

## Lab 4 - CDI - Injeção de Dependência e Contextos

Neste laboratório iremos dar vida a interface construída anteriormente, fazendo com que aplicação banco esteja funcionando 100%, implementando todas as regras de interface e utilizando as regras de negocio já desenvolvidas no módulo *Banco-Service*.

### Exercícios

**Exercício 1:** Adicionar CDI a aplicação.

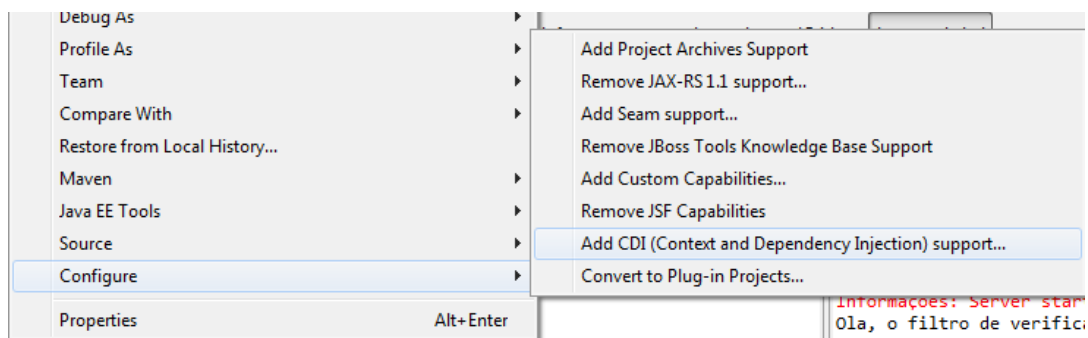
**Exercício 2:** Adicionar funcionalidades de Login e Logout na aplicação.

**Exercício 2:** Criando CRUD de clientes.

**Exercício 3:** Concluir a aplicação desenvolvendo todas as regras de interface para que haja a comunicação com as regras de negocio existente.

### Exercício 1 -Adicionar funcionalidades de Login e Logout na aplicação

1. Para adicionar o CDI a aplicação primeiro deve-se fazer com que a aplicação seja compatível com o CDI. Clique com o botão direito em cima do projeto **Banco-FrameWork > Configure > Add CDI (Context and Dependency Injection) support**. Caso essa opção não esteja disponível, será necessário instalar o Jboss Tools no seu eclipse através do marketplace.



2. Perceba que será criado o arquivo **beans.xml** dentro de WEB-INF.



3. Agora adicione a dependência do Jboss Weld e JEE 6 no **pom.xml** do pacote Banco-Service

```
<dependency>
  <groupId>org.jboss.weld.servlet</groupId>
  <artifactId>weld-servlet</artifactId>
  <version>1.1.10.Final</version>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.0.1</version>
  <scope>provided</scope>
</dependency>
</dependency>
```

```
<groupId>javax</groupId>
<artifactId>jakartaee-api</artifactId>
<version>6.0</version>
</dependency>
```

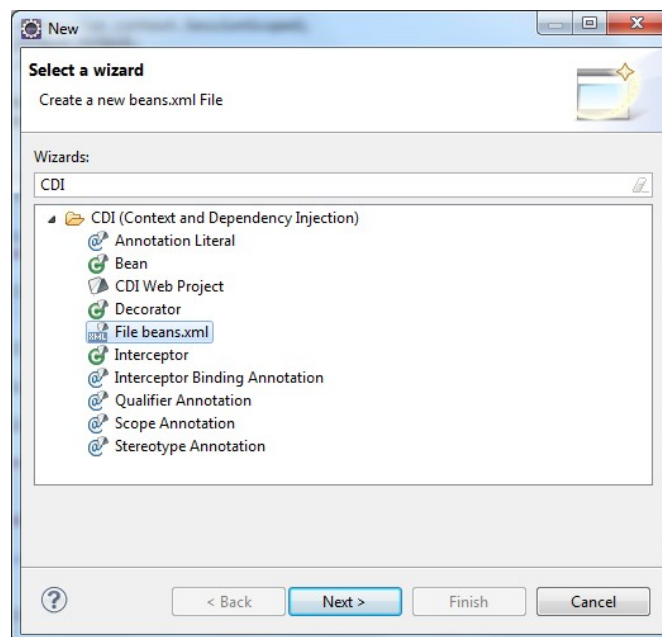
JBoss Weld é responsável pela referência de implementação do CDI que é um padrão Java de injeção de dependências e gerenciamento de ciclo de vida contextual.

- Agora modifique o **web.xml** da aplicação *Banco-FrameWork* adicionando as seguintes configurações.

```
<resource-env-ref>
  <resource-env-ref-name>BeanManager</resource-env-ref-name>
  <resource-env-ref-type>javax.enterprise.inject.spi.BeanManager</resource-env-ref-type>
</resource-env-ref>

<listener>
  <listener-class>org.jboss.weld.environment.servlet.Listener</listener-class>
</listener>
```

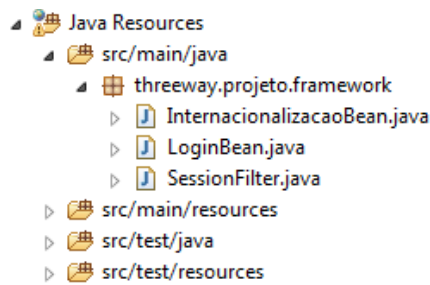
- Adicione o arquivo **META-INF/beans.xml** nos projetos *Banco-Modelo* e *Banco-Service*. Clique com o **botão direito no projeto > New > Other > File beans.xml**.



