CHAPTER 7

JSTL 1.1

JSTL 全名為JavaServer Pages Standard Tag Library ，目前最新的版本為1.1 。JSTL 是由JCP（Java Community Process）所制定的標準規格，它主要提供給Java Web 開發人員，一個標準通用的標籤函式庫。

Web 程式開發人員能夠利用JSTL 和EL 來開發Web 程式，取代傳統直接在頁面上嵌入Java 程式（Scripting）的作法，以提高程式閱讀性、維護性和方便性。

本章節中，筆者將詳細介紹如何使用JSTL 中各種不同的標籤，我們將會依序介紹條件、迴圈、URL 、U18N 、XML 、SQL 等標籤的用法，讓讀者對JSTL 有更深層的認識，並且能夠學會如何使用JSTL 。

7-1 JSTL 1.1 簡介

JavaServer Pages Standard Tag Library（1.1），它的中文名稱為JSP 標準標籤函式庫。JSTL 是一個標準且已制定好的標籤函式庫，可以應用於各種領域，例如：基本輸出、輸入、流程控制、迴圈、 XML 文件剖析、資料庫查詢及國際化和文字格式標準化的應用 等等。從下列的表格可以知道， JSTL 所提供的標籤函式庫主要分為五大類：

1. 核心標籤庫（Core tag library）
2. I18N 格式標籤庫（I18N-capable formatting tag library）
3. SQL 標籤庫（SQL tag library）
4. XML 處理（XML tag library）
5. 函式功能（Functions tag library）

JSTL	前置名稱	URI	範例
核心標籤庫	c	http://java.sun.com/jsp/jstl/core	<c:out>
I18N 格式標籤庫	fmt	http://java.sun.com/jsp/jstl/xml	<fmt:formatDate>
SQL 標籤庫	sql	http://java.sun.com/jsp/jstl/sql	<sql:query>
XML 標籤庫	xml	http://java.sun.com/jsp/jstl/fmt	<x:forBach>
函式標籤庫	fn	http://java.sun.com/jsp/jstl/functions	<fn:split>

另外， JSTL 也支援EL（Expression Language）語法，例如：我們在一個標準的JSP 頁面中可能會使用到如下的寫法：

```
<%= userList.getUser().getPhoneNumber() %>
```

使用 JSTL 搭配傳統寫法會變成這樣：

```
<c_rt:out value="<%= userList.getUser( ).getPhoneNumber( ) %>" />
```

使用 JSTL 搭配 EL ，則可以改寫成下面的寫法：

```
<c:out value="${userList.user.phoneNumber}" />
```

雖然對網頁設計者來說，如果沒有學過 Java Script 或者是第一次看到這種寫法時，可能仍然會搞不太懂，但是與 Java 語法相比而言，這應該更加容易學習。

7-1-1 安裝使用 JSTL 1.1

JSTL 1.1 必須在支援 Servlet 2.4 且 JSP 2.0 以上版本的 Container 才可以使用。JSTL 主要是由 Apache 組織的 Jakarta Project 所實作的，因此讀者可以到 http://jakarta.apache.org/builds/jakarta-taglibs/releases/standard/ 下載實作好的 JSTL 1.1 ，或者直接使用本書光碟中 JSTL 1.1 ，軟體名稱為：jakarta-taglibs-standard-current.zip 。

下載完後解壓縮，可以發現資料夾中所包含的東西如下：

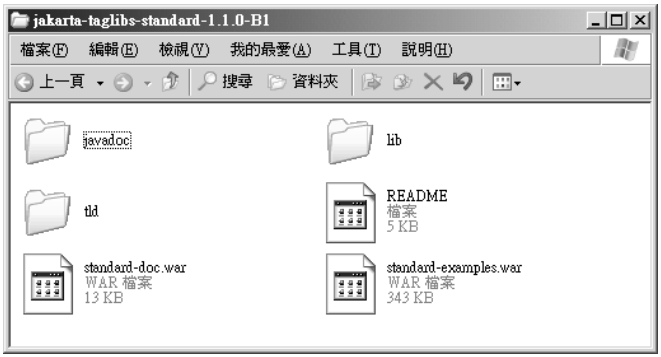

圖 7-1 jakarta-taglibs-standard-1.1.0-B1 的目錄結構

將 lib 中的 jstl.jar 、standard.jar 複製到 Tomcat 的 WEB-INF\lib 中，然後我們就可以在 JSP 網頁中使用 JSTL 了。除了複製 .jar 檔之外，我們最好也把 tld 檔的目錄也複製到 WEB-INF 中，以便日後使用。

注意

lib 目錄下，除了 jstl.jar 和 standard.jar 之外，還有 old-dependencies 目錄，這目錄裡面的東西是讓之前 JSTL 1.0 的程式，也能夠在 JSTL 1.1 環境下使用。tld 目錄下有許多 TLD 檔，其中大部分都是 JSTL 1.0 的 TLD 檔，例如：c-1_0.tld 和 c-1_0-rt.tld 。

接下來我們寫一個測試用的範例程式 - HelloJSTL.jsp ，程式主要是印出你瀏覽器的版本和歡迎的字串。

HelloJSTL.jsp

```
<%@ page contentType="text/html;charset=big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
<title> 測試你的第一個使用到 JSTL 的網頁</title>
</head>
<body>
<c:out value="歡迎測試你的第一個使用到 JSTL 的網頁"/>
</br>
你使用的瀏覽器是:
</br>
<c:out value="${header['User-Agent']}"/>

</body>
</html>
```

在 HelloJSTL.jsp 的範例裡，筆者用到核心標籤庫（Core）中的標準輸出功能和 EL 的 header 隱含物件。首先若要在 JSP 網頁中使用 JSTL 時，一定要先做下面這行宣告：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

這段宣告表示我將使用 JSTL 的核心標籤庫。一般而言，核心標籤庫的前置名稱(prefix)都為 c ，當然你也可以自行設定。不過 uri 在此時，就必須要為 http://java.sun.com/jsp/jstl/core 。

注意

JSTL 1.0 中，核心標籤庫的 uri 預設為 http://java.sun.com/jstl/core ，比 JSTL 1.1 少一個 jsp/ 的路徑。因為 JSTL 1.1 同時支援 JSTL 1.0 和 1.1 ，所以假若核心標籤庫的 uri 為 http://java.sun.com/jstl/core ，你將會使用到 JSTL 1.0 的核心標籤庫。

接下來使用到核心標籤庫中的 out 標籤，印出 value 的值。 ${header['User-Agent']}表示取得表頭裡的 User-Agent 的值，即有關使用者瀏覽器的種類。

```
<c:out value="歡迎測試你的第一個使用到JSTL的網頁" />
<c:out value="${header['User-Agent']}" />
```

