メディアプロジェクト演習1
補足資料

CGIの基本

CGIとJava Servletの違い

Java Servletの基本

インタラクティブなWebSiteとは

- Interactive
  - 「対話」または「双方向」
  - クライアントとシステムが画面を通して対話を行う形式で操作を行っていく仕組み

- 利用用途

  Website, シミュレーションシステム, ゲームなど

  「WWW =インタラクティブなメディア」

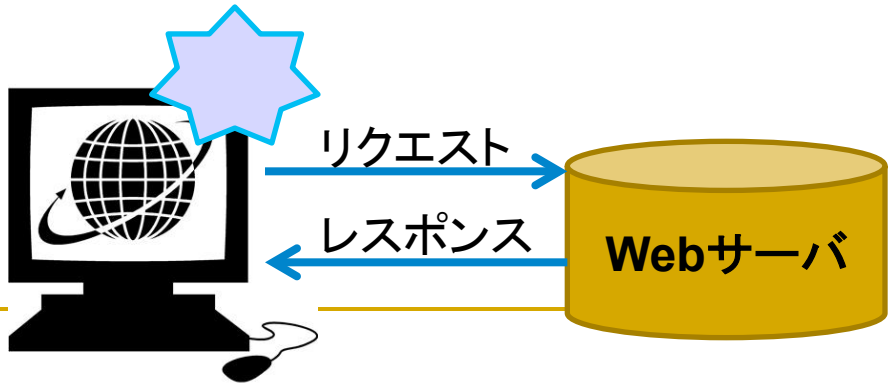
  テレビやラジオと異なり、ユーザの側から積極的に情報にアクセスする（w3c.orgより）

- WebSiteなどを介したシステム

  JavaScript, CGI, Java servlet　など・・・

JavaScriptとは

- インタプリタ型のスクリプト言語としてWebページ上で動作
  - HTML文書は静的なWebページを表示するのみ
  - JavaScriptを用いることで、動的な情報表示が可能
  - C言語と文法が少し似てる
  - あくまでも、クライアントサイドで動作（HTMLファイルへの埋め込み）
  - Javaとは全く違うもの！
- 利用用途
  - クライアント側で計算やページの動的操作
  - 目的に応じた様々なアクションなど
- 動作場所
  クライアントの環境下

