Seção 05 - Resumo

Perfil de Médico

Ao trabalharmos com o perfil de médico vamos começar a utilizar um recurso do Spring Security que nos proporcionará ter acesso a algumas informações sobre as credenciais do usuário logado. Em alguns momentos será importante ter acesso a essas informações que ficam armazenadas no sessão do Spring.

Tais informações são o nome de usuário (username) e a lista de perfis, entre algumas mais que não são interessante para nós no momento. Existem pelo menos três formas de acesso que podem ser usadas para obter tais informações e elas são declaradas nas assinaturas dos métodos dos controllers. Uma opção bastante utilizada pela grande maioria dos desenvolvedores é a anotação `@AuthenticationPrincipal` acompanhada de uma variável do tipo `User`. Veja a seguir um breve exemplo:

```
@GetMapping("/")
public String exemplo(@AuthenticationPrincipal User user) {
   Medico medico = service.buscarPorEmail(user.getUsername());
   return new ModelAndView("medico/cadastro", "medico", medico);
}
```

Veja que podemos recuperar o e-mail (username) do usuário logado via instrução `user.getUsername()` e usá-lo para localizar os dados do médico logado.

Além da anotação `@AuthenticationPrincipal` ainda poderíamos trabalhar com a interface `Authentication` do Spring Security, da seguinte forma:

```
@GetMapping("/")
public String exemplo(Authentication authentication) {
   Medico medico = service.buscarPorEmail(authentication.getName());
   return new ModelAndView("medico/cadastro", "medico", medico);
}
```

Observe que dessa vez o método que nos retorna o username do usuário logado é o `getName()` e não `getUsername()` como na classe `User`.

Por fim, podemos também usar a interface `java.security.Principal`, ela não é uma interface nativa do Spring mas sim da especificação de segurança do Java, a qual o Spring pode injetar sem qualquer problema. Inclusive a interface `Authentication`, anteriormente citada, estende `Principal`.

```
@GetMapping("/")
public String exemplo(Principal principal) {
   Medico medico = service.buscarPorEmail(principal.getName());
   return new ModelAndView("medico/cadastro", "medico", medico);
}
```

No caso de sua escolha for pela `Principal` o método para obter o acesso ao username também é o `getName()`.

Você pode estar se perguntando qual a melhor opção, na verdade não há nada que indique qual seria a melhor opção. O Spring começou com a `Principal` e a `Authentication` e com o surgimento do uso de anotações acabou também criando o recurso pela anotação. Sendo assim, fica a seu critério decidir entre qual usar em seus projetos.

Formulário de edição de senha

O código javascript apresentado em aula para o formulário de edição de senha apresentou uma falha, identificada por um dos alunos. Ao digitar a nova senha e a confirmação da nova senha, o terceiro campo de senha é habilitado. Ao excluir ou digitar errado uma das duas senhas o terceiro campo volta a ficar desabilitado. Entretanto, se apagar as duas senhas o terceiro campo permanece habilitado.

Assim sendo, o código abaixo corrige o problema:

```
$('.pass').keyup(function(){
    if($('#senha1').val() == "" || $('#senha1').val() == "") {
        $('#senha3').attr('readonly', 'readonly');
    } else {
        $('#senha1').val() === $('#senha2').val()
            ? $('#senha3').removeAttr('readonly')
            : $('#senha3').attr('readonly', 'readonly');
    }
});
```

Agradeço ao Francisco Araújo pelo código e pelo aviso sobre a correção.

Referencias

- Java Doc de java.security.Principal
- @AuthenticationPrincipal
- Java Doc Authentication

Código Fonte

Caso tenha tido algum tipo de dificuldade para acompanhar a desenvolvimento do código fonte até o final desta seção, ele está disponível na área de arquivo para download.