HelloJSTL.jsp 的執行結果，如圖 7-2 所示：

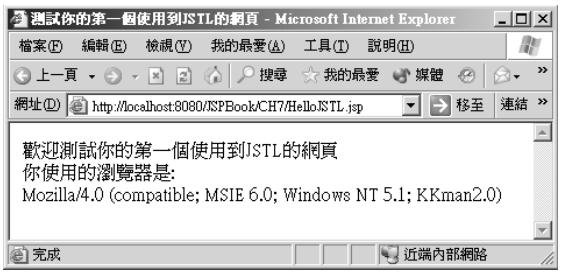

圖 7-2 HelloJSTL.jsp 執行結果

假若讀者想要自訂 taglib 的 uri 時，那就必須在 web.xml 中加入設定值。例如：假若 uri 想要改為 http://www.javaworld.com.tw/jstl/core 時， web.xml 就必須加入此段設定：

```
<web-app>
 :
 [redacted]
  <taglib>
    <[redacted]>http://[redacted]</taglib-uri>
    <[redacted]>/[redacted]</taglib-location>
  </taglib>
 [redacted]
 :
</web-app>
```

在上面的設定中， <taglib-uri> 主要是設定標籤庫的 URI ；而 <taglib-location> 則是用來設定標籤對應的 TLD 檔。因此，我們使用 <%@ taglib %> 指令時，可以直接寫成如下：

```
<%@ taglib[redacted]="c"[redacted]i="[redacted] %>
```

7-1-2 JSTL 1.1 VS JSTL 1.0

JSTL 1.0 更新至 JSTL 1.1 ，有以下幾點不同：

1. EL 原本是定義在 JSTL 1.0 ，現在 EL 已經正式納入 JSP 2.0 標準規格中，所以 JSTL 1.1 規格中，已經沒有 EL 的部分，但是 JSTL 依舊能使用 EL 。
2. JSTL 1.0 中，又有分 EL 和 RT 兩種函式庫，到了 JSTL 1.1 之後，已經不再分這兩種了。以下補充 EL 和 RT 的差別：

EL

- 完全使用 Expression Language
- 簡單
- 建議使用

RT

- 使用 Scriptlet
- Java 語法
- 給不想轉換且習慣舊表示法的開發者

筆者在此強烈建議大家使用 EL 來做，方便又簡單。

3. JSTL 1.1 新增函式(functions)標籤庫，主要提供一些好用的字串處理函式，例如：

   fn:contains 、fn:containsIgnoreCase 、fn:endsWith 、fn:indexOf 、fn:join 、fn:length 、fn:replace 、fn:split 、fn:startsWith 和 fn:substring 等等。

除了上述三項比較大的改變之外，還包括許多小改變，在此不多加說明，有興趣的讀者，可以去看 JSTL 1.1 附錄 B Changes 的部分，那裡有更詳盡的說明。

7-1-3 安裝 standard-examples

當解壓縮 jakarta-taglibs-standard-current.zip 後，資料夾內（參閱圖 7-1）有一個 standard-examples.war 的檔案，將它移至 Tomcat 的 webapps 之後，重新啟動 Tomcat 後，你會發現到，在 webapps 目錄下多一個 standard-examples 的目錄。接下來我們打開 IE ，在 URL 位置上輸入 http://localhost:8080/standard-examples ，你將會看到以下的畫面：

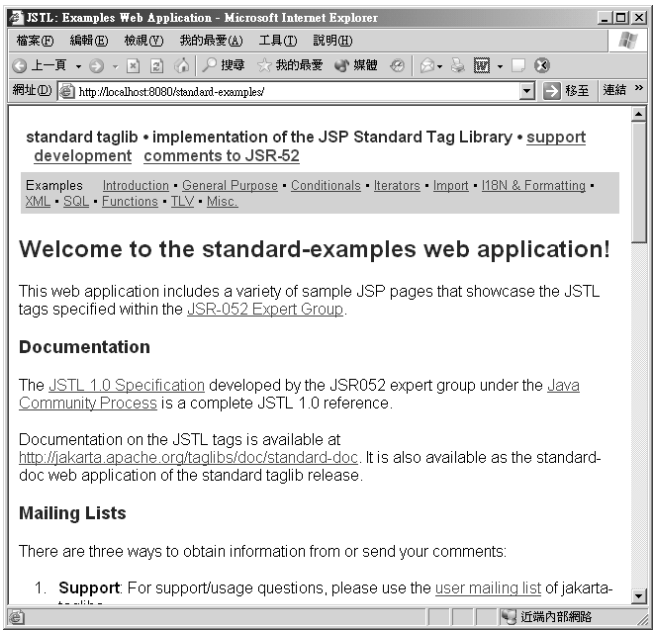

圖 7-3 standard-examples 站台

這個站台有很許多 JSTL 的範例，它包括以下幾個部分：

- General Purpose Tags
- Conditional Tags
- Iterator Tags
- Import Tags
- I18N & Formatting Tags
- XML Tags
- SQL Tags
- Functions
- Tag Library Validators
- Miscellaneous

這些範例程式幾乎涵蓋所有的JSTL 標籤函式庫，假若讀者對於那一個標籤的使用有問題時，可以先來找一找這裡的範例程式，應該或多或少會有所幫助。

7-2 核心標籤庫（Core tag library）

首先介紹的核心標籤庫（Core），它主要有：基本輸入輸出、流程控制、迭代操件和URL 操作。詳細分類如下表所示，接下來筆者將依序為讀者一一介紹每個標籤的功能。

分類	功能分類	標籤名稱
Core	運算式操作	out set remove catch
	流程控制	if choose when otherwise
	迭代操作	forEach forTokens
	URL 操作	import param url param redirect param

在 JSP 中要使用 JSTL 中的核心標籤庫時，必須使用 <%@ taglib %> 指令，並且設定 prefix 和 uri 的值，通常設定如下：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

上述的功用在於宣告將使用 JSTL 的核心標籤庫。

注意

假若沒有上述宣告指令時，將無法使用 JSTL 的核心功能，這點是讀者在使用 JSTL 時，必須要小心的地方。

7-2-1 運算式操作

運算式操作分類中包含四個標籤：<c:out> 、<c:set> 、<c:remove> 和 <c:catch> 。接下來將依序介紹這四個標籤的用法。

<c:out>

<c:out> 主要用來顯示資料的內容，就像是 <%= scripting-language %> 一樣，例如：

```
Hello ! <c:out value="${username}" />
```

語法

語法 1 ：沒有本體（body）內容

```
<c:out value="value" [escapeXml="{true|false}"] [default="defaultValue"] />
```

語法 2 ：有本體內容

<c:out value="value" [escapeXml="{true | false}"]>

default value

</c:out>

屬性

名稱	說明	EL	型態	必須	預設值
value	需要顯示出來的值	Y	Object	是	無
default	如果 value 的值為 null ，則顯示 default 的值	Y	Object	否	無
escapeXml	是否轉換特殊字元，如：< 轉換成 <	Y	boolean	否	true

注意

表格中的 EL 欄位，表示此屬性的值，是否可以為 EL 運算式，例如：Y 表示 attribute = "${運算式}" 為符合語法的， N 則反之。

Null 和 錯誤處理

- 假若 value 為 null 時，會顯示 default 的值；假若沒有設定 default 的值時，則會顯示一個空的字串。

說明

一般來說，<c:out> 預設會將 < 、 > 、' 、" 和 & 轉換為 < 、> 、' 、" 和 & 。假若不想轉換時，只需要設定 <c:out> 的 escapeXml 屬性為 fasle 就可以了。把上述整理成一個表格，方便讀者閱讀：

