Java na WEB – Servlet

Objetivo:
Ao final da aula o aluno será capaz de:

- Utilizar Servlets para gerar páginas web dinâmicas.
- Utilizar Servlets para selecionar páginas JSPs.
- Utilizar Servlets como elemento de tomada de decisão.

Sumário

Prof. Vida
vida@utfpr.edu.br

11. Servlets

Introdução

Servlets são pequenos programas escritos em Java que rodam dentro de um servidor web em um Servlet Container (Tom Cat) e precisam ser executados por uma Java Virtual Machine denominada de Máquina Servlet. Um Servlet Container é o responsável pelo ciclo de vida de um servlet.

Dois pacotes compõe a arquitetura fundamental dos servlets o `javax.servlet` e o `javax.servlet.http`.

O `javax.servlet` provê recursos genéricos para um Servlet. O `javax.servlet.http`, é específico para o ambiente WEB.

O que distingue um servlet de uma típica classe?

Um servlet é uma classe que implementa a interface `javax.servlet.Servlet`. Essa interface define métodos para inicializar um servlet, responder a solicitações e remover o servlet da memória.

Essas ações são conhecidas como ciclo de vida e são chamadas na seguinte sequência:

1. O servlet é construído e inicializado através do método `public void init()`.
2. Qualquer solicitação do cliente é encaminhada ao método `public void service(ServletRequest req, ServletResponse res)`.
3. O service é retirado da memória através do método `public void destroy()`.

Além dos três métodos apresentados acima, a interface ainda disponibiliza os seguintes métodos:

- `getServletConfig`: o servlet pode utilizar esse método para ler alguma informação de configuração.
- `getServletInfo`: método que permite ao servlet retornar informações básicas a respeito de si, tais como: autor, versão e copyright.

O que distingue um HttpServlet de uma típica classe?

Uma cubclasse de HttpServlet precisa sobre-escrever pelo menus um dos métodos abaixo:

- `doGet`, Se o servlet suporta uma solicitação HTTP GET.
- `doPost`, para a solicitação HTTP POST
- `doPut`, para a solicitação HTTP PUT
- `doDelete`, para a solicitação HTTP DELETE.
- `init` e `destroy`, para genrenciar os revursos.
- `getServletInfo`, método utilizado pelo servlet para fornecer informações a respeito de si mesmo.

Não há necessidade de sobre-escrever o método service. Service está associado às solicitações HTTP que podem ser definidas pelos métodos *doXXX* já apresentados.

Código Java de um de exemplo de HttpServlet

```
1.   package javaNaWeb.Servlets.HttpServlets;
2.   import java.io.IOException;
3.   import java.io.PrintWriter;
4.   import javax.servlet.*;
5.   import javax.servlet.http.HttpServlet;
6.   import javax.servlet.http.HttpServletRequest;
7.   import javax.servlet.http.HttpServletResponse;

8.   public class MeuPrimeiroHttpServlet extends
9.   javax.servlet.http.HttpServlet
10.  {
11.     public void destroy() {}
12.     public ServletConfig getServletConfig()
13.          {return null;}
14.     public String getServletInfo() {return null;}
15.     public void init(ServletConfig config) {}

16.     public void doGet(HttpServletRequest req
17.                     , HttpServletResponse resp)
18.               throws IOException, ServletException
19.    {doPost(req,resp);}

20.     public void doPost(HttpServletRequest req
21.                     , HttpServletResponse resp)
22.            throws IOException, ServletException
23.     { resp.setContentType("text/html");
24.       PrintWriter out = resp.getWriter();
25.       out.println("<html>"
26.          + "<title>MeuPrimeiroServlet</title>"
27.          + "<body>");
28.       out.println("monta a página html");
29.       out.println("</body></html>");
30.       out.close();
31.      }
32.  }
```

Comentários:

O resultado da compilação do Servlet necessita ser armazenado dentro da pasta **`web-inf\classes`** com a mesma estrutura de diretório definida no pacote. Se durante a compilação, o JCreator forneceu a seguinte mensagem de erro abaixo, você deverá indicar para o ambiente JCreator aonde localiza-se o pacote necessário. A Seção 13 deste material apresenta como indicar o pacote para o JCreator.

```
--------------------Configuration: CalculaFatorial - j2sdk1.4.2 <Default>--------------------
C:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\teste\WEB-INF\classes\Calcula
import javax.servlet.*;
^
```

Figura 1-Mensagem de erro de ausência de pacote

A primeira linha do código anterior define essa estrutura. A Figura abaixo ilustra essa característica.

