

Aula 4



Objetivos

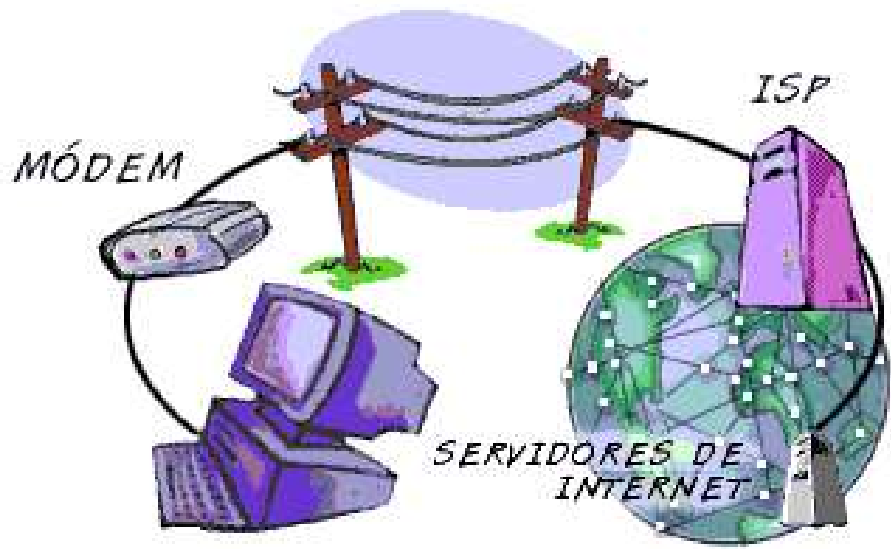
- **Conteúdo dinâmico na internet.**

Conteúdo dinâmico na internet

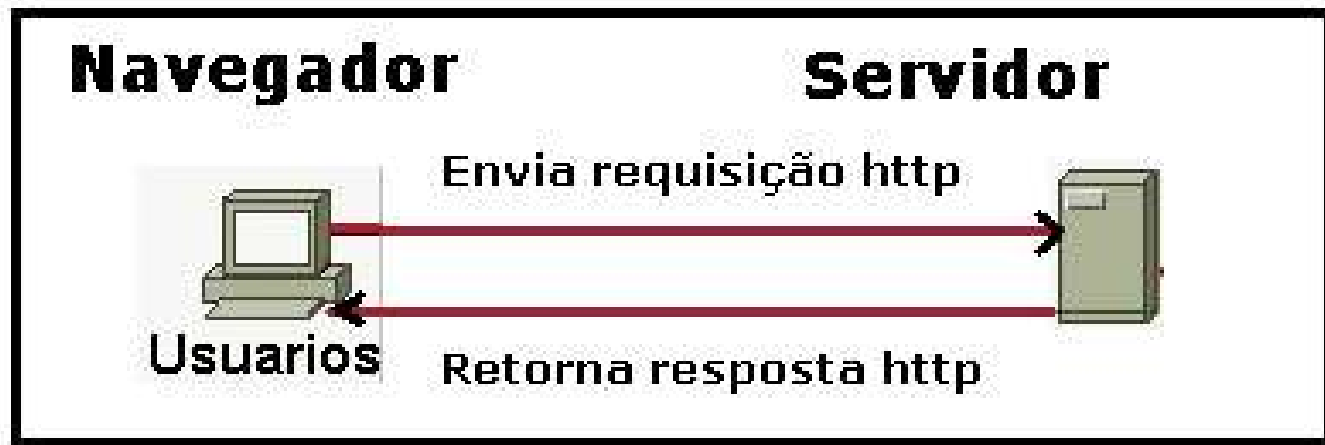


Para uma solicitação da Web mais simples, um navegador solicita um documento HTML e o servidor Web encontra o arquivo correspondente e devolve. Se o documento HTML incluir qualquer imagem, o navegador por sua vez irá submeter solicitações os documentos de imagens também. Como descrito aqui, todas essas solicitações são para arquivos estáticos. Ou seja, os documentos que são solicitados nunca mudam dependendo de quem solicitou, quando eles foram solicitados, ou que(caso haja) parâmetros adicionais foram incluídos com a solicitação.

