



Criando o projeto:

- **File > New > Project**

New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: aula13projeto

Project location

Use default location

Location: C:\Users\Sergio\Desktop\work\aula13projeto

Target runtime

Apache Tomcat v8.0

Dynamic web module version

3.1

Configuration

Default Configuration for Apache Tomcat v8.0

A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

Add project to an EAR

EAR project name: EAR

Working sets

Add project to working sets

Working sets:



Criando e mapeando as entidades:

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

import java.util.List;

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.OneToMany;
import javax.persistence.Table;

//JPA - Java Persistence API
@Entity
@Table(name = "autor")
@NamedQueries(
{
    @NamedQuery(name = "autor.findall",
    query = "select a from Autor as a order by a.nome")
}
)
public class Autor {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name = "idautor")
    private Integer idAutor;

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

    // Relacionamento (muitos).. 
    //1 Autor para muitos Livros
    //definir o nome do atributo na classe Livro que tem a FK..
    @OneToMany(mappedBy = "autor")
    //nome do atributo na Classe Livro onde esta a FK..
    private List<Livro> livros;

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



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

```
}

public Autor(Integer idAutor, String nome) {
    this.idAutor = idAutor;
    this.nome = nome;
}

public Autor(Integer idAutor, String nome, List<Livro> livros) {
    this(idAutor, nome);
    this.livros = livros;
}

public Integer getIdAutor() {
    return idAutor;
}

public void setIdAutor(Integer idAutor) {
    this.idAutor = idAutor;
}

public String getNome() {
    return nome;
}

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

public List<Livro> getLivros() {
    return livros;
}

public void setLivros(List<Livro> livros) {
    this.livros = livros;
}

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

}
```



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

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

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.NamedQueries;
import javax.persistenceNamedQuery;
import javax.persistence.Table;

@Entity
@Table(name = "livro")
@NamedQueries(
{
    @NamedQuery(name = "livro.findall",
    query = "select l from Livro as l")
}
)
public class Livro {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "idlivro")
    private Integer idLivro;

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

    @Column(name = "resumo", length = 255, nullable = false)
    private String resumo;

    @Column(name = "foto", length = 50, nullable = false, unique = true)
    private String foto;

    /*
     * Em JPA, temos 4 tipos de relacionamentos:
     * OneToOne, ManyToOne, OnetoMany e ManyToMany
     */
    @ManyToOne //muitos Livros para 1 Autor..
    @JoinColumn(name = "idautor", nullable = false) //chave estrangeira..
```



```
private Autor autor;

@ManyToOne //muitos livros para 1 Editora..
@JoinColumn(name = "ideditora", nullable = false) //chave estrangeira..
private Editora editora;

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

public Livro(Integer idLivro, String titulo, String resumo, String foto) {
    this.idLivro = idLivro;
    this.titulo = titulo;
    this.resumo = resumo;
    this.foto = foto;
}

public Livro(Integer idLivro, String titulo, String resumo, String foto,
        Autor autor, Editora editora) {
    this(idLivro, titulo, resumo, foto);
    this.autor = autor;
    this.editora = editora;
}

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 getResumo() {
    return resumo;
}
```



```
public void setResumo(String resumo) {  
    this.resumo = resumo;  
}  
  
public String getFoto() {  
    return foto;  
}  
  
public void setFoto(String foto) {  
    this.foto = foto;  
}  
  
public Autor getAutor() {  
    return autor;  
}  
  
public void setAutor(Autor autor) {  
    this.autor = autor;  
}  
  
public Editora getEditora() {  
    return editora;  
}  
  
public void setEditora(Editora editora) {  
    this.editora = editora;  
}  
  
@Override  
public String toString() {  
    return "Livro [idLivro=" + idLivro + ", titulo=" + titulo  
           + ", resumo=" + resumo + ", foto=" + foto + "]";  
}  
}  
  
package br.com.brq.entities;  
  
import java.util.List;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;
```



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

```
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "editora")
@NamedQueries(
{
    @NamedQuery(name = "editora.findall",
    query = "select e from Editora as e order by e.nome")
}
)
public class Editora {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "ideditora")
    private Integer idEditora;

