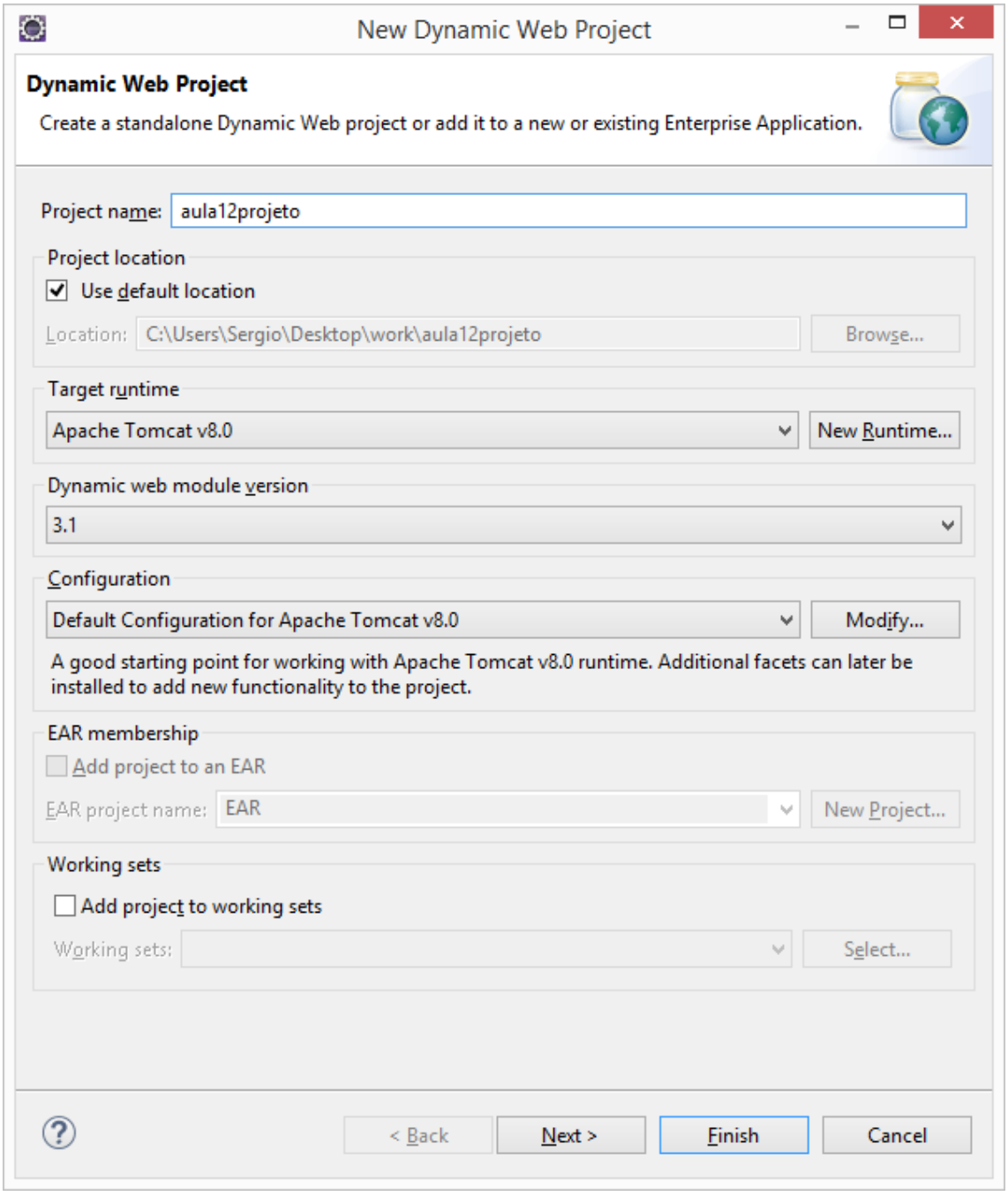
Criando o projeto:

- File > New > Dynamic Web Project



Hibernate e JPA

O objetivo do Hibernate é diminuir a complexidade entre os programas Java, baseado no modelo orientado a objeto, que precisam trabalhar com um banco de dados do modelo relacional (presente na maioria dos SGBDs). Em especial, no desenvolvimento de consultas e atualizações dos dados.

