05

Mãos à obra: Implementando a autenticação

Vamos botar pra rodar nossa página de login? :)

1) O primeiro passo seria criar a página pra isso, afinal o usuário precisará informar suas credenciais pra nossa aplicação. Vamos criar uma página (um novo arquivo `.xhtml`) chamado `login.xhtml` dentro da pasta `WebContent`. Nessa página adicione um formulário com dois `inputs` (para login e para senha):

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

    <ui:composition template="_template.xhtml">

        <ui:define name="titulo">
                Login
        </ui:define>

        <ui:define name="conteudo">
            <h:form id="login">
                <fieldset>
                    <legend>Login</legend>
                    <h:panelGrid columns="3">

                        <h:outputLabel value="Email:" for="email" />
                        <h:inputText id="email" value="#{loginBean.usuario.email}"
                            required="true">
                            <f:passThroughAttribute name="type" value="email" />
                        </h:inputText>

                        <h:message for="email" id="messageEmail" />

                        <h:outputLabel value="Senha:" for="senha" />
                        <h:inputText id="senha" value="#{loginBean.usuario.senha}"
                            required="true">
                            <f:passThroughAttribute name="type" value="password" />
                        </h:inputText>

                        <h:message for="senha" id="messageSenha" />

                        <h:commandButton value="Efetuar Login"
                            action="#{loginBean.efetuaLogin}" />
                    </h:panelGrid>
                </fieldset>
            </h:form>
        </ui:define>
    </ui:composition>
</html>
```

2) Tenho certeza que você se perguntou: Quem é esse tal de `LoginBean` e `Usuario`. Ainda não temos :(Vamos criá-los?

Crie então um novo arquivo chamado de `Usuario` dentro do pacote `br.com.caelum.livraria.modelo`:

```
@Entity
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    private Integer id;
    private String email;
    private String senha;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
        this.senha = senha;
    }

}
```

Sabemos que precisamos mapear nossas entidades no `persistence.xml`, não é mesmo? Adivinha então o que faremos com o usuário!

Adicione no `persistence.xml`:

```
<class>br.com.caelum.livraria.modelo.Usuario</class>
```

Precisamos ainda criar o `LoginBean` no pacote `br.com.caelum.livraria.bean`:

```
@ManagedBean
@ViewScoped
public class LoginBean {
```

```
    private Usuario usuario = new Usuario();

    public Usuario getUsuario() {
        return usuario;
    }
}
```

3) Se você olhou com carinho o código do `login.xhtml`, percebeu que o `commandButton` faz um **binding** com um método chamado `efetuaLogin()` dentro de `LoginBean`. Vamos recordar?

```
<h:commandButton value="Efetuar Login" action="#{loginBean.efetuaLogin}" />
```

Ótimo! Agora vamos criar esse método, então dentro de `LoginBean`:

```
public String efetuaLogin() {
    System.out.println("Fazendo login do usuário "
            + this.usuario.getEmail());

    boolean existe = new UsuarioDao().existe(this.usuario);

    if (existe) {
        return "livro?faces-redirect=true";
    }
    return null;
}
```

4) E esse `UsuarioDao`? Vamos criá-lo também dentro do pacote `br.com.caelum.livraria.dao`!

```
public class UsuarioDao {

    public boolean existe(Usuario usuario) {

        EntityManager em = new JPAUtil().getEntityManager();
        TypedQuery<Usuario> query = em
            .createQuery("select u from Usuario u where u.email = :pEmail and u.senha = :pSenha'

        query.setParameter("pEmail", usuario.getEmail());
        query.setParameter("pSenha", usuario.getSenha());

        try {
            Usuario resultado = query.getSingleResult();
        } catch (NoResultException ex) {
            return false;
        }

        em.close();

        return true;
    }
}
```

5) Quase pronto! Antes de testarmos o login, precisamos de um usuário no banco de dados. Portanto, vamos adicionar o usuário no terminal do MySQL.

Obs: Você deve rodar a aplicação uma vez e chamar, por exemplo, a página `livro.xhtml` . Assim o JPA a criará a tabela `Usuario` no banco de dados.

```
mysql> use livrariadb;
mysql> insert into Usuario(email, senha) values ('nico.steppat@caelum.com.br', '12345');
```

6) Agora vamos testar o login. Acesse a página `login.xhtml` pelo navegador e tente logar com o seus usuário cadastrado. Se tudo tiver certinho você será redirecionado ao `login.xhtml` !