Java EE7 Web
MVC實作

鄭安翔

ansel_cheng@hotmail.com

課程大綱

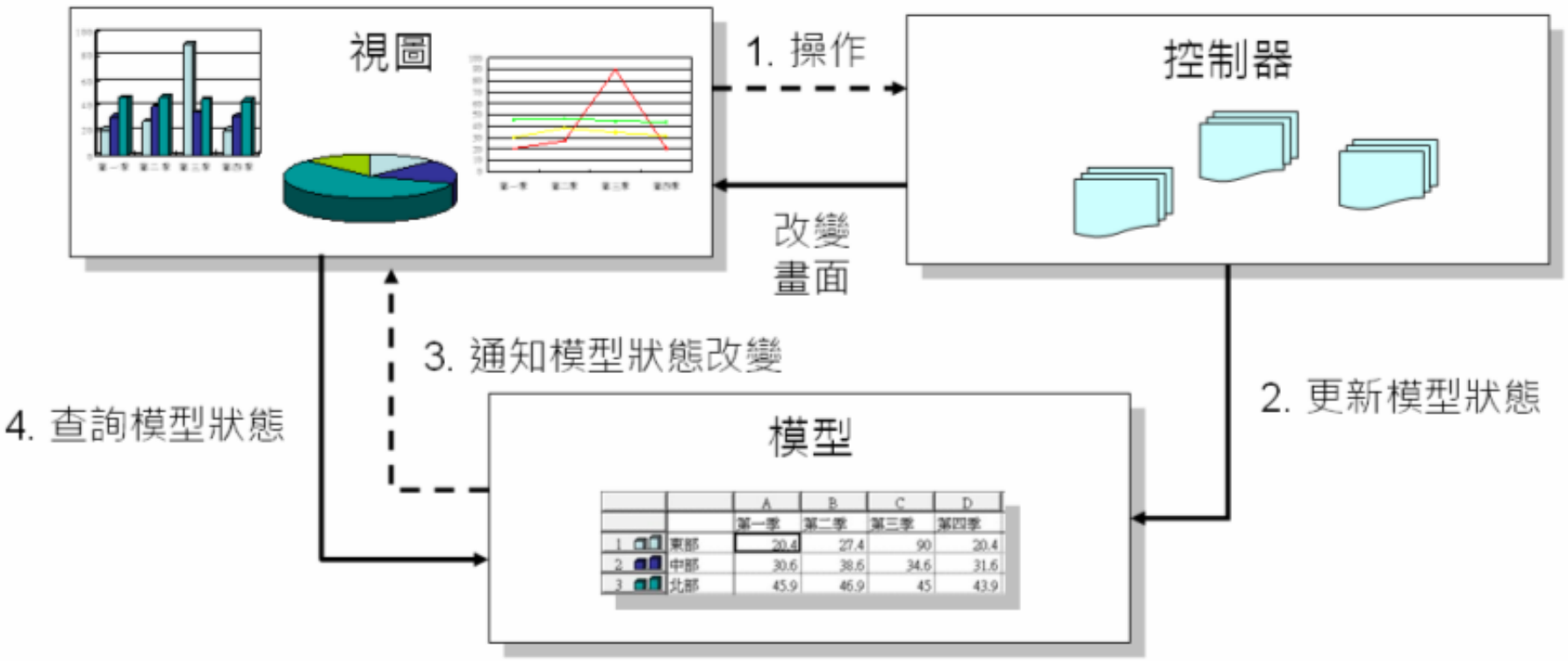
1) **MVC 設計模式**
2) 開發MVC架構程式
3) Soccer 範例

Model-View-Controller Pattern

- Problem :
  - 圖形介面應用程式中包含輸入資料檢核,處理邏輯及結果顯示,造成程式需經常修改
- Solution :
  - 將顯示邏輯，流程處理與商業邏輯分離
    - Model : 封裝商業邏輯的領域物件(Domain Object)
    - View : 呈現的使用者介面(User Interface)
    - Controller : 解析使用者的輸入行為,作用在Model上,並反映在View之上
  - Separation of Concerns

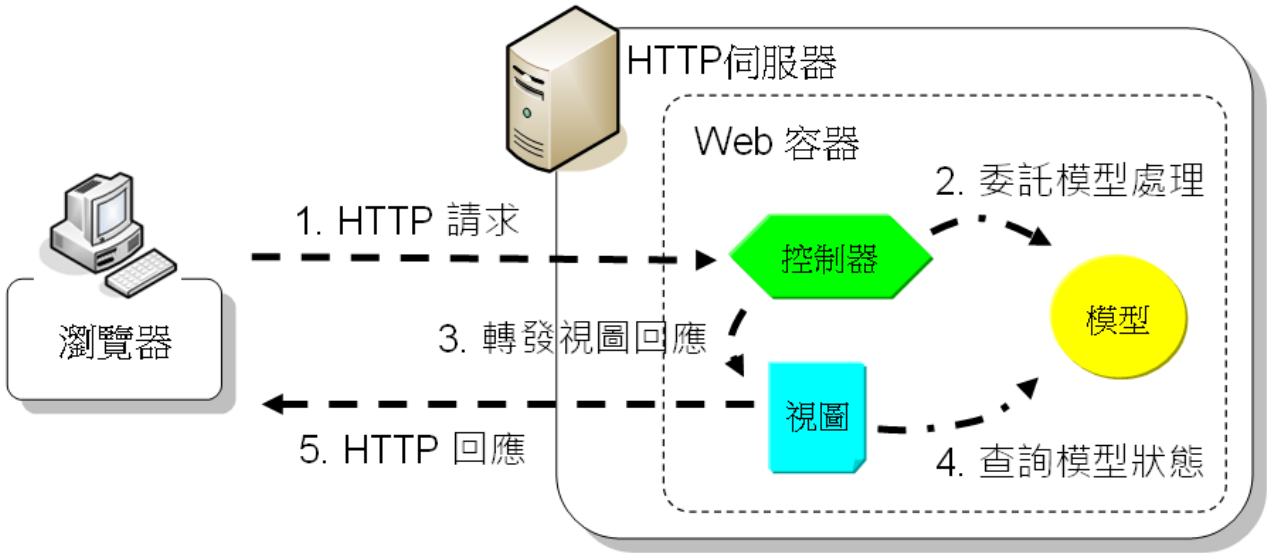
MVC

- 模型不會有畫面相關的程式碼
- 視圖負責畫面相關邏輯
- 控制器知道某個操作必須呼叫哪些模型



MVC Model 2

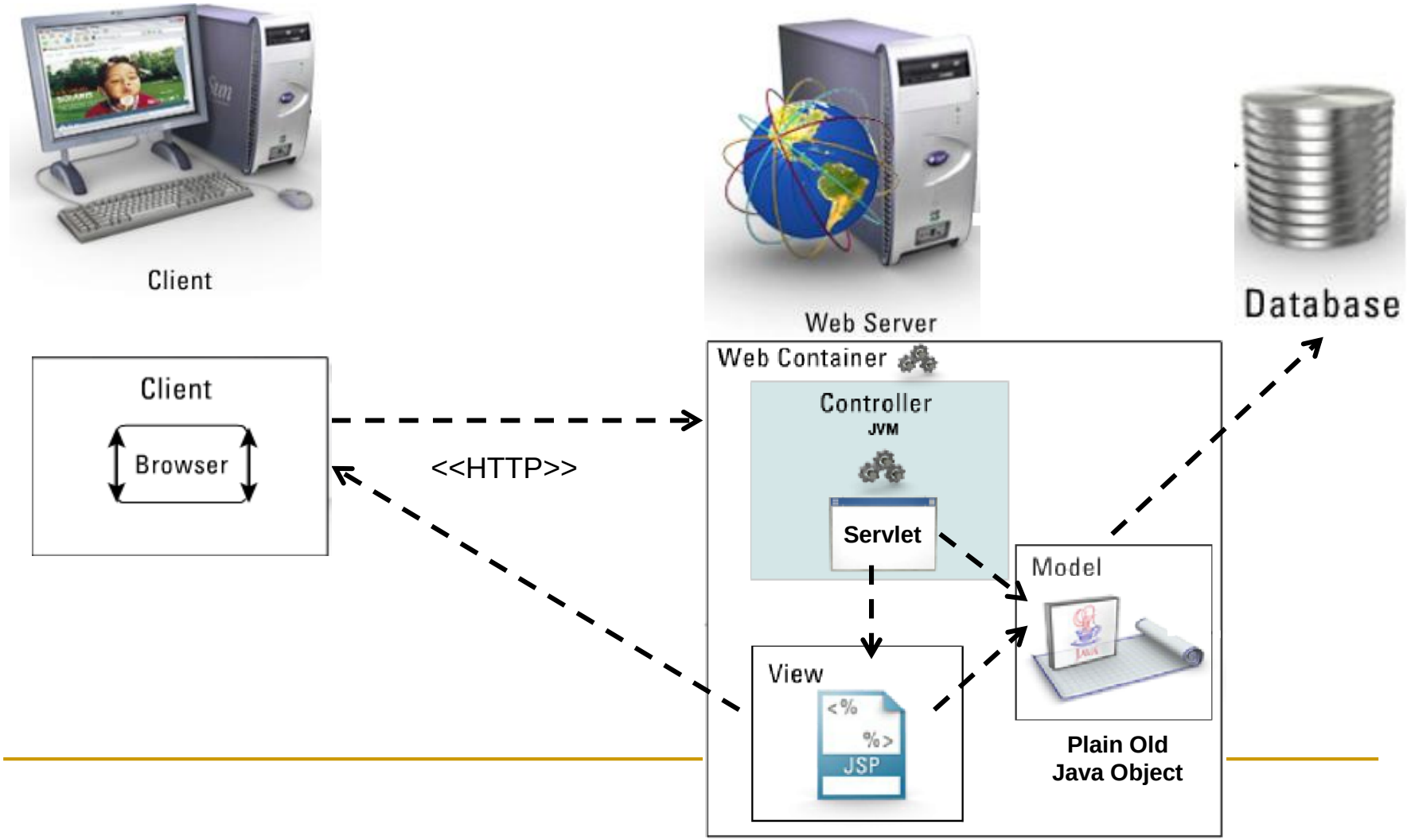
- Web應用程式是基於HTTP，為請求/回應模式
  - 由於HTTP屬於Stateless Protocol, Client-Server間的聯繫僅限於一個Connection中,因此Model 無法產生事件主動通知View進行變更.
- 套用在Web應用程式的設計上
  - 視圖部份可由網頁來實現
  - 伺服器上的資料存取或商務邏輯（Business logic）由模型負責
  - 控制元件接送瀏覽器的請求，決定呼叫哪些模型來處理



