iモードJavaがいますぐ作れる！配信できる！

iアプリ作成塾

木寺祥友＋株式会社エル・カミノ・リアル＋編集部

ついに登場！ ケータイ用Javaアプリケーション

2001年1月26日、ついにNTTドコモからJavaアプリケーション「iアプリ」に対応したP503iとF503iが発売された。これにより携帯電話でもパソコンのように、さまざまなアプリケーションをダウンロードして「インストール」できるようになった。
すでにNTTドコモの公式サイト以外の、いわゆる「勝手サイト」から次々と独自のiアプリが公開されていることからもわかるように、iアプリは単にダウンロードして「使う」だけでなく、自分で独自アプリケーションをプログラムして、多くのユーザーに配布もできるのだ。iアプリを作るには、その基礎となる「Java」をある程度理解したうえで、iアプリ独特の作法も学ばねばならないが、自分が作ったアプリケーションが携帯電話で動くのを見れば感動すること間違いない。
ここでは、公式iアプリに飽き足らないこだわり派のためにiアプリの作り方を一から指南する。必要なツールからプログラミング上の注意点まで詳細に解説するので、これを読めば誰でも自分専用のiアプリが作れるだろう。

illustration : Hada Eiji

Javaといっても恐れるに足らず！

iアプリは誰でも作れる

iアプリを作るには、まずベースとなるプログラム言語「Java」を覚えなければならない。そういうと敷居が高いと思うかもしれないが、Javaはプログラムの機能（オブジェクト）を順番に記述することでアプリケーションを開発する「オブジェクト指向」という手法を使った比較的簡単な言語だからそれほど心配はいらない。

また、iアプリの開発に使うのはJavaのなかでもJ2MEと呼ばれる携帯端末向けのプラットフォームだ。携帯電話の上で実行できるのはサーバー向けのJ2EEやパソコン、ワークステーション向きのJ2SEと比べて、はるかに機能が少ない反面、覚えることも少なくてすむ。

しかも、iアプリの作成に必要なツール類はすべて無料で配布されており、特別な費用は必要ない。やろうと思い立ったその日から誰にでも作れるものなのだ。近い将来にはウェブ用の「HTMLエディター」のようなiアプリ専用の開発ツールソフトも登場すると思われる。そうなればキーを叩いてJavaをプログラムしなくてもドラッグ＆ドロップでiアプリを作れるようになるだろう。

●iアプリ作成に必要なもの

まず、製作環境としては普通のウィンドウズやLinux、またはUNIX系のマシンがあればOKだ。
残念ながら、現在マッキントッシュでは作れない。MacOS X 正式版を待とう。

□ エディター

Javaのソースコード（.Javaファイル）は、テキストエディターや各社から発売されているJava開発ツールで書く。ウィンドウズ付属のメモ帳でも十分だ。なお、本誌CD-ROMにもエディターの定番「秀丸エディタ」を収録している（375ページ参照）。

□ JDK1.3

サン・マイクロシステムズ株式会社から提供されているJava開発者向けのツールキット。Javaコンパイラー、ファイルを圧縮するためのJavaアーカイバーが含まれている。本誌CD-ROMに収録（374ページ参照）。

□ CLDC 1.0

同じくサン・マイクロシステムズ株式会社から提供されている携帯端末向けのツールキット。事前検証ツールが含まれている。無料でダウンロードできる。

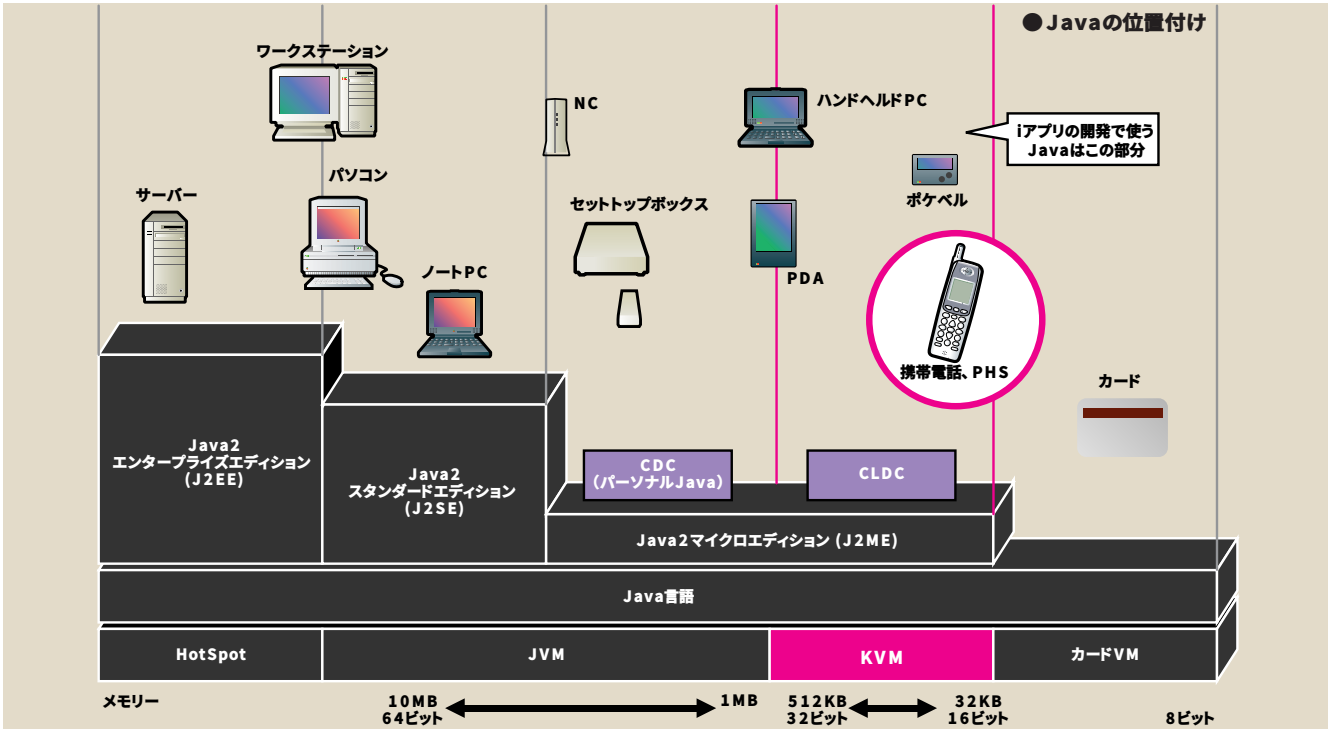
Jump jdc.sun.co.jp/wireless/

□ iモードJava拡張APIクラスライブラリー
□ iモードJava文字コンバーター

ソースコードをアプリケーションにコンパイルするにはこれらが必要だが、2月15日現在はまだ一般には公開されていない。そこで、本記事では代わりゼンテックのiアプリ用エミュレータ「i-JADE」を使ってコンパイルした（213ページ参照）。なお、NTTドコモのウェブでは近日中に公開されることが記載されている。

Jump www.nttdocomo.co.jp/mc-user/i/java/
Jump www.zentek.com/i-JADE/ja/download.html

本誌CD-ROMに収録！



HTMLエディター：ウェブページはHTMLで記述されているが、HTMLタグを知らない人でもウェブページを作れるように設計されたウェブページ作成のための専用ソフトのこと。

準備編

知らなきゃなにもはじめられない

iアプリ開発者のための基礎用語

iアプリの開発は誰にでもできるものではあるが、やはり基礎的なJavaの知識はどうしても必要になる。文法などはJavaの入門書を参照してもらうとして、ここではこれからこの記事を読むにあたって、とにかく知っておかなければ何を書いてあるかが理解できないというJavaやiアプリに関連する専門用語のなかでも特に基本となる重要なものだけを集めてみた。当然、これらの用語がわからなければ、何も作れないので、はじめてJavaに挑戦する人は必ず読んでおいて欲しい。

なお、このほかにもJavaやiアプリに関するさまざまな専門用語がある。これからより高度なアプリケーションを作っていくつもりならば、ぜひサンのJava解説サイトJump01や、NTTドコモのiアプリ解説サイトJump02にも目を通しておくことをおすすめしたい。

Jump01 www.sun.co.jp/software/java/
Jump02 503i.nttdocomo.co.jp/normal/iappli/n_iappli.html

●J2ME（Java2 Micro Edition）

Javaはサン・マイクロシステムズが開発したプログラミング言語で、完全なオブジェクト指向性を備えているのが特徴だ。基本的にJavaで書かれたソフトは特定のOSやCPUに依存することなく、どんなプラットフォームでも動作する。セキュリティーやネットワーク機能が充実しているため、ネットワーク環境での利用に適した仕様になっている。Java 2は1999年にJavaを全面的に改良したもので、新たに多くの機能が追加された。Java 2にはいくつかのバージョンがあるが、J2MEはそのなかでも一般的なJava2 スタンダードエディションのサブセット版で、特に携帯電話やPDAのような小型デバイス向けに用意された。

