Java Web

Felipe Sobreira Cassimiro

HTTP - Hypertext Transfer Protocol

Protocolo de comunicação, da camada de aplicação, utilizado para estabelecer uma troca de mensagens entre cliente e servidor.


```
GET /pagina.htm HTTP/1.1
Accept: image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-EncGET oding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible;
MSIE 5.01; Windows NT)
Host: hypothetical.ora.com
Connection: Keep-Alive
```

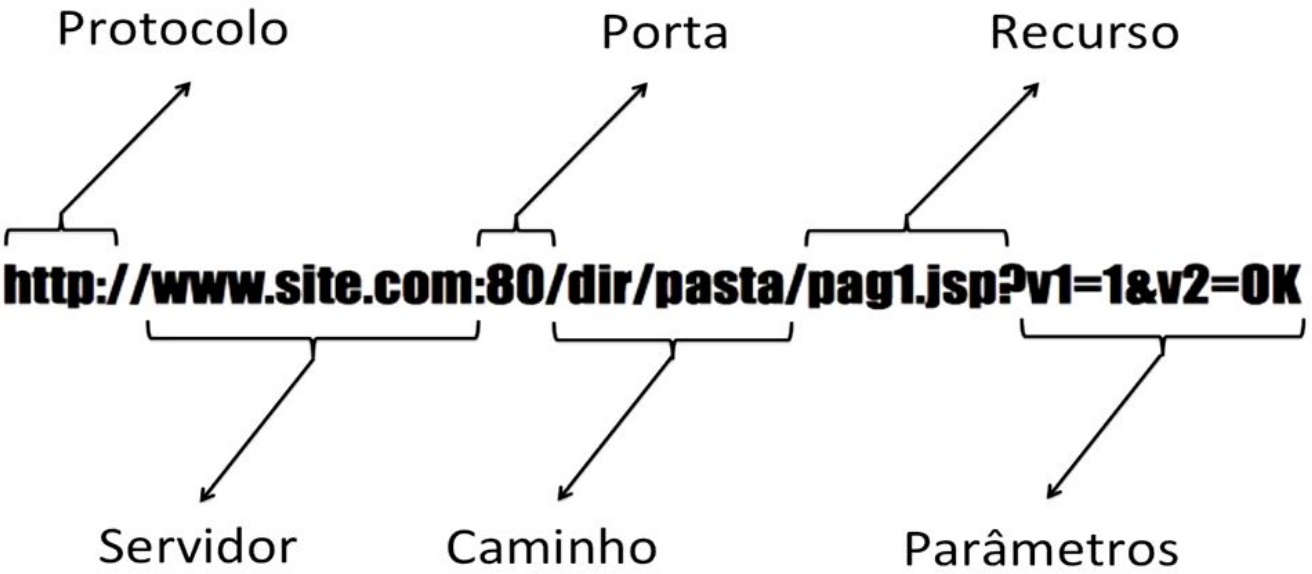
- **Método HTTP**
- **URL**
- **Parâmetros**


```
HTTP/1.1 200 OK
Date: Mon, 06 Dec 1999 20:54:26 GMT
Server: Apache/1.3.6 (Unix)
Last-Modified: Fri, 04 Oct 1996
14:06:11 GMT
ETag: "2f5cd-964-381e1bd6"
Accept-Ranges: bytes
Content-length: 327
Connection: close
Content-type: text/html

<html>Some Page... </html>
```

