JPA – RestApi – Security

Pom XML:

```
<dependencies>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jdbc</artifactId>
	</dependency>

	<dependency>
		<groupId>org.apache.tomcat.embed</groupId>
		<artifactId>tomcat-embed-jasper</artifactId>
		<scope>provided</scope>
	</dependency>

	<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
	<dependency>
		<groupId>javax.servlet</groupId>
		<artifactId>jstl</artifactId>
		<version>1.2</version>
	</dependency>

	<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
	<dependency>
		<groupId>mysql</groupId>
		<artifactId>mysql-connector-java</artifactId>
		<version>8.0.18</version>
	</dependency>


	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-rest</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-web</artifactId>
	</dependency>

	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-devtools</artifactId>
		<scope>runtime</scope>
		<optional>true</optional>
```

```
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
			<exclusions>
				<exclusion>
					<groupId>org.junit.vintage</groupId>
					<artifactId>junit-vintage-engine</artifactId>
				</exclusion>
			</exclusions>
		</dependency>


		<!--  Hibernate start -->
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>

		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-core</artifactId>
		</dependency>
		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-entitymanager</artifactId>
		</dependency>
		<!--  Hibernate end -->


		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-security</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-test</artifactId>
			<scope>test</scope>
		</dependency>


</dependencies>
```

Application.java

```
@SpringBootApplication
@EnableJpaAuditing
@EnableJpaRepositories(basePackages = "com.works.repositories")
@EntityScan(basePackages = "com.works.repomodel")
@ComponentScan(basePackages = { "com.works.restusing"})
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Application.properties

```
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp

spring.datasource.url=jdbc:mysql://localhost/dbName?useUnicode=true&
characterEncoding=utf-8&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5
InnoDBDialect
spring.jpa.hibernate.ddl-auto=update
spring.main.allow-bean-definition-overriding=true
```

DBUtil

```
@Configuration
@ComponentScan(basePackages = { "com.works.*" })
@PropertySource("classpath:application.properties")
public class DBUtil {

    @Value("${spring.datasource.url}")
    private String dbUrl;

    @Value("${spring.datasource.username}")
    private String dbUser;

    @Value("${spring.datasource.password}")
    private String dbPass;

    @Bean(name = "db")
    public DriverManagerDataSource source() {
        DriverManagerDataSource dt = new DriverManagerDataSource();
        dt.setDriverClassName("com.mysql.cj.jdbc.Driver");
        dt.setUrl(dbUrl);
        dt.setUsername(dbUser);
        dt.setPassword(dbPass);
        return dt;
    }

}
```

@Entity

```
@Table(name = "cars")
@EntityListeners(AuditingEntityListener.class)
public class Cars {
```

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

CarsRepository.java

```
@Repository
public interface CarsRepository extends JpaRepository<Cars, Long> {

    @Query("select u from User u where u.emailAddress = ?1")
           User findByEmailAddress(String emailAddress);
     }

     @Query("select u from User u where u.firstname like %?1")
       List<User> findByFirstnameEndsWith(String firstname);

     @Query(value = "SELECT * FROM USERS WHERE EMAIL_ADDRESS = ?1",
nativeQuery = true)
       User findByEmailAddress(String emailAddress);
}
```

```
@RestController
@Autowired DriverManagerDataSource db;
@Autowired CarsRepository carsRepository;
@PathVariable Optional<Integer> number

    // cars insert
    @PostMapping("/carsInsert")
    public Cars carsInsert( Cars cr ) {
        //carsRepository.findAll();
        //carsRepository.delete(cr);
        //carsRepository.findById(1)
        return carsRepository.save(cr);
    }
```

```java
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;
import java.util.List;
import java.util.Optional;



@RestController
@RequestMapping("/api/v1/products")
@Slf4j
@RequiredArgsConstructor
public class ProductAPI {
    private final ProductService productService;

    @GetMapping
    public ResponseEntity<List<Product>> findAll() {
        return ResponseEntity.ok(productService.findAll());
    }

    @PostMapping
    public ResponseEntity create(@Valid @RequestBody Product
product) {
        return ResponseEntity.ok(productService.save(product));
    }

    @GetMapping("/{id}")
    public ResponseEntity<Product> findById(@PathVariable Long id) {
        Optional<Product> stock = productService.findById(id);
        if (!stock.isPresent()) {
            log.error("Id " + id + " is not existed");
            ResponseEntity.badRequest().build();
```

