22.12.27

📌웹 시스템 구성

- Web server
  - 주로 정적인 리소스(html, jpg..)를 요청 받아 반환하는 서버
- WAS(Web Application Server)
  - 정적인 리소스를 반환할 수 있고, 비지니스 로직도 수행할 수 있는 서버
- 웹 시스템 구성

  1) WAS + DB
  - WAS가 정적 리소스에 대한 요청까지 처리 → 서버 과부하 발생 가능성 높음
  - WAS 장애 시 오류 화면도 노출되지 않음

  2) Web Server + WAS + DB
  - 요청 리소스 관리를 효율적으로 할 수 있음
  - WAS 장애 시 오류 화면 노출 가능

📌Servlet Container

- Tomcat 처럼 서블릿을 지원하는 WAS를 서블릿 컨테이너라고 함.
- 서블릿은 비지니스 로직을 제외한 모든 웹 APP 로직(ex 소켓 연결, 요청 HTTP 메시지 파싱, 응답 HTTP 메시지 생성 등)을 수행.
- HTTP 요청을 통해 urlpatterns의 URL이 호출되면 Web.xml 파일에서 매핑된 서블릿 파일을 찾아 서블릿 컨테이너에 인스턴스 생성 후 싱글톤으로 관리. 서블릿 컨테이너 종료 시 함께 종료.
- 개발자는 Http 스펙을 편리하게 사용할 수 있다.
- **WAS의 HTTP 요청 수신 및 응답 발신**

  0) WAS는 HTTP 요청에 대해 수신할 수 있는 **socket**을 열고 listen

  1) WAS는 클라이언트로 부터 HTTP 메시지로 이루어진 요청을 수신

  3) WAS는 요청을 파싱하고, 이를 **HttpServletRequest, HttpServletResponse** 객체에 저장

4) WAS는 **스레드 풀**로부터 요청을 처리할 스레드를 선택

5) 선택된 스레드를 통해 url 매핑된 서블릿 인스턴스의 로직을 수행

6) 수행한 로직의 결과를 Response 객체에 저장

7) WAS는 Response 객체를 HTTP 응답 메시지로 변환함.

8) WAS는 클라이언트로 메시지를 송신

📌스레드 풀

- 요청 마다 스레드 생성 시 → 스레드 생성 비용 큼, 높은 컨텍스트 스위칭 횟수, 하드웨어의 제한 사항 → 스레드 풀을 사용
- 특징
  - 필요한 스레드를 미리 생성하여 관리
  - 필요할 때 꺼내 쓰고, 다 쓰면 반납
  - 모두 사용 중일 때 → 기다리는 요청들을 거절하거나 대기 숫자를 지정할 수 있다.
- 장점
  - 생성, 종료 비용을 절약하고, 빠른 응답 시간을 기대할 수 있음
  - 스레드의 수를 한정하여, 요청이 안전하게 처리될 수 있도록 함.
- WAS의 주요 튜닝 포인트는 최대 스레드 수(max thread)

  → 성능 테스트가 관건. ngrinder 사용 추천
- WAS 멀티 스레드 지원의 핵심
  - 멀티 스레드에 대한 부분은 WAS가 처리
  - 개발자가 멀티 스레드 관련 코드를 신경 쓰지 않아도 됨
  - 멀티 스레드 환경이므로 싱글톤 객체(서블릿, 스프링 빈)는 조심해서 사용하기