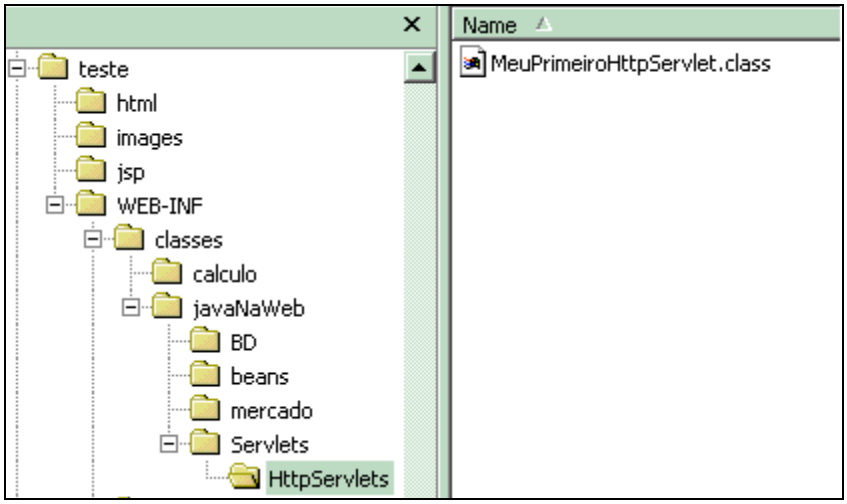

Figura 2-Estrutura de Diretório Sugerida para os Servlets

Antes da execução do HttpServlet, algumas linhas necessitam ser inseridas no arquivo web.xml para informar ao servidor a existência desse servlet.

O código em negrito a seguir necessita ser adicionado ao arquivo web.xml.

```
<servlet>
 <servlet-name>
         MeuPrimeiroHttpServlet
 </servlet-name>
 <servlet-class>
   javaNaWeb.Servlets.HttpServlets.MeuPrimeiro
   HttpServlet
 </servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name> MeuPrimeiroHttpServlet
```

```
  </servlet-name>
  <url-pattern>/MeuServlet
  </url-pattern>
</servlet-mapping>
```

A execução do Servlet

Figura 3-Exemplo de Execução do Servlet MeuServlet

12. Servlets e Documentos JSPs.

Os Servlets podem trabalhar juntos com as páginas JSPs. No próximo exemplo apresenta-se uma forma de integração. O código a seguir ilustra o comportamento de uma página JSP.

fatorialDeNovo.jsp - Notepad

File Edit Format Help

```
<!-- arquivo fatorialDeNovo.jsp -->

<html>
<body>

<% java.lang.Integer recebe = (java.lang.Integer)request.getAttribute("numero");
   int numero;

   if (recebe != null) {numero = recebe.intValue();}
   else {numero=-1;}

   if (numero>1)
   {
     out.println("<h2> O Fatorial de " + numero + "</h2>");
     for (int i = numero; i >1; i--)
       out.println("<li>O fatorial de "+ i +  " = " + i + "*" + (i-1) + "!");
   }
%>
</body>
</html>
```

Figura 4-Código do Arquivo fatorialDeNovo.jsp

Nesse código, a variável `recebe` lê o atributo “numero” como um objeto. Esse atributo será enviado por um Servlet. O método getAttribute(String) é similar ao

getParameter(String). A diferença é que o getAttribute recebe qualquer tipo de informação e não apenas String como o getParameter().

A página JSP definida na Figura 4, não foi projetada para ser chamada diretamente. Ela foi projetada para ser chamada por um servlet.

12.1 Código do Servlet

```
1  package javaNaWeb.Servlets;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5  import javax.servlet.*;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9
10
11 public class CalculaFatorial extends javax.servlet.http.HttpServlet
12 {
13   public void destroy() {}
14
15   public ServletConfig getServletConfig() {return null;}
16
17   public String getServletInfo() {return null;}
18
19   public void init(ServletConfig config) {}
20
21   public void doGet(HttpServletRequest req, HttpServletResponse resp)
22        throws IOException, ServletException
23   {
24     doPost(req,resp);
25   }
26
27   public void doPost(HttpServletRequest req, HttpServletResponse resp)
28        throws IOException, ServletException
29   {
30     int n = (int)Math.round(10*Math.random()+1);
31     req.setAttribute("numero",new Integer(n));
32
33     String pagina = "./jsp/fatorialDeNovo.jsp";
34     RequestDispatcher saida = req.getRequestDispatcher(pagina);
35     saida.forward(req,resp);
36   }
37 }
38
```

Figura 5-Código do Servlet CalculaFatorial

Quando o servlet é chamado pelo método post ou get, ele cria um objeto inteiro (Integer) com um número e coloca no objeto instância de HttpServletRequest uma chave denominada por “numero” (linas 30 e 31 do código da Figura 5).

A instância do objeto RequestDispatcher (mecanismo de localização de páginas JSPs e servlets) referenciará a página JSP desejada (linha 34 do código da Figura 5) e encaminha a solicitação para ela (linha 35 do código da Figura 5).

É importante destacar que não há nenhuma instrução out.println(). A informação é gerada dinâmicamente pelo servlet, a página JSP apenas apresenta a informação.

O servlet para ser executado necessita ser definifo no arquivo web.xml. A Figura 6 exemplifica essa definição.

