Projeto Servlet/Front-end de Exemplo – PARTE 1

Professor Vicente Paulo de Camargo

Projeto Servlet/Front-end de Exemplo

Esse projeto é uma continuação do projeto **SistemaEstoque**

O ideal é criar um projeto web

Dessa forma, crie um projeto web

Provavelmente, você já construiu o banco de dados do projeto anterior

Caso não tenha construído, segue a estrutura da tabela do banco de dados

#	Nome	Tipo de dados	Tamanho/Ite...	Unsign...	Permitir...	Zerofill	Padrão
1	id	INT	11	☐	☐	☐	AUTO_INCREMENT
2	descricao	VARCHAR	50	☐	☑	☐	NULL
3	precounit	DOUBLE		☐	☑	☐	NULL
4	quantidade	DOUBLE		☐	☑	☐	NULL

Esse banco de dados foi construído com o SGBD MySQL

Utilize qualquer SGBD para construir o seu banco de dados

Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.modelo**

Em seguida, crie a classe **Estoque.java**, com a seguinte estrutura:

```java
package br.edu.pucgoias.sistemaestoque.modelo;

public class Estoque{

    private int id;
    private String descricao;
    private double quantidade;
    private double precounit;

    private String mensagem;
    private int controle; //1=inclusão, 2=alteração, 3=exclusão

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getDescricao() {
        return descricao;
    }
    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
    public double getQuantidade() {
        return quantidade;
    }
    public void setQuantidade(double quantidade) {
        this.quantidade = quantidade;
    }
    public double getPrecounit() {
        return precounit;
    }
    public void setPrecounit(double precounit) {
        this.precounit = precounit;
    }

    public String getMensagem() {
        return mensagem;
    }
    public void setMensagem(String mensagem) {
        this.mensagem = mensagem;
    }
    public int getControle() {
        return controle;
    }
    public void setControle(int controle) {
        this.controle = controle;
    }
    @Override
    public String toString() {
        return "Estoque [id=" + id + ", descricao=" + descricao + ", quantidade=" + quantidade + ", precounit="
                + precounit + "]";
    }


}
```

Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.dao**

Em seguida, crie a classe **BaseDao.java** para conexão com o banco de dados, com a seguinte estrutura:

```java
package br.edu.pucgoias.sistemaestoque.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class BaseDao
{

    public BaseDao()
    {
        try {
        Class.forName("com.mysql.jdbc.Driver");
        }
        catch(ClassNotFoundException e) {
        }
   }

   public Connection getConnection() throws SQLException{
       String url="jdbc:mysql://localhost/sistema_estoque_web";
       Connection conn = (Connection) DriverManager.getConnection(url, "vicente", "vicente");
       return conn;
   }

   public static void main(String[] args) throws SQLException {
       BaseDao bd = new BaseDao();
       Connection conn = bd.getConnection();
       if (conn != null)
           System.out.println("Conectou !");
       else
           System.out.println("Não conectou");


   }
}
```

Ajuste apenas as configurações para acessar o seu banco de dados

Projeto Servlet/Front-end de Exemplo

Em seguida, crie a classe **EstoqueDao.java** no pacote **br.edu.pucgoias.sistemaestoque.dao**

Essa classe possui dois métodos para salvar dados e dois métodos para excluir item do estoque

No entanto, apenas os métodos correspondentes que retornam objetos de Estoque serão explicados nesse conteúdo, pois utilizam o encapsulamento de mensagem dentro do próprio objeto Estoque, facilitando o envio e visualização de mensagens para o usuário

Os outros dois métodos se assemelham com os explicados, apenas retornam um boolean para indicar se a operação foi ou não bem sucedida e, neste caso, não deixa a aplicação muito amigável para o usuário

Projeto Servlet/Front-end de Exemplo

Inicio do código da classe**EstoqueDao.java**

```java
package br.edu.pucgoias.sistemaestoque.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 com.mysql.jdbc.Statement;

import br.edu.pucgoias.sistemaestoque.modelo.Estoque;

public class EstoqueDao extends BaseDao{

    public Estoque getEstoqueViaId(int id) {

        Estoque estoque = null;
        PreparedStatement pstm = null;
        Connection conn = null;
        try {
            conn = this.getConnection();
            pstm = conn.prepareStatement("select * from estoque where id=?");
            pstm.setInt(1,id);
            ResultSet rs = pstm.executeQuery();
            if (rs.next())
            {
                estoque = criaEstoque(rs);
            }

        }
        catch(SQLException e)
        {
           estoque = null;
        }
        return estoque;
    }
```

