

DoJa2.1API 向け
i a ppli Development Kit
ユーザーズガイド

平成 14 年 12 月 17 日
株式会社 N T T ドコモ

1. はじめに	6
1.1. 「iappli Development Kit」の開発環境	6
1.1.1. 「iappliTool」開発環境	6
1.1.2. Forte for Java 開発環境	7
1.2. エミュレータの概要	8
1.2.1. エミュレータの利点	8
1.2.2. エミュレーションの範囲	8
1.3. 「iappli Development Kit」によるアプリケーションの開発	10
2. 「iappli Development Kit」のインストール	12
3. 「iappliTool」の起動	14
4. エミュレータの設定	17
4.1. ネットワークの設定	17
4.2. プロキシサーバの設定	19
4.3. エミュレーション端末の選択	19
4.4. エミュレーション端末の操作	22
4.4. コンパイラ (javac) の選択	24
5. プロジェクト	26
5.1. プロジェクトの作成～起動	26
5.2. 既存プロジェクトの読み込み	26
5.3. アプリケーションディスクリプタファイル (ADF) の編集	27
5.3.1. LastModified 項目の表記方法	32
5.4. プロジェクトの新規作成	32
5.5. プロジェクトのビルドと通常起動	34
5.6. 初回起動	37
5.7. タイマ起動	38
5.8. 外部機器起動	39
6. コンソールと「iappliTool」の終了	41
7. 他のアプリケーション起動方法	42
7.1. 待受け起動	42
7.1.1. 待ち受けアプリケーション	42
7.1.2. 待受け起動手順	43
7.2. メーラ起動	46
7.2.1. メーラ起動手順	46
7.3. ブラウザ起動	49
7.3.1. ブラウザ起動手順	49

7.4. 各種起動方法による画面遷移	54
8. ファイルの記述方法	56
8.1. 「*.mail」ファイルの記述方法	56
8.2. 「HTML」ファイルの記述方法	56
8.3. 「JAM.properties」ファイルの記述方法	57
8.4. ファイルの記述具体例	57
9. スクラッチパッド管理	58
9.1. スクラッチパッド管理ツール	58
9.2. スクラッチパッドファイル	58
9.3. スクラッチパッド管理ツールの起動	58
9.4. スクラッチパッドの内容表示	59
9.5. スクラッチパッドの削除	60
9.6. スクラッチパッド管理ツールの終了	60
10. 証明書の管理	61
11. OBEX、カメラ API、ポインティングデバイスの利用	64
11.1. OBEX の利用	64
11.1.1. TCP/IP のみのリリース向け	64
11.1.2. TCP/IP、IrDA の両方を含むリリース向け	64
11.2. カメラ API の使用	64
11.3. ポインティングデバイス API の使用	65
12. Forte for Java Community Edition の使用	67
12.1. はじめに	67
12.1.1. Forte for Java	67
12.2. 「J2ME Module for i α ppli」のインストール	67
12.2.1. 始める前に	67
12.2.2. 「J2ME Module for i α ppli」のインストール	68
12.3. 新規アプリケーションの作成	69
12.3.1. パッケージの作成	69
12.3.2. アプリケーション・ディスクリプタ・ファイル (ADF) の作成	71
12.3.3. ソースファイルの作成	73
12.3.4. ADF の設定	76
12.3.5. jar ファイルの内容の設定	78
12.3.6. アプリケーションの構築と実行	79
12.4. エミュレータの設定	80
12.4.1. デバイスの設定	80
12.4.2. プロキシサーバの設定	84
12.4.3. ログメッセージのトレース	85

付録. エミュレーション端末の設定	86
1. コメント行	86
2. エミュレータウィンドウの設定	86
3. LCD 領域の設定	86
4. フォントの設定	86
5. ボタン設定	87
6. 例	87

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。NTTドコモ（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。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、Java、JDK、Forte、J2ME および J2SE 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。サンのロゴマークは、米国 Sun Microsystems, Inc. の登録商標です。

「i モード／アイモード」は NTT ドコモの登録商標、「i アプリ／i appli／アイアプリ」は NTT ドコモの商標です。

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。

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 2000-2002 NTTDoCoMo, Inc. 2-11-1 Nagata-cho, Chiyoda-ku, Tokyo, 100-6150, Japan.

1. はじめに

このマニュアルでは、「iα ppli Development Kit」のインストール、設定及び使用方法について説明します。

このマニュアルで述べる「iα ppli Development Kit」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 Doja2.0API の知識が必要です。

本章では、「iα ppli Development Kit」についての概要を説明します。本章の内容は、次の通りです:

- 「iα ppli Development Kit」の開発環境
- エミュレータの概要
- 「iα ppli Development Kit」によるアプリケーションの開発

1.1. 「iα ppli Development Kit」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、「iα ppliTool」開発環境と Forte for Java 開発環境があります。

1.1.1. 「iα ppliTool」開発環境

「iα ppliTool」開発環境の画面は、図 1-1 のように表示されます。

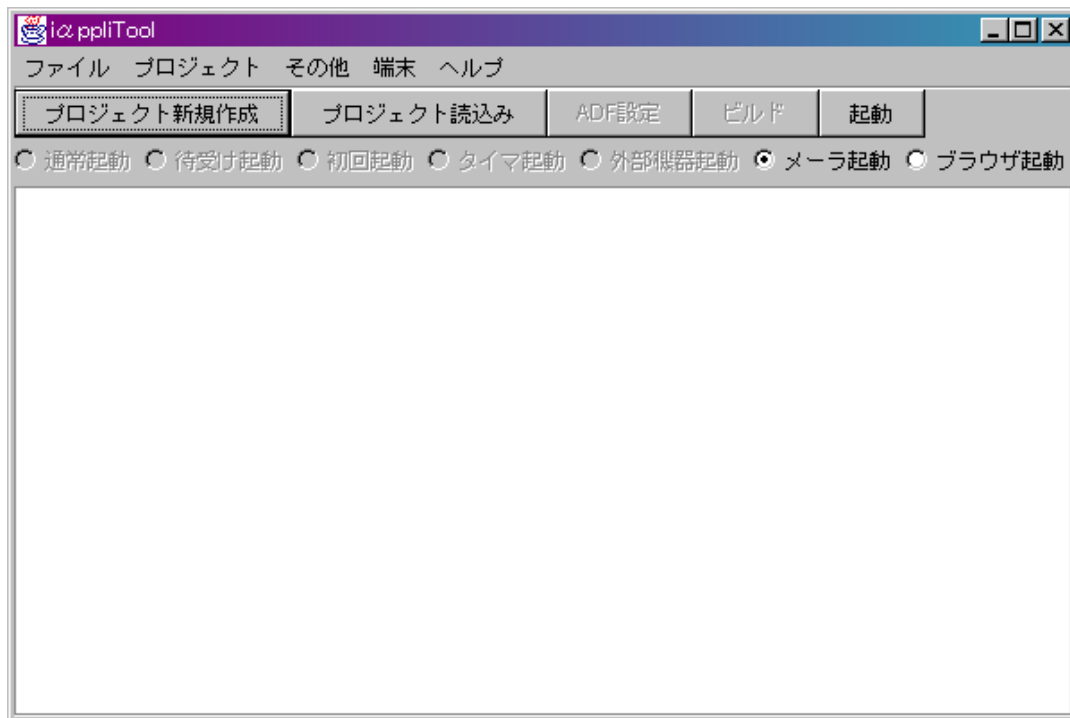


図 1-1 「iα ppliTool」画面

「iα ppliTool」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。

開発サイクルについては、「1.4. 「iα ppliTool」によるアプリケーションの開発」を参照してください。

1.1.2. 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> を参照してください）。

「iα ppli Development Kit」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を提供します。Forte for Java 開発環境では、Java アプリケーションの編集、コンパイル、ビルド及びエミュレータによる実行を全て行うことができます。詳細は「12. Forte for Java Community Edition の使用」を参照してください。

1.2. エミュレータの概要

エミュレータは、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 上で検証できます。

1.2.1. 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に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 上からエミュレートできます。

1.2.2. 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、実際の電話機の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 1 つです。

アプリケーションマネージャに関する動作のうち実際の電話機と異なる動作として以下があります。

- アプリケーション、スクラッチパッド、ADF のサイズに制限はありません。
 - アプリケーションのダウンロードや更新の機能はありません。ただし、起動種別設定機能を使用することにより IApplication.LAUNCHED_AFTER_DOWNLOAD を発生させるテストが可能です。
 - ネットワーク接続可否の設定機能はありません。
 - タイマ起動を設定する機能はありません。ただし、起動種別設定機能を使用することにより IApplication.LAUNCHED_FROM_TIMER を発生させるテストが可能です。
- アプリケーションディスクリプタファイル (ADF)

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いて実際に処理される項目と処理されない項目があります。しかし、全ての項目において値の検証は行っています。従って、アプリケーションとして特に設定する必要のない項目については、値を設定しないことを推奨します。アプリケーションディスクリプタファイルについての詳細は「5.3. アプリケーションディスクリプタファイル (ADF) の編集」を参照してください。

- 携帯電話固有の機能

エミュレータでは、携帯電話固有の機能のうちバックライトやバイブレータの状態表示など一部の機能をエミュレートします。しかし、携帯電話固有の多くの機能（電話帳や音の設定など）はエミュレートされません。

携帯電話固有の機能に関して、実際の電話機と異なる動作として以下があります。

- バックライトやバイブレータの状態はエミュレータ上のピクト領域に表示されます。
- 音量やマナーモードを設定する機能はありません。
- 電源の ON/OFF や自動電源 ON/OFF の機能はありません。
- カメラ機能（静止画、動画の撮影機能）はありません。

- 待ち受けアプリケーション

エミュレータでは待ち受けアプリケーションのライフサイクルを実現していますが、電話機に待ち受けアプリケーションを登録するための設定機能は提供しません。

待ち受けアプリケーションに関して、実際の電話機と異なる動作として以下があります。

- 待ち受け画面（待ち受け画像など）の設定/表示機能はありません。
- 待ち受けアプリケーションを電話機に設定する機能はありません。待ち受けアプリケーションとしての起動はエミュレータの起動オプションにより指定します。
- 待ち受けアプリケーションが休眠状態の際の省電力機能はエミュレートしません。

- ネイティブアプリケーション連携

エミュレータでは、ネイティブアプリケーション連携のうちブラウザ、メーラ、音声通話の各機能との連携をエミュレートする環境を提供しますが、外部機器連携をエミュレートする環境は提供しません。

ネイティブアプリケーション連携に関して、実際の電話機と異なる動作として以下があります。

- 外部機器からアプリケーションを起動する機能はありません。ただし、起動種別設定機能を使用することにより IApplication.LAUNCHED_FROM_EXT を発生させるテストが可能です。
- アプリケーションからブラウザを起動する API（IApplication クラスの launch()）を呼び出した場合、確認ダイアログの表示まで行われますが、ブラウザの起動は行いません。
- ブラウザから i アプリを起動する際、i モード通信可否の判定は行いません。また、以下の点に注意してください：
 - <A>タグに ista 属性および ijam 属性があっても無視されます。また utn 属性が指定される場合も同様に無視されます。
 - 1 つの HTML 内には、<OBJECT>タグと<A>タグをこの順序で対にして記述してください。

- 対応関係にある<OBJECT>タグと<A>タグとの間には、他の<OBJECT>タグを書かないでください。
- 1つのHTML内での各<A>タグにおける href 属性値は、異なる文字列として記述してください。
- <OBJECT>タグでは、data 属性だけが意味を持ち、id、type、declare 属性は意味を持ちません。
- 最低限、<OBJECT>タグ内においては、data 属性とその値（ADF の URL）、および、<A>タグ内においては、href 属性とその値を記述してください。
- メーラから i アプリを起動する際、以下の点に注意してください：
 - "--B:A"以降の記述については、規定のフォーマットに従っていない場合は、全て無視します。
 - メール本文は、HTML フォーマットに対応しています。
- HTTP を使用したアプリケーション
 - Connection: keep-alive は、常に設定されて要求を発行します。

1.3. 「i appli Development Kit」によるアプリケーションの開発

本節では、上述の「i appli Tool」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。

「i appli Development Kit」でサポートする開発サイクルには、次のフェーズがあります：

1. テキストエディタを使用して java ファイルを編集する（「i appli Development Kit」の範囲には含まれません）。
2. ビルド
 - 2.1. コンパイル

Java 2 Standard Edition SDK コンパイラを使用して、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。
 - 2.2. バイトコードの検証

事前検証ツールを使用して、Java クラスの事前検証を行うと共に、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。
 - 2.3. パッケージング

Java 2 Standard Edition SDK の jar ツールを使用して、事前検証された全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。
 - 2.4. 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。
3. 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てエミュレータを使用してアプリケーションを実行します。

上述の開発サイクルを図示すると、図 1-2 のようになります。

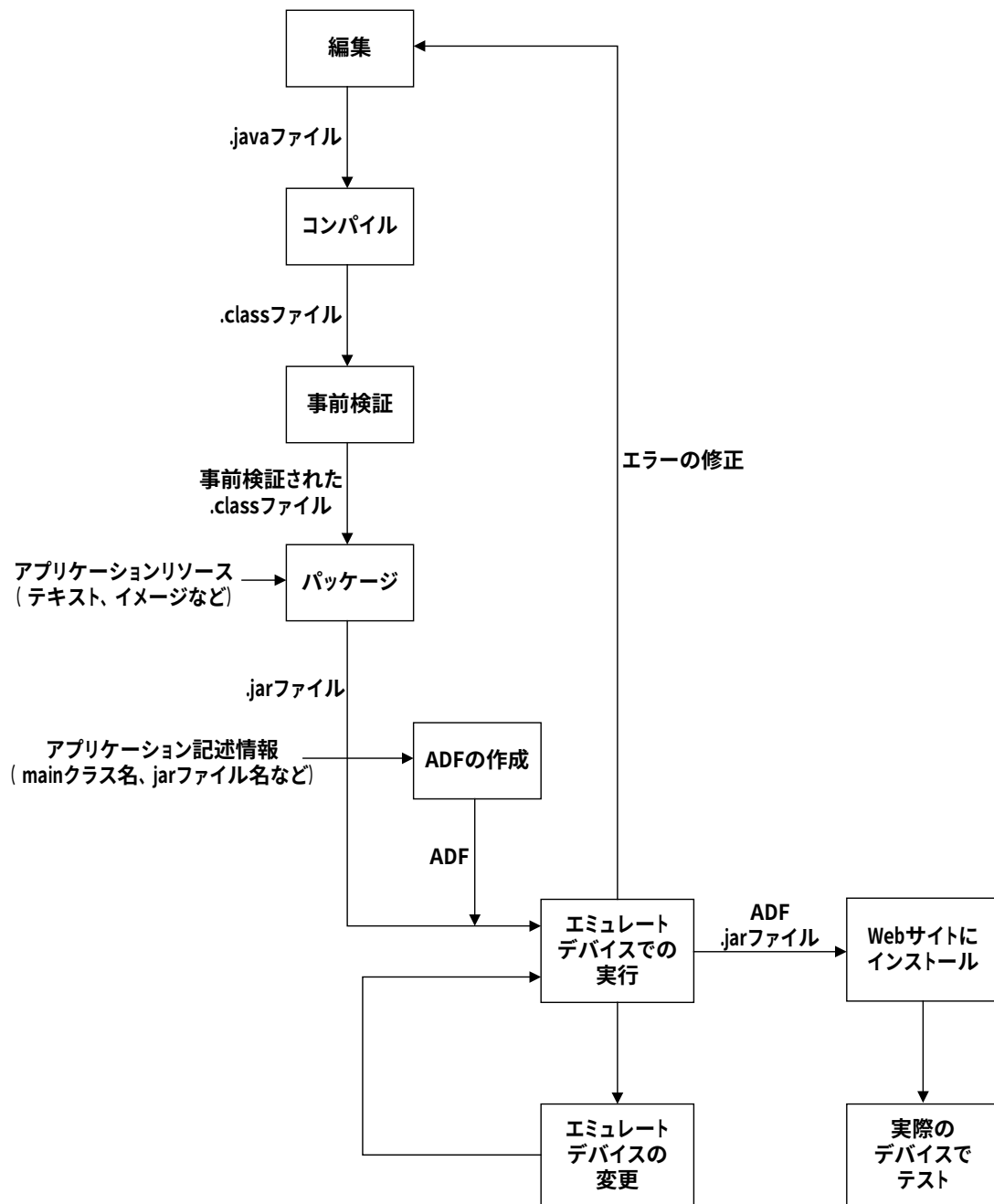


図 1-2 アプリケーション開発サイクル

2. 「iα ppli Development Kit」のインストール

「iα ppli Development Kit」をインストールする前に、「iα ppli Development Kit」の以前のバージョンが既にインストールされている場合には、これをアンインストールしてください。また、インストール（アンインストール）する際、Administrator 権限のあるユーザでログインして行ってください。「iα ppli Development Kit」をインストールするには

1. 解凍したディレクトリ内の「SETUP.EXE」をダブルクリックします。
2. インストーラのプロンプトに従ってインストールしてください^[1]。

「iα ppli Development Kit」インストール終了後、「絵文字（外字）」のインストールを行う必要があります。「絵文字（外字）」をインストールするには

1. 「iα ppli Development Kit」をインストールしたディレクトリの「gaiji」ディレクトリにある「ChangeEUDC.exe」を実行します（図 2-1）。

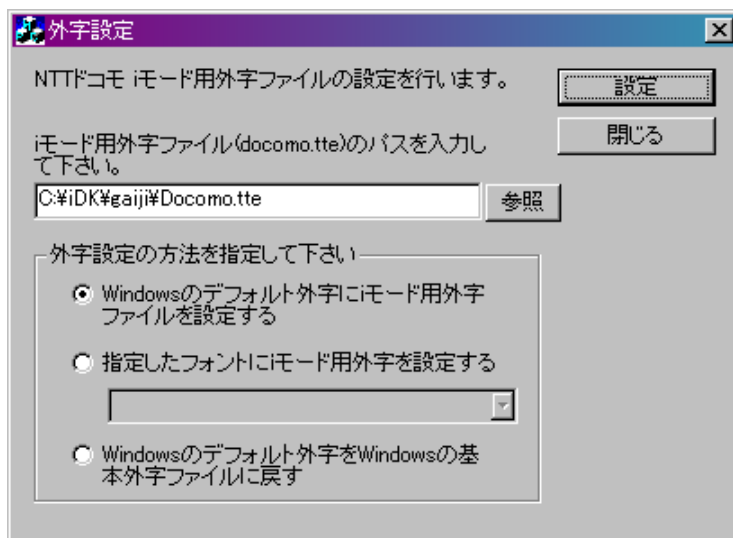


図 2-1 「ChangeEUDC.exe」の実行

2. 「Docomo.tte（i モード用外字ファイル）」のパスを指定し（デフォルトは、「ChangeEUDC.exe」のあるディレクトリ）、「外字設定の方法」を選択して（デフォルトは、「Windows のデフォルト外字に i モード用外字ファイルを設定する」）、「設定」ボタンを押します（図 2-2）。



図 2-2 絵文字（外字）のインストール終了の確認

^[1] インストールのディレクトリを指定する際「スペース（空白）」及び「2 バイト文字（全角文字）」を含まないようにしてください。これらの文字が含まれているとディレクトリを正しく扱うことができません。

3. 一度、インストールした絵文字（外字）をアンインストールするには、図 2-1 の「外字設定の方法」の一番下「Windows のデフォルト外字を Windows の基本外字ファイルに戻す」を選択して「設定」ボタンを押します。

また、「i ɑ ppli Development Kit」をアンインストールするには、「スタート」->「プログラム」->「i ɑ ppli Development Kit」->「Remove i ɑ ppli Development Kit」メニューを選択してください。

3. 「iαppliTool」の起動

「iαppliTool」を起動する前に、HTTPで通信を行うアプリケーションを使用する場合、以下の設定を行ってください。

- Windows のコントロールパネル->インターネット オプションの「詳細設定」タブ内「HTTP1.1 設定」の「HTTP1.1 を使用する」項目のチェックボックスのチェックをはずしてください（図 3-1）。

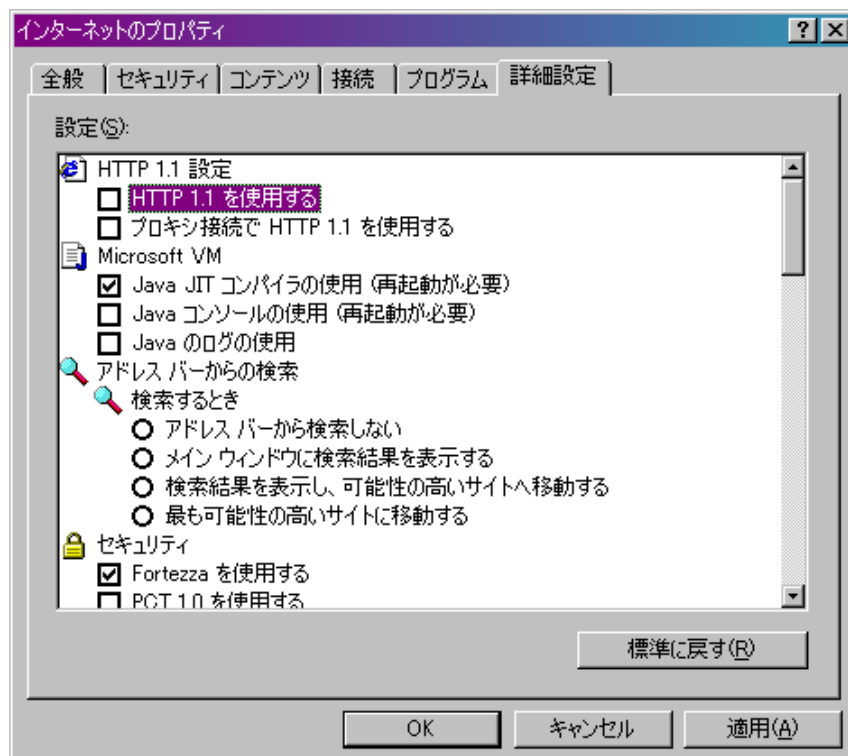


図 3-1 HTTP アプリケーションを使用する際の設定

「iαppliTool」を起動するには、「スタート」->「プログラム」->「iαppli Development Kit」->「iαppliTool」メニューを選択します^[2]。

