Javaルーキーでも大丈夫!

勝手に作ろう! iアプリ

第1回 iアプリプログラミングの基本

木寺祥友＋株式会社エル・カミノ・リアル
Jump www.ecreal.co.jp

Javaで広がるケータイワールド

今年1月、NTTドコモからJavaアプリケーション「iアプリ」に対応した503iが発売された。このiアプリは今までのようなテキストや画像のブラウジングだけでなく、アプリケーションならではのこと、たとえば音楽を鳴らす、画像を操作する、文字を操作する、といったことができる。これによりゲームやスケジュール帳など、C-HTMLではできなかった、多くの表現ができるようになったのだ。

しかも、このiアプリは一般ユーザーでもパソコン1台あれば手軽に作成できる。しかも作成に必要なツールはサン・マイクロシステムズとNTTドコモが提供しているので、新たにソフトを買い足す必要もない。特にNTTドコモが提供しているDoJa Jump という開発ツールには、パソコン上で携帯電話をエミュレートする機能が付いているので、どのような動きをするのかが、携帯電話を使わなくても確かめられるのだ。ここまで環境がそろってきたのだから、これを機会にiアプリの作り方を覚えてみよう!

というわけで、今回から6回に分けて、iアプリの基本構造からどのようにすれば503i上で動かせるのかまでを、「MyIntro」というiアプリを例に解説する。MyIntroは自己紹介を楽しく演出するもので、音を鳴らす、画像を操作する、文字を操作するといったマルチメディア機能を使っているので、これをとおしてさまざまな表現方法を学べるはずだ。インターネットマガジンのウェブサイト Jump からダウンロードもできるので、ぜひとも試してもらいたい。また、CD-ROMにはサン・マイクロシステムズの提供しているJava 2 SDKが納められているので、Dojaとともにインストールすればソースを書き直すこともできる。これらのツールの使い方に関しては第4回に解説するが、すぐに試したい人はチャレンジしてもらいたい。

なお、iアプリは機種依存性が高いのだが、この連載では全機種対応版としてのiアプリの作り方もそのつど解説していく。どの機種を持っていてもちゃんと動かせるので安心してもらいたい。

Jump www.nttdocomo.co.jp/i/java/tool.html
Jump intranet.impress.co.jp/iappli/

このような表現も自由自在

文字をタイプライターのように打ち出す画面。アプリケーションならではの表現の1つだ。このようなテキストの表現は次号で解説する。

サーバーからデータを取ってきて表示している。iアプリをダウンロードしたあとでも、データだけ自動的に更新させるといったことができる。

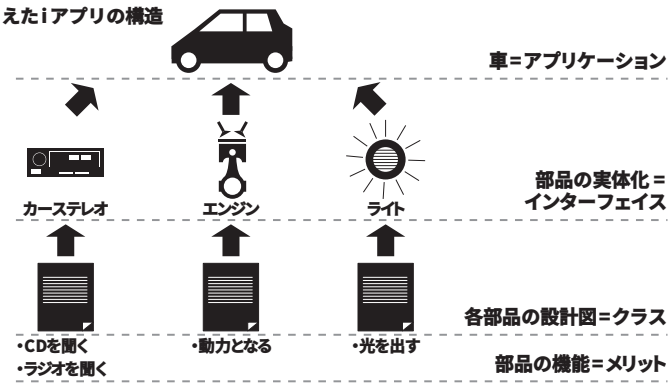
オープニング画面。音楽が流れ、幕が開き写真が出てくる仕掛けだ。画像の操作に関しては9月号、音楽に関しては10月号で解説するので、少し待って欲しい。

C-HTML：コンパクトHTMLの略称。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。

iアプリを車にたとえてみると・・・

iアプリを作るということは、Javaという言語を使ってプログラムを記述するということだ。Javaの場合、プログラミングといっても命令を羅列していくのではなく、部品を組み合わせていくイメージに近い。ここではiアプリのプログラミングとはいったいどのような作業なのかをイメージし、そしてこれからの解説で出てくる単語がどのような意味を持っているのかを理解できるように、車をたとえにして説明する。というのも、車の製造にあたって部品を組み合わせてゆくプロセスが、iアプリの構造に似ているからだ。図を見ながら読んでもらうとわかりやすいので参考にしてもらいたい。

