<스프링이란?>

- 스프링은 프레임워크다.

틀 안에서 동작한다. 틀에서 벗어나지 마라! 니 맘대로 만들지말고 틀에 맞춰서 개발해라! 그렇게 하면 너도 나처럼 좋은 프로그램을 만들 수 있을거야! --> 프레임워크

- 스프링은 오픈소스이다.

오픈소스란 소스코드가 공개되어 있는 것! 스프링이 어떻게 만들어졌는지 코드를 볼 수 있다! 그말은 내부를 뜯어 고칠 수도 있다는 것!

공부를 너무 많이해서 내부 깊숙한 것까지 알게되어 불편한 게 생겼을 때 직접 고친 후 contribute도 가능하다!

- 스프링은 IoC 컨테이너를 가진다.

IoC 컨테이너? Inversion of Control – 제어의 역전(주도권이 스프링에게 있다!)

class -> 설계도

object -> 실체화가 가능한 것

instance -> 실체화 된 것

ex) 가구 : object가 아니다! 추상적이기 때문.(abstract class)

의자, 침대 : object이다! 실체화가 가능하기 때문 -> 만들어지는 순간 intance가 된다.

내가 오브젝트를 **직접 new**를 해서 heap이라는 공간에 올리게 되면,

오브젝트를 내(메소드 내부)가 관리한다. 그러나 이걸 다른 메소드에서 사용할 때 많은 다시 new하던가 하는 복잡한 로직이 있기 때문에 Spring에서 내가 만든 class들을 스캔해서 직접 heap메모리에서 관리해주어 사용하기 편리하게 해준다. 내가 원하는 모든 class의 메소드에서 사용할 수 있다. --> 제어의 역전

- 스프링은 DI를 지원한다.

Dependency Injection 의존성 주입. 스프링이 스캔한 나의 class들을 다른 곳에서 사용하는 것을 di라고 한다. IoC와 DI에 의해 프로그래밍이 편해진다!

- 스프링은 엄청나게 많은 필터를 가지고 있다.

검열해주는 것.

스프링 자체 필터 사용 or 스프링의 많은 필터 중 사용되지 않았던 필터를 사용 or 필터 직접 생성하여 사용할 수 있다.

스프링 컨테이너의 필터 -> 인터셉터(AOP)

- 스프링은 엄청나게 많은 어노테이션을 가지고 있다. (리플렉션, 컴파일체킹)

// (주석) - 컴파일러가 무시함.

어노테이션(주석 + 힌트) : 컴파일러가 무시하지 않음.

Animal이란 객체에 run()이라는 메소드가 있다고 가정.

Dog가 Animal을 상속하고 run()을 재정의 하고 싶을 때, 윗 줄에 @override (어노테이션)을 사용하면 컴파일체킹할 때 좋음.

또한 스프링은 어노테이션 객체를 생성하여 사용함.

@Component -> 클래스를 메모리에 로딩

@Autowired -> 로딩된 객체를 해당 변수에 집어넣어

@Component

Class A {

}

-> 스프링이 class를 스캔해서 heap 메모리에 로딩함. 어노테이션으로 약속했기 때문에!

클래스 A를 클래스 B에서 사용하고 싶은 상황

어노테이션을 사용하면 편하게 불러올 수 있다.

Class B {

@Autowired

A a;

}

이때 B 클래스 내부에 어떤애(메서드,필드,어노테이션)가 있는지 분석하고 동작시키는 걸 리플렉션이라고 한다. 리플렉션은 런타임시 발생한다.

- 스프링은 MessageConverter를 가지고 있다. 기본값은 현재 Json이다.

자바 오브젝트와 파이썬 오브젝트가 서로 통신(request or response)하거나 주고받을 때 언어가 다르므로 힘들다. 그래서 그 중간에 Json오브젝트로 바꾸어 서로 주고받는다. 둘다 중간언어로 이해하기 쉬움.

자바 프로그램 -> JSON (request) -> 파이썬 프로그램

자바 프로그램 <- JSON (request) <- 파이썬 프로그램

중간언어(MessageConverter) : xml -> Json

- 스프링은 BufferedReader와 BufferedWriter를 쉽게 사용할 수 있다.

통신(영어권)

한문자 -> 8bit = 2^8 = 256. 256가지의 문자 전송가능!

8bit(1byte)씩 끊어 읽으면 영어권에서는 한 문자씩 받을 수 있다.

한국은 2byte, 중국은 3byte. 나라마다 다르다.

전세계에서 인터넷으로 커뮤니케이션하는 시대에, 자기나라 만의 방식으로 글을 인코딩하게 되면 전세계적인 데이터 통신은 불가능! 따라서 유니코드에서 utf-8 이라는 캐릭터 인코딩을 제공함. 3byte 통신.