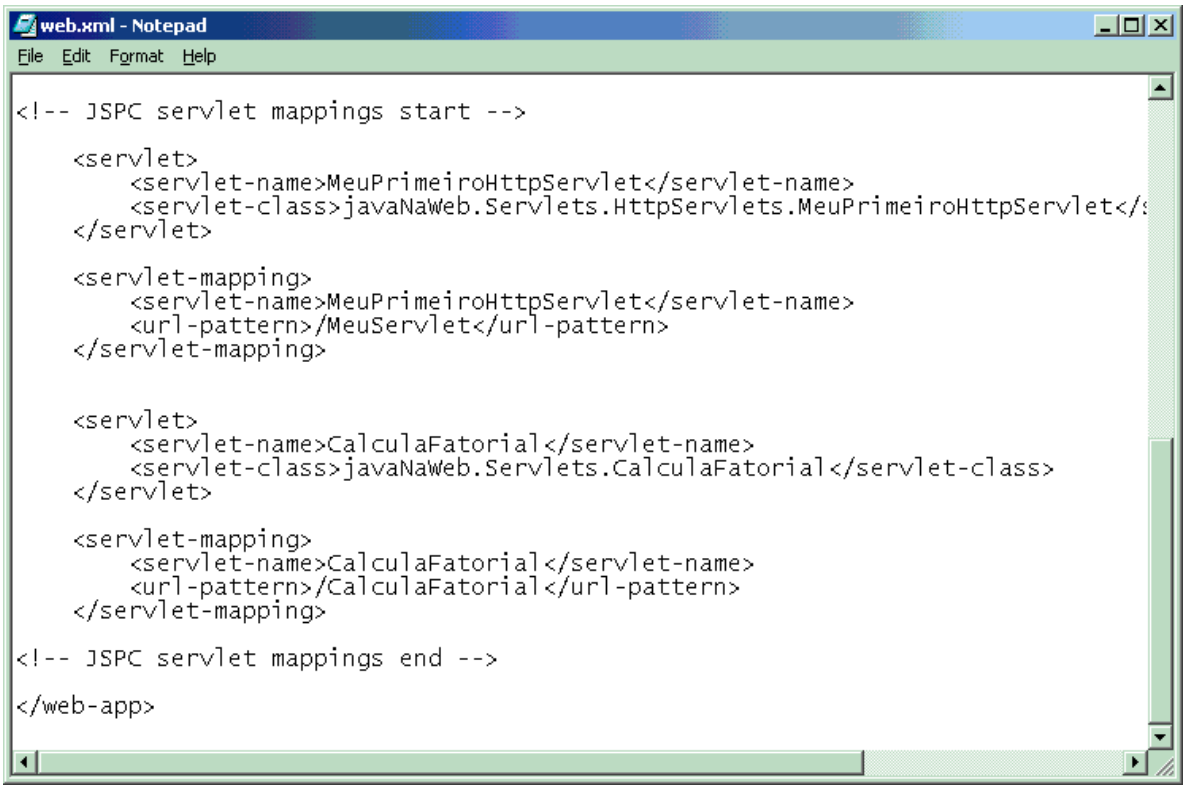

Figura 6-Configuração do Arquivo web.xml

Ao ser executado, a seguinte página dinâmica podem ser geradas:

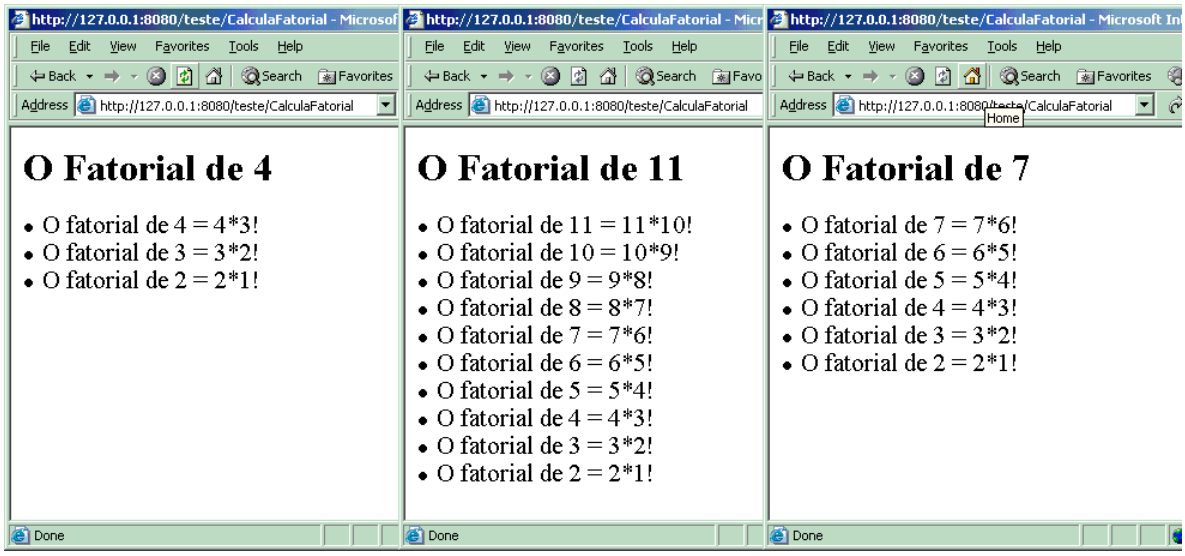

Figura 7-Exemplos de Execução do Serlvet CalculaFatorial

12.2 Servlets como mecanismo de tomada de decisão.

Um dos principais empregos de um servlet em uma aplicação é o de decidir o que apresentar ao usuário. Será apresentado um exemplo de como o Servlet pode selecionar uma página de acordo com alguma decisão lógica.

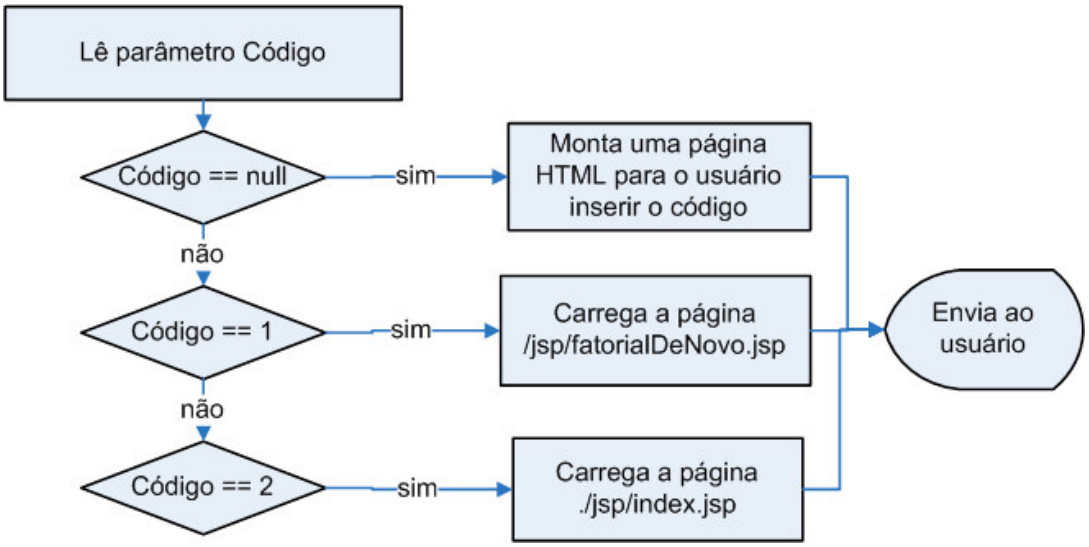

Figura 8-Lógica do Servlet

Conforme ilustrado na Figura 8, o Servlet lerá um parâmetro chamado `codigo`. Se esse parâmetro não foi fornecido pelo usuário, o seu valor será `null`. Se isso acontecer, o servlet montará uma página HTML para o usuário fornecer o código.

Se o usuário já forneceu um código, esse parâmetro é comparado com o número 1 e 2. Cada um desses valores correspodem a uma página JSP vista durante o curso. A representação da lógica da Figura 8 está em negrito no código abaixo.

O código para esse servlet é apresentado no quadro abaixo.

```
1.  ////////////////////////////////  arquivo  ServletGeral.java
2.  package javaNaWeb.Servlets;
3.  import java.io.IOException;
4.  import java.io.PrintWriter;
5.  import javax.servlet.*;
```

```
6.  import javax.servlet.http.HttpServlet;
7.  import javax.servlet.http.HttpServletRequest;
8.  import javax.servlet.http.HttpServletResponse;
9.  public class ServletGeral
10.                extends javax.servlet.http.HttpServlet
11. {
12.   public void destroy() {}
13.   public ServletConfig getServletConfig() {return null;}
14.   public String getServletInfo() {return null;}
15.   public void init(ServletConfig config) {}
16.   public void doGet(HttpServletRequest req
17.      ,HttpServletResponse resp) throws IOException
18.                                      , ServletException
19.     {  doPost(req,resp); }

20.    public void doPost(HttpServletRequest req
21.     ,HttpServletResponse resp) throws IOException
22.                                     , ServletException
23.    {
24.      String parametro = req.getParameter("codigo");
25.      if (parametro==null)
26.      {
27.        resp.setContentType("text/html");
28.        PrintWriter out = resp.getWriter();
29.        out.println("<html><body>");
30.        out.println("<form method=\'post\'>");
31.        out.println("<br>Codigo:<input type=\'text\' "
32.              + "name=\'codigo\' size=3>");
```

```
33.         out.println("<input type=\'submit\'  value=\'enviar'> "
34.                       + "</form>");
35.         out.println("</body></html>");
36.         out.close();
37.       }
38.      else
39.      {
40.        if (parametro.trim().equals("1"))
41.        { int n = (int)Math.round(10*Math.random()+1);
42.          req.setAttribute("numero",new Integer(n));
43.          String pagina = "./jsp/fatorialDeNovo.jsp";
44.          RequestDispatcher saida =
45.                       req.getRequestDispatcher(pagina);
46.          saida.forward(req,resp);
47.        }
48.        else if (parametro.trim().equals("2"))
49.        {
50.          RequestDispatcher saida =
51.                req.getRequestDispatcher("./jsp/index.jsp");
52.          saida.forward(req,resp);
53.        }
54.      }
55.    }
56. }
```