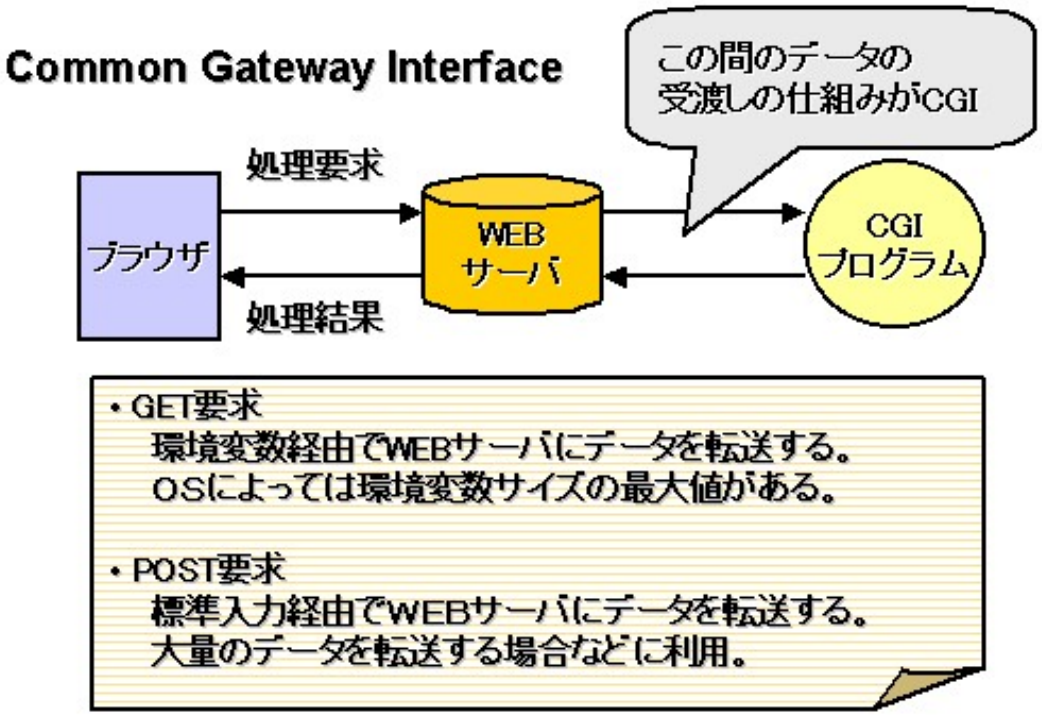


CGIとは

- Common Gateway Interface:CGI
  ブラウザから送信されたデータをWebサーバで受け、
  受け取ったデータを処理するアプリケーションに渡す仕組み

- 利用用途
  アンケートや掲示板など

- データ転送方法
  GETとPOST



データの転送方法(GET，POSTの違い)

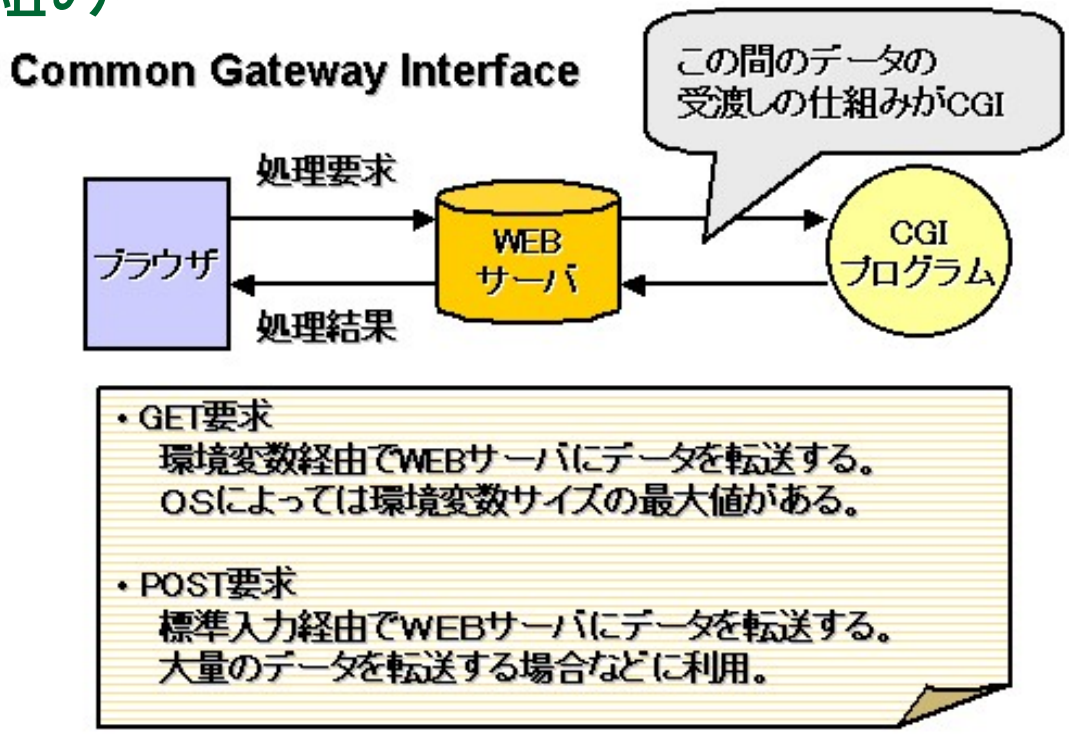
- GETとPOSTによるデータ転送
  - Webシステムを介した情報通信のメソッド
  - HTTPにおけるリクエストの形式

- GET
  - 送信するデータを引数として連結し，取り扱う
  - 人目に送信情報が表れる　＆＆　大量のデータ送信には不向き
  - CGIを指定するURLの末尾に「?」を加え、その後にURLエンコードしたデータを追加する

