권영가기 코딩단.

606. 스프링 시큐리티. → 매핑해주는것.

서블릿 = extends HttpServlet → JSP → @Controller.

스크립트릿 = JSP

초콜릿 = 초코의 small 단위

(인덱스 ⇒ 인터넷이다.) → Collection

'k', 'v' 이다. → Map.

URL = 'key' 이다 그리고 매핑 ⇒ 컨트롤러 와 한다.

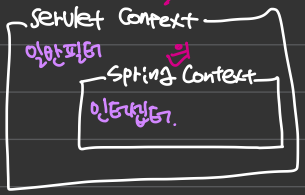
필터 와 인터셉터

↓ ↓

서블릿자원 @Bean

Context ?

이름: ServletContext.xml



스프링 시큐리티는 서블릿의 필터와 인터셉터를 이용한다.

① 인터셉터는 컨트롤러를 호출할때 관여.

② 필터와 인터셉터를 이용하면 컨텍스트를 생성 하여 처리한다. (Context는 객체이다.)

↳ Context 객체를 만들어줄게

③ 시큐리티는 컨텍스트 내에서 동작한다.

↳객체: 메모리에 올라감 ⇒ 메모리 내에서 작동.

④ 컨텍스트가 있는 Beans ⇒ 여러빈들

유니크 객체

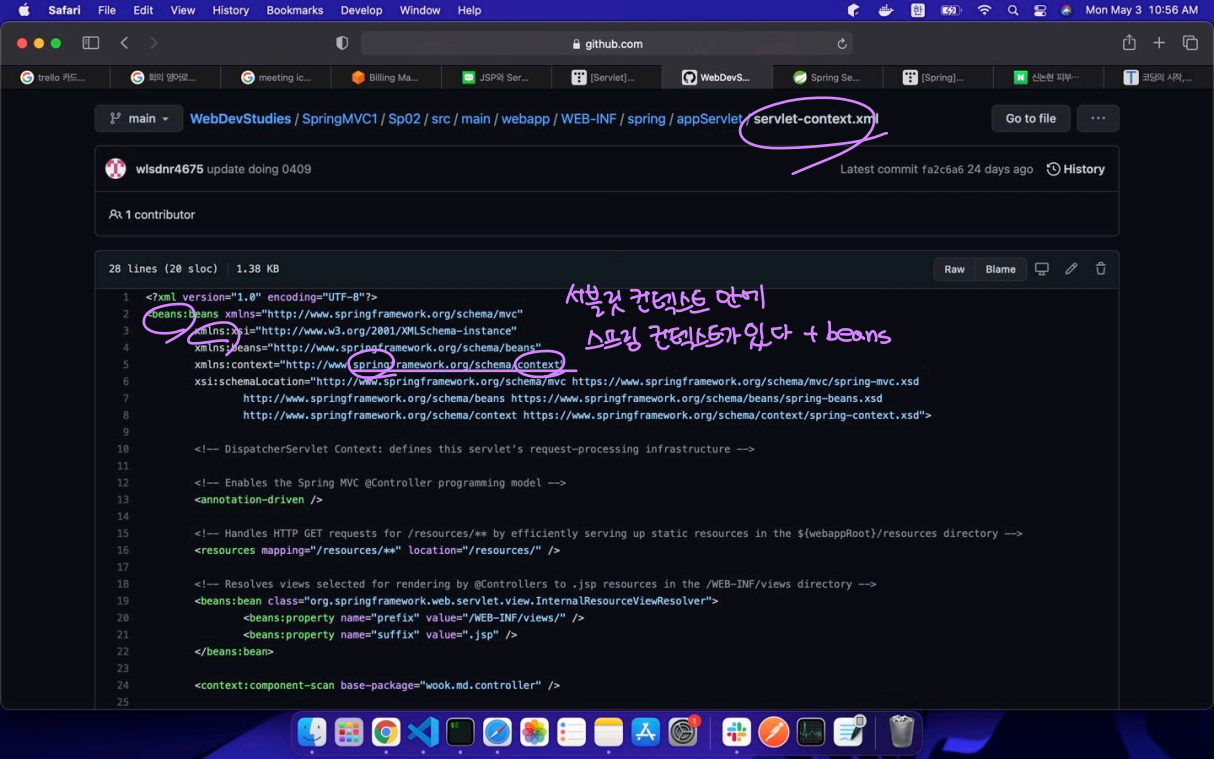
xmlns : XML NameSpace == HTML = DOM (객체화 하는 방법론)

XML 네임스페이스(namespace)

XML 네임스페이스는 XML 요소 간의 이름에 대한 충돌을 방지해 주는 방법을 제공합니다.
XML 네임스페이스는 요소의 이름과 속성의 이름을 하나의 그룹으로 묶어주어 이름에 대한 충돌을 해결합니다.

