

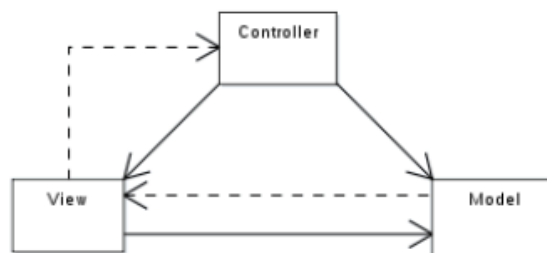
Conceitos básicos**Model, View & Controller – MVC****O que é**

Model-View-Controller (MVC) é um padrão de arquitetura de software;

- Com o aumento da complexidade das aplicações, torna-se fundamental a separação entre os dados e regras de negócio (Model) e a interface com o usuário (View);
- Alterações feitas na interface com o usuário não devem afetar a manipulação de dados; e
- Os dados podem ser reorganizados sem alterar a interface com o usuário.

O que faz?

O MVC separa as tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o usuário, introduzindo um componente entre os dois: o Controller.

**O MVC...**

É usado em padrões de projeto de software;

- Abrange mais da arquitetura de uma aplicação do que é típico para um padrão de projeto; e
- Define como os componentes da aplicação interagem, sendo considerado como um Design Pattern.

Como?

Quando se dividem os componentes em camadas, pode-se aplicar o MVC;

- A camada de negócios é o Model;
- A apresentação é a View;
- O controle é realizado pelo Controller;
- Não confundir MVC com separação de camadas;

- Camadas dizem como agrupar os componentes;
- O MVC diz como os componentes da aplicação interagem;
- O Controller despacha as solicitações ao Model; e
- A View observa o Model.

Model

- A representação "domínio" específica da informação em que a aplicação opera. Por exemplo: aluno, professor e turma fazem parte do domínio de um sistema acadêmico;
- É comum haver confusão pensando que Model é um outro nome para a camada de domínio;
- Lógica de domínio adiciona sentido a dados crus. Por exemplo: calcular se hoje é aniversário do usuário; e
- Muitas aplicações usam um mecanismo de armazenamento persistente para armazenar dados. O MVC não cita especificamente a camada para acesso aos dados, porque subentende-se que estes métodos estariam encapsulados pelo Model.

View

Visualiza o Model em uma forma específica para a interação, geralmente uma interface de usuário.

Controller

Processa e responde a eventos, geralmente ações do usuário, e pode invocar alterações no Model;

- Onde é feita a validação dos dados; e
- Onde os valores inseridos pelos usuários são filtrados.
- Controla o fluxo de execução da aplicação.

Aplicações Web

- A View é geralmente a página HTML;
- O Controller é o código que gera os dados dinâmicos para dentro do HTML; e
- O Model é representado pelo conteúdo de fato, geralmente armazenado em bancos de dados ou arquivos XML;

O Controle de Fluxo

Ainda que existam diferentes formas de MVC, o controle de fluxo, geralmente, funciona como segue:

- i) O usuário interage com a interface de alguma forma, por exemplo, o usuário aperta um botão;
- ii) O Controller manipula o evento da interface do usuário através de uma rotina pré-escrita;
- iii) O Controller acessa o Model, possivelmente atualizando-o de uma maneira apropriada, baseado na interação do usuário, por exemplo, atualizando os dados de cadastro do usuário;
- iv) Algumas implementações da View utilizam o Model para gerar uma interface apropriada, por exemplo, mostrando na tela os dados que foram alterados juntamente com uma confirmação;
- v) A View obtém seus próprios dados do Model;
- vi) O Model não toma conhecimento direto da View; e
- vii) A interface do usuário espera por próximas interações, que iniciarão o ciclo novamente.




Outra Interpretação

Existe uma forma de interpretação do padrão MVC, quando o sistema é muito pequeno, que é:

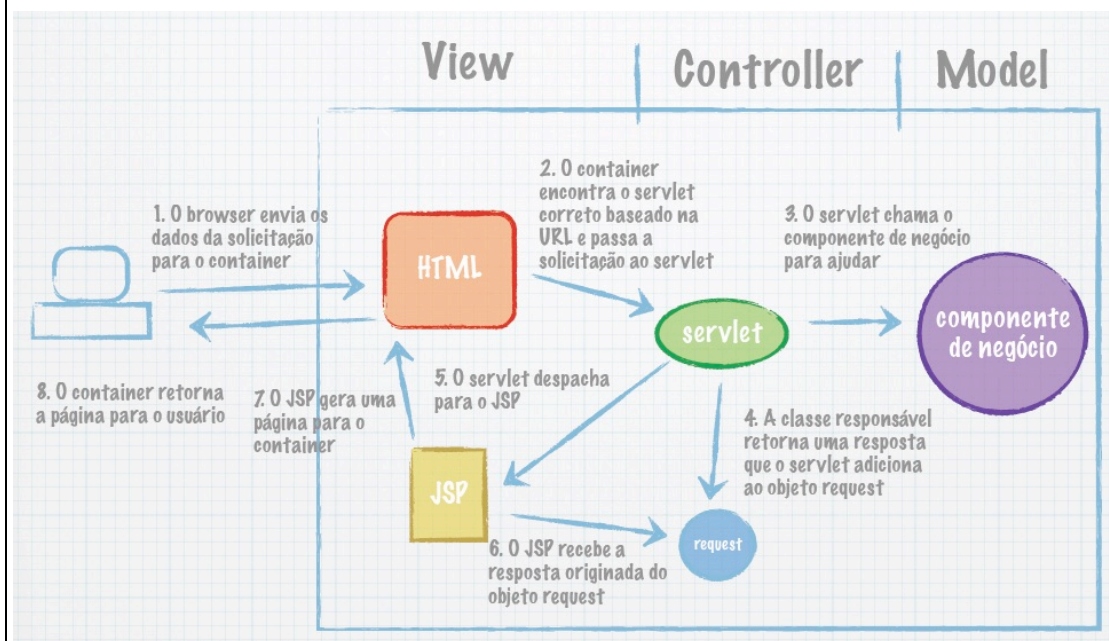
- i) View: Páginas em formato com resposta HTML (JSP, ASP, PHP etc.);
- ii) Controller: Controle da regra de negócio; e
- iii) Model: Representação dos elementos do seu domínio e interação com as ferramentas de persistência.

O MVC em Java Enterprise Edition para Web

Usando o MVC no mundo Servlet & JSP, qual o papel que cada um dos componentes desempenha:

	<u>Model</u>	<u>View</u>	<u>Controller</u>
<i>Non-Servlet Java Class</i>			
<i>JSP</i>			
<i>Servlet</i>			

Dinâmica de uma Interação (Realização de um Caso de Uso) na Web



A Essência do MVC

Separar a lógica de negócio da apresentação, colocando-se algo entre elas, para que a lógica de negócio possa agir sozinha, como uma classe Java reutilizável em outra View, sem precisar saber nada sobre ela;

- **Model:** Abriga a verdadeira lógica e o estado do modelo, conhecendo as regras para a obtenção e atualização do estado. É a única parte do programa que se comunica com o BD;
- **View:** Responsável pela apresentação, recebendo o estado do modelo do controlador. É também a parte que recebe os dados de entrada do usuário que volta ao controlador; e
- **Controller:** Retira da solicitação do usuário os dados de entrada e interpreta o que significam para o modelo. Obriga o modelo a se atualizar e disponibiliza o estado do novo modelo para a View (JSP).

Exemplo

O CRUD de Clientes que vimos fazendo até o momento já está seguindo o padrão MVC. Por exemplo, para se inserir um novo cliente o usuário clica no botão Novo Cliente e é aberto o formulário CriarCliente.jsp. O usuário preenche os dados e clica em Salvar. O servlet ManterClientesController é acionado, pega os parâmetros na requisição, reconhece que a solicitação recebida é para criar um cliente. Então instancia a classe de negócio Cliente, que cuida da inserção do cliente no banco e retorna para o servlet o cliente com o id preenchido. Então o servlet coloca um Transfer Object de

cliente na request e despacha a execução para o ListarCliente.jsp, que por sua vez apresenta dos dados para o usuário.

Este é um ciclo de execução do sistema, ou seja, uma interação, dentro do padrão MVC. Note que o Cliente.java não vai ao banco, mas sim o seu DAO. No model há outros design patterns envolvidos, como citado DAO, o TO e o Factory. Mas eles não são MVC. Para o MVC, todos fazem parte do Model.