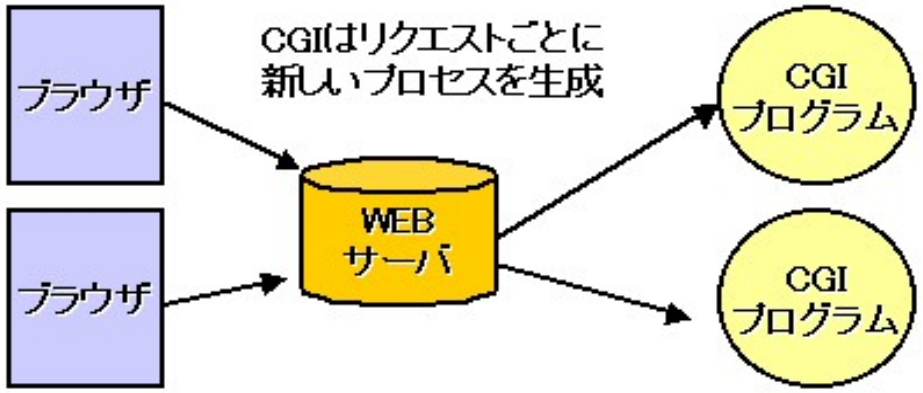
    例：http://www.hogenet.jp/cgi-bin/faq/faq.cgi?year=2011&month=6

    ここで，「?」の後ろについた「year=2011」と「month=6」が引数で，複数ある場合は「＆」で連結
- POST
  - **フォームを用いて，パラメータをつけて情報要求を行う.**
  - **大量のデータ送信に向いており，送信情報は人目にあらわれない**