MVC Model 2

- 控制器（Controller）
  - 取得請求參數、驗證請求參數、轉發請求給模型、轉發請求給畫面，這些都使用程式碼來實現
  - 適合用 Java Servlet 來實現
- 模型（Model）
  - 接受控制器的請求呼叫，負責處理商務邏輯、負責資料存取邏輯等，這部份還可依應用程式功能，產生各多種不同職責的模型物件，模型使用程式碼來實現
  - 適合用 POJO (Plain Old Java Object) 來實現
- 視圖（View）
  - 接受控制器的請求呼叫，會從模型提取運算後的結果，根據需求呈現所需的畫面，在職責分配良好的情況下，基本上可作到不出現程式碼，因此不會發生程式碼與HTML混雜在一起的情況
  - 適合用 JSP 來實現

Java MVC Model 2 Architecture



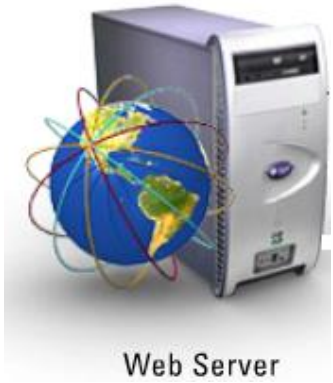
常用 Model 2 Frameworks

- Framework
  - 框架是部分實現，提供元件間溝通的基礎建設
  - 程式設計師套用特定框架,在此基礎上建立商業元件

- 常用 Model 2 frameworks：
  - Struts 1/2 from the Jakarta group
  - Spring MVC Framework
  - JavaServer™ Faces technology from Sun
  - Velocity from Apache

Java EE Architecture

Internet Client



Intranet Client

EJB

Enterprise JavaBeans

Java EE 平台

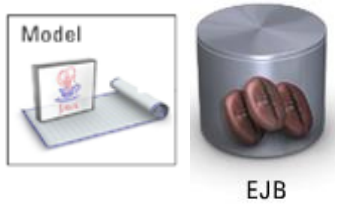
- 模組化設計
  - 商業邏輯更容易修改
- Enterprise JavaBeans 元件可使用容器提供的服務
  - 安全性
  - 交易
  - 持久性
  - 生命週期管理

開發的角色分配(job roles)

- 開發大型網路應用程式的工作角色包括：
  - 網頁設計人員
    - 負責建立應用程式的view元件
    - 主要應用技術是HTML網頁及JSP頁面。

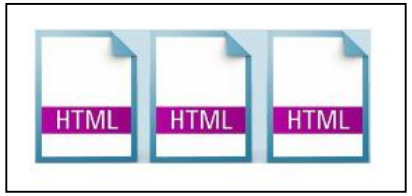
  - 網路元件開發人員
    - 負責建立應用程式的controller元件
    - 主要應用技術是Java Servlet 。

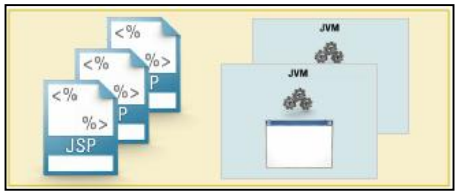
  - 商業元件開發人員
    - 負責建立應用程式的model元件
    - 主要應用技術是Java Beans、DAO元件(置於同一網路伺服器)、EJB元件(分散式遠端伺服器)。

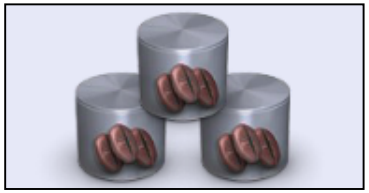


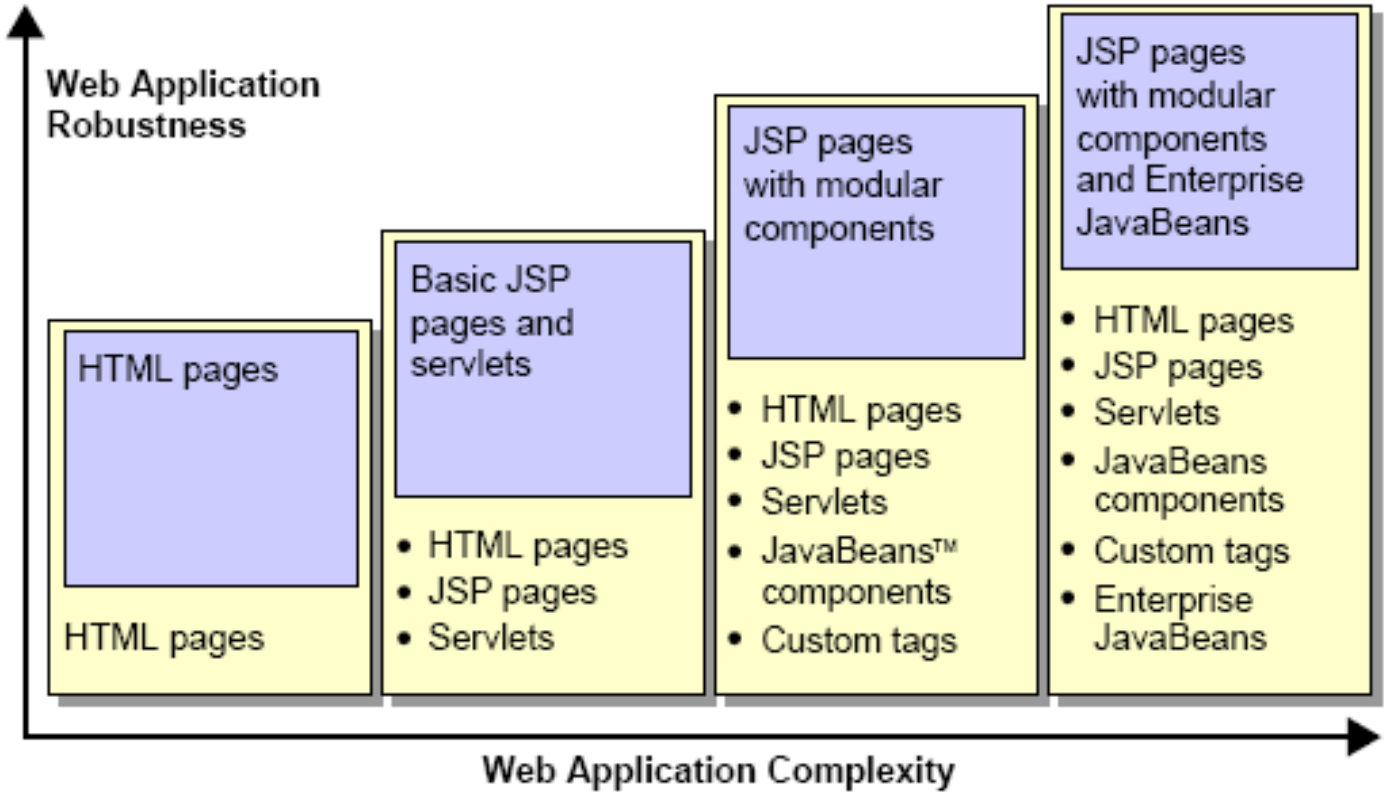
  - 資料庫管理人員 (Database Administrator)
    - 維護資料庫系統及資料表

網路應用程式階層


HTML
HTML
HTML


<%
%>
JSP
JVM
JVM

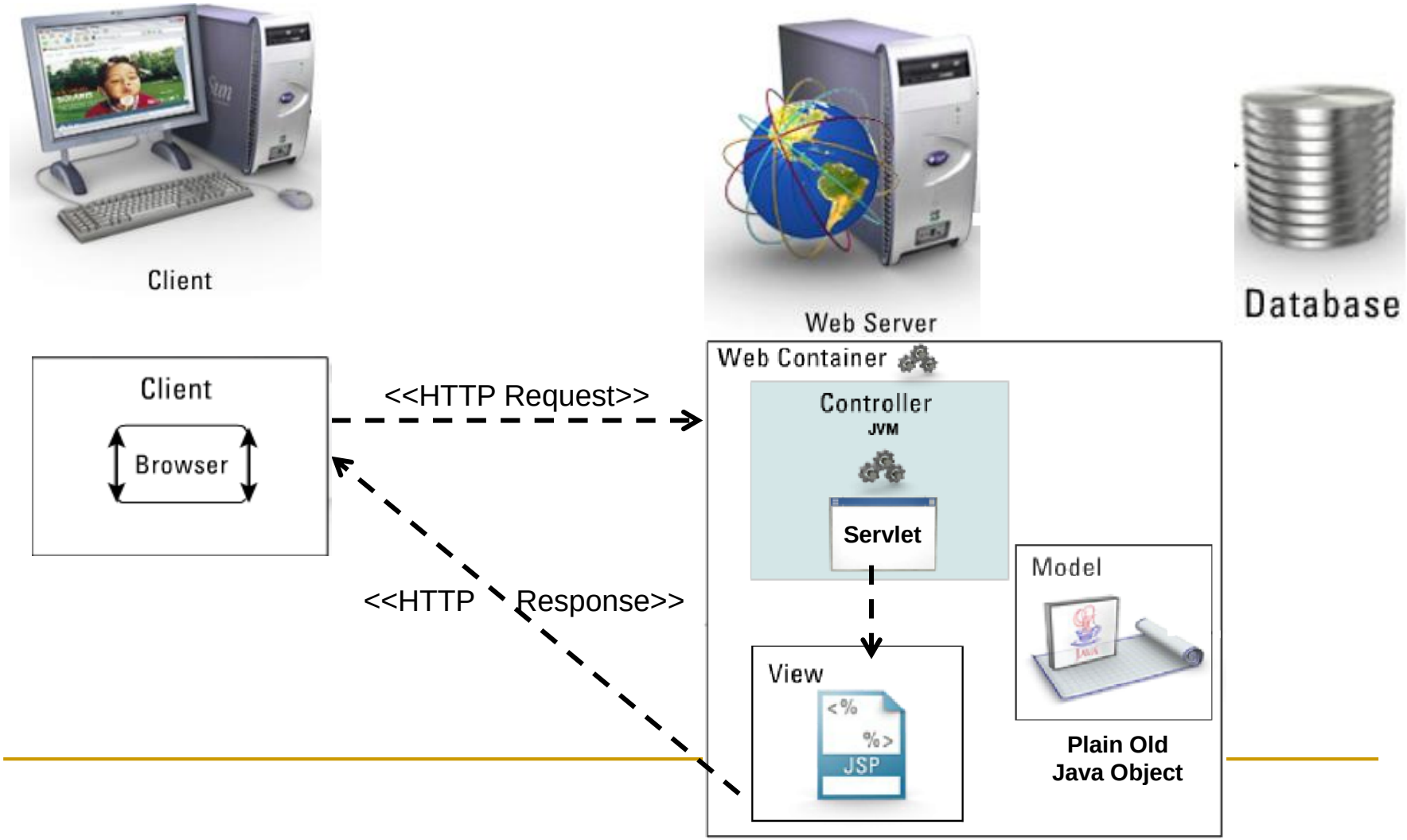



Web Application Robustness
HTML pages
HTML pages
Basic JSP pages and servlets
• HTML pages
• JSP pages
• Servlets
JSP pages with modular components
• HTML pages
• JSP pages
• Servlets
• JavaBeans™ components
• Custom tags
JSP pages with modular components and Enterprise JavaBeans
• HTML pages
• JSP pages
• Servlets
• JavaBeans components
• Custom tags
• Enterprise JavaBeans
Web Application Complexity

