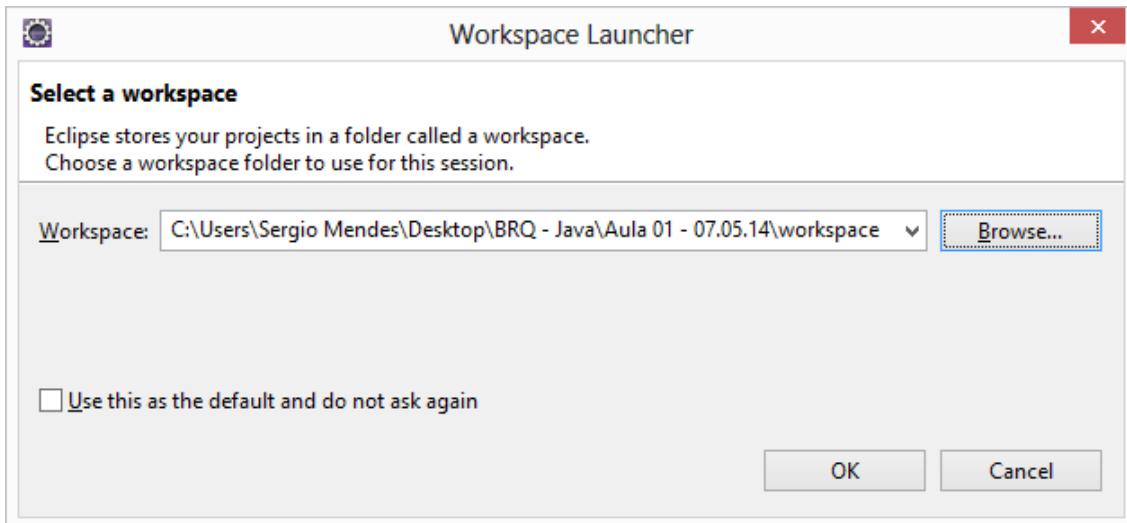


# **Treinamento Java 2014**

---



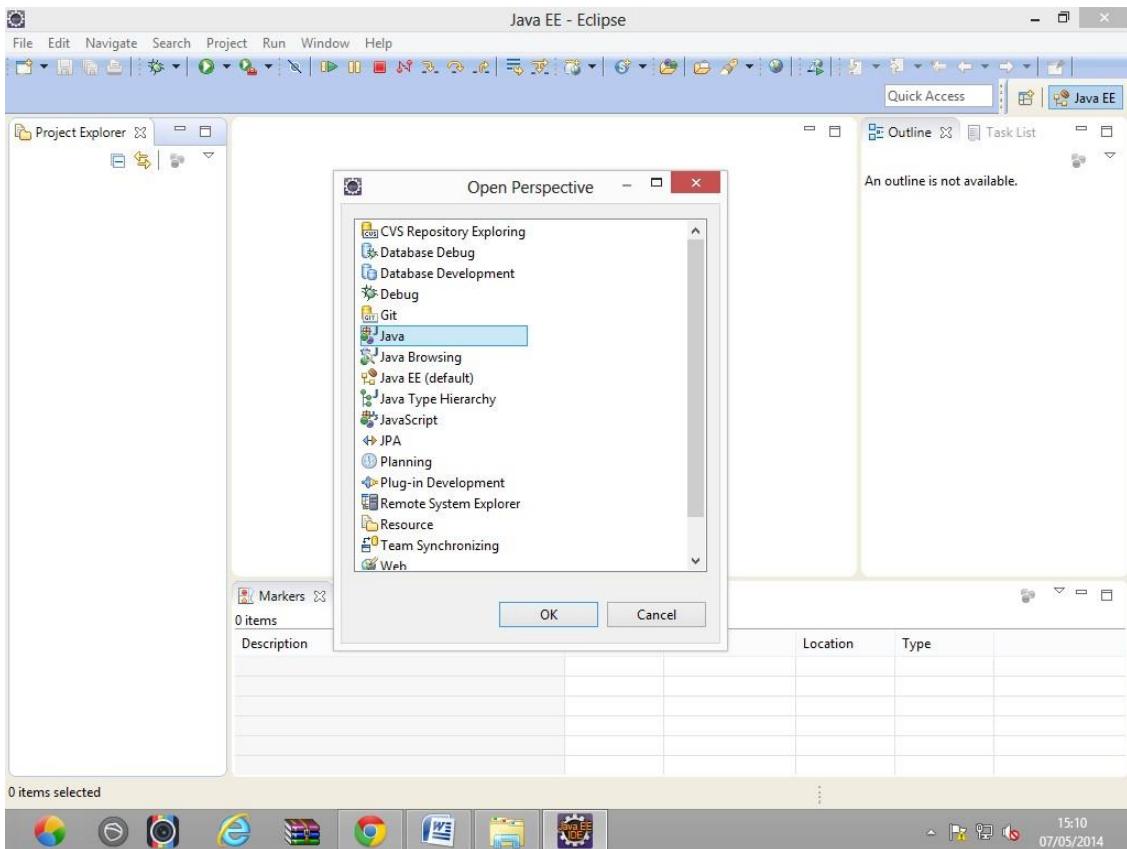
Eclipse: IDE (Ambiente Integrado de Desenvolvimento)  
Workspace (diretório de trabalho)



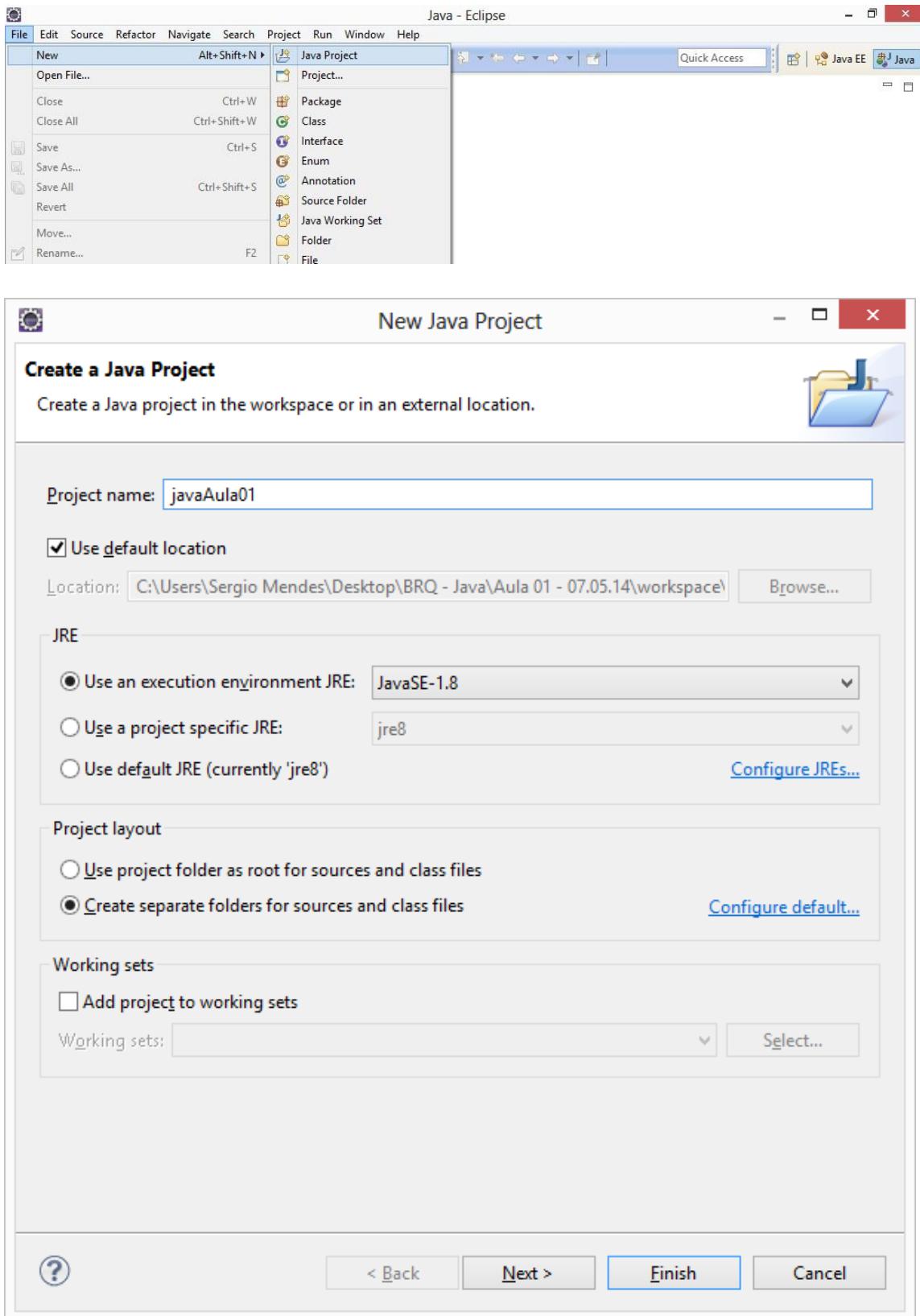
Java SE - Java Standard Edition (Local)

Java EE - Java Enterprise Edition

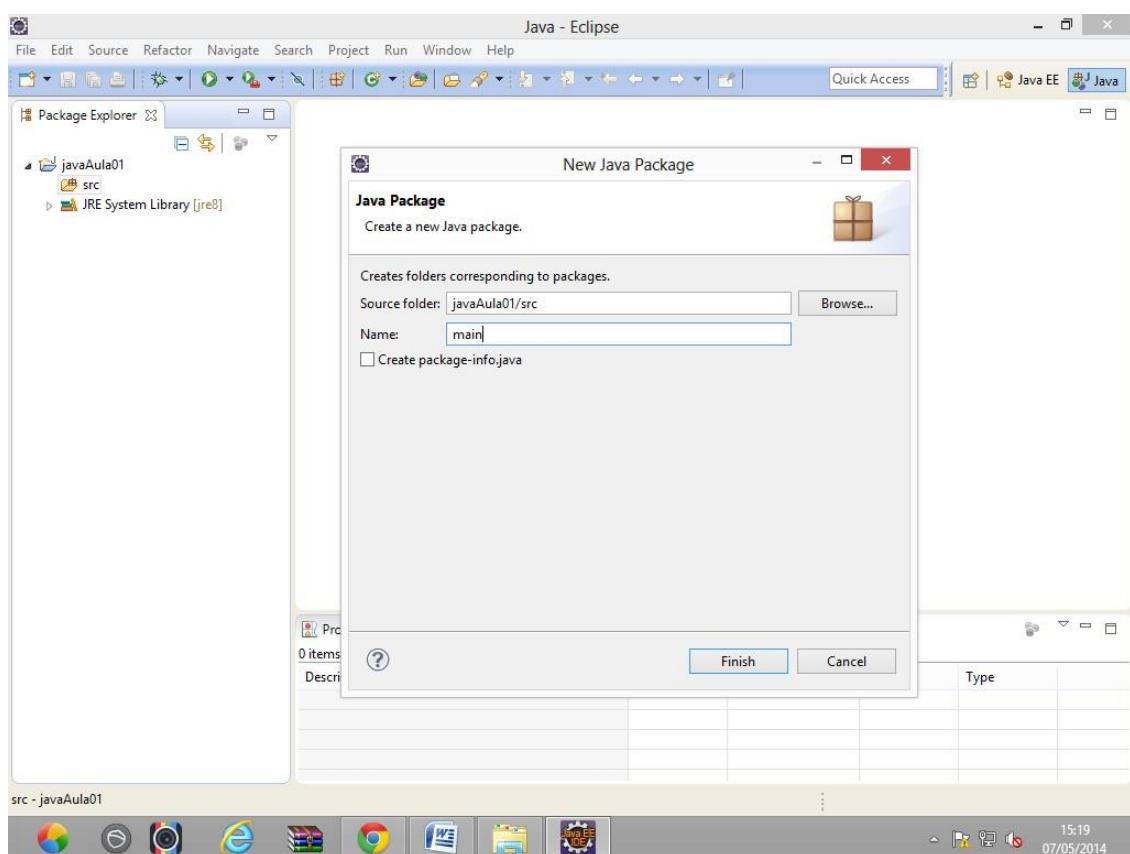
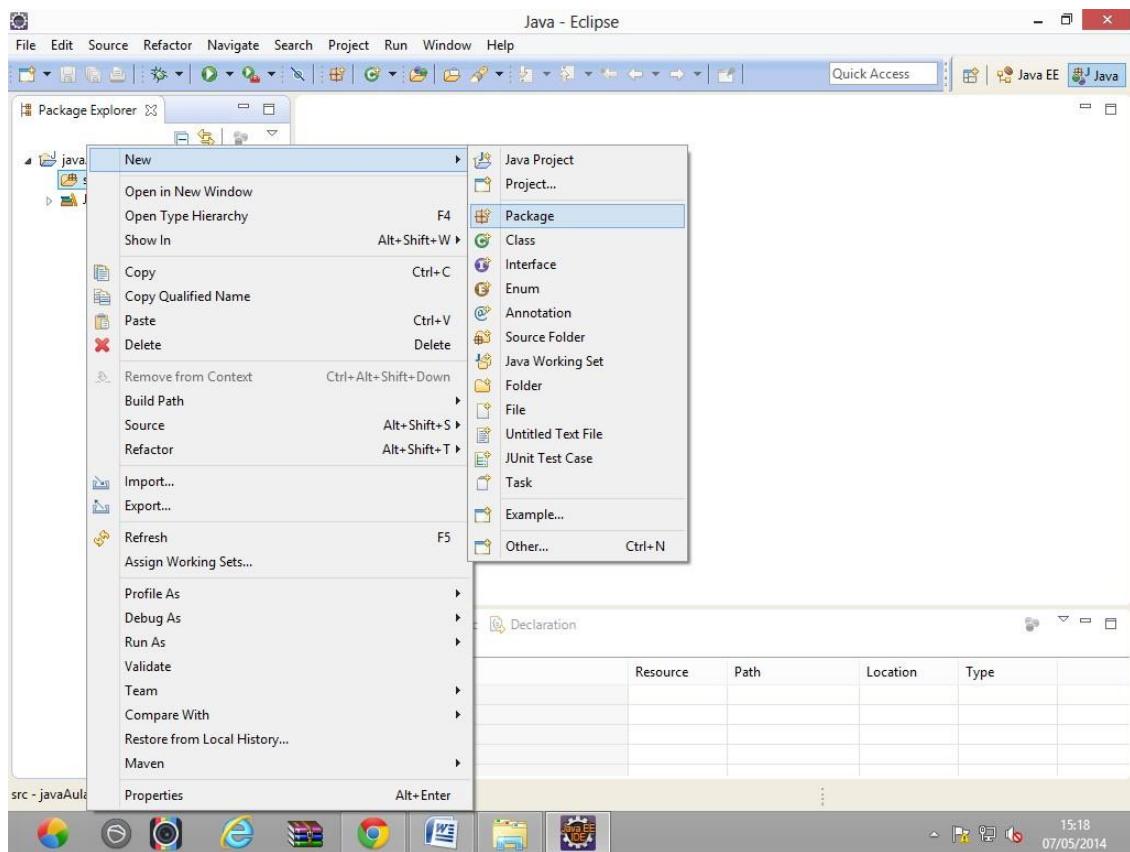
Java ME - Java Micro Edition



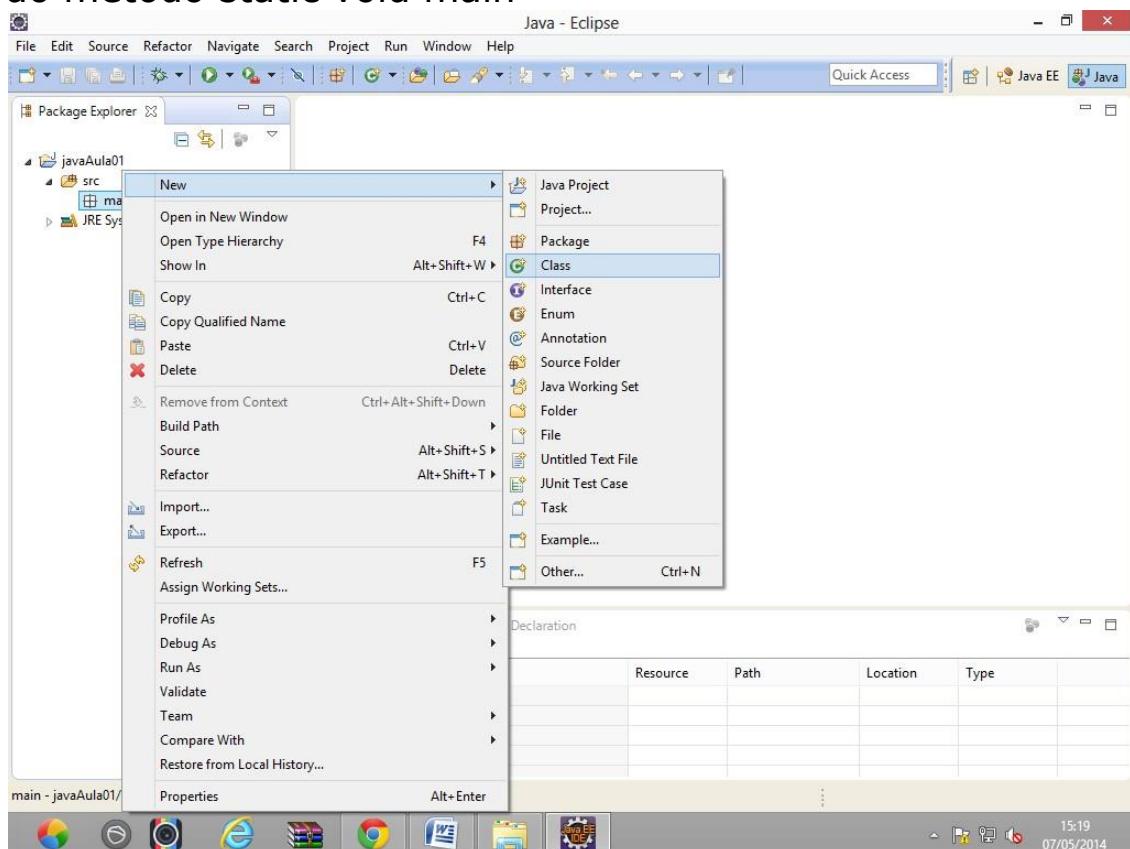
## Criando o Projeto...



# Aula 01



## Criando a Classe para executar o projeto Java SE através do método static void main



```
package main;

//Classe utilizada para executar o projeto Java local
public class Main {

    //Em java, para executar uma Classe, é necessário um
    //método com a assinatura: static void main
    public static void main(String[] args) {

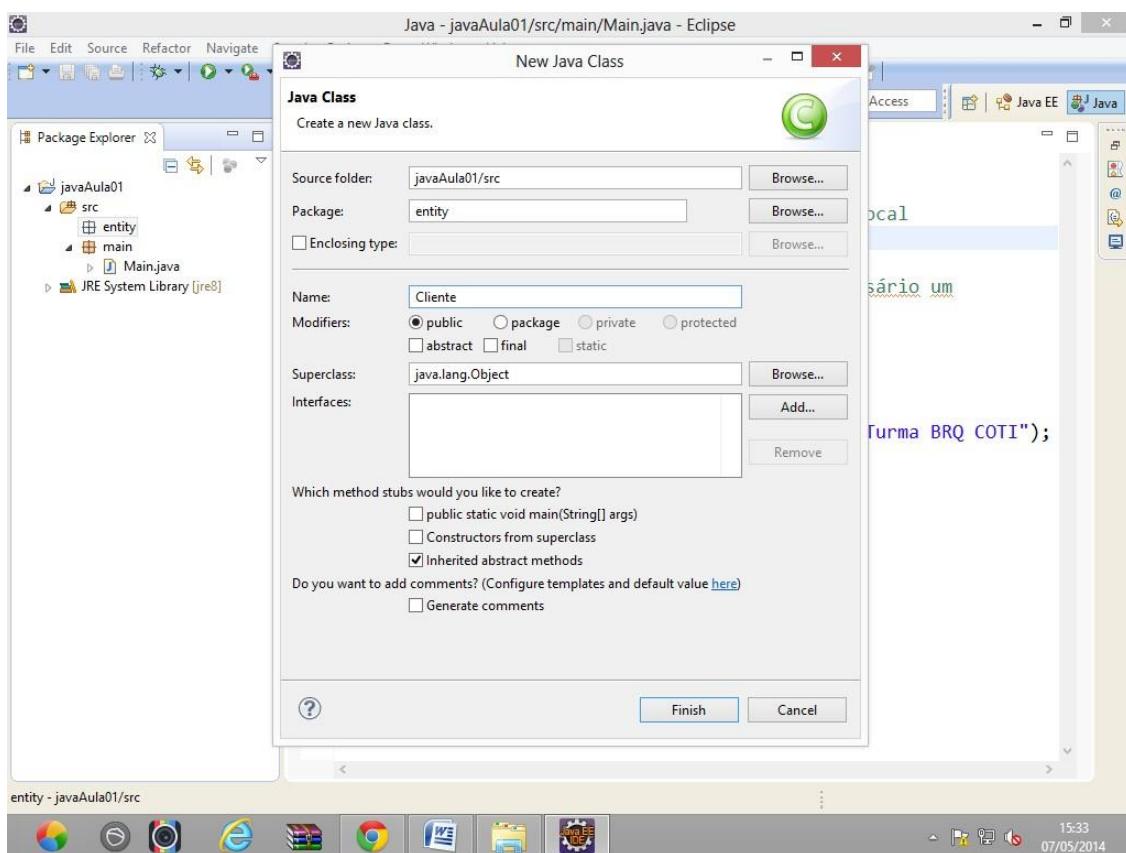
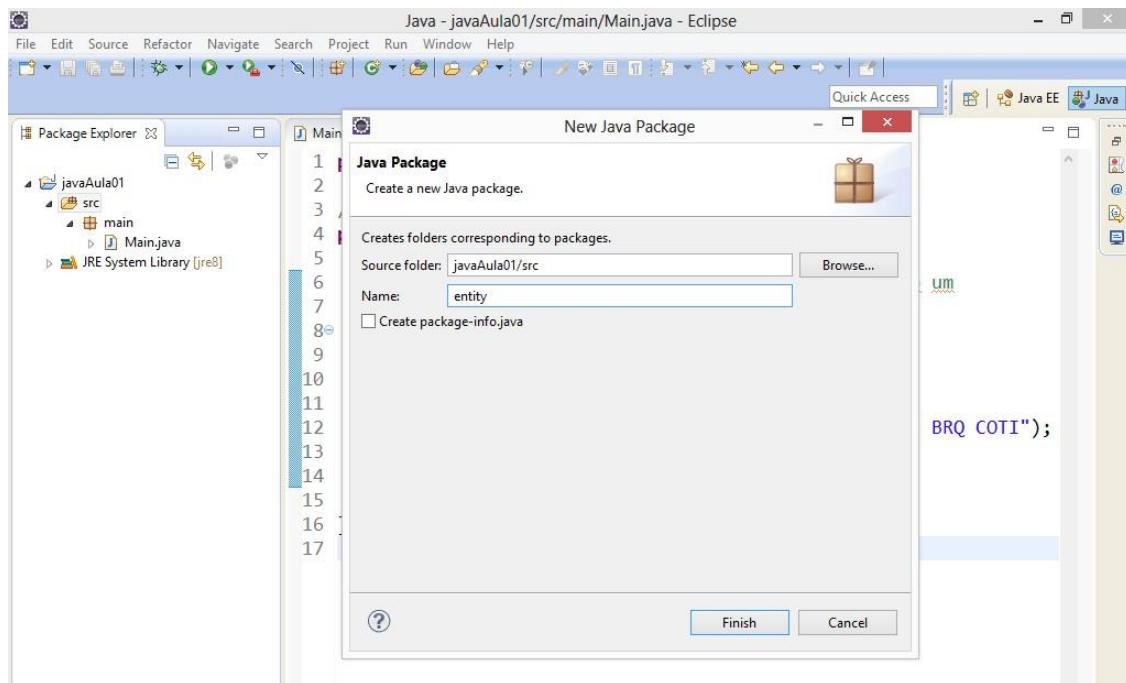
        //imprimir mensagem no console
        //sysout -> ctrl + espaço
        System.out.println("Bem vindo Java - Turma BRQ COTI");

    }
}
```

Para executar:

- Run as -> Java Application
- CTRL + F11

## Modelagem... **Cliente (entidade)** JavaBean



```
//localização do pacote
package entity;

//declaração da Classe
//Nome da Classe Cliente deve ser o nome do arquivo Cliente.java
//Deve ser uma classe de acesso publico
public class Cliente {

    //Atributos -> dados pertencentes a Classe
    //Visibilidade - tipo - nome
    //public -> acesso total ao elemento
    //tipos de dados (primitivos e wrappers (Classes))
    public Integer idCliente;
    public String nome;
    public String email;
}
```

Criando objetos...

**Cliente c1 = new Cliente();**  
[Classe] [Objeto] [Espaço de memória > Inicialização]

```
package main;

import entity.Cliente; //importando a classe Cliente

//Classe utilizada para executar o projeto Java local
public class Main {

    //Em java, para executar uma Classe, é necessário um
    //método com a assinatura: static void main
    public static void main(String[] args) {

        //Classe -> Objeto
        Cliente c1 = new Cliente();

        c1.idCliente = 1;
        c1.nome = "Sergio Mendes";
        c1.email = "sergio.coti@gmail.com";

        System.out.println("Id.....: " + c1.idCliente);
        System.out.println("Nome....: " + c1.nome);
        System.out.println("Email...: " + c1.email);

        //imprimir mensagem no console
        //syso -> ctrl + espaço
        System.out.println("Bem vindo ao Java - Turma BRQ COTI");
    }
}
```

## Testando...

Screenshot of Eclipse IDE showing the execution of a Java application.

**Top Window (Main.java):**

```

Java - javaAula01/src/main/Main.java
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Main.java Cliente.java
javaAula01
src
entity
main
JRE System Library [jre8]
Main.java
import entity.Cliente; //import
//Classe utilizada para executar
public class Main {
    //Em java, para executar
    //método com a assinatura
    public static void main(
        //Classe -> Objeto
        Cliente c1 = new Cli
        c1.idCliente = 1;
        c1.nome = "Sergio Me
        c1.email = "sergio.cot
        System.out.println("Id....:
        System.out.println("Nome...:
        System.out.println("Email...:
        System.out.println("Bem vindo ao Java - Turma BRQ COTI");
}

```

**Context Menu (Run As):**

- Java Application
- Run Configurations...
- 1 Java Application
- Alt-Shift-X, J
- Run As
- Validate
- Team
- Compare With
- Replace With
- Preferences...
- Remove from Context

**Bottom Window (Console):**

```

Java - javaAula01/src/main/Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Main.java Cliente.java Problems Javadoc Declaration Console
javaAula01
src
entity
main
JRE System Library [jre8]
Main.java
import entity.Cliente; //import
//Classe utilizada para executar
public class Main {
    //Em java, para executar
    //método com a assinatura
    public static void main(
        //Classe -> Objeto
        Cliente c1 = new Cli
        c1.idCliente = 1;
        c1.nome = "Sergio Mendes
        c1.email = "sergio.coti@gmail.com
        System.out.println("Id....: " + c1.idCliente);
        System.out.println("Nome...: " + c1.nome);
        System.out.println("Email...: " + c1.email);
        System.out.println("Bem vindo ao Java - Turma BRQ COTI");
}

```

## Encapsulamento

Toda Classe deve proteger o seu conteúdo do acesso externo. Ou seja, definir visibilidades restritas para seus elementos.

A forma mais comum de encapsulamento é realizando quando definimos os atributos de uma Classe como **privados**, e criamos métodos para controlar sua entrada ou saída de dados.

Estes métodos são chamados de set e get

- **set** - entrada
- **get** - saída

```
//localização do pacote
package entity;

//declaração da Classe
//Nome da Classe Cliente deve ser o nome do arquivo Cliente.java
//Deve ser uma classe de acesso público
public class Cliente { //JavaBean (POJO - Plain Old Java Object)

    //Atributos -> dados pertencentes à Classe
    //Visibilidade - tipo - nome
    //public -> acesso total ao elemento
    //private -> acesso somente dentro da própria Classe
    //tipos de dados (primitivos e wrappers (Classes))
    private Integer idCliente;
    private String nome;
    private String email;

    //Encapsulamento (Métodos para entrada/saída)
    //set(entrada) / get(saída)
    public void setIdCliente(Integer idCliente){ //entrada
        //this -> acesso ao elemento da Classe
        this.idCliente = idCliente;
    }

    public Integer getIdCliente(){ //saída
        return idCliente; //retornar o valor do atributo
    }

    public void setNome(String nome){
        this.nome = nome;
    }

    public String getNome(){
        return nome;
    }
}
```

```

public void setEmail(String email){
    this.email = email;
}

public String getEmail(){
    return email;
}
}

```

Testando a execução...

```

package main;

import entity.Cliente; //importando a classe Cliente
import entity.PessoaFisica;

//Classe utilizada para executar o projeto Java local
public class Main {

    //Em java, para executar uma Classe, é necessário um
    //método com a assinatura: static void main
    public static void main(String[] args) {

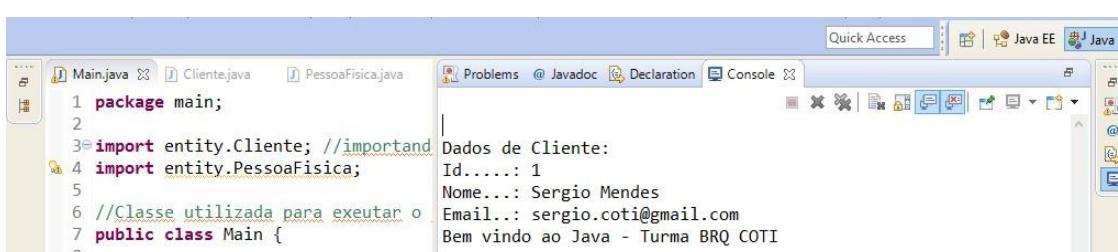
        //Classe -> Objeto
        Cliente c1 = new Cliente();
        c1.setIdCliente(1);
        c1.setNome("Sergio Mendes");
        c1.setEmail("sergio.coti@gmail.com");

        System.out.println("\nDados de Cliente:");
        System.out.println("Id.....: " + c1.getIdCliente());
        System.out.println("Nome...: " + c1.getNome());
        System.out.println("Email..: " + c1.getEmail());

        //imprimir mensagem no console
        //syso -> ctrl + espaço
        System.out.println("Bem vindo ao Java - Turma BRQ COTI");
    }
}

```

Resultado...



## Construtor

Método padrão de toda Classe para inicializar seu conteúdo, ou seja, definir espaço de memória. Toda Classe Java já possui um construtor default caso nenhum seja escrito pelo programador.

É boa prática definirmos construtores para as classes de entidade, com o objetivo de definir as maneiras de inicializar os objetos.

```
//localização do pacote
package entity;

//declaração da Classe
//Nome da Classe Cliente deve ser o nome do arquivo Cliente.java
//Deve ser uma classe de acesso publico
public class Cliente { //JavaBean (POJO - Plain Old Java Object)

    //Atributos -> dados pertencentes a Classe
    //Visibilidade - tipo - nome
    //public -> acesso total ao elemento
    //private -> acesso somente dentro da propria Classe
    //tipos de dados (primitivos e wrappers (Classes))
    private Integer idCliente;
    private String nome;
    private String email;

    //Construtor é um método que tem o mesmo nome da Classe
    //e não possui tipo definido de retorno
    public Cliente() {
        //Construtor default da Classe (vazio)
    }

    //Encapsulamento (Métodos para entrada/saída)
    //set(entrada) / get(saída)
    public void setIdCliente(Integer idCliente){ //entrada
        //this -> acesso ao elemento da Classe
        this.idCliente = idCliente;
    }

    public Integer getIdCliente(){ //saída
        return idCliente; //retornar o valor do atributo
    }

    public void setNome(String nome){
        this.nome = nome;
    }

    public String getNome(){
        return nome;
    }
}
```

```
}

public void setEmail(String email){
    this.email = email;
}

public String getEmail(){
    return email;
}
}
```

Podemos criar em uma Classe vários construtores, mantendo o mesmo nome dos métodos porem alterando a sua entrada de parâmetros. O nome desta prática em Orientação a Objetos chama-se **Sobrecarga de Métodos**.

```
//localização do pacote
package entity;

//declaração da Classe
//Nome da Classe Cliente deve ser o nome do arquivo Cliente.java
//Deve ser uma classe de acesso publico
public class Cliente { //JavaBean (POJO - Plain Old Java Object)

    //Atributos -> dados pertencentes a Classe
    //Visibilidade - tipo - nome
    //public -> acesso total ao elemento
    //private -> acesso somente dentro da propria Classe
    //tipos de dados (primitivos e wrappers (Classes))
    private Integer idCliente;
    private String nome;
    private String email;

    //Construtor é um método que tem o mesmo nome da Classe
    //e não possui tipo definido de retorno
    public Cliente() {
        //Construtor default da Classe (vazio)
    }

    //Construtor com entrada de parametros (dados)
    //Sobrecarga de Métodos (Overloading)
    //Métodos com o mesmo nome, na mesma Classe,
    //com entrada de dados diferentes
    public Cliente(Integer idCliente, String nome,
                   String email){
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
    }
}
```

```
//Encapsulamento (Métodos para entrada/saida)
//set(entrada) / get(saida)
public void setIdCliente(Integer idCliente){ //entrada
    //this -> acesso ao elemento da Classe
    this.idCliente = idCliente;
}

public Integer getIdCliente(){ //saída
    return idCliente; //retornar o valor do atributo
}

public void setNome(String nome){
    this.nome = nome;
}

public String getNome(){
    return nome;
}

public void setEmail(String email){
    this.email = email;
}

public String getEmail(){
    return email;
}
}
```

Testando...

```
package main;

import entity.Cliente; //importando a classe Cliente

//Classe utilizada para executar o projeto Java local
public class Main {

    //Em java, para executar uma Classe, é necessário um
    //método com a assinatura: static void main
    public static void main(String[] args) {

        //Classe -> Objeto
        Cliente c1 = new Cliente(1, "Ana", "ana@gmail");

        //c1.setIdCliente(1);
        //c1.setNome("Sergio Mendes");
        //c1.setEmail("sergio.coti@gmail.com");

        System.out.println("\nDados de Cliente:");
        System.out.println("Id.....: " + c1.getIdCliente());
```

```

        System.out.println("Nome...: " + c1.getNome());
        System.out.println("Email...: " + c1.getEmail());

        //imprimir mensagem no console
        //syso -> ctrl + espaço
        System.out.println("Bem vindo ao Java - Turma BRQ COTI");
    }
}

```

Podemos criar vários construtores em uma Classe. Contanto que todos os construtores tenham como características:

- Nome da Classe
- Não possuir tipo de retorno
- Possuir entrada de parâmetros diferentes

## Sobrecarga de Métodos

Métodos com o mesmo nome, na mesma Classe, porém com entrada de dados diferentes (parâmetros diferentes)

Exemplo:

### Sobrecarga de Construtores

```

//Construtor é um método que tem o mesmo nome da Classe
//e não possui tipo definido de retorno
public Cliente() {
    //Construtor default da Classe (vazio)
}

//Construtor com entrada de parametros (dados)
//Sobrecarga de Métodos (Overloading)
//Métodos com o mesmo nome, na mesma Classe, com entrada de
//dados diferentes
public Cliente(Integer idCliente, String nome, String email){
    this.idCliente = idCliente;
    this.nome = nome;
    this.email = email;
}

```

### this

Palavra reservada para fazer referencia a elementos (atributos ou métodos) da sua própria Classe.

### super

Palavra reservada para fazer referencia a elementos (atributos ou métodos) da SuperClasse, utilizado na herança.

## Relacionamentos entre Classes

Toda Classe em C# relaciona-se com outras Classes por meio do verbo **SER** ou **TER**

### Herança (**extends**)

Relacionamento baseado no uso do verbo SER. Define uma Generalização / Especialização, relacionamento de super classe com sub classe (hierarquia)

Exemplo:

PessoaFisica É UM Cliente

```
package entity;

//extends -> herança
public class PessoaFisica extends Cliente{

    private String cpf;

    public PessoaFisica() {
        //Construtor default da Classe
    }

    public PessoaFisica(Integer idCliente, String nome,
                        String email, String cpf) {
        super(idCliente, nome, email);
        this.cpf = cpf;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
}
```

Note que o Construtor da Classe PessoaFisica está utilizando a palavra reservada super para fazer referência ao construtor da sua SuperClasse Cliente

```
public PessoaFisica(Integer idCliente, String nome,
                    String email, String cpf) {
    super(idCliente, nome, email);
    this.cpf = cpf;
}
```

Executando...

```
package main;

import entity.Cliente; //importando a classe Cliente
import entity.PessoaFisica;

//Classe utilizada para executar o projeto Java local
public class Main {

    //Em java, para executar uma Classe, é necessário um
    //método com a assinatura: static void main
    public static void main(String[] args) {

        //Classe -> Objeto
        Cliente c1 = new Cliente(1, "Ana", "ana@gmail");
                    //Construtor(es)

        //c1.setIdCliente(1);
        //c1.setNome("Sergio Mendes");
        //c1.setEmail("sergio.coti@gmail.com");

        System.out.println("\nDados de Cliente:");
        System.out.println("Id.....: " + c1.getIdCliente());
        System.out.println("Nome....: " + c1.getNome());
        System.out.println("Email...: " + c1.getEmail());

        PessoaFisica pf = new PessoaFisica(2, "Joao",
                                           "joao@gmail.com", "0123456789");

        //pf.setIdCliente(2);
        //pf.setNome("Joao");
        //pf.setEmail("joao@gmail.com");
        //pf.setCpf("0123456789");

        System.out.println("\nDados de Pessoa Fisica");
        System.out.println("Id.....: " + pf.getIdCliente());
        System.out.println("Nome....: " + pf.getNome());
        System.out.println("Email...: " + pf.getEmail());
        System.out.println("Cpf.....: " + pf.getCpf());

        //imprimir mensagem no console
        //syso -> ctrl + espaço
        System.out.println("Bem vindo ao Java - Turma BRQ COTI");

    }

}
```

## Saída do Programa...

The screenshot shows the Eclipse IDE interface with the title bar "Java - javaAula01/src/main/Main.java - Eclipse". The left pane displays the Java code for Main.java, which includes imports for Client and PessoaFisica, and logic to print client and physical person details to the console. The right pane shows the "Console" tab active, displaying the printed output: "Dados de Cliente", "Dados de Pessoa Física", and a welcome message. The bottom status bar shows the date and time: "07:49 08/05/2014".

```

16     //c1.setIdCliente(1);
17     //c1.setNome("Sergio Mende
18     //c1.setEmail("sergio.coti
19
20     System.out.println("\nDado
21     System.out.println("Id.....
22     System.out.println("Nome..
23     System.out.println("Email.
24
25     PessoaFisica pf = new Pess
26
27     //pf.setIdCliente(2);
28     //pf.setNome("Joao");
29     //pf.setEmail("joao@gmail.
30     //pf.setCpf("0123456789");
31
32     System.out.println("\nDado <
33     System.out.println("Id.....
34     System.out.println("Nome.... + pf.getNome());
35     System.out.println("Email... + pf.getEmail());
36     System.out.println("Cpf.... + pf.getCpf());
37
38
39     //imprimir mensagem no console
40     //syso -> ctrl + espaço
41     System.out.println("Bem vindo ao Java - Turma BRQ COTI");
42

```

### Tipos primitivos de dados

Na linguagem Java, podemos utilizar os seguintes tipos de dados primitivos:

<b>Tipos lógicos</b>	
<b>boolean</b>	Representam apenas <b>1 bit</b> de informação (0 ou 1). Podem assumir apenas os valores <b>true</b> e <b>false</b> .
<b>Tipos caractere</b>	
<b>char</b>	Representam notação de caracteres de <b>16 bits</b> (2 bytes) para formato Unicode UTF-16. Podem assumir caracteres entre '\u0000' a '\uffff' e valores numéricos entre <b>0a 65535</b> .
<b>Tipos numéricos inteiros</b>	
<b>byte</b>	Números inteiros de <b>8 bits</b> (1 byte). Podem assumir valores entre <b>-128 a 127</b> .
<b>short</b>	Representam números inteiros de <b>16 bits</b> (2 bytes). Podem assumir valores entre <b>32.768</b> até <b>32.767</b> .
<b>Int</b>	Representam números inteiros de <b>32 bits</b> (4 bytes). Podem assumir valores entre <b>2.147.483.648</b> até <b>2.147.483.647</b> .
<b>long</b>	Representam números inteiros de <b>64 bits</b> (8 bytes). Podem assumir valores entre-

	<b>9.223.372.036.854.775.808</b> até <b>9.223.372.036.854.775.807.</b>
<b>Tipos numéricos reais</b>	
<b>float</b>	Representam números reais de <b>32 bits</b> com precisão simples. Podem assumir valores de ponto flutuante com formato definido pela especificação IEEE 754.
<b>double</b>	Representam números reais de <b>64 bits</b> com precisão dupla. Assim como o float. Podem assumir valores de ponto flutuante com formato definido pela especificação IEEE 754.

### Tipos Wrappers

As classes Wrappers são equivalentes aos tipos primitivos, mas possuem métodos que facilitam a manipulação de seus dados. Herdam de Object e pertencem ao pacote java.lang

- São elas:

<b>Tipo primitivo</b>	<b>Classe wrapper</b>
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

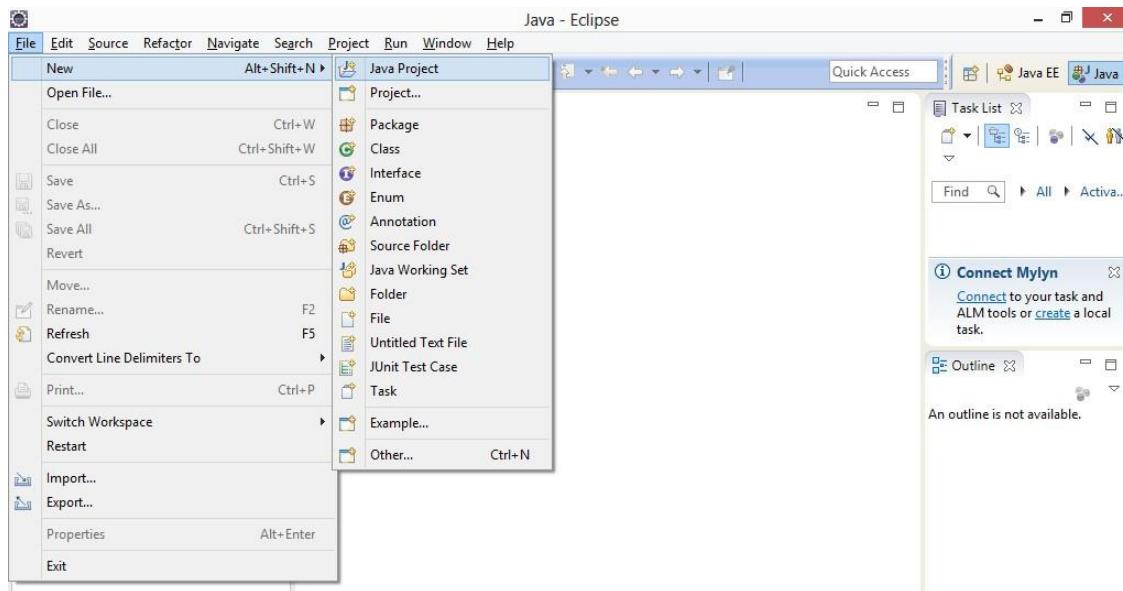
### Modificadores de acesso do Java:

Em Java, podemos criar atributos ou métodos com os seguintes modificadores de acesso:

<b>public</b>	Acesso total, qualquer classe tem acesso ao método (não indicado para usar em atributos de entidade)
<b>protected</b>	Acesso permitido por herança e por classes do mesmo pacote
<b>default / friendly</b>	Acesso permitido somente a classes do mesmo pacote, este acesso é definido quando não declaramos nenhum modificador
<b>private</b>	Acesso permitido somente dentro da própria classe. (indicado para atributos de classes de entidade)



### Criando o Projeto Java local (SE)

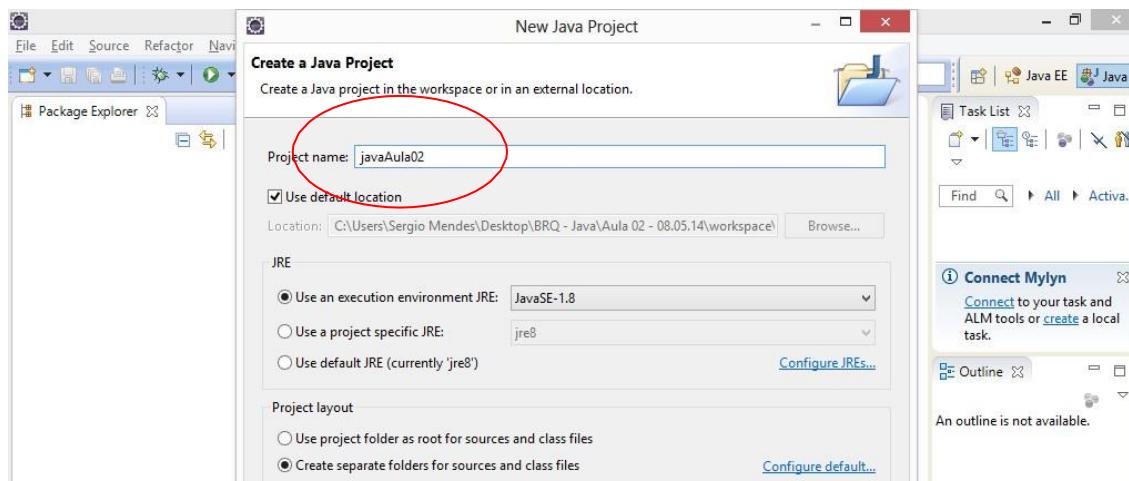


## Orientação a Objetos Modelagem de entidades (**JavaBean**)

### JavaBean

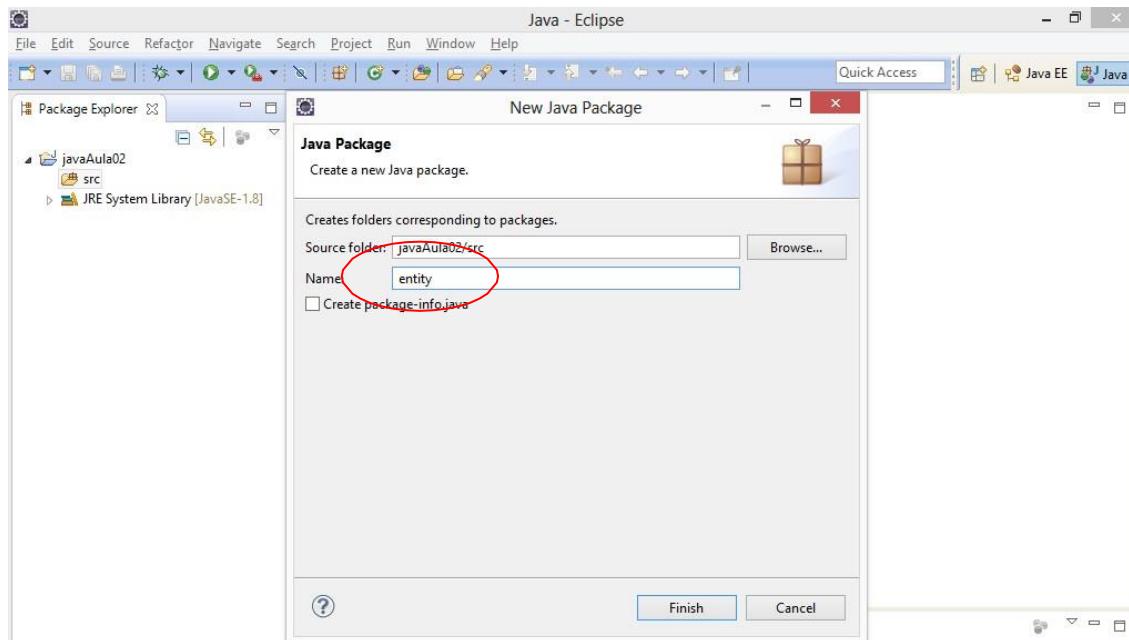
Classes Java que seguem um padrão voltado para modelagem de entidades. Devem ser classes reutilizáveis, coesas e fracamente acopladas, seguindo as especificações abaixo:

- Características:
  - Atributos privados
  - Sobrecarga de Construtores
  - Encapsulamento
  - Sobrescrita de Métodos da Classe Object
  - Serializável

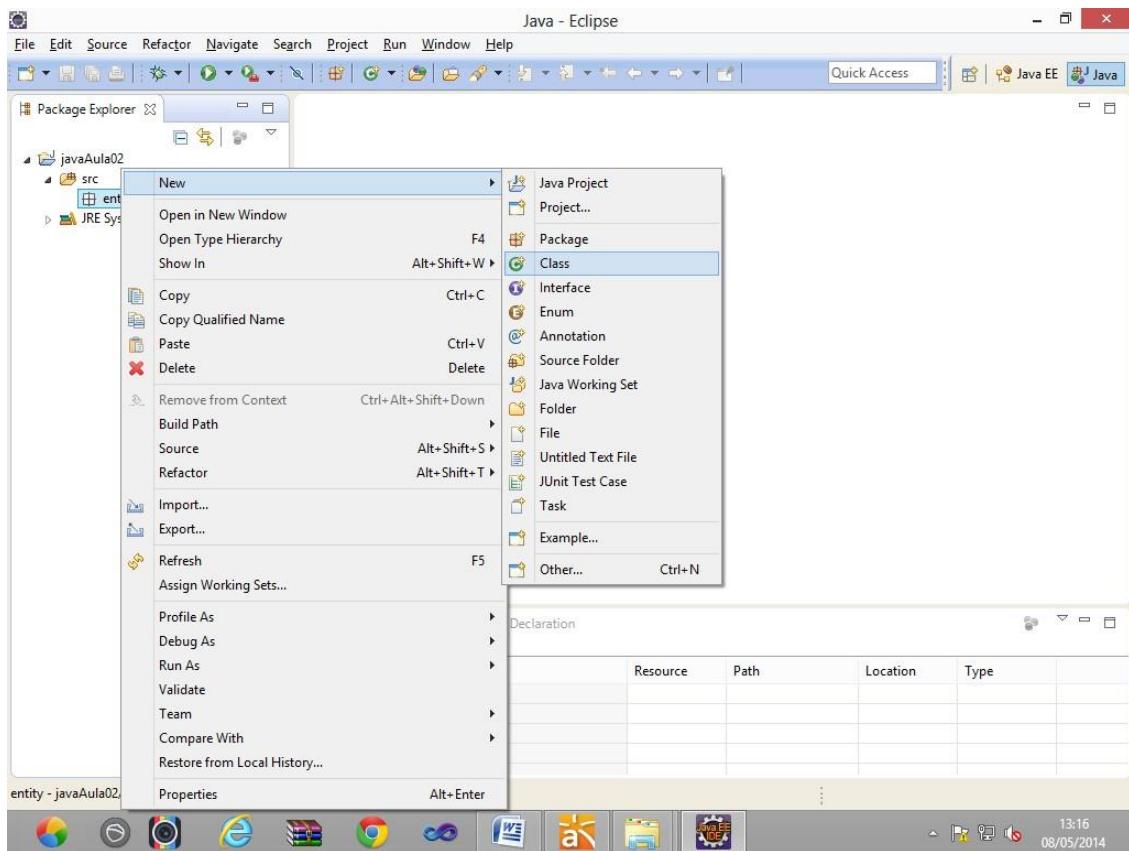




### Criando o pacote para as Classes de entidade (javabeans)



### Criando a Classe Funcionario



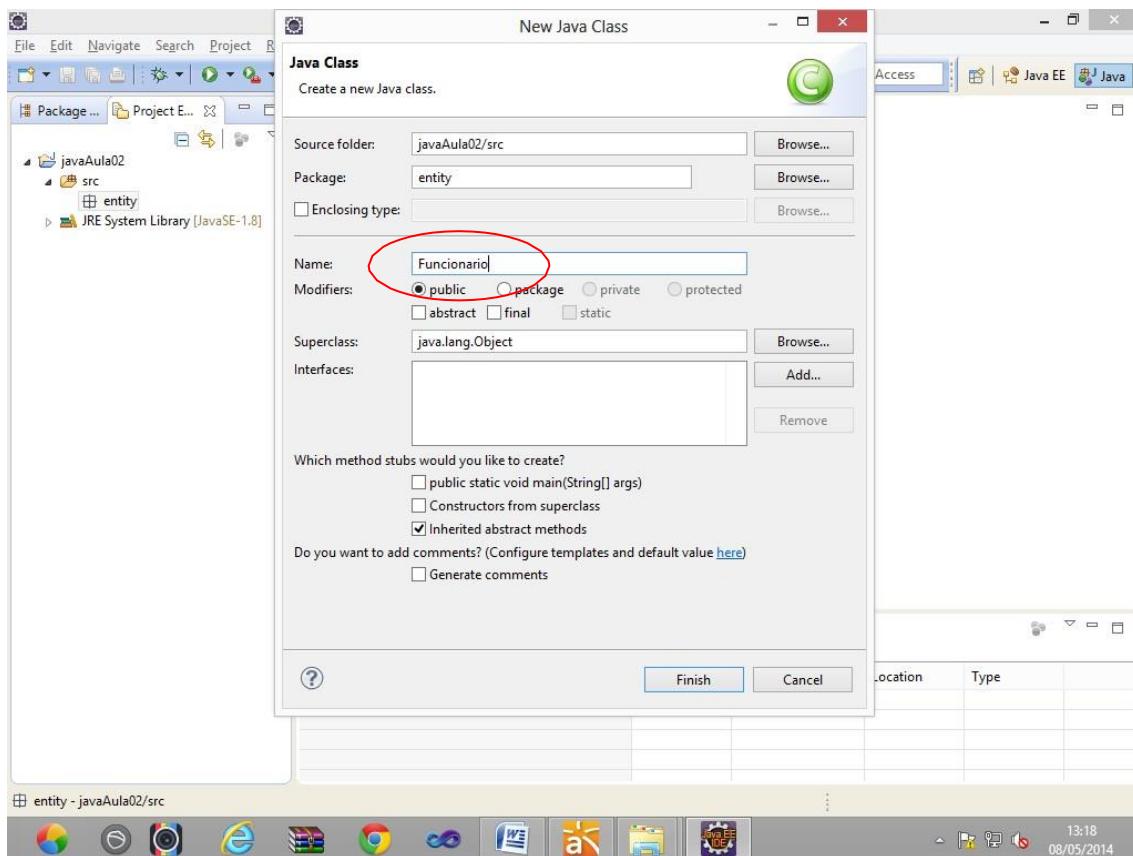


# Java WebDeveloper - BRQ

Quinta-feira, 08 de Maio de 2014

Padrão JavaBean, Classe Object, Herança, Sobrecarga e Sobreescrita de Métodos, Entrada de dados.

Aula  
02



```
package entity;

public class Funcionario { //JavaBean

    //declaração dos atributos
    private Integer idFuncionario;
    private String nome;
    private Double salario;

    public Funcionario() {
        //Construtor default -> padrão (vazio)
    }

    //Sobrecarga de Métodos (Overloading)
    public Funcionario(Integer idFuncionario, String nome,
                       Double salario){
        this.idFuncionario = idFuncionario;
        this.nome = nome;
        this.salario = salario;
    }

    public void setIdFuncionario(Integer idFuncionario){ //entrada
        this.idFuncionario = idFuncionario;
    }
}
```



```
public Integer getIdFuncionario(){ //saída
    return idFuncionario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getSalario() {
    return salario;
}

public void setSalario(Double salario) {
    this.salario = salario;
}

}
```

Note que a Classe possui atributos privados e métodos para entrada e saída dos dados denominados **set (entrada)** e **get (saída)**

A Classe também possui 2 métodos Construtores, o primeiro sendo o **Construtor default (padrão)** da Classe e o segundo sendo o **Construtor que permite a entrada de dados** para a Classe.

Quando escrevemos dois métodos com o mesmo nome, porém com entrada de dados diferentes (como no caso dos construtores), denominados este procedimento de **Sobrecarga de Métodos**

---

## Em Java, todas as classes por definição são herança de Object

Exemplo: **public class Funcionario extends Object**

A Classe Object do Java possui 3 métodos que podemos **sobrescrever** quando herdamos dela. São eles:

### **toString**

Método utilizado para retornar uma String que representa os dados do objeto, como uma “assinatura” do objeto

### **equals**

Método utilizado para comparar se dois objetos da mesma Classe são iguais, baseado em um critério definido pelo desenvolvedor.



### hashCode

Método utilizado para retornar o endereço de hashCode do objeto, utilizado por bibliotecas do Java que realizam ordenação e classificação de objetos.

## Sobrescrevendo o Método **equals**

```
package entity;

public class Funcionario{ // JavaBean

    // declaração dos atributos
    private Integer idFuncionario;
    private String nome;
    private Double salario;

    public Funcionario() {
        // Construtor default -> padrão (vazio)
    }

    // Sobrecarga de Métodos (Overloading)
    public Funcionario(Integer idFuncionario, String nome,
                       Double salario) {
        this.idFuncionario = idFuncionario;
        this.nome = nome;
        this.salario = salario;
    }

    public void setIdFuncionario(Integer idFuncionario) { // entrada
        this.idFuncionario = idFuncionario;
    }

    public Integer getIdFuncionario() { // saída
        return idFuncionario;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Double getSalario() {
        return salario;
    }
}
```



```
public void setSalario(Double salario) {
    this.salario = salario;
}

@Override
// annotation -> sobrescrita
public boolean equals(Object obj) {

    // verificar se o parametro 'obj' é do tipo Funcionario
    // Se 'obj' é uma instancia de Funcionario
    if (obj instanceof Funcionario) { // é do tipo?

        // converter 'obj' para Funcionario
        Funcionario f = (Funcionario) obj; // casting
        return f.getIdFuncionario().equals(idFuncionario);
        // if(f.getIdFuncionario() == this.idFuncionario){
        // return true;
        // }
    }

    // default retornar falso
    return false;
}
}
```

## @Override

Especifica que o método é uma sobrescrita, ou seja, que está redefinindo o método equals herdado da Classe Object.

## instanceof

Verifica se o parâmetro obj passado no método equals é do tipo Funcionario. Aqui verifica-se se um parâmetro do tipo Object (SuperTipo de todas as Classes do Java) é uma instância da Classe Funcionario.

## Funcionario f = (Funcionario) obj;

Aqui o parâmetro obj uma vez verificado como sendo uma instância de Funcionario é então convertido para o tipo Funcionario. A operação chama-se casting, onde uma variável de um tipo "maior" (superclasse) é convertido para uma variável de um tipo "menor" ou subtipo.

## return f.getIdFuncionario().equals(idFuncionario);

Aqui é realizado a regra de comparação onde verificamos se o idFuncionario do objeto 'f' é igual ao idFuncionario da Classe em que estamos. Caso sim, o retorno do método equals é verdadeiro.



Testando a execução do método equals...

```
package main;

import entity.Funcionario;

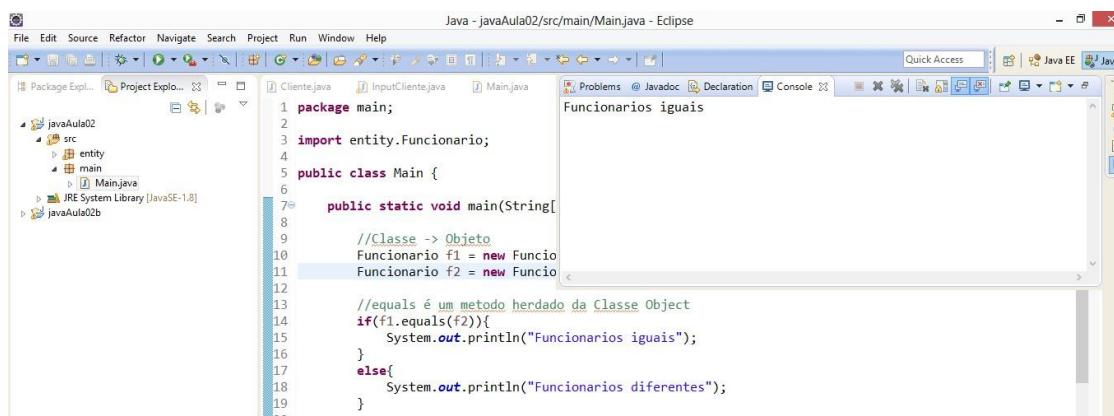
public class Main {

    public static void main(String[] args) {

        //Classe -> Objeto
        Funcionario f1 = new Funcionario(1, "Ana", 1000.0);
        Funcionario f2 = new Funcionario(1, "Rui", 2000.0);

        //equals é um metodo herdado da Classe Object
        if(f1.equals(f2)){
            System.out.println("Funcionarios iguais");
        }
        else{
            System.out.println("Funcionarios diferentes");
        }
    }
}
```

Resultado...



## Sobrescrevendo o Método **toString**

Método da Classe Object utilizado para retornar uma linha de texto contendo os dados (atributos) da sua Classe...

```
package entity;

public class Funcionario{ // JavaBean

    // declaração dos atributos
    private Integer idFuncionario;
    private String nome;
```



```
private Double salario;

public Funcionario() {
    // Construtor default -> padrão (vazio)
}

// Sobreescrita de Métodos (Overloading)
public Funcionario(Integer idFuncionario, String nome,
                    Double salario) {
    this.idFuncionario = idFuncionario;
    this.nome = nome;
    this.salario = salario;
}

public void setIdFuncionario(Integer idFuncionario) { // entrada
    this.idFuncionario = idFuncionario;
}

public Integer getIdFuncionario() { // saída
    return idFuncionario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getSalario() {
    return salario;
}

public void setSalario(Double salario) {
    this.salario = salario;
}

@Override
// annotation -> sobreescrita
public boolean equals(Object obj) {

    // verificar se o parâmetro 'obj' é do tipo Funcionario
    // Se 'obj' é uma instância de Funcionario
    if (obj instanceof Funcionario) { // é do tipo?

        // converter 'obj' para Funcionario
        Funcionario f = (Funcionario) obj; // casting
        return f.getIdFuncionario().equals(idFuncionario);
    }
    return false;
}
```



```
@Override
public String toString() {
    return idFuncionario + ", " + nome + ", " + salario;
}
```

Testando a execução do método `toString()`...

```
package main;

import entity.Funcionario;

public class Main {

    public static void main(String[] args) {

        //Classe -> Objeto
        Funcionario f1 = new Funcionario(1, "Ana", 1000.0);
        Funcionario f2 = new Funcionario(1, "Rui", 2000.0);

        //equals é um metodo herdado da Classe Object
        if(f1.equals(f2)){
            System.out.println("Funcionarios iguais");
        }
        else{
            System.out.println("Funcionarios diferentes");
        }

        System.out.println("Funcionario: " + f1); //toString
        System.out.println("Funcionario: " + f2); //toString
    }
}
```

Resultado...

The screenshot shows the Eclipse IDE interface with the code editor displaying the `Main.java` file. The code is identical to the one above, but the output window shows the results of the `System.out.println` statements:

```
Fucionarios iguais
Funcionario: 1, Ana, 1000.0
Funcionario: 1, Rui, 2000.0
```



## Sobrescrevendo o Método **hashCode**

Geralmente, o método hashCode utiliza como regra de geração de numero hashCode (que será utilizado em regras de ordenação) o mesmo atributo da classe para o qual foi feito a regra de comparação do método equals.

```
package entity;

public class Funcionario{ // JavaBean

    // declaração dos atributos
    private Integer idFuncionario;
    private String nome;
    private Double salario;

    public Funcionario() {
        // Construtor default -> padrão (vazio)
    }

    // Sobrecarga de Métodos (Overloading)
    public Funcionario(Integer idFuncionario, String nome,
                       Double salario) {
        this.idFuncionario = idFuncionario;
        this.nome = nome;
        this.salario = salario;
    }

    public void setIdFuncionario(Integer idFuncionario) { // entrada
        this.idFuncionario = idFuncionario;
    }

    public Integer getIdFuncionario() { // saída
        return idFuncionario;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Double getSalario() {
        return salario;
    }

    public void setSalario(Double salario) {
        this.salario = salario;
    }
}
```



```
@Override
// annotation -> sobreescrita
public boolean equals(Object obj) {

    // verificar se o parametro 'obj' é do tipo Funcionario
    // Se 'obj' é uma instancia de Funcionario
    if (obj instanceof Funcionario) { // é do tipo?

        // converter 'obj' para Funcionario
        Funcionario f = (Funcionario) obj; // casting
        return f.getIdFuncionario().equals(idFuncionario);
        // if(f.getIdFuncionario() == this.idFuncionario){
        // return true;
        // }
    }

    // default retornar falso
    return false;
}

@Override
public String toString() {
    return idFuncionario + ", " + nome + ", " + salario;
}

@Override
public int hashCode() { //regra de ordenação
    //Geralmente a regra de ordenação do hashCode
    //é baseada na regra do equals
    return idFuncionario.hashCode();
}
}
```

Testando de forma simples a execução do método hashCode...  
A importância do hashCode será vista em detalhes quando estudarmos a biblioteca Collections do Java

```
package main;

import entity.Funcionario;

public class Main {

    public static void main(String[] args) {

        //Classe -> Objeto
        Funcionario f1 = new Funcionario(1, "Ana", 1000.0);
        Funcionario f2 = new Funcionario(1, "Rui", 2000.0);
    }
}
```



```
//equals é um metodo herdado da Classe Object
if(f1.equals(f2)){
    System.out.println("Funcionarios iguais");
}
else{
    System.out.println("Funcionarios diferentes");
}

System.out.println("Funcionario: " + f1); //toString
System.out.println("Funcionario: " + f2); //toString

System.out.println("HashCode de f1: " + f1.hashCode());
System.out.println("HashCode de f2: " + f2.hashCode());
}
}
```

Resultado...

```
Fucionarios iguais
Funcionario: 1, Ana, 1000.0
Funcionario: 1, Rui, 2000.0
HashCode de f1: 1
HashCode de f2: 1
```

## Sobrecarga de Métodos

Em Java, a sobrecarga de métodos (**overloading**) ocorre quando em uma Classe, criamos métodos com nomes iguais, porém, com entrada de parâmetros diferentes.

Exemplo:

```
package entity;

public class Test {

    public void imprimir(int a){
        System.out.println("Imprimindo primitivo: " + a);
    }
}
```



```
public void imprimir(Integer a){  
    System.out.println("Imprimindo wrapper: " + a);  
}  
  
public static void main(String[] args) {  
  
    Test t = new Test();  
    t.imprimir(10);  
  
}  
}
```

Executando...

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java - javaAula02/src/entity/Test.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Quick Access:** A toolbar with icons for recent files, search, and other tools.
- Java EE:** A toolbar icon for Java Enterprise Edition.
- Project Explorer:** Shows the project structure:
  - javaAula02
  - src
    - entity
      - Analista.java
      - Funcionario.java
      - Programador.java
      - Test.java
      - Test2.java
      - Test3.java
    - main
      - Main.java
  - JRE System Library [JavaSE-1.8]
  - javaAula02b
- Editor:** Displays the content of `Test.java`:

```
1 package entity;
2
3 public class Test {
4
5     public void imprimir(int a){
6         System.out.println("Imprimi
7     }
8
9     public void imprimir(Integer a)
10    System.out.println("Imprimi
11 }
12
13 public static void main(String[] args) {
14
15     Test t = new Test();
16     t.imprimir(10);
17
18 }
19
20 }
21
```
- Console:** Shows the output of the application's execution:

```
<terminated> Test [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe (08/05/2014 20:43:46)
Imprimindo primitivo: 10
```
- Bottom:** The Windows taskbar with various pinned icons.

A palavra reservada **Super**

Em Java, a palavra reservada super é utilizada para fazer referência a elementos localizados na Classe "Pai" ou SuperClasse de uma herança.

Podemos utilizar também o operador super de forma a permitir que o construtor de uma SubClasse execute o Construtor de uma SuperClasse.

Em Java, quando criamos uma herança, todo construtor de uma subclasse SEMPRE executa implicitamente o construtor padrão da sua superclasse.



Exemplo...

```
package entity;

class A{
    public A() {
        System.out.println("Criando A...");
    }
}

class B extends A{
    public B() {
        super();
        System.out.println("Criando B...");
    }
}

public class Test2 {

    public static void main(String[] args) {
        B b = new B();
    }
}
```

Resultado da execução do método main...

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages `entity` and `main`, and files like `Analista.java`, `Funcionario.java`, `Programador.java`, `Test.java`, `Test2.java`, and `Test3.java`.
- Code Editor:** Displays the Java code for `Test2.java` and `entity/A.java` and `B.java`. The code is annotated with red circles and arrows highlighting the constructor calls and the `super()` keyword.
- Console View:** Shows the output of the program execution:

```
Criando A...
Criando B...
```
- Bottom Status Bar:** Shows the date and time: 08/05/2014 20:49.



Em Java, o método Construtor sempre é executado quando utilizamos o operador **new**, ou seja, é o primeiro método que toda classe executa, pois é responsável pela inicialização dos objetos.

O único tipo de programa em uma Classe que é executado ANTES do Construtor é um **bloco anônimo**, ou seja, um bloco de programação sem cabeçalho...

```
package entity;

class Teste{

    {
        //bloco anônimo -> executado antes do construtor
        System.out.println("Executando o bloco anônimo");
    }

    public Teste() {
        System.out.println("Executando o Construtor de Teste");
    }
}

public class Test3 {

    public static void main(String[] args) {

        Teste t = new Teste();
    }
}
```

Resultado...

The screenshot shows the Eclipse IDE interface with the code editor open. The code is identical to the one above, with the addition of a red oval and arrow highlighting the line `Teste t = new Teste();`. A red arrow points from this line to the `new` keyword, indicating where the object is created. The Eclipse console window shows two lines of output: "Executando o bloco anônimo" and "Executando o Construtor de Teste".



Em Orientação a Objetos, utilizamos herança quando podemos vincular uma classe à outra através do uso do verbo **SER**

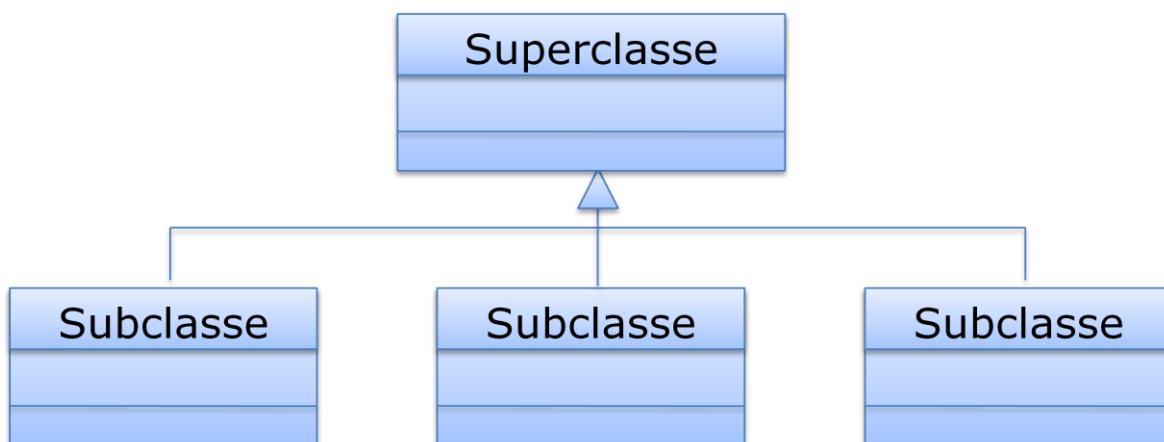
Por exemplo:

- **PessoaFísica é Pessoa**
- **Programador é Funcionário**

Erro:

**Pessoa tem Endereço**

A Herança está diretamente relacionada ao reuso de código. Sendo assim, Classes mais genéricas e menos especializadas possuem características que podem ser herdadas por classes menos genéricas, porém mais especializadas.



Exemplo:

- Analista **É UM** Funcionario
- Programador **É UM** Funcionario

```
package entity;

//Analista É-UM Funcionario
public class Analista extends Funcionario{

    private String tipo;

    public Analista() {
        //Construtor default
    }

    public Analista(Integer idFuncionario, String nome,
                    Double salario, String tipo) {
        super(idFuncionario, nome, salario);
        this.tipo = tipo;
    }
}
```



```
public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

@Override
public boolean equals(Object obj) {

    if(obj instanceof Analista){

        Analista a = (Analista) obj;

        if(a.getIdFuncionario().equals(getIdFuncionario())
            && a.getTipo().equals(tipo)){
            return true; //iguais
        }
    }

    return false; //diferentes
}

@Override
public String toString() {
    return super.toString() + ", " + tipo;
}

-----  

package entity;

public class Programador extends Funcionario{

    private String linguagem;

    public Programador() {

    }

    public Programador(Integer idFuncionario, String nome,
                       Double salario, String linguagem) {
        super(idFuncionario, nome, salario);
        this.linguagem = linguagem;
    }

    public String getLinguagem() {
```



```
        return linguagem;
    }

    public void setLinguagem(String linguagem) {
        this.linguagem = linguagem;
    }

    @Override
    public boolean equals(Object obj) {

        if(obj instanceof Programador){

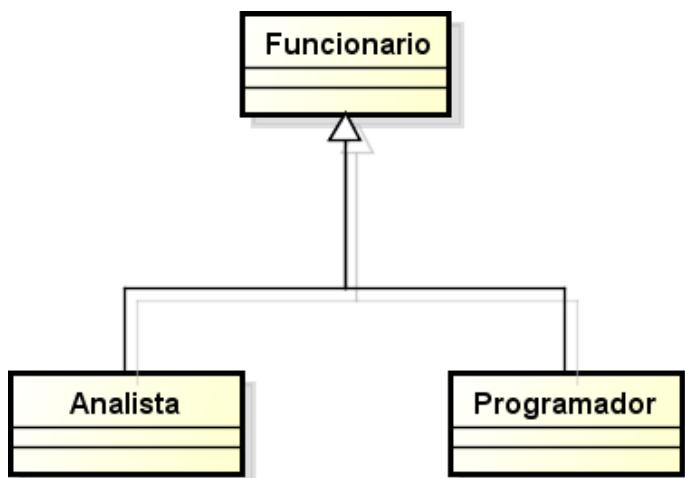
            Programador p = (Programador) obj;

            if(p.getIdFuncionario().equals(getIdFuncionario())
                && p.getLinguagem().equals(linguagem)){
                return true;
            }
        }

        return false;
    }

    @Override
    public String toString() {
        return super.toString() + ", " + linguagem;
    }
}
```

Modelagem em UML do relacionamento de herança...



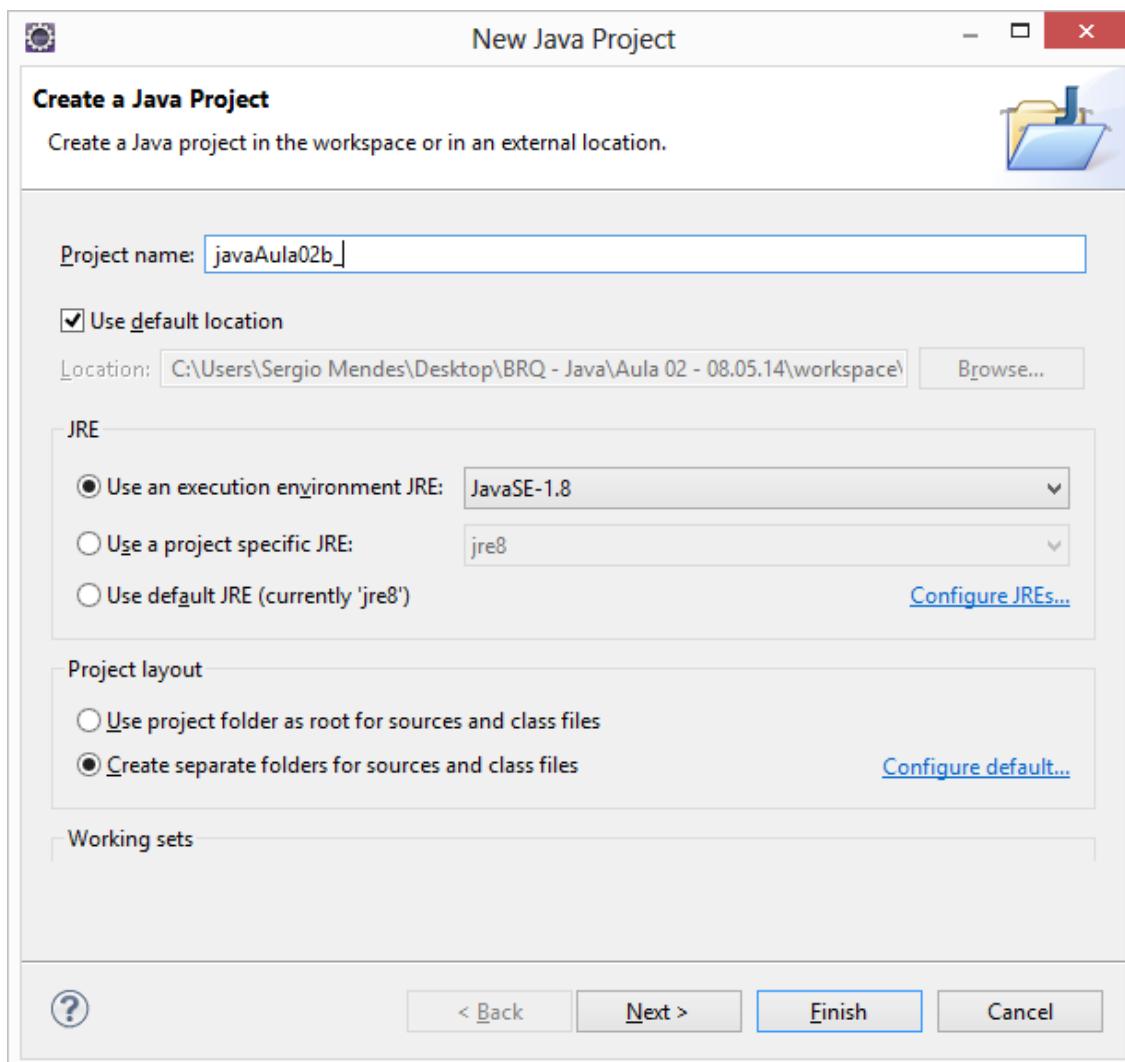
**Importante: Java não permite herança múltipla entre Classes**

Exemplo:

```
public class Estagiario extends Analista, Programador {
```



Criando um novo Projeto...



Criando o JavaBean **Cliente**...

```
package entity;

public class Cliente {

    private Integer idCliente;
    private String nome;
    private String email;

    public Cliente() {
    }

    public Cliente(Integer idCliente, String nome, String email) {
        this.idCliente = idCliente;
        this.nome = nome;
    }
}
```



```
this.email = email;
}

public Integer getIdCliente() {
    return idCliente;
}

public void setIdCliente(Integer idCliente) {
    this.idCliente = idCliente;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

@Override
public String toString() {
    return "Cliente [idCliente=" + idCliente + ", nome="
           + nome + ", email=" + email + "]";
}
}
```

Criando uma Classe para leitura de dados através do Console, utilizando a API **Scanner**

```
package view;

import java.util.Scanner;

public class InputCliente {

    // Método para ler e retornar o id de um Cliente informado
    public Integer lerIdCliente() {

        Scanner leitura = new Scanner(System.in);

        System.out.print("Informe o Id do Cliente.....: ");
        return leitura.nextInt(); // lê e retorna inteiro
    }
}
```



```
// Método para ler e retornar o nome de um Cliente informado
public String lerNome() {

    // Scanner -> Classe Java para leitura de
    //dados em Java local
    Scanner leitura = new Scanner(System.in);

    System.out.print("Informe o Nome do Cliente...: ");
    return leitura.nextLine(); // lê e retorna String
}

// Método para ler e retornar o nome de um Cliente informado
public String lerEmail() {

    // Scanner -> Classe Java para leitura de dados
    //em Java local
    Scanner leitura = new Scanner(System.in);

    System.out.print("Informe o Email do Cliente...: ");
    return leitura.nextLine(); // lê e retorna String
}
```

Executando a leitura de dados no método main...

```
package main;

import view.InputCliente;
import entity.Cliente;

public class Main {

    //Método utilizado para executar uma Classe java
    public static void main(String[] args) {

        Cliente c = new Cliente();
        InputCliente in = new InputCliente();

        c.setIdCliente(in.lerIdCliente());
        c.setNome(in.lerNome());
        c.setEmail(in.lerEmail());

        System.out.println("\nDados do Cliente:");
        System.out.println(c);
    }
}
```



# Java WebDeveloper - BRQ

Quinta-feira, 08 de Maio de 2014

Padrão JavaBean, Classe Object, Herança, Sobrecarga e Sobrescrita de Métodos, Entrada de dados.

Aula  
02

## Resultado...

The screenshot shows the Eclipse IDE interface with the following details:

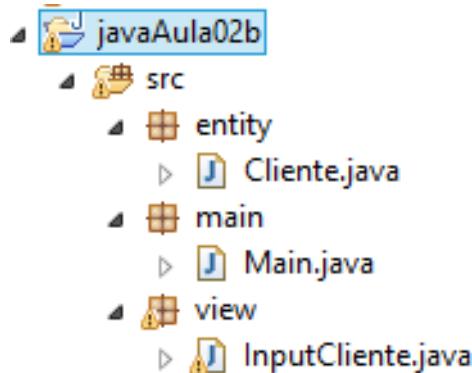
- Project Explorer:** Shows the project structure: javaAula02 and javaAula02b. Under javaAula02b, there is a src folder containing entity (Cliente.java), main (Main.java), and view (InputCliente.java).
- Main.java Content:**

```
1 package main;
2
3 import view.InputCliente;
4 import entity.Cliente;
5
6 public class Main {
7
8     //Método utilizado para executa
9     public static void main(String[
10
11         Cliente c = new Cliente();
12         InputCliente in = new InputCliente();
13
14         c.setIdCliente(in.lerIdCliente());
15         c.setNome(in.lerNome());
16         c.setEmail(in.lerEmail());
17
18         System.out.println("\nDados do Cliente:");
19         System.out.println(c);
20     }
21
22 }
```
- Console Output:** Displays the execution results:

```
<terminated: Main () [Java Application] C:\Program Files (x86)\Java\jre8\bin\java.exe (08/05/2014 21:05:13)
Informe o Id do Cliente.....: 1
Informe o Nome do Cliente...: Sergio Mendes
Informe o Email do Cliente..: sergio.coti@gmail.com

Dados do Cliente:
Cliente [idCliente=1, nome=Sergio Mendes, email=sergio.coti@gmail.com]
```

## Estrutura do projeto...





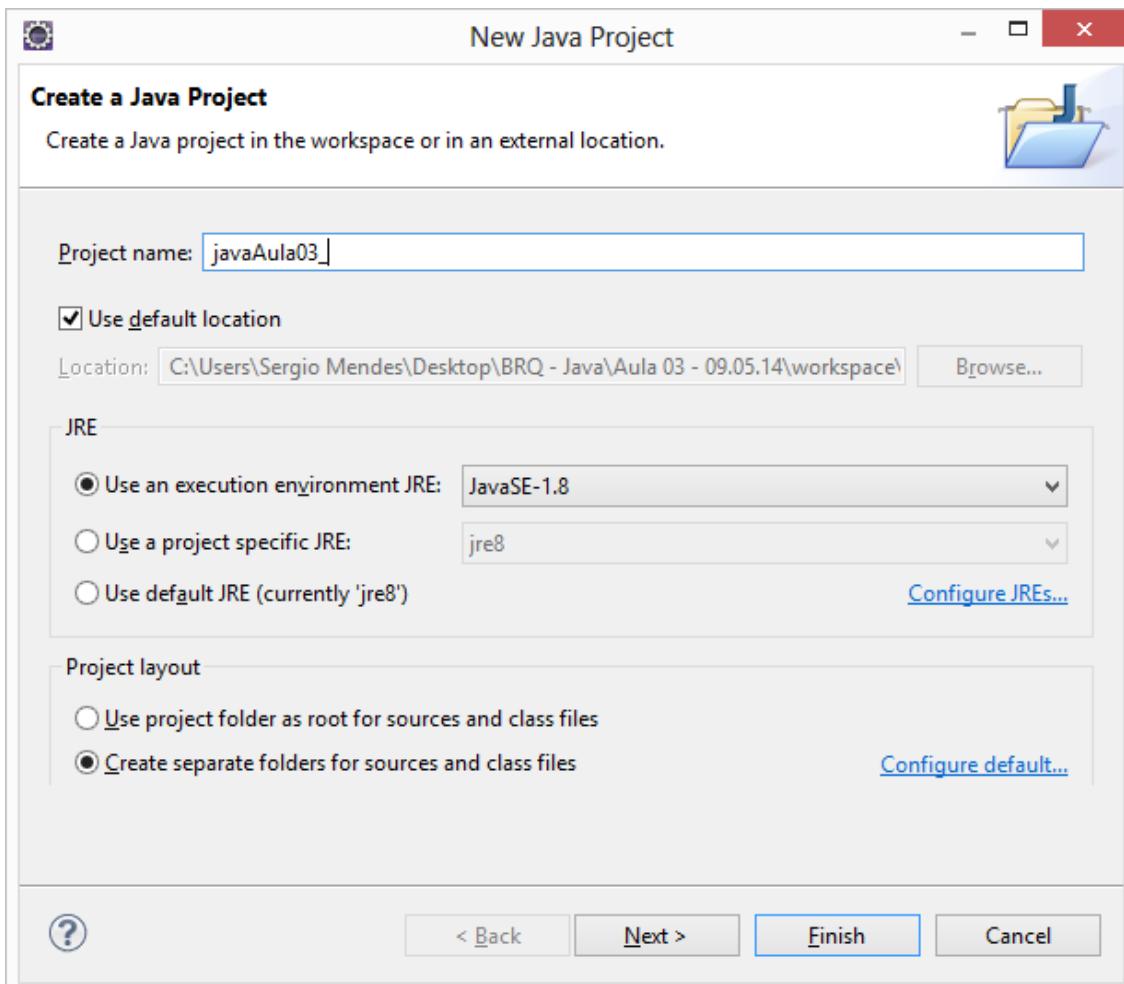
# Java WebDeveloper - BRQ

Sexta-feira, 09 de Maio de 2014

Orientação a Objetos, Tratamento de Exceções, Entrada de dados, Manipulação de Arquivos.

Aula  
03

Criando o Projeto...



Criando a entidade Produto...

```
package entity;

public class Produto { // JavaBean (Classe Java de entidade)

    // Atributos
    private Integer idProduto;
    private String nome;
    private Double preco;

    // Construtores
    public Produto() {
        // Construtor default
    }

    // Sobrecarga de Construtores
    public Produto(Integer idProduto, String nome, Double preco) {
        super();
        this.idProduto = idProduto;
    }
}
```



```
        this.nome = nome;
        this.preco = preco;
    }

    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Double getPreco() {
        return preco;
    }

    public void setPreco(Double preco) {
        this.preco = preco;
    }

    @Override
    public String toString() {
        return "Produto [idProduto=" + idProduto + ", nome="
            + nome + ", preco=" + preco + "]";
    }
}
```

Criando uma Classe para realizar a leitura dos dados de um produto informado pelo Console (Java Local)...

```
package view;

import java.util.Scanner;

public class InputProduto {

    public Integer lerIdProduto(){

        try{ //tentativa

            Scanner s = new Scanner(System.in);

```



# Java WebDeveloper - BRQ

Sexta-feira, 09 de Maio de 2014

Orientação a Objetos, Tratamento de Exceções, Entrada de dados, Manipulação de Arquivos.

Aula  
03

```
System.out.print("Informe o Id do Produto...: ");
Integer idProduto = s.nextInt();
//Lê o valor informado

if(idProduto > 0){ //teste de validação
    return idProduto; //Ok
}
else{
    //redirecionar para o catch
    //-> forçar/lançar uma exceção
    throw new Exception("Valor deve ser
                        maior que zero");
}
catch(Exception e){ //captura do erro / exceção
    System.out.println("Erro ao ler o id do produto -> "
                       + e.getMessage());
    return lerIdProduto(); //recursividade
}
}

public String lerNome(){

try{
    Scanner s = new Scanner(System.in);

    System.out.print("Informe o Nome do Produto.: ");
    String nome = s.nextLine();

    return nome;
}
catch(Exception e){
    System.out.println("Erro ao ler o nome
                      do produto -> " + e.getMessage());
    return lerNome();
}
}

public Double lerPreco(){

try{
    Scanner s = new Scanner(System.in);

    System.out.print("Informe o Preço do Produto: ");
    Double preco = s.nextDouble();

    if(preco > 0 && preco <= 10000){
        return preco;
    }
}
```



```
        else{
            //lança / dispara uma nova exceção
            throw new Exception("Valor deve estar
                                entre 1 e 10000");
        }
    }
    catch(Exception e){
        System.out.println("Erro ao ler o preco
                           do produto -> " + e.getMessage());
        return lerPreco();
    }
}
}
```

Executando através do método main...

```
package main;

import persistence.ControleProduto;
import view.InputProduto;
import entity.Produto;

public class Main {

    public static void main(String[] args) {

        Produto p = new Produto();
        InputProduto in = new InputProduto();

        //Entrada de dados
        p.setIdProduto( in.lerIdProduto() );
        p.setNome( in.lerNome() );
        p.setPreco( in.lerPreco() );

        System.out.println("\nDados do Produto:");
        System.out.println(p);
    }
}
```

- Resultado...

```
Console >
<terminated> Main [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe (10/05/2014 00:23:15)
Informe o Id do Produto...: 1
Informe o Nome do Produto.: Mouse
Informe o Preco do Produto: 30
|
Dados do Produto:
Produto [idProduto=1, nome=Mouse, preco=30.0]
```



## Tratamento de Exceções

Tem como objetivo realizar o tratamento de erros ocorridos em tempo de execução em um projeto Java de forma a tratá-los como exceções. É realizado através dos operadores try e catch.

Observe o exemplo do método de leitura do IdProduto contido na Classe de entrada de dados criada anteriormente...

```
public Integer lerIdProduto(){

    try{ //tentativa

        Scanner s = new Scanner(System.in);

        System.out.print("Informe o Id do Produto...: ");
        Integer idProduto = s.nextInt();
        //Lê o valor informado

        if(idProduto > 0){ //teste de validação
            return idProduto; //Ok
        }
        else{
            //redirecionar para o catch
            //-> forçar/lançar uma exceção
            throw new Exception("Valor deve ser
                                maior que zero");
        }
    }
    catch(Exception e){ //captura do erro / exceção

        System.out.println("Erro ao ler o id
                           do produto -> " + e.getMessage());

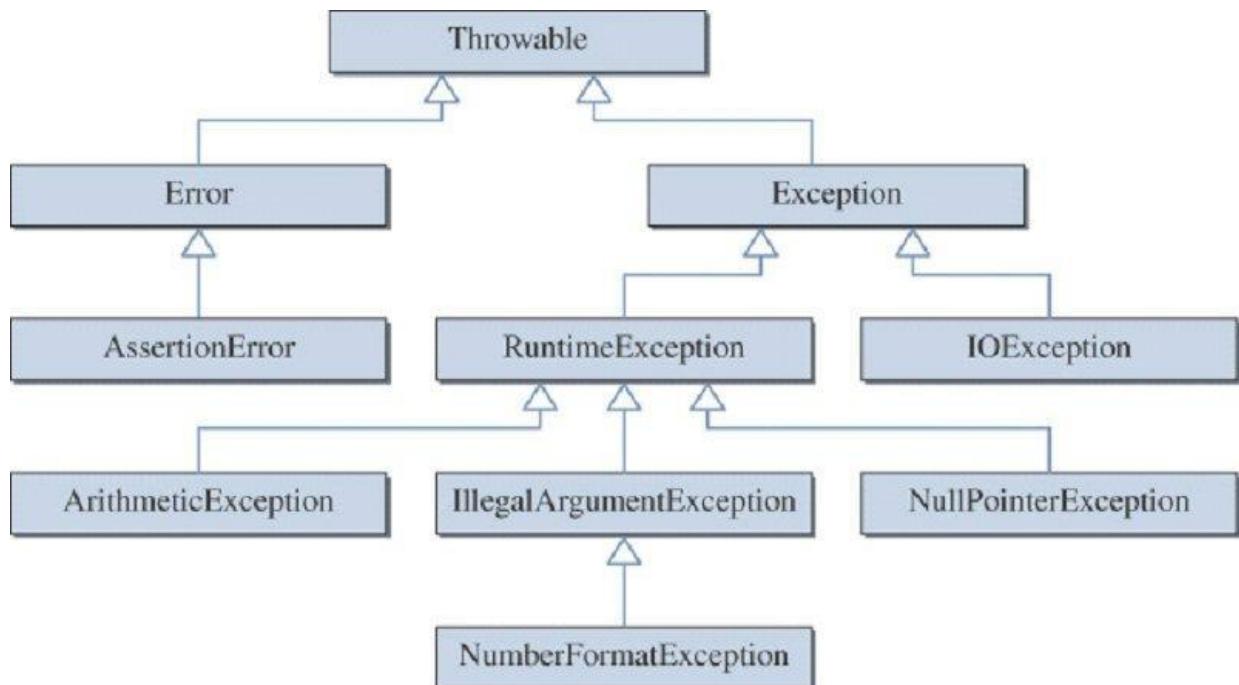
        return lerIdProduto(); //recursividade
    }
}
```

No que no bloco try iremos realizar a tentativa de leitura do valor inteiro através da Classe Scanner.

Caso a leitura produza erro, o fluxo do programa é automaticamente redirecionado para o catch, onde o erro é tratado através da Classe **Exception**.

A Classe **Exception** é uma das principais na hierarquia de exceções do Java.

## Hierarquia das principais Classes de Exceção em Java...



## throw new Exception

O Uso do operador **throw new** é feito para que um método possa forçar a execução de uma exceção. Por exemplo:

```
public Integer lerIdProduto(){

    try{ //tentativa

        Scanner s = new Scanner(System.in);
        System.out.print("Informe o Id do Produto...: ");
        Integer idProduto = s.nextInt();

        if(idProduto > 0){ //teste de validação
            return idProduto; //Ok
        }
        else{
            throw new Exception("Valor Inválido");
        }
    }
    catch(Exception e){
        System.out.println("Erro -> " + e.getMessage());
        return lerIdProduto();
    }
}
```



## Gravação de Arquivos em Java utilizando `FileWriter`

Abaixo, criaremos uma Classe para exemplificar a gravação dos dados de Produto em um arquivo txt utilizando tratamento de exceções e através da Classe **java.io.FileWriter**

```
package persistence;

import java.io.FileWriter;
import entity.Produto;

public class ControleProduto {

    //Método utilizado para gravar os dados de
    //um produto em arquivo txt
    public void gravarDados(Produto p) {

        try {
            //Classe -> FileWriter (Escrita de Arquivos)
            //true -> modo append -> adicionar conteúdo ao
            //final do arquivo
            //false -> sempre sobrescrever/substituir o
            //arquivo antigo

            //FileWriter lança (throws) IOException
            FileWriter fw = new FileWriter
                ("c:\\aula\\produtos.txt", true);

            fw.write("Dados -> " + p);
            //escrevendo no arquivo o toString da Classe
            fw.write("\n"); //quebra de linha
            fw.close(); //fechando o arquivo

            System.out.println("Dados gravados
                com sucesso.");
        } catch (Exception e) {
            System.out.println("Erro ao gravar
                arquivo: " + e.getMessage());
        }
    }

}

-----
FileWriter fw = new FileWriter("c:\\aula\\produtos.txt", true);
```

Esta linha cria um objeto `fw` para a Classe `StreamWriter`, e o caminho passado no seu construtor indica o caminho do arquivo que será gravado. O parâmetro `true` indica que a gravação sempre irá adicionar conteúdo ao arquivo, nunca sobrescrevê-lo.



Executando...

```
package main;

import persistence.ControleProduto;
import view.InputProduto;
import entity.Produto;

public class Main {

    public static void main(String[] args) {

        Produto p = new Produto();
        InputProduto in = new InputProduto();
ControleProduto c = new ControleProduto();

        //Entrada de dados
        p.setIdProduto( in.lerIdProduto() );
        p.setNome( in.lerNome() );
        p.setPreco( in.lerPreco() );

        System.out.println("\nDados do Produto:");
        System.out.println(p);
c.gravarDados(p);

    }
}
```

```
Console >
<terminated> Main [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe (10/05/2014 00:48:16)
Informe o Id do Produto...: 2
Informe o Nome do Produto.: Monitor
Informe o Preco do Produto: 250

Dados do Produto:
Produto [idProduto=2, nome=Monitor, preco=250.0]
Dados gravados com sucesso.
```

Arquivo gerado...

```
*C:\aula\produtos.txt - Notepad++
Arquivo Editar Localizar Visualizar Formatar Linguagem Configurações Macro Executar Plugins Janela ?
produtos.txt
1 Dados -> Produto [idProduto=1, nome=Mouse, preco=30.0]
2 Dados -> Produto [idProduto=2, nome=Monitor, preco=250.0]
3
4

Normal text file length : 114 lines : 4 Ln : 3 Col : 1 Sel : 0 | 0 UNIX ANSI as UTF-8 INS ..
```



## throws Exception

O uso de throws Exception na assinatura de um método obriga que este seja executado utilizando-se um try e catch. Por exemplo:

```
package persistence;

import java.io.FileWriter;
import entity.Produto;

public class ControleProduto {

    //Método utilizado para gravar os dados de
    //um produto em arquivo txt
    public void gravarDados(Produto p) {

        try {

            FileWriter fw = new FileWriter
                ("c:\\aula\\produtos.txt", true);

            fw.write("Dados -> " + p);
            fw.write("\n"); //quebra de linha

            fw.close(); //fechando o arquivo

            System.out.println("Dados gravados
                com sucesso.");
        } catch (Exception e) {
            System.out.println("Erro ao gravar
                arquivo: " + e.getMessage());
        }
    }

    public void gravarDados(Produto[] produtos)
        throws Exception{

        FileWriter fw = new FileWriter
            ("c:\\aula\\produtos.txt", true);

        for(int i = 0; i < produtos.length; i++){
            fw.write("Dados -> " + produtos[i]);
            fw.write("\n");
        }

        fw.close();
    }
}
```



### Executando...

Note que o uso do try e catch é necessário pois o método lança exceção através de throws Exception...

```
package main;

import persistence.ControleProduto;
import view.InputProduto;
import entity.Produto;

public class Main {

    public static void main(String[] args) {

        InputProduto in = new InputProduto();
        ControleProduto c = new ControleProduto();

        Produto[] produtos = new Produto[3];

        for(int i = 0; i < produtos.length; i++) {

            Produto p = new Produto();

            p.setIdProduto( in.lerIdProduto() );
            p.setNome( in.lerNome() );
            p.setPreco( in.lerPreco() );

            produtos[i] = p;
        }

        try {
            c.gravarDados(produtos);

            System.out.println("Dados gravados
                               com sucesso.");
        } catch (Exception e) {

            System.out.println("Erro -> "
                               + e.getMessage());
        }
    }
}
```



### Observação

Em caso de sobrescrita de métodos, se o método original da SuperClasse lança exceção (throws Exception), ao ser sobreescrito não precisa necessariamente lançar exceções também.

Por exemplo:

```
class A{
    public void calcular() throws Exception{
        System.out.println("Calculo A...");
    }
}

class B extends A{

    //Na sobreescrita você não é obrigado a manter
    //o throws Exception
    //Se o throws Exception for mantido ele nunca poderá ter
    //uma exceção maior do que a do metodo original

    @Override
    public void calcular(){
        System.out.println("Calculo B...");
    }
}
```

Podemos também sobreescrivar o método da superclasse utilizando uma exceção menor que a original, nunca o contrário.

```
class A{

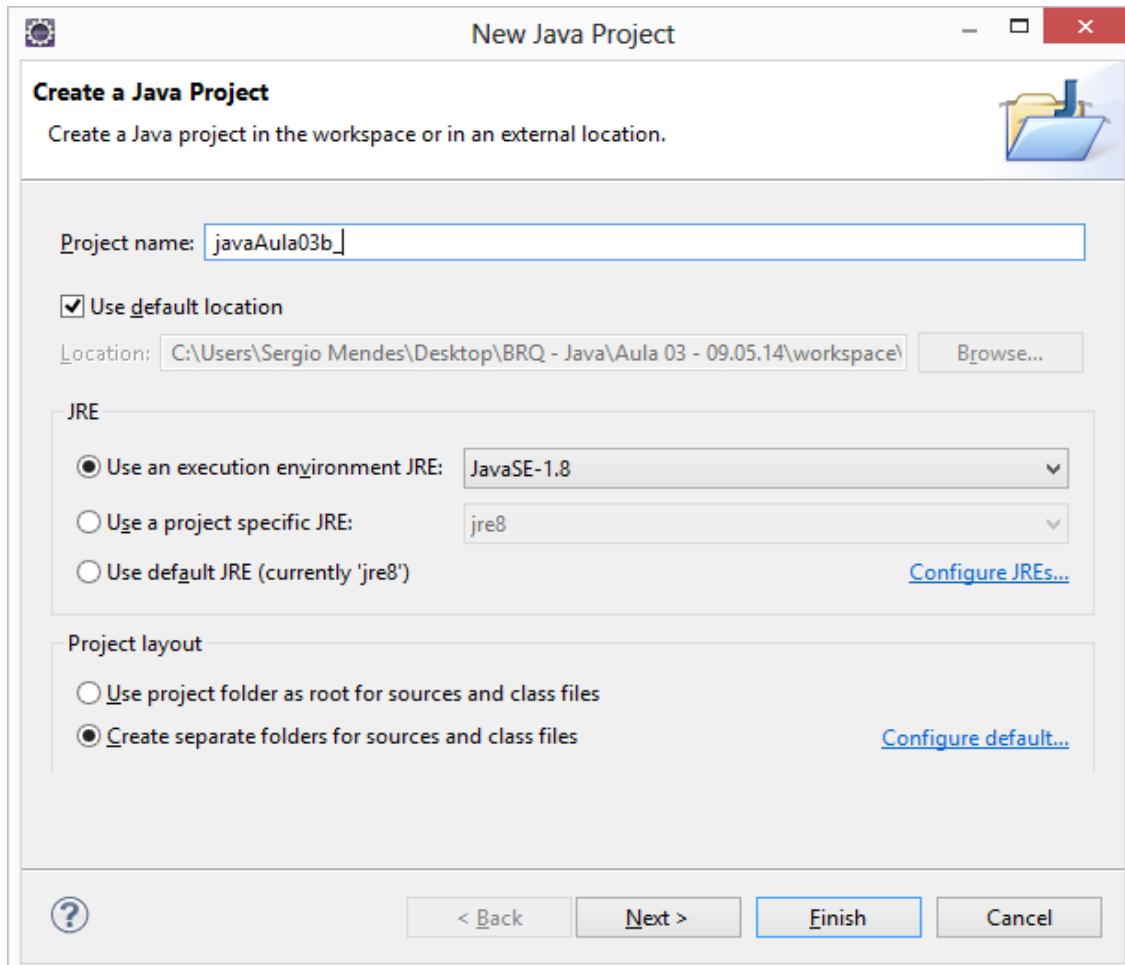
    public void calcular() throws Exception{
        System.out.println("Calculo A...");
    }
}

class B extends A{

    @Override //Sobreescrita
    public void calcular() throws RuntimeException{
        System.out.println("Calculo B...");
    }
}
```



Novo Projeto...



Criando a entidade Pessoa...

```
package entity;

public class Pessoa {

    private Integer idPessoa;
    private String nome;

    public Pessoa() {
    }

    public Pessoa(Integer idPessoa, String nome) {
        this.idPessoa = idPessoa;
        this.nome = nome;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }
}
```



```
public void setIdPessoa(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

@Override
public String toString() {
    return "Pessoa [idPessoa=" + idPessoa + ", nome="
           + nome + "]";
}

}
```

Criando uma Classe para leitura dos dados de Pessoa utilizando a biblioteca JOptionPane

### JOptionPane.showInputDialog

Utilizado para entrada de dados através de caixa de diálogo

### JOptionPane.showMessageDialog

Utilizado para impressão de mensagens através de janela de diálogo

```
package view;

import javax.swing.JOptionPane;

public class LeituraPessoa {

    public Integer lerIdPessoa() throws Exception{

        Integer idPessoa = new Integer(JOptionPane.
            showInputDialog("Informe o Id da Pessoa:"));

        if(idPessoa <= 0){
            throw new Exception("Id nao pode ser menor
                               ou igual a zero");
        }

        return idPessoa;
    }
}
```



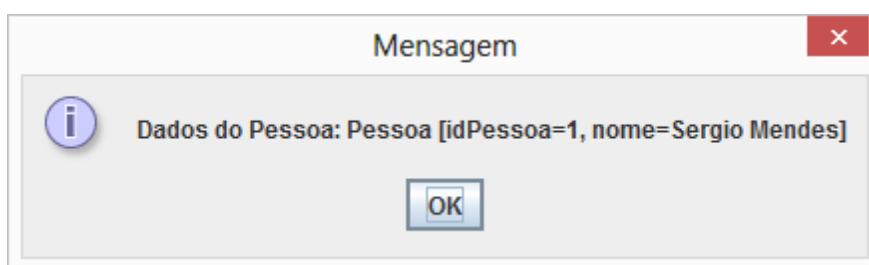
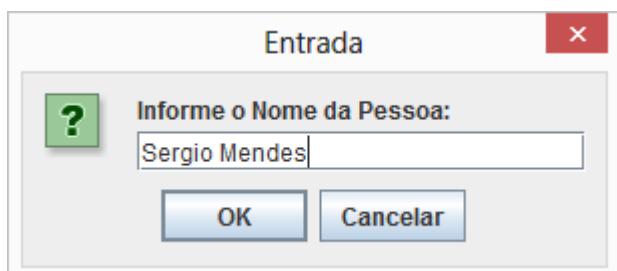
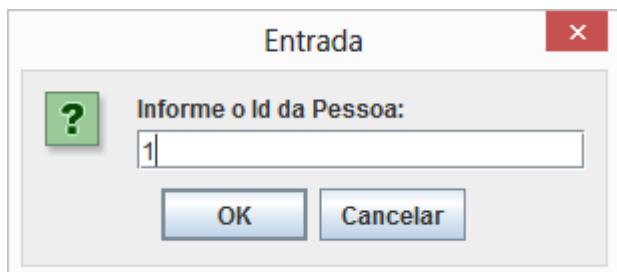
```
public String lerNome() throws Exception{  
  
    String nome = JOptionPane.showInputDialog  
        ("Informe o Nome da Pessoa:");  
  
    if(nome.length() <= 3){  
  
        throw new Exception("Nome de pessoa invalido");  
    }  
  
    return nome;  
}  
}
```

Executando através do método static void main...

```
package main;  
  
import javax.swing.JOptionPane;  
  
import view.LeituraPessoa;  
import entity.Pessoa;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Pessoa p = new Pessoa();  
        LeituraPessoa in = new LeituraPessoa();  
  
        try {  
  
            p.setIdPessoa( in.lerIdPessoa() );  
            p.setNome( in.lerNome() );  
  
            JOptionPane.showMessageDialog  
                (null, "Dados do Pessoa: " + p);  
  
        } catch (Exception e) {  
            JOptionPane.showMessageDialog  
                (null, "Erro -> " + e.getMessage());  
        }  
    }  
}
```



### Resultado...



### finally

Se utilizarmos um bloco finally após um bloco try e catch, todo e qualquer conteúdo colocado dentro do finally SEMPRE será executado após o try e catch.

### Exemplo...

```
package main;

import javax.swing.JOptionPane;

import view.LeituraPessoa;
import entity.Pessoa;

public class Main {

    public static void main(String[] args) {

        Pessoa p = new Pessoa();
        LeituraPessoa in = new LeituraPessoa();
```



```
try {  
  
    p.setIdPessoa( in.lerIdPessoa() );  
    p.setNome( in.lerNome() );  
  
    JOptionPane.showMessageDialog  
        (null, "Dados do Pessoa: " + p);  
  
} catch (Exception e) {  
    JOptionPane.showMessageDialog  
        (null, "Erro -> " + e.getMessage());  
}  
}  
  
finally{  
    JOptionPane.showMessageDialog  
        (null, "FIM DO PROGRAMA");  
}  
}  
}
```

Resultado...

The figure consists of four screenshots of Java JOptionPane dialogs:

- Entrada**: A dialog titled "Entrada" asking "Informe o Id da Pessoa:" with a text input field containing "1". It has "OK" and "Cancelar" buttons.
- Entrada**: A dialog titled "Entrada" asking "Informe o Nome da Pessoa:" with a text input field containing "Sergio Mendes". It has "OK" and "Cancelar" buttons.
- Mensagem**: A dialog titled "Mensagem" displaying "Dados do Pessoa: Pessoa [idPessoa=1, nome=Sergio Mendes]" with an "OK" button.
- Mensagem**: A dialog titled "Mensagem" displaying "FIM DO PROGRAMA" with an "OK" button.



O **finally** tem prioridade em vários tipos de execuções de um try e catch. Em resumo, se incluirmos um bloco finally após o try e catch, o finally SEMPRE será executado.

Exemplos...

```
class A{

    int Method1(){

        try{
            System.out.println("Dados gravados");

            return 1;
        }
        catch(Exception e){

            System.out.println("Erro");

            return 0;
        }
        finally{
            System.out.println("Conexao fechada");
        }

    }

    int Method2(int a, int b){

        try{
            return a / b;
        }
        catch(Exception e){

            return 0;
        }
        finally{
            System.out.println("Fim");
        }

    }

}
```



## final

O uso do operador final possui várias implicações. Por exemplo:

- Se qualificarmos uma **Classe** como **final**, esta não poderá ser extendida, ou seja, não poderá ser herdada...

```
public final class Cliente {  
}  
  
public class PessoaFisica extends Cliente{  
}
```

**Erro:** A Classe PessoaFísica não pode herdar Cliente, pois a Classe Cliente foi marcada como **final**.

Não podemos herdar Classes final

-----

- Se qualificarmos um **método** como **final**, este não poderá ser sobrescrito por nenhuma subclasse.

```
public class Cliente {  
    public final void imprimirDados(){  
    }  
  
    class PessoaFisica extends Cliente{  
        @Override //Sobrescrita  
        public void imprimirDados() {  
        }  
    }  
}
```

Não podemos sobreescrivernos métodos final.



- Se qualificarmos um **atributo** como **final**, este é entendido como uma constante, ou seja, seu valor inicial não poderá ser alterado.

```
class PessoaFisica{  
  
    //final -> quando utilizado em atributo faz com que o mesmo  
    //se torne uma constante, ou seja, seu valor não  
    //poderá ser alterado  
    private final int num = 10;  
  
    @Override //Sobrescrita  
    public void imprimirDados() {  
        num = 100;  
    }  
  
-----  
}
```

## Modificadores de visibilidade em Java:

**private** -> acesso permitido somente dentro da própria classe

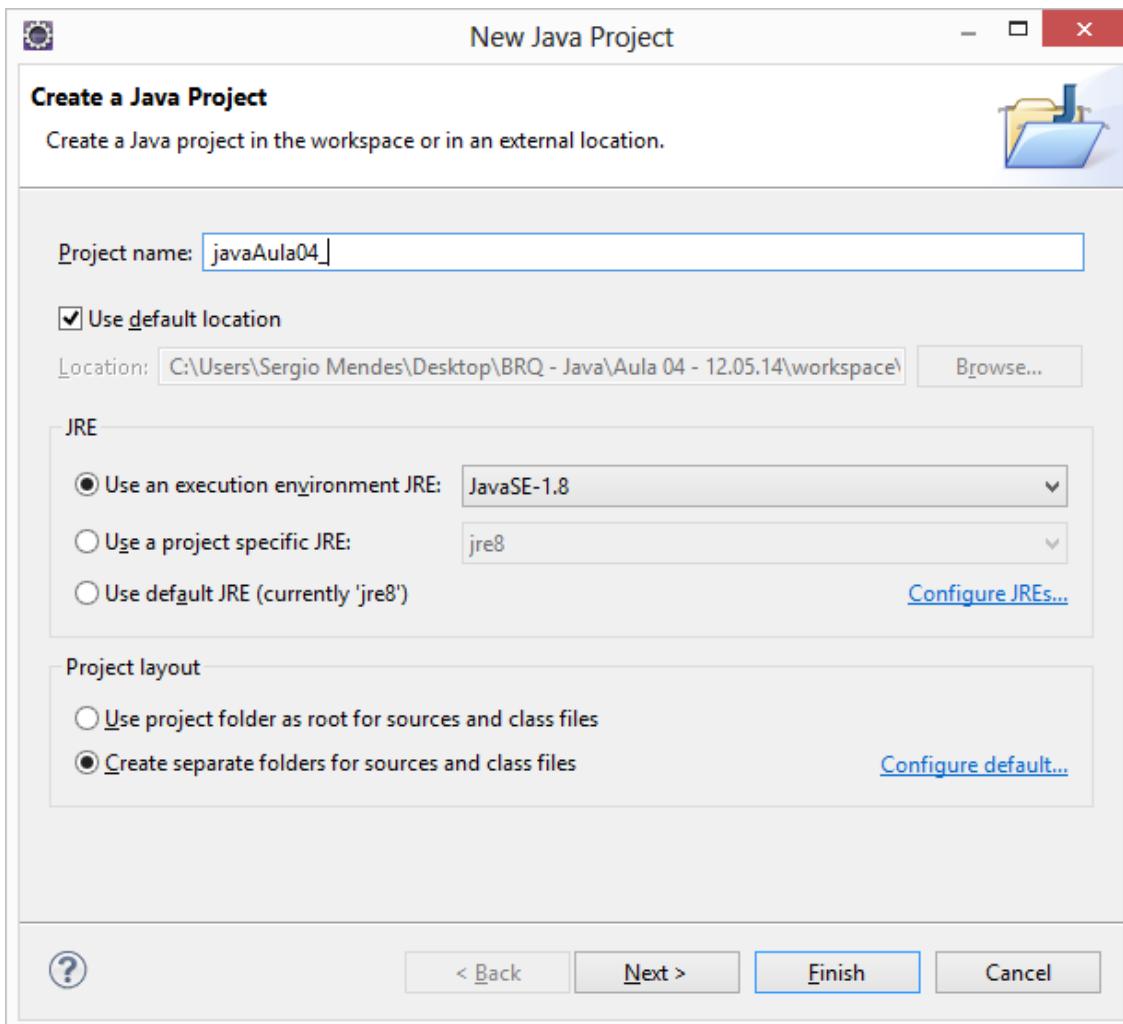
**default** -> (ausência de modificador) acesso somente dentro do pacote

**protected** -> somente por herança ou no mesmo pacote

**public** -> acesso total



### Criando o Projeto...



## Interfaces

As Interfaces são componentes da programação orientada a objetos que tem como objetivo isolar o ambiente de implementação do ambiente de execução, definindo um padrão para todas as Classes que a implementarem.

Quando um Classe implementa uma interface ela está comprometida a fornecer implementação para todos os métodos definidos na interface.

Criando uma interface para operações de arquivo...

```
package control;

public interface Arquivo {

    //Interfaces são consideradas níveis abstratos de programação,
    //tudo que se declara em uma interface ou é um protótipo ou
```



```
//uma constante...

//Regra 1: Métodos de uma interface não podem ter corpo, apenas
//assinatura (cabeçalho)

public void abrirArquivo() throws Exception;

public void gravarDados(String texto) throws Exception;

public void fecharArquivo() throws Exception;

}
```

Em Java, as interfaces são o tipo mais “abstrato” de programação.

Uma Interface possui apenas atributos constantes (cujo valor não pode ser modificado) e métodos sem corpo (apenas assinados)

Repare que os métodos da interface não possuem corpo, apenas assinatura.

Quando uma classe implementa uma interface ela é obrigada, por contrato, a implementar os métodos definidos na interface, caso isso não seja cumprido o programa irá gerar erro.

O relacionamento de uma Classe com uma interface também é do tipo **SER (É-UM)**

- Implementando a interface Arquivo para trabalhar com formato **txt**...

```
package control;

import java.io.FileWriter;

//implements -> relacionamento entre Classe e Interface
public class ArquivoTxt implements Arquivo {

    //Regra: Quando uma Classe implementa uma ou mais interfaces
    //esta Classe é obrigada a fornecer corpo para todos os métodos
    //abstratos da interface

    //Atributo
    private FileWriter arquivo;

    @Override //Sobrescrita
    public void abrirArquivo() throws Exception {

        //inicializando o arquivo em modo de escrita
        //true -> incluir conteúdo ao final do arquivo
    }
}
```



```
    arquivo = new FileWriter("c:\\aula\\\\exemplo.txt", true);  
}  
  
@Override //Sobrescrita  
public void gravarDados(String texto) throws Exception {  
    arquivo.write(texto);  
    //escrevendo o parametro texto dentro do arquivo  
    arquivo.write("\n"); //escrevendo uma quebra de linha  
}  
  
@Override //Sobrescrita  
public void fecharArquivo() throws Exception {  
    arquivo.close(); //fechando o arquivo  
}  
}
```

- Implementando a interface Arquivo para trabalhar com formato **xml**...

## XML - eXtensible Markup Language

Formato padrão para troca de dados entre aplicações web. Utiliza arvore de tags para marcação dos dados.

### Exemplo:

```
<?xml version="1.0" encoding="iso-8859-1" ?>  
  
<turma>  
  
    <aluno>  
        <codigo>1</codigo>  
        <nome>Fabrizzio Santos</nome>  
    </aluno>  
  
    <aluno>  
        <codigo>2</codigo>  
        <nome>Pedro Igor</nome>  
    </aluno>  
  
    <aluno>  
        <codigo>3</codigo>  
        <nome>Patricia Massaut</nome>  
    </aluno>  
  
</turma>
```



```
package control;

import java.io.FileWriter;

public class ArquivoXml implements Arquivo{

    //Atributo para escrita de arquivos (java.io)
    private FileWriter arquivo; //null

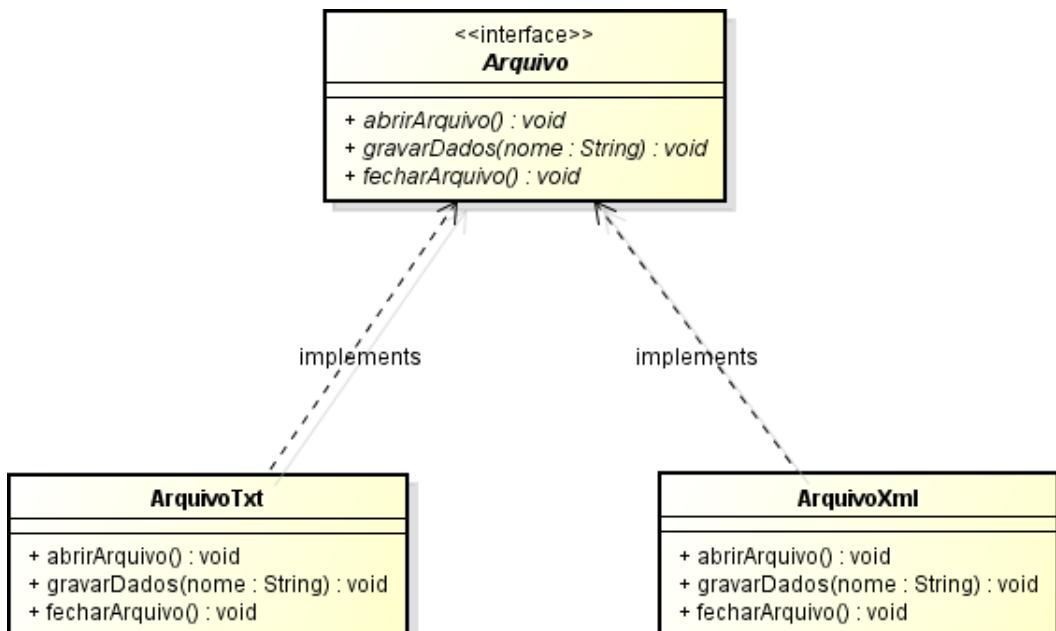
    @Override
    public void abrirArquivo() throws Exception {
        //inicializando e abrindo um arquivo para escrita
        arquivo = new FileWriter("c:\\aula\\exemplo.xml");
    }

    @Override
    public void gravarDados(String texto) throws Exception {

        arquivo.write("<?xml version='1.0' encoding='iso-8859-1' ?>");
        arquivo.write("<aula>");
            arquivo.write("<nome>" + texto + "</nome>");
        arquivo.write("</aula>");
    }

    @Override
    public void fecharArquivo() throws Exception {
        arquivo.close(); //fechando o arquivo
    }
}
```

Em UML:





- Classe para leitura...

```
package input;

import javax.swing.JOptionPane;

public class Leitura {

    //Método para ler o nome informado pelo usuario
    public String lerNome(){

        //exibe janela com o campo para preenchimento
        //e retorna o valor informado
        return JOptionPane.showInputDialog("Digite seu Nome:");
        //janela de entrada
    }

    //Método para ler o tipo de arquivo informado pelo usuario
    public String lerTipo(){

        //exibe janela com o campo para preenchimento
        //e retorna o valor informado
        return JOptionPane.showInputDialog("Digite o tipo de
            arquivo (xml ou txt):");
    }
}
```

## Polimorfismo

Significa "Muitas formas". Ocorre quando uma superclasse ou interface é instanciada através de suas subclasses de forma alterar o seu comportamento.

Por exemplo:

**Arquivo a = new ArquivoTxt();**  
[Interface] [Objeto] [Instância > Construtor da SubClasse]

- ou

**Arquivo a = new ArquivoXml();**  
[Interface] [Objeto] [Instância > Construtor da SubClasse]



# Java WebDeveloper - BRQ

Segunda-feira, 12 de Maio de 2014

Interfaces, Classes Abstratas, Polimorfismo, Relacionamento de Associação entre Classes.

Aula  
04

```
package main;

import javax.swing.JOptionPane;

import input.Leitura;
import control.Arquivo;
import control.ArquivoTxt;
import control.ArquivoXml;

public class Main {

    public static void main(String[] args) {

        Leitura in = new Leitura();
        //instanciando a Classe de leitura

        //new -> executa o construtor para
        //definir espaço de memória

        //null -> sem espaço de memória (vazio),
        //não tem ponteiro

Arquivo a = null;

        //lendo o tipo de arquivo informado pelo usuario
        String tipo = in.lerTipo();

        if(tipo.equalsIgnoreCase("txt")){
            a = new ArquivoTxt(); //polimorfismo
        }
        else if(tipo.equalsIgnoreCase("xml")){
            a = new ArquivoXml(); //polimorfismo
        }

        try{

            a.abrirArquivo();
            a.gravarDados( in.lerNome() );
            a.fecharArquivo();

            JOptionPane.showMessageDialog
                (null, "Dados gravados.");
        }
        catch(Exception e){
            JOptionPane.showMessageDialog
                (null, "Erro: " + e.getMessage());
        }
    }
}
```



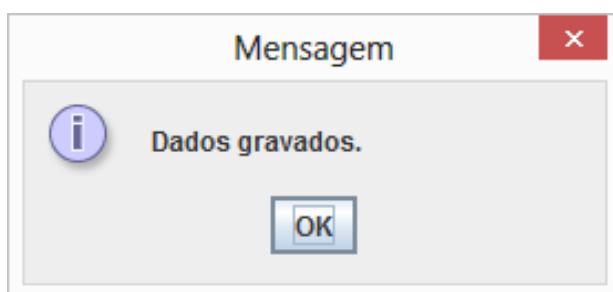
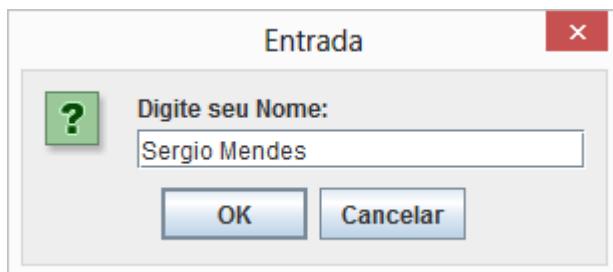
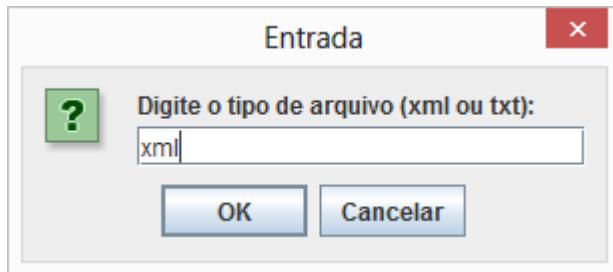
# Java WebDeveloper - BRQ

Segunda-feira, 12 de Maio de 2014

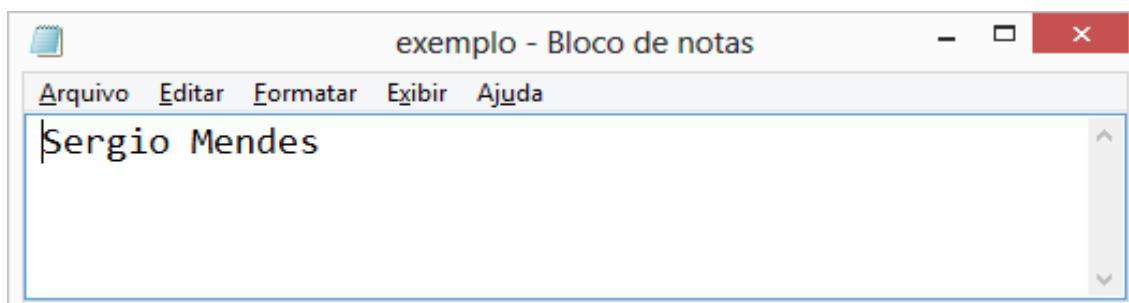
Interfaces, Classes Abstratas, Polimorfismo, Relacionamento de Associação entre Classes.

Aula  
04

Executando...



Arquivos gerados...





Demais regras sobre interfaces:

- Uma Classe pode implementar uma ou mais interfaces:  
Exemplo:

```
interface A{
    void metodoA();
}

interface B{
    void metodoB();
}

class C implements A, B{

    @Override
    public void metodoB() {
    }

    @Override
    public void metodoA() {
    }
}
```

- Uma Interface pode herdar 1 ou muitas interfaces, ou seja, podemos realizar em Java herança múltipla entre interfaces:  
Exemplo:

```
interface SerVivo{
    void respirar();
}

interface Mamifero extends SerVivo{
    void locomover();
}

interface Ave extends SerVivo{
    void voar();
}

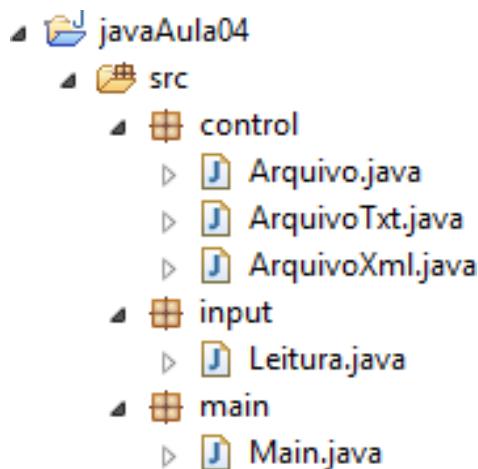
class Cachorro implements Mamifero{

    @Override
    public void respirar() {
    }

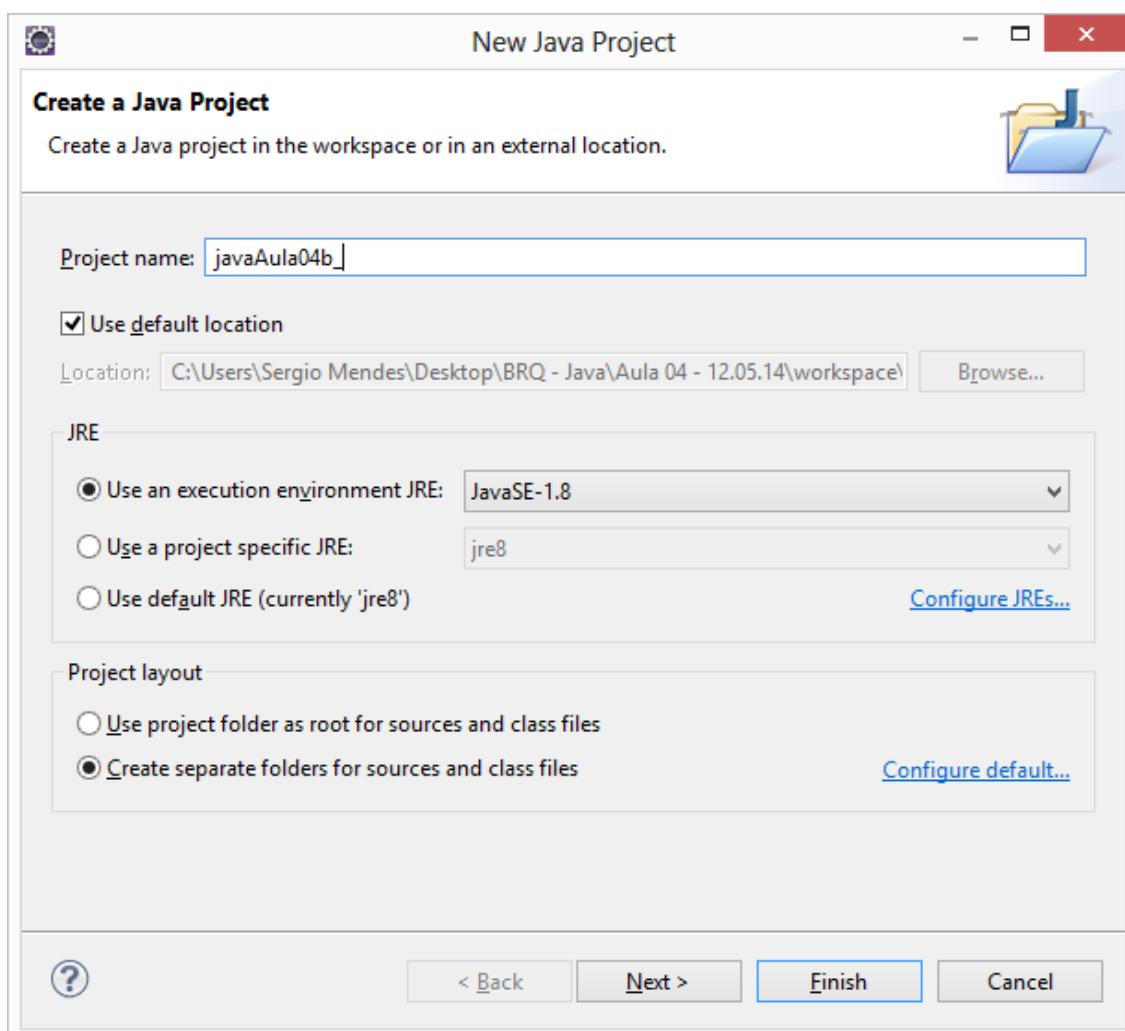
    @Override
    public void locomover() {
    }
}
```



### Estrutura do Projeto...



### Novo Projeto...





## Classes Abstratas

Uma Classe abstrata consiste de uma Classe Java padrão que pode ter também métodos abstratos, ou seja, métodos sem corpo e que deverão ser implementados por outras classes concretas que herdarem a Classe abstrata.

Exemplo:

```
package entity;

public abstract class FormaGeometrica {

    private float base;          //numero real
    private float altura; //numero real

    public FormaGeometrica() {
        // Construtor default
    }

    public float getBase() {
        return base;
    }

    public void setBase(float base) {
        this.base = base;
    }

    public float getAltura() {
        return altura;
    }

    public void setAltura(float altura) {
        this.altura = altura;
    }

    //Uma Classe abstrata pode ter também abstratos
    //ou seja, métodos que deverão ser implementados por outras
    //classes que herdarem a Classe abstrata

    public abstract float getArea();

    //Método abstrato é todo aquele que não possui corpo e
    //deverá ser implementado pelas subclasses
    //Um método abstrato só pode ser declarado dentro de uma
    //interface ou de uma classe abstrata
}
```



Herdando e implementando a Classe Abstrata...

```
package entity;

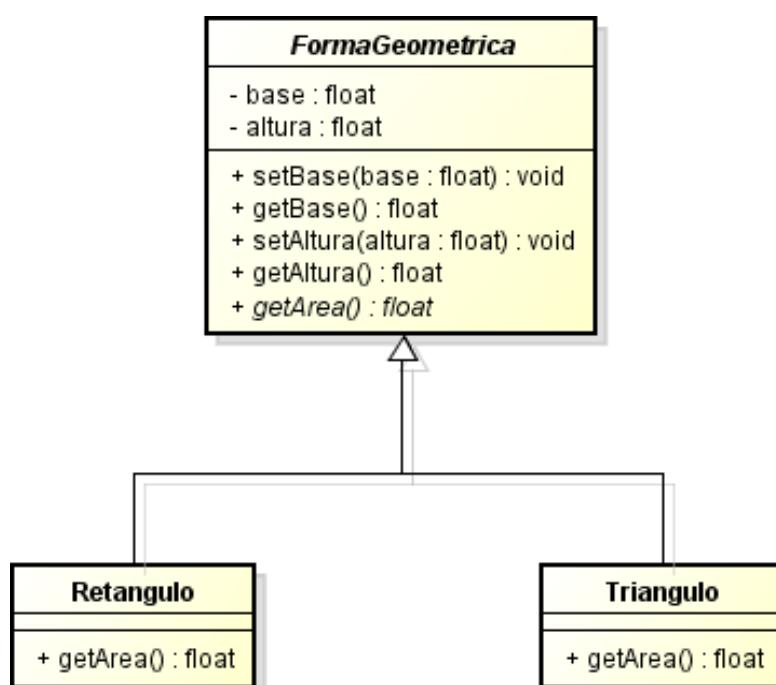
public class Retangulo extends FormaGeometrica{
    /*
     * A Classe Retangulo obrigatoriamente deve fornecer corpo
     * para o método getArea declarado na Classe Abstrata, pois
     * o método foi declarado como abstract
     */
    @Override
    public float getArea() {
        return getBase() * getAltura();
    }
}
```

```
package entity;

public class Triangulo extends FormaGeometrica{

    @Override
    public float getArea() {
        return ( getBase() * getAltura() ) / 2;
    }
}
```

Em UML:





### Realizando polimorfismo da Classe abstrata...

```
package main;

import entity.FormaGeometrica;
import entity.Retangulo;
import entity.Triangulo;

public class Main {

    public static void main(String[] args) {

        //Não é boa prática inicializar uma Classe abstrata
        //através de seu construtor...
        FormaGeometrica f = new Triangulo();
        //tipo      objeto      instancia -> comportamento

        f.setBase(10.0f);
        f.setAltura(5.0f);

        System.out.println("Base    -> " + f.getBase());
        System.out.println("Altura -> " + f.getAltura());
        System.out.println("Area do objeto: " + f.getArea());

        if(f instanceof Retangulo){
            System.out.println
                ("Calculo de Area de Retangulo");
        }
        else if(f instanceof Triangulo){
            System.out.println
                ("Calculo de Area de Triangulo");
        }
    }
}
```

### Executando...

The screenshot shows the Eclipse IDE interface with the following details:

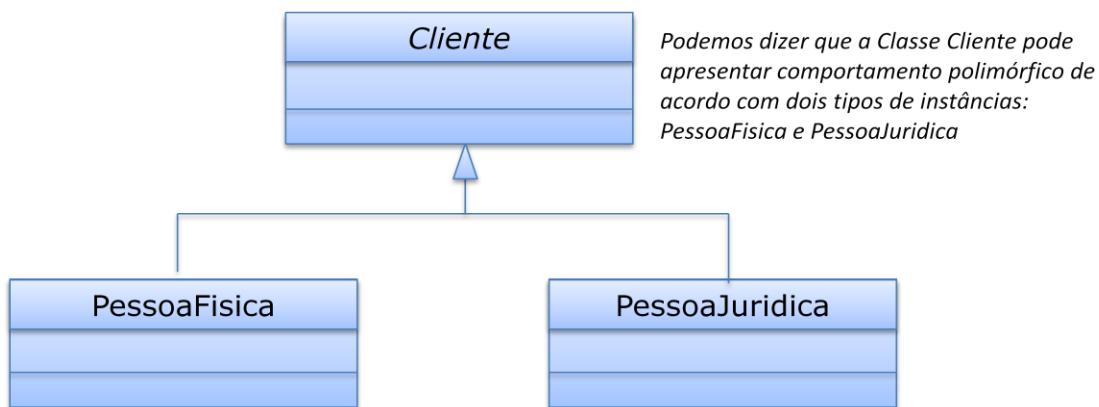
- Project Explorer:** Shows the project structure with packages `entity`, `main`, and files `FormaGeometrica.java`, `Retangulo.java`, and `Triangulo.java`.
- Code Editor:** Displays the `Main.java` file with the provided Java code.
- Console:** Shows the output of the program's execution:

```
Base    -> 10.0
Altura -> 5.0
Area do objeto: 25.0
Calculo de Area de Triangulo
```



### Importante:

Todo objeto que possa passar em mais de um teste É-UM pode ser considerado polimórfico, ou seja, referências a Classes mais genéricas terão seu comportamento definidos através de instâncias de Classes mais específicas.



Regras entre Interfaces, Classes Abstratas e Classes Concretas...

	Interface	Classe Abstrata	Classe
<b>Atributos</b>	Somente Constantes	Sim	Sim
<b>Construtores</b>	Não	Sim	Sim
<b>Métodos</b>	Não	Sim	Sim
<b>Métodos Abstratos</b>	Sim	Sim	Não
<b>Pode ser instanciado</b>	Não	Não	Sim

### Estrutura do Projeto...

- **javaAula04b**
  - **src**
    - **entity**
      - ▷ FormaGeometrica.java
      - ▷ Retangulo.java
      - ▷ Triangulo.java
    - **main**
      - ▷ Main.java



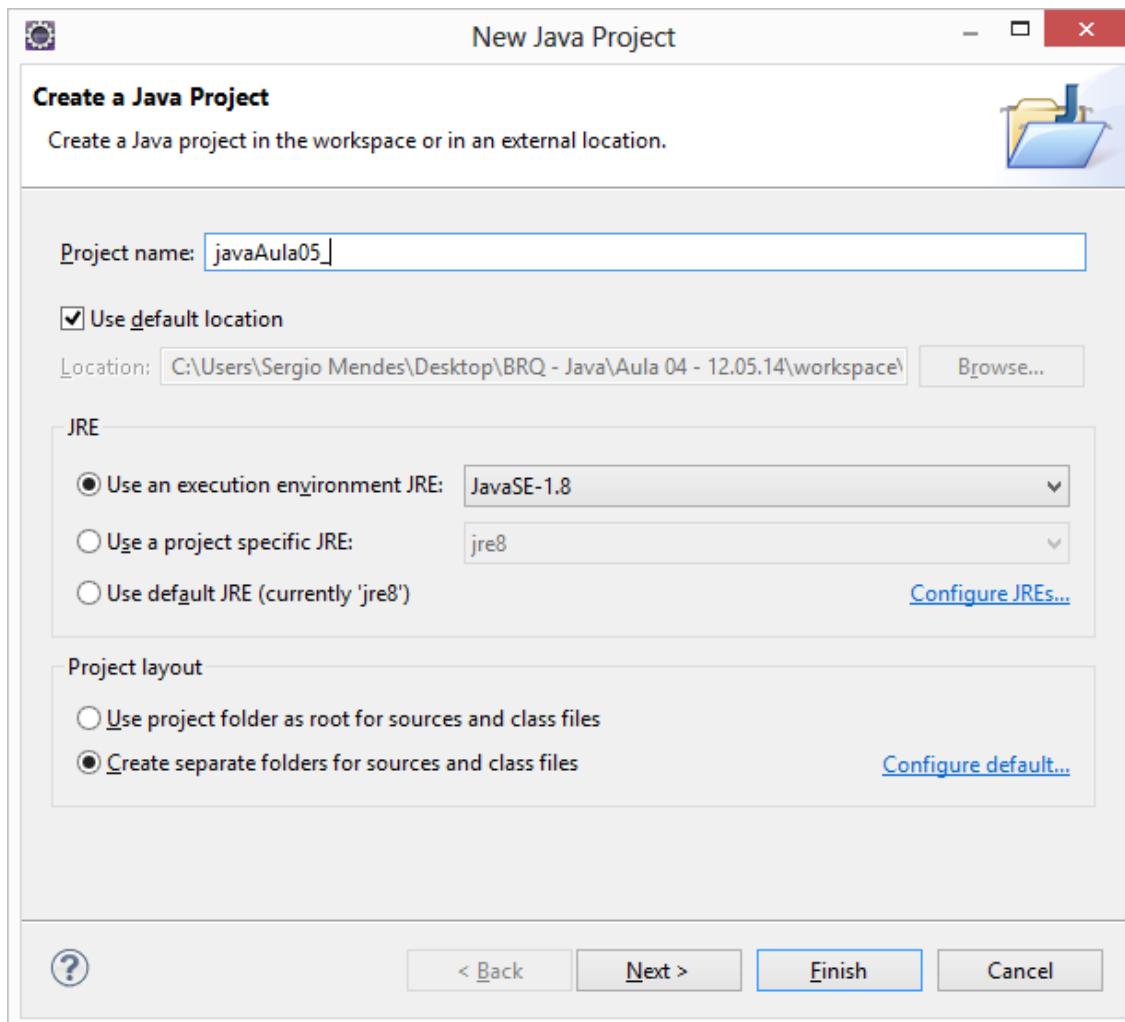
# Java WebDeveloper - BRQ

Segunda-feira, 12 de Maio de 2014

Interfaces, Classes Abstratas, Polimorfismo, Relacionamento de Associação entre Classes.

Aula  
04

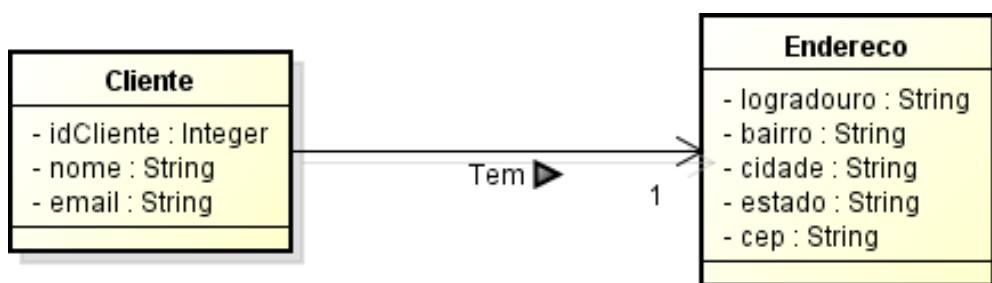
Novo Projeto...



## Relacionamento de Associação (TER)

Utilizado para relacionamentos de objetos de classes distintas. A naveabilidade é representada através de uma seta nas extremidades, pois representa o sentido em que as informações são disparadas.

Exemplo:





```
package entity;

public class Cliente {

    private Integer idCliente;
    private String nome;
    private String email;
    private Endereco endereço; // Associação (TER)

    public Cliente() {
        // Construtor default
    }

    public Cliente(Integer idCliente, String nome, String email) {
        super();
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
    }

    @Override
    public String toString() {
        return idCliente + ", " + nome + ", " + email;
    }

    public Integer getIdCliente() {
        return idCliente;
    }

    public void setIdCliente(Integer idCliente) {
        this.idCliente = idCliente;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Endereco getEndereco() {
        return endereço;
    }
}
```



```
public void setEndereco(Endereco endereco) {
    this.endereco = endereco;
}

package entity;

public class Endereco {

    private String logradouro;
    private String bairro;
    private String cidade;
    private String estado;
    private String cep;

    public Endereco() {
        // Construtor default
    }

    public Endereco(String logradouro, String bairro, String cidade,
                    String estado, String cep) {
        this.logradouro = logradouro;
        this.bairro = bairro;
        this.cidade = cidade;
        this.estado = estado;
        this.cep = cep;
    }

    @Override
    public String toString() {
        return logradouro + ", " + bairro + ", "
               + cidade + ", " + estado + ", " + cep;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getBairro() {
        return bairro;
    }

    public void setBairro(String bairro) {
        this.bairro = bairro;
    }
}
```



# Java WebDeveloper - BRQ

Segunda-feira, 12 de Maio de 2014

Interfaces, Classes Abstratas, Polimorfismo, Relacionamento de Associação entre Classes.

Aula  
04

```
public String getCidade() {
    return cidade;
}

public void setCidade(String cidade) {
    this.cidade = cidade;
}

public String getEstado() {
    return estado;
}

public void setEstado(String estado) {
    this.estado = estado;
}

public String getCep() {
    return cep;
}

public void setCep(String cep) {
    this.cep = cep;
}
}
```

Testando...

```
package main;

import entity.Cliente;
import entity.Endereco;

public class Main {

    public static void main(String[] args) {

        Cliente c = new Cliente
            (1, "Sergio", "sergio@gmail.com");

        Endereco e = new Endereco
            ("Rua A", "Centro", "Rio de Janeiro", "RJ",
            "25000-000");

        //Relacionar Cliente a Endereco
        c.setEndereco(e); //TER -> Cliente tem Endereco

        //c.setEndereco(new Endereco
        //    ("Rua A", "Centro", "Rio de Janeiro", "RJ",
        //    "25000-000"));
    }
}
```



# Java WebDeveloper - BRQ

Segunda-feira, 12 de Maio de 2014

Interfaces, Classes Abstratas, Polimorfismo, Relacionamento de Associação entre Classes.

Aula  
04

```
//Imprimindo
System.out.println("Cliente -> " + c);
System.out.println("Endereco -> " + c.getEndereco());
}

}
```

Saída:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with packages `javaAula04`, `javaAula04c`, and `src`. Under `src`, there are `entity` (containing `Cliente.java` and `Endereco.java`) and `main` (containing `Main.java`).
- Code Editor:** Displays the `Main.java` file with the following code:

```
package main;
import entity.Cliente;
public class Main {
    public static void main(String[] args) {
        Cliente c = new Cliente(1, "Sergio");
        Endereco e = new Endereco("Rua A", "Centro", "Rio de Janeiro", "RJ", "25000-000");
        c.setEndereco(e); //TER -> Cliente tem Endereco
        //c.setEndereco(new Endereco("Rua A", "Centro", "Rio de Janeiro", "RJ", "25000-000"));
        //Imprimindo
        System.out.println("Cliente -> " + c);
        System.out.println("Endereco -> " + c.getEndereco());
    }
}
```
- Console:** Shows the output of the executed code:

```
Cliente -> 1, Sergio, sergio@gmail.com
Endereco -> Rua A, Centro, Rio de Janeiro, RJ, 25000-000
```

Cliente -> 1, Sergio, sergio@gmail.com  
Endereco -> Rua A, Centro, Rio de Janeiro, RJ, 25000-000



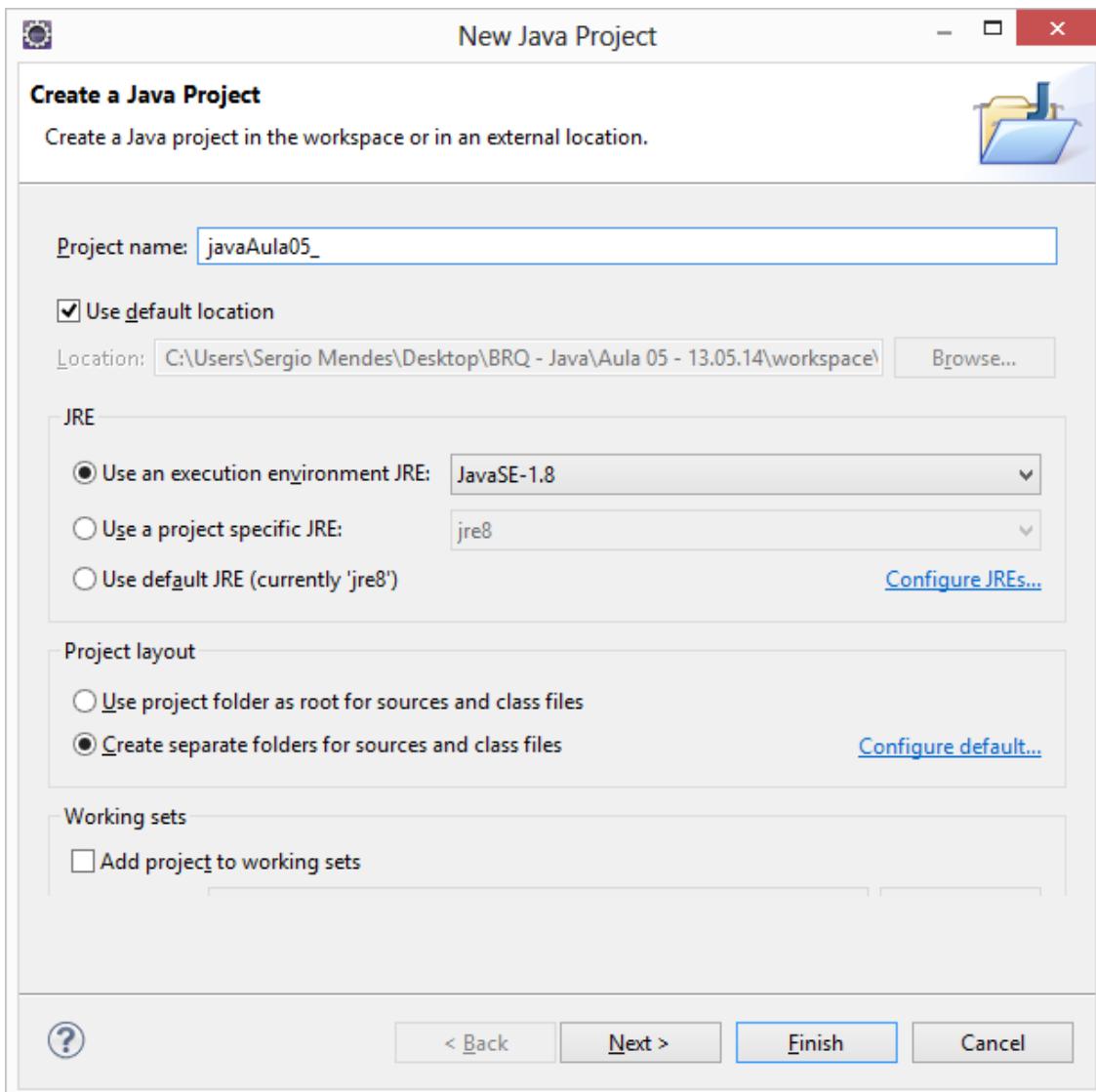
# Java WebDeveloper - BRQ

Terça-feira, 13 de Maio de 2014

Vetores, API Collections: List, ArrayList, Set, HashSet, LinkedHashSet, TreeSet, LinkedHashMap, Comparable.

Aula  
05

Criando o Projeto...



Criando um JavaBean Produto...

```
package entity;

//JavaBean -> Classe java de entidade (modelagem)
public class Produto{

    private Integer idProduto;
    private String nome;
    private Double preco;

    public Produto() {
        // Construtor default (padrão)
    }
}
```



```
// Sobrecarga de Construtores
public Produto(Integer idProduto, String nome, Double preco) {
    super();
    this.idProduto = idProduto;
    this.nome = nome;
    this.preco = preco;
}

public Integer getIdProduto() {
    return idProduto;
}

public void setIdProduto(Integer idProduto) {
    this.idProduto = idProduto;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

@Override
public String toString() {
    return idProduto + ", " + nome + ", " + preco;
}

@Override
public boolean equals(Object obj) {

    if(obj instanceof Produto){

        Produto p = (Produto) obj;

        if(p.getIdProduto().equals(idProduto)){
            return true;
        }
    }

    return false;
}
```



```
@Override
public int hashCode() {
    return idProduto.hashCode();
}
```

## Observação...

Todo método em Java que recebe como parâmetro Object pode ser executado com passagem de qualquer tipo de dado em Java.

Caso queiramos converter (casting) o valor recebido como Object para um tipo específico, é aconselhável que testemos primeiro este valor através do operador **instanceof**

Exemplo:

```
class Teste{

    public void imprimir(Object obj){

        if(obj instanceof String){

            String texto = (String) obj;

            System.out.println(texto.toUpperCase());
        }
        else{
            System.out.println("Erro");
        }
    }
}
```

Executando...

```
public static void main(String[] args) {

    Teste t = new Teste();

    t.imprimir("Sergio");
}
```



## Utilizando vetores em Java...

```
package main;

import entity.Produto;

public class Main1 {

    public static void main(String[] args) {

        //Vetor -> arranjos de dados
        Produto[] vetor = new Produto[4]; // |0|1|2|3|

        vetor[0] = new Produto(1, "Mouse", 30.0);
        vetor[1] = new Produto(2, "Celular", 200.0);
        vetor[2] = new Produto(3, "PenDrive", 80.0);
        vetor[3] = new Produto(4, "Monitor", 500.0);

        int i = 0; //contador
        while(i < vetor.length){ //enquanto

            //Capturar o Produto do vetor pela
            //posição da variável [i]
            Produto p = vetor[i];

            System.out.println("IdProduto.....: "
                + p.getIdProduto());

            System.out.println("Nome.....: "
                + p.getNome());

            System.out.println("Preço.....: "
                + p.getPreco());

            System.out.println("\n");

            i++; //incremento
        }

        for(int j = 0; j < vetor.length; j++){

            Produto p = vetor[j];
            System.out.println("Produto -> " + p); //toString
        }

        //foreach
        for(Produto p : vetor){
            System.out.println("Produto -> " + p);
        }
    }
}
```



Executando...

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows a project named "javaAula05" with packages "src" and "entity". Inside "entity" is a file "Produto.java".
- Code Editor:** Displays Main1.java with the following code:

```
int i = 0; //contador
while(i < vetor.length){ //enquanto
    //Captura o Produto do vetor
    Produto p = vetor[i];

    System.out.println("IdProduto....: " + IdProduto....);
    System.out.println("Nome.....: " + Nome....);
    System.out.println("Preco.....: " + Preco....);
    System.out.println("\n");

    i++; //incremento
}

for(int j = 0; j < vetor.length; j++){
    Produto p = vetor[j];
    System.out.println("Produto -> " + Produto -> );
}

//foreach
for(Produto p : vetor){
    System.out.println("Produto -> " + Produto -> );
}
```
- Console:** Shows the output of the program:

```
<terminated> Main1 [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe (13/05/2014 22:21:36)
IdProduto....: 1
Nome.....: Mouse
Preco.....: 30.0

IdProduto....: 2
Nome.....: Celular
Preco.....: 200.0

IdProduto....: 3
Nome.....: PenDrive
Preco.....: 80.0

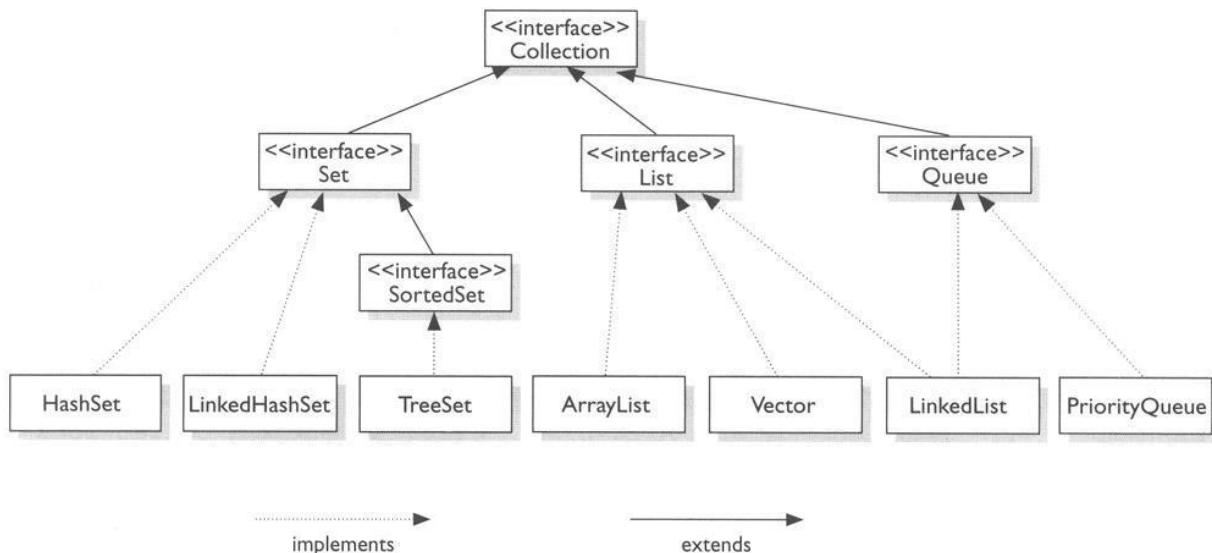
IdProduto....: 4
Nome.....: Monitor
Preco.....: 500.0

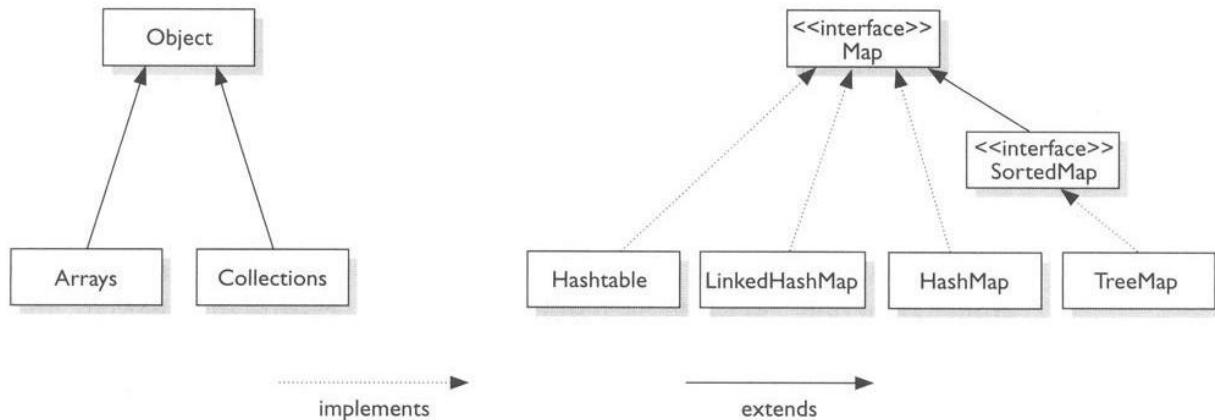
Produto -> 1, Mouse, 30.0
Produto -> 2, Celular, 200.0
Produto -> 3, PenDrive, 80.0
Produto -> 4, Monitor, 500.0
Produto -> 1, Mouse, 30.0
Produto -> 2, Celular, 200.0
```

## Collections em Java

Coleções são objetos (Classes ou Interfaces) através do qual podemos agrupar vários elementos, seguindo comportamentos que simulam o funcionamento de filas, pilhas, conjuntos, mapas, etc...

