강의 시간 : 15시간 ( 8 + 7 )

쉬는 시간 : 50분 ~ 정각

점심시간 11:50 ~ 13시

QR 태깅 13시 , 5:50 이후 퇴실

환경설정

Open JDK 11 ( unzip )

Intellij IDEA CE ( Install )

H2(memory db), MariaDB ( Install )

PostMan ( Install )

1. open JDK11 : openjdk-11+28\_windows-x64\_bin.zip 파일을 unzip 하시면 됩니다. 기존에 사용하시던 jdk가 있다면 안하셔도 됩니다.

2. Intellij IDEA CE : ideaIC-2021.2.3.zip 압축을 풀어서 인스톨하시면 됩니다. 기존에 사용하시던 IDEA 툴이 있다면 안하셔도 됩니다.

3. MariaDB10.3 : mariadb-10.3.11-winx64.zip 압축을 풀어서 인스톨하시면 됩니다.

MariaDB 설치시 유의사항

: Root 패스워드 maria / maria

: Use UTF-8 as default server's character set 반드시 체크 해주어야 한글 깨지지 않습니다.

4. Postman : Postman-win64-Setup.zip 압축을 풀어서 인스톨 하시면 됩니다.

Spring Initializer에서 SpringBoot 프로젝트를 생성할 때 추가 할 의존성

webmvc, data jpa, h2, mariadb, thymeleaf, validation, devtools, lombok (only for Dev)

C:\MySpringBoot\_App 폴더 아래에 Source Code 작성

SpringBoot 튜토리얼

<https://docs.spring.io/spring-boot/docs/current/reference/html/>

SpringBoot API Doc

<https://docs.spring.io/spring-boot/docs/current/api/>

Spring Boot가 선택한 Third-Party 라이브러리

Connection Pooling ( CP ) => HikariCP ( <https://github.com/brettwooldridge/HikariCP> )

JSON Parser => Jackson ( <https://github.com/FasterXML/jackson> )

Logging => Facade ( SLF4j https://github.com/qos-ch/slf4j)

구현체 ( Logback https://github.com/qos-ch/logback)

Spring Boot 프로젝트 => unzip => IDE 설정

@SpringBootApplication =

: **@SpringBootConfiguration + @EnableAutoConfiguration + @ComponentScan**

@Configuration ( JavaConfig는 XML 대신 클래스에 설정이 가능하도록 )

: XML 대신 Spring Bean에 대한 정보를 클래스에 선언한다.

: Java Config

: @ComponentScan

: @Component, @Service, @Repository,

@Autowired, @Resource

@Bean 과 @Component 의 차이점

: Spring Bean 임을 나타내는 어노테이션

: @Component => 클래스에 선언

: @Bean => 메서드에 선언

@Bean

public Docket api() {

return new Docket()

}

(Swagger 는 REST API docs 를 생성해주는 역할 )

**@EnableAutoConfiguration**

: SpringBoot가 제공하는 여러개의 AutoConfiguration 들을 활성화 시킨다.

Spring Boot와 Spring Framework의 차이점

=> Spring Framework : XML, Java Config 에 Bean Configuration 정보를 설정한다.

=> Spring Boot : Java Config 에 Bean Configuration 정보를 설정한다.

XML의 역할?? - 개발자가 작성한 클래스에 여러정보 Meta정보

HelloController cont = new HelloController(); (X)

HelloController 클래스에 대한 정보를 알려줘

terminal을 열고 (Only Cmd 에서만 가능합니다)

환경변수 설정

set JAVA\_HOME=C:\MySpringBoot\jdk-11

set PATH=%JAVA\_HOME%\bin;%PATH%

mvnw package

java -jar .\target\myspringbootApp-0.0.1-SNAPSHOT.jar

java -jar -Dbar .\target\myspringbootApp-0.0.1-SNAPSHOT.jar --foo

//4번째 우선순위

java -jar .\target\myspringbootApp-0.0.1-SNAPSHOT.jar --myboot.name=클라우드

java -jar .\target\myspringbootApp-0.0.1-SNAPSHOT.jar --spring.profiles.active=prod

