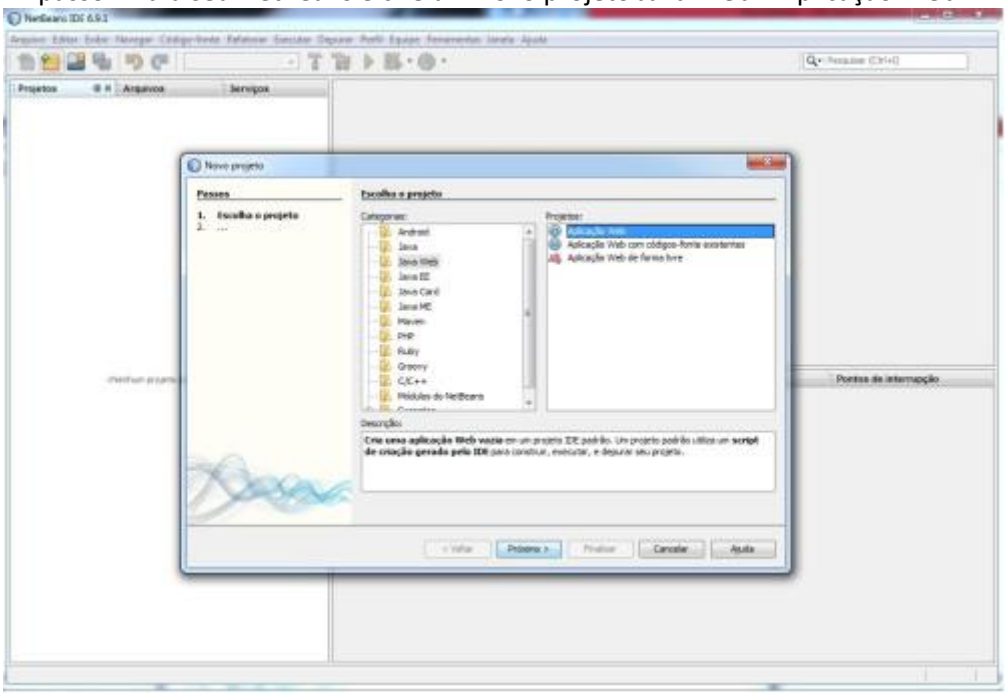
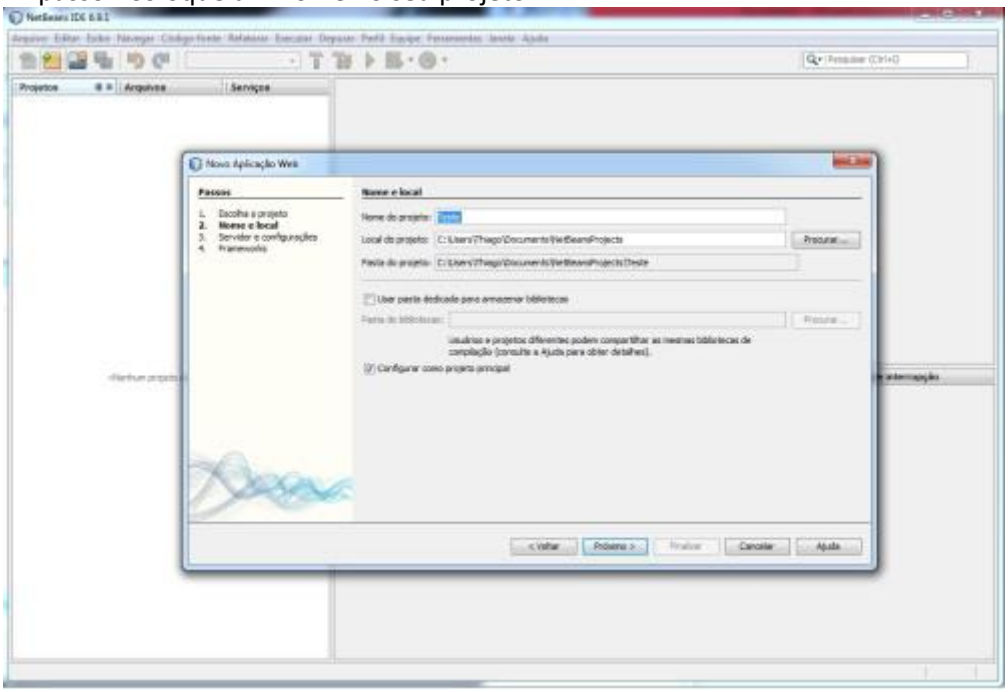
Projeto Exemplo