Sua principal característica é a transformação das classes em Java para tabelas de dados (e dos tipos de dados Java para os da SQL). O Hibernate gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL

A JPA – Java Persistence API

JPA é um framework leve, baseado em POJOS (Plain Old Java Objects) para persistir objetos Java. A Java Persistence API, diferente do que muitos imaginam, não é apenas um framework para Mapeamento Objeto-Relacional (ORM - Object-Relational Mapping), ela também oferece diversas funcionalidades essenciais em qualquer aplicação corporativa.

- Fazendo o ORM (Mapeamento Objeto Relacional) da entidade Funcionario:

```
package br.com.brq.entities;

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;

//ORM - Mapeamento Objeto Relacional
//JPA - Java Persistence API (Biblioteca para mapeamento)
```

```
@Entity //classe é uma entidade mapeada pela JPA
@Table(name = "funcionario") //nome da tabela..
@NamedQueries( //mapear as consultas HQL da entidade..
		{
			//toda consulta HQL sempre faz select na Classe e não na tabela
			@NamedQuery(name = "funcionario.findall",
				//nome da consulta..
				query = "select f from Funcionario as f"),
				//consulta HQL
			@NamedQuery(name = "funcionario.findbydata",
				query = "select f from Funcionario as f where
					f.dataAdmissao between :p1 and :p2")
		}
)
public class Funcionario {

	@Id //chave primária..
	@GeneratedValue(strategy=GenerationType.AUTO) //auto_incremento..
	@Column(name = "idfuncionario")
	private Integer idFuncionario;

	@Column(name = "nome", length = 50, nullable = false)
	private String nome;

	@Column(name = "salario", nullable = false)
	private Double salario;

	@Temporal(TemporalType.DATE) //somente data..
	@Column(name = "dataadmissao", nullable = false)
	private Date dataAdmissao;

	public Funcionario() {
		// TODO Auto-generated constructor stub
	}

	public Funcionario(Integer idFuncionario, String nome, Double salario,
				Date dataAdmissao) {
		super();
		this.idFuncionario = idFuncionario;
		this.nome = nome;
		this.salario = salario;
		this.dataAdmissao = dataAdmissao;
	}
```

```
	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;
	}

	@Override
	public String toString() {
		return "Funcionario [idFuncionario=" + idFuncionario
			+ ", nome=" + nome + ", salario=" + salario
			+ ", dataAdmissao=" + dataAdmissao + "]";
	}

}
```

HQL – Hibernate Query Language

A HQL (Hibernate Query Language) é um dialeto SQL para o Hibernate. Ela é uma poderosa linguagem de consulta que se parece muito com a SQL, mas a HQL é totalmente orientada a objeto, incluindo os paradigmas de herança, polimorfismo e encapsulamento.

No Hibernate, você pode escolher tanto usar a SQL quanto a HQL. Escolhendo a HQL, você poderá executar os pedidos SQL sobre as classes de persistência do Java ao invés de tabelas no banco de dados.

Utilizando o HQL temos a vantagem de portabilidade de banco, ou seja, suponha que estamos utilizando um banco de dados A, ao trocarmos para um banco B o HQL automaticamente cria comandos referentes a cada banco de dados. Isso facilita, pois no SQL teríamos que rastreara alterar vários códigos no sistema.

Configurando o acesso a base de dados:
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>

       <!-- Configurações para o Hibernate acessar 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/aula12</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">brqbrq</property>

    <!-- Configuração para que sempre que o hibernate execute uma instrução SQL
         no banco de dados, ele imprima o codigo SQL no console do eclipse -->
    <property name="hibernate.show_sql">true</property>

    <!-- Configuração para que o codigo SQL exibido fique formatado -->
    <property name="hibernate.format_sql">true</property>

    <!-- Incluir cada classe mapeada pela JPA -->
    <mapping class="br.com.brq.entities.Funcionario"/>

