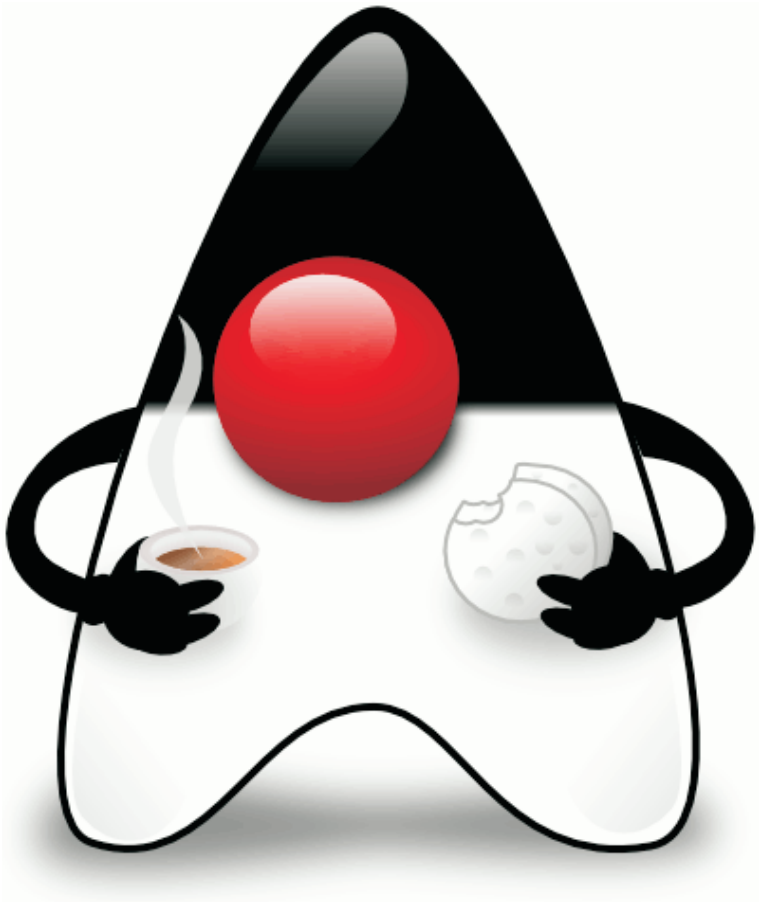
Mini-Curso

JSF - Hibernate

ceJug
Ceará Java Users Group

Desenvolvendo um aplicativo completo com JSF e Hibernate.



Mini-Curso

JSF - Hibernate

ceJug
Ceará Java Users Group

- JSF - Como ele funciona!?
  JSF como MVC.
  JSF - exemplo prático.
- O que é o Hibernate?
  Configurando o Hibernate.
- Iniciando o nosso projeto.
- Perguntas & Respostas

Mini-Curso

JSF - Hibernate

ceJug
Ceará Java Users Group

Para começar o nosso pequeno curso de JSF e Hibernate, iremos falar sobre o JSF, o que é, como funciona e como programar voltado para essa framework. Faremos um exemplo prático e comentaremos em cima do que estamos fazendo.

O que é o JSF!?

É um framework desenvolvido pela Sun Microsystems, e é parte integrante da tecnologia do mundo Java EE.

O framework Java Server Faces foi desenhado para facilitar o desenvolvimento de aplicações Web através de componentes de interface de usuário (GUI) e conecta estes componentes a objetos lógicos.

O JSF utiliza do paradigma MVC para trabalhar com sua apresentação e navegação de dados. Sua utilização é recomendada pela Sun para o desenvolvimento Web na atualizade.

Sobre o JSF!?

- Do mesmo criador do Struts;
- Paradigma de programação visual de User-interfaces aplicado à web;
- É um framework que permite a criação de aplicações Web com semântica de Swing implementando MVC;
- “Toolability = Ferramentabilidade” ;
- É uma especificação J2EE – JSR 127;
- Faces é mais fácil de aprender que Struts;
- Faces é mais componentizado;