CGIの仕組み

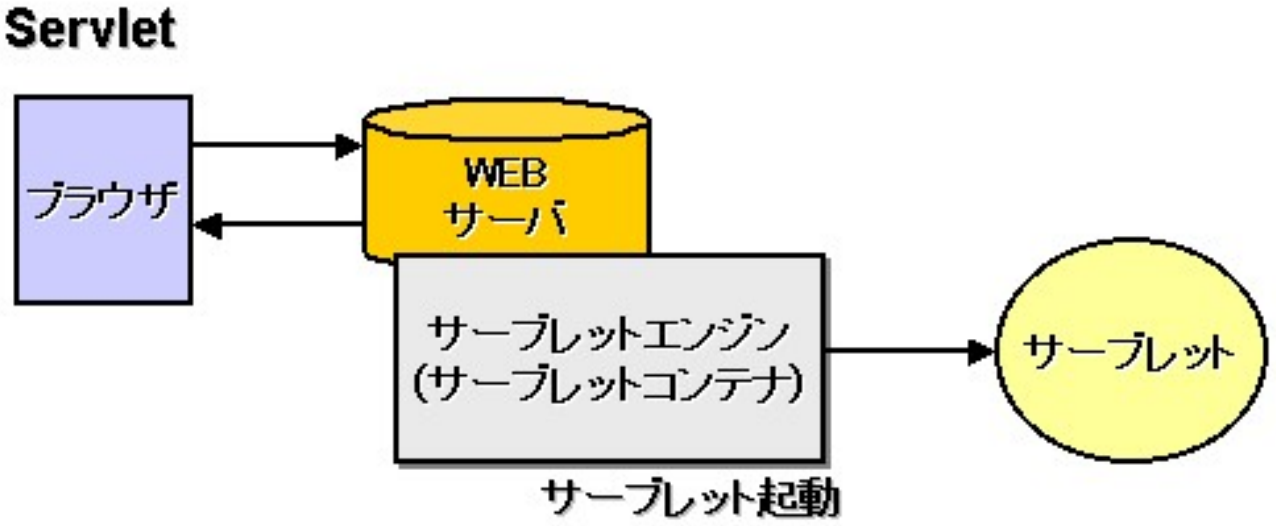


- ブラウザ側の要求ごとに応じてそれぞれ対応するプロセスが起動
- 処理の膨大化



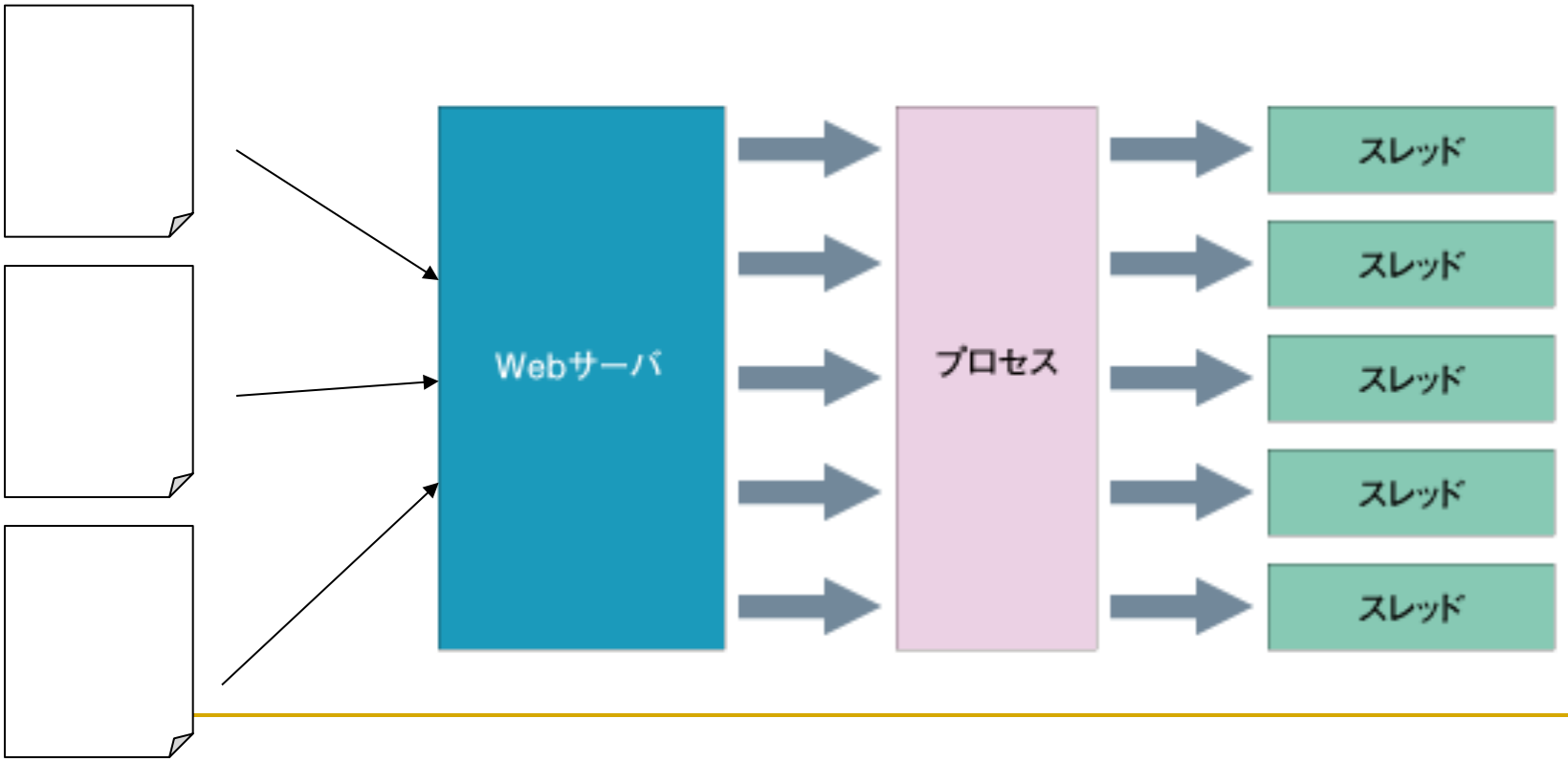
Javaサーブレット

- Java言語によって作成された，Webサーバ上で実行されるモジュール
- クライアント側からの要求に応じて，動的にHTML文書を作成し,クライアントに送信(インタラクティブを実現)