- Agora todas as classes que serão injetadas terão que implementar a interface *Serializable*, tanto as classes de *Banco-Service* quanto as de *Banco-Modelo*.

## Exercício 2 -Adicionar funcionalidades de Login e Logout na aplicação

- Copie a classe **SessionFilter.java** desenvolvida no *Banco-WebView* e cole dentro do diretório da aplicação *Banco-FrameWork*.



2. Modifique a classe **SessionFilter.java** de acordo com o código abaixo:

```
@WebFilter(filterName="/SessionFilter", urlPatterns={"/manterCliente.jsf", "/operacoesBancarias.jsf" })
public class SessionFilter implements Filter {

    public SessionFilter() {
        // TODO Auto-generated constructor stub
    }

    public void destroy() {
        // TODO Auto-generated method stub
    }

    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {

        System.out.println("Ola, o filtro de verificar sessão esta ativo!");

        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;

        String usuarioSession = (String) req.getSession().getAttribute("usuario");

        if (usuarioSession == null || usuarioSession.isEmpty()) {

            resp.sendRedirect("login.jsf");

        } else {

            chain.doFilter(request, response);

        }

    }

    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }
}
```

Lembre-se de colocar a urlPatterns de acordo com as páginas que terão controle de acesso, ou seja, páginas que passarão pelo filtro.

3. Crie a classe **LoginBean.java** e modifique de acordo com o código abaixo:

```
@Named
@RequestScoped
public class LoginBean implements Serializable {

    private static final long serialVersionUID = 1L;

    @Inject
    private FacesContext facesContext;

    @Inject
    private AgenciaService agenciaService;

    private String login;

    private String senha;

    @PostConstruct
    public void inicializaAgencia() {

        this.agenciaSistema = agenciaService.agenciaSistema();
    }
}
```

```

}

public String realizarLogin() {
    if (this.getLogin().equals("admin") && this.getSenha().equals("admin")) {
        HttpSession session = (HttpSession) facesContext.getExternalContext().getSession(false);
        session.setAttribute("usuario", this.getLogin());
        return "manterCliente";
    }

    facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Erro", "Login ou Senha Incorretos!"));
    return "login";
}

public String realizarLogout() {
    HttpSession session = (HttpSession) facesContext.getExternalContext().getSession(false);
    session.setAttribute("usuario", null);
    return "login";
}

@Produces
@RequestScoped
public FacesContext getFacesContext() {
    return FacesContext.getCurrentInstance ();
}

//Gere os métodos getters e setters para as variáveis criadas.
}

```

Utilizar a anotação `@RequestScoped` indica que a instância do objeto será criada uma vez durante cada requisição.

A anotação `@PostConstruct` faz com que o método seja executado logo após a instanciação do bean, nesse caso o método que inicializa a lista de clientes será executado.

Utilizando CDI e Injetando *FacesContext* da aplicação depois de ter produzido. `@Produces`

- Modifique a página **login.xhtml** de acordo com o código abaixo para que possa acessar seu Controller CDI e acessar suas variáveis e métodos.

```

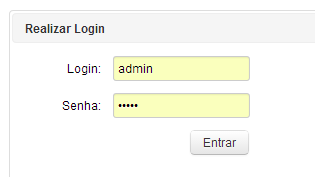
<h:form id="formLogin">
    <p:growl id="growl" autoUpdate="true" showDetail="true" />
    <p:panel styleClass="painelLogin" header="#{msg['realizarLogin']}">
        <h:panelGrid styleClass="textRight" columns="2" cellpadding="5">
            <h:outputLabel value="#{msg['login']}:" />
            <p:inputText requiredMessage="Campo Login Obrigatório" required="true" value="#{loginBean.login}"/>
            <h:outputLabel value="#{msg['senha']}:" />
            <p:password requiredMessage="Campo Senha Obrigatório" required="true" value="#{loginBean.senha}"/>
            <p:spacer width="70" height="0"/>
            <p:commandButton action="#{loginBean.realizarLogin()}" update="growl" value="#{msg['entrar']}"/>
        </h:panelGrid>
    </p:panel>
</h:form>

```

5. Agora modifique a página **topoPagina.xhtml** de modo que a opção *realizaLogout()* seja chamada.

```
<h:form id="formInternacionalizacao" style="text-align: right !important">
    <h:outputLabel value="#{msg['saudacao']} #{loginBean.login}" />
    <p:spacer width="15" />
    <p:commandLink styleClass="internacionalizacao" update="@all"
        action="#{internacionalizacaoBean.portugueseLocale()}">
        <p:graphicImage value="resources/img/brasil.png" />
    </p:commandLink>
    <p:spacer width="5" />
    <p:commandLink styleClass="internacionalizacao" update="@all"
        action="#{internacionalizacaoBean.englishLocale()}">
        <p:graphicImage value="resources/img/estados_unidos.png" />
    </p:commandLink>
    <p:spacer width="15" />
    <p:commandLink update="@all" action="#{loginBean.realizarLogout()}" value="#{msg['botao.sair']}" />
</h:form>
```

6. Execute a aplicação e antes de logar tente acessar as páginas declaradas no urlPattern do **SessionFilter.java** (*manterCliente* e *operacoesBancarias*) .