A API Collections do Java é formado pelas seguintes Interfaces e Classes do pacote `java.util`:





## Manipulando Listas...

Listas em Java são seqüência utilizadas para coleções de dados que:

- Podem conter elementos duplicados
- Possuem suporte a ordenação
- Possuem endereçamento por posição / índice

Exemplo:

```

List<Produto> lista = new ArrayList<Produto>();

package main;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import entity.Produto;

public class Main2 {

    public static void main(String[] args) {

        //criando uma lista de produto vazia
        //List      -> Interface
        //ArrayList -> Classe que implementa a interface List
        List<Produto> lista = new ArrayList<Produto>(); //vazia

        Produto p1 = new Produto(1, "Caneta", 10.0);
        Produto p2 = new Produto(2, "Borracha", 3.0);
        Produto p3 = new Produto(3, "Lapis", 1.0);

        lista.add(p1);
        lista.add(p2);
        lista.add(p3);
    }
}
  
```



```
System.out.println("Qtd de Produtos na lista: "
+ lista.size());

for(Produto p : lista){
    System.out.println("Produto -> " + p);
}

System.out.println("Produto na posição [0] -> "
+ lista.get(0));

System.out.println("Produto na posição [1] -> "
+ lista.get(1));

System.out.println("Produto na posição [2] -> "
+ lista.get(2));
}
}
```

Executando...

```
Qtd de Produtos na lista: 3
Produto -> 1, Caneta, 10.0
Produto -> 2, Borracha, 3.0
Produto -> 3, Lapis, 1.0
Produto na posição [0] -> 1, Caneta, 10.0
Produto na posição [1] -> 2, Borracha, 3.0
Produto na posição [2] -> 3, Lapis, 1.0
```

## Interface Comparable

A Interface Comparable é utilizada em Java para permitir que uma determinada Classe Javabean tenha suporte a ordenação.

Vários componentes da API do Java realizam ordenação e classificação somente de classes que implementam a interface Comparable.

Quando uma Classe implementa a interface Comparable, esta é obrigada a implementar o método **compareTo**

Implementando a interface Comparable no JavaBean Produto...

```
package entity;

//JavaBean -> Classe java de entidade (modelagem)
public class Produto implements Comparable<Produto>{

    private Integer idProduto;
    private String nome;
    private Double preco;
```



```
public Produto() {
    // Construtor default (padrão)
}

// Sobrecarga de Construtores
public Produto(Integer idProduto, String nome, Double preco) {
    super();
    this.idProduto = idProduto;
    this.nome = nome;
    this.preco = preco;
}

public Integer getIdProduto() {
    return idProduto;
}

public void setIdProduto(Integer idProduto) {
    this.idProduto = idProduto;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

@Override
public String toString() {
    return idProduto + ", " + nome + ", " + preco;
}

@Override
public boolean equals(Object obj) {
    if(obj instanceof Produto){
        Produto p = (Produto) obj;
        if(p.getIdProduto().equals(idProduto)){
            return true;
        }
    }
    return false;
}
```



```
@Override
public int hashCode() {
    return idProduto.hashCode();
}

@Override
public int compareTo(Produto p) {
    return nome.compareTo(p.getNome());
    //ordem crescente
    //return p.getNome().compareTo(nome); //ordem decrescente
}
}
```

Ordenando a lista de Produtos baseado na regra definida pelo método **compareTo...**

```
package main;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import entity.Produto;

public class Main2 {

    public static void main(String[] args) {

        //criando uma lista de produto vazia
        //List     -> Interface
        //ArrayList -> Classe que implementa a interface List
        List<Produto> lista = new ArrayList<Produto>(); //vazia

        Produto p1 = new Produto(1, "Caneta", 10.0);
        Produto p2 = new Produto(2, "Borracha", 3.0);
        Produto p3 = new Produto(3, "Lapis", 1.0);

        lista.add(p1);
        lista.add(p2);
        lista.add(p3);

        System.out.println("Qtd de Produtos na lista: "
            + lista.size());

        //Para que o Java possa ordenar a lista de produtos,
        //é necessário que criemos na
        //classe Produto uma regra de ordenação.
    }
}
```



```
//Basicamente, em Java, temos 2 maneiras de ordenar
//objetos: hashCode ou compareTo

//Para algumas bibliotecas do Java como Collections, a
//regra de ordenação dos objetos
//deveria ser feita por meio de uma interface chamada
//Comparable e de um método
//chamado compareTo

Collections.sort(lista); //ordenação
//Collections.reverse(lista); //inverte a ordem da lista

for(Produto p : lista){
    System.out.println("Produto -> " + p);
}

System.out.println("Produto na posição [0] -> "
+ lista.get(0));

System.out.println("Produto na posição [1] -> "
+ lista.get(1));

System.out.println("Produto na posição [2] -> "
+ lista.get(2));

System.out.println("Primeiro elemento da lista -> "
+ Collections.min(lista));

System.out.println("Último elemento da lista -> "
+ Collections.max(lista));
}

}
```

Executando...

Qtd de Produtos na lista: 3

Produto -> 2, Borracha, 3.0  
Produto -> 1, Caneta, 10.0  
Produto -> 3, Lapis, 1.0

Produto na posição [0] -> 2, Borracha, 3.0  
Produto na posição [1] -> 1, Caneta, 10.0  
Produto na posição [2] -> 3, Lapis, 1.0

Primeiro elemento da lista -> 2, Borracha, 3.0  
Último elemento da lista -> 3, Lapis, 1.0



## A Collection Set

A Interface Set é utilizada para definir conjuntos que não permitem elementos duplicados.

Pode ser implementada como:

¶ **HashSet**

Retorna os elementos do Set na ordem definida pela regra do método hashCode contida no JavaBean para o qual o Set foi definido.  
Sua regra para descartar objetos iguais é através do método equals

¶ **LinkedHashSet**

Retorna os elementos do Set na ordem que foram inseridos na coleção, ou seja, mantém a mesma ordem de entrada dos elementos.  
Sua regra para descartar objetos iguais é através do método equals

¶ **TreeSet**

Retorna os elementos do Set na ordem definida pelo método compareTo. Sua regra para descartar objetos iguais também é pelo método compareTo.

Exemplo...

```
Set<Produto> produtos = new HashSet<Produto>();  
ou  
Set<Produto> produtos = new LinkedHashSet<Produto>();  
ou  
Set<Produto> produtos = new TreeSet<Produto>();
```

```
package main;  
  
import java.util.Set;  
import java.util.TreeSet;  
  
import entity.Produto;  
  
public class Main3 {  
  
    public static void main(String[] args) {  
  
        //Set -> Não permite objetos duplicados, usa como regra de  
        //comparação de igualdade o método [equals], exceto quando  
        //utilizado como TreeSet  
        //descarta objetos iguais ao que já existir no Set  
  
        //LinkedHashSet -> retorna os dados na ordem de entrada  
        //HashSet           -> retorna os dados na ordem baseado no  
        //hashCode
```



# Java WebDeveloper - BRQ

Terça-feira, 13 de Maio de 2014

Vetores, API Collections: List, ArrayList, Set, HashSet, LinkedHashSet, TreeSet, LinkedHashMap, Comparable.

Aula  
05

```
//TreeSet      -> descarta pelo [compareTo] e retorna os
dados na ordem do [compareTo]
//quando uso TreeSet a ordem de descarte e ordenação é
baseada unicamente no compareTo
//TreeSet lança exceção caso a classe utilizada não
implemente Comparable

Set<Produto> produtos = new TreeSet<Produto>();

Produto p1 = new Produto(1, "Redbull", 8.0);
Produto p2 = new Produto(1, "Redbull", 6.0);
Produto p3 = new Produto(3, "Monster", 7.0);
Produto p4 = new Produto(2, "Burn", 9.0);

produtos.add(p1);
produtos.add(p2);
produtos.add(p3);
produtos.add(p4);

System.out.println("Qtd de elementos: "
+ produtos.size());

for(Produto p : produtos){
    System.out.println("Produto -> " + p);
}
}
```

Executando...

```
Java - javaAula05/src/main/Main3.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer src
javaAula05
  entity
    Produto.java
  main
    Main1.java
    Main2.java
    Main3.java
    Main4.java
    Main5.java
    Main6.java
JRE System Library [JavaSE-1.8]
Java - javaAula05/src/main/Main3.java - Eclipse
Quick Access Java EE Java
Main3.java
4 import java.util.TreeSet;
5
6 import entity.Produto;
7
8 public class Main3 {
9
10    public static void main(String[] args) {
11        //Set -> Não permite objetos dupl.
12        //comparação de igualdade o método
13        //descarta objetos iguais ao que :
14
15        //LinkedHashSet -> retorna os dados
16        //HashSet         -> retorna os dados
17        //TreeSet          -> descarta pelo |
18        //quando uso TreeSet a ordem de de
19        //TreeSet lança exceção caso a cl
20
21
22    Set<Produto> produtos = new TreeSet<Produto>();
23
24    Produto p1 = new Produto(1, "Redbu
25    Produto p2 = new Produto(1, "Redbu
26    Produto p3 = new Produto(3, "Mons
27    Produto p4 = new Produto(2, "Burn"
28
29    produtos.add(p1);
30    produtos.add(p2);
31    produtos.add(p3);
32    produtos.add(p4);
33
}
Qtd de elementos: 3
Produto -> 2, Burn, 9.0
Produto -> 3, Monster, 7.0
Produto -> 1, Redbull, 8.0
```



## Convertendo um array (vetor) simples em Lista...

```
List<Produto> lista = Arrays.asList(vetor);
```

```
package main;

import java.util.Arrays;
import java.util.List;

import entity.Produto;

public class Main4 {

    public static void main(String[] args) {

        Produto[] vetor = new Produto[3];

        vetor[0] = new Produto(1, "Diablo 3", 100.0);
        vetor[1] = new Produto(2, "Titanfall", 120.0);
        vetor[2] = new Produto(3, "Skyrim", 140.0);

        //Conversão doreta de vetor para lista
List<Produto> lista = Arrays.asList(vetor)

        for(Produto p : lista){
            System.out.println("Produto -> " + p);
        }
    }
}
```

Executando...

The screenshot shows the Eclipse IDE interface with the following details:

- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and project navigation.
- Quick Access:** A search bar at the top right.
- Java EE:** A tab labeled "Java EE" is visible on the right.
- Package Explorer:** Shows the project structure with packages "entity" and "main" under "src".
- Editor:** The main editor window displays the code for Main4.java. The code creates a list of three products (Produto) and prints them to the console.
- Console:** The bottom-right panel shows the output of the program's execution, displaying three products with their respective IDs, names, and prices.

```
package main;
import java.util.Arrays;
public class Main4 {
    public static void main(String[] args) {
        Produto[] vetor = new Produto[3];
        vetor[0] = new Produto(1, "Diablo");
        vetor[1] = new Produto(2, "Titanfall");
        vetor[2] = new Produto(3, "Skyrim");
        //Conversão direta de vetor para lista
        List<Produto> lista = Arrays.asList(vetor);
        for(Produto p : lista){
            System.out.println("Produto -> " + p);
        }
    }
}
```

Output in Console:

```
Produto -> 1, Diablo 3, 100.0
Produto -> 2, Titanfall, 120.0
Produto -> 3, Skyrim, 140.0
```



Convertendo uma lista para vetor simples...

```
Produto[] vetor2 = (Produto[]) lista.toArray();
```

```
package main;

import java.util.Arrays;
import java.util.List;
import entity.Produto;

public class Main4 {

    public static void main(String[] args) {

        Produto[] vetor = new Produto[3];
        vetor[0] = new Produto(1, "Diablo 3", 100.0);
        vetor[1] = new Produto(2, "Titanfall", 120.0);
        vetor[2] = new Produto(3, "Skyrim", 140.0);

        List<Produto> lista = Arrays.asList(vetor);

        for(Produto p : lista){
            System.out.println("Produto -> " + p);
        }

        Produto[] vetor2 = (Produto[]) lista.toArray();

        for(Produto p : vetor2){
            System.out.println("Produto -> " + p);
        }
    }
}
```

Executando...

```
Java - javaAula05/src/main/Main4.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Problems JavaDoc Declaration Console
1 package main;
2
3 import java.util.Arrays;
4
5 public class Main4 {
6
7     public static void main(String[] args) {
8
9         Produto[] vetor = new Produto[3];
10
11         vetor[0] = new Produto(1, "Diablo 3", 100.0);
12         vetor[1] = new Produto(2, "Titanfall", 120.0);
13         vetor[2] = new Produto(3, "Skyrim", 140.0);
14
15         //Conversão direta de vetor para lista
16         List<Produto> lista = Arrays.asList(vetor);
17
18         for(Produto p : lista){
19             System.out.println("Produto -> " + p);
20         }
21
22         Produto[] vetor2 = (Produto[]) lista.toArray();
23
24         for(Produto p : vetor2){
25             System.out.println("Produto -> " + p);
26         }
27     }
28 }
29
30 }
```



## Iterator

A interface iterator funciona em java similar a um cursor em banco de dados, ou seja, é utilizado para percorrer coleções de dados de forma rápida e varrendo sempre o próximo elemento caso este exista...

Para transformar uma lista em iterator podemos fazer da seguinte maneira...

```
Iterator<Produto> dados = lista.iterator();
```

Exemplo:

```
package main;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import entity.Produto;

public class Main5 {

    public static void main(String[] args) {

        List<Produto> lista = new ArrayList<Produto>();

        lista.add(new Produto(1, "Godzilla", 30.0));
        lista.add(new Produto(2, "Homem Aranha 2", 25.0));
        lista.add(new Produto(3, "X-Men", 50.0));

        //Iterator -> utilizado para varredura de dados de forma
        //sequencial, similar a um cursor em banco de dados,
        //ou seja, a varredura é feita registro a registro retornando
        //sempre um valor booleano true enquanto não for o
        //final do iterator

        Iterator<Produto> dados = lista.iterator();

        //enquanto houver um proximo elemento dentro do iterator
        while(dados.hasNext()){
            //Capturar o próximo elemento do iterator
            Produto p = dados.next();

            System.out.println("Produto -> " + p);
        }
    }
}
```



Note que a varredura feita pelo iterator sempre captura o próximo elemento da seqüência caso este exista...

```
while(dados.hasNext()){

    //Capturar o próximo elemento do iterator
    Produto p = dados.next();

    System.out.println("Produto -> " + p);
}
```

## Resultado...

The screenshot shows the Eclipse IDE interface with the following details:

- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and project navigation.
- Package Explorer:** Shows the project structure:
  - javaAula05
  - src
    - entity
      - Produto.java
    - main
      - Main1.java
      - Main2.java
      - Main3.java
      - Main4.java
      - Main5.java
      - Main6.java
  - JRE System Library [JavaSE-1.8]
- Editor:** Displays the code for Main5.java. The code uses an Iterator to print out products from a list.

```
9  public class Main5 {  
10     public static void main(String[] args) {  
11         List<Produto> lista = new ArrayList<Produto>();  
12         lista.add(new Produto(1, "Godzilla", 30.0));  
13         lista.add(new Produto(2, "Homem Aranha", 25.0));  
14         lista.add(new Produto(3, "X-Men", 50.0));  
15         //Iterator -> utilizado para varrer  
16         //sequencial, similar a um cursor  
17         //varredura é feita registro a registro  
18         //booleano true enquanto não for o final da lista  
19         Iterator<Produto> dados = lista.iterator();  
20         //enquanto houver um proximo elemento  
21         while(dados.hasNext()){  
22             //Capturar o proximo elemento  
23             Produto p = dados.next();  
24             System.out.println("Produto - " + p);  
25         }  
26     }  
27 }  
28 }  
29 }  
30 }  
31 }  
32 }  
33 }  
34 }
```
- Problems View:** Shows no errors or warnings.
- Console View:** Shows the output of the program:

```
Produto - 1, Godzilla, 30.0  
Produto - 2, Homem Aranha, 25.0  
Produto - 3, X-Men, 50.0
```
- Quick Access:** A search bar at the top right.
- Java EE:** A tab labeled Java EE is visible on the right.

## Mapas

A interface **Map** é utilizada quando desejamos armazenar valores baseados em uma chave definida. cada chave é única e pode conter um único valor associado.

Quando desejarmos recuperar um registro do mapa, podemos utilizar a chave definida para este...

```
Map<String, Produto> mapa = new LinkedHashMap<  
    String, Produto>();
```



Exemplo...

```
package main;

import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

import entity.Produto;

public class Main6 {

    public static void main(String[] args) {
        // Mapa -> armazenamento baseado em CHAVE | VALOR

        //String -> Chave do Mapa (utilizado para identificar
        //cada registro contido no mapa)
        //Produto -> Valor do Mapa (Objeto que será adicionado
        //dentro do mapa)
        Map<String, Produto> mapa = new LinkedHashMap
            <String, Produto>();

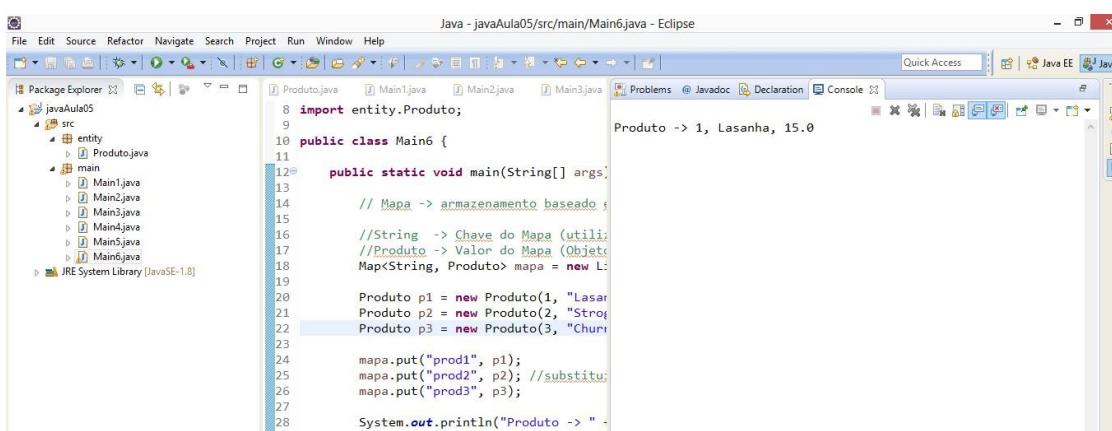
        Produto p1 = new Produto(1, "Lasanha", 15.0);
        Produto p2 = new Produto(2, "Strogonoff", 12.0);
        Produto p3 = new Produto(3, "Churrasco", 16.0);

        mapa.put("prod1", p1);
        mapa.put("prod2", p2);
        mapa.put("prod3", p3);

        System.out.println("Produto -> " + mapa.get("prod1"));
    }
}
```

**mapa.get("prod1")**

Recupera o elemento (Produto) contido no mapa na chave prod1





Percorrendo os valores e chaves do mapa utilizando foreach...

```
package main;

import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

import entity.Produto;

public class Main6 {

    public static void main(String[] args) {
        // Mapa -> armazenamento baseado em CHAVE | VALOR

        //String -> Chave do Mapa (utilizado para identificar
        //cada registro contido no mapa)
        //Produto -> Valor do Mapa (Objeto que será adicionado
        //dentro do mapa)
        Map<String, Produto> mapa = new LinkedHashMap
            <String, Produto>();

        Produto p1 = new Produto(1, "Lasanha", 15.0);
        Produto p2 = new Produto(2, "Strogonoff", 12.0);
        Produto p3 = new Produto(3, "Churrasco", 16.0);

        mapa.put("prod1", p1);
        mapa.put("prod2", p2);
        mapa.put("prod3", p3);

        System.out.println("Produto -> " + mapa.get("prod1"));

        for(Produto p : mapa.values()){
            //values() -> Cada Produto contido no mapa
            System.out.println("Produto -> " + p);
        }

        for(String chave : mapa.keySet()){
            //keySet() -> Cada Chave contida no mapa
            System.out.println("Chave -> " + chave);
        }
    }
}
```



Executando...

The screenshot shows the Eclipse IDE interface with a Java project named 'javaAula05'. The 'Main6.java' file is open in the editor, containing code that creates three products (Lasanha, Strogonoff, Churrasco) and stores them in a Map<String, Produto>. The code then prints the map's entries. The 'Console' tab shows the output of the program, which is:

```
Produto -> 1, Lasanha, 15.0
Produto -> 1, Lasanha, 15.0
Produto -> 2, Strogonoff, 12.0
Produto -> 3, Churrasco, 16.0
Chave -> prod1
Chave -> prod2
Chave -> prod3
```

### Importante:

Ao contrario do Set que descarta (despreza) elementos duplicados, caso adicionemos em um mapa dois valores na mesma chave, o segundo valor adicionado sempre substituirá o anterior e assim por diante.

Por exemplo:

```
Produto p1 = new Produto(1, "Lasanha", 15.0);
Produto p2 = new Produto(2, "Strogonoff", 12.0);
Produto p3 = new Produto(3, "Churrasco", 16.0);
```

```
mapa.put("prod1", p1);
mapa.put("prod1", p2);
//substituir o valor contido na chave prod1
```

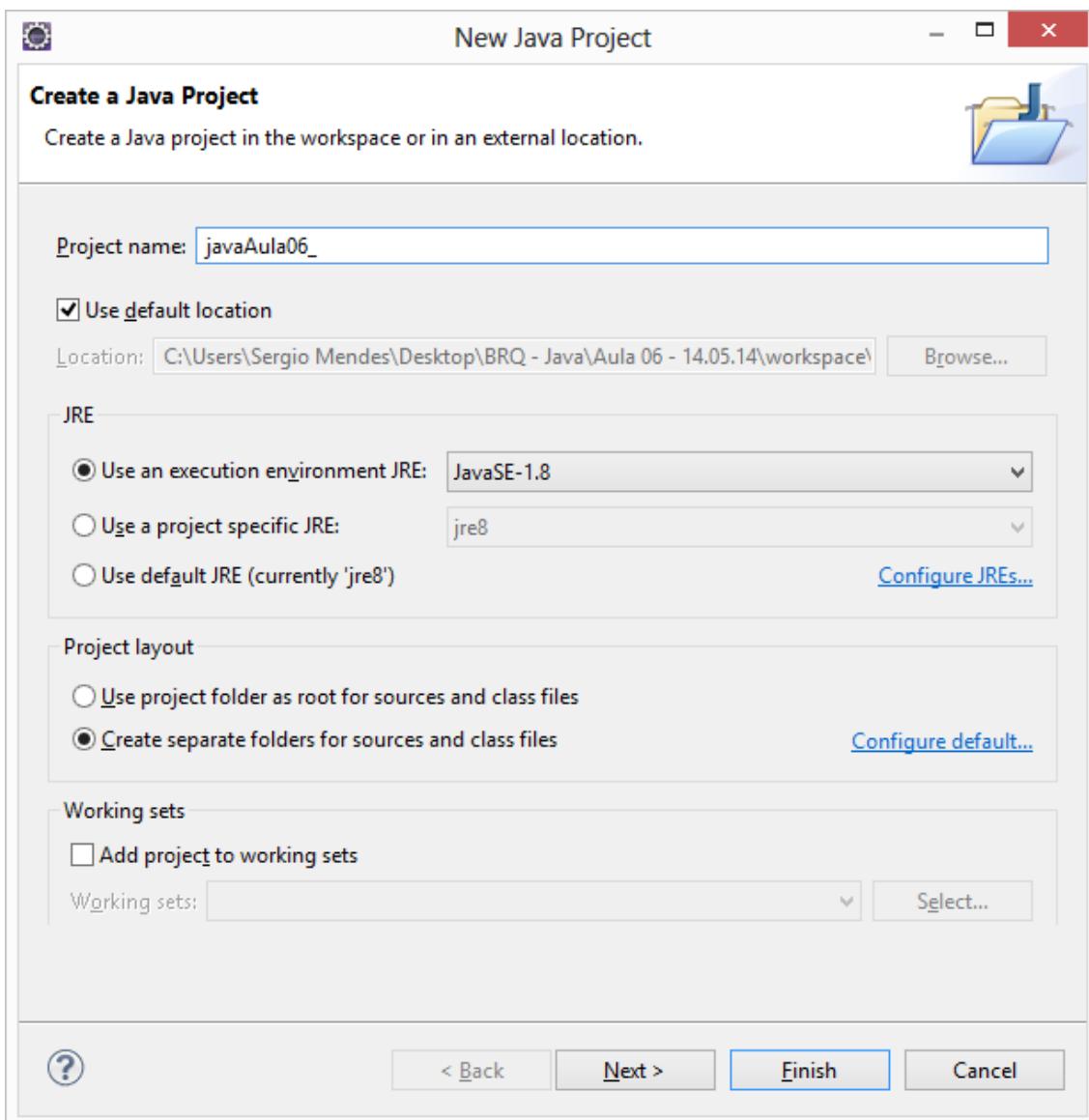
```
System.out.println("Produto -> " + mapa.get("prod1"));
```

- Saída...

Produto -> 2, Strogonoff, 12.0



### Criando o Projeto...



### Criando o JavaBean Aluno

```
package entity;

public class Aluno{

    private Integer idAluno;
    private String nome;
    private Double nota;

    public Aluno() {

}
```



```
public Aluno(Integer idAluno, String nome, Double nota) {
    super();
    this.idAluno = idAluno;
    this.nome = nome;
    this.nota = nota;
}

public Integer getIdAluno() {
    return idAluno;
}

public void setIdAluno(Integer idAluno) {
    this.idAluno = idAluno;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getNota() {
    return nota;
}

public void setNota(Double nota) {
    this.nota = nota;
}

@Override
public String toString() {
    return "Aluno [idAluno=" + idAluno + ", nome="
           + nome + ", nota=" + nota + "]";
}

}
```

## Listas (java.util.List)

Listas são utilizadas em Java para criação de seqüência de dados ou objetos. Podemos, através de uma lista:

- Armazenar quantidade indefinida de elementos
- Obter elementos da lista baseado em sua posição / índice
- Ordenar os elementos da lista, desde que estes implementem a interface Comparable



Criando uma Classe de Controle para realizar operações com List

```
package control;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import entity.Aluno;

public class ControleAluno {

    //Atributo -> armazenar muitos Alunos
    private List<Aluno> alunos; //null (sem espaço de memória)

    //método para inicializar a lista de alunos
    public void inicializarListagemAlunos(){
        alunos = new ArrayList<Aluno>(); //lista vazia!
    }

    //Método para adicionar um objeto Aluno dentro da lista
    public void adicionarAluno(Aluno a) throws Exception{
        //adiciono o objeto 'a' dentro da lista
        alunos.add(a);
    }

    //Método para retornar a quantidade de alunos contidos na lista
    public Integer obterQuantidadeAlunos() throws Exception{
        //size() -> retorna a quantidade de elementos na sequencia
        return alunos.size();
    }

    //Buscar um aluno dentro da lista pela posição dentro da lista
    public Aluno obterAluno(int posicao) throws Exception{
        //get(posicao) -> retorna 1 elemento da lista pela posição
        return alunos.get(posicao);
    }
}
```

Testando os métodos da Classe de Controle...

```
package main;

import control.ControleAluno;
import entity.Aluno;

public class Main {
```



```
public static void main(String[] args) {  
  
    Aluno a1 = new Aluno(1, "Yuri", 9.5);  
    Aluno a2 = new Aluno(2, "Cristiane", 9.0);  
    Aluno a3 = new Aluno(3, "Fagner", 10.0);  
  
    ControleAluno c = new ControleAluno();  
    c.inicializarListagemAlunos();  
  
    try {  
  
        c.adicionarAluno(a1);  
        c.adicionarAluno(a2);  
        c.adicionarAluno(a3);  
  
        System.out.println("Qtd de Alunos -> "  
                           + c.obterQuantidadeAlunos());  
  
        System.out.println("Aluno posição 0 -> "  
                           + c.obterAluno(0));  
  
        System.out.println("Aluno posição 1 -> "  
                           + c.obterAluno(1));  
  
        System.out.println("Aluno posição 2 -> "  
                           + c.obterAluno(2));  
  
    }  
    catch (Exception e) {  
        System.out.println("Erro -> " + e.getMessage());  
    }  
}  
}
```

## Resultado...

The screenshot shows the Eclipse IDE interface with the Java project 'javaAula06' open. The 'Main.java' file is selected in the editor. The code implements a simple application that creates three 'Aluno' objects and adds them to a list managed by a 'ControleAluno' object. It then prints the total number of students and each student's details. The output window displays the results:

```
Qtd de Alunos -> 3  
Aluno posição 0 -> Aluno [idAluno=1, nome=Yuri, nota=9.5]  
Aluno posição 1 -> Aluno [idAluno=2, nome=Cristiane, nota=9.0]  
Aluno posição 2 -> Aluno [idAluno=3, nome=Fagner, nota=10.0]
```



Implementando a interface Comparable na Classe Aluno

```
package entity;

public class Aluno implements Comparable<Aluno>{

    private Integer idAluno;
    private String nome;
    private Double nota;

    public Aluno() {
    }

    public Aluno(Integer idAluno, String nome, Double nota) {
        super();
        this.idAluno = idAluno;
        this.nome = nome;
        this.nota = nota;
    }

    public Integer getIdAluno() {
        return idAluno;
    }

    public void setIdAluno(Integer idAluno) {
        this.idAluno = idAluno;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Double getNota() {
        return nota;
    }

    public void setNota(Double nota) {
        this.nota = nota;
    }

    @Override
    public String toString() {
        return "Aluno [idAluno=" + idAluno + ", nome=" + nome
               + ", nota=" + nota + "]";
    }
}
```



```
@Override
public int compareTo(Aluno a) {
    //regra: comparar em ordem crescente o atributo
    //nota da classe Aluno
    //com o valor da nota obtida do objeto
    //Aluno passado no compareTo
    return nota.compareTo(a.getNota());
    //return a.getNota().compareTo(nota); //decrescente
}

}
```

Criando um método para ordenar a lista de alunos na Classe de Controle...

```
package control;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import entity.Aluno;

public class ControleAluno {

    //Atributo -> armazenar muitos Alunos
    private List<Aluno> alunos; //null (sem espaço de memória)

    //método para inicializar a lista de alunos
    public void inicializarListagemAlunos(){
        alunos = new ArrayList<Aluno>(); //lista vazia!
    }

    //Método para adicionar um objeto Aluno dentro da lista
    public void adicionarAluno(Aluno a) throws Exception{
        //adiciono o objeto 'a' dentro da lista
        alunos.add(a);
    }

    //Método para retornar a quantidade de alunos contidos na lista
    public Integer obterQuantidadeAlunos() throws Exception{
        //size() -> retorna a quantidade de elementos na sequencia
        return alunos.size();
    }

    //Buscar um aluno dentro da lista pela posição dentro da lista
    public Aluno obterAluno(int posicao) throws Exception{
        //get(posicao) -> retorna 1 elemento da lista pela posição
        return alunos.get(posicao);
    }
}
```



```
//Método para ordenar os alunos da menor para a maior nota
public void ordenarAlunos() throws Exception{
    //Collections.sort -> ordena baseado na
    //regra do Comparable
    Collections.sort(alunos);
}
```

Executando...

```
package main;

import control.ControleAluno;
import entity.Aluno;

public class Main {

    public static void main(String[] args) {

        Aluno a1 = new Aluno(1, "Yuri", 9.5);
        Aluno a2 = new Aluno(2, "Cristiane", 9.0);
        Aluno a3 = new Aluno(3, "Fagner", 10.0);

        ControleAluno c = new ControleAluno();
        c.inicializarListagemAlunos();

        try {

            c.adicionarAluno(a1);
            c.adicionarAluno(a2);
            c.adicionarAluno(a3);

            System.out.println("Qtd de Alunos -> "
                    + c.obterQuantidadeAlunos());

            c.ordenarAlunos();

            System.out.println("Aluno posição 0 -> "
                    + c.obterAluno(0));
            System.out.println("Aluno posição 1 -> "
                    + c.obterAluno(1));
            System.out.println("Aluno posição 2 -> "
                    + c.obterAluno(2));

        }
        catch (Exception e) {
            System.out.println("Erro -> " + e.getMessage());
        }
    }
}
```



# Java WebDeveloper - BRQ

Quarta-feira, 14 de Maio de 2014

List e ArrayList, Map, Queue, Enums e operador static

Aula  
06

Executando...

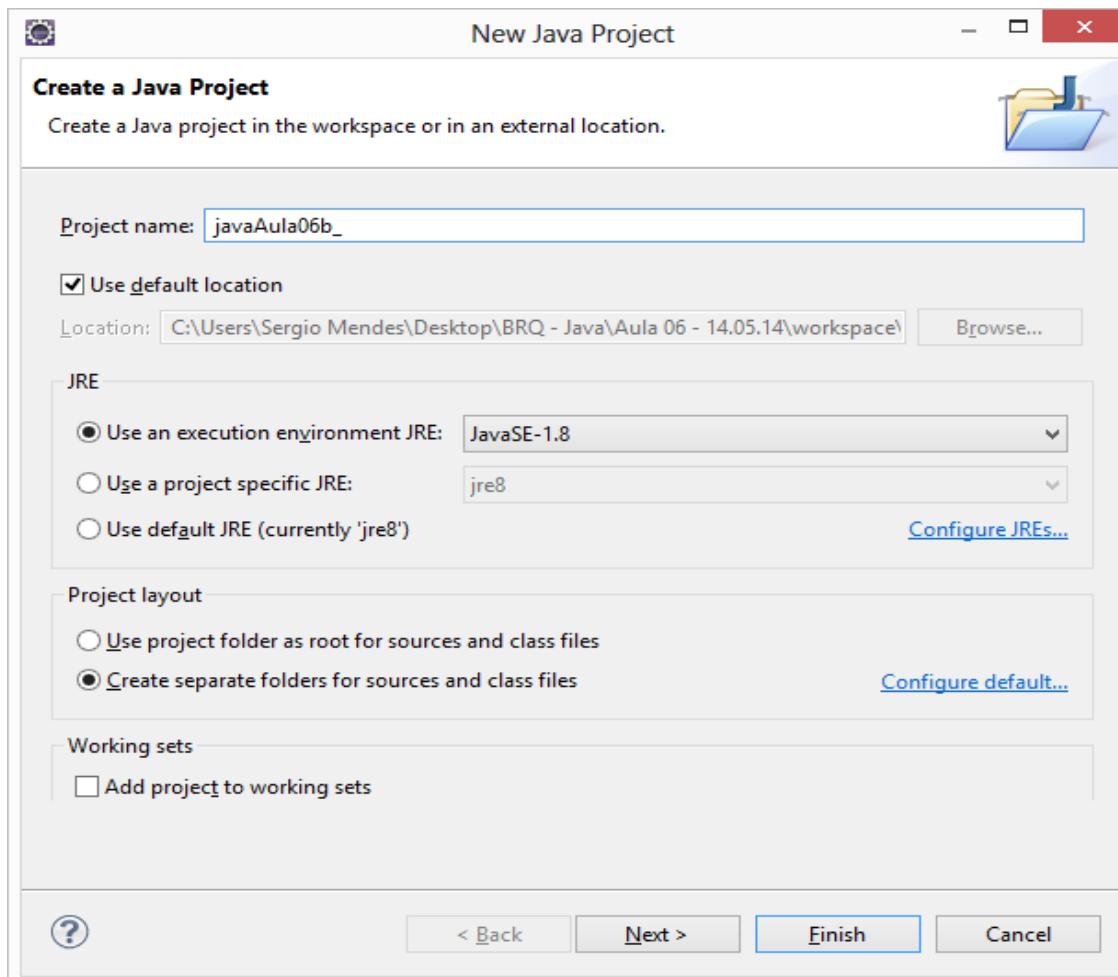
The screenshot shows the Eclipse IDE interface with the title "Java - javaAula06/src/main/Main.java - Eclipse". The code in Main.java is as follows:

```
1 package main;
2
3 import control.ControleAluno;
4
5 public class Main {
6
7     public static void main(String[] args) {
8         Aluno a1 = new Aluno(1, "Cristiane", 9.0);
9         Aluno a2 = new Aluno(2, "Yuri", 9.5);
10        Aluno a3 = new Aluno(3, "Fagner", 10.0);
11
12        ControleAluno c = new ControleAluno();
13        c.inicializarListagemAlunos();
14
15        try {
16            c.adicionarAluno(a1);
17            c.adicionarAluno(a2);
18            c.adicionarAluno(a3);
19
20            System.out.println("Qtd de Alunos -> " + c.obterQuantidadeAlunos());
21
22            c.ordenarAlunos();
23
24            System.out.println("Aluno posição 0 -> " + c.obterAluno(0));
25            System.out.println("Aluno posição 1 -> " + c.obterAluno(1));
26            System.out.println("Aluno posição 2 -> " + c.obterAluno(2));
27        } catch (Exception e) {
28            e.printStackTrace();
29        }
30    }
31}
```

The output window shows the results of the program execution:

```
Qtd de Alunos -> 3
Aluno posição 0 -> Aluno [idAluno=2, nome=Cristiane, nota=9.0]
Aluno posição 1 -> Aluno [idAluno=1, nome=Yuri, nota=9.5]
Aluno posição 2 -> Aluno [idAluno=3, nome=Fagner, nota=10.0]
```

Novo Projeto...





## Mapas (java.util.Map)

Mapas são utilizados para armazenamento de dados baseado em chave e valor. Cada objeto incluído no mapa é endereçado por uma chave única...

Criando a Classe JavaBean Usuario

```
package entity;

public class Usuario {

    private Integer idUsuario;
    private String login;
    private String senha;
    private String perfil;

    public Usuario() {
    }

    public Usuario(Integer idUsuario, String login,
                   String senha, String perfil) {
        super();
        this.idUsuario = idUsuario;
        this.login = login;
        this.senha = senha;
        this.perfil = perfil;
    }

    @Override
    public String toString() {
        return "Usuario [idUsuario=" + idUsuario + ", login="
               + login + ", senha=" + senha + ", perfil="
               + perfil + "]";
    }

    public Integer getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }
}
```



```
public String getSenha() {
    return senha;
}

public void setSenha(String senha) {
    this.senha = senha;
}

public String getPerfil() {
    return perfil;
}

public void setPerfil(String perfil) {
    this.perfil = perfil;
}

}
```

Criando a Classe de Controle para realizar operações com o Mapa

```
package control;

import java.util.LinkedHashMap;
import java.util.Map;

import entity.Usuario;

public class ControleUsuario {

    //Atributo -> Mapa de Usuarios
    private Map<String, Usuario> mapa; //Chave | Valor

    public void inicializarMapa(){

        //espaço de memória para o atributo mapa (vazio)
        mapa = new LinkedHashMap<String, Usuario>();
    }

    //adicionar um elemento no mapa
    public void adicionarUsuario(Usuario u) throws Exception{

        //chave -> u.getLogin() | valor -> u
        mapa.put(u.getLogin(), u);
    }
}
```



```
//método para obter 1 Usuario pelo login
public Usuario obterUsuario(String login) throws Exception{

    //containsKey -> verifica se o mapa possui
    //a chave especificada
    if(mapa.containsKey(login)){
        //retornar o Usuario do mapa
        //contido na chave especificada
        return mapa.get(login);
    }
    else{
        //gerando uma exceção e enviando mensagem de erro
        throw new Exception("Usuario " + login
            + " não encontrado no mapa.");
    }
}
```

Executando...

```
package main;

import control.ControleUsuario;
import entity.Usuario;

public class Main {

    public static void main(String[] args) {

        Usuario u1 = new Usuario(1, "ana", "1234",
                               "Cliente");

        Usuario u2 = new Usuario(2, "rui", "5678",
                               "Administrador");

        Usuario u3 = new Usuario(3, "leo", "9876",
                               "Funcionario");

        ControleUsuario c = new ControleUsuario();
        c.inicializarMapa();

        try{

            c.adicionarUsuario(u1);
            c.adicionarUsuario(u2);
            c.adicionarUsuario(u3);

            System.out.println("Registro -> "
                + c.obterUsuario("ana"));

            System.out.println("Registro -> "
                + c.obterUsuario("bia")); //erro
        }
    }
}
```



# Java WebDeveloper - BRQ

Quarta-feira, 14 de Maio de 2014

List e ArrayList, Map, Queue, Enums e operador static

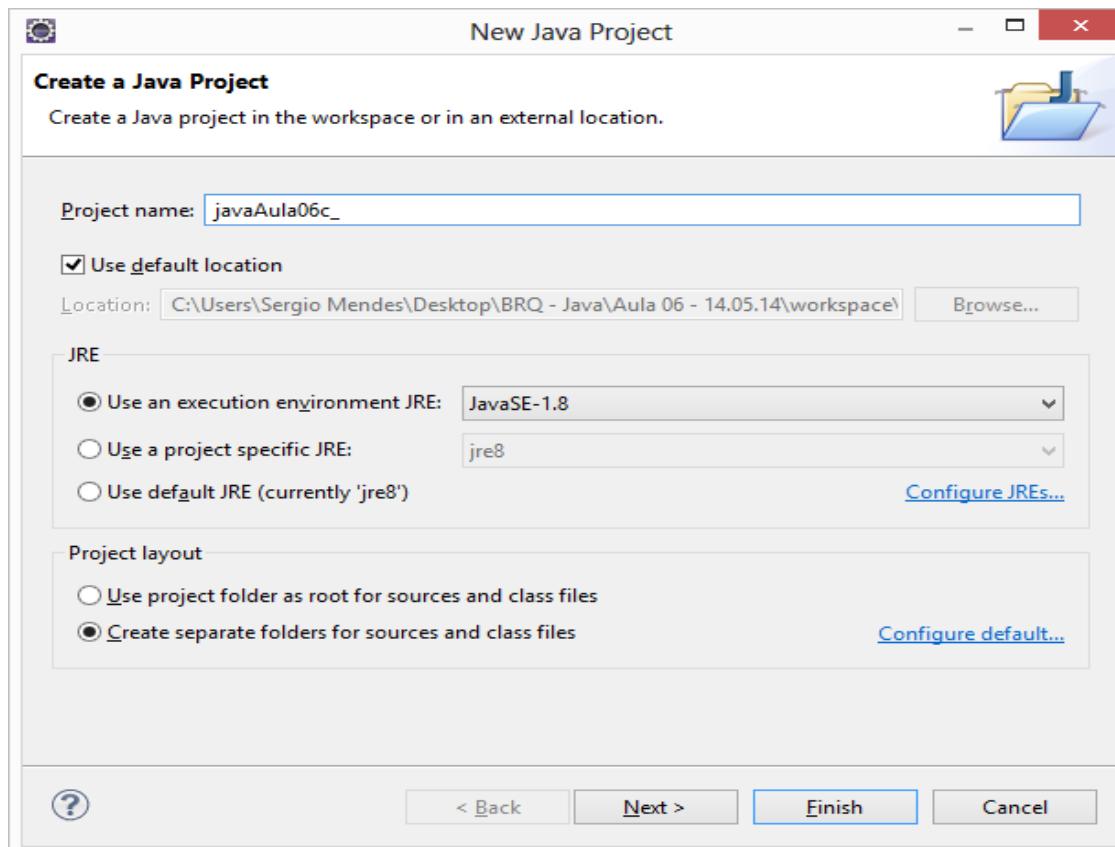
Aula  
06

```
        }
    catch(Exception e) {
        System.out.println("Erro -> " + e.getMessage());
    }
}
}
```

The screenshot shows the Eclipse IDE interface. The title bar says "Java - javaAula06b/src/main/Main.java - Eclipse". The left side has a "Package Explorer" view showing a project structure with packages like "javaAula06", "javaAula06b", and "src" containing files like "ControlUsuario.java", "Usuario.java", and "Main.java". The main editor area displays Java code for a "Main" class. The code includes imports for "control.ControlUsuario", a try-catch block for adding users to a map, and println statements for registration and errors. The status bar at the bottom right shows "Java EE".

```
Java - javaAula06b/src/main/Main.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
src
  control
    ControlUsuario.java
  entity
    Usuario.java
  main
    Main.java
JRE System Library [JavaSE-1.8]
javaAula06c
javaAula06d
javaAula07e
Main
Problems Javadoc Declaration Console
Quick Access Java EE Java
1 package main;
2
3* import control.ControlUsuario;
4
5 public class Main {
6
7     public static void main(Str
8
9         Usuario u1 = new Usuario();
10        Usuario u2 = new Usuario();
11        Usuario u3 = new Usuario();
12
13        ControlUsuario c = new
14            c.inicializarMapa();
15
16
17        try{
18
19            c.adicionarUsuario(
20            c.adicionarUsuario(
21            c.adicionarUsuario(u3);
22
23            System.out.println("Registro -> " + c.obterUsuario("ana"));
24            System.out.println("Registro -> " + c.obterUsuario("bia")); //erro
25
26
27        }catch(Exception e){
28            System.out.println("Erro -> " + e.getMessage());
29
}
```

Novo Projeto...





## Filas (java.util.Queue)

A interface Queue comporta-se como uma Fila, ou seja, sempre resgata ou retira o primeiro elemento inserido na fila.

JavaBean Cliente...

```
package entity;

public class Cliente implements Comparable<Cliente>{

    private Integer idCliente;
    private String nome;

    public Cliente() {
    }

    public Cliente(Integer idCliente, String nome) {
        super();
        this.idCliente = idCliente;
        this.nome = nome;
    }

    @Override
    public String toString() {
        return "Cliente [idCliente=" + idCliente
               + ", nome=" + nome + "]";
    }

    public Integer getIdCliente() {
        return idCliente;
    }

    public void setIdCliente(Integer idCliente) {
        this.idCliente = idCliente;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    @Override
    public int compareTo(Cliente c) {
        return idCliente.compareTo(c.getIdCliente());
    }
}
```



Criando uma Classe de Controle para execução de operações na Fila...

```
package control;

import java.util.PriorityQueue;
import java.util.Queue;

import entity.Cliente;

public class ControleCliente {

    //Atributo -> Queue (fila)
    private Queue<Cliente> fila; //null

    //Método para inicializar a fila
    public void inicializarFila(){

        //espaço de memória para a fila
        fila = new PriorityQueue<Cliente>();
        //FIFO -> Primeiro que entra, primeiro que sai
    }

    //Método para adicionar cliente na fila
    public void adicionarCliente(Cliente c) throws Exception{
        fila.add(c);
        //adiciono o parametro cliente dentro da fila
    }

    //Método que retornasse o primeiro elemento da fila
    public Cliente obterPrimeiroDaFila() throws Exception{
        //peek() -> retorna o primeiro elemento da fila
        return fila.peek();
    }

    //Método para retornar a quantidade de elementos da fila
    public Integer obterTamanhoDaFila() throws Exception{
        return fila.size(); //qtd de eleemtns
    }

    //Método para retornar e retirar o primeiro da fila
    public Cliente retirarPrimeiroDaFila() throws Exception{
        //poll() -> retorna o primeiro elemento
        //da fila, retirando-o da fila
        return fila.poll();
    }

}
```



Executando...

```
package main;

import control.ControleCliente;
import entity.Cliente;

public class Main {

    public static void main(String[] args) {

        Cliente c1 = new Cliente(1, "Joao");
        Cliente c2 = new Cliente(2, "Carlos");
        Cliente c3 = new Cliente(3, "Bruna");
        Cliente c4 = new Cliente(4, "Pedro");

        ControleCliente c = new ControleCliente();
        c.inicializarFila();

        try{

            c.adicionarCliente(c1);
            c.adicionarCliente(c2);
            c.adicionarCliente(c3);
            c.adicionarCliente(c4);

            System.out.println("Tamanho da Fila      -> "
                + c.obterTamanhoDaFila());

            System.out.println("Primeiro da Fila      -> "
                + c.obterPrimeiroDaFila());

            System.out.println("Tamanho da Fila      -> "
                + c.obterTamanhoDaFila());

            System.out.println("Atendendo 1º da Fila -> "
                + c.retirarPrimeiroDaFila());

            System.out.println("Tamanho da Fila      -> "
                + c.obterTamanhoDaFila());

            System.out.println("Primeiro da Fila      -> "
                + c.obterPrimeiroDaFila());
        }

        catch(Exception e){
            System.out.println("Erro -> " + e.getMessage());
        }

    }
}
```



# **Java WebDeveloper - BRQ**

**Quarta-feira, 14 de Maio de 2014**

## List e ArrayList, Map, Queue, Enums e operador static

# Aula 06

Java - javaAula06c/src/main/Main.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access Java EE Java

Package Explorer

javaAula06

javaAula06b

src

- control
- ControleUsuario.java
- Usuario.java

entity

- Main.java

JRE System Library [JavaSE-1.8]

javaAula06c

src

- control
- ControleCliente.java

entity

- Cliente.java

main

- Main.java

JRE System Library [JavaSE-1.8]

javaAula06d

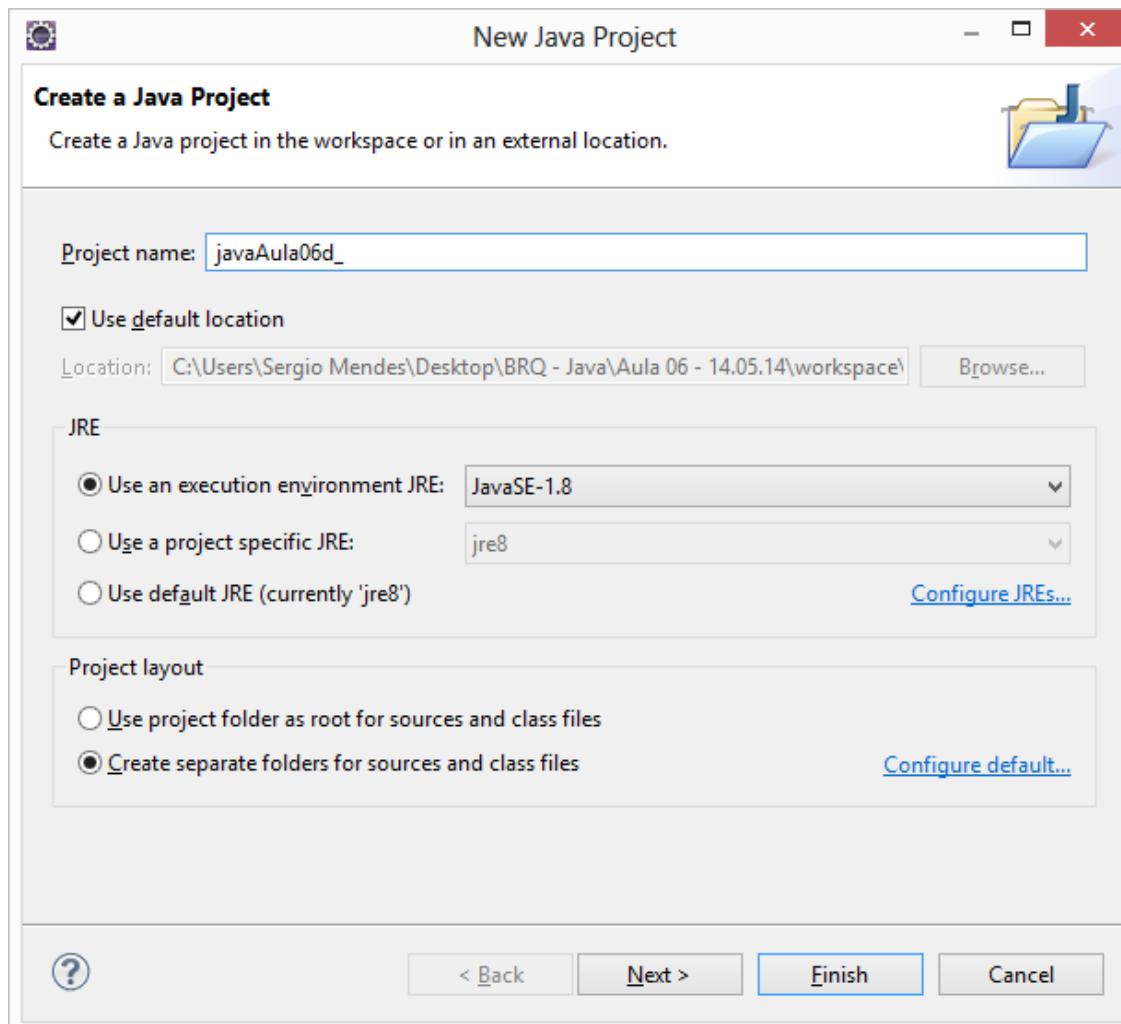
javaAula07e

ControleUsuario.java

Problems @ Javadoc Declaration Console

```
1 package main;
2
3#import control.ControleCliente;
4
5 public class Main {
6
7     public static void main(Str
8
9         Cliente c1 = new Client
10        Cliente c2 = new Client
11        Cliente c3 = new Client
12        Cliente c4 = new Client
13
14        ControleCliente c = new
15        c.inicializarFila();
16
17        try{
18
19            c.adicionarCliente(
20            c.adicionarCliente(c2,
21            c.adicionarCliente(c3);
22            c.adicionarCliente(c4);
23
24
25            System.out.println("Tamanho da Fila      -> " + c.obterTamanhoDaFila());
26            System.out.println("Primeiro da Fila     -> " + c.obterPrimeiroDaFila());
27
28            System.out.println("Tamanho da Fila      -> " + c.obterTamanhoDaFila());
29            System.out.println("Atendendo 1º da Fila -> " + c.retirarPrimeiroDaFila());
30
31            System.out.println("Tamanho da Fila      -> " + c.obterTamanhoDaFila());
```

Novo Projeto...





## Enum

Enums são Classes Java utilizadas para especificar valores constantes que poderão ser relacionados a outras classes..

- Exemplo:

```
package entity;

//tipo multivalorado
public enum Sexo {

    //Todo Enum é uma Classe composta, basicamente, de
    //valores constantes

    Masculino, Feminino

}

package entity;

public enum EstadoCivil {

    Solteiro,
    Casado,
    Viuvo,
    Divorciado

}
```

### Utilizando as Classes Enum...

```
package entity;

public class Pessoa {

    private Integer idPessoa;
    private String nome;
    private Sexo sexo;
    private EstadoCivil estadoCivil;

    public Pessoa() {

    }
```



```
public Pessoa(Integer idPessoa, String nome, Sexo sexo,
              EstadoCivil estadoCivil) {
    super();
    this.idPessoa = idPessoa;
    this.nome = nome;
    this.sexo = sexo;
    this.estadoCivil = estadoCivil;
}

@Override
public String toString() {
    return "Pessoa [idPessoa=" + idPessoa + ", nome="
           + nome + ", sexo=" + sexo + ", estadoCivil="
           + estadoCivil + "]";
}

public Integer getIdPessoa() {
    return idPessoa;
}

public void setIdPessoa(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Sexo getSexo() {
    return sexo;
}

public void setSexo(Sexo sexo) {
    this.sexo = sexo;
}

public EstadoCivil getEstadoCivil() {
    return estadoCivil;
}

public void setEstadoCivil(EstadoCivil estadoCivil) {
    this.estadoCivil = estadoCivil;
}

}
```



- Executando...

```
package main;

import entity.EstadoCivil;
import entity.Pessoa;
import entity.Sexo;

public class Main {

    public static void main(String[] args) {

        Pessoa p1 = new Pessoa
            (1, "Ana", Sexo.Feminino, EstadoCivil.Solteiro);

        Pessoa p2 = new Pessoa
            (2, "Leo", Sexo.Masculino, EstadoCivil.Casado);

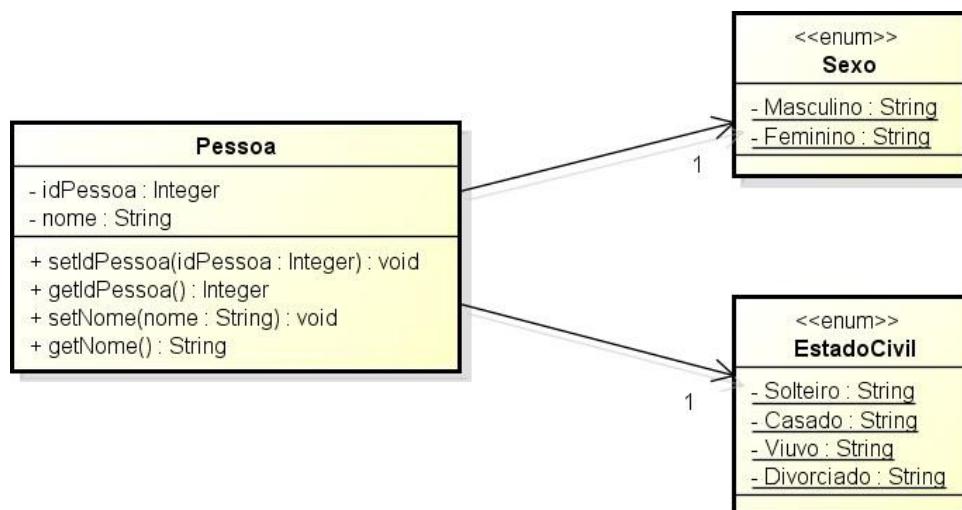
        System.out.println("Pessoa -> " + p1); //toString
        System.out.println("Pessoa -> " + p2); //toString
    }
}
```

- Saída...

The screenshot shows the Eclipse IDE interface with the title bar "Java - javaAula06d/src/main/Main.java - Eclipse". The "Console" tab is selected, displaying the output of the Java code execution:

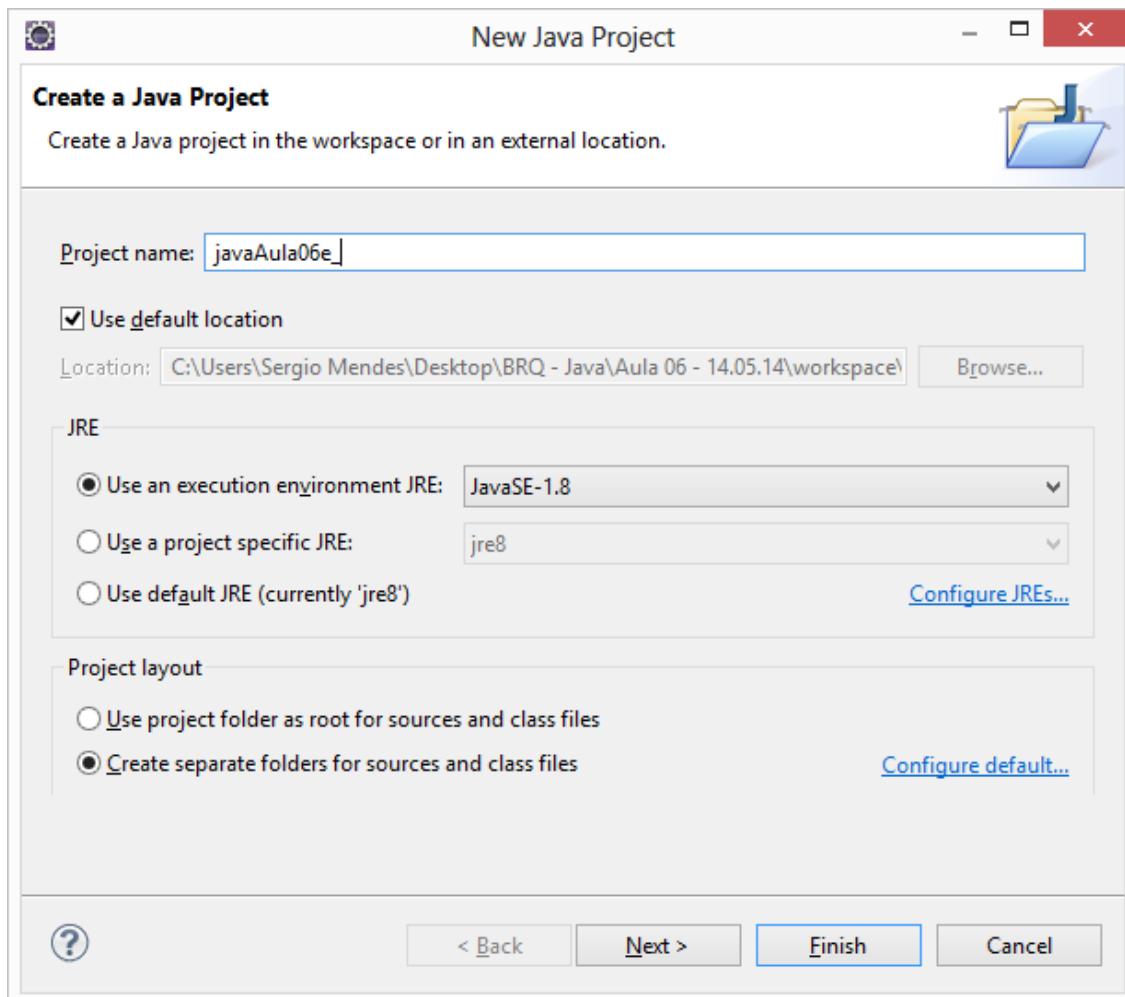
```
Pessoa -> Pessoa [idPessoa=1, nome=Ana, sexo=Feminino, estadoCivil=Solteiro]
Pessoa -> Pessoa [idPessoa=2, nome=Leo, sexo=Masculino, estadoCivil=Casado]
```

The "Package Explorer" view shows the project structure with packages like "javaAula06", "javaAula06b", "javaAula06c", and "javaAula06d". The "src" folder contains classes "Main.java", "EstadoCivil.java", "Pessoa.java", and "Sexo.java". The "Main.java" file is open in the editor.





Novo Projeto...



## Operador **static**

Utilizado para garantir que um método ou atributo de uma Classe Java já terá endereçamento de memória gerado pela própria Maquina virtual.

Não é necessário instanciar uma Classe para acessar seus atributos ou métodos se estes forem estáticos...

- Exemplo:

```
package control;

import java.text.SimpleDateFormat;
import java.util.Date;
```



# Java WebDeveloper - BRQ

Quarta-feira, 14 de Maio de 2014

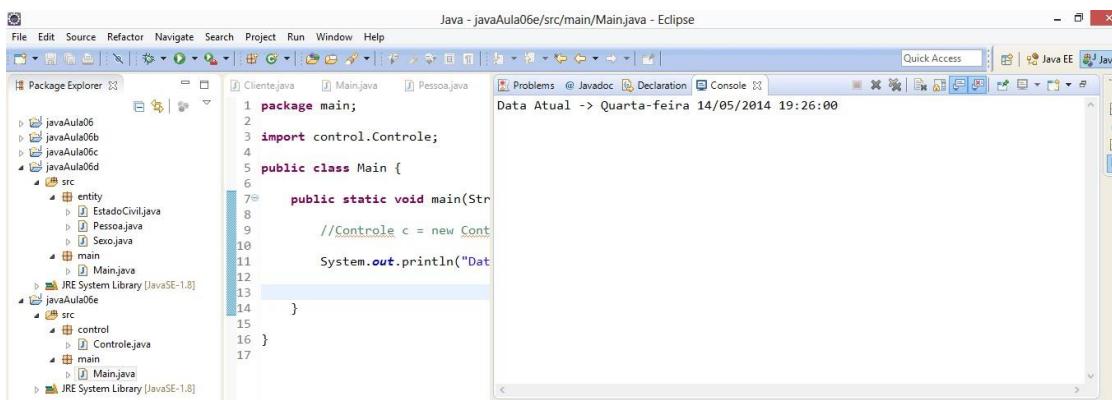
List e ArrayList, Map, Queue, Enums e operador static

Aula  
06

```
public class Controle {  
  
    public static String obterData() {  
  
        Date data = new Date();  
  
        SimpleDateFormat sdf = new SimpleDateFormat  
            ("EEEE dd/MM/yyyy HH:mm:ss");  
  
        return sdf.format(data);  
    }  
  
}
```

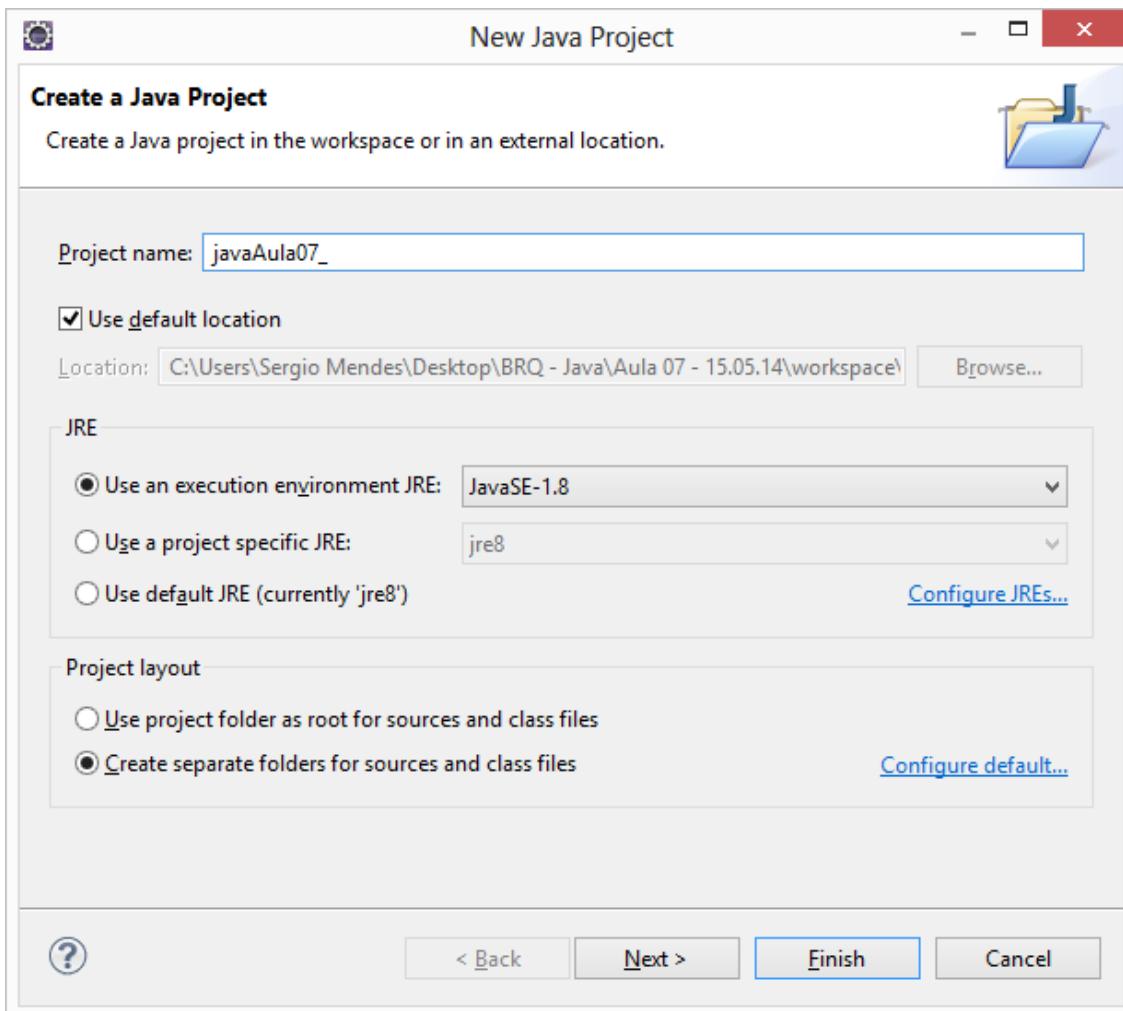
Executando...

```
package main;  
  
import control.Controle;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        //Controle c = new Controle();  
  
        System.out.println("Data Atual -> "  
            + Controle.obterData());  
    }  
}
```





### Criando o Projeto...



### static

O operador static define que um atributo ou método de uma Classe já terá endereçamento de memória garantido pela máquina virtual. Não é necessário instanciar uma Classe para executar um atributo ou método estático contido nesta.

```
package control;

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

public class Controle {

    //Constante
    //final -> não permite que o valor do atributo
    //seja alterado
```



```
private static final
    String FORMATO = "EEEE dd 'de' MMMM 'de' yyyy";

    //static -> faz com que a JVM já garanta
    //que um recurso estará com espaço de memória definido
    //O static pode ser utilizado em:
    //    -> atributos
    //    -> métodos
    //    -> classe se forem inner class

    //Método para retornar a data do Java formatada
public static String obterData(){

    //Todo new Date() já 'seta' a data
    //com o valor da data atual
    //do sistema onde a JVM está sendo executada
    Date dataAtual = new Date();

    //Classe Java para formatação de data
    /*
     * dd    -> dia    hh -> hora      EE    -> dia da
     *                   semana abreviado
     * MM    -> mês    mm -> minuto   EEEE  -> dia da
                                         semana por extenso
     * yyyy -> ano    ss -> segundo
     */
    SimpleDateFormat fmt = new SimpleDateFormat(FORMATO);

    //aplicando a formatação à data do sistema
    return fmt.format(dataAtual);
}

//Sobrecarga do método obterData
public static String obterData(int dia, int mes, int ano){

    //Date data = new Date(mes, dia, ano);
    //Calendar trata os meses baseado em
    //índice 0, ou seja jan-dez = 0-11
    Calendar cal = new GregorianCalendar
                    (ano, mes - 1, dia);

    //Transformar o calendar em date...
    Date data = cal.getTime();

    SimpleDateFormat fmt = new SimpleDateFormat(FORMATO);

    //retornar a data formatada
    return fmt.format(data);
}

}
```



## Atributos constantes

São aqueles cujo valor não poderá ser modificado. Geralmente uma constante em Java é definida como static e final e com seu nome em caixa-alta.

Exemplo:

```
private static final String FORMATO = "dd/MM/yyyy";
```

## Classe java.util.Date

Classe padrão do Java para armazenamento de Datas. Seus métodos encontram-se na maioria obsoletos, tornando o tipo Date utilizado apenas para guardar um valor data. Operações de calculo ou manipulação de data são mais indicadas para a api **Callendar**

Exemplo:

```
Date dataAtual = new Date();
```

## Callendar

Api do Java para recursos de manipulação de data e hora avançados.  
Exemplo:

```
Calendar cal = new GregorianCalendar();
```

```
public static String obterData(int dia, int mes, int ano){  
  
    //Date data = new Date(mes, dia, ano);  
    //Callendar trata os meses baseado em  
    //índice 0, ou seja jan-dez = 0-11  
    Calendar cal = new GregorianCalendar(ano, mes - 1, dia);  
  
    //Transformar o calendar em date...  
    Date data = cal.getTime();  
  
    SimpleDateFormat fmt = new SimpleDateFormat(FORMATO);  
  
    //retornar a data formatada  
    return fmt.format(data);  
}
```



Executando...

```
package main;

import control.Controle;

public class Main {

    public static void main(String[] args) {

        System.out.println("Data -> "
                           + Controle.obterData());

        System.out.println("Data -> "
                           + Controle.obterData(20, 5, 2014));

    }
}
```

Saída do Programa:

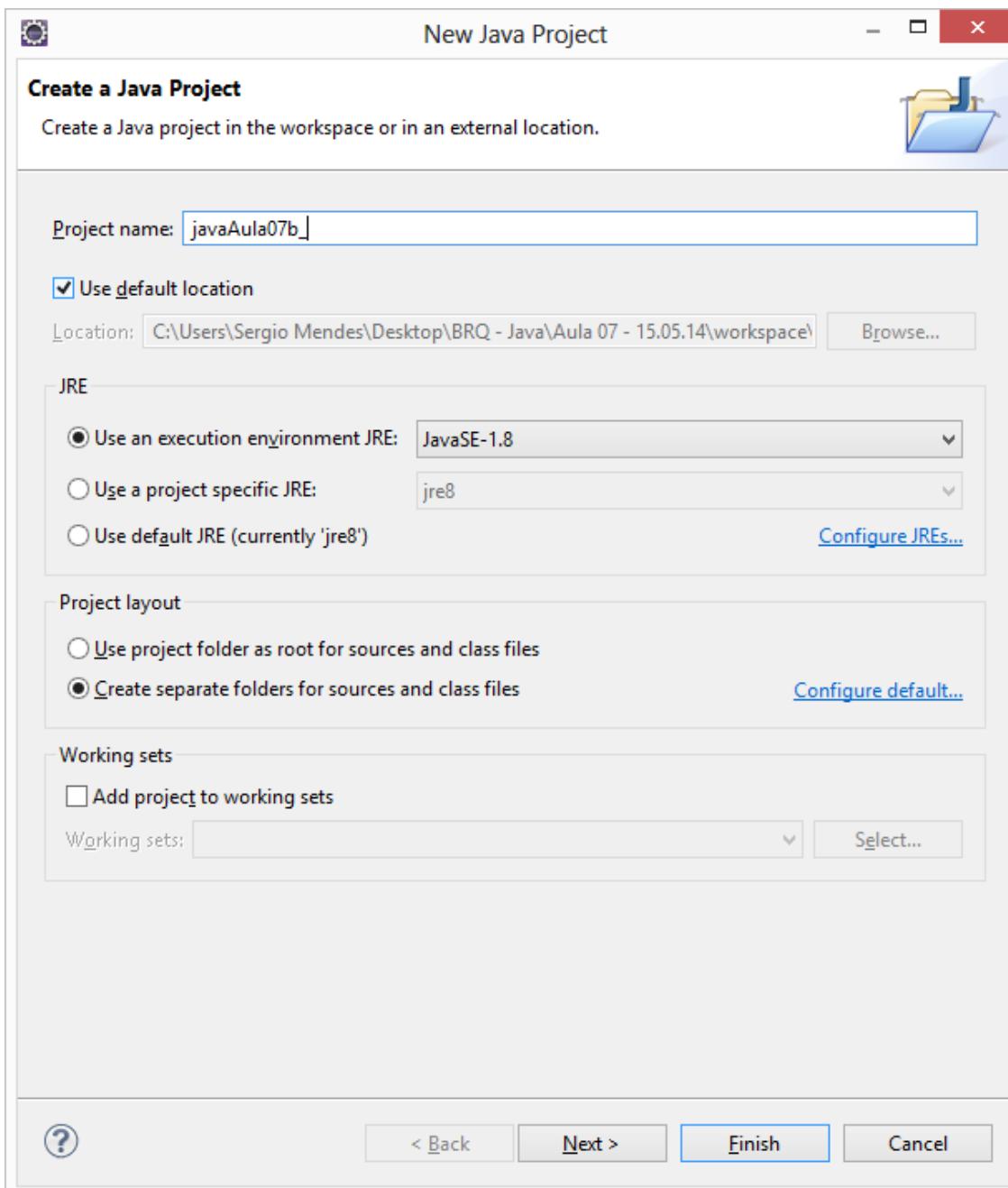
The screenshot shows the Eclipse IDE interface with the title bar "Java - javaAula07/src/main/Main.java - Eclipse". The left side displays the "Package Explorer" showing two projects: "javaAula07" and "javaAula07b". The "src" folder under "javaAula07" contains packages "control" and "main", each with a file named "Controle.java" and "Main.java" respectively. The "src" folder under "javaAula07b" contains packages "entity", "main", and "validators", each with files "Produto.java", "Venda.java", and "Main.java". The right side of the interface shows the "Console" view with the following output:  
Data -> Quinta-feira 15 de Maio de 2014  
Data -> Terça-feira 20 de Maio de 2014

Estrutura do Projeto...





Novo Projeto...



Criando uma Classe para formatação de valores decimais em moeda...

```
package util;

import java.text.NumberFormat;

public class Moeda {

    //Método para receber um numero real e
    //retorná-lo em formato moeda
```



```
public static String formatarMoeda(double valor){  
  
    //getCurrencyInstance -> aponta o NumberFormat  
    //para moeda local  
    NumberFormat fmt = NumberFormat.getCurrencyInstance();  
    fmt.setMinimumFractionDigits(2);  
    fmt.setMaximumFractionDigits(3);  
  
    return fmt.format(valor);  
}  
}
```

Criando uma Classe para formatação de valores decimais em percentual...

```
package util;  
  
import java.text.NumberFormat;  
  
public class Percentual {  
  
    //Método para receber um valor numerico decimal e retorna-lo  
    //em formato percentual...  
    public static String formatarPercentual(double valor){  
  
        //getPercentInstance -> inicializa como padrão percentual  
        NumberFormat fmt = NumberFormat.getPercentInstance();  
        fmt.setMaximumFractionDigits(2);  
  
        return fmt.format(valor);  
    }  
}
```

## Expressões Regulares

Tem como objetivo testar o conteúdo de variáveis String de forma a verificar se a mesma encontra-se dentro de um padrão estabelecido.

É uma linguagem baseada em máscara de validação. A idéia de um Regex é verificar a conformidade de uma string de texto com uma expressão (máscara) previamente definida.

Exemplo: ^[A-Za-zA-Üä-ü0-9\s]{3,30}\$

^	Marca o inicio da expressão
A-Z	Somente letras de A-Z caixa alta
a-z	Somente letras de a-z caixa baixa
À-Ü	Somente vogais acentuadas caixa alta
à-ü	Somente vogais acentuadas caixa baixa



0-9	Dígitos numéricos
\s	Espaços
{3,30}	Mínimo e máximo de caracteres permitido
\$	Marca do final da expressão

### Exemplo de validação utilizando Regex...

```
package validators;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class ValidatorNome {

    //Método estático para verificar se um nome
    //informado atende
    //ao padrão especificado

    public static void validarNomeProduto(String nome)
        throws Exception{

        //Expressões Regulares (Regex)

        //Passo 1: Criando a expressão
        Pattern regex = Pattern.compile
            ("^ [A-Za-zÀ-Ùà-ü0-9\\s] {3,30} $");

        //Passo 2: Submeter o nome informado
        //à expressão do regex
        Matcher m = regex.matcher(nome);

        //Passo 3: verifico se o regex 'aprovou'
        //o nome informado
        if( ! m.matches() ){ //se não passou na regra
            throw new Exception
                ("Nome do Produto inválido.");
        }

        /*
         *      ^      -> inicio do regex
         *      $      -> final do regex
         *      [ ]    -> caracteres permitidos
         *      { }    -> tamanho minimo e maximo de caracateres
         *      A-Z    -> letras A-Z em caixa alta
         *      a-z    -> letras a-z em caixa baixa
         *      \s     -> permitir espaços
         *      À-Ù    -> vogais acentuadas caixa-alta
         *      à-ü    -> vogais acentuadas caixa-baixa
         *      0-9    -> digitos numericos
        */
    }
}
```



## Relacionamento de Associação Muitos...

Classe Produto...

```
package entity;

public class Produto {

    private Integer idProduto;
    private String nome;
    private Double preco;

    public Produto() {
    }

    public Produto(Integer idProduto, String nome, Double preco) {
        this.idProduto = idProduto;
        this.nome = nome;
        this.preco = preco;
    }

    @Override
    public String toString() {
        return "Produto [idProduto=" + idProduto + ", nome="
               + nome + ", preco=" + preco + "]";
    }

    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Double getPreco() {
        return preco;
    }

    public void setPreco(Double preco) {
        this.preco = preco;
    }
}
```



Classe Venda...

- Venda TEM Muitos Produtos

```
package entity;

import java.util.Date;
import java.util.List;

public class Venda {

    private Integer idVenda;
    private Date dataVenda;
    private Double total;

    private List<Produto> produtos;
    //TEM Muitos Produtos

    public Venda() {
    }

    public Venda(Integer idVenda, Date dataVenda, Double total) {
        super();
        this.idVenda = idVenda;
        this.dataVenda = dataVenda;
        this.total = total;
    }

    @Override
    public String toString() {
        return "Venda [idVenda=" + idVenda + ", dataVenda="
            + dataVenda + ", total=" + total + ", produtos="
            + produtos + "]";
    }

    public Integer getIdVenda() {
        return idVenda;
    }

    public void setIdVenda(Integer idVenda) {
        this.idVenda = idVenda;
    }

    public Date getDataVenda() {
        return dataVenda;
    }

    public void setDataVenda(Date dataVenda) {
        this.dataVenda = dataVenda;
    }
}
```



```
public Double getTotal() {  
  
    total = 0.0;  
  
    for(Produto p : produtos){ //foreach  
        total += p.getPreco(); // += acumulador  
    }  
  
    return total;  
}  
  
public List<Produto> getProdutos() {  
    return produtos;  
}  
  
public void setProdutos(List<Produto> produtos) {  
    this.produtos = produtos;  
}  
}
```

Executando...

```
package main;  
  
import java.util.ArrayList;  
import java.util.Date;  
  
import util.Moeda;  
import entity.Produto;  
import entity.Venda;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Produto p1 = new Produto(1, "Audi", 68000.0);  
        Produto p2 = new Produto(2, "Golf", 90000.0);  
        Produto p3 = new Produto(3, "Astra", 45000.0);  
  
        Venda v = new Venda(1, new Date(), 0.0);  
        v.setProdutos(new ArrayList<Produto>());  
        //inicializando a lista  
  
        v.getProdutos().add(p1);  
        v.getProdutos().add(p2);  
        v.getProdutos().add(p3);  
    }  
}
```



```
System.out.println("Dados da Venda: ");

System.out.println("\tIdVenda...: " + v.getIdVenda());
System.out.println("\tData.....: " + v.getDataVenda());
System.out.println("\tTotal.....: " +
Moeda.formatarMoeda(v.getTotal()));

System.out.println("Produtos contidos na Venda:");

for(Produto p : v.getProdutos()){

    System.out.println("\t" + p);
}

}
```

Resultado...

The screenshot shows the Eclipse IDE interface with the Java - javaAula07b/src/main/Main.java file open. The code prints the details of a sale and its products. The output in the Console view shows the printed data, including the sale ID, date, total, and a list of products with their names, IDs, and prices.

```
Dados da Venda:
IdVenda...: 1
Data.....: Thu May 15 22:26:03 BRT 2014
Total.....: R$ 203.000,00

Produtos contidos na Venda:
Produto [idProduto=1, nome=Audi, preco=68000.0]
Produto [idProduto=2, nome=Golf, preco=90000.0]
Produto [idProduto=3, nome=Astra, preco=45000.0]
```

Saída...

Dados da Venda:

```
IdVenda...: 1
Data.....: Thu May 15 22:26:03 BRT 2014
Total.....: R$ 203.000,00
```

Produtos contidos na Venda:

```
Produto [idProduto=1, nome=Audi, preco=68000.0]
Produto [idProduto=2, nome=Golf, preco=90000.0]
Produto [idProduto=3, nome=Astra, preco=45000.0]
```



## Note que...

Assim como instanciamos o objeto Venda, também foi necessário inicializar a lista de Produtos contida em venda...

```
Venda v = new Venda(1, new Date(), 0.0);
v.setProdutos(new ArrayList<Produto>());
```

Assim podemos utilizar a lista (já inicializada) de forma a adicionar objetos dentro dela...

```
v.getProdutos().add(p1);
v.getProdutos().add(p2);
v.getProdutos().add(p3);
```

Ao imprimir o total da venda, o fizemos formatando o valor double obtido pelo método getTotal() utilizando o método static da Classe Moeda...

```
Moeda.formatarMoeda(v.getTotal())
```

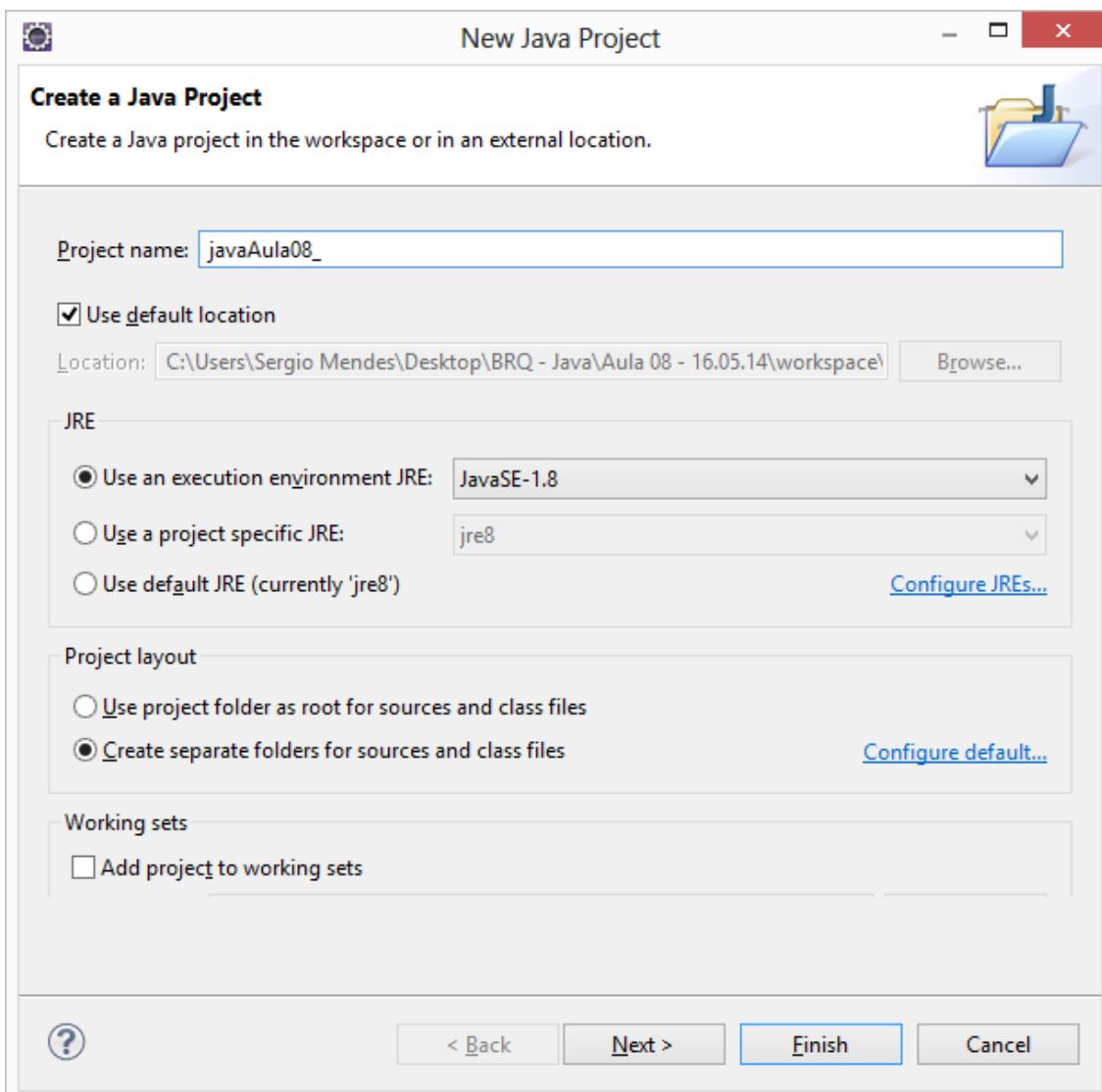
```
System.out.println("Dados da Venda: ");
System.out.println("\tIdVenda: " + v.getIdVenda());
System.out.println("\tData....: " + v.getDataVenda());
System.out.println("\tTotal...: " + Moeda.formatarMoeda(v.getTotal()));
```

- Saída...

Dados da Venda:  
IdVenda....: 1  
Data.....: Thu May 15 22:26:03 BRT 2014  
Total.....: R\$ 203.000,00



### Criando o Projeto...



### Criando o JavaBean Pessoa

```
package entity;

public class Pessoa implements Comparable<Pessoa>{

    private Integer idPessoa;
    private String nome;
    private String email;

    public Pessoa() {
        //Construtor default
    }

    public Pessoa(Integer idPessoa, String nome, String email) {
        super();
    }
}
```



```
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.email = email;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "Pessoa [idPessoa=" + idPessoa + ", nome="
               + nome + ", email=" + email + "]";
    }

    @Override
    public boolean equals(Object obj) {

        if(obj instanceof Pessoa){

            Pessoa p = (Pessoa) obj;

            if(p.getIdPessoa().equals(idPessoa)){
                return true;
            }
        }

        return false;
    }
}
```



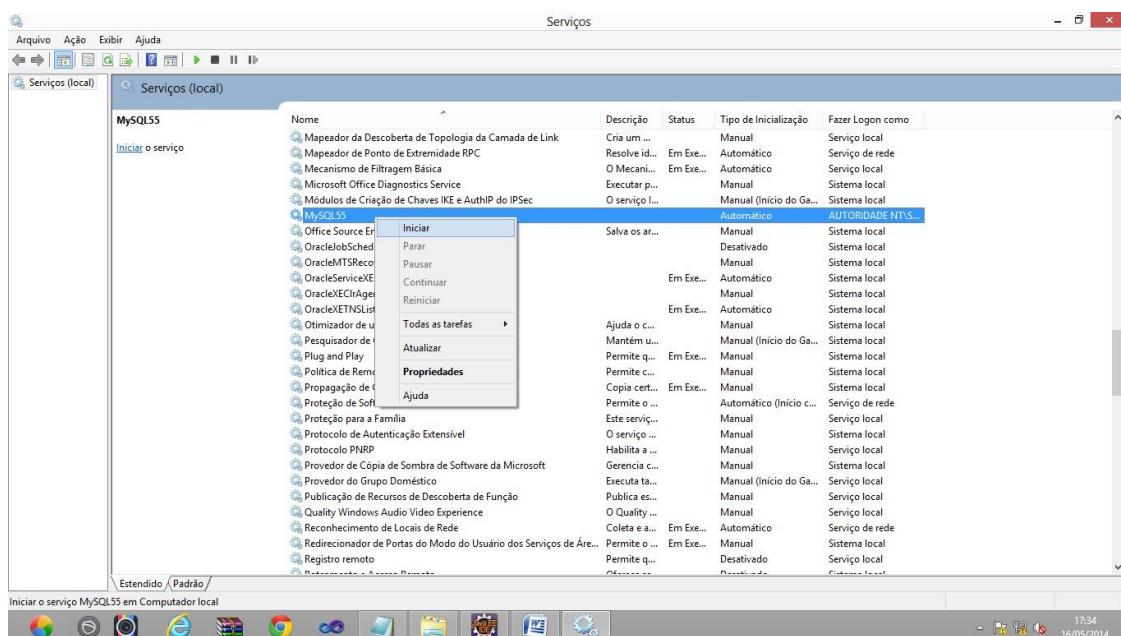
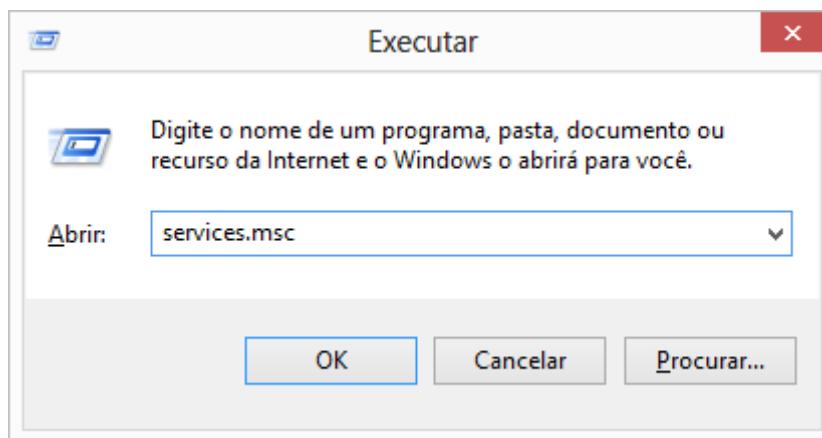
```
@Override
public int hashCode() {
    return idPessoa.hashCode();
}

@Override
public int compareTo(Pessoa p) {
    return idPessoa.compareTo(p.getIdPessoa());
}

}
```

## Acessando o MySql

- Iniciar o serviço do MySQL





Criando o Script da base de dados...

```
drop database if exists aula;
create database aula;
use aula;

create table pessoa(
    idpessoa      integer          auto_increment,
    nome           varchar(50)      not null,
    email          varchar(50)      not null unique,
    primary key(idpessoa));

show tables;

desc pessoa;

insert into pessoa values(null, 'Sergio', 'sergio@gmail.com');

select * from pessoa;

update pessoa
set
    nome = 'Sergio Mendes',
    email = 'sergio.coti@gmail.com'
where
    idpessoa = 1;

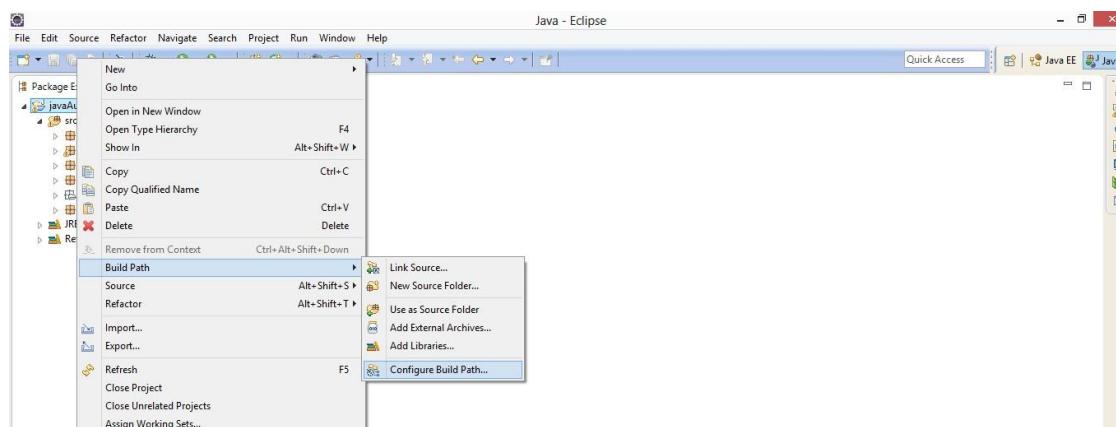
select * from pessoa;

delete from pessoa where idpessoa = 1;
select * from pessoa;
```

## Padrão DAO - Data Access Object

Utilizado em aplicações Java que acessam e manipulam bases de dados.

Passo 1: Adicionar no projeto o Driver para conexão da aplicação com o MySQL...





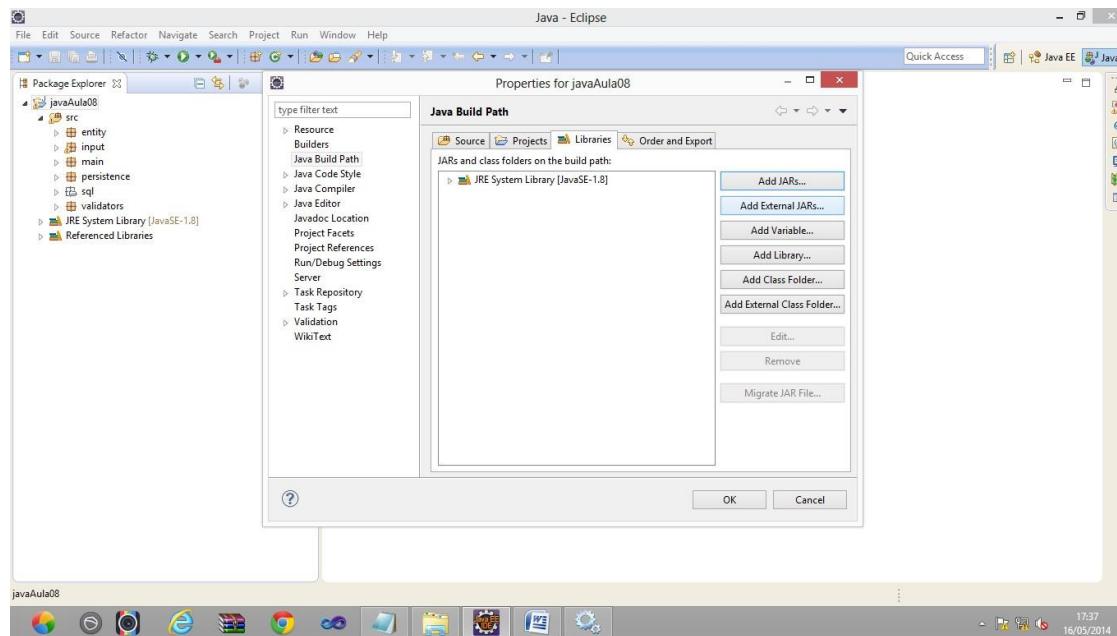
# Java WebDeveloper - BRQ

Sexta-feira, 16 de Maio de 2014

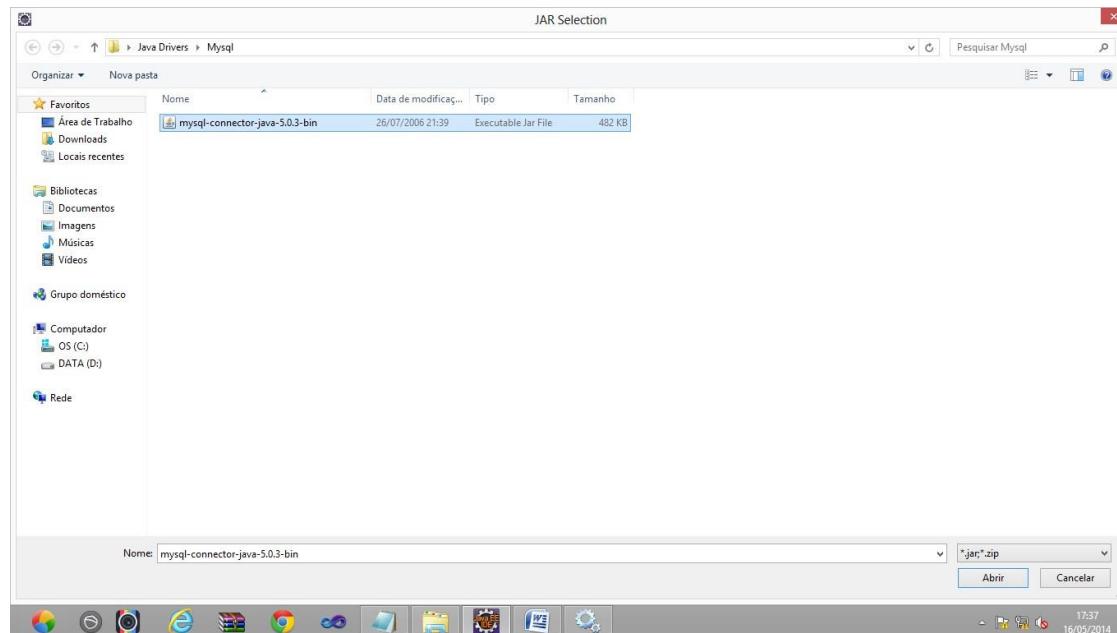
Acesso a banco de dados. Padrão DAO - Data Access Object.

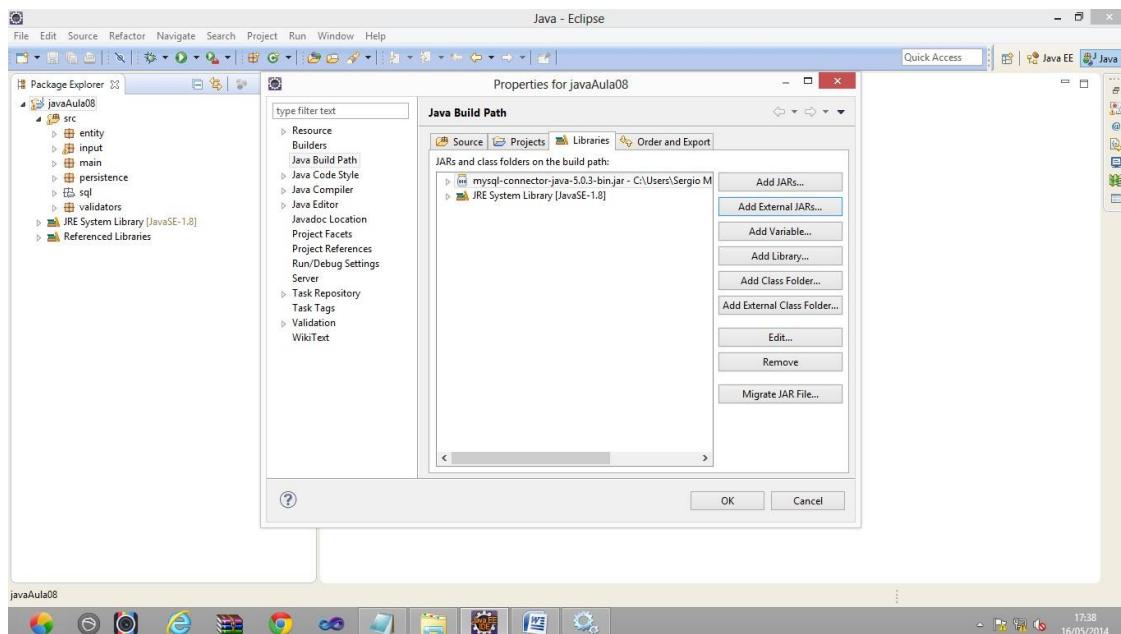
Aula  
08

## Add External Jar



Selecione o arquivo .jar referente ao Driver de conexão do MySQL





### Parte 1:

#### Criando a Classe Dao para conexão com o MySQL

```
package persistence;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

//Data Access Object
//Objeto Java para conexão com banco de dados
public class Dao {

    //Atributos

    //Realizar conexão com a base de dados
    //(abrir e fechar conexão)
    protected Connection con;

    //Executar comandos em linguagem SQL na base de dados
    protected PreparedStatement stmt;

    //Ler e resgatar dados de consultas feitas na base
    protected ResultSet rs;

    //Executar stored procedures na base de dados (futuro...)
    protected CallableStatement call;
```



```
//Método para abrir uma conexão com a base de dados
protected void openConnect() throws Exception{

    //Caminho do Driver do MySql contido no .jar
    Class.forName("com.mysql.jdbc.Driver");
    //Classe dentro do .jar

    con = DriverManager.getConnection
        ("jdbc:mysql://localhost:3306/aula", "root", "");

}

//Método para fechar conexão com a base de dados
protected void closeConnect() throws Exception{
    con.close(); //conexão fechada!
}
```

### Parte 2:

Criando a Classe PessoaDao para realização das operações de CRUD  
(Create, Read, Update e Delete)

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Pessoa;

public class PessoaDao extends Dao{

    //Método para gravar um registro de Pessoa na base de dados
    public void create(Pessoa p) throws Exception{

        openConnect(); //abrir conexão

        stmt = con.prepareStatement("insert into pessoa
                                    values(null, ?, ?)");
        stmt.setString(1, p.getNome());
        stmt.setString(2, p.getEmail());
        stmt.execute(); //executar

        closeConnect(); //fechar conexão
    }

    //Método para atualizar registro de Pessoa na base de dados
    public void update(Pessoa p) throws Exception{

        openConnect();

        stmt = con.prepareStatement("update pessoa set

```



```
        nome=?, email=? where idpessoa=?");
stmt.setString(1, p.getNome());
stmt.setString(2, p.getEmail());
stmt.setInt(3, p.getIdPessoa());
stmt.execute(); //executar

closeConnect();
}

//Método para excluir registro de Pessoa
public void delete(Integer idPessoa) throws Exception{

openConnect();

stmt = con.prepareStatement("delete from pessoa
                           where idpessoa = ?");
stmt.setInt(1, idPessoa);
stmt.execute(); //executar

closeConnect();
}

//Método para obter 1 registro de pessoa pelo id
public Pessoa findById(Integer idPessoa) throws Exception{

openConnect();

stmt = con.prepareStatement("select * from pessoa
                           where idpessoa = ?");
stmt.setInt(1, idPessoa);
rs = stmt.executeQuery();

Pessoa p = null; //sem espaço de memória

if(rs.next()){ //verifica se o ResultSet
               //obteve algum registro

p = new Pessoa( rs.getInt("idpessoa"),
                rs.getString("nome"), rs.getString("email") );

}

closeConnect();

//retornando o objeto Pessoa
return p;
}

//Método para retornar todos os registros da tabela pessoa
public List<Pessoa> findAll() throws Exception{

openConnect();
```



```
stmt = con.prepareStatement("select * from pessoa");
rs = stmt.executeQuery();

List<Pessoa> lista = new ArrayList<Pessoa>();
//lista vazia

while(rs.next()){ //enquanto houver registros

    Pessoa p = new Pessoa( rs.getInt("idpessoa"),
    rs.getString("nome"), rs.getString("email") );

    lista.add(p);
    //adicionando pessoa dentro da lista
}

closeConnect();
return lista; //retornar a lista
}

}
```

### Classe para leitura dos dados...

```
package input;

import java.util.Scanner;

import validators.PessoaValidator;

public class PessoaInput {

    //Método para ler o id da Pessoa informado
    public Integer lerIdPessoa() throws Exception{

        Scanner s = new Scanner(System.in);

        System.out.print("Informe o Id.....: ");
        return s.nextInt();
    }

    //Método para ler o nome
    public String lerNome() throws Exception{

        Scanner s = new Scanner(System.in);

        System.out.print("Informe o Nome....: ");
        return s.nextLine();
    }
}
```



```
//Método para ler o email da Pessoa
public String lerEmail() throws Exception{

    Scanner s = new Scanner(System.in);

    System.out.print("Informe o Email...: ");
    return s.nextLine();
}

}
```

### Executando...

```
package main;

import input.PessoaInput;
import persistence.PessoaDao;
import entity.Pessoa;

public class Main {

    public static void main(String[] args) {

        try{

            PessoaInput in = new PessoaInput();
            Pessoa p = new Pessoa();

            //p.setIdPessoa(in.lerIdPessoa());
            p.setNome(in.lerNome());
            p.setEmail(in.lerEmail());

            PessoaDao d = new PessoaDao();
            d.create(p);

            System.out.println("Dados gravados");

            //
            d.update(p);

            //
            //PessoaDao d = new PessoaDao();
            //PessoaInput in = new PessoaInput();

            //System.out.println("Registro -> "
            //                  + d.findById( in.lerIdPessoa() ));

            //
            //d.delete(in.lerIdPessoa());
            //System.out.println("Dados excluidos com
            //                  sucesso.");
            //

            //for(Pessoa p : d.findAll()){
            //    System.out.println("Registro -> " + p);
            }

        }
    }
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 16 de Maio de 2014

Acesso a banco de dados. Padrão DAO - Data Access Object.

Aula  
08

```
//\n}\n\n}\n\n    }\n\n    catch(Exception e){\n        System.out.println("Erro -> " + e.getMessage());\n    }\n\n}\n\n}
```

## Resultado...

```
Java - javaAula08/src/main/Main.java - Eclipse\nFile Edit Source Refactor Navigate Search Project Run Window Help\n\nPackage Explorer Problems Javadoc Declaration Console Data Source Explor... SQL Results\njavaAula08\nsrc\nentity\nPessoa.java\ninput\nPessoainput.java\nmain\nMain.java\npersistence\nDao.java\nPessoado.java\nsql\nvalidators\nPessoavalidator.java\nJRE System Library [JavaSE-1.8]\nReferenced Libraries\n\n1 package main;\n2\n3 import input.Pessoainput;\n4\n5\n6\n7\n8 public class Main {\n9\n10     public static void main(String[] args) {\n11\n12         try{\n13\n14             Pessoainput in = new Pessoainput();\n15             Pessoa p = new Pessoa();\n16\n17             //p.setidPessoa(in.lerIdPessoa());\n18             p.setNome(in.lerNome());\n19             p.setEmail(in.lerEmail());\n20\n21             Pessoado d = new Pessoado();\n22             d.create(p);\n23\n24             System.out.println("Dados gravados");\n25\n26         } // update(p);\n27\n28         //\n29         Pessoado d = new Pessoado();\n30         //Pessoainput in = new Pessoainput();\n31\n32         //System.out.println("Registro -> " + d.findById( in.lerIdPessoa() ));\n33\n34     } //\n35 }
```

## Classe para validação dos dados de Pessoa...

```
package validators;\n\nimport java.util.regex.Matcher;\nimport java.util.regex.Pattern;\n\npublic class Pessoavalidator {\n\n    public static String validarNome(String nome)\n        throws Exception{\n\n        Pattern regex = Pattern.compile\n            ("^ [A-Za-zÀ-Ùà-ü\\s] {3,30} $");\n\n        Matcher m = regex.matcher(nome);\n\n        if( ! m.matches() ){\n            throw new Exception("Nome Invalido!");\n        }\n    }\n}
```



```
        else{
            return nome;
        }
    }

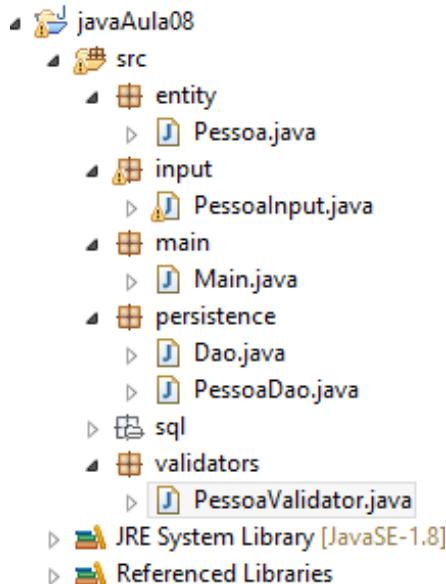
    public static String validarEmail(String email)
throws Exception{

    Pattern regex = Pattern.compile
    ("^@[a-z0-9-_.]+@[a-z.]{6,30}$");

    Matcher m = regex.matcher(email);

    if( ! m.matches() ){
        throw new Exception("Email inválido");
    }
    else{
        return email;
    }
}
}
```

Estrutura do projeto...





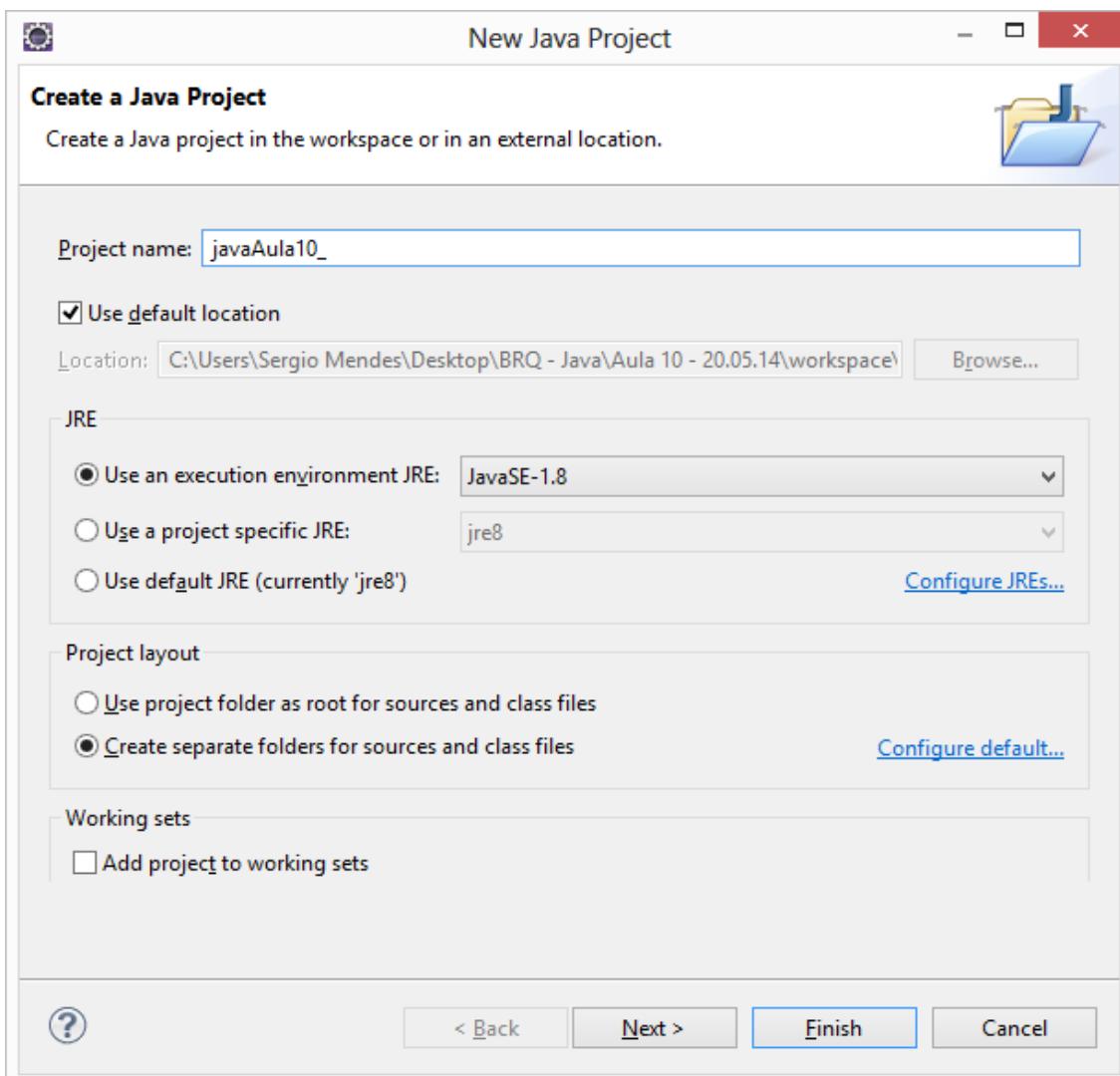
# Java WebDeveloper - BRQ

Terça-feira, 20 de Maio de 2014

Acesso a banco de dados. Padrão DAO - Data Access Object.  
Modelagem Um para Muitos. Threads e Concorrência

Aula  
**10**

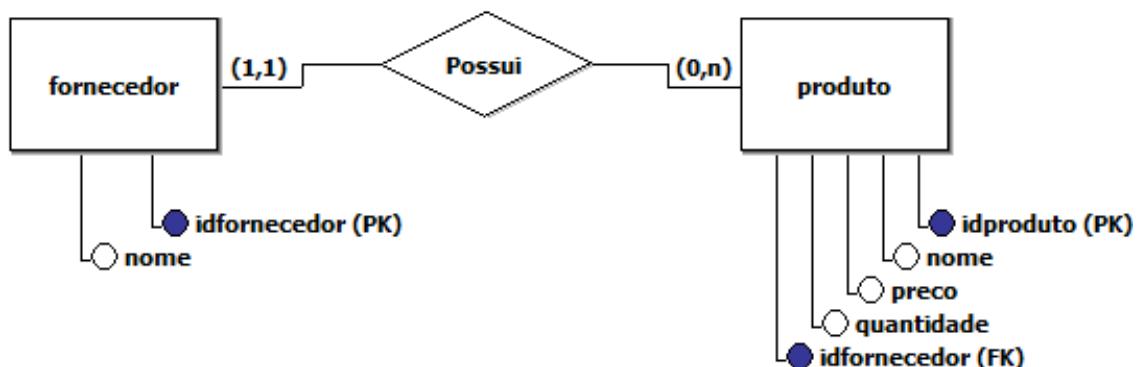
Criando o Projeto...



## Modelagem do banco de dados

Diagrama Entidade Relacionamento - DER

**DER - Diagrama Entidade Relacionamento**





Criando o Script da base de dados...

```
drop database if exists aula10;
create database aula10;
use aula10;

create table fornecedor(
    idfornecedor      integer          auto_increment,
    nome              varchar(50)       not null,
    primary key(idfornecedor));

create table produto(
    idproduto         integer          auto_increment,
    nome              varchar(50)       not null,
    preco             double           not null,
    quantidade        integer          not null,
    idfornecedor      integer          not null,
    primary key(idproduto),
    foreign key(idfornecedor) references
fornecedor(idfornecedor));

show tables;

desc fornecedor;
desc produto;
```

-----

Executando no MySql...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> desc fornecedor;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+
| idfornecedor | int(11) | NO   | PRI  | NULL    | auto_increment |
| nome          | varchar(50) | NO   | PRI  | NULL    |                |
+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)

mysql> desc produto;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+
| idproduto | int(11) | NO   | PRI  | NULL    | auto_increment |
| nome      | varchar(50) | NO   | PRI  | NULL    |                |
| preco     | double   | NO   |       | NULL    |                |
| quantidade | int(11) | NO   |       | NULL    |                |
| idfornecedor | int(11) | NO   | MUL  | NULL    |                |
+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

mysql>
```



### Criando as Classes JavaBean

Relacionamento de Associação 1 para Muitos

```
package entity;

import java.util.List;

public class Fornecedor {

    private Integer idFornecedor;
    private String nome;
    private List<Produto> produtos;

    public Fornecedor() {
    }

    public Fornecedor(Integer idFornecedor, String nome) {
        super();
        this.idFornecedor = idFornecedor;
        this.nome = nome;
    }

    @Override
    public String toString() {
        return "Fornecedor [idFornecedor=" + idFornecedor
               + ", nome=" + nome + "]";
    }

    public Integer getIdFornecedor() {
        return idFornecedor;
    }

    public void setIdFornecedor(Integer idFornecedor) {
        this.idFornecedor = idFornecedor;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public List<Produto> getProdutos() {
        return produtos;
    }
}
```



```
public void setProdutos(List<Produto> produtos) {
    this.produtos = produtos;
}

package entity;

public class Produto {

    private Integer idProduto;
    private String nome;
    private Double preco;
    private Integer quantidade;
    private Fornecedor fornecedor;

    public Produto() {
    }

    public Produto(Integer idProduto, String nome,
                   Double preco, Integer quantidade) {
        super();
        this.idProduto = idProduto;
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
    }

    @Override
    public String toString() {
        return "Produto [idProduto=" + idProduto
               + ", nome=" + nome + ", preco=" + preco +",
               quantidade=" + quantidade + "]";
    }

    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }
}
```



# Java WebDeveloper - BRQ

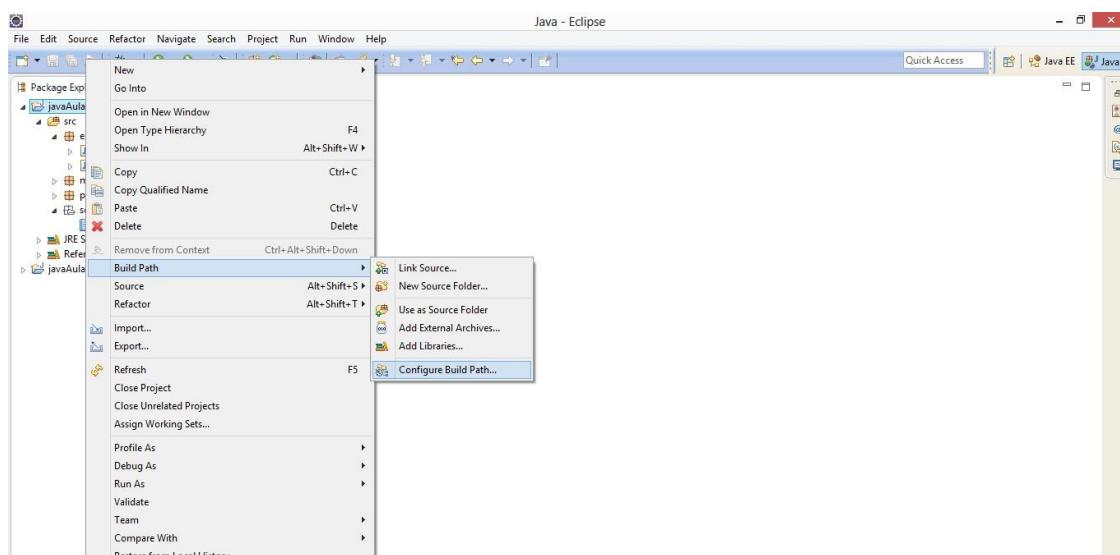
Terça-feira, 20 de Maio de 2014

Acesso a banco de dados. Padrão DAO - Data Access Object.  
Modelagem Um para Muitos. Threads e Concorrência

Aula  
**10**

```
public void setNome(String nome) {  
    this.nome = nome;  
}  
  
public Double getPreco() {  
    return preco;  
}  
  
public void setPreco(Double preco) {  
    this.preco = preco;  
}  
  
public Integer getQuantidade() {  
    return quantidade;  
}  
  
public void setQuantidade(Integer quantidade) {  
    this.quantidade = quantidade;  
}  
  
public Fornecedor getFornecedor() {  
    return fornecedor;  
}  
  
public void setFornecedor(Fornecedor fornecedor) {  
    this.fornecedor = fornecedor;  
}  
}
```

Adicionando o driver do MySql





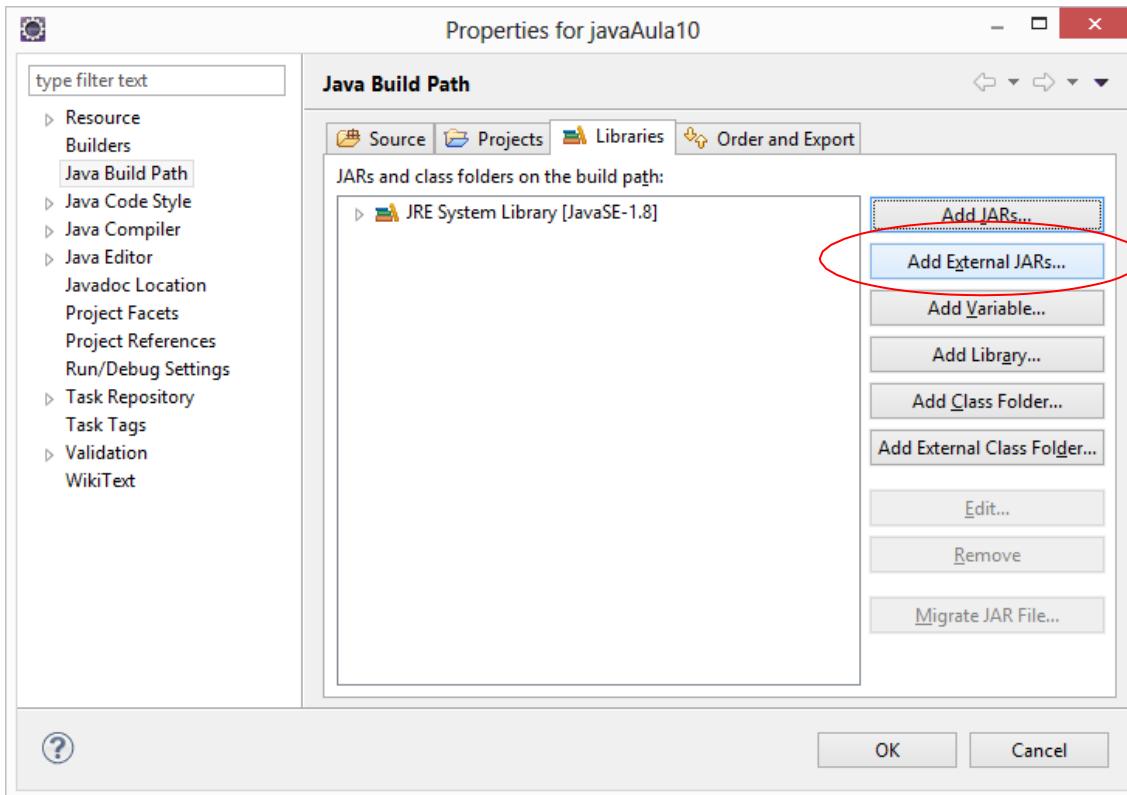
# Java WebDeveloper - BRQ

Terça-feira, 20 de Maio de 2014

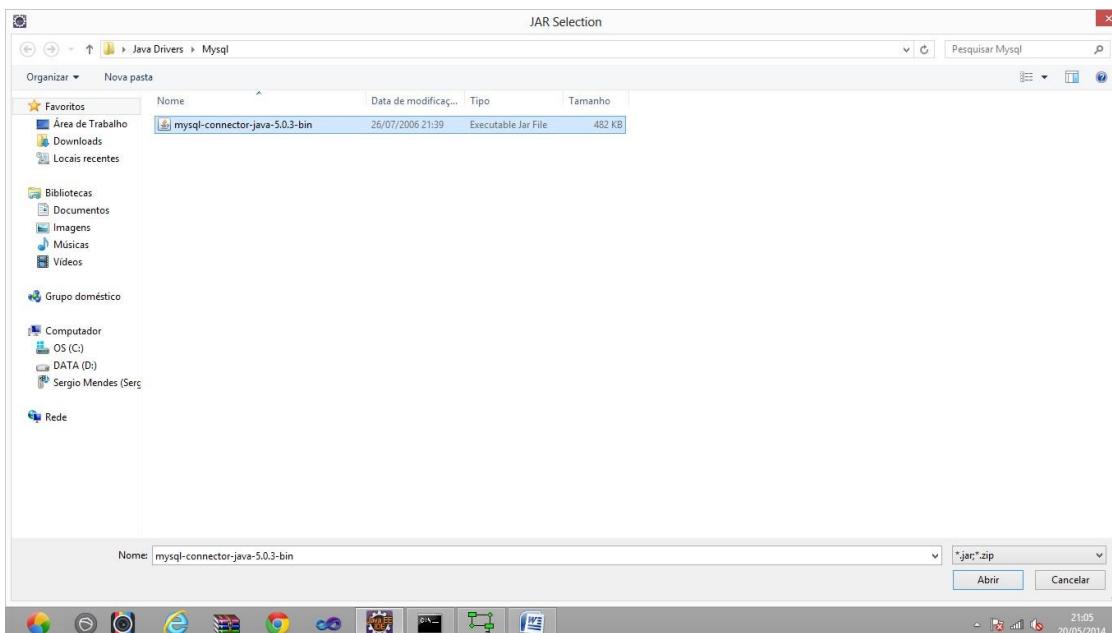
Acesso a banco de dados. Padrão DAO - Data Access Object.  
Modelagem Um para Muitos. Threads e Concorrência

Aula  
**10**

## Add External JARs



**Adicionando o arquivo**  
mysql-connector-java-5.0.1.jar





### Criando a Classe Dao

#### Data Access Object

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

//JDBC -> Java Database Connectivity
public class Dao {

    //Atributos -> protected
    protected Connection con; //conexão com o banco de dados
    protected PreparedStatement stmt; //executar comandos sql
    protected ResultSet rs; //ler registros de consultas

    //método para abrir conexão com o banco de dados
    protected void open() throws Exception{

        //Carregar o Driver do MySql
        Class.forName("com.mysql.jdbc.Driver");

        //realizar a conexão
        con = DriverManager.getConnection
        ("jdbc:mysql://localhost:3306/aula10", "root", "");
    }

    //método para fechar conexão com o banco de dados
    protected void close() throws Exception{
        con.close();
    }
}
```

### Criando a Classe FornecedorDao...

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Fornecedor;

public class FornecedorDao extends Dao{

    public void create(Fornecedor f) throws Exception{
        open();

        stmt = con.prepareStatement("insert into fornecedor
                                    values(null, ?)");
    }
}
```



# Java WebDeveloper - BRQ

Terça-feira, 20 de Maio de 2014

Acesso a banco de dados. Padrão DAO - Data Access Object.  
Modelagem Um para Muitos. Threads e Concorrência

Aula  
**10**

```
stmt.setString(1, f.getNome());
stmt.execute();

close();
}

public void delete(Integer idFornecedor) throws Exception{

open();

stmt = con.prepareStatement("delete from fornecedor
                           where idfornecedor = ?");
stmt.setInt(1, idFornecedor);
stmt.execute();

close();
}

public void update(Fornecedor f) throws Exception{

open();

stmt = con.prepareStatement("update fornecedor set nome=?"
                           "where idfornecedor=?");
stmt.setString(1, f.getNome());
stmt.setInt(2, f.getIdFornecedor());
stmt.execute();

close();
}

public Fornecedor findById(Integer idFornecedor)
throws Exception{

open();

stmt = con.prepareStatement("select * from fornecedor
                           where idfornecedor = ?");
stmt.setInt(1, idFornecedor);
rs = stmt.executeQuery();

Fornecedor f = null;

if(rs.next()){

    f = new Fornecedor( rs.getInt("idfornecedor"),
                        rs.getString("nome"));

}

close();

return f;
}

public List<Fornecedor> findAll() throws Exception{
```



```
open();

stmt = con.prepareStatement("select * from fornecedor");
rs = stmt.executeQuery();

List<Fornecedor> lista = new ArrayList<Fornecedor>();

while(rs.next()){

    Fornecedor f = new Fornecedor(
        rs.getInt("idfornecedor"), rs.getString("nome"));

    lista.add(f);
}

close();
return lista;
}
```

### Criando a Classe ProdutoDao...

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Produto;

public class ProdutoDao extends Dao{

    //Método para cadastrar produto na base
    public void create(Produto p) throws Exception{

        open();

        stmt = con.prepareStatement("insert into produto
                                    values(null, ?, ?, ?, ?)");

        stmt.setString(1, p.getNome());
        stmt.setDouble(2, p.getPreco());
        stmt.setInt(3, p.getQuantidade());
        stmt.setInt(4, p.getFornecedor().getIdFornecedor());

        stmt.execute(); //executar

        close();
    }

    //Método para excluir o produto na base
}
```



```
public void delete(Integer idProduto) throws Exception{  
  
    open();  
  
    stmt = con.prepareStatement("delete from produto  
                               where idproduto = ?");  
    stmt.setInt(1, idProduto);  
    stmt.execute(); //executar  
  
    close();  
}  
  
//Método para atualizar um registro de produto na base  
public void update(Produto p) throws Exception{  
  
    open();  
  
    stmt = con.prepareStatement("update produto set  
                               nome=?, preco=?, quantidade=?,  
                               idfornecedor=? where idproduto=?");  
  
    stmt.setString(1, p.getNome());  
    stmt.setDouble(2, p.getPreco());  
    stmt.setInt(3, p.getQuantidade());  
    stmt.setInt(4, p.getFornecedor().getIdFornecedor());  
    stmt.setInt(5, p.getIdProduto());  
  
    stmt.execute();  
  
    close();  
}  
  
//Método para buscar 1 Produto pelo id (chave primária)  
public Produto findById(Integer idProduto)  
throws Exception{  
  
    open();  
  
    stmt = con.prepareStatement("select * from produto  
                               where idproduto = ?");  
    stmt.setInt(1, idProduto);  
    rs = stmt.executeQuery();  
    //executa a consulta e retorna os registros  
  
    Produto p = null; //vazio!  
  
    if(rs.next()){//se o ResultSet obteve algum registro  
  
        p = new Produto( rs.getInt("idproduto"),  
                        rs.getString("nome"), rs.getDouble("preco"),  
                        rs.getInt("quantidade"));  
    }  
  
    close();  
}
```



```
        return p; //retornar Produto
    }

    //Método para retornar todos os registros da tabela produto
    public List<Produto> findAll() throws Exception{

        open();

        stmt = con.prepareStatement("select * from produto");
        rs = stmt.executeQuery();
        //executo a consulta e leio os registros

        List<Produto> lista = new ArrayList<Produto>();
        //lista sem elementos

        while(rs.next()){
        //enquanto o ResultSet tiver registros

            Produto p = new Produto
            ( rs.getInt("idproduto"), rs.getString("nome"),
            rs.getDouble("preco"),
            rs.getInt("quantidade"));

            lista.add(p); //adiciono produto dentro da lista
        }

        close();
        return lista;
    }

}
```

### Testando a gravação dos dados do fornecedor...

```
package main;

import persistence.FornecedorDao;
import persistence.ProdutoDao;
import entity.Fornecedor;
import entity.Produto;

public class Main {

    public static void main(String[] args) {

        try{

            Fornecedor f = new Fornecedor();
            f.setNome("Loja A");

            FornecedorDao d = new FornecedorDao();
            d.create(f);
        }
    }
}
```



# Java WebDeveloper - BRQ

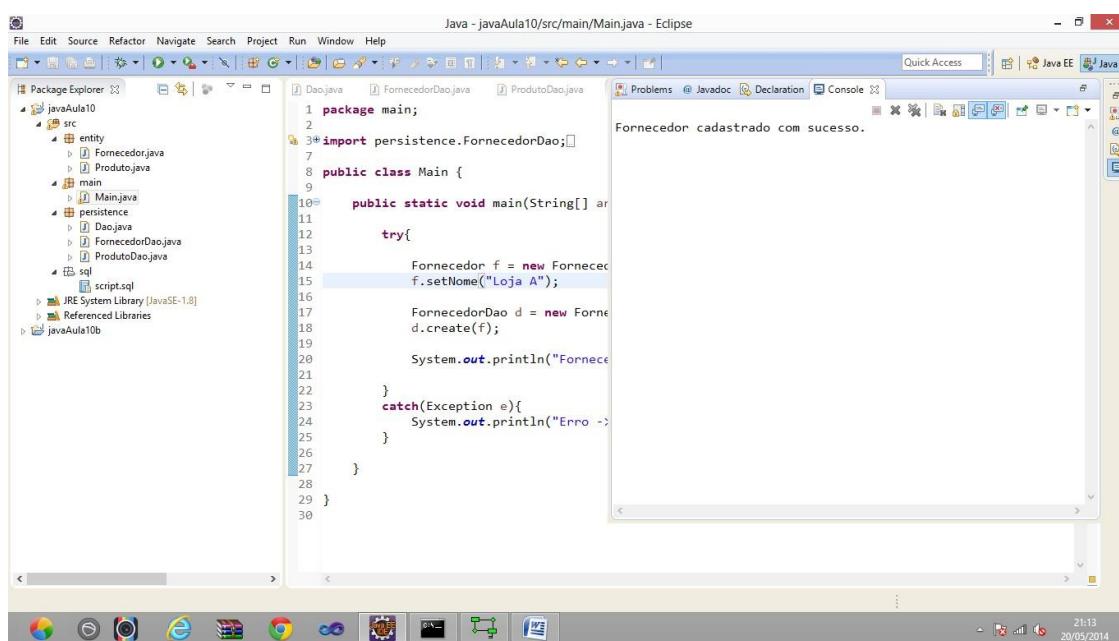
Terça-feira, 20 de Maio de 2014

Acesso a banco de dados. Padrão DAO - Data Access Object.  
Modelagem Um para Muitos. Threads e Concorrência

Aula  
**10**

```
        System.out.println("Fornecedor cadastrado  
        com sucesso.");  
  
    }  
    catch(Exception e){  
        System.out.println("Erro -> " + e.getMessage());  
    }  
}  
}
```

Executando...



Na base de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p  
mysql> select * from fornecedor;  
+-----+  
| idfornecedor | nome   |  
+-----+  
|          1 | Loja A |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Testando a gravação dos dados do produto...



```
package main;

import persistence.ProdutoDao;
import entity.Fornecedor;
import entity.Produto;

public class Main {

    public static void main(String[] args) {

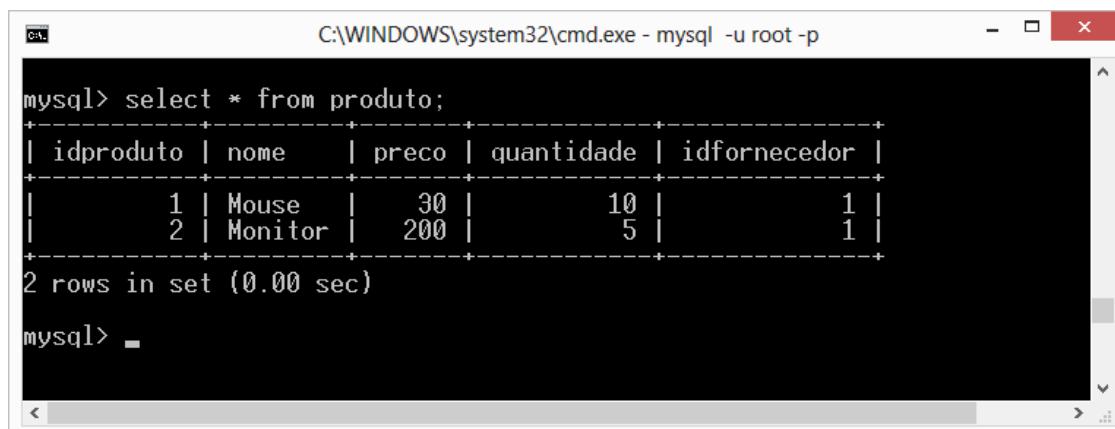
        try{
            Fornecedor f = new Fornecedor();
            f.setIdFornecedor(1);

            Produto p = new Produto();
            p.setNome("Mouse");
            p.setPreco(30.0);
            p.setQuantidade(10);
            p.setFornecedor(f); //relacionei!!

            ProdutoDao d = new ProdutoDao();
            d.create(p); //gravação

            System.out.println("Produto cadastrado
                               com sucesso.");
        }
        catch(Exception e){
            System.out.println("Erro -> " + e.getMessage());
        }
    }
}
```

Na base de dados...



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> select * from produto;
+-----+-----+-----+-----+
| idproduto | nome   | preco | quantidade | idfornecedor |
+-----+-----+-----+-----+
|      1 | Mouse  |   30 |       10 |           1 |
|      2 | Monitor |  200 |        5 |           1 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```



## Introdução sobre Threads

Diferença entre Thread e Processos: O Processo é um programa em execução, possui seu próprio espaço de endereçamento (Método, Classe, etc..)

Um processo pode ter várias Threads executando de forma concorrente. A JVM do Java permite a execução de várias Threads de forma concorrente, isto é denominado MultiThreading.

Em Java, existem basicamente duas maneiras de se trabalhar com Threads:  
Utilizando a **Classe Thread** ou a **Interface Runnable**

- Hernando a Classe Thread...

```
package estudo;

//A primeira forma de criarmos uma Thread em Java
//é herdando a Classe Thread
public class Processo01 extends Thread{

    //Threads (execução de processos paralelos)
    //toda Classe que herda Thread por boa prática deverá ter um
    //método sobrescrito chamado 'run'
    //Este método run será executado quando o processo for iniciado

    @Override
    public void run() { //Dispara o processo

        //Tudo que colocamos dentro do método run é executado
        //toda vez que a thread for iniciada
        try{
            for(int i = 0; i <= 20; i++){

                sleep( (long) (Math.random() * 1000) );
                //dormir 2 seg
                System.out.println("Executando Thread->" + i);

                if(i == 20){
                    System.out.println("TERMINEI!!!!");
                }
            }
        } catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```



# Java WebDeveloper - BRQ

Terça-feira, 20 de Maio de 2014

Acesso a banco de dados. Padrão DAO - Data Access Object.  
Modelagem Um para Muitos. Threads e Concorrência

Aula  
**10**

Executando na Main...

```
package main;

import estudo.Processo01;

public class Main01 {

    public static void main(String[] args) {

        Processo01 p1 = new Processo01();
        Processo01 p2 = new Processo01();

        //Iniciar o processo!!!
        p1.start(); //executa o run
        p2.start();
    }
}
```

Executando...

```
Java - javaAula10b/src/main/Main01.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Problems Javadoc Declaration Console
javaAula10b Dao.java FornecedorDao.java ProdutoDao.java
src entity
  Fornecedor.java
  Produto.java
main Main.java
  Main.java
persistence
  Dao.java
  FornecedorDao.java
  ProdutoDao.java
sql script.sql
JRE System Library [JavaSE-1.8]
Referenced Libraries
javaAula10b Dao.java FornecedorDao.java ProdutoDao.java
src entity
  Fornecedor.java
  Produto.java
main Main.java
  Main01.java
  Main02.java
  Main03.java
JRE System Library [JavaSE-1.8]

1 package main;
2
3 import estudo.Processo01;
4
5 public class Main01 {
6
7     public static void main(String[] args) {
8         Processo01 p1 = new Processo01();
9         Processo01 p2 = new Processo01();
10
11         //Iniciar o processo!!!
12         p1.start(); //executa o run
13         p2.start();
14
15     }
16
17
18 }
19
20

<terminated> Main01 [Java Application] C:\Program Files (x86)\Java\jre8\bin\javaw.exe (20/05/2014 21:20:33)
Executando a Thread -> 10
Executando a Thread -> 11
Executando a Thread -> 8
Executando a Thread -> 12
Executando a Thread -> 13
Executando a Thread -> 9
Executando a Thread -> 14
Executando a Thread -> 10
Executando a Thread -> 11
Executando a Thread -> 15
Executando a Thread -> 16
Executando a Thread -> 12
Executando a Thread -> 17
Executando a Thread -> 13
Executando a Thread -> 18
Executando a Thread -> 14
Executando a Thread -> 19
Executando a Thread -> 15
Executando a Thread -> 16
Executando a Thread -> 20
TERMINOU!!!!
Executando a Thread -> 17
Executando a Thread -> 18
Executando a Thread -> 19
Executando a Thread -> 20
TERMINOU!!!!
```

Note que a Classe Processo01 realiza 2 execuções do método run em paralelo, ou seja, se forma simultanea.

Quando executamos a linha abaixo, o método **run** da Classe Thread é executado...

**p1.start();**

- Implementando a Interface Runnable...



```
package estudo;

//Outra forma de criar uma Thread é implementando a interface Runnable
public class Processo02 implements Runnable{

    @Override
    public void run() {

        try{

            for(int i = 0; i <= 5; i++){

                Thread.sleep( (long) (Math.random() * 1000) );

                System.out.println("Contando... " + i);
            }
        } catch(Exception e){
            System.out.println("Erro -> " + e.getMessage());
        }
    }
}
```

Executando na Main...

```
package main;

import estudo.Processo02;

public class Main02 {

    public static void main(String[] args) {

        //Uma Classe que implementa Runnable é executada
        //da seguinte maneira...

        Processo02 p1 = new Processo02();
        Processo02 p2 = new Processo02();

        Thread t1 = new Thread(p1);
        Thread t2 = new Thread(p2);

        t1.start();
        t2.start();
    }
}
```

Executando...

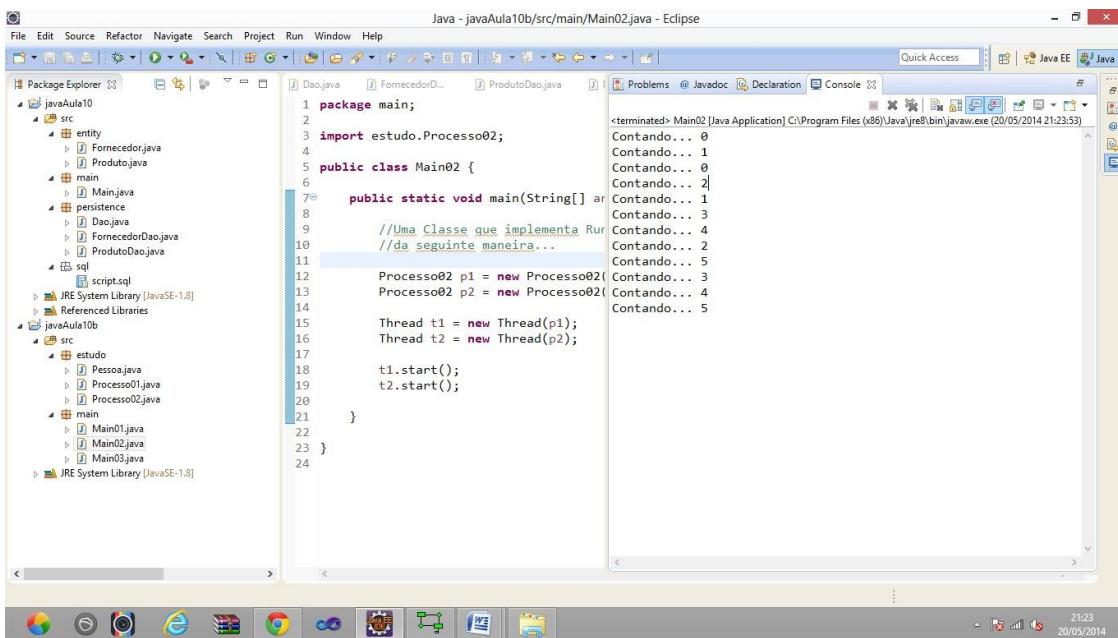


# Java WebDeveloper - BRQ

Terça-feira, 20 de Maio de 2014

Acesso a banco de dados. Padrão DAO - Data Access Object.  
Modelagem Um para Muitos. Threads e Concorrência

Aula  
**10**



## Pausar momentaneamente a execução de uma Thread

Toda Thread possui um método denominado sleep que faz que a Thread "durma" durante um período de milisegundos, interrompendo sua execução temporariamente.

**Thread.sleep(2000);**

## Apêndice:

Criando uma View na base de dados para consultar Produtos e Fornecedores...

```
create view estoque
as
select p.idproduto, p.nome, p.preco, p.quantidade, f.nome as
fornecedor
from produto p inner join fornecedor f
on f.idfornecedor = p.idfornecedor;

select * from estoque;
```



- **Java SE** -> Standard Edition (Local)
- **Java EE** -> Enterprise Edition (Web e aplicações corporativas)
  - Java Web
    - Projetos web
  - Enterprise
    - Projetos corporativos (integram o ambiente web com outros serviços e recursos)
- **Java ME** -> Micro Edition (Mobile)

## 1º Passo: Instalar o servidor (container)

Para executar um projeto java web, é necessário um servidor web (Utilizaremos em aula o Apache Tomcat)

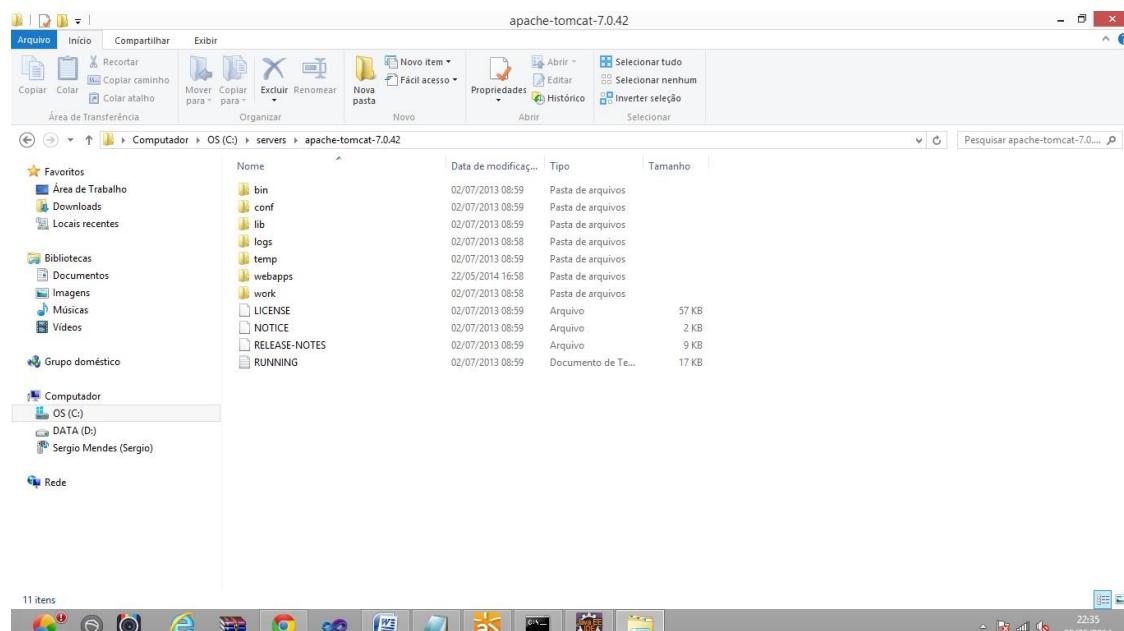
Para executar um projeto java enterprise, já é necessário um servidor de aplicação, como por exemplo: (jboss (comunidade redhat), glassfish (sun), weblogic (oracle), websphere (ibm), etc..)

### -> Apache tomcat (Servidor Java Web)

\*\* Adicionar o tomcat no Eclipse -> apontando para o diretório onde foi extraído o servidor

\*\* Alterar a porta padrão do servidor 8080

C:\servers\apache-tomcat-7.0.42





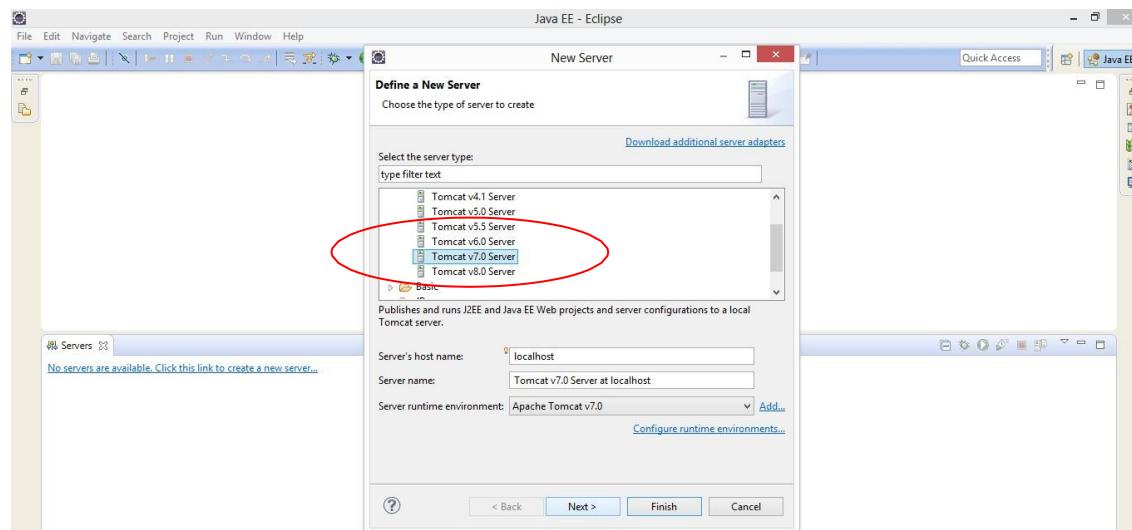
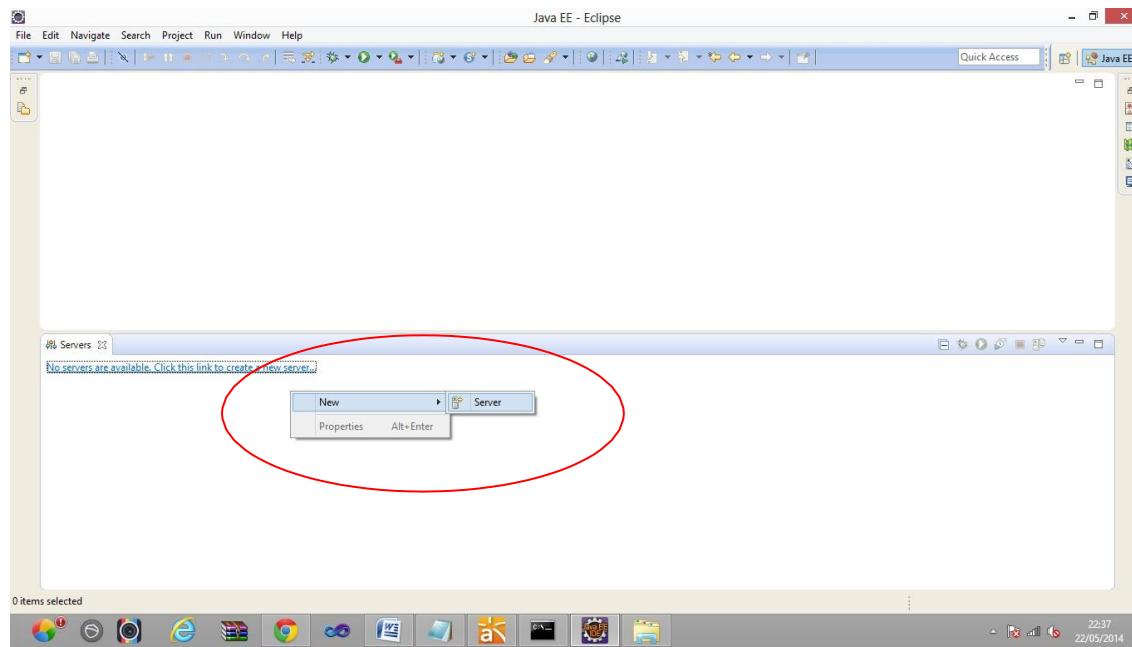
# Java WebDeveloper - BRQ

Quinta-feira, 22 de Maio de 2014

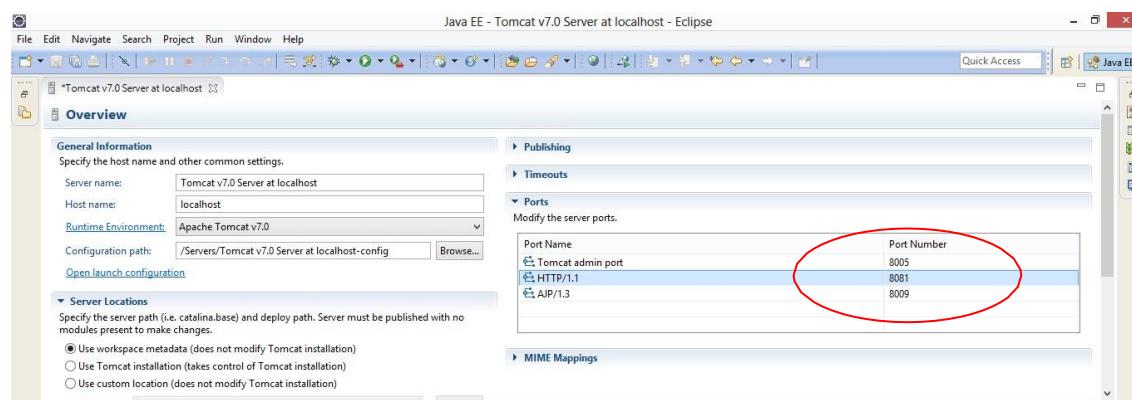
Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC e acesso a banco com DAO (JDBC)

Aula  
11

## Adicionando o Tomcat no Eclipse



## Alterando a porta padrão HTTP do servidor





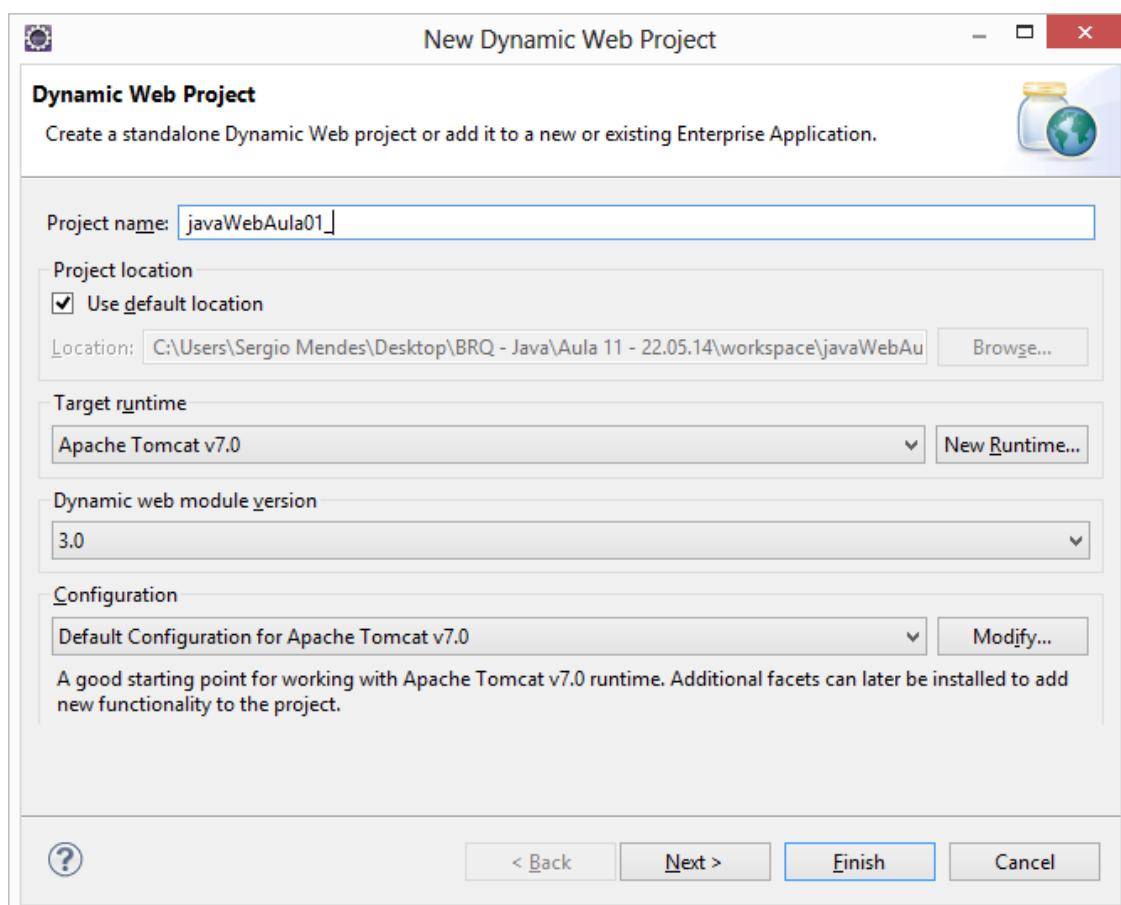
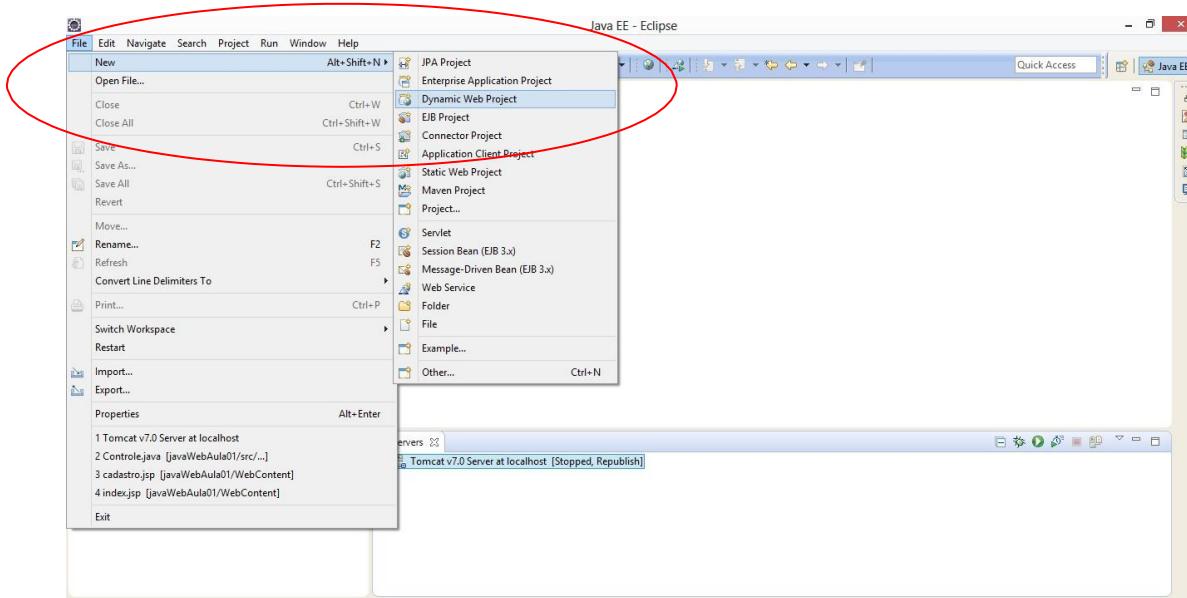
## 2º Passo: Criação do Projeto

### Dynamic Web Project

- Estrutura do Projeto Java Web:

src (pacotes e Classes do projeto)

WebContent (Páginas JSP, css, javascript, html, etc...)





**Tarefa:**

Diagrama de Caso de Uso



- Criando o JavaBean Cliente

```
package entity;

//JavaBean -> Classe Java de entidade
public class Cliente implements Comparable<Cliente>{

    //Atributos
    private Integer idCliente;
    private String nome;
    private String email;

    //Construtor padrão (default)
    public Cliente() {
        //Vazio!
    }

    //Sobrecarga de Construtores (entrada de dados)
    public Cliente(Integer idCliente, String nome, String email) {
        super();
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
    }

    public Integer getIdCliente() {
        return idCliente;
    }

    public void setIdCliente(Integer idCliente) {
        this.idCliente = idCliente;
    }

    public String getNome() {
        return nome;
    }
}
```



# Java WebDeveloper - BRQ

Quinta-feira, 22 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC e acesso a banco com DAO (JDBC)

Aula

11

```
public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

@Override
public String toString() {
    return "Cliente [idCliente=" + idCliente + ", nome="
           + nome + ", email=" + email + "]";
}

@Override
public boolean equals(Object obj) {

    if(obj instanceof Cliente){
        Cliente c = (Cliente) obj;

        if(c.getIdCliente().equals(idCliente)){
            return true;
        }
    }

    return false;
}

@Override
public int hashCode() {
    return idCliente.hashCode();
}

@Override
public int compareTo(Cliente c) {
    return idCliente.compareTo(c.getIdCliente());
}

}
```



Criando o Script da base de dados...