Conteúdo dinâmico na internet



TERMINAL-Navegador



Conteúdo dinâmico na internet



No entanto, a maioria dos dados fornecidos através da Web hoje tem uma natureza dinâmica. Como exemplo, extrato bancário, mensagens, cotações de ações, geração de contas de luz e água, etc.

O conteúdo da Web dinâmico, então, exige que o servidor da Web faça algum processamento adicional da solicitação correspondente, a fim de gerar uma resposta personalizada.

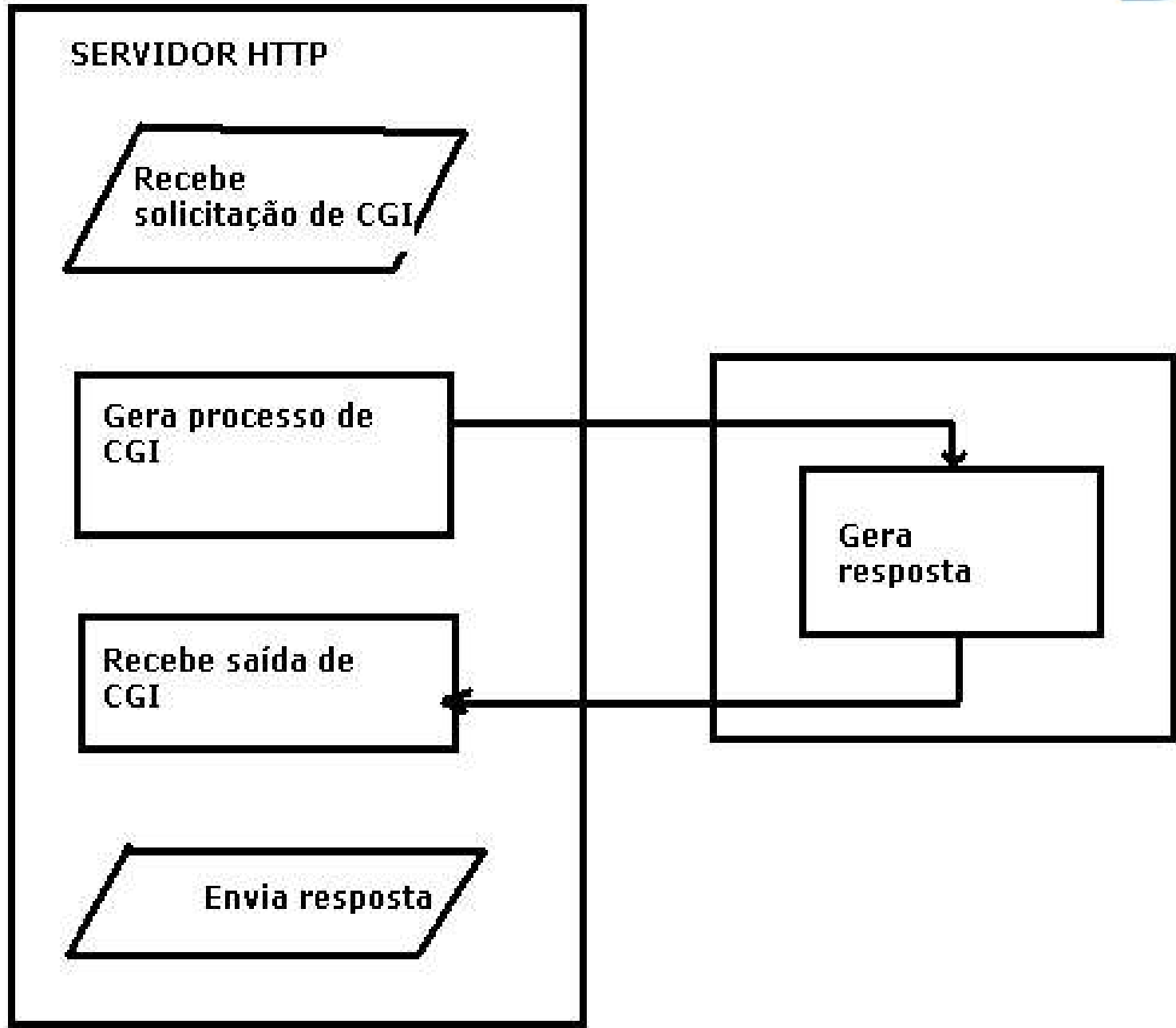
Os servidores de HTTP mais antigos não incluíam qualquer mecanismo embutido para gerar respostas dinamicamente. Ao invés disso, foram fornecidas interfaces para chamar outros programas para utilizar solicitações no conteúdo runtime. O primeiro padrão para conteúdo da Web dinâmico se baseava na Common Gateway Interface, ou CGI, que especificava um mecanismo para servidores da Web passarem as informações da solicitação para programas externos, que eram então, rodados