課程大綱

1) MVC 設計模式
2) **開發MVC架構程式**
   - **請求轉發**
   - **Controller元件訊息傳遞**
   - **JSP網頁取得訊息**
   - 取得請求參數
   - 執行商業邏輯
3) Soccer 範例

開發MVC架構程式


Client
Web Server
Database
Client
Browser
<<HTTP Request>>
<<HTTP Response>>
Web Container
Controller
JVM
Servlet
Model
View
<%
%>
JSP
Plain Old
Java Object

開發MVC架構程式

- 控制器(Controller)是網路應用程式的接觸窗口
  - 請求參數Parameters取得及驗證
    - 由HTTP Request物件getParameter()相關方法取得
    - 對請求參數執行資料轉換
    - 驗證請求參數
  - 轉發請求給模型(Model)
    - 封裝的商業邏輯
    - 可使用符合Java Beans規範的傳統Java物件(POJO)
    - 選擇適當模型並呼叫其建構子/方法,將請求參數傳入處理
  - 根據執行結果,轉發請求給視圖(view)元件
    - 資料傳遞

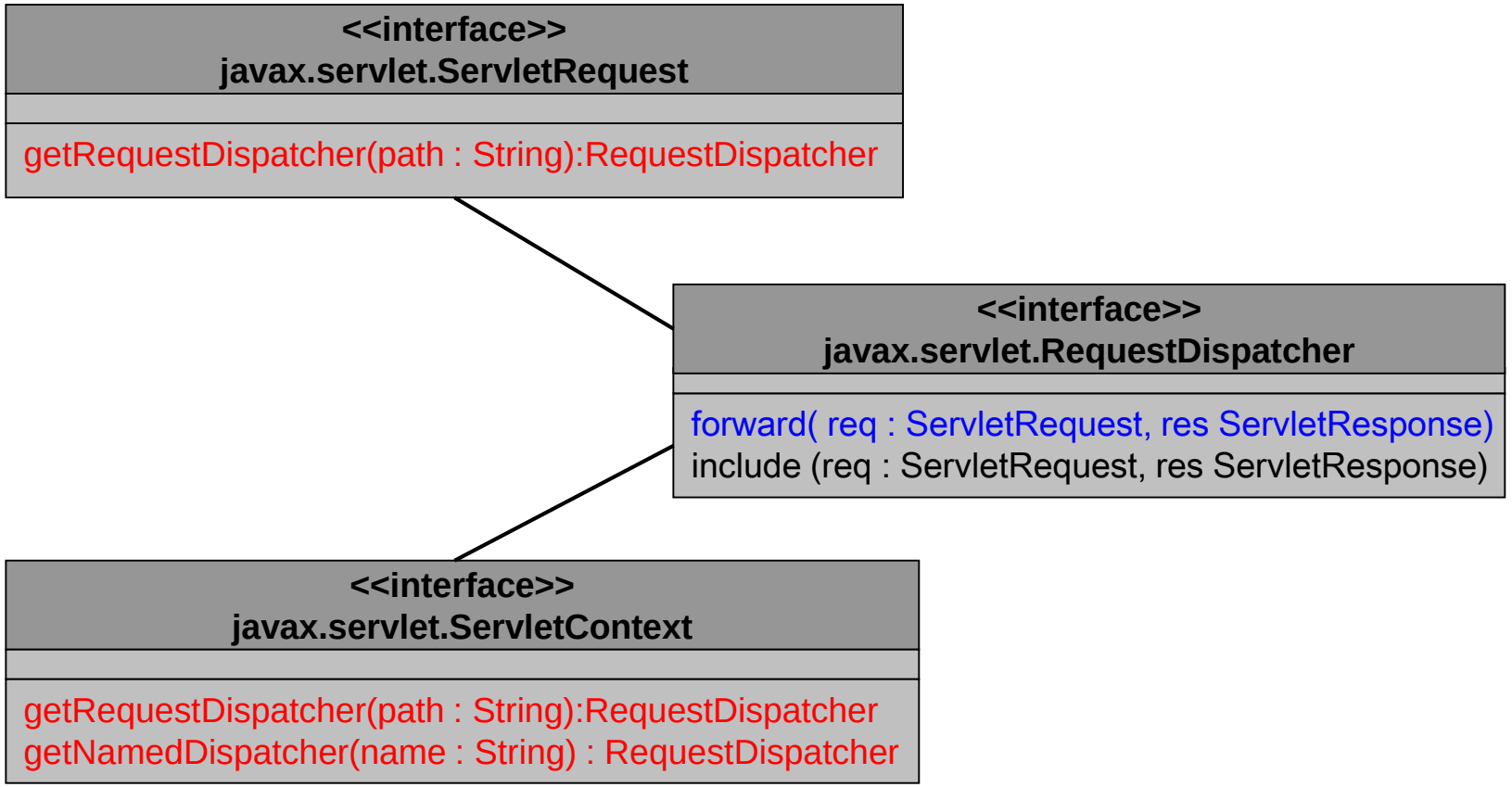
      HTTP Request 物件 setAttribute()相關方法設定
    - 執行頁面轉送

      javax.servlet.RequestDispatcher 物件 forward()方法

開發MVC架構程式

- 視圖(View)元件
  - 接受控制器(Controller)的請求呼叫
    - 根據請求呈現所需的畫面
  - 取得Controller傳遞資料
    - HTTP Request 物件 getAttribute()相關方法取得
    - EL language
  - 由模型提取運算後的結果

RequestDispatcher – 頁面轉送



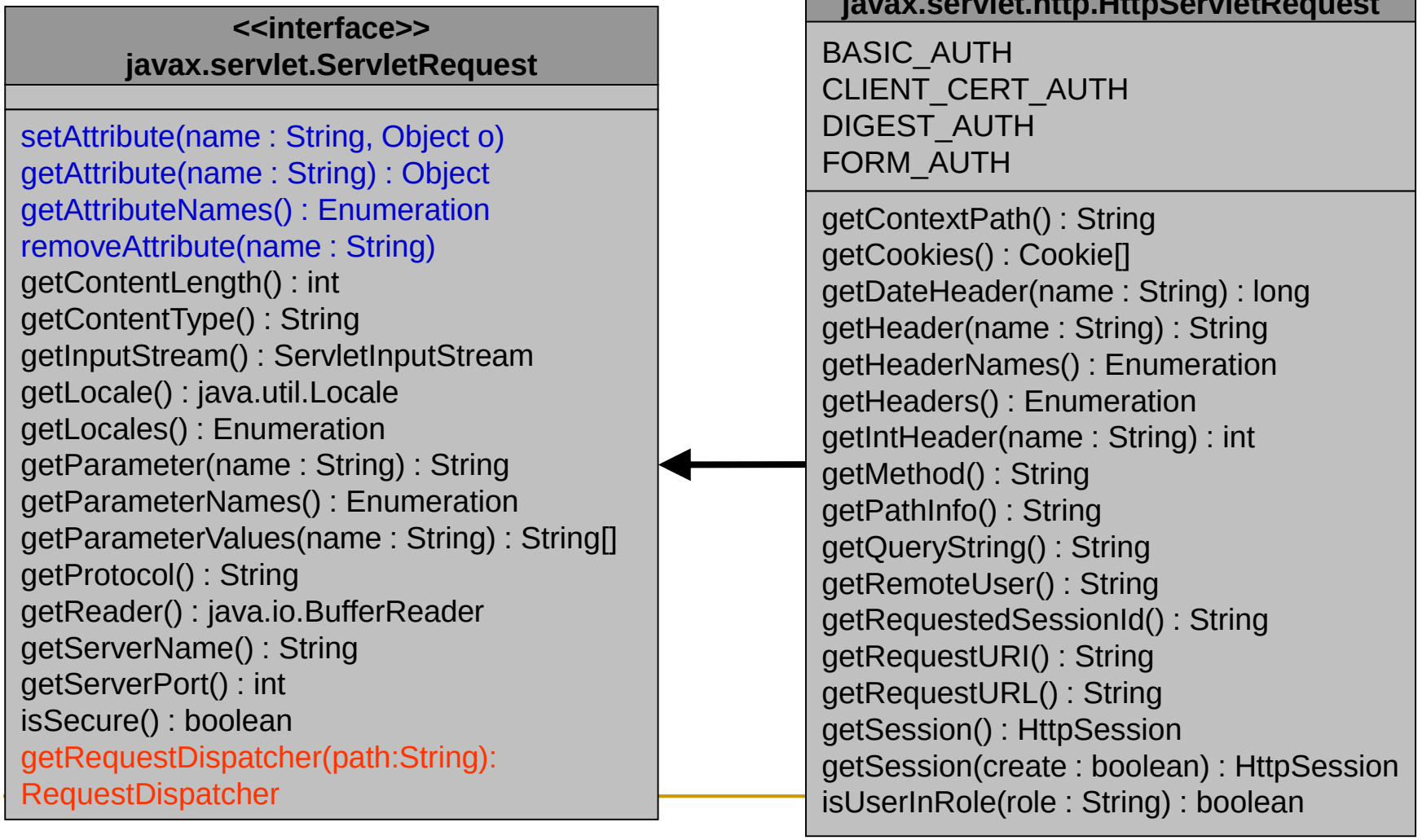
RequestDispatcher

- 注意 URL 規則
  - 絕對路徑
    - 路徑前加 /
    - 由相對根目錄 webapps/<*context*>/ 起算
  - 相對路徑
    - 路徑前不加 /
    - 由正在執行之程式當前路徑起算

Attribute 機制

- Servlet API 提供Attribute機制
  - 讓 Servlet 能夠與轉送的 JSP/Servlet 溝通
    - 同一個 HTTP Request-Response
  - ServletRequest物件中提供一個可讀寫的attribute集合(類似Map型態)
    - ServletRequest物件中parameter集合是唯讀的
  - setAttribute(String key, Object value)
  - getAttribute(String key) : Object

HttpServletRequest – 資料傳遞 Attribute



JSP頁面取得Controller傳遞資料

- JSP頁面取得Controller傳遞資料
  - 使用隱含物件request的getAttribute()方法

```
<%= request.getAttribute("attributeName" ) %>
<%
    Type obj = (Type)request.getAttribute("attributeName");
    …..
 %>
```

  - 使用Expression Language (EL)

```
${attributeName}
```

Expression Language

- 表示式語言 Expression Language (EL)
  - JSTL1.0中為方便存取資料自訂的語言
  - JSP2.0之後,可以在JSP頁面中直接使用

