# 웹개발

## 웹 어플리케이션의 이해

* 데스크톱 어플리케이션에서 웹 어플리케이션이 개발되기까지 단계를 정리한 부분

## 데스크톱 어플리케이션

* 장점 : 실행속도가 빠름
* 단점 : 배포가 번거로움(모든 PC에 수동 배포 해야함)  
   보안에 취약함(DB가 개인 PC에 설치되어 있기 때문에 보안에 취약함)

## 클라이언트, 서버 어플리케이션(단일 스레드)

* 장점 : 비즈니스 로직 변경시 서버만 변경하면 되기 때문에 배포가 편리함
* 단점 : 단일 쓰레드로 하나의 요청에 하나의 응답만 가능함  
   여러 세션에서 요청시 처리순서를 기다려야 하는 문제점 발생  
   ( 앞 요청에서 오류가 발생한 경우 계속해서 대기를 해야함)

## 클리언트, 서버 어플리케이션(다중 클리아이언트)

* 장점 : 여러 클라이언트의 요청을 처리할 있다.

🡺C/S 프로그램(웹 제외)은 위와 같이 진화를 하였지만, 근본적으로 클라이언트 프로그램이 변경되면 PC마다 재설치를 해야한다는 단점을 가지고 있다.

## 웹 어플리케이션

* 웹 서버를 개설하여 클라이언트가 보는 화면을 통합 관리할 수 있다
* 어플리케이션 서버를 개설하여 비즈니스 로직을 통합 관리할 수 있다.
* 웹 서버와, 어플리케이션 서버를 통해서 수정된 화면과 비즈니스 로직을 쉽게 배포할 수 있게 되었다.

# 웹 프로그래밍 기초

# HTTP 프로토콜

* 웹브라우저와 **웹서버** 사이의 데이터 통신 규칙
* SOAP, RESTful : 클라이언트와 서버 서버 사이의 요청하고, 응답하는 방식을 의미  
  (두 방식 모두 HTTP 프로토콜을 응용하거나 확장한 기술)
* WebDAV(Web Distributed

## 요청 헤더(GET)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GET | 공백 | / | 공백 | HTTP/1.1 |
| 메서드 |  | 요청 URI |  | HTTP 버전 |

* 메서드 : GET, POST, HEAD, PUT, DELETE, TRACE, CONNCET, OPTIONS…
* URI : 이미지 ,동영상, 어플리케이션 등의 가상 경로
* HTTP 버전 : 현재 요청 HTTP 문서의 버전

## HTTP 헤더 유형

|  |  |  |
| --- | --- | --- |
| 유형 | 헤더명 | 설명(추후 하나씩 기재 예정) |
| 일반 헤더 | Cache-Control |  |
| Connection |  |
| Date |  |
| Pragma |  |
| Trailer |  |
| Transfer-Encoding |  |
| Upgrade |  |
| Via |  |
| Warning |  |
| 요청 헤더 | Accept |  |
| Accept-Charset |  |
| Accept-Encoding |  |
| Accept-Language |  |
| Authorization |  |
| Except |  |
| From |  |
| Host |  |
| If-Match |  |
| If-Modified-Since |  |
| If-None-Match |  |
| If-Range |  |
| If-unmodified-Since |  |
| Max-Forwards |  |
| Proxy-Authorization |  |
| Range |  |
| Referer |  |
| TE |  |
| User-Agent |  |
| 응답 헤더 | Accept-Rages |  |
| Age |  |
| ETag |  |
| Location |  |
| Proxy-Authenticate |  |
| Retry-After |  |
| Server |  |
| Vary |  |
| WWW-Authenticate |  |
| 본문 헤더 | Allow |  |
| Content-Encoding |  |
| Content-Language |  |
| Content-Location |  |
| Content-MD5 |  |
| Content-Range |  |
| Content-Type |  |
| Expires |  |
| Last-Modified |  |
| 기타 확장 헤더 |  |

## 응답헤더

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HTTP/1.1 | 공백 | 200 | 공백 | OK |
| HTTP 버전 |  | 상태 코드 |  | 상태 설명 |

|  |  |
| --- | --- |
| **상태코드** | **상태설명** |
| 200 | 요청이 성공적으로 처리되었다 |
| 301 | 요청한 자원이 이동되었다.  헤더 정보에 이동 위치를 알려줄 테니 다시 요청하라 |
| 304 | 클라이언트가 임시 보관한 응답결과와 다르지 않다. |
| 400 | 잘못된 요청이다. |
| 404 | 요청한 자원을 못 찾았다.  서버 내부에서 오류가 발생하였다. |
| 500 |

# GET 요청

* GET 요청시 서버에 데이터를 전달하는 방법 기술

## GET 요청의 특징

* URL에 데이터를 포함 -> 데이터 조회에 적합
* 바이너리 및 대용량 데이터 전송 불가
* 요청라인과 헤드 필드의 최대 크기 : HTTP 사양에는 제한 사항 없음  
   대용량 URL로 인한 문제 발생-> 웹 서버에 따라 크기 제한 있음  
   (Microsoft IIS 6.0 : 16KB, Apache 웹 서버 : 8KB)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 서비스주소 | ? | V1 | = | 23 | & | V2 | = | 15 |
|  | 서비스와 데이터를 분리 | 매개변수명 |  | 값 | 데이터 구분자 | 매개변수명 |  | 값 |

## GET 요청의 문제점

* 웹 주소창에 사용자가 입력한 정보가 모두 노출되기 때문에 보안에 취약하다
* 바이너리 데이터를 전송할 수 없다.