**Gradle Project 에서의 jar 실행**

java -jar .\build\libs\myspringboot-gradle-app-0.0.1-SNAPSHOT.jar --spring.profiles.active=prod

.properties(yml) 설정된 환경변수 읽어 오는 방법

1. @Value 어노테이션 사용하기

: @Value(“${server.port}”)

String portNumber;

1. Environment의 getProperty() 메서드 사용하기

@Autowired

Environment env;

String portNumber = env.getProperty(“server.port”);

1. 사용자 정의 Properties 클래스 작성하기

함수형 인터페이스 ( Functional Interface )

=> 추상메서드를 하나만 가지고 있는 인터페이스

=> 함수형 인터페이스의 추상메서드를 재정의 하는 구문을 람다식으로 표현 할 수 있다.

**Dependency Injection(DI)** 유형

1. Setter Injection

: setter method, 변수 위에 @Autowired, @Resource 선언

@Autowired 는 Type 기반으로 의존하는 객체를 찾아서 주입

@Resource는 Bean의 name(id)으로 의존하는 객체를 찾아서 주입

2. Constructor Injection

: argument가 있는 생성자를 선언한다.

2-1. Lombok의 @RequiredArgsConstructor 어노테이션을 사용해도 가능

-----------------------------------------------

J2EE API Docs

<https://jakarta.ee/specifications/platform/8/apidocs/>

ORM(Object Relational Mapping)

Mapping Rule

Entity Class ⇔ Table

Entity Object ⇔ Row(Record)

Entity Variables ⇔ Columns

JPA(Java Persistence API)

: ORM 기능을 위한 J2EE 표준 스펙

Spring Data JPA

: Spring Boot 개발자가 Hibernate를 좀 더 쉽게 사용할 수 있도록

Hibernate 구현체를 더 추상화 한 라이브러리

Hibernate

: JPA 표준스펙을 구현한 ORM 구현체

JPA Repository query method 메소드 명 규칙

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods

CrudRepository의 메소드에 대한 설명

등록: <S **extends** T> S save(S entity);

리스트 조회: Iterable<T> findAll();

PK로 조회: Optional<T> findById(ID id);

삭제: **void** delete(T entity);

Dialect(방언, 사투리)

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDB103Dialect

public enum GenerationType {

/\*\*

\* Indicates that the persistence provider must assign

\* primary keys for the entity using an underlying

\* database table to ensure uniqueness.

\*/

*TABLE*,

/\*\*

\* Indicates that the persistence provider must assign

\* primary keys for the entity using a database sequence.

\*/

*SEQUENCE*,

/\*\*

\* Indicates that the persistence provider must assign

\* primary keys for the entity using a database identity column.

\*/

*IDENTITY*,

/\*\*

\* Indicates that the persistence provider should pick an

\* appropriate strategy for the particular database. The

\* <code>AUTO</code> generation strategy may expect a database

\* resource to exist, or it may attempt to create one. A vendor

\* may provide documentation on how to create such resources

\* in the event that it does not support schema generation

\* or cannot create the schema resource at runtime.

\*/

*AUTO*

}

**GenerationType.Auto**

MariaDB [boot\_db]> desc account;

+----------+--------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+--------------+------+-----+---------+-------+

| id | bigint(20) | NO | PRI | NULL | |

| password | varchar(255) | YES | | NULL | |

| username | varchar(255) | YES | UNI | NULL | |

+----------+--------------+------+-----+---------+-------+

hibernate\_sequence 테이블이 추가로 생성된다.

MariaDB [boot\_db]> desc hibernate\_sequence;

+-----------------------+---------------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-----------------------+---------------------+------+-----+---------+-------+

| next\_not\_cached\_value | bigint(21) | NO | | NULL | |

| minimum\_value | bigint(21) | NO | | NULL | |

| maximum\_value | bigint(21) | NO | | NULL | |

| start\_value | bigint(21) | NO | | NULL | |

| increment | bigint(21) | NO | | NULL | |

| cache\_size | bigint(21) unsigned | NO | | NULL | |

| cycle\_option | tinyint(1) unsigned | NO | | NULL | |

| cycle\_count | bigint(21) | NO | | NULL | |

+-----------------------+---------------------+------+-----+---------+-------+

**GenerationType.Identity** ( auto increment column )

+----------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+----------+--------------+------+-----+---------+----------------+

