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

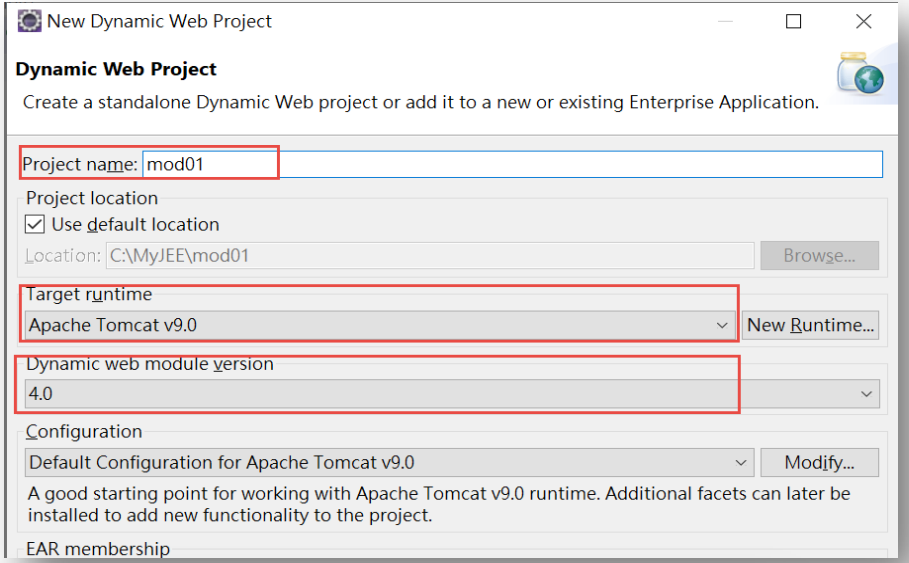
第三堂 邁入Java 網站系統-Servlet請求與回應

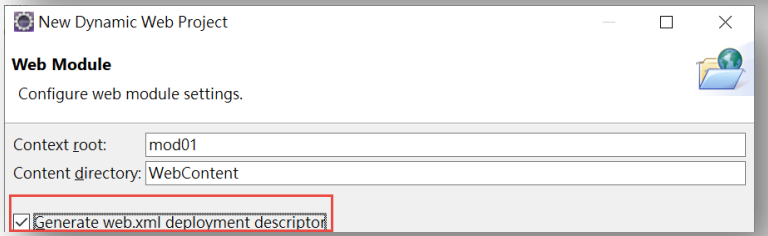
本堂教學重點

- 建構一個Java網站系統
- 我的網站第一個Servlet
- 與前端互動的ServletRequest/ServletResponse介面應用
- HTTP Request Method如何配合Servlet Method應用架構
- Servlet生命週期

建構一個Java網站系統

- 使用Eclipse新增一個Java Web專案
  - 功能表File/New/Dynamic Web Project
  - 填寫專案名稱(Project Name)
  - 設定Target Runtime 指定安裝好的Tomcat 9
  - Dynamic Web Modules Version版本4.0
  - 設定產生web.xml，網站佈署檔案。

