# Oから始めるiアプリ(2)

環境を整えることは難しくなくなりました。

前 章 ではインストールまでを体 験しましたが、この 章 ではビルドとよばれる Javaプログラミングの 一 般的な 作 法について 学 習しましょう。

また、i アプリ開 発 ツー ルを 使 えば、Javaの プログラミング 開 発 は 初 心 者 にも 容 易 になりますが、その かわり、それ ならではの 作 法 を覚 えなくてはなりません 。

ただし、その行 程 は 一 度 覚 えてしまえば あとは 繰り返しです 。

この単元で開発工程を体得しましょう。

### 1. ユー ザー が i アプリを 利 用 する 流 れ

ー 般 ユーザーが i アプリを利用するにはホームページからダウンロードして、自分の携帯電話で実行 します。

セキュリティ的には現在はキツメの設定になっていますので、実機で実行といってもできることは限られています(ユーザーのアドレス帳をiアプリでは利用できない等)。

ですから、iアプリでもダウンロード型のアプリケーションも開発できるのですが、それはダウンロードしたサイトのみデータをダウンロードできるような使用になっています。

さて、ユーザーのiアプリ利 用 は 次のような 手 順をふむことになります 。



① 503iシリーズでiアプリが 掲 載されているサイトへユーザーがアクセス する 。

(iモードを使ってiアプリ掲載サイトへアクセスし始める)



②iアプリをダウンロードするハイパーリンクをクリックする。
 (ダウンロードのハイパーリンクが表示される)



③[決定]ボタンを押す。 (ダウンロードが始まる) 14.01 14.04

iTimer 氷塔 ThelceTower kabu.com指標情報

-

× Tul 一覧





⑤ダウンロードされた i アプリが 選 ばれているのを確 認の上、[決定]ボ タンを押す。 (Javaが VMが 起動する)

⑥ i アプリが実 行される。 (このときは 通 信 費 はかかっていない。実 機 のみでの実 行。)

②の行程ではユーザーはiモードを使ってホームページにアクセスしています。

ただ、サイト側では、ホームページのハイパーリンクには、ADFと呼ばれるiアプリのファイル名や作成 年月日などがかかれた設定ファイルを読みにいくようになっています。

実機の方ではその設定ファイルの中身を読み、実際のiアプリの実体を読み込みます。 つまり、iアプリを利用してもらうために、開発者は次のファイルを用意することになります。

- (1)iアプリ ~ .jar
- (2)ADFファイル ~ .jam
- (3)ホームページファイル ~.html

(2)と(3)はテキストファイルですが、(1)はJavaで書かれたバイナリー形式です。ですから、開発
 ツールなどがなくとも(2)と(3)はテキストエディタ等で開発が可能です。

ADFファイルには、iアプリ実 行体のファイル名、最終更新日、アプリケーションサイズなどiアプリ実行体に関する情報が記述されています。

(iアプリ開発TOOLを使用すれば、自動的に生成されます)。

#### 2. i アプリ開 発 の 流 れ

i アプリ開 発の基本的な流れは Javaアプリケーション開発のそれとほぼ同じ行程を経ます。しかし、 携帯電話特有の行程であるプリーベリファイという行程が加わります。

①設計

② ソース記述
③ ビルド(コンパイル)
④ ビルドがうまくいかなければ、デバッグをし再びビルド。
⑤ プリーベリファイ
⑥ エミュレータによる実行確認
⑦ アップロード
⑧ 実機テスト

ビルド

ソースをテキストエディタ等で記述した後、保存をし、それをパソコンが理解で きるようなデータ形式に変換することをいいます。C 言語など多くの開発言語では この行程を経ます。 データ形式をパソコン用に変換して実行しますので、実行速度が一般的には 速い点が優れています。しかし、HTMLなどのスクリプト言語と比べると、開発に

は若干手間がかかります(変換してからでないと実行テストができない等)。

プリーベリファイ 一般的な Javaでは 実行 されるパソコンごとに 実行 可能 かどうかを、ダウンロード後、チェックしてから実行という手順をふみます (セキュリティ的に配慮しているのです)。

しかし、携帯電話でのJavaではそのような機能を搭載すると、携帯電話本来の軽さが損なわれてしまいますので、実行確認を事前にしておいて、ダウンロード してもらうような仕様になっています。

