Java Watch on the X》
JBuilder と Project Builder を併用する

Mac OS X 版の JBuilder 6 はすでにリリースされているが、Java VM 1.3.1 Update1 がリリースされる前は、日本語のシステムで使うのはほぼ不可能に近かった。Update1 におより、英語版として配付されている JBuilder 6 もなんとか使える範囲に入ったようなので、さっそく活用したいところだが、パッケージ形式のアプリケーションを組む場合には Developer Tools を利用したいと考えるかもしれない。また、JBuilder 自体の動作はやはり Java で動いているだけに、細かいデバッグは Project Builder の方が効率的かもしれない。そこで、JBuilder で GUI をデザインし、Project Builder でビルドするといった使い方を紹介しよう。

JBuilder 6 Personal を使ってみる

Java の開発ツールの中で、JDK を除くとユーザがいちばん多いのはボーランドの JBuilder であるという調査結果もあるほどのメジャーなツールだ。Java で作られたツールであるとは言っても、Mac 向けにリリースされたのは初めてだ。Mac OS 向けにリリースされなかったのはいろいろ理由があるとしても、JDK 1.1 ベースだったことが大きいだろう。だが、Mac OS X は比較的新しい Java VM が組み込まれている。Windows や Solaris との遅れがあったことは大きくばん回したが、その結果、JBuilder のリリースへとつながった。

JBuilder はいくつかのバージョンがあるものの、ボーランドのサイトでのオンライン販売は米国などの地域だけで、基本的には日本向けには販売されていない。日本ではまだ Mac OS X 版の JBuilder 6 は販売されていないので、結果的には米国の Borland のサイトからダウンロードできるフリー版の JBuilder 6 Personal だけが入手できるということになってしまう。Personal 版は機能制約されているが、Swing ベースの GUI をデザインツールで作成してアプリケーションを作る点については、制約なく利用できる。

◇JBuilder 6 Personal

http://www.borland.com/jbuilder/download/jb6personal_steps.html

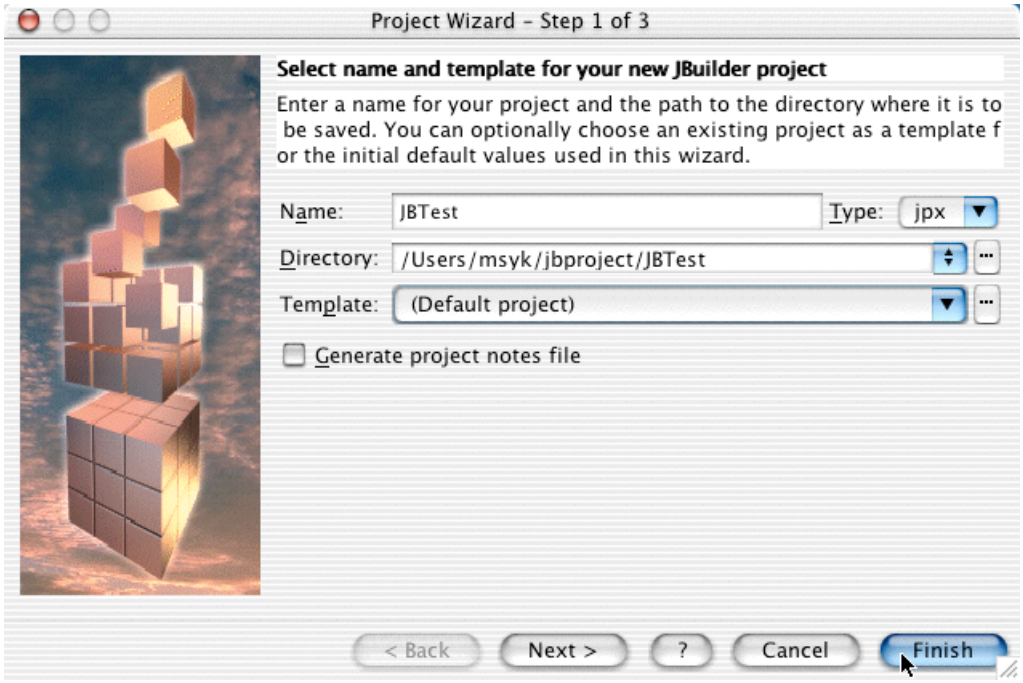
JBuilder 自体の説明も必要かもしれないが、今回はおおまかに説明しよう。JBuilder もアプリケーションとなっているが、エディタ、GUI ツール、デバッグ機能などすべて 1 つのアプリケーションに統合化されている。だから、開発作業は JBuilder を起動して作業すればそれだけで OK である。

もちろん、Java のソースのエディタなどもあるが、Java のソースは*.java ファイルに保存するなど、いくつかのファイルを作成するのは他のツールと変わらない。また、プロジェクトというものを 1 つのファイルを作って管理し、ある開発アプリケーションなどを 1 つにまとめるような状況になっているのも同様である。ただ、通常の利用では、class ファイルを特定のフォルダに生成してそれを実行する形式となっている。だから、1 つのプロジェクトでは、ファイルやフォルダがたくさん用意されることとなる。