Realizar Login

Login:

Senha:

## Exercício 3 -Criando CDRU de clientes

1. Crie a classe **ManterClienteBean.java** e modifique de acordo com o código abaixo:

```
@Named
@SessionScoped
public class ManterClienteBean implements Serializable {

    /** Atributo serialVersionUID. */
    private static final long serialVersionUID = -8735715935775329780L;

    @Inject
    private FacesContext facesContext;

    @Inject
    private Cliente cliente;

    @Inject
    private ClienteService service;

    private Collection<Cliente> clientesCadastrados;
```

```
//Gere os métodos getters e setters das variáveis criadas.
```

```
}
```

Utilizar a anotação `@SessionScoped` indica que o bean ficará ativo enquanto durar a sessão e durante várias requisições http.

2. Agora vamos modificar a página **manterCliente.xhtml** de modo que essa acesse os dados do controller.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:p="http://primefaces.org/ui"
xmlns:ui="http://java.sun.com/jsf/facelets"
template="templates/template.xhtml">

<ui:define name="conteudo">

<h:form id="manterClientesForm">

    <p:growl id="growl" autoUpdate="true" showDetail="true" />

    <p:panel id="panelCadastro" header="">

    <p:messages id="messages" showDetail="true" autoUpdate="true" closable="true" />

        <p:panelGrid style="width:100%;">

            <p:row>

                <p:column styleClass="textRight">
                    <h:outputLabel value="#{msg['form.cliente.nome']}:" />
                </p:column>

                <p:column>
                    <p:inputText style="width:100%;" value="#{manterClienteBean.cliente.nome}"
                        size="45" />
                </p:column>

                <p:column styleClass="textRight">
                    <h:outputLabel value="#{msg['form.cliente.telefone']}:" />
                </p:column>

                <p:column>
                    <p:inputMask style="width:100%;"
                        value="#{manterClienteBean.cliente.telefone}" mask="(99)9999-9999" />
                </p:column>
            </p:row>

            <p:row>

                <p:column styleClass="textRight">
                    <h:outputLabel value="#{msg['form.cliente.endereco']}:" />
                </p:column>

                <p:column colspan="3">
                    <p:inputText value="#{manterClienteBean.cliente.endereco}" style="width:100%;" />
                </p:column>
            </p:row>

            <p:row>

                <p:column styleClass="textRight">
                    <h:outputLabel value="#{msg['form.cliente.rg']}:" />
                </p:column>

                <p:column>
                    <p:inputText value="#{manterClienteBean.cliente.rg}" style="width:100%;" />
                </p:column>

                <p:column styleClass="textRight">
                    <h:outputLabel value="#{msg['form.cliente.cpf']}:" />
                </p:column>

                <p:column>
                    <p:inputMask value="#{manterClienteBean.cliente.cpf}" style="width:100%;"
                        mask="999.999.999-99" />
                </p:column>
            </p:row>
        </p:panelGrid>

    </h:form>

</ui:define>
```

```

<h:panelGrid columns="3" cellpadding="10">

    <p:commandButton actionListener="#{manterClienteBean.limpar()}"
        update="panelCadastro :formNavegacao" value="#{msg['btn.cliente.limpar']}" />

    <p:commandButton update="manterClientesForm manterClientesForm:growl :formNavegacao"
        actionListener="#{manterClienteBean.salvar()}" value="#{msg['btn.cliente.salvar']}" />

    <p:commandButton rendered="#{manterClienteBean.cliente.identificador != null}"
        update="manterClientesForm :formNavegacao" actionListener="#{manterClienteBean.excluir()}"
        value="#{msg['btn.cliente.excluir']}" />

</h:panelGrid>

</p:panel>

<p:panel>

    <p:dataTable emptyMessage="#{msg['tabela.cliente.empty']}" var="cliente"
        value="#{manterClienteBean.clientesCadastrados}" styleClass="fonte">

        <f:facet name="header">
            #{msg['topoTabelaCliente']}
        </f:facet>

        <p:column headerText="#">
            <h:outputLabel value="#{cliente.identificador}" />
        </p:column>

        <p:column headerText="#{msg['form.cliente.nome']}">
            <h:outputLabel value="#{cliente.nome}" />
        </p:column>

        <p:column headerText="#{msg['form.cliente.endereco']}">
            <h:outputLabel value="#{cliente.endereco}" />
        </p:column>

        <p:column headerText="#{msg['form.cliente.telefone']}">
            <h:outputLabel value="#{cliente.telefone}" />
        </p:column>

        <p:column headerText="#{msg['form.cliente.rg']}">
            <h:outputLabel value="#{cliente.rg}" />
        </p:column>