Esse método é responsável por receber o código de um item do estoque (id), consultar esse item e retornar o seu objeto para a classe de controle. Caso não encontre, o objeto estoque será null

Projeto Servlet/Front-end de Exemplo

EstoqueDao.java (Cont.)

```
38
39      public Estoque criaEstoque(ResultSet rs) throws SQLException {
40          Estoque estoque = new Estoque();
41          estoque.setDescricao(rs.getString("descricao"));
42          estoque.setId(rs.getInt("id"));
43          estoque.setQuantidade(rs.getDouble("quantidade"));
44          estoque.setPrecounit(rs.getDouble("precounit"));
45
46          return estoque;
47      }
48
49      public List<Estoque> getEstoqueViaNome(String nome){
50
51          List<Estoque> lista = new ArrayList<>();
52          Estoque estoque = new Estoque();
53          PreparedStatement pstm = null;
54          Connection conn = null;
55          try {
56              conn = this.getConnection();
57              String sql="select * from estoque where lower(descricao) like ? order by descricao";
58              pstm = conn.prepareStatement(sql);
59              pstm.setString(1,"%"+nome.toLowerCase()+"%");
60              ResultSet rs = pstm.executeQuery();
61              while (rs.next())
62              {
63                  estoque = criaEstoque(rs);
64                  lista.add(estoque);
65              }
66
67          }
68          catch(SQLException e)
69          {
70             lista=null;
71          }
72          return lista;
73      }
```

O método criaEstoque permite criar uma instância de um objeto, popular esse objeto com os dados de um registro e retornar esse objeto para o local que chamou o método.

O método getEstoqueViaNome recebe um nome ou parte de um nome, faz a consulta e retorna um ArrayList dos objetos dos itens encontrados. Retorna um null se não encontrou itens. Esse método não é utilizado na página index.html

Projeto Servlet/Front-end de Exemplo

O método salvarEstoqueMsg recebe um objeto Estoque. Caso o id desse objeto for = 0, então é uma inclusão. Caso contrário será uma alteração. As linhas 162 e 163 obtém o id do novo item conforme designado na linha 86. Para inclusão ou edição, o objeto Estoque é retornado com uma mensagem que será utilizada pelo controle que a enviará para o Servlet retornar via Json para a página apresentar o conteúdo da mensagem.

O método getGeneratedId obtem o id gerado pelo PreparedStatement e o retorna para o local que chamou esse método.

EstoqueDao.java (Cont.)

```
120     public Estoque salvarEstoqueMsg(Estoque estoque) {
121         boolean retorno=false;
122         String sql="";
123         PreparedStatement pstm = null;
124         Connection conn = null;
125         try {
126                 conn = this.getConnection();
127                 if (estoque.getId() == 0)
128                 {
129                     //fazer uma consulta para ver se já existe item com essa descrição
130
131                     sql= "insert into estoque (descricao, precounit, quantidade) ";
132                     sql+=" values (?, ?, ?)";
133                     pstm = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);
134                     estoque.setControle(1);
135                 }
136                 else
137                 {
138                     sql= "update estoque set descricao=?, precounit=?, quantidade=? ";
139                     sql+=" where id = ?";
140                     pstm = conn.prepareStatement(sql);
141                     estoque.setControle(2);
142                 }
143                 pstm.setString(1, estoque.getDescricao());
144                 pstm.setDouble(2, estoque.getPrecounit());
145                 pstm.setDouble(3, estoque.getQuantidade());
146                 if (estoque.getId() !=0)
147                 {
148                     pstm.setInt(4,estoque.getId());
149                 }
150                 int idAux = pstm.executeUpdate();
151                 if (idAux==0)
152                 {
153                     if (estoque.getControle()==1)
154                         estoque.setMensagem("Inclusão não foi realizada");
155                     if (estoque.getControle()==2)
156                         estoque.setMensagem("Alteração não foi realizada");
157                 }
158                 else
159                 {
160                     if (estoque.getId()==0)
161                     {
162                         int idInserir = getGeneratedId(pstm);
163                         estoque.setId(idInserir);
164                     }
165                     if (estoque.getControle()==1)
166                         estoque.setMensagem("Inclusão realizada com sucesso!!");
167                     if (estoque.getControle()==2)
168                         estoque.setMensagem("Alteração realizada com sucesso!!");
169                 }
170         }
171         catch(SQLException e)
172         {
173             //melhorar esse código
174             if (estoque.getControle()==1)
175                 estoque.setMensagem("Erro ao tentar incluir dados");
176             if (estoque.getControle()==2)
177                 estoque.setMensagem("Erro ao tentar alterar dados");
178         }
179
180
181         return estoque;
182
183     }
184     public int getGeneratedId(PreparedStatement stm) throws SQLException {
185         ResultSet rs = stm.getGeneratedKeys();
186         if (rs.next())
187         {
188             int id = rs.getInt(1);
189             return id;
190         }
191         return 0;
192     }
```

