2024.12.31.화

<스프링 MVC 1편 – 백엔드 웹 개발 핵심 기술>

섹션1. 소개

1. 강의 소개

1) 스프링MVC – 가장 인기 있는 웹 프레임워크

2) 백엔드 웹 기술을 학습하기 어려운 이유

- HTTP. HTTP에 대한 기반 지식이 약하면 웹 기술을 깊이 있게 학습하기 어려움

- 자바 백엔드 웹개발 역사 20년이나 됨. 오래됨.

- 스프링MVC. 너무 많은 기능을 제공함. 방대하고 학습할 내용 많음. 기본 구조는 변하지 않음. 기본 구조 확실히 이해하는 게 매우 중요.

3) 강의 방향

- 자바 웹 기술의 시작부터 최신 실무 기술까지

- MVC 프레임워크를 직접 코드로 개발

4) 강의 구성

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명- 스프링 MVC 1편 – 백엔드 웹 개발 핵심 기술

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명- 스프링 MVC 2편 – 백엔드 웹 개발 활용 기술

5) 정리

백엔드 개발자를 위한 모든 웹 기술 기초부터 완성까지

스프링MVC 핵심 원리와 구조 이해로 더 깊이 있는 백엔드 개발자로 성장

섹션2. 웹 애플리케이션 이해

1. 웹 서버, 웹 애플리케이션 서버

1) 웹 – HTTP 기반

웹 브라우저(클라이언트)에서 URL을 치면 인터넷을 통해 서버에 접근. 서버에서는 HTML을 만들어서 클라이언트에 전달. 웹 브라우저가 HTML 해석. 서로 전송 응답할 때 HTTP를 기반으로 동작함

2) 모든 것이 HTTP

**H**yper**T**ext **T**ransfer **P**rotocol

3) 웹 서버(Web Server)

정적 리소스 : 특정 폴더에 HTML, CSS, JS EMD 파일들을 두면 파일들을 서빙 해주는 것.

4) 웹 애플리케이션 서버

보통 와스라고 함

5) 웹서버, 와스 차이

6) 웹 시스템 구성 – WAS, DB

정적 리소스는 쌈. 애플리케이션 로직은 비쌈.

와스는 애플리케이션 로직 전담 가능

2. 서블릿

1) HTML Form 데이터 전송

전송 버튼 누르면 웹 브라우저가 요청 HTTP 메시지 만듦

2) 서버에서 처리해야 하는 업무

서블릿이 비즈니스 로직 실행 외에 나머지 작업 다 해줌

3) 서블릿

HTTP 스펙을 어느 정도 알아야함

4) 서블릿 컨테이너

싱글톤 – 객체를 하나만 생성해놓고 공유해서 쓰는 것

3. 동시 요청 – 멀티 쓰레드

1) 쓰레드

2) 쓰레드 풀

톰캣맥스커넥션

nGrider – 네이버에서 만든 성능 테스트 도구

3) WAS의 멀티 쓰레드 지원

2025.1.1.수

4. HTML, HTTP API, CSR, SSR

1) 어디까지 알아야 나요?

- 백엔드 : JSP는 거의 사장됐고, 타임리프를 공부하는 것이 나음

- SSR은 하루~3일 공부하면 됨

- 풀스택으로 다 잘하는 사람 손에 꼽음

5. 자바 백엔드 웹 기술 역사

1) 과거 기술

- 스트럿츠를 앞에 깔고 뒤에는 스프링. 스프링 MVC가 그렇게 좋지 않았음

2) 현재 사용 기술

3) 최신 기술

- 웹 서블릿: 스프링MVC는 서블릿 기반으로 동작함. 멀티쓰레드

- 웹 리액티브: 완전 최신 기술

4) 스프링 웹 플럭스

- 최소 쓰레드로 최대 성능: CPU코어로 4개가 있음. 쓰레드 수를 CPU코어나 +1개로 맞춤. 고효율로 쓰레드를 CPU코어 개수에 맞춰 놨기 때문에 쓰레드 계속 돌아갈 수 있음.

