップロードし、実際のデバイスでテストすることができます。

上で述べた開発サイクルを図示すると、図 1-2 のようになります。

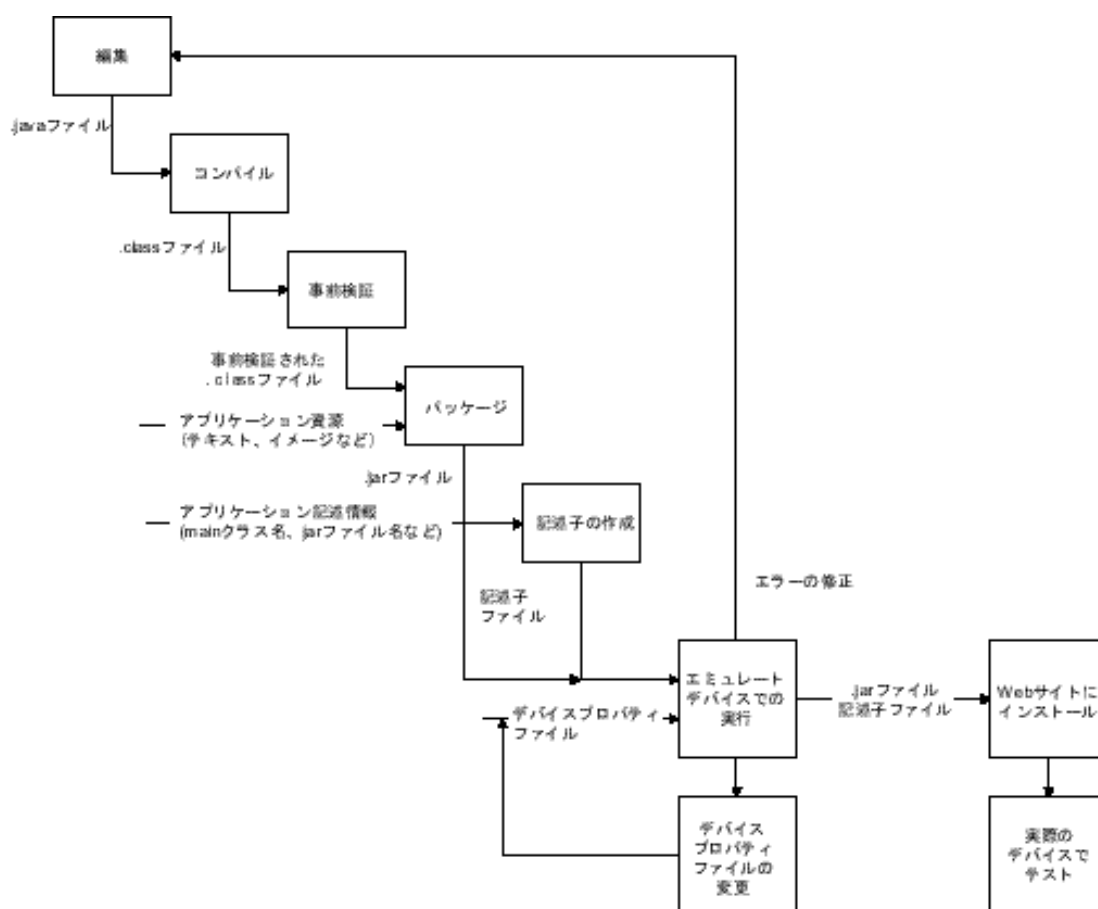


図 1-2 アプリケーション開発サイクル

第 2 章

J2ME Wireless SDK のインストール

2.1 システム要件

表 2-1 に、必須および任意のシステム要件を示します。

表 2-1 システム要件

必須のシステムコンポーネント	要件
オペレーティングシステム	Microsoft Windows 98 Second Edition
メモリ	128 MB 以上推奨
ハードディスク容量	10 MB
Java 環境	Java 2 Standard Edition (J2SE), Version 1.3 -- 次のサイトからダウンロードできます。 http://java.sun.com/products/j2se/1.3/download-windows.html
任意のシステムコンポーネント	要件
Forte For Java Community Edition	Forte For Java Community Edition 1.0 Update release 1 ¹ . 以下のサイトから日本語版がダウンロードできます。 http://www.sun.com/forte/ffj/ 追加の構成要件: Intel Pentium II プロセッサ 300MHz CPU、40M バイトのディスク容量

1. Forte For Java Community Edition 1.0 Update release 2 以降は正式サポートの対象外となります。

2.2 J2ME Wireless SDK のインストール

J2ME Wireless SDK をインストールする前に、J2ME Wireless SDK の以前のバージョンがすでにインストールされている場合は、これをアンインストールしてください。J2ME Wireless SDK をアンインストールするには、「スタート」->「プログラム」->「J2ME Wireless SDK for DoJa 1.0」メニューから「Remove Java™2 Micro Edition, Wireless SDK for DoJa APIs」メニュー項目を選択します(図 2-1)。



図 2-1 J2ME Wireless SDK のアンインストール

J2ME Wireless SDK をインストールするには

1. インストーラアイコンをダブルクリックして、J2MEWSDK4DOJA_RELEASE2.exe を実行します。
2. インストーラプログラムのプロンプトに従ってください。

注 - J2ME Wireless SDK または Forte for Java Community Edition をインストールする 際に、インストールディレクトリの名前にスペースを含めないようにして下さい。

事前検証を実行する際に名前にスペースが含まれているとディレクトリを正しく扱えません。

2.2.1 J2ME Wireless SDK ディレクトリ

本書では、J2ME Wireless SDK インストールディレクトリのトップレベルを「J2ME Wireless SDK_DIRECTORY」として解説します。

表 2-2 J2ME Wireless SDK ディレクトリの内容

ファイルまたはディレクトリ	説明
J2ME Wireless SDK_DIRECTORY \	J2ME Wireless SDK のインストールディレクトリ
J2ME Wireless SDK_DIRECTORY \License	J2ME Wireless SDK のライセンス契約書
J2ME Wireless SDK_DIRECTORY \apps	サンプルアプリケーションが入っています。さらに、KToolBarを使って作成するアプリケーションもここに追加されます。
J2ME Wireless SDK_DIRECTORY \bin	J2ME Wireless SDK の操作で使用するバッチファイルと.exe ファイルが入っています。
J2ME Wireless SDK_DIRECTORY \lib \dojaapi.jar	CLDC と DoJa API のクラスファイルが入っています。これらのファイルは、アプリケーションソースファイルのコンパイルやアプリケーションクラスのバイトコード事前検証で使用されます。
J2ME Wireless SDK_DIRECTORY \lib \dojaconv.jar	iモード対応のShift-JIS文字エンコーダのクラスファイルが入っています(詳細は、「6.3 SJIS エンコーダ」を参照)。これらのクラスファイルは、アプリケーションJavaソースファイルのコンパイルで使用されます。
J2ME Wireless SDK_DIRECTORY \lib *.zip	エミュレータとKToolBarのクラスファイルが入っています。
J2ME Wireless SDK_DIRECTORY \lib \ssl	次のファイルが入っています。 • Java Secure Socket Extensions (JSSE) v1.0.1に関連するファイル。J2ME Wireless SDKでSSL(Secure Socket Layer)をサポートするために必要です。 • クライアント証明書とサーバ証明書。詳細は、「6.1 証明書管理ツール」を参照してください。
J2ME Wireless SDK_DIRECTORY \lib \devices	エミュレータによってエミュレートされるいろいろなデバイスのデバイスプロパティファイルが入っています。デバイスプロパティファイルは、グループごとに別個のディレクトリに入っています。
J2ME Wireless SDK_DIRECTORY \tools	次のファイルが入っています。 • 開発プロセスを支援するユーティリティ • バンキングデモとスペースインベーダのサンプルアプリケーションのJavaサーブレットコード

第 3 章 KToolBar の使用

3.1 KToolBar の実行

J2ME Wireless SDK をインストールしたら、「スタート」->「プログラム」->「Java™ 2 Platform, Micro Edition, Wireless SDK for DoJa APIs」->「KToolBar」の順にメニュー項目を選択して KToolBar 開発環境を起動します。



図 3-1 J2ME Wireless SDK の起動

図 3-2 に示す最初の画面が表示され、「ファイル」、「編集」、「ヘルプ」メニューだけがアクティブになります。「プロジェクト」メニューは、プロジェクトがオープンされるとアクティブになります。

この画面では、次の 3 つの選択肢があります。

- 新しいプロジェクトを作成する。「3.3.4 新規プロジェクトの作成」を参照してください。
- 既存のプロジェクトをオープンする。「3.3.2 既存プロジェクトのオープン」を参照してください。
- エミュレータを設定する。「3.2 エミュレータの設定」を参照してください。



図 3-2 KToolBar の開始画面

3.2 エミュレータの設定

エミュレータをまだ設定していない場合は、「編集」->「設定」コマンドを使ってエミュレータを設定します(図 3-3)。これらの設定値はすべてのアプリケーションに適用されます。

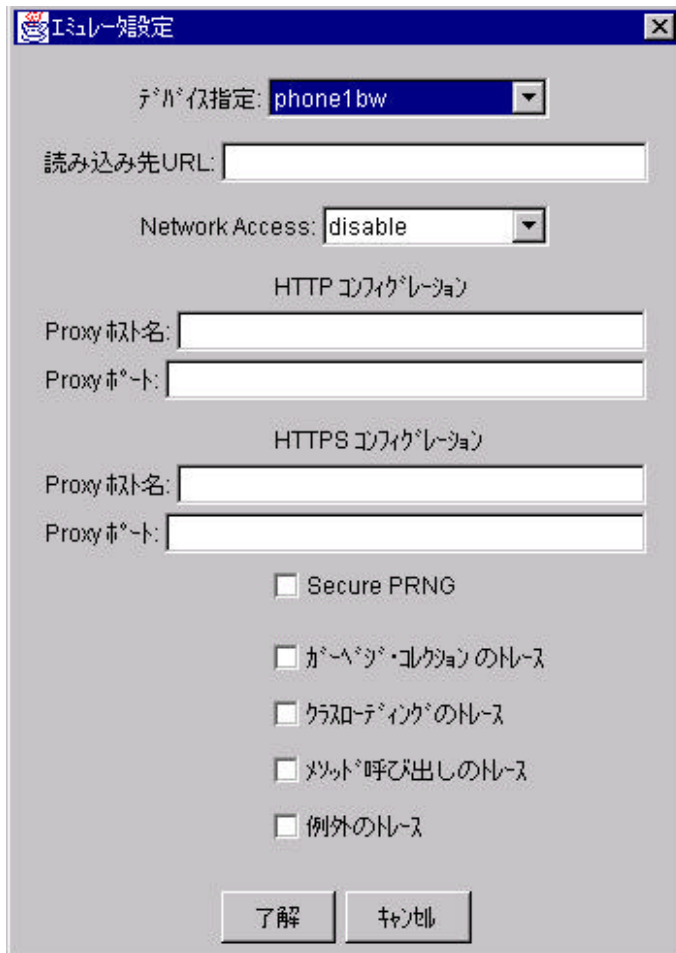


図 3-3 KToolBar の設定

設定項目とその意味は表 3-1 のとおりです。

表 3-1 設定項目

設定項目	説明	デフォルト
デバイス指定	ターゲットデバイスの名前。デバイスプロパティファイルの名前と同じです。	phone1bw
読み込み先 URL	エミュレータは、アプリケーションがこの URL からダウンロードされたものとみなします。アプリケーションは、このフィールドに指定されたホストとポートだけとしか通信できません。このフィールドの形式は、アプリケーション・ディスクリプタ・ファイルの PackageURL タグの形式と同じです。このフィールドが「*」なら、アプリケーションは、どのサーバとでも通信できます。この設定はエミュレータ上で Java アプリケーションの開発とテストを行う場合にのみ使用できます。このフィールドがブランクなら、アプリケーションは、ネットワークを全く使用できません。	なし
NetworkAccess	NetworkAccess の設定は、エミュレータが Java アプリケーションからのネットワーク接続を許可するかどうかを管理します。ネットワークアクセスの設定を有効にするには、アプリケーション・ディスクリプタ・ファイルの "UseNetwork" キーに "http" が指定されていなければならないことに注意して下さい。アプリケーション・ディスクリプタ・ファイルの "UseNetwork" キーが定義されていない場合、NetworkAccess に設定された値にかかわらずネットワーク接続を行おうとすると Java アプリケーションは終了します。 NetworkAccess の設定には、以下のような値を設定できます。 • enable - Java アプリケーションからのネットワーク接続を許可します。この接続は読み込み先 URL に設定された値に基づきコントロールされます。 • disable - Java アプリケーションからのネットワーク接続は許可されません。Java アプリケーションがネットワーク接続を行おうとした場合は SecurityException がスローされます。	disable
HTTP コンフィグレーション		
Proxy ホスト名	アプリケーションが Web サーバとの通信に使用する HTTP プロキシサーバのアドレス(「3.2.1 プロキシサーバの設定」を参照)	なし
Proxy ポート	アプリケーションが Web サーバとの通信に使用する HTTP プロキシサーバのポート	なし
HTTPS コンフィグレーション		
Proxy ホスト名	アプリケーションが Web サーバとの通信に使用する HTTPS プロキシサーバのアドレス(「3.2.1 プロキシサーバの設定」を参照)	なし
Proxy ポート	アプリケーションが Web サーバとの通信に使用する HTTPS プロキシサーバのポート	なし

	ロキシサーバのポート	
Secure PRNG	高速の代替 PRNG アルゴリズムの代わりに標準のセキュア PRNG を使用します(「3.2.2 PRNG アルゴリズムの選択」を参照)。	オフ
Tracing Log Messages		
ガベージ・コレクションのトレース	ガベージコレクション情報を表示します(「3.2.3.1 ガベージコレクションのトレース」を参照)。	オフ
クラスローディングのトレース	クラスローディング情報を表示します(「3.2.3.2 クラスローディングのトレース」を参照)。	オフ
メソッド呼び出しのトレース	メソッド呼び出しを表示します(「3.2.3.3 メソッド呼び出しのトレース」を参照)。	オフ
例外のトレース	例外を表示します(「3.2.3.4 例外のトレース」を参照)。	オフ

3.2.1 プロキシサーバの設定

エミュレータのもとで動作する Java アプリケーションからプロキシサーバ経由で Web サーバに接続する必要がある場合には(たとえば、Web サーバがファイアウォールの反対側にある場合など)、エミュレータの設定でプロキシサーバの情報を指定する必要があります。アプリケーションは HTTP か HTTPS プロトコルを使って Web サーバに接続しますが、同じアプリケーションで両方のプロトコルを使用することはできません。

HTTP と HTTPS のプロキシサーバのアドレスとポートは図 3-3 の設定画面で入力します。これらの設定値がわからない場合は、システム管理者に問い合わせして下さい。

3.2.2 PRNG アルゴリズムの選択

HTTPS プロトコルでは、非公開鍵や公開鍵の暗号化のために、安全な擬似ランダム数値ジェネレータ(PRNG)アルゴリズムが必要です。

このアルゴリズムには、次の 2 つがあります。

- 標準のセキュア PRNG アルゴリズム

J2SE に組み込まれているこのアルゴリズムは安全ではありますが、低速です。このアルゴリズムでは、Windows で J2ME Wireless SDK を起動するたびに、初期化に最大 60 秒を要することがあります。

- 高速 PRNG アルゴリズム

J2ME Wireless SDK に組み込まれている代替 PRNG アルゴリズムは、ブルートフォース攻撃(力まかせ攻撃 Brute-Force Attack)や、既知平文攻撃(Known-Plaintext Attack)、選択平文攻撃(Chosen-Plaintext Attack)、繰り返し攻撃(Replay Attack)といった攻撃に対して標準アルゴリズムほど堅牢ではありませんが、はるかに高速で、初期化は 1 秒足らずで終了します。

アプリケーションのテストでは、どちらのアルゴリズムを選択しても違いはありません。プロトコルの実際の手順はどちらも全く同じです。したがって、ネットワークを通して送信するデータが本当に機密なものである場合を除き、一般的には高速 PRNG アルゴリズムを選択します。

注 - テストするアプリケーションがネットワークを通して機密データを送信する場合は、標準のセキュア PRNG アルゴリズムを使用してください。

標準のセキュア PRNG アルゴリズムをアクティブにするには、図 3-3 の「Secure PRNG」ボックスにマークを付けます。このボックスにマークが付いていないと、高速 PRNG アルゴリズムが使用されます。

3.2.3 ログメッセージのトレース

KToolBar を設定すれば、次の 4 種類のログメッセージをトレースできます。

- ガベージコレクション
- クラスローディング
- メソッド呼び出し
- スローされた例外

図 3-3 の画面で、該当するボックスを選択してください。

これらのログメッセージは、アプリケーションの実行に従ってコンソールウィンドウに表示されます(次の各項を参照)。

3.2.3.1 ガベージコレクションのトレース

図 3-4 は、ガベージコレクションのトレース画面を表したものです。次の情報が表示されます。

- Java ヒープの割り当て
- GC スキャンが開始されたことを示す通知
- いくつのオブジェクトが割り当てられているかを示す実行時カウンタ
- 割り当てられているオブジェクトの合計サイズ

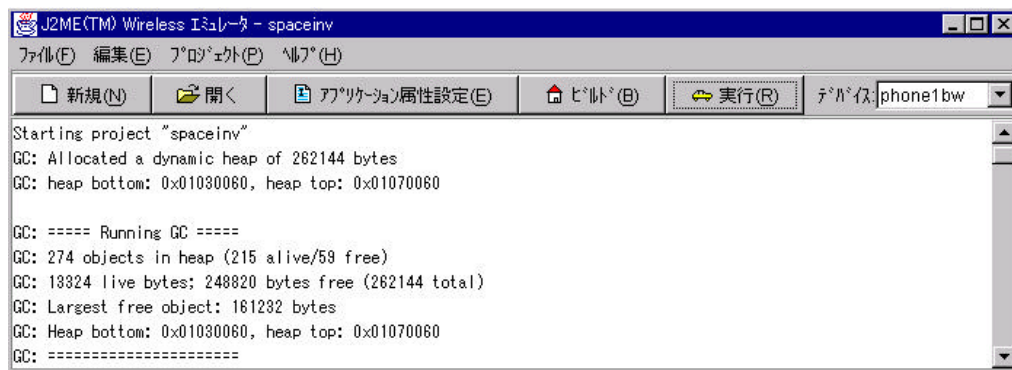


図 3-4 ガベージコレクションのトレース

3.2.3.2 クラスローディングのトレース

図 3-5 は、クラスローディングのトレース画面を表したものです。クラスがロードされるたびに、名前が表示されます。

注 - このオプションは、システムクラスには適用されません。



図 3-5 クラスローディングのトレース

3.2.3.3 メソッド呼び出しのトレース

図 3-6 は、メソッド呼び出しのトレース画面を表したものです。メソッドが呼び出されるたびに、オブジェクトとメソッドが表示されます。

注 - このオプションは非常に冗長であるため、アプリケーションの実行が遅くなる場合があります。

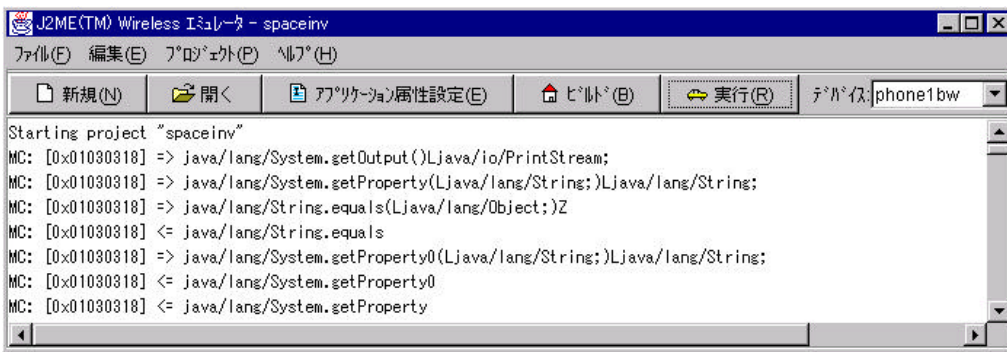


図 3-6 メソッド呼び出しのトレース

3.2.3.4 例外のトレース

図 3-7 は、例外のトレース画面を表したものです。スローされるすべての例外が表示されます(キャッチされたかどうかに関係なく表示されます)。

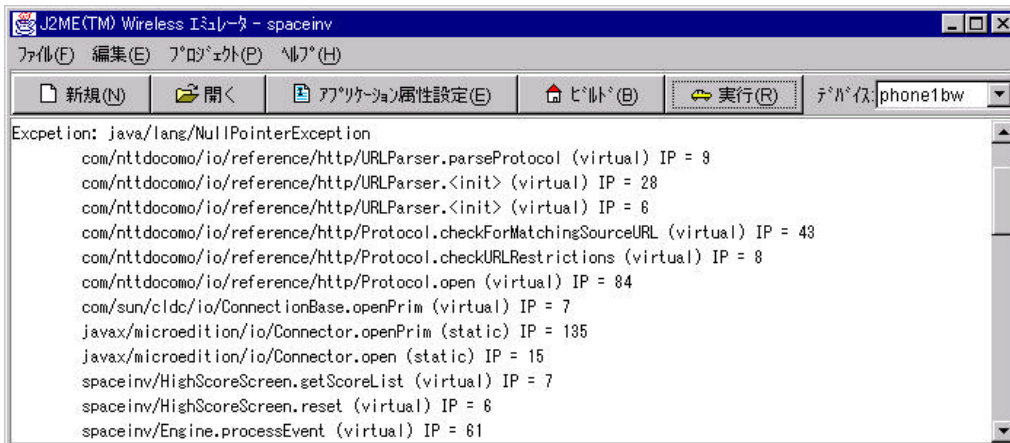


図 3-7 例外のトレース

3.3 KToolBar プロジェクト

KToolBar「プロジェクト」とは、1つのアプリケーションに属するファイルの集合です。これには、ソースコードファイルから最終的な jar ファイルやアプリケーション・ディスクリプタファイルに至るすべてのファイルの他に、プロジェクトに関するエミュレータの設定データが含まれます。プロジェクトの目的は、開発のさまざまな段階でプロジェクト名を指定するだけでアプリケーションを呼び出せるようにすることです。