```
drop database if exists aulaweb01;
create database aulaweb01;
use aulaweb01;

create table cliente(
    idcliente      integer          auto_increment,
    nome           varchar(50)       not null,
    email          varchar(50)       not null unique,
    primary key(idcliente));

show tables;

desc cliente;
```

- Executando no MySql...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root
mysql> drop database if exists aulaweb01;
Query OK, 1 row affected (0.46 sec)

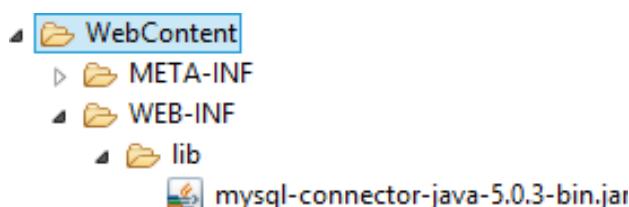
mysql> create database aulaweb01;
Query OK, 1 row affected (0.01 sec)

mysql> use aulaweb01;
Database changed
mysql> create table cliente(
    -> idcliente      integer          auto_increment,
    -> nome           varchar(50)       not null,
    -> email          varchar(50)       not null unique,
    -> primary key(idcliente));
Query OK, 0 rows affected (0.39 sec)

mysql>
mysql> show tables;
+-----+
| Tables_in_aulaweb01 |
+-----+
| cliente             |
+-----+
1 row in set (0.00 sec)
```

## Observação:

Em um projeto java web, os arquivos **.JAR** (bibliotecas) poderão ser incluídos dentro do projeto em uma pasta denominada **WEB-INF/lib**





## Criando a Classe Dao

### Data Access Object

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

    //Conexão com o banco de dados
    protected Connection con;

    //Executar comandos SQL (queries) na base de dados
    protected PreparedStatement stmt;

    //Ler registros obtidos de consultas à base de dados
    protected ResultSet rs;

    //Método para abrir conexão com o banco de dados
    protected void open() throws Exception{

        //Carregar o Driver de Conexão do MySql
        Class.forName("com.mysql.jdbc.Driver");

        //Abrir conexão -> caminho do BD | usuario | senha
        con = DriverManager.getConnection
        ("jdbc:mysql://localhost:3306/aulaweb01", "root", "");

    }

    //Método para fechar conexão com o banco de dados
    protected void close() throws Exception{
        con.close();
    }
}
```

## Criando a Classe de Persistência -> ClienteDao

```
package persistence;

import entity.Cliente;

public class ClienteDao extends Dao{

    //Método para cadastrar um novo cliente na tabela
    //método recebe um objeto do tipo Cliente (JavaBean)
    public void create(Cliente c) throws Exception{

        open(); //conexão aberta
        stmt = con.prepareStatement("insert into cliente
        values(null, ?, ?)");
    }
}
```



```
stmt.setString(1, c.getNome());
stmt.setString(2, c.getEmail());
stmt.execute(); //executar

close(); //conexão fechada
}

}
```

## JSP - Java Server Pages (.jsp)

Formato de páginas web para projetos Java

- Toda página .jsp deverá estar dentro da pasta \WebContent
- Todo projeto Java Web terá uma página inicial denominada [index.jsp]

Criando a página inicial...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>

        <!-- cabeçalho da página (titulo, css, javascript...) -->
        <title>Projeto Java Web 01</title>

    </head>

    <body>

        <!-- conteúdo (corpo) da página -->
        <h3>Projeto Controle de Clientes</h3>
        <hr/>

        Utilize os links abaixo para Manter Clientes
        <br/><br/>

        <a href="cadastro.jsp">Cadastrar Clientes</a>

    </body>

</html>
```



# Java WebDeveloper - BRQ

Quinta-feira, 22 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC e acesso a banco com DAO (JDBC)

Aula

11

Executando...



## Projeto Controle de Clientes

Utilize os links abaixo para Manter Clientes

[Cadastrar Clientes](#)



## Criando a página de Cadastro de Clientes

- Utilizando HTML5

/WebContent/cadastro.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
        <title>Projeto Java Web 01</title>
    </head>

    <body>

        <h3>Cadastro de Clientes</h3>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

        <form name="formcadastro">

            Nome do Cliente: <br/>
            <input type="text" name="nome" required="required"
                pattern="^([A-Za-zÀ-Üà-ü\s]{6,50})$"
                title="Por favor, informe o
                nome do cliente."
                placeholder="Digite o nome aqui" />
```



# Java WebDeveloper - BRQ

Quinta-feira, 22 de Maio de 2014

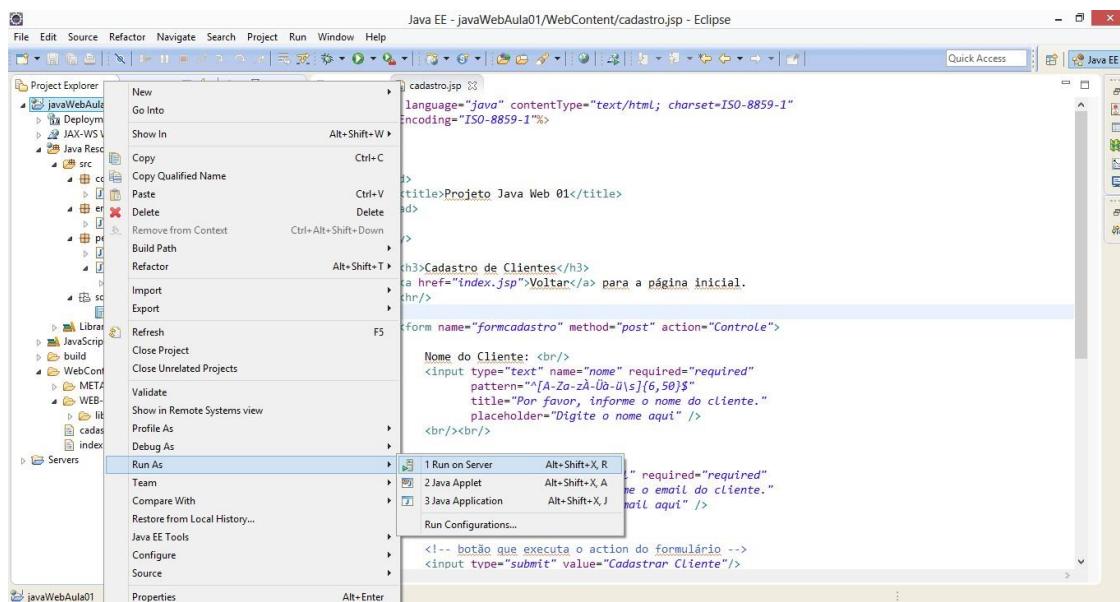
Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC e acesso a banco com DAO (JDBC)

Aula  
11

```
<br/><br/>
Endereço de Email: <br/>
<input type="email" name="email" required="required"
       title="Por favor, informe o email do
       cliente."
       placeholder="Digite o email aqui" />
<br/><br/>

        <!-- botão que executa o action do formulário -->
        <input type="submit" value="Cadastrar Cliente"/>
    </form>
</body>
</html>
```

Executando...



## Cadastro de Clientes

[Voltar para a página inicial.](#)

Nome do Cliente:

Preencha este campo.  
Por favor, informe o nome do cliente.

Endereço de Email:



## Servlets

Classes Java capazes de receber requisições e enviar respostas à páginas Web. O Servlet é um tipo de Classe java capaz de comunicar-se através do protocolo HTTP com outras páginas do projeto.

Todo Servlet pode resgatar dados e enviar respostas para as páginas de 2 maneiras: utilizando os métodos **doGet** e **doPost**

### doGet

Método executado quando a chamada a um servlet é feito utilizando na página. O protocolo GET de envio de dados (dados são enviados pela URL do navegador)

### doPost

Método executado quando a chamada a um servlet é feito utilizando na página. O protocolo POST de envio de dados (dados são enviados por um Formulário)

Os métodos doGet e doPost recebem como parâmetro dois argumentos:

#### HttpServletRequest

Utilizado para resgate dos dados enviados pela página

#### request.getParameter

resgata o valor de um campo/variável enviado pela página

#### request.setAttribute

enviar uma variável de volta para uma página, contendo objetos, dados, mensagens, listas, etc...

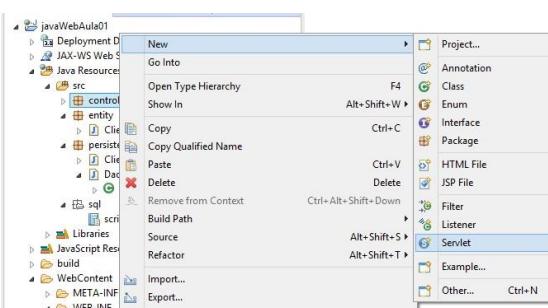
#### request.getRequestDispatcher

redireciona o fluxo de resgate e resposta de um servlet de volta para uma página

#### HttpServletResponse

Utilizado para impressão de conteúdo, redirecionamento, respostas, impressão de conteúdo HTML e etc...

## Criando o Servlet..





```
package control;

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

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 persistence.ClienteDao;
import entity.Cliente;

@WebServlet("/Controle")
public class Controle extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public Controle() {
        super();
    }

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

        PrintWriter out = response.getWriter();
        //objeto para impressão no servlet
        out.println("Método doGet foi executado!!");

    }

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

        //PrintWriter out = response.getWriter();
        //objeto para impressão no servlet
        //out.println("Método doPost foi executado!!");

        try{

            Cliente c = new Cliente(); //JavaBean

            c.setNome(request.getParameter("nome"));
            c.setEmail(request.getParameter("email"));

            ClienteDao d = new ClienteDao();
            d.create(c); //gravar

            //criar uma mensagem que será enviada para a página
            request.setAttribute("msg", "Dados gravados
com sucesso");

            request.setAttribute("cliente", c); //objeto

            //out.println("Dados gravados com sucesso.");
            //out.println("Dados resgatados com sucesso -> "
            + c); //toString
        }
    }
}
```



# Java WebDeveloper - BRQ

Quinta-feira, 22 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC e acesso a banco com DAO (JDBC)

Aula  
**11**

```
        }  
        catch (Exception e) {  
  
            request.setAttribute("msg", "Erro -> "  
                + e.getMessage());  
  
            //out.println("Erro -> " + e.getMessage());  
        }  
        finally{  
  
            //redirecionar o fluxo de volta para a página  
            request.getRequestDispatcher("cadastro.jsp").  
                forward(request, response);  
        }  
    }  
}
```

Fazendo com que o formulário envie os dados preenchidos para o Servlet Controle através do protocolo POST...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
  
<html>  
    <head>  
        <title>Projeto Java Web 01</title>  
    </head>  
  
    <body>  
        <h3>Cadastro de Clientes</h3>  
        <a href="index.jsp">Voltar</a> para a página inicial.  
        <hr/>  
  
        <form name="formcadastro" method="post"  
              action="Controle">  
  
            Nome do Cliente: <br/>  
            <input type="text" name="nome" required="required"  
                  pattern="^([A-Za-zÀ-Üà-ü\s]{6,50})$"  
                  title="Por favor, informe o  
                  nome do cliente."  
                  placeholder="Digite o nome aqui" />  
            <br/><br/>  
  
            Endereço de Email: <br/>  
            <input type="email" name="email" required="required"  
                  title="Por favor, informe o email  
                  do cliente."  
                  placeholder="Digite o email aqui" />  
            <br/><br/>
```



```
<!-- botão que executa o action do formulário -->
<input type="submit" value="Cadastrar Cliente"/>

</form>
</body>
</html>
```

## EL - Expression Languages \${}

Linguagem utilizada em páginas JSP que conseguem resgatar variáveis e dados enviados pelo servidor de forma a exibi-los na página.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <title>Projeto Java Web 01</title>
    </head>
    <body>
        <h3>Cadastro de Clientes</h3>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>
        <form name="formcadastro" method="post" action="Controle">

            Nome do Cliente: <br/>
            <input type="text" name="nome" required="required"
                pattern="^([A-Za-zÀ-Üà-ü\s]{6,50})$"
                title="Por favor, informe o nome
                do cliente."
                placeholder="Digite o nome aqui" />
            <br/><br/>
            Endereço de Email: <br/>
            <input type="email" name="email" required="required"
                title="Por favor, informe o email
                do cliente."
                placeholder="Digite o email aqui" />
            <br/><br/>
            <!-- botão que executa o action do formulário -->
            <input type="submit" value="Cadastrar Cliente"/>
        </form>

        <!-- EL (Expression Language) -->
        <!-- Exibir a mensagem -->
        <h4> ${msg} </h4>

        ${cliente.nome} <br/>
        ${cliente.email} <br/>
    </body>
</html>
```



# Java WebDeveloper - BRQ

Quinta-feira, 22 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC e acesso a banco com DAO (JDBC)

Aula

11

Executando...



## Cadastro de Clientes

[Voltar](#) para a página inicial.

Nome do Cliente:

Sergio Mendes

Endereço de Email:

sergio.coti@gmail.com



## Cadastro de Clientes

[Voltar](#) para a página inicial.

Nome do Cliente:

Digite o nome aqui

Endereço de Email:

Digite o email aqui

## Dados gravados com sucesso

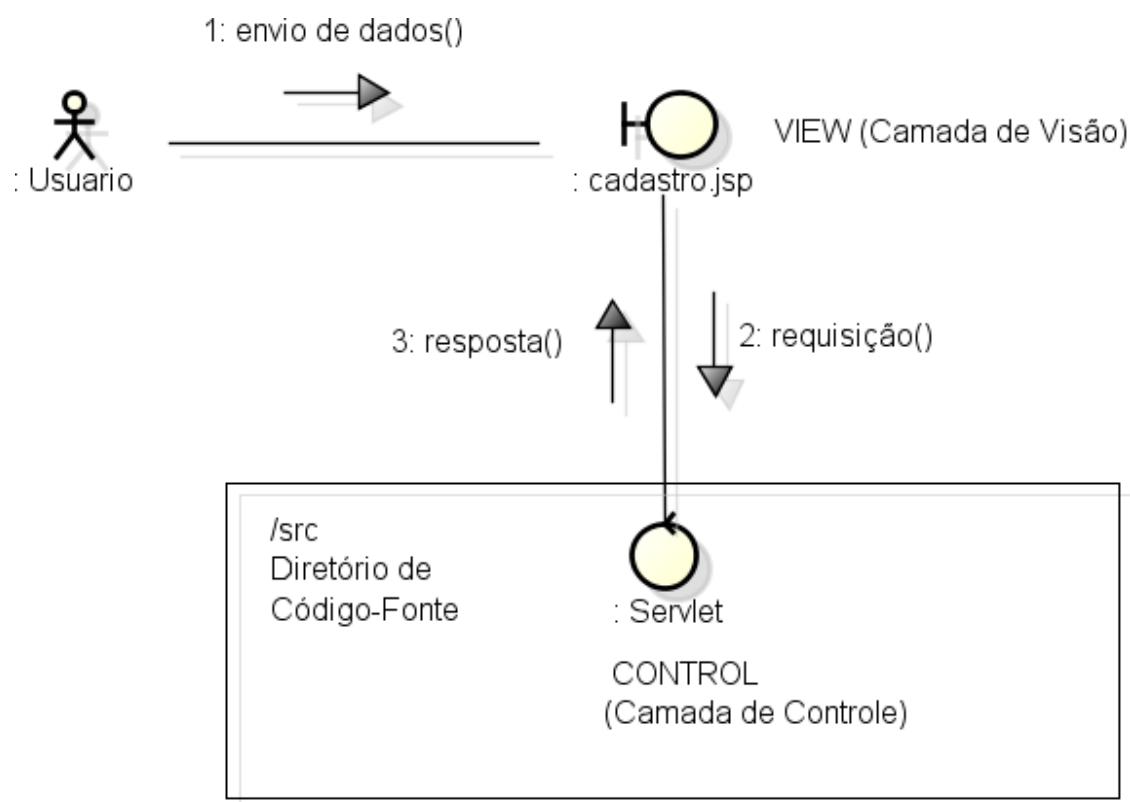
Sergio Mendes  
sergio.coti@gmail.com



No MySql...

```
C:\WINDOWS\system32\cmd.exe - mysql - u root
mysql> select * from cliente;
+-----+-----+
| idcliente | nome      | email        |
+-----+-----+
|       1 | Sergio Mendes | sergio.coti@gmail.com |
+-----+
1 row in set (0.00 sec)
```

## Diagrama de Comunicação do Projeto





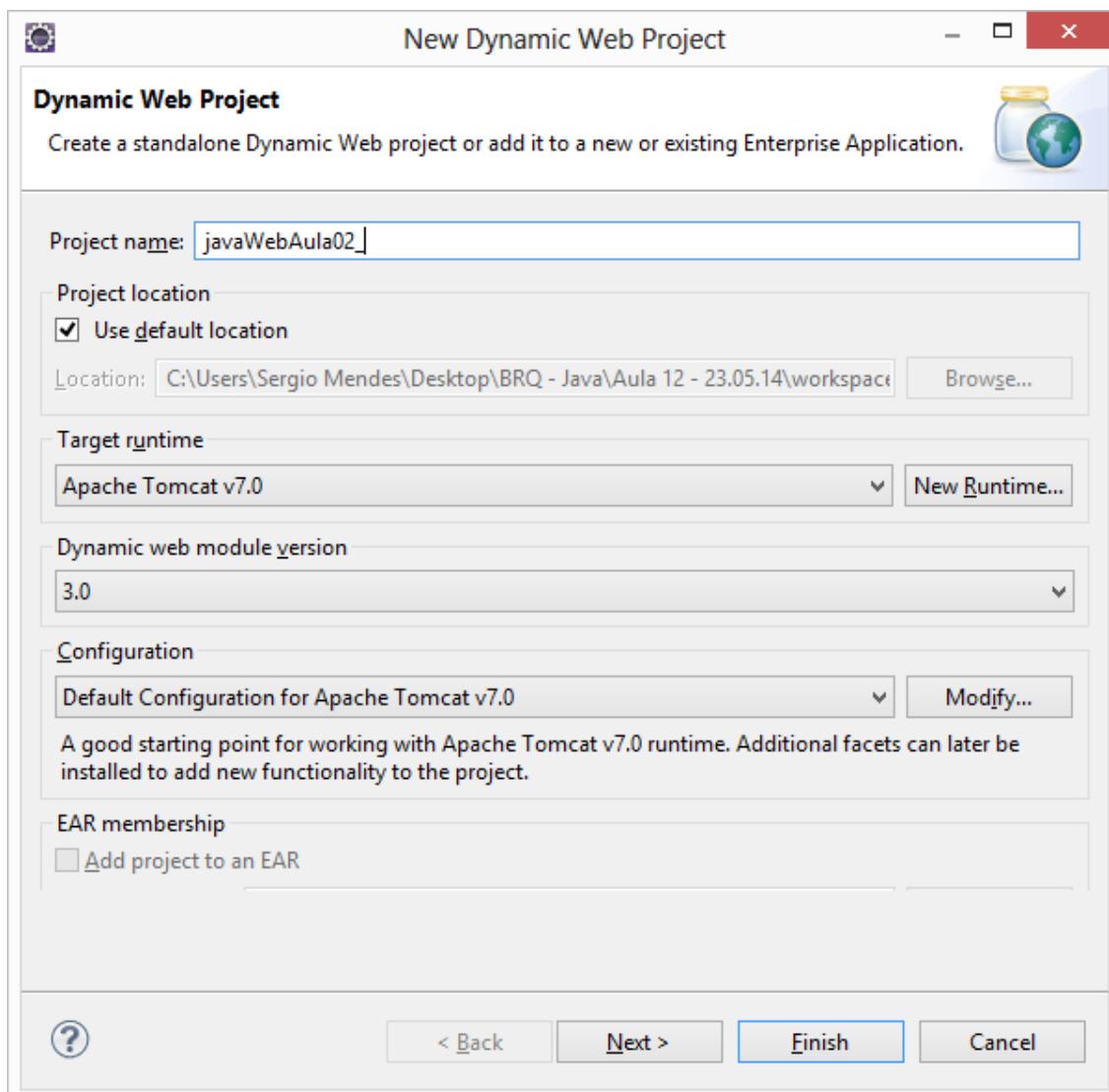
# Java WebDeveloper - BRQ

Sexta-feira, 23 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC, JDBC e JSTL Tag Libraries

Aula  
12

Criando o Projeto...



**Criando a página inicial**

\WebContent\index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
        </head>

    <body>

        <h3>Selecione a operação desejada</h3>
        <hr/>
```



```
<ul>
    <li> <a href="cadastro.jsp">Cadastrar Produtos</a> </li>
    <li> <a href="consulta.jsp">Consultar Produtos</a> </li>
</ul>

</body>

</html>
```

## Criando a página de cadastro de produtos

\WebContent\cadastro.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
    </head>

    <body>

        <h3>Cadastro de Produtos</h3>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

        <form name="formcadastro">

            Nome do Produto: <br/>
            <input type="text" name="nome" required="required"/>
            <br/><br/>

            Preço: <br/>
            <input type="text" name="preco"
                   required="required"/>
            <br/><br/>

            Quantidade: <br/>
            <input type="number" name="quantidade" min="1"
                   max="100" required="required"/>
            <br/><br/>

            <input type="submit" value="Cadastrar Produto"/>

        </form>

    </body>

</html>
```



## Criando a página de consulta de produtos

\WebContent\consulta.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
    </head>

    <body>

        <h3>Consulta de Produtos</h3>
        <a href="index.jsp">Voltar</a> para a página inicial
        <hr/>

        <form name="formconsulta">

            Informe o nome do produto desejado:
            <input type="text" name="nome" required="required"/>

            <input type="submit" value="Pesquisar Produtos"/>

        </form>

    </body>

</html>
```

Executando...

Selecione a operação desejada

- [Cadastrar Produtos](#)
- [Consultar Produtos](#)



# Java WebDeveloper - BRQ

Sexta-feira, 23 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC, JDBC e JSTL Tag Libraries

Aula  
12

Cadastro de Produtos

[Voltar para a página inicial.](#)

Nome do Produto:

Preço:

Quantidade:

Consulta de Produtos

[Voltar para a página inicial](#)

Informe o nome do produto desejado:

Criando o JavaBean Produto...

```
package entity;

public class Produto {

    private Integer idProduto;
    private String nome;
    private Double preco;
    private Integer quantidade;

    public Produto() {
        // Construtor default
    }
}
```



```
// Sobrecarga de Construtores
public Produto(Integer idProduto, String nome, Double preco,
               Integer quantidade) {
    super();
    this.idProduto = idProduto;
    this.nome = nome;
    this.preco = preco;
    this.quantidade = quantidade;
}

@Override
public String toString() {
    return "Produto [idProduto=" + idProduto + ", nome="
           + nome + ", preco=" + preco + ", quantidade="
           + quantidade + "]";
}

public Integer getIdProduto() {
    return idProduto;
}

public void setIdProduto(Integer idProduto) {
    this.idProduto = idProduto;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

public Integer getQuantidade() {
    return quantidade;
}

public void setQuantidade(Integer quantidade) {
    this.quantidade = quantidade;
}

}
```



Criando a base de dados...

```
drop database if exists aulaweb02;
create database aulaweb02;
use aulaweb02;

create table produto(
    idproduto          integer      auto_increment,
    nome               varchar(50)   not null,
    preco              double       not null,
    quantidade         integer      not null,
    primary key(idproduto));

alter table produto
add check(quantidade between 1 and 100);

show tables;

desc produto;

insert into produto values(null, 'Mouse Optico', 50.0, 10);
insert into produto values(null, 'Computador', 1500.0, 12);
insert into produto values(null, 'DVD MickeyMouse', 20.0, 3);

select * from produto;

select * from produto where nome = 'Mouse';
select * from produto where nome like 'Mouse%';
select * from produto where nome like '%Mouse';
select * from produto where nome like '%Mouse%';
```

## Classe Dao

Data Access Object

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

    protected Connection con;
    protected PreparedStatement stmt;
    protected ResultSet rs;

    protected void open() throws Exception{
        Class.forName("com.mysql.jdbc.Driver");
```



```
con = DriverManager.getConnection
("jdbc:mysql://localhost:3306/aulaweb02", "root", "");

}

protected void close() throws Exception{
    con.close();
}

}
```

### Criando a Classe ProdutoDao...

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Produto;

public class ProdutoDao extends Dao{

    //Método para gravar um registro de produto na tabela
    public void create(Produto p) throws Exception{

        open(); //abrir conexão

        stmt = con.prepareStatement("insert into produto
                                    values(null, ?, ?, ?)");
        stmt.setString(1, p.getNome());
        stmt.setDouble(2, p.getPreco());
        stmt.setInt(3, p.getQuantidade());
        stmt.execute(); //executar

        close(); //fechar conexão
    }

    //Método para consultar todos os produtos pelo nome
    public List<Produto> findAll(String nome) throws Exception{

        open(); //conexão aberta

        stmt = con.prepareStatement("select * from produto
                                    where nome like ?");
        stmt.setString(1, "%" + nome + "%");
        rs = stmt.executeQuery();
        //executo a consulta e carrego os dados no ResultSet

        List<Produto> lista = new ArrayList<Produto>();
        //lista vazia!

        while(rs.next())
        //enquanto houver registros no ResultSet
    }
}
```



```
{  
    Produto p = new Produto(  
        rs.getInt("idproduto"),  
        rs.getString("nome"),  
        rs.getDouble("preco"),  
        rs.getInt("quantidade"));  
  
    //adicionar o produto na lista  
    lista.add(p); //guardo na lista  
}  
  
close(); //conexão fechada  
return lista; //retornar a lista  
}  
  
  
public static void main(String[] args) {  
  
    try{  
  
        ProdutoDao d = new ProdutoDao();  
  
        for(Produto p : d.findAll("Mouse")){  
            System.out.println("Dados -> " + p);  
        }  
    }  
    catch(Exception e){  
        System.out.println(e.getMessage());  
    }  
}  
}
```

## JSP - Java Server Pages

Páginas Web baseadas em linguagem Java, podem conter:

- \* HTML, CSS, JavaScript, Código Java (Scriptlets)

Primeira página de um projeto Java Web tem o nome de index.jsp  
Fica localizada dentro da pasta \WebContent

## Padrão Command

Programas que realizem solicitações a um mesmo componente informando através de um flag ou variável qual rotina lógica o componente deverá realizar



Enviando o comando de ação para o Servlet...  
Página de Cadastro

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
    </head>

    <body>

        <h3>Cadastro de Produtos</h3>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

        <form name="formcadastro" method="post"
              action="ControleProduto?cmd=cadastrar">

            Nome do Produto: <br/>

            <input type="text" name="nome" required="required"/>
            <br/><br/>

            Preço: <br/>

            <input type="text" name="preco"
                   required="required"/>
            <br/><br/>

            Quantidade: <br/>

            <input type="number" name="quantidade" min="1"
                   max="100" required="required"/>
            <br/><br/>

            <input type="submit" value="Cadastrar Produto"/>

        </form>

        <p>
            ${msg}
        </p>

    </body>

</html>
```



### Página de Consulta...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>

    <head>
        </head>

    <body>

        <h3>Consulta de Produtos</h3>
        <a href="index.jsp">Voltar</a> para a página inicial
        <hr/>

        <form name="formconsulta" method="post"
              action="ControleProduto?cmd=consultar">

            Informe o nome do produto desejado:
            <input type="text" name="nome" required="required"/>

            <input type="submit" value="Pesquisar Produtos"/>

        </form>

        <p>
            ${msg}
        </p>

    </body>

</html>
```

Criando a lógica no Servlet para resgatar o comando enviado e processar as ações de cadastro e consulta...

```
package control;

import java.io.IOException;
import java.util.List;

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 persistence.ProdutoDao;
import entity.Produto;

@WebServlet("/ControleProduto")
public class ControleProduto extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleProduto() {
        super();
    }

    //Receber solicitações enviadas por protocolo GET
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response) throws
        ServletException, IOException {
    }

    //Receber solicitações enviadas por protocolo POST
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response) throws
        ServletException, IOException {

        //PrintWriter out = response.getWriter();
        //criando um objeto para impressão no servlet
        //Resgatar a variável 'cmd' (comando)

        String cmd = request.getParameter("cmd");
        //resgatando a variável enviada pelos formulários

        //Testar os valores da variável cmd
        if(cmd.equals("cadastrar")){
            //out.print("Comando cadastrar solicitado!!");

            cadastrar(request, response);
        }
        else if(cmd.equals("consultar")){
            //out.print("Comando consultar solicitado!!");
            consultar(request, response);
        }
    }

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

        try{
            Produto p = new Produto(); //JavaBean
            p.setNome( request.getParameter("nome") );
        }
    }
}
```



```
p.setPreco( new Double(
            request.getParameter("preco")) );
p.setQuantidade( new Integer(
            request.getParameter("quantidade")) );

ProdutoDao d = new ProdutoDao(); //persistência
d.create(p); //gravação

request.setAttribute("msg", "Produto cadastrado
                            com sucesso.");
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
                            + e.getMessage());
}
finally{
    //redirecionamento de volta para a página...
    request.getRequestDispatcher("cadastro.jsp").
        forward(request, response);
}

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

    try{

        String nome = request.getParameter("nome");
        //resgatando o campo nome do formulário

        ProdutoDao d = new ProdutoDao();
        List<Produto> lista = d.findAll(nome);

        request.setAttribute("dados", lista);
    }
    catch(Exception e){
        request.setAttribute("msg", "Erro -> "
                            + e.getMessage());
    }
    finally{
        request.getRequestDispatcher("consulta.jsp").
            forward(request, response);
    }

}
```



### Executando o Cadastro de Produtos...

Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:

Preço:

Quantidade:

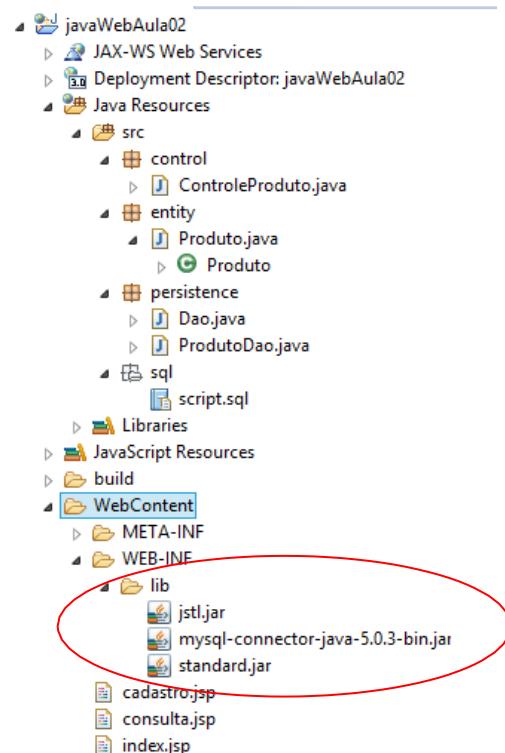
Produto cadastrado com sucesso.



## JSTL Java Standard TagLibraries

Bibliotecas de <tags> similares ao HTML, porém compatíveis com Java  
Podemos através de JSTL utilizar tags que irão interagir com dados  
enviados pelo servidor

- Incluindo as bibliotecas no projeto...





# Java WebDeveloper - BRQ

Sexta-feira, 23 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC, JDBC e JSTL Tag Libraries

Aula  
12

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>

    <head>
    </head>

    <body>
        <h3>Consulta de Produtos</h3>
        <a href="index.jsp">Voltar</a> para a página inicial
        <hr/>

        <form name="formconsulta" method="post"
              action="ControleProduto?cmd=consultar">

            Informe o nome do produto desejado:
            <input type="text" name="nome" required="required"/>

            <input type="submit" value="Pesquisar Produtos"/>
        </form>

        <p>
            ${msg}
        </p>

        <table border="1" style="width: 80%">
            <tr>
                <th>Código</th>
                <th>Nome do Produto</th>
                <th>Preço</th>
                <th>Quantidade</th>
                <th>Total</th>
            </tr>

            <c:forEach items="${dados}" var="p">
                <tr>
                    <td> ${p.idProduto} </td>
                    <td> ${p.nome} </td>
                    <td> ${p.preco} </td>
                    <td> ${p.quantidade} </td>
                    <td> ${p.preco * p.quantidade} </td>
                </tr>
            </c:forEach>

        </table>

    </body>
</html>
```



# Java WebDeveloper - BRQ

Sexta-feira, 23 de Maio de 2014

Introdução ao Java Web. Java Server Pages, Servlets, Expression Languages, Modelo MVC, JDBC e JSTL Tag Libraries

Aula

12

Executando...

Consulta de Produtos

[Voltar](#) para a página inicial

Informe o nome do produto desejado:

Código	Nome do Produto	Preço	Quantidade	Total
--------	-----------------	-------	------------	-------

Consulta de Produtos

[Voltar](#) para a página inicial

Informe o nome do produto desejado:

Código	Nome do Produto	Preço	Quantidade	Total
1	Mouse Optico	50.0	10	500.0
3	DVD MickeyMouse	20.0	3	60.0



# Java WebDeveloper - BRQ

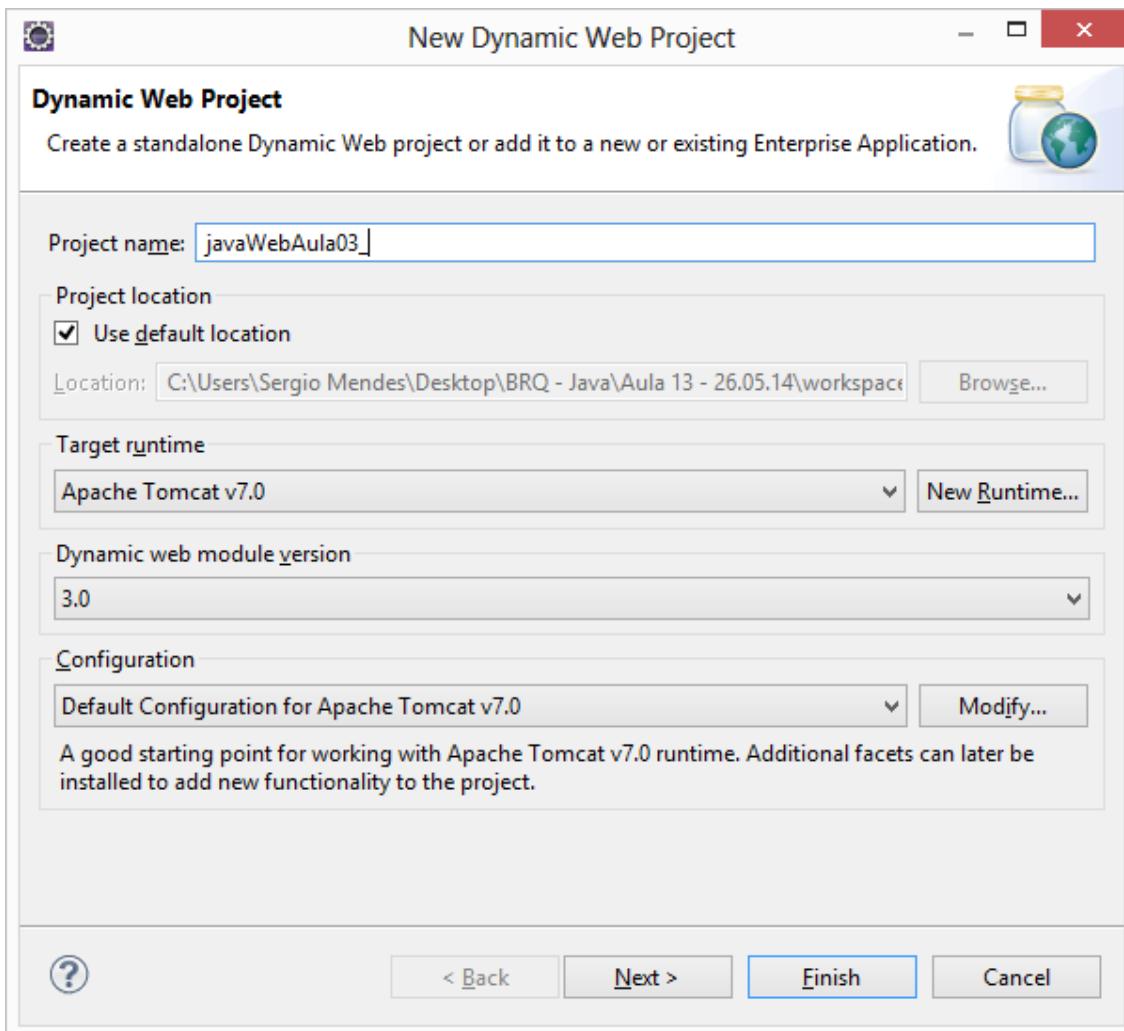
Segunda-feira, 26 de Maio de 2014

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula

13

Criando o Projeto...



Criando a página inicial...

\WebContent\index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<html>
    <head>
        <title>Java Web - Aula 03</title>
        <style>
            body { font-family: verdana; font-size: 9pt;
            padding: 40px; }
        </style>
    </head>
    <body>
        <h3>Projeto Controle de Clientes</h3>
        <hr/>
        <a href="cadastro.jsp">Cadastrar Novo Cliente</a>
    </body>
</html>
```



# Java WebDeveloper - BRQ

Segunda-feira, 26 de Maio de 2014

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula  
**13**

## JQuery

Biblioteca para JavaScript que otimiza a escrita e o uso do JavaScript em aplicações web. Possui recursos otimizados para validação de dados, desenvolvimento de interface de sistemas, ajax, etc...

- **JQuery User Interface (JqueryUI)**  
<http://jqueryui.com/>

The screenshot shows the official jQuery UI website. At the top, there's a navigation bar with links for Plugins, Contribute, Events, Support, and jQuery Foundation. Below the header, there's a banner for a "Virtual Training" event from June 17-19, 2014, which is online. The main content area has a sidebar on the left with sections for "Interactions" (Draggable, Droppable, Resizable, Selectable, Sortable) and "Widgets". The main content area contains a brief introduction to jQuery UI and a "Download jQuery UI" section featuring version 1.10.4, with links for "Custom Download", "Quick Downloads" (Stable v1.10.4, Legacy v1.9.2), and "jQuery 1.6+".

<http://jqueryui.com/themeroller/>

The screenshot shows the ThemeRoller interface, a tool for customizing jQuery UI themes. On the left, there's a sidebar with "Roll Your Own", "Gallery", and "Help" options. The main area is divided into several sections: "Accordion" (with three sections labeled "Section 1", "Section 2", and "Section 3"), "Tabs" (with tabs "First", "Second", and "Third"), and various UI components like "Button", "Autocomplete", "Spinner", "Slider", and "Datepicker". Each component has preview boxes and "Download" or "Edit" buttons. A central text area contains placeholder text for the accordion sections.



# Java WebDeveloper - BRQ

Segunda-feira, 26 de Maio de 2014

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula  
**13**

The screenshot shows the 'Download Builder' page for jQuery UI. At the top, there are quick download links for 'Stable Themes' (1.10.4) and 'Legacy Themes' (1.9.2). Below this, the 'Version' section has '1.10.4 (Stable, for jQuery1.6+)' selected. Under 'Components', 'Toggle All' is checked. The 'UI Core' section is expanded, showing 'Core' (selected), 'Widget', 'Mouse', and 'Position'. The 'Interactions' section is also expanded, showing 'Draggable', 'Droppable', 'Resizable', 'Selectable', and 'Sortable'. A note in the 'Interactions' section says: 'These add basic behaviors to any element and are used by many components below.'

Criando a página de Cadastro de Clientes  
utilizando o JQuery User Interface...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>

        <!-- referenciar o arquivo css (estilo) -->
        <link rel="stylesheet" type="text/css"
              href="css/redmond/jquery-ui-1.10.4.custom.css"/>

        <style>
            body { font-family: verdana; font-size: 9pt;
                  padding: 40px; }
        </style>

        <!-- referenciar os arquivos javascript do jquery -->

        <script type="text/javascript"
               src="js/jquery-1.10.2.js"></script>

        <script type="text/javascript"
               src="js/jquery-ui-1.10.4.custom.js"></script>
```



# Java WebDeveloper - BRQ

Segunda-feira, 26 de Maio de 2014

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula

13

```
<script type="text/javascript">

    //função principal do jquery (inicia o uso do jquery)
    $(document).ready(
        //quando a página estiver pronta (carregada)...
        function(){ //faça...

            $("#btnjanela").button();
            //formatar o botão btnjanela
            $("#btncadastro").button();
            //formatar o botão de cadastro (submit)

            //formatar a div como uma janela modal
            $("#janela").dialog(
                { width : '600', modal : true,
                  show : 'fade', hide : 'drop',
                  autoOpen : false }
            );

            //criar um evento para o botão btnjanela
            //abrir o dialog...
            $("#btnjanela").click(
                //quando o botão 'btnjanela' for clicado...
                function(){ //faça...
                    $("#janela").dialog("open");
                    //mostrar a janela de dialogo...
                }
            );

            //transformar a div 'painel'
            //em uma navegação de abas
            $("#painel").tabs();

        }
    );
}

</script>

</head>

<body>

    <h3>Cadastro de Clientes</h3>

    <a href="index.jsp">Voltar</a> para a página inicial.

    <hr/>

    <button id="btnjanela">
        Formulário de Cadastro de Clientes
    </button>

    <h4> ${msg} </h4>
```



```
<!-- Área para gerar uma janela de dialogo modal -->
<div id="janela" title="Formulário de Cadastro de Clientes">
    <p>
        Para cadastrar um novo cliente,
        preencha os campos das abas abaixo:
    </p>
    <form name="formcadastro">
        <!-- Área de navegação em abas -->
        <div id="painel">
            <ul>
                <li> <a href="#aba1">
                    Dados do Cliente</a> </li>
                <li> <a href="#aba2">
                    Dados de Endereço</a> </li>
            </ul>
            <div id="aba1">
                Nome do Cliente: <br/>
                <input type="text" name="nome"
size="60" required="required"/>
                <br/><br/>

                Endereço de Email: <br/>
                <input type="email" name="email"
size="60" required="required"/>
                <br/><br/>

                Sexo:
                <select name="sexo"
                    required="required">
                    <option value="">
                        - Escolha uma Opção -
                    </option>
                    <option value="F">Feminino</option>
                    <option value="M">Masculino</option>
                </select>
            </div>
            <div id="aba2">
                Logradouro: <br/>
                <textarea name="Logradouro" cols="60"
rows="4" required="required">
                </textarea>
                <br/><br/>

                Cidade: <br/>
                <input type="text" name="cidade"
required="required" size="40"/>
                <br/><br/>

                Estado: <br/>
                <input type="text" name="estado"
required="required" size="40"/>
            </div>
        </div>
        <p>
            <input type="submit" id="btncadastro"
value="Cadastrar Cliente"/>
        </p>
    </form>
</div>
```



# Java WebDeveloper - BRQ

Segunda-feira, 26 de Maio de 2014

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula  
**13**

```
</form>

</div>

</body>

</html>
```

Resultado...

Cadastro de Clientes

Voltar para a página anterior

Formulário de Cadastro de Clientes

Para cadastrar um novo cliente, preencha os campos das abas abaixo:

Dados do Cliente Dados de Endereço

Nome do Cliente:

Endereço de Email:

Sexo:

**Cadastrar Cliente**

Cadastro de Clientes

Voltar para a página anterior

Formulário de Cadastro de Clientes

Para cadastrar um novo cliente, preencha os campos das abas abaixo:

Dados do Cliente Dados de Endereço

Logradouro:

Cidade:

Estado:

**Cadastrar Cliente**



### Criando a base de dados...

Relacionamento 1 para 1

```
drop database if exists aulaweb03;
create database aulaweb03;
use aulaweb03;

create table cliente(
    idcliente          integer          auto_increment,
    nome               varchar(50)       not null,
    email              varchar(50)       not null unique,
    sexo               enum('Masculino', 'Feminino') not null,
    primary key(idcliente));

create table endereco(
    idendereco         integer          auto_increment,
    logradouro        varchar(255)      not null,
    cidade             varchar(50)       not null,
    estado             varchar(50)       not null,
    idcliente          integer          not null unique,
    primary key(idendereco),
    foreign key(idcliente) references cliente(idcliente));

show tables;

desc cliente;
desc endereco;
```

### Criando as Classes JavaBean...

```
package entity;

public enum Sexo {
    Feminino,
    Masculino,
}

package entity;

public class Cliente {

    private Integer idCliente;
    private String nome;
    private String email;
    private Sexo sexo;
    private Endereco endereco; // Associação -> TER-1

    public Cliente() {
        // Construtor default da Classe
    }
}
```



```
public Cliente(Integer idCliente, String nome, String email,
               Sexo sexo) {
    super();
    this.idCliente = idCliente;
    this.nome = nome;
    this.email = email;
    this.sexo = sexo;
}

@Override
public String toString() {
    return "Cliente [idCliente=" + idCliente + ", nome=" + nome
           + ", email=" + email + ", sexo=" + sexo + "]";
}

public Integer getIdCliente() {
    return idCliente;
}

public void setIdCliente(Integer idCliente) {
    this.idCliente = idCliente;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Sexo getSexo() {
    return sexo;
}

public void setSexo(Sexo sexo) {
    this.sexo = sexo;
}

public Endereco getEndereco() {
    return endereco;
}

public void setEndereco(Endereco endereco) {
    this.endereco = endereco;
}

}
```



```
package entity;

public class Endereco {

    private Integer idEndereco;
    private String logradouro;
    private String cidade;
    private String estado;
    private Cliente cliente; // Associação -> TER-1

    public Endereco() {
        // Construtor default
    }

    public Endereco(Integer idEndereco, String logradouro, String cidade,
                    String estado) {
        super();
        this.idEndereco = idEndereco;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "Endereco [idEndereco=" + idEndereco + ", logradouro="
               + logradouro + ", cidade=" + cidade + ", estado="
               + estado + "]";
    }

    public Integer getIdEndereco() {
        return idEndereco;
    }

    public void setIdEndereco(Integer idEndereco) {
        this.idEndereco = idEndereco;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }
}
```



```
public void setEstado(String estado) {
    this.estado = estado;
}

public Cliente getCliente() {
    return cliente;
}

public void setCliente(Cliente cliente) {
    this.cliente = cliente;
}

}
```

### Criando a Classe Dao

Data Access Object

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

    protected Connection con;
    protected PreparedStatement stmt;
    protected ResultSet rs;

    protected void open() throws Exception{
        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://
            localhost:3306/aulaweb03", "root", "");
    }

    protected void close() throws Exception{
        con.close();
    }
}
```

- Criando a Classe de persistência para o JavaBean Cliente

### PreparedStatement.RETURN\_GENERATED\_KEYS

Executa uma rotina de gravação em tabela retornando o id gerado pelo campo auto\_increment.

Utilizado em métodos que realizam gravação de registro em tabela e retorna o id gerado no registro gravado pela base de dados...



```
package persistence;

import java.sql.PreparedStatement;
import entity.Cliente;

public class ClienteDao extends Dao {

    //Método para gravar um Cliente na base de dados
    public int create(Cliente c) throws Exception{

        //RETURN_GENERATED_KEYS
        //Habilitando o preparedstatament para retornar a
        //chave gerada no comando insert feito na tabela

        open();

        stmt = con.prepareStatement("insert into cliente
            values(null, ?, ?, ?)",
            PreparedStatement.RETURN_GENERATED_KEYS);
        stmt.setString(1, c.getNome());
        stmt.setString(2, c.getEmail());
        stmt.setString(3, c.getSexo().name());
        stmt.execute(); //executar

        //Pegar a chave gerada (auto_incremeto)
        rs = stmt.getGeneratedKeys();
        //capturar a chave gerada
        int chave = 0; //variavel local valendo zero

        if(rs.next()){
            //se o resultset obteve algum registro
            chave = rs.getInt(1);
            //1 -> representa o unico valor
            //obtido pelo ResultSet
        }

        close();
        return chave; //retornar a chave
    }
}
```

- Criando a Classe de persistência para o JavaBean Endereco...

```
package persistence;

import entity.Endereco;

public class EnderecoDao extends Dao{

    //Método para gravar Endereco
    public void create(Endereco e) throws Exception{

        open();
```



```
        stmt = con.prepareStatement("insert into endereco values
                                    (null, ?, ?, ?, ?)");
        stmt.setString(1, e.getLogradouro());
        stmt.setString(2, e.getCidade());
        stmt.setString(3, e.getEstado());
        stmt.setInt(4, e.getCliente().getIdCliente());
        //chave estrangeira
        stmt.execute(); //executar

        close();
    }
}
```

Realizando a chamada ao Controle...

```
<form name="formcadastro" method="post"
      action="ControleCliente?cmd=cadastrar">
```

Criando o Servlet para Controle

```
package control;

import java.io.IOException;

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 persistence.ClienteDao;
import persistence.EnderecoDao;
import entity.Cliente;
import entity.Endereco;
import entity.Sexo;

@WebServlet("/ControleCliente")
public class ControleCliente extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleCliente() {
        super();
    }

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



# Java WebDeveloper - BRQ

Segunda-feira, 26 de Maio de 2014

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula  
**13**

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response) throws
                        ServletException, IOException {

    //resgatar a variavel cmd
    String comando = request.getParameter("cmd");

    //verificando o valor da variavel cmd
    if(comando.equals("cadastrar")){
        cadastrar(request, response);
    }
}

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

try{

    Cliente c = new Cliente();
    c.setNome( request.getParameter("nome") );
    //resgato campo nome
    c.setEmail( request.getParameter("email") );
    //resgato campo email

    switch(request.getParameter("sexo")) {
        //resgato campo sexo
        case "M" :
            c.setSexo(Sexo.Masculino);
            break;
        case "F":
            c.setSexo(Sexo.Feminino);
            break;
    }

    Endereco e = new Endereco();
    e.setLogradouro(request.
    getParameter("logradouro"));
    //resgato campo logradouro
    e.setCidade(request.getParameter("cidade"));
    //resgato campo cidade
    e.setEstado(request.getParameter("estado"));
    //resgato campo estado

    ClienteDao cd = new ClienteDao();
    EnderecoDao ed = new EnderecoDao();

    int chave = cd.create(c); //gravar o cliente

    c.setIdCliente(chave);
    //passando a chave para o objeto Cliente
    e.setCliente(c);
    //relacionar o endereco ao cliente
}
```



# **Java WebDeveloper - BRQ**

**Segunda-feira, 26 de Maio de 2014**

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula  
13

```
        ed.create(e); //gravar o endereço  
  
        request.setAttribute("msg", "Dados gravados  
                                com sucesso");  
    }  
    catch (Exception e) {  
  
        request.setAttribute("msg", "Erro -> "  
                            + e.getMessage());  
    }  
    finally{  
  
        request.getRequestDispatcher("cadastro.jsp").  
        forward(request, response);  
    }  
}
```

## Executando...

Cadastro de Clientes

Voltar para a página anterior

Formulário de Cadastro de Clientes

Para cadastrar um novo cliente, preencha os campos das abas abaixo:

**Dados do Cliente** **Dados de Endereço**

Nome do Cliente:  
Sergio Mendes

Endereço de Email:  
sergio.coti@gmail.com

Sexo: Masculino ▾

**Cadastrar Cliente**

**Cadastro de Clientes**

[Voltar](#) para a página inicial.

---

**Formulário de Cadastro de Clientes**

**Dados gravados com sucesso**



# Java WebDeveloper - BRQ

Segunda-feira, 26 de Maio de 2014

Java Server Pages, Desenvolvimento de interface com JQuery UI.  
Relacionamento 1 para 1 em banco de dados.

Aula

13

No banco de dados...

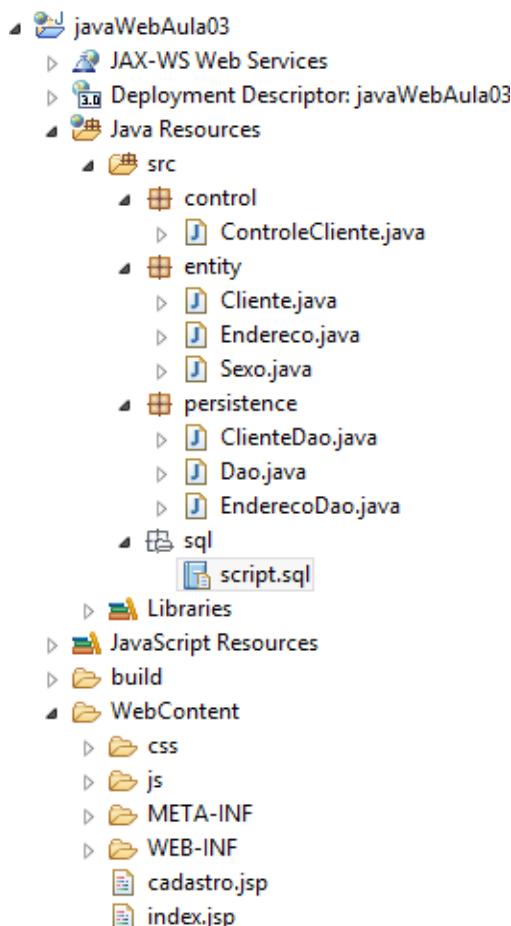
```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> select * from cliente;
+-----+-----+-----+
| idcliente | nome      | email          | sexo   |
+-----+-----+-----+
|       1 | Sergio Mendes | sergio.coti@gmail.com | Masculino |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from endereco;
+-----+-----+-----+-----+
| idendereco | logradouro | cidade        | estado | idcliente |
+-----+-----+-----+-----+
|       1 | Av Rio Branco 185 Centro | Rio de Janeiro | RJ    |       1 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

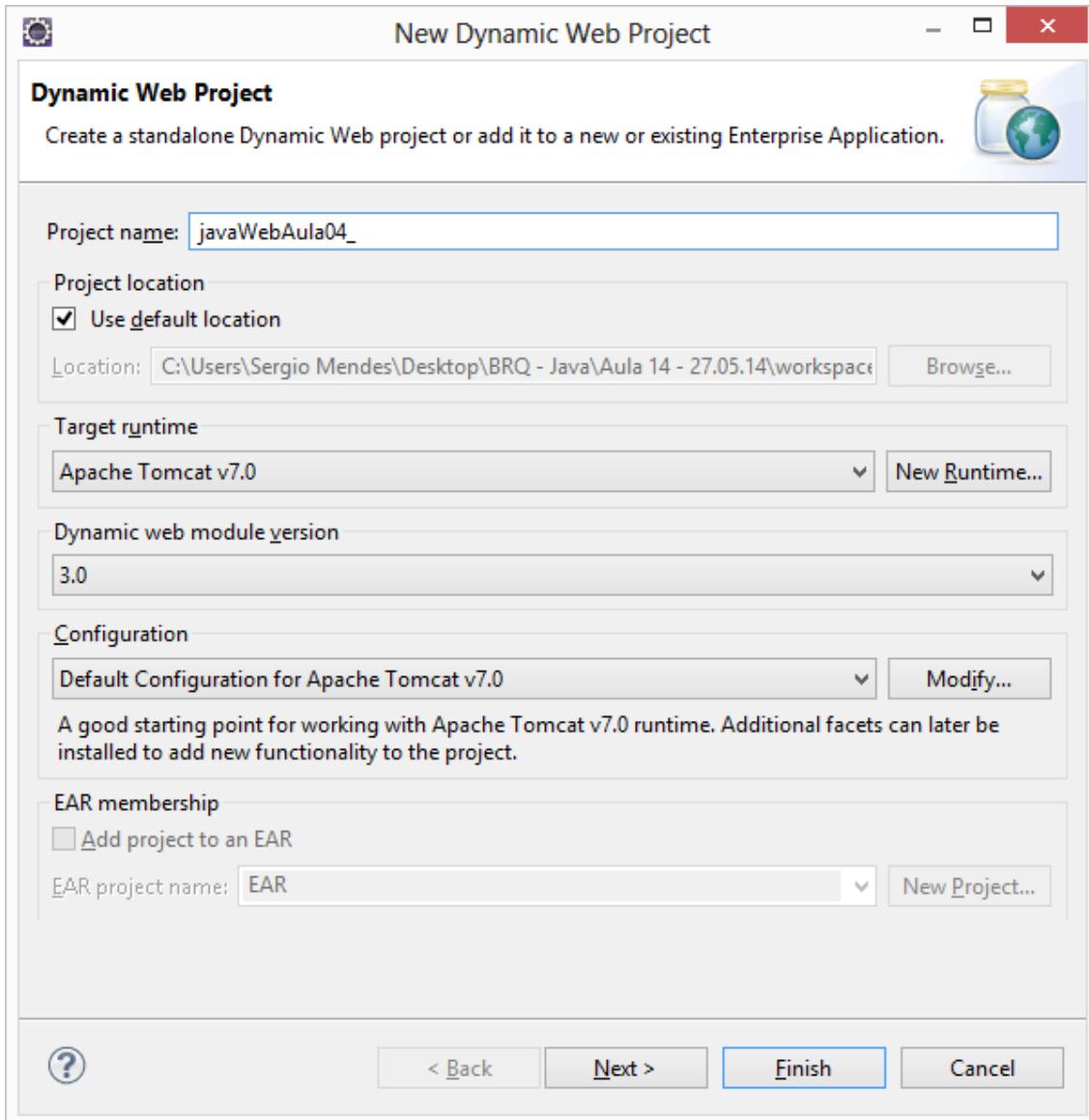
mysql> -
```

Estrutura do Projeto...





Criando o Projeto...



Criando a base de dados...

```
drop database if exists aulaweb04;
create database aulaweb04;
use aulaweb04;

create table cliente(
    idcliente      integer      auto_increment,
    nome           varchar(50)   not null,
    email          varchar(50)   not null unique,
    primary key(idcliente));

show tables;

desc cliente;
```



Criando o JavaBean Cliente...

```
package entity;

public class Cliente { // JavaBean

    private Integer idCliente;
    private String nome;
    private String email;

    public Cliente() {
        // Construtor default
    }

    // Sobrecarga de Construtores (Entrada de parametros)
    public Cliente(Integer idCliente, String nome, String email) {
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
    }

    @Override
    public String toString() {
        return "Cliente [idCliente=" + idCliente + ", nome="
               + nome + ", email=" + email + "]";
    }

    public Integer getIdCliente() {
        return idCliente;
    }

    public void setIdCliente(Integer idCliente) {
        this.idCliente = idCliente;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```



## Criando a Classe DAO

### Data Access Object

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

//Tecnologia - JDBC
public class Dao {

    protected Connection con;
    //abrir e fechar a conexão com o BD

    protected PreparedStatement stmt;
    //executar as instruções SQL

    protected ResultSet rs;
    //ler registros obtidos de consultas

    //Método para abrir conexão com o banco de dados
    protected void open() throws Exception{

        //Carregar o caminho do Driver do MySQL (.JAR)
        Class.forName("com.mysql.jdbc.Driver");
        //Realizo a conexão com o BD

        con = DriverManager.getConnection
            ("jdbc:mysql://localhost:3306/aulaweb04",
            "root", "");
    }

    //Método para fechar a conexão
    protected void close() throws Exception{
        con.close(); //conexão encerrada!
    }
}
```

## Criando a Classe de persistência para o JavaBean Cliente...

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Cliente;

public class ClienteDao extends Dao {
```



```
//Método para gravar um registro na tabela Cliente
public void create(Cliente c) throws Exception{

    open();

    stmt = con.prepareStatement("insert into cliente
                                values(null, ?, ?)");
    stmt.setString(1, c.getNome());
    stmt.setString(2, c.getEmail());
    stmt.execute(); //executar

    close();
}

//Método para excluir o registro de cliente
public void delete(Integer idCliente) throws Exception{

    open();

    stmt = con.prepareStatement("delete from cliente
                                where idcliente = ?");
    stmt.setInt(1, idCliente);
    stmt.execute();

    close();
}

//Método para atualizar o registro de cliente
public void update(Cliente c) throws Exception{

    open();

    stmt = con.prepareStatement("update cliente set
                                nome = ?, email = ? where idcliente = ?");
    stmt.setString(1, c.getNome());
    stmt.setString(2, c.getEmail());
    stmt.setInt(3, c.getIdCliente());
    stmt.execute();

    close();
}

//Método para retornar 1 Cliente pelo Id (Chave primária)
public Cliente findById(Integer idCliente)
    throws Exception{

    open();

    stmt = con.prepareStatement("select * from cliente
                                where idcliente = ?");
    stmt.setInt(1, idCliente);
    rs = stmt.executeQuery();
    //executo a consulta e leio os registros
}
```



```
Cliente c = null;
//NullPointerException (sem espaço de memória)

//verifico se o ResultSet encontrou algum registro
if(rs.next()){
//se o rs é verdadeiro

    //espaço de memória para o objeto Cliente
    //utilizando o construtor com entrada de dados
    //uso o 'rs' para trazer o valor de cada campo
    //da tabela do registro encontrado
    c = new Cliente( rs.getInt("idcliente"),
        rs.getString("nome"), rs.getString("email") );
}

close();

return c; //retornar o objeto Cliente
}

//Método para retornar uma lista com todos
//os clientes da tabela
public List<Cliente> findAll() throws Exception{

    open();

    stmt = con.prepareStatement("select * from cliente");
    rs = stmt.executeQuery();
    //executo a consulta e capturo os registros

    List<Cliente> lista = new ArrayList<Cliente>();
    //lista sem elementos

    //enquanto houver registros no ResultSet
    while(rs.next()){ //enquanto o 'rs' for verdadeiro

        //Criando objeto da classe Cliente
        Cliente c = new Cliente( rs.getInt("idcliente"),
            rs.getString("nome"), rs.getString("email") );

        //adicionar na lista
        lista.add(c);
    }

    close();

    return lista;
}
}
```



### Criando o arquivo de folha de estilos css Cascading Style Sheet

```
body /* formatação da tag <body> do HTML */
{
    font-family: verdana;          /* tipo da fonte */
    font-size: 9pt;                /* tamanho da fonte */
    padding: 40px;                 /* margem interna */
    background-color: #CCC;        /* cor de fundo */
}

#conteudo /* formatação aplicada a um elemtno HTML pelo ID */
{
    width: 0 auto;                /* largura proporcional */
    height: 0 auto;               /* altura proporcional */
    background-color: #FFF;        /* cor de fundo */
    border: 1px solid #999;        /* borda */
    padding: 20px;                 /* margem interna */
    display: block;                /* formato quadro -> bloco */
}
```

### Criando a página \index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <link rel="stylesheet" type="text/css"
            href="css/estilo.css"/>
    </head>
    <body>
        <div id="conteudo">

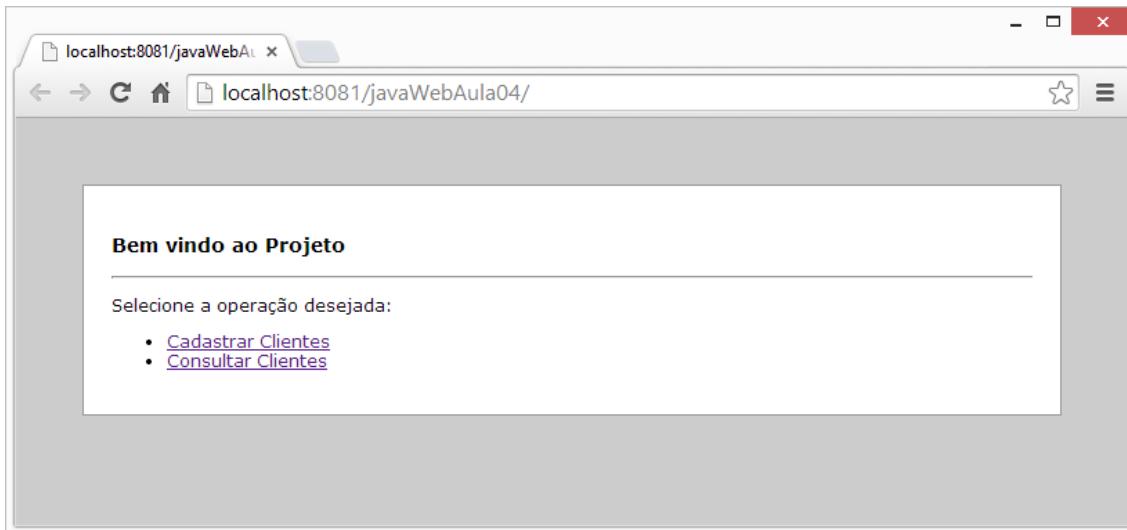
            <h3>Bem vindo ao Projeto</h3>
            <hr/>

            <p>
                Selecione a operação desejada:
            </p>

            <ul>
                <li> <a href="cadastro.jsp">
                    Cadastrar Clientes</a> </li>
                <li> <a href="consulta.jsp">
                    Consultar Clientes</a> </li>
            </ul>
        </div>
    </body>
</html>
```



Executando...



Criando a página para cadastro de Clientes...

### \cadastro.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <link rel="stylesheet" type="text/css"
            href="css/estilo.css"/>
    </head>
    <body>

        <div id="conteudo">

            <h3>Cadastro de Clientes</h3>
            <a href="index.jsp">Voltar</a>
                para a página inicial.
            <hr/>

            <form name="formcadastro" method="post"
                action="ControleCliente?cmd=cadastrar">

                Informe seu Nome: <br/>
                <input type="text" name="nome"
                    required="required"/>
                <br/><br/>

                Informe seu Email: <br/>
                <input type="email" name="email"
                    required="required"/>
                <br/><br/>
```



```
        <input type="submit"  
              value="Cadastrar Cliente"/>  
  
    </form>  
  
    <h4> ${msg} </h4>  
  
  </div>  
  
</body>  
  
</html>
```

Criando o Servlet para cadastro de Clientes...

```
package control;  
  
import java.io.IOException;  
  
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 persistence.ClienteDao;  
import entity.Cliente;  
  
@WebServlet("/ControleCliente")  
public class ControleCliente extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
  
    public ControleCliente() {  
        super();  
    }  
  
    //doGet -> utilizado quando o servlet for  
    //chamado por um LINK (URL)  
    protected void doGet(HttpServletRequest request,  
                         HttpServletResponse response) throws  
                         ServletException, IOException {  
    }  
  
    //doPost -> utilizado quando o servlet for  
    //chamado por um FORMULARIO  
    protected void doPost(HttpServletRequest request,  
                         HttpServletResponse response) throws  
                         ServletException, IOException {  
  
        //resgatar a variavel 'cmd' enviada pelos formulários  
        String cmd = request.getParameter("cmd");
```



```
if(cmd.equals("cadastrar")){
    cadastrar(request, response);
    //transferir o fluxo de execução
}
}

//Método do Servlet para executar o comando 'cadastrar'
protected void cadastrar(HttpServletRequest request,
    HttpServletResponse response) throws
    ServletException, IOException {
try{
    Cliente c = new Cliente();
    //criando objeto do JavaBean

    c.setNome( request.getParameter("nome") );
    //resgatando o valor do campo 'nome'
    c.setEmail( request.getParameter("email") );
    //resgatando o valor do campo 'email'

    ClienteDao d = new ClienteDao();
    d.create(c); //gravando o cliente

    request.setAttribute("msg", "Cliente " +
        c.getNome() + ", cadastrado com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
        + e.getMessage());
}
finally{
    request.getRequestDispatcher("cadastro.jsp").
        forward(request, response);
}
}
}
```

Executando...



# Java WebDeveloper - BRQ

Terça-feira, 27 de Maio de 2014

Java na Web, Projeto CRUD. JavaServerPages, Servlets, Expression Languages, JSTL TagLibraries

Aula  
14

Cadastro de Clientes

Voltar para a página inicial.

Informe seu Nome:

Informe seu Email:

Cadastrar Cliente

Cliente Ana Paula, cadastrado com sucesso.

No banco de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql - u root - p

mysql> select * from cliente;
+-----+-----+-----+
| idcliente | nome          | email        |
+-----+-----+-----+
|       1 | Sergio Mendes | sergio.coti@gmail.com
|       2 | Marcone Freitas | marcone@gmail.com
|       3 | Patricia Massaut | patricia@gmail.com
|       4 | Deborah Leme | deborah@bol.com
|       5 | Fagner Andre | fagner@gmail.com
|       6 | Pedro Miranda | pedro@bol.com
|       7 | Yuri de Souza Vidal | yuri@gmail.com
|       8 | Lucas Ribeiro | lucas@gmail.com
|      11 | Ana Paula | ana@gmail.com
+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>
```

Criando a página de Consulta de Clientes...

\consulta.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>

    <head>
        <link rel="stylesheet" type="text/css"
        href="css/estilo.css"/>
    </head>
```



```
<body>

    <div id="conteudo">

        <h3>Consulta de Clientes</h3>
        <a href="index.jsp">Voltar</a> para a página inicial
        <hr/>

        <form name="formconsulta" method="post"
            action="ControleCliente?cmd=consultar">

            <input type="submit"
                value="Consultar Clientes"/>

        </form>

        <h4> ${msg} </h4>

    </div>

</body>

</html>
```

Exibindo...



Criando a ação no Servlet para consulta de Clientes...

```
package control;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```



# Java WebDeveloper - BRQ

Terça-feira, 27 de Maio de 2014

Java na Web, Projeto CRUD. JavaServerPages, Servlets, Expression Languages, JSTL TagLibraries

Aula  
14

```
import javax.servlet.http.HttpServletResponse;

import persistence.ClienteDao;
import entity.Cliente;

@WebServlet("/ControleCliente")
public class ControleCliente extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleCliente() {
        super();
    }

    //doGet -> utilizado quando o servlet for chamado
    //por um LINK (URL)
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response) throws
    ServletException, IOException {
    }

    //doPost -> utilizado quando o servlet for
    //chamado por um FORMULARIO
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response) throws
    ServletException, IOException {

        //resgatar a variavel 'cmd' enviada pelos formulários
        String cmd = request.getParameter("cmd");

        if(cmd.equals("cadastrar")){
            cadastrar(request, response);
            //transferir o fluxo de execução
        }
        else if(cmd.equals("consultar")){
            consultar(request, response);
            //transferir o fluxo de execução
        }
    }

    //Método do Servlet para executar o comando 'cadastrar'
    protected void cadastrar(HttpServletRequest request,
                           HttpServletResponse response) throws
    ServletException, IOException {

        try{

            Cliente c = new Cliente();
            //criando objeto do JavaBean

            c.setNome( request.getParameter("nome") );
            //resgatando o valor do campo 'nome'
            c.setEmail( request.getParameter("email") );
            //resgatando o valor do campo 'email'

            ClienteDao d = new ClienteDao();
            d.create(c); //gravando o cliente

            request.setAttribute("msg", "Cliente " + c.getNome())
        }
    }
}
```



```
        + ", cadastrado com sucesso.");
    }
    catch(Exception e){
        request.setAttribute("msg", "Erro -> "
                               + e.getMessage());
    }
    finally{
        request.getRequestDispatcher("cadastro.jsp") .
        forward(request, response);
    }
}

//Método executado no comando 'consultar' enviado para o Servlet
protected void consultar(HttpServletRequest request,
                           HttpServletResponse response) throws
                           ServletException, IOException {

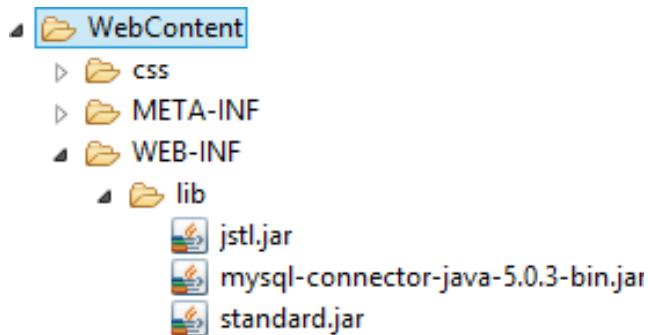
    try{
        ClienteDao d = new ClienteDao();
        //Enviar para de volta para a página a lista
        //com os Clientes obtidos do banco de dados
        request.setAttribute("lista", d.findAll());
        //lista de cliente
        request.setAttribute("msg", "Consulta de Clientes
realizada com sucesso.");
    }
    catch(Exception e){
        request.setAttribute("msg", "Erro -> "
                               + e.getMessage());
    }
    finally{
        request.getRequestDispatcher("consulta.jsp") .
        forward(request, response);
    }
}
}
```

## JSTL - JSP Standard Tag Libraries

Bibliotecas de Tags do Java (JSP) utilizadas para tratamento de dados enviados pelo servidor da aplicação para a página

- **http://java.sun.com/jsp/jstl/core**  
Principal, possui a maioria das tags lógicas (if, foreach, etc...)
- **http://java.sun.com/jsp/jstl/fmt**  
Formatação de dados (Data, Moeda, etc...)
- **http://java.sun.com/jsp/jstl/functions**  
Rotinas lógicas simples (==, >=, <=, !=)
- **http://java.sun.com/jsp/jstl/sql**  
Tags de acesso a banco de dados
- **http://java.sun.com/jsp/jstl/xml**  
Manipulação de conteúdo XML

Incluindo os arquivos .JAR para uso do JSTL



### Utilizando JSTL na página de consulta...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
    prefix="c" %>

<html>
    <head>
        <link rel="stylesheet" type="text/css"
            href="css/estilo.css"/>
    </head>
    <body>
        <div id="conteudo">
            <h3>Consulta de Clientes</h3>
            <a href="index.jsp">Voltar para a página inicial</a>
            <hr/>
            <form name="formconsulta" method="post"
                action="ControleCliente?cmd=consultar">
                <input type="submit"
                    value="Consultar Clientes"/>
            </form>
            <h4> ${msg} </h4>
        <!-- realizar um teste lógico -->
```



```
<c:if test="${lista != null}">



| <u>Código</u> | <u>Nome</u> | <u>Email</u> | <u>Excluir</u> | <u>Editar</u> |
|---------------|-------------|--------------|----------------|---------------|
|---------------|-------------|--------------|----------------|---------------|


```

Resultado...



# Java WebDeveloper - BRQ

Terça-feira, 27 de Maio de 2014

Java na Web, Projeto CRUD. JavaServerPages, Servlets, Expression Languages, JSTL TagLibraries

Aula  
14

## Realizando a Exclusão dos Clientes

Passando o ID por QueryString...

`<a href="ControleCliente?cmd=excluir&id=\${cli.idCliente}">Excluir</a>`

URL gerada...

`http://localhost:8081/javaWebAula04/ControleCliente?cmd=excluir&id=4`

## Realizando a exclusão no Servlet...

```
package control;

import java.io.IOException;

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 persistence.ClienteDao;
import entity.Cliente;

@WebServlet("/ControleCliente")
public class ControleCliente extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleCliente() {
        super();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Cliente cliente = ClienteDao.getClienteById(Integer.parseInt(request.getParameter("id")));
        if (cliente != null) {
            ClienteDao.deleteCliente(cliente);
            response.sendRedirect("index.jsp");
        } else {
            response.sendRedirect("error.jsp");
        }
    }
}
```



# Java WebDeveloper - BRQ

Terça-feira, 27 de Maio de 2014

Java na Web, Projeto CRUD. JavaServerPages, Servlets, Expression Languages, JSTL TagLibraries

Aula  
**14**

```
//doGet -> utilizado quando o servlet
//for chamado por um LINK (URL)
protected void doGet(HttpServletRequest request,
                      HttpServletResponse response) throws
ServletException, IOException {

    String cmd = request.getParameter("cmd");

    if(cmd.equals("excluir")){
        excluir(request, response);
    }
}

//doPost -> utilizado quando o servlet for
//chamado por um FORMULARIO
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response) throws
ServletException, IOException {

    //resgatar a variavel 'cmd' enviada pelos formulários
    String cmd = request.getParameter("cmd");

    if(cmd.equals("cadastrar")){
        cadastrar(request, response);
        //transferir o fluxo de execução
    }
    else if(cmd.equals("consultar")){
        consultar(request, response);
        //transferir o fluxo de execução
    }
}

//Método do Servlet para executar o comando 'cadastrar'
protected void cadastrar(HttpServletRequest request,
                          HttpServletResponse response) throws
ServletException, IOException {

try{

    Cliente c = new Cliente();
    //criando objeto do JavaBean

    c.setNome( request.getParameter("nome") );
    //resgatando o valor do campo 'nome'
    c.setEmail( request.getParameter("email") );
    //resgatamndo o valor do campo 'email'

    ClienteDao d = new ClienteDao();
    d.create(c); //gravando o cliente

    request.setAttribute("msg", "Cliente "
+ c.getNome() + ", cadastrado com sucesso.");
}
}
```



```
        catch(Exception e) {
            request.setAttribute("msg", "Erro -> "
                + e.getMessage());
        }
        finally{
            //redirecionar o fluxo de volta para a página
            request.getRequestDispatcher("cadastro.jsp") .
            forward(request, response);
        }
    }

    //Método executado no comando 'consultar'
    //enviado para o Servlet
    protected void consultar(HttpServletRequest request,
        HttpServletResponse response) throws
        ServletException, IOException {

        try{

            ClienteDao d = new ClienteDao();

            //Enviar para de volta para a página a lista
            //com os Clientes obtidos do banco de dados
            request.setAttribute("lista", d.findAll());
            //lista de cliente
            request.setAttribute("msg", "Consulta de
            Clientes realizada com sucesso.");
        }
        catch(Exception e){
            request.setAttribute("msg", "Erro -> "
                + e.getMessage());
        }
        finally{
            request.getRequestDispatcher("consulta.jsp") .
            forward(request, response);
        }
    }

    //Método para executar o comando 'excluir'
    protected void excluir(HttpServletRequest request,
        HttpServletResponse response) throws
        ServletException, IOException {

        try{

            //resgato o valor da variavel id
            //enviada pela URL (QueryString)
            Integer idCliente = new Integer
                (request.getParameter("id"));

            ClienteDao d = new ClienteDao();

            d.delete(idCliente); //excluir o cliente
        }
    }
}
```



```
        request.setAttribute("msg", "Cliente excluido  
        com sucesso.");  
  
        request.setAttribute("lista", d.findAll());  
        //carregar a lista novamente na página  
    }  
    catch(Exception e){  
        request.setAttribute("msg", "Erro -> "  
                            + e.getMessage());  
    }  
    finally{  
        request.getRequestDispatcher("consulta.jsp").  
        forward(request, response);  
    }  
}  
}
```

### Executando...

The screenshot shows a web browser window with the URL `localhost:8081/javaWebAula04/ControleCliente?cmd=excluir&id=11`. The page title is "Consulta de Clientes". It contains a link to "Voltar para a página inicial" and a button labeled "Consultar Clientes". Below these, a message "Cliente excluido com sucesso." is displayed in blue. A table lists eight clients with columns: Código, Nome, Email, Excluir, and Editar. Each row has a unique ID (1-8) and corresponding values for Name and Email. The "Excluir" and "Editar" buttons are present in each row.

Código	Nome	Email	Excluir	Editar
1	Sergio Mendes	sergio.coti@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
2	Marcone Freitas	marcone@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
3	Patrícia Massaut	patricia@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
4	Deborah Leme	deborah@bol.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
5	Fagner Andre	fagner@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
6	Pedro Miranda	pedro@bol.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
7	Yuri de Souza Vidal	yuri@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
8	Lucas Ribeiro	lucas@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>

Criando a QueryString para editar os dados do Cliente...

`<a href="ControleCliente?cmd=editar&id=\${cli.idCliente}">Editar</a>`

URL gerada...

`http://localhost:8081/javaWebAula04/ControleCliente?cmd=editar&id=4`

Criando a página de edição...



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
        <link rel="stylesheet" type="text/css"
            href="css/estilo.css"/>
    </head>

    <body>

        <div id="conteudo">

            <h3>Edição de Clientes</h3>
            <a href="index.jsp">Voltar</a>
            para a página inicial.
            <hr/>

            <form name="formmedicacao" method="post"
                action="ControleCliente?cmd=atualizar">

                Informe seu Nome: <br/>
                <input type="text" name="nome"
                    required="required" value="${cliente.nome}" />
                <br/><br/>

                Informe seu Email: <br/>
                <input type="email" name="email"
                    required="required" value="${cliente.email}" />
                <br/><br/>

                <!-- campo oculto -->
                <input type="hidden" name="id"
                    value="${cliente.idCliente}" />

                <input type="submit" value="Atualizar Dados
                    de Cliente"/>

            </form>

            <h4> ${msg} </h4>

        </div>

    </body>

</html>
```

Implementando edição e atualização no Servlet...



```
package control;

import java.io.IOException;

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 persistence.ClienteDao;
import entity.Cliente;

@WebServlet("/ControleCliente")
public class ControleCliente extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleCliente() {
        super();
    }

    //doGet -> utilizado quando o servlet for chamado
    //por um LINK (URL)
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response) throws
    ServletException, IOException {

        String cmd = request.getParameter("cmd");

        if(cmd.equals("excluir")){
            excluir(request, response);
        }
        else if(cmd.equals("editar")){
            editar(request, response);
        }
    }

    //doPost -> utilizado quando o servlet for chamado
    //por um FORMULARIO
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response) throws
    ServletException, IOException {

        //resgatar a variavel 'cmd' enviada pelos formulários
        String cmd = request.getParameter("cmd");

        if(cmd.equals("cadastrar")){
            cadastrar(request, response);
            //transferir o fluxo de execução
        }
        else if(cmd.equals("consultar")){
            consultar(request, response);
            //transferir o fluxo de execução
        }
        else if(cmd.equals("atualizar")){
            atualizar(request, response);
            //transferir o fluxo de execução
        }
    }
}
```



```
}

//Método do Servlet para executar o comando 'cadastrar'
protected void cadastrar(HttpServletRequest request,
                           HttpServletResponse response) throws
                           ServletException, IOException {

    try{

        Cliente c = new Cliente();
        //criando objeto do JavaBean

        c.setNome( request.getParameter("nome") );
        //resgatando o valor do campo 'nome'
        c.setEmail( request.getParameter("email") );
        //resgatando o valor do campo 'email'

        ClienteDao d = new ClienteDao();
        d.create(c); //gravando o cliente

        request.setAttribute("msg", "Cliente " + c.getNome()
                            + ", cadastrado com sucesso.");
    }
    catch(Exception e){
        request.setAttribute("msg", "Erro -> "
                            + e.getMessage());
    }
    finally{
        //redirecionar o fluxo de volta para a página
        request.getRequestDispatcher("cadastro.jsp").
        forward(request, response);
    }
}

//Método executado no comando 'consultar' enviado para o Servlet
protected void consultar(HttpServletRequest request,
                           HttpServletResponse response) throws
                           ServletException, IOException {

    try{

        ClienteDao d = new ClienteDao();

        //Enviar para de volta para a página a lista
        //com os Clientes obtidos do banco de dados
        request.setAttribute("lista", d.findAll());
        //lista de cliente
        request.setAttribute("msg", "Consulta de Clientes
                                realizada com sucesso.");
    }
    catch(Exception e){
        request.setAttribute("msg", "Erro -> "
                            + e.getMessage());
    }
    finally{
        request.getRequestDispatcher("consulta.jsp").
        forward(request, response);
    }
}
```

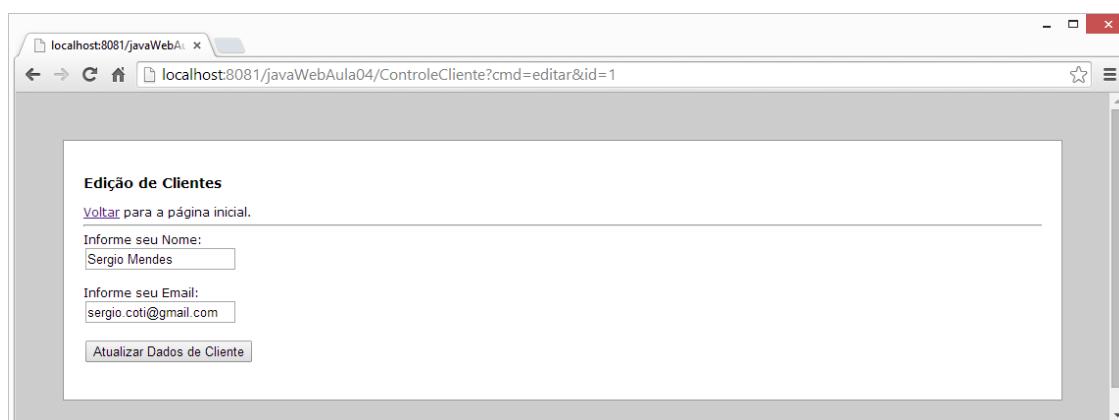


```
//Método para executar o comando 'excluir'  
protected void excluir(HttpServletRequest request,  
                      HttpServletResponse response) throws  
                      ServletException, IOException {  
  
    try{  
  
        //resgato o valor da variavel id enviada  
        //pela URL (QueryString)  
        Integer idCliente = new Integer  
            (request.getParameter("id"));  
  
        ClienteDao d = new ClienteDao();  
        d.delete(idCliente); //excluir o cliente  
  
        request.setAttribute("msg", "Cliente excluido  
                                com sucesso.");  
        request.setAttribute("lista", d.findAll());  
        //carregar a lista novamente na página  
    }  
    catch(Exception e){  
        request.setAttribute("msg", "Erro -> "  
                            + e.getMessage());  
    }  
    finally{  
        request.getRequestDispatcher("consulta.jsp") .  
        forward(request, response);  
    }  
}  
  
//Método para executar o comando 'editar'  
protected void editar(HttpServletRequest request,  
                      HttpServletResponse response) throws  
                      ServletException, IOException {  
  
    try{  
        //resgatar o id do cliente enviado  
        //pela URL (QueryString)  
        Integer idCliente = new Integer(  
            request.getParameter("id"));  
  
        ClienteDao d = new ClienteDao(); //instanciando  
  
        //variavel enviada de volta para a página  
        request.setAttribute("cliente",  
                            d.findById(idCliente));  
    }  
    catch(Exception e){  
        request.setAttribute("msg", "Erro -> "  
                            + e.getMessage());  
    }  
  
    finally{
```



```
        request.getRequestDispatcher("edicao.jsp") .  
        forward(request, response);  
    }  
}  
  
//Método para atualizar os dados do Cliente  
protected void atualizar(HttpServletRequest request,  
    HttpServletResponse response) throws  
    ServletException, IOException {  
  
    try{  
  
        Cliente c = new Cliente();  
  
        c.setIdCliente( new Integer(  
            request.getParameter("id")) );  
        //nome do campo hidden  
        c.setNome( request.getParameter("nome") );  
        //campo nome  
        c.setEmail( request.getParameter("email") );  
        //campo email  
  
        ClienteDao d = new ClienteDao();  
        d.update(c); //atualizar  
  
        request.setAttribute("msg", "Cliente atualizado  
                                com sucesso");  
        request.setAttribute("lista", d.findAll());  
        //carregar a lista novamente  
    }  
    catch(Exception e){  
        request.setAttribute("msg", "Erro -> "  
                            + e.getMessage());  
    }  
    finally{  
  
        request.getRequestDispatcher("consulta.jsp") .  
        forward(request, response);  
    }  
}
```

Executando...





# Java WebDeveloper - BRQ

Terça-feira, 27 de Maio de 2014

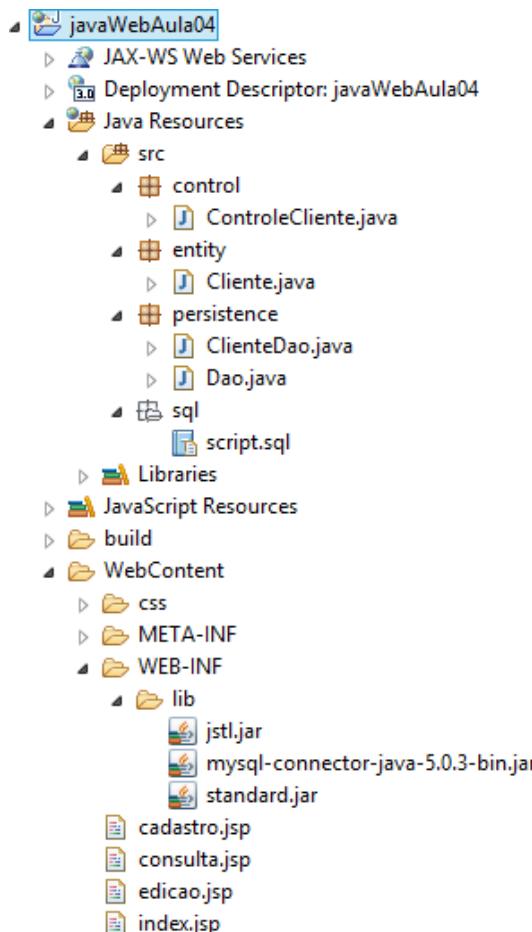
Java na Web, Projeto CRUD. JavaServerPages, Servlets, Expression Languages, JSTL TagLibraries

Aula  
14

The screenshot shows a web browser window with the URL `localhost:8081/javaWebAula04/ControleCliente?cmd=atualizar`. The page title is "Consulta de Clientes". A success message "Cliente atualizado com sucesso" is displayed above a table. The table has columns: Código, Nome, Email, Excluir, and Editar. The data rows are:

Código	Nome	Email	Excluir	Editar
1	Sergio Mendes	sergio.coti@bol.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
7	Yuri de Souza Vidal	yuri@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
8	Lucas Ribeiro	lucas@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>

Estrutura do projeto...





# Java WebDeveloper - BRQ

Quarta-feira, 28 de Maio de 2014

Java na Web, Projeto Login de Usuários. Autenticação, Session, Filters e utilização da biblioteca bootstrap

Aula  
15

Criando o Projeto...

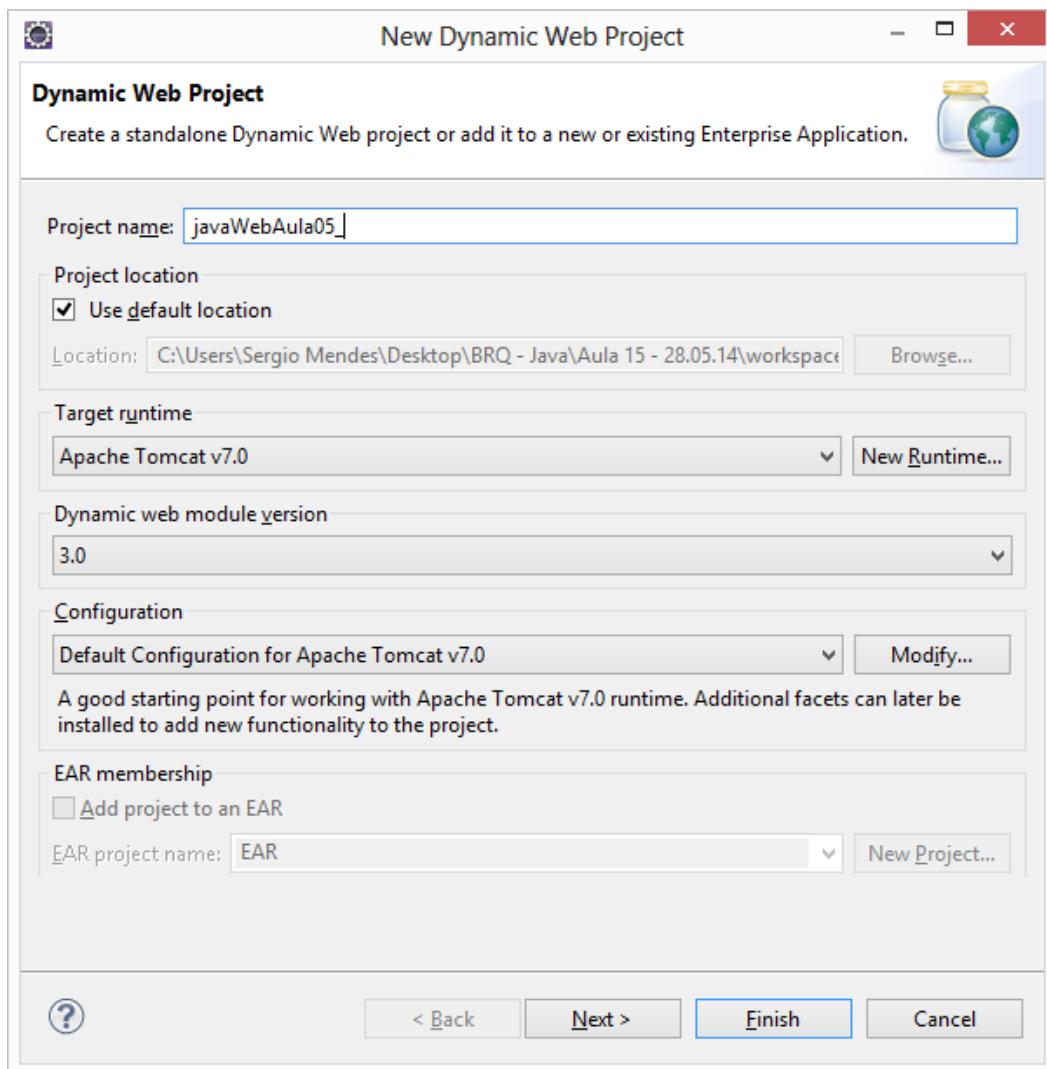
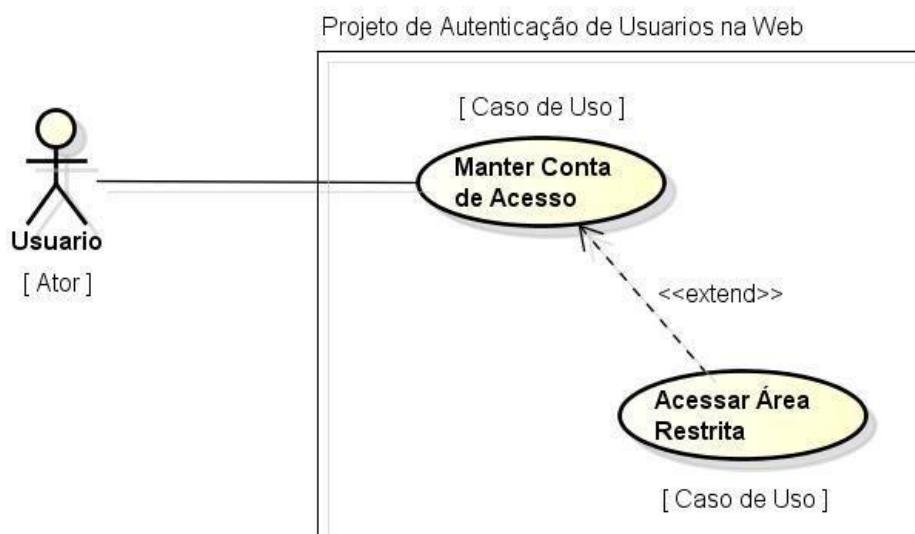


Diagrama de Caso de Uso do Projeto...





Criando a base de dados

**sql\script.sql**

```
drop database if exists aulaweb05;
create database aulaweb05;
use aulaweb05;

create table usuario(
    idusuario          integer      auto_increment,
    nome               varchar(50)   not null,
    login              varchar(20)   not null unique,
    senha              varchar(32)   not null,
    primary key(idusuario));

show tables;

desc usuario;

insert into usuario values(null, 'Deborah Leme',
                           'deborah', md5('123'));
select * from usuario;
```

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> show tables;
+-----+
| Tables_in_aulaweb05 |
+-----+
| usuario           |
+-----+
1 row in set (0.00 sec)

mysql> desc usuario;
+-----+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idusuario | int(11) | NO  | PRI | NULL    | auto_increment |
| nome     | varchar(50) | NO  |     | NULL    |             |
| login    | varchar(20) | NO  | UNI | NULL    |             |
| senha    | varchar(32) | NO  |     | NULL    |             |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.02 sec)

mysql>
```

Criando o JavaBean Usuario...

```
package entity;

public class Usuario {

    private Integer idUsuario;
    private String nome;
    private String login;
    private String senha;
```



```
public Usuario() {  
}  
  
// Construtor com entrada de parametros  
public Usuario(Integer idUsuario, String nome,  
                String login, String senha) {  
    super();  
    this.idUsuario = idUsuario;  
    this.nome = nome;  
    this.login = login;  
    this.senha = senha;  
}  
  
@Override  
public String toString() {  
    return "Usuario [idUsuario=" + idUsuario + ", nome="  
           + nome + ", login=" + login + ", senha="  
           + senha + "]";  
}  
  
public Integer getIdUsuario() {  
    return idUsuario;  
}  
  
public void setIdUsuario(Integer idUsuario) {  
    this.idUsuario = idUsuario;  
}  
  
public String getNome() {  
    return nome;  
}  
  
public void setNome(String nome) {  
    this.nome = nome;  
}  
  
public String getLogin() {  
    return login;  
}  
  
public void setLogin(String login) {  
    this.login = login;  
}  
  
public String getSenha() {  
    return senha;  
}  
  
public void setSenha(String senha) {  
    this.senha = senha;  
}  
}
```



### Criando a Classe Dao **Data Access Object**

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

    protected Connection con;
    protected PreparedStatement stmt;
    protected ResultSet rs;

    protected void open() throws Exception{

        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://
            localhost:3306/aulaweb05", "root", "");
    }

    protected void close() throws Exception{
        con.close();
    }
}
```

Criando a Classe de persistência para a entidade Usuario  
[persistence/UsuarioDao.java](#)

```
package persistence;

import entity.Usuario;

public class UsuarioDao extends Dao{

    //Método para gravar um usuario na base de dados
    public void create(Usuario u) throws Exception{

        open();

        stmt = con.prepareStatement("insert into usuario
            values(null, ?, ?, ?, md5(?))");
        stmt.setString(1, u.getNome());
        stmt.setString(2, u.getLogin());
        stmt.setString(3, u.getSenha());
        stmt.execute();

        close();
    }
}
```



```
//método para buscar 1 Usuario
public Usuario find(String login, String senha)
throws Exception{

    open();

    stmt = con.prepareStatement("select * from usuario
                                where login = ? and senha = md5(?)");
    stmt.setString(1, login);
    stmt.setString(2, senha);
    rs = stmt.executeQuery();
    //executa a consulta e retorna um ResultSet

    Usuario u = null; //NullPointerException

    //Se algum registro for encontrado...
    if(rs.next()){//se ResultSet for verdadeiro

        u = new Usuario( rs.getInt("idusuario"),
                        rs.getString("nome"), rs.getString("login"), null );

    }

    close();
    return u; //retornando o objeto 'Usuario'
}

}
```

## MD5 (Message Digest 5)

Algoritmo para criptografia de dados baseado em chave de 128bits.  
Utilizado para encriptação de dados com o objetivo de não mais descriptografar o seu valor.

```
insert into usuario
values(null, 'Deborah Leme', 'deborah', md5('123'));

select * from usuario;
```

The screenshot shows a Windows command prompt window titled 'cmd' with the path 'C:\WINDOWS\system32\cmd.exe - mysql -u root -p'. The MySQL prompt is visible at the top. Below it, the following commands and their results are shown:

```
mysql> insert into usuario values(null, 'Deborah Leme', 'deborah', md5('123'));
Query OK, 1 row affected (0.03 sec)

mysql> select * from usuario;
+-----+-----+-----+
| idusuario | nome      | login     | senha          |
+-----+-----+-----+
|       1 | Deborah Leme | deborah   | 202cb962ac59075b964b07152d234b70 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```



# Java WebDeveloper - BRQ

Quarta-feira, 28 de Maio de 2014

Java na Web, Projeto Login de Usuários. Autenticação, Session, Filters e utilização da biblioteca bootstrap

Aula  
15

Biblioteca Visual baseada em HTML  
denominada bootstrap...

<http://getbootstrap.com/>

The screenshot shows the official Bootstrap website. At the top, there's a navigation bar with links for 'Bootstrap', 'Getting started', 'CSS', 'Components', 'JavaScript', 'Customize', 'Expo', and 'Blog'. Below the navigation, there's a large purple header with the Bootstrap logo (a white 'B' inside a rounded square) in the center. The text 'The most popular front-end framework for developing responsive, mobile first projects on the web.' is displayed below the logo. A prominent 'Download Bootstrap' button is located in the center of the purple area. At the bottom of this section, it says 'Currently v3.1.1'.

The screenshot shows the 'Getting started' page of the Bootstrap website. The URL in the address bar is 'getbootstrap.com/getting-started/#download'. On the left, there are three main download sections: 'Bootstrap' (with a purple 'Download Bootstrap' button), 'Source code' (with a 'Download source' button), and 'Sass' (with a 'Download Sass' button). Each section includes a brief description. On the right, there's a sidebar with a 'Download' heading and links to various resources like 'What's included', 'Basic template', 'Examples', etc. At the bottom right of the sidebar, there's a 'Back to top' link.

## Download

Bootstrap has a few easy ways to quickly get started, each one appealing to a different skill level and use case. Read through to see what suits your particular needs.

### Bootstrap

Compiled and minified CSS, JavaScript, and fonts. No docs or original source files are included.

[Download Bootstrap](#)

Currently v3.1.1.

### Source code

Source Less, JavaScript, and font files, along with our docs.

Requires a [Less compiler](#) and [some setup](#).

[Download source](#)

### Sass

Bootstrap ported from Less to Sass for easy inclusion in Rails, Compass, or Sass-only projects.

[Download Sass](#)

### Download

[What's included](#)

[Basic template](#)

[Examples](#)

[Community](#)

[Disabling responsiveness](#)

[Migrating from 2.x to 3.0](#)

[Browser and device support](#)

[Third party support](#)

[Accessibility](#)

[License FAQs](#)

[Customizing Bootstrap](#)

[Translations](#)

[Back to top](#)

The screenshot shows a Windows taskbar with several icons. In the center, there's a progress bar for a download from 'https://github.com/twbs/bootstrap/releases/download/v3.1.1/bootstrap-3.1.1-dist.zip'. The progress bar shows approximately 10% completion. The taskbar also includes icons for the Start button, Task View, File Explorer, and other system utilities. The date and time '15/05/2014 09:04' are visible at the bottom right.



# Java WebDeveloper - BRQ

Quarta-feira, 28 de Maio de 2014

Java na Web, Projeto Login de Usuários. Autenticação, Session, Filters e utilização da biblioteca bootstrap

Aula  
15

## JQuery

Biblioteca JavaScript otimizada q com componentes para validação, interface de usuário, ajax, etc...

<http://jquery.com/>

The screenshot shows the official jQuery website at [jquery.com/](http://jquery.com/). The page features a blue header with the jQuery logo and navigation links for Download, API Documentation, Blog, Plugins, and Browser Support. Below the header, there's a section with three icons: a cube labeled "Lightweight Footprint", a stylized 'J' labeled "CSS3 Compliant", and a globe icon labeled "Cross-Browser". To the right, a large orange button says "Download jQuery v1.11.1 or v2.1.1". Further down, a "What is jQuery?" section provides a brief overview, and a "Resources" sidebar lists links to the Core API Documentation and Learning Center. At the bottom, a file manager window shows a download for "bootstrap-3.1.1-dist.zip".

The screenshot shows the "Download jQuery" page at [jquery.com/download/](http://jquery.com/download/). It features a section titled "Downloading jQuery" with text about compressed vs. uncompressed files. Below this, a paragraph explains how to download files by right-clicking and saving. A "jQuery 1.x" section follows, with a note about major changes starting in version 1.9.0. At the bottom, several download links are listed: "Download the compressed, production jQuery 1.11.1" (circled in red), "Download the uncompressed, development jQuery 1.11.1" (also circled in red), "Download the map file for jQuery 1.11.1", and "jQuery 1.11.1 release notes".



### Criando a página inicial do projeto utilizando o bootstrap /index.jsp

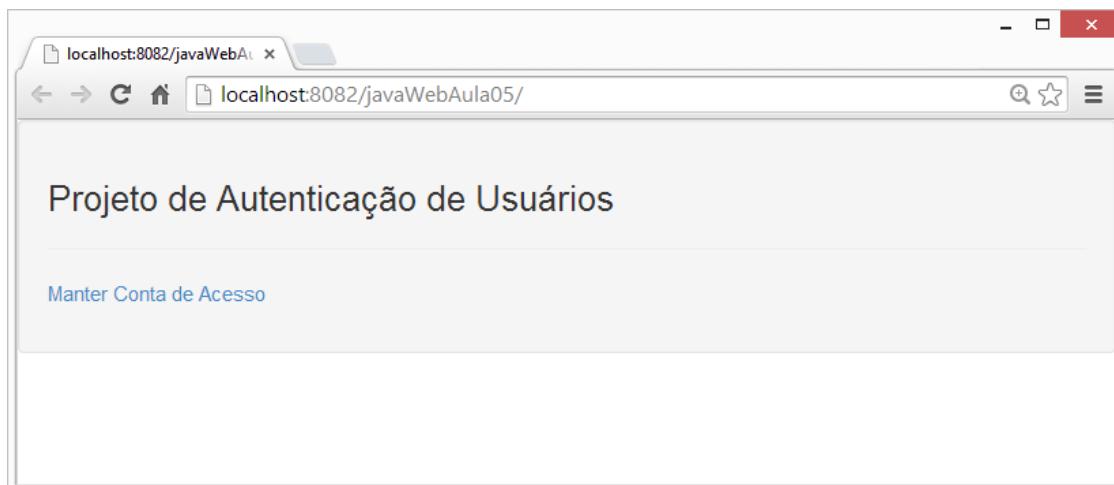
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <link rel="stylesheet" type="text/css"
            href="css/bootstrap.css"/>

        <script type="text/javascript"
            src="js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
            src="js/bootstrap.js"></script>
    </head>
    <body>
        <div class="well">
            <h3>Projeto de Autenticação de Usuários</h3>
            <hr/>
            <p>
                <a href="Login.jsp">
                    Manter Conta de Acesso
                </a>
            </p>
        </div>
    </body>
</html>
```

- Executando...





Criando a página de autenticação e cadastro de usuários...  
**/login.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>

        <link rel="stylesheet" type="text/css"
            href="css/bootstrap.css"/>

        <script type="text/javascript"
            src="js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
            src="js/bootstrap.js"></script>

    </head>

    <body>

        <div class="well">

            <h3>Login de Usuários</h3>

            <form name="formLogin" method="post" action="">

                Login: <input type="text" name="Login"
                    required="required"/>

                Senha: <input type="password" name="senha"
                    required="required"/>

                <button type="submit" class="btn btn-primary">
                    Acessar Sistema
                    <span class="glyphicon glyphicon-play-circle"></span>
                </button>

            </form>

            <p>
                <strong> ${msg} </strong>
            </p>

            <hr/>

        </div>
    </body>
</html>
```



Ainda não possui conta?

```
<button class="btn btn-success"
        data-toggle="modal" data-target="#janela">
    Cadastre-se
    <span class="glyphicon glyphicon-user"></span>
</button>

</div>

<!-- janela modal -->
<div class="modal fade" id="janela">
    <div class="modal-dialog">
        <div class="modal-content">

            <div class="modal-header">
                <h3>Cadastro de Usuário</h3>
            </div>

            <div class="modal-body">

                <form name="formcadastro" method="post" action="">

                    Nome do Usuário: <br/>
                    <input type="text" name="nome" required="required"
                           class="form-control"/>
                    <br/>

                    Login de Acesso: <br/>
                    <input type="text" name="Login" required="required"
                           class="form-control"/>
                    <br/>

                    Senha de Acesso: <br/>
                    <input type="password" name="senha" required="required"
                           class="form-control"/>
                    <br/>

                    <input type="submit" value="Cadastrar Usuário"
                           class="btn btn-success"/>

                </form>
            </div>

            <div class="modal-footer">

                <button class="btn btn-danger" data-dismiss="modal">
                    Fechar Formulário
                </button>
            </div>
        </div>
    </div>
</div>
```

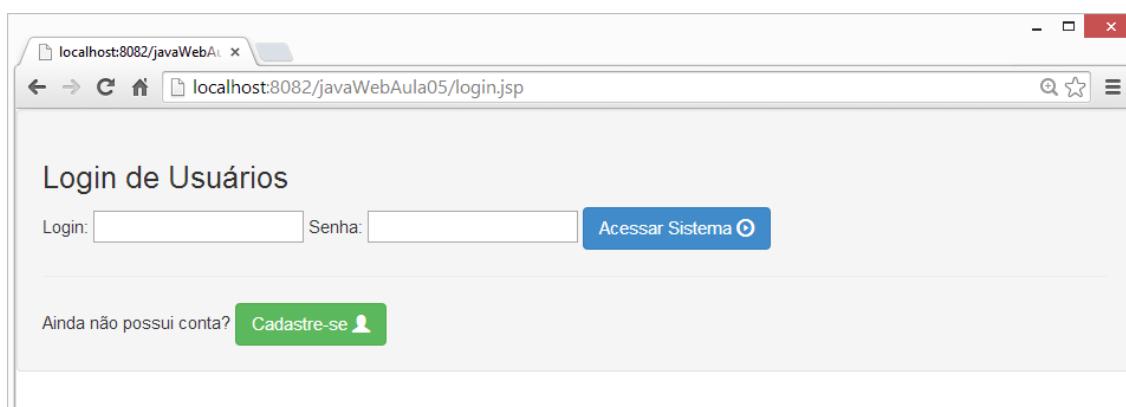


```
</div>

</div>
</div>

</body>
</html>
```

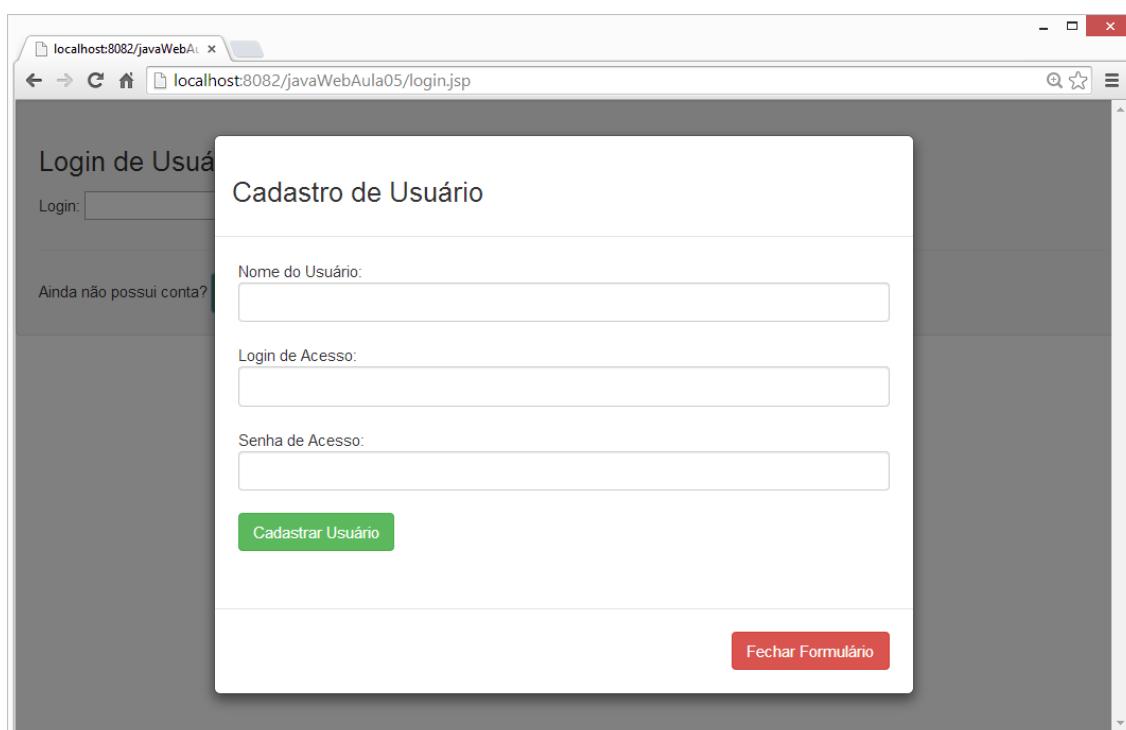
## Resultado...



Login de Usuários

Login:  Senha:  Acessar Sistema

Ainda não possui conta? [Cadastrar-se](#)



Login de Usuários

Cadastro de Usuário

Nome do Usuário:

Login de Acesso:

Senha de Acesso:

Cadastrar Usuário

Fechar Formulário

Criando a página de acesso do usuário autenticado  
**/admin/home.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
```



# Java WebDeveloper - BRQ

Quarta-feira, 28 de Maio de 2014

Java na Web, Projeto Login de Usuários. Autenticação, Session, Filters e utilização da biblioteca bootstrap

Aula  
15

```
pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <link rel="stylesheet" type="text/css"
              href="../css/bootstrap.css"/>

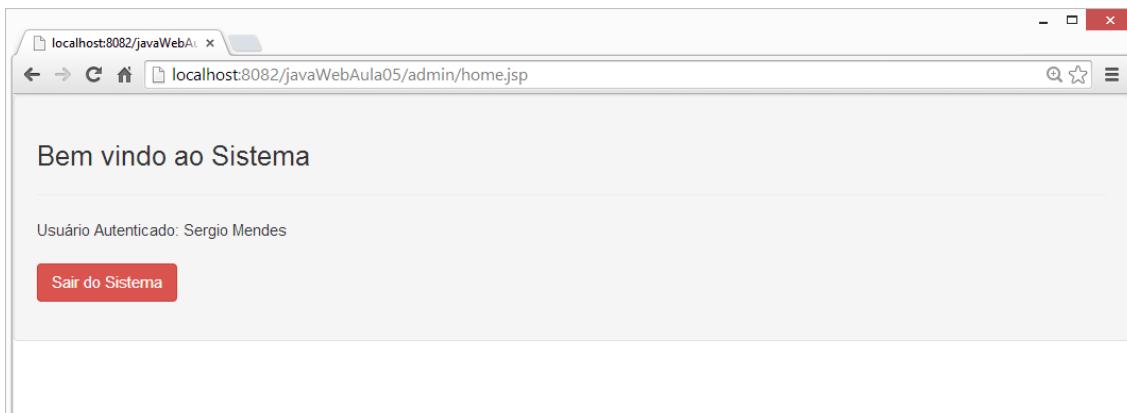
        <script type="text/javascript"
               src="../js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
               src="../js/bootstrap.js"></script>
    </head>
    <body>
        <div class="well">
            <h3> Bem vindo ao Sistema </h3>
            <hr/>

            Usuário Autenticado: ${usuario.nome}
            <!-- imprimindo um dado da sessão!!! -->
            <br/><br/>

            <form name="formLogout" method="post" action=". ">
                <input type="submit" value="Sair do Sistema"
                       class="btn btn-danger"/>
            </form>
        </div>
    </body>
</html>
```

Executando...



Criando a ação de cadastro de usuários...

**action="ControleUsuario?cmd=cadastrar"**



### ControleUsuario.java

```
package control;

import java.io.IOException;

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 javax.servlet.http.HttpSession;

import persistence.UsuarioDao;
import entity.Usuario;

@WebServlet("/ControleUsuario")
public class ControleUsuario extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleUsuario() {
        super();
    }

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

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

        String comando = request.getParameter("cmd");

        if(comando.equals("cadastrar")){
            cadastrar(request, response);
        }
    }

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

        try{
            Usuario u = new Usuario();

            u.setNome( request.getParameter("nome"));
            //campo nome
            u.setLogin(request.getParameter("login"));
        }
    }
}
```



# Java WebDeveloper - BRQ

Quarta-feira, 28 de Maio de 2014

Java na Web, Projeto Login de Usuários. Autenticação, Session, Filters e utilização da biblioteca bootstrap

Aula  
15

```
//campo login
u.setSenha(request.getParameter("senha"));
//campo senha

UsuarioDao d = new UsuarioDao();
d.create(u); //gravando

request.setAttribute("msg", "Usuario
                           cadastrado com sucesso.");
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
                           + e.getMessage());
}
finally{
    request.getRequestDispatcher("login.jsp").
    forward(request, response);
}
}

}
```

## Executando...

Login de Usuários

Login:  Senha:

Ainda não possui conta? [Cadastre-se](#)

Cadastro de Usuário

Nome do Usuário:

Login de Acesso:

Senha de Acesso:

[Cadastrar Usuário](#)

[Fechar Formulário](#)

localhost:8082/javaWebAula05/login.jsp

Login de Usuários

Login:  Senha:

[Acessar Sistema](#)

Usuario cadastrado com sucesso.

Ainda não possui conta? [Cadastre-se](#)

localhost:8082/javaWebAula05/ControleUsuario?cmd=cadastrar



Criando a ação de autenticação de usuários...

***action="ControleUsuario?cmd=autenticar"***

### **ControleUsuario.java**

```
package control;

import java.io.IOException;

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 javax.servlet.http.HttpSession;

import persistence.UsuarioDao;
import entity.Usuario;

@WebServlet("/ControleUsuario")
public class ControleUsuario extends HttpServlet {

    private static final long serialVersionUID = 1L;

    public ControleUsuario() {
        super();
    }

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

    protected void doPost(HttpServletRequest request,
                         HttpServletResponse response) throws
        ServletException, IOException {
        String comando = request.getParameter("cmd");

        if(comando.equals("cadastrar")){
            cadastrar(request, response);
        }
        else if(comando.equals("autenticar")){
            autenticar(request, response);
        }
    }

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



# Java WebDeveloper - BRQ

Quarta-feira, 28 de Maio de 2014

Java na Web, Projeto Login de Usuarios. Autenticação, Session, Filters e utilização da biblioteca bootstrap

Aula  
**15**

```
try{

    Usuario u = new Usuario();

    u.setNome( request.getParameter("nome"));
    u.setLogin(request.getParameter("login"));
    u.setSenha(request.getParameter("senha"));

    UsuarioDao d = new UsuarioDao();
    d.create(u); //gravando

    request.setAttribute("msg", "Usuario
    cadastrado com sucesso.");
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
        + e.getMessage());
}

finally{
    request.getRequestDispatcher("login.jsp").
    forward(request, response);
}

}

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

try{

    //Resgatar o conteudo do campo login
    //e do campo senha
    String login = request.getParameter("login");
    String senha = request.getParameter("senha");

    UsuarioDao d = new UsuarioDao();
    Usuario u = d.find(login, senha);
    //buscando o usuario pelo login e senha

    if( u != null ){
    //se Usuario foi encontrado (diferente de null)

        HttpSession session = request.getSession();
        //chamando a Sessão do navegador
        session.setAttribute("usuario", u);
        //guardando o objeto Usuario na sessão
        //do navegador

        //Abrir a página admin/home.jsp
        response.sendRedirect("admin/home.jsp");
        //não envia valor pra página, só redireciona!
    }

}
```



```
        }
        else{
            //gerando uma exceção -> ERRO!!
            throw new Exception("Acesso Negado.
                                Tente novamente.");
        }
    }
    catch(Exception e){
        request.setAttribute("msg", e.getMessage());
        request.getRequestDispatcher("login.jsp").
        forward(request, response);
    }
}
}
```

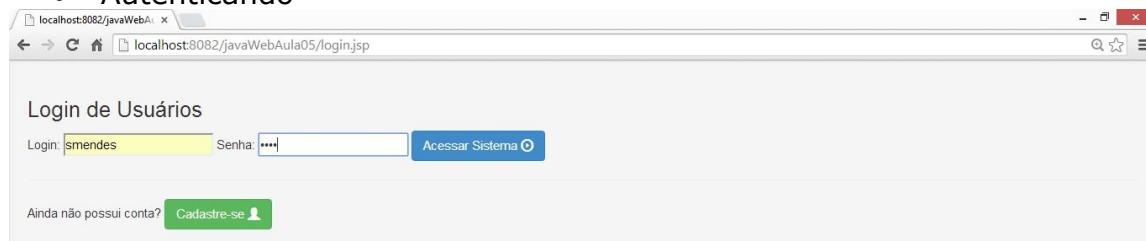
## Sessão (HttpSession)

Área de memória do navegador utilizada para armazenar informações enquanto o browser estiver aberto. Os dados de uma sessão são perdidos quando:

- O navegador é fechado
- A sessão é encerrada por programação
- A sessão expira

### Executando...

- Autenticando

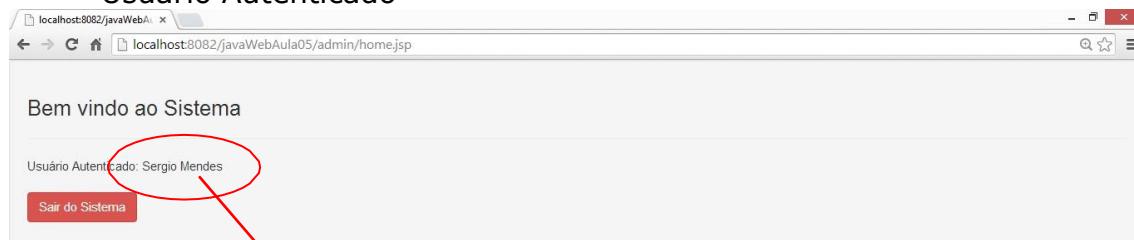


Login de Usuários

Login: smendes Senha:  Acessar Sistema

Ainda não possui conta? [Cadastrar-se](#)

- Usuário Autenticado



Bem vindo ao Sistema

Usuário Autenticado: Sergio Mendes

Sair do Sistema

**`\${usuario.nome}`**

Podemos, através de **EL (Expression Language)** imprimir na página o conteúdo da variável usuário gravada em sessão.



Criando a ação de logout de usuários...

***action="ControleUsuario?cmd=Logout"***

### **ControleUsuario.java**

```
package control;

import java.io.IOException;

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 javax.servlet.http.HttpSession;

import persistence.UsuarioDao;
import entity.Usuario;

@WebServlet("/ControleUsuario")
public class ControleUsuario extends HttpServlet {

    private static final long serialVersionUID = 1L;

    public ControleUsuario() {
        super();
    }

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

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

        String comando = request.getParameter("cmd");

        if(comando.equals("cadastrar")){
            cadastrar(request, response);
        }
        else if(comando.equals("autenticar")){
            autenticar(request, response);
        }
        else if(comando.equals("logout")){
            logout(request, response);
        }
    }

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



```
try{

    Usuario u = new Usuario();

    u.setNome( request.getParameter("nome"));
    u.setLogin(request.getParameter("login"));
    u.setSenha(request.getParameter("senha"));

    UsuarioDao d = new UsuarioDao();
    d.create(u); //gravando

    request.setAttribute("msg", "Usuario cadastrado
                                com sucesso.");
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
                            + e.getMessage());
}

finally{
    request.getRequestDispatcher("login.jsp").
    forward(request, response);
}

}

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

try{

    //Resgatar o conteudo do campo login e do campo senha
    String login = request.getParameter("login");
    String senha = request.getParameter("senha");

    UsuarioDao d = new UsuarioDao();
    Usuario u = d.find(login, senha);
    //buscando o usuario pelo login e senha

    if( u != null ){
        //se Usuario foi encontrado (diferente de null)

        HttpSession session = request.getSession();
        //chamando a Sessão do navegador

        session.setAttribute("usuario", u);
        //guardando o objeto Usuario na
        //sessão do navegador

        //Abrir a página admin/home.jsp
        response.sendRedirect("admin/home.jsp");
        //não envia valor pra página, só redireciona!

    }

    else{
        //gerando uma exceção -> ERRO! !
        throw new Exception("Acesso Negado.
                            Tente novamente.");
    }

}
```



```
        }
        catch(Exception e){
            request.setAttribute("msg", e.getMessage());
            request.getRequestDispatcher("login.jsp").
                forward(request, response);
        }
    }

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

    HttpSession session = request.getSession();
    //chamando o objeto Session do Java

    //destrói uma variável gravada em sessão
    session.removeAttribute("usuario");
    session.invalidate();
    //detona todas as variáveis da sessão!

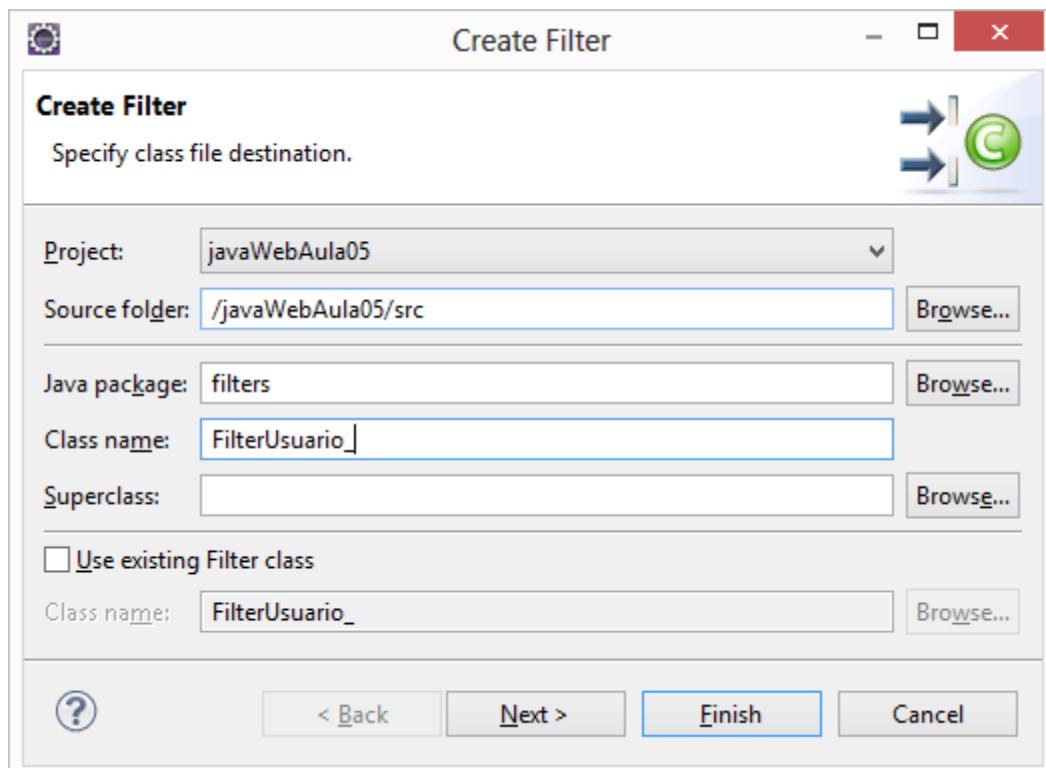
    //redirecionar de volta para login.jsp
    response.sendRedirect("login.jsp");
}
}
```

## Filter (Segurança)

Classe Java utilizada para definir regras de acesso ou de uso a determinados caminhos ou páginas dentro da aplicação.

Regra: Somente poderá acessar o conteúdo da pasta /admin um usuário que estiver autenticado.

Um usuário estará autenticado se estiver gravado em sessão!





```
package filters;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebFilter("/admin/*")
public class FilterUsuario implements Filter {

    public FilterUsuario() {
    }

    public void destroy() {
    }

    //Método onde iremos criar a regra do filter
    public void doFilter(ServletRequest request,
                         ServletResponse response,
                         FilterChain chain) throws IOException,
                         ServletException {
        //chamando a sessão do java

        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;

        HttpSession session = req.getSession();

        if(session.getAttribute("usuario") != null){
            //se o usuario existe na sessão

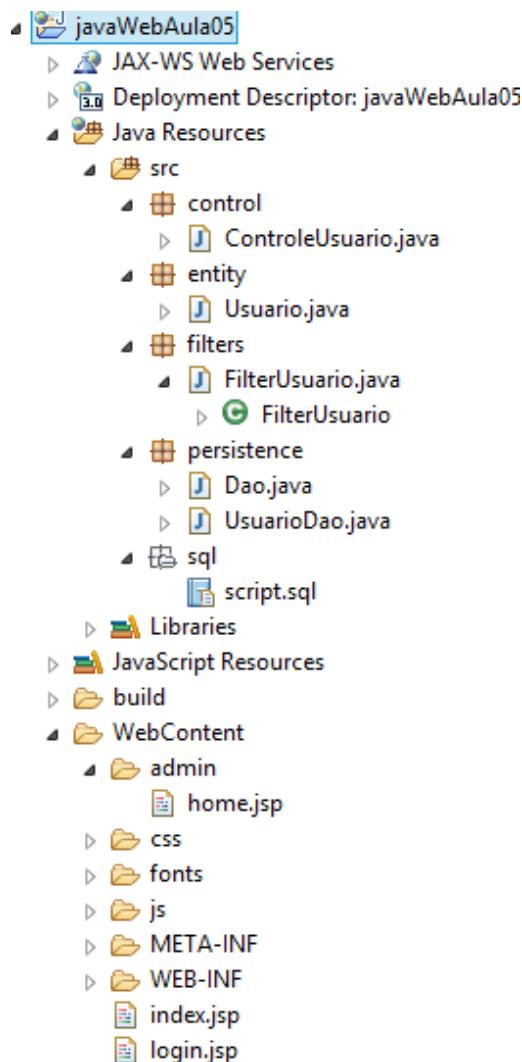
            chain.doFilter(request, response); //tudo certo!!

        }
        else{
            resp.sendRedirect("../login.jsp"); //expulso!!!
        }
    }

    public void init(FilterConfig fConfig) throws ServletException {
    }
}
```

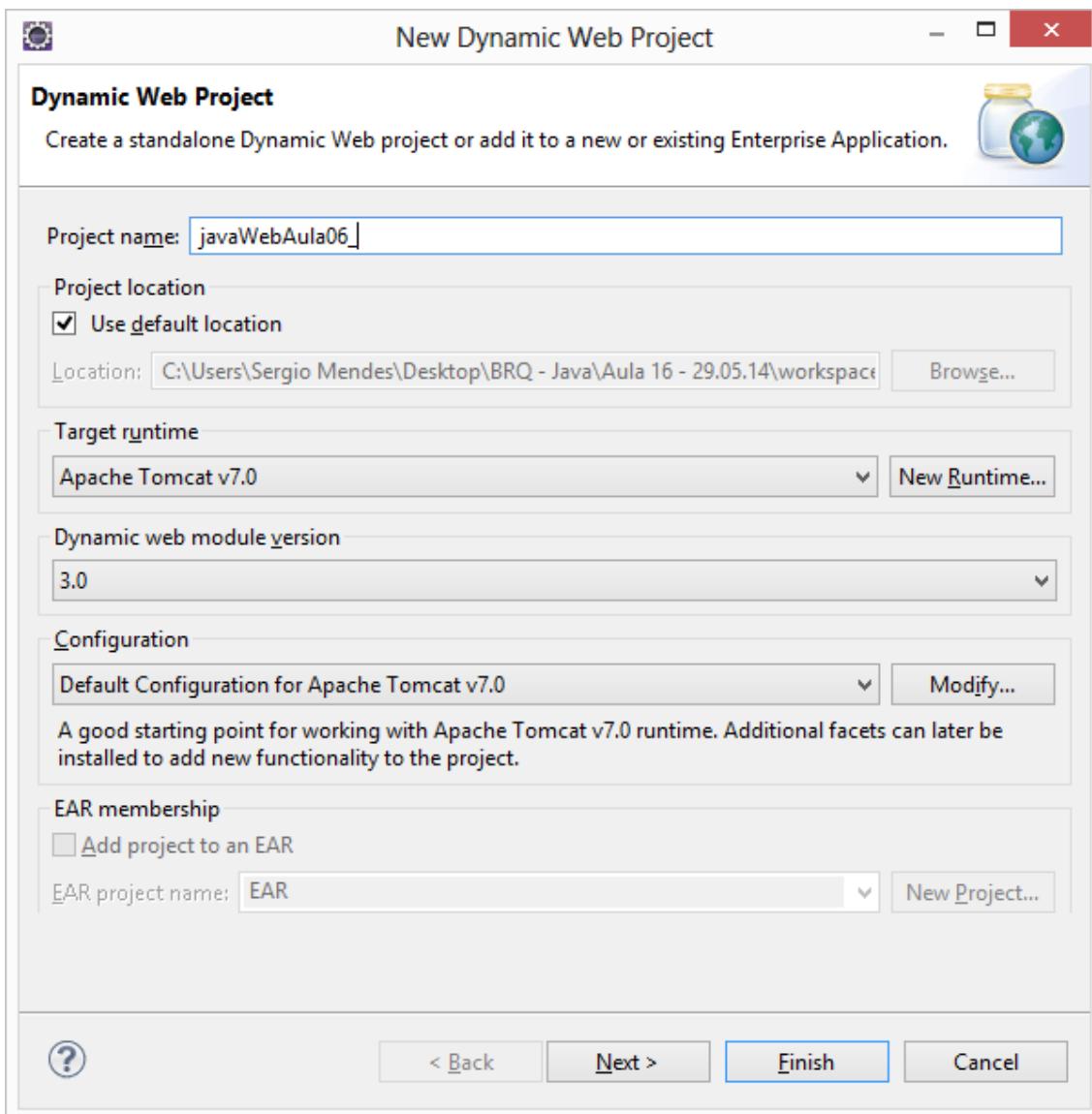


### Estrutura do Projeto...





Criando o Projeto...



Criando a base de dados...

```
drop database if exists aulaweb06;
create database aulaweb06;
use aulaweb06;

create table fornecedor(
    idfornecedor      integer          auto_increment,
    nome              varchar(50)       not null,
    descricao         varchar(255)       not null,
    primary key(idfornecedor));

desc fornecedor;

insert into fornecedor values(null, 'Loja A', 'Artigos de
Informática');
```



# Java WebDeveloper - BRQ

Quinta-feira, 29 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula

16

```
insert into fornecedor values(null, 'Loja B', 'Artigos Eletrônicos');
insert into fornecedor values(null, 'Loja C', 'Artigos de Papelaria');
insert into fornecedor values(null, 'Loja D', 'Console e Games');

select * from fornecedor;

create table produto(
    idproduto           integer          auto_increment,
    nome                varchar(50)      not null,
    preco               double           not null,
    quantidade          integer          not null,
    garantia            varchar(50)      not null,
    infoadicionais      varchar(255)     not null,
    idfornecedor        integer          not null,
    primary key(idproduto),
    foreign key(idfornecedor) references fornecedor(idfornecedor));

desc produto;
```

Modelando as Classes JavaBean...

```
package entity;

public class Produto {

    private Integer idProduto;
    private String nome;
    private Double preco;
    private Integer quantidade;
    private String garantia;
    private String infoAdicionais;

    private Fornecedor fornecedor; // Associação -> TER 1

    public Produto() {
        // Construtor vazio
    }

    public Produto(Integer idProduto, String nome, Double preco,
                   Integer quantidade, String garantia,
                   String infoAdicionais) {
        super();
        this.idProduto = idProduto;
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
        this.garantia = garantia;
        this.infoAdicionais = infoAdicionais;
    }
}
```



```
@Override
public String toString() {
    return "Produto [idProduto=" + idProduto + ", nome="
           + nome + ", preco=" + preco + ", quantidade="
           + quantidade + ", garantia=" + garantia
           + ", infoAdicionais=" + infoAdicionais + "]";
}

public Integer getIdProduto() {
    return idProduto;
}

public void setIdProduto(Integer idProduto) {
    this.idProduto = idProduto;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

public Integer getQuantidade() {
    return quantidade;
}

public void setQuantidade(Integer quantidade) {
    this.quantidade = quantidade;
}

public String getGarantia() {
    return garantia;
}

public void setGarantia(String garantia) {
    this.garantia = garantia;
}

public String getInfoAdicionais() {
    return infoAdicionais;
}
```



```
public void setInfoAdicionais(String infoAdicionais) {
    this.infoAdicionais = infoAdicionais;
}

public Fornecedor getFornecedor() {
    return fornecedor;
}

public void setFornecedor(Fornecedor fornecedor) {
    this.fornecedor = fornecedor;
}

package entity;

import java.util.Collection;

public class Fornecedor {

    private Integer idFornecedor;
    private String nome;
    private String descricao;

    private Collection<Produto> produtos; // coleção de produtos

    public Fornecedor() {
        // Construtor default
    }

    public Fornecedor(Integer idFornecedor, String nome,
                      String descricao) {
        super();
        this.idFornecedor = idFornecedor;
        this.nome = nome;
        this.descricao = descricao;
    }

    @Override
    public String toString() {
        return "Fornecedor [idFornecedor=" + idFornecedor
               + ", nome=" + nome + ", descricao="
               + descricao + "]";
    }

    public Integer getIdFornecedor() {
        return idFornecedor;
    }

    public void setIdFornecedor(Integer idFornecedor) {
        this.idFornecedor = idFornecedor;
    }
}
```



```
public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getDescricao() {
    return descricao;
}

public void setDescricao(String descricao) {
    this.descricao = descricao;
}

public Collection<Produto> getProdutos() {
    return produtos;
}

public void setProdutos(Collection<Produto> produtos) {
    this.produtos = produtos;
}

}
```

### Criando a Classe Dao

Data Access Object

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

    //Constantes
    private static final String DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATABASE = "jdbc:mysql://
                                         localhost:3306/aulaweb06";

    private static final String USER = "root";
    private static final String PASSWORD = "";

    //Atributos para manipulação do banco
    protected Connection con;
    protected PreparedStatement stmt;
    protected ResultSet rs;
```



```
//Método para abrir conexão com o banco de dados
protected void open() throws Exception{
    Class.forName(DRIVER);
    con = DriverManager.getConnection
        (DATABASE, USER, PASSWORD);
}

//Método para fechar conexão com o banco de dados
public void close() throws Exception{
    con.close();
}
```

### Criando a Classe de persistência para Fornecedor...

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Fornecedor;

public class FornecedorDao extends Dao{

    //Método para listar todos os fornecedores da tabela
    public List<Fornecedor> findAll() throws Exception{

        open();

        stmt = con.prepareStatement("select * from fornecedor");
        rs = stmt.executeQuery();

        List<Fornecedor> lista = new ArrayList<Fornecedor>();
        //lista vazia!

        //Enquanto houver registros no ResultSet
        while(rs.next()){ //enquanto o ResultSet for verdadeiro

            Fornecedor f = new Fornecedor(
                rs.getInt("idfornecedor"),
                rs.getString("nome"),
                rs.getString("descricao") );

            lista.add(f); //adiciono o objeto dentro da lista
        }

        close();
        return lista;
    }
}
```



# Java WebDeveloper - BRQ

Quinta-feira, 29 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

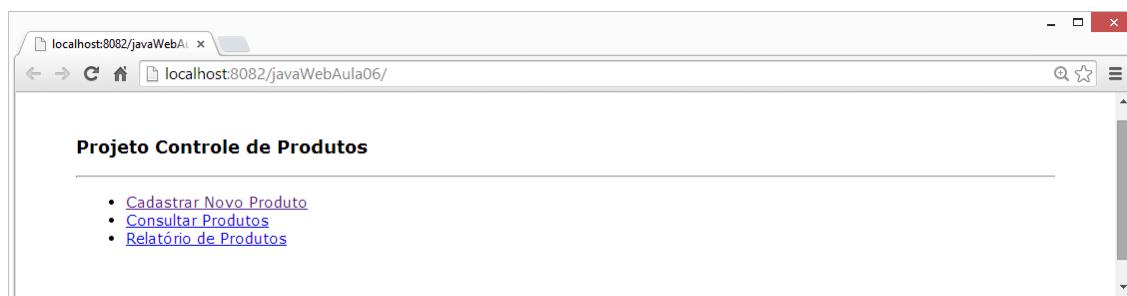
Aula  
**16**

Criando a página inicial do projeto...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <style>
            body
            {
                font-family: verdana;
                font-size: 9pt;
                padding: 40px;
            }
        </style>
    </head>

    <body>
        <h3>Projeto Controle de Produtos</h3>
        <hr/>
        <ul>
            <li>
                <a href="cadastro.jsp">
                    Cadastrar Novo Produto
                </a>
            </li>
            <li>
                <a href="consulta.jsp">
                    Consultar Produtos
                </a>
            </li>
            <li>
                <a href="relatorio.jsp">
                    Relatório de Produtos
                </a>
            </li>
        </ul>
    </body>
</html>
```





### Criando o Bean para retornar a listagem de fornecedores da base de dados...

```
package beans;

import java.util.List;

import persistence.FornecedorDao;
import entity.Fornecedor;

//Classes para retornar dados dos fornecedores para as JSPs
public class FornecedoresBean {

    // Atributo para carregar uma lista com todos
    //os fornecedores do BD
    private List<Fornecedor> listagemFornecedores; //null

    public List<Fornecedor> getListagemFornecedores() {

        try{
            FornecedorDao d = new FornecedorDao(); //persistência
            listagemFornecedores = d.findAll();
        }
        catch(Exception e){
            e.printStackTrace();
            //imprimir no log do servidor o erro
        }

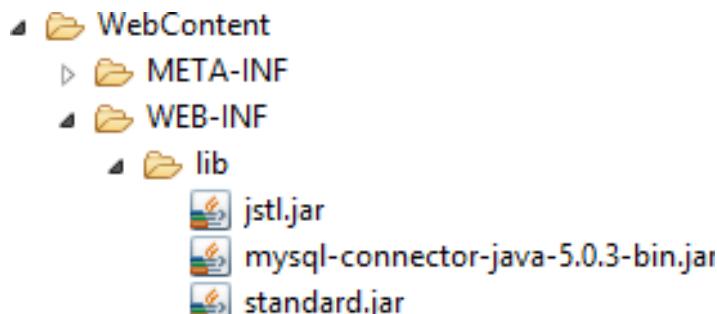
        return listagemFornecedores;
    }

    public void setListagemFornecedores(List<Fornecedor>
                                         listagemFornecedores) {
        this.listagemFornecedores = listagemFornecedores;
    }

}
```

### Incluindo as bibliotecas no Projeto

- MySql Connector
- JSTL - TagLibs





### Criando a página para cadastro de produtos...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core"
    prefix="c" %>

<!-- importar a Classe FornecedoresBean para ser utilizada--&gt;
&lt;jsp:useBean class="beans.FornecedoresBean"
    id="fb" scope="page"/&gt;

&lt;html&gt;

    &lt;head&gt;
        &lt;style&gt;
            body { font-family: verdana;
                    font-size: 9pt; padding: 40px; }
        &lt;/style&gt;
    &lt;/head&gt;

    &lt;body&gt;

        &lt;h3&gt;Cadastro de Produtos&lt;/h3&gt;
        &lt;a href="index.jsp"&gt;Voltar&lt;/a&gt; para a página inicial.
        &lt;hr/&gt;

        &lt;form name="formcadastro" method="post"
            action="ControleProduto?cmd=cadastrar"&gt;

            Nome do Produto: &lt;br/&gt;
            &lt;input type="text" name="nome"
                required="required"/&gt;
            &lt;br/&gt;&lt;br/&gt;

            Preço: &lt;br/&gt;
            &lt;input type="text" name="preco"
                required="required"/&gt;
            &lt;br/&gt;&lt;br/&gt;

            Quantidade: &lt;br/&gt;
            &lt;input type="number" name="quantidade"
                required="required" min="1" max="100"/&gt;
            &lt;br/&gt;&lt;br/&gt;

            Selecione o Fornecedor: &lt;br/&gt;
            &lt;select name="fornecedor" required="required"&gt;
                &lt;option value=""&gt;
                    - Escolha uma Opção -
                &lt;/option&gt;</pre>
```



```
<c:forEach items="${fb.listagemFornecedores}" var="f">

    <option value="${f.idFornecedor}">
        ${f.nome}, ${f.descricao}
    </option>

</c:forEach>

</select>

<br/><br/>

Selecione a Garantia: <br/>

<input type="radio" name="garantia" value="0" checked="checked"/>
    Sem Garantia

<input type="radio" name="garantia" value="30 dias"/>
    30 dias de Garantia

<input type="radio" name="garantia" value="90 dias"/>
    90 dias de Garantia

<br/><br/>

Marque os itens adicionais do produto: <br/>

<input type="checkbox" name="itens" value="Presente"/>
    Embalagem para Presente <br/>

<input type="checkbox" name="itens" value="Cartão"/>
    Cartão Presente <br/>

<input type="checkbox" name="itens" value="Entrega"/>
    Entrega em Domicilio <br/>
<br/>

<input type="submit" value="Cadastrar Produto"/>

</form>

<p>
    <strong> ${msg} </strong>
</p>

</body>

</html>
```



# Java WebDeveloper - BRQ

Quinta-feira, 29 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**16**

## Executando...



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:

Preço:

Quantidade:

Selecione o Fornecedor:

Selecionar a Garantia:  
 Sem Garantia  30 dias de Garantia  90 dias de Garantia

Marque os itens adicionais do produto:  
 Embalagem para Presente  
 Cartão Presente  
 Entrega em Domicílio



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:

Preço:

Quantidade:

Selecione o Fornecedor:

- Escolha uma Opção -

Loja A, Artigos de Informática

Loja B, Artigos Eletrônicos

Loja C, Artigos da Papelaria

Loja D, Consolas e Games

Loja E, Artigos de Vestuário

Cartão Presente

Entrega em Domicílio



## Criando a Classe ProdutoDao

```
package persistence;

import entity.Produto;

public class ProdutoDao extends Dao{

    //Método para gravar um registro de produto na tabela do banco
    public void create(Produto p) throws Exception{}
```



```
        open();

        stmt = con.prepareStatement("insert into produto
                                    values(null, ?, ?, ?, ?, ?, ?, ?)");
        stmt.setString(1, p.getNome());
        stmt.setDouble(2, p.getPreco());
        stmt.setInt(3, p.getQuantidade());
        stmt.setString(4, p.getGarantia());
        stmt.setString(5, p.getInfoAdicionais());
        stmt.setInt(6, p.getFornecedor().getIdFornecedor());
        stmt.execute(); //executar

        close();
    }

}
```

### Criando o Servlet para controle de Produtos...

```
package control;

import java.io.IOException;

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 persistence.ProdutoDao;
import entity.Fornecedor;
import entity.Produto;

@WebServlet("/ControleProduto")
public class ControleProduto extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleProduto() {
        super();
    }

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

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

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            cadastrar(request, response);
        }
    }
}
```



```
protected void cadastrar(HttpServletRequest request,
                         HttpServletResponse response) throws
                         ServletException, IOException {

    try{

        Produto p = new Produto();

        p.setNome( request.getParameter("nome") );

        p.setPreco( new Double(request.getParameter
                               ("preco")) );

        p.setQuantidade( new Integer(request.getParameter
                                      ("quantidade")) );

        p.setGarantia( request.getParameter("garantia") );

        //resgate de um campo de multipla seleção
        //((checkbox) é feito através de gerParameterValues
        //sempre retorna da página um vetor de String
        String itens[] = request.getParameterValues("itens");

        //resgatando os checkboxes marcados
        String infoAdicionais = ""; //variável String vazia

        for(int i = 0; itens != null && i < itens.length;
            i++){ //percorrendo o vetor

            infoAdicionais += itens[i];
            //concatenando o valor do campo marcado

            if(i != itens.length - 1){

                //não é a ultima posição
                infoAdicionais += ", ";
                //concatenando vírgula ao final
            }
        }

        p.setInfoAdicionais(infoAdicionais);

        //Resgatar o id do fornecedor
        Fornecedor f = new Fornecedor();
        f.setIdFornecedor( new Integer(request.
                                         getParameter("fornecedor")) );
        p.setFornecedor(f);
        //relacionando o objeto Fornecedor
        //'f' ao objeto Produto 'p'

        //Realizando a gravação
        ProdutoDao d = new ProdutoDao();
        d.create(p); //salvando...

        request.setAttribute("msg", "Dados do produto
                               gravados com sucesso.");
    }
}
```



# Java WebDeveloper - BRQ

Quinta-feira, 29 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**16**

```
        catch (Exception e) {
            request.setAttribute("msg", "Erro -> "
                + e.getMessage() );
        }
    finally{
        //redirecionamento do fluxo...
        request.getRequestDispatcher("cadastro.jsp").
        forward(request, response);
    }
}
}
```

## Executando...



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:  
Monitor

Preço:  
300

Quantidade:  
10

Selecione o Fornecedor:  
Loja C, Artigos de Papelaria

Selecionar a Garantia:  
 Sem Garantia  30 dias de Garantia  90 dias de Garantia

Marque os itens adicionais do produto:

- Embalagem para Presente
- Cartão Presente
- Entrega em Domicílio

[Cadastrar Produto](#)



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:  
\_\_\_\_\_

Preço:  
\_\_\_\_\_

Quantidade:  
\_\_\_\_\_

Selecione o Fornecedor:  
- Escolha uma Opção -

Selecionar a Garantia:  
 Sem Garantia  30 dias de Garantia  90 dias de Garantia

Marque os itens adicionais do produto:

- Embalagem para Presente
- Cartão Presente
- Entrega em Domicílio

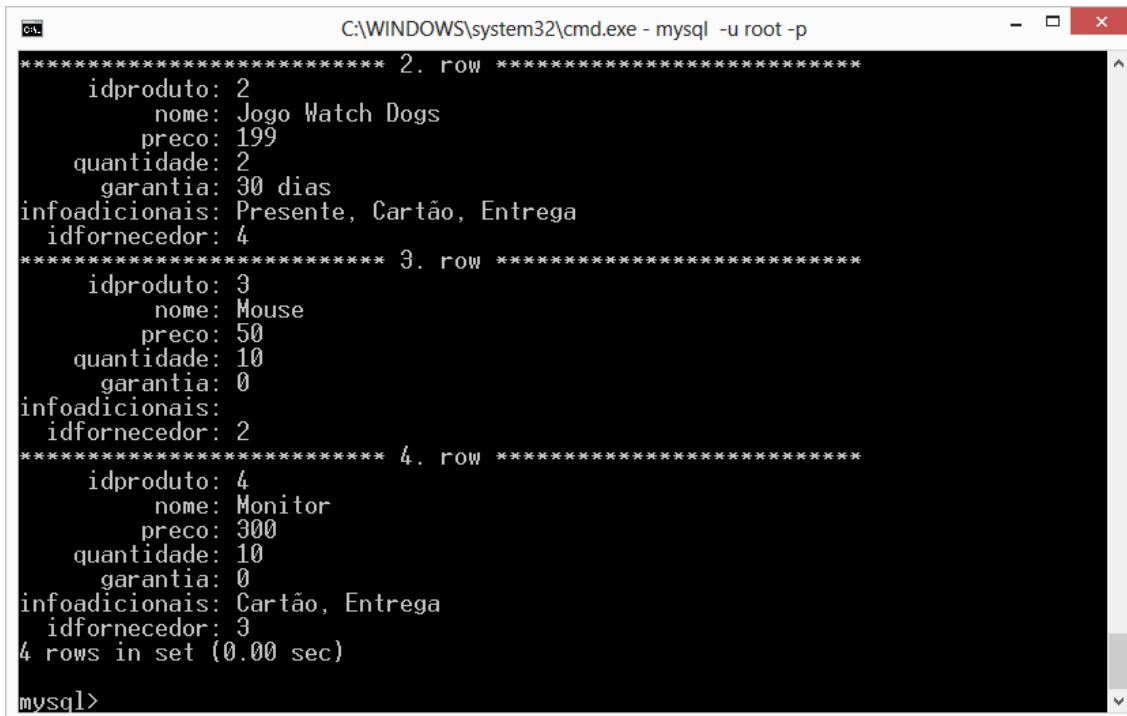
[Cadastrar Produto](#)

**Dados do produto gravados com sucesso.**





No banco de dados...



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
*****
      2. row *****
      idproduto: 2
          nome: Jogo Watch Dogs
          preco: 199
          quantidade: 2
          garantia: 30 dias
infoadicionais: Presente, Cartão, Entrega
      idfornecedor: 4
*****
      3. row *****
      idproduto: 3
          nome: Mouse
          preco: 50
          quantidade: 10
          garantia: 0
infoadicionais:
      idfornecedor: 2
*****
      4. row *****
      idproduto: 4
          nome: Monitor
          preco: 300
          quantidade: 10
          garantia: 0
infoadicionais: Cartão, Entrega
      idfornecedor: 3
4 rows in set (0.00 sec)

mysql>
```

## Criando a View no banco de dados para retornar os dados dos Produtos e Fornecedores...