 </session-factory>
</hibernate-configuration>
```

Gerando as tabelas no banco de dados:

```
package br.com.brq.util;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.cfg.Configuration;
import org.hibernate.tool.hbm2ddl.SchemaExport;

//classe para gerar as tabelas no banco de dados pelo hibernate..
public class GenerateTables {

	public static void main(String[] args) {

		try{
			//ler o arquivo de configuração do hibernate..
			Configuration cfg = new AnnotationConfiguration();
			cfg.configure("br/com/brq/config/mysql_hibernate.cfg.xml");

			//exportar (criar as tabelas)..
			SchemaExport s = new SchemaExport(cfg);
			s.create(true, true); //executando..
		}
		catch(Exception e){
			System.out.println("Erro: " + e.getMessage());
		}
	}
}
```

Criando a classe para gerar a conexão com banco de dados através do Hibernate (SessionFactory)

- HibernateUtil

```
package br.com.brq.persistence;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

//classe simples com a finalidade de ler o arquivo de configuração
//config.xml e produzir conexões com a base de dados
public class HibernateUtil {

	private static final SessionFactory sessionFactory; //conexão..
```

```
	static {
		try {
			sessionFactory = new AnnotationConfiguration()

	.configure("br/com/brq/config/mysql_hibernate.cfg.xml")
					.buildSessionFactory();
		} catch (Throwable ex) {
			throw new ExceptionInInitializerError(ex);
		}
	}

	public static SessionFactory getSessionFactory() {
		return sessionFactory;
	}
}
```

Criando a classe de persistência para Funcionario:
DAOFuncionario.java

```
package br.com.brq.persistence;

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

import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

import br.com.brq.entities.Funcionario;

public class DAOFuncionario {

	// atributos..
	private Session session; // utilizado para armazenar a conexão..
	private Transaction transaction; // commit ou rollback (transações)
	private Query query; // executar consultas
	// HQL - Hibernate Query Language (sintaxe para consultas..)

	// método para inserir um cliente na base de dados..
	public void insert(Funcionario f) throws Exception {
		// abrir conexão..
		session = HibernateUtil.getSessionFactory().openSession();
```

```
            transaction = session.beginTransaction(); // abrir transação..
            session.save(f); // inserindo..
            transaction.commit(); // executando..

            // fechar a conexão..
            session.close();
    }

    // método para atualizar um cliente na base de dados..
    public void update(Funcionario f) throws Exception {
            // abrir conexão..
            session = HibernateUtil.getSessionFactory().openSession();

            transaction = session.beginTransaction(); // abrir transação..
            session.update(f); // atualizando..
            transaction.commit(); // executando..

            // fechar a conexão..
            session.close();
    }

    // método para excluir um cliente na base de dados..
    public void delete(Funcionario f) throws Exception {
            // abrir conexão..
            session = HibernateUtil.getSessionFactory().openSession();

            transaction = session.beginTransaction(); // abrir transação..
            session.delete(f); // excluindo..
            transaction.commit(); // executando..

            // fechar a conexão..
            session.close();
    }

    //método para buscar 1 cliente pelo id..
    public Funcionario findById(Integer idFuncionario) throws Exception{
            //abrir conexão..
            session = HibernateUtil.getSessionFactory().openSession();

            //método para buscar 1 objeto pela chave..
            Funcionario f = (Funcionario) session.get(Funcionario.class,
                                                idFuncionario);
```

```
            session.close();
            return f; //retornar o objeto..
        }


        //método para executar a consulta findall..
        public List<Funcionario> findAll() throws Exception{
            //abrir conexão..
            session = HibernateUtil.getSessionFactory().openSession();

            query = session.getNamedQuery("funcionario.findall");
            //nome da consulta..
            List<Funcionario> lista = query.list();

            session.close(); //fechar a conexão..
            return lista; //retornando a lista.
        }

        //método para executar a consulta findbydata..
        public List<Funcionario> findByData(Date dataIni, Date dataFim)
        throws Exception{
            //abrir conexão..
            session = HibernateUtil.getSessionFactory().openSession();

            query = session.getNamedQuery("funcionario.findbydata");
            query.setDate("p1", dataIni); //passagem de parametro
            query.setDate("p2", dataFim); //passagem de parametro

