第1章　Servletの作成

1.1　Webサーバサイドプログラムとは

Webサーバサイドプログラムとは、WWWのサーバ側で実行されて動的にHTMLなどのコンテンツを生成するプログラムのことである。このなかでポピュラーなCGI（**C**ommon **G**ateway **I**nterface）は、Webサーバ上でプログラムを実行し、動的にHTML形式などのデータなどを作成してWebブラウザに渡すための仕組み（プログラムとWebサーバの間のデータのやりとりの約束事）である。またCGIに従って実行されるプログラム自体のこともCGIと呼ばれる。CGIを記述する言語は何でも構わないが、PerlやCを使うことが比較的多いようである。

Java AppletやJavaScriptはクライアント（Webブラウザが実行されているコンピュータ）側でプログラムが実行されるのに対し、CGIを含むサーバサイドプログラムはサーバ（HTMLファイルが置かれているコンピュータ）側でプログラムが実行されるという違いがある。例えばアクセスカウンタ・掲示板・オンラインショッピングサイトなどはクライアント上のプログラムだけでは実現できないのでサーバサイドプログラムが必要になる。

1.2　Servletとは

この演習ではサーバサイドプログラムの作成にJava Servletを用いる。Servletとは、CGIと同じようにWebサーバ側でプログラムを実行するための仕組み（約束事）である。しかしCGIとはいくつかの点で区別される[1]。

- まず、約束事はJavaのクラス/インターフェースとして提供されるので、プログラミング言語は当然Javaに限定される。
- 呼出し毎にいちいちプロセスを生成せず、スレッドとして実行するので効率が良い。
- ある程度の期間、サーバ側で接続の情報を記憶しておくことができるなど、サーバサイドプログラミングを支援するためのライブラリが充実している。

このためJavaによるサーバサイドのプログラミングとしては、CGIではなくServletを使用することが多い。例えばオンラインショッピングのためのWebサイトなどはServletが得意とする分野である。

本演習ではServletを実行するためのWebアプリケーションサーバとして、Apacheの Jakarta Tomcatを使用する。Webアプリケーションサーバは、コンテナとも呼ばれ、Webブラウザ（あるいはApacheなどのWebサーバ）からの要求を受け付け、Servlet（やJSP）を起動するプログラムである。

なお、TomcatやJavaの開発環境（JDK, Eclipse）などのインストール方法は別ドキュメントで解説する。

[1]ただし、Webサーバサイドプログラムの仕組みとしてCGIがもっとも代表的なので、 Servletを含めたWebサーバサイドプログラムの総称としてCGIという言葉を用いることもある。

有用なリンク

HTML のまとめ

- 初めての ホームページ講座 （http://www.hajimeteno.ne.jp/）

Java 関連

- Java Tips（http://www.asahi-net.or.jp/~dp8t-asm/java/tips/）
- Apache Tomcat （http://tomcat.apache.org/）

1.3 必要な Java の知識

本演習では次のような Java のごく初歩的な知識だけを仮定する。

- 制御構造（if〜else 文、for 文、while 文）の書き方がわかること。（C 言語の制御構文と同じ。）
- 継承（class〜extends）を利用してクラスを定義できること。
- メソッドの定義の書き方がわかること。（C 言語の関数定義とほとんど同じ。）
- クラス・オブジェクトの概念を理解していること。つまり、
  - （. 演算子を使って）フィールド参照・メソッド呼出しができること。
  - オブジェクトの生成（new）ができること。
  - クラス変数・クラスメソッドが使用できること。
- import 文が書けること。（C の#include に似ている。）

言い替えれば、if, else, for, while, class, extends, .（ドット）, new, import などのキーワード・演算子の使い方を理解していればよい。
あとは Java の API 仕様のドキュメント:

- http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/
  Java 2 Platform Standard Edition （Java の標準 API）— 以降、上記を単に ***(J2SEAPI)*** と記す。
- http://java.sun.com/j2ee/1.4/docs/api/
  Java 2 Platform Enterprise Edition （Servlet 関連の API）