Destaca-se novamente a necessidade de modificação do arquivo web.xml. A Figura 9 enfatiza o trecho de código que necessita ser inserido no arquivo `web.xml`.

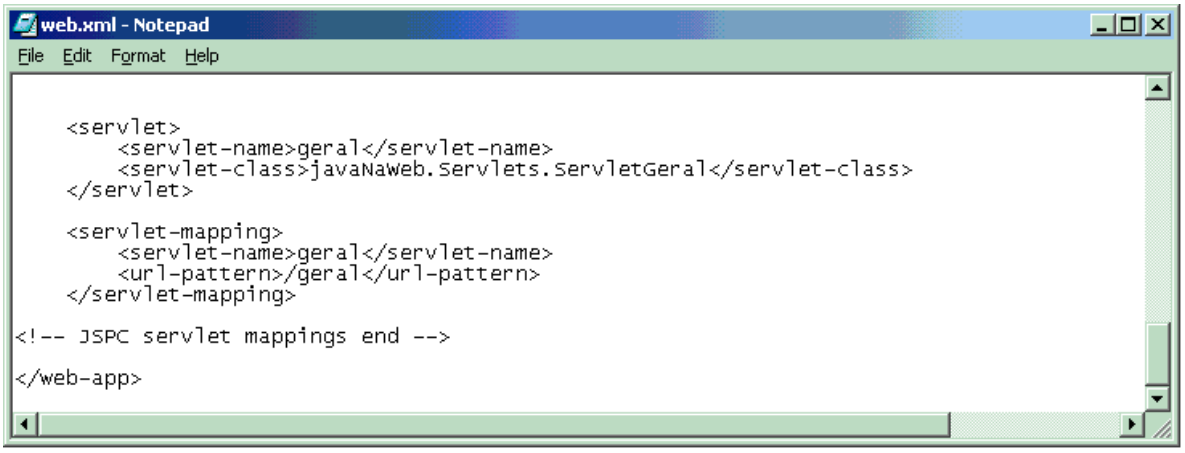

Figura 9-Trecho de Código que Necessita ser Inserido no Arquivo web.xml

A execução do Servlet é apresentada a seguir

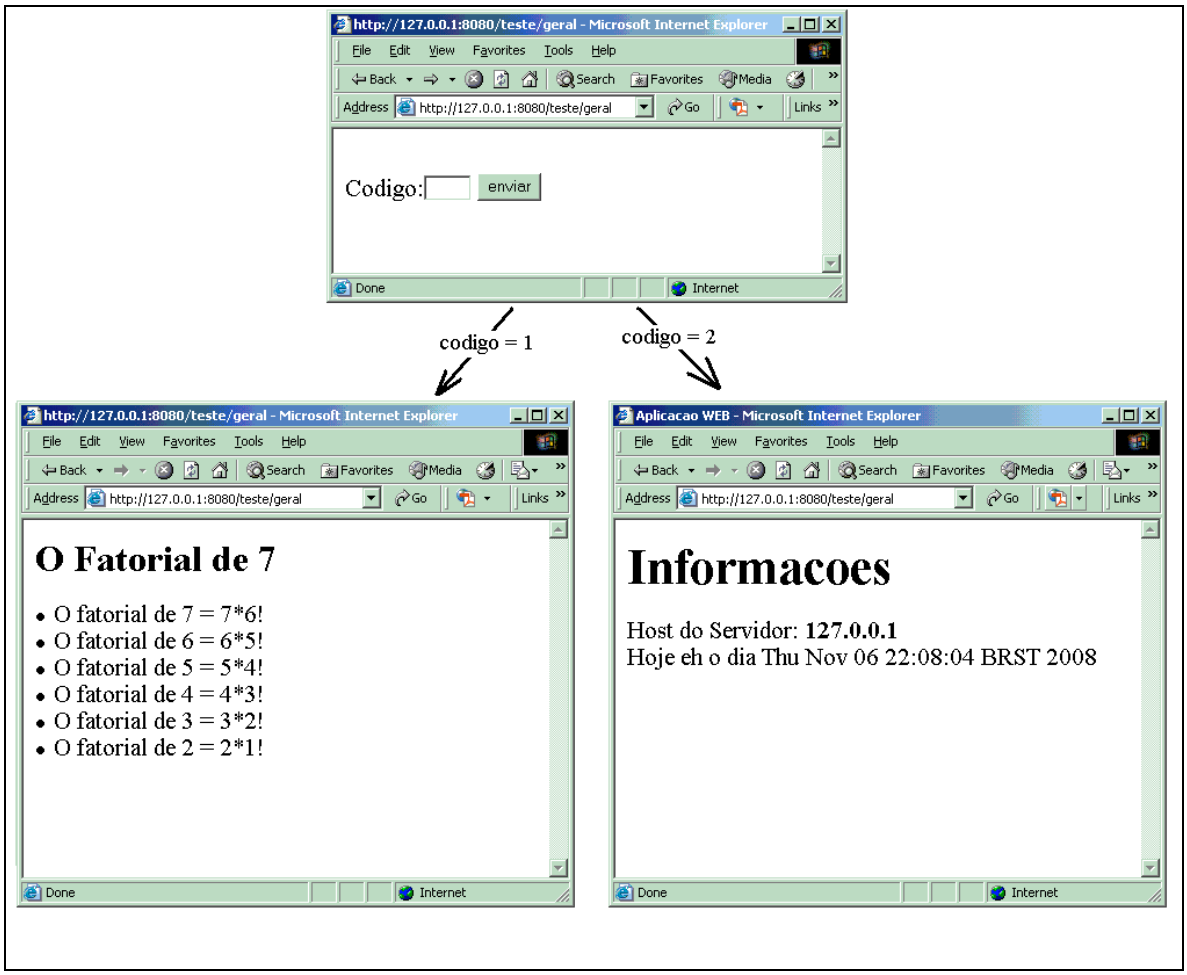

Figura 10-Execução do ServletGeral

13. Configuração do JCreator

Se durante a compilação de um servlet o JCreator forneceu a mensagem de erro da Figura 11, isso ocorreu porquê o JCreator não localizou a biblioteca `javax.servlet`.

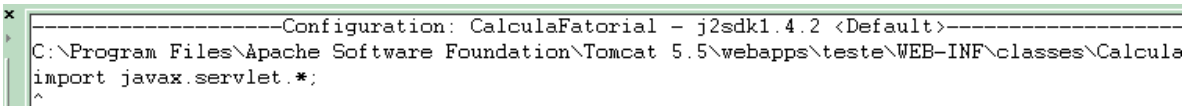

Figura 11-Mensagem de Erro do JCreator

Há a necessidade de informar ao ambiente o local onde a biblioteca está aramzenada. No elemento de menu `Project`, selecione a opção `Project Settings`. Na janela que será apresentada, selecione a guia `Required Libraries` e pressione o botão **`New...`**. A Figura 12 ilustra a nova janela que será apresentada.