            List<Funcionario> lista = query.list(); //executando..

            session.close(); //fechar conexão..
            return lista; //retornando a lista..
        }
}
```

Classe utilitaria para conversão da data resgatada na página web:

```
package br.com.brq.util;

import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
```

```
public class FormatacaoData {

	//método para converter uma string yyyy-MM-dd em Date..
	public static Date convertToDate(String data) throws Exception{

		int ano = Integer.parseInt(data.substring(0, 4));
		int mes = Integer.parseInt(data.substring(5, 7));
		int dia = Integer.parseInt(data.substring(8, 10));

		Calendar cal = new GregorianCalendar(ano, mes-1, dia);
		//retornar um date..
		return cal.getTime();
	}
}
```

ManagedBean para retornar a listagem de funcionarios do banco de dados:

- ManagedBeanFuncionario.java

```
package br.com.brq.manager;

import java.util.List;

import br.com.brq.entities.Funcionario;
import br.com.brq.persistence.DAOFuncionario;

public class ManagedBeanFuncionario {

	//atributo que retorne para a página uma lista de funcionarios
	//que será obtida pelo Hibernate...
	private List<Funcionario> listagemFuncionarios;

	public List<Funcionario> getListagemFuncionarios() {

		try{
			DAOFuncionario d = new DAOFuncionario();
			listagemFuncionarios = d.findAll();
		}
		catch(Exception e){
			e.printStackTrace(); //console do servidor..
		}
```

```
		return listagemFuncionarios;
	}
}
```

Criando o servlet para controle de Funcionarios:

```
package br.com.brq.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 br.com.brq.entities.Funcionario;
import br.com.brq.persistence.DAOFuncionario;
import br.com.brq.util.FormatacaoData;

@WebServlet("/ControleFuncionario")
public class ControleFuncionario extends HttpServlet {
	private static final long serialVersionUID = 1L;

	public ControleFuncionario() {
		super();
	}

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

		//resgatar a variavel de ação enviada pela página..
		String action = request.getParameter("action");

		if("cadastrar".equalsIgnoreCase(action)){

			try{

				Funcionario f = new Funcionario(); //entidade..
```

```
                f.setNome(request.getParameter("nome"));
                f.setSalario(Double.parseDouble
                        (request.getParameter("salario")));
                f.setDataAdmissao(FormatacaoData.
                convertToDate(request.getParameter("dataadmissao")));

                DAOFuncionario d = new DAOFuncionario();
                d.insert(f); //gravando..

                request.setAttribute("mensagem", "Funcionario "
                        + f.getNome()
                        + ", cadastrado com sucesso.");
        }
        catch(Exception e){
                //gerar mensagem de erro..
                request.setAttribute("mensagem", e.getMessage());
        }
        finally{
                //redirecionar de volta para a página..
                request.getRequestDispatcher("manterfuncionarios.jsp")
                        .forward(request, response);
        }

}
else if("excluir".equalsIgnoreCase(action)){

        try{

                //resgatar o id do funcionario..
                Integer idFuncionario = Integer.parseInt(
                                request.getParameter("id"));

                //classe de persistencia..
                DAOFuncionario d = new DAOFuncionario();
                //buscar o funcionario pelo id..
                Funcionario f = d.findById(idFuncionario);

                //excluir o funcionario..
                d.delete(f);

                request.setAttribute("mensagem", "Funcionario "
                        + f.getNome()
                        + ", excluido com sucesso.");
        }
```

```
                catch(Exception e){
                        //mensagem de erro..
                        request.setAttribute("mensagem", e.getMessage());
                }
                finally{
                        //redirecionar de volta para a página..
                        request.getRequestDispatcher("manterfuncionarios.jsp")
                                .forward(request, response);
                }
        }

