MOBILE APP WEB SERVICES SITE WITH SPRING BOOT FRAMEWORK

Report Paper : May 1

Sumeyra Betul Polat - COM-18

Ala-Too International University Computer Science Department

ACKNOWLEDGEMENT

First of all I would like to thank our lecturer 'Nurlan Shaidullaev' for supporting us learning 'Spring Boot Framework' and for supporting us when we need help. Beside from my lecturer, I like to thank my other students for their effort. We, the students, believe that we will be successful citizens and educators if we receive the necessary education to guarantee our future.In order to accomplish some things, it is necessary to work hard and make efforts. We need an environment where we can work hard to survive the challenges and go on new and bright days, and we need supporters who trust us.But success starts with self-confidence, so first of all thanks to our family and teachers.

INTRODUCTION

The main objective behind developing this Online WebSite project in Java is to provide a system for the new products, agencies where they can find the give feedback, available of seeing new brands. Not only this but even the people who love to have some shop/view the brands every day. All they need to do is to log in to the system and then, they can update themselves with the latest updates, brands, amount of products and etc. In the proposed Online Products portal project in Java, while going through the application, the user have login into the system and he/she can choose the choose, edit and delete products, which they are looking for and can buy it.

EXPLANATIONS

- I used Intellij the LINK:
https://www.jetbrains.com/idea/

- MySQL database LINK:
https://dev.mysql.com/downloads/installer/

- Java Version
https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

- Account to log in to Tomcat (Okta)
https://mvnrepository.com/artifact/com.okta.spring/okta-spring-boot-starter

- Apache Tomcat
https://tomcat.apache.org/download-90.cgi

Execution Procedures

We have an AddExceptionsHandler class in our exception package. In the code returns the: ResponseEntity<> within (errorMessage, new HttpHeaders(), HttpStatus.INTERNAL_SERVER_ERROR);

```java
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

import java.util.Date;

@ControllerAdvice
public class AppExceptionsHandler {

    @ExceptionHandler(value = {UserServiceException.class})
    public ResponseEntity<Object>
handleUserServiceServiceException(UserServiceException ex) {
        ErrorMessage errorMessage = new ErrorMessage(new Date(), ex.getMessage());
        return new ResponseEntity<>(errorMessage, new HttpHeaders(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @ExceptionHandler(value = {Exception.class})
    public ResponseEntity<Object> handleUserServiceServiceException(Exception ex)
{
        ErrorMessage errorMessage = new ErrorMessage(new Date(), ex.getMessage());
        return new ResponseEntity<>(errorMessage, new HttpHeaders(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

In the second part, there is a class to give the error message, complete with getter and setter method.

```java
import java.util.Date;

public class ErrorMessage {

    private Date timestamp;
    private String message;

    public ErrorMessage() {
    }

    public ErrorMessage(Date timestamp, String message) {
        this.timestamp = timestamp;
        this.message = message;
    }

    public Date getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(Date timestamp) {
        this.timestamp = timestamp;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

And here is the UserServiceException class to get message with the exception

```java
public class UserServiceException extends RuntimeException {

    public UserServiceException(String message) {
        super(message);
    }
}
```

This class is called the user entity in order to use and call in our project

```java
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import java.io.Serializable;

@Entity(name = "users")
public class UserEntity implements Serializable {

    private static final long serialVersionUID = 3599305611574853751L;

    @Id
    @GeneratedValue
    private long id;

    @Column(nullable = false)
    private String userId;

    @Column(nullable = false, length = 50)
    private String firstName;

    @Column(nullable = false, length = 50)
    private String lastName;

    @Column(nullable = false, length = 100, unique = true)
    private String email;

    @Column(nullable = false)
    private String encryptedPassword;

    private String emailVerificationToken;

    @Column(nullable = false)
    private Boolean emailVerificationStatus = false;

    public long getId() {
        return id;
    }

    public String getUserId() {
        return userId;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getEmail() {
        return email;
    }

    public String getEncryptedPassword() {
        return encryptedPassword;
    }

    public String getEmailVerificationToken() {
        return emailVerificationToken;
    }

    public Boolean getEmailVerificationStatus() {
        return emailVerificationStatus;
    }

    public void setId(long id) {
        this.id = id;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void setEncryptedPassword(String encryptedPassword) {
        this.encryptedPassword = encryptedPassword;
    }

    public void setEmailVerificationToken(String emailVerificationToken) {
        this.emailVerificationToken = emailVerificationToken;
    }

    public void setEmailVerificationStatus(Boolean emailVerificationStatus) {
        this.emailVerificationStatus = emailVerificationStatus;
    }
}
```

Here is our second entity class called AdressEntity this classes values are going to updated, called and used including users.

```java
import javax.persistence.*;
import java.io.Serializable;

@Entity(name = "addresses")
public class AddressEntity implements Serializable {

    private static final long serialVersionUID = -2945997946072035683L;

    @Id
    @GeneratedValue
    private long id;

    @Column(length = 50, nullable = false)
    private String addressId;

    @Column(length = 20, nullable = false)
    private String city;

    @Column(length = 20, nullable = false)
    private String country;

    @Column(length = 100, nullable = false)
    private String streetName;

    @Column(length = 7, nullable = false)
    private String postalCode;

    @Column(length = 15, nullable = false)
    private String type;

    @ManyToOne
    @JoinColumn(name="user_id")
    private UserEntity userDetails;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getAddressId() {
        return addressId;
    }

    public void setAddressId(String addressId) {
        this.addressId = addressId;
    }

    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String getStreetName() {
        return streetName;
    }

    public void setStreetName(String streetName) {
        this.streetName = streetName;
    }

    public String getPostalCode() {
        return postalCode;
    }

    public void setPostalCode(String postalCode) {
        this.postalCode = postalCode;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public UserEntity getUserDetails() {
        return userDetails;
    }

    public void setUserDetails(UserEntity userDetails) {
        this.userDetails = userDetails;
    }
}
```