```
create view estoque
as
select
    p.idproduto      as idproduto,
    p.nome           as produto,
    p.preco          as preco,
    p.quantidade     as quantidade,
    p.garantia       as garantia,
    p.infoadicionais as infoadicionais,
    f.idfornecedor   as idfornecedor,
    f.nome           as fornecedor,
    f.descricao       as descricao
from
    produto as p
inner join fornecedor as f
on p.idfornecedor = f.idfornecedor;

select * from estoque;
```



# Java WebDeveloper - BRQ

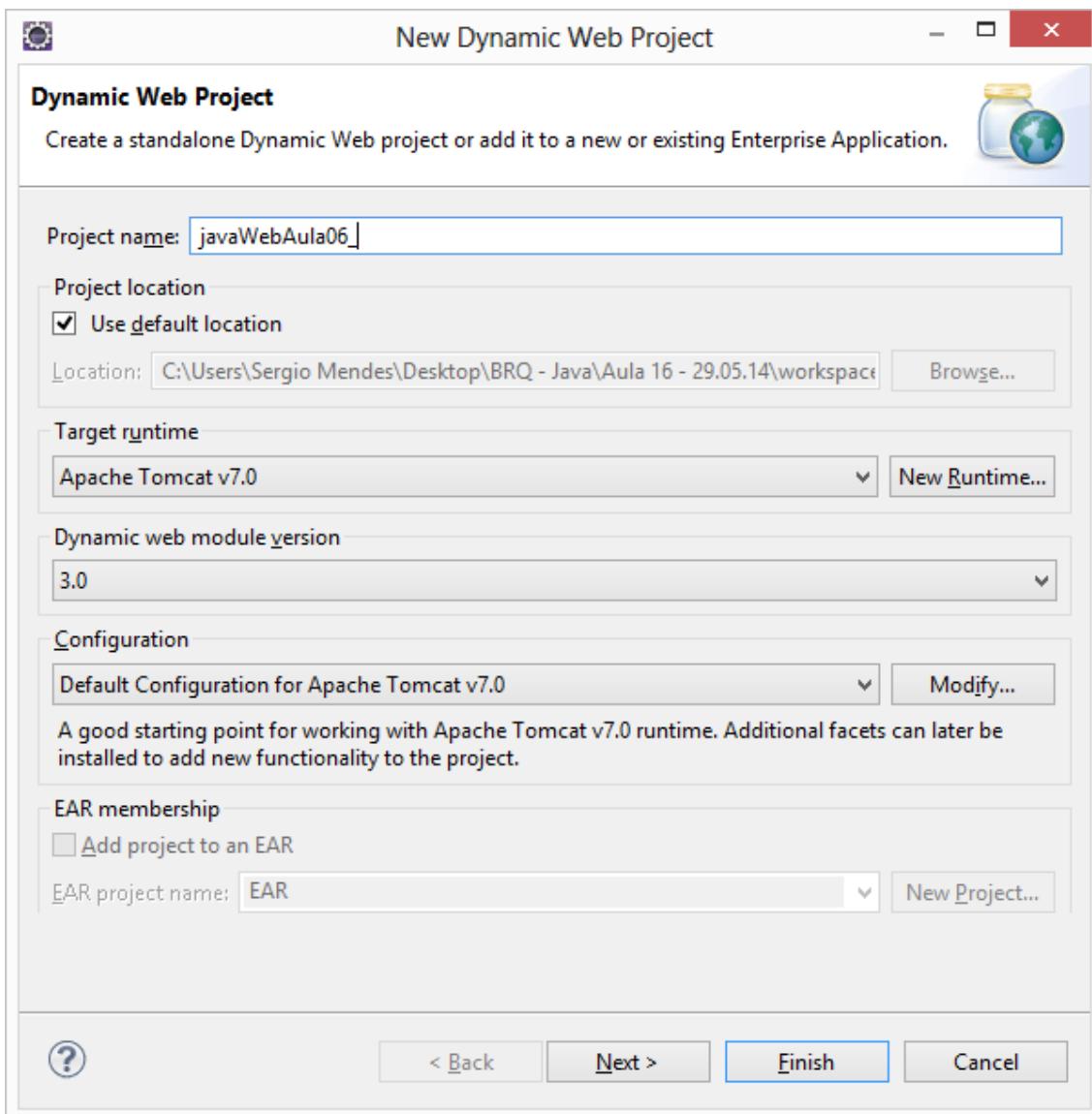
Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula

17

Criando o Projeto...



Criando a base de dados...

```
drop database if exists aulaweb06;
create database aulaweb06;
use aulaweb06;

create table fornecedor(
    idfornecedor      integer          auto_increment,
    nome              varchar(50)       not null,
    descricao         varchar(255)       not null,
    primary key(idfornecedor));

desc fornecedor;

insert into fornecedor values(null, 'Loja A', 'Artigos de
Informática');
```



```
insert into fornecedor values(null, 'Loja B', 'Artigos Eletrônicos');
insert into fornecedor values(null, 'Loja C', 'Artigos de Papelaria');
insert into fornecedor values(null, 'Loja D', 'Console e Games');

select * from fornecedor;

create table produto(
    idproduto           integer          auto_increment,
    nome                varchar(50)      not null,
    preco               double           not null,
    quantidade          integer          not null,
    garantia            varchar(50)      not null,
    infoadiccionais     varchar(255)     not null,
    idfornecedor        integer          not null,
    primary key(idproduto),
    foreign key(idfornecedor) references fornecedor(idfornecedor));

desc produto;
```

### Modelando as Classes JavaBean...

```
package entity;

public class Produto {

    private Integer idProduto;
    private String nome;
    private Double preco;
    private Integer quantidade;
    private String garantia;
    private String infoAdicionais;

    private Fornecedor fornecedor; // Associação -> TER 1

    public Produto() {
        // Construtor vazio
    }

    public Produto(Integer idProduto, String nome, Double preco,
                  Integer quantidade, String garantia,
                  String infoAdicionais) {
        super();
        this.idProduto = idProduto;
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
        this.garantia = garantia;
        this.infoAdicionais = infoAdicionais;
    }
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**17**

```
@Override
public String toString() {
    return "Produto [idProduto=" + idProduto + ", nome="
           + nome + ", preco=" + preco + ", quantidade="
           + quantidade + ", garantia=" + garantia
           + ", infoAdicionais=" + infoAdicionais + "]";
}

public Integer getIdProduto() {
    return idProduto;
}

public void setIdProduto(Integer idProduto) {
    this.idProduto = idProduto;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

public Integer getQuantidade() {
    return quantidade;
}

public void setQuantidade(Integer quantidade) {
    this.quantidade = quantidade;
}

public String getGarantia() {
    return garantia;
}

public void setGarantia(String garantia) {
    this.garantia = garantia;
}

public String getInfoAdicionais() {
    return infoAdicionais;
}
```



```
public void setInfoAdicionais(String infoAdicionais) {
    this.infoAdicionais = infoAdicionais;
}

public Fornecedor getFornecedor() {
    return fornecedor;
}

public void setFornecedor(Fornecedor fornecedor) {
    this.fornecedor = fornecedor;
}

package entity;

import java.util.Collection;

public class Fornecedor {

    private Integer idFornecedor;
    private String nome;
    private String descricao;

    private Collection<Produto> produtos; // coleção de produtos

    public Fornecedor() {
        // Construtor default
    }

    public Fornecedor(Integer idFornecedor, String nome,
                      String descricao) {
        super();
        this.idFornecedor = idFornecedor;
        this.nome = nome;
        this.descricao = descricao;
    }

    @Override
    public String toString() {
        return "Fornecedor [idFornecedor=" + idFornecedor
               + ", nome=" + nome + ", descricao="
               + descricao + "]";
    }

    public Integer getIdFornecedor() {
        return idFornecedor;
    }

    public void setIdFornecedor(Integer idFornecedor) {
        this.idFornecedor = idFornecedor;
    }
}
```



```
public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getDescricao() {
    return descricao;
}

public void setDescricao(String descricao) {
    this.descricao = descricao;
}

public Collection<Produto> getProdutos() {
    return produtos;
}

public void setProdutos(Collection<Produto> produtos) {
    this.produtos = produtos;
}

}
```

### Criando a Classe Dao

Data Access Object

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

    //Constantes
    private static final String DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATABASE = "jdbc:mysql://
                                         localhost:3306/aulaweb06";

    private static final String USER = "root";
    private static final String PASSWORD = "";

    //Atributos para manipulação do banco
    protected Connection con;
    protected PreparedStatement stmt;
    protected ResultSet rs;
```



```
//Método para abrir conexão com o banco de dados
protected void open() throws Exception{
    Class.forName(DRIVER);
    con = DriverManager.getConnection
        (DATABASE, USER, PASSWORD);
}

//Método para fechar conexão com o banco de dados
public void close() throws Exception{
    con.close();
}
```

### Criando a Classe de persistência para Fornecedor...

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Fornecedor;

public class FornecedorDao extends Dao{

    //Método para listar todos os fornecedores da tabela
    public List<Fornecedor> findAll() throws Exception{

        open();

        stmt = con.prepareStatement("select * from fornecedor");
        rs = stmt.executeQuery();

        List<Fornecedor> lista = new ArrayList<Fornecedor>();
        //lista vazia!

        //Enquanto houver registros no ResultSet
        while(rs.next()){ //enquanto o ResultSet for verdadeiro

            Fornecedor f = new Fornecedor(
                rs.getInt("idfornecedor"),
                rs.getString("nome"),
                rs.getString("descricao") );

            lista.add(f); //adiciono o objeto dentro da lista
        }

        close();
        return lista;
    }
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula

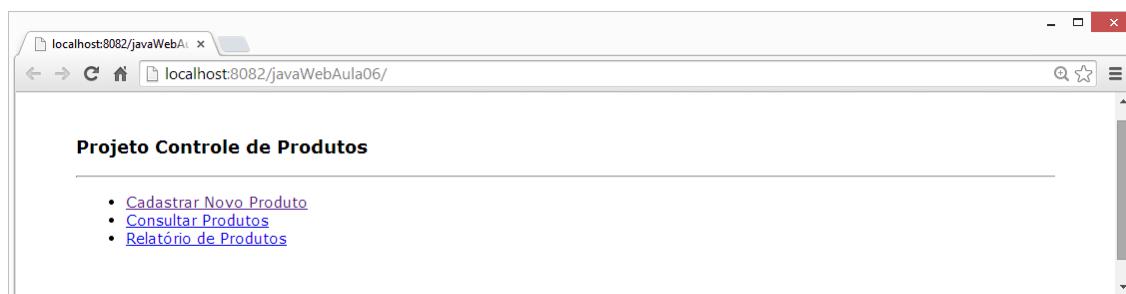
17

Criando a página inicial do projeto...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <style>
            body
            {
                font-family: verdana;
                font-size: 9pt;
                padding: 40px;
            }
        </style>
    </head>

    <body>
        <h3>Projeto Controle de Produtos</h3>
        <hr/>
        <ul>
            <li>
                <a href="cadastro.jsp">
                    Cadastrar Novo Produto
                </a>
            </li>
            <li>
                <a href="consulta.jsp">
                    Consultar Produtos
                </a>
            </li>
            <li>
                <a href="relatorio.jsp">
                    Relatório de Produtos
                </a>
            </li>
        </ul>
    </body>
</html>
```





### Criando o Bean para retornar a listagem de fornecedores da base de dados...

```
package beans;

import java.util.List;

import persistence.FornecedorDao;
import entity.Fornecedor;

//Classes para retornar dados dos fornecedores para as JSPs
public class FornecedoresBean {

    // Atributo para carregar uma lista com todos
    //os fornecedores do BD
    private List<Fornecedor> listagemFornecedores; //null

    public List<Fornecedor> getListagemFornecedores() {

        try{
            FornecedorDao d = new FornecedorDao(); //persistência
            listagemFornecedores = d.findAll();
        }
        catch(Exception e){
            e.printStackTrace();
            //imprimir no log do servidor o erro
        }

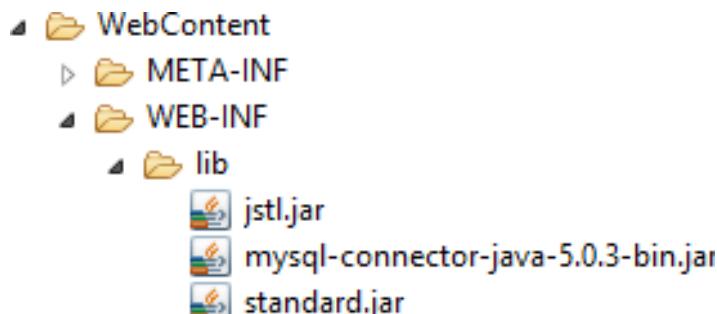
        return listagemFornecedores;
    }

    public void setListagemFornecedores(List<Fornecedor>
                                         listagemFornecedores) {
        this.listagemFornecedores = listagemFornecedores;
    }

}
```

### Incluindo as bibliotecas no Projeto

- MySql Connector
- JSTL - TagLibs





### Criando a página para cadastro de produtos...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core"
    prefix="c" %>

<!-- importar a Classe FornecedoresBean para ser utilizada--&gt;
&lt;jsp:useBean class="beans.FornecedoresBean"
    id="fb" scope="page"/&gt;

&lt;html&gt;

    &lt;head&gt;
        &lt;style&gt;
            body { font-family: verdana;
                    font-size: 9pt; padding: 40px; }
        &lt;/style&gt;
    &lt;/head&gt;

    &lt;body&gt;

        &lt;h3&gt;Cadastro de Produtos&lt;/h3&gt;
        &lt;a href="index.jsp"&gt;Voltar&lt;/a&gt; para a página inicial.
        &lt;hr/&gt;

        &lt;form name="formcadastro" method="post"
            action="ControleProduto?cmd=cadastrar"&gt;

            Nome do Produto: &lt;br/&gt;
            &lt;input type="text" name="nome"
                required="required"/&gt;
            &lt;br/&gt;&lt;br/&gt;

            Preço: &lt;br/&gt;
            &lt;input type="text" name="preco"
                required="required"/&gt;
            &lt;br/&gt;&lt;br/&gt;

            Quantidade: &lt;br/&gt;
            &lt;input type="number" name="quantidade"
                required="required" min="1" max="100"/&gt;
            &lt;br/&gt;&lt;br/&gt;

            Selecione o Fornecedor: &lt;br/&gt;
            &lt;select name="fornecedor" required="required"&gt;
                &lt;option value=""&gt;
                    - Escolha uma Opção -
                &lt;/option&gt;</pre>
```



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula

17

```
<c:forEach items="${fb.listagemFornecedores}" var="f">

    <option value="${f.idFornecedor}">
        ${f.nome}, ${f.descricao}
    </option>

</c:forEach>

</select>

<br/><br/>

Selecione a Garantia: <br/>

<input type="radio" name="garantia" value="0" checked="checked"/>
    Sem Garantia

<input type="radio" name="garantia" value="30 dias"/>
    30 dias de Garantia

<input type="radio" name="garantia" value="90 dias"/>
    90 dias de Garantia

<br/><br/>

Marque os itens adicionais do produto: <br/>

<input type="checkbox" name="itens" value="Presente"/>
    Embalagem para Presente <br/>

<input type="checkbox" name="itens" value="Cartão"/>
    Cartão Presente <br/>

<input type="checkbox" name="itens" value="Entrega"/>
    Entrega em Domicilio <br/>
<br/>

<input type="submit" value="Cadastrar Produto"/>

</form>

<p>
    <strong> ${msg} </strong>
</p>

</body>

</html>
```



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**17**

## Executando...



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:

Preço:

Quantidade:

Selecione o Fornecedor:

Selecione a Garantia:  
 Sem Garantia  30 dias de Garantia  90 dias de Garantia

Marque os itens adicionais do produto:  
 Embalagem para Presente  
 Cartão Presente  
 Entrega em Domicílio



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:

Preço:

Quantidade:

Selecione o Fornecedor:

- Escolha uma Opção -

Loja A, Artigos de Informática

Loja B, Artigos Eletrônicos

Loja C, Artigos da Papelaria

Loja D, Consolas e Games

Loja E, Artigos de Vestuário

Cartão Presente

Entrega em Domicílio



## Criando a Classe ProdutoDao

```
package persistence;

import entity.Produto;

public class ProdutoDao extends Dao{

    //Método para gravar um registro de produto na tabela do banco
    public void create(Produto p) throws Exception{}
```



```
        open();

        stmt = con.prepareStatement("insert into produto
                                   values(null, ?, ?, ?, ?, ?, ?, ?)");
        stmt.setString(1, p.getNome());
        stmt.setDouble(2, p.getPreco());
        stmt.setInt(3, p.getQuantidade());
        stmt.setString(4, p.getGarantia());
        stmt.setString(5, p.getInfoAdicionais());
        stmt.setInt(6, p.getFornecedor().getIdFornecedor());
        stmt.execute(); //executar

        close();
    }

}
```

### Criando o Servlet para controle de Produtos...

```
package control;

import java.io.IOException;

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 persistence.ProdutoDao;
import entity.Fornecedor;
import entity.Produto;

@WebServlet("/ControleProduto")
public class ControleProduto extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleProduto() {
        super();
    }

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

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

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            cadastrar(request, response);
        }
    }
}
```



```
protected void cadastrar(HttpServletRequest request,
                         HttpServletResponse response) throws
                         ServletException, IOException {

    try{

        Produto p = new Produto();

        p.setNome( request.getParameter("nome") );

        p.setPreco( new Double(request.getParameter
                               ("preco")) );

        p.setQuantidade( new Integer(request.getParameter
                                      ("quantidade")) );

        p.setGarantia( request.getParameter("garantia") );

        //resgate de um campo de multipla seleção
        //((checkbox) é feito através de gerParameterValues
        //sempre retorna da página um vetor de String
        String itens[] = request.getParameterValues("itens");

        //resgatando os checkboxes marcados
        String infoAdicionais = ""; //variável String vazia

        for(int i = 0; itens != null && i < itens.length;
            i++){ //percorrendo o vetor

            infoAdicionais += itens[i];
            //concatenando o valor do campo marcado

            if(i != itens.length - 1){

                //não é a ultima posição
                infoAdicionais += ", ";
                //concatenando vírgula ao final
            }
        }

        p.setInfoAdicionais(infoAdicionais);

        //Resgatar o id do fornecedor
        Fornecedor f = new Fornecedor();
        f.setIdFornecedor( new Integer(request.
                                       getParameter("fornecedor")) );
        p.setFornecedor(f);
        //relacionando o objeto Fornecedor
        //'f' ao objeto Produto 'p'

        //Realizando a gravação
        ProdutoDao d = new ProdutoDao();
        d.create(p); //salvando...

        request.setAttribute("msg", "Dados do produto
                               gravados com sucesso.");
    }
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**17**

```
        catch (Exception e) {
            request.setAttribute("msg", "Erro -> "
                + e.getMessage() );
        }
    finally{
        //redirecionamento do fluxo...
        request.getRequestDispatcher("cadastro.jsp").
        forward(request, response);
    }
}
}
```

## Executando...



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:  
Monitor

Preço:  
300

Quantidade:  
10

Selecione o Fornecedor:  
Loja C, Artigos de Papelaria

Selecione a Garantia:  
 Sem Garantia  30 dias de Garantia  90 dias de Garantia

Marque os itens adicionais do produto:

- Embalagem para Presente
- Cartão Presente
- Entrega em Domicílio

[Cadastrar Produto](#)



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Nome do Produto:

Preço:

Quantidade:

Selecione o Fornecedor:  
- Escolha uma Opção -

Selecione a Garantia:  
 Sem Garantia  30 dias de Garantia  90 dias de Garantia

Marque os itens adicionais do produto:

- Embalagem para Presente
- Cartão Presente
- Entrega em Domicílio

[Cadastrar Produto](#)

Dados do produto gravados com sucesso.





No banco de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
*****
      2. row *****
  idproduto: 2
      nome: Jogo Watch Dogs
      preco: 199
  quantidade: 2
  garantia: 30 dias
infoadicionais: Presente, Cartão, Entrega
  idfornecedor: 4
*****
      3. row *****
  idproduto: 3
      nome: Mouse
      preco: 50
  quantidade: 10
  garantia: 0
infoadicionais:
  idfornecedor: 2
*****
      4. row *****
  idproduto: 4
      nome: Monitor
      preco: 300
  quantidade: 10
  garantia: 0
infoadicionais: Cartão, Entrega
  idfornecedor: 3
4 rows in set (0.00 sec)

mysql>
```

**Criando a View no banco de dados para retornar os dados dos Produtos e Fornecedores...**

```
create view estoque
as
select
    p.idproduto          as idproduto,
    p.nome               as produto,
    p.preco              as preco,
    p.quantidade         as quantidade,
    p.garantia           as garantia,
    p.infoadicionais     as infoadicionais,
    f.idfornecedor       as idfornecedor,
    f.nome               as fornecedor,
    f.descricao          as descricao
from
    produto as p
inner join fornecedor as f
on p.idfornecedor = f.idfornecedor;

select * from estoque;
```



Consultando os dados obtidos através da View...

```
package persistence;

import java.util.ArrayList;
import java.util.List;

import entity.Fornecedor;
import entity.Produto;

public class ProdutoDao extends Dao{

    //Método para gravar um registro de produto na tabela do banco
    public void create(Produto p) throws Exception{

        open();

        stmt = con.prepareStatement("insert into produto
                                    values(null, ?, ?, ?, ?, ?, ?)");
        stmt.setString(1, p.getNome());
        stmt.setDouble(2, p.getPreco());
        stmt.setInt(3, p.getQuantidade());
        stmt.setString(4, p.getGarantia());
        stmt.setString(5, p.getInfoAdicionais());
        stmt.setInt(6, p.getFornecedor().getIdFornecedor());
        stmt.execute(); //executar

        close();
    }

    //Método para listar todos os registros obtidos através da View
    public List<Produto> findAll() throws Exception{

        open();

        stmt = con.prepareStatement("select * from estoque");
        rs = stmt.executeQuery();
        //executa a consulta e retorna os
        //registros para o ResultSet

        List<Produto> lista = new ArrayList<Produto>();
        //lista vazia

        while(rs.next()){ //enquanto houver registros

            Produto p = new Produto(
                rs.getInt("idproduto"),
                rs.getString("produto"),
                rs.getDouble("preco"),
                rs.getInt("quantidade"),
                rs.getString("garantia"),
                rs.getString("infoadicionais"));

            //objeto javabean

            Fornecedor f = new Fornecedor(
                rs.getInt("idfornecedor"),
                rs.getString("fornecedor"),
                rs.getString("cnpj"),
                rs.getString("razaoSocial"),
                rs.getString("fantasia"),
                rs.getString("email"),
                rs.getString("telefone"),
                rs.getString("celular"),
                rs.getString("cep"),
                rs.getString("logradouro"),
                rs.getString("numero"),
                rs.getString("complemento"),
                rs.getString("bairro"),
                rs.getString("cidade"),
                rs.getString("uf"),
                rs.getString("pais"),
                rs.getString("observacao"));
        }
    }
}
```



```
        rs.getString("descricao"));
        //objeto javabean

        p.setFornecedor(f);
        //relacionando o fornecedor ao produto

        //adicionar o produto na lista
        lista.add(p);
    }

    close();
    return lista; //retornar a lista
}

}
```

### Criando a Classe Bean para retornar os dados dos Produtos...

```
package beans;

import java.util.List;

import persistence.ProdutoDao;
import entity.Produto;

public class ProdutosBean {

    private List<Produto> listagemProdutos; // null

    public List<Produto> getListagemProdutos() {

        try{

            ProdutoDao d = new ProdutoDao(); //persistencia
            listagemProdutos = d.findAll(); //listar todos
        }
        catch(Exception e){
            e.printStackTrace();
            //imprimir o erro no console do tomcat
        }

        return listagemProdutos; //retornar a lista
    }

    public void setListagemProdutos(List<Produto> listagemProdutos)
    {
        this.listagemProdutos = listagemProdutos;
    }

}
```



### Exibindo os Produtos na página através de JSTL...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!-- Importar o conteúdo da JSTL taglib Java -->
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!-- Importar a Classe ProdutosBean -->
<jsp:useBean class="beans.ProdutosBean" id="pb" scope="page"/>

<html>

    <head>
        <style>
            body { font-family: verdana;
                    font-size: 9pt;
                    padding: 40px; }
        </style>
    </head>

    <body>

        <h3>Relação de Produtos em Estoque</h3>
        <a href="index.jsp">Voltar para a página inicial
        <hr/>

        <table border="1" style="width: 100%">

            <thead>
                <tr>
                    <th>Código</th>
                    <th>Produto</th>
                    <th>Preço</th>
                    <th>Quantidade</th>
                    <th>Total</th>
                    <th>Garantia</th>
                    <th>Informações</th>
                    <th>Fornecedor</th>
                    <th>Descrição</th>
                </tr>
            </thead>

            <tbody>

                <c:forEach items="${pb.listagemProdutos}" var="p">

                    <tr>
                        <td> ${p.idProduto} </td>
                        <td> ${p.nome} </td>
```



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**17**

```
<td> ${p.preco} </td>
<td> ${p.quantidade} </td>
<td> ${p.preco * p.quantidade} </td>
<td> ${p.garantia} </td>
<td> ${p.infoAdicionais} </td>
<td> ${p.fornecedor.nome} </td>
<td> ${p.fornecedor.descricao} </td>

</tr>

</c:forEach>

</tbody>

</table>

</body>

</html>
```

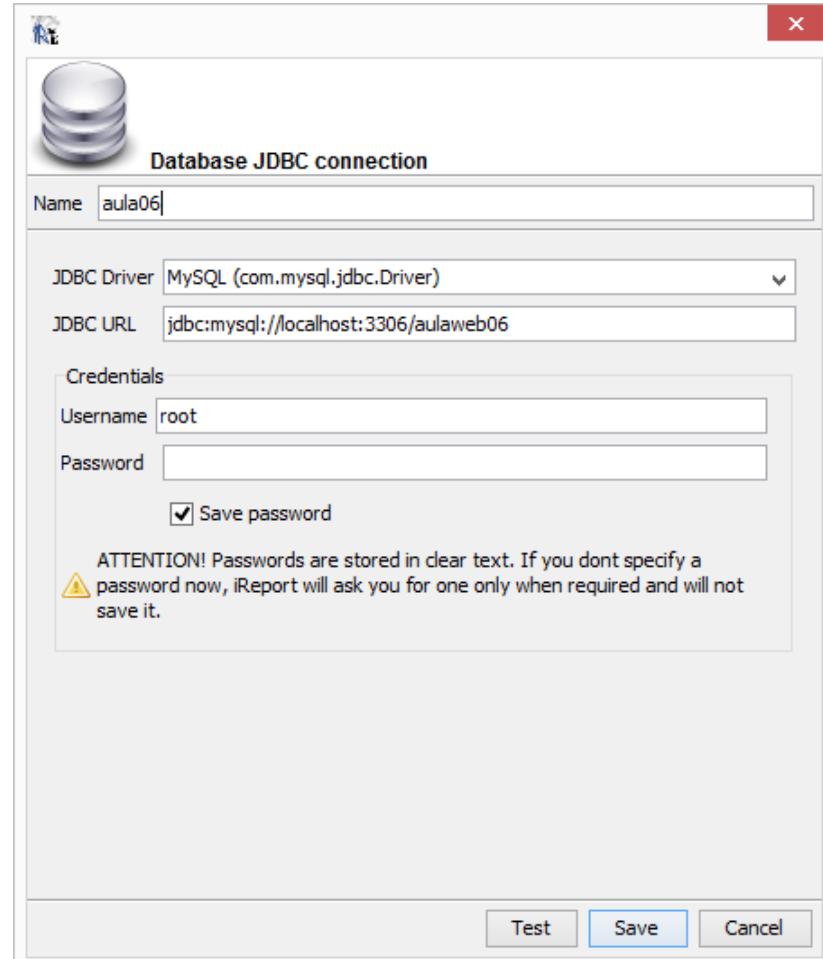
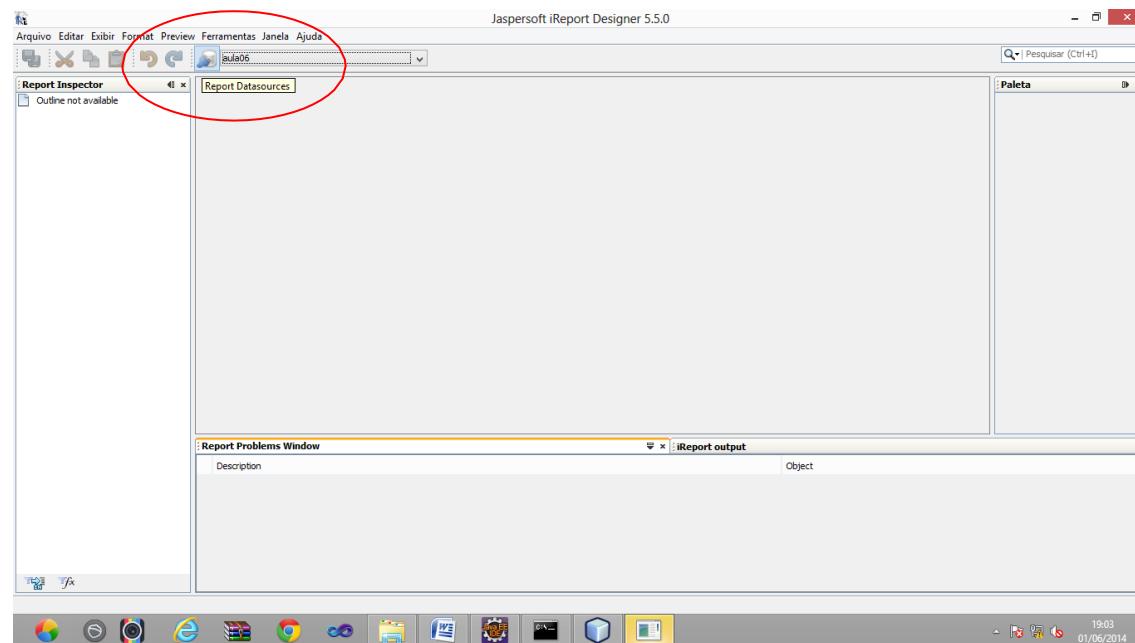
## Resultado...

Código	Produto	Preço	Quantidade	Total	Garantia	Informações	Fornecedor	Descrição
1	Mouse	30.0	10	300.0	90 dias	Presente, Cartão	Loja A	Artigos de Informática
2	Jogo Watch Dogs	199.0	2	398.0	30 dias	Presente, Cartão, Entrega	Loja D	Console e Games
3	Mouse	50.0	10	500.0	0		Loja B	Artigos Eletrônicos
4	Monitor	300.0	10	3000.0	0	Cartão, Entrega	Loja C	Artigos de Papelaria



## Utilizando o iReport

Criando uma fonte de dados





# Java WebDeveloper - BRQ

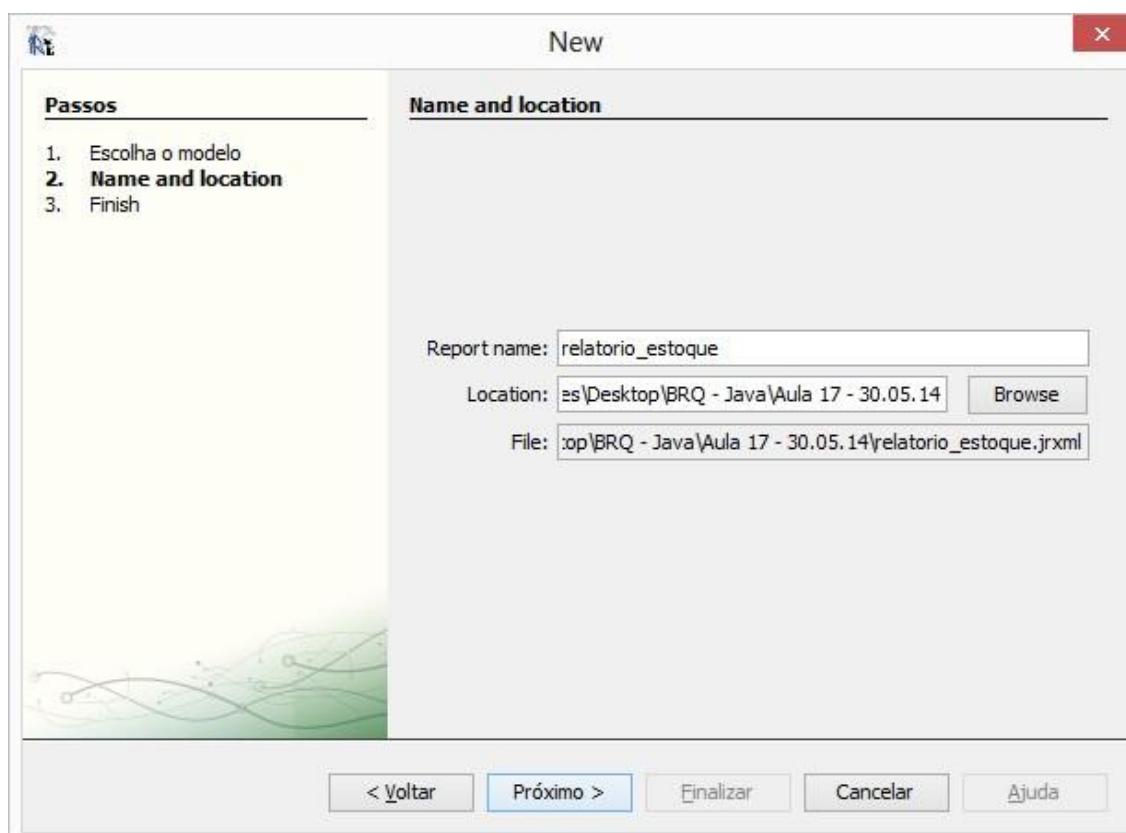
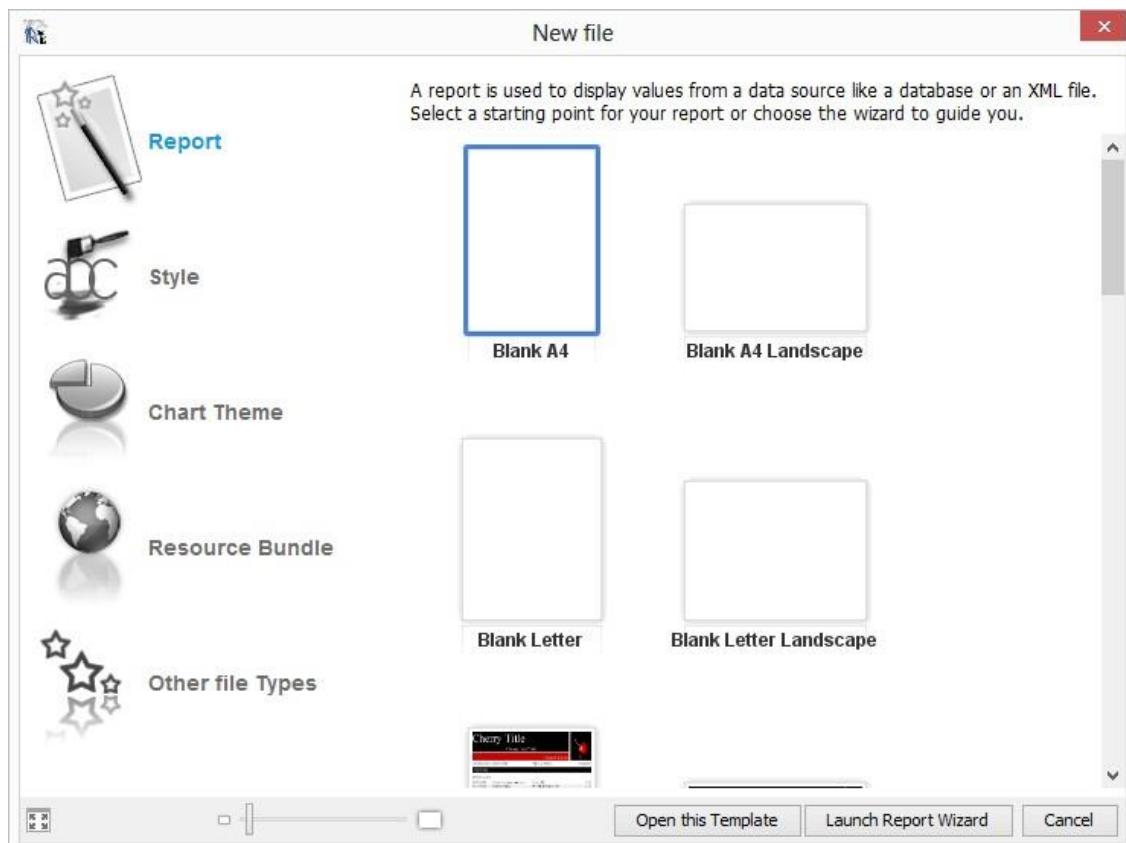
Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula

17

## Criando o Relatório...





# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**17**

## Criando a consulta para gerar os campos do relatório...

The screenshot shows the Jaspersoft iReport Designer interface. The title bar reads "Jaspersoft iReport Designer 5.5.0". The main window displays a report titled "Relatório de Produtos em Estoque". On the left, the "Report Inspector" panel lists report components like "relatorio\_estoque", "Styles", "Parameters", "Fields", etc. The toolbar at the top has several icons, with the second one from the left highlighted by a red circle. The report preview shows the title "Relatório de Produtos em Estoque" in a box.

**select \* from estoque**

The screenshot shows the "Report query" dialog box. The title bar says "Report query". The "Query language" dropdown is set to "SQL". The main area contains the SQL query "select \* from estoque". To the right, there's a note: "Drag a parameter into the query to a parameter. Hold CTL to add the parameter as query chunk." Below the query, there's a section for "Available parameters" and a "New parameter" button. At the bottom, there are buttons for "Filter expression...", "Sort options...", "Preview data ▾", "OK", and "Cancel".

Field name	Field type	Description
idproduto	java.lang.Integer	
produto	java.lang.String	
preco	java.lang.Double	
quantidade	java.lang.Integer	
garantia	java.lang.String	
infoadicionais	java.lang.String	
idfornecedor	java.lang.Integer	
fornecedor	java.lang.String	
descricao	java.lang.String	



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula  
**17**

Jaspersoft iReport Designer 5.5.0

relatorio\_estoque.jrxml

Relatório de Produtos em Estoque

Produto	preco	quantidade	garantia	infoadicionais	fornecedor
\$F{produto}	\$F{preco}	\$F{quantidade}	\$F{garantia}	\$F{infoadicionais}	\$F{fornecedor}

Page Header

Detail 1

Column Footer

Report Problems Window

**Preview do Relatorio com os dados obtidos do banco...**

Jaspersoft iReport Designer 5.5.0

relatorio\_estoque.jrxml

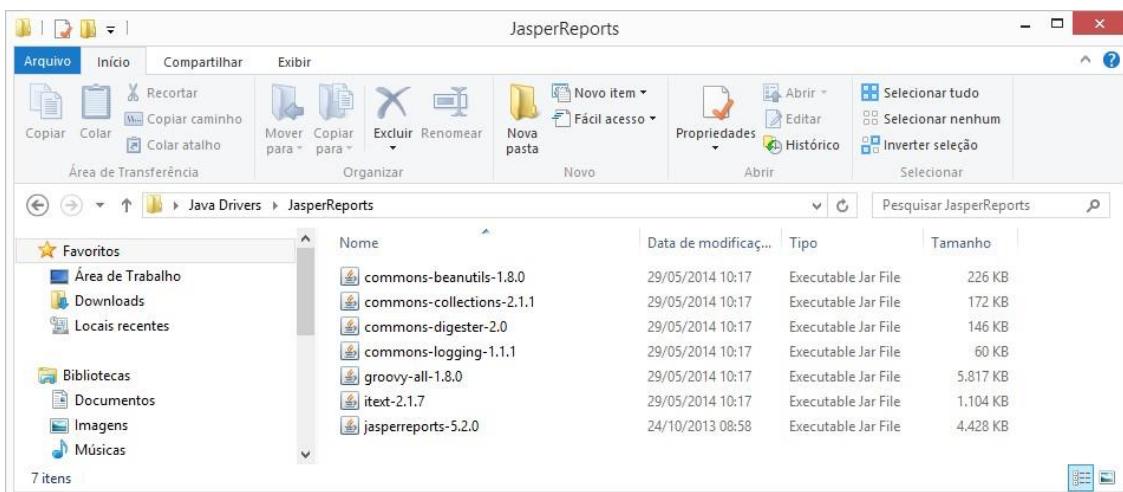
Relatório de Produtos em Estoque

Produto	preco	quantidade	garantia	infoadicionais	fornecedor
Mouse	30.0	10	90 dias	Presente, Cartão	Loja A
Jogo Watch Dogs	199.0	2	30 dias	Presente, Cartão,	Loja D
Mouse	50.0	10	0		Loja B
Monitor	300.0	10	0	Cartão, Entrega	Loja C

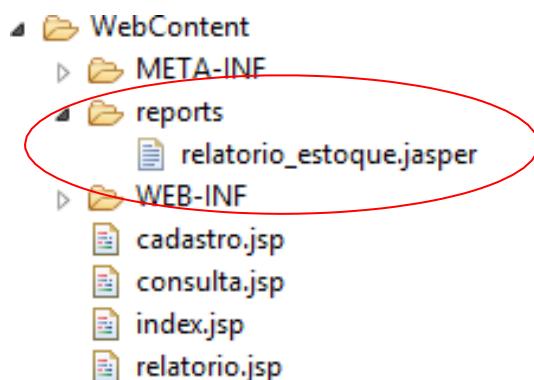


### Incluindo as bibliotecas do JasperReports no projeto...

WebContent\WEB-INF\lib



### Incluindo o arquivo .jasper no projeto



Criando a página para obter o relatório...

**relatorio.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
        <style>
            body {
                font-family: verdana;
                font-size: 9pt;
                padding: 40px;
            }
        </style>
    </head>
```



```
<body>

    <h3>Relatório de Produtos</h3>
    <a href="index.jsp">Voltar</a> para a página inicial.
    <hr/>

    <form name="formrelatorio" method="post"
          action="ControleProduto?cmd=relatorio"
          target="_blank">

        <input type="submit"
               value="Gerar Relatório de Vendas"/>

    </form>

</body>

</html>
```

Criando o método na Classe Dao para retornar a conexão com o BD para uso do relatório...

```
package persistence;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Dao {

    //Constantes
    private static final String DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATABASE = "jdbc:mysql://"
                                         localhost:3306/aulaweb06";
    private static final String USER = "root";
    private static final String PASSWORD = "";

    //Atributos para manipulação do banco
    protected Connection con;
    protected PreparedStatement stmt;
    protected ResultSet rs;

    //Método para abrir conexão com o banco de dados
    protected void open() throws Exception{
        Class.forName(DRIVER);
        con = DriverManager.getConnection
            (DATABASE, USER, PASSWORD);
    }

    //Método para fechar conexão com o banco de dados
    public void close() throws Exception{
        con.close();
    }
}
```



```
//Método estático para retornar uma conexão
//aberta com o banco de dados
//utilizado na geração do relatório
public static Connection getConnection() throws Exception{
    Class.forName(DRIVER);
    return DriverManager.getConnection
        (DATABASE, USER, PASSWORD);
}

}
```

### Implementando o Download do relatório no Servlet...

```
package control;

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

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

import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.JasperRunManager;
import persistence.Dao;
import persistence.ProdutoDao;
import entity.Fornecedor;
import entity.Produto;

@WebServlet("/ControleProduto")
public class ControleProduto extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleProduto() {
        super();
    }

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

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



```
String cmd = request.getParameter("cmd");

if(cmd.equalsIgnoreCase("cadastrar")){
    cadastrar(request, response);
}
else if(cmd.equalsIgnoreCase("relatorio")){
    relatorio(request, response);
}
}

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

try{

    Produto p = new Produto();

    p.setNome( request.getParameter("nome") );
    p.setPreco( new Double(request.
        getParameter("preco")) );
    p.setQuantidade( new Integer(request.
        getParameter("quantidade")) );
    p.setGarantia( request.getParameter("garantia") );

    //resgate de um campo de multipla seleção (checkbox)
    //é feito através de gerParameterValues
    //sempre retorna da página um vetor de String
    String itens[] = request.getParameter
    Values("itens"); //resgatando os checkboxes marcados
    String infoAdicionais = ""; //variavel String vazia

    for(int i = 0; itens != null && i < itens.length;
        i++){ //percorrendo o vetor
        infoAdicionais += itens[i];
        //concatenando o valor do campo marcado
        if(i != itens.length - 1){
            //não é a ultima posição
            infoAdicionais += ", ";
            //concatenando virgula ao final
        }
    }

    p.setInfoAdicionais(infoAdicionais);

    //Resgatar o id do fornecedor
    Fornecedor f = new Fornecedor();
    f.setIdFornecedor( new Integer(request.
        getParameter("fornecedor")) );
    p.setFornecedor(f);
    //relacionando o objeto Fornecedor 'f'
    //ao objeto Produto 'p'
```



```
//Realizando a gravação
ProdutoDao d = new ProdutoDao();
d.create(p); //salvando...

request.setAttribute("msg", "Dados do produto
                           gravados com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
                           + e.getMessage());
}
finally{
    //redirecionamento do fluxo...
    request.getRequestDispatcher("cadastro.jsp").
    forward(request, response);
}
}

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

try{

//pegar o caminho o arquivo do relatório
//path -> caminho onde está o arquivo
//do relatório (WebContent)
InputStream path = getServletContext().
    getResourceAsStream("/reports/
        relatorio_estoque.jasper");

//converter o relatório para PDF, preparando-o
//para download
byte[] arquivo = JasperRunManager
    .runReportToPdf(path, null, Dao.getConnection());

ServletOutputStream download = response.
    getOutputStream(); //objeto de download
download.write(arquivo);
//escrever o conteúdo do download
download.flush();
//encerra o processo para o navegador
//iniciar o download
}
catch(Exception e){
    PrintWriter out = response.getWriter();
    out.print("Erro ao gerar relatório
        de produtos -> " + e.getMessage());
}
}
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 30 de Maio de 2014

Relacionamentos 1 para Muitos. UseBeans, JSTL,  
Geração de Relatórios utilizando iReport.

Aula

17

Executando...

The screenshot shows a web browser window with the URL `localhost:8082/javaWebAula06/relatorio.jsp`. The page title is "Relatório de Produtos". Below the title, there is a link "[Voltar](#) para a página inicial." and a button labeled "Gerar Relatório de Vendas".

The screenshot shows a web browser window with the URL `localhost:8082/javaWebAula06/ControleProduto?cmd=relatorio`. The page title is "Relatório de Produtos em Estoque". Below the title, there is a table displaying product information:

Produto	preco	quantidade	garantia	infoadicionais	fornecedor
Mouse	30.0	10	90 dias	Presente, Cartão	Loja A
Jogo Watch Dogs	199.0	2	30 dias	Presente, Cartão,	Loja D
Mouse	50.0	10	0		Loja B
Monitor	300.0	10	0	Cartão, Entrega	Loja C



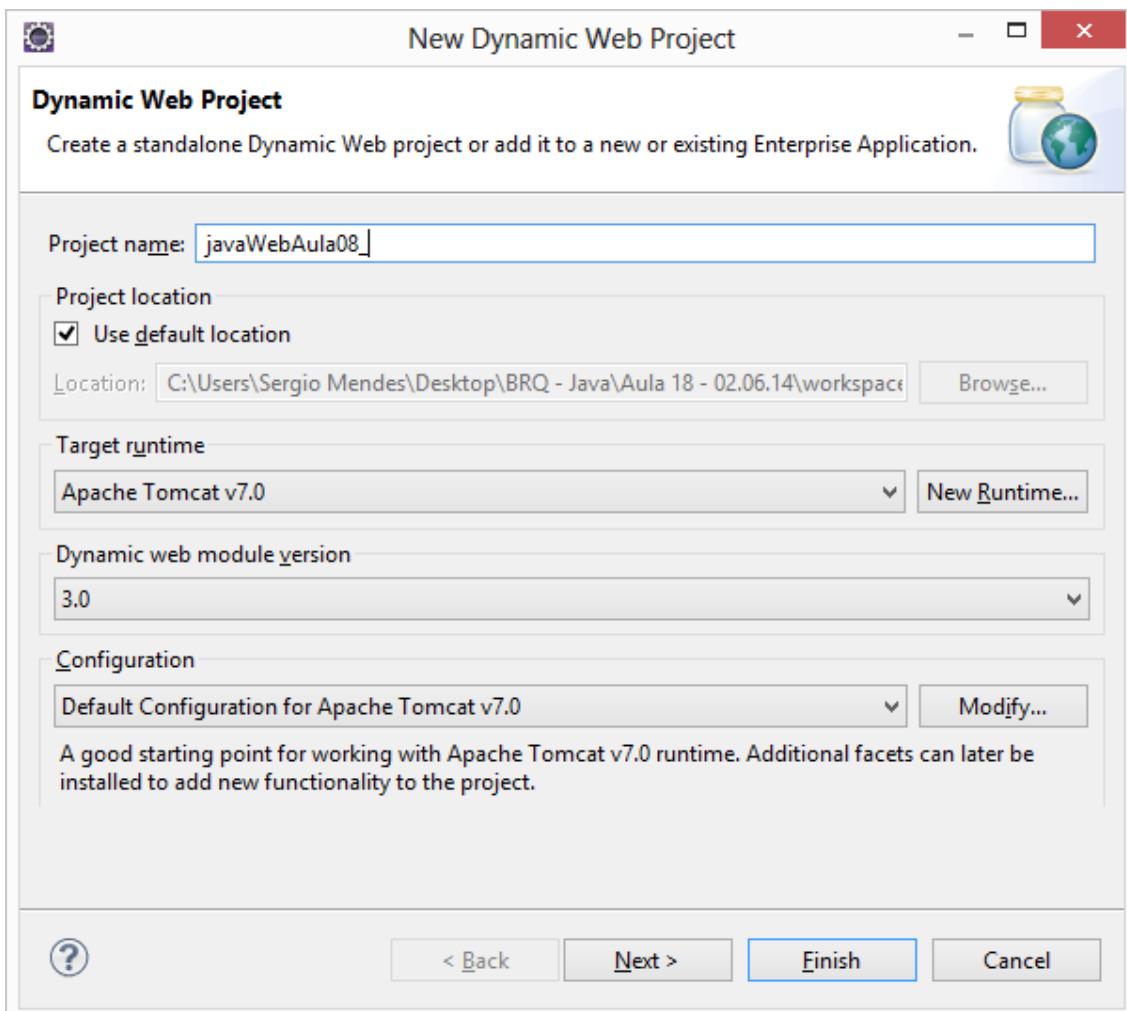
# Java WebDeveloper - BRQ

Segunda-feira, 02 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**18**

Criando o Projeto...



Criando a página inicial do projeto...

**\WebContent\index.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
        </head>

    <body>

        <h3>Projeto controle de Funcionários</h3>
        <hr/>

        <ul>
            <li> <a href="cadastro.jsp">
                Cadastrar Funcionarios</a> </li>
```



```
<li> <a href="consulta.jsp">  
          Consultar Funcionarios</a> </li>  
</ul>  
</body>  
</html>
```

Criando a página para cadastro de funcionários  
**\WebContent\cadastro.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
  
<html>  
  <head>  
  </head>  
  
  <body>  
    <h3>Cadastro de Funcionários</h3>  
    <a href="index.jsp">Voltar</a> para a página inicial.  
    <hr/>  
  
    <form name="formcadastro" method="post" action="">  
  
      Nome do Funcionário: <br/>  
      <input type="text" name="nome"  
             required="required"/>  
      <br/><br/>  
  
      Salário: <br/>  
      <input type="text" name="salario"  
             required="required"/>  
      <br/><br/>  
  
      Data de Admissão: <br/>  
      <input type="date" name="dataadmissao"  
             required="required"/>  
      <br/><br/>  
  
      Tipo: <br/>  
      <select name="tipo" required="required">  
        <option value="">- Selecione uma Opção -</option>  
        <option value="Con">Funcionario Contratado</option>  
        <option value="Ter">Funcionario Teceirizado</option>  
        <option value="Est">Funcionario Estagiario</option>  
      </select>  
      <br/><br/>  
  
      <input type="submit" value="Cadastrar Funcionário"/>  
    </form>  
  </body>  
</html>
```



### Criando a página de consulta de funcionários \WebContent\consulta.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
        </head>

    <body>

        <h3>Consulta de Funcionários</h3>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

    </body>
</html>
```

### Executando...

The screenshot shows a web browser window with the URL `localhost:8082/javaWebAula08/`. The title bar says "Projeto controle de Funcionários". The page content contains a bulleted list:

- [Cadastrar Funcionarios](#)
- [Consultar Funcionarios](#)

The screenshot shows a web browser window with the URL `localhost:8082/javaWebAula08/cadastro.jsp`. The title bar says "Cadastro de Funcionários". The page content includes the following fields:  
- Nome do Funcionário:   
- Salário:   
- Data de Admissão:  (formato dd/mm/aaaa)  
- Tipo:  (dropdown menu with placeholder "- Selecione uma Opção -")  
- Cadastrar Funcionário:



Criando a base de dados...

**sql\script.sql**

```
drop database if exists aulaweb08;
create database aulaweb08;
use aulaweb08;

create table funcionario(
    idfuncionario      integer          auto_increment,
    nome               varchar(50)       not null,
    salario            double           not null,
    dataadmissao       date             not null,
    tipo               varchar(3)        not null,
    primary key(idfuncionario);

alter table funcionario
add check( tipo in('Con', 'Ter', 'Est') );

alter table funcionario
add check( salario >= 650 );

show tables;
desc funcionario;
```

Executando no MySql...

The screenshot shows a Windows Command Prompt window titled 'cmd' with the path 'C:\WINDOWS\system32\cmd.exe - mysql -u root -p'. The window displays the following MySQL session:

```
+-----+
| Tables_in_aulaweb08 |
+-----+
| funcionario         |
+-----+
1 row in set (0.00 sec)

mysql> desc funcionario;
+-----+-----+-----+-----+-----+-----+
| Field   | Type    | Null | Key | Default | Extra   |
+-----+-----+-----+-----+-----+-----+
| idfuncionario | int(11) | NO  | PRI | NULL    | auto_increment |
| nome          | varchar(50)| NO |     | NULL    |           |
| salario        | double   | NO  |     | NULL    |           |
| dataadmissao  | date    | NO  |     | NULL    |           |
| tipo          | varchar(3) | NO  |     | NULL    |           |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```



## JPA - Java Persistence API

Biblioteca Java utilizada para Mapeamento Objeto / Relacional (ORM)

ORM é uma técnica utilizada para descrever para frameworks ou tecnologias Classes JavaBean de forma que estas sejam interpretadas como tabelas ou entidades de uma base de dados.

Cada JavaBean mapeado com a JPA poderá ser utilizado com algum framework que saiba interpretar este mapeamento e automatizar as rotinas no BD (crud) para o JavaBean Mapeado.

Este mapeamento é feito através de **@Annotations**

- Mapeando a Classe Funcionário...

```
package entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity //mapeando a Classe como uma entidade de um BD
@Table(name="funcionario") //nome da tabela do banco de dados
public class Funcionario {

    @Id //chave primária
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="idfuncionario") //coluna da tabela
    private Integer idFuncionario;

    @Column(name="nome", length=50, nullable=false)
    private String nome;

    @Column(name="salario", nullable=false)
    private Double salario;

    @Temporal(TemporalType.DATE) //campo somente data
    @Column(name="dataadmissao", nullable=false)
    private Date dataAdmissao;

    @Column(name="tipo", length=3, nullable=false)
    private String tipo;
```



```
public Funcionario() {
    // Construtor default
}

public Funcionario(Integer idFuncionario,
                    String nome, Double salario,
                    Date dataAdmissao, String tipo) {
    super();
    this.idFuncionario = idFuncionario;
    this.nome = nome;
    this.salario = salario;
    this.dataAdmissao = dataAdmissao;
    this.tipo = tipo;
}

@Override
public String toString() {
    return "Funcionario [idFuncionario="
        + idFuncionario + ", nome=" + nome
        + ", salario=" + salario + ", dataAdmissao="
        + dataAdmissao + ", tipo=" + tipo + "]";
}

public Integer getIdFuncionario() {
    return idFuncionario;
}

public void setIdFuncionario(Integer idFuncionario) {
    this.idFuncionario = idFuncionario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getSalario() {
    return salario;
}

public void setSalario(Double salario) {
    this.salario = salario;
}

public Date getDataAdmissao() {
    return dataAdmissao;
}

public void setDataAdmissao(Date dataAdmissao) {
    this.dataAdmissao = dataAdmissao;
}
```

```

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}
}

```

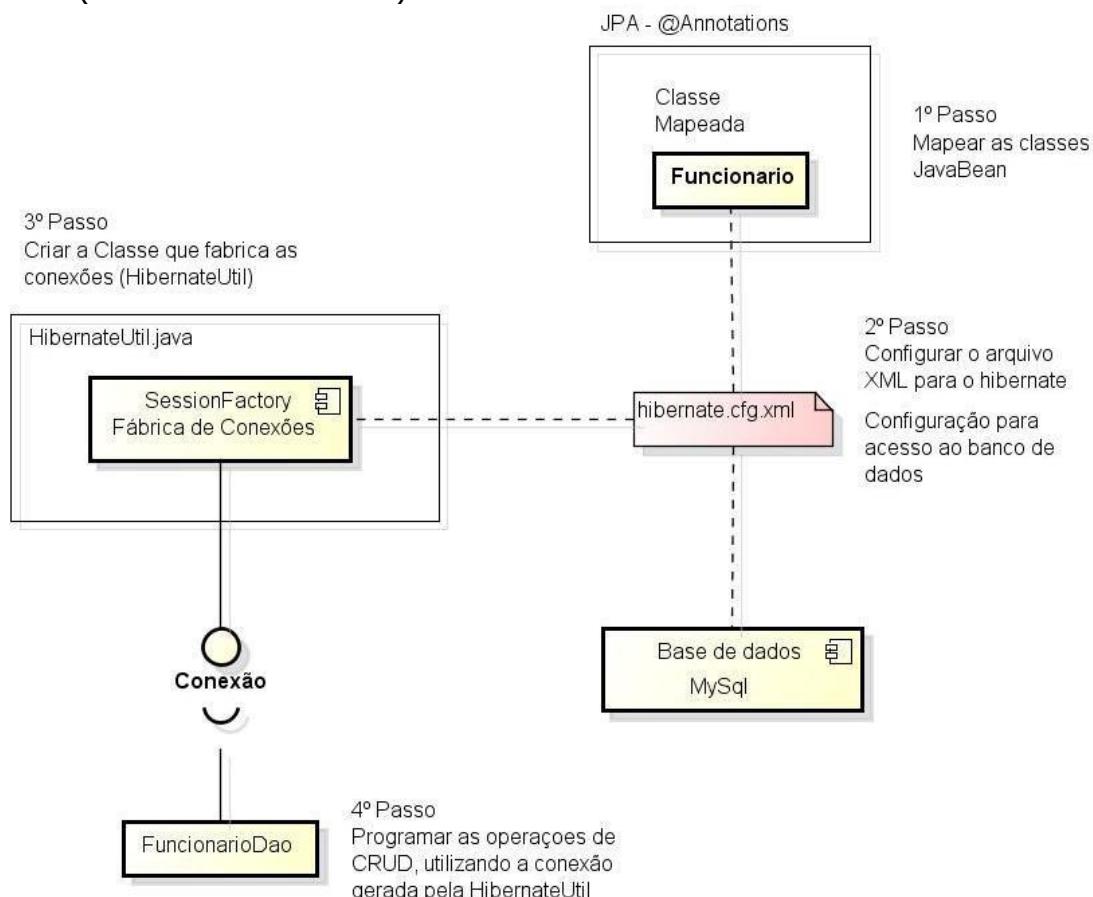
## HIBERNATE

Framework Java para acesso a banco de dados baseado em Mapeamento Objeto Relacional.

Consegue interpretar as @Annotations da JPA e estabelecer conexão com a base de dados além de prover as operações de CRUD, etc...

Ao contrário do JDBC, não iremos crar uma Classe Dao para conectar na base de dados. O Hibernate utiliza arquivos .xml para configurar o caminho, usuario, senha do banco, etc...  
\config\mysql\_hibernate.cfg.xml

Para que Hibernate possa conectar-se a uma base de dados, é necessário programar um componente chamado se SessionFactory (Fábrica de Conexões).





Criando o arquivo **mysql\_hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>

    <session-factory>

        <!-- Configurando o acesso ao banco de dados -->
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect
        </property>
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/aulaweb08
        </property>
        <property name="hibernate.connection.username">
            root
        </property>
        <property name="hibernate.connection.password"></property>

        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.format_sql">true</property>

        <!-- Configurando as Classes mapeadas pela JPA -->
        <mapping class="entity.Funcionario" />

    </session-factory>

</hibernate-configuration>
```

**Criando a Classe HibernateUtil para gerar a fábrica de conexões com a base de dados...**

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    //bloco estatico que configura a sessionfactory
    //baseado no conteudo do arquivo cfg.xml
    static {

        try {
```



```
sessionFactory = new AnnotationConfiguration().
    configure( "config/mysql_hibernate.cfg.xml" ).
    buildSessionFactory();

} catch (Throwable ex) {

    System.err.println("Initial SessionFactory
creation failed." + ex);

    throw new ExceptionInInitializerError(ex);
}

//Método estatico que retorna a fabrica
//de conexões configurada
public static SessionFactory getSessionFactory() {
    return sessionFactory;
}
```

### Criando a Classe de persistência para Funcionario...

```
package persistence;

import hibernate.HibernateUtil;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Funcionario;

public class FuncionarioDao {

    //Atributos
    //JDBC -> Connection, PreparedStatement, ResultSet
    private Session session;
    //obter uma conexão com o BD
    private Transaction transaction;
    //transações na base de dados
    private Query query;
    //executar consultas (HQL -> Hibernate Query Language)

    //Método para gravar um registro na tabela de Funcionario
    public void create(Funcionario f) throws Exception{
        session = HibernateUtil.getSessionFactory().
            openSession();
        transaction = session.beginTransaction();
        session.save(f);
        transaction.commit(); //executar
        session.close();
    }
}
```



```
public void delete(Funcionario f) throws Exception{
    session = HibernateUtil.getSessionFactory() .
        openSession();
    transaction = session.beginTransaction();
    session.delete(f);
    transaction.commit(); //executar
    session.close();
}

public void update(Funcionario f) throws Exception{
    session = HibernateUtil.getSessionFactory() .
        openSession();
    transaction = session.beginTransaction();
    session.update(f);
    transaction.commit(); //executar
    session.close();
}

public Funcionario findById(Integer idFuncionario)
throws Exception{
    session = HibernateUtil.getSessionFactory() .
        openSession();
    //O Hibernate já possui um método que retorna 1
    //registro pela chave primária
    //get -> método do hibernate que retorna
    //1 registro pela chave primária
    Funcionario f = (Funcionario) session.
        get(Funcionario.class, idFuncionario);

    session.close(); //fecho a conexão
    return f; //retrorar o objeto funcionario
}

public List<Funcionario> findAll() throws Exception{
    session = HibernateUtil.getSessionFactory() .
    openSession();

    //HQL -> Hibernate Query Language
    query = session.createQuery("from Funcionario");
    //select * from funcionario

    List<Funcionario> lista = query.list();
    //transforma os registros da query em lista

    session.close(); //fechar a conexão
    return lista; //retornar a lista
}

}
```



### Implementando o cadastro dos Funcionários...

***action="ControleFuncionario?cmd=cadastrar"***

#### Criando o Servlet...

```
package control;

import java.io.IOException;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

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 persistence.FuncionarioDao;
import entity.Funcionario;

@WebServlet("/ControleFuncionario")
public class ControleFuncionario extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleFuncionario() {
        super();
    }

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

    protected void doPost(HttpServletRequest request,
                         HttpServletResponse response) throws
        ServletException, IOException {
        String cmd = request.getParameter("cmd");

        if(cmd.equals("cadastrar")){
            cadastrar(request, response);
        }
    }

    protected void cadastrar(HttpServletRequest request,
                           HttpServletResponse response) throws
        ServletException, IOException {
        try{
            //Resgatando o conteúdo do campo 'dataadmissao' e
            //convertendo-o para Date
            String data = request.getParameter("dataadmissao");
            //Exemplo: 2014/06/02
        }
    }
}
```



```
int ano = new Integer(data.substring(0, 4));

int mes = new Integer(data.substring(5, 7));

int dia = new Integer(data.substring(8, 10));

Calendar cal = new GregorianCalendar
(ano, mes - 1, dia);

//resgatando os dados do funcionario
Funcionario f = new Funcionario();
//instanciando o JavaBean

f.setNome(request.getParameter("nome"));
f.setSalario(new Double(request.
getParameter("salario")));
f.setTipo(request.getParameter("tipo"));
f.setDataAdmissao(cal.getTime());
//data do objeto Calendar

FuncionarioDao d = new FuncionarioDao();
d.create(f); //gravação

request.setAttribute("msg", "Funcionário
cadastrado com sucesso");
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
+ e.getMessage());
}
finally{
    request.getRequestDispatcher("cadastro.jsp").
    forward(request, response);
}

}
```

### Exemplo de Uso do Substring para captura do conteúdo da data...

```
public static void main(String[] args) {

String data = "02/06/2014";

int dia = new Integer(data.substring(0, 2));
int mes = new Integer(data.substring(3, 5));
int ano = new Integer(data.substring(6, 10));

Calendar c = new GregorianCalendar(ano, mes - 1, dia);
Date d = c.getTime();

System.out.println(d);

}
```



# Java WebDeveloper - BRQ

Segunda-feira, 02 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**18**

## Executando...

Cadastro de Funcionários

[Voltar](#) para a página inicial.

Nome do Funcionário:

Salário:

Data de Admissão:

Tipo:

Cadastro de Funcionários

[Voltar](#) para a página inicial.

Nome do Funcionário:

Salário:

Data de Admissão:

Tipo:

Funcionário cadastrado com sucesso

No banco de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
mysql> select * from funcionario;
+-----+-----+-----+-----+-----+
| idfuncionario | nome | salario | dataadmissao | tipo |
+-----+-----+-----+-----+-----+
| 1 | Marcone | 2500 | 2014-06-02 | Est |
| 2 | Genisson | 2000 | 2014-06-04 | Est |
| 3 | Pedro | 2300 | 2014-06-03 | Est |
| 4 | Ana Paula | 2000 | 1990-10-10 | Est |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```



### Criando o Bean para listar os Funcionarios...

```
package beans;

import java.util.List;

import persistence.FuncionarioDao;
import entity.Funcionario;

//Classe utilizada na pagina através da tag <jsp:useBean/>
public class FuncionariosBean {

    // Atributo para carregar os dados do banco
    private List<Funcionario> listagemFuncionarios;

    public List<Funcionario> getListagemFuncionarios() {

        try{
            FuncionarioDao d = new FuncionarioDao();
            listagemFuncionarios = d.findAll();
        }
        catch(Exception e){
            e.printStackTrace(); //console do tomcat
        }

        return listagemFuncionarios;
    }

    public void setListagemFuncionarios
        (List<Funcionario> listagemFuncionarios) {
        this.listagemFuncionarios = listagemFuncionarios;
    }

}
```

### Utilizando JSTL para exibir os Funcionários na página de consulta...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<!-- importando a biblioteca de tags (JSTL TagLibrary) -->
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="f" %>

<!-- referenciar a classe FuncionariosBean -->
<jsp:useBean class="beans.FuncionariosBean"
    id="fb" scope="page"/>

<html>
```



# Java WebDeveloper - BRQ

Segunda-feira, 02 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**18**

```
<head>

</head>

<body>

    <h3>Consulta de Funcionários</h3>
    <a href="index.jsp">Voltar</a> para a página inicial.
    <hr/>

    <table border="1" style="width: 100%;">

        <thead>
            <tr>
                <th> Código </th>
                <th> Nome do Funcionário </th>
                <th> Salário </th>
                <th> Data de Admissão </th>
                <th> Tipo </th>
            </tr>
        </thead>

        <tbody>

            <c:forEach items="${fb.listagemFuncionarios}"
                       var="f">

                <tr>
                    <td> ${f.idFuncionario} </td>
                    <td> ${f.nome} </td>
                    <td> <f:formatNumber value="${f.salario}"
                                         type="currency"/>
                    </td>
                    <td> <f:formatDate
                         value="${f.dataAdmissao}"
                         pattern="EE dd/MM/yyyy"/>
                    </td>
                    <td> ${f.tipo} </td>
                </tr>
            
```

Executando...



# Java WebDeveloper - BRQ

Segunda-feira, 02 de Junho de 2014

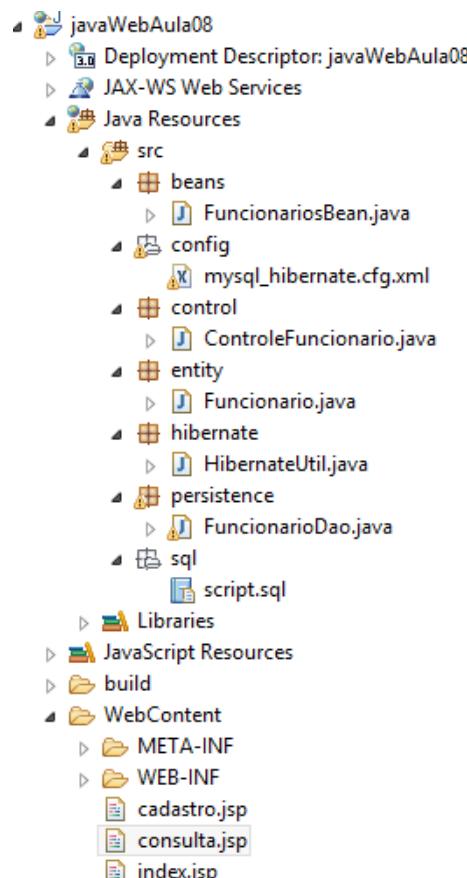
Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**18**

The screenshot shows a web browser window with the URL `localhost:8082/javaWebAula08/consulta.jsp`. The page title is "Consulta de Funcionários". Below the title, there is a link "[Voltar](#) para a página inicial.". A table displays four rows of employee information:

Código	Nome do Funcionário	Salário	Data de Admissão	Tipo
1	Marcone	R\$ 2.500,00	Seg 02/06/2014	Est
2	Genisson	R\$ 2.000,00	Qua 04/06/2014	Est
3	Pedro	R\$ 2.300,00	Ter 03/06/2014	Est
4	Ana Paula	R\$ 2.000,00	Qua 10/10/1990	Est

## Estrutura do projeto...





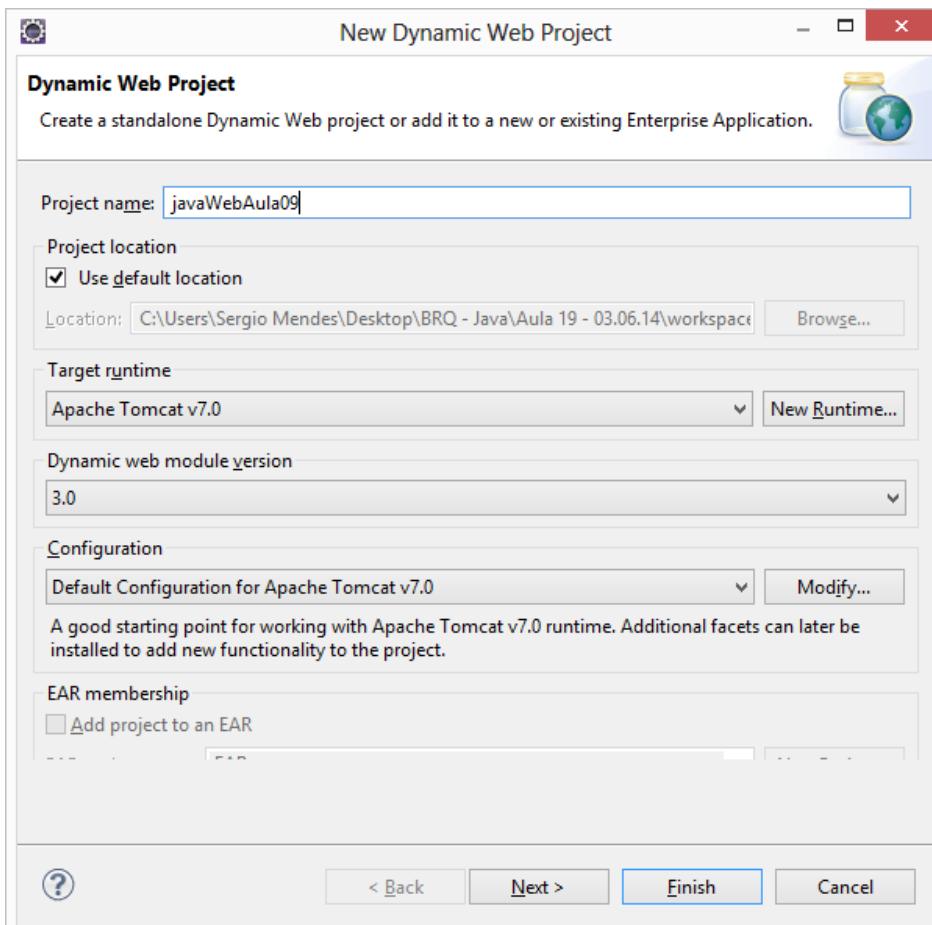
# Java WebDeveloper - BRQ

Terça-feira, 03 de Junho de 2014

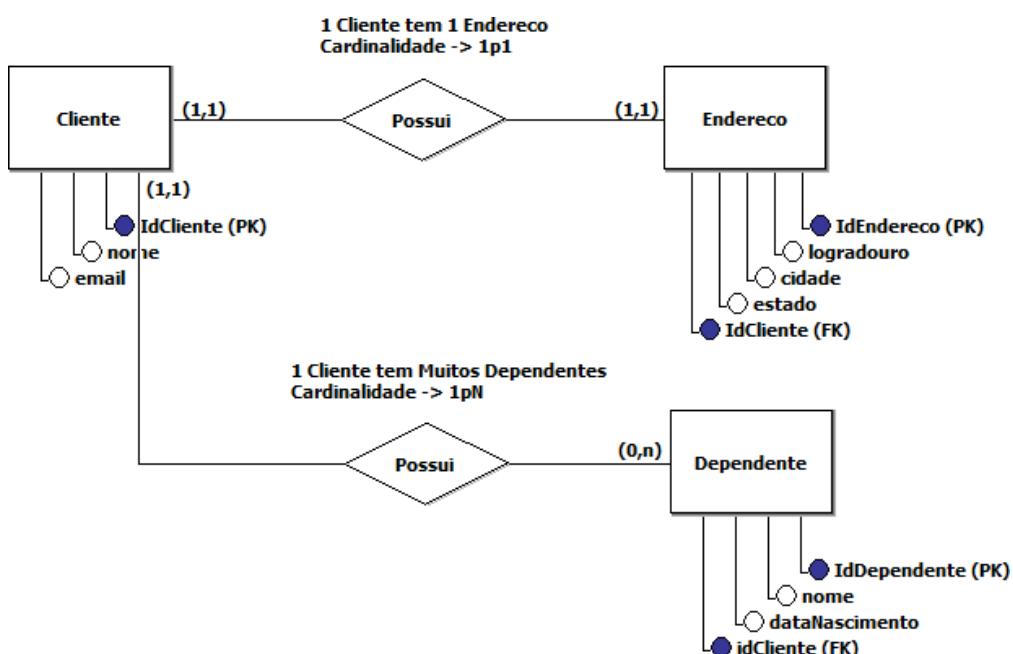
Persistência de dados com Hibernate. Mapeamento Objeto Relacional utilizando JPA - Java Persistence API.

Aula  
19

Criando o Projeto Web...



## DER - Diagrama de Entidade Relacionamento





### Passo 1: Criar as classes JavaBean

```
package entity;

import java.util.Collection;

public class Cliente {

    private Integer idCliente;
    private String nome;
    private String email;

    private Endereco residencia; // TER-1
    private Collection<Dependente> dependentes; // TER-Muitos

    public Cliente() {
    }

    public Cliente(Integer idCliente, String nome, String email) {
        super();
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
    }

    @Override
    public String toString() {
        return "Cliente [idCliente=" + idCliente + ", nome="
               + nome + ", email=" + email + "]";
    }

    public Integer getIdCliente() {
        return idCliente;
    }

    public void setIdCliente(Integer idCliente) {
        this.idCliente = idCliente;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }
}
```



# Java WebDeveloper - BRQ

Terça-feira, 03 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**19**

```
public void setEmail(String email) {
    this.email = email;
}

public Endereco getResidencia() {
    return residencia;
}

public void setResidencia(Endereco residencia) {
    this.residencia = residencia;
}

public Collection<Dependente> getDependentes() {
    return dependentes;
}

public void setDependentes(Collection<Dependente> dependentes) {
    this.dependentes = dependentes;
}
}

-----
package entity;

public class Endereco {

    private Integer idEndereco;
    private String logradouro;
    private String cidade;
    private String estado;

    private Cliente morador; // TER - 1

    public Endereco() {
    }

    public Endereco(Integer idEndereco, String logradouro,
                    String cidade, String estado) {
        super();
        this.idEndereco = idEndereco;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "Endereco [idEndereco=" + idEndereco
            + ", logradouro=" + logradouro + ", cidade=" + cidade
            + ", estado=" + estado + "]";
    }
}
```



```
public Integer getIdEndereco() {
    return idEndereco;
}

public void setIdEndereco(Integer idEndereco) {
    this.idEndereco = idEndereco;
}

public String getLogradouro() {
    return logradouro;
}

public void setLogradouro(String logradouro) {
    this.logradouro = logradouro;
}

public String getCidade() {
    return cidade;
}

public void setCidade(String cidade) {
    this.cidade = cidade;
}

public String getEstado() {
    return estado;
}

public void setEstado(String estado) {
    this.estado = estado;
}

public Cliente getMorador() {
    return morador;
}

public void setMorador(Cliente morador) {
    this.morador = morador;
}
}
```

---

```
package entity;

import java.util.Date;

public class Dependente {

    private Integer idDependente;
    private String nome;
```



```
private Date dataNascimento;

private Cliente cliente; // TER - 1

public Dependente() {

}

public Dependente(Integer idDependente, String nome,
                  Date dataNascimento) {
    super();
    this.idDependente = idDependente;
    this.nome = nome;
    this.dataNascimento = dataNascimento;
}

@Override
public String toString() {
    return "Dependente [idDependente=" + idDependente
           + ", nome=" + nome + ", dataNascimento="
           + dataNascimento + "]";
}

public Integer getIdDependente() {
    return idDependente;
}

public void setIdDependente(Integer idDependente) {
    this.idDependente = idDependente;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Date getDataNascimento() {
    return dataNascimento;
}

public void setDataNascimento(Date dataNascimento) {
    this.dataNascimento = dataNascimento;
}

public Cliente getCliente() {
    return cliente;
}
```



```
public void setCliente(Cliente cliente) {  
    this.cliente = cliente;  
}  
-----
```

## JPA - Java Persistence API

Biblioteca de Anotações do java para mapeamento objeto relacional

### @Annotations

- 4 tipos de cardinalidades que podemos mapear em JPA:

@OneToOne	1 para 1
@OneToMany	1 para Muitos
@ManyToOne	Muitos para 1
@ManyToMany	Muitos para Muitos

```
package entity;  
  
import java.util.Collection;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.OneToMany;  
import javax.persistence.OneToOne;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "cliente")  
public class Cliente {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "idcliente")  
    private Integer idCliente;  
  
    @Column(name = "nome", length = 50, nullable = false)  
    private String nome;  
  
    @Column(name = "email", length = 50, nullable = false,  
            unique = true)  
    private String email;
```



```
@OneToOne(mappedBy = "morador") //1 para 1
private Endereco residencia; // TER-1

@OneToMany(mappedBy = "cliente") //1 para Muitos
private Collection<Dependente> dependentes; // TER-Muitos

public Cliente() {
}

public Cliente(Integer idCliente, String nome, String email) {
    super();
    this.idCliente = idCliente;
    this.nome = nome;
    this.email = email;
}

@Override
public String toString() {
    return "Cliente [idCliente=" + idCliente + ", nome="
           + nome + ", email=" + email + "]";
}

public Integer getIdCliente() {
    return idCliente;
}

public void setIdCliente(Integer idCliente) {
    this.idCliente = idCliente;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Endereco getResidencia() {
    return residencia;
}
```



```
public void setResidencia(Endereco residencia) {  
    this.residencia = residencia;  
}  
  
public Collection<Dependente> getDependentes() {  
    return dependentes;  
}  
  
public void setDependentes(Collection<Dependente> dependentes) {  
    this.dependentes = dependentes;  
}  
}
```

-----

```
package entity;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.OneToOne;  
import javax.persistence.Table;  
  
@Entity  
@Table(name = "endereco")  
public class Endereco {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "idendereco")  
    private Integer idEndereco;  
  
    @Column(name = "logradouro", length = 200, nullable = false)  
    private String logradouro;  
  
    @Column(name = "cidade", length = 100, nullable = false)  
    private String cidade;  
  
    @Column(name = "estado", length = 25, nullable = false)  
    private String estado;  
  
    @OneToOne // 1 para 1  
    @JoinColumn(name="idclientefk", unique = true)  
    //foreign key -> chave estrangeira  
    private Cliente morador; // TER - 1
```



```
public Endereco() {  
}  
  
public Endereco(Integer idEndereco, String logradouro,  
                 String cidade, String estado) {  
    super();  
    this.idEndereco = idEndereco;  
    this.logradouro = logradouro;  
    this.cidade = cidade;  
    this.estado = estado;  
}  
  
@Override  
public String toString() {  
    return "Endereco [idEndereco=" + idEndereco + ",  
            logradouro=" + logradouro + ", cidade=" + cidade  
            + ", estado=" + estado + "]";  
}  
  
public Integer getIdEndereco() {  
    return idEndereco;  
}  
  
public void setIdEndereco(Integer idEndereco) {  
    this.idEndereco = idEndereco;  
}  
  
public String getLogradouro() {  
    return logradouro;  
}  
  
public void setLogradouro(String logradouro) {  
    this.logradouro = logradouro;  
}  
  
public String getCidade() {  
    return cidade;  
}  
  
public void setCidade(String cidade) {  
    this.cidade = cidade;  
}  
  
public String getEstado() {  
    return estado;  
}  
  
public void setEstado(String estado) {  
    this.estado = estado;  
}
```



```
public Cliente getMorador() {
    return morador;
}

public void setMorador(Cliente morador) {
    this.morador = morador;
}
}
```

---

```
-----
package entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
```

```
@Entity
@Table(name = "dependente")
public class Dependente {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "iddependente")
    private Integer idDependente;

    @Column(name = "nome", length = 50, nullable = false)
    private String nome;

    @Temporal(TemporalType.DATE) // somente data
    @Column(name = "datanasc", nullable = false)
    private Date dataNascimento;

    @ManyToOne //Muitos para 1
    @JoinColumn(name = "idclientefk")
    private Cliente cliente; // TER - 1

    public Dependente() {

    }
}
```



```
public Dependente(Integer idDependente, String nome,
                   Date dataNascimento) {
    super();
    this.idDependente = idDependente;
    this.nome = nome;
    this.dataNascimento = dataNascimento;
}

@Override
public String toString() {
    return "Dependente [idDependente=" + idDependente
           + ", nome=" + nome + ", dataNascimento="
           + dataNascimento + "]";
}

public Integer getIdDependente() {
    return idDependente;
}

public void setIdDependente(Integer idDependente) {
    this.idDependente = idDependente;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Date getDataNascimento() {
    return dataNascimento;
}

public void setDataNascimento(Date dataNascimento) {
    this.dataNascimento = dataNascimento;
}

public Cliente getCliente() {
    return cliente;
}

public void setCliente(Cliente cliente) {
    this.cliente = cliente;
}

}
```



## Hibernate

Framework para acesso a base de dados utilizando como padrão o Mapeamento Objeto Relacional (feito pela JPA)

### 1º Passo: criar o arquivo **hibernate.cfg.xml**

Contém as configurações para acesso ao banco de dados e às classes mapeadas pela JPA.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>

<!-- Configuração para que o hibernate accesse a base de dados -->
<property name="hibernate.dialect">
    org.hibernate.dialect.MySQLDialect
</property>

<property name="hibernate.connection.driver_class">
    com.mysql.jdbc.Driver
</property>

<property name="hibernate.connection.url">
    jdbc:mysql://localhost:3306/aulaweb09
</property>

<property name="hibernate.connection.username">
    root
</property>

<property name="hibernate.connection.password"></property>

<!-- toda operação feita pelo no BD ser exibida em SQL no tomcat -->
<property name="hibernate.show_sql">true</property>

<!-- para que o sql exibido apareça formatado e alinhado no tomcat -->
<property name="hibernate.format_sql">true</property>

<!-- Classes mapeadas pela JPA -->
<mapping class="entity.Cliente" />
<mapping class="entity.Endereco" />
<mapping class="entity.Dependente" />

</session-factory>
</hibernate-configuration>
```



Programa para ser executado em um método Main que, através do **hibernate.cfg.xml** exporte e gere as tabelas no banco de dados baseado no mapeamento

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.cfg.Configuration;
import org.hibernate.tool.hbm2ddl.SchemaExport;

public class ExportSql {

    public static void main(String[] args) {

        try{

            //Ler o arquivo hibernate.cfg.xml
            Configuration cfg = new AnnotationConfiguration();
            //apontar o caminho do arquivo cfg.xml
            cfg.configure("config/mysql_hibernate.cfg.xml");

            //Gerar as tabelas no banco
            new SchemaExport(cfg).create(true, true);

            System.out.println("\nTabelas criadas
                com sucesso!");
        }
        catch(Exception e){
            System.out.println("Erro ao gerar tabelas -> "
                + e.getMessage());
        }
    }
}
```

### Script gerado...

```
alter table dependente
    drop
    foreign key FKFE6ED5463AB2FC53

alter table endereco
    drop
    foreign key FK672D67C93AB2FC53

drop table if exists cliente

drop table if exists dependente

drop table if exists endereco
```



```
create table cliente (
    idcliente integer not null auto_increment,
    email varchar(50) not null unique,
    nome varchar(50) not null,
    primary key (idcliente)
)

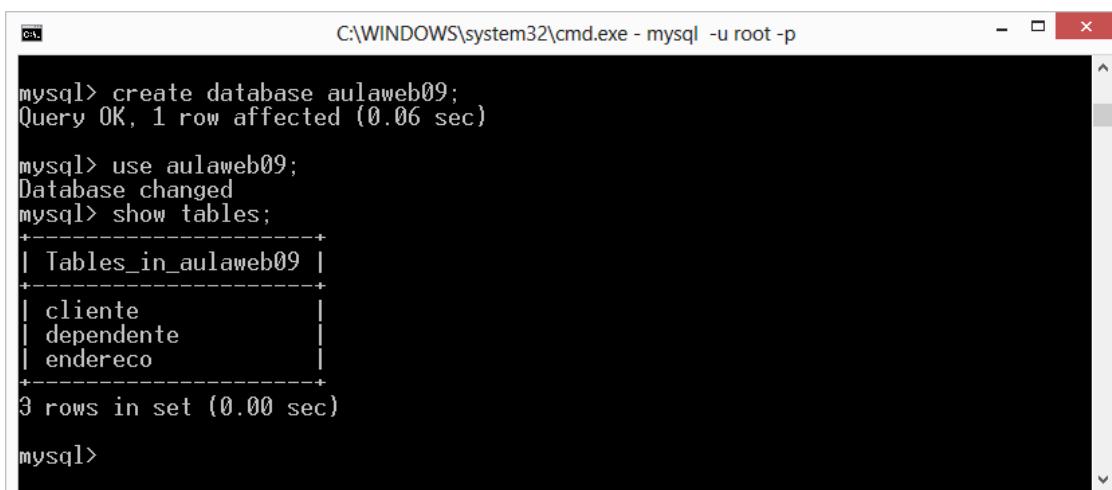
create table dependente (
    iddependente integer not null auto_increment,
    datanasc date not null,
    nome varchar(50) not null,
    idclientefk integer,
    primary key (iddependente)
)

create table endereco (
    idendereco integer not null auto_increment,
    cidade varchar(100) not null,
    estado varchar(25) not null,
    logradouro varchar(200) not null,
    idclientefk integer unique,
    primary key (idendereco)
)

alter table dependente
    add index FKFE6ED5463AB2FC53 (idclientefk),
    add constraint FKFE6ED5463AB2FC53
        foreign key (idclientefk)
        references cliente (idcliente)

alter table endereco
    add index FK672D67C93AB2FC53 (idclientefk),
    add constraint FK672D67C93AB2FC53
        foreign key (idclientefk)
        references cliente (idcliente)
```

### No MySQL...



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> create database aulaweb09;
Query OK, 1 row affected (0.06 sec)

mysql> use aulaweb09;
Database changed
mysql> show tables;
+-----+
| Tables_in_aulaweb09 |
+-----+
| cliente             |
| dependente          |
| endereco            |
+-----+
3 rows in set (0.00 sec)

mysql>
```



### WebContent/index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<html>

    <head>
        </head>

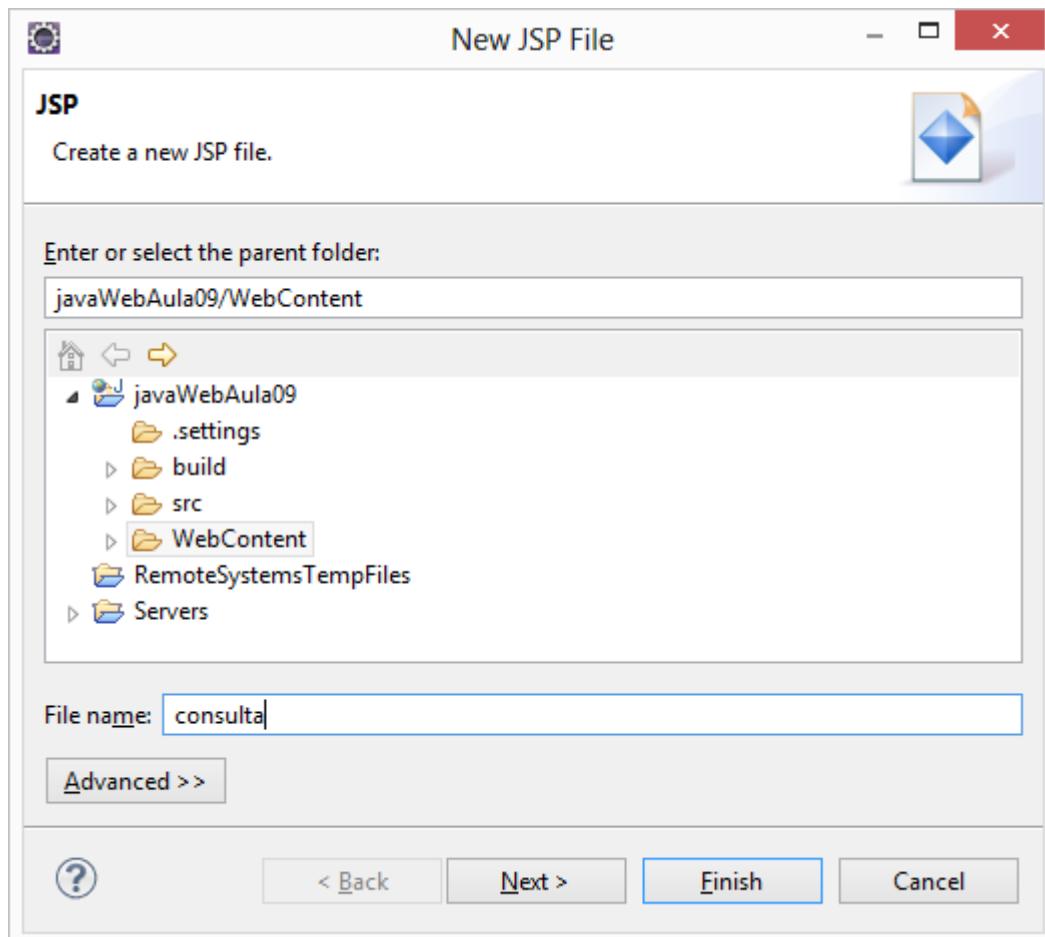
    <body>

        <h3>Bem vindo ao projeto</h3>
        <hr/>

        <a href="consulta.jsp">Consultar Clientes</a>

    </body>

</html>
```





### WebContent/consulta.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<html>

    <head>
        </head>

    <body>

        <h3>Consulta de Clientes</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <form name="formconsulta" method="post" action="">

            <input type="submit" value="Consultar Clientes"/>

        </form>
    </body>
</html>
```





Para que possamos acessar a base de dados através do Hibernate, é necessário uma classe que produza as conexões...

### HibernateUtil.java

Gerar a SessionFactory

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {

            sessionFactory = new AnnotationConfiguration().
                configure("config/mysql_hibernate.cfg.xml").
                buildSessionFactory();
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory
                               creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

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

---

Criando a Classe de persistência...

### HQL - Hibernate Query Language

```
package persistence;

import hibernate.HibernateUtil;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Cliente;
```



```
public class ClienteDao{  
  
    //Armazenar uma conexão aberta com o banco de dados  
    private Session session;  
    //Realizar transações na base de dados (commit, rollback)  
    private Transaction transaction;  
    //Executar consultas na base de dados (HQL - Hibernate Query Language)  
    private Query query;  
  
    //Método para listar todos os cliente da base de dados  
    @SuppressWarnings("unchecked")  
    public List<Cliente> findAll() throws Exception{  
  
        session = HibernateUtil.getSessionFactory().openSession();  
  
        query = session.createQuery("from Cliente");  
        List<Cliente> lista = query.list();  
        //retorna uma lista com os dados  
  
        session.close(); //fechando a conexão  
        return lista; //retornar o conteúdo da lista  
    }  
}
```

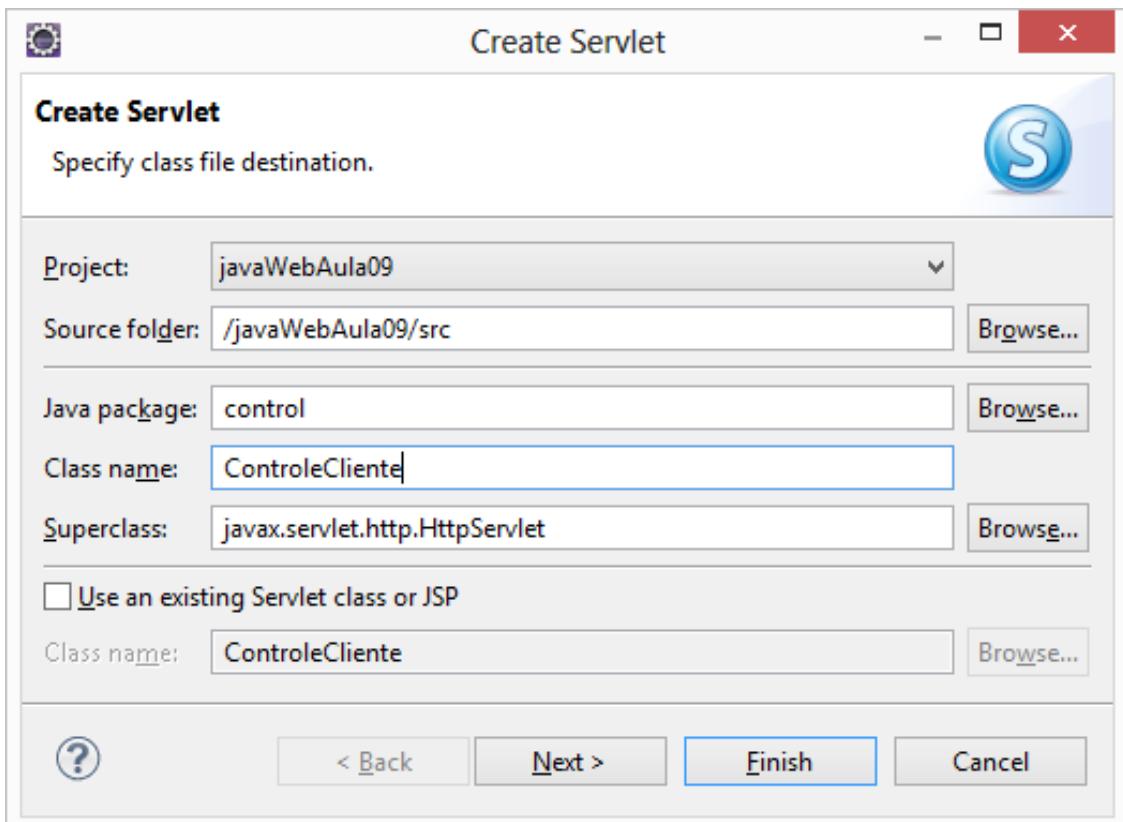
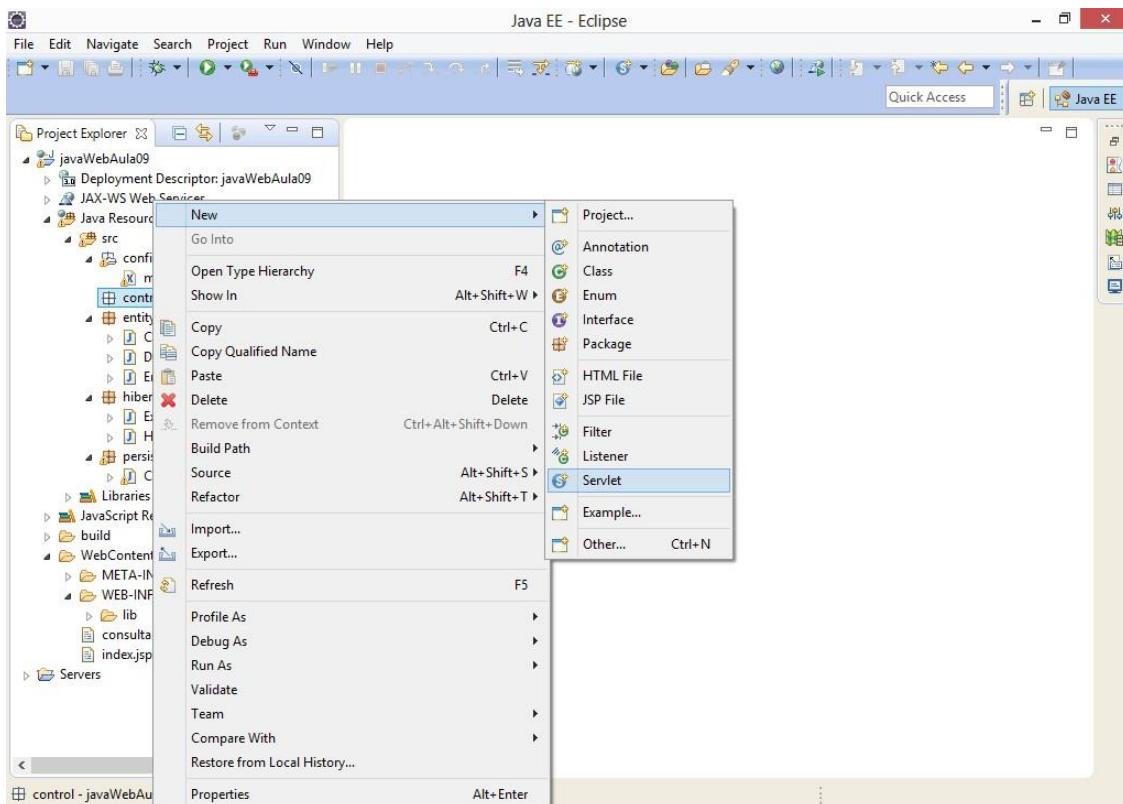
---

## Criando a ação de consulta...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
  
<html>  
  
    <head>  
    </head>  
  
    <body>  
  
        <h3>Consulta de Clientes</h3>  
        <a href="index.jsp">Voltar</a>  
        <hr/>  
  
        <form name="formconsulta" method="post"  
              action="ControleCliente?action=consultar">  
  
            <input type="submit" value="Consultar Clientes"/>  
  
        </form>  
  
    </body>  
  
</html>
```



### Criando o Servlet...





```
package control;

import java.io.IOException;

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 persistence.ClienteDao;

@WebServlet("/ControleCliente")
public class ControleCliente extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleCliente() {
        super();
    }

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

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

        String action = request.getParameter("action");

        if(action.equalsIgnoreCase("consultar")){
            consultar(request, response);
        }
    }

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

        try{
            ClienteDao d = new ClienteDao();
            //classe de persistência
            request.setAttribute("lista", d.findAll());
            //lista de clientes obtida do banco de dados

        }
        catch(Exception e){
            request.setAttribute("msg", "Erro -> "
                               + e.getMessage());
        }
    }
}
```



```
        finally{
            request.getRequestDispatcher("consulta.jsp") .
                forward(request, response);
        }
-----
}
```

## Utilizando as Taglibs JSTL

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fnc" %>

<html>
    <head>
    </head>
    <body>
        <h3>Consulta de Clientes</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>
        <form name="formconsulta" method="post"
            action="ControleCliente?action=consultar">
            <input type="submit" value="Consultar Clientes"/>
        </form>
        <p>
            ${msg}
        </p>
    </body>
</html>
```

-----  
Cadastrando clientes na base de dados...

```
use aulaweb09;

insert into cliente(nome, email) values('Ana', 'ana@gmail.com');
insert into cliente(nome, email) values('Rui', 'rui@gmail.com');
insert into cliente(nome, email) values('Leo', 'leo@gmail.com');

insert into endereço(logradouro, cidade, estado, idclientefk)
values('Rua A', 'Rio de Janeiro', 'RJ', 1);

insert into endereço(logradouro, cidade, estado, idclientefk)
values('Rua B', 'Sao Paulo', 'SP', 2);

insert into endereço(logradouro, cidade, estado, idclientefk)
values('Rua C', 'Belo Horizonte', 'MG', 3);
```



```
insert into dependente(nome, datanasc, idclientefk)
values('Joao', '2000-10-01', 1);

insert into dependente(nome, datanasc, idclientefk)
values('Maria', '2000-10-01', 1);

insert into dependente(nome, datanasc, idclientefk)
values('Carla', '2000-12-20', 2);

select * from cliente;
select * from endereco;
select * from dependente;
```

---

Exibindo a listagem de Clientes na página...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fnc" %>

<html>
    <head>
        </head>
    <body>
        <h3>Consulta de Clientes</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>
        <form name="formconsulta" method="post"
            action="ControleCliente?action=consultar">
            <input type="submit" value="Consultar Clientes"/>
        </form>

        <p>
            ${msg}
        </p>

        <c:forEach items="${lista}" var="c">
            Código do Cliente: ${c.idCliente} <br/>
            Nome: ${c.nome} <br/>
            Email: ${c.email} <br/>

            <hr/>
        </c:forEach>

    </body>
</html>
```

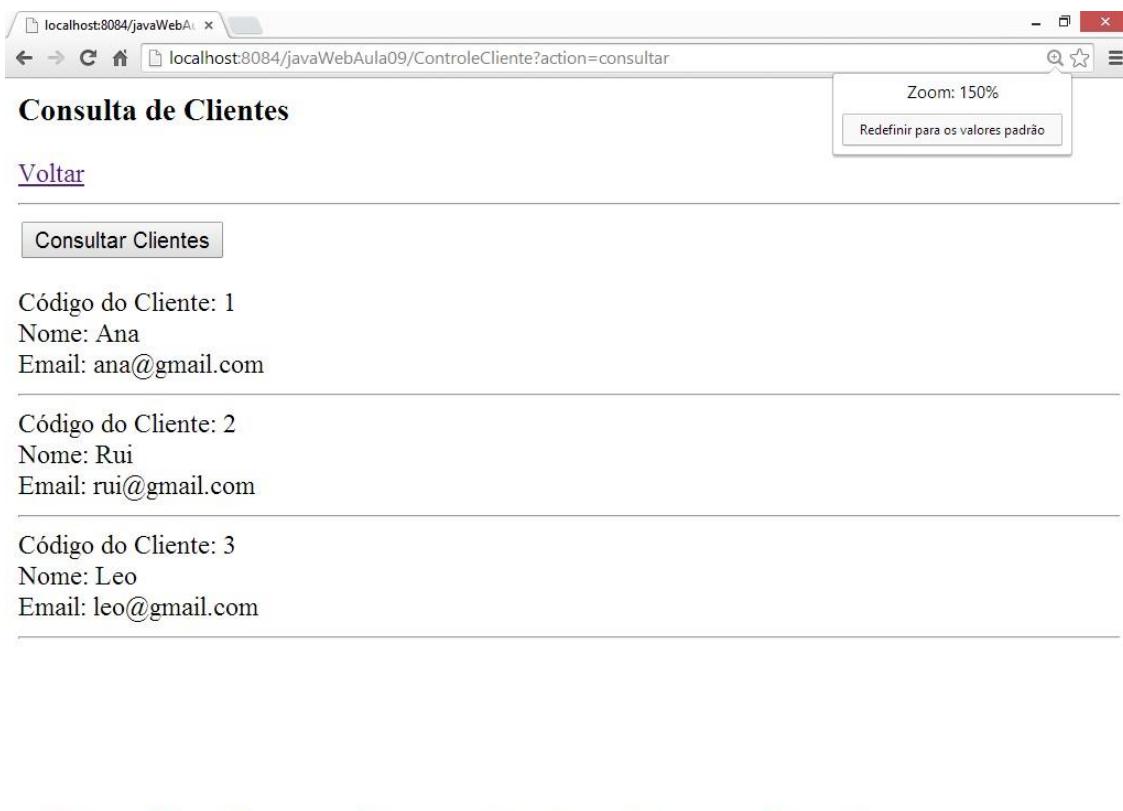


# Java WebDeveloper - BRQ

Terça-feira, 03 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**19**



Código do Cliente: 1  
Nome: Ana  
Email: ana@gmail.com

Código do Cliente: 2  
Nome: Rui  
Email: rui@gmail.com

Código do Cliente: 3  
Nome: Leo  
Email: leo@gmail.com

Exibindo também a residência do Cliente...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fnc" %>

<html>

    <head>
    </head>

    <body>

        <h3>Consulta de Clientes</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <form name="formconsulta" method="post"
              action="ControleCliente?action=consultar">
            <input type="submit" value="Consultar Clientes"/>
        </form>

        <p>
            ${msg}
        </p>
    </body>
</html>
```



# Java WebDeveloper - BRQ

Terça-feira, 03 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**19**

```
<c:forEach items="${lista}" var="c">

    Código do Cliente: ${c.idCliente} <br/>
    Nome: ${c.nome} <br/>
    Email: ${c.email} <br/>

    <br/>

    Residência: ${c.residencia.logradouro} <br/>
    Cidade: ${c.residencia.cidade} <br/>
    Estado: ${c.residencia.estado} <br/>

    <br/>
    <hr/>

</c:forEach>

</body>

</html>
```

**Regra:** Todo relacionamento no Hibernate do tipo **1 para 1** já faz um **join implícito** na consulta, ou seja, toda consulta baseada em cliente já traria os dados do endereço

The screenshot shows a web browser window with two tabs open, both pointing to `localhost:8084/javaWebAula09/ControleCliente?action=consultar`. The main content area is titled "Consulta de Clientes". It contains three entries, each representing a client with their details and an associated address:

- Código do Cliente: 1  
Nome: Ana  
Email: ana@gmail.com  
Residência: Rua A  
Cidade: Rio de Janeiro  
Estado: RJ
- Código do Cliente: 2  
Nome: Rui  
Email: rui@gmail.com  
Residência: Rua B  
Cidade: São Paulo  
Estado: SP
- Código do Cliente: 3  
Nome: Leo  
Email: leo@gmail.com  
Residência: Rua C  
Cidade: Belo Horizonte  
Estado: MG



No hibernate, todo relacionamento Um para Muitos, Muitos para Um ou Muitos para Muitos não é capaz de retornar os dados da classe relacionada como ocorreu no modo 1 para 1.

O Hibernate define carregamento de dados em relacionamentos de 2 maneiras:

☒ **LAZY (default)**

Preguiçoso

Toda consulta em uma Classe A somente traz os dados da Classe B relacionada se este relacionamento for do tipo 1 para 1  
Para trazer dados de classes relacionadas que utilizam outras cardinalidades, deveremos escrever o JOIN na HQL

☒ **EAGER**

Guloso

Toda consulta em uma Classe A, independente da cardinalidade do relacionamento, sempre trará os dados relacionados na Classe B, seja qual for o tipo de relacionamento. O JOIN sempre é feito!

```
package entity;

import java.util.Collection;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "cliente")
public class Cliente {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idcliente")
    private Integer idCliente;

    @Column(name = "nome", length = 50, nullable = false)
    private String nome;

    @Column(name = "email", length = 50, nullable = false,
            unique = true)
    private String email;

    @OneToOne(mappedBy = "morador") //1 para 1
    private Endereco residencia; // TER-1
}
```



# Java WebDeveloper - BRQ

Terça-feira, 03 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**19**

```
//EAGER -> Carregamento guloso (sempre fará
//o JOIN com Dependente)
@OneToMany(mappedBy = "cliente", fetch=FetchType.EAGER)
//1 para Muitos
private Collection<Dependente> dependentes; // TER-Muitos

public Cliente() {
}

public Cliente(Integer idCliente, String nome, String email) {
    super();
    this.idCliente = idCliente;
    this.nome = nome;
    this.email = email;
}

@Override
public String toString() {
    return "Cliente [idCliente=" + idCliente + ", nome="
           + nome + ", email=" + email + "]";
}

public Integer getIdCliente() {
    return idCliente;
}

public void setIdCliente(Integer idCliente) {
    this.idCliente = idCliente;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Endereco getResidencia() {
    return residencia;
}

public void setResidencia(Endereco residencia) {
    this.residencia = residencia;
}

public Collection<Dependente> getDependentes() {
    return dependentes;
}
```



```
public void setDependentes(Collection<Dependente> dependentes) {  
    this.dependentes = dependentes;  
}  
}
```

### Exibindo os dependentes...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
  
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>  
<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fnc" %>  
  
<html>  
  
    <head>  
        </head>  
  
    <body>  
  
        <h3>Consulta de Clientes</h3>  
        <a href="index.jsp">Voltar</a>  
        <hr/>  
  
        <form name="formconsulta" method="post"  
              action="ControleCliente?action=consultar">  
  
            <input type="submit" value="Consultar Clientes"/>  
        </form>  
  
        <p>  
            ${msg}  
        </p>  
  
        <c:forEach items="${lista}" var="c">  
  
            Código do Cliente: ${c.idCliente} <br/>  
            Nome: ${c.nome} <br/>  
            Email: ${c.email} <br/>  
  
            <br/>  
  
            Residência: ${c.residencia.logradouro} <br/>  
            Cidade: ${c.residencia.cidade} <br/>  
            Estado: ${c.residencia.estado} <br/>  
  
            <br/>
```



# Java WebDeveloper - BRQ

Terça-feira, 03 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**19**

Dependentes: \${fnc:length(c.dependentes)}

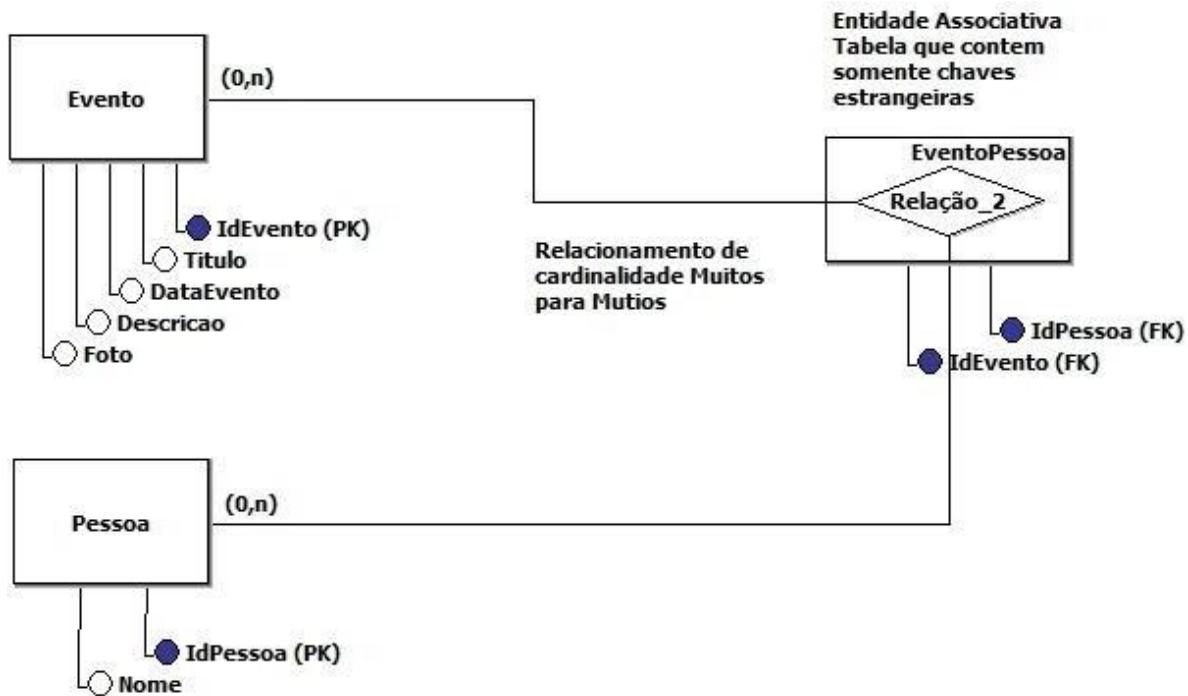
```
<ul>
    <c:forEach items="${c.dependentes}" var="d">
        <li>
            ${d.nome},
            <fmt:formatDate value="${d.dataNascimento}"
                pattern="dd/MM/yyyy"/>
        </li>
    </c:forEach>
</ul>
<hr/>
</c:forEach>
</body>
</html>
```

Resultado...

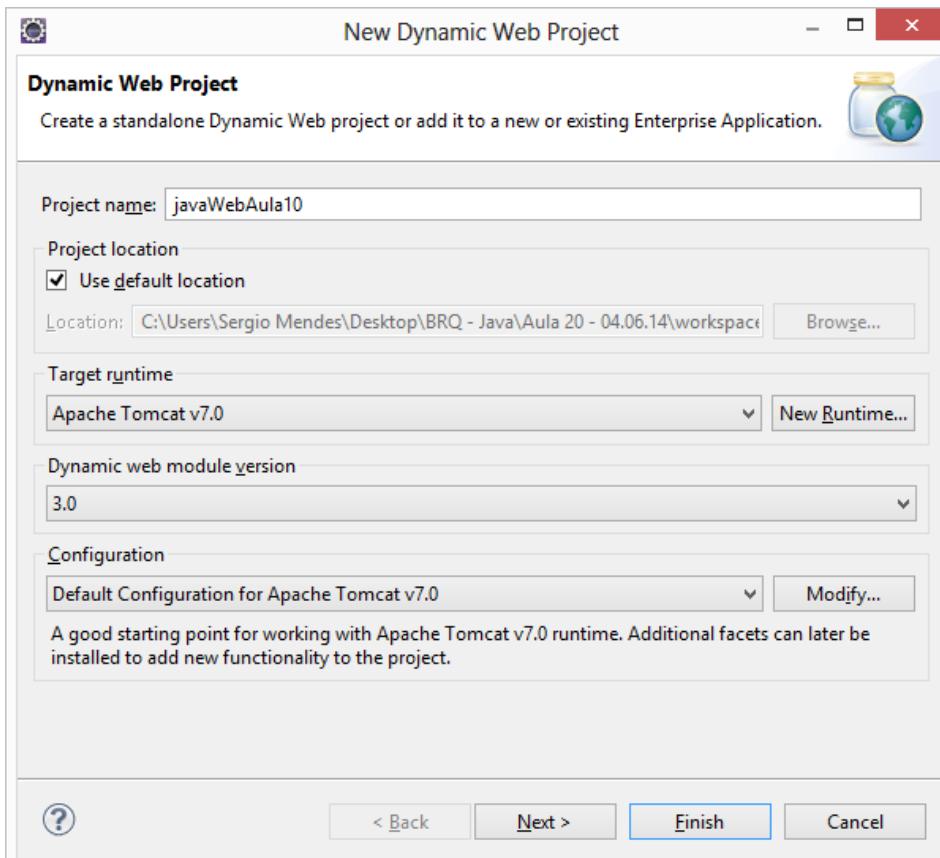
The screenshot shows a web browser window with the URL `localhost:8084/javaWebAula09/ControleCliente?action=consultar`. The page title is "Consulta de Clientes".  
For Client ID 1:  
- Name: Ana  
- Email: ana@gmail.com  
- Address: Rua A  
- City: Rio de Janeiro  
- State: RJ  
- Dependents: 2  
 - João, 01/10/2000  
 - Maria, 01/10/2000  
For Client ID 2:  
- Name: Rui  
- Email: rui@gmail.com  
- Address: Rua B  
- City: São Paulo  
- State: SP  
- Dependents: 1



### Diagrama de Entidade Relacionamento (DER)



### Criando o Projeto...





Script da base de dados...

```
drop database if exists aulaweb10;
create database aulaweb10;
use aulaweb10;

create table pessoa(
    idpessoa      integer          auto_increment,
    nome          varchar(50)       not null,
    primary key(idpessoa));

create table evento(
    idevento      integer          auto_increment,
    titulo        varchar(50)       not null,
    dataevento    date             not null,
    descricao     text             not null,
    foto          varchar(255)      not null,
    primary key(idevento));

create table eventopessoa(
    idevento      integer          not null,
    idpessoa      integer          not null,
    primary key(idevento, idpessoa),
    foreign key(idevento) references evento(idevento),
    foreign key(idpessoa) references pessoa(idpessoa));

show tables;

desc pessoa;
desc evento;
desc eventopessoa;
```

### No MySql...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql>
mysql> desc pessoa;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra   |
+-----+-----+-----+-----+-----+
| idpessoa | int(11) | NO  | PRI | NULL    | auto_increment |
| nome | varchar(50) | NO  |     | NULL    |           |
+-----+-----+-----+-----+-----+
2 rows in set (0.09 sec)

mysql> desc evento;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra   |
+-----+-----+-----+-----+-----+
| idevento | int(11) | NO  | PRI | NULL    | auto_increment |
| titulo | varchar(50) | NO  |     | NULL    |           |
| dataevento | date | NO  |     | NULL    |           |
| descricao | text | NO  |     | NULL    |           |
| foto | varchar(255) | NO  |     | NULL    |           |
+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

mysql> desc eventopessoas;
+-----+-----+-----+-----+-----+
| Field | Type  | Null | Key | Default | Extra   |
+-----+-----+-----+-----+-----+
| idevento | int(11) | NO  | PRI | NULL    |           |
| idpessoa | int(11) | NO  | PRI | NULL    |           |
+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```



### Criando os JavaBeans...

```
package entity;

import java.util.Collection;

public class Pessoa {

    private Integer idPessoa;
    private String nome;
    private Collection<Evento> eventos; // Muitos eventos

    public Pessoa() {
        // Construtor default
    }

    public Pessoa(Integer idPessoa, String nome) {
        super();
        this.idPessoa = idPessoa;
        this.nome = nome;
    }

    @Override
    public String toString() {
        return "Pessoa [idPessoa=" + idPessoa + ", nome="
               + nome + "]";
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Collection<Evento> getEventos() {
        return eventos;
    }

    public void setEventos(Collection<Evento> eventos) {
        this.eventos = eventos;
    }
}
```



```
package entity;

import java.util.Collection;
import java.util.Date;

public class Evento {

    private Integer idEvento;
    private String titulo;
    private Date dataEvento;
    private String descricao;
    private String foto;

    private Collection<Pessoa> pessoas; //Relacionamento -> Muitos

    public Evento() {
    }

    public Evento(Integer idEvento, String titulo, Date dataEvento,
                  String descricao, String foto) {
        super();
        this.idEvento = idEvento;
        this.titulo = titulo;
        this.dataEvento = dataEvento;
        this.descricao = descricao;
        this.foto = foto;
    }

    @Override
    public String toString() {
        return "Evento [idEvento=" + idEvento + ", titulo="
               + titulo + ", dataEvento=" + dataEvento
               + ", descricao=" + descricao + ", foto="
               + foto + "]";
    }

    public Integer getIdEvento() {
        return idEvento;
    }

    public void setIdEvento(Integer idEvento) {
        this.idEvento = idEvento;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
}
```



```
public Date getDataEvento() {
    return dataEvento;
}

public void setDataEvento(Date dataEvento) {
    this.dataEvento = dataEvento;
}

public String getDescricao() {
    return descricao;
}

public void setDescricao(String descricao) {
    this.descricao = descricao;
}

public String getFoto() {
    return foto;
}

public void setFoto(String foto) {
    this.foto = foto;
}

public Collection<Pessoa> getPessoas() {
    return pessoas;
}

public void setPessoas(Collection<Pessoa> pessoas) {
    this.pessoas = pessoas;
}
}
```

## JPA - Java Persistence API

Biblioteca Java para mapeamento Objeto / Relacional

### @Annotations

```
package entity;

import java.util.Collection;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
```



```
@Entity //entidade
@Table(name = "pessoa") //tabela
public class Pessoa {

    @Id //chave primária
    @GeneratedValue(strategy = GenerationType.AUTO) //auto_increment
    @Column(name = "idpessoa") //nome do campo
    private Integer idPessoa;

    @Column(name = "nome", length = 50, nullable = false)
    //nome do campo
    private String nome;

    //mappedBy -> nome do atributo Collection onde
    //foi mapeado a JoinTable
    @ManyToMany(mappedBy = "pessoas")
    //muitas pessoas para muitos eventos
    private Collection<Evento> eventos; // Muitos eventos

    public Pessoa() {
        // Construtor default
    }

    public Pessoa(Integer idPessoa, String nome) {
        super();
        this.idPessoa = idPessoa;
        this.nome = nome;
    }

    @Override
    public String toString() {
        return "Pessoa [idPessoa=" + idPessoa + ", nome="
            + nome + "]";
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```



```
public Collection<Evento> getEventos() {  
    return eventos;  
}  
  
public void setEventos(Collection<Evento> eventos) {  
    this.eventos = eventos;  
}  
}
```

---

```
package entity;  
  
import java.util.Collection;  
import java.util.Date;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.JoinTable;  
import javax.persistence.ManyToMany;  
import javax.persistence.Table;  
import javax.persistence.Temporal;  
import javax.persistence.TemporalType;  
  
@Entity  
@Table(name = "evento")  
public class Evento {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    @Column(name = "idevento")  
    private Integer idEvento;  
  
    @Column(name = "titulo", length = 50, nullable = false)  
    private String titulo;  
  
    @Temporal(TemporalType.DATE) //somente data  
    @Column(name = "dataevento", nullable = false)  
    private Date dataEvento;  
  
    @Column(name = "descricao", nullable = false)  
    private String descricao;  
  
    @Column(name = "foto", length = 255, nullable = false)  
    private String foto;
```



```
@ManyToMany
@JoinTable( //mapeamento da entidade associativa
            name = "eventopessoa", //nome da tabela de juncão
            joinColumns = @JoinColumn(name = "idevento"),
            //chave estrangeira para a tabela evento
            inverseJoinColumns = @JoinColumn(name = "idpessoa")
            //chave estrangeira para a tabela pessoa
)
private Collection<Pessoa> pessoas; // Relacionamento -> Muitos

public Evento() {
}

public Evento(Integer idEvento, String titulo, Date dataEvento,
              String descricao, String foto) {
    super();
    this.idEvento = idEvento;
    this.titulo = titulo;
    this.dataEvento = dataEvento;
    this.descricao = descricao;
    this.foto = foto;
}

@Override
public String toString() {
    return "Evento [idEvento=" + idEvento + ", titulo="
           + titulo + ", dataEvento=" + dataEvento +",
           descricao=" + descricao + ", foto=" + foto + "]";
}

public Integer getIdEvento() {
    return idEvento;
}

public void setIdEvento(Integer idEvento) {
    this.idEvento = idEvento;
}

public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public Date getDataEvento() {
    return dataEvento;
}
```



```
public void setDataEvento(Date dataEvento) {
    this.dataEvento = dataEvento;
}

public String getDescricao() {
    return descricao;
}

public void setDescricao(String descricao) {
    this.descricao = descricao;
}

public String getFoto() {
    return foto;
}

public void setFoto(String foto) {
    this.foto = foto;
}

public Collection<Pessoa> getPessoas() {
    return pessoas;
}

public void setPessoas(Collection<Pessoa> pessoas) {
    this.pessoas = pessoas;
}
}
```

---

## Arquivo de configuração para o hibernate **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <session-factory>

        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect
        </property>

        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>

    </session-factory>

```



```
<property name="hibernate.connection.url">  
    jdbc:mysql://localhost:3306/aulaweb10  
</property>  
  
<property name="hibernate.connection.username">root</property>  
<property name="hibernate.connection.password"></property>  
  
<property name="hibernate.show_sql">true</property>  
<property name="hibernate.format_sql">true</property>  
  
<mapping class="entity.Pessoa"/>  
<mapping class="entity.Evento"/>  
  
</session-factory>  
  
</hibernate-configuration>
```

## HibernateUtil

Classe para gerar as conexões com o banco de dados através de um componente do hibernate denominado: **SessionFactory**  
Este componente utiliza o arquivo **cfg.xml**

```
package hibernate;  
  
import org.hibernate.cfg.AnnotationConfiguration;  
import org.hibernate.SessionFactory;  
  
public class HibernateUtil {  
  
    private static final SessionFactory sessionFactory;  
  
    static {  
        try {  
  
            sessionFactory = new AnnotationConfiguration().  
                configure("config/mysql_hibernate.cfg.xml").  
                buildSessionFactory();  
        } catch (Throwable ex) {  
  
            System.err.println("Initial SessionFactory  
creation failed." + ex);  
            throw new ExceptionInInitializerError(ex);  
        }  
    }  
  
    public static SessionFactory getSessionFactory() {  
        return sessionFactory;  
    }  
}
```



### Classes de persistência...

```
package persistence;

import hibernate.HibernateUtil;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Pessoa;

public class PessoaDao {

    private Session session;
    //armazenar conexão aberta com o BD
    private Transaction transaction;
    //executar rotinas transacionais
    private Query query;
    //consultas (HQL -> Hibernate Query Language)

    public void create(Pessoa p) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        //conexão aberta
        transaction = session.beginTransaction();
        //iniciar uma transação
        session.save(p); //insert
        transaction.commit(); //executar
        session.close(); //fechar a conexão
    }

    public void remove(Pessoa p) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        //conexão aberta
        transaction = session.beginTransaction();
        //iniciar uma transação
        session.delete(p); //delete
        transaction.commit(); //executar
        session.close(); //fechar a conexão
    }

    public void update(Pessoa p) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        //conexão aberta
        transaction = session.beginTransaction();
        //iniciar uma transação
        session.update(p); //update
    }
}
```



```
        transaction.commit(); //executar
        session.close(); //fechar a conexão
    }

    public Pessoa findById(Integer idPessoa) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        Pessoa p = (Pessoa) session.get(Pessoa.class, idPessoa);
        //buscando pela chave primária

        session.close();
        return p; //retornar Pessoa
    }

    public List<Pessoa> findAll() throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        //conexão aberta
        query = session.createQuery("from Pessoa");
        List<Pessoa> lista = query.list();
        //gera a lista com os registros

        session.close();
        return lista; //retornar a lista
    }
}

-----
package persistence;

import hibernate.HibernateUtil;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Evento;
import entity.Pessoa;

public class EventoDao {

    private Session session;
    //armazenar conexão aberta com o BD
    private Transaction transaction;
    //executar rotinas transacionais
    private Query query;
    //consultas (HQL -> Hibernate Query Language)
}
```



```
public void create(Evento e) throws Exception{

    session = HibernateUtil.getSessionFactory().openSession();
    //conexão aberta
    transaction = session.beginTransaction();
    //iniciar uma transação
    session.save(e); //insert
    transaction.commit(); //executar
    session.close(); //fechar a conexão
}

public void remove(Evento e) throws Exception{

    session = HibernateUtil.getSessionFactory().openSession();
    //conexão aberta
    transaction = session.beginTransaction();
    //iniciar uma transação
    session.delete(e); //delete
    transaction.commit(); //executar
    session.close(); //fechar a conexão
}

public void update(Evento e) throws Exception{

    session = HibernateUtil.getSessionFactory().openSession();
    //conexão aberta
    transaction = session.beginTransaction();
    //iniciar uma transação
    session.update(e); //update
    transaction.commit(); //executar
    session.close(); //fechar a conexão
}

public Evento findById(Integer idEvento) throws Exception{

    session = HibernateUtil.getSessionFactory().openSession();
    Evento e = (Evento) session.get(Evento.class, idEvento);
    //buscando pela chave primária
    session.close();
    return e; //retornar Pessoa
}

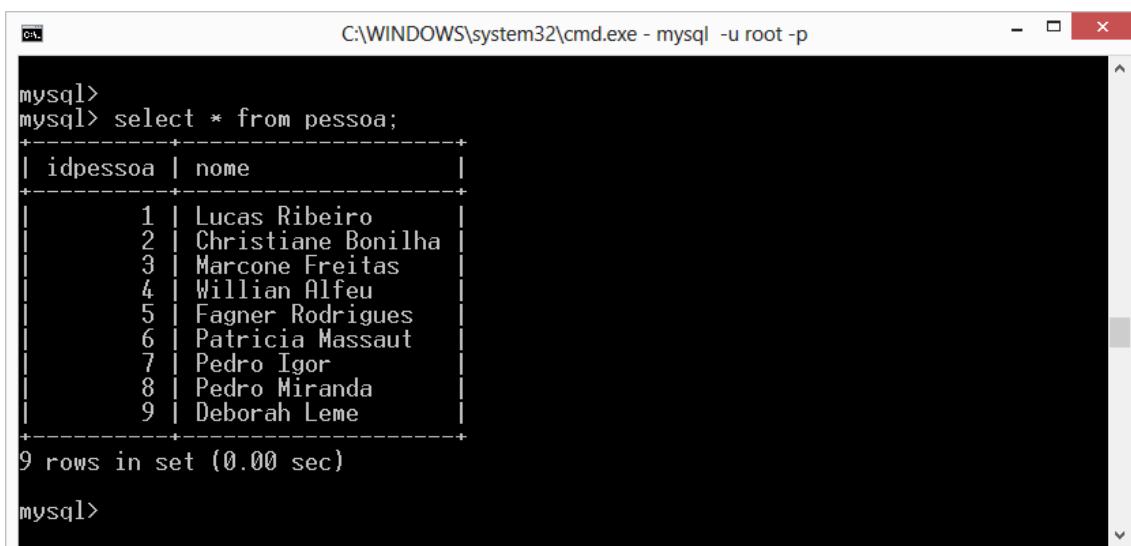
public List<Evento> findAll() throws Exception{
    session = HibernateUtil.getSessionFactory().openSession();
    //conexão aberta
    query = session.createQuery("from Evento");
    List<Evento> lista = query.list();
    //gera a lista com os registros
    session.close();
    return lista; //retornar a lista
}
```



### Cadastrando Pessoas na base de dados...

```
insert into pessoa values(null, 'Lucas Ribeiro');
insert into pessoa values(null, 'Christiane Bonilha');
insert into pessoa values(null, 'Marcone Freitas');
insert into pessoa values(null, 'Willian Alfeu');
insert into pessoa values(null, 'Fagner Rodrigues');
insert into pessoa values(null, 'Patricia Massaut');
insert into pessoa values(null, 'Pedro Igor');
insert into pessoa values(null, 'Pedro Miranda');
insert into pessoa values(null, 'Deborah Leme');

select * from pessoa;
```



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql>
mysql> select * from pessoa;
+-----+-----+
| idpessoa | nome      |
+-----+-----+
|       1 | Lucas Ribeiro |
|       2 | Christiane Bonilha |
|       3 | Marcone Freitas |
|       4 | Willian Alfeu |
|       5 | Fagner Rodrigues |
|       6 | Patricia Massaut |
|       7 | Pedro Igor |
|       8 | Pedro Miranda |
|       9 | Deborah Leme |
+-----+-----+
9 rows in set (0.00 sec)

mysql>
```

### Criando a página inicial...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>
    <head>
        </head>
    <body>
        <h3>Projeto Controle de Eventos</h3>
        <hr/>

        <ul>
            <li> <a href="cadastro.jsp">Marcar Eventos</a> </li>
            <li> <a href="consulta.jsp">Exibir Eventos</a> </li>
        </ul>

    </body>
</html>
```



### Página de Cadastro de eventos...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
    </head>

    <body>
        <h3>Marcar Eventos</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <form name="formcadastro" method="post" action="">

            Título do Evento: <br/>
            <input type="text" name="titulo"
                   required="required"/>
            <br/><br/>

            Data do Evento: <br/>
            <input type="date" name="dataevento"
                   required="required"/>
            <br/><br/>

            Descrição: <br/>
            <textarea name="descricao" required="required"
                      cols="42" rows="4"></textarea>
            <br/><br/>

            Foto do Evento (URL): <br/>
            <input type="text" name="foto" size="50"
                   required="required"/>
            <br/><br/>

            Selecione o(s) participante(s): <br/>
            <select name="participantes" required="required"
                    multiple="multiple">
            </select>
            <br/><br/>

            <input type="submit" value="Cadastrar Evento"/>

            <p>
                <strong> ${msg} </strong>
            </p>

        </form>
    </body>
</html>
```



# Java WebDeveloper - BRQ

Quarta-feira, 04 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**20**

localhost:8082/javaWebAula10/cadastro.jsp

**Marcar Eventos**

[Voltar](#)

Titulo do Evento:

Data do Evento:  
 dd/mm/aaaa

Descrição:

Foto do Evento (URL):

Selecione o(s) participante(s):



New Java Class

**Java Class**  
Create a new Java class.

Source folder:

Package:

Enclosing type:

Name:

Modifiers:  public  package  private  protected  
 abstract  final  static

Superclass:

Interfaces:



```
package beans;

import java.util.List;

import persistence.PessoaDao;
import entity.Pessoa;

//Bean para retornar dados de pessoas
public class PessoasBean {

    private List<Pessoa> listagemPessoas; //null;

    public List<Pessoa> getListagemPessoas() {

        try{

            PessoaDao d = new PessoaDao();
            listagemPessoas = d.findAll();
        }
        catch(Exception e){
            e.printStackTrace(); //log do tomcat (servidor)
        }

        return listagemPessoas;
    }

    public void setListagemPessoas(List<Pessoa> listagemPessoas) {
        this.listagemPessoas = listagemPessoas;
    }
}
```

Exibindo os dados de Pessoa na página  
de Cadastro através de **JSTL** (tagLibraries)

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<jsp:useBean class="beans.PessoasBean" id="pb" scope="page"/>

<html>
    <head>
    </head>

    <body>

        <h3>Marcar Eventos</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>
```



```
<form name="formcadastro" method="post" action="">

    Título do Evento: <br/>
    <input type="text" name="titulo"
        required="required"/>
    <br/><br/>

    Data do Evento: <br/>
    <input type="date" name="dataevento"
        required="required"/>
    <br/><br/>

    Descrição: <br/>
    <textarea name="descricao" required="required"
        cols="42" rows="4"></textarea>
    <br/><br/>

    Foto do Evento (URL): <br/>
    <input type="text" name="foto" size="50"
        required="required"/>
    <br/><br/>

    Selecione o(s) participante(s): <br/>
    <select name="participantes" required="required"
        multiple="multiple"
        style="width : 320px; height: 100px;">

        <c:forEach items="${pb.listagemPessoas}"
            var="p">

            <option value="${p.idPessoa}">
                ${p.nome}
            </option>

        </c:forEach>

        </select>
        <br/><br/>

        <input type="submit" value="Cadastrar Evento"/>

        <p>
            <strong> ${msg} </strong>
        </p>

    </form>
</body>
</html>
```



### Resultado...

localhost:8082/javaWebAula10/cadastro.jsp

**Marcar Eventos**

[Voltar](#)

Título do Evento:

Data do Evento:

dd/mm/aaaa

Descrição:

Foto do Evento (URL):

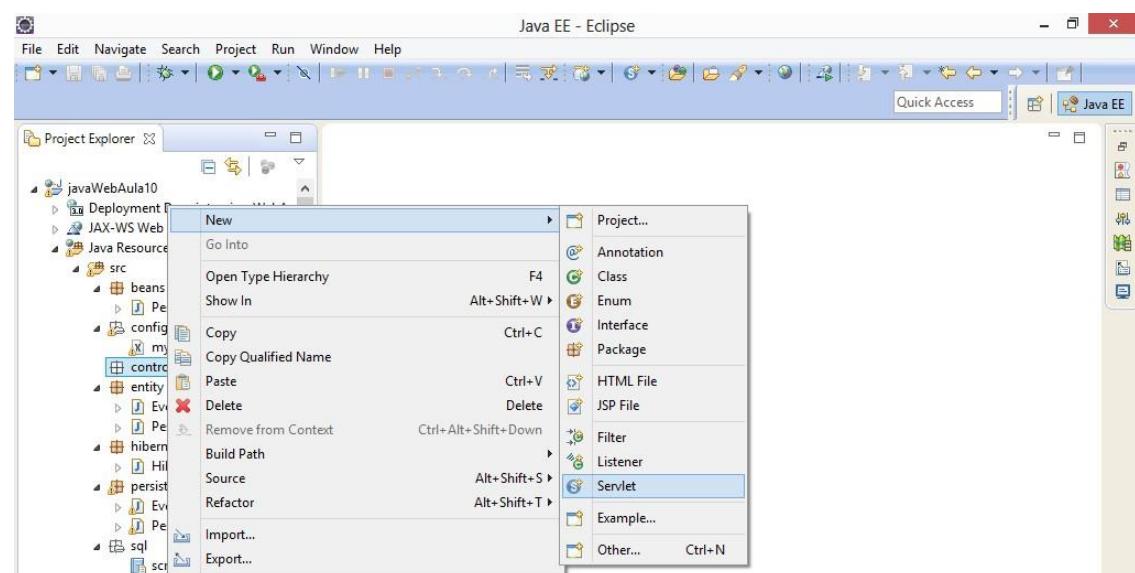
Selezione o(s) participante(s):

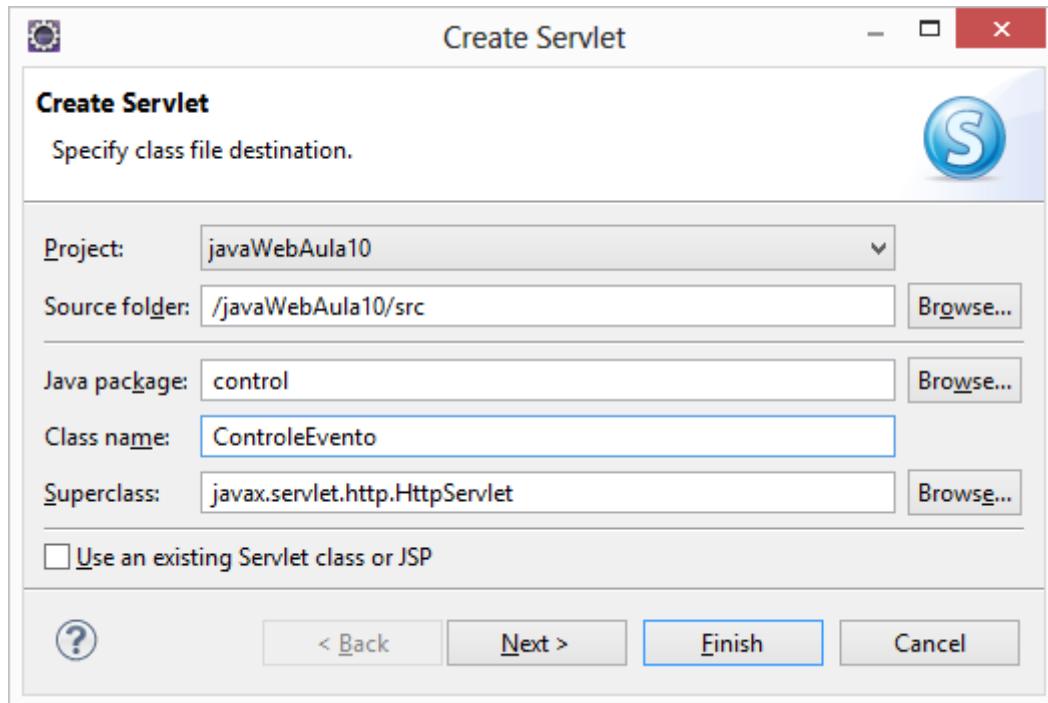
- Lucas Ribeiro
- Christiane Bonilha
- Marcone Freitas
- Willian Alfeu
- Fagner Rodrigues
- Patricia Massaut

Cadastrar Evento

### Criando o evento de cadastro...

```
<form name="formcadastro" method="post"
      action="ControleEvento?action=cadastrar">
```





- Classe para tratamento da data...

```
package util;

import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

public class Data {

    //Método estático para receber uma String e
    //retorna-la em formato Date
    //data -> formato yyyy/MM/dd
    public static Date convertToDate(String data) throws Exception{

        int ano = new Integer( data.substring(0, 4) );
        int mes = new Integer( data.substring(5, 7) );
        int dia = new Integer( data.substring(8, 10) );

        Calendar cal = new GregorianCalendar(ano, mes - 1, dia);

        //retornar o conteúdo do Calendar como Date
        return cal.getTime();
    }
}
```



### Cadastrando somente o evento...

```
package control;

import java.io.IOException;

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 persistence.EventoDao;
import util.Data;
import entity.Evento;

@WebServlet("/ControleEvento")
public class ControleEvento extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleEvento() {
        super();
    }

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

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

        String action = request.getParameter("action");

        if(action.equalsIgnoreCase("cadastrar")){
            cadastrar(request, response);
        }
    }

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

        try{
            Evento e = new Evento();

            e.setTitulo( request.getParameter("titulo") );
            e.setFoto( request.getParameter("foto") );
            e.setDescricao( request.getParameter("descricao") );
            e.setDataEvento( Data.convertToDate(request.
                getParameter("dataevento")) );

            EventoDao d = new EventoDao();
            d.create(e); //gravando
        }
    }
}
```



# Java WebDeveloper - BRQ

Quarta-feira, 04 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**20**

```
        request.setAttribute("msg", "Evento criado  
        com sucesso.");  
    }  
    catch(Exception e){  
        request.setAttribute("msg", "Erro ao  
        criar evento -> " + e.getMessage());  
    }  
    finally{  
        request.getRequestDispatcher("cadastro.jsp") .  
        forward(request, response);  
    }  
}  
}
```

## Executando...

## No banco de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p  
mysql> select * from evento\G  
***** 1. row *****  
idevento: 1  
titulo: Prova de Java  
dataevento: 2014-06-04  
descricao: Prova de Java Web  
foto: http://4.bp.blogspot.com/-8J0llxbclmU/U065MF6LL5I/AAAAAAAAY0/OB9NmJBgwj8/s1600/004.png  
1 row in set (0.00 sec)  
mysql> -
```



## Cadastrando o evento com participantes

```
package control;

import java.io.IOException;
import java.util.ArrayList;

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 persistence.EventoDao;
import persistence.PessoaDao;
import util.Data;
import entity.Evento;
import entity.Pessoa;

@WebServlet("/ControleEvento")
public class ControleEvento extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleEvento() {
        super();
    }

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

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

        String action = request.getParameter("action");

        if(action.equalsIgnoreCase("cadastrar")){
            cadastrar(request, response);
        }
    }

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

        try{
            Evento e = new Evento();
            e.setPessoas( new ArrayList<Pessoa>() );
            //espaço de memória para Collection

            e.setTitulo( request.getParameter("titulo") );
            e.setFoto( request.getParameter("foto") );
            e.setDescricao( request.getParameter("descricao") );
            e.setDataEvento( Data.convertToDate(
                request.getParameter("dataevento")) );
        }
    }
}
```



```
//resgatando o campo de seleção
String[] participantes = request.getParameterValues
("participantes");

PessoaDao pd = new PessoaDao();

for(int i = 0; i < participantes.length; i++){
//percorrendo o vetor

    int idPessoa = new Integer(participantes[i]);
    //cada posição do vetor
    e.getPessoas().add( pd.findById(idPessoa) );
}

EventoDao d = new EventoDao();
d.create(e); //gravando

request.setAttribute("msg", "Evento criado
com sucesso.");
}

catch(Exception e){
    request.setAttribute("msg", "Erro ao criar
    evento -> " + e.getMessage());
}

finally{
    request.getRequestDispatcher("cadastro.jsp").forward
    (request, response);
}
}

}
```

### Resultado...



#### Marcar Eventos

[Voltar](#)

Título do Evento:

Data do Evento:

 dd/mm/aaaa

Descrição:

Foto do Evento (URL):

Selecione o(s) participante(s):

- Lucas Ribeiro
- Christiane Bonilha
- Marcone Freitas
- Willian Alfeu
- Fagner Rodrigues
- Patricia Massaut

Evento criado com sucesso.



### No banco de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
***** 2. row *****
idevento: 2
    titulo: Prova de Java
dataevento: 2014-06-05
    descricao: Prova de Hibernate
    foto: http://4.bp.blogspot.com/-8J011xbcLmU/U065MF6LL5I/AAAAAAAAY0/0B9NmJ
Bgwj8/s1600/004.png
2 rows in set (0.00 sec)

mysql> select * from eventopessoa;
+-----+-----+
| idevento | idpessoa |
+-----+-----+
|       2 |       1 |
|       2 |       2 |
|       2 |       3 |
|       2 |       6 |
+-----+-----+
4 rows in set (0.00 sec)

mysql> -
```

### Realizando a Consulta de Eventos...

Incluindo o parâmetro **EAGER** para que a consulta de eventos carregue também os dados dos participantes.

```
package entity;

import java.util.Collection;
import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "evento")
public class Evento {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idevento")
    private Integer idEvento;
```



```
@Column(name = "titulo", length = 50, nullable = false)
private String titulo;

@Temporal(TemporalType.DATE) //somente data
@Column(name = "dataevento", nullable = false)
private Date dataEvento;

@Column(name = "descricao", nullable = false)
private String descricao;

@Column(name = "foto", length = 255, nullable = false)
private String foto;

@ManyToMany (fetch=FetchType.EAGER)
@JoinTable( //mapeamento da entidade associativa
            name = "eventopessoa",
            //nome da tabela de junção
            joinColumns = @JoinColumn(name = "idevento"),
            //chave estrangeira para a tabela evento
            inverseJoinColumns = @JoinColumn(name =
            "idpessoa") //chave estrangeira para
                           //a tabela pessoa
)
private Collection<Pessoa> pessoas;
// Relacionamento -> Muitos

public Evento() {
}

public Evento(Integer idEvento, String titulo,
              Date dataEvento, String descricao, String foto) {
    super();
    this.idEvento = idEvento;
    this.titulo = titulo;
    this.dataEvento = dataEvento;
    this.descricao = descricao;
    this.foto = foto;
}

@Override
public String toString() {
    return "Evento [idEvento=" + idEvento + ", titulo="
           + titulo + ", dataEvento=" + dataEvento +",
           descricao=" + descricao + ", foto="
           + foto + "]";
}

public Integer getIdEvento() {
    return idEvento;
}

public void setIdEvento(Integer idEvento) {
    this.idEvento = idEvento;
}
```



```
public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public Date getDataEvento() {
    return dataEvento;
}

public void setDataEvento(Date dataEvento) {
    this.dataEvento = dataEvento;
}

public String getDescricao() {
    return descricao;
}

public void setDescricao(String descricao) {
    this.descricao = descricao;
}

public String getFoto() {
    return foto;
}

public void setFoto(String foto) {
    this.foto = foto;
}

public Collection<Pessoa> getPessoas() {
    return pessoas;
}

public void setPessoas(Collection<Pessoa> pessoas) {
    this.pessoas = pessoas;
}
}
```

## Criando a Classe EventosBean para listagem dos eventos...

```
package beans;

import java.util.List;

import persistence.EventoDao;
import entity.Evento;

//Bean para retornar dados de eventos
public class EventosBean {
```



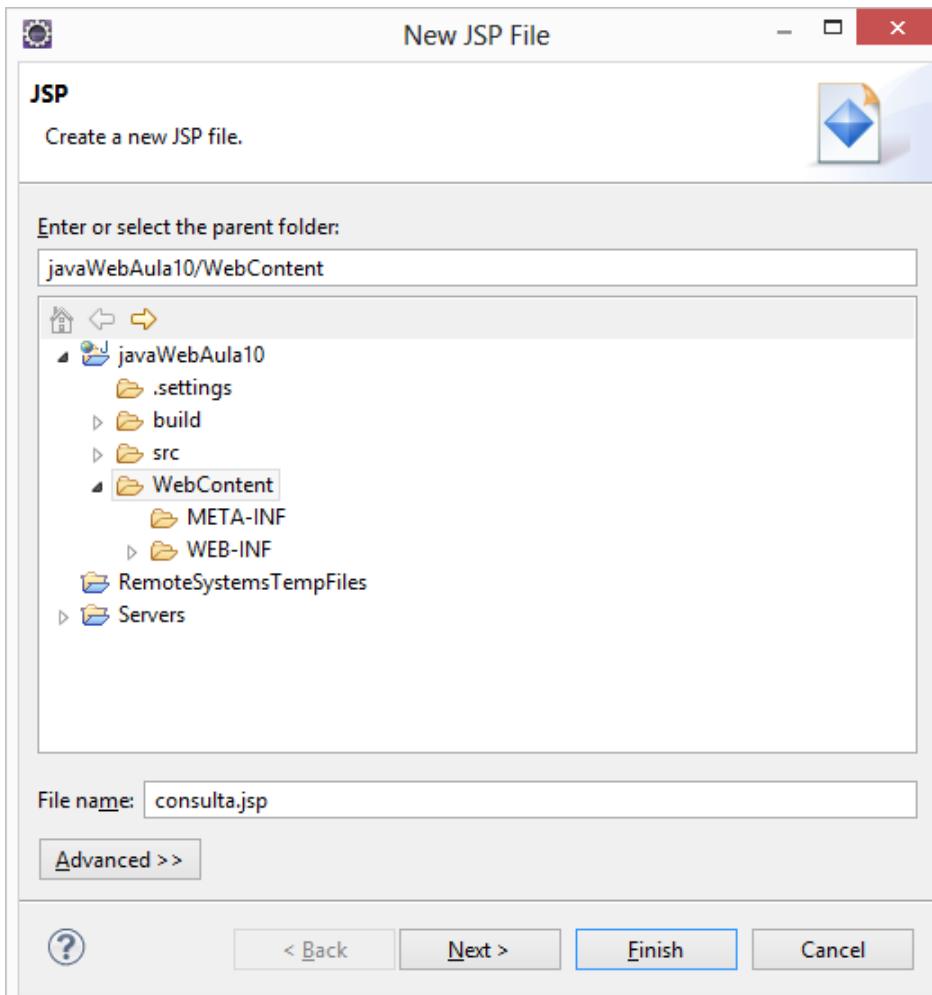
```
private List<Evento> listagemEventos; //null;

public List<Evento> getListagemEventos() {
    try{

        EventoDao d = new EventoDao();
        listagemEventos = d.findAll();
    }
    catch(Exception e){
        e.printStackTrace(); //log do tomcat (servidor)
    }
    return listagemEventos;
}

public void setListagemEvento(List<Evento> listagemEventos) {
    this.listagemEventos = listagemEventos;
}
}
```

Criando a página de consulta...





```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="f" %>

<jsp:useBean class="beans.EventosBean" id="eb" scope="page"/>

<html>

    <head>

        </head>

    <body>

        <h3>Relação de Eventos cadastrados:</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <c:forEach items="${eb.listagemEventos}" var="e">

            <h4> ${e.titulo} </h4>

            Marcado para: <f:formatDate value="${e.dataEvento}"
                pattern="EEEE dd/MM/yyyy"/>

            <p>
                
            </p>

            <p>
                ${e.descricao}
            </p>

            Participantes:

            <ul>
                <c:forEach items="${e.pessoas}" var="p">
                    <li> ${p.nome} </li>
                </c:forEach>
            </ul>

            <hr/>

        </c:forEach>

    </body>

</html>
```



# Java WebDeveloper - BRQ

Quarta-feira, 04 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento Objeto Relacional  
utilizando JPA - Java Persistence API.

Aula  
**20**

## Resultado...

Entrada - sergio.coti@gmail.com | localhost:8082/javaWebAula10 | campus2\_sala\_aula1.jpg | Facebook

localhost:8082/javaWebAula10/consulta.jsp

Relação de Eventos cadastrados:

Voltar

Prova de Java

Marcado para: Quarta-feira 04/06/2014



Prova de Java Web

Participantes:

Prova de Java

Marcado para: Quinta-feira 05/06/2014



Prova de Hibernate

Participantes:

- Lucas Ribeiro
- Christiane Bonilha
- Marcone Freitas
- Patricia Massaut

13:07  
06/06/2014



# Java WebDeveloper - BRQ

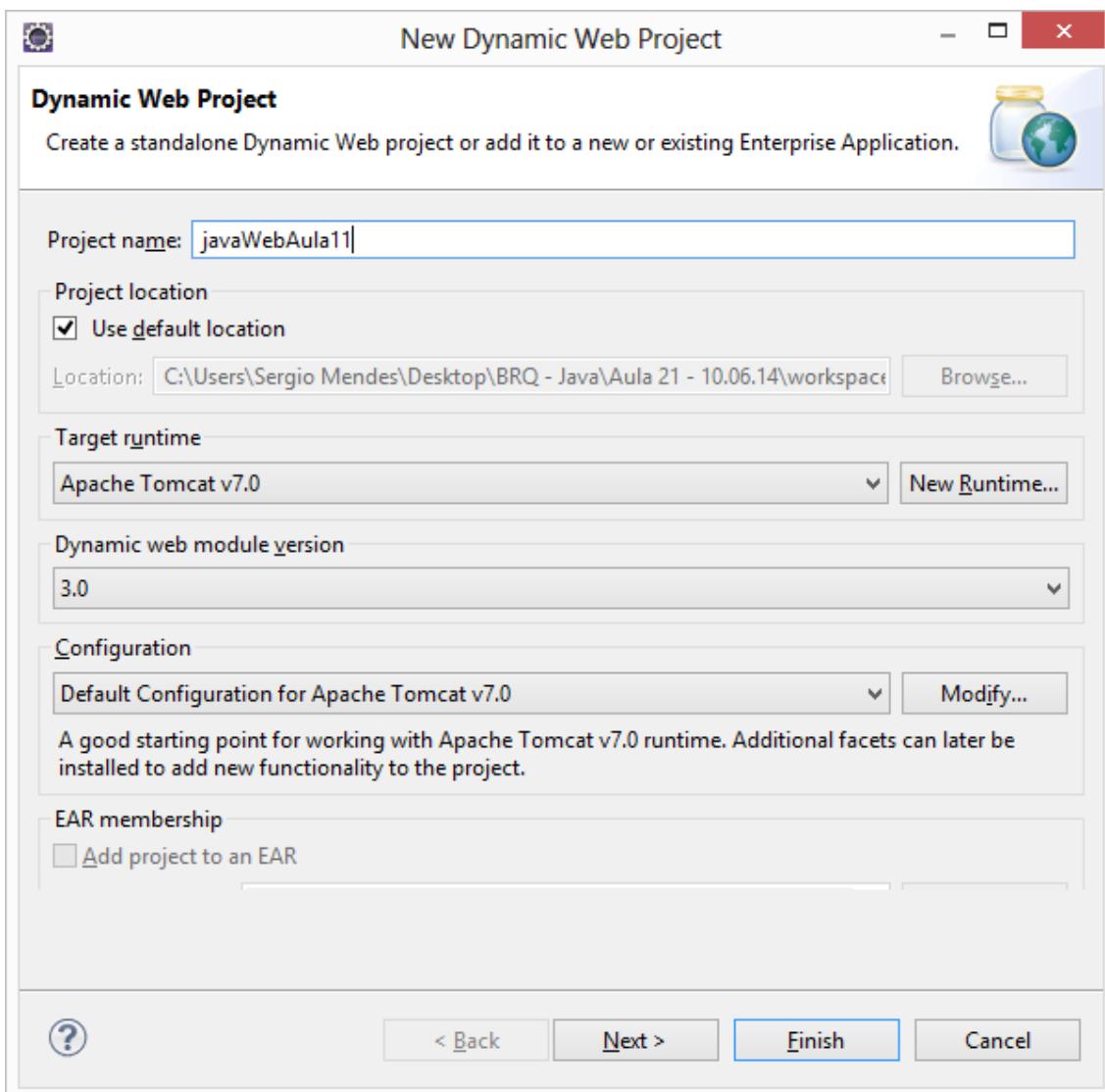
Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

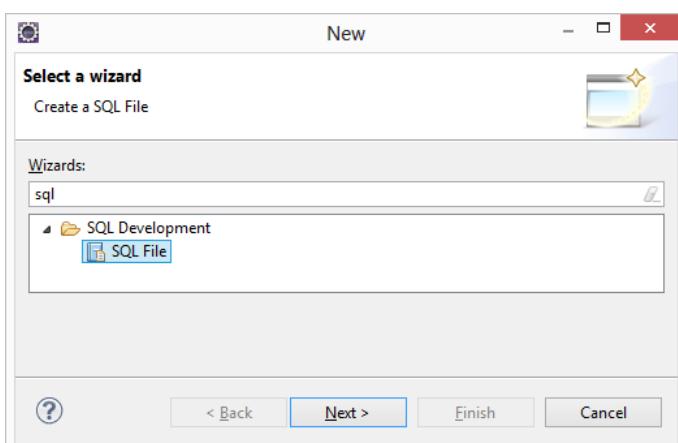
21

Criando o Projeto...