    }

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

    protected void doPost(HttpServletRequest request,
                        HttpServletResponse response)
                throws ServletException, IOException {
                execute(request, response);
    }
}
```

Criando as páginas no projeto:

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Projeto</title>

      <!-- Arquivos CSS do bootstrap -->
      <link rel="stylesheet" type="text/css"
            href="css/bootstrap.min.css"/>
      <link rel="stylesheet" type="text/css"
            href="css/bootstrap-theme.min.css"/>

</head>
```

```
<body class="container">

	<h2>Projeto Controle de Funcionarios</h2>
	<hr/>

	<a href="manterfuncionarios.jsp">Manter Funcionarios</a>

</body>
</html>
```

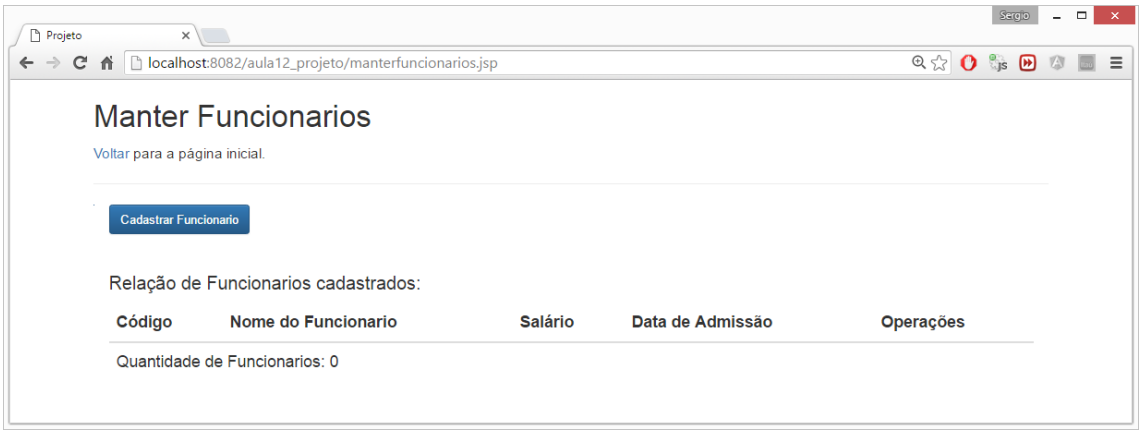
- manterfuncionarios.jsp

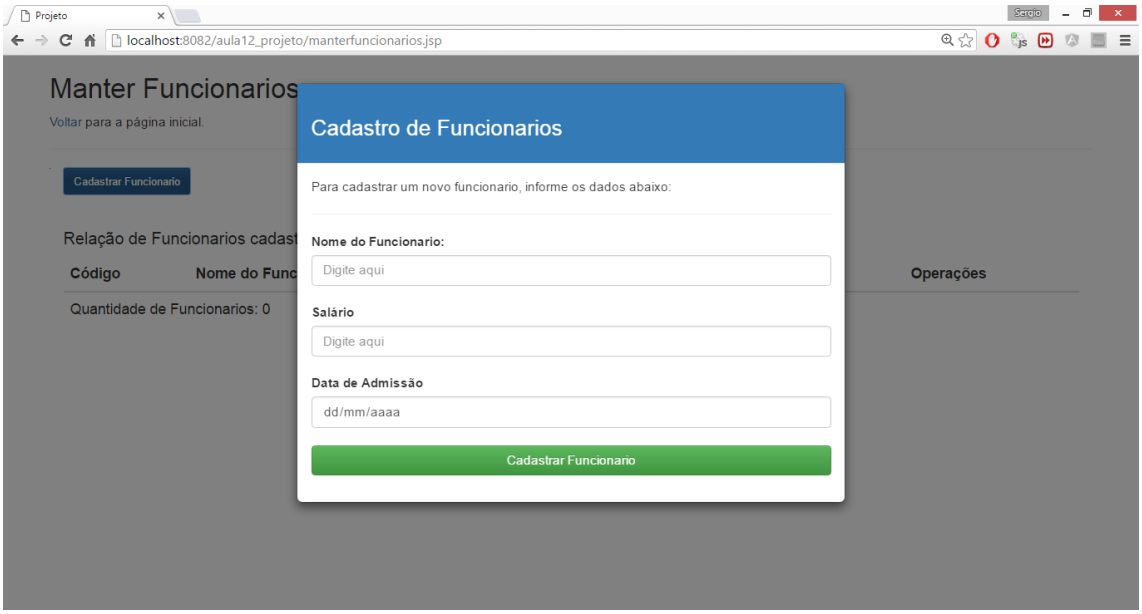
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!-- Referenciar o ManagedBean -->
<jsp:useBean class="br.com.brq.manager.ManagedBeanFuncionario"
id="mb"></jsp:useBean>

