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

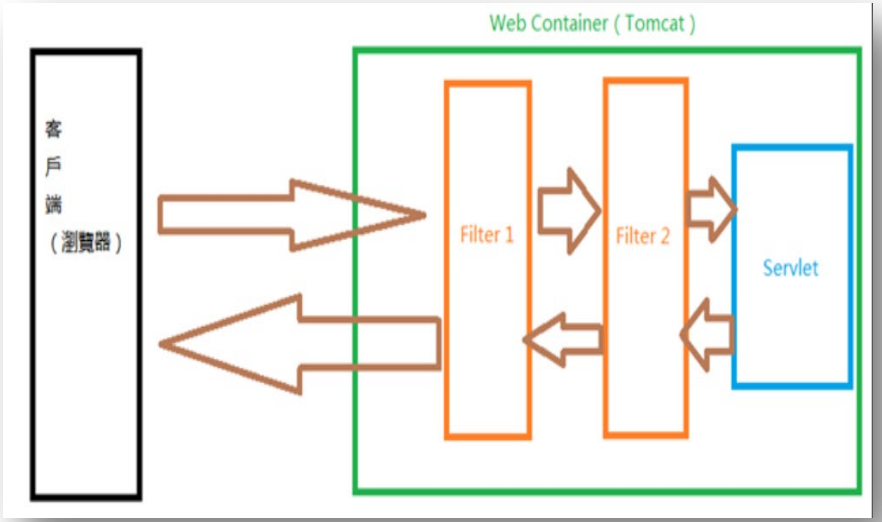
# 第五堂 Web Interceptor應用-Filter實作

# 本堂教學重點

- Filter生命週期設定與初始化組態應用
- Filter配合RequestDispatcher應用
- 多層Filter攔截架構設計

# Filter生命週期設定與初始化組態應用

- Filter介面，用來設定網站系統的攔截器(Inteceptor)處理機制。
- ServletRequest與ServletResponse都會經過(Passing)佈置在網站系統上相關Filter。
- 作業方式需求
  - 用在安全性或者稽核等作業上。
  - 檔案下載壓縮作業。
  - 傳遞內容編碼規則。
  - 傳遞物件的序列化與反序列化。
  - 有些Filter設定在Web Server上進行。
- 屬於應用系統範圍設定。



# Filter介面

- Filter Instance必須實作在javax.servlet.Filter介面，
- 介面中有三個必須實作的方法：
  - init()
  - destory()
  - doFilter()
- init()是Filter物件被載入時執行的方法，而destory()是 Filter物件釋放時執行的方，doFilter()則是實作Filter功能的核心，所需要進行的處理程序都需在doFilter() 方法中完成。

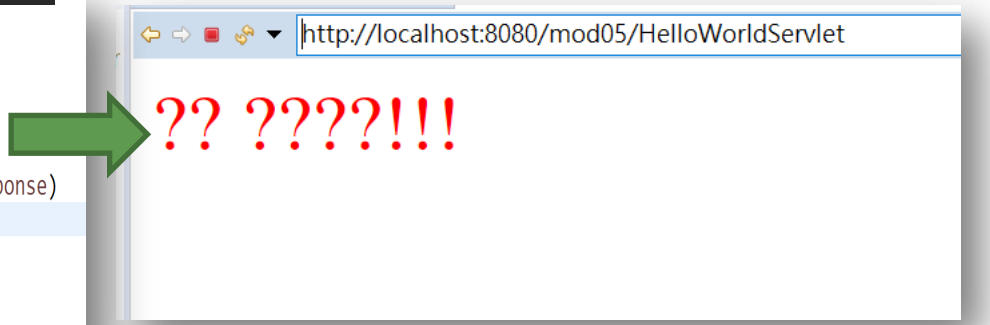
| 方法 | 述 |
| --- | --- |
| init(FilterConfig config) | Filter物件初始化所引發的事件程序。 |
| doFilter(ServletRequest request,ServletResponse response,FilterChain chain) | 每當客戶端通過請求/回應對應鏈(Chain)中的資源請求時，都會通過容器調用布置的Filter的doFilter方法執行。 |
| destroy() | Filter物件釋放時引發的事件。 |

# Filter介面應用-設定網站Encoder

- 網站撰寫Servlet物件，常常需要設定ServletRequest與ServletResponse編碼規則(Encoding)。
- 是否可以交由一個Filter佈署，進行整個網站請求與回應的統一編碼規則設定。

HelloWorldServlet.java

```java
@WebServlet("/HelloWorldServlet")
public class HelloWorldServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out=response.getWriter(); //參考出Writer
        out.println("<font size='6' color='red'>您好 世界和平!!!</font>");

    }


}
```




