# Introdução ao Desenvolvimento Web

JAVASERVER FACES E HIBERNATE COM MVC

PROF. PABLO VARGAS

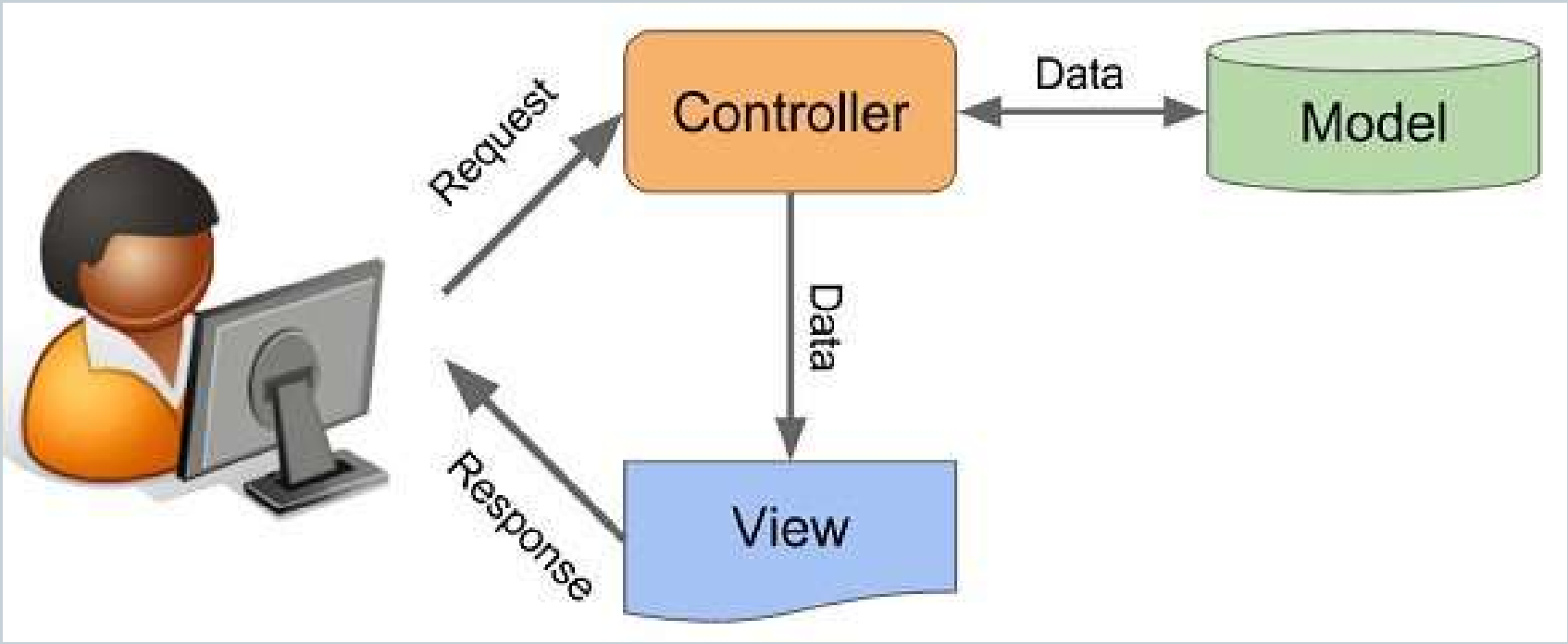
# Tópicos Abordados

- MVC
- Separando as responsabilidades.
- Cadastro de Usuários.
- Administração de Usuários.

# MVC

- “é um padrão de arquitetura muito utilizado no desenvolvimento de projeto web.”
- Idealizado em 1979 por um norueguês chamado Trygve Reenskaug.
- Sistema separado em 03 camadas:
  - Model: camada de acesso a dados e regra de negócio.
  - View: interface do sistema. (Arquivos JSF e XHTML)
  - Controller: controla as camadas model e view. (Managed Bean)

# MVC



# MVC

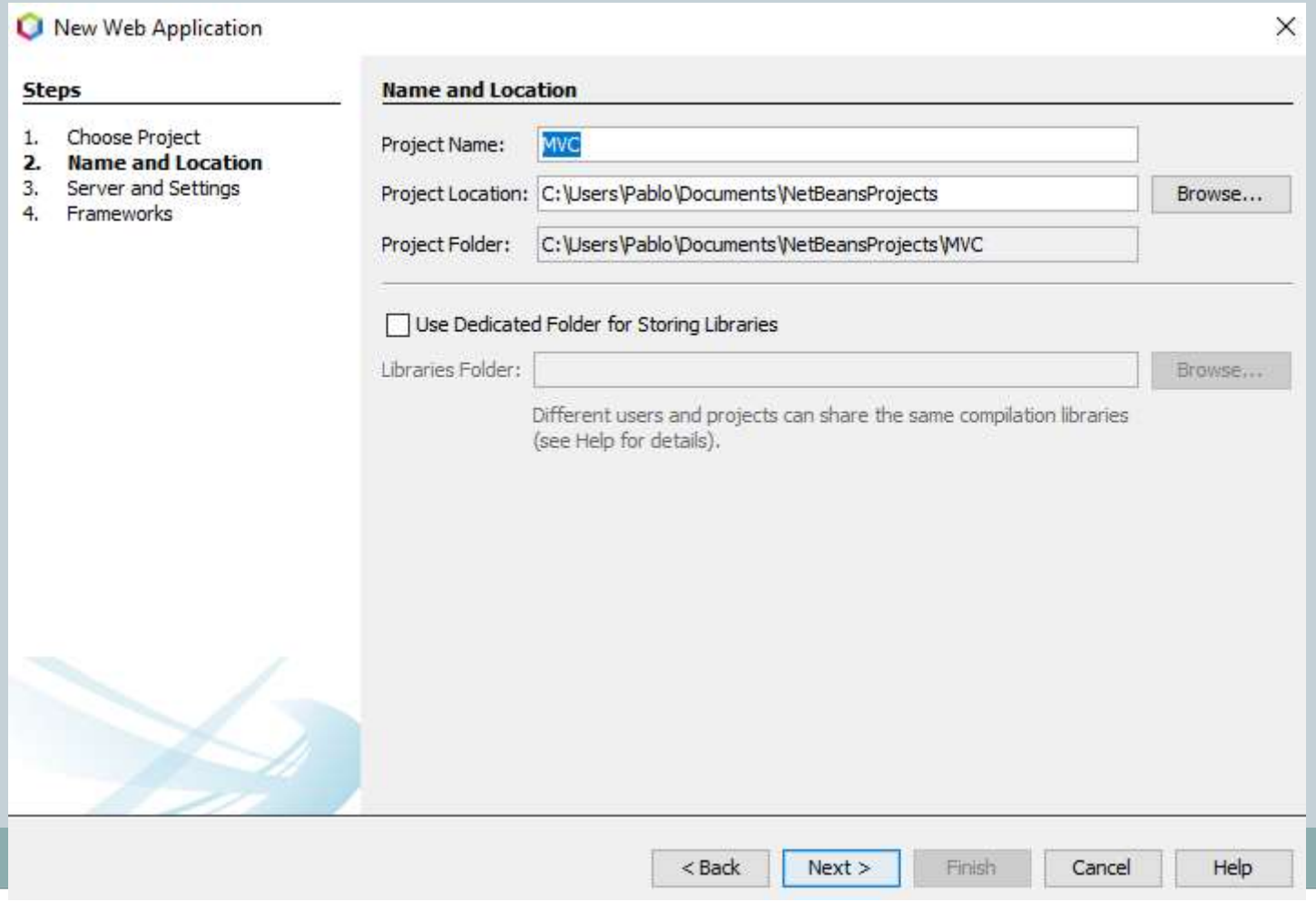
- https://www.youtube.com/watch?v=jyTNhT67ZyY

