上級プログラミング2（第11回）

工学部　情報工学科
木村昌臣
中島毅

今日のテーマ

- 先回までの復習

- Webアプリケーションプログラム
  - JSP　JDBCのjarファイルを追加してください
  - JavaBeansをつかったプログラム

サーブレット(Servlet)

- Webアプリケーションサーバで動作する
  Javaプログラム　→　動的にWebページを作る
- HttpServletクラスを継承して作る
  - 次の三つのパッケージをインポートする必要あり
    - java.io.*
    - javax.servlet.*
    - javax.servlet.http.*
  - HttpServletRequestで入力、
    HttpServletResponseで出力を行う

JSP

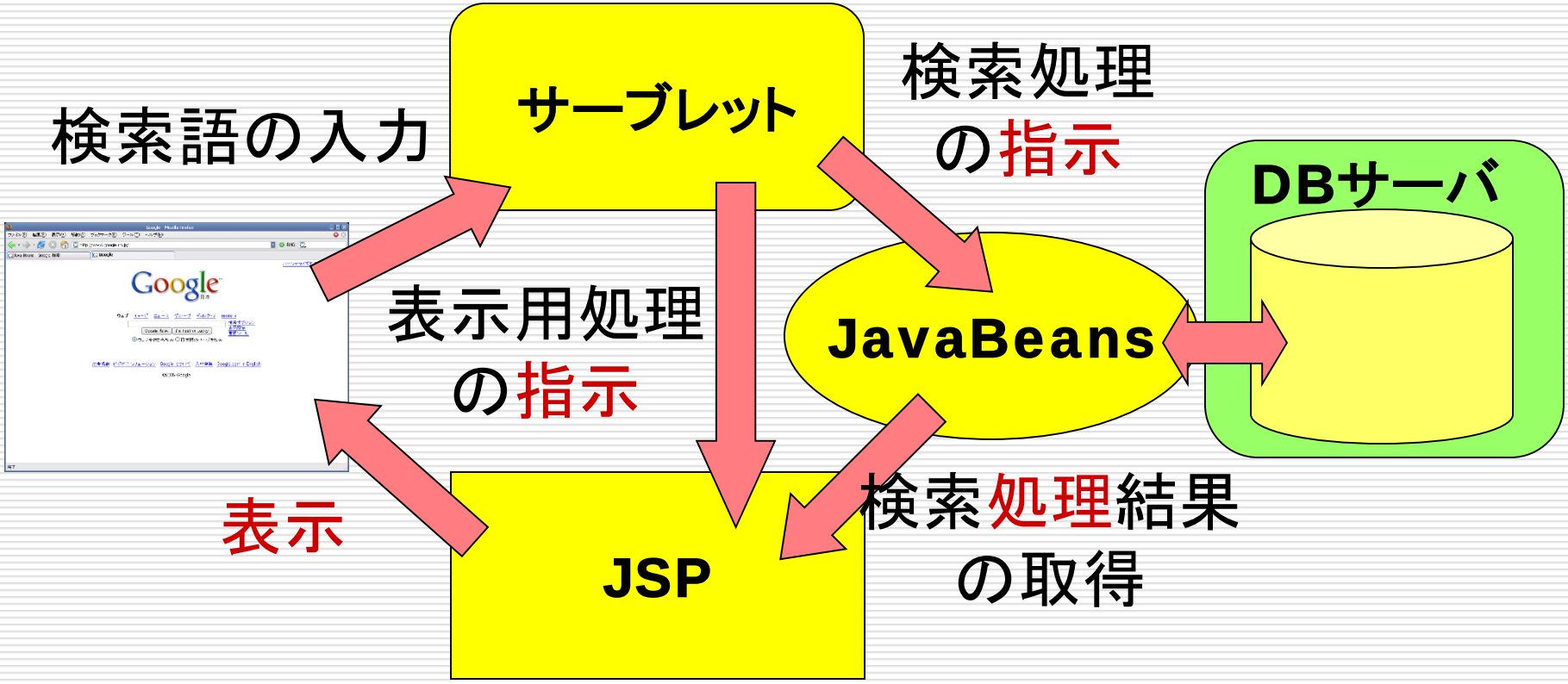
- Java Server Page
- HTMLに Javaのプログラム処理を付加したもの
  - サーブレットは最終的にHTML形式で出力
    →ページのレイアウトの調整を詳細に行うことは困難
  - JSPでは，Webページのレイアウトを決めて（HTMLベース）、必要な処理（Javaプログラム）を付加する

JavaBeans

- いろいろなWebアプリケーションで共通に使われるJavaクラス群をモジュール化してまとめたもの
  - データ接続用など
  - クラスのプロパティ（変数）にアクセスするときには、