        <p:column headerText="#{msg['form.cliente.cpf']}">
            <h:outputLabel value="#{cliente.cpf}" />
        </p:column>

        <p:column headerText="#{msg['tabela.acoes']}">
            [<p:commandLink value="Carregar"
                update="manterClientesForm:panelCadastro :formNavegacao">
                <f:setPropertyActionListener target="#{manterClienteBean.cliente}"
                    value="#{cliente}" />
            </p:commandLink>]
        </p:column>

    </p:dataTable>

</p:panel>

</h:form>

</ui:define>

</ui:composition>

```

3. Veja que estou utilizando a ação de limpar, salvar e excluir. Estas ainda não foram criadas no **ManterClienteBean.java**. Implemente esses métodos **públicos** . Lembre-se de inicializar a lista de clientes cadastrados para poder ver os dados na dataTable.

```
@PostConstruct
private void inicializaLista() {
    this.clientesCadastrados = this.getService().listarTodosClientes();
}

public void limpar() {
    this.cliente = new Cliente();
}

public void salvar() {
    try {
        if (this.cliente.getIdentificador() == null) {
            this.service.salvar(this.cliente);
        } else {
            this.service.atualizar(this.cliente);
        }
        this.inicializaLista();
        this.limpar();
        facesContext.addMessage(null, new FacesMessage
            (FacesMessage.SEVERITY_INFO, "Sucesso", "Cliente salvo com sucesso"));
    } catch (CamposObrigatoriosException e) {
        facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Erro", e.getMessage()));
    }
}

public void excluir() {
    this.service.excluir(this.cliente);
    this.limpar();
    this.inicializaLista();
}
```


4. Veja que não estou implementado regras de negócio, e sim usando as que já existem no projeto *Banco-Service*.



5. Execute a aplicação e veja o resultado da implementação.

Projeto Banco 3Way NetWorks

Bem vindo, Sr.(a) admin



Sair

Banco: Banco Java Brasil    Agência: 3way NetWorks!

Manter Cliente

Operações Bancárias

Nome:

Telefone:

Endereço:

Registro Geral:

CPF:

Limpar

Salvar

Clientes Cadastrados

#	Nome	Endereço	Telefone	Registro Geral	CPF	Ações
1	wilker teste				111.111.111-11	<a href="#">[Carregar]</a>
2	teste atualiza		(12)1516-5116		222.222.222-22	<a href="#">[Carregar]</a>
4	ewq				646.546.546-46	<a href="#">[Carregar]</a>
5	teste2				164.646.464-64	<a href="#">[Carregar]</a>

**Exercício 4** -Concluir a aplicação desenvolvendo todas as regras de interface para que haja a comunicação com as regras de negocio existente.

- Vamos criar as regras de navegação entre **manterCliente.xhtml** e **operacoesBancarias.xhtml**. Modifique a página **template.xhtml** de modo que essa acesse a agencia e banco do sistema e adicione a seguinte regra de navegação:
  - Somente quando um cliente estiver selecionado será habilitado o botão de redirecionar para **operacoesBancarias.xhtml**.

```

<h:panelGrid width="100%" cellpadding="10">

<h:panelGrid columns="5" cellpadding="5">
    <h:outputLabel value="#{msg['banco']}: " />
    <h:outputLabel value="#{loginBean.agenciaSistema.banco.nome}" />
    <p:spacer width="15"/>
    <h:outputLabel value="#{msg['agencia']}: " />
    <h:outputLabel value="#{loginBean.agenciaSistema.nome}" />
</h:panelGrid>
<p:separator />
<p:panel>

    <h:panelGrid style="width:100%;" cellpadding="5">
        <h:form id="formNavegacao">

            <h:panelGrid columns="2">
                <p:commandButton icon="ui-icon-person" action="manterCliente"
                    value="#{msg['titulo.aba.cliente']}" />

                <p:commandButton icon="ui-icon-calculator"
                    disabled="#{manterClienteBean.cliente.identificador == null}"
                    action="#{operacoesBancariasBean.selecionaCliente(manterClienteBean.cliente)}"
                    value="#{msg['titulo.aba.operacoesBancarias']}" />
            </h:panelGrid>
        </h:form>
        <ui:insert name="conteudo" />
    </h:panelGrid>
</p:panel>
</h:panelGrid>

```

2. Crie a classe **OperacoesBancariasBean.java** e modifique criando os seguintes atributos.

