Introduction to JavaServer Pages(JSP)

html

- <html>
- <head>
- <meta http-equiv="Content-Type" content="text/html; charset=BIG5">
- <title>Insert title here</title>
- </head>
- <body>
- Hello! World!
- </body>
- </html>

Servlet

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloWorld extends HttpServlet {
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
PrintWriter out = resp.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>Hello Servlet</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1> Hello! World!</h1>");
out.println("</body>");
out.println("</html>");
out.close();
}
}
```

JSP

- <%@ page language="java" contentType="text/html; charset=BIG5" pageEncoding="BIG5"%>
  <!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=BIG5">
- <title>Insert title here</title>
- </head>
- <body>
- <% out.println("Hello! World!"); %>
- </body>
- </html>

JSP

- <%@ page language="java" contentType="text/html; charset=BIG5" pageEncoding="BIG5"%>
  <!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=BIG5">
- <title>Insert title here</title>
- </head>
- <body>
- Hello! World!
- </body>
- </html>

JSP 的構成元件

- JSP 網頁的 Elements 部分有以下三種元件：
  - ✓ Directives Elements （指令元件）
  - ✓ Action Elements （動作元件）
  - ✓ Scripting Elements （描述語言元件）

指令元件（Directives）
- 簡介指令元件

- JSP 指令元件用於指定 JSP 網頁有關輸出方式、引用套件、載入檔案...等相關設定。

- 指令元件並不會輸出任何資料至前端，且有效範圍僅限於使用該指令的 JSP 網頁，指令元件共有以下三種:
  - ✓ 網頁指令（The page directive）
  - ✓ 載入指令（The include directive）
  - ✓ 標籤資料庫指令（The taglib directive）

指令元件（Directives）
- 簡介指令元件

- 指令元件的設定語法如下:

  <%@ 指令名稱 指令**1 =** 值**,** 指令**2 =** 值**,…**
  %>

  由於設定值均為字串，設定時必須運用『"』標示。

指令名稱	意義
page	網頁指令
include	載入指令
taglib	標籤資料庫指令

指令元件（Directives）
- 網頁指令

- 網頁指令的設定語法如下:

  <%@ page 屬性 1 = 值 屬性 2 **=** 值 **... %>**

- language
  - ✓ 定義 JSP 網頁所使用的描述語言
  - ✓ 若所使用的 JSP 引擎支援 Java 以外的語言時，可使用此指令指定 JSP 網頁使用的語言，語法如下:

    <%@ page language=" 描述語言 **"%>**

指令元件（Directives）
- 網頁指令

- extends
  - ✓ 指定 JSP 網頁編譯後產生的 Servlet，應延伸哪一個超類別（亦稱父類別）語法如下:

    <%@ page extends=" 父類別名稱 **"%>**
  - ✓ 以下敘述設定 Servlet 衍生於 HttpServlet 類別:

    <%@ page extends="HttpServlet"%>

指令元件（Directives）
- 網頁指令

- import
  - ✓ 指定 JSP 網頁所使用的 Java 套件，語法如下：

    <%@ page import=" 套件 **1,** 套件 **2, ..."%>**
  - ✓ import 指令是最常用的網頁指令，也是唯一可以多次設定的指令，且累加每個設定。
  - ✓ 以下語法設定使用 java.io 套件的所有類別，與 java.util 套件的 Date 類別：

    <%@ page import="java.io.*,

    java.util.Date"%>

指令元件（Directives）
- 網頁指令

- session
  - ✓ 設定此網頁被瀏覽時，是否可使用代表使用者連線的 session 物件，語法如下：

    <%@ page session="true|false"%>

  - ✓ 預設值為 true

指令元件（Directives）
- 網頁指令

- buffer
  - ✓ 設定此網頁輸出時，是否使用緩衝區，語法如下：

    <%@ page buffer="none| 緩衝區大小 **kb"%>**

  - ✓ JSP 網頁所使用緩衝區大小的設定，將因 JSP 容器的不同而有所不同。

指令元件（Directives）
- 網頁指令

- Resin 伺服器的 JSP 容器，預設使用的緩衝區大小為 8 kb
- 欲自行設定時，緩衝區大小必須是 8kb 的倍數
- 若設定為 none，表示輸出 JSP 網頁時，將不使用緩衝區
- 以下敘述將設定使用 16 kb 緩衝區

<%@ page buffer="16kb"%>

指令元件（Directives）
- 網頁指令

- autoFlush
  - ✓ 設定當緩衝區滿了後，是否自動輸出緩衝區內的資料，語法如下：

    <%@ page autoFlush="true|false"%>
  - ✓ 預設值為 true