이러한 XML 네임스페이스는 URI(Uniform Resource Identifiers)로 식별됩니다.

(RequestMapping 처럼 URL)


Safari File Edit View History Bookmarks Develop Window Help
Mon May 3 10:56 AM
github.com
main
WebDevStudies / SpringMVC1 / Sp02 / src / main / webapp / WEB-INF / spring / appServlet / servlet-context.xml
Go to file
wlsdnr4675 update doing 0409
Latest commit fa2c6a6 24 days ago
History
1 contributor
28 lines (20 sloc) 1.38 KB
Raw
Blame
서블릿 컨텍스트 안에
스프링 컨텍스트가 있다 + beans
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/mvc https://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">
<!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />
<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the ${webappRoot}/resources directory -->
<resources mapping="/resources/**" location="/resources/" />
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<beans:property name="prefix" value="/WEB-INF/views/" />
<beans:property name="suffix" value=".jsp" />
</beans:bean>
<context:component-scan base-package="wook.md.controller" />

① 필터

필터는 servlet 컨텍스트 안에 있다.

type 객체 Identify

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ⇒ Mapping.

String : URI

web.xml

```
<servlet>
        <servlet-name>appServlet</servlet-
    name>
        <servlet-
    class>org.springframework.web.servlet.Dispatch
    erServlet</servlet-class>
        <init-param>
            <param-
    name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/
    appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

1번을 실행 하고

↓

그안에 스프링 컨텍스트 + Beans 가있다.

⇒ 필터 + 인터셉터 = 시큐리티 생성

→ 서블릿 컨텍스트 안에서 시큐리티를 생성해 야한다.

①

Web.xml 실행 → servlet-context.xml 실행 → beans 와 Spring context 생성

② web.xml 이동 filter mapping → 서블릿 컨텍스트에 넘어감.

Auth vs Author

Auth : 자신이 증명할 자료 제시 → Consumer

author : 남이 부여 → Supplier

(*Filter / Interceptor : 속성값)

Authentication / Authorization

1) 1차 경비아저씨

Auth Manager 에게 토큰을 파라미터로 제공.

Provider Manager 인증처리위임

↓

Auth Provider

UserDetailsService 권한 정보처리까지 한참

로그인과 로그아웃 처리

접근제한 설정

① 컨트롤러에 가서 3가지 URL 작성.

1) 로그인 X 접속가능 URL : "/all"

2) 로그인 O " URL "/{id}"

3) 관리자만 접속 가능 URL "/admin"

4) build.gradle 에가서 시큐리티 DI 한다.

implementation 'org.springframework.boot:spring-boot-starter-security'

5) 부트가 시큐리티에서 제공하는 기본로그인 화면으로 이동.

콘솔에 있는 Password를 가지고, User 라는 id로 접속해서 로그인 시도.

6) 권한 설정은 ROLE.* 으로 한다.

순서 ↓

<Manager>

<Provider>

<Service> → ROLE.* 하는 것은 권한 부여하는 코드이다.

*인증이나 권한이 안되면 오류메세지는 403 Forbidden 에러메시지를 보게된다.

* Security Exeption.

-enum : 여러개의 상수 값들의 집합.

enum 상수값이 불규칙한 경우에는, 상수 이름( ) 에 값을 입력해서 사용한다.

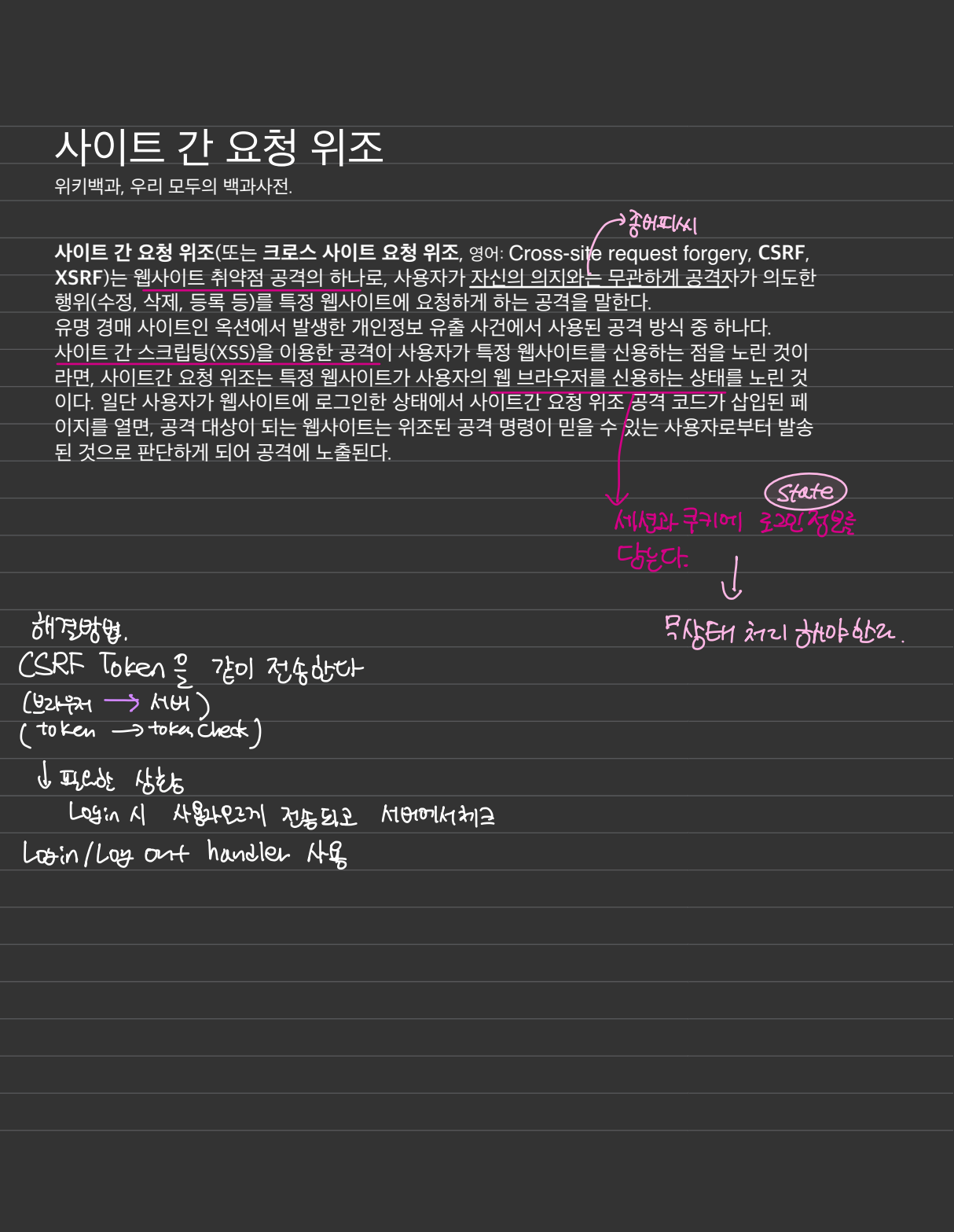
```
@RequiredArgsConstructor
@Getter
public enum ErrorCode {
    // public static final 401_ERROR = "blah blah";
    AUTHENTICATION_FAILED(401, "AUTH_001",
"AUTHENTICATION_FAILED"), LOGIN_FAILED(401,
"AUTH_002", "LOGIN_FAILED"),
    ACCESS_FAILED(401, "AUTH_003", "ACCESS_FAILED"),
TOKE_GENERATION_FAILED(500, "AUTH_004",
"TOKE_GENERATION_FAILED");

