Desenvolvimento Web II

Aula 15 - Incorporando o Banco de Dados em Aplicações JSF - Parte 2

Apresentação

Olá, seja bem-vindo(a) à nossa aula! Continuaremos com a incorporação de um banco de dados em um sistema de informação web desenvolvido com o framework JSF. Finalizamos nossa última aula criando a classe para gerenciamento de conexões com nossa base de dados e nossas tabelas para armazenamento de informações. Nesta aula iremos melhorar essa aplicação que passará a cadastrar informações no banco de dados, ou seja, nas tabelas as quais criamos na aula anterior. Assim, esse sistema será mais completo e poderá servir como ponto de partida para você desenvolver um produto comercial. Vamos começar?

Objetivos

- Aprender como implementar as classes para consulta e inserção em bancos de dados;
- Entender como alterar a aplicação JSF para inserir e consultar informações em uma base de dados.

Implementando as classes para realização de consultas e inserções no banco de dados

Então, já temos o banco de dados e as tabelas criadas para a nossa aplicação. Precisamos definir as classes para realização de consultas e inserções nessas tabelas. Para isso, criaremos no pacote br.ufrn.imd.dao as seguintes classes: CobradorDao, EmpresaDao, LinhaDao, MotoristaDao e OnibusDao. Todas as classes possuirão métodos para manipular as respectivas entidades no banco de dados SITURB. A classe CobradorDao, apresentada na Listagem 1, possui um método para consultar todos os cobradores (buscarTodosCobradores) e um método para inserir um cobrador (inserirCobrador) no banco de dados.

Observe que estamos usando as APIs do pacote java.sql.* para realização das nossas consultas e inserções nas nossas tabelas. Essas são as APIs fornecidas pela plataforma Java para manipulações em bancos de dados.

```java
package br.ufrn.imd.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import br.ufrn.imd.model.Cobrador;

/**
* Classe para manipulação de informações de cobradores na base de dados.
* @author itamir.filho
*
*/
public class CobradorDao {

   /**
    * Lista todos os cobradores cadastrados.
    * @return
    */
   public List<Cobrador> buscarTodosCobradores() {
      List<Cobrador> resultado = new ArrayList<Cobrador>();
      Connection con = GerenciadorConexao.getConexao();
      String sql = "select * from cobrador";
      try {
         PreparedStatement ps = con.prepareStatement(sql);
         ResultSet rs = ps.executeQuery();
         while (rs.next()) {
            Cobrador cobrador = new Cobrador();
            cobrador.setCpf(rs.getString("cpf"));
            cobrador.setEndereco(rs.getString("endereco"));
            cobrador.setMatricula(rs.getString("matricula"));
            cobrador.setNome(rs.getString("nome"));
            resultado.add(cobrador);
         }

      } catch (SQLException e) {
         e.printStackTrace();
      }
      return resultado;
   }


   /**
    * Método para realizar a inserção de um cobrador no BD.
    * @param cobrador
    */
   public void inserirCobrador(Cobrador cobrador) {
      Connection con = GerenciadorConexao.getConexao();
```

```
52      String sql = "insert into cobrador values (?,?,?,?)";
53      try {
54         PreparedStatement ps = con.prepareStatement(sql);
55         ps.setString(1, cobrador.getNome());
56         ps.setString(2, cobrador.getCpf());
57         ps.setString(3, cobrador.getMatricula());
58         ps.setString(4, cobrador.getEndereco());
59         ps.executeUpdate();
60      } catch (SQLException e) {
61         e.printStackTrace();
62      }
63
64   }
65 }
```

Listagem 1 - Código da classe CobradorDao.