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

    // Relacionamento (muitos)...
    @OneToMany(mappedBy = "editora")
    // nome do atributo na classe Livro que contem a FK..
    private List<Livro> livros;

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

    public Editora(Integer idEditora, String nome) {
        this.idEditora = idEditora;
        this.nome = nome;
    }

    public Editora(Integer idEditora, String nome, List<Livro> livros) {
        this(idEditora, nome);
        this.livros = livros;
    }
}
```



```
public Integer getIdEditora() {
    return idEditora;
}

public void setIdEditora(Integer idEditora) {
    this.idEditora = idEditora;
}

public String getNome() {
    return nome;
}

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

public List<Livro> getLivros() {
    return livros;
}

public void setLivros(List<Livro> livros) {
    this.livros = livros;
}

@Override
public String toString() {
    return "Editora [idEditora=" + idEditora + ", nome=" + nome + "]";
}
}
```

Configurando o Hibernate:

```
<?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/aula13</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">brqbrq</property>
    </session-factory>
</hibernate-configuration>
```



```
<property name="hibernate.show_sql">true</property>
<property name="hibernate.format_sql">true</property>

<mapping class="br.com.brq.entities.Autor"/>
<mapping class="br.com.brq.entities.Editora"/>
<mapping class="br.com.brq.entities.Livro"/>

</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;

public class GenerateTables {

    public static void main(String[] args) {

        try{

            Configuration cfg = new AnnotationConfiguration();
            cfg.configure("br/com/brq/config/mysql_hibernate.cfg.xml");

            SchemaExport s = new SchemaExport(cfg);
            s.create(true, true);
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }

    }
}
```

Criando a classe 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 um DAO genérico para as operações de INSERT, UPDATE e DELETE

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

public interface IDAOGeneric<T> {

    void insert(T obj) throws Exception; //método abstrato.

    void update(T obj) throws Exception; //método abstrato.

    void delete(T obj) throws Exception; //método abstrato.

}

package br.com.brq.persistence.generics;

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



```
import br.com.brq.persistence.HibernateUtil;

public abstract class DAOGeneric<T> implements IDAOGeneric<T> {

    protected Session session;
    protected Transaction transaction;
    protected Query query;

    @Override
    public void insert(T obj) throws Exception {
        session = HibernateUtil.getSessionFactory().openSession();

        transaction = session.beginTransaction();
        session.save(obj);
        transaction.commit();

        session.close();
    }

    @Override
    public void update(T obj) throws Exception {
        session = HibernateUtil.getSessionFactory().openSession();

        transaction = session.beginTransaction();
        session.update(obj);
        transaction.commit();

        session.close();
    }

    @Override
    public void delete(T obj) throws Exception {
        session = HibernateUtil.getSessionFactory().openSession();

        transaction = session.beginTransaction();
        session.delete(obj);
        transaction.commit();

        session.close();
    }

}
```



Criando as classes de persistência para Autor, Livro e Editora

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

import java.util.List;

import br.com.brq.entities.Autor;
import br.com.brq.persistence.generics.DAOGeneric;

public class DAOAutor extends DAOGeneric<Autor>{

    //método para buscar 1 autor pelo id..
    public Autor findById(Integer idAutor) throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();

        Autor a = (Autor) session.get(Autor.class, idAutor);

        session.close();
        return a; //retornando o objeto..
    }

    //método para listar todos os autores..
    public List<Autor> findAll() throws Exception{
        session = HibernateUtil.getSessionFactory().openSession();

        query = session.getNamedQuery("autor.findall");
        List<Autor> lista = query.list();

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

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

import java.util.List;

import br.com.brq.entities.Editora;
import br.com.brq.persistence.generics.DAOGeneric;
```



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

```
public class DAOEditora extends DAOGeneric<Editora> {

    public Editora findById(Integer idEditora) throws Exception {
        session = HibernateUtil.getSessionFactory().openSession();

        Editora e = (Editora) session.get(Editora.class, idEditora);

        session.close();
        return e;
    }

    public List<Editora> findAll() throws Exception {
        session = HibernateUtil.getSessionFactory().openSession();

        query = session.getNamedQuery("editora.findall");
        List<Editora> lista = query.list();

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

}
```