字元	Entity
<	&lt;
>	&gt;
'	&#039;
"	&#034;
&	&amp;

範例

```
<c:out value="Hello JSP 2.0 !! " />
<c:out value="${ 3 + 5 }" />
<c:out value="${ param.data }" default="No Data" />
<c:out value="<p> 有特殊字元</p>" />
<c:out value="<p> 有特殊字元</p>" escapeXml="false" />
```

1. 在網頁上顯示 Hello JSP 2.0 !!
2. 在網頁上顯示 8 。
3. 在網頁上顯示由表單傳送過來的 data 參數之值，假若沒有 data 參數或 data 參數的值為 null 時，則網頁上會顯示 No Data 。
4. 在網頁上顯示 有特殊字元 。
5. 在網頁上顯示 有特殊字元。

<c:set>

<c:set> 主要用來將變數儲存至 JSP 範圍中或是 JavaBean 的屬性中。

語法

語法 1 ：將 value 的值，儲存至範圍為 scope 的 varName 變數之中

<c:set value="value" var="varName" [scope="{ page | request | session | application }"]/>

語法 2 ：將本體內容的資料，儲存至範圍為 scope 的 varName 變數之中

<c:set var="varName" [scope="{ page | request | session | application }"]>

... 本體內容

</c:set>

語法 3 ：將 value 的值，儲存至 target 物件的屬性中

<c:set value="value" target="target" property="propertyName" />

語法 4 ：將本體內容的資料，儲存至 target 物件的屬性中

<c:set target="target" property="propertyName">

... 本體內容

</c:set>

屬性

名稱	說明	EL	型態	必須	預設值
value	要被儲存的值	Y	Object	否	無
var	欲存入的變數名稱	N	String	否	無
scope	var 變數的 JSP 範圍	N	String	否	page
target	為一 JavaBean 或 java.util.Map 物件	Y	Object	否	無
property	指定 target 物件的屬性	Y	String	否	無

Null 和錯誤處理

- 語法 3 和 4 會產生例外錯誤，有以下兩種情況：
  - target 為 null
  - target 不是 java.util.Map 或 JavaBean 物件

- 假若 value 為 null 時：將由儲存變數改為移除變數
  - 語法 1 ：由 var 和 scope 所定義的變數，將被移除
    - 若 scope 有指定時，則 PageContext.removeAttribute(varName, scope)
    - 若 scope 未指定時，則 PageContext.removeAttribute(varName)
  - 語法 3 ：
    - 假若 target 為 Map 時，則 Map.remove(property)
    - 假若 target 為 JavaBean 時， property 指定的屬性為 null

說明

使用 <c:set> 時， var 主要用來存放運算式的結果 ； scope 則是用來設定儲存的範圍，例如：假若 scope="session"，則將會把資料儲存在 session 中。如果 <c:set> 中沒有指定 scope 時，則它會預設存在 Page 範圍裡。

注意

var 和 scope 這兩個屬性不能使用運算式來表示，例如：我們不能寫成 scope="${ourScope}"或者是 var="${username}" 。

我們考慮下列的寫法：

```
<c:set var="number" scope="session" value="${1 + 1}"/>
```

把 1+1 的結果 2 ，儲存到 number 變數中。如果 <c:set> 沒有 value 屬性時，此時 value 之值為 <c:set> 和 </c:set> 之間本體內容我們看一下下面的範例 ：

```
<c:set var="number" scope="session">
<c:out value="${1+1}" />
</c:set>
```

上面的 <c:out value="${1+1}" /> 部分我們可以改寫成2 或是 <%=1+1%> 結果都會一樣，也就是說 <c:set> 是把本體（body）運算後的結果來當作value 的值。另外， <c:set> 會把 body 中最開頭和結尾的空白部分去掉。如：

```
<c:set var="number" scope="session">
        1 + 1
</c:set>
```

則 number 中儲存的值為 1 + 1 而不是 1 + 1 。

範例

```
<c:set var="number" scope="request" value="${1 + 1}" />
<c:set var="number" scope="session" />
${3 + 5}
</c:set>
<c:set var="number" scope="request" value="${ param.number }" />
<c:set target="User" property="name" value="${ param.Username}" />
```

1. 將 2 存入 Request 範圍的 number 變數中。
2. 將 8 存入 Session 範圍的 number 變數中。
3. 假若 ${param.number}為 null 時，則移除 Request 範圍的 number 變數；若 ${param.number}不為 null 時，則將 ${param.number}的值存入 Request 範圍的 number 變數中。

4. 假若 ${param.Username}為 null 時，則設定 User（JavaBean）的 name 屬性為 null ；若 ${param.Username}不為 null 時，則將 ${param.Username}的值存入 User（JavaBean）的 name 屬性（setter 機制）。

注意

上述範例 3 中，假若 ${param.number}為 null 時，則表示移除 Request 範圍的 number 變數。

<c:remove>

<c:remove> 主要用來移除變數。

語法

```
<c:remove var="varName" [scope="{ page | request | session | application }"] />
```

屬性

名稱	說明	EL	型態	必須	預設值
var	欲移除的變數名稱	N	String	是	無
scope	var 變數的 JSP 範圍	N	String	否	page

說明

<c:remove> 必須要有 var 屬性，即要被移除的屬性名稱，scope 則可有可無，例如：

```
<c:remove var="number" scope="session" />
```

將 number 變數從 Session 範圍中移除。若我們不設定 scope ，則 <c:remove> 將會從 Page 、Request 、Session 然後 Application 的順序尋找是否存在名稱為 number 的資料，若有找到時，則將它移除掉，如果都沒有發現時，則不會做任何的事情。

範例

筆者在這寫一個使用到 <c:set> 和 <c:remove> 的範例，讓讀者可以更快了解如何使用它們，此範例的名稱為 Core_set_remove.jsp 。

Core_set_remove.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_set_remove.jsp</title>
</head>
<body>

<h2><c:out value="<c:set> 和 <c:remove> 的用法" /></h2>

<c:set scope="page" var="number">
<c:out value="${1+1}"/>
</c:set>

<c:set scope="request" var="number">
<%= 3 %>
</c:set>

<c:set scope="session" var="number">
```

```
4
</c:set>

初始設定
<table border="1" width="30%">
<tr>
  <th>pageScope.number</th>
  <td><c:out value="${pageScope.number}" default="No Data" /></td>
</tr>
<tr>
  <th>requestScope.number</th>
  <td><c:out value="${requestScope.number}" default="No Data" /></td>
</tr>
<tr>
  <th>sessionScope.number</th>
  <td><c:out value="${sessionScope.number}" default="No Data" /></td>
</tr>
</table></br>

<c:out value='<c:remove var="number" scope="page" /> 之後'/>
<c:remove var="number" scope="page" />
<table border="1" width="30%">
<tr>
  <th>pageScope.number</th>
  <td><c:out value="${pageScope.number}" default="No Data" /></td>
</tr>
<tr>
  <th>requestScope.number</th>
  <td><c:out value="${requestScope.number}" default="No Data" /></td>
</tr>
<tr>
  <th>sessionScope.number</th>
```

```
  <td><c:out value="${sessionScope.number}" default="No Data" /></td>
</tr>
</table></br>