図 3-2、図 3-3 に示す最初の画面が表示されます。

図 3-2 は、「iαppliTool」のメインウィンドウであり、ソースコードのコンパイル、ADF の編集、ネットワークの設定、アプリケーションの実行などを行います。

図 3-3 は、「エミュレータ端末」ウィンドウであり、実際のアプリケーションを表示するウィンドウです。この画面がエミュレータの待受け画面となります。

^[2] 「iαppliTool」起動時に JDK がインストールされている必要があります。また、JDK がインストールされていても、プロジェクトのビルド（「5.5. プロジェクトのビルドと通常起動」を参照）を行う際、「iαppliTool」内で JDK がインストールされているディレクトリを正しく取得できない場合があります。この場合は、「環境変数 JDK_HOME に JDK をインストールしたディレクトリを設定してください」というメッセージが出力されますので、環境変数 JDK_HOME に JDK をインストールしたディレクトリを設定した後、改めて「iαppliTool」を起動してください。

この時点で図 3-2 では、次の操作が可能となります:

- ネットワークの設定をする（「4.1. ネットワークの設定」を参照）
- エミュレーション端末の選択（「4.3. エミュレーション端末の選択」を参照）
- コンパイラ（javac）の選択（「4.4. コンパイラ（javac）の選択」を参照）
- 既存プロジェクトを読み込む（「5.2. 既存プロジェクトの読み込み」を参照）
- 新規プロジェクトを作成する（「5.4. プロジェクトの新規作成」を参照）
- メーラ起動（「7.2. メーラ起動」を参照）
- ブラウザ起動（「7.3. ブラウザ起動」を参照）

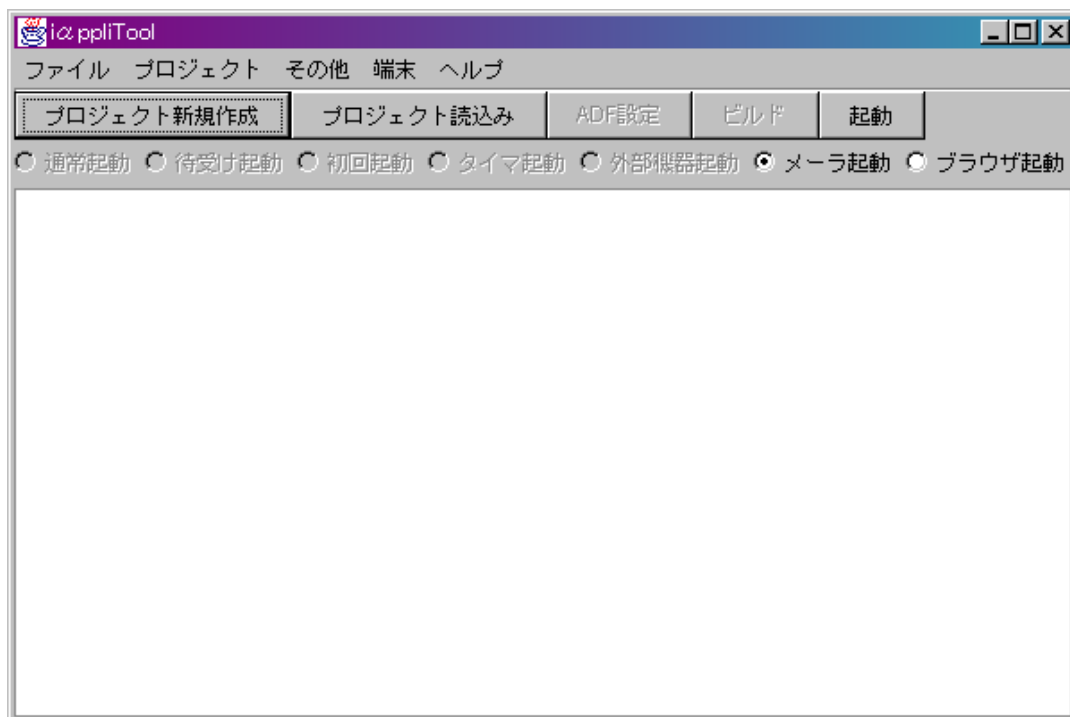


図 3-2 「iα ppliTool」のメインウインドウ



図 3-3 エミュレータ端末ウインドウ

4. エミュレータの設定

4.1. ネットワークの設定

1. メニューから「その他」→「ネットワーク設定...」を選択します（図 4-1）。

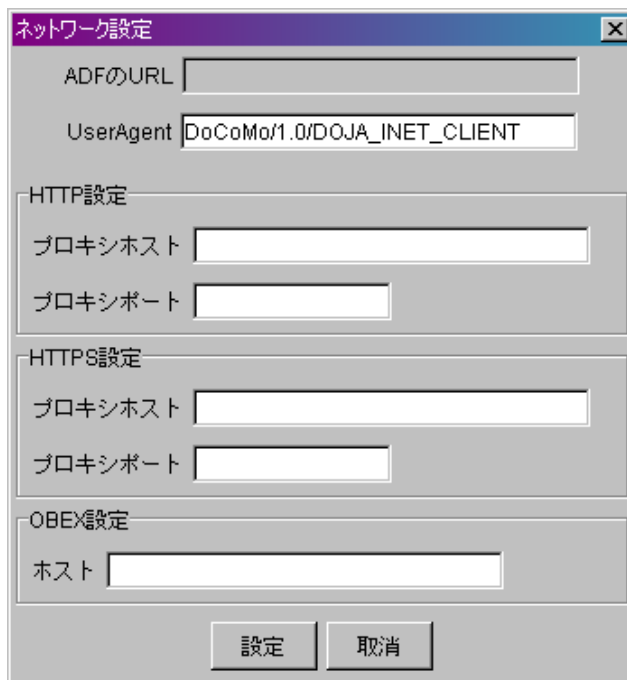


図 4-1 ネットワークの設定ウインドウ

2. 各設定項目を設定します。

設定項目は表 4-1 に示す通りです。

表 4-1 エミュレータの設定項目

設定項目	説明	デフォルト
------	----	-------

ADF の URL		PackageURL タグに相対 URL が指定された場合、本 URL がベースの URL となります。アプリケーションはこの PackageURL のホストとポートのみ通信可能です。 このフィールドに指定する URL は、ADF の PackageURL タグのフォーマットと同様です。 また、このフィールドは、アプリケーションごとに設定可能なので、「iα ppliTool」でプロジェクトが読み込まれている場合に設定可能です。さらに、このフィールドに設定した URL は、ファイルに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし
UserAgent		HTTP の「UserAgent」ヘッダ。 このフィールドに設定した「UserAgent」ヘッダは、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	DoCoMo/1.0/D OJA_INET_CLI ENT
HTTP 設定	プロキシ ホスト	アプリケーション及び「iα ppliTool」のブラウザが Web サーバとの通信に使用する HTTP プロキシサーバのホスト名。「4.2. プロキシサーバの設定」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし
	プロキシ ポート	アプリケーション及び「iα ppliTool」のブラウザが 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番号。 「4.2. プロキシサーバの設定」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし
HTTPS 設定	プロキシ ホスト	アプリケーションが Web サーバとの通信に使用する HTTPS プロキシサーバのホスト名。「4.2. プロキシサーバの設定」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし

	プロキシ 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番号。「4.2. プロキシサーバの設定」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし
OBEX 設定	ホスト	アプリケーションが OBEX を使用して通信する際の IP アドレス。OBEX over TCP を用いる場合に、接続先ノードの IP アドレスを指定します。指定がない場合は、127.0.0.1（ローカルホスト）に接続を確認する試みをします。なお、本設定は、ObexClientConnection クラスの connect メソッドで意味をなします。「11. OBEX の利用」を参照。 また、このフィールドに設定したホスト名は、レジストリに保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。	なし

4.2. プロキシサーバの設定

エミュレータのもとで動作する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、エミュレータの設定でプロキシサーバの情報を設定する必要があります。アプリケーションは、HTTP か HTTPS を使用して 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 のホストとポートは、図 4-1 のエミュレータの設定画面で設定します。

なお、HTTPS の際に使用する証明書の管理については「10. 証明書の管理」を参照してください。

4.3. エミュレーション端末の選択

1. メニューから「端末」を選択し、その中からエミュレートする端末をチェックします（図 4-2）。

このメニューには、ビルドしたアプリケーションに対して使用できる端末が表示されます。なお、ここで選択した端末は、レジストリに保存されるので、次回「iα ppliTool」を起動したとき最初に選択されている端末として使用されます。

また、各端末のイメージを図 4-3, 図 4-4 に示します。

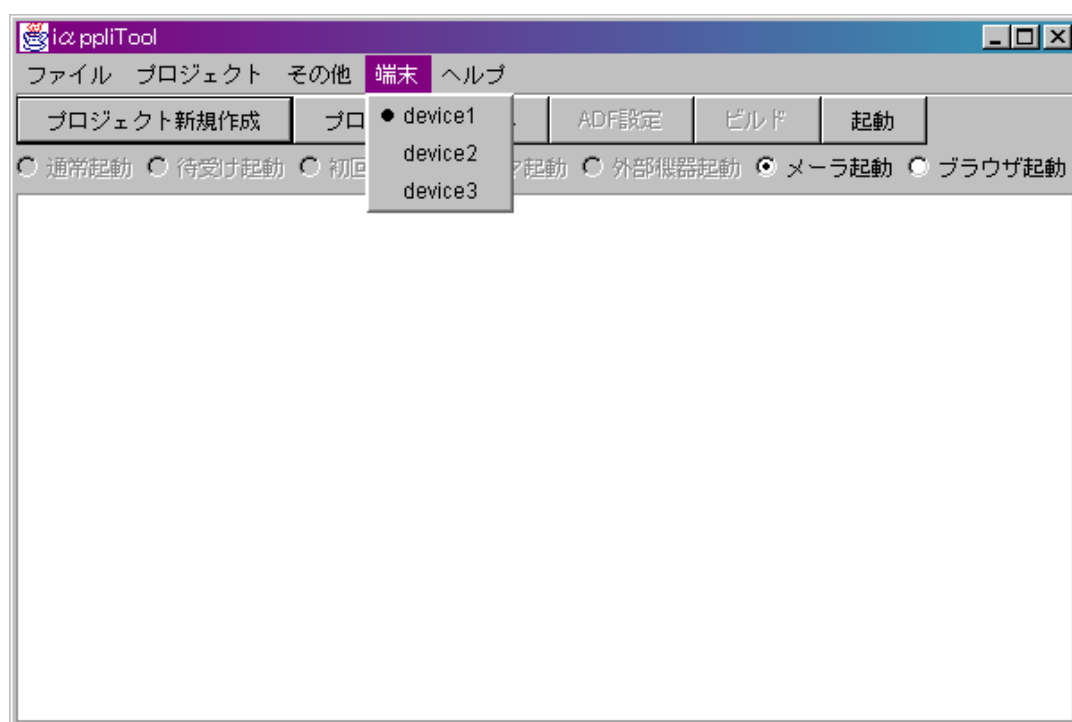


図 4-2 端末の選択



図 4-3 エミュレーション端末（左から「device1」、「device2」）



図 4-4 エミュレーション端末（「device3」）

4.4. エミュレーション端末の操作

図 4-5 を用いてエミュレーション端末の基本操作を説明します。

なお、基本操作はどのエミュレータ端末でも同様です。

①. 発信ボタン

- 通常のアプリケーションおよび待ち受けアプリケーション動作中に押すことにより、通常の通話を開始した状態になり、アプリケーションはサスペンドします。

②. 終話ボタン

- ①の「発信」ボタンを押した状態（通話中）のときに押すことにより、通話が終了し、サスペンドしていたアプリケーションがレジュームします。
- 上記の状態以外のときに押すことにより、通常のアプリケーションおよび待ち受けアプリケーションを強制終了させます。

③. 待受ボタン

- 待ち受けアプリケーションが、「休眠状態」あるいは「非活性化状態」のときに押すことにより「活性化状態」になります（「7.1.1. 待ち受けアプリケーション」参照）。このボタンは、キーボードの「D」キーに対応しています。

④. 決定ボタン

- このボタンは、キーボードの「Enter（リターン）」キーに対応しています。

⑤. 方向ボタン（上下左右ボタン）

- このボタンは、キーボードの「上下左右（↑↓←→）」キーにそれぞれ対応しています。

⑥. 数字ボタン（0～9 ボタン）

- このボタンは、キーボードの「0～9」キーにそれぞれ対応しています。

⑦. LCD

- アプリケーション起動中に LCD 上でマウスを右クリックすることで、「（端末の）折りたたみ状態」、「メールの受信状態」および「メッセージ（メッセージフリー、メッセージリクエスト）の受信状態」の参照・変更ができます。この3つの状態には、それぞれ以下に示す属性があります：
 - 「折りたたみ状態」
 - 開く
 - 閉じる
 - 「メールの受信状態」
 - メールなし
 - 未読メールあり
 - センターに未読メールあり
 - 「メッセージの受信状態」
 - メッセージなし
 - 未読メッセージあり
 - センターに未読メッセージあり
- 上記の3つの状態をアプリケーションから取得するには、`PhoneSystem.getAttribute(int attr)`メソッドを使用します（詳細は、API ドキュメント参照）。なお、「メールの受信状態」および「メッセージの受信状態」をこのメソッドで取得する際、アプリケーションディスクリプタファイル（ADF）の設定項目「GetSysInfo」を設定しておく必要があります。

上記以外に、「ソフトキー1（左側）」「ソフトキー2（右側）」は、キーボードの「A」「S」キーにそれぞれ対応しています（ただし、テキスト編集中は無効）。

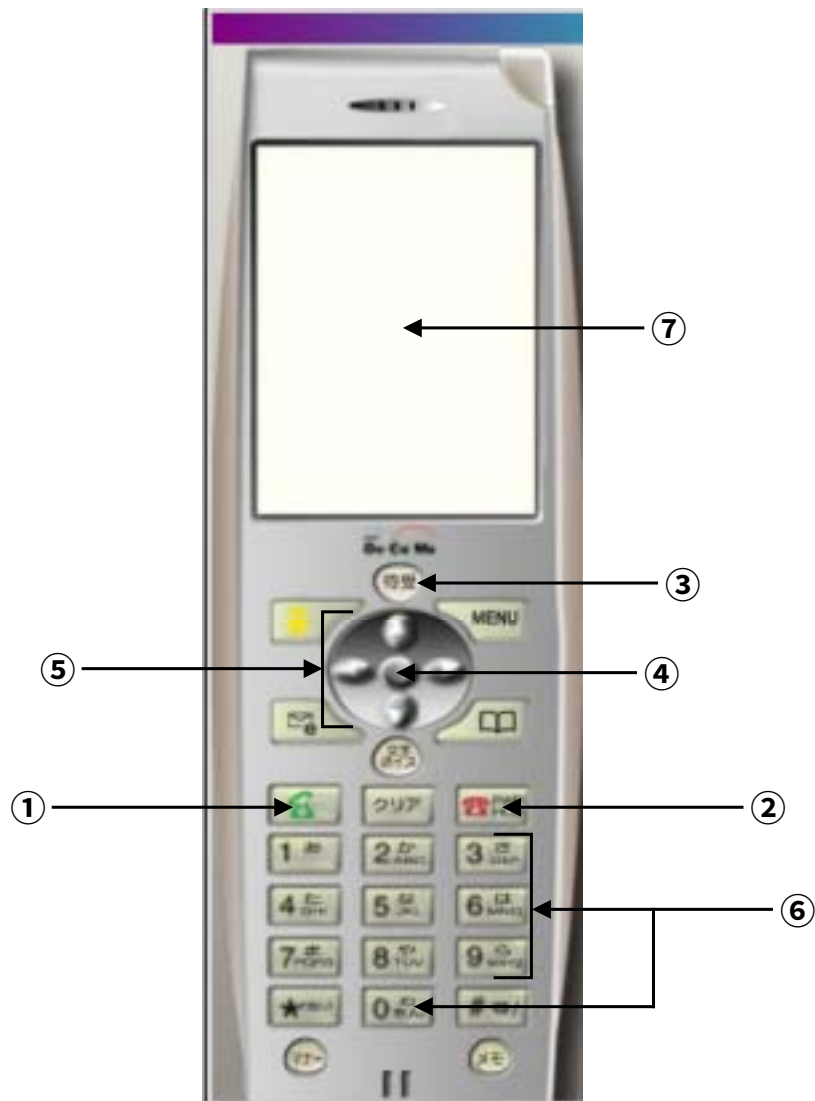


図 4-5 エミュレーション端末

4.4. コンパイラ (javac) の選択

メニューから「その他」->「sun.tools.javac.Mainを使用する」を選択することにより、コンパイラ (javac) を選択することができます（図 4-6）。ここで選択されたコンパイラ (javac) は、プロジェクトのビルド時（「5.5. プロジェクトのビルドと通常起動」を参照）に使用されるコンパイラ (javac) です。

デフォルトでは「com.sun.tools.javac.Main」を使用しますが、「sun.tools.javac.Mainを使用する」チェックボックスをチェックすることにより「sun.tools.javac.Main」を使用します。

また、選択されたコンパイラ（javac）の情報は保存されるので、次回「iα ppliTool」起動時に自動的に設定されます。

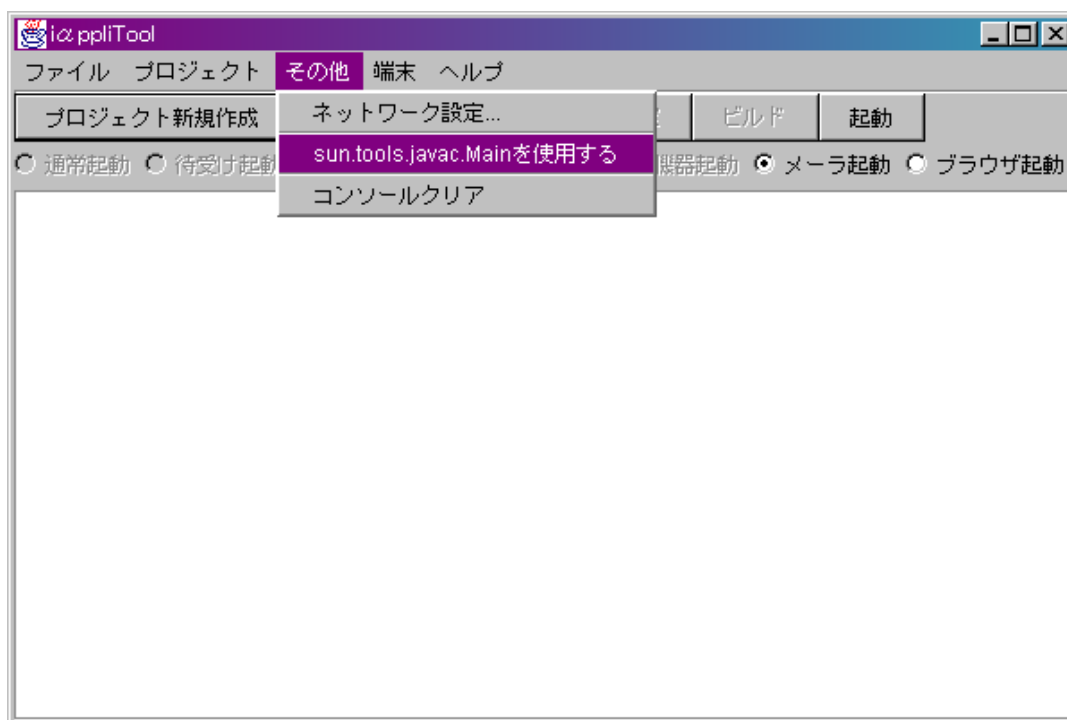


図 4-6 コンパイラ（javac）の選択

5. プロジェクト

「プロジェクト」とは、1つのアプリケーションに対するファイルの集合です。これには、ソースコードから最終的な 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（ADF）に至る全てのファイルが含まれます。プロジェクトの目的は、開発の様々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。

プロジェクトにおける開発作業項目には、以下のものがあります：

- 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を行います。

- プロジェクトの起動

選択するエミュレーション端末上でプロジェクトを起動し、アプリケーションの動作をエミュレートします。

5.1. プロジェクトの作成～起動

「iα ppliTool」では、プロジェクトの作成から起動までに以下に示すことができます：

- 既存プロジェクトの読み込み（「5.2. 既存プロジェクトの読み込み」を参照）
- アプリケーションディスクリプタファイル（ADF）の変更（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」を参照）
- 新規プロジェクトの作成（「5.4. プロジェクトの新規作成」を参照）
- プロジェクトのビルドと通常起動（「5.5. プロジェクトのビルドと通常起動」を参照）

