



巨匠線上真人

Java Web OCE JWCD元件系統 開發認證

www.pcschoolonline.com.tw



巨匠線上真人

Java Web OCE JWCD元件系統開發認證

第十一堂

網站應用系統狀態管理

本堂教學重點

- ◆ Page Level狀態機制與應用
- ◆ Request狀態管理與應用
- ◆ Session狀態管理與應用-登入驗證作業
- ◆ Session狀態稽核流程應用

Page Level狀態機制與應用

- ◆ JSP一般在存取上可以使用Scriptlet設定區域變數即可。但區域變數無法被EL直接存取輸出。
- ◆ 可以使用內建物件pageContext.setAttribute Method 進行Page層級的Attribute參考。形成在同一個Page中可以透過EL輸出。

pagecontextdemo.jsp

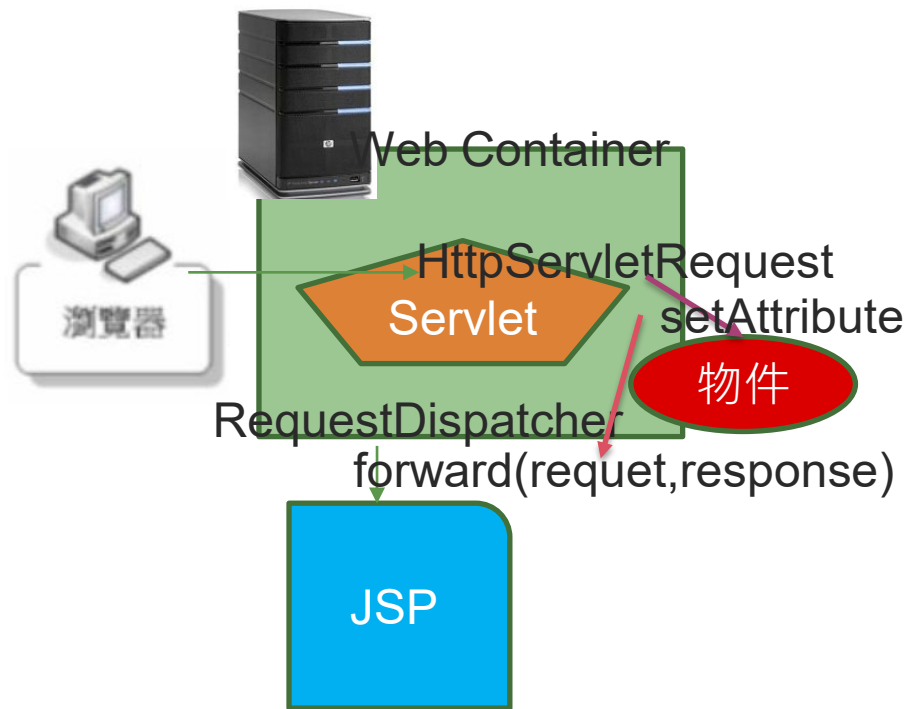
```
<body>
<%
    String com="巨匠電腦";
    //使用pageContext內建物件進行Attribute參考
    pageContext.setAttribute("company", com);
%>
<!-- 使用EL運算輸出 無法運算區域變數 -->
<div>公司行號:${com}</div>
<div>公司行號:${company}</div>
</body>
```

0http://localhost:8080/mod11/pagecontextdemo.jsp

公司行號:
公司行號:巨匠電腦

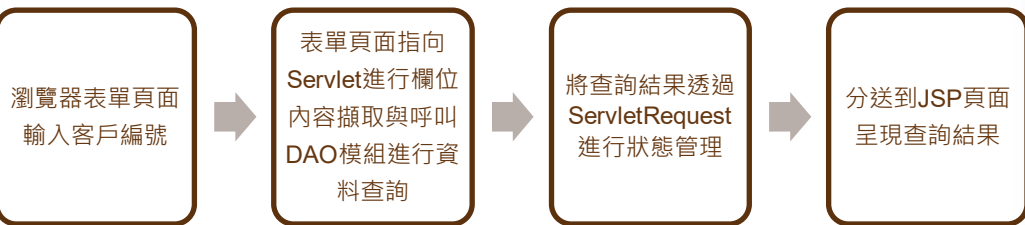
Request狀態管理與應用

- ◆ ServletRequest狀態管理，往往應用在類似MVC架構上的設計，因為Servlet Container並沒有預設如MVC的ValueStack狀態管理環境，所以在進行分派View時，無法使用參數傳遞架構來持續處理的結果或狀態。
- ◆ 這時候我們需要RequestDispatcher進行分送View時，且需要透過ServletRequest執行setAttribute Method進行狀態持續作業。
- ◆ 狀態管理生命週期可以限制在Response回應之後，自動釋放。



Model 2 查詢客戶資料Request狀態管理應用

- ◆ 進行MySQL sakila資料庫Customer資料查詢應用
- ◆ 採用Model2架構(MVC)依照輸入的客戶編號進行查詢



```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>客戶資料查詢</title>
</head>
<body>
<form method="post" action="CustomersQry">
  <div>客戶編號</div>
  <input type="text" name="customerid" value="查詢"/>
</form>
</body>
</html>
```

customersqry.html

規劃 DAO Model

- ◆ 規劃JavaBean類別配合介面泛型規劃，動態配合查詢方法回應的型別。
- ◆ 設計客戶資料查詢類別，實作IDAO介面查詢方法。
- ◆ 採用Property Injection注入DataSource物件

IDao.java

```
package com.gjun.domain;

import java.sql.SQLException;
//使用Generic設計，可以配合類別實作時設定相對的Entity class
public interface IDao<T> {
    public void setDataSource(DataSource datasource);
    //查詢單筆方法
    public T selectForObject(String sql,Object...args) throws SQLException;
}
```

