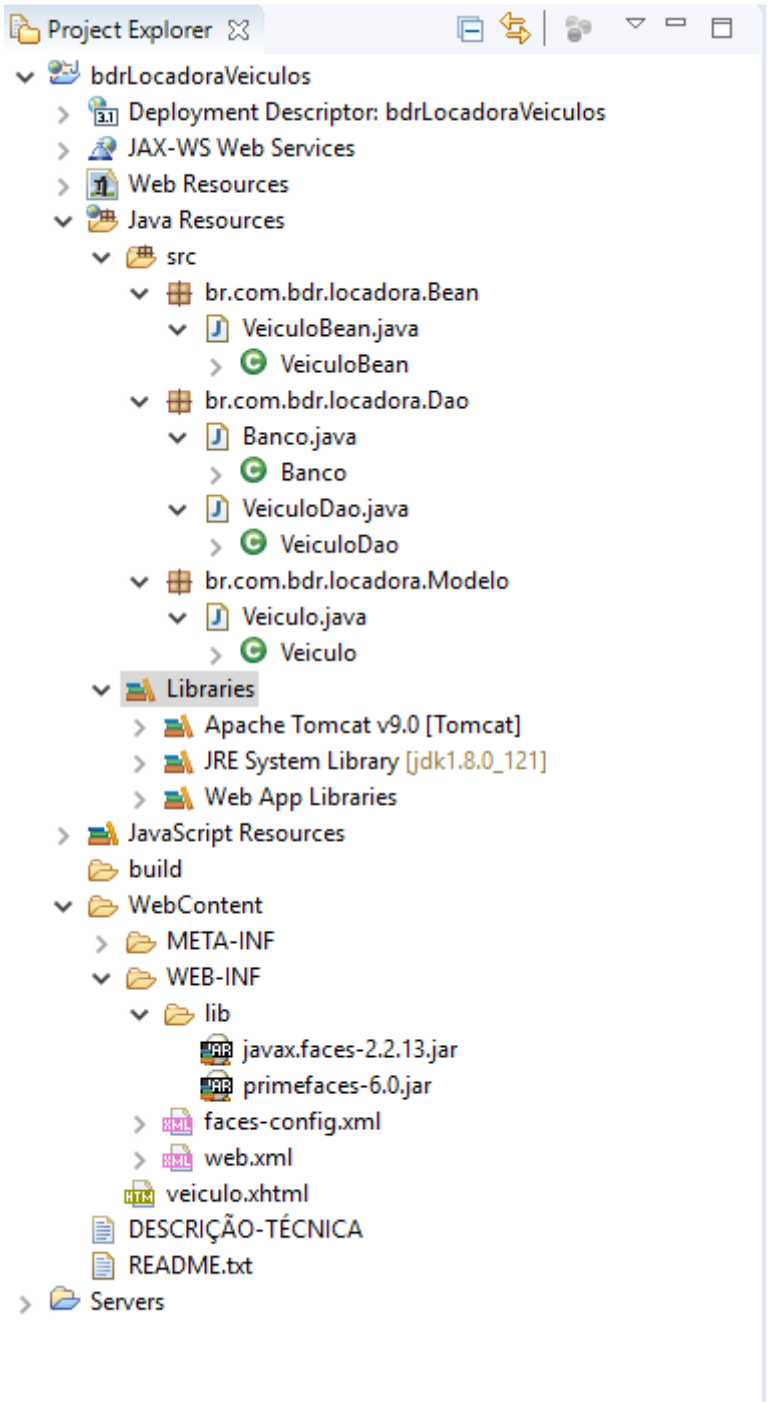
Software Project: Cadastro de Automóveis para Locadora de Veículos
Version: 1.0.0.0
See also: Software Release
Size: 6.86 MB
Autor: Gilson da Silva
Descrição:  Dynamic web Project - Software de exemplo, criado para atender aos requisitos de uma Avaliação Prática para vaga de Analista Desenvolvedor Java Júnior.



Contém as configurações e declarações JSF.

