

如何開發 Servlet ServletConfig 與 ServletContext



段維瀚 老師



綱 要

- 一、**ServletConfig** 介面
- 二、**ServletContext** 介面
- 三、**Servlet** 彼此分享資源的方法
- 四、協調行程 **RequestDispatch**



綱 要

- 一、**ServletConfig** 介面
- 二、**ServletContext** 介面
- 三、**Servlet** 彼此分享資源的方法
- 四、協調行程 **RequestDispatch**



一、*ServletConfig* 介面

- *ServletConfig interface* 是定義在
 - *javax.servlet package*
 - *servlet container* 會呼叫 *init(ServletConfig config)* 並將組態 *config* 傳遞給 *GenericServlet*，接下來讓我們進一步來了解 *ServletConfig* 提供了哪些實用的方法。
 - 利用 *ServletConfig* 所提供的方法來取得初始參數值。



一、*ServletConfig* 介面

- *ServletConfig* 介面所提供的 4 個方法：

方法	描述
<i>String getInitParameter(String name)</i>	取得參數 name 的內容
<i>Enumeration getinitParameterNames()</i>	取得所有參數名稱
<i>ServletContext getServletContext()</i>	抓得 ServletContext 物件
<i>String getServletName()</i>	取得 Servlet 的名稱

ServletConfig 僅提供 **getXXX()** 方法，您無法增加或修改任何屬於 **ServletConfig** 物件中的參數。



一、*ServletConfig* 介面

```
<servlet>
    <servlet-name>DBConnectionServlet</servlet-name>
    <servlet-class>chapter4.DBConnectionServlet</servlet-class>
    <init-param>
        <param-name>driverClassName</param-name>
        <param-value>com.microsoft.jdbc.sqlserver.SQLServerDriver</param-value>
    </init-param>
    <init-param>
        <param-name>dbURL</param-name>
        <param-value>
            jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=pubs
        </param-value>
    </init-param>
    <init-param>
        <param-name>userName</param-name>
        <param-value>sa</param-value>
    </init-param>
    <init-param>
        <param-name>pwd</param-name>
        <param-value></param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

第一時間載入

一、*ServletConfig* 介面

- *DBConnectionServlet.java*