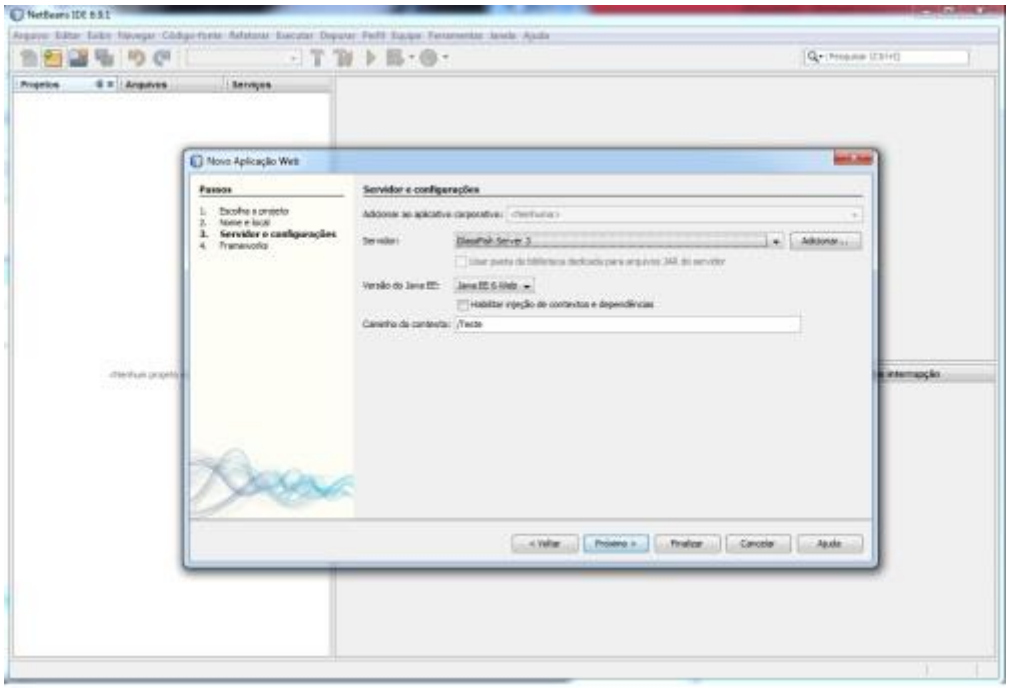
1º passo: Abra seu NetBeans e crie um novo projeto Java Web > Aplicação Web.



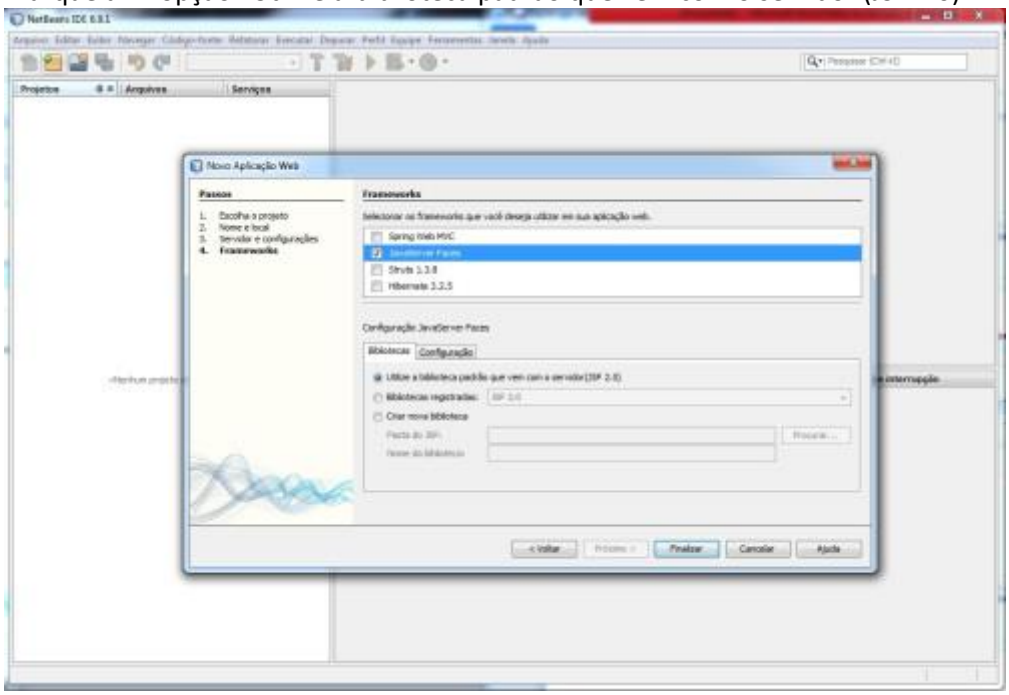
2º passo: Coloque um nome no seu projeto.