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    id="WebApp_ID" version="3.1">

    <display-name>bdrLocadoraVeiculos</display-name>

    <!-- difinindo arquivo que será inicado pela aplicação -->

    <welcome-file-list>
        <welcome-file>veiculo.xhtml</welcome-file>
    </welcome-file-list>

    <!-- cofiguração da Servlet que representa o Controlador -->

    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>


    <!-- alterando o mapeamento padrão para todas os páginas com extensão .xhtml -->

    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>


    <!-- chamada dos fontes do primefaces -->

    <context-param>
        <param-name>primefaces.FONT_AWESOME</param-name>
        <param-value>true</param-value>
    </context-param>

</web-app>
```

Classe que representa o modelo da aplicação

Veiculo.java

```java
package br.com.bdr.locadora.Modelo;

/**
 * Classe representa o Modelo
 */
public class Veiculo {

    /**
     * @param id
     *            - mostrado na interface como campo Código, é auto incremento e
     *            pode ser informado diretamente no campo, ao informar este
     *            valor a aplicação irá verificar se existe alguma entidade para
     *            alteração
     */
    private int id;
    /**
     * @param Campo
     *            de entrada de informação Placa do veículo
     */
    private String placa;
    /**
     * @param Campo
     *            de entrada de informação Marca do veículo
     */
    private String marca;
    /**
     * @param Campo
     *            de entrada de informação Modelo do veículo
     */
    private String modelo;
    /**
     * @param Campo
     *            de entrada de informação Tipo Combustível do veículo
     */
    private String combustivel;
    /**
     * @param Campo
     *            de entrada de informação Cor do veículo
     */
    private String cor;

    /**
     * @param Campo de entrada de informação Valor da Diária do veículo par alocação.
     * */
    private Double vl_diaria;

```

E seus respectivos geter e seters;

```
28
29      public String getMarca() {
30          return marca;
31      }
32
33      public void setMarca(String marca) {
34          this.marca = marca;
35      }
36
37      public String getModelo() {
38          return modelo;
39      }
40
41      public void setModelo(String modelo) {
42          this.modelo = modelo;
43      }
44
45      public String getCombustivel() {
46          return combustivel;
47      }
48
49      public void setCombustivel(String combustivel) {
50          this.combustivel = combustivel;
51      }
52
53      public String getCor() {
54          return cor;
55      }
56
57      public void setCor(String cor) {
58          this.cor = cor;
59      }
60
61      public Double getVl_diaria() {
62          return vl_diaria;
63      }
64
65      public void setVl_diaria(Double vl_diaria) {
66          this.vl_diaria = vl_diaria;
67      }
68  }
```

VeiculoDao.java

```java
package br.com.bdr.locadora.Dao;

import java.util.List;

/**
 * Classe DAO, responsavel pela comunicação com o banco de dados
 * */
public class VeiculoDao {

    /**
     * @param instancia de Classe Banco, utilizada para simular um banco em memória
     * */
    private Banco banco = new Banco();

    /**
     * @param Método que simula a chamada para persistir os dados
     * em banco fará a persistencia em memoria no arraylist
     * recebe e envia a entidade veiculo
     * */
    public void salva(Veiculo veiculo) {
        banco.save(veiculo);
    }

    /**
     * @param efetua chamada para listar todos os dados cdastrados
     * */
    public List<Veiculo> todosVeiculos() {
        return banco.listaVeiculos();
    }

    /**
     * @param efetua chamada para remover a entidade da memória
     * */
    public void removerVeiculo(Veiculo rem) {
        banco.remover(rem);
    }

}
```

arquivo responsável pela visão no MVC, contem as chamadas de URI, e a implementação dos formulários associados ao Bean uma classe gerenciada pelo JSF.

veiculo.xhtml

```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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:p="http://primefaces.org/ui"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

 <!-- Fim das chamadas de URi para utilização das tags de prefixo -->

<!-- tags de cabecalho necessárias para o funcionamento dos componentes primefaces -->

<h:head></h:head>
<h:body style="text-align: center; background: #CFD8DC;">

    <p:outputPanel style="margin: 40px 20px; font-size: 32px; background: #34495E; height: 50px; color: white; padding-top: 5px;">BDR - Locadora de Veículos</p:outputPanel>

    <!-- Formulário que contem os campos para entradas de dados do usuário -->

    <h:form id="frmCadastroVeiculos"
        style="width: 500px; text-align: center;">

        <p:messages />

        <p:fieldset legend="Cadastro de Automóveis">

            <h:panelGrid columns="2" style="text-align: left;">

                <p:outputLabel value="Código: *" for="id" />
                <p:inputText id="id" value="#{veiculoBean.veiculo.id}" />

                <p:outputLabel value="Placa: *" for="placa" />
                <p:inputMask id="placa" value="#{veiculoBean.veiculo.placa}"
                    mask="aaa-9999" />

                <p:outputLabel value="Marca: *" for="marca" />
                <p:inputText id="marca" value="#{veiculoBean.veiculo.marca}" />

                <p:outputLabel value="Modelo: *" for="modelo" />
                <p:inputText id="modelo" value="#{veiculoBean.veiculo.modelo}" />