- **Código de Status**
- **Tipo de Conteúdo**
- **Conteúdo**

HTTP - Hypertext Transfer Protocol

Exemplo de URL

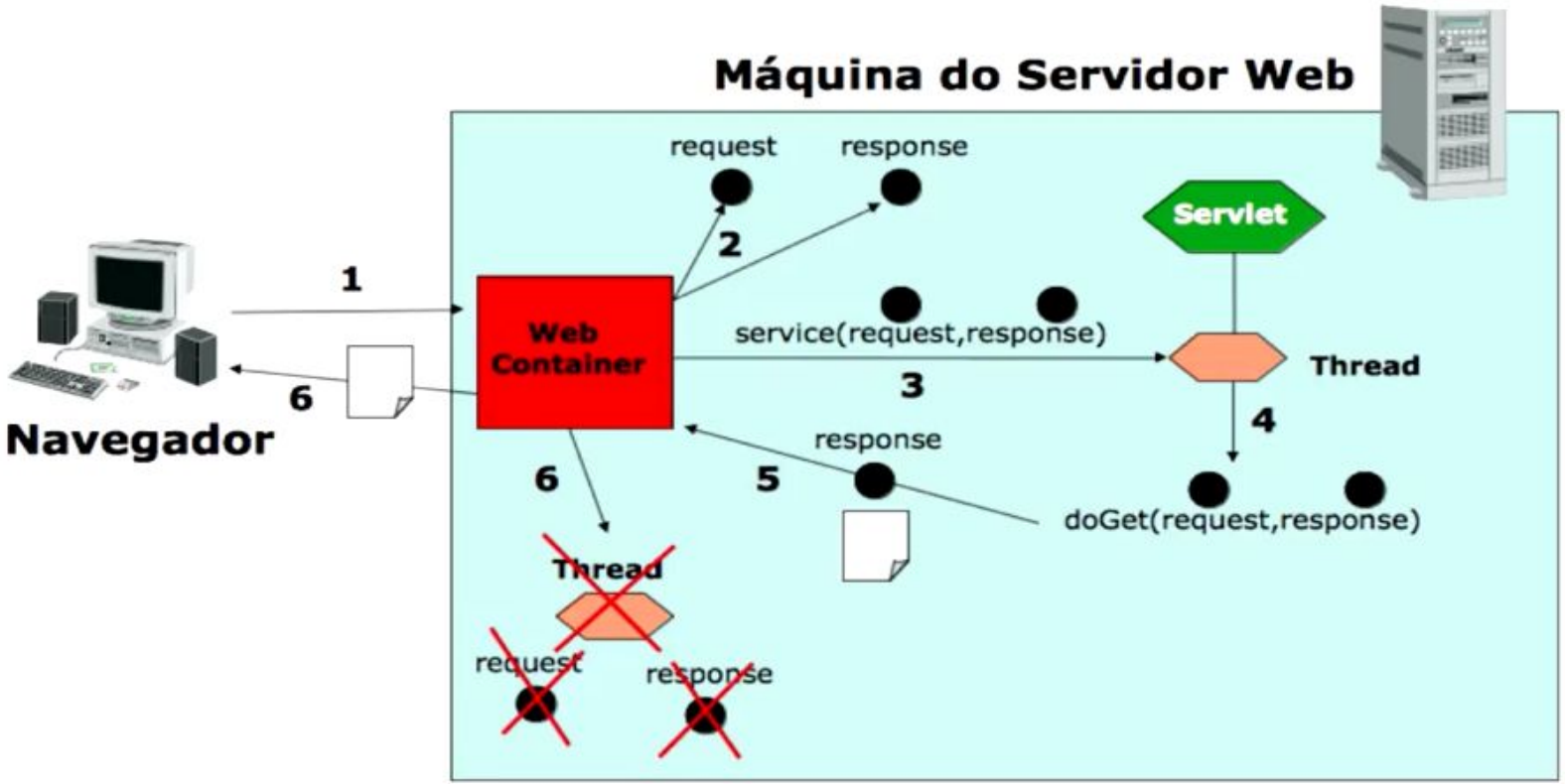


HTTP - Hypertext Transfer Protocol

Métodos HTTP

- GET
  - Normalmente utilizado para acesso a um link
  - Parâmetros passados junto com a URL
  - Pode ser repetido pois é utilizado somente como acesso
- POST
  - Normalmente usado para submissão de formulários
  - Parâmetros são passados no corpo da submissão
  - A repetição pode causar *efeito colateral no servidor (ex: submissão de formulário de cadastro várias vezes)
- Qual método usar? - O resultado da requisição pode ser adicionado aos favoritos?

Arquitetura



Arquitetura

1. Navegador submete uma requisição
2. Servlet container recebe a requisição e cria objetos para representar o request e o response.
3. É criada uma thread que executará o método service de uma Servlet.
4. É executado o método doGet ou doPost, ou qualquer outro que recebe os objetos request e response como parâmetro.
5. O método envia a resposta para o container.
6. O container responde ao navegador.

Obs.: O mesmo servlet pode ser executado por Threads diferentes, ou seja, clientes diferentes podem fazer a mesma requisição que serão tratadas separadamente, dando início ao conceito de concorrência.

Servlet

- Responsável por criar páginas dinâmicas com Java.
- É uma **classe** capaz de gerar conteúdo de resposta para as requisições de um cliente. Essas requisições são feitas a partir de chamadas **HTTP.**

- O comportamento da servlet é definido na classe abstrata **HttpServlet** do pacote **javax.servlet**.
- Uma subclasse do HttpServlet, deve sobrescrever, pelo menos um método como o doGet, doPost, etc.