: BASE64 인코딩 방식을 이용하여 바이너리 데이터(이미지,동영상)를 문자화해서 보낼 수는 있지만, 헤더에 용량제한으로 적합하지 않다.

## GET 요청시 HTML Form의 Method 속성 값 : get (기본값이 get 이므로 기재하지 않아도 무관)

**<form action=”LoginServlet” method = “get” >**

# POST 요청

## POST 요청의 특징

* URL에 데이터가 포함되지 않음->외부노출방지
* 메시지 본문에 데이터 포함->실행 결과 공유 불가
* 바이너리 및 대용량 전송 가능

## POST 요청시 HTML Form의 Method 속성 값 : post

**<form action=”LoginServlet” method = “post” >**

## POST 바이너리 데이터 전송(파일업로드)

* 파일 업로드 폼 소스
* Enctype 기본값 : ‘application/x-www-form-urlencoded’

|  |
| --- |
| <form action="FileUploadServlet" method="post"  enctype="**multipart/form-data**">  사진: <input type="file" name="photo"><br>  설명: <textarea name="description" cols="50" rows="3"></textarea><br>  <input type="submit" value="추가"><br>  </form> |

* 결과

|  |
| --- |
| * POST /Lesson02/FileUploadServlet HTTP/1.1 * Host: localhost:8080 * Content-Length: 879694 * Cache-Control: max-age=0 * Origin: http://localhost:8080 * Upgrade-Insecure-Requests: 1 * Content-Type: multipart/form-data; * boundary**=----WebKitFormBoundaryWCCbgrgoeullXWFH** * User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36 * Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8 * Referer: http://localhost:8080/Lesson02/MultipartTest.html * Accept-Encoding: gzip, deflate, br * Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7 * **------WebKitFormBoundaryWCCbgrgoeullXWFH** * Content-Disposition: form-data; name="photo"; filename="Chrysanthemum.jpg" * Content-Type: image/jpeg * //사진 바이너리 데이터 * ���� * . * . * . * . * . * **------WebKitFormBoundaryWCCbgrgoeullXWFH** * Content-Disposition: form-data; name="description" * 국화다~ |

## POST 일반 방식(문자전송) 멀티파트(이미지,동영상) 전송의 차이점

* 일반 방식(문자전송)에 사용되는 매개변수들은 &를 사용하여 구분이 가능하나 사진, 동영상 같은 바이너리 데이터는 &로 매개변수를 구분할 수 없다.
* 따라서 멀티파트 전송 방식을 사용하여 3) 의 결과와 같이 파트 구분자를 써서 전송하는 방식을 사용한다..

# 서블릿(Sevlet) 프로그래밍

* 서블릿이란 웹브라우저와 웹서버를 활용하여 좀 더 쉽게 서버 어플리케이션을 개발할 수 있도록 하는 기술이다.

## CGI 프로그램과 서블릿

* CGI 규칙이란 ‘Common Gateway Interface’의 약자로 웹서버와 프로그램 사이의 데이터를 주고 받는 방식을 의마한다.
* 웹서버에 의해 실행되며, CGI 규칙에 따라서 웹서버와 프로그램 사이에 데이터를 주고 받도록 작성된 프로그램을 ‘CGI 프로그램’이라고 한다.
* 따라서 CGI 규칙을 지킨다면 어떤 프로그램이든 웹서버와 데이터를 주고 받을 수 있다.

웹브라우저

웹서버

프로그램

요청

응답

실행

결과

**CGI** 규칙

## 서블릿

* JAVA로 만든 CGI 프로그램을 서블릿 이라고 한다!
* **자바 서블릿이 다른 CGI 프로그램과 다른 점은, 웹서버와 직접 데이터를 주고받지 않으며, 전문 프로그램에 의해 관리된다는 것입니다.(?)  
  🡺 아래 나오는 서블릿 컨테이너를 의미하며, CGI 인터페이스를 정의한 CLASS를 의하는 것으로 생각됨.**

### 서블릿 컨테이너

* 서블릿의 생성과 실행, 소멸 등 생명주기를 관리하는 프로그램을 서블릿 컨테이너라고 불린다.
* 서블릿 컨테디너가 서블릿을 대신하여 CGI 규칙에 따라 웹서버와 데이터를 주고 받는다.  
  따라서 서블릿 개발자는 CGI 규칙에 대해 알 필요가 없게 된다.
* 대신 서블릿 컨테이터아ㅗ 서블릿 사이의 규칙을 알아야 한다.

웹서버

서블릿  
컨테이너

서블릿  
(.class)