<c:out value='<c:remove var="number" /> 之後'/>
<c:remove var="number" />
<table border="1" width="30%">
<tr>
  <th>pageScope.number</th>
  <td><c:out value="${pageScope.number}" default="No Data" /></td>
</tr>
<tr>
  <th>requestScope.number</th>
  <td><c:out value="${requestScope.number}" default="No Data" /></td>
</tr>
<tr>
  <th>sessionScope.number</th>
  <td><c:out value="${sessionScope.number}" default="No Data" /></td>
</tr>
</table>
</body>
</html>
```

筆者一開始各在 Page 、Request 和 Session 三個屬性範圍中儲存名稱為 number 的變數。然後先使用 <c:remove var="number" scope="page" /> 把 Page 中的 number 變數移除，最後再使用 <c:remove var="number" /> 把所有屬性範圍中 number 的變數移除。Core_set_remove.jsp 執行結果如下：

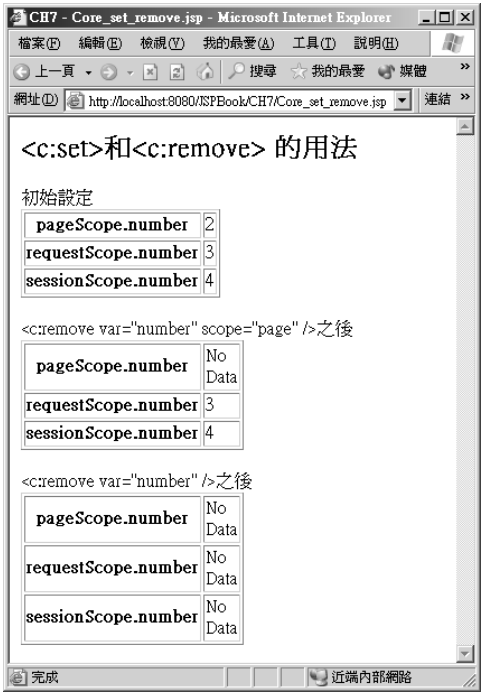

圖 7-4 Core_set_remove.jsp 執行結果

<c:catch>

<c:catch> 主要用來處理產生錯誤的例外狀況，並且將錯誤訊息儲存起來。

語法

```
<c:catch [var="varName"] >
... 欲抓取錯誤的部分
</c:catch>
```

屬性

名稱	說明	EL	型態	必須	預設值
var	用來儲存錯誤訊息的變數	N	String	否	無

說明

<c:catch> 主要將可能發生錯誤的部分放在 <c:catch> 和 </c:catch> 之間。如果真的發生錯誤，可以將錯誤訊息儲存至 varName 變數中，例如：

```
<c:catch var="message">
 :  //可能發生錯誤的部分
</c:catch>
```

另外，當錯誤發生在 <c:catch> 和 </c:catch> 之間時，則只有 <c:catch> 和 </c:catch> 之間的程式會被中止忽略，但整個網頁不會被中止。

範例

筆者寫一個簡單的範例，檔名為 Core_catch.jsp ，來讓大家看一下 <c:catch> 的使用方式

Core_catch.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_catch.jsp</title>
</head>
<body>

<h2><c:out value="<c:catch> 的用法" /></h2>

<c:catch var="error_Message">
<%
```

```
            String eFormat = "not number";
            int i = Integer.parseInt(eFormat);
%>
</c:catch>
${error_Message}
</body>
</html>
```

筆者將一個字串轉成數字，如果字串可以轉為整數，則不會發生錯誤。但是這裡筆者故意傳入一個不能轉成數字的字串，讓 <c:catch> 之間產生錯誤。當錯誤發生時，它會自動將錯誤存到 error_Message 變數之中，最後再用 <c:out> 把錯誤訊息顯示出來。執行結果如圖 7-5 所示：

圖 7-5 Core_catch.jsp 執行結果

我們可以發現到網頁確實印出格式錯誤的訊息。如果我們不使用 <c:catch> ，把範例中的 <c:catch> 和 </c:catch> 拿掉，就會有下面的結果，如圖 7-6 所示：

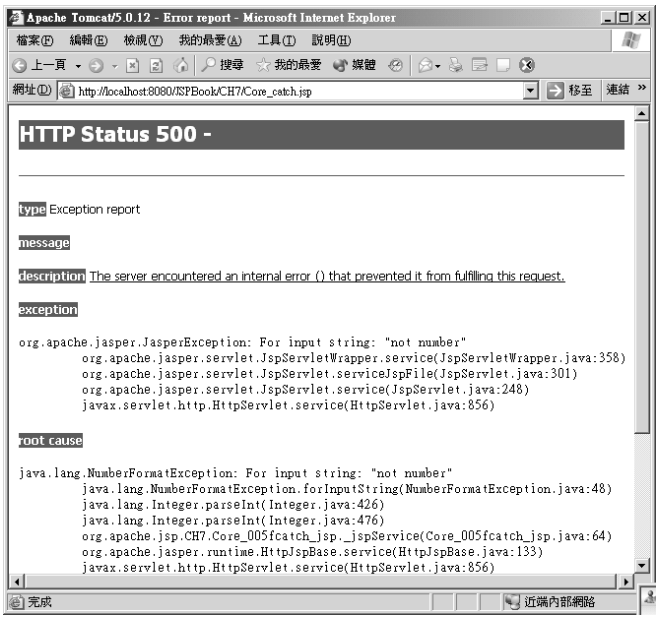

圖 7-6 Core_catch.jsp 沒有 <c:catch> 和 </c:catch> 的執行結果

7-2-2 流程控制

流程控制分類中包含四個標籤：<c:if> 、<c:choose> 、<c:when> 和 <c:otherwise> ，筆者依此順序說明這四個標籤的使用。

<c:if>

<c:if> 的用途就和我們一般在程式中用的 if 一樣。

語法

語法 1 ：沒有本體內容（body）

```
<c:if test="testCondition" var="varName" [scope="{page|request|session|application}"]/>
```

語法 2 ：有本體內容

<c:if test="testCondition" [var="varName"] [scope="{page | request | session | application}"]>

本體內容

</c:if>

屬性

名稱	說明	EL	型態	必須	預設值
test	如果運算式的結果為 true 則執行本體內容， false 則相反	Y	boolean	是	無
var	用來儲存 test 運算後的結果，即 true 或 false	N	String	否	無
scope	var 變數的 JSP 範圍	N	String	否	page

說明

<c:if> 標籤必須要有 test 屬性，當 test 中的運算式結果為 true 時，則會執行本體內容；如果為 false 時，則不會執行。例如： ${param.username == 'admin'}，如果 param.username 等於 admin 時，結果為 true ；若它的內容不等於 admin 時，則為 false 。

接下來我們來看下列的範例 ：

```
<c:if test="${param.username == 'admin' }">
ADMIN 您好!! //body 部分
</c:if>
```

如果名稱等於 admin 則會印出"ADMIN 您好!! // body 部份 "的動作，但是相反則不會執行 <c:if> 的 body 部分，所以不會印出來"ADMIN 您好!! body 部份"。另外 <c:if> 的本體內容除了能放純文字，還可以放任何 JSP 程式碼（Scriptlet）、JSP 標籤或者是 HTML 碼。

