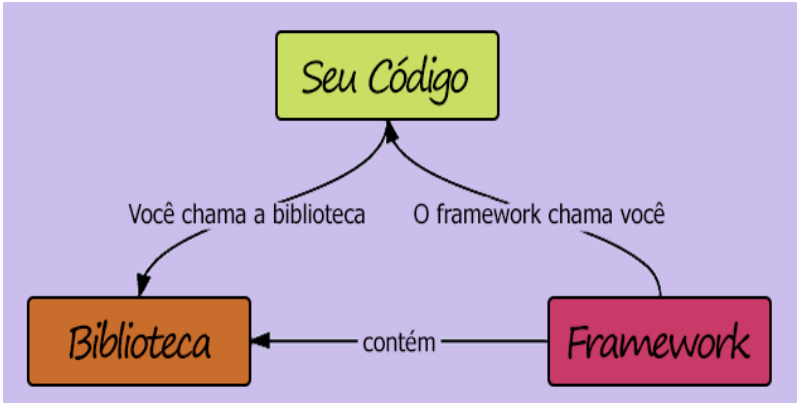
Desenvolvimento Rápido de Aplicações

Java Server Faces - JSF

Profa. Joyce Miranda

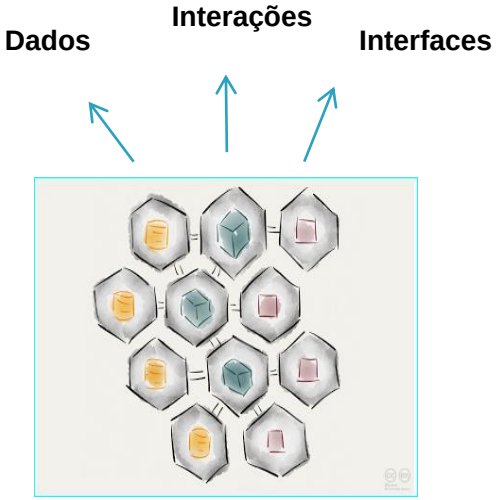
Java Server Faces - JSF

- O que é JSF?
  - Framework MVC para desenvolvimento web em JAVA.
- Detalhando..
  - Framework
    - Tornar mais simples o desenvolvimento de aplicações, disponibilizando uma série de funcionalidades já prontas.
      - Abstrair questões sobre arquitetura da aplicação
      - Manter foco na lógica de negócios



Java Server Faces - JSF

- O que é JSF?
  - Framework MVC para desenvolvimento web em JAVA.
- Detalhando..
  - MVC
    - Padrão de desenvolvimento que organiza o código da aplicação de acordo com responsabilidades específicas.



Java Server Faces - JSF

- O que é JSF?
  - Framework MVC para desenvolvimento web em JAVA.
- Detalhando..
  - **Desenvolvimento web**
    - Aplicação WEB
      - Software projetado para ser executado em navegadores usando a Internet ou Intranets.

Aplicações Desktop

- *Fat Client*: Processamento na Máquina do Cliente

Aplicações WEB

- *Thin Client*: Processamento em um Servidor Web

Java Server Faces - JSF

- O que é JSF?
  - Framework MVC para desenvolvimento web em JAVA.
- Detalhando..
  - Desenvolvimento web
    - Arquitetura Aplicação WEB

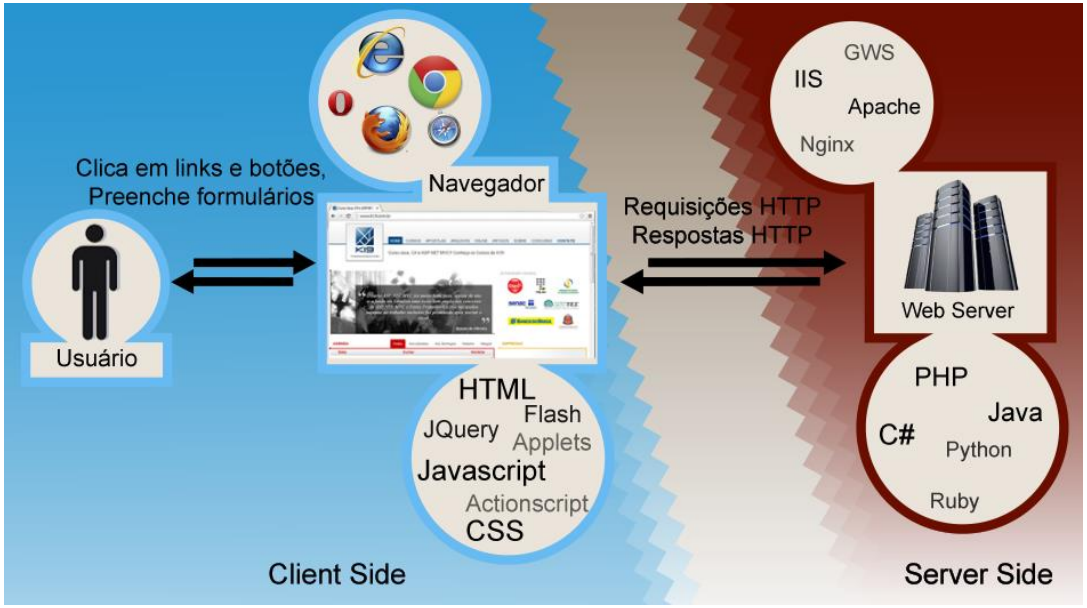
    - Servidor WEB
      - Software ou computador capaz de aceitar pedidos HTTP dos clientes e servi-los com respostas HTTP.

Java Server Faces - JSF

- O que é JSF?
  - Framework MVC para desenvolvimento web em JAVA.
- Detalhando..
  - **Desenvolvimento web**
    - Tecnologias para desenvolvimento WEB
      - Front-End
        - Tecnologias para projetar interfaces e promover interação com o usuário do lado do cliente
      - Back-End
        - Tecnologias para dinamizar o site através de linguagens de programação que rodam do lado do servidor web.

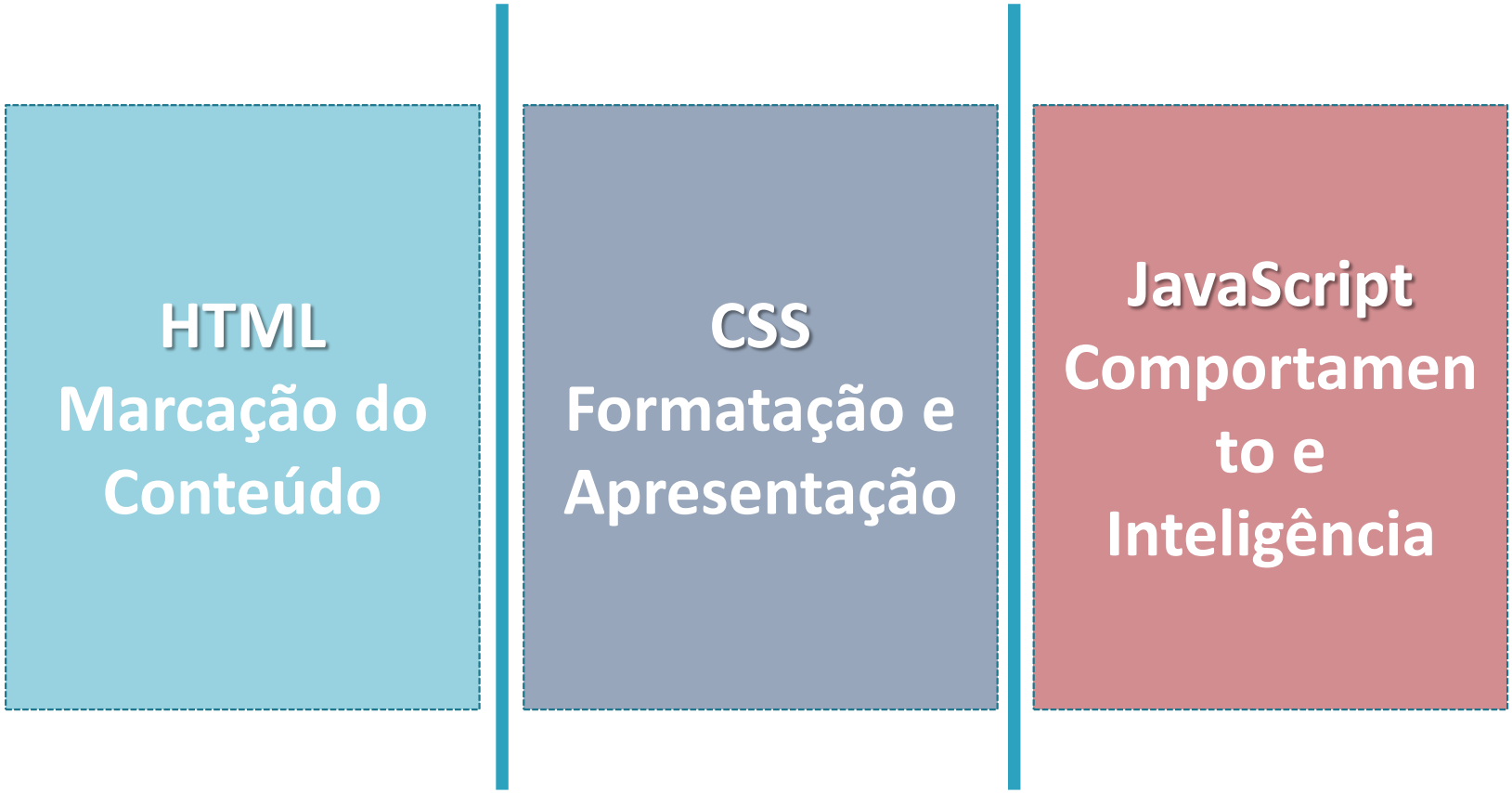
Java Server Faces - JSF

- O que é JSF?
  - Framework MVC para desenvolvimento web em JAVA.
- Detalhando..
  - Desenvolvimento web
    - Arquitetura Aplicação WEB



Java Server Faces - JSF

- Tecnologias Front-End



Java Server Faces - JSF

- Estrutura básica de um documento HTML
  - Os comandos HTML são representados por meio de tags (rótulos).

```
<TAG> Texto marcado pela Tag </TAG>
```

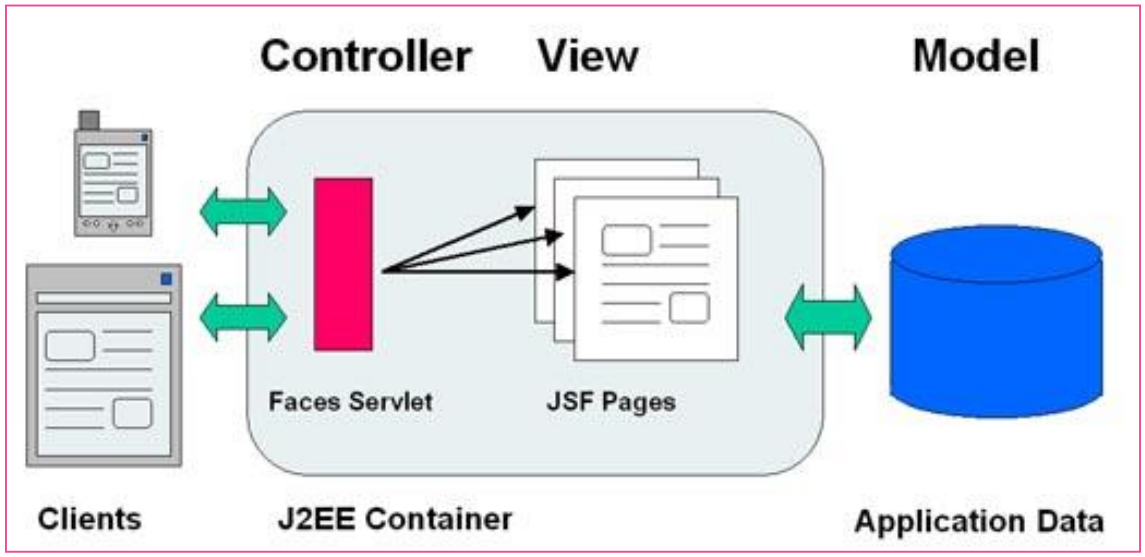
```
1  <!doctype html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>Título da Página</title>
6      </head>
7      <body>
8          <!-- conteúdo da página -->
9      </body>
10 </html>
```

Java Server Faces - JSF

- Características JSF
  - É uma especificação JAVA para a construção de interfaces de usuário baseadas em componentes para aplicações web.
  - *Component-Based*
    - Permite criar aplicações utilizando componentes visuais pré-definidos.
      - Capaz de criar uma interface web mais rica e menos complexa de ser implementada.
    - Adota um modelo de programação dirigido a eventos.
      - As ações dos componentes são baseadas em eventos.
      - Os componentes encapsulam detalhes do protocolo HTTP (requisiçãoxresposta).

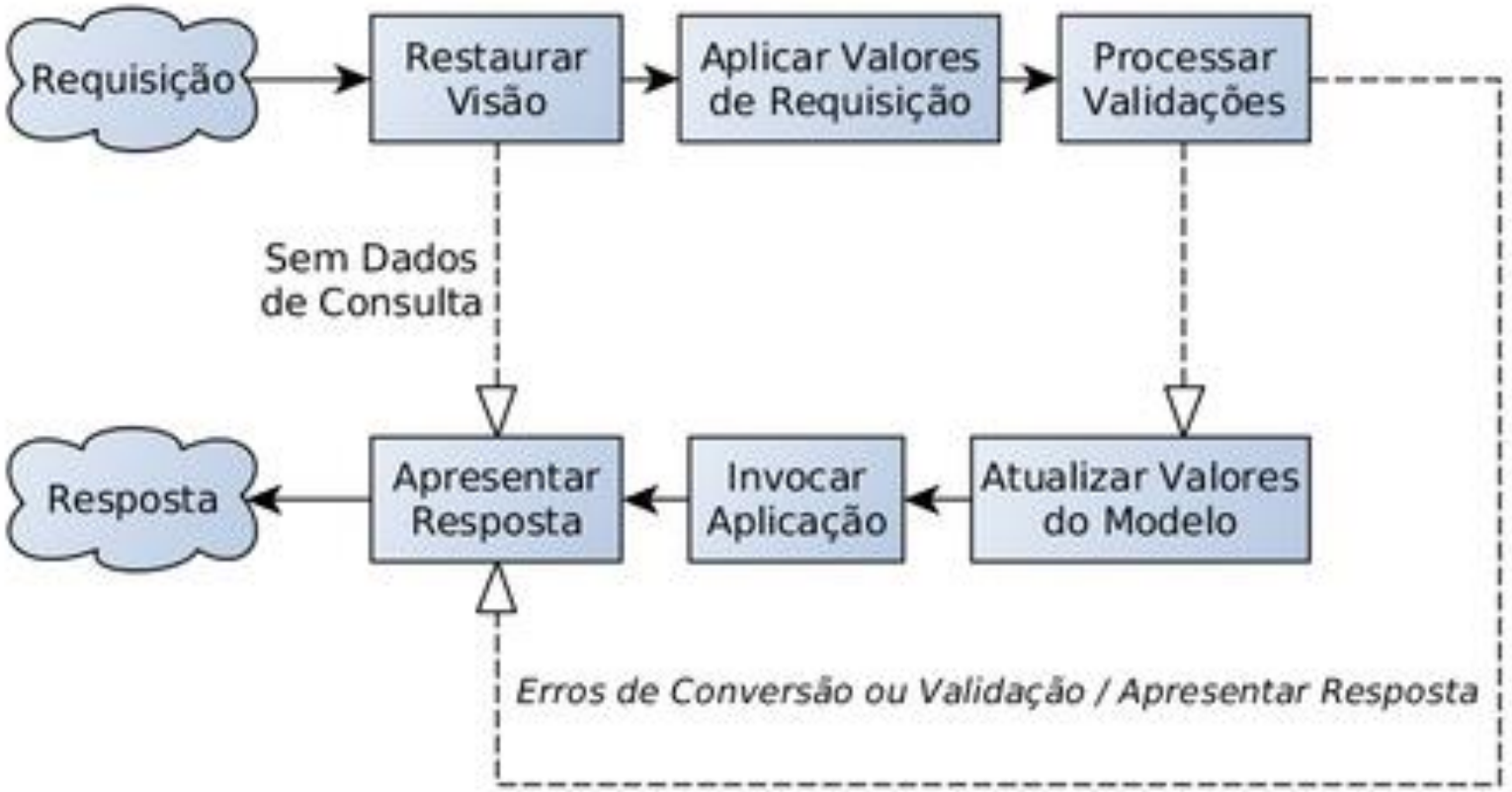
Java Server Faces - JSF

- Características JSF
  - O JSF é fortemente baseado nos padrões *MVC* e *Front Controller*.
  - *Front Controller*
    - Todas as requisições são recebidas pelo mesmo componente.
      - Em uma aplicação JSF, toda requisição é recebida pelo Faces Servlet.