요청 위임

응답

실행

결과

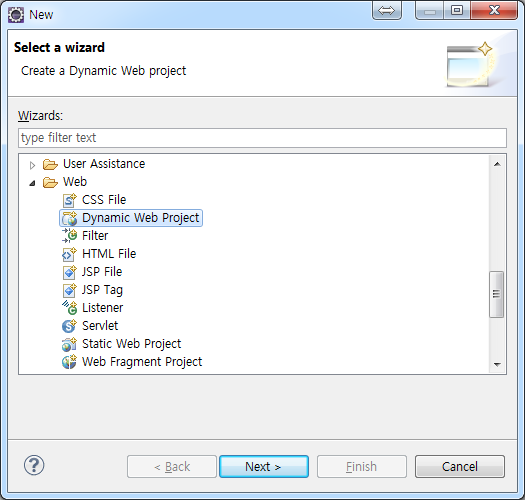
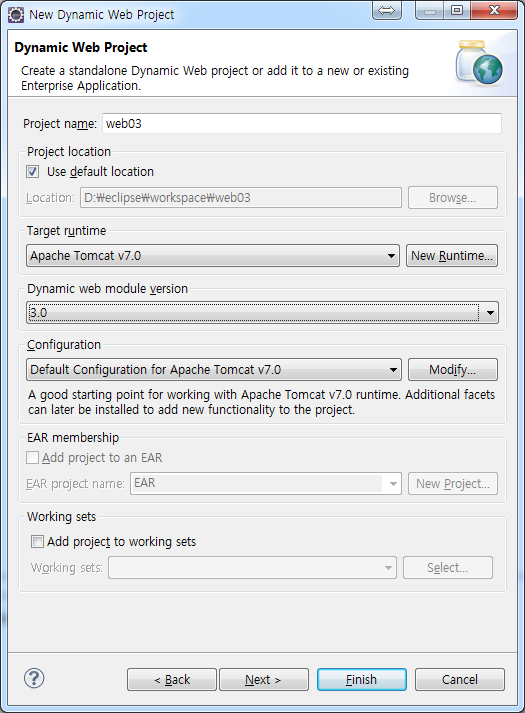
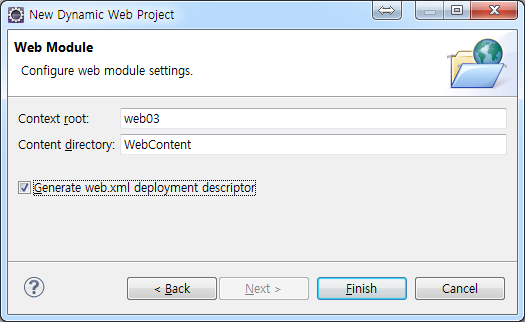
**CGI** 규칙

**서블릿** 규칙

시작 : 2018.01.03 20:44

# 웹 프로젝트 준비

## 웹 프로젝트 생성

1. Dynamic Web Project 생성  
   
2. 프로젝트 정보 입력  
   Target runtime : 웹서버  
   Dynamic web module version : 서블릿의 버전을 의미  
   
3. 웹어플리케이션 배치 정보 설정  
   Context root : 웹어플리케이션 이름을 의미  
   Content directory : 웹 콘텐츠 파일을 저장할 작업 폴더명  
   Generate web.xml deployment descriptor : 웹어플리케이션 배치 설명서(web.xml)   
    파일 자동 생성  
   

http://127.0.0.1:9999/**web03/Hello**

웹서버 주소

서비스 포트 번호

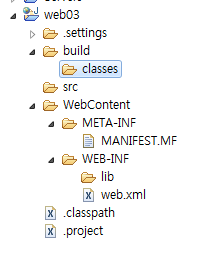
웹어플리케이션 이름

서블릿 이름

종료 : 2018.01.03 21:44 (책p103~P115, 총p12)

시작 : 2018.01.04 21:00

## 웹프로젝트 폴더 구조



- src : 자바 소스 파일을 위치함(서블릿 클래스, 필터, 리스너, 프로퍼티( .properties ) 파일 등)

- build / classes : 컴파일된 자바 클래스 파일(.java) 위치함

- WebContent : HTML(.html), CSS(.css), JavaScript(.js), JSP, 이미지 파일 등 웹 콘텐츠가 위치함  
 **웹 어플리케이션이 웹서버에 배치될 때 이 폴더의 내용물이 그대로 복사됨**

- WebContent / WEB-INF : 웹어플리케이션의 설정과 관련된 파일이 위치함  
 **이 폴더의 있는 파일은 클라이언트에서 요청 불가**

- WebContent / WEB-INF / web.xml :   
웹어플리케이션의 배치 설명서(Deployment Descriptor) 파일  
‘DD 파일’이라고도 부름 **서블릿, 필터, 리스너, 매개변수, 기본 웹페이지 등 웹 어플리케이션 컴포넌트들의 배치 정보를 이 파일에 기록한다.  
서블릿 컨테이너는 클라이언트 요청을 처리할 때 이 파일의 정보를 참고하여 컴포넌트들을 찾는다.**

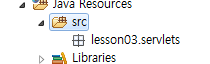
**-** WebContent / WEB-INF / lib :

JAR( JAVA Archive ) 파일을 두는 폴더  
 아카이브 파일은 .class 파일과 .properties 파일을 모아놓은 보관 파일이다.