## Script do banco de dados

Modelagem 1 para 1



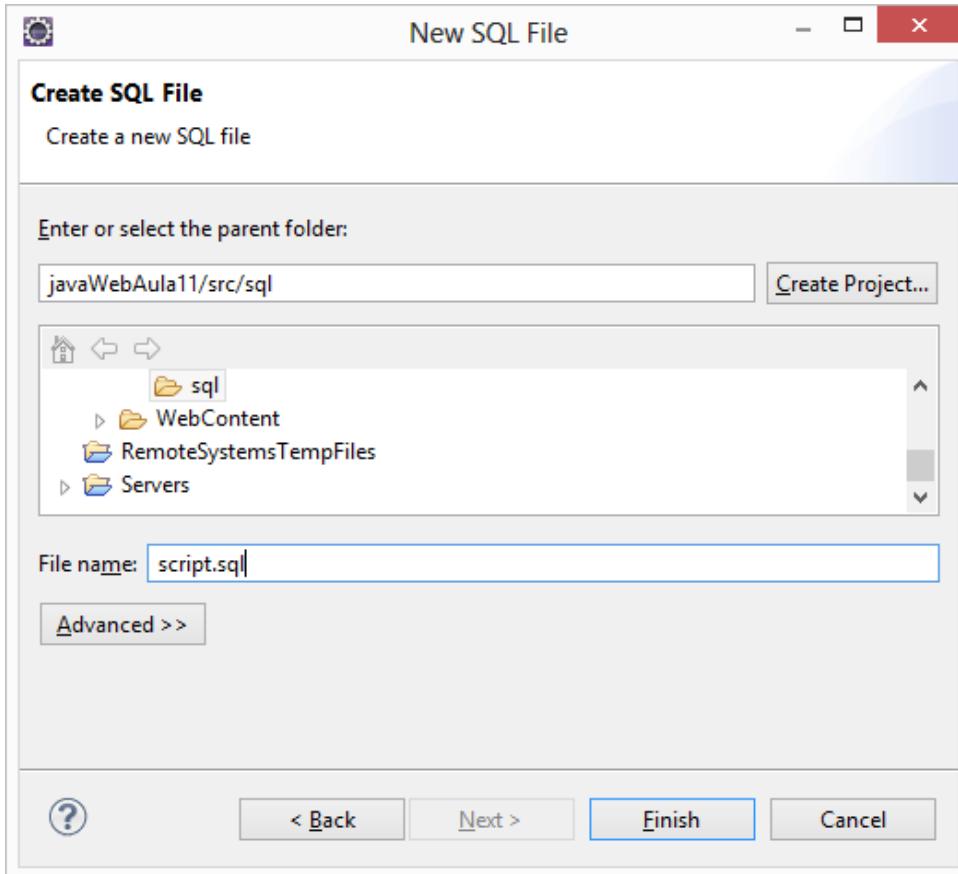


# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21



```
drop database if exists aulaweb11;
create database aulaweb11;
use aulaweb11;

create table pessoa(
    idpessoa      integer      auto_increment,
    nome          varchar(50)   not null,
    email         varchar(50)   not null unique,
    datacadastro   timestamp   not null,
    primary key(idpessoa));

create table endereco(
    idendereco    integer      auto_increment,
    logradouro    varchar(150)  not null,
    cidade        varchar(50)   not null,
    estado         varchar(50)   not null,
    idpessoa      integer      not null unique,
    primary key(idendereco));

alter table endereco add constraint fk_endereco_pessoa
foreign key(idpessoa) references pessoa(idpessoa)
on delete cascade;

show tables;
desc pessoa;
desc endereco;
```



## Criando o JavaBeans

```
package entity;

import java.util.Date;

public class Pessoa {

    private Integer idPessoa;
    private String nome;
    private String email;
    private Date dataCadastro;

    private Endereco endereço; // TER 1 (Associação)

    public Pessoa() {
        // Construtor default
    }

    public Pessoa(Integer idPessoa, String nome, String email,
                  Date dataCadastro) {
        super();
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.email = email;
        this.dataCadastro = dataCadastro;
    }

    @Override
    public String toString() {
        return "Pessoa [idPessoa=" + idPessoa + ", nome="
               + nome + ", email=" + email + ", dataCadastro="
               + dataCadastro + "]";
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```



```
public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Date getDataCadastro() {
    return dataCadastro;
}

public void setDataCadastro(Date dataCadastro) {
    this.dataCadastro = dataCadastro;
}

public Endereco getEndereco() {
    return endereco;
}

public void setEndereco(Endereco endereco) {
    this.endereco = endereco;
}
}
```

---

```
package entity;

public class Endereco {

    private Integer idEndereco;
    private String logradouro;
    private String cidade;
    private String estado;

    private Pessoa pessoa;
    // Relacionamento TER-1 (Associação)

    public Endereco() {
        // Construtor default
    }

    public Endereco(Integer idEndereco, String logradouro,
                    String cidade, String estado) {
        super();
        this.idEndereco = idEndereco;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
    }
}
```



```
@Override
public String toString() {
    return "Endereco [idEndereco=" + idEndereco
           + ", logradouro=" + logradouro + ", cidade=" + cidade
           + ", estado=" + estado + "]";
}

public Integer getIdEndereco() {
    return idEndereco;
}

public void setIdEndereco(Integer idEndereco) {
    this.idEndereco = idEndereco;
}

public String getLogradouro() {
    return logradouro;
}

public void setLogradouro(String logradouro) {
    this.logradouro = logradouro;
}

public String getCidade() {
    return cidade;
}

public void setCidade(String cidade) {
    this.cidade = cidade;
}

public String getEstado() {
    return estado;
}

public void setEstado(String estado) {
    this.estado = estado;
}

public Pessoa getPessoa() {
    return pessoa;
}

public void setPessoa(Pessoa pessoa) {
    this.pessoa = pessoa;
}

}
```



## JPA - Java Persistence API

Biblioteca de anotações do Java para mapeamento objeto / relacional

```
package entity;

import java.util.Date;

import javafx.scene.text.TextBoundsType;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "pessoa")
public class Pessoa {

    @Id //primary key
    @GeneratedValue(strategy=GenerationType.AUTO) //auto_increment
    @Column(name="idpessoa")
    private Integer idPessoa;

    @Column(name="nome", length=50, nullable=false)
    private String nome;

    @Column(name="email", length=50, nullable=false, unique=true)
    private String email;

    @Temporal(TemporalType.TIMESTAMP) //data e hora
    @Column(name="datacadastro", nullable=false)
    private Date dataCadastro;

    @OneToOne(mappedBy="pessoa", cascade = CascadeType.REMOVE)
    //nome do atributo onde foi mapeado a @JoinColumn
    private Endereco endereco; // TER 1 (Associação)

    public Pessoa() {
        // Construtor default
    }

    public Pessoa(Integer idPessoa, String nome, String email,
                  Date dataCadastro) {
        this.idPessoa = idPessoa;
        this.nome = nome;
```



```
        this.email = email;
        this.dataCadastro = dataCadastro;
    }

@Override
public String toString() {
    return "Pessoa [idPessoa=" + idPessoa + ", nome=" + nome
           + ", email=" + email + ", dataCadastro="
           + dataCadastro + "]";
}

public Integer getIdPessoa() {
    return idPessoa;
}

public void setIdPessoa(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Date getDataCadastro() {
    return dataCadastro;
}

public void setDataCadastro(Date dataCadastro) {
    this.dataCadastro = dataCadastro;
}

public Endereco getEndereco() {
    return endereco;
}

public void setEndereco(Endereco endereco) {
    this.endereco = endereco;
}
}
```



```
package entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "endereco")
public class Endereco {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idendereco")
    private Integer idEndereco;

    @Column(name = "logradouro", length = 150, nullable = false)
    private String logradouro;

    @Column(name = "cidade", length = 50, nullable = false)
    private String cidade;

    @Column(name = "estado", length = 50, nullable = false)
    private String estado;

    @OneToOne
    @JoinColumn(name = "idpessoa", nullable = false,
    unique = true) //Foreign Key
    private Pessoa pessoa; // Relacionamento TER-1 (Associação)

    public Endereco() {
        // Construtor default
    }

    public Endereco(Integer idEndereco, String logradouro,
                    String cidade, String estado) {
        this.idEndereco = idEndereco;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
    }

    @Override
    public String toString() {
        return "Endereco [idEndereco=" + idEndereco
            + ", logradouro=" + logradouro + ", cidade="
            + cidade + ", estado=" + estado + "]";
    }
}
```



```
public Integer getIdEndereco() {
    return idEndereco;
}

public void setIdEndereco(Integer idEndereco) {
    this.idEndereco = idEndereco;
}

public String getLogradouro() {
    return logradouro;
}

public void setLogradouro(String logradouro) {
    this.logradouro = logradouro;
}

public String getCidade() {
    return cidade;
}

public void setCidade(String cidade) {
    this.cidade = cidade;
}

public String getEstado() {
    return estado;
}

public void setEstado(String estado) {
    this.estado = estado;
}

public Pessoa getPessoa() {
    return pessoa;
}

public void setPessoa(Pessoa pessoa) {
    this.pessoa = pessoa;
}
}
```

## HQL - Hibernate Query Language

Linguagem utilizada pelo framework hibernate para escrita de consultas em aplicações Java. Uma consulta escrita em HQL é interpretada e 'traduzida' para o dialeto SQL do banco de dados em que estamos trabalhando.



## @NamedQueries

Annotation utilizada para mapeamento das consultas escritas no projeto para uma determinada Classe.

```
package entity;

import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "pessoa")
@NamedQueries(
{
    @NamedQuery(name="pessoa.listartodos",
        query="select p from Pessoa as p inner join
        p.endereco as e"),

    @NamedQuery(name="pessoa.listarpornome",
        query="select p from Pessoa as p inner join
        p.endereco as e where p.nome like :p1"),
}
)
public class Pessoa {

    @Id //primary key
    @GeneratedValue(strategy=GenerationType.AUTO) //auto_increment
    @Column(name="idpessoa")
    private Integer idPessoa;

    @Column(name="nome", length=50, nullable=false)
    private String nome;

    @Column(name="email", length=50, nullable=false, unique=true)
    private String email;

    @Temporal(TemporalType.TIMESTAMP) //data e hora
    @Column(name="datacadastro")
    private Date dataCadastro;
```



```
@OneToOne(mappedBy="pessoa", cascade = CascadeType.REMOVE)
// nome do atributo onde foi mapeado a @JoinColumn
private Endereco endereco; // TER 1 (Associação)

public Pessoa() {
    // Construtor default
}

public Pessoa(Integer idPessoa, String nome, String email,
              Date dataCadastro) {
    super();
    this.idPessoa = idPessoa;
    this.nome = nome;
    this.email = email;
    this.dataCadastro = dataCadastro;
}

@Override
public String toString() {
    return "Pessoa [idPessoa=" + idPessoa + ", nome=" + nome
           + ", email=" + email + ", dataCadastro="
           + dataCadastro + "]";
}

public Integer getIdPessoa() {
    return idPessoa;
}

public void setIdPessoa(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Date getDataCadastro() {
    return dataCadastro;
}
```



```
public void setDataCadastro(Date dataCadastro) {
    this.dataCadastro = dataCadastro;
}

public Endereco getEndereco() {
    return endereco;
}

public void setEndereco(Endereco endereco) {
    this.endereco = endereco;
}
}
```

---

### mysql\_hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>

<session-factory>

    <property name="hibernate.dialect">
        org.hibernate.dialect.MySQLDialect
    </property>

    <property name="hibernate.connection.driver_class">
        com.mysql.jdbc.Driver
    </property>

    <property name="hibernate.connection.url">
        jdbc:mysql://localhost:3306/aulaweb11
    </property>

    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">coti</property>

    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.format_sql">true</property>

    <mapping class="entity.Pessoa"/>
    <mapping class="entity.Endereco"/>

</session-factory>

</hibernate-configuration>
```



## HibernateUtil

Classe para gerar as conexões no BD através da Factory do Hibernate

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {

            sessionFactory = new AnnotationConfiguration().
                configure("config/mysql_hibernate.cfg.xml").
                buildSessionFactory();

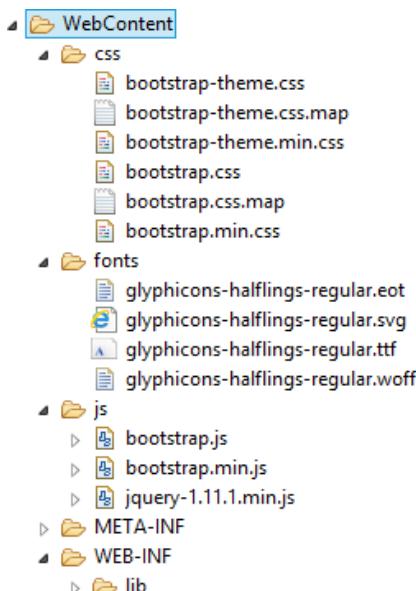
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory
                creation failed." + ex);

            throw new ExceptionInInitializerError(ex);
        }
    }

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

Incluindo o bootstrap no projeto...

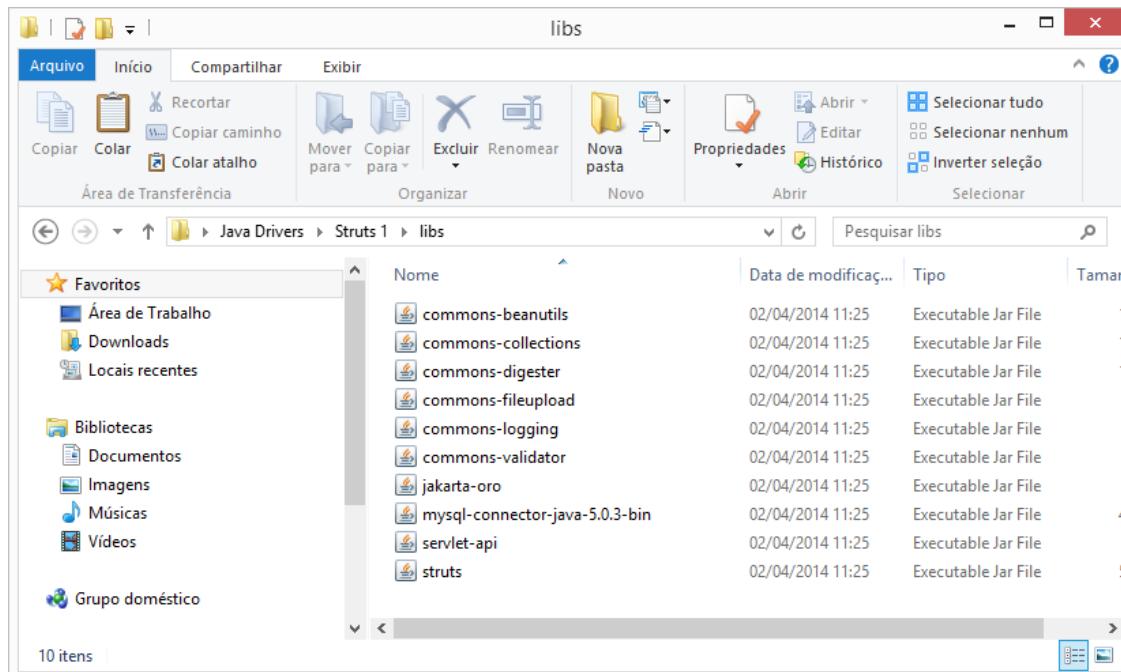




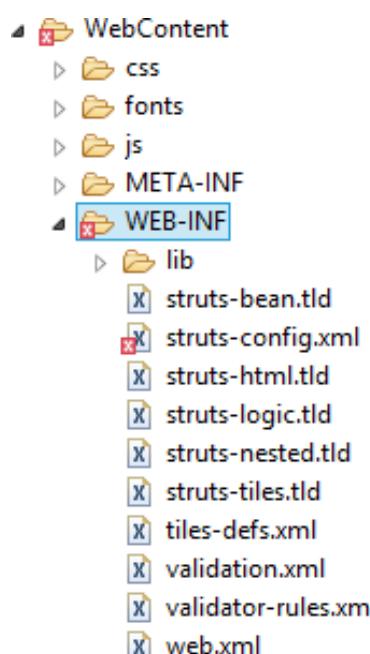
## Jakarta Struts

Framework da comunidade Java (tomcat) voltado para o desenvolvimento de aplicações JavaWeb baseado no padrão **MVC**.

- Incluir as bibliotecas do struts no projeto...  
**\WEB-INF\lib**

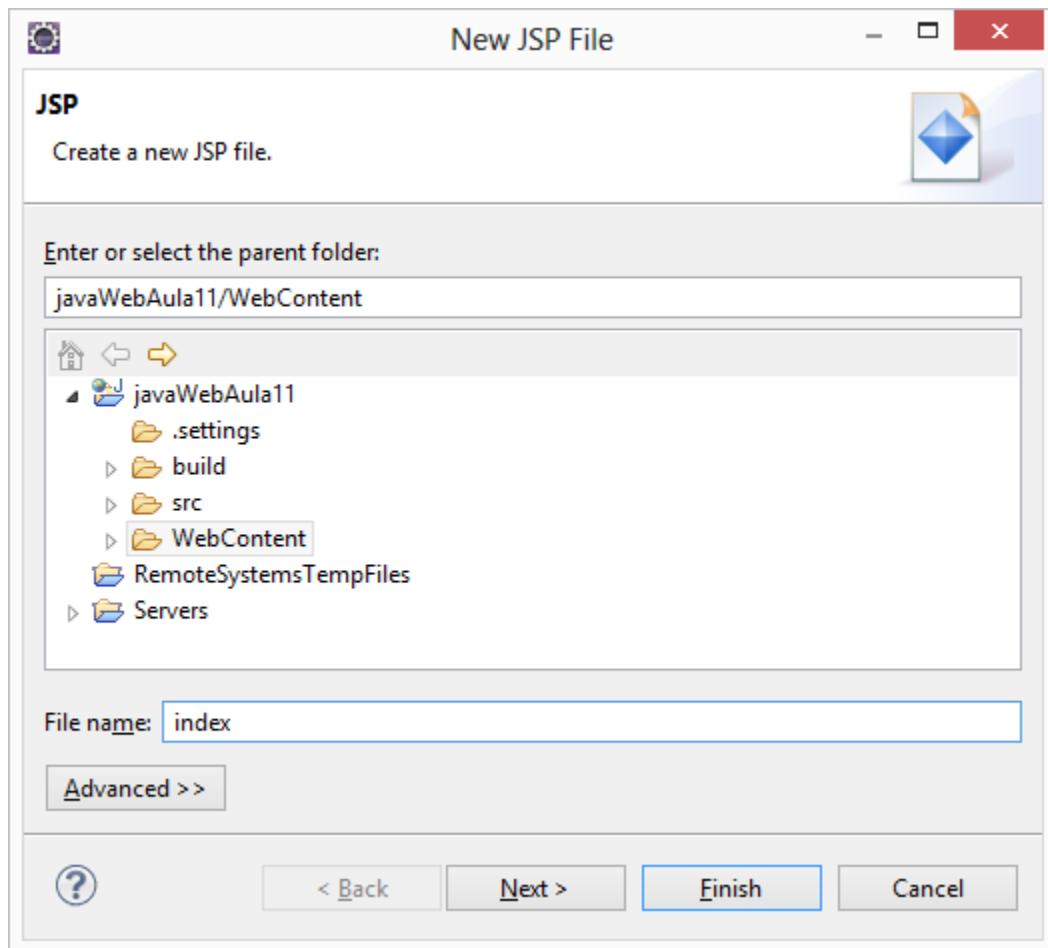


- Arquivos de configuração do Struts (.xml)  
**\WEB-INF**





### Criando a página inicial do projeto...



### \WebContent\index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>

<html>
  <head>
    <link rel="stylesheet" href="css/bootstrap.css"
          type="text/css"/>
    <script type="text/javascript"
           src="js/jquery-1.11.1.min.js"></script>
    <script type="text/javascript"
           src="js/bootstrap.js"></script>
  </head>
```



```
<body>

    <div class="well">

        <h3>Projeto Controle de Pessoas</h3>
        <hr/>

        <ul>
            <li> <a href="cadastro.jsp">
                Cadastrar Pessoas
            </a>
            </li>

            <li> <a href="consulta.jsp">
                Consultar Pessoas
            </a>
            </li>
        </ul>

    </div>

</body>

</html>
```

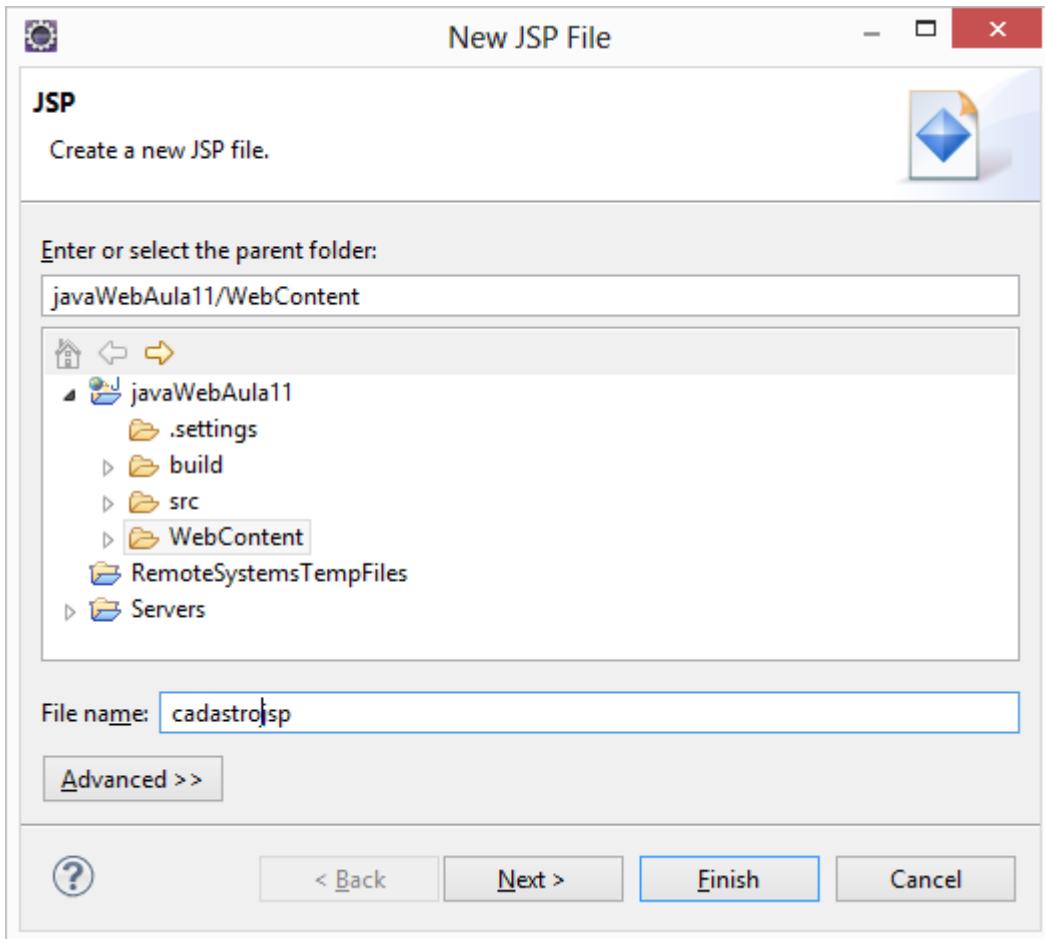
## Executando...

localhost:8081/javaWebAula11/

Projeto Controle de Pessoas

- Cadastrar Pessoas
- Consultar Pessoas





## O Struts possui 3 principais taglibs

Bibliotecas de Tags do Java

- **HTML** Componentes de formulario ou página
- **BEAN** Exibir mensagens ou dados
- **LOGIC** Funções lógicas (for, if, etc...)

### /WebContent/cadastro.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>

<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>

<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
<html>

    <head>

        <link rel="stylesheet" href="css/bootstrap.css"
              type="text/css"/>

        <script type="text/javascript"
               src="js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
               src="js/bootstrap.js"></script>
    </head>

    <body>
        <div class="well">

            <h3>Cadastro de Pessoas</h3>
            <a href="index.jsp">Voltar</a>
            para a página inicial.
            <hr/>

            <html:form method="post" action="">

                Nome da Pessoa: <br/>
                <html:text property="" />
                <br/><br/>

                Email: <br/>
                <html:text property="" />
                <br/><br/>

                Logradouro: <br/>
                <html:text property="" />
                <br/><br/>

                Cidade: <br/>
                <html:text property="" />
                <br/><br/>

                Estado: <br/>
                <html:text property="" />
                <br/><br/>

                <html:submit value="Cadastrar Pessoa"/>

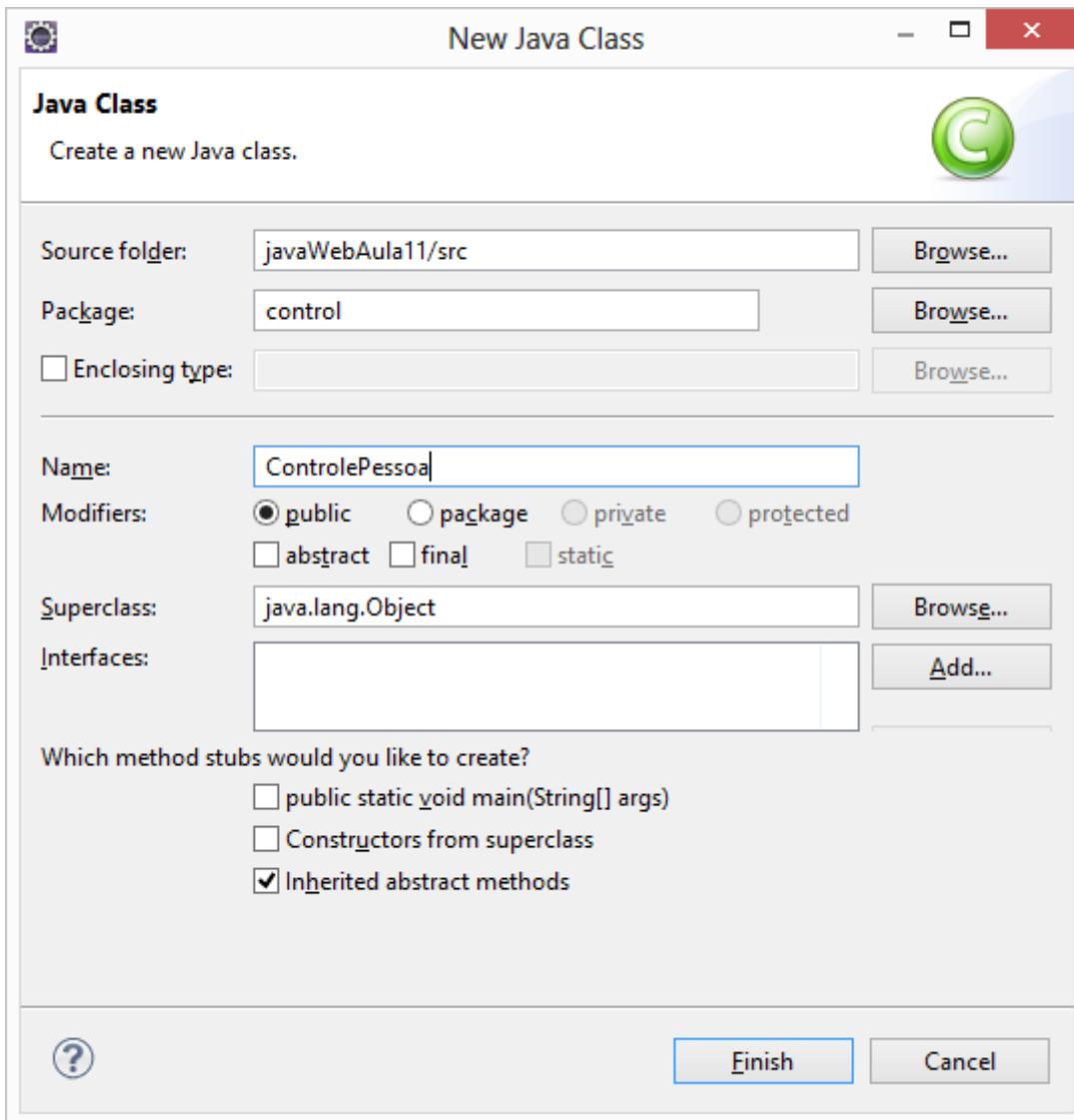
            </html:form>

        </div>
    </body>
</html>
```



## Classes Action

Tipo de Classe no Struts utilizado para definir ações e controles tal qual um Servlet em JavaWeb padrão. Uma Classe Action é a camada de controle em um projeto Struts.



Toda Classe definida como Controle no Struts deverá herdar **Action**

```
package control;

import org.apache.struts.action.Action;

public class ControlePessoa extends Action{}
```



Toda Classe Action do Struts deve estar mapeada no principal arquivo de configuração do struts, denominado **struts-config.xml**

## \WEB-INF\struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

    <!-- Mapear as Classes de Controle -->
    <action-mappings>

        <!-- Mapeamento do ControlePessoa -->
        <!-- path  => Nome utilizado para acionar este controle -->
        <!-- scope => Armazenamento os dados recebidos pelo Controle -->
        <!-- type  => Caminho da Classe de Controle -->
        <action path="/ControlePessoa" scope="request"
                type="control.ControlePessoa">
            </action>

    </action-mappings>

    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
                      value="/WEB-INF/tiles-defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames"
                      value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
    </plug-in>
</struts-config>
```

O Struts é configurado para executar suas classes de Ação (Actions) utilizando a extensão **.do**

## \WEB-INF\web.xml

```
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>

    <head>

        <link rel="stylesheet" href="css/bootstrap.css"
            type="text/css"/>

        <script type="text/javascript"
            src="js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
            src="js/bootstrap.js"></script>

    </head>

    <body>

        <div class="well">

            <h3>Cadastro de Pessoas</h3>
            <a href="index.jsp">Voltar</a>
            para a página inicial.
            <hr/>

            <html:form method="post"
                action="ControlePessoa.do">

                Nome da Pessoa: <br/>
                <html:text property="" />
                <br/><br/>

                Email: <br/>
                <html:text property="" />
                <br/><br/>

                Logradouro: <br/>
                <html:text property="" />
                <br/><br/>

                Cidade: <br/>
                <html:text property="" />
                <br/><br/>

            </html:form>
        </div>
    </body>
</html>
```



```
Estado: <br/>
<html:text property="" />
<br/><br/>

<html:submit value="Cadastrar Pessoa" />

</html:form>

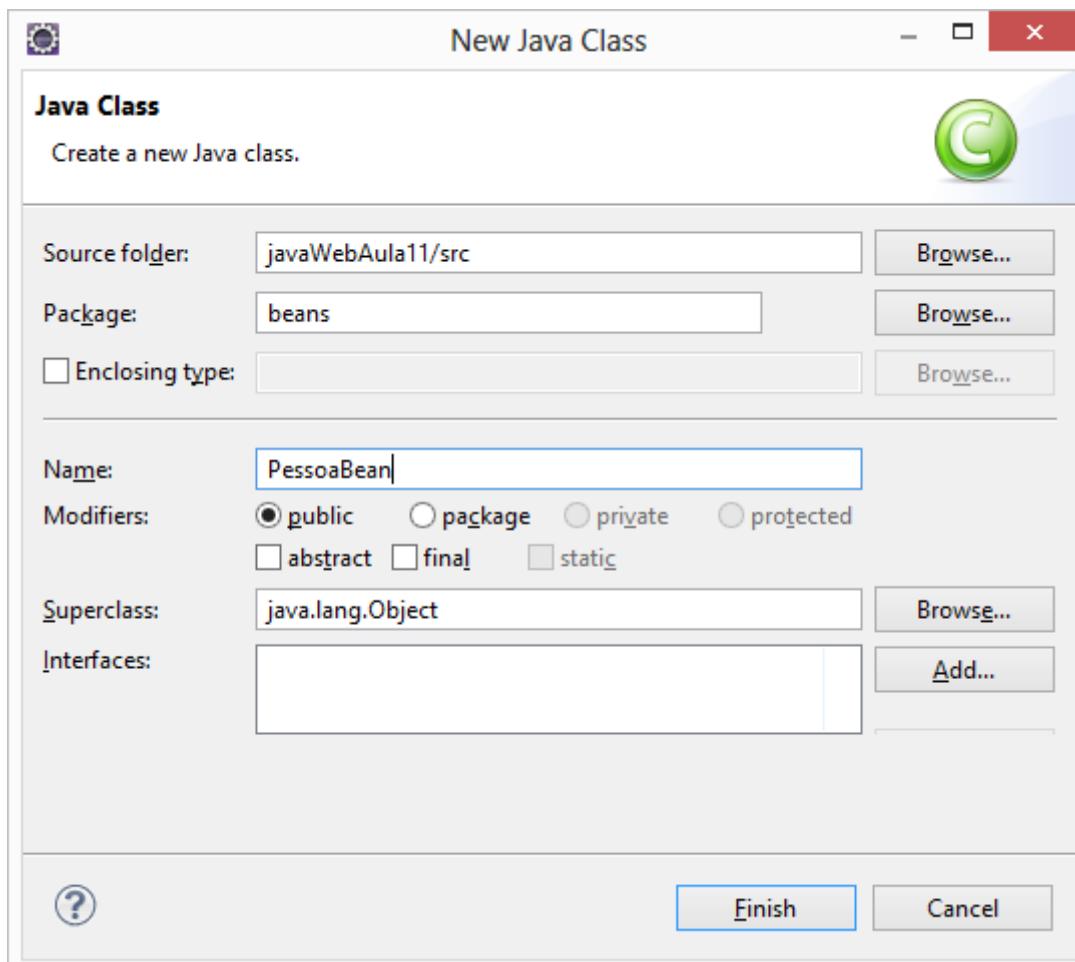
</div>

</body>

</html>
```

## Camada de Modelo do Struts

Parte do projeto Struts responsável pelos dados de entrada ou saída do projeto. Neste projeto, criaremos uma Classe **PessoaBean** que será utilizada para armazenar e validar os dados enviados pelo formulário de cadastro de Pessoa.





Toda Classe no Struts que representa uma **camada de modelo** (entrada ou saída de dados) deverá herdar de ActionForm

```
package beans;

import org.apache.struts.action.ActionForm;

import entity.Endereco;
import entity.Pessoa;

public class PessoaBean extends ActionForm{

    // Atributo para capturar os dados do formulário
    private Pessoa pessoa; //entrada de dados do formulário

    public PessoaBean() {
        pessoa = new Pessoa();
        //inicializando (espaço de memória)
        pessoa.setEndereco(new Endereco());
        //inicializando o atributo endereço
    }

    public Pessoa getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoa pessoa) {
        this.pessoa = pessoa;
    }

}
```

O ActionForm deverá ser mapeado no struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

    <!-- Mapeamento das Classes Bean (Modelo) -->
    <form-beans>

        <form-bean name="PessoaBean"
                    type="beans.PessoaBean" />

    </form-beans>
```



# **Java WebDeveloper - BRQ**

**Terça-feira, 10 de Junho de 2014**

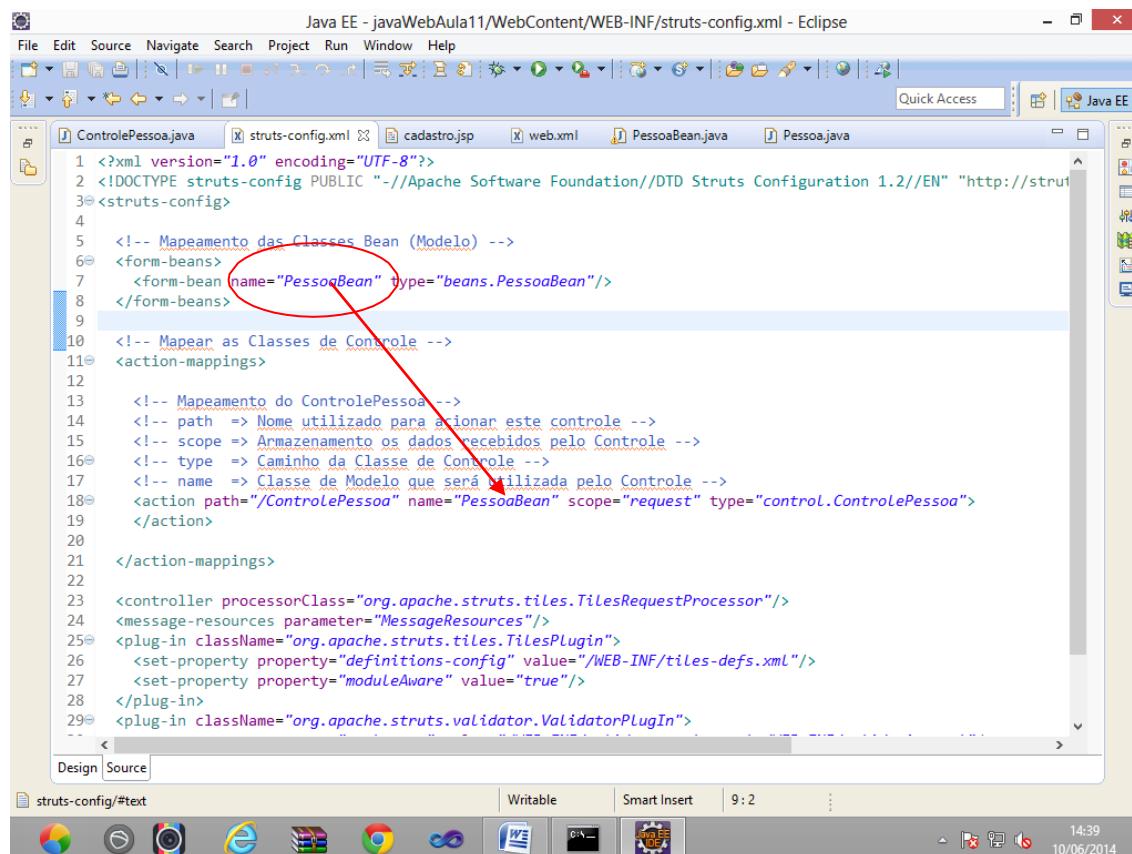
## Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

# Aula 21

```
<!-- Mapear as Classes de Controle -->
<action-mappings>

    <!-- Mapeamento do ControlePessoa -->
    <!-- path  => Nome utilizado para acionar este controle -->
    <!-- scope => Armazenamento os dados recebidos pelo Controle -->
    <!-- type   => Caminho da Classe de Controle -->
    <!-- name   => Classe de Modelo utilizada pelo Controle -->
    <action path="/ControlePessoa" name="PessoaBean"
            scope="request" type="control.ControlePessoa">
        </action>

    </action-mappings>
    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
                     value="/WEB-INF/tiles-defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames"
                     value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
    </plug-in>
</struts-config>
```





### Apontando os campos do formulário para armazenamento no atributo '**pessoa**' da Classe PessoaBean

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-Logic"
    prefix="logic" %>

<html>

    <head>

        <link rel="stylesheet" href="css/bootstrap.css"
            type="text/css"/>

        <script type="text/javascript"
            src="js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
            src="js/bootstrap.js"></script>

    </head>

    <body>

        <div class="well">

            <h3>Cadastro de Pessoas</h3>
            <a href="index.jsp">Voltar</a>
            para a página inicial.
            <hr/>

            <html:form method="post" action="ControlePessoa.do">

                Nome da Pessoa: <br/>
                <html:text property="pessoa.nome"/>
                <br/><br/>

                Email: <br/>
                <html:text property="pessoa.email"/>
                <br/><br/>

                Logradouro: <br/>
                <html:text property="pessoa.
                    endereço.Logradouro"/>
                <br/><br/>

            </html:form>

        </div>

    </body>

</html>
```



```
Cidade: <br/>
<html:text property="pessoa.
endereco.cidade"/>
<br/><br/>

Estado: <br/>
<html:text property="pessoa.
endereco.estado"/>
<br/><br/>

<html:submit value="Cadastrar Pessoa"/>

</html:form>

</div>

</body>

</html>
```

A Classe de Controle do Struts não utiliza doGet ou doPost, substituindo-os por um unico método chamado **execute**

```
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class ControlePessoa extends Action{

    @Override
    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {

        return super.execute(mapping, form, request, response);
    }
}
```



### Executando...

Cadastro de Pessoas

Voltar para a página inicial.

Nome da Pessoa:

Email:

Logradouro:

Cidade:

Estado:

Cadastrar Pessoa

### Estilizando o formulário com bootstrap...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>

<html>

    <head>

        <link rel="stylesheet" href="css/bootstrap.css"
        type="text/css"/>

        <script type="text/javascript"
        src="js/jquery-1.11.1.min.js"></script>
        <script type="text/javascript"
        src="js/bootstrap.js"></script>
```



```
</head>

<body>

    <div class="well">

        <h3>Cadastro de Pessoas</h3>
        <a href="index.jsp">Voltar</a>
        para a página inicial.
        <hr/>

        <html:form method="post" action="ControlePessoa.do"
                   style="width: 400px;">

            Nome da Pessoa: <br/>
            <html:text property="pessoa.nome"
                      styleClass="form-control"/>
            <br/>

            Email: <br/>
            <html:text property="pessoa.email"
                      styleClass="form-control"/>
            <br/>

            Logradouro: <br/>
            <html:text property="pessoa.endereco.
                        Logradouro" styleClass="form-control"/>
            <br/>

            Cidade: <br/>
            <html:text property="pessoa.endereco.cidade"
                      styleClass="form-control"/>
            <br/>

            Estado: <br/>
            <html:text property="pessoa.endereco.estado"
                      styleClass="form-control"/>
            <br/>

            <html:submit value="Cadastrar Pessoa"
                         styleClass="btn btn-success"/>

        </html:form>

    </div>

</body>

</html>
```



### Executando...

Cadastro de Pessoas

Voltar para a página inicial.

Nome da Pessoa:

Email:

Logradouro:

Cidade:

Estado:

Cadastrar Pessoa



```
package persistence;

import hibernate.HibernateUtil;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Pessoa;

public class PessoaDao {

    private Session session;
    private Transaction transaction;
    private Query query;

    //Método para gravar os dados de Pessoa
    public void create(Pessoa p) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.save(p);
        transaction.commit();
        session.close();
    }

}
```



Realizando a Action no formulário do Struts...

```
<html:form method="post"
           action="ControlePessoa.do?cmd=cadastrar">

package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import persistence.PessoaDao;
import beans.PessoaBean;

public class ControlePessoa extends Action{

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        //resgatando o cmd
        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form, request, response);
        }

        return super.execute(mapping, form, request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
                                   ActionForm form, HttpServletRequest request,
                                   HttpServletResponse response)
        throws Exception {

        try{
            PessoaBean pb = (PessoaBean) form;

            PessoaDao d = new PessoaDao();
            d.create(pb.getPessoa());

            request.setAttribute("msg", "Dados gravados
                                  com sucesso.");
        }
        catch(Exception e){
            request.setAttribute("msg", "Erro -> "
                               + e.getMessage());
        }
    }
}
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
//Nome de um redirecionamento
return mapping.findForward("cadastro_sucesso"); //flag

}

}

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

    <!-- Mapeamento das Classes Bean (Modelo) -->
    <form-beans>
        <form-bean name="PessoaBean" type="beans.PessoaBean"/>
    </form-beans>

    <!-- Mapear as Classes de Controle -->
    <action-mappings>

        <!-- Mapeamento do ControlePessoa -->
        <!-- path => Nome utilizado para acionar este controle -->
        <!-- scope => Armazenamento os dados recebidos pelo Controle -->
        <!-- type => Caminho da Classe de Controle -->
        <!-- name => Classe de Modelo utilizada pelo Controle -->
        <action path="/ControlePessoa" name="PessoaBean" scope="request"
               type="control.ControlePessoa">

            <!-- mapeamento do redirecionamento -->
            <forward path="/cadastro.jsp"
                     name="cadastro_sucesso"/>

        </action>

    </action-mappings>

    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config" value="/WEB-INF/tiles-
defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames"
                     value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
    </plug-in>
</struts-config>
```



Exibindo a mensagem

```
<p>
    <bean:write name="msg" ignore="true"/>
</p>
```

Gravando Endereco...

```
package persistence;

import hibernate.HibernateUtil;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Endereco;

public class EnderecoDao {

    private Session session;
    private Transaction transaction;
    private Query query;

    public void create(Endereco e) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.save(e);
        transaction.commit();
        session.close();
    }
}
```

-----

```
package beans;

import org.apache.struts.action.ActionForm;

import entity.Endereco;
import entity.Pessoa;

public class PessoaBean extends ActionForm{

    // Atributo para capturar os dados do formulario
    private Pessoa pessoa; //entrada de dados do formulário
    private Endereco endereco; //entrada de dados do formulário
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
public PessoaBean() {
    pessoa = new Pessoa(); //inicializando (espaço de memória)
    endereco = new Endereco(); //inicializando (espaço de memória)
}

public Pessoa getPessoa() {
    return pessoa;
}

public void setPessoa(Pessoa pessoa) {
    this.pessoa = pessoa;
}

public Endereco getEndereco() {
    return endereco;
}

public void setEndereco(Endereco endereco) {
    this.endereco = endereco;
}
}

-----
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>

<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>

<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>

    <head>

        <link rel="stylesheet" href="css/bootstrap.css"
        type="text/css"/>

        <script type="text/javascript"
        src="js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
        src="js/bootstrap.js"></script>

    </head>

    <body>
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
<div class="well">

    <h3>Cadastro de Pessoas</h3>
    <a href="index.jsp">Voltar</a>
    para a página inicial.
    <hr/>

    <html:form method="post"
        action="ControlePessoa.do?cmd=cadastrar"
        style="width: 400px;">

        Nome da Pessoa: <br/>
        <html:text property="pessoa.nome"
        styleClass="form-control"/>
        <br/>

        Email: <br/>
        <html:text property="pessoa.email"
        styleClass="form-control"/>
        <br/>

        Logradouro: <br/>
        <html:text property="endereco.Logradouro"
        styleClass="form-control"/>
        <br/>

        Cidade: <br/>
        <html:text property="endereco.cidade"
        styleClass="form-control"/>
        <br/>

        Estado: <br/>
        <html:text property="endereco.estado"
        styleClass="form-control"/>
        <br/>

        <html:submit value="Cadastrar Pessoa"
        styleClass="btn btn-success"/>

    </html:form>

    <p>
        <bean:write name="msg" ignore="true"/>
    </p>

</div>

</body>

</html>
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

localhost:8081/javaWebA1 localhost:8081/javaWebAula11/ControlePessoa.do?cmd=cadastrar

### Cadastro de Pessoas

[Voltar para a página inicial.](#)

Nome da Pessoa:

Email:

Logradouro:

Cidade:

Estado:

[Cadastrar Pessoa](#)

Dados gravados com sucesso.



No banco de dados

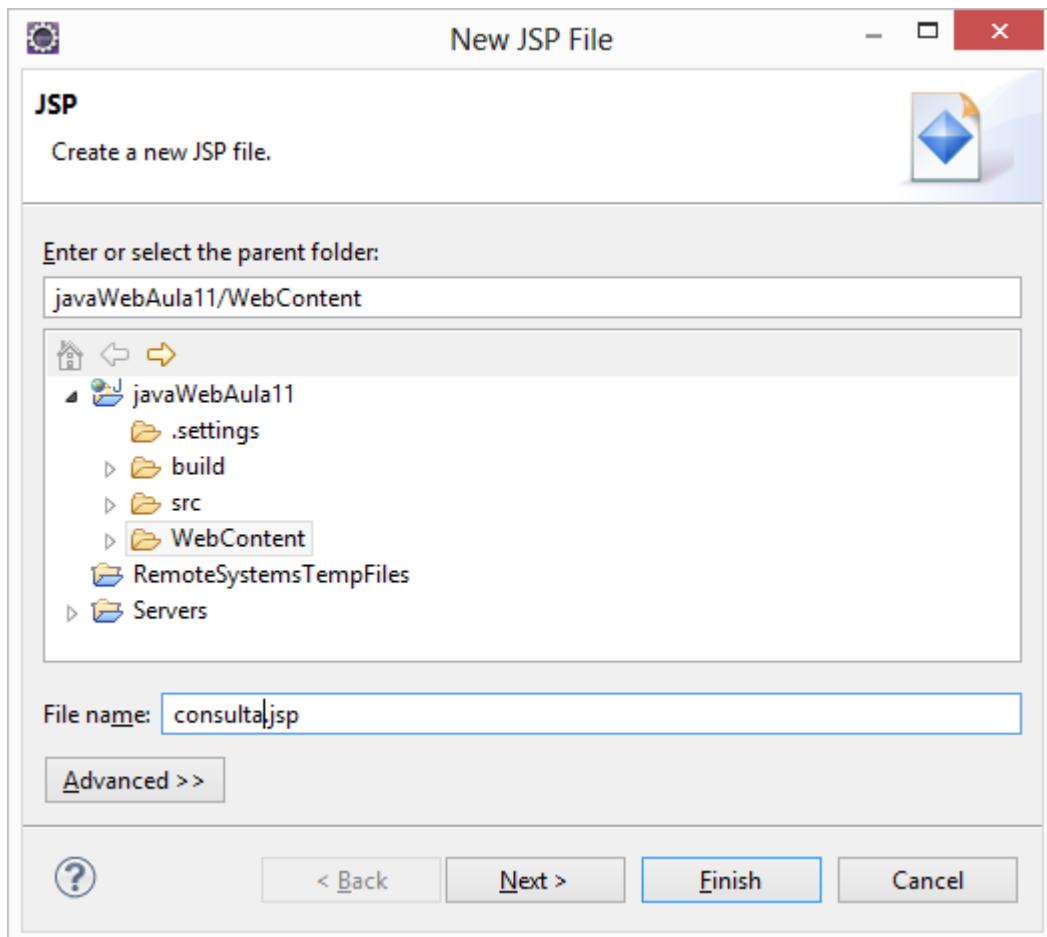
```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
mysql> select * from pessoa;
+-----+-----+-----+-----+
| idpessoa | nome      | email           | datacadastro |
+-----+-----+-----+-----+
|     1    | Sergio Mendes | sergio.coti@gmail.com | 2014-06-10 16:30:54 |
|     4    | Sergio Mendes | sergio.coti@yahoo.com | 2014-06-10 16:40:41 |
|     5    | Sergio Mendes | sergio.coti@hotmail.com | 2014-06-10 16:49:40 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from endereco;
+-----+-----+-----+-----+
| idendereco | logradouro | cidade        | estado | idpessoa |
+-----+-----+-----+-----+
|       1     | Rua A       | Rio de Janeiro | RJ     |      5    |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```



### Criando a página de consulta...



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>

<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>

<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>

<html>
    <head>

        <link rel="stylesheet" href="css/bootstrap.css"
              type="text/css"/>
        <script type="text/javascript"
               src="js/jquery-1.11.1.min.js"></script>
        <script type="text/javascript"
               src="js/bootstrap.js"></script>
    </head>
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
<body>

    <div class="well">

        <h3>Consulta de Pessoas</h3>
        <a href="index.jsp">Voltar</a>
        para a página inicial.
        <hr/>

        <html:form method="post"
                  action="ControlePessoa.do?cmd=consultar">

            Informe o nome da Pessoa:
            <html:text property="pessoa.nome"/>

            <html:submit value="Pesquisar Dados"
                         styleClass="btn btn-success"/>

        </html:form>

    </div>
</body>
</html>
```

localhost:8081/javaWebAula11/consulta.jsp

Consulta de Pessoas

[Voltar para a página inicial.](#)

Informe o nome da Pessoa:  Pesquisar Dados



17:09  
10/06/2014



```
package persistence;

import hibernate.HibernateUtil;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Pessoa;

public class PessoaDao {

    private Session session;
    private Transaction transaction;
    private Query query;

    //Método para gravar os dados de Pessoa
    public void create(Pessoa p) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.save(p);
        transaction.commit();
        session.close();
    }

    //Método para listar pessoas pelo nome
    public List<Pessoa> findAll(String nome) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();

        query = session.getNamedQuery("pessoa.listarpornome");
        query.setString("p1", nome + "%");
        List<Pessoa> lista = query.list();

        session.close();
        return lista;
    }
}

-----
package control;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import persistence.EnderecoDao;
import persistence.PessoaDao;
import beans.PessoaBean;
```



```
import entity.Pessoa;

public class ControlePessoa extends Action{

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        //resgatando o cmd
        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("consultar")){
            return consultar(mapping, form, request, response);
        }

        return super.execute(mapping, form, request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
                                   ActionForm form, HttpServletRequest request,
                                   HttpServletResponse response)
        throws Exception {

        try{
            PessoaBean pb = (PessoaBean) form;

            PessoaDao pd = new PessoaDao();
            EnderecoDao ed = new EnderecoDao();

            //Relacionar Endereco à Pessoa
            pb.getEndereco().setPessoa(pb.getPessoa());

            pd.create(pb.getPessoa()); //gravando pessoa
            ed.create(pb.getEndereco());
            //gravando endereço contido em pessoa

            request.setAttribute("msg", "Dados gravados
                                      com sucesso.");
        }
        catch(Exception e){
            request.setAttribute("msg", "Erro -> "
                                  + e.getMessage());
        }

        //Nome de um redirecionamento
        return mapping.findForward("cadastro_sucesso"); //flag
    }
}
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
public ActionForward consultar(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
    throws Exception {

    try{

        //Resgatar PessoaBean (Classe de modelo no Struts)
        PessoaBean pb = (PessoaBean) form;

        PessoaDao d = new PessoaDao();
        List<Pessoa> lista = d.findAll(pb.getPessoa().getNome());

        request.setAttribute("msg", "Quantidade
            de pessoas encontradas: " + lista.size());

        request.setAttribute("lista", lista);
        //enviando a lista de pessoas
    }
    catch(Exception e){
        request.setAttribute("msg", "Erro -> "
            + e.getMessage());
    }

    return mapping.findForward("consulta_sucesso"); //flag
}
}

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

<!-- Mapeamento das Classes Bean (Modelo) -->
<form-beans>
    <form-bean name="PessoaBean" type="beans.PessoaBean"/>
</form-beans>

<!-- Mapear as Classes de Controle -->
<action-mappings>

    <!-- Mapeamento do ControlePessoa -->
    <!-- path => Nome utilizado para acionar este controle -->
    <!-- scope => Armazenamento os dados recebidos pelo Controle -->
    <!-- type => Caminho da Classe de Controle -->
    <!-- name => Classe de Modelo que será utilizada pelo Controle -->
    <action path="/ControlePessoa" name="PessoaBean" scope="request"
        type="control.ControlePessoa">

        <!-- mapeamento do redirecionamento -->
        <forward path="/cadastro.jsp" name="cadastro_sucesso"/>
        <forward path="/consulta.jsp" name="consulta_sucesso"/>

```



```
</action>

</action-mappings>

<controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
<message-resources parameter="MessageResources"/>
<plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config"
        value="/WEB-INF/tiles-defs.xml"/>
    <set-property property="moduleAware" value="true"/>
</plug-in>
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
</struts-config>
```

### Na página de consulta...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>

<html>

    <head>

        <link rel="stylesheet" href="css/bootstrap.css"
            type="text/css"/>

        <script type="text/javascript"
            src="js/jquery-1.11.1.min.js"></script>

        <script type="text/javascript"
            src="js/bootstrap.js"></script>

    </head>

    <body>

        <div class="well">
```



```
<h3>Consulta de Pessoas</h3>
<a href="index.jsp">Voltar</a>
para a página inicial.
<hr/>

<html:form method="post"
action="ControlePessoa.do?cmd=consultar">

    Informe o nome da Pessoa:
    <html:text property="pessoa.nome"/>

    <html:submit value="Pesquisar Dados"
styleClass="btn btn-success"/>

</html:form>

<p>
    <bean:write name="msg" ignore="true"/>
</p>

<logic:present name="Lista">

    <table class="table table-condensed"
style="width: 100%">

        <thead>
            <tr>
                <th> Código </th>
                <th> Nome da Pessoa </th>
                <th> Email </th>
                <th> Data de Cadastro </th>
                <th> Logradouro </th>
                <th> Cidade </th>
                <th> Estado </th>
            </tr>
        </thead>

        <tbody>

            <logic:iterate name="Lista" id="p">

                <tr>
                    <td> <bean:write name="p"
property="idPessoa"/> </td>

                    <td> <bean:write name="p"
property="nome"/> </td>

                    <td> <bean:write name="p"
property="email"/> </td>
```



# Java WebDeveloper - BRQ

Terça-feira, 10 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
21

```
<td> <bean:write name="p"
property="dataCadastro"/> </td>

<td> <bean:write name="p"
property="endereco.Logradouro"/> </td>

<td> <bean:write name="p"
property="endereco.cidade"/> </td>

<td> <bean:write name="p"
property="endereco.estado"/> </td>

</tr>

</logic:iterate>
</tbody>
</table>
</logic:present>
</div>
</body>
</html>
```

## Executando...

The screenshot shows a web browser window with the URL `localhost:8082/javaWebAula11/ControlePessoa.do?cmd=consultar`. The page title is "Consulta de Pessoas". It contains a search form with a text input field containing "M" and a green "Pesquisar Dados" button. Below the form, a message says "Quantidade de pessoas encontradas: 2". A table displays two rows of data:

Código	Nome da Pessoa	Email	Data de Cadastro	Logradouro	Cidade	Estado
6	Marcone Freitas	marcone@gmail.com	2014-06-11 11:03:38.0	Rua A	São Paulo	SP
7	Maria Silva	maria@bol.com	2014-06-11 11:55:09.0	Rua B	São Paulo	SP



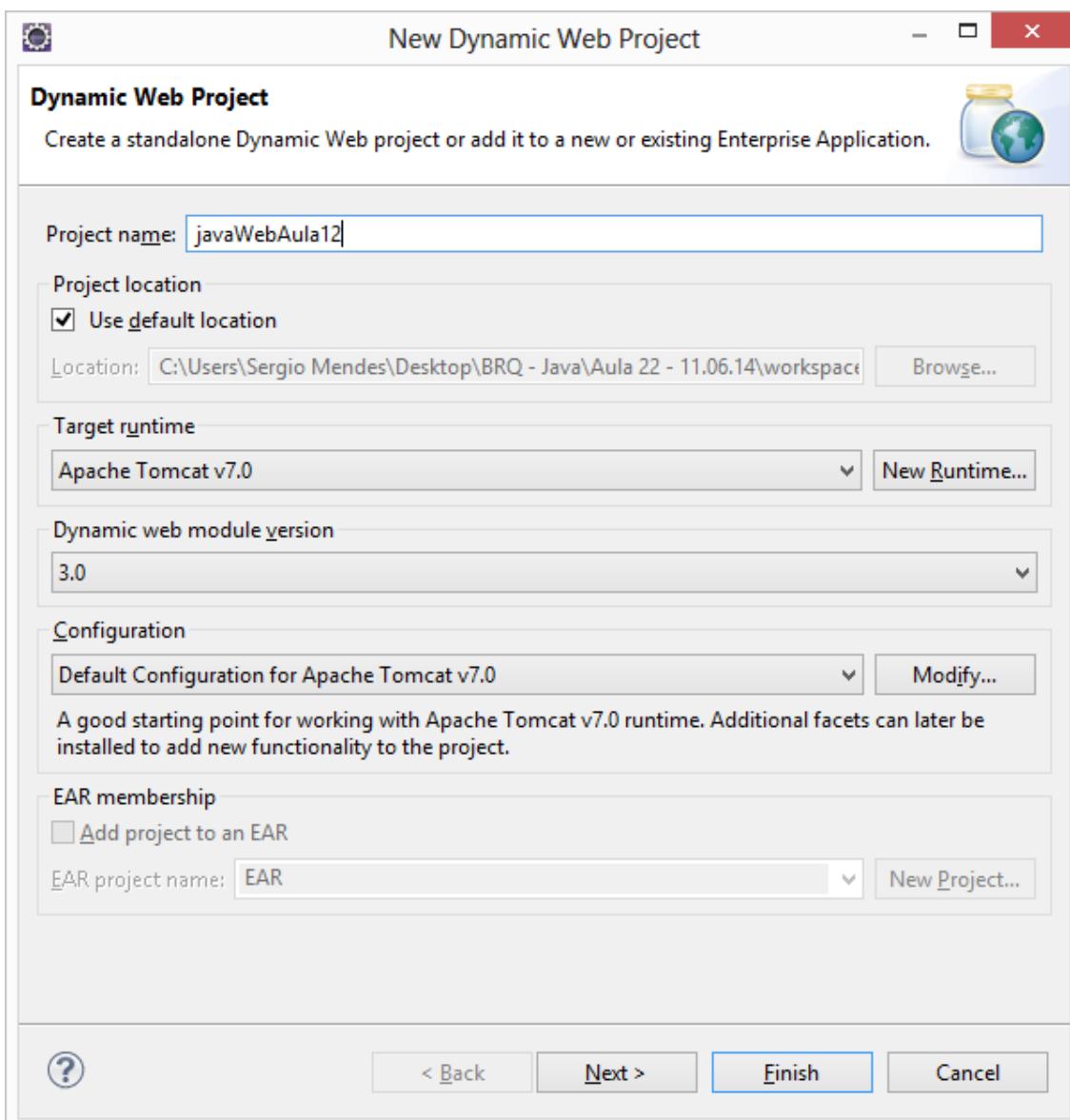
# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

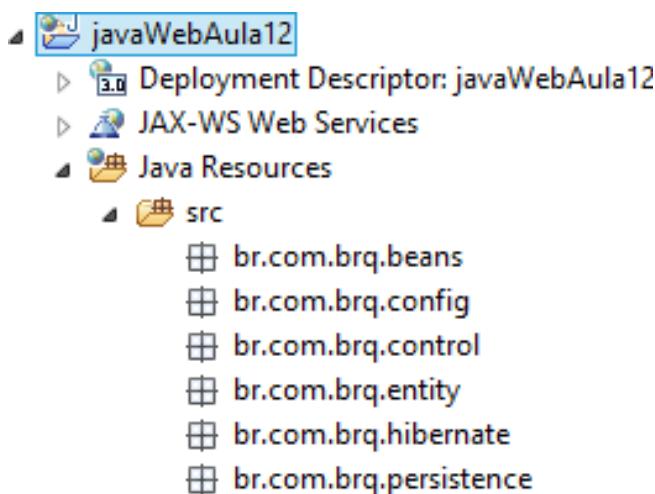
Aula

22

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

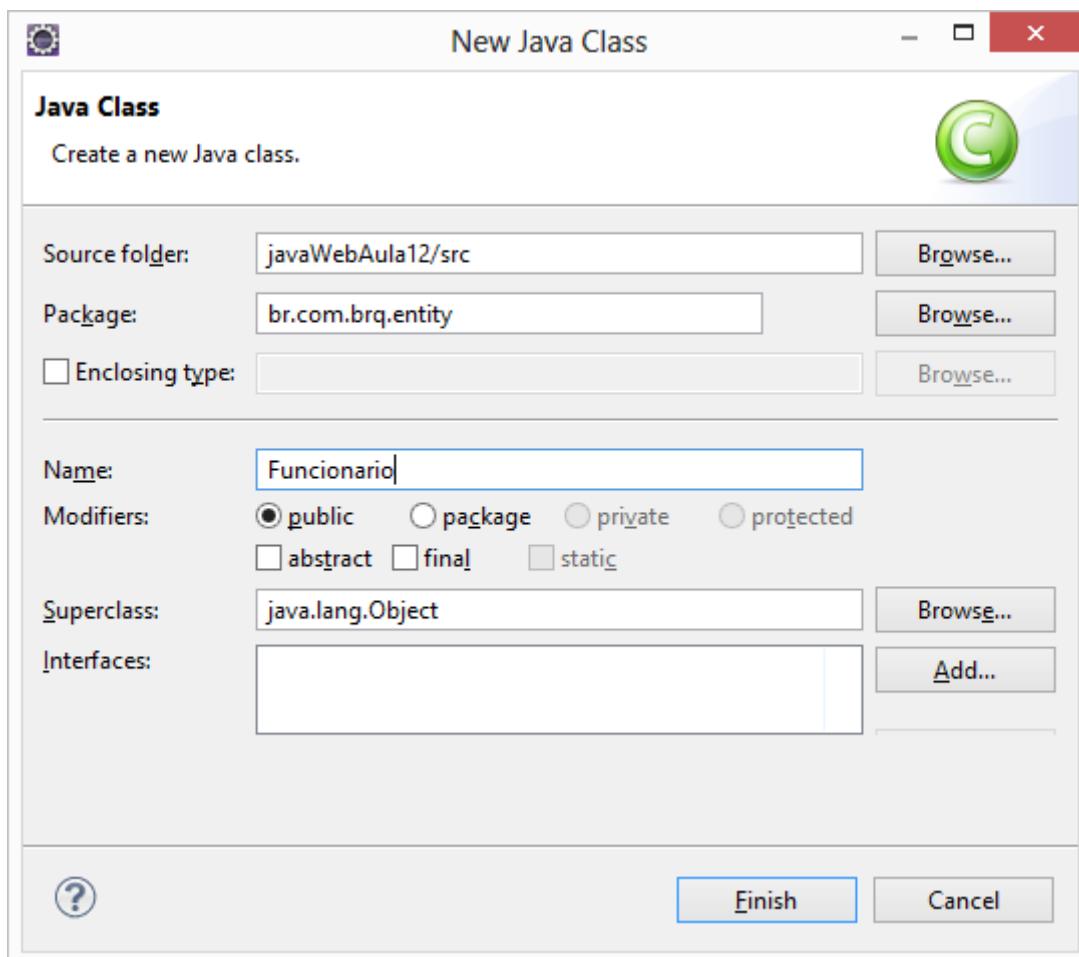


Estrutura de pacotes do projeto...





## JavaBean para Funcionário



```
package br.com.brq.entity;

import java.util.Date;

public class Funcionario {

    private Integer idFuncionario;
    private String nome;
    private Double salario;
    private String sexo;
    private String funcao;
    private String atribuicoes;
    private Date dataCadastro;

    public Funcionario() {
    }

    public Funcionario(Integer idFuncionario, String nome,
                       Double salario, String sexo, String funcao,
                       String atribuicoes, Date dataCadastro) {
```



```
super();
this.idFuncionario = idFuncionario;
this.nome = nome;
this.salario = salario;
this.sexo = sexo;
this.funcao = funcao;
this.atribuicoes = atribuicoes;
this.dataCadastro = dataCadastro;
}

@Override
public String toString() {
    return "Funcionario [idFuncionario=" + idFuncionario
        + ", nome=" + nome + ", salario=" + salario
        + ", sexo=" + sexo + ", funcao=" + funcao
        + ", atribuicoes=" + atribuicoes
        + ", dataCadastro=" + dataCadastro + "]";
}

public Integer getIdFuncionario() {
    return idFuncionario;
}

public void setIdFuncionario(Integer idFuncionario) {
    this.idFuncionario = idFuncionario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getSalario() {
    return salario;
}

public void setSalario(Double salario) {
    this.salario = salario;
}

public String getSexo() {
    return sexo;
}

public void setSexo(String sexo) {
    this.sexo = sexo;
}
```



```
public String getFuncao() {
    return funcao;
}

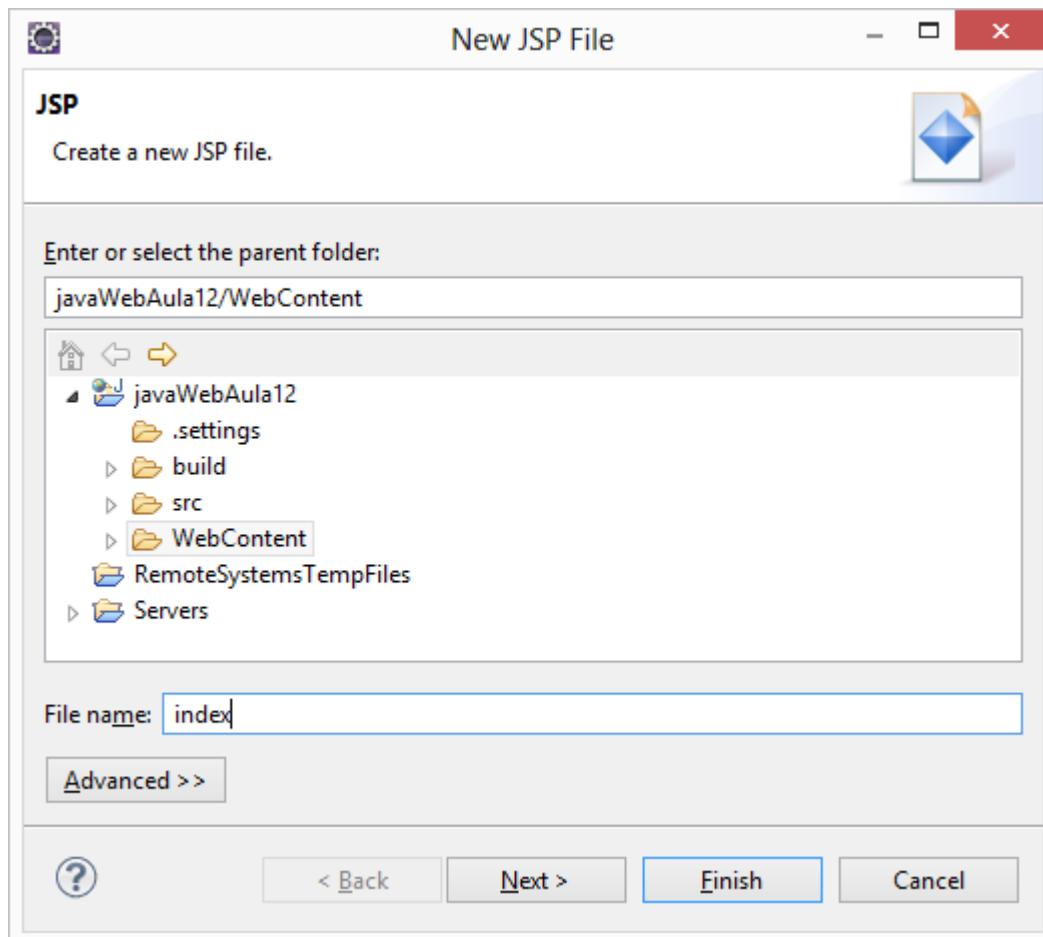
public void setFuncao(String funcao) {
    this.funcao = funcao;
}

public String getAtribuicoes() {
    return atribuicoes;
}

public void setAtribuicoes(String atribuicoes) {
    this.atribuicoes = atribuicoes;
}

public Date getDataCadastro() {
    return dataCadastro;
}

public void setDataCadastro(Date dataCadastro) {
    this.dataCadastro = dataCadastro;
}
}
```





# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

22

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>

        <style type="text/css">

            body { font-family: verdana;
                    font-size: 9pt;
                    padding: 30px; }

        </style>

    </head>

    <body>

        <h3>Projeto Controle de Funcionários</h3>
        <hr/>

        <a href="cadastro.jsp">Clique aqui</a>
           para cadastrar um novo funcionário

    </body>

</html>
```

Executando...





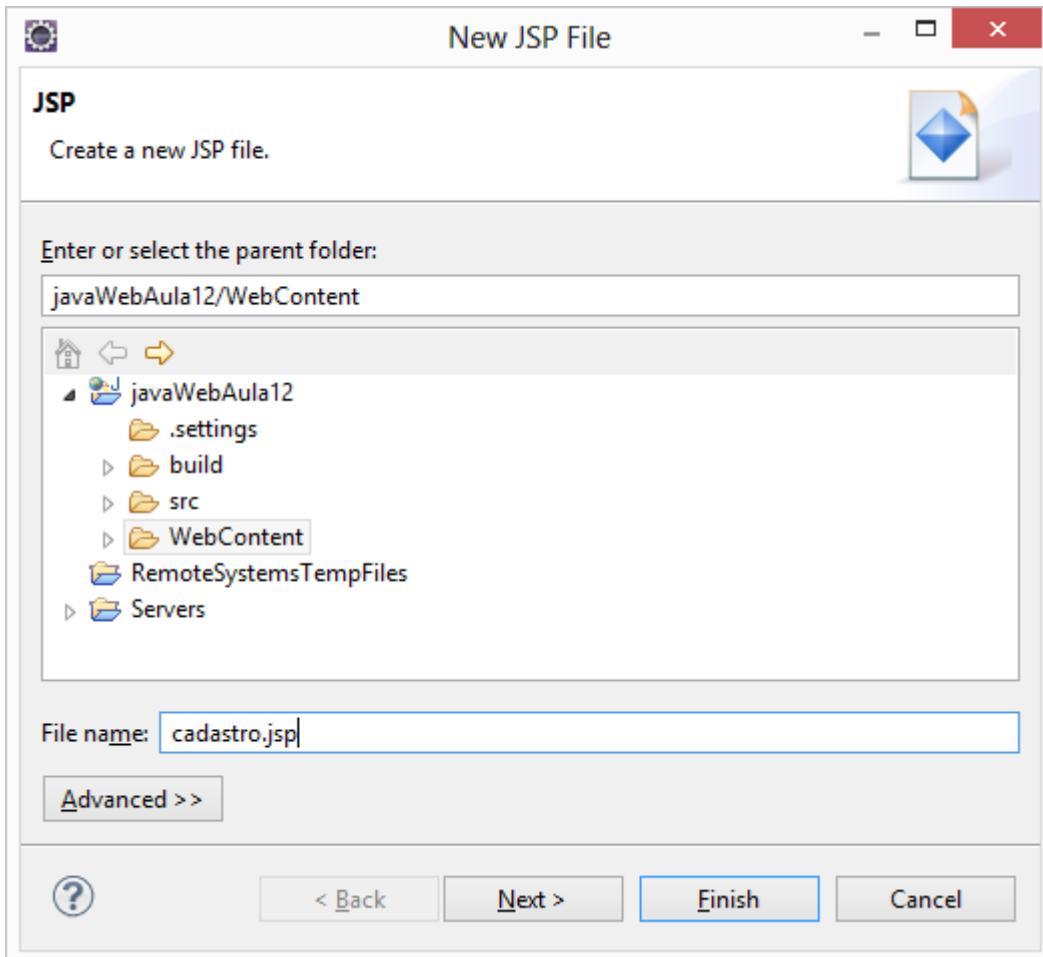
# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

22



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>

    <head>

        <style type="text/css">
            body { font-family: verdana; font-size: 9pt;
                    padding: 30px; }
        </style>

    </head>
```



# Java WebDeveloper - BRQ

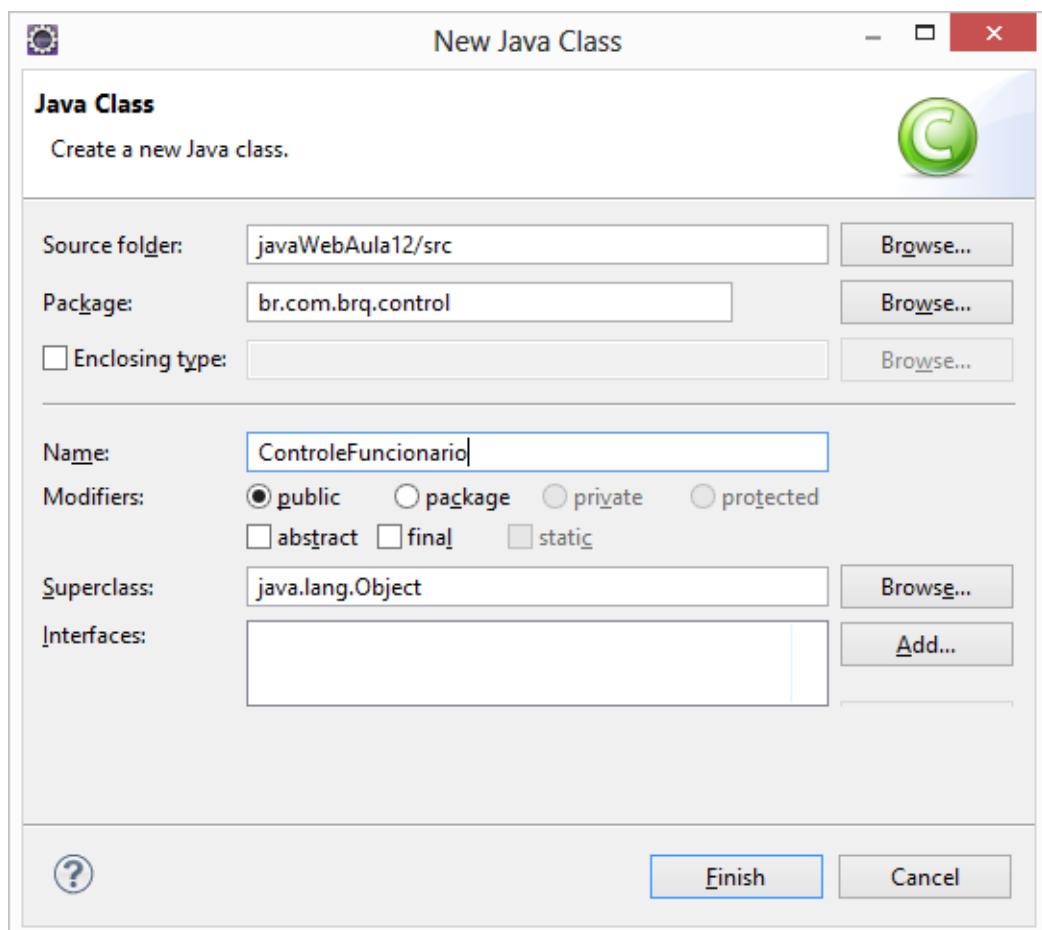
Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

22

```
<body>  
    <h3>Cadastro de Funcionários</h3>  
    <a href="index.jsp">Voltar</a>  
    <hr/>  
  
</body>  
  
</html>
```





```
package br.com.brq.control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

//Toda Classe do Struts que exerce papel de controle herda 'Action'
public class ControleFuncionario extends Action {

    //É necessário sobrescrever o método que fará a
    //função de doGet e doPost denominado 'execute'

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        return super.execute(mapping, form, request, response);
    }
}
```

### struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

    <form-beans>
        <form-bean name="" type="" />
    </form-beans>

    <action-mappings>

        <action path="/ControleFuncionario" scope="request"
               type="br.com.brq.control.ControleFuncionario">
        </action>

    </action-mappings>

    <controller
        processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
```

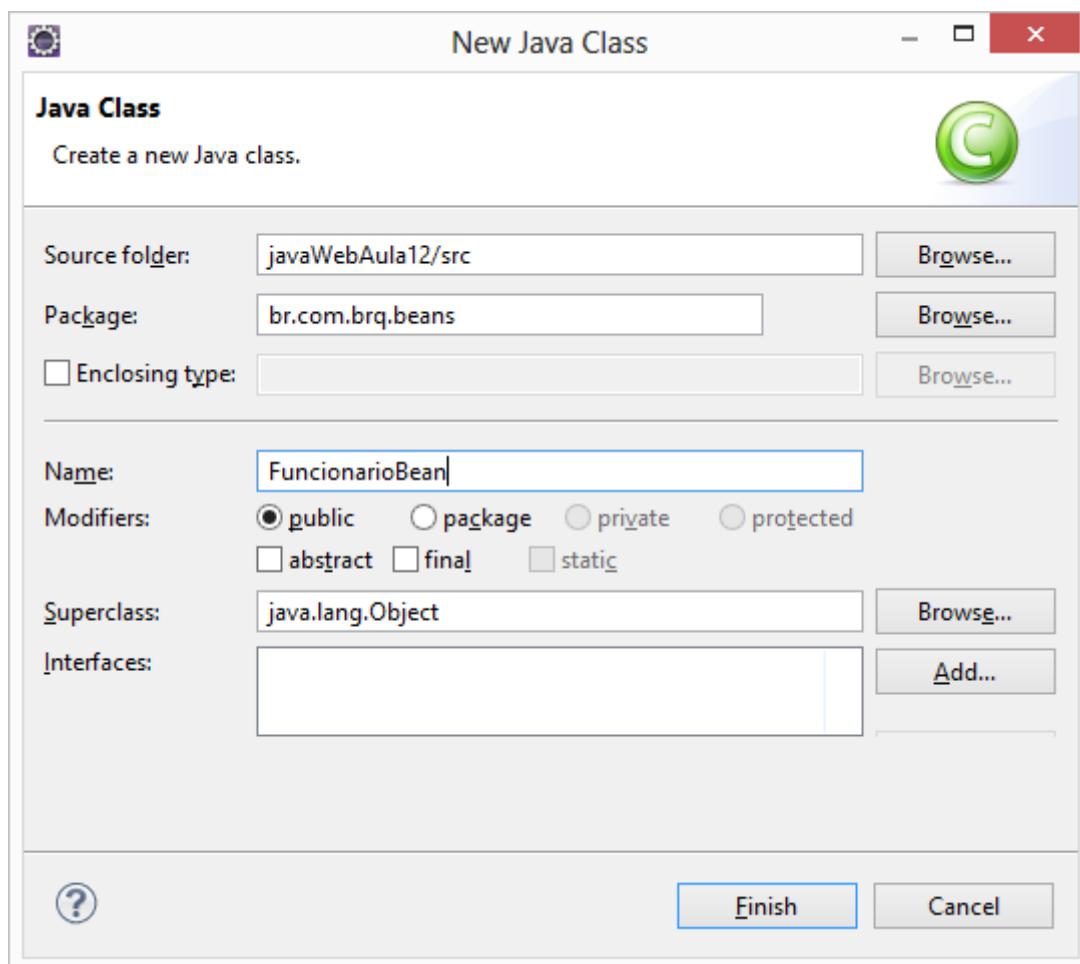


```
<set-property property="definitions-config"
    value="/WEB-INF/tiles-defs.xml"/>
<set-property property="moduleAware" value="true"/>
</plug-in>

<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-
        rules.xml,/WEB-INF/validation.xml"/>
</plug-in>

</struts-config>
```

## Camada de Modelo de dados do Struts



```
package br.com.brq.beans;

import org.apache.struts.action.ActionForm;

import br.com.brq.entity.Funcionario;

//Toda classe em struts que representa a camada de
//modelo herda ActionForm
//Definir atributos para entrada ou saída de dados,
```



```
//utilizando os JavaBeans
public class FuncionarioBean extends ActionForm{

    //Atributo para capturar os campos do formulário
    private Funcionario funcionario; //null

    public FuncionarioBean() {
        funcionario = new Funcionario(); //instanciando
    }

    public Funcionario getFuncionario() {
        return funcionario;
    }

    public void setFuncionario(Funcionario funcionario) {
        this.funcionario = funcionario;
    }
}
```

### struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

    <form-beans>
        <form-bean name="FuncionarioBean"
                   type="br.com.brq.beans.FuncionarioBean"/>
    </form-beans>

    <action-mappings>

        <action path="/ControleFuncionario"
               name="FuncionarioBean"
               scope="request"
               type="br.com.brq.control.ControleFuncionario">
        </action>

    </action-mappings>

    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
                     value="/WEB-INF/tiles-defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>

```



```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-
        rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
</struts-config>
```

---

## Criando o formulário...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="logic" %>

<html>

    <head>

        <style type="text/css">
            body { font-family: verdana;
                font-size: 9pt; padding: 30px; }
        </style>

    </head>

    <body>

        <h3>Cadastro de Funcionários</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <html:form method="post"
            action="ControleFuncionario.do?cmd=cadastrar">

            Nome do Funcionário: <br/>
            <html:text property="funcionario.nome"/>
            <br/><br/>

            Salário: <br/>
            <html:text property="funcionario.salario"/>
            <br/><br/>

            Sexo: <br/>
            <html:radio value="Masculino"
                property="funcionario.sexo"/> Masculino

        </html:form>

    </body>
</html>
```



# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
**22**

```
<html:radio value="Feminino"
property="funcionario.sexo"/> Feminino
<br/><br/>

Selecione a Função: <br/>
<html:select property="funcionario.funcao">

    <html:option value="">
    - Selecione uma Opção -</html:option>

    <html:option value="Programador">
Programador de Sistemas</html:option>

    <html:option value="Analista">
Analista de Software</html:option>

    <html:option value="DBA">
Administrador de Banco de Dados</html:option>

    <html:option value="Designer">
Desenvolvedor de Interface Web</html:option>

</html:select>
<br/><br/>

Selecione suas atribuições: <br/>

</html:form>

</body>

</html>
```

The screenshot shows a web browser window with the URL `localhost:8081/javaWebAula12/cadastro.jsp`. The page title is "Cadastro de Funcionários". The form contains the following fields:

- Voltar link
- Nome do Funcionário:** text input field
- Salário:** text input field
- Sexo:** radio button group with options "Masculino" and "Feminino". "Masculino" is selected.
- Selecionar a Função:** dropdown menu with option "- Selecione uma Opção -".
- Selecione suas atribuições:** text area.



# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

22

```
package br.com.brq.beans;

import java.util.ArrayList;
import java.util.List;

import org.apache.struts.action.ActionForm;

import br.com.brq.entity.Funcionario;

//Toda classe em struts que representa a camada de modelo herda
ActionForm
//Definir atributos para entrada ou saida de dados, utilizando os
JavaBeans
public class FuncionarioBean extends ActionForm{

    //Atributo para capturar os campos do formulário
    private Funcionario funcionario; //null

    //Atributo para gerar os itens do campo CheckBox (popular)
    private List<String> atribuicoes; //null

    //Atributo para capturar os itens marcados nos checkboxes
    private String[] itensMarcados; //null

    public FuncionarioBean() {

        funcionario = new Funcionario(); //instanciando

        atribuicoes = new ArrayList<String>();
        //instanciando a lista
        atribuicoes.add("Construção de Sistemas");
        atribuicoes.add("Design de Páginas Web");
        atribuicoes.add("Modelagem de Banco de dados");
        atribuicoes.add("Documentação de Sistemas");
        atribuicoes.add("Teste de Software");
        atribuicoes.add("Captura de Requisitos");
    }

    public Funcionario getFuncionario() {
        return funcionario;
    }

    public void setFuncionario(Funcionario funcionario) {
        this.funcionario = funcionario;
    }

    public List<String> getAtribuicoes() {
        return atribuicoes;
    }

    public void setAtribuicoes(List<String> atribuicoes) {
        this.atribuicoes = atribuicoes;
    }
}
```



```
}

public String[] getItensMarcados() {
    return itensMarcados;
}

public void setItensMarcados(String[] itensMarcados) {
    this.itensMarcados = itensMarcados;
}
-----
```

Na página de Cadastro...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>

    <head>

        <style type="text/css">
            body { font-family: verdana;
                    font-size: 9pt;
                    padding: 30px; }
        </style>

    </head>

    <body>

        <h3>Cadastro de Funcionários</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <html:form method="post"
action="ControleFuncionario.do?cmd=cadastrar">

            Nome do Funcionário: <br/>
            <html:text property="funcionario.nome"/>
            <br/><br/>
```



```
Salário: <br/>
<html:text property="funcionario.salario"/>
<br/><br/>

Sexo: <br/>
<html:radio value="Masculino"
property="funcionario.sexo"/> Masculino

<html:radio value="Feminino"
property="funcionario.sexo"/> Feminino
<br/><br/>

Selecione a Função: <br/>

<html:select property="funcionario.funcao">

    <html:option value="">- Selecione uma Opção -
    </html:option>

    <html:option value="Programador">
Programador de Sistemas</html:option>

    <html:option value="Analista">
Analista de Software</html:option>

    <html:option value="DBA">Administrador de
Banco de Dados</html:option>

    <html:option value="Designer">Desenvolvedor
de Interface Web</html:option>

</html:select>
<br/><br/>

Selecione suas atribuições: <br/>
<logic:iterate name="FuncionarioBean"
property="atribuicoes" id="item">

    <html:multibox name="FuncionarioBean"
property="itensMarcados">

        <bean:write name="item"/>
        <!-- valor do campo -->

    </html:multibox>

    <bean:write name="item"/>
    <br/>

</logic:iterate>
<br/>
```



# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

22

```
<html:submit value="Cadastrar Funcionário"/>

<p>
    <bean:write name="msg" ignore="true"/>
</p>

</html:form>

</body>

</html>
```

Executando...

## Mapeamento da Classe Funcionário JPA - Java Persistence API

```
package br.com.brq.entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```



```
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name="funcionario")
@NamedQueries(
{
    @NamedQuery(name="funcionario.listartodos",
                query="select f from Funcionario
                      as f order by f.nome asc")
}
)
public class Funcionario {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="idfuncionario")
    private Integer idFuncionario;

    @Column(name="nome", nullable = false, length=50)
    private String nome;

    @Column(name="salario", nullable = false)
    private Double salario;

    @Column(name="sexo", nullable = false, length=15)
    private String sexo;

    @Column(name="funcao", nullable = false, length=50)
    private String funcao;

    @Column(name="atribuicoes", nullable = false, length=255)
    private String atribuicoes;

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name="datacadastro")
    private Date dataCadastro;

    public Funcionario() {
    }

    public Funcionario(Integer idFuncionario, String nome,
                       Double salario, String sexo, String funcao,
                       String atribuicoes, Date dataCadastro) {
        super();
        this.idFuncionario = idFuncionario;
        this.nome = nome;
```



```
        this.salario = salario;
        this.sexo = sexo;
        this.funcao = funcao;
        this.atribuicoes = atribuicoes;
        this.dataCadastro = dataCadastro;
    }

@Override
public String toString() {
    return "Funcionario [idFuncionario=" + idFuncionario
           + ", nome=" + nome + ", salario=" + salario
           + ", sexo=" + sexo + ", funcao=" + funcao
           + ", atribuicoes=" + atribuicoes
           + ", dataCadastro=" + dataCadastro + "]";
}

public Integer getIdFuncionario() {
    return idFuncionario;
}

public void setIdFuncionario(Integer idFuncionario) {
    this.idFuncionario = idFuncionario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getSalario() {
    return salario;
}

public void setSalario(Double salario) {
    this.salario = salario;
}

public String getSexo() {
    return sexo;
}

public void setSexo(String sexo) {
    this.sexo = sexo;
}

public String getFuncao() {
    return funcao;
}
```



```
public void setFuncao(String funcao) {
    this.funcao = funcao;
}

public String getAtribuicoes() {
    return atribuicoes;
}

public void setAtribuicoes(String atribuicoes) {
    this.atribuicoes = atribuicoes;
}

public Date getDataCadastro() {
    return dataCadastro;
}

public void setDataCadastro(Date dataCadastro) {
    this.dataCadastro = dataCadastro;
}
}
```

## hibernate.cfg.xml

```
C:\WINDOWS\system32\cmd.exe - mysql -u root
mysql> create database aulaweb12;
Query OK, 1 row affected (0.00 sec)

mysql>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect
        </property>
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/aulaweb12
        </property>
    </session-factory>
</hibernate-configuration>
```



```
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password"></property>

<property name="hibernate.show_sql">true</property>
<property name="hibernate.format_sql">true</property>

<mapping class="br.com.brq.entity.Funcionario"/>

</session-factory>
</hibernate-configuration>
```

### Criando o programa para gerar o conteúdo da base de dados...

```
package br.com.brq.hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.cfg.Configuration;
import org.hibernate.tool.hbm2ddl.SchemaExport;

public class ExportSql {

    public static void main(String[] args) {

        Configuration cfg = new AnnotationConfiguration();
        cfg.configure("br/com/brq/config/
                      mysql_hibernate.cfg.xml");

        new SchemaExport(cfg).create(true, true);
    }
}
```

Executando...

```
log4j:WARN No appenders could be found for logger
(org.hibernate.cfg.annotations.Version).
log4j:WARN Please initialize the log4j system properly.
```

```
drop table if exists funcionario

create table funcionario (
    idfuncionario integer not null auto_increment,
    atribuicoes varchar(255) not null,
    datacadastro datetime,
    funcao varchar(50) not null,
    nome varchar(50) not null,
    salario double precision not null,
    sexo varchar(15) not null,
    primary key (idfuncionario)
)
```



```
C:\WINDOWS\system32\cmd.exe - mysql -u root
mysql> create database aulaweb12;
Query OK, 1 row affected (0.00 sec)

mysql> use aulaweb12;
Database changed
mysql> desc funcionario;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idfuncionario | int(11) | NO  | PRI | NULL    | auto_increment |
| atribuicoes | varchar(255) | NO  |     | NULL    |              |
| datacadastro | datetime   | YES |     | NULL    |              |
| funcao      | varchar(50)  | NO  |     | NULL    |              |
| nome        | varchar(50)  | NO  |     | NULL    |              |
| salario     | double     | NO  |     | NULL    |              |
| sexo        | varchar(15) | NO  |     | NULL    |              |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)

mysql>
```

## HibernateUtil

Classe para gerar as conexões no banco de dados através do Hibernate.

```
package br.com.brq.hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {

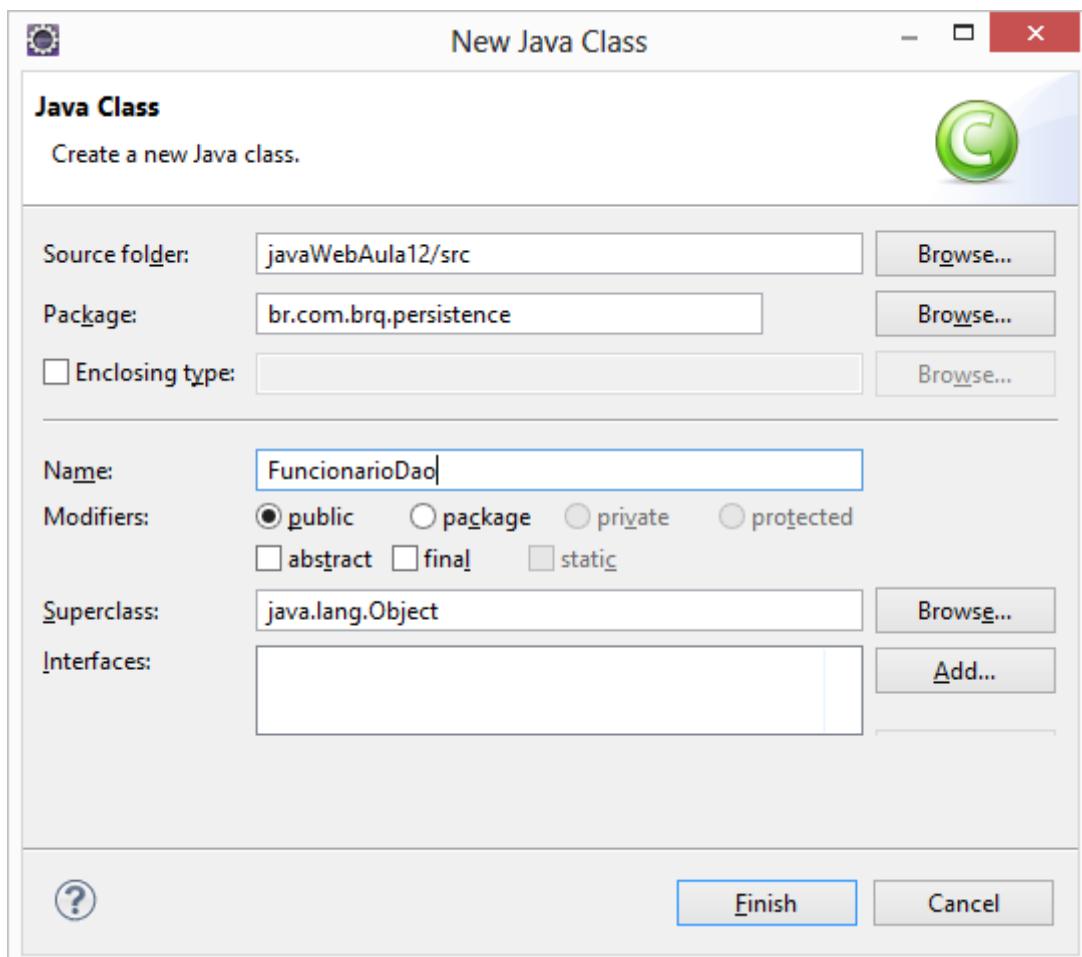
            sessionFactory = new AnnotationConfiguration()
                .configure("br/com;brq/config/
mysql_hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory
creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

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



## Classe de persistência...



```
package br.com.brq.persistence;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import br.com.brq.entity.Funcionario;
import br.com.brq.hibernate.HibernateUtil;

public class FuncionarioDao {

    private Session session;
    private Transaction transaction;
    private Query query;

    public void save(Funcionario f) throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
```



```
        session.saveOrUpdate(f);
        transaction.commit(); //executar
        session.close();
    }

    public void delete(Funcionario f) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.delete(f);
        transaction.commit(); //executar
        session.close();
    }

    public Funcionario findById(Integer idFuncionario)
        throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        Funcionario f = (Funcionario) session.get(
            Funcionario.class, idFuncionario);

        session.close();
        return f;
    }

    public List<Funcionario> findAll() throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        query = session.getNamedQuery("funcionario.listartodos");
        List<Funcionario> lista = query.list();

        session.close();
        return lista;
    }
}

-----
package br.com.brq.control;

import java.util.Date;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import br.com.brq.beans.FuncionarioBean;
import br.com.brq.persistence.FuncionarioDao;
```



```
//Toda Classe do Struts que exerce papel de controle herda
'Action'
public class ControleFuncionario extends Action {

    //É necessário sobrescrever o método que fará
    //a função de doGet e doPost denominado 'execute'

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form,
                              request, response);
        }

        return super.execute(mapping, form,
                            request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
                                   ActionForm form, HttpServletRequest request,
                                   HttpServletResponse response)
        throws Exception {

        try{

            FuncionarioBean fb = (FuncionarioBean) form;
            //Classe de Modelo

            String atribuicoes = "";
            for(int i = 0; i < fb.getItensMarcados() .
                length; i++){

                atribuicoes += fb.getItens
                               Marcados() [i] + " ";

            }

            fb.getFuncionario().setAtribuicoes(atribuicoes);
            fb.getFuncionario().setDataCadastro(new Date());
            //data do sistema

            FuncionarioDao d = new FuncionarioDao();
            d.save(fb.getFuncionario()); //salvar

            request.setAttribute("msg", "Funcionário
                                      cadastrado com sucesso.");
        }
    }
}
```



```
        catch(Exception e) {
            request.setAttribute("msg", "Erro -> " +
                e.getMessage());
        }

        return mapping.findForward("cadastro_sucesso");
    }

}

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

<form-beans>
    <form-bean name="FuncionarioBean"
        type="br.com.brq.beans.FuncionarioBean"/>
</form-beans>

<action-mappings>

    <action path="/ControleFuncionario" name="FuncionarioBean"
        scope="request"
        type="br.com.brq.control.ControleFuncionario">

        <forward name="cadastro_sucesso"
            path="/cadastro.jsp"/>

    </action>

</action-mappings>

<controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
<message-resources parameter="MessageResources"/>
<plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config" value="/WEB-INF/tiles-
defs.xml"/>
    <set-property property="moduleAware" value="true"/>
</plug-in>
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
</struts-config>
```



Executando...

Cadastro de Funcionários

[Voltar](#)

Nome do Funcionário:  
Marcone

Salário:  
2000.0

Sexo:  
 Masculino  Feminino

Seleciona a Função:  
Programador de Sistemas

Seleciona suas atribuições:

Construção de Sistemas  
 Design de Páginas Web  
 Modelagem de Banco de dados  
 Documentação de Sistemas  
 Teste de Software  
 Captura de Requisitos

Funcionário cadastrado com sucesso.

## Listando os funcionários através da Classe FuncionarioBean

```
package br.com.brq.beans;

import java.util.ArrayList;
import java.util.List;

import org.apache.struts.action.ActionForm;

import br.com.brq.entity.Funcionario;
import br.com.brq.persistence.FuncionarioDao;

//Toda classe em struts que representa a camada de modelo herda ActionForm
//Definir atributos para entrada ou saída de dados, utilizando os JavaBeans
public class FuncionarioBean extends ActionForm{

    //Atributo para capturar os campos do formulário
    private Funcionario funcionario; //null

    //Atributo para gerar os itens do campo CheckBox (popular)
    private List<String> atribuicoes; //null
```



```
//Atributo para capturar os itens marcados nos checkboxes
private String[] itensMarcados; //null

//Atributo para gerar a listagem com os funcionários
private List<Funcionario> listagemFuncionarios; //null

public FuncionarioBean() {

    funcionario = new Funcionario(); //instanciando

    atribuicoes = new ArrayList<String>();
    //instanciando a lista
    atribuicoes.add("Construção de Sistemas");
    atribuicoes.add("Design de Páginas Web");
    atribuicoes.add("Modelagem de Banco de dados");
    atribuicoes.add("Documentação de Sistemas");
    atribuicoes.add("Teste de Software");
    atribuicoes.add("Captura de Requisitos");

    itensMarcados = new String[]{ }; //vetor vazio
}

public Funcionario getFuncionario() {
    return funcionario;
}

public void setFuncionario(Funcionario funcionario) {
    this.funcionario = funcionario;
}

public List<String> getAtribuicoes() {
    return atribuicoes;
}

public void setAtribuicoes(List<String> atribuicoes) {
    this.atribuicoes = atribuicoes;
}

public String[] getItensMarcados() {
    return itensMarcados;
}

public void setItensMarcados(String[] itensMarcados) {
    this.itensMarcados = itensMarcados;
}

public List<Funcionario> getListagemFuncionarios() {

    try{
        FuncionarioDao d = new FuncionarioDao();
        listagemFuncionarios = d.findAll();
    }
}
```



```
        catch (Exception e) {
            e.printStackTrace();
        }

        return listagemFuncionarios;
    }

    public void setListagemFuncionarios
        (List<Funcionario> listagemFuncionarios) {
        this.listagemFuncionarios = listagemFuncionarios;
    }

}
```

Iterando os dados na página...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="Logic" %>

<html>

    <head>

        <style type="text/css">
            body { font-family: verdana;
                    font-size: 9pt;
                    padding: 30px; }
        </style>

    </head>

    <body>

        <h3>Cadastro de Funcionários</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <html:form method="post"
            action="ControleFuncionario.do?cmd=cadastrar">

            Nome do Funcionário: <br/>
            <html:text property="funcionario.nome"/>
            <br/><br/>
```



# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
**22**

```
Salário: <br/>
<html:text property="funcionario.salario"/>
<br/><br/>

Sexo: <br/>
<html:radio value="Masculino"
            property="funcionario.sexo"/> Masculino

<html:radio value="Feminino"
            property="funcionario.sexo"/> Feminino
<br/><br/>

Selecione a Função: <br/>
<html:select property="funcionario.funcao">

    <html:option value="">
        - Selecione uma Opção -</html:option>

    <html:option value="Programador">
        Programador de Sistemas</html:option>

    <html:option value="Analista">
        Analista de Software</html:option>

    <html:option value="DBA">Administrador de
Banco de Dados</html:option>

    <html:option value="Designer">Desenvolvedor
de Interface Web</html:option>

</html:select>
<br/><br/>

Selecione suas atribuições: <br/>

<logic:iterate name="FuncionarioBean"
                property="atribuicoes" id="item">

    <html:multibox name="FuncionarioBean"
                    property="itensMarcados">
        <bean:write name="item"/>
    </html:multibox>

    <bean:write name="item"/>
    <br/>

</logic:iterate>
```



# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
22

```
<br/>

<html:submit value="Cadastrar Funcionário"/>

<p>
    <bean:write name="msg" ignore="true"/>
</p>

<table border="1" style="width:100%;">

    <tr>
        <th>Código</th>
        <th>Nome</th>
        <th>Salário</th>
        <th>Sexo</th>
        <th>Função</th>
        <th>Atribuições</th>
        <th>Data de Cadastro</th>
    </tr>

    <logic:iterate name="FuncionarioBean"
                   property="ListagemFuncionarios" id="f">

        <tr>
            <td> <bean:write name="f" property="idFuncionario"/> </td>
            <td> <bean:write name="f" property="nome"/> </td>
            <td> <bean:write name="f" property="salario"/> </td>
            <td> <bean:write name="f" property="sexo"/> </td>
            <td> <bean:write name="f" property="funcao"/> </td>
            <td> <bean:write name="f" property="atribuicoes"/> </td>
            <td> <bean:write name="f" property="dataCadastro"/> </td>
        </tr>

        </logic:iterate>

    </table>

</html:form>

</body>

</html>
```



# Java WebDeveloper - BRQ

Quarta-feira, 11 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
**22**

## Executando...

Cadastro de Funcionários

Voltar

Nome do Funcionário:  
Sergio Mendes

Salário:  
1000.0

Sexo:  
Masculino

Selecionar a Função:  
Analista de Software

Selecionar suas atribuições:

Construção de Sistemas  
 Design de Páginas Web  
 Modelagem de Banco de dados  
 Documentação de Sistemas  
 Teste de Software  
 Captura de Requisitos

Cadastrar Funcionário

Fucionário cadastrado com sucesso.

Código	Nome	Salario	Sexo	Função	Atribuições	Data de Cadastro
1	Marcone	2000.0	Masculino	Programador	Construção de Sistemas Design de Páginas Web Documentação de Sistemas	2014-06-11
2	Sergio Mendes	1000.0	Masculino	Analista	Construção de Sistemas Documentação de Sistemas Teste de Software	2014-06-11

## No banco de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root

mysql> select * from funcionario\G
***** 1. row *****
idfuncionario: 1
atribuicoes: Construção de Sistemas Design de Páginas Web Documentação de Sistemas
datacadastro: 2014-06-11
funcao: Programador
nome: Marcone
salario: 2000
sexo: Masculino
***** 2. row *****
idfuncionario: 2
atribuicoes: Construção de Sistemas Documentação de Sistemas Teste de Software
datacadastro: 2014-06-11
funcao: Analista
nome: Sergio Mendes
salario: 1000
sexo: Masculino
2 rows in set (0.00 sec)

mysql>
```



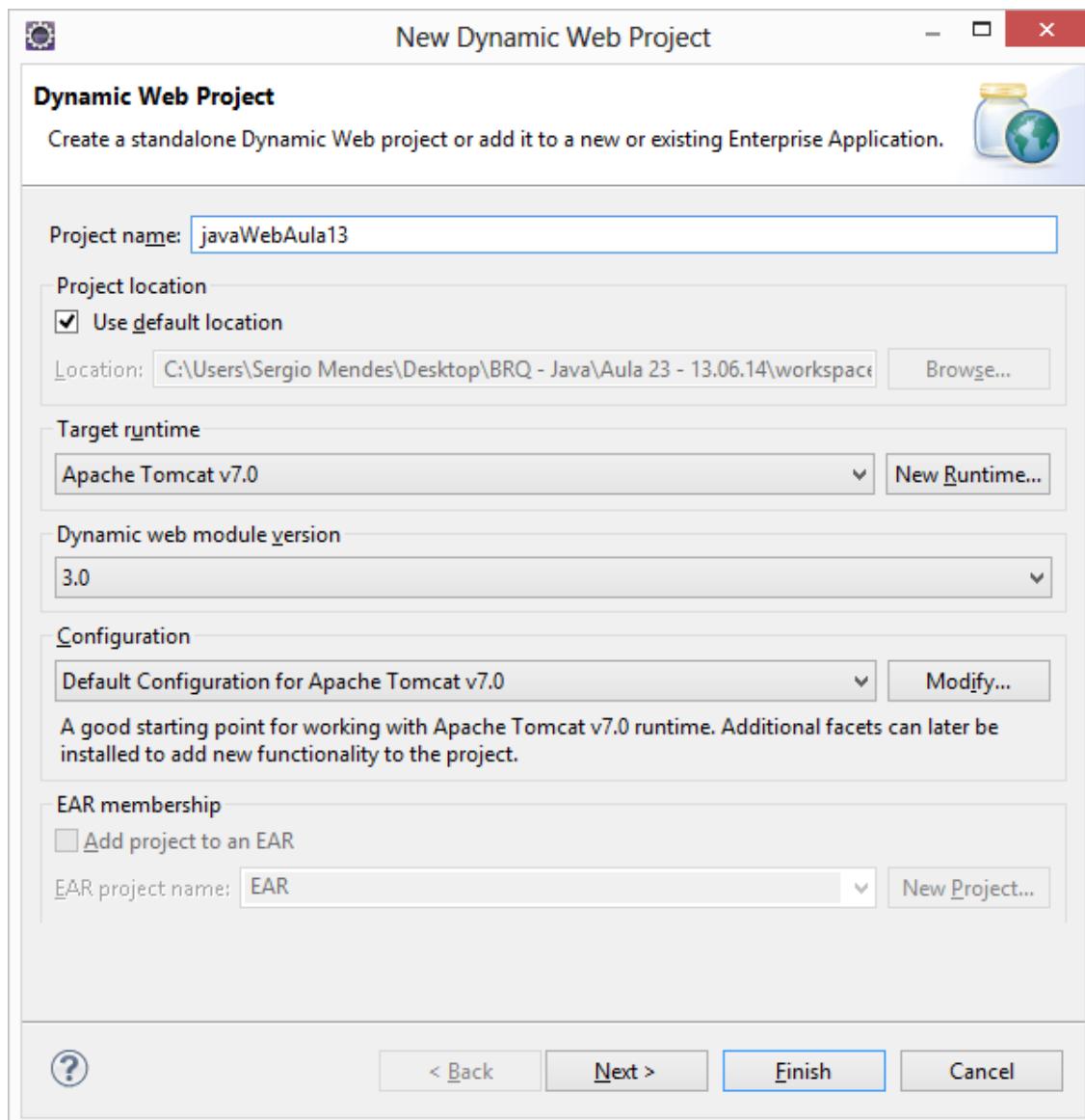
# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

23



Criando a tabela Pessoa...

```
drop database if exists aulaweb13;
create database aulaweb13;
use aulaweb13;

create table pessoa(
    idpessoa      integer          auto_increment,
    nome          varchar(50)       not null,
    email         varchar(50)       not null,
    primary key(idpessoa));

desc pessoa;
```



```
package entity;

public class Pessoa {

    private Integer idPessoa;
    private String nome;
    private String email;

    public Pessoa() {
        // Construtor default
    }

    public Pessoa(Integer idPessoa, String nome, String email) {
        super();
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.email = email;
    }

    @Override
    public String toString() {
        return "Pessoa [idPessoa=" + idPessoa + ", nome="
               + nome + ", email=" + email + "]";
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```



## Mapeamento Objeto / Relacional

JPA - Java Persistence API (@Annotations)

```
package entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "pessoa")
@NamedQueries(
{
    @NamedQuery(name="pessoa.listartodos",
                query="select p from Pessoa as p
                      order by p.nome asc")
}
)
public class Pessoa {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idpessoa")
    private Integer idPessoa;

    @Column(name = "nome", length = 50, nullable = false)
    private String nome;

    @Column(name = "email", length = 50, nullable = false)
    private String email;

    public Pessoa() {
    }

    public Pessoa(Integer idPessoa, String nome, String email) {
        super();
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.email = email;
    }

    @Override
    public String toString() {
        return "Pessoa [idPessoa=" + idPessoa + ", nome="
               + nome + ", email=" + email + "]";
    }
}
```



```
public Integer getIdPessoa() {
    return idPessoa;
}

public void setIdPessoa(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}
}
```

---

### hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">
    org.hibernate.dialect.MySQLDialect
</property>
<property name="hibernate.connection.driver_class">
    com.mysql.jdbc.Driver
</property>
<property name="hibernate.connection.url">
    jdbc:mysql://localhost:3306/aulaweb13
</property>

<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password"></property>

<property name="hibernate.show_sql">true</property>
<property name="hibernate.format_sql">true</property>
```



```
<mapping class="entity.Pessoa"/>

</session-factory>
</hibernate-configuration>
```

---

## HibernateUtil

Gerar a fábrica de conexões do Hibernate

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {

            sessionFactory = new AnnotationConfiguration().
                configure("config/mysql_hibernate.cfg.xml").
                buildSessionFactory();
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory
                creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

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

## Classe de persistência...

```
package persistence;

import hibernate.HibernateUtil;
import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

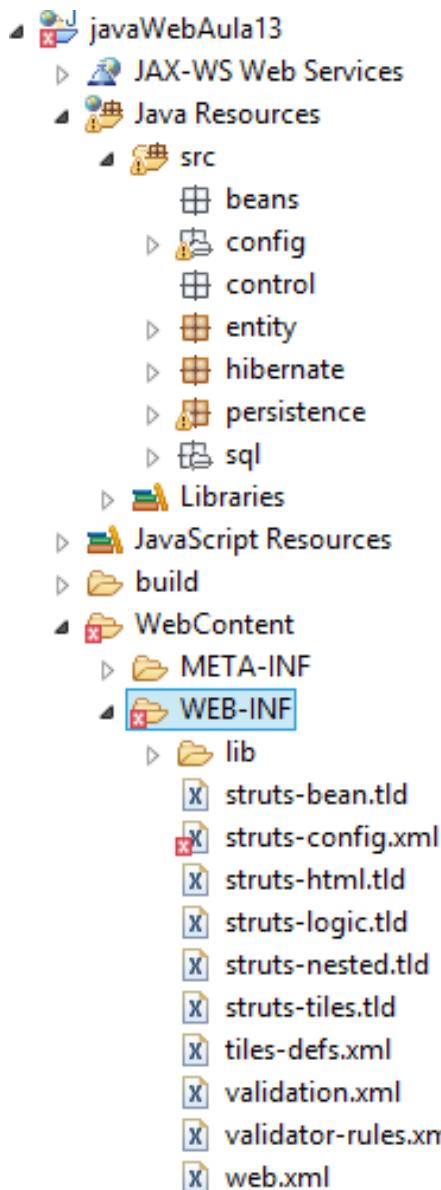
import entity.Pessoa;
```



```
public class PessoaDao {  
  
    private Session session;  
    //conexão com o banco de dados  
    private Transaction transaction;  
    //realizar transações (salvar, excluir ou atualizar)  
    private Query query;  
    //consultas escritas em linguagem HQL  
  
    public void create(Pessoa p) throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
  
        transaction = session.beginTransaction();  
        session.save(p);  
        transaction.commit();  
        session.close();  
    }  
  
    public void delete(Pessoa p) throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
  
        transaction = session.beginTransaction();  
        session.delete(p);  
        transaction.commit();  
        session.close();  
    }  
  
    public void update(Pessoa p) throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
  
        transaction = session.beginTransaction();  
        session.update(p);  
        transaction.commit();  
        session.close();  
    }  
  
    public Pessoa findById(Integer idPessoa) throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
        Pessoa p = (Pessoa) session.get(Pessoa.class, idPessoa);  
        session.close();  
        return p;  
    }  
  
    public List<Pessoa> findAll() throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
        query = session.getNamedQuery("pessoa.listartodos");  
        List<Pessoa> lista = query.list();  
        //executo a consulta e retorno os registros  
        session.close();  
        return lista;  
    }  
}
```



### Incluindo o Struts no Projeto...



### Criando a Classe de Controle para Pessoa...

```
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
```



```
public class ControlePessoa extends Action {  
  
    @Override  
    public ActionForward execute(ActionMapping mapping,  
                                ActionForm form, HttpServletRequest request,  
                                HttpServletResponse response)  
        throws Exception {  
  
        return super.execute(mapping, form, request, response);  
    }  
}
```

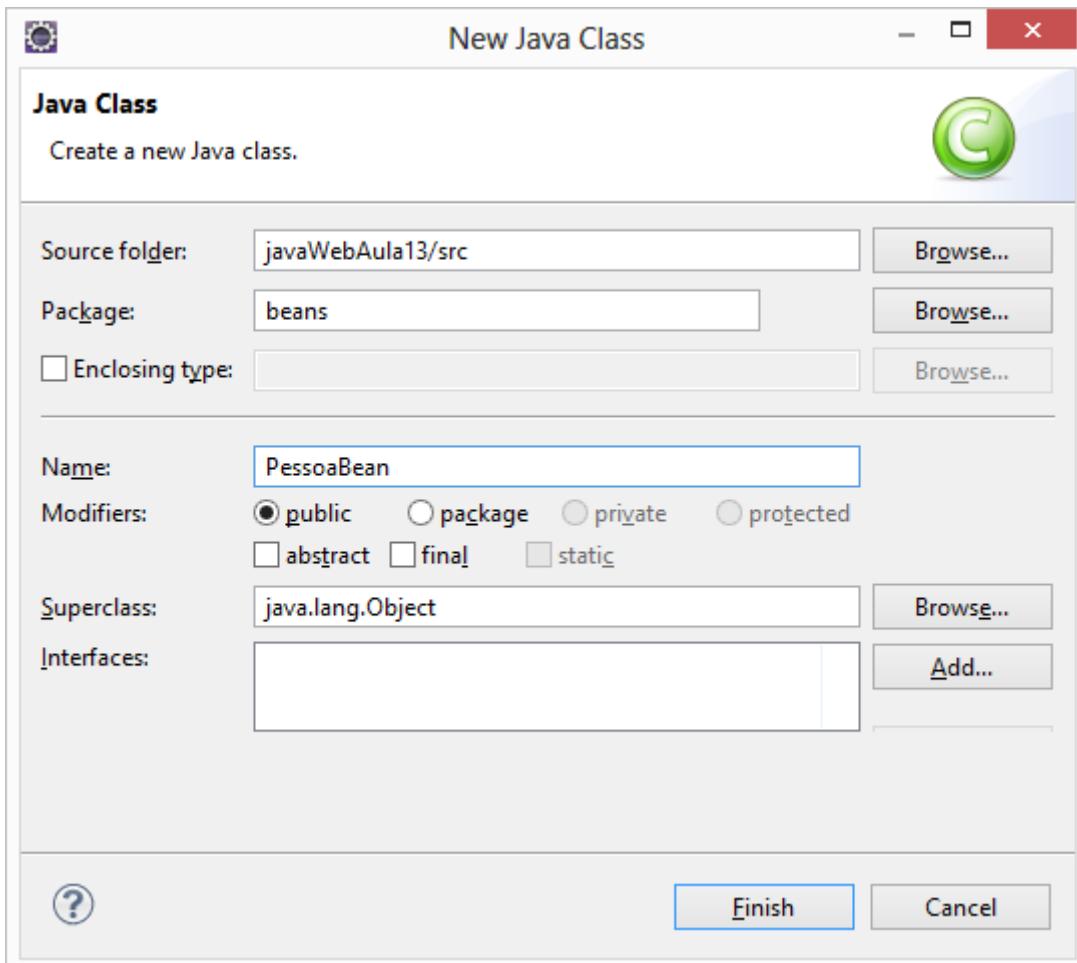
## WEB-INF/struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD  
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-  
config_1_2.dtd">  
<struts-config>  
  
<form-beans>  
    <form-bean name="" type="" />  
</form-beans>  
  
<action-mappings>  
  
    <action path="/ControlePessoa" scope="request"  
           type="control.ControlePessoa">  
        </action>  
  
</action-mappings>  
  
<controller  
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>  
    <message-resources parameter="MessageResources"/>  
    <plug-in className="org.apache.struts.tiles.TilesPlugin">  
        <set-property property="definitions-config"  
                     value="/WEB-INF/tiles-defs.xml"/>  
        <set-property property="moduleAware" value="true"/>  
    </plug-in>  
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">  
        <set-property property="pathnames" value="/WEB-INF/validator-  
rules.xml,/WEB-INF/validation.xml"/>  
    </plug-in>  
  
</struts-config>
```



## Camada de Modelo

Classe responsável pela entrada ou saída de dados do projeto



```
package beans;

import java.util.Collection;

import org.apache.struts.action.ActionForm;

import entity.Pessoa;

public class PessoaBean extends ActionForm {

    // Atributo para captura dos dados (cadastro, edição)
    private Pessoa pessoa;

    // Atributo para listar/exibir todas as pessoas cadastradas
    private Collection<Pessoa> listagemPessoa;

    public PessoaBean() {
        pessoa = new Pessoa(); // espaço de memória
    }
}
```



```
public Pessoa getPessoa() {
    return pessoa;
}

public void setPessoa(Pessoa pessoa) {
    this.pessoa = pessoa;
}

public Collection<Pessoa> getListagemPessoa() {
    return listagemPessoa;
}

public void setListagemPessoa(Collection<Pessoa> listagemPessoa)
{
    this.listagemPessoa = listagemPessoa;
}
}
```

## WEB-INF/struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

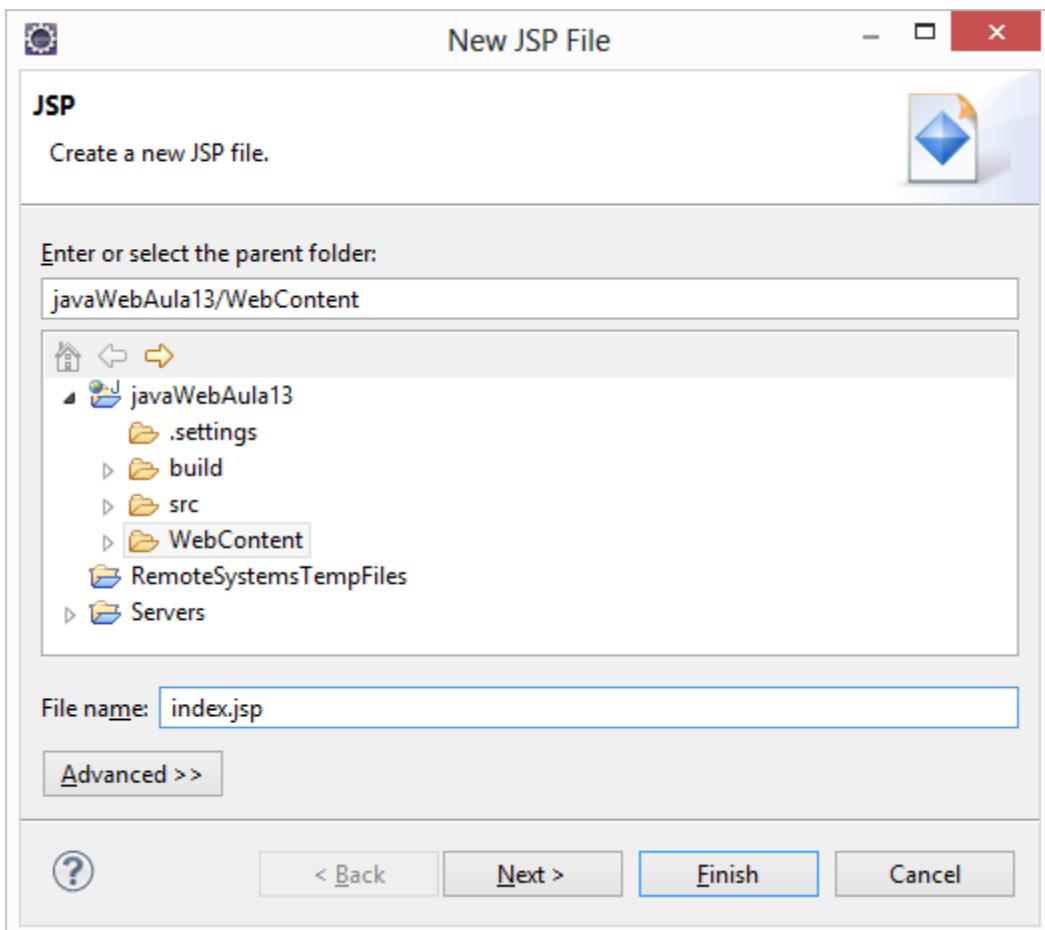
<form-beans>
    <form-bean name="PessoaBean" type="beans.PessoaBean"/>
</form-beans>

<action-mappings>
    <action path="/ControlePessoa" name="PessoaBean"
        scope="request" type="control.ControlePessoa">
        </action>
</action-mappings>

<controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
<message-resources parameter="MessageResources"/>
<plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config"
        value="/WEB-INF/tiles-defs.xml"/>
    <set-property property="moduleAware" value="true"/>
</plug-in>
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
</struts-config>
```



Criando a página inicial do projeto...



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

    <head>
        <style type="text/css">
            body { font-family: 'Verdana'; font-size: 9pt;
                    padding: 40px; }
        </style>
    </head>

    <body>

        <h3>Bem vindo ao Projeto</h3>
        <hr/>

        <a href="cadastro.jsp">Manter dados de Pessoas</a>

    </body>

</html>
```

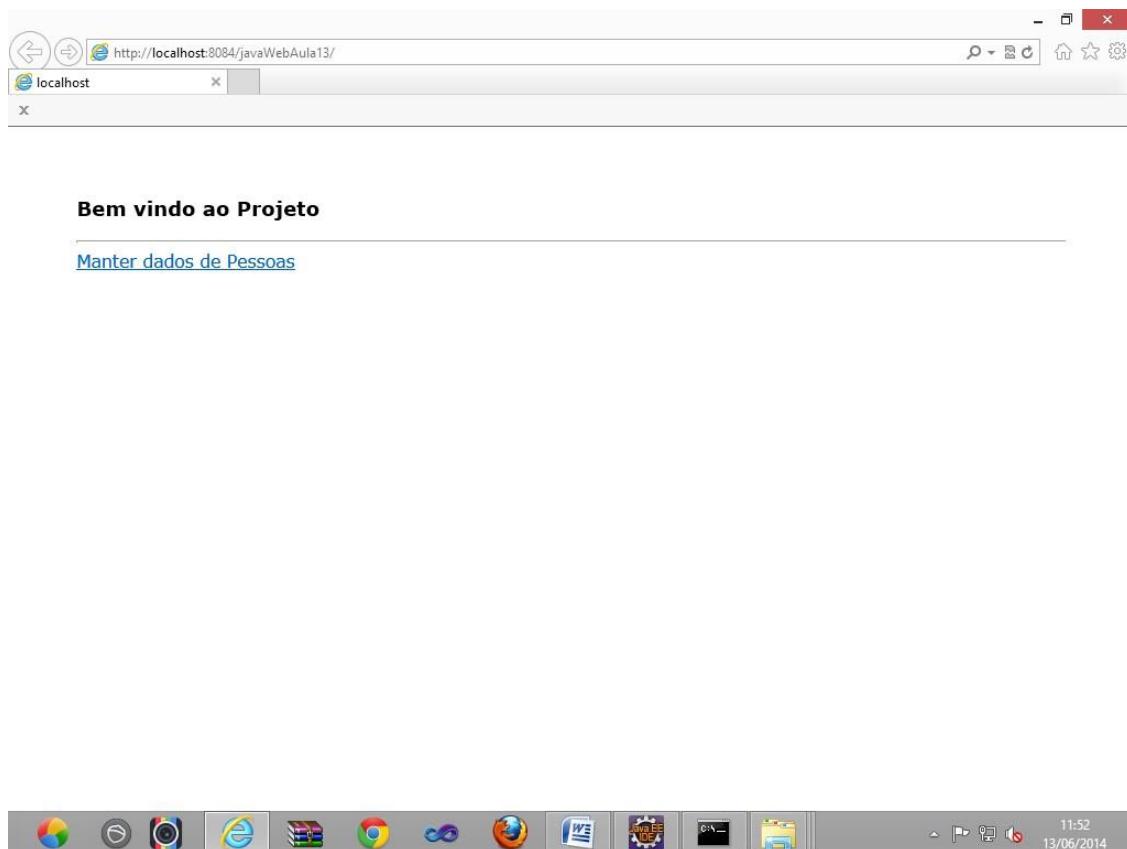


# Java WebDeveloper - BRQ

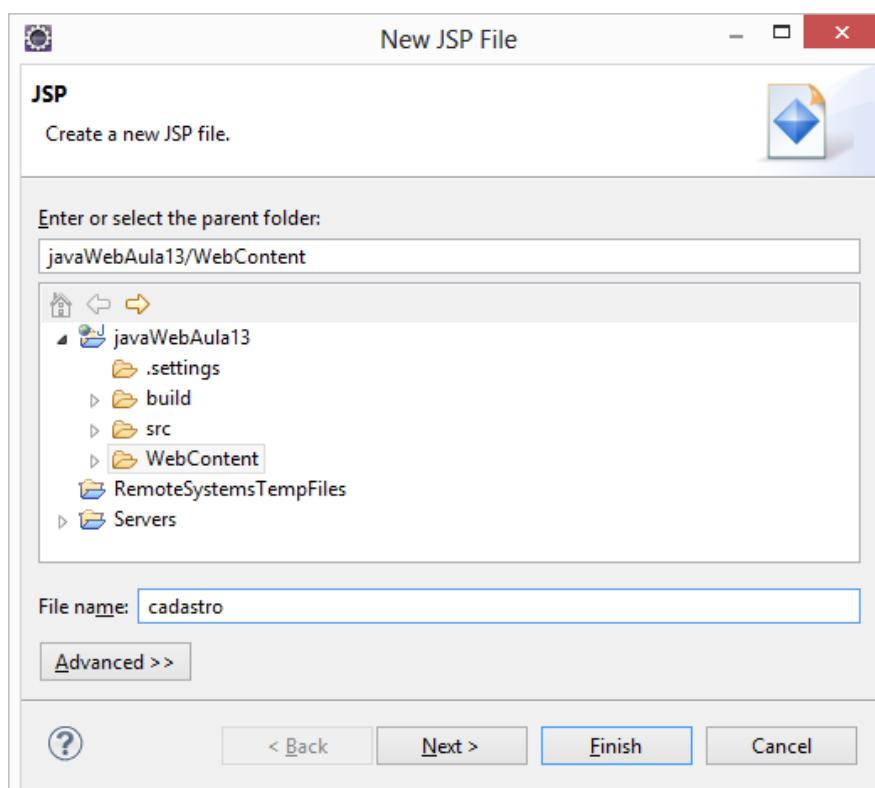
Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23



Criando a página de cadastro de pessoas...





```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>

    <head>
        <style type="text/css">
            body { font-family: 'Verdana';
                    font-size: 9pt; padding: 40px; }
        </style>
    </head>

    <body>

        <h3>Cadastro de Pessoas</h3>
        <a href="index.jsp">Voltar</a> para a página inicial
        <hr/>

        <html:form method="post"
                   action="ControlePessoa.do?cmd=cadastrar">

            Informe o Nome: <br/>
            <html:text property="pessoa.nome"/>
            <br/><br/>

            Informe o Email: <br/>
            <html:text property="pessoa.email"/>
            <br/><br/>

            <html:submit value="Cadastrar Pessoa"/>

            <p>
                <bean:write name="msg" ignore="true"/>
            </p>

            <h4>Relação de pessoas cadastradas</h4>

            <table border="1" style="width: 80%;
                           font-size: 9pt;">
                <thead>
                    <tr>
                        <th>Código</th>
                        <th>Nome da Pessoa</th>
                        <th>Endereço de Email</th>
                        <th>Excluir</th>
                    </tr>
                <tbody>
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

```
<th>Editar</th>
</tr>
</thead>

<tbody>
</tbody>

<tfoot>
<tr>
<td colspan="3">
Quantidade de registro(s): </td>
<td colspan="2"> <a href="">
Gerar Relatório</a> </td>
</tr>
</tfoot>

</table>

</html:form>

</body>

</html>
```



## Cadastro de Pessoas

[Voltar](#) para a página inicial

Informe o Nome:

Informe o Email:

## Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
Quantidade de registro(s):				<a href="#">Gerar Relatório</a>





## Implementando o cadastro de Pessoas

```
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import persistence.PessoaDao;
import beans.PessoaBean;

public class ControlePessoa extends Action {

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form,
                            request, response);
        }

        return super.execute(mapping, form,
                            request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
                                   ActionForm form, HttpServletRequest request,
                                   HttpServletResponse response)
        throws Exception {

        try{
            PessoaBean pb = (PessoaBean) form;
            //Classe FormBean

            PessoaDao d = new PessoaDao();
            d.create( pb.getPessoa() ); //gravando

            request.setAttribute("msg", "Dados gravados
                                com sucesso.");
        }
        catch(Exception e){
            request.setAttribute("msg", "Erro -> "
                                + e.getMessage());
        }
    }
}
```



```
//redirecionamento  
return mapping.findForward("pgcadastro");  
}  
}
```

## Mapeando o redirecionamento no struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD  
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-  
config_1_2.dtd">  
  
<struts-config>  
  
<form-beans>  
    <form-bean name="PessoaBean" type="beans.PessoaBean"/>  
</form-beans>  
  
<action-mappings>  
  
    <action path="/ControlePessoa" name="PessoaBean" scope="request"  
          type="control.ControlePessoa">  
  
        <forward path="/cadastro.jsp" name="pgcadastro"/>  
  
    </action>  
  
</action-mappings>  
  
<controller  
      processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>  
    <message-resources parameter="MessageResources"/>  
    <plug-in className="org.apache.struts.tiles.TilesPlugin">  
        <set-property property="definitions-config"  
                      value="/WEB-INF/tiles-defs.xml"/>  
        <set-property property="moduleAware" value="true"/>  
    </plug-in>  
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">  
        <set-property property="pathnames"  
                      value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>  
    </plug-in>  
  
</struts-config>
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23



## Cadastro de Pessoas

[Voltar](#) para a página inicial

Informe o Nome:

Ana Paula

Informe o Email:

ana@gmail.com

Dados gravados com sucesso.

## Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
Quantidade de registro(s):				<a href="#">Gerar Relatório</a>



## No MySql

```
C:\WINDOWS\system32\cmd.exe - mysql -u root
mysql> select * from pessoa;
+----+-----+-----+
| idpessoa | nome      | email      |
+----+-----+-----+
|    1 | Sergio Mendes | sergio@gmail.com |
|    2 | Ana Paula   | ana@gmail.com  |
+----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

## Listando pessoas...

```
package beans;

import java.util.Collection;

import org.apache.struts.action.ActionForm;

import persistence.PessoaDao;
import entity.Pessoa;
```



```
public class PessoaBean extends ActionForm {  
  
    // Atributo para captura dos dados (cadastro, edição)  
    private Pessoa pessoa;  
  
    // Atributo para listar/exibir todas as pessoas cadastradas  
    private Collection<Pessoa> listagemPessoa; //null  
  
    public PessoaBean() {  
        pessoa = new Pessoa(); // espaço de memória  
    }  
  
    public Pessoa getPessoa() {  
        return pessoa;  
    }  
  
    public void setPessoa(Pessoa pessoa) {  
        this.pessoa = pessoa;  
    }  
  
    public Collection<Pessoa> getListagemPessoa() {  
  
        try{  
  
            PessoaDao d = new PessoaDao();  
            //classe de persistência  
            listagemPessoa = d.findAll();  
  
        }  
        catch(Exception e){  
            e.printStackTrace();  
        }  
  
        return listagemPessoa;  
    }  
  
    public void setListagemPessoa(Collection<Pessoa>  
        listagemPessoa) {  
        this.listagemPessoa = listagemPessoa;  
    }  
}  
  
-----  
  
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
  
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"  
prefix="html" %>  
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"  
prefix="bean" %>  
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"  
prefix="Logic" %>
```



```
<html>

    <head>
        <style type="text/css">
            body { font-family: 'Verdana';
                    font-size: 9pt; padding: 40px; }
        </style>
    </head>

    <body>

        <h3>Cadastro de Pessoas</h3>
        <a href="index.jsp">Voltar</a> para a página inicial
        <hr/>

        <html:form method="post"
                   action="ControlePessoa.do?cmd=cadastrar">

            Informe o Nome: <br/>
            <html:text property="pessoa.nome"/>
            <br/><br/>

            Informe o Email: <br/>
            <html:text property="pessoa.email"/>
            <br/><br/>

            <html:submit value="Cadastrar Pessoa"/>

            <p>
                <bean:write name="msg" ignore="true"/>
            </p>

            <h4>Relação de pessoas cadastradas</h4>

            <table border="1" style="width: 80%;
                           font-size: 9pt;">
                <thead>
                    <tr>
                        <th>Código</th>
                        <th>Nome da Pessoa</th>
                        <th>Endereço de Email</th>
                        <th>Excluir</th>
                        <th>Editar</th>
                    </tr>
                </thead>

                <tbody>

                    <logic:iterate name="PessoaBean"
                                   property="ListagemPessoa" id="p">
                        <tr>
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

```
<td> <bean:write name="p" property="idPessoa"/> </td>
<td> <bean:write name="p" property="nome"/> </td>
<td> <bean:write name="p" property="email"/> </td>
<td> <a href="">Excluir</a> </td>
<td> <a href="">Editar</a> </td>
</tr>
</logic:iterate>

</tbody>
<tfoot>
    <tr>
        <td colspan="3">
            Quantidade de registro(s):
            <bean:size name="PessoaBean"
            property="listagemPessoa" id="qtd"/>
            <bean:write name="qtd"/>
        </td>
        <td colspan="2"> <a href="">Gerar Relatório</a>
        </td>
    </tr>
</tfoot>
</table>
</html:form>
</body>
</html>
```



## Cadastro de Pessoas

[Voltar](#) para a página inicial

Informe o Nome:

Informe o Email:

## Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
2	Ana Paula	ana@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
3	Marcone Freitas	marcone@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
1	Sergio Mendes	sergio@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
Quantidade de registro(s): 3			<a href="#">Gerar Relatório</a>	





Criando os links para as ações de exclusão e update...

```
<a href='ControlePessoa.do?cmd=excluir&id=<bean:write name="p" property="idPessoa"/>'>ExcluirEditar</a>
```

---

## QueryString

Passagem de variáveis por URL (barra de endereço)

```
http://.../.../ControlePessoa.do?cmd=excluir&id=2
http://.../.../ControlePessoa.do?cmd=editar&id=2
```

---

```
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import persistence.PessoaDao;
import beans.PessoaBean;
import entity.Pessoa;

public class ControlePessoa extends Action {

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form, request,
                            response);
        }
    }
}
```



```
else if(cmd.equalsIgnoreCase("excluir")) {  
    return excluir(mapping, form,  
                   request, response);  
}  
  
return super.execute(mapping, form,  
                     request, response);  
}  
  
public ActionForward cadastrar(ActionMapping mapping,  
                               ActionForm form, HttpServletRequest request,  
                               HttpServletResponse response)  
throws Exception {  
  
try{  
    PessoaBean pb = (PessoaBean) form;  
    //Classe FormBean  
  
    PessoaDao d = new PessoaDao();  
    d.create( pb.getPessoa() ); //gravando  
  
    request.setAttribute("msg", "Dados gravados  
                           com sucesso.");  
}  
catch(Exception e){  
    request.setAttribute("msg", "Erro -> "  
                         + e.getMessage());  
}  
  
//redirecionamento  
return mapping.findForward("pgcadastro");  
}  
  
public ActionForward excluir(ActionMapping mapping,  
                            ActionForm form, HttpServletRequest request,  
                            HttpServletResponse response)  
throws Exception {  
  
try{  
    Integer idPessoa = new Integer(request.  
                                    getParameter("id"));  
  
    PessoaDao d = new PessoaDao();  
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

```
//Obter o objeto Pessoa pelo id
Pessoa p = d.findById(idPessoa);
//buscando pessoa pelo id
//Excluir pessoa
d.delete(p); //removendo...

request.setAttribute("msg", "Pessoa " +
p.getNome() + ", excluido com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
+ e.getMessage());
}

return mapping.findForward("pgcadastro");
}
}
```



## Cadastro de Pessoas

[Voltar](#) para a página inicial

Informe o Nome:

Informe o Email:

[Cadastrar Pessoa](#)

Pessoa Sergio Mendes, excluido com sucesso.

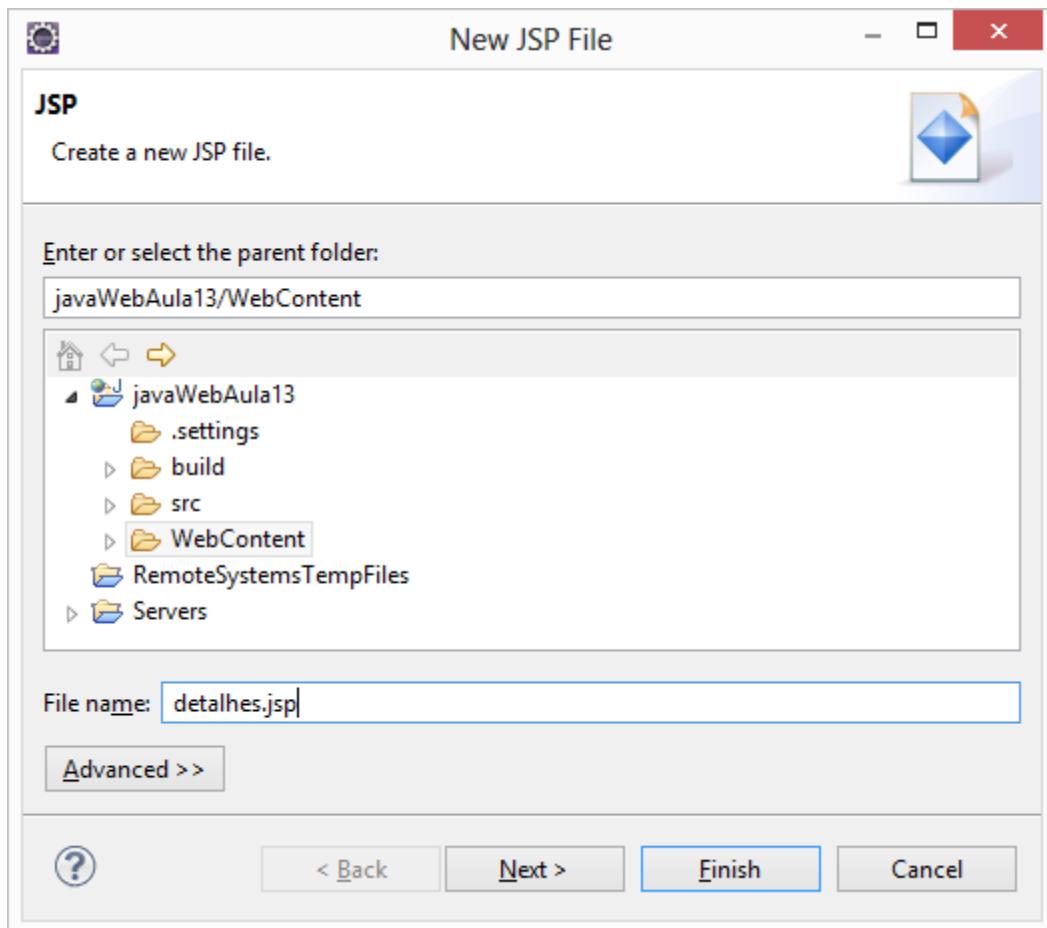
## Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
2	Ana Paula	ana@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
3	Marcone Freitas	marcone@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
Quantidade de registro(s): 2			<a href="#">Gerar Relatório</a>	





### Editando os dados de Pessoa...



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>

    <head>
        <style type="text/css">
            body { font-family: 'Verdana';
            font-size: 9pt;
            padding: 40px; }
        </style>
    </head>
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

23

```
<body>

    <h3>Editar Registro de Pessoa</h3>
    <a href="cadastro.jsp">Voltar</a>
    para a página de cadastro.
    <hr/>

    <html:form method="post"
        action="ControlePessoa.do?cmd=atualizar">

        Código da Pessoa: <br/>
        <html:text property="pessoa.idPessoa"
            readonly="true"/>
        <br/><br/>

        Nome: <br/>
        <html:text property="pessoa.nome"/>
        <br/><br/>

        Email: <br/>
        <html:text property="pessoa.email"/>
        <br/><br/>

        <html:submit value="Atualizar Dados"/>

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

---

```
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import persistence.PessoaDao;
import beans.PessoaBean;
import entity.Pessoa;

public class ControlePessoa extends Action {

    @Override
    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
```



```
String cmd = request.getParameter("cmd");

if(cmd.equalsIgnoreCase("cadastrar")){
    return cadastrar(mapping, form, request, response);
}
else if(cmd.equalsIgnoreCase("excluir")){
    return excluir(mapping, form, request, response);
}
else if(cmd.equalsIgnoreCase("editar")){
    return editar(mapping, form, request, response);
}

return super.execute(mapping, form, request, response);
}

public ActionForward cadastrar(ActionMapping mapping,
                               ActionForm form, HttpServletRequest request,
                               HttpServletResponse response)
throws Exception {

try{
    PessoaBean pb = (PessoaBean) form; //Classe FormBean

    PessoaDao d = new PessoaDao();
    d.create( pb.getPessoa() ); //gravando

    request.setAttribute("msg", "Dados gravados
                           com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
                           + e.getMessage());
}

//redirecionamento
return mapping.findForward("pgcadastro");
}

public ActionForward excluir(ActionMapping mapping,
                            ActionForm form, HttpServletRequest request,
                            HttpServletResponse response)
throws Exception {

try{
    Integer idPessoa = new Integer
        (request.getParameter("id"));

    PessoaDao d = new PessoaDao();
}
```



```
//Obter o objeto Pessoa pelo id
Pessoa p = d.findById(idPessoa);
//buscando pessoa pelo id
//Excluir pessoa
d.delete(p); //removendo...

request.setAttribute("msg", "Pessoa "
+ p.getNome() + ", excluido com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
+ e.getMessage());
}

return mapping.findForward("pgcadastro");
}

public ActionForward editar(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
throws Exception {

try{

    //resgatano o id enviado pela URL (QueryString)
    Integer idPessoa = new Integer(request.
        getParameter("id"));

    PessoaDao d = new PessoaDao();
    Pessoa p = d.findById(idPessoa);
    //obtendo Pessoa pelo id

    //Resgatando o PessoaBean
    PessoaBean pb = (PessoaBean) form;

    //passando o objeto 'p' para a Classe PessoaBean
    pb.setPessoa(p);
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
+ e.getMessage());
}

return mapping.findForward("pgdetalhes");
}
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23



## Cadastro de Pessoas

[Voltar](#) para a página inicial

Informe o Nome:

Informe o Email:

[Cadastrar Pessoa](#)

## Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
2	Ana Paula	ana@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
3	Marcone Freitas	marcone@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
Quantidade de registro(s): 2				<a href="#">Gerar Relatório</a>



## Editar Registro de Pessoa

[Voltar](#) para a página de cadastro.

Código da Pessoa:

Nome:

Email:

[Atualizar Dados](#)





## Atualizando os dados de Pessoa

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>

    <head>
        <style type="text/css">
            body { font-family: 'Verdana'; font-size: 9pt;
padding: 40px; }
        </style>
    </head>

    <body>

        <h3>Editar Registro de Pessoa</h3>
        <a href="cadastro.jsp">Voltar</a>
        para a página de cadastro.
        <hr/>

        <html:form method="post"
                    action="ControlePessoa.do?cmd=atualizar">

            Código da Pessoa: <br/>
            <html:text property="pessoa.idPessoa"
                      readonly="true"/>
            <br/><br/>

            Nome: <br/>
            <html:text property="pessoa.nome"/>
            <br/><br/>

            Email: <br/>
            <html:text property="pessoa.email"/>
            <br/><br/>

            <html:submit value="Atualizar Dados"/>

        </html:form>

    </body>

</html>
```



```
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import persistence.PessoaDao;
import beans.PessoaBean;
import entity.Pessoa;

public class ControlePessoa extends Action {

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("excluir")){
            return excluir(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("editar")){
            return editar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("atualizar")){
            return atualizar(mapping, form, request, response);
        }

        return super.execute(mapping, form, request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
                                   ActionForm form, HttpServletRequest request,
                                   HttpServletResponse response)
        throws Exception {

        try{
            PessoaBean pb = (PessoaBean) form; //Classe FormBean

            PessoaDao d = new PessoaDao();
            d.create( pb.getPessoa() ); //gravando
        }
    }
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

23

```
        request.setAttribute("msg", "Dados gravados  
        com sucesso.");  
    }  
    catch(Exception e){  
        request.setAttribute("msg", "Erro -> "  
                            + e.getMessage());  
    }  
  
    //redirecionamento  
    return mapping.findForward("pgcadastro");  
}  
  
  
public ActionForward excluir(ActionMapping mapping,  
                           ActionForm form, HttpServletRequest request,  
                           HttpServletResponse response)  
throws Exception {  
  
    try{  
  
        Integer idPessoa = new Integer(  
            request.getParameter("id"));  
  
        PessoaDao d = new PessoaDao();  
  
        //Obter o objeto Pessoa pelo id  
        Pessoa p = d.findById(idPessoa);  
        //buscando pessoa pelo id  
        //Excluir pessoa  
        d.delete(p); //removendo...  
  
        request.setAttribute("msg", "Pessoa "  
                            + p.getNome() + ", excluido com sucesso.");  
    }  
    catch(Exception e){  
        request.setAttribute("msg", "Erro -> "  
                            + e.getMessage());  
    }  
  
    return mapping.findForward("pgcadastro");  
}  
  
  
public ActionForward editar(ActionMapping mapping,  
                           ActionForm form, HttpServletRequest request,  
                           HttpServletResponse response)  
throws Exception {  
  
    try{
```



```
//resgatando o id enviado pela URL (QueryString)
Integer idPessoa = new Integer(
    request.getParameter("id"));

PessoaDao d = new PessoaDao();
Pessoa p = d.findById(idPessoa);
//obtendo Pessoa pelo id

//Resgatando o PessoaBean
PessoaBean pb = (PessoaBean) form;

//passando o objeto 'p' para a Classe PessoaBean
pb.setPessoa(p);
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
        + e.getMessage());
}

return mapping.findForward("pgdetalhes");
}

public ActionForward atualizar(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
throws Exception {

try{

    //Classe de modelo do MVC
    PessoaBean pb = (PessoaBean) form;

    PessoaDao d = new PessoaDao();
    d.update( pb.getPessoa() ); //atualizar pessoa
                                //contido no PessoaBean

    request.setAttribute("msg", "Dados atualizados
        com sucesso.");
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
        + e.getMessage());
}

//redirecionamento
return mapping.findForward("pgcadastro");
}
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23



## Editar Registro de Pessoa

[Voltar](#) para a página de cadastro.

Código da Pessoa:

4

Nome:

Sergio Mendes

Email:

sergio@bol.com



## Cadastro de Pessoas

[Voltar](#) para a página inicial

Informe o Nome:

Sergio Mendes

Informe o Email:

sergio@bol.com

Dados atualizados com sucesso.

## Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
2	Ana Paula	ana@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
3	Marcone Freitas	marcone@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
4	Sergio Mendes	sergio@bol.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
Quantidade de registro(s): 3			<a href="#">Gerar Relatório</a>	



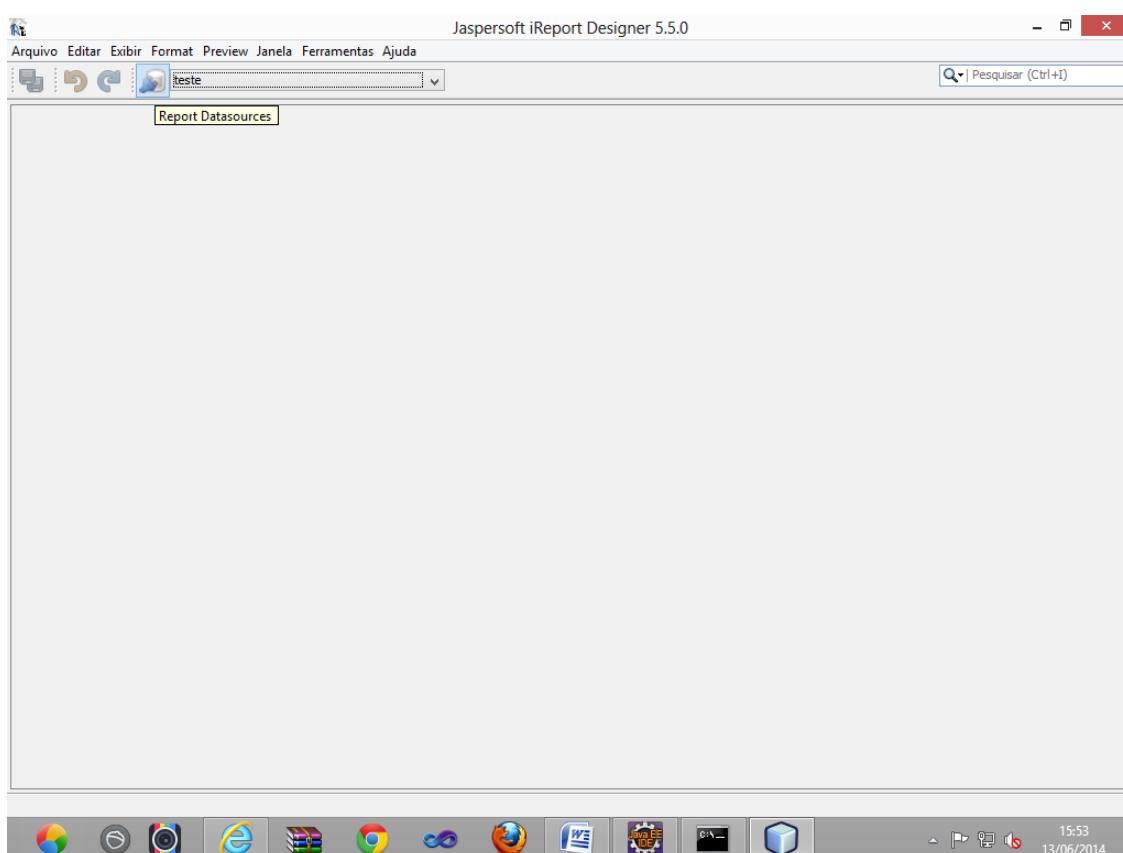


## Criando relatório com iReport



### Passo 1: Conectar o iReport ao banco de dados

Report DataSource



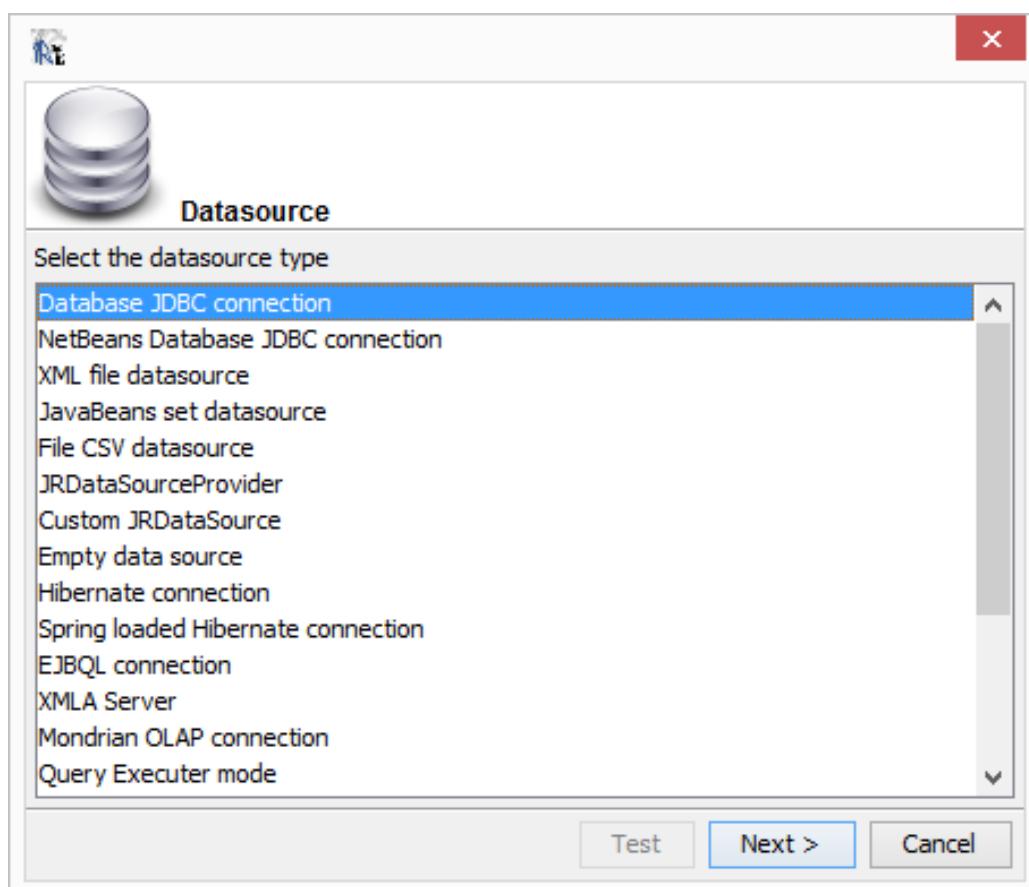
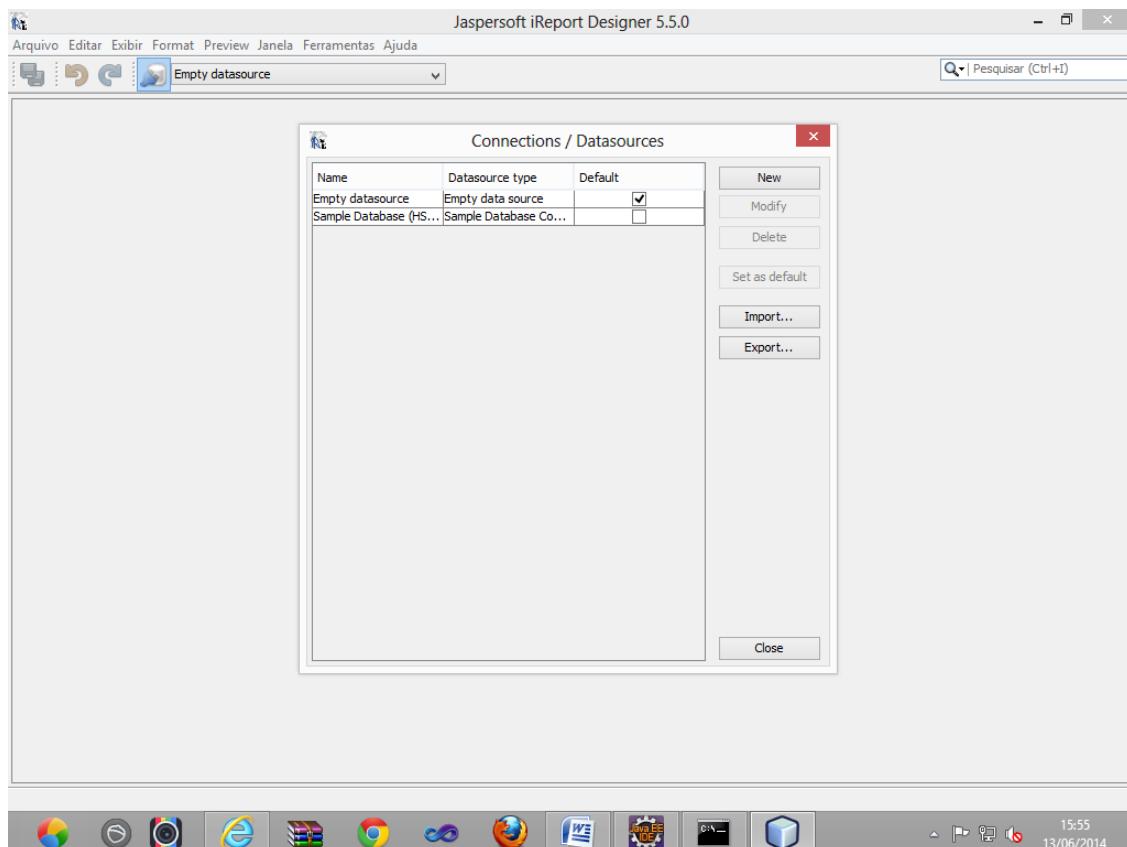


# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23





# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

**Database JDBC connection**

Name: aula13

JDBC Driver: MySQL (com.mysql.jdbc.Driver)

JDBC URL: jdbc:mysql://localhost/aulaweb13

Credentials:

Username: root

Password:

Save password

**ATTENTION!** Passwords are stored in clear text. If you don't specify a password now, iReport will ask you for one only when required and will not save it.

