巨匠線上真人

Java Web OCE JWCD元件系統開發認證

巨匠線上真人

Java Web OCE JWCD元件系統開發認證

第七堂 Java網站UI設計 -JSP基礎語法

本堂教學重點

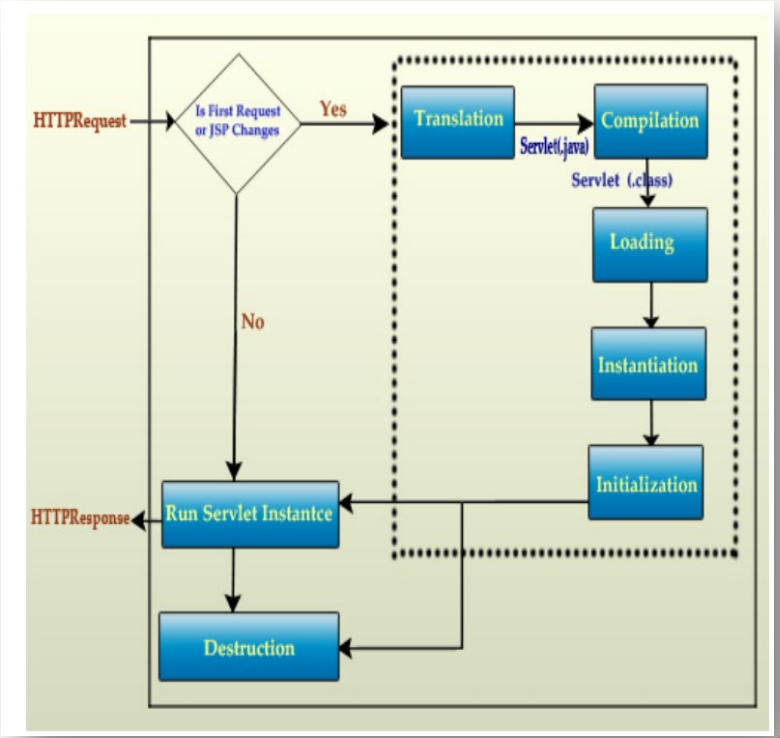
- JSP 運作流程/生命週期
- JSP 語法元素
- JSP @page 屬性宣告
- JSP 隱含物件
- JSP 跳脫字元
- JSP <jsp-property-group> 標籤

JSP 運作流程/生命週期

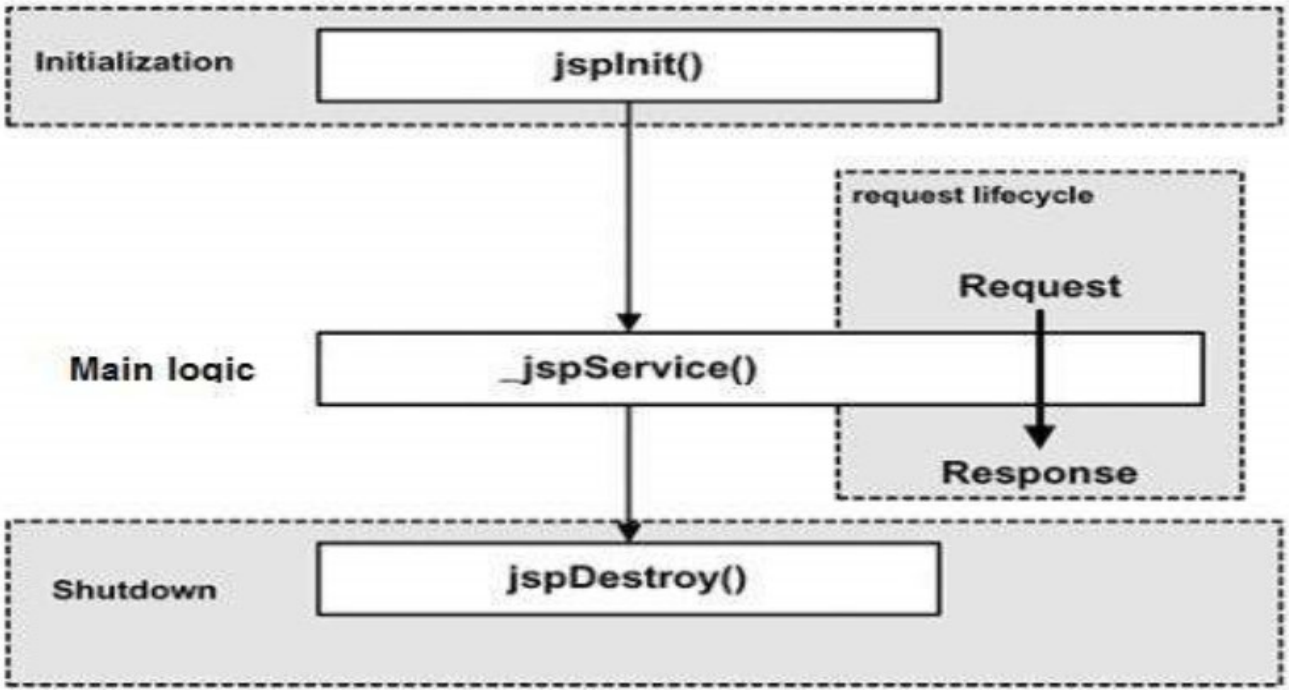
- JavaEE 的主要規格包含有:
  - Servlet
  - JSP-ava Server Page
  - JNDI-The Java Naming and Directory Interface
  - EJB –Enterprise JavaBean
- 其中具有網站系統層級的是 Servlet 與 JSP
- JSP主要是應用程式開發更容易聚焦在UI(瀏覽器使用者畫面)設計，並且可以透過Scriptlet撰寫Java程式碼在網頁中，讓網頁具有動態功能，可以輕鬆結合畫面設計與Server Side程式碼存取後端資源的設計優勢。
- JSP並非是直譯式的Java程式，是經由轉譯與編譯，並且產生一個Servlet物件進行執行，且轉換成HTML在回應到前端。

