스프링 부트

1.정의

-스프링 프레임 워크를 사용하는 프로젝트를 간편하게 설정할 수 있는 스프링 프레임워크의 서브 프로젝트

-단독 실행 가능한 스프링 앱을 생성하고 가능한 빨리 실행되게 설계

-최소한의 설정으로 쉽게 웹 어플을 실행

2.장점

-매우 빠르게 모든 스프링 개발에 관한 경험을 광범위하게 제공한다.

-프로젝트 환경 구축에서 큰 영역을 차지하는 비기능적인 기능들을 제공한다.(내장형 서버, 시큐리티, 측정, 상태점검, 외부 설정)

**Configuration(내부에 컴포넌트 있음)=xml설정 파일 대용**

빈을 수동으로 등록하기 위해 @Configuration을 추가하고 @Bean을 사용해 수동으로 빈을 등록할 수 있다.

-의존성 주입을 위해서 @Configuration을 사용하게 된다.

-빈 등록 시 싱글톤(singleton)이 되도독 보장해준다.

(특정 클래스가 단 하나만의 인스턴스를 생성하여 사용하기 위한 패턴)

-그래서 객체가 여러 번 생성되더라도 한번만 생성된다.

Component-scan

빈으로 등록 될 준비를 마친 클래스들을 스캔하여 빈으로 등록해주는 것이다.

ConfigurationClassParser가 Configuration클래스를 파싱한다.

ComponentScan 설정을 파싱

Base-package에 설정한 패키지를 기준으로

ComponentScanAnnotationParser가 스캔하기 위한 설정을 파싱

Base-package 설정을 바탕으로 모든 클래스를 로딩

클래스로더가 로딩한 클래스들을 BeanDefinition으로 정의

정의를 토대로 Bean을 만든다.

@MapperScan

매퍼를 하나씩 등록하는게 아닌 패키지 경로를 지정하여 이하 위치에 있는 인터페이스들은 전부 Mapper로 사용가능

**Controller**

-사용자의 요청이 진입하는 지점

-요청에 따라 어떤 처리를 할지 결정만을 한다.

-사용자에게 View를 응답으로 보내준다.

-컨트롤러가 붙은 클래스의 내부 메소드에 추가 설정이 없으면

View Resolver를 통해 view파일의 완전 경로로 만들어 진다.

Prefix와 suffix를 통해 완전 경로를 만드는데

앞쪽 파일 경로가 prefix /resources/static/등등

뒤쪽 파일종류가 suffix가 된다.(.jsp .html 등등)

컨트롤러 메소드는 Model이라는 타입의 객체를 파라미터로 받을 수 있다.

JSP는 request나 session을 사용해 받았는데

스프링은 model을 사용해 받는다.

기본적으로 Java Beans 규칙에 맞는 객체는 자동으로 화면에 전달한다.

생성자가 없거나 빈 생성자이며 getter/setter를 가진 클래스의 객체이다.

이때 기본 자료형은

1.Model타입 선언 후 addAttribute()를 통해 전달

2.@modelattribute(“파라미터명”)int page 어노테이션 사용

RedirectAttributes

일회성으로 데이터를 전달한다.

addFlashAttribute() 메소드는 (이름,값)을 파라미터로 이용하여 화면에 딱한번만 사용하고 증발해버리는 데이터를 전달한다.

(session으로 값을 보내기 때문에 값이 보이지 않는다.)

addAttribute() 리다이렉트할 주소 뒤에 쿼리스트링으로 데이터를 전달한다.

컨트롤러 리턴타입

**String** : jsp를 이용하는 경우 jsp파일의 이름을 나타냄

+상황에 따라 다른 화면을 보여줄 필요가 있을 때 유용하다.

(if~else 처리)

Return문에 특별한 키워드를 붙일 수 있다.

Redirect:리다이렉트

Forward:포워딩(생략시 얘가 기본)

**Void** : 호출하는 URL과 동일한 이름의 jsp를 나타냄

-ViewResolver설정과 맞물려 작동한다.

Vo,DTO : 주로 json타입의 데이터를 만들어서 반환하는 용도

스프링 프레임워크에서 제공하는 클래스 중 HttpEntity라는 클래스가 존재한다.

