高度情報処理演習B

Java Servlet/Java Server Pages

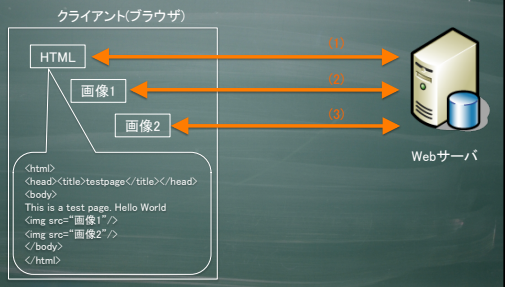
Servlet/JSP

- Servlet?
  - Javaでウェブアプリケーションをつくるための仕組み
  - Javaのクラスで作成する
  - 最新 – Servlet 3.0 (2011/8)
- JSP?
  - HTMLの中にJavaでロジックを組み込む
  - 初回アクセス時にServletにコンパイルされる
  - 最新 – JSP 2.2 (2011/8)

HTTPの復習

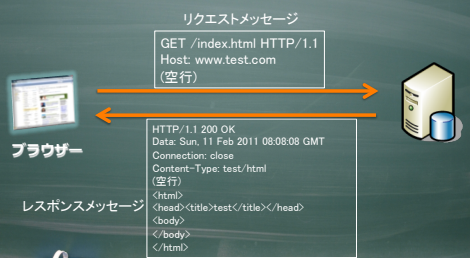
- HTTPの特徴
  - 少ない命令から構成され，シンプル
  - 複雑なネゴシエーションを必要とせず，処理のオーバヘッドが少ない
  - ステートレスな構造
- HTTPによる通信
  - クライアントからサーバへのリクエスト
  - サーバからクライアントへのレスポンス
  - バイナリではなく簡単なテキスト(メッセージ)

ブラウザとサーバのやり取り



HTTPメッセージの基本

- 簡単なリクエスト/レスポンスの例



リクエストライン

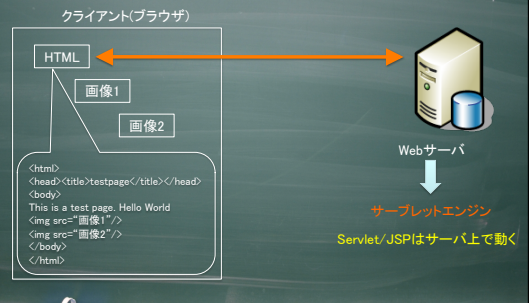
メソッド　リクエストURI　HTTPバージョン

e.x) GET /index.html HTTP/1.1

代表的なメソッド

メソッド	機能
GET	指定したURLが示すリソースを取得する
POST	指定したURLにデータを転送する．リクエストのボディには転送するデータが含まれる
HEAD	指定したURLが示すリソースを取得するときのヘッダだけを取得する
PUT	指定したURLが示すリソースに対して，データを転送して置き換える
DELETE	指定したURLが示すリソースを削除する

Servlet/JSPに必要な環境



サーブレットエンジン

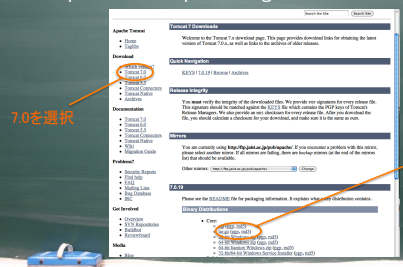
- Tomcat – Apache Software Foundation
  - デファクトスタンダード
- Resin – CAUCHO
- WebShere – IBM
- WebLogic – Oracle
- Jetty – Eclipse
  - お薦めだが．．．

目標

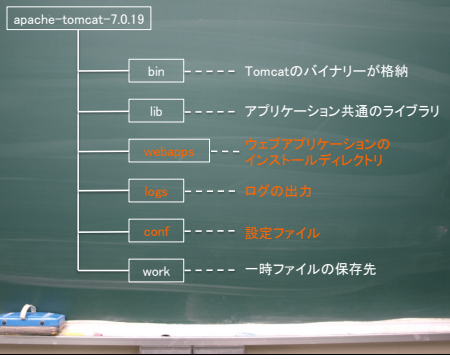
- Linux環境でTomcatを動かす
  - 環境変数の設定
    - CATALINA_HOME, JAVA_HOME…
  - 起動/終了の手順
- HelloWorldサーブレットを動かす
  - サーブレットをコンパイルして適切に配置
  - J2EEのディレクトリ構造を知る
  - サーブレットの設定をする

ダウンロード

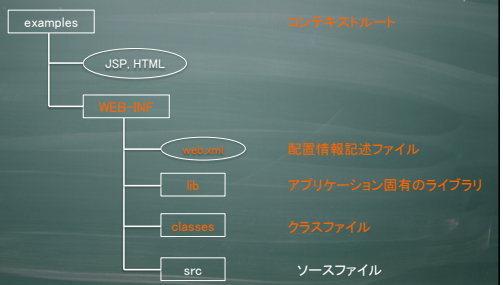
- Tomcatのダウンロード
  - http://tomcat.apache.org/



展開後のディレクトリ構成



ウェブアプリケーションの配置 – J2EEの設定

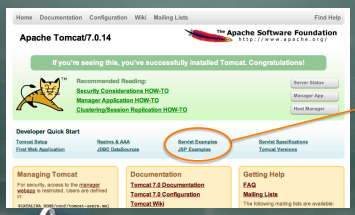


Tomcatを動かすための設定

- CATALINA_HOME
  - Tomcatが展開されたディレクトリ
    - apache-tomcat-7.0.19
- JAVA_HOME
  - JDKのインストール先
    - /usr/java/jdk1.6
- いずれも環境変数で設定
  - .cshrc, .bashrcなど