●KVM（Kilobyte Virtual Machine）

Javaは、基本的にはどんなプラットフォームでも動作できるように、実行時に「Javaバーチャルマシン」（JVM）と呼ばれるソフトウェアによってOSを覆い、そのプラットフォーム上でプログラムを動かすための形式「ネイティブコード」に変換する。一般にJavaの動作が通常のプログラミング言語で開発されたソフトよりも遅いといわれるのは、この変換を行うためだ。

KVMはそのなかでもメモリーやCPU、電池消費に制約がある携帯デバイス向けに設計されたキロバイト単位のメモリーで動作する組み込み機器用のJavaバーチャルマシンのこと。ケー・バーチャルマシンと読む場合もあるがこのKは携帯電話のKではないので勘違いしないように。

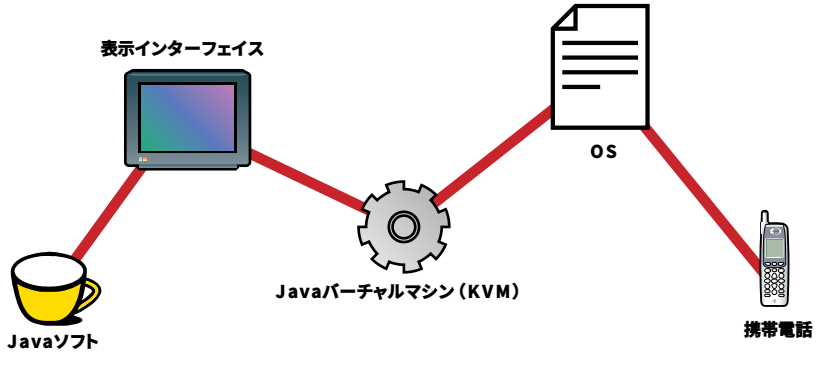
●JAR（Java ARchiver）

Javaの圧縮形式の1つ。ソースコードをコンパイル（変換）して、Javaプログラムを実行するのに必要なクラスやデータのファイルを配布のためにまとめるためのフォーマット。「ジャー」と読む。

●スクラッチパッド

iアプリで作成されたデータを携帯電話本体に保存しておくための記憶媒体のこと。最低5キロバイトは設定が可能。機種によってはさらに大きく設定できるものもある。一度記憶させた内容は電源を切っても消えない。

携帯電話におけるJavaの構造



携帯電話におけるJavaアプリケーションの構造はまず、携帯電話自体のOS上にどんなOSであってもあたかもJava専用機であるかのようにアプリケーションを動かすための一種のソフトであるJavaバーチャルマシンが乗り、そのうえに機種ごとの表示インターフェイスにのっとった形で各Javaソフトが動作する。そのためiアプリには機種依存の問題が出てくる。

コンパイル：プログラムを作る際には、まず人間がわかる形のプログラミング言語を使ってソフトの設計図とでもいうべき「ソースコード」を作るが、これをコンピュータが理解し、実行するための形式である「オブジェクトコード」に変換すること。

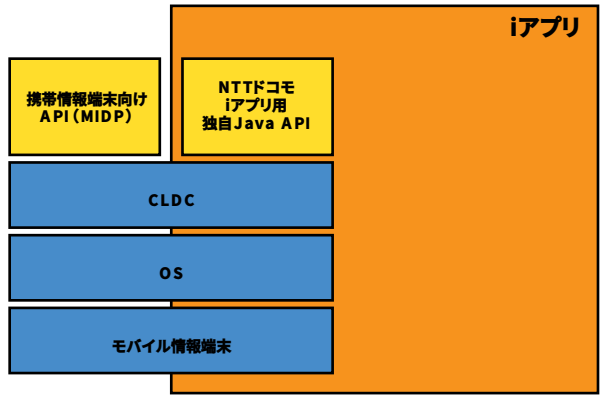
●CLDC（Connected, Limited Device Configuration）

これはJ2MEのクラス構成を定義するものだ。J2MEの構成で規定されるのは、スタンダードエディションおよびJavaバーチャルマシンの機能のどの部分をサブセット化するかということや、ネットワークやセキュリティー、サポートされるコアプラットフォームなどについてだ。これらはすべてある特定のJavaを組み込んだ製品をサポートするために規定される。CLDCはJ2MEのプロファイルの基礎となるもので、J2MEのプロファイルは、たとえば携帯電話など特定のデバイスのためのAPIや機能の付加的なセットを定義するものだ。構成およびプロファイルの詳細はサンのサイトで定義されている。

kJump01 java.sun.com/products/cldc/

●ソフトキー

503iから新しく登場した機能で、携帯電話画面の一番下（左右）に表示されたものを携帯電話本体のボタンで操作させることができるもの。iアプリ以外の機能でもある。



CLDCの上に乗るiアプリのAPIはサンが標準で持っているMIDP（Mobile Information Device Profile）ではなく、NTTドコモ独自のAPIを用いている。今後発売されるauやJフォンのJava端末はMIDPを使うといわれている。

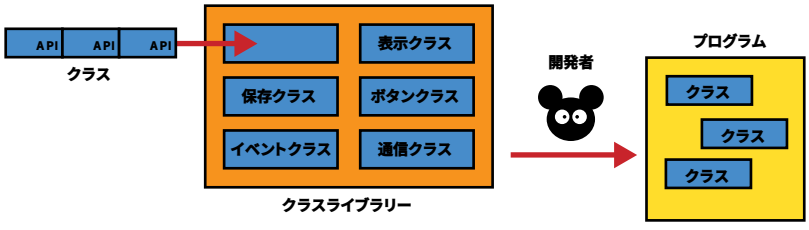
●クラスライブラリー

iアプリ（またはJava）を組むためにあらかじめ短い指示をプログラム化してあるもの。Javaを用いてある特定の機能を持ったプログラムを「クラス」という形で部品化し、それら関連する複数のファイルを1つにまとめたものをクラスライブラリーと呼ぶ。これは共通する属性やメソッド（手続き）を持ったオブジェクト群をまとめたものであり、プログラムの「部品」として利用できるため、一般的によく使われる汎用的な「部品」をクラスライブラリーに集めておけば、開発者の手間を軽減できる。また、これがあることによってプログラムを短く簡素に仕上げられるというメリットもある。

●API（Application Program Interface）

ソフトを開発する際に使えるコマンド（命令）や関数を集めたもののこと。同時に、それらのコマンドや関数を使う場合のプログラム的なルールも示す。多くのソフトで共用できるような一般的な機能をこうした形でまとめておくことで、各ソフト開発者はそうしたルールに従ってその機能を「呼び出す」だけで、その機能を利用できるため、プログラミングの手間を大幅に削減できる。iアプリでは、高低2つのレベルが用意されている。高レベルAPIはコンポーネントを組み合わせてアプリケーションのユーザーインターフェイスを作成するため、開発者の自由度は限定されるが、少ないコード行でコンパクトHTMLと同程度のアプリケーションを作成できる。ドコモによるとハードウェア特性を考慮する必要性が低く、異なるメーカーの携帯電話でも容易に動作させられるされるが、実際のところは機種依存の部分が多い。

一方、低レベルAPIは携帯電話の画面サイズやキーパッドなど、ハードウェアの特性を考慮しながらアプリケーションを作成できるため自由度は増すが、グラフィックスの描画や画面サイズと論理座標値といった細かい点を考慮する必要があり、より高度な知識が要求される。



一般的な目的ごとにあらかじめ作られたクラスを用いることによって開発者はプログラミングの手間を軽減できる。

コンパクトHTML（コンパクト・エッチ・ティー・エム・エル／Compact HyperText Markup Language）：携帯電話やPDAなどの携帯情報端末向けのコンテンツ記述言語。HTMLのサブセットとして定義され、HTMLと下位互換性がある。NTTドコモの「iモード」などで利用されている。

準備編

作る前に知っておきたい！

iアプリ作成 3つのポイント

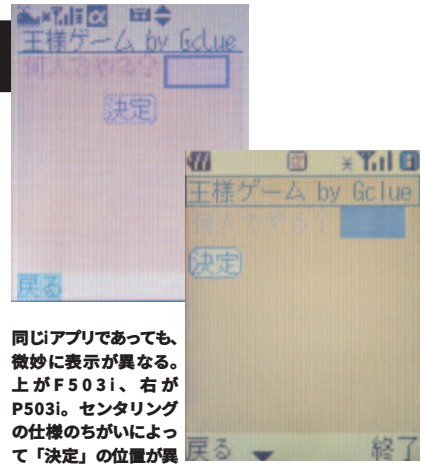
iアプリは機種依存が大きい

Javaで記述されたプログラムはプラットフォームや機種に依存することなく、どんな端末でも動くことが特徴である。iアプリもJavaで記述されており、ドコモの503iシリーズも携帯端末用のJavaバーチャルマシン「KVM」というソフトウェアでOSを覆うことで、携帯電話をJava専用マシンに見立てて、アプリケーションを動かす仕組みを取っている。