HTTP요청(Request) 또는 응답(Response)에 해당하는

HttpHeader와 HttpBody를 포함하는 클래스이다.

ResponseEntity : response할 때 HTTP헤더 정보와 내용을 가공

사용자의 HttpRequest에 대한 응답 데이터를 포함하는 클래스

HttpStatus, HttpHeaders, HttpBody를 포함한다.

상태코드 Status

헤더 Headers

응답데이터 ResponseData

를 담는 생성자도 존재한다.

Model, ModelAndView : Model로 데이터를 반환하거나 화면까지 지정

HttpHeaders : 응답에 내용없이 HTTP헤더 메시지만 전달하는 용도

**@Controller(내부에 @컴포넌트가 있다)**

-MVC패턴에 포함된다.(C의 역할을 맡는다-IoC컨테이너에 등록한다.)

**@RestController(내부에 @컴포넌트가 있다)**

-

-@Controller는 반환 값이 String이면 뷰이름으로 인식된다.

-@RestController는 HTTP 메세지 바디에 바로 입력한다.

따라서 실행 결과로 ok 메시지를 받을 수 있다.

@ResponseBody와 관련이 있다.

**HTTP 요청 매핑(RequestMapping)(메소드)**

|  |  |
| --- | --- |
| @RequestMapping | url자체를 메소드와매핑시키는 것 |
| @GetMapping | GET 요청 |
| @PostMapping | POST 요청 |
| @PutMapping | PUT 요청 |
| @DeleteMapping | DELETE 요청 |
| @PatchMapping | PATCH 요청 |

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Action | 역할 | 페이로드 |
| GET | Index/retrieve | 모든/특정 리소스 조회 | X |
| POST | Create | 리소스를 생성 | O |
| PUT | Replace | 리소스의 전체 교체 | O |
| PATCH | Modify | 리소스 일부 수정 | O |
| DELETE | delete | 모든/특정 리소스 삭제 | X |

1. URI는 정보의 자원을 표현해야 한다.
2. 자원에 대한 행위는 Method로 표현한다.

Rest API의 구성

|  |  |  |
| --- | --- | --- |
| 구성 요소 | 내용 | 표현방법 |
| Resource | 자원 | HTTP URI |
| Verb | 자원에 대한 행위 | HTTP Method |
| Representations | 자원에 대한 행위의 내용 | HTTP Message Pay Load |

-고객의 요청에 정보를 어떤 컨트롤러가 처리할지를 매핑하기 위한 어노테이션이다.

-추가로 경로를 여러 개 설정할 수 있다.

**@RequestParam(“키네임”)**

-서블렛에 GET요청때 파라미터로 데이터를 넘기는데 똑같다.

**@PathVariable(경로 변수)사용**

**@RequestMapping등등은 method를 변수로 지정하여 사용가능하다.**

@GetMapping(“/mapping/{userid}/name/{name}”)

@PathVariable({“userid”}) String id, @PathVariable String name

다중으로 사용 가능

또는

@PathVariable String userid

특정 파라미터 조건 매핑

@GetMapping(value=”/mapping-param”, params=”mode=debug”)

파라미터에 mode=debug가 존재해야 호출한다.

!mode나 mode!=debug등으로 사용가능하며 사용하는 일은 거의 없다.

특정 헤더 조건 매핑

Headers=”mode=debug”

미디어 타입 조건 매핑

Consume

**Domain**

-Setter/Getter등 VO가 들어있는 폴더

**Mapper**

Mapping 파일에 기재된 SQL을 호출하기 위한 인터페이스이다.

**Service(MVC에서 Model)**

**-클라이언트의 요청에 대한 비즈니스 로직을 수행하는 Component이다.**

**-사용자가 url에 따른 Controller가 호출되면,**

**컨트롤러는 해당 비즈니스 로직을 수행하기 위해 Service를 호출한다.**

**Service는 로직 수행 결과를 Controller에 반환한다.**

**넘겨야하는 파라미터 값이 2개 이상이면**

**@Param(“넘길 이름”) Long bno로 넘겨야한다.**

**@Service**

Service 인터페이스와 Service클래스로 나뉘는데

인터페이스를 사용할 경우 클래스간의 결합이 느슨해져서

로직이 바뀔경우 Implement만 갈아끼우면 된다.

