Java魔法營

高級(基石之作)

羯聲 capricorn.rich.cat@gmail.com

Java Web

應用與實務

- Servlet
  - Servlet Model
  - Web Application
  - Web Container
  - Session
  - Listener
  - Security

Java Web

應用與實務

- Servlet Model
  - HTTP
  - Request & Response
  - Servlet Lifecycle

Java Web

Servlet Model

- 超文字傳輸協議(HyperText Transfer Protocol, HTTP)
  - 無狀態協議(stateless)
  - client-server
  - HTTP version
  - request
    - HTTP method
  - response
    - HTTP status code

Servlet Model

HTTP

- 無狀態協議(stateless)
  - 每個request都是獨立的，server不會保留任何狀態
  - 使用cookie跟session解決request的前後關聯

Servlet Model

HTTP

- client 客戶端
  - 發送request並處理response
  - 瀏覽器、手機、PC
- server 服務器端
  - 處理request並將結果response
  - 被動的等待request
  - 預設使用TCP port 80

Servlet Model

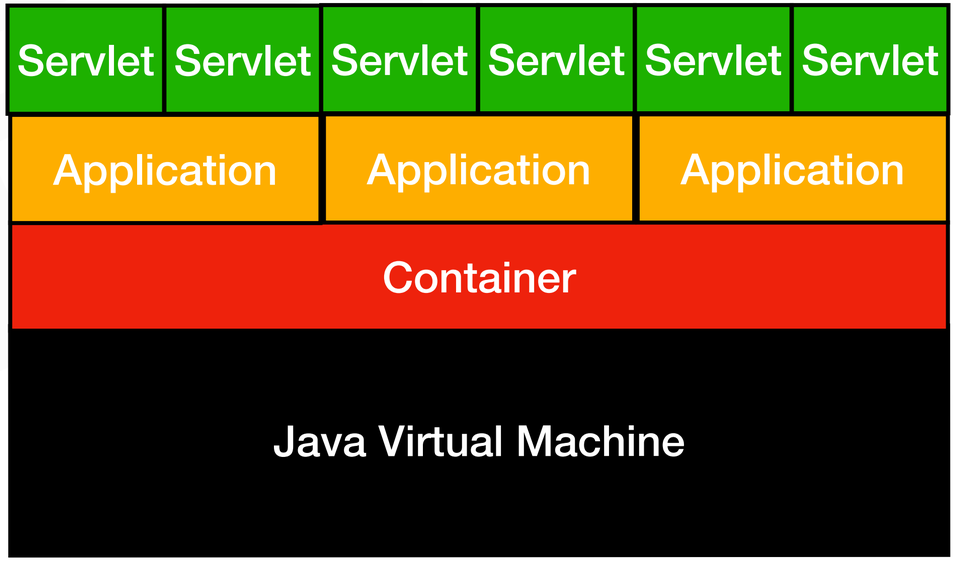
HTTP

- HTTP version
  - HTTP/0.9
    - 只有GET及HTML
  - HTTP/1.0
    - 增加一些功能，如POST、status code
  - HTTP/1.1
    - 持久連線(1個TCP連線可發送多個request)
    - 管道(依順發送request，不等待response，前後request可能阻塞)
  - HTTP/2
    - 多路復用(並發request)
    - server push(伺服器推送資源)
  - HTTP/3
    - TCP改為QUIC

Servlet Model

HTTP

- 提供HTTP服務
  - Web Server
    - Apache, Nginx, IIS
  - Web Container (Servlet)
    - Tomcat, GlassFish, Jetty, WildFly
    - JBoss, WebLogic, WebSphere



Servlet Model

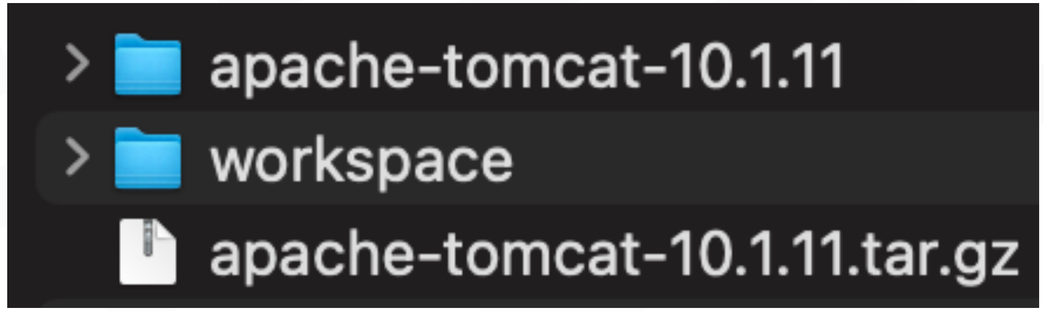
HTTP

- 動態網站
  - 早期使用CGI(Common Gateway Interface)
  - Java使用Servlet進行Request & Response的處理
    - 使用Web Container管理Servlet
    - Web Container不建議直接對外開放
      - 使用Apache或Nginx做反向代理(reverse proxy)
  - Java外的其他選擇：ASP, PHP, NodeJS, Python

Servlet Model

HTTP

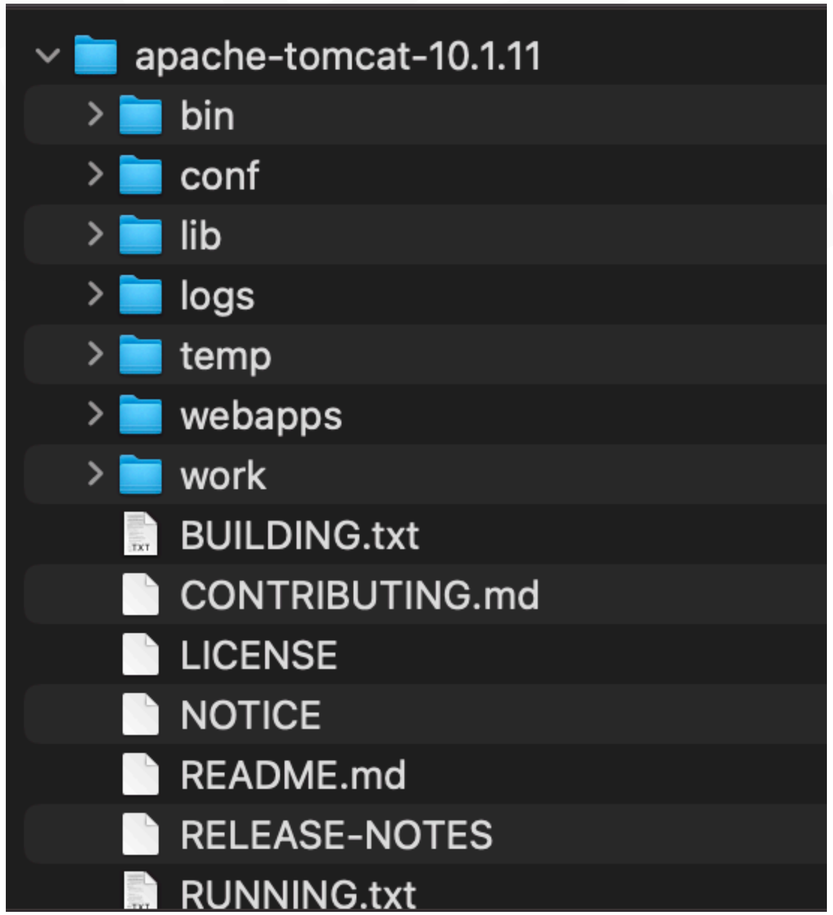
- 準備工作區
  - workspace
- 安裝Tomcat
  - 下載後解壓縮



Servlet Model

HTTP

- Tomcat裡的目錄
  - bin　相關執行檔
  - conf　相關設定檔
  - lib　相關函式庫
  - logs　日誌檔
  - temp　暫存檔
  - webapps　應用程式
  - work　JSP工作目錄



Servlet Model

HTTP

- Tomcat裡會用到的檔案
  - bin 相關執行檔
    - startup.sh
    - shutdown.sh
    - catalina.sh run
  - conf 相關設定檔
    - server.xml
    - tomcat-users.xml
    - web.xml
  - lib 相關函式庫
    - servlet-api.jar
    - jsp-api.jar
  - logs 日誌檔
  - temp 暫存檔
  - webapps 應用程式
    - war佈署位置
  - work JSP工作目錄

Servlet Model

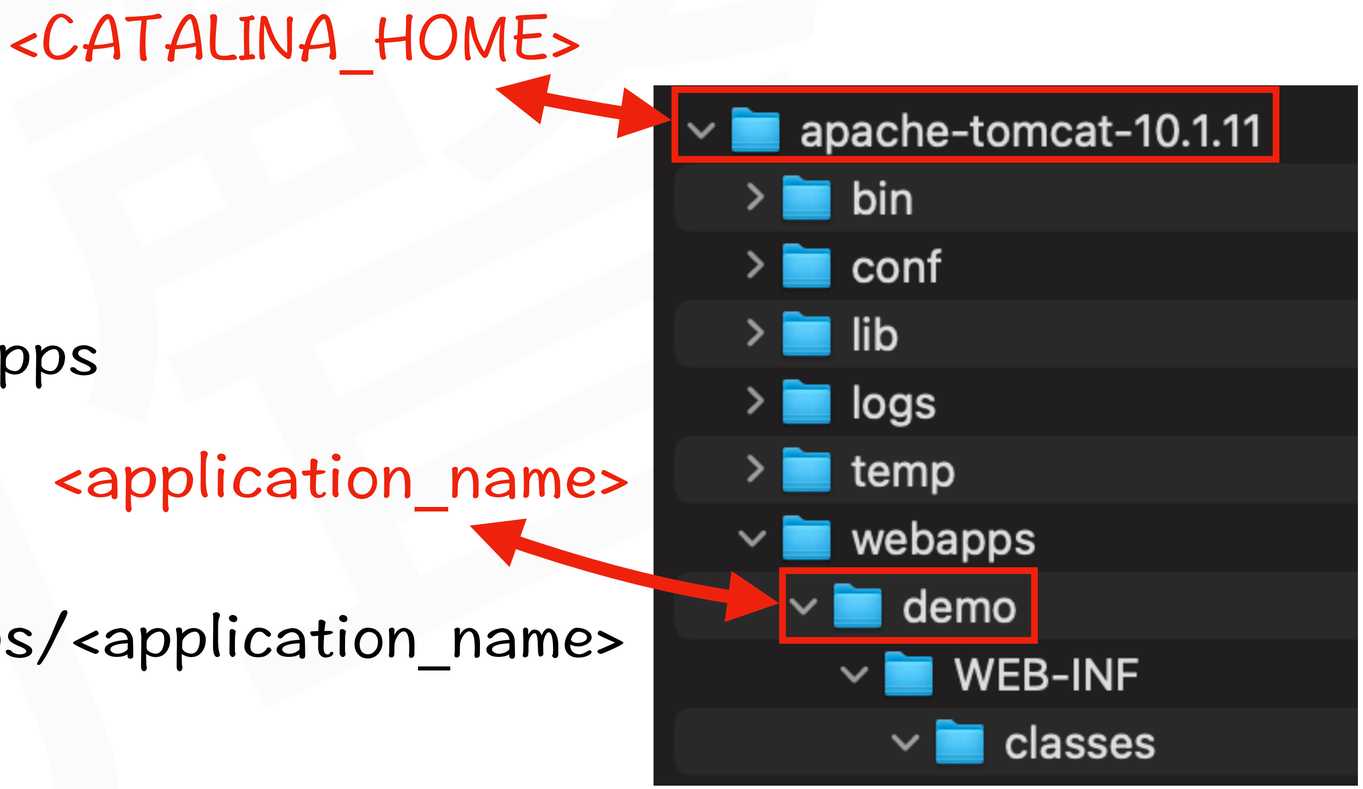
HTTP

- Servlet
  - 準備應用程式佈署目錄
  - 在工作區撰寫Servlet
  - 編譯Servlet
  - 執行網頁

Servlet Model

HTTP

- 準備應用程式佈署目錄
  - 進入<CATALINA_HOME>/webapps
  - 建立應用程式名稱的目錄
    - <CATALINA_HOME>/webapps/<application_name>
  - 建立servlet存放目錄
    - <CATALINA_HOME>/webapps/<application_name>/WEB-INF/classes

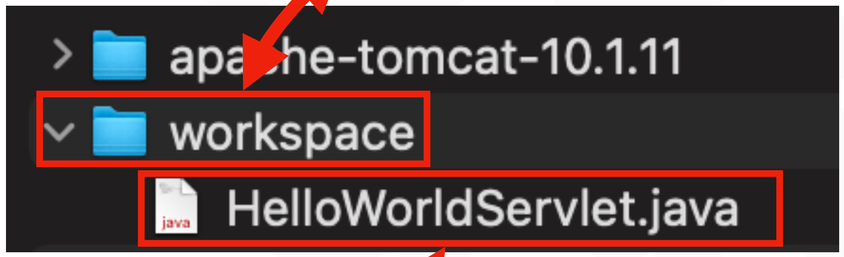


Servlet Model

HTTP

- 在工作區撰寫Servlet

工作區目錄



Servlet

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebServlet("/hello")
public class HelloWorldServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Hello, World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Servlet Model

HTTP

- 編譯Servlet
  - javac -d ../apache-tomcat-10.1.11/webapps/demo/WEB-INF/classes -cp ../apache-tomcat-10.1.11/lib/servlet-api.jar *.java
- 執行網頁
  - 啟動tomcat
    - 進入tomcat目錄，執行 ./startup.sh 或 ./catalina.sh run
  - 使用瀏覽器
  - http://localhost:8080/demo/hello

IP or domain

webapps下的目錄名稱

http://localhost:8080/demo/hello

協定名稱

port

servlet設定的名稱

Java Web

Servlet Model

- Request
  - header與body組成，中間隔一個空白行
  - header
    - request line
      - HTTP method
      - resources URI
      - HTTP version
    - header field (parameter)
  - body
    - application/x-www-form-urlencoded
    - multipart/form-data
    - XML
    - JSON

Servlet Model

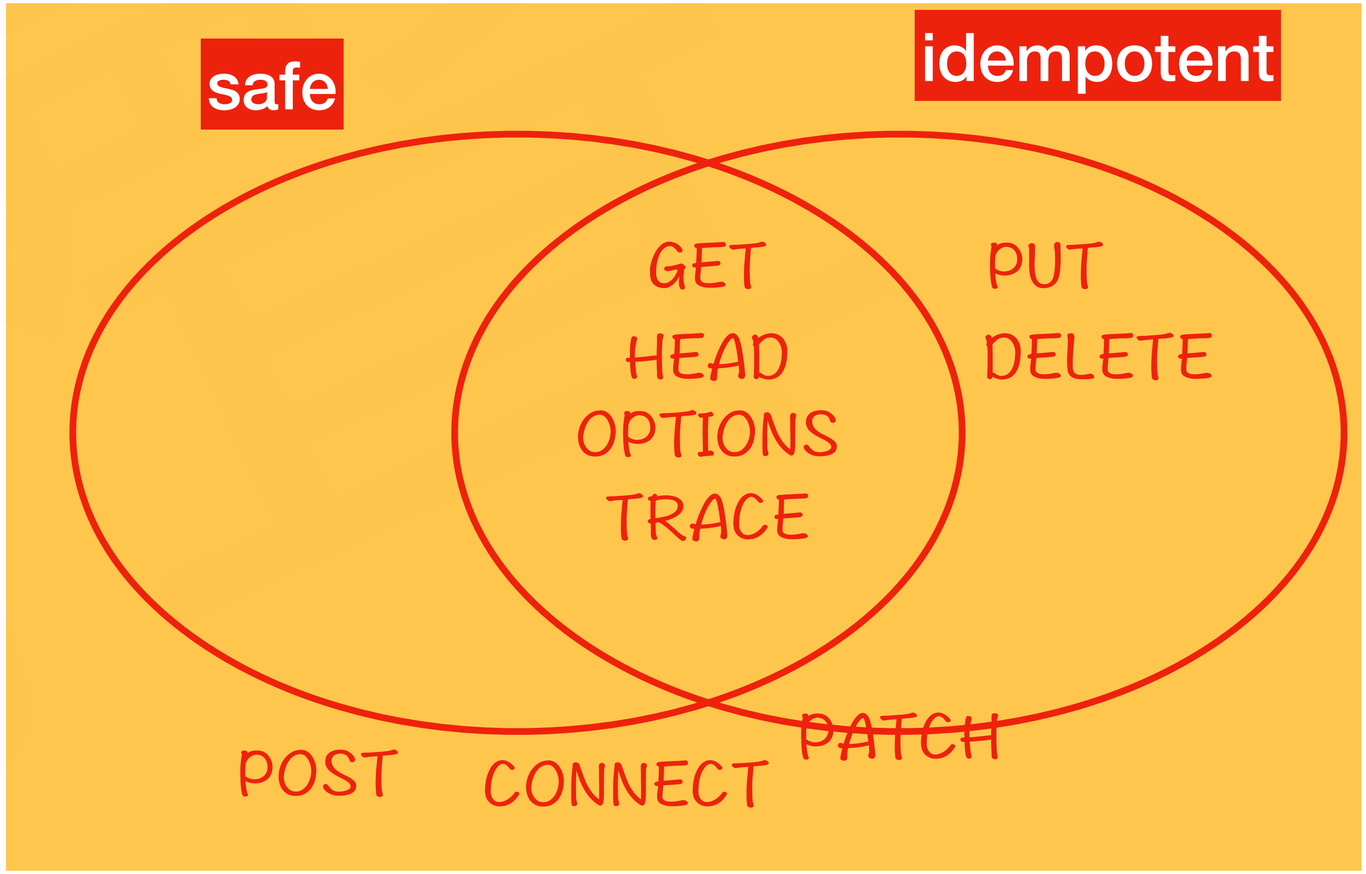
Request

- HTTP method
  - GET 取得資源
  - POST 新增資源
  - PUT 修改資源(完整)
  - DELETE 刪除資源
  - HEAD 取得資源的資訊
    - response header
- OPTIONS 測試伺服器功能
- TRACE 測試伺服器連線
  - 可能不支援
- CONNECT 要求代理伺服器對遠端伺服器進行連線
  - 通用使用在SSL
- PATCH 修改資源(部份)
  - 可能不支援