5.2. 既存プロジェクトの読み込み

既存プロジェクトを読み込むには

1. メニューから「プロジェクト」→「読み込み…」を選択するか、「プロジェクト読み込み」ボタンを押します。

「プロジェクト読み込み」ウィンドウが表示されます（図 5-1）。

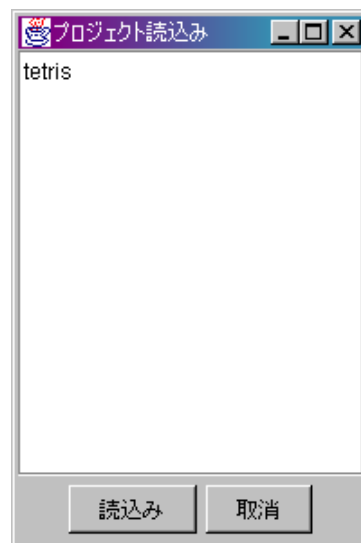


図 5-1 プロジェクト読み込みウィンドウ

2. プロジェクト名を選択し「読み込み」ボタンを押します。

メインウィンドウのタイトルバーにプロジェクト名が表示されます（図 5-2）。

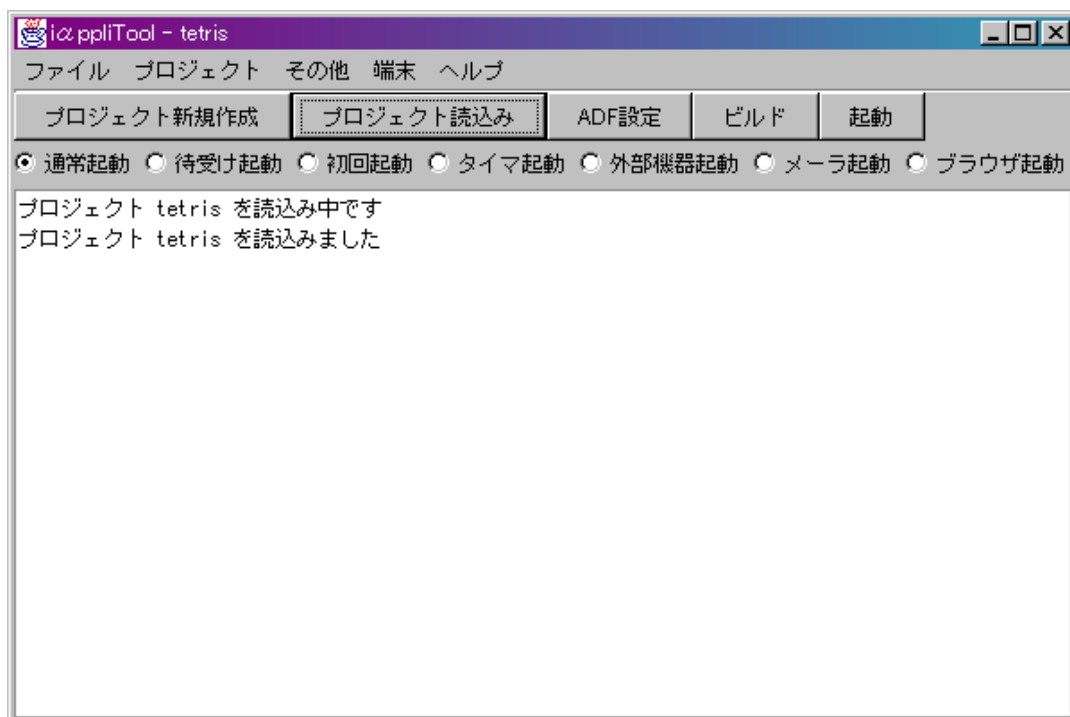


図 5-2 既存プロジェクト読み込み後のメインウィンドウ

5.3. アプリケーションディスクリプタファイル (ADF) の編集

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

1. メニューから「プロジェクト」→「ADF 設定...」を選択するか、「ADF 設定」ボタンを押します。

「ADF 設定」ウィンドウが表示されます（図 5-3）。

図 5-3 ADF 設定ウィンドウ

2. 必要な設定項目を入力します。

「設定」ボタンを押すことにより変更が有効となります。各設定項目は、表 5-1 に示す通りです。

表 5-1 アプリケーションディスクリプタファイル(ADF)の設定項目

設定項目	説明	種別
AppName	アプリケーション名（最大 16 バイト）。設定する文字列は、SJIS を使用する。	必須
AppVer	アプリケーションのバージョン（最大 10 バイト）。この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
PackageURL	アプリケーションの場所を表す URL（最大 255 バイ	必須

	ト)。設定する文字列は、ASCII 形式とする。ただし、サーバのアドレスを、IP アドレスを直接指定する形式で記述することはできません。 新規プロジェクトを作成したときのデフォルトの値は、アプリケーションの JAR ファイルの名前になっているので、適切な URL に変更してください。 また、この項目に相対 URL を設定した場合、「4.1. ネットワークの設定」で設定した「ADF の URL」フィールドがベースの URL となります。 アプリケーションはこの PackageURL のホストとポートのみ通信可能です。	
AppSize	アプリケーションサイズ（バイト単位）。0 または負数が指定された場合は、アプリケーションの起動を禁止します。この項目は、ビルドの JAR ファイル生成時に自動的に設定されます。	必須
ConfigurationVer (KvmVer)	コンフィグレーション(KVM)のバージョン。KvmVer と ConfigurationVer はどちらも同じ意味とします。ただし、両方指定された場合は、アプリケーションの起動を禁止します。設定する文字列は、CLDC-1.0 とし、それ以外を設定した場合は、アプリケーションの起動を禁止します。但し、大文字・小文字の区別はしません。この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
ProfileVer	プロファイルのバージョン。設定する文字列は、DoJa-2.0 とし、それ以外を設定した場合は、アプリケーションの起動を禁止します。但し、大文字・小文字の区別はしません。この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
SPSize	スクラッチパッドのサイズ（バイト単位）。エミュレータ上で指定できる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。この項目を設定しない場合のデフォルト値は 0 バイトとなります。また、負数、整数以外が設定された場合は、アプリケーションの起動を禁止します。	オプション
AppClass	アプリケーションのメインクラス名（最大 255 バイト）。このクラス名は、com.nttdocomo.ui.IApplication のサブクラスの名前でなければなりません。設定する文字列は、ASCII 形式とし、255 バイトを超えて設定した場合は、アプリケーションの起動を禁止します。	必須
AppParam	メインクラスのパラメータ（最大 255 バイト）。スペースで区切って設定します。アプリケーションは、com.nttdocomo.ui.IApplication.getArgs() メソッドを使用してこのパラメータを取得します。このメソッドは、この項目が設定されていないと null を返します。設定できる文字列は ASCII 形式とし、255 バイト以内ならパラメータの個数に制限はありません。しかし、255 バイトを超えて設定した場合は、アプリケーションの起動を禁止します。	オプション
LastModified	アプリケーションの最終更新時刻。設定する文字列の形式については「5.3.1. LastModified 項目の表記方法」を参照。この項目は、ビルドの JAR ファイル生成時に自動的に	必須

	に設定されます。	
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。有効な値は”http”（ASCII形式とし、大文字・小文字の区別はしません）のみとし、”http”以外の文字を設定した場合は、アプリケーションの起動を禁止します。また、この項目に何も設定しなかった場合は、ネットワーク接続不可なアプリケーションとみなされますので、ネットワーク接続しようとする例外が発生します。	オプション
TargetDevice	アプリケーションが動作する機種名（最大 128 バイト）。この項目は、特定の機種に対してのみアプリケーションのダウンロードを許可したいときに設定します。設定する文字列は、ASCII形式とし、大文字・小文字の区別はしません。コンマで区切られた文字列を 1 機種とします。複数の機種をターゲットとしている場合は、”X504i,Y504i,Z504i”のように設定します。また、この項目に何も設定しなかった場合は、全ての機種をターゲットとしているとみなされます。また、同一シリーズ全機種をターゲットとする場合は、”all504”のようにシリーズ番号の前に”all”を付加します。この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
LaunchAt	起動時間間隔指定。アプリケーションを一定時間ごとに自動起動させるときに設定します。 形式: I h (“h”は起動間隔（時間単位）) “h”に 0 が設定された場合は、無効なものとみなします。有効桁数は 2 桁とします。また、設定する文字列は、ASCII形式とし、それ以外が設定された場合は、アプリケーションの起動を禁止します。但し、大文字・小文字の区別はしません。	オプション
MyConcierge	待ち受けアプリケーションかどうかを表すフラグ。この項目は、エミュレータ上で待受け起動可能であるかどうかの判定に使用します（「7.1. 待受け起動」参照）。有効な値は”Yes”（ASCII形式とし、大文字・小文字の区別はしません）のみとし、”Yes”以外の文字を設定した場合は、アプリケーションの起動を禁止します。また、この項目に何も設定しない場合は、待受け起動しないアプリケーションであるとみなし、エミュレータ上で待受け起動を禁止します。	オプション
UseTelephone	電話の音声通話機能を利用するかどうかを表すフラグ。有効な値は”call”（ASCII形式とし、大文字・小文字の区別はしません）のみとし、”call”以外の文字を設定した場合は、アプリケーションの起動を禁止します。また、この項目に何も設定しない場合は、電話の音声通話機能を利用しないアプリケーションであるとみなします。	オプション
UseBrowser	アプリケーションからブラウザ機能を利用するかどうかを表すフラグ。有効な値は”launch”（ASCII形式とし、大文字・小文字の区別はしません）のみとし、”launch”以外の文字を設定した場合は、アプリケーションの起動を禁止します。また、この項目に何も設定しない場合は、ブ	オプション

	ラウザ機能を利用しないとみなしますので、アプリケーションからブラウザ機能を使用すると例外が発生します。	
LaunchByMail	アプリケーションの起動を許可するメールアドレス（最大 50 バイト）。この項目に”any”を設定した場合は、任意のメールアドレスからのアプリケーションの起動を許可します。”any”以外のメールアドレスを設定した場合は、アプリケーションを起動しようとしているメールアドレスの From: アドレスと、この項目で設定した値が完全に後方一致するときのみ起動を許可します。それ以外はアプリケーションの起動を禁止します。また、この項目に複数のメールアドレスを設定した場合は、先頭の 1 つのメールアドレスのみ有効とします。設定する文字列は、ASCII 形式とします。	オプション
LaunchByBrowser	アプリケーションの起動を許可する URL（最大 255 バイト）。この項目に”any”を設定した場合は、任意の URL からのアプリケーションの起動を許可します。”any”以外の URL を設定した場合は、アプリケーションを起動しようとしている URL と、この項目で設定した値が完全に前方一致するときのみ起動を許可します。また、この項目に複数の URL を設定した場合は、先頭の 1 つの URL のみ有効とします。なお、先頭から空白、タブまでを 1 つの URL とみなします。設定する文字列は、ASCII 形式とします。	オプション
AllowPushBy	アプリケーションのプッシュ起動を許可する外部インタフェース（最大 255 バイト）。この項目に「”Irda”:<起動コマンド>」を設定した場合は、Irda からアプリケーションの起動を許可します（<起動コマンド>はアプリケーション起動チェックの際に使用します。なお、<起動コマンド>は 1 文字以上の文字列とします）。”Irda”以外の文字を指定した場合は、アプリケーションの起動を禁止します。また、複数の外部インタフェースを設定する場合は、空白、タブで機器を区切ります。この項目に複数の外部インタフェースを設定した場合は、先頭の 1 つの外部インタフェースのみ有効とします。	オプション
AppTrace	アプリケーションのトレース機能を利用するかどうかを表すフラグ。有効な値は”on”（ASCII 形式とし、大文字・小文字の区別はしません）のみとし、”on”以外の文字を設定した場合は、アプリケーションの起動を禁止します。また、この項目に何も設定しなかった場合は、アプリケーションのトレース機能を使用しないものとみなします。この項目は、エミュレータでは使用しないので設定しなくても問題ありません。	オプション
DrawArea	アプリケーションが使用する描画領域。この項目に以下の形式に従った値を設定した場合: 形式: (横のドット数) x (縦のドット数) 96 <= (横のドット数) <= 端末の最大横ドット数 72 <= (縦のドット数) <= 端末の最大縦ドット数 アプリケーション起動時の画面を、端末の描画領域の中央に設定します。その際、描画領域周囲の表示色は実装	オプション

	依存とし、アプリケーションから描画を行うことはできません。上記の形式（設定可能なドット数は正の整数のみ）以外の値を設定した場合は、アプリケーションの起動を禁止します。また、この項目に何も設定しなかった場合は、通常通りの描画とします。	
GetSysInfo	メール、メッセージピクト情報取得可否を表すフラグ。有効な値は"yes"（ASCII形式とし、大文字・小文字の区別はしません）のみとし、"yes"以外の文字を設定した場合は、アプリケーションの起動を禁止します。また、この項目に何も設定しない場合は、メール、メッセージピクト情報を取得しないものとみなします。	オプション

5.3.1. LastModified 項目の表記方法

ADF の LastModified 項目の表記方法を下記の通り規定します。

- 表記法は、RFC2822 の「3.3. Date and Time Specification」に準拠します。ただし、「4.3. Obsolete Date and Time」に記述されている内容には対応しておりません。以下に規定した項目については、これに従うものとします。
- 日付の表記は、ゼロサプレスの有無の関わらず可能とします。
例:
Sun, 1 Apr 2001 19:05:50 +0900 /* OK (ゼロサプレス無) */
Sun, 01 Apr 2001 19:05:50 +0900 /* OK (ゼロサプレス有) */
- 時、分、秒の表記は、ゼロサプレスの有無の関わらず可能とします。
例:
Sun, 01 Apr 2001 19: 5:50 +0900 /* OK (ゼロサプレス無) */
Sun, 01 Apr 2001 19:05:50 +0900 /* OK (ゼロサプレス有) */
- 曜日名の表記は、"Mon"、"Tue"、"Wed"、"Thu"、"Fri"、"Sat"、"Sun"とします。大文字・小文字の区別はしません。これに合致しない場合は、不正とみなします。
例: MON, mon, MoN, mOn /* OK */
- 月名の表記は、"Jan"、"Feb"、"Mar"、"Apr"、"May"、"Jun"、"Jul"、"Aug"、"Sep"、"Oct"、"Nov"、"Dec"とします。大文字・小文字の区別はしません。これに合致しない場合は、不正とみなします。
例: JAN, jan, JaN, jAn /* OK */
- 日付と時刻の有効範囲を閏年まで含めてチェックします。但し、日付と曜日の整合性はチェックしません。
例:
Sun, 01 Apr 2001 19:05:50 +0900 /* OK */
Tue, 01 Apr 2001 19:05:50 +0900 /* OK (曜日の整合性チェックなし) */
Sun, 01 Apr 2001 25:05:50 +0900 /* NG (時刻が不正) */
Sun, 31 Apr 2001 19:05:50 +0900 /* NG (4/31 は存在しない) */
Sun, 29 Feb 2001 19:05:50 +0900 /* NG (2001/2/29 は存在しない) */
Sun, 29 Feb 2004 19:05:50 +0900 /* OK (2004/2/29 は存在する) */
- タイムゾーンの表記は省略可能とし、省略された場合は+0900 と解釈します。また、タイムゾーンのアルファベット表記 ("GMT"、"JST") の可能とします。

5.4. プロジェクトの新規作成

新しいプロジェクトを作成するには

1. メニューから「プロジェクト」→「新規作成...」を選択するか、「プロジェクト新規作成」ボタンを押します。

「新規作成」ウインドウが表示されます（図 5-4）。

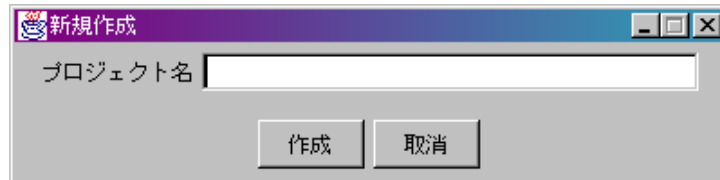


図 5-4 プロジェクト新規作成ウインドウ

2. 新しいプロジェクト名を入力し、「作成」ボタンを押します^[3]。

メインウインドウのタイトルバーにプロジェクト名が表示されます（図 5-5）。

プロジェクトのソースコード（.java ファイル）とリソースファイル（アイコンなどのイメージファイルなど）をどこに置くべきか、また、スクラッチパッドの格納場所を示すメッセージが、コンソールに以下の形式で表示されます：

「iα ppli Development Kit」を C:\iDK にインストールした場合

ソースコードの場所: “C:\iDK\apps\<プロジェクト名>\src”

リソースファイルの場所: “C:\iDK\apps\<プロジェクト名>\res”

スクラッチパッドの場所: “C:\iDK\apps\<プロジェクト名>\sp”

^[3] 新規プロジェクトを作成した際に、エミュレータは以下のディレクトリとファイルを作成します：

「iα ppli Development Kit」を C:\iDK にインストールした場合

ソースコードの格納場所: C:\iDK\apps\<プロジェクト名>\src

リソースファイルの格納場所: C:\iDK\apps\<プロジェクト名>\res

スクラッチパッドの格納場所: C:\iDK\apps\<プロジェクト名>\sp

JAR、JAM ファイルの格納場所: C:\iDK\apps\<プロジェクト名>\bin

JAM ファイル: C:\iDK\apps\<プロジェクト名>\bin\<プロジェクト名>.jam

この時点での JAM ファイルの内容は「PackageURL =<プロジェクト名>.jar」のみです。また、ビルドした際に設定された ADF の項目や「5.3. アプリケーションディスクリプタファイル（ADF）の編集」で編集された ADF はこの JAM ファイルに保存されます。

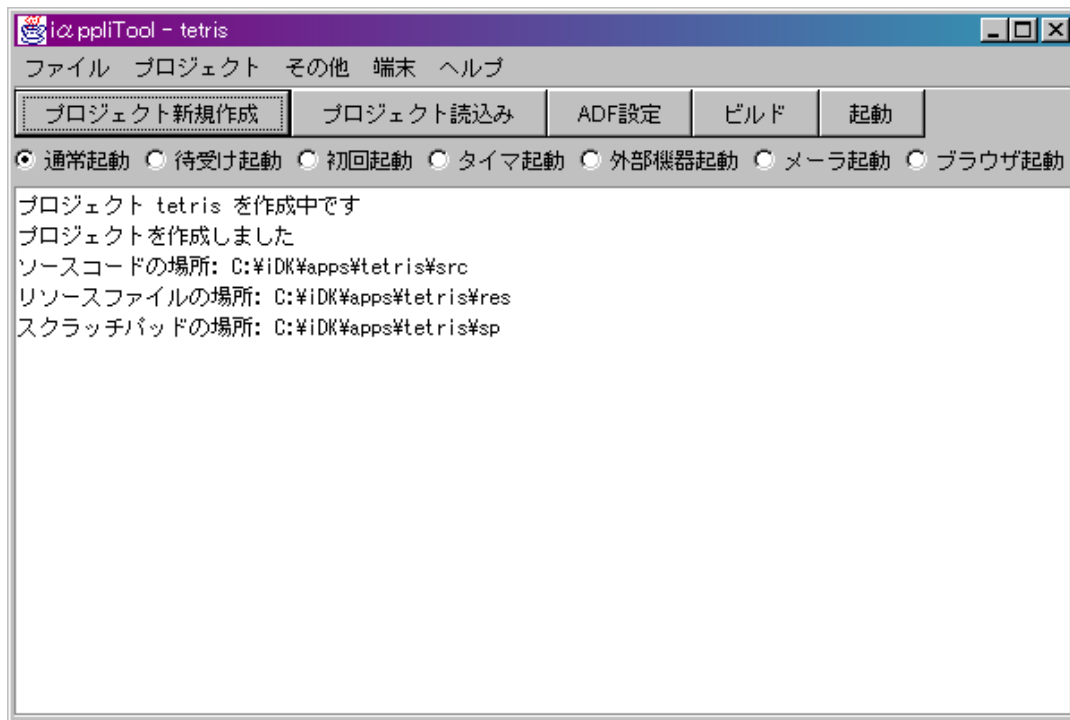


図 5-5 新規プロジェクト作成後のメインウィンドウ

3. ソースコードとリソースファイルをそれぞれのディレクトリに置きます。

5.5. プロジェクトのビルドと通常起動^[4]

プロジェクトのビルドと通常起動を行うには

1. メニューから「プロジェクト」→「ビルド」を選択するか、「ビルド」ボタンを押します（図 5-6）。

ソースコードのコンパイルと事前検証を行い、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^[5]。

^[4] 通常起動とは、com.nttdocomo.ui.IApplication クラスを継承しているアプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する（実機において、通常のメニューからの起動（起動タイプが Iapplication.LAUNCHED_FROM_MENU）に相当）ことです。また、「iα ppliTool」の「起動」ボタンから直接起動する起動方法として「初回起動」（「5.6. 初回起動」参照）、「タイマ起動」（「5.7. タイマ起動」参照）および「外部機器起動」（「5.8. 外部機器起動」参照）もあります。

^[5] 正確には、既に存在する<プロジェクト名>.jam という ADF の内容を更新します（「AppSize」と「LastModified」属性が設定されます）。また、「ビルド」時に「環境変数 JDK_HOME に JDK をインストールしたディレクトリを設定してください」というメッセージが出力された場合、環境変数 JDK_HOME に JDK をインストールしたディレクトリを設定した後、改めて「iα ppliTool」を起動してください。