    直接、変数にアクセスするのではなく、メソッド経由でアクセスする
    - set変数名()・・・・変数に値のセット
    - get変数名()・・・・変数の値を取得

それぞれの役割（検索エンジンを例として）



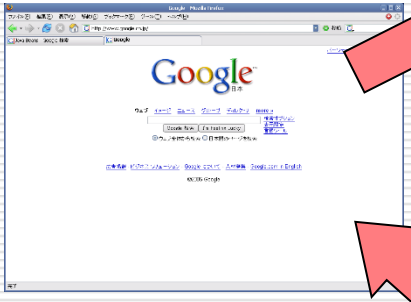
※ただし、JavaBeansやJSPの役割をサーブレットだけで実現することもある

MVCモデル

リクエストに応じて適切な処理を指示する

Control

サーブレット



JavaBeans

JSP

ビジネスロジックのモデリングを行う

Model

出力を行う

View

Webアプリケーションプログラミング

JSP復習

サンプルプログラムを，JSPを使うように改造 ～まずは出力用のHTMLを作る～

```
<html>
<head>
<meta http-equiv="Content-Type"
                content="text/html; charset=Shift_JIS">
<title>Hello world</title>
</head>
<body>
<h1> こんにちは </h1>
<h2>  ○○○さん、こんにちは！</h2>
</body>
```

※ファイルの拡張子をjspとする（この例ではoutput.jspとする）

次のように処理を追加する(赤字部分)

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<html>
<head>
<title>Hello world</title>
</head>
<body>
<h1> こんにちは </h1>
   <%
       String name=request.getParameter("MYNAME");
   %>
<h2>ようこそ！<%= name %>さん、こんにちは！</h2>
</body>
</html>
```

pageディレクティブ

- JSPをどのように処理したらいいのか指示
  - importしたいパッケージを指定
  - バッファサイズを指定
  - 文字コードなどを指定（contentType)

```
<%@ page contentType=
        "text/html; charset=Shift_JIS" %>
```

※ちなみに、JSP内部の<% ～ %>はJavaによる処理と関係する部分である

pageディレクティブの構文

```
<%@ page
        [ language="java" ]
        [ extends="package.class" ]
        [ import="{package.class | package.*}, ..." ]
        [ session="true|false" ]
        [ buffer="none|8kb|sizekb" ]
        [ autoFlush="true|false" ]
        [ isThreadSafe="true|false" ]
        [ info="text" ]
        [ errorPage="relativeURL" ]
        [ contentType="mimeType [ ; charset=characterSet ]" |
                "text/html ; charset=ISO-8859-1" ]
        [ isErrorPage="true|false" ]
        [ pageEncoding="characterSet | ISO-8859-1" ]
        [ isELIgnored="true|false"]
%>
```

スクリプトレット：Javaプログラムの埋込み

- <% と %> で囲まれた部分にJavaのコードを書くことが出来る

```
<%
   String name=request.getParameter("MYNAME");
%>
```

- HTMLをまたいで書くことも出来る

```
<%  for(int i=0; i<10; i++){          %>
        <H1>10回書くぞ！！！</H1><HR>
<%  }                                 %>
```

代入

- HTMLの中に変数の値を埋め込みたいときは
  ＜％＝ 変数 ％＞とする。
  ※ ％と＝の間にスペースなどを入れないこと
- 処理結果を埋め込むときによく利用される

＜h2＞ようこそ！＜％＝ name ％＞さん、こんにちは！＜/h2＞

サーブレットの変更

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SimpleServlet2 extends HttpServlet {
    public void doGet(HttpServletRequest request,
                              HttpServletResponse response)
        throws IOException, ServletException
    {    //JSPのURL
        String url="/servlet/output.jsp";
        RequestDispatcher dispatcher
            =getServletContext().getRequestDispatcher(url);
        dispatcher.forward(request, response);
    }
}
```

RequestDispatcherクラス

- サーブレットからJSPへブラウザからのリクエストを転送し、残りの処理の実行を指示
- getServletContext().getRequestDispatcher(url)で生成
（urlはJSPのURL文字列)
- forwardメソッドで転送

```
dispatcher.forward(request, response);
```

実行結果


Servlet Examples - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(
戻る 検索 お気に入り
アドレス(D) http://localhost/webapps/input2.html 移動 リンク
Kimura
送信
ページが表示されました
イントラネット
Hello world - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(
戻る 検索 お気に入り
アドレス(D) http://localhost/webapps/servlet/Simple 移動 リンク
こんにちは
ようこそ！Kimuraさん、こんにちは！
ページが表示されました
イントラネット

