

LeeJin0527 / Algorithm Public

Code

Issues 1

Pull requests

Actions

Projects

Wiki

Security

Insights

Edit

New issue

[Jump to bottom](#)

spring #1

 Open

LeeJin0527 opened this issue now · 0 comments



LeeJin0527 commented now

Spring Framework

기본

Spring이란 무엇인가요?

Spring이란 JAVA 기술들을 더 쉽게 사용할 수 있게 해주는 오픈소스 프레임 워크이다.

Spring, Spring MVC, Spring Boot의 차이점에 대해 각각 설명해 주세요.

spring :개발자는 많은 코드를 작성

spring boot :코드 줄일 수 있음 , 개발 시간을 줄이고 생산성을 높임

의존성 주입이나 버전관리가 편리

spring MVC: Spring 프레임워크에서 제공하는 웹 모듈이다.

개발에 더 많은 시간이 걸림

Spring 버전 별 특성에 대해서 아는대로 답변해 주세요.

Spring 3.2.X

Java 5의 기능(제네릭, 가변 매개변수 등)을 사용하여 개정되었다.

이로 인해서 BeanFactory 등 핵심 API가 업데이트 되었다.

@async 주석을 통해 비동기 메서드 호출을 지원하기 시작했다.

하나의 Spring.jar로 제공하던 것을 여러 Spring 모듈의 jar 파일로 나누어 제공하기 시작했다.

(ex : spring-core, spring-web 등)

SPEL(Spring Expression Language) 가 도입되어 XML 및 Annotation 기반 Bean정의에서 사용할 수 있게 되었다.

이로 인해서 외부 프로퍼티 파일이나 환경변수에서 값을 가져오기 쉬워졌다.

Java 클래스로부터 **@configuration**, **@bean** 등의 Annotation을 사용해서 직접 메타 데이터를 설정하고, DI 지원을 받을 수 있다.

OXM(Object Xml Mapping)을 사용하여 Bean을 XML형태로 관리할 수 있게 되었다.

Rest API 에 대한 지원이 추가되었다.

서버로서는 기존 MVC Framework 레벨에서 Annotation 기반 확장이 추가되었다.

클라이언트로서는 RestTemplate 을 추가해 지원한다.

H2등의 Embedded Database를 지원한다.

2016년 12월 31일부로 개발 및 지원이 종료되었다.

Spring 4.3.X

Java 8 기능을 완전히 지원하기 시작하였다.

Java 6, Java 7 의 고유 기능들에 대해서도 각각 지원한다.

Starter Pack의 등장으로 초기 설정이 보다 용이해졌다.

Groovy 를 통한 Bean 설정이 가능하다.

Core Container 들의 기능 지원이 확대되었다.

예를 들어, Spring Data Repository 를 사용하고 있다면 간단한 구현으로 주입할 수 있다. (**@Autowired** RepositorycustomerRepository)

meta-annotation 지원과 함께 custom-annotation 을 만들 수 있다.

Bean 관리가 더 용이해졌다.

@order 어노테이션을 통해 배열과 리스트 형태의 Bean을 정렬 할 수 있다.

@lazy 어노테이션을 통해 Lazy Injection이 가능하다.

@RestController 등 Web 개발 도구의 지원이 강화되었다.

WebSocket이나 STOMP 등의 프로토콜을 지원하여 양방향 통신이 가능하다.

테스트 환경이 개선되어 Framework 레벨에서 테스트 유틸리티를 지원한다.

(ex. AopTestUtils, ReflectionTestUtils(개선))

2020년 12월 31일부로 개발 및 지원이 종료될 예정이다.

Spring 5.X

전체 프레임워크가 Java 8 을 기반 소스코드로 삼으며, 제네릭과 람다 등을 통해 가독성이 향상 되었다.

JDK 9와도 완벽 호환된다.

Jackson 2.9, Protobuf 3, Reactor 3.1과의 호환 추가

Spring WebFlux 추가, 비동기와 년-블로킹 이벤트 루프 모델 사용 가능

Kotlin 지원

Junit 5 지원

5.0.x 버전은 2020년 10월까지 지원되며, 5.1.x 버전과 5.2.x 버전은 각각 2020년 10월, 2021년 말까지 활발히 개발될 것이다. 5.3.x 버전은 알파버전으로, 2024년까지 지원이 제공된다.