Projeto Servlet/Front-end de Exemplo

EstoqueDao.java (Cont.)

```
213     public Estoque excluirMsg(int id) {
214         Estoque estoque=null;
215 
216         PreparedStatement pstm = null;
217         Connection conn = null;
218         try {
219             conn = this.getConnection();
220             String sql="delete from estoque where id = ?";
221             pstm = conn.prepareStatement(sql);
222             pstm.setInt(1,id);
223             int conta =pstm.executeUpdate();
224             if (conta>0)
225             {
226                 estoque = new Estoque();
227                 estoque.setControle(3);
228                 estoque.setMensagem("Exclusão efetuada com sucesso");
229             }
230         }
231         catch(SQLException e)
232         {
233             estoque = new Estoque();
234             estoque.setMensagem("Erro de exclusão");
235         }
236         return estoque;
237     }
```

O método excluirMsg recebe o código de um item do estoque (id) e efetua a sua exclusão do banco de dados, retornando um objeto Estoque com a mensagem que será apresentada pelo front-end

Projeto Servlet/Front-end de Exemplo

EstoqueDao.java (Cont.)

```
241     public List<Estoque> getTodos(){
242
243         List<Estoque> lista = new ArrayList<>();
244         Estoque estoque = new Estoque();
245         PreparedStatement pstm = null;
246         ResultSet rs;
247         Connection conn = null;
248         try {
249             conn = this.getConnection();
250             String sql="select * from estoque order by descricao";
251             pstm = conn.prepareStatement(sql);
252             rs = pstm.executeQuery();
253             while (rs.next())
254             {
255                 estoque = criaEstoque(rs);
256                 lista.add(estoque);
257             }
258
259         }
260         catch(SQLException e)
261         {
262            return lista;
263         }
264         return lista;
265
266     }
```

O método getTodos retorna uma lista com todos os itens cadastrados no banco de dados. Note que o select faz com que a busca se apresente por ordem alfabética da descrição do item

Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.controle**

Em seguida, crie a classe **EstoqueControle.java**, com a seguinte estrutura:

```java
package br.edu.pucgoias.sistemaestoque.controle;

import java.util.List;

public class EstoqueControle {

    private EstoqueDao ed = new EstoqueDao();

    public List<Estoque> getEstoque(){
        List<Estoque> estoques = ed.getTodos();
        return estoques;
    }

    public Estoque getEstoquePorId(int id) {
        return ed.getEstoqueViaId(id);
    }

    public boolean excluir(int id) {
        return ed.excluir(id);
    }

    public Estoque excluirMsg(int id) {
        return ed.excluirMsg(id);
    }

    public boolean salvar(Estoque estoque) {
        return ed.salvarEstoque(estoque);
    }

    public Estoque salvarComMsg(Estoque estoque) {
        return ed.salvarEstoqueMsg(estoque);
    }

    public List<Estoque> buscaEstoquePorNome(String nome){
        return ed.getEstoqueViaNome(nome);
    }
}
```

Os métodos excluir e salvar não estão sendo utilizados pela aplicação, pois não utilizam a mensagem dentro do objeto Estoque. Os métodos dessa classe são auto explicativos. Ressalta-se que essa classe é instanciada nos respectivos Servlets que recebem os retornos. Estes são enviados para as chamadas via Ajax na página index.html.

Projeto Servlet/Front-end de Exemplo

Crie o pacote **br.edu.pucgoias.sistemaestoque.servlets**

Esse pacote armazenará todos os Servlets da aplicação.

