

# Session 管理機制



段維瀚 老師





# 綱 要

- 一、HttpSession 運作原理
- 二、如何使用 HttpSession ?
- 三、HttpSession 監聽器與事件
- 四、管理 Session Tracking 的技術





# 綱 要

- 一、HttpSession 運作原理
- 二、如何使用 HttpSession ?
- 三、HttpSession 監聽器與事件
- 四、管理 Session Tracking 的技術





# 使用 Cookie 儲存用戶資料

- Cookie 藉由 Web 伺服器的回應送出
- Cookie 在用戶端電腦儲存
- Cookie 用不同Web伺服器的網域名稱來區隔儲存
- Cookie 在請求網頁時會自動送出
- Cookie 有使用期限
- Cookie 是 HTTP-Only 加強安全性



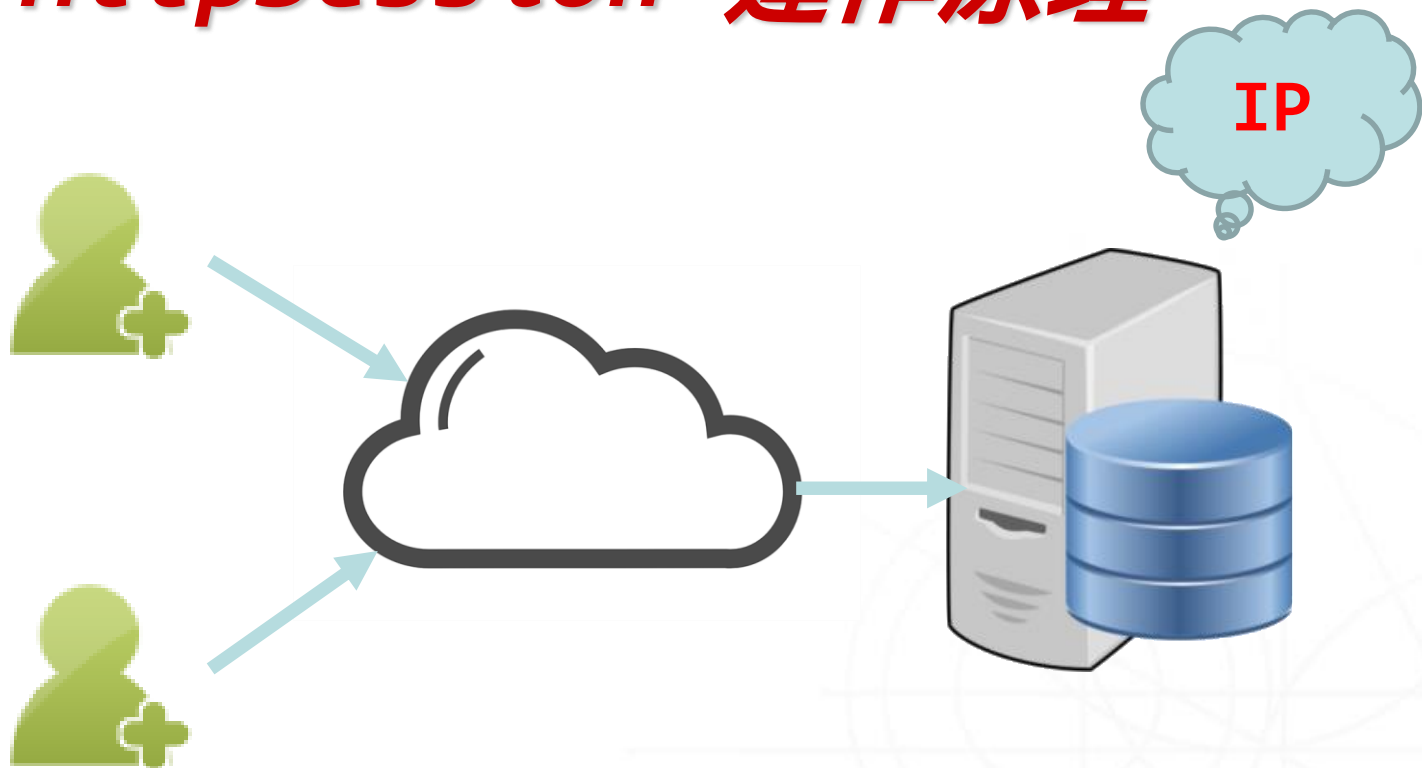


# 使用 Cookie 儲存用戶資料限制

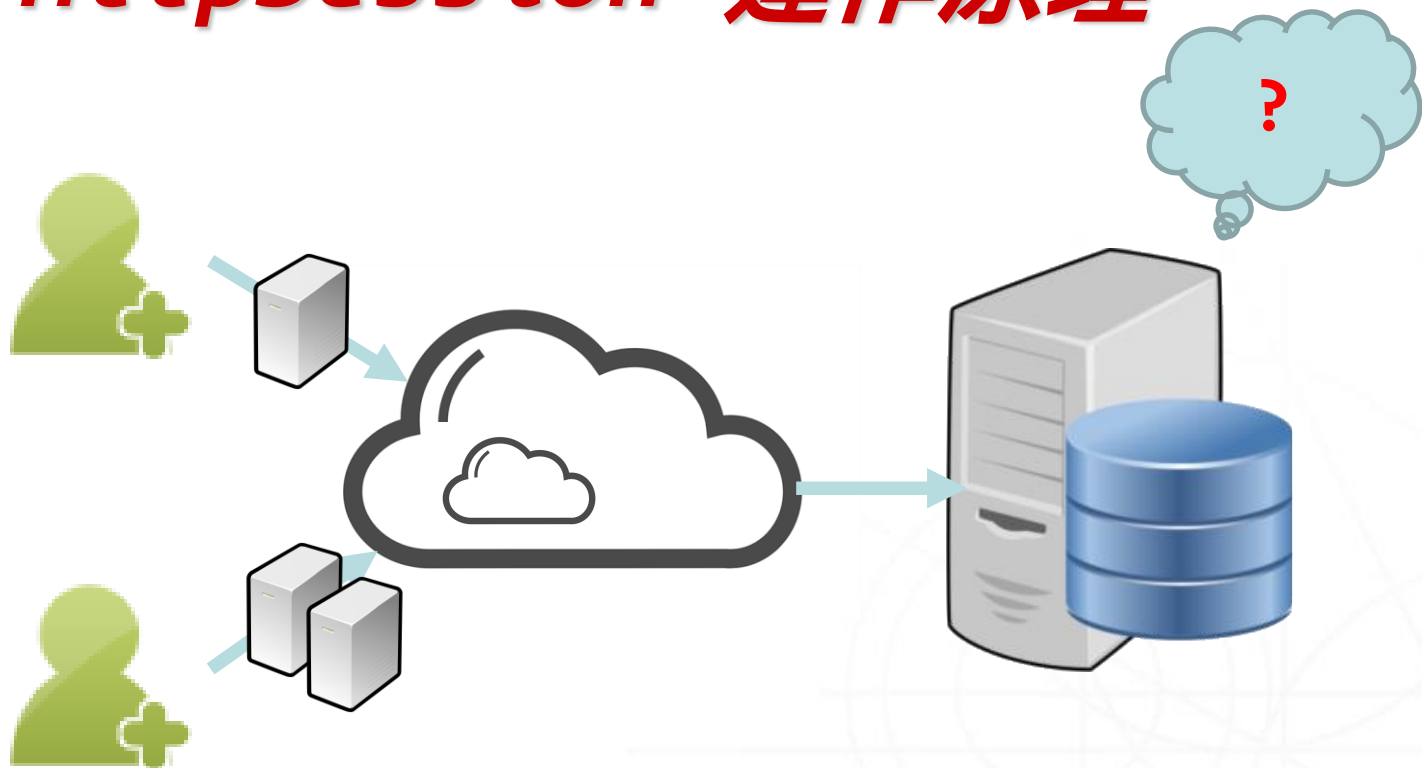
- 每個網站可儲存 20 個
- 每個 Cookie 的大小不超過 4KB
  - 不建議存放大筆資料
- 以上跟瀏覽器預設限制有關