KToolBar プロジェクトにおける開発作業項目には、次のものがあります。

- プロジェクトのビルド - プロジェクトのコンパイル、事前検証、パッケージ化を1つの操作で連続的に行います。
- プロジェクトの実行 - 選択するエミュレーションデバイスでプロジェクトを実行し、アプリケーションの動作をエミュレートします。

3.3.1 KToolBar プロジェクトウィンドウ

既にプロジェクトが作成されているか、オープンされていると、「プロジェクトの選択」メニューが表示され、プロジェクトの選択を行うことにより、プロジェクトが読み込まれた状態になります。KToolBar では、次のことができます。

- 既存プロジェクトのオープン
- アプリケーション・ディスクリプタ・ファイルの作成/変更
- 新規プロジェクトの作成
- プロジェクトのビルドと実行

定義されたプロジェクトは常に最新の状態で保存され、後でオープンすることができます。

3.3.2 既存プロジェクトのオープン

既存プロジェクトをオープンするには、以下の手順に従います。

1. 「ファイル」->「開く」の順でプロジェクトコマンドを選択するか、「開く」ボタンをクリックします。

既存プロジェクトのリストからプロジェクトを選択することを求めるプロンプトが表示されます(図 3-8)。

2. 必要なプロジェクト名を選択します。

「プロジェクトの選択」ウィンドウが表示され、プロジェクト画面タイトルにプロジェクト名が表示されます(図 3-8)。

これで、画面のすべてのプロジェクトメニュー項目が使用可能になります。

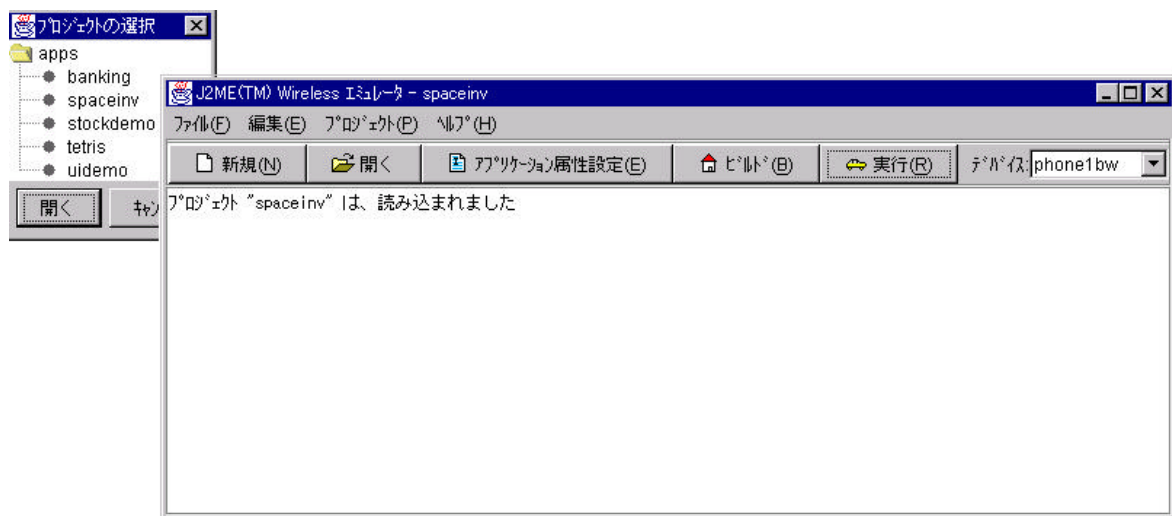


図 3-8 プロジェクトのオープン

3.3.3 アプリケーション・ディスクリプタ・ファイルの作成/変更

アプリケーション・ディスクリプタ・ファイルを編集するには、メニューから「プロジェクト」->「アプリケーション属性設定」の順でコマンドを選択するか、「アプリケーション属性設定」ボタンをクリックします。

「アプリケーション属性設定」画面が表示されます(図 3-9)。



図 3-9 アプリケーション属性設定

設定項目は、表 3-2 に示すとおりです。

表 3-2 プロジェクトの設定項目

設定項目	説明
必須フィールド:¹	
AppName	アプリケーション名(最大 16 バイト)。AppName には、SJIS 文字を使用できます。
AppClass	アプリケーションの main クラス。AppClass では、com.nttdocomo.ui.IApplication を継承しなければなりません。
AppSize	アプリケーションサイズ(バイト単位)。このフィールドは、JAR ファイルの生成時にビルドプロセスによって自動的に生成されます。ユーザーが編集することはできません。
PackageURL	アプリケーションの場所を表す URL(最大 255 バイト) 新規のプロジェクトが作成されたときのデフォルト値はアプリケーションの JAR ファイルの名前になっています。このフィールドは実際に配置される場所の URL に変更して下さい。
LastModified	日本標準時(JST)による最終更新時刻 形式: Fri, 25 Feb 2000 12:04:25 このフィールドは、JAR ファイルの生成時にビルドプロセスによって自動的に生成されます。ユーザーが編集することはできません。
任意フィールド:	
AppVer	アプリケーションのバージョン。このキーは、参考情報であり、エミュレータでは使用されません。
AppParam	Main クラスのパラメータをスペースで区切って指定します(最大 255 バイト)。Java アプリケーションは、com.nttdocomo.ui.IApplication.getArgs()メソッドを使ってこれらのパラメータを取り出します。このメソッドは、このキーが指定されていないと、null を返します。設定できる文字列は ASCII のみです。
KvmVer	KVM バージョン(たとえば、CLDC-1.0)。このキーを指定しないと、アプリケーションは CLDC Virtual Machine のすべてのバージョンで動作するものとみなされます。
SPsize	ScratchPad のサイズ(バイト単位)。エミュレータ上で指定できる ScratchPad サイズに制限はありませんが、実際の端末では機種によりサイズの制限があります。実際の端末での ScratchPad サイズを検討する際には、『i モード対応 Java コンテンツ開発ガイド～詳細編～ 第 3 章 設計上の考慮事項』を参照して下さい。なおこのキーを指定しないと、ScratchPad を使用することができません。
UseNetwork	アプリケーションでネットワーク通信を使用することを表すフラグ。このキーを指定しないと、アプリケーションからネットワーク接続を行うことはできません。有効な値は「http」だけです。
TargetDevice	ターゲットデバイスをコンマで区切って指定します。アプリケーションでメーカー固有の拡張 API を使用するとき指定します。
LaunchAt	起動時間間隔指定。アプリケーションを一定時間毎に自動起動させるときに指定します。 形式: I h(h は起動間隔(時間単位))

1. 必須キーは、有効な値をアプリケーション・ディスクリプタ・ファイルに必ず指定する必要があります。
2. 各設定項目の詳細については、『i モード対応 Java コンテンツ開発ガイド～詳細編～』の「第 9 章 アプリケーションのビルド、テスト、およびパッケージ化」も参考にして下さい。

3.3.4 新規プロジェクトの作成

新しいプロジェクトを作成するには

1. 「ファイル」->「新規作成」の順でコマンドを選択するか、「新規」ボタンをクリックして、新しいプロジェクトをオープンします。

プロジェクト名入力プロンプトが表示されます(図 3-10)。

2. 新しいプロジェクト名を入力します。

プロジェクト画面が表示され、画面のタイトルにプロジェクト名が表示されます(図 3-10)。

プロジェクトの Java ソースファイルとアプリケーションリソースファイルをどこに移動すべきかを示すメッセージがコンソールに次の形式で表示されます。

例 J2ME Wireless SDK を C:\J2MEWSDK4DOJA にインストールした場合

Java ソースコードの場所 "C:\J2MEWSDK4DOJA\apps\<プロジェクト名>\src"

アプリケーションリソースファイルの場所 "C:\J2MEWSDK4DOJA\apps\<プロジェクト名>\res"

3. Java ソースファイルとアプリケーションリソースファイルをそれぞれのディレクトリに移動します。

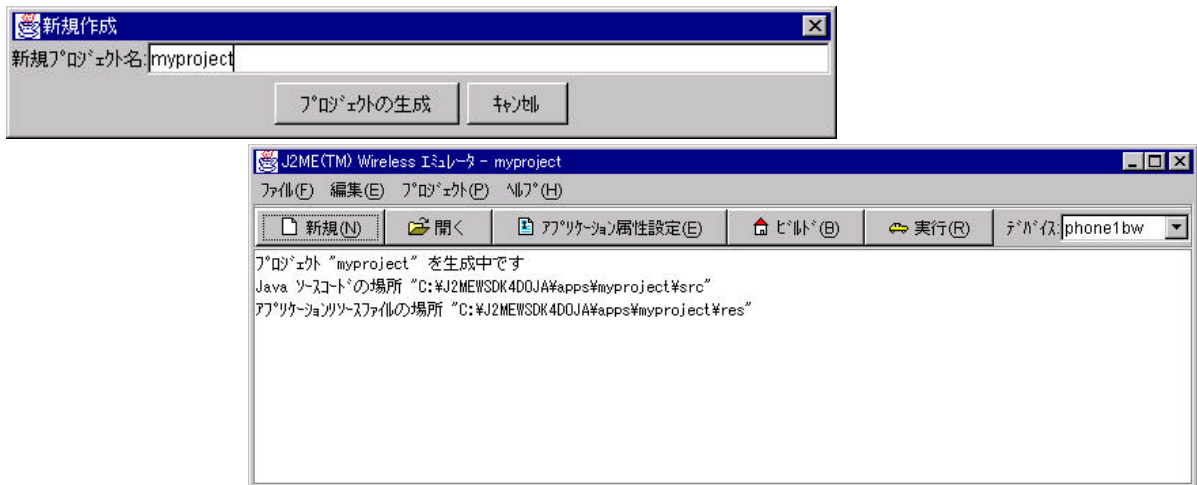


図 3-10 プロジェクトの作成

3.3.5 プロジェクトのビルドと実行

プロジェクトメニューコマンド

プロジェクト開発サイクルでは、次のプロジェクトメニューコマンドを使用します。

- 「プロジェクト」->「ビルド」(または「ビルド」ボタン)。アプリケーション Java ソースファイルのコンパイルと事前検証を行い、アプリケーション JAR ファイルとアプリケーション・ディスクリプタ・ファイルを作成します。
- 「デバイス：」メニュー。エミュレートするターゲットデバイスを選択します。このメニューには、ロードしたアプリケーションに対して使用できるデバイスが表示されます。
- 「プロジェクト」->「実行」(または「実行」ボタン)。ロードしたプロジェクトを、選択したデバイスで実行します。

メッセージコンソール

システムやアプリケーション実行中のトレースおよびアプリケーションからの標準出力へのメッセージ出力は、メッセージコンソールに表示されます。「編集」->「コンソールのクリア」を選択すると、画面のメッセージが消去されます。

KToolBar の終了

「KToolBar」を終了するには、「ファイル」->「終了」の順でメニュー項目を選択します。

3.4 サンプルプロジェクト

J2ME Wireless SDK のインストールディレクトリには、次のサンプルアプリケーションが含まれています。

- バンキングデモ(サーバコードが入っているディレクトリ: J2ME Wireless SDK_DIRECTORY¥tools¥bankingserver)
- 株取引デモ
- UI デモ
- スペースインベーダ(サーバコードが入っているディレクトリ: J2ME Wireless SDK_DIRECTORY¥tools¥scoreserver)
- テトリス

第 4 章

Forte for Java Community Edition の使用

4.1 はじめに

この章は、次のトピックから構成されています。

- 添付されているサンプルアプリケーションのビルドと実行を行うための手順
- Forte for Java のもとで実行するアプリケーションのビルドと実行に必要な手順の詳細説明

4.1.1 Forte for Java

モジュラー環境の Forte for Java には Java アプリケーションの作成に必要な編集、コンパイル、実行の機能が統合されており、ユーザーの生産性を向上させることができます。J2ME Wireless モジュールは、J2ME Wireless SDK for the DoJa API 環境向けアプリケーションのコンパイルや実行を行うための Forte for Java の拡張モジュールです。開発者は、Forte for Java の中からエミュレータを直接起動することができます。

Forte for Java Community Edition のダウンロードや、Forte for Java の使用とインストールに関する追加ドキュメントは、Forte for Java の Web サイト <http://www.sun.com/forte/ffj/ce/> を参照してください。

表 4-1 に、この章で使用する用語の定義を示します。

表 4-1 用語の定義

用語	定義
アプリケーション	エミュレータで実行する J2ME Wireless SDK for the DoJa API アプリケーション
プロジェクト	ファイル、ディレクトリ、および、IDE の現在の状態を Forte for Java がグループ化したもの
パッケージ	クラスファイルからなる Java パッケージ
ファイルシステム	ネイティブのファイルシステム
マウントされたファイルシステム	Forte for Java の内部で使用するためにマウントしたディレクトリ
アプリケーション・ディスクリプタ・ファイル(ADF)	アプリケーション情報を記述するテキストファイル

4.2 サンプルアプリケーションのビルドと実行

この節では、Forte for Java の使い方の概要と、サンプルアプリケーションのビルドと実行に関する手順を説明します。

ここではアプリケーションの実行方法を説明しますが、アプリケーションは、J2ME Wireless SDK インストーラによって C:\forte4j\Development にあらかじめコピーされ、Forte for Java のデフォルトプロジェクトウィンドウに表示されているものとします。

主なステップは次の 2 つです。

- Forte for Java の起動
- アプリケーションのビルドと実行

これらのステップについて、次の各項で詳しく説明します。

4.2.1 Forte for Java の起動

1. 「スタート」->「プログラム」を選択して Forte for Java を実行します。

コンパイラタイプに J2ME Wireless コンパイラを設定するかどうかを示すプロンプトが表示されます(図 4-1)。

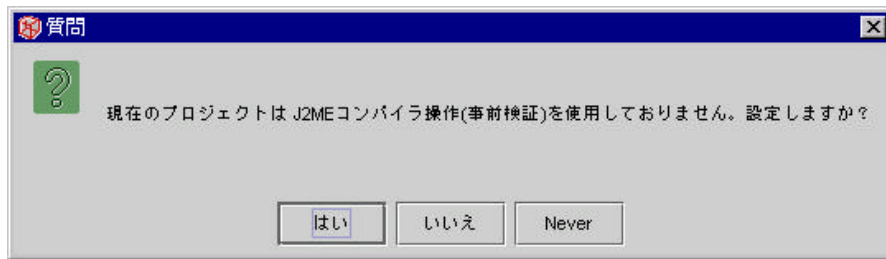


図 4-1 J2ME コンパイラの設定

2. プロンプトで「はい」をクリックします。

4.2.2 アプリケーションのビルドと実行

「エクスプローラ [ファイルシステム]」ウィンドウが表示されます(図 4-2)。サンプルディレクトリが C:\forte4j\Development ディレクトリの下に表示されます。

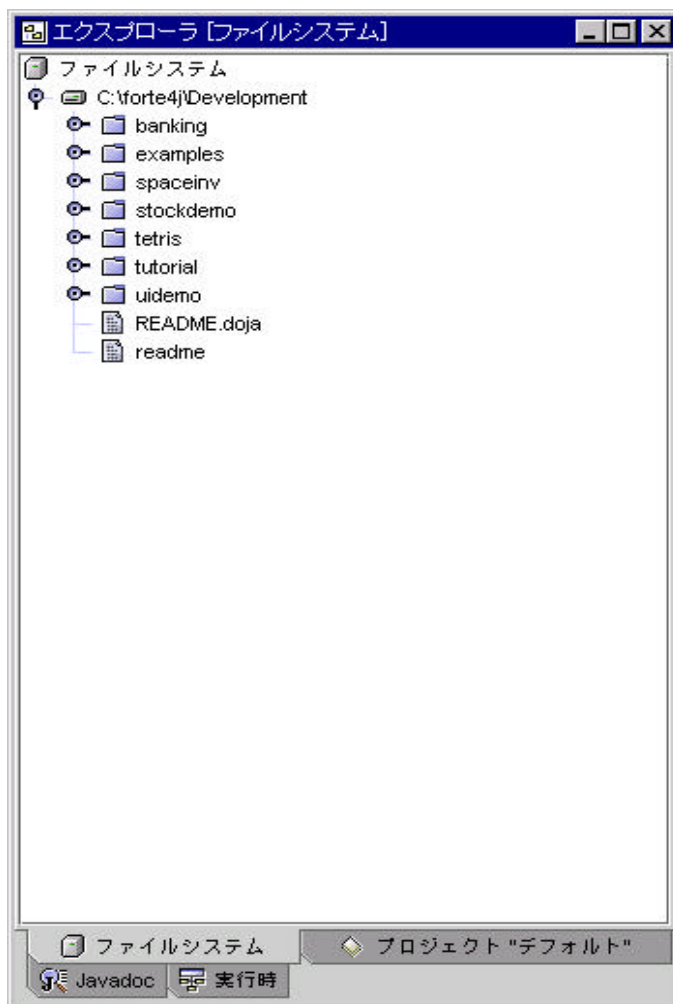


図 4-2 Forte for Java サンプルディレクトリ

ビルドと実行を行う(例 uidemo)サンプルディレクトリを見つけます。

1. uidemo のディレクトリをオープンします。

uidemo ディレクトリの内部には、uidemo アプリケーションのすべてのソースファイルが表示されます。このディレクトリには、アプリケーション・ディスクリプタ・ファイル (ADF)もあります。