                <p:outputLabel value="Combustivel: *" for="combustivel" />
                <p:inputText id="combustivel"
                    value="#{veiculoBean.veiculo.combustivel}" />

                <p:outputLabel value="Cor: *" for="cor" />
                <p:inputText id="cor" value="#{veiculoBean.veiculo.cor}" />
```

```xml
                 <p:outputLabel value="Valor Diária: *" for="vl_diaria" />
                 <p:inputNumber id="vl_diaria"
                     value="#{veiculoBean.veiculo.vl_diaria}" decimalSeparator="," thousandSeparator=".">
                     <p:ajax update="vl_diaria" />
                 </p:inputNumber>

                 <p:commandButton value="Salvar" action="#{veiculoBean.cadastra}"
                     update="@form :frmAutomoveisCadastrados" process="@form"
                     style="width: 140px;" />

            </h:panelGrid>

        </p:fieldset>

    </h:form>

    <!-- Estilização do Título da Grid -->
    <p:outputPanel style="margin: 15px 20px; font-size: 26px;">Automóves Cadastrados</p:outputPanel>


     <!-- Componente datatable da UI primefaces dentro do Formulário frmAutomoveisCadastrados com suas respectivas colunas onde são listados os dados cadastrados
        sendo possível em suas duas últimas colunas Editar e ou Remover as informações.
     -->

    <h:form id="frmAutomoveisCadastrados">
        <p:dataTable value="#{veiculoBean.veiculos}" var="car" emptyMessage="Nenhum Automóvel Cadastrado"
            id="tabelaAutomoveis" paginator="true" rows="4">

            <p:column sortBy="#{car.id}">
                <f:facet name="header">Código</f:facet>
                <h:outputText value="#{car.id}"/>
            </p:column>

            <p:column>
                <f:facet name="header">Placa</f:facet>
                <h:outputText value="#{car.placa}" />
            </p:column>

            <p:column sortBy="#{car.marca}">
                <f:facet name="header">Marca</f:facet>
                <h:outputText value="#{car.marca}" />
            </p:column>

            <p:column>
                <f:facet name="header">Modelo</f:facet>
                <h:outputText value="#{car.modelo}" />
            </p:column>
```

```
            <p:column>
                <f:facet name="header">Combustível</f:facet>
                <h:outputText value="#{car.combustivel}" />
            </p:column>

            <p:column>
                <f:facet name="header">Cor</f:facet>
                <h:outputText value="#{car.cor}" />
            </p:column>

            <p:column>
                <f:facet name="header">Valor Diária</f:facet>
                <h:outputText value="#{car.vl_diaria}" />
            </p:column>

            <p:column>
                <f:facet name="header">Editar</f:facet>
                <h:commandLink value="Editar" action="#{veiculoBean.carregar(car)}" />
            </p:column>

            <p:column>
                <f:facet name="header">Remover</f:facet>
                <p:commandLink value="x" action="#{veiculoBean.remover(car)}" update="tabelaAutomoveis" process="@this"/>
            </p:column>

        </p:dataTable>

    </h:form>
</h:body>
</html>

```

Classe que simulação operações no banco

Banco.java

```java
package br.com.bdr.locadora.Dao;

import java.util.ArrayList;

public class Banco {

    /**
     * instanciando array que armazenará os dados persistidos
     */
    public static List<Veiculo> veiculos = new ArrayList<Veiculo>();

    /**
     * @param cria a sequência para o ID sempre que for um novo cadastro
     */
    private static int chave = 1;

    /**
     * @param recebe a entidade Veiculo, verifica se existe um ID igual, se existir
     * efetua alteração do contrário cria um novo registro
     */
    public void save(Veiculo veiculo) {

        int verifica = 0;
        for (Veiculo busca : veiculos) {
            if (busca.getId() == veiculo.getId()) {
                verifica++;
            }
        }

        if (verifica > 0) {
            for (Veiculo x : veiculos) {
                if (x.getId() == veiculo.getId()) {
                    x.setPlaca(veiculo.getPlaca());
                    x.setMarca(veiculo.getMarca());
                    x.setModelo(veiculo.getModelo());
                    x.setCombustivel(veiculo.getCombustivel());
                    x.setCor(veiculo.getCor());
                    x.setVl_diaria(veiculo.getVl_diaria());
                }
            }
        } else {

            veiculo.setId(chave++);
            veiculos.add(veiculo);

        }
    }

    /**
     * @param recebe a entidade e remove do array
     */
    public void remover(Veiculo rem) {
        veiculos.remove(rem);
    }

