u59●Servletによるフォワード

EJBの分散オブジェクト（MVCモデル）

「直列化による転送モデル」の実習で作成したBeanをServletでインスタンス化してjspへフォワードするMVCモデルを作成します。これはEJBの分散オブジェクトと呼ばれる技術です。システムの構成と設計からプログラムを実装して動きを確認してみましょう。

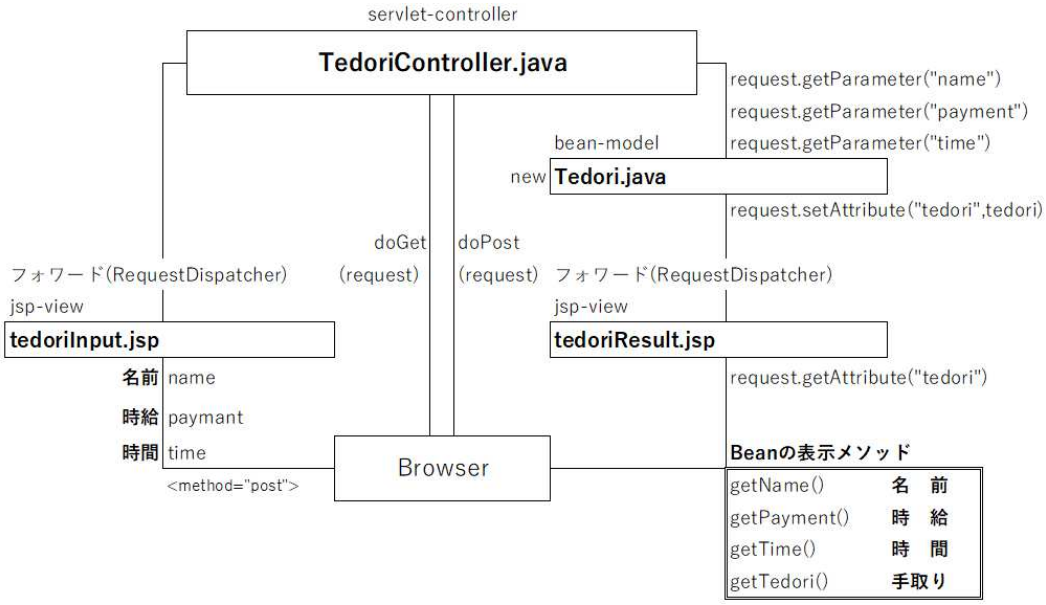
●仕様

Webブラウザで表示された画面に名前と時給と労働時間を入力し計算ボタンをクリックすると、10%の所得税を差し引いた手取り金額を表示する画面へ遷移するWebアプリケーションを作成する。

●システム構成

Model: Tedori.java(**package:**jp.ict.aso.model)
View:tedoriInput.jsp,tedoriResult.jsp
Controller: TedoriController.java(**package:**jp.ict.aso.controller)
url-pattern:/Tedori
directory:~libcon/WEB-INF/jsp,~libcon/WEB-INF/classes

●内部設計



モデル(M)の作成

●モデル(JavaBeans)の制作：

「直列化による転送モデル」の実習で作成済

コントローラ(C)の作成

●コントローラ(Servlet)の制作：

JavaBeansをWebアプリケーションで実現するための制御部分であるコントローラをServletで実現してみましょう。
Servletファイルの保存先は～/libcon/WEB-INF/classesですよ！

[root@aso classes]# pwd
/var/lib/tomcat9/webapps/libcon/WEB-INF/classes
[root@aso classes]# **vi TedoriController.java**
↓　**InputServlet.java**のソースファイルからコピーすると少し楽です

```
 1  package jp.ict.aso.controller;
 2
 3  import java.io.*;
 4  import javax.servlet.*;
 5  import javax.servlet.annotation.*;
 6  import javax.servlet.http.*;
 7  import jp.ict.aso.model.*;
 8
 9  @WebServlet("/Tedori")
10   public class TedoriController extends HttpServlet {
11
12    protected void doGet(HttpServletRequest request,
13      HttpServletResponse response)
14      throws ServletException, IOException {
15
16      // フォワード
17      RequestDispatcher dispatcher =
18      request.getRequestDispatcher
19        ("/WEB-INF/jsp/tedoriInput.jsp");
20      dispatcher.forward(request, response);
21    }
22
23    protected void doPost(HttpServletRequest request,
24      HttpServletResponse response)
```

```
    throws ServletException, IOException {

    // リクエストパラメータを取得
    request.setCharacterEncoding("UTF-8");  // 日本語変換
    String name = request.getParameter("name"); // 名前
    String payment = request.getParameter("payment"); //
時給
    String time = request.getParameter("time"); // 労働時
間

    // インスタンス化して入力値をプロパティに設定
    Tedori tedori =
      new Tedori(name,
                 Integer.parseInt(payment),
                 Double.parseDouble(time)
          );

    // リクエストスコープに保存
    request.setAttribute("tedori", tedori);

    // フォワード
    RequestDispatcher dispatcher =
    request.getRequestDispatcher
      ("/WEB-INF/jsp/tedoriResult.jsp");
    dispatcher.forward(request, response);
  }
}