Java Server Faces - JSF

- Faces Servlet
  - Processamento de Requisição – Ciclo de Vida



Java Server Faces - JSF

- *Managed Beans*
  - Classes da camada Model que interagem com componentes JSF.
  - São responsáveis por:
    - Fornecer dados que serão exibidos nas telas.
    - Receber os dados enviados nas requisições.
    - Executar tarefas de acordo com as ações dos usuários.
  - Os componentes JSF interagem com as classes *Managed Beans* por meio do acesso a seus métodos.
  - São POJOS anotados com *@ManagedBean*
    - POJO (Plain Old Java Object)
      - Classe JAVA que expõe atributos por meio de métodos *get* e set.

Java Server Faces - JSF

- *Managed Beans*
  - Criação
    - Criar uma classe Java com a anotação *@ManagedBean*.

```
@ManagedBean
public class TesteBean {
   ...
}
```

    - Por padrão, o JSF assume que o nome do *Managed Bean* é o nome da classe com a primeira letra minúscula.
    - Para o exemplo acima, o nome do *Managed Bean* é *testeBean*.

Java Server Faces - JSF

- *Managed Beans*
  - Propriedades

```
@ManagedBean
public class TesteBean {
  private int numero;

  public int setNumero(int numero) {
    this.numero = numero;
  }

  public int getNumero() {
    return numero;
  }
}
```

  - Recuperando e exibindo valores na página da aplicação
    - *Expression Language* (**#{}**)

```
Valor: #{testeBean.numero}
```

Java Server Faces - JSF

- Estrutura Básica de uma página JSF
  - As telas são definidas em arquivos XHTML (HTML + XML)
  - Os componentes visuais que constituem as telas são adicionados por meio de *tags*.
  - A especificação do JSF define uma grande quantidade de *tags* e as classifica em bibliotecas.
    - *HTML* (http://java.sun.com/jsf/html)
      - Construção de telas gerando o HTML adequado
    - *Core* (http://java.sun.com/jsf/core)
      - Tratadores de eventos e validadores
    - *Facelets* (http://java.sun.com/jsf/facelets)
      - Utilização de *templates* e reutilização de código.

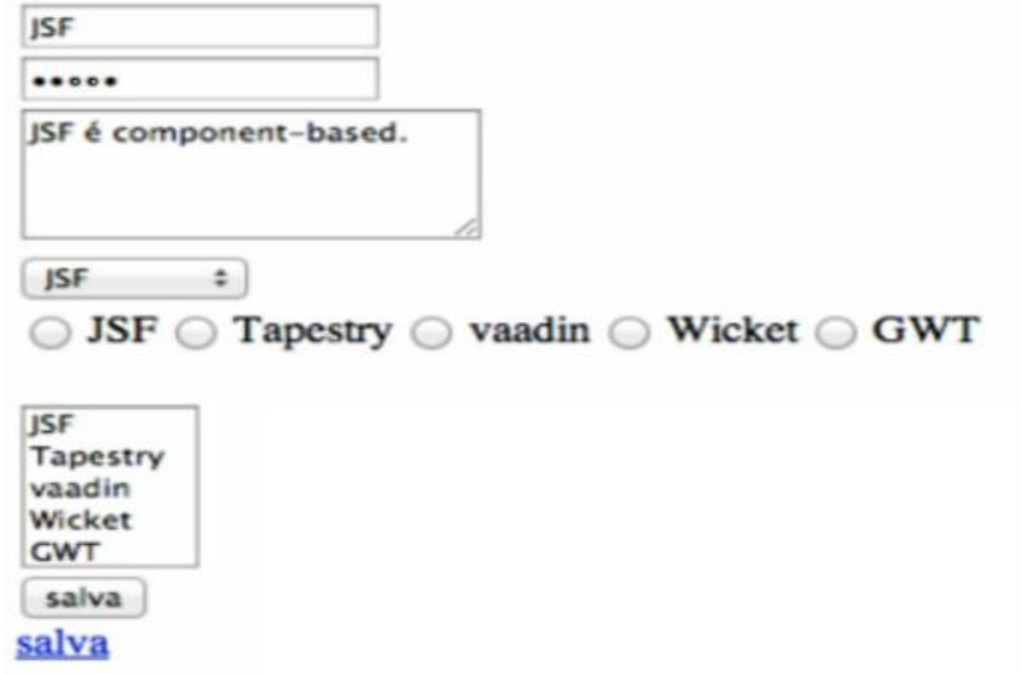
Java Server Faces - JSF

- Estrutura Básica de uma página JSF
  - Importando *tags* de componentes
    - As bibliotecas de tags que serão utilizadas para construir a página devem ser "importadas" através do pseudo-atributo xmlns aplicado à tag <html>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

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

<h:head>
  <title>K19 Treinamentos</title>
</h:head>

<h:body>
  <h:outputText value="Estrutura básica de uma tela JSF" />
</h:body>
</html>
```

Java Server Faces - JSF

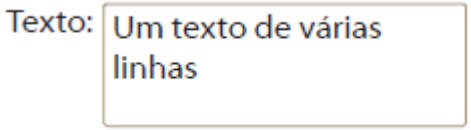
- Componentes Visuais
  - **Formulários**: **<h:form>**
    - Comporta componentes que permitem a interação com os usuários:
      - caixas de texto, caixas de seleção, rótulos, botões e links.



Java Server Faces - JSF

- Componentes Visuais
  - **Caixas de Texto**
    - **<h:inputText>**
      - Permite que o usuário digite uma linha de texto.

    - **<h:inputTextarea>**
      - Permite que o usuário digite várias linhas de texto.



    - **<h:inputSecret>**
      - Oculta o que foi digitado.

Java Server Faces - JSF

- Componentes Visuais
  - **Caixas de Texto**
    - *Binding*
      - Associação de uma caixa de texto a uma propriedade de um *Managed Bean* através do atributo *value*.

```
@ManagedBean
public class TesteBean {
  private int numero;

  public int setNumero(int numero) {
    this.numero = numero;
  }

  public int getNumero() {
    return numero;
  }
}
```

```
<h:inputText value="#{testeBean.numero}" />
```

Java Server Faces - JSF

- Componentes Visuais
  - **Rótulos <h:outputLabel>**
    - Indica o que deve ser preenchido em uma caixa de texto.

```
<h:outputLabel value="Nome: " for="nome"/>
<h:inputText id="nome"/>
```

Java Server Faces - JSF

- Componentes Visuais
  - **Textos h:outputText**
    - Apresentação de conteúdo textual.

```
<h:outputText value="Curso: #{curso.sigla} - #{curso.descricao}"/>
```

```
@ManagedBean
public class OlaMundoBean {

  public String getHorario() {
    SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");
    return "Atualizado em " + sdf.format(new Date());
  }
}
```

```
<h:outputText value="#{olaMundoBean.horario}" />
```

Java Server Faces - JSF

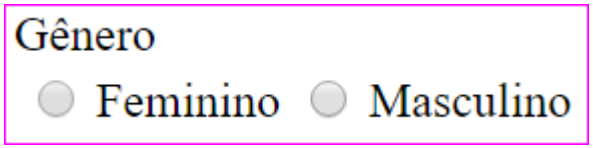
- Componentes Visuais
  - **Textos Formatados <h:outputFormat>**
    - Concatenação de texto

```
<h:outputFormat value="Preço do produto {0}: R$ {1}">
  <f:param value="#{lojaBean.produto.nome}"/>
  <f:param value="#{lojaBean.produto.preco}"/>
</h:outputFormat>
```

Java Server Faces - JSF

- Componentes Visuais
  - **Campos de Seleção**
    - **<h:selectOneRadio>**

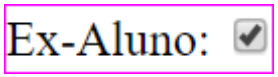
```
<h:outputLabel value="Gênero " for="genero" />
<h:selectOneRadio value="#{componentesBean.genero}" id="genero" >
    <f:selectItem itemValue ="F" itemLabel="Feminino" />
    <f:selectItem itemValue ="M" itemLabel="Masculino" />
</h:selectOneRadio>
```



Java Server Faces - JSF

- Componentes Visuais
  - **Campos de Seleção**
    - **<h:selectBooleanCheckbox>**
      - Seleção Binária

```
<h:outputLabel value="Ex-Aluno: " for="exAluno" />
<h:selectBooleanCheckbox value="#{componentesBean.exAluno}" id="exAluno" />
```



- Caixas desse tipo devem ser vinculadas a propriedades booleanas.

Java Server Faces - JSF

- Componentes Visuais
  - **Campos de Seleção**
    - **<h:selectManyCheckbox>**
      - Seleção Múltipla
        - Opção Estática

```
<h:selectManyCheckbox value="#{componentesBean.cursosEscolhidosE}" id="cursosE" >
    <f:selectItem itemValue ="C1" itemLabel="POO - Estático" />
    <f:selectItem itemValue ="C2" itemLabel="DRA - Estático" />
</h:selectManyCheckbox>
```

Java Server Faces - JSF

- Componentes Visuais
  - **Campos de Seleção**
    - **<h:selectManyCheckbox>**
      - Seleção Múltipla
        - Opção Dinâmica

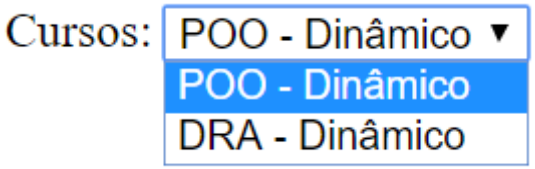
```
<h:selectManyCheckbox
    value="#{componentesBean.cursosEscolhidosD}" id="cursosD" >
    <f:selectItems value="#{componentesBean.cursos}"
        var="curso" itemValue="#{curso.codigo}" itemLabel="#{curso.nome}"/>
</h:selectManyCheckbox>
```

Cursos:
☐ POO - Dinâmico ☐ DRA - Dinâmico

Java Server Faces - JSF

- Componentes Visuais
  - **Campos de Seleção**
    - **<h:selectOneMenu>**
      - Seleção Única
        - Opção Dinâmica

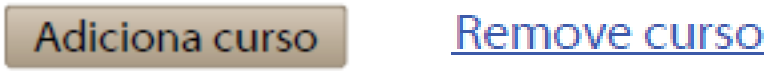
```
<h:selectOneMenu
    value="#{componentesBean.cursosEscolhidos}" id="cursosSOM" >
    <f:selectItems value="#{componentesBean.cursos}"
        var="curso" itemValue="#{curso.codigo}" itemLabel="#{curso.nome}"/>
</h:selectOneMenu>
```



Java Server Faces - JSF

- Componentes Visuais
  - **Botões e Links**
    - **<h:commandButton> e <h:commandLink>**
      - Enviam os dados de um formulário HTML para o servidor através do método POST do HTTP.
      - Podem ser associados a métodos de ação de um *Managed Bean* através dos atributos *action* e *actionListener.*

```
<h:commandButton value="Adiciona curso" action="#{cursosBean.adicionaCurso}"/>
<h:commandLink value="Remove curso" action="#{cursosBean.removeCurso}"/>
```



Java Server Faces - JSF

- Componentes Visuais
  - **Botões e Links**
    - **<h:commandButton> e <h:commandLink>**
      - *action*: deve ser usado para executar uma lógica de negócio ou indicar navegação entre páginas.

```
@ManagedBean
public class TesteBean {
  private int numero;

  public void incrementaNumero() {
    this.numero = numero + 1;
  }

  // GETTERS E SETTERS
}
```

```
<h:form>
  <h:commandButton value="Incrementa" action="#{testeBean.incrementaNumero}" />
</h:form>
```

Java Server Faces - JSF

- Componentes Visuais
  - **Botões e Links**
    - **<h:commandButton> e <h:commandLink>**
      - *action*: deve ser usado para executar uma lógica de negócio ou indicar navegação entre páginas.

```
<h:commandButton value="Salva" action="#{produtoBean.salva}" />
```

```
@ManagedBean
public class ProdutoBean {
  private Produto produto = new Produto();
  private List<Produto> produtos = new ArrayList<Produto>();

  public String salva() {
    this.produtos.add(this.produto);
    this.produto = new Produto();
    return "lista-produtos";
  }

  // GETTERS E SETTERS
}
```

Java Server Faces - JSF

- Componentes Visuais
  - **Botões e Links**
    - **<h:commandButton> e <h:commandLink>**
      - *actionListener*: executar uma lógica relacionada a *view* ou disparar uma ação antes de uma lógica de negócio.

```
<h:commandButton value="Clique Aqui" actionListener="#{cliqueBean.mudaTexto}"/>
```

```
@ManagedBean
public class CliqueBean {
  public void mudaTexto(ActionEvent e) {
    UICommand c = (UICommand) e.getComponent();
    c.setValue("Clicado");
  }
}
```

Java Server Faces - JSF

- Componentes Visuais
  - **Botões e Links**
    - **<h:button> e <h:link>**
      - Realizam requisições HTTP do tipo GET.
      - Utiliza o atributo *outcome* para definir a página de destino.
      - Esses componentes não são utilizados para submeter formulários HTML.

```
<h:button value="Lista de cursos" outcome="lista-cursos" />
<h:link value="Home" outcome="home" />
```

Java Server Faces - JSF

- Componentes Visuais
  - **Botões e Links**
    - **<h:outputLink>**
      - Permite apenas a criação de links HTML que realizam requisições do tipo GET.
      - A URL da requisição é definida explicitamente no atributo *value*.