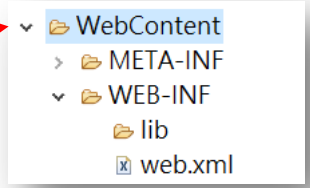


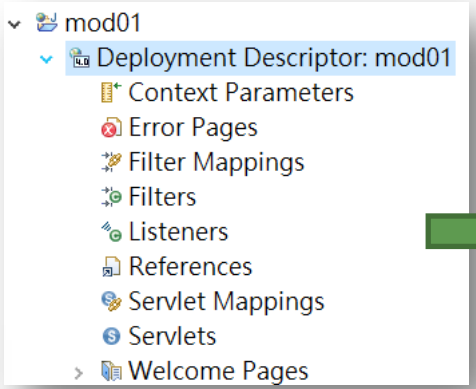


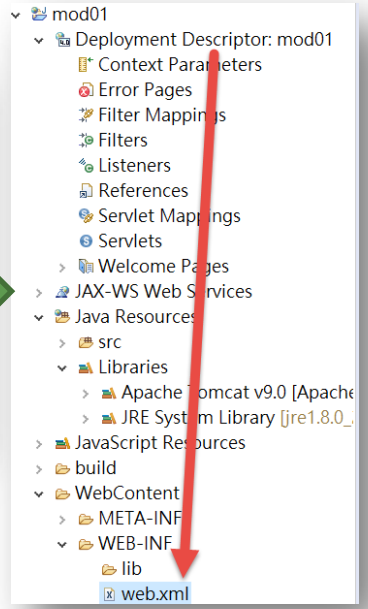
最後一個步驟
設定產生web.xml

Eclipse Java Web專案架構說明

- 網站根目錄
- 網站參考Java Resource API
- 佈署與描述檔案定義項目管理
- 自行撰寫的Java Source

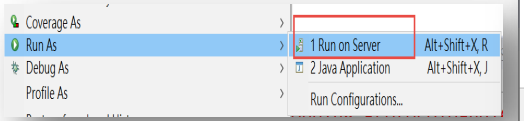


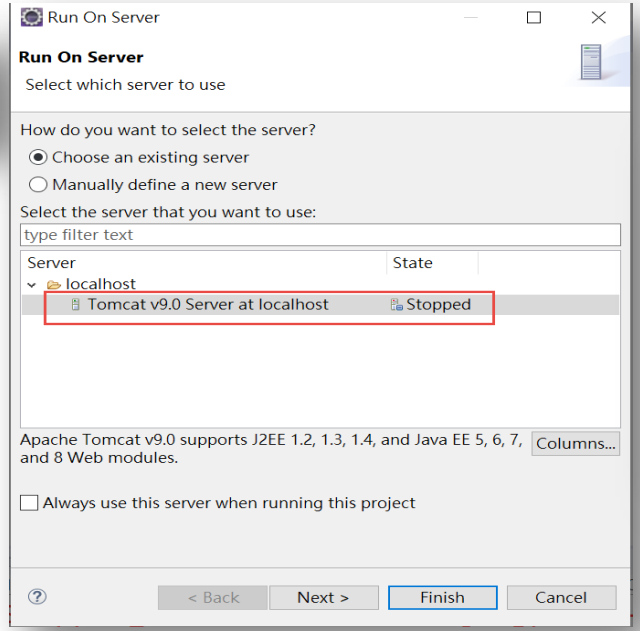


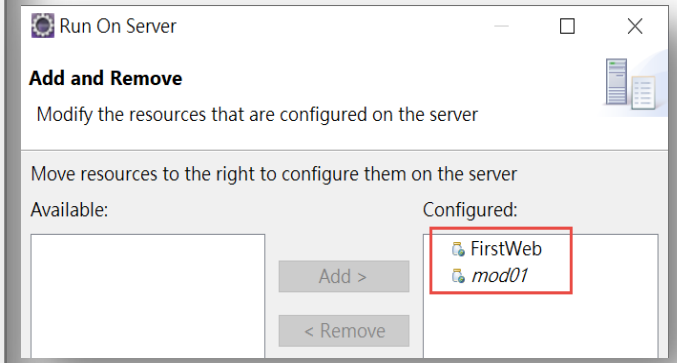


Java Web配置執行階段的Web Container

- 設定Run As/Run On Server
- 配置Tomcat 9 進行執行配置

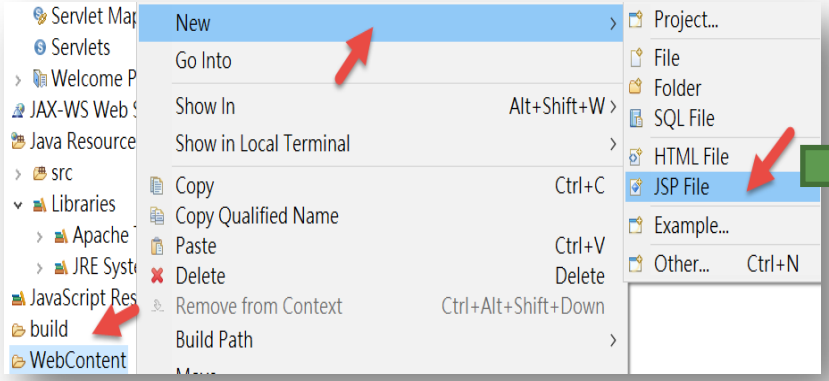


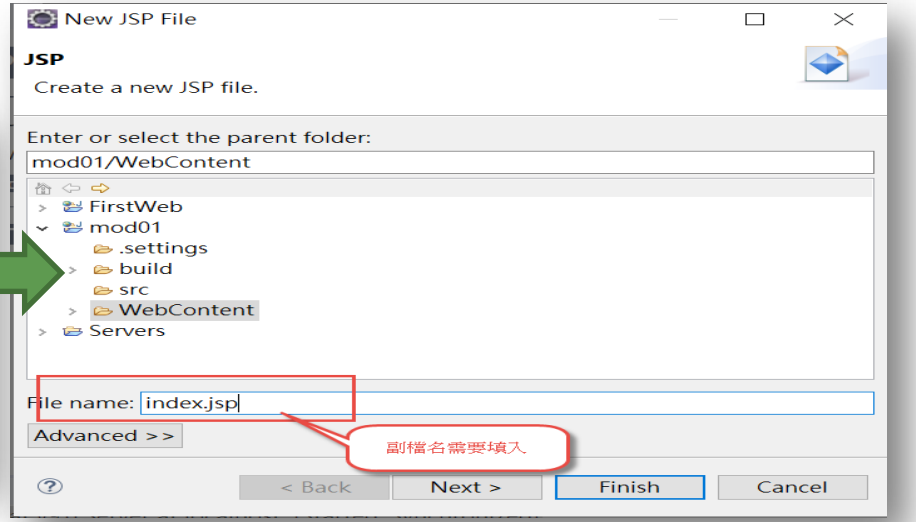




加入首頁index.jsp

- 在網站根目錄下加入index.jsp首頁。
- 執行首頁於瀏覽器上
- 確定網站可以於開發階段配置Web Container執行。

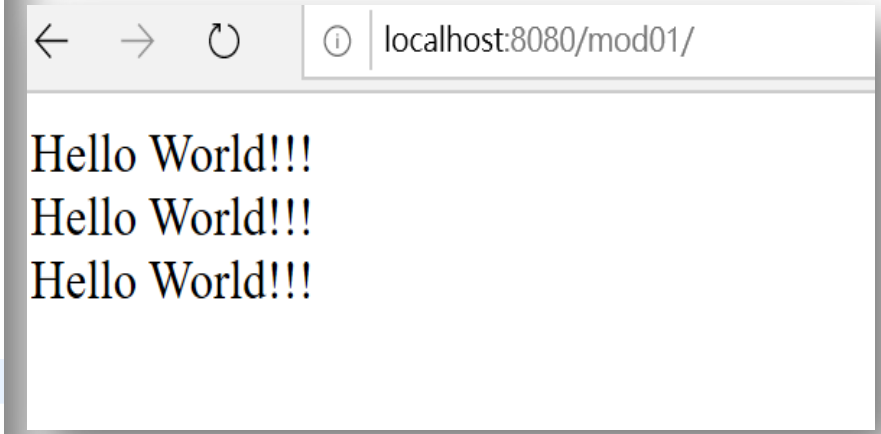




index.jsp

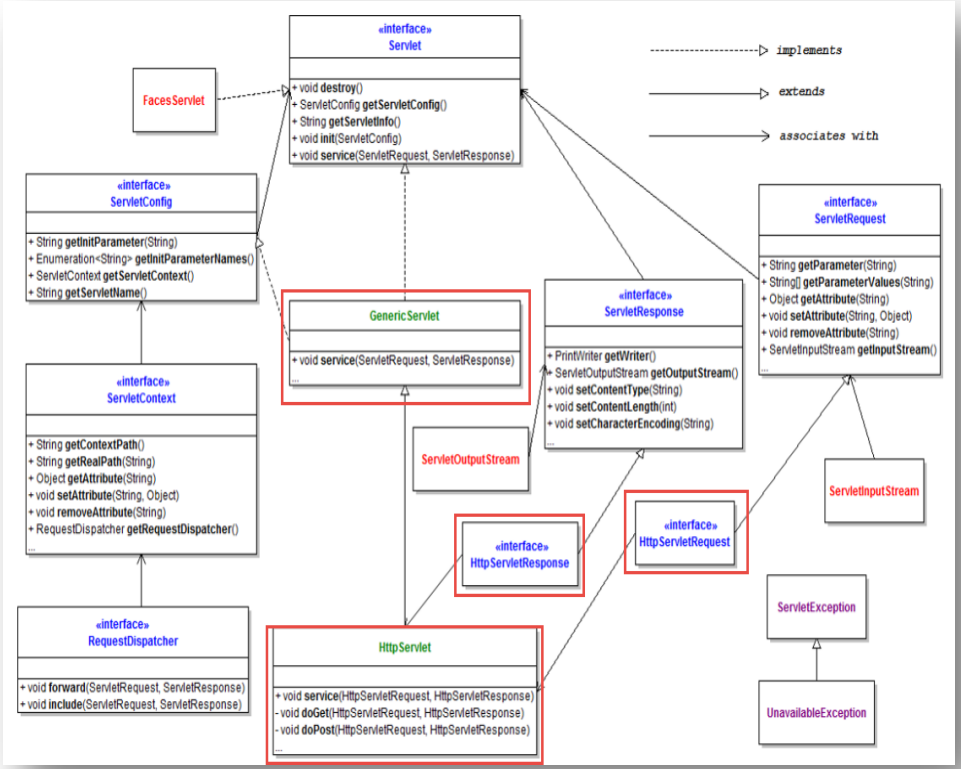
index.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>我是首頁</title>
</head>
<body>
<!-- 重要的話說三遍 -->
<%
    for(int start=0;start<3;start++){
        out.println("<li>Hello World!!!");
    }
%>
</body>
</html>
```



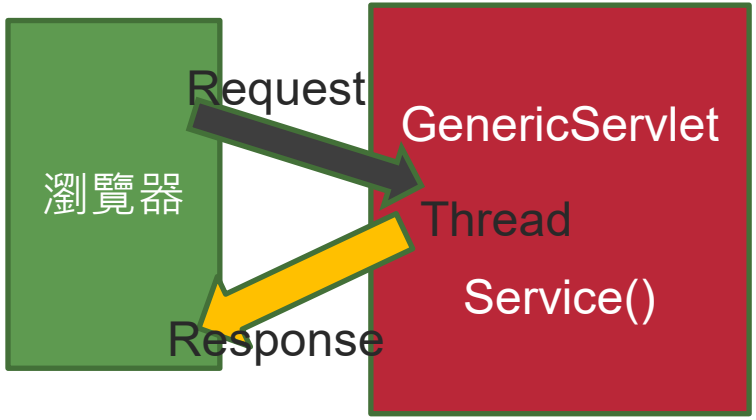
我的網站第一個Servlet

- Java EE 提供二種 Servlet 套件供開發人員使用
  - **javax.servlet**
    - 在 javax.servlet 套件中我們可以使用 GenericServlet 抽象類別，開發一般性的 Servlet 供伺服器使用。而 GenericServlet 實作 Servlet Interface。
