EJB + JPA – Enterprise JavaBeans + Java Persistence API

Tiago Alves de Oliveira

tiagofga@gmail.com

EJB + JPA

- Hoje vamos criar um EJB juntamente com JPA.
- Criaremos um exemplo de login e cadastro de Estudante.
- Utilizaremos EJB no projeto, juntamente com persistência do Banco.

EJB + JPA

- Para começar vamos criar o banco de dados.
- Crie um schema chamado **aula07**.
- Crie a tabela **login** como abaixo:

Campo	Tipo
userId	int
userName	varchar(45)
password	Varchar(45)

EJB + JPA

- Agora crie a tabela estudante como abaixo:

Campo	Tipo
idestudante	int
nome	varchar(45)
sobrenome	Varchar(45)

EJB + JPA

- Agora vamos criar um projeto Java Web.
- Crie um projeto JavaWeb chamado **exemploejbjpa.**
- Selecione o Glassfish como servidor.
- Selecione JavaServer Faces e coloque como padrão de URL *.faces.

EJB + JPA

- Clique com o botão direito do projeto e vá em Novo -> Outros -> Glassfish -> Pool de Conexões JDBC.
- Deixe o nome como connectionPool.
- Selecione a opção Nova configuração usando banco de dados se selecione **MySQL (Driver MM MySQL)**.
- Clique em próximo.

EJB + JPA

- Altere a URL para jdbc:mysql://localhost:3306/aula07.
- Coloque o usuário e senha do MYSQL e finalize.

EJB + JPA

- Agora vamos criar o Recurso JDBC.
- Clique com o botão direito do mouse e clique em Novo -> Outros -> Glassfish -> Recurso JDBC.
- Selecione a opção Usar Pool de Conexão JDBC existente.
- Coloque o connectionPool
- Coloque o nome de **jdbc/meuRecursoEJBJPA**.

EJB + JPA

- Agora vamos criar uma Unidade de Persistência.
- Clique com o botão direito do mouse e clique em Novo -> Outros -> Persistência -> Unidade de Persistência.
- Coloque no nome como **exemploEJBJPAPU**.
- Em Origem de Dados coloque **jdbc/meuRecursoEJBJPA.**
- Em Estratégia de Geração de Tabelas deixe como nenhum.

EJB + JPA

- Crie 3 páginas JSF com os nomes login.xhtml, erro.xhtml e index.xhtml.

- Copie o código do dropbox e cole nessas páginas.

EJB + JPA

- login.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Login</title>
    </h:head>
    <h:body>
        <h:form>
            Usuário: <h:inputText value="#{loginMB.login.userName}"/> <br/>
            Senha: <h:inputText value="#{loginMB.login.password}"/> <br/>

            <h:commandButton value="Logar" action="#{loginMB.logar()}"/>
        </h:form>
    </h:body>
</html>
```

EJB + JPA

- erro.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        <h:form>
            Usuário não cadastrado.
            <br/>
            <h:commandButton value="Voltar" action="login"/>
        </h:form>
    </h:body>
</html>
```

EJB + JPA

- index.xhtml (1)

```
<h:body>
    <f:view>
        <h:form>
            Seja bem vindo <h:outputText value="#{loginMB.login.userName}!"/>


            <br/><br/><br/>

            Código: <h:inputText value="#{estudanteMB.estudante.idestudante}"/> <br/>
            Nome: <h:inputText value="#{estudanteMB.estudante.nome}" /> <br/>
            Sobrenome: <h:inputText value="#{estudanteMB.estudante.sobrenome}" /><br/>

            <h:commandButton value="Adicionar" action="#{estudanteMB.crud('Add')}"/>
             
            <h:commandButton action="#{estudanteMB.crud('Edit')}" value="Editar" />
             
            <h:commandButton value="Excluir" action="#{estudanteMB.crud('Deletar')}"/>
             
            <h:commandButton value="Procurar" action="#{estudanteMB.crud('Search')}"/>
             
```

EJB + JPA

- index.xhtml (2)

```
            <br/>
            <h:dataTable width="50%" var="item" value="#{estudanteMB.allEstudantes}">
                <h:column>
                    <f:facet name="header">
                        <h:outputText value="Nome"/>
                    </f:facet>
                    <h:outputText value="#{item.nome}" />
                </h:column>
                <h:column>
                    <f:facet name="header">
                        <h:outputText value="Sobrenome"/>
                    </f:facet>
                    <h:outputText value="#{item.sobrenome}" />
                </h:column>
            </h:dataTable>
        </h:form>
    </f:view>


</h:body>
```

EJB + JPA

- Crie um pacote com.model.
- Crie 2 Classes de Entidades chamadas Estudante e Login (Botão Direito no projeto Novo -> Outros -> Persistência -> Classe de Entidade).
- Cole o código do dropbox nelas.