# 一、HttpSession 運作原理



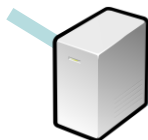
# 一、HttpSession 運作原理



# 一、HttpSession 運作原理

存入  
Cookies

A001



自動  
產生

A001





# 一、HttpSession 運作原理

A001



A001



# 一、HttpSession 運作原理

A001

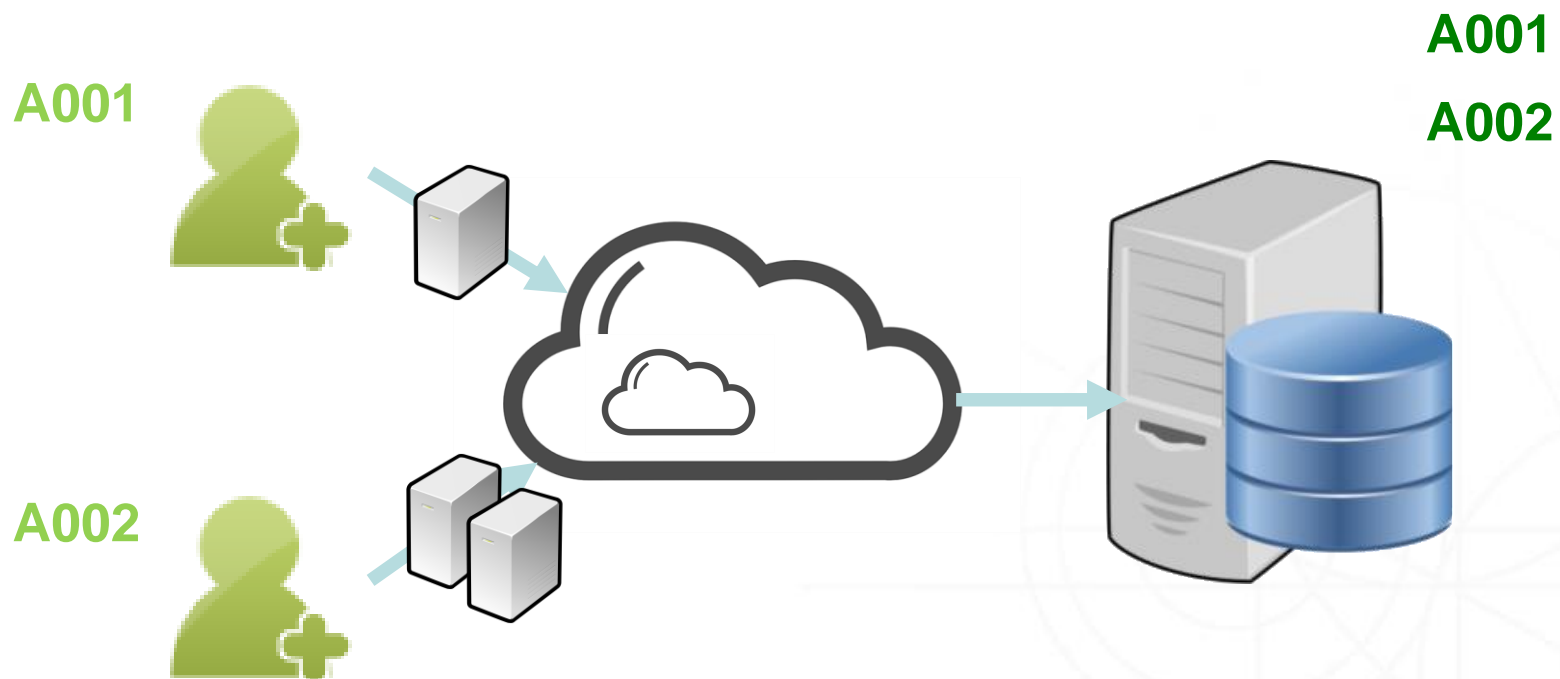


Timeout

~~A001~~



# 一、HttpSession 運作原理





# 一、HttpSession 運作原理

- Session 與 cookies :

- 當 Server 產生 SessionID 時，接下來會將她傳給 client 端，client 端接到後就會儲存在 cookie 中，以便日後提取之用。

- Client 端與 Server 端如何傳遞 SessionID ??