Separando por papel MCV:

- **view:** CriarCliente.jsp, ListarClientes.jsp
- **model:** Cliente.java, ClienteDAO.java e ClienteTO.java
- **controller:** ManterClienteController.java

Outro exemplo, usando a sessão. Lembre-se que a diferença entre a requisição e a sessão é que a sessão é durável. Para melhorar a usabilidade do sistema, toda vez que o usuário fizer uma alteração a partir da tela listagem de clientes (ListarClientes.jsp), quando ele voltar para a tela de listagem a lista deve ser montada novamente, a partir da escolha que ele havia feito (buscar todos os clientes ou fazer a busca por palavra chave), levando em conta a alteração que ele fez:

- se alterou um cliente, os dados alterados devem aparecer na lista;
- se excluiu um cliente, ele não deve mais aparecer na lista;
- se visualizou um cliente, a lista deve continuar igual.

Para que isso seja feito, o ArrayList<Cliente> deve ser colocado na session, e não na request. O servlet ManterClienteController deve cuidar da manutenção da lista também, e não apenas da manutenção do banco. E o ListarClientesController deve estar preparado para jogar nulo na lista quando o usuário clica no menu, pois esta ação indica que ele quer começar novamente.

O código exemplo do CRUD está abaixo.

Exemplo View: index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %><!DOCTYPE
html>
<html lang="pt-br">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>cerveja.biz - Cadastros</title>

    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet">
</head>

<body>
    <!-- Barra superior com os menus de navegação -->
    <c:import url="Menu.jsp"/>
    <!-- Container Principal -->
    <div id="main" class="container">
        <h3 class="page-header"><u>Cadastros</u></h3>

        </div>
    <script src="js/jquery.min.js"></script>
```

```
<script src="js/bootstrap.min.js"></script>
</body>

</html>
```

Exemplo View: Menu.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#navbar" aria-expanded="false" aria-
controls="navbar">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="index.jsp">cerveja.biz</a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
            <ul class="nav navbar-nav navbar-right">
                <li><a
href="listar_clientes.do?acao=reiniciar">Clientes</a>
                </li>
                <li><a href="#">Marcas</a>
                </li>
                <li><a href="#">Estilos</a>
                </li>
                <li><a href="#">Países</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
```

Exemplo Controller: ManterClientesController.java

```
package controller;

import java.io.IOException;
import java.util.ArrayList;
import javax.servlet.RequestDispatcher;
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 javax.servlet.http.HttpSession;
import model.Cliente;
```

```

import service.ClienteService;

@WebServlet("/ManterCliente.do")
public class ManterClienteController extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String pAcao = request.getParameter("acao");
        String pId = request.getParameter("id");
        String pNome = request.getParameter("nome");
        String pFone = request.getParameter("fone");
        String pEmail = request.getParameter("email");
        int id = -1;
        try {
            id = Integer.parseInt(pId);
        } catch (NumberFormatException e) {

        }

        Cliente cliente = new Cliente();
        cliente.setId(id);
        cliente.setNome(pNome);
        cliente.setFone(pFone);
        cliente.setEmail(pEmail);
        ClienteService cs = new ClienteService();
        RequestDispatcher view = null;
        HttpSession session = request.getSession();

        if (pAcao.equals("Criar")) {
            cs.criar(cliente);
            ArrayList<Cliente> lista = new ArrayList<>();
            lista.add(cliente);
            session.setAttribute("lista", lista);
            view = request.getRequestDispatcher("ListarClientes.jsp");
        } else if (pAcao.equals("Excluir")) {
            cs.excluir(cliente.getId());
            ArrayList<Cliente> lista =
                (ArrayList<Cliente>)session.getAttribute("lista");
            lista.remove(busca(cliente, lista));
            session.setAttribute("lista", lista);
            view = request.getRequestDispatcher("ListarClientes.jsp");
        } else if (pAcao.equals("Alterar")) {
            cs.atualizar(cliente);
            ArrayList<Cliente> lista =
                (ArrayList<Cliente>)session.getAttribute("lista");
            int pos = busca(cliente, lista);
            lista.remove(pos);
            lista.add(pos, cliente);
            session.setAttribute("lista", lista);
            request.setAttribute("cliente", cliente);
            view = request.getRequestDispatcher("VisualizarCliente.jsp");
        }
    }
}

```

```

    } else if (pAcao.equals("Visualizar")) {
        cliente = cs.carregar(cliente.getId());
        request.setAttribute("cliente", cliente);
        view = request.getRequestDispatcher("VisualizarCliente.jsp");
    } else if (pAcao.equals("Editar")) {
        cliente = cs.carregar(cliente.getId());
        request.setAttribute("cliente", cliente);
        view = request.getRequestDispatcher("AlterarCliente.jsp");
    }

    view.forward(request, response);
}

public int busca(Cliente cliente, ArrayList<Cliente> lista) {
    Cliente to;
    for(int i = 0; i < lista.size(); i++){
        to = lista.get(i);
        if(to.getId() == cliente.getId()){
            return i;
        }
    }
    return -1;
}
}