```
@Named
@SessionScoped
public class OperacoesBancariasBean implements Serializable {

    private static final long serialVersionUID = -1259503110150417090L;

    @Inject
    private FacesContext facesContext;

    @Inject
    private AgenciaService agenciaService;

    @Inject
    private TransacaoService transacaoService;

    @Inject
    private ManterClienteBean clienteBean;

    @Inject
    private ContaService contaService;

    private Conta contaCliente;

    private Cliente clienteSelecionado;

    private boolean flagDialogAbrirConta;

    private boolean flagDialogDeposito;

    private boolean flagDialogSaque;

    private boolean flagDialogTransferencia;

    private Double valorDeposito;

    private Double valorSaque;

    private Double valorTransferencia;

    private int numeroContaTransferencia;

    private Collection<Transacao> transacoesConta;

    //Gere os métodos getters e setters das variáveis criadas.
}
```

3. Agora modifique a tela **operacoesBancarias.xhtml** de modo que essa acesse os dados do controller.

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:p="http://primefaces.org/ui"
xmlns:ui="http://java.sun.com/jsf/facelets"
template="templates/template.xhtml">

    <ui:define name="conteudo">
        <h:form id="opBancariasForm">
            <p:panel header="">

                <p:messages id="messages" showDetail="true" autoUpdate="true" closable="true" />

                <h:panelGrid columns="2" cellpadding="5">

                    <h:outputLabel value="Cliente:" />
                    <h:outputLabel value="#{operacoesBancariasBean.clienteSelecionado.nome}" />

                </h:panelGrid>

                <h:panelGrid columns="8" cellpadding="5">

                    <h:outputLabel value="Nº Conta:" />
                    <h:outputLabel rendered="#{operacoesBancariasBean.contaCliente.identificador != null}"
                        value="#{operacoesBancariasBean.contaCliente.numero}" />

                    <p:spacer width="10" />

                    <h:outputLabel value="Data Abertura:" />

                </h:panelGrid>

            </p:panel>
        </h:form>
    </ui:define>
</ui:composition>
```

```

<h:outputLabel rendered="#{operacoesBancariasBean.contaCliente.identificador != null}"
    value="#{operacoesBancariasBean.contaCliente.dataAbertura}" >
    <f:convertDateTime pattern="dd/MM/yyyy"/>
</h:outputLabel>

<p:spacer width="10" />

<h:outputLabel value="Saldo:" />
<h:outputLabel rendered="#{operacoesBancariasBean.contaCliente.identificador != null}"
    value="#{operacoesBancariasBean.contaCliente.saldo}" >
    <f:convertNumber currencySymbol="R$" type="currency" groupingUsed="true" />
</h:outputLabel>

</h:panelGrid>

<p:spacer height="20" />
<p:separator />
<p:spacer height="20" />

<h:panelGrid columns="4" cellpadding="2">

    <p:commandButton value="Abrir Conta" update=":idDialogAbrirConta"
        disabled="#{operacoesBancariasBean.contaCliente.identificador != null}"
        oncomplete=PF('dialogAbrirConta ').show();">
        <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogAbrirConta}"
            value="true" />
    </p:commandButton>

    <p:commandButton value="Deposito" update=":idDialogDeposito"
        disabled="#{operacoesBancariasBean.contaCliente.identificador == null}"
        oncomplete=PF('dialogDeposito ').show();">
        <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogDeposito}"
            value="true" />
    </p:commandButton>

    <p:commandButton value="Saque" update=":idDialogSaque"
        disabled="#{operacoesBancariasBean.contaCliente.identificador == null}"
        oncomplete=PF('dialogSaque ').show();">
        <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogSaque}" value="true" />
    </p:commandButton>

    <p:commandButton value="Transferencia"
        disabled="#{operacoesBancariasBean.contaCliente.identificador == null}"
        onclick=PF('dialogTransf').show();"/>

</h:panelGrid>

</p:panel>
</p:panel>

<p:dataTable emptyMessage="#{msg['tabela.transacao.empty']}" var="transacao"
    value="#{operacoesBancariasBean.transacoesConta}" styleClass="fonte"
    rowStyleClass="#{operacoesBancariasBean.determinaCorLinhaTabela(transacao)}">

    <f:facet name="header">
        #{msg['topoTabelaTransacao']}
    </f:facet>

    <p:column headerText="#">
        <h:outputLabel value="#{transacao.identificador}" />
    </p:column>

    <p:column headerText="#{msg['form.transacao.tipoTransacao']}">
        <h:outputLabel value="#{transacao.tipoTransacao}" />
    </p:column>

    <p:column headerText="#{msg['form.transacao.contaCredito']}">
        <h:outputLabel
            value="#{transacao.contaCredito.titular.nome}/#{transacao.contaCredito.numero}" />
    </p:column>

    <p:column headerText="#{msg['form.transacao.contaDebito']}">
        <h:outputLabel
            value="#{transacao.contaDebito.titular.nome}/#{transacao.contaDebito.numero}" />
    </p:column>

    <p:column headerText="#{msg['data']}">
        <h:outputLabel value="#{transacao.data}" >
            <f:convertDateTime pattern="dd/MM/yyyy"/>
        </h:outputLabel>
    </p:column>

    <p:column headerText="#{msg['valor']}">
        <h:outputLabel value="#{transacao.valor}" >

```

```

<f:convertNumber currencySymbol="R$" type="currency"
groupingUsed="true" />
</h:outputLabel>
</p:column>
</p:dataTable>
</p:panel>
</h:form>

<ui:include src="dialogAbrirConta.xhtml" />
<ui:include src="dialogDeposito.xhtml" />
<ui:include src="dialogSaque.xhtml" />
<ui:include src="dialogTransferencia.xhtml" />
</ui:define>

</ui:composition>