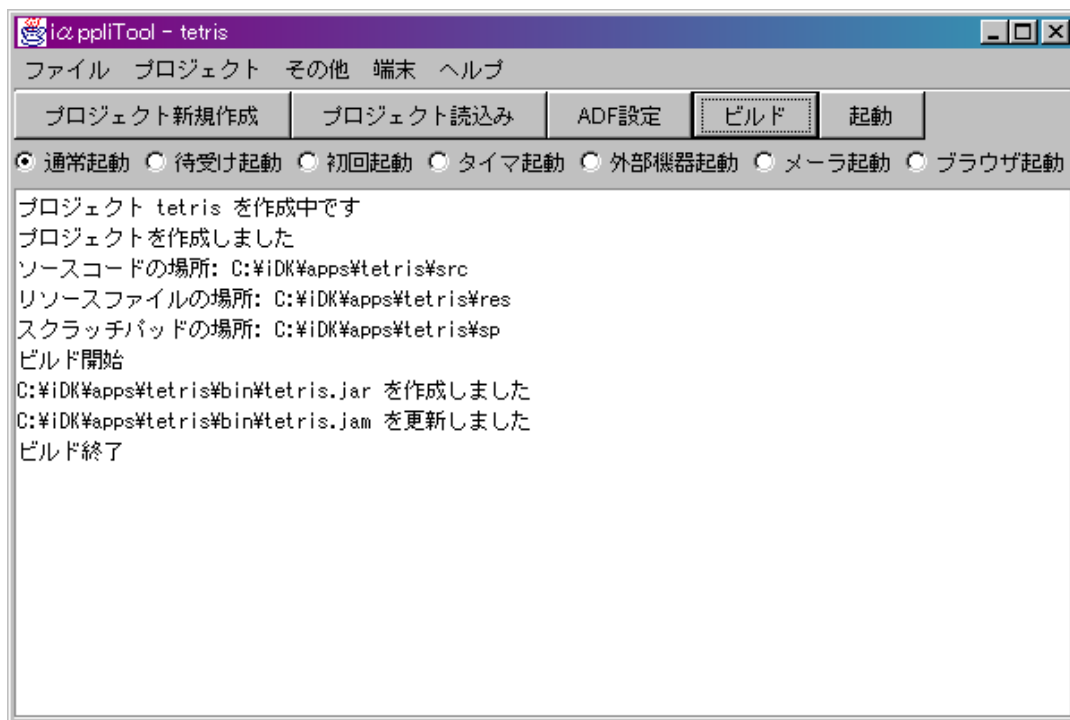


図 5-6 プロジェクトのビルド

2. メニューから「プロジェクト」→「ADF 設定...」を選択するか、「ADF 設定」ボタンを押します（図 5-7）。

アプリケーションを起動するために必要な ADF 項目を入力し、「設定」ボタンを押します（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」参照）。

AppName	
AppVer	
PackageURL	tetris.jar
AppSize	8110
ConfigurationVer	
ProfileVer	
SPsize	
AppClass	
AppParam	
LastModified	Mon, 03 Dec 2001 16:28:12
UseNetwork	
TargetDevice	
LaunchAt	
MyConcierge	
UseTelephone	
UseBrowser	
LaunchByMail	
LaunchByBrowser	
AllowPushBy	
AppTrace	
DrawArea	
GetSysInfo	

図 5-7 ADF の設定

3. 「起動」ボタンを押します。

ビルドしたアプリケーションを選択した端末で起動します（図 5-8）。このとき、エミュレータが以下の状態の時にはアプリケーションを起動できません：

- メーラ（「7.2. メーラ起動」を参照）が起動している
- ブラウザ（「7.3. ブラウザ起動」を参照）が起動している
- 待ち受け起動以外でアプリケーションが起動している
- 待ち受け起動しているアプリケーションが活性化状態あるいはサスペンド状態になっている（「7.1. 待ち受け起動」を参照）

デフォルトの起動方法は「通常起動」ですが、その他に「待ち受け起動」「メーラ起動」「ブラウザ起動」の3種類の起動方法があります。これらの起動方法に関しては「7. 他のアプリケーション起動方法」で説明します。



図 5-8 アプリケーション起動

5.6. 初回起動

初回起動とは、com.nttdocomo.ui.IApplication クラスを継承しているアプリケーションを「iαppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

「通常起動」は起動タイプが「LAUNCHED_FROM_MENU」ですが、「初回起動」は起動タイプが「LAUNCHED_AFTER_DOWNLOAD」となります。

実機において、「通常起動」とは通常のメニューからの起動であり、「初回起動」とはダウンロード直後（通常起動の1回目）の起動に相当するものです。

初回起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 5-9 に示すように、メインウィンドウの5つのラジオボタンから「初回起動」をチェックし、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「5.5. プロジェクトのビルドと通常起動」参照）。

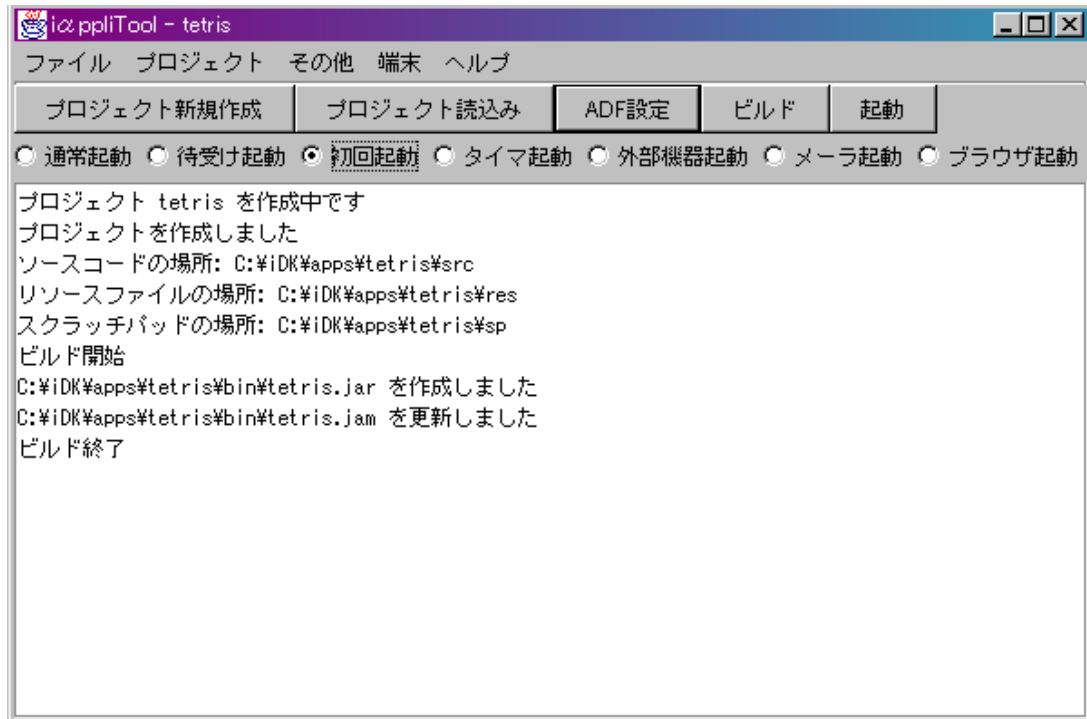


図 5-9 初回起動準備

5.7. タイマ起動

タイマ起動とは、com.nttdocomo.ui.IApplication クラスを継承しているアプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「通常起動」は起動タイプが「LAUNCHED_FROM_MENU」ですが、「タイマ起動」は起動タイプが「LAUNCHED_FROM_TIMER」となります
- アプリケーションディスクリプタファイル（ADF）の設定項目「LaunchAt」に適切な値を設定した場合のみ起動可能となります（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」参照）

実機において、「通常起動」とは通常のメニューからの起動であり、「タイマ起動」とはタイマエージェントにより指定された時刻の起動に相当するものです。

タイマ起動するには、プロジェクトを読み込み、ビルド、ADFの設定を行った後、図 5-10 に示すように、メインウィンドウの7つのラジオボタンから「タイマ起動」をチェックし、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「5.5. プロジェクトのビルドと通常起動」参照）。

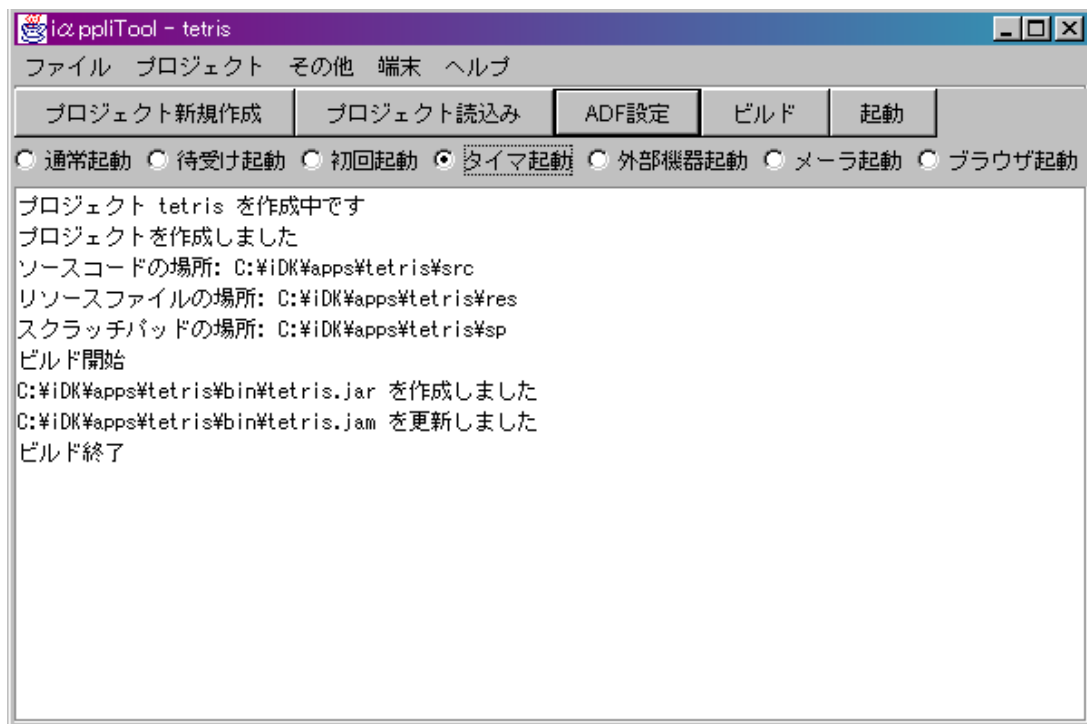


図 5-10 タイマ起動準備

5.8. 外部機器起動

外部機器起動とは、com.nttdocomo.ui.IApplication クラスを継承しているアプリケーションを「iα ppliTool」の「起動」ボタンから直接起動する起動方法の1つですが、「通常起動」とは以下の点のみ異なります：

- 「通常起動」は起動タイプが「LAUNCHED_FROM_MENU」ですが、「外部機器起動」は起動タイプが「LAUNCHED_FROM_EXT」となります
- アプリケーションディスクリプタファイル（ADF）の設定項目「AllowPushBy」に適切な値を設定した場合のみ起動可能となります（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」参照）

実機において、「通常起動」とは通常のメニューからの起動であり、「外部機器起動」とは外部インタフェースのトリガによる起動に相当するものです。

外部機器起動するには、プロジェクトを読み込み、ビルド、ADF の設定を行った後、図 5-11 に示すように、メインウィンドウの7つのラジオボタンから「外部機器起動」をチェックし、「起動」ボタンを押します。

また、起動の際の制約などに関しては「通常起動」の場合と同様です（「5.5. プロジェクトのビルドと通常起動」参照）。

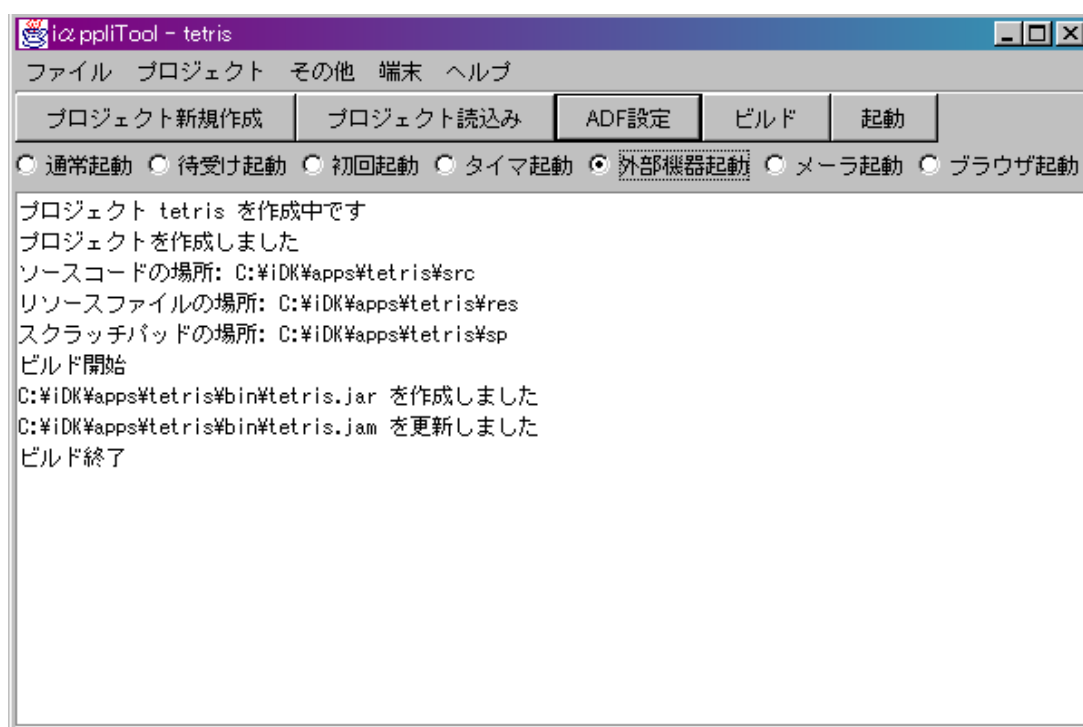


図 5-11 外部機器起動準

6. コンソールと「iαppliTool」の終了

- コンソール

起動中のアプリケーションからの標準出力・エラー出力は、コンソールに表示されます^[6]。メニューから「その他」→「コンソールのクリア」を選択すると、コンソールに表示されているメッセージが消去されます。

- 「iαppliTool」の終了

「iαppliTool」を終了するには、メニューから「ファイル」→「終了」を選択します。

^[6] ただし、絵文字は「・」、JIS X 0208 で定義されている漢字以外の漢字は「?」で表示されます。

7. 他のアプリケーション起動方法

アプリケーションの起動方法としては、通常起動（「通常起動」、「初回起動」、「タイマ起動」、「外部機器起動」）の他に「待受け起動」「メーラ起動」「ブラウザ起動」の3種類の起動方法があります。

7.1. 待受け起動

「待受け起動」とは、アプリケーションを「待ち受けアプリケーション」として起動することです。

7.1.1. 待ち受けアプリケーション

「待ち受けアプリケーション」とは、待受け画面になることができるアプリケーションのことで、com.nttdocomo.ui.MApplication クラスを継承する必要があります。表 7-1 に「待受け画面」と「待ち受けアプリケーション」との比較を示します。

表 7-1 「待受け画面」と「待ち受けアプリケーション」の比較

待受け画面設定	表示	動作
待受け画像 (GIF・JPEG)	静止画またはアニメ画像	キー押下、開閉に合わせてアニメ表示
待ち受けアプリケーション	待ち受けアプリケーションによる描画	特定キー押下でキーイベントを有効化

待ち受けアプリケーションは、通常のアプリケーションと同様にキー入力を受取る状態（活性化状態）と、待受け画面と同様に画面の表示は行わぬがキー入力は受取らない状態（非活性化状態）、及び一時停止した状態（休眠状態あるいはサスペンド状態）の3つの状態を持っています。待ち受けアプリケーションが実行中であっても、非活性化状態及び休眠状態では端末のネイティブ機能を優先します。

待ち受けアプリケーションが非活性化状態及び休眠状態では、キーイベントの受信とソフトキーの表示は行えず、全てネイティブアプリケーション側で処理します。一方、活性化状態ではキーイベントの受信とソフトキーの表示は、待ち受けアプリケーションで行います。

図 7-1 に待ち受けアプリケーションの状態遷移を示します。

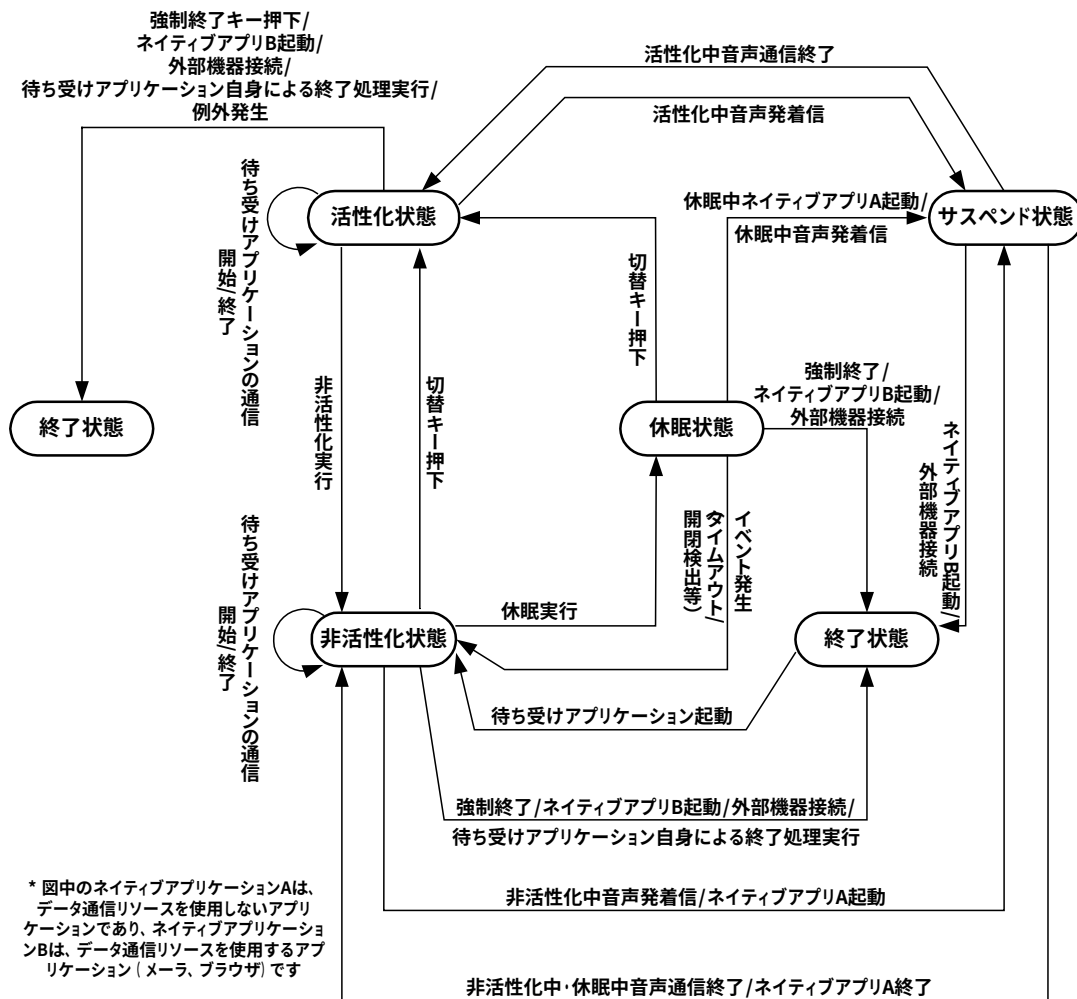


図 7-1 待ち受けアプリケーションの状態遷移

7.1.2. 待受け起動手順

1. メインウィンドウの7つのラジオボタンから「待受け起動」をチェックします (図 7-2)。

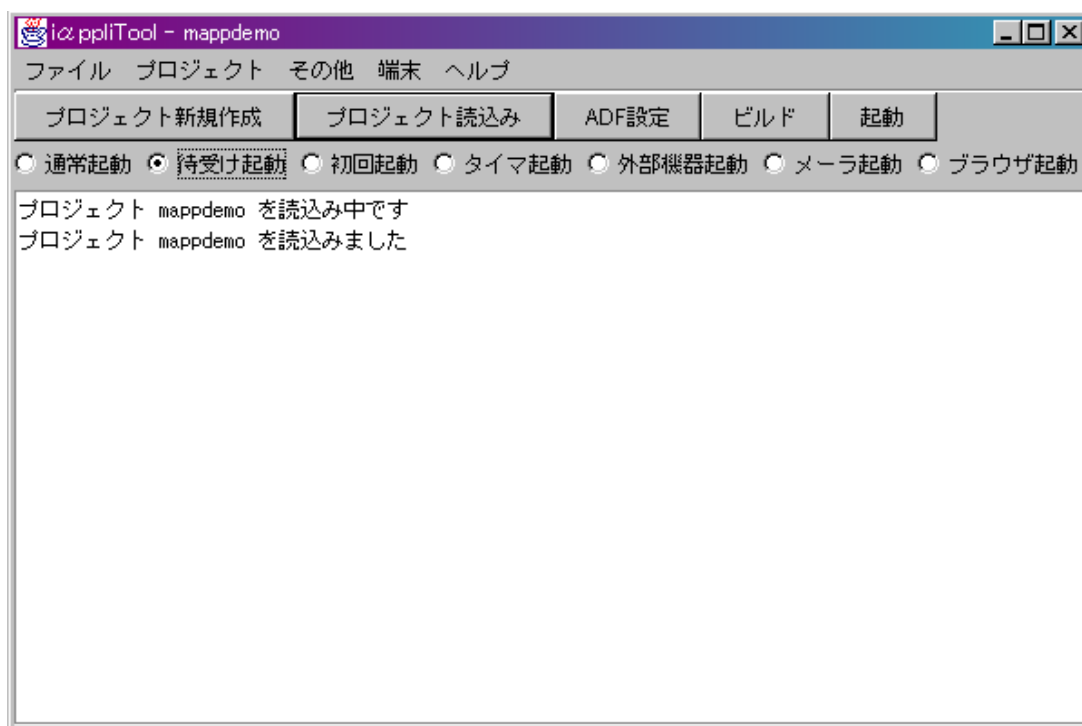


図 7-2 待受け起動準備

2. 「起動」ボタンを押します。

アプリケーションを待ち受けアプリケーションとして起動するためには、以下の条件を満たしている必要があります：

- アプリケーションを待ち受けアプリケーションとして作成していること（com.nttdocomo.ui.MApplication クラスを継承していること）。
- ADF の項目で「MyConcierge」に「Yes」が設定されていること（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」を参照）。
- アプリケーションやメーラ（「7.2. メーラ起動」を参照）、ブラウザ（「7.3. ブラウザ起動」を参照）が起動していないこと。

com.nttdocomo.ui.MApplication クラスの継承の有無と MyConcierge の設定の有無との関係を表 7-2 に示します。

表 7-2 Mapplication クラス継承の有無と MyConcierge 設定の有無の関係

	MyConcierge 設定有	MyConcierge 設定無
Mapplicatoin 継承有	起動	エラー
Mapplication 継承無	エラー	エラー

上記の条件を満たしていない場合には、アプリケーションの起動を禁止するダイアログあるいはコンソールにエラーメッセージが表示されます。

上記の条件を満たしている場合には、アプリケーションを待ち受けアプリケーションとして起動します（図 7-3）。



図 7-3 待ち受けアプリケーション起動

3. 待ち受けアプリケーションが「非活性化状態」あるいは「休眠状態」にあるときには、以下のアプリケーションが起動可能です：
 - 待ち受け起動以外のアプリケーション（「5.5. プロジェクトのビルドと通常起動」、「5.6. 初回起動」、「5.7. タイマ起動」および「5.8. 外部機器起動」を参照）
 - メーラ（「7.2. メーラ起動」を参照）
 - ブラウザ（「7.3. ブラウザ起動」を参照）