2. アプリケーションを実行するには、エクスプローラ内の ADF を右クリックし、「実行」メニュー項目をクリックします(図 4-3)。

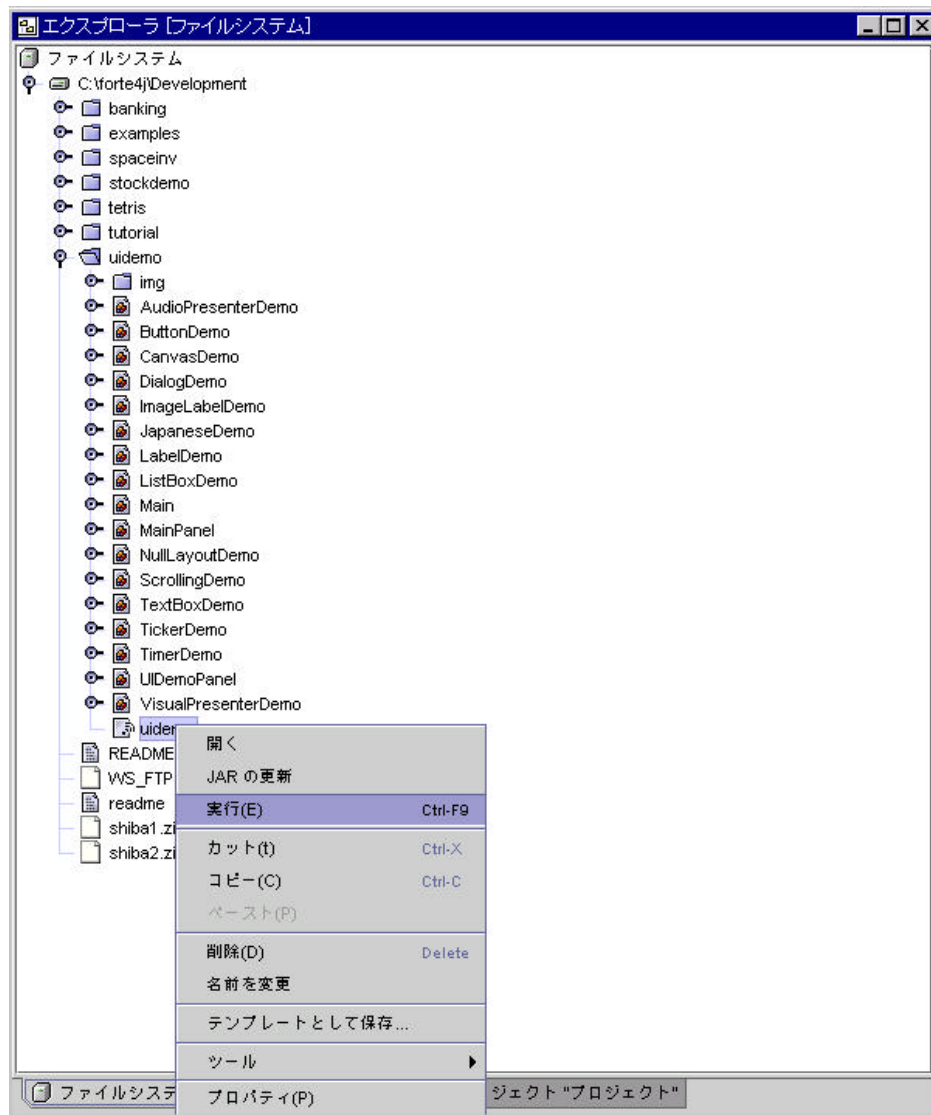


図 4-3 アプリケーションの実行

エミュレータ上で動作するアプリケーションが表示されます(図 4-4)。画面には、携帯電話のイメージとともに、動作するアプリケーションの名前を示す「実行の表示」画面が表示されます。



図 4-4 動作するアプリケーション

4.3 新規アプリケーションの作成

この節では、新しいアプリケーションの作成方法について説明します。

主なステップは次のとおりです。

- プロジェクトの作成
- ファイルシステムの構成
- アプリケーション・ディスクリプタ・ファイル(ADF)の作成
- ソースファイルの作成
- ADF の設定
- プロジェクト jar ファイルの内容を設定
- アプリケーションのビルドと実行

これらのステップについて、次の各項で詳しく説明します。

4.3.1 プロジェクトの作成

1. 「プロジェクト」->「新規プロジェクト」の順でメニュー項目を選択します。
「新規プロジェクトを作成」の画面が表示されます(図 4-5)。

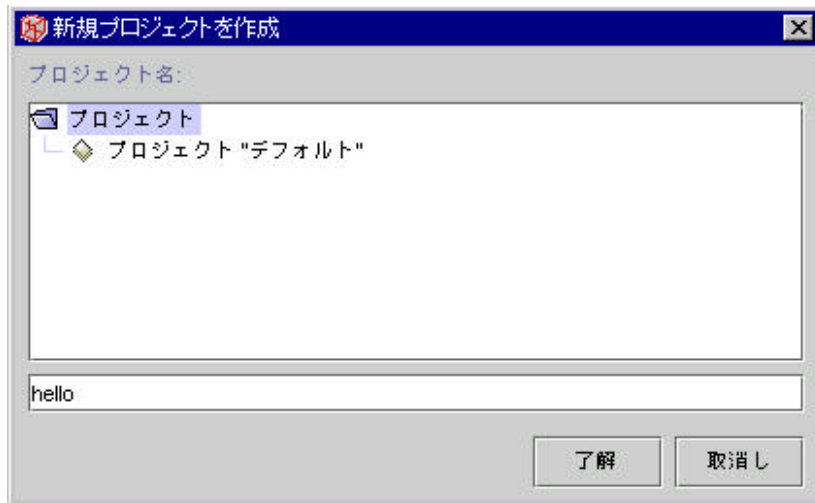


図 4-5 「新規プロジェクトを作成」の画面

2. データ入力域に新しいプロジェクトの名前を入力します。
図 4-5 は、新しいプロジェクト名として「hello」を入力した例です。
3. 「了解」をクリックします。
新しいプロジェクトが作成されます。

注 - 新しいアプリケーションを作成するたびに新しいプロジェクトを開始する必要はありませんが、できるだけそうすることをお勧めします。そうすることで柔軟性が増すだけでなく、Forte for Java を完全な IDE としてよりよく活用することができます。

4.3.2 ファイルシステムの設定

新しいファイルシステムを使用するか、既存のファイルシステムを継続して使用するかをたずねるプロンプトが表示されます(図 4-6)。

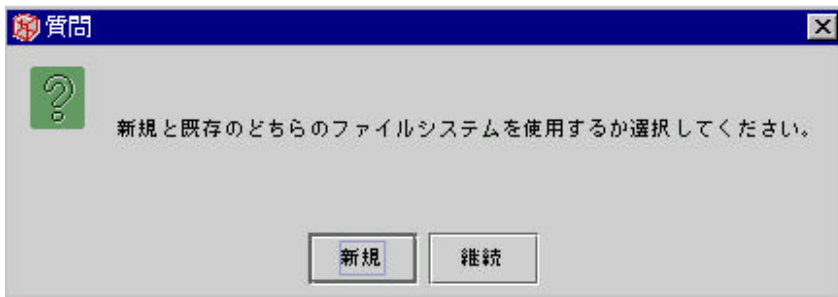


図 4-6 ファイルシステムの設定プロンプト

オプションの意味は次のとおりです。

- 新規 - このプロジェクトのために新しいファイルシステムが作成され、プロジェクトのアプリケーションから使用できる新しいディレクトリがマウントされます。
- 継続 - 前のプロジェクトの一部としてマウントされるすべてのディレクトリが新しいプロジェクトに移されます。

プロジェクトごとに新しいファイルシステムを使用することをお勧めします。

1. どちらかのオプションをクリックします。

プロジェクトがすでにオープンされている場合は、それを保存するかどうかについて応答する必要があります(図 4-7)。

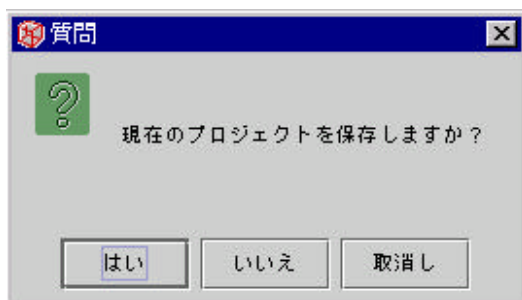


図 4-7 現在のプロジェクトの保存プロンプト

2. 適切な応答をクリックします。

「ディレクトリをマウント」のダイアログが表示されます(図 4-8)。

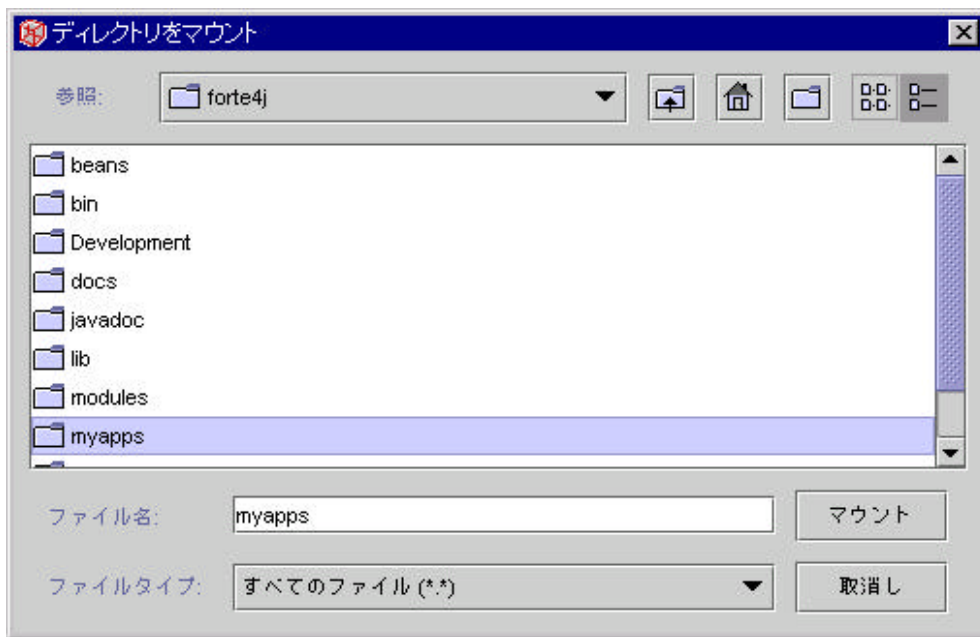


図 4-8 「ディレクトリをマウント」ダイアログ

注 - 「ディレクトリをマウント」のダイアログは、「エクスプローラ [ファイルシステム]」ダイアログ中の [ファイルシステム] を右クリックし、次に「ディレクトリを追加」をクリックすることによって表示することもできます。

3. 「ディレクトリをマウント」のダイアログの「参照」ドロップダウンメニューからプロジェクトの一部としてマウントするディレクトリを選択します(図 4-9)。

マウントするディレクトリは、ソースファイルのパッケージがある場所になしてください。

たとえば、図 4-9 では、マウントするディレクトリとして forte4j \Development \examples が選択されています。

注 - アプリケーションを作成するには、すべてのソースファイルが同じパッケージに属し、かつ、そのパッケージが、マウントするディレクトリ内になければなりません。

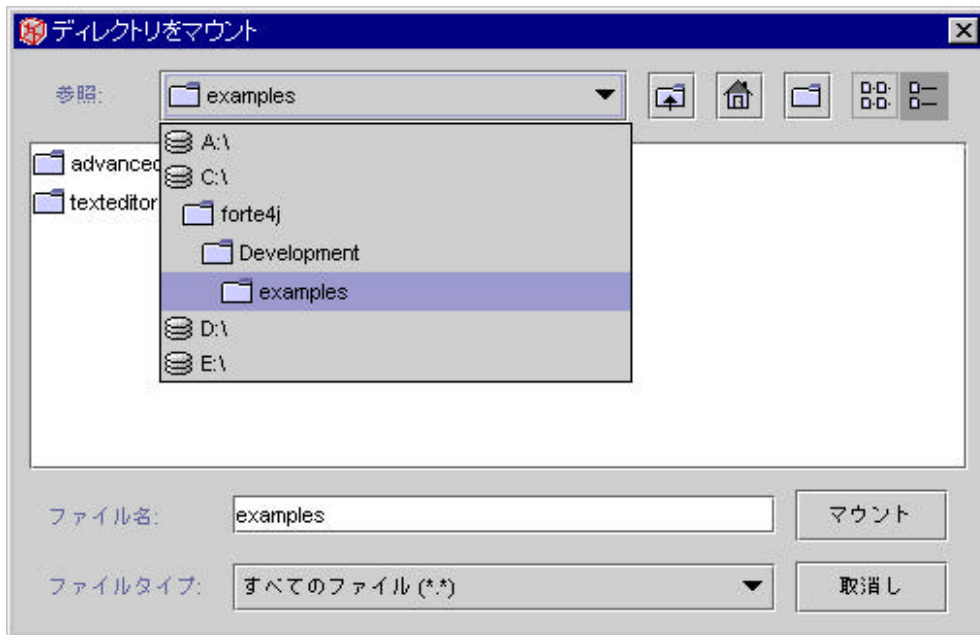


図 4-9 マウントするディレクトリの選択

4. 「マウント」をクリックします。

マウントするディレクトリが「エクスプローラ [ファイルシステム]」ウィンドウに表示されます。

5. 新しくマウントするディレクトリをプロジェクトに追加するには、マウントするディレクトリを選択して右ボタンをクリックし、「ツール」->「プロジェクトに追加」の順でメニュー項目をクリックします(図 4-10)。

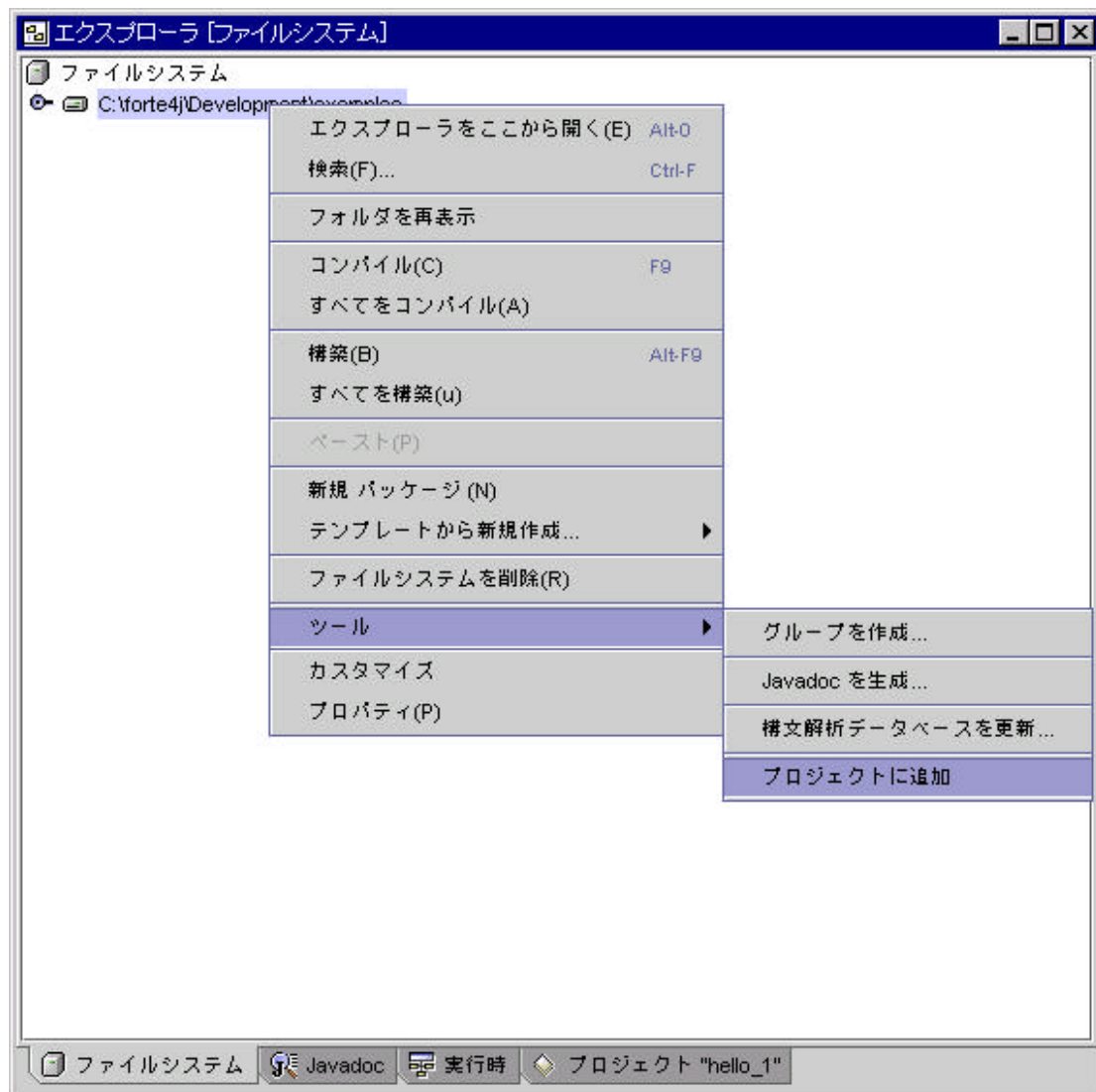


図 4-10 マウントするディレクトリをプロジェクトに追加

4.3.3 アプリケーション・ディスクリプタ・ファイル(ADF)の作成

ADF の作成では、「エクスプローラ [プロジェクト]」画面を使用します。

1. 「エクスプローラ」画面の「プロジェクト」タブをクリックします。
「エクスプローラ [プロジェクト]」画面が表示されます。
2. プロジェクトのルートディレクトリを選択し、右ボタンをクリックしてメニューを表示し、次に「テンプレートから新規作成」->「J2ME Wireless」->「DOJA」の順でメニュー項目をクリックします(図 4-11)。



図 4-11 ADFダイアログの表示

ADF を指定するダイアログが表示されます(図 4-12)。

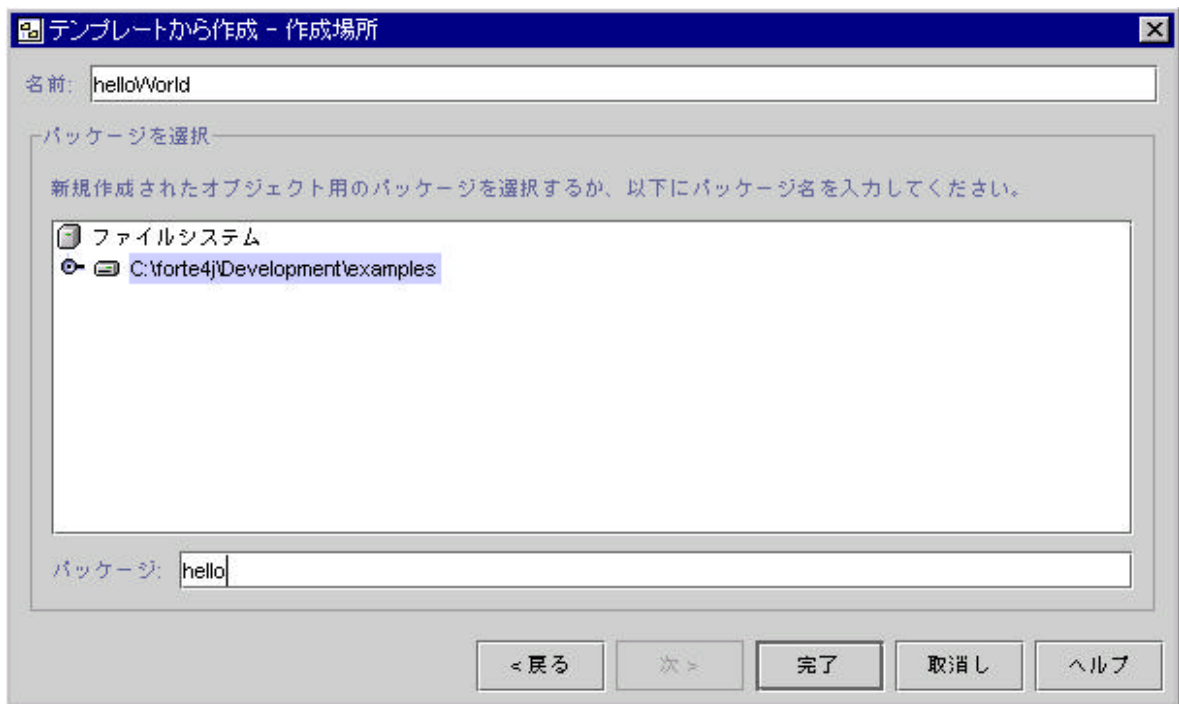


図 4-12 ADF の指定

3. 「名前:」の後に ADF の名前を入力します。
4. 「パッケージ」フィールドに、ソースファイルを入れるパッケージの名前を入力します。
パッケージディレクトリがない場合は、Forte for Java がユーザーに代って作成します。
5. 「完了」をクリックします。
hello ディレクトリの下に ADF ファイル helloWorld が表示されます。