除了 test 屬性之外， <c:if> 還有另外兩個屬性 var 和 scope 。當我們執行 <c:if> 的時候，可以將這次判斷後的結果存放到屬性 var 裡； scope 則是設定 var 的屬性範圍。哪些情況才會用到 var 和 scope 這兩個屬性呢？例如：當運算式過長的時，我們會希望拆開處理，或是之後還需要使用此結果時，我們也可以用它先將結果暫時保留，以便之後使用。

範例

筆者寫一個簡單的範例，名稱為 Core_if.jsp 。

Core_if.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_if.jsp</title>
</head>
<body>

<h2><c:out value="<c:if> 的用法" /></h2>

<c:if test="${param.username == 'Admin'}" var="condition" scope="page">
您好 Admin 先生
</c:if></br>

執行結果為:${condition}
</body>
</html>
```

筆者在判斷使用者送來的參數中，如果 username 的值等於 Admin 時則會將 condition 設為 true 並存放於 pageScope 中，否則存放 false 於 condition 中。最後在印出結果。因為 JSTL 會自動找尋 condition 所存在的屬性範圍，因此只需使用 ${condition}，而不用 ${pageScope.condition}。Core_if.jsp 執行結果如圖 7-7 ：

注意

執行本範例時，請在 core_if.jsp 後加上 ?username=Admin 。

圖 7-7 Core_if.jsp 執行結果

<c:choose>

<c:choose> 本身只當做 <c:when> 和 <c:otherwise> 的父標籤。

語法

```
<c:choose>
    本體內容( <when> 和 <otherwise> )
</c:choose>
```

屬性

無

限制

<c:choose> 的本體內容只能有：

- 空白
- 1 或多個 <c:when>
- 0 或多個 <c:otherwise>

說明

若要使用 <c:when> 和 <c:otherwise> 來做流程控制時，它們兩者都必須為 <c:choose> 的子標籤，即：

```
<c:choose>
 :
<c:when>
</c:when>
 :
<c:otherwise>
</c:otherwise>
 :
</c:choose>
```

<c:when>

<c:when> 的用途就和我們一般在程式中用的 when 一樣。

語法

<c:when test="testCondition" >

本體內容

</c:when>

屬性

名稱	說明	EL	型態	必須	預設值
test	如果運算式的結果為 true 則執行本體內容，false 則相反	Y	boolean	是	無

限制

- <c:when> 必須在 <c:choose> 和 </c:choose> 之間
- 在同一個 <c:choose> 中時，<c:when> 必須在 <c:otherwise> 之前

說明

<c:when> 必須有 test 屬性，當 test 中的運算式結果為 true 時，則會執行本體內容；如果為 false 時，則不會執行。

<c:otherwise>

在同一個 <c:choose> 中，當所有 <c:when> 的條件都沒有成立時，則執行 <c:otherwise> 的本體內容。

語法

<c:otherwise>

本體內容

</c:otherwise>

屬性

無

限制

- <c:otherwise> 必須在 <c:choose> 和 </c:choose> 之間
- 在同一個 <c:choose> 中時， <c:otherwise> 必須為最後一個標籤

說明

在同一個 <c:choose> 中，假若所有 <c:when> 的 test 屬性都不為 true 時，則執行 <c:otherwise> 的本體內容。

範例

筆者舉一個典型的 <c:choose> 、<c:when> 和 <c:otherwise> 範例：

```
<c:choose>

<c:when test="${condition1}">
condition1 為 true
</c:when>

<c:when test="${ condition2}">
condition2 為 true
</c:when>
```

```
<c:otherwise>
condition1 和 condition2 都為 false
</c:otherwise>

</c:choose>
```

範例說明：當 condition1 為 true 時，會印出"condition1 為 true"；當 condition1 為 false 且 condition2 為 true 時，會印出"condition2 為 true"，如果兩者都為 false 的話，則會印出"condition1 和 condition2 都為 false"。

注意

假若 condition1 和 condition2 兩者都為 true 時，此時只會印出"condition1 為 true"，這是因為同一個 <c:choose> 下，當有好幾個 <c:when> 都符合條件時，只能有一個 <c:when> 成立。

7-2-3 迭代操作

迭代（Iterate）操作主要包含兩個標籤：<c:forEach> 和 <c:forTokens>，筆者依此順序說明這兩個標籤的使用。

<c:forEach>

<c:forEach> 為迴圈控制，它可以將集合(Collection)中的成員循序瀏覽過一遍。運作方式為當條件符合時，就會持續重複執行 <c:forEach> 的本體內容。

語法

語法 1 ：迭代一集合物件之所有成員

<c:forEach [var="varName"] items="collection" [varStatus="varStatusName"]

[begin="begin"] [end="end"] [step="step"]>

本體內容

</c:forEach>

語法 2 ：迭代指定的次數

<c:forEach [var="varName"] [varStatus="varStatusName"] begin="begin"

end="end" [step="step"]>

本體內容

</c:forEach>

屬性

名稱	說明	EL	型態	必須	預設值
var	用來存放現在指到的成員	N	String	否	無
items	被迭代的集合物件	Y	Arrays Collection Iterator Enumera-tion Map String	否	無
varStatus	用來存放現在指到的相關成員資訊	N	String	否	無
begin	開始的位置	Y	int	否	0
end	結束的位置	Y	int	否	最後一個成員
step	每次迭代的間隔數	Y	int	否	1

限制

- 假若有 begin 屬性時， begin 必須大於等於 0
- 假若有 end 屬性時，必須大於 begin
- 假若有 step 屬性時， step 必須大於等於 0

Null 和錯誤處理

- 假若 items 為 null 時，則表示為一空的集合物件
- 假若 begin 大於或等於 items 時，則迭代不運算

說明

如果要循序瀏覽一個集合物件，並將它的內容印出來，就必須有 items 屬性。

範例

下面的範例 Core_forEach.jsp ，是將陣列中的成員一個個印出來：

Core_forEach.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_forEach.jsp</title>
</head>
<body>

<h2><c:out value="<c:forEach> 的用法" /></h2>
```

```
<%
        String atts[] = new String [5];
        atts[0]="hello";
        atts[1]="this";
        atts[2]="is";
        atts[3]="a";
        atts[4]="pen";
        request.setAttribute("atts", atts);
%>

<c:forEach items="${atts}" var="item" >
${item}</br>
</c:forEach>

</body>
</html>
```

範例中，筆者先產生一個字串陣列，然後將此陣列 atts 儲存至 Request 的屬性範圍中，再用 <c:forEach> 將它循序瀏覽一遍。這裡 items 表示被瀏覽的集合物件，var 為用來存放集合物件中成員，最後使用 <c:out> 將 item 的內容印出來。執行結果如下圖 7-8 所示：

圖 7-8 Core_forEach.jsp 執行結果

注意

varName 的範圍只存在<c:forEach> 的本體中，如果超出了本體，則不能再取得varName 的值。上個例子中，若${item}是在 </c:forEach> 之後執行時，如：

```
<c:forEach items="${atts}" var="item" >
</c:forEach>
${item}</br>
```