Servlet Model

Request

- HTTP method
  - 安全性(Safe)
    - 不會改變資源的狀態
  - 等冪性(Idempotent)
    - 執行一次或多次結果都一樣



Servlet Model

Request

- URL
  - scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]
  - https://www.example.com:80/path/to/myfile.html?key1=value1#SomewhereInTheDocument
  - queryString的格式為 ?name1=value1&name2=value2
    - 多值 ?name1=value1&name11=value12&name2=value21&name2=value22

Servlet Model

Request

- resources URI
  - URL Encoding 百分號編碼
  - 當URL中包含非字母數字或者特定符號，或使用POST請求並且內容類型為"application/x-www-form-urlencoded"，請求本體中的參數也需要進行URL編碼
    - 特殊字元轉成 "%"+ASCII十六進位碼
    - 非ASCII字元轉成 "%"+UTF-8十六進位碼

Servlet Model

Request

- header field (parameter)
  - 用來描述request和request body的資料，格式為 name: value1, value2
    - Accept：告訴server，client可以接受的回應類型
    - Authorization：傳送驗證身分的資料
    - Content-Type：request body的資料類型
    - User-Agent：描述client端的訊息，如作業系統、瀏覽器
    - Host：client送出的URL中host跟port，於虛擬主機環境可使server知道client實際的去向
      - 虛擬主機：1個IP對應多個網站

Servlet Model

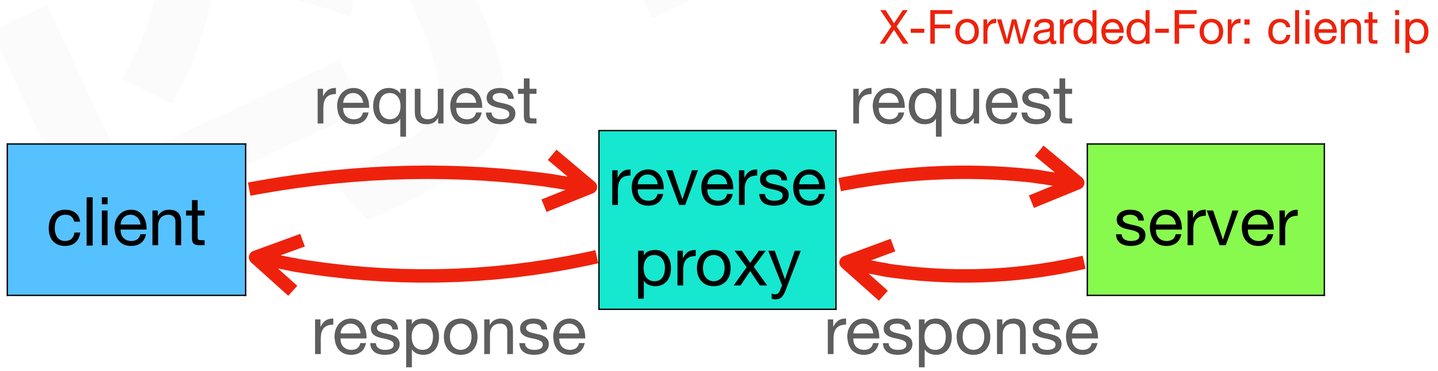
Request

- header field (parameter)
  - 用來描述request和request body的資料，格式為 name: value1, value2
    - Origin：跨網站發送request，需加上原網站的網址
      - CORS：Cross-Origin Resource Sharing(跨來源資源共享)

Servlet Model

Request

- header field (parameter)
  - 用來描述request和request body的資料，格式為 name: value1, value2
    - X-Forwarded-For：reverse proxy或load balancer標示真正client的IP



Servlet Model

Request

- header field (parameter)
  - 用來描述request和request body的資料，格式為 name: value1, value2
    - Cookie：存放response header中Set-Cookie的值，server可以使用這些Cookie來識別和追蹤client
      - 會話管理
      - 個人化設定
      - 追蹤和分析使用者行為

Servlet Model

Request

- application/x-www-form-urlencoded
  - 在request body中傳送表單資料
  - 格式與query string相同
    - name1=value1&name2=value2
    - name1=value1&name11=value12&name2=value21&name2=value22
  - 需進行URL Encoding

Servlet Model

Request

- multipart/form-data
  - 在request body中傳送二進位檔案或多部份資料
    - 每個部份都有自己的標頭資訊
  - 傳送二進位檔案需進行base64編碼

每個部份的開始

資料結束

```
--boundary
Content-Disposition: form-data; name="field1"

value1
--boundary
Content-Disposition: form-data; name="field2"; filename="example.txt"
Content-Type: text/plain

example data
--boundary--
```

Servlet Model

Request

- multipart/form-data
  - 在request body中傳送二進位檔案或多部份資料
    - 每個部份都有自己的標頭資訊
  - 傳送二進位檔案需進行base64編碼

```
--boundary
Content-Disposition: form-data; name="field1"

value1
--boundary
Content-Disposition: form-data; name="field2"; filename="example.txt"
Content-Type: text/plain

example data
--boundary--
```

每個部份的標頭

Servlet Model

Request

- multipart/form-data
  - 在request body中傳送二進位檔案或多部份資料
    - 每個部份都有自己的標頭資訊
  - 傳送二進位檔案需進行base64編碼

參數名稱

```
--boundary
Content-Disposition: form-data; name="field1"

value1
--boundary
Content-Disposition: form-data; name="field2"; filename="example.txt"
Content-Type: text/plain

example data
--boundary--
```

參數值

Servlet Model

Request

- multipart/form-data
  - 在request body中傳送二進位檔案或多部份資料
    - 每個部份都有自己的標頭資訊
  - 傳送二進位檔案需進行base64編碼

檔名及類型

```
--boundary
Content-Disposition: form-data; name="field1"

value1
--boundary
Content-Disposition: form-data; name="field2"; filename="example.txt"
Content-Type: text/plain

example data
--boundary--
```

Servlet Model

Request

- binary(base64)
  - HTTP只能傳送純文字內容，二進制檔案需透過base64轉換為文字
    - A-Z, a-z, 0-9, +, / 共64個
  - Man -> ASCII的 77, 97, 110
    - 轉二進制 77 = 01001101, 97 = 01100001, 110 = 01101110
    - 八個位元為一組 01001101 01100001 01101110
    - 改成六個位元 010011 010110 000101 101110
      - 位元數需為3的倍數(其實是3跟8的公因數24)，不足補0
    - 19, 22, 5, 46 對照 base64 得到 TWFu

Servlet Model

Request

- binary(base64)
  - HTTP只能傳送純文字內容，二進制檔案需透過base64轉換為文字
    - A-Z, a-z, 0-9, +, / 共64個
  - Ma -> ASCII的 77, 97
    - 轉二進制 77 = 01001101, 97 = 01100001
    - 八個位元為一組 01001101 01100001
    - 改成六個位元 010011 010110 000100 000000
      - 位元數需為3的倍數(其實是3跟8的公因數24)，不足補0
    - 19, 22, 4 對照 base64 得到 TWF=

Servlet Model

Request

- XML(Extensible Markup Language)
  - 標記語言，用於編碼文檔中的結構化資訊
    - 可自定格式
    - 可使用DTD(Document Type Definition)進行規範
  - 標籤有起始和結尾
    - 每個標籤都有名稱和屬性
    - 標籤本體可包含文字和標籤

起始標籤

標籤屬性

```
<persons>
    <person id="1">
        <name>John Doe</name>
        <age>30</age>
        <city>New York</city>
    </person>
    <person id="2">
        <name>Emily Johnson</name>
        <age>25</age>
        <city>Los Angeles</city>
    </person>
</persons>
```

標籤本體包含文字

標籤本體包含標籤

結尾標籤

Servlet Model

Request

persons包含0~n個person

person包含name, age, city

```
<!DOCTYPE persons [
<!ELEMENT persons (person*)>
<!ELEMENT person (name, age, city)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT city (#PCDATA)>
]>
```

person的屬性id

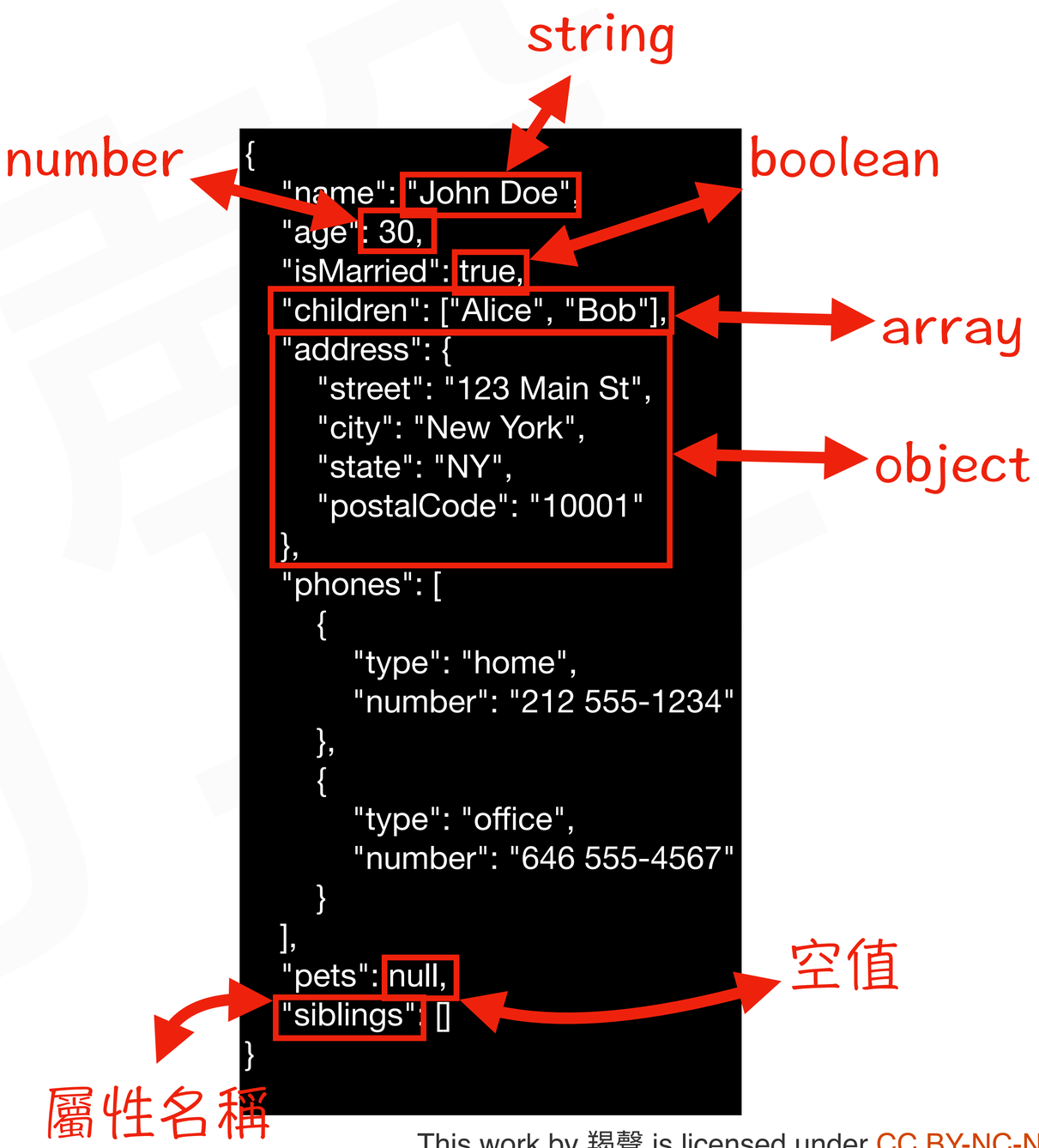
name, age, city包含文字

```
<persons>
  <person id="1">
    <name>John Doe</name>
    <age>30</age>
    <city>New York</city>
  </person>
  <person id="2">
    <name>Emily Johnson</name>
    <age>25</age>
    <city>Los Angeles</city>
  </person>
</persons>
```

Servlet Model

Request

- JSON(JavaScript Object Notation)
  - 輕量級資料交換格式
  - 資料型態
    - number 有正負，不分整數浮點
    - string 使用" "
    - boolean
    - array
    - object




```
{
  "name": "John Doe",
  "age": 30,
  "isMarried": true,
  "children": ["Alice", "Bob"],
  "address": {
    "street": "123 Main St",
    "city": "New York",
    "state": "NY",
    "postalCode": "10001"
  },
  "phones": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "pets": null,
  "siblings": []
}
```

Servlet Model

Request

- RESTful
  - 每個資源都有獨立的URI
  - 使用HTTP method進行資料的增刪改查
    - GET查詢, POST新增, PUT修改, DELETE刪除
  - 無狀態，需自行實作JWT(JSON Web Token)
  - 資料格式可用XML或JSON，以JSON為主
  - 可參考JAX-RS 2.0 及 Jersey

Servlet Model

Request

- HttpServletRequest
  - String getParameter(String name)
  - query string 或 form

```html
<!DOCTYPE html>
<html>
<body>

<h2>HTML Form</h2>

<form action="/demo/parameters" method="POST">
  <label for="name">Name:</label><br>
  <input type="text" id="name" name="name" value="John Doe"><br>
  <label for="age">Age:</label><br>
  <input type="number" id="age" name="age" value="30"><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();
        String name = request.getParameter("name");
        String age = request.getParameter("age");
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

取得請求參數

Java Web

Servlet Model

- Response
  - header與body組成，中間隔一個空白行
  - header
    - response line
      - HTTP version
      - HTTP status code
      - HTTP status message
    - header field
  - body
    - XML
    - JSON
    - binary(base64)
    - HTML

Servlet Model

Response

- HTTP status code
  - 1xx(資訊回應)
  - 2xx(成功)
  - 3xx(重導向)
  - 4xx(客戶端錯誤)
  - 5xx(伺服器錯誤)

Servlet Model

Response

- HTTP status code
  - 1xx(資訊回應)
    - 100 Continue：客戶端應當繼續請求
  - 2xx(成功)
    - 200 OK：請求已成功
    - 201 Created：新的資源已經建立
    - 204 No Content：回應的訊息主體內並無實體
    - 206 Partial Content：客戶端進行範圍請求，伺服器成功執行了部分的 GET 請求
  - 3xx(重導向)
    - 302 Found：請求的資源已經移動到新的位置

Servlet Model

Response

- HTTP status code
  - 4xx(客戶端錯誤)
    - 400 Bad Request：請求語法錯誤
    - 401 Unauthorized：未驗證
    - 403 Forbidden：未授權
    - 404 Not Found：找不到請求的資源
    - 405 Method Not Allowed：伺服器不支援請求的HTTP方法
    - 415 Unsupported Media Type：伺服器不支援請求的內容類型
  - 5xx(伺服器錯誤)
    - 500 Internal Server Error：伺服器在處理請求時遇到了意外的情況
    - 502 Bad Gateway：伺服器作為閘道器或代理，從上游伺服器接收到無效的響應
    - 503 Service Unavailable：伺服器目前無法處理請求

Servlet Model

Response

- header field
  - Content-Disposition：指示資源如何處理，例如提示瀏覽器以附件形式下載資源
  - Content-Length：回應主體的長度，以byte為單位
  - Content-Type：回應主體的媒體類型

Servlet Model

Response

- header field
  - Set-Cookie：設定Cookie
    - 定義使用範圍
      - Domain：可以取得cookie的網域
      - Path：可以取得cookie的路徑
    - 定義刪除時間
      - Expires：定義過期時間
      - Max-Age：定義最大秒數，較Expires優先
    - 定義安全性
      - Secure：只能在HTTPS中取得cookie
      - HttpOnly：只能在HTTP、HTTPS取得cookie，JS無法取得