また、上記のアプリケーションが終了すると自動的に以前起動していた待ち受けアプリケーションが「非活性化状態」で再起動されます。

7.2. メーラ起動

「メーラ起動」とは、アプリケーションをメールから起動することです。

7.2.1. メーラ起動手順

1. メインウィンドウの5つのラジオボタンから「メーラ起動」をチェックします（図 7-4）。

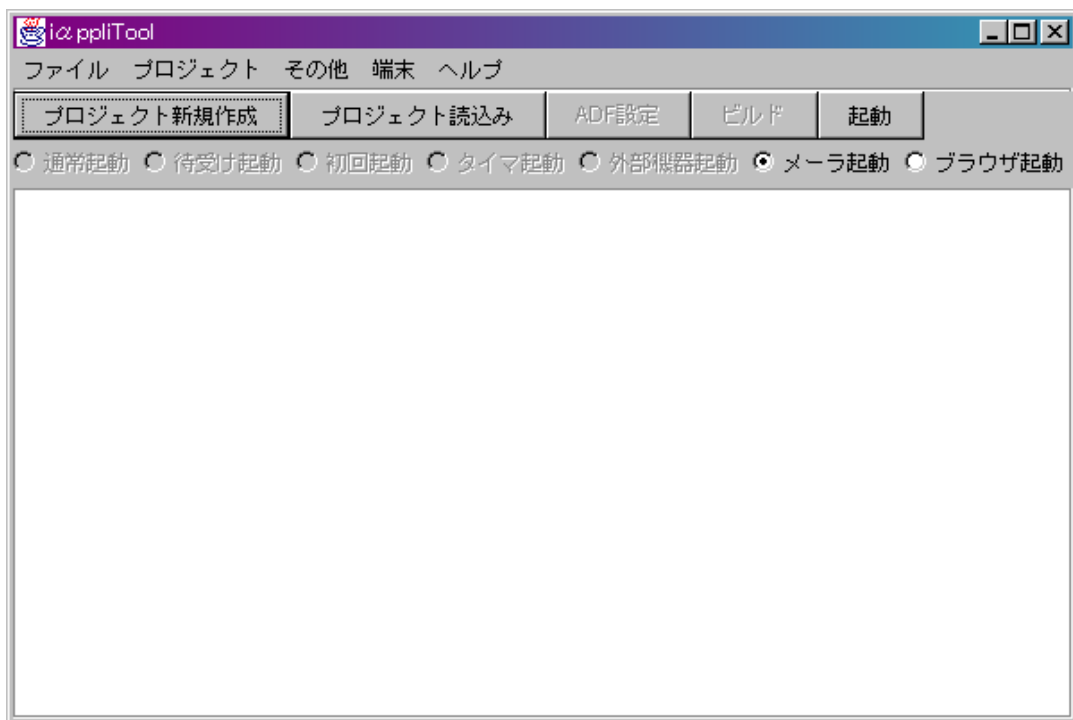


図 7-4 メーラ起動準備

1. 「起動」ボタンを押します。

メーラウィンドウが表示されます（図 7-5）。このとき、エミュレータが以下の状態の時にはメーラウィンドウは表示されません:

- 既にメーラが起動している
- ブラウザ（「7.3. ブラウザ起動」を参照）が起動している
- アプリケーションが待ち受け起動以外で起動している（「5.5. プロジェクトのビルドと通常起動」、「5.6. 初回起動」、「5.7. タイマ起動」および「5.8. 外部機器起動」を参照）
- アプリケーションが待受け起動している（活性化状態あるいはサスペンド状態のとき）（「7.1. 待受け起動」を参照）

図 7-5 の上方には、受信メールの一覧が表示されます（「iopl Development Kit」を「C:\iDK」にインストールした場合、「C:\iDK\lib\mail」に置かれている

「*.mail」ファイル（*は0、1、2、・・・）のヘッダ部分が表示されます（「*.mail」ファイルの記述方法は、「8.1. 「*.mail」ファイルの記述方法」を参照）。

受信メールの一覧からメールを選択すると、図 7-5 の下方に選択したメールの内容が表示されます。

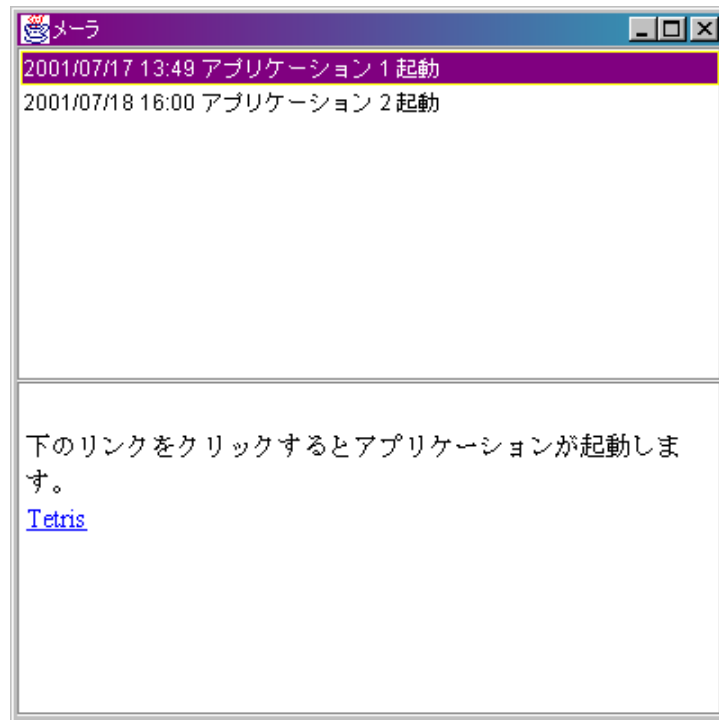


図 7-5 メールウインドウ

2. 図 7-5 の下方に表示されているリンク（青い文字）をクリックします。

アプリケーションをメールから起動するためには、以下の条件を満たしている必要があります：

- ADF の項目で「LaunchByMail」に「any」あるいは起動を許可するメールアドレスを設定し（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」を参照）、そのメールアドレスがメールの From と一致していること。
- 「*.mail」ファイルの ADF の項目に設定されている URL をキーとして「JAM.properties」ファイル（「ia ppli Development Kit」を「C:\iDK」にインストールした場合、「C:\iDK\lib」に置かれている）から得られるローカルの ADF の名前（*.jam）が指定のディレクトリ（「ia ppliTool」を「C:\iDK」にインストールした場合、「C:\iDK\apps\<プロジェクト名>/bin」）に置かれていること（「JAM.properties」ファイルの記述方法は、「8.3. 「JAM.properties」ファイルの記述方法」を参照）。

上記の条件を満たしていない場合には、アプリケーションの起動を禁止するダイアログあるいはコンソールにエラーメッセージが表示されます。

上記の条件を満たしている場合には、アプリケーションをメーラ起動するための確認ダイアログが表示され（図 7-6）、「はい」ボタンを押すとアプリケーションが起動します（図 7-7）。

メーラから待ち受けアプリケーションを起動した場合、待ち受けアプリケーションとしては動作せず通常のアプリケーションとして動作します。



図 7-6 起動確認ダイアログ



図 7-7 メーラ起動されたアプリケーション

3. メーラ起動されたアプリケーションの終了方法によりその後自動起動するアプリケーションが異なります:

- メーラ起動されたアプリケーションを正常終了（アプリケーションが `IApplication.terminate()` メソッドを呼び出して終了した場合）

再びメーラウィンドウが表示されます。メーラウィンドウを終了する（ウィンドウの右上の「×」を押す）と、もし、メーラを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待ち受け画面に戻ります。

- メーラ起動されたアプリケーションを強制終了（端末の「終話ボタン」を押した場合）

メーラウィンドウは表示されず、もし、メーラを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待ち受け画面に戻ります。

7.3. ブラウザ起動

「ブラウザ起動」とは、アプリケーションをブラウザから起動することです。

7.3.1. ブラウザ起動手順

1. メインウィンドウの5つのラジオボタンから「ブラウザ起動」をチェックします（図 7-8）。

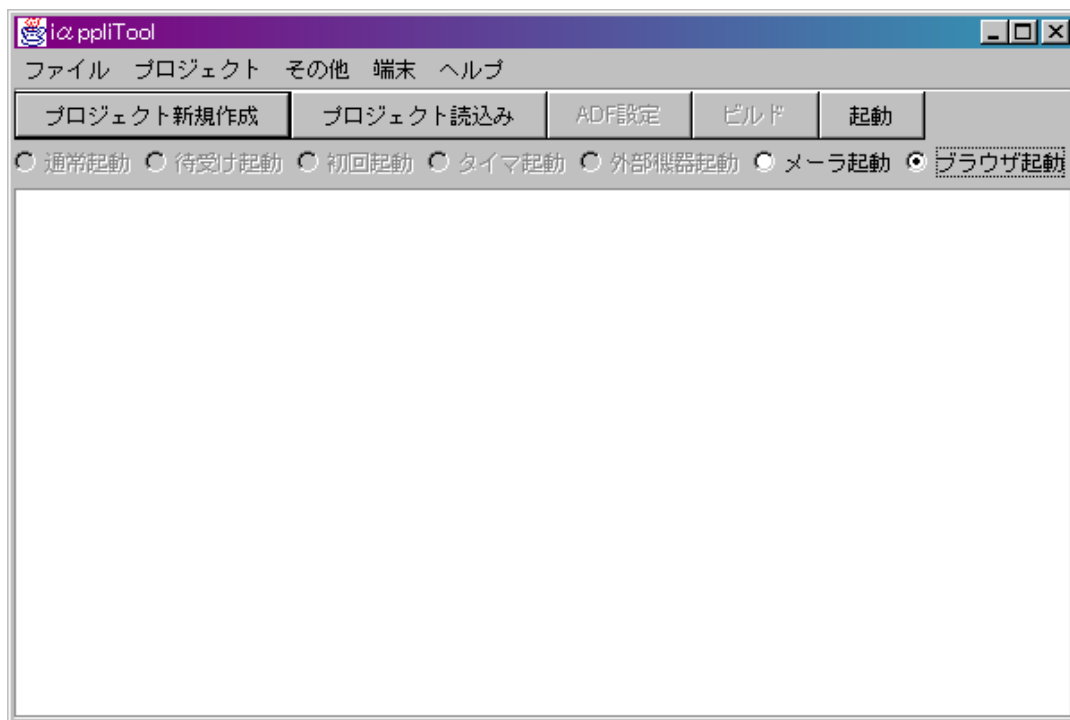


図 7-8 ブラウザ起動準備

2. 「起動」ボタンを押します。

ブラウザウインドウが表示されます（図 7-9）。このとき、エミュレータが以下の状態の時にはブラウザウインドウは表示されません:

- 既にブラウザが起動している
- メーラ（「7.2. メーラ起動」を参照）が起動している
- アプリケーションが待ち受け起動以外で起動している（「5.5. プロジェクトのビルドと通常起動」、「5.6. 初回起動」、「5.7. タイマ起動」および「5.8. 外部機器起動」を参照）
- アプリケーションが待受け起動している（活性化状態あるいはサスペンド状態のとき）（「7.1. 待受け起動」を参照）

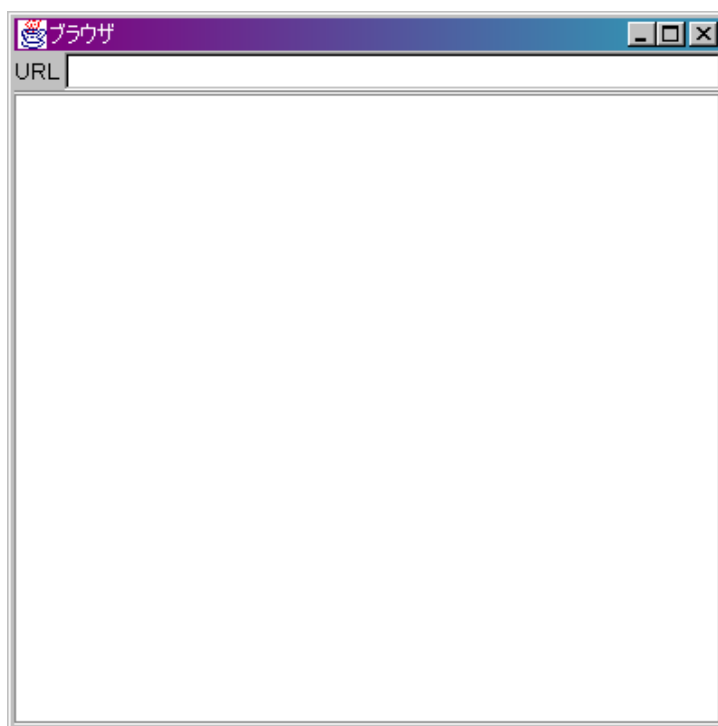


図 7-9 ブラウザウインドウ

3. 図 7-9 の上方の URL フィールドにブラウザ起動するための URL（この URL で指定した HTML ファイルの記述方法は「8.2. 「HTML」ファイルの記述方法」を参照）を入力し、リターンキーを押します（図 7-10）。

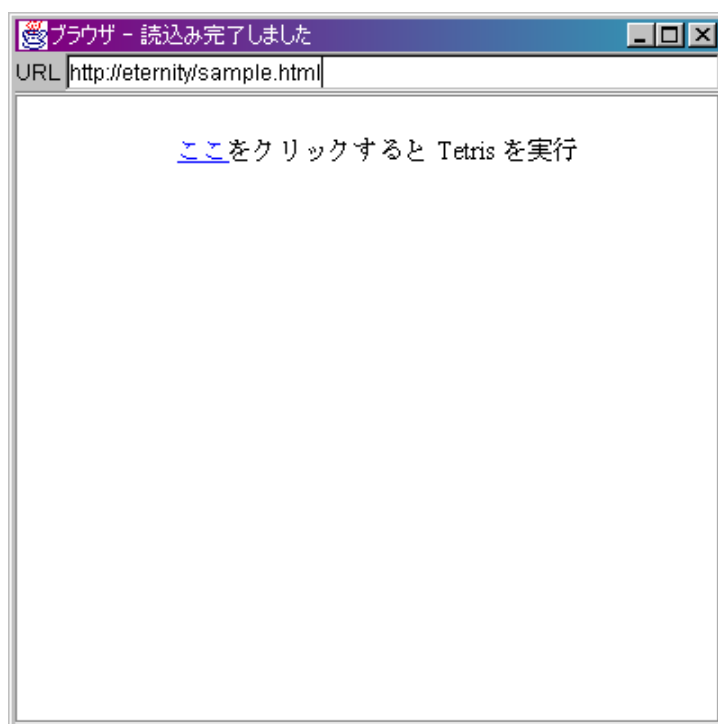


図 7-10 URL の表示

4. 図 7-10 のリンク（青い文字）をクリックします。

アプリケーションをブラウザから起動するためには、以下の条件を満たしている必要があります:

- ADF の項目で「LaunchByBrowser」に「any」あるいは起動を許可する URL が設定されていること（「5.3. アプリケーションディスクリプタファイル（ADF）の編集」を参照）。
- HTML ファイルの OBJECT タグの中に設定されている data の URL をキーとして「JAM.properties」ファイル（「8.3. 「JAM.properties」ファイルの記述方法」を参照）から得られるローカルの ADF の名前（*.jam）が指定のディレクトリ（「iα ppli Development Kit」を「C:\iDK」にインストールした場合、「C:\iDK\apps\<プロジェクト名>/bin」）に置かれていること。

上記の条件を満たしていない場合には、アプリケーションの起動を禁止するダイアログあるいはコンソールにエラーメッセージが表示されます。

上記の条件を満たしている場合には、アプリケーションをブラウザ起動するための確認ダイアログが表示され（図 7-11）、「はい」ボタンを押すとアプリケーションが起動します（図 7-12）。

ブラウザから待ち受けアプリケーションを起動した場合、待ち受けアプリケーションとしては動作せずに通常のアプリケーションとして動作します。



図 7-11 起動確認ダイアログ



図 7-12 ブラウザ起動されたアプリケーション

5. ブラウザ起動されたアプリケーションの終了方法によりその後自動起動するアプリケーションが異なります:
 - ブラウザ起動されたアプリケーションを正常終了（アプリケーションが `IApplication.terminate()` メソッドを呼び出して終了した場合）

再びブラウザウィンドウが表示されます。ブラウザウィンドウを終了する（ウィンドウの右上の「×」を押す）と、もし、ブラウザを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待受け画面に戻ります。
 - ブラウザ起動されたアプリケーションを強制終了（端末の「終話ボタン」を押した場合）

ブラウザウィンドウは表示されず、もし、ブラウザを起動する際に待ち受けアプリケーションが動作していたのであればそれを起動し、そうでなければ通常の待ち受け画面に戻ります。

7.4. 各種起動方法による画面遷移

これまで、「iα ppliTool」でのアプリケーションの起動方法には、以下の4つの方法があると説明してきました:

- 通常起動（「通常起動」、「初回起動」、「タイマ起動」、「外部機器起動」）
- 待ち受け起動
- メーラ起動
- ブラウザ起動

ここで、これらを組み合わせた場合の画面遷移を図 7-13 に示します。

「iα ppliTool」実行中、画面に表示されているアプリケーションは、常に1つだけです。

また、図中の「*1」と「*2」は、それぞれ以下に示す条件で起動可能であることを意味します:

- *1:
 - 通常アプリが起動していない
 - 待ち受けアプリケーションが起動していない
 - メーラが起動していない
 - ブラウザが起動していない
- *2:
 - 通常アプリが起動していない
 - 待ち受けアプリケーションが非活性化状態あるいは休眠状態である
 - メーラが起動していない
 - ブラウザが起動していない

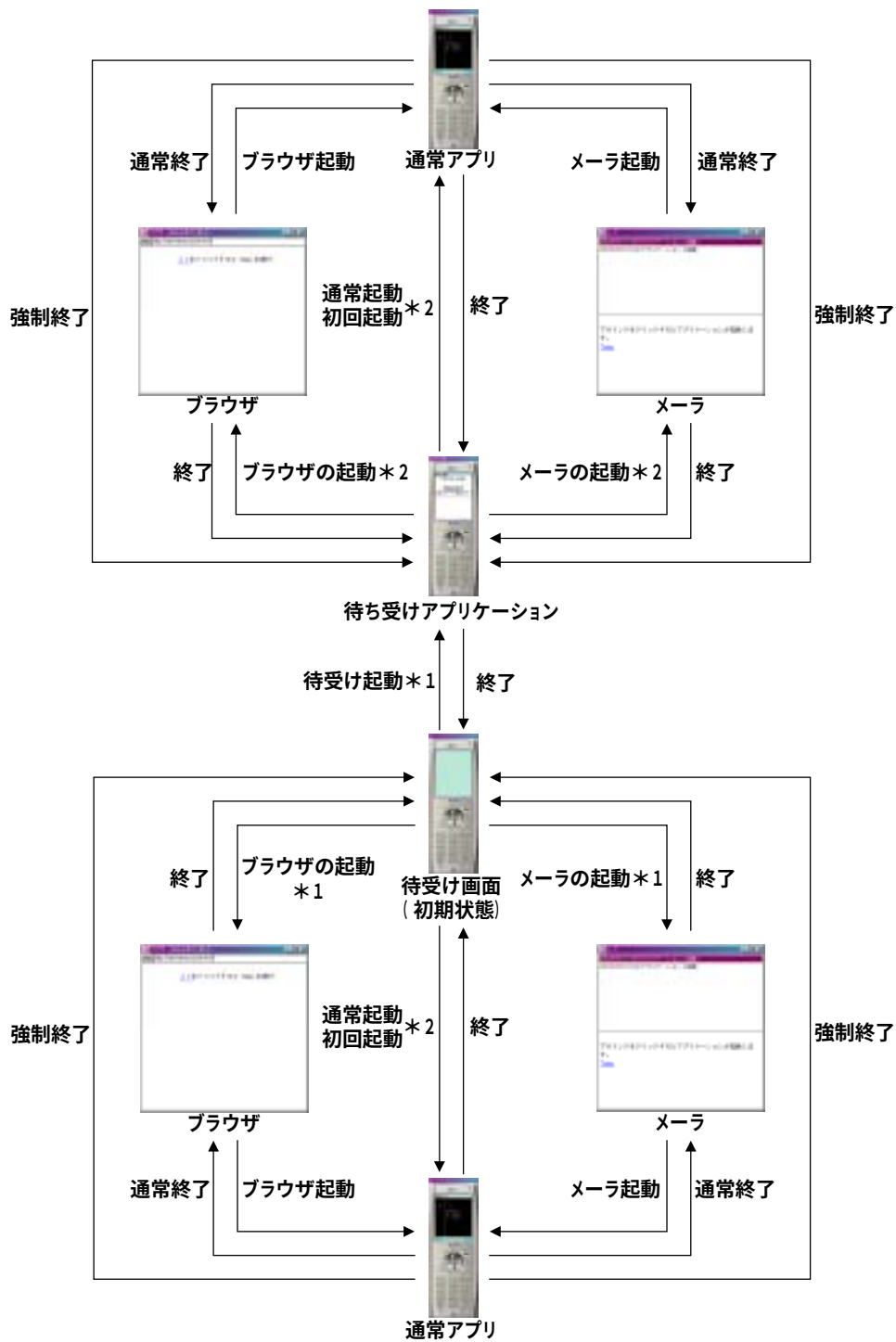


図 7-13 「iαppliTool」における画面遷移

8. ファイルの記述方法

本章では、各種ファイルの記述方法を説明します。各ファイルの場所は、以下の通りです:

「iα ppli Development Kit」を「C:\iDK」にインストールした場合、

- 「*.mail」ファイル: C:\iDK\lib\mail
- 「JAM.properties」ファイル: C:\iDK\lib
- 「HTML」ファイル: Web サーバの適切な場所

8.1. 「*.mail」ファイルの記述方法

「*.mail」ファイルの記述方法は、以下に示す通りです:

```
From: xxx@yyy.ne.jp
To: zzz@docomo.ne.jp
Subject: i アプリ起動( IApplication)
Date: 2001/08/21 12:34
```

…メール本文…

```
--B:A
TEXT="iappli"
ADF="http://*****"
"A"="B"
"C"="D"
```

/** 送信者メールアドレス **/

/** 受信者メールアドレス **/

/** Subject **/

/** Date **/

/** リンクを表示する際の文字列 **/

/** 起動させるアプリケーションの ADF の URL **/

/** アプリケーションに渡すパラメータ **/

/** アプリケーションに渡すパラメータ **/

8.2. 「HTML」ファイルの記述方法

「HTML」ファイルの記述方法は、以下に示す通りです:

```
<HTML>
<BODY>
<OBJECT declare id ="App1"
  data="http://*****"
  type="application/x-jam">
<PARAM NAME="A" VALUE="B">
<PARAM NAME="C" VALUE="D">
</OBJECT>
<A ista="#App1" href="****.html">ここ</A>をクリック
</BODY>
</HTML>
```

/** 起動させるアプリケーションの URL **/

8.3. 「JAM.properties」ファイルの記述方法

「JAM.properties」ファイルの記述方法は、以下の通りです:

```
http\://サーバ名/iappli.jam = iappli.jam
http\://サーバ名/mappli.jam = mappli.jam
```

/** サーバ名と起動したいアプリケーションの ADF
の名称 **/

「http\://(ダウンロード元の URL)=(ローカルにある ADF の名前)」と記述します。

ファイルを複数許可する場合は、1行ずつ記述する。

コロン(:)を区切り記号ではなく、文字として認識させるために必ず「http」の後に「\」をつけること。

8.4. ファイルの記述具体例

「http://www.aaa.bbb.ccc」の「zzz.jam」がダウンロードされていて、メールあるいはブラウザから起動する場合:

- メール起動
 - 「*.mail」ファイルの ADF には、
「ADF="http://www.aaa.bbb.ccc/zzz.jam"」と記述します。
 - 「JAM.properties」ファイルには、
「http\://www.aaa.bbb.ccc/zzz.jam = zzz.jam」と記述します。
- ブラウザ起動
 - 「HTML」ファイルの OBJECT タグの data には、
「data="http://www.aaa.bbb.ccc/zzz.jam"」と記述します。
 - 「JAM.properties」ファイルには、
「http\://www.aaa.bbb.ccc/zzz.jam = zzz.jam」と記述します。

9. スクラッチパッド管理

本章では、「iαppliTool」に含まれている、スクラッチパッド管理ツールについて説明します。

9.1. スクラッチパッド管理ツール

スクラッチパッド管理ツールは、アプリケーションのスクラッチパッドファイルを 16 進数でダンプする非常にシンプルなツールです。

スクラッチパッド管理ツールを使用すれば、アプリケーションからスクラッチパッドにデータが適切に書込まれているかどうかを確認できます。また、スクラッチパッドを削除することもできます。

9.2. スクラッチパッドファイル

スクラッチパッドは、実際の端末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。スクラッチパッドには、文字列以外にアプリケーションで使用可能な基本タイプのデータを格納できるため、通常のテキストビューアなどで表示することができません。

エミュレータは、スクラッチパッドファイルを以下のディレクトリに格納します:

「iαppli Development Kit」を C:\iDK にインストールした場合
“C:\iDK\apps\<プロジェクト名>\sp”

スクラッチパッドファイルは<プロジェクト名>.scr です。

9.3. スクラッチパッド管理ツールの起動

1. メニューから「プロジェクト」→「スクラッチパッド...」を選択します。

「スクラッチパッド管理ツール」ウインドウが表示されます（図 9-1）。



図 9-1 スクラッチパッド管理ツール

9.4. スクラッチパッドの内容表示

図 9-1 に示すように、スクラッチパッド管理ツールを起動すると、現在アクティブなプロジェクトのスクラッチパッドの内容が、16進数で表示されます。ただし、内容の編集はできません。

また、他のプロジェクトのスクラッチパッドの内容を表示するには

1. 「読み込み」ボタンを押します。

「スクラッチパッドファイル選択」ダイアログが表示されます（図 9-2）。

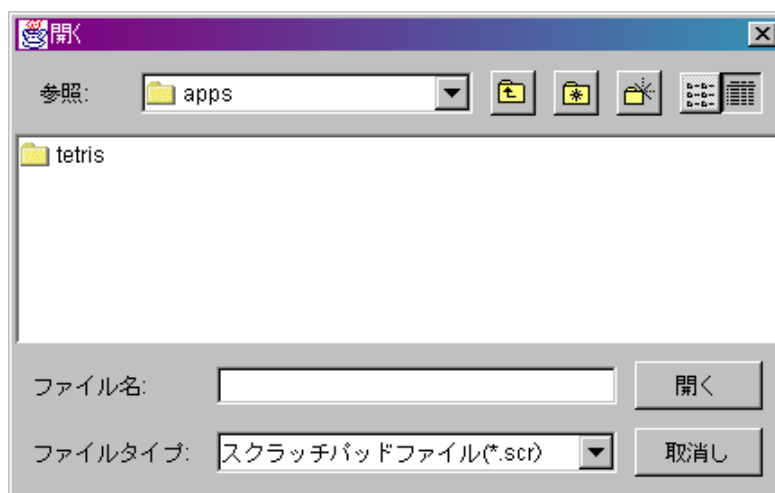


図 9-2 スクラッチパッドファイル選択ダイアログ

2. 内容を表示するスクラッチパッドファイルを選択し、「開く」ボタンを押します。

図 9-1 のスクラッチパッドの内容表示エリアに、スクラッチパッドの内容が表示されます。

9.5. スクラッチパッドの削除

スクラッチパッドを削除するには、図 9-1 の「削除」ボタンを押します。

現在、内容が表示されているスクラッチパッド（ファイル）が削除されます。

また、メニューから「プロジェクト」→「スクラッチパッドの削除」を選択することで、現在読込まれているプロジェクトのスクラッチパッドを削除することもできます（スクラッチパッドが存在しない場合、「スクラッチパッドがありません」というダイアログが表示されます）。

9.6. スクラッチパッド管理ツールの終了

スクラッチパッド管理ツールを終了するには、図 9-1 の「終了」ボタンを押します。

スクラッチパッド管理ツールウィンドウが閉じます。

10. 証明書の管理

「ia ppliTool」上で HTTPS 通信を行う際に使用する証明書は、OS 組み込みのものを使用します。よって、「ia ppliTool」では管理しません。

そこで、証明書を管理するためには OS にあらかじめインストールされている証明書管理アプリケーションを使用します。

1. 「コントロールパネル」→「インターネットオプション」をダブルクリックします(図 10-1)。

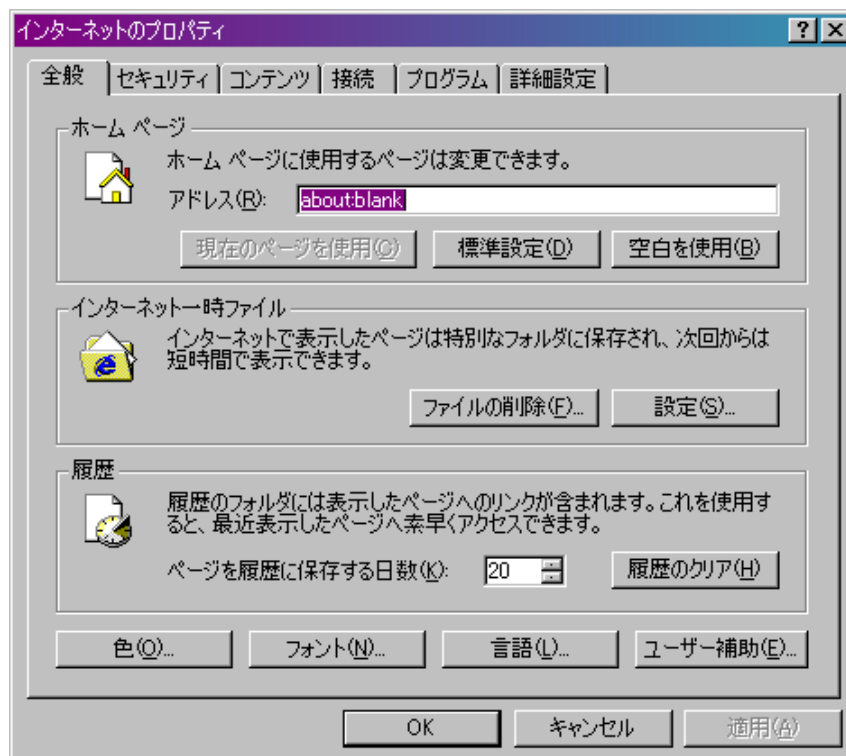


図 10-1 インターネットのプロパティ

2. 図 10-1 の「コンテンツ」タブを選択します (図 10-2)。

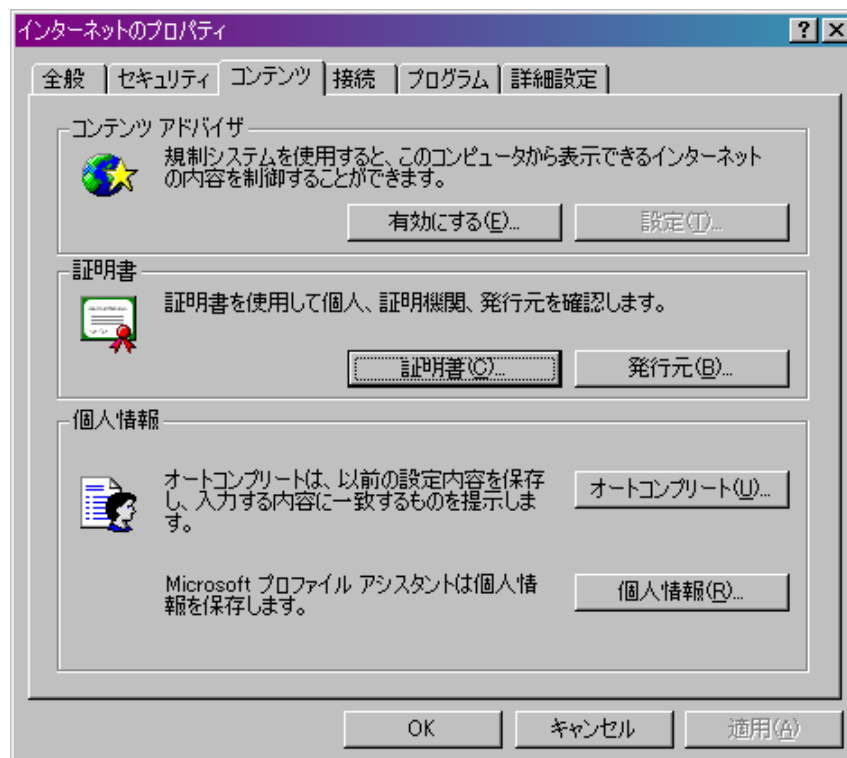


図 10-2 インターネットのプロパティ（コンテンツ）

3. 図 10-2 の「証明書 (C) ...」ボタンを押します（図 10-3）。

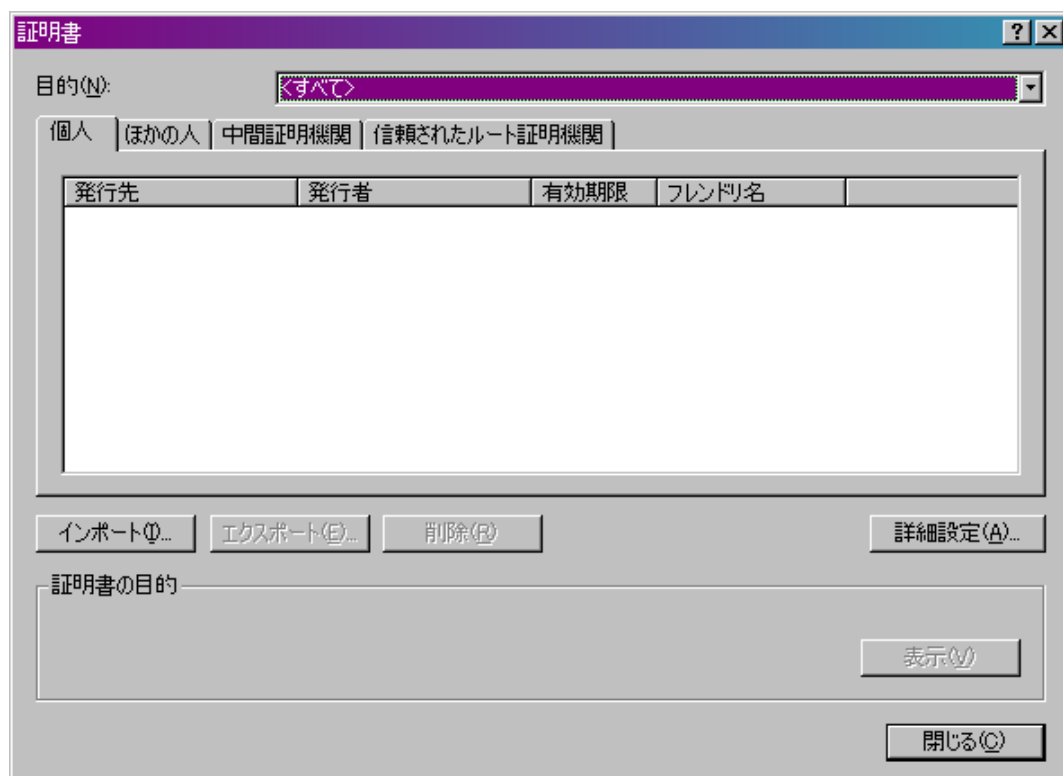


図 10-3 証明書ウィンドウ

4. 図 10-3 の証明書ウィンドウで証明書の管理を行います。使用方法については、OS のドキュメントを参照してください。

11. OBEX、カメラ API、ポインティングデバイスの利用

11.1. OBEX の利用

11.1.1. TCP/IP のみのリリース向け

「iα ppli Development Kit」では、TCP/IP を用いて OBEX 通信することができます。赤外線を用いた OBEX 通信はできません。
通信先のホストの設定は、「iα ppliTool」のメニューから「その他」→「ネットワークの設定...」の選択で表示される画面の「OBEX 設定」で行います。何も設定しない場合には、127.0.0.1（ローカルホスト）に通信します。使用される TCP のポート番号は 650 番です。ポート番号を変更することはできません。Forte for Java での設定は「12.4.1. デバイスの設定」を参照してください。

11.1.2. TCP/IP、IrDA の両方を含むリリース向け

「iα ppli Development Kit」では、TCP/IP もしくは赤外線を用いて OBEX 通信することができます。
TCP/IP を用いて OBEX 通信を行う場合、「iα ppliTool」の実行ファイルが置かれているディレクトリ（「c:\iDK」にインストールした場合、「c:\iDK\bin」に置かれる）にある libobex_tcp.dll を libobex.dll に名前を変更してください。通信先のホストの設定は、「iα ppliTool」のメニューから「その他」→「ネットワークの設定...」の選択で表示される画面の「OBEX 設定」で行います。何も設定しない場合には、127.0.0.1（ローカルホスト）に通信します。使用される TCP のポート番号は 650 番です。ポート番号を変更することはできません。Forte for Java での設定は「12.4.1. デバイスの設定」を参照してください。
赤外線を用いて OBEX 通信を行う場合、「iα ppliTool」の実行ファイルが置かれているディレクトリにある libobex_irda.dll を libobex.dll に名前を変更してください。赤外線を利用する場合は、ホストの設定の必要はありません。
なお、インストール時の libobex.dll は libobex_tcp.dll であり、TCP/IP での通信となります。

「iα ppli Development Kit」と移動機間で赤外線による OBEX 通信は、Windows の DLL に修正が必要となるため、正式にはサポートしておりません。なお、「iα ppli Development Kit」同士の赤外線による OBEX 通信は可能です。

11.2. カメラ API の使用

「iα ppliTool」には、カメラ機能（静止画、動画の撮影機能）がないため、ファイルから静止画及び動画を読み込むことにより対応します。

まず、静止画を撮影する（取込む）ためには、com.nttdocomo.opt.ui.Camera クラスの takePicture() メソッドを呼出すことにより図 11-1 に示す「ファイル選択ダイアログ」が表示されるので、目的の静止画を選択することにより取込み可能となります。

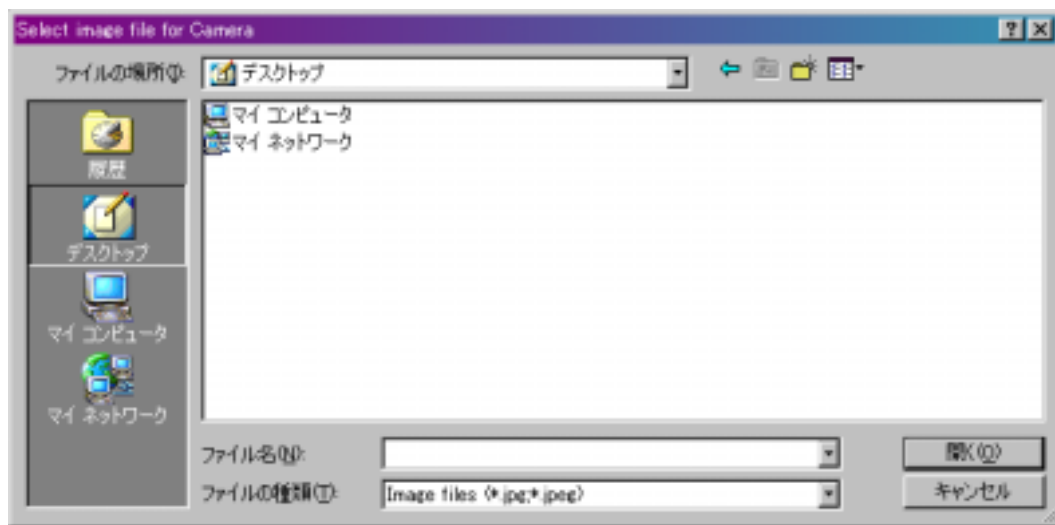


図 11-1 ファイル選択ダイアログ（静止画）

また、動画を撮影する（取込む）ためには、com.nttdocomo.opt.ui.Camera クラスの takeMovie() メソッドを呼出することにより図 11-2 に示す「ファイル選択ダイアログ」が表示されるので、目的の動画を選択することにより取込み可能となります。

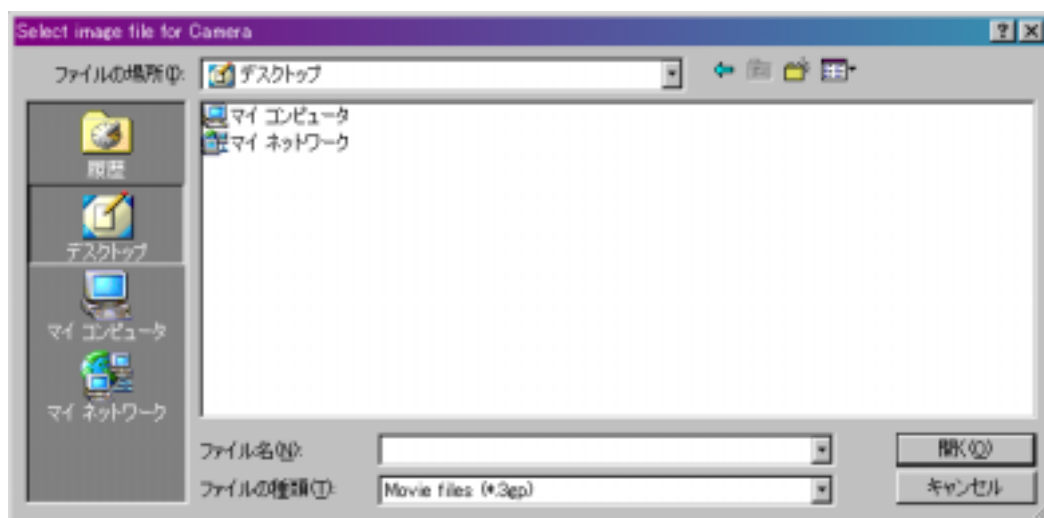


図 11-2 ファイル選択ダイアログ（動画）

取込んだ静止画については「iαppliTool」にて表示可能ですが、動画については表示（再生）できません。

11.3. ポインティングデバイス API の使用

「iαppliTool」には、ポインティングデバイス機能がないため、「iαppliTool」の決定キー上を右クリックしながらマウスをドラッグすることによりポインティングデバイスを操作します。

カレントフレームを Canvas に設定し、com.nttdocomo.opt.ui.PointingDevice クラスの setEnabled() メソッドでポインティングデバイスを有効にしてポインティングデバイス进行操作すると画面上にカーソルが表示されます。getX(),getY()メソッドを呼び出すことで画面上のカーソルの位置を取得することが可能です。