| id | bigint(20) | NO | PRI | NULL | auto\_increment |

| password | varchar(255) | YES | | NULL | |

| username | varchar(255) | YES | UNI | NULL | |

+----------+--------------+------+-----+---------+----------------+

*@RestController = @Controller + @ResponseBody*

Response (응답) : *@ResponseBody*

: Java Object -> JSON 변환 처리는 Jackson이 담당 함

: 변환된 데이터를 응답(response) body에 담아 주는 역할

Request (요청) : *@RequestBody*

: JSON -> Java Object 변환 처리는 Jackson이 담당함

: 변환된 데이터를 요청(request)에 담아서 컨트롤러의 메서드의 아규먼트로 매핑 해주는 역할

변환 처리가 누가 할까요? Jackson

등록

POST

<http://localhost:8087/users>

header

content-type:application/json

body

{

"name":"스프링",

"email":"spring@a.com"

}

Id로 조회 GET

<http://localhost:8087/users>/{id}

email로 조회 GET

<http://localhost:8087/users>/email/spring@a.com/

목록조회 GET

<http://localhost:8087/users>

수정 PATCH

header

content-type:application/json

body

<http://localhost:8087/users>/{email}

{

"name":"Spring"

}

삭제 DELETE

<http://localhost:8087/users>/{id}

<http://localhost:8087/users>/xml

Thymeleaf Docs

<https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>

Thymeleaf-Spring Docs

https://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html

J2EE API Docs

<https://jakarta.ee/specifications/platform/8/apidocs/>

Hibernate Validator API

https://docs.jboss.org/hibernate/validator/6.2/api/

Spring Framework5 API Docs

<https://docs.spring.io/spring-framework/docs/5.3.24/javadoc-api/>

javax validation의 어노테이션

@NotEmpty vs @NotBlank

@NotEmpty : white space(“ “) 는 허용

@NotBlank : 문자열의 공백을 제거(trim)하고 체크 하므로

white space(“ “) 허용하지 않음

—------------------------

CORS(Cross Origin Resource Sharing)

=> SOP를 우회하기 위한 방법

Ajax(Asynchronous Javascript and Xml)

: Xml(JSON) - 서버에서 보내준 데이터

: Asynchronous - 비동기 XmlHttpRequest(XHR)