    private final int status;
    private final String code;
    private final String message;
```

단, 반드시 생성자에 필드값이 할당 되어야 함으로 @RequiredArgsConstructor 가 필요하다. 상수는 Setter가 필요 x, Getter만 추가해야 한다.

사이트 간 요청 위조

위키백과, 우리 모두의 백과사전.

사이트 간 요청 위조(또는 **크로스 사이트 요청 위조**, 영어: Cross-site request forgery, **CSRF**, **XSRF**)는 웹사이트 취약점 공격의 하나로, 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 하는 공격을 말한다.
유명 경매 사이트인 옥션에서 발생한 개인정보 유출 사건에서 사용된 공격 방식 중 하나다.
사이트 간 스크립팅(XSS)을 이용한 공격이 사용자가 특정 웹사이트를 신용하는 점을 노린 것이라면, 사이트간 요청 위조는 특정 웹사이트가 사용자의 웹 브라우저를 신용하는 상태를 노린 것이다. 일단 사용자가 웹사이트에 로그인한 상태에서 사이트간 요청 위조 공격 코드가 삽입된 페이지를 열면, 공격 대상이 되는 웹사이트는 위조된 공격 명령이 믿을 수 있는 사용자로부터 발송된 것으로 판단하게 되어 공격에 노출된다.



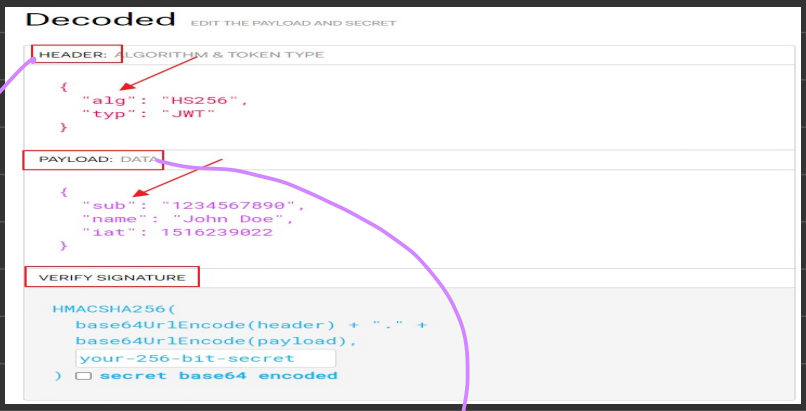
J SON 웹 토큰(JSON Web Token, JWT, "jot"[1])은 선택적 서명 및 선택적 암호화를 사용하여 데이터는 만들기 위한 인터넷 표준으로 페이로드는 몇몇 클레임(claim) 표명(assert)을 처리하는 JSON을 보관하고 있다. 토큰은 비공개 시크릿 키 또는 공개/비공개 키를 사용하여 서명된다. 이를테면 서버는 "관리자로 로그인됨"이라는 클레임이 있는 토큰을 생성하여 이를 클라이언트에 제공할 수 있다. 그러면 클라이언트는 해당 토큰을 사용하여 관리자로 로그인됨을 증명한다. 이 토큰들은 한쪽 당사자의 비공개 키(일반적으로 서버의 비공개 키)에 의해 서명이 가능하며 이로써 해당 당사자는 최종적으로 토큰이 적법한지를 확인할 수 있다. 일부 적절하고 신뢰할만한 수단을 통해 다른 당사자가 상응하는 공개키를 소유하는 경우 이 경우 또한 토큰의 적법성 확인이 가능하다. 토큰은 크기가 작고[2] URL 안전으로 설계되어 있으며[3] 특히 웹 브라우저 통합 인증(SSO) 컨텍스트에 유용하다. JWT 클레임은 아이덴티티 제공자와 서비스 제공자 간(또는 비즈니스 프로세스에 필요한 클레임)의 인가된 사용자의 아이덴티티를 전달하기 위해 보통 사용할 수 있다.[4][5]
JWT는 따른 JSON 기반 표준에 의존한다: JSON 웹 시그너처 및 JSON 웹 암호화.[1][6][7]

Client / Server

https://jwt.io

JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.
JWT.IO allows you to decode, verify and generate JWT.

CSRF 토큰은 더 이상 표준이 아니다. JWT 가 표준이다. 그래서 리덕스에서 기존 액션이 표준이 아니고, Flex Standard Action 이라는 새로운 액션을 만들었다. 이 액션이 기존 액션과 다른 점은 페이로드를 갖고 있는 것이다.




