**Servlet&JSP 개념정리**

**HTTP : HyperText Transfer Protocol**

**WAS : Web Application Server** ( Web Server + Web Container )

Web Server - http service ( 정적인 콘텐츠 )

Web Container - Servlet&JSP 실행환경을 제공

**Web Browser** :  http 에 의거 http header 정보를  해석하고,

body 부분의 html(javascript,css) 를 해석해 웹화면을 제공

**HTML** : HyperText Markup Language

웹 페이지의 정적 컨텐츠를 담당

**JavaScript** : 웹 페이지의 행위(동적컨텐츠)를 담당 (최근에는 서버 사이드에서도 역할을 한다)

**CSS** : 웹 페이지의 디자인 스타일을 담당

web.xml: DD - Deployment Descriptor ( 배포 기술서 )

웹 어플리케이션의 설정정보를 기술

-> WAS 가 해당 웹어플리케이션 처음 실행할 때

해당 파일(web.xml)을 읽어서 로딩한다

sample code )특정 서블릿에 대한 web.xml 설정

<servlet>

<servlet-name>hello</servlet-name>

<servlet-class>packagename.HelloServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>hello</servlet-name>

<url-pattern>/hello</url-pattern>

</servlet-mapping>

**Web Container ( Servlet Container )** :

웹 서버의 컴포넌트 중 하나로 자바[서블릿](https://ko.wikipedia.org/wiki/%EC%84%9C%EB%B8%94%EB%A6%BF)과 상호작용한다. 웹 컨테이너는 서블릿의 생명주기를 관리하고,[URL](https://ko.wikipedia.org/wiki/URL)과 특정 서블릿을 맵핑하며 URL 요청이 올바른 접근 권한을 갖도록 보장한다.

아파치 톰캣 ( Apache Tomcat ) 은 서블릿 컨테이너로 오픈소스 웹 컨테이너이다.

**Servlet** : 자바 웹 어플리케이션을 위한 기술

Model2 Architecture (MVC Pattern) 에서는 Controller 영역 담당

**JSP** :  자바 서버 페이지

동적 웹페이지 개발을 위한 자바 기술

HTML 내에 자바 코드를 삽입하는 형식으로 개발한다

웹컨테이너에 의해 서블릿으로 변환되어 실행된다

Model2 Architecture (MVC Pattern) 에서는 View 영역의 주 기술이 된다

**HTTP Request Method( 요청 방식 )**

1. Get 요청방식:  정보 조회 용도 , url 에 노출 , 주소 직접 입력 , 대용량 데이터 전송에는 적합하지 않다

Post 요청방식: 정보 전송 용도, url에 노출 x , ( 로그인 , 회원 가입 , 대용량 데이터 )

HTTP status

- 404  not found : 요청 자료가 서버에 존재하지 않을 때

- 405 not allowed  : 요청 메서드를 처리할 자원이 없을 때

- HTTP Status 500 : Internal Server Error 서버 에러 및 예외,컴파일

서블릿 라이프 사이클 (Servlet Life Cycle)

라이프사이클 관련 주요 메서드 ( Servlet Interface 의 abstract method )

1) init() : 해당 서블릿에 대한 요청시 단 한번만 실행

2) service() : 클라이언트 요청시마다 매번 실행

3) destroy() : 웹 컨테이너 종료전(서비스 해제전) 한번 실행

ex) LoginServlet 으로 클라이언트 100명이 접속하여 로그인했다.

LoginServlet 객체 생성 : 1번

LoginServlet init() 실행 : 1번

LoginServlet service() 실행 : 100번

LoginServlet destroy() 실행 : 서비스중이면 0번

**ServletConfig** :

서블릿 객체 당 하나 생성 , 웹 컨테이너에 의해 init() 실행시 주입된다.

