

스프링은 여러 가지 기술들의 복합체이다.

스프링 데이터 : nosql, sql 데이터 베이스든 crud는 비슷한데 이를 편리하게 사용할 수 있도록 도와줌. -> 스프링 데이터 jpa를 제일 많이 사용

스프링 세션 : 세션 관리

스프링 시큐리티 : 보안과 관련됨.

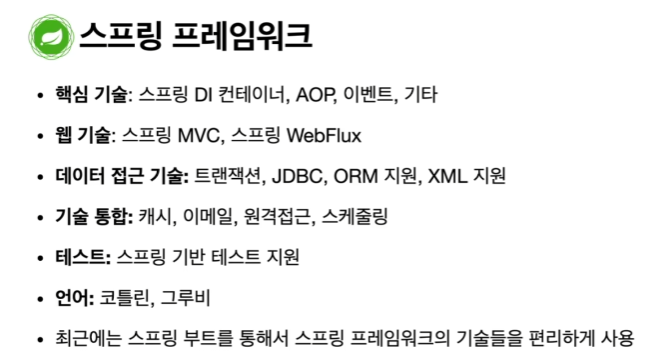
스프링 rest docs : api문서화를 편하게 해줌

스프링 배치 : 배치 처리에 특화

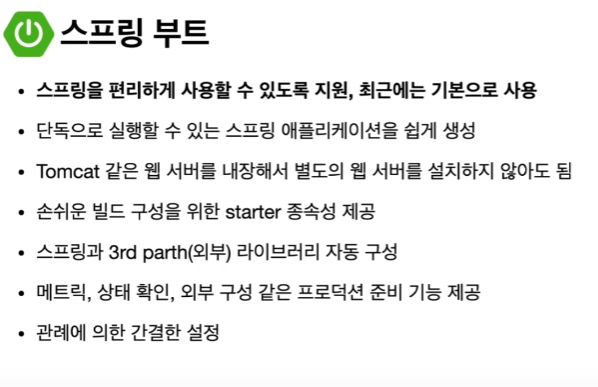
스프링 클라우드 : 클라우드와 관련

spring.io -> project -> overview

가장 핵심은 스프링 프레임워크이고, 모든 기술을 편리하게 사용할 수 있게 도와주는 게 스프링 부트.



스프링 DI 컨테이너, AOP, 스프링 MVC, 트랜잭션, JDBC, 테스트, 언어 등 복합적인 기능 제공.



톰캣 같은 웹서버를 내장해서 단독으로 실행할 수 있는 스프링 애플리케이션을 쉽게 생성할 수 있다. 예전에는 스프링을 가지고 웹서비스를 만들 때, 빌드를 한 다음 톰캣 서버를 별도로 설치하고, 톰캣 서버의 특정 위치에 스프링 프로젝트로 빌드된 파일을 교체해주는 방식을 사용.

현재는 빌드하고 서버 띄우는 것까지 제공해줌.

예전에는 여러 라이브러리를 추가하는 작업을 해줘야했는데, starter를 사용하여 특정 기능을 사용할 때 필요한 라이브러리들을 묶음으로 땡겨옴.

스프링과 외부 라이브러리의 자동 구성. 스프링 프레임워크와 외부 라이브러리와의 버전에 따른 비호환 문제를 해결해줌. 알아서 버전을 챙겨서 지원을 해줌. 외부라이브러리의 버전을 크게 걱정하지 않아도 됨.

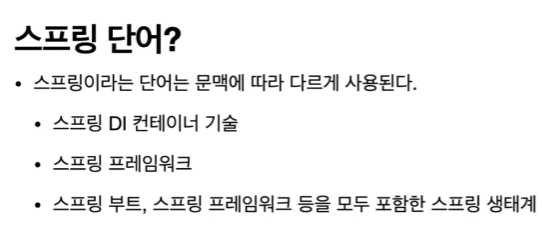
운영환경에서 모니터링하는 것이 중요한데, 스프링 부트가 이를 제공해줌.

스프링 프레임워크만 사용할 때 설정하는 것이 어려워졌는데, 부트를 사용할 땐 기본적인 설정을 간결하게 제공해주고, 커스터마이징도 할 수 있게 해줌.

\*\* 외부 라이브러리 버전 최적화, 웹서버 내장, 설정의 간소화, 모니터링 기능 제공 \*\*

스프링 부트는 스프링 프레임워크와 별도로 사용할 수 있는 것은 아니다.

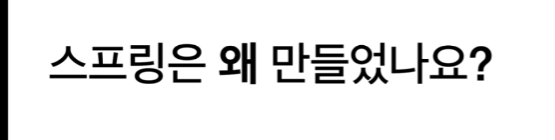
스프링 부트는 스프링의 다양한 기능들을 중간에서 효율적으로 사용할 수 있도록 도와주는 역할을 한다.



스프링 1 : 스프링 빈을 관리하는 di컨테이너 기술을 제공함.

스프링 2 : 스프링 프레임워크

스프링 3 : 스프링이 제공하는 모든 생태계



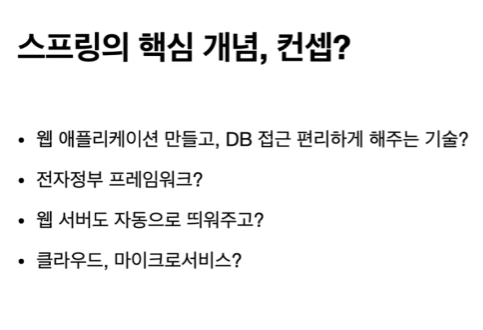
로드존슨이 만들었음.