JBuilder でプロジェクトを作成する

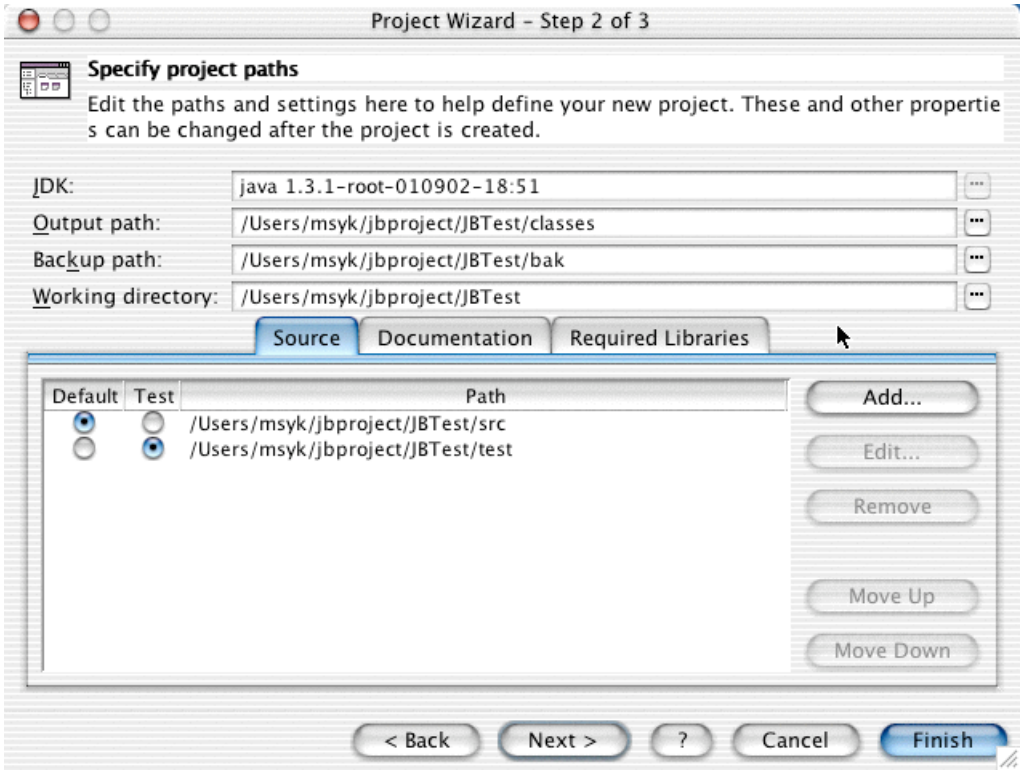
実際に JBuilder でアプリケーションを作ってみよう。プロジェクトを作成するには、File メニューから New Project を選択するのが 1 つの方法だ。以下のように、Project Wizard のダイアログボックスが出て来るので、必要な情報を入力する。最初は、プロジェクト名を入力するが、JBuilder では、ホームフォルダの直下に jbproject フォルダを作り、そこのプロジェクトを集めておくといったことがデフォルトになっている。

Project Wizard の最初の画面



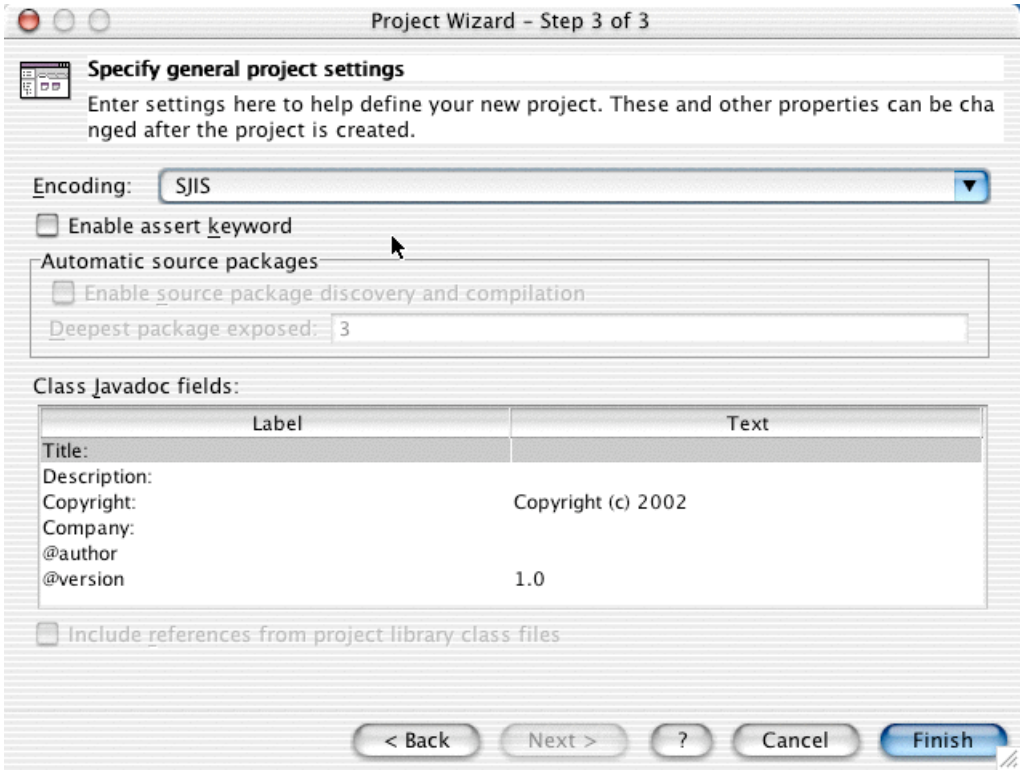
Project Wizard の 2 つ目のダイアログボックスではいろいろな設定があるが、基本的にはこのままでいいだろう。ここで、Output Path は生成した Class ファイルが作られる場所であるが、プロジェクトのフォルダの下にある classes フォルダとなっている。また、ソースはプロジェクトのフォルダの下にある src フォルダとなっているが、こうしたフォルダ構成が JBuilder でのプロジェクトの基本となっている。

Project Wizard の 2 つ目の画面



そして、Project Wizard の 3 つ目の画面では、JavaDoc のフィールド設定などがあるが、ここでは Encoding で SJIS つまり Shift JIS を選択しておくのが基本だろう。そして、Finish ボタンをクリックすることで、プロジェクトが新たに作られる。