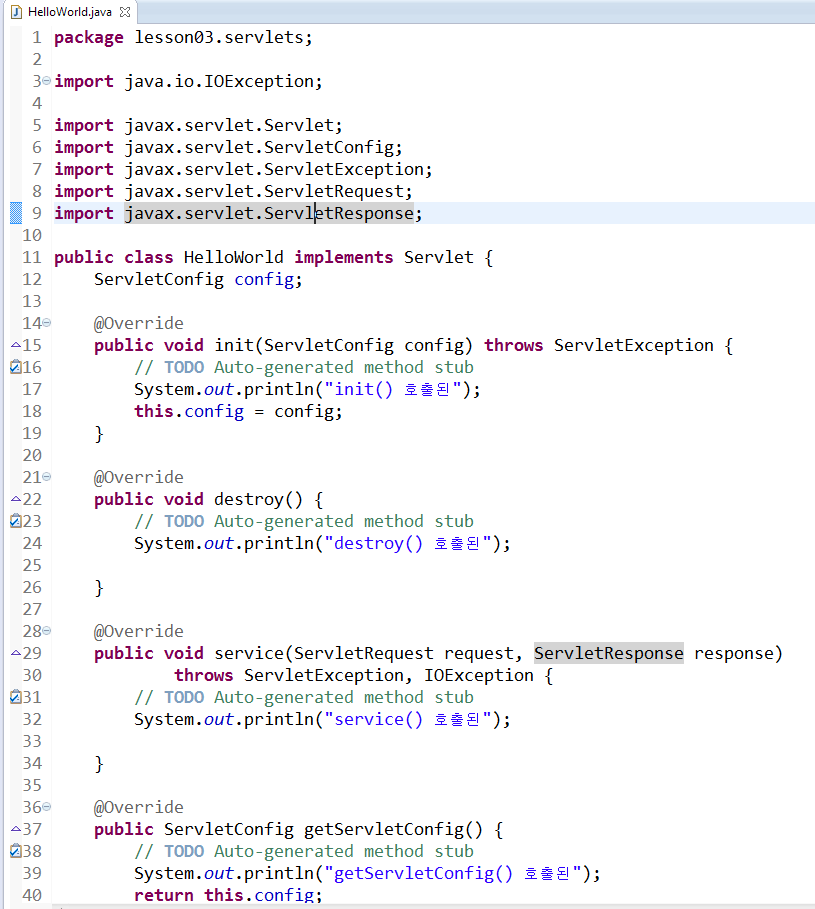
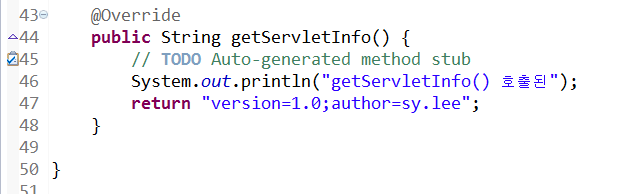
## 서블릿 만들기

* 해당 절에서는 서블릿 구동 방법 및 배치 설명서(web.xml) 설정 방법에 대해서 기재한다.

### 패키지 생성



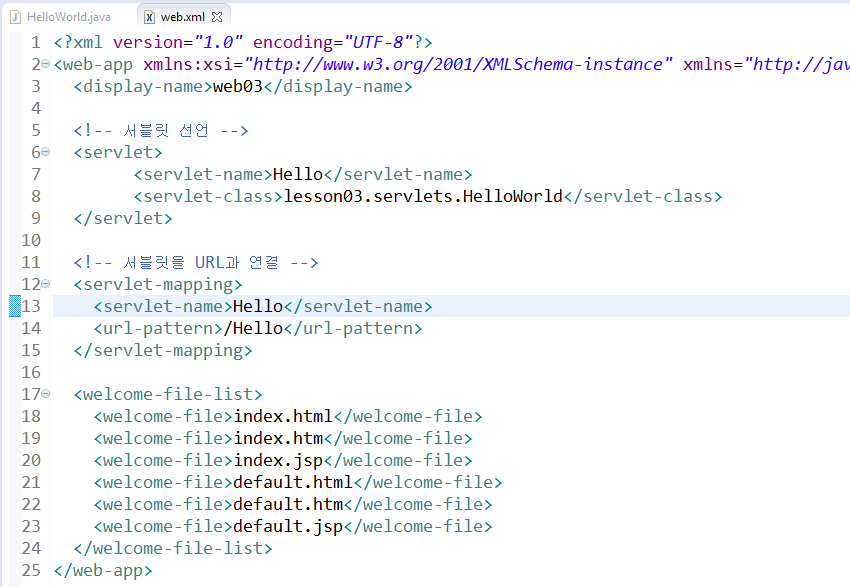
### 서블릿 클래스 작성

* 서블릿 컨테이너가 서블릿에 대해 호출할 메서드를 정의한 것이 서블릿 인터페이스 이다.
* init( ) : 서블릿 컨테이너가 서블릿을 생성한 후 초기화 작업을 수행하기 위해 호출하는 메서드로서, 서블릿이 클라이언트의 요청을 처리하기 전에 준비할 작업을 여기서 기재하면 된다.
* service( ) : 클라이언트가 요청할 때 마다 호출되는 메서드.
* destroy( ) : 서블릿 컨테이너가 종료되거나 웹 어플리케이션이 멈출 때, 또는 해당 서블릿을 비활성화 시킬 때 호출된다. 따라서 마무리 로직을 여기서 작성하면 됨.
* getServletconfig( ) : 서블릿 설정 정보를 다루는 ServletConfig 객체를 반환함  
   서블릿 초기 매개변수 값, 서블릿 이름, 서블릿 환경정보 확인 가능
* getServletInfo( ) : 서블릿에 대한 정보를 리턴함