■車にたとえたiアプリの構造



まずiアプリを語るうえで絶対に覚えておかなければならないのが「クラス」である。簡単に言うとクラスとは、「ある機能に関連した情報や操作を1つにまとめたもの」をいう。車の設計にたとえると、設計図のような位置付けのものと考えてもらってもよい。エンジンの設計図はエンジンクラス、車全体の設計図は車クラスといった感じである。

次に「メソッド」である。メソッドとは、クラスのなかにある機能をあらわしたものといえる。設計図ならば、それぞれの機能をどう実現するかを表現した部分がメソッドの内容となる。先ほどのエンジンクラスを例にとると、エンジンの始動や停止といった機能がメソッドにあたる。車を動かすときには、エンジンクラスのエンジン始動メソッドを使えばよいのだ。

そして次が「インスタンス」だ。今までは「設計図」の話ばかりだったが、車が部品の設計図を集めてもそれだけでは動かないのと同じように、iアプリも設計図だけでは動かない。設計図から部品という実体を作り、それを組み合わせてはじめて車が走るように、iアプリも設計図から実体化させる必要があるのだ。この“実体化”がインスタンスだ。

ここで理解しにくいのが、プログラムソースのなかで「実体を作る」という命令を書かなくてはならないという点だろう。これは、エンジンクラスという設計図のなかには「エンジンをいつ製造して、いつ納品してもらう」など、製造指示までが含まれていると考えるとよい。「車を走らせるのにエンジンが必要なので、エンジンの設計書をもとにエンジンを作れ」がインスタンスの生成、「エンジン、点火しろ」がメソッドの呼び出し、そんなイメージでよいだろう。

これで全体的な構造の概観が見えてきただろうか。インスタンスがちょっと難しいかもしれないが、少しずつ自分なりのイメージを深めていって欲しい。

ソースを書くうえでの約束ごと

インデントについて

iアプリはプログラム内でメソッドやクラスなどのまとまりを表すのに、中括弧{　}を使っている。少々長いソースになると、これが山と出てくる。これを、なんの工夫もなく記述していくと、とてもではないが中括弧がどこからどこまでを括っているのかわからなくなってしまう。そこで、決まりごととして“インデント”というものを使う。「{」から「}」の間のすべてをスペース2つぶんだけ右にずらして記述することで、中括弧の表す範囲を明確に表現できる。いわばソースコードを見やすくする知恵のようなものである。

```
{  //第1段落の始め
  {  //第2段落の始め
    {  //第3段落の始め

    }  //第3段落の終わり
  }  //第2段落の終わり
}  //第1段落の終わり
```

コメントについて

また、ソースを見やすくする知恵にはもう1つ「コメント」がある。このコメントは、プログラム上はまったく関係がない。人間がソースを見たときにわかりやすいようにするために書いておくメモのようなものである。書き方は2種類で、「/*」と「*/」で囲む方法と、「//」以降に記述する方法がある。「//」で指定すると、改行するまでがコメントとなる。

```
/*
  ここはコメントです。
*/
//ここはコメントです。
ここはコメントではありません。
```

それでは、次のページでは実際のソースを掘り下げながら見てみよう。

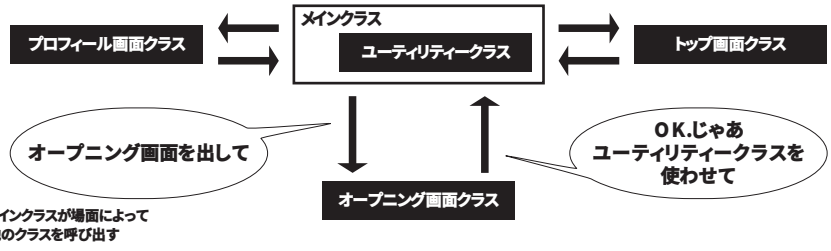
JAVA（ジャバ）：サン・マイクロシステムズ社の開発したプログラミング言語。ネットワーク環境で利用されることを前提とし、Javaで開発されたソフトウェアは特定のOSに依存せず、基本的にどのようなプラットフォーム上でも動作することになっている。

iアプリプログラミングのイロハ

iアプリは、メインクラスを中心にして構成されている。右図をみれば、その関係がわかるだろう。このMyIntroの場合、オープニングの画面の動作と、トップ画面の動作、プロフィール画面クラスの動作とで、それぞれクラスを形成している。