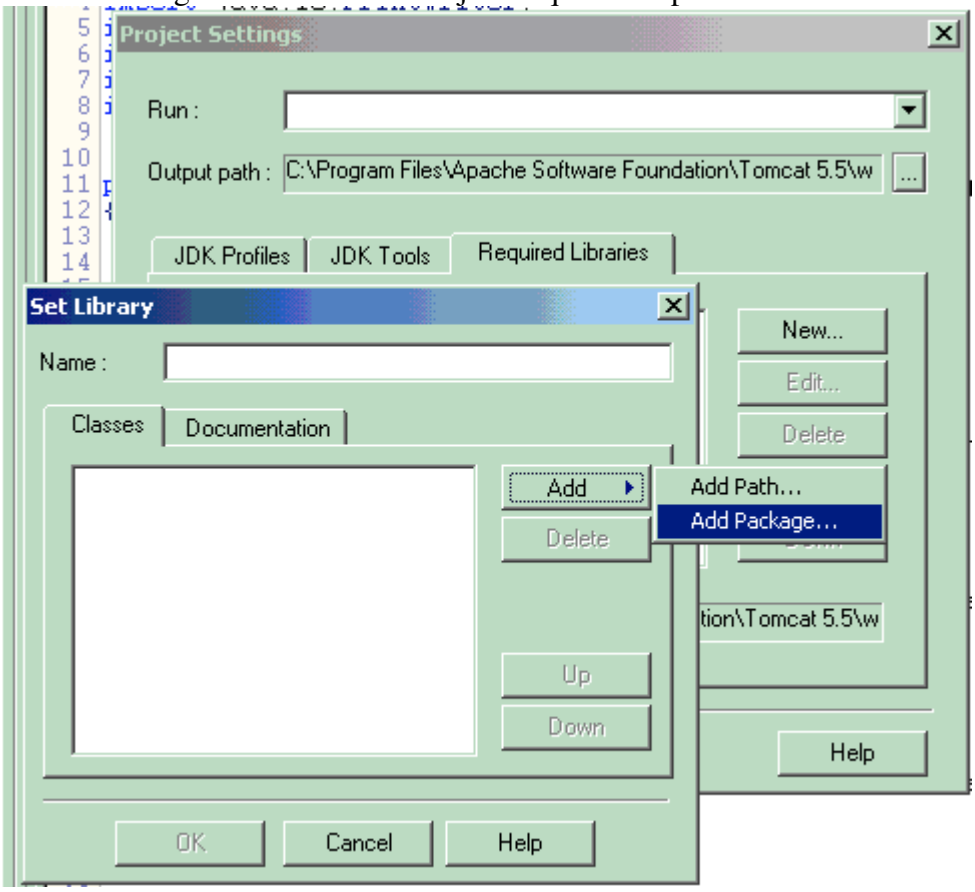

Figura 12-Interface para definição de Pacotes

Selecione a opção Add Package... conforme ilustrado na Figura 12 e em seguida indique o pacote `servlet-api.jar` armazenado no diretório **common\lib** do **TomCat**, conforme ilustrado nas Figuras 13 e 14.

A Figura 15 ilustra a finalização da configuração do JCreator para acessar pacotes externos.