```
<h:outputLink  value="http://www.ifam.edu.br">
    Página do IFAM
</h:outputLink>
```

Página do IFAM

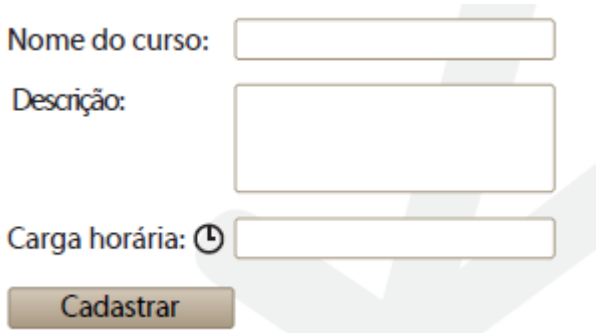
Java Server Faces - JSF

- Componentes Visuais
  - **Imagens**
    - **<h:graphicImage>**

```
<h:graphicImage value="img/download.png" width="50px" height="50px"  />
```

Java Server Faces - JSF

- Componentes Visuais
  - **Componentes de Organização**
    - **<h:panelGrid>**
      - Organiza os elementos em uma grade.
    - **<h:panelGroup>**
      - Permite que diversos componentes sejam tratados como um único componente.




```
<h:form>
    <h:panelGrid columns="2">
        <h:outputLabel value="Nome do curso:" for="nome"/>
        <h:inputText id="nome"/>

        <h:outputLabel value ="Descrição:" for="descricao"/>
        <h:inputTextarea id="descricao"/>

        <h:panelGroup >
            <h:outputLabel value="Carga horária : " for ="carga-horaria"/>
            <h:graphicImage name ="imgs/download.png" />
        </h: panelGroup >
        <h:inputText id="carga-horaria"/>

        <h:commandButton value="Cadastrar"/>
    </h:panelGrid >
</h:form>
```

Java Server Faces - JSF

- Componentes Visuais
  - **Tabelas**
    - **<h:dataTable>**
      - Atributo value: associa uma lista de elementos à tabela

Código	Nome	Adicionar turma
0	POO - Dinâmico	Adicionar turma
1	DRA - Dinâmico	Adicionar turma

```
<h:dataTable value="#{componentesBean.cursos}" var="curso">

<h:column>
   <f:facet name="header">Código </f:facet> #{curso.codigo}
</h:column>

<h:column>
   <f:facet name="header">Nome </f:facet> #{curso.nome}
</h:column>

<h:column>
   <f:facet name="header">Adicionar turma </f:facet>
   <h:commandLink value=" Adicionar turma " action="#{componentesBean.adicionarTurma(curso)}" />
</h:column >

</h:dataTable >
```

Java Server Faces - JSF

- Praticando...
  - Crie um projeto JSF de forma a garantir que o código abaixo seja devidamente interpretado em um navegador web.

```
<h:form>
    <h:outputLabel for="nome" value="Digite seu nome:"/>
    <h:inputText id="nome" value="#{helloJSF.nome}"/>
    <h:commandButton value="Exibe Mensagem" action="#{helloJSF.exibeMensagem}"/>
    <h:outputLabel for="nome" value="#{helloJSF.mensagem}"/>
</h:form>
```

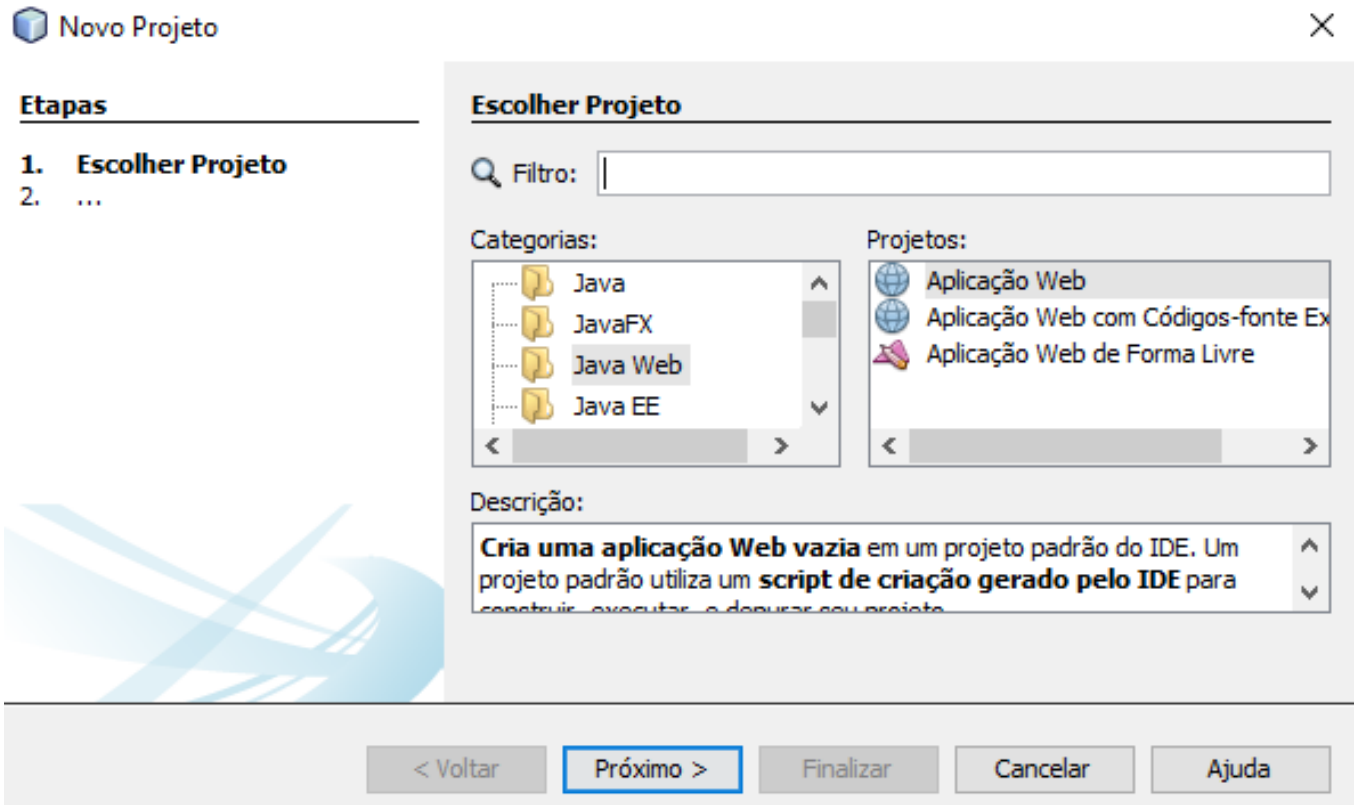
Java Server Faces - JSF

- Praticando...
  - Criando *Managed Bean*: HelloJSF

```
@ManagedBean
public class HelloJSF {

    private String nome;
    private String mensagem;

    public String getNome() {...3 linhas }
    public void setNome(String nome) {...3 linhas }
    public String getMensagem() {...3 linhas }
    public void setMensagem(String mensagem) {...3 linhas }

    public void exibeMensagem(){
        mensagem = "Olá " + this.nome;
    }

}
```

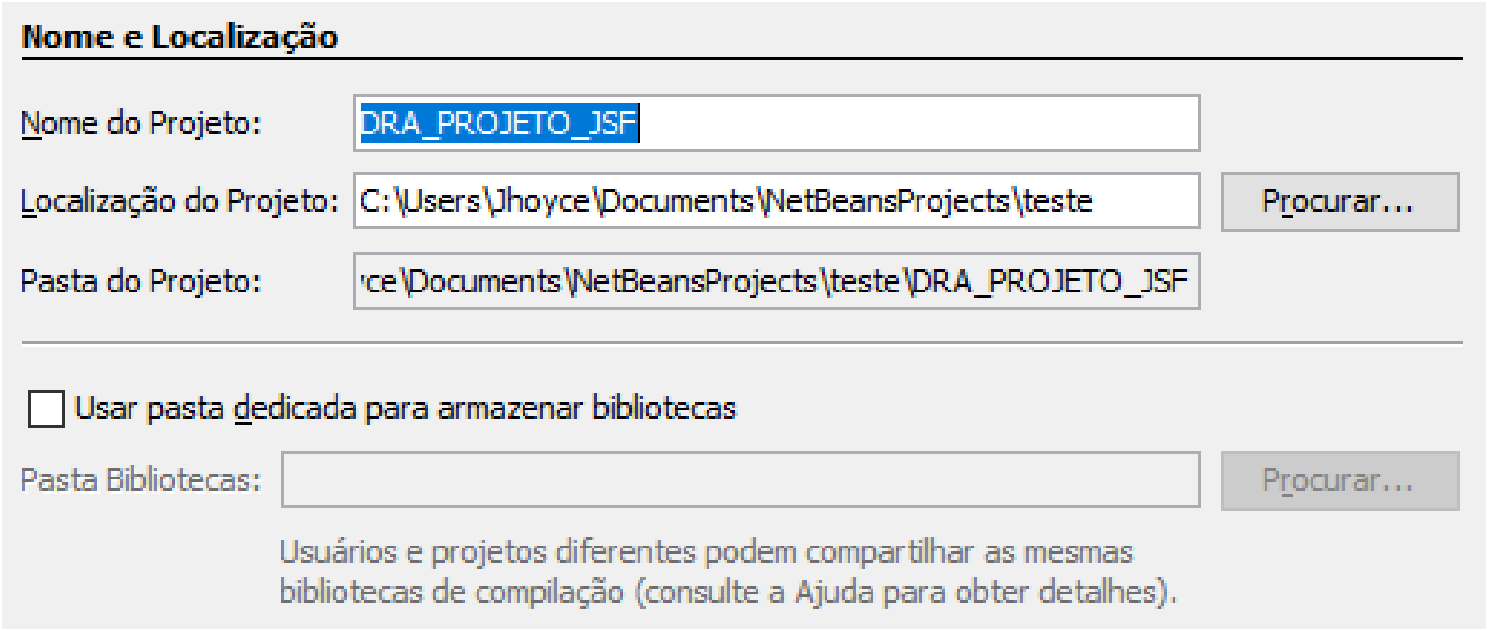
Java Server Faces - JSF

- Praticando...
  - Criando um Projeto WEB



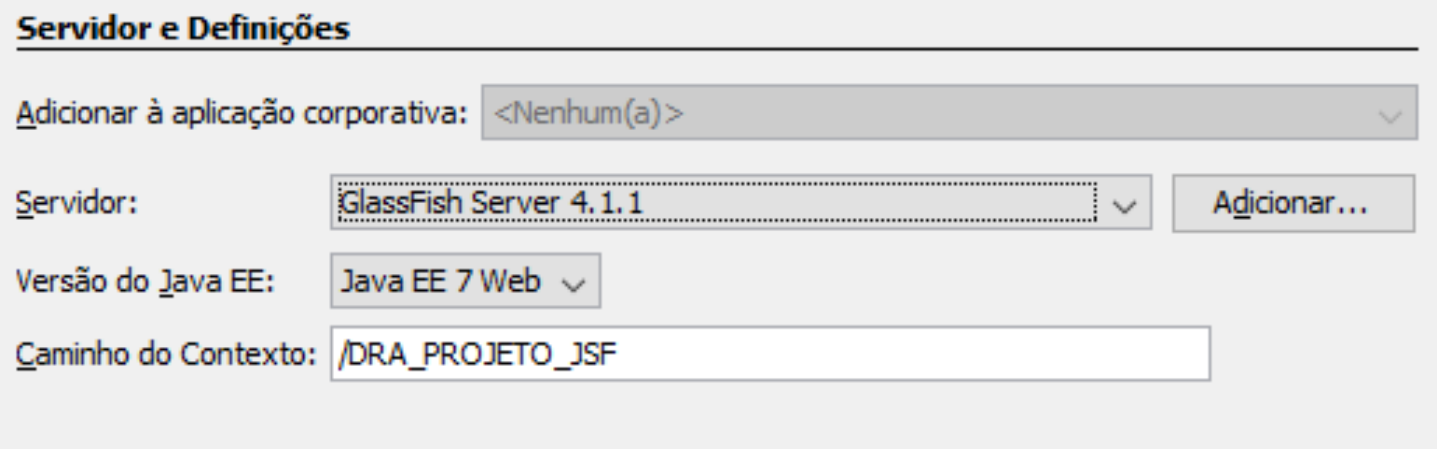
Java Server Faces - JSF

- Praticando...
  - Definindo Nome do Projeto



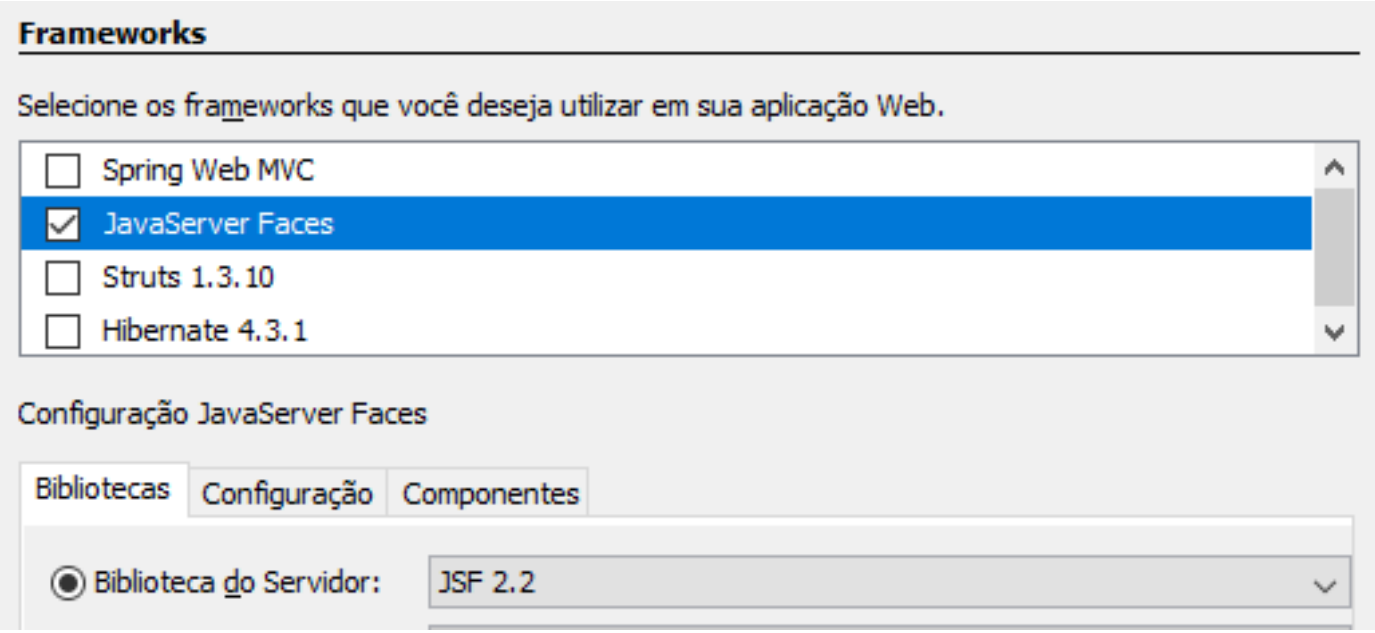
Java Server Faces - JSF

- Praticando...
  - Definindo Servidor



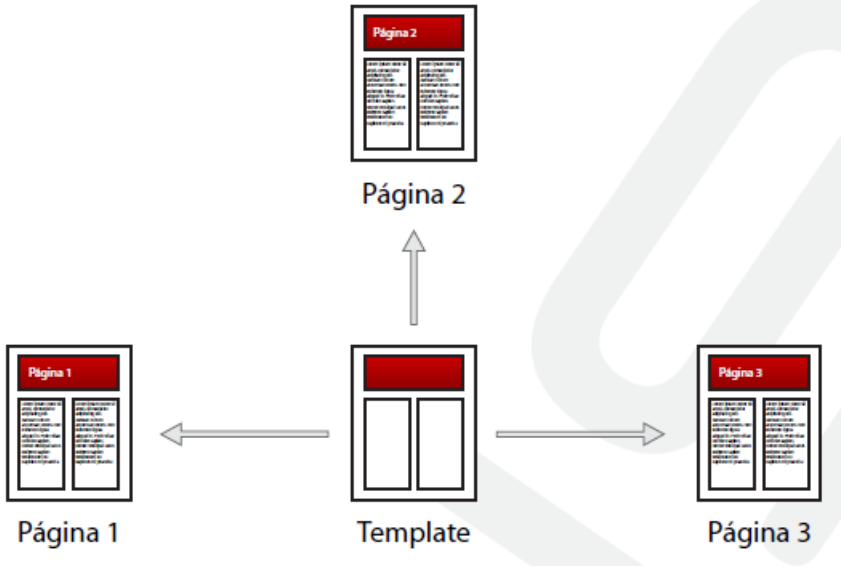
Java Server Faces - JSF

- Praticando...
  - Escolhendo Framework JSF



Java Server Faces - JSF

- Reutilização de Código
  - *Facelets*
    - Projeto definido para facilitar o processo de desenvolvimento e manutenção/gerenciamento das telas de uma aplicação JSF
  - *Templates*
    - Reutilização de código de Telas



Java Server Faces - JSF

- *Templates*
  - Definindo um *template.*
    - Criar um arquivo XHTML e definir o posicionamento dos trechos estáticos e dinâmicos.
    - Importar biblioteca de *tags* do *Facelets*

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
```