そうだとすると、P503iであろうとF503iであろうと同じようにアプリケーションが動くと思えるのだが、実際には両者の間では同じiアプリであっても微妙に見え方が違ったり動作スピードが違ったりする。これは、端末ごとにCPU速度や画面サイズが異なることが原因だ。どんな端末でも動くことと同じ速度や見え方で動くことは違う。たとえばJavaは文章1つを例にとってもHTMLのように自然に改行されないので、あらかじめ画面サイズやCPU速度を計算して、機種ごとに微妙に書き方を変えてプログラムしなくてはならない。そうしていないiアプリでは、たとえば同じブロック崩しゲームでもP503iだとちょうどいい速さだが、F503iでは速すぎて難しいといった現象が出てくる。

すべての端末向けの機能を実装したプログラムも組めないことはないが、10Kという制限内でそれを納めるのはかなり難しい。将来、実行ファイルの容量に10Kという制限がなくなるまでは端末の仕様にあわせていくつかのJavaプログラムを書く必要があるだろう。

なお、iアプリが「ドコモ仕様のJava」と言われるように、auやJフォンなど他のキャリアが採用する予定のJavaとの間の互換性が保証されていないという問題もある。



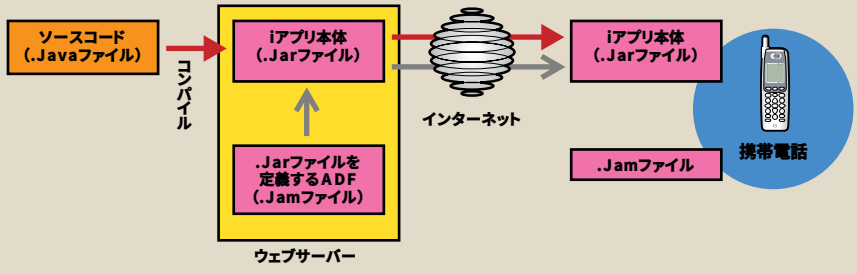
同じアプリであっても、微妙に表示が異なる。上がF503i、右がP503i。センタリングの仕様のちがいによって「決定」の位置が異なって見える。

iアプリにはJavaの常識は通用しない

iアプリはJavaベースではあるが、実は本来のJavaとは微妙に違う点があることにも注意したい。というのも、iアプリでは携帯電話という限られた資源のなかで動作させるためにNTTドコモが独自の仕様を設定しているからだ。

そのため、たとえばJavaの得意とするクラス分けや効率化を進めるとかえってメモリーを多くとってしまい、その結果iアプリの容量制限である10K以内に納まりきらないということがある。また、使い終わったオブジェクトはメモリー容量が少ないのでその都度、解放しなければならないといったことや、過去にJava向けに作ったライブラリーは、必ずしもiアプリのライブラリーに含まれているわけではないので、そのままでは流用できないといったことが挙げられる。これらは基本的には非力なデバイス向けのJava特有の現象で、容量制限の違いこそあれauやJフォンでもほぼ同じである。

また、そのほかにもJava経験者にとってあたりまえのことがiアプリでは通用しないこともある。たとえば、スクラッチパッドの容量をあらかじめJamファイルで定義しておく必要があるとか、普通なら当然Jarファイルで書かれているべきことが、実はJamファイルで定義されているといったことだ。これらはJava経験者のほうがかえって間違いやすいので、最初にしっかりとiアプリ独特の作法を認識しておく必要があるだろう。



iアプリは、ソースコード（.Javaファイル）をコンパイル、圧縮したJarファイルとJarを定義するJamファイルの2つで構成されている。iアプリを携帯電話にダウンロードするには、まず最初に容量の小さいJamファイルをダウンロードして、Jarの保存に必要な記憶容量や互換性をチェックする。JamはJarの制御的な役割も担っているため、一般的なJavaの常識とは異なる記述を要求する場合がある。

ライブラリー：ある特定の動作をするプログラムを部品化したものをいくつも集めて1つにまとめることで、別のプログラムからでも、それらの機能を容易に利用できるようにしたもの。ライブラリー単独では機能せず、他のプログラムの一部として組み込まれることではじめて動作する。

iアプリには2つの種類がある

ひとくちにiアプリといっても、実は次の2つの種類が存在する。そのため、開発の際には最初に「スタンドアローン型」と「ネットワーク型」（クライアントサーバー型）どちらのアプリケーションを作るかということを考慮しなければならない。

スタンドアローン型というのは、携帯電話単体で動くアプリケーションのことだ。そのためiアプリのデータ容量制限である10Kバイトだけで完結させなければならない。しかし、この場合は最初にiアプリをダウンロードさえしてしまえば、あとはたとえ地下や山奥など、電波の届かない場所であってもアプリケーションは完全に利用できる。

一方、ネットワーク接続型はアプリケーションを起動するたびに通信を行い、サーバー上のデータを利用しながら実行するので、実質的に10Kバイトのファイル容量制限の壁がなくなる。しかし、iアプリを動かすたびに通信するので通信できる場所でしか使えないし、利用するたびにパケット料金がかかる。

どちらのタイプを作るかによって最初から設計が変わってくるため、まずタイプを考えなくてはならないというわけだ。

●スタンドアローンとネットワーク型の比較

	スタンドアローン型	ネットワーク型
利用可能範囲	特に制限無し	携帯電話の通信圏内のみ
パケット通信料	一度ダウンロードしてしまえば以後はかからない	通信機能を使うたびにパケット通信料がかかる
動作速度	通信速度の影響は受けないので、動作速度は常に一定	通信速度の影響を受けるので、混雑によって反応が遅くなることもある
アプリケーションの容量制限	最大10Kまでだが機種によってはそれ以上可能な場合もある	iアプリ本体では10Kまでだが、サーバーを使うことで見かけ上大きくできる
データ保存場所	スクラッチパッドの最大5Kだが機種によりそれ以上使えるものもある	スクラッチパッドに加えてネットワークの記憶領域も使える
向いてる分野	ゲームなど、特にデータ自体が変化しないもの	株価情報などデータが変動するものや、地図など巨大なデータベースを利用するもの

●iアプリでできることできないこと

iアプリでは、パソコンのようにアプリケーションをインストールできるが、パソコンのように万能なわけではない。たとえば、iモードでは、「Phone to」機能を使って、iモードブラウザーから直接電話をかけられるが、iアプリではそうしたiモード標準の機能でもできないものがある。ここでは、そうしたiアプリの「できること、できないこと」をざっと挙げてみた。

●Iアプリでできること、できないこと

できる	できない
・文章の保存	・iアプリから電話帳にアクセスして電話番号をかける
・画像の表示やグラフの描画	・複数のiアプリを起動する
・iメロディの再生	・自分のダウンロードしたiアプリを人にあげる
・サーバーのデータを表示	・iアプリからiモードブラウザーを起動させる

※機種によってできないこともできる可能性がある

Jamファイルの重要性

iアプリでは、Jamファイルが非常に重要な役割を持つことを認識する必要がある。iアプリにおけるユーザー記憶領域であるスクラッチパッドの容量を設定したり、バージョンの管理をするほか、アプリケーション名などの設定をJarではなくあらかじめJamファイルに書き込む方式をとっている。iアプリでは本体であるJarファイルですべてのことを完結させるのではなく、Jamファイルの補助があってはじめて動作するのだ。

■JarではなくJamファイルで設定すること

- アプリケーション名
- バージョン
- iアプリのあるURL（同じ階層にあればJarファイル名）
- iアプリデータサイズ
- Javaバーチャルマシンのバージョン
- アプリケーションのクラス名
- 更新日
- スクラッチパッドの領域
- ネットワークの設定（実はhttpしか対応していない）

Phone to機能：iモードに装備されている機能の1つで、iモード対応のウェブページ上に記載された電話番号をクリックするだけで電話がかけられるというもの。

基礎編

通信圏外でも携帯電話単体でiアプリが動く

スタンドアローン型を作る！

最初に構造を理解しよう

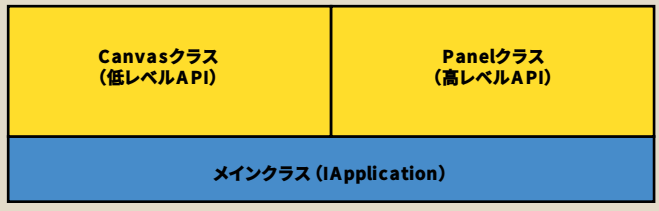
それでは、実際にiアプリを作っていこう。まず最初は基礎編としてスタンドアローン型を作ってみる。実際のJavaの記述の仕方といった言語に関する部分はたとえば小社刊の「Java入門」（1,631円、税抜）のようなJavaの入門書を参考にしてもらうとして、ここでは「iアプリを作る際のポイント」に焦点を当てて解説していく。