```
Set-Cookie: name=John Doe; Expires=Wed, 21 Oct 2023 07:28:00 GMT; Secure; HttpOnly
Set-Cookie: age=30; Max-Age=86400; Domain=example.com; Secure; HttpOnly
Set-Cookie: city=New York; Max-Age=86400; Path=/; Secure; HttpOnly
```

Servlet Model

Response

- HTML(HyperText Markup Language)
  - 定義網頁上有哪些元件
- CSS(Cascading Style Sheets)
  - 定義各個元件的外觀長像及位置
- JS(JavaScript)
  - 定義各個元件的行為

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>綜合範例</title>
    <style>
        #myParagraph {
            color: blue;
            font-size: 20px;
        }

        .highlight {
            color: red;
        }
    </style>
</head>
<body>

<p id="myParagraph">這是一個段落。</p>
<button onclick="changeText()">點擊我改變段落文字和顏色</button>

<script>
    function changeText() {
        const paragraph = document.getElementById('myParagraph');
        paragraph.textContent = "段落的文字已被改變!";
        paragraph.classList.add('highlight');
    }
</script>

</body>
</html>
```

Servlet Model

Response

- <!DOCTYPE html>
  - 宣告HTML5
- <html>
  - HTML根標籤
- <head>
  - 給瀏覽器看的內容
- <body>
  - 給人看的內容

Servlet Model

Response

- <head>
  - <title>
    - 網頁標題
- <body>
  - <form>
    - 表單資料
  - <h1>~<h6>
    - 字體大小，h1最大
  - <p>
    - 段落
- <a>
  - 超鏈結
- <img>
  - 圖片
- , <ol>,
  - 無序、有序列表
- <table>, <thead>, <tbody>, <tr>, <td>
  - 建立表格
- <input>
  - 輸入資料

Servlet Model

Response

- <form>
  - action：目標url
  - method：GET, POST
    - PUT, DELETE瀏覽器可以不支援
  - enctype
    - application/x-www-form-urlencoded
    - multipart/form-data

Servlet Model

Response

- <input>
  - name：參數名稱
  - value：參數值
  - type：標籤類型
    - text, password, hidden
    - radio, checkbox
    - file
    - button, reset, submit
- <textarea>
- <select>
  - <option>

Servlet Model

Response

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>表單範例</title>
</head>
<body>

<form action="/submit" method="post" enctype="multipart/form-data">
    <label for="username">使用者名稱:</label>
    <input type="text" id="username" name="username"><br><br>
    <label for="pwd">密碼:</label>
    <input type="password" id="pwd" name="pwd"><br><br>
    <input type="hidden" name="hiddenField" value="hiddenValue"><br>

    <label>性別:</label>
    <input type="radio" id="male" name="gender" value="male">
    <label for="male">男</label>
    <input type="radio" id="female" name="gender" value="female">
    <label for="female">女</label><br><br>

    <input type="checkbox" id="subscribe" name="subscribe" value="yes">
    <label for="subscribe">訂閱電子報</label><br><br>
```

```
    <label for="file">選擇檔案:</label>
    <input type="file" id="file" name="file"><br><br>

    <label for="comments">評論:</label><br>
    <textarea id="comments" name="comments" rows="4" cols="50"></textarea><br><br>

    <label for="cars">選擇一輛車:</label>
    <select name="cars" id="cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="mercedes">Mercedes</option>
        <option value="audi">Audi</option>
    </select><br><br>

    <input type="button" value="點擊我"><br><br>
    <input type="reset" value="重設"><br><br>
    <input type="submit" value="提交">
</form>

</body>
</html>
```

Servlet Model

Response

- HttpServletResponse
  - void setContentType(String type)
  - PrintWriter getWriter()

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

回應內容類型

回應內容

Java Web

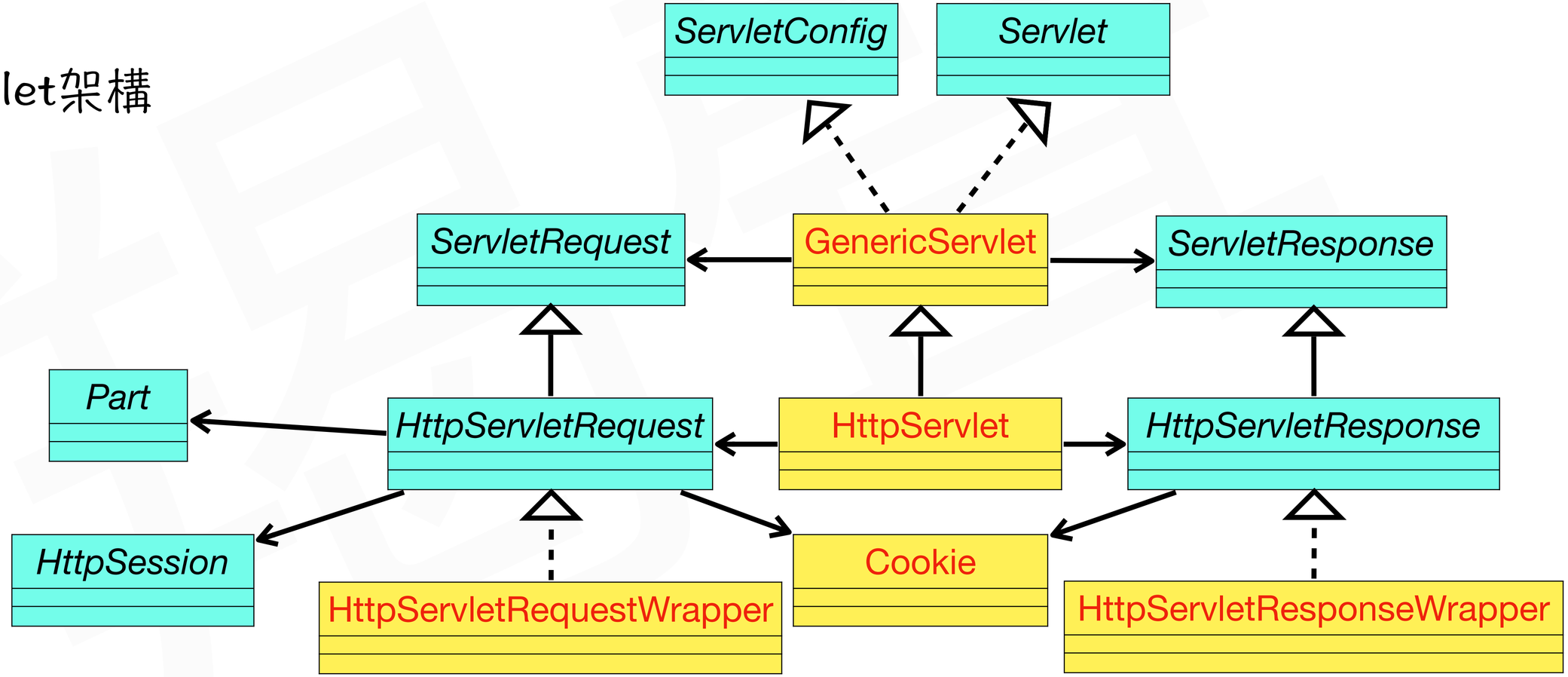
Servlet Model

- Servlet
  - Servlet架構
  - Servlet Lifecycle

Servlet Model

Servlet

- Servlet架構



Servlet Model

Servlet

- Servlet Lifecycle
  - Servlet類別載入
  - Servlet實例化
  - void init()
  - void service(ServletRequest, ServletResponse)
  - void destroy()

Servlet Model

Servlet

- void init()
  - Servlet實例化後呼叫
  - public void init() throws ServletException
  - 若執行失敗則表示servlet實例化失敗，容器將嚐試重新實例化
  - 發生ServletException或UnavailableException則停止實例化
    - UnavailableException is a ServletException
    - GenericServlet實作Servlet，init(ServletConfig)呼叫init()

Servlet Model

Servlet

- Servlet
  - public void service(ServletRequest, ServletResponse) throws ServletException, IOException
    - 輸入request，輸出response
  - 容器接收到request，會從thread pool中取出thread來執行service
    - request及response為區域變數，本身就是thread-safe

Servlet Model

Servlet

- GenericServlet
  - 抽象類別
  - ServletContext getServletContext()
  - ServletConfig getServletConfig()
  - String getInitParameter(String name)
  - Enumeration<String> getInitParameterNames()
  - void init()
  - void init(ServletConfig)
  - void service(ServletRequest req, ServletResponse resp)

Servlet Model

Servlet

- HttpServlet
  - 抽象類別
  - void service(ServletRequest req, ServletResponse resp)
    - void service(HttpServletRequest req, HttpServletResponse resp)
      - void doGet(HttpServletRequest req, HttpServletResponse resp)
      - void doPost(HttpServletRequest req, HttpServletResponse resp)
      - void doPut(HttpServletRequest req, HttpServletResponse resp)
      - void doDelete(HttpServletRequest req, HttpServletResponse resp)
      - void doHead(HttpServletRequest req, HttpServletResponse resp)
      - void doOptions(HttpServletRequest req, HttpServletResponse resp)
      - void doTrace(HttpServletRequest req, HttpServletResponse resp)

Servlet Model

Servlet

- void destroy()
  - Servlet撤離時呼叫，用於釋放資源
  - init()失敗時不會呼叫

Java Web

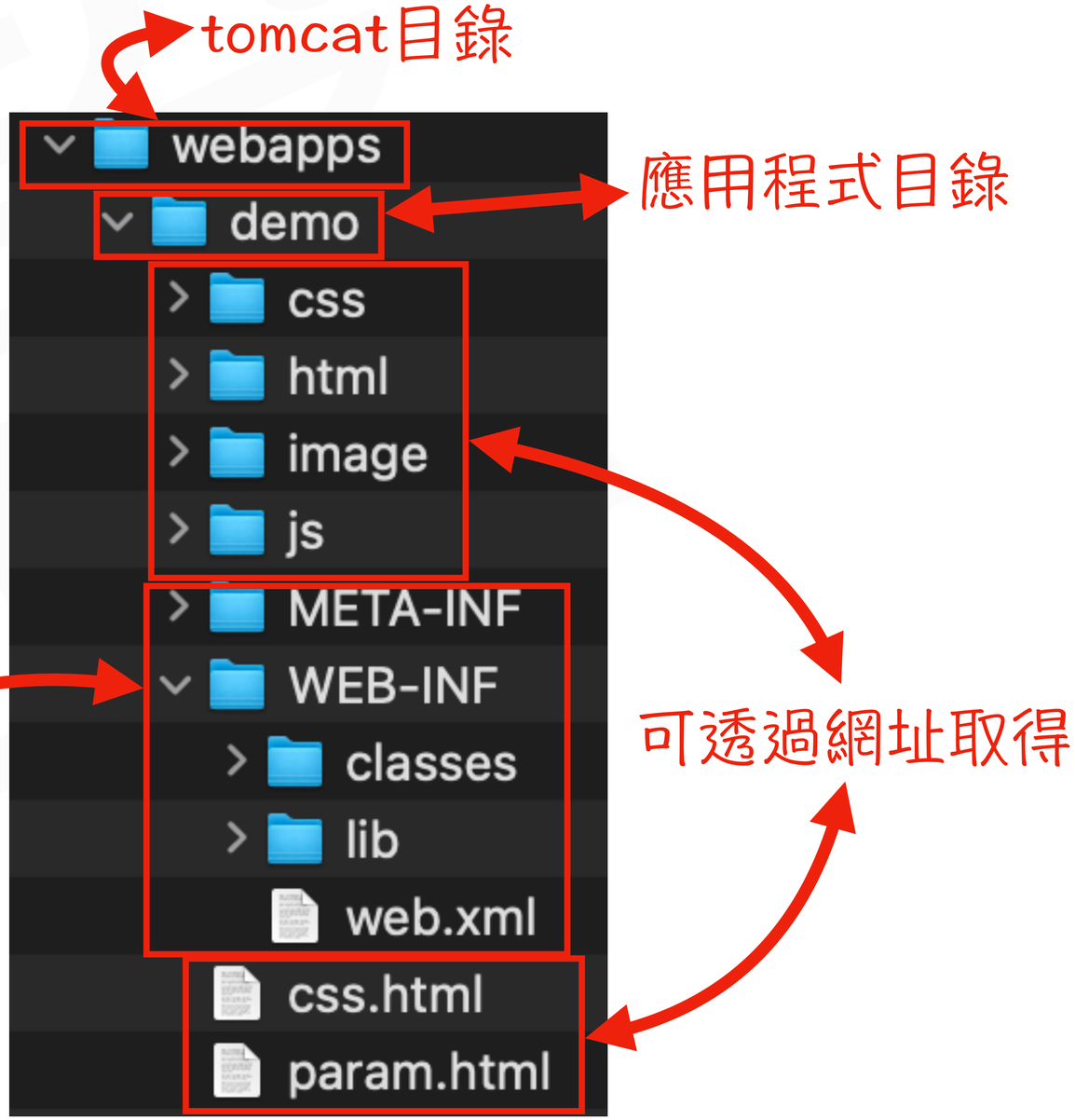
應用與實務

- Web Application
  - 佈署結構
  - 佈署描述檔
  - 打包應用程式

Java Web

Web Application

- 佈署結構
  - MEAT-INF及WEB-INF中的資源無法直接取得
  - classes放置編譯後的class
  - lib放置函式庫的jar
  - web.xml應用程式佈署檔
  - 使用jar命令將此結構打包為demo.war



Java Web

Web Application

- 佈署描述檔(web.xml)
  - 2.3版前，描述順序需符合規則
  - 2.4版後無順序
  - 3.0後可不需此檔案
    - 使用Annotation代替

Java Web

Web Application

- 佈署描述檔(web.xml)

```
<web-app>
  <description>
  <default-context-path>
  <request-character-encoding>
  <response-character-encoding>
  <display-name>
  <icon>
  <distributable>
  <context-param>
  <filter>
  <filter-mapping>
  <listener>
  <servlet>
  <servlet-mapping>
  <session-config>
  <mime-mapping>
  <welcome-file-list>
  <error-page>
  <jsp-config>
  <security-constraint>
  <login-config>
  <security-role>
```

```
<web-app>
  <env-entry>
  <ejb-ref>
  <ejb-local-ref>
  <service-ref>
  <resource-ref>
  <resource-env-ref>
  <message-destination-ref>
  <locale-encoding-mapping-list>
```

Java Web

Web Application

- 佈署描述檔(web.xml)

```
<servlet>
  <description>
  <display-name>
  <icon>
    <small-icon>
    <large-icon>
  <servlet-name>
  <servlet-class>
  <jsp-file>
  <init-param>
    <description>
    <param-name>
    <param-value>
  <load-on-startup>
  <run-as>
    <description>
    <role-name>
  <security-role-ref>
    <description>
    <role-name>
    <role-link>
  <async-supperted>
```

```
<servlet-mapping>
  <servlet-name>
  <url-pattern>
<welcome-file-list>
  <welcome-file>
<error-page>
  <error-code>
  <location>
<mime-mapping>
  <extension>
  <mime-type>
```

Java Web

Web Application

- 佈署描述檔(web.xml)
  - servlet-class輸入class的完整名稱
  - init-param初始參數
  - load-on-startup開機載入順序
    - 1為最小值，值越小越優先
  - url-pattern為對應的網址

servlet設定

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <request-character-encoding>UTF-8</request-character-encoding>

    <servlet>
        <servlet-name>Parameter</servlet-name>
        <servlet-class>edu.javaweb.ParameterServlet</servlet-class>
        <init-param>
            <param-name>exampleParam</param-name>
            <param-value>Example Value</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Parameter</servlet-name>
        <url-pattern>/parameters</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <error-page>
        <error-code>404</error-code>
        <location>/html/404.jsp</location>
    </error-page>
</web-app>
```

開機載入順序

Java Web

Web Application

- 佈署描述檔(web.xml)
  - servlet-class輸入class的完整名稱
  - init-param初始參數
  - load-on-startup開機載入順序
    - 1為最小值，值越小越優先
  - url-pattern為對應的網址

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <request-character-encoding>UTF-8</request-character-encoding>

    <servlet>
        <servlet-name>Parameter</servlet-name>
        <servlet-class>edu.javaweb.ParameterServlet</servlet-class>
        <init-param>
            <param-name>exampleParam</param-name>
            <param-value>Example Value</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Parameter</servlet-name>
        <url-pattern>/parameters</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <error-page>
        <error-code>404</error-code>
        <location>/html/404.jsp</location>
    </error-page>
</web-app>
```

完整class名稱

初始參數

對應網址

Java Web

Web Application

- 佈署描述檔(web.xml)
  - welcome-file-list網址為目錄時的預設檔案
  - error-page錯誤碼對應檔案

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <request-character-encoding>UTF-8</request-character-encoding>

    <servlet>
        <servlet-name>Parameter</servlet-name>
        <servlet-class>edu.javaweb.ParameterServlet</servlet-class>
        <init-param>
            <param-name>exampleParam</param-name>
            <param-value>Example Value</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Parameter</servlet-name>
        <url-pattern>/parameters</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <error-page>
        <error-code>404</error-code>
        <location>/html/404.jsp</location>
    </error-page>
</web-app>
```

目錄預設檔案設定

錯誤頁面設定

Java Web

Web Application

- 佈署描述檔(web.xml)

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

public class ParameterServlet extends HttpServlet {
    public void init(){
        System.out.println("load servlet: ParameterServlet");
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");

        String exampleParam = getServletConfig().getInitParameter("exampleParam");

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
        out.println("<p>Example Parameter: " + exampleParam + "</p>");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

