Java Enterprise Edition - Java EE

JPA e JSF

com Exemplos Básicos usando o NetBeans e GlassFish

Sumário

1. Criando uma nova base de dados para o exemplo

Crie um novo usuário com o login: **javaee**, senha: **javaee** e crie uma base de dados com o mesmo nome do usuário (**javaee**), com todas as permissões.

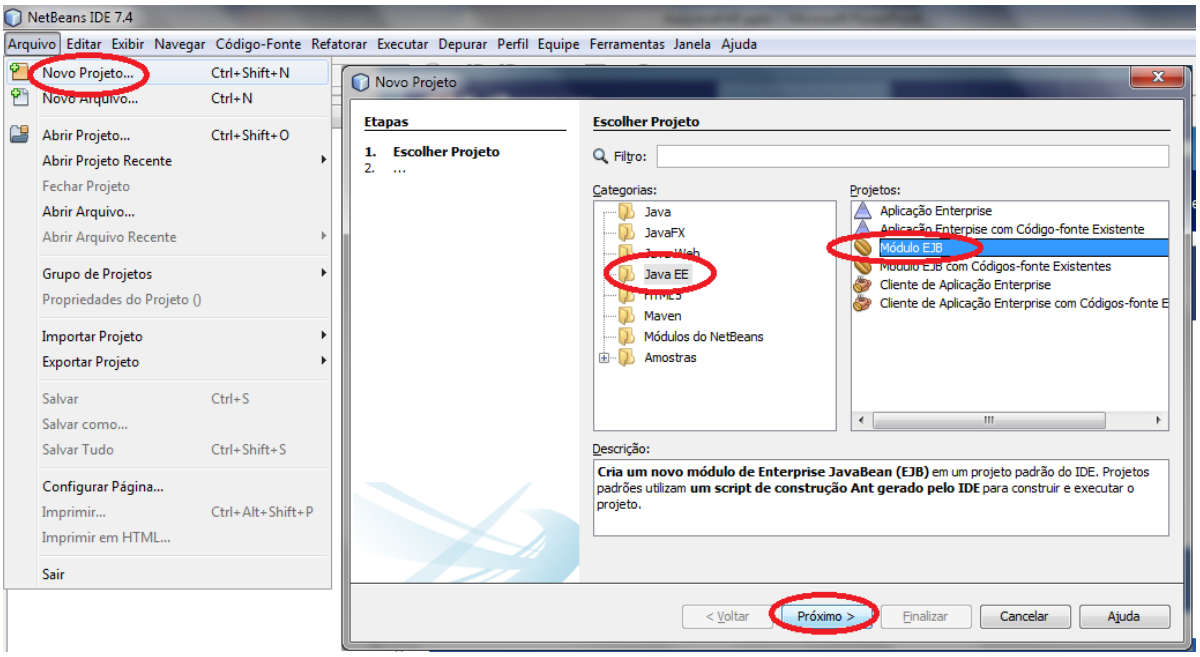
Na base **javaee**, crie uma tabela **pessoa** com os seguintes campos:

- cpf: char(15), chave primária
- nome: varchar(50)
- email: varchar(100)

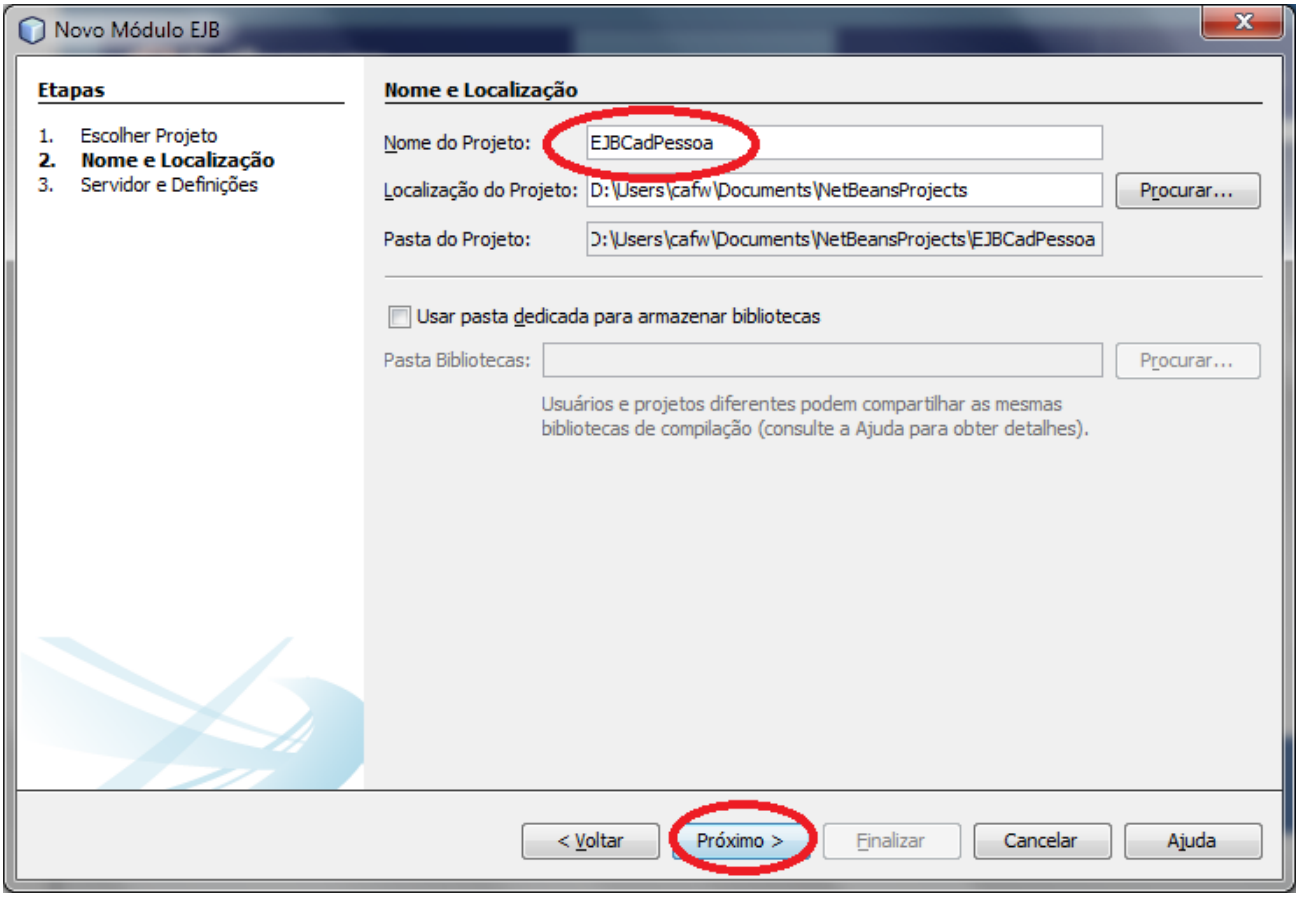
2. Criando um Módulo Enterprise Java Bean (Java EE)

Vamos criar um EJB, com a lógica do negócio que fará o Cadastro de Pessoa (cpf - chave primária, nome, email) numa base MySQL. Este EJB executará num servidor de aplicações GlassFish e será usado por uma aplicação Desktop, uma aplicação Web e por um Web Service acessado por um Android.

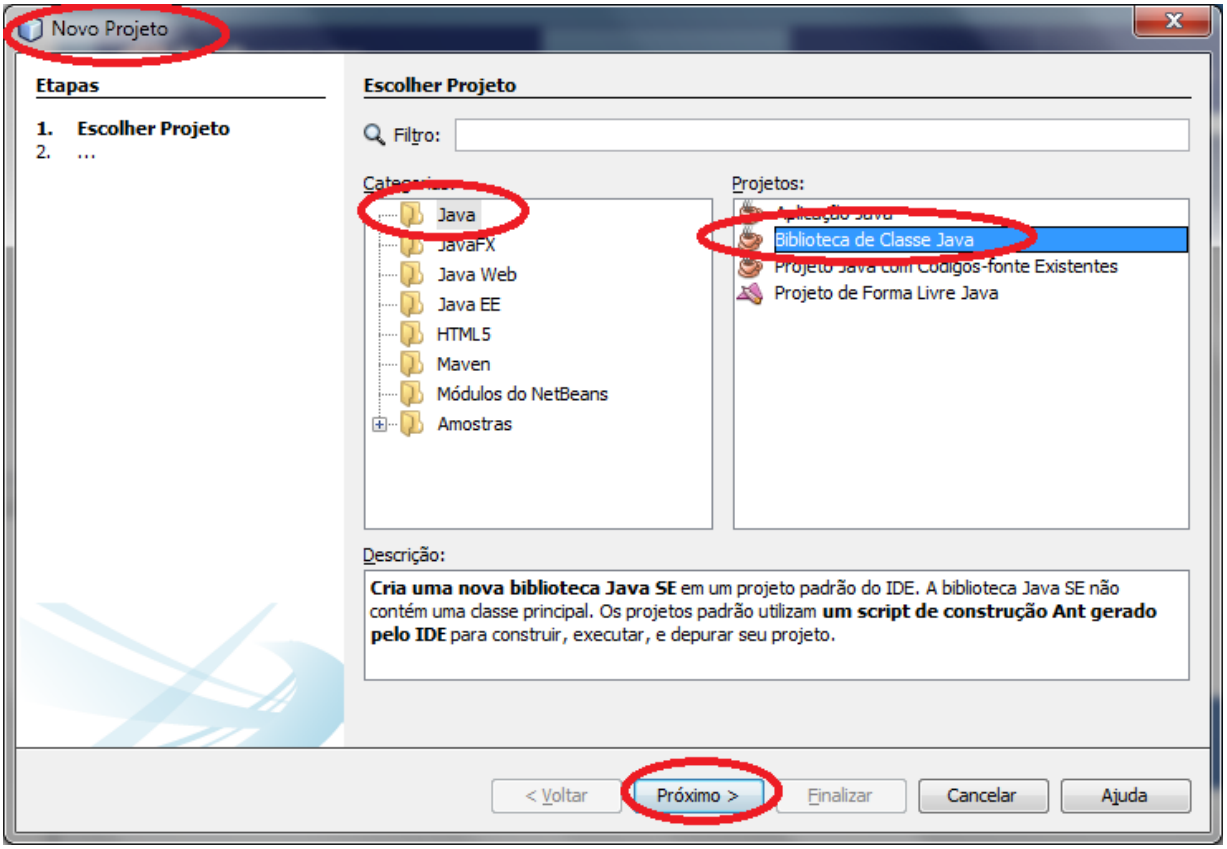
Para isso, crie um Novo Projeto => Java EE => Módulo EJB.



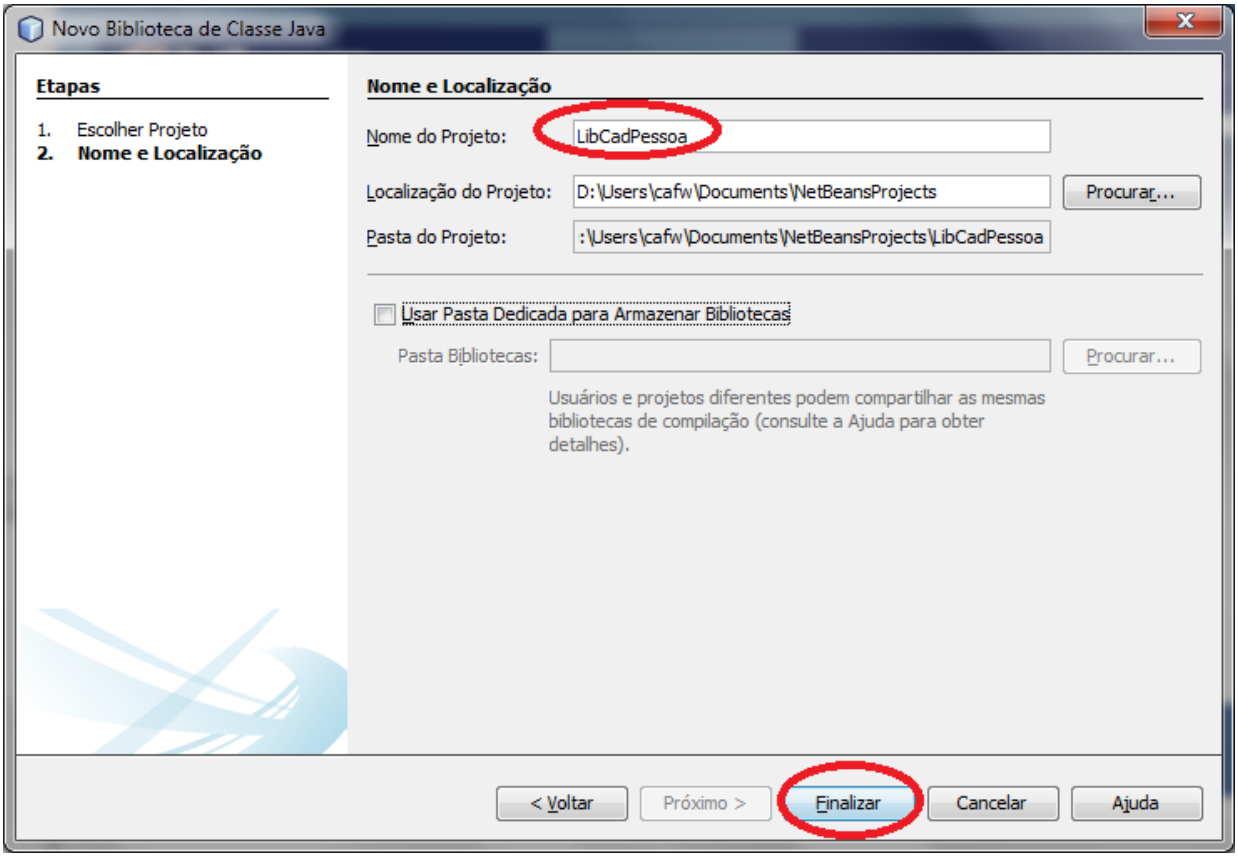
Dê o nome de EJBCadPessoa



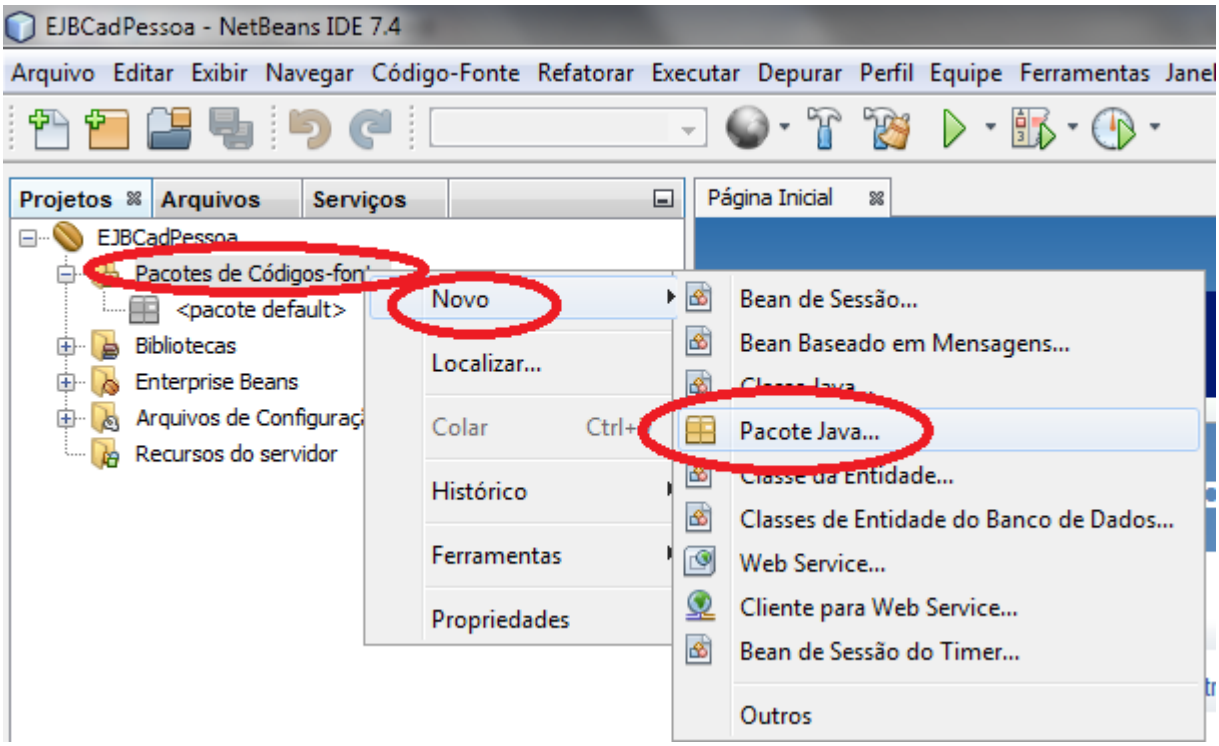
Crie também um Novo Projeto => Java => Biblioteca de Classe Java, para armazenar a interface remota do EJB.

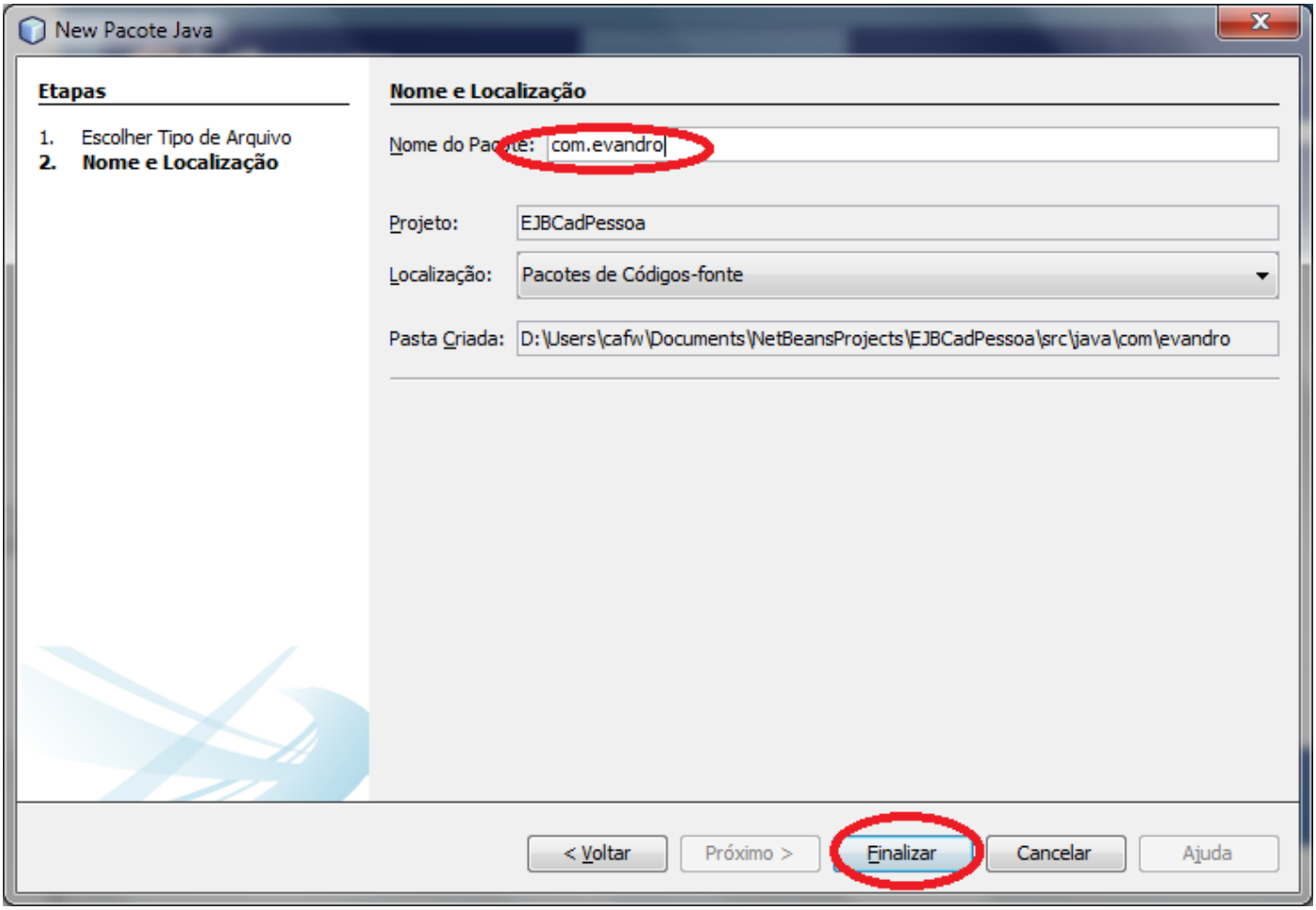


Dê o nome de LibCadPessoa

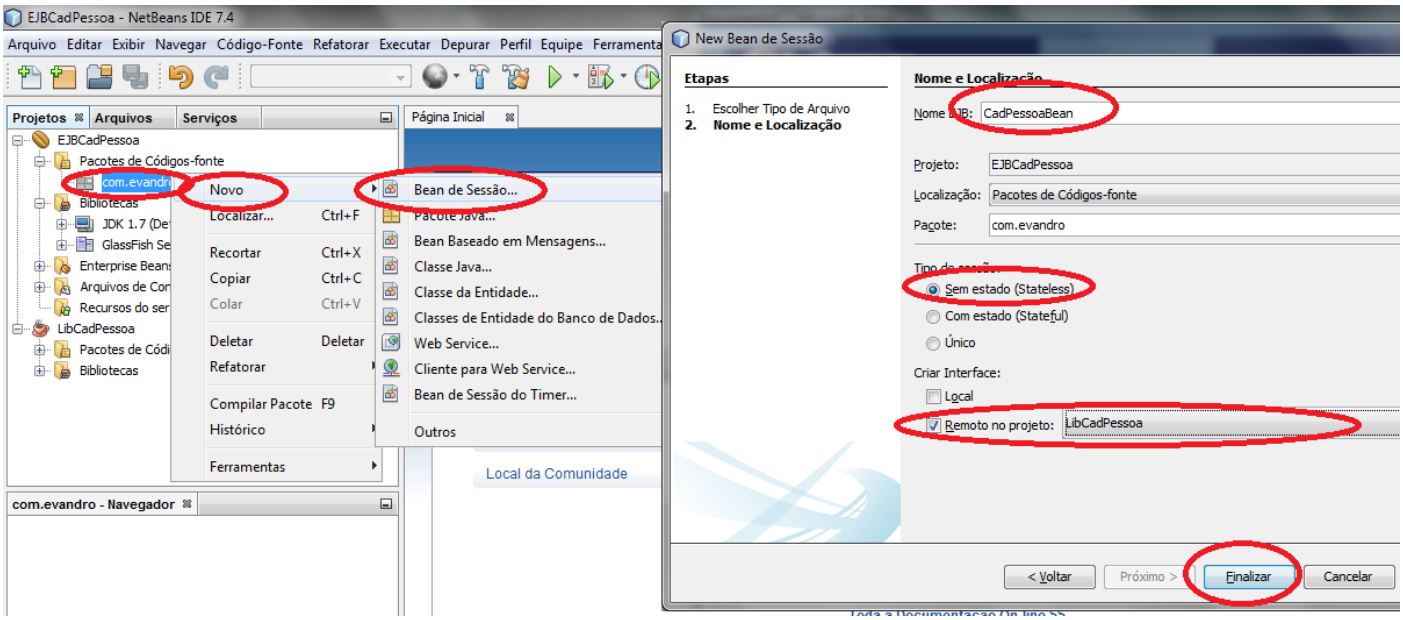


No EJBCadPessoa, clique com o botão direito e crie um novo Pacote Java, com um nome significativo, por exemplo: com.evandro