Componentes para JSF

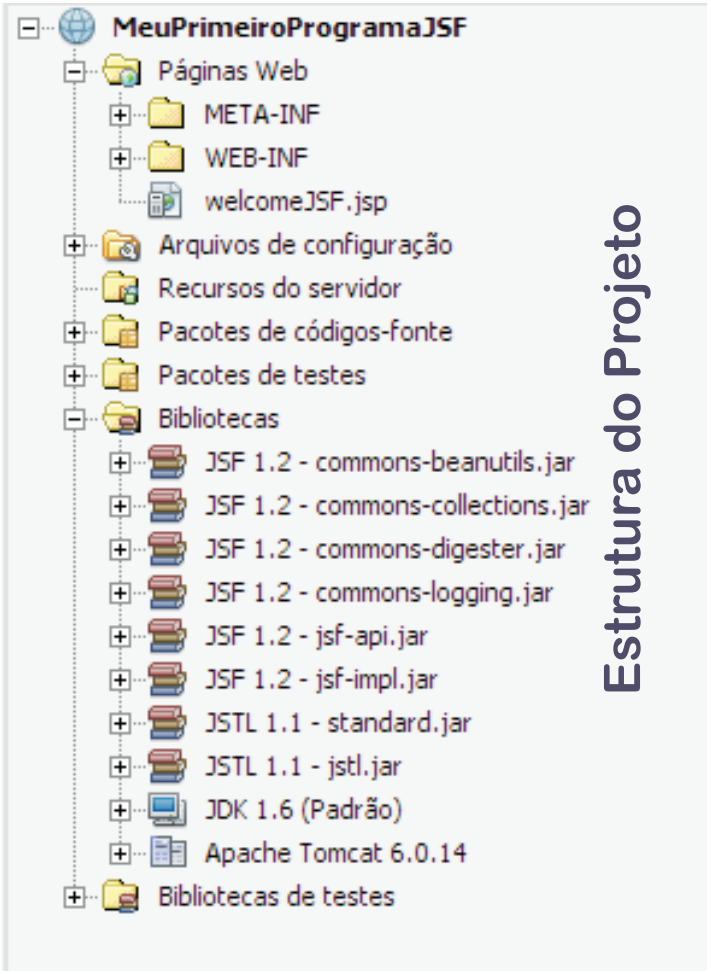
- DataGrid;
- Tabbed Panel;
- PanelGrid;
- SelectOneMenu, SelectOneRadio, SelectOneListBox
- SelectManyMenu, SelectManyRadio,
- SelectManyListBox;
- FileUpload;
- Auto-complete AJAX;
- Muitos outros...

Aplicação de Exemplo

Para nossa primeira aplicação de exemplo iremos criar o famoso "Hello World".

Crie um novo projeto e chame-o de "MeuPrimeiroProgramaJSF".
Na opção de selecionar o servidor escolha "Tomcat".
Na seção de escolha do nosso framework escolha o "JavaServer Faces" ou "JSF".

Pronto, teremos um projeto criado, totalmente voltado para o JSF!



*Obs.: A IDE utilizada nesse mini-curso é o Netbeans 6.0

Aplicação de Exemplo

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <f:view>
            <h1><h:outputText value="JavaServer Faces" /></h1>
        </f:view>
    </body>
</html>
```

Note que temos logo de cara um arquivo “welcomeJSF.jsp” que é um default do projeto, iremos alterá-lo para que possamos entender um pouco dos componentes.

Aplicação de Exemplo

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
```

Para utilizar a framework JSF, precisamos declarar nos cabeçalhos de nossas páginas *.jsp as bibliotecas que utilizaremos no nosso caso aqui a JSF CORE e a JSF HTML, que é a padrão de todos desenvolvimento.

```
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <f:view>
            <h1><h:outputText value="JavaServer Faces" /></h1>
        </f:view>
    </body>
</html>
```

Aplicação de Exemplo

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <f:view>
            <h1><h:outputText value="JavaServer Faces" /></h1>
        </f:view>
    </body>
</html>
```

Aqui temos o uso das bibliotecas para utilização do JSF. Por padrão toda página JSF tem que ter o <f:view>. Esta tag é o início da árvore de componentes, em seguida temos as tags de JSF HTML, <h:outputText>.

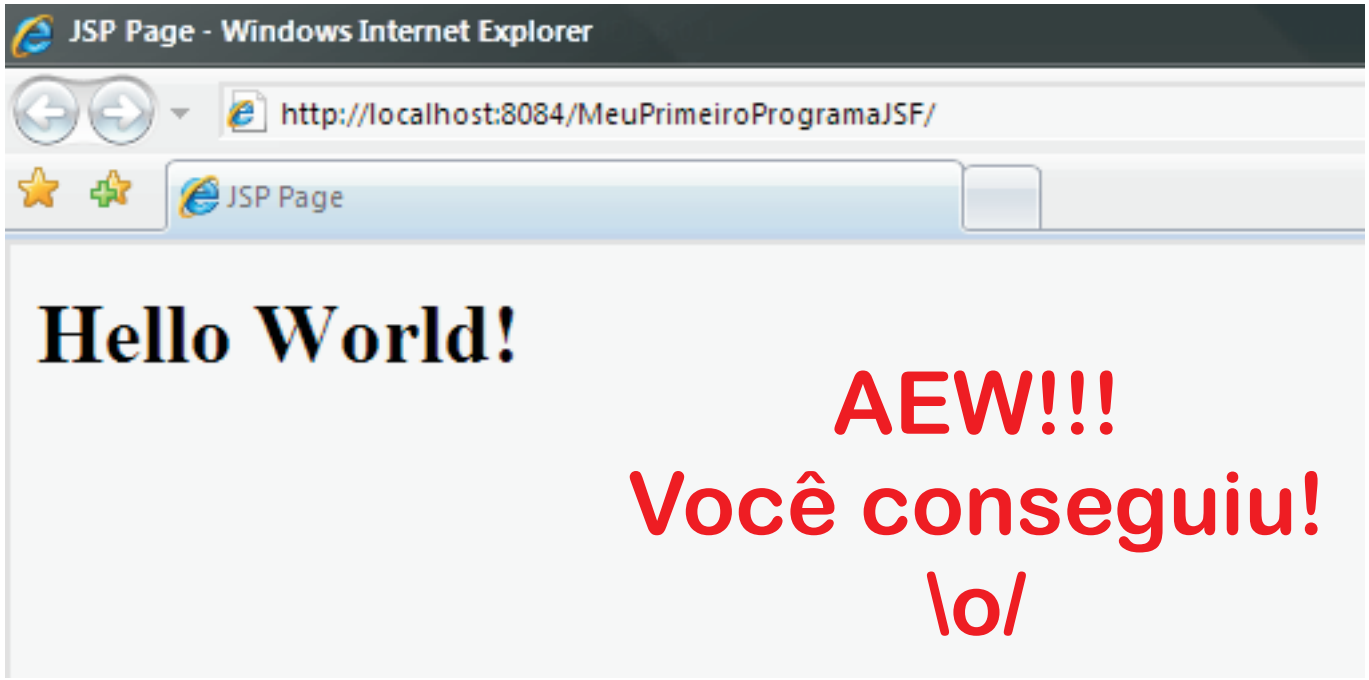
Note que o JSF está interagindo normalmente com tags HTML padrão. Isso é mais uma vantagem que o JSF nos traz. Mais a frente veremos como isso nos poder ser útil, muito útil. ;)

Aplicação de Exemplo

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <f:view>
            <h1><h:outputText value="Hello World!" /></h1>
        </f:view>
    </body>
</html>
```