JSP - Java Server Pages

- Tecnologia baseada em servlets
- Páginas Java contendo instruções HTML.
- No primeiro carregamento da página, o código Java é compilado gerando um servlet que é executado.
- Chamadas subsequentes à página são enviadas diretamente ao servlet.
- Expression Language ${ expr } - para remover um pouco do código Java

Tag Library - JSTL

- Coleção de bibliotecas que permite escrever páginas JSPs sem Java.
- Melhor legibilidade do código
- Simplesmente uma página JSP contendo um conjunto de tags

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

```jsp
<body>

    <jsp:useBean id="dao" class="br.com.contatos.jdbc.dao.ContatoDAO"/>

    <table>
      <c:forEach var="contato" items="${dao.listaContatos}">
        <tr>
          <td>${contato.nome}</td>
          <td>${contato.email}</td>
          <td>${contato.endereco}</td>
          <td>${contato.dataNascimento.time}</td>
        </tr>
      </c:forEach>
    </table>

</body>
```

JDBC - Java Database Connectivity

- **Interfaces do pacote java.sql**
  - **Connection** - define métodos para manipular o banco de dados.
  - **DriverManager** - responsável por se comunicar com todos os drivers disponíveis.
  - **PreparedStatement** - implementa um objeto que representa uma instrução SQL pré-compilada.
  - **ResultSet** - tabela de dados representando o resultado de uma consulta no banco

JDBC - Java Database Connectivity

- **Driver** - classes que se comunicam com o banco de dados.
  - postgresql, mysql, …
- **ConnectionFactory** - fábrica de conexões.
- **JavaBean** - classe com métodos getters e setters, utilizado como modelo para o banco de dados.
- **DAO** - objeto responsável por acessar os dados

MVC

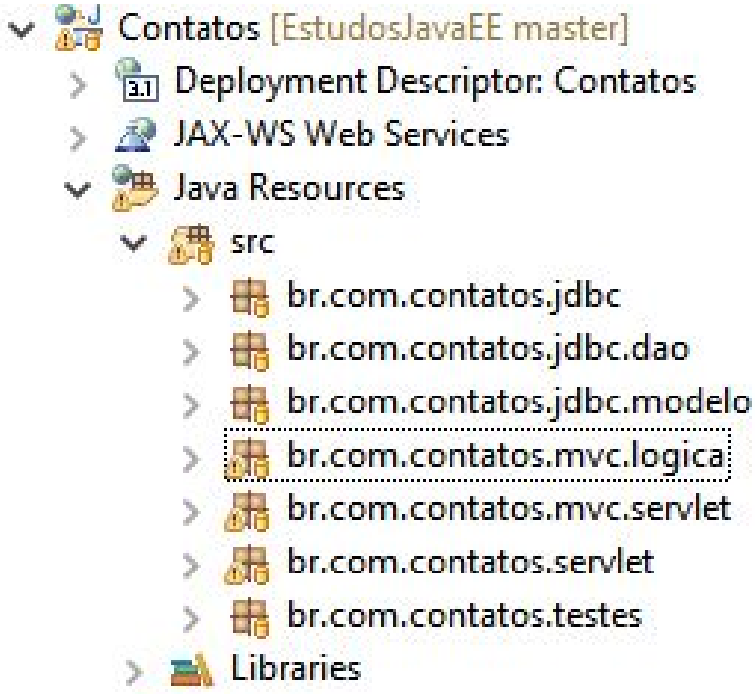
- Padrão arquitetural
- Dispatch das requisições, para que o JSP seja renderizado após uma servlet executar suas regras de negócios

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String parametro = request.getParameter("logica");
    String nomeDaClasse = "br.com.contatos.mvc.logica." + parametro;

    try {

        //Intancia uma classe, que ficará na memória para ser reutilizada
        Class classe = Class.forName(nomeDaClasse);

        Logica logica = (Logica) classe.newInstance();
        String pagina = logica.executa(request, response);

        request.getRequestDispatcher(pagina).forward(request, response);

    } catch (Exception e) {
        throw new ServletException("A lógica de negócios causou uma exceção", e);
    }

}
```

Estrutura da aplicação

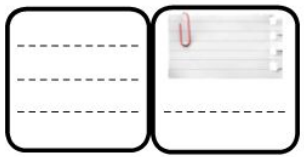
- **src** = código fonte
- **build** = classes compiladas
- **WebContent** = raiz do conteúdo a ser exibido no navegador
- **WEB-INF** = configurações e recursos para o projeto funcionar no servidor
- **WEB-INF/lib** = bibliotecas necessárias para a aplicação web

Estrutura da aplicação



Aplicação

Home Adicionar contato



Agenda de contatos

Nome	Email	Endereço	Data de nascimento	
Felipe	felipesobreira.c@hotmail.com	Rua X	02/08/2017	Remover
Maria da silva	maria@email.com	Av. XI	28/08/2017	Remover
João Alfredo	alfjoao55@email.com	Av. IV	18/08/2017	Remover
Aline line	line@email.com	Av. XIII	20/08/2017	Remover
Karlison carlos	carlos@email.com	Av. II	16/08/2017	Remover

Referências

- CAELUM. Java para desenvolvimento web. Disponível em: https://www.caelum.com.br/apostila-java-web/
- ITA. Desenvolvimento ágil com java avançado. Disponível em: https://www.coursera.org/learn/desenvolvimento-agil-com-java-avancado