${item} 則不會顯示item 的內容。

<c:forEach> 除了支援陣列之外，還有標準 J2SE 的集合類型，例如：ArrayList 、List 、LinkedList 、Vector 、Stack 和 Set 等等；另外還包括 java.util.Map 類的物件，例如：HashMap 、Hashtable 、Properties 、Provider 和 Attributes 。

<c:forEach> 還有begin 、end 和step 這三種屬性：begin 主要用來設定在集合物件中開始的位置(注意：第一個位置為 0)； end 來設定結束的位置；而 step 則是用來設定現在指到的成員和下一個將被指到成員之間的間隔。我們將之前的範例改成如下：

Core_forEach1.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_forEach1.jsp</title>
</head>
<body>
```

```
<h2><c:out value="<c:forEach> begin 、end 和step 的用法" /></h2>

<%
        String atts[] = new String [5];
        atts[0]="hello";
        atts[1]="this";
        atts[2]="is";
        atts[3]="a";
        atts[4]="pen";
        request.setAttribute("atts", atts);
%>

<c:forEach items="${atts}" var="item" begin="1" end="4" step="2" >
${item}</br>
</c:forEach>

</body>
</html>
```

<c:forEach> 中指定的集合物件 atts 將會從第 2 個成員開始到第 5 個成員，並且每執行一次迴圈，都會間隔一個成員瀏覽。因此結果會是只印出 atts[1]和 atts[3]的內容，如下圖 7-9 ：

圖 7-9 Core_forEach1.jsp 執行結果

為了方便詳細介紹 begin 、end 和 step 的不同設定下所產生的結果，筆者將上面的範例改成如下：

```
<%
        int atts[] = {1,2,3,4,5,6,7,8,9,10};
        request.setAttribute("atts", atts);
%>
<c:forEach items="${atts}" var="item" begin="0" end="9" step="1" >
${item}</br>
</c:forEach>
```

這邊筆者改變 begin 、end 和 step 的值時，在網頁上輸出結果的變化。如下表：

begin	end	step	結果
-	-	-	1 2 3 4 5 6 7 8 9 10
5	-	-	6 7 8 9 10
-	5	-	1 2 3 4 5 6
-	-	5	1 6
5	5	-	6
5	5	5	6
0	8	2	1 3 5 7 9
0	8	3	1 4 7
0	8	4	1 5 9
15	20	-	無
20	8	-	空白結果
0	20	-	1 2 3 4 5 6 7 8 9 10

從上表可以發現到：

1. 當 begin 超過 end 時將產生空白結果。

2. 當 begin 雖然小於 end 的值，但是當兩者都大過容器的大小時，將不會輸出任何東西。
3. 最後如果只有 end 的值超過集合物件的大小，則輸出就和沒有設定 end 的情況相同。
4. <c:forEach> 並不只是用來瀏覽集合物件而已，讀者可以從上表中發現，items 並不是一定要有的屬性，但是當沒有使用 items 屬性時，就一定要使用 begin 和 end 這兩個屬性。底下為一個簡單的範例：

Core_forEach2.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_forEach2.jsp</title>
</head>
<body>

<h2><c:out value="<c:forEach> 迴圈" /></h2>

<c:forEach begin="1" end="10" var="item" >
${item}</br>
</c:forEach>

</body>
</html>
```

上述範例中，我們並沒有執行瀏覽集合物件，只有設定 begin 和 end 屬性的值，這樣子它就變成一個普通的迴圈。此範例是將迴圈設定為：從 1 開始跑到 10 ，

總共會重複迴圈 10 次，並且將數字放到 item 的屬性當中。Core_forEach2.jsp 執行結果，如下圖 7-10 ：

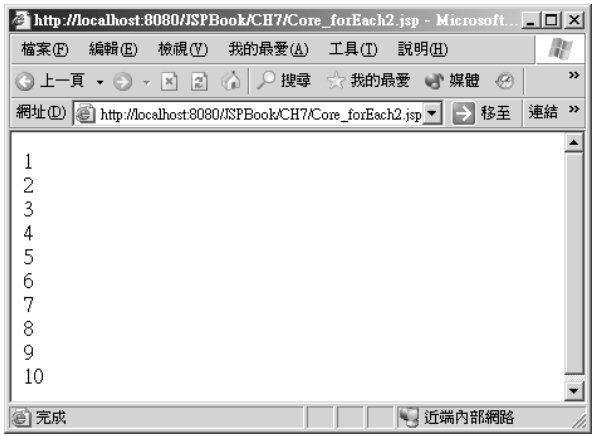

圖 7-10 Core_forEach2.jsp 執行結果

當然它也可以搭配 step 使用，如果將 step 設定為 2 ，結果如下圖 7-11 ：

圖 7-11 當 step 設定為 2 時

另外， <c:forEach> 還提供 varStatus 屬性，主要用來存放現在指到之成員的相關資訊。例如：我們寫成 varStatus="s"，那麼將會把資訊存放在名稱為 s 的屬性當中。varStatus 屬性還有另外四個屬性： index 、 count 、 first 和 last ，它們各自代表的意義如下表：

屬性	型態	意義
index	number	現在指到成員的索引
count	number	總共指到成員的總數
first	boolean	現在指到的成員，是否為第一個成員
last	boolean	現在指到的成員，是否為最後一個成員

我們可以使用 varStatus 屬性，取得迴圈正在瀏覽之成員的資訊，底下為一個簡單的範例：

Core_forEach3.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_forEach3.jsp</title>
</head>
<body>

<h2><c:out value="<c:forEach> varStatus 的四種屬性" /></h2>

<%
        String atts[] = new String [5];
        atts[0]="hello";
        atts[1]="this";
        atts[2]="is";
        atts[3]="a";
        atts[4]="pen";
        request.setAttribute("atts", atts);
%>
```

```
<c:forEach items="${atts}" var="item"varStatus="s">
<h2><c:out value="${item}"/>的四種屬性： </h2>
index ： ${s.index}</br>
count ： ${s.count}</br>
first ： ${s.first}</br>
last ： ${s.last}</br>
</c:forEach>