Tomcatの起動/停止/確認

- 起動 – ${CATALINA_HOME}/bin/startup.sh
- 停止 – ${CATALINA_HOME}/bin/shutdown.sh
- 確認 – http://localhost:8080/ にアクセス



エラーが出る場合

1. ${CATALINA_HOME}/logs/catalina.outを確認
2. ${CATALINA_HOME}/bin/shutdown.sh
3. 問題の修正
4. ${CATALINA_HOME}/bin/startup.sh
5. http://localhost:8080/　で確認
6. 1～5の繰り返し

HelloWorldサーブレットを動かす

- 手順
  1. ソースコードの作成
  2. コンパイル
  3. アプリケーションとして配置
  4. サーブレットの設定
  5. 確認

1. ソースコードの作成

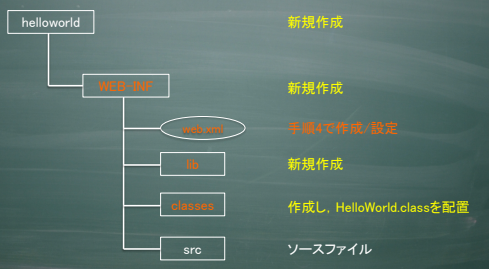
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloWorldServlet extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                        throws IOException, ServletException {
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Hello World</title>");
                out.println("<body>");
                out.println("Hello World");
                out.println("</body>");
                out.println("</head>");
                out.println("</html>");
        }
}
```

2. コンパイル

1. ${CATALINA_HOME}/lib/servlet-api.jarにクラスパスを通す
2. % javac HelloWorld.java
   - HelloWorld.classが作成される

3. アプリケーションとして配置



4. サーブレットの配置 – web.xmlの設定

1. Examples/WEB-INF/web.xmlをコピー
2. 不要な部分を削除
   - 最低限必要なタグ
     - servlet
     - servlet-mapping
   - それ以外は削除
3. WEB-INF以下に配置

web.xmlの例

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

  <!-- Define servlets that are included in the example application -->

  <servlet>
    <servlet-name>HelloWorld</servlet-name>
    <servlet-class>HelloWorldServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
  </servlet-mapping>
</web-app>
```

サーブレット名とクラスの定義

classesに配置されたクラス
パッケージも含む

URLとサーブレットのマッピング

確認

- http://localhost:8080/helloworld/HelloWorld
  （helloworld：フォルダ名，HelloWorld：サーブレット名）
- エラーの場合
  - 手順1～4までをよく確認
  - どこかで間違っているはず

サーブレットの基本

- 主要パッケージ
  - javax.servlet
  - javax.servlet.http
  - 詳しくはjavadoc参照
- サーブレットの作成
  - HttpServletを継承
  - 必要なメソッドをオーバライド
    - doGet/doPostなど

HttpServletクラスのメソッド

HTTPのメソッド	メソッド
GET	proteced void doGet
POST	protected void doPost
PUT	protected void doPut
DELETE	protected void doDelete
HEAD	protected void doHead
TRACE	protected void doTrace
OPTIONS	protected void doOptions

引数はいづれも，HttpServletRequest req, HttpServletResponse res

メソッドの処理 – doGetの場合

オーバライド

```
public class HelloWorldServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
                                                    throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Hello World</title>");
        out.println("<body>");
        out.println("Hello World");
        out.println("</body>");
        out.println("</head>");
        out.println("</html>");
    }
}
```

Content-Typeの指定

HTMLの出力

HttpServletRequest – リクエストの取得

- (ブラウザから)サーブレットに送られてくる情報を取得
- 主要メソッド
  - String getParameter(String name)
    - 名前を指定して値を取得
  - String[] getParameterValues(String name)
    - 名前を指定して値を複数取得
      - 同一の名前で複数の値が送られてくる場合
  - String getHeader(String name)
    - 名前を指定してヘッダの値を取得
  - その他詳しくはjavadoc

HttpServletResponse – クライアントへの出力

- クライアント(ブラウザ)に大してHTMLやヘッダを出力
- 主要メソッド
  - void setContentType(String type)
    - コンテントタイプを指定
  - PrintWriter getWriter() throws IOException
    - 出力ストリームであるWriterを取得
  - void setRedirect(String location)
    - クライアントをリダイレクトする
  - void setStatus(int sc)/void setHeader(String name, String value)
    - ステータスコードを設定/ヘッダを設定

セッション管理 – HttpSession

- セッションの開始
  - HttpSession ses = request.getSession(true/false)
    - true : セッションが開始されていなければ作成して返す
    - false : セッションが開始されていなければnullを返す
- 値の設定と取得
  - void setAttribute(String key, Object value)
  - Object getAttribute(String key)
- 値の削除
  - void removeAttribute(String key)

Servletで3画面構成の会員登録システムを作る

- RegistServlet
  - 名前と住所の入力を受けつけ，登録ボタンのクリックで確認画面に遷移
- ConfirmServlet
  - 入力された名前と住所を表示し，完了ボタンのクリックで完了画面へ遷移
- CompleteServlet
  - 完了画面を表示．入力情報は保存する必要なし