まず、最初に理解しておきたいのはiアプリは「IApplication」というNTTドコモが設定したiアプリ独自のメインクラスを使ってJavaを記述する必要があるということだ。基本的には、このメインクラスの上に高レベルAPIである「Panel」クラス、または低レベルAPIである「Canvas」クラスを用いて、実際に携帯電話での表示のさせ方や、各ボタンを押したときの動作のさせ方を記述する（右上図）。Panaelはある動作や結果に対しての雛形がすでにあるのに対し、Canvasはその雛形自体を自作する必要がある。当然、難易度はCanvasのほうが高い。なお、PanelとCanvasはそれぞれを交互に表示させることはできるが、同時に表示させることはできない。

実際にプログラムを書く際にはまず最初に核となるメインクラスを書いたあと、表示に関するPanelまたはCanvasクラスを書き、次にボタンを押したときにどのような動作するかというような部分を書き、最後にスクラッチパッドに保存を要するようなアクションについての記述をしていくと理解しやすいだろう（下図）。

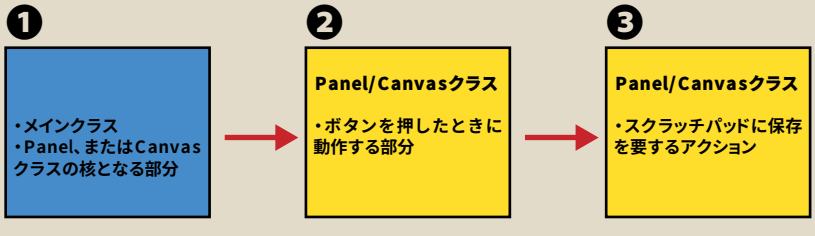
この記事では、スタンドアローン型のiアプリの構造を理解するためのサンプルとしてPanelクラスだけを用いたごく簡単な備忘録（メモ帳）を作ってみた。続く214ページからは、このメモ帳を構成するすべての<u>ソースコード</u>を掲載しているので、それに沿ってプログラム上の注意点を逐一解説していく。まずは、これをもとに全体の構造を理解したうえで、より高度な作り込みができるCanvasを使ったiアプリに挑戦するといいだろう。

iアプリの基本構造

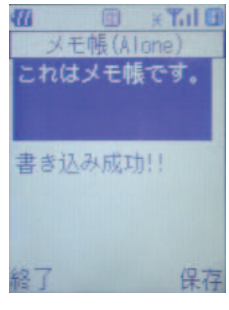


「Panel」は高レベルAPI（コンポーネントと呼ばれる部品が用意されていて比較的簡単に扱える）のためのコンテナであり、「Canvas」は低レベルAPI（扱いが難しいが複雑なことができる）のためのコンテナである。高レベルAPIのコンポーネントはすべてPanelに、低レベルAPIはCanvasを使用することで表示される。画面に出せるのはPanelかCanvasのどちらか1つだけで、高レベルAPIと低レベルAPIが1画面に混在したiアプリは作成できない。ただし、それぞれを複数用意して切り替えることはできる。

プログラムを書く手順



今回作成した、サンプルのメモ帳。ごくシンプルな機能しかできないが、iアプリの構造を理解するには最適だろう。なお、このメモ帳のプログラムデータは本誌CD-ROMに収録されている（374ページ参照）ので、自分の携帯電話にインストールして実際に使うこともできる。



これが今回サンプルとして作成したメモ帳アプリケーションの画面。

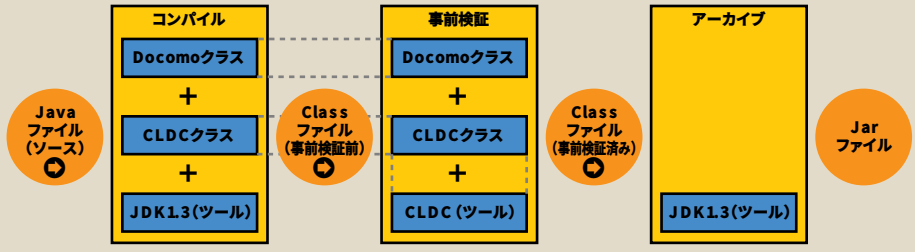
ソースコード：ソフトの開発者がプログラミングする際にプログラミング言語を使ってソフトウェアの設計図とでもいうもの。そのままではコンピュータ上で実行できないため、プログラムをコンピュータ上で実行するためには、別途「コンパイラ」などのソフトウェアを使ってコンピュータが理解できる「オブジェクトコード」変換しなければならない。

ソースができたらコンパイルしよう

214ページのようなソースコードが完成したら、それを実際に携帯電話で動かすためにコンパイル（変換）をしよう。それには、まずJavaソースコードをコンパイルしてJavaクラス（.classファイル）にしなければならない。次にJavaクラスの事前検証を行う。そして、画像やサウンドとともに事前検証されたクラスをJavaの圧縮形式である.jarファイルに変換してアプリケーションの形にする。

本来、この作業を行うためには「JDK1.3」と「CLDC1.0」、「ドコモのクラスライブラリー」の3つが必要になる（207ページ参照）。しかし、現時点（2月15日現在）では、「ドコモのクラスライブラリー」がまだ一般に公開されていないため、正規の方法ではコンパイルできない。

iアプリの標準的なコンパイル手順



クラスライブラリーがない場合のコンパイル法

そこで、i-JADE Liteを使った方法を簡単に紹介する。i-JADE LiteはZENTEK（株式会社ゼンテック・テクノロジー・ジャパン）が提供しているiアプリの統合的な開発環境の1つで、ユーザー登録をすると無償で入手できる。

なお、くわしい手順はギガフロップス株式会社が運営しているサイト“GIGA APPLI”で提供されているiアプリ開発支援環境“GADeK”を参考にして欲しい。

Jump www.zentek.com/ja/
Jump www.g-appli.net

クラスライブラリーがない場合のコンパイルの手順

❶JDK1.3、J2ME-CLDC1.0、i-JADE Liteをインストールする。

▼

❷C:¥i-jade¥i-jade-f.jar をc:¥classes に展開する。

▼

❸C:¥j2me_cldc¥bin¥api¥classes¥java をc:¥classes にコピーする。

▼

❹ソースコードのあるフォルダーにclassとpreverifiedというフォルダーを作成。

▼

❺コンパイルを実行する。
```
> javac -g:none -bootclasspath c:¥i-jade¥i-jade-f.jar;c:¥j2me_cldc¥bin¥api¥classes -d class *.java
```

▼

❻事前検証を実行する。
```
> preverify -classpath c:¥classes; -d preverified class
```

▼

❼圧縮を実行する。
```
>jar cvfM NetMemo.jar -C preverified .
```

●NTTドコモに聞くiアプリの今後

iアプリに10Kの容量制限があるのには2つの理由があります。1つは端末のメモリーやCPUの問題です。2つ目は9600bpsという通信速度の問題です。iアプリのダウンロードに何時間もかかるのでは困りますよね。しかし、これらは端末の性能や通信速度が上がり、より高度な圧縮技術が登場すれば解消され、いずれは10Kの制限もなくなるでしょう。ちなみに初期のファミコンも10Kくらいでしたから、これでもけっこうおもしろいことはできると思います。なお、iアプリにはたとえばアプリから直接電話がかけられないとか、iアプリを友達にあげられないとか、ダウンロードしたサイトにしかデータを保存できないといった、いくつかの「できないこと」がありますが、これは悪意のプログラムが勝手に電話をかけてユーザーの損害となるのを防止したり、コンテンツプロバイダーの著作権を保護したりするために設けたものです。権利や安全の保護と利便性の関係は難しい側面があります。また、画面サイズやCPU速度などの違いによって端末ごとにiアプリの動作が一定でないことについては、弊社はあくまでも枠決めをするだけなので、その部分は端末メーカーに任せています。そうでないと、端末の違いは色や形だけになってしまいますよね。そのなかでユーザーに支持される仕様がいずれはデファクトスタンダードになっていくのでしょう。

株式会社NTTドコモ
モバイルマルチメディア事業本部
ゲートウェイビジネス部
コンテンツ開拓担当課長
山口善輝

デファクトスタンダード：標準として定められた訳ではないが、市場の趨勢から自動的にはは標準としての位置付けを与えられた仕様のこと。事実上の標準として市場のトレンドを左右するだけの影響力をを持つことが多い。