取得初始參數

Java Web

佈署描述檔

- 使用Annotation佈署
  - 方便設定
  - 依賴性提高

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet(
    urlPatterns = "/parameters",
    loadOnStartup = 1,
    initParams={
        @WebInitParam(name="exampleParam", value="Example Value")
    }
)
public class ParameterServlet extends HttpServlet {
    public void init(){
        System.out.println("load servlet: ParameterServlet");
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");

        String exampleParam = getServletConfig().getInitParameter("exampleParam");

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
        out.println("<p>Example Parameter: " + exampleParam + "</p>");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

Annotation佈署

Java Web

佈署描述檔

- url patterns
  - 完全路徑
    - /path
  - 最長路徑
    - /path/*
  - 延伸路徑
    - *.jsp
  - 預設路徑
    - /

Java Web

佈署描述檔

- requestURI
  - requestURI = contextPath + servletPath + pathInfo
  - requestURI: request.getRequestURI()
  - contextPath: request.getContextPath()
    - 應用程式名稱
  - servletPath: request.getServletPath()
    - 取得完整、最長(除 * 部份)、延伸、預設路徑
  - pathInfo: request.getPathInfo()
    - 取得最長路徑的 * 部份

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/path/*")
public class PathServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        System.out.println(request.getRequestURI());
        System.out.println(request.getContextPath());
        System.out.println(request.getServletPath());
        System.out.println(request.getPathInfo());

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
    }
}
```

取得各種路徑

Java Web

佈署描述檔

- ServletConfig
  - Servlet初始化時會將Servlet的設定(標註或web.xml)封裝成ServletConfig
  - String getInitParameter(String name)
  - Enumeration<String> getInitParameterNames()
  - ServletContext getServletContext()
  - String getServletName()

使用config

與Servlet的無差異

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet(
    urlPatterns = "/parameters",
    loadOnStartup = 1,
    initParams={
        @WebInitParam(name="exampleParam", value="Example Value")
    }
)
public class ParameterServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");

        var config = getServletConfig();
        String exampleParam1 = config.getInitParameter("exampleParam");
        String exampleParam2 = getInitParameter("exampleParam");
        String servletName = config.getServletName();
        var context = config.getServletContext();

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
        out.println("<p>Example Parameter1: " + exampleParam1 + "</p>");
        out.println("<p>Example Parameter2: " + exampleParam2 + "</p>");
        out.println("<p>Servlet Name: " + servletName + "</p>");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

Java Web

Web Application

- 打包應用程式
  - WAR
  - Web Fragment

Java Web

打包應用程式

- WAR(Web Application Archive)
  - 把所有檔案打包成1個方便佈署
  - 進到應用程式目錄中
    - cd webapps/<app_name>
  - 使用jar指令打包
    - jar -cf <app_name>.war *

Java Web

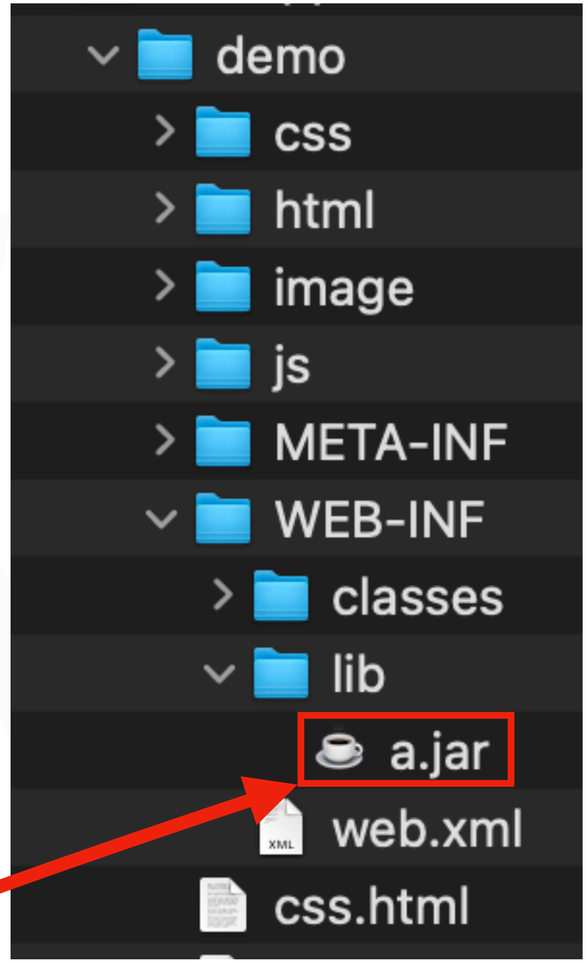
打包應用程式

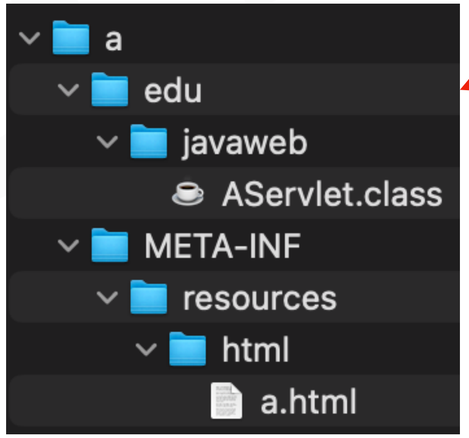
- Web Fragment
  - 模組化web設計
  - 將web相關檔案打包成jar放置在/WEB-INF/lib
  - 在web.xml中，web-app的metadata-complete屬性值需為false
    - 能否使用annotation佈署也與metadata-complete有關

Java Web

打包應用程式

- Web Fragment
  - 資源檔放置jar檔中的/META-INF/resources
  - Servlet可用Annotation佈署
  - 佈署描述檔為/META-INF/web-fragment.xml

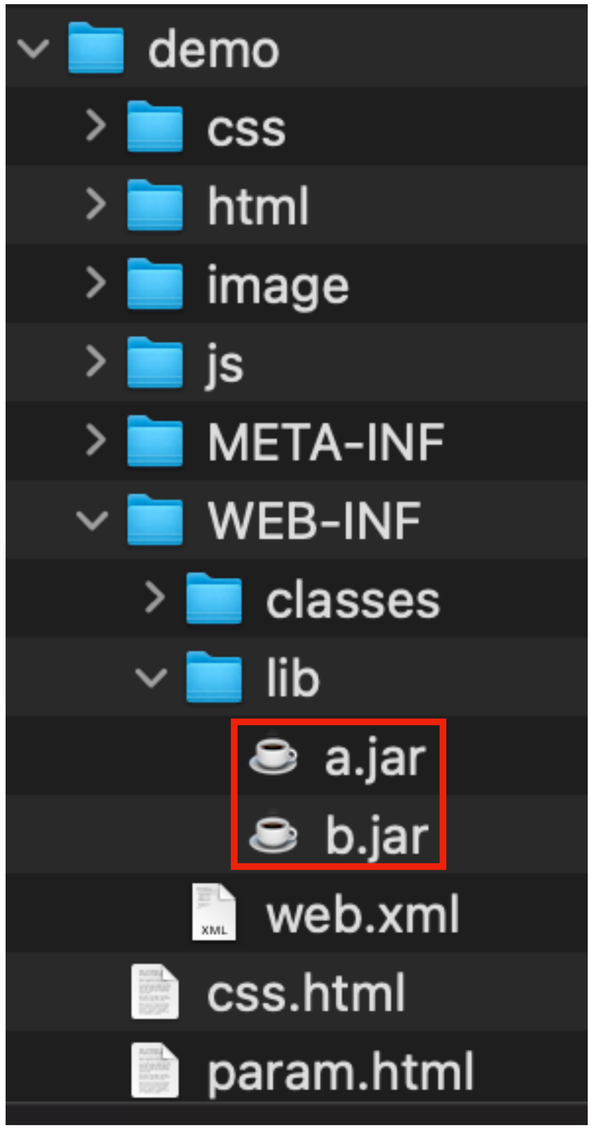




Java Web

打包應用程式

- Web Fragment
  - 若有多個，依JVM載入順序佈署
  - 若需有順序
    - 絕對順序
      - web.xml中定義
    - 相對順序
      - web-fragment.xml中定義



Java Web

打包應用程式

- 絕對順序
  - 使用<absolute-ordering>

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <request-character-encoding>UTF-8</request-character-encoding>

    <absolute-ordering>
      <name>A</name>
      <name>B</name>
    </absolute-ordering>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <error-page>
        <error-code>404</error-code>
        <location>/html/404.jsp</location>
    </error-page>
</web-app>
```

Java Web

打包應用程式

- 相對順序
  - <ordering>
  - <after>
    - 在B之後
  - <before>
    - 在其他之前
  - <others>

```
<?xml version="1.0" encoding="UTF-8"?>
<web-fragment xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-fragment_4_0.xsd"
    version="4.0">
    <name>A</name>
    <ordering>
        <after>
            <name>B</name>
        </after>
    </ordering>
</web-fragment>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<web-fragment xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-fragment_4_0.xsd"
    version="4.0">
    <name>B</name>
    <ordering>
        <before><others/></before>
    </ordering>
</web-fragment>
```

Java Web

應用與實務

- Web Container
  - ServletContext
  - HttpServletRequest, HttpServletResponse
  - AsyncContext
  - Filter, Wrapper
  - WebSocket

Java Web

Web Container

- ServletContext
  - 代表整個web應用程式，所有的Servlet共用
  - 取得方式
    - Servlet中的 ServletContext getServletContext()
    - ServletConfig中的 ServletContext getServletContext()

Web Container

ServletContext

- 取得應用程式的初始化參數

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <request-character-encoding>UTF-8</request-character-encoding>
    <context-param>
        <param-name>websiteName</param-name>
        <param-value>JavaWebExample</param-value>
    </context-param>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <error-page>
        <error-code>404</error-code>
        <location>/html/404.jsp</location>
    </error-page>
</web-app>
```

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet(
    urlPatterns = "/parameters",
    loadOnStartup = 1,
    initParams={
        @WebInitParam(name="exampleParam", value="Example Value")
    }
)
public class ParameterServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");

        var config = getServletConfig();
        String exampleParam1 = config.getInitParameter("exampleParam");
        String exampleParam2 = getInitParameter("exampleParam");
        String servletName = config.getServletName();
        var context = config.getServletContext();
        var websiteName = context.getInitParameter("websiteName");

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
        out.println("<p>Example Parameter1: " + exampleParam1 + "</p>");
        out.println("<p>Servlet Name: " + servletName + "</p>");
        out.println("<p>Website Name: " + websiteName + "</p>");
    }
}
```

Web Container

ServletContext

- 屬性(Attribute)
  - 可以變的參數(Parameter)
  - 值不限於字串
  - 四個範圍
    - ServletContext
    - Session
    - Request
    - Page
- 方法
  - Object getAttribute(String name)
  - Enumeration<String> getAttributeNames()
  - void setAttribute(String name, Object obj)
  - void removeAttribute(String name)

Web Container

ServletContext

- ServletContext Attribute
  - 所有Servlet共用

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet(
    urlPatterns="/a"
)
public class AServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        String name = request.getParameter("name");
        String age = request.getParameter("age");
        var application = getServletContext();
        application.setAttribute("appname", "JavaWeb");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello A!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
    }
}
```

取得屬性

刪除屬性

設定屬性

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet(
    urlPatterns="/b"
)
public class BServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        String name = request.getParameter("name");
        String age = request.getParameter("age");
        var application = getServletContext();
        var appname = application.getAttribute("appname");
        application.removeAttribute("appname");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello B!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
        out.println("<p>appname: " + appname + "</p>");
    }
}
```

Web Container

ServletContext

- 取得資源
  - String getRealPath(String path)
  - URL getResource(String path)
  - InputStream getResourceAsStream(String path)

真實檔案路徑

URL格式路徑

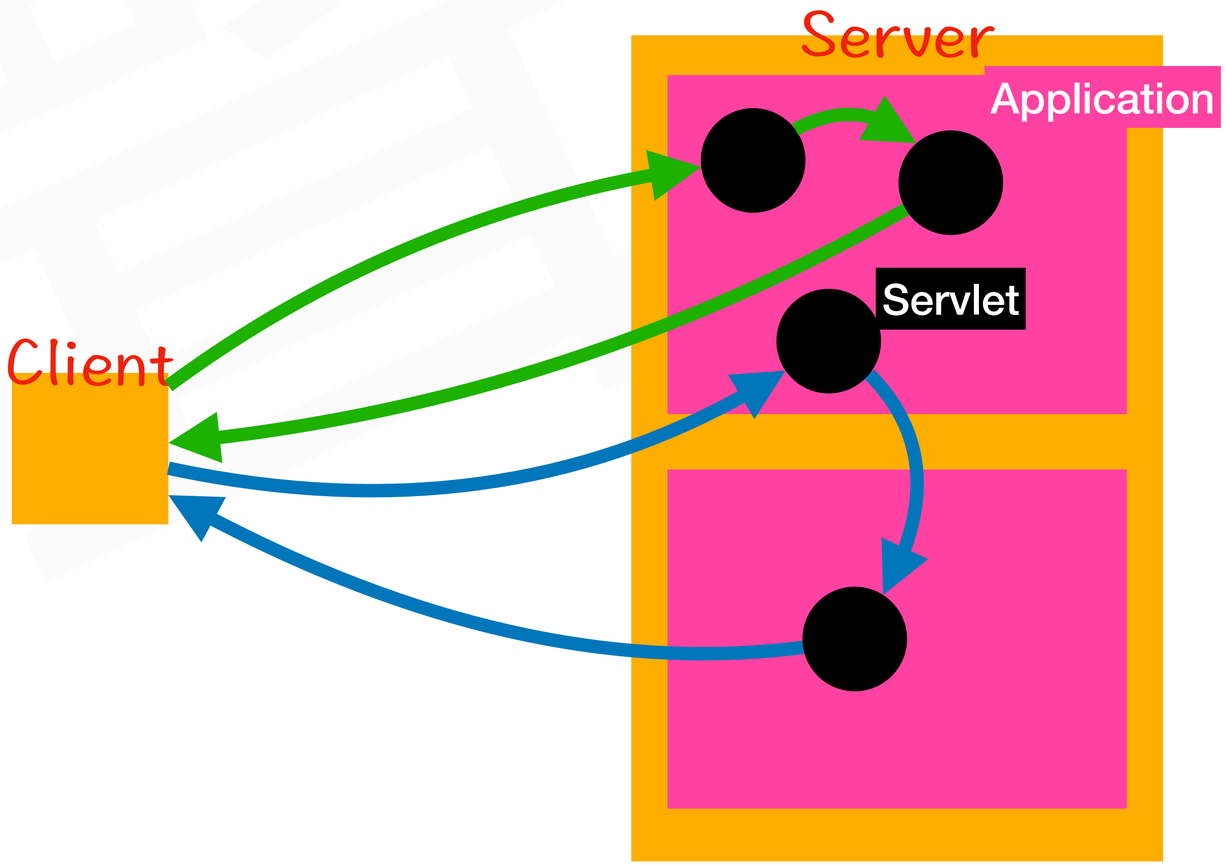
取得檔案內容

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/path")
public class PathServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/plain");
        var application = getServletContext();
        var realPath = application.getRealPath("/WEB-INF/web.xml");
        System.out.println(realPath);

        var uri = application.getResource("/WEB-INF/web.xml");
        System.out.println(uri);

        var resource = application.getResourceAsStream("/WEB-INF/web.xml");

        PrintWriter out = response.getWriter();
        BufferedReader reader = new BufferedReader(
                new InputStreamReader(resource));
        String line;
        while ((line = reader.readLine()) != null) {
            out.println(line);
        }
        reader.close();
    }
}
```

Web Container

ServletContext

- 需求轉發
  - 接收request和送出response是不同Servlet
    - 同一個應用程式
    - 不同一個應用程式
      - 必需容器支援
  - 轉發方式
    - forward的response是新的
    - include的response是原本的



Web Container

ServletContext

- 需求轉發時，容器會保留資料在Request Attribute

forward	include
jakarta.servlet.forward.request_uri	jakarta.servlet.include.request_uri
jakarta.servlet.forward.context_path	jakarta.servlet.include.context_path
jakarta.servlet.forward.servlet_path	jakarta.servlet.include.servlet_path
jakarta.servlet.forward.path_info	jakarta.servlet.include.path_info
jakarta.servlet.forward.query_string	jakarta.servlet.include.query_string

Web Container

ServletContext

- forward的時候
  - 需求轉發前不可有任何的response
  - 若尚未確認前，在轉發後，先前的response將被清空
  - 若已確認則丟出IllegalStateException

Web Container

ServletContext

- 需求轉發
  - RequestDispatcher getRequestDispatcher (String path)

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet(
    urlPatterns="/a"
)
public class AServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        request.setAttribute("message", "Hello from AServlet!");
        var application = getServletContext();
        RequestDispatcher dispatcher = application.getRequestDispatcher("/b");
        dispatcher.forward(request, response);
    }
}
```