</body>
</html>
```

結果如下圖 7-12 所示：

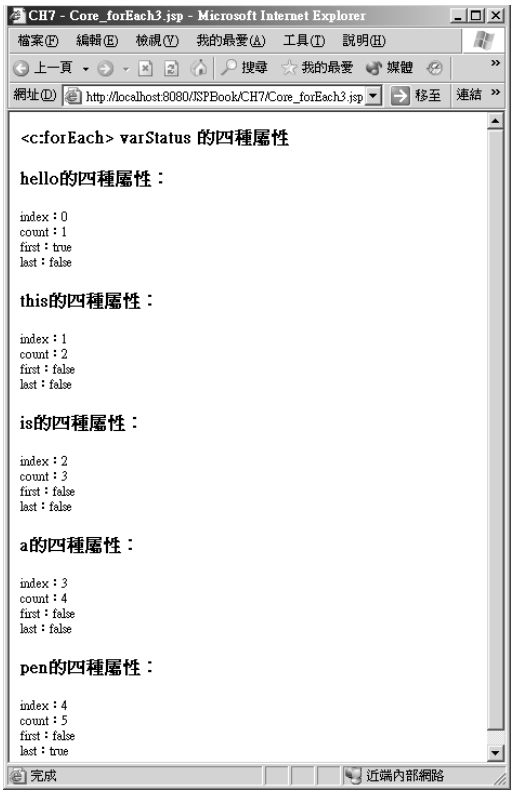

圖 7-12 Core_forEach3.jsp 執行結果

<c:forTokens>

<c:forTokens> 用來瀏覽一字串中所有的成員，字串成員是由定義符號（delimiters）所分隔的。

語法

<c:forTokens items="stringOfTokens" delims="delimiters" [var="varName"]
[varStatus="varStatusName"] [begin="begin"] [end="end"] [step="step"]>
本體內容
</c:forTokens>

屬性

名稱	說明	EL	型態	必須	預設值
var	用來存放現在指到的成員	N	String	否	無
items	被迭代的字串	Y	String	是	無
delims	定義用來分割字串的字元	N	String	是	無
varStatus	用來存放現在指到的相關成員資訊	N	String	否	無
begin	開始的位置	Y	int	否	0
end	結束的位置	Y	int	否	最後一個成員
step	每次迭代的間隔數	Y	int	否	1

限制

- 假若有 begin 屬性時， begin 必須大於等於 0 。
- 假若有 end 屬性時，必須大於 begin 。
- 假若有 step 屬性時， step 必須大於等於 0 。

Null 和錯誤處理

- 假若 items 為 null 時，則表示為一空的集合物件
- 假若 begin 大於或等於 items 的大小時，則迭代不運算

說明

<c:forTokens> 的 begin 、 end 、 step 、 var 和 varStatus 用法都和 <c:forEach> 一樣，因此筆者在這裡就只介紹 items 和 delims 這兩個屬性： items 的內容必須為字串；而 delims 是用來分割 items 中定義的字串之字元。

範例

底下為一個典型的 <c:forTokens> 的範例：

```
<c:forTokens items="A,B,C,D,E" delims="," var="item" >
${item}
</c:forTokens>
```

上面範例執行後，將會在網頁中輸出 ABCDE 。它會把符號(,)當作分割的標記，拆成 5 個部分，也就是執行迴圈 5 次，但是並沒有將 A,B,C,D,E 中的(,)印出來。items 也可以放入 EL 的運算式，如下：

```
<%
  String phoneNumber = "123-456-7899";
  request.setAttribute("userPhone", phoneNumber);
%>
<c:forTokens items="${userPhone}" delims="-" var="item" >
${item}
</c:forTokens>
```

這個範例將會在網頁上列印 1234567899 ，也就是把 123-456-7899 以(-)當作分割標記，將字串拆為 3 份，每次執行一次迴圈就將瀏覽的部分放到名稱為 item 的屬性當中。 delims 不只能指定一種字元來分割字串，它還可以一次設定多個分割字串用的字元。如下面這個範例：

```
<c:forTokens items="A,B;C-D,E" delims=",;-" var="item" >
${item}
</c:forTokens>
```

此範例會在網頁輸出 ABCDE ，也就是說 delims 可以一次設定所有想當作分割字串用的字元。其實用 <c:forEach> 也能做到分割字串，寫法如下：

```
<c:forEach items="A,B,C,D,E" var="item" >
${item}
</c:forEach>
```

上述範例同樣也會在網頁輸出 ABCDE 。<c:forEach> 並沒有 delims 這個屬性，因此 <c:forEach> 無法設定分割字串用的字元，而 <c:forEach> 分割字串用的字元只有(,)，這和使用 <c:forTokens> ， delims 屬性設為(,)的結果相同。所以如果使用 <c:forTokens> 來分割字串，功能和彈性上會比使用 <c:forEach> 來的較大。

7-2-4 URL 操作

JSTL 包含三個與 URL 操作有關的標籤，它們分別為：<c:import> 、<c:redirect> 和 <c:url> 。它們主要的功能是：用來將其他文件的內容包含起來、網頁的導向，還有 url 的產生。筆者將依序介紹這三個標籤。

<c:import>

<c:import> 可以把其他靜態或動態文件包含至本身 JSP 網頁。它和 JSP Action 的 <jsp:include> 最大的差別在於：<jsp:include> 只能包含和自己同一個 web application 下的文件；而 <c:import> 除了能包含和自己同一個 web application 的文件外，亦可以包含不同 web application 或者是其他網站的文件。

語法

語法 1 ：

```
<c:import url="url" [context="context"] [var="varName"] [scope="{page|request|session|application}"] [charEncoding="charEncoding"]>
本體內容
</c:import>
```

語法 2 ：

```
<c:import url="url" [context="context"] varReader="varReaderName"
[charEncoding="charEncoding"]>
本體內容
</c:import>
```

屬性

名稱	說明	EL	型態	必須	預設值
url	一文件被包含的位址	Y	String	是	無
context	相同 Container 下，其他 web 站台必須以 / 開頭	Y	String	否	無
var	儲存被包含的文件的內容（以 String 型態存入）	N	String	否	無
scope	var 變數的 JSP 範圍	N	String	否	Page

名稱	說明	EL	型態	必須	預設值
charEncoding	被包含文件之內容的編碼格式	Y	String	否	無
varReader	儲存被包含的文件的內容（以 Reader 型態存入）	N	String	否	無

Null 和錯誤處理

- 假若 url 為 null 或空的時，會產生 JspException

說明

首先 <c:import> 中必須要有 url 屬性，它是用來設定被包含網頁的位址。它可以為絕對位址或是相對位址，使用絕對位址的寫法如下：

```
<c:import url="http://java.sun.com" />
```

<c:import> 就會把 http://java.sun.com 的內容加到網頁中。

另外 <c:import> 也支援 FTP 協定，假設現在有一個 FTP 站台，位址為 ftp.javaworld.com.tw ，它裡面有一個文件 data.txt ，那麼我們可以寫成如下方式將其內容顯示出來：

```
<c:import url="ftp://ftp.cse.yzu.edu.tw/data.txt" />
```

如果是使用相對位址，假設存在一個文件名為 Hello.jsp ，它和使用 <c:import> 的網頁存在於同一個 webapps 的資料夾時， <c:import> 的寫法如下：

```
<c:import url="Hello.jsp" />
```

如果使用 / 為開頭，那麼就表示跳到 web 站台的根目錄下，以 Tomcat 為例，即 webapps 目錄。假設一個文件為 hello.txt ，存在於 webapps/examples/images 裡，而 context 為 examples ，我們可以寫成以下方式將 hello.txt 文件包含進我們的 JSP 頁面之中：

```
<c:import url="images/hello.txt" />
```

接下來如果要包含同一個伺服器上，但並非同一個web 站台的文件時，就必須加上 context 屬性。假設此伺服器上另外還有一個 web 站台，名為 others ，others 站台底下有一個資料夾為jsp ，且裡面有 index.html 這個文件，那麼我們就可以寫成如下方式將此文件包含進來：

```
<c:import url="/jsp/index.html" context="/others" />
```

注意

被包含文件的web 站台必須在server.xml 中裡定義過，且 <Context> 的 crossContext 屬性值必須為 true ，如此一來， others 目錄下的文件才可以被其他 web 站台呼叫。

server.xml 的設定範例如下：