その事前確認をプリーベリファイといいます。

ー 般 的には、この行 程で最も開発時のミスが発覚しますので、開発者にとって は助かると言えば助かります。

また、プリーベリファイが 成 功 した 後 は、jarファイルという実 行 形 式 の バイナリー ファイルに、画 像 ファイル など 必 要 なファイルをすべてまとめてくれます。

エミュレータ 携帯電話に限らず現在のパソコンでは、ゲーム機などの処理を模倣し、実際 にそのマシンがパソコン上で動いているようにさせるソフトがあります。

そのソフトを総称してエミュレータといいます。

i アプリ開発では、実機でのテストはパケット料金がかかることや、プリーベリファ イの必要性があるため、パソコンに携帯電話と同等の実行環境を用意する必要 があるのです。その代表的なソフトがi−JADEです。

現 在 では 、p 、n 、f を 模 倣 した 動 作 確 認 が できるようになっています 。 開 発 では 実 機 テストの 前 に、エミュレータでテストするのです 。

【ワンポイント】DoJaだけではダメ? !

2001年4月27日にNTTドコモ社から発表されたエミュレータで、すでに次のような実機とのズレが発 覚しています。

(1) MLDファイルが再生されない

→ MIDIという形 式 は 再 生 可 能 な の で、テストは MIDI形 式 にし、確 認 後 は MLDというi メロディの 形 式 に 変 換 して アップロー ドするようになります。i ー JADEでは 正しく再 生 されます。

(2)表示可能領域が違う

→ 一 般 的な 実 機 では 120\*130 ドットという解 像 度 ですが、DoJaではそれよりも小さいので、画 像 をあ つかうi アプリを開 発 の際には、やはりI-JADEを併 用 することになります。

しかし、実際にはi – JADEでも色再現が実機と違ったり、実行速度が機種ごとに違ったり(一般的にはpは遅いようです。アクションゲームなどには適していない、というのが業界の共通した考えのようです)2001年4月27日現在発表されているDやSOなどのエミュレータは発表されてませんので、やはり最終的には実機での確認が必要になります。パケット料金が早く下がるといいですね。

## 3.はじめての i アプリ作 成 ~ 作 業 フォルダの 作 成 ~

それでは、実際にiアプリを開発してみましょう。

プログラミングの勉強で一番最初に知る、画面上に文字を表示させる最も伝統的な課題です。 仕様ソフトは、SeiichiNishimura氏のiアプリ開発Tool DoJa版です(前節参照)。

【課題1】画面上に「Hello!PC」と赤い色で表示させなさい。

- 【仕様】 表示文字
  - 表 示 色 表 示 座 標 値 ソース名
- : Hello!PC : getColorOfRGB(255,0,0) ・・・赤です(詳しくは後章)。 : x座標30 y座標60 : Mk01
- 【参考画面】



```
【入力ソース】------
import com.nttdocomo.ui.*;
public class Mk01 extends IApplication {
      public void start(){
             Display.setCurrent(new Mk01d());
      }
}
class Mk01d extends Canvas{
      public void paint(Graphics g){
             g.lock();
             g.clearRect(0,0,getWidth(),getHeight());
             g.setColor(g.getColorOfRGB(255,0,0));
             g.drawString("Hello!PC",30,60);
             g.unlock(true);
      }
}
                         _____
```

①[スタート]→[プログラム]→[Seiichi Nishimura iアプリ開発]→[iアプリ開発Tool DoJa版]をクリック します。



(iアプリ開発 Tool DoJa版が起動します。2001年04月27日現在はバージョンが0.7になっています)