ソースを書く上でのポイントはココ！

●核となるパネル記述部分
●ボタン操作に関する部分
●スクラッチパッドに関する記述

```
/**
 * タイトル:   メモ帳(Alone)
 * 会社名:     株式会社　エル・カミノ・リアル
 * @version   1.0
 */

import java.io.*;  ←❶
import javax.microedition.io.*;  ←❶
import com.nttdocomo.ui.*;  ←❷

// アプリケーションのメインクラス
public class MemoAlone extends IApplication  ←❸
{
  // アプリケーションが起動したら呼ばれる
  public void start()  ←❸
  {
    // カレントフレームの設定をする
    MainPanel  p = new MainPanel( this );  ←❹
    Display.setCurrent( p );  ←❺

    // バックライトをオンにする
    PhoneSystem.setAttribute( PhoneSystem.DEV_BACKLIGHT, PhoneSystem.ATTR_
BACKLIGHT_ON );  ←❻
  }
}

// メモ帳画面クラス（パネルクラス）
class MainPanel extends Panel implements SoftKeyListener  ←❼
{
  // テキストボックスのサイズ
  public  static  final int     ROW = 18;
  public  static  final int     COL = 3;  ←❽

  private TextBox               textBox;  // テキストボックス
  private Label                 lbl;      // ステータスを表示するラベル  ←❾

  // 初期化を行う
  public MainPanel( IApplication iapp )
  {

    // タイトルを付ける
    setTitle( "メモ帳(Alone)" );  ←❿

    // ソフトキーにラベルをつける
    setSoftLabel( Frame.SOFT_KEY_1, "終了" );
    setSoftLabel( Frame.SOFT_KEY_2, "保存" );  ←⓫

    // ソフトキーリスナを登録する
    setSoftKeyListener( this );  ←⓬

    // パネルにコンポーネントを追加する
    textBox = new TextBox( "", ROW, COL, TextBox.DISPLAY_ANY );  ←⓭
    lbl = new Label( "ステータス" );

    add( textBox );
    add( lbl );  ←⓭

    // ScratchPadからデータを読み込む
    readScratch();  ←⓮
  }
```

メインクラス

パネルクラス

Ⓐ 使用するクラスライブラリーのインポート（使えるようにすること）をしている。

❶ CLDCのI/O（入出力）関係のライブラリー。

❷ NTTドコモ独自のユーザーインターフェイス関係のライブラリー。下記の「IApplication」クラスがこのパッケージの中にあるので、iアプリを作成する場合には絶対に必要。

Ⓑ Javaではまずクラスを定義をし、それぞれのクラスを組み合わせてアプリケーションを作るが、iアプリのメインクラスは必ず「IApplication」を継承する必要がある。なお、メインクラスはJamファイルで指定する。通常のJavaアプリでは「public static void main」(String[]args)の中で自分自身をインスタンス化（new）するが、iアプリではJamがアプリケーションを起動した時点で、メインクラスがインスタンス化されているので、その必要はない。

❸ iアプリを起動させるとまずメインクラスの「start」メソッドが呼ばれる。iアプリはアプリケーションだが、イメージ的にはアプレットに近い。

❹ iアプリで何かを表示するためにはコンテナ（器）を作らなくてはならない。コンテナには、「Panel」と「Canvas」の2種類がある。高レベルAPIのコンポーネントはすべてPanelを、低レベルAPIはCanvasを使用することで表示される。

❺ ここで表示する「Panel」を指定している。iアプリでは、一度に1つのPanelまたは、Canvasしか表示できない。

❻ この命令でバックライトをオンにする。F503iはアプリ起動後すぐにバックライトがオフになるので、今回はバックライトをオンにするようにした。503iになって消費電力が増えたため。

Ⓓ ここではPanelを親クラスにして「MainPanel」という名前のクラスを定義している。これが、このプログラムのコンテナになっている。

❼ 緑色の部分がPanelでソフトキーを使う際の基本形。Panelのイベント処理にはリスナーが採用されている。「SoftKeyListener」はソフトキーを使う時にイベントを受けるクラスが実装すべきインターフェイス。これ以外にもコンポーネントのイベントを受ける「ComponentListener」やキー入力のイベントを受ける「KeyListener」などがある。ただし、Canvasのイベント処理は扱いが異なり、Canvasクラスの持つ「processEvent」メソッドを使用する。

❽ 下で作成するテキストボックスの縦横の大きさ（文字数）を設定。ここではあとで変更しやすいように変数にしている。

❾ コンポーネント自体を表す変数を定義している。

Ⓒ この部分がMainPanelのコンストラクター。コンストラクターはクラスが生成された時に呼び出されるメソッド（手続き）。MainPanelを使うときにはじめに実行したい命令などを書いておく。

❿ Panelにはタイトルをつけられる。タイトルは画面の一番上に表示される。この命令で「メモ帳(Alone)」というタイトルをつけている。

⓫ ソフトキーにラベルをつける。今回はわかりやすく "終了" "保存" とした。Frame.SOFT_KEY_1とFrame.SOFT_KEY_2は、それぞれ左側のソフトキーと右側のソフトキーを表す定数。

⓬ ソフトキーリスナーの登録。ソフトキーを使う時に必要となる。この命令でソフトキーが押されたときにこのクラスの中にある「softKeyPressed」や「softKeyReleased」が呼ばれるようになる。

⓭ ここでコンポーネントを生成してPanelに追加している。テキストボックスやラベルなどのコンポーネントをPanelに載せるときは、newを使ってコンポーネントを生成したあとに、addでPanelに追加する。

⓮ スクラッチパッドに保存しておいた内容を起動した時に表示するための記述。readScratch()については次のページで説明する。

アプレット：WWWブラウザのコンテンツに埋め込まれ、実行時にはクライアントにダウンロードして動作する小さなアプリケーションプログラムのこと。「Java言語」で記述され、HTML言語に張り付けられる程度の小さなプログラムを指すものを「Javaアプレット」と呼ぶ。

パネルクラス

```
// ソフトキーが押されたら時に呼ばれる                                    E
public void softKeyPressed( int softKey )
{
  // ソフトキー1(左側のキー)が押されたら、アプリケーションを終了させる
  if( Frame.SOFT_KEY_1 == softKey )
  {
    // アプリケーションを終了する
    IApplication.getCurrentApp().terminate();  ←⑮
  }
  else if( Frame.SOFT_KEY_2 == softKey )
  {
    // ScratchPadへデータが正しく書き込まれたかどうか確かめるために
    // データを書き込んだ後、データを読み込む
    writeScratch();
    readScratch();  ←⑯
  }
}

// ソフトキーが離された時に呼ばれる
public void softKeyReleased( int softKey ){}  ←⑰

// ScratchPadへの書き込み                                                F
public void writeScratch()
{
  try
  {
    // テキストをScratchPadへ書き込む
⑱→ DataOutputStream out = Connector.openDataOutputStream( "scratchpad:///0" );
    out.writeUTF( textBox.getText() );  ←⑲
    out.close();  ←⑳

    lbl.setText( "書き込み成功!!" );  ←㉑
  }
  catch( IOException e )
  {
    lbl.setText( "ScratchPadへの書き込みエラー" );  ←㉒
  }
}

// ScratchPadの読み込み
public void readScratch()
{
  String str = "";  ←㉓

  try
  {
    // ScratchPadのテキストを読み込む
㉔→ DataInputStream in = Connector.openDataInputStream( "scratchpad:///0" );
    str = in.readUTF();  ←㉕
    in.close();  ←㉖
    // ScratchPadから読み込んだテキストをテキストボックスへ表示する
    textBox.setText( str );  ←㉗
  }
  catch( IOException e )
  {
    lbl.setText( "ScratchPadの読み込みエラー" );
  }

}                                                                       G
```

Ⓔ ソフトキーが押された時に呼ばれるメソッド。ここにソフトキーが押された時の動作を記述する。softKey変数に押されたソフトキーの種類を表す値が入っている。

⑮ アプリケーションを終了させる命令。終了ボタン（ソフトキー1）が押された時に実行される。

⑯ writeScratchでスクラッチパッドに保存している。readScratchは本来必要ないが、確認のため保存した内容をもう一度読み出して表示させている。両方とも以下で定義しているので、そこで説明する。

⑰ ソフトキーが離された時に呼ばれるメソッド。ソフトキーのイベントには「押された時」と「離された時」があるが、キーを押しっぱなしにするなど特別な操作を必要としていなければ、どちらか一方に実行させたい動作だけ書けばよい。ただし、中身がカラでも定義はしなければならない。

Ⓕ 今回作成したスクラッチパッドへ保存するためのメソッド。初級者には少々手強い部分だが、テキストボックスにある文章をそのままスクラッチパッドに保存したい場合は、この形をほぼそのまま流用できる。
このサンプルでは文字列の扱いを楽にするために、InputStreamを継承するDataInputStreamを使ってUTF形式でスクラッチパッドに書き込んだが、文字コードを意識さえすれば、どんな形式でもかまわない。

⑱ スクラッチパッドを出力のために開く。

⑲ スクラッチパッドにテキストボックスの内容を保存。

⑳ スクラッチパッドを閉じる。

㉑ 確認のためラベルに「書き込み成功!!」と表示。

㉒ 失敗した場合にラベルへ「ScratchPadへの書き込みエラー」と表示。

Ⓖ 今回作成したスクラッチパッドから読み出すためのメソッド。基本的な形は保存の時と同じ。

㉓ 読み出しの作業用に文字列変数を作る。

㉔ 読み出しのためスクラッチパッドを開く。

㉕ スクラッチパッドの内容を読み出す。

㉖ スクラッチパッドを閉じる。

㉗ 読み出した内容(str)をテキストボックスにセットする。

iアプリのLayoutManagerは、現在のところ、機種依存の部分が大きい。標準のLayoutManager以外は、メーカー仕様が発表されるまで分からないが、現在使用できる標準のLayoutManagerも、その動作は機種依存である。また、初期に発売されたF503iとP503iでもコンポーネントの表示や配置のされ方からすでに違いがある。iアプリの開発者、特に多くの機種にも対応させたい場合は注意して欲しい。