Altere o valor do <h:outputText> para "Hello World". E mande executar o projeto.

Aplicação de Exemplo



Aplicação de Exemplo

Bom... Até agora nada de tão espetacular, mas vamos engrossar aqui um pouco desse caldo.

Na nossa segunda aplicação com JSF, iremos trabalhar com JavaBean, como passar valores numa outra página, como alterar o nosso objeto, como validar algumas coisas.

NetBeans

Mão a obra!

Crie um novo projeto e chame-o de "PrimJSFDinamico".
Selecione o "Tomcat" como servidor e no framework marque somente "JavaServer Faces"

Aplicação de Exemplo

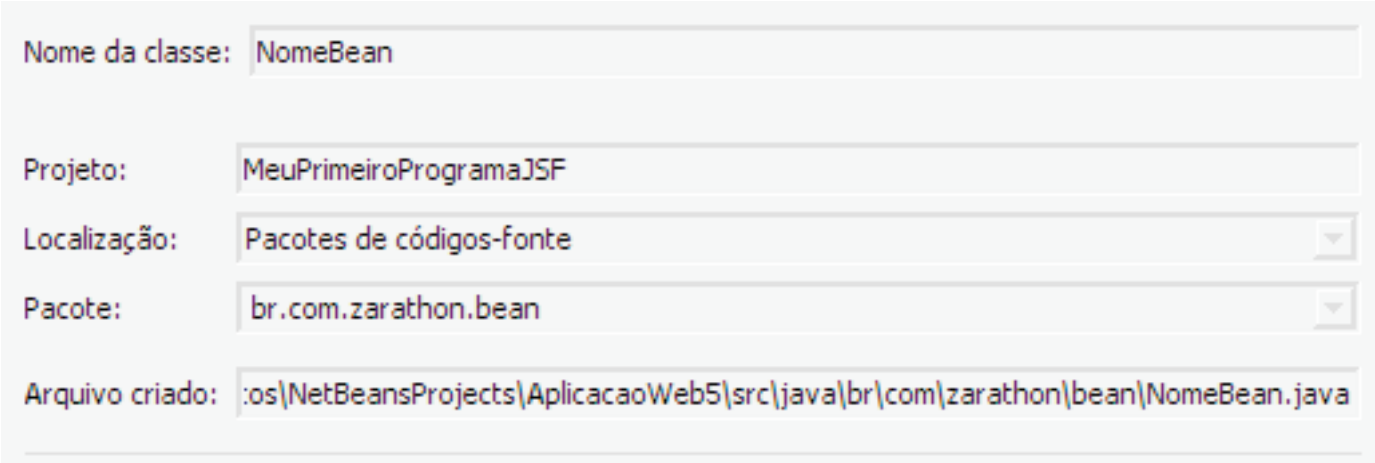
Nessa nossa segunda aplicação teremos um campo para o envio de nomes. Este exercício contará com uma validação, para o caso do usuário entrar com um valor inválido, não alfabético, retornando um erro. Caso retorne o erro, além de manter preenchido o campo digitado, também mostrará uma mensagem, solicitando a alteração.

Let's go!

Aplicação de Exemplo

O Javabean mostrado a seguir será o responsável pela comunicação entre as página inicial, que o usuário digitará o nome, em um formulário, e a página que resultará na mensagem de boas vindas, caso esta seja submetida com sucesso.

Crie uma nova classe, File>New File> Java>Class, como o nome de "NomeBean" e a coloque em um pacote de beans, nesse caso "br.com.zarathon.bean".



Aplicação de Exemplo

Código de NomeBean

```
package br.com.zarathon.bean;

import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;

public class NomeBean {

    private String nome = null;

    public String getNome() {
        return nome;
    }

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

    public String acao() {
        boolean sucesso = true;
        FacesContext context = FacesContext.getCurrentInstance();
        if (nome != null) {
```

Com a instância de FacesContext você obtém todas as informações de estado por requisição usadas para o processamento de um pedido JSF. O médoto getCorrentInstance() obtém a instância atual da classe FacesContext.