Customer.java

```
package com.gjun.domain;
//Java Bean
public class Customer implements java.io.Serializable {
    //封裝欄位
    private short customerId;
    private String firstName;
    private String lastName;
    private String email;
    public short getCustomerId() {
        return customerId;
    }
    public void setCustomerId(short customerId) {
        this.customerId = customerId;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getEmail() {
        return email;
    }
}
```

客戶查詢DAO類別實作

- ◆ 介面Idao方法實作，並且指定泛型類別Customer Javabaen。

```
public class CustomerDao implements IDao<Customer>{  
    //Attribute  
    private DataSource datasource;  
  
    //注入DataSource物件(連接工廠)  
    @Override  
    public void setDataSource(DataSource datasource) {  
        this.datasource=datasource;  
    }  
}
```

```
public Customer selectForObject(String sql, Object... args) throws SQLException {  
    Customer customer=null;  
    //判斷DataSource  
    if(datasource==null) {  
        throw new SQLException("資料來源物件尚未注入!!!");  
    }else  
    {  
        //透過DataSource物件生產一個連接物件(及時連接資料庫)  
        Connection connection=datasource.getConnection();  
        //配合傳遞進來的SQL語法產生PreparedStatement物件  
        PreparedStatement pre=connection.prepareStatement(sql);  
        //掃描參數 設定參數內容  
        for(int index=0;index<args.length;index++) {  
            pre.setObject(index+1, args[index]);  
        }  
        //執行查詢產生ResultSet  
        ResultSet rs=pre.executeQuery();  
        //判斷是否有找到相對客戶資料  
        if(rs.next()) {  
            //建構Customer Javabean物件  
            customer=new Customer();  
            customer.setCustomerId(Short.parseShort(rs.getObject("Customer_id").toString()));  
            customer.setFirstName(rs.getObject("First_Name").toString());  
            customer.setLastName(rs.getObject("Last_Name").toString());  
            customer.setEmail(rs.getObject("Email").toString());  
        }  
        connection.close();  
        return customer;  
    }  
}
```


Controller-Servlet設計

- ◆ 用來界接前端View，擷取傳遞進來的表單欄位(查詢鍵值)。
- ◆ 用來建構執行階段產生資料庫的連接物件的DataSource物件(連接工廠)。
- ◆ 用來建構自訂的DAO物件進行資料庫記錄查詢。
- ◆ 控制查詢結果，且透過
 - ◇ RequestServlet.setAttribute()方法進行執行階段的查詢結果狀態管理。
- ◆ 透過RequestDispatcher分送不同的View。

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //擷取客戶編號
    String customerId=request.getParameter("customerId");
    //建構DataSource物件
    BasicDataSource dataSource=new BasicDataSource();
    dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
    dataSource.setUrl("jdbc:mysql://localhost:3306/sakila?useSSL=false"
        + "&serverTimezone=UTC&characterEncoding=utf-8");
    dataSource.setUsername("root");
    dataSource.setPassword("1111");
    //呼叫DAO模組進行資料查詢
    IDao<Customer> dao=new CustomerDao();
    dao.setDataSource(dataSource); //注入DataSource(Dependency Injection)
    //呼叫查詢方法
    Customer customer;
    try {
        customer = dao.selectForObject("select * from customer where customer_id=?",customerId);
    }
    //判斷是否查詢有記錄
    if(customer==null) {
        String msg=String.format("您查詢的客戶編號:%s 記錄不存在!!",customerId);
        //進入request狀態管理
        request.setAttribute("msg",msg);
        //派送到View
        RequestDispatcher disp=request.getRequestDispatcher("notfound.jsp");
        disp.forward(request, response);
    }
}
```

查詢結果狀態管理與派送View

```
//透過 ServletRequest進行查詢結果狀態管理
```

```
request.setAttribute("customer",customer);
```

```
//派送到View
```

```
RequestDispatcher disp=request.getRequestDispatcher("found.jsp");
```

```
disp.forward(request, response);
```

http://localhost:8080/mod11/customerqry.html

客戶編號

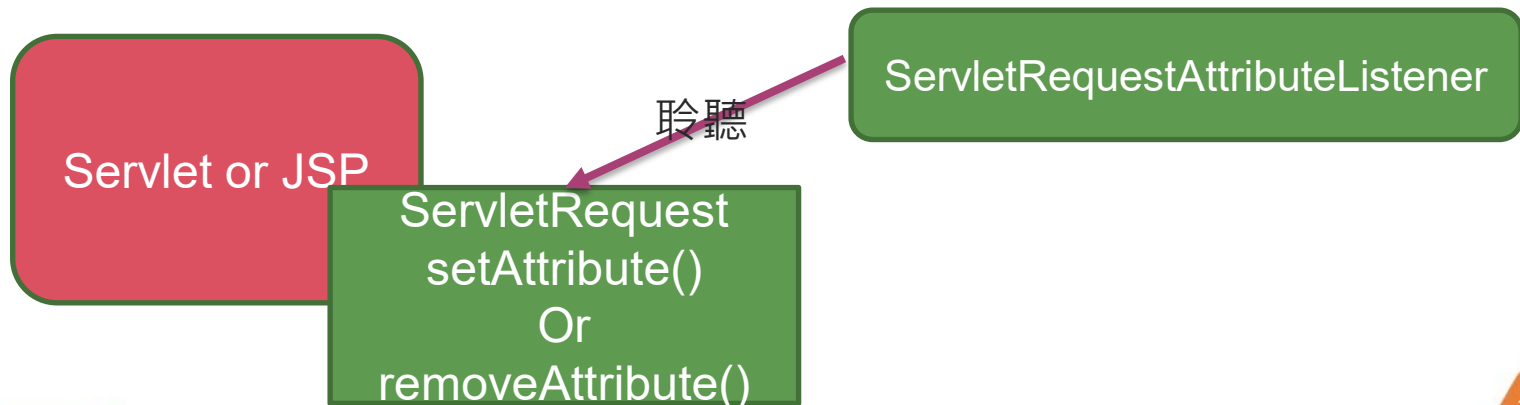


http://localhost:8080/mod11/CustomersQry

客戶編號:1
First Name:MARY
Last Name:SMITH
EMAIL:MARY.SMITH@sakilacustomer.org

Request參照物件聆聽稽核流程

- ◆ 網站系統運作ServletRequest 進行Attribute參照或者移除或者修改作業，可以安置一個Listener進行稽核或者日誌作業。
- ◆ 如前面客戶查詢作業，可以特別針對使用者查詢的客戶結果進行稽核作業。
- ◆ 實作ServletRequestAttributeListener介面，並且進行佈署。



設計類別實作ServletRequestAttributeListener

◆ ServletRequestAttributeListener 具有三個方法(事件)。

- ◆ attributeAdded
- ◆ attributeRemoved
- ◆ attributeReplaced