図 11-3 画面上に表示されたカーソル

12. Forte for Java Community Edition の使用

12.1. はじめに

この章は、次のトピックから構成されています。

- 「J2ME Module for iappli」のインストール
- 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しい説明

12.1.1. Forte for Java

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 Module for iappli」は、i アプリ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詳細なドキュメントは、Forte for Java の Web サイト <http://www.sun.com/ffj/> を参照してください。

表 12-1 に、この章で使用する用語の定義を示します。

表 12-1 用語の定義

用語	定義
アプリケーション	エミュレータ上で実行される i アプリ
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

12.2. 「J2ME Module for iappli」のインストール

本節では、Forte for Java 用の J2ME Module をインストールする手順を説明します。

12.2.1. 始める前に

Forte for Java で「J2ME Module for iαppli」を使用するには、Forte for Java と Java 2 Platform, Standard Edition SDK をインストールしてから、「J2ME Module for iαppli」をインストールする必要があります。

Forte for Java は、次の Web サイト (<http://www.sun.com/ffj/>) からダウンロードできます。J2SE SDK は、Sun の Java ソフトウェアの Web サイト (<http://java.sun.com/j2se/1.3/>) からダウンロードできます。

12.2.2. 「J2ME Module for iαppli」のインストール

「iαppli Development Kit」のインストーラを使用して、Forte for Java 用のモジュールをインストールします。インストーラの「インストール先の選択」画面に続いて、「J2ME Module for iαppli」をインストールするためのオプションが表示されます。

「J2ME Module for iαppli」をインストールするための画面を次の図に示します。

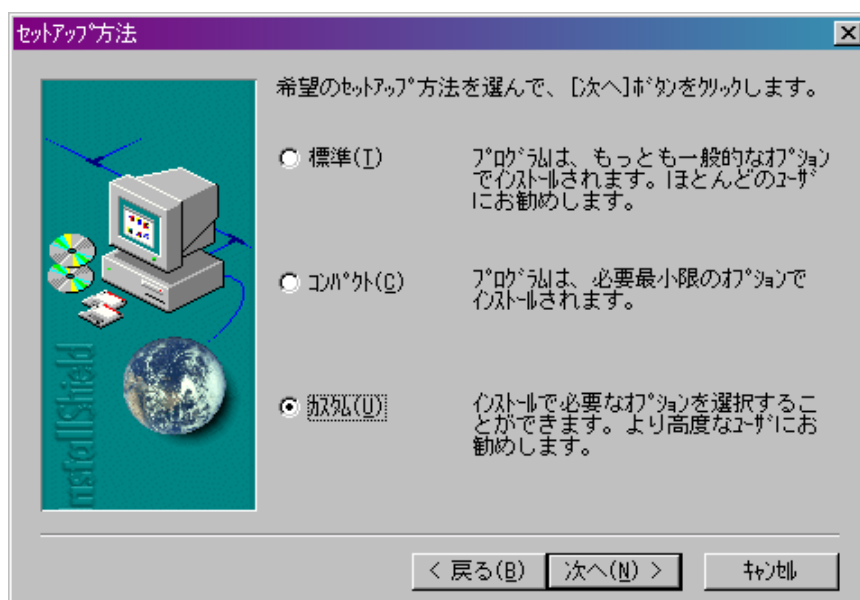


図 12-1 カスタムを選択

「セットアップ方法」画面で「カスタム」オプションを選択し、「次へ」をクリックします。

コンピュータにインストールされている Forte for Java が自動的に検出され、モジュールがインストールされます。

インストールプログラムが Forte for Java を検出できなかった場合は、「Forte インストールディレクトリの指定」画面にて次の手順を実行します。

- A) 「参照」をクリックし、「ディレクトリの選択」ダイアログを開きます。
- B) Forte for Java のインストール先のディレクトリを選択し、「OK」をクリックします。
- C) 「次へ」をクリックし、次の画面に進みます。これで、「J2ME Module for iαppli」が「iαppliTool」とともにインストールされます。

注意：iappli モジュールを、Sun から提供される MIDP 向け開発環境「J2ME Wireless Toolkit」の提供する Forte モジュール (kjava) がインストールされている Forte 環境で使用する場合は kjava モジュールを無効にしてから使用してください。また、kjava モジュールを使用する場合は iappli モジュールを無効にしてから使用してください。モジュールを無効にする方法については、Forte のドキュメントを参照してください。

12.3. 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
- 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。

12.3.1. パッケージの作成

アプリケーションのソースファイルと属性ファイルを収容するパッケージを作成します。
IDE のエクスプローラ[ファイルシステム]からパッケージを作成するには、次の手順に従います。

1. sampledир ファイルシステムを右クリックします。

メニューが表示されます。



図 12-2 エクスプローラウィンドウ

2. 「新規パッケージ」を選択します。
「新規パッケージを作成」ダイアログが表示されます。
3. パッケージ名を入力し、「了解」をクリックします。

たとえば、パッケージ名を hello と入力します。これで hello という名前のフォルダが作成されます。

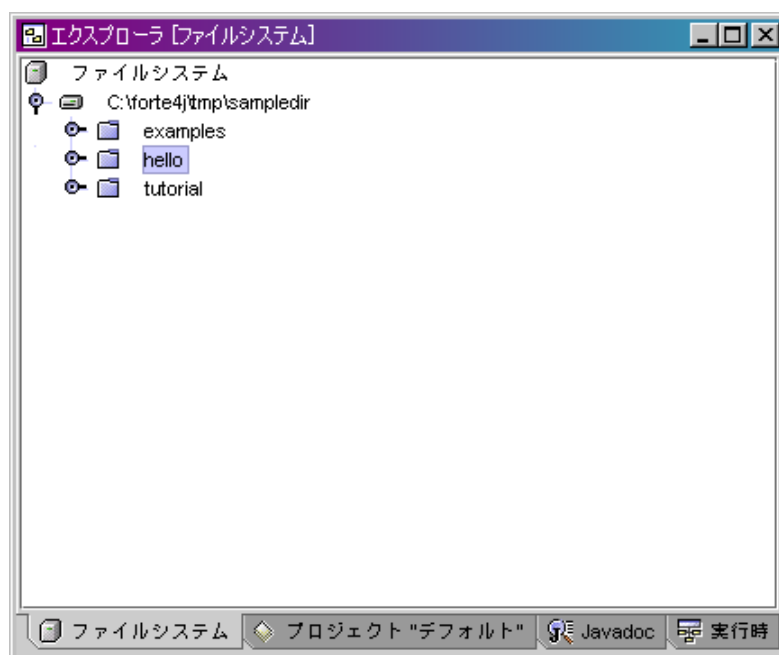


図 12-3 エクスプローラで新しいパッケージを作成

12.3.2. アプリケーション・ディスクリプタ・ファイル (ADF) の作成

IDE のエクスプローラから ADF を作成するためには次の手順に従います。

1. アプリケーションを収容するために作成したパッケージを右クリックします。

前の例で作成した hello をクリックし、メニューを表示させます。

2. 「新規」 → 「DoJa」 → 「DOJA」を選択します。

「テンプレートウィザードから新規作成」ウィザードが起動します。

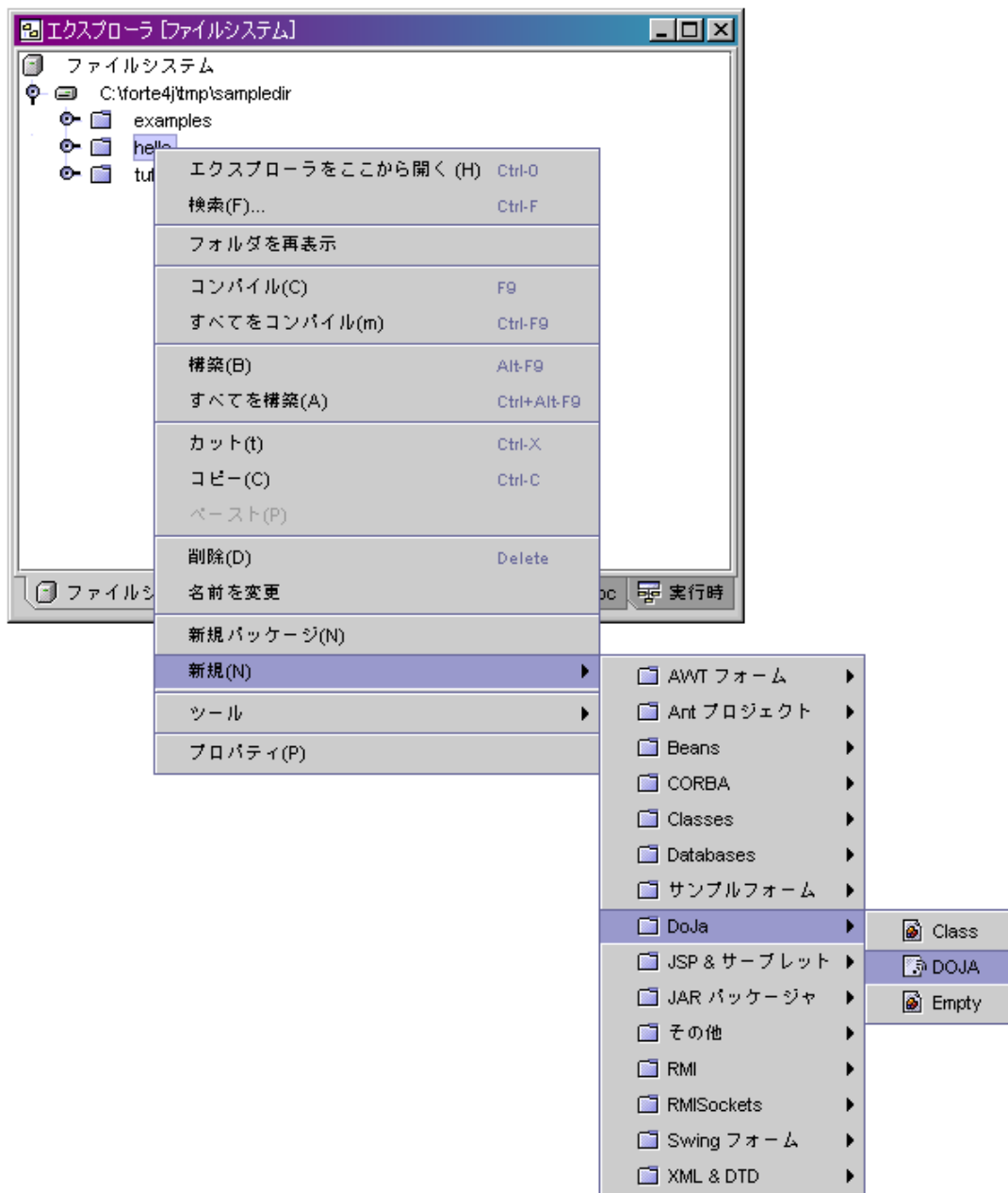


図 12-4 ADF の新規作成

- 作成する ADF の名前を入力し、「次へ」をクリックします。

たとえば hello と入力します。

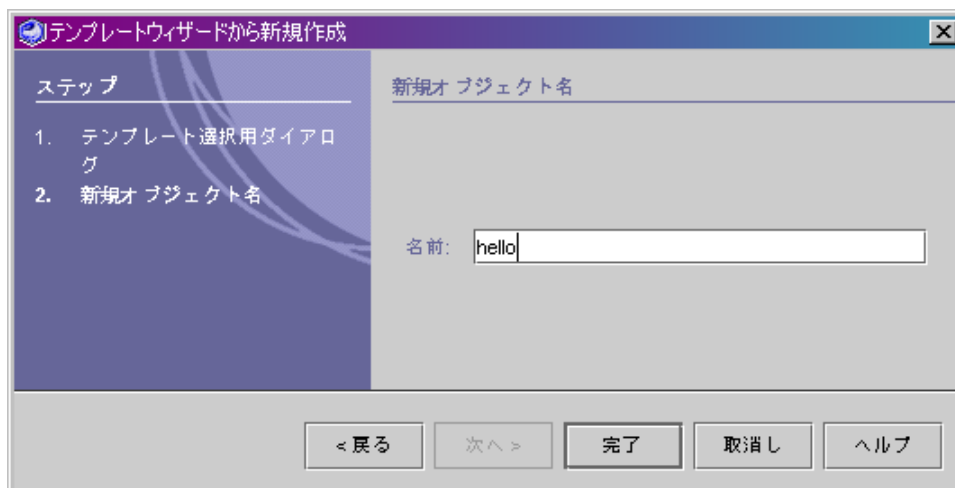


図 12-5 テンプレートウィザード

4. 「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 hello が作成されます。

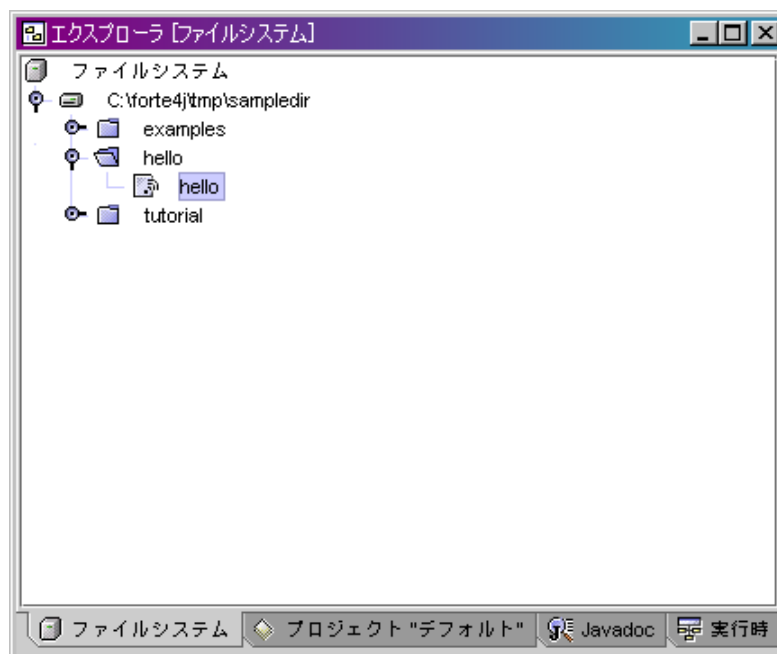


図 12-6 ADF ファイルが表示されたエクスプローラ画面

12.3.3. ソースファイルの作成

1. アプリケーションのソースファイルを作成します。

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、「新規」 → 「DoJa」 → 「Class」を選択します。