```

4. Crie as funções necessárias para atender as requisições da página **operacoesBancarias.xhtml**. Modifique a classe **OperacoesBancariasBean.java** adicionando os seguintes métodos:

```

public String determinaCorLinhaTabela(Transacao transacao) {
    if (transacao.getTipoTransacao() == EnumTipoTransacao.DEPOSITO) {
        return "deposito";
    } else if (transacao.getTipoTransacao() == EnumTipoTransacao.SAQUE) {
        return "saque";
    } else {
        return "transferencia";
    }
}

public void inicializaOperacoesConta() {
    this.valorDeposito = 0.0;
    this.valorTransferencia = 0.0;
    this.valorSaque = 0.0;
    this.contaCliente = this.contaService.buscaContaPorCliente(clienteSelecionado);

    if (this.contaCliente != null && this.contaCliente.getIdentificador() != null) {
        this.setTransacoesConta(this.transacaoService.listarTransacoesPorConta(contaCliente.getIdentificador()));
    } else {
        this.setContaCliente(new Conta());
        this.getContaCliente().setTitular(clienteSelecionado);
        this.getContaCliente().setDataAbertura(new Date());
        this.getContaCliente().setTipoConta(EnumTipoConta.CONTA_PESSOAL);
        this.getContaCliente().setAgencia(this.agenciaService.agenciaSistema());
    }
}

public String selecionaCliente(Cliente cliente) {
    this.setClienteSelecionado(cliente);
    this.inicializaOperacoesConta();
    return "operacoesBancarias";
}

```

5. Execute a aplicação testando a regra de navegação.

Banco: Banco Java Brasil Agência: 3way NetWorks!

---

---

Cliente: teste2  
 N° Conta:      Data Abertura:      Saldo:

---

---

Historico de Transações					
#	Tipo Transação	Titular/Conta Crédito	Titular/Conta Débito	Data	Valor
Nenhuma Transação Cadastrada					

## 6. Agora vamos adicionar as regras de tela na página (Dialog) **dialogAbrirConta.xhtml**.

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:p="http://primefaces.org/ui"
xmlns:ui="http://java.sun.com/jsf/facelets">

<p:dialog id="idDialogAbrirConta" appendTo="@{body}" visible="#{operacoesBancariasBean.flagDialogAbrirConta}"
header="Abertura de Conta" widgetVar="dialogAbrirConta" modal="true" closable="false" showEffect="explode"
hideEffect="bounce" resizable="false">

  <h:form id="formDialogAbrirConta">

    <p:messages id="messagesDialog1" showDetail="true" autoUpdate="true" closable="true" />

    <h:panelGrid columns="2" cellpadding="5">

      <h:outputLabel value="Data Abertura:" />

      <p:inputText disabled="true" value="#{operacoesBancariasBean.contaCliente.dataAbertura}"
        <f:convertDateTime pattern="dd/MM/yyyy"/>
      </p:inputText>

      <h:outputLabel value="Numero da Conta:" />

      <p:inputText value="#{operacoesBancariasBean.contaCliente.numero}" />

      <h:outputLabel value="Saldo Inicial:" />

      <p:inputText value="#{operacoesBancariasBean.contaCliente.saldo}" />

    </h:panelGrid>

    <p:spacer height="5"/>

    <p:separator />

    <h:panelGrid columns="2" cellpadding="10">

      <p:commandButton value="Abrir Conta" update=":idDialogAbrirConta :opBancariasForm"
        actionListener="#{operacoesBancariasBean.abrirConta()}"
        <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogAbrirConta}"
          value="false" />
      </p:commandButton>

      <p:commandButton value="Cancelar" update=":idDialogAbrirConta :opBancariasForm">

        <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogAbrirConta}"
          value="false" />
      </p:commandButton>

    </h:panelGrid>

  </h:form>

</p:dialog>

</html>
```

7. Modifique a classe **OperacoesBancariasBean.java** adicionando o método utilizado na página **dialogAbrirConta.xhtml**.

```
public void abrirConta() {  
    try {  
        this.contaService.salvar(this.getContaCliente());  
        this.inicializaOperacoesConta();  
        this.flagDialogAbrirConta = Boolean.FALSE;  
        facesContext.addMessage(null, new FacesMessage  
            (FacesMessage.SEVERITY_INFO, "Sucesso", "Abertura de conta concluída!"));  
    } catch (CamposObrigatoriosException e) {  
        facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "ERRO", e.getMessage()));  
    }  
}
```

8. Agora vamos adicionar as regras de tela na página (Dialog) **dialogDeposito.xhtml**.

```
<html xmlns="http://www.w3.org/1999/xhtml"  
xmlns:h="http://java.sun.com/jsf/html"  
xmlns:f="http://java.sun.com/jsf/core"  
xmlns:p="http://primefaces.org/ui"  
xmlns:ui="http://java.sun.com/jsf/facelets">  
  
    <p:dialog id="idDialogDeposito" appendTo="@{body}" visible="#{operacoesBancariasBean.flagDialogDeposito}" header="Deposito"  
    widgetVar="dialogDeposito" modal="true" closable="false" showEffect="explode" hideEffect="bounce" resizable="false">  
  
        <h:form id="formDialogDeposito">  
  
            <h:panelGrid columns="2" cellpadding="5">  
  
                <h:outputLabel value="#{msg['valor']}:" />  
  
                <p:inputText value="#{operacoesBancariasBean.valorDeposito}" />  
  
            </h:panelGrid>  
  
            <p:spacer height="5"/>  
            <p:separator />  
  
            <h:panelGrid columns="2" cellpadding="10">  
  
                <p:commandButton value="Ok" update=":idDialogDeposito :opBancariasForm" actionListener="#{operacoesBancariasBean.efetu-  
arDeposito()}">  
  
                    <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogDeposito}" value="false" />  
  
                </p:commandButton>  
  
                <p:commandButton value="Cancelar" update=":idDialogDeposito :opBancariasForm">  
  
                    <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogDeposito}" value="false" />  
  
                </p:commandButton>  
  
            </h:panelGrid>  
        </h:form>  
    </p:dialog>  
</html>
```

9. Modifique a classe **OperacoesBancariasBean.java** adicionando o método utilizado na página **dialogDeposito.xhtml**.

```
public void efetuarDeposito() {  
    if (valorDeposito != null && valorDeposito > 0) {  
        this.contaService.depositar(contaCliente, valorDeposito);  
        this.inicializaOperacoesConta();  
        facesContext.addMessage(null, new FacesMessage  
            (FacesMessage.SEVERITY_INFO, "Sucesso", "Deposito Realizado com sucesso!"));  
    } else {  
        facesContext.addMessage(null, new FacesMessage  
            (FacesMessage.SEVERITY_ERROR, "Erro.", "Valor válido não informado!"));  
    }  
}
```

10. Agora vamos adicionar as regras de tela na página (Dialog) **dialogSaque.xhtml**.

```
<html xmlns="http://www.w3.org/1999/xhtml"  
xmlns:h="http://java.sun.com/jsf/html"  
xmlns:f="http://java.sun.com/jsf/core"  
xmlns:p="http://primefaces.org/ui"  
xmlns:ui="http://java.sun.com/jsf/facelets">  
  
<p:dialog id="idDialogSaque" appendTo="@{body}" visible="#{operacoesBancariasBean.flagDialogSaque}"  
header="Saque" widgetVar="dialogSaque" modal="true" showEffect="explode" hideEffect="bounce"  
closable="false" resizable="false">  
  
<h:form id="formDialogSaque">  
  
    <h:panelGrid columns="2" cellpadding="5">  
  
        <h:outputLabel value="#{msg['valor']}" />  
  
        <p:inputText value="#{operacoesBancariasBean.valorSaque}" />  
  
    </h:panelGrid>  
  
    <p:spacer height="5"/>  
    <p:separator />  
  
    <h:panelGrid columns="2" cellpadding="10">  
  
        <p:commandButton value="Ok" update=":idDialogSaque :opBancariasForm"  
            actionListener="#{operacoesBancariasBean.efetuarSaque()}">  
  
            <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogSaque}" value="false" />  
  
        </p:commandButton>  
  
        <p:commandButton value="Cancelar" update=":idDialogSaque :opBancariasForm">  
  
            <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogSaque}" value="false" />  
  
        </p:commandButton>  
    </h:panelGrid>  
  
</h:form>  
</p:dialog>  
</html>
```

11. Modifique a classe **OperacoesBancariasBean.java** adicionando o método utilizado na página **dialogSaque.xhtml**.

```
public void efetuarSaque() {  
    if (valorSaque != null && valorSaque > 0) {  
        try {  
            this.contaService.sacar(contaCliente, valorSaque);  
            this.inicializaOperacoesConta();  
            facesContext.addMessage(null, new FacesMessage(  
                FacesMessage.SEVERITY_INFO, "Sucesso", "Saque Realizado com sucesso!"));  
        } catch (SaldoInsuficienteException e) {  
            facesContext.addMessage(null, new FacesMessage(FacesMessage.SEVERITY_ERROR, "Erro.", e.getMessage()));  
        }  
    } else {  
        facesContext.addMessage(null, new FacesMessage(  
            FacesMessage.SEVERITY_ERROR, "Erro.", "Valor válido não informado!"));  
    }  
}
```

12. Agora vamos adicionar as regras de tela na página (Dialog) **dialogTransferencia.xhtml**.

```
<html xmlns="http://www.w3.org/1999/xhtml"  
xmlns:h="http://java.sun.com/jsf/html"  
xmlns:f="http://java.sun.com/jsf/core"  
xmlns:p="http://primefaces.org/ui"  
xmlns:ui="http://java.sun.com/jsf/facelets">  
  
<p:dialog id="idDialogTransf" appendTo="@body" visible="#{operacoesBancariasBean.flagDialogTransferencia}" header="Transferência"  
widgetVar="dialogTransf" modal="true" showEffect="explode" hideEffect="bounce" closable="false" resizable="false">  
  
