Autenticação e Autorização

A autenticação é o processo pelo qual um sistema verifica a identidade de um usuário, garantindo que ele é quem diz ser. Este processo é essencial para a segurança de qualquer aplicação, pois impede que usuários não autorizados acessem dados ou funcionalidades sensíveis. A autorização, por outro lado, é o mecanismo que determina o que um usuário autenticado pode ou não fazer dentro do sistema. Após a autenticação, a autorização define os níveis de acesso e permissões do usuário. Nesta aula, exploraremos esses conceitos fundamentais e aprenderemos a implementá-los utilizando Spring Boot, garantindo a segurança e a eficiência da nossa aplicação.

Conceito Básico de Autenticação e Autorização

A autenticação é o processo pelo qual um sistema verifica a identidade de um usuário, garantindo que ele é quem diz ser. Este processo é essencial para a segurança de qualquer aplicação, pois impede que usuários não autorizados acessem dados ou funcionalidades sensíveis. Por outro lado, a autorização é o mecanismo que determina o que um usuário autenticado pode ou não fazer dentro do sistema. Após a autenticação, a autorização define os níveis de acesso e permissões do usuário.

Para implementar autenticação e autorização em uma aplicação Spring Boot, é necessário configurar o Spring Security. Um exemplo prático dessa configuração envolve a criação de um filtro de segurança, onde o token de autenticação é extraído das requisições. Esse token é então validado para garantir que o usuário tenha permissão para acessar o recurso solicitado.

Para validar essa configuração, utilizamos ferramentas como o Postman para enviar requisições à API. Primeiro, enviamos uma requisição sem autenticação e esperamos uma mensagem de erro. Em seguida, enviamos uma requisição com um token válido e esperamos um comportamento de sucesso. A configuração do Spring Security envolve anotações específicas, como @Configuration e @EnableWebSecurity, que habilitam a segurança na aplicação.

A seguir, um exemplo de configuração de segurança no Spring Boot:

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .csrf().disable()
            .authorizeRequests()
            .antMatchers("/login").permitAll()
            .anyRequest().authenticated()
            .and()
            .addFilter(new JWTAuthenticationFilter(authenticationManager()));
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

Este exemplo demonstra a desativação do CSRF, a configuração de permissões para o endpoint de login e a adição de um filtro de autenticação JWT. O `PasswordEncoder` é utilizado para encriptar senhas no banco de dados, aumentando a segurança da aplicação.

Tipos de Autenticação e Autorização

Existem diversos tipos de autenticação e autorização, cada um com suas próprias características e níveis de segurança. Os métodos de autenticação incluem:

1. Usuário e Senha: O método mais comum, onde o usuário fornece um nome de usuário e uma senha.

2. Autenticação de Dois Fatores (2FA): Adiciona uma camada extra de segurança, onde além da senha, o usuário deve fornecer um código recebido por SMS ou gerado por um aplicativo autenticador.

3. Autenticação Multifator (MFA): Combina múltiplos métodos de autenticação, como senha, código SMS e perguntas de segurança.

4. Biometria: Utiliza características físicas ou comportamentais únicas do usuário, como impressões digitais ou reconhecimento facial.

Na autorização, os tipos mais comuns são:

1. Token: Um token válido é gerado e utilizado para validar o acesso do usuário a recursos específicos.

2. Políticas: Define permissões baseadas em papéis de usuário, recursos e ambientes de desenvolvimento.

3. Hierarquia: Acesso baseado na posição do usuário dentro de uma organização.

4. Temporal: Acesso concedido por um período de tempo determinado, como 15 minutos após a autenticação.

Um exemplo prático de autenticação de dois fatores no Spring Boot pode ser implementado da seguinte forma:

```java
public class TwoFactorAuthService {
    public boolean validateTotp(String username, String totp) {
        User user = userRepository.findByUsername(username);
        String secret = user.getSecret();
        return TimeBasedOneTimePasswordUtil.validateCurrentNumber(secret, totp, 0);
    }
}
```

Neste exemplo, a validação do TOTP (Time-Based One-Time Password) é realizada utilizando uma chave secreta associada ao usuário, aumentando a segurança da autenticação.

Adição de Autenticação no Projeto

Para adicionar autenticação em um projeto Spring Boot, começamos atualizando o diagrama de entidade-relacionamento para incluir o atributo `username` na tabela de usuários. Em seguida, adicionamos as dependências necessárias no arquivo `pom.xml`:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

Após configurar as dependências, adaptamos a entidade `User` para implementar `UserDetails` do Spring Security, que exige a implementação de métodos obrigatórios, como `getAuthorities`, `isAccountNonExpired`, `isAccountNonLocked`, `isCredentialsNonExpired` e `isEnabled`.