Passamos para a classe EmpresaDao que é apresentada na Listagem 2. Nessa classe temos dois métodos: um para consultar todas as empresas inseridas (buscarTodasEmpresas) e outro para inserção de uma empresa (inserirEmpresa) na base de dados.

```java
package br.ufrn.imd.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import br.ufrn.imd.model.Empresa;

/**
* Classe para manipulação de informações das empresas na base de dados.
* @author itamir.filho
*
*/
public class EmpresaDao {

   /**

    * Lista todas as empresas cadastradas.
    * @return
    */
   public List<Empresa> buscarTodasEmpresas() {
      List<Empresa> resultado = new ArrayList<Empresa>();
      Connection con = GerenciadorConexao.getConexao();
      String sql = "select * from empresa";
      try {
         PreparedStatement ps = con.prepareStatement(sql);
         ResultSet rs = ps.executeQuery();
         while (rs.next()) {
            Empresa empresa = new Empresa();
            empresa.setCnpj(rs.getString("cnpj"));
            empresa.setRazaoSocial(rs.getString("razao_social"));
            resultado.add(empresa);
         }

         } catch (SQLException e) {
            e.printStackTrace();
         }
      return resultado;
      }


   /**
    * Método para realizar a inserção de uma empresa no BD.
    * @param empresa
    */
   public void inserirEmpresa(Empresa empresa) {
      Connection con = GerenciadorConexao.getConexao();
      String sql = "insert into empresa values (?,?)";
```

```
52        try {
53            PreparedStatement ps = con.prepareStatement(sql);
54            ps.setString(1, empresa.getRazaoSocial());
55            ps.setString(2, empresa.getCnpj());
56            ps.executeUpdate();
57        } catch (SQLException e) {
58            e.printStackTrace();
59        }
60
61    }
62 }
```

Listagem 2 - Código da classe EmpresaDao.

Passamos para a classe LinhaDao apresentada na Listagem 3, que possui dois métodos: um para consultar todas as linhas inseridas (buscarTodasLinhas) e outro para inserção de uma linha (inserirLinha) na base de dados.

```java
package br.ufrn.imd.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import br.ufrn.imd.model.Linha;

/**
* Classe para manipulação de informações de linhas na base de dados.
* @author itamir.filho
*
*/
public class LinhaDao {

   /**
    * Lista todas as linhas cadastradas.
    * @return
    */
   public List<Linha> buscarTodasLinhas() {
      List<Linha> resultado = new ArrayList<Linha>();
      Connection con = GerenciadorConexao.getConexao();
      String sql = "select * from linha";
      try {
         PreparedStatement ps = con.prepareStatement(sql);
         ResultSet rs = ps.executeQuery();
         while (rs.next()) {
            Linha linha = new Linha();
            linha.setIdent(rs.getString("ident"));
            linha.setOrigem(rs.getString("origem"));
            linha.setDestino(rs.getString("destino"));
            linha.setHoraSaida(rs.getString("hora_saida"));
            linha.setHoraChegada(rs.getString("hora_chegada"));
            resultado.add(linha);
         }

         } catch (SQLException e) {
            e.printStackTrace();
         }
         return resultado;
      }


   /**
    * Método para realizar a inserção de uma linha no BD.
    * @param linha
    */
   public void inserirLinha(Linha linha) {
```

```
52      Connection con = GerenciadorConexao.getConexao();
53      String sql = "insert into linha values (?,?,?,?,?)";
54      try {
55         PreparedStatement ps = con.prepareStatement(sql);
56         ps.setString(1, linha.getIdent());
57         ps.setString(2, linha.getOrigem());
58         ps.setString(3, linha.getDestino());
59         ps.setString(4, linha.getHoraSaida());
60         ps.setString(5, linha.getHoraChegada());
61         ps.executeUpdate();
62      } catch (SQLException e) {
63         e.printStackTrace();
64      }
65
66   }
67 }
```

Listagem 3 - Código da classe LinhaDao.