```

Exemplo Controller: ListarClientesController.java

```

package controller;

import java.io.IOException;
import java.util.ArrayList;
import javax.servlet.RequestDispatcher;
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 javax.servlet.http.HttpSession;
import service.VendedorService;
import model.Cliente;

@WebServlet("/listar_clientes.do")
public class ListarClientesController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String chave = request.getParameter("data[search]");
        String acao = request.getParameter("acao");
        VendedorService vendedor = new VendedorService();
        ArrayList<Cliente> lista = null;
        HttpSession session = request.getSession();
        if (acao.equals("buscar")) {
            if (chave != null && chave.length() > 0) {

```

```

        lista = vendedor.listarClientes(chave);
    } else {
        lista = vendedor.listarClientes();
    }
    session.setAttribute("lista", lista);
} else if (acao.equals("reiniciar")) {
    session.setAttribute("lista", null);
}

RequestDispatcher dispatcher = request
    .getRequestDispatcher("ListarClientes.jsp");
dispatcher.forward(request, response);
}

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

```

Exemplo View: ListarClientes.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
    <!DOCTYPE html>
    <html lang="pt-br">

    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-
scale=1">
        <title>Buscar Clientes</title>

        <link href="css/bootstrap.min.css" rel="stylesheet">
        <link href="css/style.css" rel="stylesheet">

    </head>

    <body>
        <!-- Modal -->
        <div class="modal fade" id="delete-modal" tabindex="-1"
role="dialog" aria-labelledby="modalLabel">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <button type="button" class="close" data-
dismiss="modal" aria-label="Fechar"><span aria-hidden="true">&times;</span>
                        </button>
                        <h4 class="modal-title" id="modalLabel">Excluir
Cliente</h4>
                    </div>
                    <div class="modal-body">
                        Deseja realmente excluir este cliente?
                    </div>
                    <div class="modal-footer">
```



```

        <form action="ManterCliente.do" method="post">
            <input type="hidden" name="id"
id="id_excluir" />
            <button type="submit" class="btn btn-
primary" name="acao" value="Excluir">Sim</button>
            <button type="button" class="btn btn-
default" data-dismiss="modal">N&atilde;o</button>
        </form>
    </div>
</div>
</div>
<!-- /.modal -->
<!-- Barra superior com os menus de navega&ccedil;o -->
<c:import url="Menu.jsp"/>
<!-- Container Principal -->
<div id="main" class="container">
    <form action="listar_clientes.do" method="post">
        <div id="top" class="row">
            <div class="col-md-3">
                <h2>Clientes</h2>
            </div>

            <div class="col-md-6">
                <div class="input-group h2">
                    <input name="data[search]" class="form-
control" id="search" type="text" placeholder="Pesquisar Clientes (deixe
vazio para trazer todos)">
                    <span class="input-group-btn">
                        <button class="btn btn-primary" type="submit" name="acao"
value="buscar">
                            <span class="glyphicon glyphicon-search"></span>
                        </button>
                    </span>
                </div>
            </div>

            <div class="col-md-3">
                <a href="CriarCliente.jsp" class="btn btn-
primary pull-right h2">Novo Cliente</a>
            </div>
        </div>
    </form>
    <hr />
    <c:if test="${not empty lista}">
        <div id="list" class="row">

            <div class="table-responsive col-md-12">
                <table class="table table-striped" cellpadding="0"
cellpadding="0">
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Nome</th>
                            <th>Celular</th>
                            <th>E-Mail</th>
                            <th class="actions">A&ccedil;oes</th>

```

```

        </tr>
    </thead>
    <tbody>
    <c:forEach var="cliente" items="${lista }">
        <tr>
            <td>
                ${cliente.id }
            </td>
            <td>
                ${cliente.nome }
            </td>
            <td>
                ${cliente.fone }
            </td>
            <td>
                ${cliente.email }
            </td>
            <td class="actions">
                <a class="btn btn-success
btn-xs" href="ManterCliente.do?acao=Visualizar&id=${cliente.id
}">Visualizar</a>
                <a class="btn btn-warning
btn-xs" href="ManterCliente.do?acao=Editar&id=${cliente.id }">Editar</a>
                <button id="btn${cliente.id
}" type="button" class="btn btn-danger btn-xs" data-toggle="modal" data-
target="#delete-modal" data-cliente="${cliente.id }">Excluir</button>
            </td>
        </tr>
    </c:forEach>
    </tbody>
</table>