Aplicação de Exemplo

Código de NomeBean (Continuação)

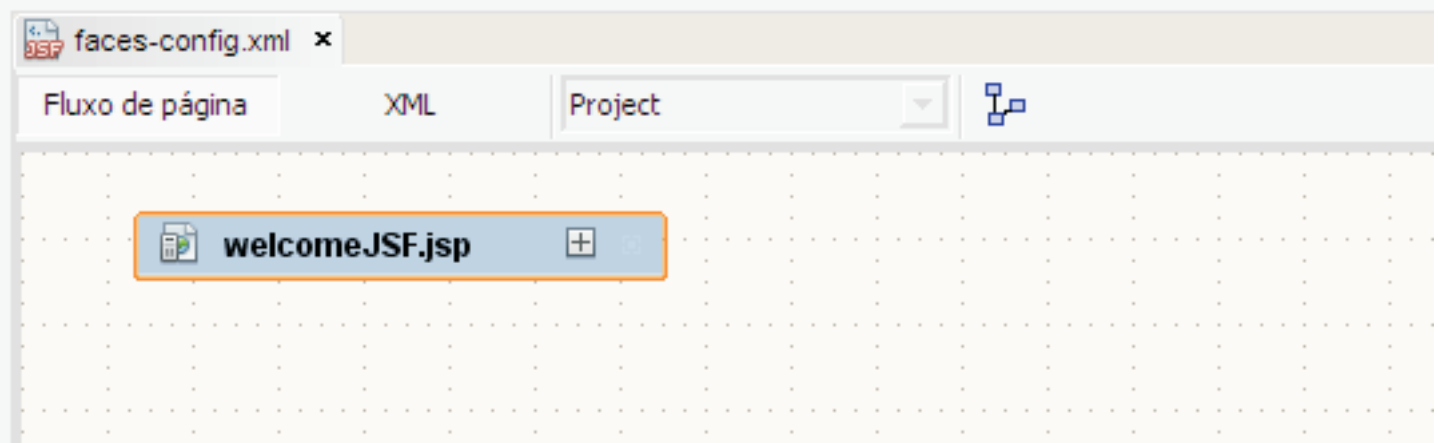
```
            for (int i = 0; i < nome.length(); i++) {
                char c = nome.charAt(i);
                if (!Character.isLetter(c)) {
                    String msg = "Digite somente caracteres alfabéticos.";
                    FacesMessage message = new FacesMessage(msg);
                    context.addMessage("form", message);
                    sucesso = false;
                    break;
                }
            }
        } else {
            sucesso = false;
        }
        return (sucesso ? "sucesso" : "falha");
    }
}
```

Para adicionar uma mensagem, a classe FacesMessage representa uma única validação ou mensagem que é tipicamente associada à um componente particular na view. Neste caso, o método addMessage(), da instância FacesContext, é chamado. A mensagem anexada está associada ao componente UI, se este não for nulo.

Aplicação de Exemplo

Configurando a navegação da sua aplicação

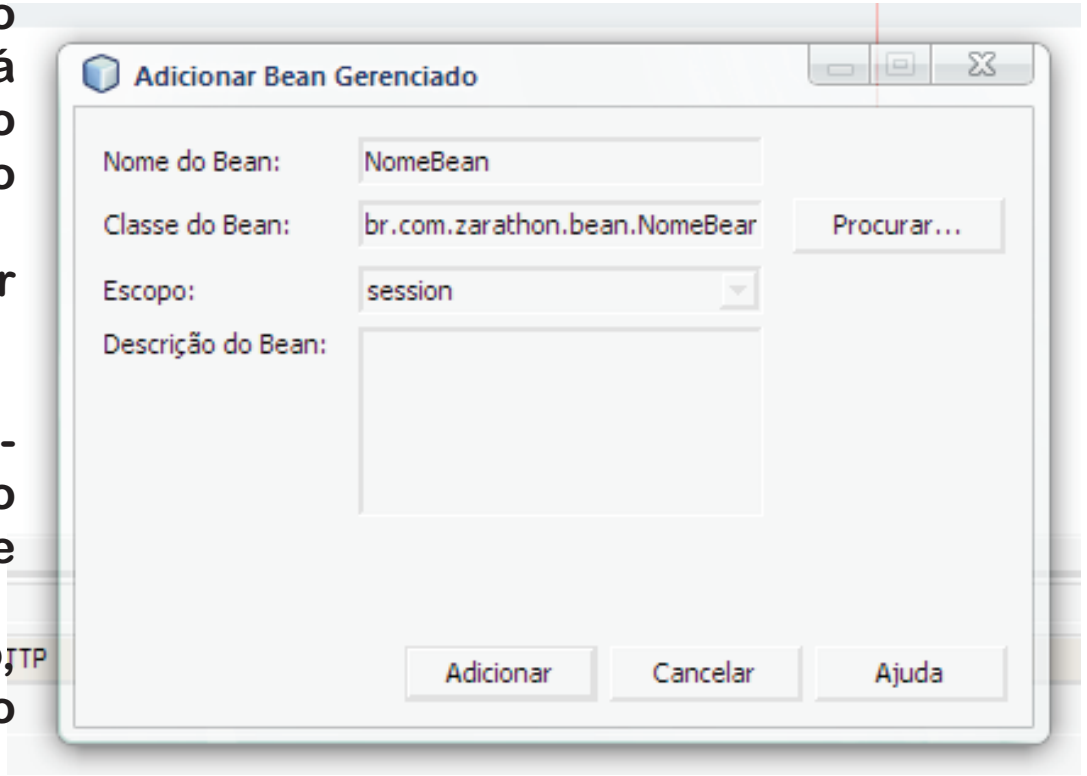
Toda a navegação da sua aplicação passa pelo arquivo de configuração "faces-config.xml". Desta forma, este arquivo já deve ter sido adicionado pela IDE, uma vez que o projeto tem uma pré-configuração para JavaServer Faces.



