

Introdução a Servlets

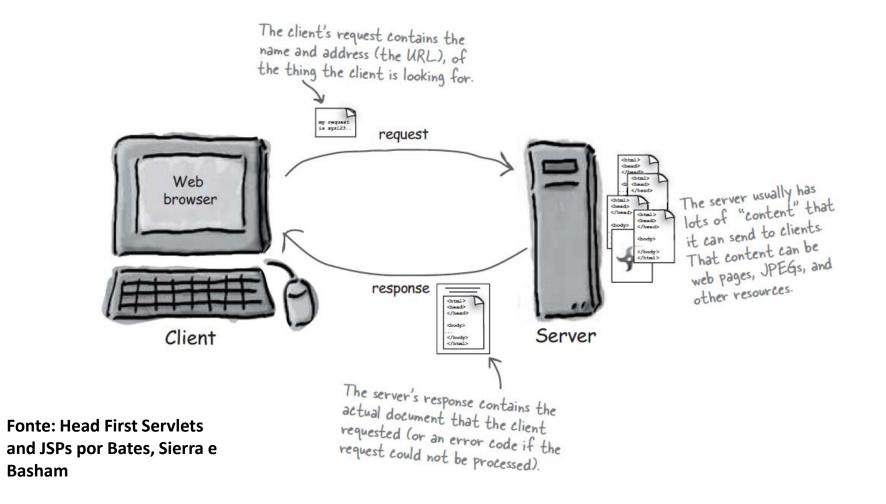
Laboratório de Programação (5COP011) Prof. Bruno Bogaz Zarpelão

Objetivos

- Conceitos básicos:
 - interação cliente-servidor;
 - HTTP;
 - páginas estáticas x páginas dinâmicas;
 - Servlets
- Primeiro servlet.



Interação cliente servidor





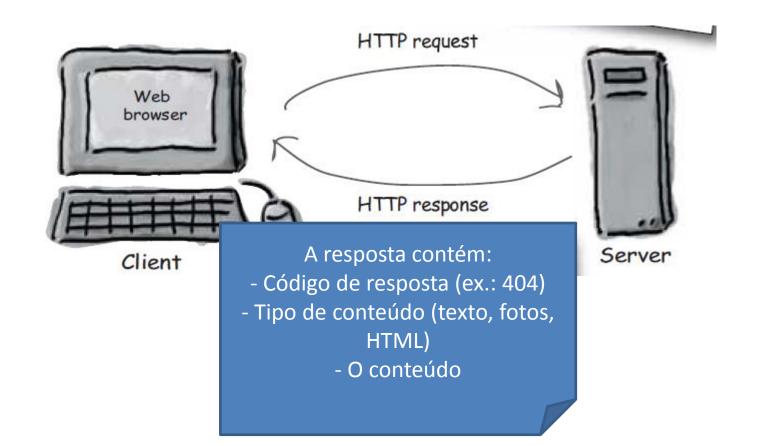
Interação cliente servidor

- O cliente e o servidor falam dois "idiomas":
 - HTML: o conteúdo retornado pelo servidor é composto por um conjunto de instruções em HTML;
 - HTTP: tanto as requisições quanto as respostas são sempre escritas no formato do HTTP;



HTTP

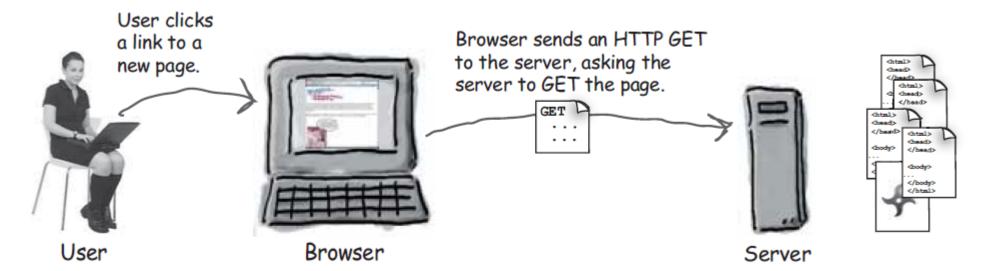
A requisição contém:
- método HTTP (get, post, etc)
-Objeto a ser acessado (URL)
- parâmetros





