Roteiro para desenvolvimento da aula sobre Struts 2.0

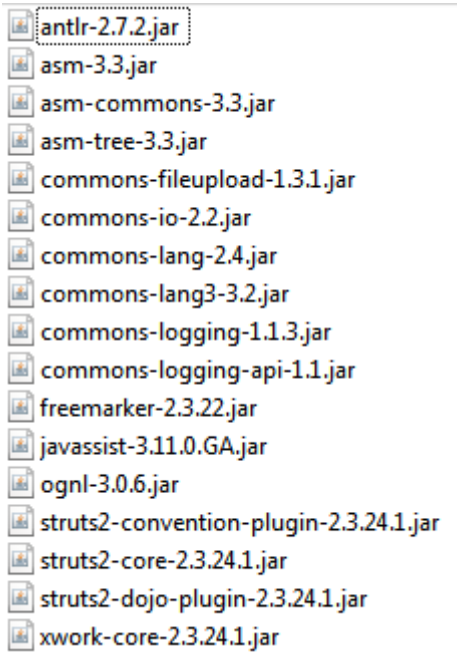
Este modelo apresenta os procedimentos para elaboração de uma aplicação **Struts 2.0** contendo um formulário.

Para desenvolver este exercício, siga estes passos:

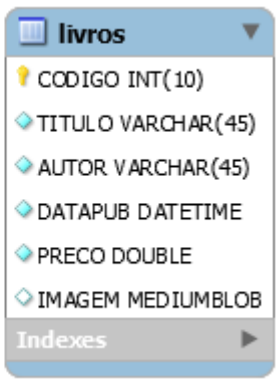
1. Defina um novo projeto **Dynamic Web Project**.

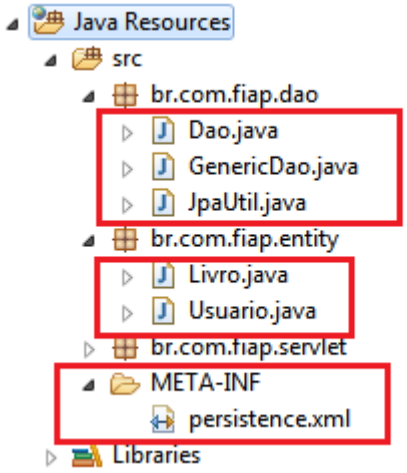
2. No link http://struts.apache.org/ obter a api Struts2, ou recuperar os arquivos abaixo no portal da Fiap:

3. Inserir as bibliotecas:



4. Usar o banco de dados e a estrutura de classes definidos abaixo, e disponibilizados no portal:





5. Criar a seguinte classe referente ao *action* Livro (observe as anotações):

```
package br.com.fiap.action;

import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.struts2.convention.annotation.Action;
import org.apache.struts2.convention.annotation.Result;
import org.apache.struts2.convention.annotation.Results;

import br.com.fiap.dao.GenericDao;
import br.com.fiap.entity.Livro;

import com.opensymphony.xwork2.ActionSupport;

@Results({
    @Result(name="ok", location="/menu.jsp"),
    @Result(name="erro", location="/erro.jsp"),
    @Result(name="input", location="/cadLivros.jsp"),
    @Result(name="lista", location="/listaLivros.jsp")
})
public class LivroAction extends ActionSupport{

    private static final long serialVersionUID = 1L;

    private File figura;
    private Livro livro;
    private List<Livro> listaLivros;

    public LivroAction(){
        livro = new Livro();
        listaLivros = new ArrayList<>();
    }

    public File getFigura() {
```

```
        return figura;
    }

    public void setFigura(File figura) {
        this.figura = figura;
    }

    public Livro getLivro() {
        return livro;
    }

    public void setLivro(Livro livro) {
        this.livro = livro;
    }


    public List<Livro> getListaLivros() {
        return listaLivros;
    }


    public void setListaLivros(List<Livro> listaLivros) {
        this.listaLivros = listaLivros;
    }

    @Action(value="/cadastro")
    public String incluir(){
        try {

            InputStream inputStream = new FileInputStream(figura);
            byte[] imagem = new byte[(int)figura.length()];
            inputStream.read(imagem, 0, (int)figura.length());
            livro.setImagem(imagem);

            GenericDao<Livro> dao = new GenericDao<Livro>(Livro.class);
            dao.adicionar(livro);

            return "ok";
        } catch (Exception e) {
            return "erro";
        }
    }

    @Action(value="/consulta")
    public String listar(){
        GenericDao<Livro> dao = new GenericDao<Livro>(Livro.class);
        listaLivros = dao.listar();
        return "lista";
    }
}
```

