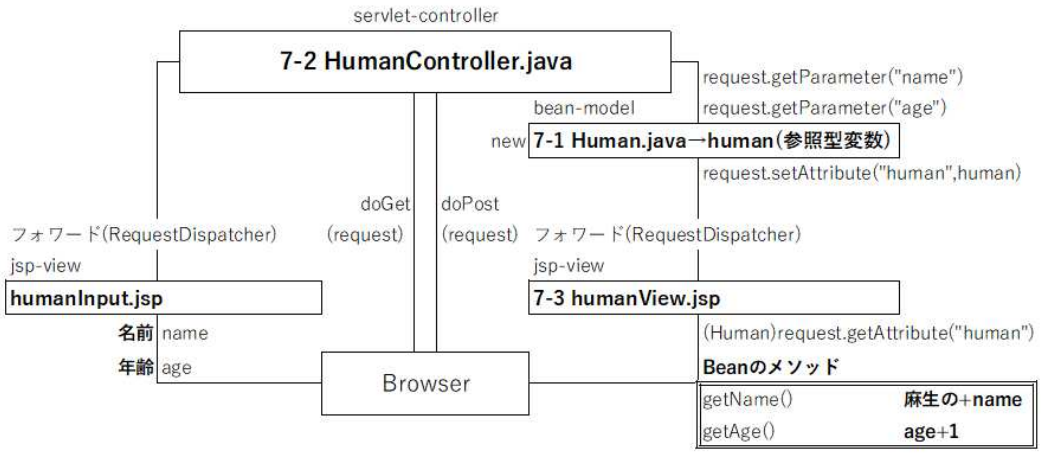
u第７章●MVCアーキテクチャ実装手順

リクエストスコープ

教科書の内容に従いHumanアプリケーションをMVCアーキテクチャで実装します。モデルのパッケージ名は**model**とします。ブラウザから名前と年齢を入力して、名前には「麻生の」を追加し年齢は「＋１」を行った結果を返します。

●システム構成
Model: Human.java(**package:**model)
View:humanInput.jsp,humanView.jsp
Controller: HumanController.java(**package:**servlet)
url-pattern:/Human
directory:~book/WEB-INF/jsp,~book/WEB-INF/classes

●内部設計



モデル(M)の作成

●モデル(JavaBeans)の制作：コード7－1

(1)まずは業務ロジックとなるJavaBeansプログラムを作成します
JavaBeansの保存先は～/book/WEB-INF/classesですよ！

[root@aso classes]# pwd
/var/lib/tomcat9/webapps/book/WEB-INF/classes

[root@aso classes]#
[root@aso classes]# **vi Human.java**

```
 1  package model;
 2  import java.io.Serializable;
 3
 4  public class Human implements Serializable {
 5    private String name;
 6    private int age;
 7
 8    public Human() {
 9    }
10     public Human(String name, int age) {
11       this.name = name;
12       this.age = age;
13     }
14     public String getName() {
15       return name;
16     }
17     public void setName(String name) {
18       this.name = "麻生の"+name;        //koko
19     }
20     public int getAge() {
21       return age;
22     }
23     public void setAge(int age) {
24       this.age = age+1;               //koko
25     }
26  }
27
```

[root@aso classes]#
[root@aso classes]# **javac -d . Human.java**
[root@aso classes]#
[root@aso classes]# **java model.Human**
Error: Main method not found in class model.Human, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
[root@aso classes]#

これは当然単独では動きませんよ！

(2)Beanを動かしてみましょう
ではJavaアプリケーションとしてこの**Beanを実行するコード**はどのように書けばよいでしょうか？

[root@aso classes]# **vi HumanMain.java**

```
 1  package jp.ict.aso;
 2  import model.*;
 3
 4  public class HumanMain{
 5   public static void main(String[] args){
 6
 7    System.out.print("INPUT NAME:");
 8    String name = new
java.util.Scanner(System.in).nextLine();
 9    System.out.print("INPUT AGE:");
10    int  age = new java.util.Scanner(System.in).nextInt();
11
12    Human h = new Human();
13    h.setName(name);
14    h.setAge(age);
15
16    System.out.print(h.getName()+"さんは");
17    System.out.println("来年"+h.getAge()+"歳です");
18   }
19  }
20
```