UTF（ユー・ティー・エフ／UCS Transformation Format）：ユニコードであるUCS-2の定義に基づいて記述された文字列をコンピュータが扱えるように数値化（バイト列に変換）する方式。

上級編

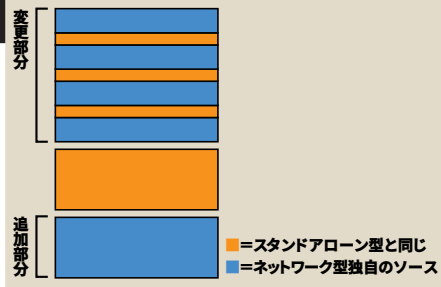
サーバーを使ってより高度なiアプリを実現する

ネットワーク型を作る

通信部分以外はスタンドアローン型と同一

ネットワーク型に挑戦する人は、最初に本誌付録CD-ROM🄰（374ページ参照）収録の「Netmemo.java」というソースを見てほしい。すると、行数は多いが内容的には先ほど作ったスタンドアローン型によく似ていることに気付くはずだ。実際、これはサーバーとの通信に関する記述以外は同じだ。そこで、ここではスタンドアローン型と異なる部分のみを解説する。スタンドアローン型のソースに一部新たなソースを加えている部分は「変更部分」、一方、まったく新しいソースを追加している部分は「追加部分」として、以下にそれぞれのポイントを挙げてみた。なお、コンパイルの仕方はスタンドアローン型とまったく同じだ。

スタンドアローン型との相違点



変更部分のソースに関するポイント

①ライブラリーのインポート

```
import java.io.*;
import javax.microedition.io.*;
import com.nttdocomo.io.*;
import com.nttdocomo.ui.*;
```

ネットワーク型ではサーバーと通信を行うため、http通信用のライブラリーのインポートを指示する「import com.nttdocomo.io.*;」を追加する必要がある。

②メモ帳画面クラス

```
// メモ帳画面クラス（パネルクラス）
class   MainPanel   extends   Panel   implements   SoftKeyListener,
ComponentListener
{
  // CGI名
  public  static  final String  UPLOAD_CGI    = "upload.cgi";
  public  static  final String  DOWNLOAD_CGI  = "download.cgi";

  // レスポンス定数
  public  static  final int     NG  = '0';
  public  static  final int     OK  = '1';
```

ネットワーク型ではアップロード／ダウンロードボタンを使うため、「SoftKeyListener」のほかに「ComponentListener」も実装する。また、サーバー側でCGIスクリプトを動かすのでサーバー側に置くCGIスクリプトの名称を”” 内に記述する必要がある。同時に、今回作成したCGIスクリプトで決めた戻り値をレスポンス定数として書いておく。これはCGIがうまく動作したかどうかを確認するためのものだ。

③アップロード／ダウンロードボタンの変数

```
  // テキストボックスに表示されているテキストをアップロードするボタン
  private Button              up;

  // サーバーからテキストをダウンロードし、テキストボックスへ表示するボタン
  private Button              down;
```

アップロードボタンとダウンロードボタンのための変数。これを「テキストボックス」と「ステータスを表示するラベル」の間に挿入する。

④コンポーネントリスナーの登録

```
    // コンポーネントリスナーを登録する
    setComponentListener( this );
```

ComponentListenerをパネルに登録。これで、「ボタンを押す」などの「コンポーネントイベント」が発生した時にこのクラスの中にある「componentAction」が呼ばれる。

⑤コンポーネントの追加

```
    // パネルにコンポーネントを追加する
    textBox = new TextBox( "", ROW, COL, TextBox.DISPLAY_ANY );
    up = new Button( "アップロード" );
    down = new Button( "ダウンロード" );
    lbl = new Label( "ステータス" );

    add( textBox );
    add( up );
    add( down );
    add( lbl );
```

アップロードボタンとダウンロードボタンを生成し、パネルに追加する。

⑥コンポーネントイベントの処理

```
// コンポーネントで発生したイベントの処理
  public void componentAction( Component c, int type, int param )
  {
    if( c.equals( up ) )
    {
      // サーバーへテキストデータをアップロードする
      txtUpload();
    }
    else if( c.equals( down ) )
    {
      // サーバーからテキストをダウンロードする
      txtDownload();
    }
  }
```

コンポーネントイベントが発生した時に呼ばれるメソッド。引数のcがイベントの発生したコンポーネントを表す。「txtUpload();」はイベントの発生したコンポーネントがアップロードボタン(up)だった時に、アップロード処理を実行する。「txtDownload();」はイベントの発生したコンポーネントがダウンロードボタン(down)だった時に、ダウンロード処理を実行する。

スクリプト：機械語への変換（コンパイル）を自動化することで、簡単に実行できるようにした形式の簡易プログラムのこと。スクリプトを記述するための言語にはたとえばPerlやVBスクリプト、Javaスクリプトなどがある。

新たな追加部分のソースに関するポイント

アップロードのためのメソッド

```
// サーバーへテキストデータをアップロードする
  public void txtUpload()
  {
    int res;  // レスポンス値 ←❶

    try
    {
      lbl.setText( "アップロード中" ); ←❷

      // 接続設定を行う                                          A
      HttpConnection conn = ( HttpConnection )Connector.open(
                            IApplication.getCurrentApp().getSourceURL()
                            + UPLOAD_CGI,
                            Connector.READ_WRITE, true );  ←❸

      // 要求メソッドとコンテンツタイプの設定を行う
      conn.setRequestMethod( HttpConnection.POST ); ←❹
      conn.setRequestProperty( "Content-Type", "text/plain" ); ←❺

      // 出力ストリームを取り出す
      OutputStream out = conn.openOutputStream(); ←❻

      // データを書き込み（まだ接続が確立されていないので実際は、
      // バッファに入れられる）
      out.write( textBox.getText().getBytes() ); ←❼

      // 出力ストリームをクローズする                            B
      out.close(); ←❽

      // 実際にリモート資源に接続する
      conn.connect(); ←❾

      // レスポンスを受け取る
      InputStream   in = conn.openInputStream(); ←❿
      res = in.read(); ←⓫                                      C
      in.close(); ←⓬

      // 接続をクローズする
      conn.close(); ←⓭

      // 結果を表示する
      if( res != OK )
      {
        lbl.setText( "アップロードに失敗" );
        return;
      }

      lbl.setText( "アップロード成功!!" );                      D
    }
    catch(Exception e)
    {
      lbl.setText( "アップロードに失敗" );
    }

  }
```

❶ 処理が成功したかどうかを確認するCGIからの戻り値を入れるための変数。

❷ 作業状況をステータス表示用に作ったラベルに表示。

Ⓐ この間がアップロード用のhttp通信の一連の流れになっている。

❸ HTTP通信用のコネクションを開くためにはじめに行うべきこと。この時点ではまだ通信は始まっていない。「IApplication.getCurrentApp().getSourceURL()」でこのアプリケーションをダウンロードした元のURLが取得できる。

❹ HTTP通信用のメソッドの指定。ここではPOSTを使用している。HTMLの送信フォームなどで使われているPOSTと同じ意味。

❺ 通信内容のコンテキストタイプを指定。今回はプレーンテキスト形式を使用している。

Ⓑ ここでアップロードする内容を準備する。具体的にはHTTP通信用のコネクションの出力ストリームにセット。まだ、接続が確立されていないので、実際はバッファに入れられる。

❻ 出力ストリームを開く。

❼ テキストボックスの内容を出力ストリームに書き出す。ここでの「getBytes()」メソッドで、UNICODEから503i標準の文字コード(SJIS)のバイト列に変換されている。

❽ 出力ストリームを閉じる。

❾ ここで、実際の通信が実行される。ブラウザーでいうところのURLを指定した瞬間から必要なデータのダウンロードが完了するまでの間と同じである。

Ⓒ ここでCGIから送られてきたデータを取り出す。具体的にはHTTP通信用のコネクションの入力ストリームから取り出す。今回作成したアップロード用のCGIは、成功したかどうかを確認する戻り値（1バイト）のみを返すようにしている（前に設定したOKかNGが入っている）。