Um exemplo da classe `User` implementando `UserDetails`:

```java
@Entity
public class User implements UserDetails {
    // Atributos como id, username, password, etc.

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return Collections.singleton(new SimpleGrantedAuthority("ROLE_USER"));
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }


    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

Além disso, configuramos o serviço de autenticação para gerenciar usuários e validar credenciais. No `AuthenticationService`, implementamos métodos como `loadUserByUsername` e `generateToken` para criar tokens

JWT. A implementação do filtro de segurança (`SecurityFilter`) garante que cada requisição seja validada com base no token fornecido.

Validação da Autenticação

A validação da autenticação é crucial para garantir que apenas usuários autorizados acessem recursos específicos. Utilizamos o `AuthenticationManager` do Spring Security para gerenciar o processo de autenticação e validação. Durante a inicialização da aplicação, verificamos se todas as configurações de segurança estão corretas e se as dependências necessárias estão presentes.

Um exemplo de validação de autenticação utilizando JWT:

```java
@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    @Autowired
    private JwtTokenProvider tokenProvider;

    @Override
     protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {
        String token = tokenProvider.getTokenFromRequest(request);
        if (token != null && tokenProvider.validateToken(token)) {
            Authentication authentication = tokenProvider.getAuthentication(token);
            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
        filterChain.doFilter(request, response);
    }
}
```

Neste exemplo, o filtro `JwtAuthenticationFilter` intercepta cada requisição, extrai o token JWT do cabeçalho da requisição e valida sua autenticidade. Se o token for válido, a autenticação é estabelecida no contexto de segurança da aplicação, permitindo o acesso ao recurso solicitado.

Esses conceitos e implementações formam a base para a criação de uma aplicação segura utilizando autenticação e autorização com Spring Boot.

GitHub da Disciplina

Você pode acessar o repositório da disciplina no GitHub a partir do seguinte link:

https://github.com/FaculdadeDescomplica/Framework. Esse espaço é o seu portal para mergulhar fundo no universo da aprendizagem interativa. Nele, você encontrará todos os códigos, além dos links para os arquivos e dados.

Conteúdo Bônus

Para aprofundar o conhecimento sobre Autenticação e Autorização, recomendo o seguinte conteúdo bônus gratuito:

Título: Building a RESTful Web Service with Spring Boot

Plataforma: Spring.io

Este tutorial oficial da Spring proporciona uma introdução prática ao desenvolvimento de uma API RESTful utilizando Spring Boot. O material abrange desde a configuração inicial do projeto até a implementação de autenticação e autorização, oferecendo uma base sólida para entender como esses conceitos são aplicados no desenvolvimento real de software.

Razões para recomendação:

1. Prático e didático: O tutorial é estruturado de forma prática e didática, ideal para alunos de graduação.

2. Atualizado: Sendo material oficial da Spring, está sempre atualizado com as melhores práticas e últimas versões do framework.

3. Gratuito e acessível: Totalmente gratuito e disponível online, permitindo fácil acesso e estudo a qualquer momento.

4. Exemplos práticos: Inclui exemplos de código e exercícios que ajudam a consolidar o aprendizado.

5. Compreensivo: Cobre desde conceitos básicos até implementações avançadas, proporcionando uma compreensão abrangente do tema.

Referência Bibliográfica

CARDOSO, L. da C. Design de aplicativos. Intersaberes: 2022

ELMASRI, R.; NAVATHE, S. B. Sistemas de banco de dados. 7.ed. Pearson: 2018.

JOÃO, B. do N. Usabilidade e interface homem-máquina. Pearson: 2017

LEAL, G. C. L. Linguagem, programação e banco de dados: guia prático de aprendizagem. Intersaberes: 2015.

MEDEIROS, L. F. de. Banco de dados: princípios e prática. Intersaberes: 2013.

PUGA, S.; FRANÇA, E.; GOYA, M. Banco de dados: implementação em SQL, PL/SQL e Oracle 11g. Pearson: 2013.

SETZER, V. W.; SILVA, F. S. C. Bancos de dados. Blucher: 2005.

VICCI, C. (Org.). Banco de dados. Pearson: 2014.

Ir para exercício