[root@aso classes]# **javac -d . HumanMain.java**
[root@aso classes]# **java jp.ict.aso.HumanMain**
INPUT NAME:磯そだち
INPUT AGE:24
麻生の磯そだちさんは来年25歳です
[root@aso classes]#

コントローラ(C)の作成

●コントローラ(Servlet)の制作：コード7－2

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～/book/WEB-INF/classesですよ！

[root@aso classes]# pwd
/var/lib/tomcat9/webapps/book/WEB-INF/classes
[root@aso classes]#
[root@aso classes]# **vi HumanController.java**
↓ **コード6-1のForwardServlet.javaのソースファイルからコピーすると少し楽です**
（TedoriController.javaからコピーするともっと楽です！）

```
 1  package servlet;
 2
 3  import java.io.IOException;
 4  import javax.servlet.RequestDispatcher;
 5  import javax.servlet.ServletException;
 6  import javax.servlet.annotation.WebServlet;
 7  import javax.servlet.http.HttpServlet;
 8  import javax.servlet.http.HttpServletRequest;
 9  import javax.servlet.http.HttpServletResponse;
10  import model.*;
11
12  @WebServlet("/Human")
13  public class HumanController extends HttpServlet {
14
15    protected void doGet(HttpServletRequest request,
16        HttpServletResponse response)
17        throws ServletException, IOException {
18
19      // フォワード
20        RequestDispatcher dispatcher =
21          request.getRequestDispatcher
22              ("/WEB-INF/jsp/humanInput.jsp");
23        dispatcher.forward(request, response);
24    }
25
26    protected void doPost(HttpServletRequest request,
27        HttpServletResponse response)
28        throws ServletException, IOException {
29
30      // リクエストパラメータを取得
31      request.setCharacterEncoding("UTF-8");
32      String name = request.getParameter("name"); // 名前
33      String age = request.getParameter("age"); // 年齢
34
35      // 入力値をプロパティに設定
```

```
36      Human h = new Human();
37      h.setName(name);
38      h.setAge(Integer.parseInt(age));
39
40      // リクエストスコープに保存
41      request.setAttribute("human", h);
42
43      // フォワード
44      RequestDispatcher dispatcher =
45          request.getRequestDispatcher
46              ("/WEB-INF/jsp/humanView.jsp");
47      dispatcher.forward(request, response);
48    }
49  }
50
```

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 . HumanController.java**
[root@aso classes]#

ビュー(v)の作成

●ビュー(JSP)の制作：コード7-3

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～/book/WEB-INF/jspですよ！

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

```
1  <%@ page language="java" contentType="text/html;
charset=UTF-8"
2      pageEncoding="UTF-8" %>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
```

```
 7  <title>名前と年齢</title>
 8  </head>
 9  <body>
10  <h1>名前と年齢の設定</h1>
11  <form action="Human" method="post">
12  名前:<input type="text" name="name">
13  年齢:<input type="text" name="age">
14  <input type="submit" value="GO">
15  </form>
16  </body>
17  </html>
18
```

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

```
 1  <%@ page language="java" contentType="text/html;
charset=UTF-8"
 2      pageEncoding="UTF-8" %>
 3  <%@ page import="model.Human" %>
 4  <%
 5  // リクエストスコープに保存されたHumanを取得
 6  Human h = (Human) request.getAttribute("human");
 7  %>
 8  <!DOCTYPE html>
 9  <html>
10  <head>
11  <meta charset="UTF-8">
12  <title>来年の年齢</title>
13  </head>
14  <body>
15  <h1>来年の年齢</h1>
16  <%= h.getName() %>さんは来年<%= h.getAge() %>歳
です
17  <a href="Human">戻る</a>
18  </body>
19  </html>
20
```

再起動と動作確認

●Tomcatを再起動

[root@aso WEB-INF]# **systemctl restart tomcat9**
[root@aso WEB-INF]#

●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/book/Human

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

画面が遷移しても～book/Human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。