```

リクエストパラメータは文字列（String）でしか取得できません。
よって、Beanの引数に合わせて**String→Integer**あるいは**String→Double**の変換が必要です。

viewにあたるjspをまだ作成していませんので、とりあえずコンパイルだけ通しておきましょう。

[root@aso classes]# **javac -d . TedoriController.java**

ビュー(v)の作成

●ビュー(JSP)の制作：

上記Javaプログラムを Webアプリケーションで実現するための入出力の表示部分であるviewをJSPで実現してみましょう。
JSPファイルの保存先は～/libcon/WEB-INF/jspですよ！

[root@aso jsp]# pwd
/var/lib/tomcat9/webapps/libcon/WEB-INF/jsp
[root@aso jsp]# **vi tedoriInput.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>入力画面</title>
</head>
<body>
<h1>名前と時給と労働時間の設定</h1>
<form action="Tedori" method="post"><p>
名前:<input type="text" name="name" required>
時給:<input type="text" name="payment" required>
時間:<input type="text" name="time" required></p>
<input type="submit" value="支払い計算" required>
</form>
</body>
</html>

```

[root@aso jsp]# **vi tedoriResult.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
<%@ page import="jp.ict.aso.model.Tedori" %>
<%
// リクエストスコープに保存されたTedoriを取得
Tedori t = (Tedori)request.getAttribute("tedori");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>手取り計算</title>

```

```
13  </head>
14  <body>
15  <h1>手取り計算結果</h1>
16  <%= t.getName() %>さんの計算結果
17  <p>
18  時　　給：<%= t.getPayment() %>円
19  労働時間：<%= t.getTime() %>時間
20  手取り額：<%= t.getTedori() %>円
21  </p>
22  <a href="Tedori">戻る</a>
23  </body>
24  </html>
25
```

再起動と動作確認

●Tomcatを再起動

[root@aso jsp]# **systemctl restart tomcat9**

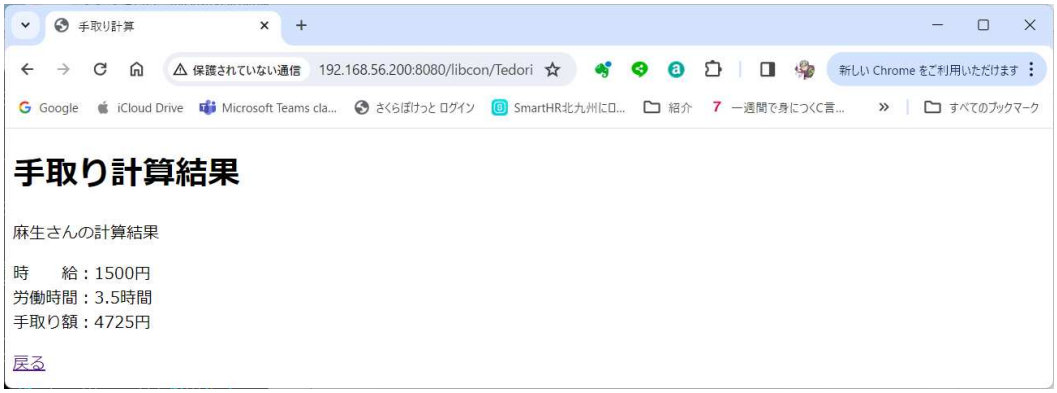
●動作確認

下記のURLにアクセスして確認します。

http://192.168.56.200:8080/libcon/Tedori

エラー処理は実装していません。
名前：の欄は文字列で、時給：と時間：の欄は数字を入れてください！！

画面が遷移しても～libcon/Tedoriからurlが変わらないことに注目してください。



※うまくいかないときはソースコードを良く見直しましょう。

ブラウザはキャッシュを読み込みますので　CTRL＋再読み込み　で対処しましょう。
またServletのリコンパイル時はこまめにtomcatを再起動したほうが安心です。