JSP生命週期

- 1.Page translation-網頁轉譯成Java Code
- 2.Page compilation-網頁編譯
- 3.Load class-載入類別
- 4.Create instance-產生一個個體物件(Just One)
- 5.呼叫 jspInit() 方法進行初始化
- 6.呼叫 _jspService()進行ServletRequest與ServletResponse操作
- 7.呼叫 jspDestroy()，釋放掉物件引發的程序



JSP 處理過程

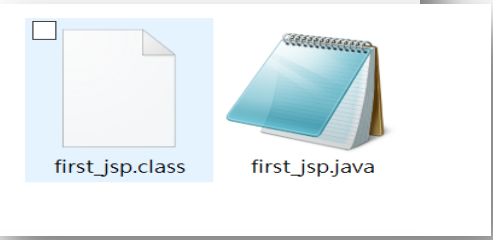

Initialization
jspInit()
request lifecycle
Request
Main logic
_jspService()
Response
Shutdown
jspDestroy()

JSP Translation-轉譯成Servlet

- 當撰寫好 JSP Page 時，第一次被請求時，會透過Web Container會先將JSP Page轉換成 Servlet Java原始程式碼 (*.java)
- 接下來進行編譯(*.class)與執行。
- 所以我們說JSP Pages 就是 Servlet。
- JSP 在 轉譯階段會預先解釋 <%@ .. %> Page Directive(網頁指示詞)針對網頁物件初始化設定。

原始HTML標籤

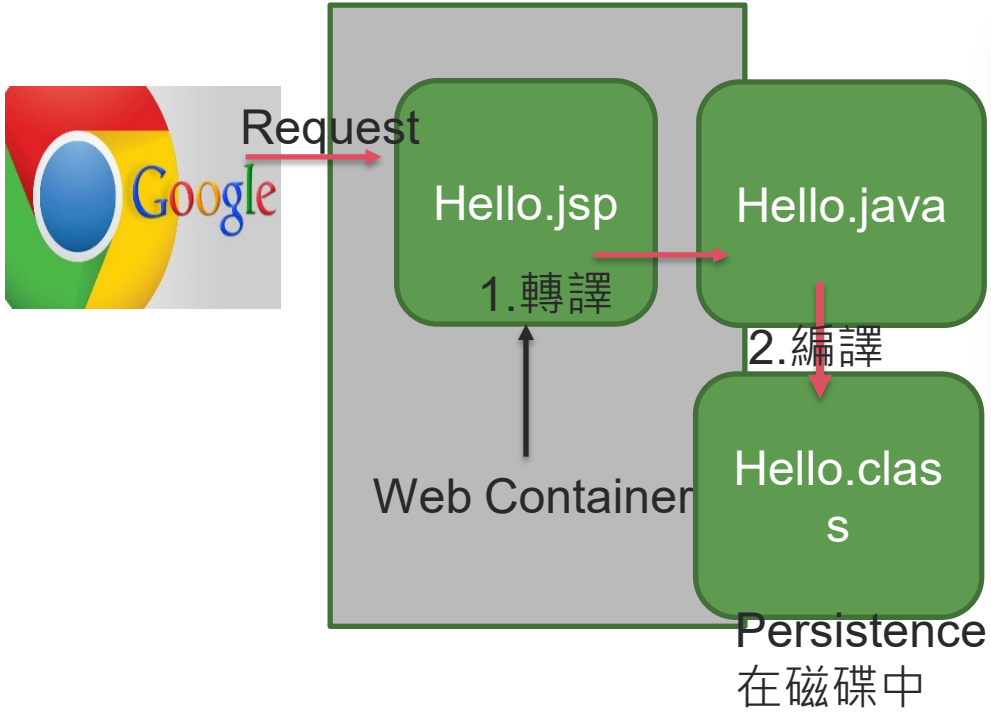
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<div>您好</div>
</body>
</html>
```



轉譯的Java Code

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class first_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
                 org.apache.jasper.runtime.JspSourceImports {

  private static final javax.servlet.jsp.JspFactory _jspxFactory =
          javax.servlet.jsp.JspFactory.getDefaultFactory();

  private static java.util.Map<java.lang.String,java.lang.Long> _jspx_dependants;

  private static final java.util.Set<java.lang.String> _jspx_imports_packages;

  private static final java.util.Set<java.lang.String> _jspx_imports_classes;

  static {
    _jspx_imports_packages = new java.util.HashSet<>();
    _jspx_imports_packages.add("javax.servlet");
    _jspx_imports_packages.add("javax.servlet.http");
    _jspx_imports_packages.add("javax.servlet.jsp");
```

JSP 的生命週-編譯




```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class first_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
                 org.apache.jasper.runtime.JspSourceImports {

  private static final javax.servlet.jsp.JspFactory _jspxFactory =
          javax.servlet.jsp.JspFactory.getDefaultFactory();

  private static java.util.Map<java.lang.String,java.lang.Long> _jspx_dependants;

  private static final java.util.Set<java.lang.String> _jspx_imports_packages;

  private static final java.util.Set<java.lang.String> _jspx_imports_classes;

  static {
    _jspx_imports_packages = new java.util.HashSet<>();
    _jspx_imports_packages.add("javax.servlet");
    _jspx_imports_packages.add("javax.servlet.http");
    _jspx_imports_packages.add("javax.servlet.jsp");
```

JSP 的生命週期-API依據

- 雖然我們說JSP就是Servlet，但是在轉譯的原始程式碼，JSP是採用JSP API的HttpJspPage介面當作基底。
- HTTPJspPage則繼承Servlet Interface。
- HttpJspPage提供jspService()而JspPage 介面所提供的jspInit() 與 jspDestroy()。