Project Wizard でエンコードの設定を行う

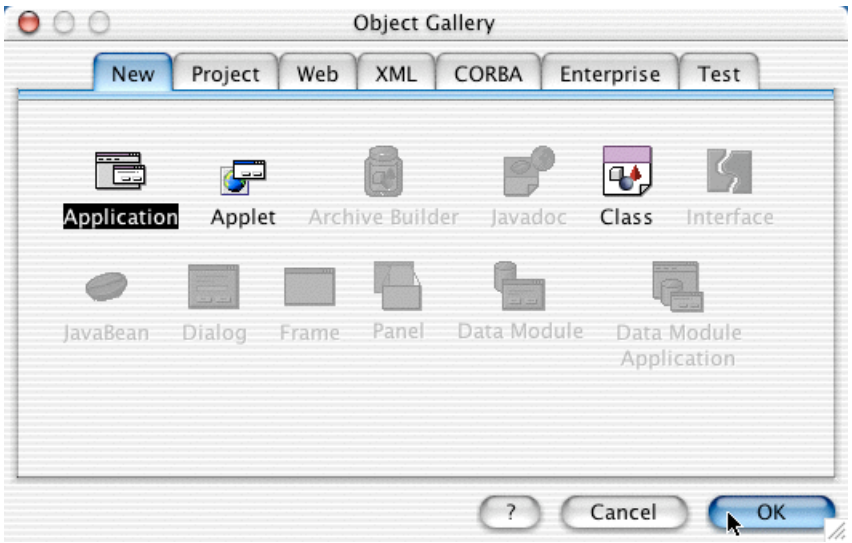


こうして作られたプロジェクトのフォルダ構成を Finder でチェックしていただきたい。プロジェクトのフォルダにはいくつかファイルがあるが、プロジェクトのファイルは JBuilder アプリケーションに関連付けられていないので、プロジェクトのファイルをダブルクリックして開くといったことができない。これはリリースのアップで改善されることを期待するしかないだろう。

プロジェクトにアプリケーションを追加する

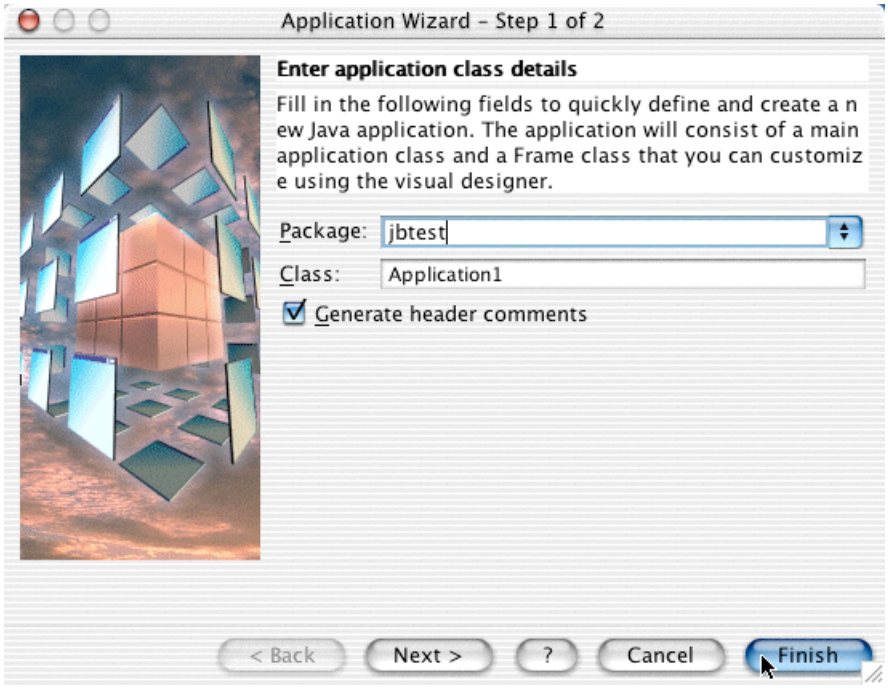
JBuilder ではアプリケーションについても基本的なひな形を簡単に作成する機能が用意されている。File メニューの New（Command+N）などで Object Gallery を呼び出して行うことができる。以下のように、Personal 版は限られたオブジェクトしか作ることはできないが、幸い Application の作成はできる。

Application を作成する



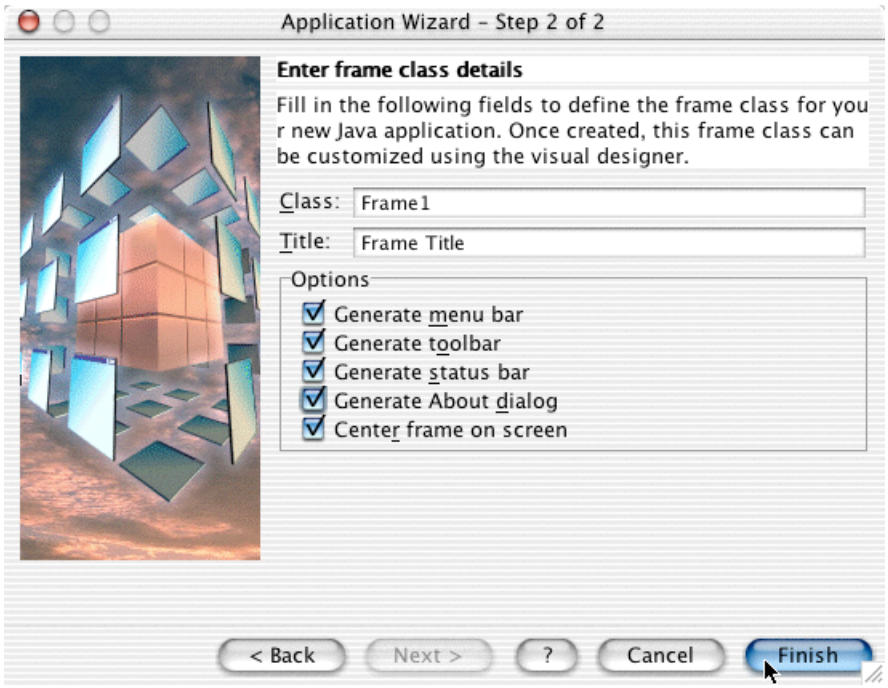
そうすると、Application Wizard が呼び出される。最初の画面では、パッケージ名とクラス名を入力する。デフォルトでもかまわないけども、JBuilder ではパッケージをきちんとつけるというのがスタイルである。Project Builder ではパッケージの指定は任意的な感じだが、JBuilder ではきちんとパッケージを指定するのが基本である。デフォルトでは、プロジェクト名を全部小文字にするなど、システム的には自動的に作ってくれる。