また、「ユーティリティークラス」という、どのクラスからでも使うメソッドを集めたクラスがあるのだが、通常iアプリではメインクラスに組み込まれる。これは、複数のクラスを形成すると全体の容量が増えてしまうことに由来する。総容量を10キロバイト以内にしなければならないというiアプリの厳しい制限から逃れるための知恵なのだ。

■メインクラスと他のクラスの関係(MyIntroの場合)



パッケージを呼び出す【9-12行目】

1つのアプリケーション内で使われるメソッドやクラスのすべてをソース内で定義するわけではない。パッケージという、すでに用意してあるクラスの集まりを呼び込んで使うこともあるのだ。たとえば11行目で、パッケージの1つでDisplayクラスなどが入っている"com.nettdocomo.ui"を呼び込んでいるのだが、これにより36行目のようにDisplayクラスが使えるようになるのだ。このようにパッケージを呼び出すことを「インポート」と言い、以下のように記述する。

```
import [パッケージ]
```

メインクラスをつくる【17行目】

アプリケーションの中心であるメインクラスの宣言方法は以下のようになる。

```
[宣言子] class [クラスの名前] [オプション]
```

[宣言子]:このクラスがどのようなものなのかを宣言している。"public"は、「みんなが見られる」といった意味合い。くわしく述べると難しいので、ここでは「メインクラスは"public"」と覚えてしまおう。

[オプション]:ここでは、"extends IApplication"となっているが、これは親クラスから機能を引継ぐ「継承」をするという意味である。これで何も記述することなく、親クラスのメソッドが使える。この場合は、IApplicationというクラスを継承し、これを使えるようにしている。継承の記述方法は、オプション部を"extends [親クラス]"とすればよい。ちなみに、メインクラスはIApplicationクラスを継承しなくてはならない。

変数をつくる【23-25行目】

iアプリでは、数字や文字、インスタンスなどは、器である変数に入れてやり取りする。ごはんを茶碗に盛るようなものだ。そのためには、その変数にどのようなものを入れるのかを宣言する必要がある。

```
[宣言子] [変数の型] [変数の名前];
```

MyIntro.java(メインクラス)

```
1    /**
2     * タイトル:  MyIntro
3     * 著作権:   Copyright (c) 2001
4     * 会社名:   El Camino Real, inc.
5     * @author Shinichi Iwashita
6     * @version 1.0
7     */
8
9    import java.io.*;
10   import javax.microedition.io.*;
11   import com.nttdocomo.ui.*;
12   import com.nttdocomo.io.*;
13
14   /**
15    * アプリケーションのメインクラス
16    */
17   public class MyIntro extends IApplication
18   {
19     // 定数(システム)
20     private static final int MAX_MSG_LEN = 1000;
      // メッセージの最大長(半角)
21
22     // 変数
23     static TopFrm  topF;   // トップパネル
24     static OpenFrm  opnF;   // オープニングキャンバス
25     static ProfiFrm proF;   // プロフィールパネル
26
27     /**
28      * アプリケーションが起動したら呼ばれる
29      */
30     public void start()
31     {
32       // フレームを準備
33       topF = new TopFrm();
34       opnF = new OpenFrm();
35       proF = new ProfiFrm();
36       Display.setCurrent( opnF );
37
38     }

※共通メソッドのため中略※

271  } // MyIntroクラスの最後
```

■:変数名 ■:コメント ■:外部ファイル名やカスタマイズ可能なところ

定義:メソッドやクラス、変数などを使う際には、それがどのようなものなのかをコンピュータに認識させる必要がある。これを定型に従ってプログラムソース上で行うことを定義という。

[宣言子]:"static"の部分。これは少し特殊な宣言子なので、くわしくは次の機会に説明する。一般的に、このクラスのなかだけでしか使わない変数の場合は"private"を使う。
[変数の型]:この変数が何を格納するものかを書く。ここではインスタンスを格納する変数を宣言しているので、クラス名が書かれている。
[変数の名前]:名前。変数の場合は先頭を小文字にするのが決まりである。

最初に動くメソッドをつくる【30-38行目】

iアプリは、起動するとまず最初にメインクラスのなかのstartメソッドに記述されている動きをする。従って、メインクラスのなかには必ずstartメソッドが必要になる。記述方法は以下のとおりだ。

```
public void start ()  {
  ※ここに一番最初に実行する内容を記述する※
}
```