    /**
     * retorna todos os registros
     */
    public List<Veiculo> listaVeiculos() {
        return veiculos;
    }

}
```

Bean Gerenciado pela aplicação, possui regras de negócio como validações e faz a comunicação entre visão e modelo para persistência dos dados.

VeiculoBean.java

```java
package br.com.bdr.locadora.Bean;

import java.util.List;

/**
 * Classe que representa o Controler
 */
@ManagedBean
public class VeiculoBean {

    /**
     * Instacia veículo
     */
    private Veiculo veiculo = new Veiculo();

    /**
     * Instacia Dao
     */
    private VeiculoDao dao = new VeiculoDao();

    public Veiculo getVeiculo() {
        return veiculo;
    }

    /**
     * método mapeado na visão, chama a validação de campos e se nao retornar
     * erro delega para classe dao dar seguencia. após instancia a entidade para
     * limpar os dados dos componentes da tela.
     */
    public void cadastra() {

        if (validarCampos()) {

            this.dao.salva(veiculo);
            this.veiculo = new Veiculo();

        }
    }

    /**
     * Chamada para o método remover do Dao, remove a entidade.
     */
    public void remover(Veiculo veiculo) {
        dao.removerVeiculo(veiculo);
    }

    /**
     * Este Método carrega os dados no formulário quando clicado na opção Editar
     * da Grid
     */
    public void carregar(Veiculo veiculo) {
        this.veiculo = veiculo;
```

```
59          this.veiculo = veiculo;
60      }
61
62      /**
63       * Chamada para o método que lista todos as entidades;
64       */
65      public List<Veiculo> getVeiculos() {
66          return this.dao.todosVeiculos();
67      }
68
69      /**
70       * méthodo de validação de campos
71       *
72       */
73      private boolean validarCampos() {
74
75          /**
76           * Valida se Codigo é negativo mostra mensagem do contrário o Modelo
77           * setará sempre 0, e a classe banco cria novo registro se este for null
78           * ou zero ou se numero não estiver em memória.
79           */
80          if (veiculo.getId() < 0) {
81              FacesContext.getCurrentInstance().addMessage("id",
82                      new FacesMessage("Deixe em branco ou Zero (0) para seguir seguência automática."
83                              + " Informe o Código e as Demais informações para Editar um Automóvel."
84                              + " Ou ainda poderá visualizar os dados para edição pelo link na grid a baixo."));
85              return false;
86          }
87
88          /**
89           * Valida campo Placa, não pode ser nulo
90           */
91          if (veiculo.getPlaca().isEmpty()) {
92              FacesContext.getCurrentInstance().addMessage("placa",
93                      new FacesMessage("A placa é uma informação Obrigatória."));
94              return false;
95          }
96
97          /**
98           * Valida campo Marca, não pode ser nulo
99           */
100         if (veiculo.getMarca().isEmpty()) {
101             FacesContext.getCurrentInstance().addMessage("ano",
102                     new FacesMessage("A marca é uma informação Obrigatória."));
103             return false;
104         }
```

```
105
106         /**
107          * Valida campo Modelo, não pode ser nulo
108          */
109         if (veiculo.getModelo().isEmpty()) {
110             FacesContext.getCurrentInstance().addMessage("modelo",
111                     new FacesMessage("O modelo é uma informação Obrigatória."));
112             return false;
113         }
114
115         /**
116          * Valida campo Combustivel, não pode ser nulo
117          */
118         if (veiculo.getCombustivel().isEmpty()) {
119             FacesContext.getCurrentInstance().addMessage("combustivel",
120                     new FacesMessage("O Combustível é uma informação Obrigatória."));
121             return false;
122         }
123
124         /**
125          * Valida campo Cor, não pode ser nulo
126          */
127         if (veiculo.getCor().isEmpty()) {
128             FacesContext.getCurrentInstance().addMessage("cor",
129                     new FacesMessage("A cor é uma informação Obrigatória."));
130             return false;
131         }
132
133         /**
134          * Valida campo Valor Diária, não pode ser nulo
135          */
136         if (veiculo.getVl_diaria() == null || veiculo.getVl_diaria() < 150) {
137             FacesContext.getCurrentInstance().addMessage("vl_diaria",
138                     new FacesMessage("O Valor mínimo para locação diária é R$ 150,00."));
139             return false;
140         }
141
142         /**
143          * Se todos ok retorna verdadeiro para proseguir senão devolve msg para
144          * usuário na tela.
145          */
146         return true;
147
148     }
149
150 }
```