アプリケーションのクラスと組み入れるパッケージを指定する



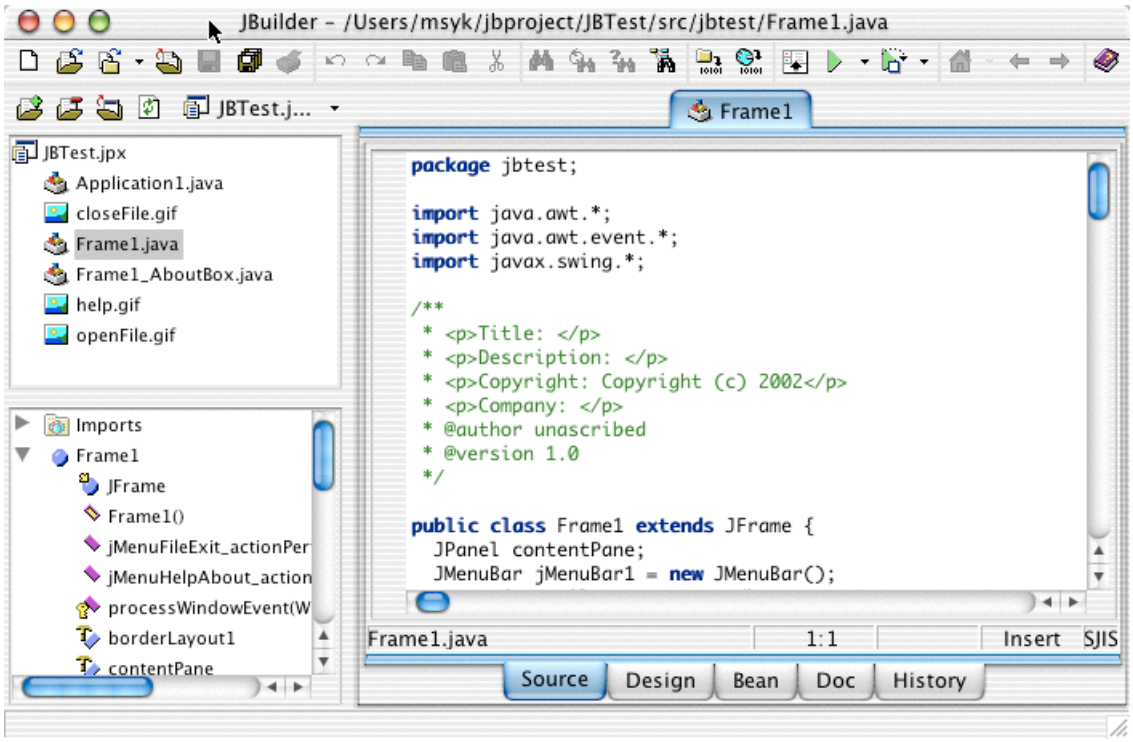
2 つめのダイアログボックスでは、アプリケーションのメインクラスに加えて、ウインドウを表示するためのクラスの定義が行われる。クラス名などは適当に入力するとしても、Option として、メニューバーやステータスバーなどさまざまなものが自動的に作られる。ある意味では便利だが、生成されたソースを一度は解析しておく必要はあるだろう。

ウインドウの諸定義を行う



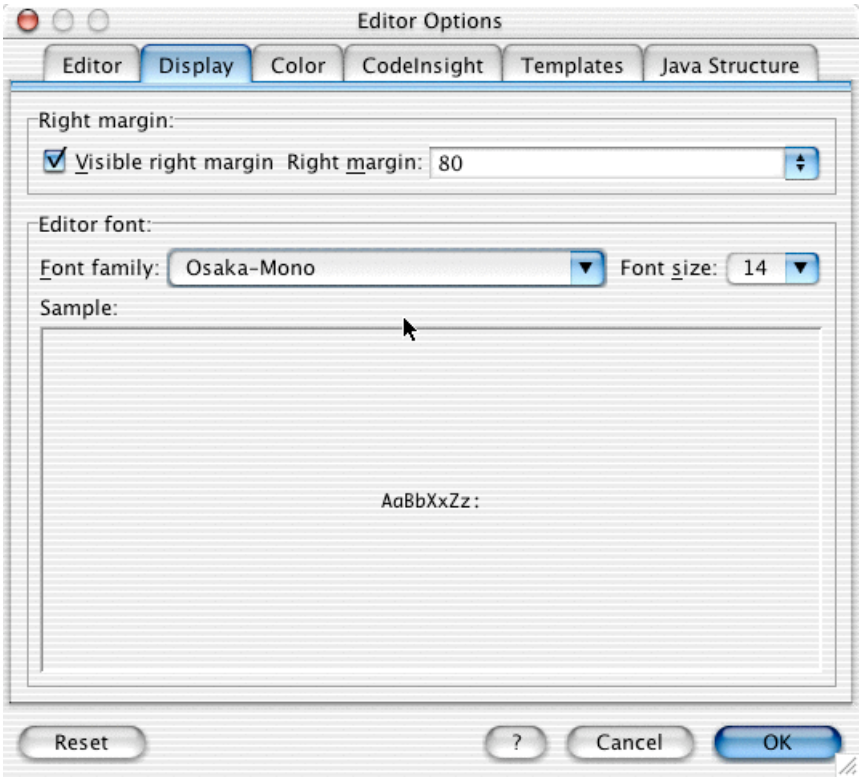
こうしてプロジェクトには、アプリケーションの本体とも言える Application1 と、ウインドウを表示するための Frame1 というクラスが定義された。プロジェクトではそれらのファイルが見えているが、アバウトボックスのウインドウのクラスや、ツールバーのアイコンの GIF ファイルなども見えている。

