FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA
Departamento de Informática

Introdução aos Sistemas e Redes de Computadores 2010/11

12ª Folha de Exercícios

Assunto: O protocolo HTTP: acesso a servidores Web usando Java; *servlet containers* e desenvolvimento de um aplicação Web usando *servlets*.

Aviso importante: o trabalho desenvolvido no ponto B constitui o último elemento de avaliação das aulas práticas da cadeira. Ver no final deste documento que material é preciso enviar para o docente do turno prático a que está inscrito.

Nesta aula proceder-se-á à programação de algumas aplicações em Java Pretende-se que faça o desenvolvimento do código Java do cliente e do servidor usando o ambiente especificada na ficha da aula prática anterior: interpretador de linha de comandos do Windows, compilador *javac*, editor de texto *Notepad++,* etc. Recorde que tem de actualizar a variável de ambiente PATH para incluir a directoria onde está o compilador de Java para *bytecode*;

```
PATH=%PATH%;C:\Program Files\Java\jdk1.6.0_07\bin
```

e que quando invoca o interpretador de bytescodes Java tem de garantir que a variável de ambiente CLASSPATH inclui a directoria corrente

```
set CLASSPATH=.
```

Para obter informação sobre os métodos e as classes relacionadas com *sockets* pode consultar os seguintes documentos "on line"

http://download.oracle.com/javase/tutorial/networking/sockets/
http://www.ibm.com/developerworks/java/tutorials/j-sockets/

Veja também no CLIP os slides apresentados nas aulas teóricas e teórico-práticas do mês de Janeiro de 2011.

A. Classes Java para acesso a servidores Web

Consulte a página "Working with URLs"

http://download.oracle.com/javase/tutorial/networking/urls/index.html

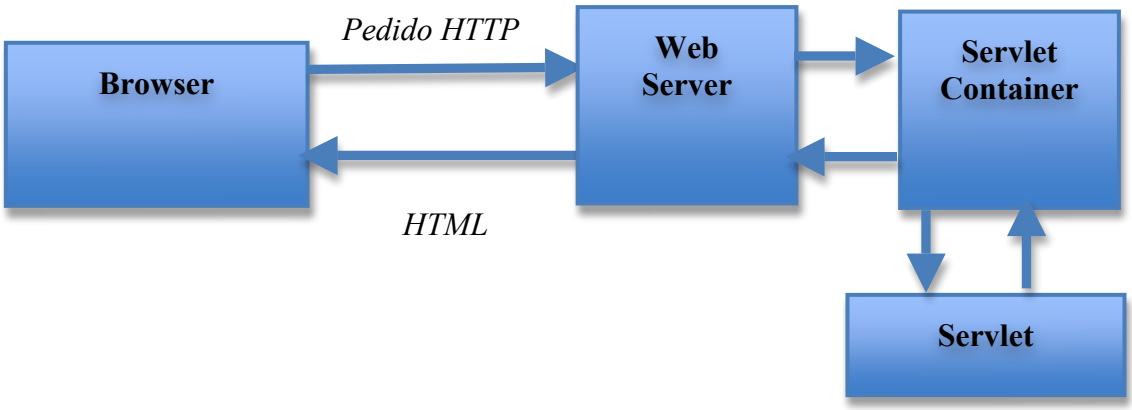
Na secção "Reading Directly from a URL" veja um exemplo de um programa Java que lê directamente o conteúdo de um URL e escreve esse conteúdo no terminal.

```
import java.net.*;
import java.io.*;
public class URLReader {
   public static void main(String[] args) throws Exception {
         URL yahoo = new URL("http://www.yahoo.com/");
         BufferedReader in = new BufferedReader(
                                 new InputStreamReader(
                                 yahoo.openStream()));

         String inputLine;

         while ((inputLine = in.readLine()) != null)
            System.out.println(inputLine);

         in.close();
   }
}
```

Experimente esse programa e modifique-o de forma a que o URL a consultar seja um argumento da linha de comando.

B. Desenvolvimento de uma aplicação usando *servlets*

Os *servlets* são programas Java que são executados por um servidor Web. Normalmente um servidor Web de uso geral como o Apache ou o IIS usa um outro servidor conhecido por *servlet container* para colocar em execução um programa em Java. A forma de invocar um *servlet* é referenciar o seu URL no browser. O *servlet container* coloca o programa Java em execução na máquina do servidor e entrega o output do programa ao browser. Tipicamente, o output do *servlet* usa a norma HTML.



Jetty

Neste trabalho[1], vamos usar o software *jetty*, que é de domínio público e de código aberto. O jetty está inteiramente escrito em Java e assegura simultaneamente as funções de *Web Server* e *Servlet. Container*. Faça download do ficheiro *zip* que contém a versão 7.2.2 do jetty a partir de

http://asc.di.fct.unl.pt/~pm/jetty-distribution-7.2.2.v20101205.zip , ou
http://download.eclipse.org/jetty/7.2.2.v20101205/dist/jetty-distribution-7.2.2.v20101205.zip

Expanda o ficheiro zip numa directoria à sua escolha. Passará a existir uma directoria *jetty-distribution-7.2.2v20101205*. Abra um terminal e faça cd para a directoria para onde foi extraído o *jetty*. Para colocar o jetty em execução faça

```
java –jar start.jar
```

O servidor arranca e escreve algumas mensagens no terminal. Lance um Web browser e faça acesso ao URL http://localhost:8080 . Se aparecer uma página com o título "Welcome to Jetty 7" tudo correu bem. Para parar o *jetty*, faça control-C no terminal.