進行需求轉發

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet(
    urlPatterns="/b"
)
public class BServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        var out = response.getWriter();
        String message = (String) request.getAttribute("message");
        out.println("<h1>" + message + "</h1>");
        out.println("<p>This response is generated by BServlet.</p>");
    }
}
```

Java Web

Web Container

- HttpServletRequest
  - 處理請求參數與標頭
  - 請求參數編碼
  - 處理請求本體
  - RequestDispatcher容器保留屬性

Web Container

HttpServletRequest

- 處理請求標頭
  - String getHeader(String name)
  - Enumeration<String> getHeaders(String name)
  - Enumeration<String> getHeaderNames()
  - long getDateHeader(String name)
  - int getIntHeader(String name)

取得標頭

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        var userAgent = request.getHeader("User-Agent");

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
        out.println("<p>User-Agent: " + userAgent + "</p>");
    }
}
```

Web Container

HttpServletRequest

- 處理Cookie
  - Cookie[] getCookies()
    - String getName()
    - String getValue()
    - 其他內容在request中沒有

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        var cookies = request.getCookies();
        for(var c: cookies){
            System.out.println(c.getName());
            System.out.println(c.getValue());
        }

        String name = request.getParameter("name");
        String age = request.getParameter("age");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
    }
}
```

取得cookie及內容

Web Container

HttpServletRequest

- 處理請求參數
  - String getParameter(String name)
  - String[] getParameterValues(String name)
  - Map<String, String[]> getParameterMap()
  - Enumeration<String> getParameterNames()

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        String name = request.getParameter("name");
        String age = request.getParameter("age");
        request.getParameterMap()
            .entrySet()
            .stream()
            .map(e->e.getKey()+"= "
                +Arrays.toString(e.getValue()))
            .forEach(System.out::println);
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
    }
}
```

取得請求參數

Web Container

HttpServletRequest

- 處理請求路徑
  - HttpServletMapping getHttpServletMapping()
    - String getMatchValue()
    - String getPattern()
    - String getServletName()
    - MappingMatch getMappingMatch()
      - CONTEXT_ROOT
      - DEFAULT
      - EXACT
      - EXTENSION
      - PATH

URI Path (in quotes)	matchValue	pattern	mappingMatch
""	""	""	CONTEXT_ROOT
"/index.html"	""	/	DEFAULT
"/MyServlet"	MyServlet	/MyServlet	EXACT
"/foo.extension"	foo	*.extension	EXTENSION
"/path/foo"	foo	/path/*	PATH

Web Container

HttpServletRequest

- 處理請求路徑
  - /path/app

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/path/*")
public class PathServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        var mapping = request.getHttpServletMapping();
        System.out.println(mapping.getMatchValue());
        System.out.println(mapping.getPattern());
        System.out.println(mapping.getServletName());
        System.out.println(mapping.getMappingMatch());

        var out = response.getWriter();
        out.println("ok");
        out.close();
    }
}
```

取得HttpServletMapping

app

/path/*

PATH

edu.javaweb.PathServlet

Web Container

HttpServletRequest

- 請求參數編碼
  - POST
    - request.setCharacterEncoding("UTF-8")
    - 需於取得請求參數前呼叫
  - GET
    - String name = new String(request.getParameter("name").getBytes("ISO-8859-1"), "UTF-8")
      - ISO-8859-1為原始編碼
      - UTF-8為目標編碼
  - web.xml
    - <request-character-encoding>UTF-8</request-character-encoding>

Web Container

HttpServletRequest

- 請求參數編碼

所有處理前呼叫

Servlet處理需以html的編碼為主

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>

<h2>HTML Form</h2>

<form action="/demo/parameters"
      enctype="application/x-www-form-urlencoded" method="POST">
  <label for="name">Name:</label><br>
  <input type="text" id="name" name="name" value="John Doe"><br>
  <label for="age">Age:</label><br>
  <input type="number" id="age" name="age" value="30"><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html");
        String name = request.getParameter("name");
        String age = request.getParameter("age");
        request.getParameterMap()
            .entrySet()
            .stream()
            .map(e->e.getKey()+"= "
                +Arrays.toString(e.getValue()))
            .forEach(System.out::println);
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
        out.println("<p>Name: " + name + "</p>");
        out.println("<p>Age: " + age + "</p>");
    }
    public void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

Web Container

HttpServletRequest

- 處理請求本體(InputStream)
  - getReader(), getInputStream()可取得請求本體
  - 若HTML的發送表單中含有檔案，可做為檔案上傳(form的enctype屬性需為multipart/form-data)
  - 所有參數須自行解析及處理

Web Container

HttpServletRequest

- 處理請求本體(InputStream)

multipart/form-data

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>

<h2>HTML Form</h2>

<form action="/demo/parameters"
      enctype="multipart/form-data" method="POST">
  <label for="photo">Photo:</label><br>
  <input type="file" name="file"><br><br>
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        saveRequestBodyToFile(request,"/tmp/a.png");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
    }
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
    private void saveRequestBodyToFile(HttpServletRequest request,
                String filePath) throws IOException {
        try (FileOutputStream out = new FileOutputStream(filePath);
             InputStream in = new BufferedInputStream(request.getInputStream())) {
            byte[] buffer = new byte[1024];
            int length;
            while ((length = in.read(buffer)) != -1) {
                out.write(buffer, 0, length);
            }
        }
    }
}
```

取得請求本體

Web Container

HttpServletRequest

- 處理請求本體(Part)
  - getPart(), getParts()可取得上傳檔案
  - 處理上傳檔案的Servlet需有@MultipartConfig的標註或於web.xml中設定<multipart-config>
  - @MultipartConfig上的屬性
    - location：上傳檔案暫存位置，預設值為ServletContextAttribute中的javax.servlet.context.tempdir
    - fileSizeThreshold：上傳檔案大小超過此byte數，則寫入暫存位置，否則放在記憶體中(型態long)
    - maxFileSize：單檔最大byte數(型態long)
    - maxRequestSize：request最大byte數(型態long)

Web Container

HttpServletRequest

- 處理請求本體(Part)
  - String getContentType()
  - String getHeader(String name)
  - Collection<String> getHeaderNames()
  - Collection<String> getHeaders(String name)
  - InputStream getInputStream()
  - String getName()
  - long getSize()
  - String getSubmittedFileName()
  - void write(String fileName)

取得Part

將檔案存到硬碟

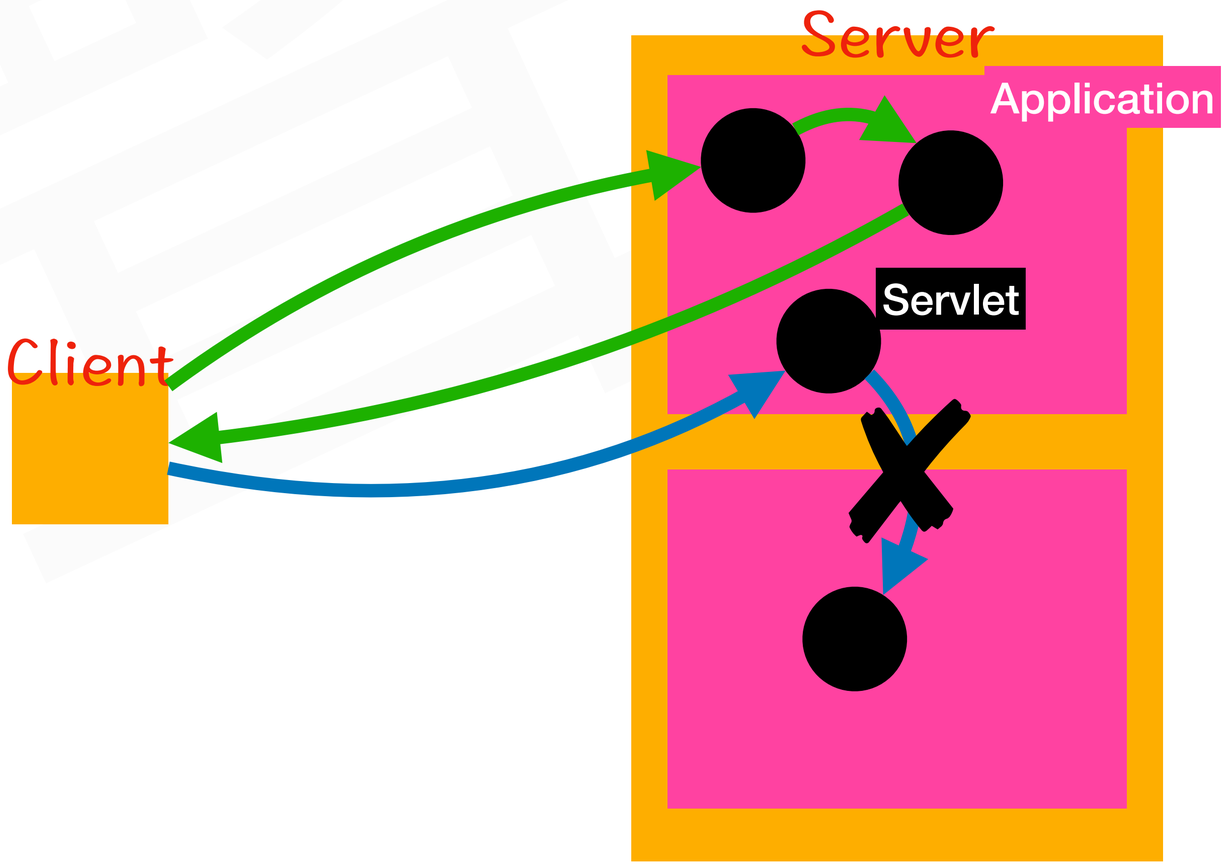
透過Part的header取得檔名

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet("/parameters")
@MultipartConfig
public class ParameterServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        for (Part part : request.getParts()) {
            String fileName = extractFileName(part);
            part.write("/tmp/" + fileName);

        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
    }
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);
    }
    private String extractFileName(Part part) {
        String contentDisp = part.getHeader("content-disposition");
        String[] items = contentDisp.split(";");
        for (String s : items) {
            if (s.trim().startsWith("filename")) {
                return s.substring(s.indexOf("=") + 2, s.length() - 1);
            }
        }
        return "";
    }
}
```

Web Container

HttpServletRequest

- RequestDispatcher容器保留屬性
  - 必需同一個應用程式
  - 其他特性參考ServletContext



Web Container

HttpServletRequest

- 需求轉發
  - RequestDispatcher getRequestDispatcher (String path)

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet(
    urlPatterns="/a"
)
public class AServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        request.setAttribute("message", "Hello from AServlet!");
        RequestDispatcher dispatcher = request.getRequestDispatcher("/b");
        dispatcher.forward(request, response);
    }
}
```

進行需求轉發

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
@WebServlet(
    urlPatterns="/b"
)
public class BServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        var out = response.getWriter();
        String message = (String) request.getAttribute("message");
        out.println("<h1>" + message + "</h1>");
        out.println("<p>This response is generated by BServlet.</p>");
    }
}
```

Java Web

Web Container

- HttpServletResponse
  - 處理標頭及緩衝區
  - 處理本體
  - 需求轉向

Web Container

HttpServletResponse

- 處理標頭
  - 設值
    - void addDateHeader(String name, long date)
    - void addHeader(String name, String value)
    - void addIntHeader(String name, int value)
    - void setDateHeader(String name, long date)
    - void setHeader(String name, String value)
    - void setIntHeader(String name, int value)
    - void setStatus(int sc)
    - void setCharacterEncoding(String charset)
    - void setContentLength(int len)
    - void setContentType(String type)
    - void setLocale(Locale loc)

Web Container

HttpServletResponse

- 處理標頭
  - 取值
    - boolean containsHeader(String name)
    - String getHeader(String name)
    - Collection<String> getHeaderNames()
    - Collection<String> getHeaders(String name)
    - int getStatus()
    - String getCharacterEncoding()
    - String getContentType()
    - Locale getLocale()

Web Container

HttpServletResponse

- 處理標頭

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doPost(HttpServletRequet request,
            HttpServletResponse response)
            throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html");
        response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate");
        response.setHeader("Pragma", "no-cache");
        response.setDateHeader("Expires", 0);
        response.setHeader("X-Custom-Header", "CustomValue");
        PrintWriter out = response.getWriter();
        out.println("<h1>" + "Hello World!" + "</h1>");
    }
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        doPost(request, response);
    }
}
```

設定回應編碼
需在所有回應之前

設定回應標頭

Web Container

HttpServletResponse

- 處理標頭
  - 在web.xml中設定回應編碼
    - <response-character-encoding>UTF-8</response-character-encoding>

Web Container

HttpServletResponse

- 處理緩衝區
  - void setBufferSize(int size)：設定回應緩衝區
  - int getBufferSize()：取得回應緩衝區大小
  - void flushBuffer()：強制將緩衝區資料回應給client
  - boolean isCommitted()：檢查回應是否已確認
    - 回應被確認(呼叫flushBuffer()或是緩衝區已滿)，回應被確認後就不能再修改標頭及狀態碼
    - 已確認表示回應已輸出至client，除非有必要否則不需手動commit，容器會自動處理
  - reset()：重置所有response內容
  - resetBuffer()：重置緩衝區，狀態碼及標頭不會重置

Web Container

HttpServletResponse

- 處理緩衝區

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setBufferSize(8192);
        response.getWriter().write("This is some initial content. ");
        if (!response.isCommitted()) {
            response.getWriter().write("More content can be written. ");
        }
        response.flushBuffer();
        response.getWriter().write("This content is written after flushing. ");
    }
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        doPost(request, response);
    }
}
```

設定緩衝區
需在所有回應之前
否則丟出IllegalStateException

操作緩衝區
無特別需要
使用預設值即可

Web Container

HttpServletResponse

- 處理Cookie
  - 設值
    - void setComment(String comment)
    - void setDomain(String domain)
    - void setHttpOnly(boolean isHttpOnly)
    - void setMaxAge(int expiry)
    - void setPath(String path)
    - void setSecure(boolean flag)
    - void setValue(String newValue)
    - void setVersion(int v)

Web Container

HttpServletResponse

- 處理Cookie
  - 取值
    - String getComment()
    - String getDomain()
    - boolean isHttpOnly()
    - int getMaxAge()
    - String getName()
    - String getPath()
    - boolean getSecure()
    - String getValue()
    - int getVersion()

Web Container

HttpServletResponse

- 處理Cookie

設定Cookie

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet("/parameters")
public class ParameterServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        Cookie cookie = new Cookie("username", "john");
        cookie.setMaxAge(60 * 60 * 24);
        response.addCookie(cookie);
    }
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        doPost(request, response);
    }
}
```

Web Container

HttpServletResponse

- 處理本體
  - 輸出字元
    - PrintWriter getWriter()
  - 輸出位元
    - ServletOutputStream getOutputStream()

Web Container

HttpServletResponse

- 處理本體

設定檔案相關標頭

輸出檔案

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/path/download")
public class PathServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        var outStream = response.getOutputStream();
        String filePath = "/etc/passwd";
        File downloadFile = new File(filePath);
        FileInputStream inStream = new FileInputStream(downloadFile);
        String mimeType = getServletContext().getMimeType(filePath);
        if (mimeType == null) {
            mimeType = "application/octet-stream";
        }
        response.setContentType(mimeType);
        response.setContentLength((int) downloadFile.length());
        String headerKey = "Content-Disposition";
        String headerValue = String.format("attachment; filename=\"%s\"", downloadFile.getName());
        response.setHeader(headerKey, headerValue);
        try (outStream;inStream) {
            byte[] buffer = new byte[4096];
            int bytesRead = -1;
            while ((bytesRead = inStream.read(buffer)) != -1) {
                outStream.write(buffer, 0, bytesRead);
            }
        }
    }
}
```

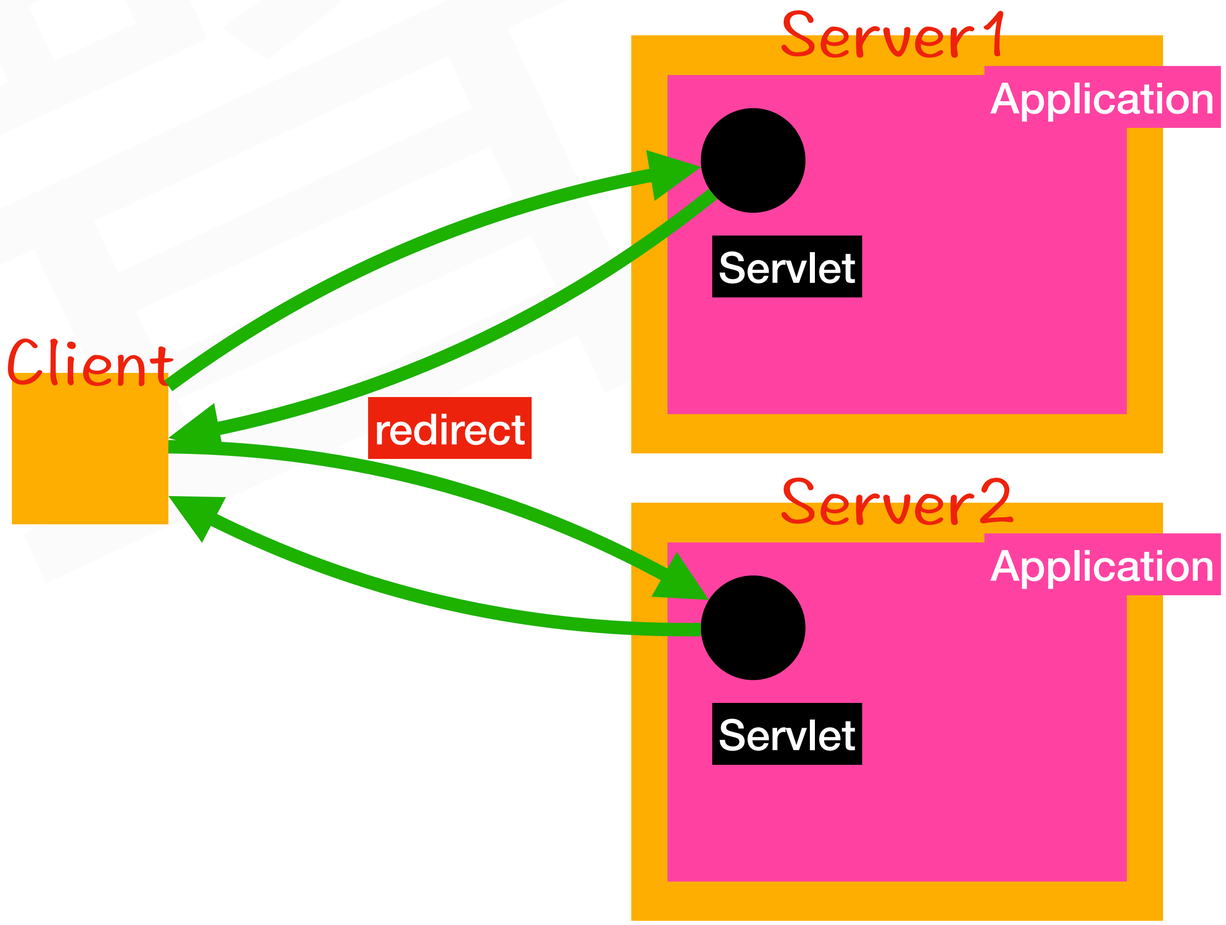
attachment讓瀏覽器直接下載

inline用瀏覽器的方式開啟檔案

Web Container

HttpServletResponse

- 需求轉向
  - 使瀏覽器轉向
    - void sendRedirect(String location)
  - 送出錯誤碼
    - void sendError(int sc)
    - void sendError(int sc, String msg)



Web Container

HttpServletResponse

- 需求轉向