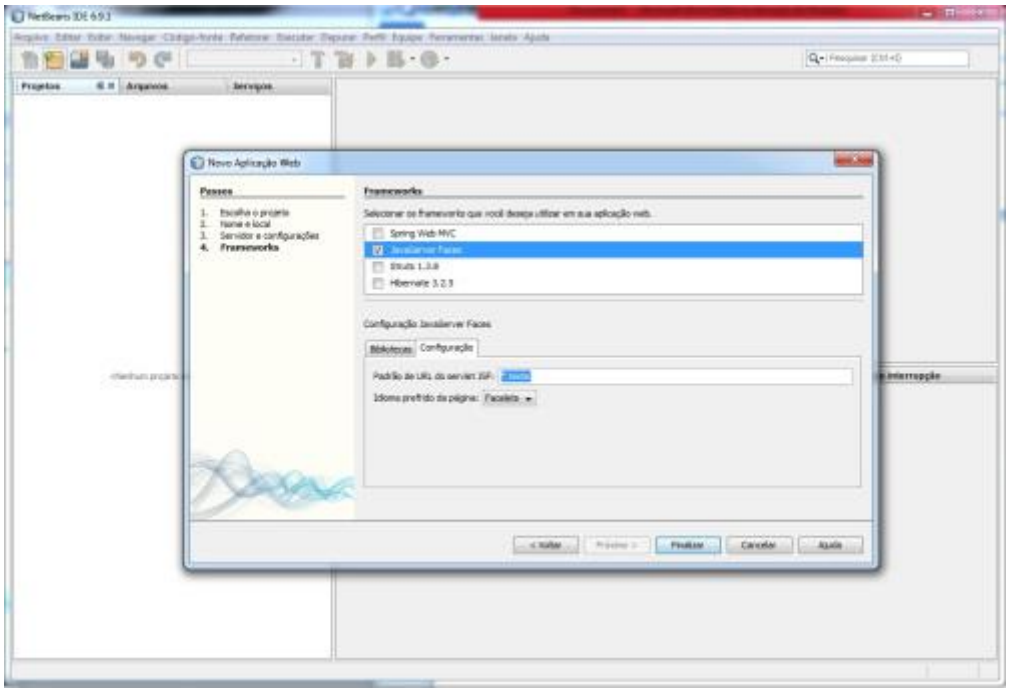
3º passo: escolha o servidor GLASSFISH , e a versão do Java EE .



4º passo: escolha o Framework JavaServer Faces que é o que vamos trabalhar, as bibliotecas marque a 1º opção “Utilize a biblioteca padrão que vem com o servidor (JSF 2.0)”.

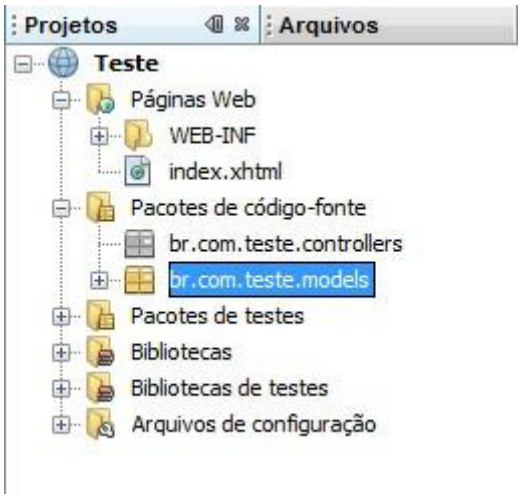


5º passo: na aba Configurações opção Padrão de URL do servlet JSF: mude para qualquer extensão que deseje, (ex: *.teste) assim todas minhas paginas serão da seguinte forma: “minhapagina.teste”. e pronto finalize e temos um projeto JSF 2.0.



Se você mandar executar o projeto já temos um “Hello Facelets” pronto, suponho que seja básico de mais fazer um “Hello World”, vamos progredir e criar um formulário simples, mas antes tenho que explicar um pouco mais sobre o JSF, ele é um framework possui uma arquitetura MVC(Model-View-Controller), Model será nossas classe, View a telas de entrada e saída de dados, e Controller será o Managed Beans, ou seja Bens Gerenciados, onde vai conter nossa lógica de negócio ou programação, com isso seu código fica organizado e de fácil manutenção e outros fatores, exemplo na sua View(tela) não tem código java, somente seus componentes de tela e assim fica fácil para um design trabalhar com o visual, um bom artigo sobre isso é do Rafael Pontes referência em JSF, então vamos lá.

– Crie um pacote models e um controllers, para suas classes e os Managed Beans assim fica melhor visualizar seus arquivos.