Aplicação de Exemplo

Configurando a navegação da sua aplicação

Iremos agora adicionar o nosso NomeBean ao nosso framework JSF, pois o mesmo poderá alterar os atributos do mesmo e efetuando seus métodos. para isso mude a visualização do faces-config.xml de PageFlow para XML.
Feito isso, você verá a página XML que fica por trás do PageFlow.
Adicionaremos agora o nosso NomeBean.
Em uma linha vazia, entra as tags <faces-config> e </faces-config>, clique com o botão direito e selecione “Add Manager Bean” que fica no menu JavaServer Faces.
Abrirá uma janelinha, como essa aí do lado, basta agora só preencher com os nosso dados.



Aplicação de Exemplo

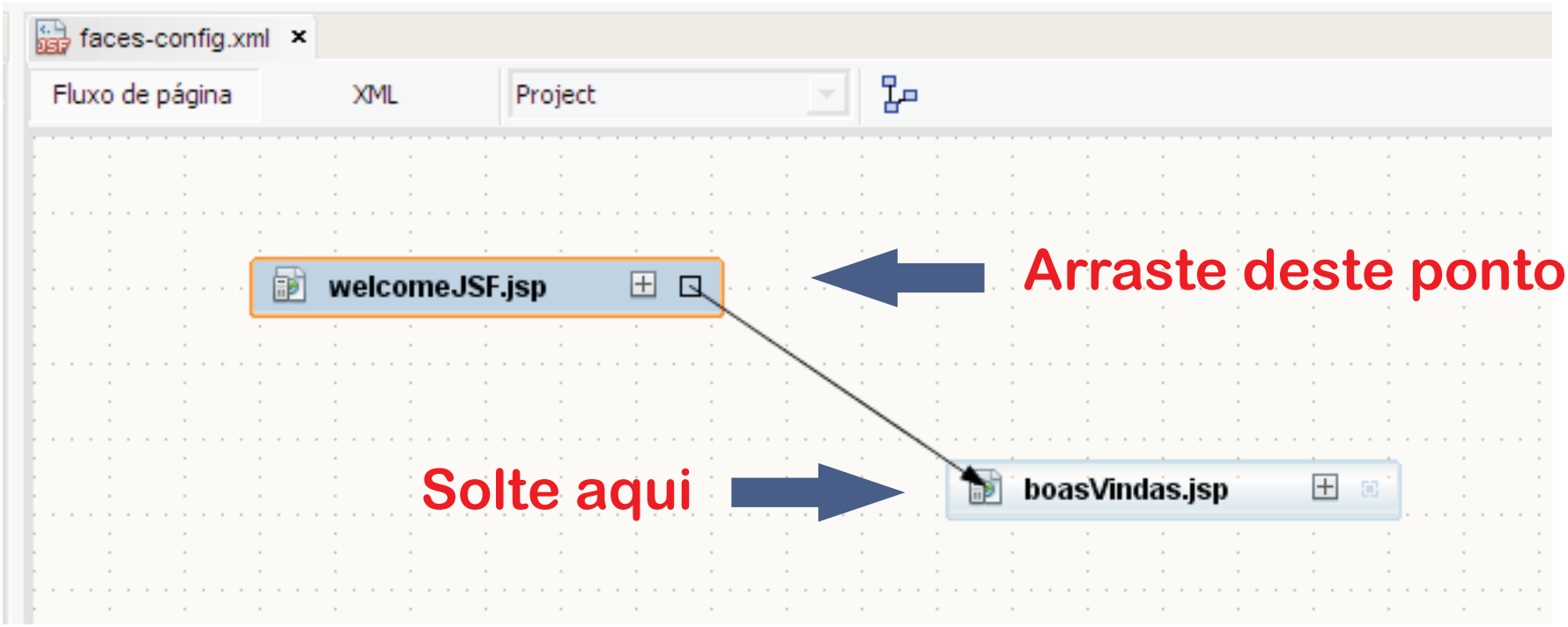
Configurando a navegação da sua aplicação

Crie um novo arquivo jsp, que exibirá a nossa mensagem de boas vindas, chame-o de “boasVindas”. Deixe-o em branco por enquanto, estamos aqui configurando nossa navegação e não nossa programação.

Retornando à faces-config.xml, agora você possui duas páginas sendo representadas graficamente no PageFlow. No canto direito da imagem que representa a página welcome.jsp, existe uma quadrado. Arrastando deste quadrado, você faz uma linha de navegação, ao qual pode apontar para a mesma página ou para outra. Arraste de welcome.jsp até a imagem que representa boasVindas.jsp e solte.