HTTP - Método get

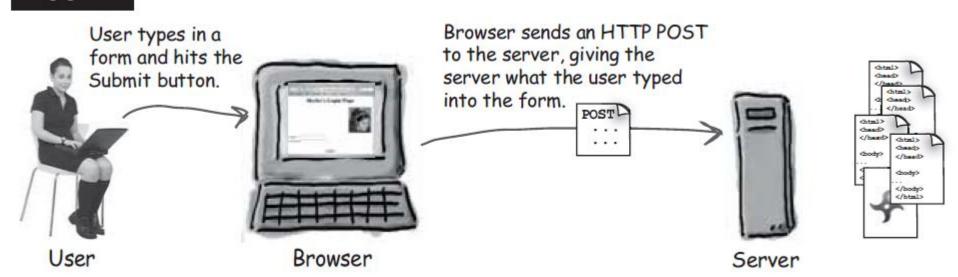
GET





HTTP – Método post

POST





URL (Uniform Resource Locator)

Protocol: Tells the server which communications protocol (in this case HTTP) will be used. Port: This part of the URL is optional. A single server supports many ports. A server application is identified by a port. If you don't identified by a port in your URL, then specify a port in your URL, and as luck port 80 is the default, and as luck would have it, that's the default port for web servers.

Resource: The name of the content being requested. This could be an HTML page, a servlet, an image, PDF, music, video, or anything else the server feels like serving. If this optional part of the URL is left out, most web servers will look for index.html by default.

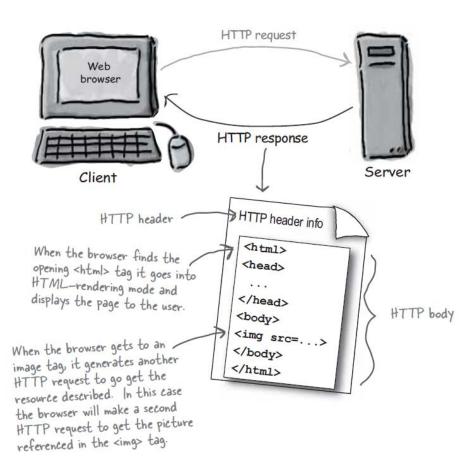
http://www.wickedlysmart.com:80/beeradvice/select/beer1.html

Server: The unique name of the physical server you're looking for. This name maps to a unique IP address. IP addresses are numeric and take the form "xxx. yyy.zzzaaa". You can specify an IP address here instead of a server name, but a server name is a lot easier to remember.

Path: The path to the location, on the server, of the resource being requested. Because most of the early servers on the web ran Unix, Unix syntax is still used to describe the directory hierarchies on the web server.



HTML



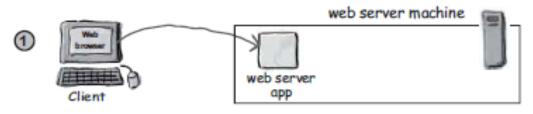


Páginas estáticas, páginas dinâmicas e CGI

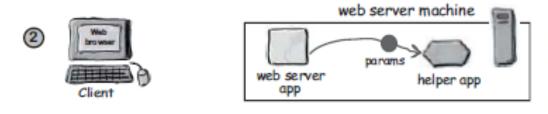
- O HTML, o HTTP e os servidores Web "puros" (ex.: Apache HTTP Server) só conseguem trabalhar com páginas estáticas.
- Para ter conteúdo dinâmico, precisamos de software extra (CGI – Common Gateway Interface):
 - PHP, Pearl, C, etc.



CGI



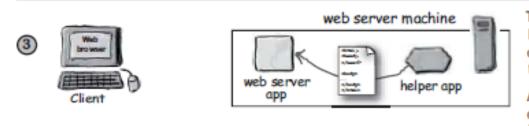
User clicks a link that has a URL to a CGI program instead of a static page.



Web server application "sees" that the request is for a helper program, so the web server bunches and runs the program. The web server app sends along any parameters from a GET or POST.