<!-- Declarar as taglibs -->
<%@ 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>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Projeto</title>

	<!-- Arquivos CSS do bootstrap -->
	<link rel="stylesheet" type="text/css"
		href="css/bootstrap.min.css"/>

	<link rel="stylesheet" type="text/css"
		href="css/bootstrap-theme.min.css"/>

	<!-- Bloco CSS (Folha de estilo) -->
	<style type="text/css">
		/* formatação das mensagens de erro.. */
		label.error
		{
			color: #FF0000;
```

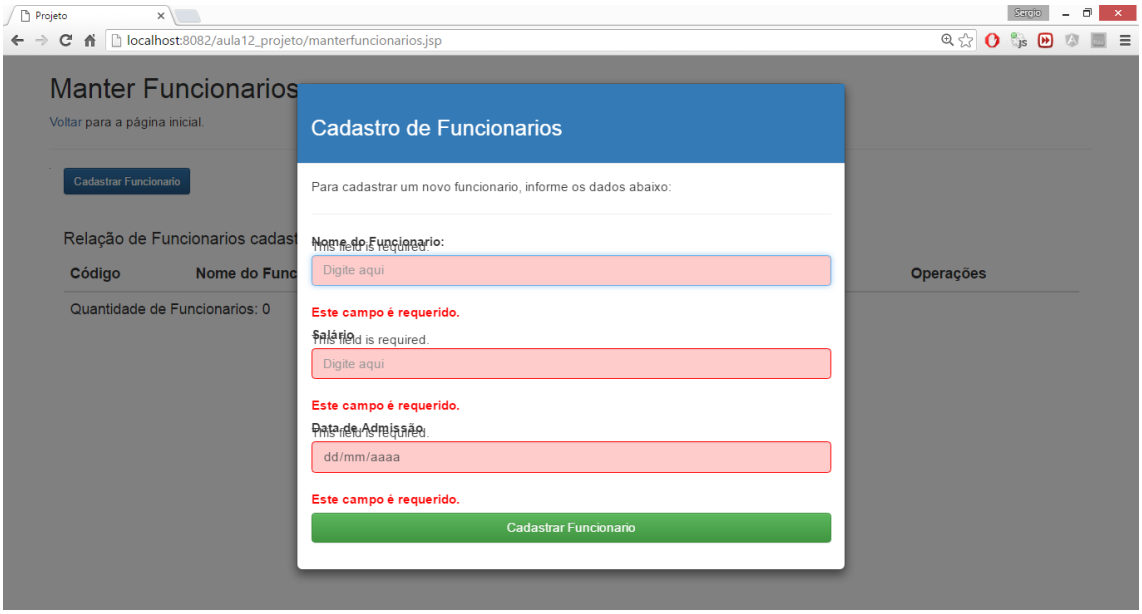
```
                font-weight: bold;
            }

            /* formatação nos campos que obtiveram erro */
            input.error
            {
                border: 1px solid #FF0000;
                background-color: #FFCCCC;
            }

        </style>

        <!-- Arquivos JavaScript -->
        <script type="text/javascript"
            src="js/jquery-1.12.4.min.js"></script>
        <script type="text/javascript"
            src="js/bootstrap.min.js"></script>
        <script type="text/javascript"
            src="js/jquery.validate.min.js"></script>
        <script type="text/javascript"
            src="js/messages_pt_BR.min.js"></script>

        <!-- bloco javascript -->
        <script type="text/javascript">

            //função inicial do jquery..
            $(document).ready( //quando a página estiver carregada..
                function(){ //faça..