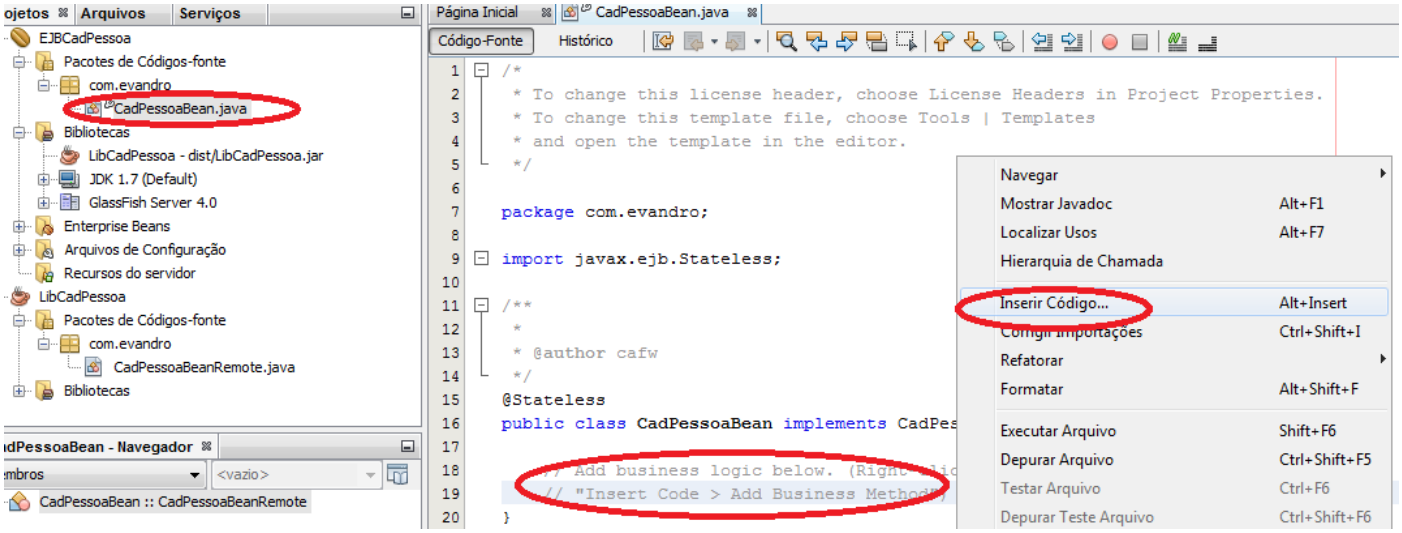




Nesse pacote criado, crie um Novo Bean de Sessão Stateless, com o nome CadPessoaBean e com Interface Remota no projeto LibCadPessoa.

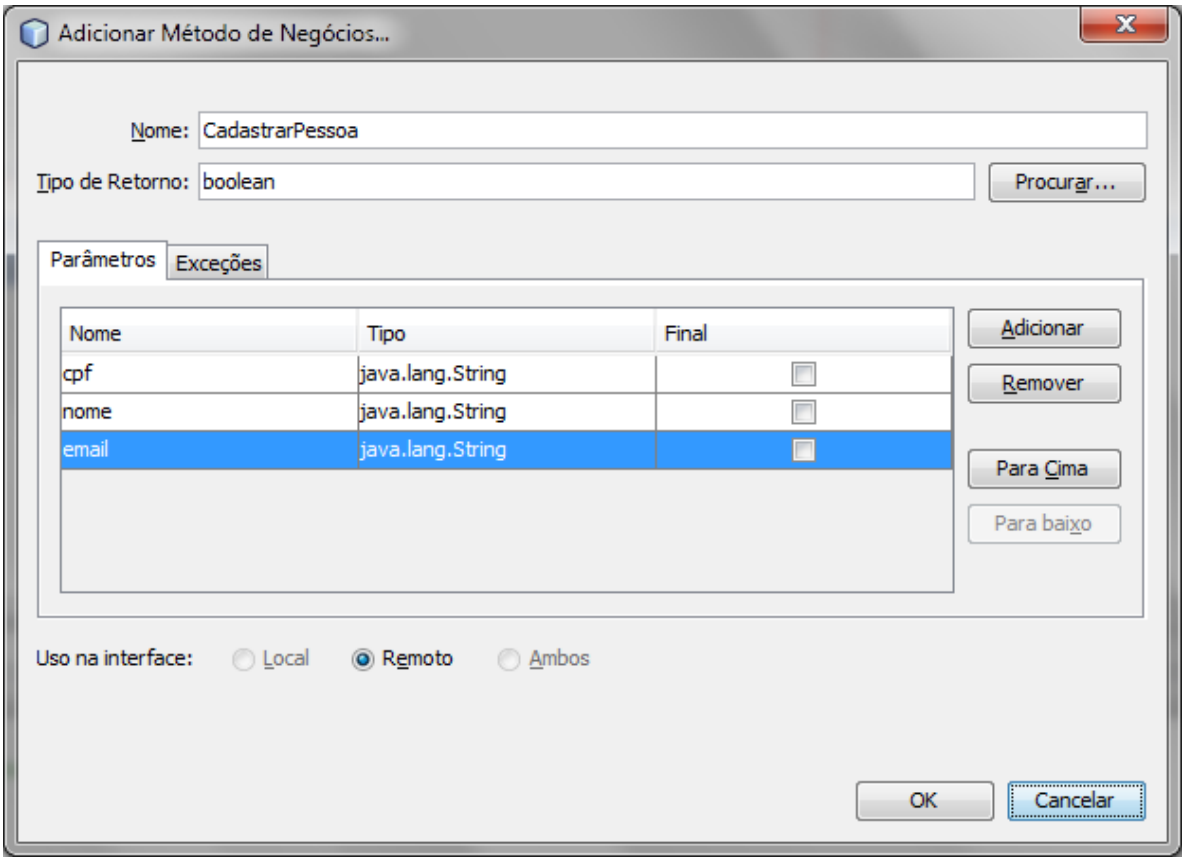


Clique com o botão direito sobre o código do CadPessoaBean.java e clique em Inserir Código...



Clique em Inserir Código => Adicionar Método de Negócios.

Dê o nome do método de CadastrarPessoa, defina o Tipo de Retorno como Boolean e adicione 3 parâmetros do tipo String (cpf, nome, email).

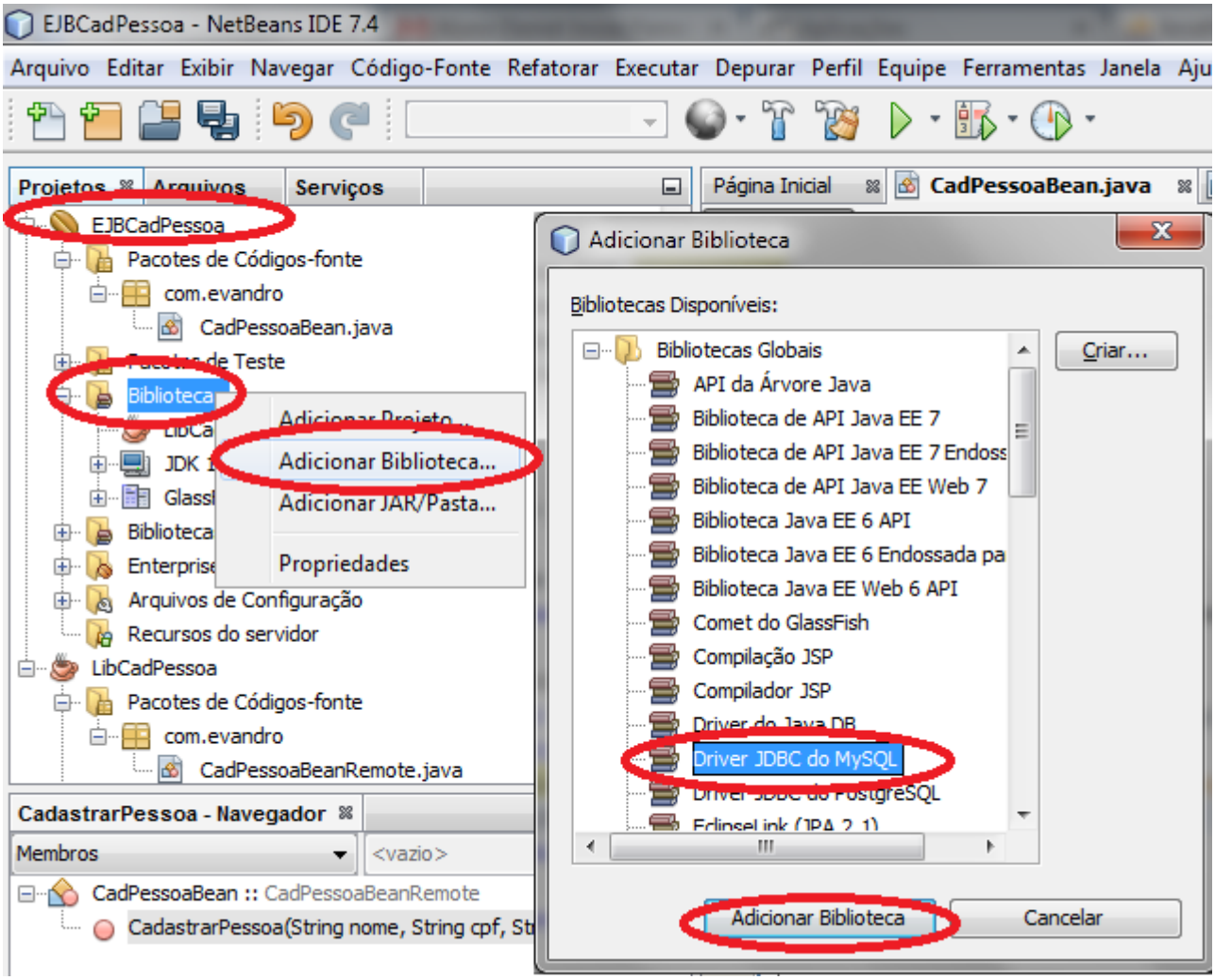


Implemente o código CadPessoaBeanRemote com uma conexão com o MySQL e a respectiva inserção, como o que está a seguir:

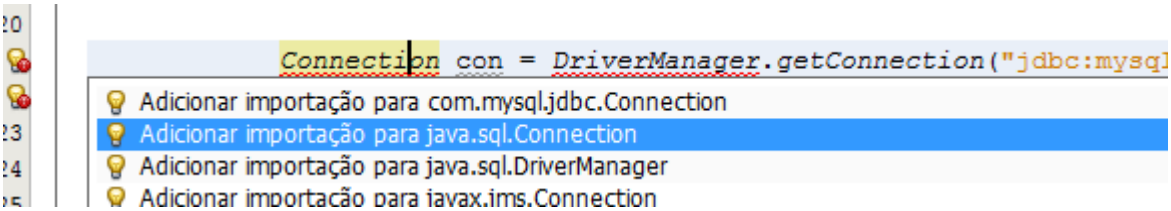
```
public class CadPessoaBean implements CadPessoaBeanRemote {

    @Override
    public boolean CadastrarPessoa(String cpf, String nome, String email) {
        try {
            Connection con =
DriverManager.getConnection("jdbc:mysql://localhost/javaee?user=javaee&password=javaee");
            PreparedStatement pstmt = con.prepareStatement(" INSERT INTO pessoa (cpf, nome,
email) VALUES( ?, ?, ?)");
```

```
            pstmt.setString(1, cpf);
            pstmt.setString(2, nome);
            pstmt.setString(3, email);
            pstmt.executeUpdate();
        } catch (SQLException err) {
            return false;
        }
        return true;
    }
}
```

Adicione a biblioteca do driver JDBC MySQL no projeto



Corrija as importações, cuidando para escolher a importação de java.sql e não da com.mysql.jdbc.

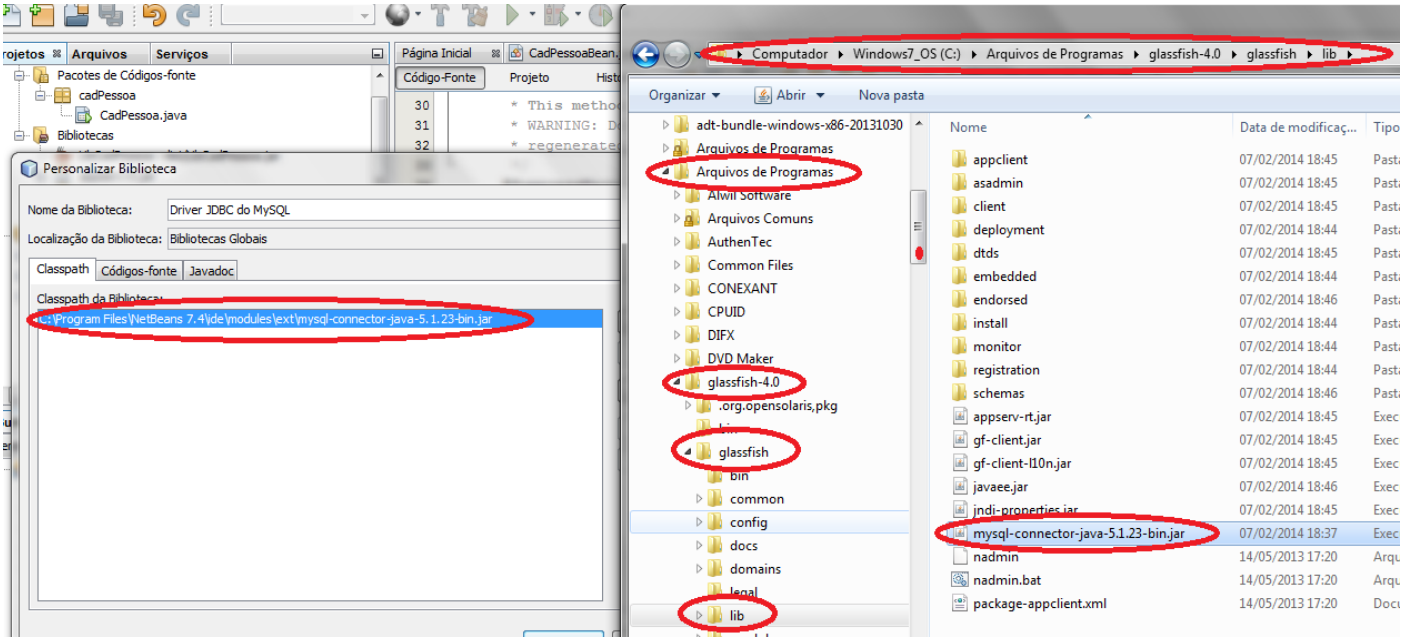


```
18      @Override
        public boolean CadastrarPessoa(String cpf, String
20          try {
21
                Connection con = DriverManager.getConnecti
24          Adicionar importação para java.sql.DriverManager
25          Criar class "DriverManager" no pacote com.evandro
            Criar class "DriverManager" em com.evandro.CadPessoaBean
```

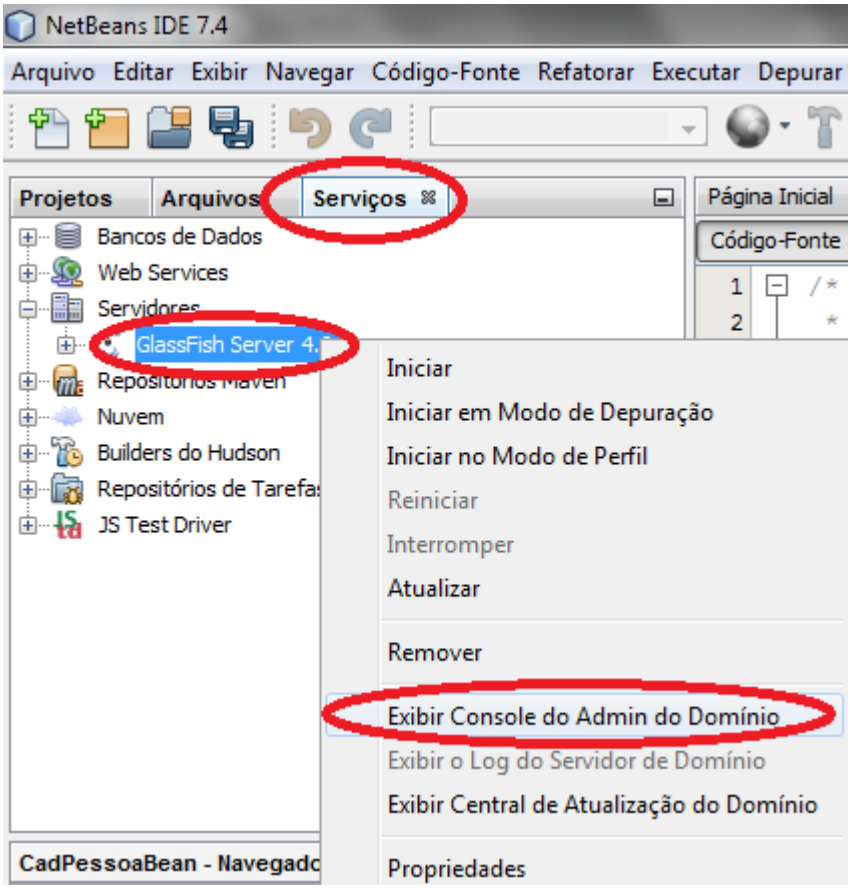
```
23              Connection con = DriverManager.getConnect
                PreparedStatement pstmt = con.prepareState
25          Adicionar importação para com.mysql.jdbc.PreparedStatement
26          Adicionar importação para java.sql.PreparedStatement
27          Criar class "PreparedStatement" no pacote com.evandro
28          Criar class "PreparedStatement" em com.evandro.CadPessoaBean
```

```
            } catch (SQLException ex) {
25          Adicionar importação para java.sql.SQLException
26          Criar class "SQLException" no pacote com.evandro
27          Criar class "SQLException" em com.evandro.CadPessoaBean
28      }
```

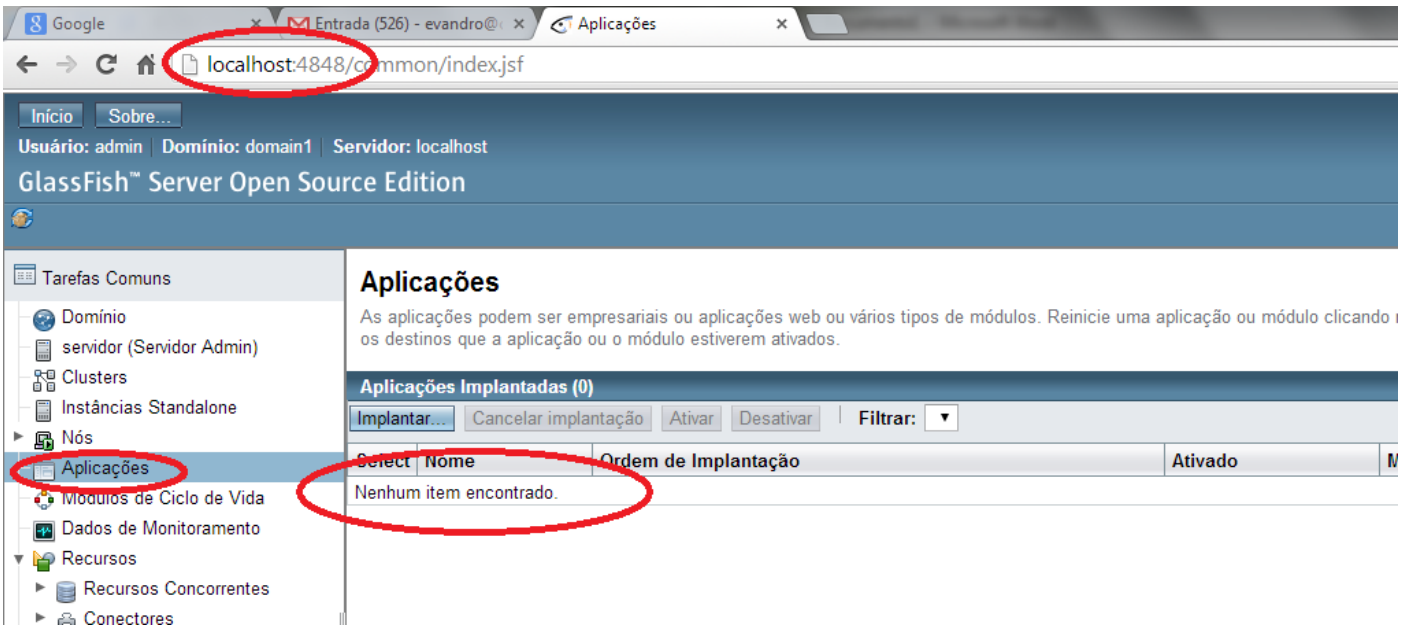
Copie o arquivo mysql-connector-java-5.1.23-bin.jar para a pasta lib do GlassFish.