Em seguida, pesquise na Internet o arquivo **gson-2.3.1.jar**, que é uma biblioteca da Google que permite manipular as informações para que possam ser retornadas no formato JSON

Copie esse arquivo para a pasta **WebContent/WEB-INF/lib** do seu projeto

Projeto Servlet/Front-end de Exemplo

Em seguida, crie a classe (Servlet) **ServletAll.java**, com a seguinte estrutura:

```
1  package br.edu.pucgoias.sistemaestoque.servlets;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import java.util.ArrayList;
6  import java.util.List;
7
8  import javax.servlet.ServletException;
9  import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import com.google.gson.Gson;
15 import com.google.gson.GsonBuilder;
16
17 import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
18 import br.edu.pucgoias.sistemaestoque.modelo.Estoque;
19
20 @WebServlet("/servletall")
21 public class ServletAll extends HttpServlet {
22     private static final long serialVersionUID = 1L;
23     private Gson gson;
24     public ServletAll() {
25         super();
26     }
27
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29
30         gson = new GsonBuilder().setPrettyPrinting().create();
31         Estoque estoque;
32         List<Estoque> lista = new ArrayList<>();
33
34         String retorno ="ERRO";
35         boolean acao=false;
36         EstoqueControle ec = new EstoqueControle();
37         lista = ec.getEstoque();
38
39         if (lista != null )
40         {
41             String retornoJsonString = this.gson.toJson(lista);
42             PrintWriter out = response.getWriter();
43             response.setContentType("application/json");
44             response.setCharacterEncoding("UTF-8");
45             out.print(retornoJsonString);
46             out.flush();
47         }
48         else
49         {
50             retorno="Itens inexistentes";
51             String retornoJsonString = this.gson.toJson(retorno);
52             PrintWriter out = response.getWriter();
53             response.setContentType("application/json");
54             response.setCharacterEncoding("UTF-8");
55             out.print(retornoJsonString);
56             out.flush();
57         }
58     }
59
60     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
61         doGet(request, response);
62     }
63
64 }
```

O objeto **gson** da classe **Gson** é declarado na linha 23. Em seguida, esse objeto é criado na linha 30. Essa forma de criação permite apresentar os objetos Json mais estruturados. Note a chamada ao método **getEstoque** da classe **EstoqueControle.java**, que retorna uma lista de objetos do estoque, a qual é convertida em um array de objetos na linha 41, caso possua elementos.

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletConsultaPorID.java**, com a seguinte estrutura:

```java
package br.edu.pucgoias.sistemaestoque.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
import br.edu.pucgoias.sistemaestoque.modelo.Estoque;

@WebServlet("/servletconsultaporid")
public class ServletConsultaPorID extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private Gson gson;
    public ServletConsultaPorID() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        gson = new GsonBuilder().setPrettyPrinting().create();
        Estoque estoque = new Estoque();

        int id=0;
        String strId= request.getParameter("id");

        id = Integer.parseInt(strId);

        String retorno ="ERRO";
        boolean acao=false;
        if ((strId == null || strId.length()==0) || strId.isEmpty() && id == 0)
            estoque.setMensagem("Informação inválida");
        else
        {
            EstoqueControle ec = new EstoqueControle();
            estoque = ec.getEstoquePorId(id);

        }
        if (estoque !=null)
        {
            String retornoJsonString = this.gson.toJson(estoque);
            PrintWriter out = response.getWriter();
            response.setContentType("application/json");
            response.setCharacterEncoding("UTF-8");
            out.print(retornoJsonString);
            out.flush();
        }
        else
        {
            String retornoJsonString = this.gson.toJson(retorno);
            PrintWriter out = response.getWriter();
            response.setContentType("application/json");
            response.setCharacterEncoding("UTF-8");
            out.print(retornoJsonString);
            out.flush();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}
```

Esse Servlet retorna um objeto instanciado para armazenar o item pesquisado ou retorna uma mensagem caso o item não seja encontrado no banco de dados.

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletEditarPorID.java**, com a seguinte estrutura:

```java
package br.edu.pucgoias.sistemaestoque.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
import br.edu.pucgoias.sistemaestoque.modelo.Estoque;

@WebServlet("/servleteditarporid")
public class ServletEditarPorID extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private Gson gson;

    public ServletEditarPorID() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        gson = new GsonBuilder().setPrettyPrinting().create();
        Estoque estoque=null;
        String descricao=request.getParameter("descricao");
        double precounit=0;
        double quantidade=0;
        int id=0;
        String strPu = request.getParameter("precounit");
        String strQt = request.getParameter("quantidade");
        String strId= request.getParameter("id");

        precounit = Double.parseDouble(strPu);
        quantidade = Double.parseDouble(strQt);
        id = Integer.parseInt(strId);

        String retorno ="ERRO";
        boolean acao=false;
        if ((descricao == null || descricao.length()==0) && id == 0)
        {
            estoque = new Estoque();
            estoque.setId(0);
            estoque.setMensagem("Descrição inválida");
        }
        else
        {
                estoque=new Estoque();
                estoque.setDescricao(descricao);
                estoque.setPrecounit(precounit);
                estoque.setQuantidade(quantidade);
                estoque.setId(id);
                EstoqueControle ec = new EstoqueControle();
                estoque = ec.salvarComMsg(estoque);
        }

        String retornoJsonString = this.gson.toJson(estoque);
        PrintWriter out = response.getWriter();
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        out.print(retornoJsonString);
        out.flush();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

}
```