```
//佈署Listener
@WebListener()
public class RequestAttributeHandler implements ServletRequestAttributeListener {

    //聆聽新增Attribute
    @Override
    public void attributeAdded(ServletRequestAttributeEvent srae) {
        //開啟Log File
        try {
            File file=new File("c:/data/log.txt");
            if(!file.exists())
            {
                return;
            }
            OutputStream os=new FileOutputStream(file,true);
            OutputStreamWriter writer=new OutputStreamWriter(os,"UTF-8");

            //寫出資訊
            writer.write(String.format("request管理的物件:%s\n",
                srae.getValue().toString()));

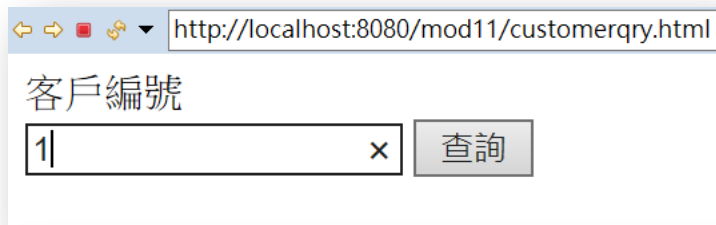
            writer.flush();
            writer.close();
        } catch (FileNotFoundException e) {

        } catch (UnsupportedEncodingException e) {

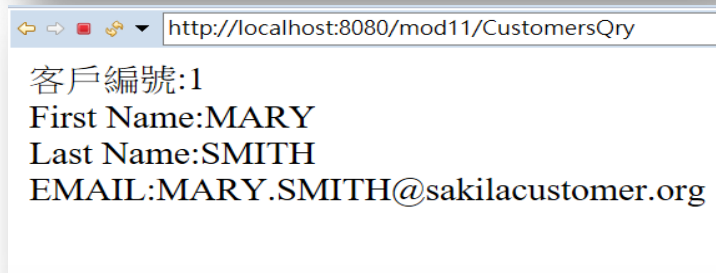
        }
    }
}
```

實作結果

- ◆ 查詢客戶資料之後，產生一筆稽核記錄。

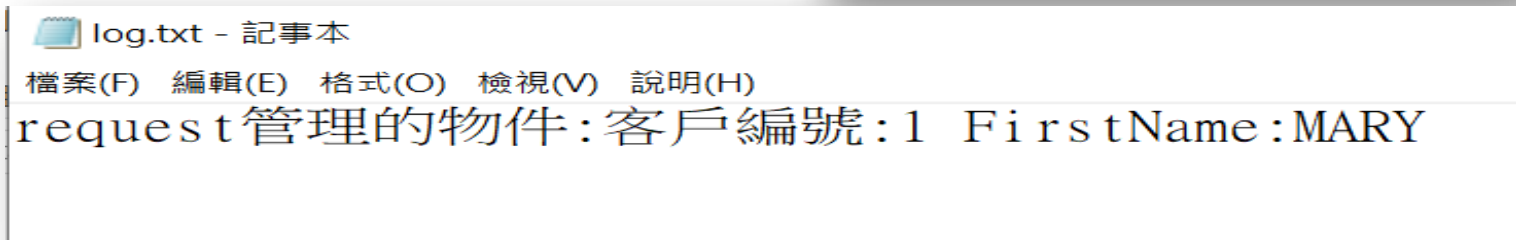


A screenshot of a web browser window with the address bar showing `http://localhost:8080/mod11/customerqry.html`. The page contains a form with the label "客戶編號" (Customer Number) above a text input field. The input field contains the number "1" and has a close button (x) on its right. To the right of the input field is a button labeled "查詢" (Query).



A screenshot of a web browser window with the address bar showing `http://localhost:8080/mod11/CustomersQry`. The page displays the following information:

- 客戶編號:1
- First Name:MARY
- Last Name:SMITH
- EMAIL:MARY.SMITH@sakilacustomer.org

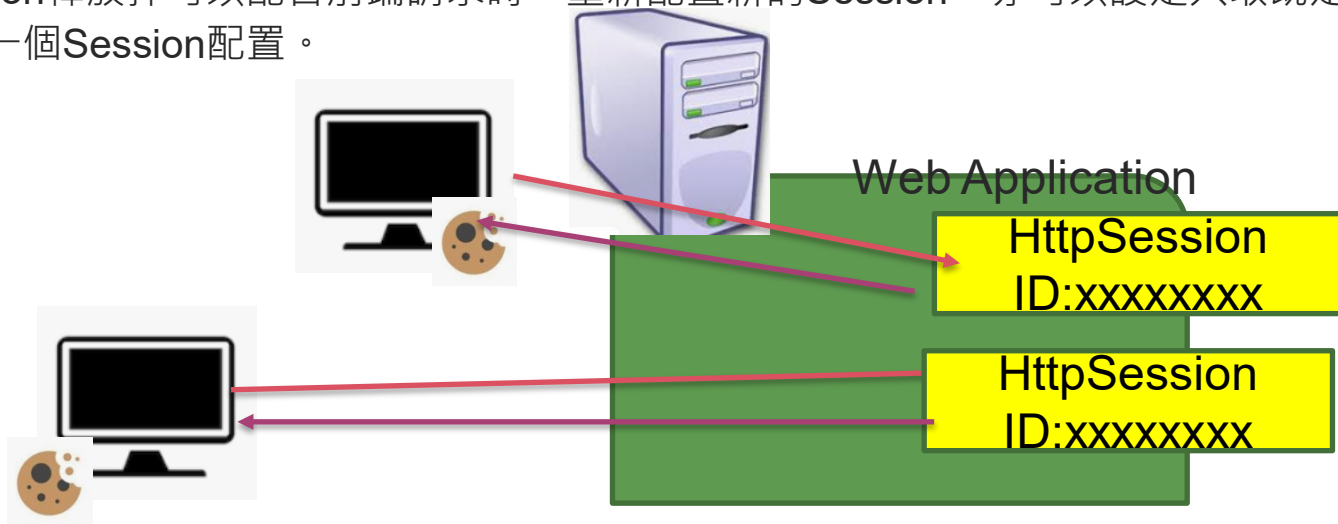


A screenshot of a Notepad window titled "log.txt - 記事本". The menu bar includes "檔案(F)", "編輯(E)", "格式(O)", "檢視(V)", and "說明(H)". The text content of the window is:

```
request管理的物件:客戶編號:1 First Name:MARY
```

Session狀態管理與應用-登入驗證作業

- ◆ Session狀態管理一般存活於一個網站系統中。並且配合前端進行個別Session(交談)管理。
- ◆ 適用在網站較長週期的物件管理。
- ◆ 可用在購物車多數量的物件或者登入之後的Server Side憑證管理的作業。
- ◆ 具有Timeout預設生命週期。
- ◆ Session釋放掉可以配合前端請求時，重新配置新的Session，亦可以設定只取既定Session，不產生一個Session配置。

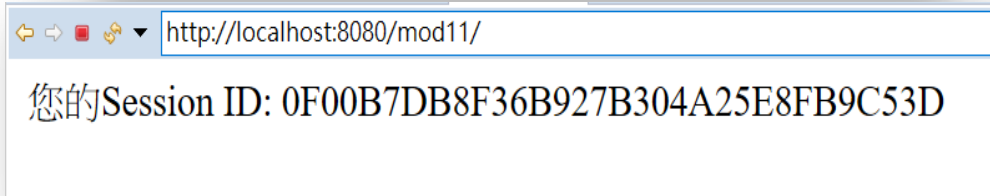


Session運作原理

index.jsp

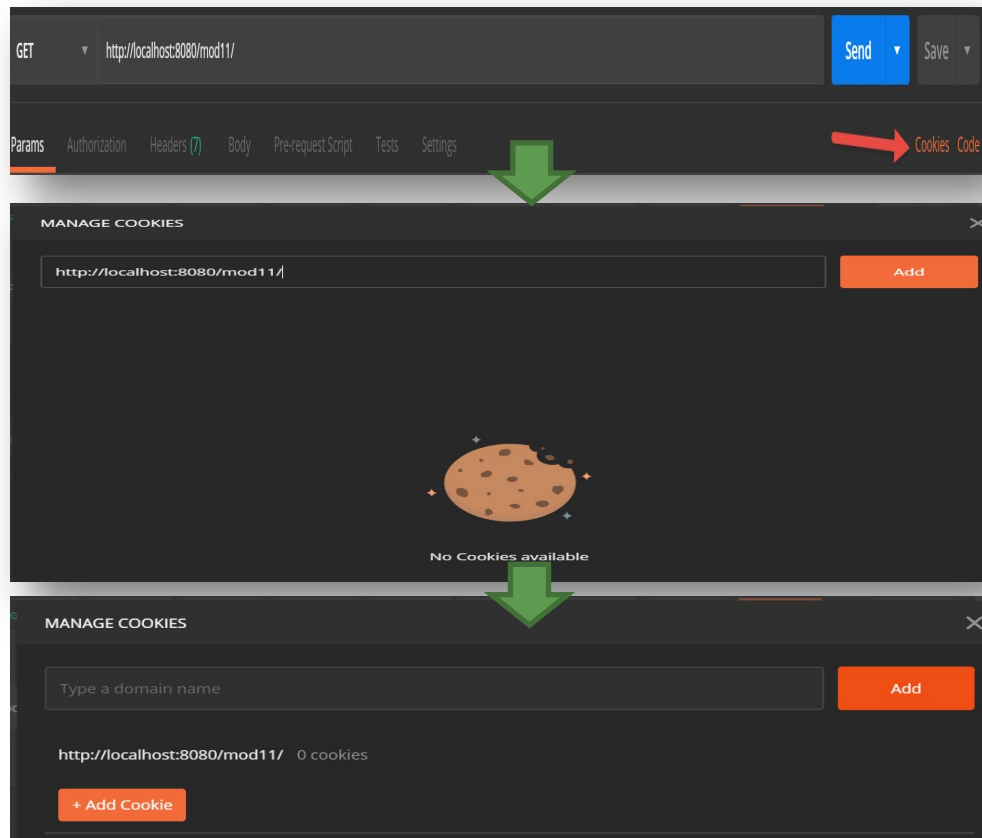
- ◆ 網站系統接受到前端進入網站時，則會配置一個Session物件，並且發出Session ID，前端則儲存在Cookie中(前提是前端需要有Cookie Container-如瀏覽器等裝置)。
- ◆ 一般發出Session ID會封裝在Response的Http Header中。

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Session</title>
</head>
<body>
<div>您的Session ID:
<%=session.getId()%>
</div>
</body>
</html>
```



透過Postman工具檢查Header

- ◆ Postman工具預設不會啟動Cookie Container。
- ◆ 需要加入請求的Domain Name。
- ◆ 提出網站第一次請求之後，後端Session會將Session ID封裝到前端的Cookie中，且設定Header Name預設為JSESSIONID。



檢視Http Header Cookie

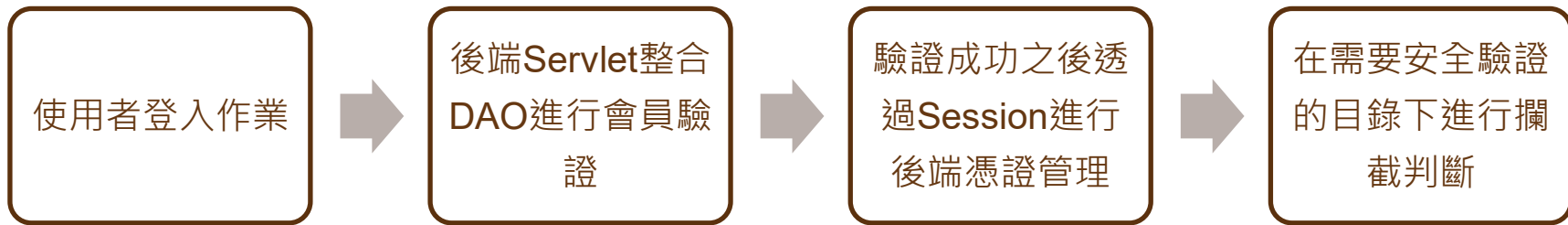
Body Cookies (1) **Headers (4)** Test Results

Status: 200 OK Time: 12ms Size: 378 B

KEY	VALUE
Set-Cookie ⓘ	JSESSIONID=69EC2A7445C016E1BF1E8C08936F3B9D; Path=/mod11; HttpOnly
Content-Type ⓘ	text/html;charset=UTF-8
Content-Length ⓘ	182
Date ⓘ	Mon, 16 Dec 2019 02:39:30 GMT

利用Session持續使用者登入驗證狀態

- ◆ Session可以解決Request與Response的非狀態管理問題(Stateless)。
- ◆ 主要是配合前端的交談形成一個網站系統持續狀態的方式，可以進行在一個網站系統進行持續登入狀態或者購物車等資訊管理。
- ◆ 設計一個登入驗證之後的Server Side憑證管理作業。



使用者登入作業

- ◆ 提供登入表單頁面，設定文字輸入方塊，並且明確表單欄位name Attribute設定，用來進行表單欄位參數傳遞參照。
- ◆ `<form>` action設定指向Servlet urel-pattern端點位址。

login.html

```
<body>
  <fieldset>
    <legend>登入作業</legend>
    <form method="post" action="LoginValidServlet">
      <div>使用者名稱</div>
      <input type="text" name="username"/>
      <div>使用者密碼</div>
      <input type="password" name="password"/>
      <input type="submit" value="登入"/>
    </form>
  </fieldset>
</body>
```

DAO Model設計-1

- ◆ 規劃一個對應資料表欄位的Javabean(Entity)類別。
- ◆ 設計Idao介面查詢方法規範。
- ◆ 設計Member會員資料表查詢的DAO類別，且實作Idao介面。

member table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
username	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Member.java

```
package com.gjun.domain;
//Javabean
public class Member implements java.io.Serializable{
    //Attribute
    private String username;
    private String password;
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

DAO Model設計-2

IDao.java

```
package com.gjun.domain;

import java.sql.SQLException;
//使用Generic設計，可以配合類別實作時設定相對的Entity class
public interface IDao<T>{
    public void setDataSource(DataSource datasource);
    //查詢單筆方法
    public T selectForObject(String sql,Object...args) throws SQLException;
}
```



MemberDao.java

```
@Override
public Member selectForObject(String sql, Object... args) throws SQLException {
    Member member=null;
    //判斷DataSource
    if(datasource==null) {
        throw new SQLException("資料來源物件尚未注入!!!");
    }else
    {
        //透過DataSource物件生產一個連接物件(及時連接資料庫)
        Connection connection=datasource.getConnection();
        //配合傳遞進來的SQL語法產生PreparedStatement物件
        PreparedStatement pre=connection.prepareStatement(sql);
        //掃描參數 設定參數內容
        for(int index=0;index<args.length;index++) {
            pre.setObject(index+1, args[index]);
        }
        //執行查詢產生ResultSet
        ResultSet rs=pre.executeQuery();
        //判斷是否有找到相對客戶資料
        if(rs.next()) {
            //建構Customer Javabean物件
            member=new Member();
            member.setUsername(rs.getString("username"));
            member.setPassword(rs.getString("password"));
        }
        connection.close();
        return member;
    }
}
```

使用者驗證Servlet設計



```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
```

```
    //擷取表單欄位
```

```
    String username=request.getParameter("username");
```

```
    String password=request.getParameter("password");
```

```
    //建構DataSource物件
```

```
    BasicDataSource dataSource=new BasicDataSource();
    dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
    dataSource.setUrl("jdbc:mysql://localhost:3306/sakila?useSSL=false"
        + "&serverTimezone=UTC&characterEncoding=utf-8");
    dataSource.setUsername("root");
    dataSource.setPassword("1111");
```

```
    //建構自訂DAO物件 進行登入會員驗證
```

```
    IDao<Member> dao=new MemberDao();
```

```
    //注入DataSource
```

```
    dao.setDataSource(dataSource);
```

```
try {
```

```
    Member member=dao.selectForObject("select username,password from member where username=? and password=?"
        ,username,password);
```

```
    if(member==null) {
```

```
        //登入失敗
```

```
        String msg=String.format("%s 帳號不存在!登入失敗!!!",username);
```

```
        request.setAttribute("msg", msg);
```

```
        request.getRequestDispatcher("loginfailure.jsp").forward(request, response);
```

```
    }else
```

```
    {
```

```
        //登入成功
```

```
        //Session管理憑證
```

```
        HttpSession session=request.getSession();
```

```
        session.setAttribute("cred",member.getUsername());
```

```
        request.setAttribute("member",member);
```

```
        //派送頁面
```

```
        request.getRequestDispatcher("loginok.jsp").forward(request, response);
```

```
    }
```

```
} catch (SQLException e) {
```

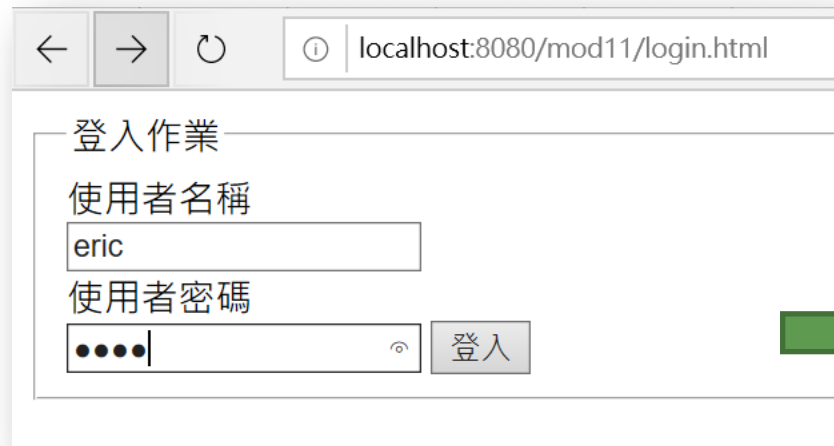
```
    // TODO Auto-generated catch block
```

```
    e.printStackTrace();
```

```
}
```

驗證成功的
Session狀態管理

Session狀態管理範例執行結果



登入作業

使用者名稱

eric

使用者密碼

●●●●

登入



登入成功!! 歡迎你:eric

安全目錄攔截驗證-持續Session作業

- ◆ 設想進入網站特定虛擬目錄下，需要確定合法使用者，才能請求相對虛擬目錄下的資源。
- ◆ 透過登入驗證過後的HttpSession參照的資源進行驗證。
- ◆ 面對目錄下多個資源請求，設計一個Filter篩選機制，進行後端持續的憑證驗證作業。

<http://www.gjun.com.tw/myweb/acct/report1.jsp>

Sub web入口

請求報表資源

必須登入驗證後才能通行的財務虛擬目錄

設計Filter驗證Session持續的憑證

- ◆ 設計類別實作javax.servlet.Filter。
- ◆ 進行佈署Filter攔截的虛擬目錄。

Authorization.java

```
package com.gjun.filter;

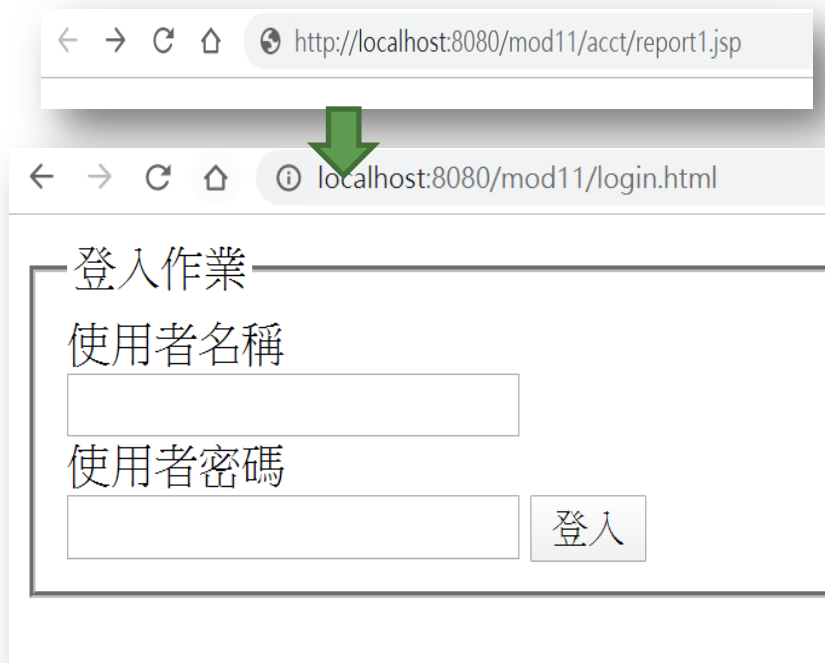
import java.io.IOException;

//佈署Filter與指定urlPattern指向虛擬目錄acct
@WebFilter(urlPatterns= {"/acct/*"})
public class AuthorizationFilter implements Filter {

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest=(HttpServletRequest)request;
        //判斷後端是否有針對該前端持續的Session狀態cred
        if(httpRequest.getSession().getAttribute("cred")==null) {
            //非法進入(尚未登入)
            ((HttpServletRequest)response).sendRedirect("../login.html");
        }
        chain.doFilter(request, response);
    }
}
```

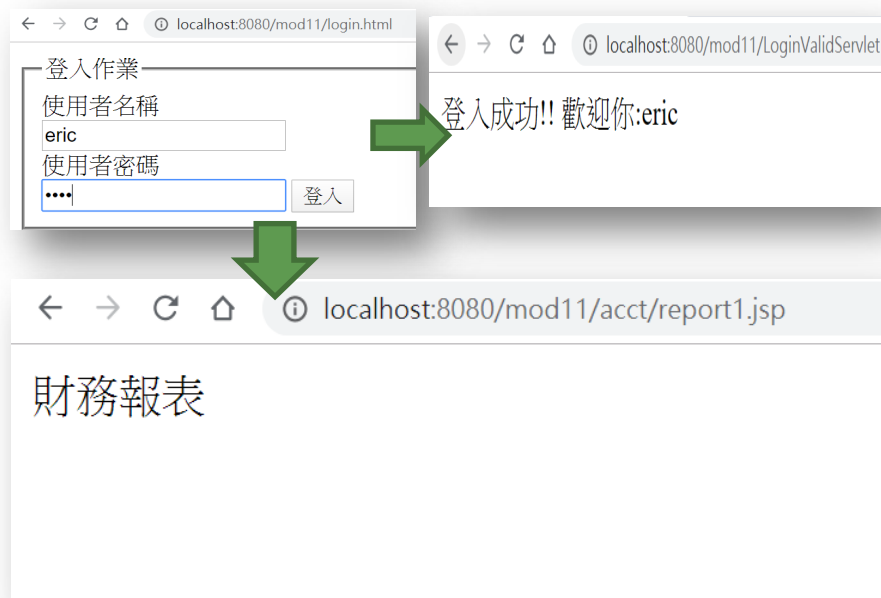
安全目錄下資源存取驗證

- ◆ 尚未登入驗證，直接存取目錄下諮詢



The screenshot shows a web browser with the address bar set to `http://localhost:8080/mod11/acct/report1.jsp`. A green arrow points down to another browser window. This second window has the address bar set to `localhost:8080/mod11/login.html` and displays a login form titled "登入作業". The form contains two input fields: "使用者名稱" (Username) and "使用者密碼" (Password), with a "登入" (Login) button to the right of the password field.

- ◆ 登入驗證成功之後，存取安全目錄下的資源



The screenshot shows a sequence of browser windows. The first window has the address bar set to `localhost:8080/mod11/login.html` and shows the login form with "使用者名稱" (Username) set to "eric" and "使用者密碼" (Password) masked with dots. A green arrow points to the right, where a second window shows the message "登入成功!! 歡迎你:eric". Another green arrow points down to a third window with the address bar set to `localhost:8080/mod11/acct/report1.jsp`, which displays the text "財務報表" (Financial Statement).

Session狀態稽核流程應用

- ◆ Session 聆聽器區分為五種介面
 - ◆ HttpSessionAttributeListener
 - ◆ HttpSessionBindingListener
 - ◆ HttpSessionListener
 - ◆ HttpSessionActivationListener
 - ◆ HttpSessionIdListener
- ◆ 我們採用實作相對的介面，並且進行佈署，即可監聽Session相關觸發點引發的事件，進行事件程序處理。

HttpSessionAttributeListener

- ◆ 聆聽當HttpSession執行setAttribute()與removeAttribute()或者setAttribute針對已經存在的attribute name進行修改時，引發相關的事件程序處理。
- ◆ HttpSessionAttributeListener方法

Modifier and Type	Method and Description
default void	attributeAdded (HttpSessionBindingEvent se) Notification that an attribute has been added to a session.
default void	attributeRemoved (HttpSessionBindingEvent se) Notification that an attribute has been removed from a session.
default void	attributeReplaced (HttpSessionBindingEvent se) Notification that an attribute has been replaced in a session.

聆聽使用者登入驗證成功進行的憑證管理

佈署

- ◆ 當登入驗證成功，透過 `HttpSession.setAttribute()` 進行憑證管理時，引發的事件程序。
- ◆ 進行登出時，移除 `HttpSession` 參考的憑證。

```
//聆聽Session Attribute參照狀態時引發事件
@WebListener()
public class SessionAttributeHandler implements HttpSessionAttributeListener{

    @Override
    public void attributeAdded(HttpSessionBindingEvent se){
        String name=se.getName();
        Object value=se.getValue();
        String msg =String.format("新增Attribute Name:%s 值:%s", name,value);
        //呼叫處理Output method
        try {
            AppUtility.writerSessionAttributeLog("c:/data/session.log",msg);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

AppUtility.java

```
public class AppUtility {
    public static void writerSessionAttributeLog(String fileName,String message) throws Exception {
        try {
            //建立串流 開啟檔案
            OutputStream os=new FileOutputStream(fileName,true);
            //建構Writer
            OutputStreamWriter writer=new OutputStreamWriter(os,"UTF-8");
            writer.write(message+"\n");
            writer.flush();
            writer.close();
        } catch (FileNotFoundException e) {
            throw e;
        } catch (UnsupportedEncodingException e) {
            throw e;
        } catch (IOException e) {
            throw e;
        }
    }
}
```

SessionAttributeHandler.java

Login登入驗證通過

Session Attribute監聽產生Log

◆ 使用者登入驗證成功

localhost:8080/mod11/login.html

登入作業

使用者名稱
eric

使用者密碼
...