6. Definir a página de cadastro cadLivros.jsp (observe o atributo action do formulário):

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
```

```
<%@ taglib prefix="sx" uri="/struts-dojo-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<sx:head/>
<title>Cadastro de Livros</title>
</head>
<body>
    <h1>Cadastro de Livros</h1>
    <s:form action="cadastro" method="post" enctype="multipart/form-data">
      <s:textfield name="livro.codigo" label="Código" size="10" />
      <s:textfield name="livro.titulo" label="Título" size="30" />
      <s:textfield name="livro.autor" label="Autor" size="30" />
      <sx:datetimepicker name="livro.dataPublicacao"
                label="Data Publicação" displayFormat="dd/MM/yyyy"/>
      <s:textfield name="livro.preco" label="Preço" size="30" />
      <s:file name="figura" label="Selecione a imagem:" />
      <s:submit name="submit" label="Incluir" align="center" />
    </s:form>
</body>
</html>
```

7. No arquivo web.xml, realizar as configurações:

```
<?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_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>ExemploStruts2</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <filter>
        <filter-name>struts2</filter-name>
        <filter-class>
          org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
         </filter-class>
    </filter>
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

</web-app>
```

8. Na seqüência, definir as páginas JSP para listagem de livros, erro e menu:

menu.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ taglib uri="/struts-tags" prefix="s" %>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
<title>Login</title>
</head>
<body>
    <ul>
        <li><s:a href="cadLivros.jsp">Cadastro de Livros</s:a> </li>
        <li><s:a href="consulta.action">Lista de Livros</s:a> </li>
    </ul>
</body>
</html>
```

erro.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>Erro</title>
</head>
<body>
    <h2>Por favor, preencher corretamente os dados solicitados!</h2>
</body>
</html>
```

listaLivros.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Lista de Livros</title>
</head>
<body>
    <h1>Lista de Livros</h1>
    <table>
        <tr>
            <td>CÓDIGO</td>
            <td>TÍTULO</td>
            <td>AUTOR</td>
        </tr>

        <s:iterator value="listaLivros">
            <tr>
                <td><s:property value="codigo" /></td>
                <td><s:property value="titulo" /></td>
                <td><s:property value="autor" /></td>
            </tr>

        </s:iterator>
    </table>
</body>
</html>
```

9. Executar a aplicação a partir do menu.jsp

10. Como exercício, incluir na listagem um link par exibir a figura do livro, e implementar o recurso para Login.

    Para exibir a imagem, um exemplo de aplicação é mostrado abaixo:

    Local onde a imagem será exibida:

```
<img src="imagem?id=${livro.codigo}"/>
```

    Servlet para recuperar a imagem:

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

    public ServletImagem() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        int id = Integer.parseInt(request.getParameter("id"));
        GenericDao<Livro> dao = new GenericDao<>(Livro.class);
        byte[] imagem = dao.buscar(id).getImagem();
        response.setContentType("image/jpeg");
        ServletOutputStream os = response.getOutputStream();
        os.write(imagem);
        os.close();
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        // TODO Auto-generated method stub
    }
}
```

Na sequencia, realizar uma pesquisa sobre os elementos a seguir:

@RequiredFieldValidation

@IntRangeValidation

@EmailValidation

@DateRangeValidation

@ExpressionValidation

Com estes elementos, criar uma aplicação (não é necessário acessar banco de dados) que as contemple, além dos Actions definidos anteriormente.

Bom trabalho a todos!

Prof. Emilio