# Separando as responsabilidades

- Uma classe, uma responsabilidade.
  - Sistemas com responsabilidades separadas e bem definidas tende a ter boa manutenibilidade.
- Arquitetura de 03 Camadas separa as responsabilidades.
  - Acesso aos dados: onde ocorre todas operações  com os dados.
  - Regras de negócio: são responsáveis pelas tomadas de decisão.
  - Apresentação: representa as telas do sistema. Exibe e coleta informações.
- POJO (Plain and Old Java Object): São as classes que trafegam pelas camadas. É recomendado nessa classe apenas suas propriedades e métodos get/set.

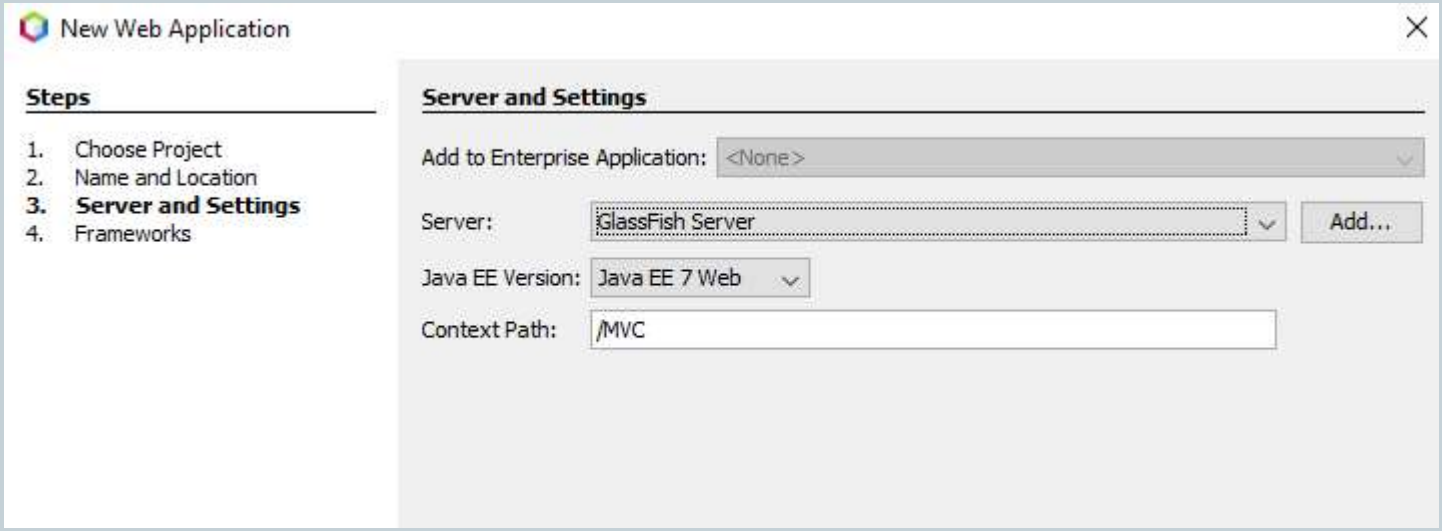
# Cadastro de Usuários

- Crie um novo projeto chamado MVC



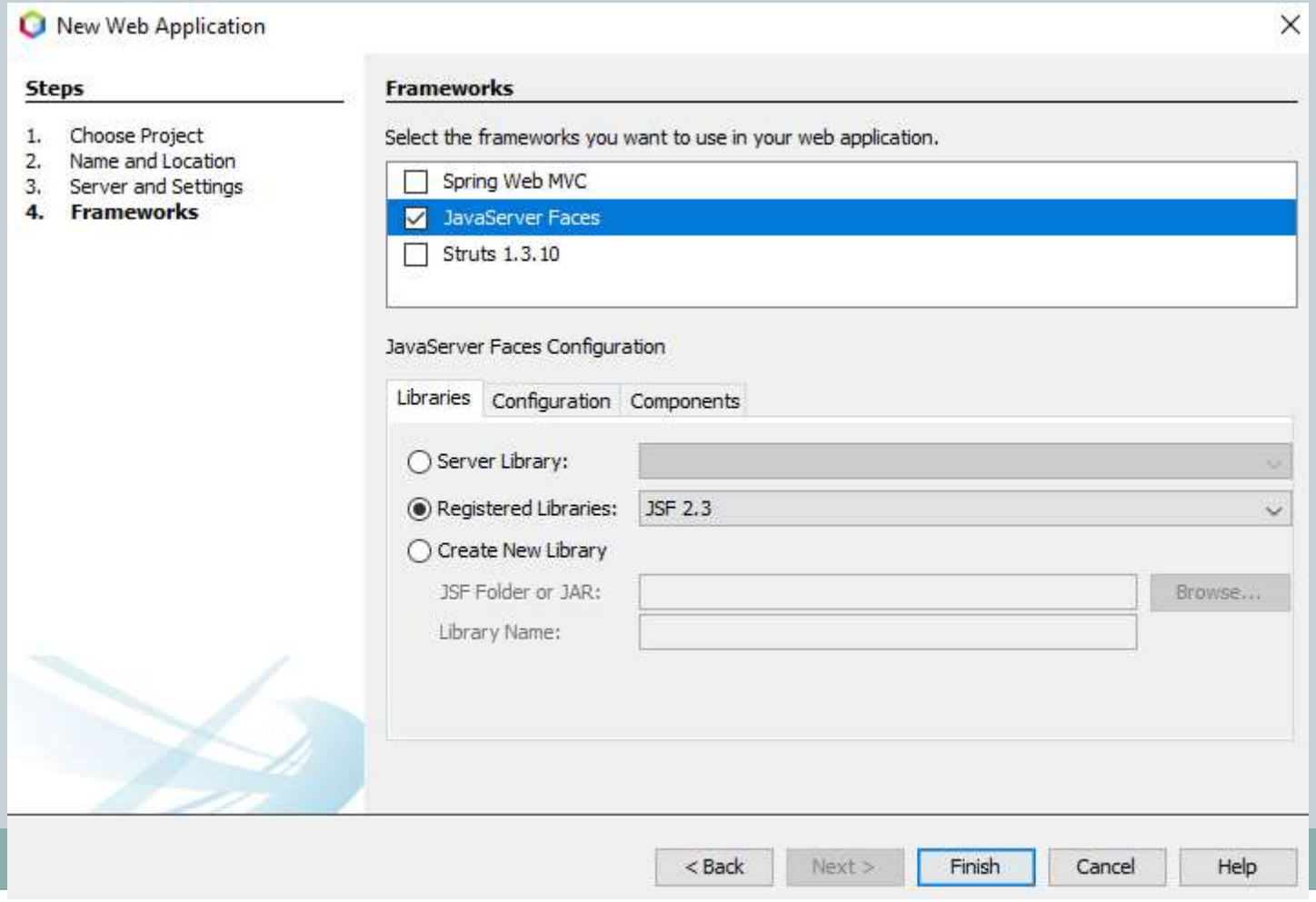
# Cadastro de Usuários

- Crie um novo projeto chamado MVC



# Cadastro de Usuários

- Crie um novo projeto chamado MVC

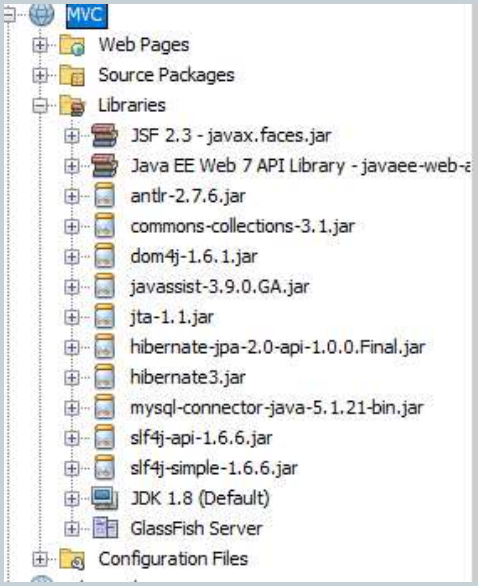


# Cadastro de Usuários

- Adicione todas os jars dos exemplos anteriores ao projeto principal.
- Copie a classe “HibernateUtil” do exemplo anterior no pacote “util”.
- Crie o BD idw no MySQL.
  - Create database  idw;

