02

Mãos à obra: Protegendo a aplicação com UsuarioDTO

Nossa aplicação está vulnerável ao ataque conhecido como **Mass Assignment**, onde a usuária Joviane havia conseguido se cadastrar como administrador da Alura Shows ao alterar o atributo **roles** para ROLE_ADMIN. Com isso, precisamos proteger nossa aplicação, a primeira forma que iremos utilizar é um padrão de projeto chamado de DTO (*Data Transfer Object*), onde criaremos nossa classe modelo chamado de UsuarioDTO exatamente com os atributos que esperamos ser recebidos pelo formulário, ou seja, sem o atributo **roles**. Vá para o pacote modelo e crie a classe **UsuarioDTO** com os quatro atributos (email, senha, nome, nomeImagem).

```
public class UsuarioDTO {

    private String email;
    private String senha;
    private String nome;
    private String nomeImagem;
}
```

Essa classe será utilizada para transferir os valores vindos do formulário para a classe Usuario, criando assim o nosso objeto Usuario, não precisaremos dos getters, somente dos setters. Para isso, clique no teclado **CTRL + 3** e digite **ggas** (Generate getters and setters) e clique na opção **Select Setters**:

```
public class UsuarioDTO {

    private String email;
    private String senha;
    private String nome;
    private String nomeImagem;

    public void setEmail(String email) {
        this.email = email;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public void setNomeImagem(String nomeImagem) {
        this.nomeImagem = nomeImagem;
    }
}
```

Posteriormente, crie o método **montaUsuario**, onde devemos instanciar a classe Usuario e passamos esses dados no construtor da classe.

```
public Usuario montaUsuario(){
    Usuario usuario = new Usuario(email,senha,nome,nomeImagem);
    return usuario;
}
```

Ao fazermos isso, será necessário ir na classe Usuário e criar esse construtor:

```
...
public Usuario(String email, String senha, String nome, String nomeImagem){
    this.email=email;
    this.senha=senha;
    this.nome=nome;
    this.nomeImagem=nomeImagem;
}

....
```

Feito isso, volte a classe **UsuarioController** e no método registrar faça a associação com a classe **UsuarioDTO**:

```
@RequestMapping(value = "/registrar", method = RequestMethod.POST)
    public String registrar(MultipartFile imagem,
            @ModelAttribute("usuarioRegistro") UsuarioDTO usuarioDTO,
....
```

Feito isso, precisaremos chamar o método **montaUsuario** presente na classe **UsuarioDTO** para termos nosso objeto do tipo **Usuario**, vamos declarar a variável como usuarioRegistro para que a aplicação continue funcionando:

```
@RequestMapping(value = "/registrar", method = RequestMethod.POST)
    public String registrar(MultipartFile imagem,
            @ModelAttribute("usuarioRegistro") UsuarioDTO usuarioDTO,
            RedirectAttributes redirect, HttpServletRequest request,
            Model model, HttpSession session) throws IllegalStateException, IOException {

        Usuario usuarioRegistro = usuarioDTO.montaUsuario();

....
```

Agora volte para aplicação e tente realizar o cadastro de um novo usuário, por exemplo o Pedro, fazendo a mesma manipulação HTML para alterar o atributo roles que a Joviane tinha realizada na etapa anterior.

```
<input type="hidden" name="roles[0].name" value="ROLE_ADMIN"/>
```

Esse usuário foi cadastrado com a ROLE_ADMIN? Foi possível fazer login na parte administrativa?