4.3.4 ソースファイルの作成

ここでは、アプリケーションのソースファイルを作成します。

1. パッケージのディレクトリを選択し、右ボタンをクリックして、ソースファイルをパッケージディレクトリに作成します(図 4-13)。

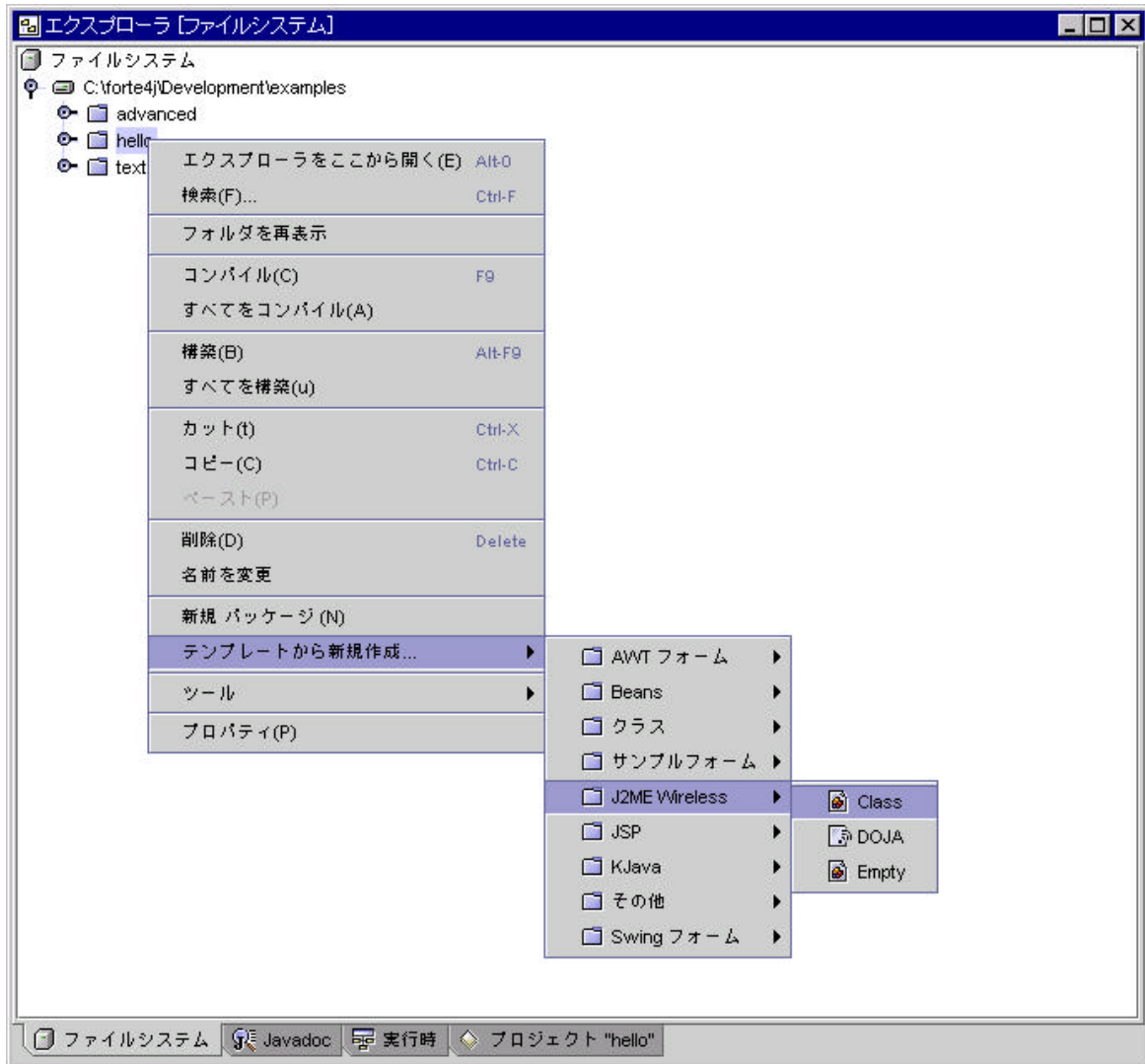


図 4-13 ソースファイルの作成

「ソースファイルの作成」ダイアログが表示されます(図 4-14)。

2. 「名前:」の後にソースファイルの名前を入力します(図 4-14)。ソースファイルの名前は ADF の AppClass で指定したクラス名と同じ名前をつけて下さい。

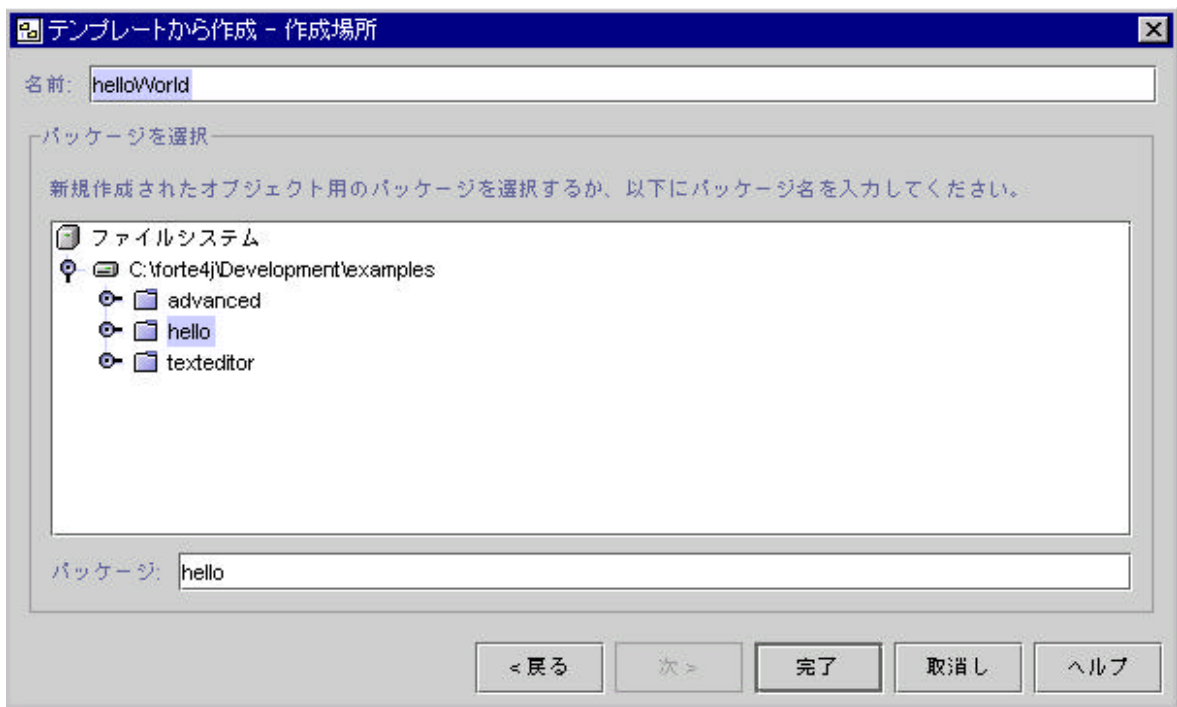


図 4-14 ソースファイル名の指定

ソースファイルを作成すると、ソースファイルエディタが表示され、そこにソースファイルのスケルトンがロードされます。

3. ソースファイルに「println("HelloWorld")」を追加します(図 4-15)。

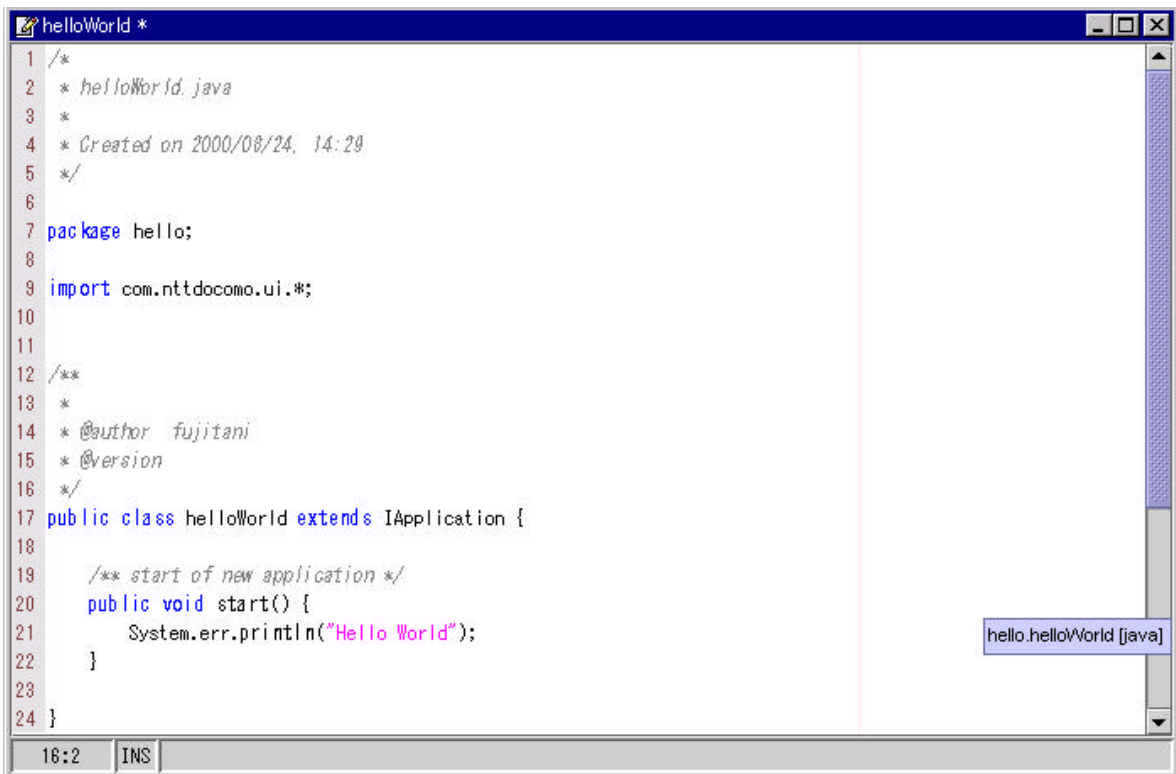


図 4-15 ソースファイルの編集

4. 「ファイル」->「保存」の順でメニュー項目を選択するか、Ctrl-S を押して、ソースファイルを保存します。

ソースファイル helloWorld.java が、パッケージ hello の一部としてディレクトリ hello の下に表示されます。

注 - アプリケーションの main クラスの名前は、アプリケーション・ディスクリプタ・ファイルの AppClass キーで指定した名前と同じにする必要があります。

4.3.5 ADF の設定

1. 「エクスプローラ [プロジェクト hello]」画面で ADF helloWorld を選択し、右ボタンをクリックしてオープンします(図 4-16)。

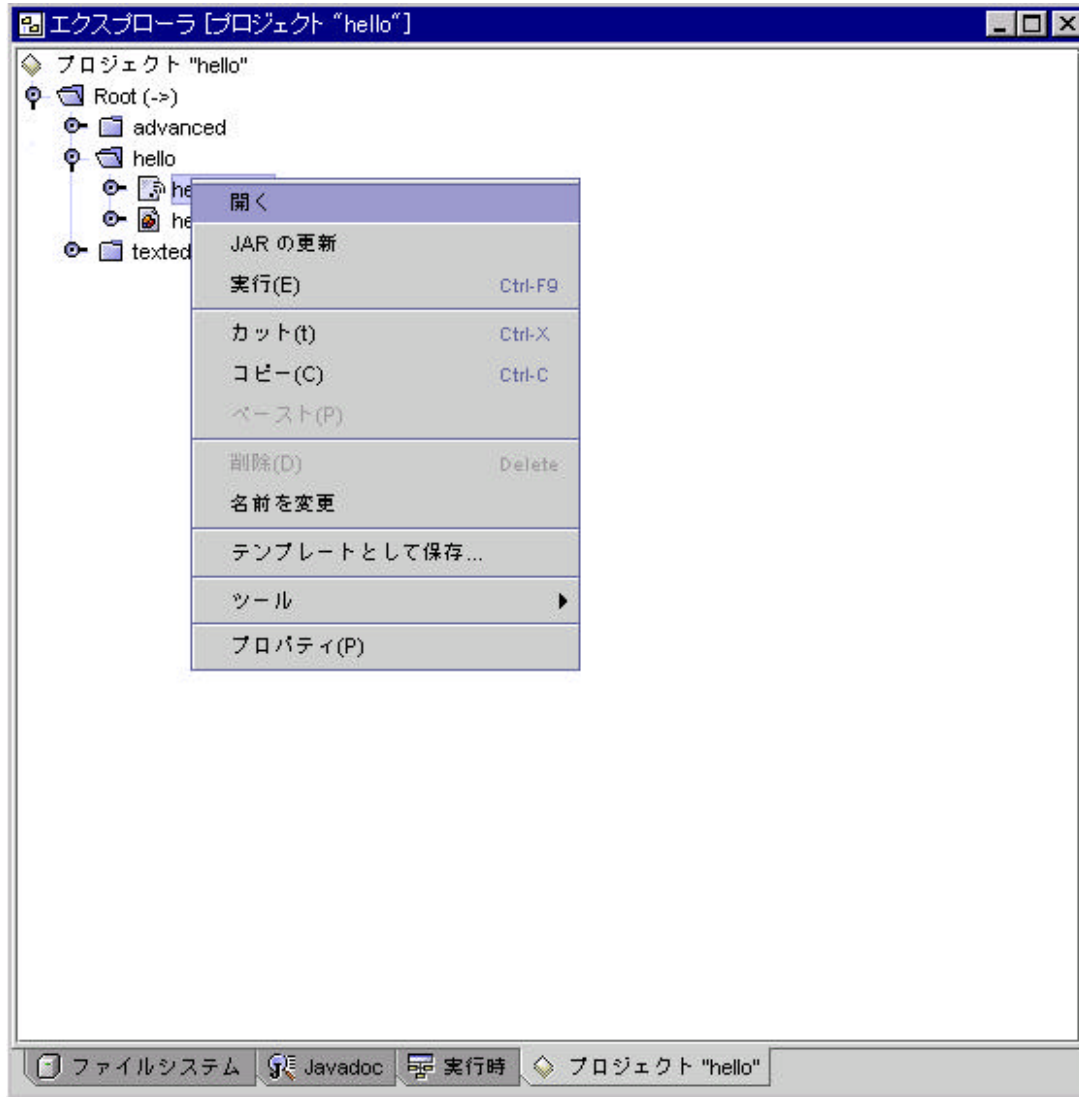


図 4-16 ADF のオープン

ADF 設定ダイアログが表示されます(図 4-17)。



図 4-17 ADF ディスクリプタ画面

設定項目は、表 4-2 に示すとおりです。

表 4-2 ADF の設定項目

設定項目	説明
必須フィールド: ¹	
AppName	アプリケーションの名前(最大 16 バイト)。AppName には、SJIS 文字を使用できません。
AppClass	アプリケーションの main クラス。AppClass では、com.nttdocomo.ui.IApplication を継承しなければなりません。
AppSize	アプリケーションサイズ(バイト単位)。このフィールドは、JAR ファイルの生成時にビルドプロセスによって自動的に生成されます。ユーザーが編集することはできません。
PackageURL	アプリケーションの場所を表す URL(最大 255 バイト) 新規のプロジェクトが作成されたときのデフォルト値はアプリケーションの JAR ファイルの名前になっています。このフィールドは実際に配置される場所の URL に変更して下さい。
LastModified	日本標準時(JST)による最終更新時刻 形式: Fri, 25 Feb 2000 12:04:25 このフィールドは、JAR ファイルの生成時にビルドプロセスによって自動的に生成されます。ユーザーが編集することはできません。
任意フィールド:	
AppVer	アプリケーションのバージョン。このキーは、参考情報であり、エミュレータでは使用しません
AppParam	Main クラスのパラメータをスペースで区切って指定します(最大 255 バイト)。Java アプリケーションは、com.nttdocomo.ui.IApplication.getArgs()メソッドを使ってこれらのパラメータを取り出します。このメソッドは、このキーが指定されていないと、null を返します。
KvmVer	KVM バージョン(たとえば、CLDC-1.0)。このキーを指定しないと、アプリケーションは CLDC Virtual Machine のすべてのバージョンで動作するものとみなされます。
SPsize	ScratchPad のサイズ(バイト単位)。エミュレータ上で指定できる ScratchPad サイズに制限はありませんが、実際の端末では機種によりサイズの制限があります。実際の端末での ScratchPad サイズを検討する際には、『i モード対応 Java コンテンツ開発ガイド～詳細編～ 第 3 章 設計上の考慮事項』を参照して下さい。なおこのキーを指定しないと、ScratchPad を使用することはできません。
UseNetwork	アプリケーションでネットワーク通信を使用することを表すフラグ。このキーを指定しないと、アプリケーションからネットワーク接続を行うことはできません。有効な値は「http」だけです。
TargetDevice	ターゲットデバイスをコマンドで区切って指定します。アプリケーションでメーカー固有の拡張 API を使用するとき指定します。
LaunchAt	起動時間間隔指定。アプリケーションを一定時間毎に自動起動させるときに指定します。 形式: Ih(h は起動間隔(時間単位))

1. 必須キーは、有効な値とともにアプリケーション・ディスクリプタ・ファイルに必ず指定する必要があります。
2. 各設定項目の詳細については、『i モード対応 Java コンテンツ開発ガイド～詳細編～』の「第 9 章 アプリケーションのビルド、テスト、およびパッケージ化」も参考にして下さい。

4.3.6 プロジェクト jar ファイルの内容を設定

ここでは、プロジェクト jar ファイルの内容を指定する方法について説明します。プロジェクト jar に追加したいディレクトリを選択すると、この情報がアプリケーションのビルドで使用されます。