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

import java.util.List;

import br.com.brq.entities.Livro;
import br.com.brq.persistence.generics.DAOGeneric;

public class DAOLivro extends DAOGeneric<Livro>{

    public Livro findById(Integer idLivro) throws Exception {
        session = HibernateUtil.getSessionFactory().openSession();

        Livro l = (Livro) session.get(Livro.class, idLivro);

        session.close();
        return l;
    }

}
```



```
public List<Livro> findAll() throws Exception {  
    session = HibernateUtil.getSessionFactory().openSession();  
  
    query = session.getNamedQuery("livro.findall");  
    List<Livro> lista = query.list();  
  
    session.close();  
    return lista;  
}  
  
}
```

Camada de Apresentação

Criando a página inicial do 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>  
  
<link rel="stylesheet" type="text/css"  
      href="css/bootstrap.min.css"/>  
  
<link rel="stylesheet" type="text/css"  
      href="css/bootstrap-theme.min.css"/>  
  
<script type="text/javascript" src="js/jquery-1.12.4.min.js"></script>  
  
<script type="text/javascript" src="js/bootstrap.min.js"></script>  
  
</head>  
<body class="container">  
  
<jsp:include page="templates/topo.jsp"></jsp:include>  
  
<div class="col-md-12">  
    <h4>Bem vindo ao Projeto</h4>  
</div>  
  
</body>  
</html>
```



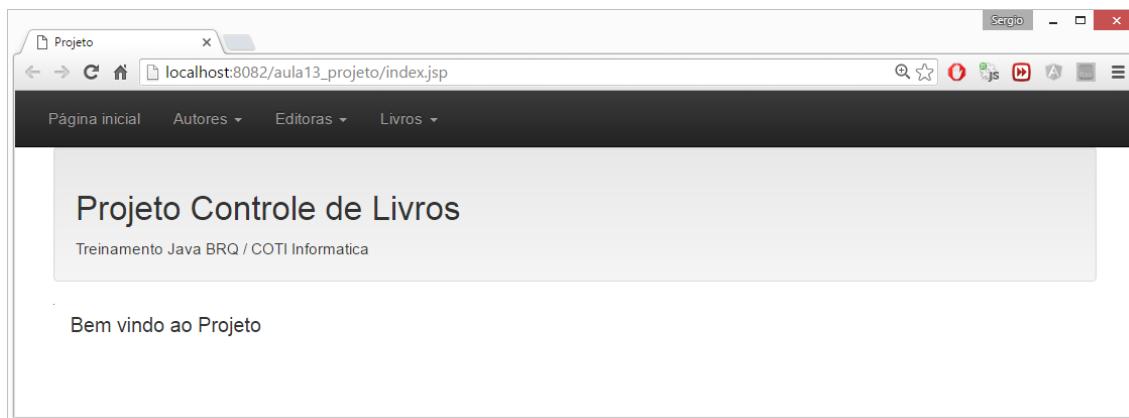
TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

Executando:



Implementando o cadastro de autores:

cadastroAutor.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>

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

<style type="text/css">
    label.error { color: #FF0000; }
</style>

<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">

    $(document).ready( //quando a página carregar...
        function(){ //faça..

            //validar o formulário..
            $("#formulario").validate(
            {
                rules:{
                    nome : "required"
                },
            
```



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

```
        messages:{  
            nome:{  
                required : "Por favor,  
                informe o nome do Autor."  
            }  
        }  
    }  
);  
});  
  