- 優點
  - 存取資料語法簡單,使用方便
  - 未指定變數範圍時,自動尋找變數
  - 自動轉變資料型態

Expression Language

- Expression Language 語法

  ${ *Expression Language* }

  - EL表示式執行的值會併到輸出串流(out)裏

    <h1>Welcome ${name}</h1>

  - 存取HTTP Request 的請求參數

    ${param.parameterName}

  - 存取Java Bean 內的getter方法傳回值

    - 需先將bean設為request Attribute

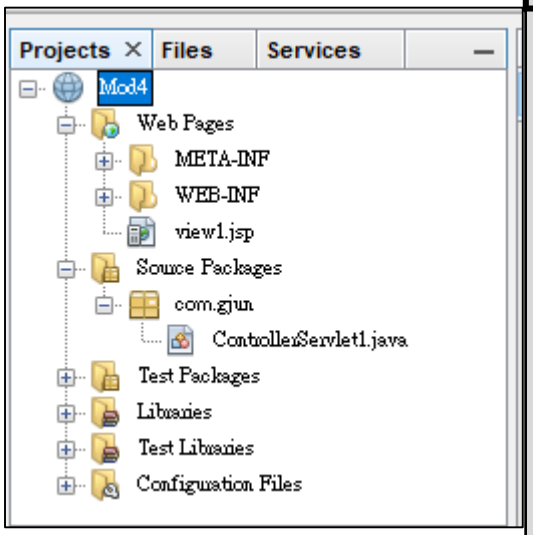
    ${beanName.fieldName}

    ${beanName.methodName}

  - 舊版JSP標準的網路元件容器,EL 功能預設關閉,需使用JSP指令標籤開啟

    <%@ page isELIgnored=“false” %>

開發MVC架構程式範例



ControllerServlet1.java

```
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name="ControllerServlet", urlPatterns={"/ControllerServlet1"})
public class ControllerServlet1 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                                    throws ServletException, IOException {
        RequestDispatcher rd = request.getRequestDispatcher("view1.jsp");
        request.setAttribute("aValue", 99);
        rd.forward(request, response);
    }
}
```

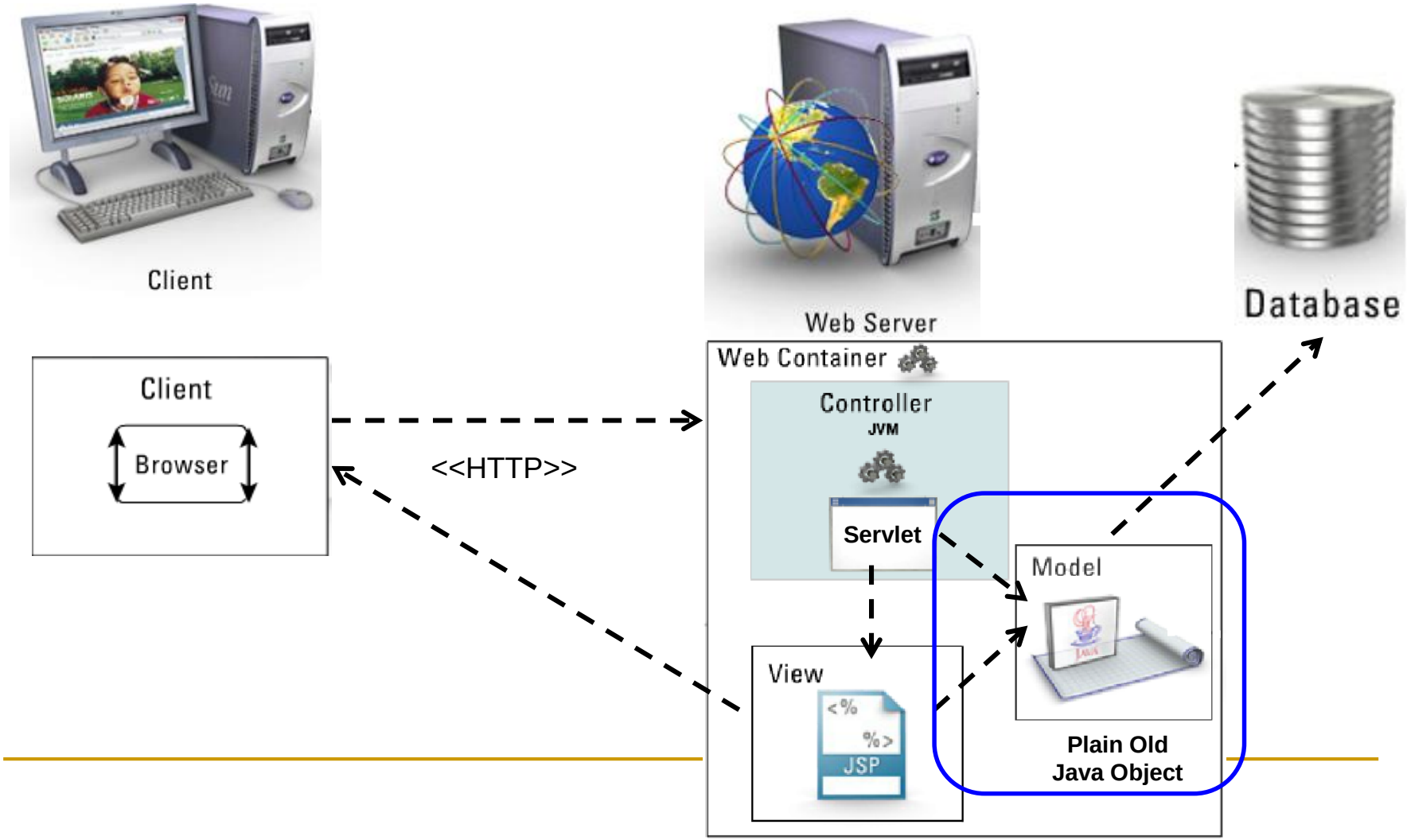
view1.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page isELIgnored="false"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body>
      <h1>Hello World!</h1>
      <p>This page was forwarded to by the ControllerServlet
      along with a data item called "aValue" which had the value
      ${aValue}</p>
   </body>
</html>
```

課程大綱

1) MVC 設計模式
2) **開發MVC架構程式**
   - 請求轉發
   - Controller元件訊息傳遞
   - JSP網頁取得訊息
   - **取得請求參數**
   - **執行商業邏輯**
3) Soccer 範例

完整 MVC 架構程式


Client
Web Server
Database
Client
Browser
<<HTTP>>
Web Container
Controller
JVM
Servlet
Model
View
<%
%>
JSP
Plain Old
Java Object

完整 MVC架構程式

- 控制器(Controller)是網路應用程式的接觸窗口
  - 請求參數Parameters取得及驗證
    - 由HTTP Request物件getParameter()相關方法取得
    - 對請求參數執行資料轉換
    - 驗證請求參數
  - 轉發請求給模型(Model)
    - 封裝的商業邏輯
    - 可使用符合Java Beans規範的傳統Java物件(POJO)
    - 選擇適當模型並呼叫其建構子/方法,將請求參數傳入處理
  - 根據執行結果,轉發請求給視圖(view)元件
    - 資料傳遞

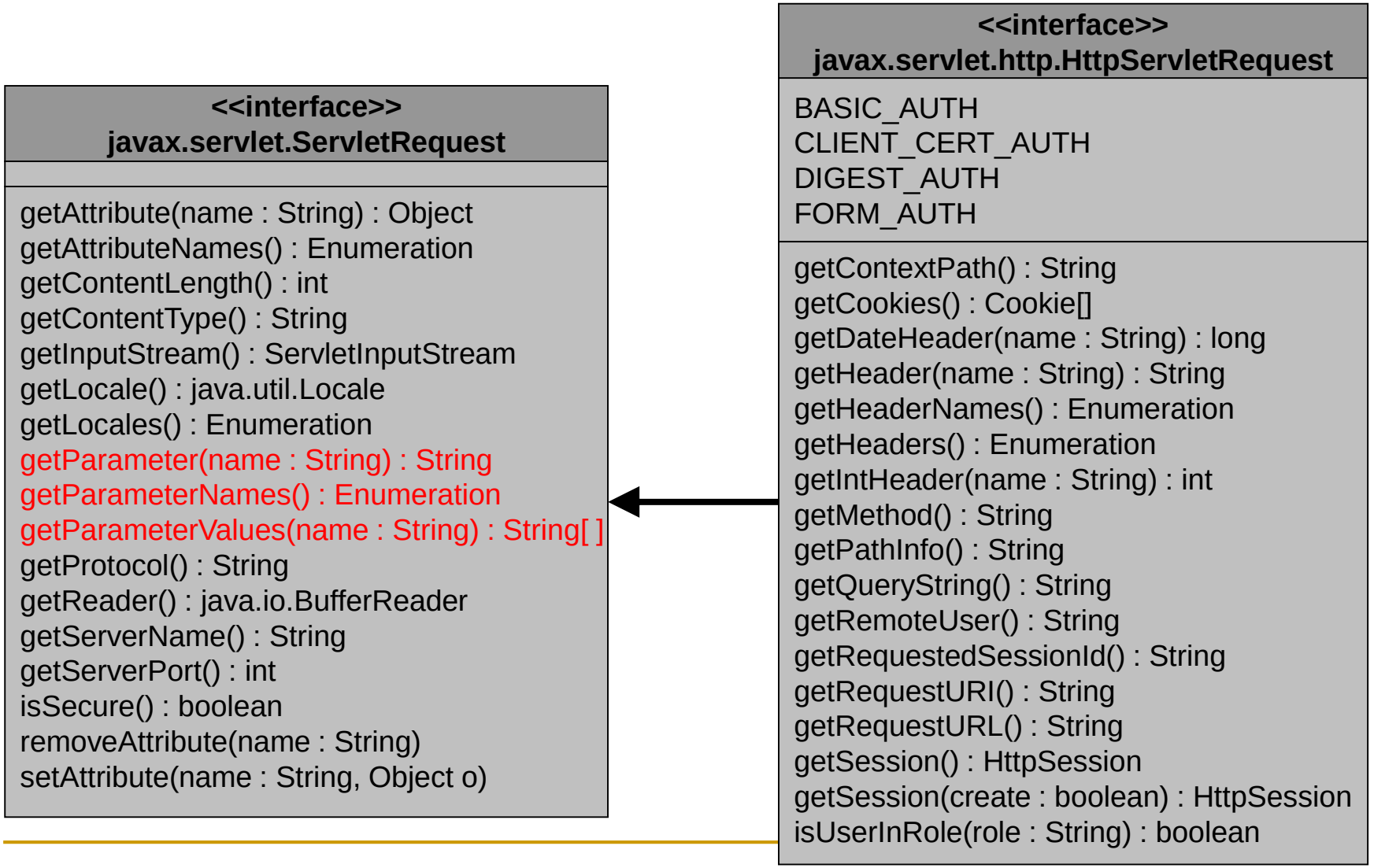
      HTTP Request 物件 setAttribute()相關方法設定
    - 執行頁面轉送

      javax.servlet.RequestDispatcher 物件 forward()方法

完整 MVC架構程式

- 視圖(View)元件
  - 接受控制器(Controller)的請求呼叫
    - 根據請求呈現所需的畫面
  - 取得Controller傳遞資料
    - HTTP Request 物件 getAttribute()相關方法取得
    - EL language
  - 由模型提取運算後的結果

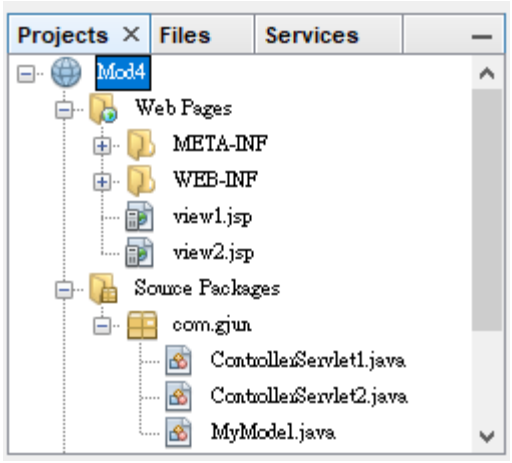
HttpServletRequest 取得表單參數



模型(Model)

- 模型Model
  - 真實世界的商業物件
    - 通常為永續性的資料
    - 包含複雜的物件導向行為
  - 可使用 JavaBeans規範
- JavaBeans 規範
  - 類別需為public
  - 類別需實作 java.io.Serializable
  - 提供無參數建構子no-argument constructor
  - 屬性均不為public
  - 屬性需定義一組存取屬性的方法 getXXX(), setXXX()
    - Read Only屬性：只定義getter
    - Read-Write屬性：定義getter, setter

完整 MVC架構程式範例



MyModel.java

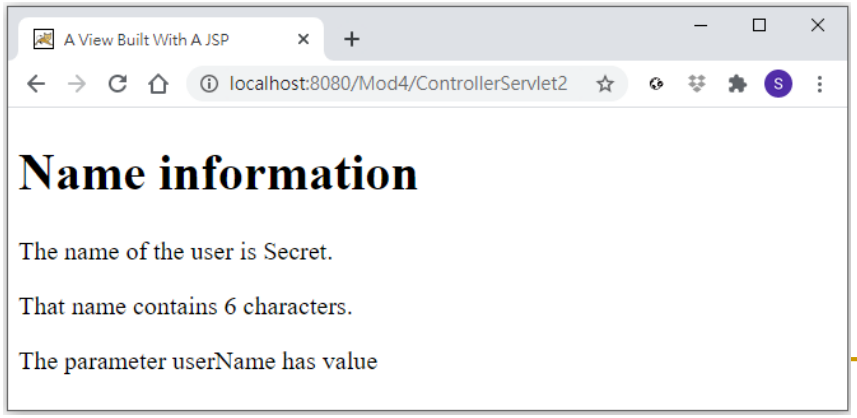
```
public class MyModel {
    private String name;
    public void setName(String s) {
        name = s;
    }
    public String getName() {
        return name;
    }
    public int getNameLength() {
        return name.length();
    }
}
```

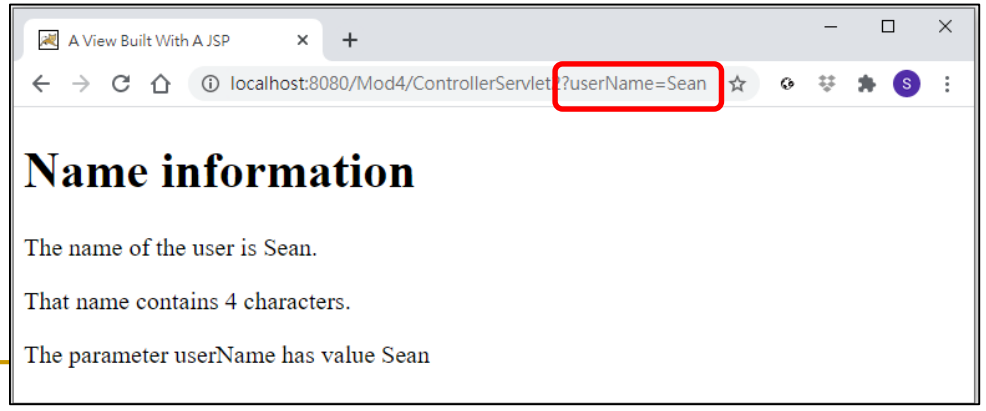
ControllerServlet2.java