Esse Servlet os campos de um item de estoque que foram editados no front end. Em seguida, efetua a atualização dos dados e retorna o objeto atualizado no formato Json.

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletExcluirPorID.java**, com a seguinte estrutura:

```java
package br.edu.pucgoias.sistemaestoque.servlets;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
import br.edu.pucgoias.sistemaestoque.modelo.Estoque;

@WebServlet("/servletexcluirporid")
public class ServletExcluirPorID extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private Gson gson;
    public ServletExcluirPorID() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        gson = new GsonBuilder().setPrettyPrinting().create();
        Estoque estoque = new Estoque();

        int id=0;
        String strId= request.getParameter("id");

        id = Integer.parseInt(strId);
        boolean acao=false;
        String retorno ="ERRO";
        if ((strId == null || strId.length()==0) || strId.isEmpty() && id == 0)
            retorno ="Informação inválida";
        else
        {
            EstoqueControle ec = new EstoqueControle();
            estoque = ec.excluirMsg(id);
        }
        }
            String retornoJsonString = this.gson.toJson(estoque);
            PrintWriter out = response.getWriter();
            response.setContentType("application/json");
            response.setCharacterEncoding("UTF-8");
            out.print(retornoJsonString);
            out.flush();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }
}
```

Esse Servlet recebe o código de um item do estoque e efetua sua exclusão, retornando um objeto, no formato Json, com uma mensagem para o front-end.

Projeto Servlet/Front-end de Exemplo

Crie a classe (Servlet) **ServletSalvarDados.java**, com a seguinte estrutura:

```java
package br.edu.pucgoias.sistemaestoque.servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import br.edu.pucgoias.sistemaestoque.controle.EstoqueControle;
import br.edu.pucgoias.sistemaestoque.modelo.Estoque;

@WebServlet("/servletsalvardados")
public class ServletSalvarDados extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private Gson gson;

    public ServletSalvarDados() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        gson = new GsonBuilder().setPrettyPrinting().create();
        String descricao=request.getParameter("descricao");
        double precounit=0;
        double quantidade=0;
        int id=0;
        String strPu = request.getParameter("precounit");
        String strQt = request.getParameter("quantidade");
        String strId= request.getParameter("id");

        if (strPu ==null || strPu.length() ==0 || strPu.isEmpty())
            precounit=0;
        else
            precounit = Double.parseDouble(strPu);

        if (strQt  == null || strQt.length() ==0 || strQt.isEmpty())
            quantidade = 0;
        else
            quantidade = Double.parseDouble(strQt);

        if (strId ==null || strId.length() ==0 || strId.isEmpty())
            id=0;
        else
            id = Integer.parseInt(strId);

        Estoque estoque=null;
        String retorno ="ERRO";
        List<Estoque> lista = new ArrayList<>();
        boolean acao=false;
        if (descricao == null || descricao.length()==0 || descricao.isEmpty())
        {
            estoque.setMensagem("Descrição inválida");
        }
        else
        {
            estoque = new Estoque();
            estoque.setDescricao(descricao);
            estoque.setPrecounit(precounit);
            estoque.setQuantidade(quantidade);
            estoque.setId(id);
            EstoqueControle ec = new EstoqueControle();
            estoque = ec.salvarComMsg(estoque);
            lista.add(estoque);
        }

        String retornoJsonString ="";
        retornoJsonString=this.gson.toJson(lista);
        PrintWriter out = response.getWriter();
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        out.print(retornoJsonString);
        out.flush();
    }
```

Esse Servlet recebe os dados para serem atualizados no banco de dados, podendo ser inclusão ou edição. Retorna um objeto do item do estoque com uma mensagem para ser visualizada no front-end. A parte final desse servlet não foi apresentada, pois é o método correspondente ao post.

Projeto Servlet/Front-end de Exemplo

FIM