Vamos criar a classe MotoristaDao que é apresentada na Listagem 4. Nessa classe temos dois métodos: um para consultar todas os motoristas inseridos (buscarTodosMotoristas) e outro para inserção de um motorista (inserirMotorista) na base de dados.

```java
package br.ufrn.imd.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import br.ufrn.imd.model.Motorista;

/**
* Classe para manipulação de informações de motoristas na base de dados.
* @author itamir.filho
*
*/
public class MotoristaDao {

   /**
    * Lista todos os motoristas cadastrados.
    * @return
    */
   public List<Motorista> buscarTodosMotoristas() {
      List<Motorista> resultado = new ArrayList<Motorista>();
      Connection con = GerenciadorConexao.getConexao();
      String sql = "select * from motorista";
      try {
         PreparedStatement ps = con.prepareStatement(sql);
         ResultSet rs = ps.executeQuery();
         while (rs.next()) {
            Motorista motorista = new Motorista();
            motorista.setCpf(rs.getString("cpf"));
            motorista.setEndereco(rs.getString("endereco"));
            motorista.setMatricula(rs.getString("matricula"));
            motorista.setNome(rs.getString("nome"));
            motorista.setRegistroCnh(rs.getString("registro_cnh"));
            motorista.setCategoriaCnh(rs.getString("categoria_cnh"));
            resultado.add(motorista);
         }
         } catch (SQLException e) {
            e.printStackTrace();
         }
         return resultado;
      }


   /**
    * Método para realizar a inserção de um motorista no BD.
    * @param motorista
    */
   public void inserirMotorista(Motorista motorista) {
```

```
52      Connection con = GerenciadorConexao.getConexao();
53      String sql = "insert into motorista values (?,?,?,?,?,?)";
54      try {
55         PreparedStatement ps = con.prepareStatement(sql);
56         ps.setString(1, motorista.getNome());
57         ps.setString(2, motorista.getCpf());
58         ps.setString(3, motorista.getMatricula());
59         ps.setString(4, motorista.getEndereco());
60         ps.setString(5, motorista.getRegistroCnh());
61         ps.setString(6, motorista.getCategoriaCnh());
62         ps.executeUpdate();
63      } catch (SQLException e) {
64         e.printStackTrace();
65      }
66
67   }
68 }
```

Listagem 4 - Código da classe MotoristaDao.

Por fim, chegamos à classe OnibusDao apresentada na Listagem 5, que possui dois métodos: um para consultar todos os ônibus inseridos (buscarTodosOnibus) e outro para inserção de um ônibus (inserirOnibus) na base de dados.

```java
package br.ufrn.imd.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import br.ufrn.imd.model.Cobrador;
import br.ufrn.imd.model.Empresa;
import br.ufrn.imd.model.Linha;
import br.ufrn.imd.model.Motorista;
import br.ufrn.imd.model.Onibus;

/**
* Classe para manipulação de informações de onibus na base de dados.
* @author itamir.filho
*
*/
public class OnibusDao {

   /**
    * Lista todos os ônibus cadastrados.
    * @return
    */
   public List<Onibus> buscarTodosOnibus() {
      List<Onibus> resultado = new ArrayList<Onibus>();
      Connection con = GerenciadorConexao.getConexao();
      String sql = "select * from onibus";
      try {
         PreparedStatement ps = con.prepareStatement(sql);
         ResultSet rs = ps.executeQuery();
         while (rs.next()) {
            Onibus onibus = new Onibus();
            onibus.setAno(rs.getInt("ano"));
            onibus.setMarca(rs.getString("marca"));
            onibus.setModelo(rs.getString("modelo"));

            Cobrador cobrador = new Cobrador();
            cobrador.setNome(rs.getString("nome_cobrador"));
            onibus.setCobrador(cobrador);

            Motorista motorista = new Motorista();
            motorista.setNome(rs.getString("nome_motorista"));
            onibus.setMotorista(motorista);

            Empresa empresa = new Empresa();
            empresa.setRazaoSocial(rs.getString("razao_social_empresa"));
            onibus.setEmpresa(empresa);

```

```
52              Linha linha = new Linha();
53              linha.setIdent(rs.getString("ident_linha"));
54              onibus.setLinha(linha);
55
56              resultado.add(onibus);
57          }
58          } catch (SQLException e) {
59              e.printStackTrace();
60          }
61          return resultado;
62      }
63
64
65      /**
66       * Método para realizar a inserção de um ônibus no BD.
67       * @param onibus
68       */
69      public void inserirOnibus(Onibus onibus) {
70          Connection con = GerenciadorConexao.getConexao();
71          String sql = "insert into onibus values (?,?,?,?,?,?,?)";
72          try {
73              PreparedStatement ps = con.prepareStatement(sql);
74              ps.setString(1, onibus.getMarca());
75              ps.setString(2, onibus.getModelo());
76              ps.setInt(3, onibus.getAno());
77              ps.setString(4, onibus.getEmpresa().getRazaoSocial());
78              ps.setString(5, onibus.getLinha().getIdent());
79              ps.setString(6, onibus.getCobrador().getNome());
80              ps.setString(7, onibus.getMotorista().getNome());
81              ps.executeUpdate();
82          } catch (SQLException e) {
83              e.printStackTrace();
84          }
85
86      }
87  }
```

Listagem 5 - Código da classe OnibusDao.

