今から始める Cocoa プログラミング》
文書ファイルを扱うアプリケーションを作る(補足)
NSBundle

言語ごとに異なる文字列を得るには、NSBundle の機能を使うのがいちばん手軽だろう。言語リソースの変換テーブルを作成しておき、そこで、英語のメッセージとその言語のメッセージの対応表を作っておく。そうすれば、実行時に選択されている言語に合わせた文字列を取得することができる。

NSBundle と言語リソース

前回の記事では、nib ファイルを言語ごとに用意するということを説明した。ウインドウのボタン名など、nib ファイルで定義されるものについては、この方法で十分に国際化対応が可能になる。ただし、こうした文字列は、nib ファイルに定義しているものだけでなく、ソースの中にもでてくる。もちろん、それらをテーブルとして管理して、言語に応じて…というのは簡単だが、なかなか後手後手になって、最後に苦労するということもあるかもしれない。しかしながら、NSBundle の機能を使えば、やや語弊があるかもしれないが、けっこう気楽に文字列の多言語対応ができてしまうのである。

NSBundle は、もちろんバンドルを扱うためのクラスだ。バンドルとは、実体はフォルダであるが、フォルダの中のものを一括してまとめて扱うための仕組みの一般的な名称である。アプリケーションやフレームワークはバンドルの一種である。アプリケーションの中は一定の決められたフォルダ構成となっている。こうした取り決めを、バンドルを扱うクラスで隠ぺいすることで、具体的な相対パスを知らなくても決められたメソッドを利用することで、バンドル内のさまざまなリソースにアクセスできるというわけである。

文字列の多言語対応については、その変換テーブルを言語リソースに置くのが基本だ。たとえば、日本語だと、パッケージの中の Resources フォルダにある Japanese.lproj フォルダに置く。だが、NSBundle では、現在選択されている言語に応じて言語リソースを探し、優先順位に従って適切な言語リソースフォルダのテーブルファイルを参照するといったことを背後で自動的に行ってくれる。だから、プログラムの中ではファイルのパスは書かなくてもいいということである。あとは、決められた名前で決められた位置に、テーブルとなるファイルを置いておくだけでいいのである。

NSBundle の文字列変換機能

まず、NSBundle にあるメソッドを紹介しておこう。NSBundle にはいろいろな機能があるが、ここでは、文字列の多言語対応にかかわるもののうち、便利な機能にしぼって紹介する。いくつかのメソッドがあるが、通常は以下の 1 つのメソッドを使うだけで事は足りると思われる。動作については、表だけでは説明しきれないので、その後の本文を参照していただきたい。

指定した文字列に対し、変換テーブルに従って現在の言語に応じた文字列を得る（Static）		
String NSBundle.localizedString(String key) String NSBundle.localizedString(String key, String comment) String NSBundle.localizedString(String key, String tableName, String comment)		
戻り値	規則に従って変換された文字列	
引数	key	もとになる文字列
	comment	コメント
	tableName	テーブル名（テーブルとなるファイルのファイル名）。指定がない場合には、Localizable が指定されたものとみなす
備考	メソッド localizedString として、この 3 つのメソッド以外に、異なるバンドルにあるテーブルを利用するものも定義されている	

まず、プログラム中の文字列は、ダブルクォーテーションで囲った “String” といったものが代表的なリテラル表記である。その、文字列が、言語に応じてバリエーションを持たせる必要がある。したがって、1 つの文字列に対して、複数の言語ごとの表現が存在するということになる。

こうした対応を取るために、たとえば、ある文字列が「msg1」だというキーワードで与えて、それを元に、言語ごとの文字列を “Processing” “処理中” などと表示するということは 1 つの方法だろう。このように、キーと値を対応付けるというやり方は、Cocoa でもよく見られる手法だ。ローカライズ文字列の処理はそれが基本とはなっている。しかしながら、もう少し簡便な方法を取っている。

まず、最初に、英語の表現を基本として、英語の文字列を決める。たとえば、”Processing” だったとしよう。英語が言語として選択されている場合には、その文字列を使う。一方、日本語が言語として選択されているのであれば、Processing をキーにして、「テーブル」を参照し、そこから “処理中” という文字列を得るという処理を行うことにする。つま