ダウンロードのためのメソッド

```
// サーバーからテキストをダウンロードする
public void txtDownload()
{
  int     res;  // レスポンス値

  try
  {
    lbl.setText( "ダウンロード中" );

    // 接続設定を行う                                          E
    HttpConnection conn = ( HttpConnection )Connector.open(
                          IApplication.getCurrentApp().getSourceURL()
                          + DOWNLOAD_CGI, Connector.READ, true );  ←⓮

    // 要求メソッドの設定する
    conn.setRequestMethod( HttpConnection.GET ); ←⓯

    // 実際にリモート資源に接続する
    conn.connect();

    // レスポンスを受け取る
    InputStream   in = conn.openInputStream();

    // ダウンロードする文字列の長さと同じ領域を確保する
    byte [] by = new byte[ in.available() - 1 ]; ←⓰

    // 結果を表示する
    res = in.read();
    if( res != OK )
    {
      lbl.setText( "ダウンロードに失敗" );
      return;
    }

    // テキストを読み込む
    in.read( by ); ←⓱

    // サーバーからの送られてくる文字コードによっては、
    // 文字コードの変更を行わなければならない
    String str = new String( by ); ←⓲

    // テキストボックスに表示する
    textBox.setText( str ); ←⓳

    // 入力ストリームをクローズする                            F
    in.close();

    // 接続をクローズする
    conn.close();

    lbl.setText( "ダウンロード成功!!" );
  }
  catch( Exception e )
  {
    lbl.setText( "ダウンロードに失敗" );
  }
}
```

❿ 入力ストリームを開く。

⓫ 入力ストリームから、用意しておいた変数(res)に1バイトだけ読み込む。

⓬ 入力ストリームを閉じる。

⓭ はじめに開いたコネクションを閉じる。これは忘れやすいので注意

Ⓓ 戻り値を判定して、結果をステータス表示用に作ったラベルに表示。ここでの失敗はCGIの内部で何らかのエラー（データ保存用のファイルが開けないなど）の発生を示す。

Ⓔ この間がダウンロード用のHTTP通信の一連の流れになっている。

⓮ アップロードとの違いは次の2つ。(1)ダウンロード用に用意したCGI(DOWNLOAD_CGI)になっている。(2)データを参照するだけなので2番目の引数が「Connector.READ」（読み込みだけ）になっている。

⓯ アップロードとの違いは、データを参照するだけなので、メソッドを「GET」にしたこと。

Ⓕ ここでCGIから送られてきたデータを取り出す。今回作成したダウンロード用のCGIは、先頭の1バイトがアップロードと同じ戻り値になっていて、2バイト目以降がダウンロードしたテキストの内容になっている。

⓰ ダウンロードしたテキストを入れるための変数を用意する。「in.available()」でダウンロードした全体の大きさ（バイト数）が取得できるので、そこから戻り値分の1バイトを引く。

⓱ 続きからbyの長さだけデータを読み込む。結果的に、残り全部（テキスト部分）を読み込むことになる。

⓲ ここでダウンロードしたテキスト（SJISのバイト列：by)からUNICODE文字列(str)への変換が行われている。見逃しやすいが、文字化けした場合などは要注意。サーバー側でSJISにすることによって問題はなくなる。

⓳ ダウンロードしたテキストをテキストボックスに表示する。

CGI（シー・ジー・アイ／Common Gateway Interface）：WWWサーバーで、外部プログラムを起動して、その結果をWWWクライアント（ブラウザー）に返す仕組み。ユーザーのリクエストによってウェブページの表示内容を変えたり、ユーザーの入力したフォームのデータをサーバー側へ読みこんだりできる。

上級編

自慢の傑作が完成したら

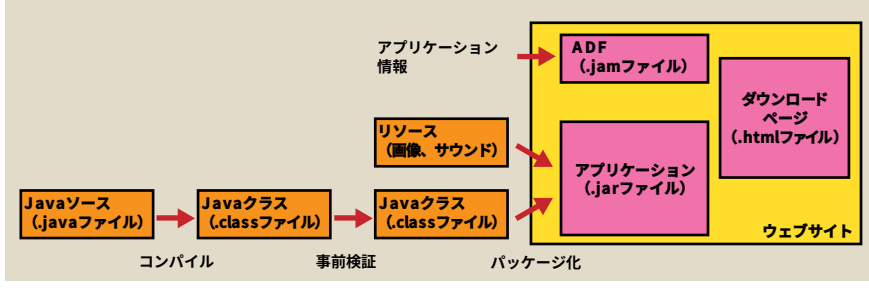
iアプリをウェブで公開しよう！

公開にはHTMLのほか、.Jar、.Jamの3つが必要

これまでのページで、iアプリの本体である.Jarファイルができあがったわけだが、iアプリができあがったら、ウェブで多くの人に公開しよう。ここでは、iアプリを公開するための手順を解説する。iアプリは本体である「.jarファイル」のほかに、端末にアプリケーションの解説をするADF（Application Descriptor File）である「.Jamファイル」で構成されているため、ウェブで公開するには別途この.Jamファイルを作る必要がある。このファイルの目的は、アプリケーションを携帯電話に正しくダウンロードできるかどうかをiモードのJAM（Java Application Manager）が判定できるようにすることだ。ADFの容量は100から300バイト程度と小さいため、iアプリ全体をダウンロードする前にADFだけをダウンロードして、保存に必要な記憶容量や互換性をチェックすれば、通信コストの面から有利になる。サーバーからADFをダウンロードさせるには、HTMLの<OBJECT>タグでADFを参照させるようにすればよい。

あとは、これらのデータを自分のホームページエリアに上げておけば、iモードのウェブ画面からiアプリをダウンロードできるようになるわけだ。

●iアプリ公開の手順



ソースコードをコンパイルしてJavaクラスを作ったら、まず事前検証をする。問題がなければ画像や音楽などのファイルも集めてiアプリの実行ファイル（.Jarファイル）に圧縮、統合する。あとはダウンロードするためのHTMLファイルとともに、このJarファイルと、それを定義するJamファイルをウェブサーバーにアップロードすればOKだ。

●HTMLファイルの例

```
<html>
<head>
<title>NetMemo</title>
</head>
<body>

<OBJECT declare id="NetMemo" data="NetMemo.jam" type="application/x-jam"></OBJECT>
<A ijam="#NetMemo" href="NetMemo.html">NetMemo</A>

</body>
</html>
```

iアプリは、JamファイルとJar ファイル（iアプリ本体）から構成されている。ウェブサーバーに置かれたJamファイルをダウンロードするには、コンパクトHTML のOBJECT タグでJamファイルを参照する。

●Jamファイルの作り方

基本的には右の例を参考にすればよいのだが、ここで注意したいのは、1つのキーエントリーを1行で記述しなければならない決まりがあるため、行の最後は必ず改行で終わらせなければならないことだ。例中の「PackageURL」というのはJarファイル(iアプリ本体)のある場所を示しているので、この部分を自作のiアプリのファイル名にすればよい。

また、スクラッチパッドに書いた文章を保存しておくようなiアプリの場合には、「SPsize」の指定が必要である。最大5kまで指定できるので今回の例では1024バイトを指定した。なお、ネットワーク型iアプリの場合には「UseNetwork = http」という記述を忘れないようにしないと通信できなくなることにも注意したい。

●Jamファイルの例

```
AppName = NetMemo
AppVer = 1.0
PackageURL = NetMemo.jar
AppSize = 2622
KvmVer = CLDC-1.0
SPsize = 1024
AppClass = NetMemo
LastModified = Mon, 05 Feb 2001 17:48:24
UseNetwork = http
```


HTML（エイチ・ティー・エム・エル／Hyper Text Markup Language）：ホームページを設定するためのページ記述（マークアップ）言語。文書中に「タグ」と呼ばれる命令を挿入して、見出し、段落といった文書構造を指定したり、画像やリンクを設定したりする。HTMLにはバージョンがあり、1997年12月に発表された4.0が最新の規格だ。

CD-ROMに収録したデータの使い方

iアプリを理解するためにも、ぜひ本誌付属のCD-ROMに入っているサンプルのデータも参照して欲しい（374ページ）。今回製作したスタンドアローン型は「MemoAlone」、ネットワーク型は「NetMemo」というフォルダーに関連ファイルはすべて入っている。ADF（.Jamファイル）や、ソースコードは（.javaファイル）、エディターやメモパッドで見ることができるほか、より詳細なコメントも書かれているので、今後、同じ様なiアプリを作る時には参考にしてもらいたい。

なお、今回作ったサンプルを実際に自分の携帯電話にダウンロードして実証する場合も、自分のホームページエリアなどにファイルを置いて503iからダウンロードすればOKだ。その際、「NetMemo」の場合はPerlのCGIが使えることが前提である。CGIファイルとdatファイルの属性変更を忘れないようにしたい。なお、クライアントサーバー型iアプリはダウンロードしたサイトだけにしかアクセスできないほか、書き込まれたメモの内容のファイルはダウンロードしたサイト上にあるので不用意なファイルの移動は厳禁であることにも注意したい。