```
        }

        return ResponseEntity.ok(stock.get());
    }

    @PutMapping("/{id}")
    public ResponseEntity<Product> update(@PathVariable Long id,
@Valid @RequestBody Product product) {
        if (!productService.findById(id).isPresent()) {
            log.error("Id " + id + " is not existed");
            ResponseEntity.badRequest().build();
        }

        return ResponseEntity.ok(productService.save(product));
    }

    @DeleteMapping("/{id}")
    public ResponseEntity delete(@PathVariable Long id) {
        if (!productService.findById(id).isPresent()) {
            log.error("Id " + id + " is not existed");
            ResponseEntity.badRequest().build();
        }

        productService.deleteById(id);

        return ResponseEntity.ok().build();
    }
}
```

SpringSecurityConfig.java

```
@Configuration
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter   {
    @Autowired DataSource db;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exce
ption {
          auth.jdbcAuthentication().dataSource(db)
        .authoritiesByUsernameQuery("select mail, role from user where mail
 = ?")
        .usersByUsernameQuery("select mail, pass, statu from user where mai
l = ?");
    }



    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http
        .httpBasic()
        .and()
        .authorizeRequests()
        .antMatchers("/**").hasRole("admin")
        //.antMatchers("/user","/userx").hasRole("user")
        .anyRequest()
        .authenticated()
        //.antMatchers("/**").hasAuthority("admin")
        //.antMatchers("/allUser").hasRole("user")
        ;
        //.antMatchers("/").permitAll()
        //.antMatchers("/userAll").hasAnyRole("user")
        //.and().formLogin().loginPage("/login")
```

```
        //;
        http.csrf().disable();
    }

    @Bean
    public static NoOpPasswordEncoder passwordEncoder() {
     return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();
    }
  }
```

Aut Control

```
    public static Object userStatu() {
        Collection<? extends GrantedAuthority> aut = SecurityContextHol
der.getContext().getAuthentication().getAuthorities();
        System.out.println("Data : " + aut.toArray()[0]);

        return aut.toArray();
    }
```

ReponseEntity Fnc

```
private void getEmployeeById()
    {
        long start = System.currentTimeMillis();
        try {
            final String uri = "https://www.jsonbulut.com/json/product.php?
ref=5380f5dbcc3b1021f93ab24c3a1aac24&start=0";
```

```
            Map<String, String> params = new HashMap<String, String>();
            //params.put("id", "1");

            RestTemplate restTemplate = new RestTemplate();
            ResponseEntity<String> result = restTemplate.getForEntity(uri,
String.class);
            Gson gs = new Gson();
            List<Product> prl =  gs.fromJson(result.getBody(), JsonData.cla
ss).getProducts();
            long end = System.currentTimeMillis();
            long statu = end - start;
            System.out.println("RestTemplate : " + statu);
            System.out.println("free RestTemplate: " + rn.freeMemory());
            System.out.println("result : --
" + prl.get(0).getBilgiler().size());

            //ResponseEntity<JsonData> resultt = restTemplate.getForEntity(
uri, JsonData.class);
            //System.out.println("Size : -
- " + resultt.getBody().getProducts().size());

            //result.getProducts().get(0).getBilgiler().forEach(item -> {
            //  System.out.println("restTemplate item : " + item.getProduct
Name());
            //});
        } catch (Exception e) {
            System.err.println("template : " + e);
        }

    }
```

---

http://raufferlobo.com/en/post/caching-data-with-spring-cache

---

@EnableCaching

```
public class Application {
@Cacheable(value = "cacheUser")
@Scheduled(fixedDelay = 10 * 60 * 1000 ,  initialDelay
= 500)
```

```
public interface ExchangeRate {

    Map<String, Double> findExchangeRates();

}
@Service
public class FixerExchangeRate implements ExchangeRate {

    @Cacheable("exchangeRates") // exchangeRates is the cache name
    public Map<String, Double> findExchangeRates() {

        RestTemplate restTemplate = new RestTemplate();

        ResponseEntity<FixerRate> response =
restTemplate.getForEntity("https://api.fixer.io/latest", FixerRate.class);

        FixerRate rate = response.getBody();

        return rate.getRates();

    }

}
@Cacheable("exchangeRates") // exchangeRates is the cache name
public Map<String, Double> findExchangeRates(String base) {

    RestTemplate restTemplate = new RestTemplate();

    ResponseEntity<FixerRate> response =
restTemplate.getForEntity("https://api.fixer.io/latest?base="+base,
FixerRate.class);

    FixerRate rate = response.getBody();

    return rate.getRates();

}
@Cacheable(cacheNames="exchangeRates", key="#filter.base") // value of base
property will be the key
public Map<String, Double> findExchangeRates(Filter filter) {
...
}
@Scheduled(fixedDelay=60 * 60 * 1000) // 1 hour
@CacheEvict(cacheNames="exchangeRates", allEntries=true)
public void evictExchangeRatesCache() {
    // no code is needed here
}
```