```
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/path/*")
public class PathServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        var mapping = request.getHttpServletMapping();
        var path = mapping.getMatchValue();
        if("google".equals(path)){
            response.sendRedirect("https://www.google.com");
        } else {
            response.sendError(HttpServletResponse.SC_NOT_FOUND, path+" not found.");
        }
    }
}
```

送出302使瀏覽器轉向

送出錯誤碼

Java Web

Web Container

- AsyncContext
  - 非同步的功能
  - 使用非同步
  - 佈署非同步

Web Container

AsyncContext

- 非同步的功能
  - 不等待request完成處理，就將thread還回容器，並使用另一個thread繼續執行，可提高應用程式的吞吐量
    - 另一thread可由容器的另一個thread pool提供或是自行新增thread
    - 接收request和送出response是不同thread
  - 可在新的thread中直接進行response
  - 若新增thread需注意，在高流量下可以產生效能瓶頸

Web Container

AsyncContext

- 使用非同步
  - void start(Runnable run)
  - void complete()
  - ServletRequest getRequest()
  - ServletResponse getResponse()
  - long getTimeout()
  - void setTimeout(long timeout)
  - void dispatch()
  - void dispatch(String path)
  - void addListener(AsyncListener listener)
  - void addListener(AsyncListener listener, ServletRequest req, ServletResponse resp)

Web Container

AsyncContext

- 佈署非同步
  - Servlet及Filter皆可使用
  - Annotation
    - WebServlet加上asyncSupported=true
  - web.xml
    - <servlet>標籤中加上<async-supported>true</async-supported>
  - 未設定會丟出IllegalStateException

Web Container

AsyncContext

- 佈署非同步
  - dispatch()的保留屬性
    - jakarta.servlet.async.request_uri: request.getRequestURI()
    - jakarta.servlet.async.context_path: request.getContextPath()
    - jakarta.servlet.async.servlet_path: request.getServletPath()
    - jakarta.servlet.async.path_info: request.getPathInfo(()
    - jakarta.servlet.async.quest_string: request.getQuery()

Web Container

AsyncContext

- 佈署非同步

```java
new Thread(new Runnable() {
    @Override
    public void run() {
        try {
            Thread.sleep(5000);
            PrintWriter out = asyncContext.getResponse().getWriter();
            out.write("Async response after 5 seconds!");
            out.close();
            asyncContext.complete();
        } catch (InterruptedException | IOException e) {
            e.printStackTrace();
        }
    }
}).start();
```

新增thread

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet(urlPatterns="/parameters",
    asyncSupported=true
)
public class ParameterServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        var asyncContext = request.startAsync();

        asyncContext.start(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep(5000);
                    PrintWriter out = asyncContext.getResponse().getWriter();
                    out.write("Async response after 5 seconds!");
                    out.close();
                    asyncContext.complete();
                } catch (InterruptedException | IOException e) {
                    e.printStackTrace();
                }
            }
        });
    }
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        doPost(request, response);
    }
}
```

取得非同步物件

開始非同步

完成非同步

Java Web

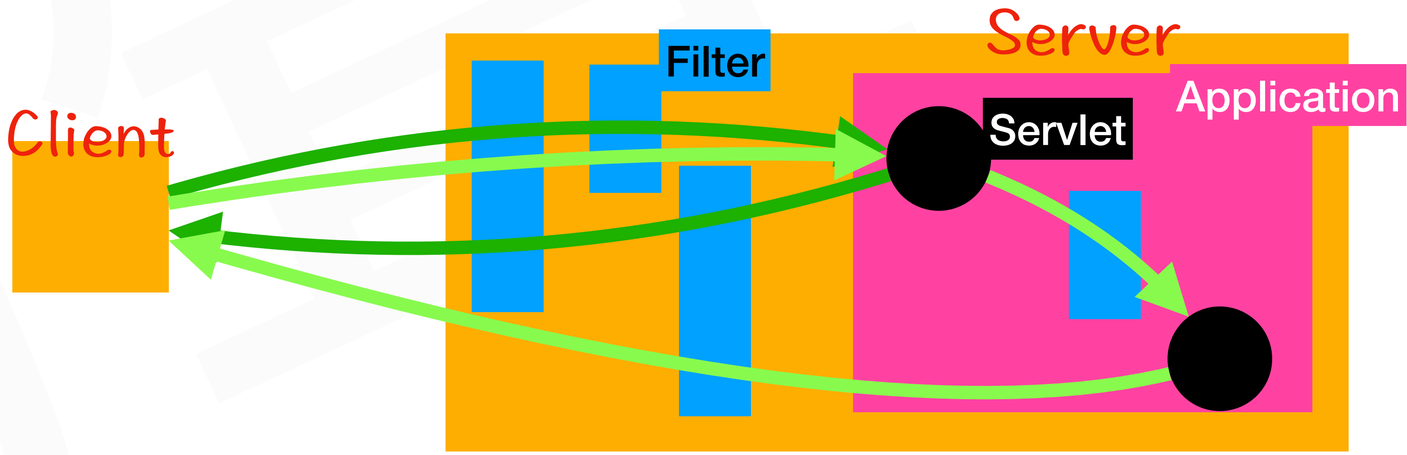
Web Container

- Filter
  - 使用時機
  - 生命週期
  - 架構及佈署

Web Container

Filter

- 使用時機
  - 在Servlet之前跟之後，攔截request及response
  - 不能影響主要邏輯
    - 驗證過濾、授權過濾、圖檔轉換、資料壓縮、操作記錄、加密
  - 使用wrapper改變request及response



Web Container

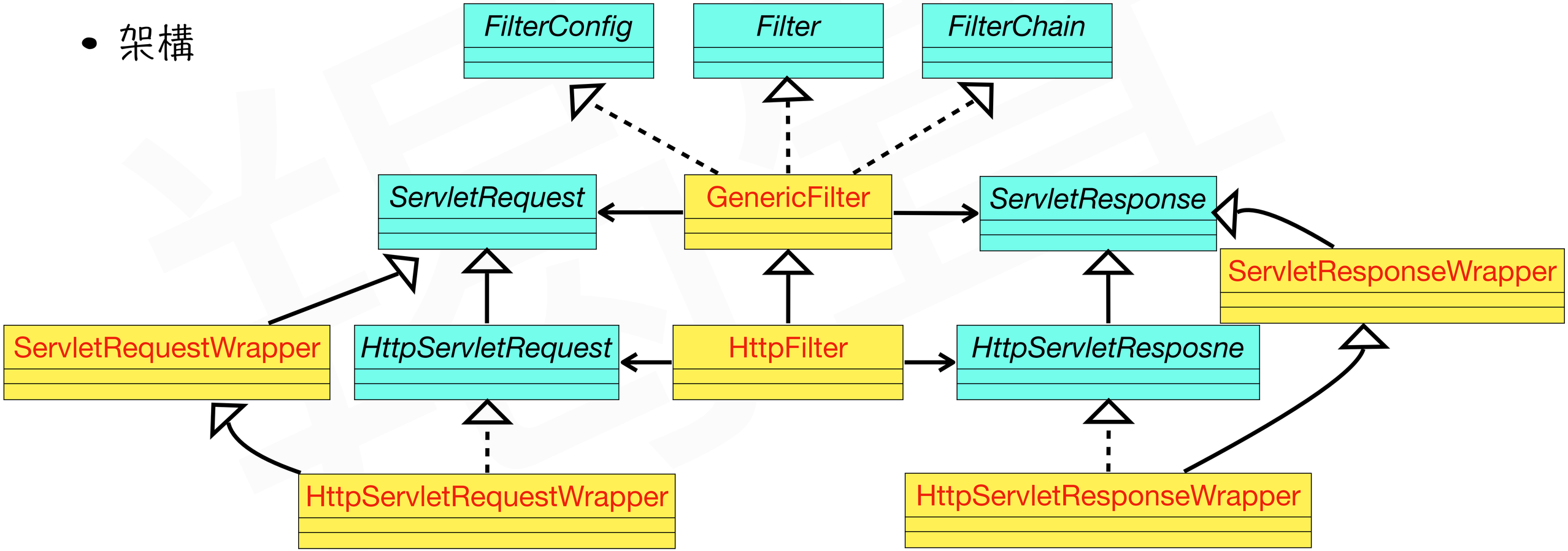
Filter

- 生命週期
  - 載入Filter
  - 實例化Filter
  - void init(FilterConfig config)
  - void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
    - void chain.doFilter(request, response)
  - void destroy()

Web Container

Filter

- 架構



Web Container

Filter

指定Servlet名稱或對應路徑

- 佈署

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/path")
public class PathServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        var name = request.getParameter("username");
        var out = response.getWriter();
        out.println("Hello "+name);
        System.out.println("servlet");
        out.close();
    }
}
```

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebFilter(urlPatterns="/path")
public class PathFilter extends HttpFilter {
    public void doFilter(HttpServletRequest request,
            HttpServletResponse response, FilterChain chain)
            throws ServletException, IOException {
        System.out.println("before");
        chain.doFilter(request, response);
        System.out.println("after");
    }
}
```

進入下一個Filter或Servlet

此行之前在Servlet之前執行

此行之後在Servlet之後執行

Java Web

Web Container

- Wrapper
  - 包裹原始的請求及回應，以修改或增強特性
    - 需搭配Filter使用
  - 請求包裹器
  - 回應包裹器

Web Container

Wrapper

- 請求包裹器

建構子輸入原始request物件

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

public class RequestWrapper extends HttpServletRequestWrapper {
  public RequestWrapper(HttpServletRequest req){
    super(req);
  }
  @Override
  public String getParameter(String name){
    return getRequest().getParameter(name).toUpperCase();
  }
}
```

取得原始request

複寫原始方法

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebFilter(urlPatterns="/path")
public class PathFilter extends HttpFilter {
    public void doFilter(HttpServletRequest request,
            HttpServletResponse response, FilterChain chain)
            throws ServletException, IOException {
        System.out.println("before");
        var req = new RequestWrapper(request);
        chain.doFilter(req, response);
        System.out.println("after");
    }
}
```

在Filter中使用包裹器

使Servlet取得包裹器

Web Container

Wrapper

- 回應包裹器

建構子輸入原始response物件

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

public class ResponseWrapper extends HttpServletResponseWrapper {
  public ResponseWrapper(HttpServletResponse resp){
    super(resp);
  }
  @Override
  public PrintWriter getWriter() throws IOException{
    System.out.println("print");
    return getResponse().getWriter();
  }
}
```

取得原始response　複寫原始方法

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebFilter(urlPatterns="/path")
public class PathFilter extends HttpFilter {
    public void doFilter(HttpServletRequest request,
            HttpServletResponse response, FilterChain chain)
            throws ServletException, IOException {
        System.out.println("before");
        var req = new RequestWrapper(request);
        var resp = new ResponseWrapper(response);
        chain.doFilter(req, resp);
        System.out.println("after");
    }
}
```

在Filter中使用包裹器

使Servlet取得包裹器

Java Web

Web Container

- WebSocket
  - 建立一個client跟server之間的全雙工通道
  - 通道建立後，server不需request就可直接發送訊息給client
  - 編譯時需加入websocket-api.jar及websocket-client-api.jar

Web Container

WebSocket

- 使用WebSocket
  - @ServerEndpoint：作為WebSocket的Server端
    - value：為Server的URI，可設定路徑參數
  - @ClientEndpoint：作為WebSocket的Client端
  - @OnOpen：WebSocket連線開啟時呼叫
  - @OnClose：WebSocket連線關閉時呼叫
  - @OnError：WebSocket連線錯誤時呼叫
  - @OnMessage：接收到訊息時呼叫
    - maxMessageSize：訊息最大大小
  - @PathParam：路徑參數
    - value：路徑參數名稱

Web Container

WebSocket

路徑參數值

```html
<!DOCTYPE html>
<html>
<head>
    <title>WebSocket Chat</title>
</head>
<body>

<h2>WebSocket Chat</h2>
<textarea id="chatBox" rows="10" cols="30"></textarea><br>
<input type="text" id="message">
<button onclick="sendMessage()">Send</button>

<script>
    var ws = new WebSocket("ws://localhost:8080/demo/chat/1");

    ws.onmessage = function(event) {
        var chatBox = document.getElementById("chatBox");
        chatBox.value += event.data + "\n";
    };

    function sendMessage() {
        var message = document.getElementById("message").value;
        ws.send(message);
        document.getElementById("message").value = "";
    }
</script>

</body>
</html>
```