**Test** **Save** **Cancel**

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Janela Ferramentas Ajuda

New... Open Open Recent File Salvar Ctrl+S Salvar como... Salvar todos Ctrl+Shift+S Configurar página... Imprimir... Ctrl+Alt+Shift+P Imprimir em HTML... Sair

Pesquisar (Ctrl+I)



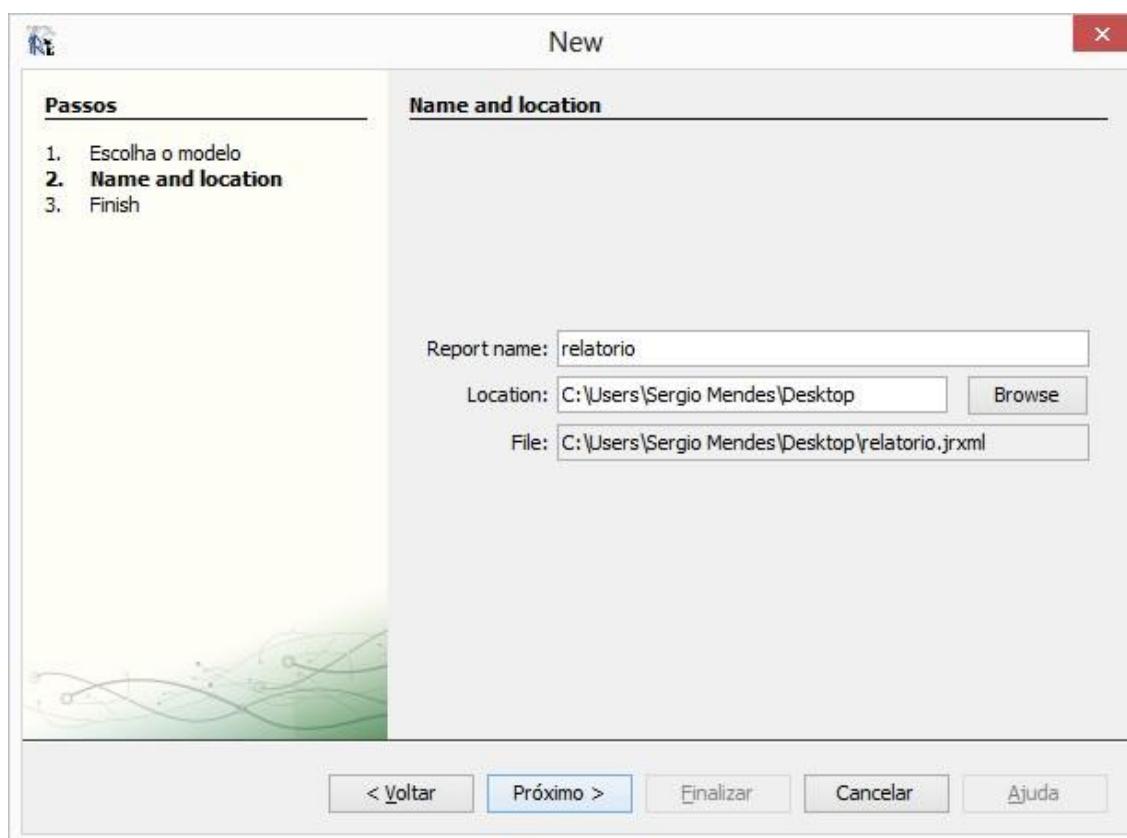
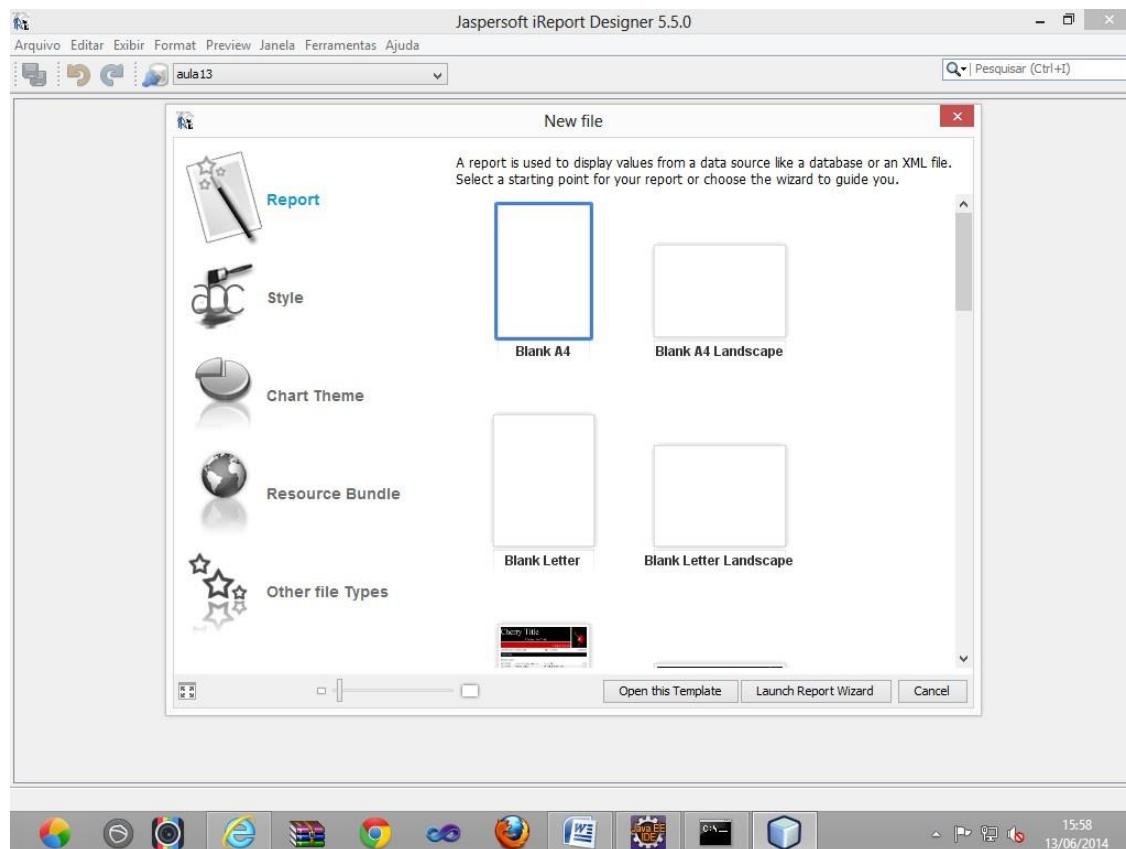
# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Aula

23

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1





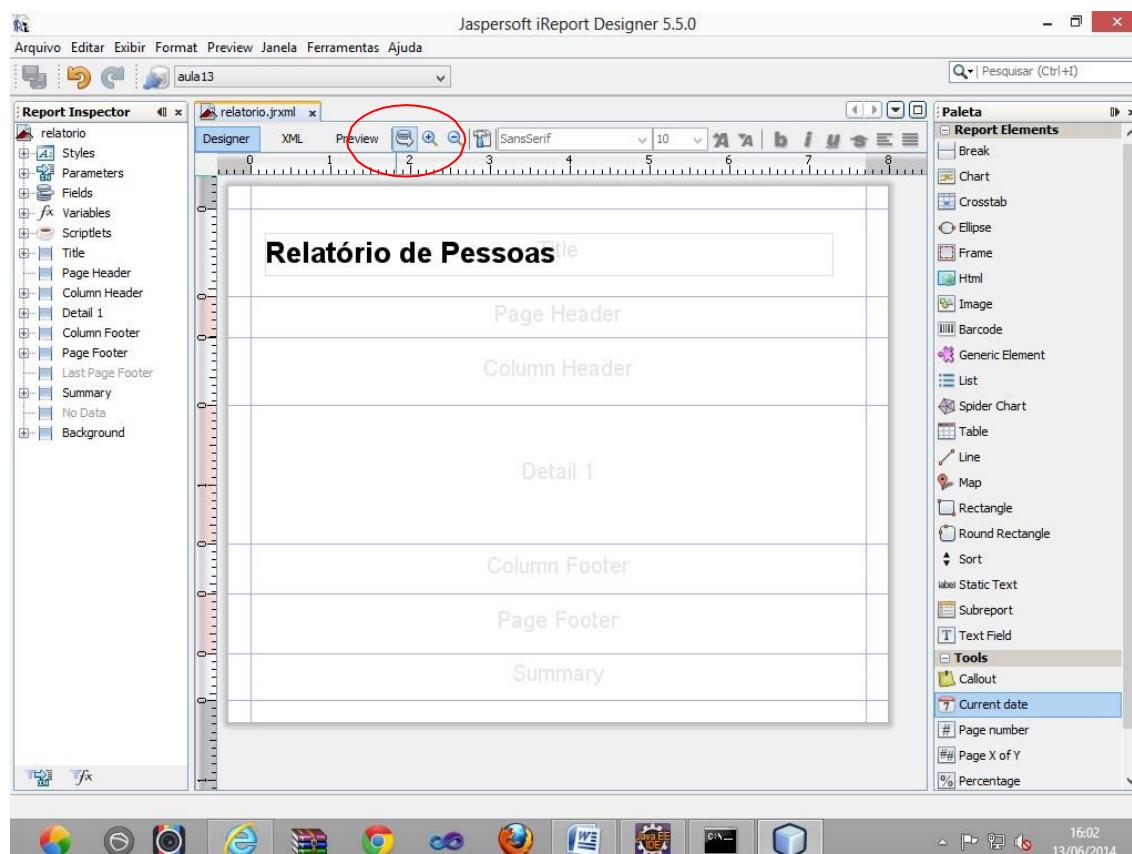
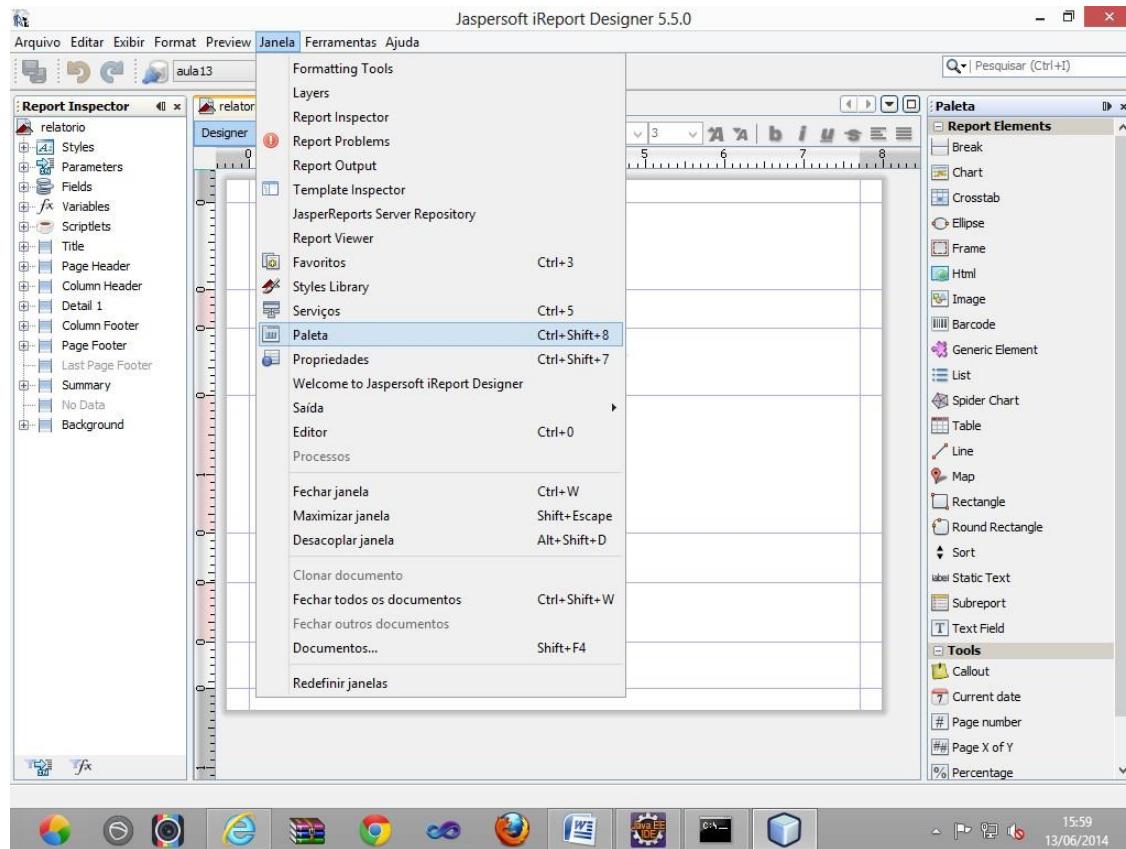
# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

23





# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

23

Report query

Report query JavaBean Datasource DataSource Provider CSV Datasource Excel Datasource

Query language SQL

```
select * from pessoa order by nome asc
```

Load query Save query

Drag a parameter into the query to a parameter. Hold CTL to add the parameter as query chunk.

Available parameters

New parameter

Ready

Automatically Retrieve Fields Read Fields Query d... Send to cli...

Field name	Field type	Description
idpessoa	java.lang.Integer	
nome	java.lang.String	
email	java.lang.String	

Filter expression... Sort options... Preview data OK Cancel

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Janela Ferramentas Ajuda

aula13

Report Inspector

relatorio

- Styles
- Parameters
- Fields
  - idpessoa
  - nome
  - email
- f/x
- Variables
- Scriptlets
- Title
- Page Header
- Column Header
  - label idpessoa
  - label nome
  - label email
- Detail 1
  - \$F{idpessoa}
  - \$F{nome}
  - \$F{email}
- Column Footer
- Page Footer
- Last Page Footer
- Summary
- No Data
- Background

relatorio.jrxml

Designer XML Preview

Relatório de Pessoas

Código Nome da Pessoa Endereço de Email

\$F{idpessoa} \$F{nome} \$F{email}

Detail 1

Page Header

Column Footer

Page Footer

Summary

Paleta

- Report Elements
  - Break
  - Chart
  - Crosstab
  - Ellipse
  - Frame
  - Html
  - Image
  - Barcode
  - Generic Element
  - List
  - Spider Chart
  - Table
  - Line
  - Map
  - Rectangle
  - Round Rectangle
  - Sort
  - Static Text
  - Subreport
  - Text Field
- Tools
  - Callout
  - Current date
  - Page number
  - Page X of Y
  - Percentage

Salvar todos concluído.

16:06 13/06/2014



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula

23

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Janela Ferramentas Ajuda

aula13

Report Inspector

relatorio

- Styles
- Parameters
- Fields
- Variables
- Scriptlets
- Title
  - ibm Relatório de Pessoas
- Page Header
- Column Header
- Detail 1
- Column Footer
- Page Footer
- Last Page Footer
- Summary
- No Data
- Background

Designer XML Preview SansSerif 26 1 2 3 4 5 6 7 8

Pattern editor

Category Sample

Date Sexta-feira 13 Junho 2014

Time

Type:  
13/06/2014  
06/13/2014  
2014/06/13  
**Sexta-feira 13 Junho 2014**  
Junho 13, 2014  
13/06  
13/06/14  
13-Jun  
13-Jun-14

Pattern EEEEE dd MMMMM yyyy

Apply Cancel

Paleta

- Chart
- Crosstab
- Ellipse
- Frame
- Html
- Image
- Barcode
- Generic Element
- List
- Spider Chart
- Table
- Line
- Map
- Rectangle
- Round Rectangle
- Sort
- label Static Text
- Subreport
- Text Field
- Tools
  - Callout
  - Current date**
  - # Page number
  - ## Page X of Y
  - % Percentage
  - # Total pages
  - Web Framework

16:07 13/06/2014

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Janela Ferramentas Ajuda

aula13

Report Inspector

relatorio

- Styles
- Parameters
- Fields
- Variables
- Scriptlets
- Title
- Page Header
- Column Header
- Detail 1
- Column Footer
- Page Footer
- Last Page Footer
- Summary
- No Data
- Background

Designer XML Preview SansSerif 10 1 2 3 4 5 6 7 8

Relatório de Pessoas

new java.util.Date()

Page Header

Código Nome da Pessoa Endereço de Email

\$F{idpessoa} \$F{nome} \$F{email}

Detail 1

Column Footer

Page Footer Page "+\$V" "+\$V

Summary Page Footer

Paleta

- Chart
- Crosstab
- Ellipse
- Frame
- Html
- Image
- Barcode
- Generic Element
- List
- Spider Chart
- Table
- Line
- Map
- Rectangle
- Round Rectangle
- Sort
- label Static Text
- Subreport
- Text Field
- Tools
  - Callout
  - Current date
  - Page number**
  - Page X of Y
  - Percentage
  - Total pages
  - Web Framework

16:09 13/06/2014



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

The screenshot shows the Jaspersoft iReport Designer interface. The main window displays a report titled "Relatório de Pessoas" with the subtitle "Sexta-feira 13 Junho 2014". The report contains a table with three columns: "Código", "Nome da Pessoa", and "Endereço de Email". The data is as follows:

Código	Nome da Pessoa	Endereço de Email
2	Ana Paula	ana@gmail.com
3	Marcone Freitas	marcone@gmail.com
4	Sergio Mendes	sergio@bol.com

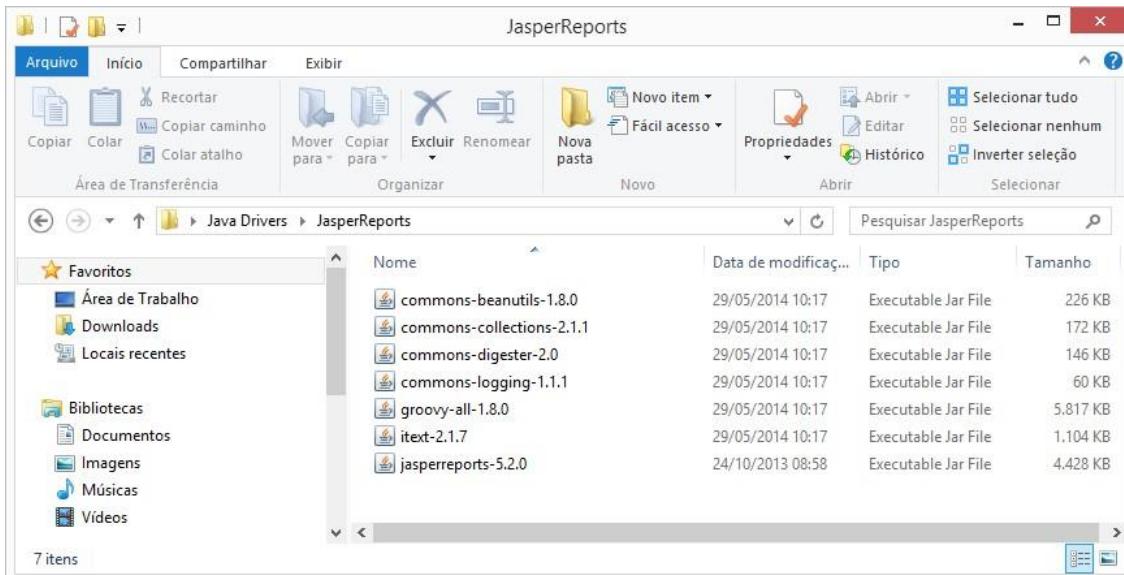
## Incluindo o arquivo do relatório no projeto...

The screenshot shows the Java EE - Eclipse IDE interface. The "Project Explorer" view on the left shows a project named "javaWebAula13". Inside the "reports" folder, the "relatorio.jasper" file is selected. The Eclipse toolbar and status bar are visible at the bottom.



```
<tfoot>
  <tr>
    <td colspan="3">
      Quantidade de registro(s):
      <bean:size name="PessoaBean"
      property="listagemPessoa" id="qtd"/>
      <bean:write name="qtd"/>
    </td>
    <td colspan="2"> <a href="ControlePessoa.do?cmd=relatorio">
      Gerar Relatório</a>
    </td>
  </tr>
</tfoot>
```

## Bibliotecas do JasperReports



```
package control;

import hibernate.HibernateUtil;
import java.io.InputStream;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import net.sf.jasperreports.engine.JasperRunManager;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
```



```
import persistence.PessoaDao;
import beans.PessoaBean;
import entity.Pessoa;

public class ControlePessoa extends Action {

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("excluir")){
            return excluir(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("editar")){
            return editar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("atualizar")){
            return atualizar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("relatorio")){
            return relatorio(mapping, form, request, response);
        }

        return super.execute(mapping, form, request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        try{
            PessoaBean pb = (PessoaBean) form; //Classe FormBean

            PessoaDao d = new PessoaDao();
            d.create( pb.getPessoa() ); //gravando

            request.setAttribute("msg", "Dados gravados
                                com sucesso.");
        }
        catch(Exception e){
            request.setAttribute("msg", "Erro -> "
                                + e.getMessage());
        }
    }
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

```
//redirecionamento
return mapping.findForward("pgcadastro");
}

public ActionForward excluir(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
throws Exception {

try{

    Integer idPessoa = new Integer(
        request.getParameter("id"));

    PessoaDao d = new PessoaDao();

    //Obter o objeto Pessoa pelo id
    Pessoa p = d.findById(idPessoa);
    //buscando pessoa pelo id
    //Excluir pessoa
    d.delete(p); //removendo...

    request.setAttribute("msg", "Pessoa " + p.getNome()
        + ", excluido com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
        + e.getMessage());
}

return mapping.findForward("pgcadastro");
}

public ActionForward editar(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
throws Exception {

try{

    //resgatano o id enviado pela URL (QueryString)
    Integer idPessoa = new Integer(
        request.getParameter("id"));

    PessoaDao d = new PessoaDao();
    Pessoa p = d.findById(idPessoa);
    //obtendo Pessoa pelo id
    //atualizar Pessoa
    d.update(p);
    request.setAttribute("msg", "Pessoa "
        + p.getNome() + " editado com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
        + e.getMessage());
}

return mapping.findForward("pgcadastro");
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

```
//Resgatando o PessoaBean
PessoaBean pb = (PessoaBean) form;

//passando o objeto 'p' para a Classe
//PessoaBean
pb.setPessoa(p);
}

catch(Exception e){
    request.setAttribute("msg", "Erro -> "
                           + e.getMessage());
}

return mapping.findForward("pgdetalhes");
}

public ActionForward atualizar(ActionMapping mapping,
                               ActionForm form, HttpServletRequest request,
                               HttpServletResponse response)
throws Exception {

try{

    //Classe de modelo do MVC
    PessoaBean pb = (PessoaBean) form;

    PessoaDao d = new PessoaDao();
    d.update( pb.getPessoa() );
    //atualizar pessoa contido no PessoaBean

    request.setAttribute("msg", "Dados atualizados com sucesso.");
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
                           + e.getMessage());
}

//redirecionamento
return mapping.findForward("pgcadastro");
}

public ActionForward relatorio(ActionMapping mapping,
                               ActionForm form, HttpServletRequest request,
                               HttpServletResponse response)
throws Exception {

try{

    //Passo 1: Localizar documento do relatorio
    //dentro do projeto
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

```
//caminho do arquivo .jasper localizado
//dentro do projeto
InputStream path = getServlet().getServletContext().
getResourceAsStream("/reports/relatorio.jasper");

//Passo 2: Transformar o documento em PDF
//passando a conexão do banco
byte[] pdf = JasperRunManager.runReportToPdf(
    path, null, HibernateUtil.
    getSessionFactory().openSession().
    connection());

//Passo 3: download do PDF
ServletOutputStream out = response.
getOutputStream(); //objeto para download

response.setHeader("Content-disposition",
    "filename=relatorio.pdf");
response.setContentType("application/pdf");

out.write(pdf);
//enviar o conteúdo do pdf para download
out.flush(); //limpar
out.close(); //fechar
}
catch(Exception e){
    request.setAttribute("msg", "Erro -> "
        + e.getMessage());
}
return null;
}
```

Cadastro de Pessoas

Voltar para a página inicial

Informe o Nome:

Informe o Email:

Cadastrar Pessoa

Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
2	Ana Paula	ana@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
3	Marcone Freitas	marcone@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
4	Sergio Mendes	sergio@bol.com	<a href="#">Excluir</a>	<a href="#">Editar</a>

Quantidade de registro(s): 3

Gerar Relatório



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

The screenshot shows a Windows desktop environment. A web browser window is open, displaying a report titled "Relatório de Pessoas". The report lists four individuals with their codes, names, and email addresses. The browser's address bar shows the URL <http://localhost:8084/javaWebAula13/ControlePessoa.do?cmd=relatorio>. The taskbar at the bottom shows various application icons, and the system tray indicates the date as 13/06/2014 and the time as 16:39.

Código	Nome da Pessoa	Endereço de Email
2	Ana Paula	ana@gmail.com
3	Marcone Freitas	marcone@gmail.com
4	Sergio Mendes	sergio@bol.com

The screenshot shows a Linux desktop environment. A web browser window is open, displaying a report titled "Relatório de Pessoas". The report lists four individuals with their codes, names, and email addresses. The browser's address bar shows the URL <http://localhost:8084/javaWebAula13/ControlePessoa.do?cmd=relatorio>. The taskbar at the bottom shows various application icons, and the system tray indicates the date as 13/06/2014 and the time as 16:48.

Código	Nome da Pessoa	Endereço de Email
2	Ana Paula	ana@gmail.com
3	Marcone Freitas	marcone@gmail.com
4	Sergio Mendes	sergio@bol.com

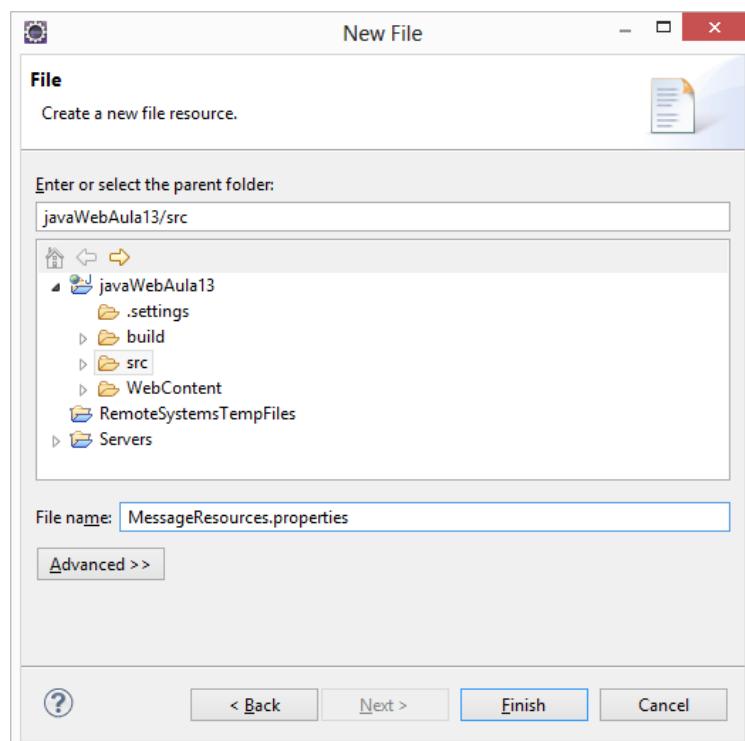


## Supporte do Struts à Internacionalização

### Arquivo padrão para mensagens do sistema

The screenshot shows the Eclipse IDE interface with the title bar "Java EE - javaWebAula13/WebContent/WEB-INF/struts-config.xml - Eclipse". The editor window displays the XML configuration for Struts. The code includes sections for form-beans, action-mappings, controller, and message-resources. A specific line of code, <message-resources parameter="MessageResources"/>, is highlighted in blue, indicating it is selected or being edited.

```
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "http://strut^
<struts-config>
  <form-beans>
    <form-bean name="PessoaBean" type="beans.PessoaBean"/>
  </form-beans>
  <action-mappings>
    <action path="/ControlePessoa" name="PessoaBean" scope="request" type="control.ControlePessoa">
      <forward path="/cadastro.jsp" name="pgcadastro"/>
      <forward path="/detalhes.jsp" name="pgdetalhes"/>
    </action>
  </action-mappings>
  <controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
  <message-resources parameter="MessageResources"/>
  <plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml"/>
    <set-property property="moduleAware" value="true"/>
  </plug-in>
  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
  </plug-in>
</struts-config>
```





#CHAVE=MENSAGEM

```
projeto.titulo=Projeto Controle de Pessoas
projeto.subtitle=Manter dados de pessoas
projeto.nome=Informe o nome da pessoa
projeto.email=Informe o email da pessoa
```

-----

### Exibindo as mensagens na página...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="logic" %>

<html>

    <head>
        <style type="text/css">
            body { font-family: 'Verdana'; font-size: 9pt;
                padding: 40px; }
        </style>
    </head>

    <body>

        <h3> <bean:message key="projeto.titulo"/> </h3>

        <p>
            <bean:message key="projeto.subtitle"/>
        </p>

        <a href="index.jsp">Voltar</a> para a página inicial
        <hr/>

        <html:form method="post"
            action="ControlePessoa.do?cmd=cadastrar">

            <bean:message key="projeto.nome"/> <br/>
            <html:text property="pessoa.nome"/>
            <br/><br/>
```



# Java WebDeveloper - BRQ

Sexta-feira, 13 de Junho de 2014

Persistência de dados com Hibernate. Mapeamento de NamedQueries com HQL. Desenvolvimento de projetos web com Struts 1

Aula  
23

```
<bean:message key="projeto.email"/> <br/>
<html:text property="pessoa.email"/>
<br/><br/>

<html:submit value="Cadastrar Pessoa"/>

<p>
    <bean:write name="msg" ignore="true"/>
</p>
```



## Projeto Controle de Pessoas

Manter dados de pessoas

[Voltar](#) para a página inicial

Informe o nome da pessoa

Informe o email da pessoa

## Relação de pessoas cadastradas

Código	Nome da Pessoa	Endereço de Email	Excluir	Editar
2	Ana Paula	ana@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
3	Marcone Freitas	marcone@gmail.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
4	Sergio Mendes	sergio@bol.com	<a href="#">Excluir</a>	<a href="#">Editar</a>
Quantidade de registro(s): 3			<a href="#">Gerar Relatório</a>	





# Java WebDeveloper - BRQ

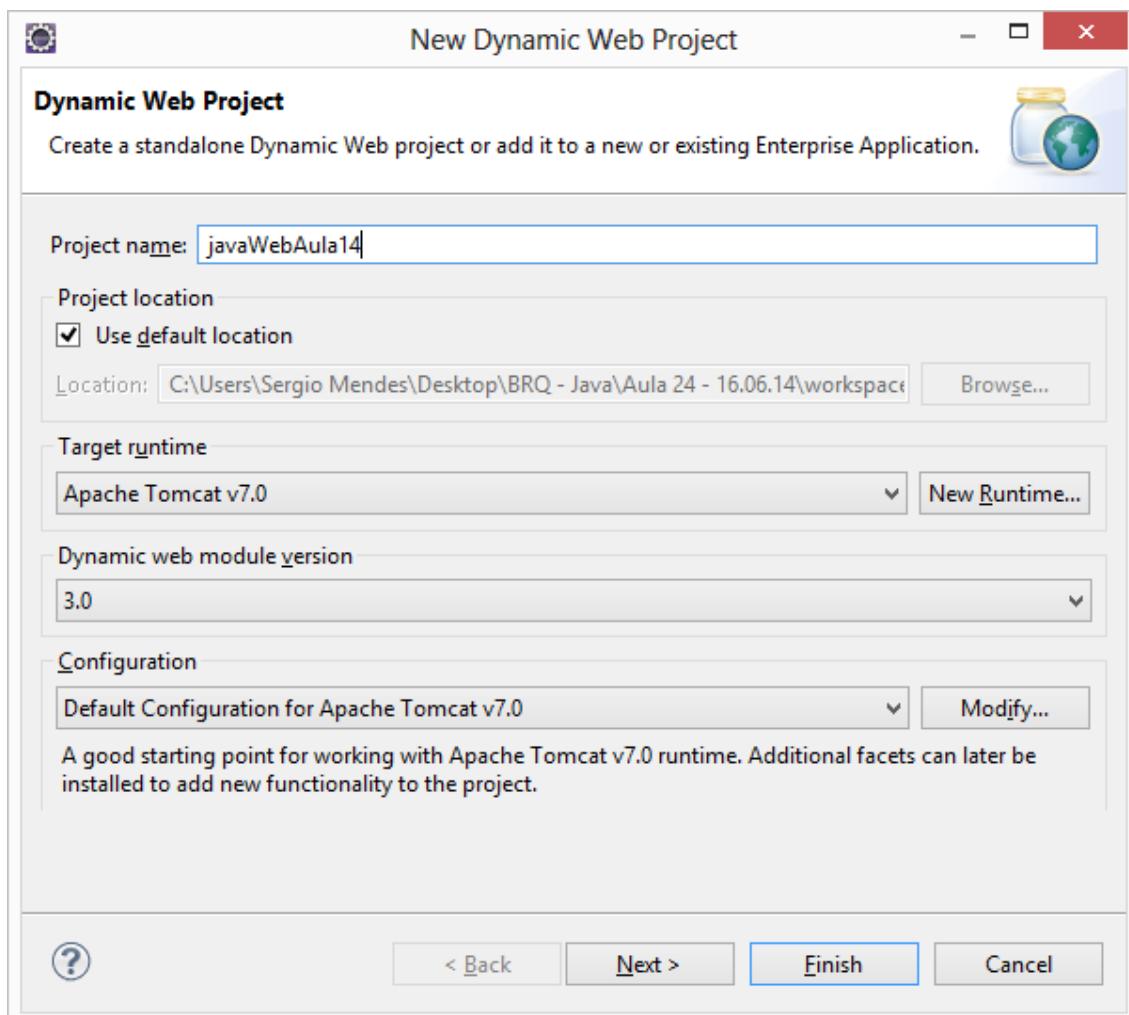
Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula

24

Novo projeto...



```
drop database if exists aulaweb14;
create database aulaweb14;
use aulaweb14;

create table usuario(
    idusuario      integer          auto_increment,
    nome           varchar(50)       not null,
    login          varchar(50)       not null unique,
    senha          varchar(50)       not null,
    foto           varchar(50)       not null,
    primary key(idusuario));

show tables;

desc usuario;
```



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> create database aulaweb14;
Query OK, 1 row affected (0.00 sec)

mysql> use aulaweb14;
Database changed
mysql>
mysql> create table usuario(
    -> idusuario          integer          auto_increment,
    -> nome               varchar(50)      not null,
    -> login              varchar(50)      not null unique,
    -> senha              varchar(50)      not null,
    -> foto               varchar(50)      not null,
    -> primary key(idusuario));
Query OK, 0 rows affected (0.57 sec)

mysql>
mysql> show tables;
+-----+
| Tables_in_aulaweb14 |
+-----+
| usuario            |
+-----+
1 row in set (0.00 sec)

mysql>
mysql> desc usuario;
+-----+-----+-----+-----+-----+-----+
| Field   | Type    | Null | Key  | Default | Extra       |
+-----+-----+-----+-----+-----+-----+
| idusuario | int(11) | NO   | PRI  | NULL    | auto_increment
| nome      | varchar(50)| NO  |      | NULL    |
| login     | varchar(50)| NO  | UNI  | NULL    |
| senha     | varchar(50)| NO  |      | NULL    |
| foto      | varchar(50)| NO  |      | NULL    |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.02 sec)

mysql>
```

```
package entity;

public class Usuario {

    private Integer idUsuario;
    private String nome;
    private String login;
    private String senha;
    private String foto;

    public Usuario() {
    }

    public Usuario(Integer idUsuario, String nome,
                   String login, String senha, String foto) {
        super();
        this.idUsuario = idUsuario;
        this.nome = nome;
        this.login = login;
        this.senha = senha;
        this.foto = foto;
    }
}
```



```
@Override
public String toString() {
    return "Usuario [idUsuario=" + idUsuario + ", nome="
           + nome + ", login=" + login + ", senha=" + senha
           + ", foto=" + foto + "]";
}

public Integer getIdUsuario() {
    return idUsuario;
}

public void setIdUsuario(Integer idUsuario) {
    this.idUsuario = idUsuario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getLogin() {
    return login;
}

public void setLogin(String login) {
    this.login = login;
}

public String getSenha() {
    return senha;
}

public void setSenha(String senha) {
    this.senha = senha;
}

public String getFoto() {
    return foto;
}

public void setFoto(String foto) {
    this.foto = foto;
}

}
```



## JPA - Java Persistence API

Mapeamento da Classe em relação à sua respectiva entidade da base de dados (ORM)

```
package entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "usuario")
@NamedQueries(
{
    @NamedQuery(name="usuario.autenticar",
        query="select u from Usuario as u where
            u.login = :login and u.senha = :senha"),
    @NamedQuery(name="usuario.verificarlogin",
        query="select count(u) from Usuario as u where
            u.login = :login")
}
)
public class Usuario {

    @Id // chave primária
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idusuario")
    private Integer idUsuario;

    @Column(name = "nome", length = 50, nullable = false)
    private String nome;

    @Column(name = "login", length = 50, nullable = false,
        unique = true)
    private String login;

    @Column(name = "senha", length = 50, nullable = false)
    private String senha;

    @Column(name = "foto", length = 50, nullable = false)
    private String foto;

    public Usuario() {
        // Construtor default
    }
}
```



```
public Usuario(Integer idUsuario, String nome, String login,
               String senha, String foto) {
    super();
    this.idUsuario = idUsuario;
    this.nome = nome;
    this.login = login;
    this.senha = senha;
    this.foto = foto;
}

@Override
public String toString() {
    return "Usuario [idUsuario=" + idUsuario + ", nome="
           + nome + ", login=" + login + ", senha=" + senha
           + ", foto=" + foto + "]";
}

public Integer getIdUsuario() {
    return idUsuario;
}

public void setIdUsuario(Integer idUsuario) {
    this.idUsuario = idUsuario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getLogin() {
    return login;
}

public void setLogin(String login) {
    this.login = login;
}

public String getSenha() {
    return senha;
}

public void setSenha(String senha) {
    this.senha = senha;
}
```



```
public String getFoto() {  
    return foto;  
}  
  
public void setFoto(String foto) {  
    this.foto = foto;  
}  
}
```

---

### hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
    <session-factory>  
  
        <property name="hibernate.dialect">  
            org.hibernate.dialect.MySQLDialect  
        </property>  
  
        <property name="hibernate.connection.driver_class">  
            com.mysql.jdbc.Driver  
        </property>  
  
        <property name="hibernate.connection.url">  
            jdbc:mysql://localhost:3306/aulaweb14  
        </property>  
  
        <property name="hibernate.connection.username">root</property>  
        <property name="hibernate.connection.password"></property>  
  
        <property name="hibernate.show_sql">true</property>  
        <property name="hibernate.format_sql">true</property>  
  
        <mapping class="entity.Usuario"/>  
  
    </session-factory>  
</hibernate-configuration>
```

---

### HibernateUtil.java

```
package hibernate;  
  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.AnnotationConfiguration;
```



```
public class HibernateUtil {  
    private static final SessionFactory sessionFactory;  
  
    static {  
        try {  
  
            sessionFactory = new AnnotationConfiguration().  
                configure("config/mysql_hibernate.cfg.xml").  
                buildSessionFactory();  
        } catch (Throwable ex) {  
  
            System.err.println("Initial SessionFactory  
creation failed." + ex);  
            throw new ExceptionInInitializerError(ex);  
        }  
    }  
  
    public static SessionFactory getSessionFactory() {  
        return sessionFactory;  
    }  
}
```

---

## Classe de persistência para Usuario...

```
package persistence;  
  
import hibernate.HibernateUtil;  
  
import org.hibernate.Query;  
import org.hibernate.Session;  
import org.hibernate.Transaction;  
  
import entity.Usuario;  
  
public class UsuarioDao {  
  
    private Session session;  
    //conexão gerada pela sessionFactory  
    private Transaction transaction; //transações (commmit)  
    private Query query;  
    //executar consultas em HQL - Hibernate Query Language  
  
    //Método para gravar um novo Usuario  
    public void create(Usuario u) throws Exception{  
        session = HibernateUtil.  
            getSessionFactory().openSession();  
        transaction = session.beginTransaction();  
        session.save(u);  
        transaction.commit();  
    }
```



```
        session.close();
    }

    //Método para autenticar o usuario pelo Login e Senha
    public Usuario authenticate(String login, String senha)
        throws Exception{
        session = HibernateUtil.
            getSessionFactory().openSession();

        query = session.getNamedQuery("usuario.autenticar");
        //executando a query mapeada
        query.setString("login", login);
        //parametro -> :login
        query.setString("senha", senha);
        //parametro -> :senha
        Usuario u = (Usuario) query.uniqueResult();
        //retorna 1 unico registro ou null

        session.close();
        return u; //retornar o usuario
    }

    //Método para verificar se um login informado ja
    //existe na base de dados
    public Boolean hasLogin(String login) throws Exception{
        session = HibernateUtil.
            getSessionFactory().openSession();

        query = session.getNamedQuery
            ("usuario.verificarlogin");
        query.setString("login", login);
        //toda query count no hibernate retorna um
        //inteiro do tipo long
        Long qtd = (Long) query.uniqueResult();

        session.close();
        return qtd > 0;
    }
}
```

---

## Criando o Controle para Usuario

```
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
```



```
import org.apache.struts.action.ActionMapping;

public class ControleUsuario extends Action {
    @Override
    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {
        return super.execute(mapping, form, request, response);
    }
}
```

## WEB-INF/struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>

    <form-beans>
        <form-bean name="" type="" />
    </form-beans>

    <action-mappings>

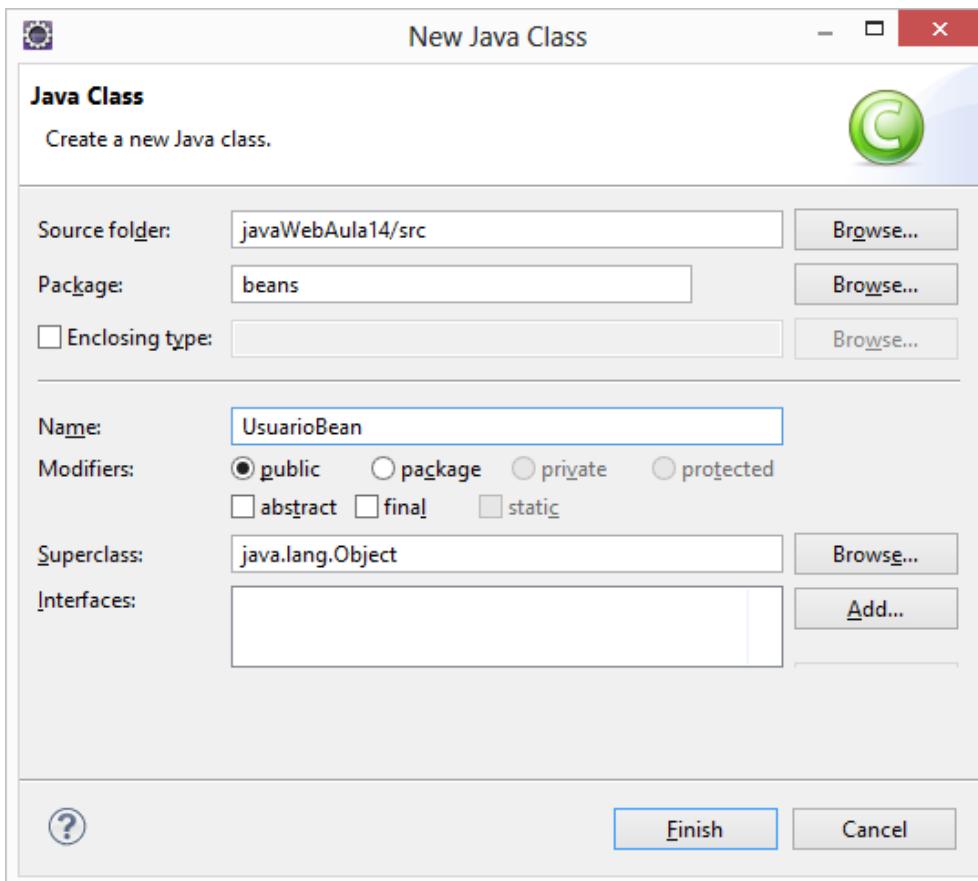
        <action path="/ControleUsuario" scope="request"
            type="control.ControleUsuario">
        </action>

    </action-mappings>

    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
            value="/WEB-INF/tiles-defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames"
            value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
    </plug-in>
    </struts-config>
```



### Classe de Modelo de dados no Struts...



```
package beans;

import org.apache.struts.action.ActionForm;

import entity.Usuario;

public class UsuarioBean extends ActionForm {

    private Usuario usuario; // javaBean

    public UsuarioBean() {
        usuario = new Usuario(); // espaço de memória
    }

    public Usuario getUsuario() {
        return usuario;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }
}
```



## WEB-INF/struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>

    <form-beans>
        <form-bean name="UsuarioBean"
                    type="beans.UsuarioBean"/>
    </form-beans>

    <action-mappings>

        <action path="/ControleUsuario"
                name="UsuarioBean"
                scope="request"
                type="control.ControleUsuario">
            </action>
    </action-mappings>

    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
                     value="/WEB-INF/tiles-defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames" value="/WEB-INF/validation-rules.xml,/WEB-INF/validation.xml"/>
    </plug-in>
</struts-config>
```



---

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>
    <head>
        <style type="text/css">
            body { font-family: verdana;
                    font-size: 9pt; padding: 30px; }
        </style>
    </head>
```



# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula  
24

```
<body>

    <h3>Bem vindo ao Projeto</h3>
    <hr/>

    <a href="Login.jsp">Acessar Sistema</a>

</body>

</html>
```

---

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-Logic"
    prefix="logic" %>

<html>

    <head>
        <style type="text/css">
            body { font-family: verdana; font-size: 9pt;
                padding: 30px; }
        </style>
    </head>

    <body>

        <h3>Autenticação de Usuários</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <html:form method="post"
            action="ControleUsuario.do?cmd=autenticar">

            Login de Acesso:
            <html:text property="usuario.Login"/>

            Senha:
            <html:password property="usuario.senha"/>

            <html:submit value="Acessar Sistema"/>

            <p>
                <bean:write name="msg" ignore="true"/>
            </p>
    
```



# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula  
24

```
<p>
    Ainda não possui conta de acesso?
    <a href="#">Clique aqui</a>
</p>

</html:form>

</body>

</html>
```

Executando...

## index.jsp

Bem vindo ao Projeto

[Acessar Sistema](#)

## login.jsp

Autenticação de Usuários

[Voltar](#)

Login de Acesso:  Senha:

Ainda não possui conta de acesso? [Clique aqui](#)



Criando o formulário de cadastro...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="logic" %>

<html>

    <head>

        <link rel="stylesheet"
            href="css/redmond/jquery-ui-1.10.4.custom.css"
            type="text/css"/>

        <script type="text/javascript"
            src="js/jquery-1.10.2.js"></script>

        <script type="text/javascript"
            src="js/jquery-ui-1.10.4.custom.js"></script>

        <style type="text/css">
            body { font-family: verdana;
                    font-size: 9pt;
                    padding: 30px; }
        </style>

    </head>

    <body>

        <h3>Autenticação de Usuários</h3>
        <a href="index.jsp">Voltar</a>
        <hr/>

        <html:form method="post"
            action="ControleUsuario.do?cmd=autenticar">

            Login de Acesso:
            <html:text property="usuario.Login"/>

            Senha:
            <html:password property="usuario.senha"/>

            <html:submit value="Acessar Sistema"/>

    </body>

```



# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula  
24

```
<p>
    <bean:write name="msg" ignore="true"/>
</p>

<p>
    Ainda não possui conta de acesso?
    <a href="#">Clique aqui</a>
</p>

</html:form>

<div id="janela">

    <!-- Todo formulário que faz upload de arquivo deve ter o parametro => enctype -->

    <html:form method="post"
    action="ControleUsuario.do?cmd=cadastrar"
    enctype="multipart/form-data">

        Nome do Usuário: <br/>
        <html:text property="usuario.nome"/>
        <br/><br/>

        Login de Acesso: <br/>
        <html:text property="usuario.Login"/>
        <br/><br/>

        Senha: <br/>
        <html:password property="usuario.senha"/>
        <br/><br/>

        Foto: <br/>
        <html:file property="imagem"/>
        <br/><br/>

        <html:submit value="Cadastrar Novo Usuário"/>

    </html:form>

</div>

</body>

</html>
```



## FormFile

Classe do struts para armazenamento de arquivos utilizados para upload na aplicação.

```
package beans;

import org.apache.struts.action.ActionForm;
import org.apache.struts.upload.FormFile;

import entity.Usuario;

public class UsuarioBean extends ActionForm {

    private Usuario usuario;
    // javaBean
    private FormFile imagem;
    // atributo para capturar o arquivo 'upado'

    public UsuarioBean() {
        usuario = new Usuario(); // espaço de memória
    }

    public Usuario getUsuario() {
        return usuario;
    }

    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }

    public FormFile getImagem() {
        return imagem;
    }

    public void setImagem(FormFile imagem) {
        this.imagem = imagem;
    }
}
```



# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula  
24

Autenticação de Usuários

Voltar

Login de Acesso:  Senha:  Acessar Sistema

Ainda não possui conta de acesso? [Clique aqui](#)

Nome do Usuário:

Login de Acesso:

Senha:

Foto:

Escolher arquivo Nenhum arquivo selecionado

Cadastrar Novo Usuário



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>

<html>

    <head>

        <link rel="stylesheet"
              href="css/redmond/jquery-ui-1.10.4.custom.css"
              type="text/css"/>

        <script type="text/javascript"
               src="js/jquery-1.10.2.js"></script>

        <script type="text/javascript"
               src="js/jquery-ui-1.10.4.custom.js"></script>
```



# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula  
24

```
<style type="text/css">
    body { font-family: verdana; font-size: 9pt;
           padding: 30px; }
</style>

<script type="text/javascript">

    $(document).ready( //documento estiver pronto...
        function(){ //faça...

            $("#janela").dialog( { width: '400',
                                   modal : true, autoOpen : false } );

            //criando o evento...
            $("#link").click( //ao clicar no link...
                function(){ //faça...
                    $("#janela").dialog("open");
                    //exibir a janela de diálogo
                }
            );
        }
    );
}

</script>

</head>

<body>

    <h3>Autenticação de Usuários</h3>
    <a href="index.jsp">Voltar</a>
    <hr/>

    <html:form method="post"
               action="ControleUsuario.do?cmd=autenticar">

        Login de Acesso:
        <html:text property="usuario.login"/>

        Senha:
        <html:password property="usuario.senha"/>

        <html:submit value="Acessar Sistema"/>

        <p>
            <bean:write name="msg" ignore="true"/>
        </p>

        <p>
            Ainda não possui conta de acesso?
            <a href="#" id="Link">Clique aqui</a>
        </p>
    </html:form>
</body>
```



```
</p>

</html:form>

<div id="janela" title="Formulário de Cadastro">

    <!-- Todo formulário que faz upload de arquivo deve ter o parametro => enctype -->
    <html:form method="post"
        action="ControleUsuario.do?cmd=cadastrar"
        enctype="multipart/form-data">

        Nome do Usuário: <br/>
        <html:text property="usuario.nome"/>
        <br/><br/>

        Login de Acesso: <br/>
        <html:text property="usuario.login"/>
        <br/><br/>

        Senha: <br/>
        <html:password property="usuario.senha"/>
        <br/><br/>

        Foto: <br/>
        <html:file property="imagem"/>
        <br/><br/>

        <html:submit value="Cadastrar Novo Usuário"/>

    </html:form>

</div>

</body>

</html>
```





# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula  
**24**

The screenshot shows a web browser window with the URL `localhost:8081/javaWebAula14/login.jsp#`. The page title is "Autenticação de Usuários" and includes a "Voltar" link. Below it is a login form with fields for "Login de Acesso" and "Senha", and a "Acessar Sistema" button. A link "Ainda não possui conta de acesso? Clique aqui" is present. Overlaid on the page is a modal dialog titled "Formulário de Cadastro". It contains fields for "Nome do Usuário", "Login de Acesso", "Senha", and a file upload field for "Foto" with the placeholder "Escolher arquivo Nenhum arquivo selecionado". A "Cadastrar Novo Usuário" button is at the bottom of the modal. The browser's taskbar at the bottom shows various icons and the date/time "16/06/2014 14:13".

## Arquivo de mensagens do Struts MessageResources.properties

The screenshot shows the NetBeans IDE interface with a "New File" dialog open. The "File" tab is selected, and the "Create a new file resource." instruction is visible. In the "Enter or select the parent folder:" field, the path `javaWebAula14/src` is entered. The "File name:" field contains `MessageResources.properties`. At the bottom of the dialog are buttons for "?", "< Back" (disabled), "Next >" (disabled), "Finish", and "Cancel".



### MessageResources.properties

```
erro.nome=Por favor, informe um nome de usuário válido.  
erro.login=Por favor, informe um login de usuário válido.  
erro.senha=Por favor, informe uma senha válida.  
erro.imagem=Por favor, selecione uma imagem válida.
```

-----

Método para validação de dados do Struts utilizado nas classes ActionForm (Camada de Modelo)

```
@Override  
public ActionErrors validate(ActionMapping mapping,  
                             HttpServletRequest request) {  
  
    return super.validate(mapping, request);  
}
```

-----

```
package beans;  
  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
  
import javax.servlet.http.HttpServletRequest;  
  
import org.apache.struts.action.ActionErrors;  
import org.apache.struts.action.ActionForm;  
import org.apache.struts.action.ActionMapping;  
import org.apache.struts.action.ActionMessage;  
import org.apache.struts.upload.FormFile;  
  
import entity.Usuario;  
  
public class UsuarioBean extends ActionForm {  
  
    private Usuario usuario; // javaBean  
    private FormFile imagem;  
    // atributo para capturar o arquivo 'upado'  
  
    public UsuarioBean() {  
        usuario = new Usuario(); // espaço de memória  
    }
```



```
//Métpdp para validar os dados do UsuarioBean
@Override
public ActionErrors validate(ActionMapping mapping,
    HttpServletRequest request) {

    //Classe do Struts utilizada para gerar erros de validação
    ActionErrors erros = new ActionErrors();

    Pattern regexNome = Pattern.compile
        ("^[A-Za-zÀ-Ùà-ü\\s]{6,50}$");

    if(usuario.getNome() != null){
        //se nome do usuario não está vazio

        Matcher m = regexNome.matcher(usuario.getNome());
        //aplicando o regex ao atributo nome

        if( ! m.matches()){
            //se o regex não aprovou o nome do usuario
            erros.add("nomeinvalido", new ActionMessage("erro.nome"));
        }
    }

    //retornando o objeto que contem os erros do struts
    return erros;
}

public Usuario getUsuario() {
    return usuario;
}

public void setUsuario(Usuario usuario) {
    this.usuario = usuario;
}

public FormFile getImagen() {
    return imagem;
}

public void setImagen(FormFile imagem) {
    this.imagem = imagem;
}

}
```



## WEB-INF/struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>

<form-beans>
    <form-bean name="UsuarioBean" type="beans.UsuarioBean"/>
</form-beans>

<action-mappings>

    <action input="/Login.jsp" validate="true"
            path="/ControleUsuario"
            name="UsuarioBean" scope="request"
            type="control.ControleUsuario">
        </action>
    </action-mappings>

    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
                    value="/WEB-INF/tiles-defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames" value="/WEB-INF/validation-rules.xml,/WEB-INF/validation.xml"/>
    </plug-in>
</struts-config>
```

---

```
-----
package control;

import java.io.File;
import java.io.FileOutputStream;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
```



```
import org.apache.struts.action.ActionMapping;

import persistence.UsuarioDao;
import beans.UsuarioBean;
import entity.Usuario;

public class ControleUsuario extends Action {

    @Override
    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("autenticar")){
            return autenticar(mapping, form, request, response);
        }

        return super.execute(mapping, form, request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {

        try{

            //Resgatar o UsuarioBean
            UsuarioBean ub = (UsuarioBean) form;
            UsuarioDao d = new UsuarioDao();

            if( ! d.hasLogin(ub.getUsuario().getLogin()) ){
                //se login não existe

                String path = "C:\\\\Users\\\\Sergio
Mendes\\\\Desktop\\\\BRQ - Java\\\\Aula 24 -
16.06.14\\\\workspace\\\\javaWebAula14\\\\
WebContent\\\\img\\\\";

                String nomeFoto = ub.getUsuario().getLogin()
                    + ".jpg";

                //Upload do arquivo
                File f = new File(path + nomeFoto);
                //caminho e o local do arquivo que
                //será gravado
            }
        }
    }
}
```



```
        FileOutputStream out = new
        FileOutputStream(f); //gravação do arquivo
        out.write(ub.getImagen().getFileData());

        //Upload
        out.flush();
        out.close();

        //setar o nome da foto do usuario
        ub.getUsuario().setFoto(nomeFoto);

        //cadastrando o usuario
        d.create(ub.getUsuario());

        request.setAttribute("msg", "Usuario
        cadastrado com sucesso.");

        ub.setUsuario(new Usuario()); //limpar os
        //dados do UsuarioBean
    }
    else{
        throw new Exception("Login já encontra-se
        em uso, por favor tente outro.");
    }
}
catch(Exception e){
    request.setAttribute("msg", e.getMessage());
}

return mapping.findForward("login");
//forward (redirecionamento)
}

public ActionForward autenticar(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
throws Exception {

    return super.execute(mapping, form, request, response);
}
}
```



# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula  
24

**Autenticação de Usuários**

[Voltar](#)

Login de Acesso:  Senha:  Acessar Sistema

Ainda não possui conta de acesso? [Clique aqui](#)

**Formulário de Cadastro**

Nome do Usuário:

Login de Acesso:

Senha:

Foto:

**Autenticação de Usuários**

[Voltar](#)

Login de Acesso:  Senha:  Acessar Sistema

Usuario cadastrado com sucesso.

Ainda não possui conta de acesso? [Clique aqui](#)



### Criando a ação de autenticação...

```
package control;

import java.io.File;
import java.io.FileOutputStream;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import persistence.UsuarioDao;
import beans.UsuarioBean;
import entity.Usuario;

public class ControleUsuario extends Action {

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd");

        if(cmd.equalsIgnoreCase("cadastrar")){
            return cadastrar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("autenticar")){
            return autenticar(mapping, form, request, response);
        }
        else if(cmd.equalsIgnoreCase("logout")){
            return logout(mapping, form, request, response);
        }

        return super.execute(mapping, form, request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
                                  ActionForm form, HttpServletRequest request,
                                  HttpServletResponse response)
        throws Exception {

        try{

            //Resgatar o UsuarioBean
            UsuarioBean ub = (UsuarioBean) form;
```



```
UsuarioDao d = new UsuarioDao();

if( ! d.hasLogin(ub.getUsuario().getLogin()) ){
    //se login não existe

    String path = "C:\\\\Users\\\\Sergio
Mendes\\\\Desktop\\\\BRQ - Java\\\\Aula 24 -
16.06.14\\\\workspace\\\\javaWebAula14\\\\
WebContent\\\\img\\\\";
    String nomeFoto = ub.getUsuario().getLogin()
        + ".jpg";

    //Upload do arquivo
    File f = new File(path + nomeFoto);
    //caminho e o local do arquivo que
    //será gravado

    FileOutputStream out = new
        FileOutputStream(f);
    //gravação do arquivo
    out.write(ub.getImagen().getFileData());
    //Upload
    out.flush();
    out.close();

    //setar o nome da foto do usuario
    ub.getUsuario().setFoto(nomeFoto);

    //cadastrando o usuario
    d.create(ub.getUsuario());

    request.setAttribute("msg", "Usuario
cadastrado com sucesso.");

    ub.setUsuario(new Usuario());
    //limpar os dados do UsuarioBean
}

else{
    throw new Exception("Login já encontra-se
em uso, por favor tente outro.");
}

}

catch(Exception e){
    request.setAttribute("msg", e.getMessage());
}

return mapping.findForward("login"); //forward
(redirecionamento)
}
```



```
public ActionForward autenticar(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
    throws Exception {

    try{

        UsuarioBean ub = (UsuarioBean) form;
        //classe de modelo do struts
        UsuarioDao d = new UsuarioDao();
        //classe de persistência

        //resgatar um usuario do banco de dados
        //através do login e senha
        Usuario u = d.authenticate(
            ub.getUsuario().getLogin(),
            ub.getUsuario().getSenha()));

        if( u != null ){ //se usuario foi encontrado!

            //Armazenar os dados do usuario em sessão
            //memória no navegador
            HttpSession session = request.getSession();
            session.setAttribute("usuario", u); //gravando
            o objeto usuario em sessão!

            return mapping.findForward("autenticado");
            //redirecionamento
        }
        else{
            throw new Exception("Acesso Negado.");
        }
    }
    catch(Exception e){
        request.setAttribute("msg", e.getMessage());
    }

    return mapping.findForward("login");
}

public ActionForward logout(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response)
    throws Exception {

    HttpSession session = request.getSession();
    session.removeAttribute("usuario");
    session.invalidate();

    request.setAttribute("msg", "Obrigado por usar o
        sistema.");
}
```



```
        return mapping.findForward("login");
    }
}
```

### Criando a página do administrador...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="Logic" %>

<html>
    <head>
        <style type="text/css">
            body { font-family: verdana; font-size: 9pt;
            padding: 30px; }
        </style>
    </head>
    <body>
        <h3>Bem vindo à Área restrita do Sistema</h3>
        <hr/>

        <p>
            <img src='/javaWebAula14/img/<bean:write
            name="usuario" property="foto"/>' width="200"/>
        </p>

        Nome do Usuário: <bean:write name="usuario"
            property="nome"/> <br/>

        Login de Acesso: <bean:write name="usuario"
            property="Login"/> <br/>

        <hr/>

        <p>
            <a href="ControleUsuario.do?cmd=Logout">
                Sair do Sistema
            </a>
        </p>
    </body>
</html>
```



# Java WebDeveloper - BRQ

Segunda-feira, 16 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA

Aula

24



Bem vindo à Área restrita do Sistema



Nome do Usuário: Power Ranger Verde  
Login de Acesso: rangerverde

[Sair do Sistema](#)



Bem vindo à Área restrita do Sistema



Nome do Usuário: Power Ranger Vermelho  
Login de Acesso: rangervermelho

[Sair do Sistema](#)



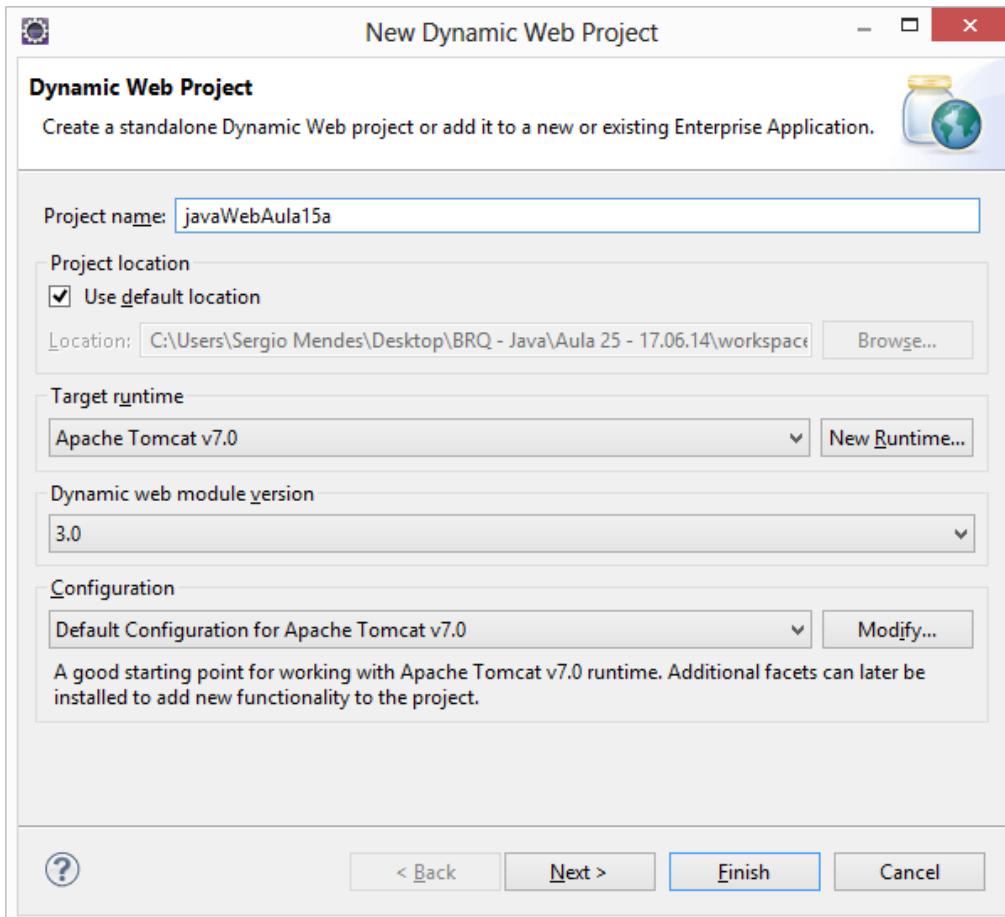
# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

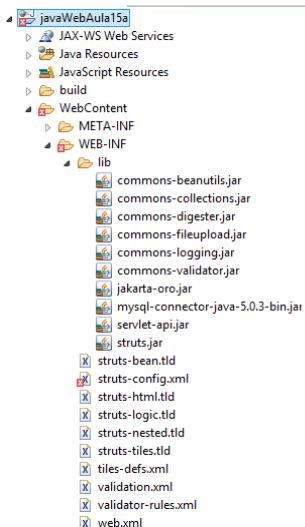
Aula  
25

Novo projeto...



- Criação de Layouts (Tiles)
- Validação

Incluindo o struts no projeto...



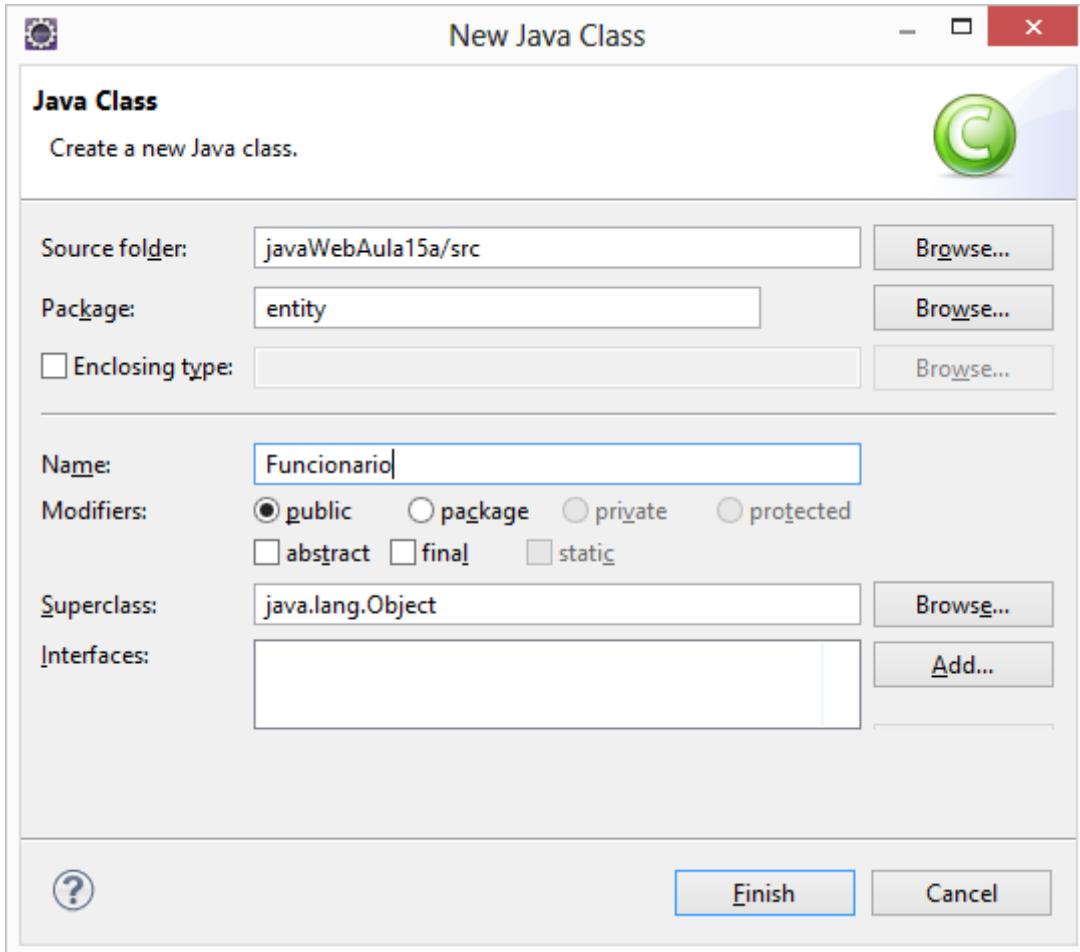


# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
**25**



```
package entity;

public class Funcionario {

    private String nome;
    private Double salario;

    public Funcionario() {
        // Construtor default
    }

    public Funcionario(String nome, Double salario) {
        super();
        this.nome = nome;
        this.salario = salario;
    }

    @Override
    public String toString() {
        return "Funcionario [nome=" + nome + ", salario=" +
               salario + "]";
    }
}
```



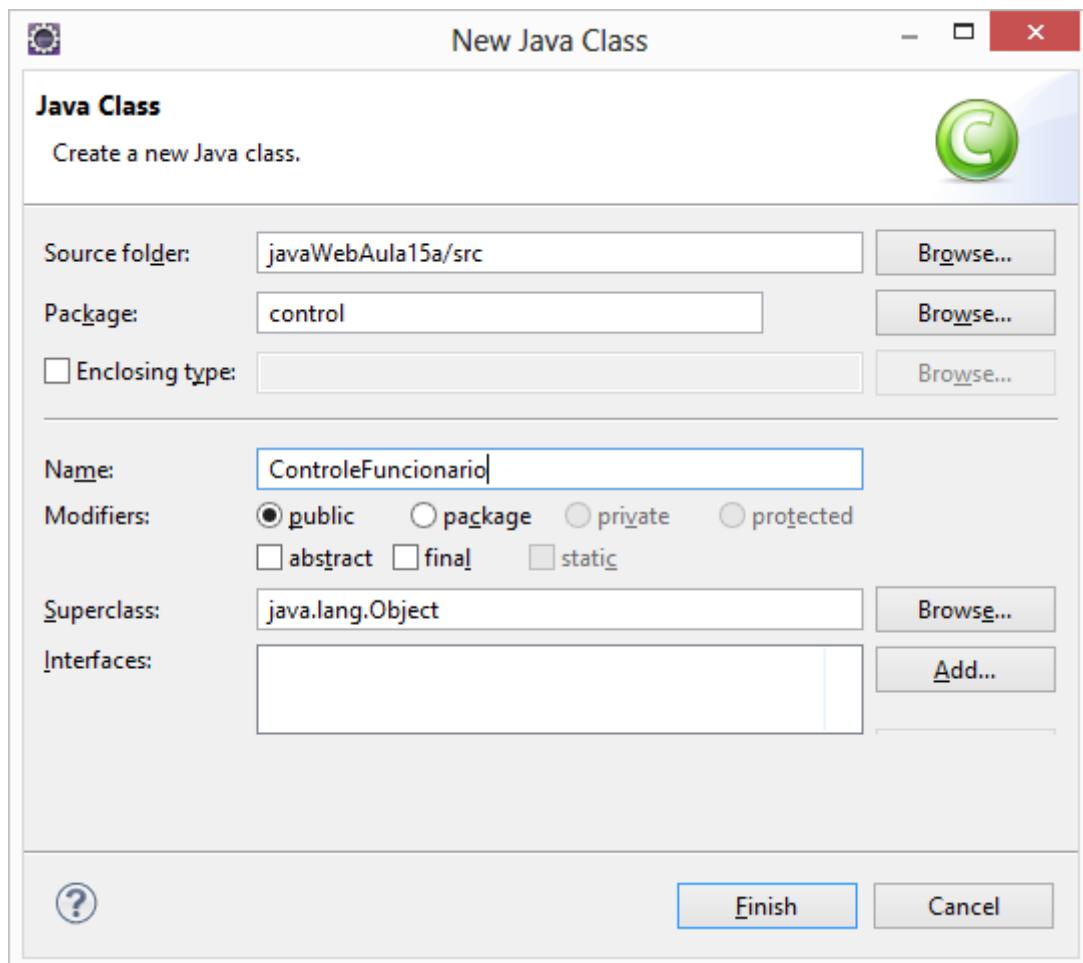
# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
**25**

```
public String getNome() {  
    return nome;  
}  
  
public void setNome(String nome) {  
    this.nome = nome;  
}  
  
public Double getSalario() {  
    return salario;  
}  
  
public void setSalario(Double salario) {  
    this.salario = salario;  
}  
}
```



```
package control;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;
```



```
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class ControleFuncionario extends Action{

    @Override
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form, HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        return super.execute(mapping, form, request, response);
    }
}
```

---

## Camada de Modelo de dados

```
package beans;

import org.apache.struts.action.ActionForm;

import entity.Funcionario;

public class FuncionarioBean extends ActionForm {

    // Atributo para armazenar os dados do funcionario
    private Funcionario funcionario;

    public FuncionarioBean() {
        //instancia (inicializa)
        funcionario = new Funcionario();
    }

    public Funcionario getFuncionario() {
        return funcionario;
    }

    public void setFuncionario(Funcionario funcionario) {
        this.funcionario = funcionario;
    }

}
```



## WEB-INF/struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-config_1_2.dtd">
<struts-config>

    <form-beans>
        <form-bean name="FuncionarioBean"
                    type="beans.FuncionarioBean"/>
    </form-beans>

    <action-mappings>

        <action path="/ControleFuncionario" scope="session"
               name="FuncionarioBean"
               type="control.ControleFuncionario">
            <forward path="/cadastro.jsp" name="sucesso"/>
        </action>
    </action-mappings>

    <controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
    <message-resources parameter="MessageResources"/>

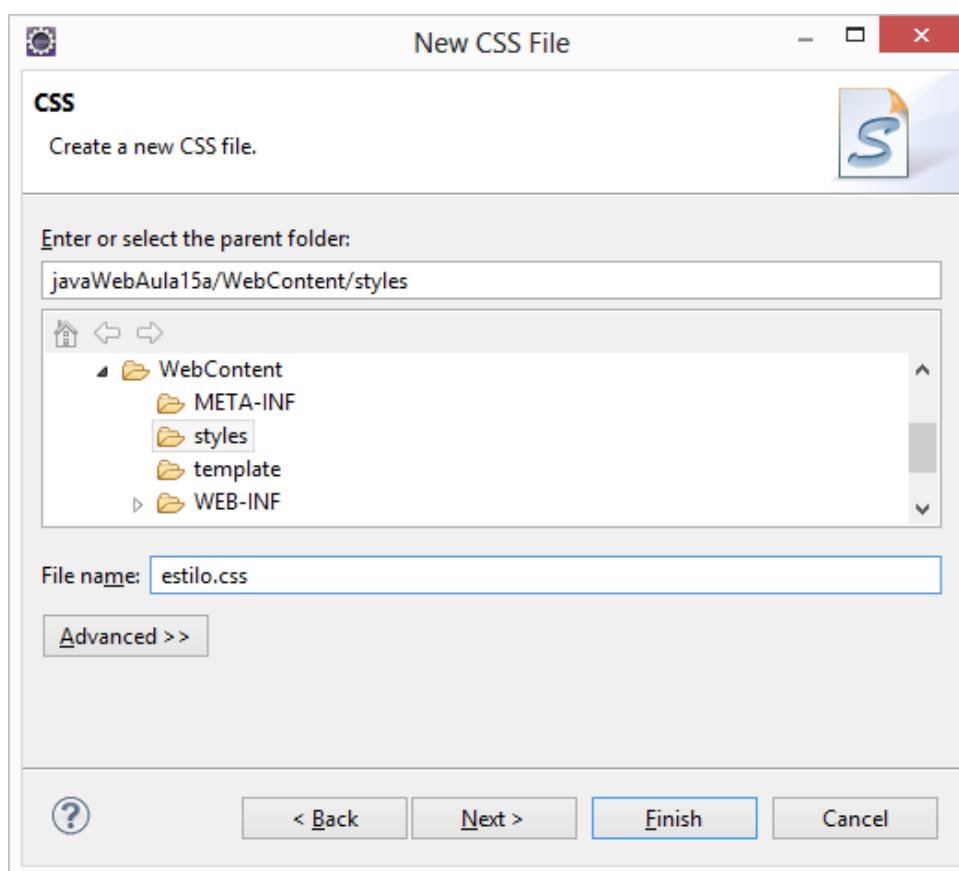
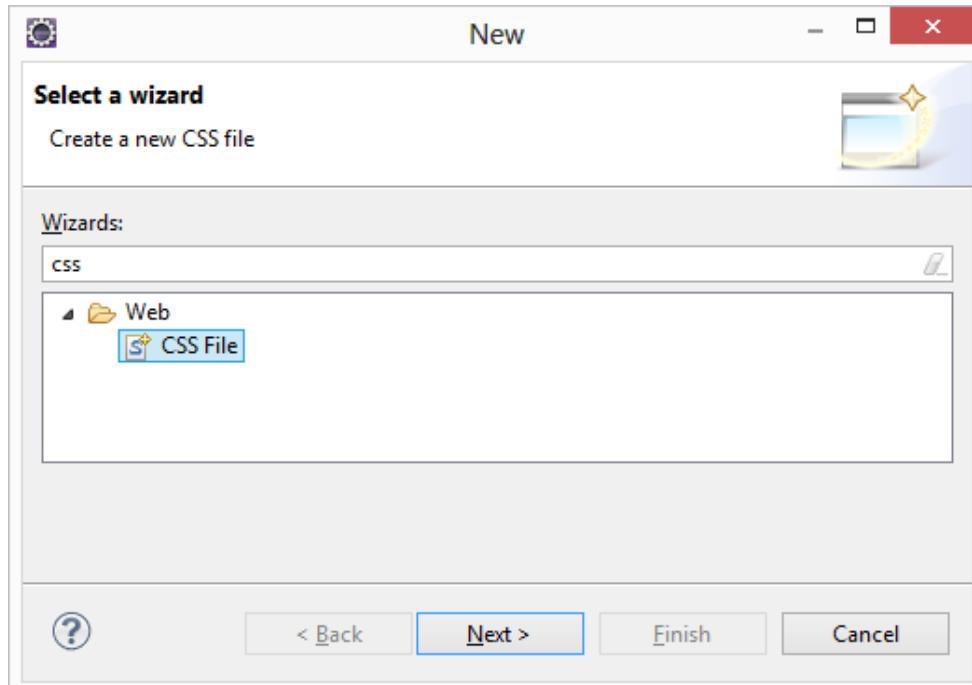
    <plug-in className="org.apache.struts.tiles.TilesPlugin">
        <set-property property="definitions-config"
                     value="/WEB-INF/tiles-defs.xml"/>
        <set-property property="moduleAware" value="true"/>
    </plug-in>

    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
    </plug-in>
</struts-config>
```



## Tiles (templates de páginas)

### 1) Criando o arquivo de folha de estilos





### /styles/estilo.css

```
body /* formatação da tag <body> da página */
{
    font-family: verdana;          /* tipo da fonte */
    font-size: 9pt;                /* tamanho da fonte */
    padding: 40px;                 /* margem interna */
    background-color: #CCC;        /* cor de fundo */
}

.topo /* estilo definido por 'class' */
{
    width: 0 auto;                /* largura dinamica */
    height: 0 auto;               /* altura dinamica */
    background-color: #EAEAEA;     /* cor de fundo */
    padding: 20px;                 /* margem interna */
}

.menu /* estilo definido por 'class' */
{
    width: 0 auto;                /* largura dinamica */
    height: 0 auto;               /* altura dinamica */
    background-color: #FAFAFA;     /* cor de fundo */
    padding: 20px;                 /* margem interna */
}

.pagina /* estilo definido por 'class' */
{
    width: 0 auto;                /* largura dinamica */
    height: 0 auto;               /* altura dinamica */
    background-color: #FFF;        /* cor de fundo */
    padding: 20px;                 /* margem interna */
}
```

---

### /template/topo.jsp

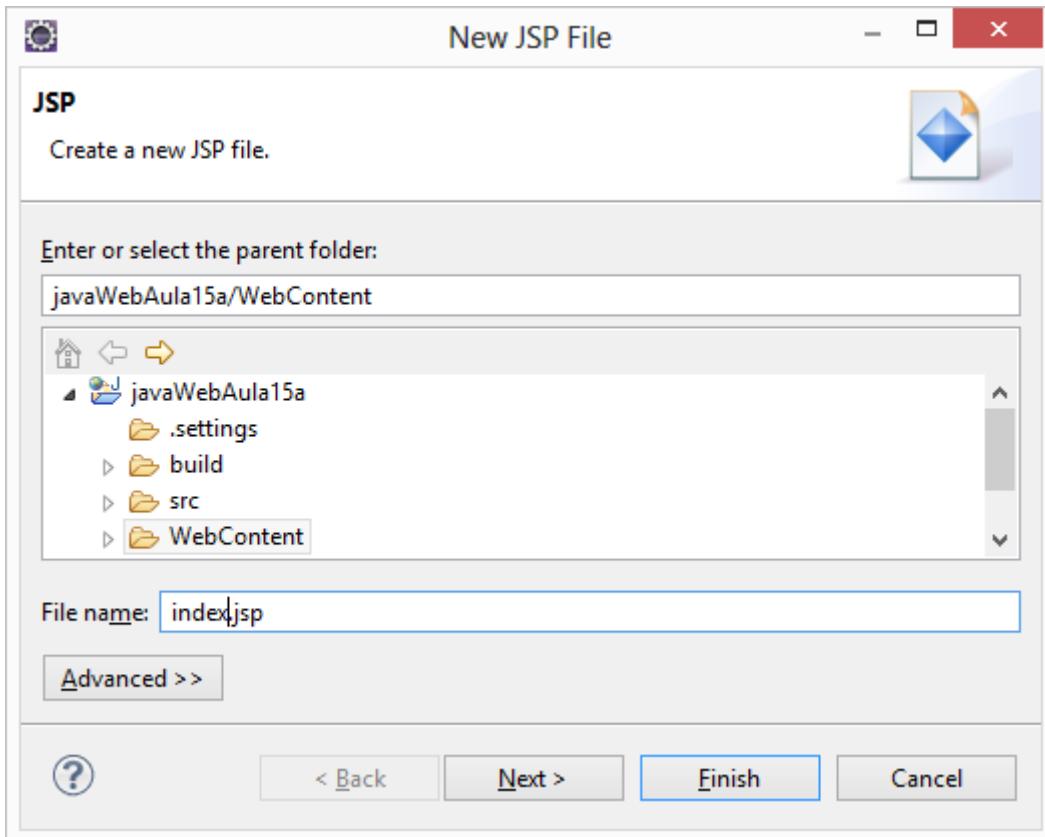
```
<h3>Projeto Struts - Controle de Funcionários</h3>
Tecnologias: Struts Tiles, Validação e Sessão
```

### /template/menu.jsp

```
<ul>
    <li> <a href="cadastro.jsp">Cadastrar novo Funcionário</a> </li>
    <li> <a href="consulta.jsp">Consultar Funcionários</a> </li>
</ul>
```



### Página inicial do projeto... /index.jsp



Montando o conteúdo...

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="Logic" %>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
    prefix="tiles" %>

<html>

    <head>
        <link rel="stylesheet" href="styles/estilo.css"
            type="text/css"/>
    </head>
```



```
<body>

    <div class="topo">
        <tiles:insert page="/template/topo.jsp"/>
    </div>

    <div class="menu">
        <tiles:insert page="/template/menu.jsp"/>
    </div>

    <div class="pagina">

        Bem vindo ao projeto.

    </div>

</body>

</html>
```

## /cadastro.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="logic" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
    prefix="tiles" %>

<html>

    <head>
        <link rel="stylesheet" href="styles/estilo.css"
            type="text/css"/>
    </head>

    <body>

        <div class="topo">
            <tiles:insert page="/template/topo.jsp"/>
        </div>

        <div class="menu">
            <tiles:insert page="/template/menu.jsp"/>
        </div>

    </body>
```



# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
**25**

```
<div class="pagina">

    <html:form method="post"
    action="ControleFuncionario.do?cmd=cadastrar">
        <p>
            Para cadastrar um novo funcionário,
            preencha os campos abaixo:
        </p>

        Nome do Funcionário: <br/>
        <html:text property="funcionario.nome"/>
        <br/><br/>

        Salário: <br/>
        <html:text property="funcionario.salario"/>
        <br/><br/>

        <html:submit value="Cadastrar Funcionário"/>

        <p>
            <bean:write name="msg" ignore="true"/>
        </p>
    </html:form>
</div>
</body>
</html>
```

localhost:8084/javaWebAula15a/cadastro.jsp

Projeto Struts - Controle de Funcionários

Tecnologias: Struts Tiles, Validação e Sessão

- [Cadastrar novo Funcionário](#)
- [Consultar Funcionários](#)

Para cadastrar um novo funcionário, preencha os campos abaixo:

Nome do Funcionário:

Salário:

Cadastrar Funcionário

09:15 17/06/2014



### Validação dos campos do formulário...

```
package beans;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

import entity.Funcionario;

public class FuncionarioBean extends ActionForm {

    // Atributo para armazenar os dados do funcionario
    private Funcionario funcionario;

    public FuncionarioBean() {
        //instancia (inicializa)
        funcionario = new Funcionario();
    }

    //Método sobreescrito da Classe ActionForm utilizado
    //para validação dos dados
    @Override
    public ActionErrors validate(ActionMapping mapping,
                                HttpServletRequest request) {

        //Classe para geração de erros no struts
        ActionErrors erros = new ActionErrors();
        //vazia (nenhum erro!)

        return erros;
    }

    public Funcionario getFuncionario() {
        return funcionario;
    }

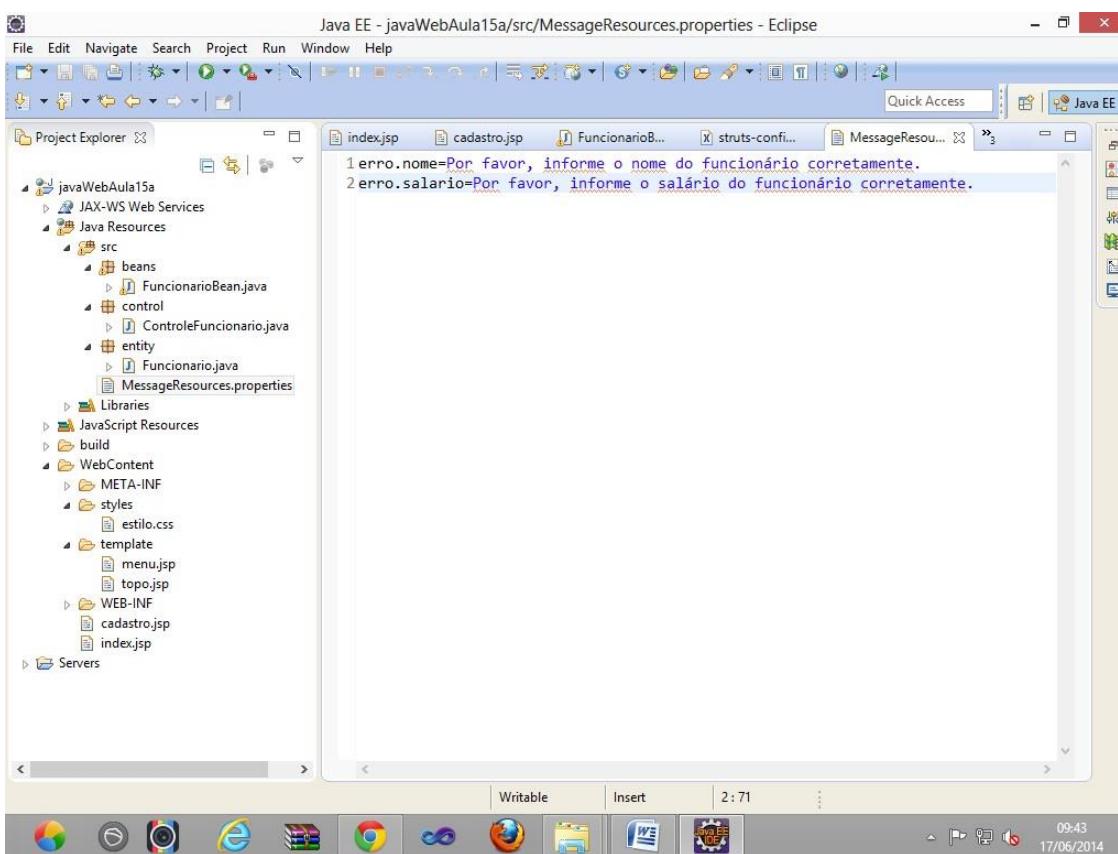
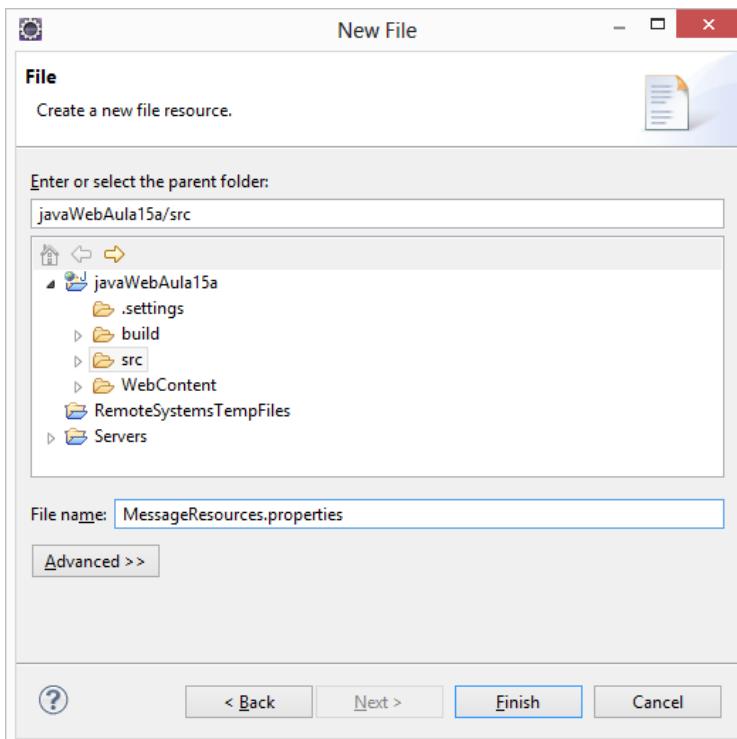
    public void setFuncionario(Funcionario funcionario) {
        this.funcionario = funcionario;
    }

}
```



## MessageResources.properties

Arquivo de mensagens padrão do struts





erro.nome=Por favor, informe o nome do funcionário corretamente.  
erro.salario=Por favor, informe o salário do funcionário corretamente.

---

## Validando o nome do funcionários

```
package beans;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;

import entity.Funcionario;

public class FuncionarioBean extends ActionForm {

    // Atributo para armazenar os dados do funcionario
    private Funcionario funcionario;

    public FuncionarioBean() {
        //instancia (inicializa)
        funcionario = new Funcionario();
    }

    //Método sobrescrito da Classe ActionForm utilizado para
    //validação dos dados
    @Override
    public ActionErrors validate(ActionMapping mapping,
                                HttpServletRequest request) {

        //Classe para geração de erros no struts
        ActionErrors erros = new ActionErrors();
        //vazia (nenhum erro!)

        //Validação do nome do funcionário
        if(funcionario.getNome() != null){

            Pattern regex = Pattern.compile
                ("^[A-Zá-zÀ-Ùà-ü\\s]{3,30}");
            //expressão de validação

            Matcher m = regex.matcher(funcionario.getNome());
            //aplicando a expressão ao nome

            if( ! m.matches() ){

```



```
//se o nome do funcionario NÃO passou no regex
erros.add("nomeinvalido", new
           ActionMessage("erro.nome"));
    }
}

return erros;
}

public Funcionario getFuncionario() {
    return funcionario;
}

public void setFuncionario(Funcionario funcionario) {
    this.funcionario = funcionario;
}
}

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD
Struts Configuration 1.2//EN" "http://struts.apache.org/dtds/struts-
config_1_2.dtd">
<struts-config>

<form-beans>
    <form-bean name="FuncionarioBean"
                type="beans.FuncionarioBean"/>
</form-beans>

<action-mappings>

    <action path="/ControleFuncionario"
            scope="session"
            input="/cadastro.jsp"
            validate="true"
            name="FuncionarioBean"
            type="control.ControleFuncionario">
        <forward path="/cadastro.jsp" name="sucesso"/>
    </action>
</action-mappings>

<controller
processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
<message-resources parameter="MessageResources"/>
```



# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
**25**

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
    <set-property property="definitions-config" value="/WEB-INF/tiles-
defs.xml"/>
    <set-property property="moduleAware" value="true"/>
</plug-in>
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-
rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
</struts-config>
```

---

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
prefix="logic" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
prefix="tiles" %>

<html>

    <head>
        <link rel="stylesheet" href="styles/estilo.css"
        type="text/css"/>
    </head>

    <body>

        <div class="topo">
            <tiles:insert page="/template/topo.jsp"/>
        </div>

        <div class="menu">
            <tiles:insert page="/template/menu.jsp"/>
        </div>

        <div class="pagina">

            <html:form method="post"
            action="ControleFuncionario.do?cmd=cadastrar">

                <p>
                    Para cadastrar um novo funcionário,
                    preencha os campos abaixo:
                </p>

```



# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
25

```
Nome do Funcionário: <br/>
<html:text property="funcionario.nome"/>
<span class="erro"> <html:errors
property="nomeinvalido"/> </span>
<br/><br/>

Salário: <br/>
<html:text property="funcionario.salario"/>
<span class="erro"> <html:errors
property="salarioinvalido"/> </span>
<br/><br/>

<html:submit value="Cadastrar Funcionário"/>

<p>
    <bean:write name="msg" ignore="true"/>
</p>

</html:form>
</div>
</body>
</html>
```

Projeto Struts - Controle de Funcionários

Tecnologias: Struts Tiles, Validação e Sessão

- [Cadastrar novo Funcionário](#)
- [Consultar Funcionários](#)

Para cadastrar um novo funcionário, preencha os campos abaixo:

Nome do Funcionário:  
123123 Por favor, informe o nome do funcionário corretamente.

Salário:  
0.0

Cadastrar Funcionário



# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
25

The screenshot shows the Eclipse IDE interface with the Java EE perspective selected. The central editor window displays the code for FuncionarioBean.java. The code implements validation logic for a Funcionario object using regular expressions and ActionErrors. It checks the name and salary fields against specific criteria. The code is annotated with comments explaining the validation logic. The status bar at the bottom right shows the date and time: 17/06/2014, 10:10.

```
Java EE - javaWebAula15a/src/beans/FuncionarioBean.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
estilo.css topo.jsp menu.jsp index.jsp cadastro.jsp FuncionarioB... struts-confi... MessageResou...
26  @Override
27  public ActionErrors validate(ActionMapping mapping,
28      HttpServletRequest request) {
29
30      //Classe para geração de erros no struts
31      ActionErrors erros = new ActionErrors(); //não tem erros
32
33      //Validação do nome do funcionário
34      if(funcionario.getNome() != null){
35
36          Pattern regex = Pattern.compile("^[A-Za-zÀ-Ùà-ù\\s]{3,30}"); //expressão de validação
37          Matcher m = regex.matcher(funcionario.getNome()); //aplicando a expressão ao nome
38
39          if( ! m.matches() ){ //se o nome do funcionário NÃO passou no regex
40              erros.add("nomeinvalido", new ActionMessage("erro.nome"));
41          }
42      }
43      //Validação do salário do funcionário
44      if(funcionario.getSalario() != null){
45          //Verificar se o salário é maior ou igual ao salário mínimo
46          if(funcionario.getSalario() < 720.00){
47              erros.add("salarioinvalido", new ActionMessage("erro.salario"));
48          }
49      }
50
51      return erros;
52  }
53
54  public Funcionario getFuncionario() {
55      return funcionario;
56  }
}
Writable Smart Insert 47 : 77
10:10 17/06/2014
```

```
package beans;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import entity.Funcionario;
public class FuncionarioBean extends ActionForm {
    // Atributo para armazenar os dados do funcionário
    private Funcionario funcionario;
    public FuncionarioBean() {
        //instancia (inicializa)
        funcionario = new Funcionario();
    }
    //Método sobreescrito da Classe ActionForm utilizado para
    //validação dos dados
```



```
@Override
public ActionErrors validate(ActionMapping mapping,
    HttpServletRequest request) {

    //Classe para geração de erros no struts
    ActionErrors erros = new ActionErrors();
    //vazia (nenhum erro!)

    //Validação do nome do funcionário
    if(funcionario.getNome() != null){

        Pattern regex = Pattern.compile
            ("^[A-Za-zA-Üä-ü\s]{3,30}");
        //expressão de validação
        Matcher m = regex.matcher(funcionario.getNome());
        //aplicando a expressão ao nome

        if( ! m.matches() ){
            //se o nome do funcionario NÃO passou no regex
            erros.add("nomeinvalido",
                new ActionMessage("erro.nome"));
        }
    }

    //Validação do salário do funcionário

    if(funcionario.getSalario() != null){

        //Verificar se o salário é maior ou igual
        //ao salário mínimo

        if(funcionario.getSalario() < 720.00){

            erros.add("salarioinvalido",
                new ActionMessage("erro.salario"));
        }
    }

    return erros;
}

public Funcionario getFuncionario() {
    return funcionario;
}

public void setFuncionario(Funcionario funcionario) {
    this.funcionario = funcionario;
}
```



# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
25

## Exibindo a listagem de funcionários...

```
package beans;

import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;

import entity.Funcionario;

public class FuncionarioBean extends ActionForm {

    // Atributo para armazenar os dados do funcionario
    private Funcionario funcionario;

    //Atributo para gerar uma lista de funcionários
    private List<Funcionario> listagemFuncionarios;
```



```
public FuncionarioBean() {  
    //instancia (inicializa)  
    funcionario = new Funcionario();  
    listagemFuncionarios = new  
        ArrayList<Funcionario>();  
}  
  
//Método sobrescrito da Classe ActionForm utilizado para  
//validação dos dados  
@Override  
public ActionErrors validate(ActionMapping mapping,  
    HttpServletRequest request) {  
  
    //Classe para geração de erros no struts  
    ActionErrors erros = new ActionErrors();  
    //vazia (nenhum erro!)  
  
    //Validação do nome do funcionário  
    if(funcionario.getNome() != null){  
  
        Pattern regex = Pattern.compile  
            ("^[A-Za-zÀ-Üà-ü\\s]{3,30}");  
        //expressão de validação  
        Matcher m = regex.matcher(funcionario.getNome());  
        //aplicando a expressão ao nome  
  
        if( ! m.matches() ){  
            //se o nome do funcionario NÃO passou no regex  
            erros.add("nomeinvalido",  
                new ActionMessage("erro.nome"));  
        }  
    }  
    //Validação do salário do funcionário  
    if(funcionario.getSalario() != null){  
        //Verificar se o salário é maior ou igual  
        //ao salário mínimo  
        if(funcionario.getSalario() < 720.00){  
            erros.add("salarioinvalido",  
                new ActionMessage("erro.salario"));  
        }  
    }  
  
    return erros;  
}  
  
public Funcionario getFuncionario() {  
    return funcionario;  
}  
  
public void setFuncionario(Funcionario funcionario) {  
    this.funcionario = funcionario;  
}
```



```
public List<Funcionario> getListagemFuncionarios() {
    return listagemFuncionarios;
}

public void setListagemFuncionarios(List<Funcionario>
    listagemFuncionarios) {
    this.listagemFuncionarios = listagemFuncionarios;
}

}

-----
package control;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import beans.FuncionarioBean;

public class ControleFuncionario extends Action{

    @Override
    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
        throws Exception {

        String cmd = request.getParameter("cmd"); //resgatar o cmd

        if(cmd.equalsIgnoreCase("cadastrar")){
            //verifico o valor do cmd

            return cadastrar(mapping, form, request, response);
            //chamada ao método 'cadastrar'
        }

        return super.execute(mapping, form, request, response);
    }

    public ActionForward cadastrar(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response)
```



```
throws Exception {  
  
    //Resgatar o Bean (camada de modelo de dados)  
    FuncionarioBean fb = (FuncionarioBean) form;  
  
    //adicionar na lista o funcionario preenchido  
    //no formulário  
  
    fb.getListagemFuncionarios().add( fb.getFuncionario() );  
  
    fb.setFuncionario(new Funcionario());  
    //novo espaço de memória  
  
    //mensagem  
    request.setAttribute("msg",  
        "Funcionario cadastrado com sucesso.");  
  
    return mapping.findForward("sucesso");  
}  
}
```

### Cadastrando...

The screenshot shows a web application running on a local server at port 8084. The title of the browser window is 'localhost:8084/javaWebAula15a'. The page itself is titled 'Projeto Struts - Controle de Funcionários' and mentions 'Tecnologias: Struts Tiles, Validação e Sessão'. It has a navigation menu with links to 'Cadastrar novo Funcionário' and 'Consultar Funcionários'. Below the menu, there's a form for entering employee information. The 'Nome do Funcionário:' field contains 'Sergio Mendes', and the 'Salário:' field contains '1000.0'. A button labeled 'Cadastrar Funcionário' is present. A success message 'Funcionario cadastrado com sucesso.' is displayed below the form. The browser's taskbar at the bottom shows various open tabs and icons.



### consulta.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="logic" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-tiles"
    prefix="tiles" %>

<html>

    <head>
        <link rel="stylesheet" href="styles/estilo.css"
            type="text/css"/>
    </head>

    <body>

        <div class="topo">
            <tiles:insert page="/template/topo.jsp"/>
        </div>

        <div class="menu">
            <tiles:insert page="/template/menu.jsp"/>
        </div>

        <div class="pagina">

            <html:form method="post"
                action="ControleFuncionario.do">

                <p>
                    Relação de Funcionários cadastrados
                </p>

                <logic:iterate name="FuncionarioBean"
                    property="listagemFuncionarios" id="f">

                    <strong>Nome do Funcionário: </strong>
                    <bean:write name="f" property="nome"/>
                    <br/>

                    <strong>Salário: </strong>
                    <bean:write name="f" property="salario"/>
                    <br/>

                <hr/>
            </html:form>
        </div>
    </body>
</html>
```



```
</logic:iterate>

</html:form>

</div>

</body>

</html>
```

Executando...

The screenshot shows a web browser window with the URL `localhost:8084/javaWebAula15a/consulta.jsp`. The page title is **Projeto Struts - Controle de Funcionários**. Below the title, it says **Tecnologias: Struts Tiles, Validação e Sessão**. There is a list of links:

- [Cadastrar novo Funcionário](#)
- [Consultar Funcionários](#)

Below the links, there is a section titled "Relação de Funcionários cadastrados". It lists three entries:

- Nome do Funcionário:** Sergio Mendes  
**Salário:** 1000.0
- Nome do Funcionário:** Marcone  
**Salário:** 2000.0
- Nome do Funcionário:** Deborah  
**Salário:** 3000.0

The browser's taskbar at the bottom shows various icons, and the status bar indicates the time is 10:44 and the date is 17/06/2014.



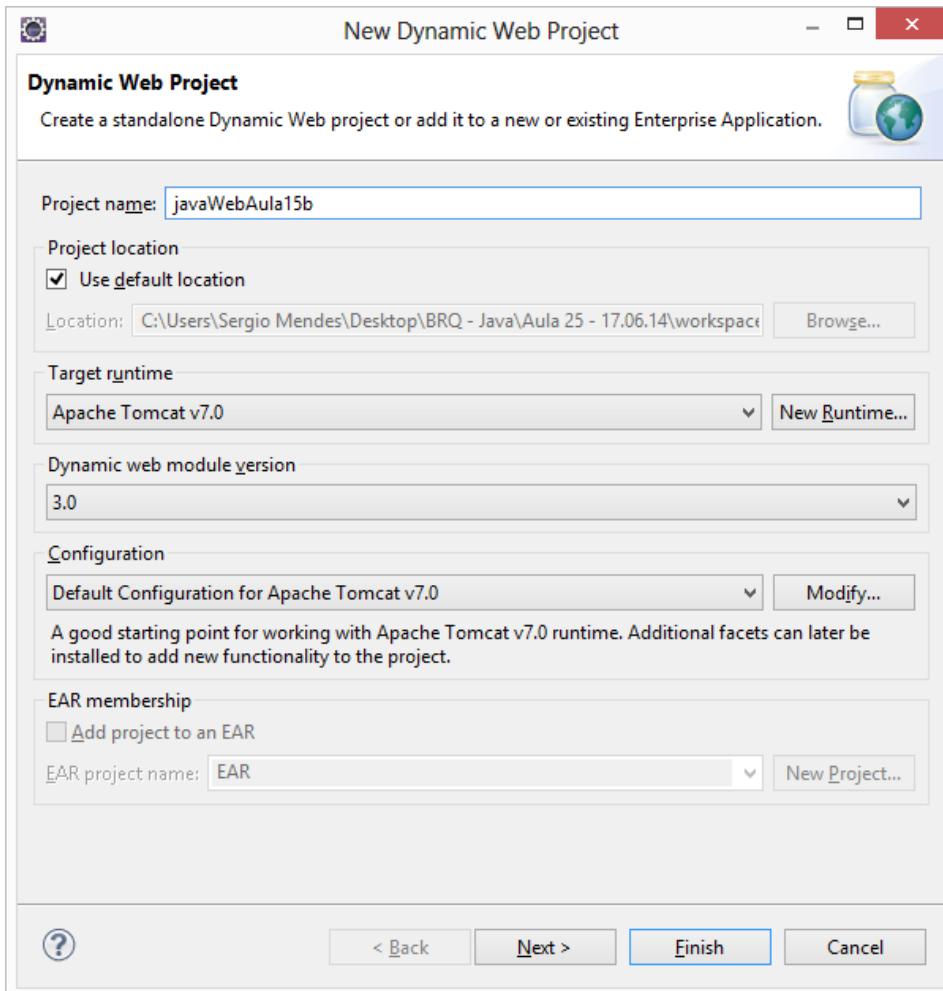
# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

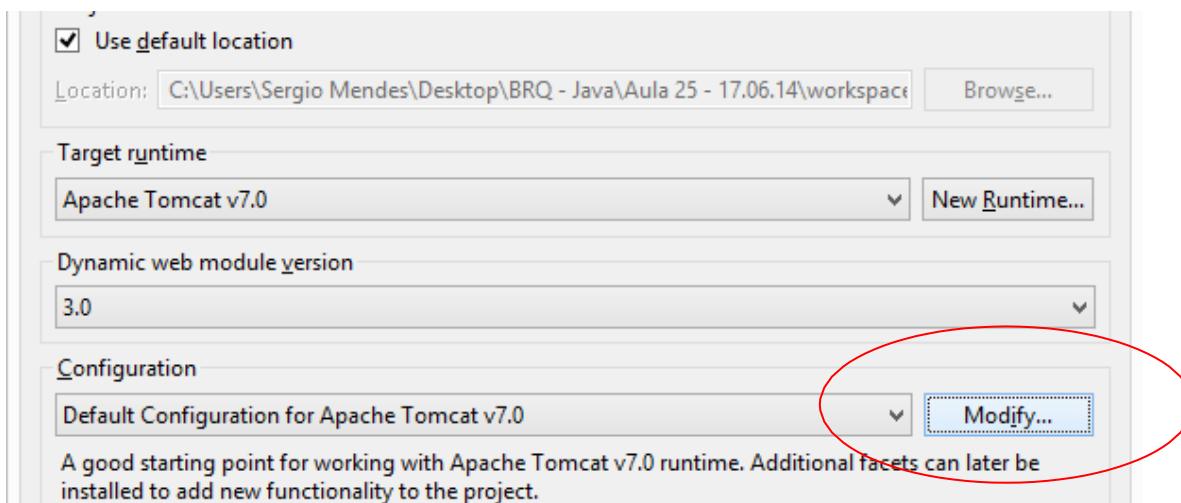
Aula  
**25**

## **Novo projeto...**



## **Java Server Faces**

Framework Java Web para criação de aplicações baseadas em camadas e com suporte a desenvolvimento de interfaces ricas.



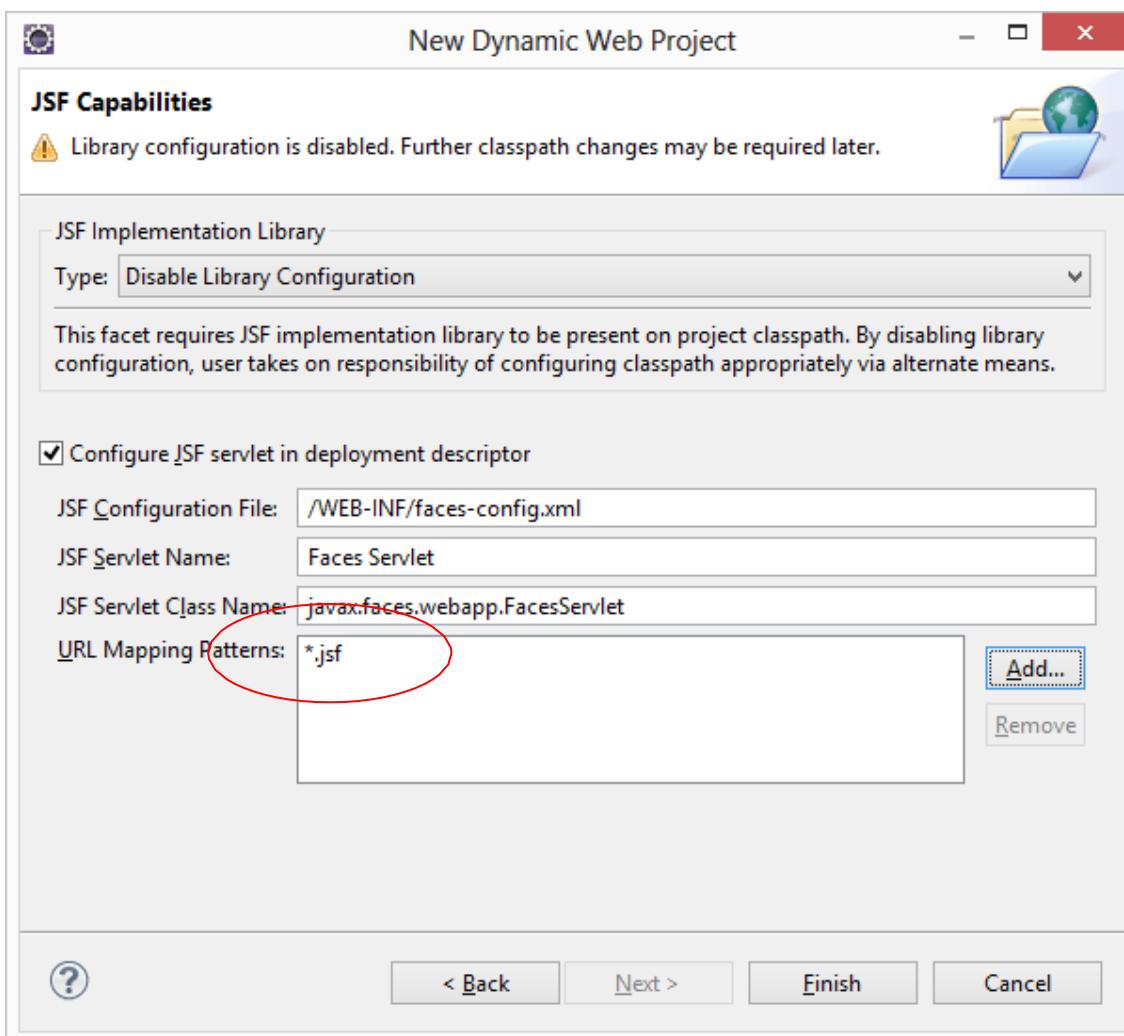
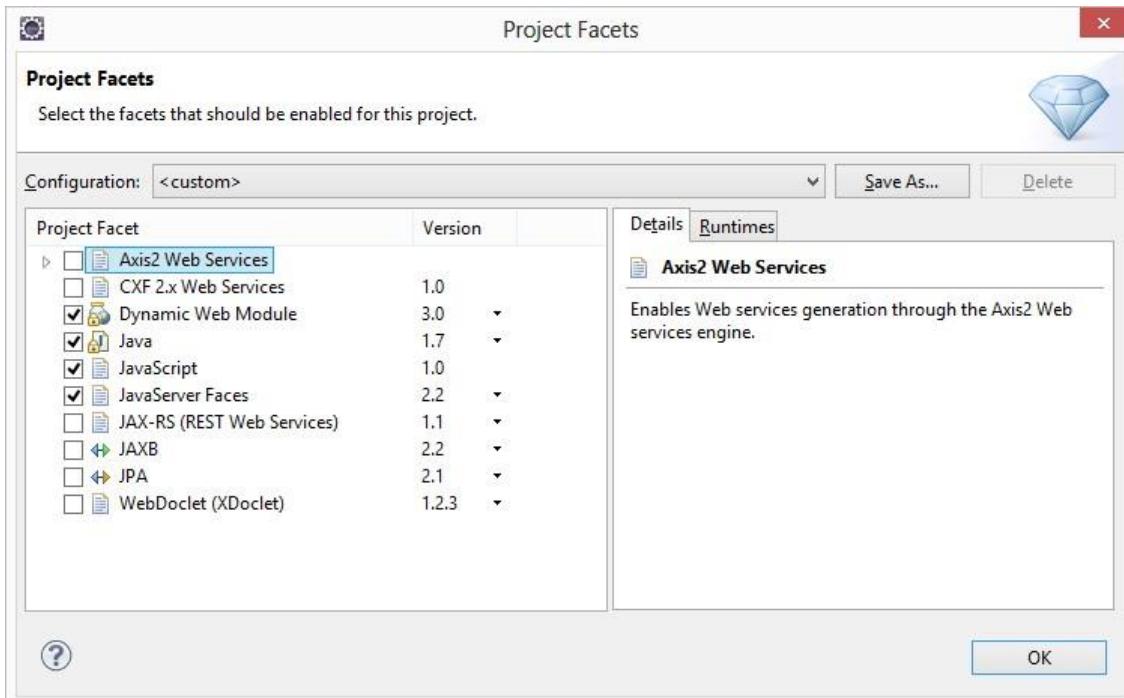


# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
25





## WEB-INF/web.xml

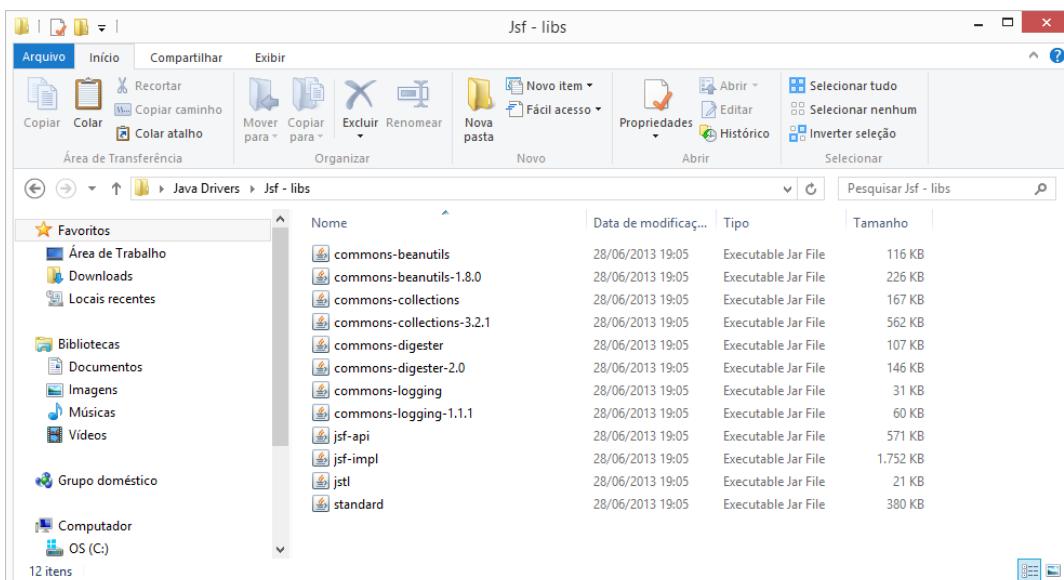
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>javaWebAula15b</display-name>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
</web-app>
```

## WEB-INF/faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
  http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
  version="2.2">

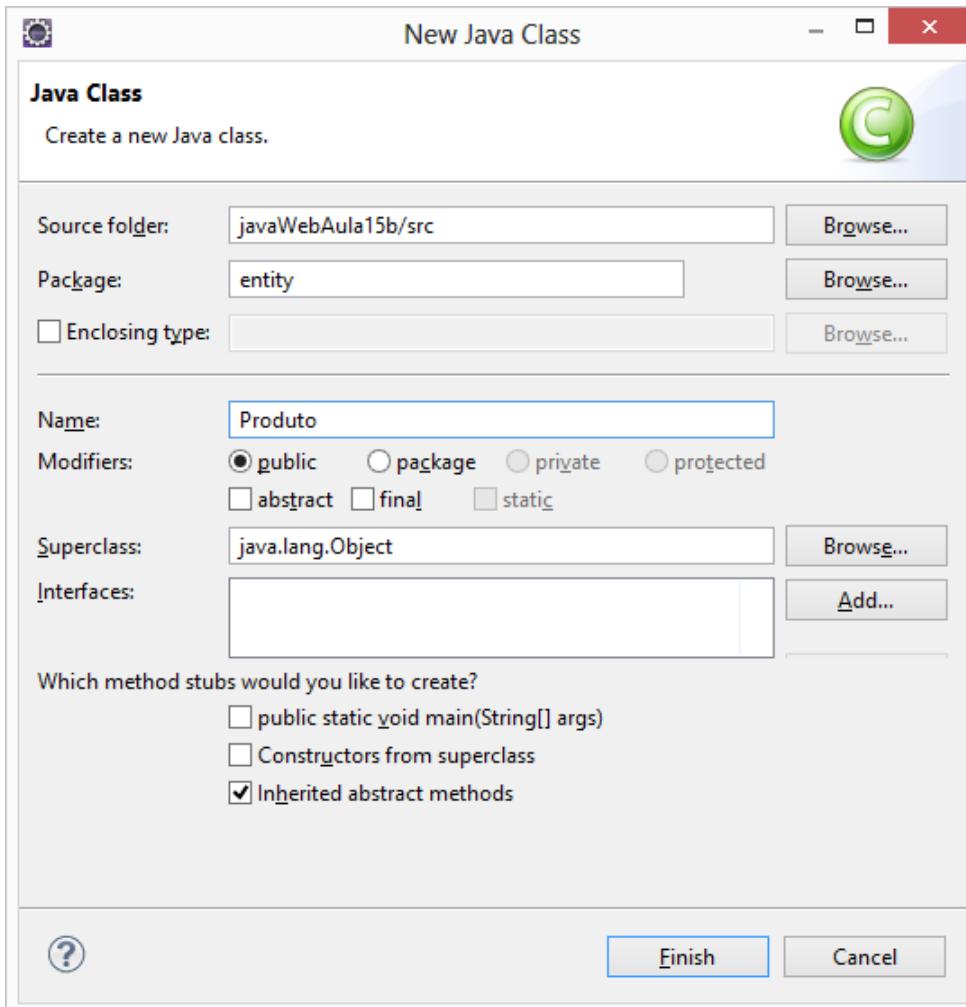
</faces-config>
```

## Incluindo os .JARs do JSF





### Criando o JavaBean



```
package entity;

import java.util.Date;

public class Produto {

    private String nome;
    private Date dataCompra;
    private Double preco;

    public Produto() {
        // Construtor default
    }

    public Produto(String nome, Date dataCompra, Double preco) {
        super();
        this.nome = nome;
        this.dataCompra = dataCompra;
        this.preco = preco;
    }
}
```



```
@Override
public String toString() {
    return "Produto [nome=" + nome + ", dataCompra="
           + dataCompra + ", preco=" + preco + "]";
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Date getDataCompra() {
    return dataCompra;
}

public void setDataCompra(Date dataCompra) {
    this.dataCompra = dataCompra;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}
}
```

---

## Página inicial /index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>

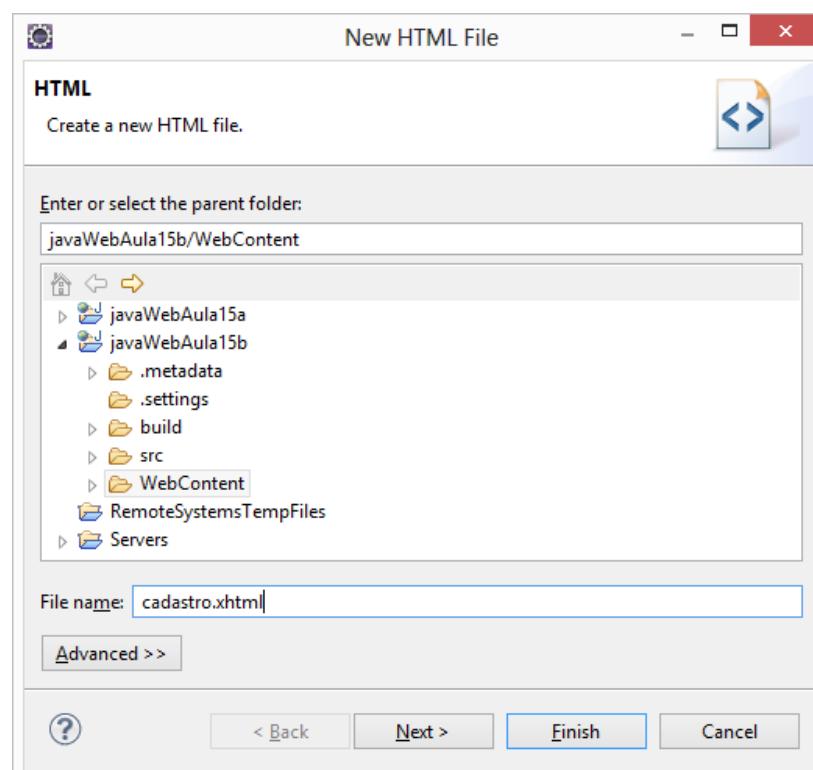
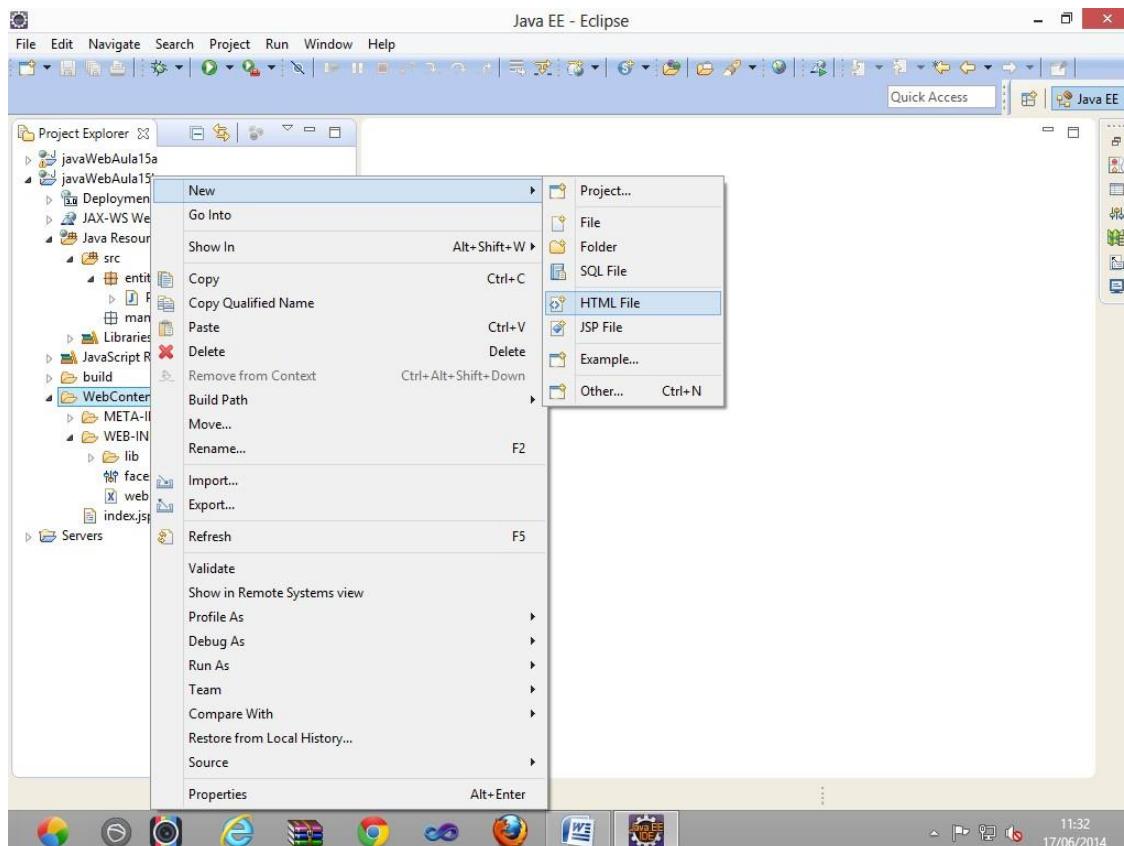
    <head>
    </head>

    <body>
        <a href="cadastro.jsf">Controle de Produtos</a>
    </body>

</html>
```



Toda página chamada com extensão .jsf deverá ser criada como .xhtml



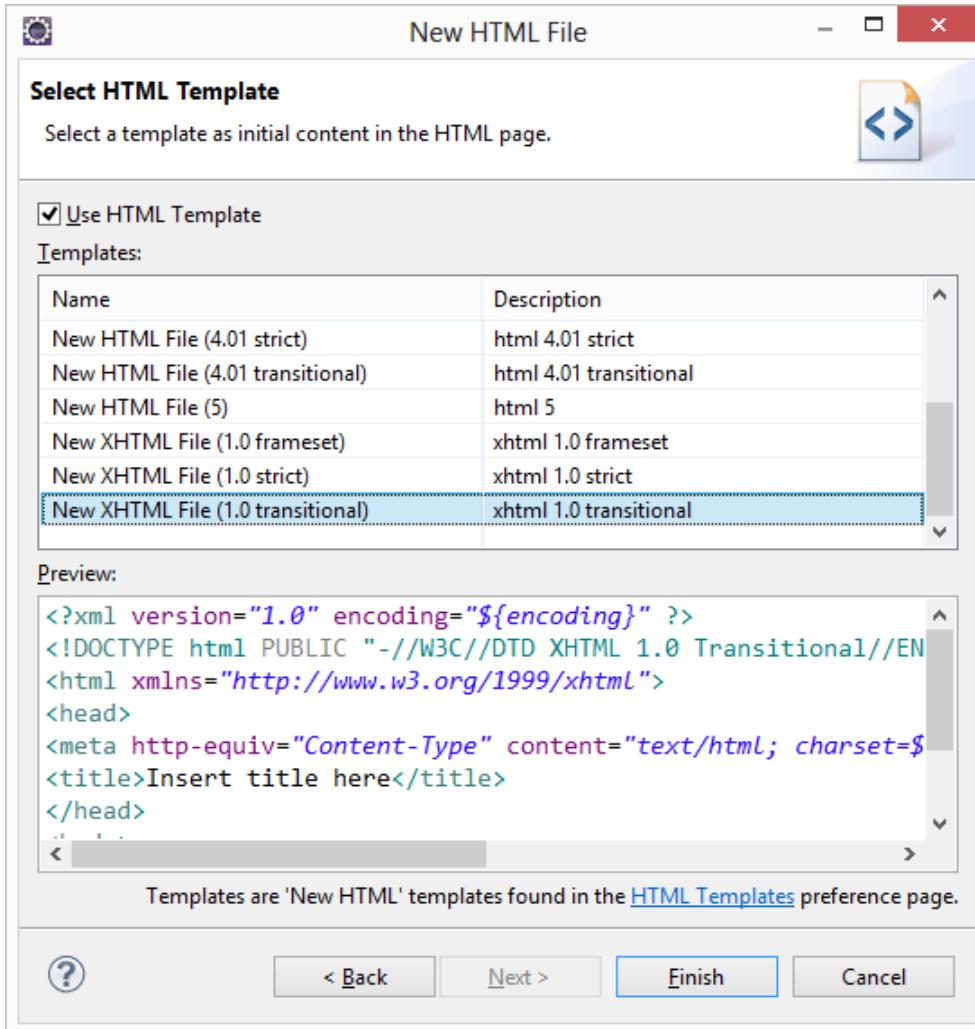


# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
25



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1" />

<title>Insert title here</title>
</head>

<body>
</body>
</html>
```



# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
**25**

<http://www.primefaces.org/>

Galeria de componentes visuais para o JSF

The screenshot shows the PrimeFaces homepage. At the top, there's a navigation bar with links for Overview, Demos, Development, Support, and Social. Below the navigation is the PrimeFaces logo and the text "Ultimate JSF Framework". There are three main buttons: DEMO, DOWNLOAD, and LEARN. A video player is embedded on the page, showing a video titled "PF5" with a duration of 01:26. Below the video is a screenshot of a Java IDE (NetBeans) showing a project structure. The system tray at the bottom right indicates the date as 17/06/2014 and the time as 11:39.

<http://www.primefaces.org/downloads>

The screenshot shows the "Community Downloads" section of the PrimeFaces website. It features a table with columns for Version, Binary, Source, and Bundle. The "Binary" and "Source" columns are circled in red. The table lists versions from 2.2.1 up to 5.0, along with their corresponding source code and bundle files. The "Binary" column contains links to download jars, and the "Source" column contains links to download zip files.

Version	Binary	Source	Bundle
5.0	<a href="#">primefaces-5.0.jar</a>	<a href="#">primefaces-5.0-sources.jar</a>	-
4.0	<a href="#">primefaces-4.0.jar</a>	<a href="#">primefaces-4.0-sources.jar</a>	<a href="#">primefaces-4.0.zip</a>
3.5	<a href="#">primefaces-3.5.jar</a>	<a href="#">primefaces-3.5-sources.jar</a>	<a href="#">primefaces-3.5.zip</a>
3.4.2	<a href="#">primefaces-3.4.2.jar</a>	<a href="#">primefaces-3.4.2-sources.jar</a>	<a href="#">primefaces-3.4.2.zip</a>
3.4.1	<a href="#">primefaces-3.4.1.jar</a>	<a href="#">primefaces-3.4.1-sources.jar</a>	<a href="#">primefaces-3.4.1.zip</a>
3.4	<a href="#">primefaces-3.4.jar</a>	<a href="#">primefaces-3.4-sources.jar</a>	<a href="#">primefaces-3.4.zip</a>
3.3.1	<a href="#">primefaces-3.3.1.jar</a>	<a href="#">primefaces-3.3.1-sources.jar</a>	<a href="#">primefaces-3.3.1.zip</a>
3.3	<a href="#">primefaces-3.3.jar</a>	<a href="#">primefaces-3.3-sources.jar</a>	<a href="#">primefaces-3.3.zip</a>
3.2	<a href="#">primefaces-3.2.jar</a>	<a href="#">primefaces-3.2-sources.jar</a>	<a href="#">primefaces-3.2.zip</a>
3.1.1	<a href="#">primefaces-3.1.1.jar</a>	<a href="#">primefaces-3.1.1-sources.jar</a>	<a href="#">primefaces-3.1.1.zip</a>
3.1	<a href="#">primefaces-3.1.jar</a>	<a href="#">primefaces-3.1-sources.jar</a>	<a href="#">primefaces-3.1.zip</a>
3.0.1	<a href="#">primefaces-3.0.1.jar</a>	<a href="#">primefaces-3.0.1-sources.jar</a>	<a href="#">primefaces-3.0.1.zip</a>
3.0	<a href="#">primefaces-3.0.jar</a>	<a href="#">primefaces-3.0-sources.jar</a>	<a href="#">primefaces-3.0.zip</a>
2.2.1	<a href="#">primefaces-2.2.1.jar</a>	<a href="#">primefaces-2.2.1-sources.jar</a>	<a href="#">primefaces-2.2.1.zip</a>

Showcase

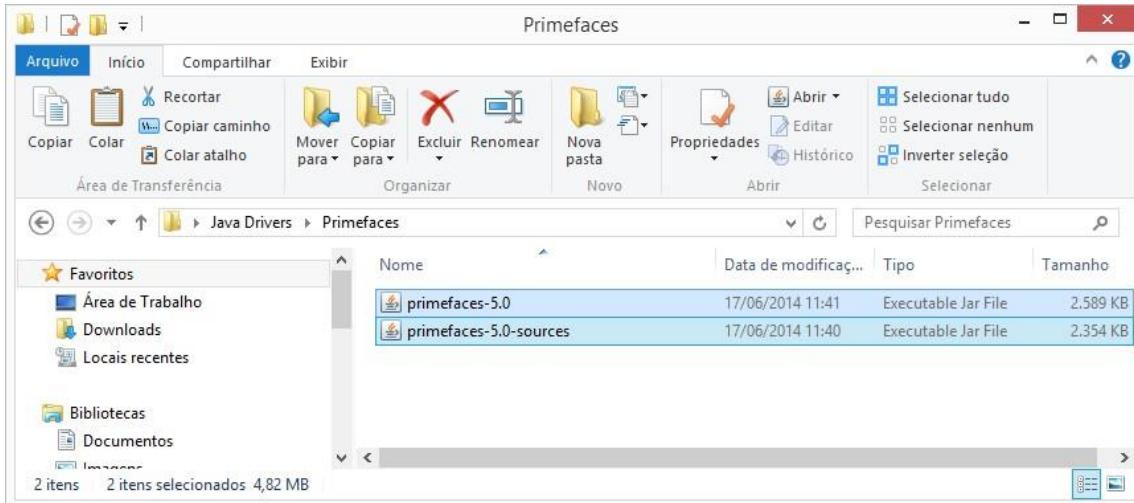


# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
25



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
      xmlns:p="http://primefaces.org/ui">
<h:head>

</h:head>
<h:body>

    <p:panel header="Projeto Controle de Produtos">

        <p:commandButton value="Cadastrar Novo Produto"
                          onclick="PF('janela').show()"/>

    </p:panel>

    <p:dialog widgetVar="janela" header="Formulário de Cadastro"
              modal="true">

        <h:form>

            <h:panelGrid columns="2">

                <h:outputText value="Nome do Produto:"/>
                <p:inputText/>

                <h:outputText value="Preço:"/>
                <p:inputText/>

                <h:outputText value="Data de Compra:"/>
                <p:calendar/>

            </h:panelGrid>
        
```



```
<p:commandButton value="Realizar Cadastro"/>

<p:messages/>

</h:form>

</p:dialog>

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

## Aplicando validação nos campos...

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
      xmlns:p="http://primefaces.org/ui">
<h:head>

</h:head>
<h:body>

    <p:panel header="Projeto Controle de Produtos">

        <p:commandButton value="Cadastrar Novo Produto"
                          onclick="PF('janela').show()"/>

    </p:panel>

    <p:dialog widgetVar="janela" header="Formulário de Cadastro"
               modal="true">

        <h:form id="formcadastro">

            <h:panelGrid columns="2">

                <h:outputText value="Nome do Produto:"/>
                <p:inputText required="true"
                           requiredMessage="Por favor, informe o nome
                           do produto"/>

                <h:outputText value="Preço:"/>
                <p:inputText required="true"
                           requiredMessage="Por favor, informe o preço
                           do produto"/>

            </h:panelGrid>
        </h:form>
    </p:dialog>

```



```
<h:outputText value="Data de Compra:"/>
<p:calendar required="true"
requiredMessage="Por favor, informe a data
de compra"/>

</h:panelGrid>

<p:commandButton value="Realizar Cadastro"
ajax="true" update=":formcadastro"/>

<p:messages/>

<p:growl/>

</h:form>

</p:dialog>

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

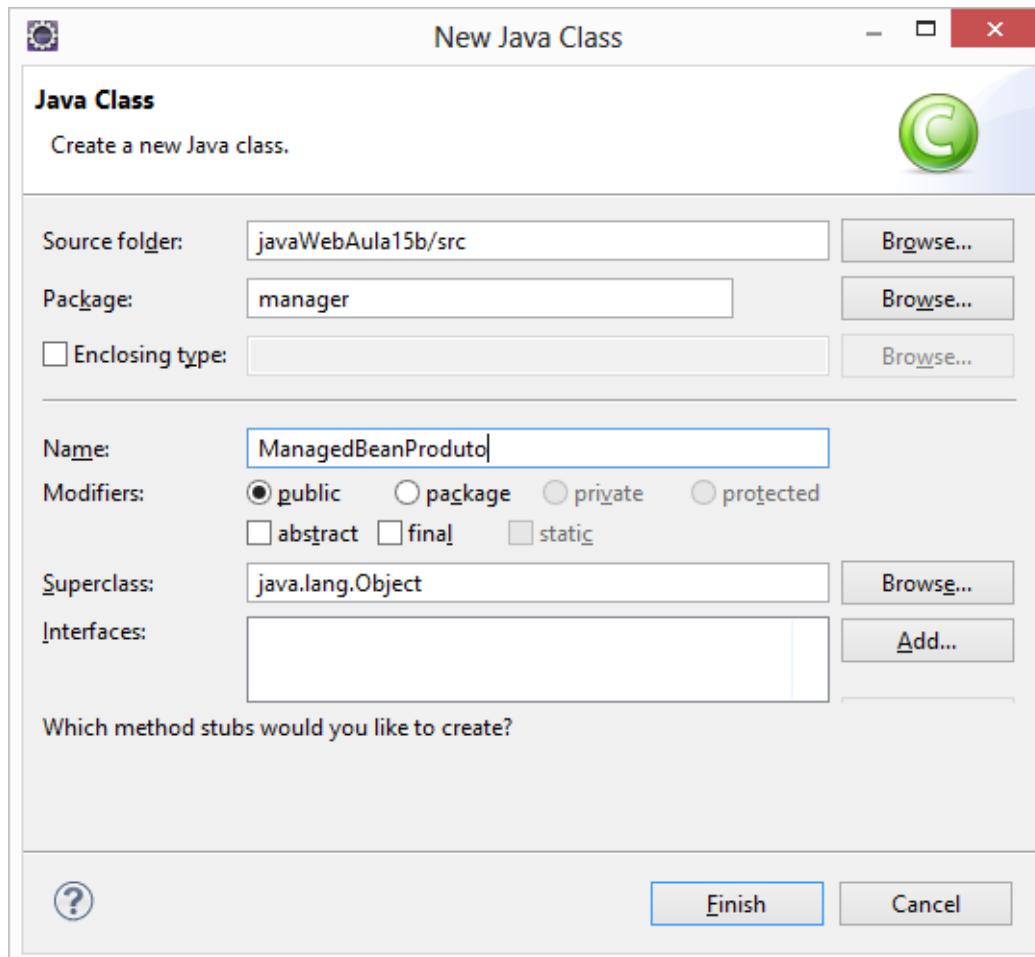
Executando...

The screenshot shows a web browser window with the URL [localhost:8084/javaWebAula15b/cadastro.jsf](http://localhost:8084/javaWebAula15b/cadastro.jsf). The page has a header 'Projeto Controle de Produtos' and a button 'Cadastrar Novo Produto'. A modal dialog box titled 'Formulário de Cadastro' is open, containing three input fields: 'Nome do Produto', 'Preço', and 'Data de Compra', all of which have red borders around them. To the right of the dialog, three validation messages are shown in a tooltip: 'Por favor, informe o nome do produto', 'Por favor, informe o preço do produto', and 'Por favor, informe a data de compra'. The browser's taskbar at the bottom shows icons for various applications like Google Chrome, Internet Explorer, and Firefox.



## ManagedBean

São classes que fazem o papel de camada de controle quando camada de modelo de dados. Gerenciador.



```
package manager;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

import entity.Produto;

@ManagedBean(name="mbProduto")
@RequestScoped
public class ManagedBeanProduto {

    // Capturar os dados do formulário
    private Produto produto;

    public ManagedBeanProduto() {
        produto = new Produto(); // inicializando
    }
}
```



# Java WebDeveloper - BRQ

Terça-feira, 17 de Junho de 2014

Projeto Login de Usuários, Upload de imagens. Segurança e Validação de dados, persistência com Hibernate e JPA Introdução ao JSF.

Aula  
**25**

```
//Método para cadastrar Produto
public void cadastrar(){

}

public Produto getProduto() {
    return produto;
}

public void setProduto(Produto produto) {
    this.produto = produto;
}

-----  

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
      xmlns:p="http://primefaces.org/ui">
<h:head>

</h:head>
<h:body>

    <p:panel header="Projeto Controle de Produtos">
        <p:commandButton value="Cadastrar Novo Produto"
                          onclick="PF('janela').show()"/>
    </p:panel>

    <p:dialog widgetVar="janela" header="Formulário de Cadastro"
               modal="true">

        <h:form id="formcadastro">
            <h:panelGrid columns="2">
                <h:outputText value="Nome do Produto:"/>
                <p:inputText value="#{mbProduto.producto.nome}"
                           required="true" requiredMessage="Por
                           favor, informe o nome do produto"/>
            </h:panelGrid>
        </h:form>
    </p:dialog>

```



```
<h:outputText value="Preço:"/>
<p:inputText
    value="#{mbProduto.producto.preco}"
    required="true" requiredMessage="Por
    favor, informe o preço do produto"/>

<h:outputText value="Data de Compra:"/>
<p:calendar
    value="#{mbProduto.producto.dataCompra}"
    required="true" requiredMessage="Por
    favor, informe a data de compra"/>

</h:panelGrid>

<p:commandButton value="Realizar Cadastro"
    ajax="true" update=":formcadastro"
    action="#{mbProduto.cadastrar}"/>

<p:messages/>

<p:growl/>

</h:form>

</p:dialog>

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

Implementando o método no ManagedBean

```
package manager;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

import entity.Produto;

@ManagedBean(name="mbProduto")
@RequestScoped
public class ManagedBeanProduto {

    // Capturar os dados do formulário
    private Produto produto;

    public ManagedBeanProduto() {
        produto = new Produto(); // inicializando
    }
}
```



```
//Método para cadastrar Produto
public void cadastrar(){

    FacesContext context = FacesContext.getCurrentInstance();
    FacesMessage message = new FacesMessage("Dados do
        Produto: " + produto); //toString

    //Exibir a mensagem dentro do formulario de cadastro
    context.addMessage("formcadastro", message);
}

public Produto getProduto() {
    return produto;
}

public void setProduto(Produto produto) {
    this.produto = produto;
}

}
```

### Executando...

The screenshot shows a browser window at `localhost:8084/javaWebAula15b/cadastro.jsf`. The page title is "Projeto Controle de Produtos". A button labeled "Cadastrar Novo Produto" is visible. A modal dialog titled "Formulário de Cadastro" contains fields for "Nome do Produto" (Mouse), "Preço" (30.0), and "Data de Compra" (17/06/14). A blue "Realizar Cadastro" button is at the bottom. A success message in the dialog says: "Dados do Produto: Produto [nome=Mouse, dataCompra=Tue Jun 17 00:00:00 BRT 2014, preço=30.0]". Above the dialog, a larger message box displays the same information: "Dados do Produto: Produto [nome=Mouse, dataCompra=Tue Jun 17 00:00:00 BRT 2014, preço=30.0]" with an info icon.



# Java WebDeveloper - BRQ

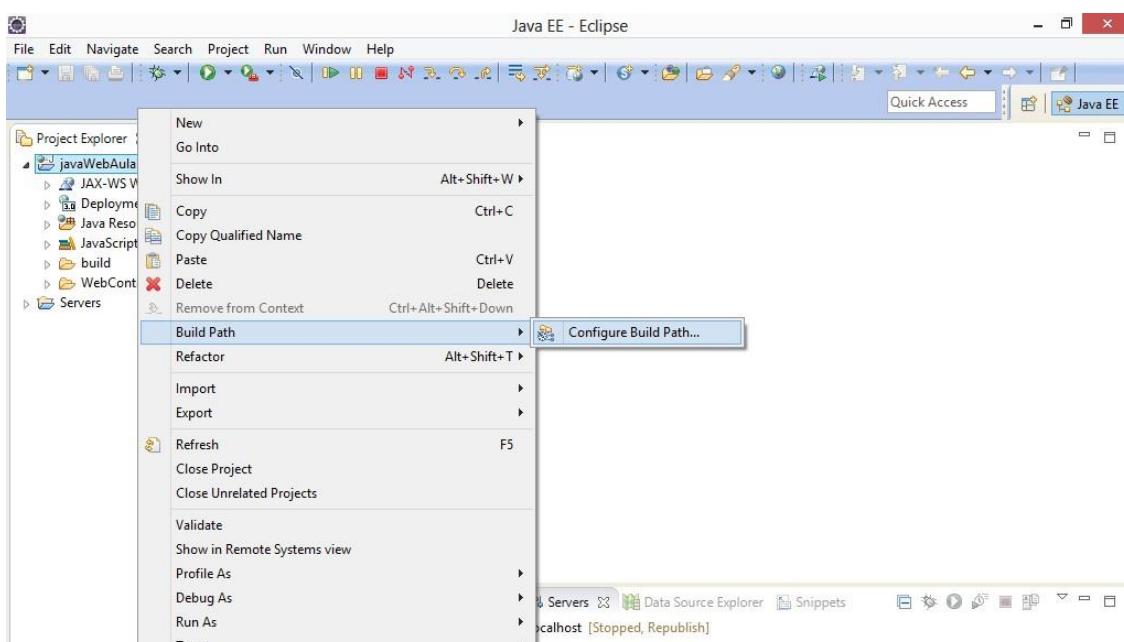
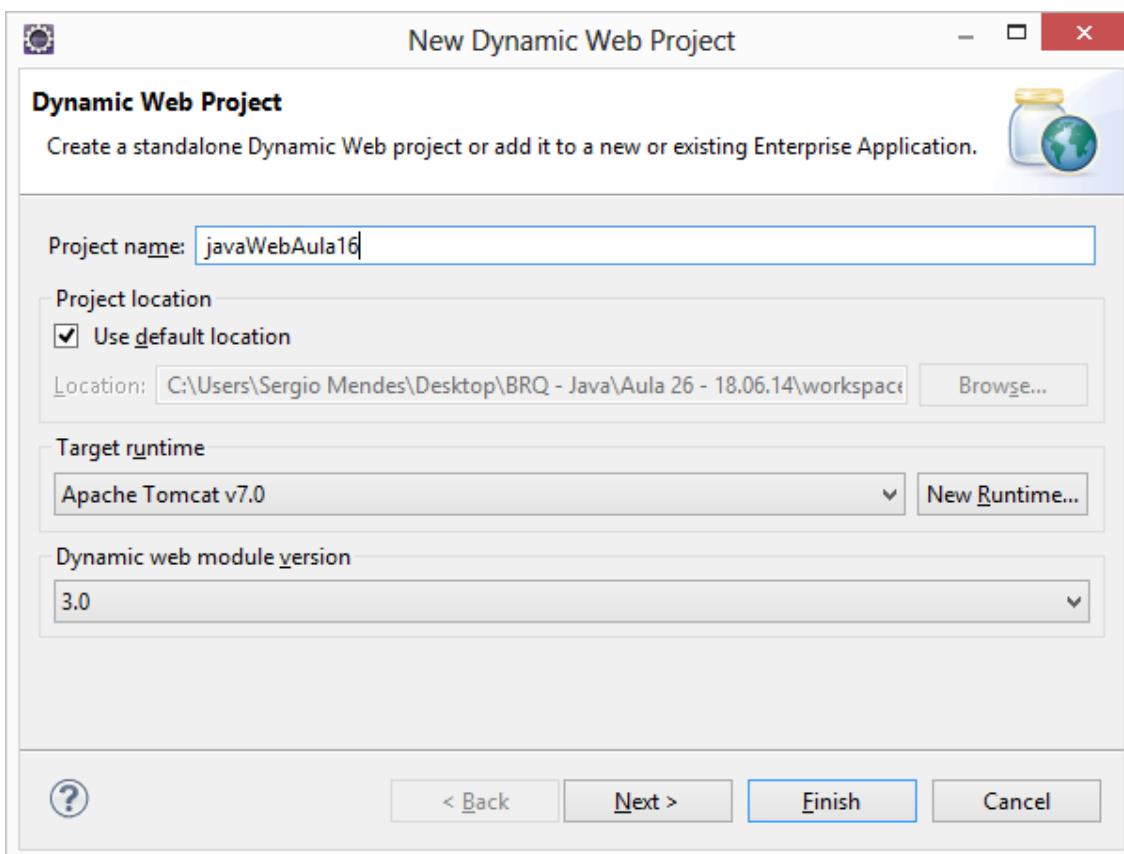
Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

## Java Server Faces

- JSF para desenvolvimento web
- Primefaces para interface de usuário
- Hibernate e JPA
- iReport



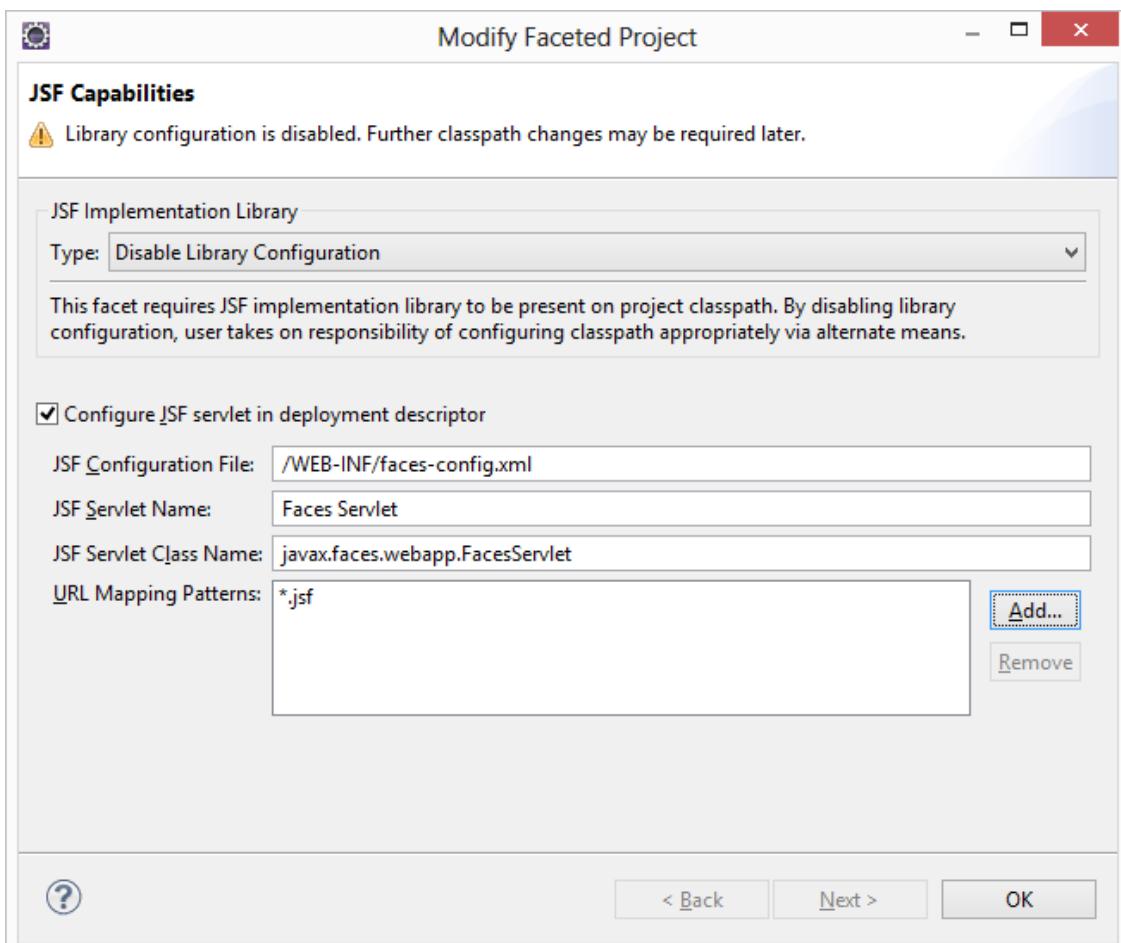
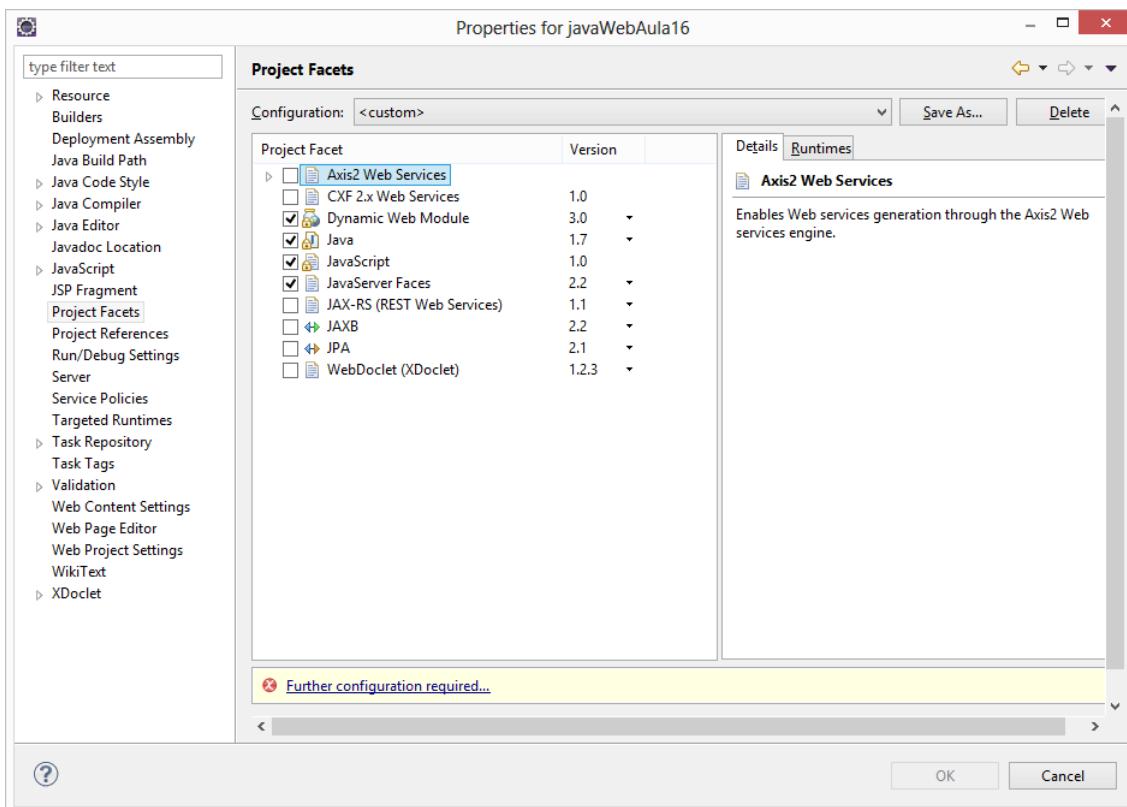


# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**





## /WEB-INF/web.xml

Arquivo de configuração padrão do projeto.

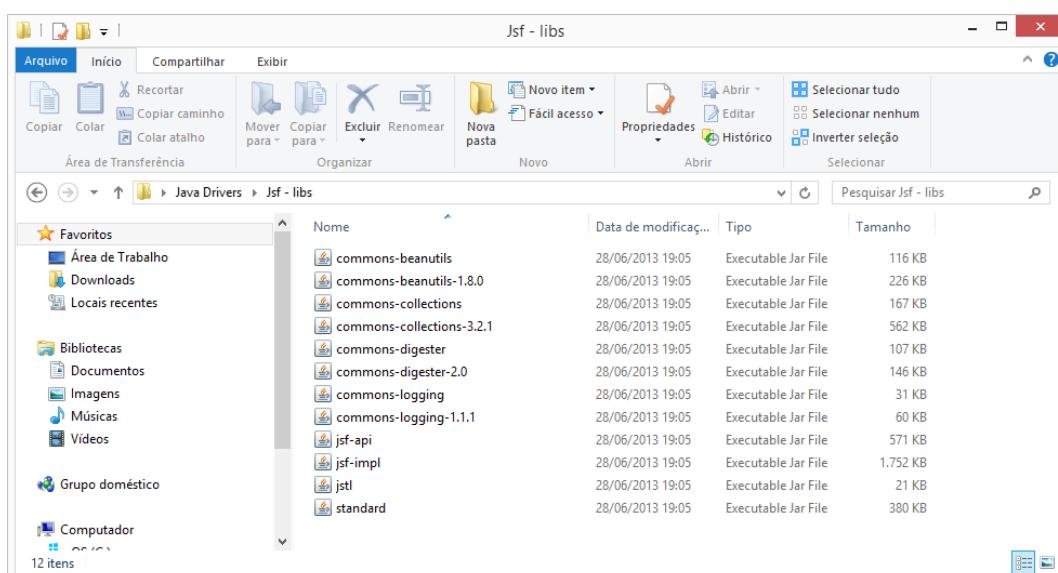
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <display-name>javaWebAula16</display-name>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
</web-app>
```

## /WEB-INF/faces-config.xml

Arquivo de configuração do JSF.