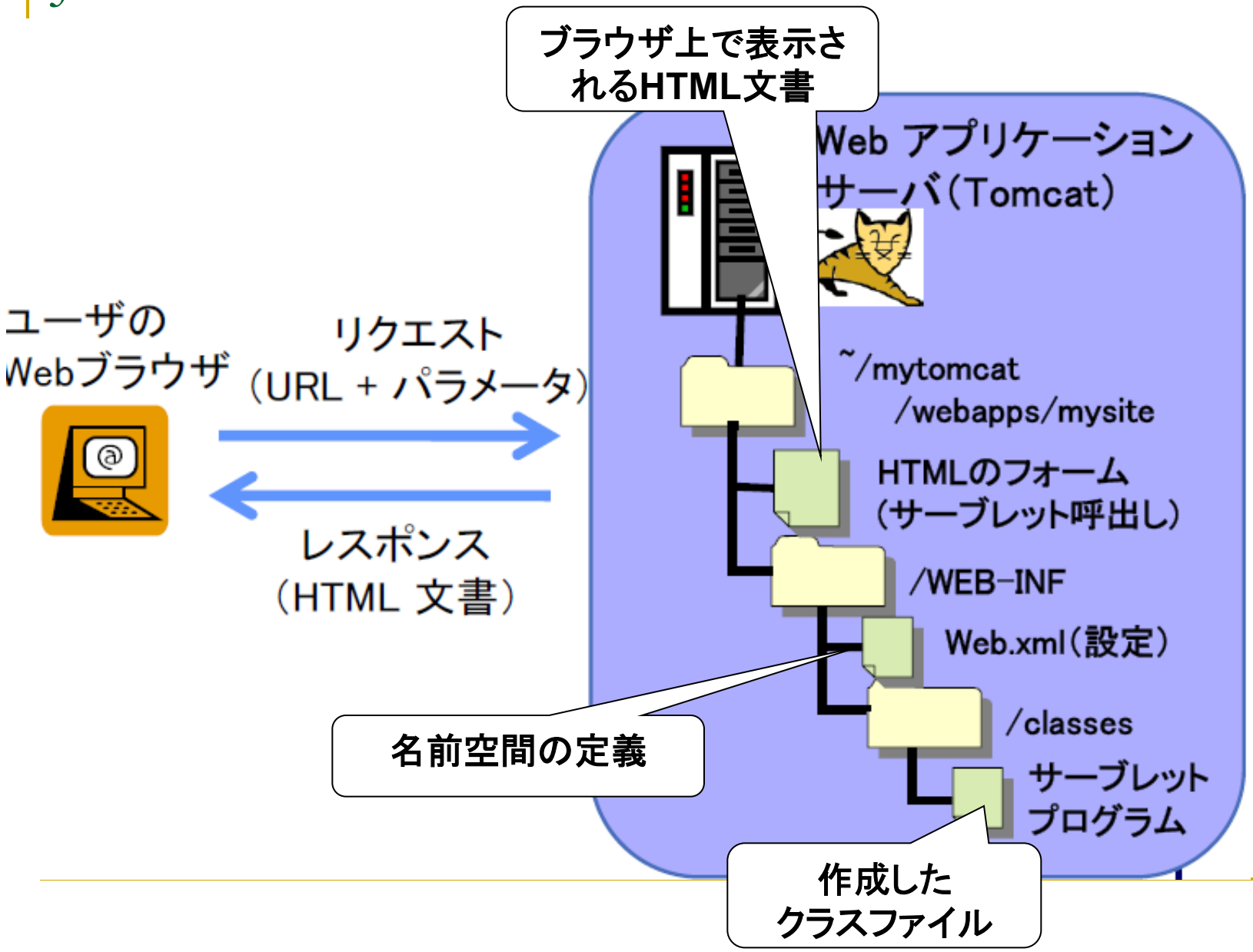


Javaサーブレット

- サーバコンテナといわれるサーバー上で，処理
- Webサーバー上でひとつのインスタンス（プロセス）を共有
- 要求をひとつのスレッドとして起動するので,負担をかけない.