などで必要に応じてメソッドの使い方などを調べる必要がある。

1.4 Servlet の作成

CGI も Servlet も、単純に言ってしまえば、HTML のデータ[2]を生成するプログラムである。
次に示すのは現在の時刻を表示する Servlet である。

[2]JPEG や PNG など HTML 以外のデータを出力する CGI や Servlet もあるが、はじめは簡単のために HTML を生成するプログラムのみ扱う。

ファイル MyDate.java

```
import java.io.*;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyDate extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request, HttpServletResrponse response)
              throws IOException {
    response.setContentType("text/html; charset=Windows-31J");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body>");

    Date d = new Date();
    out.println(d.toString());

    out.println("</body></html>");
    out.close();
  }
}
```

Servlet は HttpServlet というクラスを継承して作成する。このクラスに Servlet として必要なほとんどの機能が実装されているので、必要なところのみ書き換えれば Servlet が実行できるようになっている。

```
import javax.servlet.*;
import javax.servlet.http.*;
```

の import 文は大抵の Servlet で必要で、javax.servlet および javax.servlet.http というパッケージに属するクラスを利用するために必要である。

Servlet の処理は基本的に doGet（または doPost — 後述）というメソッドの中に記述する。上のメソッド定義の最初の部分:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException
```

からわかるように、doGet/doPost は HttpServletRequest 型、HttpServletResponse 型の 2 つの引数を取る。最後の throws IOException の部分は、この doGet というメソッドが、IOException という例外を発生するかも知れないということを宣言する Java の構文である。

このメソッドの最初の文の

```
response.setContentType("text/html; charset=Windows-31J");
```

は、以下に続くデータが HTML のデータで文字コードが Windows-31J であるということをブラウザに伝える役割を持つ。

また、

```
PrintWriter out = response.getWriter();
```

は、ブラウザにデータを送るための**出力ストリーム**を取得する。これ以降 out オブジェクトの println（あるいは print）メソッドを呼び出すことにより、データを出力することができる。

なお出力の最後に out.close() を呼び出してストリームを閉じておく。

問 1.4.1 上の *Servlet* プログラムで現在の秒によって、ブラウザに表示されるときの色が変わるようにせよ。例えば、*0*〜*19* 秒が黒、*20*〜*39* 秒が青、*40*〜*59* 秒が赤など。

ヒント:

- 現在の “秒” を求めるためには *Calendar.getInstance().get(Calendar.SECOND)* という式を用いる。*((J2SEAPI)/java/util/Calendar.html)*
- 色を変えるには *HTML* のタグ * ... *などを用いる。
  （*HTML* の規格では、上の *red* の周りの引用符は上のように一重引用符「'」でも二重引用符「"」でも良い。*Java*（*C* でも同じ）のプログラムで、二重引用符「"」自体を出力したいときは、*out.println("<font color=¥"red¥">")* のように、「"」の前に バックスラッシュ「¥」をつける必要がある。

問 1.4.2 現在の秒によって、ブラウザに表示されるページの背景画像が変わるようにせよ。

ヒント:

- 背景画像を変えるには *HTML* のタグ *<body background=' ... '> ... </body>*などを使用する。
- 素材: *http://www.3776m.com/sozai/*（素材の館）
  http://www.ushikai.com/（牛飼いとアイコンの部屋）

1.5 （参考）Servlet の設置

本演習では Servlet のコンパイルと設置には Eclipse の WTP プラグインを使用する。このプラグインの使用法は別のドキュメントで説明する。以下では、Eclipse を使用しないで手作業でコンパイル・設置する方法を、概略だけ述べる。

Servlet を実行するには、まずコンパイルが必要である。次のコマンドで .class ファイルを作成する。