プロジェクトに作られたいくつかのクラスや画像ファイル



ここで、エディタ部分で利用するフォントについてはカスタマイズできるので、設定を行っておこう。Tools メニューの Editor Options を選択して、Display のタブを選択すると、エディタで使うフォントを指定できる。もちろん、好みでかまわないが、Osaka-Mono で 14 ポイントあたりが無難なところだと思われる。

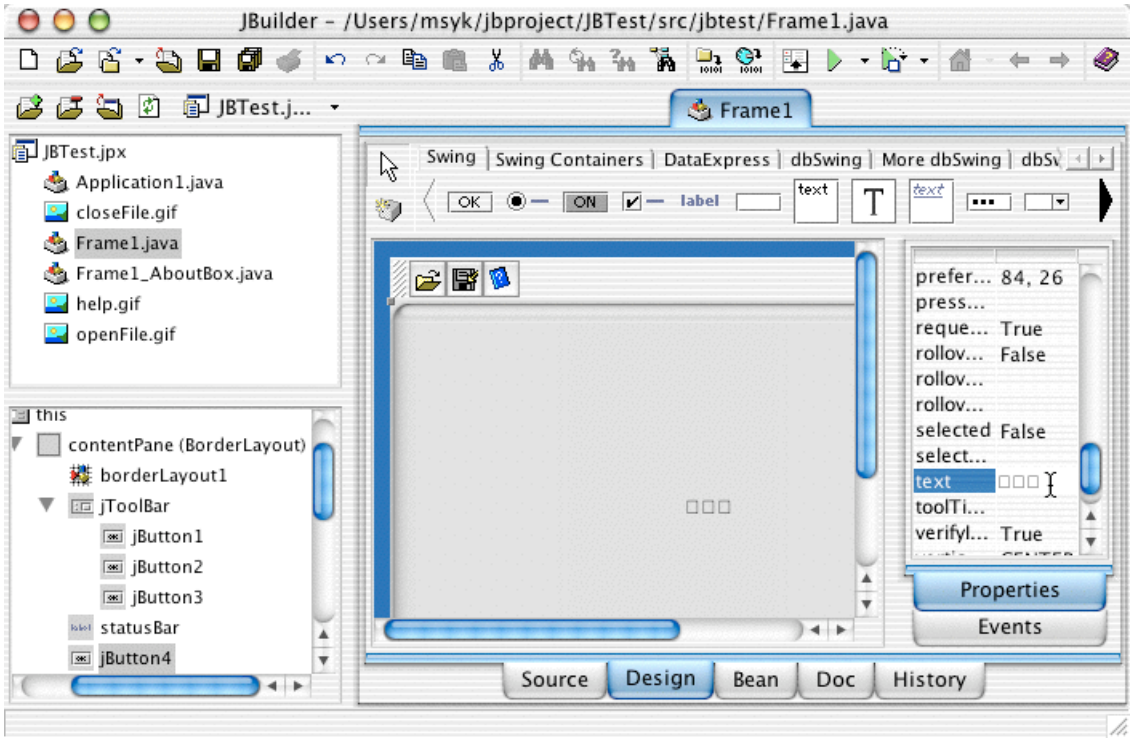
エディタで使うフォントを指定した



JBuilder のデザインツールを使う

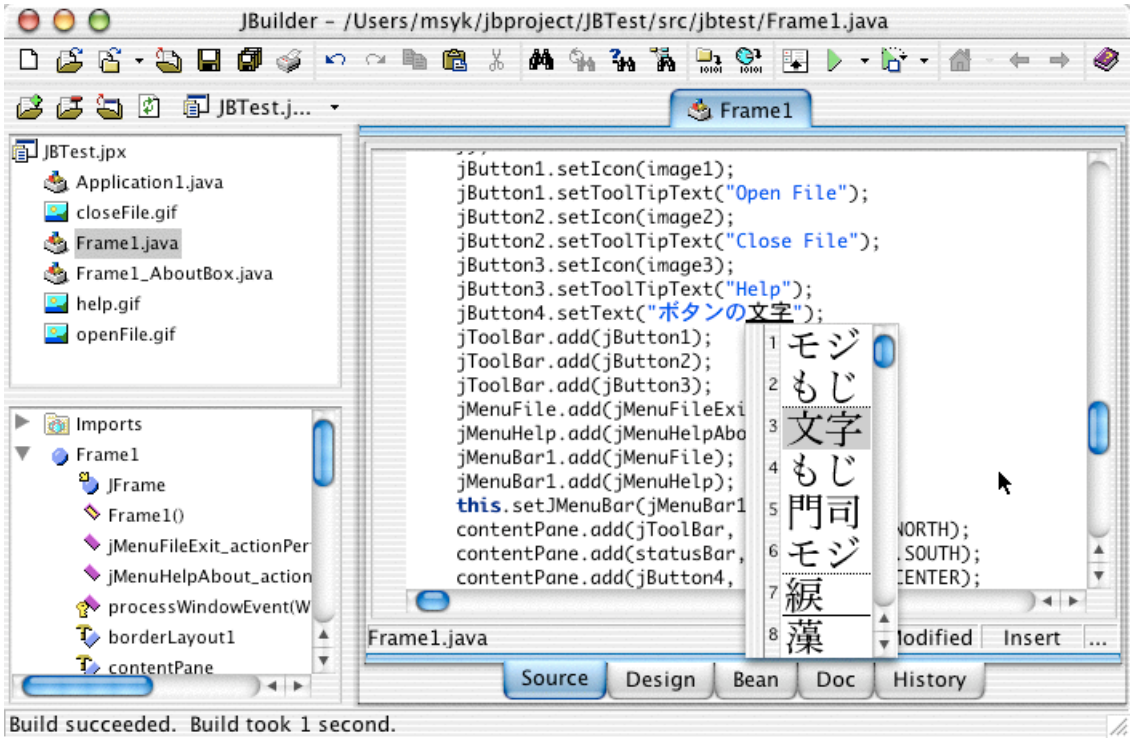
さっそく、JBuilder の GUI 生成機能を使ってみよう。ここではウインドウは Frame1 クラスであるので、そのクラスを開いておき、下側のタブで Design を選択すると、最初は時間がかかるが、GUI のデザイン画面に切り替わる。デザイン機能の使い方もここでは省略するが、ポイントはレイアウトの機能をうまく理解して、結果として生成されるソースをある程度確認しながらオブジェクトとを配置するというあたりだろう。なお、自由にレイアウトしたいなら。とりあえずはレイアウトを null にするという方法もある。ここで、配置したボタンの Text プロパティに日本語文字列を設定してみたのだが、プロパティのリストでも、デザインの画面でも、文字化けが発生した。