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class first_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
                 org.apache.jasper.runtime.JspSourceImports {
```

因為使用Tomcat 所以轉譯成 Tomcat JSP基底類別

org.apache.jasper.runtime

Class HttpJspBase

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--org.apache.jasper.runtime.HttpJspBase
```

All Implemented Interfaces:

javax.servlet.jsp.HttpJspPage, javax.servlet.jsp.JspPage, java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

javax.servlet.jsp

Interface HttpJspPage

All Superinterfaces:

JspPage, Servlet

JSP 與 Servlet 生命週期對照

- JSP : jspInit() /Servlet : init()
- JSP : _jspService() / Servlet : service()
- JSP : jspDestroy() / Servlet : destroy()

Method Summary	
void	_jspService(HttpServletRequest request, HttpServletResponse response) The _jspService()method corresponds to the body of the JSP page.

Methods inherited from interface javax.servlet.jsp.JspPage
jspDestroy, jspInit

JSP生命週期-jspInit()

- JSP產生的物件只有一個，物件的初始化則執行jspInit() Method進行設定。
- 可以使用JSP Declare Scriptlet進行jspInit() Overriding。
- 在網頁中，自訂初始化Scriptlet程序程序:
  - <%! **public void jspInit() {….}**%>

```
<body>
<%!
    private String companyName;
    //Overriding 初始化
    public  void jspInit(){
        this.companyName="巨匠電腦";
}
%>
<div style="font-size:48px;color:red">公司名稱:<%=this.companyName%></div>
</body>
```

JSP 的生命週期-_jspService

- JSP整個運作都在 _jspService() 中，其中包含 HTML 標籤、JSP 所撰寫的scriptlet 與 JSP expression 或者自訂標籤等，都是這一個方法允運作的一部分。
- 撰寫JSP不能透過Declare進行 _jspService() Overriding。
  - 因為 Web Container會將 JSP Page 中有關 HTML、scriptlet 與 expression產生Java Code在 _jspService() 中，所以不能在JSP中進行Overriding。

```
<body>
<%!
    private String companyName;
    //Overriding 初始化
    public  void jspInit(){
        this.companyName="巨匠電腦";
}
%>
<div style="font-size:48px;color:red">公司名稱:<%=this.companyName%></div>
<%
    //區域變數定義
    String address="台北市公園路";
    out.println(address);
%>

</body>
</html>
```

Expression

Scriptlet

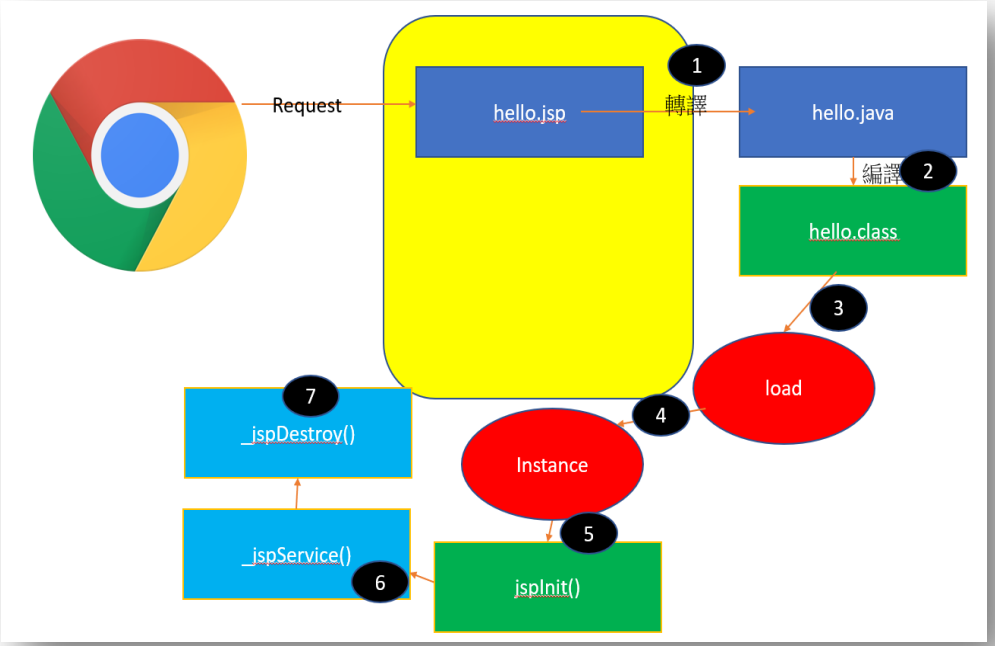
```
public final class jspinitdemo_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
                 org.apache.jasper.runtime.JspSourceImports {


    private String companyName;
    //Overriding 初始化
    public  void jspInit(){
            this.companyName="巨匠電腦";
}
```

Overriding

```
out.write("\r\n");
out.write("<div style=\"font-size:48px;color:red\">公司名稱:");
out.print(this.companyName);
out.write("</div>\r\n");

  //區域變數定義
  String address="台北市公園路";
  out.println(address);
```

JSP 的生命週期- jspDestroy()

- 當 JSP不再提供服務時，進行銷毀時所呼叫的方法(事件程序)。
- 與 Servlet destroy() 方法一般。



JSP 語法元素

- ***JSP*** 在Script語法分為:

<% scripting-language-statements %>

```
<%
String username = request.getParameter("username");
if ( username != null && username.length() > 0 )
{
 %>
```

<%= scripting-language-expression %>

```
<h2><font color="black"><%= resp %>!</font></h2>
```

<%! scripting-language-declaration %>

```
<%!
private BookDBAO bookDBAO;
public void jspInit() {
bookDBAO =
     (BookDBAO)getServletContext().getAttribute("bookDB");
     if (bookDBAO == null)
          System.out.println("Couldn't get database.");
}
%>
```