해당 서블릿이 실행되는데 필요한 설정정보

ex) Spring Famework에서 제공하는 서블릿인 DispatcherServlet은 자신의 설정파일(spring 설정) 위치를 필요로 한다. 이 때 개발자는 ServletConfig의 init param 즉 초기 파라미터를 이용해 spring 설정파일 위치를 ServletCofig 객체에 저장하여 전달할 수 있다.

**servletContext** : 웹어플리케이션 당 하나 생성되는 객체

웹 어플리케이션 내에 있는 모든 서블릿과 jsp들이 공유하는 자원

ServletContext 객체를 사용하기 위해서 개별 Servlet과 jsp 들은 ServletConfig 객체가 존재해야 한다.

ex) 사원 (서블릿)들은 각각 사원등(servletconfig)을 발급받는다.

이 발급 받은 사원증이 있어야만 사내 카페와 체육 시설, 인트라넷(ServletContext)을 이용할 수 있다.

**servletContextListener :**

웹 어플리 케이션 최소 실행과 종료 시점의 이벤트를 감지하여

최초 실행시 contextInitalized() 메서드를 실행하고

종료 직전 contextDestroyed() 메서드를 실행하게 한다.

contextInitalized() : 웹어플리케이션 차원의 초기 작업을 명시

contextDestroyed() :  웹어플리케이션 종료 전 수행해야 할 작업을 명시

JSP LifeCycle

- JSP LifrCycle은 서블릿 라이프 사이클과 동일하다

JSP는 웹컨테이너에 의해 HttpServlet의 자식 클래스로 변환되기 때문이다

JSP 주석 : <%-- JSP 주석은 브라우저에 노출되지 않는다. --%>

HTML 주석 : <!-- ~~ -->

CSS 주석 : /\* ~~ \*/

javascript 주석 : //~~ or /\* ~~ \*/

-JSP 기본 문법

<% %> scriptlet : jsp가 java 로 생성될 때 service() 메서드 내에 삽입

<%! %> declaration : jsp가 java로 생성될 때 인스턴스 변수 또는 메서드로 변환

<%= %> expression : jsp가 java로 생성될 때 service() 메서드 내에서 out.print()로 삽입

<%@ %> directive : 웹컨테이너에 알려줄 jsp 정보를 기술 ex) import or errorPage or session 등..

**Singleton Design Pattern ( ex - RequestMapping , 인사부장 )**

: 시스템에 객체를 단 한번 생성해서 공유해 사용하고자 할 때 적용하는 디자인패턴

how? 외부에서 객체 생성할 수 없게 생성자에 private 을 명시

static 변수 초기화 시점에 자신의 객체를 생성

static 메서드로 한번 생성된 객체를 공유한다.

ex) 회원수 조회.jsp ----new MemberDAO()--> MemberDAO()

검색.jsp ----new MemberDAO()--> 등

위와 같이 동일한 객체를 반복해서 생성해 사용하므로 메모라낭비-->

이 경우 Singleton Pattern 을 적용해 시스템 상에서 단 한번 객체를 생성해

여러 곳(jsp) 에서 사용하게 한다.

이 후 Spring Framework 에서 객체 운영의 기본 방식이 Singleton 으로 운용된다.

class MemberDAO(){

private static MemberDAO instance;

private MemberDAO throws Exception{

}

public static MemberDAO getInstance(){

if(instance==null)

instance=new MemberDAO();

return instance;

}

}

**Model2 Archiecture ( MVC pattern )**

: 웹 어플리케이션 설계 방식 중 하나

MVC Design pattern을 근간으로 설계한다.

기존 Model 설계방식과의 차이점은 프리젠테이션 계층이 view 영역과 Conroller 영역으로 분할

Model : 데이터베이스 연동과 비즈니스 로직을 담당

-> 주 기술은 자바

View : 클라이언트에게 응답할 동적인 화면을 담당

-> 주 기술은 jsp

Controller : 클라이언트의 요청을 분석, Model과 연동하여 연동결과를 view에게 전달하는 역할을 담당

-> 주 기술은 servlet

**java web application 이동 방식**

**1) forward 방식**

: client는 이동 여부를 알 수 없고, 웹 컨테이너 상에서 이동된다. 기존 request 와 response 이 유지

forward 방식에서는 주로 request의 setAttribute(name,value) 를 통해

Controller 가 view 에 정보를 공유한다.