インスタンスをつくる【33-35行目】

クラスは設計図のようなもので、そこに書かれている機能を使うには、生成して実体化させる必要がある。

```
[クラス型の変数] = new [クラス名]
```

生成されたインスタンスは変数に格納されるので、クラス型の変数が必要になる。この前に用意しておこう。

メソッドを使う【36行目】

クラスが実体化してインスタンスになると、その機能を使えるようになる。記述方法は、以下のとおりだ。

```
[インスタンス].[メソッド]([引数])
```

引数とは、そのメソッドに与える値のことだ。値と言っても、数だけではないので注意してもらいたい。36行目では、インスタンスを格納した変数（opnF）を指定している。
ところで、その36行目で呼び出しているメソッドのsetCurrentはDisplayクラスのものなのだが、メソッドを生成することなく使われていることに気が付いただろうか。実はメソッドの中にはインスタンスを作らなくても使える「スタティックメソッド」というものがあり、このメソッドもこれにあたる。これらは例外として考えておこう。

次回以降は、iアプリの文字や画像、音楽などのマルメディア機能をどうのようにして実現させるのかを見てゆく。来月は文字列をタイプライターのように打ち出すメソッドを中心にして進める。楽しみにしてもらいたい。

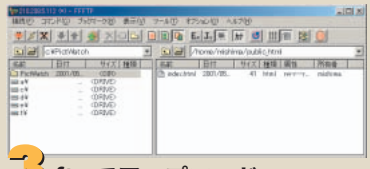
今月のオマケ！

3分で完成するオリジナルiアプリ
「自分の顔の時計を作ろう」！

ここでは、ちょっとした変更だけで作れる自分だけのiアプリを紹介してゆく。
今月は好みの画像を背景にできる時計「PictWatch」だ。

1 まずはダウンロード!

PictWatchは、Yahoo! internet.impress.co.jp/iappli/に置いてあるので、PictWatchフォルダーを丸ごとダウンロードしてしまおう。



3 ftpでアップロード

変更したPictWatchフォルダーをそのまま自分のホームページエリアなどにアップロードしよう。

4 完成!

アップロードしたフォルダーのなかにあるPictWatch.htmlにiモードでアクセスすれば、オリジナルiアプリをダウンロードできる。これだけで自分のiアプリを使えるぞ。

2 画像を差し替えろ

```
1 PackageURL = PictWatch.jar↓
2 AppSize = 8042↓
3 AppName = PictWatch↓
4 AppVer = 1.0↓
5 AppParam = ↓
6 AppClass = PictWatch↓
7 SPsize = 5120↓
8 UseNetwork = http↓
9 LastModified = Tue, 01 May 2001 20:38:48↓
10 KvmVer = CLDC-1.0↓
11 [EOF]
```

PictWatchフォルダーのなかには、back.gifがある。これを差し換えるだけで、背景の画像を変更できる。ただ気をつけてもらいたいのが、gifファイルの大きさだ。5120バイトまでしか入らないのだ。それ以上になる場合はjamファイルのSPsizeにある数字を大きくする必要がある。最大で10240バイトまで大きくできるぞ。

オリジナルiアプリ募集中!

今月のオマケ!は、バイト数の制限のなかでどれだけ画像をキレイにできるかがポイントだ。うまくできたら、オリジナルiアプリをアップロードしたサイトのURLを編集部宛にメールしよう。もしできなくても、質問でも感想でもなんでもOKだ。ドシドシ送ってきてもらいたい。宛先は次のとおり。多数のメールを期待している。 im-iappli@impress.co.jp

もっと知りたいキミにはコレ!

『今すぐできるiアプリプログラミング』
小社発行 本体価格2,400円
来月なんて待ってられない、もっとくわしくいろいろと知りたい!と思っている君にはコレがオススメ。この連載で取り上げているiアプリについても解説しているので、一緒に読んでいけばもっとよくわかるぞ。

Jam(Java Application Manager)ファイル:拡張子.jamをつけたシフトJISコードのテキストファイル。iアプリがダウンロードされる際に1番最初に読み込まれ、携帯の空き容量や互換性をチェックするのに使われる。

［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