01

Salvando senha com código Hash

Transcrição

Alteramos a classe `UsuarioDaoImpl` para trabalhar com o `EntityManager`. Em seguida alteramos a *query*, onde primeiro criamos e levamos para o banco de dados, depois pegamos os parâmetros do formulário de autenticação do usuário. Conseguimos proteger a aplicação contra ataques de injeção de código SQL. Mas como os usuários são armazenados no banco de dados?

Olhando os usuário cadastrados na tabela `usuario`, veremos que a senha está sendo armazenada em texto puro. Qualquer pessoa que tenha acesso ao banco saberá as senha, o que não é bom para usuários. Precisamos encontrar uma forma de proteger as senhas com um código difícil de ser decifrado, esse código é o ***Hash***. Para podermos trabalhar com *Hash*, usaremos uma classe chamada `BCrypt`.

Antes de iniciarmos as modificações, pararemos o Tomcat. Queremos fazer a alteração da senha para o código *Hash* antes de objeto que representa o usuário ser persistido. Na classe `UsuarioDaoImpl` dentro do método `salva(Usuario usuario)`, chamaremos o método estático `BCrypt.gensalt()` para criar o **salto** que é um valor aleatório que será adicionado a senha. Em seguida chamaremos o `BCrypt.hashpw(usuario.getSenha(), salto)`, onde passamos a senha que será transformada em *Hash* e o salto que será adicionado. Agora resta associar a o retorno que é a senha com o *Hash* no objeto do `usuario`.

```
public void salva(Usuario usuario){
    String salto = BCrypt.gensalt();
    String senhaHashed = BCrypt.hashpw(usuario.getSenha(), salto);
    usuario.setSenha(senhaHashed);
    manager.persist(usuario);
}
```

Porém essa lógica de transformar a senha em *Hash* e associar ao `usuario` não é função do método `salva(Usuario usuario)`, por isso selecionaremos todo código e clicaremos com o botão direito do mouse, no menu vamos em "Refactor > Extract Method". Na janela ***Extract Method*** colocaremos o nome do método de `transformaASenhaDoUsuarioEmHash`. Os métodos ficarão da seguinte forma:

```
public void salva(Usuario usuario){
    transformaASenhaDoUsuarioEmHash(usuario);
    manager.persist(usuario);
}

private void transformaASenhaDoUsuarioEmHash(Usuario usuario){
    String salto = BCrypt.gensalt();
    String senhaHashed = BCrypt.hashpw(usuario.getSenha(), salto);
    usuario.setSenha(senhaHashed);
}
```

Cadastraremos um novo usuário para testar a senha em *Hash*. Iniciaremos o Tomcat e acessaremos a aplicação, em seguida acessaremos "Usuário > Registrar". Colocaremos as seguintes informações par o cadastro:

- **Nome:** `Ana`

- **E-mail** `ana@gmail.com`
- **Senha:** `789`
- **Imagem perfil:** `ana.jpg`

Conseguimos fazer o registro, mas como ficou salvo no banco de dados? Acessando as informações dos usuários cadastrados na tabela `usuario`, veremos que a **Ana** foi cadastrada com a senha em *Hash*. Qualquer pessoa que acessar o banco de dados, não saberá dizer qual senha ela usou para se cadastrar, deixando a aplicação bem mais segura.

E na hora de se autenticar? Faremos o *logout* na conta da Ana e acessaremos o formulário de ***Login***. No campo **E-mail** colocaremos `ana@gmail.com e no campo **Senha** colocaremos 789`, clicaremos em "LOG IN". Receberemos a mensagem de "Usuário não encontrado".

Isso aconteceu porque a *query* que estamos enviando ao banco de dados está procurando por um usuário com o e-mail `ana@gmail.com e senha 789`. Como a Ana está salva no banco com a senha utilizando o código Hash, recebemos a mensagem de "Usuário não encontrado".

Veremos como resolver esse problema de autenticação.