Java Servletのファイル構成



Javaサーブレットの準備

1. Tomcatの設定
2. LinkするためのHTMLファイルの用意
3. サーブレットプログラム「~~.java」、「~~.class」の作成
4. コンテンツ定義のための「web.xml」の用意
5. その他，必要に応じて

Javaサーブレットの準備（ 1. Tomcatの設定）

1. Tomcat

サーブレットやJSPを実行するためのサーブレットコンテナ（サーブレットエンジン）のこと

サーブレットコンテナとは，HTMLなどのWebページを動的に生成することができるJavaサーブレットを動作させるためのソフトウェア

2. 以下のスクリプトを実行

/kyozai/amaeda/mp1/scripts/tomcatSetup.sh（初期環境設定）

$CATALINA_HOME/bin/startup.sh（Tomcatの起動）

3. URLの確認

- http://www.ritsumei.ac.jp/~loginID:8080 （WAN側からの確認）
- http://localhost:8080/ (各自のPC，ローカル環境からの確認)

Servletの動作は必ずPort8080を用いること

Javaサーブレットの準備
(2. Linkするための HTMLファイルの用意)

Webアプリケーションなので,必ずHTML上にリンクさせましょう.

```
<html>
  <head>
      <title>HelloWorld</title>
  </head>
  <body>
     <a href="http://localhost:8080/mysite/Myserve">
        クリックして下さい
      </a>
     <br>
   </body>
</html>
```

Javaサーブレットの準備 (サーブレット用のプログラムの作成 ～～.java)

- サーブレットは実際にdoXxxxメソッドを呼び出す（ロジック）

```
/* Sample9 Myserv.java */
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Myserv extends HttpServlet {

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

            Response.setContentType(“text/html;charset=utf-8”);
            PrintWriter out = response.getWriter();
            out.println(“<html><body><h1>Hell World!</h1>
                        </body></html>”);
        }
}
```

サーブレットとなるMyserv

サーブレットの親クラス

doGet: GETリクエスト

HTML文書の記述
この文書がサーバ側から動的に提供

Javaサーブレットの準備 (サーブレット用のプログラムの作成 ～～.java)

- サーブレットは実際にdoXxxxメソッドを呼び出す（ロジック）

```
/* Sample9 Myserv.java */
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Myserv extends HttpServlet {

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

            Response.setContentType("text/html;charset=utf-8");
            PrintWriter out = response.getWriter();
            out.println("<html><body><h1>Hell World!</h1>
                        </body></html>");
        }
}
```

サーブレットに必要なクラスをインポート

HttpServletのdoGetメソッドはここでオーバーライド(再定義)されている

オーバーライドとは：
継承時のスーパークラスで定義されたメソッドと同じ名前、引数を持つメソッドを、サブクラスで再定義すること

Javaサーブレットの準備
(サーブレット用のプログラムの作成 ～～.java)

```
/* Sample9 Myserv.java */
/*<snip>*/
public class Myserv extends HttpServlet {

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

        tentType(“text/html;charset=utf-8”);

        PrintWriter out = response.getWriter();

        out.println(“<html>\n”);
        out.println(“<body>”);
        out.println(“<h1>” );
        out.println(“Hell World!” );
        out.println(“</h1>” );
        out.println(“</body>” );
        out.println(“</html>”);
    } }
```

クライアントからの要求オブジェクト

クライアントからの応答オブジェクト

文字列を表示するための
PrintWriterオブジェクトを取得

文字列出力

先のスライドを分解した形

Javaサーブレットの準備(コンテンツの設定：web.xml)

- サーブレットの呼び出し方や初期値などの設定を行う
- サーブレットを動作させるためには必ず用意する

web.xmlの基本的な構成

```
<?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_2_5.xsd" version="2.5">

    <servlet>
        . . . . . .
    </servlet>

</web-app>
```

DTD宣言

利用するサーブレットの定義

Javaサーブレットの準備(コンテンツの設定：web.xml)