デザイン画面では日本語は文字化けする



一方、このまま、Source のタブを選択して、ソース表示にすると、日本語はまったく問題なく利用できる。だから、ボタン名を日本語にしたい場合には、Design では適当な文字を設定しておいて、正しい文字列をソースの側で設定するという方法で問題なく、日本語のダイアログボックスは作成できる。なお、ソースエディタは、Java 1.3.1 Update1 以前だと編集がおかしくなっていたのだが、Update1 により何の問題もなく、きちんと日本語の編集ができるようになっている。

ソースエディタでは日本語は問題なく利用できる



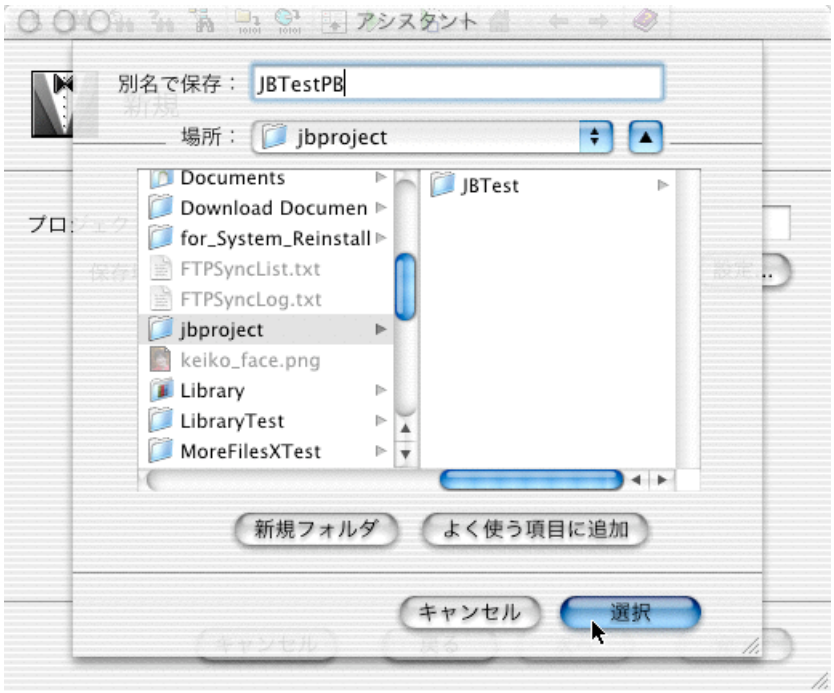
Project Builder のプロジェクトと統合する

次に、JBuilder のプロジェクトで生成された Java のソースを Project Builder で利用できるようにする。Project Builder でプロジェクトを作るとき、ここでは「空白のプロジェクト」を選択してみた。

空白のプロジェクトを用意する

そして、JBTestPB と言う名前でとりあえず、JBuilder のプロジェクトのフォルダと同じ位置にここでは作っておく。Project Builder のプロジェクトは必ずフォルダを作るようなので、作ってから後で Finder を使って移動するなどする。