🛃 Do Ja版 🛛 🗖 💌	
バージョン 使い方	【 リ ノ 小 イ ノ ト 】 バ ノ リ 用 9元 lool DoJa 颀 の 画 面 ① i アプリを新 規に 作 成 する 際 に、プロ ジェクトというi アプリ専 用 フォルダ の トラカま のち 佐 ろの ですが、 ろわち 佐 成 」ます
環境設定	しょうなものを1Fるのとりか、と4iを1F放しより。
新規プロジェクト①	② プロジェクトを作 成 後 、実 際 のソースを入 力 していくのですが、そのソー スファイルがリストアップされる 欄 です 。
	③ プロジェクトを 作 成 後 、この [追 加 ] ボタンをクリックし、ソースを記 述 す る 実 際 のファイル 名 (メインクラス名と表 記 されます)を入 力します。
Javaソースファイル	
メインクラス	④メインクラスを追加した後、この[編集]ボタンをクリックし、ソースを入力 していきます。クリックするとデフォルトではメモ帳が起動するようになって
(2)	
	⑤ 新規にプロジェクトを作成する際には、メインクラス欄にリストアップさ
迦興縣	れているファイルをこのボタンを使って削除します。   その際にけ 実際のフォルダからも削除されてしまうので 注音が必要
Dry Harts	です。必要なソースファイルは必ず別な場所に移動しておきましょう。
6	⑥ 後 述 の 画 像 ファイルを 扱 いには 、この 欄 に 、[追 加] ボタンをクリックし 、 そのファイルをリストアップしておかなくてはなりません 。
追加 編集 削除	⑦ NTTドコモ社から提供されているDoJaにて実行 確認する際には、この
L <sup>*</sup> ルト <sup>*</sup> JAM変更	ボタンをクリックして、確 認 します 。 すぐ上 に、プリーベリファイなどのメッセー ジも表 示 されます 。
	│ │⑧ i ー J A D E を 使 って 実 行 確 認 する 際 にクリックします 。
DOJA	下にあるメニューで、p503i、n503i、f503i用のi ー JADEを起動できます。
設定 デバッグテスト	⑦ ◎ 0 動 作 ニュトギタンけ プリー ペリコッイが 戌 ユー てち すぐに 消っ
	てしまうことがあります。その際には、ソース内正しくない記述がある場合
	がほとんどですので、プリーベリファイが成功してもソースを再確認するよ
	うにしてみ ましょう。 
iJAD 8 動作テスト P マ デバッグテスト	① の上 部にある[環 境 設 定]ボタンを使 えば、ソースを入 力するときのエ ディタを変 更 することができます。使い 慣れたものがあれば、それを設 定し ておくと便 利 でしょう。

② [新規プロジェクト]ボタンをクリックします。
 (新規プロジェクト保存ダイアログが表示されます)

③保存する場所に(C:)ドライブを選びます。

新規プロジェクト	
保存する場所型:	2

④ [新しいフォルダの作成]ボタンをクリックします。

<u>?×</u>	
M 📬 🛅	
い 新しいフォルダの	作成

⑤ 直 接 入 カモードで kyouzaiと 入 力します。

🔜 kyouzai	_	
	h	ì

⑥[ENTER]キーを押します。

⑦ 作った kyouzaiフォルダをダブルクリックして、開きます。

新規プロジェクト				<u>?×</u>
保存する場所①:	🔁 kyouzai	•	🗈 🗹	📸 📰 📰
ファイル名(N):	Mk01			保存(S)
ファイルの種類(工):	ijp7ァイル		•	キャンセル
	·			//,

⑧ 直 接 入 カモードで、Mk01と入 カします。
 ⑨ [保 存]ボタンをクリックします。

【ワンポイント】作業場所は(C:)ドライブ内で。

i アプリで 開 発 するといっても、Javaというプログラミング 言 語 の 開 発 作 法 に 従うことは 変 わりません。 Javaでは、コンパイル (ビルド)という変 換 作 業 をする 際 に、Windows以 下 で 動く、DOSというプログ ラムが 基 本 になります。

× kyouzai\_iappli

 $\times 01$  Mk

その変 換 作 業 においては、(C:)ドライブ以 下 にフォルダがあり、その名 前も・・・、

・直接入力モードで、英数字を含む8文字以内。

- ・数字や記号でその名前がはじまってはいけない。
- ・大文字小文字は認識しない(フォルダ名)。

上記の条件を満たしたソースでないと、正しくビルドできません。 それは、Javaがビルドする際に、DOSというプログラムのルールに依存するからなのです。

# 4.はじめての i アプリ作 成 ~ 作 業フォルダの 作 成 ~

①メインクラス側の[追加]ボタンをクリックします。

追加	稨
UV-277	

② 直 接 入 カモードで、Mk01と入 カし、[OK]ボタンをクリックします。

新しいりラン	えの追加		×
クラス名	Mk01		
🗹 דער 🗹	をかんのメインクラスにする		
		ОК	キャンセル

③[編集]ボタンをクリックします。

Javaソースファイル
メインクラス Mk01
Mk01.java
リソースファイル(GIFなど)