り、英語の場合の文字列がキーとなって、テーブルからの表引きを行うというようにしたいということである。
ここで、テーブルというのは、実際にはファイルである。基本的には次のような規則での作成となる。

- テーブルの中身はテキストファイルである。
- 各行に、キー = 値; の形式で、対応表を作る。
- キーや値は、ダブルクォーテーションで囲う。
- /* */でのコメントも記載できる。
- テーブルは、言語ごとに作成され、アプリケーションパッケージの Resouces フォルダの中の言語リソースのフォルダ（日本語なら、Japanese.lproj フォルダ）に存在する必要がある。
- テーブルファイルのファイル名には、.strings が拡張子としてつけられる。
- .strings を含まないファイル名を「テーブル名」として参照する。

こうして、テキストファイルでテーブルを用意しておけば、実行時に、テーブルに応じて変換された文字列が、プログラムの中で得られるという具合だ。ただ、これをさらにやりやすいような仕組みが用意されている。

ローカライズ文字列の実際

こうしたテーブルのひな形を作成してくれるコマンドがあり、それを利用するのが手軽である。一方、作成されたアプリケーションのパッケージに、テーブルとなるファイルがなければならないので、事実上はそのファイルは Project Builder に登録しておかなければならない。こうした一連の作業を行なうには、手順的には次のようになるだろう。
たとえば、ある変数に文字列を入力している以下のような部分があるとする。

```
String s = “Please Input”;
```

まず、この部分を、次のように、ローカライズ文字列対応のメソッドを通すようにして得られるようにする。

```
String s = NSBundle. localizedString (“Please Input”, “Comment1”);
```

このように、NSBundle のクラスメソッドで localizedString メソッドを通すようにしておくのである。2 つ目の引数は適当でかまわないが、後で、テーブルのファイルのどこにでてくるかを確認してもらいたい。
続いて、Terminal で以下のようにコマンドを実行する。ここで、カレントディレクトリは、プロジェクトの存在するディレクトリだとする。上記のステートメントがソースファイルは、プロジェクトのフォルダのルートにあるとする。そこで、ソースをもとに、英語を日本語に変換するテーブルをファイルのひな形を生成する。

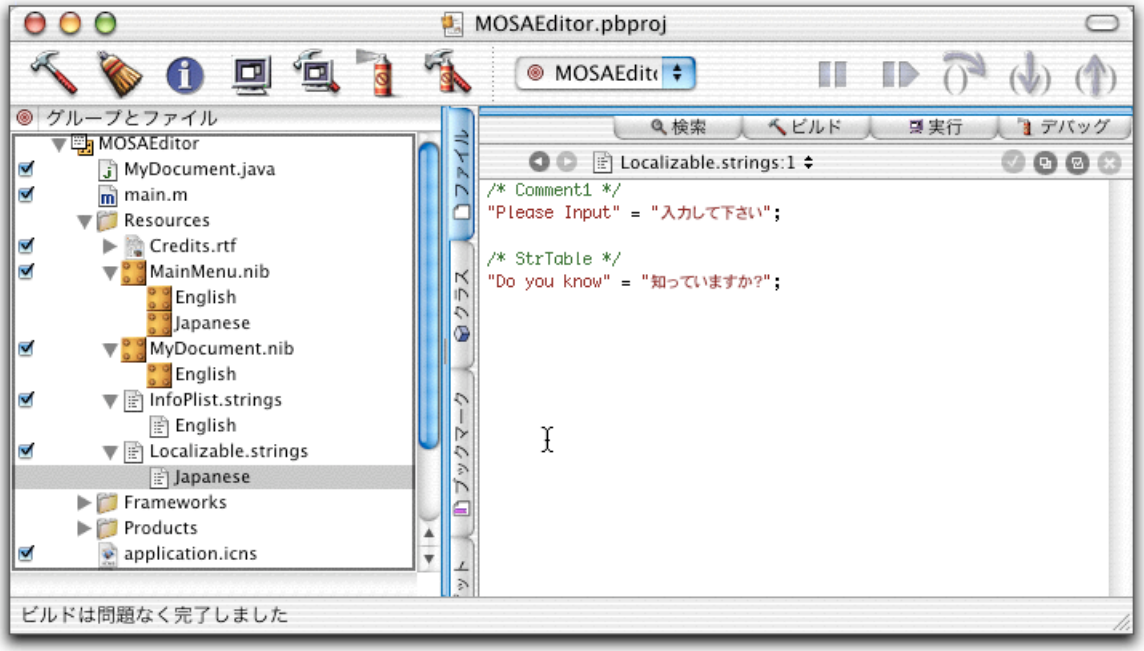
```
% genstrings –j –o Japanese.lproj/Localizable.strings PreferencesWindow.java
```