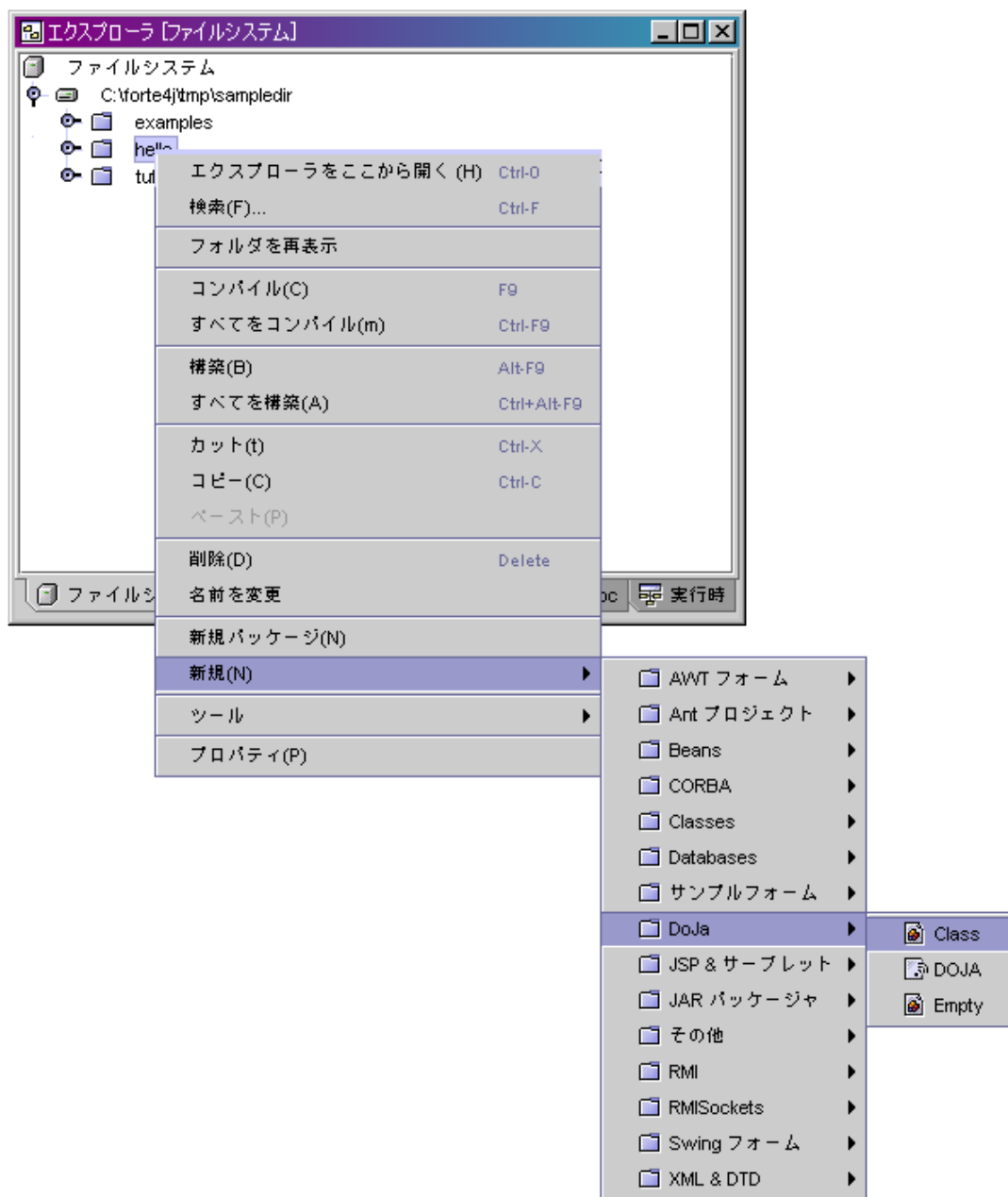


図 12-7 ソースファイルの作成

「テンプレートウィザードから新規作成」ウィザードが起動されるので、新規オブジェクト名にアプリケーションのソースファイル名を指定して「完了」をクリックします。

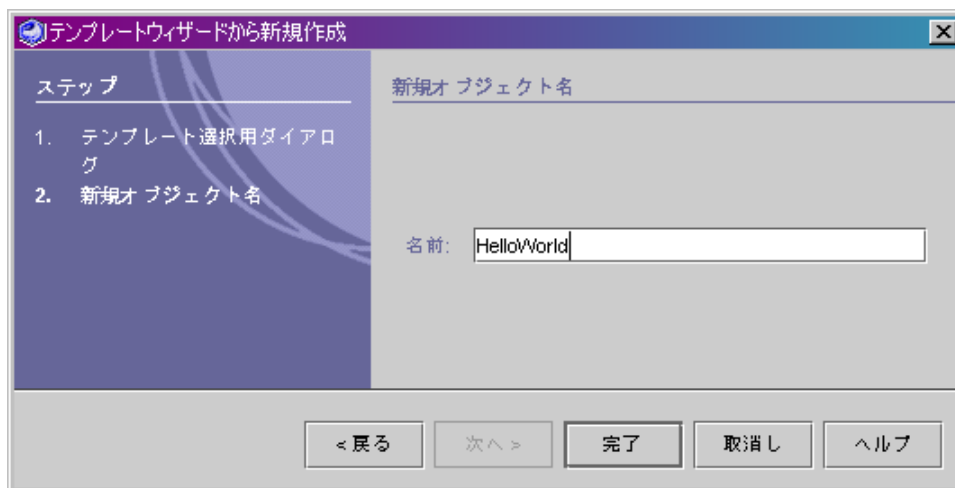


図 12-8 ソースファイル名の指定

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。

2. ソースファイルを編集します。

ソースファイルに「`System.out.println("HelloWorld");`」を追加します。

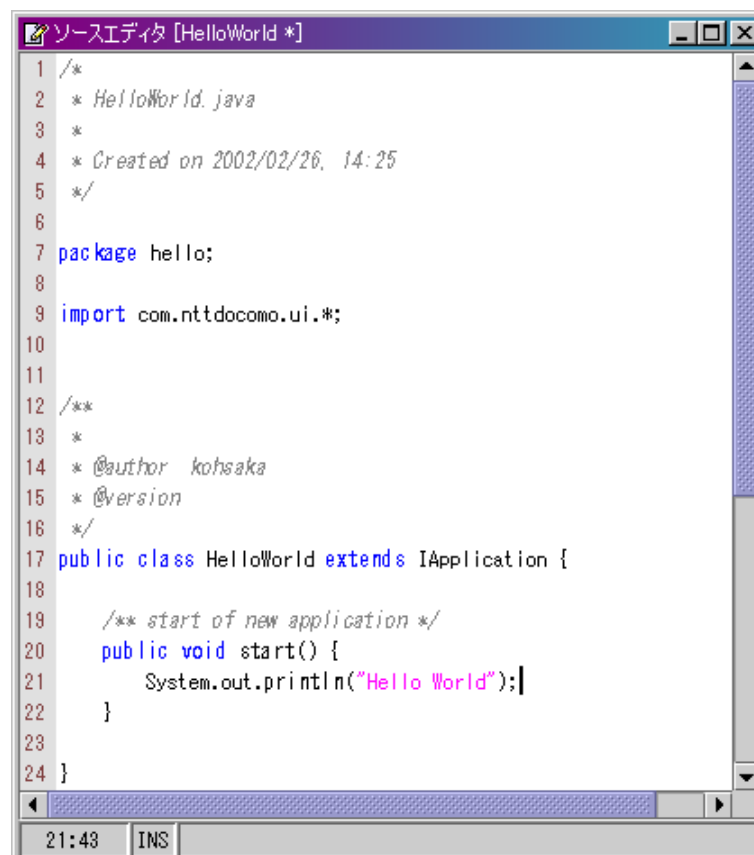


図 12-9 ソースファイルの編集

3. ソースファイルを保存します。

「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。

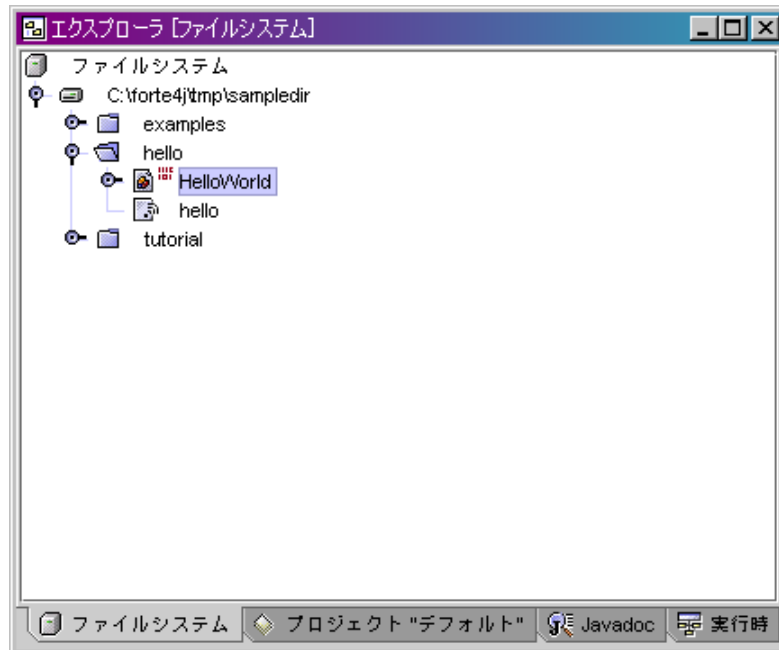


図 12-10 ソースファイルの追加

注意：アプリケーションの起動クラスの名前は、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。

12.3.4. ADF の設定

「エクスプローラ[ファイルシステム]」画面で ADF hello を選択し、右ボタンをクリックしてメニューを表示させて、「開く」を選択します。

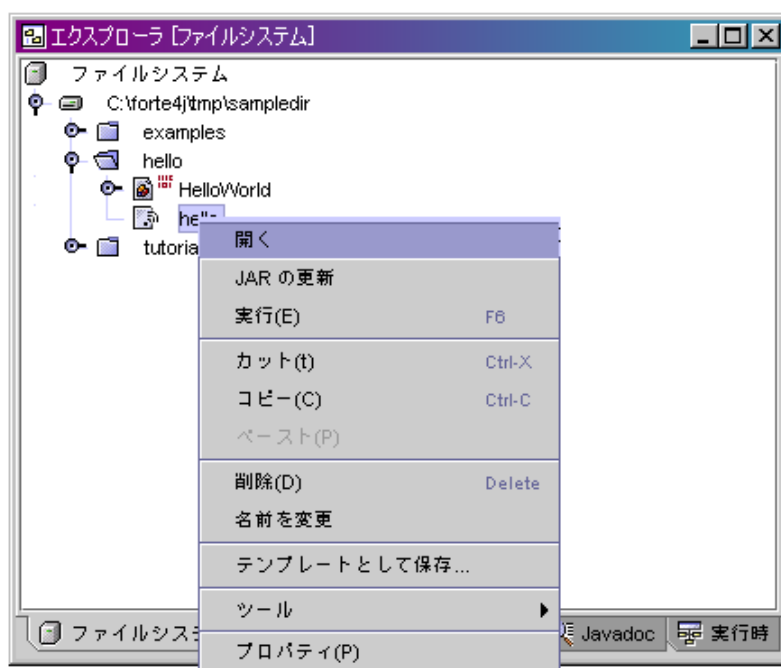


図 12-11 ADF のオープン

ADF 設定ダイアログが表示されます。



図 12-12 ADF 設定画面

設定項目は、表 12-2 に示すとおりです。

表 12-2 ADF の設定項目

設定項目	説明
必須フィールド ^[7]	
AppName	アプリケーションの名前（最大 16 バイト）。
AppClass	アプリケーションの main クラス。 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パッケージの名前になっています。このフィールドは実際のメインクラス名に変更して下さい。
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。 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。

注意：オプションのフィールドについては、「5.3. アプリケーションディスクリプタファイル（ADF）の編集」を参照してください。

12.3.5. jar ファイルの内容の設定

ここでは、jar ファイルの内容を指定する方法について説明します。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. ADF をオープンし、「コンテンツ」タブをクリックします。

^[7] 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。

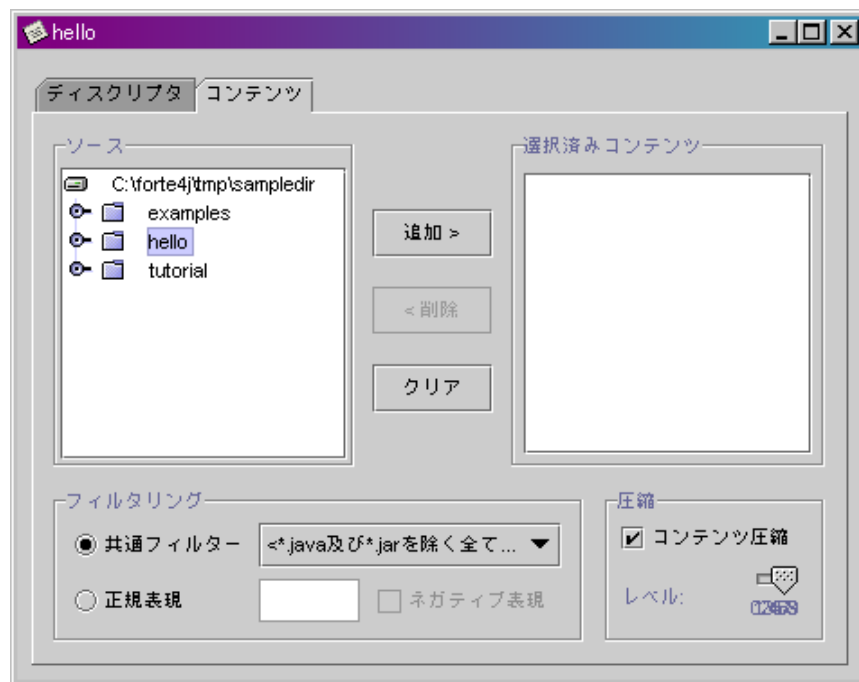


図 12-13 コンテンツの指定

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。

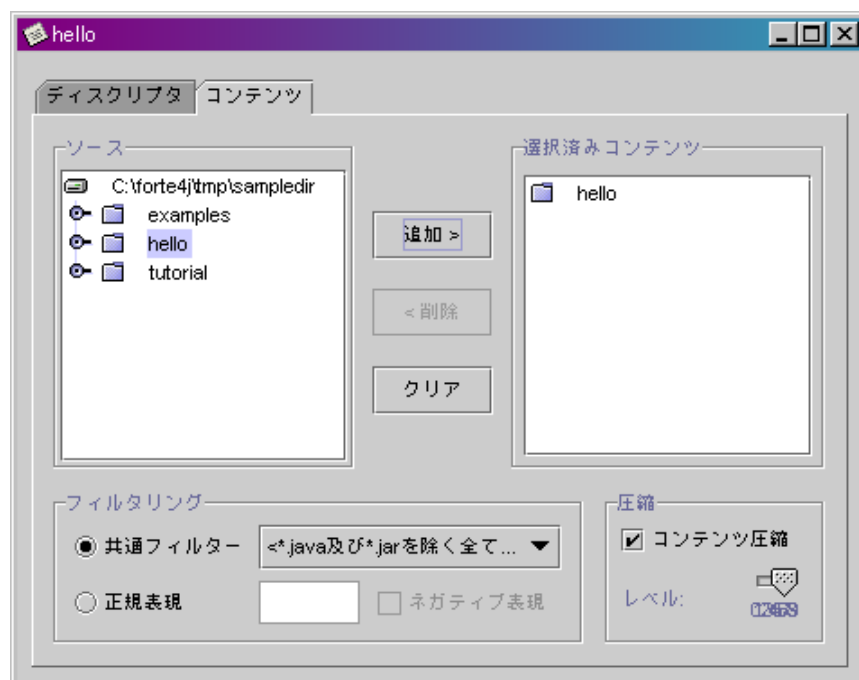


図 12-14 コンテンツを Jar ファイルに転送

12.3.6. アプリケーションの構築と実行

アプリケーションを構築および実行するには、IDE のエクスプローラ[ファイルシステム] 画面にて、ADF を選択してから右ボタンをクリックし、「実行」を選択します。

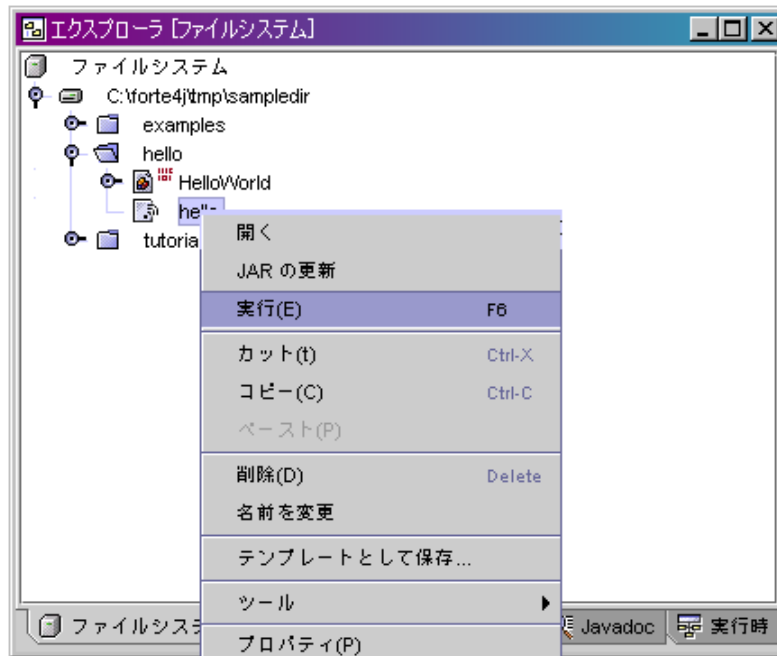


図 12-15 アプリケーションの実行

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。

12.4. 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。

- デバイスの設定
- プロキシの設定
- トレースの設定

12.4.1. デバイスの設定

「J2ME Module for iαppli」では、エミュレータのデバイスの設定を行うことができます。設定を行うには、次の手順に従ってください。

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」→「iαppli Emulator」を右クリックし、次に「カスタマイズ」をクリックします。

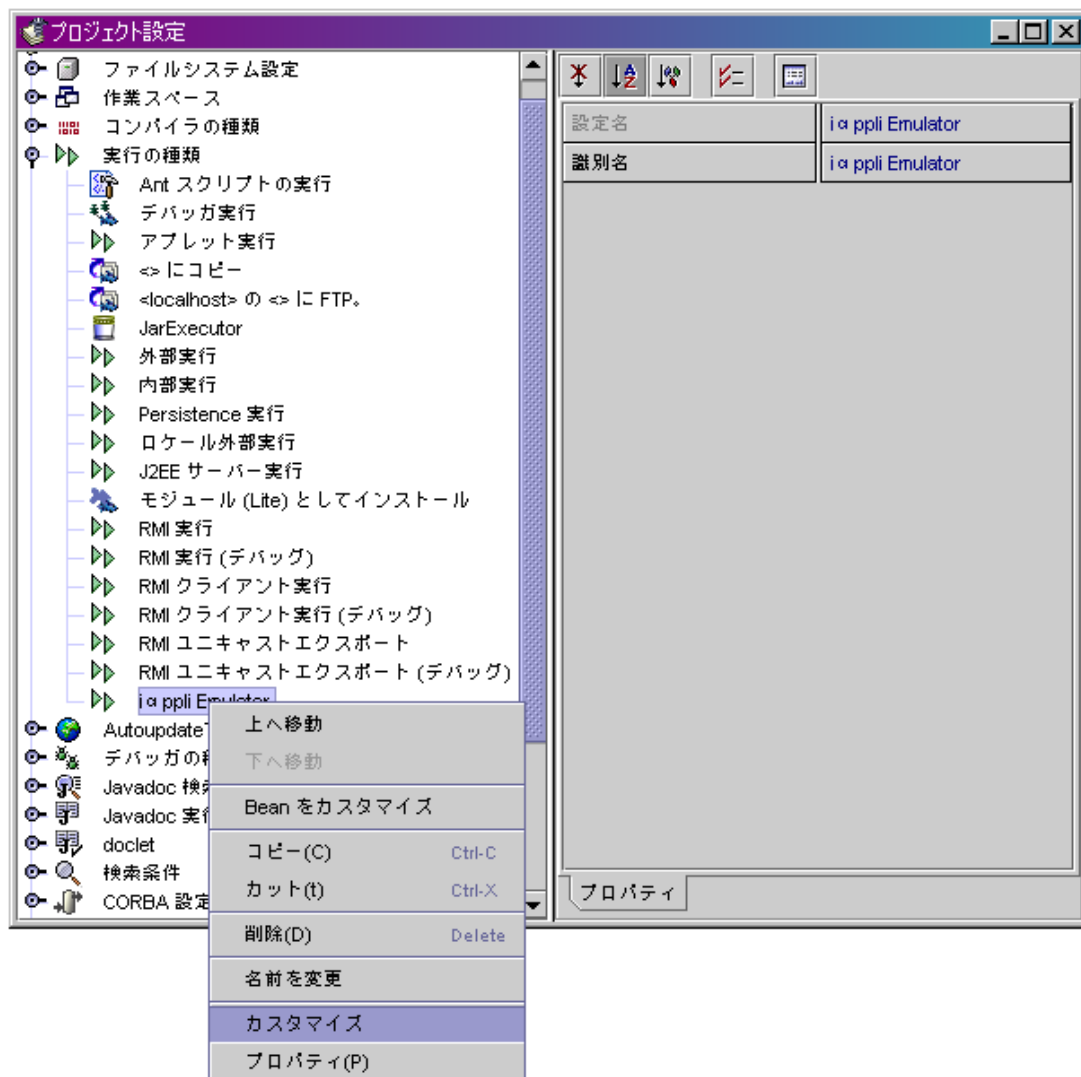


図 12-16 エミュレータの設定

「カスタマイザダイアログ」(図 12-17)が表示されます。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。

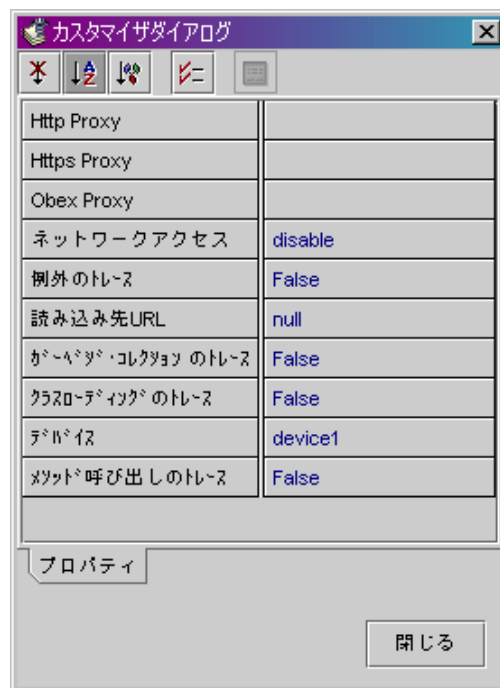


図 12-17 エミュレータの設定画面

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 12-18）。

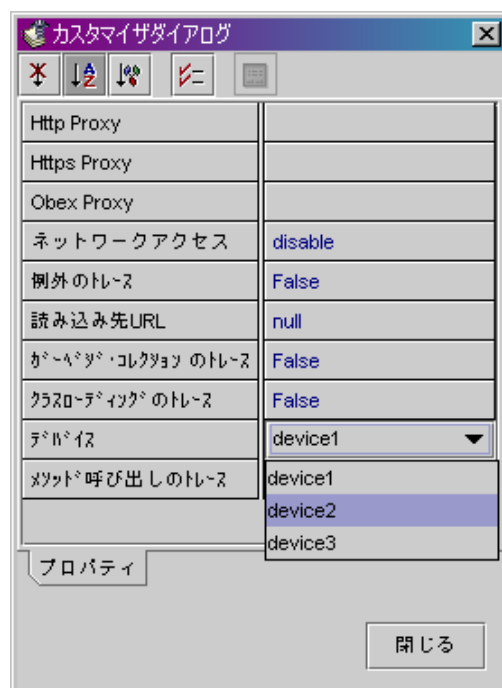


図 12-18 デバイスの変更

エミュレータ設定項目は表 12-3 のとおりです。

表 12-3 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。	device1
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
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	disable

	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。	
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。	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。	なし
Obex	アプリケーションが OBEX を使用して通信する際の IP アドレス。OBEX over TCP を用いる場合に、接続先ノードの IP アドレス（ホスト名）を指定します。指定がない場合は、127.0.0.1（ローカルホスト）に接続を試みる試みをします。なお、本設定は、ObexClientConnection クラスの connect メソッドで意味をなします。「11. OBEX の利用」を参照。	なし
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	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	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	オフ
例外のトレース	例外を表示します	オフ

12.4.2. 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設定画面(図 12-19)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。



図 12-19 プロキシの設定

12.4.3. ログメッセージのトレース

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

エミュレータの設定画面（図 12-17）で、該当するエントリをオンにしてください。

付録. エミュレーション端末の設定

エミュレーション端末設定ファイルを用意することで、任意の端末画像を使ってエミュレータを実行することが可能になります。エミュレーション端末設定ファイルは "lib\skin\端末名\skin.conf" です。

1. コメント行

先頭に # がある行はコメント行として無視されます。

2. エミュレータウィンドウの設定

```
screen_width = [エミュレータウィンドウの幅]
screen_height = [エミュレータウィンドウの高さ]
default_image = [スキン画像ファイル名]
pressed_image = [ボタンが押された状態のスキン画像ファイル名]
```

通常エミュレータウィンドウの幅と高さはスキン画像の幅と高さに合わせる。

3. LCD 領域の設定^[8]

```
lcd_position = [LCD 位置の X 座標], [LCD 位置の Y 座標]
LCD の位置はエミュレータウィンドウの左上を原点とする。
lcd_width = [LCD の幅]
lcd_height = [LCD の高さ]
lcd_system_height = [LCD のシステムアイコン表示領域の高さ]
lcd_system_width = [アンテナアイコンの幅], [バックライトマークの幅], [パイプリータマークの幅], [スクロールマークの幅], [I アプリマークの幅]
lcd_color = [RED], [GREEN], [BLUE]
system_image = [システムアイコン画像ファイル名]
```

4. フォントの設定

```
font_size = [スモールフォントサイズ], [ミディアムフォントサイズ], [ラージフォントサイズ]
font_size を指定しない場合のフォントサイズは、10, 12, 14 になります (Windows2000)。
```

注意：実際にエミュレータで利用されるフォントサイズはエミュレータの実行環境に依存します。

^[8] Display.getWidth()は lcd_width の値と同じになります。Display.getHeight()は、lcd_height の値から、lcd_system_height の値とソフトウェア領域の高さ（デフォルトフォントの高さ）を引いた値になります。

5. ボタン設定

- システムボタンの設定

system_button = [ボタン番号], [ボタン位置の X 座標], [ボタン位置の Y 座標], [ボタンの幅], [ボタンの高さ]

- 通常ボタンの設定

button = [ボタン番号], [ボタン位置の X 座標], [ボタン位置の Y 座標], [ボタンの幅], [ボタンの高さ]

ボタンの位置はエミュレータウィンドウの左上を原点とする。

システムボタンのボタン番号は

- 1 --- 通話開始ボタン
- 2 --- クリアボタン
- 3 --- 通話終了ボタン

通常ボタンのボタン番号は DOJA のキー番号に対応している。

6. 例

```
# skin.conf
# Unknown key KEY_F00KUP
# Unknown key KEY_F00KDOWN
screen_width=216
screen_height=619
default_image = device1-off.jpg
pressed_image = device1-on.jpg
system_image = system.gif
warning_image= warn.gif
error_image=error.gif
info_image=info.gif
lcd_color=186, 231, 217
lcd_position = 27, 63
lcd_width = 160
lcd_height = 207
lcd_system_height = 15
lcd_system_width = 36, 12, 13, 11, 20
system_button=1, 31, 431, 44, 25
system_button=2, 86, 431, 44, 25
system_button=3, 141, 431, 44, 25
button=9, 141, 521, 42, 25
button=21, 33, 324, 55, 26
button=19, 98, 378, 21, 21
button=26, 93, 405, 28, 23
button=22, 129, 324, 54, 26
button=16, 69, 358, 26, 17
button=0, 86, 551, 44, 25
button=10, 31, 551, 44, 25
```

button=1, 31, 461, 44, 25
button=30, 42, 581, 24, 22
button=24, 94, 303, 28, 23
button=2, 86, 461, 44, 25
button=27, 33, 382, 55, 26
button=11, 141, 551, 42, 25
button=3, 141, 461, 42, 25
button=31, 149, 581, 25, 22
button=18, 122, 358, 26, 17
button=4, 31, 491, 44, 25
button=28, 129, 382, 54, 26
button=5, 86, 491, 44, 25
button=6, 141, 491, 42, 25
button=17, 98, 332, 22, 22
button=7, 31, 521, 44, 25
button=20, 97, 357, 23, 20
button=8, 86, 521, 44, 25