Alterando os managed beans para utilizar o banco de dados

Com a implementação das classes definidas no pacote br.ufrn.imd.dao já podemos alterar nossos Managed Beans para persistir e consultar informações no banco de dados da nossa aplicação. Como fazer isso? Simples! Vamos incluir nos nossos controllers, definidos no pacote br.ufrn.imd.controllers, chamadas aos

métodos de inserção e busca definidos nas classes do pacote br.ufrn.imd.dao. Essas alterações se concentrarão nos métodos de obter as listas e nos métodos para cadastrar.

A Listagem 6 apresenta o código do controller para cadastro de cobradores. Observe que as alterações foram realizadas apenas nos métodos cadastrar() e getCobradores().

```java
package br.ufrn.imd.controllers;

import java.util.ArrayList;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import br.ufrn.imd.dao.CobradorDao;
import br.ufrn.imd.model.Cobrador;

/**
* Controller para cadastrar cobradores.
* @author itamir.filho
*
*/
@ManagedBean
@SessionScoped
public class CadastrarCobradorMBean {

   private Cobrador cobrador;

   private List<Cobrador> cobradores;

   public CadastrarCobradorMBean() {
      cobrador = new Cobrador();
      cobradores = new ArrayList<Cobrador>();
   }

   public String entrarCadastro(){
      return "/form_cobrador.jsf";
   }

   public String voltar(){
      return "/index.jsf";
   }

   public String cadastrar() {
      CobradorDao cobradorDao = new CobradorDao();
      cobradorDao.inserirCobrador(cobrador);
      cobrador = new Cobrador();
      FacesMessage msg = new FacesMessage("Cobrador cadastrado com sucesso!");
      msg.setSeverity(FacesMessage.SEVERITY_INFO);
      FacesContext.getCurrentInstance().addMessage("", msg);
      return "/form_cobrador.jsf";
   }

   public Cobrador getCobrador() {
      return cobrador;
```

```
52      }
53
54      public void setCobrador(Cobrador cobrador) {
55          this.cobrador = cobrador;
56      }
57
58      public List<Cobrador> getCobradores() {
59          CobradorDao cobradorDao = new CobradorDao();
60          cobradores = cobradorDao.buscarTodosCobradores();
61          return cobradores;
62      }
63
64      public void setCobradores(List<Cobrador> cobradores) {
65          this.cobradores = cobradores;
66      }
67  }
```

Listagem 6 - Código da classe CadastrarCobradorMBean.

A Listagem 7 apresenta o código da classe CadastrarEmpresaMBean. Observe que as alterações se concentraram nos métodos cadastrar() e getEmpresas().

```java
package br.ufrn.imd.controllers;

import java.util.ArrayList;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import br.ufrn.imd.dao.EmpresaDao;
import br.ufrn.imd.model.Empresa;

/**
* Controller para cadastro das empresas.
* @author itamir.filho
*
*/
@ManagedBean
@SessionScoped
public class CadastrarEmpresaMBean {

   private Empresa empresa;

   private List<Empresa> empresas;

   public CadastrarEmpresaMBean() {
      empresa = new Empresa();
      empresas = new ArrayList<Empresa>();
   }

   public String entrarCadastro(){
      return "/form_empresa.jsf";
   }

   public String voltar(){
      return "/index.jsf";
   }

   public String cadastrar() {
      EmpresaDao empresaDao = new EmpresaDao();
      empresaDao.inserirEmpresa(empresa);
      empresa = new Empresa();
      FacesMessage msg = new FacesMessage("Empresa cadastrada com sucesso!");
      msg.setSeverity(FacesMessage.SEVERITY_INFO);
      FacesContext.getCurrentInstance().addMessage("", msg);
      return "/form_empresa.jsf";
   }

   public Empresa getEmpresa() {
      return empresa;
```

```
52    }
53
54    public void setEmpresa(Empresa empresa) {
55       this.empresa = empresa;
56    }
57
58    public List<Empresa> getEmpresas() {
59       EmpresaDao empresaDao = new EmpresaDao();
60       empresas = empresaDao.buscarTodasEmpresas();
61       return empresas;
62    }
63
64    public void setEmpresas(List<Empresa> empresas) {
65       this.empresas = empresas;
66    }
67 }
```

Listagem 7 - Código da classe CadastrarEmpresaMBean.