<h:form id="formDialogTransf">  
<p:panel header="Dados da Conta">  
    <h:panelGrid columns="4" cellpadding="5">  
        <h:outputLabel value="Nº:" />  
        <p:inputText size="15" value="#{operacoesBancariasBean.contaCliente.numero}" disabled="true"/>  
        <h:outputLabel value="Data Abertura:" />  
        <p:inputText size="35" value="#{operacoesBancariasBean.contaCliente.dataAbertura}" disabled="true">  
            <f:convertDateTime pattern="dd/MM/yyyy"/>  
        </p:inputText>  
        <h:outputLabel value="Saldo:" />  
        <p:inputText size="15" value="#{operacoesBancariasBean.contaCliente.saldo}" disabled="true">  
            <f:convertNumber currencySymbol="R$" type="currency" groupingUsed="true" />  
        </p:inputText>  
        <h:outputLabel value="Titular da Conta:" />  
        <p:inputText size="35" value="#{operacoesBancariasBean.contaCliente.titular.nome}" disabled="true"/>  
    </h:panelGrid>  
</p:panel>  
<p:spacer height="5" />  
<p:panel header="Dados Transferência">  
    <h:panelGrid columns="2" cellpadding="5">  
        <h:outputLabel value="Nº Conta Destino:" />  
        <p:inputText value="#{operacoesBancariasBean.numeroContaTransferencia}" />  
        <h:outputLabel value="Valor da Transferência:" />  
        <p:inputText value="#{operacoesBancariasBean.valorTransferencia}" />  
    </h:panelGrid>  
</p:panel>  
  
<p:spacer height="5" />  
<p:separator />  
<h:panelGrid columns="2" cellpadding="10">
```



```
<p:commandButton value="Realizar Transferência" update=":idDialogTransf:opBancariasForm"
  actionListener="#{operacoesBancariasBean.relizarTransferencia()}">
  <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogTransferencia}" value="false" />
</p:commandButton>

<p:commandButton value="Cancelar" update=":idDialogTransf:opBancariasForm">
  <f:setPropertyActionListener target="#{operacoesBancariasBean.flagDialogTransferencia}" value="false" />
</p:commandButton>

</h:panelGrid>
</h:form>
</p:dialog>
</html>
```

13. Modifique a classe **OperacoesBancariasBean.java** adicionando o método utilizado na página **dialogTransferencia.xhtml**.

```
public void relizarTransferencia() {
    if (valorTransferencia != null && valorTransferencia > 0) {
        try {
            Conta contaDestinoTransferencia = this.contaService.buscaContaPorNumero(this.numeroContaTransferencia);



            this.contaCliente = this.contaService.buscaContaPorNumero(this.contaCliente.getNumero());
            this.contaService.transferir(contaCliente, valorTransferencia, contaDestinoTransferencia);
            this.inicializaOperacoesConta();

            facesContext.addMessage(null, new FacesMessage
                (FacesMessage.SEVERITY_INFO, "Sucesso", "Transferência Realizado com sucesso!"));
        } catch (ContaNaoExisteException ex) {
            facesContext.addMessage(null, new FacesMessage
                (FacesMessage.SEVERITY_ERROR, "Erro.", ex.getMessage()));
        } catch (SaldoInsuficienteException e) {
            facesContext.addMessage(null, new FacesMessage
                (FacesMessage.SEVERITY_ERROR, "Erro.", e.getMessage()));
        }
    } else {
        facesContext.addMessage(null, new FacesMessage
            (FacesMessage.SEVERITY_ERROR, "Erro.", "Valor válido não informado!"));
    }
}
```

14. Execute a aplicação e execute todas as funcionalidades da aplicação Banco.

Projeto Banco 3Way NetWorks

Bem vindo, Sr.(a) admin

Sair

Banco: Banco Java Brasil

Agência: 3way NetWorks!

Manter Cliente

Operações Bancárias

Cliente: ewq

Nº Conta: 100

Data Abertura: 29/01/2014

Saldo: R\$ 50,00

Abrir Conta

Deposito

Saque

Transferencia

Historico de Transações

#	Tipo Transação	Titular/Conta Crédito	Titular/Conta Débito	Data	Valor
7	DEPOSITO	ewq/100	/	30/01/2014	R\$ 50,00
8	DEPOSITO	ewq/100	/	30/01/2014	R\$ 100,00
9	DEPOSITO	ewq/100	/	30/01/2014	R\$ 100,00
10	SAQUE	ewq/100	/	30/01/2014	R\$ 350,00
11	TRANSFERENCIA	wilker teste/100	ewq/100	30/01/2014	R\$ 50,00
12	DEPOSITO	ewq/100	/	30/01/2014	R\$ 100,00
13	DEPOSITO	ewq/100	/	30/01/2014	R\$ 100,00
14	DEPOSITO	ewq/100	/	30/01/2014	R\$ 100,00
15	DEPOSITO	ewq/100	/	30/01/2014	R\$ 100,00
16	DEPOSITO	ewq/100	/	30/01/2014	R\$ 100,00
17	TRANSFERENCIA	wilker teste/100	ewq/100	30/01/2014	R\$ 200,00
18	TRANSFERENCIA	ewq/100	ewq/100	30/01/2014	R\$ 150,00
19	SAQUE	ewq/100	/	30/01/2014	R\$ 50,00