指令元件（Directives）
- 網頁指令

- isThreadSafe
  - ✓ 定義是否以執行緒方式回應瀏覽此 JSP 網頁的要求訊息，語法如下：

    <%@ page isThreadSafe ="true|false"%>

  - ✓ 預設值為 true
  - ✓ 若設定為 false，回應 JSP 網頁時，將產生新的行程

指令元件（Directives）
- 網頁指令

- info
  - ✓ 此網頁的說明資訊，設定語法如下：

    <%@ page info=" 網頁說明資訊 **"%>**
  - ✓ 預設狀況為略過此屬性
  - ✓ 使用此屬性時，在 JSP 網頁中，呼叫 Servlet 類別的 getServletInfo 方法將可取得此資訊，語法如下：

    <%= getServletInfo() %>

指令元件（Directives）
- 網頁指令

- errorPage
  - ✓ 設定執行此 JSP 網頁時，若發生錯誤，顯示錯誤訊息之網頁的 URL 路徑，設定語法如下：

    <%@ page errorPage=“ 錯誤訊息網頁的 **URL** 路徑 **"%>**
  - ✓ 在設定 errorPage 屬性的網頁中，錯誤訊息將以 throw 敘述丟出。
  - ✓ 而被設定為錯誤訊息網頁的 JSP 網頁，將利用 exception 隱含物件取得錯誤訊息。

指令元件（Directives）
- 網頁指令

- isErrorPage
  - ✓ 設定此 JSP 網頁是否為錯誤訊息網頁，設定語法如下:

    <%@ page isErrorPage=
    "true|false"%>

  - ✓ 預設值為 false

指令元件（Directives）
- 網頁指令

- 當設定為 true 時， JSP 網頁將可存取隱含的 exception 物件，並透過該物件取得從發生錯誤之網頁，所傳出的錯誤訊息。
- 取得錯誤訊息的語法如下：

  <%= exception.getMessage() %>

  或

  <%= exception.toString() %>

指令元件（Directives）
- 網頁指令

- contentType
  - ✓ 設定 JSP 網頁輸出資料時，所使用的字元壓縮方式，以及所使用的字元集。
  - ✓ 當撰寫中文網頁時，必須運用以下語法，設定 contentType 屬性：

    <%@ page contentType="text/html; charset=Big5"%>
  - ✓ 此屬性的預設值為『 text/html; charset= ISO-8859-1 』

指令元件（Directives）
- 載入指令

- 載入指令用於將某檔案載入網頁，載入指令的設定語法如下:

  <%@ include file = " 檔案名稱 **"%>**

  以下敘述將載入 heading.inc 檔:

  <%@ include file = "heading.inc"%>

指令元件（Directives）
- 標籤資料庫指令

- 標籤資料庫是由使用者自行定義的網頁標籤。當欲使用自訂的網頁標籤時，您必須在 JSP 網頁中，指定欲載入標籤資料的 URI 位置，譯為統一資源識別子），並定義標籤的前置標記，語法如下：

  <%@ taglib uri = " 標籤資料庫的 **URI "**

  prefix= " 前置標記 **"%>**

動作元件

- 簡介動作元件

- 動作元件用於執行一些標準常用的 JSP 網頁動作，例如：將網頁轉向、使用 Java Bean 、設定 Java Bean 的屬性等。

動作元件

- 簡介動作元件

- 在 JSP 中，動作元件共有以下幾種:
  - ✓ <jsp:useBean>
  - ✓ <jsp:setProperty>
  - ✓ <jsp:getProperty>
  - ✓ <jsp:param>
  - ✓ <jsp:include>
  - ✓ <jsp:forward>
  - ✓ <jsp:plugin>

動作元件 - <jsp:useBean>、<jsp:setProperty> 與 <jsp:getProperty>

- <jsp:useBean>

  此動作元件用於宣告 JSP 網頁中，欲使用的 JavaBean 物件。

- <jsp:setProperty>

  此動作元件在 JSP 網頁中，將用於設定所使用 JavaBean 物件的屬性。

動作元件 - <jsp:useBean>、<jsp:setProperty> 與 <jsp:getProperty>

- <jsp:getProperty>
  此動作元件在 JSP 網頁中，將用於取得所使用 JavaBean 物件的屬性。

動作元件
- <jsp： param>

- <jsp:param> 動作用於傳送參數，必須配合 <jsp:include> 、 <jsp:forward> 與 <jsp:plugin> 動作一起使用。

- 使用語法如下:

  <jsp:param name = 參數名稱 **value =** 值 **/>**

動作元件

- <jsp： include> 與 <jsp：forward>

<jsp:include>

- <jsp:include> 動作用於動態載入 HTML 網頁或者 JSP 網頁，語法如下：

<jsp:include page = 網頁名稱 **>**
<jsp:param name = 參數名稱 **1 value =** 值 **1 />**
<jsp:param name = 參數名稱 **2 value =** 值 **2 />**
……
<jsp:include/>

動作元件

- <jsp： include> 與 <jsp：forward>

- 若不傳遞參數時，則語法如下:

  <jsp:include page = 網頁名稱 **/>**

- 當被載入網頁與載入網頁不在同一個資料夾內時，則可用相對路徑或絕對路徑的方式指定網頁位置。

- 當載入 HTML 網頁時，並不需要傳入參數。

動作元件

- <jsp： include> 與 <jsp：forward>

- 對於載入 JSP 網頁的檔案，副檔名不建議使用 .jsp，可以使用 .jspf、.jsf、.inc... 等自訂名稱。
- 若使用 <jsp:param> 將參數傳遞給 JSP 網頁時，在 JSP 網頁中，將可利用下面的語法取得傳入之參數。

request.getParameter(" 參數名稱 ");

動作元件

- <jsp： include> 與 <jsp：forward>

<jsp:forward>

- 動作用於將瀏覽器顯示的網頁，導向至另一個 HTML 網頁或者 JSP 網頁，語法如下：