Webアプリケーションプログラミング

JavaBeansを使ってみる

JavaBeans

- Webアプリケーションで共通に使われるJavaクラス（Bean）群をまとめたもの
  - 引数なしのコンストラクタのみ
    - コンストラクタの引数に入るデータが何かは自明でないため、必要なデータのセットはsetterで行う
  - プロパティを扱うことにより処理を行う
    - privateなフィールドを持つ
    - getterやsetterを持ち、そのフィールドがカプセル化されている
  - シリアライズ（永続化）可能

JavaBeans

- Beanが持つ情報をプロパティと呼ぶ
- プロパティはフィールドとsetterメソッド、getterメソッドの組で実現される
  - フィールドはprivateで宣言
  - setフィールド名()・・・・setter(変数に値のセット)
  - getフィールド名()・・・・getter(変数の値を取得)

直接、変数にアクセスするのではなく、メソッド経由でアクセスするカプセル化が行われていることに注意

サンプルプログラムにJavaBeansの処理を付け加えてみる

- Beanの仕様:
  「入力された名前の文字数を数えて表示」
- JSPからBeanのインスタンスを呼び出す時
  - Beanのクラスは必ずパッケージする(そうしないと, Tomcatではエラー)
  - JSPやサーブレットに加えるべき処理は後述

Bean

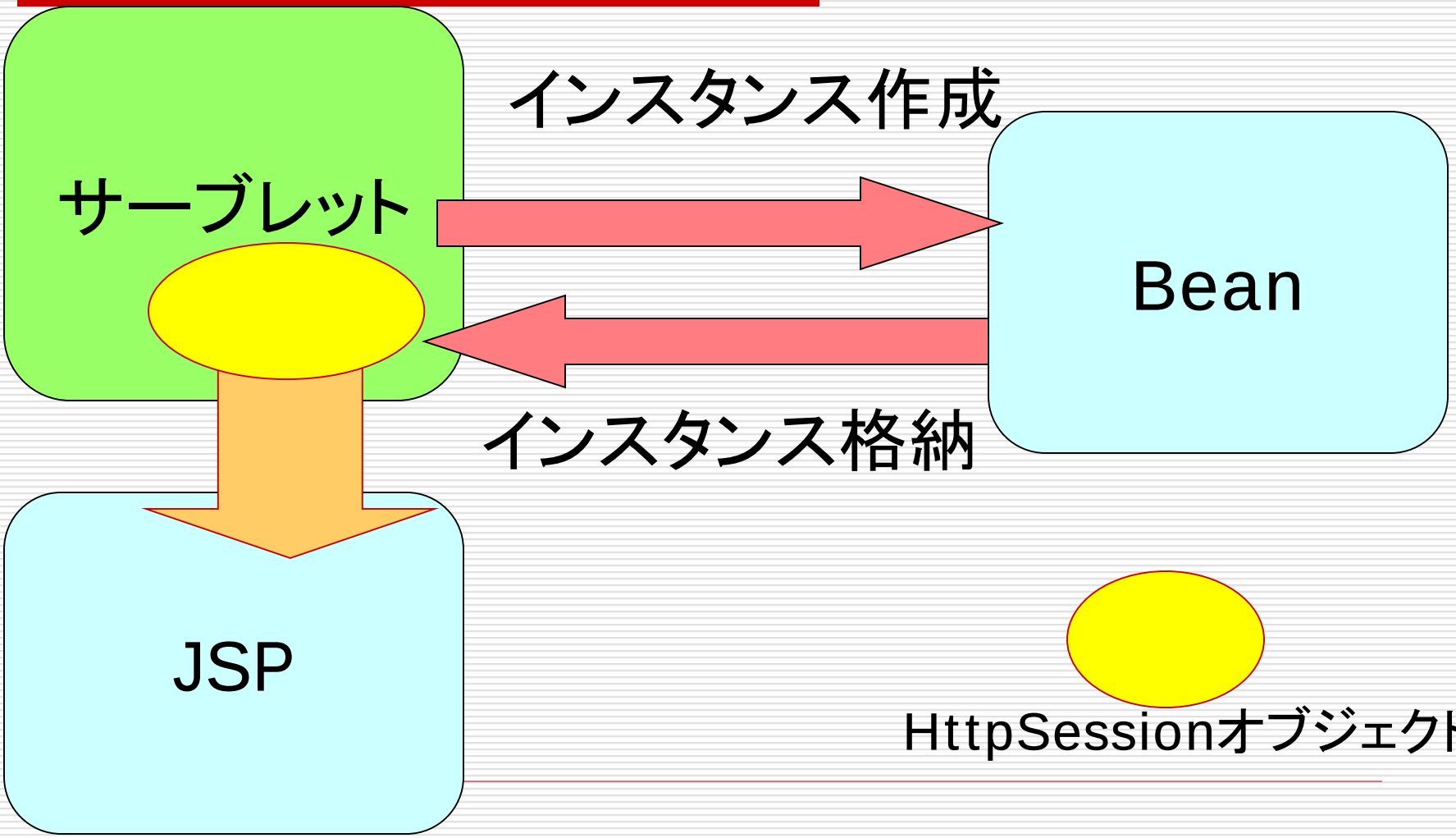
```
package simple;
public class SimpleBean {
	private String name="";
	private String message="";

	public void setName(String name){
		this.name=name;
	}
	public String getName(){
		return name;
	}
	public String getMessage(){
		return message;
	}
	public void calc(){
		int i=name.length();
		this.message="あなたの名前は、"+i+"文字です。";
	}
}
```