Spring Framework의 생명 주기에 대해서 말해주세요.

스프링 컨테이너 생명주기
GenericXmlApplicationContext를 이용한
스프링 컨테이너 초기화 (생성)
getBean()을 이용한 빈(Bea)객체 사용
close()를 이용한 스프링 컨테이너 종료 (소멸)

빈(Bea)객체 생명주기

스프링 컨테이너 초기화 - (Bea)객체 생성 및 주입
스프링 컨테이너 종료 - (Bea)객체 소멸
빈(Bea)객체의 생명주기는 스프링 컨테이너의 생명주기와 같이 한다.

빈(Bea)객체가 생성되고 소멸될때에 어느 특정한 작업을 할수가있는데
첫번째로는

1. 인터페이스를 구현함으로써 생성과 소멸시 작동될 행동을 적어줄수있다.
2. 빈객체를 만들때 해당 클래스에 메서드를 만들어주면 생성과 소멸시 할 행동을 적어줄수있다.

Bea이란 무엇인가요?

상속을 받지않고 가장 기본적인 field와 getter, setter만 가지고 있는 객체가 bea이다.
Spring에서 POJO(plain old java object)는 beas라고 불린다.
bea를 생성하기 위해서는 xml config file의 설정을 통해 Spring container에 의해 생성된다.
xml config file에서는 bea의 lifecycle과 dependency를 설정할 수 있다.
Spring container에서 객체를 생성해주기 때문에 개발자가 new 생성자로 객체를 생성할 필요가 없다.
container에 의해 생성된 bea에 접근하기 위해서는 getBean("[bea_id]")로 통해 bea에 대한 reference 값을 가져올 수 있다.

Interceptor와 Filter의 차이점을 말해주세요.

[필터(Filter)란?]

필터(Filter)는 J2EE표준 스펙 기능으로 디스패처 서블릿(Dispatcher Servlet)에 요청이 전달되기 전/후에 url 패턴에 맞는 모든 요청에 대해 부가작업을 처리할 수 있는 기능을 제공한다. 즉, 스프링 컨테이너가 아닌 톰캣과 같은 웹컨테이너에 의해 관리가 되므로 디스패처 서블릿으로 가기 전에 요청을 처리하는 것이다.

[인터셉터(Interceptor)란?]

인터셉터(Interceptor)는 J2EE 표준 스펙인 필터(Filter)와 달리 Spring이 제공하는 기술로써, 디스패처 서블릿(Dispatcher Servlet)이 컨트롤러를 호출하기 전과 후에 요청과 응답을 참조하거나 가공할 수 있는 기능을 제공한다. 즉, 웹 컨테이너에서 동작하는 필터와 달리 인터셉터는 스프링 컨텍스트에서 동작을 하는 것이다.

IOC와 DI에 대해서 설명해주세요.

DI(Dependency Injection)

DI(Dependency Injection)란 스프링이 다른 프레임워크와 차별화되어 제공하는 의존 관계 주입 기능으로, 객체를 직접 생성하는 게 아니라 외부에서 생성한 후 주입 시켜주는 방식이다.

DI(의존성 주입)를 통해서 모듈 간의 결합도가 낮아지고 유연성이 높아진다.

loc(Inversion of Control)

IoC(Inversion of Control)란 "제어의 역전" 이라는 의미로, 말 그대로 메소드나 객체의 호출작업을 개발자가 결정하는 것이 아니라, 외부에서 결정되는 것을 의미한다.

IoC는 제어의 역전이라고 말하며, 간단히 말해 "제어의 흐름을 바꾼다"라고 한다.

객체의 의존성을 역전시켜 객체 간의 결합도를 줄이고 유연한 코드를 작성할 수 있게 하여 가독성 및 코드 중복, 유지 보수를 편하게 할 수 있게 한다.

Container란 무엇인가요?

스프링 컨테이너는 자바 객체의 생명 주기를 관리하며, 생성된 자바 객체들에게 추가적인 기능을 제공하는 역할을 합니다. 여기서 말하는 자바 객체를 스프링에서는 빈(Bean)이라고 부릅니다. 그리고 저번 시간에 배웠던 IoC와 DI의 원리가 이 스프링 컨테이너에 적용됩니다.

VO, DTO, DAO에 대해서 각각 설명해 주세요.

링크텍스트

1. DAO(Data Access Object)

DAO는 DB의 data에 접근하기 위한 객체로 직접 DB에 접근하여 데이터를 삽입, 삭제, 조회 등 조작할 수 있는 기능을 수행한다.

DataBase 접근을 하기 위한 로직과 비즈니스 로직을 분리하기 위해 사용한다.

DAO의 경우는 DB와 연결할 Connection 까지 설정되어 있는 경우가 많다.

현재 많이 쓰이는 Mybatis 등을 사용할 경우 커넥션풀까지 제공되고 있기 때문에 DAO를 별도로 만드는 경우는 드물다.

2. DTO(Data Transfer Object)

DTO는 계층간(Controller, View, Business Layer) 데이터 교환을 위한 자바 빈즈(Java Beans)를 의미한다.

DTO는 로직을 가지지 않는 데이터 객체이고 getter/setter메소드만 가진 클래스를 의미한다.

DTO(Data Transfer Object)는 데이터 전송(이동) 객체라는 의미를 가지고 있다.

DTO는 주로 비동기 처리를 할 때 사용한다.

계층간 데이터 교환을 위한 객체(Java Beans)이다.

DB의 데이터를 Service나 Controller 등으로 보낼 때 사용하는 객체를 말한다.

즉, DB의 데이터가 Presentation Logic Tier로 넘어올때는 DTO로 변환되어 오고가는 것이다.

로직을 갖고 있지 않는 순수한 데이터 객체이며, getter/setter 메서드만을 갖는다.

또한 Controller Layer에서 Response DTO 형태로 Client에 전달한다.

3. VO(Value Object)

DTO와 달리 VO는 Read-Only속성을 값 오브젝트이다.

자바에서 단순히 값 타입을 표현하기 위해 불변 클래스(Read-Only)를 만들어 사용한다.

예를 들면 빨강은 Color.RED, 초록은 Color.GREEN 이렇게 단순히 값만 표현하기 위해 getter기능만 존재한다.

VO의 핵심 역할은 equals()와 hashCode() 를 오버라이딩 하는 것이다.

VO 내부에 선언된 속성(필드)의 모든 값들이 VO 객체마다 값이 같아야, 똑같은 객체라고 판별한다.

VO는 Getter와 Setter를 가질 수 있으며, VO는 테이블 내에 있는 속성 외에 추가적인 속성을 가질 수 있으며, 여러 테이블(A, B, C)에 대한 공통 속성을 모아서 만든 BaseVO 클래스를 상속받아서 사용할 수 도있습니다.

MVC

MVC에 대해서 설명해 주세요.

모델: 데이터와 비즈니스 로직을 관리합니다.

뷰: 레이아웃과 화면을 처리합니다.

컨트롤러: 명령을 모델과 뷰 부분으로 라우팅합니다.

Servlet이 무엇인가요? (사실 이건 Java 섹션에 있는게 맞음)

서블릿이란 자바를 사용하여 웹을 만들기 위해 필요한 기술입니다. 그런데 좀더 들어가서 설명하면 클라이언트가 어떠한 요청을 하면 그에 대한 결과를 다시 전송해주어야 하는데, 이러한 역할을 하는 자바 프로그램입니다.

예를 들어, 어떠한 사용자가 로그인을 하려고 할 때, 사용자는 아이디와 비밀번호를 입력하고, 로그인 버튼을 누릅니다.

그때 서버는 클라이언트의 아이디와 비밀번호를 확인하고, 다음 페이지를 띄워주어야 하는데, 이러한 역할을 수행하는

것이 바로 서블릿(Servlet)입니다. 그래서 서블릿은 자바로 구현 된 *CGI라고 흔히 말합니다.

Dispatcher-Servlet이란 무엇인가요?

가장 앞단에서 HTTP 프로토콜로 들어오는 모든 요청을 가장 먼저 받아 적합한 컨트롤러에 위임해주는 프론트 컨트롤러(Front Controller)라

Spring MVC는 DispatcherServlet이 등장함에 따라 web.xml의 역할을 상당히 축소시켜주었습니다. 기존에는 모든 서블릿에 대해 URL 매핑을 활용하기 위해서 web.xml에 모두 등록해주어야 했지만, dispatcher-servlet이 해당 어플리케이션으로 들어오는 모든 요청을 핸들링해주고 공통 작업을 처리면서 상당히 편리하게 이용할 수 있게 되었습니다.

Spring MVC에서 HTTP 요청이 들어왔을 때의 흐름을 설명해 주세요.

Step1. DispatcherServlet은 web.xml에 정의된 URL 패턴에 맞는 요청을 받고 URL 컨트롤러의 맵핑 작업을 HandlerMapping에 요청

Step2. HandlerMapping은 URL을 기준으로 어떤 컨트롤러를 사용할지 결정, 결과는 HandlerExecution Chain객체에 담아서 리턴하는데, 요청에 해당하는 Interceptor가 있을 경우 함께 줌.

Step3. HandlerAdapter는 컨트롤러의 메소드를 호출하는 역할을 하는데 실행될 Interceptor가 있을 때는 Interceptor의 preHandle() 메소드를 실행한 다음 컨트롤러의 메소드를 호출하여 요청 처리

Step4. 컨트롤러는 요청을 처리 한 뒤 처리한 결과 및 ModelAndView를 DispatcherServlet에 전달.

Step5. DispatcherServlet은 컨트롤러에서 전달받은 View 이름과 매칭되는 실제 View 파일을 찾기 위해 ViewResolver에게 요청.

Step6. ViewResolver는 컨트롤러가 처리한 결과를 보여줄 뷰를 결정, 컨트롤러에서 전달받은 View 이름의 앞뒤로 prefix, suffix 프로퍼티를 추가한 값이 실제 사용할 뷰 파일 경로가 됨. ViewResolver는 맵핑되는 View 객체를 DispatcherServlet에 전달.

Step7. DispatcherServlet은 ViewResolver에 전달받은 View Model을 넘겨서 클라이언트에게 보여줄 화면을 생성.

++ preHandle() : 컨트롤러가 호출되기 전에 실행,

++ prefix : 접두어

++ suffix : 접미사

ETC

전산 기본

TDD란 무엇인가요?

테스트 주도 개발입니다 .
반복 테스트를 이용한 소프트웨어 방법론으로 작은 단위의 테스트 케이스를 작성하고 이를 통과하는 코드를 추가하
는 단계를 반복하여 구현합니다.

프레임워크와 라이브러리 차이는 무엇인가요?

프레임워크는 소프트웨어의 특정문제를 해결하기 위해서 상호 협력하는 클래스와 인터페이스의 집합이고 라이브러
리는 단순 활용이 가능한 도구들의 집합입니다.

디자인 패턴이란 무엇인가요?

과거의 소프트웨어 개발 과정에서 발견된 설계의 노하우를 축적하여 그 방법에 이름을 붙여서 이후에 재사용하기 좋
은 형태로 특정 규약을 만들어서 정리한 것입니다.

• GoF 디자인 패턴의 분류

생성 (Creational) 패턴	구조 (Structural) 패턴	행위 (Behavioral) 패턴
<ul style="list-style-type: none">추상 팩토리 (Abstract Factory)빌더 (Builder)팩토리 메서드 (Factory Methos)프로토타입 (Prototype)싱글턴 (Singleton)	<ul style="list-style-type: none">어댑터 (Adapter)브리지 (Bridge)컴퍼지트 (Composite)데커레이터 (Decorator)퍼사드 (Facade)플라이웨이트 (Flyweight)프록시 (Proxy)	<ul style="list-style-type: none">책임 연쇄 (Chain of Responsibility)커맨드 (Command)인터프리터 (Interpreter)이터레이터 (Iterator)미디에이터 (Mediator)메멘토 (Memento)옵서버 (Observer)테이트 (State)스트래티지 (Strategy)템플릿 메서드 (Template Method)비지터 (Visitor)

생성 (Creational) 패턴

객체 생성에 관련된 패턴

객체의 생성과 조합을 캡슐화해 특정 객체가 생성되거나 변경되어도 프로그램 구조에 영향을 크게 받지 않도록 유연
성을 제공한다.

구조 (Structural) 패턴

클래스나 객체를 조합해 더 큰 구조를 만드는 패턴

예를 들어 서로 다른 인터페이스를 지닌 2개의 객체를 묶어 단일 인터페이스를 제공하거나 객체들을 서로 묶어 새로
운 기능을 제공하는 패턴이다.

행위 (Behavioral) 패턴

객체나 클래스 사이의 알고리즘이나 책임 분배에 관련된 패턴

한 객체가 혼자 수행할 수 없는 작업을 여러 개의 객체로 어떻게 분배하는지, 또 그렇게 하면서도 객체 사이의 결합
도를 최소화하는 것에 중점을 둔다.

GoF 디자인 패턴의 종류

생성(Creational) 패턴**추상 팩토리(Abstract Factory)**

구제적인 클래스에 의존하지 않고 서로 연관되거나 의존적인 객체들의 조합을 만드는 인터페이스를 제공하는 패턴
팩토리 메서드(Factory Method)

객체 생성 처리를 서브 클래스로 분리해 처리하도록 캡슐화하는 패턴

싱글톤(Singleton)

전역 변수를 사용하지 않고 객체를 하나만 생성하도록 하며, 생성된 객체를 어디에서든지 참조할 수 있도록 하는 패턴

구조(Structural) 패턴**컴퍼지트(Composite)**

여러 개의 객체들로 구성된 복합 객체와 단일 객체를 클라이언트에서 구별 없이 다루게 해주는 패턴

데코레이터(Decorator)

객체의 결합을 통해 기능을 동적으로 유연하게 확장할 수 있게 해주는 패턴

행위(Behavioral) 패턴**옵서버(Observer)**

한 객체의 상태 변화에 따라 다른 객체의 상태도 연동되도록 일대다 객체 의존 관계를 구성하는 패턴

스테이트(State)

객체의 상태에 따라 객체의 행위 내용을 변경해주는 패턴

스트래티지(Strategy)

행위를 클래스로 캡슐화해 동적으로 행위를 자유롭게 바꿀 수 있게 해주는 패턴

템플릿 메서드(Template Method)

어떤 작업을 처리하는 일부분을 서브 클래스로 캡슐화해 전체 일을 수행하는 구조는 바꾸지 않으면서 특정 단계에서 수행하는 내역을 바꾸는 패턴

커맨드(Command)

실행될 기능을 캡슐화함으로써 주어진 여러 기능을 실행할 수 있는 재사용성이 높은 클래스를 설계하는 패턴

애자일 방법론이란?

애자일 방법론은 계획 → 설계(디자인) → 개발(발전) → 테스트 → 검토(피드백) 순으로 반복적으로 진행된다. 계획을 세운 후 다음 단계까지 기다려서 절차대로 진행하는 폭포수 모델과 달리 먼저 진행 후 분석, 시험, 피드백을 통하여 개선하여 나가는 진행 모델이다.

앞을 예측하며 개발하지 않고, 일정한 주기를 가지고 계속 검토해 나가며 필요할 때마다 요구사항을 더하고 수정하여 커다랗게 살을 붙이면서 개발해 프로세스 모델 방식이다

도커란 무엇인가요?

컨테이너 기반의 오픈소스 가상화 플랫폼

서버에서도 다양한 OS 환경, 여러 프로그램들을 화물에 비유하여 컨테이너에 실어 여러 곳으로 운반하여 배포할 수 있다는 개념

**Assignees**

No one—assign yourself

**Labels**

None yet



Projects



None yet

Milestone



No milestone

Linked pull requests




Successfully merging a pull request may close this issue.

None yet

1 participant



 Pin issue 