  - **javax.servlet.http**
    - 透過網站系統架構，需要用到 HTTP 協定。通常我們會使用 HttpServlet 類別當作配接器類別依據(Adapter Pattern)來開發網站系統中的Servlet類別。
  - Servlet API線上文件
    - https://www.tutorialspoint.com/servlets/servlets-annotations.htm



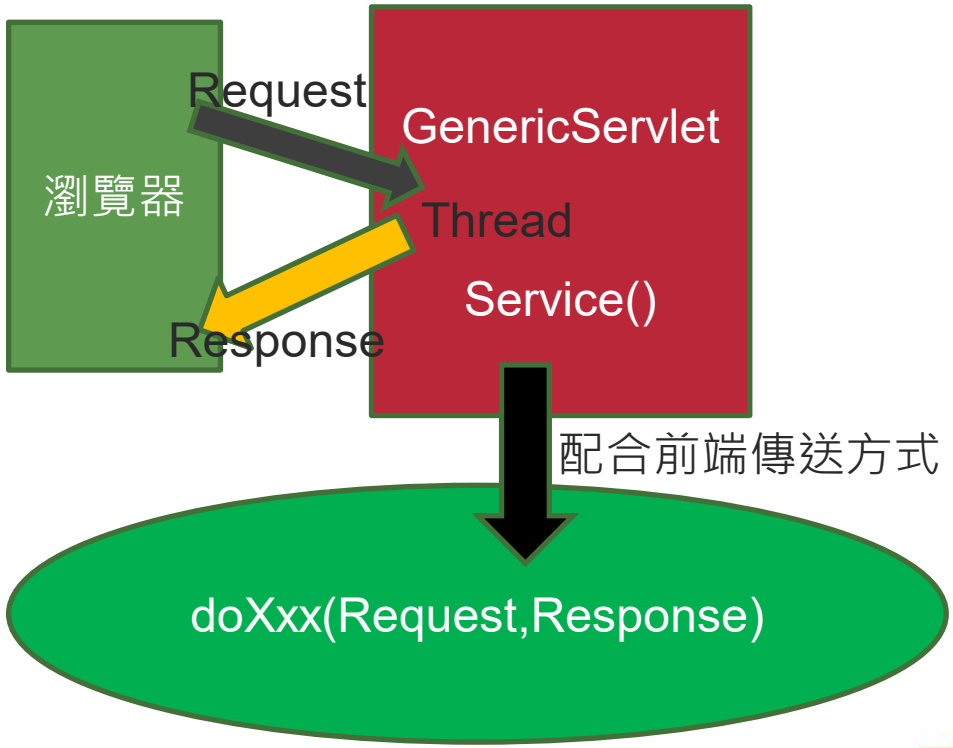
GenericServlet抽象類別

- javax.servlet.GenericServlet是一個抽象類別，用來定義Servlet API整個配接器(Adapter)工程。用來管理 Servlet 的生命週期 init()、service()，destroy() 。
- 開發一個類別繼承GenericServlet 抽象類別時，時必須要覆寫 service() 方法並處裡 client 端的請求。(有別於採用HttpServlet當作父類別時，是可以覆寫doXxx()方法進行。)



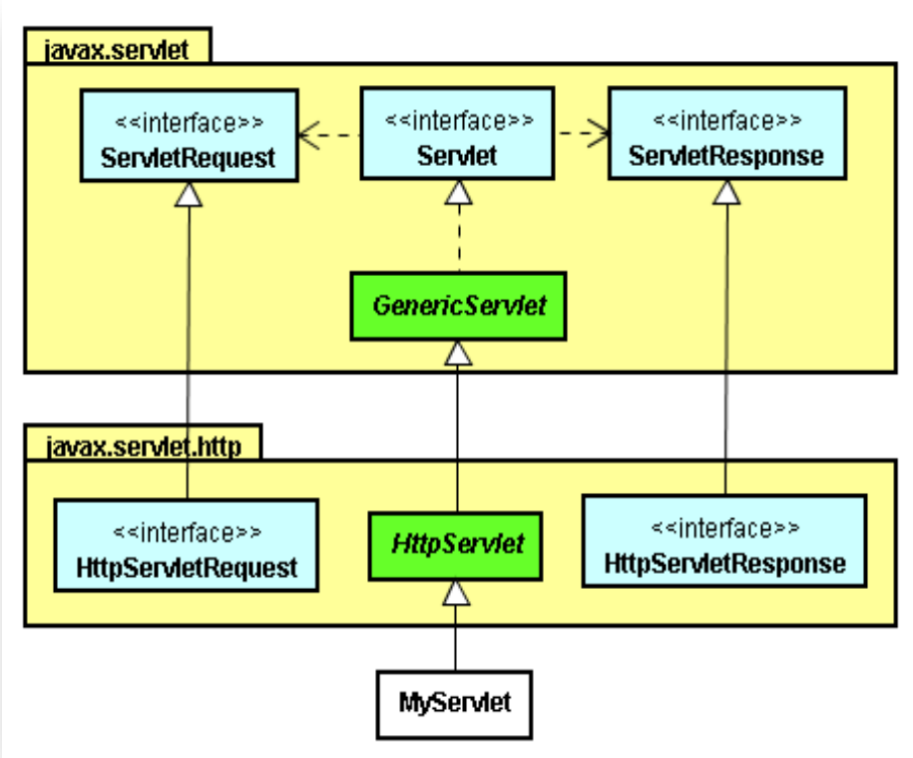
HttpServlet

- 通訊協定使用到Http協定時，可以使用HttpServlet進行繼承衍生與實作。
- 一般非必要，不要去覆寫service method。
- Service() 會自動根據前端所請求的 HTTP Request Method， 進行分派到doXXX() 相對的方法，如您有覆寫doXxx()方法。
- 如GET則分派到 doGet() 方法。
- POST 則分派到 doPost() 方法。
- 所以有關處理請求與回應的程式碼都在 doXXX() 方法中撰寫即可。



GenericServlet vs HttpServlet 類別

- 兩個均為抽象類別abstract class，作為Adapter Pattern應用。
- 一個實做到HTTP Class。
- GenericServlet為HttpServlet父類別。



我的第一個Servlet-HelloServlet

- 繼承HttpServlet class
- 覆寫doGet() Method，配合前端採用HTTP Request為GET方式
- 透過HttpServletResponse回應一個簡單的字串Hello World到前端。

HelloServlet.java