Exemplo de um servlet e da sua colocação em operação

O servidor *jetty* tem uma directoria chamada *webapps* que tem várias subdirectorias uma para cada aplicação Web disponibilizada (ou seja, por cada *servle*t). Para criar uma aplicação Web acessível através de um *servlet*, teremos de criar uma directoria com o nome da aplicação. Nessa directoria vamos colocar subdirectorias e ficheiros de acordo com o exemplo dado a seguir:

1. Dentro de *webapps* crie uma directoria chamada *hello*. A aplicação que estamos a criar será invocada quando escrevermos no browser http://localhost:8080/hello
2. Dento da directoria *hello* crie uma directoria *WEB-INF*. Dentro da directoria *WEB-INF* crie uma directoria chamada *classes* . Esta directoria vai conter o código da JVM que é executado quando o *servlet* é invocado.
3. Na directoria hello crie um ficheiro chamado *índex.html*. Esta é página que se obtém quando se faz http://localhost:8080/hello O ficheiro deve conter o seguinte:

```
<html>
<head><title>Example Web Application</title></head>
<body>
<p>This is a static document with a form in it.</p>
<form method="POST" action="servlet"/>
<input name="field" type="text" />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

[1] Esta secção é uma adaptação da página *http://www.seas.upenn.edu/~cis330/jetty.html*

4. Vamos agora fazer o código do *servlet*. O *servlet* vai tratar os dados contidos nos *forms* presentes na página criada em 3. Crie um ficheiro *HelloServlet.java* com o seguinte código:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloServlet extends HttpServlet {

  protected void doGet(HttpServletRequest req,  HttpServletResponse resp)
      throws ServletException, IOException     {

      String q = req.getParameter("q");
      PrintWriter out = resp.getWriter();
      out.println("<html>");
      out.println("<body>");
      out.println("The paramter q was \"" + q + "\".");
      out.println("</body>");
      out.println("</html>");
   }

   protected void doPost(HttpServletRequest req, HttpServletResponse resp)
         throws ServletException, IOException     {

     String field = req.getParameter("field");
     PrintWriter out = resp.getWriter();
     out.println("<html>");
     out.println("<body>");
     out.println("You entered \"" + field + "\" into the text box.");
     out.println("</body>");
     out.println("</html>");
  }
}
```

5. Para compilar esta classe precisamos das bibliotecas de *servlets*. Estas bibliotecas estão na directoria do jetty. Faça cd para a directoria onde está *HelloServlet.java* e dê o comando:

```
javac  –cp DIRECTORIA_DO_JETTY/lib/servlet-api2.5.jar HelloServlet.java
```

Copie o ficheiro *HelloServlet.class* para *hello/WEB-INF/classes*

6. Para indicar ao *jetty* que é o código criado em 5 que deve ser chamada para tratar o pedido http://localhost:8080/hello é preciso criar o ficheiro *web.xml* na directoria *WEB-INF* . O conteúdo de *web.xml* deve ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <display-name>Example</display-name>
    <!-- Declare the existence of a servlet. -->
    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>HelloServlet</servlet-class>
    </servlet>

    <!-- Map URLs to that servlet. -->
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/servlet</url-pattern>
    </servlet-mapping>
 </web-app>
```

7. Lance novamente o *jetty*. Se introduzir no seu browser http://localhost:8080/hello o que acontece?

Repare que há dois métodos na classe do servlet, *doGet()* e *doPost()*, que correspondem aos dois tipos de pedidos HTTP vistos na aula teórico-prática.

Trabalho a desenvolver

Desenvolva um servlet chamado *Factorial* deve poder ser invocado a partir de um *browser*. O funcionamento deve corresponder aos seguintes passos:

1. Inserindo no browser o URL http://localhost:8080/Factorial deve ser apresentada um página com um *form* em que há lugar para introduzir um número N e um botão para submeter o pedido.
2. O browser invoca a operação no *servlet* enviando ao servidor Web o comando GET /CGI-BIN/Factorial?numero=32 (supondo que o utilizador colocou 32 no *form*)
3. O jetty recebe o pedido e lança o *servlet Factorial* em acção; previamente definiu no seu ambiente *numero=32*
4. O *servlet* gera o conteúdo de uma página HTML: essa página vai contar o factorial do número que foi enviado pelo browser. O *servlet* envia a página HTML pelo seu canal *standard* de saída
5. O *jetty* encaminha os caracteres recebido do *servlet* para o browser
6. O browser mostra a página enviada

O que tem de ser feito nos vários passos:

1. Escrever o conteúdo da página HTML e colocá-la na directoria *.../webapps/Factorial* com o nome *index.html*; isto pode ser testado usando o *browser*
2. Nada é preciso fazer aqui. Se quiser testar sem usar o *browser* pode usar uma versão alterada do programa que foi apresentado na parte A.
3. Para que o *servlet* seja activado é preciso colocar na directoria *WEB-INF* um ficheiro *web.xml* com o conteúdo adequado
4. Neste ponto é preciso escrever o código Java do servlet. Uma vez compilado o código Java é preciso colocá-lo na directoria *.../webapps/Factorial/WEB-INF/classes*
5. Nada é preciso fazer
6. Também nada há a fazer

Material a enviar

Coloque num ficheiro zip os seguintes elementos:

- O código Java do servlet *FactorialServlet.java*
- O conteúdo da directoria *webapps/Factorial*

Envie o ficheiro para o docente do turno prático a que está inscrito.