產生編碼不對應的問題

# 設計實作Filter介面的Encoder類別

- 實作Filter Interface的Encoder類別設計。
- 採用web.xml佈署Filter，聆聽整個網站進行ServletRequest與
- ServletResponse編碼。
- 實作doFilter，借助傳遞進來的ServletRequest與ServletResponse進行編碼設定。
- 最後必須執行FilterChain.doFilter()，進行連鎖指向的作業，否則整個請求會卡在這裡，使用者端沒有任何回應訊息。

EncoderHandler.java

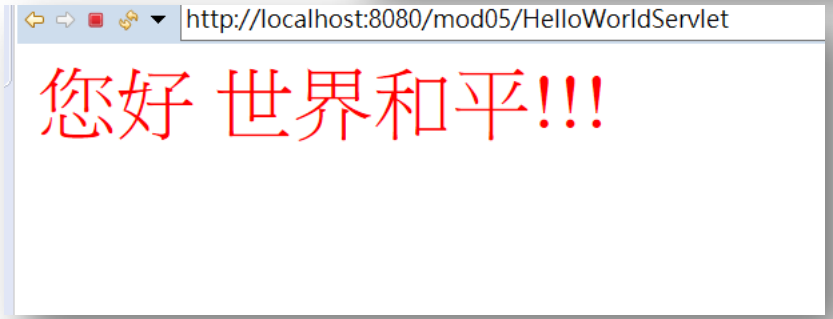
```
//實作Filter介面
public class EncoderHandler implements Filter {

    //保留空參數建構子
    public EncoderHandler() {


    public void destroy() {
        // TODO Auto-generated method stub
    }

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException {
        //設定request內容擷取編碼
        request.setCharacterEncoding("UTF-8"); //採用萬碼編碼
        response.setCharacterEncoding("UTF-8");
        chain.doFilter(request, response);//往下交付
    }

    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }
```

# 採用web.xml(DD File)佈署Filter

- 採用<filter>與<filter-mapping>兩個元素進行佈署與描述
- 使用<filter-mapping>描述的是篩選的範圍與方式
  - /* 指定網站根目錄下進行篩選。

```
<!-- 佈署Filter -->
<filter>
  <filter-name>encoding</filter-name>
  <filter-class>com.gjun.filter.EncoderHandler</filter-class>
</filter>
<!-- 過濾的url pattern設定 -->
<filter-mapping>
  <filter-name>encoding</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

針對者整個網站系統進行攔截處理

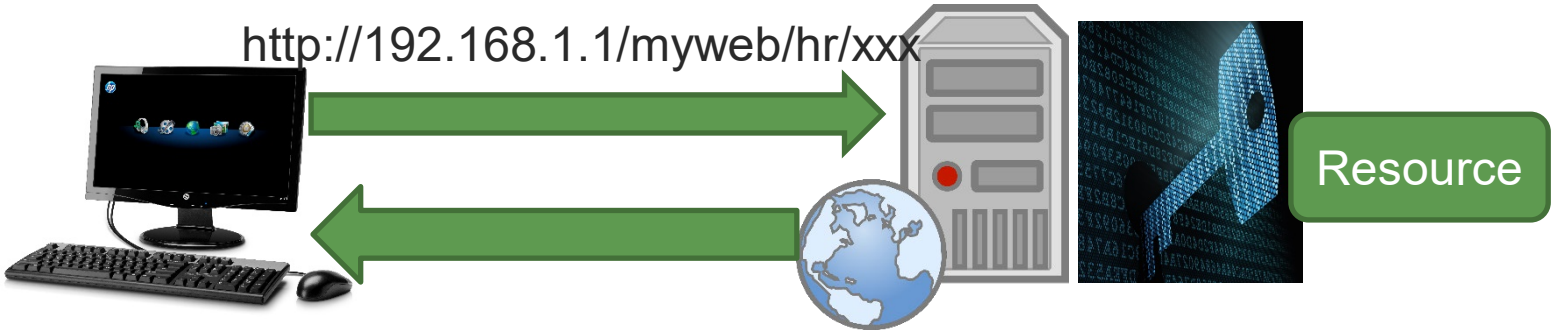


# 採用@WebFilter Annoation進行佈署

- @WebFilter不同的設定方式
  - @WebFilter("/*")
  - @WebFilter(servletNames={“Encoder"})
  - @WebFilter(
    urlPatterns={“/*”},
    initParams={
    @WebInitParam(name = “參數名稱1”, value = “參數值"),
    @WebInitParam(name = "參數名稱2", value = "參數值")
    })
- Filter 的執行順序建議使用 web.xml 的配置 ， 透過 @WebFilter 沒有執行順序的配置設定。

# 網站特定目錄下的安全性控制

- 撰寫一個針對網站入口hr(人事資源)的目錄下的所有檔案存取進行控制
- 需要具有驗證過後的傳遞在前端的憑證(Cookie)進行通行驗證
- 透過實作Filter介面的類別進行佈署與過濾安全性規則



# 設計一個擷取前端Cookie憑證的Filter

- 實作Filter介面，進行前端瀏覽器的Cookie內容擷取。判斷是否具有一個Cookie name-hr的憑證內容。
- 進行通行HR(人事資源)目錄下的網頁之前的安全性篩選作業。
- 如果驗證通過，則採用FilterChain.doFilter()交付往下走的鏈條。
- 驗證不通過則卡在這裡，使用端沒有任何畫面訊息。

HRSecurityHandler.java

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
    //參考出前端的Cookie
    boolean r=false;
    Cookie[] cookies=((HttpServletRequest)request).getCookies();
    if(cookies!=null) {
        //掃描Cookie是否具有相關的憑證
        for(Cookie cookie:cookies) {
            //前端憑證Cookie name定義為cred
            if(cookie.getName().equals("cred")) {
                r=true;
                break;
            }
        }
    }
    //判斷是否有憑證 如果有往下鏈結
    if(r) {
    chain.doFilter(request, response);
    }
}
```