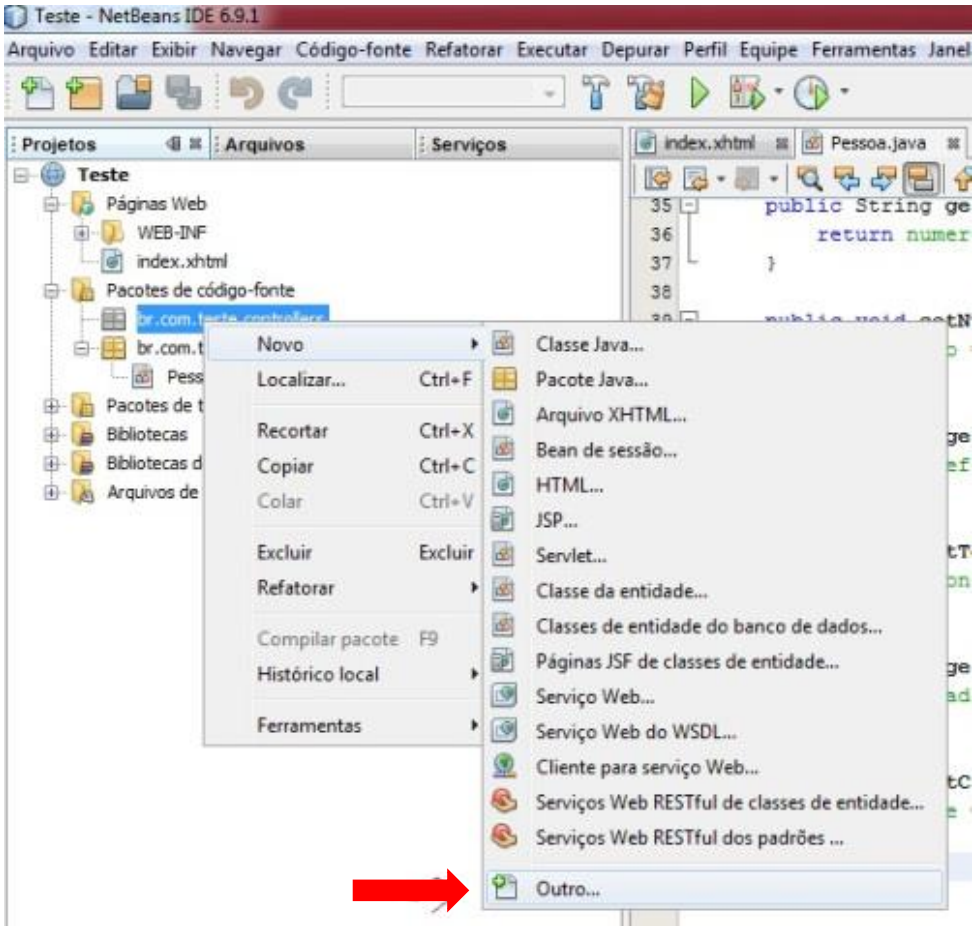
– Crie uma classe java chamada Pessoa dentro do pacote models, com os atributos nome, endereço, numero, telefone e cidade.

Código Pessoa.java:

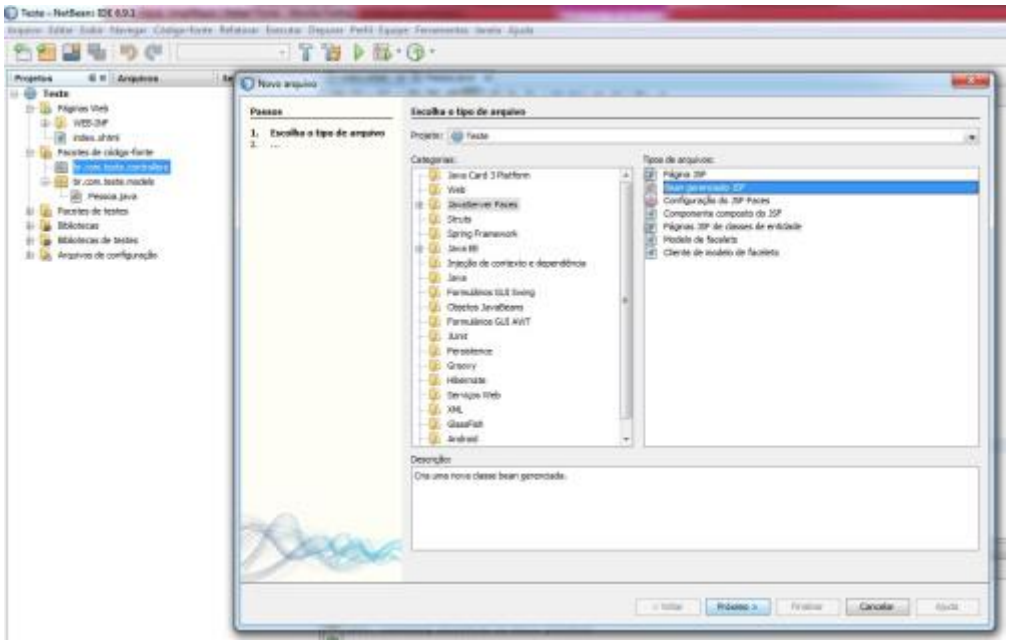
```
1   /*
2    * To change this template, choose Tools | Templates
3    * and open the template in the editor.
4    */
5
6   package br.com.teste.models;
7
8   /**
9    *
10   * @author AulaQi
11   */
12  public class Pessoa {
13   private String nome;
14   private String endereco;
15   private String numero;
16   private String telefone;
17   private String cidade;
18
19   public String getCidade() {
20   return cidade;
21   }
22
23   public void setCidade(String cidade) {
24   this.cidade = cidade;
25   }
26
27   public String getEndereco() {
28   return endereco;
29   }
30
31   public void setEndereco(String endereco) {
32   this.endereco = endereco;
33   }
```

```
34
35      public String getNome() {
36      return nome;
37      }
38
39      public void setNome(String nome) {
40      this.nome = nome;
41      }
42
43      public String getNumero() {
44      return numero;
45      }
46
47      public void setNumero(String numero) {
48      this.numero = numero;
49      }
50
51      public String getTelefone() {
52      return telefone;
53      }
54
55      public void setTelefone(String telefone) {
56      this.telefone = telefone;
57      }
58
59     }
```