사용자가 재요청을 할 경우 ( 새로고침 )

기존 request가 유지되므로 재동작을 한다. (정보 생성, 수정 삭제에는 적합하지 않다.)

2) redirect 방식

: 서버(controller)에서 클라이언트에게 특정 url로 이동하여 응답받을 것을 지정하는 방식

forward와 다르게 url이 변경되고 request와 response는 유지되지 않는다.

(정보 생성, 수정, 삭제시 용이하다.)

Session 과 Cookie

: 사용자 상태 정보를 일정 조건동안 유지 ,

Http는 stateless 한 특성을 가진다. 즉 사용자에 대한 상태정보를 유지하지 않는다.

Seesion과 Cookie는 이러한 Http적 특성을 보완하기 위한 기술이다.

1) Cookie: 사용자 상태 정보를 클라이언트 측에 저장

상태 정보를 클라이언트 측에 저장하기에 데이터 타입 및 용량이 제안이 있다.

쿠키 생성 후 전송시에 유효시간을 정할 수 있다.

ex - cookie.setMaxAge(60)초 동안 유지

만약 유효시간을 정하지 않으면 브라우저 실행시에만 유지

참고 ) 쿠키 삭제는 cookie.setMaxAge(0) 을 실행하면 된다.

2) HttpSession: 사용자 상태 정보를 서버  측에 저장

상태 정보를 서버측에 저장하기에 데이터 타입 및 용량의 제한이 없다.

request.getSession() or request.getSession(true)

: 기존 세션이 있으면 세션을 반환하고 없으면 새로 세션을 생성

request.getSession(false)

: 기존 세션이 있으면 기존 세션을 반환하고

없으면 null 을 반환

세션 관련 주요 메서드

session.setAttribute(name , value) : 세션에 특정 정보를 저장

ex ) 로그인 인증시 아이디와 패스워드가 데이터베이스 정보와 일치하면 회원정보를 session에 저장할 때 위 메서드를 사용한다.

session.getAttribute(name) : Object

세션에 저장된 특정 정보를 반환받을 때 사용한다.

session.removeAttribute(name) : 세션에 저장된 특정 정보를 삭제할 때 사용

session.invalidate() : 세션 무효화, 즉 사용하지 못하도록 한다.

ex) 로그아웃 사용할 때 사용

HttpSession 실행 흐름

WAS(Web Container)

client browser ------- request ------->  if(아이디, 패스워드 일치하면)

request.getSession() 실행

세션이 있으면 기존세션 반환, 없으면 새로 생성한다.

서버상에 저장되어 있는 세션 정보와

클라이언트 쿠키에 있는 세션 아이디 정보를 비교해

기존 세션 존재 유무를 판단한다.

먄약 처음 요청(로그인)일 경우 즉 세션이 없을 경우에는 클라 이언트 측으로 발급한 세션 아이디 정보를 쿠키로 전송시킨다.

만약 로그인 작업이면 이 시점에 database로 부터 조회한 회 원정보를 session 에 setAttribute(name,value) 메서드로 할 댕해야 한다.

<------- response ------------ 처음으로 세션을 생성했을 때는

위 설명대로 세션아이디를 쿠키로 전송

인증받은 회원만 접속할 수 있는 서비스 (ex - 카페 or 매일 )

WAS(Web Container)

HttpSession session = request.getSession(false);

if(session!=null){ // 기존 세션이 없으면 null 반환

Member member = (Member)session.getAttribute("mvo");

if(member!=null){ // 로그인시 할당한 인증정보

카페 or 메일 서비스를 제공

}

}