註冊一個有參數的路徑

取得路徑參數

```java
package edu.javaweb;
import jakarta.websocket.*;
import jakarta.websocket.server.*;
import java.io.IOException;
import java.util.concurrent.*;
import java.util.*;

@ServerEndpoint("/chat/{number}")
public class ChatServer {
    private static final Set<Session> sessions = new CopyOnWriteArraySet<>();

    @OnOpen
    public void onOpen(Session session, @PathParam("number") String number) {
        sessions.add(session);
    }
    @OnClose
    public void onClose(Session session) {
        sessions.remove(session);
    }
    @OnMessage
    public void onMessage(String message, Session session,
                          @PathParam("number") String number) throws IOException {
        for (Session s : sessions) {
            if (s.isOpen() && !session.getId().equals(s.getId())) {
                s.getBasicRemote().sendText(message);
            }
        }
    }
    @OnError
    public void onError(Session session, Throwable throwable) {
        sessions.remove(session);
        try {
            session.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Java Web

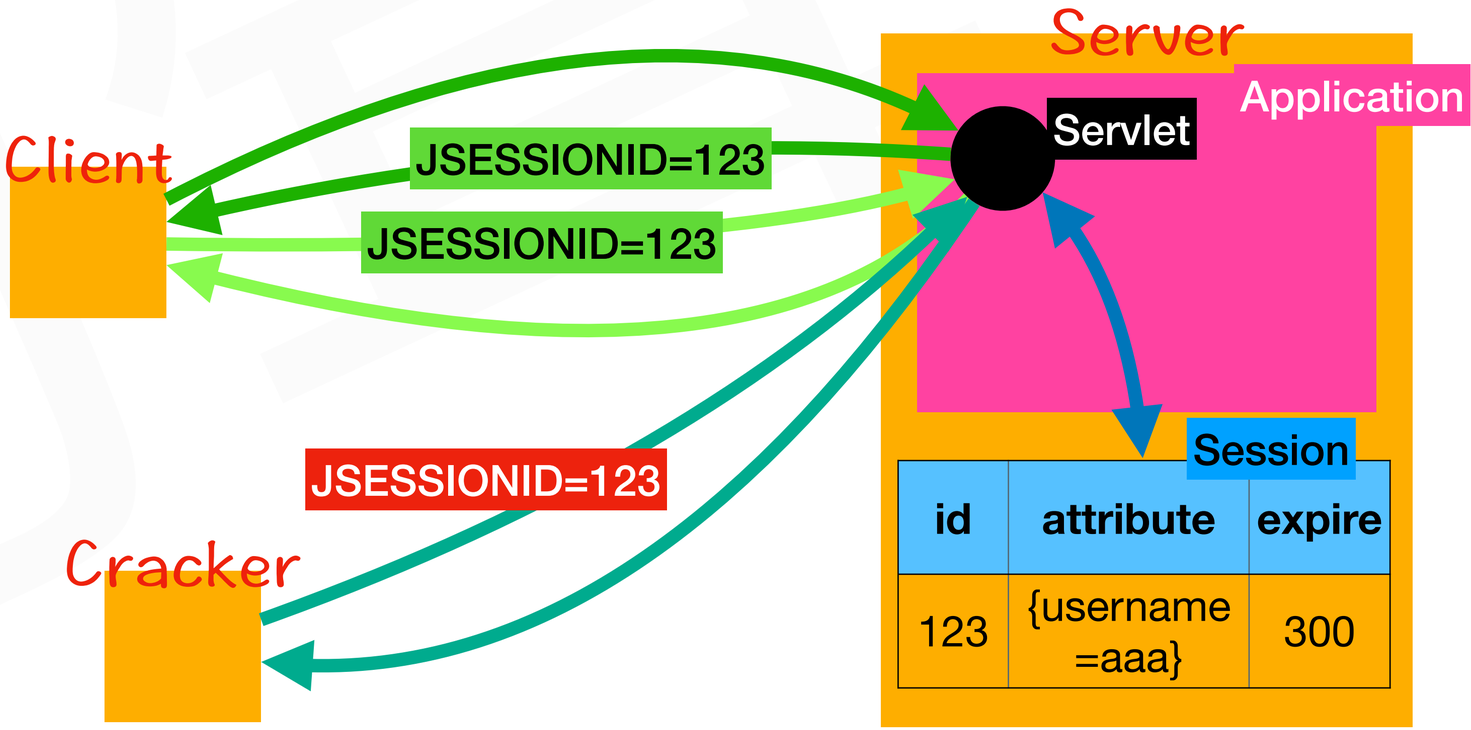
應用與實務

- Session
  - Session的用途
  - 取得與管理Session
  - 分散式Session

Java Web

Session

- Session的用途
  - 因應http的無狀態，server要識別同一client的連續動作，在cookie或url上加入session id
  - session有其對應範圍的attribute，用於存放相關資料
  - 每個session都是隔離開來的，server只認session id
    - Session Hijacking：Session欄截，不同client使用相同session



Session

取得Session

- 從request取得Session
  - HttpSession getSession(boolean create)
    - true表示不存在會new新的
    - false表示不存在回傳null
  - HttpSession getSession()
    - 與true相同
  - var session = request.getSession()
  - 使用session.isNew()可測試session是否為新的
    - 前當的request產生的session為新
    - session.getId()取得session id

取得session

使用SessionAttribute

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet(urlPatterns="/parameters")
public class ParameterServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
        HttpSession session = request.getSession();
        String user = (String) session.getAttribute("user");
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        if (user == null) {
            String newUser = request.getParameter("user");
            if (newUser != null && !newUser.isEmpty()) {
                session.setAttribute("user", newUser);
                out.println("<html><body>");
                out.println("<p>First time visiting. "+
                    "Session attribute 'user' is set to: " +
                    newUser + "</p>");
                out.println("</body></html>");
            } else {
                out.println("<html><body>");
                out.println("<p>Please provide a \"user\""+
                            " parameter in the request.</p>");
                out.println("</body></html>");
            }
        } else {
            out.println("<html><body>");
            out.println("<p>Welcome back, " + user + "!</p>");
            out.println("</body></html>");
        }
    }
}
```

Session

取得Session

- 從request取得Session
  - boolean isRequestedSessionIdFromCookie()
    - session來自於cookie
  - boolean isRequestedSessionIdFromURL()
    - session來自於URL

Session

取得Session

- 從request取得Session
  - session來自於cookie
  - request.getCookies()取得JSESSIONID與session.getId()值相同
  - 修改Cookie中存在Session ID的Key
    - 於web.xml中的<session-config>中的<cookie-config>元素中設定
      - <name>設定key的名稱
      - <http-only>設定是否為僅http可使用此cookie
    - 透過ServletContext.getSessionCookieConfig().setName()來設定，但須於server啟動時設定；可搭配ServletContextListener使用

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
        http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="false">
  <session-config>
    <session-timeout>60</session-timeout>
    <cookie-config>
      <name>jsessionid</name>
      <http-only>true</http-only>
    </cookie-config>
  </session-config>
</web-app>
```

session取自cookie的設定

Session

取得Session

- 從request取得Session
  - session來自於URL
    - response中，會在URL加上jsessionid的參數
    - String encodeURL (String url)
      - 在回傳的URL中加上jsessionid
    - String encodeRedirectURL (String url)
      - 在sendRedirect時使用的URL

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;

@WebServlet("/path")
public class PathServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        var out = response.getWriter();
        response.setContentType("text/html");

        var contextPath = request.getContextPath();
        var encodedURL = response.encodeURL(
                                contextPath+"/parameters");
        out.println("<a href='" + encodedURL
                + "'>Click here for encoded URL</a><br>");
    }
}
```

產生超鏈結的URL

Session

管理Session

- 管理Session的逾時機制
  - web.xml中設定
  - 程式碼中設定

Session

管理Session

- web.xml中設定
  - <session-config>
    - <session-timeout>
    - 以分為單位

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
        http://xmlns.jcp.org/xml/ns/javaee
        http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="false">
  <session-config>
    <session-timeout>60</session-timeout>
    <cookie-config>
      <name>jsessionid</name>
      <http-only>true</http-only>
    </cookie-config>
  </session-config>
</web-app>
```

60分鐘後session自動逾時(失效)

Session

管理Session

- 程式碼中設定
  - session.setMaxInactiveInterval(int)
    - 設定逾時時間，秒為單位
    - 可透過ServletContext做全域設定
      - void setSessionTimeout(int sessionTimeout)
      - 分為單位，僅影響新的session
  - session.invalidate()
    - 使session立即逾時(失效)
  - 可使用request.changeSessionId()來變更Session ID

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebServlet("/hello")
public class HelloWorldServlet extends HttpServlet {

    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws IOException, ServletException {
        response.setContentType("text/html");
        var session = request.getSession();
        session.invalidate();
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>Hello, World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

session立即逾時(失效)

Session

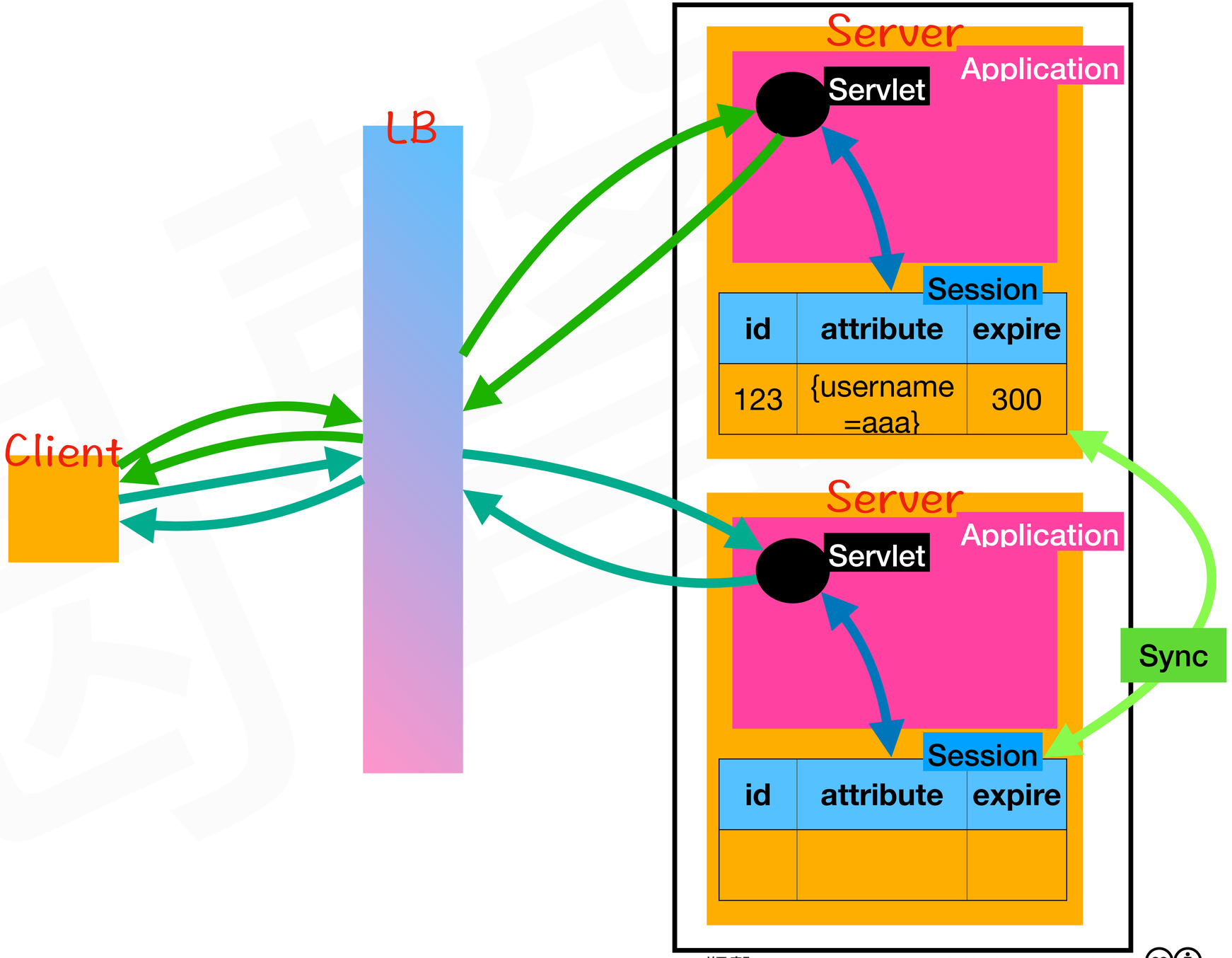
管理Session

	web.xml中的<session-timeout>	setMaxInactiveInterval(int)
範圍	在web應用程式裡建立的所有Session預設值	只限方法被呼叫的Session值
單位	分	秒
0值的意義	Session不會逾時	Session立即逾時
負數值的意義	Session不會逾時	Session不會逾時

Session

分散式Session

- 分散式Session
  - 容器叢集
  - web.xml
    - <distributable/>



Java Web

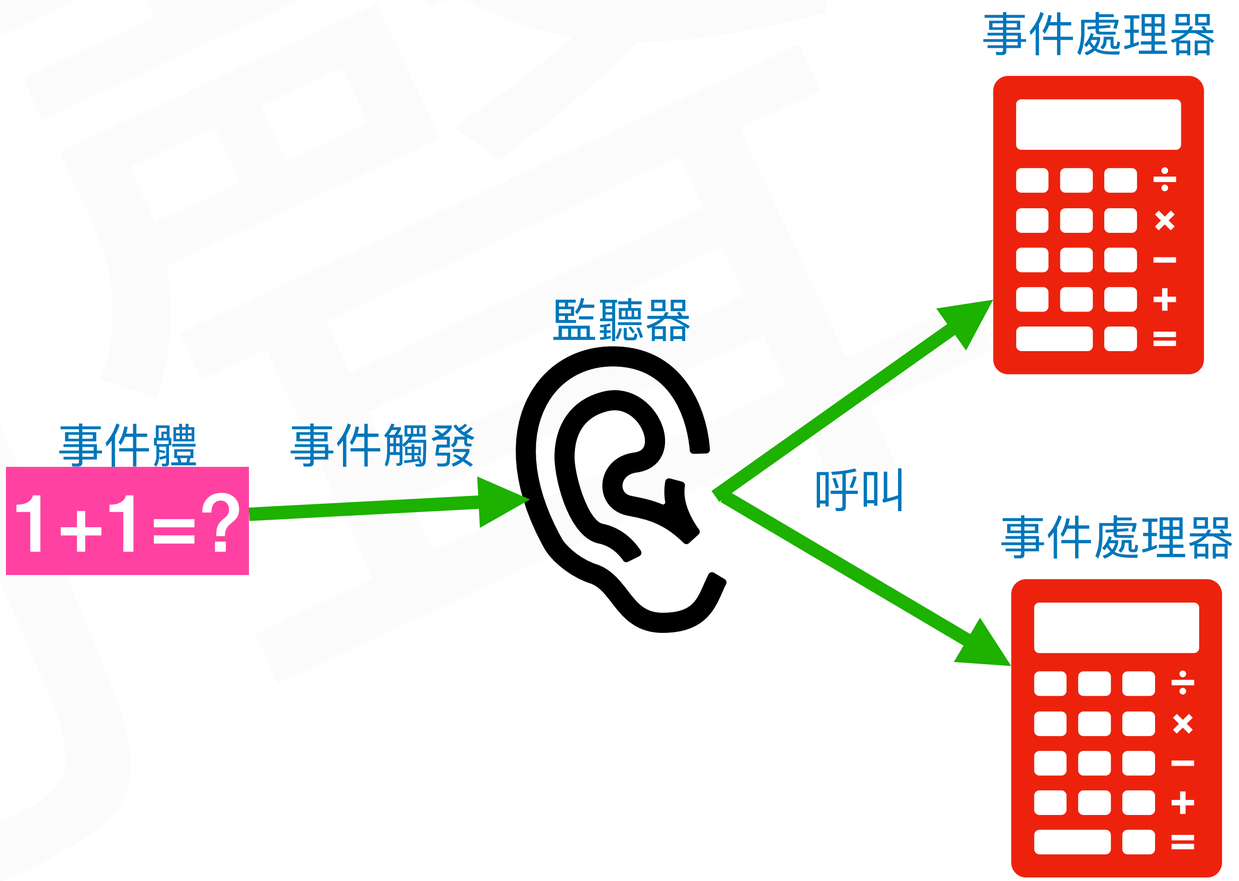
應用與實務

- Listener
  - 事件驅動
  - 設定監聽器
  - 監聽器類型

Java Web

Listener

- 事件驅動
  - 事件 (Event)：一個特定的事情發生，由使用者或系統觸發
  - 事件體 (Event Object)：發生事生的對象。事件體包含事件類型、來源
  - 監聽器 (Listener)：等待特定事件發生，並呼叫事件處理器
  - 事件處理器 (Event Handler)：處理事件的方法



Java Web

Listener

- 設定監聽器
  - 只需宣告給容器，容器會自行判斷監聽器類型
  - web.xml
    - <listener>
      - <listener-class>
  - Annotation
    - @WebListener

Java Web

Listener

- 監聽器的類型
  - 生命週期監聽器
  - 屬性監聽器
  - 其他監聽器

Listener

監聽器的類型

- 生命週期監聽器介面
  - ServletContextListener
    - void contextInitialized(ServletContextEvent sce)
    - void contextDestroyed(ServletContextEvent sce)
  - ServletContextEvent
    - ServletContext getServletContext()

註冊監聽器

監聽器類型

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebListener
public class MyListener implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent sce) {
        ServletContext sc = sce.getServletContext();
        System.out.println("Web application " +
            sc.getServletContextName() + " is starting...");
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {
        ServletContext sc = sce.getServletContext();
        System.out.println("Web application " +
            sc.getServletContextName() + " is shutting down...");
    }
}
```

監聽應用程式佈署

監聽應用程式反佈署

Listener

監聽器的類型

- 生命週期監聽器介面
  - HttpSessionListener
    - void sessionCreated(HttpSessionEvent se)
    - void sessionDestroyed(HttpSessionEvent se)
  - HttpSessionEvent
    - HttpSession getSession()

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebListener
public class MyListener implements HttpSessionListener {
    @Override
    public void sessionCreated(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        System.out.println("Session created with ID: "
            + session.getId());
    }

    @Override
    public void sessionDestroyed(HttpSessionEvent se) {
        HttpSession session = se.getSession();
        System.out.println("Session with ID: "
            + session.getId() + " has been destroyed.");
    }
}
```

監聽器類型

監聽session建立

監聽session逾時

Listener

監聽器的類型

- 生命週期監聽器介面
  - ServletRequestListener
    - void requestInitialized(ServletRequestEvent sre)
    - void requestDestroyed(ServletRequestEvent sre)
  - ServletRequestEvent
    - ServletContext getServletContext()
    - ServletRequest getServletRequest()

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebListener
public class MyListener implements ServletRequestListener {
    @Override
    public void requestInitialized(ServletRequestEvent sre) {
        ServletRequest request = sre.getServletRequest();
        System.out.println("ServletRequest initialized. Remote IP: "
            + request.getRemoteAddr());
    }

    @Override
    public void requestDestroyed(ServletRequestEvent sre) {
        ServletRequest request = sre.getServletRequest();
        System.out.println("ServletRequest destroyed. Remote IP: "
            + request.getRemoteAddr());
    }
}
```

監聽器類型

監聽request建立

監聽request銷毀

Listener

監聽器的類型

- 屬性監聽器
  - ServletContextAttributeListener
    - void attributeAdded(ServletContextAttributeEvent event)
    - void attributeRemoved(ServletContextAttributeEvent event)
    - void attributeReplaced(ServletContextAttributeEvent event)
  - ServletContextAttributeEvent
    - String getName()
    - Object getValue()