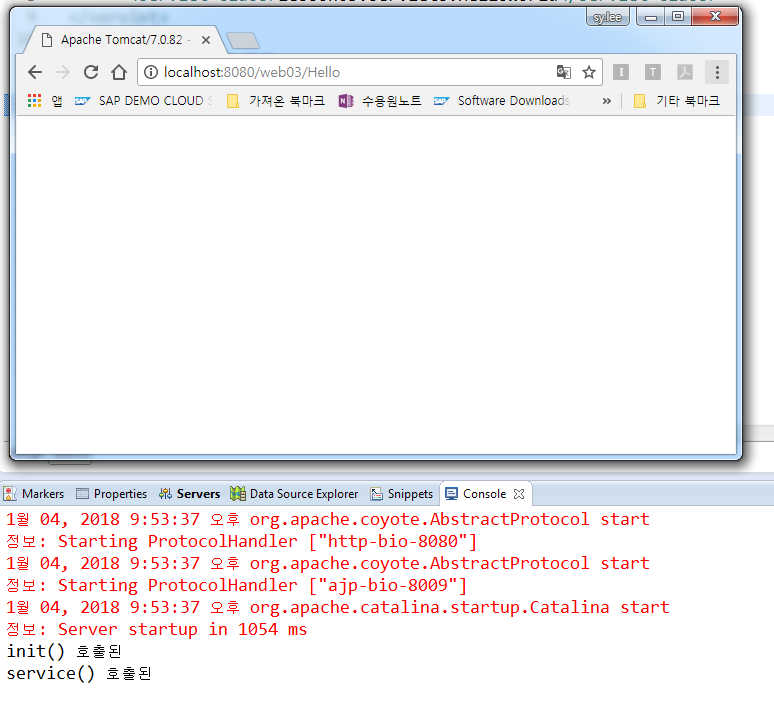
## 서블릿 배치 정보 작성 및 실행

1. 서블릿 배치 정보 작성



1. 웹서버 실행 결과

* 그림과 같이 /web03/Hello URL에 페이지가 열리는 것을 확인하였고, 서버 콘솔로그에 HelloWorld 서블릿에서 작성한 로그가 출력된 것을 확인할 수 있다.
* 웹페이지가 열릴 떄, init() -> service() 순서로 호출되는 것을 알 수 있다.

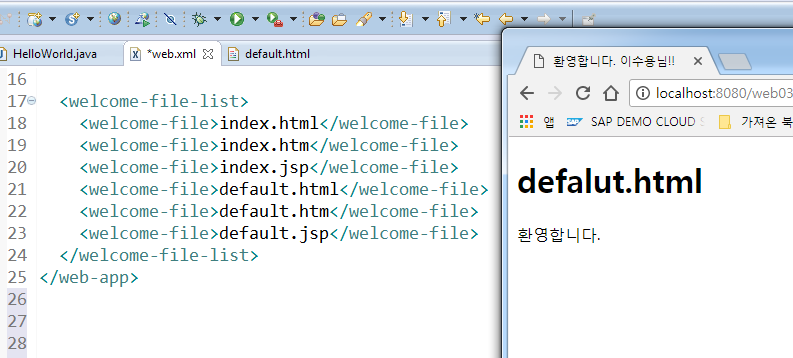


종료 : 2018.01.04 23:00

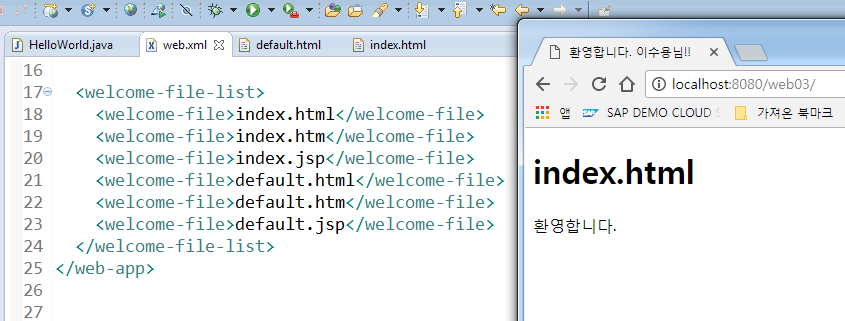
시작 : 2018.01.06

## 웹컴 파일들

* 웰컴 파일의 이름은 web.xml의 <welcome-file-list> 태그를 사용하여 설정할 수 있다.
* 해당 태그에 적힌 순서대로 조회하여 웹컴 파일을 실행한다.
* 아래 그림과 같이 default.html을 생성을 하고, <http://localhost:8080/web03> URL을 입력하면 웹컴 파일중 하나인 default.html이 화면에 출력되는 것을 확인할 수 있다.



* Index.html 파일을 생성을 하면 defult.html 파일이 웰컴 페이지로 열리는 것이 아니라 default.html이 웰컴페이지로 열리는 것을 확인 할 수 있었다.
* Web.xml에 등록된 순서대로 조회 후 먼저 검색된 웰컴 파일이 열리는 것을 확인 할 수 있다.