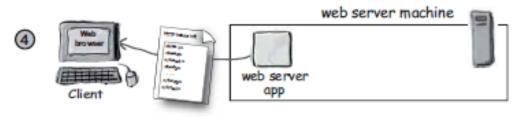


CGI



The helper app constructs the brand new page (that has the current date inserted) and sends the HTML back to the server.

As far as the web server is concerned, the HTML from the helper app is a static page.



The helper application is shut down, and the client gets back an HTML page that has the current date as part of its now-static content.



Java e aplicações Web

- Também podemos construir páginas dinâmicas (aplicações Web) utilizando Java.
- Para executar uma aplicação Web Java, precisamos de um container Web.
- O container Web é uma aplicação executada em conjunto com o servidor HTTP para trabalhar com conteúdos dinâmicos.



Java e aplicações Web

- A aplicação Web que desenvolvemos na linguagem Java é implantada em um container Web.
- O container Web executa essa aplicação Java para tratar requisições HTTP e respondê-las.
- O Apache Tomcat é um exemplo de container Web bem tradicional.



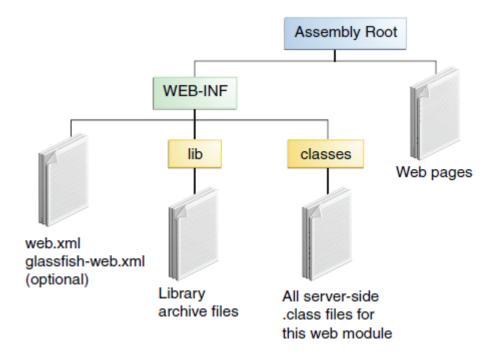
Estrutura de uma aplicação Web Java

- Para implantar uma aplicação Web Java em um container Web, algumas regras devem ser seguidas.
- Os códigos fonte e demais arquivos referentes à aplicação Web devem ser colocados em um arquivo no formato .war (Web Archive).



Estrutura de uma aplicação Web Java

O conteúdo do Web Archive deve seguir essa estrutura:



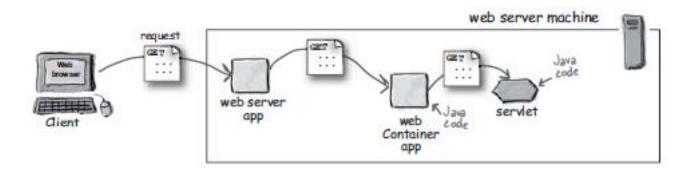


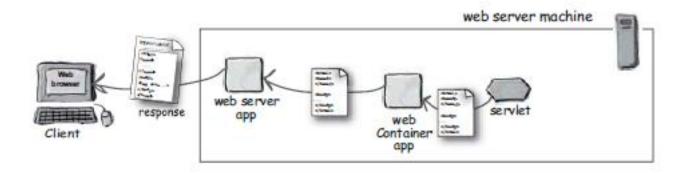
Servlets

 Os Servlets são uma solução presente no J2EE que permitem a construção de aplicações Web com Java.



Servlets

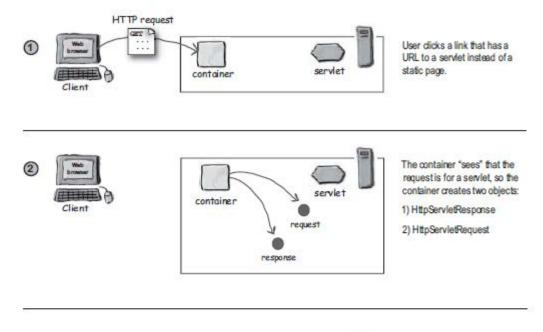




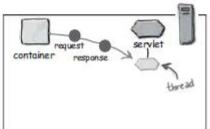




Manipulação de um request





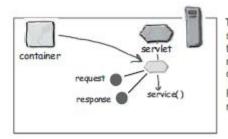


The container finds the correct servlet based on the URL in the request, creates or allocates a thread for that request, and passes the request and response objects to the servlet thread.