# 模擬一個login登入作業與發出Cookie憑證

- 設計一個login.jsp表單頁面
- 請求一個Servlet進行驗證
- 驗證通過發出一個Cookie到前端瀏覽器(In-memory)

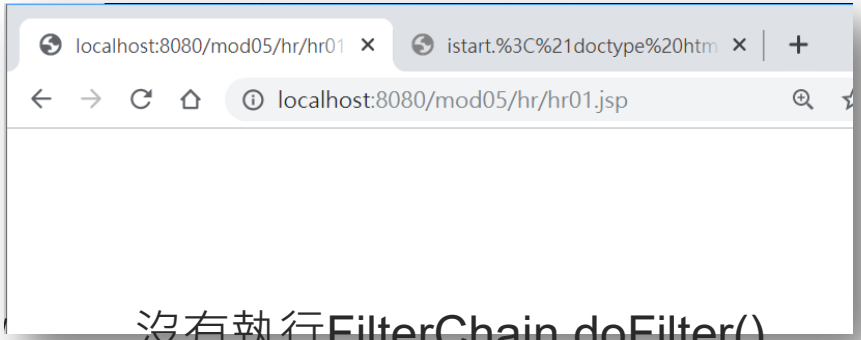
login.jsp

```
<body>
    <form method="post" action="ValidController">
        <div>使用者名稱</div>
        <input type="text" name="username"/>
        <div>密碼</div>
        <input type="password" name="password"/>
        <input type="submit" value="登入"/>
    </form>
</body>
```

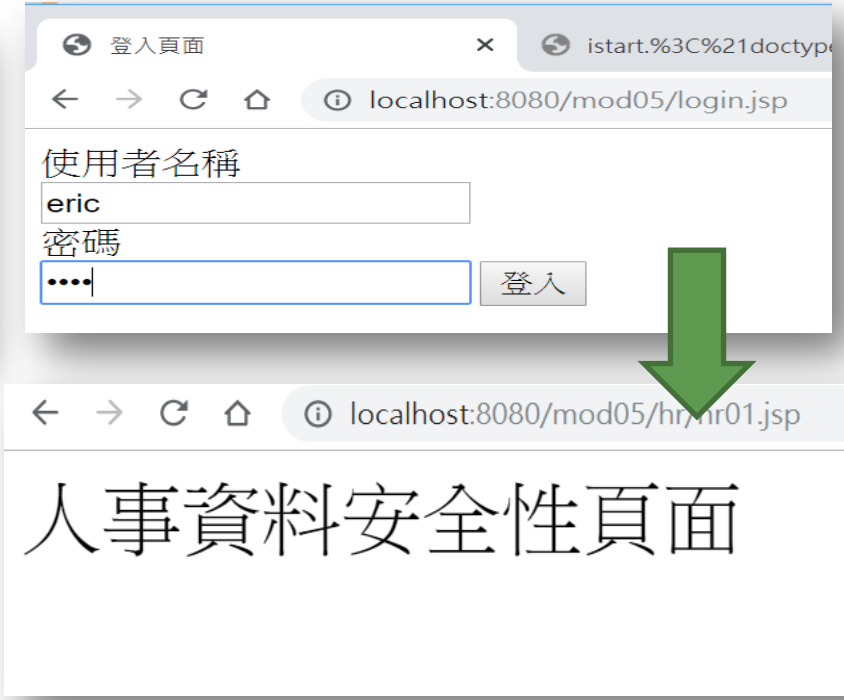
ValidController.java

```
@WebServlet("/ValidController")
public class ValidController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //取出使用者名稱與密碼
        String username=request.getParameter("username");
        String password=request.getParameter("password");
        //進行使用者驗證
        //發出前端Cookie憑證
        Cookie cookie=new Cookie("cred",username);
        response.addCookie(cookie);
        response.getWriter().println("驗證通過了!!");
    }

}
```