```
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name="ControllerServlet2", urlPatterns={"/ControllerServlet2"})
public class FullControllerExample extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
                         response)  throws ServletException, IOException {
        RequestDispatcher rd = request.getRequestDispatcher("view2.jsp");
        MyModel model = new MyModel();
        String name = request.getParameter("userName");
        if (name == null) {     name = "Secret";     }
        model.setName(name);
        request.setAttribute("model", model);
        rd.forward(request, response);
    }
}
```

view2.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page isELIgnored="false"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>A View  Built With A JSP</title>
   </head>
   <body>
      <h1>Name information</h1>
      <p>The name of the user is ${model.name}.</p>
      <p>That name contains ${model.nameLength} characters.</p>
      <p>The parameter userName has value ${param.userName}  </p>
      </p>
   </body>
</html>
```





請求參數取得的設計問題

- 視圖View元件取得請求參數的方法
  - 由request中直接取得
  - 由Controller元件將其設為attribute,由attribute取得
  - 間接由模型取得請求數據
  - 哪一個較好？
- 良好的設計
  - 降低耦合性(Low coupling)
    - 將會獨立改變的事物與無關的東西分開
    - 請求數據直接從視圖中取得會增加元件之間的耦合性

Lab 1

- 使用NetBeans IDE，建立 Anniversary 網路應用專案
- 設計Model類別AnniversaryModel.java
  - 結婚周年與材質名稱對照表如右表
  - 定義適當的屬性及方法
  - 符合JavaBeans規則
- 建立index.html
  - 使用表單Form, 供使用者輸入結婚周年數
  - 將表單內容傳送到AnniversaryController (Servlet)

周年	材質名稱
1	Paper
2	Cotton
3	Leather
4	Linen, Silk
5	Wood
6	Iron
7	Wool, Copper
8	Bronze
9	Pottery, China
10	Tin, Aluminum
11	Steel
12	Silk
13	Lace
14	Ivory
15	Crystal
20	China
25	Silver
30	Pearl
35	Coral
40	Ruby
45	Sapphire
50	Gold
55	Emerald
60	Diamond

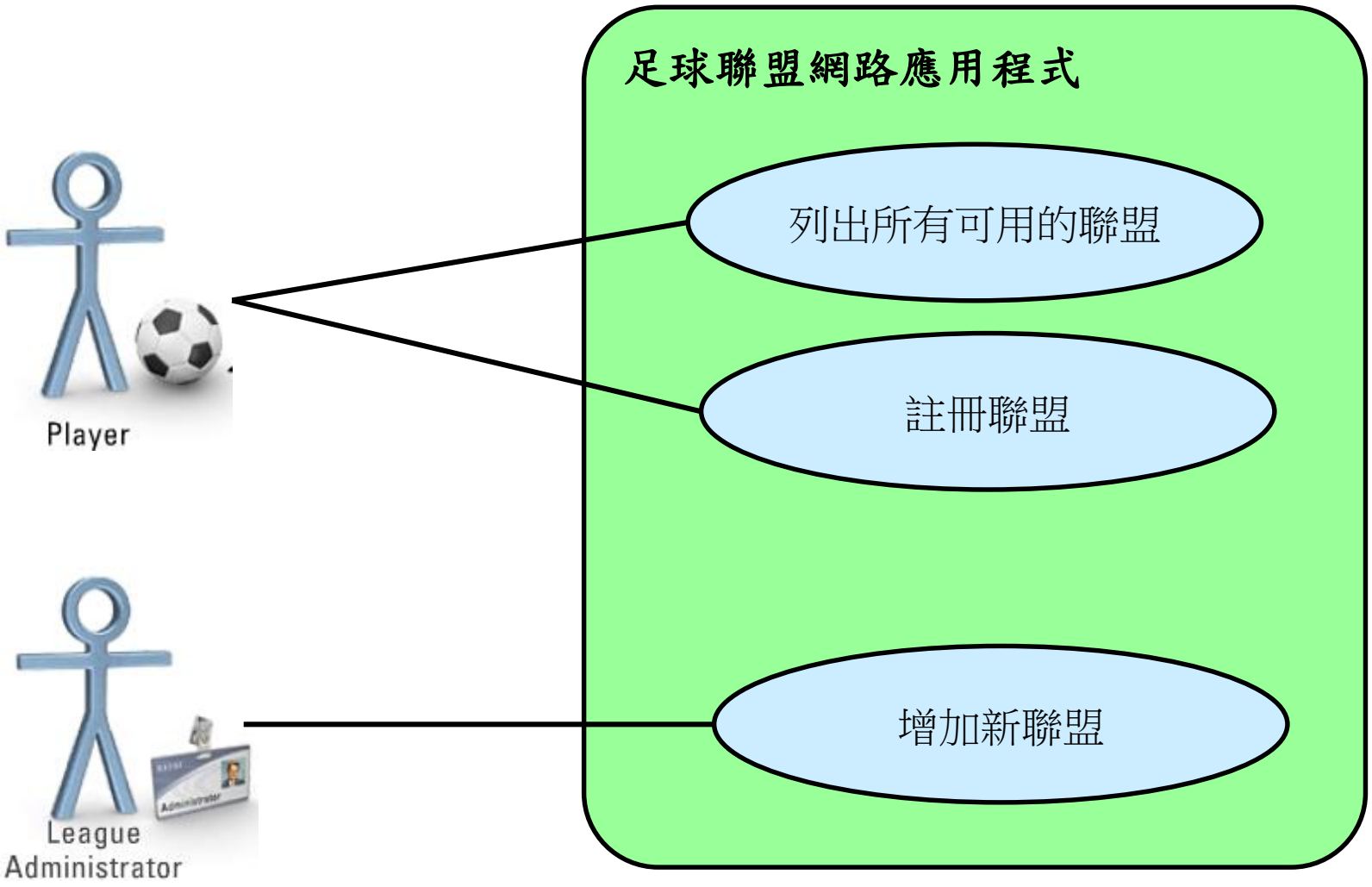
Lab 1

- 建立AnniversaryController.java
  - 建立AnniversaryModel物件
  - 取的結婚周年數請求參數,設定Model物件year 欄位
  - 將Model物件設為請求Attribute
  - 轉送至anniversaryView.jsp
- 建立 anniversaryView.jsp
  - 使用EL, 取得AnniversaryModel的year及material屬性值
  - 顯示於網頁
- 測試、執行