- 當建立 session 完成時，server 回應給 client 端的 HTTP headers 上就會增加一項特殊列：

`cookie=JSESSIONID=4823565E3AA9A1F31D9DF5AFF2003BAF`

這一行 cookie 就是 server 藉由 HTTP headers 傳遞給 client 端的 sessionID，並由 client 接到後寫入 cookie。

- 接下來只要 client 與 server 端是利用 HTTP 協定溝通，一定都會包含這一行 HTTP headers 的特殊列 (cookie=XXX)。



# 一、HttpSession 運作原理

## • HTTP Headers :

```
accept = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,  
application/x-shockwave-flash, application/vnd.ms-excel,  
application/vnd.ms-powerpoint, application/msword, */*  
referer = http://localhost:8080/SCWCD/  
accept-language = zh-tw  
accept-encoding = gzip, deflate  
user-agent = Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;  
(R1 1.3))  
host = localhost:8080  
connection = Keep-Alive  
cookie = JSESSIONID=227FB44B5CD604C26169276D56B5EB08
```

*SessionID*



# 綱 要

- 一、HttpSession 運作原理
- 二、如何使用 HttpSession ?
- 三、HttpSession 監聽器與事件
- 四、管理 Session Tracking 的技術





# 一、如何使用 *HttpSession* ?

- *Session* (連線階段)是一種在 *request* 與 *response* 之間一系列連續不間斷的機制，以加強 *client* 與 *server* 端之間的溝通能力，用意是在彌補 *HTTP* 協定中 *stateless* 的特性。
- 當 *client* 端與 *Server* 第一次建立連線且要求 *session* 需求時，之後 *server* 端在初始時建立 *session* 的同時會產生一組該 *session* 的 *session ID* 並且傳回給 *client* 端，之後 *client* 端與 *server* 就根據此 *session ID* 來彼此溝通。





# 一、如何使用 *HttpSession* ?

- *Session ID* :

- 是一組由 *server* 所產生的唯一識別碼 (*unique identifier*) 。
- *Client* 端接收到 *session ID* 時預設會將此 *session ID* 存放到 *cookies*，以便 *server* 隨時檢查與判讀。
- 當 *Session* 的生命消失時 *session ID* 的合法使用權也立即消失。








# 一、如何使用 *HttpSession* ?

- *Session* 會封裝在 *HttpServletRequest* 物件中，所以我們可以利用 *HttpServletRequest* 針對 *session* 所提供的方法來加以運用。

方法名稱	說明
<b><i>HttpSession getSession(boolean create)</i></b>	回傳目前封裝在 <b><i>request</i></b> 的 <b><i>session</i></b> ，參數 <b><i>create</i></b> 表示是假如 <b><i>session</i></b> 不存在的話是否要建立新的 <b><i>session</i></b> ？
<b><i>HttpSession getSession()</i></b>	與呼叫 <b><i>getSession(true)</i></b> 同





# 一、如何使用 *HttpSession* ?

- 取得 *session* (連線階段)

- `request.getSession(true);`

- `true` 表示續用 *session*，若無則創建

- `request.getSession();`  
相同效果。

- `false` 表示續用*session*，若無則「不」創建






# 一、如何使用 *HttpSession* ?

- *setAttribute()*、*getAttribute()*
  - 新增、設定以及擷取儲存在 *session* 中的資料。

方法名稱	說明
<b><i>void setAttribute(String name, Object value)</i></b>	將參數 <b><i>name</i></b> 及其內容 <b><i>value</i></b> 新增或修改至 <b><i>session</i></b> 物件中。
<b><i>Object getAttribute(String name)</i></b>	於 <b><i>session</i></b> 中將參數 <b><i>name</i></b> 的內容值取出。



# 一、如何使用 *HttpSession* ?

## • 範例：

```
// 新增 session attribute  
String tech = "Java";  
String[] examInfo = {"310-090", "SCWCD", "1.4"};  
session.setAttribute("Tech", tech);  
session.setAttribute("ExamInfo", examInfo);  
// 擷取 session attribute  
String tech = (String)session.getAttribute("Tech");  
String[] examInfo =  
(String[])session.getAttribute("ExamInfo");
```