Conteúdo dinâmico na internet



pelo servidor Web para gerar resposta no tempo de execução. A linguagem popular para escrever programas de CGI, mas os códigos de CGI podem ser escritos em qualquer linguagem que possa ser chamada como um programa independente pelo servidor HTTP.

Conteúdo dinâmico na internet



Conteúdo dinâmico na internet



Outras tecnologias como:

- ColdFusion;
- Active Server Pages;
- Server-Side JavaScript;
- PHP.

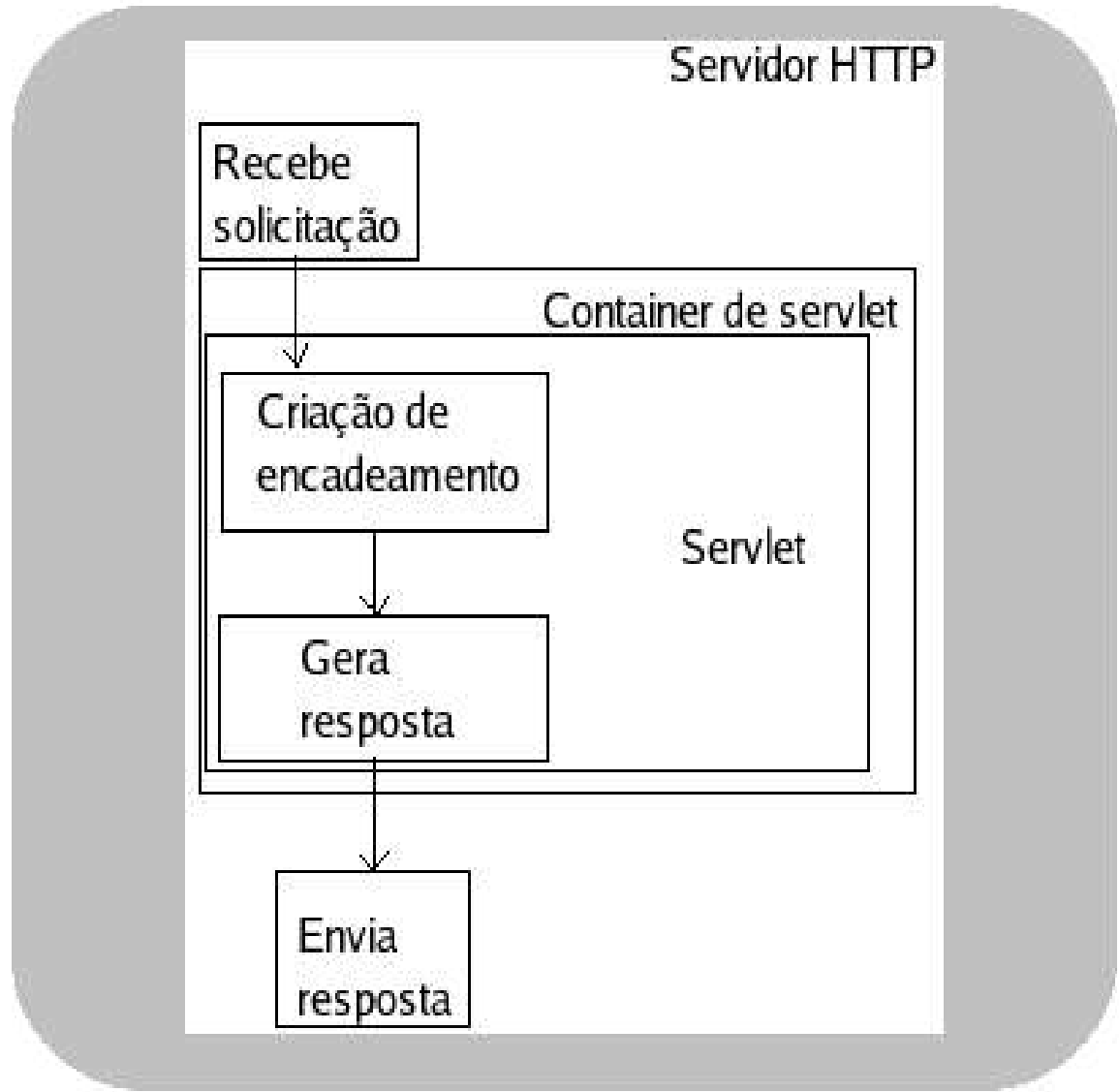
Conteúdo dinâmico na internet



Devido à importância da geração de conteúdo dinâmico para desenvolvimento da Web, foi natural que a SUN propusesse extensões para Java em seu domínio. Da mesma forma que a SUN introduziu applets como pequenas aplicações baseadas em Java para adicionar funcionalidade interativa aos navegadores da Web, em 1996, a SUN introduziu servlets como pequenas aplicações baseadas em Java para adicionar funcionalidade dinâmica a servidores da Web.

Diferentemente dos programas tradicionais de CGI que necessitam da criação de um novo processo para tratar nova solicitação, todos os servlets associados com um servidor da Web rodam dentro de um único processo. Este processo roda uma Java Virtual Machine(JVM), que é o programa específico de plataforma para rodar programas de Java compilados (compatíveis com várias plataformas).

Conteúdo dinâmico na internet



Conteúdo dinâmico na internet



Ao invés de criar um processo para cada solicitação, a JVM, cria um encadeamento para tratar de cada solicitação de servlet. Os encadeamentos de Java tem muito menos overhead do que os processos completos dentro da memória do processador já alocada pela JVM, tornando a execução do servlet consideravelmente mais eficiente do que o processamento