```
public String createToken() {
        try {
            Map<String, Object> headers = new HashMap<>();
            headers.put("alg", "HS256");
            headers.put("typ", "JWT");
            Map<String, Object> payloads = new HashMap<>();
            payloads.put("data", "My First JWT");
            long expirationTime = 1000 * 60L * 60L * 2L; // 토큰 유효 시간 2시간
            Date ext = new Date();
            ext.setTime(ext.getTime() + expirationTime);
            // Jwts j = new JWts(getName() + .....) 와 같다
            return
Jwts.builder().setHeader(headers).setClaims(payloads).setSubject("user").setExpiration(ext)
                    .signWith(SignatureAlgorithm.HS256, key.getBytes()).compact();
        } catch (SecurityException e) {
            log.info("Invalid JWT Signature");

        } catch (MalformedJwtException e) {
            log.info("Invalid JWT Token");

        } catch (ExpiredJwtException e) {
            log.info("Expired JWT Token");

        } catch (UnsupportedJwtException e) {
            log.info("Unsupported JWT Signature");

        } catch (IllegalArgumentException e) {
            log.info("JWT tokens compact of handler are invalid");

        }
        // 일부러 널을 준다 캐치 5개 에러 말고 다른것이 나올때 널 포인트 익셉션 주는게 더 낫다. (보안이 뚫리
는 것보단)
        return null;

    }
```

Key, Value의 조합 Map사용하기.

스웨거

문서 생성을 위한 플러그인, 스웨거 기반 어노테이션으로 애플리케이션을 검사한다.

```
implementation group: 'javax.validation', name: 'validation-api', version:
'2.0.1.Final'
    compile group: 'io.springfox', name: 'springfox-swagger2', version: '2.9.2'
    compile group: 'io.springfox', name: 'springfox-swagger-ui', version: '2.9.2'
```

@ApiOperation : mappring 에 대한 전반적인 설명과 필요 조건에 대한 설정
@ApiResponses : 반환 코드에 대해서 message 를 커스텀하게 해줍니다.

출처: https://heowc.tistory.com/23 [허원철의 개발 블로그]

Member VO 에는 username 과 password가 있어야한다

enum 타입의 Role을 생성한다.
→Role을 스트링화 한다

