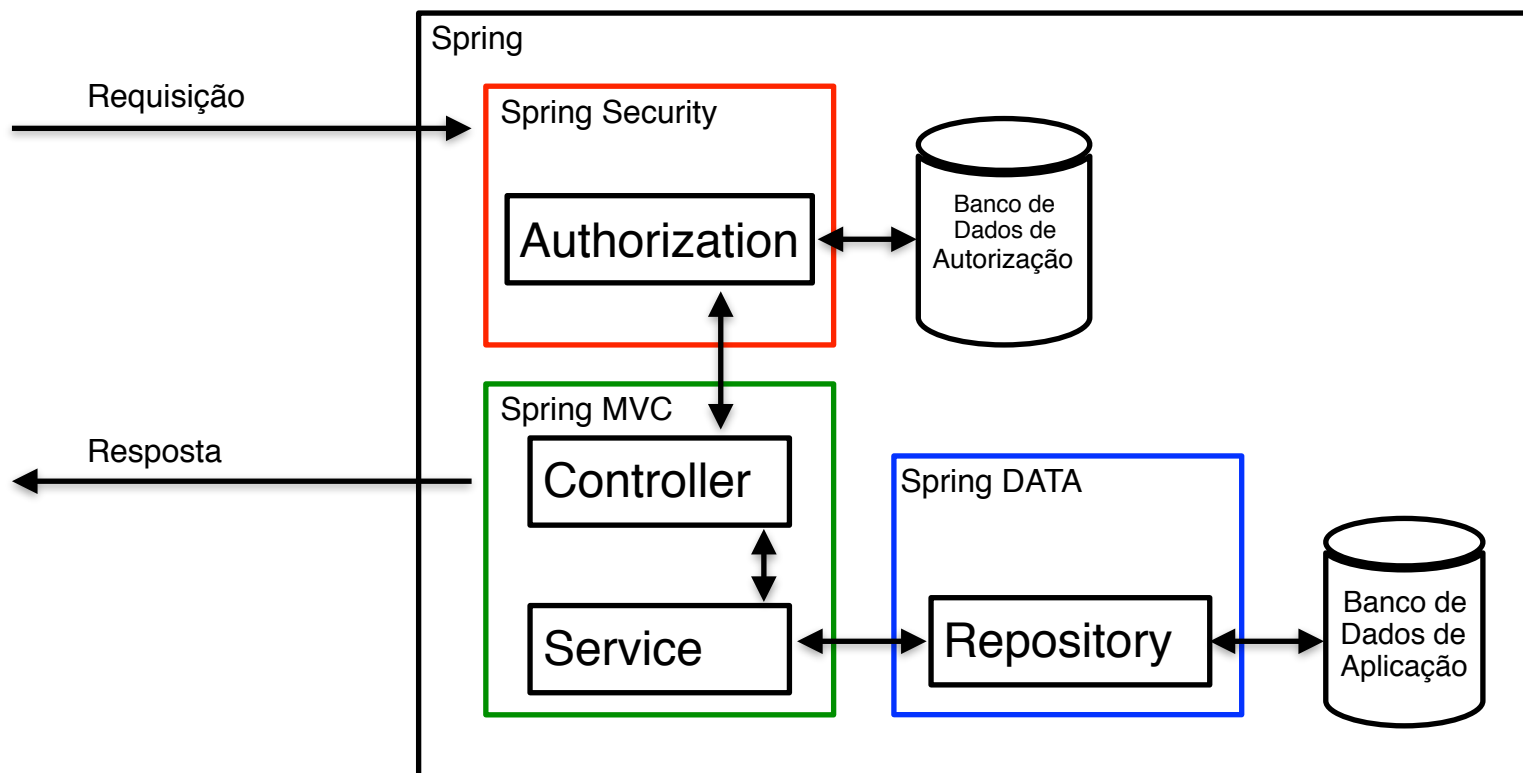




Spring Security


O **Spring Security** oferece uma grande gama de soluções de segurança para aplicações Java Web, tais como mecanismos de autenticação e autorização em vários tipos de serviços (LDAP, Kerberos, OAUTH2, etc.)



