Aula 12: Configurando o Spring Security para Autenticação Stateless

September 18, 2025

1 Introdução

Na Aula 11, criamos a interface `UsuarioRepository` e a classe `AutenticacaoService` para suportar a autenticação com `Spring Security`. Nesta aula, configuraremos o `Spring Security` para operar em modo **stateless**, adequado para APIs REST, desabilitando o comportamento padrão (formulário de login e bloqueio de requisições) e a proteção contra CSRF. Também organizaremos o pacote `infra` em subpacotes para melhor estruturação. As alterações serão feitas no `Visual Studio Code` ou sua IDE preferida, com testes no Insomnia e no navegador.

2 Organizando o Pacote `infra`

Para manter o pacote `med.voll.api.infra` organizado, criamos dois subpacotes:

- `infra.exception`: Para a classe `TratadorDeErros`, que lida com exceções.
- `infra.security`: Para configurações de segurança do `Spring Security`.

Movemos a classe `TratadorDeErros` para o pacote `med.voll.api.infra.exception` usando a funcionalidade de refatoração da IDE. A nova estrutura é:

```
med.voll.api.infra
├── exception
│   └── TratadorDeErros.java
├── security
```

3 Criando a Classe `SecurityConfigurations`

No pacote `med.voll.api.infra.security`, criamos a classe `SecurityConfigurations` para definir as configurações do `Spring Security`. O código é:

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

import org.springframework.context.annotation.Configuration;
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().build();
    }
}
```

3.1 Explicação do Código

- `@Configuration`: Marca a classe como uma configuração do Spring, carregada automaticamente ao iniciar o projeto.
- `@EnableWebSecurity`: Habilita a personalização das configurações de segurança do `Spring Security`.
- `@Bean`: Indica que o método `securityFilterChain` retorna um objeto gerenciado pelo Spring.
- `SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception`: Configura o processo de autenticação e autorização.
- `http.csrf().disable()`: Desativa a proteção contra CSRF, já que usaremos tokens JWT, que oferecem proteção equivalente.
- `sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)`: Define a autenticação como **stateless**, sem armazenamento de sessões no servidor.
- `and().build()`: Constrói o objeto `SecurityFilterChain` com as configurações definidas.

4 Testando as Alterações

Após salvar as alterações, o projeto é reiniciado automaticamente (via Spring DevTools). Testamos no Insomnia e no navegador para verificar o comportamento:

4.1 Testes no Insomnia

- **Requisição**: `GET http://localhost:8080/medicos/6999` (ID inexistente)
- **Resultado**: Código 404 (Not Found), tratado pela classe `TratadorDeErros`, indicando que a requisição não é mais bloqueada pelo `Spring Security`.
- **Requisição**: `GET http://localhost:8080/medicos`
- **Resultado**: Código 200 (OK) com a lista de médicos em JSON:

```
[
    {
        "id": 6,
        "nome": "Nome do Médico",
        "email": "medico1@voll.med",
        "crm": "233444",
        "especialidade": "ORTOPEDIA"
    },
    {
        "id": 7,
        "nome": "Nome do Médico",
        "email": "medico2@voll.med",
        "crm": "1234580",
        "especialidade": "ORTOPEDIA"
    }
]
```

- **Requisição**: `GET http://localhost:8080/medicos/7`
- **Resultado**: Código 200 (OK) com os detalhes do médico.

4.2 Testes no Navegador

- **URL**: `http://localhost:8080/medicos`
- **Resultado**: Exibe a lista de médicos em JSON, sem redirecionamento para o formulário de login, confirmando que o comportamento padrão do `Spring Security` foi desativado.

5 Impacto das Configurações

Com a classe `SecurityConfigurations`, desabilitamos o formulário de login e o bloqueio automático de requisições (anteriormente retornando 401 Unauthorized). A API agora opera em modo **stateless**, mas ainda não possui autenticação configurada, permitindo acesso irrestrito aos endpoints. Nas próximas aulas, implementaremos o endpoint de login e a geração de tokens JWT para proteger as requisições.

6 Próximos Passos

Na próxima aula, criaremos o endpoint de login, configuraremos a geração de tokens JWT e definiremos regras de autorização para proteger os endpoints da API. Continue praticando no `Visual Studio Code` ou sua IDE preferida!

7 Dica do Professor

- **Aprofunde-se em Spring Security**: Consulte a documentação do Spring Security sobre configuração stateless (`https://docs.spring.io/spring-security/reference/features/configuration/httpsecurity.html`) para entender opções avançadas.
- **Comunidade no X**: Siga perfis como @SpringSecMaster e @RestSecurity no X para dicas sobre configurações de segurança em APIs REST.
- **Pratique**: Teste outros endpoints (ex.: `POST /medicos`, `GET /pacientes`) no Insomnia para confirmar que estão liberados e planeje como protegê-los na próxima aula.