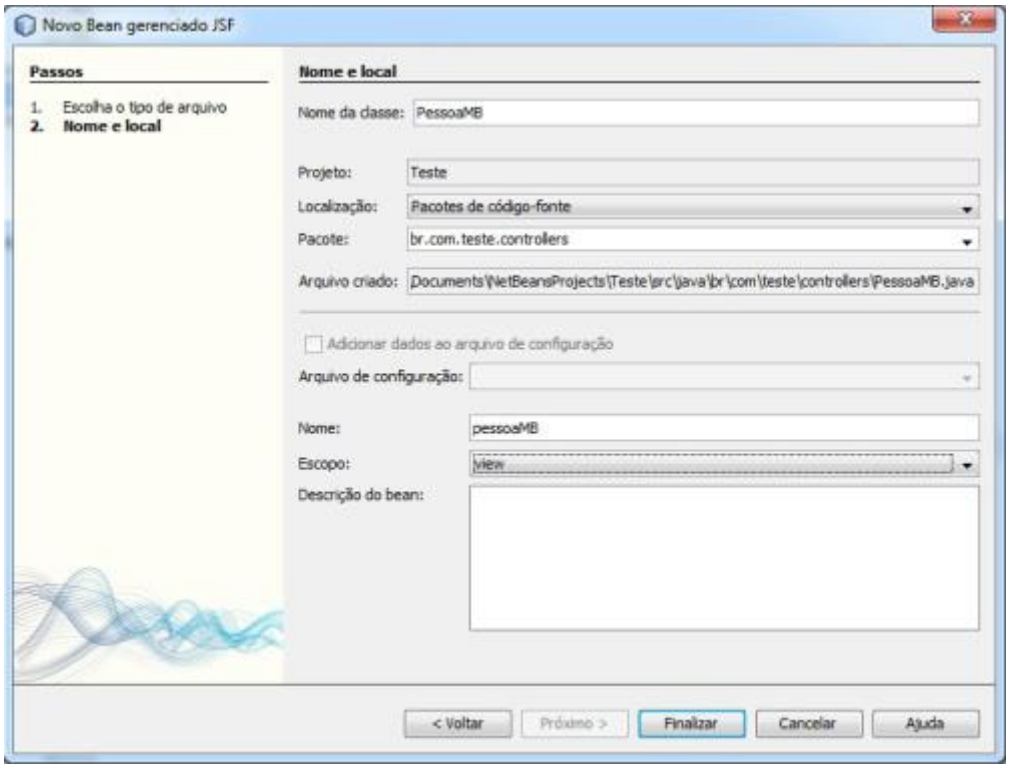
– Crie um Bean Gerenciado no pacote controllers, veja como encontrar.