サーブレットへの処理の追加

- Beanのインスタンスを作成（当然、Beanが属するパッケージはインポートしておく）。
- 作成したインスタンスをJSPから参照できるオブジェクトに格納する
  - HttpServletRequestオブジェクト
    - 一回のリクエストの範囲
  - HttpSessionオブジェクト
    - 同じクライアントマシンからのリクエストの範囲内
    - Webアプリケーション全体（複数ユーザで共有する情報を格納）

JSPへのBeanオブジェクトの渡し方



サーブレット

```
....
import simple.*;

public class SimpleServlet3 extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException
    {   String name=request.getParameter("MYNAME");
        HttpSession session= request.getSession(true);
        SimpleBean sb = new SimpleBean();
        sb.setName(name);
        sb.calc();
        session.setAttribute("sb",sb);
        ....
```

暗黙オブジェクト

- JSPには、事前に宣言することなく利用できるオブジェクトがある（以下、主要なもの）
  - out
    - コンテンツを出力するためのオブジェクト
  - request ： HttpServletRequest オブジェクトに対応
    - リクエスト（要求）情報にアクセスするオブジェクト
  - session ： HttpSession オブジェクトに対応
    - セッション内で共有可能な情報を格納するオブジェクト
  - response: HttpServletResponse オブジェクトに対応
    - レスポンス(応答)情報にアクセスするオブジェクト
  - application: ServletContext オブジェクトに対応
    - 複数ユーザー間で共有可能な情報を格納するオブジェクト

JSP

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<%@ page import="simple.SimpleBean" %>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
                                        charset=Shift_JIS">
<title>Hello world</title>
</head>
<body>
<h1> こんにちは </h1>
<%
  SimpleBean sb= (SimpleBean)session.getAttribute("sb");
%>
<h2>ようこそ！<%= sb.getName() %>さん、こんにちは！</h2>
余計なお世話ですが、<%= sb.getMessage() %>
</body>
</html>
```

jsp:useBeanタグ と jsp:getPropertyタグ

- JSPはHTMLベースの記載方法をとるので、なるべくタグで表現したい。そのため、以下のスクリプトレットは次のように置き換えることができる。

```
<%
  SimpleBean sb= (SimpleBean)session.getAttribute("sb");
%>
<jsp:useBean id=“sb" class="simple.SimpleBean"
                                        scope=“session" />

<%= sb.getName() %>
<jsp:getProperty name=“sb" property=“name" />
```

スコープ

JSP

```
<%@ page contentType="text/html; charset=Shift_JIS" %>
<jsp:useBean id=“sb" class=" simple.SimpleBean"
                                        scope=“session" />
<html>
<head>
<title>Hello world</title>
</head>
<body>
<h1> こんにちは </h1>
<h2>ようこそ！
<jsp:getProperty name=“sb" property=“name" />
さん、こんにちは！</h2>
余計なお世話ですが、
<jsp:getProperty name=“sb" property=“message" />
</body>
</html>
```

スコープ

- page
  - JSP内のみで有効＝ローカル変数と同じ
- request
  - Webブラウザからの要求がでてから処理結果がブラウザに返るまで有効
- session
  - ユーザが、そのWebアプリケーションを利用している間ずっと有効（ブラウザを閉じるまで）
- application
  - ユーザによらずWebアプリケーション全体で有効

Sessionがなぜ必要か

- HTTPは、もともとHTMLファイルをサーバからダウンロードすることが目的のプロトコル
- HTTPでは，状態を持てない（ステートレス）
  - HTMLを要求するユーザが同一か否かに無頓着

  → ショッピングサイトのようにユーザとサーバの間で情報をなんどもやり取りする場合には不向き

  → この欠点を補うため、クッキーの機能を利用。HttpSessionオブジェクトにより、アクセスしているユーザの判別およびデータの引継ぎを行う

期末テストに向けて

期末テストは，来週1限に実施（2つの教室に分かれて実施するので注意）

- 持ち込み不可
- 各回の内容をきちんと復習しておくこと
- 範囲はまんべんなく
  - Javaの文法
  - ネットワークプログラミング、GUIプログラミング、データベースプログラミング、Webアプリケーションプログラミング

など