Java Server Faces - JSF

- *Templates*
  - Definindo um *template.*
    - **<ui:insert>**
      - Indica o posicionamento dos trechos dinâmicos.

```
<h:body>
    <div id="header">
        <hr/>******Cabeçalho******
    </div>

    <ui:insert name="conteudo">
        Espaço para o conteúdo da tela
    </ui:insert>

    <div id="footer" style="text-align:center">
        <hr/>******Rodapé*******
    </div>
</h:body>
```

template.xhtml

Java Server Faces - JSF

- *Templates*
  - Utilizando um *template*.
    - **<ui:composition>**
      - Indica o posicionamento dos trechos dinâmicos.

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

 <ui:define name ="conteudo">
    <h:form>
    <h:outputLabel value ="Nome:" for ="campo-nome" />
    <h:inputText id="campo-nome" />
    <h:commandButton value="Enviar" />
    </h:form>
 </ui:define>

 </ui:composition>
```

******Cabeçalho******

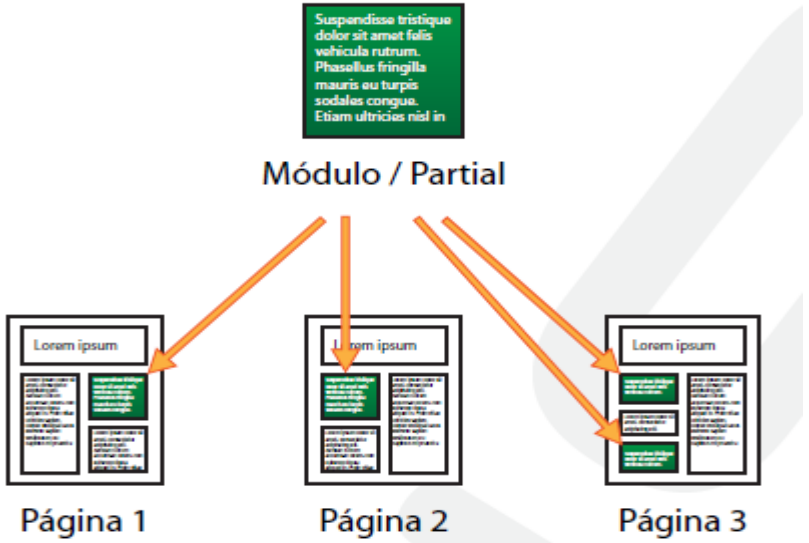
Nome: Enviar

******Rodapé******

telaUsandoTemplate.xhtml

Java Server Faces - JSF

- Modularização
  - Os trechos estáticos ou dinâmicos definidos em um *template* possuem posição fixa.
  - Em determinadas situações, é necessário tornar flexível o posicionamento de um determinado trecho.
  - Nessas situações aplica-se o conceito de Modularização.



Java Server Faces - JSF

- Modularização
  - 1. Coloca-se o conteúdo do módulo em um arquivo XHTML separado.

formulario-de-contato.xhtml

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

  <h:form>
    <h:panelGrid>
      <h:outputLabel value="Mensagem: " for="mensagem"/>
      <h:inputTextarea id="mensagem" value="#{mensagensBean.mensagem}"/>
      <h:commandButton value="Enviar" action="#{mensagensBean.envia}"/>
    </h:panelGrid>
  </h:form>
</ui:composition>
```

  - 2. Faz-se a inclusão do arquivo XHTML onde se fizer necessário.

```
<ui:include src="/formulario-de-contato.xhtml"/>
```

Java Server Faces - JSF

- Modularização
  - Passando parâmetros entre módulos.

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

  <h:dataTable value="#{livros}" var="livro">
    ...
  </h:dataTable>
</ui:composition>
```

lista-livros.xhtml

```
<ui:include src="/lista-livros.xhtml">
  <ui:param name="livros" value="#{livrosBean.listaDosLivrosMaisVendidos}"/>
</ui:include>
```

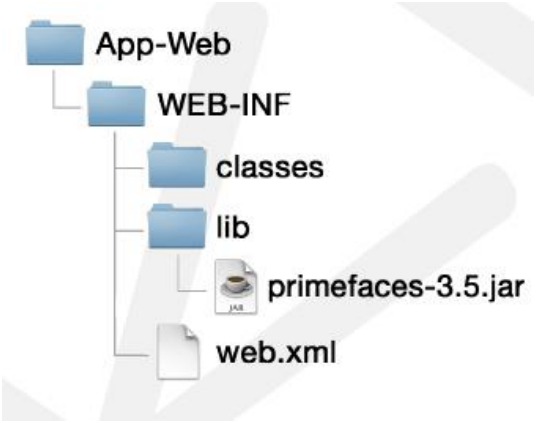
lista-livros-maisVendidos.xhtml

```
<ui:include src="/lista-livros.xhtml">
  <ui:param name="livros" value="#{livrosBean.listaDosLivrosMaisCaros}"/>
</ui:include>
```

lista-livros-maisCaros.xhtml

Java Server Faces - JSF

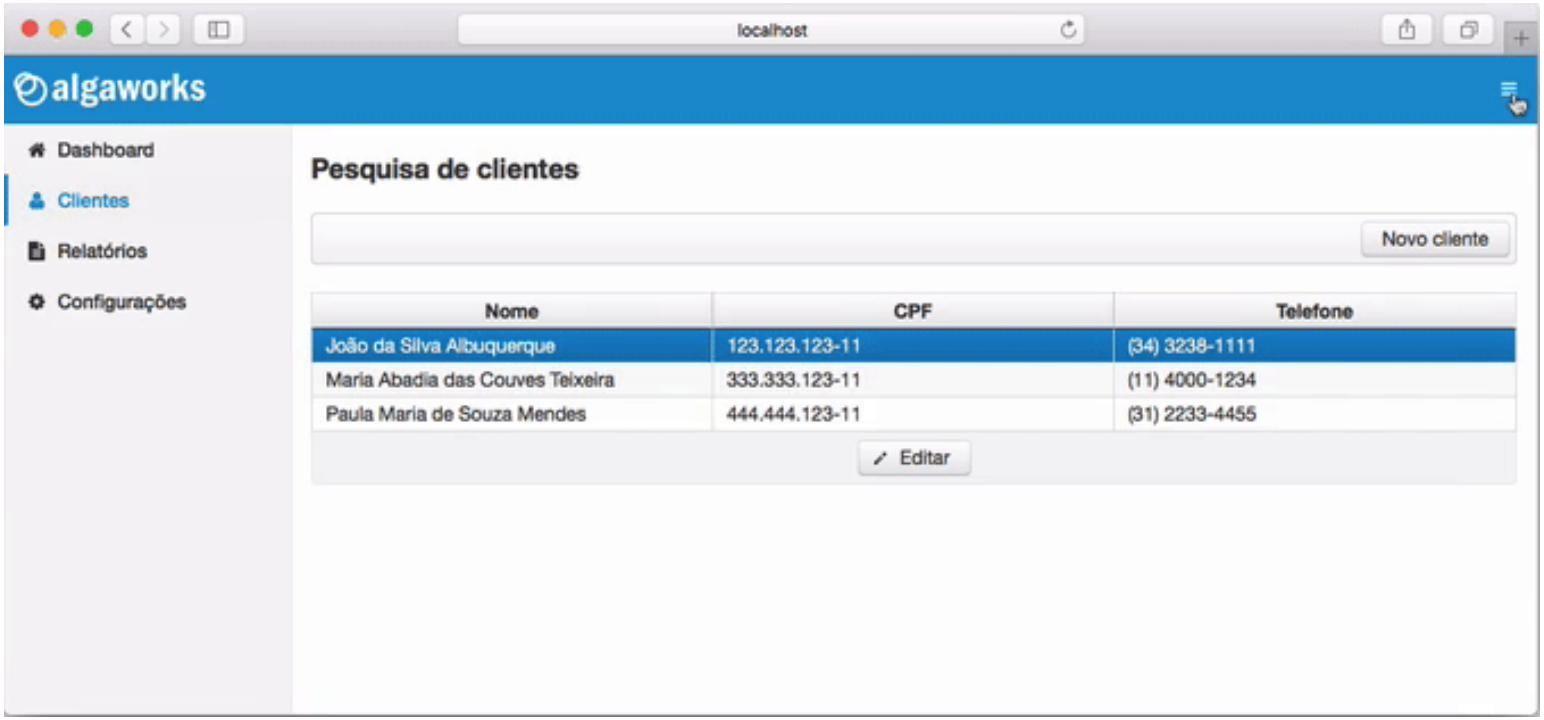
- PrimeFaces
  - Biblioteca de componentes ricos para aplicações criadas com Java Server Faces.
  - Instalação