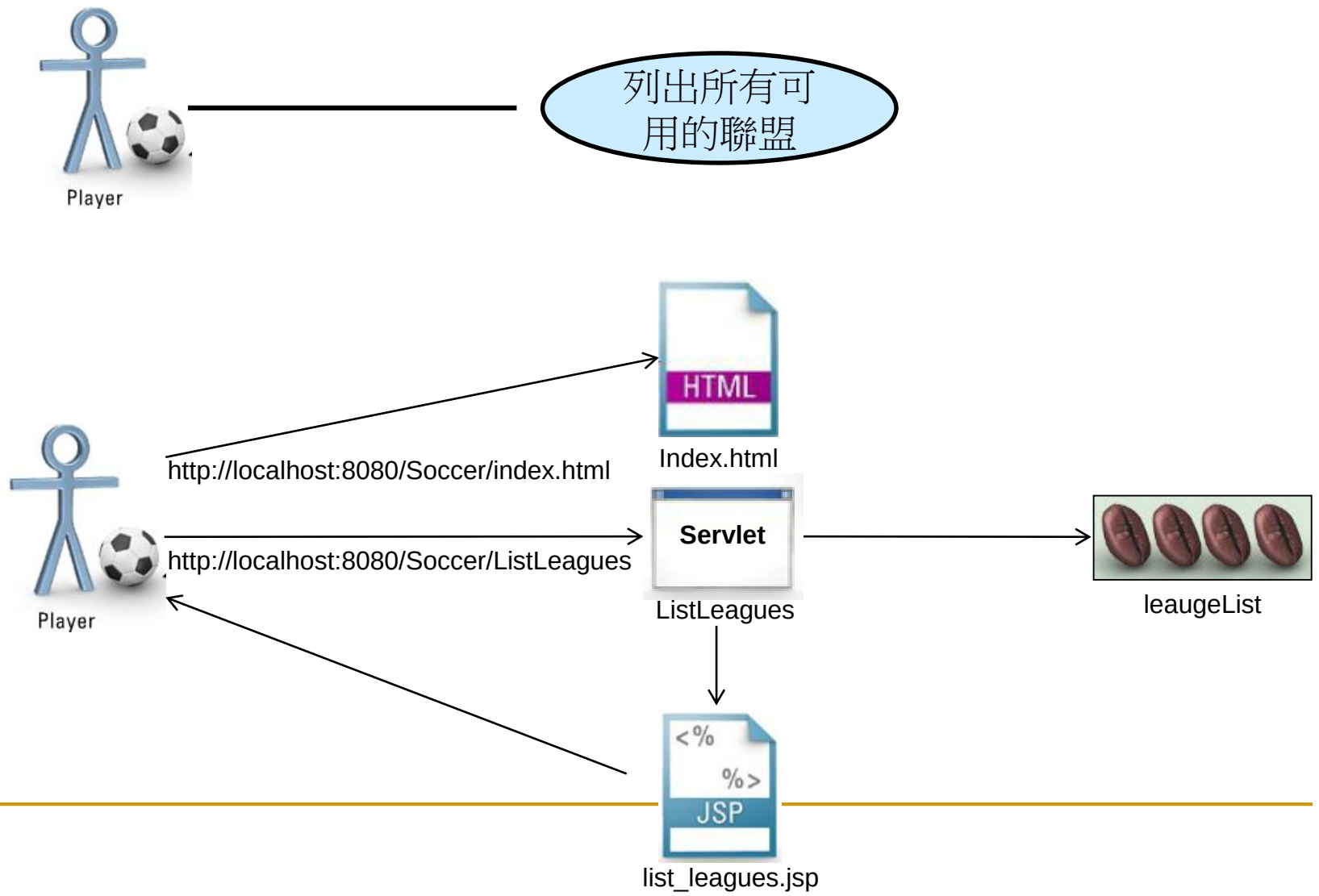
課程大綱

1) MVC 設計模式
2) 開發MVC架構程式
3) **Soccer 範例**

Soccer 範例

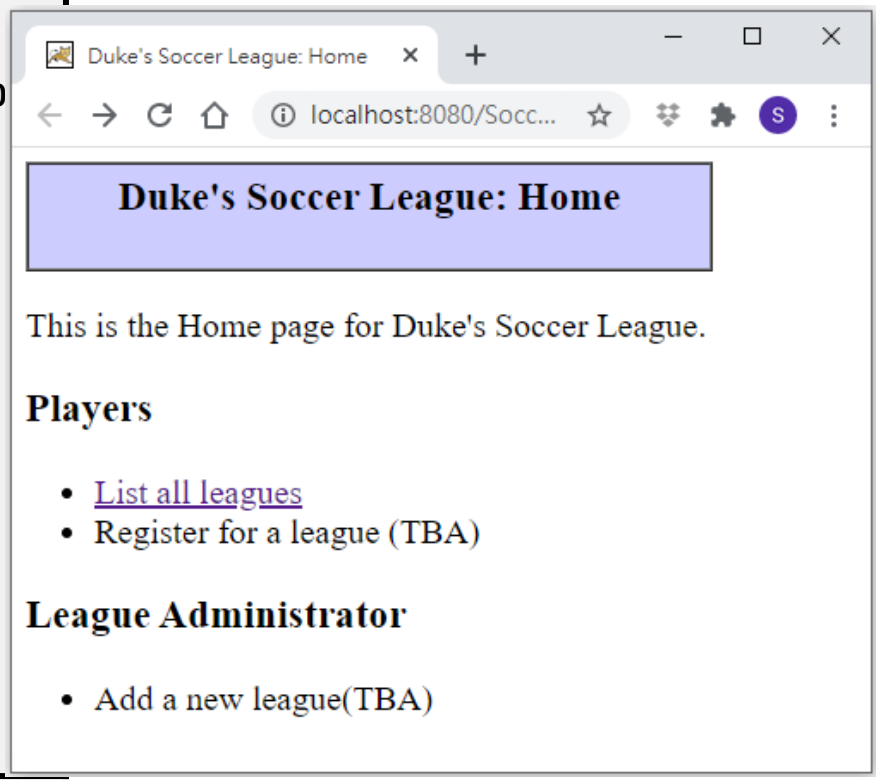

足球聯盟網路應用程式
列出所有可用的聯盟
註冊聯盟
增加新聯盟
Player
League
Administrator

Soccer 範例


Player
列出所有可用的聯盟
HTML
Index.html
http://localhost:8080/Soccer/index.html
Player
Servlet
http://localhost:8080/Soccer/ListLeagues
ListLeagues
leaugeList
<%
%>
JSP
list_leagues.jsp

Soccer 範例

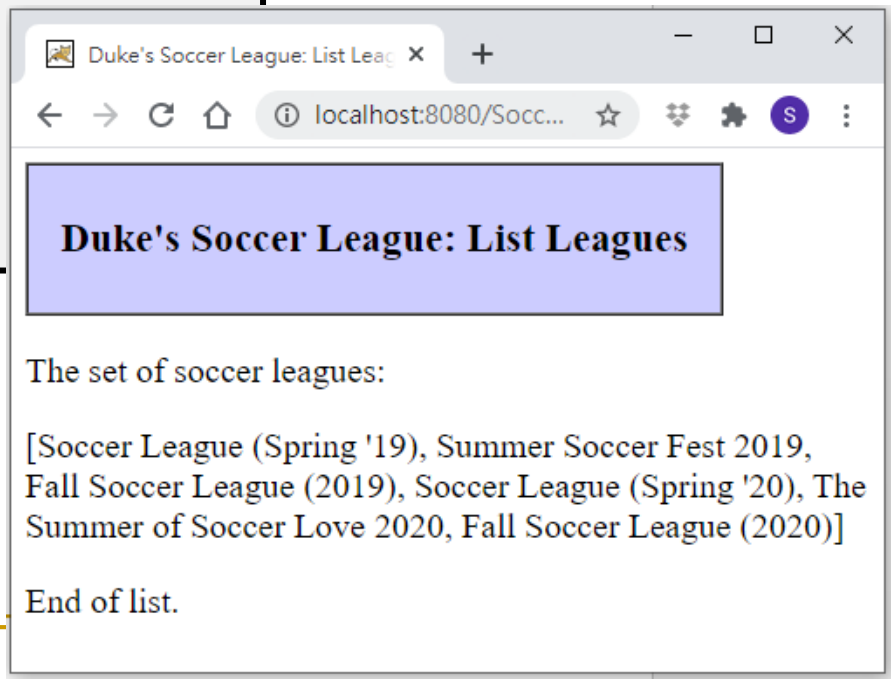
```
<html>
   <head>
      <title>Duke's Soccer League: Home</title>
   </head>
   <body>
      <table border='1' cellpadding='5' cellspacing='0' width='400
         <tr bgcolor='#CCCCFF' align='center'
              valign='center' height='20'>
            <td><h3>Duke's Soccer League: Home</h3></td>
         </tr>
      </table>
      <p>This is the Home page for Duke's Soccer League. </p>
      <h3>Players</h3>
      <ul>
         <li><a href='ListLeagues'>List all leagues</a></li>
         <li>Register for a league (TBA)</li>
      </ul>
      <h3>League Administrator</h3>
      <ul>
         <li>Add a new league(TBA)</a></li>
      </ul>
   </body>
</html>
```



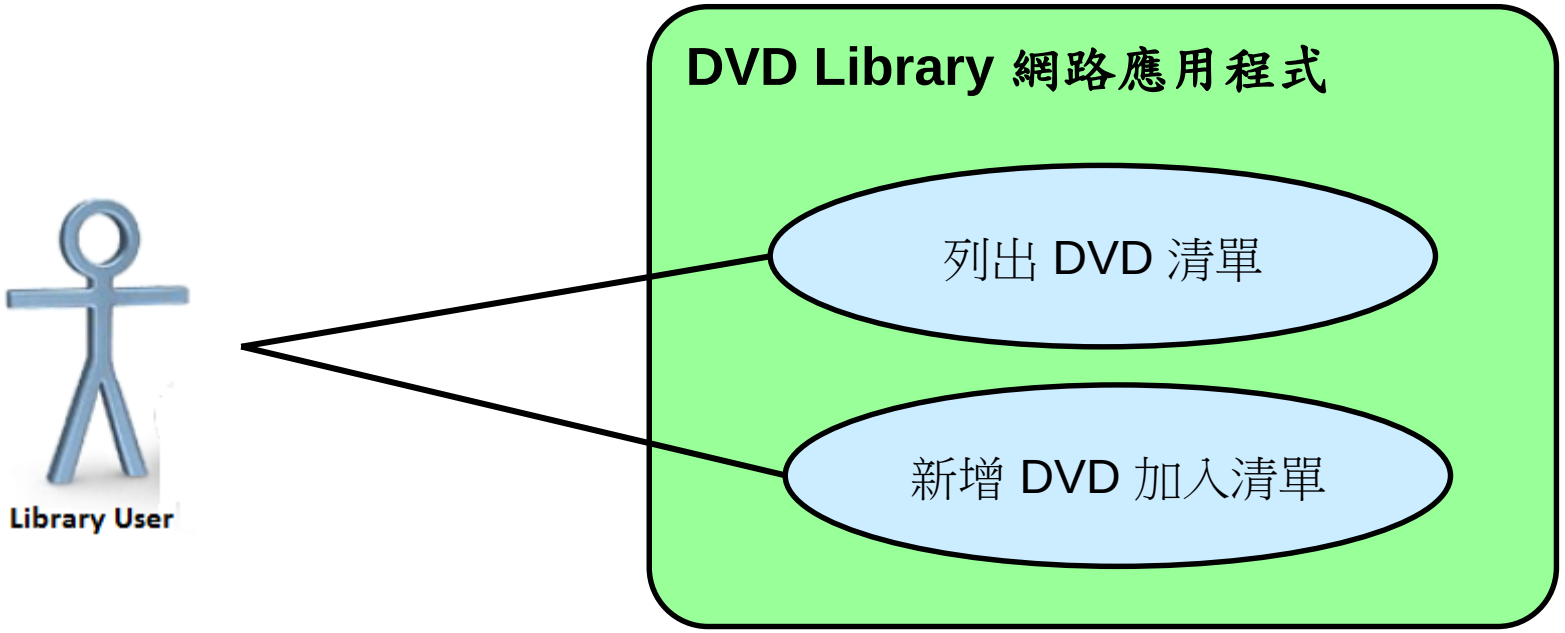
```
package controller;
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
import java.util.*;
import model.*;

@WebServlet(name = "ListLeagues", urlPatterns = {"/ListLeagues"})
public class ListLeagues extends HttpServlet {
   private List leagueList = null;
   protected void processRequest(HttpServletRequest request, HttpServletRespo
                                 response) throws ServletException, IOExcepti
      // Create the set of leagues
      leagueList = new LinkedList();
      leagueList.add( new League(2019, "Spring", "Soccer League (Spring '19)") );
      leagueList.add( new League(2019, "Summer", "Summer Soccer Fest 2019") )
      leagueList.add( new League(2019, "Fall", "Fall Soccer League (2019)") );
      leagueList.add( new League(2020, "Spring", "Soccer League (Spring '20)") );
      leagueList.add( new League(2020, "Summer", "The Summer of Soccer Love
      leagueList.add( new League(2020, "Fall", "Fall Soccer League (2020)") );

      RequestDispatcher rd = request.getRequestDispatcher("list_leagues.jsp");
      request.setAttribute("leagueList", leagueList);
      rd.forward(request, response);
   }
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {  processRequest(request, response);    }
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {  processRequest(request, response);    }
}
```

```
package model;
public class League {
   int year;
   String season;
   String title;
   public League(int year, String season,
                                 String title) {
      this.year = year;
      this.season = season;
      this.title = title;
   }
   public int getYear() {
      return year;
   }
   public String getSeason() {
      return season;
   }
   public String getTitle() {
      return title;
   }
   public String toString() {
      return title;
   }
}
```