This class is going to read values from property file

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.env.Environment;
import org.springframework.stereotype.Component;

@Component
public class AppProperties {

    @Autowired
    private Environment env;

    public String getTokenSecret() {
        return env.getProperty("tokenSecret");
    }
}
```

Here is our AuthenticationFilter class, its going to be called only when the code is succesfull

And the message is called only when successfull authentication is conducted, otherwise its going to be called.

Furthermore the Jwts dependecy in pom.xml generates web token, and will send back server response inside. Going to be used in the future requests which requires authentication permissions.

```java
import com.dmytroverner.mobileappws.SpringApplicationContext;
import com.dmytroverner.mobileappws.dto.UserDto;
import com.dmytroverner.mobileappws.model.request.UserLoginRequestModel;
import com.dmytroverner.mobileappws.model.response.ErrorMessages;
import com.dmytroverner.mobileappws.service.UserService;
import com.fasterxml.jackson.databind.ObjectMapper;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.User;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilt
er;

import javax.servlet.FilterChain;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;

public class AuthenticationFilter extends UsernamePasswordAuthenticationFilter {

    private final AuthenticationManager authenticationManager;

    public AuthenticationFilter(AuthenticationManager authenticationManager) {
        this.authenticationManager = authenticationManager;
    }

    @Override
    public Authentication attemptAuthentication(HttpServletRequest request,
                                                HttpServletResponse response)
throws AuthenticationException {
        try {
            UserLoginRequestModel credentials = new ObjectMapper()
                    .readValue(request.getInputStream(),
UserLoginRequestModel.class);
```

```
            return authenticationManager.authenticate(
                    new UsernamePasswordAuthenticationToken(
                            credentials.getEmail(),
                            credentials.getPassword(),
                            new ArrayList<>()
                    )
            );
        } catch (IOException e) {
            throw new
RuntimeException(ErrorMessages.AUTHENTICATION_FAILED.getErrorMessage());
        }
    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request,
                                            HttpServletResponse response,
                                            FilterChain chain,
                                            Authentication authResult) {
        String userName = ((User) authResult.getPrincipal()).getUsername();

        String token = Jwts.builder()
                .setSubject(userName)
                .setExpiration(new Date(System.currentTimeMillis() +
SecurityConstants.EXPIRATION_TIME))
                .signWith(SignatureAlgorithm.HS512,
SecurityConstants.getTokenSecret())
                .compact();
        UserService userService = (UserService)
SpringApplicationContext.getBean("userServiceImpl");
        UserDto userDto = userService.getUser(userName);

        response.addHeader(SecurityConstants.HEADER_STRING,
SecurityConstants.TOKEN_PREFIX + token);
        response.addHeader("UserID", userDto.getUserId());
    }
}
```

Gets the value from Header field named 'Authorization' inside the request And if the header contains 'Messager' Prefix inside its not null

```
import io.jsonwebtoken.Jwts;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import
org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;

public class AuthorizationFilter extends BasicAuthenticationFilter {

    public AuthorizationFilter(AuthenticationManager authenticationManager) {
        super(authenticationManager);
    }

    @Override
    protected void doFilterInternal(HttpServletRequest req,
                                    HttpServletResponse res,
                                    FilterChain chain) throws IOException,
ServletException {
        String header = req.getHeader(SecurityConstants.HEADER_STRING);

        if (header == null || !header.startsWith(SecurityConstants.TOKEN_PREFIX))
{
            chain.doFilter(req, res);
            return;
        }

        UsernamePasswordAuthenticationToken authentication =
getAuthentication(req);
        SecurityContextHolder.getContext().setAuthentication(authentication);
        chain.doFilter(req, res);
    }

    private UsernamePasswordAuthenticationToken
getAuthentication(HttpServletRequest request) {
        String token = request.getHeader(SecurityConstants.HEADER_STRING);

        if (token != null) {
            token = token.replace(SecurityConstants.TOKEN_PREFIX, "");

            String user = Jwts.parser()
                    .setSigningKey(SecurityConstants.getTokenSecret())
                    .parseClaimsJws(token)
                    .getBody()
                    .getSubject();
            if (user != null) {
                return new UsernamePasswordAuthenticationToken(user, null, new
ArrayList<>());
            }
        }
        return null;
    }
}
```

Becouse this class is not a bean in order to get accessed to the environment the Token Secret is going the be readed from application.properties in order to get accessed as i called at the beginning

```java
import com.dmytroverner.mobileappws.SpringApplicationContext;

public class SecurityConstants {
    public static final long EXPIRATION_TIME = 864000000; // 10 days
    public static final String TOKEN_PREFIX = "Bearer ";
    public static final String HEADER_STRING = "Authorization";
    public static final String SIGN_UP_URL = "/users";

    public static String getTokenSecret() {
        AppProperties appProperties = (AppProperties)
SpringApplicationContext.getBean("appProperties");
        return appProperties.getTokenSecret();
    }
}
```