- <servlet>~ </servlet>内に利用するサーブレット名を記述する
- サーブレットを動作させるためには必ず用意する

servlet の基本的な構成

```
（続き）
    <servlet>
        <servlet-name>Myserv</servlet-name>
        <servlet-class>Myserv</servlet-class>
    </servlet>
```

サーブレットの名前

利用するクラスファイル名の定義

- 以上の記述をサーブレットの数だけ定義する

Javaサーブレットの準備(コンテンツの設定：web.xml)

- <servlet-mapping>~ </servlet-mapping>内に利用する
  サーブレット名を記述する
- リクエストされた URLがどのサーブレットに処理されるかを示す

servlet-mappingの基本的な構成

```
（続き）
    <servlet-mapping>
        <servlet-name>Myserv</servlet-name>
        <url-pattern>Myserv</url-pattern>
    </servlet-mapping>
```

サーブレットの名前

サーブレットURLの指定

- 以上の記述をサーブレットの数だけ定義する

Javaサーブレットの doPostについて1 (クライアント側への最初に応答する)

```
/* Sample10 ReadForm.java */
 /* HTTP の GET メソッドで呼び出される関数 */

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

          response.setContentType("text/html;c
          PrintWriter out = response.getWriter();
          out.println("<html><head><meta http-equiv=\"C      t-Type\"
         content=\"text/html; charset=utf-8\"></head><body>");
          out.println("<form action=\"ReadForm\" method=\"post\">");
          out.println("名前を入力してください:");
          out.println("<input type=\"text\" name=\"username\">");
          out.println("<input type=\"submit\" value=\"送信\">");
          out.println("</form></body></html>");
```

まずはdoGetで情報取得のサーブレットを行う

Postを用いた情報取得の指示

usernameというパラメータに文字列入力の指示

Javaサーブレットの doPostについて1 (クライアントの要求をフォームで受け付ける処理)

Postの処理はdoPostメソッドのオーバーライド

```
/* Sample10 ReadForm.java */

/*snip */
/* HTTP の POST メソッドで呼び出される関数 */
    public void doPost(HttpServletRequest request, HttpServletResponse
                                        response) throws ServletException, IOException {
            request.setCharacterEncoding("UTF-8");
            String username = request.getParameter("username");
            response.setContentType("text/html;charset=utf-8");

            PrintWriter out = response.getWriter();
            out.println("<html><head><meta http-equiv=\"Content-Type\"
                        content=\"text/html;  charset=utf-8\">
                         </head><body><h1>こんにちは, " + username + "さん</h1>
                         </body></html>");
        }
    }
```

JavaサーブレットのdoPostについて2 (クライアントから送られた要求の処理)

```
/* Sample10 ReadForm.java */
//snip
/* HTTP の POST メソッドで呼び出される関数 */
    public void doPost(HttpServletRequest request, HttpServletResponse
                                        response) throws ServletException, IOException {

            request.setCharacterEncoding("UTF-8");

            String username = request.getParameter("username");

            response.setContentType("text/html;charset=utf-8");

 /*<snip>
 */
```

データ文字コードの指定

doGetのパラメータ「username」をStringオブジェクトusernameに入れる

コンテントタイプの設定

JavaサーブレットのdoPostについて2 (クライアントに対する応答HTMLの処理)

```
/* Sample10 ReadForm.java */
//snip
/* HTTP の POST メソッドで呼び出される関数 */
    public void doPost(HttpServletRequest request, HttpServletResponse
                                        response) throws ServletException, IOException {
            /*snip
              */
            PrintWriter out = response.getWriter();
            out.println("<html><head><meta http-equiv=\"Content-Type\"
                        content=\"text/html;  charset=utf-8\">
                         </head><body><h1>こんにちは，" + username + "さん</h1>
                         </body></html>");
        }
    }
```

先ほど得られたString　usernameの文字列オブジェクト

JavaサーブレットのdoPostのHTML

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"></head>
<body>
<form action=”ReadForm” method=”post”>
<input type=”text” name=”username”>
<input type=”submit” value=”送信”>
</form>
</body></html>
```