</div>
</div>
<!-- /#list -->

<div id="bottom" class="row">
    <div class="col-md-12">
        <!-- paginação ainda não foi implementada -->
        <ul class="pagination">
            <li class="disabled"><a>&lt; Anterior</a>
            </li>
            <li class="disabled"><a>1</a>
            </li>
            <li><a href="#">2</a>
            </li>
            <li><a href="#">3</a>
            </li>
            <li class="next"><a href="#" rel="next">Próximo
            &gt;</a>
            </li>
        </ul>
        <!-- /.pagination -->
    </div>
</div>
</c:if>
<!-- /#bottom -->

```

```

        </div>
        <!-- /#main -->
        <script src="js/jquery.min.js"></script>
        <script src="js/bootstrap.min.js"></script>
        <script type="text/javascript">
            $("#delete-modal").on('show.bs.modal', function(event) {
                var button = $(event.relatedTarget); //botao que
disparou a modal
                var recipient = button.data('cliente');
                $("#id_excluir").val(recipient);
            });
        </script>
    </body>

</html>

```

Exemplo Service: VendedorService.java

```

package service;

import java.util.ArrayList;
import model.Cliente;
import dao.ClienteDAO;

public class VendedorService {
    ClienteDAO dao;

    public VendedorService(){
        dao = new ClienteDAO();
    }
    public ArrayList<Cliente> listarClientes(){
        return dao.listarClientes();
    }
    public ArrayList<Cliente> listarClientes(String chave){
        return dao.listarClientes(chave);
    }
}

```

Exemplo Model: ClienteDAO.java

```

package dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import model.Cliente;

public class ClienteDAO {
    public int criar(Cliente cliente) {
        String sqlInsert = "INSERT INTO cliente(nome, fone, email) VALUES (?,
?, ?)";
    }
}

```

```

// usando o try with resources do Java 7, que fecha o que abriu
try (Connection conn = ConnectionFactory.obtemConexao();
    PreparedStatement stm = conn.prepareStatement(sqlInsert);) {
    stm.setString(1, cliente.getNome());
    stm.setString(2, cliente.getFone());
    stm.setString(3, cliente.getEmail());
    stm.execute();
    String sqlQuery = "SELECT LAST_INSERT_ID()";
    try (PreparedStatement stm2 = conn.prepareStatement(sqlQuery);
        ResultSet rs = stm2.executeQuery();) {
        if (rs.next()) {
            cliente.setId(rs.getInt(1));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
} catch (SQLException e) {
    e.printStackTrace();
}
return cliente.getId();
}

public void atualizar(Cliente cliente) {
    String sqlUpdate = "UPDATE cliente SET nome=?, fone=?, email=? WHERE
id=?";
    // usando o try with resources do Java 7, que fecha o que abriu
    try (Connection conn = ConnectionFactory.obtemConexao();
        PreparedStatement stm = conn.prepareStatement(sqlUpdate);) {
        stm.setString(1, cliente.getNome());
        stm.setString(2, cliente.getFone());
        stm.setString(3, cliente.getEmail());
        stm.setInt(4, cliente.getId());
        stm.execute();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void excluir(int id) {
    String sqlDelete = "DELETE FROM cliente WHERE id = ?";
    // usando o try with resources do Java 7, que fecha o que abriu
    try (Connection conn = ConnectionFactory.obtemConexao();
        PreparedStatement stm = conn.prepareStatement(sqlDelete);) {
        stm.setInt(1, id);
        stm.execute();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public Cliente carregar(int id) {
    Cliente cliente = new Cliente();
    cliente.setId(id);
    String sqlSelect = "SELECT nome, fone, email FROM cliente WHERE
cliente.id = ?";
    // usando o try with resources do Java 7, que fecha o que abriu
    try (Connection conn = ConnectionFactory.obtemConexao();
        PreparedStatement stm = conn.prepareStatement(sqlSelect);) {

```