```
:
<Context path="/others" docBase="others" debug="0"
         reloadable="true" crossContext="true"/>
:
```

除此之外， <c:import> 也提供var 和scope 屬性。當var 屬性存在時，雖然同樣會把其他文件的內容包含進來，但是它並不會輸出至網頁上，而是以 String 的型態，儲存至 varName 中。scope 則是設定 varName 的範圍。儲存之後的資料，我們在需要用時，可以將它取出來，如下：

```
<c:import url="/images/hello.txt" var="s" scope="session" />
```

我們可以把常重複使用的商標、歡迎語句或者是版權宣告，用此方法儲存起來，想輸出在網頁上時，再把它匯入進來。假若想要改變文件內容時，可以只改變被包含的文件，不用修改其他網頁。

另外，可以在 <c:import> 的本體內容中使用 <c:param> ，它的功用主要是：可以將參數遞給被包含的文件，它有兩個屬性 name 和 value ，如下表：

名稱	說明	EL	型態	必須	預設值
name	參數名稱	Y	String	是	無
value	參數的值	Y	String	否	本體內容

這兩個屬性都可以使用 EL ，所以當我們寫成如下：

```
<c:import url="http://java.sun.com" >
<c:param name="test" value="1234" />
</c:import>
```

這樣子的做法等於是包含一個文件，並且所指定的網址會變成如下：

```
http://java.sun.com?test=1234
```

範例

下面為一個使用到 <c:import> 、 <c:param> 及屬性範圍的範例，Core_import.jsp 和 Core_imported.jsp ：

Core_import.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_import.jsp</title>
</head>
```

```
<body>

<h2><c:out value="<c:import> 的用法" /></h2>

<c:set var="input1" value="使用屬性範圍傳到 Core_imported.jsp 中"
scope="request"/>
包含 core_imported.jsp 中 <hr/>

<c:import url="Core_imported.jsp" >
<c:param name="input2" value="使用 <c:param> 傳到 Core_imported.jsp 中"/>
</c:import><hr/>

${output/}

</body>
</html>
```

程式中，筆者分別使用 <c:set> 和 <c:param> 來傳遞參數。

Core_imported.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>

<html>
<head>
  <title>CH7 - Core_imported.jsp</title>
</head>
<body>
```

```
<fmt:requestEncoding value="Big5" />
<c:set var="output1" value="使用屬性範圍傳到Core_import.jsp 中" scope="request"/>
${input}
<c:out value="${param.input2}"escapeXml="true"/>

</body>
</html>
```

core_imported.jsp 是被包含的文件，它會把從 core_import.jsp 傳來的參數分別輸出到頁面上，必須注意的是 input1 參數是使用屬性範圍來傳遞，因此可以直接用 ${input1}，而 input2 則必須使用 ${param.input2}來得到參數。

此外，筆者還使用 <c:set> 來傳遞值給Core_import.jsp ，這就是<c:param> 無法做到的動作， <c:param> 只能從包含端丟給被包含端，但是存在屬性範圍中，可以讓包含端也能得到被包含端傳來的資料。Core_import.jsp 執行結果如下：

圖 7-13 Core_import.jsp 執行結果

<c:url>

<c:url> 主要用來產生一個 URL 。

語法

語法 1 ：沒有本體內容

<c:url value="value" [context="context"] [var="varName"] [scope="{page | request | session | application}"] />

語法 2 ：本體內容代表查詢字串(Query String)參數

<c:url value="value" [context="context"] [var="varName"]

[scope="{page | request | session | application}"] >

<c:param> 標籤

</c:url>

屬性

名稱	說明	EL	型態	必須	預設值
value	執行的 URL	Y	String	是	無
context	相同 Container 下，其他 web 站台必須以 / 開頭	Y	String	否	無
var	儲存被包含文件的內容（以 String 型態存入）	N	String	否	無
scope	var 變數的 JSP 範圍	N	String	否	Page

說明

在這邊筆者直接使用例子來說明。

```
<c:url value="http:// www.javaworld.com.tw " >
<c:param name="param" value="value"/>
</c:url>
```

讀者可以發現 <c:url> 也可以搭配 <c:param> 使用，上面執行結果將會產生一個網址為 http://www.javaworld.com.tw?param=value ，我們更可以搭配著 HTML 的 <a> 使用，如下：

```
<a href="
<c:url value="http:// www.javaworld.com.tw " >
<c:param name="param" value="value"/>
</c:url>"> 台灣 Java 技術論壇</a>
```

另外 <c:url> 還有三個屬性，分別為 context 、var 和 scope 。context 屬性和之前的 <c:import> 相同，可以用來產生一個其他 web 站台的網址。如果 <c:url> 有 var 屬性時，則網址會被存到 varName 中，而不會直接輸出網址。

哪些狀況下才會去使用 <c:url> ？例如：當我們需要動態產生網址時，有可能傳遞的參數不固定，或者是需要一個網址能連至同伺服器的其他 web 站台之文件，而且 <c:url> 更可以將產生的網址儲存起來重複使用。另外，在以前我們必須使用相對位址或是絕對位址去取得需要的圖檔或文件，現在我們可以直接利用 <c:url> 直接從 web 站台的角度來設定需要的圖檔或文件的位址，如下：

```
<img src="<c:url value="/images/code.gif" />" />
```

如此就會自動產生連到 image 資料夾裡的 code.gif 的位址，不再需要耗費精神計算相對位址，並且當網域名稱改變時，也不用再改變絕對位址。

<c:redirect>

<c:redirect> 可以將用戶端的請求，從一個 JSP 網頁導向到其他文件。

語法

語法 1 ：沒有本體內容

```
<c:redirect url="url" [context="context"] />
```

語法 2 ：本體內容代表查詢字串(Query String)參數

```
<c:redirect url="url" [context="context"] >
  <c:param>
</c:redirect >
```

屬性

名稱	說明	EL	型態	必須	預設值
url	導向的目標位址	Y	String	是	無
context	相同 Container 下，其他 web 站台必須以 / 開頭	Y	String	否	無

說明

url 就是設定要被導向到的目標位址，它可以是相對或絕對位址。例如：我們寫成如下：

```
<c:redirect url="http://www.javaworld.com.tw" />
```

那麼網頁將會自動導向到 http://www.javaworld.com.tw 。另外，我們也可以加上 context 這個屬性，用來導向至其他 web 站台上的文件，例如：導向到 /others 下的/jsp/index.html 時，寫法如下：

```
<c:redirect url="/jsp/index.html" context="/others" />
```

<c:redirect> 的功能不止可以導向網頁，同樣它還可以傳遞參數給目標文件。在這裡我們同樣使用 <c:param> 來設定參數名稱和內容。

範例

Core_redirect.jsp

```
<%@ page contentType="text/html;charset=Big5" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
<head>
  <title>CH7 - Core_redirect.jsp</title>
</head>
<body>

<h2><c:out value="<c:redirect> 的用法" /></h2>

<c:redirect url="http://java.sun.com">
<c:param name="param" value="value"/>
</c:redirect>

<c:out value="不會執行喔!!!" />

</body>
</html>
```

執行結果如下圖，可以發現網址部分為 http://java.sun.com/?param=value ：

圖 7-14 Core_redirect.jsp 執行結果