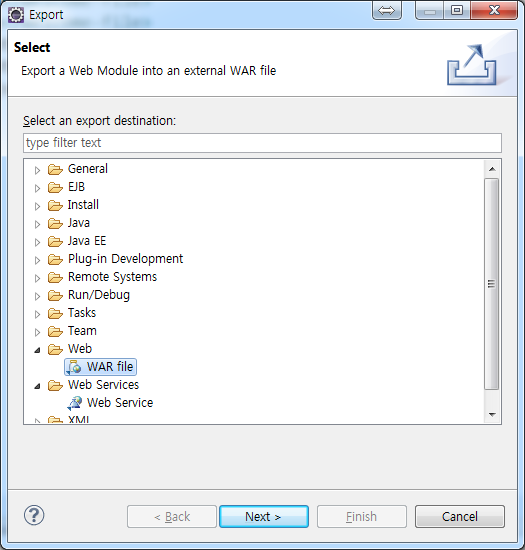


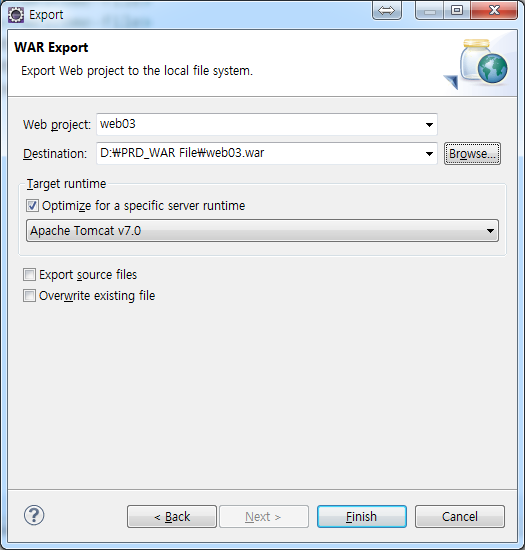
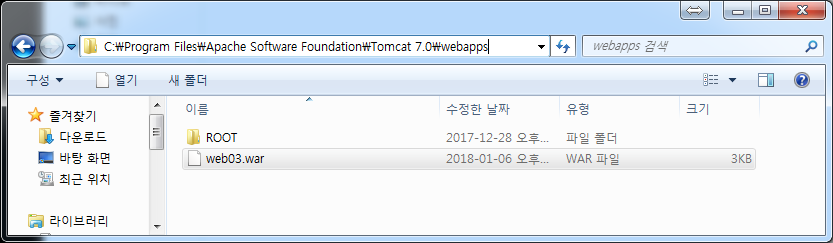
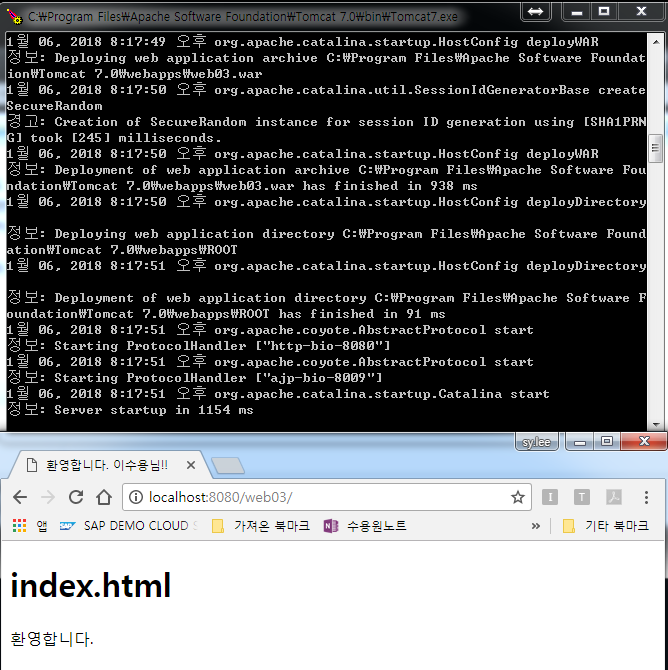
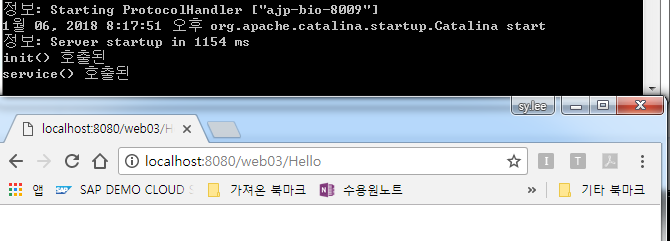
## 운영 서버에 배치하지

* 실제 운영하는 서버에 배치할 때는 배치할 파일들을 하나의 웹 아카이브 파일(.war)로 만들어서 배치 폴더에 복사한다.

### 웹 아카이브 파일 생성

* Ecilpse > File > Export 클릭 > Web > WAR File 선택 > Next



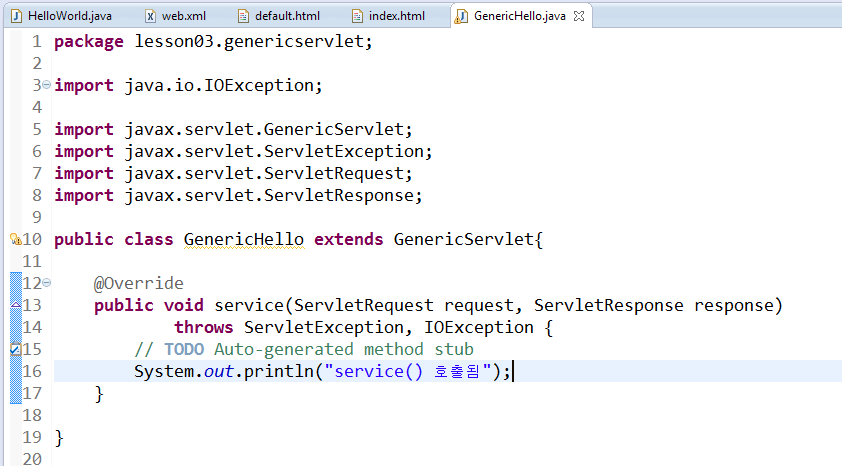
* Web Project 명과 파일 생성위치를 지정 후 Finish 클릭
* 
* Web03.war 파일을 톰캣 서버가 설치된 폴더에 아래와 같이 위치 시킨다
* 
* 톱캣을 실행시키고, 웹페이지를 열면 아래와 같이 정상적으로 웰컴파일이 실행되는 것을 알 수 있다.
* 
* 앞서 만든 Hello 서블릿도 정상 실행되는 것을 볼 수 있다.
* 