JSP 語法元素-Demo

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP init範例</title>
</head>
<body>
<%!
    private String companyName;
    //Overriding 初始化
    public  void jspInit(){
        this.companyName="巨匠電腦";
}
%>
<div style="font-size:48px;color:red">公司名稱:<%=this.companyName%></div>
<%
    //區域變數定義
    String address="台北市公園路";
    out.println(address);
%>

</body>
</html>
```

JSP註解 - Comments

- 在 JSP Page Scriptlets，可以使用單行或者註解進行說明。或者使用標籤撰寫。
- 語法：
  - 單行：<% // … %>
  - 多行：<% /* … */%>
  - HTML 註解：<!-- … -- >

```
<body>
<%!
    //單行註解說明
    private String companyName;
    //Overriding 初始化
    public  void jspInit(){
        this.companyName="巨匠電腦";
}
%>
<!-- 直接嵌入在標籤中的運算式 -->
<div style="font-size:48px;color:red">公司名稱:<%=this.companyName%></div>
<%
    /*區域變數定義
    *區域變數只活在_jspService Method中
    */
    String address="台北市公園路";
    out.println(address);
%>

</body>
```

JSP指令- @page Directives(指示詞)

- 使用<%@...%>宣告語法
- 於 轉譯時期（編譯時期之前）時預先提供給 JSP網頁物件的相關設定。
- 支援有:
  - @page
  - @include
  - @taglib

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP init範例</title>
```

@page Directive

- 使用import attribute引用package
- 使用session attribute開啟或者關閉Session狀態管理
- isScriptingEnabled attribute可以關閉JSP Script執行。
- isErrorPage設定這一個網頁為其他Page導向過來的錯誤攔截與處理的Page。isErrorPage=true，可以讓這一個Page的exception內建物件啟動作用。

屬性	目的
buffer	Specifies a buffering model for the output stream.
autoFlush	Controls the behavior of the servlet output buffer.
contentType	Defines the character encoding scheme.
errorPage	Defines the URL of another JSP that reports on Java unchecked runtime exceptions.
isErrorPage	Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute.
extends	Specifies a superclass that the generated servlet must extend
import	Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes.
info	Defines a string that can be accessed with the servlet's getServletInfo() method.
isThreadSafe	Defines the threading model for the generated servlet.
language	Defines the programming language used in the JSP page.
session	Specifies whether or not the JSP page participates in HTTP sessions
isELIgnored	Specifies whether or not EL expression within the JSP page will be ignored.
isScriptingEnabled	Determines if scripting elements are allowed for use.

@page Directive Demo

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<%@ page  info="我是JSP網頁 由Eric設計"  %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>@page Directivie應用</title>
</head>
<body>
<%
    String hello="您好";
    out.println(this.getServletInfo());
%>
```

指令 – include Directives

- 目的：告知 Web Container 目前的 JSP page 所要含入的外部 page (HTML、JSP)來源。
- 含入在網頁中呈現。
  - 宣告方式：
    - <%@ include file="banner.html" %> 預先將 banner.html包含在該 JSP page 一起編譯中與執行。
    - 指令級的 include 是屬於靜態(static)的，也就是說file attribute不能使用運算式或者 EL。

banner.html

```
1 <!DOCTYPE html>
2 <a href="www.hinet.net">hinet</a>
3 <a href="www.pcschool.com.tw">pcschool</a>
```

pagedirectivedemo.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
 <%@ page  info="我是JSP網頁 由Eric設計"  %>
 <%@ include file="banner.html" %>
<!DOCTYPE html>
<html>
<head>
```

http://localhost:8080/mod07/pagedirectivedemo.jsp

hinet pcschool
我是JSP網頁 由Eric設計

taglib指令 - Directives

- 目的：告知 Web Container，該 JSP page 要使用自訂標籤或者是JSTL支援的Tag。
  - 宣告方式：

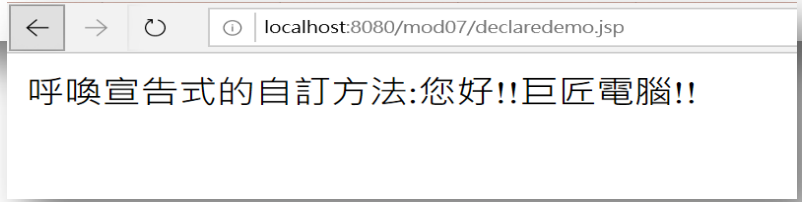
    - <%@ taglib prefix="c" uri="xxx.tld"%>
      JSP page 將可以針對 uri 所指定的 custom tags 利用前置詞 foo 進行存取的動作。
      - <c: XXX />

JSP page 屬性宣告

- 使用宣告式語法(Declare)
- <%!....%>宣告類別層級的attribute/Method。
- 可以透過Declare語法進行HttpJspPage 生命週期的方法Overriding。

declaredemo.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>宣告式語法應用</title>
<%!
    //定義Attribute
    private String helloMessage;
    //Overriding jspInit()
    public void jspInit(){
        this.helloMessage="您好!!巨匠電腦!!";
    }
    //定義method
    public String getGreeting()
    {
        return this.helloMessage;
    }
%>
</head>
<body>
<div>呼喚宣告式的自訂方法:<%=this.getGreeting()%></div>
</body>
</html>
```

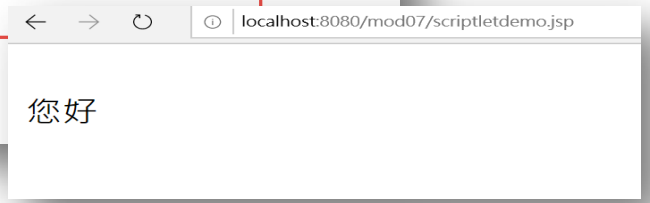


JSP程式片段 - Scriptlets

- 轉譯在 JSP _jspService()方法內的小程序(片段)。
- Scriptlet 是 JSP page 中的 java 程式片段。
- 可以被拆撰寫在JSP中的任何位置。因為Scriptlet為_jspService實作的一部分，所以不能使用類別定義方式，如設定Modifier修飾詞或者自訂Method等。
  - 宣告方式：<% … %>。

scriptletdemo.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Scriptlet Demo</title>
</head>
<body>
<%
    //定義區域變數
    String hello="您好";
%>
<br/>
<%
    //使用區域變數
    out.println(hello);
%>
</body>
</html>
```



JSP精簡表示(運算式) - Expressions

- 也是JSP_jspService()方法內的邏輯程序一部分。
- Expression 是 JSP 中的特殊程式碼撰寫方式，只要有回傳值的運算或是方法皆可，主要用來呈現內容。
  - 宣告方式：<%= XXX %>
  - 使用 運算式時，不可在程式碼後面加上分號 「;」，設是因為本身的實作就是out.print()-帶出輸出。

```
<title>宣告式語法應用</title>
<%!
    //定義Attribute
    private String helloMessage;
    //Overriding jspInit()
    public void jspInit(){
        this.helloMessage="您好!!巨匠電腦!!";
    }
    //定義method
    public String getGreeting()
    {
        return this.helloMessage;
    }
%>
</head>
<body>
<div>呼喚宣告式的自訂方法:<%=this.getGreeting()%></div>
</body>
</html>
```

JSP Action Elements

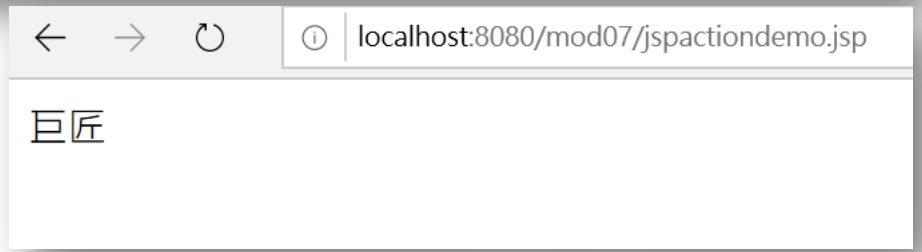
- Action Element是 JSP 的特殊用法。允許 JSP page 在執行期間給予適當的指令(背後使用Custom tag class支撐)，並且直接呼叫 Web Container 予以執行。
  - 宣告：<jsp:actionName attribute-list />
    - <jsp: XXX /> 是 Web Container內建的命令，當然 JSP 也允許開發者自行設計 action 這就是所謂的自訂標籤技術。

JSP Action Tags	Description
jsp:forward	forwards the request and response to another resource.
jsp:include	includes another resource.
jsp:useBean	creates or locates bean object.
jsp:setProperty	sets the value of property in bean object.
jsp:getProperty	prints the value of property of the bean.
jsp:plugin	embeds another components such as applet.
jsp:param	sets the parameter value. It is used in forward and include mostly.
jsp:fallback	can be used to print the message if plugin is working. It is used in jsp:plugin.

JSP Action應用範例

- 使用<jsp:userBean>建構一個Javabean物件
- <jsp:setProperty>設定物件屬性內容
- <jsp:getProperty>取得特定物件的屬性值

```
<jsp:useBean id="customers" class="com.gjun.domain.Customers"></jsp:useBean>
<jsp:setProperty property="customerid" name="customers" value="0001"/>
<jsp:setProperty property="companyName" name="customers" value="巨匠"/>
<jsp:setProperty property="address" name="customers" value="高雄市"/>

<!-- 呈現資料 -->
<jsp:getProperty property="companyName" name="customers"/>
```



JSP @page 屬性宣告

- @page Directive指示詞，會在網頁轉譯時，用來定義網頁物件的初始屬性。

屬性	屬性
buffer	info
autoFlush	isThreadSafe
contentType	language
errorPage	session
isErrorPage	isELIgnored
extends	isScriptingEnabled
import	

@page import attribute

- JSP預設import package
  - javax.servlet
  - javax.servlet.http
  - javax.servlet.jsp
- 如同Java Source寫法一般，在@page directive 使用import attribute引用其他的package與法。
  - import="java.util.List,java.util.ArrayList"
  - import="java.util.*"

defaultimport.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    import="java.util.*" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    List<String> names=new ArrayList<String>();
    names.add("allen");
    names.add("barry");
    names.add("cathy");
%>
<div>姓名1:<%=names.get(0)%></div>
<div>姓名2:<%=names.get(1)%></div>
<div>姓名3:<%=names.get(2)%></div>
</body>
</html>
```

@page isELIgnored

- 在JSP中可以使用EL(Expression Language)，使用EL內建物件或者是不同層級的Attribute，進行快速呈現。
- 亦可以呼叫函數(Function)，自行規劃的Function進行運算輸出。
- **isELIgnored預設為false，如果將這一個屬性設定為true，則撰寫在網頁中的EL將會不被執行，直接網頁文字內容輸出。**

iselignoreddemo.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    isELIgnored="false" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    pageContext.setAttribute("name","Eric Chen");
%>
<div>您的姓名:${pageScope.name}</div>
</body>
</html>
```

localhost:8080/mod07/iselignoreddemo.jsp

您的姓名:Eric Chen

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    isELIgnored="true" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%
    pageContext.setAttribute("name","Eric Chen");
%>
<div>您的姓名:${pageScope.name}</div>
</body>
</html>
```

localhost:8080/mod07/iselignoreddemo.jsp

您的姓名:${pageScope.name}

@page errorPage/isErrorPage應用

- isErrorPage預設為false，如果定義為true，則將此網頁設定為其他JSP設定errorPage屬性指定的例外處理導向的網頁。
- 網頁內建物件exception，需要在isErrorPage宣告為true，才能使用，但是這樣的網頁將無法獨立運作。

myerror.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    errorPage="errorhandler.jsp" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>我發生錯誤</title>
</head>
<body>
<%
    int result=10/0;
%>
</body>
</html>
```

不合理的運算

errorhandler.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    isErrorPage="true" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>錯誤處理網頁</title>
</head>
<body>
<div>您發生的錯誤訊息:<%=exception.getMessage()%></div>
</body>
</html>
```

設定為錯誤處裡委派的網頁

內建exception物件可以運行

@page session attribute

- session attribute預設狀態為true，主要用來設定該JSP是否啟動內建session物件(javax.servlet.http.Session介面)。
- 如果設定false，則將無法使用到session內物件進行進行session範圍內的狀態參考與使用。

sessionadd.jsp

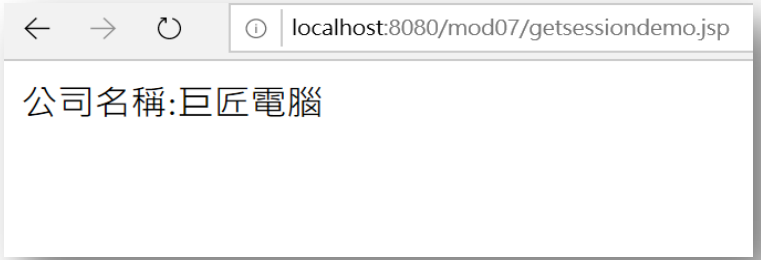
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    session="true" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        String companyName="巨匠電腦";
        //進入Session狀態管理
        session.setAttribute("com",companyName);
    %>
    <div>透過Session參照一個字串物件完成</div>
</body>
</html>
```

預設為true，可以不用特別設定

getsession.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <div>公司名稱:${sessionScope.com}</div>
</body>
</html>
```

使用EL內建物件參照出Session管理的物件內容



JSP 隱含物件

- JSP實際上被轉換一個Servlet Java Source。但卻為了設計方便性提供使用畫面高階設計方式，讓我們在網頁中直接編輯HTML/CSS/JavaScript與Java Scriptlet。
- 但在原始Servlet API底層常用的或者轉譯成_jspService()方法中定義的Request/Response參數等。均有一套標準的變數參照相對的物件，讓我們是前在網頁撰寫中直接使用，而無須再針對這些物件進行建立。
- 這些對應轉譯Servlet程式碼常用Servlet API物件的參數與變數，被稱呼為JSP隱含物件或者內建物件。

```
  public void _jspService(final javax.servlet.http.HttpServletRequest request,
Ofinal javax.servlet.http.HttpServletResponse response)
      throws java.io.IOException, javax.servlet.ServletException {

    if (!javax.servlet.DispatcherType.ERROR.equals(request.getDispatcherType())) {
      final java.lang.String _jspx_method = request.getMethod();
      if ("OPTIONS".equals(_jspx_method)) {
        response.setHeader("Allow","GET, HEAD, POST, OPTIONS");
        return;
      }
      if (!"GET".equals(_jspx_method) && !"POST".equals(_jspx_method) && !"HEAD".equals(_jspx_method)) {
        response.setHeader("Allow","GET, HEAD, POST, OPTIONS");
        response.sendError(HttpServletResponse.SC_METHOD_NOT_ALLOWED, "JSPs only permit GET, POST or HEAD.
        return;
      }
    }

    final javax.servlet.jsp.PageContext pageContext;
    final javax.servlet.ServletContext application;
    final javax.servlet.ServletConfig config;
    javax.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    javax.servlet.jsp.JspWriter _jspx_out = null;
    javax.servlet.jsp.PageContext _jspx_page_context = null;
```

JSP內建物件

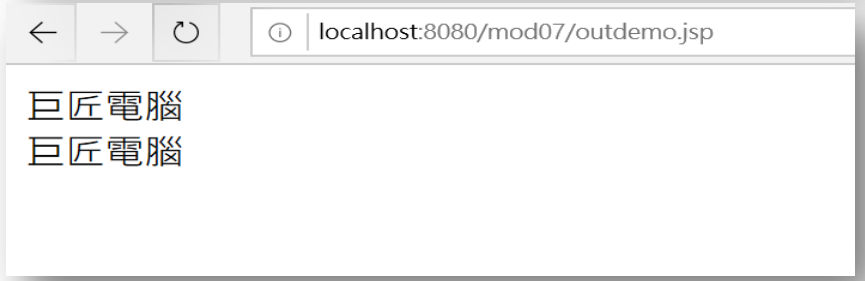
內建物件	說明
out	對應 JspWriter 物件，其內部關聯一個 PrintWriter 物件。
request	對應 HttpServletRequest 物件，實際上為_jspService參數。
response	對應 HttpServletResponse 物件,實際上為_jspService參數。
config	對應 ServletConfig 物件。
application	對應 ServletContext 物件。
session	對應 HttpSession 物件。
pageContext	對應 PageContext 物件，它提供了 JSP 頁面資源的封裝，並可設定頁面範圍屬性。
exception	對應 Throwable 物件，代表由其他JSP 頁面丟出的例外物件，只會出現於 JSP 錯誤頁面（isErrorPage 設定為 true 的 JSP 頁面）。
page	對應 this

JSP out內建物件

- 如同透過ServletResponse參照出的PrintWriter物件一般。
- 但在JSP API則對應JspWriter，仍屬於java.io.Writer的延伸。
- 透過Scriptlet或者進行網頁內容印出作業。
- 如同Scriptlet Expression轉譯為JspWriter進行輸出。當我們使用Expressions敘述中不要再使用out內建物件。
  - <%=out.println(“hello”)%> 將會有錯誤。

outdemo.jsp

```
<body>
    <%
        //設定區域變數
        String com="巨匠電腦";
        out.println(com);
    %>
    <div><%=com%></div>
</body>
```



JSP request內建物件

- 對應物件為HttpServletRequest物件。
- 封裝前端所有資訊進入這一個JSP頁面。可以獲取前端相關資訊進行處理或者呈現處理。

customers.html

```
<form method="post" action="customersprocess.jsp">
    <div>客戶編號</div>
    <input type="text" name="customerid"/>
    <div>公司行號</div>
    <input type="text" name="companyName"/>
    <div>聯絡地址</div>
    <input type="text" name="address"/>
    <div>連絡電話</div>
    <input type="text" name="phone"/>
    <input type="submit" value="傳送"/>
</form>
```

customersprocess.jsp

```
<body>
    <%
    request.setCharacterEncoding("UTF-8");
    //擷取表單欄位內容
    String customerid=request.getParameter("customerid");
    String companyName=request.getParameter("companyName");
    String address=request.getParameter("address");
    String phone=request.getParameter("phone");
    %>
    <br/>
    <div>客戶編號:<%=customerid%></div>
    <div>公司行號:<%=companyName%></div>
    <div>聯絡地址:<%=address%></div>
    <div>連絡電話:<%=phone%></div>
</body>
```

Jsp response內建物件

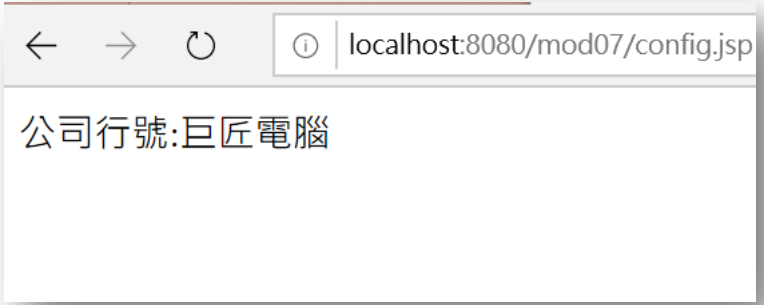
- 對應HttpServletResponse物件。
- 用來回應資訊到相對的瀏覽器。

customersprocess.jsp

```
<body>
    <%
    request.setCharacterEncoding("UTF-8");
    //擷取表單欄位內容
    String customerid=request.getParameter("customerid");
    String companyName=request.getParameter("companyName");
    String address=request.getParameter("address");
    String phone=request.getParameter("phone");
    %>
    <br/>
    <div>客戶編號:<%=customerid%></div>
    <div>公司行號:<%=companyName%></div>
    <div>聯絡地址:<%=address%></div>
    <div>連絡電話:<%=phone%></div>
    <div></div>
    <%
        PrintWriter writer=response.getWriter();
        writer.println("Hello World!!");

    %>
</body>
```

JSP config內建物件

- 對應ServletConfig，JSP轉譯就是相對於Servlet。所以具有Servlet進行初始化資訊設定的架構。
- 可以使用web.xml佈署JSP，且部署初始化參數進行傳遞，封裝到ServletConfig物件中。



configdemo.jsp

```
<body>
    <%
        //取出初始化參數
        String com=config.getInitParameter("com");
    %>
    <div>公司行號:<%=com%></div>
</body>
```

web.xml

```
<servlet>
  <servlet-name>config</servlet-name>
  <jsp-file>/configdemo.jsp</jsp-file>
  <init-param>
      <param-name>com</param-name>
      <param-value>巨匠電腦</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>config</servlet-name>
  <url-pattern>/config.jsp</url-pattern>
</servlet-mapping>
```

JSP appliction內建物件

- 對應ServletContext物件。
- 可以獲取應用系統相關的資源。
- 如取得虛擬目錄對應的實際目錄等操作。

applicationdemo.jsp

```
<body>
    <!-- 獲取應用系統資訊 -->
    <%
        String app=application.getInitParameter("appname");
        out.println(app);
    %>
</body>
```

web.xml

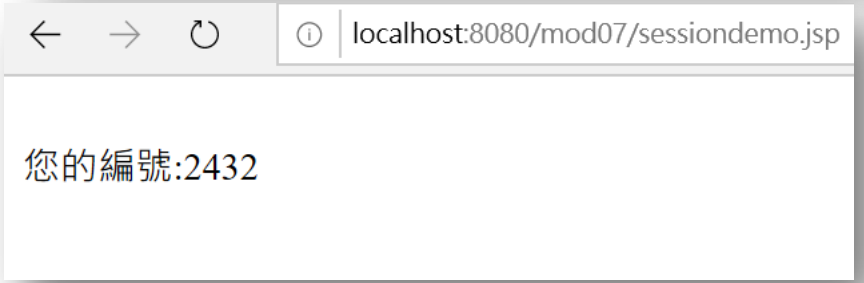
```
<context-param>
  <param-name>appname</param-name>
  <param-value>MOD07展示系統</param-value>
</context-param>
```

JSP session內建物件

- 對應HttpSession物件。
- 用來設定或者取得Session範圍的狀態與資訊。
- 如使用者個別狀態ID等(隨機碼的管理與參考)。
- 或者購物車狀態管理作業。

sessiondemo.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Session狀態管理與應用</title>
</head>
<body>
    <%
        //產生隨機碼
        int rad=(int)(Math.random()*10000)+1;
        session.setAttribute("number",rad);
    %>
    <br/>
    <div>您的編號:<%=(Integer)session.getAttribute("number")%></div>
</body>
</html>
```



JSP pageContext內建物件

- pageContext對應的物件為PageContext。
- 主要封裝網頁層級的資訊，進行Page Level狀態管理用。
- 如使用EL只能應用在相關可視範圍的狀態，不能操作區域變數，因此可以在網頁中產生的區域變數封裝到Page層級的資訊。即可使用EL運算輸出。

pagecontextdemo.jsp

```
<body>
    <%
        //區域變數設定
        String com="巨匠電腦";
    %>
    <!-- 使用EL輸出 -->
    <div>公司行號:${com}</div>
</body>
```

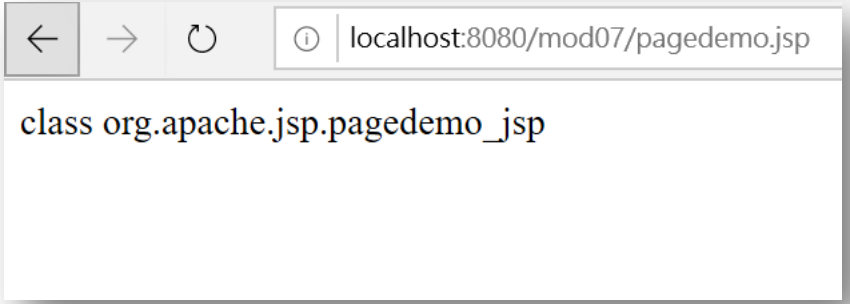
無法運算


```
<body>
    <%
        //區域變數設定
        String com="巨匠電腦";
        pageContext.setAttribute("company", com);
    %>
    <!-- 使用EL輸出 -->
    <div>公司行號:${company}</div>
</body>
```

JSP page內建物件

- Page對應的就是this，網頁物件。
- 可以操作基底類別的方法與屬性。

```
<body>
    <%!
    public String hello(){
        return "大家好";
    }
    %>

    <%=page.getClass()%>
</body>
```



JSP可視化範圍應用

- 可視化一般用在網站系統爭對不同層級的狀態(State)進行參考與管理。
- 如購物車的狀態參考。
- 如應用系統共用的狀態參考等。
- JSP支援有相對於Servlet定義的四個範圍的可視化參考應用。

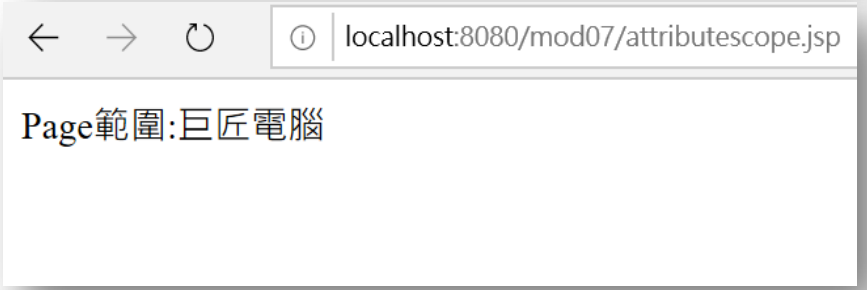
內建物件	存在範圍	對應API
pageContext	page層級，存在一個相對網頁層級範圍	PageContext
request	單一請求(Request)範圍上。	ServletRequest
session	限制在一個使用者端的請求上，橫跨同一個應用系統範圍上。	HttpSession
application	限制在一個應用系統上，形成應用系統共用的參考。	ServletContext

如何進行不同可視範圍的物件操考

- 通用使用了setAttribute(“自訂參照名稱”,Object)，即使參照了基本型別，則會自動進行Autoboxing封裝呈相對的物件進行參考。
- 在不同範圍設定同樣的參照名稱，透過EL進行可見範圍的物件取得，則會使用最小可視範圍進行。
- 透過getAttribute()進行範圍參照物件的取得，一律採用Object型別回應，明確操作時必須進行轉型。

attributescope.jsp

```
<body>
    <%
        String com="巨匠電腦";
        pageContext.setAttribute("com",com);
        String branch="台北認證中心";
        session.setAttribute("com",branch);
    %>
    <!-- page level -->
    <div>Page範圍:${com}</div>
</body>
```



JSP 跳脫字元

escapedemo.jsp

```
<body>
    <%
        String content="Bill say "<Java> Good!!!"";
        out.print(content);
    %>
</body>
```

- JSP使用到單(‘)雙(“)引號、反斜線(\)、<%@、<%! 、<%=、<%、%>、<%--與 %-->這些與Scriptlet語法有關或者是與字串定義有關的字元，將會與語法產生衝突，該如何避開?
- 可以使用跳脫字元 “\”避開即可。

```
org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: [11] in the jsp file: [/escapedemo.jsp]
Java cannot be resolved to a variable
8: </head>
9: <body>
10:     <%
11:         String content="Bill say "<Java> Good!!!"";
12:         out.print(content);
13:     %>
14: </body>
```

```
<%
    String content="Bill say \"<Java> Good!!!\"";
    out.print(content);
%>
```

JSP <jsp-property-group> 元素

- JSP 2.0支援JSP屬性佈署元素，用意定義。
- 支援有八個元素進行設定。
- 設定EL運算語言的支援性。
- 設定JSP執行Scriptlet支援性。
- 設定JSP網頁編碼規則。
- 設定JSP抬頭與頁尾的include。
  - 需要設定為.jspf頁面。

```
<servlet-mapping>
  <servlet-name>config</servlet-name>
  <url-pattern>/config.jsp</url-pattern>
</servlet-mapping>
<jsp-config>
  <jsp-property-group>
      <description>JSP Script教學說明</description>
      <url-pattern>*.jsp</url-pattern>
      <el-ignored>true</el-ignored>
  </jsp-property-group>
</jsp-config>
```

Lab

- JSP生命週期為何
- JSP支援的Script語法區分哪幾種
- JSP Directive區分哪幾種，與其應用方式
- 如何在JSP引用JSTL Tag Library
- 如何透過Action Element設計一個客戶資料存取表單網頁
- JSP內建物件區分那些