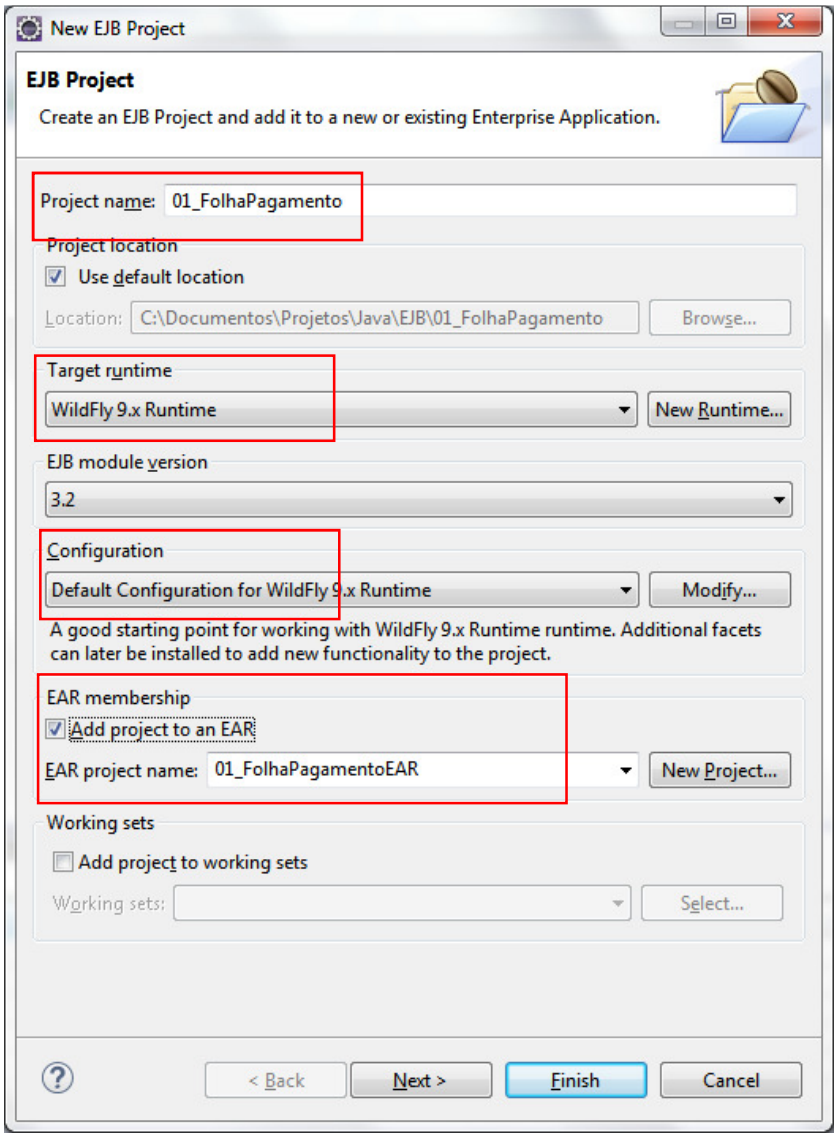
Introdução aos Enterprise JavaBeans

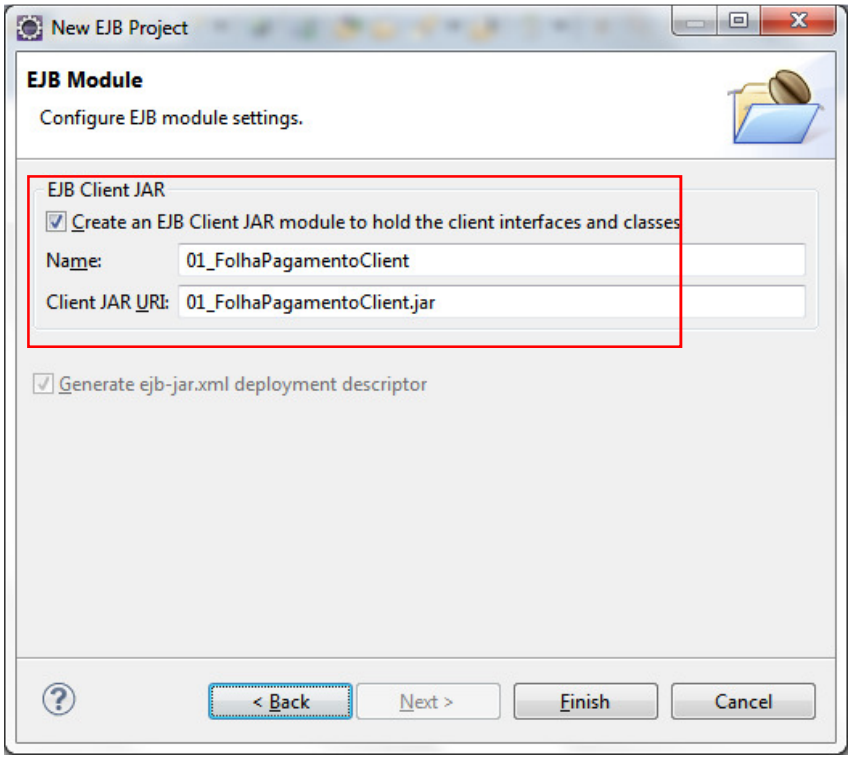
Neste roteiro desenvolveremos os primeiros exemplos com EJB 3.

Exemplo 01: Componentes de servidor e cliente no mesmo projeto.

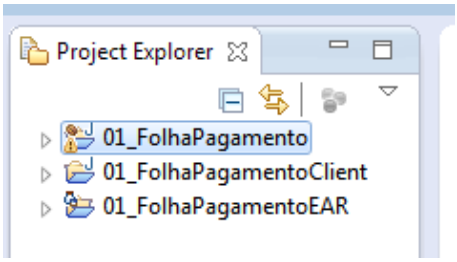
1. No Eclipse, criar um projeto do tipo **EJB Project**. Nomeá-lo como **01_FolhaPagamento**. Incluir o servidor **Wildfly-[versão]**, e manter as configurações mostradas abaixo:



2. Avance duas vezes, e mantenha a seleção a seguir:



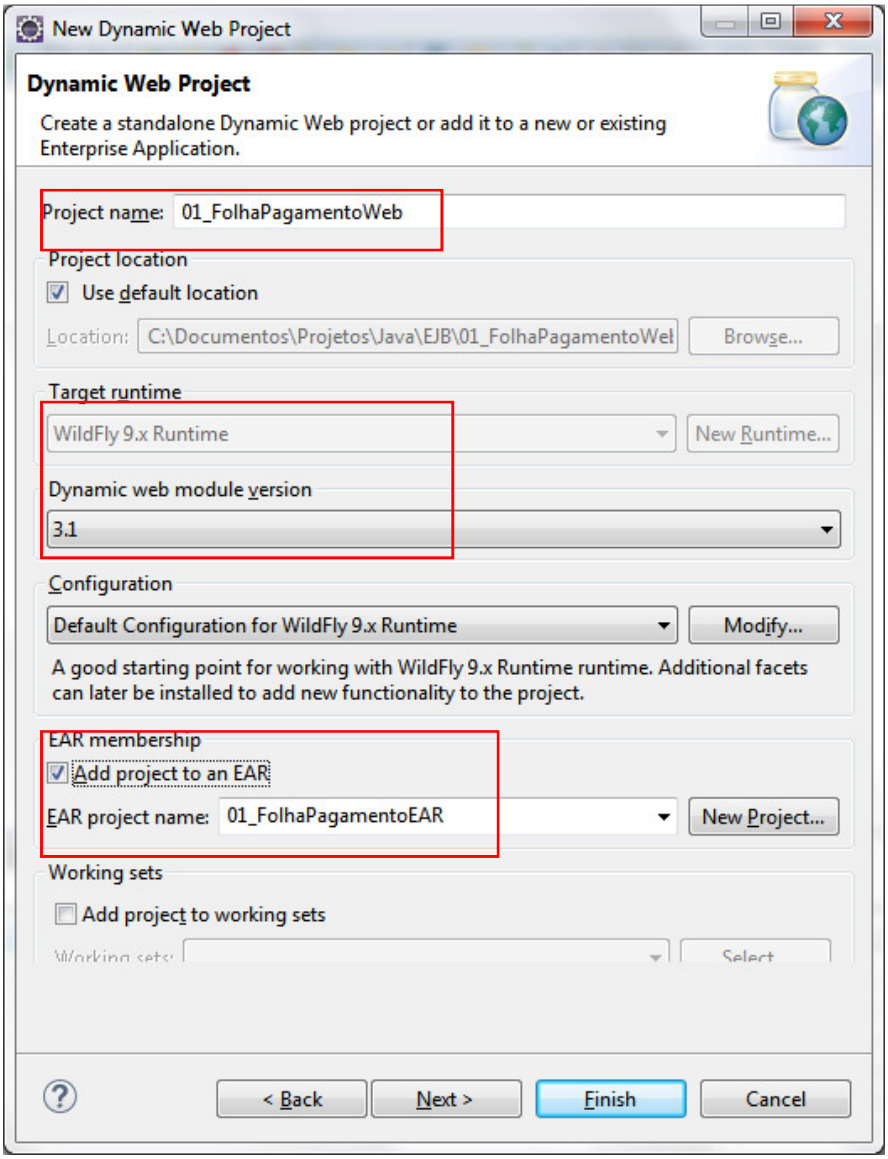
3. Ao finalizar, devemos ter os três projetos listados abaixo:



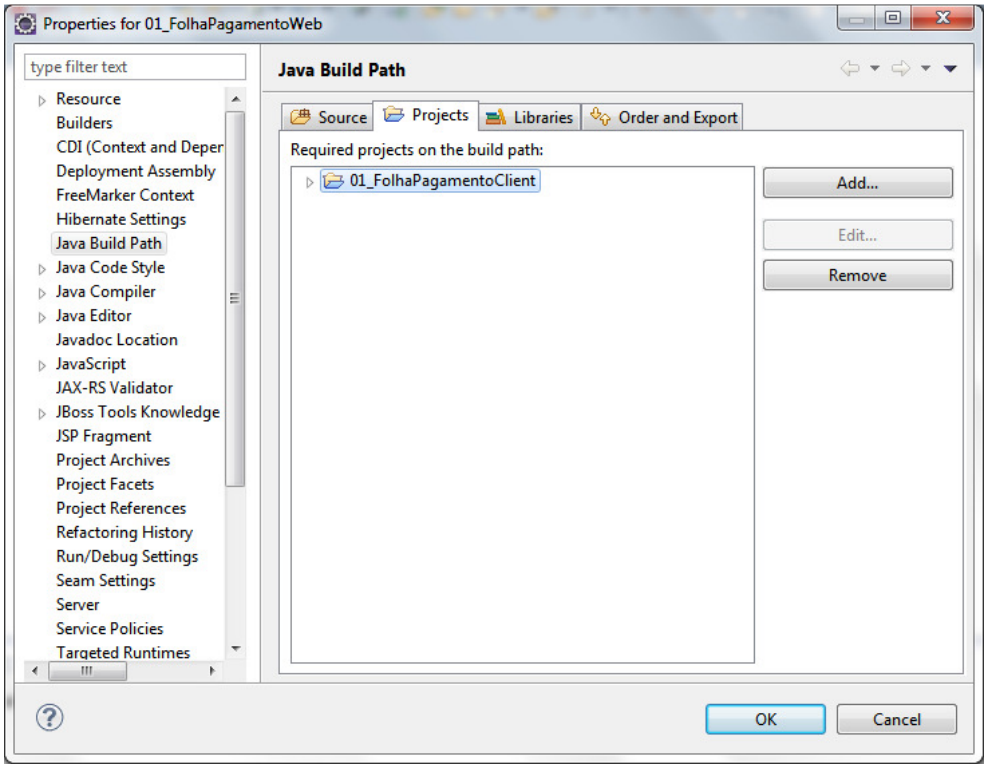
Os projetos possuirão os seguintes elementos:

- **01_FolhaPagamento** – Classes que implementam os *Stateless Session Beans*.
- **01_FolhaPagamentoClient** – Interfaces que definem as operações dos *Stateless Session Beans*.
- **01_FolhaPagamentoEAR** – Responsável por empacotar todos os módulos da aplicação.

4. Criar um novo projeto WEB (Dynamic Web Project) para consumir o EJB. Nomeie-o como **01_FolhaPagamentoWeb**, mantendo as configurações mostradas a seguir:



5. Adicionar o projeto **01_FolhaPagamentoClient** como dependência do novo projeto web (Configuração do Build Path):



6. No projeto **01_FolhaPagamentoClient** criar uma interface chamada **FolhaPagamento**, cujo modelo é ilustrado a seguir:

```
package br.com.fiap.interfaces;

public interface FolhaPagamento {
 void setSalario(double salario);
 void setDesconto(double taxa);
 double calcularINSS();
 double calcularSalarioLiquido();
}
```

7. No projeto **01_FolhaPagamento** criar a classe **FolhaPagamentoBean**, conforme ilustrado a seguir:

```
package br.com.fiap.bean;

import javax.ejb.Local;
import javax.ejb.Stateful;

import br.com.fiap.interfaces.FolhaPagamento;

@Stateful
@Local(FolhaPagamento.class)
public class FolhaPagamentoBean implements FolhaPagamento{

 private double salario, taxa;
 @Override
 public void setSalario(double salario) {
   this.salario = salario;
```

```
    }

    @Override
    public void setDesconto(double taxa) {
       this.taxa = taxa;
    }

    @Override
    public double calcularINSS() {
       return salario * taxa / 100;
    }

    @Override
    public double calcularSalarioLiquido() {
       return salario - calcularINSS();
    }
  }
```

8. No projeto **01_FolhaPagamentoWeb**, criar um Servlet e uma página JSP, de acordo com os modelos dados a seguir:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Folha Pagamento</title>
</head>
<body>
   <form action="folha" method="post">
      Informe o valor do salário:<br/>
      <input type="text" name="salario" size="10" /><br/>
      <input type="submit" value="Enviar" />
   </form>
</body>
</html>
```

```
package br.com.fiap.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import br.com.fiap.interfaces.FolhaPagamento;

@WebServlet("/folha")
```

```
public class ServletFolha extends HttpServlet {
   private static final long serialVersionUID = 1L;

   @EJB
   FolhaPagamento fp;
    public ServletFolha() {
        super();
        // TODO Auto-generated constructor stub
    }

   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
   }

   protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
      PrintWriter out = response.getWriter();
      response.setContentType("text/html");

      try {
         double salario = Double.parseDouble(request.getParameter("salario"));
         fp.setSalario(salario);
         fp.setDesconto(10);
         out.println("Salário Bruto: " + salario);
         out.println("<br/>Saláro Líquido: " + fp.calcularSalarioLiquido());
      } catch (Exception e) {
         out.print(e.getMessage());
      }
   }
}
```

9. Para executar, selecione a opção Run As... -> Run On Server e certifique-se que o arquivo EAR esteja na lista de execução.

10. Executa a página JSP, e constatar o resultado.

Exemplo 02: Calculadora usando EJB e JSF

1. Criar um novo projeto **EJB Project** chamado **02_Calculadora**. Marcar a opção "Add Project to na EAR" chamado **02_CalculadoraEAR** (se não existir, cria-lo).

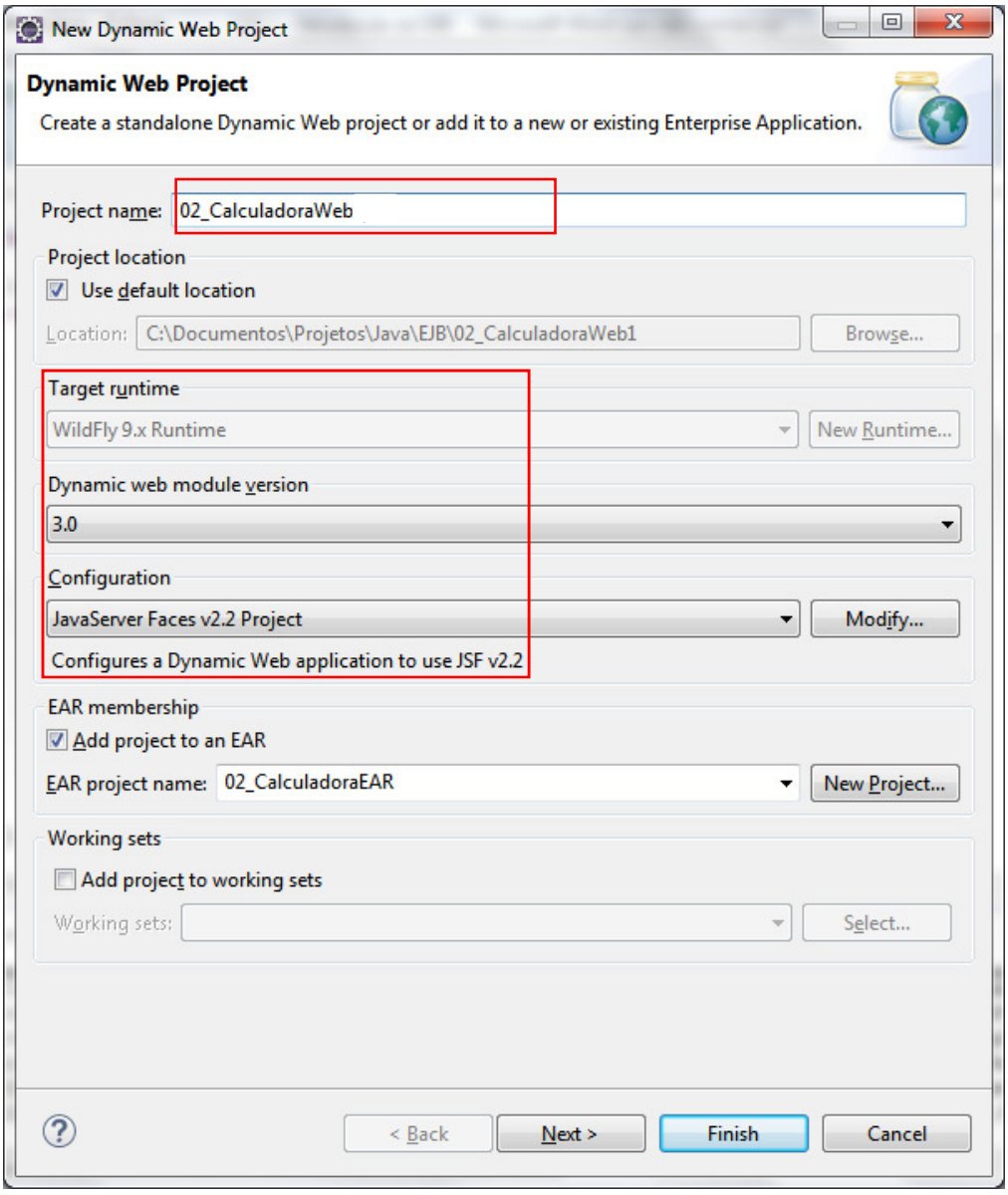
2. Criar uma interface chamada **CalculadoraLocal**, no projeto 02_CalculadoraClient, com as definições das quatro operações de uma calculadora. Marcar esta interface com a anotação **@Local**:

```
package br.com.fiap.interfaces;

import javax.ejb.Local;

@Local
public interface CalculadoraLocal {
 double somar(double x, double y);
 double subtrair(double x, double y);
 double multiplicar(double x, double y);
 double dividir(double x, double y);
}
```

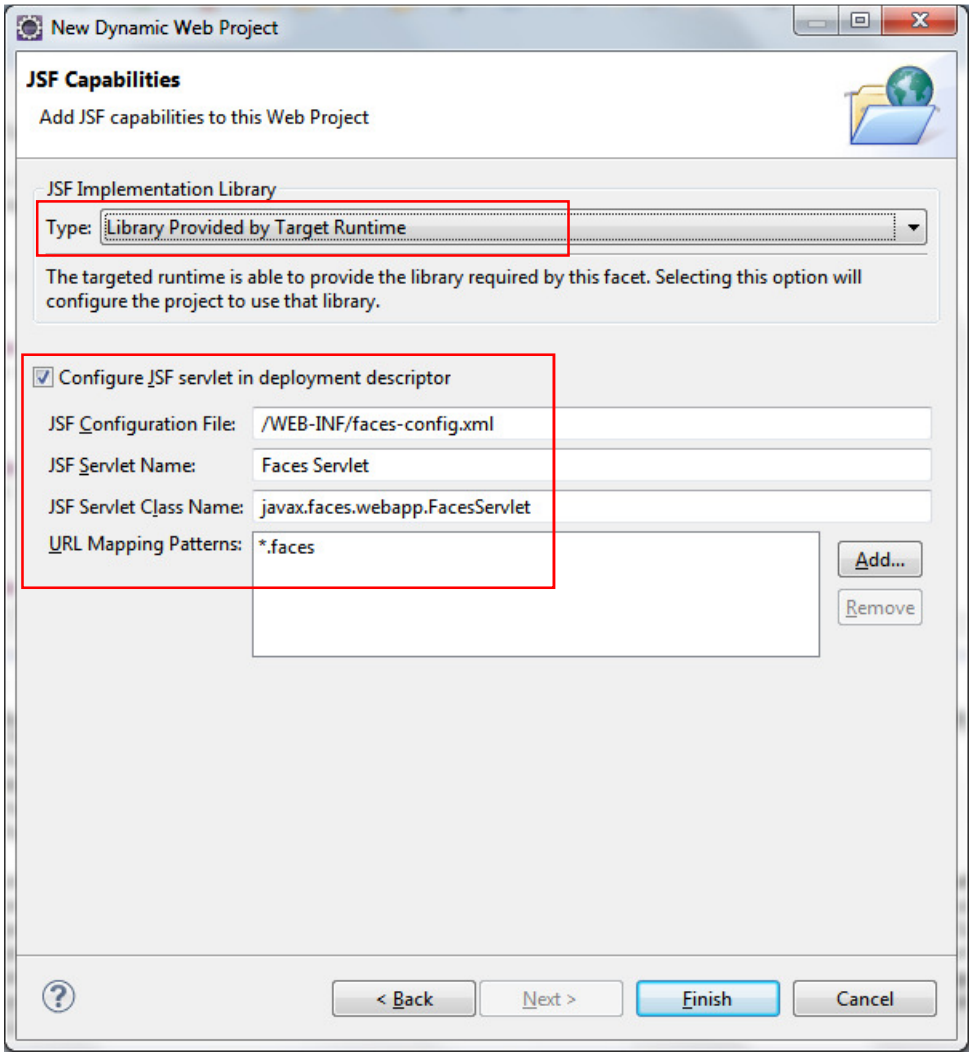
3. No projeto **02_Calculadora**, definir a classe que representará o componente EJB. A classe deverá se chamar **CalculadoraBean**, e deverá implementar a interface do item 2:

```
package br.com.fiap.bean;

import javax.ejb.Stateless;

import br.com.fiap.interfaces.CalculadoraLocal;

@Stateless
public class CalculadoraBean implements CalculadoraLocal{

 @Override
 public double somar(double x, double y) {
    return x + y;
 }

 @Override
 public double subtrair(double x, double y) {
    return x - y;
 }

 @Override
 public double multiplicar(double x, double y) {
    return x * y;
 }

 @Override
 public double dividir(double x, double y) {
    return x / y;
 }
}
```

4. Criar um novo projeto web (Dynamic Web Project) baseado em JSF, chamado **02_CalculadoraWeb**. Neste projeto, selecionar a opção *JavaServer Faces 2.2* como Configuration, incluindo-o ao EAR:



5. Avançar, selecionar a opção que permite criar o arquivo web.xml, e manter a configuração abaixo:



6. Neste projeto, adicionar uma referencia ao projeto 02_CalculadoraClient.

7. No projeto WEB criar um *Managed Bean* para conter os resultados:

```
package br.com.fiap.mb;

import javax.ejb.EJB;
import javax.faces.bean.ManagedBean;

import br.com.fiap.interfaces.CalculadoraLocal;

@ManagedBean(name="calcMB")
public class CalculadoraManagedBean {


 @EJB
 private CalculadoraLocal calc;

 private double x, y;

 private String resultado;
```

```
  public double getX() {
     return x;
  }

  public void setX(double x) {
     this.x = x;
  }

  public double getY() {
     return y;
  }

  public void setY(double y) {
     this.y = y;
  }

  public String getResultado() {
     return resultado;
  }

  public void setResultado(String resultado) {
     this.resultado = resultado;
  }

  public void somar(){
     double result = calc.somar(x, y);
     setResultado("Soma = " + result);
  }
  public void subtrair(){
     double result = calc.subtrair(x, y);
     setResultado("Subtração = " + result);
  }
  public void multiplicar(){
     double result = calc.multiplicar(x, y);
     setResultado("Multiplicação = " + result);
  }
  public void dividir(){
     double result = calc.dividir(x, y);
     setResultado("Divisão = " + result);
  }
}
```

8. Definir, no projeto Web, um arquivo xhtml para testar a calculadora remota!!!

```
<!DOCTYPE html>
<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>Calculadora</title>
</h:head>
<h:body>
 <f:view>
    <h:form>
```

```
            <h:outputText value="Valor de X:" />
            <h:inputText value="#{calcMB.x}" />
            <br/>
            <h:outputText value="Valor de Y:" />
            <h:inputText value="#{calcMB.y}" />
            <br/>
            <h:commandButton value="Soma" action="#{calcMB.somar}" />
            <h:commandButton value="Subtração" action="#{calcMB.subtrair}" />
            <h:commandButton value="Multiplicação" action="#{calcMB.multiplicar}" />
            <h:commandButton value="Divisão" action="#{calcMB.dividir}" />
            <br/>
            <h:outputText value="#{calcMB.resultado}" />
        </h:form>
    </f:view>
</h:body>
</html>
```

9. Para executar, selecionar Run As -> Run on Server, como no exemplo anterior.

Bom trabalho!

Prof. Emilio