1. ADE をオープンし、「コンテンツ」タブをクリックします。

「Project Jar content」ダイアログが表示されます(図 4-18)。

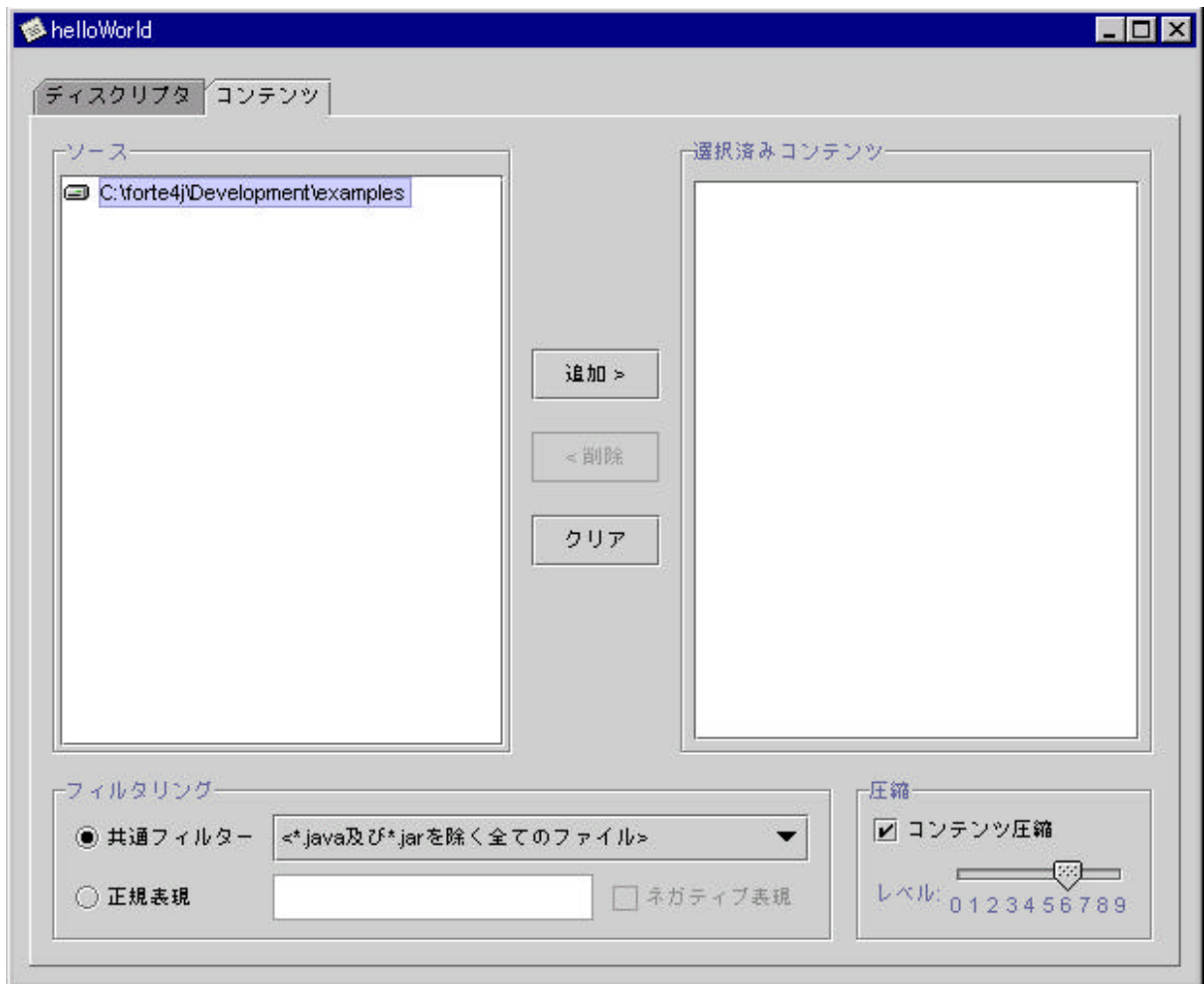


図 4-18 内容の指定

2. クラスファイルを jar ファイルに転送するため、ソースファイルのあるディレクトリ（ここでは hello ディレクトリ）を選択して追加します(図 4-19)。

注 - フィルタを指定することもできます。ここでは、共通フィルターが使用されます。



図 4-19 内容を Jar ファイルに転送

4.3.7 アプリケーションのビルドと実行

ビルドしたアプリケーションを実行するには、その ADF を選択してから右ボタンをクリックします(図 4-20)。

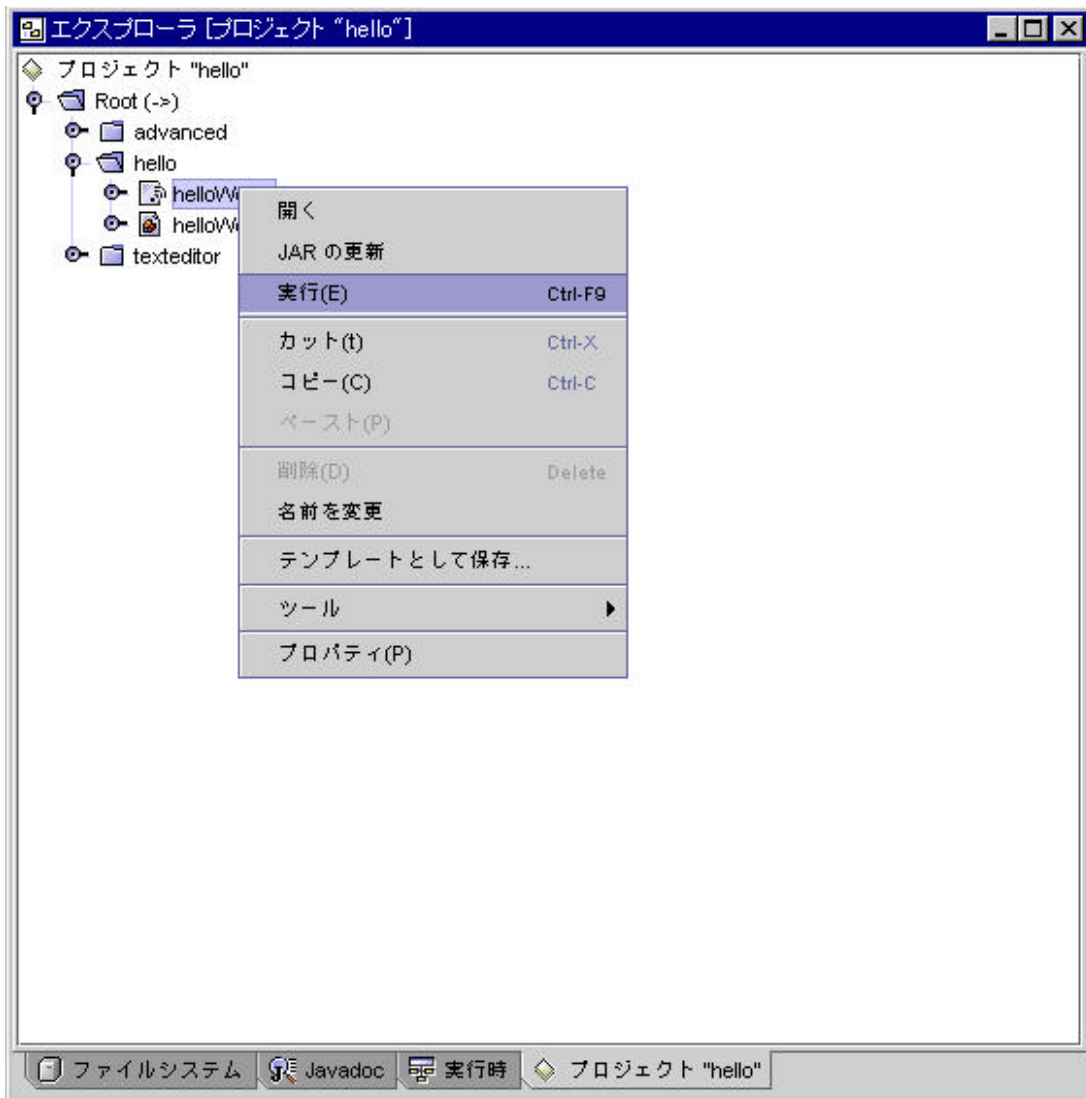


図 4-20 アプリケーションの実行

これによって、アプリケーション helloWorld がビルドされ、実行されます。

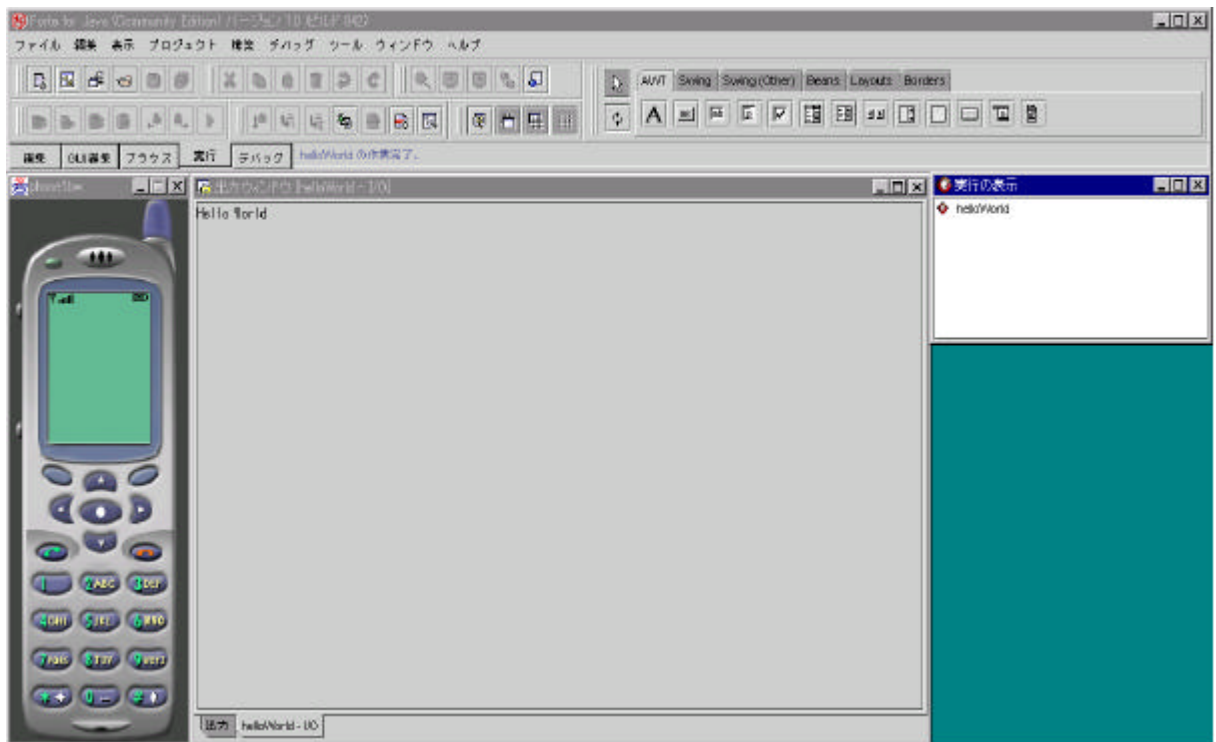


図 4-21 動作するアプリケーション

動作中のデバイスエミュレーションが表示されます(図 4-21)。

4.4 エミュレータの設定

この節では、エミュレータの設定方法について説明します。内容は次のとおりです。

- デバイスの設定
 - プロキシの設定
1. メインメニューバーから「プロジェクト」->「設定」の順でコマンドを選択して、「プロジェクト設定」ウィンドウを表示します。
 2. 「プロジェクト設定」ウィンドウで「実行の種類」->「エミュレーター実行」->「J2ME Wireless Emulator」を右クリックし、次に「カスタマイズ」をクリックします(図 4-22)。

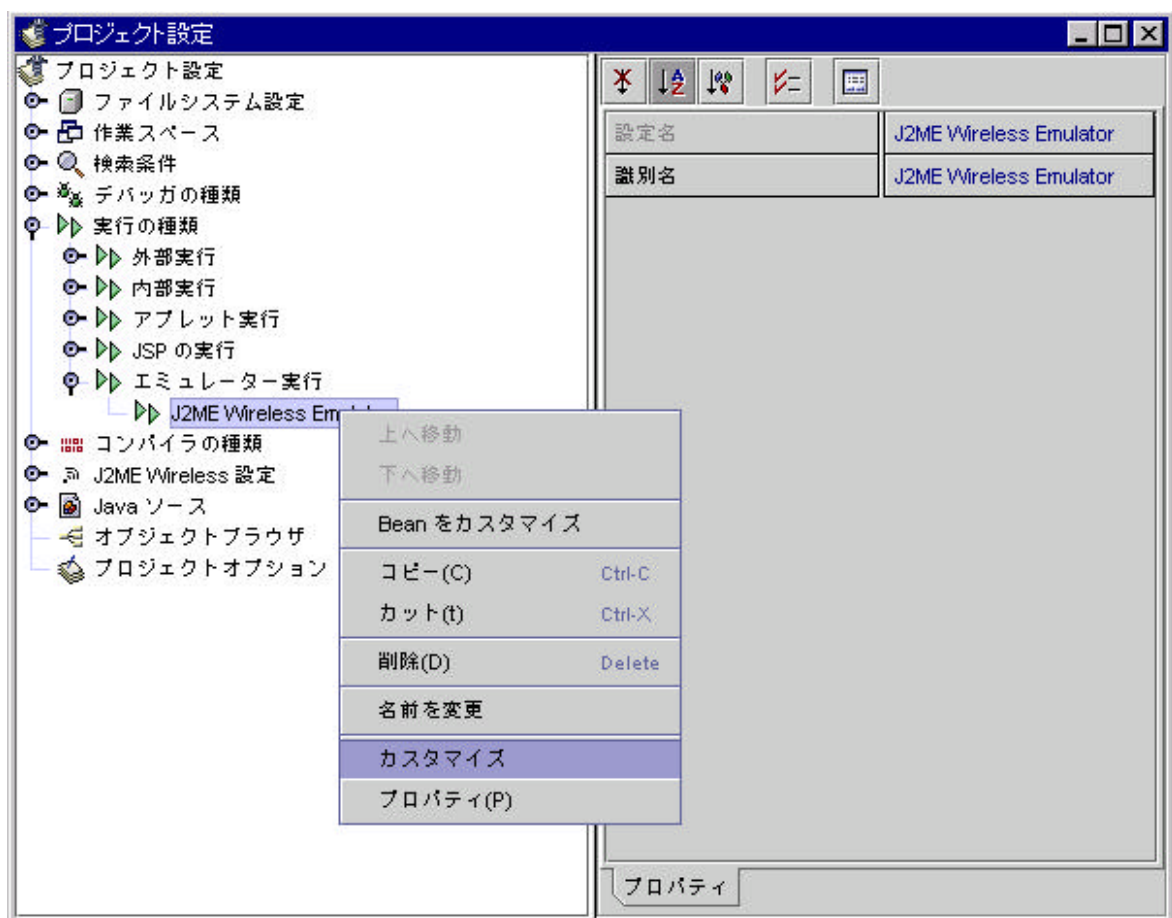


図 4-22 エミュレータの設定

「カスタマイズ ダイアログ」が表示されます(図 4-23)。左側の欄には設定項目が、右側の欄には現在の設定値がそれぞれ表示されます

3. 変更したい設定項目の値を選択します。

選択項目がある設定項目の場合には下矢印が表示され、設定可能な値がドロップダウンメニューに表示されます。値を記述する設定項目の場合は入力欄が編集可能状態になります。

4. 必要な設定項目をクリックします。

たとえば、デバイスを変更する場合は、デバイスの値(図では phone1bw)をクリックし、下矢印をクリックして、新しい値を選択します。



図 4-23 カスタマイザダイアログ

エミュレータ設定項目は表 4-3 のとおりです。

表 4-3 エミュレータの設定項目

設定項目	説明	デフォルト
デバイス	ターゲットデバイスの名前。デバイスプロパティファイルの名前と同じです。	phone1bw
読み込み先 URL	エミュレータは、アプリケーションがこの URL からダウンロードされたものとみなします。アプリケーションは、このフィールドに指定されたホストとポートだけとしか通信できません。このフィールドの形式は、ADF の PackageURL タグの形式と同じです。 このフィールドが「*」なら、アプリケーションは、どのサーバとも通信できます。この設定はエミュレータ上で Java アプリケーションの開発とテストを行う場合にのみ使用できます。 このフィールドがブランクなら、アプリケーションは、ネットワークを全く使用できません。	なし
ネットワークアクセス	ネットワークアクセスの設定は、エミュレータが Java アプリケーションからのネットワーク接続を許可するかどうかを管理します。ネットワークアクセスの設定を有効にするには、アプリケーション・ディスクリプタ・ファイルの "UseNetwork" キーに "http" が指定されていないかならないことに注意して下さい。 アプリケーション・ディスクリプタ・ファイルの "UseNetwork" キーが定義されていない場合、NetworkAccess に設定された値にかかわらずネットワーク接続を行おうとすると Java アプリケーションは終了します。 NetworkAccess の設定には、以下のような値を設定できます。 • enable - Java アプリケーションからのネットワーク接続を許可します。この接続は読み込み先 URL に設定された値に基づきコントロールされます。 • disable - Java アプリケーションからのネットワーク接続は許可されません。Java アプリケーションがネットワーク接続を行おうとした場合は SecurityException がスローされます。	disable
HttpProxy	アプリケーションが Web サーバとの通信に使用する HTTP プロキシサーバのアドレスとポート。形式は <address>:<port> です (「4.4.1 プロキシサーバの設定」を参照)。	なし
HttpsProxy	アプリケーションが Web サーバとの通信に使用する HTTPS プロキシサーバのアドレスとポート。形式は <address>:<port> です (「4.4.1 プロキシサーバの設定」を参照)。	なし
Secure Prng	高速の代替 PRNG アルゴリズムの代わりに標準のセキュア PRNG を使用します (「4.4.2 PRNG アルゴリズムの選択」を参照)。	オフ
ガーベージコレクションのトレース	ガーベージコレクション情報を表示します (「4.4.3.1 ガーベージコレクションの追跡」を参照)。	オフ
クラスローディングのトレース	クラスローディング情報を表示します (「4.4.3.2 クラスローディングの追跡」を参照)。	オフ

メソッド呼び出しの トレース	メソッド呼び出しを表示します(「4.4.3.3 メソッド呼び出しの追跡」を参照)。	オフ
例外のトレース	例外を表示します(「4.4.3.4 例外の追跡」を参照)。	オフ

4.4.1 プロキシサーバの設定

エミュレータのもとで動作するJava アプリケーションがプロキシサーバ経由でWebサーバに接続する必要がある場合には(たとえば、Webサーバがファイアウォールの反対側にある場合など)、エミュレータの設定でプロキシサーバの情報を指定する必要があります。アプリケーションはHTTP か HTTPS プロトコルを使って Web サーバに接続しますが、同じアプリケーションで両方のプロトコルを使用することはできません。

HTTP と HTTPSのプロキシサーバのアドレスとポートは図4-23 の設定画面で入力します。これらの設定値がわからない場合は、システム管理者に問い合わせして下さい。

4.4.2 PRNG アルゴリズムの選択

HTTPS プロトコルでは、非公開鍵や公開鍵の暗号化のために、安全な擬似ランダム数値ジェネレータ(PRNG)アルゴリズムが必要です。

このアルゴリズムには、次の 2 つがあります。

- 標準のセキュア PRNG アルゴリズム

J2SE に組み込まれているこのアルゴリズムは安全ではありますが、低速です。このアルゴリズムでは、Windows システムで J2ME Wireless SDK を起動するたびに、初期化に最大 60 秒を要することがあります。

- 高速 PRNG アルゴリズム

J2ME Wireless SDK に組み込まれている代替 PRNG アルゴリズムは、ブルートフォース攻撃(力まかせ攻撃 Brute-Force Attack)や、既知平文攻撃(Known-Plaintext Attack)、選択平文攻撃(Chosen-Plaintext Attack)、繰り返し攻撃(Replay Attack)といった攻撃に対して標準アルゴリズムほど堅牢ではありませんが、はるかに高速で、初期化は 1 秒足らずで終了します。

アプリケーションのテストでは、どちらのアルゴリズムを選択しても違いはありません。プロトコルの実際の手順はどちらも全く同じです。したがって、ネットワークを通して送信するデータが本当に機密なものである場合を除き、一般的には高速 PRNG アルゴリズムを選択します。

注 - テストするアプリケーションがネットワークを通して機密データを送信する場合は、標準のセキュア PRNG アルゴリズムを使用してください。

標準のセキュア PRNG アルゴリズムをアクティブにするには、図 4-23 の「SecurePmg」エントリをオンにします。オフにすると、高速 PRNG アルゴリズムが使用されます。

4.4.3 ログメッセージのトレース

KToolBar の設定により、次の 4 種類のログメッセージをトレースできます。

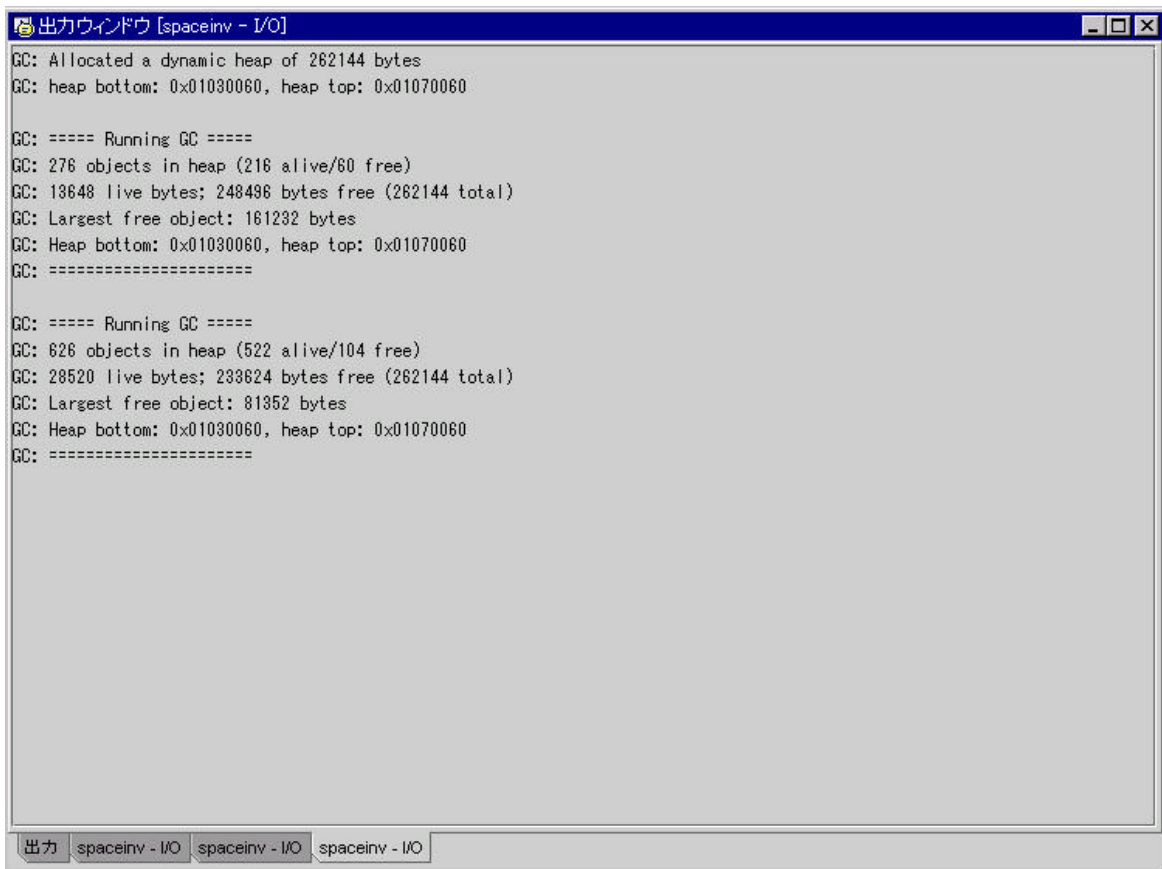
- ガベージコレクション
- クラスローディング
- メソッド呼び出し
- スローされた例外

図 4-23 の画面で、該当するエントリをオンにしてください。

4.4.3.1 ガベージコレクションのトレース

図 4-24 は、ガベージコレクションのトレース画面を表したものです。次の情報が表示されます。

- Java ヒープの割り当て
- GC スキャンが開始されたことを示す通知
- いくつかのオブジェクトが割り当てられているかを示す実行時カウンタ
- 割り当てられているオブジェクトの合計サイズ



```
出力ウィンドウ [spaceinv - I/O]
GC: Allocated a dynamic heap of 262144 bytes
GC: heap bottom: 0x01030060, heap top: 0x01070060

GC: ==== Running GC ====
GC: 276 objects in heap (216 alive/60 free)
GC: 13648 live bytes; 248496 bytes free (262144 total)
GC: Largest free object: 161232 bytes
GC: Heap bottom: 0x01030060, heap top: 0x01070060
GC: =====

GC: ==== Running GC ====
GC: 626 objects in heap (522 alive/104 free)
GC: 28520 live bytes; 233624 bytes free (262144 total)
GC: Largest free object: 81352 bytes
GC: Heap bottom: 0x01030060, heap top: 0x01070060
GC: =====
```

図 4-24 ガベージコレクションのトレース

4.4.3.2 クラスローディングのトレース

図 4-25 は、クラスローディングのトレース画面を表したものです。クラスがロードされるたびに、名前が表示されます。

注 - このオプションは、システムクラスには適用されません。

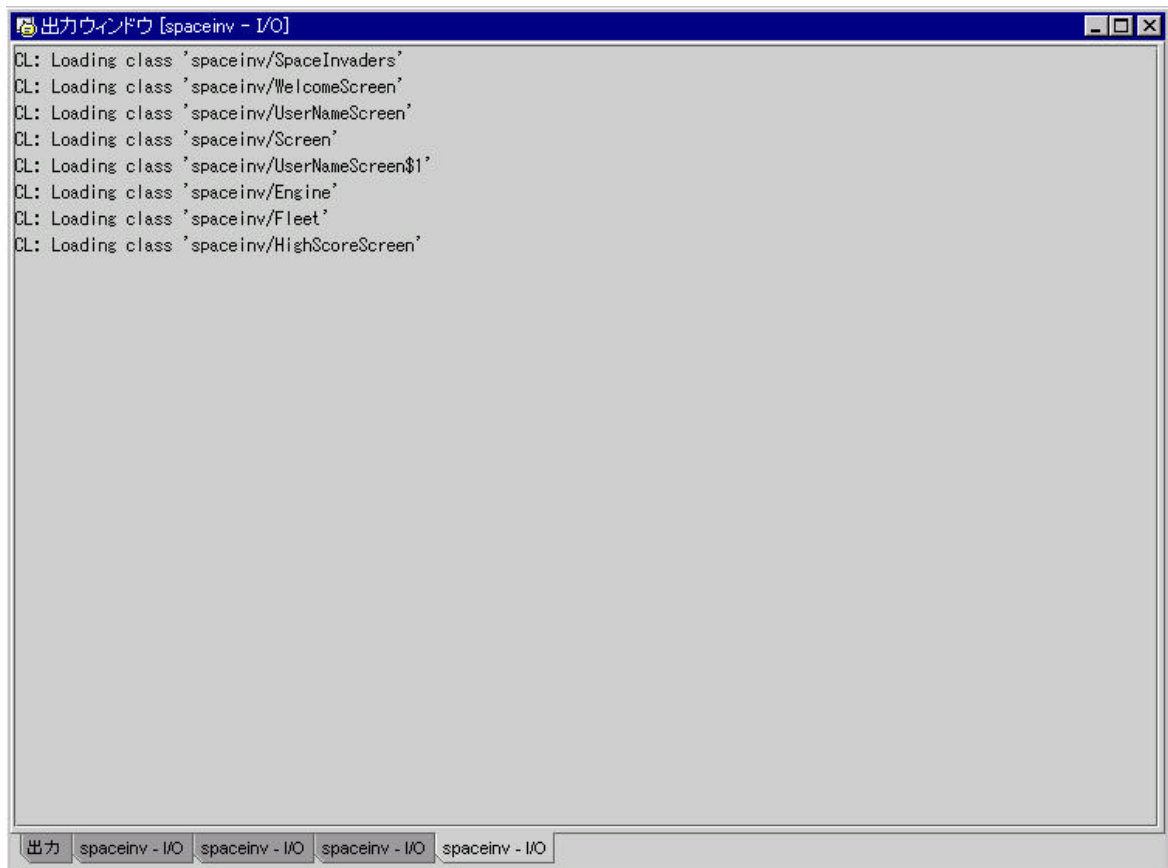
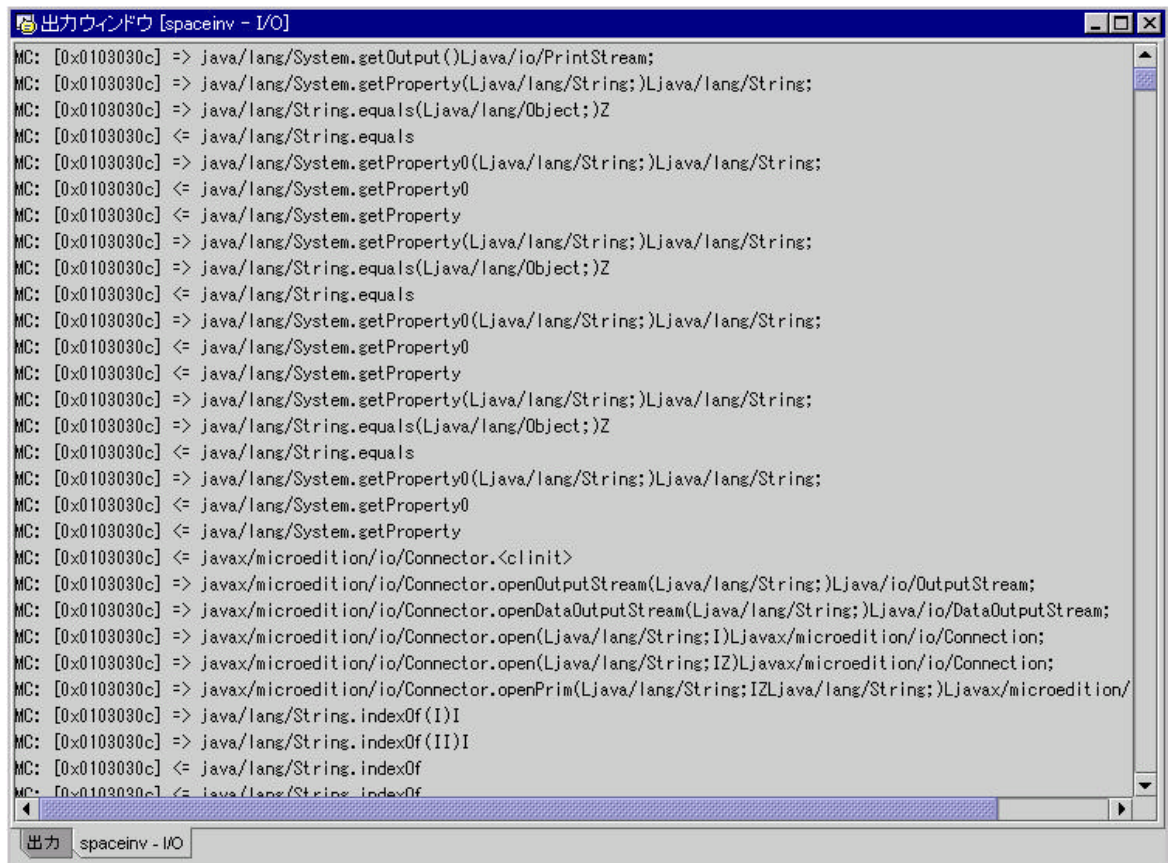


図 4-25 クラスローディングのトレース

4.4.3.3 メソッド呼び出しのトレース

図 4-26 は、メソッド呼び出しのトレース画面を表したものです。メソッドが呼び出されるたびに、オブジェクトとメソッドが表示されます。

注 - このオプションは非常に冗長であるため、アプリケーションの実行が遅くなることがあります。



```
出力ウィンドウ [spaceinv - I/O]
MC: [0x0103030c] => java/lang/System.getOutputStream()Ljava/io/PrintStream;
MC: [0x0103030c] => java/lang/System.getProperty(Ljava/lang/String;)Ljava/lang/String;
MC: [0x0103030c] => java/lang/String.equals(Ljava/lang/Object;)Z
MC: [0x0103030c] <= java/lang/String.equals
MC: [0x0103030c] => java/lang/System.getProperty0(Ljava/lang/String;)Ljava/lang/String;
MC: [0x0103030c] <= java/lang/System.getProperty0
MC: [0x0103030c] <= java/lang/System.getProperty
MC: [0x0103030c] => java/lang/System.getProperty(Ljava/lang/String;)Ljava/lang/String;
MC: [0x0103030c] => java/lang/String.equals(Ljava/lang/Object;)Z
MC: [0x0103030c] <= java/lang/String.equals
MC: [0x0103030c] => java/lang/System.getProperty0(Ljava/lang/String;)Ljava/lang/String;
MC: [0x0103030c] <= java/lang/System.getProperty0
MC: [0x0103030c] <= java/lang/System.getProperty
MC: [0x0103030c] => java/lang/System.getProperty(Ljava/lang/String;)Ljava/lang/String;
MC: [0x0103030c] => java/lang/String.equals(Ljava/lang/Object;)Z
MC: [0x0103030c] <= java/lang/String.equals
MC: [0x0103030c] => java/lang/System.getProperty0(Ljava/lang/String;)Ljava/lang/String;
MC: [0x0103030c] <= java/lang/System.getProperty0
MC: [0x0103030c] <= java/lang/System.getProperty
MC: [0x0103030c] <= javax/microedition/io/Connector.<clinit>
MC: [0x0103030c] => javax/microedition/io/Connector.openOutputStream(Ljava/lang/String;)Ljava/io/OutputStream;
MC: [0x0103030c] => javax/microedition/io/Connector.openDataOutputStream(Ljava/lang/String;)Ljava/io/DataOutputStream;
MC: [0x0103030c] => javax/microedition/io/Connector.open(Ljava/lang/String;I)Ljavax/microedition/io/Connection;
MC: [0x0103030c] => javax/microedition/io/Connector.open(Ljava/lang/String;IZ)Ljavax/microedition/io/Connection;
MC: [0x0103030c] => javax/microedition/io/Connector.openPrim(Ljava/lang/String;IZLjava/lang/String;)Ljavax/microedition/
MC: [0x0103030c] => java/lang/String.indexOf(I)I
MC: [0x0103030c] => java/lang/String.indexOf(II)I
MC: [0x0103030c] <= java/lang/String.indexOf
MC: [0x0103030c] <= java/lang/String.indexOf
```

図 4-26 メソッド呼び出しのトレース

4.4.3.4 例外のトレース

図 4-27 は、例外のトレース画面を表したものです。スローされるすべての例外が表示されます(キャッチされたかどうかに関係なく表示されます)。

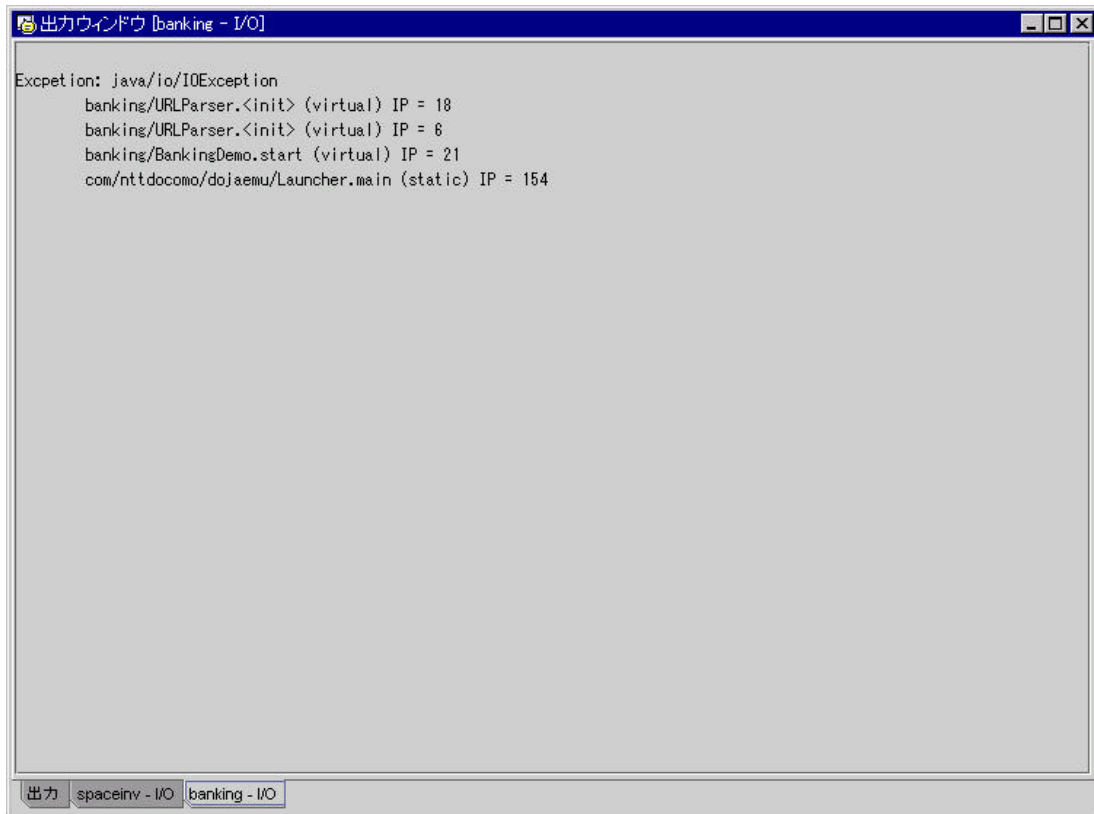


図 4-27 例外のトレース

第 5 章 コマンド行からの操作

この章では、J2ME Wireless SDK のツールをコマンド行から実行する方法について説明します。

5.1 コンパイラの操作

コマンド行からアプリケーションをコンパイルするには、次のコマンドを入力します。

```
javac -encoding SJIS -i -J-Xbootclasspath/a:<SDK>\lib\dojaconv.jar  
-g:none -d <TMP_DIR> -bootclasspath <SDK>\lib\dojaapi.jar  
-sourcepath <SRC_DIR> -classpath<TMP_DIR>[;<DEST_DIR>] <SRC>
```

<SDK>は、J2ME Wireless SDK がインストールされているディレクトリ、<SRC_DIR>はソースファイルのあるディレクトリ、<TMP_DIR>はコンパイル済みのクラスファイルを格納するディレクトリ、<SRC>はコンパイルの対象となるソースファイル、<DEST_DIR>は事前検証済みのクラスファイルを格納するディレクトリです。

コンパイル済みのクラスは、-d オプションで指定されたディレクトリ<TMP_DIR>に格納されます。このクラスファイルは、5.2 章で解説する事前検証を行うことで KVM 上で使用できるようになります。5.2 章のコマンド行の例を実行することで<TMP_DIR>のクラスファイルが事前検証され、その結果がディレクトリ<DEST_DIR>に格納されます。

たとえばカレントディレクトリが C:\J2ME\SDK4DOJA\apps\tetris で、その下の src ディレクトリ内にある tetris パッケージに属する Tetris.java をコンパイルする際には、以下のように入力します。

```
javac -encoding SJIS -i -J-Xbootclasspath/a:C:\J2ME\SDK4DOJA\lib\dojaconv.jar  
-g:none -d tmpclasses -bootclasspath C:\J2ME\SDK4DOJA\lib\dojaapi.jar  
-sourcepath src -classpath tmpclasses;classes src\tetris\tetris.java
```

なお、5.1 章および 5.2 章の処理実行前に、以下の処理を行う必要があります。

- ディレクトリ<TMP_DIR>と<DEST_DIR>の作成
- <JDK1.3>\bin と<SDK>\bin を PATH に追加（<JDK1.3>はJDK1.3がインストールされているディレクトリです）

5.2 バイトコード事前検証プログラムの実行

コマンド行からアプリケーションの事前検証を実行するには、次のコマンドを入力します。

```
preverify -classpath <CLASSPATH> -d <DEST_DIR> -check <TMP_DIR>
```