```
public static Role of(String code) {
        return Arrays.stream(Role.values()).filter(i ->
i.getCode().equals(code)).findAny().orElse(UNKNOWN
    }
```

Member Vo와 MemberDto에 List<Role>을 연관관계로 연동한다.

도메인 설정 종료.

시큐리티설정 시작

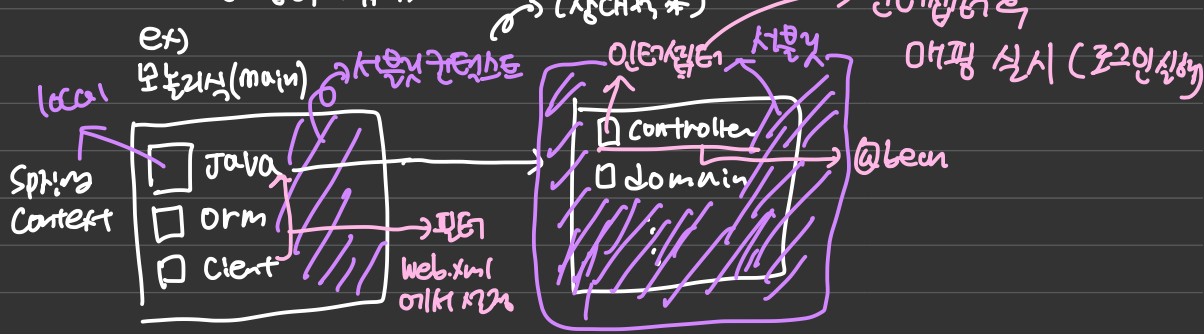
5/4 시큐리티 2.

복습 Spring 시큐리티 처리.

- 필터와 인터셉터를 이용한다. ← 필터가 먼저!
- 서블릿 컨텍스트(JSP+컨트롤러) 내부의 스프링 컨텍스트(Spring-context.xml) 에 인터셉터가 있다.

서블릿 C > 스프링 C

ex) 모놀리식(main) → 서블릿 컨텍스트 (상대적, *)

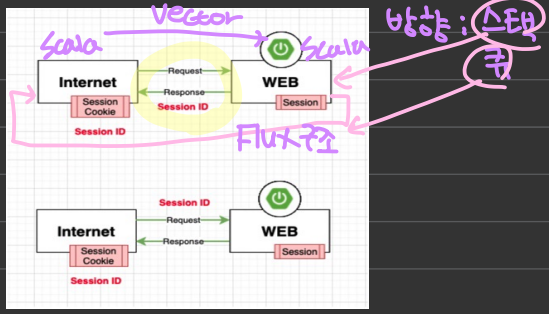


*토큰의 역할: 필터와 인터셉터를 통과하게 위한 키의 기능.

뚫림 legacy : CRSF 토큰 → 컨트롤러에 생성 → Session 저장방식.

JWT : Bean에 생성

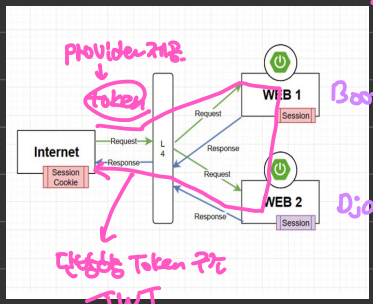
Session 모놀리식



방향: 스테틱 쿠

JWT 마이크로

한번의 결제(token) 2번째→ 4번째 → 8번째...



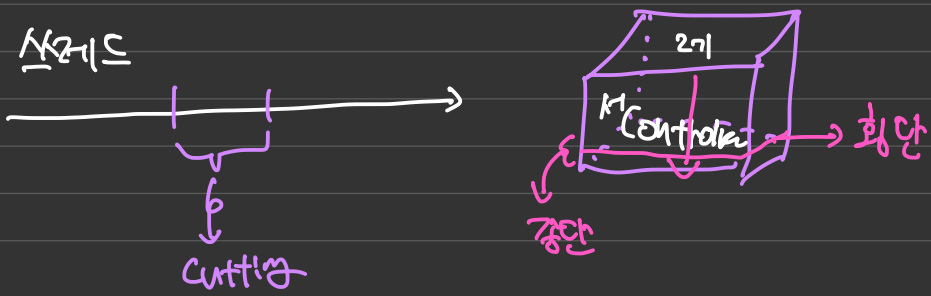
OOP : 클래스기반 / FP : 함수기반

→ OOP

컴퓨팅에서 관점 지향 프로그래밍(aspect-oriented programming, AOP)은 횡단 관심사(cross-cutting concern)의 분리를 허용함으로써 모듈성을 증가시키는 것이 목적인 프로그래밍 패러다임이다. 코드 그 자체를 수정하지 않는 대신 기존의 코드에 추가 동작(어드바이스)을 추가함으로써 수행하며, "함수의 이름이 'set'으로 시작하면 모든 함수 호출을 기록한다"와 같이 어느 코드가 포인트컷(pointcut) 사양을 통해 수정되는지를 따로 지정한다. 이를 통해 기능의 코드 핵심부를 어수선하게 채우지 않고도 비즈니스 로직에 핵심적이지 않은 동작들을 프로그램에 추가할 수 있게 한다. 관점 지향 프로그래밍은 관점 지향 소프트웨어 개발의 토대를 형성한다.
관점 지향 소프트웨어 개발은 온전한 엔지니어링 분야를 가리키는 반면에 관점 지향 프로그래밍은 소스 코드 레벨에서 관심사들의 모듈화를 지원하는 프로그래밍 메서드들과 도구들을 포함하고 있다.
관심 지향 프로그래밍은 프로그램 로직을 명확한 부분들(이른바 "관심사")로 나누는 것을 수반한다. 거의 모든 프로그래밍 패러다임들은 관심사들을 별도의 독립적인 엔티티로 그룹화하고 캡슐화하는 것을 어느 정도는 지원하며, 이는 이러한 관심사들을 구현, 추상화, 합성하기 위해 사용할 수 있는 추상화(예: 함수, 프로시저, 모듈, 클래스, 메서드)를 제공함으로써 수행된다. 일부 관심사들은 프로그램 내의 여러 추상적 개념들에 영향을 미치며 이러한 형태의 구현체를 거역한다. 이러한 관심사들을 크러스 커팅 관심사(cross-cutting concerns)라고 부른다.
횡단 관심사의 전형적인 예로 로깅을 들 수 있는데 로깅 전략이 필연적으로 시스템 상에서 로그되는 모든 부분에 영향을 미치기 때문이다. 그러므로 로깅은 로그가 되는 모든 클래스들과 메서드들을 횡단한다.
모든 AOP 구현체들은 각각의 관심사를 한 자리에 캡슐화하는 횡단 표현식들을 일부 보유하고 있다. 구현체들 간의 차이점은 제공되는 생성자들의 권한, 보안, 사용성에 기인한다.

Concern 은 제거해야할 대상 이다.
ㄴ반복적인, 사전조건, 사후조건 ⇒ Cutting 해야한다 + 불리해야한다.

⇒ Concern과 Core logic 을 분리한 상태의 코딩 : Aspect = Concern + Core



프록시 패턴

AOP

- 원래 하려던 기능을 수행하며 그외의 부가적인 작업(로깅, 인증, 네트워크 통신 등)을 수행하기에 좋습니다.
- 비용이 많이 드는 연산(DB 쿼리, 대용량 텍스트 파일 등)을 실제로 필요한 시점에 수행할 수 있습니다.
- 사용자 입장에서는 프록시 객체나 실제 객체나 사용법은 유사하므로 사용성이 좋습니다.

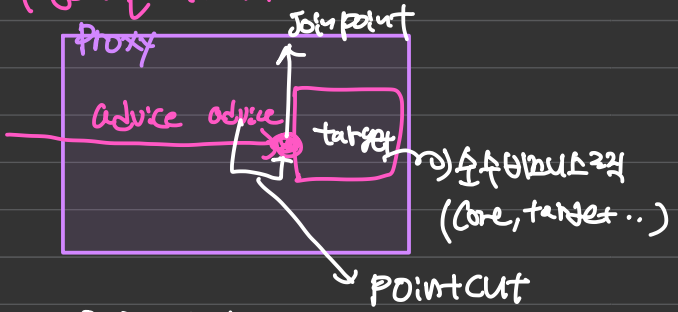
→ 따로 클래스화 : advice

⇒ Concern과 Core를 나누어서 Concern을 제공하는 패턴 (target)

조인되는지점 조인 포인트

- Real Object, Proxy Object는 동일한 인터페이스를 구현합니다.
- Proxy Object는 메서드 수행시 실제 객체(Real Object)의 메서드에 위임합니다.

↳ 위임패턴 ≒ 프록시



로드밸런싱이 Concern을 정의해 놓음.
↳ 대표적인 Filter / Interceptor ..

⇒ AOP는 Core 로직을 먼저 작성한 Concern을 결합하여 로직이 완성된 상태

Concern은 OOP 로직에서 공통으로 중복된 부분이다.나눗셈 로직에서 0으로 나누는 코드를 if 문으로 걸러내는 코드가 Concern에 해당한다.[코드1] cross-cutting concern의 전형적인 예는 로깅이다. 로깅은 로그가 되는 모든 클래스들과 메서드들을 횡단한다.

AOP은 cross-cutting concern의 분리해서 모듈성을 증가시키는 프로그래밍이다. 그리고 Concern들의 모듈화 프로그래밍 개념 포함하고 있다. AOP은 관심사로 나누는 것을 수반한다.
Concern을 별도의 독립적인 Entity로 그룹화하고 캡슐화한다. 이는 Concern들에 함수를 제공함으로써 수행된다. 모든 AOP 구현체들은 각각의 Concern를 캡슐화하는 횡단 표현식들을 가지고 있다. 구현체들 간의 차이점은 제공되는 생성자들의 권한, 보안, 사용성에 기인한다.

Target 은 Business Logic 에서 Conern 이 제거된 Core 이며, JoinPoint 메소드를 갖는다.
Target 을 Wrapping 하는 객체가 Proxy 이다. 외부에서 호출은 Proxy를 통해 내부적으로 Target 의 JoinPoint를 호출한다. 이 때 JoinPoint 중에서 Ascpect을 구현한 Advice 와 결합되는 코드가 PointCut 이다. Cross-cutting concern은 추상적 개념의 구현체를 거역한다. OOP 에 추가 동작(어드바이스)을 추가함으로써 수행하며, "함수의 이름이 'set'으로 시작하면 모든 함수 호출을 기록한다"와 같이 어느 코드가 포인트컷(pointcut) 사양을 통해 수정되는지를 따로 지정한다.

Ant 표기법

? : 1개의 문자와 매칭 (matches single character)
* : 0개 이상의 문자 또는 파일와 매칭 (matches zero or more characters)
** : 0개 이상의 디렉토리 매칭 (matches all files / directories)

DTO → Entity로 바꿔주는것이 Model Mapper.

model mapping. → <K, V>

map → JSON 바꿔줌.

Why Map?

Applications often consist of similar but different object models, where the data in two models may be similar but the structure and concerns of the models are different. Object mapping makes it easy to convert one model to another, allowing separate models to remain segregated.

Why ModelMapper?

The goal of ModelMapper is to make object mapping easy, by automatically determining how one object model maps to another, based on conventions, in the same way that a human would - while providing a simple, refactoring-safe API for handling specific use cases.