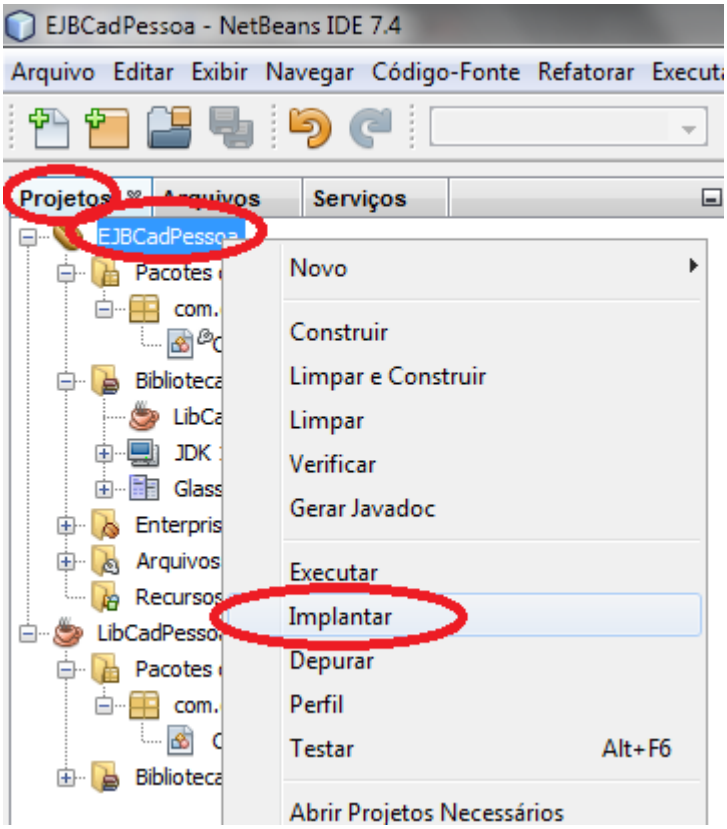
Acesse o Console do Admin do Domínio do GlassFish



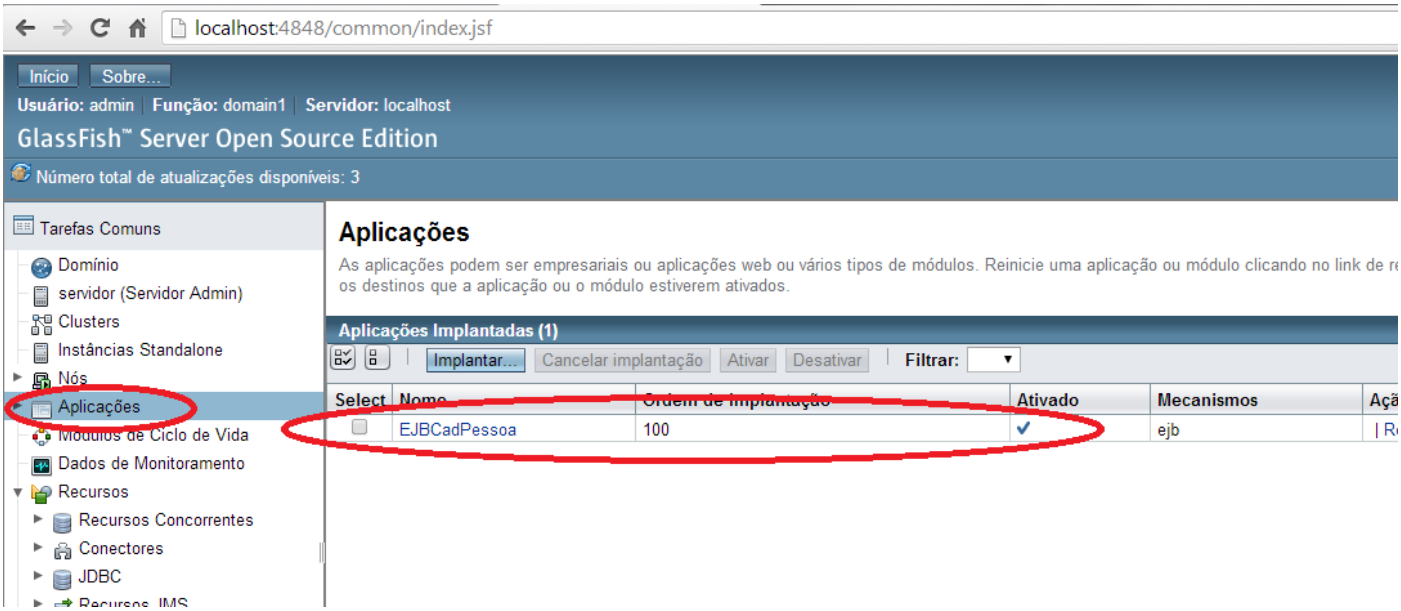
Se não funcionar, clique em Iniciar e depois acesse diretamente pelo navegador: http://localhost:4848/



Implante o projeto EJBCadPessoa no servidor GlassFish



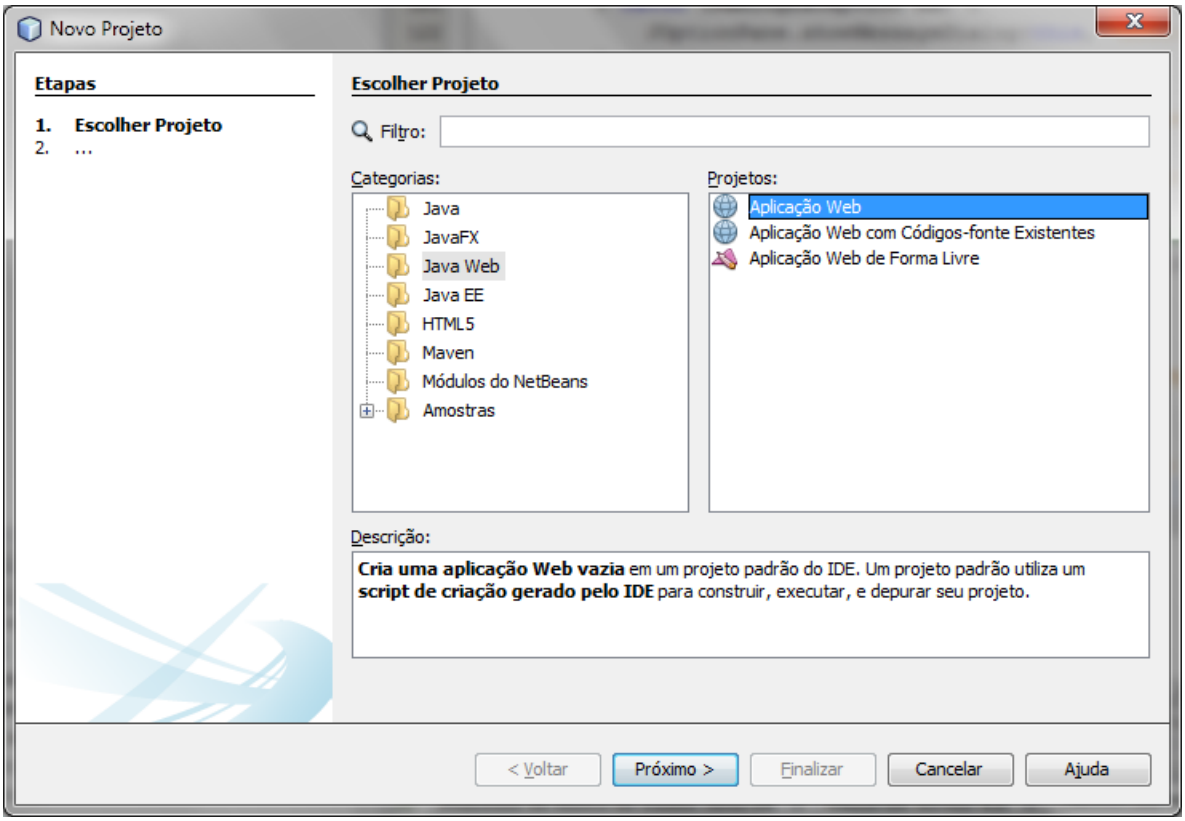
Verifique se ele foi implantado e está ativado

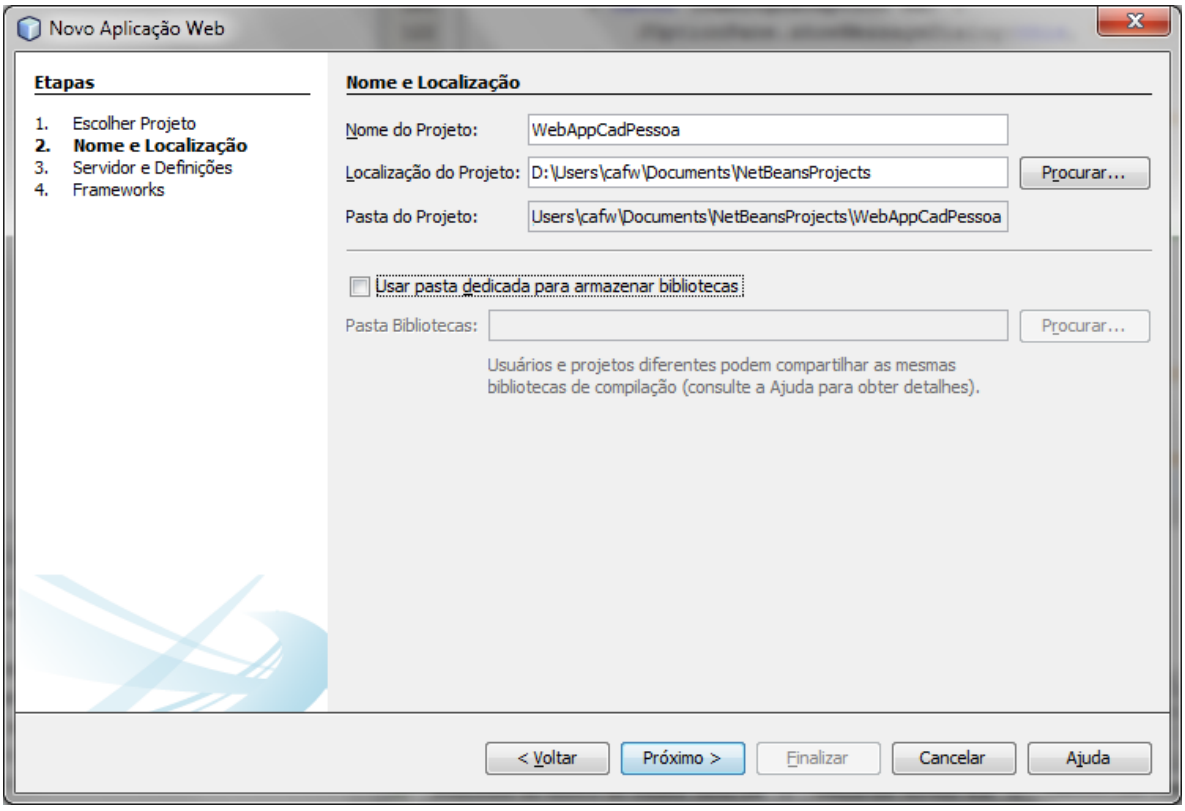


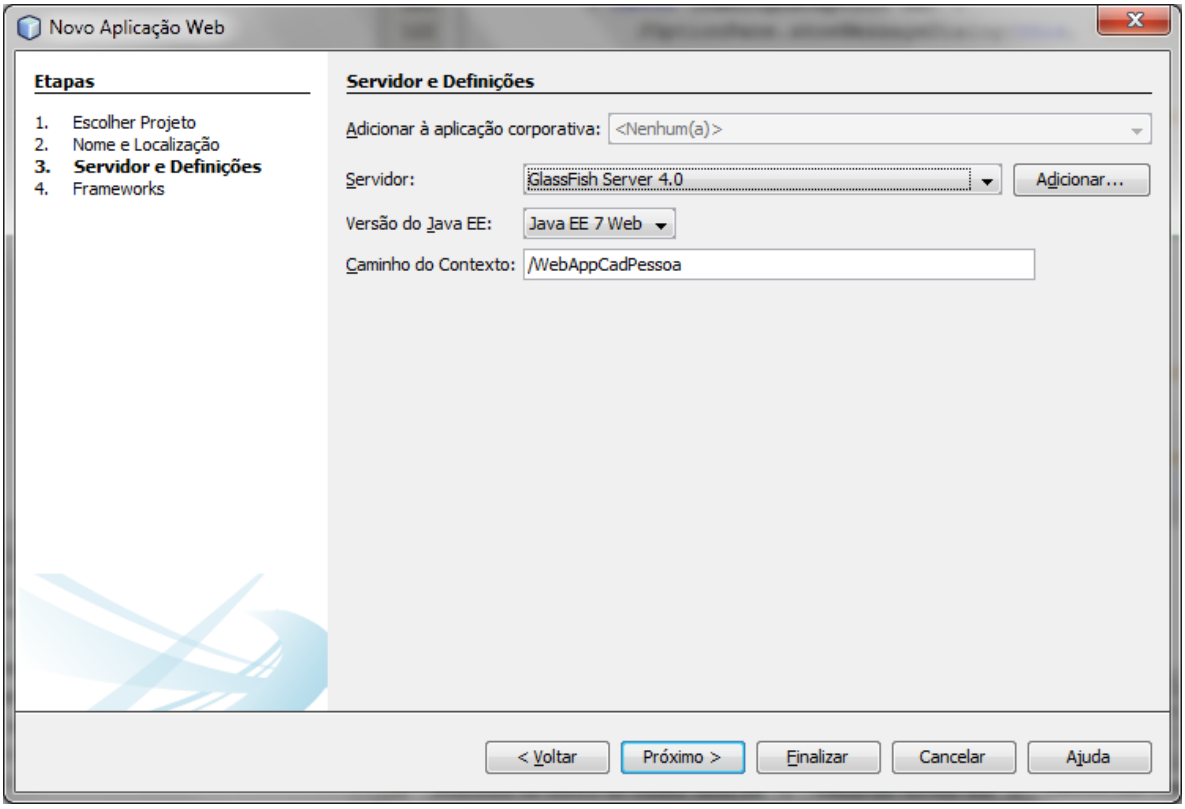
3. Criando uma Aplicação Web com JSP para acessar o EJB

O EJB também pode ser acessado a partir de uma aplicação WEB Java com JSP.

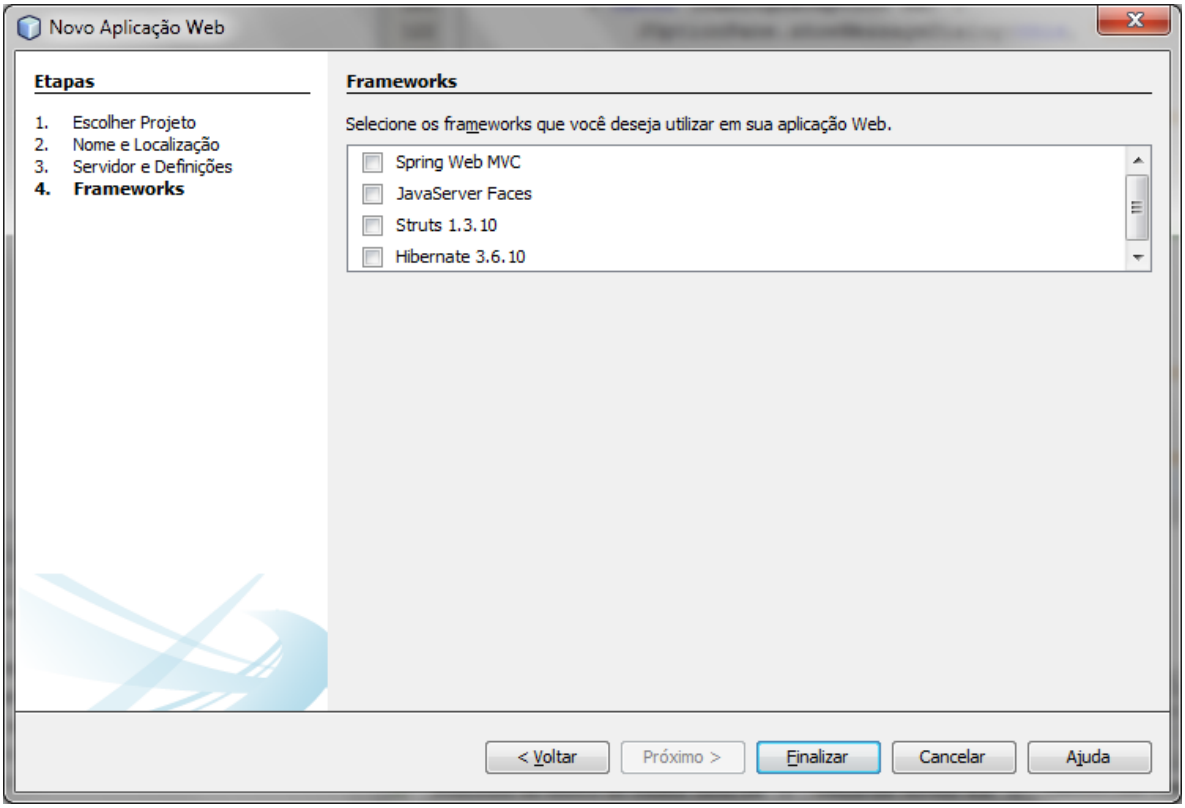
Para isso, crie um Novo Projeto => Java Web => Aplicação Web com o nome WebAppCadPessoa