<CLASSPATH>はすべてのクラスを置く場所、<DEST_DIR>は事前検証したクラスを格納するディレクトリ、<TMP_DIR>はまだ検証されていないクラスが格納されているディレクトリです。

それぞれのパラメータの意味は、次のとおりです。

-classpath <CLASSPATH>	クラスのロード時に使用されるディレクトリ(セミコロンで区切る)または jar ファイル
-d <DEST_DIR>	出力クラスが書き込まれるディレクトリ。デフォルトの出力ディレクトリは、\output です。
-check	クラス内で浮動小数点や finalize メソッドが使用されていないかチェックします。アプリケーションで浮動小数点や finalize メソッドが使用されていると、エラーを通知します。

たとえば、ソースファイルをコンパイルした後に tmpclasses ディレクトリにあるクラスに対して事前検証を行う場合は、次のコマンドを入力します。

```
preverify -classpath <SDK>\lib\dojapi.jar -d classes -check tmpclasses
```

5.3 アプリケーション JAR ファイルの作成

アプリケーション JAR ファイルの作成には標準の J2SE jar ツールが使用され、すべてのアプリケーションクラスとリソースファイルがパッケージされます。

参考例としてはサンプルアプリケーションにある "build.bat" を使用して下さい。

5.4 アプリケーション・ディスクリプタ・ファイルの作成

アプリケーション・ディスクリプタ・ファイルは、どんなテキストエディタでも作成することができます。このファイルは通常 ".jam" という拡張子をつけて保存します。

サンプルアプリケーションのアプリケーション・ディスクリプタ・ファイルを参考に使用してください。また各キーの詳しい記述方法は表 3-2 および『i モード対応 Java コンテンツ開発ガイド～詳細編～』第 9 章「アプリケーションのビルド、テスト、およびパッケージ化」を参照して下さい。

5.5 エミュレータの操作

5.5.1 エミュレータの設定

コマンド行からエミュレータを起動する場合には、次の J2SE プロパティを設定することができます。

表 5-1 J2SE エミュレータのプロパティ

プロパティ名	デフォルト値	必須/任意	Ktoolbar で設定可能	説明
kvem.home	なし	必須	不可能	文字列。J2ME Wireless SDK がインストールされているディレクトリのトップディレクトリ
kvem.japanese_font	True	任意	不可能	ブール値。非日本語 Windows で日本語フォントの検索を可能にするかどうか
http.proxyHost	なし	任意	可能	文字列。HTTP プロキシとして使用するホスト
http.proxyPort	なし	任意	可能	整数。HTTP プロキシとして使用するポート
https.proxyHost	なし	任意	可能	文字列。HTTPS プロキシとして使用するホスト
https.proxyPort	なし	任意	可能	整数。HTTPS プロキシとして使用するポート
secure.prng	False	任意	可能	ブール値。安全な擬似ランダム数値ジェネレータを使用する。安全度は増しますが、速度は遅くなります。
doja.sourceURL	なし	任意	可能	文字列。Jar ファイルのダウンロード元 URL。この値は、 <code>com.nttdocomo.ui.IApplication.getSourceURL()</code> から返されます。HTTP 通信を行うアプリケーションでは、ここで指定するホスト・ポートにのみ接続可能です。
doja.networkAccess	disable	任意	可能	文字列。ネットワークアクセスの設定を行います。有効な値は値は"enable"または"disable"です。
doja.trace.gc	False	任意	可能	ブール値。ガベージコレクションの実行時トレース情報をオンにします。
doja.trace.calls	False	任意	可能	ブール値。メソッド呼び出しの実行時トレース情報をオンにします。
doja.trace.class	False	任意	可能	ブール値。クラスローディングの実行時トレース情報をオンにします。
doja.trace.exceptions	False	任意	可能	ブール値。例外の実行時トレース情報をオンにします。

5.5.2 エミュレータの起動

5.5.2.1 コマンド行からの起動

エミュレータは、スタンドアロン Java アプリケーションとしてコマンド行から実行できます。現在のディレクトリは、次のものであるとみなされます。

```
J2ME Wireless SDK_DIRECTORY\bin
```

5.5.2.2 コマンド行の構文

エミュレータを呼び出すコマンドの構文は、次のとおりです(すべてを 1 行に指定する)。

```
java -cp .. \lib\dojatest.zip;..\lib\kenv.zip -Dkvem.home=.. ["-D<property>=<value>"]*  
com.nttdocomo.dojaemu.tests.Tester jam <device name> <Descriptor file path>
```

表 5-2 に、com.nttdocomo.dojaemu.tests.Tester クラスのパラメータを示します。

表 5-2 コマンド行呼び出しのパラメータ

パラメータ	説明
["-D<property>=<value>"]* ¹	(任意)一連のプロパティ定義(表 5-1 を参照)。個々のプロパティ定義は、二重引用符で囲む必要があります。例: "-Dkvem.japanese_font=false"
<device name>	エミュレータが使用するデバイス定義の名前(たとえば、phone1bw や phone1color など)
<Descriptor file path>	実行するアプリケーションを記述した JAM ファイルの相対または絶対パス。例: ..\apps\spaceinv\bin\spaceinv.jam

1. アスタリスク「*」は、「説明」欄の例に示すように、この形式を繰り返し指定できることを表します。

第 6 章

J2ME Wireless SDK ユーティリティ

この章では、J2ME Wireless SDK に含まれている次のユーティリティについて説明します。

- 証明書管理ツール
- ScratchPad ビューア
- SJIS エンコーダ

6.1 証明書管理ツール

J2ME Wireless SDK に含まれている GUI ベースの証明書管理ツール (CMT)では、サーバとクライアント間の安全な HTTPS 通信に必要な X.509 証明書を管理します。CMT では、サーバ証明書とクライアント証明書をクライアント側で管理することができます。J2ME Wireless SDK でのクライアントはエミュレータです。エミュレータは、セキュアサーバとの HTTPS 接続を確立する Java アプリケーションを実行します。エミュレータは、Web サーバと通信する際に、実際のデバイスと同じように証明書を使用します。

証明書は<SDK>\ lib \ ssl \ clientcerts と<SDK>\ lib \ ssl \ servercerts に格納されます。

6.1.1 暗号化された証明書

HTTPS プロトコルでは、通信に参加する一部または全部の対象者が本人であるかどうかを検証するために、暗号化された証明書が使用されます。

HTTPS では、証明書は 2 つのグループに分けられます。

- サーバ証明書 - サーバが本物であることを証明する。
- クライアント証明書 - クライアントが本人であることを証明する。

これら 2 つの証明書はタイプごとに別個のリストに保持され、別々に管理されます。
なお、実際の Java 対応 i モード携帯電話ではクライアント証明書はサポートされません。

証明書の名前

証明書の名前には、次の 2 つの側面があります。

- 個々の証明書の名前は、それが属する証明書リストにおいて固有のものでなければなりません。この名前は、コード化された証明書自体に組み込まれるか、ユーザーが入力する必要があります。
- クライアント証明書とサーバ証明書は別々のファイルに保存されます。これらのファイルの名前は、<SDK>lib\ssl\clientcerts と <SDK>\ lib\ssl\servercerts にあらかじめ決められています。

6.1.1.1 証明書の取得

ここでは、新しい証明書の取得方法について説明します。

新しい証明書を取得する方法には、次の 2 つがあります。

- 自署証明書を生成する。
- 証明書を CA から取得する。

自署証明書の生成

自署証明書は、J2SE ディストリビューションに含まれている keytool ツールを使って生成できます(keytool に関するドキュメントについては、<http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html> を参照してください)。

証明書をインポートするためには、.cert という拡張子を持つファイルに証明書を格納する必要があります。

証明書を CA から取得

1. 証明書署名要求(CSR)を生成します。

CSR は、上で述べた keytool ツールを使って生成することもできます。さらに、多くの Web サーバには、CSR を生成する機能があります。詳細は、それぞれのマニュアルを参照してください。

2. 著名な CA に依頼して証明書または要求に署名してもらいます。

これによって、CSR は完全な証明書になります。この依頼は、電子メールやブラウザを通してオンラインで行うことができます。完全な証明書は、通常、RFC1421 に従って ASCII でコード化されています。

新しい証明書の挿入方法については、74 ページの「6.1.3.3 新しい証明書を証明書リストに挿入」を参照してください。

トラステッド証明書の検証

証明書をトラステッド証明書としてインポートする前に、証明書を注意深くチェックする必要があります。

まず、表示された証明書のフィンガープリントが、予期されたものであるか確認します。証明書のフィンガープリントは次のように表示されます。

Certificate Fingerprints:

MD5: 11:81:AD:92:C8:E5:0E:A2:01:2E:D4:7A:D7:5F:07:6F

SHA1: 20:B6:17:FA:EF:E5:55:8A:D0:71:1F:E8:D6:9D:C0:37:13:0E:5E:FE

次に、その証明書を送信した人にコンタクトし、その人が示すフィンガープリントとこのフィンガープリントを比較します。これらのフィンガープリントが一致すれば、証明書が伝送中に他のもので置き換えられているおそれはありません。

6.1.2 CMT の起動

Certificate Manager の起動方法は、使用する開発環境によって異なります。

- KToolBar - 次の図のように「編集」->「証明書管理」の順でメニュー項目を選択します。

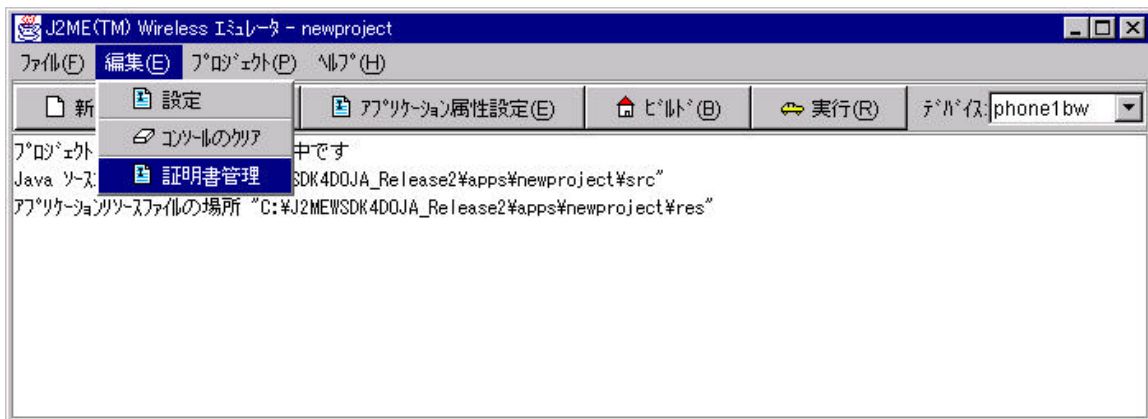


図 6-1 CMT を KToolBar から起動

- Forte for Java - 次の図のように、ADF を右クリックし、次に「ツール」->「証明書管理の参照」を選択します。

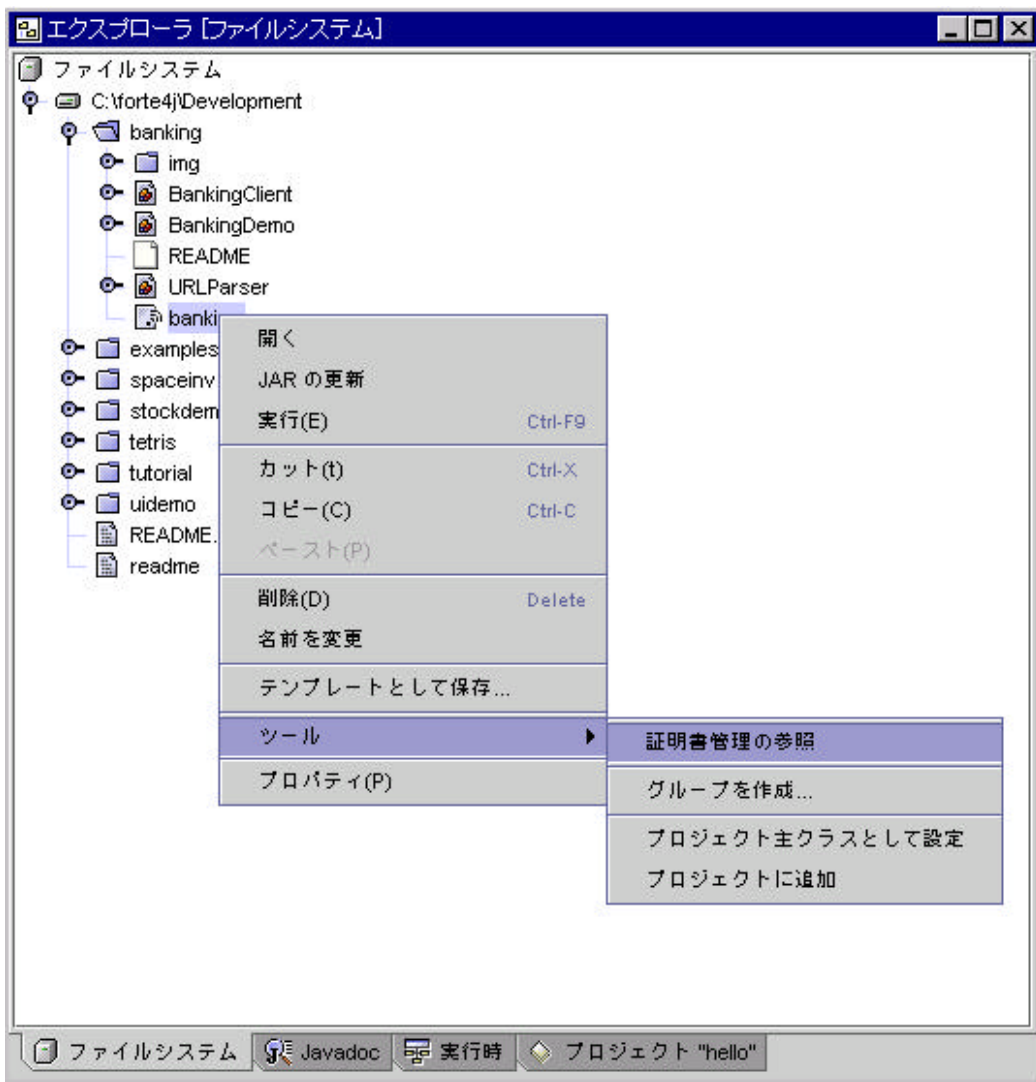


図 6-2 Forte から CMT を起動

- コマンド行 - 以下に示すように、コマンド行から CMT を起動します。

```
java.exe -Dkvm.home=<J2ME Wireless SDK_DIRECTORY>
```

```
-jar <J2ME Wireless SDK_DIRECTORY>\tools\certmgr\certmgr.jar
```

あるいは、J2ME Wireless の簡単なバッチファイル

<SDK_DIRECTORY>\bin\certmgr.bat を使ってこのツールを起動します。

6.1.3 CMT メインウィンドウ

CMT メインウィンドウが表示されます(図 6-3)。

このウィンドウでは、次のことができます。

- 個々の証明書を表示する
- クライアント証明書リストとサーバ証明書リストを表示する
- 新しい証明書を挿入する
- 証明書を削除する
- 証明書を保存する
- CMT を終了する

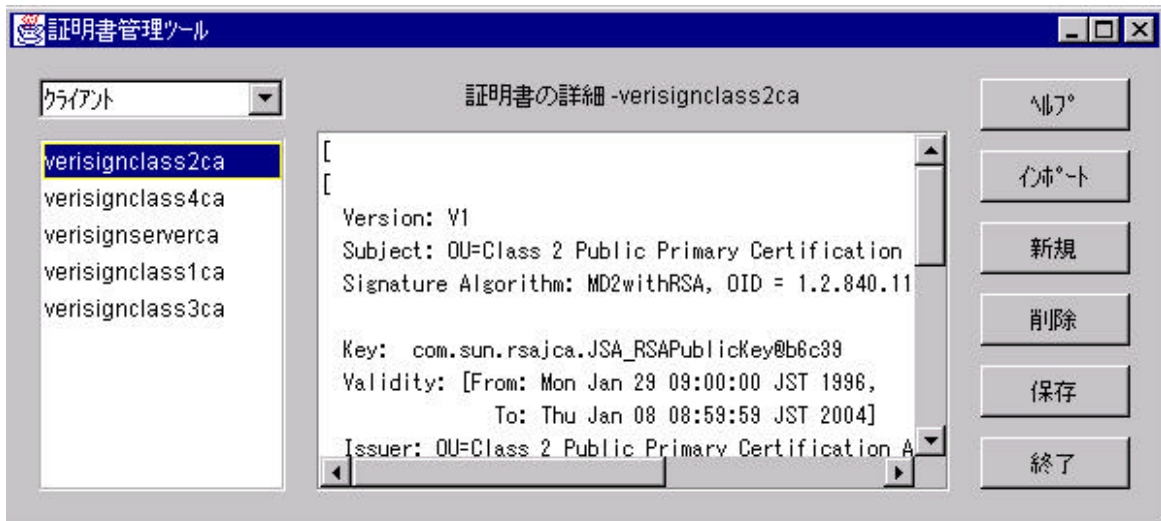


図 6-3 CMT ダイアログ

6.1.3.1 証明書の表示

図 6-3 の左ウィンドウには、CMT に格納されているクライアント証明書のリストが表示されます。

中央のウィンドウには、左のウィンドウで名前が選択された証明書の詳細が表示されます。

6.1.3.2 クライアント証明書リストとサーバ証明書リストの表示

上で述べたように、証明書はクライアント証明書とサーバ証明書という 2 つのリストに分けられます。CMT の画面では、同時に 1 つのリストだけを表示できます。図 6-3 には、クライアントの証明書リストが表示されています。プルダウンメニューの「クライアント」か「サーバー」を選択すれば、2 つのリストを交互に切り替えることができます。現在のリストに対して変更を行った後リストを切り替えると、変更の保存を求めるプロンプトが表示されます。

6.1.3.3 新しい証明書を証明書リストに挿入

ここでは、「6.1.1.1 証明書の取得」で述べたどちらかの方法を使って新しい証明書をすでに取得しているものとします。

さらに、自署証明書が拡張子.cert のファイルに格納されているものとします。

完全な証明書は、電子メールメッセージか Web ページに組み込まれた形で CA から提供されます。この証明書は、必要に応じて拡張子.cert のファイルに保存したり、電子メールや Web ページにそのまま残しておくこともできます。

CMT では、次の 2 つの方法で新しい証明書を証明書リストに挿入することができます。

- 新しい証明書をファイルからインポートする (証明書が拡張子.cert のファイルに格納されている場合)。
- 新しい証明書をダイアログボックスに貼り付ける (証明書が電子メールや Web ページに組み込まれている場合)。

必要に応じて都合のよい方を選択してください。

新しい証明書をファイルから挿入

拡張子.cert のファイルに格納されている証明書をインポートするには

1. 「インポート」ボタンをクリックします。

ファイルを選択するポップアップウィンドウが表示され、拡張子.cert のファイルが表示されます。

2. 必要なファイルを選択し、「開く」をクリックします。

証明書がリストに挿入されます。

新しい証明書をダイアログボックスに貼り付ける

電子メールメッセージや Web ページに組み込まれている新しい証明書を導入するには

1. 電子メールメッセージや Web ページに組み込まれている完全な証明書をシステムクリップボードにコピーします。
2. 現在アクティブな証明書リストのすべての項目を選択解除します。それには、Ctrl キーを押したまま、選択されている項目をクリックします。