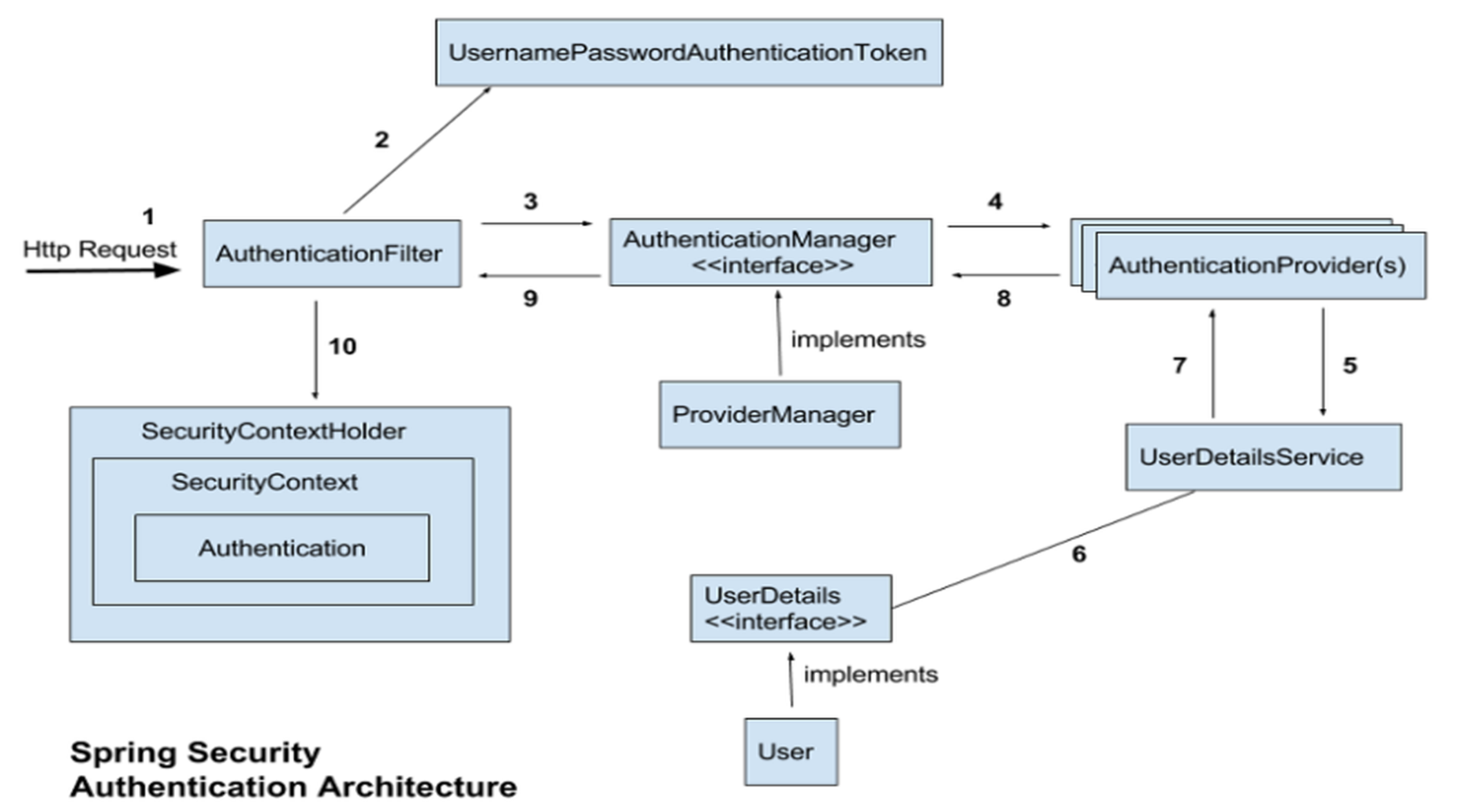
: Javascript - html을 동적으로 생성

: SOP(Single Origin Policy)

Client (ReactJS) -> SpringBoot(REST API)

<http://localhost:3000> -> <http://localhost:8087>

Spring Security Architecture



**IDEA 단축키**

**ctrl + shift + f10 : Java run**

**ctrl + shift + t : 테스트 케이스 추가**

**ctrl + alt + o(오우) : auto import**

**ctrl + alt + v : return type 자동 생성 ( eclipse 는 alt + shift + l(엘) )**

**ctrl + alt + shift + l(대문자엘) : code format**

**alt + insert : generate constructor, getter, setter**

**alt + enter : import , create new class(interface, enum)**

**alt + shift + insert : column selection mode**

**ctrl + alt + m : extract method**

**ctrl + alt + n : inline method**

**ctrl + o (오우) : override method**

**ctrl + i (아이) : implements method**

IDEA 설정

1. JDK11 위치 알려주기

File -> Project Structure

: project -> project sdk - Add sdk

1. File -> Settings

: Font 설정

: Lombok 설정

**Plugins > Lombok Plugin install** 확인

Plugins > installed

Lombok 설치된 것을 확인한다.

​**Build, Execution, Deployment > Compiler > Annotation Processors >**

**Enable annotation processing** 체크

: Auto Import 설정

**File -> Settings -> Editor -> General -> Auto Import**

**1) Insert imports on paste 옵션은 코드를 붙여넣기 한 경우의 import문 삽입 옵션으로 Always,**

**Ask, Never을 사용할 수 있다.**

**Always : 모두 자동 import**

**Ask : 확인 후 선택**

**Never : 자동 import 하지 않음**

**2) Add unambiguous imports on the fly 옵션 체크**

: Devtools 관련 설정

​**Build, Execution, Deployment > Compiler**

**- Build Project Automatically 체크합니다.**

**Advanced Settings**

**- Allow auto-make to start even-if developed application is currently running**

: properties file에서 한글이 깨지지 않게 하려면

**Editor -> File Encodings**

**- project encoding 을 utf-8 로 선택**