## GenericServlet의 사용

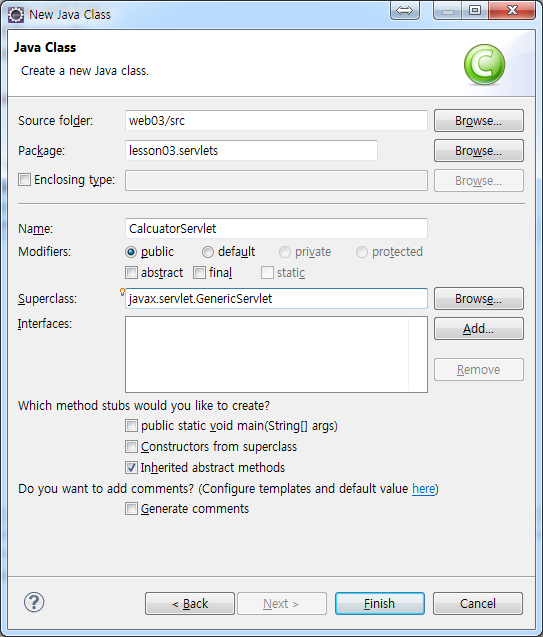
* GenericServlet 추상 클래스를 사용하여 서블릿을 만들어 본다.
* 웹 브라우저에서 서블릿을 데이터를 보내는 방법과 서블릿에서 이 데이터를 꺼내는 방법도 알아본다.
* 서블릿에서 작업한 결과를 웹 브라우저로 보내는 방법도 알아본다.

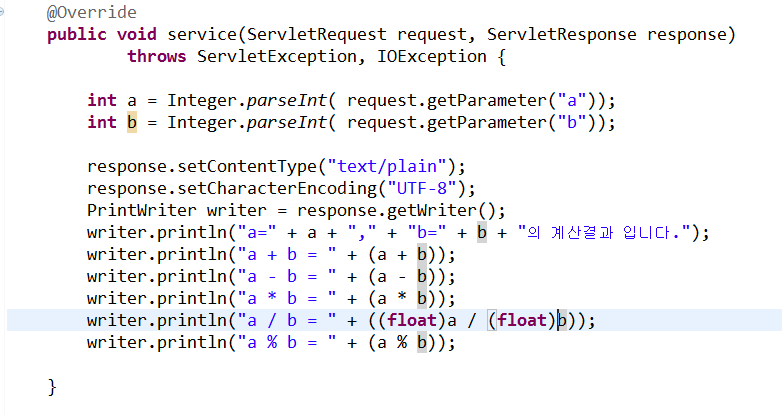
### GenericServlet의 용도

* 앞서 배운 Servlet 인터페이스를 구현하기 위해서는 init, destroy, getServletConfig, getServletInfo 메소를 모두 구현해야하는 단점이 있었는데, GenericServlet은 위 매소드들에 대해서 미리 구현하여 상속을 해주고 있다.
* 따라서 GenericServlet을 상속받는 다면, Servlet 인터페이스 메서드 중에서 service() 메소드만 구현하면 된다.



### 계산기 서블릿 작성



* 아래와 같이 service() 메소드 로직 구성
* 

### Servlet Requst 주요 메서드

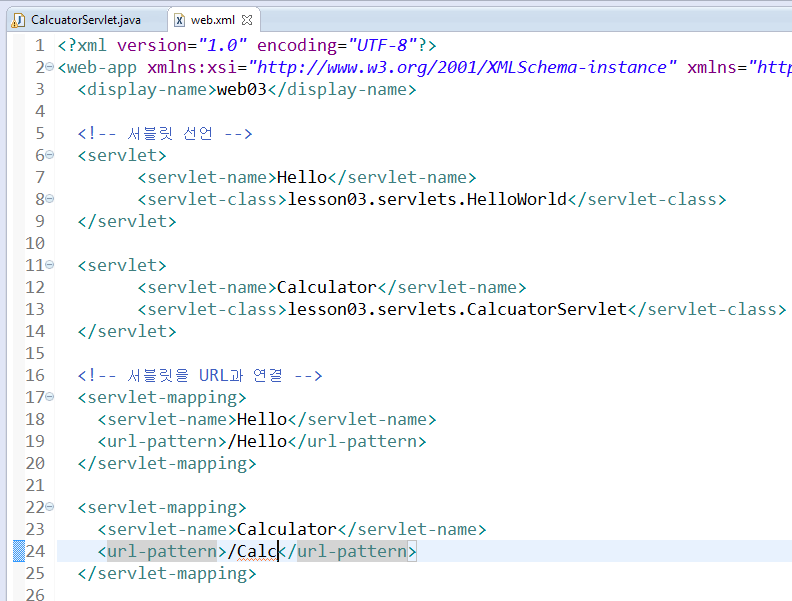
|  |  |
| --- | --- |
| 메서드 | 설명 |
| getRemoteAddr() | 서비스를 요청한 클라이언트의 IP수조를 반환 |
| getSchema() | 클라이언트가 요청한URI 형식 Scheme을 반환, 즉 URL에서 ‘:’ 문자 전에 오는 값을 변환 |
| getProtocol() | 요청 프로토콜의 이름과 버전을 반환  예 : HTTP/1.1 |
| getParamterNames() | 요청 정보에서 매개변수 이름만 추출하여 반환 |
| getParameterValues() | 요청 정보에서 매개변수 값만 추출하여 반환 |
| getParameterMap() | 요청 정보에서 매개변수들을 Map 객체에 담아서 반환 |
| SetCharacterEncoding() | POST 요청의 매개변수에 대해 문자 집합을 설정.  기본값은 ISO-8859-1로 설정되어 있다. 매개변수의 문자 집합을 정확히 지정해야만 제대로 변환된 유니코드 값을 반환합니다. 매개변수의 문자 집합을 지정하지 않으면 무조건 ISO-8859-1이라 가정하고 유니코드로 변환합니다. 주의할 점은 처음 getParameter()를 호출하기 전에 이 메서드를 먼저 호출해야만 적용됩니다. |