これで CMT の「証明書の詳細」ウィンドウは空の状態になるはずです。

3. クリップボードの内容を CMT の「証明書の詳細」ウィンドウに貼り付けます。
4. 「新規」ボタンをクリックします。

「証明書の詳細」ウィンドウに表示されている証明書が証明書リストに挿入されます。証明書の名前は、CMT によってその証明書から抽出されます。

新しい証明書の別名を指定

上のどちらの方法でも、CMT が証明書に対する別名を見つけられない場合は、新しい証明書の別名入力を要求するプロンプトが表示されます(図 6-4)。

1. 新しい証明書固有の名前を入力し、「了解」をクリックします。

証明書がリストに挿入されます。

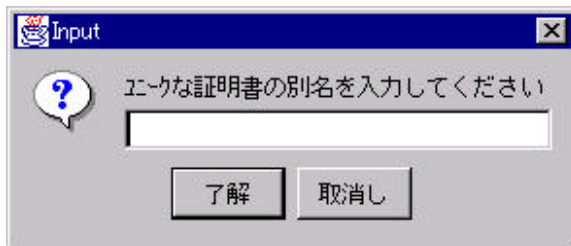


図 6-4 新しい証明書の名前の指定

6.1.3.4 証明書の削除

表示されているリストから証明書を削除するには、削除する証明書を選択し、「削除」ボタンをクリックします。選択した項目が証明書リストから削除されます。

証明書を間違えて削除してしまった場合には、その証明書の .cert ファイルからインポートし直すことができます。このファイルがない場合は、変更を保存せずに CMT を終了すれば、証明書は削除されません。

6.1.3.5 証明書の保存

新しい証明書の追加や証明書の削除といった変更を証明書固有の格納場所に保存します。

6.1.3.6 終了

CMT を終了するには、「終了」をクリックします。変更が行われている場合は、保存を求めるプロンプトが表示されます。

6.2 ScratchPad ビューア

ScratchPad ビューア (SPV)は、アプリケーションの ScratchPad ファイルを 16 進数でダンプする非常にシンプルなユーティリティです。完全に Java で書かれたこのツールは、`<J2ME Wireless SDK_DIRECTORY>\tools\spview\spview.jar` ファイルにあります。

ScratchPad ビューアを使えば、アプリケーションからデータが ScratchPad に適切に書き込まれているかどうかを確認できます。

6.2.1 ScratchPad ファイル

ScratchPad は、実際のデバイスでは不揮発性の記憶領域ですが、エミュレータでは 1 つのファイルです。ScratchPad には、文字列以外に i モード対応 Java で使用可能な基本タイプのデータを格納できるため、ScratchPad の内容を通常のテキストビューアで表示することはできません。

エミュレータは、ScratchPad ファイルを、Windows の %TEMP% 環境変数が指すディレクトリに格納します。ScratchPad ファイルの拡張子は .scr です。TEMP 変数は Windows が自動的に設定するため、通常、ユーザーが変更する必要はありません。

6.2.2 SPV の起動

SPV の起動方法は、使用する開発環境によって異なります。

- KToolBar - 次図のように「プロジェクト」->「スクラッチパッドの表示」の順でメニュー項目を選択します。



図 6-5 KToolBar から SPV を起動

- Forte for Java - 次図のように、ADF を右クリックし、「ツール」->「スクラッチパッドの参照」を選択します。

注 「スクラッチパッドの参照」のメニュー項目は、アプリケーションがスクラッチパッドファイルを持っている場合にのみ使用可能になります。そうでない場合にはこのメニュー項目は「ツール」メニュー上に表示されません。

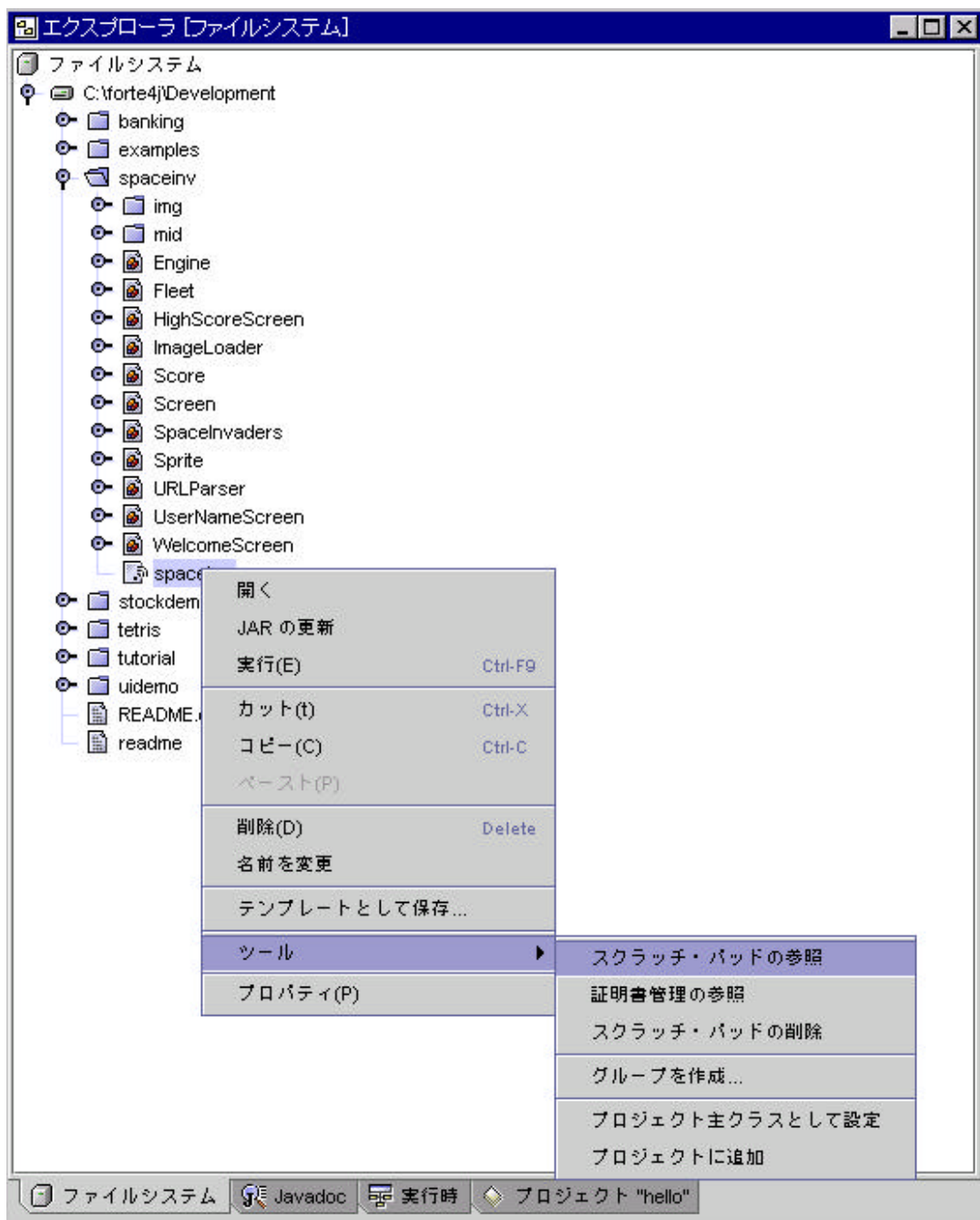


図 6-6 Forte for Java から SPV を起動

- コマンド行 - 次図のようにコマンド行から SPV を起動します。

```
java -jar <J2ME Wireless SDK_DIRECTORY>\tools\spview\spview.jar <file-name>  
<file-name>引数は任意です。
```

SPV を起動する簡単なバッチファイルが<J2ME Wireless SDK_DIRECTORY>\bin\spview.bat で提供されています。

コマンド行にファイル名を指定しない場合は、SPV は「ファイル選択」ダイアログを開きます(図 6-7)。

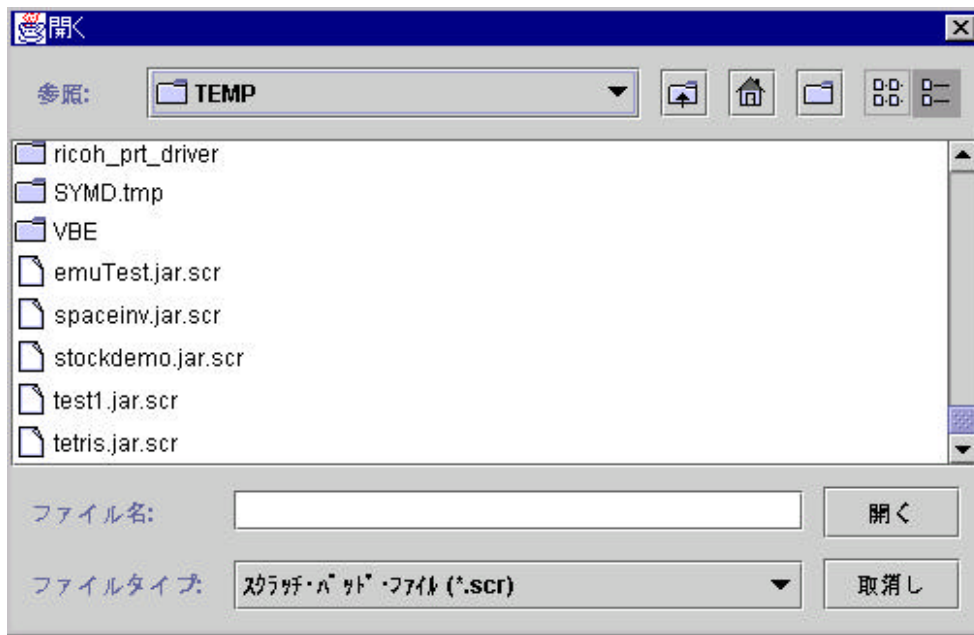


図 6-7 ファイル選択ダイアログ

デフォルトで SPV は、Windows の %TEMP% 環境変数が指すディレクトリを探します。このディレクトリは、エミュレータが ScratchPad ファイルを格納するところです。

6.2.3 Scratchpad 内容の表示

コマンド行又は「ファイル選択」ダイアログを使って ScratchPad をロードすると、ファイルの内容が 16 進数で表示されます(図 6-8)。

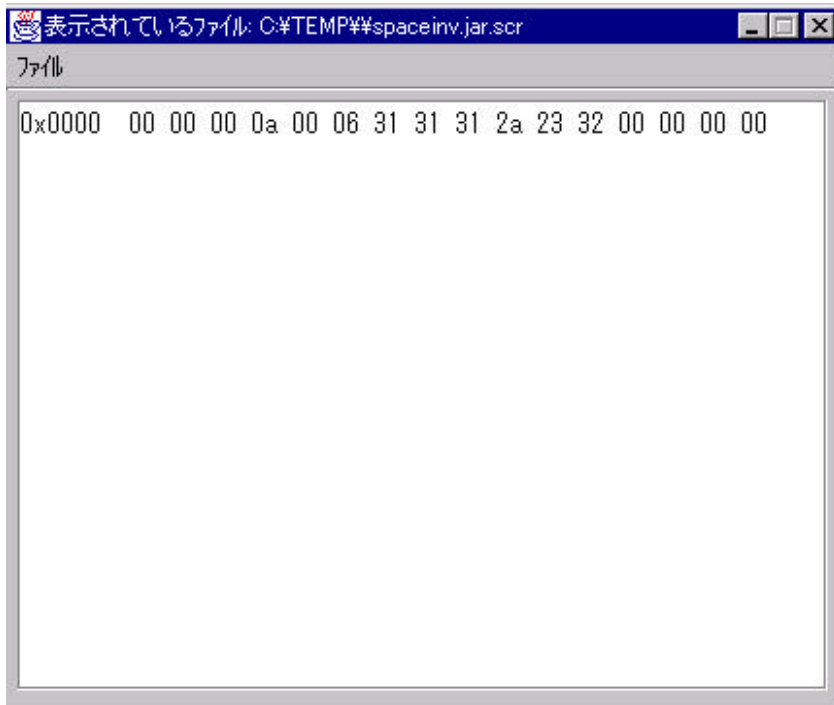


図 6-8 ファイル内容の表示

矢印キーを使えば、ScratchPad ファイルの内容を簡単にスクロールできます。ただし、ファイルの編集はできません。

6.2.4 別の ScratchPad ファイルの表示

別の ScratchPad ファイルを表示するには、「ファイル」->「開く」の順でメニュー項目を選択します(図 6-9)。「ファイルの選択」ダイアログが表示されますので、新しいファイル名を入力します。

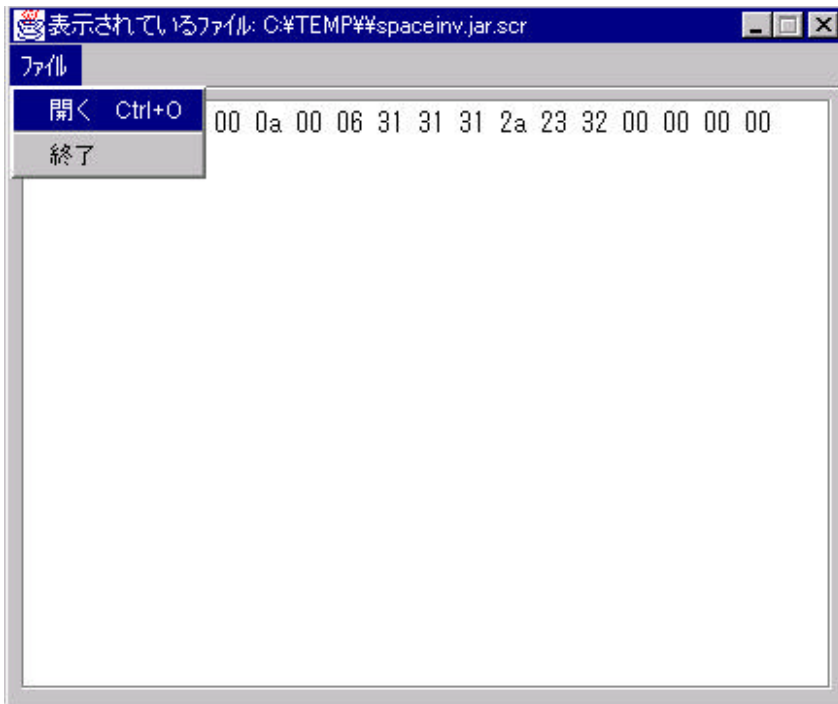


図 6-9 File メニュー

6.2.5 SPV の終了

SPV を終了するには、「ファイル」->「終了」の順でメニュー項目を選択します。

6.2.6 アプリケーション ScratchPad ファイルの削除

テストの間にアプリケーションの ScratchPad データを削除する必要がある場合は、KToolBar や Forte for Java を使って ScratchPad ファイルを削除することができます。

- KToolBar - 次図のように「プロジェクト」->「スクラッチパッドの削除」の順でメニュー項目を選択します。



図 6-10 KToolBar で ScratchPadを削除

- Forte for Java - 次図のように、ADF を右クリックし、「ツール」->「スクラッチ・パッドの削除」を選択します。

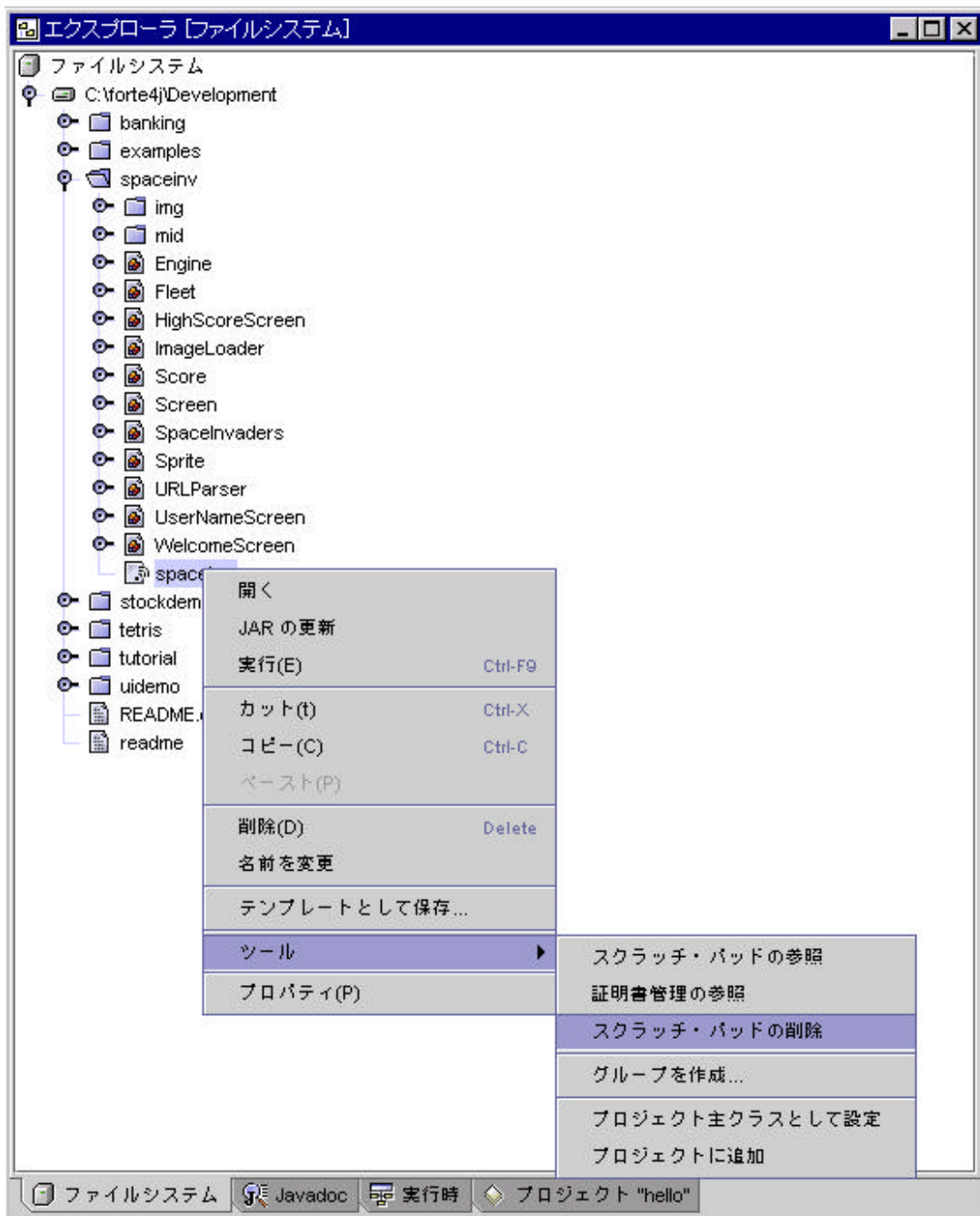


図 6-11 Forte for Java で ScratchPad ファイルを削除

6.3 SJIS エンコーダ

i モード対応 Java アプリケーションの Java ソースファイルには、SJIS_i(NTT ドコモが SJIS を拡張して定義した絵文字を含むコードセット)でコード化された文字列が含まれていることがあります。

Java ソースファイルのコンパイルは、Forte for Java、Ktoolbar、またはコマンド行を使って行うことができます。Forte for Java と Ktoolbar では、SJIS_i コードを自動的に認識するようにコンパイラが設定されているため、ユーザーは何もする必要はありません。

.java ソースをコマンド行からコンパイルする場合は、次のパラメータを追加で javac に渡す必要があります。

```
-encoding SJIS_i -JXbootclasspath/a:C:\KVMSDK4DOJA\lib\dojaconv.jar
```

ファイル dojaconv.jar には、SJIS_i コードを認識する特別なコード変換プログラムが含まれています。javac プログラムの制約のため、-J-Xbootclasspath/a:オプションを使ってこの JAR ファイルを javac に渡す必要があります。たとえば、次のように指定します。

```
javac -encoding SJIS_i -JXbootclasspath/a:C:\J2MEWSDK4DOJA\lib\dojaconv.jar -bootclasspath C:\J2MEWSDK4DOJA\lib\dojaapi.jar \MyTest.java
```

詳しい例については、J2ME Wireless SDK がインストールされているディレクトリの apps\uidemo\build.bat ファイルを参照してください。

SJIS_i コードの文字を含む .java ソースファイルを使用するサンプルプログラムについては、J2ME Wireless SDK がインストールされているディレクトリの apps\uidemo\src\uidemo\JapaneseDemo.java ファイルを参照してください。