JSP 內建物件

JSP 中的四種屬性範圍

- 屬性範圍：就是指一個物件可以跨多少個 JSP 頁面之後可以繼續使用

- JSP 中有以下**四種屬性範圍：**
  - page
  - request
  - session
  - application

- 在整個 JAVA WEB 中屬性操作使用如下的方法：

設置屬性：
public void setAttribute(String name,Object attribute)

取得屬性： public Object getAttribute(String name)

刪除屬性： public Object removeAttribute(String name)

Page(pageContext)

- 在一個 JSP 頁面上設置的屬性只能在一個頁面取得，跳轉到其他頁面則此屬性消失
- <%@page contentType="text/html;charset=big5"%>
- <%@page import="java.util.*"%>
- <%
- 	// 設置兩個屬性
- 	pageContext.setAttribute("uname","HELLO") ;
- 	pageContext.setAttribute("udate",new Date()) ;
- 	// 取得屬性
- 	String name = (String)pageContext.getAttribute("uname") ;
- 	Date date = (Date)pageContext.getAttribute("udate") ;
- %>
- <h1>name --> <%=name%></h1>
- <h1>date --> <%=date%></h1>

Request: 可以把屬性保存在一次伺服器跳轉範圍之中

注意 : 如果使用超連結的跳轉形式，還是不能夠取得所設值

```
<%@page contentType="text/html;charset=big5"%>
<%@page import="java.util.*"%>
<%
request.setAttribute("uname","HELLO") ;
request.setAttribute("udate",new Date()) ;
%>
<jsp:forward page="RequestScopeDemo02.jsp"/>
```

```
<%@page contentType="text/html;charset=big5"%>
<%@page import="java.util.*"%>
<%
String name = (String)request.getAttribute("uname") ;
Date date = (Date)request.getAttribute("udate") ;
%>
<h1>name --> <%=name%></h1>
<h1>date --> <%=date%></h1>
```

session 屬性範圍無論頁面怎樣跳轉，都可以保存下來，但是只針對於同一個瀏覽器打開的相關頁面

```
<%@page contentType="text/html;charset=big5"%>
<%@page import="java.util.*"%>
<%
session.setAttribute("uname","HELLO") ;
session.setAttribute("udate",new Date()) ;
%>
```

```
<%@page contentType="text/html;charset=big5"%>
<%@page import="java.util.*"%>
<%
String name = (String) session.getAttribute("uname") ;
Date date = (Date) session.getAttribute("udate") ;
%>
<h1>name --> <%=name%></h1>
<h1>date --> <%=date%></h1>
```

application 範圍，是把屬性設置在整個伺服器上，所有的用戶都可以進行訪問。

```
<%@page contentType="text/html;charset=big5"%>
<%@page import="java.util.*"%>
<%
application.setAttribute("uname","HELLO") ;
application.setAttribute("udate",new Date()) ;
%>
```

```
<%@page contentType="text/html;charset=big5"%>
<%@page import="java.util.*"%>
<%
String name = (String) application.getAttribute("uname") ;
Date date = (Date) application .getAttribute("udate") ;
%>
<h1>name --> <%=name%></h1>
<h1>date --> <%=date%></h1>
```

Request 物件

- request.getParameter()
- request.getParameterValues()
- request.setCharacterEncoding()// 解決中文亂碼
- Get & Post & URL 傳值
- getMethod, getRemoteAddr, getServletPath, getContextPath

Response 物件

- response 物件實作 javax.servlet.http.HttpServletResponse 介面，用於處理 JSP 網頁回傳給瀏覽端之運算結果的回應訊息。

- response.sendRedirect(“網頁路徑與名稱”); //<jsp:forward>, 利用伺服器端將資料先輸出到緩衝區機制，取消前網頁輸出換成新網頁； sendRedirect 則是利用 HTTP Header，對瀏覽器重導網頁

- **呼叫 setHeader 方法，設定網頁每 20 秒更新一次：**
  response.setHeader("Refresh", "20");

Cookie 的使用

- Cookie，是儲存於客戶端，供瀏覽器與 Web 伺服器互通資料用的純文字檔（ .txt ）。
- 當 Web 應用程式被執行時，將某些資料儲存於 Cookie 中，供執行程式時使用，這類變數稱為 Cookie 變數。
- 實際執行 JSP 網頁時，使用 session 物件需配合 Cookie 。

將資料儲存至 Cookie 的步驟如下：

- **將欲儲存的資料建立為 Cookie 變數。**

  Cookie 變數名稱 = new Cookie(" 資料名稱 ", 值);

- **呼叫 response 物件的 addCookie 方法，將前述 Cookie 變數加入 Cookie。**

  response.addCookie(變數名稱);

```
Date Now = new Date();
 // 建立 Now 變數

Cookie Time = new Cookie
    ("Now",
String.valueOf(Now.getTim
    e()));
// 將 Now 變數建立為 Cookie
    變數

response.addCookie(Time);
//Cookie 變數加入 Cookie
```