Manipulação de um request

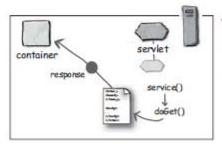




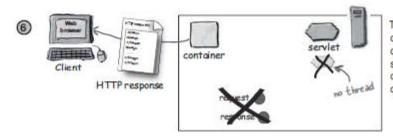
The container calls the service's service() method. Depending on the type of request, the service() method calls either the doGet() or doPost() method.

For this example, we'll assume the request was an HTTP GET.





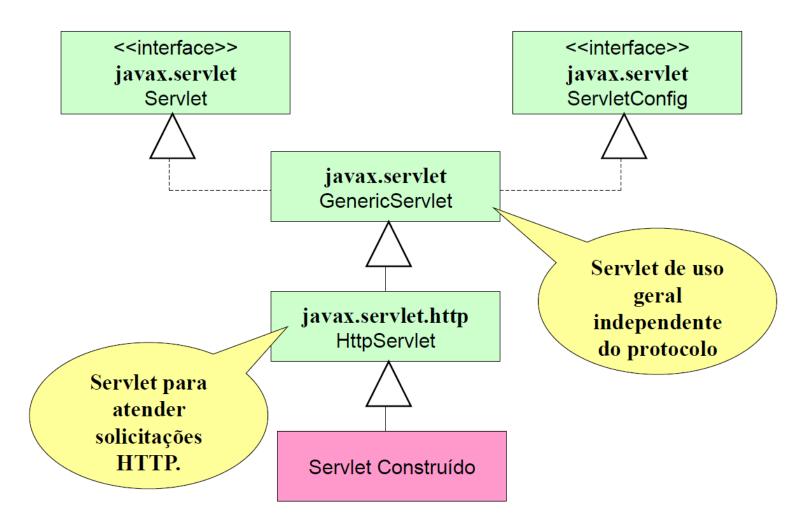
The doGet() method generates the dynamic page and stuffs the page into the response object. Remember, the container still has a reference to the response object!



The thread completes, the container converts the response object into an HTTP response, sends it back to the client, then deletes the request and response objects.



Servlets





Exemplo de código de Servlet

```
SERVLET EXAMPLE
                                                           Servlets are not part of the standard SDK,
                                                                    they are part of the J2EE
import java.io.*;
                                                              Servlets normally extend HttpServlet
import javax.servlet.*;
import javax.servlet.http.*;
public class ServWelcome extends HttpServlet
                                                              The response to be sent to the client
  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws IOException, ServletException
                                                            Details of the HTTP request from the client
    response.setContentType("text/html");
                                                             Set the response type to text/html (this is
    PrintWriter out = response.getWriter();
                                                                             normal)
    out.println("<HTML>");
    out.println("<HEAD><TITLE>First Servlet Program</TITLE></HEAD>");
                                                                                  This HTML text is
    out.println("<BODY>");
                                                                                   sent to the client
    out.println("<H1>Welcome to Servlets</H1>");
    out.println("</BODY>");
                                                                                 Don't forget to close
    out.println("</HTML>");
                                                                                 the connection with
    out.close(); -
                                                                                      the client
```



Servlets

- Os métodos doGet e doPost são usados para processar requisições HTTP do tipo get e post, respectivamente;
- Para fazer o mapeamento entre a URL e o Servlet que deve ser executado, nós utilizamos um arquivo conhecido como deployment-descriptor. O nome deste arquivo é "web.xml";



Mapeamento de uma URL para o Servlet

Primeiro, declara-se o servlet, indicando a classe e dando um nome:

```
<servlet>
    <servlet-name> primeiraServlet </servlet-name>
    <servlet-class> br.uel.HelloWorldServlet </servlet-class>
</servlet>
```

Depois, define-se o padrão de URL que deverá ser tratado por esse servlet:

```
<servlet-mapping>
    <servlet-name> primeiraServlet </servlet-name>
        <url-pattern> /oi </url-pattern>
</servlet-mapping>
```



Mapeamento de uma URL para o Servlet

 Também podemos fazer isso com as annotations do Java (à partir do J2EE 6/Servlets 3):

```
@WebServlet(name = "primeiraServlet", urlPatterns = {"/oi"})
public class HelloWorldServlet extends HttpServlet {
...
}
```