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
  http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
  version="2.2">

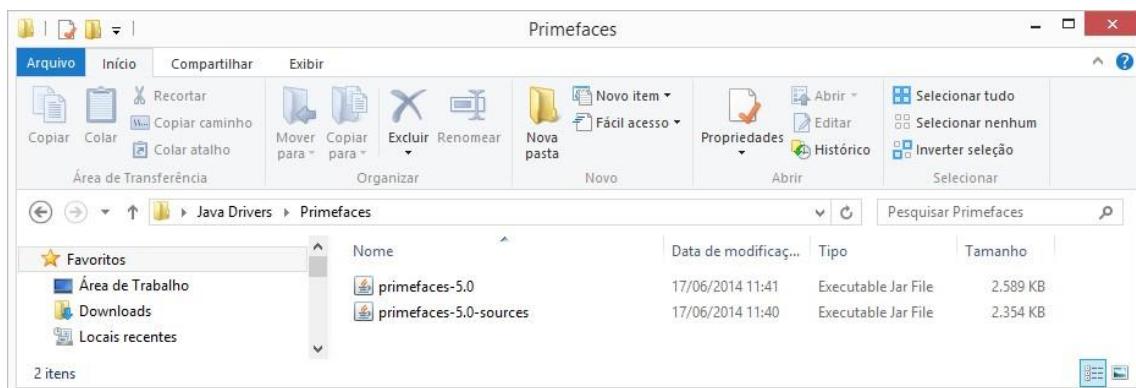
</faces-config>
```



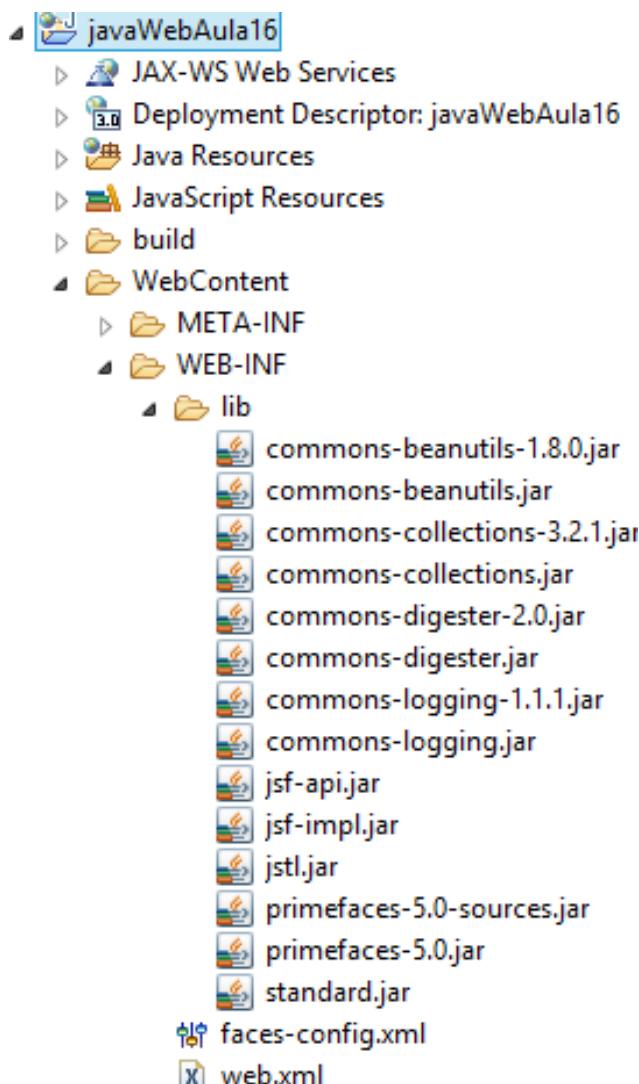


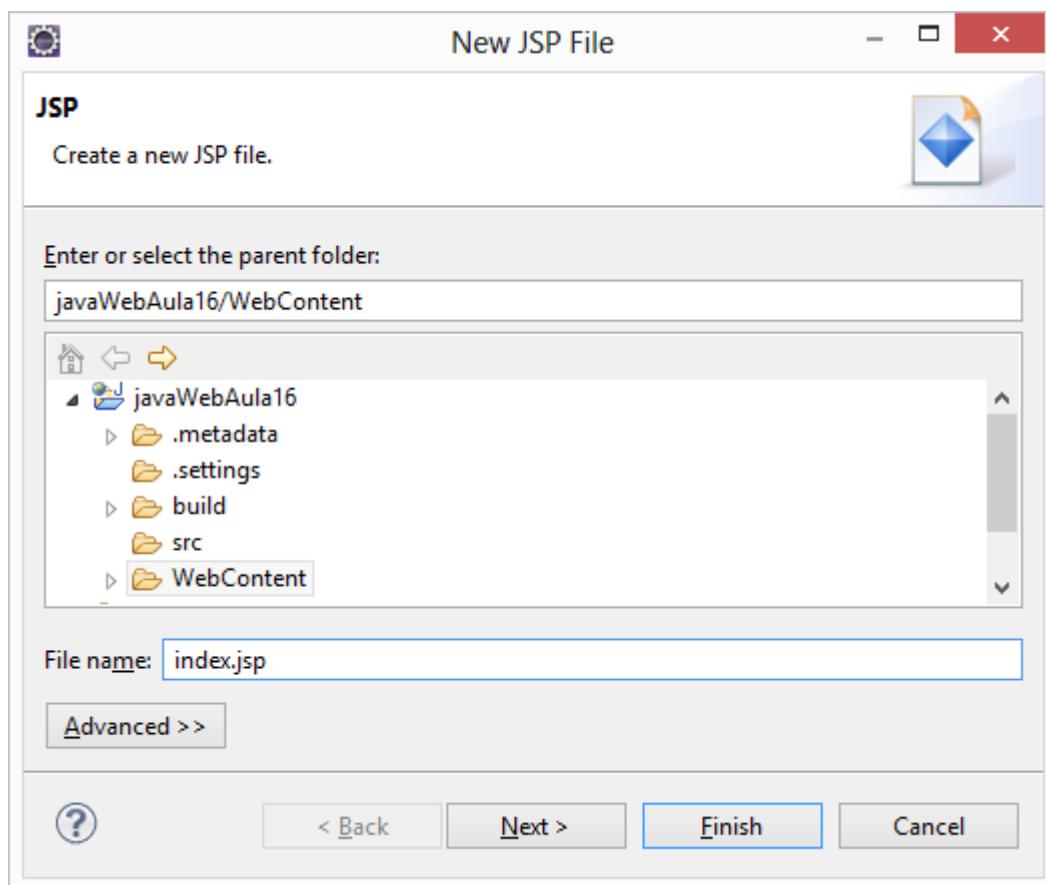
## Primefaces

Galeria de componentes visuais para o JSF.

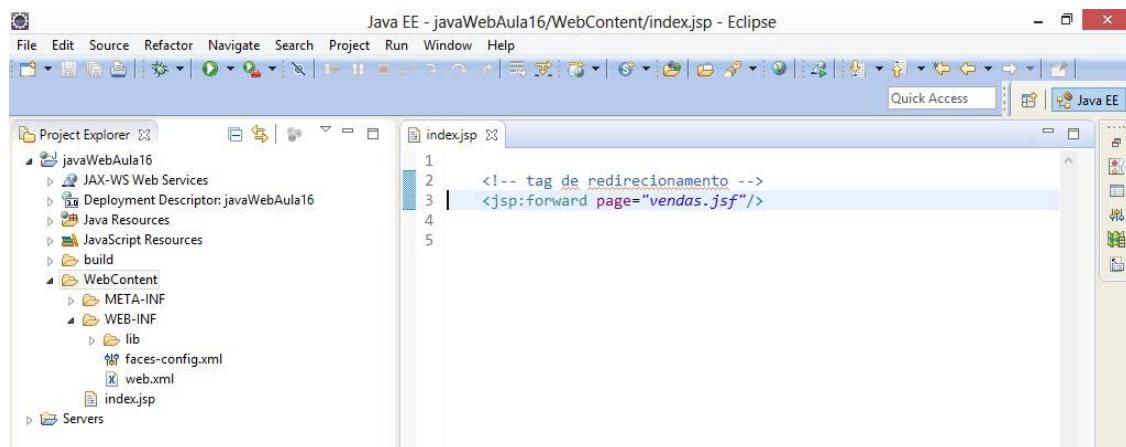


## Estrutura do projeto...





## Página inicial do projeto...



```
<!-- tag de redirecionamento -->
<jsp:forward page="vendas.jsf"/>
```

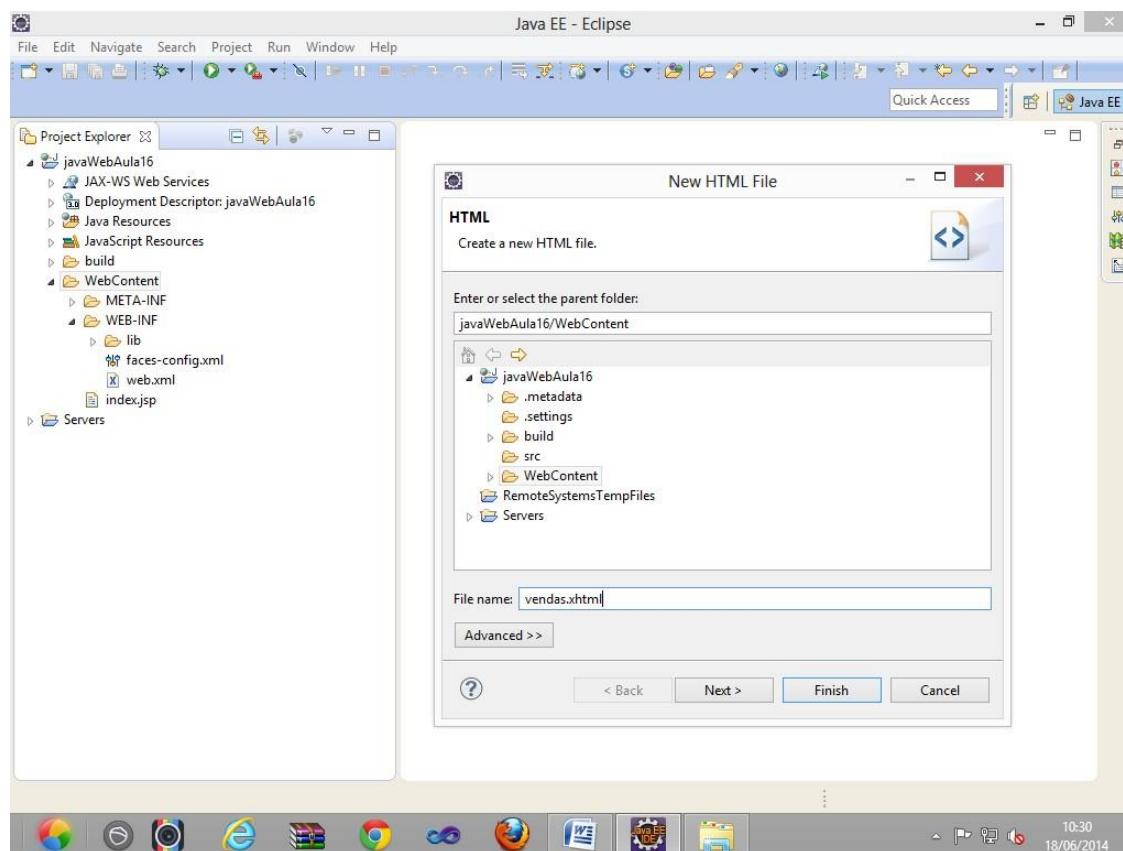
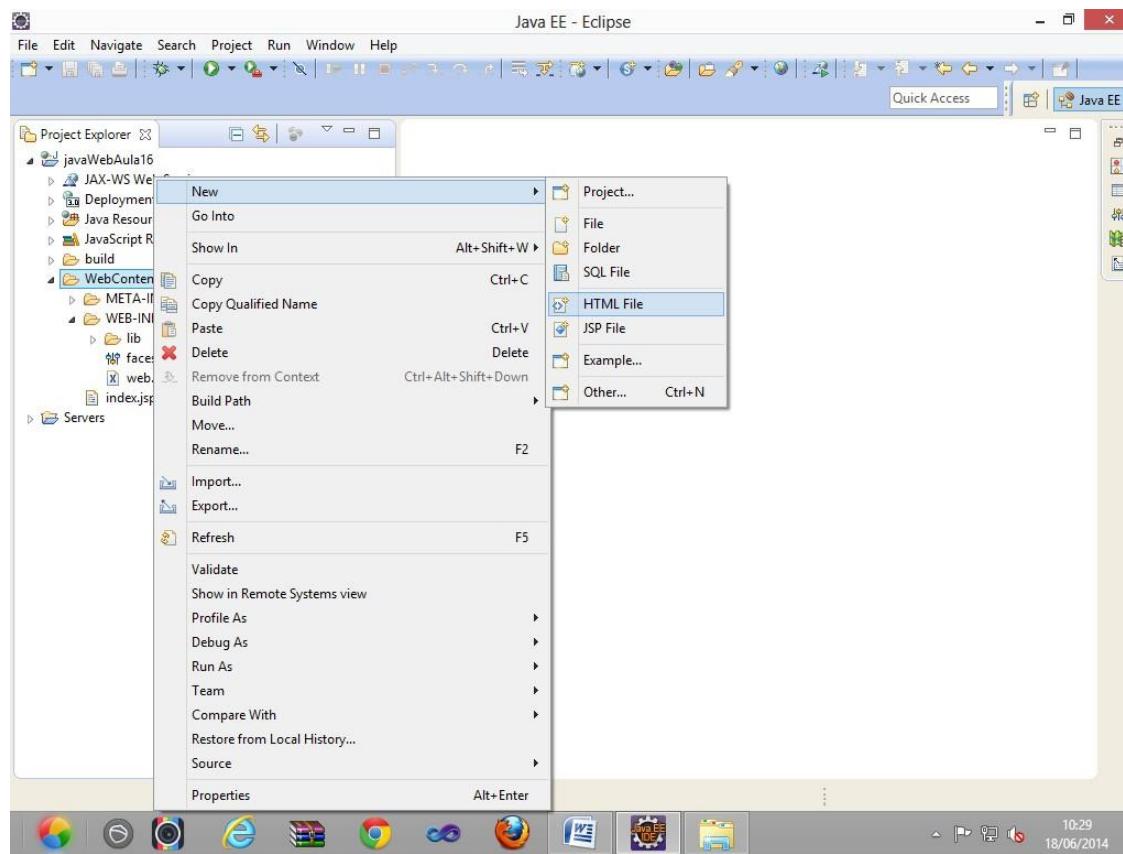


# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**



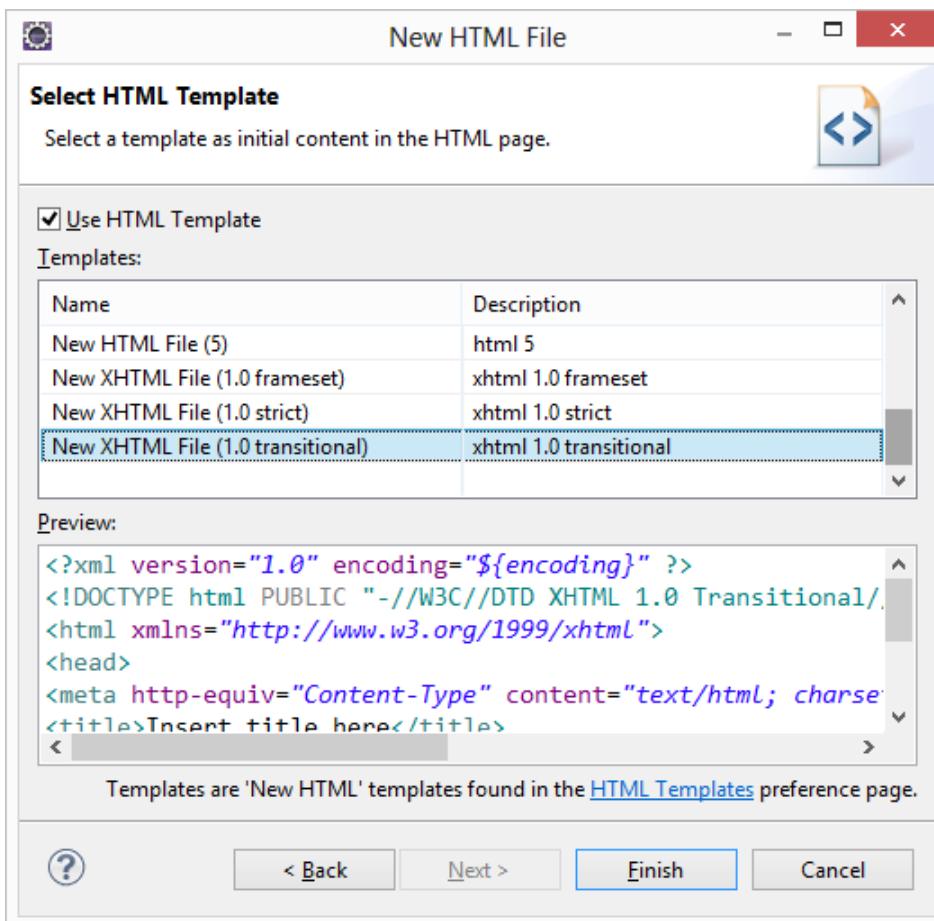


# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**



## Criando a página de vendas...

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
      xmlns:p="http://primefaces.org/ui">

<h:head>

</h:head>
<h:body>

    <p:layout fullPage="true">

        <p:layoutUnit position="north"
                      header="Aula de JSF e Primefaces">

            <h:outputText value="Projeto Controle de Vendas"
                          style="font-size: 20pt;"/>
        </p:layoutUnit>
    
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

```
<p:layoutUnit position="west" header="Menu de Opções"
    collapsible="true">

    <h:outputText value="Selecione a operação
        desejada:"/>
    <br/><br/>

    <h:form>
        <p:panelMenu>
            <p:submenu label="Manter Vendas">

                <p:menuitem value="Cadastrar Nova
                    Venda" onclick="
                        PF('janelacadastro').show()"/>

                <p:menuitem value="Gráfico de
                    Vendas" onclick="
                        PF('janelagrafico').show()"/>

            </p:submenu>
        </p:panelMenu>
    </h:form>

</p:layoutUnit>

<p:layoutUnit position="center">

    <h:form>

        <p:dataTable emptyMessage="Não há vendas
            cadastradas" paginator="true">
        </p:dataTable>

        <p:commandButton value="Exibir Relatório
            de Vendas"/>

    </h:form>

</p:layoutUnit>

</p:layout>

<!-- Janela para cadastro de vendas -->
<p:dialog widgetVar="janelacadastro"
    header="Cadastro de Vendas" modal="true">

    <h:form id="formcadastro">
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula

26

```
<h:panelGrid columns="2" style="padding: 20px;">

    <h:outputText value="Data da Venda:"/>
    <p:calendar/>

    <h:outputText value="Valor da Venda:"/>
    <p:inputText/>

    <h:outputText value="Selecione o Vendedor:"/>

    <p:selectOneMenu>

        <f:selectItem itemValue=""
                      itemLabel="- Escolha uma Opção -"
                      noSelectionOption="true"/>

        <f:selectItem itemValue="Patricia"
                      itemLabel="Patricia Massaut" />

        <f:selectItem itemValue="Genisson"
                      itemLabel="Genisson Silva" />

        <f:selectItem itemValue="Pedro"
                      itemLabel="Pedro Igor" />

        <f:selectItem itemValue="Rafael"
                      itemLabel="Rafael Ferreira" />

    </p:selectOneMenu>

</h:panelGrid>

<p:commandButton value="Cadastrar Venda"/>

</h:form>

</p:dialog>

<!-- Janela para gráfico de vendas -->
<p:dialog widgetVar="janelagrafico"
           header="Gráfico de Vendas" modal="true">

</p:dialog>

</h:body>

</html>
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

Aula de JSF e Primefaces

Projeto Controle de Vendas

Menu de Opções

Selecionar a operação desejada:

Manter Vendas

Cadastrar Nova Venda

Gráfico de Vendas

Não há vendas cadastradas

Exibir Relatório de Vendas

Aula de JSF e Primefaces

Projeto Controle de Vendas

Menu de Opções

Selecionar a operação desejada:

Manter Vendas

Cadastrar Nova Venda

Gráfico de Vendas

Não há vendas cadastradas

Exibir Relatório de Vendas

Cadastro de Vendas

Data da Venda:

Valor da Venda:

Selecionar o Vendedor:

Cadastrar Venda



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.

Persistência de dados com Hibernate e JPA.

Aula

26

## Selecionar um tema para o primefaces...

The screenshot shows a browser window displaying the PrimeFaces website at [www.primefaces.org](http://www.primefaces.org). The main navigation bar includes links for Overview, Demos, Development, Support, and Social. A dropdown menu under the Development link is open, showing options like Getting Started, Documentation, Downloads, Wiki, Theme Gallery, and Extensions. Below the navigation, there are three prominent buttons: DEMO, DOWNLOAD, and LEARN. A large video player is centered on the page, showing a thumbnail for 'PF5' and a video titled 'PRIMEFACES NEXTCEN'. Below the video player, a smaller window shows a preview of an 'nb80 overview video' on Vimeo. The overall layout is clean and modern, typical of a developer-focused framework documentation site.

<http://www.primefaces.org/themes>

The screenshot shows a browser window displaying the PrimeFaces theme gallery at [www.primefaces.org/themes](http://www.primefaces.org/themes). The page displays a grid of 24 different calendar skins, each showing a unique design. The skins are arranged in four rows of six. Each calendar is accompanied by a magnifying glass icon, likely for a larger view or details. The themes include: afterdark, afternoon, aristo (default), black-tie, blitz, bluesky (richfaces), casablanca (trinidad), cruise, Cupertino, dark-hive, delta, dot-luv, eggplant, excite-bike, flick, glass-x (richfaces), home, hot-sneaks, humanity, le-frog, midnight (wijmo), mint-choc, overcast, pepper-grinder, redmond, rocket (wijmo), sam, smoothness, and south-street. The browser toolbar at the bottom shows icons for various web browsers and tools. The status bar at the bottom right indicates the time as 11:39 and the date as 18/06/2014.



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula

26

<http://repository.primefaces.org/org/primefaces/themes/>

The screenshot shows a Windows desktop environment with a taskbar at the bottom containing icons for various applications like Internet Explorer, Google Chrome, and Mozilla Firefox. A web browser window is open, displaying the URL [repository.primefaces.org/org/primefaces/themes/](http://repository.primefaces.org/org/primefaces/themes/). The page title is "Index of /org/primefaces/themes". Below the title is a table with the following columns: Name, Last modified, Size, and Description. The table lists numerous theme directories, each with a small folder icon and a timestamp indicating when it was last modified. The themes listed include afterdark, afternoon, afterwork, all-themes, aristo, black-tie, blitz, bluesky, bootstrap, casablanca, czuza, cupertino, dark-hive, delta, dot-hiv, eggplant, excite-bike, flick, glass-x, and home. The timestamp for most entries is 21-Apr-2013 23:00, except for aristo which is 05-Sep-2011 15:12.

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>	-	-	
<a href="#">afterdark/</a>	21-Apr-2013 22:59	-	
<a href="#">afternoon/</a>	21-Apr-2013 22:59	-	
<a href="#">afterwork/</a>	21-Apr-2013 23:00	-	
<a href="#">all-themes/</a>	21-Apr-2013 23:18	-	
<a href="#">aristo/</a>	05-Sep-2011 15:12	-	
<a href="#">black-tie/</a>	21-Apr-2013 23:01	-	
<a href="#">blitzer/</a>	21-Apr-2013 23:01	-	
<a href="#">bluesky/</a>	21-Apr-2013 23:02	-	
<a href="#">bootstrap/</a>	21-Apr-2013 23:02	-	
<a href="#">casablanca/</a>	21-Apr-2013 23:03	-	
<a href="#">czuza/</a>	21-Apr-2013 23:03	-	
<a href="#">cupertino/</a>	21-Apr-2013 23:04	-	
<a href="#">dark-hive/</a>	21-Apr-2013 23:04	-	
<a href="#">delta/</a>	21-Apr-2013 23:05	-	
<a href="#">dot-hiv/</a>	21-Apr-2013 23:05	-	
<a href="#">eggplant/</a>	21-Apr-2013 23:06	-	
<a href="#">excite-bike/</a>	21-Apr-2013 23:06	-	
<a href="#">flick/</a>	21-Apr-2013 23:07	-	
<a href="#">glass-x/</a>	21-Apr-2013 23:07	-	
<a href="#">home/</a>	21-Apr-2013 23:08	-	

The screenshot shows a Windows desktop environment with a taskbar at the bottom containing icons for various applications like Internet Explorer, Google Chrome, and Mozilla Firefox. A web browser window is open, displaying the URL [repository.primefaces.org/org/primefaces/themes/bootstrap/](http://repository.primefaces.org/org/primefaces/themes/bootstrap/). The page title is "Index of /org/primefaces/themes/bootstrap". Below the title is a table with the following columns: Name, Last modified, Size, and Description. The table lists several files and directories related to the Bootstrap theme, including 1.0.8, 1.0.9, 1.0.10, maven-metadata.xml, maven-metadata.xml.md5, and maven-metadata.xml.sha1. The timestamp for 1.0.8 is 28-Aug-2012 22:39, while the other entries are from 21-Apr-2013 23:03. The sizes for the XML files are 374, 32, and 40 respectively.

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>	-	-	
<a href="#">1.0.8/</a>	28-Aug-2012 22:39	-	
<a href="#">1.0.9/</a>	17-Dec-2012 20:10	-	
<a href="#">1.0.10/</a>	21-Apr-2013 23:03	-	
<a href="#">maven-metadata.xml</a>	21-Apr-2013 23:03	374	
<a href="#">maven-metadata.xml.md5</a>	21-Apr-2013 23:03	32	
<a href="#">maven-metadata.xml.sha1</a>	21-Apr-2013 23:03	40	

Apache/2.2.17 (Ubuntu) Server at repository.primefaces.org Port 80



## Copiar para WEB-INF/lib

The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays a Java Web project named 'javaWebAula16' with its structure: JAX-WS Web Services, Deployment Descriptor: javaWebAula16, Java Resources, JavaScript Resources, build, WebContent, META-INF, and WEB-INF. Inside WEB-INF, there is a 'lib' folder containing several JAR files: bootstrap-1.10.jar, commons-beanutils-1.8.0.jar, commons-beanutils.jar, commons-collections-3.2.1.jar, commons-collections.jar, commons-digester-2.0.jar, commons-digester.jar, commons-logging-1.1.1.jar, commons-logging.jar, jsf-api.jar, jsf-impl.jar, jstl.jar, primefaces-5.0-sources.jar, primefaces-5.0.jar, standard.jar, faces-config.xml, and web.xml. The code editor on the right shows the XML code for 'vendas.xhtml'. The code includes HTML-like tags such as <h:form>, <h:panelGrid>, <h:outputText>, <p:calendar>, <h:commandButton>, and a <p:selectOneMenu> dropdown menu with several options like 'Patrícia', 'Genisson', and 'Pedro Igor'.

## WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
  <display-name>javaWebAula16</display-name>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>

  <context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>redmond</param-value>
  </context-param>

</web-app>
```

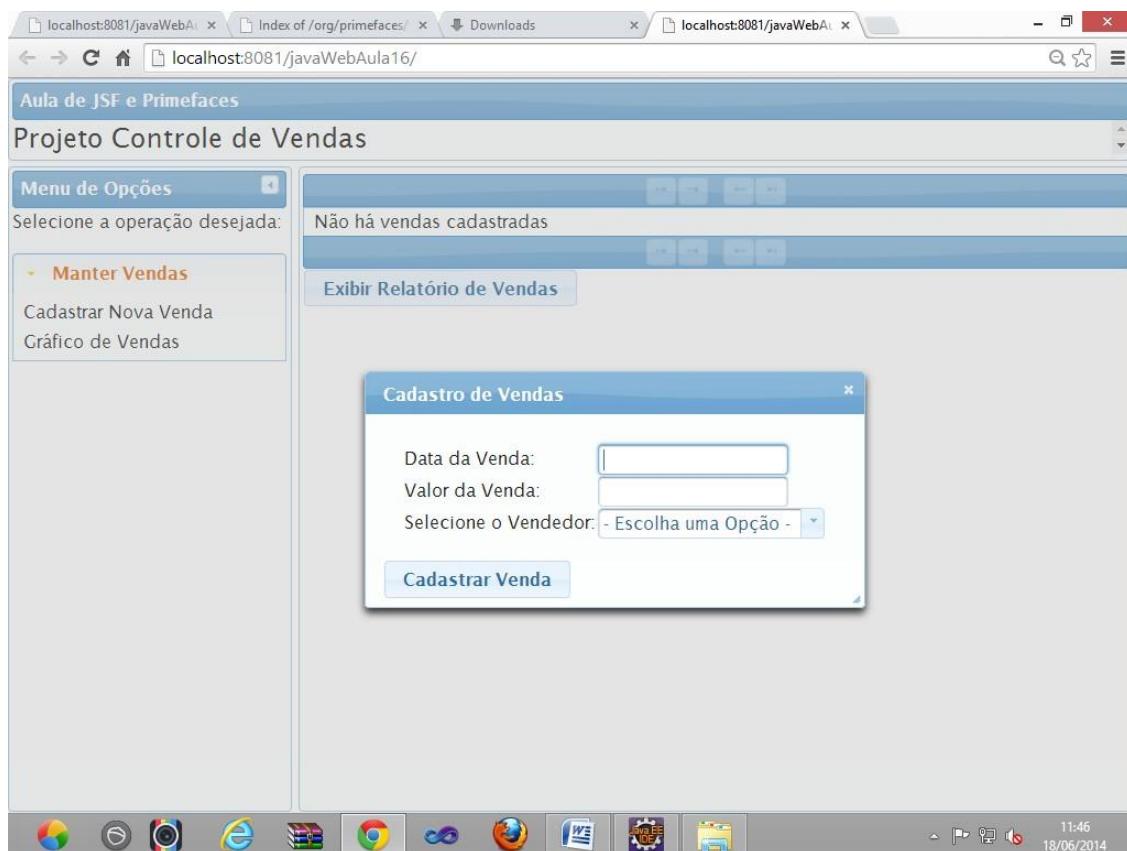


# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**



## HibernateUtil.java

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {

            sessionFactory = new AnnotationConfiguration().
                configure("config/mysql_hibernate.cfg.xml").
                buildSessionFactory();
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory
                creation failed." + ex);

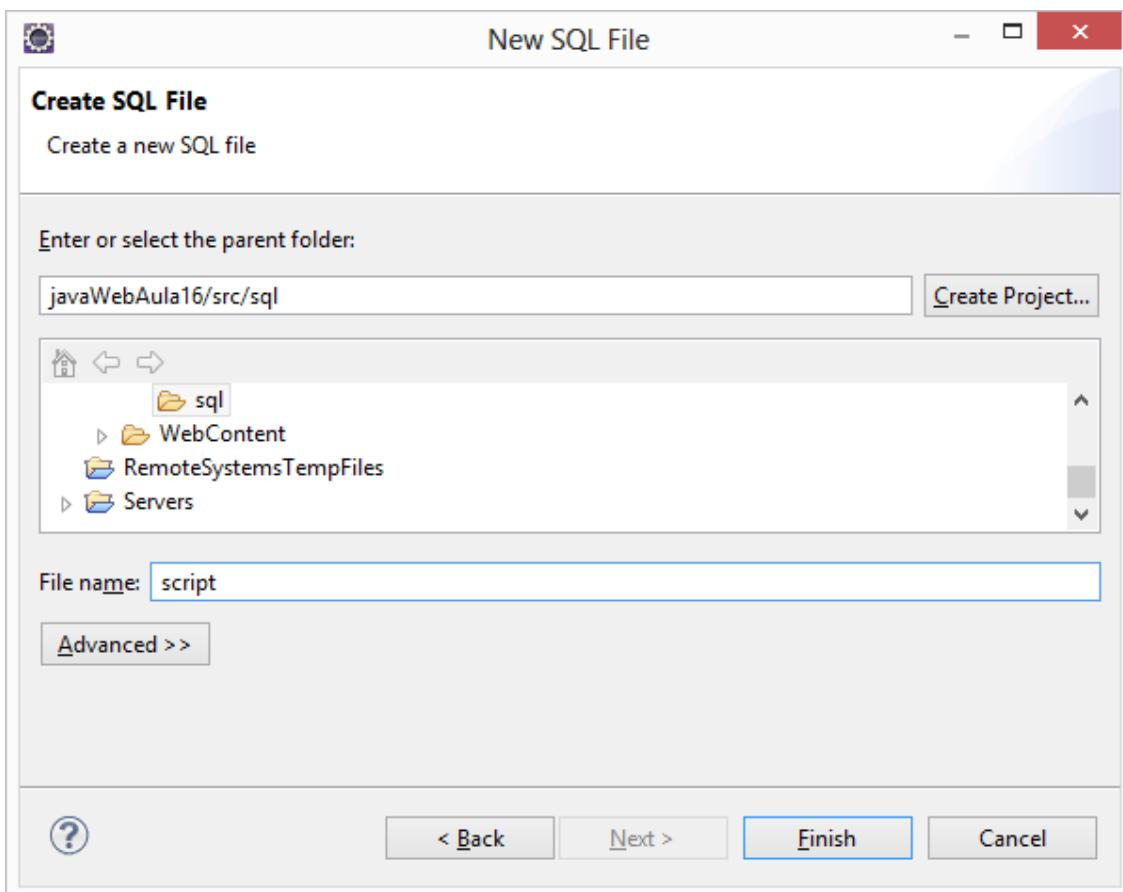
            throw new ExceptionInInitializerError(ex);
        }
    }
}
```



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

---

### Criando a base de dados...



```
drop database if exists aulaweb16;  
create database aulaweb16;  
use aulaweb16;  
  
create table venda(  
    idvenda          integer      auto_increment,  
    datavenda        date        not null,  
    valor            double      not null,  
    vendedor         varchar(50) not null,  
    primary key(idvenda));  
  
show tables;  
  
desc venda;
```

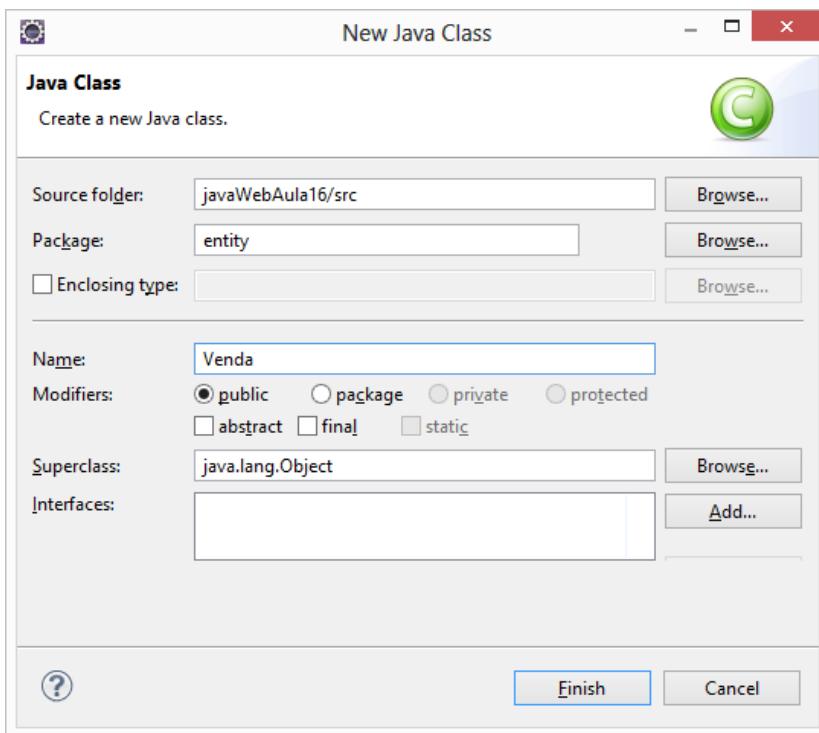


# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**



```
package entity;

import java.util.Date;

public class Venda {

    private Integer idVenda;
    private Date dataVenda;
    private Double valor;
    private String vendedor;

    public Venda() {
        // Construtor default
    }

    public Venda(Integer idVenda, Date dataVenda,
                 Double valor, String vendedor) {
        super();
        this.idVenda = idVenda;
        this.dataVenda = dataVenda;
        this.valor = valor;
        this.vendedor = vendedor;
    }

    @Override
    public String toString() {
        return "Venda [idVenda=" + idVenda + ", dataVenda="
               + dataVenda + ", valor=" + valor + ", vendedor="
               + vendedor + "]";
    }
}
```



```
public Integer getIdVenda() {
    return idVenda;
}

public void setIdVenda(Integer idVenda) {
    this.idVenda = idVenda;
}

public Date getDataVenda() {
    return dataVenda;
}

public void setDataVenda(Date dataVenda) {
    this.dataVenda = dataVenda;
}

public Double getValor() {
    return valor;
}

public void setValor(Double valor) {
    this.valor = valor;
}

public String getVendedor() {
    return vendedor;
}

public void setVendedor(String vendedor) {
    this.vendedor = vendedor;
}
-----
```

## Realizando o mapeamento da Classe JavaBean JPA - Java Persistence API

```
package entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.Temporal;
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula

26

```
import javax.persistence.TemporalType;

@Entity
@Table(name = "venda")
@NamedQueries(
{
    @NamedQuery(name="venda.listartodos",
        query="select v from Venda as v order by v.valor desc")
}
)
public class Venda {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idvenda")
    private Integer idVenda;

    @Temporal(TemporalType.DATE)
    @Column(name = "datavenda", nullable=false)
    private Date dataVenda;

    @Column(name = "valor", nullable=false)
    private Double valor;

    @Column(name = "vendedor", length=50, nullable=false)
    private String vendedor;

    public Venda() {
        // Construtor default
    }

    public Venda(Integer idVenda, Date dataVenda,
                 Double valor, String vendedor) {
        super();
        this.idVenda = idVenda;
        this.dataVenda = dataVenda;
        this.valor = valor;
        this.vendedor = vendedor;
    }

    @Override
    public String toString() {
        return "Venda [idVenda=" + idVenda + ", dataVenda="
            + dataVenda + ", valor=" + valor + ", vendedor="
            + vendedor + "]";
    }

    public Integer getIdVenda() {
        return idVenda;
    }
}
```



```
public void setIdVenda(Integer idVenda) {
    this.idVenda = idVenda;
}

public Date getDataVenda() {
    return dataVenda;
}

public void setDataVenda(Date dataVenda) {
    this.dataVenda = dataVenda;
}

public Double getValor() {
    return valor;
}

public void setValor(Double valor) {
    this.valor = valor;
}

public String getVendedor() {
    return vendedor;
}

public void setVendedor(String vendedor) {
    this.vendedor = vendedor;
}
```

---

### **mysql\_hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect
        </property>
        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/aulaweb16
        </property>
        <property name="hibernate.connection.username">
            root
        </property>
```



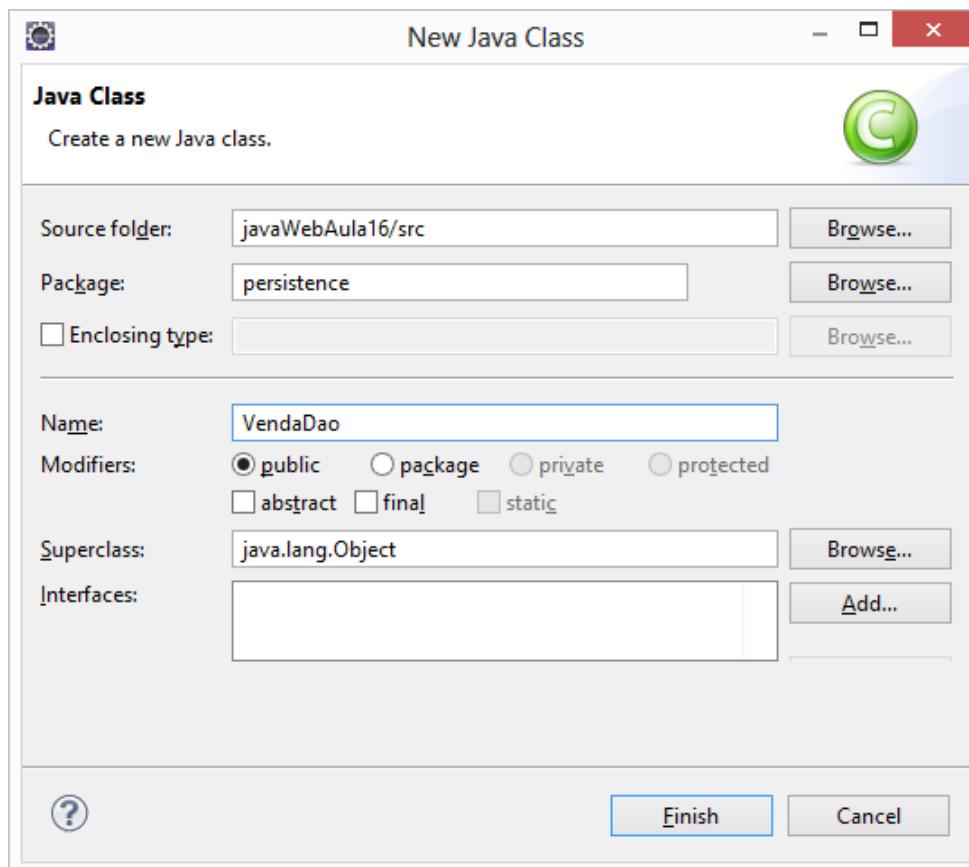
```
<property name="hibernate.connection.password"></property>

<property name="hibernate.show_sql">true</property>
<property name="hibernate.format_sql">true</property>

<mapping class="entity.Venda" />

</session-factory>
</hibernate-configuration>
```

### Classe de persistência...



```
package persistence;

import hibernate.HibernateUtil;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Venda;
```



```
public class VendaDao {  
  
    private Session session;  
    private Transaction transaction;  
    private Query query;  
  
    public void save(Venda v) throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
        transaction = session.beginTransaction();  
        session.saveOrUpdate(v); //gravar ou atualizar  
        transaction.commit();  
        session.close();  
    }  
  
    public void delete(Venda v) throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
        transaction = session.beginTransaction();  
        session.delete(v); //excluindo  
        transaction.commit();  
        session.close();  
    }  
  
    public Venda findById(Integer idVenda) throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
        Venda v = (Venda) session.get(Venda.class, idVenda);  
  
        session.close();  
        return v; //retornar o objeto  
    }  
  
    public List<Venda> findAll() throws Exception{  
        session = HibernateUtil.getSessionFactory().openSession();  
        query = session.getNamedQuery("venda.listartodos");  
        List<Venda> lista = query.list();  
  
        session.close();  
        return lista;  
    }  
}
```

---

## ManagedBeans (**gerenciador de beans**)

Classes utilizadas pelo framework JSF para realização de entrada / saída de dados bem como operações de controle (gravação, consulta, etc...)

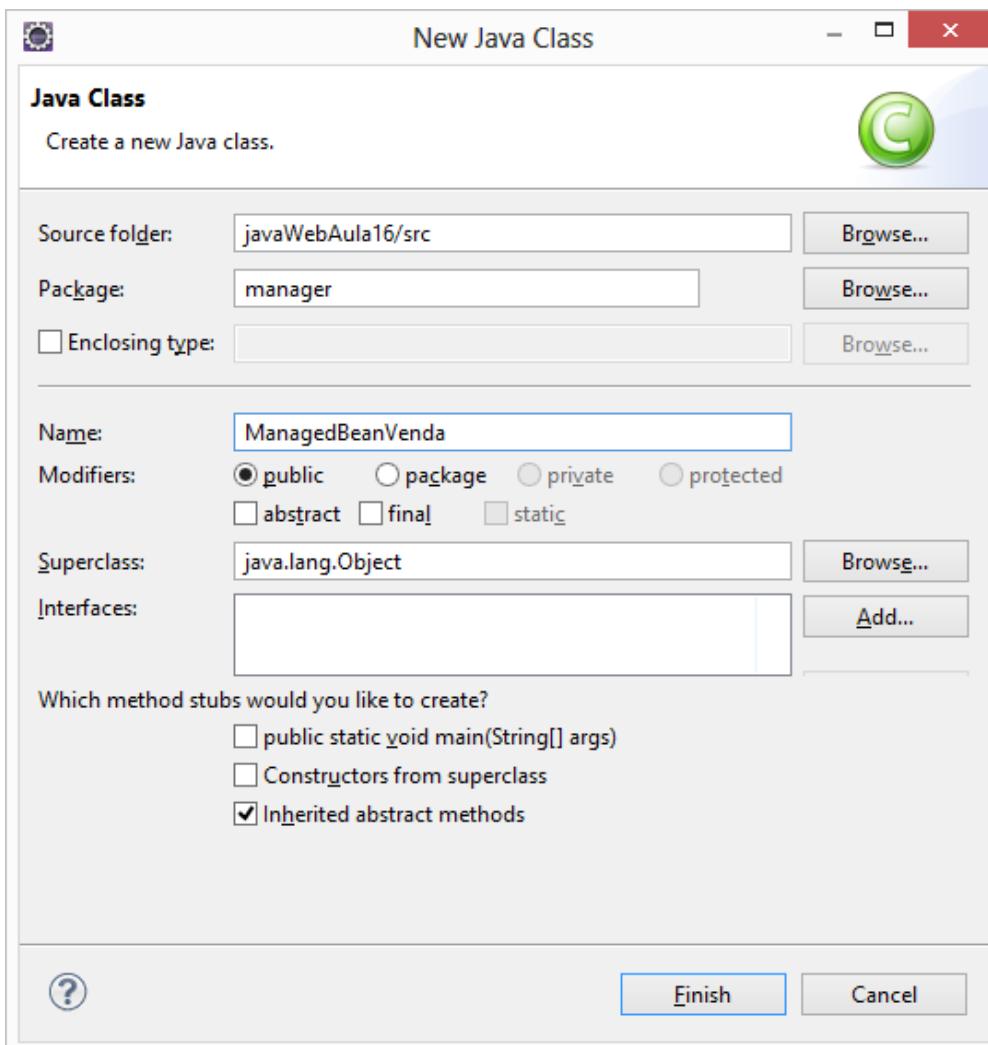


# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**



```
package manager;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

import entity.Venda;

@ManagedBean(name = "mbVenda")
@RequestScoped
public class ManagedBeanVenda {

    // Atributo para capturar os dados do formulário de cadastro
    private Venda venda; // null

    public ManagedBeanVenda() {
        venda = new Venda();
        // inicializando o atributo (espaço de memória)
    }
}
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula

26

```
//Método para gravação da venda
public void cadastrar(){

}

public Venda getVenda() {
    return venda;
}

public void setVenda(Venda venda) {
    this.venda = venda;
}
-----
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
      xmlns:p="http://primefaces.org/ui">

<h:head>

</h:head>
<h:body>

<p:layout fullPage="true">

    <p:layoutUnit position="north"
                   header="Aula de JSF e Primefaces">
        <h:outputText value="Projeto Controle de Vendas"
                      style="font-size: 20pt;"/>
    </p:layoutUnit>

    <p:layoutUnit position="west" header="Menu de Opções"
                  collapsible="true">

        <h:outputText value="Selecione a operação
                           desejada:"/>
        <br/><br/>

        <h:form>
            <p:panelMenu>
                <p:submenu label="Manter Vendas">
                    <p:menuitem value="Cadastrar Nova
                                   Venda" onclick="PF
                                   ('janelacadastro').show()"/>
                </p:submenu>
            </p:panelMenu>
        </h:form>
    </p:layoutUnit>
</p:layout>
```



```
<p:menuitem value="Gráfico de
Vendas" onclick="PF
('janelagrafico').show()"/>
</p:submenu>
</p:panelMenu>
</h:form>

</p:layoutUnit>

<p:layoutUnit position="center">
<h:form>

    <p:dataTable emptyMessage="Não há vendas
    cadastradas" paginator="true">
    </p:dataTable>

    <p:commandButton value="Exibir Relatório
    de Vendas"/>

</h:form>

</p:layoutUnit>

</p:layout>

<!-- Janela para cadastro de vendas -->
<p:dialog widgetVar="janelacadastro" header="Cadastro de Vendas"
modal="true">

    <h:form id="formcadastro">
        <h:panelGrid columns="2" style="padding: 20px;">
            <h:outputText value="Data da Venda:"/>
            <p:calendar
value="#{mbVenda.venda.dataVenda}"/>
            <h:outputText value="Valor da Venda:"/>
            <p:inputText value="#{mbVenda.venda.valor}"/>
            <h:outputText value="Selecione o Vendedor:"/>
            <p:selectOneMenu
value="#{mbVenda.venda.vendedor}">
                <f:selectItem itemValue="" itemLabel="-
Escolha uma Opção -"
noSelectionOption="true"/>
                <f:selectItem itemValue="Patricia"
itemLabel="Patricia Massaut" />
                <f:selectItem itemValue="Genisson"
itemLabel="Genisson Silva" />
                <f:selectItem itemValue="Pedro"
itemLabel="Pedro" />
            </p:selectOneMenu>
        </h:panelGrid>
    </h:form>
</p:dialog>
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

```
itemLabel="Pedro Igor" />

<f:selectItem itemValue="Rafael"
itemLabel="Rafael Ferreira" />

</p:selectOneMenu>

</h:panelGrid>

<p:commandButton value="Cadastrar Venda"
action="#{mbVenda.cadastrar}"
ajax="true" update=":formcadastro"/>

<p:message for="formcadastro"/>
<p:growl/>

</h:form>

</p:dialog>

<!-- Janela para gráfico de vendas -->
<p:dialog widgetVar="janelagrafico"
header="Gráfico de Vendas" modal="true">

</p:dialog>

</h:body>

</html>
```

---

```
package manager;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

import persistence.VendaDao;
import entity.Venda;

@ManagedBean(name = "mbVenda")
@RequestScoped
public class ManagedBeanVenda {

    // Atributo para capturar os dados do formulário de cadastro
    private Venda venda; // null
```



```
public ManagedBeanVenda() {
    venda = new Venda();
    // inicializando o atributo (espaço de memória)
}

//Método para gravação da venda
public void cadastrar(){

    String mensagem = null;

    try{

        VendaDao d = new VendaDao();

        d.save(venda);
        //gravar o atributo venda da Classe

        venda = new Venda();
        //novo espaço de memória ao atributo

        mensagem = "Venda cadastrada com sucesso.";
    }
    catch(Exception e){

        mensagem = "Ocorreu um erro -> "
                    + e.getMessage();
    }

    //Exibir a mensagem dentro do formcadastro
    FacesMessage msg = new FacesMessage(mensagem);
    FacesContext.getCurrentInstance().
        addMessage("formcadastro", msg);
}

public Venda getVenda() {
    return venda;
}

public void setVenda(Venda venda) {
    this.venda = venda;
}

}
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

localhost:8081/javaWebAula16/

Aula de JSF e Primefaces

Projeto Controle de Vendas

Menu de Opções

Selecionar a operação desejada:

- Manter Vendas
  - Cadastrar Nova Venda
  - Gráfico de Vendas

Cadastro de Vendas

Data da Venda: 18/06/14

Valor da Venda: 1500

Selecionar o Vendedor: - Escolha uma Opção -

- Patricia Massaut
- Genisson Silva
- Pedro Igor
- Rafael Ferreira

Cadastrar Venda

Não há vendas cadastradas

Exibir Relatório de Vendas

localhost:8081/javaWebAula16/

Aula de JSF e Primefaces

Projeto Controle de Vendas

Menu de Opções

Selecionar a operação desejada:

- Manter Vendas
  - Cadastrar Nova Venda
  - Gráfico de Vendas

Cadastro de Vendas

Data da Venda:

Valor da Venda:

Selecionar o Vendedor: - Escolha uma Opção -

Cadastrar Venda

Venda cadastrada com sucesso.

Venda cadastrada com sucesso.



## No banco de dados

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
mysql> select * from venda;
+----+-----+-----+-----+
| idvenda | datavenda | valor | vendedor |
+----+-----+-----+-----+
| 1 | 2014-06-18 | 1500 | Patricia |
| 2 | 2014-06-18 | 2000 | Genisson |
| 3 | 2014-06-18 | 2000 | Rafael |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

## Exibindo a listagem de vendas

```
package manager;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;

import persistence.VendaDao;
import entity.Venda;

@ManagedBean(name = "mbVenda")
@RequestScoped
public class ManagedBeanVenda {

    // Atributo para capturar os dados do formulário de cadastro
    private Venda venda; // null

    //Atributo para gerar a listagem com as vendas que serão
    //exibidas na datatable
    private DataModel listagemVendas;

    public ManagedBeanVenda() {
        venda = new Venda();
        // inicializando o atributo (espaço de memória)
    }

    //Método para gravação da venda
    public void cadastrar(){

        String mensagem = null;
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula

26

```
try{

    VendaDao d = new VendaDao();
    d.save(venda); //gravar o atributo venda da Classe

    venda = new Venda(); //novo espaço
    //de memória ao atributo
    mensagem = "Venda cadastrada com sucesso.";
}

catch(Exception e){
    mensagem = "Ocorreu um erro -> " + e.getMessage();
}

//Exibir a mensagem dentro do formcadastro
FacesMessage msg = new FacesMessage(mensagem);
FacesContext.getCurrentInstance().addMessage
    ("formcadastro", msg);
}

public Venda getVenda() {
    return venda;
}

public void setVenda(Venda venda) {
    this.venda = venda;
}

public DataModel getListagemVendas() {

    try{

        VendaDao d = new VendaDao();
        //classe de persistência

        //carregando a lista de vendas
        //dentro do componente DataModel do JSF
        listagemVendas = new ListDataModel(d.findAll());

    }

    catch(Exception e){
        e.printStackTrace();
        //exibir o erro no console do servidor
    }

    return listagemVendas;
}

public void setListagemVendas(DataModel listagemVendas) {
    this.listagemVendas = listagemVendas;
}
}
```



### Exibindo os dados na DataTable...

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
      xmlns:p="http://primefaces.org/ui">

    <h:head>
    </h:head>
    <h:body>

        <p:layout fullPage="true">

            <p:layoutUnit position="north"
                           header="Aula de JSF e Primefaces">
                <h:outputText value="Projeto Controle de Vendas"
                              style="font-size: 20pt;"/>
            </p:layoutUnit>

            <p:layoutUnit position="west" header="Menu de Opções"
                           collapsible="true">

                <h:outputText value="Selecione a operação
                                  desejada:"/>
                <br/><br/>

                <h:form>
                    <p:panelMenu>
                        <p:submenu label="Manter Vendas">
                            <p:menuitem value="Cadastrar Nova
                                Venda" onclick="PF
                                    ('janelacadastro').show()"/>
                            <p:menuitem value="Gráfico de
                                Vendas" onclick="PF
                                    ('janelagrafico').show()"/>
                        </p:submenu>
                    </p:panelMenu>
                </h:form>

            </p:layoutUnit>

            <p:layoutUnit position="center">

                <h:form id="formconsulta">

                    <p:dataTable emptyMessage="Não há vendas
                        cadastradas" paginator="true" rows="10"
                        value="#{mbVenda.listagemVendas}" var="v">
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula

26

```
<f:facet name="header">
    <h:outputText value="Listagem de
        Vendas realizadas"/>
</f:facet>

<p:column headerText="Código">
    <h:outputText
        value="#{v.idVenda}"/>
</p:column>

<p:column headerText="Valor da venda">
    <h:outputText value="#{v.valor}">
        <f:convertNumber
            type="currency"/>
    </h:outputText>
</p:column>

<p:column headerText="Data da Venda">
    <h:outputText
        value="#{v.dataVenda}">
        <f:convertDateTime
            pattern="EE dd/MM/yyyy"/>
    </h:outputText>
</p:column>

<p:column headerText="Vendedor">
    <h:outputText
        value="#{v.vendedor}"/>
</p:column>

<p:column headerText="Excluir"
    width="10%"
    style="text-align: center;">
    <p:commandButton icon="ui-icon-
        trash"/>
</p:column>

</p:dataTable>

<p:commandButton value="Exibir Relatório
    de Vendas"/>

</h:form>

</p:layoutUnit>

</p:layout>

<!-- Janela para cadastro de vendas -->
<p:dialog widgetVar="janelacadastro" header="Cadastro
    de Vendas" modal="true">
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula

26

```
<h:form id="formcadastro">

    <h:panelGrid columns="2" style="padding: 20px;">

        <h:outputText value="Data da Venda:"/>
        <p:calendar
            value="#{mbVenda.venda.dataVenda}" />

        <h:outputText value="Valor da Venda:"/>
        <p:inputText value="#{mbVenda.venda.valor}" />

        <h:outputText value="Selecione o Vendedor:"/>
        <p:selectOneMenu
            value="#{mbVenda.venda.vendedor}">
            <f:selectItem itemValue="" itemLabel="-
                Escolha uma Opção -"
                noSelectionOption="true"/>
            <f:selectItem itemValue="Patricia"
                itemLabel="Patricia Massaut" />
            <f:selectItem itemValue="Genisson"
                itemLabel="Genisson Silva" />
            <f:selectItem itemValue="Pedro"
                itemLabel="Pedro Igor" />
            <f:selectItem itemValue="Rafael"
                itemLabel="Rafael Ferreira" />
        </p:selectOneMenu>

    </h:panelGrid>

    <p:commandButton value="Cadastrar Venda"
        action="#{mbVenda.cadastrar}"
        ajax="true" update=":formcadastro,
            :formconsulta"/>

    <br/><br/>
    <p:message for="formcadastro"/>
    <p:growl/>

</h:form>

</p:dialog>

<!-- Janela para gráfico de vendas -->
<p:dialog widgetVar="janelagrafico"
    header="Gráfico de Vendas" modal="true">

</p:dialog>

</h:body>

</html>
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

Screenshot of a Java web application showing a list of sales and a sidebar menu.

**Menu de Opções:**

- Manter Vendas
  - Cadastrar Nova Venda
  - Gráfico de Vendas

**Listagem de Vendas realizadas:**

Código	Valor da venda	Data da Venda	Vendedor	Excluir
8	R\$ 6.000,00	Qui 19/06/2014	Rafael	
12	R\$ 5.000,00	Qua 18/06/2014	Pedro	
4	R\$ 5.000,00	Qua 18/06/2014	Rafael	
5	R\$ 4.000,00	Qui 19/06/2014	Patricia	
11	R\$ 3.500,00	Qui 12/06/2014	Rafael	
7	R\$ 2.500,00	Qui 19/06/2014	Pedro	
3	R\$ 2.000,00	Qua 18/06/2014	Rafael	
9	R\$ 2.000,00	Qua 25/06/2014	Rafael	
2	R\$ 2.000,00	Qua 18/06/2014	Genisson	
10	R\$ 1.500,00	Sex 13/06/2014	Pedro	

**Exibir Relatório de Vendas**

Screenshot of the same Java web application, showing a modal dialog for adding a new sale.

**Menu de Opções:**

- Manter Vendas
  - Cadastrar Nova Venda
  - Gráfico de Vendas

**Listagem de Vendas realizadas:**

Código	Valor da venda	Data da Venda	Vendedor	Excluir
8	R\$ 6.000,00	Qui 19/06/2014	Rafael	
12	R\$ 5.000,00	Qua 18/06/2014	Pedro	
4	R\$ 5.000,00	Qua 18/06/2014	Rafael	
5				
11				
7				
3				
9				
2	R\$ 2.000,00	Qua 18/06/2014	Genisson	
10	R\$ 1.500,00	Sex 13/06/2014	Pedro	

**Cadastro de Vendas**

Data da Venda:

Valor da Venda:

Selecione o Vendedor:

**Cadastrar Venda**



- Excluindo as Vendas

```
<p:column headerText="Excluir" width="10%"  
          style="text-align: center;">  
  
    <p:commandButton icon="ui-icon-trash"  
                      action="#{mbVenda.excluir}"  
                      update=":formconsulta" ajax="true"/>  
  
</p:column>
```

---

## getRowIndex

Método da Classe **DataModel** que resgata o objeto referente à linha clicada na tabela pelo usuário.

```
public void excluir(){  
  
    String mensagem = null;  
  
    try{  
  
        //Resgatar o objeto Venda clicado pelo usuário no botão  
        Venda v = (Venda) listagemVendas.getRowData();  
  
        VendaDao d = new VendaDao();  
        d.delete(v); //excluir  
  
        mensagem = "Venda excluída com sucesso";  
    }  
    catch(Exception e){  
  
        mensagem = "Ocorreu um erro -> " + e.getMessage();  
    }  
  
    //Exibir a mensagem no formconsulta  
    FacesMessage msg = new FacesMessage(mensagem);  
    FacesContext.getCurrentInstance().addMessage("formconsulta", msg);  
}
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

The screenshot shows a Java web application running on localhost:8081. The title bar says "localhost:8081/javaWebAula16". The main content area has a blue header "Projeto Controle de Vendas". On the left, there's a sidebar with a "Menu de Opções" button and a "Manter Vendas" link. The main area is titled "Listagem de Vendas realizadas" and contains a table with 10 rows of sales data:

Código	Valor da venda	Data da Venda	Vendedor	Excluir
8	R\$ 6.000,00	Qui 19/06/2014	Rafael	
13	R\$ 5.000,00	Qua 18/06/2014	Genisson	
4	R\$ 5.000,00	Qua 18/06/2014	Rafael	
5	R\$ 4.000,00	Qui 19/06/2014	Patricia	
11	R\$ 3.500,00	Qui 12/06/2014	Rafael	
7	R\$ 2.500,00	Qui 19/06/2014	Pedro	
3	R\$ 2.000,00	Qua 18/06/2014	Rafael	
9	R\$ 2.000,00	Qua 25/06/2014	Rafael	
2	R\$ 2.000,00	Qua 18/06/2014	Genisson	
10	R\$ 1.500,00	Sexta 13/06/2014	Pedro	

Below the table, there are buttons for navigating between pages (1, 2, etc.) and a link "Exibir Relatório de Vendas". A message box at the bottom says "Venda excluída com sucesso!" (Sale deleted successfully!).

The screenshot shows the Eclipse IDE interface with the title "Java EE - javaWebAula16/src/manager/ManagedBeanVenda.java - Eclipse". The left panel shows the "Project Explorer" with the project structure. The right panel shows the code editor for "ManagedBeanVenda.java". The code is as follows:

```
Java EE - javaWebAula16/src/manager/ManagedBeanVenda.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
ManagedBeanVenda.java
public void
String
try{
from
order by
Hibernate:
delete
from
where
}
catch(E
select
//Exibi
FacesMessage msg = new FacesMessage("Venda excluída com sucesso!");
FacesContext.getCurrentInstance().addMessage("formconsulta", msg);
}

public Venda getVenda() {
    return varda;
}

public void setVenda(Venda varda) {
    this.varda = varda;
}
```

The code editor shows code completion suggestions for "venda" and "Hibernate" methods.



## iReport

```
select
    vendedor,
    sum(valor) as total
from venda
group by vendedor
order by total desc;
```

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> select
    -> vendedor,
    -> sum(valor) as total
    -> from venda
    -> group by vendedor
    -> order by total desc;
+-----+-----+
| vendedor | total |
+-----+-----+
| Rafael   | 18500 |
| Genisson | 7000  |
| Patricia | 5500  |
| Pedro    | 5000  |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Jaspersoft iReport Designer 5.5.0

New file

A report is used to display values from a data source like a database or an XML file. Select a starting point for your report or choose the wizard to guide you.

Report

Style

Chart Theme

Resource Bundle

Other file Types

Blank A4

Blank A4 Landscape

Blank Letter

Blank Letter Landscape

Open this Template

Launch Report Wizard

Cancel



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

New

**Passos**

1. Escolha o modelo
2. **Name and location**
3. Finish

**Name and location**

Report name: relatoriovendas

Location: C:\Users\Sergio Mendes\Desktop

File: C:\Users\Sergio Mendes\Desktop\relatoriovendas.jrxml

< Voltar  Finalizar Cancelar Ajuda

New

**Passos**

1. Escolha o modelo
2. Name and location
3. **Finish**

**Finish**

**Congratulations!**

You have successfully created a new report.  
Click finish to generate it.

< Voltar  Finalizar Cancelar Ajuda



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

aula13

Report Inspector Report Datasources relatoriovendas.jrxml

Designer XML Preview DejaVu Sans 3 4 5 6 7 8

Paleta Report Elements

- Break
- Chart
- Crosstab
- Ellipse
- Frame
- Html
- Image
- Barcode
- Generic Element
- List
- Spider Chart
- Table
- Line
- Map
- Rectangle
- Round Rectangle
- Sort
- Static Text
- Subreport
- Text Field
- Tools
- Callout
- Current date
- Page number
- Page X of Y
- Percentage

Title

Page Header

Column Header

Detail 1

Column Footer

Page Footer

Summary

Report Inspector

relatoriovendas

- Styles
- Parameters
- Fields
- f/x Variables
- Scriptlets
- Title
- Page Header
- Column Header
- Detail 1
- Column Footer
- Page Footer
- Last Page Footer
- Summary
- No Data
- Background

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

aula13

Pesquisar (Ctrl+I)

15:56 18/06/2014

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

aula13

Report Inspector Connections / Datasources

Name	Datasource type	Default
Empty datasource	Empty data source	<input type="checkbox"/>
Sample Database (HS...)	Sample Database Conn...	<input type="checkbox"/>
aula13	Database JDBC conn...	<input checked="" type="checkbox"/>

New Modify Delete Set as default Import Export Close

Report Inspector

relatoriovendas

- Styles
- Parameters
- Fields
- f/x Variables
- Scriptlets
- Title
- Page Header
- Column Header
- Detail 1
- Column Footer
- Page Footer
- Last Page Footer
- Summary
- No Data
- Background

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

aula13

Pesquisar (Ctrl+I)

15:56 18/06/2014



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

aula13

Report Inspector

Datasource

Select the datasource type

Database JDBC connection

Netbeans Database JDBC connection

XML file datasource

JavaBeans set datasource

File CSV datasource

JRDataSourceProvider

Custom JRDatasource

Empty data source

Hibernate connection

Spring loaded Hibernate connection

EJBQL connection

XMLA Server

Mondrian OLAP connection

Query Executer mode

Microsoft Excel (xls) data source

Microsoft Excel 2007 (xlsx) data source

MongoDB Connection

Remote XML file datasource

JSON datasource

Sample Database Connection

Hadoop Hive Connection

Test Next > Cancel

Paleta

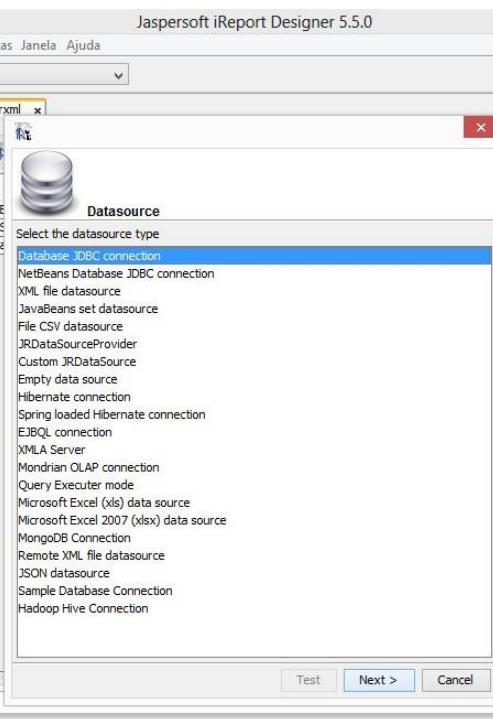
Report Elements

- Break
- Chart
- Crosstab
- Ellipse
- Frame
- HTML
- Image
- Barcode
- Generic Element
- List
- Spider Chart
- Table
- Line
- Map
- Rectangle
- Round Rectangle
- Sort
- Static Text
- Subreport
- Text Field

Tools

- Callout
- Current date
- Page number
- Page X of Y
- Percentage

15:56 18/06/2014



Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

aula13

Report Inspector

Database JDBC connection

Name: aula16

JDBC Driver: MySQL (com.mysql.jdbc.Driver)

JDBC URL: jdbc:mysql://localhost/aulaweb16

Credentials

Username: root

Password:

Save password

ATTENTION! Passwords are stored in clear text. If you don't specify a password now, iReport will ask you for one only when required and will not save it.

Test Save Cancel

Paleta

Report Elements

- Break
- Chart
- Crosstab
- Ellipse
- Frame
- HTML
- Image
- Barcode
- Generic Element
- List
- Spider Chart
- Table
- Line
- Map
- Rectangle
- Round Rectangle
- Sort
- Static Text
- Subreport
- Text Field

Tools

- Callout
- Current date
- Page number
- Page X of Y
- Percentage

15:57 18/06/2014





# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

aula16

Report Inspector

relatoriovendas

Styles

Parameters

Fields

Variables

Scriptlets

Title

Page Header

Column Header

Detail 1

Column Footer

Page Footer

Last Page Footer

Summary

No Data

Background

Designer XML Preview

DejaVu Sans

Font Size: 3

Font Style: A A B I U

Page Inspector

Paleta

Report Elements

- Break
- Chart
- Crosstab
- Ellipse
- Frame
- Image
- Barcode
- Generic Element
- List
- Spider Chart
- Table
- Line
- Map
- Rectangle
- Round Rectangle
- Sort
- Static Text
- Subreport
- Text Field
- Tools
- Callout
- Current date
- Page number

Report query

Report query JavaBean Datasource DataSource Provider CSV Datasource Excel Datasource

Query language SQL

```
select
    vendedor,
    sum(valor) as total
from venda
group by vendedor
order by total desc;
```

Load query Save query

Drag a parameter into the query to a parameter. Hold CTL to add the parameter as query chunk.

Available parameters

New parameter

Ready

Automatically Retrieve Fields Read Fields Query d... Send to cli...

Field name	Field type	Description
vendedor	java.lang.String	
total	java.lang.Double	

Filter expression... Sort options... Preview data OK Cancel



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

relatoriovendas.jrxml

Designer XML Preview SansSerif 14 A A b i u s

Report Inspector

Vendedor Total de vendas

\$F{vendedor} \$F{total}

Page Header Column Footer

Title Page Footer

Page Footer Summary

Detail 1

Column Header

label vendedor label total

label total

Column Footer

Page Footer

Last Page Footer

Summary

No Data

Background

Paleta

Chart Crosstab

Edit expression

Field pattern

Padding And Borders

Hyperlink

Copiar Ctrl+C

Recortar Ctrl+X

Colar Ctrl+V

Excluir Excluir

Copy format

Paste format

Transform to

Group selected element(s)

Ungroup selected element(s)

Bring To Front

Bring Forward

Send Backward

Send To Back

Align

Size

Position

Horizontal Spacing

Vertical Spacing

Organize As Table Ctrl+Shift+O

Send to layer

PDF 508 Tags

16:03 18/06/2014

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda

relatoriovendas.jrxml

Designer XML Preview SansSerif 14 A A b i u s

Report Inspector

Vendedor

\$F{vendedor}

Category Sample

Number R\$ 1.234,43

Date

Time

Currency

Decimal places: 2

Percentage

Scientific

Custom Format

Pattern #,##0.00

Apply Cancel

Paleta

Chart Crosstab Ellipse Frame Html Image Barcode Generic Element List Spider Chart Table Line Map Rectangle Round Rectangle Sort Static Text Subreport Text Field Tools Callout Current date Page number Page X of Y Percentage Total pages Web Framework

16:03 18/06/2014



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda aula16

relatoriovendas.jrxml

Report Inspector

Designer XML Preview SansSerif 10 A B I U

Relatório de Vendas por Vendedor

new java.util.Date()

Page Header

Vendedor Total de vendas

\$F{vendedor} \$F{total}

Column Footer

Page Footer

Summary

Detail 1

label vendedor

label total

label static text

new java.util.D...

Parameters

Fields

fx Variables

Scriptlets

Title

Background

Page Header

Column Header

No Data

Paleta

- Chart
- Crosstab
- Ellipse
- Frame
- Image
- Barcode
- Generic Element
- List
- Spider Chart
- Table
- Line
- Map
- Rectangle
- Round Rectangle
- Sort
- label Static Text
- Subreport
- Text Field
- Tools
- Callout
- Current date
- Page number
- Page X of Y
- Percentage
- Total pages
- Web Framework

Salvar todos concluído.

16:04 18/06/2014

Jaspersoft iReport Designer 5.5.0

Arquivo Editar Exibir Format Preview Ferramentas Janela Ajuda aula16

relatoriovendas.jrxml

Designer XML Preview 100%

Relatório de Vendas por Vendedor

Quarta-feira 18 Junho 2014

Vendedor	Total de vendas
Rafael	R\$ 18.500,00
Genisson	R\$ 7.000,00
Patricia	R\$ 5.500,00
Pedro	R\$ 5.000,00

16:05 18/06/2014

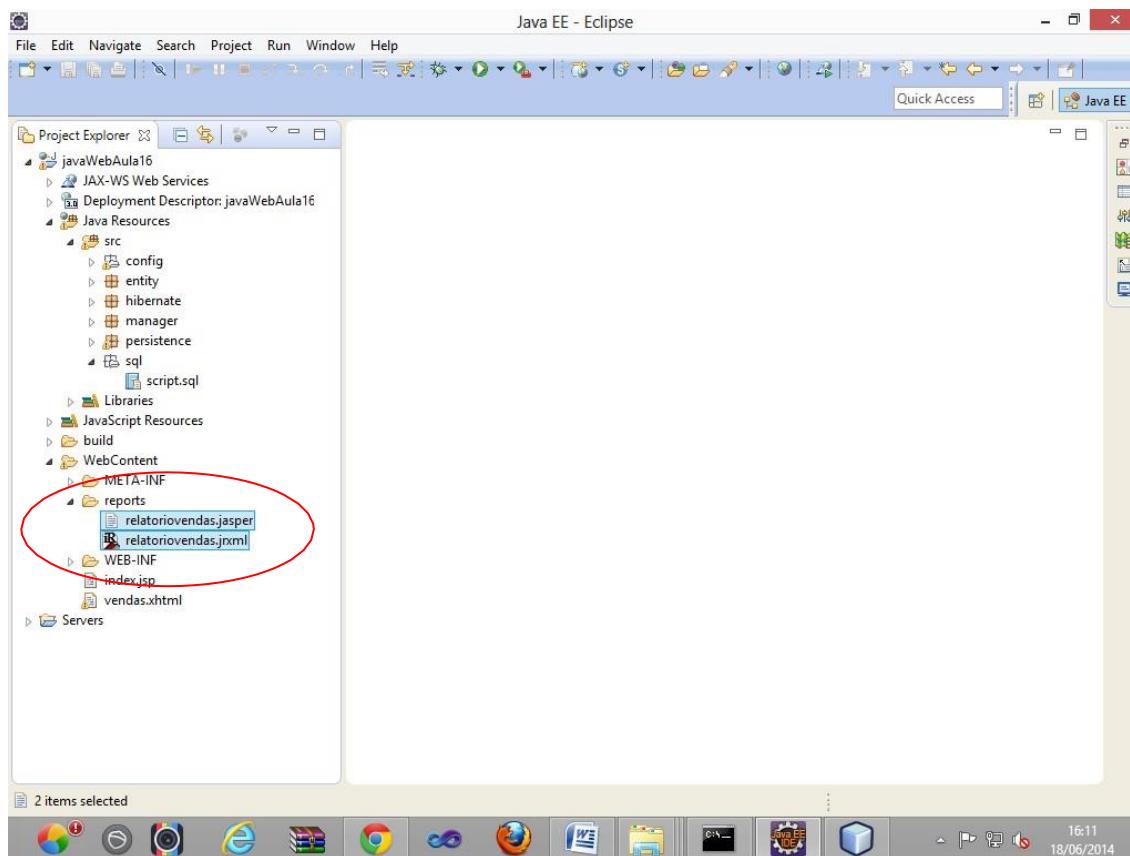


# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**



## Gerando o relatório no JSF...

```
<p:commandButton  
    value="Exibir Relatório de Vendas"  
    ajax="false"  
    action="#{mbVenda.relatorio}"/>
```

---

```
-----  
  
public void relatorio(){  
  
try{  
  
    //Caminho do Relatorio no WebContent  
    InputStream path = FacesContext.getCurrentInstance().  
        getExternalContext().  
  
    getResourceAsStream("/reports/relatoriovendas.jasper");  
  
    //Transformar o relatorio em arquivo PDF passando  
    //a conexão do banco  
    byte[] pdf = JasperRunManager.runReportToPdf(path, null,  
        HibernateUtil.getSessionFactory().openSession().connection());
```



```
HttpServletResponse response = (HttpServletResponse)
    FacesContext.getCurrentInstance().  
  
getExternalContext().getResponse();  
  
//Download  
ServletOutputStream out = response.getOutputStream();  
out.write(pdf);  
out.flush();  
out.close();  
}  
catch(Exception e){  
    e.printStackTrace(); //mensagem no tomcat  
}  
}
```

## Executando...

Código	Valor da venda	Data da Venda	Vendedor	Excluir
8	R\$ 6.000,00	Qui 19/06/2014	Rafael	
13	R\$ 5.000,00	Qua 18/06/2014	Genisson	
4	R\$ 5.000,00	Qua 18/06/2014	Rafael	
5	R\$ 4.000,00	Qui 19/06/2014	Patricia	
11	R\$ 3.500,00	Qui 12/06/2014	Rafael	
7	R\$ 2.500,00	Qui 19/06/2014	Pedro	
3	R\$ 2.000,00	Qua 18/06/2014	Rafael	
9	R\$ 2.000,00	Qua 25/06/2014	Rafael	
2	R\$ 2.000,00	Qua 18/06/2014	Genisson	
10	R\$ 1.500,00	Sexta 13/06/2014	Pedro	



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

The screenshot shows a web application running on localhost:8081. The URL in the address bar is localhost:8081/javaWebAula16/vendas.jsf. The page title is "Relatório de Vendas por Vendedor" (Sales Report by Salesperson) and the subtitle is "Quarta-feira 18 Junho 2014". The content is a table with two columns: "Vendedor" (Salesperson) and "Total de vendas" (Total sales). The data is as follows:

Vendedor	Total de vendas
Rafael	R\$ 18.500,00
Genisson	R\$ 7.000,00
Patricia	R\$ 5.500,00
Pedro	R\$ 5.000,00

Criando a consulta de somatorio de vendas em HQL - Hibernate Query Language.

```
@Entity
@Table(name = "venda")

@NamedQueries(
{
    @NamedQuery(name="venda.listartodos",
                query="select v from Venda as v
                      order by v.valor desc"),

    @NamedQuery(name="venda.somatoriovendas",
                query="select v.vendedor,
                           sum(v.valor)
                     from Venda as v
                   group by v.vendedor")
}
)

public class Venda {
```



```
package persistence;

import hibernate.HibernateUtil;

import java.util.List;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Venda;

public class VendaDao {

    private Session session;
    private Transaction transaction;
    private Query query;

    public void save(Venda v) throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.saveOrUpdate(v); //gravar ou atualizar
        transaction.commit();
        session.close();
    }

    public void delete(Venda v) throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.delete(v); //excluindo
        transaction.commit();
        session.close();
    }

    public Venda findById(Integer idVenda) throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();
        Venda v = (Venda) session.get(Venda.class, idVenda);

        session.close();
        return v; //retornar o objeto
    }

    public List<Venda> findAll() throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();
        query = session.getNamedQuery("venda.listartodos");
        List<Venda> lista = query.list();

        session.close();
        return lista;
    }
}
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

```
public List<Object[]> somatorioVendas() throws Exception{
    session = HibernateUtil.getSessionFactory().openSession();
    query = session.getNamedQuery("venda.somatoriovendas");
    List<Object[]> lista = query.list();

    session.close();
    return lista;
}

-----
<!-- Janela para gráfico de vendas -->
<p:dialog widgetVar="janelagrafico" header="Gráfico de Vendas"
modal="true">

    <h:form>

        <p:pieChart shadow="true" showDataLabels="true"
            title="Gráfico de vendas por Vendedor"
            value="#{mbVenda.dadosGrafico}"
            style="width: 500px; height: 500px;"
            legendPosition="w"
        />

    </h:form>
</p:dialog>
-----
//Atributo para gerar os dados do gráfico
private PieChartModel dadosGrafico;

public PieChartModel getDadosGrafico() {
    try{
        dadosGrafico = new PieChartModel();
        //inicializando o atributo

        VendaDao d = new VendaDao();
        for(Object[] registro : d.somatorioVendas()){

            String vendedor = (String) registro[0];
            Double total      = (Double) registro[1];
    
```



# Java WebDeveloper - BRQ

Quarta-feira, 18 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Persistência de dados com Hibernate e JPA.

Aula  
**26**

```
//escrevendo os dados no grafico
dadosGrafico.set(vendedor, total);

}

catch(Exception e){
    e.printStackTrace(); //imprimir o erro no tomcat
}

return dadosGrafico;
}

public void setDadosGrafico(PieChartModel dadosGrafico) {
    this.dadosGrafico = dadosGrafico;
}
```

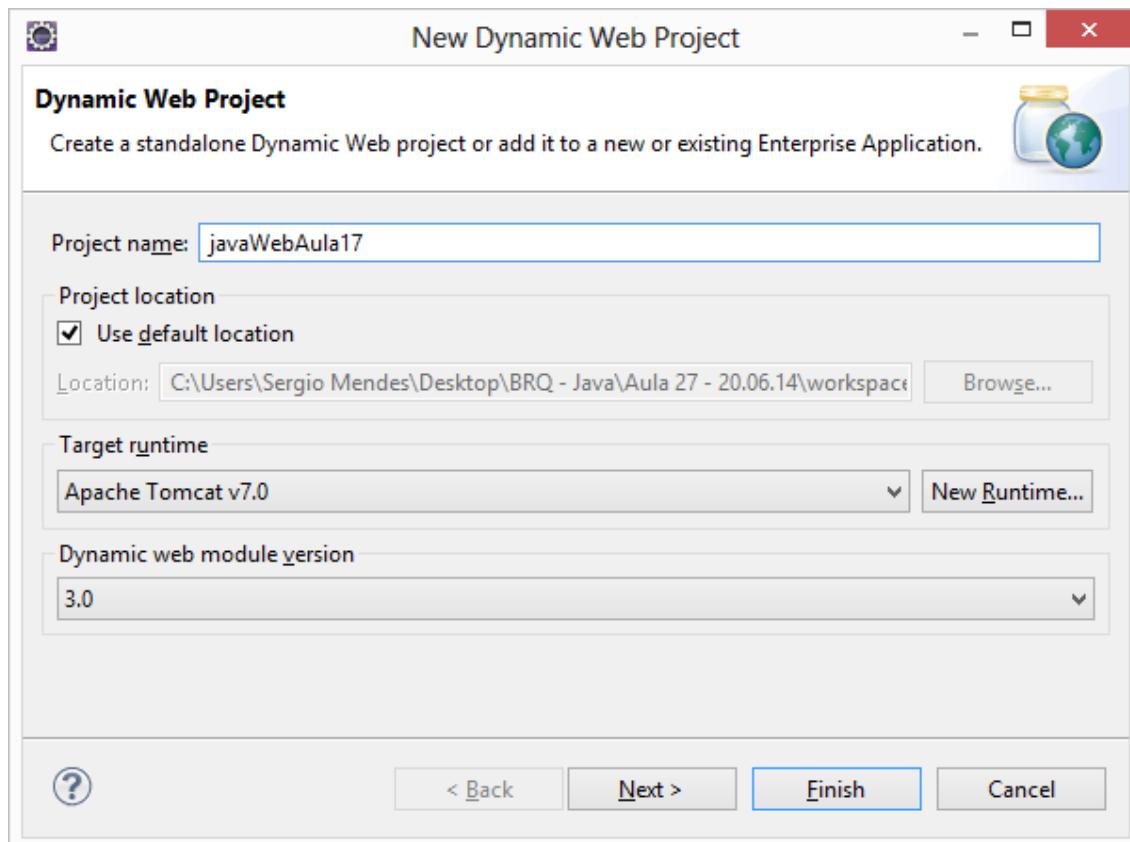
Exibindo...

The screenshot shows a web browser window with the URL `localhost:8081/javaWebAula16/`. The page title is "Aula de JSF e Primefaces" and the subtitle is "Projeto Controle de Vendas". On the left, there is a sidebar menu titled "Menu de Opções" with the option "Manter Vendas" expanded, showing "Cadastrar Nova Venda" and "Gráfico de Vendas". The main content area has a title "Gráfico de Vendas" and a subtitle "Gráfico de vendas por Vendedor". It displays a pie chart with three segments: Genisson (51%), Pedro (14%), and Patricia (15%). To the right of the chart is a table titled "Vendedores" with the following data:

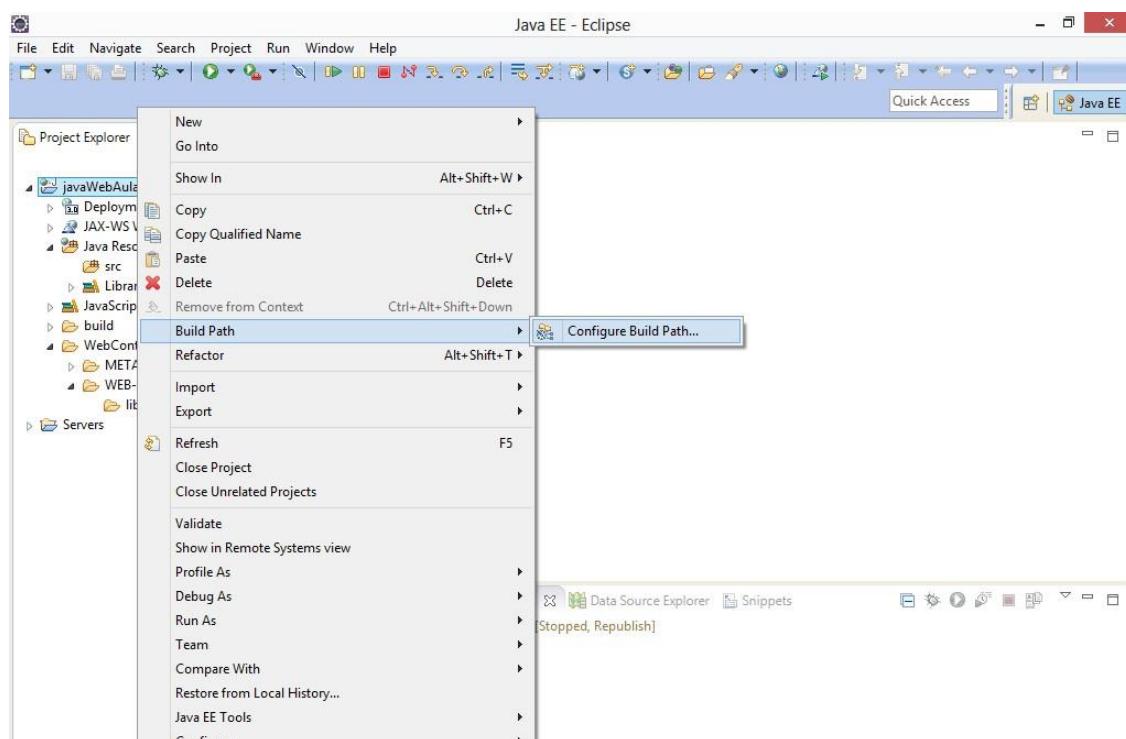
Vendedor	Excluir
Rafael	[Delete icon]
Genisson	[Delete icon]
Rafael	[Delete icon]
Patricia	[Delete icon]
Rafael	[Delete icon]
Pedro	[Delete icon]
Rafael	[Delete icon]
Rafael	[Delete icon]
Genisson	[Delete icon]
Pedro	[Delete icon]



Criando o projeto...



## Adicionando o JavaServerFaces



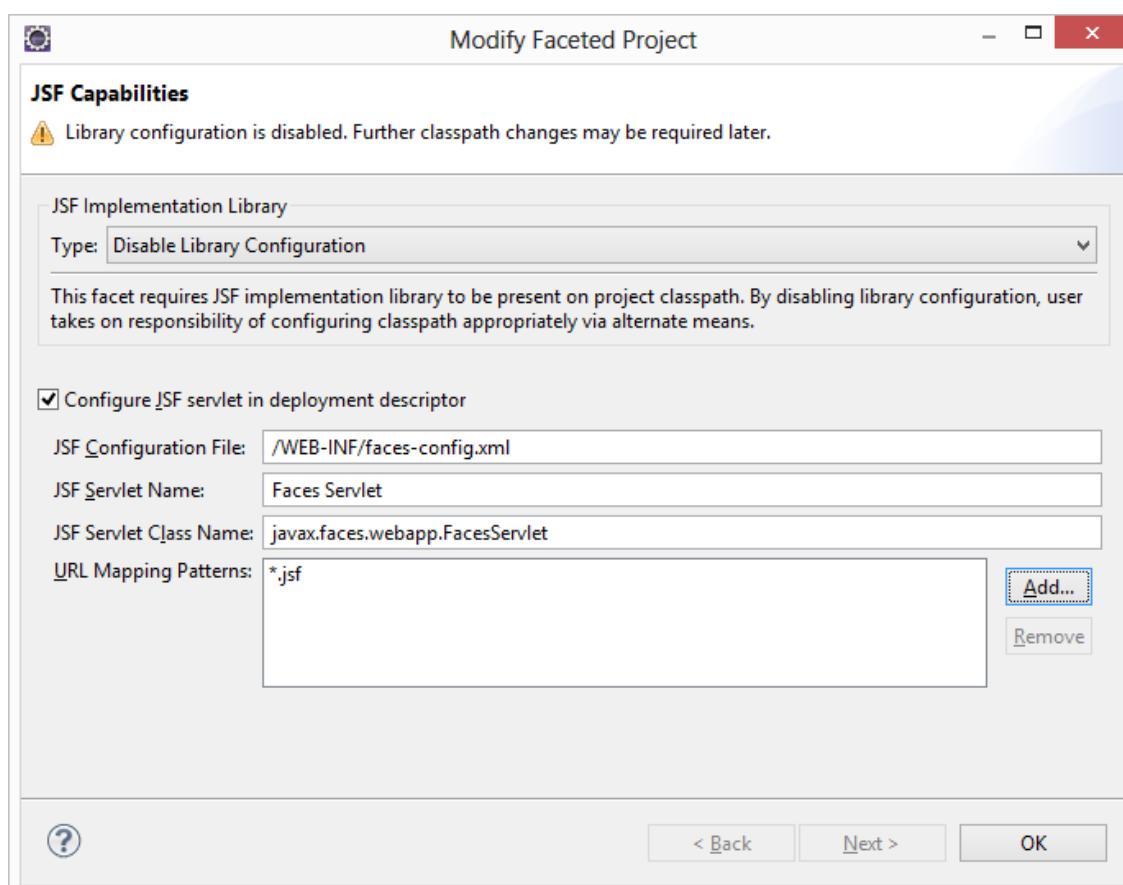
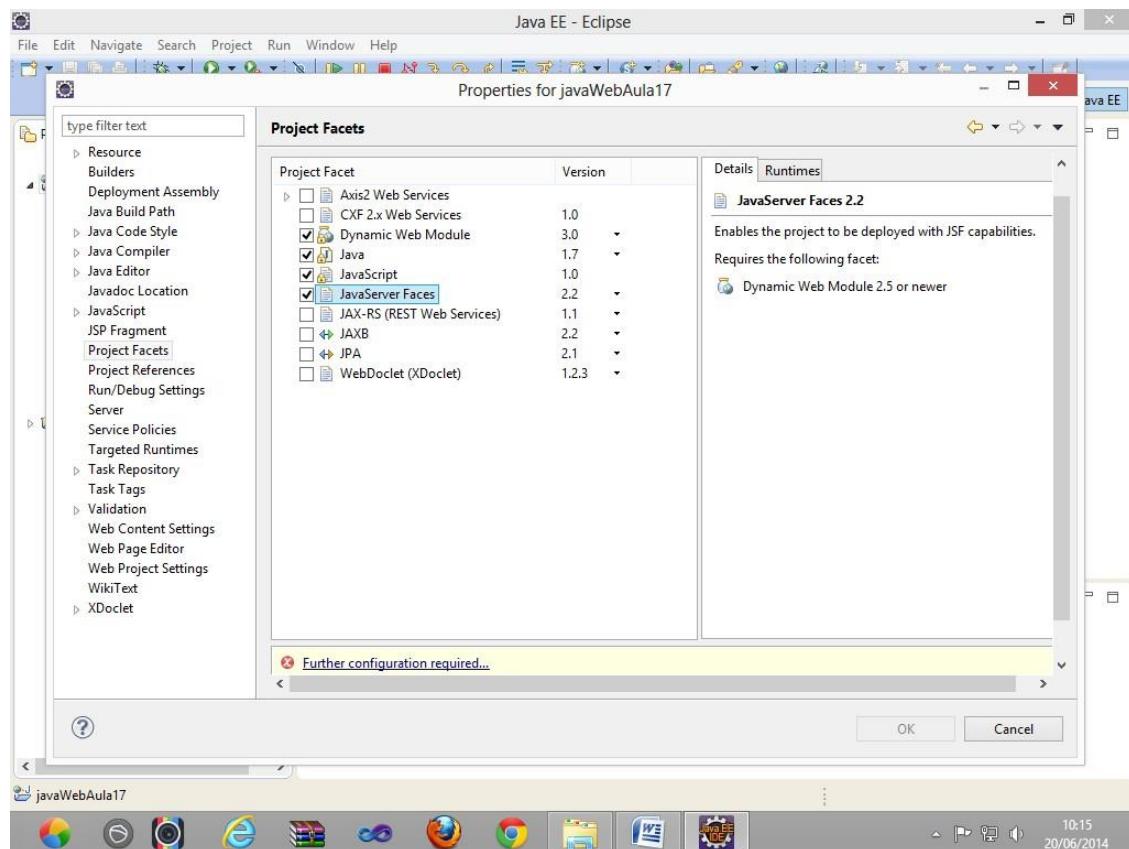


# Java WebDeveloper - BRQ

Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula  
**27**





## Bibliotecas para o Jsf e Primefaces

Jsf - libs			
Área de Transferência			
Organizar			
<input type="checkbox"/> Arquivo	<input type="checkbox"/> Início	<input type="checkbox"/> Compartilhar	<input type="checkbox"/> Exibir
<input type="checkbox"/> Copiar	<input type="checkbox"/> Colar	<input type="checkbox"/> Recortar	<input type="checkbox"/> Copiar caminho
<input type="checkbox"/> Colar atalho	<input type="checkbox"/> Mover para	<input type="checkbox"/> Copiar para	<input type="checkbox"/> Excluir
<input type="checkbox"/> Organizar	<input type="checkbox"/> Renomear	<input type="checkbox"/> Fácil acesso	<input type="checkbox"/> Nova pasta
	<input type="checkbox"/> Novo	<input type="checkbox"/> Propriedades	<input type="checkbox"/> Abrir
		<input type="checkbox"/> Editar	<input type="checkbox"/> Histórico
		<input type="checkbox"/> Selecionar tudo	<input type="checkbox"/> Selecionar nenhum
		<input type="checkbox"/> Inverter seleção	<input type="checkbox"/> Selecionar
Pesquisar Jsf - libs			
Favoritos			
Área de Trabalho			
Downloads			
Locais recentes			
Bibliotecas			
Documentos			
Imagens			
Músicas			
Vídeos			
Grupo doméstico			
Computador			
12 itens			

Primefaces			
Área de Transferência			
Organizar			
<input type="checkbox"/> Arquivo	<input type="checkbox"/> Início	<input type="checkbox"/> Compartilhar	<input type="checkbox"/> Exibir
<input type="checkbox"/> Copiar	<input type="checkbox"/> Colar	<input type="checkbox"/> Recortar	<input type="checkbox"/> Copiar caminho
<input type="checkbox"/> Colar atalho	<input type="checkbox"/> Mover para	<input type="checkbox"/> Copiar para	<input type="checkbox"/> Excluir
<input type="checkbox"/> Organizar	<input type="checkbox"/> Renomear	<input type="checkbox"/> Fácil acesso	<input type="checkbox"/> Nova pasta
	<input type="checkbox"/> Novo	<input type="checkbox"/> Propriedades	<input type="checkbox"/> Abrir
		<input type="checkbox"/> Editar	<input type="checkbox"/> Histórico
		<input type="checkbox"/> Selecionar tudo	<input type="checkbox"/> Selecionar nenhum
		<input type="checkbox"/> Inverter seleção	<input type="checkbox"/> Selecionar
Pesquisar Primefaces			
Favoritos			
Área de Trabalho			
Downloads			
Locais recentes			
2 itens			

WebContent
META-INF
WEB-INF
lib
commons-beanutils-1.8.0.jar
commons-beanutils.jar
commons-collections-3.2.1.jar
commons-collections.jar
commons-digester-2.0.jar
commons-digester.jar
commons-logging-1.1.1.jar
commons-logging.jar
jsf-api.jar
jsf-impl.jar
jstl.jar
primefaces-5.0-sources.jar
primefaces-5.0.jar
standard.jar
faces-config.xml
web.xml

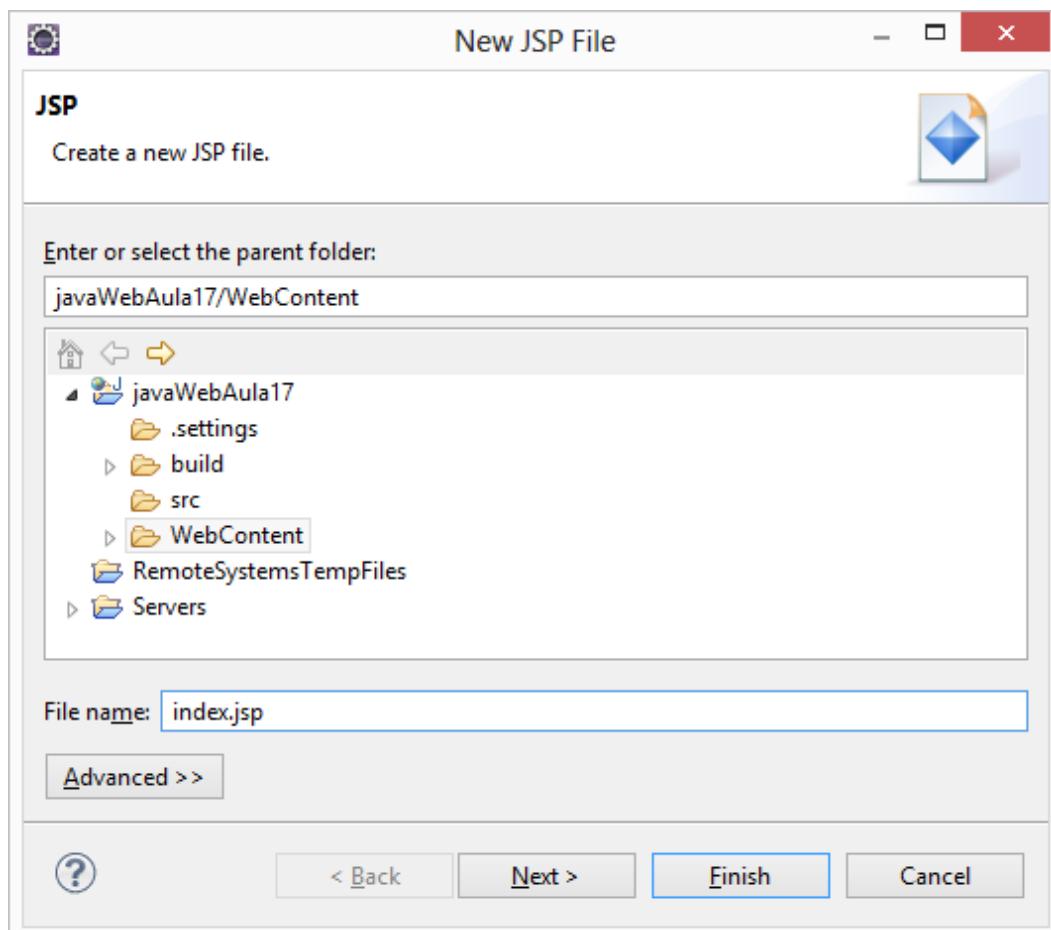
## Caso de Uso do Projeto



## Facelets

Recurso do Java Server Faces utilizado para montagem de layouts em páginas JSF (**Templates**)

- Criando a página inicial...





# Java WebDeveloper - BRQ

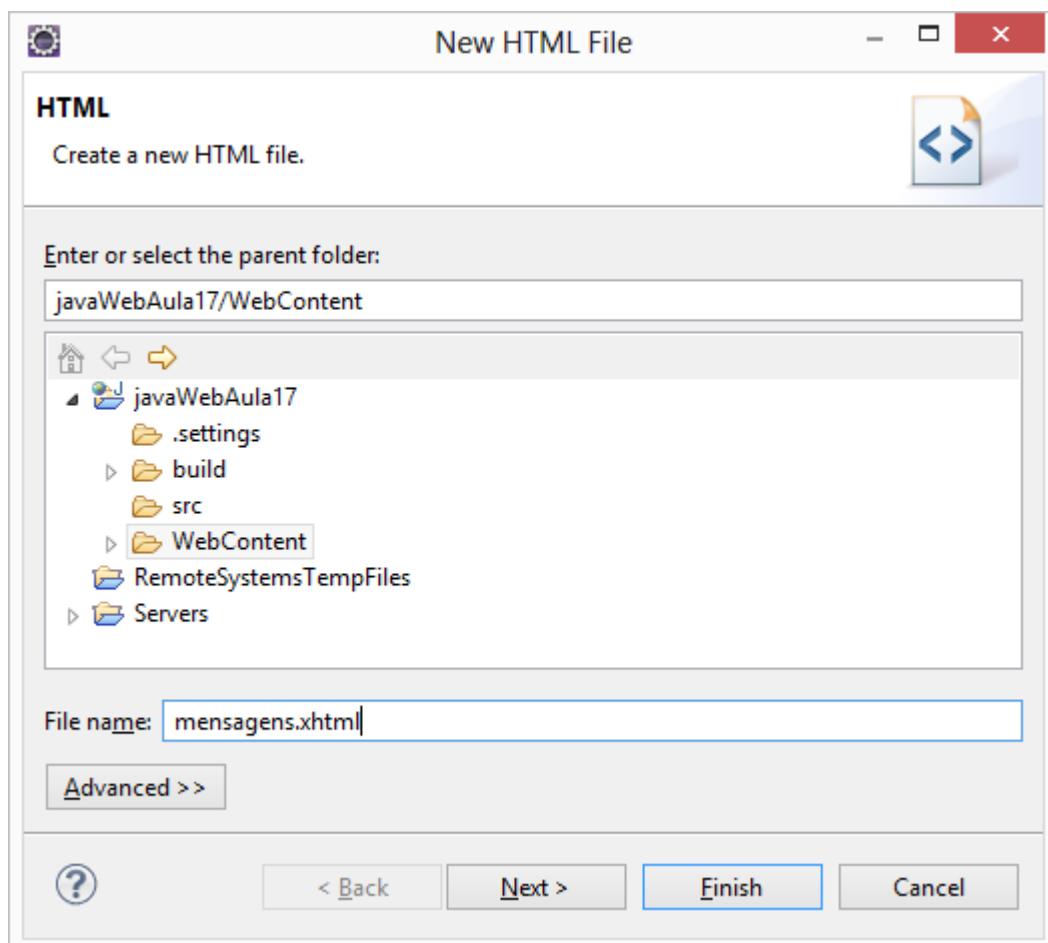
Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula

27

```
<!-- Redirecionamento do JSP -->
<jsp:forward page="mensagens.jsf"/>
```



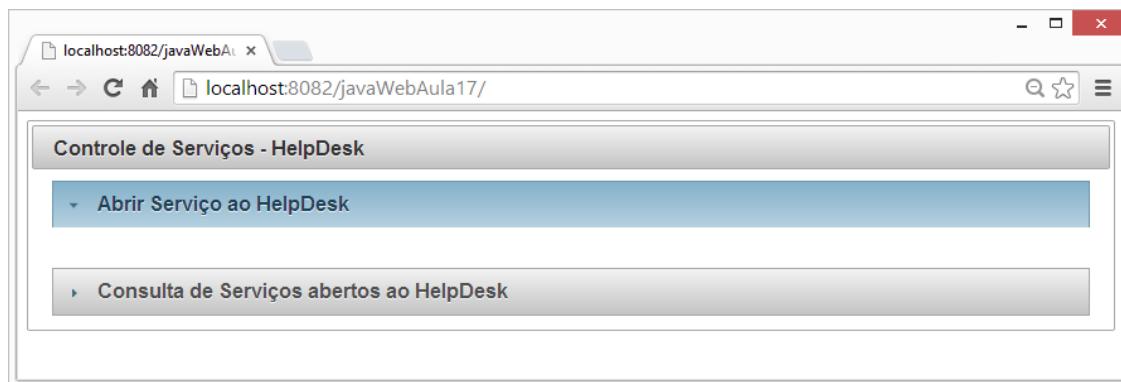
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">

<h:head>
</h:head>
<h:body>

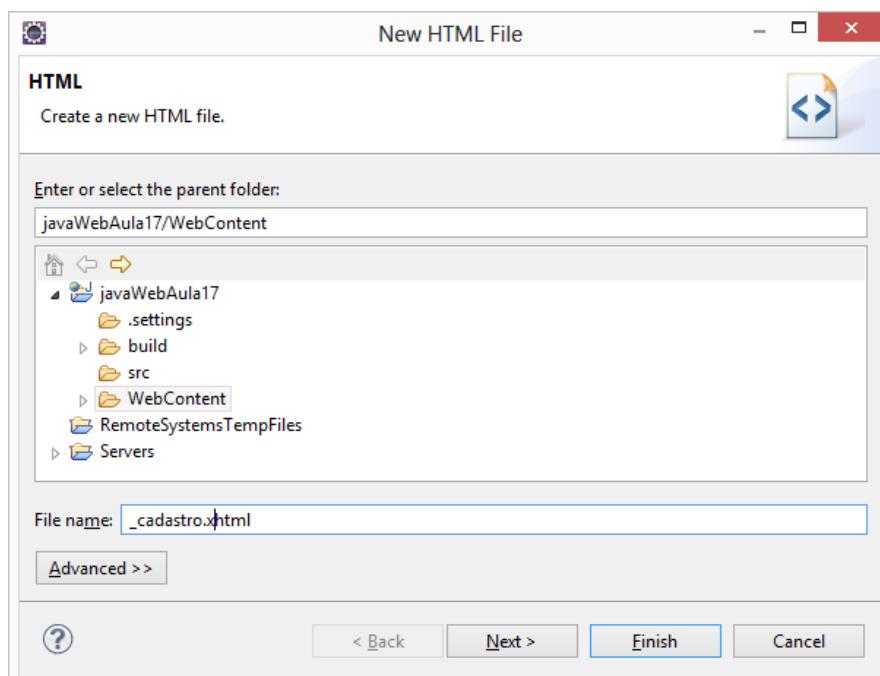
    <p:panel header="Controle de Serviços - HelpDesk">
        <p:accordionPanel>
```



```
<p:tab title="Abrir Serviço ao HelpDesk">  
</p:tab>  
  
<p:tab title="Consulta de Serviços abertos  
ao HelpDesk">  
</p:tab>  
  
</p:accordionPanel>  
  
</p:panel>  
  
</h:body>  
  
</html>
```



## Criando as subpáginas de cadastro e consulta de serviços





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

    <h:form id="formcadastro">

        <h:outputText value="Para cadastrar um novo chamado
            ao helpdesk, preencha os campos abaixo:"/>

    </h:form>

</ui:fragment>
```

---

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.org/ui">

    <h:head>
    </h:head>
    <h:body>

        <p:panel header="Controle de Serviços - HelpDesk">

            <p:accordionPanel>

                <p:tab title="Abrir Serviço ao HelpDesk">
                    <ui:include src="_cadastro.xhtml"/>
                </p:tab>

                <p:tab title="Consulta de Serviços
                    abertos ao HelpDesk">
                </p:tab>

            </p:accordionPanel>

        </p:panel>
    </h:body>
</html>
```



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

    <h:form id="formcadastro">

        <h:outputText value="Para cadastrar um novo chamado
            ao helpdesk, preencha os campos abaixo:"/>

        <p:separator/>

        <h:panelGrid columns="2">

            <h:outputText value="Nome do Solicitante: "/>
            <p:inputText size="80"/>

            <h:outputText value="Tipo do Serviço: "/>

            <p:selectOneMenu>
                <f:selectItem itemValue=""
                    itemLabel="- Escolha uma Opção -"
                    noSelectionOption="true"/>
                <f:selectItem itemValue="Manutenção"
                    itemLabel="Manutenção de Equipamentos"/>
                <f:selectItem itemValue="Software"
                    itemLabel="Instalação ou Reparo de Software"/>
                <f:selectItem itemValue="Sistema"
                    itemLabel="Problemas de utilização
                        dos Sistemas"/>
            </p:selectOneMenu>

            <h:outputText value="Data de abertura
                do chamado: "/>
            <p:calendar pattern="dd/MM/yyyy HH:mm" />

            <h:outputText value="Severidade: "/>

            <p:selectOneRadio layout="pageDirection">
                <f:selectItem itemValue="Baixa"
                    itemLabel="Severidade Baixa"/>
                <f:selectItem itemValue="Media"
                    itemLabel="Severidade Media"/>
                <f:selectItem itemValue="Alta"
                    itemLabel="Severidade Alta"/>
            </p:selectOneRadio>

        </h:panelGrid>

        <p:editor/>
```



# Java WebDeveloper - BRQ

Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula  
**27**

```
<p>
    <p:commandButton value="Cadastrar Serviço"
                      icon="ui-icon-disk"/>
</p>

<p:message for="formcadastro"/>
<p:growl/>

</h:form>

</ui:fragment>
```

## Resultado...

The screenshot shows a web browser window with the URL `localhost:8082/javaWebA17/`. The page title is "Controle de Serviços - HelpDesk". A navigation bar at the top has items like "Abrir Serviço ao HelpDesk" and "Consulta de Serviços abertos ao HelpDesk". The main content area contains a form for creating a new service ticket. It includes fields for "Nome do Solicitante" (with a dropdown menu), "Tipo do Serviço" (a dropdown menu showing "- Escolha uma Opção -"), "Data de abertura do chamado" (a date input field), and "Severidade" (radio buttons for "Severidade Baixa", "Severidade Media", and "Severidade Alta"). Below these fields is a rich text editor toolbar. At the bottom of the form is a button labeled "Cadastrar Serviço".

## \_consulta.xhtml

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

    <h:form id="formconsulta">
```



```
<p:dataGrid paginator="true" rows="10" emptyMessage="Não
há serviços abertos para o HelpDesk">

    <f:facet name="header">
        <h:outputText value="Relação de
                        Serviços Abertos"/>
    </f:facet>

</p:dataGrid>

</h:form>

</ui:fragment>
```

-----

## **servicos.xhtml**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!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"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">

<h:head>
</h:head>
<h:body>

    <p:panel header="Controle de Serviços - HelpDesk">

        <p:accordionPanel>

            <p:tab title="Abrir Serviço ao HelpDesk">
                <ui:include src="_cadastro.xhtml"/>
            </p:tab>

            <p:tab title="Consulta de Serviços abertos
                            ao HelpDesk">
                <ui:include src="_consulta.xhtml"/>
            </p:tab>

        </p:accordionPanel>

    </p:panel>

</h:body>

</html>
```



# Java WebDeveloper - BRQ

Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula

27

The screenshot shows a Java web application interface titled "Controle de Serviços - HelpDesk". A sidebar on the left contains two items: "Abrir Serviço ao HelpDesk" (selected) and "Consulta de Serviços abertos ao HelpDesk". The main content area displays a form for creating a new service request. It includes fields for "Nome do Solicitante" (Name of Requester), "Tipo do Serviço" (Service Type, dropdown menu), "Data de abertura do chamado" (Ticket opening date, date picker), and "Severidade" (Severity, radio buttons for Low, Medium, and High). Below these fields is a rich text editor toolbar. At the bottom of the form is a "Cadastrar Serviço" (Create Service) button. The browser's address bar shows "localhost:8082/javaWebAula17/".

The screenshot shows a Java web application interface titled "Controle de Serviços - HelpDesk". The sidebar on the left shows "Abrir Serviço ao HelpDesk" and "Consulta de Serviços abertos ao HelpDesk" (selected). The main content area displays a table titled "Relação de Serviços Abertos" (List of Open Services). A message at the top of the table says "Não há serviços abertos para o HelpDesk" (No open services for the HelpDesk). The browser's address bar shows "localhost:8082/javaWebAula17/".





# Java WebDeveloper - BRQ

Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula  
**27**

## Incluindo template do primefaces...

<http://www.primefaces.org/themes>

The screenshot shows a browser window displaying the PrimeFaces Themes page. At the top, there's a navigation bar with links for Overview, Demos, Development, Support, and Social. Below the navigation, a section titled "Themes" is shown. A note states: "PrimeFaces is integrated with the ThemeRoller CSS Framework. You can either choose from the 38 pre-designed free themes, purchase the ELITE themes or create your own using easily the online theme generator tool of ThemeRoller." A section for "ELITE Themes" follows, with a note that they are available to ELITE & PRO users exclusively. Below this, several theme examples are displayed, including "MetroUI" which is highlighted. Each example shows a screenshot of a web application interface with a specific theme applied. At the bottom of the screenshot, the Windows taskbar is visible with various icons for applications like Google Chrome, File Explorer, and others.

<http://repository.primefaces.org/org/primefaces/themes/delta/1.0.10/>

The screenshot shows a browser window displaying the contents of the "delta/1.0.10" directory on the PrimeFaces repository. The title bar says "Index of /org/primefaces/themes/delta/1.0.10". The page has a header with columns for Name, Last modified, Size, and Description. Below the header is a table listing files:

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">delta-1.0.10.jar</a>	21-Apr-2013 23:05	53K	
<a href="#">delta-1.0.10.jar.md5</a>	21-Apr-2013 23:05	32	
<a href="#">delta-1.0.10.jar.sha1</a>	21-Apr-2013 23:05	40	
<a href="#">delta-1.0.10.pom</a>	21-Apr-2013 23:05	619	
<a href="#">delta-1.0.10.pom.md5</a>	21-Apr-2013 23:05	32	
<a href="#">delta-1.0.10.pom.sha1</a>	21-Apr-2013 23:05	40	

At the bottom of the page, a footer note reads: "Apache/2.2.17 (Ubuntu) Server at repository.primefaces.org Port 80"



# Java WebDeveloper - BRQ

Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula  
**27**

## /WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">
    <display-name>javaWebAula17</display-name>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>
    <context-param>
        <param-name>primefaces.THEME</param-name>
        <param-value>delta</param-value>
    </context-param>
</web-app>
```



## JavaBean - Servico

```
package entity;

import java.util.Date;

public class Servico {

    private Integer idServico;
    private String solicitante;
    private String tipo;
    private Date dataServico;
    private String severidade;
    private String descricao;

    public Servico() {
    }

    public Servico(Integer idServico, String solicitante,
                   String tipo, Date dataServico, String severidade,
                   String descricao) {
        super();
        this.idServico = idServico;
        this.solicitante = solicitante;
        this.tipo = tipo;
        this.dataServico = dataServico;
        this.severidade = severidade;
        this.descricao = descricao;
    }

    @Override
    public String toString() {
        return "Servico [idServico=" + idServico + ",\n" +
               "solicitante=" + solicitante + ", tipo=" + tipo + ",\n" +
               "dataServico=" + dataServico + ", severidade=" +\n" +
               "severidade + ", descricao=" + descricao + "]";
    }

    public Integer getIdServico() {
        return idServico;
    }

    public void setIdServico(Integer idServico) {
        this.idServico = idServico;
    }

    public String getSolicitante() {
        return solicitante;
    }
```



```
public void setSolicitante(String solicitante) {
    this.solicitante = solicitante;
}

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public Date getDataServico() {
    return dataServico;
}

public void setDataServico(Date dataServico) {
    this.dataServico = dataServico;
}

public String getSeveridade() {
    return severidade;
}

public void setSeveridade(String severidade) {
    this.severidade = severidade;
}

public String getDescricao() {
    return descricao;
}

public void setDescricao(String descricao) {
    this.descricao = descricao;
}

-----  
  
drop database if exists aulaweb17;
create database aulaweb17;
use aulaweb17;

create table servico(
    idservico          integer      auto_increment,
    solicitante        varchar(50)   not null,
    tipo               varchar(50)   not null,
    dataservico        timestamp    not null,
    severidade         varchar(50)   not null,
    descricao          text         not null,
    primary key(idservico));
```



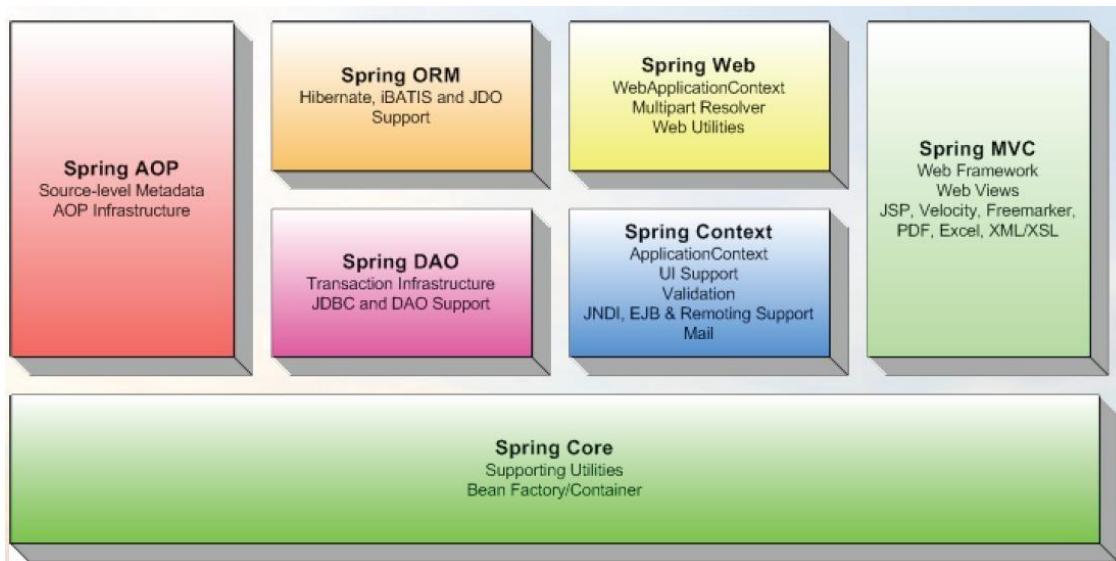
## Spring Framework

Framework Java voltado para acrescentar 2 características novas ao projeto:

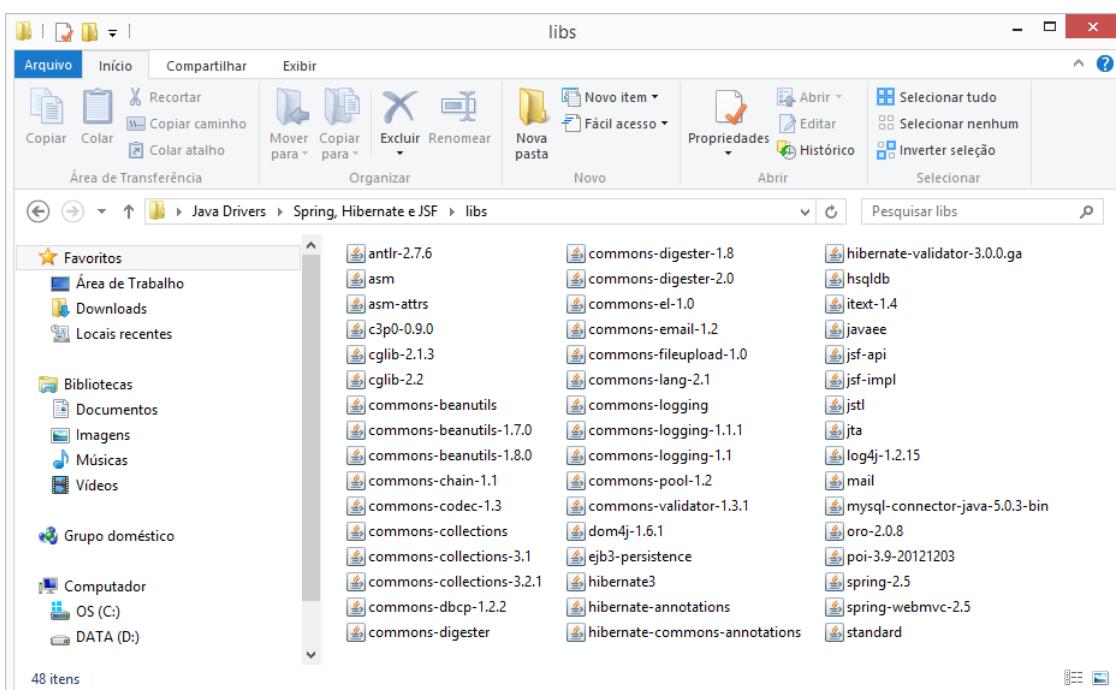
- **DI Dependency Injection**  
Injeção de dependência

- ## ?

# **IoC Inverse of Control**



### **Incluir no projeto as bibliotecas necessárias**





## \WEB-INF

### ② **applicationContext.xml**

Arquivo de configuração para o framework Spring

### ② **faces-config.xml**

Arquivo de configuração padrão do JSF

### ② **web.xml**

Arquivo de configuração padrão do projeto Java Web

## WEB-INF/applicationContext.xml

Arquivo para configuração do framework Spring

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">

    <bean id="ds" class="org.apache.commons.dbcp.BasicDataSource"
          destroy-method="close">

        <property name="driverClassName"
                  value="com.mysql.jdbc.Driver" />
        <property name="url"
                  value="jdbc:mysql://localhost:3306/" />
        <property name="username" value="root" />
        <property name="password" value="" />
    </bean>
    <bean id="sf"
          class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">

        <property name="dataSource" ref="ds" />

        <property name="annotatedClasses">
            <list>
                <value></value>
            </list>
        </property>
    
```



```
<property name="hibernateProperties">
    <props>
        <prop key="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect</prop>
        <prop key="hibernate.show_sql">true</prop>
        <prop key="hibernate.format_sql">true</prop>
    </props>
</property>
</bean>
</beans>
```

## WEB-INF/web.xml

Adicionar no web.xml um mapeamento para fazer com que o projeto Java Web reconheça o framework Spring

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    version="3.0">
    <display-name>javaWebAula17</display-name>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/applicationContext.xml
        </param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.
            context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>
            javax.faces.webapp.FacesServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>
```



```
<context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>excite-bike</param-value>
</context-param>

</web-app>
```

-----

## WEB-INF/faces-config.xml

Adicionar no arquivo de configuração do JSF um mapeamento para o Java Server Faces reconheça o framework Spring.

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
    version="2.2">

    <application>
        <el-resolver>
            org.springframework.web.jsf.el.
            SpringBeanFacesELResolver
        </el-resolver>
    </application>

</faces-config>
```

-----

## IoC Inverse of Control

Inversão de Controle

O Spring é um framework que 'absorve' a maioria das tecnologias Java Web inclusive seus frameworks. Por exemplo: O Spring possui internamente um módulo para funcionamento do Hibernate que simplifica o uso deste framework.

```
package entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```



```
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "servico")
public class Servico {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idservico")
    private Integer idServico;

    @Column(name = "solicitante", length = 50, nullable = false)
    private String solicitante;

    @Column(name = "tipo", length = 50, nullable = false)
    private String tipo;

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "dataservico", nullable = false)
    private Date dataServico;

    @Column(name = "severidade", length = 50, nullable = false)
    private String severidade;

    @Column(name = "descricao", length = 500, nullable = false)
    private String descricao;

    public Servico() {
    }

    public Servico(Integer idServico, String solicitante,
                   String tipo, Date dataServico, String severidade,
                   String descricao) {
        super();
        this.idServico = idServico;
        this.solicitante = solicitante;
        this.tipo = tipo;
        this.dataServico = dataServico;
        this.severidade = severidade;
        this.descricao = descricao;
    }

    @Override
    public String toString() {
        return "Servico [idServico=" + idServico + ", "
               + solicitante + ", tipo=" + tipo + ", "
               + dataServico + ", severidade=" +
               severidade + ", descricao=" + descricao + "]";
    }
}
```



```
public Integer getIdServico() {
    return idServiço;
}

public void setIdServico(Integer idServiço) {
    this.idServiço = idServiço;
}

public String getSolicitante() {
    return solicitante;
}

public void setSolicitante(String solicitante) {
    this.solicitante = solicitante;
}

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public Date getDataServiço() {
    return dataServiço;
}

public void setDataServiço(Date dataServiço) {
    this.dataServiço = dataServiço;
}

public String getSeveridade() {
    return severidade;
}

public void setSeveridade(String severidade) {
    this.severidade = severidade;
}

public String getDescrição() {
    return descrição;
}

public void setDescrição(String descrição) {
    this.descrição = descrição;
}

}
```



## Configurar o Hibernate através do Spring WEB-INF/applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">

    <!-- Mapeamento da conexão com o banco de dados (DataSource) -->
    <bean id="conexao"
        class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">

        <property name="driverClassName"
            value="com.mysql.jdbc.Driver" />
        <property name="url"
            value="jdbc:mysql://localhost:3306/aulaweb17" />
        <property name="username" value="root" />
        <property name="password" value="" />

    </bean>

    <!-- Mapeamento do framework Hibernate
        (embutido dentro do Spring) -->
    <bean id="hibernate"
        class="org.springframework.orm.hibernate3.annotation.
        AnnotationSessionFactoryBean">

        <!-- Definir qual conexão com banco será
            utilizada pelo Hibernate do Spring -->
        <property name="dataSource" ref="conexao" />

        <property name="annotatedClasses">
            <list>
                <value>entity.Servico</value>
            </list>
        </property>

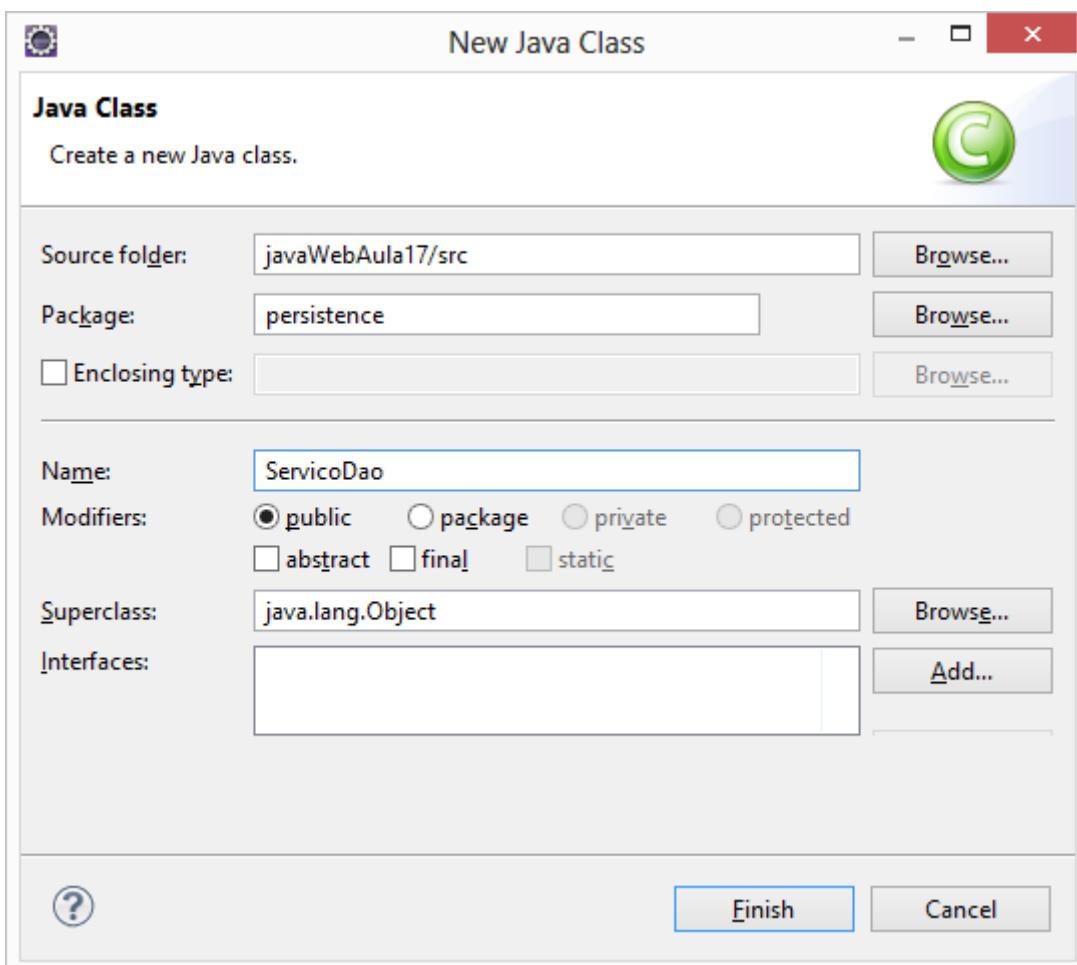
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">
                    org.hibernate.dialect.MySQLDialect
                </prop>
            </props>
        </property>
    
```



```
<prop key="hibernate.show_sql">true</prop>
<prop key="hibernate.format_sql">true</prop>
</props>
</property>
</bean>

</beans>
```

## Classe de persistência...



```
package persistence;

import java.util.List;
import org.springframework.orm.hibernate3.HibernateTemplate;
import entity.Servico;

public class ServicoDao {
```

```
    private HibernateTemplate hibernate; //null
```



```
public void create(Servico s) throws Exception{
    hibernate.persist(s);
}

public void update(Servico s) throws Exception{
    hibernate.merge(s);
}

public void delete(Servico s) throws Exception{
    hibernate.delete(s);
}

public Servico findById(Integer idServico) throws Exception{
    return (Servico) hibernate.get(Servico.class, idServico);
}

@SuppressWarnings("unchecked")
public List<Servico> findAll() throws Exception{
    return hibernate.find("select s from Servico as s
                           order by s.dataServiço desc");
}
}
```

---

## Mapear no Spring a Classe ServicoDao

```
<!-- Mapeamento para a Classe ServicoDao -->
<bean id="servicodao" class="persistence.ServicoDao">
    <!-- Passar para a Classe ServicoDap as configurações
        para ativação do Hibernate -->
</bean>

package persistence;

import java.util.List;

import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateTemplate;

import entity.Servico;

public class ServicoDao {

    private HibernateTemplate hibernate; //null

    //Injeção de dependência
    //Método para receber a configuração do hibernate
    //enviada pelo Spring
```



```
//O applicationContext.xml irá enviar uma SessionFactory
//pronta para ativar o atributo 'hibernate'
public void setSessionFactory(SessionFactory
                               sessionFactory){
    //inicializando o atributo HibernateTemplate utilizando
    //a sessionFactory de entrada
    hibernate = new HibernateTemplate
                (sessionFactory);
}

public void create(Servico s) throws Exception{
    hibernate.persist(s);
}

public void update(Servico s) throws Exception{
    hibernate.merge(s);
}

public void delete(Servico s) throws Exception{
    hibernate.delete(s);
}

public Servico findById(Integer idServico) throws Exception{
    return (Servico) hibernate.get(Servico.class, idServico);
}

@SuppressWarnings("unchecked")
public List<Servico> findAll() throws Exception{
    return hibernate.find("select s from Servico as s
                           order by s.dataServiço desc");
}

}

-----
<!-- Mapeamento para a Classe ServicoDao -->
<bean id="servicodao" class="persistence.ServicoDao">

    <!-- Passar para a Classe ServicoDao as configurações para
        ativação do Hibernate -->
    <!-- name => nome do parametro do método 'setSessionFactory'
        na Classe 'ServicoDao' -->
    <!-- ref  => configuração do hibernate criada neste xml que
        está sendo passada para o sessionFactory -->

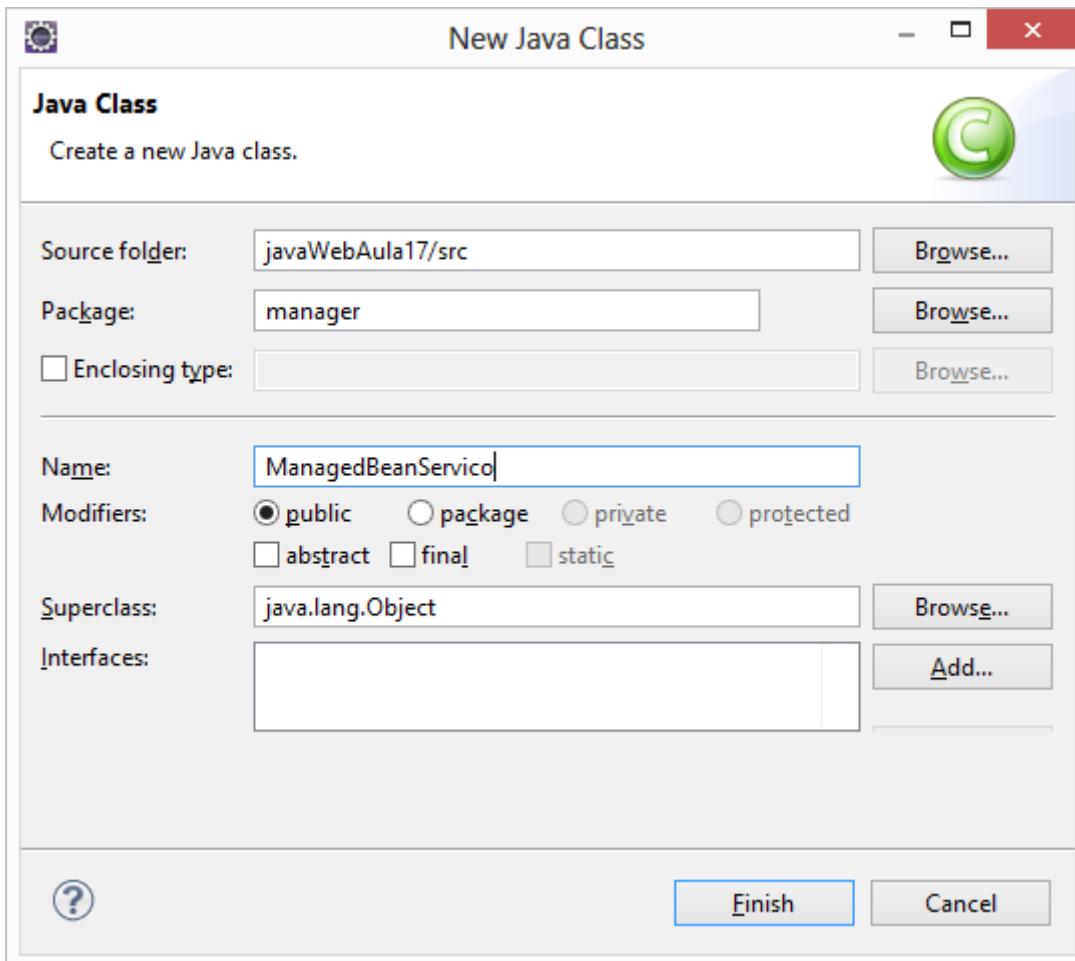
    <property name="sessionFactory" ref="hibernate"/>

</bean>
```



## ManagedBean

Classe de gerenciamento do JSF



```
package manager;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;

import entity.Servico;

@ManagedBean(name = "mbServico")
@RequestScoped
public class ManagedBeanServico {

    // Atributo para armazenar os dados do formulário de cadastro
    private Servico servico; // null

    public ManagedBeanServico() { // construtor da classe
        servico = new Servico(); // inicializando
    }
}
```



# Java WebDeveloper - BRQ

Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula

27

```
//Método para realizar o cadastro do serviço
public void cadastrar(){

}

public Servico getServico() {
    return servico;
}

public void setServico(Servico servico) {
    this.servico = servico;
}
}

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

    <h:form id="formcadastro">

        <h:outputText value="Para cadastrar um novo chamado
            ao helpdesk, preencha os campos abaixo:"/>

        <p:separator/>

        <h:panelGrid columns="2">

            <h:outputText value="Nome do Solicitante: "/>
            <p:inputText size="80"
                value="#{mbServiço.servico.solicitante}" />

            <h:outputText value="Tipo do Serviço: "/>
            <p:selectOneMenu value="#{mbServiço.servico.tipo}">
                <f:selectItem itemValue="" itemLabel="-
                    Escolha uma Opção -"
                    noSelectionOption="true"/>
                <f:selectItem itemValue="Manutenção"
                    itemLabel="Manutenção de Equipamentos"/>
                <f:selectItem itemValue="Software"
                    itemLabel="Instalação ou Reparo de Software"/>
                <f:selectItem itemValue="Sistema"
                    itemLabel="Problemas de utilização
                    dos Sistemas"/>
            </p:selectOneMenu>

            <h:outputText value="Data de abertura
                do chamado: "/>
        </h:panelGrid>
    </h:form>
</ui:fragment>
```



```
<p:calendar pattern="dd/MM/yyyy HH:mm"
value="#{mbServico.servico.dataServico}">

<h:outputText value="Severidade: "/>
<p:selectOneRadio layout="pageDirection"
value="#{mbServico.servico.severidade}">
    <f:selectItem itemValue="Baixa"
itemLabel="Severidade Baixa"/>
    <f:selectItem itemValue="Media"
itemLabel="Severidade Media"/>
    <f:selectItem itemValue="Alta"
itemLabel="Severidade Alta"/>
</p:selectOneRadio>

</h:panelGrid>

<p:editor value="#{mbServico.servico.descricao}">

<p>
    <p:commandButton value="Cadastrar Serviço"
icon="ui-icon-disk"
ajax="false" action="#{mbServico.cadastrar}" />
</p>

<p:message for="formcadastro"/>
<p:growl/>

</h:form>

</ui:fragment>
```

---

## Implementando o cadastro de Servicos

Injeção de dependência (DI)

```
<!-- Mapeamento para a Classe ServicoDao -->
<bean id="servicodao_spring"
class="persistence.ServicoDao">

<!-- Passar para a Classe ServicoDap as configurações para ativação do
Hibernate name => nome do parametro do método 'setSessionFactory' na
Classe 'ServicoDao' ref => configuração do hibernate criada neste xml
que está sendo passada para o sessionFactory -->

<property name="sessionFactory" ref="hibernate"/>

</bean>
```



```
package manager;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;

import com.sun.faces.config.FacesConfigInfo;

import persistence.ServicoDao;
import entity.Servico;

@ManagedBean(name = "mbServico")
@RequestScoped
public class ManagedBeanServico {

    // Atributo para armazenar os dados do formulário de cadastro
    private Servico servico; // null

    // Atributo para a classe de persistencia
    @ManagedProperty(value="#{servicodao_spring}")
    private ServicoDao servicoDao; // null

    public ManagedBeanServico() { // construtor da classe
        servico = new Servico(); // inicializando
    }

    // Método para realizar o cadastro do serviço
    public void cadastrar() {

        String mensagem = null;

        try{

            //gravando o serviço
            servicoDao.create(servico);
            servico = new Servico(); //novo espaço de memória
            mensagem = "Serviço cadastrado com sucesso.";
        }
        catch(Exception e){
            mensagem = e.getMessage();
        }

        FacesMessage msg = new FacesMessage(mensagem);
        FacesContext.getCurrentInstance().
            addMessage("formcadastro", msg);
    }

    public Servico getServico() {
        return servico;
    }
}
```



```
public void setServico(Servico servico) {  
    this.servico = servico;  
}  
  
public ServicoDao getServicoDao() {  
    return servicoDao;  
}  
  
public void setServicoDao(ServicoDao servicoDao) {  
    this.servicoDao = servicoDao;  
}  
}
```

---

## Configurando o envio de email no Spring

WEB-INF/applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
       xmlns:p="http://www.springframework.org/schema/p"  
       xmlns:aop="http://www.springframework.org/schema/aop"  
       xmlns:tx="http://www.springframework.org/schema/tx"  
       xsi:schemaLocation="http://www.springframework.org/schema/beans  
                           http://www.springframework.org/schema/beans/spring-beans-2.0.xsd  
                           http://www.springframework.org/schema/aop  
                           http://www.springframework.org/schema/aop/spring-aop-2.5.xsd  
                           http://www.springframework.org/schema/tx  
                           http://www.springframework.org/schema/tx/spring-tx-2.5.xsd">  
  
    <!-- Mapeamento da conexão com o banco de dados (DataSource) -->  
    <bean id="conexao"  
          class="org.apache.commons.dbcp.BasicDataSource"  
          destroy-method="close">  
  
        <property name="driverClassName"  
                  value="com.mysql.jdbc.Driver" />  
        <property name="url"  
                  value="jdbc:mysql://localhost:3306/aulaweb17" />  
        <property name="username" value="root" />  
        <property name="password" value="" />  
  
    </bean>  
  
    <!-- Mapeamento do framework Hibernate  
        (embutido dentro do Spring) -->  
    <bean id="hibernate"  
          class="org.springframework.orm.hibernate3.annotation.  
                AnnotationSessionFactoryBean">
```



```
<!-- Definir qual conexão com banco será utilizada pelo
Hibernate do Spring -->
<property name="dataSource" ref="conexao" />

<property name="annotatedClasses">
    <list>
        <value>entity.Servico</value>
    </list>
</property>

<property name="hibernateProperties">
    <props>
        <prop key="hibernate.dialect"
              >org.hibernate.dialect.MySQLDialect
        </prop>
        <prop key="hibernate.show_sql">true</prop>
        <prop key="hibernate.format_sql">true</prop>
    </props>
</property>
</bean>

<!-- Mapeamento para a Classe ServicoDao -->
<bean id="servicodao_spring" class="persistence.ServicoDao">
    <!-- Passar para a Classe ServicoDap as configurações
        para ativação do Hibernate -->
    <!-- name => nome do parametro do método
        'setSessionFactory' na Classe 'ServicoDao' -->
    <!-- ref => configuração do hibernate criada neste xml
        que está sendo passada para o sessionFactory -->
    <property name="sessionFactory" ref="hibernate"/>

</bean>

<!-- Mapeamento para envio de email -->
<bean id="mailSender"
      class="org.springframework.mail.javamail.JavaMailSenderImpl" >
    <property name="username" value="cotiexemplo@gmail.com" />
    <property name="password" value="@coticoti@" />
    <property name="javaMailProperties">
        <props>
            <prop
                key="mail.smtp.user">cotiexemplo@gmail.com</prop>
            <prop key="mail.smtp.password">@coticoti@</prop>
            <prop key="mail.smtp.host">smtp.gmail.com</prop>
            <prop key="mail.smtp.port">587</prop>
            <prop key="mail.smtp.auth">true</prop>
            <prop key="mail.smtp.starttls.enable">true</prop>
        </props>
    </property>
</bean>
</beans>
```



## ManagedBean

```
package manager;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import javax.mail.internet.MimeMessage;

import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;

import persistence.ServicoDao;
import entity.Servico;

@ManagedBean(name = "mbServico")
@RequestScoped
public class ManagedBeanServico {

    // Atributo para armazenar os dados do formulário de cadastro
    private Servico servico; // null

    // Atributo para a classe de persistencia
    @ManagedProperty(value="#{servicodao_spring}")
    //injecão de dependêcia
    private ServicoDao servicoDao; // null

    //Atributo para receber a configuração de envio de email
    //mapeada pelo Spring
    @ManagedProperty(value="#{mailSender}")
    private JavaMailSender email;

    public ManagedBeanServico() { // construtor da classe
        servico = new Servico(); // inicializando
    }

    // Método para realizar o cadastro do serviço
    public void cadastrar() {

        String mensagem = null;

        try{

            //gravando o serviço
            servicoDao.create(servico);

            //Envio do email
            MimeMessage texto = email.createMimeMessage();
            MimeMessageHelper helper = new MimeMessageHelper
```



```
(texto, false, "utf-8");

        texto.setContent(servico.getDescricao(),
        "text/html"); //corpo do email
        helper.setFrom("cotiexemplo@gmail.com"); //remetente
        helper.setTo("sergio.coti@gmail.com");
        //destinatario
        helper.setSubject("Serviço: " + servico.getTipo()
        + ", Severidade: " + servico.getSeveridade());

        email.send(texto); //envio a mensagem

        servico = new Servico(); //novo espaço de memória
        mensagem = "Serviço cadastrado com sucesso.";
    }
    catch(Exception e){
        mensagem = e.getMessage();
    }

    FacesMessage msg = new FacesMessage(mensagem);
    FacesContext.getCurrentInstance().addMessage
    ("formcadastro", msg);
}

public Servico getServico() {
    return servico;
}

public void setServico(Servico servico) {
    this.servico = servico;
}

public ServicoDao getServicoDao() {
    return servicoDao;
}

public void setServicoDao(ServicoDao servicoDao) {
    this.servicoDao = servicoDao;
}

public JavaMailSender getEmail() {
    return email;
}

public void setEmail(JavaMailSender email) {
    this.email = email;
}

}
```



## Exibindo a listagem de Servicos na página

```
//Atributo para gerar os dados do dataGrid
private DataModel dados;

public DataModel getDados() {

    try{

        //carregando a lista de serviços dentro do DataModel
        dados = new ListDataModel(servicoDao.findAll());

    }
    catch(Exception e){
        e.printStackTrace();
    }

    return dados;
}

public void setDados(DataModel dados) {
    this.dados = dados;
}
```

-----

### \_consulta.xhtml

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

    <h:form id="formconsulta">

        <p:dataGrid paginator="true" rows="10"
            emptyMessage="Não há serviços abertos para o HelpDesk"
            value="#{mbServico.dados}" var="s" columns="3">

            <f:facet name="header">
                <h:outputText value="Relação de Serviços Abertos"/>
            </f:facet>

            <p:panel header="Ordem de serviço">

                <h:panelGrid columns="2">

                    <h:outputText value="Código do Serviço: "/>
                    <h:outputText value="#{s.idServico}"/>
                
```



# Java WebDeveloper - BRQ

Sexta-feira, 20 de Junho de 2014

Desenvolvimento web com Java Server Faces e Primefaces.  
Injeção de dependência com Spring Framework. Envio de Emails.

Aula  
**27**

```
<h:outputText value="Solicitante: "/>
<h:outputText value="#{s.solicitante}">

<h:outputText value="Tipo: "/>
<h:outputText value="#{s.tipo}">

<h:outputText value="Severidade: "/>
<h:outputText value="#{s.severidade}">

<h:outputText value="Data do Serviço: "/>
<h:outputText value="#{s.dataServico}">
    <f:convertDateTime
        pattern="EE dd/MM/yyyy"/>
</h:outputText>

</h:panelGrid>

</p:panel>

</p:dataGrid>

</h:form>

</ui:fragment>
```



# Java WebDeveloper - BRQ

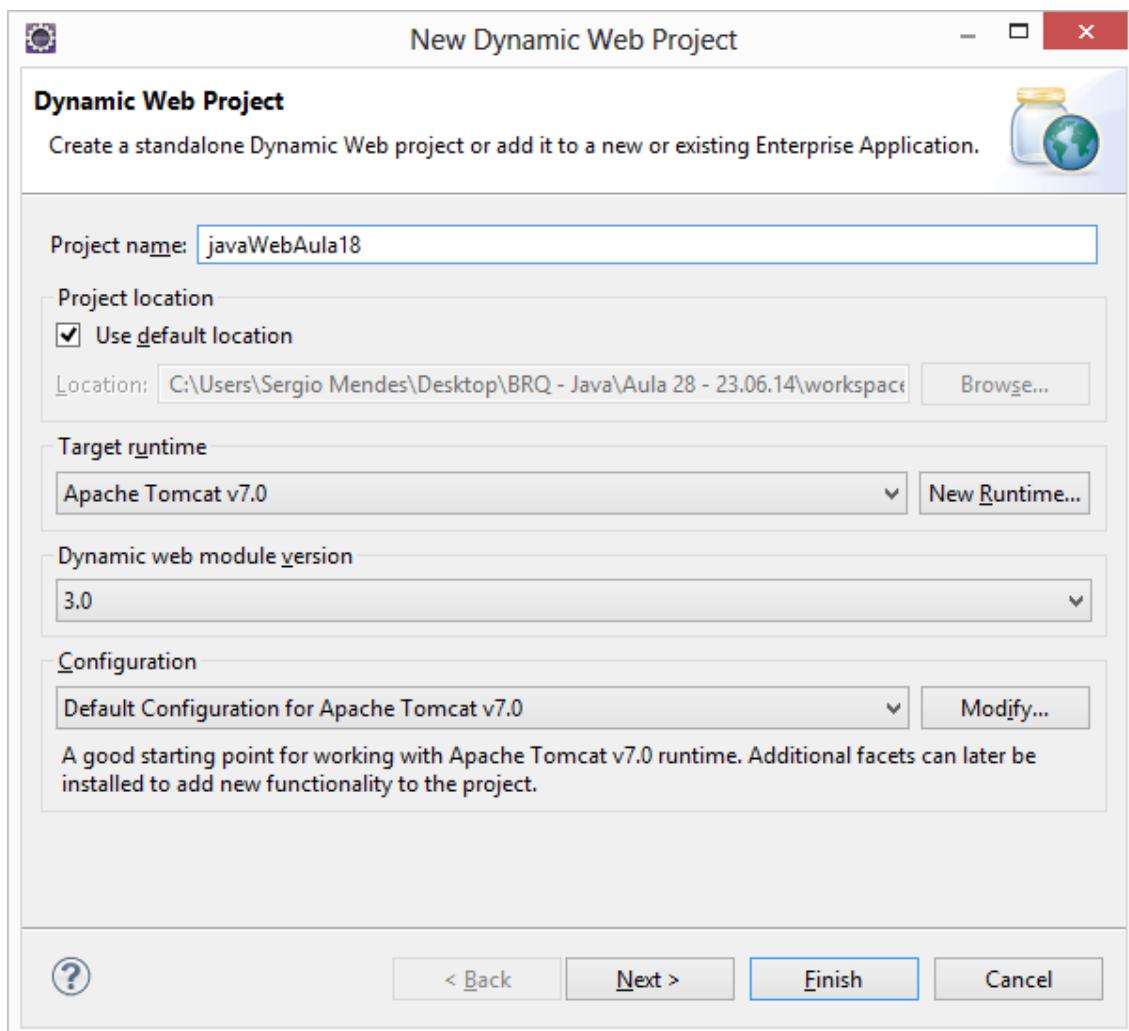
Segunda-feira, 23 de Junho de 2014

Desenvolvimento Java Web com Struts 2.  
Acesso a banco de dados com Hibernate e JPA.

Aula

28

Novo Projeto...



## Arquivo de script do banco de dados...

/sql/script.sql

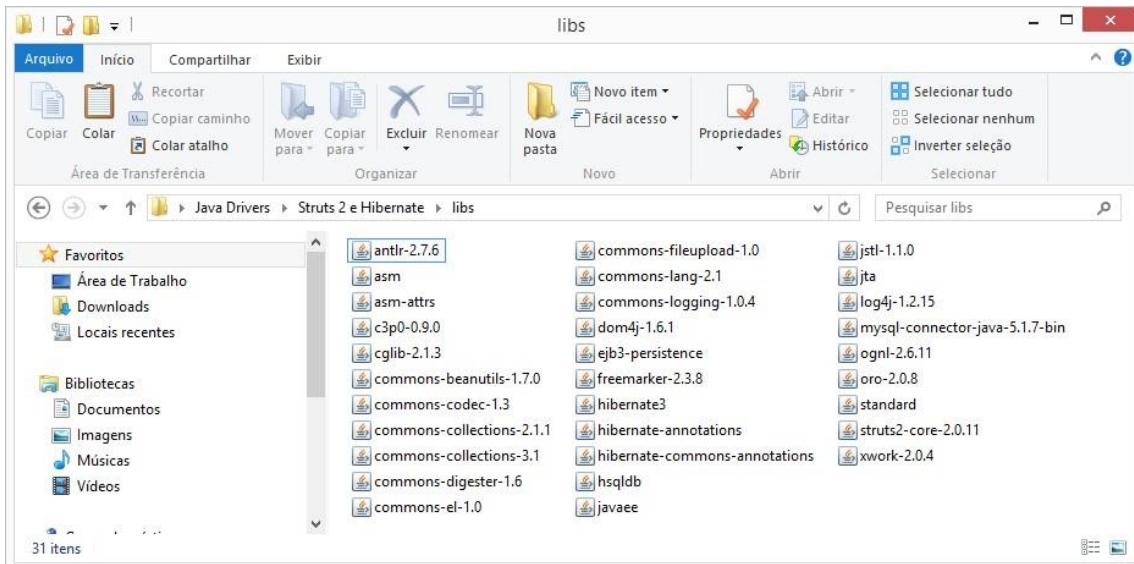
```
drop database if exists aulaweb18;
create database aulaweb18;
use aulaweb18;

create table produto(
    idproduto      integer      auto_increment,
    nome           varchar(50)   not null,
    preco          double       not null,
    quantidade     integer       not null,
    datacompra     date        not null,
    primary key(idproduto));

show tables;
desc produto;
```



Incluindo as bilbiotecas no projeto...



## Classe de entidade...

```
package entity;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
@Table(name = "produto")
public class Produto {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idproduto")
    private Integer idProduto;

    @Column(name = "nome", nullable = false, length = 50)
    private String nome;

    @Column(name = "preco", nullable = false)
    private Double preco;

    @Column(name = "quantidade", nullable = false)
    private Integer quantidade;
```



```
@Temporal(TemporalType.DATE)
@Column(name = "datacompra", nullable = false)
private Date dataCompra;

public Produto() {
}

public Produto(Integer idProduto, String nome, Double preco,
               Integer quantidade, Date dataCompra) {
    super();
    this.idProduto = idProduto;
    this.nome = nome;
    this.preco = preco;
    this.quantidade = quantidade;
    this.dataCompra = dataCompra;
}

@Override
public String toString() {
    return "Produto [idProduto=" + idProduto + ", nome="
           + nome + ", preco=" + preco + ", quantidade="
           + quantidade + ", dataCompra=" + dataCompra + "]";
}

public Integer getIdProduto() {
    return idProduto;
}

public void setIdProduto(Integer idProduto) {
    this.idProduto = idProduto;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

public Integer getQuantidade() {
    return quantidade;
}
```



```
public void setQuantidade(Integer quantidade) {
    this.quantidade = quantidade;
}

public Date getDataCompra() {
    return dataCompra;
}

public void setDataCompra(Date dataCompra) {
    this.dataCompra = dataCompra;
}
}
```

---

## conifg/mysql\_hibernate.cfg.xml

Configuração do Hibernate para acesso a uma base de dados

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>

        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect
        </property>

        <property name="hibernate.connection.driver_class">
            com.mysql.jdbc.Driver
        </property>

        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/aulaweb18
        </property>

        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"></property>

        <property name="hibernate.format_sql">true</property>
        <property name="hibernate.show_sql">true</property>

        <mapping class="entity.Produto"/>

    </session-factory>

</hibernate-configuration>
```



## HibernateUtil.java

Classe para fabricar as conexões com o banco de dados...