要做適當的轉型

# 一、如何使用 *HttpSession* ?

## • 範例：

```
// 新增 session attribute  
String tech = "Java";  
String[] examInfo = {"310-090", "SCWCD", "1.4"};  
session.setAttribute("Tech", tech);  
session.setAttribute("ExamInfo", examInfo);  
// 擷取 session attribute  
String tech = (String)session.getAttribute("Tech");  
String[] examInfo =  
(String[])session.getAttribute("ExamInfo");
```

要做適當的轉型



# 一、如何使用 *HttpSession* ?

- 讓 *session* 失效的 2 種方法：

- 直接刪除

- *void invalidate();*

- *Session* 逾時 (*timeout*) :

- *void setMaxInactiveInterval(30 ← 秒);*

- *int getMaxInactiveInterval();*

- *<session-config>*

- <session-timeout>*

- 30 ← 分鐘*

- <session-timeout>*

- </session-config>*





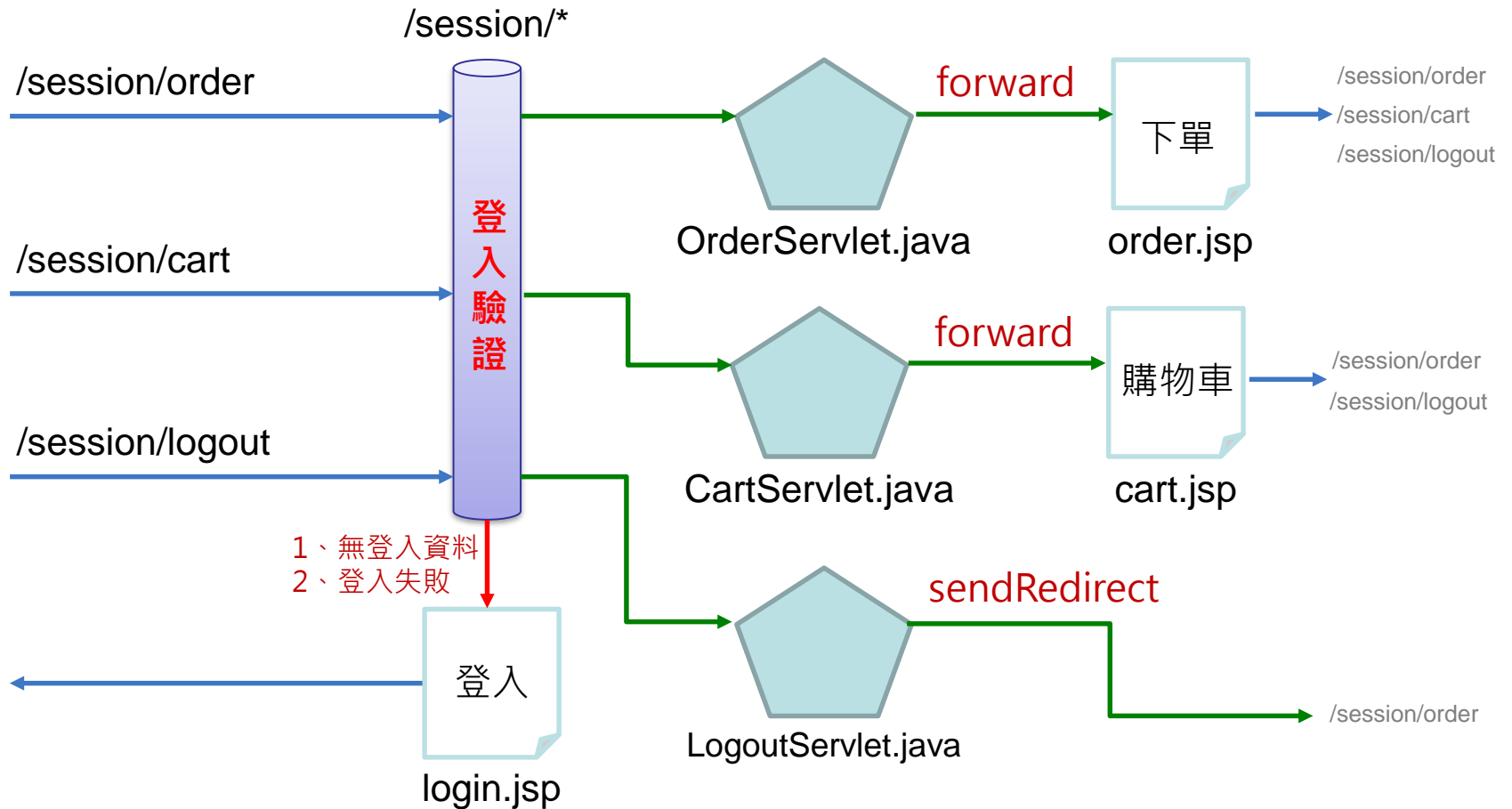
# 一、如何使用 *HttpSession* ?