A Listagem 8 apresenta o código da classe CadastrarLinhaMBean. Observe que as alterações se concentraram nos métodos cadastrar() e getLinhas().

```java
package br.ufrn.imd.controllers;

import java.util.ArrayList;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import br.ufrn.imd.dao.LinhaDao;
import br.ufrn.imd.model.Linha;

/**
* Controller para cadastrar linhas de ônibus.
* @author itamir.filho
*
*/
@ManagedBean
@SessionScoped
public class CadastrarLinhaMBean {

   private Linha linha;

   private List<Linha> linhas;

   public CadastrarLinhaMBean() {
      linha = new Linha();
      linhas = new ArrayList<Linha>();
   }

   public String entrarCadastro(){
      return "/form_linha.jsf";
   }

   public String voltar(){
      return "/index.jsf";
   }

   public String cadastrar() {
      LinhaDao linhaDao = new LinhaDao();
      linhaDao.inserirLinha(linha);
      linha = new Linha();
      FacesMessage msg = new FacesMessage("Linha cadastrada com sucesso!");
      msg.setSeverity(FacesMessage.SEVERITY_INFO);
      FacesContext.getCurrentInstance().addMessage("", msg);
      return "/form_linha.jsf";
   }

   public Linha getLinha() {
      return linha;
```

```
52      }
53
54      public void setLinha(Linha linha) {
55          this.linha = linha;
56      }
57
58      public List<Linha> getLinhas() {
59          LinhaDao linhaDao = new LinhaDao();
60          linhas = linhaDao.buscarTodasLinhas();
61          return linhas;
62      }
63
64      public void setLinhas(List<Linha> linhas) {
65          this.linhas = linhas;
66      }
67
68  }
```

Listagem 8 - Código da classe CadastrarLinhaMBean.

A Listagem 9 apresenta o código da classe CadastrarMotoristaMBean. Observe, mais uma vez, que as alterações se concentraram nos métodos cadastrar() e getMotoristas().

```java
package br.ufrn.imd.controllers;

import java.util.ArrayList;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import br.ufrn.imd.dao.MotoristaDao;
import br.ufrn.imd.model.Motorista;

/**
* Controller para cadastrar motoristas.
* @author itamir.filho
*
*/
@ManagedBean
@SessionScoped
public class CadastrarMotoristaMBean {

   private Motorista motorista;

   private List<Motorista> motoristas;

   public CadastrarMotoristaMBean() {
      motorista = new Motorista();
      motoristas = new ArrayList<Motorista>();
   }

   public String entrarCadastro(){
      return "/form_motorista.jsf";
   }

   public String voltar(){
      return "/index.jsf";
   }

   public String cadastrar() {
      MotoristaDao motoristaDao = new MotoristaDao();
      motoristaDao.inserirMotorista(motorista);
      motorista = new Motorista();
      FacesMessage msg = new FacesMessage("Motorista cadastrado com sucesso!");
      msg.setSeverity(FacesMessage.SEVERITY_INFO);
      FacesContext.getCurrentInstance().addMessage("", msg);
      return "/form_motorista.jsf";
   }

   public Motorista getMotorista() {
      return motorista;
```

```
52    }
53
54    public void setMotorista(Motorista motorista) {
55       this.motorista = motorista;
56     }
57
58    public List<Motorista> getMotoristas() {
59       MotoristaDao motoristaDao = new MotoristaDao();
60       motoristas = motoristaDao.buscarTodosMotoristas();
61       return motoristas;
62    }
63
64    public void setMotoristas(List<Motorista> motoristas) {
65       this.motoristas = motoristas;
66    }
67 }
```

Listagem 9 - Código da classe CadastrarMotoristaMBean.