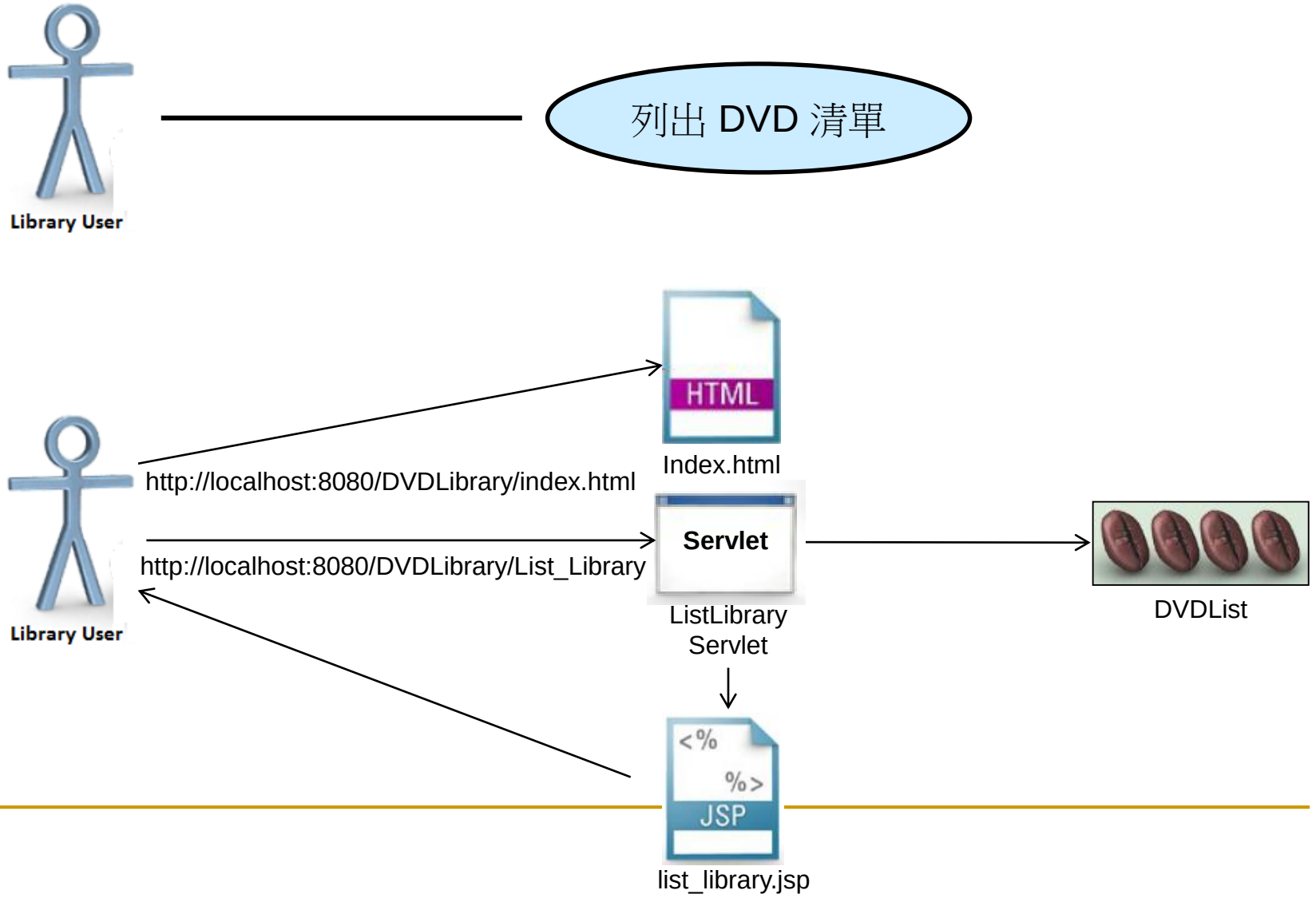
```
<!DOCTYPE html>
<%@ page contentType="text/html"%>
<%@ page import="model.*" %>
<%@ page import="java.util.*" %>

<html>
   <head>
      <title>Duke's Soccer League: List Leagues</title>
   </head>
   <body bgcolor='white'>
      <!-- Page Heading -->
      <table border='1' cellpadding='5' cellspacing='0' width='400'>
         <tr bgcolor='#CCCCFF' align='center' valign='center' height='20'>
            <td><h3>Duke's Soccer League: List Leagues</h3></td>
         </tr>
      </table>
      <p>The set of soccer leagues:</p>
         ${leagueList}
      <p>End of list.</p>
   </body>
</html>
```



Lab2 DVDLibrary 分析設計

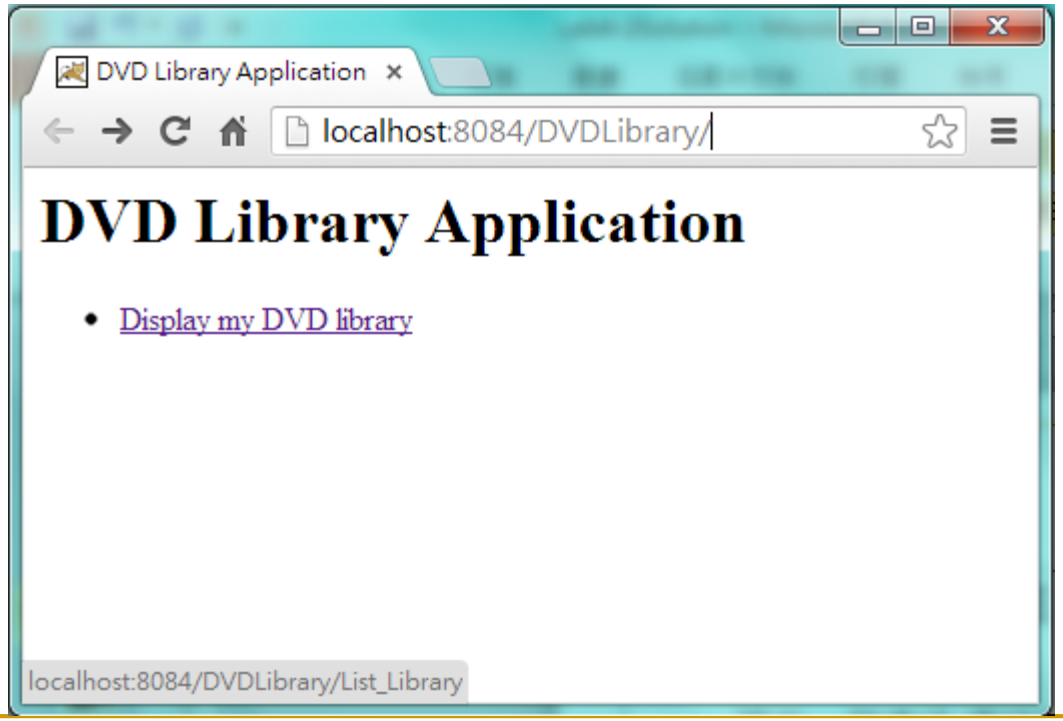


Lab2 DVDLibrary 應用程式



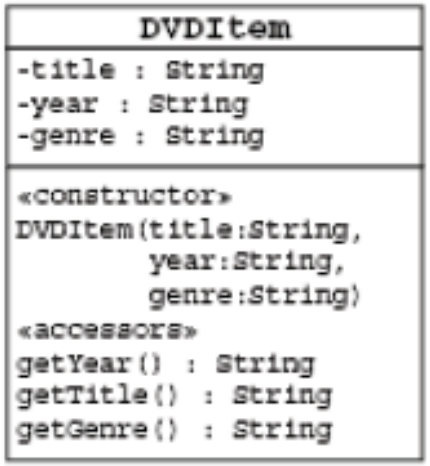
Lab2

- 使用NetBeans IDE，建立DVDLibrary 網路應用專案
- 編輯 index.html
  - 提供一個連結,連結至List_Library
  - 畫面如下



Lab2

- 撰寫model.DVDItem.java
  - 模型物件設計如右：
  - 定義三個String屬性：title, year, genre
  - 提供建構子及屬性 getter

