

CETEJ35 - Java Web - JAVA_XXX (2024_01)

[Meus cursos](#) / [CETEJ35 - Web \(2024_01\)](#) / [Semana 10: 04/11 a 10/11](#) / [Listeners & Cookies](#)

Listeners & Cookies

✓ **Feito:** Ver

A fazer: Gastar pelo menos 20 minutos na atividade

A fazer: Passar pela atividade até o fim

Fecha: segunda-feira, 2 dez. 2024, 00:00

Nesta aula vamos falar sobre dois tópicos complementares no contexto do aplicativo que estamos usando como exemplo, mas importantes no desenvolvimento de aplicações Web.

COOKIES

Cookies representam informações que são armazenadas como um par de valores do tipo texto no navegador do usuário. Enquanto um valor representa o nome do cookie, o outro valor representa o conteúdo armazenado no cookie.

Cookies são muito comuns em páginas Web. Com certeza você já acessou algum website que te informa que seus dados serão armazenados para 'melhorar sua experiência no site'. Normalmente, esses dados são salvos em cookies.



Diferente da sessão, um cookie armazena os dados no navegador do usuário em vez de um espaço na memória do servidor. Isso é importante porque a memória do servidor é encerrada no logout, enquanto o cookie pode permanecer no navegador. Outra diferença é que o cookie armazena um par de **Strings**, enquanto uma sessão pode armazenar um objeto.

É claro que o usuário pode sempre se livrar dos cookies limpando o cache do navegador, ou com a expiração dos cookies.

Dessa forma, sempre que você acessar uma determinada página, seu aplicativo pode verificar pela existência de cookies criados previamente. Se eles existirem, você pode aproveitar os dados para melhorar a experiência do usuário e indicar produtos relacionados com pesquisas prévias, por exemplo.

O gerenciamento de cookies no Spring Boot é muito similar ao da sessão. O cookie também é injetado com injeção de dependência. Dessa forma, tudo que precisamos fazer é acessar o objeto e adicionar valores.

Para mostra o uso de cookies, vamos gravar em um cookie o momento em que cada operação do CRUD foi acessada pela última vez. Para isso, vamos precisar alterar os métodos que representam o CRUD na classe **cidade.CidadeController**.

As alterações são sempre as mesmas: (i) adicionar um parâmetro do tipo **javax.servlet.http.HttpServletResponse**, e (ii) usar esse objeto para adicionar um cookie com os valores definidos.

O objeto **javax.servlet.http.HttpServletResponse** representa o resultado de uma solicitação que é enviada como retorno para o navegador. É esse objeto tem o método **addCookie()**. Esse método recebe como parâmetro um objeto do tipo **javax.servlet.http.Cookie**, representando um par de valores **String**, sendo que o primeiro é nome do cookie e o segundo é o valor que o cookie representa.

```
28     @GetMapping("/")
29     public String listar(
30         Model memoria,
31         Principal usuario,
32         HttpSession sessao,
33         HttpServletResponse response) {
34
35         response.addCookie(new Cookie("listar", LocalDateTime.now().toString()));
```

Na Figura acima podemos ver como o método `listar()` ficou após a modificação. A linha 33 mostra a definição da variável `response`, como parâmetro do tipo `javax.servlet.http.HttpServletResponse`. Já a linha 35 mostra como o método `addCookie()` é usado para adicionar um novo cookie.

Nesse exemplo, o nome do cookie é `listar`, e o valor armazenado no cookie é extraído do objeto `java.time.LocalDateTime`, representando o dia e hora atuais.

Veja como ficou a replicação dessa ação no método `criar()`. Nesse trecho de código, as mudanças estão nas linhas 53 e 55.

```
48     @PostMapping("/criar")
49     public String criar(
50         @Valid Cidade cidade,
51         BindingResult validacao,
52         Model memoria,
53         HttpServletResponse response) {
54
55         response.addCookie(new Cookie("criar", LocalDateTime.now().toString()));
```