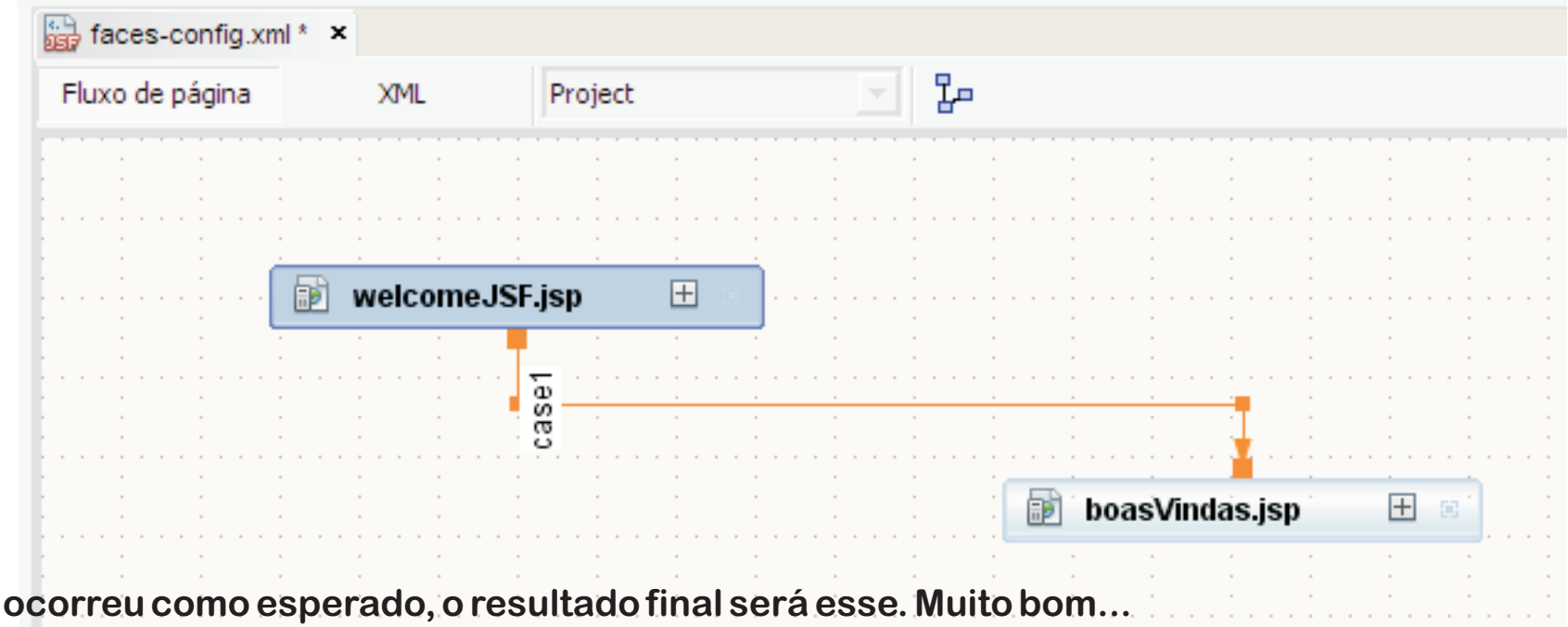
Aplicação de Exemplo

Configurando a navegação da sua aplicação



Aplicação de Exemplo

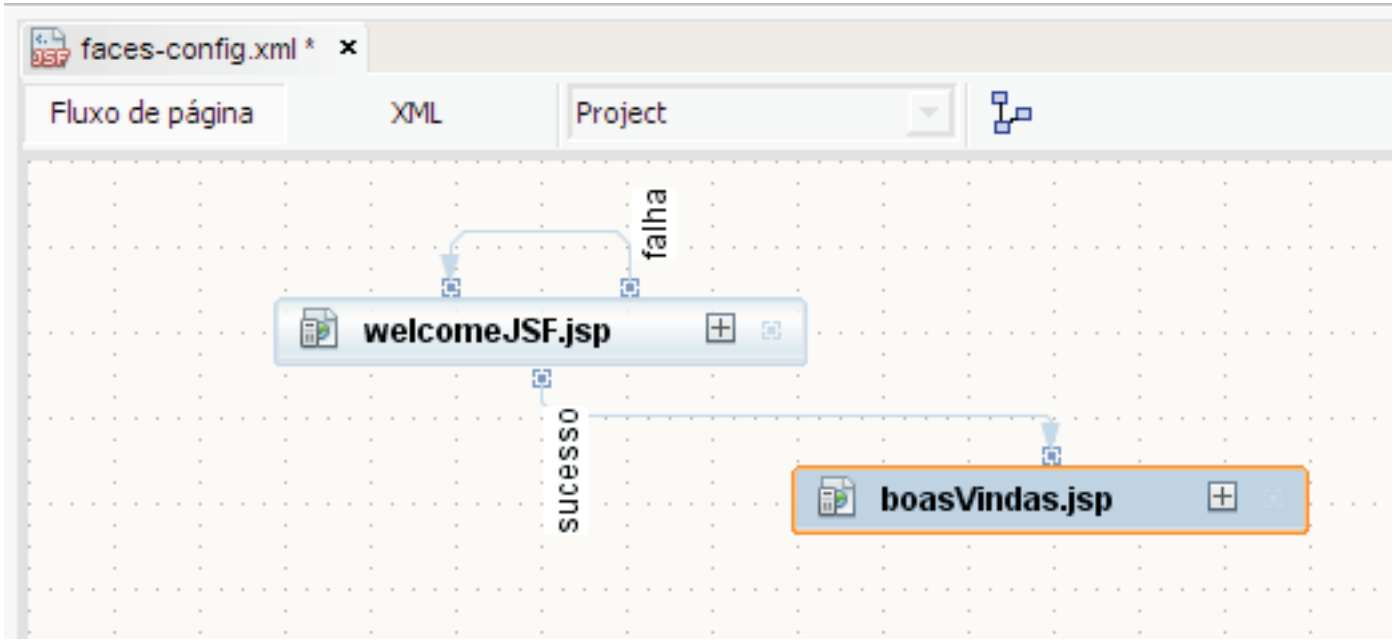
Configurando a navegação da sua aplicação