```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

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

<h:head>
  <title>K19 Treinamentos</title>
</h:head>

<h:body>
  ...
</h:body>
</html>
```

Java Server Faces - JSF

- PrimeFaces
  - Responsividade



Java Server Faces - JSF

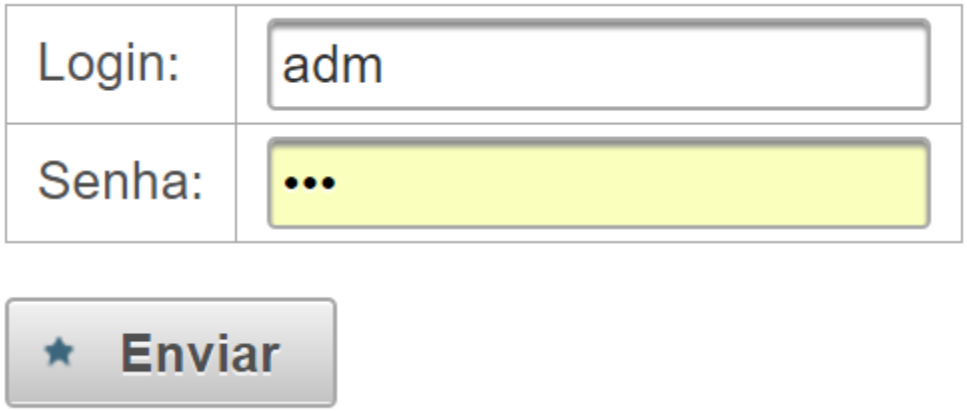
- PrimeFaces
  - https://www.primefaces.org/showcase/

Java Server Faces - JSF

- Desenvolvimento com JSF + PrimeFaces + JPA
  - Tutorial adaptado do conteúdo disponível em:
    - http://www.devmedia.com.br/java-web-criando-uma-tela-de-login-com-jpa-jsf-primefaces-e-mysql/32456
  - Pré-requisito
    - Serviço do MySQL ativo.
    - *Schema* de banco de dados criado (Ex. sysControleAcademico)
    - Tabela criada no *schema* de banco de dados referenciado (Ex. usuario)

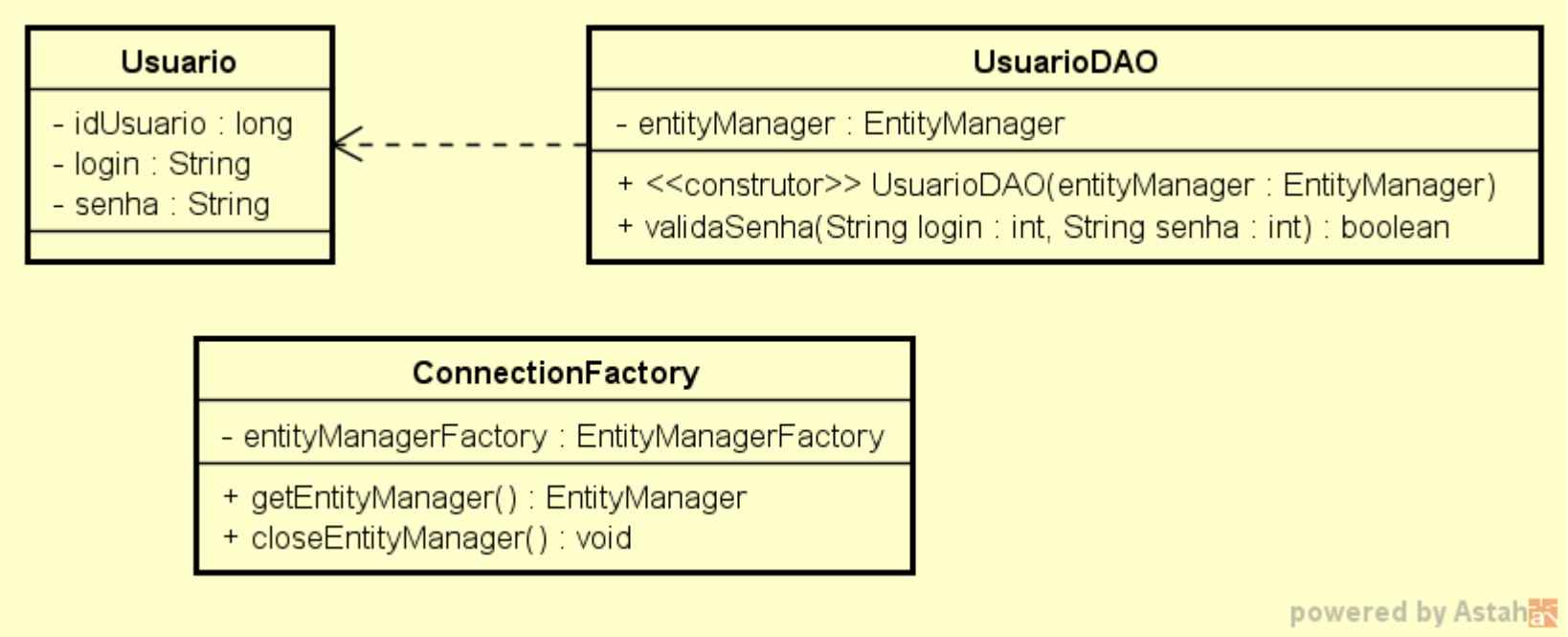
Java Server Faces - JSF

- Desenvolvimento com JSF + PrimeFaces + JPA
  - Formulário de Login



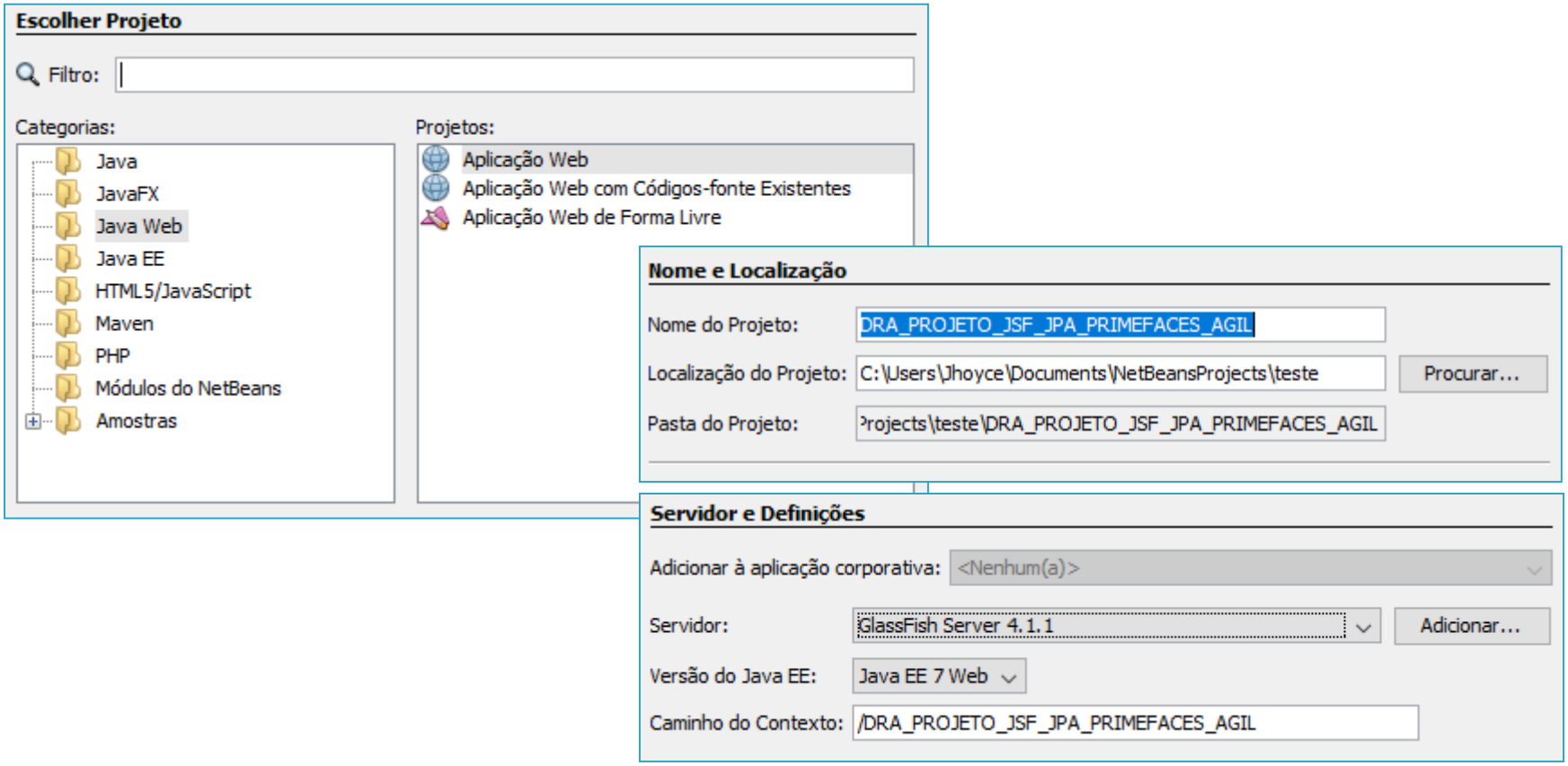
Java Server Faces - JSF

- Desenvolvimento com JSF + *PrimeFaces* + JPA
  - Formulário de *Login*
    - Classes Principais



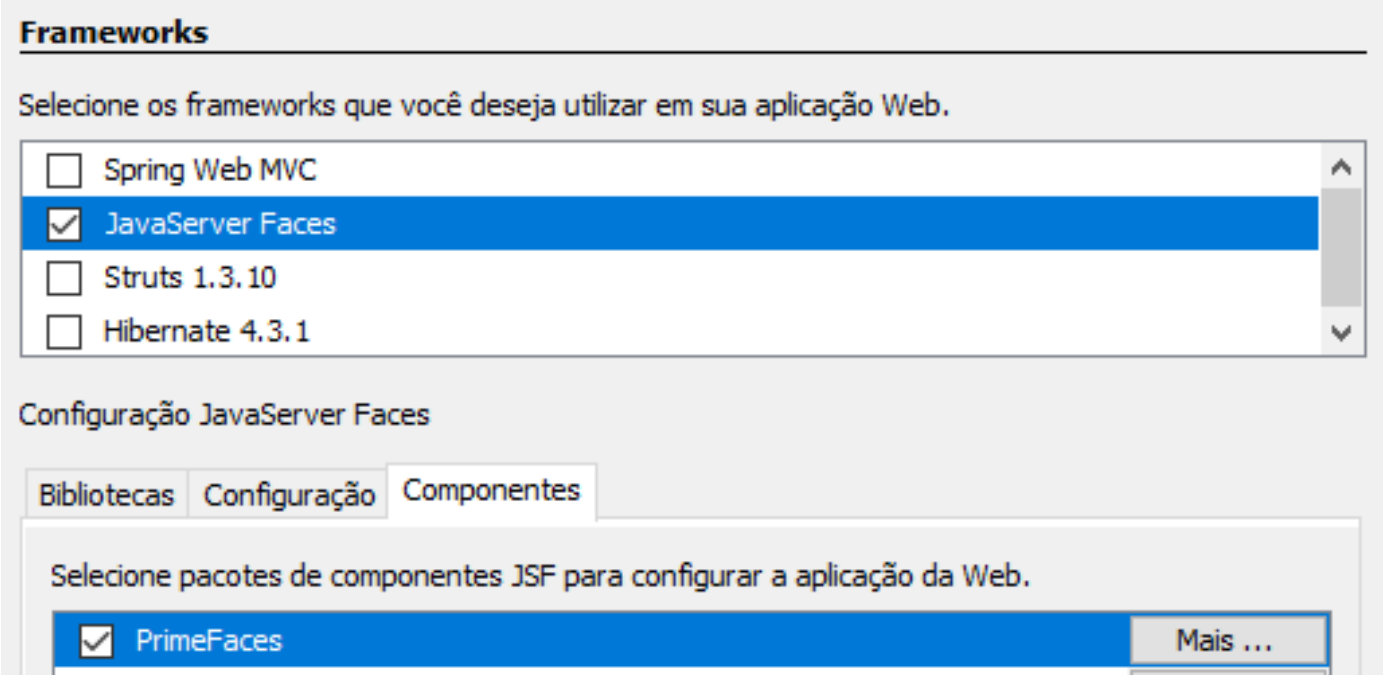
Java Server Faces - JSF

- Desenvolvimento com JSF + *PrimeFaces* + JPA
  - Criando o Projeto



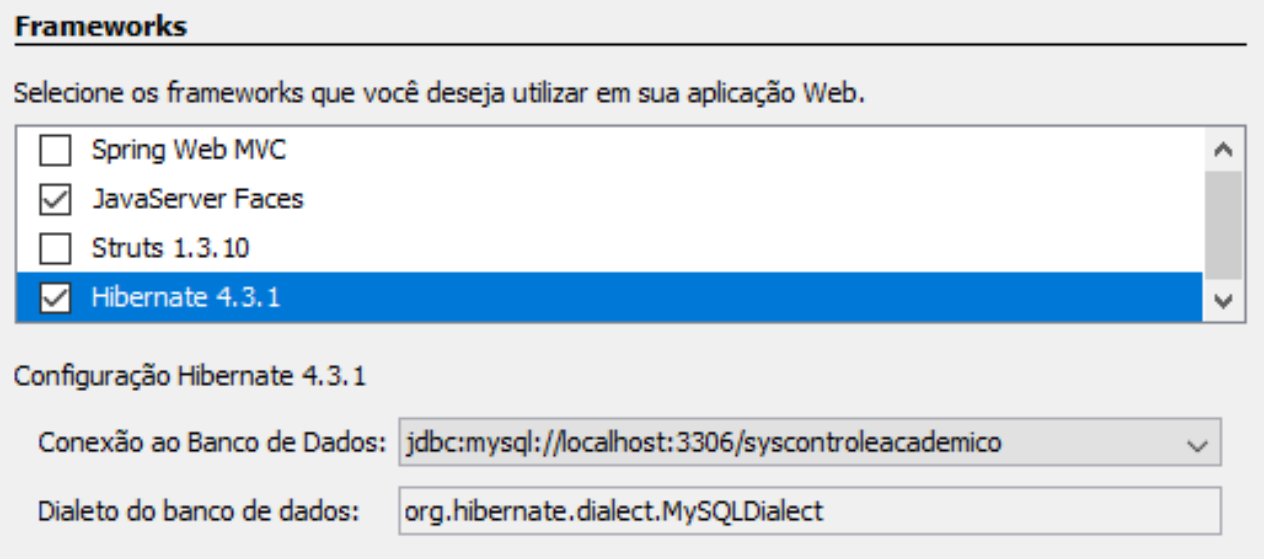
Java Server Faces - JSF

- Desenvolvimento com JSF + PrimeFaces + JPA
  - Criando o Projeto



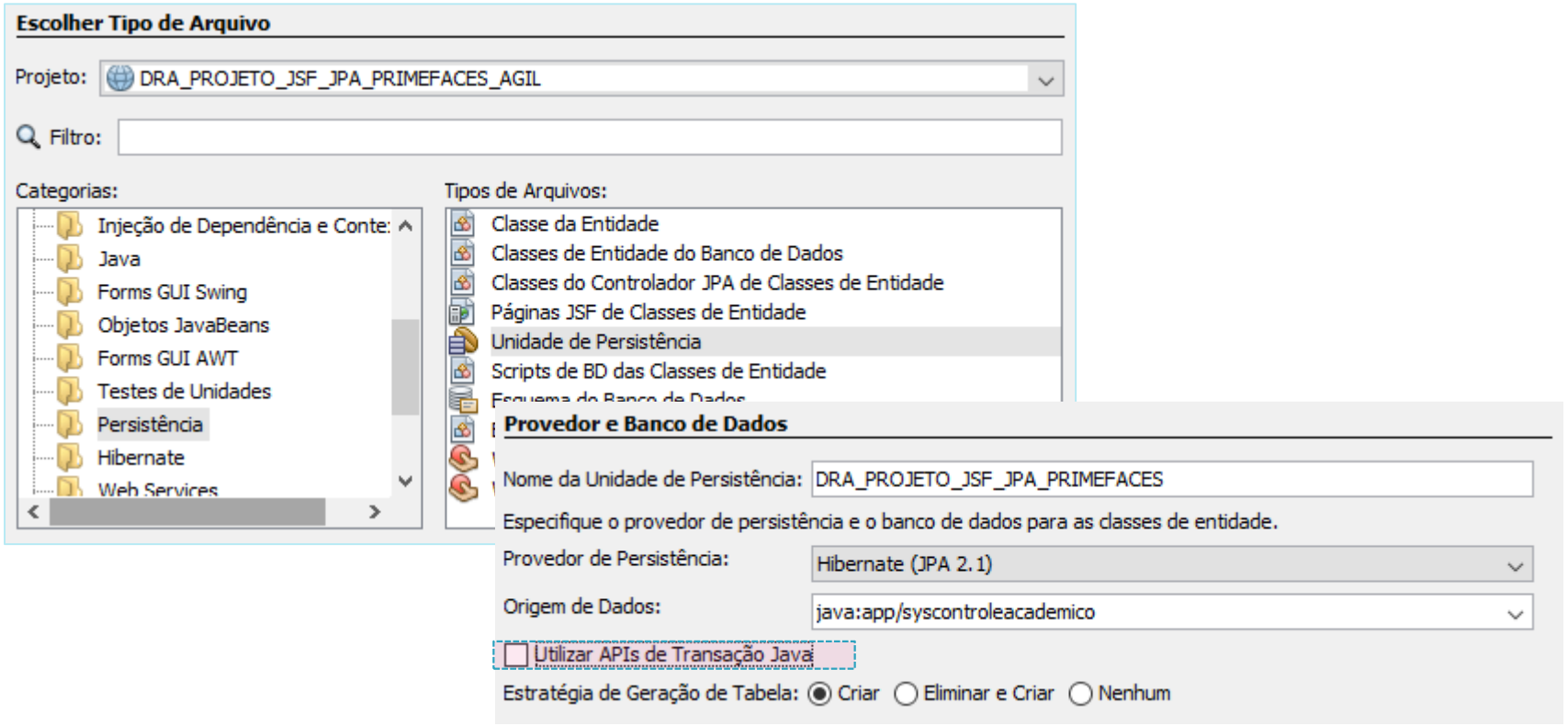
Java Server Faces - JSF

- Desenvolvimento com JSF + PrimeFaces + JPA
  - Criando o Projeto



Java Server Faces - JSF

- Desenvolvimento com JSF + PrimeFaces + JPA
  - Criando Unidade de persistência (persistence.xml)



Java Server Faces - JSF

- Desenvolvimento com JSF + PrimeFaces + JPA
  - Criando Unidade de persistência (persistence.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="DRA_PROJETO_JSF_JPA_PRIMEFACES"
                    transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <non-jta-data-source>java:app/syscontroleacademico</non-jta-data-source>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties/>
  </persistence-unit>
</persistence>
```

Java Server Faces - JSF

ConnectionFactory
- entityManagerFactory : EntityManagerFactory
+ getEntityManager( ) : EntityManager + closeEntityManager( ) : void

