**Interceptor**

HandlerInterceptor 를 구현하거나 HandlerInterceptorAdapter 를 상속받아서 구현한다

Aop와 유사한 기능을 제공하게 된다

각 메소드마다 공통적으로 HttpServletRequest 와 HttpServletResponse 를 가지고 있다

@Override 를 사용해서 구현해야 작동이 된다

preHandle - 컨트롤러의 메소드가 호출되기전에 preHandle이 호출된다 boolean형

postHandle - 컨트롤러 메소드가 호출되기전에 postHandle이 호출된다 void형

afterHandle - View 처리가 완료되면 호출된다 void형

spring mvc 관련설정에서 Interceptor를 사용하겠다고 등록을 해줘야한다

//인터셉터 등록

**public** **void** addInterceptors(InterceptorRegistry re) {

WebMvcConfigurer.**super**.addInterceptors(re);

Interceptor1 i1=**new** Interceptor1();

inter2 i2=**new** inter2();

InterceptorRegistration r1=re.addInterceptor(i1);

r1.addPathPatterns("/t1");

// addPathPatterns는 여러개를 동시에 등록가능

// excludePathPatterns 는 입력된 경로를 제외한다

}

**properties 를 활용한 db 연결**

**값이 넣어질 db와 같은 구조의 객체생성,**

프로퍼티 파일을 만들어서 db연결에 필요한 driver, url, name, password 를 작성한다

서블릿에서 @PropertySource를 사용 프로퍼티파일의 경로를 직접 지정한다

인터페이스를 생성

어노테이션 ( @Insert , @Select 등) 을 활용하여 db쿼리문을 작성하고 메소드를 만든다

예 )@Insert("insert into mvc\_table(d1,d2,d3) values(#{d1},#{d2},#{d3})")

**void** in\_sert(Data data);

서블릿에서 @Value 로 properties에 작성된 값들을 가져온다

@Value("${db.classname}")

**private** String db\_classname;

서블릿에서 db연결을 위한 @Bean 등록을 해준다 ( 고정된 작업 이므로 생성한 인터페이스의 이름만 변경된다

@Bean

**public** BasicDataSource datasource() {

BasicDataSource so = **new** BasicDataSource();

so.setDriverClassName(db\_classname);

so.setUrl(db\_url);

so.setUsername(db\_username);

so.setPassword(db\_password);

**return** so;

}

@Bean

**public** SqlSessionFactory fac(BasicDataSource so) **throws** Exception{

SqlSessionFactoryBean fac = **new** SqlSessionFactoryBean();

fac.setDataSource(so);

SqlSessionFactory f = fac.getObject();

**return** f;

}

@Bean

**public** MapperFactoryBean<mapperInterface> mapper(SqlSessionFactory fac) **throws** Exception {

MapperFactoryBean<mapperInterface> fa = **new** MapperFactoryBean<mapperInterface>(mapperInterface.**class**);

fa.setSqlSessionFactory(fac);

**return** fa;

}

\* xml 파일로 작성시 WEB-INF에서 mapper폴더와 mapper.xml 파일을 만든 뒤 sql문을 작성한다

서블릿.xml 에서 같은 설정을 xml 에 맞게 작성하면 작동됨

쿠키 복습

Cookie cookie = new Cookie(); 를 통해 생성하며

response를 통해 addCookie로 브라우저에 등록한다

사용하기 위해 가져올 때는 @CookieValue 를 통해

설정한 이름으로 값을 가져오거나

request.getCookies로 가져 올 수있다

STS 를 3.9.11 버전 다운 받고 lombok을 홈페이지에서 다운 받는다

lombok.jar 를 실행해서 sts 를 경로지정해주면 eclipse 디렉토리에

lombok이 추가가 된다

STS 에서 프로젝트를 legacy 프로젝트,spring mvc 로 생성한뒤

pom.xml 에서 스프링 프레임 워크를 5.2.7 버전,

junit 을 4.12 버전, maven 을 1.8 버전으로 버전을 변경한다

lombok이 maven 에 없을 시 <https://mvnrepository.com/> 에서 lombok을

검색해서 pom.xml 에 추가해준다

lombok 을 사용하면 @ToString 을 사용할 수 있고

override 를 하지않아도 객체를 주소값이 아닌 문자로 받을 수있다