Habilitando a Segurança

build.gradle

```
dependencies {  
    compile('org.springframework.boot:spring-boot-starter-data-jpa')  
    compile('org.springframework.boot:spring-boot-starter-thymeleaf')  
    compile('org.springframework.boot:spring-boot-starter-security')  
    compile('org.thymeleaf.extras:thymeleaf-extras-springsecurity4')  
    compile('org.springframework.boot:spring-boot-starter-web')  
    runtime('mysql:mysql-connector-java')  
    compileOnly('org.projectlombok:lombok')  
    testCompile('org.springframework.boot:spring-boot-starter-test')  
    testCompile('org.springframework.security:spring-security-test')  
}
```

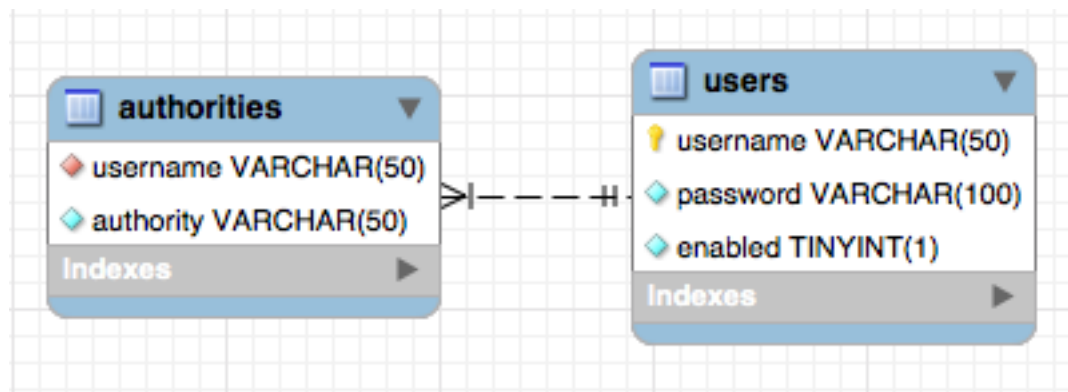


Base de dados de Autorização

Configuração do DataSource
para a base de dados de
autorização

```
@Configuration
public class AppConfig {
    @Autowired
    private Environment env;

    @Bean(name = "DataSource")
    public DriverManagerDataSource dataSource() {
        DriverManagerDataSource driverManagerDataSource = new DriverManagerDataSource();
        driverManagerDataSource.setDriverClassName(
            env.getProperty("spring.datasource.driver-class-name"));
        driverManagerDataSource.setUrl(
            env.getProperty("spring.datasource.url"));
        driverManagerDataSource.setUsername(
            env.getProperty("spring.datasource.username"));
        driverManagerDataSource.setPassword(
            env.getProperty("spring.datasource.password"));
        return driverManagerDataSource;
    }
}
```



Configurando os Acessos

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true,
    securedEnabled = true, jsr250Enabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private DataSource dataSource;
```

Habilitam as anotações de verificação de autorização

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
            .antMatchers("/", "/index.html", "/styles/**",
                "/imagens/**", "/bower_components/**").permitAll()
            .anyRequest().authenticated()
            .and()
        .formLogin()
            .loginPage("/login")
            .permitAll()
            .and()
        .logout()
            .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
            .logoutSuccessUrl("/login")
            .permitAll();
}
```

Descrevem as regras de acesso e autorização

Determina a URL da página de login

Determina a URL da página de Logout

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.jdbcAuthentication()
        .dataSource(dataSource)
        .usersByUsernameQuery("select username, password, enabled from users where username=?")
        .authoritiesByUsernameQuery("select username, authority from authorities where username=?")
        .passwordEncoder(new BCryptPasswordEncoder());
}
```

Determina as queries para acessar a base de autorização

```
<div th:if="${param.error}"
    class="alert alert-danger alert-dismissible fade show shadow-lg p-4 mb-4">
    <button type="button" class="close" data-dismiss="alert">&times;</button>
    <strong>Senha inválida</strong>
</div>

<div class="container p-2">
    <div class="d-flex justify-content-center quadro">
        <div>
            <form th:action="@{/login}" method="post">
                <div class="form-group">
                    <label>Usuário</label>
                    <input class="form-control" type="text" name="username"/>
                </div>
                <div class="form-group">
                    <label>Senha</label>
                    <input class="form-control" type="password" name="password"/>
                </div>
                <div>
                    <input class="btn btn-primary shadow border border-light"
                        type="submit" value="Enviar"/>
                </div>
            </form>
        </div>
    </div>
</div>
</div>
```

Formulário de Login

Apresenta mensagem de
falha de autenticação

Construindo um Componente