- 함수형 스타일로 개발: 동시에 상품조회, 주문서버 조회 등등 여러 개 해야 할 때 자바코드로 하면 쉽지 않음. 함수형으로 하면 깔끔해짐

- 서블릿 기술 사용X: Netty라는 웹 프레임워크를 가지고 구현함

- RDB: 관계형 데이터베이스 쓰기 어려움

- 성능이 굉장히 중요하고 여러 개 API 호출하고 되게 복잡할 때 사용할 만 함

5) 자바 뷰 템플릿 역사

2025.1.5.일

**스프링 MVC 1편 – 백엦드 웹 개발 핵심 기술（프로젝트 생성)**

서블릿은 스프링이 필요 없지만, 스프링 부트가 톰캣 서버를 내장하고 있기때문에 스프링 부트 사용

war를 써야 jsp를 쓸 수 있음.

war는 별도의 톰캣을 쓰거나 내장 톰캣을 쓸 때 사용함.

dependency 추가

spring web – 아파치톰캣을 쓰고, 내장서버 사용 가능.

lombok

롬복 사용하기 위해서

lombok plugin 설치 -> endable annotaion processing 체크

포스트맨 다운로드

Hello 서블릿

스프링과 서블릿은 전혀 관련 없지만, 스프링에서 환경설정하는 것이 좋음.

톰캣을 깔지 않아도 됨.

서블릿은 톰캣같은 웹애플리케이션 서버를 직접 설치하고, 그 위에 서블릿 코드를 ᅟᅳᆯ래스 파일로 빌드해서 올린 다음, 톰캣 서버를 실행하면 된다.

@ServletComponentScan ->스프링 annotation 패키지를 포함하여 서블릿을 찾아서 자동으로 등록하고 실행할 수 있도록 해줌.

// 서블릿 자동등록

서블릿 클래스는 HttpServlet을 상속해야함

@WebServlet(name = “helloServlet”, urlPattern = “/hello”)

HttpServlet 상속 클래스는 service라는 메소드를 Override해야함.

HttpServletRequest - > interface 톰캣이나 netty등에서 표준 스펙에 맞게 구현함.

서블릿이 없으면 http 내용을 다 파싱해서 query param값을 가져와야함.

서블릿을 사용하면 request.getParameter로 query param을 쉽게 가져올 수 있음.

응답.

response.setContentType(“text/plain”); -> HTTP헤더에 들어감.

response.setCharacterEncoding(“UTF-8”); -> HTTP헤더에 들어감.

response.getWritet().write(“hello”); -> HTTP 메시지 바디에 들어감.

request header :웹브라우저가 보내는 것.

http 스펙에 맞춰서 직접 http를 만들려면 오래 걸리지만 servlet을 사용하면 편하게 http통신을 할 수 있음.

url 요청이 들어오면 자동으로 servlet의 service 메소드를 실행해준다.

main – resource - > application.properties

logging.level.org.apache.cotoye.http11=trace 추가 (스프링 3.2이상 사용시)

: http 정보를 로그에서 다 확인할 수 있음.

개발서버에 위 설정을 적용하는 것을 추천. 운영서버에 하면 성능이 저하될 수 있다.

스프링 부트가 내장 톰캣 서버를 띄움. 톰캣 서버에 있는 서블릿 컨테이너가 서블릿을 다 생성해줌.

index.html – 웰컴 페이지(도메인으로 접근했을 때 첫 화면)

HttpServletRequest

HTTP 요청 메시지를 개발자가 직접 파싱해서 사용해도 되지만, 매우 불편하다. 서블릿은 개발자가 HTTP 요청 메시지를 편리하게 사용할 수 있도록 개발자 대신에 HTTP요청 메시지를 파싱하고, HttpServletRequest 객체에 담아서 제공한다.

HttpServletRequest는

- 임시 저장소 기능을 제공함.

요청이 들어오고, 요청이 나갈 때까지 유지되는 저장소를 제공함.

request.getAttribute(“”), request.setAttribute(“”)

- 세션 관리 기능.

request.getSession()

HTTP RequestHeader

startline 정보 :

request.getMethod();

request.getProtocol();

request.getScheme();

request.getRequestURL();

request,getRequestURI();

equest.getQueryString();

request.isSecure();

ctrl +　alt + shift + t를 입력하면 메소드를 추출해서 만들어줌.

private void printHeaders(HttpServletRequest request)｛

Enumeration<String> headerNames =rrequest.getHeaderNames();

while (headerNames.hasMoreElements()) {

String headerName = headerNames.getNextElement();

System.out.println(“headerName : ” + headerName);

}

｝

// 요즘 스타일

request.getHeaderNames().asIterater().forEachRemaning(headerName -> System.out.println(“headerName :＂+ hearderName);

contentType은 http 메시지바디에 내용이 있어야 넘어옴. (get방식은 바디에 내용이 없음)

http 기타 정보 :　네트워크 맺어지면서 생기는 정보