

Disciplina Regular 3

Desenvolvimento Web com Java EE

Graduação em Engenharia de Software - 2020

Etapa 6 Aula 2

Segurança de Aplicações Web

Para acompanhar pelo Moodle você
deve estudar a etapa 5

Etapa 6 Aula 2 - Competências

- **Competências Trabalhadas Nesta Etapa:**
 - **Estabelecer o gerenciamento de estado em aplicações Java Web:**
 - Compreender os conceitos sobre sessão e seu ciclo de vida.
 - Implementar Autenticação e Autorização declarativa (JEE).
 - **Implementar Autenticação e Autorização declarativa (Spring Security).**

Segurança Declarativa

Spring Security

Spring Security

- O Spring Security é uma estrutura de autenticação e controle de acesso poderosa e altamente personalizável. É o padrão de fato para proteger aplicativos baseados em Spring.
- O Spring Security é uma estrutura que se concentra no fornecimento de autenticação e autorização para aplicativos Java.
- Como todos os projetos do Spring, o poder real do Spring Security é encontrado na facilidade com que pode ser estendido para atender aos requisitos personalizados.

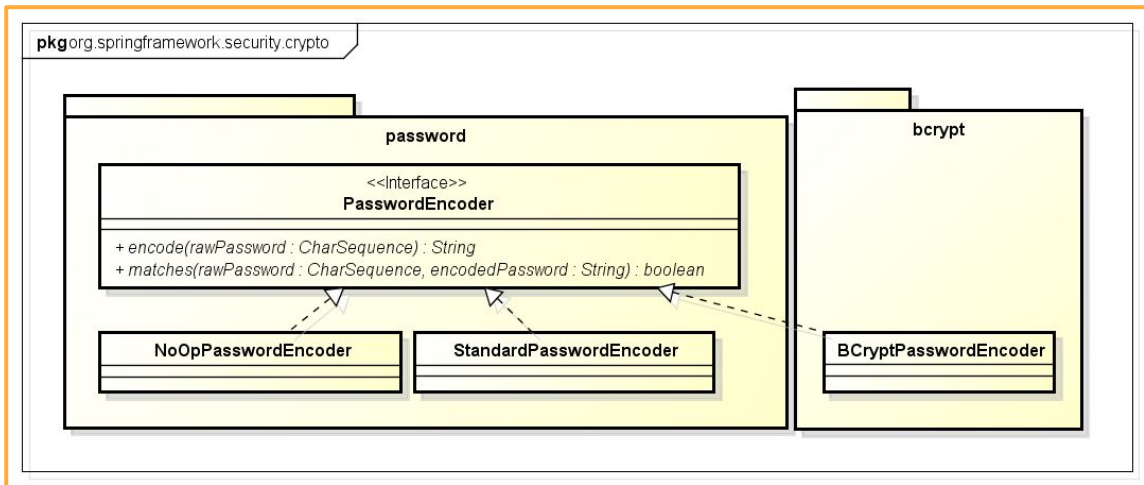
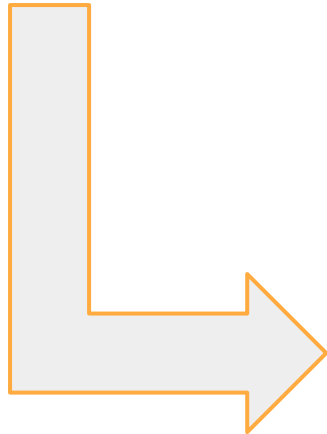
Spring Security

- Dependências do Maven:

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>5.3.2.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-web</artifactId>
  <version>5.3.2.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-core</artifactId>
  <version>5.3.2.RELEASE</version>
</dependency>
```