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static {
        try {

            sessionFactory = new AnnotationConfiguration().
                configure("config/mysql_hibernate.cfg.xml").
                buildSessionFactory();
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory
creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

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

---

## Classe de Persistência persistence/ProdutoDao.java

## Criteria

API Para consultas do Hibernate baseado não em sintaxe HQL mas sim em métodos que reproduzem consultas de banco de dados...

```
package persistence;

import hibernate.HibernateUtil;
import java.util.List;

import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.criterion.MatchMode;
```



```
import org.hibernate.criterion.Order;
import org.hibernate.criterion.Restrictions;

import entity.Produto;

public class ProdutoDao {

    private Session session;
    private Transaction transaction;
    private Criteria criteria;

    public void save(Produto p) throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.saveOrUpdate(p);
        transaction.commit();
        session.close();
    }

    public void delete(Produto p) throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();
        transaction = session.beginTransaction();
        session.delete(p);
        transaction.commit();
        session.close();
    }

    public Produto findById(Integer idProduto) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        Produto p = (Produto) session.get
            (Produto.class, idProduto);

        session.close();
        return p;
    }

    //Método para listar todos os produtos
    public List<Produto> findAll() throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        criteria = session.createCriteria(Produto.class);
        criteria.addOrder(Order.asc("nome")); //order by nome asc

        List<Produto> lista = criteria.list();
        //todos os registros
        session.close();

        return lista;
    }
}
```



```
//Método para listar os produtos pelo nome
public List<Produto> findAll(String nome) throws Exception{

    session = HibernateUtil.getSessionFactory().openSession();

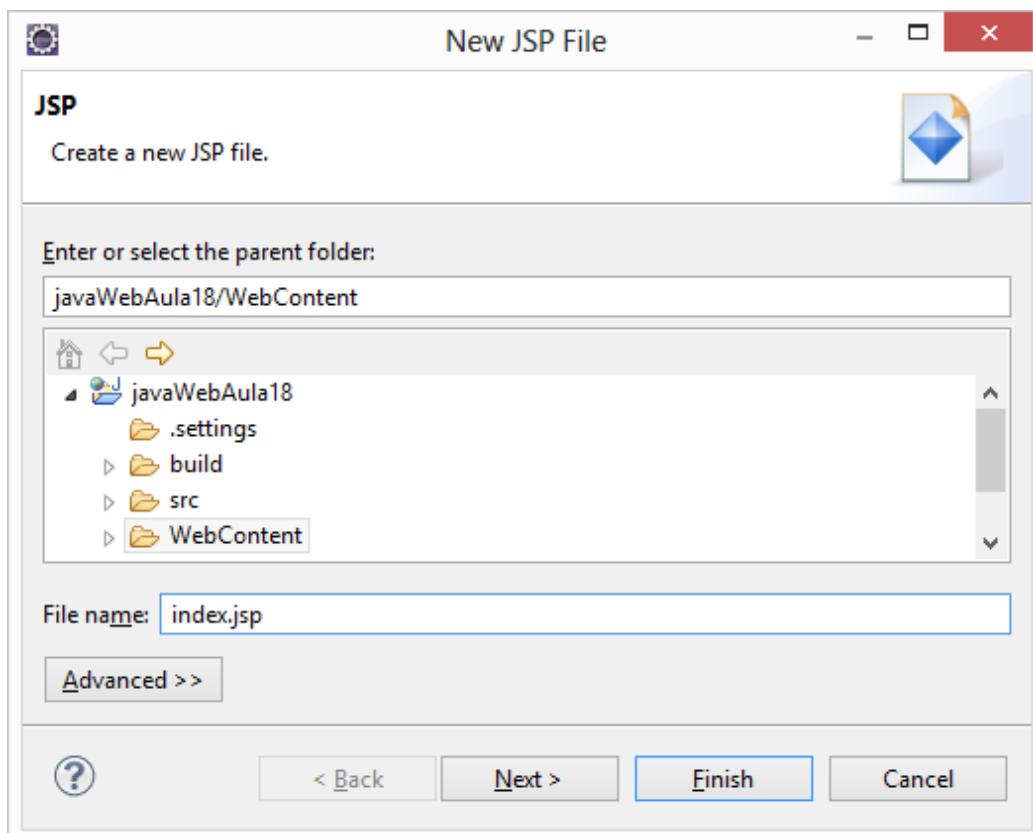
    criteria = session.createCriteria(Produto.class);
    criteria.add(Restrictions.like
        ("nome", nome, MatchMode.START));

    //MatchMode.START      -> Começando com
    //MatchMode.END        -> Terminando com
    //MatchMode.EXACT      -> Exatamente igual
    //MatchMode.ANYWHERE   -> Contendo

    List<Produto> lista = criteria.list();
    session.close();

    return lista;
}
```

## Página inicial do projeto...





```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<html>
    <head>
        </head>

    <body>
        <h3>Projeto Controle de Produtos</h3>
        <hr/>

        <ul>
            <li> <a href="cadastro.jsp">Cadastrar Produtos</a> </li>
            <li> <a href="consulta.jsp">Consultar Produtos</a> </li>
        </ul>
    </body>
</html>
```

---

## Struts 2

Utiliza dois arquivos de configuração .xml

- **WEB-INF\web.xml**

Contém a configuração para que o struts 2 seja reconhecido pelo projeto java web.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <filter>
        <filter-name>struts2</filter-name>
        <filter-class>
            org.apache.struts2.dispatcher.FilterDispatcher
        </filter-class>
    </filter>
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```



- **src\struts.xml**

Substituir o **struts-config.xml** utilizado na versão 1 do framework

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

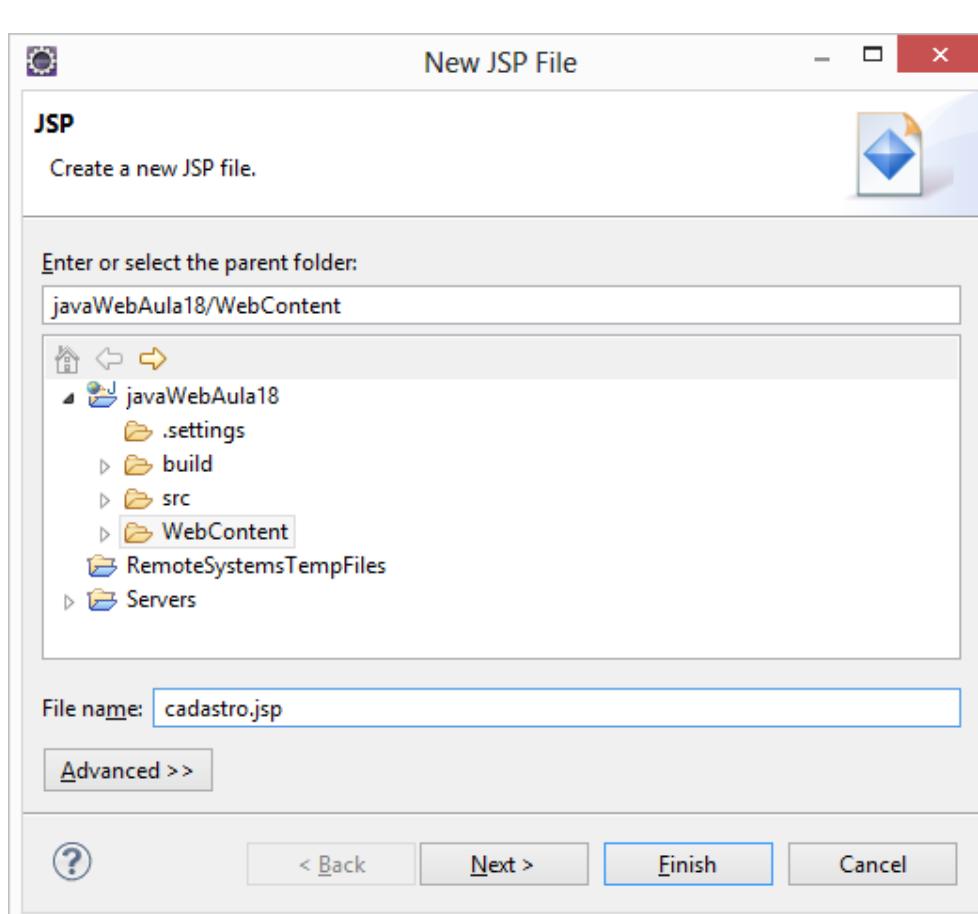
    <package name="" extends="struts-default">

        <action name="" class="" method="">
            <result name="success"></result>
        </action>

    </package>

</struts>
```

-----  
Criando a página de cadastro....





```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="/struts-tags" prefix="s" %>

<html>
    <head>
        <s:head/>
        <!-- Ativar os componentes de estilo do struts 2 -->
    </head>
    <body>
        <h3>Cadastro de Produtos</h3>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

        <s:form method="post" action="">

            <s:textfield label="Nome do Produto" name="" />
            <s:textfield label="Preço" name="" />
            <s:textfield label="Quantidade" name="" />
            <s:datetimepicker label="Data de Compra"/>

            <s:submit value="Cadastrar Produto"/>
        </s:form>
    </body>
</html>
```

localhost:8081/javaWebAula18/cadastro.jsp

**Cadastro de Produtos**

Voltar para a página inicial.

*Nome do Produto:*

*Preço:*

*Quantidade:*

*Data de Compra:*

S	T	Q	Q	S	S	D
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

2013 2014 2015



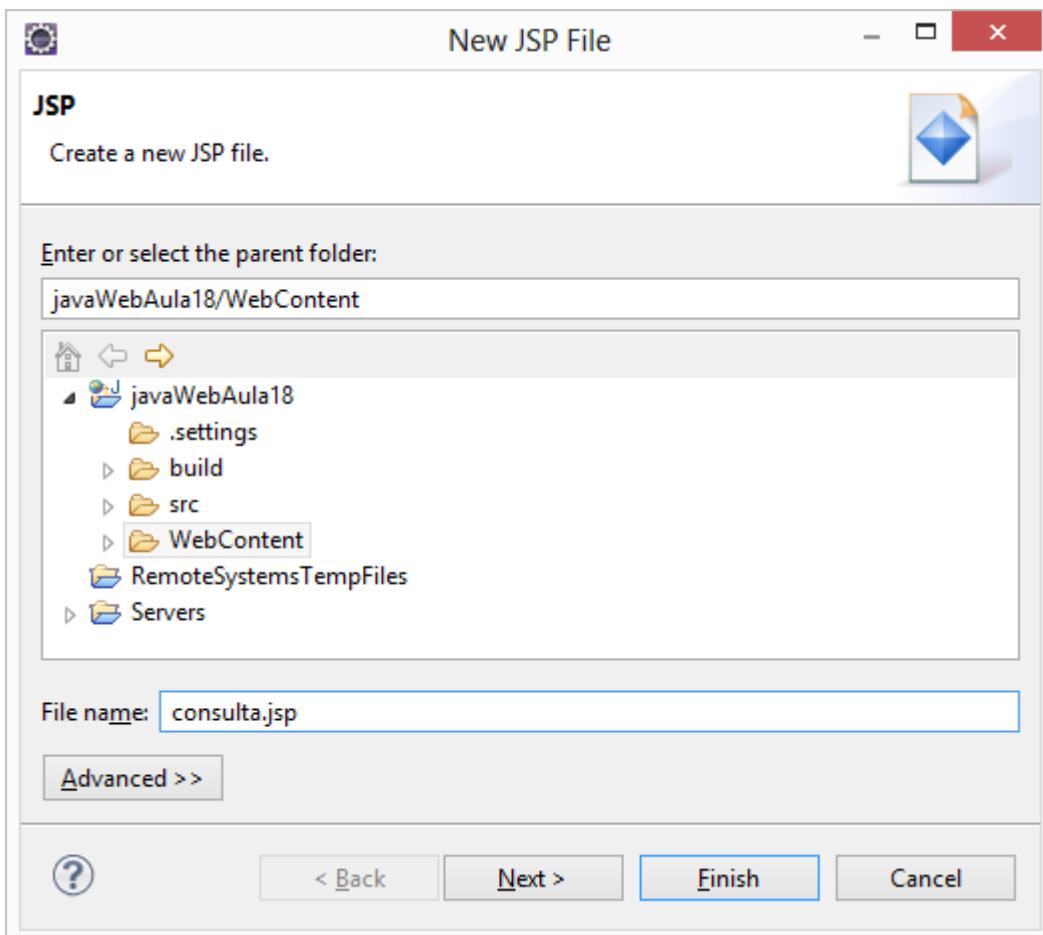


# Java WebDeveloper - BRQ

Segunda-feira, 23 de Junho de 2014

Desenvolvimento Java Web com Struts 2.  
Acesso a banco de dados com Hibernate e JPA.

Aula  
**28**



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="/struts-tags" prefix="s" %>

<html>

    <head>
        <s:head/>
        <!-- Ativar os componentes de estilo do struts 2 -->
    </head>

    <body>

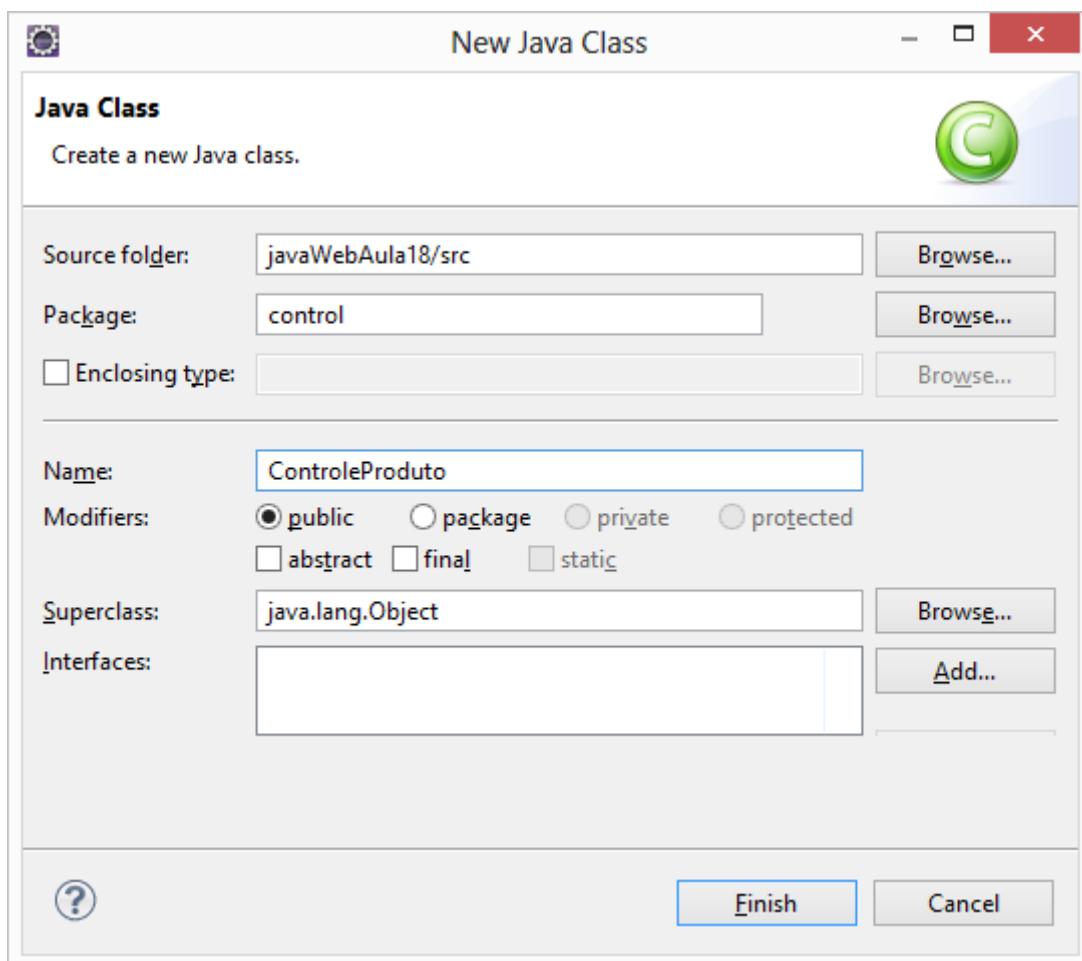
        <h3>Consulta de Produtos</h3>
        <a href="index.jsp">Voltar para a página incial
        <hr/>

    </body>

</html>
```



## Criando o Controle no Struts 2



```
package control;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

import entity.Produto;

public class ControleProduto
    extends ActionSupport
    implements ModelDriven<Produto>{}
```

### ActionSupport

Qualifica a Classe como um Controle no Struts 2

### ModelDriven

Especifica qual a classe de modelo que será utilizada para captura, entrada ou saída de dados.



```
package control;

import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

import entity.Produto;

public class ControleProduto
    extends ActionSupport
    implements ModelDriven<Produto>{

    //Atributo para capturar os dados do formulário de cadastro
    private Produto produto; //null (sem espaço de memória)

    public ControleProduto() { //construtor padrão
        produto = new Produto(); //inicializar o atributo
    }

    public Produto getProduto() {
        return produto;
    }

    public void setProduto(Produto produto) {
        this.produto = produto;
    }

    @Override
    public Produto getModel() {
        //retornar o atributo da Classe que
        //representa a modelagem de dados
        return produto;
    }
}

-----
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<%@taglib uri="/struts-tags" prefix="s" %>

<html>

    <head>
        <s:head>
        <!-- Ativar os componentes de estilo do struts 2 -->
    </head>

    <body>

        <h3>Cadastro de Produtos</h3>
```



```
<a href="index.jsp">Voltar</a> para a página inicial.  
<hr/>  
  
<s:form method="post" action="">  
  
    <s:textfield label="Nome do Produto"  
                name="produto.nome"/>  
  
    <s:textfield label="Preço"  
                name="produto.preco"/>  
  
    <s:textfield label="Quantidade"  
                name="produto.quantidade"/>  
  
    <s:datetimepicker label="Data de Compra"  
                      name="produto.dataCompra"/>  
  
    <s:submit value="Cadastrar Produto"/>  
  
</s:form>  
  
</body>  
  
</html>
```

---

```
package control;  
  
import javax.servlet.http.HttpServletRequest;  
  
import org.apache.struts2.ServletActionContext;  
  
import persistence.ProdutoDao;  
  
import com.opensymphony.xwork2.ActionContext;  
import com.opensymphony.xwork2.ActionSupport;  
import com.opensymphony.xwork2.ModelDriven;  
  
import entity.Produto;  
  
public class ControleProduto extends ActionSupport implements  
ModelDriven<Produto>{  
  
    //Atributo para capturar os dados do formulário de cadastro  
    private Produto produto; //null (sem espaço de memória)  
  
    public ControleProduto() { //construtor padrão  
        produto = new Produto(); //inicializar o atributo  
    }
```



```
//Método para realizar o cadastro do produto
public String cadastrar(){

    String mensagem = null;

    try{

        ProdutoDao d = new ProdutoDao();
        d.save(produto);

        mensagem = "Produto cadastrado com sucesso.";
    }
    catch(Exception e){

        mensagem = "Erro -> " + e.getMessage();
    }

    HttpServletRequest request = (HttpServletRequest)
        ActionContext.getContext().
        get(ServletActionContext.HTTP_REQUEST);
    //gerando um objeto request

    //gerar mensagem para o usuário
    request.setAttribute("msg", mensagem);

    //retornar constantes
    return SUCCESS;
}

public Produto getProduto() {
    return produto;
}

public void setProduto(Produto produto) {
    this.produto = produto;
}

@Override
public Produto getModel() {
    //retornar o atributo da Classe que representa
    //a modelagem de dados
    return produto;
}

}
```



## Mapeamento do Controle no struts.xml

Arquivo de configuração padrão do struts 2

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

    <!-- Mapear todas as ações do sistema
        relacionadas a Produto -->
    <package name="produto" extends="struts-default">

        <!-- Mapeamento da ação de cadastro de produtos -->
        <action name="cadastrarproduto"
            class="control.ControleProduto"
            method="cadastrar">

            <!-- Mapeamento do retorno do método -->
            <result name="success">cadastro.jsp</result>

        </action>

    </package>

</struts>

-----
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="/struts-tags" prefix="s" %>

<html>

    <head>
        <s:head/>
    </head>

    <body>

        <h3>Cadastro de Produtos</h3>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

        <s:form method="post"
            action="cadastrarproduto.action">

            <s:textfield label="Nome do Produto"
                name="produto.nome"/>

    </body>
</html>
```





# Java WebDeveloper - BRQ

Segunda-feira, 23 de Junho de 2014

Desenvolvimento Java Web com Struts 2.  
Acesso a banco de dados com Hibernate e JPA.

Aula  
**28**

```
<s:textfield label="Preço"
             name="produto.preco"/>

<s:textfield label="Quantidade"
             name="produto.quantidade"/>

<s:datepicker label="Data de Compra"
               name="produto.dataCompra"/>

<s:submit value="Cadastrar Produto"/>

<p>
    ${msg}
</p>

</s:form>

</body>

</html>
```

Executando...

Cadastro de Produtos

[Voltar](#) para a página inicial.

Produto cadastrado com sucesso.

Nome do Produto:

Preço:

Quantidade:

Data de Compra:





No banco de dados...

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

mysql> select * from produto;
+-----+-----+-----+-----+-----+
| idproduto | nome | preco | quantidade | datacompra |
+-----+-----+-----+-----+-----+
| 1 | Mouse | 30 | 10 | 2014-06-23 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Criando o método na Classe de Controle para consultar produtos...

```
package control;

import java.util.List;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.ServletActionContext;
import persistence.ProdutoDao;

import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

import entity.Produto;

public class ControleProduto extends ActionSupport implements
ModelDriven<Produto>{

    //Atributo para capturar os dados do formulário de cadastro
    private Produto produto; //null (sem espaço de memória)

    //Atributo para gerar uma lista com todos os produtos
    //do banco de dados
    private List<Produto> listagemProdutos;
    //null (sem espaço de memória)

    public ControleProduto() { //construtor padrão
        produto = new Produto(); //inicializar o atributo
    }

    //Método para realizar o cadastro do produto
    public String cadastrar(){

        String mensagem = null;

        try{
```



```
        ProdutoDao d = new ProdutoDao();
        d.save(produto);

        mensagem = "Produto cadastrado com sucesso.";
    }
    catch(Exception e){
        mensagem = "Erro -> " + e.getMessage();
    }

    HttpServletRequest request = (HttpServletRequest)
        ActionContext.getContext().
        get(ServletActionContext.HTTP_REQUEST);
    //gerando um objeto request

    //gerar mensagem para o usuario
    request.setAttribute("msg", mensagem);

    //retornar constantes
    return SUCCESS;
}

//Método para carregar conteudo no atributo List<Produto>
public String consultar(){

    try{

        ProdutoDao d = new ProdutoDao();

        //executar a consulta de produtos no
        //banco e popular a lista da classe de controle
        listagemProdutos = d.findAll();
        //listar todos os produtos
    }
    catch(Exception e){
        //imprimir a pilha de erro no
        //console do tomcat (servidor)
        e.printStackTrace();
    }

    return SUCCESS;
}

public Produto getProduto() {
    return produto;
}

public void setProduto(Produto produto) {
    this.produto = produto;
}
```



```
public List<Produto> getListagemProdutos() {
    return listagemProdutos;
}

public void setListagemProdutos(List<Produto> listagemProdutos)
{
    this.listagemProdutos = listagemProdutos;
}

@Override
public Produto getModel() {
    //retornar o atributo da Classe que representa
    //a modelagem de dados
    return produto;
}
}
```

## Mapeamento do Controle no struts.xml

Arquivo de configuração padrão do struts 2

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

    <!-- Mapear todas as ações do sistema
        relacionadas a Produto -->
    <package name="produto" extends="struts-default">

        <!-- Mapeamento da ação de cadastro de produtos -->
        <action name="cadastrarproduto"
            class="control.ControleProduto"
            method="cadastrar">
            <!-- Mapeamento do retorno do método -->
            <result name="success">cadastro.jsp</result>
        </action>

        <!-- Mapeamento da ação de consulta de produtos -->
        <action name="consultarproduto"
            class="control.ControleProduto"
            method="consultar">
            <!-- Mapeamento do retorno do método -->
            <result name="success">consulta.jsp</result>
        </action>

    </package>
</struts>
```



### index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<html>

    <head>
        </head>

    <body>

        <h3>Projeto Controle de Produtos</h3>
        <hr/>

        <ul>
            <li> <a href="cadastro.jsp">
                Cadastrar Produtos
            </a> <
        /li>
        <li> <a href="consultarproduto.action">
                Consultar Produtos
            </a>
        </li>
    </ul>

    </body>
</html>
```

---

### consulta.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@taglib uri="/struts-tags" prefix="s" %>

<html>

    <head>
        <s:head/> <!-- Ativar os componentes de estilo
                    do struts 2 -->
    </head>

    <body>

        <h3>Consulta de Produtos</h3>
```



# Java WebDeveloper - BRQ

Segunda-feira, 23 de Junho de 2014

Desenvolvimento Java Web com Struts 2.  
Acesso a banco de dados com Hibernate e JPA.

Aula  
**28**

```
<a href="index.jsp">Voltar</a> para a página inicial
<hr/>

<table border="1" style="width: 80%">

    <thead>
        <tr>
            <th>Código</th>
            <th>Nome do Produto</th>
            <th>Preço</th>
            <th>Quantidade</th>
            <th>Data de Compra</th>
        </tr>
    </thead>

    <tbody>
        <s:iterator value="ListagemProdutos">
            <tr>
                <td> <s:property value="idProduto"/> </td>
                <td> <s:property value="nome"/> </td>
                <td> <s:property value="preco"/> </td>
                <td> <s:property value="quantidade"/> </td>
                <td> <s:property value="dataCompra"/> </td>
            </tr>
        </s:iterator>
    </tbody>

    <tfoot>
        <tr>
            <td colspan="5">Quantidade de Produtos:
            <s:property value="listagemProdutos.
                size()"/> </td>
        </tr>
    </tfoot>
</table>

</body>

</html>
```



# Java WebDeveloper - BRQ

Segunda-feira, 23 de Junho de 2014

Desenvolvimento Java Web com Struts 2.  
Acesso a banco de dados com Hibernate e JPA.

Aula

28

## Resultado...



### Cadastro de Produtos

[Voltar](#) para a página inicial.

Produto cadastrado com sucesso.

*Nome do Produto:*

*Preço:*

*Quantidade:*

*Data de Compra:*



### Consulta de Produtos

[Voltar](#) para a página inicial

Código	Nome do Produto	Preço	Quantidade	Data de Compra
3	Computador	1200.0	10	23/06/14
2	Monitor	200.0	5	24/06/14
1	Mouse	30.0	10	23/06/14
Quantidade de Produtos: 3				



# Java WebDeveloper - BRQ

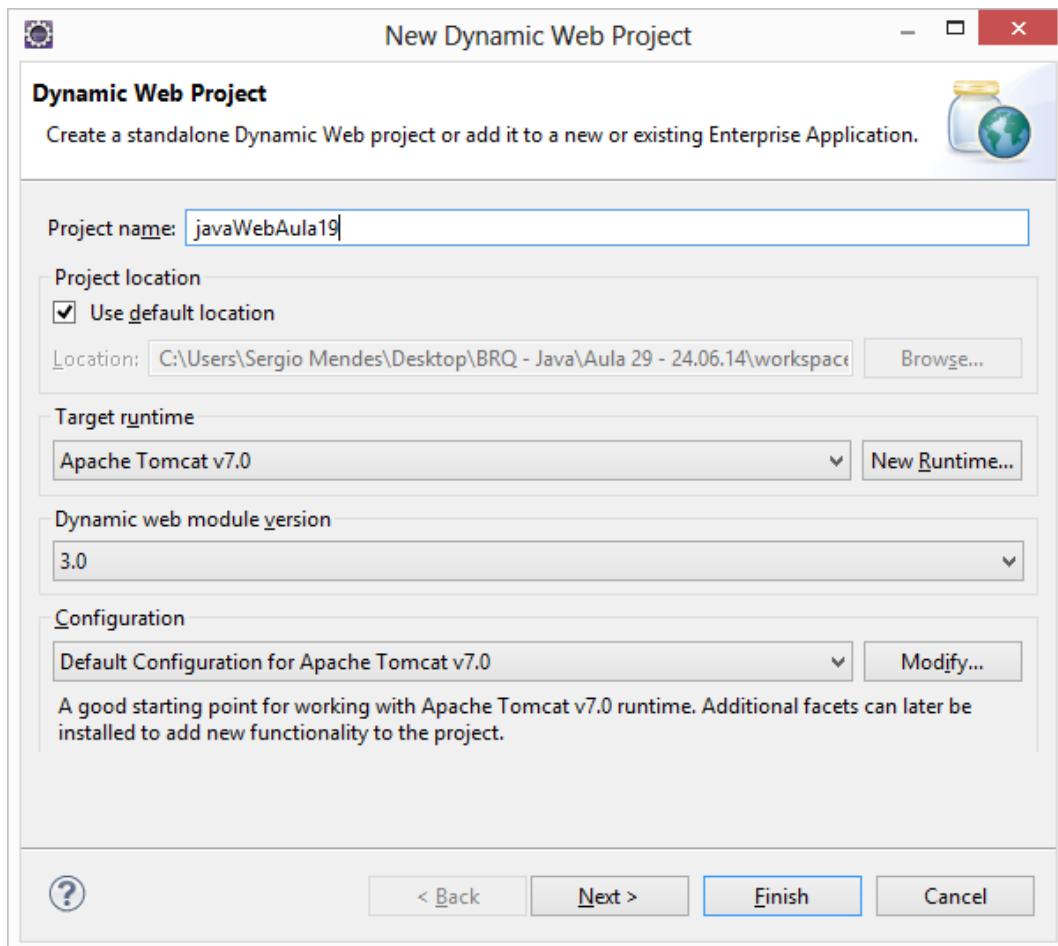
Terça-feira, 24 de Junho de 2014

Teste de Software com JUnit.

Aula

29

Novo Projeto...



Script do banco de dados...

```
drop database if exists aulaweb19;
create database aulaweb19;
use aulaweb19;

create table cliente(
    idcliente      integer          auto_increment,
    nome           varchar(50)       not null,
    email          varchar(50)       not null,
    cpf            varchar(11)        not null unique,
    primary key(idcliente));

show tables;

desc cliente;

insert into cliente values(null, 'Sergio Mendes',
'sergio.coti@gmail.com', '12345678900');

select * from cliente;
```



## Classe de entidade (JavaBean)

JPA - Java Persistence API

```
package entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "cliente")
@NamedQueries(
{
    @NamedQuery(name="cliente.obterporcpf",
                query="select c from Cliente as c
                      where c.cpf = :cpf")
}
)
public class Cliente {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idcliente")
    private Integer idCliente;

    @Column(name = "nome", nullable = false, length = 50)
    private String nome;

    @Column(name = "email", nullable = false, length = 50)
    private String email;

    @Column(name = "cpf", nullable = false, length = 11,
            unique = true)
    private String cpf;

    public Cliente() {
    }

    public Cliente(Integer idCliente, String nome,
                  String email, String cpf) {
        super();
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
        this.cpf = cpf;
    }
}
```



```
@Override
public String toString() {
    return "Cliente [idCliente=" + idCliente + ", nome="
           + nome + ", email=" + email + ", cpf=" + cpf + "]";
}

public Integer getIdCliente() {
    return idCliente;
}

public void setIdCliente(Integer idCliente) {
    this.idCliente = idCliente;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getCpf() {
    return cpf;
}

public void setCpf(String cpf) {
    this.cpf = cpf;
}
}
```

---

## config/mysql\_hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">
            org.hibernate.dialect.MySQLDialect
        </property>
    </session-factory>
</hibernate-configuration>
```



```
<property name="hibernate.connection.driver_class">
    com.mysql.jdbc.Driver
</property>
<property name="hibernate.connection.url">
    jdbc:mysql://localhost:3306/aulaweb19
</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password"></property>

<property name="hibernate.show_sql">true</property>
<property name="hibernate.format_sql">true</property>

<mapping class="entity.Cliente"/>

</session-factory>

</hibernate-configuration>
```

---

## persistence/HibernateUtil.java

```
package hibernate;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {

            sessionFactory = new AnnotationConfiguration().
                configure("config/mysql_hibernate.cfg.xml").
                buildSessionFactory();
        } catch (Throwable ex) {

            System.err.println("Initial SessionFactory
                creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

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



## Persistência de dados...

```
package persistence;

import hibernate.HibernateUtil;

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import entity.Cliente;

public class ClienteDao {

    private Session session;
    private Transaction transaction;
    private Query query;

    //Método para buscar 1 Cliente pelo cpf
    public Cliente find(String cpf) throws Exception{

        session = HibernateUtil.getSessionFactory().openSession();
        query = session.getNamedQuery("cliente.obterporcpf");
        query.setString("cpf", cpf);

        Cliente c = (Cliente) query.uniqueResult();
        //retornar 1 registro

        session.close();
        return c;
    }
}
```

---

## JUnit 4

Framework Java para teste de software do tipo Caixa-Branca  
Realização de Testes unitários, ou seja, que visam validar e verificar  
cada programa escrito no sistema (Classe, método, etc..)

### CST01 - Manter Clientes

#### Roteiro 01

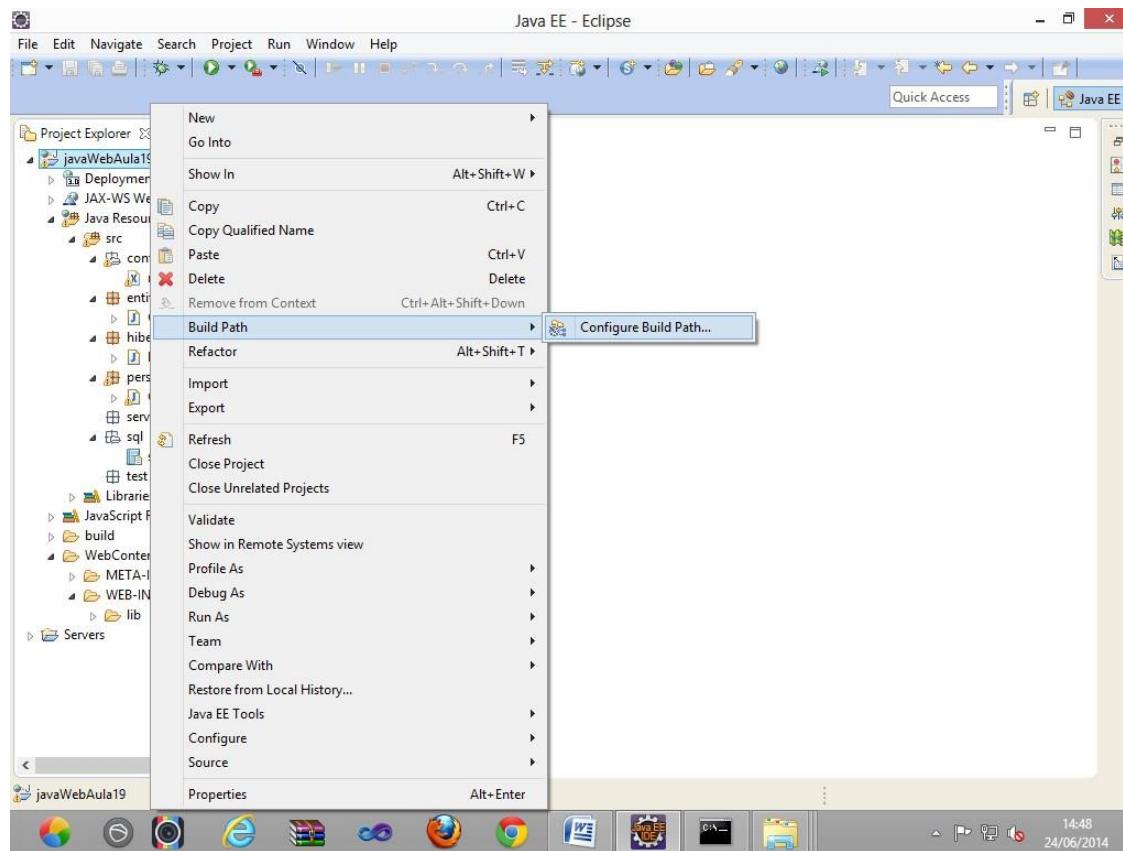
Entrada de dados: CPF 12345678900  
Resultado Esperado: Cliente: Sergio Mendes encontrado

#### Roteiro 02

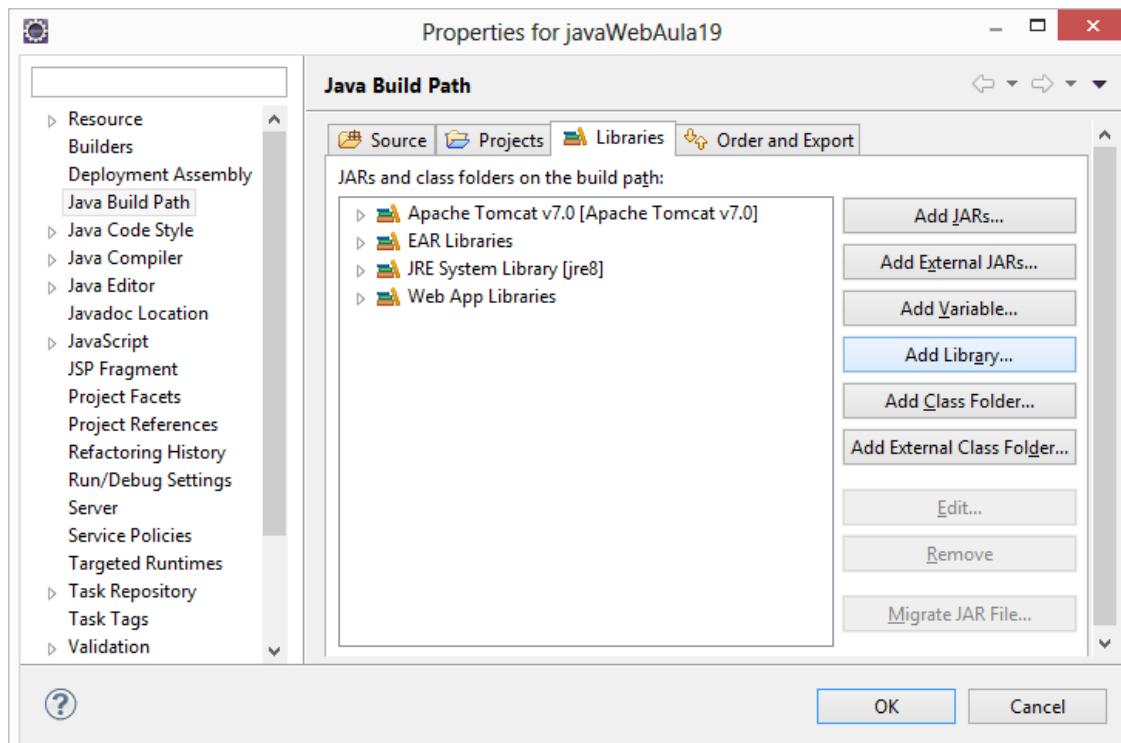
Entrada de dados: CPF 11111111111  
Resultado Esperado: Cliente não encontrado



### Adicionar o JUnit ao projeto...



### Add Library



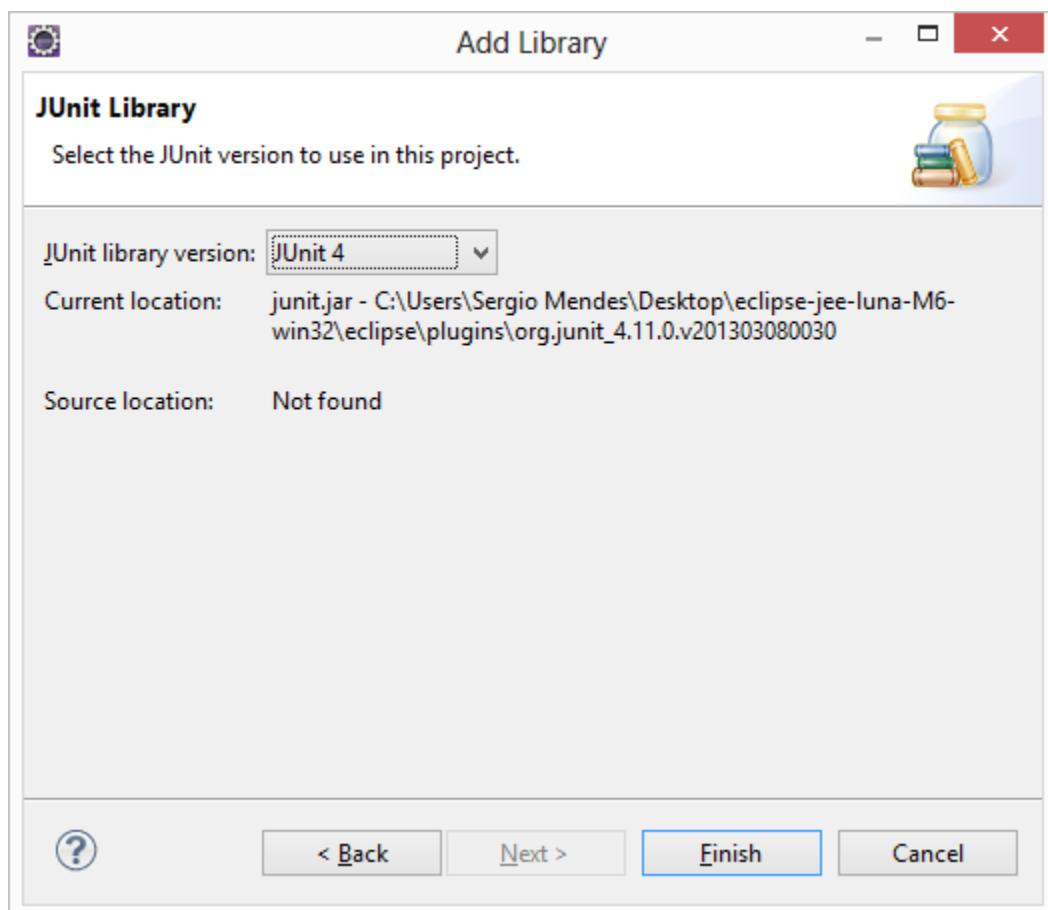
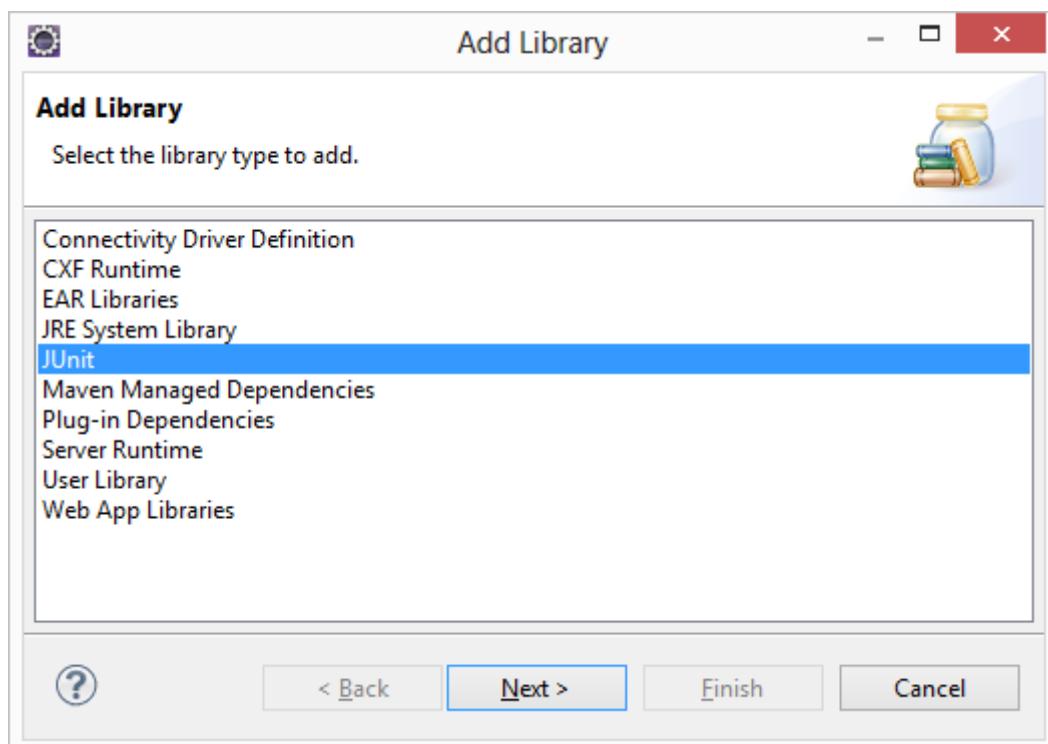


# Java WebDeveloper - BRQ

Terça-feira, 24 de Junho de 2014

Teste de Software com JUnit.

Aula  
29



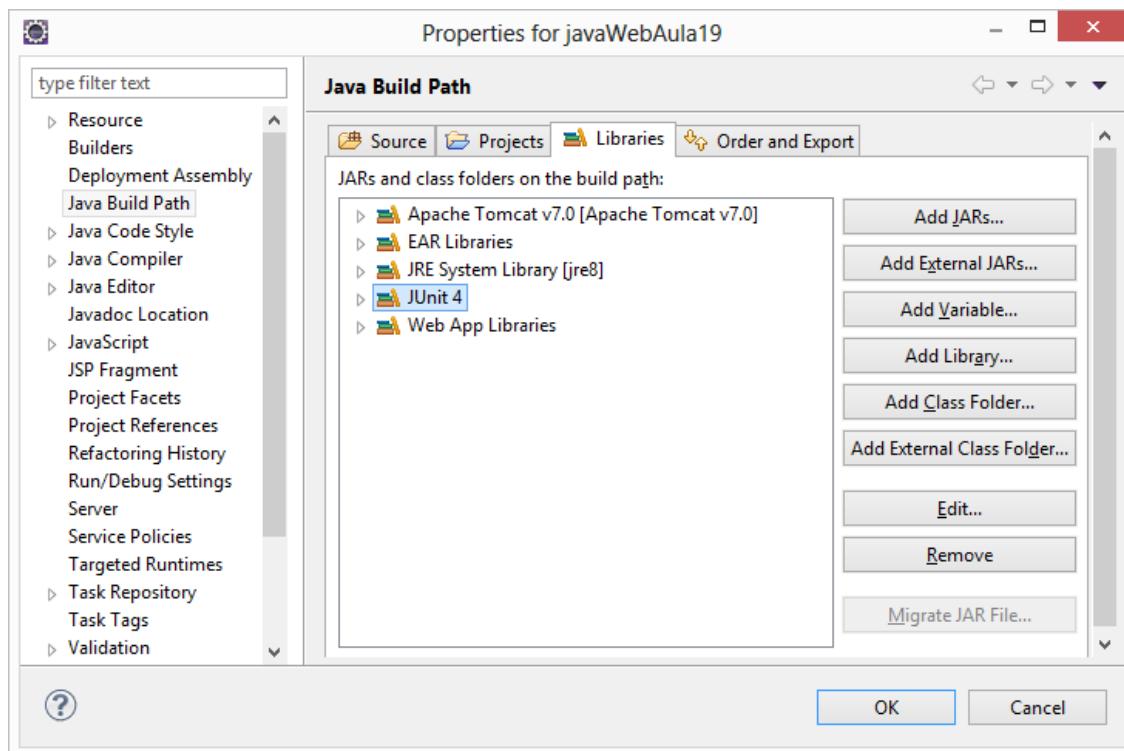


# Java WebDeveloper - BRQ

Terça-feira, 24 de Junho de 2014

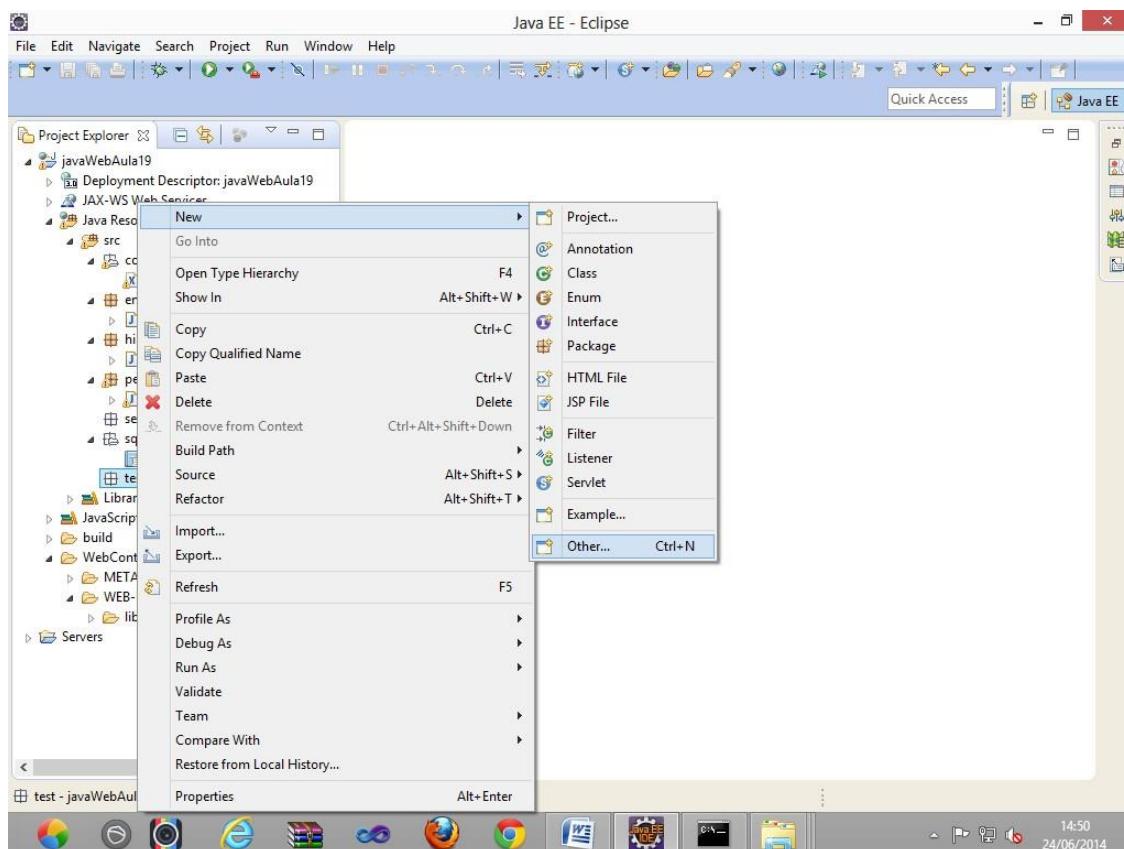
Teste de Software com JUnit.

Aula  
29



## Criando o Caso de Teste (TestCase)

Classe para realização de rotinas de teste no JUnit



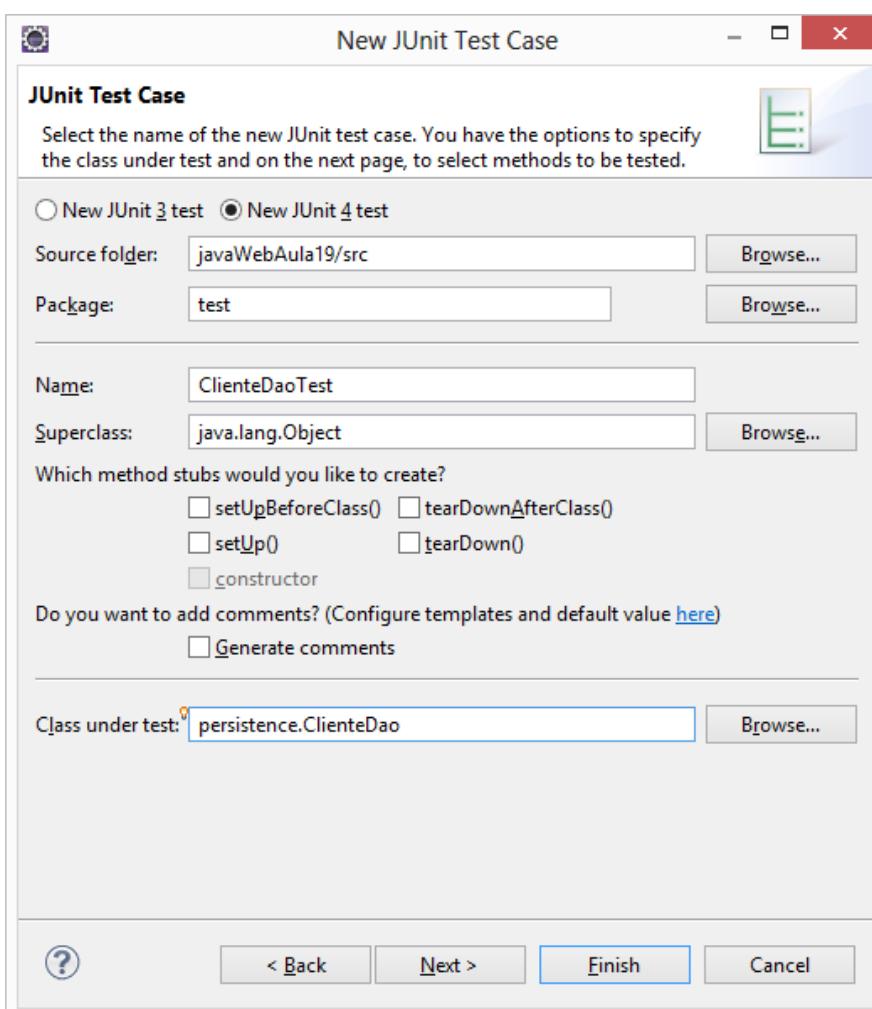
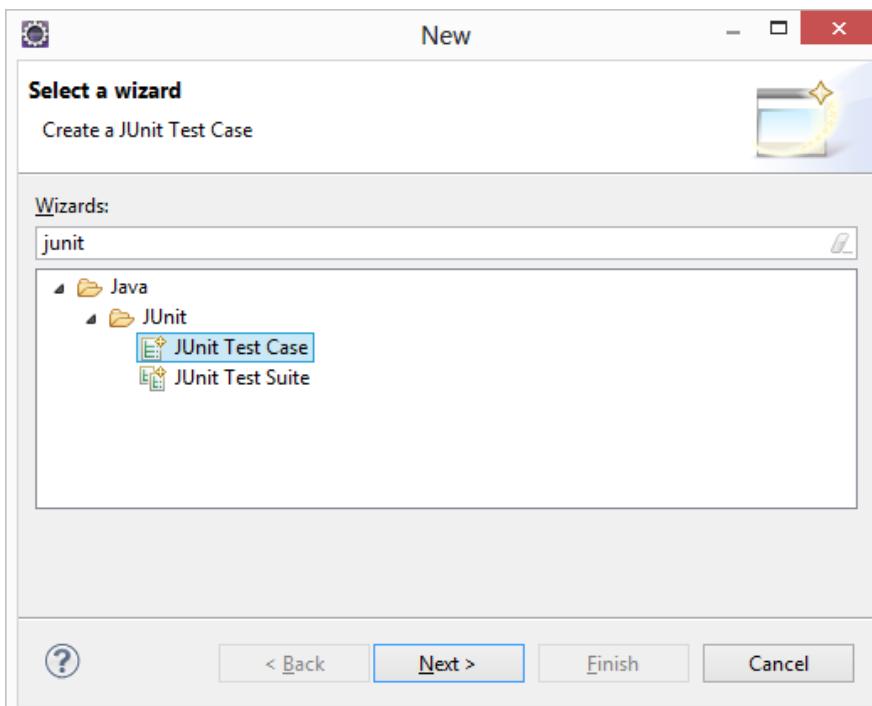


# Java WebDeveloper - BRQ

Terça-feira, 24 de Junho de 2014

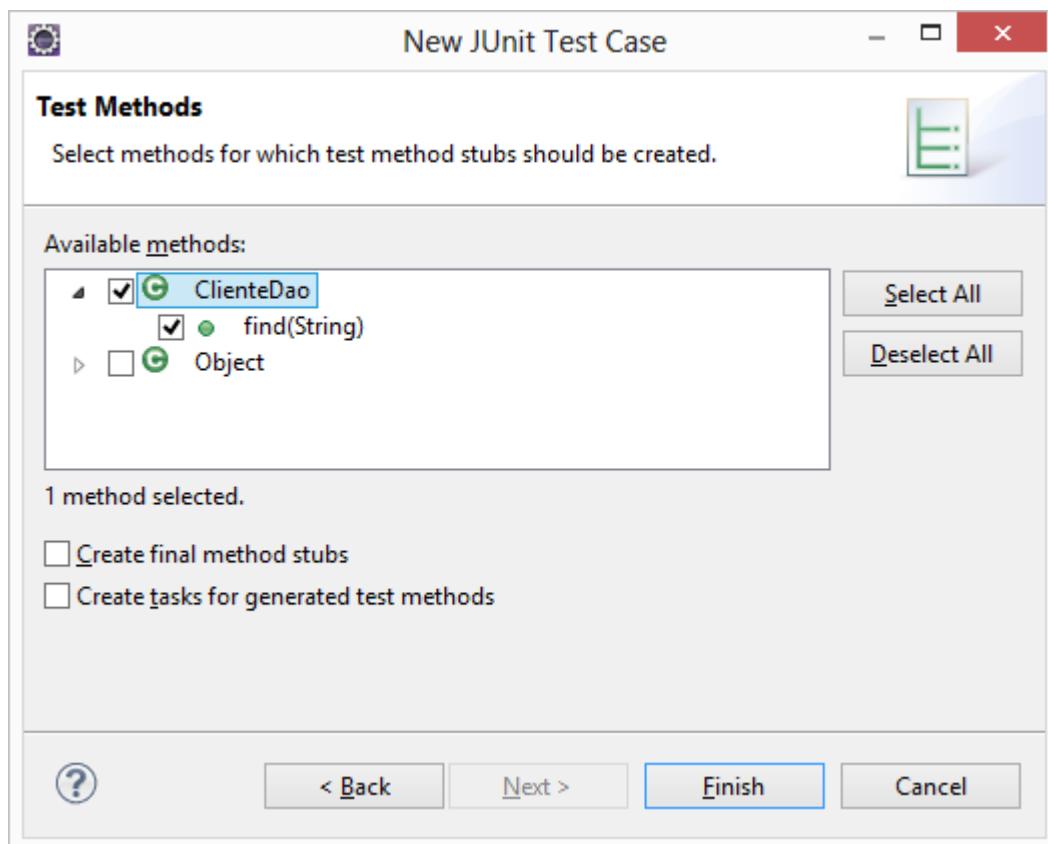
Teste de Software com JUnit.

Aula  
29





## Selecionar os métodos que serão testados

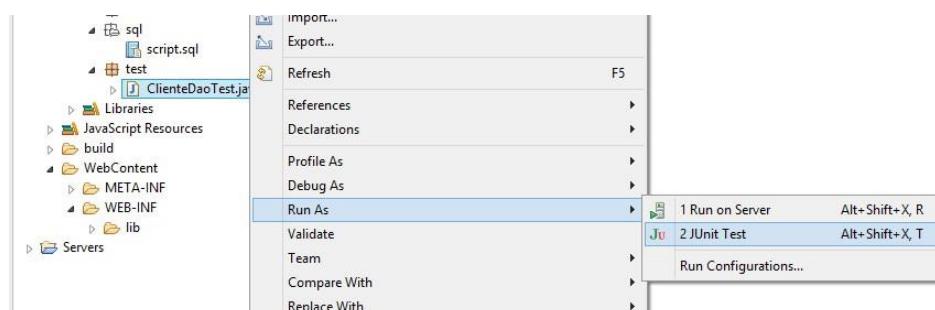


```
package test;

import static org.junit.Assert.*;
import org.junit.Test;

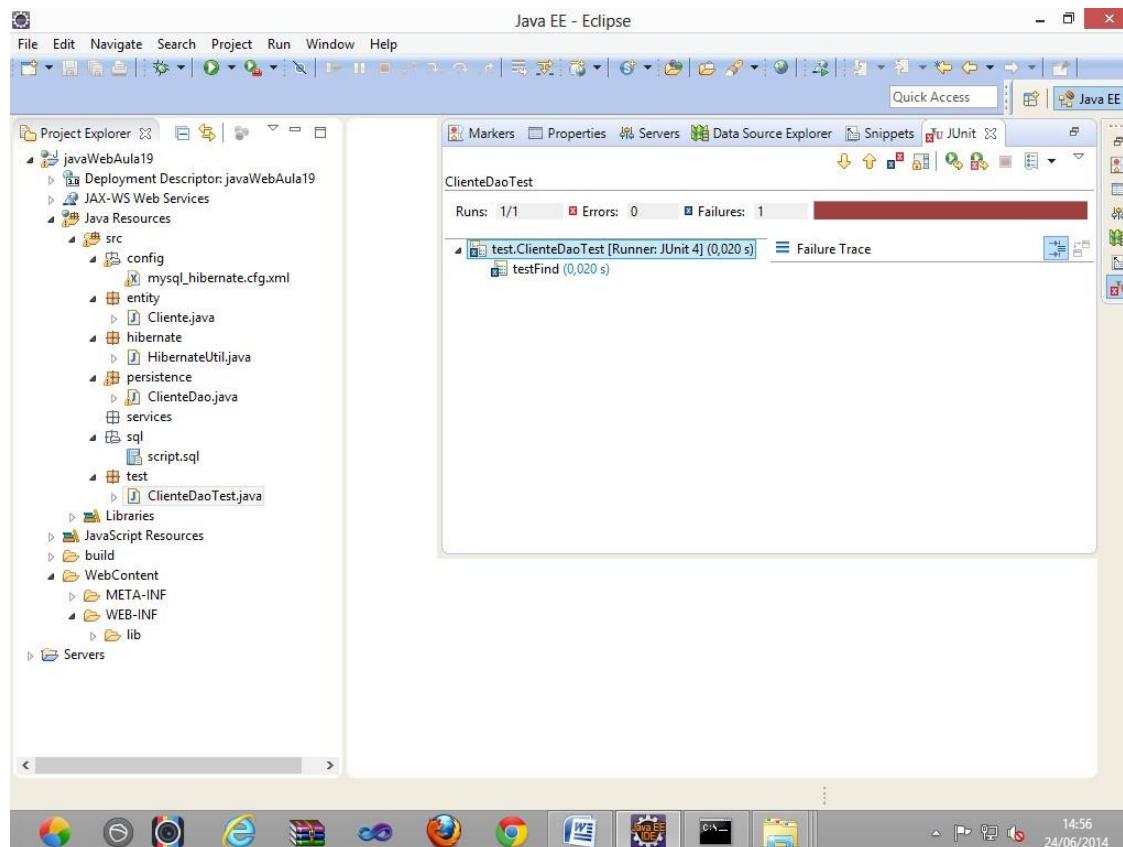
public class ClienteDaoTest {

    @Test
    public void testFind() {
        fail("Not yet implemented");
    }
}
```





Falha...



## CST01 - Manter Clientes

### Roteiro 01

Entrada de dados: CPF 12345678900

Resultado Esperado: Cliente: Sergio Mendes encontrado

### Roteiro 02

Entrada de dados: CPF 11111111111

Resultado Esperado: Cliente não encontrado

```
-----  
package test;  
  
import static org.junit.Assert.*;  
import junit.framework.Assert;  
  
import org.junit.Test;  
  
import persistence.ClienteDao;  
import entity.Cliente;
```



```
public class ClienteDaoTest {  
  
    @SuppressWarnings("deprecation")  
    @Test  
    public void testFind() {  
  
        try{  
  
            ClienteDao d = new ClienteDao();  
  
            //Entrada de dados  
            String cpf1 = "12345678900";  
            String cpf2 = "11111111111";  
  
            //Execução das rotinas  
            Cliente c1 = d.find(cpf1);  
            Cliente c2 = d.find(cpf2);  
  
            //Verificar se o resultado obtido é  
            //igual ao esperado  
  
            //nome do cliente c1 é igual a 'Sergio Mendes'  
            Assert.assertTrue(c1.getNome().  
                equals("Sergio Mendes")); //Passou  
  
            //verificando se o objeto 'c2' é null -> Cliente  
            //não foi encontrado  
            Assert.assertNull(c2); //Passou  
        }  
        catch(Exception e){  
            //Falhou...  
            fail("Falha ao obter Cliente -> " + e.getMessage());  
        }  
    }  
}
```

## ***assertTrue***

Verifica se uma condição produz resultado verdadeiro

```
Assert.assertTrue(c1.getNome().equals  
    ("Sergio Mendes"));
```

## ***assertNull***

Verifica se um parametro passado tem valor null (vazio)

```
Assert.assertNull(c2);
```

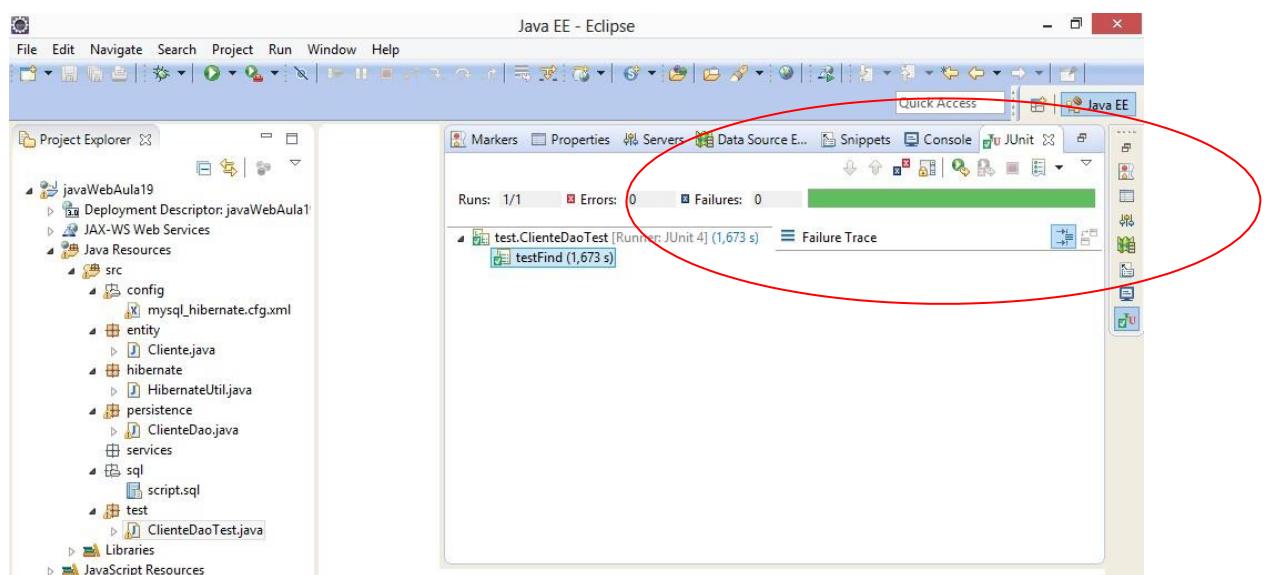
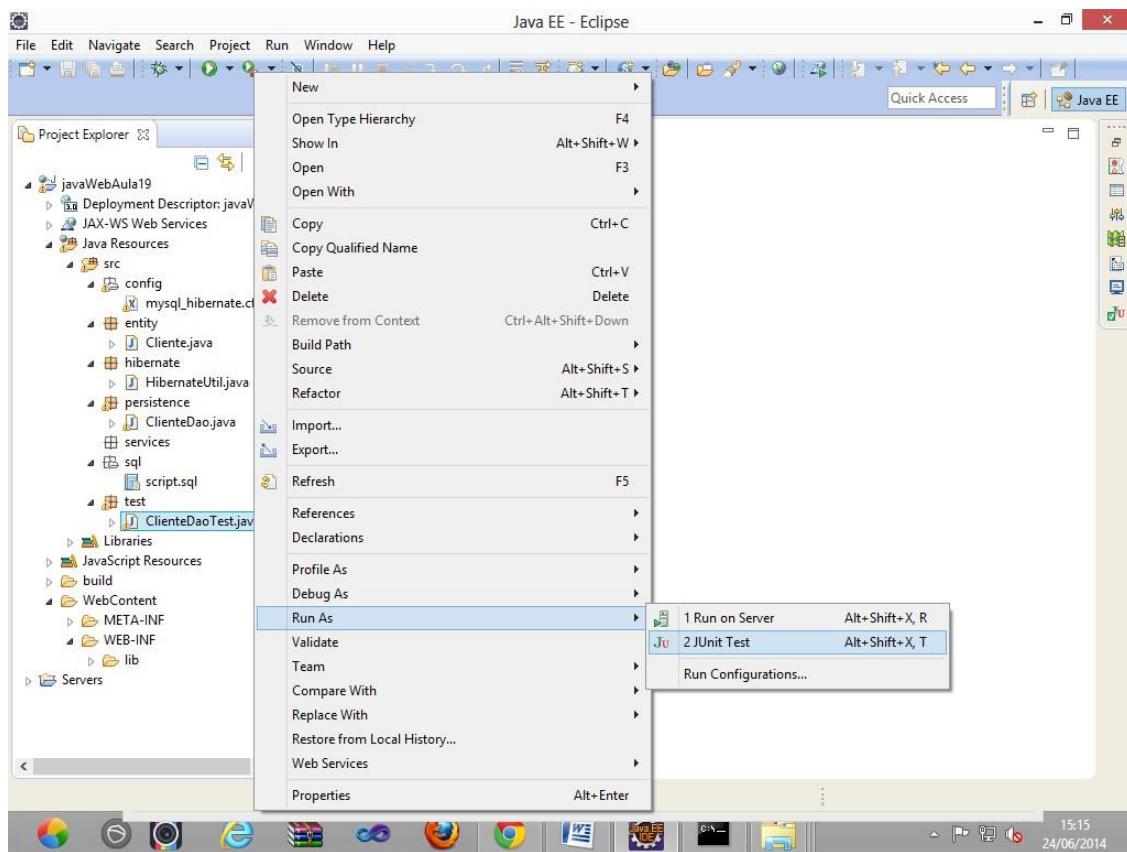


# Java WebDeveloper - BRQ

Terça-feira, 24 de Junho de 2014

Teste de Software com JUnit.

Aula  
29





# Java WebDeveloper - BRQ

Quarta-feira, 25 de Junho de 2014

Publicação de Projetos Web.

Aula

30

Publicação de Projeto JavaWeb no Site cloudbess.net

1- Acessar o site

<http://www.cloudbees.com/>

2- Criar uma conta

Pode criar no proprio site

Link com a sua conta da google.



# Java WebDeveloper - BRQ

Quarta-feira, 25 de Junho de 2014

Publicação de Projetos Web.

Aula  
30

## 3- Selecionar Apps

The screenshot shows the CloudBees GrandCentral dashboard. At the top, there are several service icons: ClickStart (Integrated Services), RDBMS@cloud (Application Service), RDBMS@cloud (Database Service), Forge (Source Repositories), Jenkins CI (Build Service), and WFLAVI@cloud (Data Integration). Below these are sections for 'CloudBees Resources' (Repositories, Applications, Databases, Builds) and 'Getting Started' (with links to Jenkins, RDBMS, Continuous deployment, and CloudBees for Eclipse). A 'Community Links' section features a hexagonal network diagram with various tools like Play, Git, and MySQL. On the right, there's a 'Recent Blog Entries' sidebar with links to Docker on DBA, Scaling Jenkins Horizontally, and other posts. A 'Try our new UI' button is also visible.

## 4- Criar uma Application

The top screenshot shows the CloudBees PaaS Console Applications page. It displays a message 'You don't have any applications created, you might like to create one.' with a 'Create Application' button. The sidebar includes options for Regions (CloudBees - US), Applications (Manage, Add New Application), Databases (Manage, Add New Database), and Ecosystem Services (New Relic Monitoring, Papertrail Logs, SendGrid Mail Server, Websolr Search Indexes, AppDynamics Monitoring). The bottom screenshot shows a 'Create an Application' dialog box. It has fields for 'Application name' (containing 'jpf') and 'Application core runtime' (set to 'JVM Web Application (WAR)'). There are 'Finish' and 'Cancel' buttons at the bottom. The sidebar on the left is identical to the top screenshot.



### 5- Configurado o ambiente da aplicação

The screenshot shows the CloudBees Application Development Center interface. On the left, a sidebar lists 'Regions' (CloudBees - US), 'Applications' (Manage, Add New Application), 'Databases' (Manage, Add New Database), and 'Ecosystem Services' (New Relic Monitoring, Papertrail Logs, SendGrid Mail Server, Websolv Search Indexes, AppDynamics Monitoring). The main panel displays the 'jsf' application details: Owner: [redacted], Created: 2014 Feb 7, Status: active, ID: ivitoria1/jsf, Location: http://jsf.ivitoria1.cloudbees.net. Below this are four icons: download, edit, delete, and refresh. A navigation bar at the top right includes Development, Operations, Logs, and Configuration. The 'Development' tab is selected. The 'Application Development Center' section congratulates the user on the successful deployment and provides instructions for further steps, including terminal commands for building and deploying using the CloudBees SDK. A link to upload a WAR file is also present. The 'Deployments' section shows a single entry: Date: 2014 February 7 18:37:42 UTC-2, Message: Initial create, with an 'Upload new version' button below it.

The screenshot shows the 'Application under development' page for the 'jsf' application. It features a header with the CloudBees logo and the URL jsf.ivitoria1.cloudbees.net. The main content area is titled 'CloudBees' and 'This application under development'. It informs the user that their new application is running successfully on CloudBees RUN@cloud. Below this, a 'What to do next:' section includes a 'Take me to the Console' button and a 'Support' link (Access support, Wiki, Forum etc.). To the right, a 'KnowledgeBase' section lists links to RUN@Cloud knowledgebase and tutorials, discussion forums and Q&A, reporting issues or problems, suggesting ideas or requesting features, announcements, and the CloudBees blog. A 'More...' link is also present.



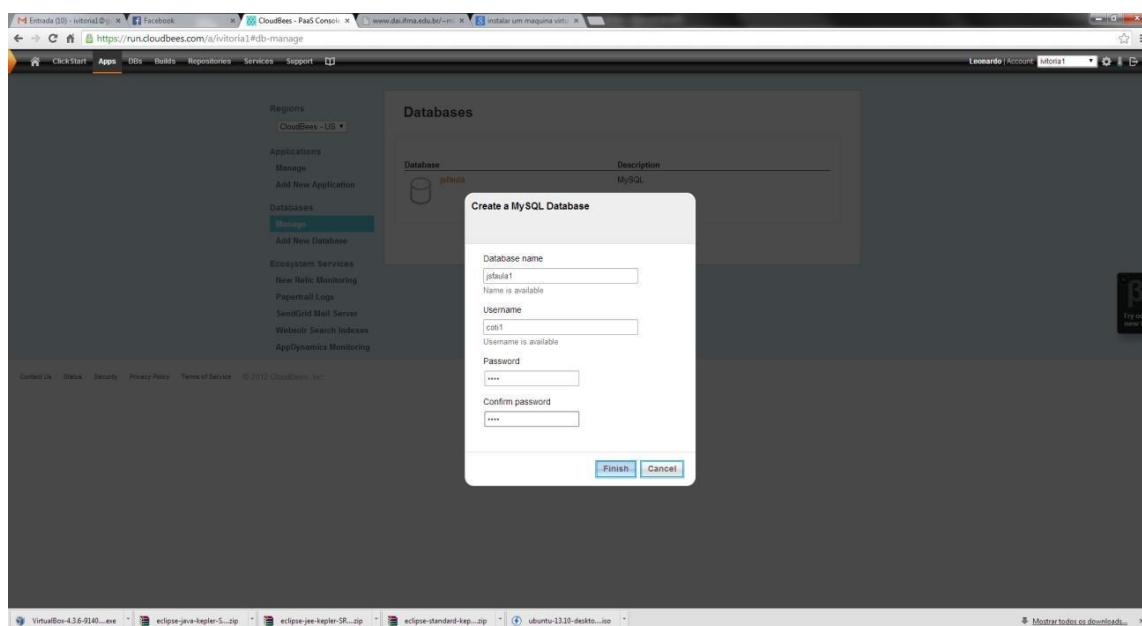
# Java WebDeveloper - BRQ

Quarta-feira, 25 de Junho de 2014

Publicação de Projetos Web.

Aula  
**30**

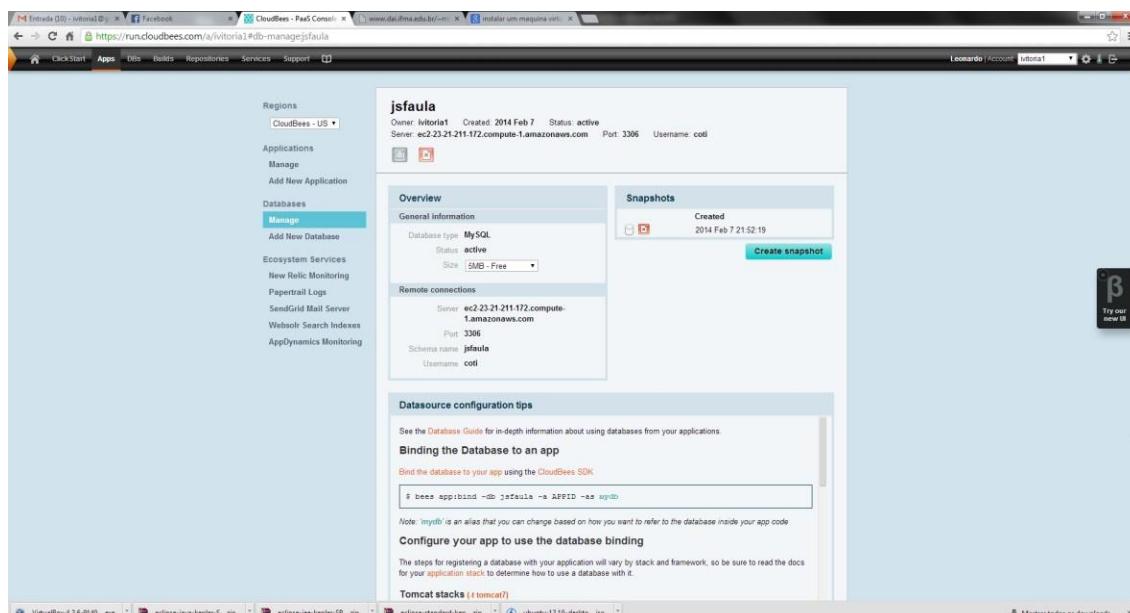
## 6- Criar o banco de dados



The screenshot shows the CloudBees PaaS Console interface. On the left, there's a sidebar with 'Regions' set to 'CloudBees - US', 'Applications' (Manage, Add New Application), 'Databases' (Manage, Add New Database), and 'Ecosystem Services' (New Relic Monitoring, Papertrail Logs, SendGrid Mail Server, Webtier Search Indexes, AppDynamics Monitoring). The main area is titled 'Databases' and shows a table with one entry: 'jsfaula' (MySQL). A modal window titled 'Create a MySQL Database' is open, prompting for 'Database name' (jsfaula1), 'Username' (coti), and 'Password' (coti). Below the fields are 'Confirm password' and 'Finish' (highlighted in blue) and 'Cancel' buttons.

## 7- Na Tela abaixo , pega o caminho do banco no servidor

**Servidor : ec2-23-21-211-172.compute-1.amazonaws.com**  
**Usuario : coti**  
**Senha: coti**

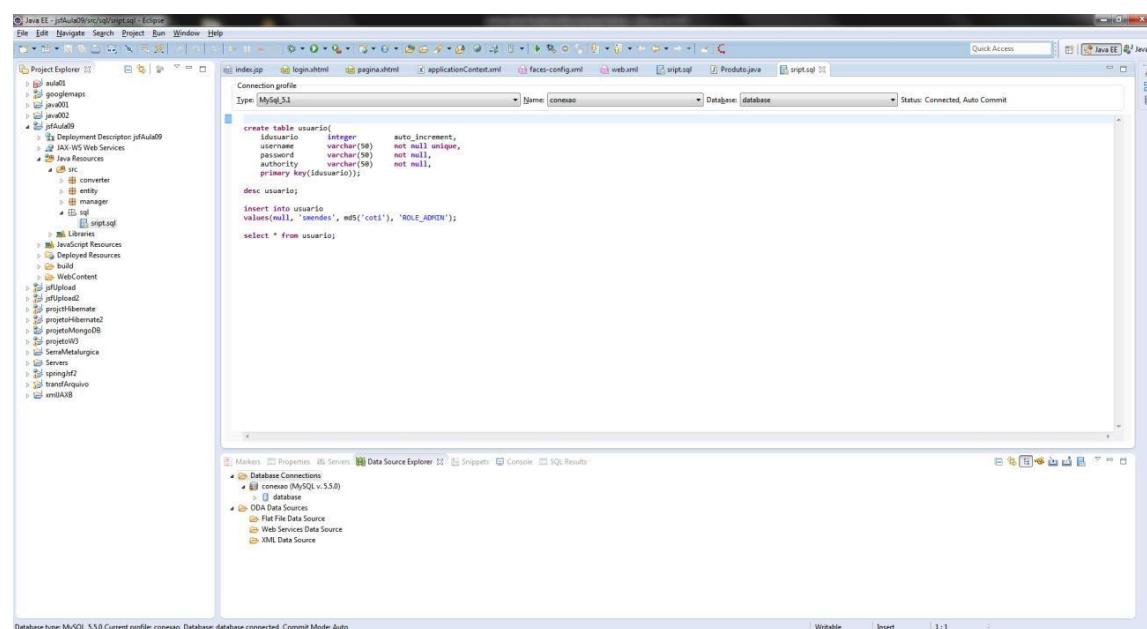


The screenshot shows the CloudBees PaaS Console interface. The left sidebar is identical to the previous screenshot. The main area shows the 'jsfaula' database details. It includes an 'Overview' section with 'General Information' (Owner: ivitoria1, Created: 2014 Feb 7, Status: active, Server: ec2-23-21-211-172.compute-1.amazonaws.com, Port: 3306, Username: coti) and a 'Schemas' section listing 'jsfaula' with 'Username: coti'. Other sections like 'Schemas', 'Tables', and 'Data' are also visible. At the bottom, there are tips for 'Binding the Database to an app' and 'Configure your app to use the database binding'.

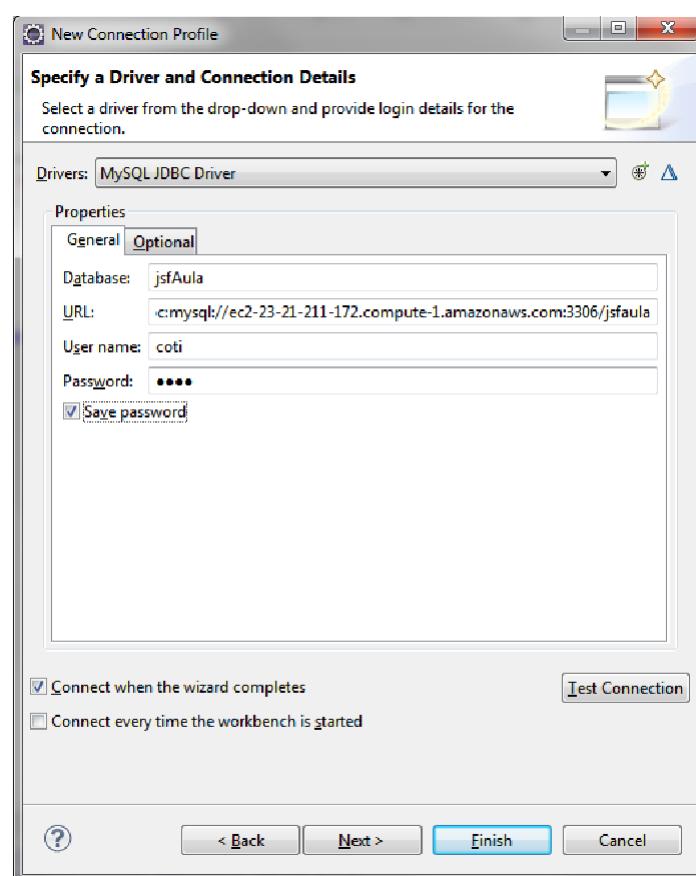


### No Eclipse

#### Criar uma conexão no Data Source Explorer



#### Criando da Conexao no Eclipse





# Java WebDeveloper - BRQ

Quarta-feira, 25 de Junho de 2014

Publicação de Projetos Web.

Aula  
30

Executar as query's escritas no arquivo de script

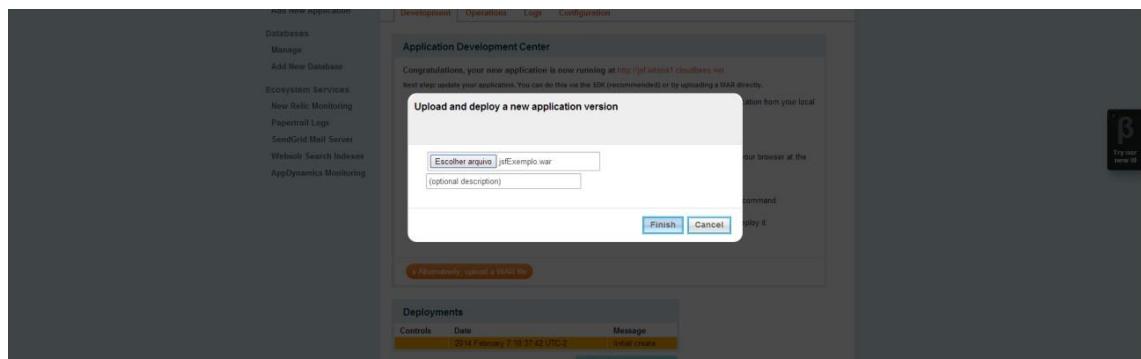
```
create table usuario(
    idusuario      integer      auto_increment,
    username        varchar(50)   not null unique,
    password        varchar(50)   not null,
    authority       varchar(50)   not null,
    primary key(idusuario));

desc usuario;

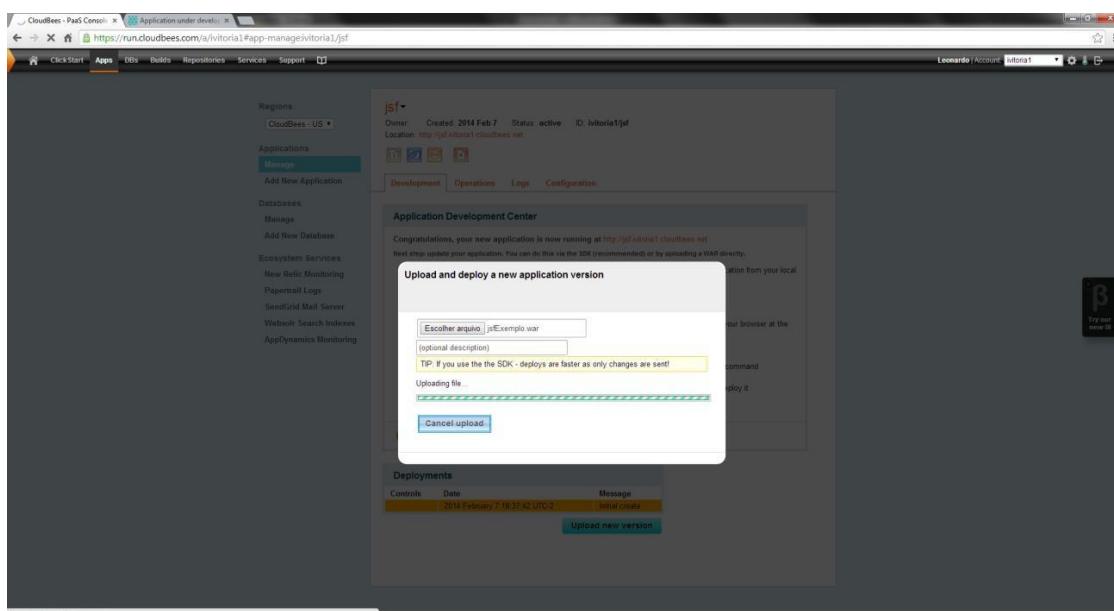
insert into usuario
values(null, 'smendes', md5('coti'), 'ROLE_ADMIN');

select * from usuario;
```

8- Adicionar um projeto war



9 - Carregando o Projeto War





# Java WebDeveloper - BRQ

Quarta-feira, 25 de Junho de 2014

Publicação de Projetos Web.

Aula  
30

## 10 - Projeto War no ambiente

The screenshot shows the CloudBees PaaS Console interface. On the left, there's a sidebar with 'Regions' (CloudBees - US), 'Applications' (Message selected), 'Add New Application', 'Databases', 'Manage', 'Add New Database', 'Ecosystem Services', 'New Relic Monitoring', 'Papertrail Logs', 'SendGrid Mail Server', 'Websolr Search Indexes', and 'AppDynamics Monitoring'. The main area has tabs for 'Development', 'Operations', 'Logs', and 'Configuration'. Under 'Development', it says 'Congratulations, your new application is now running at <http://jsf.itoria1.cloudbees.net>'. It provides instructions for updating the application: 'If you haven't done so already, install the CloudBees SDK so you can build and test your application from your local machine.', 'Use the SDK's create command to get started fast', 'bee create jsf', 'cd jsf', 'Build and test your application updates using the CloudBees SDK run command and pointing your browser at the locally running application (<http://localhost:8080>)', 'Use your favorite tools to edit the application source code', 'When ready, deploy your updated application to CloudBees using the CloudBees SDK deploy command', 'bee deploy', and 'If you already have your own WAR-based project, just use the bee app deploy command to deploy it', 'bee app-deploy WAR\_FILE -a itoria1/jsf'. Below this is a 'Deployments' section with a table:

Controls	Date	Message
	2014 February 7 18:59:21 UTC-2	Uploaded
	2014 February 7 18:37:42 UTC-2	Initial create

At the bottom right of the main area, there's a 'Try our new UI' button.

## 11 - Executando na web

The screenshot shows a web browser window with the URL [jsf.itoria1.cloudbees.net/login.jsf](http://jsf.itoria1.cloudbees.net/login.jsf). The page title is 'Login de Usuários'. It contains two input fields: 'Login de Acesso:' and 'Senha:', and a 'Acessar Sistema' button.

link da Aplicação exemplo  
<http://jsf.sergio.cloudbees.net/pagina.jsf>



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro – RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)



# COTI Informática Java Design Patterns Apostila

---

COTI Informática - Av. Rio Branco, 185 - Sala 904  
Rio de Janeiro – RJ Tel. 21-2262-9043  
[www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

### Design Patterns - Introdução

Os padrões de projeto buscam descrever uma solução geral reutilizável para um problema recorrente no desenvolvimento de sistemas de *software* orientados a objetos.

Não é um código final, é uma descrição ou modelo de como resolver o problema do qual trata, que pode ser usada em muitas situações diferentes.

Os Padrões de Projeto normalmente definem as relações e interações entre as classes ou objetos, sem especificar os detalhes das classes ou objetos envolvidos, ou seja, estão num nível de generalidade mais alto.

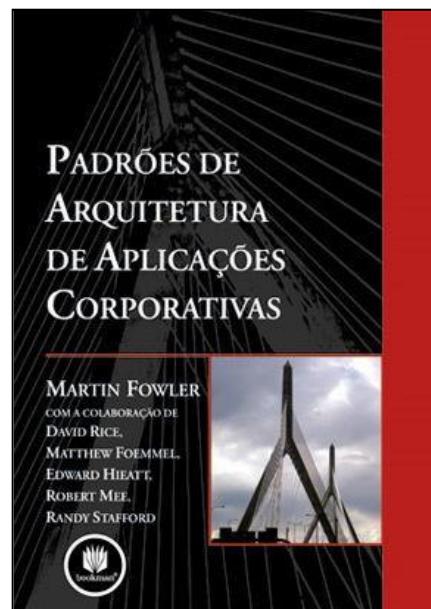
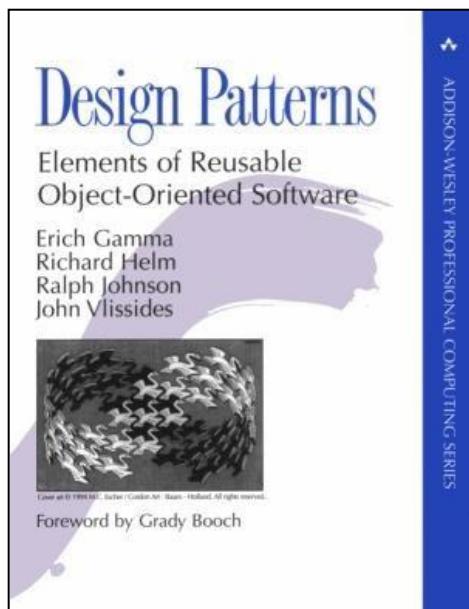
#### Os padrões de projeto:

- Visam facilitar a reutilização de soluções de desenho - isto é, soluções na fase de projeto do software.
- Estabelecem um vocabulário comum de desenho, facilitando comunicação, documentação e aprendizado dos sistemas desoftware.

Os design patterns são organizados em três famílias:

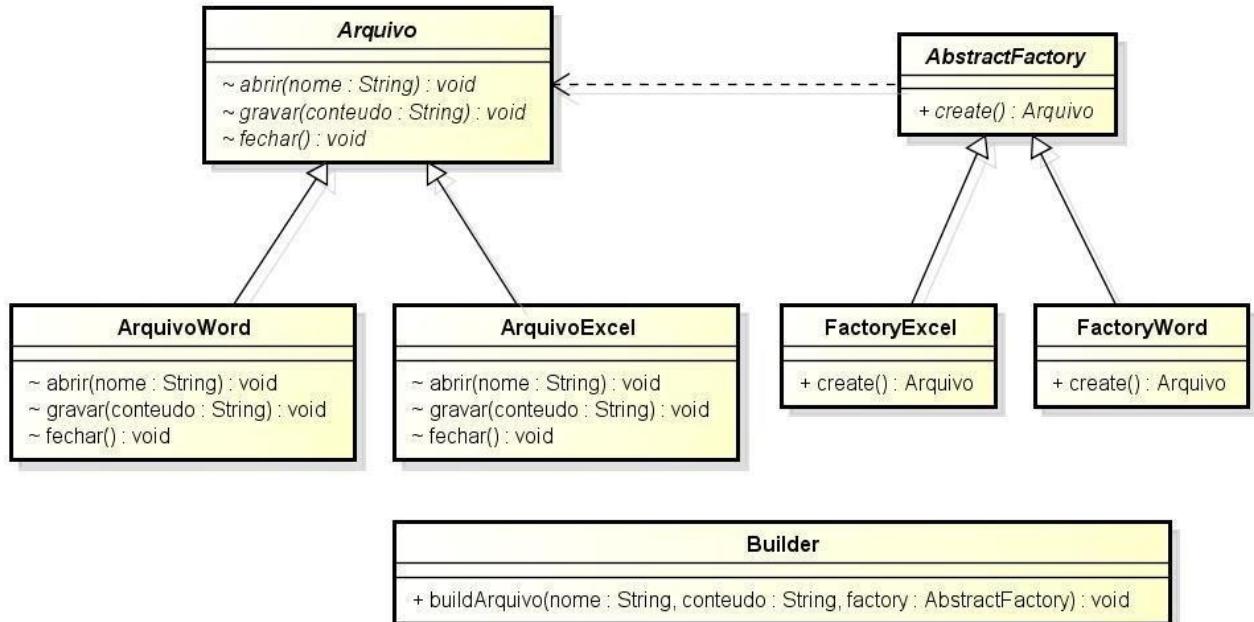
- **Padrões de criação:** relacionados à criação de objetos
- **Padrões estruturais:** tratam das associações entre classes e objetos.
- **Padrões comportamentais:** tratam das interações e divisões de responsabilidades entre as classes ou objetos.

#### Bibliografia recomendada:



## Abstract Factory

Este padrão permite a criação de famílias de objetos relacionados ou dependentes por meio de uma única interface e sem que a classe concreta seja especificada.



```

package abstractfactory;

public abstract class AbstractFactory {

    public abstract Arquivo create();

}

package abstractfactory;

import java.io.FileWriter;

public abstract class Arquivo {

    protected FileWriter arquivo;

    abstract void abrir(String nome) throws Exception;

    abstract void gravar(String conteudo) throws Exception;

    abstract void fechar() throws Exception;

}

package abstractfactory;

import java.io.File;
import java.io.FileWriter;
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
public class ArquivoExcel extends Arquivo {

    @Override
    void abrir(String nome) throws Exception {
        arquivo = new FileWriter(new File("d:\\\\" + nome + ".csv"));
    }

    @Override
    void gravar(String conteudo) throws Exception {
        arquivo.write(conteudo);
    }

    @Override
    void fechar() throws Exception {
        arquivo.close();
    }
}

package abstractfactory;

import java.io.File;
import java.io.FileWriter;

public class ArquivoWord extends Arquivo {

    @Override
    void abrir(String nome) throws Exception {
        arquivo = new FileWriter(new File("d:\\\\" + nome + ".doc"));
    }

    @Override
    void gravar(String conteudo) throws Exception {
        arquivo.write(conteudo);
    }

    @Override
    void fechar() throws Exception {
        arquivo.close();
    }
}

package abstractfactory;

public class Builder {

    public void buildArquivo(String nome, String conteudo,
                            AbstractFactory factory) throws
    Exception{

        Arquivo a = factory.create();

        a.abrir(nome);
        a.gravar(conteudo);
        a.fechar();
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
}
```

```
package abstractfactory;
```

```
public class FactoryExcel extends AbstractFactory {
```

```
    @Override
    public Arquivo create() {
        return new ArquivoExcel();
    }
}
```

```
package abstractfactory;
```

```
public class FactoryWord extends AbstractFactory {
```

```
    @Override
    public Arquivo create() {
        return new ArquivoWord();
    }
}
```

```
package abstractfactory;
```

```
import java.util.Scanner;
```

```
public class Main {
```

```
    public static void main(String[] args) {
        Builder builder = new Builder();
        AbstractFactory factory = null;

        try {
            System.out.println("Tipo: ");
            String opcao = new Scanner(System.in).nextLine();

            if (opcao.equalsIgnoreCase("word")) {
                factory = new FactoryWord();
            } else if (opcao.equalsIgnoreCase("excel")) {
                factory = new FactoryExcel();
            }

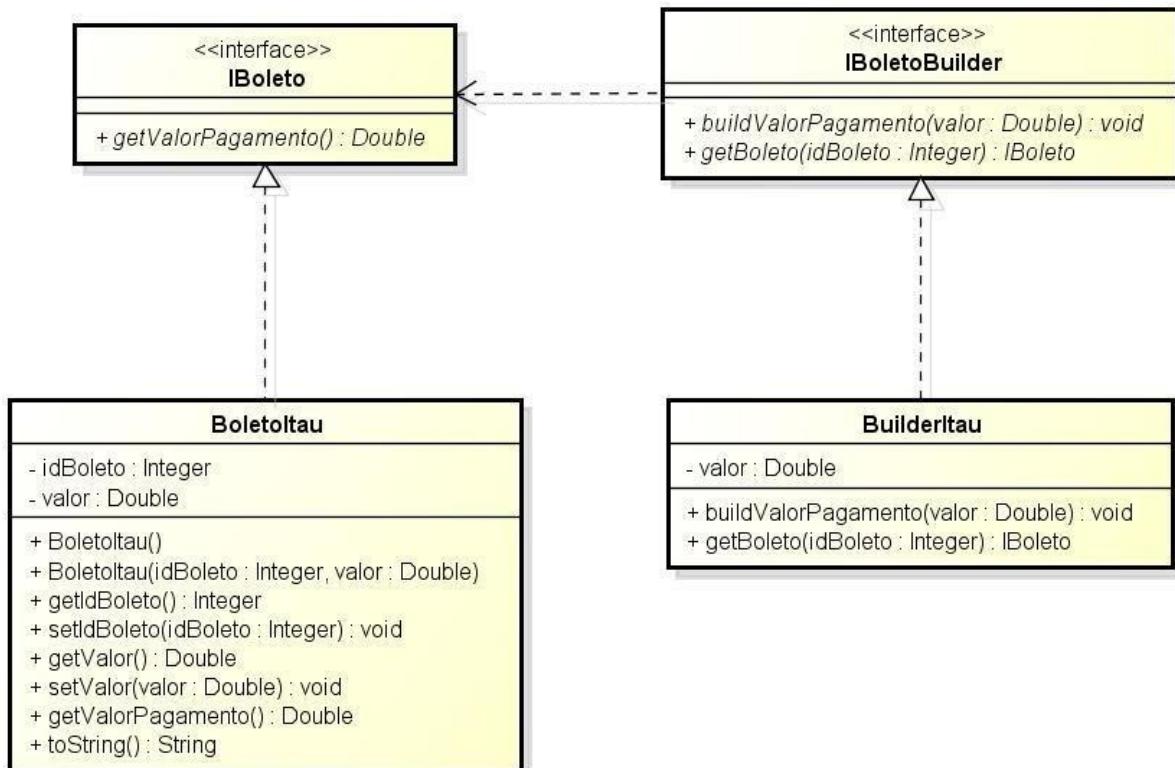
            builder.buildArquivo("teste", "Testando
                                  Patterns", factory);

            System.out.println("Dados gravados");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Builder

Permite a separação da construção de um objeto complexo da sua representação, de forma que o mesmo processo de construção possa criar diferentes representações.



```

package builder;

public class Boletoltau implements IBoleto {

    private Integer idBoleto;
    private Double valor;

    public Boletoltau() {
    }

    public Boletoltau(Integer idBoleto, Double valor) {
        super();
        this.idBoleto = idBoleto;
        this.valor = valor;
    }

    public Integer getIdBoleto() {
        return idBoleto;
    }

    public void setIdBoleto(Integer idBoleto) {
        this.idBoleto = idBoleto;
    }

    public Double getValor() {
        return valor;
    }

    public void setValor(Double valor) {
        this.valor = valor;
    }

    public Double getValorPagamento() {
        return valor;
    }

    @Override
    public String toString() {
        return "Boletoltau [idBoleto=" + idBoleto + ", valor=" + valor + "]";
    }
}
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
}

public Double getValor() {
    return valor;
}

public void setValor(Double valor) {
    this.valor = valor;
}

@Override
public Double getValorPagamento() {
    return valor * 1.12;
}

@Override
public String toString() {
    return "BoletoItau [idBoleto=" + idBoleto
        + ", valor=" + valor
        + ", Valor Pagamento =" + getValorPagamento()
        + "]";
}

}

package builder;

public class BuilderItau implements IBoletoBuilder {

    private Double valor;

    @Override
    public void buildValorPagamento(Double valor) {
        this.valor = valor;
    }

    @Override
    public IBoleto getBoleto(Integer idBoleto) {
        return new BoletoItau(idBoleto, valor);
    }

}

package builder;

public interface IBoleto {

    Double getValorPagamento();

}
package builder;
```

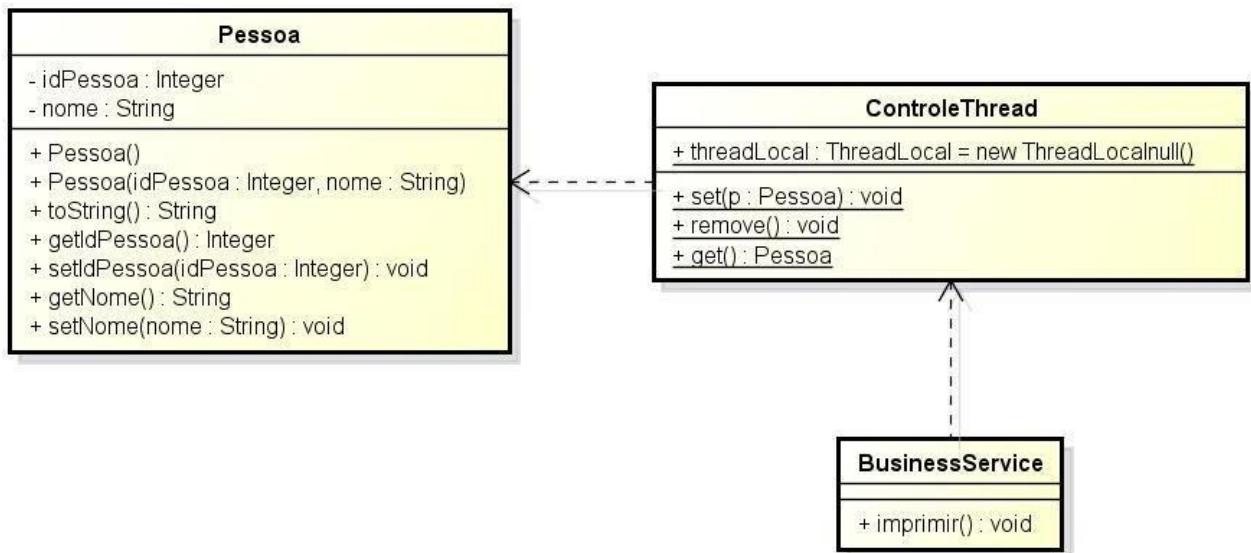


## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 www.cotiinformatica.com.br

```
public interface IBoletoBuilder {  
  
    void buildValorPagamento(Double valor);  
    IBoleto getBoleto(Integer idBoleto);  
}  
package builder;  
  
public class Main {  
  
    IBoletoBuilder builder;  
  
    public Main(IBoletoBuilder builder) {  
        this.builder = builder;  
    }  
  
    public static void main(String[] args) {  
  
        Main m = new Main(new BuilderItau());  
        m.builder.buildValorPagamento(1000.0);  
        System.out.println(m.builder.getBoleto(1));  
  
    }  
}
```

## Business Service



```
package business;  
  
public class BusinessService {  
  
    public void imprimir() {  
  
        Pessoa p = ControleThread.get();  
        System.out.println(p);  
    }  
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package business;

@SuppressWarnings("unchecked")
public class ControleThread {

    @SuppressWarnings("rawtypes")
    public static final ThreadLocal threadLocal
        = new ThreadLocal();

    public static void set(Pessoa p) {
        threadLocal.set(p);
    }

    public static void remove() {
        threadLocal.remove();
    }

    public static Pessoa get() {
        return (Pessoa) threadLocal.get();
    }

}

package business;

public class Pessoa {

    private Integer idPessoa;
    private String nome;

    public Pessoa() {
    }

    public Pessoa(Integer idPessoa, String nome) {
        super();
        this.idPessoa = idPessoa;
        this.nome = nome;
    }

    @Override
    public String toString() {
        return "Pessoa [idPessoa=" + idPessoa + ", nome="
            + nome + "]";
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro – RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

}

package business;

public class Main extends Thread {

    public static void main(String args[]) {

        Thread threadUm    = new Main();
        Thread threadDois = new Main();

        threadUm.start();
        threadDois.start();
    }

    @Override
    public void run() {

        for(int i = 1; i < 10; i++){

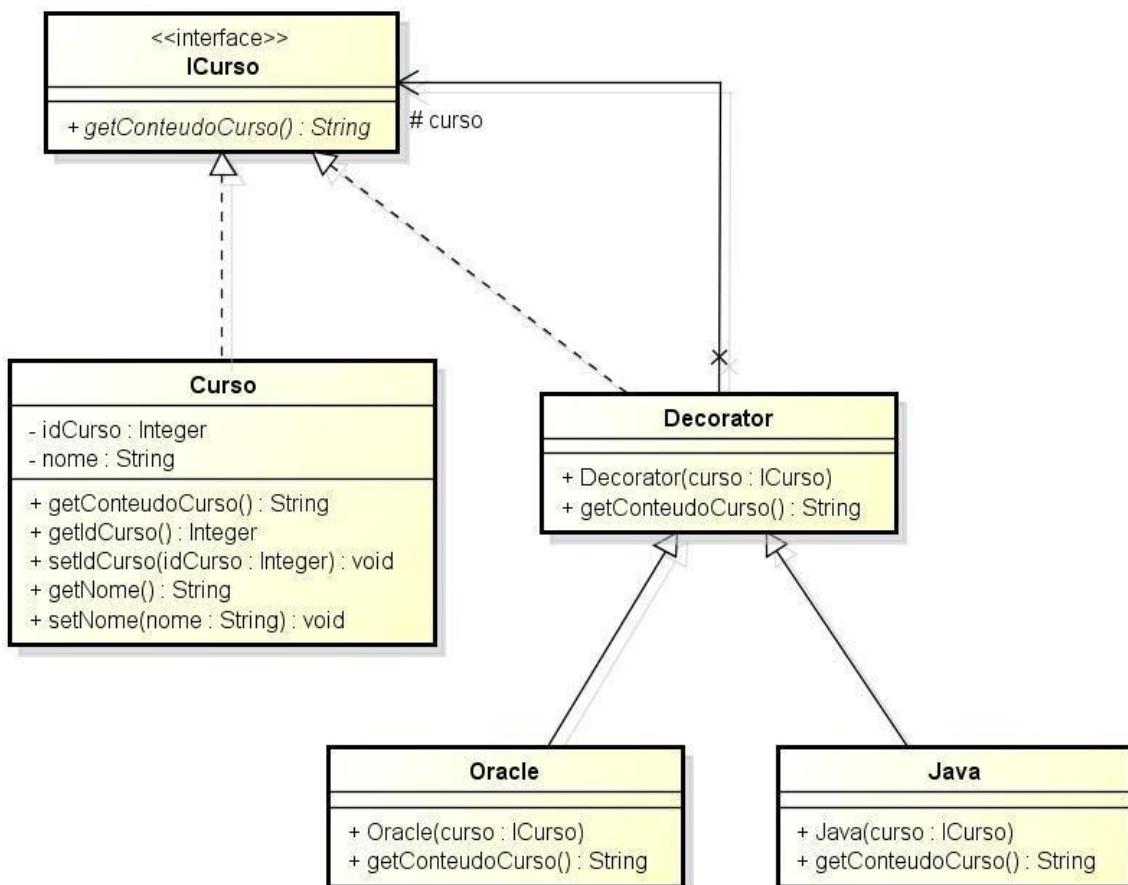
            Pessoa p = new Pessoa(i, "Belem -> " + i);
            ControleThread.set(p);

            new BusinessService().imprimir();
            ControleThread.remove();

        }
    }
}
```

## Decorator

O padrão de projeto (Design Pattern) Decorator tem como principal objetivo a decoração de classes em tempo de execução, isto é, adicionar novos produtos e/ou novas responsabilidades à objetos dinamicamente sem alterar o código das classes existentes.



```

package decorator;

public class Curso implements ICurso {

    private Integer idCurso;
    private String nome;

    @Override
    public String getConteudoCurso() {
        return idCurso + ", " + nome;
    }

    public Integer getIdCurso() {
        return idCurso;
    }

    public void setIdCurso(Integer idCurso) {
        this.idCurso = idCurso;
    }
}
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

}

package decorator;

public class Decorator implements ICurso {

    protected ICurso curso;

    public Decorator(ICurso curso) {
        this.curso = curso;
    }

    @Override
    public String getConteudoCurso() {
        return curso.getConteudoCurso();
    }
}

package decorator;

public interface ICurso {

    String getConteudoCurso();
}

package decorator;

public class Java extends Decorator {

    public Java(ICurso curso) {
        super(curso);
    }

    @Override
    public String getConteudoCurso() {
        return super.getConteudoCurso() + ", Java Orientado
               a Objetos";
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package decorator;

public class Oracle extends Decorator {

    public Oracle(ICurso curso) {
        super(curso);
    }

    @Override
    public String getConteudoCurso() {
        return super.getConteudoCurso() + ", Oracle DBA";
    }
}

package decorator;

public class Main {

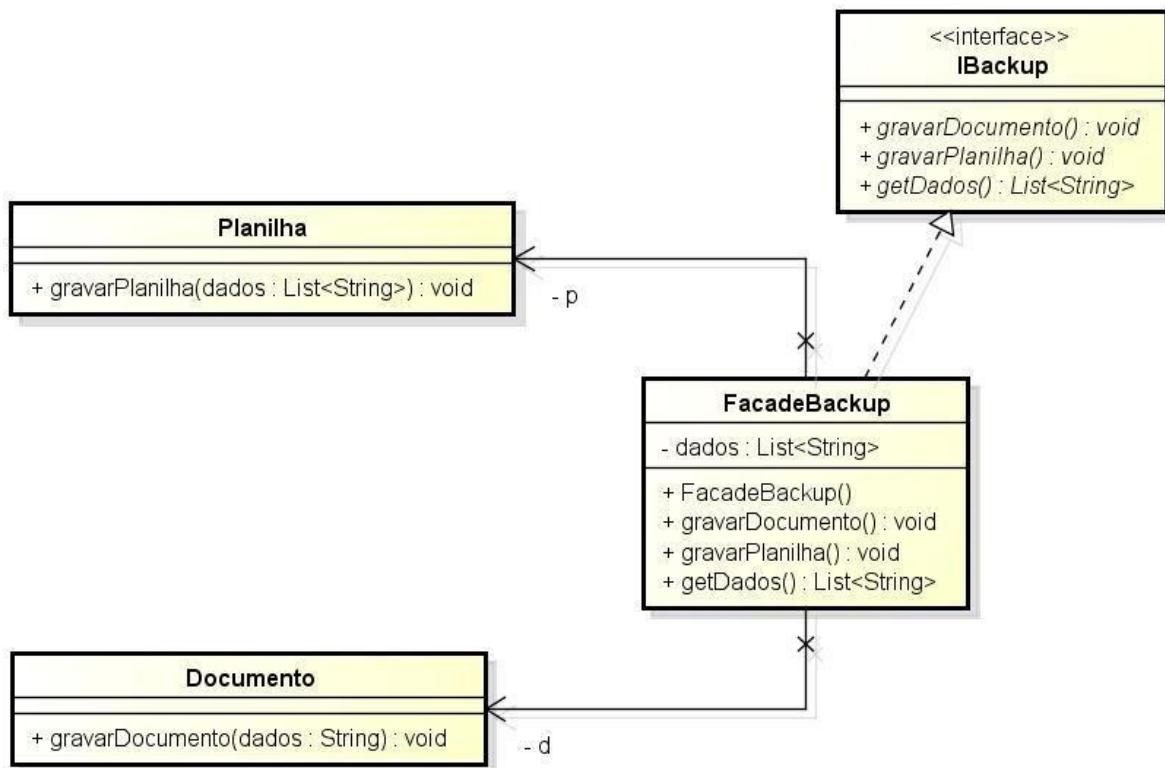
    public static void main(String[] args) {

        Curso c = new Curso();
        c.setIdCurso(1);
        c.setNome("Intensivo");

        ICurso curso = new Java(new Oracle(c));
        System.out.println(curso.getConteudoCurso());
    }
}
```

## Facade

Um façade (*fachada*) é um objeto que disponibiliza uma interface simplificada para uma das funcionalidades de uma API, por exemplo.



```

package facade;

import java.io.File;
import java.io.FileWriter;

public class Documento {

    public void gravarDocumento(String dados) throws Exception{
        FileWriter w = new FileWriter(new
File("d:\\\\facade.doc"), true);
        w.write(dados);
        w.close();
    }
}

package facade;

import java.util.ArrayList;
import java.util.List;
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
public class FacadeBackup implements IBackup {

    private List<String> dados;
    private Documento d;
    private Planilha p;

    public FacadeBackup() {
        dados = new ArrayList<String>();
        d = new Documento();
        p = new Planilha();
    }

    @Override
    public void gravarDocumento() throws Exception {
        d.gravarDocumento(dados.toString());
    }

    @Override
    public void gravarPlanilha() throws Exception {
        p.gravarPlanilha(dados);
    }

    public List<String> getDados() {
        return dados;
    }
}

package facade;

import java.util.List;

public interface IBackup {

    void gravarDocumento() throws Exception;
    void gravarPlanilha() throws Exception;
    List<String> getDados();
}

package facade;

import java.io.File;
import java.io.FileWriter;
import java.util.List;

public class Planilha {

    public void gravarPlanilha(List<String> dados)
        throws Exception {

```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
FileWriter w = new FileWriter(new
File("d:\\facade.csv"), true);

for (String linha : dados) {

    w.write(linha);
    w.write("\n");
}
w.close();
}

}

package facade;

public class Main {

    public static void main(String[] args) {

        IBackup b = new FacadeBackup();

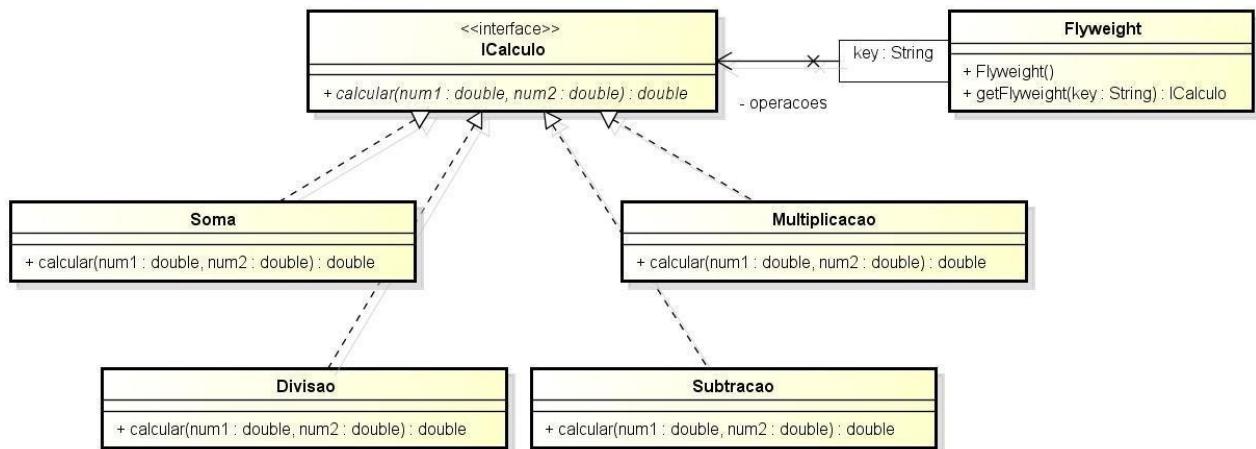
        b.getDados().add("Aula de Java");
        b.getDados().add("Design Patterns");
        b.getDados().add("Facade");

        try{
            b.gravarDocumento();
            b.gravarPlanilha();

            System.out.println("Backup realizado");
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```

## Flyweight

*Flyweight* é um padrão de projeto de software apropriado quando vários objetos devem ser manipulados, e esses não suportam dados adicionais. No padrão *flyweight* não existem ponteiros para os métodos do dado, pois isto consome muita memória. Em contrapartida são chamadas sub-rotinas diretamente para acessar o dado.



```

package flyweight;

public class Divisao implements ICalculo {

    @Override
    public double calcular(double num1, double num2) {
        try {
            return num1 / num2;
        }
        catch (ArithmaticException e) {
            e.printStackTrace();
            return 0.;
        }
    }
}
  
```

```

package flyweight;

import java.util.HashMap;
import java.util.Map;

public class Flyweight {

    private Map<String, ICalculo> operacoes;

    public Flyweight() {
        operacoes = new HashMap<String, ICalculo>();
    }
}
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
public ICalculo getFlyweight(String key) {  
  
    if ( ! operacoes.containsKey(key) ) {  
  
        if ("soma".equals(key)) {  
            operacoes.put(key, new Soma());  
        }  
        else if("sub".equals(key)){  
            operacoes.put(key, new Subtracao());  
        }  
        else if("mult".equals(key)){  
            operacoes.put(key, new Multiplicacao());  
        }  
        else if("div".equals(key)){  
            operacoes.put(key, new Divisao());  
        }  
    }  
  
    return operacoes.get(key);  
}  
  
}  
  
package flyweight;  
  
public interface ICalculo {  
  
    double calcular(double num1, double num2);  
}  
  
}  
  
package flyweight;  
  
public class Multiplicacao implements ICalculo {  
  
    @Override  
    public double calcular(double num1, double num2) {  
        return num1 * num2;  
    }  
}  
  
}  
  
package flyweight;  
  
public class Soma implements ICalculo {  
  
    @Override  
    public double calcular(double num1, double num2) {  
        return num1 + num2;  
    }  
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package flyweight;

public class Subtracao implements ICalculo {

    @Override
    public double calcular(double num1, double num2) {
        return num1 - num2;
    }
}

package flyweight;

public class Main {

    public static void main(String[] args) {

        Flyweight f = new Flyweight();

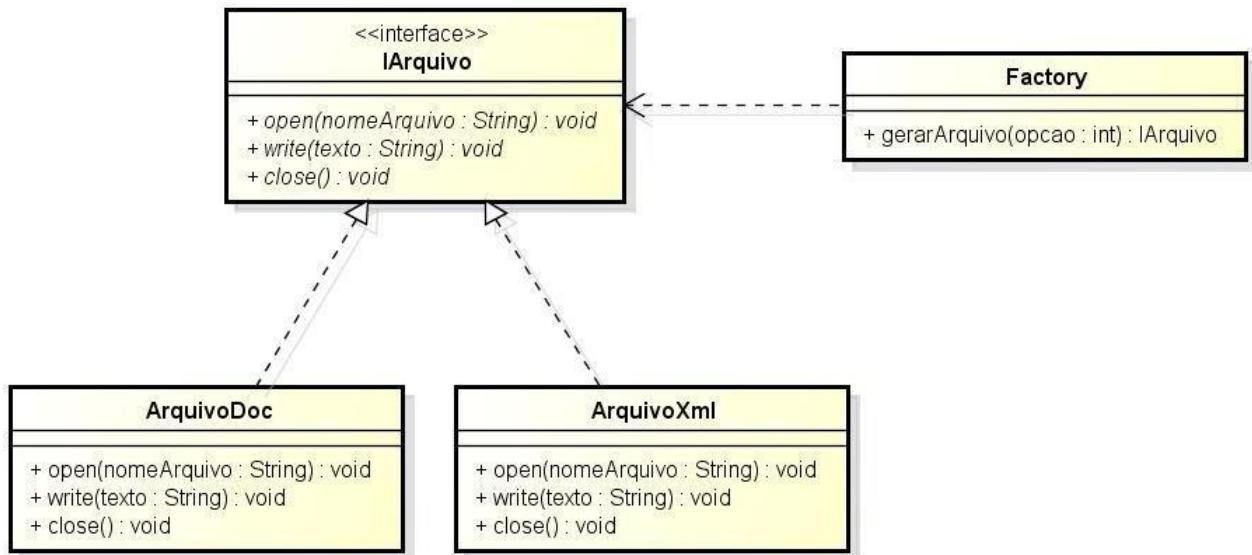
        for (int i = 0; i < 5; i++) {

            ICalculo soma = f.getFlyweight("soma");
            System.out.println("Somando: " + soma.calcular(i,
i));

            ICalculo multiplicacao = f.getFlyweight("mult");
            System.out.println("Multiplicando: " +
multiplicacao.calcular(i, i));
        }
    }
}
```

## Factory Method

Define uma interface para criação de um objeto, mas permite que as subclasses escolham quais classes instanciar. O factory method permite delegar a instanciação para as subclasses.



```

package factory;

import java.io.File;
import java.io.FileWriter;

public class ArquivoDoc implements IArquivo {

    private FileWriter fw;

    @Override
    public void open(String nomeArquivo) throws Exception {
        fw = new FileWriter(new File("d:\\\" + nomeArquivo
            + ".doc"));
    }

    @Override
    public void write(String texto) throws Exception {
        fw.write(texto);
    }

    @Override
    public void close() throws Exception {
        fw.close();
    }
}
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package factory;

public class Factory {

    public IArquivo gerarArquivo(int opcao) {

        switch (opcao) {
        case 1:
            return new ArquivoDoc();
        case 2:
            return new ArquivoXml();
        }

        return null;
    }
}

package factory;

public interface IArquivo {

    void open(String nomeArquivo) throws Exception;
    void write(String texto) throws Exception;
    void close() throws Exception;
}

package factory;

public class Main {

    public static void main(String[] args) {

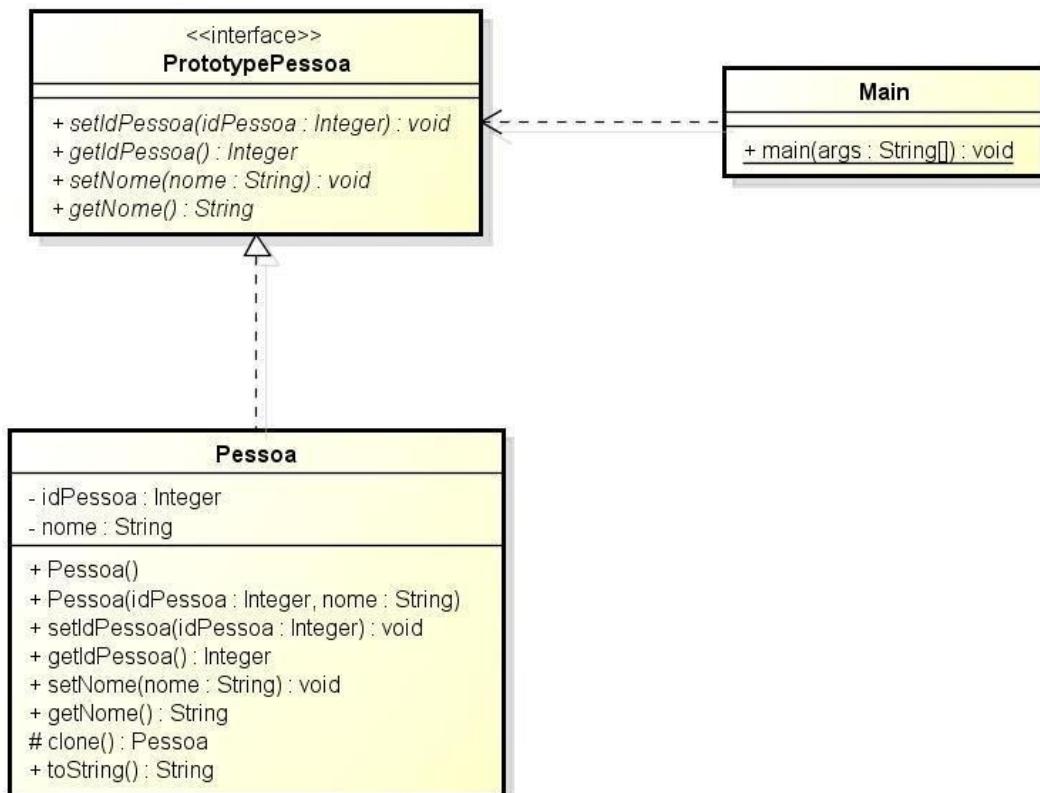
        IArquivo a = new Factory().gerarArquivo(2);

        try{
            a.open("dados");
            a.write("Aula de Design Pattern");
            a.close();

            System.out.println("Dados gravados com sucesso");
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```

## Prototype

É um padrão de projeto de software (*design pattern*, em inglês) que permite a criação de objetos a partir de um modelo original, ou protótipo.



```

package prototype;

public class Pessoa implements PrototypePessoa, Cloneable {

    private Integer idPessoa;
    private String nome;

    public Pessoa() {
    }

    public Pessoa(Integer idPessoa, String nome) {
        super();
        this.idPessoa = idPessoa;
        this.nome = nome;
    }

    @Override
    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }
}
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
@Override
public Integer getIdPessoa() {
    return idPessoa;
}

@Override
public void setNome(String nome) {
    this.nome = nome;
}

@Override
public String getNome() {
    return nome;
}

@Override
protected Pessoa clone() throws CloneNotSupportedException {
    return (Pessoa) super.clone();
}

@Override
public String toString() {
    return "Pessoa [idPessoa=" + idPessoa + ", nome="
           + nome + "]";
}

}

package prototype;

public interface PrototypePessoa {

    void setIdPessoa(Integer idPessoa);
    Integer getIdPessoa();

    void setNome(String nome);
    String getNome();

}

package prototype;

public class Main {

    public static void main(String[] args) {

        try {

            Pessoa p = new Pessoa();

            for (int i = 1; i <= 5; i++) {
```



# **Java Design Patterns – Apostila**

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
        PrototypePessoa prototype = p.clone();

        prototype.setIdPessoa(i);
        prototype.setNome("Belem " + i);

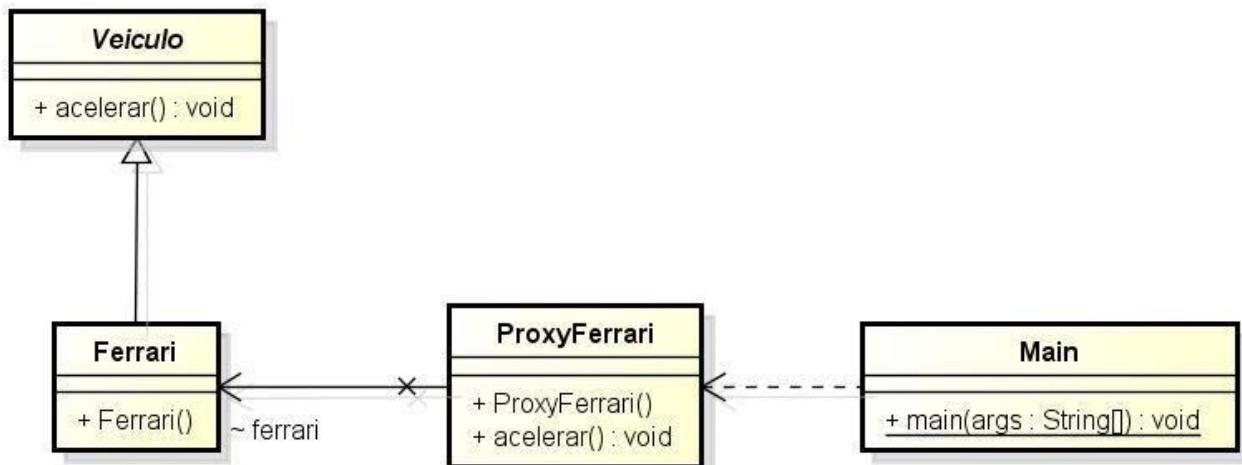
        System.out.println(prototype);
    }

} catch (Exception e) {
    System.out.println(e.getMessage());
}

}
```

# Proxy

Um proxy, em sua forma mais geral, é uma classe que funciona como uma interface para outra classe. A classe proxy poderia conectar-se a qualquer coisa: uma conexão de rede, um objeto grande em memória, um arquivo, ou algum recurso que é difícil ou impossível de ser duplicado.



```
package proxy;

public class Ferrari extends Veiculo {

    public Ferrari() {
        try {
            Thread.sleep(5000);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package proxy;

import java.text.SimpleDateFormat;
import java.util.Date;

public class ProxyFerrari {

    Ferrari ferrari;

    public ProxyFerrari() {

        String tempo = new SimpleDateFormat("HH:mm:ss") .
            format(new Date());

        System.out.println("Ferrari parada em: " + tempo);
    }

    public void acelerar(){

        if(ferrari == null){
            ferrari = new Ferrari();
        }

        ferrari.acelerar();
    }
}

package proxy;

import java.text.SimpleDateFormat;
import java.util.Date;

public abstract class Veiculo {

    public void acelerar(){

        String tempo = new SimpleDateFormat("HH:mm:ss") .
            format(new Date());

        System.out.println(this.getClass().getSimpleName() +
            ", chegou a 100km/h em " + tempo);
    }
}

package proxy;

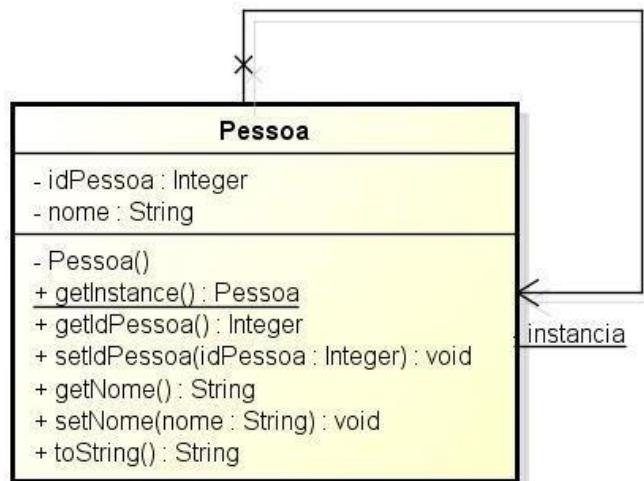
public class Main {

    public static void main(String[] args) {

        ProxyFerrari proxy = new ProxyFerrari();
        proxy.acelerar();
    }
}
```

## Singleton

Este padrão garante a existência de apenas uma instância de uma classe, mantendo um ponto global de acesso ao seu objeto.



```

package singleton;

public final class Pessoa {

    private Integer idPessoa;
    private String nome;
    private static Pessoa instancia;

    private Pessoa() {

    }

    public static Pessoa getInstance() {
        if (instancia == null)
            instancia = new Pessoa();
        return instancia;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
  
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
@Override
public String toString() {
    return "Pessoa [idPessoa=" + idPessoa + ", nome="
           + nome + "]";
}

}

package singleton;

public class Main {

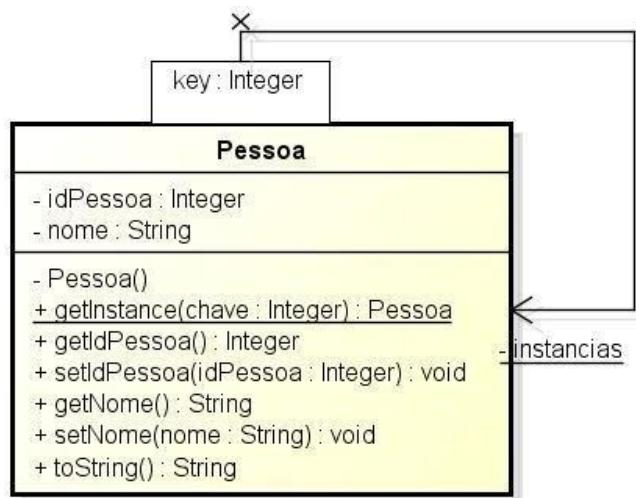
    public static void main(String[] args) {

        Pessoa p1 = Pessoa.getInstance();
        Pessoa p2 = Pessoa.getInstance();

        p1.setIdPessoa(1);
        p1.setNome("Belem");

        p2.setIdPessoa(2);
        p2.setNome("NetCat");

        System.out.println(p1);
        System.out.println(p2);
    }
}
```



```
package singletonmapa;

import java.util.HashMap;
import java.util.Map;

public class Pessoa {
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
private static final Map<Integer, Pessoa> instancias
    = new HashMap<>();

private Integer idPessoa;
private String nome;

private Pessoa() {
}

public static synchronized Pessoa getInstance(Integer chave)
{
    Pessoa instancia = instancias.get(chave);

    if (instancia == null) {
        instancia = new Pessoa();

        instancias.put(chave, instancia);
    }

    return instancia;
}

public Integer getIdPessoa() {
    return idPessoa;
}

public void setIdPessoa(Integer idPessoa) {
    this.idPessoa = idPessoa;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

@Override
public String toString() {
    String saida = "Pessoas: ";
    for (Pessoa p : instancias.values()) {
        saida += "\n" + p.getIdPessoa() + ", " +
p.getNome();
    }

    return saida;
}
}

package singletonmapa;

public class Main {
```



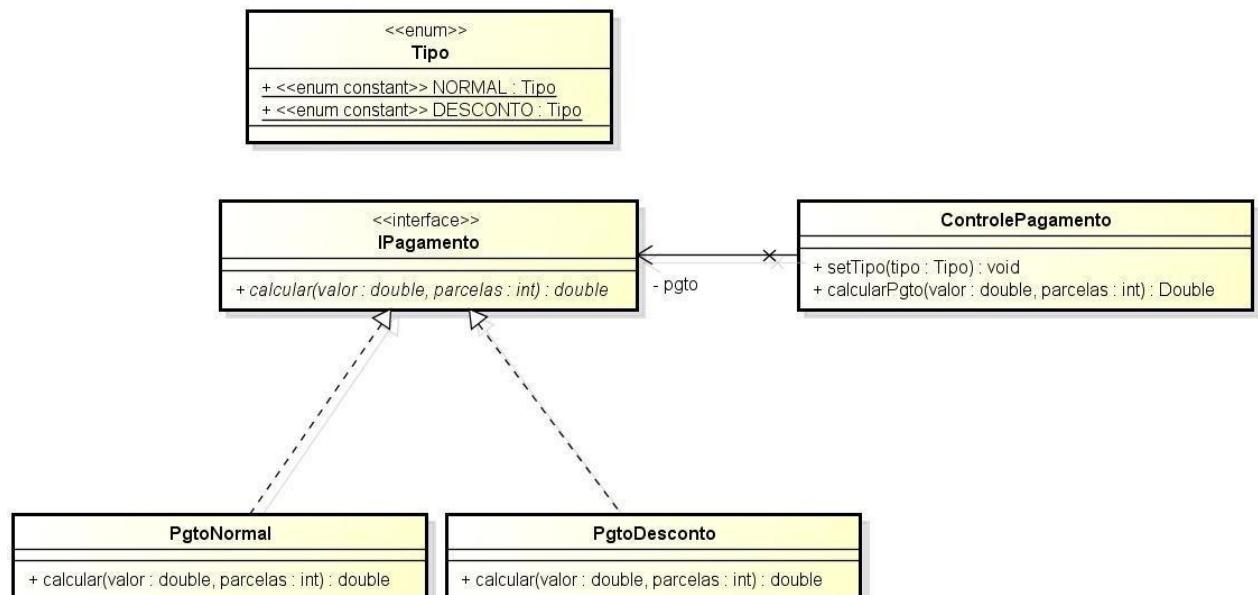
## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 www.cotiinformatica.com.br

```
public static void main(String[] args) {  
  
    Pessoa p1 = Pessoa.getInstance(1);  
    p1.setIdPessoa(1);  
    p1.setNome("Sergio");  
  
    Pessoa p2 = Pessoa.getInstance(2);  
    p2.setIdPessoa(2);  
    p2.setNome("Pedro");  
  
    Pessoa p3 = Pessoa.getInstance(2);  
    p3.setIdPessoa(3);  
    p3.setNome("Belem");  
  
    System.out.println(p1);  
    System.out.println(p2);  
}  
  
}
```

## State

*State* é um padrão de projeto de software usado quando o comportamento de um objeto muda, dependendo do seu estado.



```
package state;  
  
public class ControlePagamento {  
  
    private IPagamento pgto;  
  
    public void setTipo(Tipo tipo) {  
  
        switch (tipo) {
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro – RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
case NORMAL:  
    pgto = new PgtoNormal();  
    break;  
  
case DESCONTO:  
    pgto = new PgtoDesconto();  
    break;  
}  
}  
  
public Double calcularPgto(double valor, int parcelas) {  
    return pgto.calcular(valor, parcelas);  
}  
}  
  
  
package state;  
  
public interface IPagamento {  
  
    double calcular(double valor, int parcelas);  
}  
  
  
package state;  
  
public class PgtoDesconto implements IPagamento {  
  
    @Override  
    public double calcular(double valor, int parcelas) {  
        return parcelas * (valor * 0.9);  
    }  
}  
  
  
package state;  
  
public class PgtoNormal implements IPagamento {  
  
    @Override  
    public double calcular(double valor, int parcelas) {  
        return parcelas * valor;  
    }  
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 www.cotiinformatica.com.br

```
package state;

public enum Tipo {
    NORMAL, DESCONTO
}

package state;

public class Main {

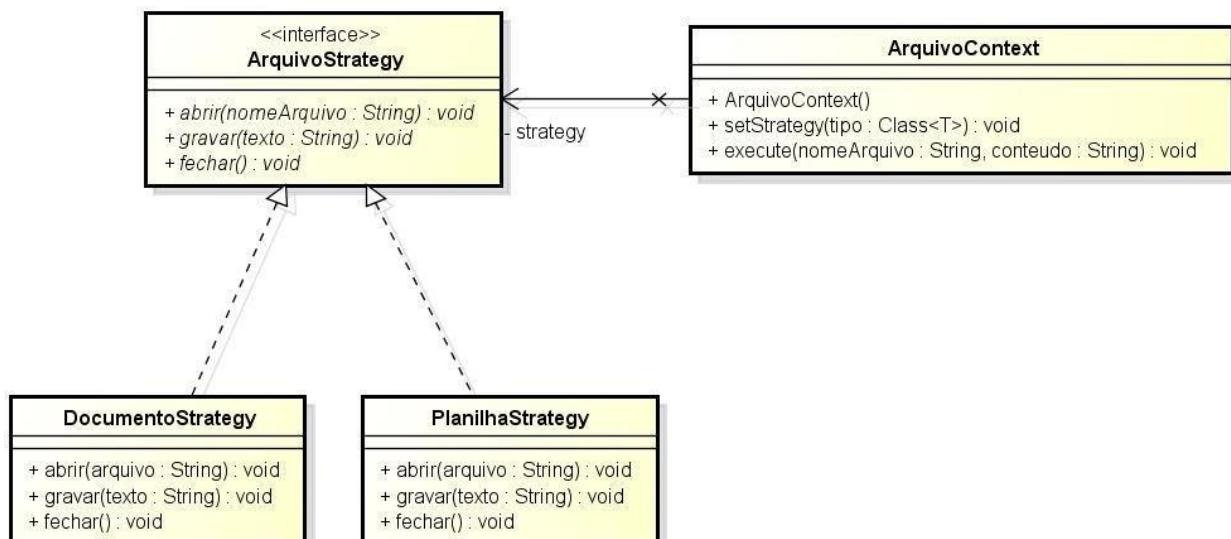
    public static void main(String[] args) {
        ControlePagamento c = new ControlePagamento();

        c.setTipo(Tipo.NORMAL);
        System.out.println("Pgto: " + c.calcularPgto(100., 10));

        c.setTipo(Tipo.DESCONTO);
        System.out.println("Pgto: " + c.calcularPgto(100., 10));
    }
}
```

## Strategy

O objetivo é representar uma operação a ser realizada sobre os elementos de uma estrutura de objetos. O padrão Strategy permite definir novas operações sem alterar as classes dos elementos sobre os quais opera. Definir uma família de algoritmos e encapsular cada algoritmo como uma classe, permitindo assim que elas possam ter trocados entre si. Este padrão permite que o algoritmo possa variar independentemente dos clientes que o utilizam.





## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package strategy;

public class ArquivoContext {

    private ArquivoStrategy strategy;

    public ArquivoContext() {

    }

    public <T extends ArquivoStrategy> void setStrategy
        (Class<T> tipo) {
        try {
            this.strategy = tipo.newInstance();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void execute(String nomeArquivo, String conteudo)
        throws Exception {

        strategy.abrir(nomeArquivo);
        strategy.gravar(conteudo);
        strategy.fechar();

    }
}
```

```
package strategy;

public interface ArquivoStrategy {

    void abrir(String nomeArquivo) throws Exception;
    void gravar(String texto) throws Exception;
    void fechar() throws Exception;
}
```

```
package strategy;

import java.io.File;
import java.io.FileWriter;

public class DocumentoStrategy implements ArquivoStrategy {

    private FileWriter writer;

    @Override
    public void abrir(String arquivo) throws Exception {
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
writer = new FileWriter(new File("d:\\" + arquivo
        + ".doc"));
}

@Override
public void gravar(String texto) throws Exception {
    writer.write(texto + "\n");
}

@Override
public void fechar() throws Exception {
    writer.close();
}
}

package strategy;

import java.io.File;
import java.io.FileWriter;

public class PlanilhaStrategy implements ArquivoStrategy {

    private FileWriter writer;

    @Override
    public void abrir(String arquivo) throws Exception {
        writer = new FileWriter(new File("d:\\" + arquivo
            + ".csv"));
    }

    @Override
    public void gravar(String texto) throws Exception {
        writer.write(texto + "\n");
    }

    @Override
    public void fechar() throws Exception {
        writer.close();
    }
}

package strategy;

public class Main {

    public static void main(String[] args) {
        ArquivoContext c = new ArquivoContext();
        c.setStrategy(DocumentStrategy.class);

        try {
            c.execute("aula", "testando design patterns");
        }
    }
}
```



# **Java Design Patterns – Apostila**

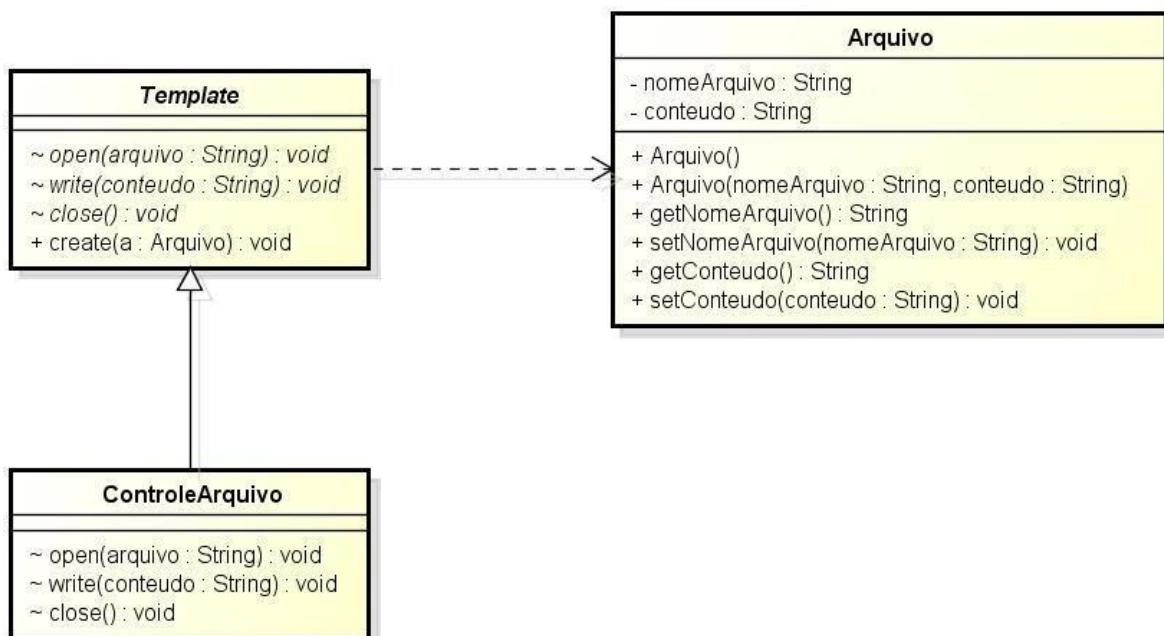
COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
        System.out.println("Dados gravados.");

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
```

# Template Method

Um Template Method auxilia na definição de um algoritmo com partes do mesmo definidos por Método abstratos. As subclasses devem se responsabilizar por estas partes abstratas, deste algoritmo, que serão implementadas, possivelmente de várias formas, ou seja, cada subclass irá implementar à sua necessidade e oferecer um comportamento concreto construindo todo o algoritmo.



```
package template;

public class Arquivo {

    private String nomeArquivo;
    private String conteudo;

    public Arquivo() {
    }

    public Arquivo(String nomeArquivo, String conteudo) {
        super();
        this.nomeArquivo = nomeArquivo;
        this.conteudo = conteudo;
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
public String getNomeArquivo() {
    return nomeArquivo;
}

public void setNomeArquivo(String nomeArquivo) {
    this.nomeArquivo = nomeArquivo;
}

public String getConteudo() {
    return conteudo;
}

public void setConteudo(String conteudo) {
    this.conteudo = conteudo;
}
}

package template;

import java.io.File;
import java.io.FileWriter;

public class ControleArquivo extends Template {

    private FileWriter fw;

    @Override
    void open(String arquivo) throws Exception {
        fw = new FileWriter(new File("d://" + arquivo), true);
    }

    @Override
    void write(String conteudo) throws Exception {
        fw.write(conteudo);
    }

    @Override
    void close() throws Exception {
        fw.close();
    }
}

package template;

public abstract class Template {

    abstract void open(String arquivo) throws Exception;
    abstract void write(String conteudo) throws Exception;
    abstract void close() throws Exception;

    public final void create(Arquivo a) throws Exception {
        open(a.getNomeArquivo());
        write(a.getConteudo());
        close();
    }
}
```



# Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro – RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package template;

public class Main {

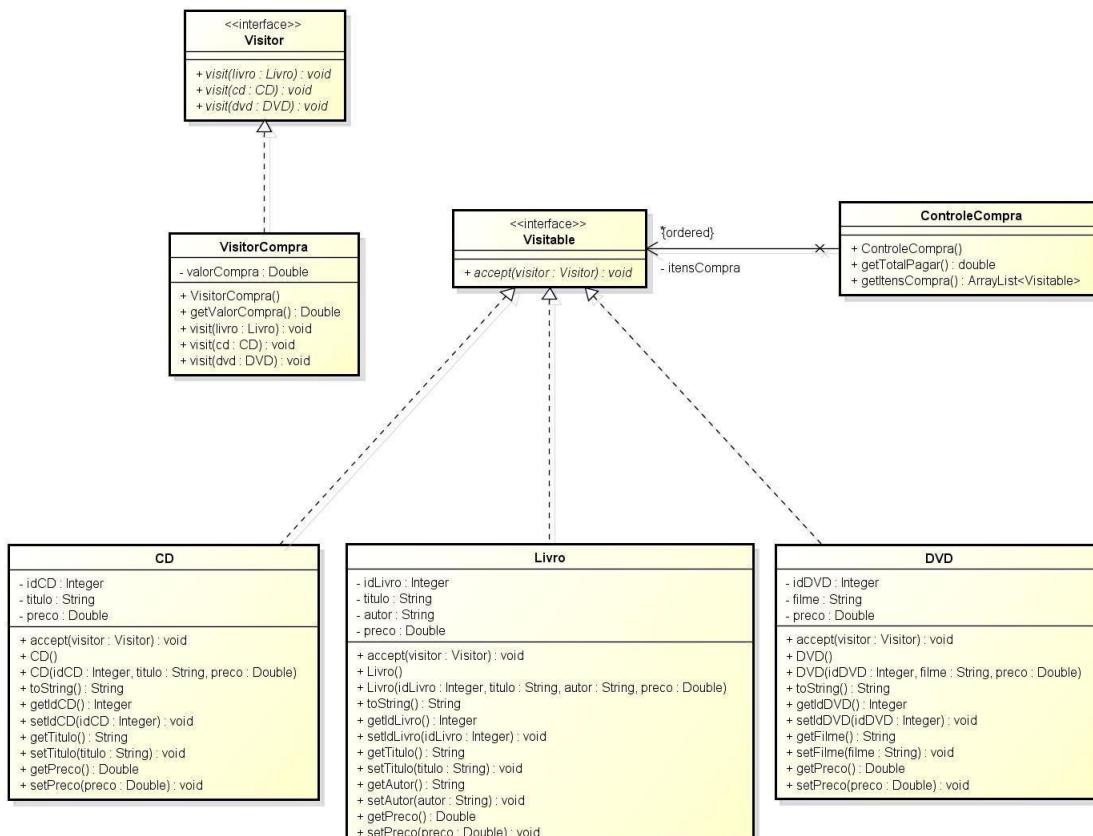
    public static void main(String[] args) {

        Arquivo a = new Arquivo("texto.txt", "Aula de Java");
        ControleArquivo c = new ControleArquivo();

        try{
            c.create(a);
            System.out.println("Dados gravados");
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```

## Visitor

Em programação orientada a objetos e engenharia de software, o *visitor pattern* é um padrão de projeto comportamental. Representa uma operação a ser realizada sobre elementos da estrutura de um objeto. O Visitor permite que se crie uma nova operação sem que se mude a classe dos elementos sobre as quais ela opera. É uma maneira de separar um algoritmo da estrutura de um objeto. Um resultado prático é a habilidade de adicionar novas funcionalidades a estruturas de um objeto pré-existente sem a necessidade de modificá-las.





## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package visitor;

public class CD implements Visitable {

    private Integer idCD;
    private String titulo;
    private Double preco;

    @Override
    public void accept(Visitor visitor) {
        visitor.visit(this);
    }

    public CD() {
    }

    public CD(Integer idCD, String titulo, Double preco) {
        super();
        this.idCD = idCD;
        this.titulo = titulo;
        this.preco = preco;
    }

    @Override
    public String toString() {
        return "CD [idCD=" + idCD + ", titulo=" + titulo
               + ", preco=" + preco + "]";
    }

    public Integer getIdCD() {
        return idCD;
    }

    public void setIdCD(Integer idCD) {
        this.idCD = idCD;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public Double getPreco() {
        return preco;
    }

    public void setPreco(Double preco) {
        this.preco = preco;
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro – RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package visitor;

import java.util.ArrayList;

public class ControleCompra {

    private ArrayList<Visitable> itensCompra;

    public ControleCompra() {
        itensCompra = new ArrayList<Visitable>();
    }

    public double getTotalPagar() {
        VisitorCompra visitor = new VisitorCompra();

        for (Visitable item : itensCompra) {
            item.accept(visitor);
        }

        return visitor.getValorCompra();
    }

    public ArrayList<Visitable> getItensCompra() {
        return itensCompra;
    }
}

package visitor;

public class DVD implements Visitable {

    private Integer idDVD;
    private String filme;
    private Double preco;

    @Override
    public void accept(Visitor visitor) {
        visitor.visit(this);
    }

    public DVD() {
    }

    public DVD(Integer idDVD, String filme, Double preco) {
        super();
        this.idDVD = idDVD;
        this.filme = filme;
        this.preco = preco;
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
@Override
public String toString() {
    return "DVD [idDVD=" + idDVD + ", filme=" + filme
           + ", preco=" + preco + "]";
}

public Integer getIdDVD() {
    return idDVD;
}

public void setIdDVD(Integer idDVD) {
    this.idDVD = idDVD;
}

public String getFilme() {
    return filme;
}

public void setFilme(String filme) {
    this.filme = filme;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

}

package visitor;

public class Livro implements Visitable {

    private Integer idLivro;
    private String titulo;
    private String autor;
    private Double preco;

    @Override
    public void accept(Visitor visitor) {
        visitor.visit(this);
    }

    public Livro() {
    }

    public Livro(Integer idLivro, String titulo, String autor,
                Double preco) {
        super();
        this.idLivro = idLivro;
        this.titulo = titulo;
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
        this.autor = autor;
        this.preco = preco;
    }

@Override
public String toString() {
    return "Livro [idLivro=" + idLivro + ", titulo="
           + titulo + ", autor=" + autor + ", preco=" + preco
           + "]";
}

public Integer getIdLivro() {
    return idLivro;
}

public void setIdLivro(Integer idLivro) {
    this.idLivro = idLivro;
}

public String getTitulo() {
    return titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public String getAutor() {
    return autor;
}

public void setAutor(String autor) {
    this.autor = autor;
}

public Double getPreco() {
    return preco;
}

public void setPreco(Double preco) {
    this.preco = preco;
}

}

package visitor;

public interface Visitable {

    public void accept(Visitor visitor);

}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package visitor;

public interface Visitor {

    public void visit(Livro livro);

    public void visit(CD cd);

    public void visit(DVD dvd);
}

package visitor;

public class VisitorCompra implements Visitor {

    private transient Double valorCompra;

    public VisitorCompra() {
        valorCompra = 0.;
    }

    public Double getValorCompra() {
        return valorCompra;
    }

    @Override
    public void visit(Livro livro) {
        // Livro tem desconto de 10%
        valorCompra += livro.getPreco() - (livro.getPreco()
            * 0.10);
    }

    @Override
    public void visit(CD cd) {
        // CD tem desconto de 5%
        valorCompra += cd.getPreco() - (cd.getPreco() * 0.05);
    }

    @Override
    public void visit(DVD dvd) {
        // Filme não tem desconto
        valorCompra += dvd.getPreco();
    }
}
```



## Java Design Patterns – Apostila

COTI Informática - Av. Rio Branco, 185 - Sala 904 - Rio de Janeiro - RJ  
Tel. 21-2262-9043 [www.cotiinformatica.com.br](http://www.cotiinformatica.com.br)

```
package visitor;

public class Main {

    public static void main(String[] args) {
        ControleCompra c = new ControleCompra();

        c.getItensCompra().add(new Livro(1, "Senhor dos Aneis",
                                         "Tolkien", 100.0));

        c.getItensCompra().add(new Livro(2, "Codigo daVinci",
                                         "Dan Brown", 50.0));

        c.getItensCompra().add(new CD(3, "Legião Urbana", 40.0));

        c.getItensCompra().add(new DVD(4, "Se Beber não case",
                                       20.0));

        c.getItensCompra().add(new CD(5, "Depois da Terra", 30.0));

        System.out.println("Itens: " + c.getItensCompra());
        System.out.println("Total a pagar: " + c.getTotalPagar());
    }
}
```