```
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
//我的第一個Servlet
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
        //透過HttpServletResponse參考出PrintWriter
        PrintWriter out=response.getWriter();
        //輸出文字串到前端
        out.println("<font size='7' color='red'>Hello World!!!</font>");
    }


}
```

撰寫Servlet必須import相關的資料類型

繼承HttpServlet抽象類別

Overriding doGet()

Servlet佈署與描述-web.xml

- 一個Servlet Class就只是一個類別規劃，如何產生一個Instance物件與如何具有URL。
- 網站是一個環境配合的應用系統(需要Web Container)，Servlet類別需要進行佈署(Deployment)，榮同一個資源先行佈署一般。才有機會產生一個Instance進行服務。
- 描述對外的網址，如此才機會被連結或者請求到。
- 使用web.xml 進行Deployment(佈署)與Descriptor(描述)。

```
<!-- 佈署Servlet -->
<servlet>
  <servlet-name>hello</servlet-name>
  <servlet-class>com.gjun.view.HelloServlet</servlet-class>
</servlet>
<!-- 描述對外網址端點 -->
<servlet-mapping>
  <servlet-name>hello</servlet-name>
  <url-pattern>/hello.view</url-pattern>
</servlet-mapping>
```

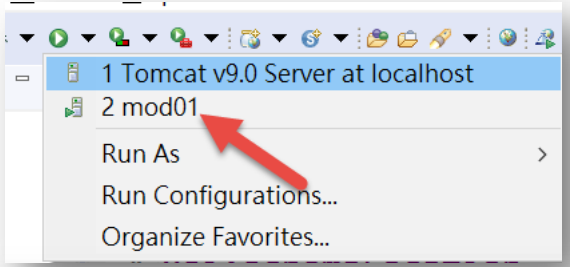
完整的敘述

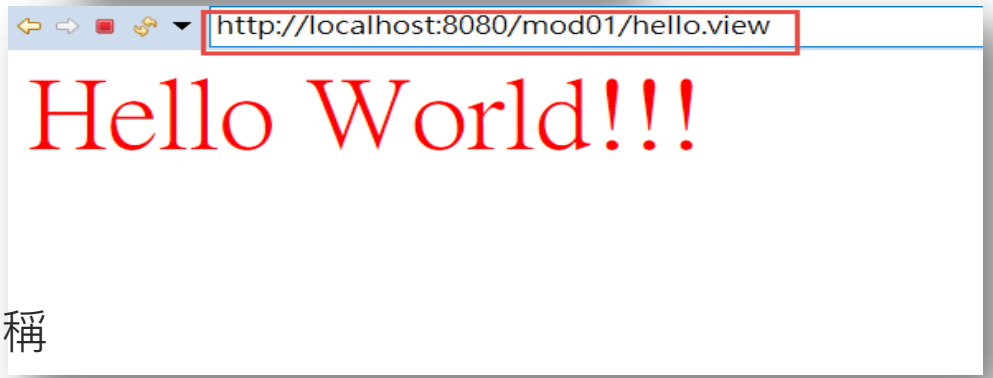
內部參照定義名稱對應

/ 網站根目錄
需要設定

運行Web Container與網站系統

- 觸發工具列上 
- 執行mod01 網站系統
- 啟動Tomcat Web Container
- URL網站入口





http通訊協定

網站系統名稱

http://localhost:8080/mod01/hello.view

Host Name

url-pattern 端點位址

我的第一個Servlet
採用Servlet 3.0 Annotation進行佈署

- Servlet 3.0可以使用Annotation(標註)進行Servlet 佈署與描述。
- 使用@WebSerlvet Annotation設定
- 撰寫Servlet Class可以同時進行佈署與描述。