</script>  
  
</head>  
<body class="container">  
    <jsp:include page="templates/topo.jsp"></jsp:include>  
    <div class="col-md-12">  
        <h4>Cadastro de Autores</h4>  
        Para incluir um novo Autor, informe os dados abaixo:  
        <hr/>  
        <div class="col-md-4">  
            <form id="formulario" name="formulario" method="post"  
                  action="ControleAutor?action=cadastrar">  
                <label>Nome do Autor:</label>  
                <input type="text" id="nome" name="nome"  
                      class="form-control"  
                      placeholder="Digite aqui"/>  
                <br/>  
                <input type="submit" value="Cadastrar Autor"  
                      class="btn btn-success"/>  
                <br/><br/>  
                ${mensagem}  
            </form>  
        </div>  
    </div>  
</body>  
</html>
```

The screenshot shows a web browser window with the URL `localhost:8082/aula13_projeto/cadastroAutor.jsp`. The page title is "Projeto Controle de Livros" and the subtitle is "Treinamento Java BRQ / COTI Informatica". The main content area is titled "Cadastro de Autores" and contains the instruction "Para incluir um novo Autor, informe os dados abaixo:". Below this is a text input field with the placeholder "Digite aqui" and a green "Cadastrar Autor" button.



ControleAutor.java

```
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.Autor;
import br.com.brq.persistence.DAOAutor;

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

    public ControleAutor() {
        super();
    }

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

        String action = request.getParameter("action");

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

            try{
                Autor a = new Autor();
                a.setNome(request.getParameter("nome"));

                DAOAutor d = new DAOAutor();
                d.insert(a); //gravando..

                request.setAttribute("mensagem", "Autor "
                    + a.getNome()
                    + ", cadastrado com sucesso.");
            }
            catch(Exception e){
                request.setAttribute("mensagem", e.getMessage());
            }
        }
    }
}
```



```
        }
    finally{
        request.getRequestDispatcher("cadastroAutor.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 a página de cadastro de Editoras: cadastroEditora.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>

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

<style type="text/css">
    label.error { color: #FF0000; }
</style>

<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>
```



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

```
<script type="text/javascript">

$(document).ready( //quando a página carregar...
    function(){ //faça...

        //validar o formulário..
        $("#formulario").validate(
            {
                rules:{
                    nome : "required"
                },
                messages:{
                    nome:{ required : "Por favor, informe o nome da Editora."
                }
            }
        );
    }
);

</script>

</head>
<body class="container">
    <jsp:include page="templates/topo.jsp"></jsp:include>
    <div class="col-md-12">

        <h4>Cadastro de Editoras</h4>
        Para incluir uma nova Editora, informe os dados abaixo:
        <hr/>

        <div class="col-md-4">

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

                <label>Nome da Editora:</label>
                <input type="text" id="nome" name="nome"
                       class="form-control"
                       placeholder="Digite aqui"/>
                <br/>

                <input type="submit" value="Cadastrar Editora"
                       class="btn btn-success"/>
                <br/><br/>

                ${mensagem}

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



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

Criando o Servlet:

```
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.Editora;
import br.com.brq.persistence.DAOEditora;

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

    public ControleEditora() {
        super();
    }

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

        String action = request.getParameter("action");

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



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

```
Editora e = new Editora();
e.setNome(request.getParameter("nome"));

DAOEditora d = new DAOEditora();
d.insert(e); //gravando..

request.setAttribute("mensagem", "Editora "
+ e.getNome()
+ ", cadastrado com sucesso.");
}

catch(Exception e){
    request.setAttribute("mensagem", e.getMessage());
}
finally{
    request.getRequestDispatcher("cadastroEditora.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 os ManagedBeans para obter a listagem de Autores e Editoras:

```
package br.com.brq.managedbeans;

import java.util.List;

import br.com.brq.entities.Editora;
import br.com.brq.persistence.DAOEditora;

public class ManagedBeanEditora {

    private List<Editora> listagemEditoras;
```



```
public List<Editora> getListagemEditoras() {  
    try{  
        DAOEditora d = new DAOEditora();  
        listagemEditoras = d.findAll();  
    }  
    catch(Exception e){  
        e.printStackTrace();  
    }  
  
    return listagemEditoras;  
}  
}  
  
package br.com.brq.managedbeans;  
  
import java.util.List;  
  
import br.com.brq.entities.Autor;  
import br.com.brq.persistence.DAOAutor;  
  
public class ManagedBeanAutor {  
  
    private List<Autor> listagemAutores;  
  
    public List<Autor> getListagemAutores() {  
        try{  
            DAOAutor d = new DAOAutor();  
            listagemAutores = d.findAll();  
        }  
        catch(Exception e){  
            e.printStackTrace();  
        }  
  
        return listagemAutores;  
    }  
}
```