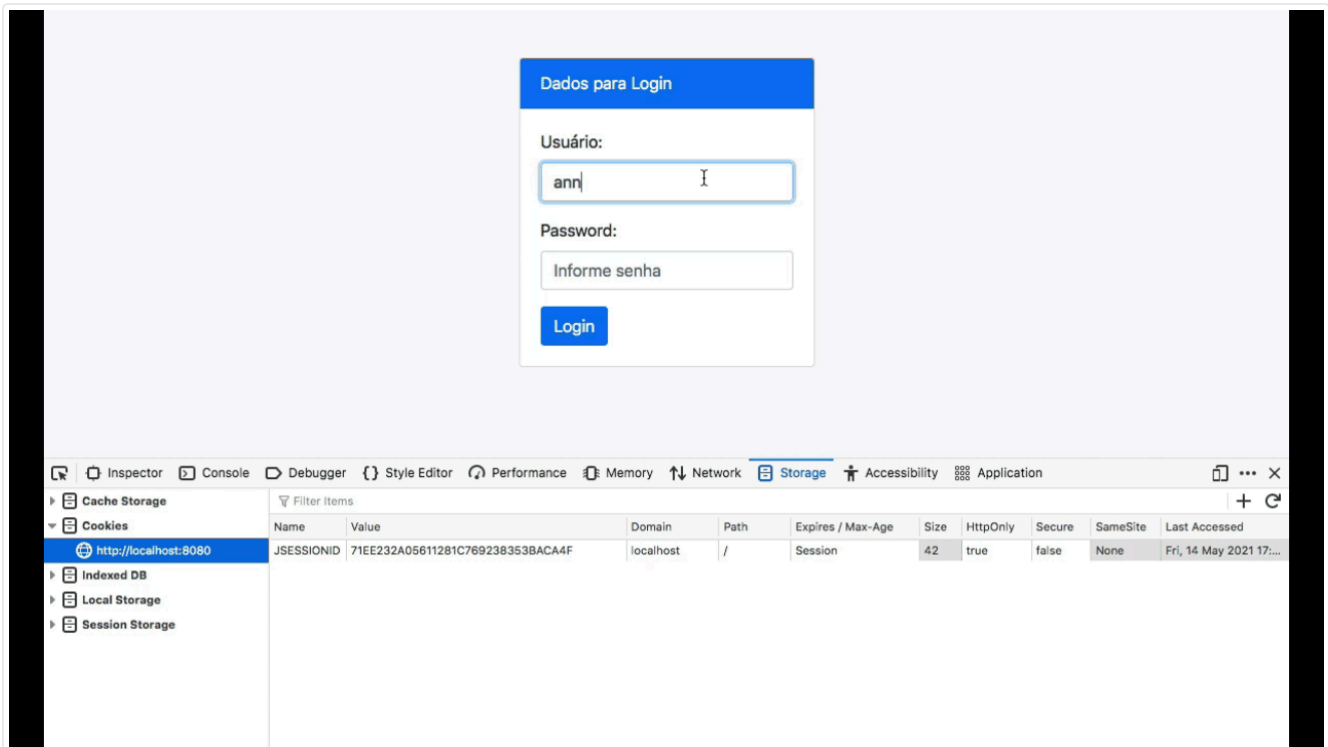
Veja como ficou a replicação dessa ação no método `excluir()`. Nesse trecho de código, as mudanças estão nas linhas 82 e 84.

```
78     @GetMapping("/excluir")
79     public String excluir(
80         @RequestParam String nome,
81         @RequestParam String estado,
82         HttpServletResponse response) {
83
84         response.addCookie(new Cookie("excluir", LocalDateTime.now().toString()));
```

Veja como ficou a replicação dessa ação no método `alterar()`. Nesse trecho de código, as mudanças estão nas linhas 114 e 116.

```
109    @PostMapping("/alterar")
110    public String alterar(
111        @RequestParam String nomeAtual,
112        @RequestParam String estadoAtual,
113        Cidade cidade,
114        HttpServletResponse response) {
115
116        response.addCookie(new Cookie("alterar", LocalDateTime.now().toString()));
```

Feito isso, é possível acessar o aplicativo e verificar, diretamente do navegador, os cookies que são adicionados à medida que os métodos são acessados.



Para mostrar como ler um cookie, vamos criar um novo método nessa classe chamado `mostrar()`, que retorna uma `String`, representando o resultado que deve ser apresentado na página Web. Esse método recebe uma `String` como parâmetro, representando o valor armazenado no cookie. O nome da variável de acesso ao parâmetro deve ser o mesmo usado para nomear o cookie. Nesse exemplo, vamos usar `listar`, que é o nome do cookie criado para guardar o horário do último acesso ao método `listar()`.

Vamos mapear esse método com a URL `/mostrar`. Para isso, basta adicionar `@GetMapping("/mostrar")` imediatamente antes da definição do método. Como esse método retorna uma `String` em vez de uma página Web, precisamos adicionar a anotação `org.springframework.web.bind.annotation.ResponseBody`, imediatamente antes da definição do método. Veja como ficou o resultado.

```

134     @GetMapping("/mostrar")
135     @ResponseBody
136     public String mostraCookieAlterar(@CookieValue String listar) {
137         return "Último acesso ao método listar(): " + listar;
138     }

```

A anotação `org.springframework.web.bind.annotation.CookieValue`, adicionada imediatamente antes da definição do parâmetro do método, é usada pelo Spring Boot para mapear o cookie cujo nome corresponde ao nome do parâmetro (nesse caso, `listar`). Por fim, o corpo do método simplesmente retorna o valor em `listar` concatenado com um texto indicando o que esse valor representa.