```
package com.gjun.view;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(name="EricHelloServlet",urlPatterns= {"/erichello.view"})
public class EricHelloServlet extends HttpServlet {

    //支援前端採用Http Request GET方式
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //透過HttpServletResponse參考出PrintWriter物件，輸出文字串到前端
        response.getWriter().println("<font size='6' color='blue'>Hello World Eric</font>");
    }

}
```

佈署內部的 **Servlet Name**

描述url pattern
對外的端點，可以多個設定，字串陣列型別

測跑結果

- http://localhost:8080/mod01/erichello.view

web.xml與@WebServlet ServletName 佈署避免重複

- web.xml與@WebServlet描述一個Servlet Name，可能因為命名一樣而造成衝突。這時候會採用web.xml定義為優先。
- Servlet name為內部識別參照一個Servlet Class用，具有唯一性(Unique)命名規則

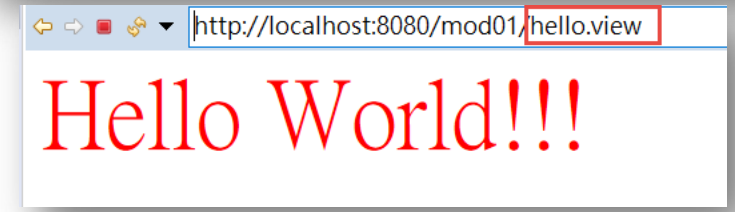
```
<servlet>
  <servlet-name>hello</servlet-name>
  <servlet-class>com.gjun.view.HelloServlet</servlet-class>
</servlet>
<!-- 描述對外網址端點 -->
<servlet-mapping>
  <servlet-name>hello</servlet-name>
  <url-pattern>/hello.view</url-pattern>
</servlet-mapping>
```

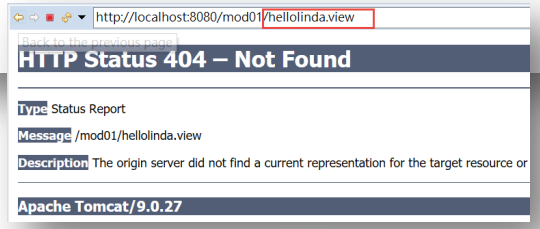
```
package com.gjun.view;

import java.io.IOException;
//我的第一個Servlet
@WebServlet(name="hello",urlPatterns="hellolinda")
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
        //透過HttpServletResponse參考出PrintWriter
        PrintWriter out=response.getWriter();
        //輸出文字串到前端
        out.println("<font size='7' color='red'>Hello World!!!</font>");
    }

}
```





與前端互動的ServletRequest ServletResponse介面應用

- ServletRequest主要是Servlet透過service Method，定義注入前端請求的資訊界面。
- 主要配合一個請求的生命週期進行與前端訊息互通的介面。

ServletRequest Method	說明
String getParameter(String name)	用來取得表單欄位或者QueryString參數名稱的內容。
String[] getParameterValues(String name)	取得指定的參數的多值內容。
Enumeration getParameterNames()	取得所有表單欄位或者URL傳遞的參數，回應一個列舉類型進行走訪逐一問出。

配合表單欄位進行傳遞方式

- <Form>表單標籤，透過action attribute指定一個Servlet URL目標
- 將表單中的表單欄位(Form Field)透過ServletRequest傳遞到後端Servlet相對方法，進行截取與處理。
- 表單欄位需要設定name attribute配合，以及Method進行傳送方法設定。
- ServletRequest.**getParameter(“name”)**進行擷取。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>您想修的課程</title>
</head>
<body>
<form method="post" action="selectcourse">
    <div>您想選修的課程</div>
    <select name="selcourse">
        <option>Python</option>
        <option>Java</option>
        <option>C#</option>
        <option>VB</option>
    </select>
    <input type="submit" value="傳送">
</form>
</body>
</html>
```

Servlet urlPattern

表單欄位名稱

```
@WebServlet("/selectcourse")
public class SelectCourseServlet extends HttpServlet{

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
        //設定回應編碼
        response.setContentType("text/html;charset=UTF-8");
        String item=request.getParameter("selcourse");
        response.getWriter().println("<h1>你挑選的課程:"+item);

    }

}
```

ServletRequest.getParameter ("form field name")

配合表單欄位進行傳遞方式 Demo

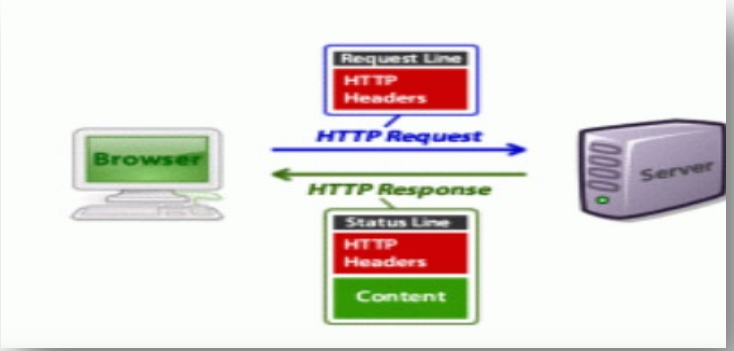
表單Get與Post Request Method有何異同

- <form method="GET" ...>
- <form method="POST" ...>
- 無論此請求中使用的特定HTTP方法如何，HttpServletRequest都將以相同的方式擷取參數名稱(表單欄位)。
  參數的存在並不一定意味著它實際上具有非空或有效值。
- 在使用參數值之前，首先驗證它們是很重要的。可以採用JavaScript進行前端驗證或者後端驗證。
- HTTP協議將所有參數作為字串傳輸，因此您可能必須解析值並轉換數據類型，可以使用Wrapper class進行。

- GET 傳送方式，如同信封內不裝信件的寄送方式，就像是明信片一樣，空間有限。
- 採用 POST 就是信封內有裝信件的寄送方式（信封有內容物）或者如同寄送包裹一般。

請求中傳遞的Header

- HTTP傳送部分可區分為
  - Header
    - HTTP Headers是HTTP請求和相應的核心，它承載了關於用戶端流覽器，請求頁面，伺服器等相關的資訊。
  - Body
    - HTML網頁內容
    - 文件
    - 非文字格式的檔案



▲ 要求標頭
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-Hant-TW, zh-Hant; q=0.8, en-US; q=0.5, en; q=0.3
Cache-Control: no-cache
Host: widgets.wp.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G...
▲ 回應標頭
cache-control: max-age=31536000
content-encoding: gzip
content-type: text/css
date: Sat, 02 Nov 2019 14:48:29 GMT
etag: W/"5d923dd8-293"
expires: Sat, 31 Oct 2020 13:08:24 GMT
server: nginx
vary: Accept-Encoding
x-ac: 2.tpe _bur
x-nc: HIT tpe 1

分析 Header

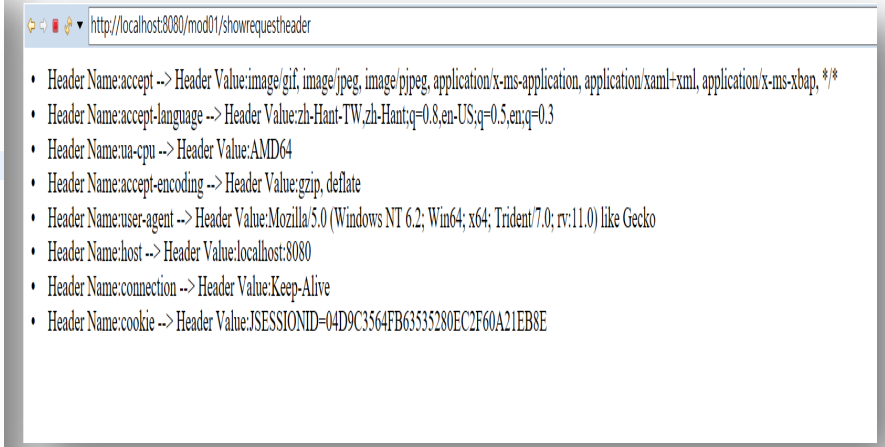
方法名稱	說明
String getHeader(String headerName)	取得指定的Header Name名稱的內容值。
Enumeration getHeaders(String headerName)	取得指定的Header Name的集合資料(多值)
Enumeration getHeaderNames()	取得所有Header Name回應一個列舉進行走訪逐一參考。
int getIntHeader(String headerName)	取得Header Name回應一個整數。

取得一個請求的Header所有狀態

- 透過HttpServletRequest.getHeaderNames()取出所有Header Name
  - Enumeration介面操作走訪
- 走訪每一個Header Name，相對Header Name取出相對的Header值

```
@WebServlet("/showrequestheader")
public class ShowRequestHeaderServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException
    {
        //取出Request 所有的Header Name
        Enumeration<String> headerNames=request.getHeaderNames();
        //走訪每一個Header Name
        while(headerNames.hasMoreElements()) {
            //問出相對的Header Name
            String name=headerNames.nextElement();
            //取出相對的Header name對應的value
            String value=request.getHeader(name);
            //輸出
            response.getWriter().printf("<li>Header Name:%s --> Header Value:%s",name,value);
        }
    }
```



ServletResponse送出回應

- ServletResponse介面。主要是回應訊息到前端。
- 當一個Serlvet產生一個個體物件之後，前端請求，ServletResponse物件會於service()被注入(Injection)。
- 利用 ServletResponse 所提供的方法來回應訊息內型與內容到前端：
  - setContentType()
  - getWriter()
  - getOutputStream()
- 利用 HTTPServletResponse 所提供的方法來回應，主要是針對Http這邊延伸的部分：
  - setHeader()
  - setStatus()
  - sendRedirect()
  - sendError()

回應訊息的編碼與Content Type

- ServletResponse.setContentType()
  - 設定 response 的內容型態與編碼方式：
    - setContentType("text/html;charset=utf-8");
    - setContentType Method 預設為 text/html
    - setContentType Method應宣告在 PrintWriter 或任何回應物件宣告之前
    - Content Type(或者Media Type/MIME Type)參考文件
      - https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types

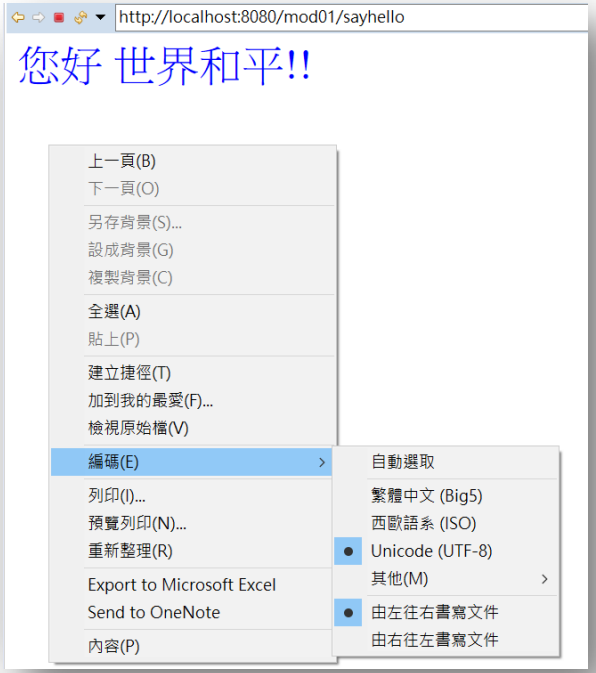
ServletResponse回應HTML文字串

SayHelloServlet.java

```java
@WebServlet("/sayhello")
public class SayHelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //設定回應編碼與Content Type
        response.setContentType("text/html;charset=UTF-8");
        //取出PrintWriter 寫出字串
        PrintWriter out=response.getWriter();
        out.println("<font size='6' color='blue'>您好 世界和平!!</font>");

    }
```