Por fim, vamos à Listagem 10, a qual apresenta o código da classe CadastrarOnibusMBean. Observe que as alterações se concentraram nos métodos cadastrar() e getListagem().

```java
package br.ufrn.imd.controllers;

import java.util.ArrayList;
import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;

import br.ufrn.imd.dao.OnibusDao;
import br.ufrn.imd.model.Cobrador;
import br.ufrn.imd.model.Empresa;
import br.ufrn.imd.model.Linha;
import br.ufrn.imd.model.Motorista;
import br.ufrn.imd.model.Onibus;

/**
* Controller para cadastrar os ônibus.
* @author itamir.filho
*
*/
@ManagedBean
@SessionScoped
public class CadastrarOnibusMBean {

   private Onibus onibus;

   private List<Onibus> listagem;

   public CadastrarOnibusMBean() {
      iniciarValores();
      listagem = new ArrayList<Onibus>();
   }

   private void iniciarValores() {
      onibus = new Onibus();
      onibus.setMotorista(new Motorista());
      onibus.setLinha(new Linha());
      onibus.setEmpresa(new Empresa());
      onibus.setCobrador(new Cobrador());
   }

   public String entrarCadastro(){
      return "/form_onibus.jsf";
   }

   public String listar(){
      return "/list_onibus.jsf";
   }

```

```
52      public String voltar(){
53          return "/index.jsf";
54      }
55
56      public String cadastrar() {
57          OnibusDao onibusDao = new OnibusDao();
58          onibusDao.inserirOnibus(onibus);
59          iniciarValores();
60          FacesMessage msg = new FacesMessage("Ônibus cadastrado com sucesso!");
61          msg.setSeverity(FacesMessage.SEVERITY_INFO);
62          FacesContext.getCurrentInstance().addMessage("", msg);
63          return "/form_onibus.jsf";
64      }
65
66      public Onibus getOnibus() {
67          return onibus;
68      }
69
70      public void setOnibus(Onibus onibus) {
71          this.onibus = onibus;
72      }
73
74      public List<Onibus> getListagem() {
75          OnibusDao onibusDao = new OnibusDao();
76          listagem = onibusDao.buscarTodosOnibus();
77          return listagem;
78      }
79
80      public void setListagem(List<Onibus> listagem) {
81          this.listagem = listagem;
82      }
83  }
```

Listagem 10 - Código da classe CadastrarOnibusMBean.

Pronto, com essas alterações nos controllers que se resumem a instanciar classes do pacote br.ufrn.imd.dao e utilizar seus métodos de inserção e busca, nossa aplicação JSF está pronta para utilizar o banco de dados “siturb” para armazenar suas informações. Certifique-se que o driver do MySQL (arquivo mysql-connector-java-8.0.11.jar) está instalado na aplicação tanto no Java Build Path como na pasta WEB-INF/lib (se tiver problemas com o seu projeto não localizando o driver MySQL). Após essas alterações, inicie sua aplicação e acesse a URL http://localhost:8080/SITURB/ para ver o resultado.

Assim, chegamos ao fim da nossa aula! Agora você já sabe como criar aplicações web com o framework JSF e utilizar um banco de dados para armazenamento de informações. Que tal pegar aquela ideia e criar um produto comercial?

Resumo

Nessa aula finalizamos a integração da nossa aplicação JSF, denominada Sistema Integrado de Transportes Urbanos (SITURB), com o banco de dados. Para isso, criamos classes que realizam consultas e inserções no banco de dados utilizando a API JDBC e nosso gerenciador de conexão. Além disso, alteramos os controllers dessa aplicação para instanciar e utilizar os métodos definidos nessas classes. E por fim, com essas implementações e modificações, a nossa aplicação JSF passou a utilizar o banco de dados MySQL para armazenamento de informações.

Referências

<http://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html>.

<http://dev.mysql.com/downloads/connector/j/>.

<http://dev.mysql.com/doc/refman/5.6/en/index.html>.

<http://dev.mysql.com/doc/refman/5.6/en/sql-syntax.html>.