```
web.xml x
Source General Servlets Filters Pages References Security History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/X
3
4 <context-param
5     <param-name>contextConfigLocation</param-name>
6     <param-value>/WEB-INF/applicationContext.xml,/WEB-INF/applicationSecurity.xml</param-value>
7 </context-param>
8 <listener>
9     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
10 </listener>
11 <servlet>
12     <servlet-name>dispatcher</servlet-name>
13     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
14     <load-on-startup>2</load-on-startup>
15 </servlet>
16 <servlet-mapping>
17     <servlet-name>dispatcher</servlet-name>
18     <url-pattern>/</url-pattern>
19 </servlet-mapping>
20 <filter>
21     <filter-name>springSecurityFilterChain</filter-name>
22     <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
23 </filter>
24 <filter-mapping>
25     <filter-name>springSecurityFilterChain</filter-name>
26     <url-pattern>/*</url-pattern>
27 </filter-mapping>
28 <session-config>
29     <session-timeout>
30         30
31     </session-timeout>
32 </session-config>
```

```
applicationSecurity.xml x
Source History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <b:beans xmlns="http://www.springframework.org/schema/security"
3     xmlns:b="http://www.springframework.org/schema/beans"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://www.springframework.org/schema/beans
6         http://www.springframework.org/schema/beans/spring-beans.xsd
7         http://www.springframework.org/schema/security
8         http://www.springframework.org/schema/security/spring-security.xsd">
9
10    <b:bean
11        id="passwordEncoder"
12        class="org.springframework.security.crypto.password.NoOpPasswordEncoder"
13        factory-method="getInstance"
14    />
```



Configurações de HTTP

- **Cross-Site Request Forgery** (CSRF ou XSRF), em português falsificação de solicitação entre sites, também conhecido como ataque de um clique (one-click attack) ou montagem de sessão (session riding), **é um tipo de exploit malicioso de um website, no qual comandos não autorizados são transmitidos a partir de um usuário em quem a aplicação web confia.**
- Há muitos meios em que um site web malicioso pode transmitir tais comandos, tags de imagem especialmente criadas, formulários ocultos e XMLHttpRequests de JavaScript, por exemplo, podem funcionar sem a interação do usuário ou mesmo seu conhecimento.
- Diferente do cross-site scripting (XSS), que explora a confiança que um usuário tem para um site específico, o CSRF explora a confiança que um site tem no navegador de um usuário.

Configurações de HTTP

- CSRF:

```
<http auto-config='true'>  
  <csrf disabled="true"/>  
  
  <intercept-url pattern="/index.jsp" access="permitAll()"/>  
  <intercept-url pattern="/imagens/**" access="permitAll()"/>  
  <intercept-url pattern="/css/**" access="permitAll()"/>  
  <intercept-url pattern="/js/**" access="permitAll()"/>  
  
  <intercept-url pattern="/actors/**" access="hasAnyAuthority('administrador', 'usuario')"/>  
  <http-basic />  
</http>
```

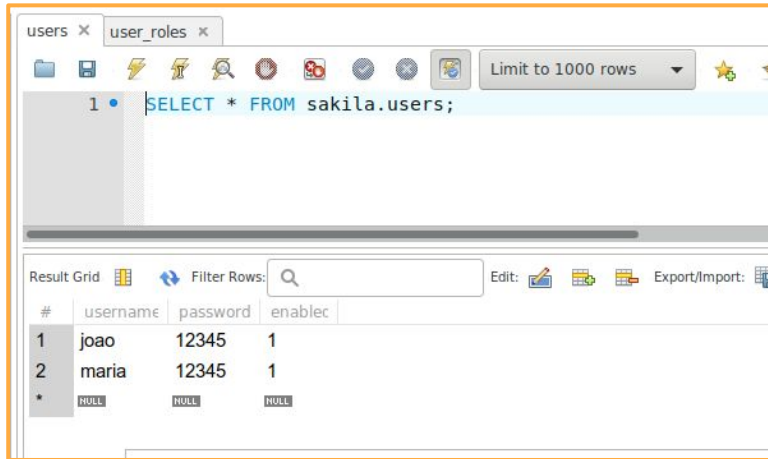
Configurações de HTTP

- Autorização:

```
<http auto-config='true'>
  <csrf disabled="true"/>
  <intercept-url pattern="/index.jsp" access="permitAll()" />
  <intercept-url pattern="/imagens/**" access="permitAll()" />
  <intercept-url pattern="/css/**" access="permitAll()" />
  <intercept-url pattern="/js/**" access="permitAll()" />
  <intercept-url pattern="/actors/**" access="hasAnyAuthority('administrador', 'usuario')" />
  <http-basic />
</http>
```

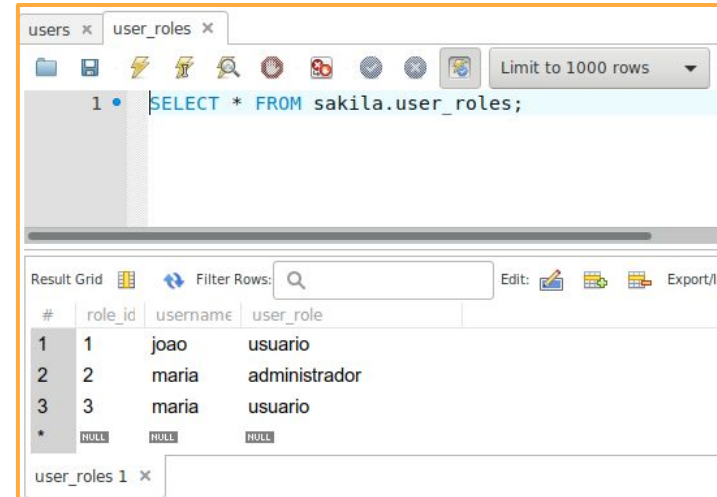
Configurações de HTTP

- Authentication Manager:



The screenshot shows a database client interface with two tabs: 'users' and 'user_roles'. The 'users' tab is active. The SQL editor contains the query `SELECT * FROM sakila.users;`. Below the editor, the 'Result Grid' displays the data from the 'users' table. The table has columns: #, username, password, and enablec. The data shows two rows: one for 'joao' and one for 'maria', both with password '12345' and 'enablec' value '1'. A third row is marked with asterisks and NULL values.

#	username	password	enablec
1	joao	12345	1
2	maria	12345	1
*	NULL	NULL	NULL



The screenshot shows the same database client interface, but with the 'user_roles' tab active. The SQL editor contains the query `SELECT * FROM sakila.user_roles;`. The 'Result Grid' displays the data from the 'user_roles' table. The table has columns: #, role_id, username, and user_role. The data shows three rows: one for 'joao' with role_id '1' and user_role 'usuario', one for 'maria' with role_id '2' and user_role 'administrador', and one for 'maria' with role_id '3' and user_role 'usuario'. A third row is marked with asterisks and NULL values.

#	role_id	username	user_role
1	1	joao	usuario
2	2	maria	administrador
3	3	maria	usuario
*	NULL	NULL	NULL

Configurações de HTTP

- Authentication Manager:

```
<authentication-manager>
  <authentication-provider>

    <password-encoder ref="passwordEncoder" />
    <jdbc-user-service data-source-ref="dataSource"
      users-by-username-query="select username,password, enabled from users where username=?"
      authorities-by-username-query="select username, user_role from user_roles where username=?" />
  </authentication-provider>
</authentication-manager>
```

Exercício

- Criar um projeto que contenha uma pasta “protegido” onde deve-se colocar um “index.jsp”.
- No “index.jsp” principal, criar um link para o recurso “/protegido/index.jsp”.
- Criar a configuração de autenticação e autorização que solicite o login para acessar o recurso “/protegido/index.jsp”.
 - Implementar a autenticação BASIC.
 - Depois alterar para FORM.