ServletResponse回應非文字

- 採用getOutputStream()方法取出OutputStream物件。
- 可以回應非文字檔案至前端，如圖片或者是PowerPoint/PDF等檔案。
- 在同一個Servlet撰寫中不能同時使用PrintWriter()與OutputStream()兩種方併行。進行文字檔與檔案下載作業，將會產生IllegalStateException例外錯誤。

PhotoLoadServlet.java

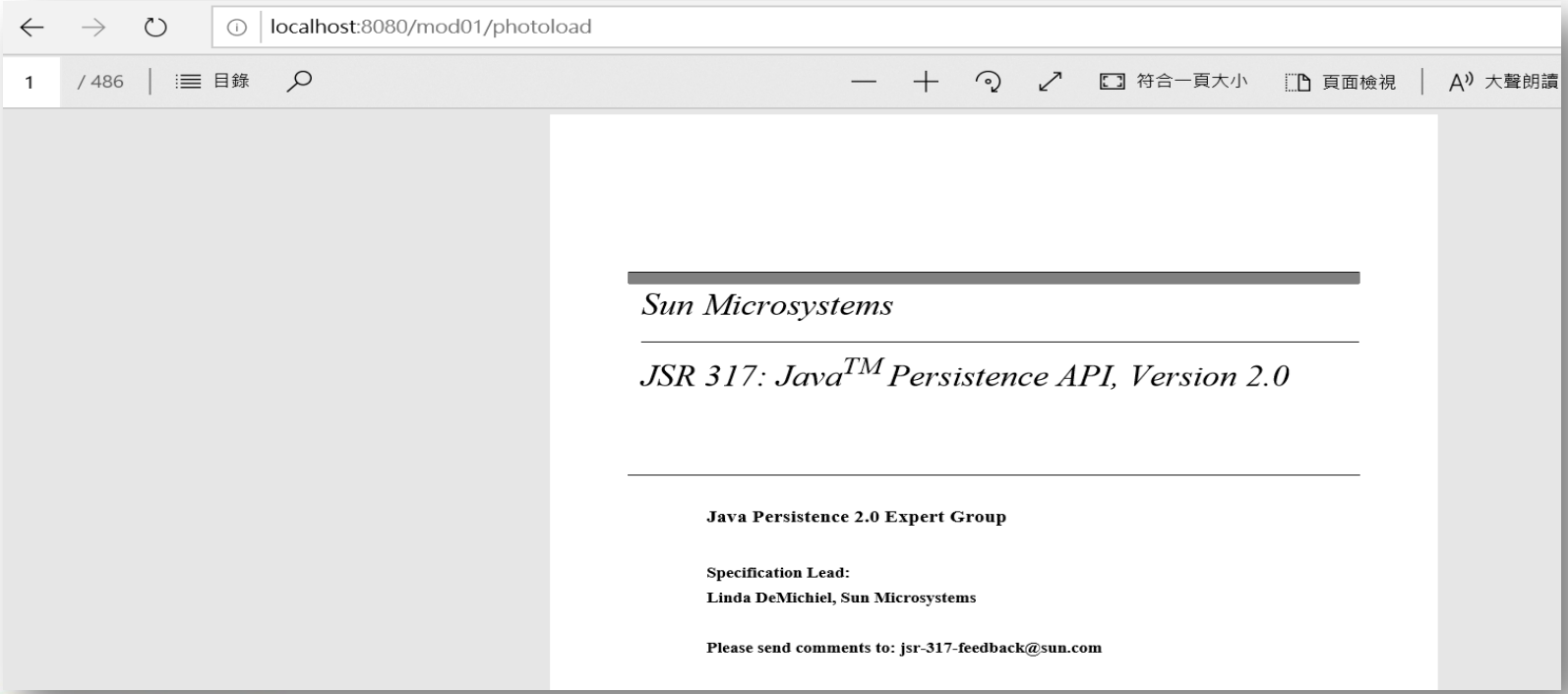
```
//檔案下載Servlet
@WebServlet("/photoload")
public class PhotoLoadServlet extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Servlet
        //設定回應的ContentType
        response.setContentType("application/pdf");
        //網站相對目錄下的文件實際目錄對應 取得應用系統對應的物件ServletContext
        ServletContext context=this.getServletContext();
        InputStream is=new FileInputStream(context.getRealPath("/docs/Persistence JPA.pdf"));
        //讀取檔案至byte陣列緩存區
        byte[] buffer=new byte[is.available()];
        is.read(buffer,0,buffer.length);
        //取出Response參照下的OutputStream
        OutputStream out=response.getOutputStream();
        out.write(buffer,0,buffer.length);
        out.flush();
        is.close();
        out.close();


    }
```