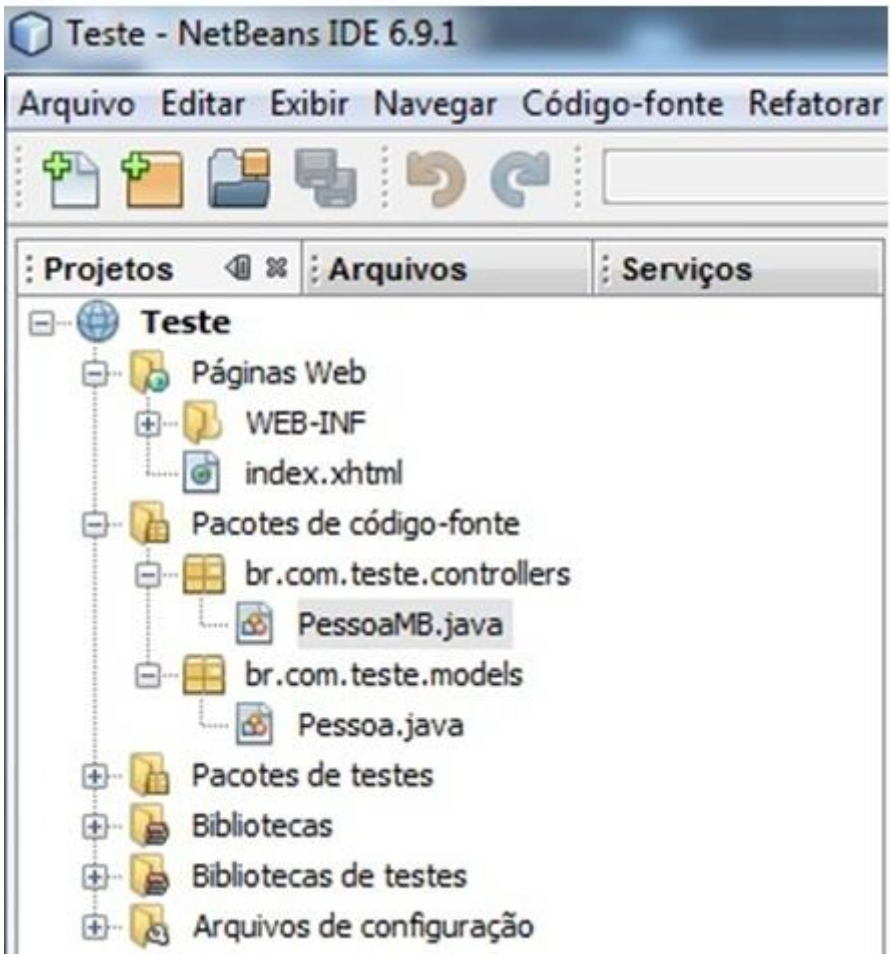
– Escolha a Categoria JavaServer Faces e selecione Bean Gerenciado JSF.



– Vamos dar o nome de PessoaMB, escolha o escopo “View”, em outro momento explico o que seria isso.



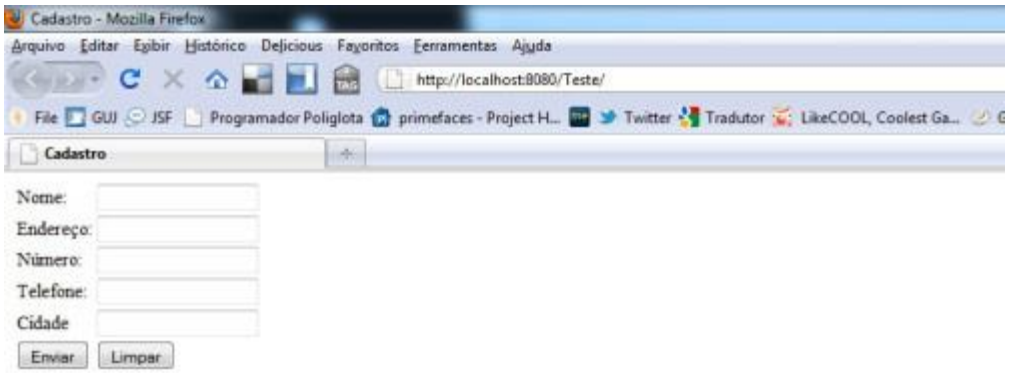
– Seu projeto desse estar com essa cara agora, temos o index.xhtml que foi gerado ao criar o projeto, temos nossa classe modelo Pessoa e o controller PessoaMB.



– Abra o index.xhtml e vamos criar nosso formulário:
Código Index.xhtml:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!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:f="http://java.sun.com/jsf/core">
 <h:head>
 <title>Cadastro</title>
 </h:head>
 <h:body>
 <h:form>
 <h:panelGrid columns="3">
 <h:outputLabel value="Nome:"/>
 <h:inputText id="nome"  size="20" />
 <h:message for="nome" />

 <h:outputLabel value="Endereçoo:"/>
 <h:inputText id="end"  size="20"  >
 <h:message for="end" />

 <h:outputLabel value="Número:"/>
 <h:inputText id="nro"  size="20" />
 <h:message for="nro" />

 <h:outputLabel value="Telefone:"/>
 <h:inputText id="tel"  size="20" />
 <h:message for="tel" />

 <h:outputLabel value="Cidade:"/>
 <h:inputText id="mun"  size="20" />
 <h:message for="mun" />

 <h:commandButton id="btnE" value="Enviar" />
 <h:commandButton id="btnL" value="Limpar"/>

 </h:panelGrid>
 </h:form>
 </h:body>
</html>
```

Resultado não é com visual tão atrativo, em outro momento vamos trabalhar com o visual.



– Abra seu controller PessoaMB, vamos implementar a interface Serializable explico mais para frente o que seria isso, e instancie um objeto pessoa, e uma lista do tipo pessoa para armazenar as pessoas.
Código PessoaMB.java:

```
1   /*
2    * To change this template, choose Tools | Templates
3    * and open the template in the editor.
4    */
5   package br.com.teste.controllers;
6
7   import br.com.teste.models.Pessoa;
8   import java.io.Serializable;
9   import java.util.ArrayList;
10  import java.util.List;
11  import javax.faces.bean.ManagedBean;
12  import javax.faces.bean.ViewScoped;
13
14  /**
15   *
16   * @author AulaQi
17   */
18  @ManagedBean
19  @ViewScoped
20  public class PessoaMB implements Serializable {
21
22      private Pessoa pessoa = new Pessoa();
23      private List<Pessoa> pessoaLista = new ArrayList<Pessoa>();
24
25      /** Creates a new instance of PessoaMB */
26      public PessoaMB() {
27      }
28
29      /**Getters e Setters */
30      public Pessoa getPessoa() {
31          return pessoa;
32      }
33
34      public void setPessoa(Pessoa pessoa) {
35          this.pessoa = pessoa;
```

```
36      }
37
38      public List<Pessoa> getPessoaLista() {
39         return pessoaLista;
40      }
41
42      public void setPessoaLista(List<Pessoa> pessoaLista) {
43         this.pessoaLista = pessoaLista;
44      }
```

– Na Index.xhtml e vamos adicionar os values nos inputs.
Agora que instanciamos a classe no Bean podemos acessar os atributos através dele.
Ex: inputText id=”teste” value=”#{seuBean.classe.atributo}” />

Código parte Index.xhtml:

```
1       <h:panelGrid columns="3">
2
3       <h:outputLabel value="Nome:"/>
4
5       <h:inputText id="nome" value="#{pessoaMB.pessoa.nome}" size="20" />
6
7       <h:message for="nome" />
8
9       <h:outputLabel value="Endereço:"/>
10
11      <h:inputText id="end" value="#{pessoaMB.pessoa.endereco}" size="20"  >
12
13      <h:message for="end" />
14
15      <h:outputLabel value="Número:"/>
16
17      <h:inputText id="nro" value="#{pessoaMB.pessoa.numero}" size="20" />
18
19      <h:message for="nro" />
20
21      <h:outputLabel value="Telefone:"/>
22
23      <h:inputText id="tel" value="#{pessoaMB.pessoa.telefone}" size="20" />
24
25      <h:message for="tel" />
26
27      <h:outputLabel value="Cidade:"/>
28
29      <h:inputText id="mun" value="#{pessoaMB.pessoa.cidade}" size="20" />
30
31      <h:message for="mun" />
32
33      <h:commandButton id="btnE" value="Enviar" type="submit"
34      action="#{pessoaMB.salvarPessoa}"/>
35
```

```
36      <h:commandButton id="btnL" value="Limpar" type="reset"/>
37       </h:panelGrid>
```

– No PessoaMB e vamos criar nosso método para salvar essas pessoas.

Logo após os Getters e Setters adicone o método salvar.

```
1       /**Métodos */
2
3       public void salvarPessoa() {
4
5       //adicionando pessoas a lista
6       pessoaLista.add(pessoa);
7       //instanciado uma nova para ser cadastrada
8       pessoa = new Pessoa();
9
10      }
```

– Na Index.xhtml e vamos adicionar ação para o botão.
É o mesmo sentido do value, mas nesse caso é uma action="#{meuBean.meuMetodo}".
Código parte Index.xhtml:

```
1       <h:commandButton id="btnE" value="Enviar" type="submit" action="#{pessoaMB.salvarPessoa}"/>
```

Não comentei o atributo do commandButton type="" />
Para envio de formulário utiliza o "submit" que é defaut, e no caso do segundo que é para limpeza usamos o "reset" que tem a função de limpar o formulário.
– Nosso cadastro está pronto, agora vamos apresentar na tela esses dados que inserimos na lista de pessoas, para isso utilizaremos um dataTable, veja como:
Código parte Index.xhtml:

```
1       <h:panelGrid>
2        <h:dataTable id="tablePessoa" value="#{pessoaMB.pessoaLista}" var="p" title="Pessoas
3       Cadastradas" border="1" rows="10" >
4        <h:column>
5        <f:facet name="header">
6        <h:outputText value="NOME" />
7        </f:facet>
8        <h:outputText value="#{p.nome}"/>
9        </h:column>
10       <h:column>
11       <f:facet name="header">
12       <h:outputText value="ENDEREÃ‡O" />
13       </f:facet>
14       <h:outputText value="#{p.endereco}"/>
15       </h:column>
16       <h:column>
17       <f:facet name="header">
18       <h:outputText value="NÃšMERO" />
19       </f:facet>
20       <h:outputText value="#{p.numero}"/>
21       </h:column>
```