從 Cookie 取得資料的步驟如下：

```
<%
 Cookie IntVal = new Cookie("IntVal", "100");
Cookie temp = null;
response.addCookie(IntVal);

Cookie[] cookies = request.getCookies();
// 取得 Cookie 資料
if(cookies != null) // 判斷是否成功取得 Cookie 資料
{
int cookielen = cookies.length; // 取得 Cookie 變數陣列的長度
for (int i = 0; i < cookielen; i++) {
temp = cookies[i]; // 取得 cookies 陣列中的 Cookie 變數
 if (temp.getName().equals("IntVal")) {
// 判斷是否取得名為 IntVal 的 Cookie 資料
%>
 Cookie 中 <Font color = blue>IntVal</Font> 變數的值為  <Font color = red>
<%= temp.getValue()%>
</Font><BR>
<%
} } } else // 若無法取得 Cookie 資料則執行下面的敘述
{
%>
無法取得 Cookie
<BR>
 <% } %>
```

Session 物件

- session 物件將實作 javax.servlet.http.HttpSession 型別，**每個客戶端的連線，均對應一個 session 物件**。
- 當資料存入 session 物件，使用者在連線過程所瀏覽的 JSP 網頁，均可存取這些資料，達到跨網頁分享資料的目的。
- session 物件的生命週期，以設定有效時間的方式加以規範。
-
- 當使用者利用瀏覽器瀏覽某 JSP 網頁時，該網頁的 Servlet 中，將宣告一個用於代表該連線的 session 物件，並將一個用於識別 session 物件的 sessionID 寫入客戶端的 Cookie 中，供 Web 應用程式識別該物件。

為何不用 Cookie

- Cookie 僅能儲存文字資料，無法儲存物件類型的資料。
- 欲使用 Cookie 時，必須客戶端瀏覽器支援才行，但程式設計師無法掌握客戶端是否支援。
- Cookie 存在於客戶端，在資料的存取上較不方便。

- **long getCreationTime();//** 取得產生 **session** 物件的時間。
- **String getId();//** 取得 **session** 物件的識別值。
- **long getLastAccessedTime();//** 取得客戶端最後送出 **session** 要求的時間。將取得自 1970/01/01 午夜 **GMT** 開始計算迄今的毫秒數。
- **long getMaxInactiveInterval();//** 取得各要求間可允許的最大閒置時間。（單位：秒）
- **boolean isNew();//** 判斷 **session** 物件是否為剛產生的，是則傳回 **true**，反之傳回 **false**。
- **Void invalidate();//** 刪除 **session** 物件，並清除物件內儲存的所有屬性。
- **Long setMaxInactiveInterval();//** 設定各要求間可允許的最大閒置時間。（單位：秒）

Web 應用程式

- 在 Web 站台上，利用數個網頁共同完成某特定工作的系統，我們稱之為**Web 應用程式**。

- Web 應用程式的定義，大多以資料夾的方式界定，同一個 Web 應用程式的網頁，均置於同一資料夾，然後在 JSP 容器上，設定該資料夾及其子資料夾下的 JSP 網頁成為一個 Web 應用程式。
-
- 當執行 Web 應用程式時，由於整個執行過程，是由許多 JSP 網頁共同完成，某些資料必須提供各網頁共享。

- 當被指定為 Web 應用程式的資料夾中，任一個 JSP 網頁被執行時，即視為該 Web 應用程式開始執行。

- Web 應用程式的終止，在主觀上可以視所有執行 Web 應用程式的連線均斷線時為終止。但在 Resin 及 Tomcat 中，當伺服器終止時， application 物件才會終止。

定義 Tomcat 的 Web 應用程式

- 自動部署 web 應用程式於檔案系統的方式
  - 將 WAR 檔或 web 應用程式的目錄（包括所有的內容）複製到 tomcat/*webapps* 目錄中。
- WEB-INF 資料夾
  - 當定義某個資料夾為 Web 應用程式時，您可以在該資料夾下，建立 WEB-INF 子資料夾。
  - 該資料夾內將放置設定 Web 應用程式的 web.xml 檔、 JavaBeans 類別，編譯後的 Servlet 類別。

- application 物件實做 javax.servlet.ServletContext 介面。

- 在 JSP 網頁內， application 物件代表該 JSP 網頁存在的 Web 應用程式，且 Web 應用程式內所有 JSP 網頁，與所有連線的使用者，均分享存取同一個 application 物件

- 當 Web 伺服器上有一個以上 Web 應用程式時，則不同使用者瀏覽不同應用程式時，將產生不同 application 物件

- application 物件的五種常用方法
  - 屬性存取方法
  - 取得版本資訊
  - 將記錄寫至記錄檔
  - 取得 Web 應用程式的初始參數
  - 取得伺服器的相關資源