④直接入力モードで正しく入力します。



⑤ [ファイル]→[上書き保存]を選びます。 (とくに画面上変化はありません。) ⑥ [ビルド]ボタンをクリックします。



([ビルド]ボタンをクリックすると、ソースが正しければ以下のようにメッセージが表示されます。

コンパイル中 → フツーヘツファイ中	$\rightarrow$	jarアーカイブの作成中。	
アウトフット ビルトキま成功しました 作成されたjarファイルのサイスキ 841 ハイト			× 閉Uる

(ソースが 不 適 切 な 場 合 は 以 下 のように表 示 されますので、メモ 帳 画 面 で 再 確 認 し、再 び 上 書 き 保 存 し、[ビルド]ボタンをクリックし、ビルドを試 みます。)

ፖሳትን የሃት	<u>×</u>
C:¥kyouzai¥Mk01.java:13: 式の開始が不正です。 	
エラー 1 個	
	<b>T</b>

(正しくビルドされた場合には、(C:)ドライブのkyouzaiフォルダには以下のように保存されています。)

🚖 C:¥kyouzai						<u>- 🗆 ×</u>
] ファイル(E)	編集( <u>E</u> )	表示(⊻) 移動(	<u>(G</u> ) お気に入り( <u>A</u> )	ヘルプ(田)		
🗭 🗸	→ ・ 進む	た。 上へ 切り	/ Ling Ci 加 コピー 貼り	13日本 12月 1月日 - 元に原	) 入 えす 削除	»
דיאס 🖸	🗋 C:¥kyouz	ai				•
preverified	unverified	کی Mk01.htm	Mk01.jam	Mk01.jar	Mk01.java	
。 6 個のオブジェク	ŀ		1.55KB 🛄 74	ב°שעב		

⑦ [動作テスト]ボタンをクリックします。

iJADE	動作テスト
P	デバッグテスト

マイコンピューオ マイコンピューオ マイトドキュントト Ditement Explorer コンピューオ ロンピューオ ロンピューオ ロンピューオ ロンピューオ ロンピューオ ロンピューオ ロンピューオ ロンピューオ ロンピューオ	Acrobat Reader 4.0 Explorer.exe - X.BLIte2 i72+1- EU/5- Acrobat 4.0 NotePP.exe Mk03.java Mk03.java	PDF Output i77)IRSE ir it Mk01.java al.bmp Mk02.java Mid_mid.exe Photoshop Mk04.java		パージョン     1       パージョン     1       現境認知     新規フロジ:       フロジェクトを     3       Java)~2.7.7(ル     1       メ(ソウラス     Mk01       通知     編集       リンースファイル(GIF       通知     編集       ビルド     リ       DOJA     動ff       設定     アハン       延明書     スクラッチハ・アトド       IJADE     動f       ア     アハン	■ 更い方 一 別 小 小 小 「 別 除 一 一 一 一 一 一 一 一 一 一 一 一 一		×
<b>∭1</b> ,29−1	,	≥  ] <u></u>	太郎Lite - [0から	<mark>ม<sub>ีอ่า</sub>DoJali</mark> ช	<u>@</u> ]Mk01.java - メモ帳	 ي بينين من	

(i - JADEが 起 動し、実 行 画 面 が 表 示 されます)

⑧ i ー JADEの[×]ボタンをクリックし、i ー JADEを終了させます。 (動作テストごとに新規のウィンドウで実行画面が表示されるため)

⑨ iアプリ開発 Tool DoJa版の[×]ボタンをクリックし、終了します。

#### 【ワンポイント】開発はたいへん!?

iアプリ開発Tool DoJa版を使えば、ビルドやjarファイルの生成が楽になったといっても、ソース記述のたいへんさは変わりません。

市 販の開発ツールなどでは長い命令などは自動的に、つづりが表示されたり、カッコの数もチェック してくれたりして、とても便利です。

メモ帳で記述する以上は、ミスは当たり前と考えましょう。回数をこなしてくるとメッセージから、だい たいの場所を推察できるようになります。

初歩の段階でよくあるミスは次のようなものです。

・カッコの数があっていない。

・スペルミス × IApp1ication O IApplication (1と小文字のLを間違えている)

・セミコロンが抜けている。(これは Javaの文法として、命令の最後には必ずつけるものなので、しっかりと体に覚え込ませましょう。)

・修正をしているのに、上書き保存をせずにビルドしている。(上書き保存をしなければ、ビルドは前のソースを使って行われます。また、メモ帳では[CTRL]+[S]というショートカットも使えませんので、注意が必要です。)

・意外と、voidをviodと記述するように順番違いというつづりミスも多いです。