登入

localhost:8080/mod11/LoginValidServlet

登入成功!! 歡迎你:eric

session.log - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

新增Attribute Name:cred 值:eric

session.log

◆ 登出時釋放掉相對的 HttpSession 參考的憑證，產生了稽核。

Report1.java

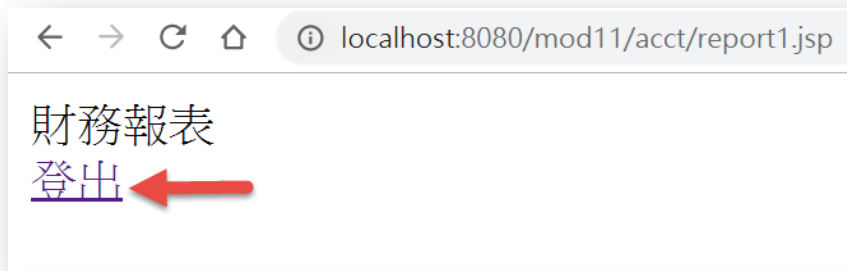
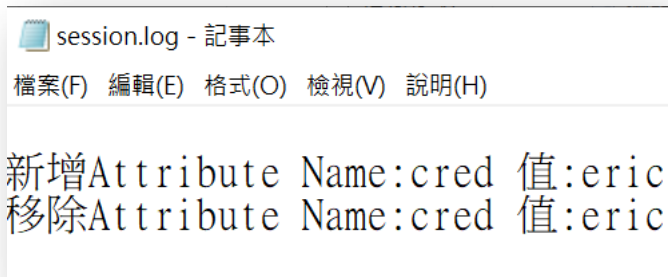
```
<meta charset=  
<title>Insert title here</title>  
</head>  
<body>  
<div>財務報表</div>  
<a href="../LogoutServlet" >登出</a>  
</body>
```

LogoutServlet.java

```
@WebServlet("/LogoutServlet")  
public class LogoutServlet extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
  
    protected void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        //取出HttpSession  
        HttpSession session=request.getSession();  
        if(session.getAttribute("cred")!=null) {  
            //釋放掉參考的憑證  
            session.removeAttribute("cred");  
            //導向輸出  
            response.sendRedirect("login.html");  
        }  
    }  
}
```

Logout產生的稽核

- ◆ SessionAttributeListener.
attributeRemoved Method應用



HttpSessionBindingListener

- ◆ 聆聽一Session綁定或者移除綁定引發的事件。

← → ↻ ⬆ ⓘ localhost:8080/mod11/LoginValidServlet

登入成功!! 歡迎你:eric

sessionbind.log - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

新增Attribute Binding Name:cred 值:eric

SessionBindingHandler.java

```
import com.gjun.domain.AppUtility;
//聆聽Session Binding Attribute參照狀態時引發事件
@WebListener()
public class SessionBindingHandler implements HttpSessionAttributeListener
{
    @Override
    public void attributeAdded(HttpSessionBindingEvent se) {
        String name=se.getName();

        Object value=se.getValue();
        String msg =String.format("新增Attribute Binding Name:%s 值:%s", name,value);
        //呼叫處理Output method
        try {
            AppUtility.writerSessionAttributeLog("c:/data/sessionbind.log",msg);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```


HttpSessionListener 聆聽前端配置Session狀態

- ◆ Session具有生命週期，採用Slide Time方式配置。
- ◆ 借助web.xml進行配置timeout時間。
- ◆ 或者透過程式化進行強制釋放。
- ◆ 可以借助HttpSessionListener建立聆聽配置前端的 HttpSession生命週期狀態。

Session timeout寫出Log

session.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

Session ID:CC021C4442CFD2481912C454BFEE4EFF 建立起來!!

Session ID:CC021C4442CFD2481912C454BFEE4EFF 被釋放!!

SessionHandler.java

```
@WebListener()  
public class SessionHandler implements HttpSessionListener{  
  
    @Override  
    public void sessionCreated(HttpSessionEvent se) {  
        String sessionId=se.getSession().getId();  
        String msg=String.format("Session ID:%s 建立起來!!\n",sessionId);  
        try {  
            AppUtility.writerSessionAttributeLog("c:/data/session.txt",msg);  
        } catch (Exception e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
    }  
  
    @Override  
    public void sessionDestroyed(HttpSessionEvent se) {  
        String sessionId=se.getSession().getId();  
        String msg=String.format("Session ID:%s 被釋放!!\n",sessionId);  
        try {  
            AppUtility.writerSessionAttributeLog("c:/data/session.txt",msg);  
        } catch (Exception e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
    }  
}
```

web.xml

```
<session-config>  
    <session-timeout>1</session-timeout>  
</session-config>
```

HttpSessionActivationListener

- ◆ 絕大部份的情況下，幾乎不會使用到 HttpSessionActivationListener。
- ◆ 使用到分散式環境，應用程式的需要被參考的物件，分散在多個 JVM 之中。當 HttpSession 要從一個 JVM 遷移至另一個 JVM 時，必須先在原本的 JVM 上序列化 (Serialize) 參照的物件。
- ◆ 實作 HttpSessionActivationListener，就會呼叫 sessionWillPassivate() 方法，而 HttpSession 遷移至另一個 JVM 後，就會對所有屬性物件作反序列化，此時會呼叫 sessionDidActivate() 方法。
 - ◇ public void sessionDidActivate(HttpSessionEvent se)
 - ◇ public void sessionWillPassivate(HttpSessionEvent se)

HttpSessionIdListener 應用

- ◆ Servlet 3.1 新增特性
- ◆ 主要聆聽Session Id被修改時引發處理事件。

```
package com.gjun.filter;
import java.io.IOException;

//佈署Filter與指定urlPattern指向虛擬目錄acct
@WebFilter(urlPatterns= {"/acct/*"})
public class AuthorizationFilter implements Filter {

    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest=(HttpServletRequest)request;
        //判斷後端是否有針對該前端持續的Session狀態cred
        if(httpRequest.getSession().getAttribute("cred")==null) {
            //非法進入(尚未登入)
            ((HttpServletResponse)response).sendRedirect("../login.html");
        }
        chain.doFilter(request, response);
    }
}
```

Lab

- ◆ 如何在一個JSP進行Page Level範圍狀態管理與應用
- ◆ 如何使用Request狀態管理設計Model2架構的派送狀態與應用
- ◆ 實現一個客戶資料查詢與DAO設計模組整合應用
- ◆ 如何設計各階層狀態產生與管理的Listener，設計相關的Log稽核