# login登入作業 Demo

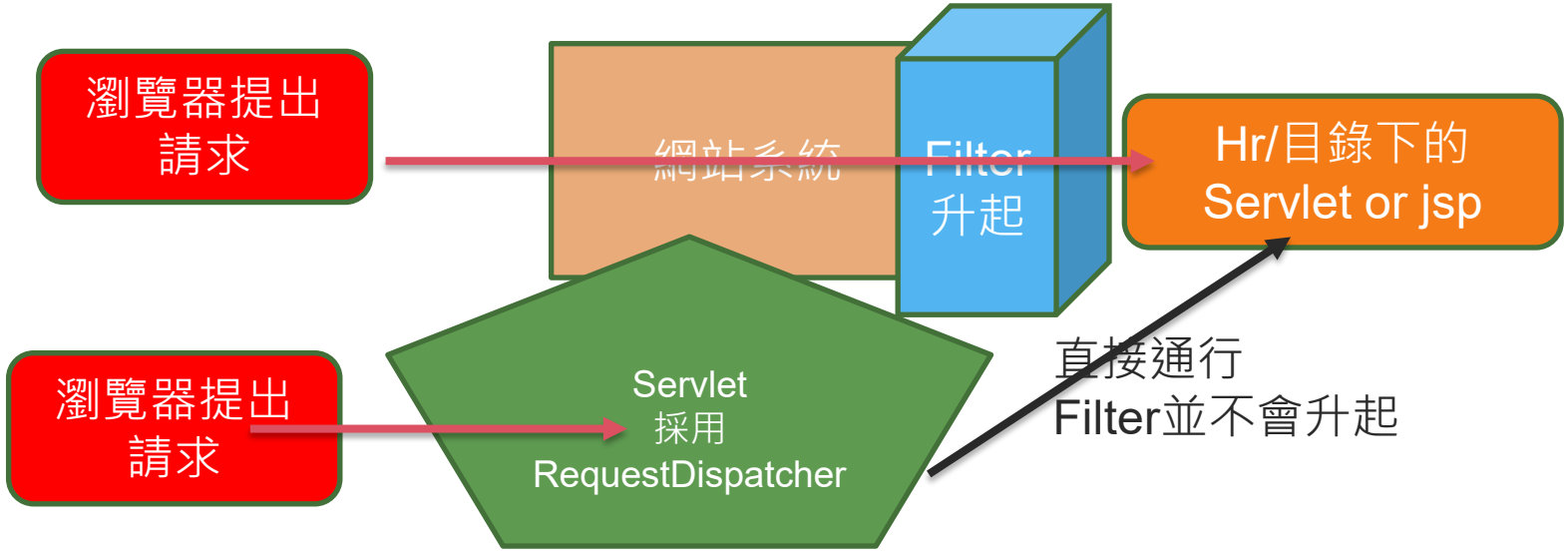



沒有執行FilterChain.doFilter()
卡住了



# Filter配合RequestDispatcher應用

- Filter可以透過<filter-mapping>設定網站過濾(引發)Filter，進行攔截的目錄(Path)。
- 我們在Servlet或者JSP運行中，可以使用RequestDispatcher進行分派到指定的目標，而這樣的分派方式預設不受Filter Mapping指定的Path加以控制。



# 同一個Filter是否可以佈署不同的篩選目錄架構

- 使用<dispatcher>設定FORWARD/INCLUDE與REQUEST安全性攔截機制

```
@WebFilter(
        urlPatterns = { "/hr/*" },
        initParams = {
                @WebInitParam(name = "target", value = "人事資源系統")
        },
        dispatcherTypes= {DispatcherType.REQUEST,
                DispatcherType.FORWARD,
                DispatcherType.INCLUDE}
        )
public class HRSecurityHandler implements Filter {
```

# 多層Filter攔截架構設計

- Filter介面攔截可以多重設定，但不能使用Annotation進行。需要使用web.xml進行順序性設定。
- 多重的Filter之間具有執行順序。
- 第一層進行Request與Response傳輸編碼設定。
- 第二層進行hr資料夾下的請求安全性驗證

```
<!-- 過濾的url pattern設定 -->
<filter-mapping>
  <filter-name>encoding</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>hrvalid</filter-name>
  <url-pattern>/hr/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```

# Lab

- Filter介面的生命週期為何
- 設計一個Filter進行網站編碼設定
- 如何採用@Filter進行Filter佈署
- 設計一個Filter進行網站安全性驗證作業