Se tudo ocorreu como esperado, o resultado final será esse. Muito bom...
Agora temos a nossa primeira navegação, porém ela gera um texto estranho para nós, CASE1. Iremos alterar esse texto para algum texto que faça sentido para nós, altere para "sucesso", tudo minusculo. Isso quer nos dizer que quando vier uma resposta com texto "sucesso" ele redirecionará para a página boasVindas.jsp. Faça o mesmo processo só que agora arraste a resposta para o próprio welcomeJSF.jsp, e no texto que aparecer altere para "falha", tudo minusculo.

Aplicação de Exemplo

Configurando a navegação da sua aplicação



Se você fez tudo certo, você terá algo desse tipo.

Parabéns, sua visualização está concluída!

Veja a aba XML e veja o código gerado.

Aplicação de Exemplo

Até agora nada de páginas... Mas isso acabou...
Vamos agora montar nossas páginas da nossa aplicação.

welcomeJSF.jsp

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Primeira Aplicação com JSF</title>
    </head>
    <body>
        <f:view>
            <h:form id="form">
                Digite seu nome:
                <h:inputText id="nome" value="#{NomeBean.nome}"
                             required="true" requiredMessage="Campo Obrigatório!"/>
                <h:commandButton action="#{NomeBean.acao}" value="Enviar" id="submit" />
                <br/>
                <h:messages/>
            </h:form>
        </f:view>
    </body>
</html>
```

Aplicação de Exemplo

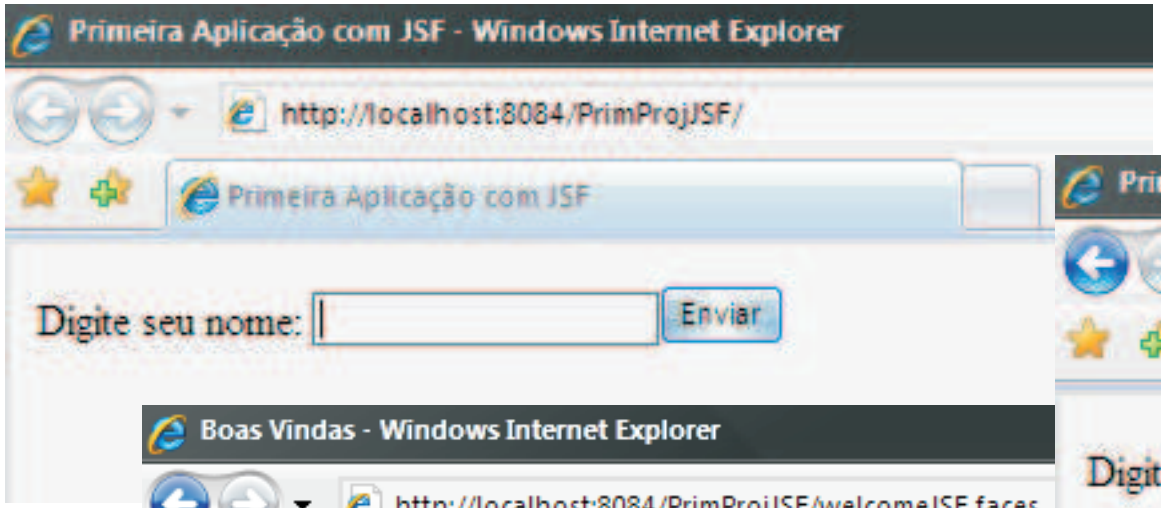
Até agora nada de páginas... Mas isso acabou...
Vamos agora montar nossas páginas da nossa aplicação.

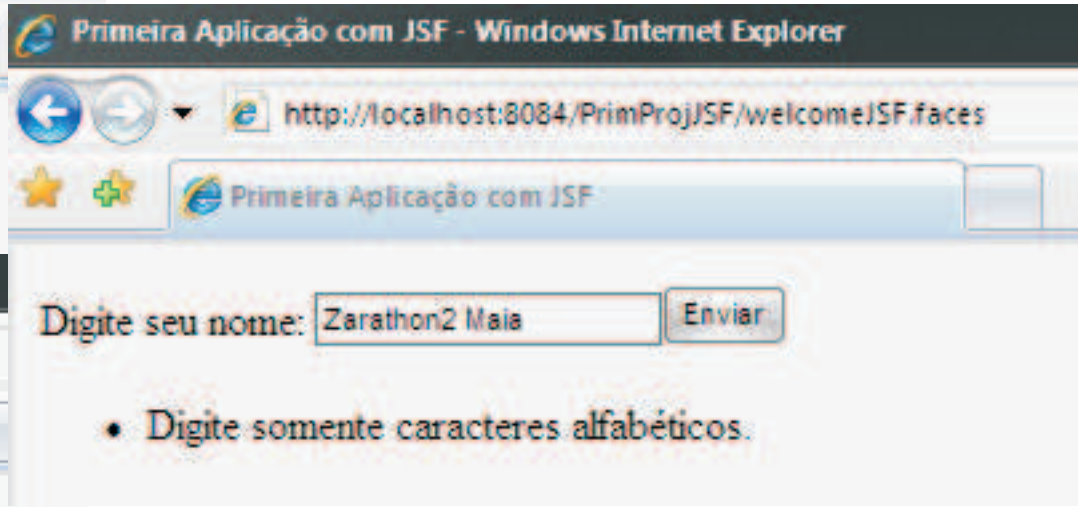
boasVindas.jsp

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Boas Vindas</title>
    </head>
    <body>
        <f:view>
            <h2><h:outputText value="Olá, #{NomeBean.nome}"/></h2>
        </f:view>

    </body>
</html>
```