```
javac -classpath servlet-api.jar MyDate.java
```

“servlet-api.jar” の部分は、実際には ServletAPI が含まれている JAR ファイル（Java のライブラリファイル）へのパスに置き換える。これは通常、***(TOMCAT)***/common/lib/servlet-api.jar[3]となる。

Servlet を実際に設置するには、生成されたクラスファイル（ソースファイルの名前が MyDate.java の場合、MyDate.class）を、Servlet の仕様で定められたディレクトリ構成:

```
- （Web アプリケーションルート）
    - WEB-INF
        - web.xml （設定ファイル）
        - classes
            - （class ファイル）
        - lib
            - （JAR ファイル）
```

の classes というディレクトリの下に置き、さらに、web.xml という設定ファイルにパスを記述する必要がある。

web.xml の設定例:

[3] ***(TOMCAT)*** は Tomcat をインストールしたディレクトリのことを指す。 これは Tomcat のバージョンにより異なる。また、パスの区切りは Windows では通常 “¥” だが、このプリントでは一般的な “/” を使用する。

```
<web-app>
  ...
  <servlet>
    <servlet-name>MyDate</servlet-name>
    <servlet-class>MyDate</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyDate</servlet-name>
    <url-pattern>/MyDate</url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

この例は、MyDate クラスに MyDate というサーブレットの名前をつけ、さらに MyDate という名前のサーブレットを/MyDate という URL のパスでアクセスできるようにするための設定である。これは冗長に見えるが、同じクラスを別の名前のサーブレットとして起動することができるようになっているためである。

ただし、開発中にはいちいち web.xml ファイルを書き換えるのは面倒なので、クラスファイルを classes フォルダに置くだけで Servlet が実行できるように、invoker サーブレットというものを有効にすることもできる。これは Tomcat 全体の設定ファイルの web.xml に設定することで有効化できる。***(TOMCAT)***/conf/web.xml のなかで、次のようになっている部分を探して、下線の部分を付け加える。

```
<!-- -->
    <servlet>
        <servlet-name>invoker</servlet-name>
        <servlet-class>
          org.apache.catalina.servlets.InvokerServlet
        </servlet-class>
        <init-param>
            <param-name>debug</param-name>
            <param-value>0</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
    </servlet>
<!-- -->
```

```
<!-- -->
    <servlet-mapping>
        <servlet-name>invoker</servlet-name>
        <url-pattern>/servlet/*</url-pattern>
    </servlet-mapping>
<!-- -->
```

こうしておくと、例えば/servlet/MyDate というパスで、MyDate クラスにアクセスすることができる。

> **重要:** invoker サーブレットはセキュリティ上問題となることがあるので、開発時以外は無効にしておく。

Web アプリケーションルートは任意の場所に置くことができるが、Tomcat の設定ファイルの server.xml（***(TOMCAT)***/conf/server.xml）にその場所を記述する必要がある。

```
...
<Server ... >
 ...
  <Service ... >
 ...
    <Engine ... >
      ...
      <Host ... >
        ...
        <Context path="/SoftEngEnshu" reloadable="true"
                 docBase="C:¥somewhere¥SoftEngEnshu" />
        ...
      </Host>
    </Engine>
  </Service>
</Server>
```

この例では、C:¥somewhere¥SoftEngEnshu というフォルダをルートフォルダとする Web アプリケーションが SoftEngEnshu というパスでアクセスできることになる。さきほどの web.xml の設定例とあわせると、http://*hostname*:8080/SoftEngEnshu/MyDate（invoker サーブレットを使っている場合は、http://*hostname*:8080/SoftEngEnshu/servlet/MyDate）という URL で MyDate サーブレットの実行結果を見ることができる。*hostname* の部分は Tomcat を実行しているホストの名前または IP アドレスである。

> Servlet の開発中はサーブレットと WWW ブラウザは同一のコンピュータで実行していることが多いので、その場合は *hostname* は localhost（あるいは 127.0.0.1）となる。

キーワード:

CGI, Perl, Servlet, Tomcat, Web アプリケーション, HttpServlet クラス, doGet メソッド, throws