# Cadastro de Usuários

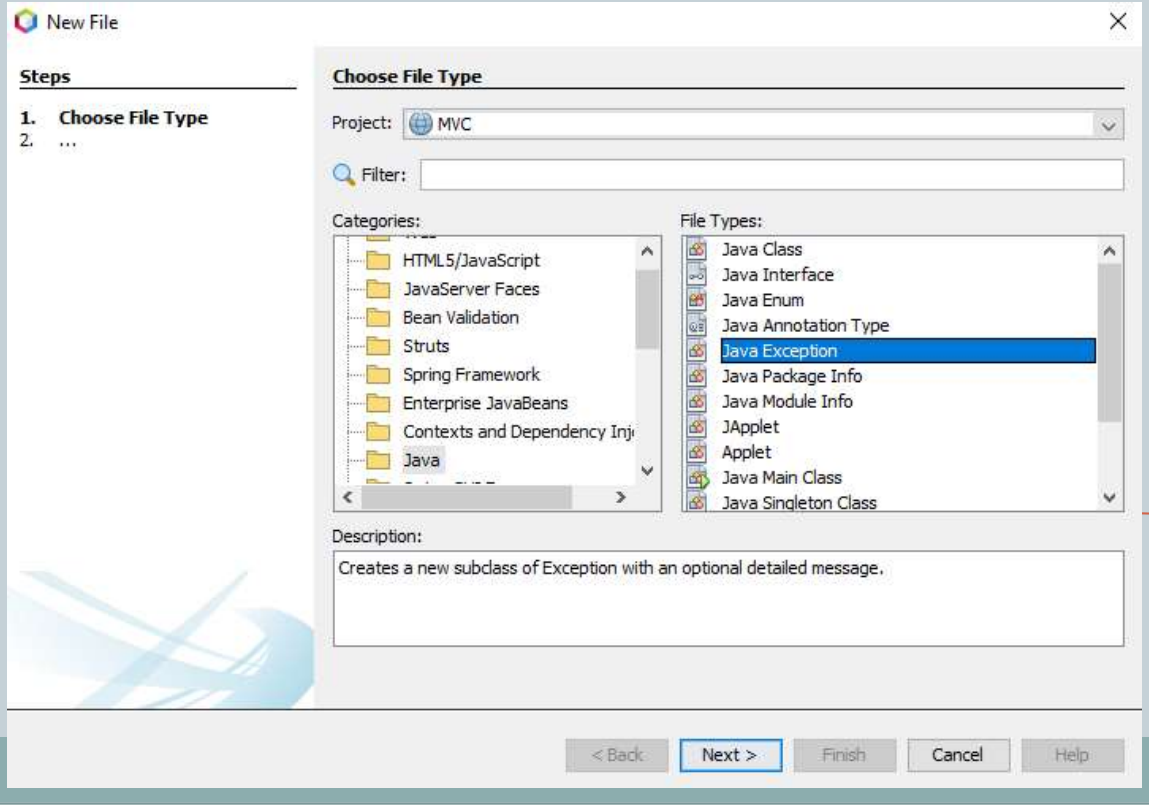
- Biblioteca do projeto e Classe HibernateUtil.java:




```
public class HibernateUtil {

    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            AnnotationConfiguration cfg = new AnnotationConfiguration();
            cfg.configure("hibernate.cfg.xml");
            return cfg.buildSessionFactory();
        } catch (Throwable e) {
            System.out.println("Criação inicial do objeto SessionFactory falhou. Erro: " + e);
            throw new ExceptionInInitializerError(e);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

# Cadastro de Usuários

- Criação das exceções-padrão.
  - Crie os arquivos exceções java “DAOException”, “RNException” e “UtilException” no pacote “util”



# Cadastro de Usuários

- Crie a classe “Usuario” no pacote “usuario” com o seguinte código.

```
@Entity
@Table(name = "usuario")
public class Usuario implements Serializable {

    @Id
    @GeneratedValue
    @Column(name = "codigo", nullable = false)
    private Integer codigo;
    @Column(name = "nome")
    private String nome;
    @Column(name = "email")
    private String email;
    @Column(name = "login")
    @org.hibernate.annotations.NaturalId
    private String login;
    @Column(name = "senha")
    private String senha;
    @Column(name = "nascimento")
    private Date nascimento;
    @Column(name = "celular")
    private String celular;
    @Column(name = "ativo")
    private boolean ativo;
```

# Cadastro de Usuários

```
@ElementCollection(targetClass = String.class)
@JoinTable(
        name = "usuario_permissao",
        uniqueConstraints = {
            @UniqueConstraint(columnNames = {"usuario", "permissao"})},
        joinColumns = @JoinColumn(name = "usuario"))
@Column(name = "permissao", length = 50)
private Set<String> permissao = new HashSet<String>();

//gerar getters e setters (alt + insert)
//gerar equals e hashCode (alt + insert)
```

# Cadastro de Usuários

- Add os seguintes imports:
  - import java.io.Serializable;
  - import javax.persistence.Column;
  - import javax.persistence.ElementCollection;
  - import javax.persistence.Entity;
  - import javax.persistence.GeneratedValue;
  - import javax.persistence.Id;
  - import javax.persistence.JoinColumn;
  - import javax.persistence.JoinTable;
  - import javax.persistence.Table;
  - import javax.persistence.UniqueConstraint;

# Cadastro de Usuários

- Crie a interface “UsuarioDAO” no pacote “usuario” com o seguinte código.

```
public interface UsuarioDAO {

        public void salvar(Usuario usuario);

        public void atualizar(Usuario usuario);

        public void excluir(Usuario usuario);

        public Usuario carregar(Integer codigo);

        public Usuario buscarPorLogin(String login);

        public List<Usuario> listar();
}
```

# Cadastro de Usuários

- Crie a classe “UsuarioDAOHibernate” no pacote “usuario” com o seguinte código.

```
public class UsuarioDAOHibernate implements UsuarioDAO {
    private Session session;

    public void setSession(Session session) {
        this.session = session;
    }

   public void salvar(Usuario usuario) {
                this.session.saveOrUpdate(usuario);
        }

        public void atualizar(Usuario usuario) {
                this.session.merge(usuario);
        }

        public void excluir(Usuario usuario) {
                this.session.delete(usuario);
        }

        public Usuario carregar(Integer codigo) {
                return (Usuario) this.session.get(Usuario.class, codigo);
        }
```

# Cadastro de Usuários

```
        public Usuario buscarPorLogin(String login) {
                String hql = "select u from Usuario u where u.login = :login";
                Query consulta = this.session.createQuery(hql);
                consulta.setString("login", login);

                return (Usuario) consulta.uniqueResult();
        }

        @SuppressWarnings("unchecked")
        public List<Usuario> listar() {
                return this.session.createCriteria(Usuario.class).list();
        }
}
```

# Cadastro de Usuários

- Crie a classe “DAOFactory” no pacote “util” com o código abaixo.

```
public class DAOFactory {

        public static UsuarioDAO criarUsuarioDAO() {
                UsuarioDAOHibernate usuarioDAO = new UsuarioDAOHibernate();
                usuarioDAO.setSession(HibernateUtil.getSessionFactory().getCurrentSession());
                return usuarioDAO;
        }
}
```

# Cadastro de Usuários

- Crie a classe “ConexaoHibernateFilter” no pacote “web.filter”.