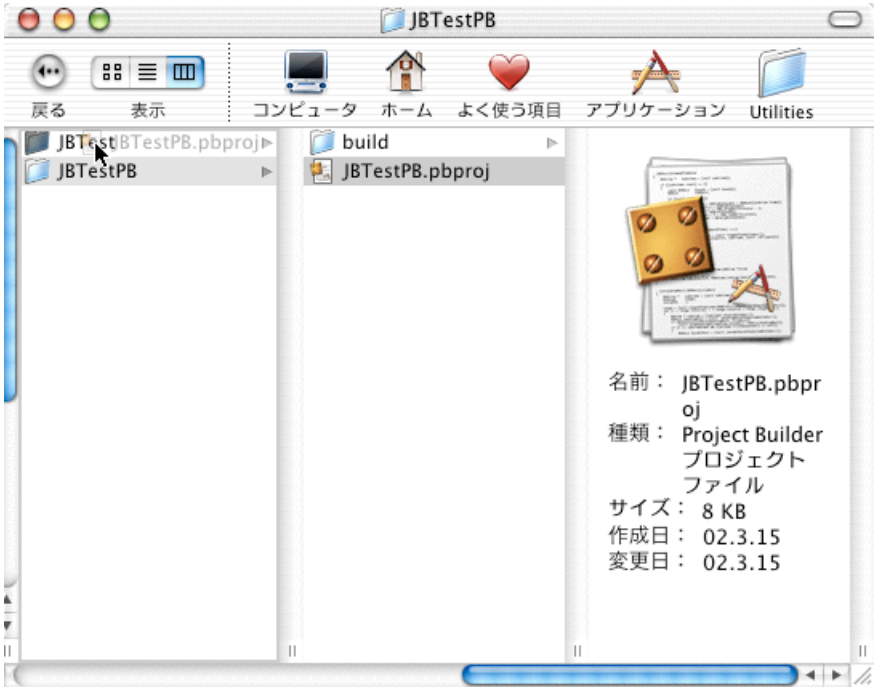
とりあえず適当な名前と場所にプロジェクトを作る



ここでプロジェクトを閉じるか、Project Builder を終了するなどしてから、Finder で作ったプロジェクトのファイルを、JBuilder のプロジェクトのフォルダに移動しておく。

単にドラッグすればいい。ここでは Project Builder のプロジェクトファイルだけが欲しいので、作成されたフォルダは削除しておいてもいいだろう。

プロジェクトのファイルを JBuilder 側のフォルダに移動しておく



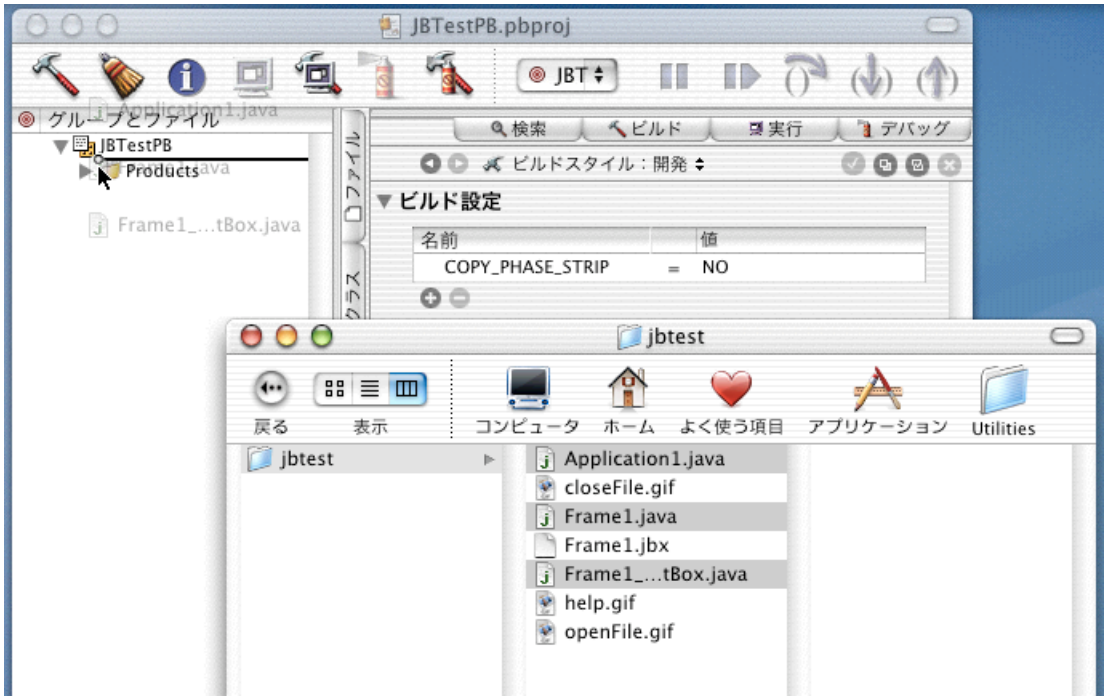
そして、Project Builder のプロジェクトファイルをダブルクリックして、プロジェクトを開く。「プロジェクト」メニューの「新規ターゲット」を選択して、新しくターゲットを作成する。ターゲットの種類は、Java の Application を選べば良いだろう。また、名前ももちろん適当につけておく。

Java のアプリケーションのターゲットを追加する

続いて、JBuilder が生成したソースを、Project Builder 側にも登録する。もちろん、Finder からドラッグ&ドロップする方法でも、メニューから選択しても良いが、いずれにしても、プロジェクトのターゲットに登録しておくようにする。JBuilder では、クラス定義にパッケージを含めるが、パッケージの階層に応じたフォルダ構成を自動的に作る。これは Java の作法だが、Project Builder を使う上ではパッケージをあまり意識しないかもしれない。だが、JBuilder ではパッケージ階層とフォルダの階層が合っていないといけないので、ソースファイルの位置は移動させないでおこう。また、Project Builder でソースを追加するときには気をつける必要があるポイントだ。