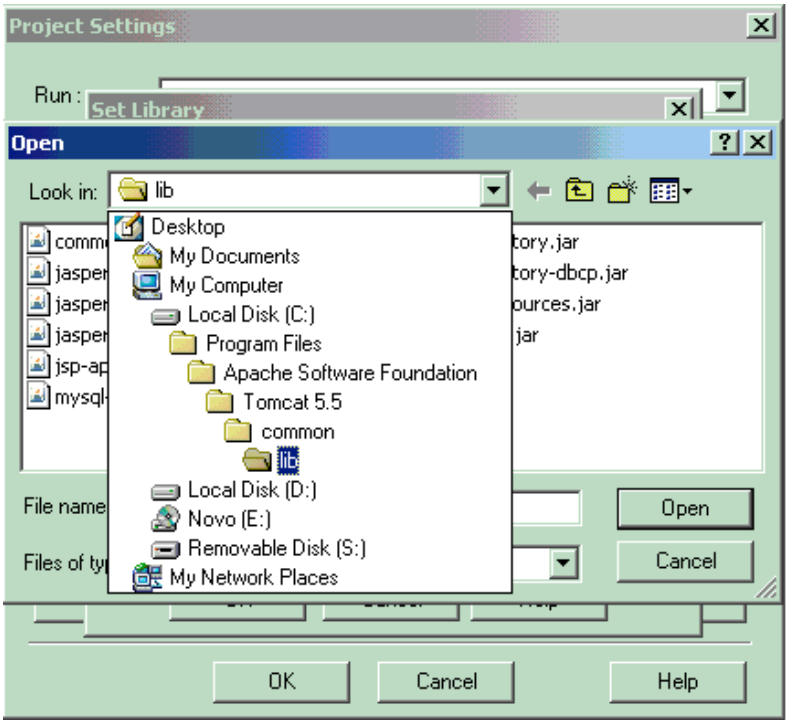

Figura 13-Diretório onde o pacote está localizado

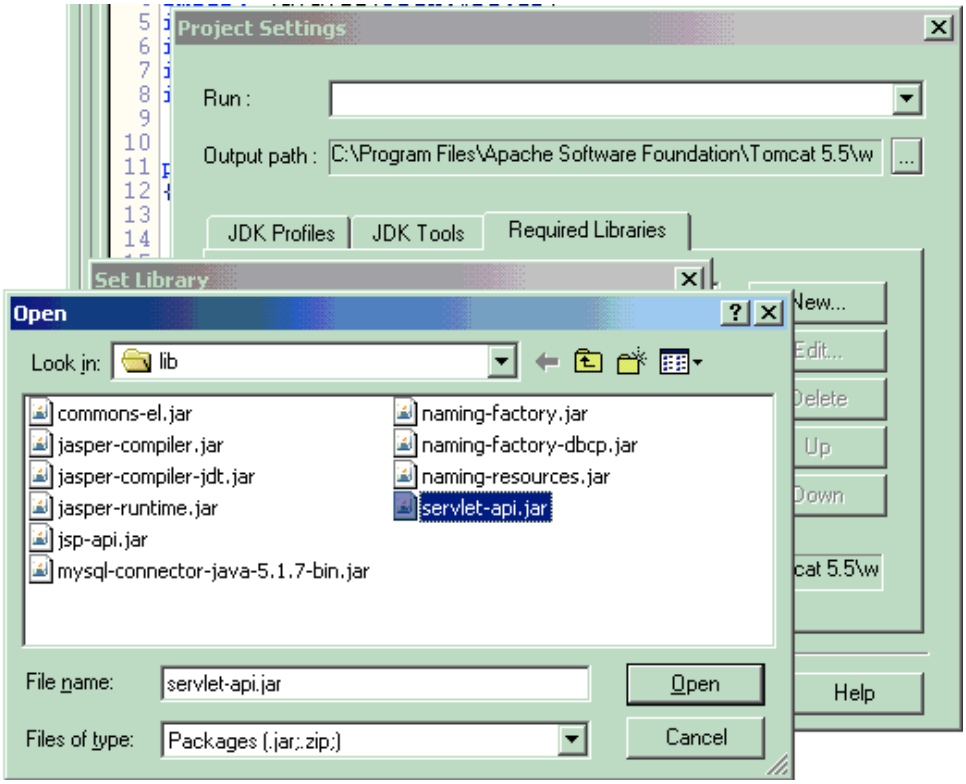

Figura 14-Pacote a ser selecionado

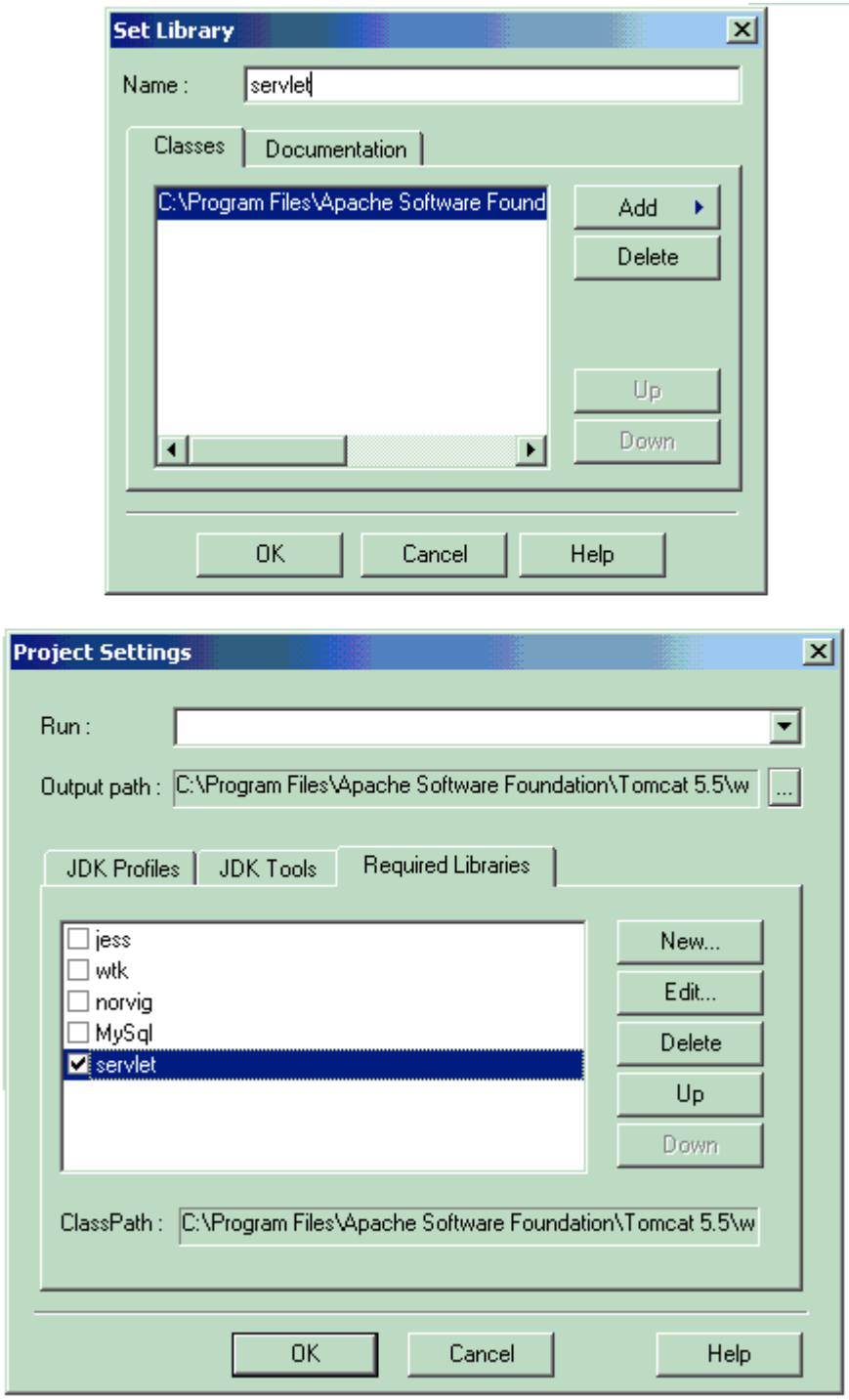

Figura 15-Finalização da definição de bibliotecas a serem utilizadas no projeto

14. O que estudar a partir de agora?

Durante esse curso, através de quatro tutoriais, foi introduzido alguns aspectos da tecnologia Java para a produção de sites dinâmicos. Outros aspectos e avanços tecnológicos

apresentados pela tecnologia Java são interessantes e recomenda-se ao programador Java conhecê-los.

Como recomendação de uma possível continuidade deste estudo, recomenda-se o seguinte livro:

Java EE 5: guia prático: Scriptles, Servlets e JavaBeans.
Autor: Kleitor Frankilnt Correa de Araújo.
Editora: Érica – SP, 2006.