Aplicação de Exemplo

Muito bem, agora só o "Gran Finale", mande rodar a aplicação e veja o resultado!





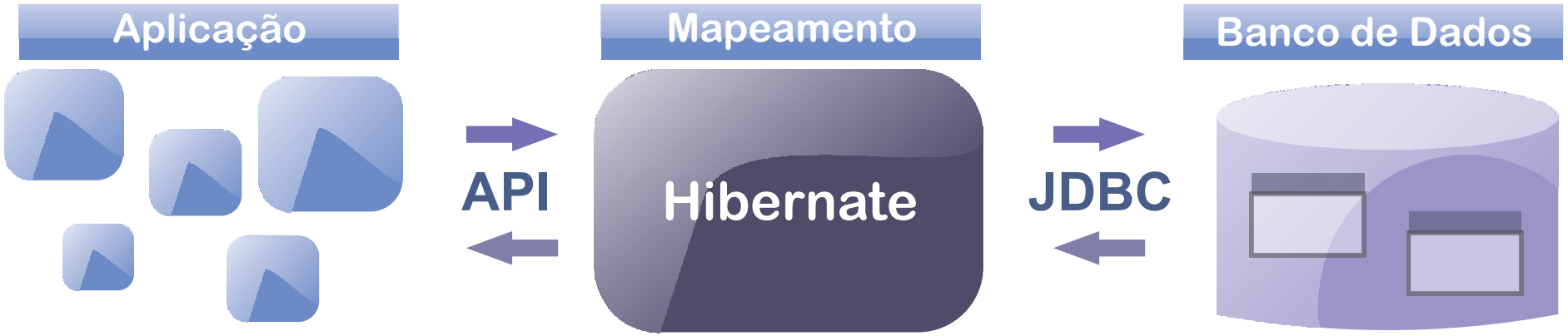
Hibernate

persistência... persistência... persistência...

Hibernate

Visão Geral

- O Hibernate é um framework de mapeamento objeto-relacional para a linguagem Java
- Conjunto de classes, interfaces e configuração que permite simplificar o trabalho de persistir e recuperar objetos Java em banco de dados relacionais



ceJug
Ceará Java Users Group

Hibernate

Configurando o Hibernate

- Configuração feita através do arquivo hibernate.cfg.xml
- Deve estar localizado na raiz do classpath
- Localização default
- Para projetos maven, utilizar a pasta src/main/resources
- Configurações contém
- Parâmetros de acesso a base de dados
- Pool de conexões
- Entidades a serem persistidas

ceJug
Ceará Java Users Group

Hibernate

NA PRÁTICA!

Início do Projeto

Iremos desenvolver uma aplicação que cadastre autores e esses autores poderão ter livros publicados. Faremos também um controle de login do administrador do cadastro. E por fim faremos a impressão dos autores com seus respectivos livros.

O PROJETO!

Configurando o arquivo "hibernate.cfg.xml".
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

Mini-Curso
JSF - Hibernate

ceJug
Ceará Java Users Group

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

O PROJETO!

Configurando o arquivo “hibernate.cfg.xml”.
Criando a classe Usuário.
Mapeando a classe Usuário.
Gerando as tabelas do banco.
Teste da classe Usuário.
Criação do DAO da classe Usuário.
Criando o HibernateUtil.
Criando o DAOFactory.
Transformando o DAO em DAO Genérico.
Criando páginas para cadastro de usuários e lista de usuário.
Criando a classe Autor.
Criando páginas para cadastro de autores e lista de autores.
Criando a classe Livro.
Criando páginas para cadastro de livros e lista de livros.
Fazendo o sistema de Login.
Montando nosso controle.

ceJug
Ceará Java Users Group

DÚVIDAS!?

Bibliografia

Apostilas da Caelum FJ21, FJ28

Livro Desenvolvendo aplicações Web com Netbeans 6.0, Edson Gonçalves, Ed. Ciência Moderna

Apresentação: HIBERNATE de Marcelo Mrack