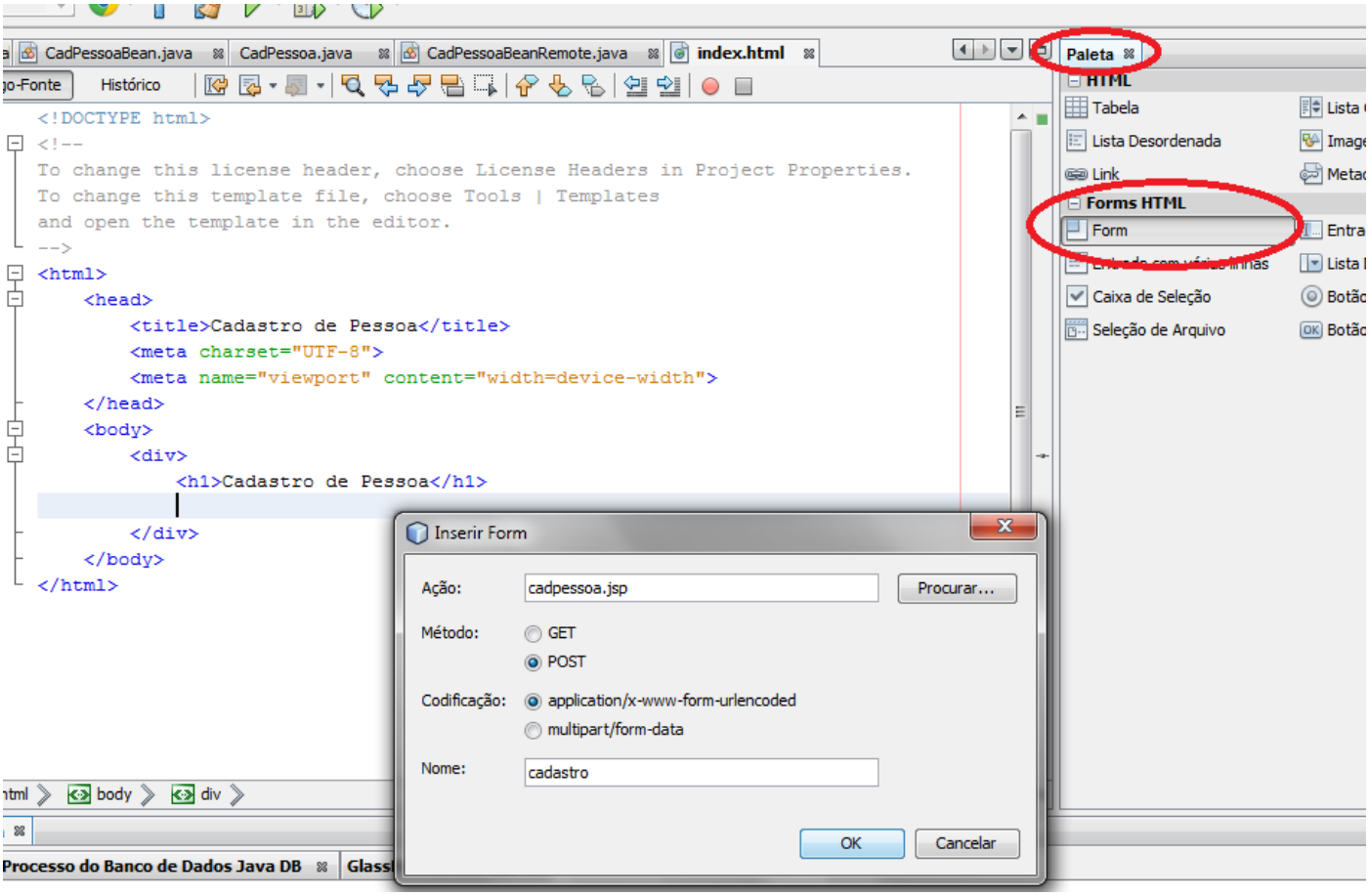




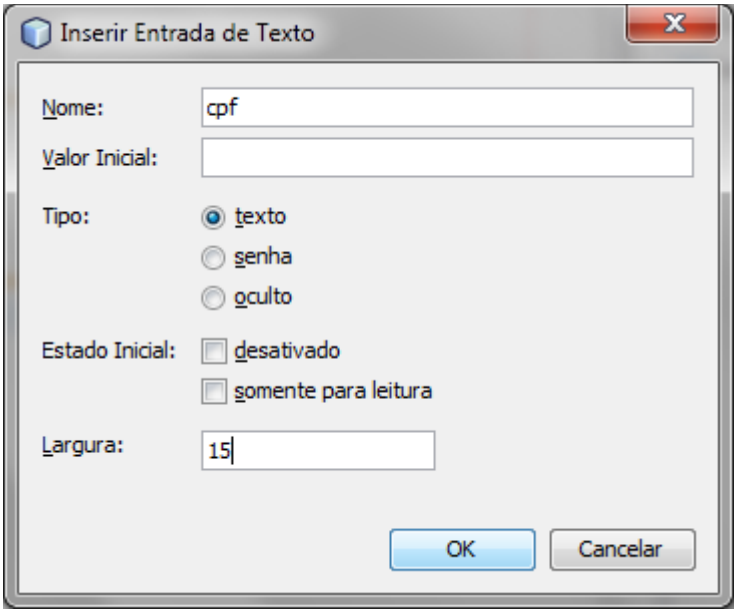
Deixe desmarcado os frameworks...

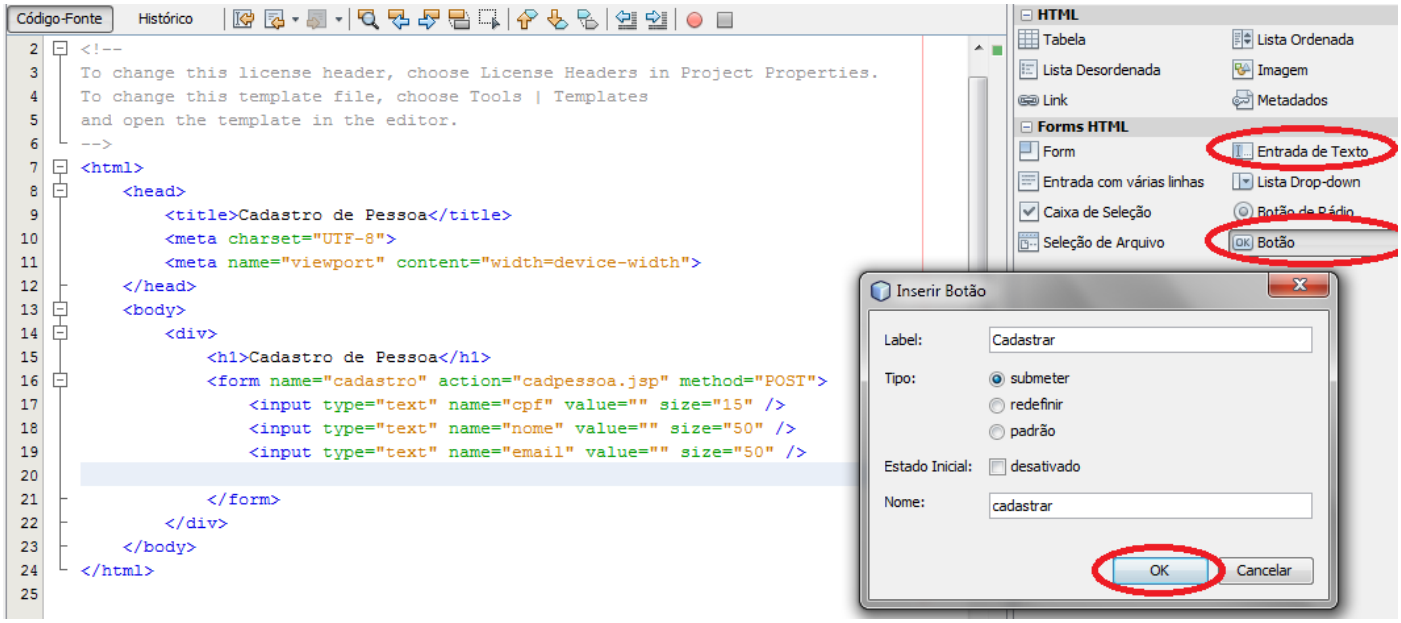


Altere o código HTML da página inicial (index.html) alterando o título e criando um formulário, que chamará o cadpessoa.jsp



Insira 3 campos de Entrada de Texto (cpf, nome, email) e um botão (submit) para fazer o envio dos dados e realizar o cadastro.



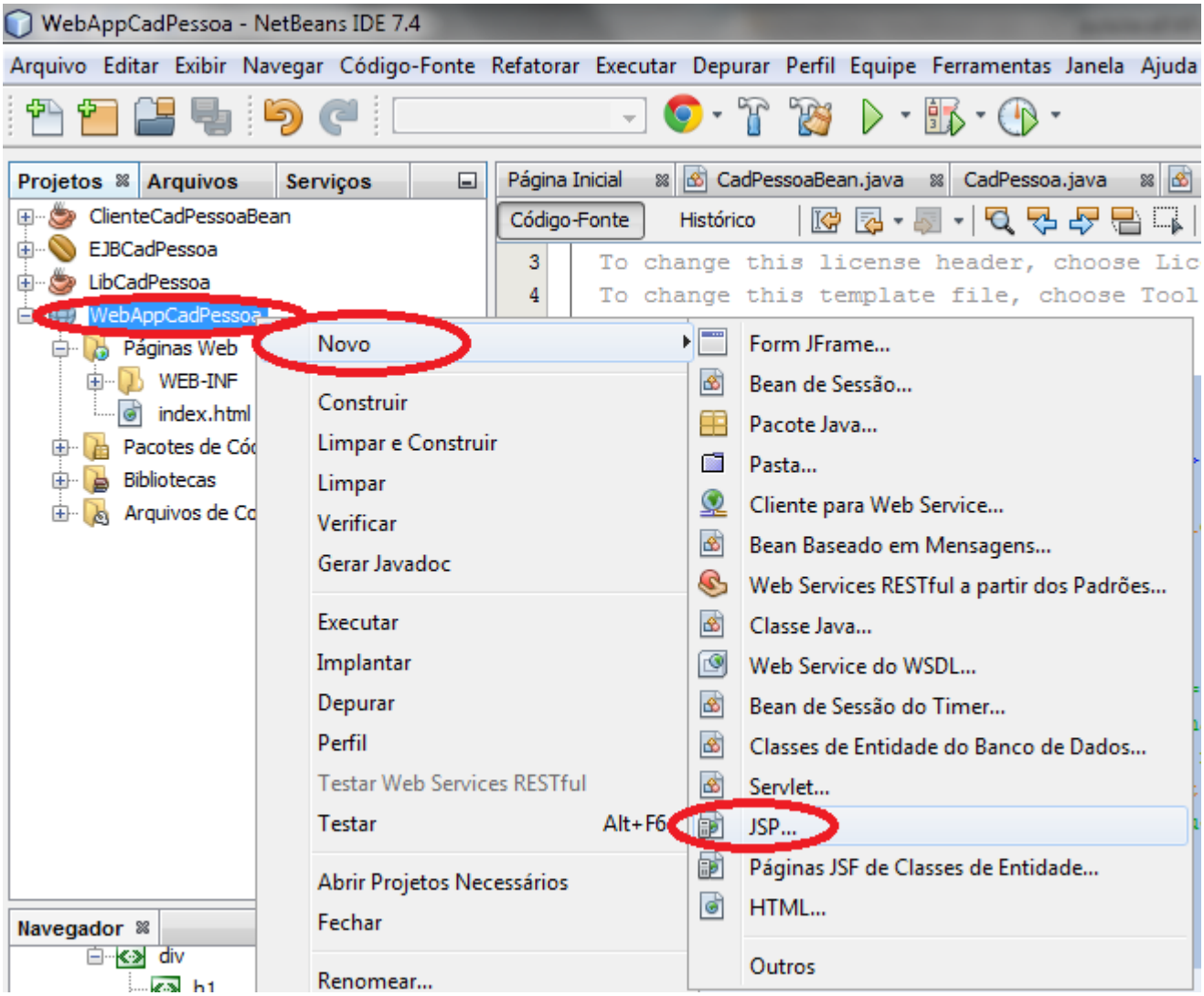


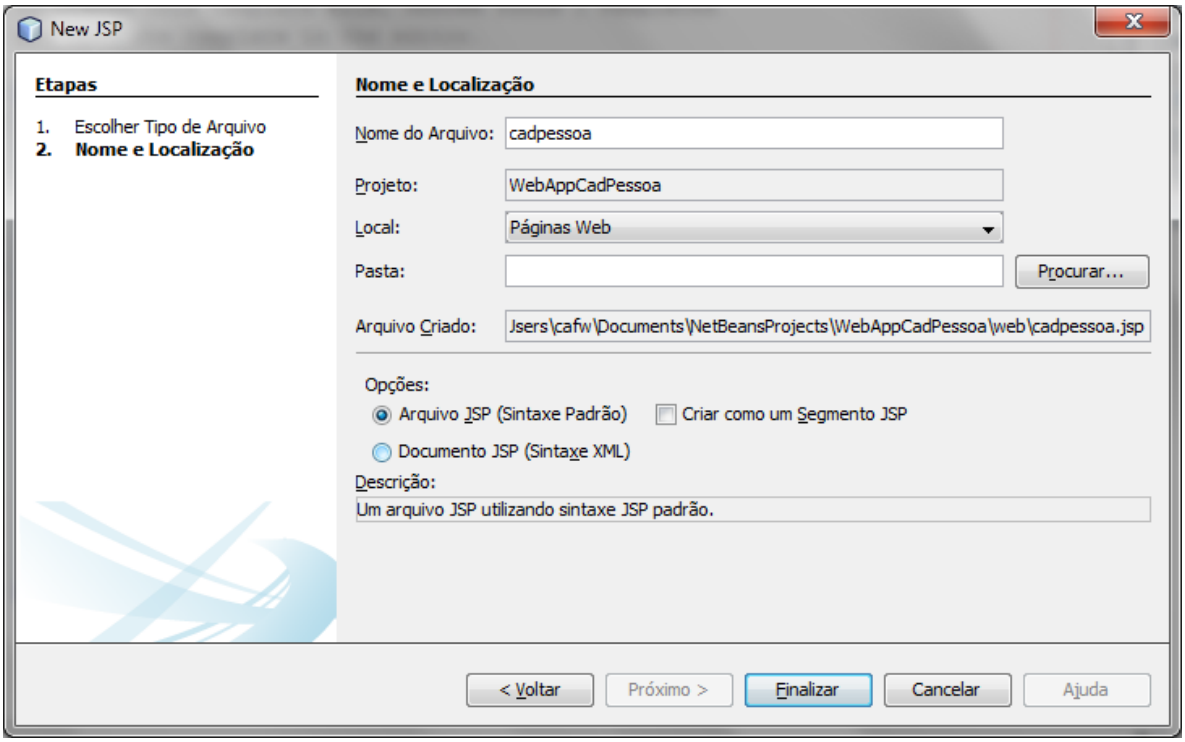
O código deve ficar semelhante a esse:

```
<html>
    <head>
        <title>Cadastro de Pessoa</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div>
            <h1>Cadastro de Pessoa</h1>
            <form name="cadastro" action="cadpessoa.jsp" method="POST">
                CPF: <input type="text" name="cpf" value="" size="15" /><br>
                Nome: <input type="text" name="nome" value="" size="50" /><br>
                E-Mail: <input type="text" name="email" value="" size="50" /><br>
                <input type="submit" value="Cadastrar" name="cadastrar" /><br>
            </form>
        </div>
    </body>
</html>
```

Vamos criar a página JSP que receberá os dados e fará a chamada do EJB para realizar o cadastro.

Sobre o WebAppCadPessoa, clique com o botão direito e escolha Novo => JSP com o nome cadpessoa


WebAppCadPessoa - NetBeans IDE 7.4
Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda
Projetos
Arquivos
Serviços
ClienteCadPessoaBean
EJBCadPessoa
LibCadPessoa
WebAppCadPessoa
Páginas Web
WEB-INF
index.html
Pacotes de Cód
Bibliotecas
Arquivos de Co
Página Inicial
CadPessoaBean.java
CadPessoa.java
Código-Fonte
Histórico
3 To change this license header, choose Lic
4 To change this template file, choose Tool
Novo
Construir
Limpar e Construir
Limpar
Verificar
Gerar Javadoc
Executar
Implantar
Depurar
Perfil
Testar Web Services RESTful
Testar Alt+F6
Abrir Projetos Necessários
Fechar
Renomear...
Form JFrame...
Bean de Sessão...
Pacote Java...
Pasta...
Cliente para Web Service...
Bean Baseado em Mensagens...
Web Services RESTful a partir dos Padrões...
Classe Java...
Web Service do WSDL...
Bean de Sessão do Timer...
Classes de Entidade do Banco de Dados...
Servlet...
JSP...
Páginas JSF de Classes de Entidade...
HTML...
Outros
Navegador
div
h1

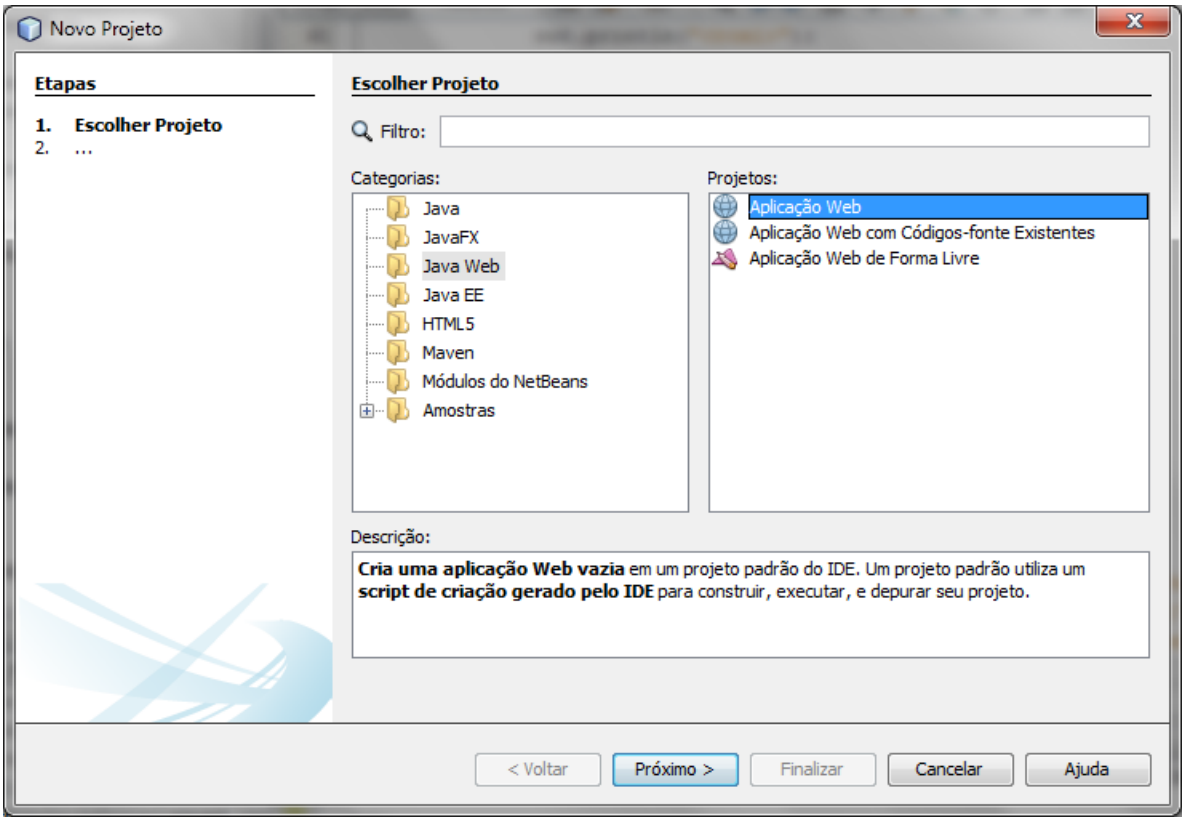

New JSP
Etapas
1. Escolher Tipo de Arquivo
2. Nome e Localização
Nome e Localização
Nome do Arquivo: cadpessoa
Projeto: WebAppCadPessoa
Local: Páginas Web
Pasta:
Procurar...
Arquivo Criado: Jsers\cafw\Documents\NetBeansProjects\WebAppCadPessoa\web\cadpessoa.jsp
Opções:
Arquivo JSP (Sintaxe Padrão)
Criar como um Segmento JSP
Documento JSP (Sintaxe XML)
Descrição:
Um arquivo JSP utilizando sintaxe JSP padrão.
< Voltar
Próximo >
Finalizar
Cancelar
Ajuda

Altere o cadpessoa.jsp para fazer a chamada do EJB `EJBCadPessoa/CadPessoaBean`. Essa página envia um retorno para a página web sob a forma de script com um alerta informando se o cadastro foi realizado ou não e mostra novamente o mesmo formulário de cadastro, para fazer novas inserções.

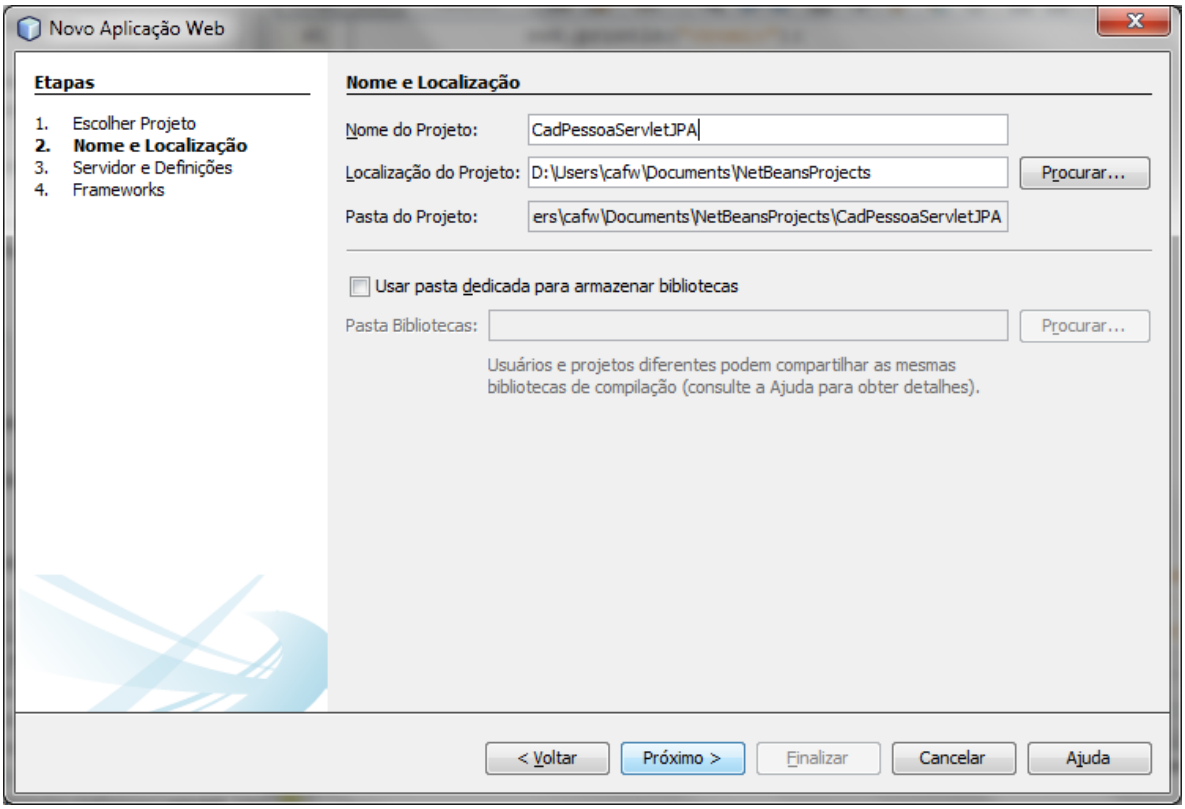
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cadastro de Pessoa</title>
    </head>
    <body>
        <h1>Cadastro de Pessoa</h1>
        <div>
            <%@page language = "java" %>
            <%@page import = "java.util.*" %>
            <%@page import = "com.evandro.CadPessoaBeanRemote" %>
            <%@page import = "java.util.Properties" %>
            <%@page import = "java.util.logging.Level" %>
            <%@page import = "java.util.logging.Logger" %>
            <%@page import = "javax.naming.InitialContext" %>
            <%@page import = "javax.naming.NamingException" %>

            <%
                if (request.getParameter("nome") != null) {
                    InitialContext ctx = null;
                    CadPessoaBeanRemote cp;
                    cp = null;
                    try {
                        ctx = new InitialContext(); // ser for aplicação local ou
                /*   o codigo a seguir se for aplicação remota
                         Properties p = new Properties();
                         p.put("org.omg.CORBA.ORBInitialHost", "192.168.1.175");
                         p.put("org.omg.CORBA.ORBInitialPort", "3700");
                         ctx = new InitialContext(p); */
                    } catch (NamingException ex) {
                        out.println("<script>alert('Erro: " + ex + "');</script>");
                    }
                    try {
                        cp = (CadPessoaBeanRemote)
ctx.lookup("java:global/EJBCadPessoa/CadPessoaBean");
                    } catch (NamingException ex) {
                        out.println("<script>alert('Erro: " + ex + "');</script>");
                    }
                    if (cp != null && cp.CadastrarPessoa(request.getParameter("cpf"),
request.getParameter("nome"), request.getParameter("email"))) {
                        out.println("<script>alert('" + request.getParameter("nome") + "
Cadastrado.');</script>");
                        try {
                            ctx.close();
                        } catch (NamingException ex) {
                            out.println("<script>alert('Erro: " + ex + "');</script>");
                        }
                    } else {
                        out.println("<script>alert('Erro no cadastro.');</script>");
                    }
                }
            %>
        </div>
        <div>
            <form name="cadastro" action="cadpessoa.jsp" method="POST">
                CPF: <input type="text" name="cpf" value="" size="15" /><br>
                Nome: <input type="text" name="nome" value="" size="50" /><br>
                E-Mail: <input type="text" name="email" value="" size="50" /><br>
                <input type="submit" value="Cadastrar" name="cadastrar" /><br>
            </form>
        </div>

    </body>
</html>
```

4. Criando uma Aplicação Web com servlet e JPA para fazer o cadastro de Pessoa

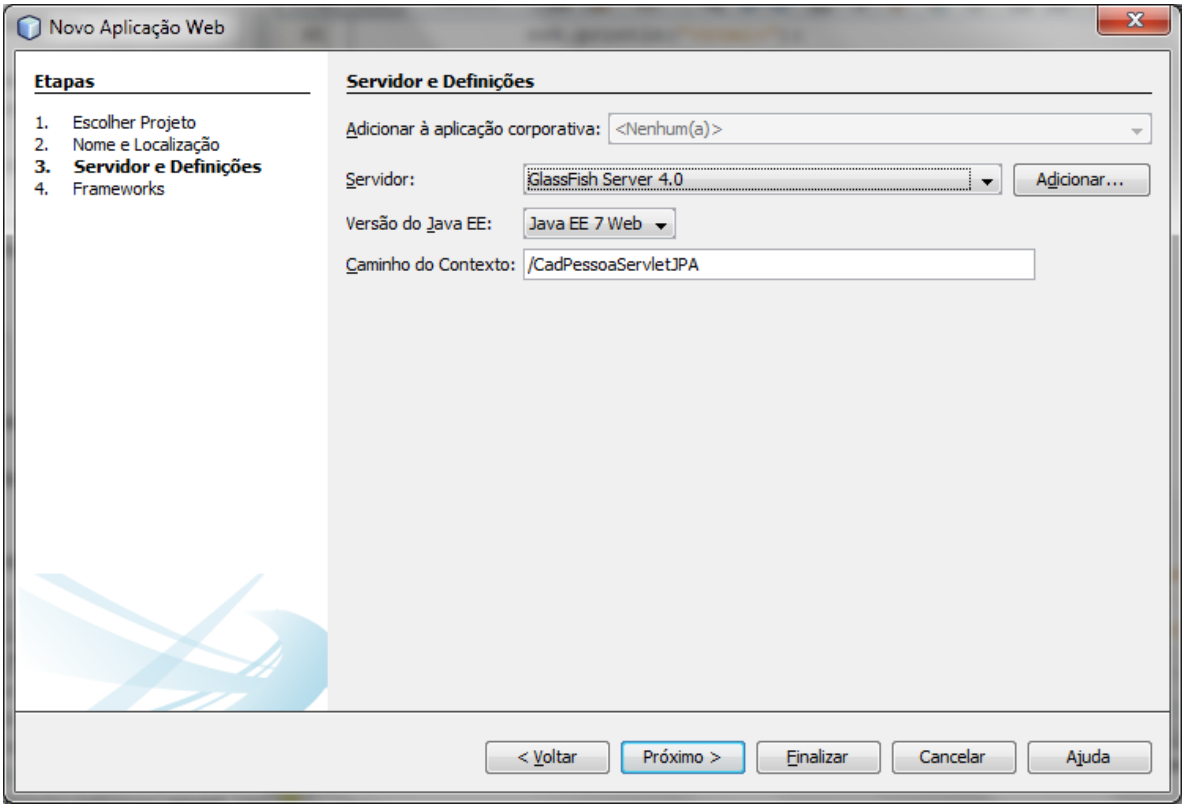
Crie um novo projeto: Java Web => Aplicação Web



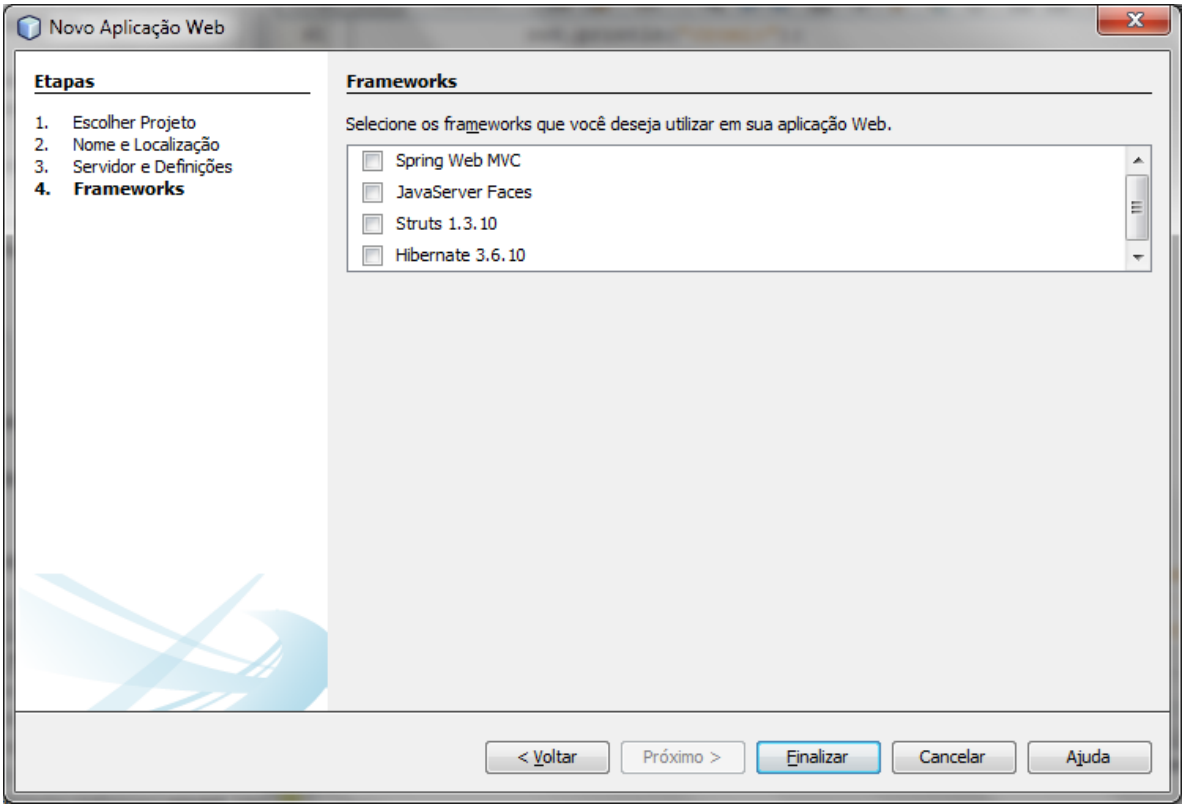
Dê o nome de CadPessoaServletJPA, e clique em Próximo



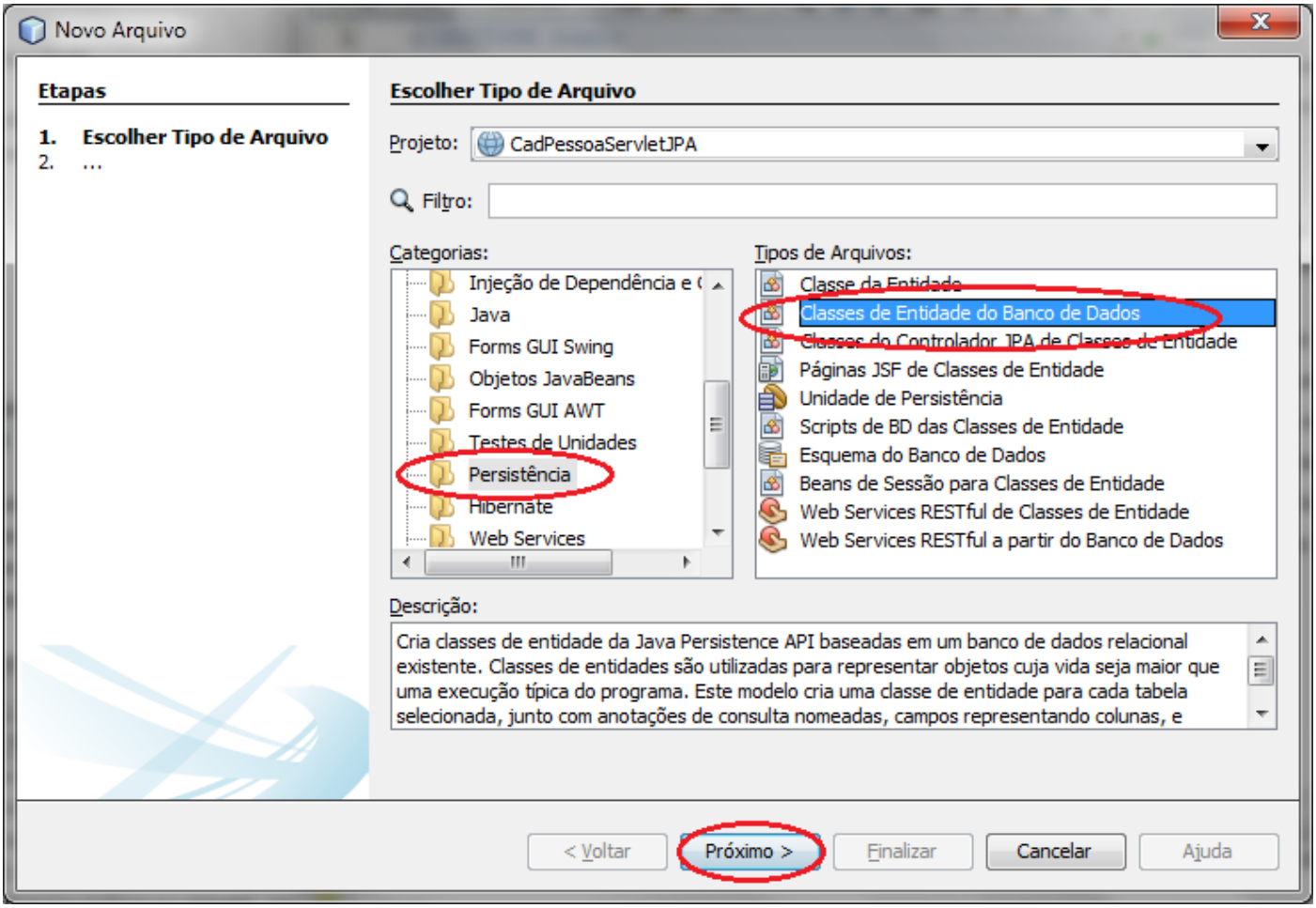
Escolha o servidor GlassFish e clique em Próximo



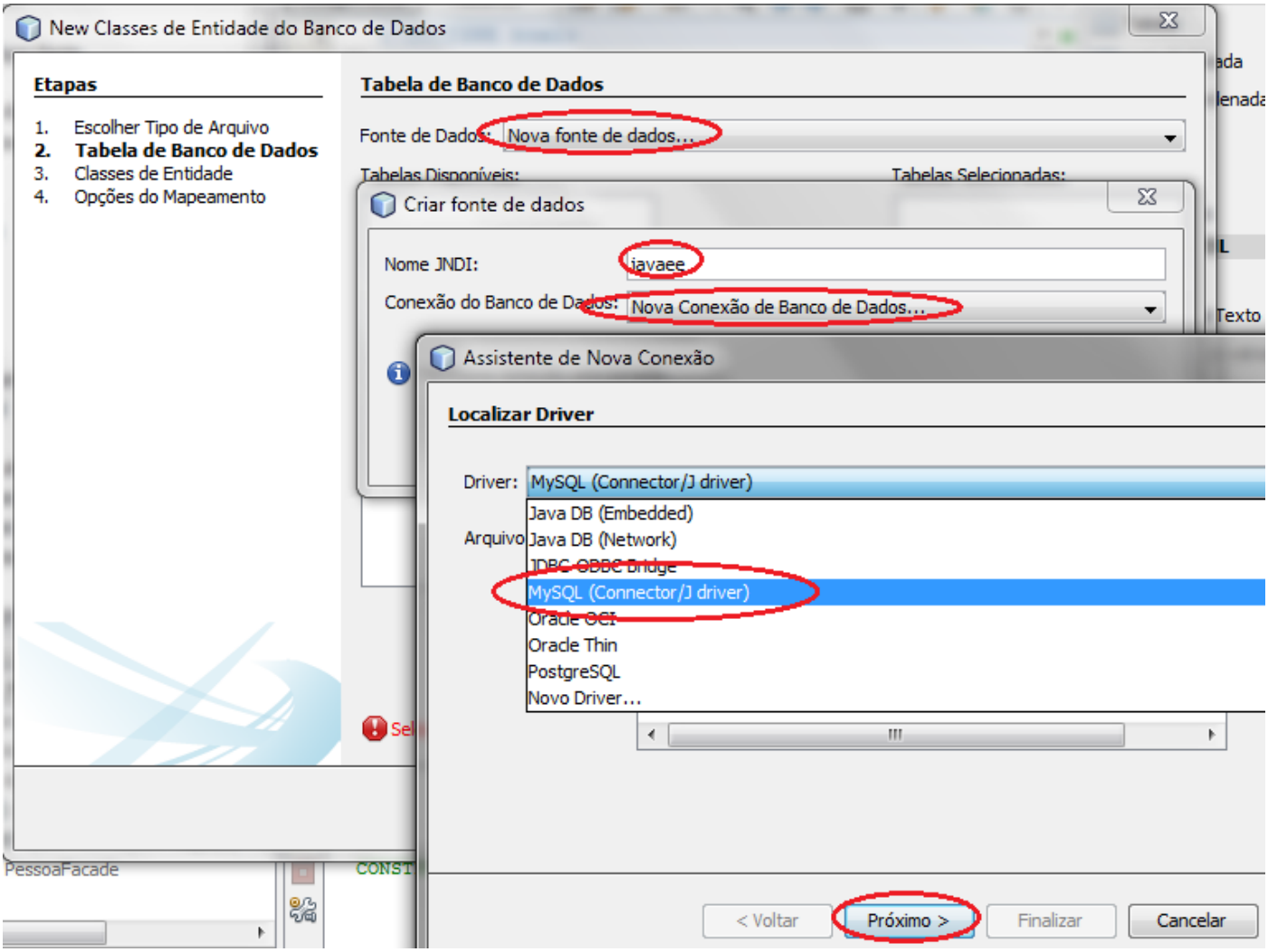
não escolha nenhum framework, e clique em Finalizar



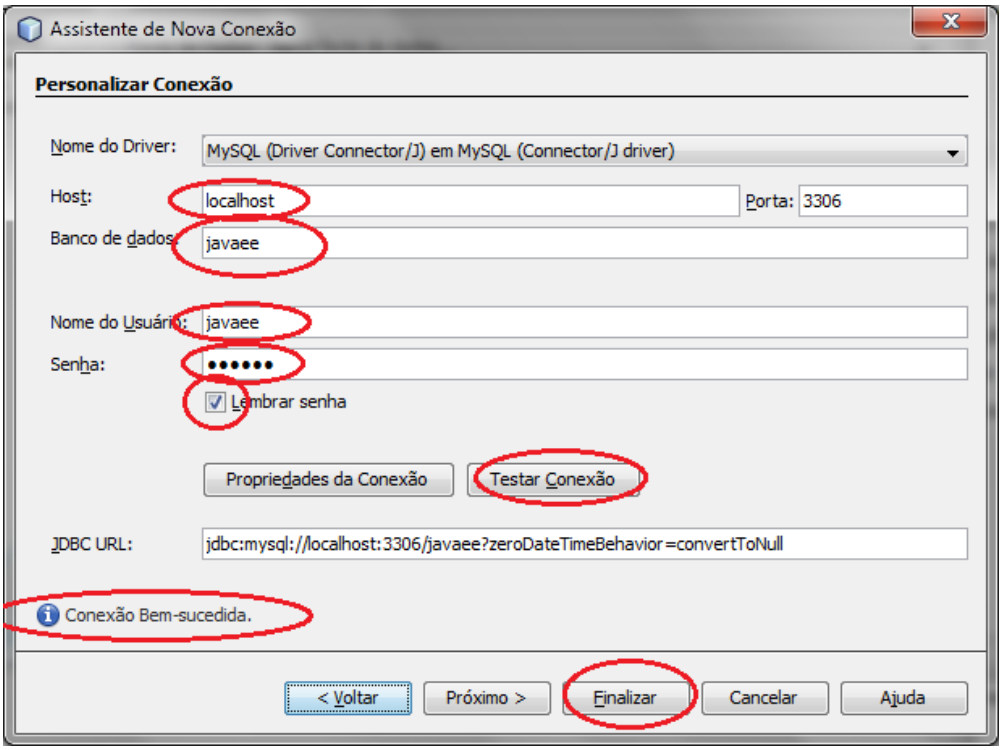
Clique sobre o projeto e escolha Novo => Outros => Persistência => Classes de Entidade de Banco de Dados, clique em Próximo



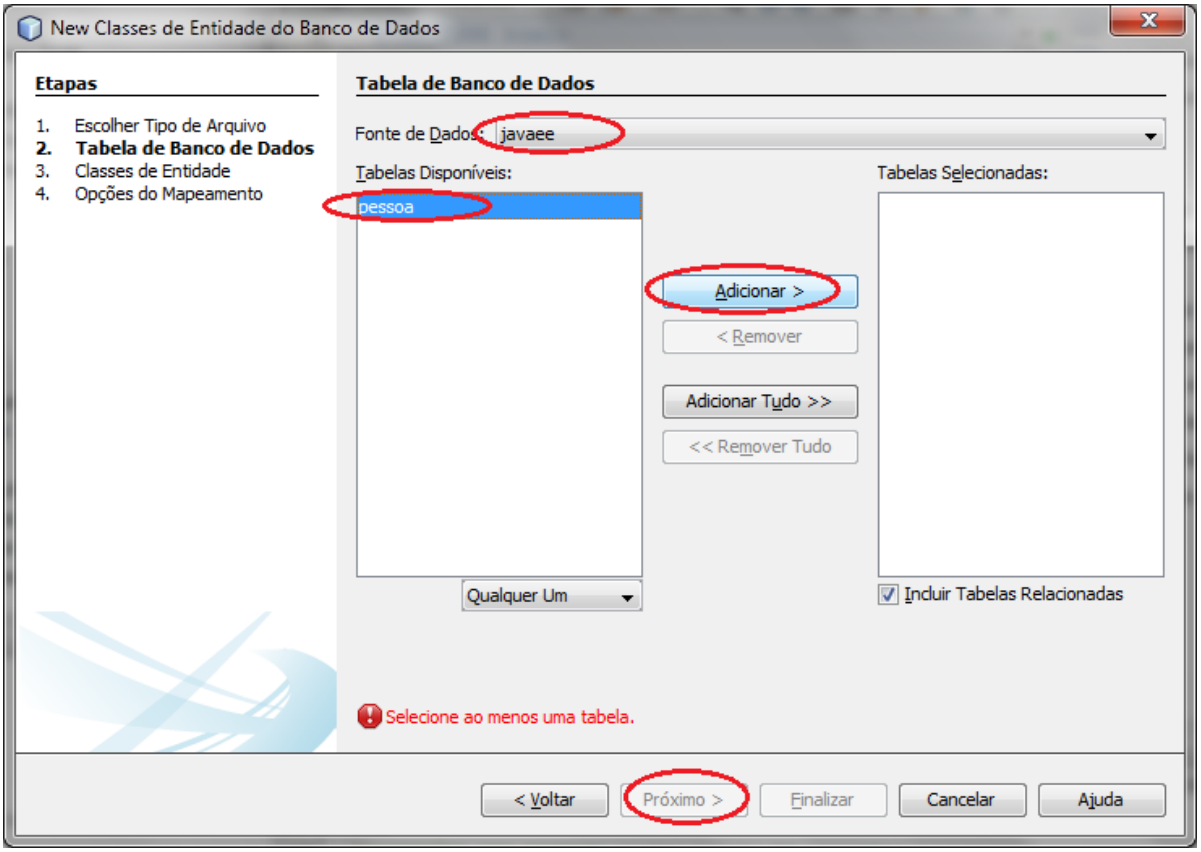
Escolha Nova Fonte de Dados => Coloque o nome javaee => Nova Conexão de Banco de Dados => Driver MySQL (Connector/J driver) => Próximo



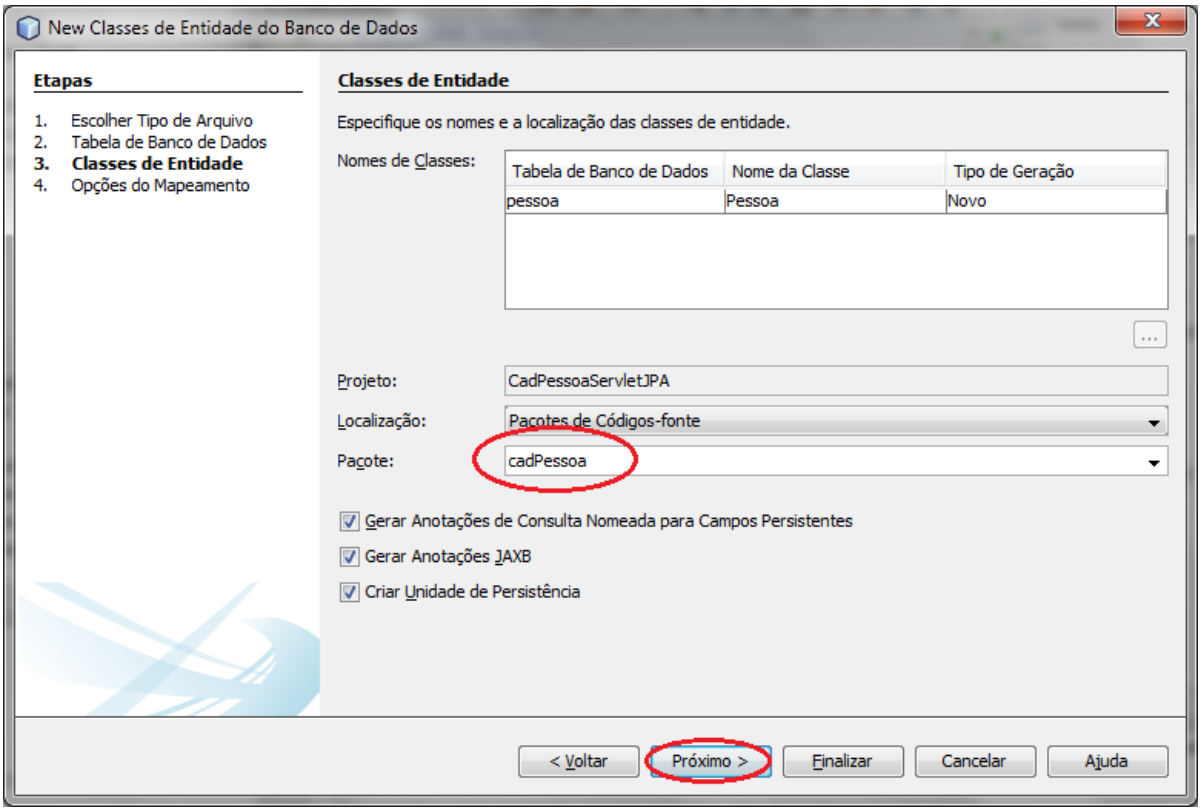
Preencha os parâmetros do SGBD e clique em Testar Conexão... depois em Finalizar



Escolha a Fonte de Dados: javaee, clique sobre a Tabela Disponível: pessoa => Adicionar => Próximo

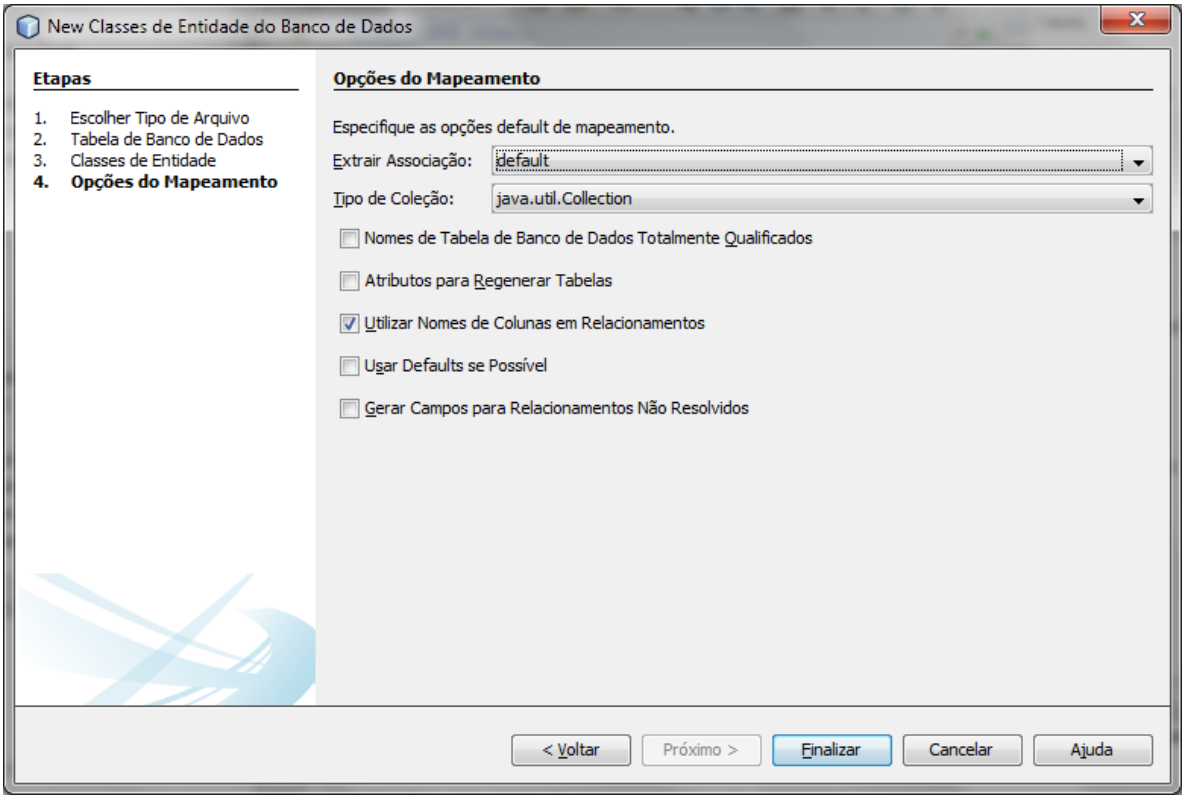


Dê o nome do pacote de cadPessoa, clique em próximo

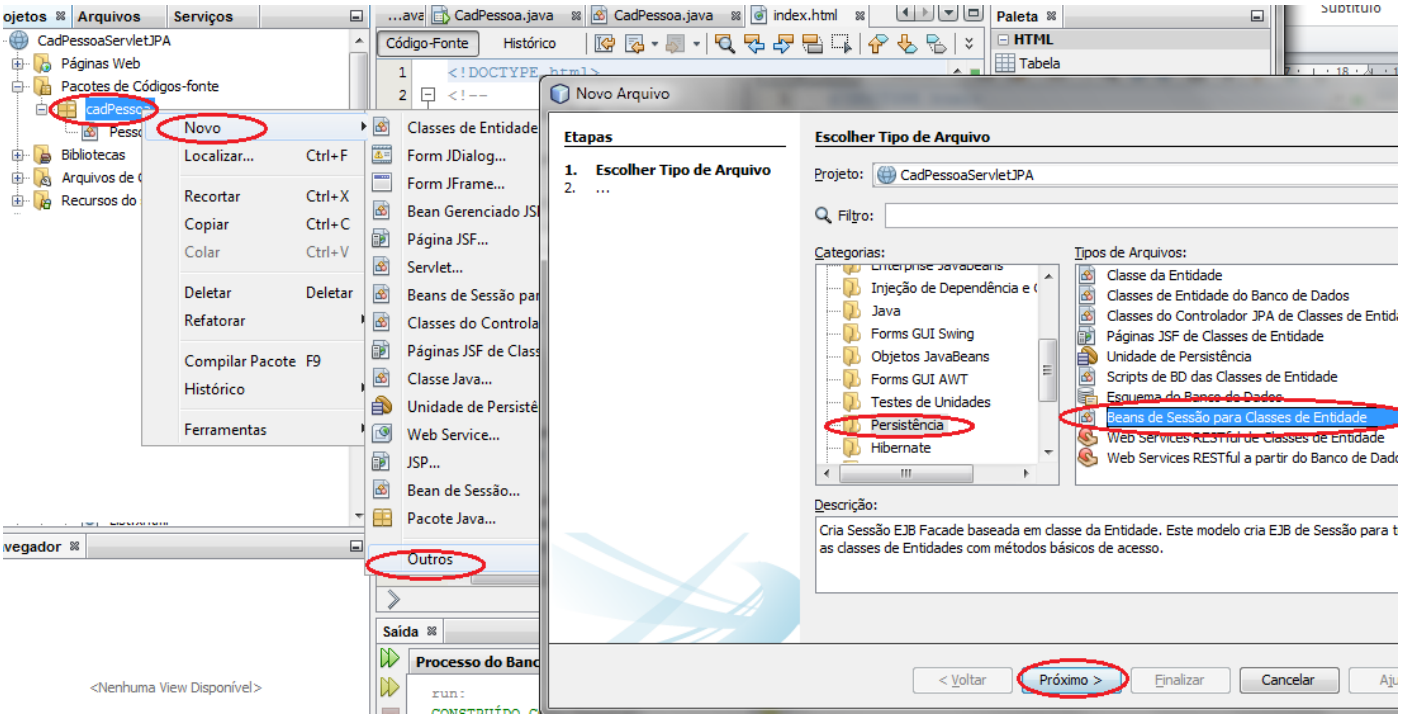


Na tela seguinte, deixe os valores padrão e clique em Finalizar.

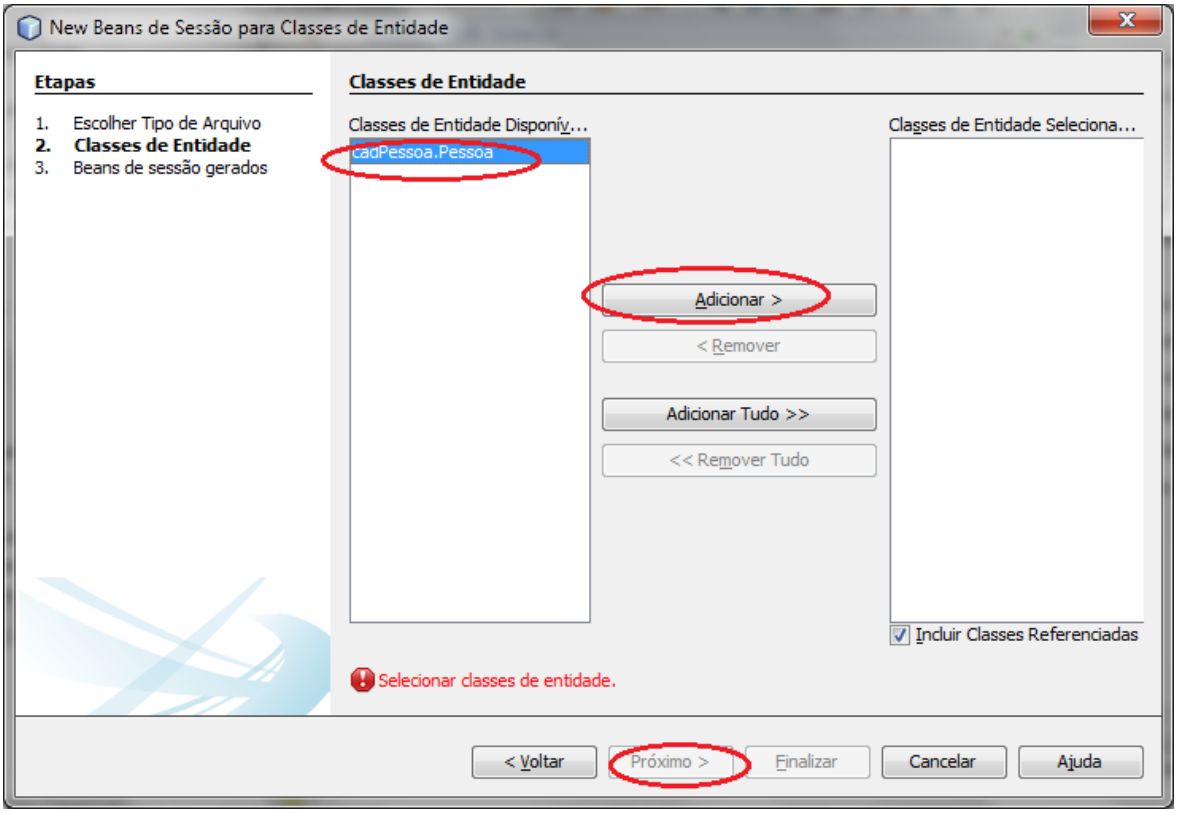
Isso vai criar a classe Pessoa.java



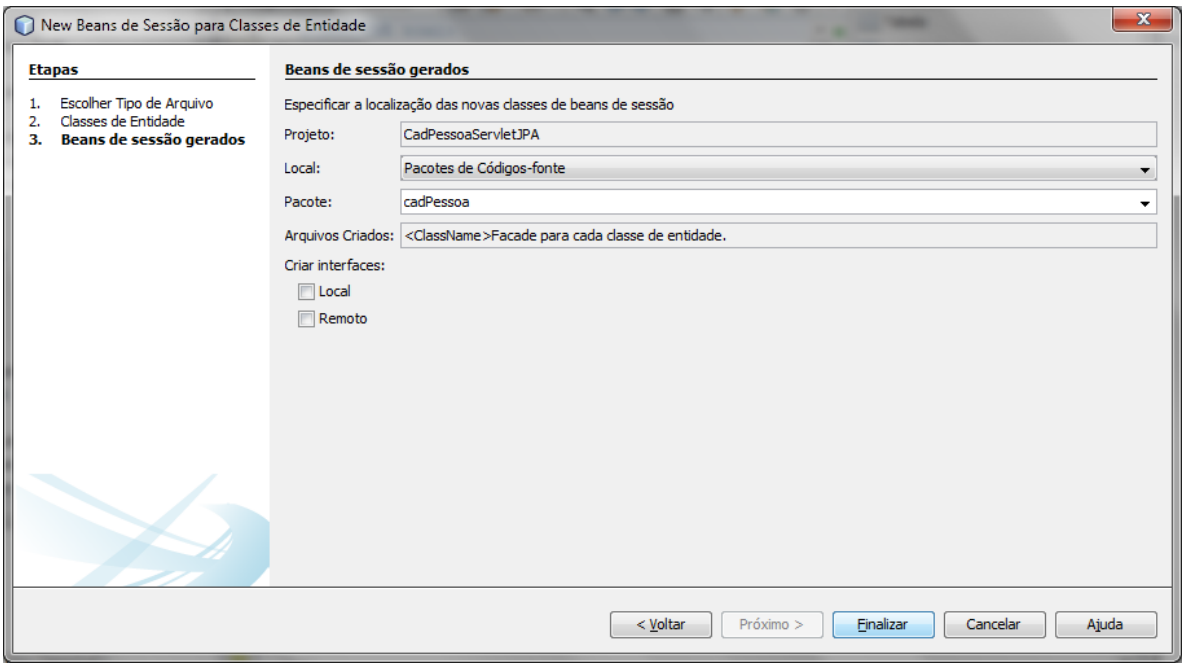
Clique com o botão direito sobre o pacote cadPessoa => Novo => Outros => Persistência => Beans de Sessão para Classes de Entidade => Próximo



Clique sobre a classe de Entidade Disponível: cadPessoa.Pessoa => Adicionar => Próximo

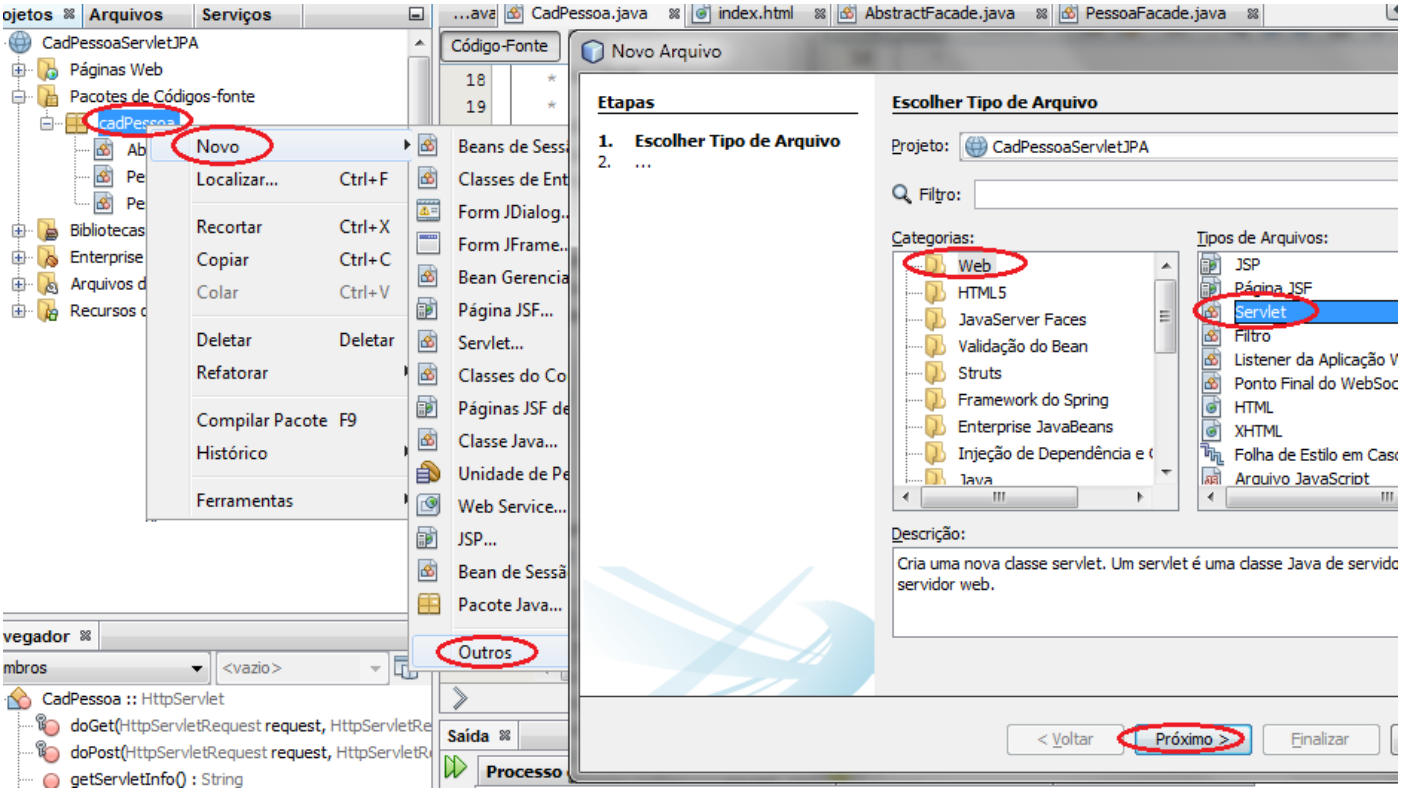


Mantenha os valores padrão e clique em Finalizar

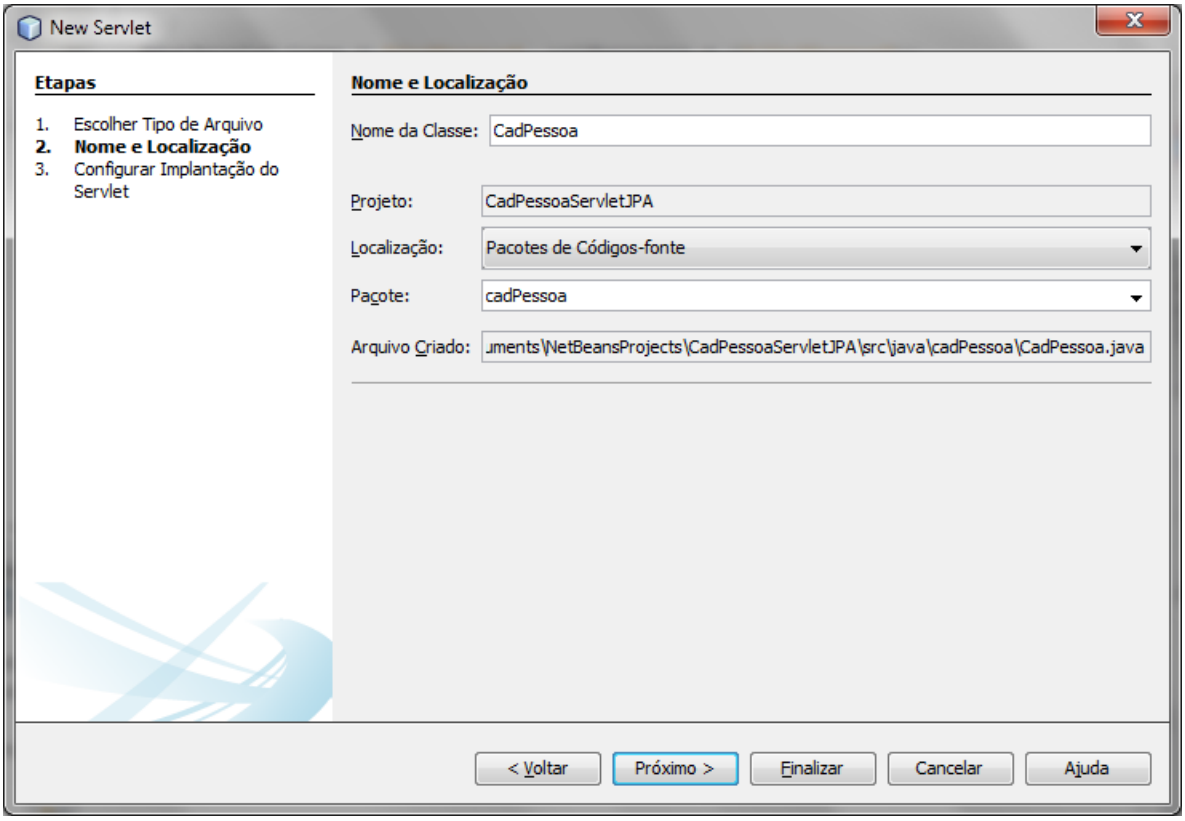


Isso vai criar a classe PessoaFacade.java, que é a "fachada" para acessar a classe Pessoa.

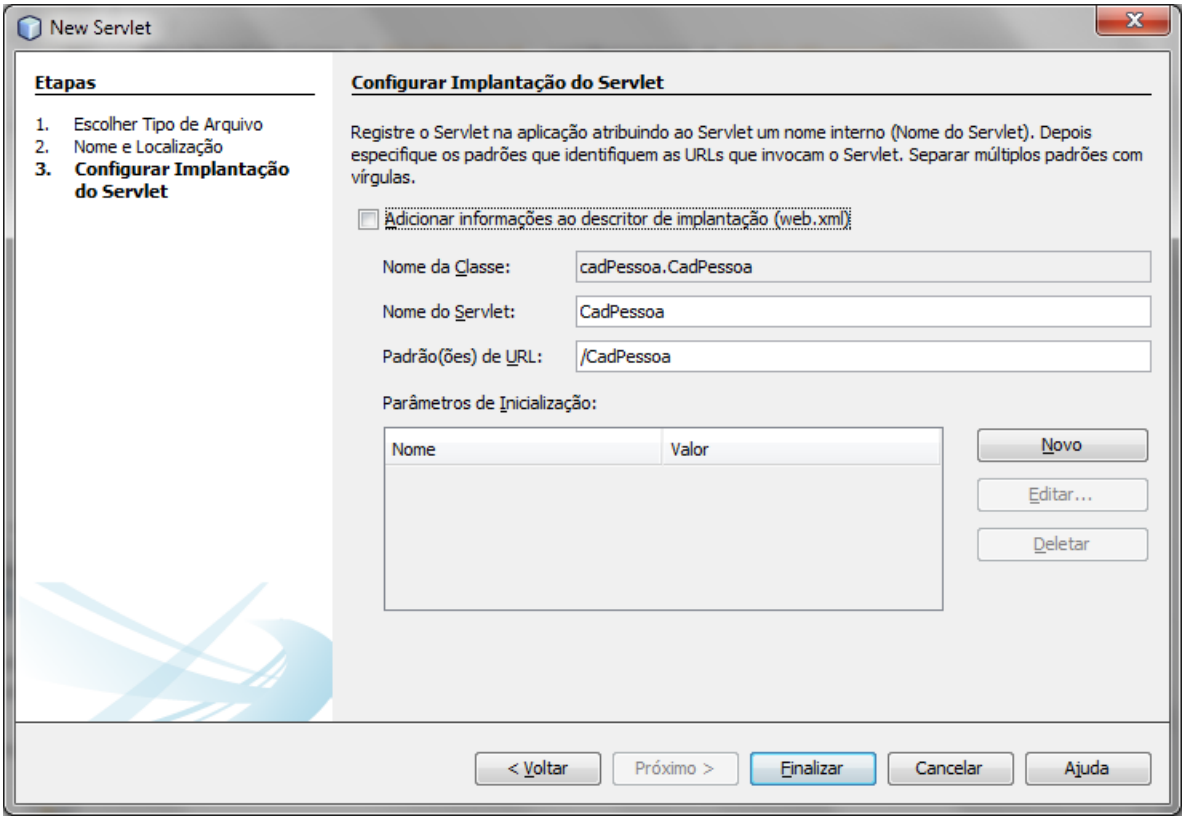
Agora vamos criar um servlet para fazer o cadastro. Clique sobre o pacote cadPessoa => Novo => Outros => Web => Servlet => Próximo



Dê o nome CadPessoa, e clique em Próximo



mantenha os valores padrão e clique em Finalizar



Inclua a anotação @EJB que cria um Bean pessoaFacade, bem no início da classe:

```
public class CadPessoa extends HttpServlet {

    @EJB
    private PessoaFacade pessoaFacade;
```

e corrija a importação para javax.ejb.EJB

altere o método para que fique com o seguinte código:

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Cadastro de Pessoa</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Cadastro de Pessoas</h1>");
            if (request.getParameter("cpf") != null) {
                Pessoa p = new Pessoa();
                p.setCpf(request.getParameter("cpf"));
                p.setNome(request.getParameter("nome"));
                p.setEmail(request.getParameter("email"));
                try {
                    pessoaFacade.create(p);
                    out.println("Cadastro com sucesso.");
                } catch (Exception e) {
                    out.println("Erro no Cadastro.");
                }
            }
            out.println("<hr>");
            out.println("<form name=\"cadastro\" action=\"CadPessoa\" method=\"POST\">");
            out.println("CPF: <input type=\"text\" name=\"cpf\" value=\"\" size=\"15\" /><br>");
```

```
            out.println("Nome: <input type=\"text\" name=\"nome\" value=\"\" size=\"50\"
/><br>");
            out.println("E-Mail: <input type=\"text\" name=\"email\" value=\"\" size=\"50\"
/><br>");
            out.println("<input type=\"submit\" value=\"Cadastrar\" name=\"cadastrar\" /><br>");
            out.println("</form>");
            out.println("</body>");
            out.println("</html>");
        }
```

Clique sobre o projeto e escolha Implantar

Clique sobre o servlet CadPessoa e escolha Executar Arquivo (não precisa informar parâmetros adicionais). Ou acesse diretamente através do browser: http://localhost:8080/CadPessoaServletJPA/CadPessoa

Também poderíamos fazer um servlet que mostra um registro do cadastro de pessoa ou todos os registros usando JPA.

O seguinte código mostra um registro do cadastro, enviado através do parametro id e depois mostra todos os registros da tabela.

```
 @WebServlet(name = "MostraPessoa", urlPatterns = {"/MostraPessoa"})
public class MostraPessoa extends HttpServlet {

    @EJB
    private PessoaFacade pessoaFacade;

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet MostraPessoa</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet MostraPessoa at " + request.getContextPath() + "</h1>");
            if (request.getParameter("id") != null) {
                Pessoa p = pessoaFacade.find(request.getParameter("id"));
                out.println("<hr>");
                out.println("Procurando pelo CPF: " + request.getParameter("id") + "<br>");

                if (p != null) {
                    out.println("CPF: " + p.getCpf() + "<br>");
                    out.println("Nome: " + p.getNome() + "<br>");
                    out.println("E-mail: " + p.getEmail() + "<br>");
                } else {
                    out.println("Não encontrado.<br>");

                }
            }
            out.println("<hr>");
            //mostrar todos os registros
            out.println("Quantidade de registros: " + String.valueOf(pessoaFacade.count()) +
"<br>");
            Collection<Pessoa> cp = pessoaFacade.findAll();
```

```
            for (Pessoa pp : cp) {
                out.println(pp.getCpf() + " - " + pp.getNome() + " - " + pp.getEmail() +
"<br>");
            }
            out.println("</body>");
            out.println("</html>");
        }
    }
```

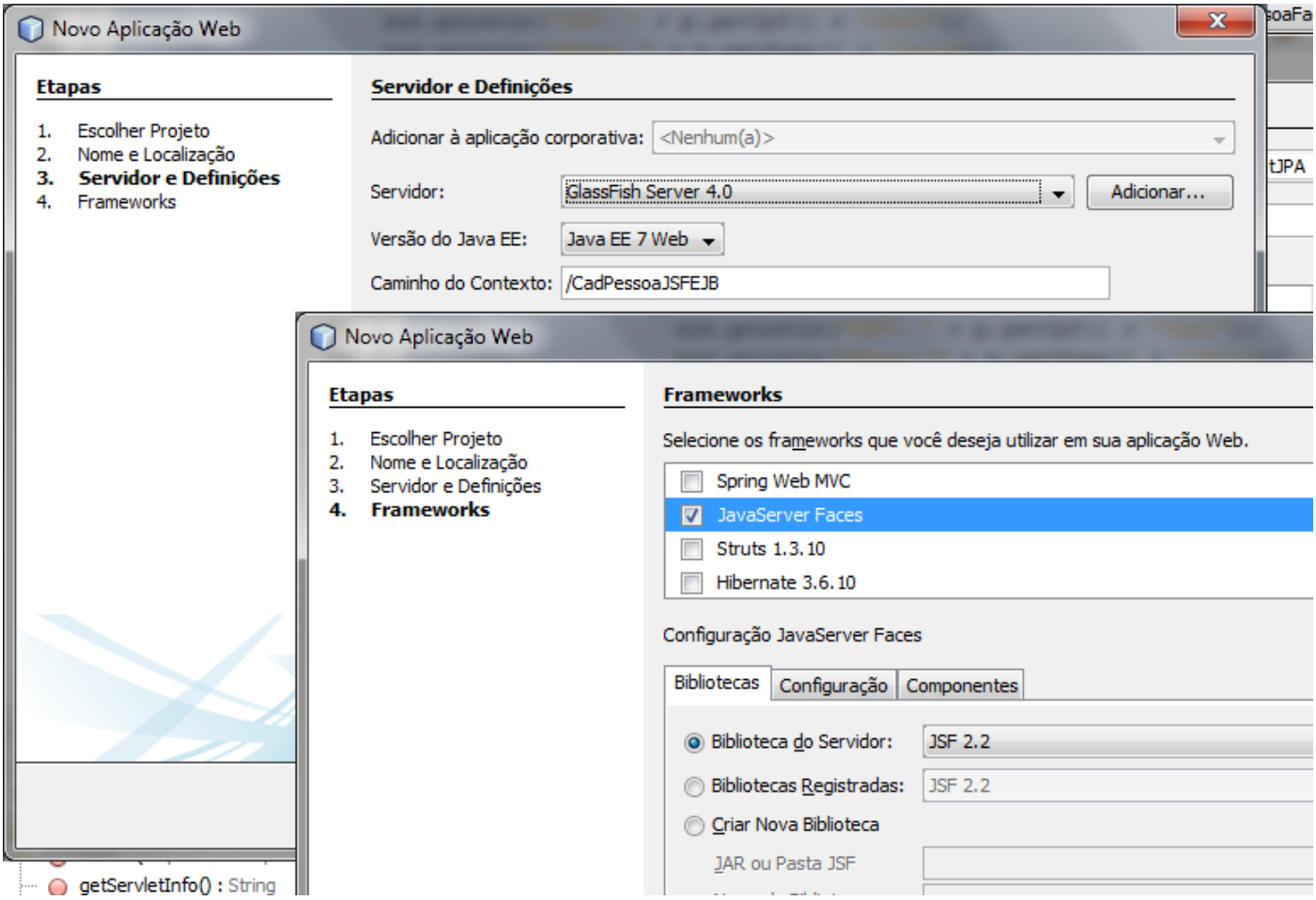
Para acessar o servlet, deve-se enviar o parametro id, a partir de outra pagina com um form ou diretamente na URL:
http://localhost:8080/CadPessoaServletJPA/MostraPessoa?id=123

5. Criando uma Aplicação Web com Java Server Faces que acessa o EJB

Podemos criar uma aplicação Web com JSF que faz acesso ao nosso Java Bean, com a lógica do negócio.

Para isso, crie um Novo Projeto => Java Web => Aplicação Web => Nome do Projeto: CadPessoaJSFEJB => Próximo => Servidor: GlassFish => Próximo

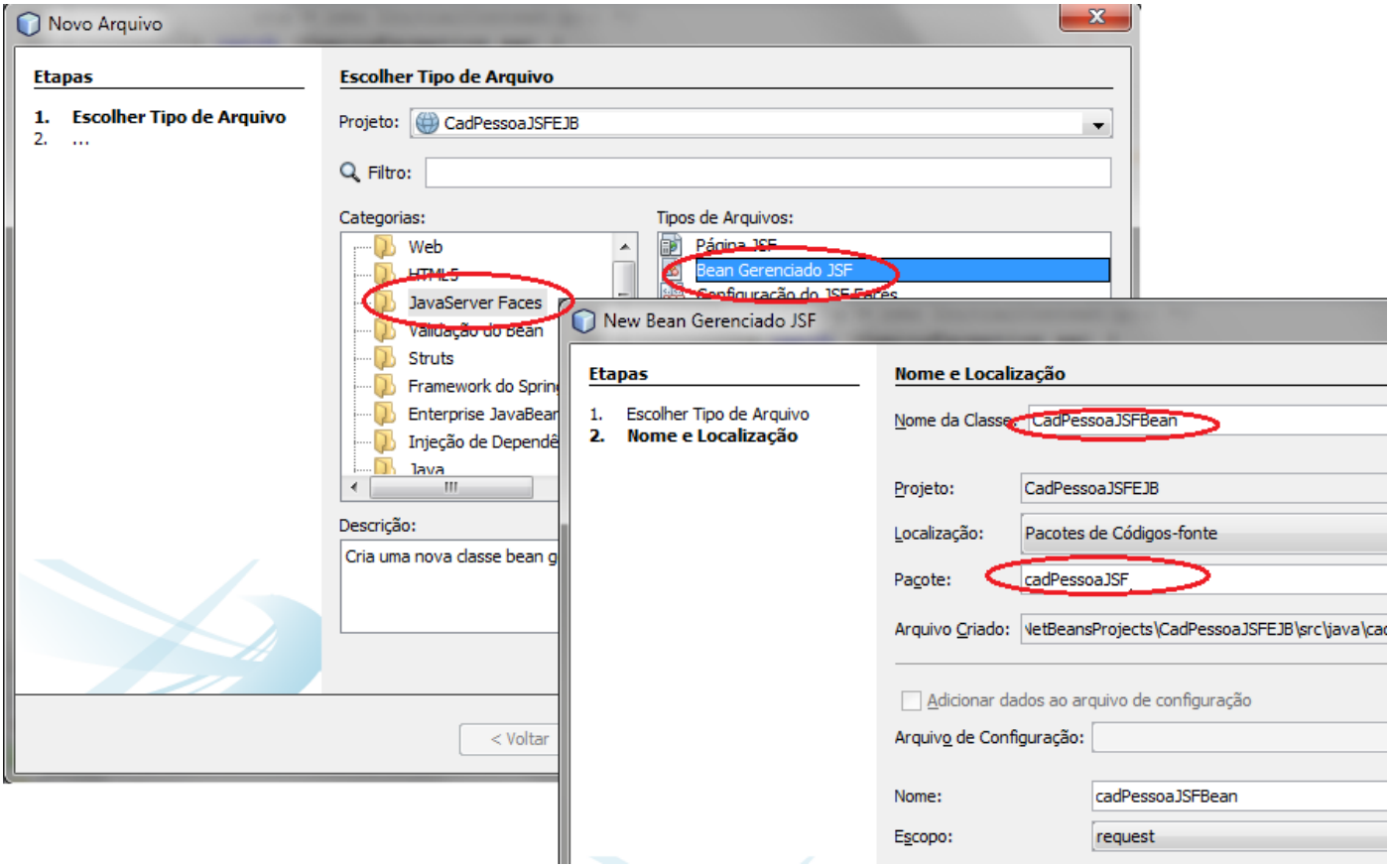
Marque o Framework: Java Server Faces e clique em Finalizar



Para que as páginas web JSF funcionem, é necessário criar um Java Bean que fará o gerenciamento dos dados com a JSF.

Clique sobre o projeto CadPessoaJSFEJB e escolha Novo => Outros => Java Server Faces => Bean Gerenciado JSF => Próximo

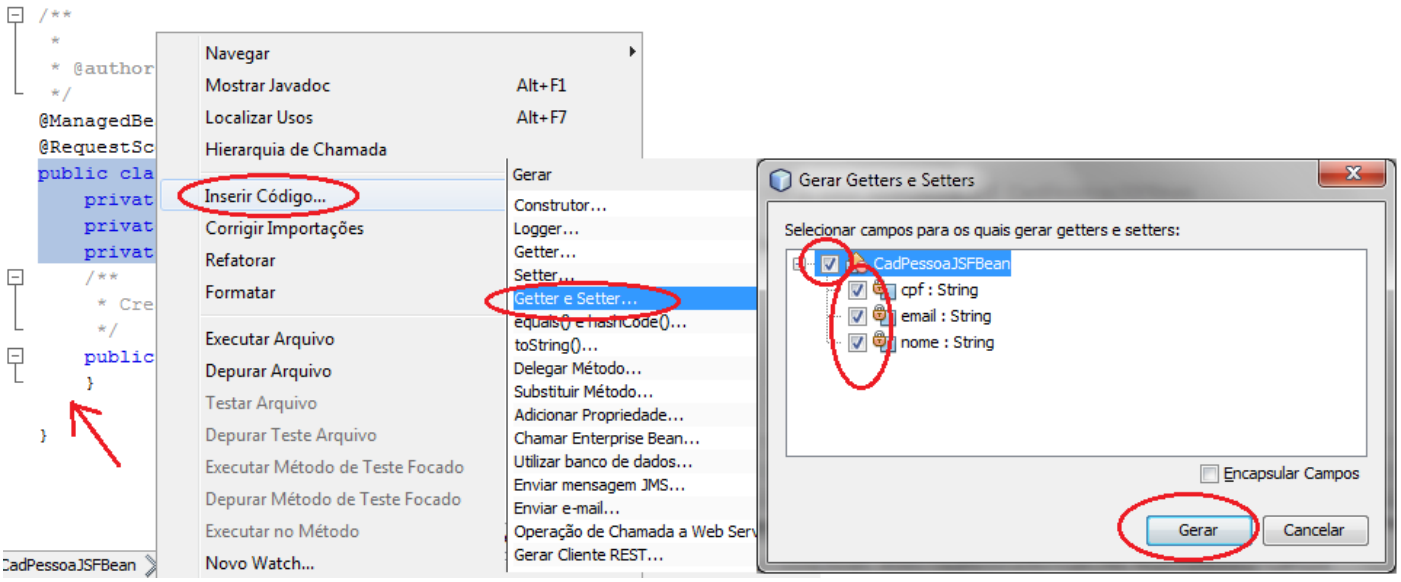
Dê o nome de CadPessoaJSFBean e coloque no pacote cadPessoaJSF e clique em Finalizar



Declare 3 propriedades (variáveis) logo no início da classe:

```
public class CadPessoaJSFBean {
    private String cpf;
    private String nome;
    private String email;
```

Clique no final do código, antes da última chave com o botão direito e escolha: Inserir Código => Getter e Setter para as propriedades criadas anteriormente:



Adicione a biblioteca LibCadPessoa.jar

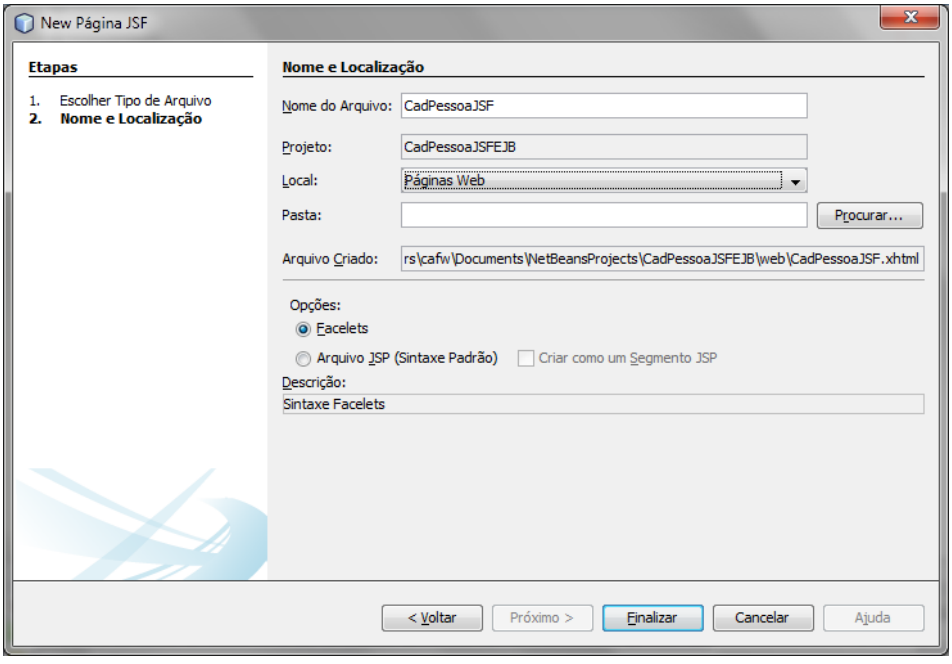
Crie o seguinte método ao na classe CadPessoaJSFBean, antes da última chave, que fará o acesso ao nosso EJB e sincronizará as propriedades cpf, nome e email com a página JSF:

```
public void cadastrar() {
        InitialContext ctx = null;
        CadPessoaBeanRemote cp;
        cp = null;
        try {
            ctx = new InitialContext();// ser for aplicação local ou
            /* o codigo a seguir se for aplicação remota
             Properties p = new Properties();
             p.put("org.omg.CORBA.ORBInitialHost", "192.168.1.175");
             p.put("org.omg.CORBA.ORBInitialPort", "3700");
             ctx = new InitialContext(p); */
        } catch (NamingException ex) {
            System.out.println("Erro: " + ex);
        }
        try {
            cp = (CadPessoaBeanRemote) ctx.lookup("java:global/EJBCadPessoa/CadPessoaBean");
        } catch (NamingException ex) {
            System.out.println("Erro: " + ex);
        }
        if (cp != null && cpf.length()>1 && cp.CadastrarPessoa(cpf, nome, email)) {
            nome = "";
            cpf = "";
            email = "";
            FacesMessage msg = new FacesMessage("Cadastro ok...");
            FacesContext.getCurrentInstance().addMessage(null, msg);
        } else {
           FacesMessage msg = new FacesMessage("Erro no Cadastro.");
            FacesContext.getCurrentInstance().addMessage(null, msg);
        }
    }
```

Agora só precisamos de uma página JSF que acessará o CadPessoaJSFBean, que por sua vez acessa o nosso EJB com a lógica do negócio e a base de dados.

Clique sobre o projeto CadPessoaJSFEJB e escolha Novo => Outros => Java Server Faces => Página JSF => Próximo

Dê o nome de CadPessoaJSF e deixe os demais valores como estão e clique em Finalizar



Altere o código da página CadPessoaJSF.xhtml para:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Cadastro de Pessoa</title>
    </h:head>
    <h:body>
        Cadastro de Pessoa
        <h:form>
            <h:panelGrid columns="2">
                CPF: <h:inputText value="#{cadPessoaJSFBean.cpf}"/>
                Nome: <h:inputText value="#{cadPessoaJSFBean.nome}"/>
                E-Mail: <h:inputText value="#{cadPessoaJSFBean.email}"/>
                <h:commandButton value="Cadastrar" action="#{cadPessoaJSFBean.cadastrar}"/>
                <h:messages globalOnly="true"/>
            </h:panelGrid>
        </h:form>
    </h:body>
</html>
```

Salve, implante e teste a página => (Executar Arquivo ou acessar:
http://localhost:8080/CadPessoaJSFEJB/faces/CadPessoaJSF.xhtml

6. Criando uma Aplicação Web com Java Server Faces e JPA

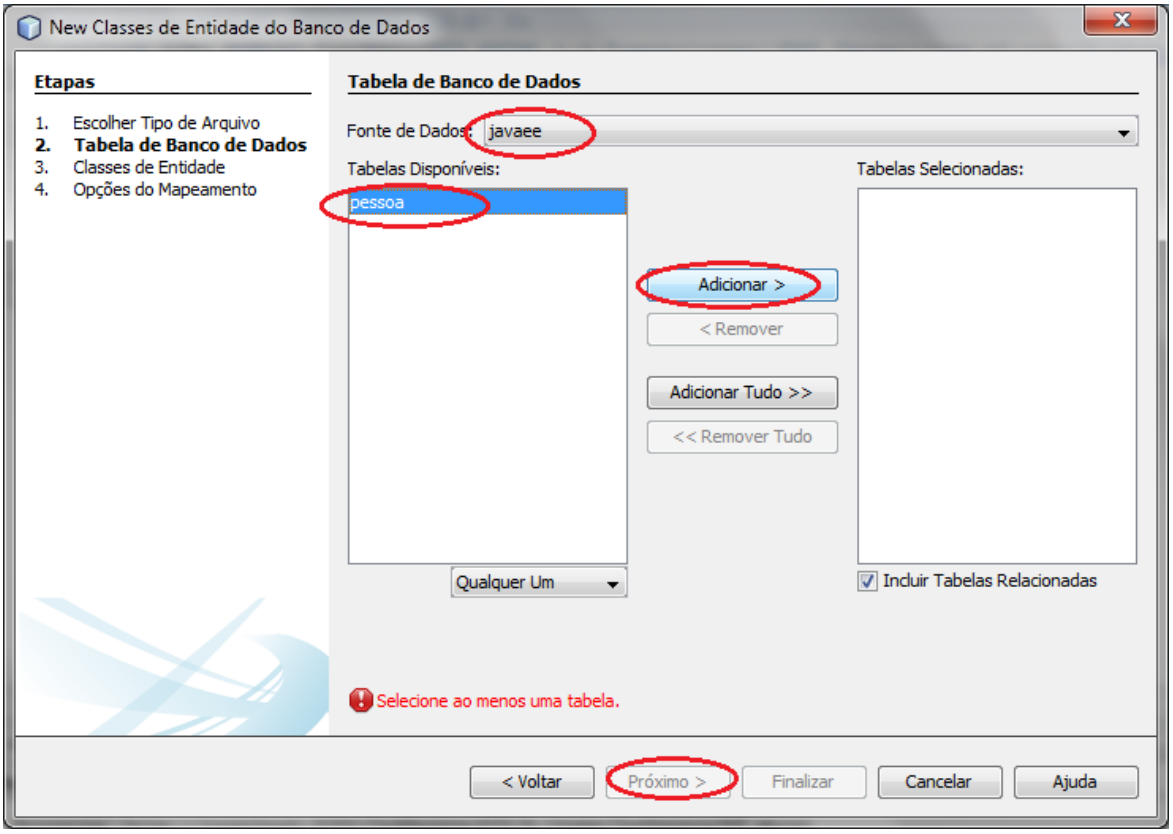
Podemos criar uma aplicação Web com JSF que acesso os dados através da camada de persistência JPA.

Para isso, crie um Novo Projeto => Java Web => Aplicação Web => Nome do Projeto: CadPessoaJSFJPA => Próximo => Servidor: GlassFish => Próximo

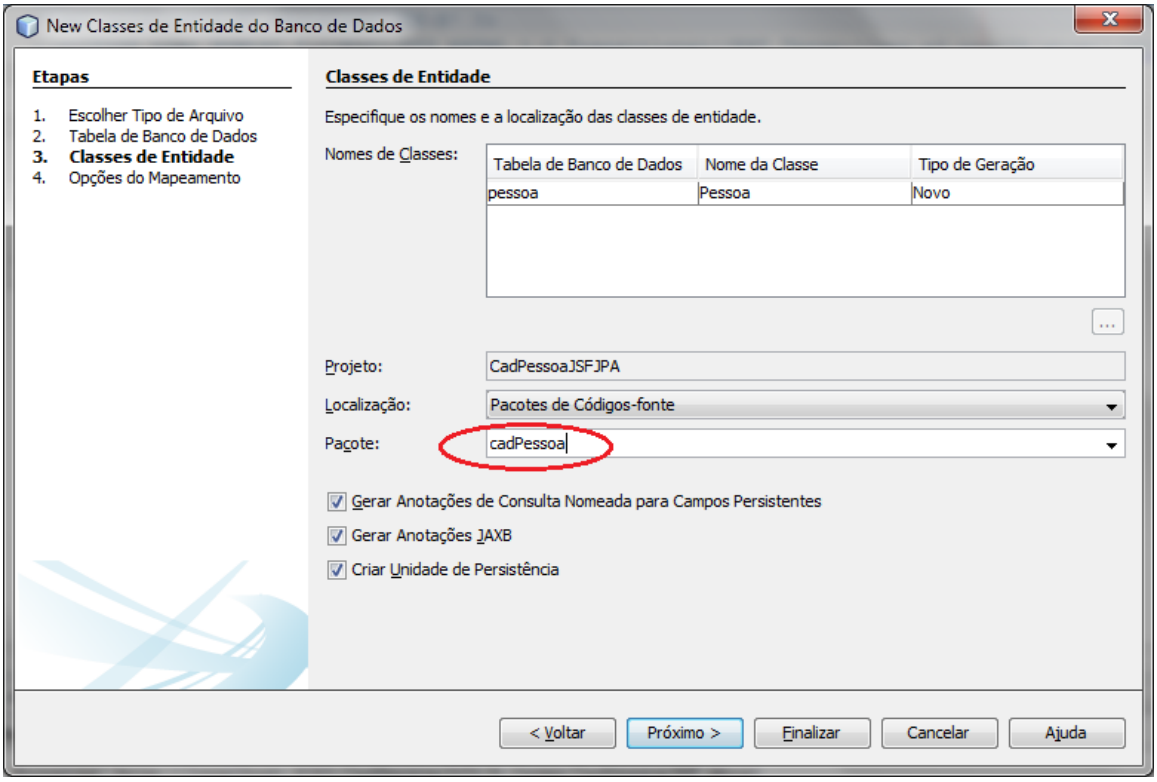
Marque o Framework: Java Server Faces e clique em Finalizar

Sobre o projeto, crie um Novo => Outros => Persistência => Classes de Entidade de Banco de Dados => Próximo

=> Fonte deDados: javaee => Marque a tabela pessoa => Adicionar => Próximo



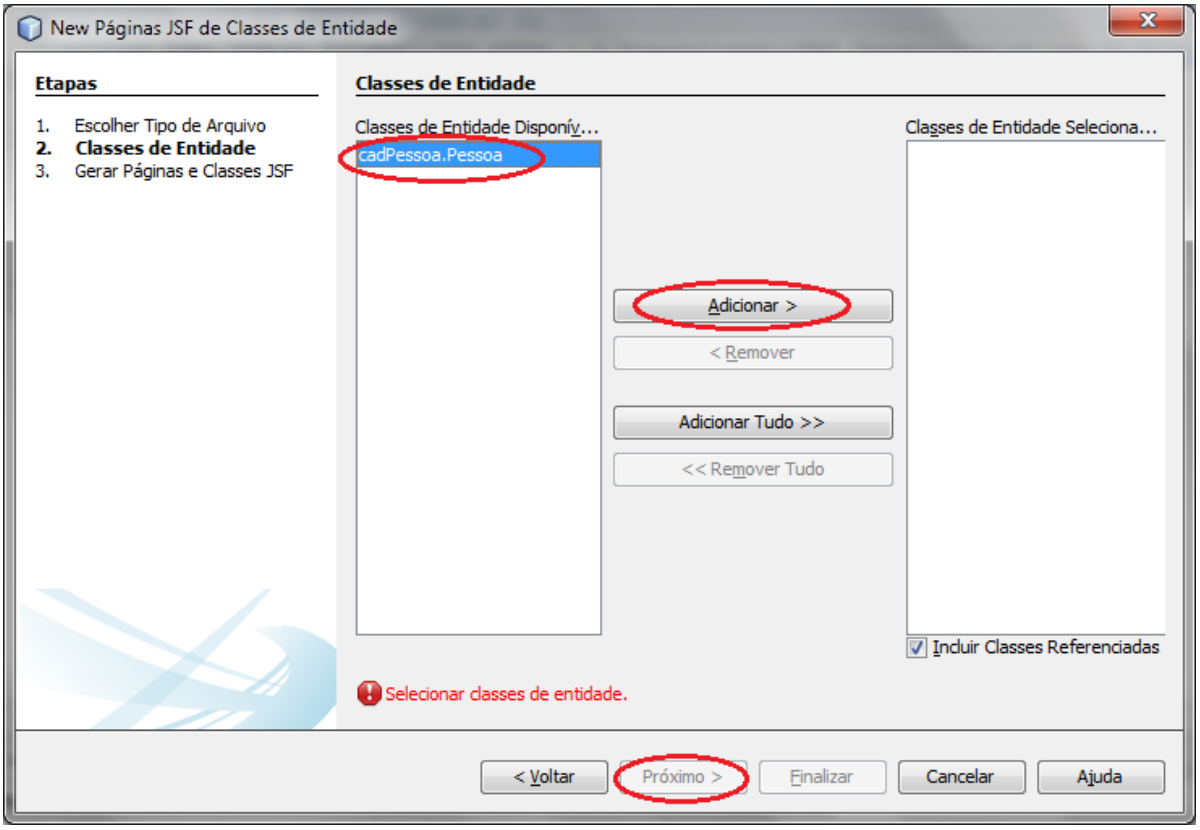
Coloque no pacote cadPessoa



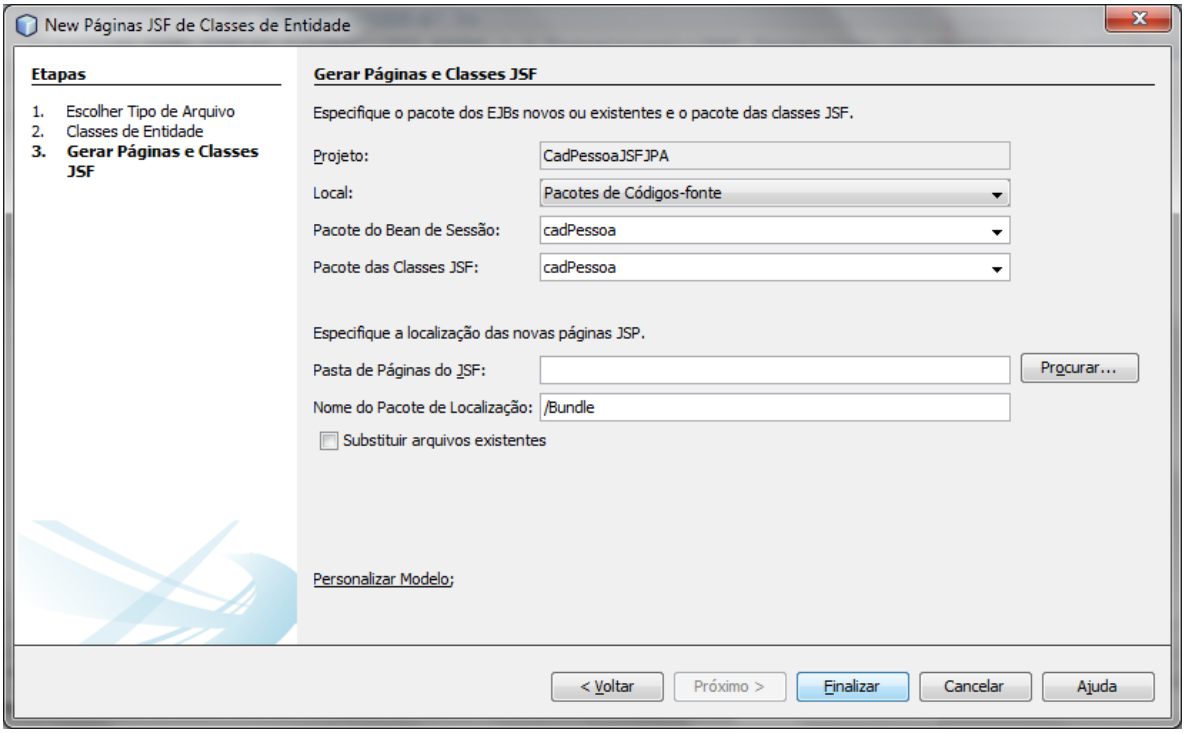
=> Próximo => Finalizar

Sobre o projeto, crie um Novo => Outros => JavaServer Faces => Páginas JSF de Classes de Entidades

Selecione cadPessoa.pessoa => Adicionar => Próximo



Mantenha os valores padrão e clique em Finalizar



Pronto!!!!

Clique sobre o projeto e escolha Implantar

Clique sobre o projeto e escolha Executar

Se não gostou dos textos em inglês, altere o arquivo Bundle.properties (pacotes de Código-fonte => <pacote default> traduzindo o texto exibido. Salve e implante novamente.