```
public class ConexaoHibernateFilter implements Filter {

        private SessionFactory  sf;

        public void init(FilterConfig config) throws ServletException {
                this.sf = HibernateUtil.getSessionFactory();
        }

        public void destroy() {
        }
```

# Cadastro de Usuários

```
        public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,
                FilterChain chain) throws ServletException {

                try {

                        this.sf.getCurrentSession().beginTransaction();

                        chain.doFilter(servletRequest, servletResponse);

                        this.sf.getCurrentSession().getTransaction().commit();
                        this.sf.getCurrentSession().close();

                } catch (Throwable ex) {
                        try {
                                if (this.sf.getCurrentSession().getTransaction().isActive()) {
                                        this.sf.getCurrentSession().getTransaction().rollback();
                                }
                        } catch (Throwable t) {
                                t.printStackTrace();
                        }
                        throw new ServletException(ex);
                }
        }

}
```

# Cadastro de Usuários

- Modifique o trecho abaixo no arquivo “web.xml”.

```
<context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
</context-param>
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
<welcome-file-list>
    <welcome-file>Publico/login.jsf</welcome-file>
</welcome-file-list>
```

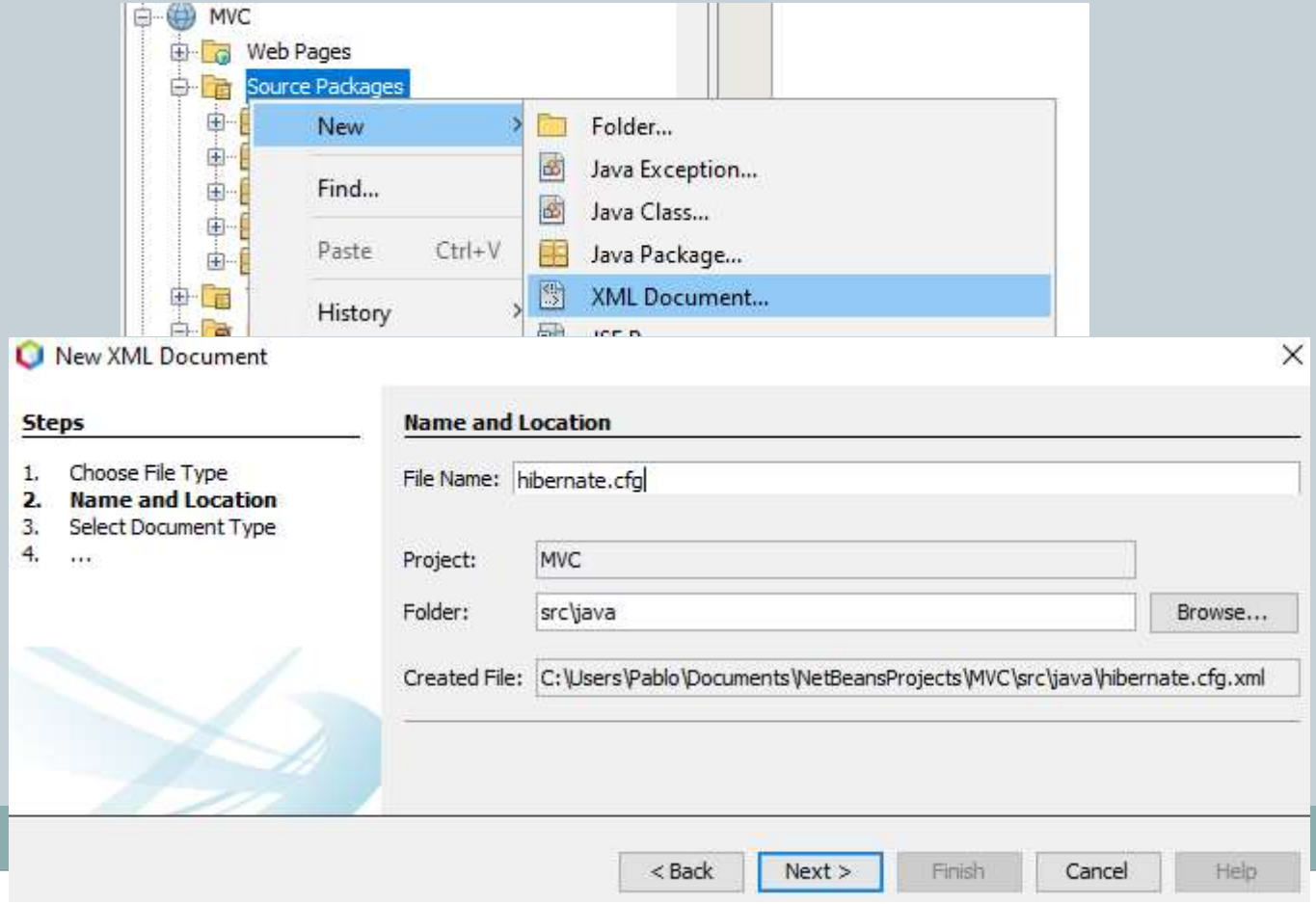
# Cadastro de Usuários

- Inclua o trecho abaixo no arquivo “web.xml”.

```
<filter>
    <filter-name>conexaoFilter</filter-name>
    <filter-class>web.filter.ConexaoHibernateFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>conexaoFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

