Aula 9: Iniciando com Spring Security

September 18, 2025

1 Introdução

Na Aula 8, apresentamos os conceitos de autenticação e autorização em APIs REST, destacando a importância do Spring Security para proteger nossa API. Nesta aula, adicionaremos o Spring Security ao projeto, incluindo suas dependências no pom.xml e observando seu comportamento padrão, que bloqueia todas as requisições e exige autenticação. Veremos também a diferença entre autenticação stateful (aplicações web) e stateless (APIs REST). As alterações serão feitas no Visual Studio Code ou sua IDE preferida, com testes no Insomnia e no navegador.

2 Adicionando o Spring Security

Para usar o Spring Security, adicionaremos suas dependências ao arquivo pom.xml. No site do Spring Initializr (https://start.spring.io), com as opções Maven Project, Java, e Spring Boot 3.0.0, selecionamos a dependência Spring Security. As dependências geradas são:

```
<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>
```

No Visual Studio Code ou sua IDE preferida, abrimos o pom.xml e inserimos essas dependências antes do fechamento da tag <dependencies>:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" ...>
    <dependencies>
        <!-- Outras dependências -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <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>
</project>
```

Após salvar, paramos a aplicação (atalho: `Ctrl + F12`) e recarregamos o Maven para baixar as dependências, clicando no ícone de recarga no painel do Maven. Reiniciamos o projeto para aplicar as alterações.

3 Comportamento Padrão do Spring Security

Ao iniciar o projeto, o console exibe uma senha gerada automaticamente, junto com a mensagem:

```
Using generated security password: 520dbc8a-b02a-44dq-a259-0afdb628a93c
This generated password is for development use only. Your security configuration must be
```

O `Spring Security` configura um usuário padrão (`user`) com essa senha para ambiente de desenvolvimento. Além disso, ele bloqueia todas as requisições da API, exigindo autenticação.

3.1 Testando no Insomnia

Testamos a requisição `GET http://localhost:8080/medicos` no Insomnia:

- **Resultado**: Código 401 (Unauthorized) com a mensagem "No body returned for response".

O mesmo ocorre com outras URLs, como `GET http://localhost:8080/medicos/6`. Todas as requisições são bloqueadas, pois o `Spring Security` exige autenticação.

3.2 Testando no Navegador

No navegador, ao acessar `http://localhost:8080/medicos`, somos redirecionados para um formulário de login do `Spring Security` com os campos `Username` e `Password`. Usando `user` como nome de usuário e a senha gerada (ex.: `520dbc8a-b02a-44dq-a259-0afdb628a93c`), a autenticação é bem-sucedida, e a lista de médicos é exibida:


```
[
    {
        "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"
    }
]
```


4 Limitations do Comportamento Padrão

O comportamento padrão do `Spring Security` (formulário de login e autenticação stateful) é adequado para aplicações web, mas nossa API REST deve ser **stateless**, sem sessões ou formulários no backend. O front-end ou aplicativo mobile será responsável pelo formulário de login, enquanto a API gerenciará autenticação via tokens JWT. Nas próximas aulas, configuraremos o `Spring Security` para implementar autenticação e autorização stateless.

5 Próximos Passos

Na próxima aula, configuraremos o `Spring Security` para suportar autenticação com JWT, criando uma tabela de usuários, um endpoint de login e lógica para geração de tokens. Continue praticando no `Visual Studio Code` ou sua IDE preferida!

6 Dica do Professor

- **Aprofunde-se em Spring Security**: Consulte a documentação oficial do Spring Security (`https://spring.io/projects/spring-security`) para entender configurações iniciais e autenticação stateless.
- **Comunidade no X**: Siga perfis como @SpringSecExpert e @RestAPIDev no X para dicas sobre segurança em APIs REST e configurações do Spring Security.
- **Pratique**: Teste outras requisições (ex.: `POST /medicos`) no Insomnia para confirmar o bloqueio do Spring Security e experimente o formulário de login no navegador com a senha gerada.