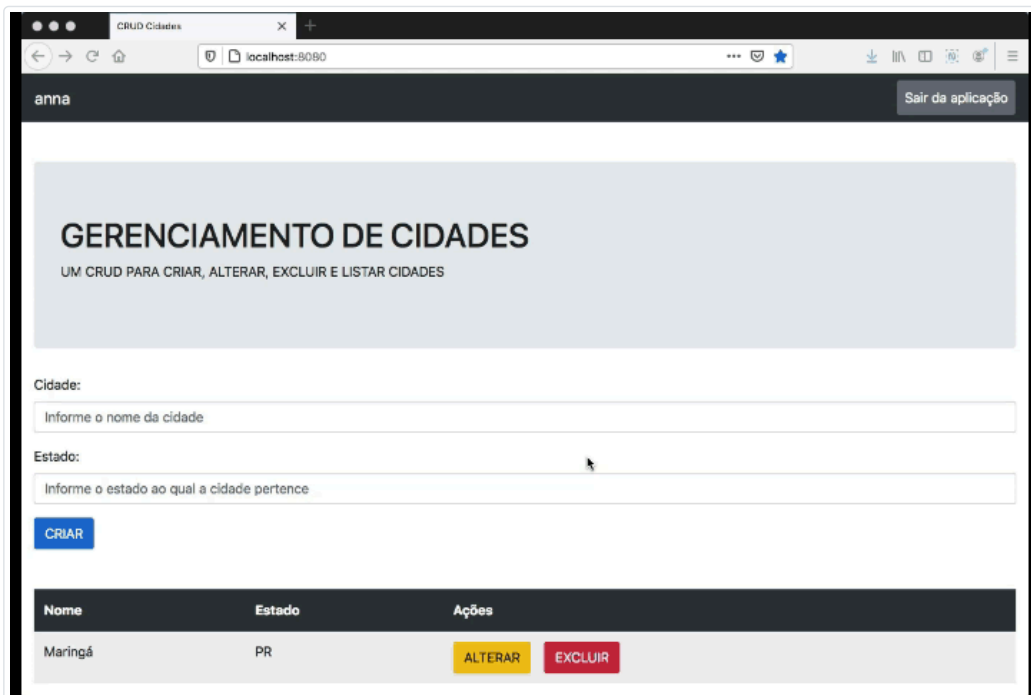
Como nossa aplicação está usando autenticação e autorização, precisamos adicionar uma regra permitindo acesso à URL `/mostrar`. Para isso, precisamos adicionar uma regra de permissão na classe `crudidades.SecurityConfig`.

```
SecurityConfig.java

13 @EnableWebSecurity
14 @Configuration
15 public class SecurityConfig {
16
17     @Bean
18     protected SecurityFilterChain filter(HttpSecurity http) throws Exception {
19         return http
20             .csrf().disable()
21             .authorizeHttpRequests()
22             .requestMatchers("/").hasAnyRole("listar", "admin")
23             .requestMatchers("/criar", "/excluir", "/preparaAlterar", "/alterar").hasRole("admin")
24             .requestMatchers("/mostrar").authenticated()
25             .anyRequest().denyAll()
26             .and()
27             .formLogin()
28             .loginPage("/login.html").permitAll()
29             .and()
30             .logout().permitAll()
31             .and()
32             .build();
33     }
34 }
```

Snipped

Adicionamos a linha 24. Nessa linha, além de definir a nova URL, também adicionamos uma permissão dizendo que qualquer usuário autenticado tem acesso à URL. Agora é só executar a aplicação, acessar o a página principal e depois acessar a nova URL.



O código desenvolvido nesta Seção está disponível no [Github](#), na branch `semana08-30-misc-cookies`.


Retroceder

Avançar

[◀ Verificação de aprendizado - Segurança](#)

Seguir para...

Avaliação Oficial ►


✉ Contate o suporte do site 

Você acessou como RAFAEL ROCHA DA SILVA PROENCA (Sair)
CETEJ35 - Web (2024_01)

- Tema
 - Adaptable
 - Boost
 - Clássico
- Campus
 - Apucarana
 - Campo Mourão
 - Cornélio Procópio
 - Curitiba
 - Dois Vizinhos
 - Francisco Beltrão
 - Guarapuava
 - Londrina
 - Medianeira
 - Pato Branco
 - Ponta Grossa
 - Reitoria
 - Santa Helena
 - Toledo
- UTFPR
 - Ajuda
 - Chat UTFPR
 - Calendário Acadêmico
 - Biblioteca
 - e-Mail
 - Nuvem (OwnCloud)
 - Produção Acadêmica
 - Secretaria Acadêmica
 - Sistemas Corporativos
 - Sistema Eletrônico de Informação - SEI
 - Suporte ao usuário
 - Criação de curso
 - Comunidade
 - Português - Brasil (pt_br)
 - Deutsch (de)
 - English (en)
 - Português - Brasil (pt_br)

Resumo de retenção de dados

Baixar o aplicativo móvel.

🔊 Dê um feedback sobre este software 

Universidade Tecnológica Federal do Paraná - UTFPR
Suporte ao usuário