                    //aplicar validação ao formulario (jquery validate)
                    $("#formulario").validate();

                }
            );

        </script>

    </head>
    <body class="container">

        <h2>Manter Funcionarios</h2>
        <a href="index.jsp">Voltar</a> para a página inicial.
        <hr/>

        <div class="col-md-12">

            <button data-target="#janela" data-toggle="modal"
                class="btn btn-primary btn-sm">
                Cadastrar Funcionario
            </button>

            <br/>

            <h4>${mensagem}</h4>
```

```
<br/>

<h4>Relação de Funcionarios cadastrados:</h4>

<table class="table table-hover">
	<thead>
		<tr>
			<th>Código</th>
			<th>Nome do Funcionario</th>
			<th>Salário</th>
			<th>Data de Admissão</th>
			<th>Operações</th>
		</tr>
	</thead>
	<tbody>

		<c:forEach items="${mb.listagemFuncionarios}"
				var="f">
		<tr>
			<td>${f.idFuncionario}</td>
			<td>${f.nome}</td>
			<td>
				<fmt:formatNumber type="currency"
				value="${f.salario}"/>
			</td>
			<td>
				<fmt:formatDate pattern="dd/MM/yyyy"
				value="${f.dataAdmissao}"/>
			</td>
			<td> <a
					href="ControleFuncionario?action
					=excluir&id=${f.idFuncionario}"
					class="btn btn-danger btn-sm"
					onclick="return confirm('Deseja
					excluir?');">Excluir</a> </td>
		</tr>
		</c:forEach>

	</tbody>
	<tfoot>
		<tr>
			<td colspan="5">Quantidade de Funcionarios:
				${fnc:length(mb.listagemFuncionarios)}
			</td>
		</tr>
	</tfoot>
</table>


</div>

<!-- Janela de cadastro do funcionario.. -->
<div id="janela" class="modal fade">
	<div class="modal-dialog">
		<div class="modal-content">
```

```
                        <div class="modal-header bg-primary">
                              <h3>Cadastro de Funcionarios</h3>
                        </div>
                        <div class="modal-body">

                              Para cadastrar um novo funcionario,
                              informe os dados abaixo:
                              <hr/>

                        <form name="formulario" id="formulario"
                        method="post"
                        action="ControleFuncionario?action=cadastrar">

                        <label>Nome do Funcionario:</label>
                        <input type="text" name="nome"
                              class="form-control required"
                              placeholder="Digite aqui"/>
                        <br/>

                        <label>Salário</label>
                        <input type="text" name="salario"
                              class="form-control required number"
                              placeholder="Digite aqui"/>
                        <br/>

                        <label>Data de Admissão</label>
                        <input type="date" name="dataadmissao"
                                    class="form-control required"/>
                        <br/>

                        <input type="submit" value="Cadastrar Funcionario"
                              class="btn btn-success btn-block"/>

                        </form>

                        </div>
                  </div>
            </div>
      </div>
</body>
</html>
```




Projeto
localhost:8082/aula12_projeto/manterfuncionarios.jsp
Manter Funcionarios
Voltar para a página inicial.
Cadastrar Funcionario
Relação de Funcionarios cadast
Código
Nome do Func
Operações
Quantidade de Funcionarios: 0
Cadastro de Funcionarios
Para cadastrar um novo funcionario, informe os dados abaixo:
Nome do Funcionario:
Digite aqui
Salário
Digite aqui
Data de Admissão
dd/mm/aaaa
Cadastrar Funcionario


Projeto
localhost:8082/aula12_projeto/manterfuncionarios.jsp
Manter Funcionarios
Voltar para a página inicial.
Cadastrar Funcionario
Relação de Funcionarios cadast
Código
Nome do Func
Operações
Quantidade de Funcionarios: 0
Cadastro de Funcionarios
Para cadastrar um novo funcionario, informe os dados abaixo:
Nome do Funcionario:
This field is required.
Digite aqui
Este campo é requerido.
Salário
This field is required.
Digite aqui
Este campo é requerido.
Data de Admissão
This field is required.
dd/mm/aaaa
Este campo é requerido.
Cadastrar Funcionario