캐시를 사용할 수 있게 된다.

Webapp

Ajax

JSTL

편의성 어노테이션

**@Log4j**

-JAVA를 사용하여 Log를 남기는 도구

-아파치에서 만든 오픈소스 라이브러리

FATAL->ERROR->WARN->INFO->DEBUG->TRACE

순으로 심한 버그에서 광범위한 문제를 볼 수 있음

롬북

참고

HTTP 요청 매핑

<https://ittrue.tistory.com/235>

맵퍼 인터페이스

<https://hyejin.tistory.com/261>

@RestController

https://backendcode.tistory.com/213

RestController

@Controller+@ResponseBody+@RequestBody가 추가 된 것이다.

-주된 용도는 JSON형태로 객체 데이터를 반환하는 것이다.

-객체를 ResponseEntity로 감싸서 반환한다.

@ResponseBody : 자바 객체(VO)를 데이터(JSON)로 바꿔 Body안에 넣어주는 어노테이션

@RequestBody : Body안에 데이터를 자바 객체(VO)형태로 변환해주는 어노테이션

HikariCP

빠르고 간단하게 믿을 수 있는 JDBC 연결 풀

데이터베이스 연결을 관리해 주는 도구

설정된 커넥션의 사이즈만큼의 연결을 허용하며 HTTP 요청에 대해 순차적으로 DB 커넥션을 처리해 주는 기능을 수행

Mybatis

쿼리문을 복잡하게 입력하지 않고 실제 쿼리문과 유사하게 작성할 수 있으며 데이터베이스 쿼리와 프로그래밍 코드를 분리하여

유지보수성과 생산성을 높일 수 있다.

Ajax

동기식 처리모델

순차적으로 실행되며 어떤 작업이 수행 중이면 다음 작업은 대기하게 된다.

비동기식 처리모델

자바

List

객체를 일렬로 늘어놓은 구조이다

객체를 인덱스로 관리하기 때문에 객체를 저장하면 자동으로 인덱스를 부여하고 검색, 삭제할 수 있는 기능을 제공한다.

List에는 ArrayList, Vector, LinkedList등이 있다.

add() : 객체 추가 메소드

get() : 객체를 찾아오는 메소드

remove() : 객체를 삭제하는 메소드

ArrayList란

List 인터페이스를 구현한 클래스로 컬렉션 프레임워크에서 가장 많이 사용된다.

객체가 인덱스로 관리되기에 배열과 유사하다

자동으로 저장용량이 늘어나게 된다.

데이터가 연속적으로 있어 순서가 유지된다.

인터페이스

상속

클래스 타입

Spring Security

인증, 권한 부여 및 보호 기능을 제공하는 프레임워크이다.

인증 : 해당 사용자가 본인이 맞는지를 확인하는 절차

인가 : 인증된 사용자가 요청된 자원에 접근가능한가를 결정하는 절차

인증 방식

1. Credential 방식 : username, password를 이용하는 방식
2. 이중 인증(twofactor 인증) : 사용자가 입력한 개인 정보를 인증 후, 다른 인증 체계를 이용하여 두가지의 조합으로 인증하는 방식이다.
3. 하드웨어 인증 : 자동차 키와 같은 방식

이중 Spring Security는 credential 기반의 인증을 취합니다.

Principal : 아이디

Credential : 비밀번호

특정 자원에 대한 접근을 제어하기 위해서는 권한을 가지게 된다.

특정 권한을 얻기 위해서는 유저는 인증정보가 필요하고 관리자는 해당 정보를 참고해 권한을 인가한다.

보편적으로 username-password 패턴의 인증방식을 거치기 때문에 스프링 시큐리티는 principal-credential 패턴을 가지게 된다.

Spring Security의 특징

Filter를 기반으로 동작한다.

Spring MVC와 분리되어 관리하고 동작할 수 있다.

Bean으로 설정할 수 있다.

Spring Security 3.2부터는 XML설정을 하지 않아도 된다.

Spring Security Architectere