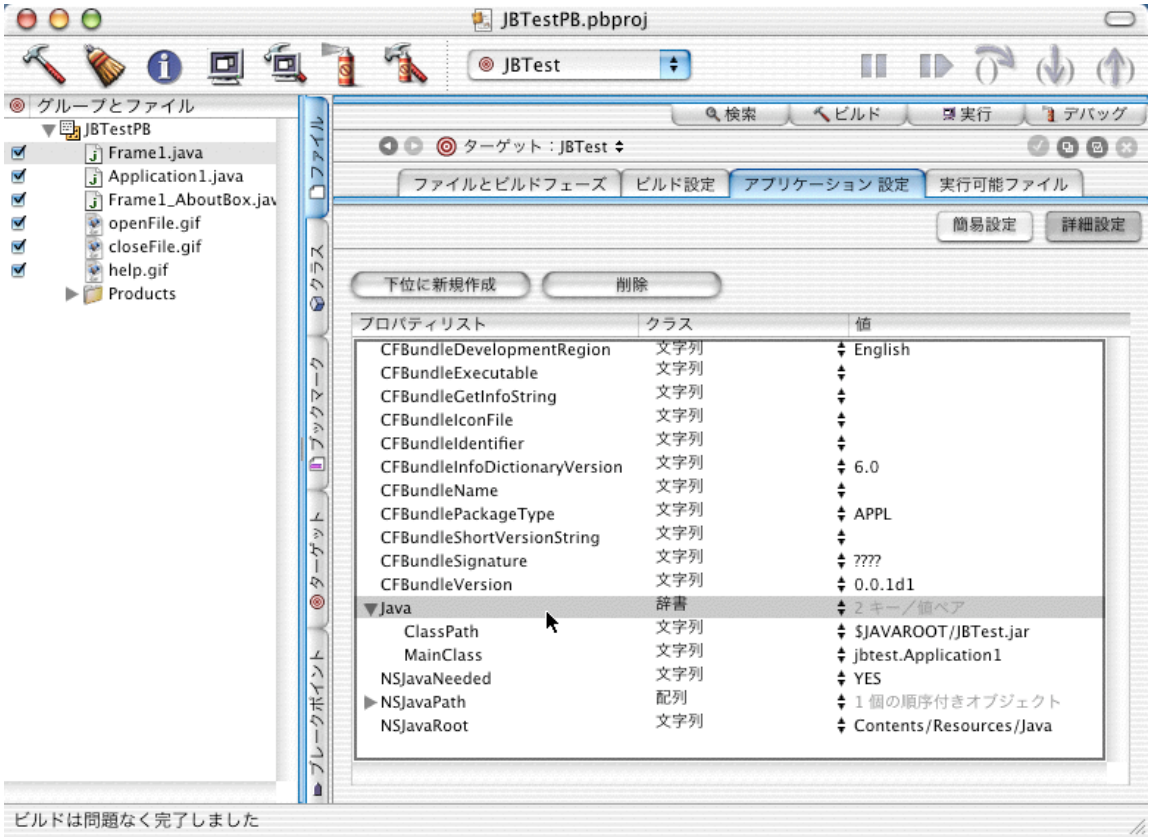
また、GIF ファイルについても基本的にはプロジェクトに登録しておき、生成したパッケージに含めるようにするのが基本だが、これについては後で注釈する。

ソースファイルを Project Builder にも登録する。



Project Builder 側では、ターゲットとして Java の Application を選択した場合でも、いくつかの設定は追加しなければならない。「プロジェクト」メニューの「アクティブターゲットの編集」（Command+option+E）を選択して、「アプリケーション設定」のタブを選択し、「詳細設定」のボタンをクリックする。ここで、Java という辞書クラスのプロパティを作成し、さらに ClassPath と MainClass のプロパティを設定する。設定の方法については、以前に説明した通りであるが、作成される jar ファイルをクラスファイルとして参照すると同時に、メインクラスは、パッケージ名を含めたクラス名を記述しなければならない。

アプリケーション設定を追加する



これでビルドをすると、Mac OS X のアプリケーションパッケージとなったアプリケーションが生成される。

Project Builder での生成されたアプリケーション

まず、未だに Project Builder で生成された Swing のアプリケーションは、日本語文字列はそのままでは文字化けする。CFBunldeDevelopmentRegion を Japanese にするという方法もあるが、Mac でしか使わないのなら、あっさり、フォント設定するのがとりあえずは手軽だろう。ソースでいちいちと思うと面倒だが、JBuilder の Design 機能で、オブジェクトとをまとめて選択して、Osaka フォントに設定すればいいことなので、作業的にも面倒は少ない。

こうして 2 つの開発ツールで同一のソースファイルを編集するとなると、競合の問題が発生してしまう。したがって、JBuilder から Project Builder に移るときには、ソースファイルをいったん閉じる。そして、改めて開く。逆も同様にする。これが基本だ。JBuilder では閉じるキーボードショートカットが Command+W なのに対して、Project Builder は Command+Shift+W となっている。Project Builder で Command+W をするとプ

ロジェクトのウインドウ自体が閉じられてしまう。ちょっとうっとおしいけど、すぐになれるだろう。

JBuilder でアプリケーションを作るときにツールバーを指定すると、アイコンの GIF ファイルを用意する。GIF ファイル自体は、ウインドウのクラスと同じフォルダにあるという仮定でプログラムが組まれている。プログラムをそのまま生かす形にしたいのなら、生成される jar ファイルの中に GIF ファイルをアーカイブしなければならない。たとえば、

```
jar uf build/JBTest.app/Contents/Resources/Java/JBTest.jar -C src jbtest/closeFile.gif
```

のようにして、closeFile.gif という GIF ファイルを、フォルダ階層に応じて jar ファイルにパッケージできる。これを必要なファイルの数だけ実行するのだが、ビルドフェーズでシェルスクリプトを実行するようにすると、良いだろう。

JBuiler で作ったアプリケーションをパッケージとして生成したい場合では、以上の要にうまう使い分ければそれほど難しいことではないだろう。

ᔓᔓᔓᔓᔓᔓᔓこの項、以上ᔓᔓᔓᔓᔓᔓᔓ[新居雅行]ᔓᔓᔓᔓᔓᔓᔓ