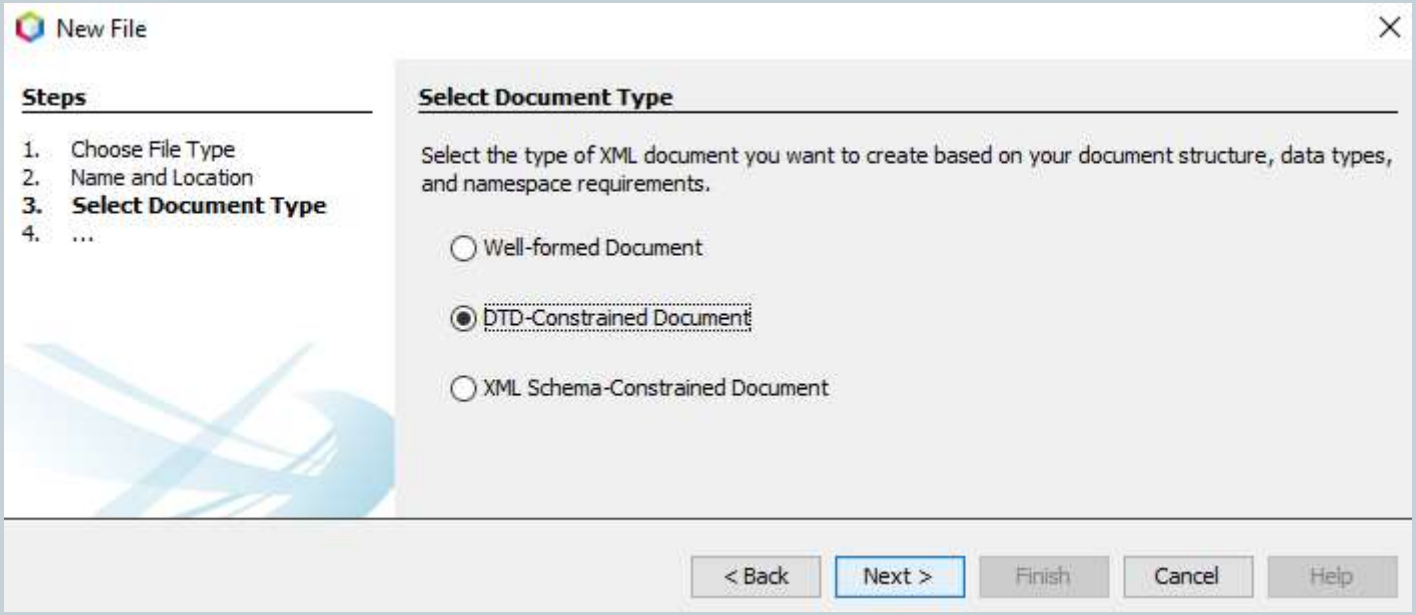
# Cadastro de Usuários

- Crie o arquivo XML do Hibernate com o nome “hibernate.cfg” igual ao exemplo anterior.



# Cadastro de Usuários

- Selecione o tipo de documento DTD, clique em next e na próxima janela finish.



# Cadastro de Usuários

- Modifique o arquivo XML do Hibernate com o seguinte código.

```
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost/idw</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>
        <property name="current_session_context_class">thread</property>
        <property name="hibernate.hbm2ddl.auto">create</property>
        <mapping class="usuario.Usuario"/>
    </session-factory>
</hibernate-configuration>
```

# Cadastro de Usuários

- Crie a classe “UsuarioRN” no pacote “usuario” com o código abaixo.

```
public class UsuarioRN {
    private UsuarioDAO  usuarioDAO;

        public UsuarioRN() {
                this.usuarioDAO = DAOFactory.criarUsuarioDAO();
        }

        public Usuario carregar(Integer codigo) {
                return this.usuarioDAO.carregar(codigo);
        }

        public Usuario buscarPorLogin(String login) {
                return this.usuarioDAO.buscarPorLogin(login);
        }

        public void salvar(Usuario usuario) {

                Integer codigo = usuario.getCodigo();
                if (codigo == null || codigo == 0) {
                        this.usuarioDAO.salvar(usuario);
                } else {
                        this.usuarioDAO.atualizar(usuario);
                }
        }
```

# Cadastro de Usuários

```
        public void excluir(Usuario usuario) {
                this.usuarioDAO.excluir(usuario);
        }

        public List<Usuario> listar() {
                return this.usuarioDAO.listar();
        }
}
```

# Cadastro de Usuários

- Modifique o código da classe “UsuarioBean” para o da figura abaixo.
- Add os seguintes imports:
  - import javax.enterprise.context.RequestScoped;
  - import javax.inject.Named;
  - import usuario.Usuario;

```
@Named
@RequestScoped
public class UsuarioBean {
    private Usuario usuario = new Usuario();
    private String confirmarSenha;
    //gerer getters e setters
}
```

# Cadastro de Usuários

- Crie o arquivo “usuario.xhtml” na pasta “Publico” com o código abaixo.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <title>Cadastro de Usuários</title>
    </h:head>
    <h:body>
        <h1>Cadastro de Usuários</h1>
        <h:form id="cadastro">
            <h:messages/>
            <h:inputHidden value="#{usuarioBean.usuario.codigo}"/>
            <h:inputHidden value="#{usuarioBean.usuario.ativo}"/>
            <h:panelGrid columns="2">
                <h:outputLabel value="Nome:" for="nome"/>
                <h:inputText id="nome" label="Nome" value="#{usuarioBean.usuario.nome}" size="30" maxlength="30"
                             required="true" requiredMessage="Você não tem nome?">
                    <f:validateLength minimum="10" maximum="30"/>
                </h:inputText>
```

```
<h:outputLabel value="Data Nascimento:" for="data"/>
<h:inputText id="data" label="Data Nascimento" value="#{usuarioBean.usuario.nascimento}" size="10"
             maxlength="10" required="true">
        <f:convertDateTime dateStyle="medium"/>
</h:inputText>

<h:outputLabel value="Celular:" for="celular"/>
<h:inputText id="celular" label="Celular" value="#{usuarioBean.usuario.celular}" size="10"/>

<h:outputLabel value="e-Mail:" for="email"/>
<h:panelGroup>
        <h:inputText id="email"
                label="e-Mail"
                value="#{usuarioBean.usuario.email}"
                size="50" maxlength="50" required="true"
                validatorMessage="e-Mail inválido">
                <f:validateRegex pattern="[a-zA-Z0-9\-\_\.]+@[a-zA-Z0-9\-\_\.]+"/>
        </h:inputText>
        <h:message for="email"/>
</h:panelGroup>
```

```
                    <h:outputLabel value="Login:" for="login"/>
                    <h:inputText id="login" label="Login" value="#{usuarioBean.usuario.login}" size="15" maxlength="15" required="true"
                                 validatorMessage="Login deve ter no mínimo 5 e no máximo 15 caracters e só pode ter os simbolos '.' e '_'.">
                        <f:validateRegex pattern="([a-z]|[0-9]|[.]|[_]){5,15}"/>
                    </h:inputText>