자바에서 데이터를 통신할 때 Byte Stream을 통해서 InputStreamReader(바이트를 문자로 바꿔서 전달)로 여러개의 문자를 배열형태로 받아왔는데, 배열은 길이가 정해져있으므로 많이 설정하면 낭비가 되고, 적게 설정하면 문자가 버려지는 문제발생.

따라서 BufferedReader사용. 가변길이의 문자를 받을 수 있다.

JSP에선 데이터를 받을 때 request.getReader() 함수를 사용하면 BufferedReader일을 해줌.

데이터를 쓸 때는 PrintWriter 함수를 사용하면 BufferedWriter 일을 해줌. 문자열로 가변길이의 데이터를 쓸 수 있다.

어노테이션으로 사용가능

@ResponseBody -> BufferedWriter

@RequestBody -> BufferedReader

- 스프링은 계속 발전중이다.

<JPA란?>

- JPA는 Java Persistence API 이다.

Persistence(영속성) : 데이터를 생성한 프로그램의 실행이 종료되더라도 사라지지 않는 데이터의 특성을 의미. 영속성은 파일 시스템, 관계형 데이터베이스 혹은 객체 데이터베이스 등을 활용하여 구현한다.

API : Application Programming Interface

인터페이스 : 상하관계가 존재하는 약속

프로토콜 : 관계가 동등한 약속

수많은 프로토콜로 만들어진 ‘인터넷'

JPA

Java 자바 프로그래밍을 할 때

Persistence 영구적으로 데이터를 저장하기 위해 필요한

Application Programming Interface 인터페이스

- JPA는 ORM 기술이다.

-> Object-Relational Mapping

ORM -> 나의 하인!

자바는 클래스를 통해서 데이터베이스에 있는 테이블을 모델링해야 한다.

Class Team {

int id;

String name; → DB세상에 있는 데이터를 자바세상에 모델링한다!

String year;

}

위 클래스를 실행하면 아래의 데이터베이스 테이블을 자동으로 만들어주는 것을 ORM이라고 한다.

Team 테이블

id int

Name varchar

year varchar

- JPA는 반복적인 CRUD 작업을 생략하게 해준다.

Select, Delete, Update, Insert 등 굉장히 반복적으로 일어나는 일들.

위의 일들을 할 때 자바프로그램에서 DB에 커넥션을 요청함. -> DB가 요청자의 신분을 확인하고 세션열어줌. -> 자바에서 Connection을 가짐. -> 자바가 DB에 쿼리 전송 -> DB에서 Data를 자바프로그램에 응답 -> but, Data type이 서로 달라 자바프로그램은 해당 데이터를 자바 object로 변경해야함 -> 단순한 반복 로직..노가다..굉장히 귀찮음!

이런 일을 줄일 수 있는 것이 JPA!

1. 전송된 쿼리에 대한 응답이 있을 때, 데이터를 받고 받은 데이터를 자바 오브젝트로 바꾸고 연결된 세션을 끊고, 연결된 커넥션을 끊는 이런 모든 과정을 함수 하나로 가능하게 해줌!!

- JPA는 영속성 컨텍스트를 가지고 있다.

영속성 : 데이터를 영구적으로 저장하게 해주는 속성 -> 자바에서는 DB에 저장.

컨텍스트 : Context. 모든 정보를 가지고 있는 것.

길동 -> 영숙 : 난 영숙이 너의 모든 컨텍스트를 가지고 있어. == 영숙이의 모든것들을 알고있다!!

영숙 -> 길동 : 변태.

영속성 컨텍스트는 자바가 DB에 데이터를 저장하고, 가져오는 일련의 모든 과정에 중간에서 관여한다. 자바에서 DB로 바로 가는 게 아니라, 영속성 컨텍스트를 거쳐서 데이터가 처리된다. 자바, 영속성컨텍스트, DB는 서로 동기화 되어있다.

- JPA는 ORM을 통해 DB와 OOP의 불일치성을 해결하기 위한 방법론을 제공한다. (DB는 객체저장 불가능)

DB에선 객체저장이 불가능하므로 foreign key를 사용하지만, 자바에선 객체사용이 가능하므로, CRUD를 할때 JPA가 ORM을 통해 자동으로 객체를 매핑해준다.

OOP(Object-Oriented Programming) - 객체 지향 프로그래밍

- JPA는 OOP의 관점에서 모델링을 할 수 있게 해준다. (상속, 콤포지션, 연관관계)

- 방언 처리가 용이하여 Migration 하기 좋음. 유지보수에도 좋음.

- JPA는 쉽지만 어렵다.