```
public void init() {  
    // ... block of code  
    ServletConfig config = getServletConfig();  
    String driverClassName =  
        config.getInitParameter("driverClassName");  
    String dbURL = config.getInitParameter("dbURL");  
    String userName = config.getInitParameter("userName");  
    String pwd = config.getInitParameter("pwd");  
    // ... block of code  
}  
public void doGet() { // 自行實作}  
public void destroy() { // 自行實作}
```

取得 **ServletConfig** 組態

利用
getInitParameter()
取得 **web.xml** 中所設定
的參數內容值

綱 要

- 一、**ServletConfig** 介面
- 二、**ServletContext** 介面
- 三、**Servlet** 彼此分享資源的方法
- 四、協調行程 **RequestDispatch**



二、*ServletContext* 介面

- *ServletContext interface* 是定義在 *javax.servlet package* 中：
 - *ServletContext interface* 定義了一系列與 *servlet container* 溝通的方法，藉此了解 *servlet* 與 *servlet container* 的環境資訊。
 - 每個 *web application* 都一定有一個且只有一個(*only one*) *ServletContext*。



二、*ServletContext* 介面

- 獲取資源

- *getResource()* & *getResourceAsStream()*

- *URL getResource(String path)* :

- 擷取 *path* 路徑所指向的資源。
 - 限制：*path* 必須以 “/” 開頭、所指向的資源不能是 *jsp file* 否則會發生 *unprocessed data* 的例外，這是因為 *jsp* 是屬於 *active resource* 所以 *jsp* 的原始碼時是受保護的。不過可以利用 *getRealPath()* 來解決這個問題。

- *InputStream getResourceAsStream(String Path)*

- 擷取 *path* 路徑(*URL*)所指向的資源串流。
 - *getResource().openStream = getResourceAsStream()*



綱 要

- 一、**ServletConfig** 介面
- 二、**ServletContext** 介面
- 三、**Servlet** 彼此分享資源的方法
- 四、協調行程 RequestDispatch



三、*Servlet* 彼此分享資源的方法

- 在 *Servlet* 中屬性資料會以 3 中不同的模式儲存於 *container* 中，而每一種模式都賦予不同的生命週期。
 - 這 3 種模式分別是 *ServletRequest*、*HttpSession* 與 *ServletContext*。
 - 下面將說明資料儲存物件與該資料的可視範圍(*scope*)：
 - *ServletRequest*：存在於 *request* 的生命週期。
 - *HttpSession*：相同的 *active client* 可以存取。
 - *ServletContext*：存在於 *active web application* 中，當下所有 *servlet* 皆可以存取。



三、*Servlet* 彼此分享資源的方法

- *ServletContext interface* 提供下列幾種方法來存取這些資源：

方法	說明
<i>Object getAttribute(String name)</i>	取得 name 的內容。
<i>Enumeration getAttributeNames()</i>	取得 contener 中所有可用的資源名稱。
<i>void setAttribute(String name, Object value)</i>	新增或修改資料的內容。

綱 要

- 一、**ServletConfig** 介面
- 二、**ServletContext** 介面
- 三、**Servlet** 彼此分享資源的方法
- 四、協調行程 **RequestDispatch**



四、協調 *Servlet* 行程 - *RequestDispatcher*

- 協調/分派 *Servlet*：
 - 在 *Servlet* 中我們可以利用 *RequestDispatcher* 來將目前所接收到的 *Request* 分派到另一個所指定的資源(*Active JSP* 或 *Servlet* 等...)中。
 - 首先我們可以利用 *getRequestDispatcher()* 來取得 *ServletContext* 的 *RequestDispatcher* 物件。
 - *RequestDispatcher rd = context.getRequestDispatcher("/servlet/IncludedPage");*



四、協調 *Servlet* 行程 - *RequestDispatcher*

- *ServletContext* 與 *ServletRequest* 都各自有提供 *getRequestDispatcher()* 方法：
 - 目的都在取得 *RequestDispatcher* 物件。
 - *ServletRequest.getRequestDispatcher(path)* 中的 *path* 參數是以“*..*/”開頭的路徑，例如：*request.getRequestDispatcher(../XXX")*
 - *ServletContext.getRequestDispatcher(path)* 中的 *path* 參數是以“/”開頭的路徑，例如：*request.getRequestDispatcher("/XXX")*。



四、協調 *Servlet* 行程 - *RequestDispatcher*

- *forward()* 與 *include()* 方法：

- 她們是 *RequestDispatcher* 介面所提供的方法。
- *forward()* 簡單的說就是將未完成的工作交給另一個 *Servlet* 來繼續執行，而該 *Servlet* 將可以繼續維護原始 *Servlet* 所屬的 *request* 與 *response* 物件。但最後主導權會是在被 *forward* 的 *Servlet*。
 - 當 *Servlet* 欲呼叫 *forward()* 方法前不能有 *response committed* 的動作。
- *include()* 顧名思義就是加入包含之意，目的就是 *Servlet* 在執行的過程中可以利用 *include()* 方法加入其他的 *Servlet* 一同工作，最後主導權則仍然還是在原來的 *Servlet*。



四、協調 *servlet* 行程 - *RequestDispatch*

- *forward()* v.s. *sendRedirect()*

- *forward()* → *RequestDispatch.forward()* 是 *RequestDispatch* 下的方法。
- *sendRedirect()* → *HttpServletResponse.sendRedirect()* 是 *HttpServletResponse* 下的方法。
- 因為 *forward()* 是屬於 *Server-Side* 的機制所以 *servlet* 所屬的 *request* 與 *response* 可以繼續維護。
- 而 *sendRedirect* 在機制上是屬於 *client-side*，目的是將重導位置直接丟給 *browser* 所以無法繼續維護 *request* 與 *response*。

