Aula 21: Configurando Autenticação com SecurityContextHolder

September 18, 2025

1 Introdução

Na Aula 20, implementamos a validação do token JWT no filtro `SecurityFilter`, extraindo o sujeito (login do usuário) do token. Nesta aula, configuraremos o `Spring Security` para liberar o endpoint `/login` sem autenticação e exigir autenticação para os demais endpoints. Além disso, usaremos o `SecurityContextHolder` no `SecurityFilter` para registrar o usuário autenticado com base no token validado. As alterações serão feitas no `Visual Studio Code` ou sua IDE preferida, com testes no Insomnia.

2 Configurando o `SecurityConfigurations`

No `SecurityConfigurations` (pacote `med.voll.api.infra.security`), atualizamos o método `securityFilterChain` para definir regras de autorização:

```
package med.voll.api.infra.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
    org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class SecurityConfigurations {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
        Exception {
        return http.csrf().disable()
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and().authorizeRequests()
                .antMatchers(HttpMethod.POST, "/login").permitAll()
                .anyRequest().authenticated()
                .and().build();
    }
}
```

2.1 Explicação do Código

- `.authorizeRequests()`: Configura regras de autorização para requisições.

- `.antMatchers(HttpMethod.POST, "/login").permitAll()`: Libera o endpoint `POST /login` sem autenticação.
- `.anyRequest().authenticated()`: Exige autenticação para todas as outras requisições.
- **Observação**: O método `authorizeRequests` está obsoleto em versões recentes do Spring Security. Na atividade complementar "authorizeRequests deprecated", o instrutor explica como usar a nova API (ex.: `authorizeHttpRequests`).

3 Atualizando o `SecurityFilter`

No `SecurityFilter` (pacote `med.voll.api.infra.security`), ajustamos o método `recuperarToken` para tratar a ausência do cabeçalho `Authorization` e adicionamos a lógica de autenticação com `SecurityContextHolder`:

```
package med.voll.api.infra.security;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import med.voll.api.domain.usuario.UsuarioRepository;
import org.springframework.beans.factory.annotation.Autowired;
import
    org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;
import java.io.IOException;

@Component
public class SecurityFilter extends OncePerRequestFilter {

    @Autowired
    private TokenService tokenService;

    @Autowired
    private UsuarioRepository repository;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
        HttpServletResponse response, FilterChain filterChain) throws
        ServletException, IOException {
        var tokenJWT = recuperarToken(request);
        if (tokenJWT != null) {
            var subject = tokenService.getSubject(tokenJWT);
            var usuario = repository.findByLogin(subject);
            var authentication = new UsernamePasswordAuthenticationToken(usuario,
                null, usuario.getAuthorities());
            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
        filterChain.doFilter(request, response);
    }

    private String recuperarToken(HttpServletRequest request) {
        var authorizationHeader = request.getHeader("Authorization");
        if (authorizationHeader != null) {
            return authorizationHeader.replace("Bearer ", "");
        }
        return null;
    }
}
```

3.1 Explicação do Código

- `@Autowired private UsuarioRepository repository`: Injeta o repositório para buscar o usuário pelo login.
- `recuperarToken`: Retorna `null` se o cabeçalho `Authorization` estiver ausente, evitando exceção.
- `if (tokenJWT != null)`: Valida o token apenas se presente.
- `tokenService.getSubject(tokenJWT)`: Obtém o sujeito (login) do token.
- `repository.findByLogin(subject)`: Busca o usuário no banco de dados.
- `UsernamePasswordAuthenticationToken`: Cria um objeto de autenticação com o usuário e suas permissões.
- `SecurityContextHolder.getContext().setAuthentication`: Registra o usuário como autenticado no contexto do Spring.

4 Testando as Configurações

No Insomnia, testamos duas requisições:

- **Efetuar Login** (`POST http://localhost:8080/login`):

```
{
    "login": "usuario@voll.med",
    "senha": "123456"
}
```

  Retorna código 200 (OK) com o token JWT encapsulado, pois o endpoint está liberado (`permitAll()`).

- **Listagem de Médicos** (`GET http://localhost:8080/medicos`): Com o token no cabeçalho `Authorization: Bearer <token>`, retorna código 200 (OK) e o JSON:

```
[
    {
        "id": 1,
        "nome": "Nome do Médico",
        "email": "medico1@voll.med",
        "crm": "123456",
        "especialidade": "ORTOPEDIA"
    },
    ...
]
```

  Sem o token, retorna erro 403 (Forbidden), pois o endpoint exige autenticação.

5 Próximos Passos

Configuramos o `Spring Security` para liberar o endpoint `/login` e exigir autenticação para os demais, usando o `SecurityContextHolder` para registrar o usuário autenticado. Na próxima aula, abordaremos a atualização do código para a nova API do Spring Security, substituindo `authorizeRequests` devido à sua depreciação. Continue praticando no `Visual Studio Code` ou sua IDE preferida!

6 Dica do Professor

- **Aprofunde-se em Autenticação**: Consulte a documentação do Spring Security sobre `SecurityContextHo` (`https://docs.spring.io/spring-security/reference/servlet/authentication/architecture.html#servlet-securitycontextholder`) para entender o contexto de autenticação.
- **Comunidade no X**: Siga perfis como @SpringSecMaster e @AuthGuru no X para dicas sobre autenticação e segurança em APIs.
- **Pratique**: No Insomnia, teste a requisição `GET /medicos` com e sem o token no cabeçalho `Authorization` e observe as respostas (200 OK ou 403 Forbidden).