```
22      <h:column>
23      <f:facet name="header">
24      <h:outputText value="TELEFONE" />
25      </f:facet>
26      <h:outputText value="#{p.telefone}"/>
27      </h:column>
28      <h:column>
29      <f:facet name="header">
30      <h:outputText value="CIDADE" />
31      </f:facet>
32      <h:outputText value="#{p.cidade}"/>
33      </h:column>
34      </h:dataTable>
        </h:panelGrid>
```

Veja que no value do dataTable utilizei a lista que criamos no bean, value=”#{pessoaMB.pessoaLista}” e guardei numa variável var=”p” e através dela acesso os atributos específicos que estão armazenados na lista de pessoas.

Está pronto agora podemos visualizar o conteúdo que foi inserido na lista, podemos adicionar uma validação simples só para dar um “UP” no nosso formulário e mostrar a finalidade daquele , e que não é tão complicado em alguns casos validar no JSF.

No inputText /> que quiser que seja um campo obrigatório coloque o atributo required=”true” que torna o campo obrigatório e requiredeMessage=”Campo obrigatório” que é a mensagem que será exibida quando não satisfazer a condição, caso esqueça verá uma mensagem default do JSF. Ex:

Código validando com required:

```
1       <h:outputLabel value="Nome:"/>
2       <h:inputText id="nome" value="#{pessoaMB.pessoa.nome}" size="20"  required="true"
3       requiredMessage="Campo Obrigatório"/>
        <h:message for="nome" />
```

- Código final do index.xhtml:

```
1       <?xml version='1.0' encoding='UTF-8' ?>
2
3       <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
4       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
5
6       <html xmlns="http://www.w3.org/1999/xhtml"
7
8       xmlns:h="http://java.sun.com/jsf/html"
9
10      xmlns:f="http://java.sun.com/jsf/core">
11
12      <h:head>
13
14      <title>Cadastro</title>
15
16      </h:head>
17
18      <h:body>
19
20      <h:form>
```

```
21
22      <h:panelGrid columns="3">
23
24      <h:outputLabel value="Nome:"/>
25
26      <h:inputText id="nome" value="#{pessoaMB.pessoa.nome}" size="20"  required="true"
27      requiredMessage="Campo Obrigatório"/>
28
29      <h:message for="nome" />
30
31      <h:outputLabel value="Endereço:"/>
32
33      <h:inputText id="end" value="#{pessoaMB.pessoa.endereco}" size="20" required="true"
34      requiredMessage="Campo Obrigatório"/>
35
36      <h:message for="end" />
37
38      <h:outputLabel value="Número:"/>
39
40      <h:inputText id="nro" value="#{pessoaMB.pessoa.numero}" size="20" />
41
42      <h:message for="nro" />
43
44      <h:outputLabel value="Telefone:"/>
45
46      <h:inputText id="tel" value="#{pessoaMB.pessoa.telefone}" size="20" />
47
48      <h:message for="tel" />
49
50      <h:outputLabel value="Cidade:"/>
51
52      <h:inputText id="mun" value="#{pessoaMB.pessoa.cidade}" size="20" required="true"
53      requiredMessage="Campo Obrigatório"/>
54
55      <h:message for="mun" />
56
57      <h:commandButton id="btnE" value="Enviar" type="submit"
58      action="#{pessoaMB.salvarPessoa}"/>
59
60      <h:commandButton id="btnL" value="Limpar" type="reset"/>
61
62      </h:panelGrid>
63
64      </h:form>
65
66      <h:panelGrid>
67
68      <h:dataTable id="tablePessoa" value="#{pessoaMB.pessoaLista}" var="p" title="Pessoas
69      Cadastradas" border="1" rows="10" >
70
71      <h:column>
72
```

```
73          <f:facet name="header">
74
75          <h:outputText value="NOME" />
76
77          </f:facet>
78
79          <h:outputText value="#{p.nome}"/>
80
81          </h:column>
82
83          <h:column>
84
85          <f:facet name="header">
86
87          <h:outputText value="ENDEREÇO" />
88
89          </f:facet>
90
91          <h:outputText value="#{p.endereco}"/>
92
93          </h:column>
94
95          <h:column>
96
97          <f:facet name="header">
98
99          <h:outputText value="NÚMERO" />
100
101         </f:facet>
102
103         <h:outputText value="#{p.numero}"/>
104
105         </h:column>
106
107         <h:column>
108
109         <f:facet name="header">
110
111         <h:outputText value="TELEFONE" />
112
113         </f:facet>
114
115         <h:outputText value="#{p.telefone}"/>
116
117         </h:column>
118
119         <h:column>
120
121         <f:facet name="header">
122
123         <h:outputText value="CIDADE" />
124
```

```
125         </f:facet>
126
127         <h:outputText value="#{p.cidade}"/>
128
129         </h:column>
130
131         </h:dataTable>

            </h:panelGrid>

            </h:body>

            </html>
```

Resultado Final: