01. 스프링 부트 살펴보기

스프링 부트 소개

- 스프링 부트 (spring boot) 는 **스프링을 기반**으로 실무 환경에서 사용 가능한 수준의 **독립실행형 애플리케이션을 복잡한 고민 없이 빠르게 작성**할 수 있게 도와주는 여러가지 도구의 모음이다.
    - 스프링 부트로 스프링 애플리케이션을 만든다.
    - 빠른 구현이 가능하고 점점 커지는 서비스에 필요한 수준을 제공한다.
    - 스프링 != 스프링 부트

스프링 부트의 핵심 목표

- 매우 빠르고 광범위한 영역의 스프링 개발 환경을 제공
- 강한 주장을 가지고 즉시 적용 가능한 기술 조합을 제공하면서, 필요에 따라 원하는 방식으로 손쉽게 변형 가능
    - 강한 주장 → 스프링 부트가 '이런이런 걸 사용해서 이런 구성으로 사용하면 돼' 라고 결정해버림
    - 원한다면 변경해서 사용할 수 있음
- 프로젝트에서 필요로 하는 다양한 비기능적인 기술(내장형 서버, 보안, 메트릭, 상태 체크, 외부 설정 방식 등) 제공
- 코드 생성이나 XML 설정을 필요로 하지 않음
    - 스프링에서는 XML을 작성해야 했음
    - 코드 생성 방식으로 기술을 제공하지 않음

스프링 부트의 역사

- containerless web application architecture를 지원해달라는 요청으로 부터 시작 (2012) → 꾸준히 발전

Containerless (컨테이너리스 웹 애플리케이션 아키텍처)

- 여기서 말하는 container란??
    - web container?
        - 서버에 있는 web component(하나의 서비스를 담당하는, ex. 회원가입 서비스)를 관리하는 역할
        - web component를 메모리에 올리고 (ex. new로 인스턴스 생성) 서비스가 되는 동안 메모리에서 관리해주는 역할
        - life cycle 관리
        - 여러 web component를 관리
        - web client가 요청한 작업을 룰을 따라 어느 web component에 할당할 것인지 결정(라우팅, 맵핑)
- 자바에서는,,,
    - web container == Servlet Container
        - tomcat(servlet container)
    - web component == servlet

- spring container는?
    - servlet container 뒤에 존재
    - 여러 개의 bean(component)를 가지고 있는 spring container
    - servlet을 통해서 web으로 들어온 요청을 받아 spring container에게 넘겨줌 -> 어느 bean이 요청을 처리하게끔 할당

- spring container가 servlet container를 대체하면 안될까?
    - 놉, java의 표준 web 기술을 사용하려면 servlet container가 필요함

servlet container 설정의 번거로움

- spring container만 개발해서 띄우고 싶은데 servlet container도 개발해야해
    - XML파일 작성 불가피
    - 초반 작업임
- folder 구조도 맞춰야함
- servlet container는 독립적인 서버 프로그램이기 때문에 tomcat 같은걸 환경에 맞게 설치하고 실행시켜야함
    - war로 압축된 servlet container를 배포해야함
- servlet container의 설정도 많음
    - logging 등

이것들이 초반 설정 작업들이고 개발하는 동안 계속 신경써야되는 것들이 아님 -> 매번 번거로움

게다가 tomcat 말고 다른 웹서버를 쓰면 더 번거로워짐

servlet container가 없는 web architecture를 만들어줘

따라서 ~~servlet container가 없는~~ web architecture를 만들어줘! -> sevlet container를 신경쓰지 않게 해줘! == spring boot

- 독립실행형 애플리케이션 (standalone application)
    - servlet container를 초기에 띄우는 작업이 필요하지 않아?
    - main method를 실행하면 servlet container와 관련된 모든 작업들이 실행되는 것 (= containerless)

Opinionated (자기 주장이 강한!)

내가 다 정해줄게, 넌 개발만 해

스프링 프레임워크의 설계 철학

- 극단적인 유연함 추구
    - 다양한 환경에 적용하기 위해서
- 다양한 관점을 수용
    - 다양한 기술의 관점을 수용
    - 다양한 기술의 장점들을 수용
- Not opinionated
    - 스프링 부트와 달리,,
    - 너네가 선택하는 선택지를 포용해줄게
- 수많은 선택지를 다 포용
- 하지만... 선택하는 고민을 결국 개발자가 다 해야된다는 것

스프링 부트의 설계 철학

- Opinionated
    - 기술적 고민은 하지 않도록 결정해줄테니
        - 지금까지의 best practice로
            - 업계에서 검증된 스프링 생태계 프로젝트, 표준 자바 기술, 오픈소스 기술의 종류와 의존 관계, 사용 버전을 정해줌
            - 각 기술을 스프링에 적용하는 방식(DI 구성)과 디폴트 설정 값 제공
    - 너네는 애플리케이션을 빠르게 개발해

- 일단 정해주는 대로 빠르게 개발하고 고민은 나중에
- 스프링을 잘 활용하는 뛰어난 방법을 제공

유연한 확장

- 스프링 부트에 내장된 디폴트 구성을 커스터마이징하는 매우 자연스럽고 유연한 방법 제공
- 스프링 부트가 스프링을 사용하는 방식을 이해한다면 언제라도 스프링 부트를 제거하고 원하는 방식으로 재구성 가능
- 스프링 부트처럼 기술과 구서을 간편하게 제공하는 나만의 모듈 작성 가능

스프링 부트의 이해

스프링 부트를 이해하게 되면

- 스프링 부트가 스프링의 기술을 어떻게 활용하는지 배우고 응용할 수 있다.
- 스프링 부트가 선택한 기술, 자동으로 만들어주는 구성, 디폴트 설정이 어떤 것인지 확인할 수 있어야 한다.
- 필요할 때 부트의 기본 구성을 수정하거나, 확장할 수 있다.
- 나만의 스프링 부트 모듈을 만들어 활용할 수 있다.

강의 목표

- 스프링 부트로 만든 스프링 애플리케이션의 기술과 구성 정보를 직접 확인할 수 있다.
- 적용 가능한 설정 항목을 파악할 수 있다.
- 직접 만든 빈 구성 정보를 적용하고, 그에 따른 변화를 분석할 수 있다.
- 스프링 부트의 기술을 꼼꼼히 살펴볼 수 있다.