Criando a página de cadastro de Livro:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"  
pageEncoding="ISO-8859-1"%>  
  
<!-- Referenciar os ManagedBeans -->  
<jsp:useBean class="br.com.brq.managedbeans.ManagedBeanAutor"  
id="mbAutor"></jsp:useBean>
```



```
<jsp:useBean class="br.com.brq.managedbeans.ManagedBeanEditora"
id="mbEditora"></jsp:useBean>

<!-- TagLibraries (JSTL) -->
<%@ 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>

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

<style type="text/css">
    label.error { color: #FF0000; }
</style>

<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">

$(document).ready(
    function(){
        $("#formulario").validate(
            {
                rules:{
                    titulo : "required",
                    resumo : "required",
                    foto : "required",
                    autor : "required",
                    editora : "required"
                },
                messages:{
                    titulo :
                    { required : "Por favor, informe o titulo do Livro." },
                    resumo :
                    { required : "Por favor, informe o resumo do Livro." },
                    foto :
                    { required : "Por favor, envie a foto do Livro." },
                    autor :
                    { required : "Por favor, selecione o autor do Livro." },
                    editora :
                    { required : "Por favor, selecione a editora do Livro." }
                }
            );
    }
);

```



```
);

</script>

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

    <jsp:include page="templates/topo.jsp"></jsp:include>

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

        <h4>Cadastro de Livros</h4>
        Para incluir um novo Livro, informe os dados abaixo:
        <hr/>

        <div class="col-md-4">

            <form id="formulario" name="formulario"
                  method="post" enctype="multipart/form-data"
                  action="ControleLivro?action=cadastrar">

                <label>Título do Livro:</label>
                <input type="text" id="titulo" name="titulo"
                       class="form-control"
                       placeholder="Digite aqui"/>
                <br/>

                <label>Resumo do Livro:</label>
                <textarea id="resumo" name="resumo"
                          class="form-control"></textarea>
                <br/>

                <label>Foto do Livro:</label>
                <input type="file" id="foto" name="foto"
                       class="form-control"/>
                <br/>

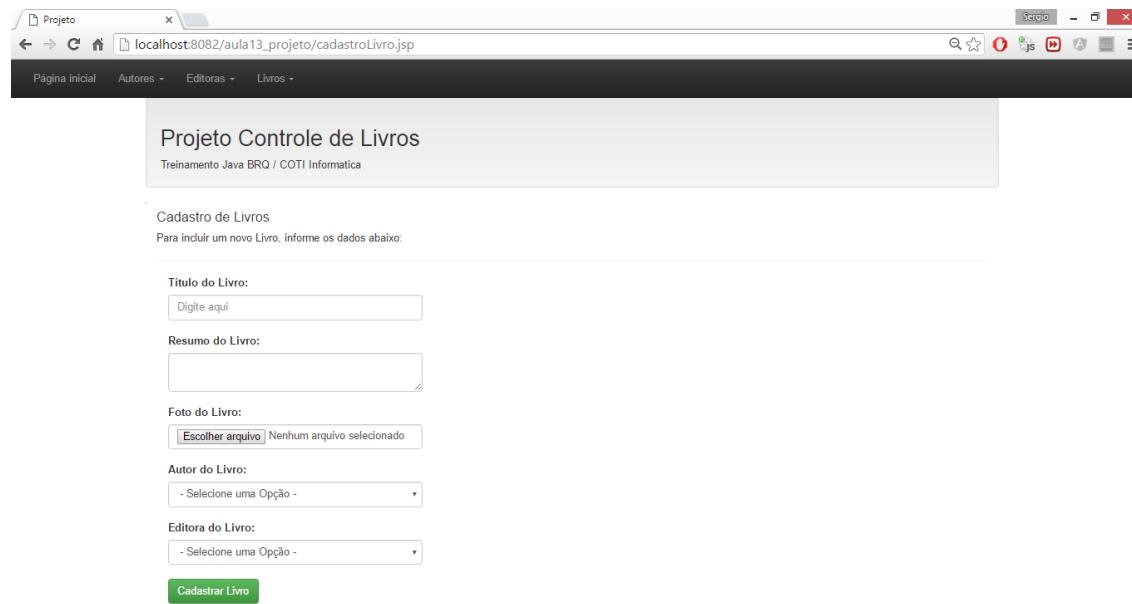
                <label>Autor do Livro:</label>
                <select id="autor" name="autor"
                       class="form-control">
                    <option value="">- Selecione uma Opção -
                    </option>
                    <c:forEach items="${mbAutor.listagemAutores}"
                               var="a">
                        <option value="${a.idAutor}">
                            ${a.nome} </option>
                    </c:forEach>
                </select>
                <br/>