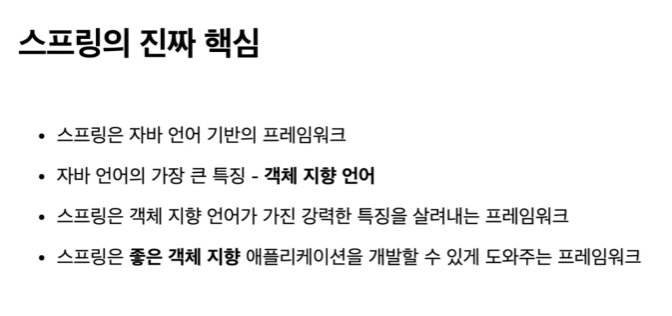
이 기술을 왜 만들었는가? 이 기술의 핵심 컨셉은?

모든 기술의 제작 이유와, 컨셉은 항상 단순하다.

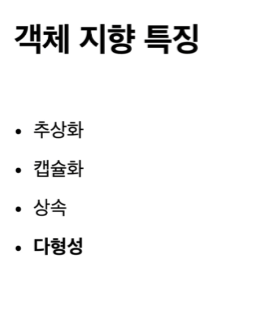
로드존슨의 3만줄의 코드에서 시작함.

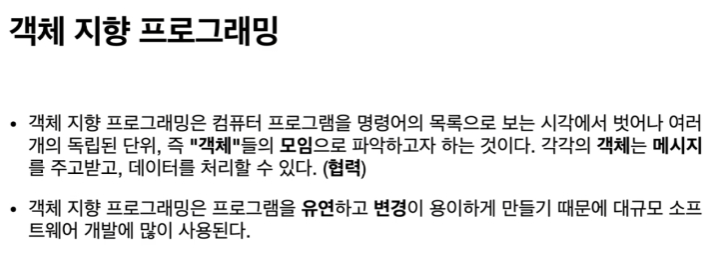
핵심 컨셉을 알아야 나머지 부차적인 기술들도 쉽게 익힐 수 있음.



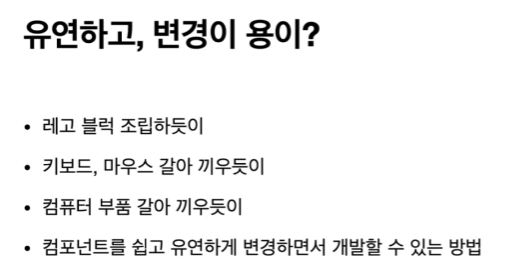


당시 EJB를 사용하면 여러 가지를 상속받으면서 ejb의 의존적으로 코드를 짜게 되므로 코드가 복잡해졌다. 객체 지향이 가진 좋은 장점을 잊어버림. ejb의 스타일이 아니면 개발을 하기 어렵게 됨.

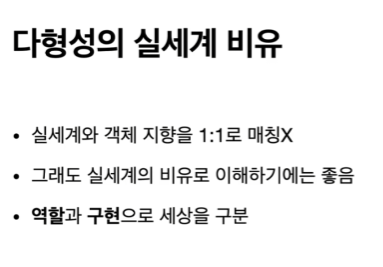




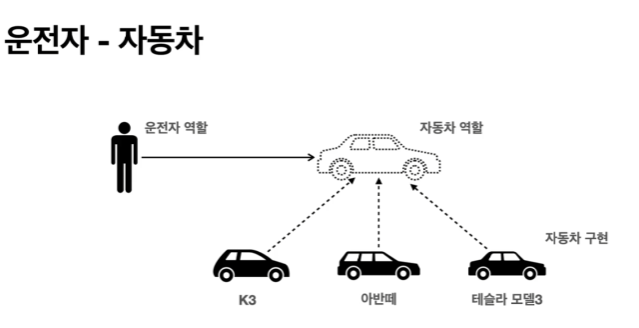
객체 지향의 핵심 : 컴퓨터 프로그램이 절차적인 명령들의 목록으로 보는 것이 아닌, 개체들끼리 메시지를 주고받고, 데이터를 처리하는 여러 개체들간의 상호작용으로 보며, 각 개체들은 서로 유연하게 연결되어있어, 변경이 용이하도록 만들어준다. 대규모 프로그램을 개체들로 부품화하여, 조립하고 각 부품들을 필요에 따라 변경할 수 있는 편의성을 제공해준다.



이를 다형성이라고 말한다.



이세상을 역할(인터페이스) 역할을 구현한(implement) 대상으로 구분함.



운전자는 k3를 타다가 차종을 바꾸더라도, 운전을 할 수 있다. 운전자는 구현체의 구동 및 구현 방식을 모르더라도, 자동차의 역할을 알고 있으면 얼마든지 자동차를 구현한 실제 인스턴스들을 운전할 수 있다. 자동차의 역할과 구현을 구분한 건, 사용자(운전자)의 편의성을 제공하기 위해 구분했다. 완전히 새로운 자동차가 나왔다 하더라도, 기존 자동차 역할에 맞게 구현되어있다면, 얼마든지 사용자가 자동차를 운전할 수 있기에 자동차의 기술을 계속 확장할 수 있다.

클라이언트에게 영향을 주지 않고, 새로운 기능을 제공할 수 있게 된다. 클라이언트는 새로운 걸 배우지 않아도 됨.