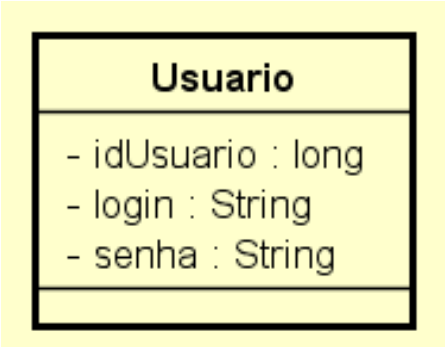
- Desenvolvimento com JSF + PrimeFaces + JPA
  - Criando Classe de Conexão (*ConnectionFactory.java*)

```
public class ConnectionFactory {

    private static EntityManagerFactory entityManagerFactory;

    public static EntityManager getEntityManager(){
        if(entityManagerFactory==null|| !entityManagerFactory.isOpen()){
            System.out.println("Criando EntityManager....");
            entityManagerFactory =
            Persistence.createEntityManagerFactory("DRA_PROJETO_JSF_JPA_PRIMEFACES");
        }
        return entityManagerFactory.createEntityManager();
    }

    public static void closeEntityFactory(){
        entityManagerFactory.close();
    }

}
```

Java Server Faces - JSF

- Desenvolvimento com JSF + *PrimeFaces* + JPA
  - Criando Classe de Entidade (Usuario.java)

```
@Entity
public class Usuario {
    @Id
    @GeneratedValue
    private Long idUsuario;
    @Column(unique=true)
    private String login;
    private String senha;

    //gets e sets
```

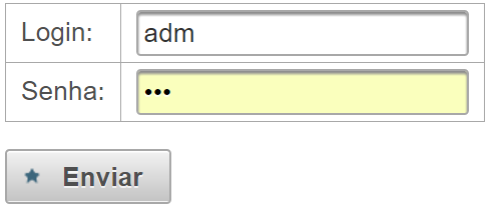


UsuarioDAO
- entityManager : EntityManager
+ <<construtor>> UsuarioDAO(entityManager : EntityManager) + validaSenha(usuario : Usuario) : boolean

- Desenvolvimento com JSF + *PrimeFaces* + JPA
  - Criando Classe de Entidade DAO (UsuarioDAO.java)

```
public class UsuarioDAO {

    private final EntityManager entityManager ;

    UsuarioDAO(EntityManager entityManager){
        this.entityManager = entityManager;
    }

    public boolean validaSenha(Usuario usuario){
        //monta consulta
        TypedQuery<Long> query = entityManager.createQuery(
            "select count(u) from Usuario as u "
              + "where u.login = :login "
              + "  and u.senha = :senha ", Long.class);
        query.setParameter("login", usuario.getLogin());
        query.setParameter("senha", usuario.getSenha());
        long quantidade = query.getSingleResult();
        if(quantidade > 0){return true;}else{return false;}
    }
}
```

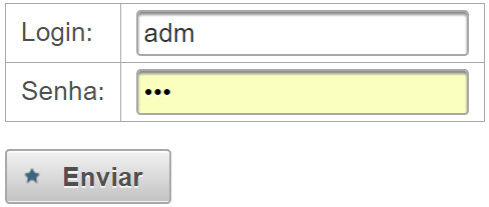
Java Server Faces - JSF



- Desenvolvimento com JSF + *PrimeFaces* + JPA
  - ManagedBean (ValidaSenhaBean.java)

```
@ManagedBean
public class ValidaSenhaBean {

    private Usuario usuario = new Usuario() ;
    private UsuarioDAO dao =  new UsuarioDAO(ConnectionFactory.getEntityManager());

    public String validaSenha(){

        boolean validou = dao.validaSenha(usuario);
        if(validou){
            return "/index";
        }else{
            FacesContext.getCurrentInstance().addMessage(
               null,
               new FacesMessage(FacesMessage.SEVERITY_ERROR,
                    "Usuário não encontrado!",
                    "Erro no Login!"));
                return "/faces/formLogin";
        }
    }

    public Usuario getUsuario() {

    public void setUsuario(Usuario usuario) {

}
```

Java Server Faces - JSF

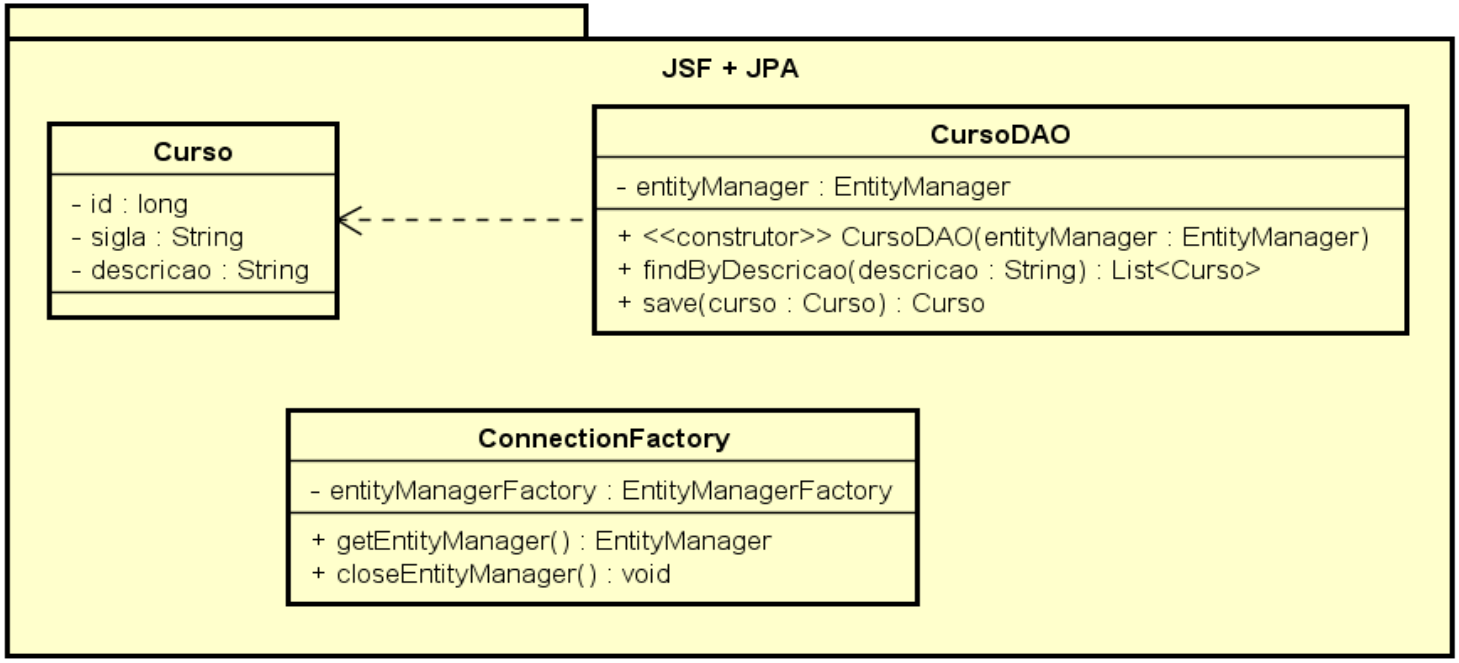


- Desenvolvimento com JSF + *PrimeFaces* + JPA
  - Criando a interface gráfica (formLogin.xhtml)

```
<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">
  <h:head><title>Página JSF</title></h:head>
  <h:body>
    <h:form>
        <p:messages id="messages" />
        <p:panelGrid columns="2">
            <p:outputLabel for="login" value="Login:" />
            <p:inputText id="login" value="#{validaSenhaBean.usuario.login}" />

            <p:outputLabel for="senha" value="Senha:" />
            <p:password id="senha" value="#{validaSenhaBean.usuario.senha}" />

            <p:commandButton value="Enviar" icon="ui-icon-star"
                             action="#{validaSenhaBean.validaSenha}" ajax="false">
            </p:commandButton>
        </p:panelGrid>
    </h:form>
   </h:body>
</html>
```

  - Em caso de erro na execução (jandex): Importar biblioteca *jandex.jar*
  - https://www.youtube.com/watch?v=GRYkZR2SxfU

- Tarefa de Implementação
  - Desenvolvimento com JSF + *PrimeFaces* + JPA
    - Criar telas de listagem de cursos e cadastro de cursos.

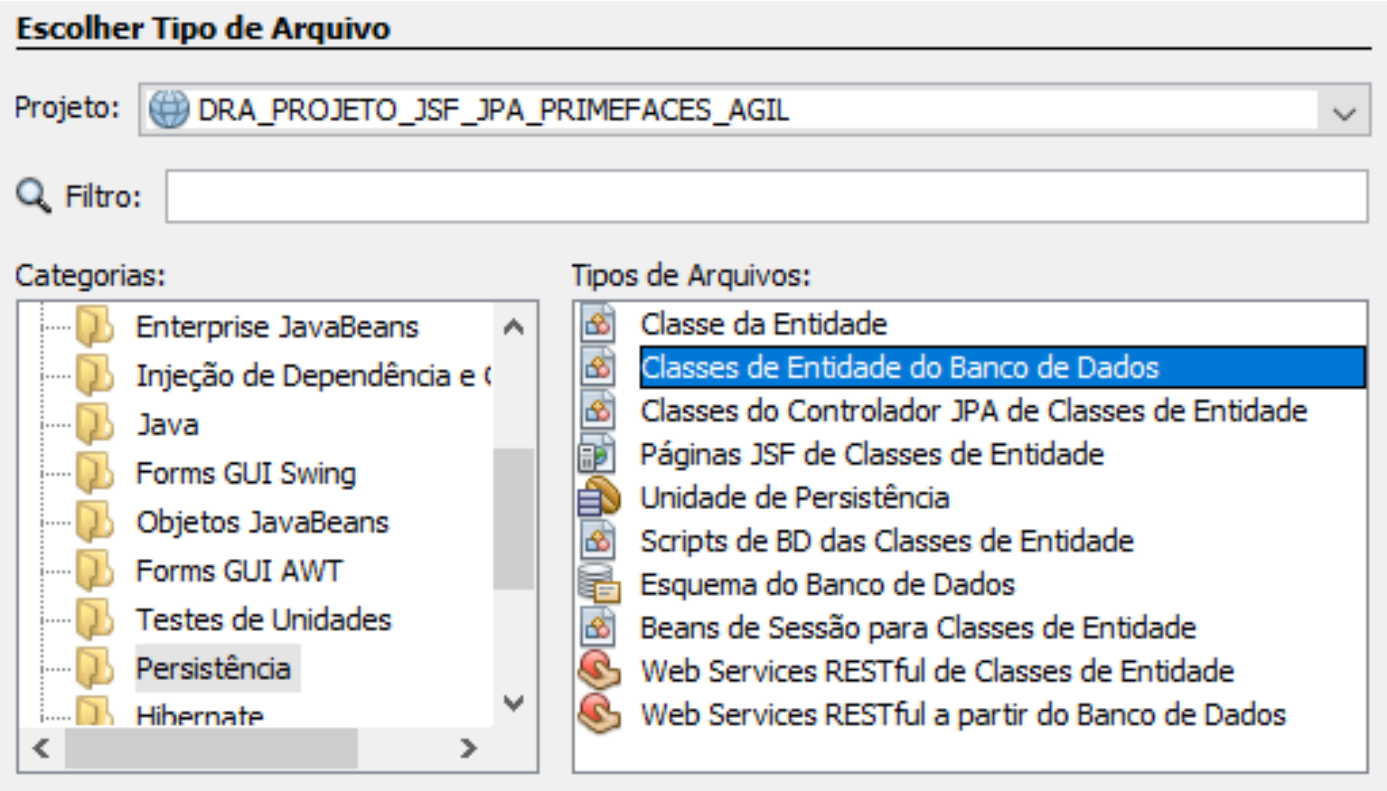


Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Tutorial adaptado do conteúdo disponível em:
    - https://www.scribd.com/document/104338116/Tutorial-Projeto-Web-usando-JSF-com-CRUD-em-JPA-Netbeans-7
  - Pré-requisito
    - Serviço do MySQL ativo.
    - *Schema* de banco de dados criado (Ex. sysControleAcademico)
    - Tabela criada no *schema* de banco de dados referenciado (Ex. curso)

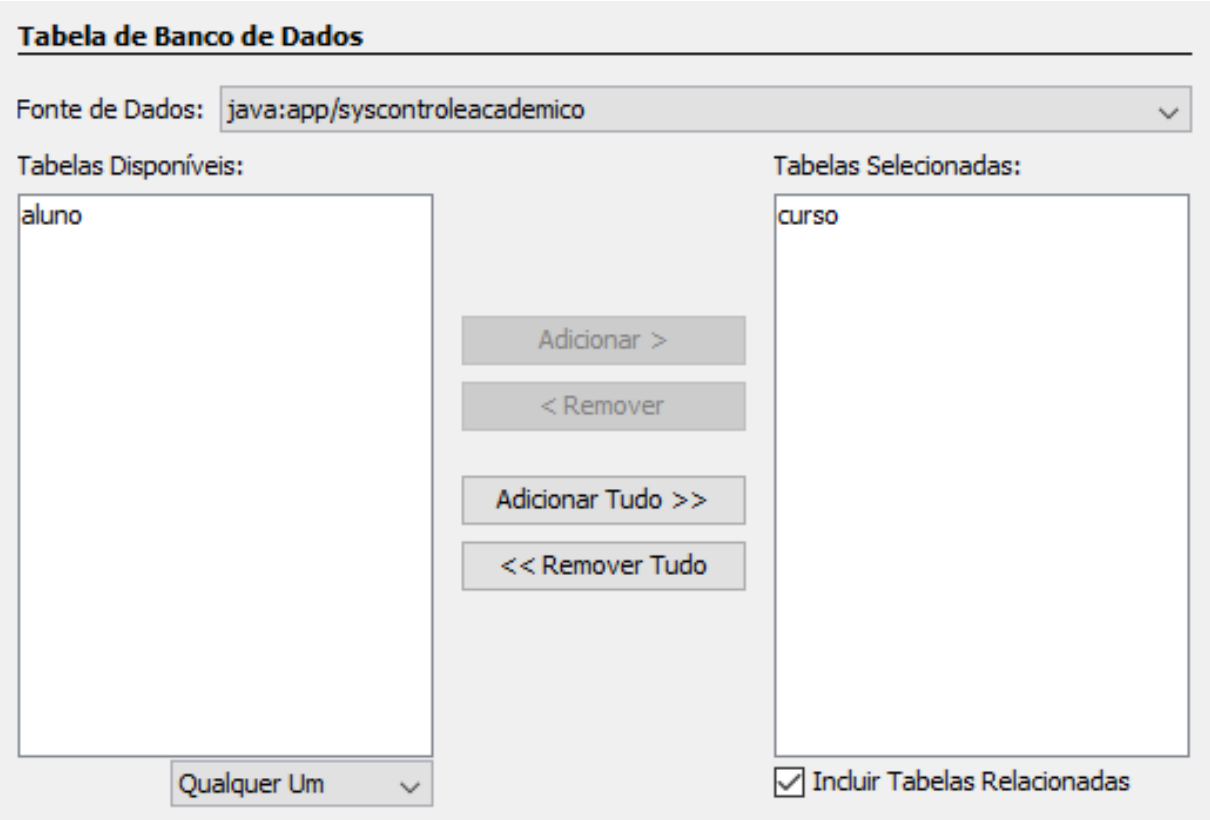
Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Criando Classe de Entidade do Banco de Dados