                <label>Editora do Livro:</label>
                <select id="editora" name="editora"
                       class="form-control">
```



```
<option value="">- Selecione uma Opção -  
</option>  
  
<c:forEach  
    items="${mbEditora.listagemEditoras}"  
    var="e">  
    <option value="${e.idEditora}">  
        ${e.nome} </option>  
</c:forEach>  
  
</select>  
<br/>  
  
<input type="submit" value="Cadastrar Livro"  
       class="btn btn-success"/>  
<br/><br/>  
${mensagem}  
  
</form>  
  
</div>  
  
</div>  
  
</body>  
</html>
```

Executando:



Servlet para Controle de Livros

```
package br.com.brq.control;
```



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula
13

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.Part;

import br.com.brq.entities.Livro;
import br.com.brq.persistence.DAOAutor;
import br.com.brq.persistence.DAOEditora;
import br.com.brq.persistence.DAOLivro;

@WebServlet("/ControleLivro")
@MultipartConfig() //habilitando o servlet a receber um upload..
public class ControleLivro extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public ControleLivro() {
        super();
    }

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

        String action = request.getParameter("action");

        if("cadastrar".equalsIgnoreCase(action)){
            try{
                Livro l = new Livro(); //entidade..

                l.setTitulo(request.getParameter("titulo"));
                l.setResumo(request.getParameter("resumo"));

                UUID guid = UUID.randomUUID();
                //GUID - Global unique identifier..
                l.setFoto(guid.toString() + ".jpg");

                //obter o autor e a editora do livro..
                DAOAutor daoAutor = new DAOAutor();
                DAOEditora daoEditora = new DAOEditora();


```



TREINAMENTO JAVA – BRQ/SP

Terça-feira, 31 de Maio de 2016

JSP, Servlets, Filters e JSTL. Acesso a banco de dados com Hibernate.
Mapeamento de Relacionamentos. Upload de Arquivos

Aula

13

```
DAOLivro daoLivro = new DAOLivro();

//resgatar o autor e a editora selecionados..
Integer idAutor = Integer.parseInt(
    request.getParameter("autor"));
Integer idEditora = Integer.parseInt(
    request.getParameter("editora"));

//buscar autor e editora e relacionar ao livro..
l.setAutor(daoAutor.findById(idAutor)); //relacionando..
l.setEditora(daoEditora.findById(idEditora)); //relacionando..

//cadastrar o livro..
daoLivro.insert(l);

//upload...
//resgatar o campo foto (file)
Part foto = request.getPart("foto"); //arquivo..
//definir o local onde o arquivo será salvo..
//String pasta = getServletContext().getRealPath("/img");
String pasta = "C:\\\\Users\\\\treina2\\\\Desktop\\\\workspace
    \\\\aula13_projeto\\\\WebContent\\\\img";

FileOutputStream stream = new FileOutputStream(
    pasta + "/" + l.getFoto());
InputStream input = foto.getInputStream(); //lendo o arquivo..
byte[] buffer = new byte[1024];
while(input.read(buffer) > 0){
    stream.write(buffer);
}
stream.close();

request.setAttribute("mensagem", "Livro " + l.getTitulo()
    + ", cadastrado com sucesso.");
}

catch(Exception e){
    e.printStackTrace();
    request.setAttribute("mensagem", e.getMessage());
}

finally{
    request.getRequestDispatcher("cadastroLivro.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);  
}  
}
```

Continua...