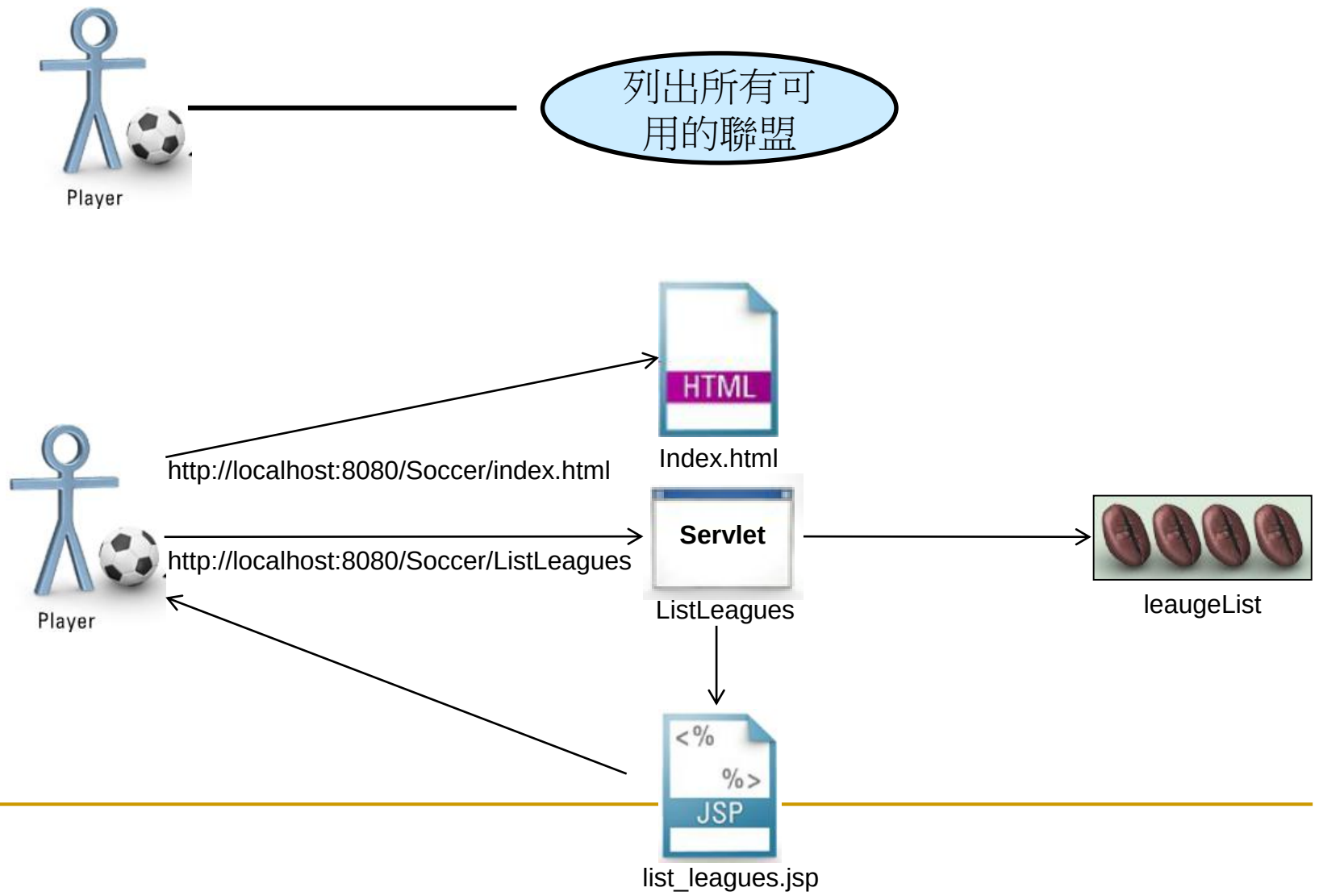


- 撰寫controller.ListLibraryServlet.java
  - Servlet URL 為 List_Library
  - 撰寫 private List<DVDItem> createDvdList() 方法
    - 建立三個DVDItem物件
    - 加入List清單後傳回
  - processRequest()方法中呼叫createDvdList(),取得DVD清單
  - 將DVD清單加入請求(Request)中DVDList屬性(attribute)
  - 請求轉送至list_library.jsp

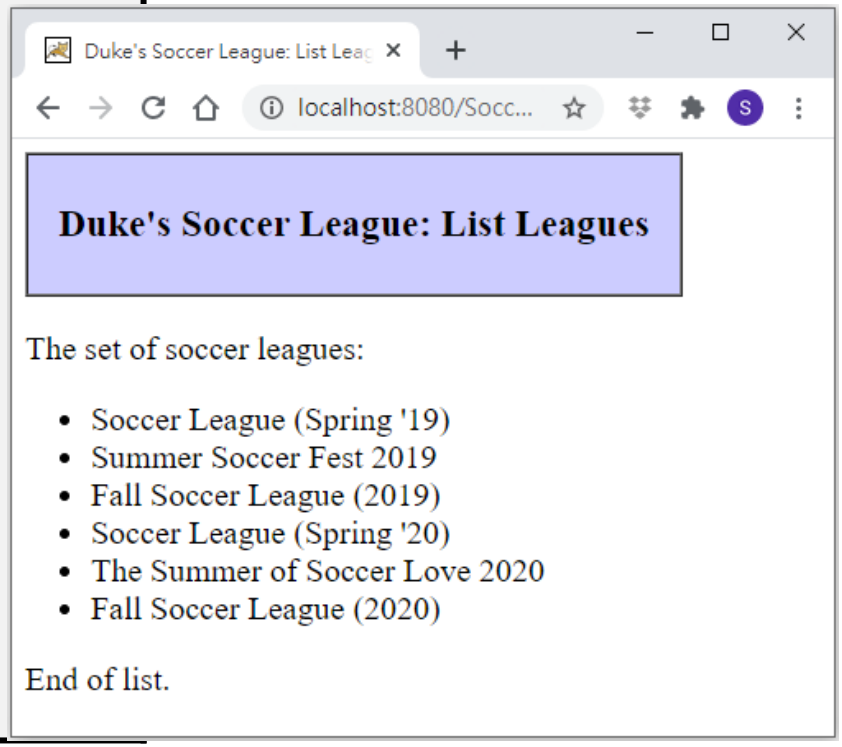
Lab2

- 撰寫list_library.jsp
  - 使用EL取得DVDList屬性值顯示於網頁
- 測試、執行

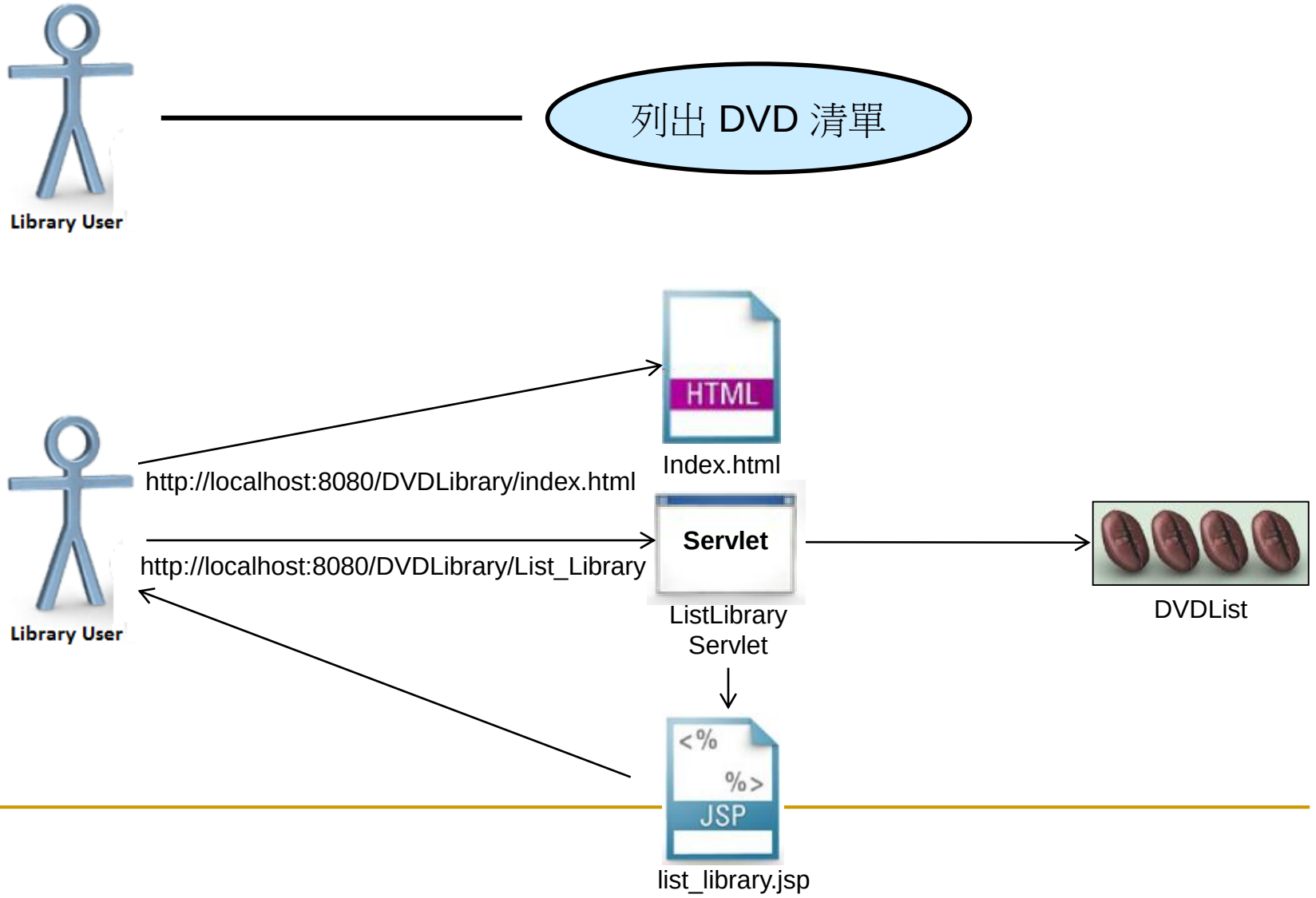
Soccer 範例


Player
列出所有可用的聯盟
HTML
Index.html
http://localhost:8080/Soccer/index.html
Player
Servlet
http://localhost:8080/Soccer/ListLeagues
ListLeagues
leaugeList
<%
%>
JSP
list_leagues.jsp

```jsp
<!DOCTYPE html>
<%@ page contentType="text/html"%>
<%@ page session="false" %>
<%@ page import="model.*" %>
<%@ page import="java.util.*" %>
<%! pirvate String pageTitle ="Duke's Soccer League: List Leagues" %>

<html>
   <head>
      <title><%=pageTitle%></title>
   </head>
   <body bgcolor='white'>
   <!-- Page Heading -->
   <table border='1' cellpadding='5' cellspacing='0' width='400'>
      <tr bgcolor='#CCCCFF' align='center' valign='center' height='20'>
         <td><h3><%=pageTitle%></h3></td>
      </tr>
   </table>
   <p>The set of soccer leagues:</p>
   <ul>
   <%
      List leagueList = (List)(request.getAttribute("leagueList"));
      Iterator items = leagueList.iterator();
      while ( items.hasNext() ) {
         League league = (League) items.next();
   %>
   <li><%= league.getTitle() %></li>
   <%
      }
   %>
   </ul>
   <p>End of list.</p>
   </body>
</html>
```



Lab3 DVDLibrary 應用程式



Lab3

- 改寫 list_library.jsp
  - 使用JSP scripting 元素
  - 取得dvdList屬性
  - 將DVD清單資訊顯示於網頁如下
- 測試、執行