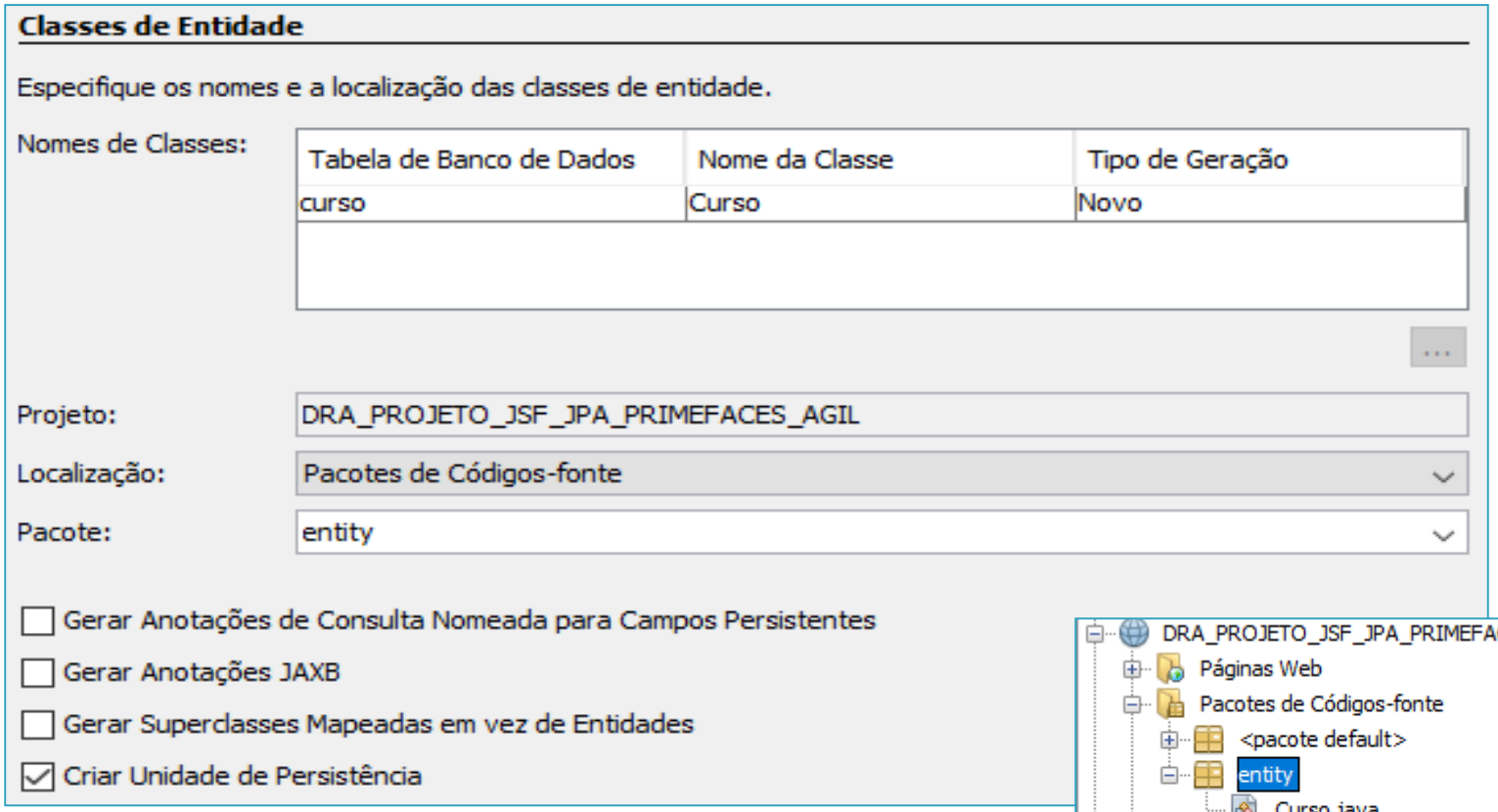
Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Criando Classe de Entidade do Banco de Dados



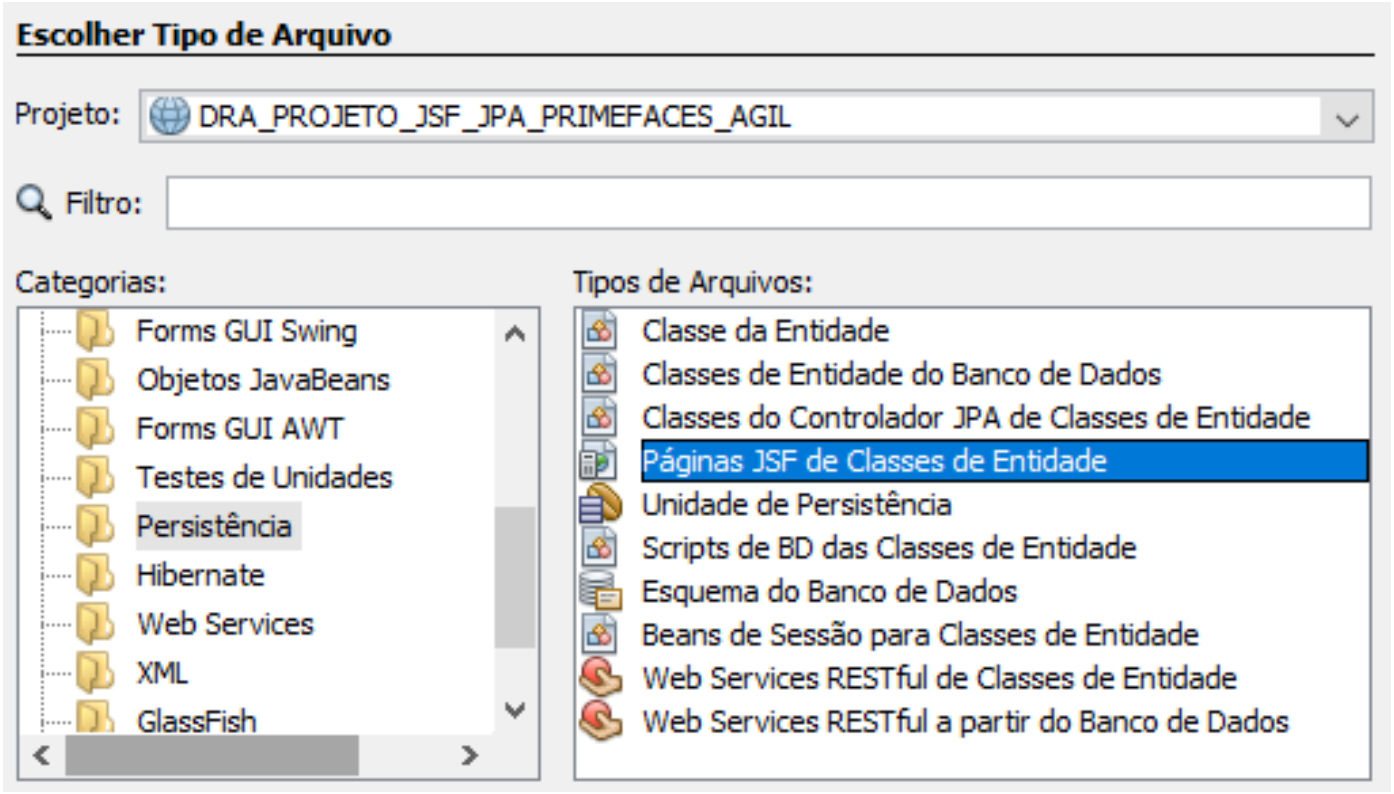
Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Criando Classe de Entidade do Banco de Dados



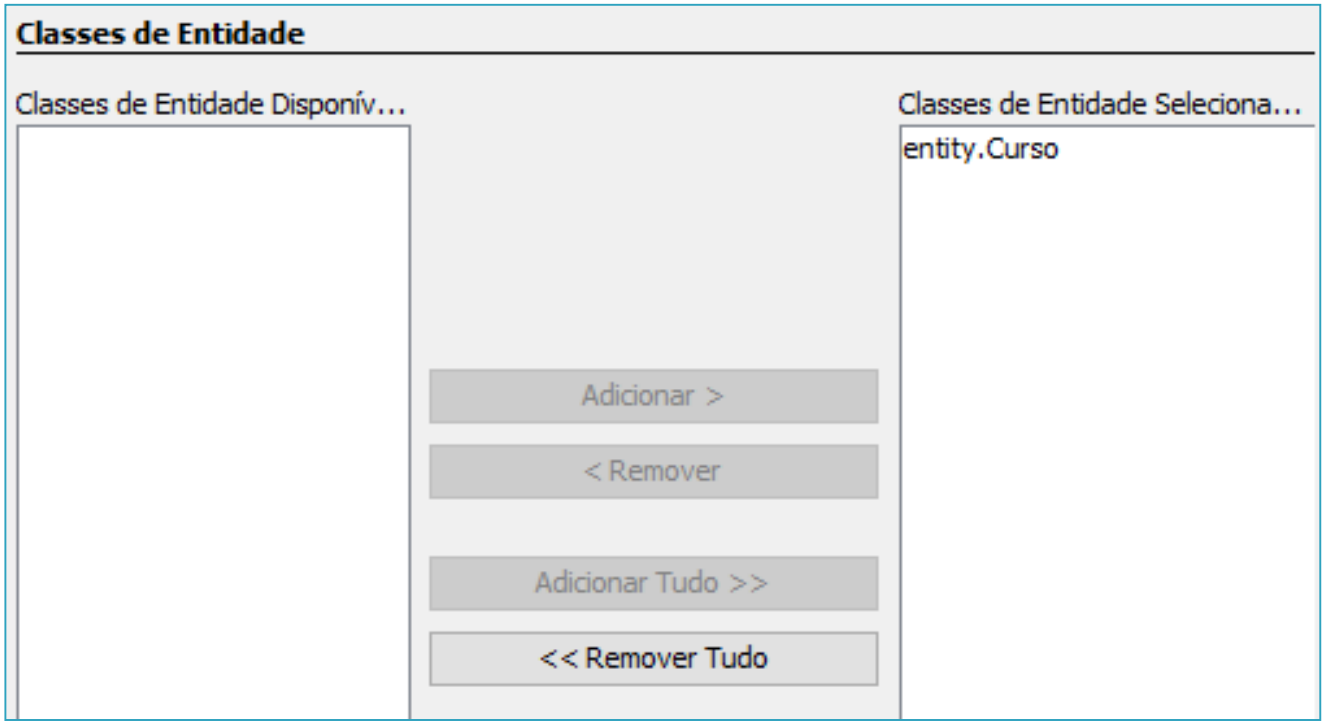
Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Criando Páginas JSF de Classes de Entidade



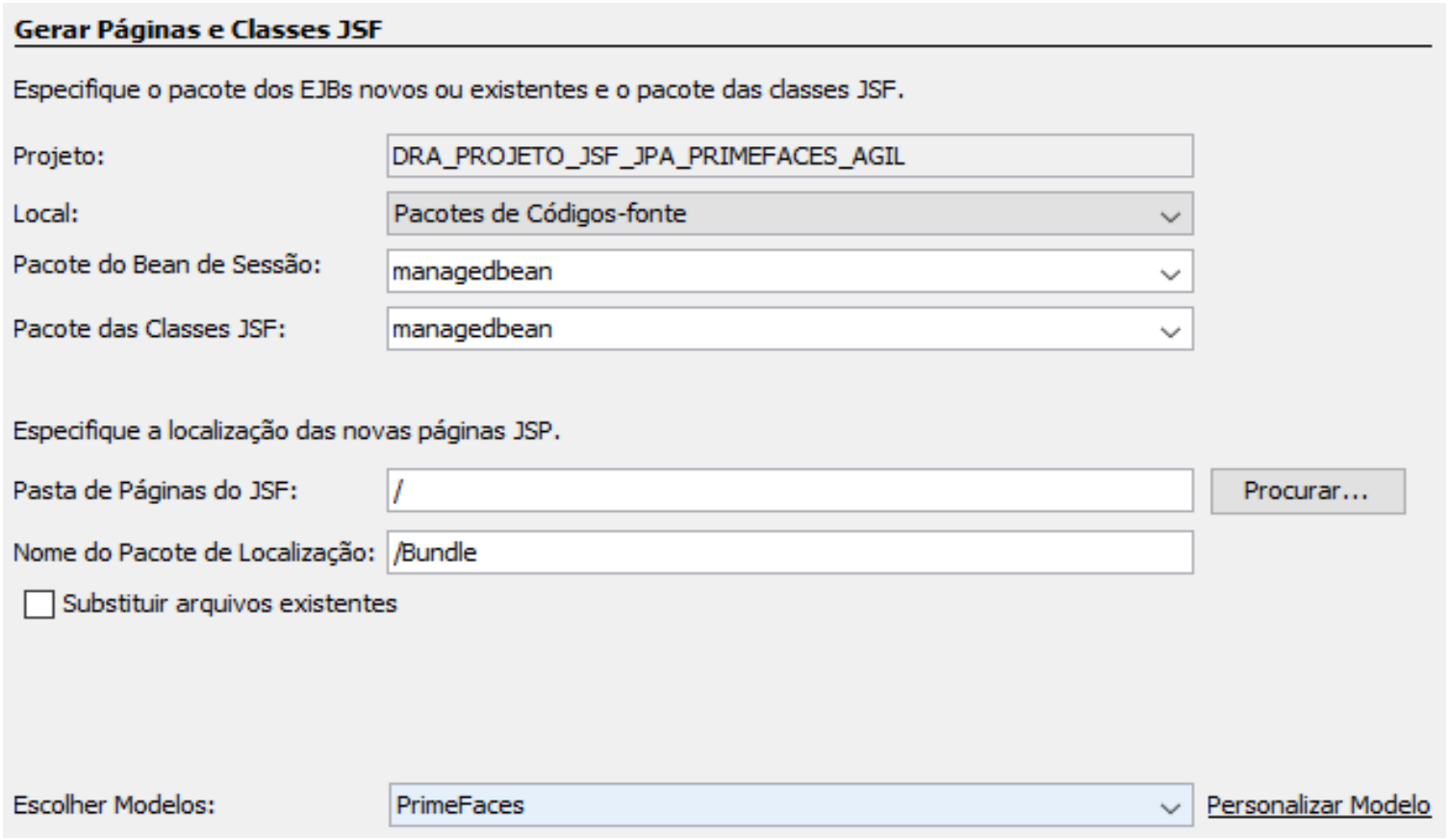
Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Criando Páginas JSF de Classes de Entidade