```
<jsp:forward page = " 網頁名稱 ">
    <jsp:param name = " 參數名稱 1" value = " 值 1" />
    <jsp:param name = " 參數名稱 2" value = " 值 2" />
    ……
<jsp:forward/>
```

動作元件

- <jsp： include> 與 <jsp：forward>

- 若不傳遞參數時，則語法如下:

  <jsp:forward page = 網頁名稱 **/>**

- 當欲導向至的目的網頁與進行導向動作之網頁，兩者不在同一個資料夾內時，則可用相對路徑或絕對路徑的方式指定目的網頁的位置。

動作元件

- <jsp： include> 與 <jsp：forward>

- 此外，當目的網頁為 HTML 網頁時，並不需要傳入參數。
- 若使用 <jsp:param> 將參數傳遞給 JSP 網頁時，在 JSP 網頁中將可利用下面的語法取得傳入的參數。

request.getParameter(" 參數名稱 ");

動作元件

- <jsp： plugin>

- <jsp:plugin> 動作用於載入 Java Applet 或者 Java Bean，用途與 HTML 語法中的 <Applet> 及 <Object> 標籤相同。語法如下:

  <jsp:plugin type =“plugin 類型” **code=“** 儲存類別的檔案名稱 **"**

  codebase=" 類別路徑 **" {align="** 對齊方式 **"}**

  {archive=" 相關檔案路徑 **"} {height="** 高度 **"}**

  {width=" 寬度 **"}** **{hspace="** 水平間距 **"}**

  {vspace=" 垂直間距 **"} {jrevesion="Java** 環境版本 **"}**

  {name=" 物件名稱 **"}**

動作元件

- <jsp： plugin>