EJB + JPA

- Login.java

```
@Entity
@Table
@NamedQueries({
    @NamedQuery(name = "Login.getAll", query = "SELECT e FROM Login e")})
public class Login implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userId;

    @Column
    private String userName;

    @Column
    private String password;
```

EJB + JPA

- Estudante.java

```java
@Entity
@Table
@NamedQueries({@NamedQuery(name = "Estudante.getAll", query = "SELECT e FROM Estudante e order by e.idestudante")})
public class Estudante implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column
    private Long idestudante;

    @Column
    private String nome;

    @Column
    private String sobrenome;

    public Estudante() {
    }

    public Estudante(Long idestudante, String nome, String sobrenome) {
        this.idestudante = idestudante;
        this.nome = nome;
        this.sobrenome = sobrenome;
    }
```

EJB + JPA

- Crie um pacote com.dao.
- Crie 2 Bean de Sessão, locais e sem estado, chamados EstudanteDAO e LoginDAO (Botão Direito no projeto Novo -> Outros ->Enterprise JavaBeans -> Bean de Sessão).
- Coloque o código do dropbox neles.

EJB + JPA

- LoginDAOLocal.java

```
@Local
public interface LoginDAOLocal {

    public boolean checkUser(String userName, String password);
}
```

EJB + JPA

- LoginDAO.java

```
@Stateless
public class LoginDAO implements LoginDAOLocal {

    @PersistenceContext
    private EntityManager em;

    @Override
    public boolean checkUser(String userName, String password) {
        List<Login> s = (List<Login>) em.createQuery("select e from Login e where "
                + "e.userName='" + userName + "' and e.password='" + password + "'").getResultList();
        System.out.println("is list empty ?" + s.isEmpty() + " for the" + userName + " and " + password);
        return !s.isEmpty();
    }
    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")
}
```

EJB + JPA

- EstudanteDAOLocal.java

```
@Local
public interface EstudanteDAOLocal {

    void addEstudante(Estudante estudante);

    void editEstudante(Estudante estudante);

    void deleteEstudante(Long studentId);

    Estudante getEstudante(Long studentId);

    List<Estudante> getAllEstudantes();
}
```

EJB + JPA

- EstudanteDAOL.java (1)

```
@Stateless
public class EstudanteDAO implements EstudanteDAOLocal {

    @PersistenceContext
    private EntityManager em;

    @Override
    public void addEstudante(Estudante estudante) {
        em.merge(estudante);
        em.flush();
    }

    @Override
    public void editEstudante(Estudante estudante) {
        em.merge(estudante);
        em.flush();
    }

    @Override
    public void deleteEstudante(Long studentId) {
        em.remove(getEstudante(studentId));
        em.flush();
    }
```

EJB + JPA

- EstudanteDAOL.java (2)

```
@Override
public Estudante getEstudante(Long studentId) {
    em.flush();
    return em.find(Estudante.class, studentId);
}

@Override
public List<Estudante> getAllEstudantes() {
    em.flush();
    return em.createNamedQuery("Estudante.getAll").getResultList();
}
// Add business logic below. (Right-click in editor and choose
// "Insert Code > Add Business Method")
```

EJB + JPA

- Para finalizar crie um pacote com.managedbean.
- Crie 2 Managed Beans (Botão Direito no projeto Novo -> Outros -> JavaServer Faces -> Bean Gerenciado JSF) chamados EstudanteMB.java e LoginMB.java.
- Copie o código do Dropbox nas classes.

EJB + JPA

- LoginMB.java

```
@ManagedBean
@SessionScoped
public class LoginMB {

    @EJB
    private LoginDAOLocal loginDao;

    private Login login = new Login();

    public Login getLogin() {
        return login;
    }

    public void setLogin(Login login) {
        this.login = login;
    }

    public String logar() {
        boolean check = loginDao.checkUser(login.getUserName(), login.getPassword());

        if (check) {
            return "index";
        } else {
            return "erro";
        }

    }
```

EJB + JPA

- EstudanteMB.java

```
@ManagedBean
@SessionScoped
public class EstudanteMB {

    @EJB
    private EstudanteDAOLocal estudanteDao;

    private Estudante estudante = new Estudante();
    private List<Estudante> allEstudantes;

    public String crud(String acao) {
        Estudante e = new Estudante(estudante.getIdestudante(), estudante.getNome(), estudante.getSobrenome());
        switch (acao) {
            case "Add":
                estudanteDao.addEstudante(e);
                break;
            case "Edit":
                estudanteDao.editEstudante(e);
                break;
            case "Delete":
                estudanteDao.deleteEstudante(e.getIdestudante());
                break;
            case "Search":
                estudanteDao.getEstudante(e.getIdestudante());
                break;
        }
        estudante = new Estudante();
        allEstudantes = estudanteDao.getAllEstudantes();
        return "index";
    }
```

EJB + JPA

- **Exercício**
  - Termine de realizer o crud. Somente o método adicionar está pronto. As ações da persistência já está corretas. Só é associar as operações corretamente.