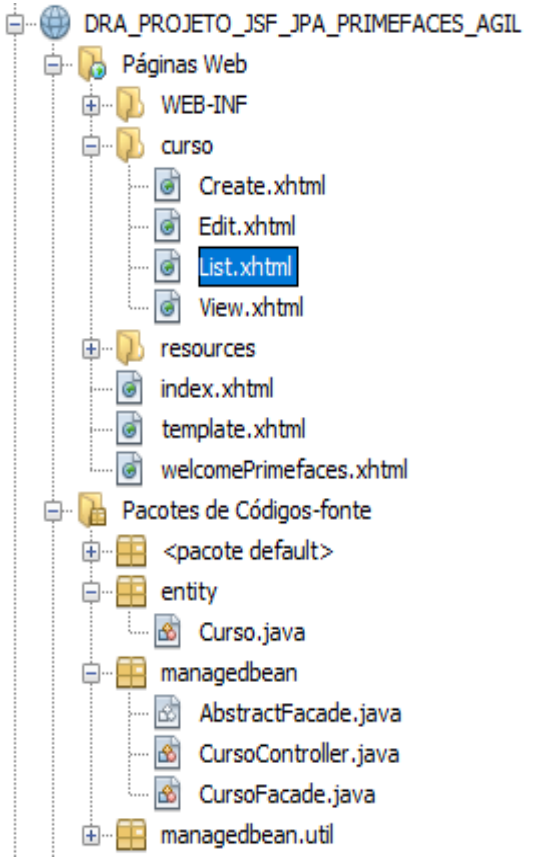
Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Criando Páginas JSF de Classes de Entidade



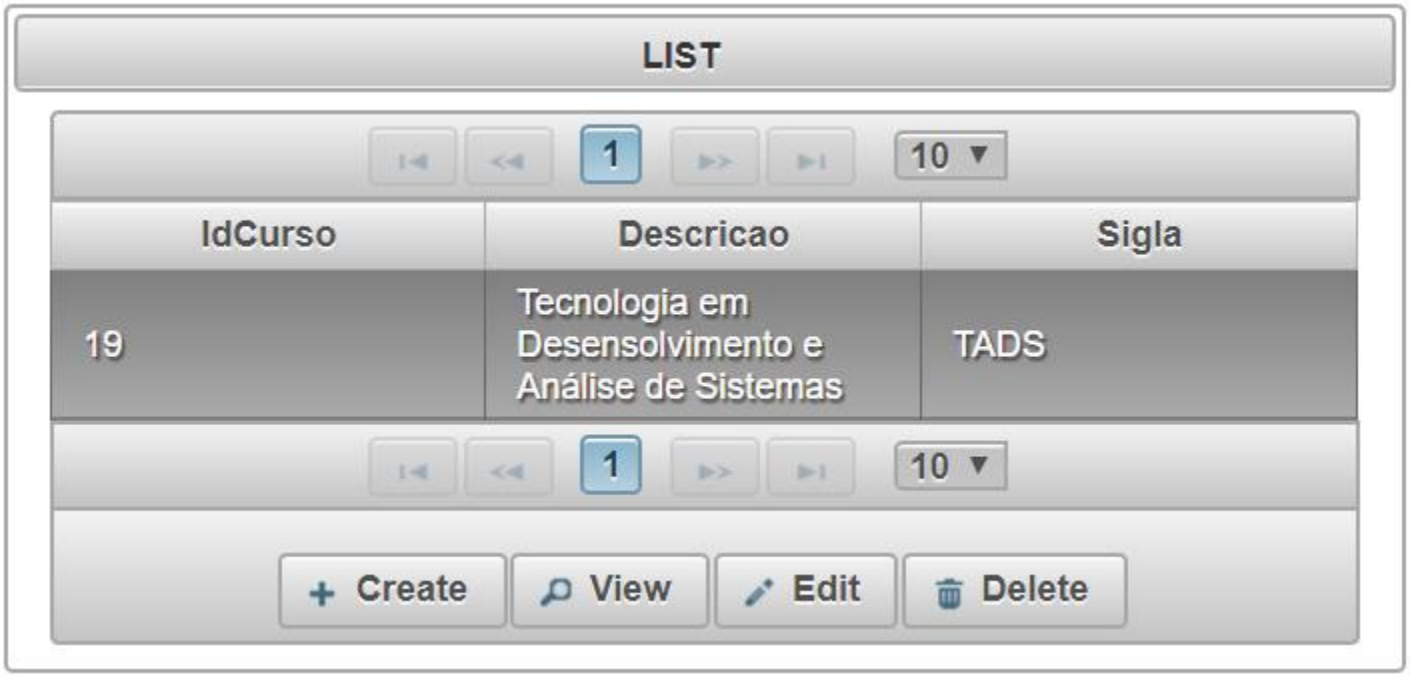
Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Criando Páginas JSF de Classes de Entidade



Java Server Faces - JSF

- Desenvolvimento **Ágil** com JSF + PrimeFaces + JPA
  - Resultado Final



EXTRAS

Java Server Faces - JSF

- Adicionando CSS

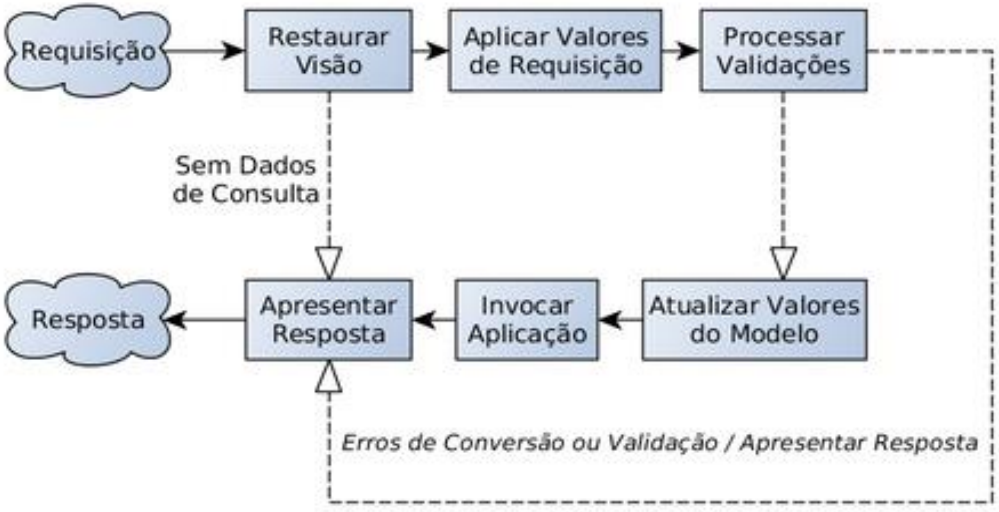

```
<h:outputStylesheet name="css/estilos.css" />
```

- Adicionando JavaScript

```
<h:outputScript name="k19.js" library="javascript" target="head" />
```

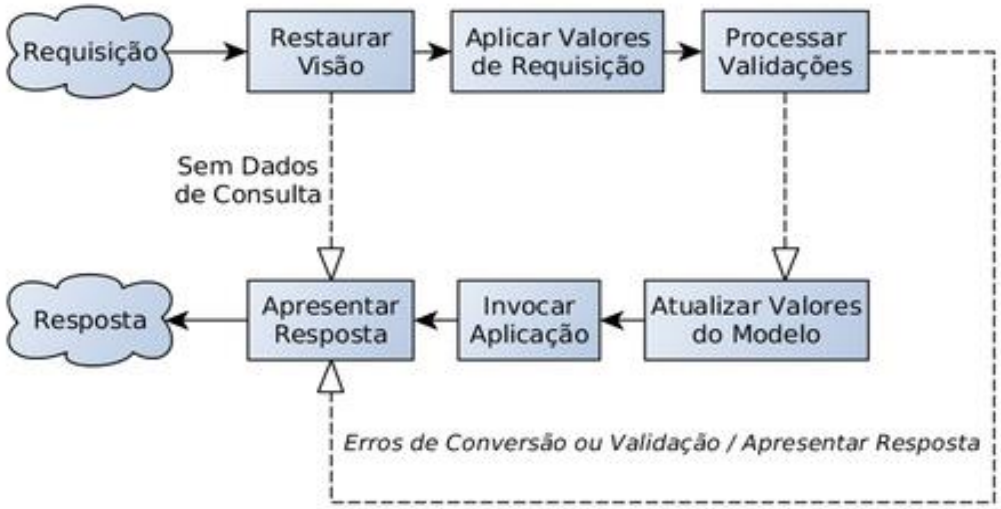
Java Server Faces - JSF

- Faces Servlet - Processamento de Requisição – Ciclo de Vida
  1. **Restaurar Visão**
     - O objetivo principal desta fase é construir/reconstruir a árvore de componentes.
     - Salva o estado da árvore no objeto **FacesContext**.



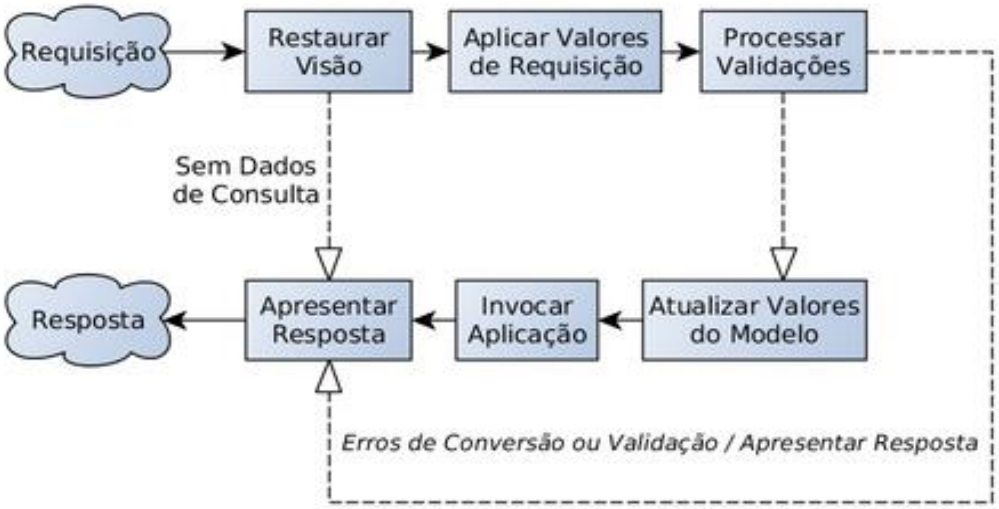
Java Server Faces - JSF

- Faces Servlet - Processamento de Requisição – Ciclo de Vida
  2. **Aplicar Valores de Requisição**
     - A árvore construída na etapa anterior é percorrida e cada um dos seus componentes é “decodificado”.
       - No processo de decodificação, cada componente extrai da requisição os dados associados a esse componente e se atualiza com essas informações.
       - Os eventos de ação são identificados e registrados para serem tratados posteriormente.



Java Server Faces - JSF

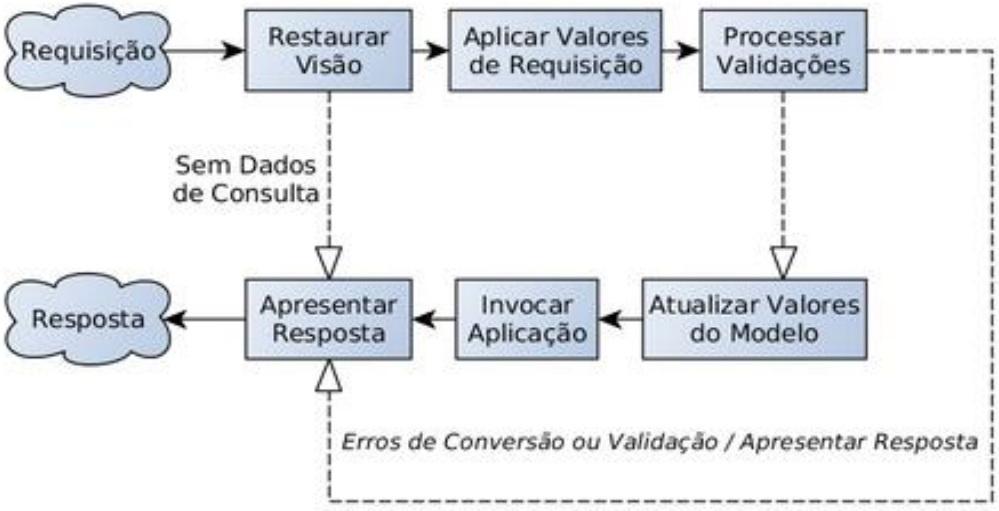
- Faces Servlet - Processamento de Requisição – Ciclo de Vida
  3. **Processar Validações**
     - Os componentes com valores submetidos pelo usuário são convertidos e validados, caso haja algum validador registrado para esse componente.
       - Se ocorrer algum problema, mensagens de erro são adicionadas ao contexto da requisição e o fluxo é redirecionado para a fase “Apresentar Resposta”.
       - Caso contrário, processo continua na fase “Atualizar Valores do Modelo”.



Java Server Faces - JSF

- Faces Servlet - Processamento de Requisição – Ciclo de Vida

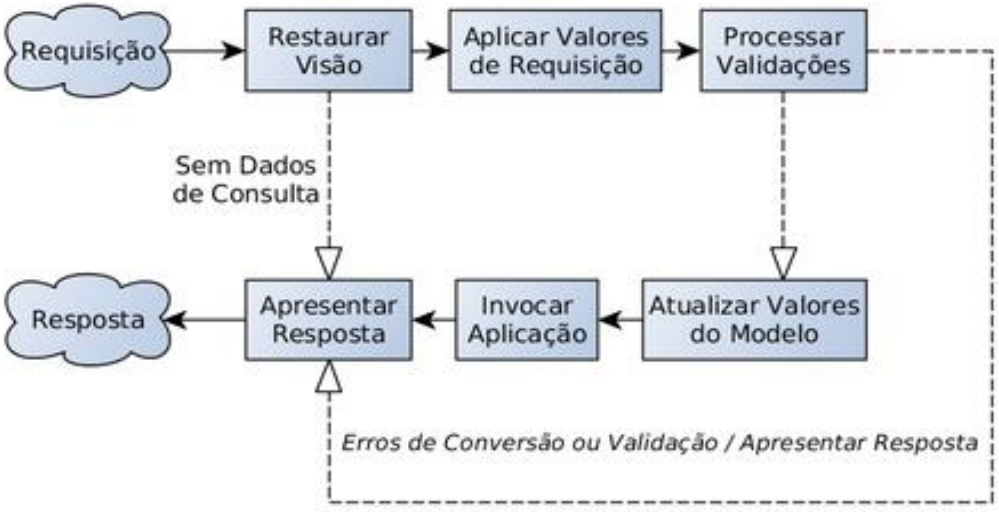
4. **Atualizar Valores de Modelo**
   - Os valores contidos em cada componente da árvore, já convertidos e validados na fase anterior, são armazenados em propriedades de objetos definidos pela aplicação (Managed Beans)



Java Server Faces - JSF

- Faces Servlet - Processamento de Requisição – Ciclo de Vida

5. **Invocar aplicação**
   - As tarefas correspondentes ao evento que disparou a requisição (normalmente um clique em um botão ou link) serão executadas.
   - A próxima tela a ser apresentada ao usuário é determinada pelo resultado do método que implementa a lógica de negócio executado nesta fase.



Java Server Faces - JSF

- Faces Servlet - Processamento de Requisição – Ciclo de Vida

6. **Apresentar resposta**
   - A próxima tela é gerada e enviada ao navegador do usuário.
   - Uma representação desta tela também é armazenada a fim de ser usada na fase “Restaurar Visão” da próxima requisição.