- *<session-timeout>* v.s *setMaxInactiveInterval()*
  - *<session-timeout>* 的時間單位是分鐘
  - *setMaxInactiveInterval()* 的時間單位是秒
  - *<session-timeout>* 可以利用 0 或任何負整數(例：-1)來表示 *session* 將永遠有效 (*never expire*) 。
  - *setMaxInactiveInterval(-1)* 則也可以帶入負整數方式來表示 *session* 將永遠有效 (*never expire*)，不過不可以放 " 0 " 。



# Session 登入驗證流程

LoginFilter.java







# 綱 要

- 一、HttpSession 運作原理
- 二、如何使用 HttpSession ?
- 三、HttpSession 監聽器與事件
- 四、管理 Session Tracking 的技術





## 三、*HttpSession* 監聽器與事件

- *Servlet API* 針對 *Session* 提供了 5 個傾聽 (*listeners*) 與 2 個事件 (*events*) :
  - *HttpSessionAttributeListener*
    - *HttpSessionBindingEvent*
  - *HttpSessionBindingListener*
    - *HttpSessionBindingEvent*
  - *HttpSessionListener*
    - *HttpSessionEvent*
  - *HttpSessionActivationListener*
    - *HttpSessionEvent*
  - *HttpSessionIdListener*
    - *HttpSessionEvent*





## 三、HttpSession 監聽器與事件

- 5 個 *Listeners* :

1. *HttpSessionAttributeListener*
  2. *HttpSessionBindingListener*
  3. *HttpSessionListener*
  4. *HttpSessionActivationListener*
  5. *HttpSessionIdListener*
- 皆繼承 *java.util.EventListener*

- 2 個 *events* :

1. *HttpSessionBindingEvent*  
繼承 *javax.servlet.http.HttpSessionEvent*
2. *HttpSessionEvent*  
繼承 *java.util.EventObject*





## 三、*HttpSessionListener*

- 當 *session* 被建立或被銷毀時可利用 *HttpSessionListener* 來接收通知，例如：要知道目前的 *active* 的連線數量。
- *HttpSessionListener* 介面提供 2 個要實作的方法分別是：
  - *void sessionCreated();*
  - *void sessionDestroyed();*





## 三、HttpSessionListener

- HttpSessionListener 2 個方法：

方法名稱	說明
<b>void sessionCreated(HttpSession Event se)</b>	當 <b>session</b> 建立時會呼叫此方法。
<b>void sessionDestroyed(HttpSessi onEvent se)</b>	當 <b>session</b> 銷毀/逾時會呼叫此方法。



## 三、HttpSessionListener

- HttpSessionListener 範例：

```
public class SessionCounter
    implements HttpSessionListener {
    public void sessionCreated (HttpSessionEvent evt) {
        setActiveSessions(1);
    }
    public void sessionDestroyed (HttpSessionEvent evt) {
        setActiveSessions(-1);
    }
    // block of code ...
}
```





## 三、HttpSessionListener

- **web.xml 部署：**

```
<listener>
  <listener-class>
    chapter9.SessionCounter
  </listener-class>
</listener>
```

- **@WebListener**

```
@WebListener()
public class MyListener implements HttpSessionListener {
  ...
  ...
}
```





## 三、HttpSessionAttributeListener

- 監聽 `session.setAttribute()`

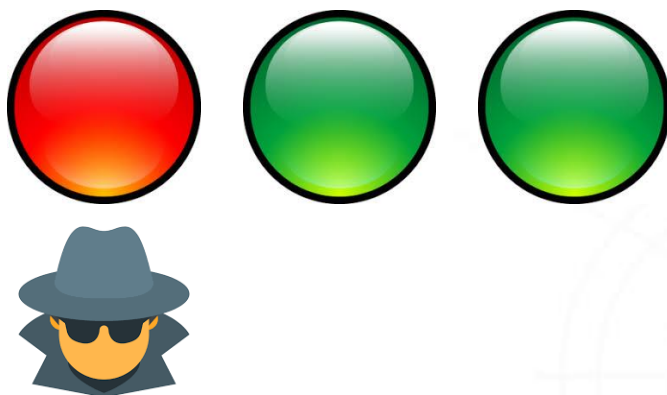
方法名稱	說明
<b><code>void attributeAdded(HttpSession BindingEvent event)</code></b>	當 session 屬性資料被加入
<b><code>void attributeRemoved(HttpSessi onBindingEvent event)</code></b>	當 session 屬性資料被移除
<b><code>void attributeReplaced(HttpSessi onBindingEvent event)</code></b>	當 session 屬性資料被修改





### 三、*HttpSessionBindingListener*

- 透過 *HttpSessionBindingListener* 來實現監控某特定 session 資料物件是否繫結與斷開？



- 不需部署設定
  - 不用 *web.xml* 與 *@WebListener*





## 三、HttpSessionBindingListener

- 方法：

方法名稱	說明
<b>void valueBound(HttpSessionBindingEvent event)</b>	通知該物件已經被 <b>session</b> 繫結。
<b>void valueUnbound(HttpSessionBindingEvent event)</b>	通知該物件已經被 <b>session</b> 斷開繫結。

當 **Binding** 事件發生時，**container** 會自行去呼叫相對應的方法。






## 三、HttpSessionBindingListener

- 範例：

```
public class Login implements HttpSessionBindingListener {  
  
    // 商業邏輯實作...  
  
    public void valueBound(HttpSessionBindingEvent e) {  
        // block of code  
    }  
    public void valueUnbound(HttpSessionBindingEvent e) {  
        // block of code  
    }  
}
```





### 三、*HttpSessionActivationListener*

- 在分散式環境下若要將 *session* 遷移到其他 *JVMs* 時，可以實作

*HttpSessionActivationListener*來監控/  
擷取 *session* 在轉移過程中的事件。

- *HttpSessionActivationListener* 實作時要在 *web.xml* 中設定 / 部署。





## 三、HttpSessionActivationListener

### • 方法

方法名稱	說明
<b><i>void sessionDidActivate(HttpSessionEvent se)</i></b>	當 <b><i>session</i></b> 轉移完成後會呼叫此方法。
<b><i>void sessionWillPassivate(HttpSessionEvent se)</i></b>	當 <b><i>session</i></b> 從 <b><i>A-JVMs</i></b> 轉移到 <b><i>B-JVMs</i></b> 時會呼叫此方法。





## 三、HttpSessionIdListener

- HttpSessionIdListener
  - sessionIdChanged(HttpSessionEvent event, String oldSessionId)
    - 需要實作，實作類別可以在 HttpSession 的 Session ID 發生變化時，會呼叫 sessionIdChanged() 方法。
  - Servlet 3.1 新增





# 綱 要

- 一、HttpSession 運作原理
- 二、如何使用 HttpSession ?
- 三、HttpSession 監聽器與事件
- 四、**管理 Session Tracking 的技術**





## 四、管理 *Session Tracking* 的技術

- *Session Tracking* 指的是讓 *Server* 端能夠追蹤 *Client* 的狀態，在技術上要能夠管理 *Session Tracking* 可藉由下列幾種方式：
  - 利用持續性的 *cookies* 。
    - 這種方式鮮少人使用因為 *cookies* 並不安全。
  - 利用隱藏的表單欄位。
    - 這是最早被用來模擬 *Session Tracking* 的技術，不過在開發與維護上都很麻煩而缺乏安全性。
  - 使用 *HttpSession API* 。
    - 這是最容易達成也是效率最好的技術，不過 *client* 端必須要支援 *cookie* 。
  - *URL Rewriting* (重寫包含額外參數的 *URL*)
    - 這是 *Servlet* 為了讓那些不支援 *cookie* 的瀏覽器也能享有 *session tracking* 服務所發展出來的技術，不過 *URL Rewriting* 會讓系統增加許多額外的負荷。







## 四、管理 Session Tracking 的技術

- Session 與 cookies :

- 當 Server 產生 SessionID 時，接下來會將她傳給 client 端，client 端接到後就會儲存在 cookie 中，以便日後提取之用。

- Client 端與 Server 端如何傳遞 SessionID ??

- 當建立 session 完成時，server 回應給 client 端的 HTTP headers 上就會增加一項特殊列：

`cookie=JSESSIONID=4823565E3AA9A1F31D9DF5AFF2003BAF`

這一行 cookie 就是 server 藉由 HTTP headers 傳遞給 client 端的 sessionID，並由 client 接到後寫入 cookie。

- 接下來只要 client 與 server 端是利用 HTTP 協定溝通，一定都會包含這一行 HTTP headers 的特殊列 (cookie=XXX)。



## 四、管理 *Session Tracking* 的技術

- 有 *cookie* 欄位的 *HTTP Headers* :

```
accept = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,  
application/x-shockwave-flash, application/vnd.ms-excel,  
application/vnd.ms-powerpoint, application/msword, */*  
referer = http://localhost:8080/SCWCD/  
accept-language = zh-tw  
accept-encoding = gzip, deflate  
user-agent = Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;  
(R1 1.3))  
host = localhost:8080  
connection = Keep-Alive  
cookie = JSESSIONID=227FB44B5CD604C26169276D56B5EB08
```

*SessionID*



## 四、管理 *Session Tracking* 的技術

- *Session* 與 *URL Rewriting* :
- 在 *client* 不支援 *cookie* 的情況下我們除了可以使用隱藏的表單欄位的技術之外，另一個就是利用 *URL Rewriting* 的技術，讓繁雜的 *SessionID* 維護工作丟給 *servlet container* 去處理。
- 在實作 *URL Rewriting* 時實際上與一般利用 *HttpSession API* 沒甚麼兩樣，差別僅在於在包裝 *URL* 時要使用 *encodeURL()* 方法來加以編碼，為的是要讓新的 *URL* 中包含 *session ID*。





## 四、管理 *Session Tracking* 的技術

- *HttpServletResponse* 提供的 *URL* 編碼方法

方法名稱	說明
<b><i>String encodeURL(String url)</i></b>	將 <b><i>URL</i></b> 加以編碼。
<b><i>String encodeRedirectURL(String url)</i></b>	欲重導 <b><i>URL</i></b> 並加以編碼，在功能上相當於 <b><i>sendRedirect()</i></b> 。

- 在運作上這 2 個方法在第一時間會去偵測 *client* 端有無支援 *cookie*，來決定是否要將 *session ID* 與 *URL* 一同編碼。

## 四、管理 Session Tracking 的技術

- URL Rewriting 範例：

```
...  
HttpSession session = req.getSession(true);  
String tech = "Java";  
session.setAttribute("Tech", tech);  
...  
pw.println("[<a href=\"  
res.encodeURL("/SCWCD/servlet/GetSession") +  
\">GetSession</a>]");  
...
```

使用 "/" 開頭時要包含 **Context Path**

## 四、管理 Session Tracking 的技術

- *Client* 端支援 cookies 時：

```
<html></body>  
[<a href="/SCWCD/servlet/GetSession">GetSession</a>]  
</body></html>
```

*Client* 端不支援 cookies 時：

```
<html></body>  
[<a  
href="/SCWCD/servlet/GetSession;sessionId=5FAFF05008  
6BD26930E7852746F01309">GetSession</a>]  
</body></html>
```

用「;」隔開