### ServletResponse

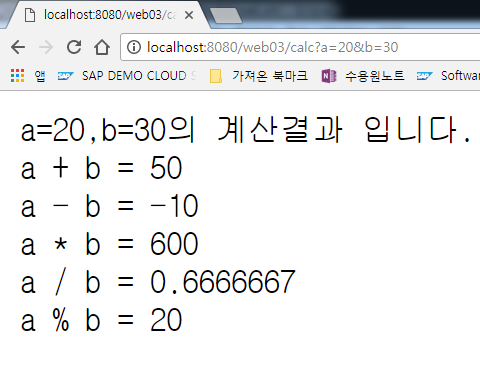
* 응답과 관련된 기능을 제공
* 클라이언트에게 출력하는 데이터의 인코딩 타입을 섲렁하고, 문자집합을 지정하며, 출력 데이터는 임시 보관하는 버퍼의 크기를 조정하거나, 데이터를 출력하기 위해 출력 스트림을 준비할 때 이 객체를 사용한다.

|  |  |
| --- | --- |
| 메서드 | 설명 |
| getContentType() | 출력할 데이터의 인코딩 형식과 문자 집합을 지정한다.  이렇게 클라이언트에게 출력할 데이터의 정보를 알려주어야 클라이언트는 그 형식에 맞추어 올바르게 화면에 출력(Rendering)할 수 있다. 예) response.setContentType(“text/plain”);  Response.setContentType(“text/plain;chartset=UTF-8”); |
| setCharacterEncoding() | 출력할 문자 집합을 지정한다.  예) 문자집합을 UTF-8 로 설정  response.setCharacterEncoding(“UTF-8”); 문자집합을 지정하지 않으면 모든 문자를 영어로 간주하고 한글은 ?? 문자로 변환되어 출력된다. |
| getWriter() | 출력 스크림 객체를 반환  동영상, 이미지 같은 바이너리 데이터를 출력하고 싶을 때는 getOutputStream()을 사용해야한다.  예) PrintWriter writer = response.getWriter();  사용전에 반드시 문자집합을 지정해야한다,( getContentType, setCharacterEncoding ) |

### CalculatorServlet 배치 및 실행

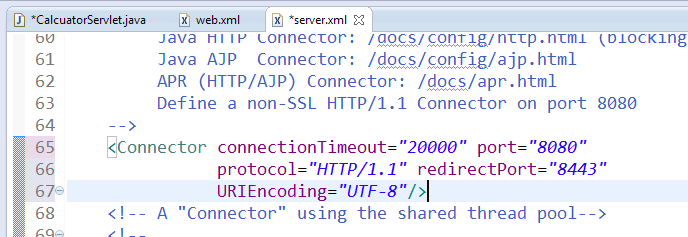


실행 URL에 a,b 매개변수를 작성하여 실행하면 아래와 같은 결과가 출력되는 것을 확인할 수 있다.



### GET 요청으로 넘어온 매개변수 값의 인코딩 설정

GET 요청은 매개변수 갓이 URI에 포함되기 때문에 setCharactoreEncoding()으로는 문자 집합을 설정할 수 없다. 따라서 서블릿 컨테이너에서 권고하는 지침에 따라 URI의 인코딩 형식을 섲렁해야 한다.

* 톱캣에서는 톰캣 실행 환경 폴더에서 server.xml 파일에 아래와 같이 입력한다.
* 

### @WebServlet 애노테이션을 이용한 서블릿 배치 정보 설정

* Servlet 3.0 부터는 애노테이션으로 서블릿 배치 정보를 설정할 수 있다.
* web.xml 파일의 서블릿 매핑을 주석처리하고 아래와 같이 CalcuatorServler 클래스에 애노테이션을 지정하여도 동일한 URL 연결을 해줄 수 있다.

### @WebServlet 애노테이션의 주요 속성

|  |  |
| --- | --- |
| 메서드 | 설명 |
| name | * 서블릿의 이름을 설정하는 속성, 기본값은 빈 문자열이다.   예) @Webservlet(name=”서블릿이름”) |
| urlPatterns | 서블릿의 URL 목록을 설정하는 속성  속성값으로 String 배열이 온다. 기본값은 빈 배열({})이다.  에) @WebServlet(urlPatterns=”/calc”)  @WebServlet(urlPatterns={“/calc”})  @WebServlet(urlPatterns={“/calc”, “/calc.do”, “/calculator.action”}) |
| value | urlPatterns와 같은 용도, 애노테이션 문법에서는 속성 이름이 ‘value’인 경우 이름 없이 값만 설정할 수 있다.  예) @WebServlet(value=”/calc”)  @WebServlet(”/calc”)  만약 value 속성 이외 다른 속성도 함께 사용한다면 value 속성의 이름을 생략할 수 없다.  예) @WebServlet(value=”/calc”, name=”Calculator”) 🡪 가능  @WebServlet(”/calc”, name=”Calculator”) 🡪 불가능 |