de CGI. Já que a JVM persiste além de uma única solicitação, os servlets também podem evitar muitas operações demoradas, como conexão a um banco de dados, ao compartilhá-los entre todas as solicitações. Ao mesmo tempo, pelo fato dos servlets serem escritos em Java, eles se aproveitam de todos os benefícios da plataforma de Java básica; um modelo de programação orientado a objetos, gerenciamento automático de memória, portabilidade compatível com várias plataformas e acessam a rica coleção de APIs de Java agora disponível para acessar banco de dados, servidores de diretório, recursos de rede e assim por diante.

Conteúdo dinâmico na internet



ATIVIDADE

Conteúdo dinâmico na internet



Aplicações Ações

New Web Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

Source Structure: ▼

Add to Enterprise Application:

Server: ▼

J2EE Version: ▼

Context Path:

Set as Main Project

< Back Next > Finish Cancel Help

Turning on modules...done.

Terminal (3) javancado_ Gerenciado Java Maga: processo_s /home/carlc NetBeans I

Conteúdo dinâmico na internet



Aplicações Ações

NetBeans IDE 4.1 Beta - Aula04

File Edit View Build Run Refactor Versioning Tools Window Help

Close Unrelated Files Show Related Files

Files Runtime

- WEB-INF
 - web.xml
 - index.jsp
- Source Packages (circled)
- <default package>
- Test Packages
- Configuration Files

Navigator

<Not supported document>

Output HTTP Monitor

Terminal (3) javancado_ Gerenciado Java Maga: processo_s /home/carlc NetBeans I

Conteúdo dinâmico na internet



Criar um pacote de nome `br.javanoroeste`

Conteúdo dinâmico na internet



Aplicações Ações

NetBeans IDE 4.1 Beta - Aula04

File Edit View

New Folder

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Folder Name:

Project:

Parent Folder:

Created Folder:

< Back Next > Finish Cancel Help

Terminal (3) javancado_ Gerenciado Java Maga processo_s /home/carlc NetBeans I

Conteúdo dinâmico na internet



Copie este código:

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
// TODO output your page here
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet exemplo da aula 4</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Servlet exemplo da " +
request.getContextPath () + "</h1>");
out.println("</body>");
out.println("</html>");

out.close();
```

para...

Conteúdo dinâmico na internet



Apague a linha "`processRequest(request, response)`",
cole aqui:

```
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left
/** Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    PrintWriter out = response.getWriter();
    // TODO output your page here
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet exemplo da aula 4</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Servlet exemplo da " + request.getContextPath () + "</h1>");
    out.println("</body>");
    out.println("</html>");

    out.close();
}
```

Conteúdo dinâmico na internet

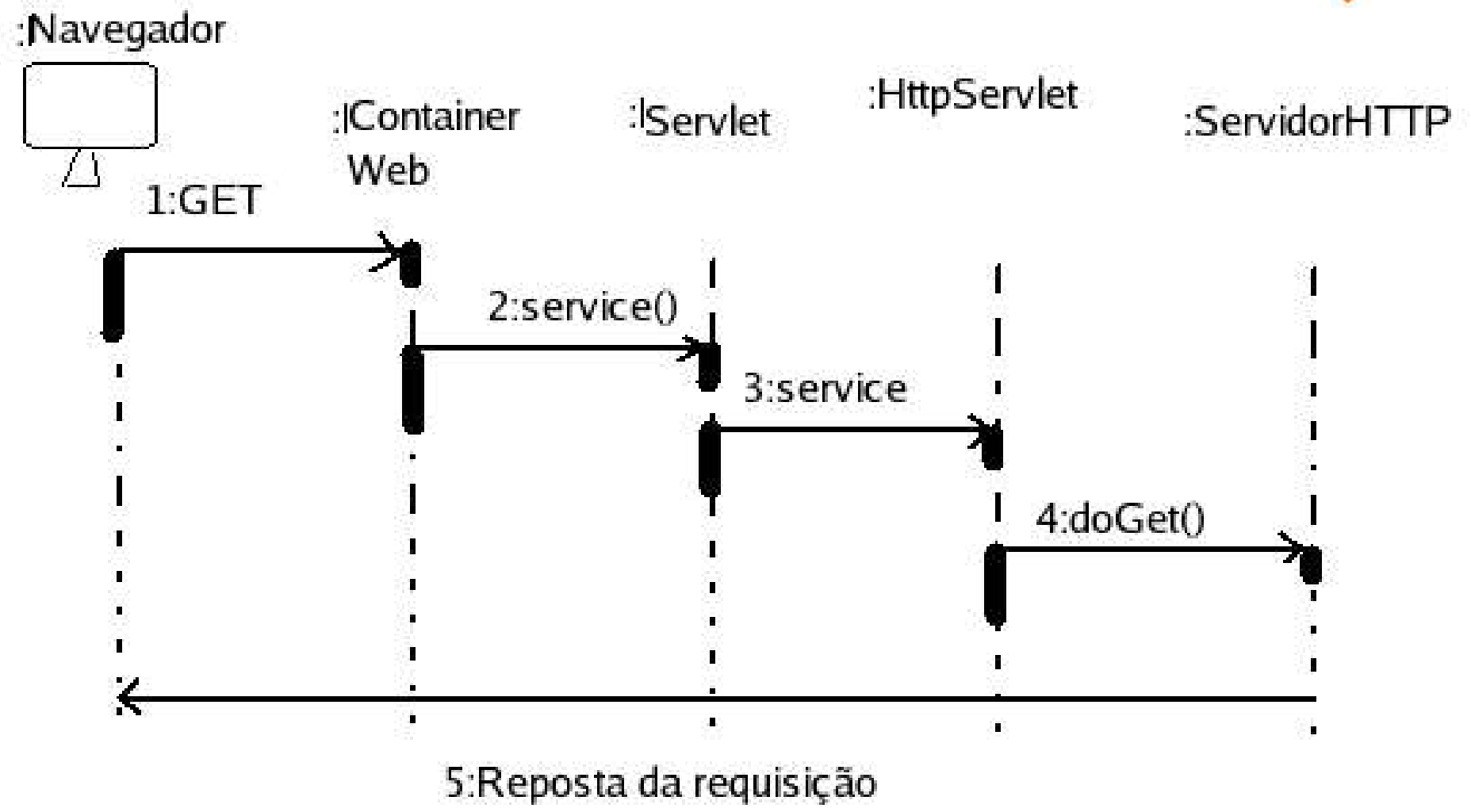


As classes de servlet são responsáveis pela definição dos métodos de serviço para tratar dos vários tipos de solicitações de HTTP, incluindo um método **doGet()** para tratar de solicitações **GET** de HTTP e um método **doPost ()** para tratar de solicitações **POST** de HTTP.

Em nosso exemplo de servlet, apenas retorna uma página HTML simples, contendo a frase "Servlet exemplo da" mais o retorno do método que contém o valor do "path" no qual foi requisitado o servlet, em nosso caso "/Aula04".

No método `doGet()` existem dois objetos como argumentos (**HttpServletRequest request**, **HttpServletResponse response**), necessários para a requisição e a resposta, que em qualquer informação que seja necessária para gerar resposta de ser obtida por meio da requisição, em qualquer que seja a resposta, ela deve ser entregue a métodos do objeto de resposta.

Conteúdo dinâmico na internet



Conteúdo dinâmico na internet



Código	Significado
200	OK
400	Requisição mal formada
403	Acesso negado
404	Recurso(página, imagem etc.) inexistente
500	Erro no servidor (requisição não pode ser tratada)



Elementos básicos do descritor Web

Toda aplicação Web executada num container, é configurada através de um descritor(Web descriptor), representando pelo arquivo *web.xml*, contido no diretório *WEB-INF* da aplicação. É o descritor que são definidos os servlets presentes na aplicação, o mapeamento dos servlets e URLs, as páginas de erros da aplicação, os filtros e listeners disponíveis e vários outros aspectos de uma aplicação Web.



Elementos básicos do descritor Web

Um exemplo de *web.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app                                version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

<!--elementos de web-app, em qualquer ordem-->

</web-app>
```