                    <h:outputLabel value="Senha:" for="senha"/>
                    <h:inputSecret id="senha" label="Senha" value="#{usuarioBean.usuario.senha}" size="10" maxlength="10" required="true"
                                   redisplay="true"/>

                    <h:outputLabel value="Confirmar Senha:" for="confirmarsenha"/>
                    <h:inputSecret id="confirmarsenha" label="Confirmar Senha" value="#{usuarioBean.confirmarSenha}" size="10" maxlength="10"
                                   required="true" redisplay="true"/>
                </h:panelGrid>

                <h:commandButton action="#{usuarioBean.salvar}" value="Salvar"/>
            </h:form>
        </h:body>
</html>
```

# Cadastro de Usuários

- Adicione o método novo e salvar no código da classe UsuarioBean.

```
public String novo() {
            this.usuario = new Usuario();
            this.usuario.setAtivo(true);
            return "usuario";
    }
public String salvar() {
            FacesContext context = FacesContext.getCurrentInstance();

            String senha = this.usuario.getSenha();
            if (!senha.equals(this.confirmarSenha)) {
                    FacesMessage facesMessage = new FacesMessage("A senha não foi confirmada corretamente");
                    context.addMessage(null, facesMessage);
                    return null;
            }
            UsuarioRN usuarioRN = new UsuarioRN();
            usuarioRN.salvar(this.usuario);

            return "usuarioSucesso";
    }
```

# Cadastro de Usuários

- Crie o arquivo “login.xhtml” na pasta “Publico”.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Login</title>
    </h:head>
    <h:body>
        <h1> Login</h1>
        <hr/>
        <h:form>
            <h:commandButton action="#{usuarioBean.novo}" value="Registre-se"/>
        </h:form>
        <hr/>
    </h:body>
</html>
```

# Cadastro de Usuários

- Crie o arquivo “usuarioSucesso.xhtml” na pasta “Publico”.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Cadastro de Usuários</title>
    </h:head>
    <h:body>
      <h1><h:outputText value="#{usuarioBean.usuario.nome}"/>, seja Bem-vindo</h1>

                Seu cadastro foi efetuado com sucesso.
                <br/>
                Clique <a href="login.jsf">aqui</a> para efetuar o login.
    </h:body>
</html>
```

# Cadastro de Usuários

- Execute o projeto e verifique se ocorreu tudo ok.
- No projeto a camada de acesso a dados pode ser vista na interface UsuarioDAO e na classe UsuarioDAOHibernate. A camada de regra de negócio é notada pela classe UsuarioRN. As camadas de controle e apresentação são, respectivamente, os Managed Bean (UsuarioBean) e os arquivos XHTML. O POJO é representado pela classe Usuario.
- A tag @org.hibernate.annotations.NaturalID serve para indicar a **chave natural da tabela**, ou seja, no caso do projeto, não haverá login repetido no sistema.

# Cadastro de Usuários

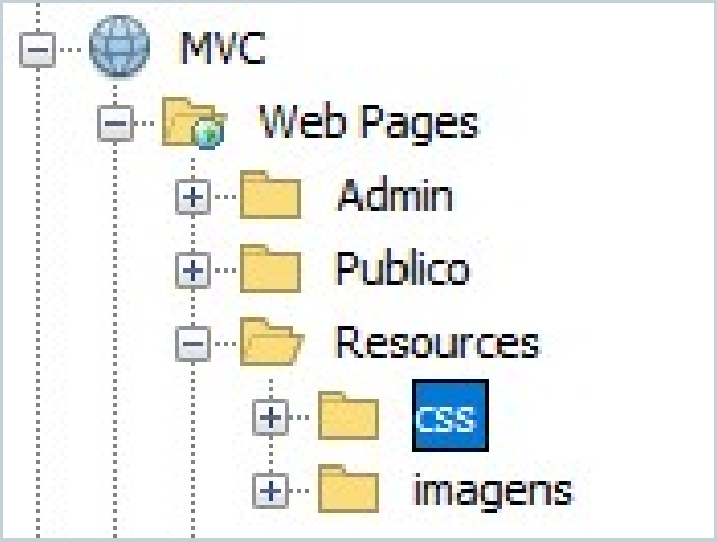
- Na classe UsuarioDAOHibernate:
  - Temos propriedade session do tipo Session e é através dela que conseguimos fazer operações do hibernate chegar ao BD.
  - A tag "this.session.createCriteria(Usuario.class).list()" no método "listar()" realiza uma consulta ao BD retornando todos os registros e colunas da tabela mapeada pela classe "Usuario".
  - Em "buscaPorLogin(String login)" é utilizado o HQL (Hibernate Query Language), que é uma linguaguem semelhante a SQL porem adaptada ao Objeto-Relacional.
- A classe DAOFactory serve como um construtor de DAOs. Ela permite que a classe UsuarioRN obtenha instancia de UsuarioDAO sem saber como funciona internamente.

# Cadastro de Usuários

- A classe “ConexaoHibernateFilter” intercepta todas as requisições que chegam no servidor e define o inicio e fim do processamento de requisições no servidor (Conhecido como uma técnica chamada **Open Session in View** que está documentada no hibernate).
  - No arquivo web.xml do projeto foram passadas, através das tag <filter> e <filter-mapping>, as configurações da classe Filter para que intercepte as requisições.
- Na tag <property name = “current_session_context_class”>thread </property> está determinando que toda sessão aberta do hibernate seja tratada como uma sessão diferente, ou seja, uma nova thread.

# Administração de usuários

- Crie as pastas “Admin”, “resources” e dentro de “resources”, as pastas “css” e “imagens”.



# Administração de usuários

- Crie o arquivo “principal.xhtml” dentro da pasta “Admin” com o seguinte código.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
<h:head>
        <title>Cadastro de Usuários</title>
    </h:head>

    <h:body>
        <h1>Cadastro de Usuários</h1>
        <h:form>
            <h:messages/>
            <h:dataTable value="#{usuarioBean.lista}" var="usuario" rules="rows"
                         cellpadding="5">
                <f:facet name="caption">
                    A listagem abaixo exibe a relação de todos os usuários do sistema.
                    Você poderá realizar ativação e desativação, edição e exclusão para os usuários.
                </f:facet>
                <f:facet name="header">Listagem de Usuários</f:facet>
                <f:facet name="footer">Final da listagem</f:facet>
```