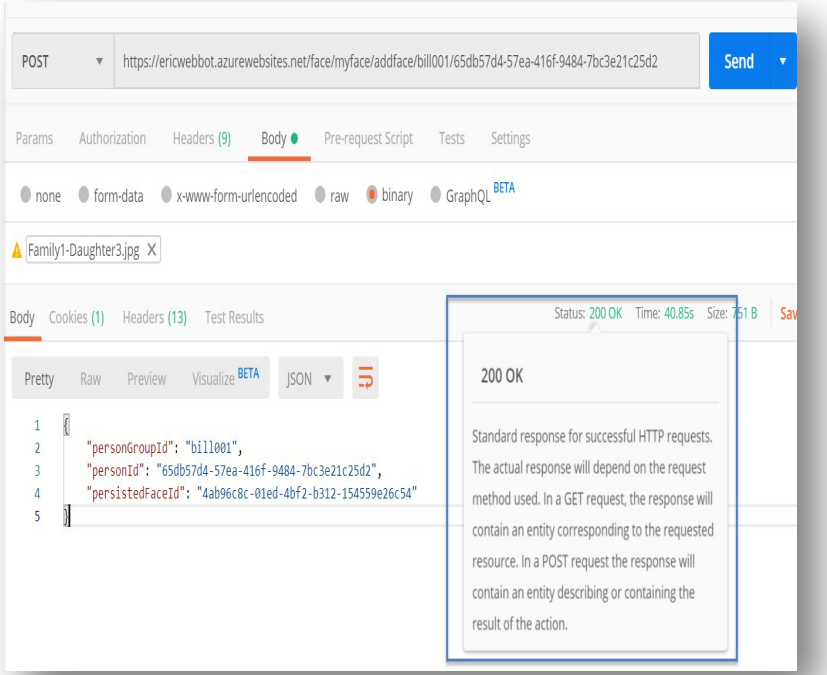
ServletResponse回應非文字

- PhotoLoadServlet.java 執行結果



送出錯誤回應

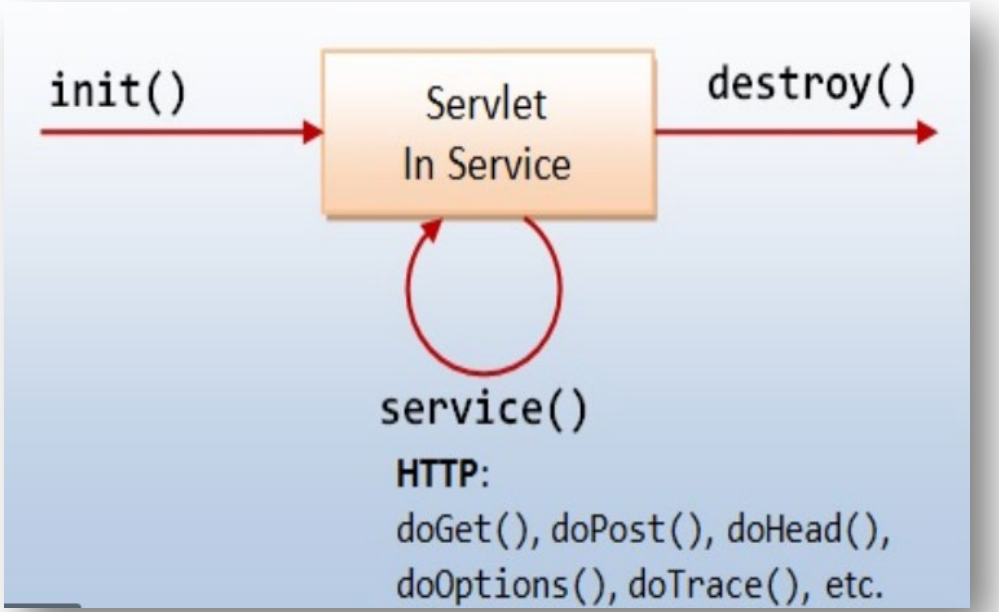
- HttpServletResponse.sendError()
  - 傳送所指定的狀態碼(Status code)前端，瀏覽器會顯示適當的資訊。
  - Status Code請Follow HTTP狀態碼相關設定
  - https://developer.mozilla.org/zh-TW/docs/Web/HTTP/Status
  - res.sendError(res.SC_UNAUTHORIZED)
    res.sendError(res.SC_UNAUTHORIZED, "This is UNAUTHORIZED");
- 若要自定 error 訊息給 client 端請使用 sendError() 而不建議使用 setStatus()



使用postman送出一張相片
到Face AI進行註冊與訓練回應的結果

Servlet生命週期

- Servlet 生命週期
  - 載入：Load
  - 初始化：Initializing
  - 執行與銷毀：Servcing & Destroy
  - 卸載：Unload



Servlet 生命週期的方法

方法名稱	簡要說明
void init(ServletConfig config)	Web Container呼叫此方法來進行物件初始化，必須呼叫父類型處理ServletConfig。
void init()	可覆寫此方法自行撰寫 Servlet 初始化時的客製化流程。
void service(ServletRequest req, ServletResponse res)	服務前端的要求 (Request)，並且注入封裝前端所有資訊的ServletRequest與ServletResponse物件。
void destroy()	一個Servlet被卸離(unload)時，引發Servlet會執行destroy Method。可以進行資源回收程序設計。

載入：Load

- 當 Web Container 啟動時，會依照 web.xml或 @Annotation 中的定義載入 Servlet 實體。
- Web Container 會呼叫 Class.forName(“className”).newInstance()(Reflection機制) ，以建立 Servlet 的物件個體。
- 接著呼叫 Servlet 的預設建構子 (空參數建構子) 。

初始化：Initializing

- Initializing
  - init(ServletConfig config)與init()差異
  - 實作上以覆寫 init() 為主，若一定要覆寫 init(ServletConfig config)，記得要針對傳遞進來的ServletConfig進行處理。一般在程序中撰寫super.init(ServletConfig config) ，目的是將現行的組態傳給 GenericServlet。

- Servlet 物件會產生幾個？
  - 一般的情況下， 只會在初始時呼叫 init(ServletConfig config) 一次，因而僅會有一個物件產生。另外 init() 也因此只會呼叫一次。接下來任何的請求，都是service配合執行緒進行之。

執行與銷毀

- Destroy：
  - 當 Web Container 認為 一個Servlet 已經過久沒有被使用了，就會自行呼叫該 Servlet 的 destroy() 方法，將資源回收給系統。
  - 另外當 Web Container 正在卸載時，也會呼叫 Servlet 的 destroy() 方法。
  - 我們會覆寫destroy Method一般用來回收資源，如資料連接資源，或者網路連線資源等。

執行與銷毀

- 覆寫 init() 與 destroy() 的時機
  - 一般而言我們會將 servlet 在 init() 所宣告使用的資源，並於 destroy() 中釋放
  - 例如：I/O、Socket、資料庫連線等。

卸載：Unload

- Unload：
  - 當 Servlet 的 destroy() 被呼叫時，表示這一個Servlet 的生命週期即將結束
  - 並且系統會針對該 Servlet 物件進行資源回收動作。
  - 不論是Servlet 因閒置過久，是因 Web Container 進行卸離(停止服務)，Servlet 都將會被 unload下來。

Lab

- 建構網站中的第一個Servlet並且進行佈署與執行
- 採用web.xml進行Servlet部署
- 採用@WebServlet Annotation進行Servlet佈署
- 如何透過Servlet擷取到前端瀏覽器傳送進來的表單資訊
- 撰寫一個Servlet擷取前端傳送進來的Http Headers
- 透過Servlet回應一個字串訊息到前端
- 撰寫一個擷取磁碟中的圖片進行下載瀏覽器端Servlet
- Servlet生命週期有哪些