```
public interface SecurityCheck {  
    boolean isInRole(String role);  
    String getUsername();  
}
```

Interface publica para o
componente de verificação de
autorização

```
@Component  
public class SecurityFacade implements SecurityCheck {  
    @Override  
    public boolean isInRole(String role) {  
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
        if (auth != null) {  
            UserDetails principal = (UserDetails)auth.getPrincipal();  
            return principal.getAuthorities().stream()  
                .filter(grant -> grant.getAuthority().equals(role))  
                .findAny()  
                .isPresent();  
        }  
        return false;  
    }  
  
    @Override  
    public String getUsername() {  
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
        if (auth != null) {  
            UserDetails principal = (UserDetails)auth.getPrincipal();  
            return principal.getUsername();  
        } else {  
            return null;  
        }  
    }  
}
```

Implementação da **interface**
no **Componente** que verifica a
autorização

Autorizando o Acesso

```
@Controller
@SessionAttributes({ "cadastro", "listaservicos" })
public class CadastroController {
    @Autowired
    private CadastroService clienteDao;
    @Autowired
    private ServicoService servicoDao;
    @Autowired
    private SecurityCheck security;

    @RequestMapping("/cadastra")
    public ModelAndView cadastra(@ModelAttribute("cliente") @Valid Cliente cliente, BindingResult result,
        RedirectAttributes redirectAttributes) {

        if(!security.isInRole("ADMIN")) {
            ModelAndView model = new ModelAndView("editaCadastro");
            model.addObject("mensagem", "O Acesso foi Negado");
            model.addObject("tipo.mensagem", "alert-danger");
            return model;
        } else if (result.hasErrors()) {
            return new ModelAndView("editaCadastro", "cliente", cliente);
        } else {
            Integer id = cliente.getIdCliente();
            clienteDao.salvar(cliente);

            if (id == null) {
                redirectAttributes.addFlashAttribute("mensagem", "Cliente cadastrado");
                redirectAttributes.addFlashAttribute("tipo.mensagem", "alert-success");
                return new ModelAndView("redirect:/editaCadastro");
            } else {
                redirectAttributes.addFlashAttribute("mensagem", "Cliente atualizado");
                redirectAttributes.addFlashAttribute("tipo.mensagem", "alert-success");
                return new ModelAndView("redirect:/listaCadastro");
            }
        }
    }
}
```

Faz referência ao **Componente** de verificação de autorização

Verificação se o usuário logado tem autorização

@Secured - Utilizada para especificar uma lista de acessos permitidos.

@RoleAllowed - Esta anotação é da especificação **JSR-250** e é semelhante a **@Secured**.

@PreAuthorize - Fornece o controle de acesso antes do acesso ao método e é baseado em expressões Spring.

@PostAuthorize - Fornece o controle de acesso após o acesso ao método e é baseado em expressões Spring.

Acesso as autorizações em HTML

Verificando a permissão
do usuário autenticado

```
<li sec:authorize="hasAuthority('ADMIN')" class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown">Usuários</a>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="editaUsuario">Cadastra Usuários</a>
    <a class="dropdown-item" href="listaUsuario">Lista Usuários</a>
  </div>
</li>
```

Verificando se o usuário
está autenticado

```
<div th:unless="${#authorization.expression('isAuthenticated()')}">
  <a class="text-white" th:href="@{/login}">
    login
    <span class="fa fa-sign-in-alt p-2"></span>
  </a>
</div>
<div th:if="${#httpServletRequest.remoteUser}">
  <a class="text-white" th:href="@{/logout}">
    logout<span th:text="${#httpServletRequest.remoteUser}" class="p-2"></span>
    <span class="fa fa-sign-out-alt p-2"></span>
  </a>
</div>
```

Obtendo o nome do
usuário autenticado

<https://docs.spring.io/spring-security/site/docs/5.0.7.RELEASE/reference/htmlsingle>

Spring Security Reference

Authors

Ben Alex , Luke Taylor , Rob Winch , Gunnar Hillert , Joe Grandja , Jay Bryant

5.0.7.RELEASE

Copyright © 2004-2017

Copies of this document may be made for your own use and for distribution to others, provided that you do not charge any fee for such copies and further provided that each copy contains this Copyright Notice, whether distributed in print or electronically.

Table of Contents

I. Preface

1. Getting Started