# Administração de usuários

```
<h:column>
    <h:commandLink action="#{usuarioBean.ativar}">
        <h:graphicImage library="imagens" name="usuario_ativo_#{usuario.ativo}.png" style="border:0"/>
        <f:setPropertyActionListener target="#{usuarioBean.usuario}" value="#{usuario}"/>
    </h:commandLink>
</h:column>
<h:column>
    <f:facet name="header">Código</f:facet>
    <h:outputText value="#{usuario.codigo}" />
</h:column>
<h:column>
    <f:facet name="header">Nome</f:facet>
    <h:outputText value="#{usuario.nome}" />
</h:column>
<h:column>
    <f:facet name="header">e-Mail</f:facet>
    <a href="mailto:#{usuario.email}">#{usuario.email}</a>
</h:column>
```

# Administração de usuários

```
                <h:column>
                    <h:commandLink action="#{usuarioBean.editar}">
                        <h:graphicImage library="imagens" name="editar16.png" style="border:0"/>
                        <f:setPropertyActionListener target="#{usuarioBean.usuario}" value="#{usuario}"/>
                        <f:setPropertyActionListener target="#{usuarioBean.destinoSalvar}" value="/admin/principal"/>
                    </h:commandLink>
                </h:column>
                <h:column>
                    <h:commandLink action="#{usuarioBean.excluir}" onclick=
                        "if (!confirm('Confirma a exclusão do usuário #{usuario.nome}?'))return false;">
                        <h:graphicImage library="imagens" name="excluir16.png" style="border:0"/>
                        <f:setPropertyActionListener target="#{usuarioBean.usuario}" value="#{usuario}"/>
                    </h:commandLink>
                </h:column>
            </h:dataTable>
        </h:form>
    </h:body>
</html>
```

# Administração de usuários

- Adicione as variáveis abaixo na classe “UsuarioBean”. Gere o get e set apenas da variável “destinoSalvar”.

```
private List<Usuario> lista;
private String destinoSalvar;
```

# Administração de usuários

- Ainda em “UsuarioBean”, adicione o método “getLista()” abaixo. É ele que retorna a lista de todos os usuários para o dataTable.

```
public List<Usuario> getLista() {
    if (this.lista == null) {
        UsuarioRN usuarioRN = new UsuarioRN();
        this.lista = usuarioRN.listar();
    }
    return this.lista;
}
```

# Administração de usuários

- Ainda em “UsuarioBean”, adicione os métodos abaixo.

```
public String editar() {
    this.confirmarSenha = this.usuario.getSenha();
    return "/Publico/usuario";
}

public String excluir() {
    UsuarioRN usuarioRN = new UsuarioRN();
    usuarioRN.excluir(this.usuario);
    this.lista = null;
    return null;
}
```

# Administração de usuários

- Ainda em “UsuarioBean”, adicione o método “ativar()” abaixo. Esse método ativa e desativa um usuário de utilizar a aplicação.

```
public String ativar() {
    if (this.usuario.isAtivo()) {
        this.usuario.setAtivo(false);
    } else {
        this.usuario.setAtivo(true);
    }

    UsuarioRN usuarioRN = new UsuarioRN();
    usuarioRN.salvar(this.usuario);
    return null;
}
```

# Administração de usuários

- Dentro do método “novo()” adicione a linha abaixo.

```
public String novo() {
    this.destinoSalvar = "usuarioSucesso";
    this.usuario = new Usuario();
    this.usuario.setAtivo(true);
    return "usuario";
}
```

# Administração de usuários

- Modifique o método “salvar()” para o código abaixo.

```
public String salvar() {
    FacesContext context = FacesContext.getCurrentInstance();
    String senha = this.usuario.getSenha();
    if (!senha.equals(this.confirmarSenha)) {
        FacesMessage facesMessage = new FacesMessage("A senha não foi confirmada corretamente");
        context.addMessage(null, facesMessage);
        return null;
    }
    UsuarioRN usuarioRN = new UsuarioRN();
    usuarioRN.salvar(this.usuario);

    return this.destinoSalvar;
}
```

# Administração de usuários

- Adicione o trecho em destaque na página usuario.xhtml.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <title>Cadastro de Usuários</title>
    </h:head>
    <h:body>
        <h1>Cadastro de Usuários</h1>
        <h:form id="cadastro">
            <h:messages/>
            <h:inputHidden value="#{usuarioBean.usuario.codigo}"/>
            <h:inputHidden value="#{usuarioBean.usuario.ativo}"/>
            <h:inputHidden value="#{usuarioBean.destinoSalvar}"/>
            <h:panelGrid ...42 lines />

            <h:commandButton action="#{usuarioBean.salvar}" value="Salvar"/>
        </h:form>
    </h:body>
</html>
```

# Administração de usuários

- Execute a aplicação e acesse diretamente a pagina http://localhost:8080/MVC/Admin/principal.jsf.
- Verifique se está tudo funcionando corretamente.

# Atividades

- Leia o artigo abaixo e faça uma resenha sobre:
  - https://sol.sbc.org.br/index.php/wcama/article/view/6429/6325

# Extras

- https://www.youtube.com/watch?v=APKdyX0xKLQ
- https://www.youtube.com/watch?v=jyTNhT67ZyY
- https://www.youtube.com/watch?v=ZW2JLtX4Dag
- https://www.youtube.com/watch?v=3Rlm70kz0Kc