Listener

監聽器的類型

- ServletContextAttributeListener

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebListener
public class MyListener implements ServletContextAttributeListener{
    @Override
    public void attributeAdded(ServletContextAttributeEvent event) {
        System.out.println("An attribute was added to the ServletContext: " +
                           event.getName() + " = " + event.getValue());
    }
    @Override
    public void attributeRemoved(ServletContextAttributeEvent event) {
        System.out.println("An attribute was removed from the ServletContext: " +
                           event.getName() + " (previous value: " + event.getValue() + ")");
    }
    @Override
    public void attributeReplaced(ServletContextAttributeEvent event) {
        System.out.println("An attribute was replaced in the ServletContext: " +
                           event.getName() + " (previous value: " + event.getValue() + ")");
    }
}
```

監聽器類型

監聽新增屬性

監聽移除屬性

監聽修改屬性

Listener

監聽器的類型

- 屬性監聽器
  - HttpSessionAttributeListener
    - void attributeAdded(HttpSessionBindingEvent event)
    - void attributeRemoved(HttpSessionBindingEvent event)
    - void attributeReplaced(HttpSessionBindingEvent event)
  - HttpSessionBindingEvent
    - String getName()
    - Object getValue()

Listener

監聽器的類型

- HttpSessionAttributeListener

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebListener
public class MyListener implements HttpSessionAttributeListener{
    @Override
    public void attributeAdded(HttpSessionBindingEvent event) {
        System.out.println("An attribute was added to the HttpSession: " +
                           event.getName() + " = " + event.getValue());
    }
    @Override
    public void attributeRemoved(HttpSessionBindingEvent event) {
        System.out.println("An attribute was removed from the HttpSession: " +
                           event.getName() + " (previous value: " + event.getValue() + ")");
    }
    @Override
    public void attributeReplaced(HttpSessionBindingEvent event) {
        System.out.println("An attribute was replaced in the HttpSession: " +
                           event.getName() + " (previous value: " + event.getValue() + ")");
    }
}
```

監聽器類型

監聽新增屬性

監聽移除屬性

監聽修改屬性

Listener

監聽器的類型

- 屬性監聽器
  - ServletRequestAttributeListener
    - void attributeAdded(ServletRequestAttributeEvent srae)
    - void attributeRemoved(ServletRequestAttributeEvent srae)
    - void attributeReplaced(ServletRequestAttributeEvent srae)
  - ServletRequestAttributeEvent
    - String getName()
    - Object getValue()

Listener

監聽器的類型

- ServletRequestAttributeListener

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebListener
public class MyListener implements ServletRequestAttributeListener{
    @Override
    public void attributeAdded(ServletRequestAttributeEvent event) {
        System.out.println("An attribute was added to the ServletRequest: " +
                           event.getName() + " = " + event.getValue());
    }
    @Override
    public void attributeRemoved(ServletRequestAttributeEvent event) {
        System.out.println("An attribute was removed from the ServletRequest: " +
                           event.getName() + " (previous value: " + event.getValue() + ")");
    }
    @Override
    public void attributeReplaced(ServletRequestAttributeEvent event) {
        System.out.println("An attribute was replaced in the ServletRequest: " +
                           event.getName() + " (previous value: " + event.getValue() + ")");
    }
}
```

監聽器類型

監聽新增屬性

監聽移除屬性

監聽修改屬性

Listener

監聽器的類型

- 其他監聽器
  - HttpSessionBindingListener
  - HttpSessionActivationListener
  - HttpSessionIdListener
  - AsyncListener
  - ReadListener
  - WriteListener

Listener

監聽器的類型

- HttpSessionBindingListener
  - void valueBound (HttpSessionBindingEvent event)
  - void valueUnbound (HttpSessionBindingEvent event)
- HttpSessionBindingEvent
  - String getName()
  - Object getValue()

監聽器類型

```java
package edu.javaweb;

import jakarta.servlet.http.*;

public class User implements HttpSessionBindingListener {

    private String name;

    public User(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public void valueBound(HttpSessionBindingEvent event) {
        System.out.println("User object with name: "
            + name + " was added to session.");
    }

    @Override
    public void valueUnbound(HttpSessionBindingEvent event) {
        System.out.println("User object with name: "
            + name + " was removed from session.");
    }
}
```

監聽物件被加入Session

監聽物件被移除Session

Listener

監聽器的類型

- HttpSessionActivationListener
  - void sessionDidActivate (HttpSessionEvent se)
  - void sessionWillPassivate (HttpSessionEvent se)
- HttpSessionEvent
  - HttpSession getSession()

監聽器類型

```
package edu.javaweb;

import jakarta.servlet.http.*;

public class User implements HttpSessionActivationListener{

    private String name;

    public User(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    @Override
    public void sessionWillPassivate(HttpSessionEvent event) {
        System.out.println("Session for user: "
            + name + " will be passivated.");
    }

    @Override
    public void sessionDidActivate(HttpSessionEvent event) {
        System.out.println("Session for user: "
            + name + " has been activated.");
    }
}
```

分散式session中，監聽物件被序列化

分散式session中，監聽物件被反序列化

Listener

監聽器的類型

- HttpSessionIdListener
  - void sessionIdChanged (HttpSessionEvent event, String oldSessionId)
- HttpSessionEvent
  - HttpSession getSession()

```
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

@WebListener
public class MyListener implements HttpSessionIdListener {
    @Override
    public void sessionIdChanged(HttpSessionEvent event, String oldSessionId) {
        System.out.println("Session ID changed from: "
            + oldSessionId + " to: " + event.getSession().getId());
    }
}
```

監聽器類型

監聽Session Id更換

Listener

監聽器的類型

- AsyncListener
  - void onStartAsync(AsyncEvent event)
  - void onComplete(AsyncEvent event)
  - void onError(AsyncEvent event)
  - void onTimeout(AsyncEvent event)
- AsyncEvent
  - AsyncContext getAsyncContext()
  - ServletRequest getSuppliedRequest()
  - ServletResponse getSuppliedResponse()
  - Throwable getThrowable()
- asyncContext.addListener(new MyListener())

```java
package edu.javaweb;

import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;

public class MyListener implements AsyncListener{
    @Override
    public void onComplete(AsyncEvent event) throws java.io.IOException {
        System.out.println("Async operation completed.");
    }

    @Override
    public void onTimeout(AsyncEvent event) throws java.io.IOException {
        System.out.println("Async operation timed out.");
    }

    @Override
    public void onError(AsyncEvent event) throws java.io.IOException {
        System.out.println("Error occurred in async operation: "
            + event.getThrowable());
    }

    @Override
    public void onStartAsync(AsyncEvent event) throws java.io.IOException {
        System.out.println("Async operation started.");
    }
}
```

監聽器類型

監聽非同步完成

監聽非同步逾時

監聽非同步錯誤

監聽非同步開始

但實際註冊Listener時

非同步動作已經開始

Listener

監聽器的類型

- ReadListener
  - void onDataAvailable()
  - void onAllDataRead()
  - void onError (Throwable t)

監聽器類型，使用NIO讀取請求本體

```java
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet(urlPatterns="/parameters", asyncSupported = true)
public class ParameterServlet extends HttpServlet {
    @Override
    public void doPost(HttpServletRequest req, HttpServletResponse resp)
                throws ServletException, IOException {
        AsyncContext asyncContext = req.startAsync();
        ServletInputStream input = req.getInputStream();
        input.setReadListener(new MyReadListener(input, resp));
    }
    class MyReadListener implements ReadListener {
        private ServletInputStream input = null;
        private HttpServletResponse resp = null;
        private StringBuilder receivedData = new StringBuilder();

        public MyReadListener(ServletInputStream in, HttpServletResponse r) {
            input = in;
            resp = r;
        }
        @Override
        public void onDataAvailable() throws IOException {
            byte[] buffer = new byte[512];
            int length = -1;
            while (input.isReady() && (length = input.read(buffer)) != -1) {
                receivedData.append(new String(buffer, 0, length));
            }
        }
        @Override
        public void onAllDataRead() throws IOException {
            System.out.println("Data received: " + receivedData.toString());
            resp.getWriter().write("Data received: " + receivedData.toString());
        }
        @Override
        public void onError(Throwable t) {
            t.printStackTrace();
        }
    }
}
```

第一次可讀取資料時呼叫

所有資料都讀完的時候呼叫

發生錯誤的時候呼叫

Listener

監聽器的類型

- WriteListener
  - void onWritePossible()
  - void onError (Throwable t)

第一次可輸出資料時呼叫

發生錯誤的時候呼叫

監聽器類型，使用NIO輸出回應本體

```
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet(urlPatterns="/parameters", asyncSupported = true)
public class ParameterServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        AsyncContext asyncContext = request.startAsync();
        ServletOutputStream outputStream = response.getOutputStream();
        outputStream.setWriteListener(new WriteListener() {
            @Override
            public void onWritePossible() throws IOException {
                while (outputStream.isReady()) {
                    outputStream.write("Hello, World!".getBytes());
                    outputStream.close();
                }
            }

            @Override
            public void onError(Throwable t) {
                getServletContext().log("Error in WriteListener", t);
            }
        });
    }
}
```

Java Web

應用與實務

- Security
  - 安全性
  - 容器的安全性
  - https

Java Web

Security

- 安全性
  - 驗證(Authentication)：確認使用者身分
    - 帳號/密碼、OTP、第三方驗證、FIDO
  - 授權(Authorization)：賦予使用者權限
    - 使用權限檢查
  - 記錄(Accounting)：記錄使用者操作
    - 追察、計價
  - 資料完整性(Data Integrity)：確保資料收送雙方的內容一致
    - hash演算
  - 隱密性或資料隱私性(Confidentiality or Data Privacy)：確保資料收送雙方才能看
    - 加密演算(對稱式加密、非對稱式加密)

Java Web

Security

- 容器的安全性
  - 容器式安全
  - 程設式安全

Security
容器的安全性

- 容器式安全
  - 驗證
  - 授權
    - 加密

Security
容器的安全性

- 容器式安全(驗證)
  - auth-method
    - BASIC
      - HTTP基本驗證
    - FORM
      - 表單驗證
      - j_security_check, j_username, j_password
    - DIGEST
      - HTTP摘要驗證
    - CLIENT-CERT
      - client提供憑證

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <login-config>
        <auth-method>FORM</auth-method>
        <form-login-config>
            <form-login-page>/login.jsp</form-login-page>
            <form-error-page>/error.jsp</form-error-page>
        </form-login-config>
    </login-config>

</web-app>
```

表單驗證成功與失敗頁面

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Login Page</title>
</head>
<body>
    <h2>Please Login</h2>
    <form action="j_security_check" method="post">
        <div>
            <label for="j_username">Username:</label>
            <input type="text" id="j_username" name="j_username" required>
        </div>
        <div>
            <label for="j_password">Password:</label>
            <input type="password" id="j_password" name="j_password" required>
        </div>
        <div>
            <input type="submit" value="Login">
        </div>
    </form>
</body>
</html>
```

表單驗證的參數

Security

容器的安全性

- 容器式安全(授權)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <security-constraint>
        <web-resource-collection>
          <web-resource-name>admin</web-resource-name>
          <url-pattern>/admin</url-pattern>
          <http-method>GET</http-method>
          <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>ADMIN</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
    <security-role>
        <role-name>ADMIN</role-name>
    </security-role>

</web-app>
```

受保護的資源

未設定則七大方法皆啟動
空標籤則都不啟動

可使用Annotation設定

可使用資源的角色
需對應<security-role>

需對應容器角色

Security
容器的安全性

- 容器式安全(加密)
  - NONE
    - 不做保護
  - INTEGRAL
    - SSL/TLS身份驗證
    - 不加密
  - CONFIDENTIAL
    - SSL/TLS加密

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <security-constraint>
        <web-resource-collection>
          <web-resource-name>admin</web-resource-name>
          <url-pattern>/admin</url-pattern>
          <http-method>GET</http-method>
          <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>ADMIN</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
    <security-role>
        <role-name>ADMIN</role-name>
    </security-role>

</web-app>
```

client與server的連線保護

Security

容器的安全性

- 容器式安全(Annotation)
  - 未設定@HttpConstraint將拒絕所有請求

```
package edu.javaweb;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;

@WebServlet(urlPatterns="/admin")
@ServletSecurity(
  value=@HttpConstraint(rolesAllowed = {"admin"}),
  httpMethodConstraints = {
    @HttpMethodConstraint(value="GET", rolesAllowed = {"ADMIN"},
        transportGuarantee = TransportGuarantee.CONFIDENTIAL),
    @HttpMethodConstraint(value="POST", rolesAllowed = {"ADMIN"},
        transportGuarantee = TransportGuarantee.CONFIDENTIAL)
  }
)
public class ParameterServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("admin");
    }
}
```

Security
容器的安全性

- 程設式安全
  - boolean authenticate(HttpServletResponse response)：是否已驗證
  - void login(String username, String password)：進行驗證
  - void logout()：登出
  - Principal getUserPrincipal()：取得 Principal物件
    - String getName()：取得使用者名稱
  - String getRemoteUser()：取得使用者名稱
  - boolean isUserInRole(String role)：測試使用者是否為特定角色

Security

容器的安全性

- 程設式安全
  - 測試特定角色時，若不想寫死角色名稱，可在web.xml裡<servlet>中加入 <security-role-ref>
  - <security-role-ref>
    - <role-name>：程式中的名稱
    - <role-link>：對應到<security-role>的名稱

```
<servlet>
  <servlet-name>admin</servlet-name>
  <servlet-class>edu.javaweb.PathServlet</servlet-class>
  <security-role-ref>
    <role-name>administrator</role-name>
    <role-link>ADMIN</role-link>
  </security-role-ref>
</servlet>
```

Security

容器的安全性

- 程設式安全

判斷驗證狀態

進行驗證

登出

判斷角色

```java
package edu.javaweb;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;
@WebServlet(urlPatterns="/parameters")
public class ParameterServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String action = request.getParameter("action");
        if ("authenticate".equals(action)) {
            if (request.authenticate(response)) {
                response.getWriter().write("Authenticated as: "
                    + request.getUserPrincipal().getName());
            } else {
                response.getWriter().write("Authentication failed or was cancelled.");
            }
        } else if ("login".equals(action)) {
            try {
                request.login("username", "password");
                response.getWriter().write("Login successful for: "
                    + request.getUserPrincipal().getName());
            } catch (ServletException e) {
                response.getWriter().write("Login failed: " + e.getMessage());
            }
        } else if ("logout".equals(action)) {
            request.logout();
            response.getWriter().write("Logged out successfully.");
        } else if ("checkRole".equals(action)) {
            if (request.isUserInRole("admin")) {
                response.getWriter().write("User has 'admin' role.");
            } else {
                response.getWriter().write("User does not have 'admin' role.");
            }
        }
    }
}
```

Java Web

Security

- https
  - 加密連線
  - Push

Security

https

- keystore
  - cd <CATALINA_HOME>/conf
  - keytool -genkey -alias tomcat -keyalg RSA -keystore tomcat.jks
  - 設定server.xml
  - 重新啟動容器
  - https://localhost:8443

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true">
    <SSLHostConfig>
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
        <Certificate certificateKeystoreFile="conf/tomcat.jks"
                     type="RSA" certificateKeystorePassword="123456" />
    </SSLHostConfig>
</Connector>
```

憑證產生時的密碼

Security

https

- openssl
  - cd <CATALINA_HOME>/conf
  - openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
  - openssl pkcs12 -export -in cert.pem -inkey key.pem -out tomcat.p12 -name tomcat
  - 設定server.xml
  - 重新啟動容器
  - https://localhost:8443

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
           maxThreads="150" SSLEnabled="true">
    <UpgradeProtocol class="org.apache.coyote.http2.Http2Protocol" />
    <SSLHostConfig>
        <Certificate certificateKeystoreFile="conf/tomcat.p12"
                     certificateKeystoreType="PKCS12"
                     certificateKeystorePassword="123456" />
    </SSLHostConfig>
</Connector>
```

憑證產生時的密碼

Security

https

- Push
  - 必須在HTTP/2的https環境可運行
    - client跟server同時支援HTTP/2
  - 靜態資源才可推送
    - css, js, image

取得PushBuilder

```
package http2;
import java.io.*;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
@WebServlet("/push")
public class SimpleImagePush extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
            throws ServletException, IOException {
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        PrintWriter pw = resp.getWriter();
        PushBuilder pb = req.newPushBuilder();
        if (pb != null) {
            pb.path("servlets/images/code.gif");
            pb.push();
            pw.println("<html>");
            pw.println("<body>");
            pw.println("<p>The following image was provided via a push request.</p>");
            pw.println("<img src=\"" + req.getContextPath() + "/servlets/images/code.gif\"/>");
            pw.println("</body>");
            pw.println("</html>");
            pw.flush();
        } else {
            pw.println("<html>");
            pw.println("<body>");
            pw.println("<p>Server push requests are not supported by this protocol.</p>");
            pw.println("</body>");
            pw.println("</html>");
        }
    }
}
```

不存在則不支援