最後に参考までに、今回のサンプルで使用したクラスの一覧を右に挙げておく。実際にiアプリで使えるクラス自体はこの倍以上あるので、くわしくはNTTドコモのウェブにある仕様書 を参照して欲しい。

www.nttdocomo.co.jp/mc-user/i/java/

著者紹介

木寺祥友（きでら・よしとも）

95年に初期のJavaプロジェクトにたずさわった経験から、全米各地をまわりジェームズ・ゴスリングなどのJava開発者にインタビューし「Javaを創った人々」アスキー刊を執筆。アスキーで「Java訪ねて三千里」や、日経マルチメディアで「Drキデラの AVC指南」を連載。現在は株式会社エル・カミノ・リアル代表取締役。携帯電話とJavaが最大のテーマ。

YKidera@aol.com
www.el-camino-real.net

●今回のサンプルで使用したiアプリのクラス（API）一覧

提供元	クラス名	概要
ドコモ	IApplication	アプリケーションの雛型を提供。 iアプリのアプリケーションは必ずこのクラスを継承して作成しなければならない。
	start	アプリケーションが起動したら呼ばれるメソッド。
	terminate	アプリケーションを終了。
	getCurrentApp	カレントのアプリケーションオブジェクトを取得。
	getSourceURL	アプリケーションがダウンロードされた元の URL を取得。
	PhoneSystem	携帯電話のデバイスを定義する。 プラットフォームのネイティブリソースにアクセスする手段を提供する。
	setAttribute	ネイティブリソースの制御を行う。
	DEV_BACKLIGHT	バックライトを表わす (=0)。
	ATTR_BACKLIGHT_ON	バックライトの属性の1つで、オンすることを表わす (=1)。
	Display	デバイスのビューを定義する。 端末のスクリーンとキーパッドを抽象化したもの。 スクリーンやキーパッドの情報を取得するために使う。
	setCurrent	カレントのフレームを設定する。
	Frame	アプリケーションの表示面を提供し、持つべきインターフェイスを定める。
	setSoftLabel	ソフトキーのラベル文字列を設定する。
	SOFT_KEY_1	ソフトキー1 (=0)。
	SOFT_KEY_2	ソフトキー2 (=1)。
	Panel	高レベルAPI のための表示面を定義する。 これは高レベル API で使用するフレームクラスで、コンポーネントを張り付けるための親オブジェクトとなる。パネルがスクロールするかどうかは機種による。
	setSoftKeyListener	ソフトキーリスナーを登録する。
	setComponentListener	コンポーネントリスナーを登録する。
	setTitle	フレームのタイトル文字列を設定する。
	SoftKeyListener	ソフトキーの入力のリスナーを定義する。 高レベル API において、ソフトキーイベントを受け取るオブジェクトが実装すべきインターフェイスを定める。
	softKeyPressed	ソフトキーが押された時に呼ばれるメソッド。
	softKeyReleased	ソフトキーが離された時に呼ばれるメソッド。
	Ticker	ティッカーテープ表示 (マーキー) のコンポーネントを定義する。 高レベル APIで使用されるユーザーインターフェイス部品の1つ。
	TextBox	テキスト入力コンポーネントを定義する。 高レベル API で使用するユーザーインターフェイス部品の1つで、テキスト入力のためのコンポーネント。
	setText	テキスト文字列を設定する。
	DISPLAY_ANY	表示文字列をそのまま表示するモードにすることを表わす(=0)。
	Label	テキストを表示するコンポーネントを定義する。高レベル API で使用するユーザーインターフェイス部品の1つで、文字列を表示するためのコンポーネント。 1行テキスト表示のみをサポートする。
	setText	コンポーネントのラベル文字列を設定する。
	ComponentListener	コンポーネントのイベントのリスナを定義する。高レベル API において、コンポーネントで発生したイベントを受け取るオブジェクトが実装すべきインターフェイスを定める。
	componentAction	コンポーネントで発生したイベントを受け取る。
	HttpConnection	HTTP プロトコル形式の接続を定義する。
	setRequestMethod	メソッドを設定する。
	setRequestProperty	ヘッダーのプロパティー値を設定する。
	connect	HTTP で接続し、メッセージを送受信する。
	close	接続を閉じて、リソースを開放する。
	GET	HTTP リクエストメソッドの"GET"を表わす文字列。
	POST	HTTP リクエストメソッドの"POST"を表わす文字列。
	Component	コンポーネント (部品) を定義する。 高レベル API の部品を表わす抽象クラスの1つ。
CLDC	Connector	すべての接続オブジェクトの作成時に使用する「static メソッド」のプレースホルダー。
	open	Connection を作成およびオープンする。
	openDataOutputStream	接続出力ストリームを作成およびオープンする。
	openDataInputStream	接続入力ストリームを作成およびオープンする。
	openOutputStream	接続出力ストリームを作成およびオープンする。
	openInputStream	接続入力ストリームを作成およびオープンする。
	READ_WRITE	アクセスモード。
	READ	アクセスモード。
	DataOutputStream	これを使うと、アプリケーションはプリミティブ型のJava データを移植性のある形で出力ストリームに書き込める。
	writeUTF	文字列を、マシンに依存しない UTF-8 エンコーディングを使った形式にして基本となる出力ストリームに書き込む。
	close	出力ストリームを閉じ、これに関連するすべてのシステムリソースを解放する。
	DataInputStream	これにより、アプリケーションはプリミティブ型のJava データを基本となる入力ストリームからマシンに依存せずに読み込める。
	readUTF	Java 修正 UTF-8 形式でコード化された Unicode 文字列表現をストリーム in から読む。
	close	入力ストリームを閉じて、このストリームと関連するすべてのシステムリソースを解放する。
	InputStream	この抽象クラスはバイト入力ストリームを表現するすべてのクラスのスーパークラスである。
	read	入力ストリームからバイトデータを読み込む。
	close	この入力ストリームを閉じて、そのストリーム関連するすべてのシステムリソースを解放する。
	available	この入力ストリームのメソッドの次の呼び出し側からブロックされることなく、この入力ストリームから読み込める (またはスキップできる) バイト数を返す。
	OutputStream	この抽象クラスはバイトの出力ストリームを表現するすべてのクラスのスーパークラスである。出力ストリームは、出力バイトを受け付けて、特定の受け手に送る。
	write	この出力ストリームに指定されたバイトを書き込む。
	close	このストリームを閉じ、このストリームに関連するすべてのシステムリソースを解放する。
	String	String クラスは文字列を表わす。Javaプログラム内の""abc""などのリテラル文字列はすべて、このクラスのインスタンスとして実行する。
	Object	Object クラスはクラス階層のルートである。すべてのクラスはスーパークラスとしてObject を持つ。配列を含むすべてのオブジェクトはこのクラスのメソッドを実装する。
	equals	このオブジェクトと他のオブジェクトが等しいかどうかを示す。
	Exception	Exception クラスとそのサブクラスは通常のアプリケーションでキャッチされる可能性のある状態を示す。Throwableの形式の1つ。
	IOException	なんらかの入出力例外の発生を通知するシグナルを出す。入出力処理の失敗、または割り込みの発生によって作成される例外の一般的なクラス。

黒字＝クラスもしくはインターフェイス　赤字＝メソッド　青字＝フィールド

Perl（パール）：ラリー・ウォールによって開発されたプログラミング言語。当初、テキスト処理に重点を置いた言語として開発されたが、言語仕様の拡張がなされ、ファイル操作やネットワークプログラミングなども可能となっている。最近ではウェブページに双方向性をもたらすために、CGIに利用されることも多い。

［インターネットマガジン　バックナンバーアーカイブ］ ご利用上の注意

このPDFファイルは、株式会社インプレスR&D（株式会社インプレスから分割）が1994年～2006年まで発行した月刊誌『インターネットマガジン』の誌面をPDF化し、「インターネットマガジン　バックナンバーアーカイブ」として以下のウェブサイト「All-in-One INTERNET magazine 2.0」で公開しているものです。

http://i.impressRD.jp/bn

このファイルをご利用いただくにあたり、下記の注意事項を必ずお読みください。

- 記載されている内容（技術解説、URL、団体・企業名、商品名、価格、プレゼント募集、アンケートなど）は発行当時のものです。
- 収録されている内容は著作権法上の保護を受けています。著作権はそれぞれの記事の著作者（執筆者、写真の撮影者、イラストの作成者、編集部など）が保持しています。
- 著作者から許諾が得られなかった著作物は収録されていない場合があります。
- このファイルやその内容を改変したり、商用を目的として再利用することはできません。あくまで個人や企業の非商用利用での閲覧、複製、送信に限られます。
- 収録されている内容を何らかの媒体に引用としてご利用する際は、出典として媒体名および月号、該当ページ番号、発行元（株式会社インプレスR&D）、コピーライトなどの情報をご明記ください。
- オリジナルの雑誌の発行時点では、株式会社インプレス R&D（当時は株式会社インプレス）と著作権者は内容が正確なものであるように最大限に努めましたが、すべての情報が完全に正確であることは保証できません。このファイルの内容に起因する直接的および間接的な損害に対して、一切の責任を負いません。お客様個人の責任においてご利用ください。

このファイルに関するお問い合わせ先
株式会社インプレスR&D
All-in-One INTERNET magazine 編集部
im-info@impress.co.jp