```
{nspluginurl=" 供 NC 使用的 plugin 載入位置 "}
{iepluginurl=" 供 ie 使用的 plugin 載入位置 "} >
{
   <jsp:params>
     <jsp:param name =  參數名稱 1 value =  值 1 />
     <jsp:param name =  參數名稱 2 value =  值 2 />
     ……
   <jsp:params/>
}
{  <jsp:fallback> 錯誤訊息 <jsp:frallback/> }
<jsp:plugin/>
```

描述語言元件

- 簡介描述語言元件

- 描述語言元件是 JSP 網頁中，主要撰寫程式碼的部份，也是 JSP 網頁執行後，輸出大部份資料的部份，此元件基本語法為:

  <% 程式碼 **%>**

- 描述語言元件主要有三種:
  - ✓ Declarations （宣告敘述）
  - ✓ Scriptlets （程式碼區段）
  - ✓ Expressions （表示式）

描述語言元件

- 宣告敘述

- 宣告敘述用於宣告 JSP 網頁內的變數與函數，這些經過宣告的變數與函數，將成為 Servlet 類別的屬性與方法，宣告敘述的語法如下：

```
<%!
    ..... // 宣告敘述
%>
```

描述語言元件

- 小文稿元件 (Scriplet)

- JSP 網頁中，大多數的程式碼均撰寫於小文稿元件裡，定義小文稿元件時，將運用『 <% 』與『 %> 』標記標示，語法如下：

```
<%
    ..... // 程式碼
%>
```

描述語言元件

- 小文稿元件

- 在小文稿元件內，除了可以運用 Java 語言的註解方式，撰寫程式註解，還可以運用以下語法，建立獨立的註解說明:

  <%-- 註解 **--%>**

- 若欲建立可出現在瀏覽端的註解時，須運用 HTML 語法的註解方式，如下所述:

  <!-- 註解 **-->**

描述語言元件
- 表示式敘述

- 表示式是一個簡化的 out.println 敘述，其語法如下：

  <%= 欲輸出資料 **%>**

  請注意，欲輸出資料僅能有一行，且結尾不需要加上『;』。。

JSP 網頁的隱含物件

- 在 JSP 網頁中，有一些已經完成定義的物件，稱之為隱含物件。
- 使用這些物件時，可以不經過宣告，即可使用，例如前面我們所使用的 out 物件，就是一個隱含物件。

JSP 網頁的隱含物件

- 處理資料輸出 / 輸入的隱含物件:

隱含物件	用途	有效範圍	對應之 Servlet
out	標準輸出物件，用於將資料輸出至回應客戶端之資料流	page	javax.servlet.JspWriter
response	用於設定 JSP 回應客戶端資料流的物件	page	javax.servlet.HttpServletResponse
request	取得客戶端資訊的物件	request	javax.servlet.HttpServletRequest

JSP 網頁的隱含物件

- 處理執行時期共用資料的隱含物件:

隱含物件	用途	有效範圍	對應之 Servlet
application	提供 JSP 網頁執行時的重要資料	application	javax.servlet.http.ServletContext
pageContext	用於存取 JSP 網頁於執行時期，所需使用的屬性與方法	page	javax.servlet.jsp.PageContext
session	同一連線過程中產生的 session 資料	session	javax.servlet.jsp.HttpSession

JSP 網頁的隱含物件

- 處理錯誤的 Exception 隱含物件：

  Exception 隱含物件用於處理 JSP 網頁執行發生錯誤時，所丟出的例外物件。

- 取得編譯 JSP 網頁產生之 Servlet 類別相關資訊

包含 [illegible] 下表含納各隱含物件的用途：

隱含物件	用途	有效範圍	對應之 Servlet
config	JSP 的設定資源	page	javax.servlet.ServletConfig
page	代表目前網頁，相當於 Java 中的 this 物件。	page	java.lang.Object

屬性與方法的宣告
- JSP 網頁與物件導向觀念

- 單從 JSP 網頁來看，必須懂得物件導向觀念，才能使用各種隱含物件，以及運用套件中的類別。
- 實際上，當 JSP 網頁編譯成 Servlet 後，一個 JSP 網頁就是一個 Servlet 的類別。

屬性與方法的宣告
- JSP 網頁與物件導向觀念

- 當然在 JSP 網頁裡，可以為這個 Servlet 類別定義屬性與方法。
- 只是在 JSP 網頁中，並看不出所定義的屬性與方法，具備了物件導向觀念中屬性與方法的意義。
- 使用上也是這樣，屬性將成為該 JSP 網頁的全域變數，且存取該 JSP 網頁之連線將共用，方法則被當成為函數。

屬性與方法的宣告

- 函數（方法）的定義

- 在 JSP 網頁中，利用『 <%! …… %> 』敘述宣告函數時，該函數將成為 servlet 的方法，定義語法如下：