```

        stm.setInt(1, cliente.getId());
        try (ResultSet rs = stm.executeQuery();) {
            if (rs.next()) {
                cliente.setNome(rs.getString("nome"));
                cliente.setFone(rs.getString("fone"));
                cliente.setEmail(rs.getString("email"));
            } else {
                cliente.setId(-1);
                cliente.setNome(null);
                cliente.setFone(null);
                cliente.setEmail(null);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    } catch (SQLException e1) {
        System.out.print(e1.getStackTrace());
    }
    return cliente;
}

public ArrayList<Cliente> listarClientes() {
    Cliente cliente;
    ArrayList<Cliente> lista = new ArrayList<>();
    String sqlSelect = "SELECT id, nome, fone, email FROM cliente";
    // usando o try with resources do Java 7, que fecha o que abriu
    try (Connection conn = ConnectionFactory.obtemConexao();
        PreparedStatement stm = conn.prepareStatement(sqlSelect);) {
        try (ResultSet rs = stm.executeQuery();) {
            while (rs.next()) {
                cliente = new Cliente();
                cliente.setId(rs.getInt("id"));
                cliente.setNome(rs.getString("nome"));
                cliente.setFone(rs.getString("fone"));
                cliente.setEmail(rs.getString("email"));
                lista.add(cliente);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    } catch (SQLException e1) {
        System.out.print(e1.getStackTrace());
    }
    return lista;
}

public ArrayList<Cliente> listarClientes(String chave) {
    Cliente cliente;
    ArrayList<Cliente> lista = new ArrayList<>();
    String sqlSelect = "SELECT id, nome, fone, email FROM cliente where upper(nome) like ?";
    // usando o try with resources do Java 7, que fecha o que abriu
    try (Connection conn = ConnectionFactory.obtemConexao();
        PreparedStatement stm = conn.prepareStatement(sqlSelect);) {
        stm.setString(1, "%" + chave.toUpperCase() + "%");
        try (ResultSet rs = stm.executeQuery();) {
            while (rs.next()) {
                cliente = new Cliente();

```

```

        cliente.setId(rs.getInt("id"));
        cliente.setNome(rs.getString("nome"));
        cliente.setFone(rs.getString("fone"));
        cliente.setEmail(rs.getString("email"));
        lista.add(cliente);
    }
} catch (SQLException e) {
    e.printStackTrace();
}
} catch (SQLException e1) {
    System.out.print(e1.getStackTrace());
}
return lista;
}
}

```

Exemplo Model: Cliente.java

```

package model;

import java.io.Serializable;

public class Cliente implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String nome;
    private String fone;
    private String email;

    public Cliente() {
    }

    public int getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getFone() {
        return fone;
    }

    public void setFone(String fone) {
        this.fone = fone;
    }
}

```

```

public String getEmail() {
    return email;
}

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

@Override
public String toString() {
    return "Cliente [id=" + id + ", nome=" + nome + ", fone=" + fone
        + ", email=" + email + "]";
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    Cliente other = (Cliente) obj;
    if (email == null) {
        if (other.email != null)
            return false;
    } else if (!email.equals(other.email))
        return false;
    if (fone == null) {
        if (other.fone != null)
            return false;
    } else if (!fone.equals(other.fone))
        return false;
    if (id != other.id)
        return false;
    if (nome == null) {
        if (other.nome != null)
            return false;
    } else if (!nome.equals(other.nome))
        return false;
    return true;
}
}

```

Exercícios

1. Revisão da Teoria

a. Importe o arquivo exemplo_clientes.war. Faça deployment no Tomcat e rode. Depois analise o código para entendê-lo.

2. Exercício para Nota (correção em aula)

a. Termine o CRUD de Pais, criando telas para alterar, excluir e listar todos os países. Use o modelo sugerido na WebDevAcademy, conforme bibliografia abaixo. Este modelo foi usado no CRUD de clientes.

Bibliografia

WebDevAcademy; **Tutorial: Criando Telas de CRUD com Bootstrap 3**; Disponível em <<http://webdevacademy.com.br/tutoriais/crud-com-bootstrap-3-parte1/>>. Acesso em 07/02/2017.

DEITEL, Harvey M.; DEITEL, Paul J; FURMANKIEWICZ, Edson. **Java: como programar**. 6. ed. atual. São Paulo: Pearson, 2005. xl, 1110 p. ISBN 8576050196 (Broch.)
nota: as outras edições não abordam JSP

BASHAM, Bryan; SIERRA, Kathy; BATES, Bert. **Use a cabeça!: Servlets & JSP**. 2. ed. Rio de Janeiro: Alta Books, 2008-2010. xxxii, 879 p. ISBN 9788576082941 (broch.)

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao processo unificado**. 3. ed. Porto Alegre: Bookman, 2007. 695 p. ISBN 9788560031528 (broch.)