Elementos básicos do descritor Web

No caso de servlets, estamos interessados em dois elementos:

<servlet>

<servlet-mapping>

O descritor deve conter um elemento **<servlet>** para cada servlet disponível na aplicação, com os sub-elementos **<servlet-name>** e **<servlet-class>** definindo nome do servlet (que deve ser único dentro da aplicação) e a classe que o implementa. Exemplo:



Elementos básicos do descritor Web

```
<servlet>
  <servlet-name>
    ServidorHTTP
  </servlet-name>
  <servlet-class>
    br.javanoroeste.ServidorHTTP
  </servlet-class>
</servlet>
  <servlet-mapping>
    <servlet-name>ServidorHTTP</servlet-name>
    <url-pattern>/ServidorHTTP</url-pattern>
  </servlet-mapping>
```



Elementos básicos do descritor Web

Redirecionamento para um servlet

No arquivo web.xml apague estas linhas:

```
<welcome-file>  
  index.html  
</welcome-file>
```

Cole estas linhas:

```
<servlet-mapping>  
  <servlet-name>ServidorHTTP</servlet-name>  
  <url-pattern>/index.html</url-pattern>  
</servlet-mapping>
```



Referências Bibliográficas :

FIELDS, D.K.; KOLB, M.A. **Desenvolvendo na Web com JavaServer Pages**. Riode Janeiro: Ed. Ciência Moderna, 2000.

Leme, F. **Pente Fino-Servlets Parte 1: Conceitos e técnicas básicas**. JavaMagazine Edição 18.

Lozano, F. **Java Livre-Servlets no Tomcat 5- Aplicações Web MVC em Java**. JavaMagazine Edição 20.