```
回傳資料型別 函數名稱（引數串列） {
    ……  // 程式碼
  return 回傳值;
}
```

屬性與方法的宣告
- 函數（方法）的定義

- 下面是語法各部份的說明:
  - ✓ 回傳資料型別

    此函數完成計算後，所回傳資料的型別。若沒有回傳值時，需設為 void 。
  - ✓ 函數名稱

    定義函數的名稱。

屬性與方法的宣告
- 函數（方法）的定義

- ✓ 參數串列

  函數被呼叫時，接收傳入資料的引數。引數的宣告語法如下：

  型別 引數 **1,** 型別 引數 **2, ...**

- ✓ return 回傳值

  將函數的運算結果傳出。如果方法沒有傳出值，則此敘述可省略。

屬性與方法的宣告
- 函數（方法）的定義

- 當欲呼叫函數時，只要在程式中運用以下的語法即可呼叫：

  函數名稱 （參數 **1**, 參數 **2**, … , 參數 **N**）

屬性與方法的宣告

- 屬性的宣告

- 當在『 <%! ..... %> 』敘述內宣告變數時，在 JSP 網頁內，則該變數則會成為所有存取該 JSP 網頁連線共用的變數。
- 而宣告在小文稿元件內的變數，其有效範圍僅止於完成宣告之敘述後的 JSP 網頁，且每個存取該 JSP 網頁的連線不共享該變數。

屬性與方法的宣告

- 屬性的資料分享問題

- 在『<%! ..... %>』敘述內定義的變數，將成為 Servlet 類別的屬性，在物件導向觀念裡，此屬性的有效範圍是整個 Servlet 類別產生的實例，因此，屬性是一個實例變數。

屬性與方法的宣告

- 屬性的資料分享問題

- 但由於回應連線要求時，JSP 容器並不會產生新的 Servlet 類別實例，而是由已存在的實例建立新的執行緒回應。此時，屬性便成了數個回應連線之執行緒共同存取的變數。

屬性與方法的宣告

- 屬性的資料分享問題

- 問題來了，既然是各連線共同存取，這意味著屬性資料，並不只是單一連線所獨有，因此，較為重要的資料，以及每個連線應該獨有的資料，均不可運用屬性儲存，否則資料會有外洩或產生錯誤的疑慮。

屬性與方法的宣告

- 屬性的資料分享問題

- 當然某些特殊情況下，您希望將資料分享給存取同一 JSP 網頁的各連線時，您可以運用屬性。
- 屬性的有效範圍，與儲存在 Application 物件內的變數不同，是單一 JSP 網頁的所有連線，不是特定 Web 應用程式的所有連線。

jspInit() 與 jspDestroy()

- 若欲在 JSP 網頁開始執行時，進行某些資料的起始，您可利用 jspInit 函數完成。
- 此函數將在 JSP 網頁被執行時呼叫，且當 JSP 網頁重新整理時，並不會被再度執行。
- 當關閉伺服器時， jspDestroy 函數將被呼叫，您可利用該函數進行資料的善後處理。