ここで、genstrings というコマンドを使うのだが、このコマンドにはマニュアルがない。そこで、単にこのコマンド名だけを入力すれば、ヘルプ表示がされ、もちろん参考にはなるが、間違いもあるようで、気をつける必要がある。このコマンドは、本来は Objective-C 向けであるが、-j オプションをつけることで、Java のソースにも適用きる。そして、引数に指定したソースファイルから、localizedString 等のメソッドをさがして、その文字列の対応表を、-o オプションで指定したファイルに書き出すのである。いずれにしても、以上の操作で、プログラムのソースファイルから localizedString メソッドを使っている部分を探し出し、文字列変換のテーブルを行うための日本語リソースとしての Localizable.strings ファイルが出来上がった。
続いて、こうして作成されたファイルを、Project Builder で、プロジェクトに登録する。「プロジェクト」メニューから「ファイルを追加」（Command+otption+A）を選択する。そして、genstrings コマンドで生成したテーブルファイルを選択する。

Localizable.strings を選択する

すると、プロジェクトに Localizable.strings ファイルが加わるが、この項目は、言語対応となっており、その下位の項目に、Japanese が含まれている。これをクリックして、右側でそのファイルの内容を修正すればよい。最初は「"Please Input" = "Please Input";」のようになっているが、そのイコールの右側を日本語に書き換えればよい。

Localizable.strings の日本語対応テーブルを編集した



なお、文字列の書き換え作業は、エディタで行ってもいいのであるが、このファイルはいずれにしても、プロジェクトに登録しておく必要があるので、Project Builder で編集

作業をするのが手軽だろう。日本語はシフト JIS コードでかまわないようである。

ここで、もとの localizedString メソッドの 2 つ目の引数は、テーブルファイルの中で、コメントとして見えているのがわかる。だから、テーブルを後で編集しやすくするために、適当なコメントを作っておけばいいということである。

こうしておけば、さきほどのソースの Stiring クラスの文字列 s では、英語だと「Please Input」、日本語だと「入力して下さい」という文字列になるのである。LocalizedString メソッドでテーブル名を指定しない場合には、自動的に、Localizable.strings ファイルから参照されるということになっている。なお、英語の場合は、English.lproj フォルダにある Localizable.strings ファイルを参照するはずであるが、そのファイルが存在しないので、もともとの「Please Input」という文字列のままになるということである。日本語だと、Localizable.strings が存在するので、それに従って変換作業が行なわれるということである。

Localizable.strings 以外のテーブルを使用する

なお、NSBundle の localizedString メソッドでテーブル名を指定するメソッドもある。ただし、このメソッドは、genstrings コマンドではエラーとなるので、このコマンドのサポートは受けられない。ただし、同一言語でさらに状況に応じてメッセージを切り替えたいときには、複数のテーブルファイルを使うことも必要だろう。このとき、

```
String s = NSBundle. localizedString (“Please Input”, “MyTable”, “Comment1”);
```

としておくと、たとえば日本語なら、Japanese.lproj フォルダにある MyTable.strings というテーブルファイルを参照して置き換えを行うというように動作をする。従って、プロジェクトの中の Japanase.lproj フォルダに MyTable.strings ファイルを作っておき、それをプロジェクトに登録をしておかなければならないということになる。

（この項、続く）