Programação WEB

Java Server Pages e Servlets

Douglas Nassif Roma Junior

douglas.junior@grupointegrado.br

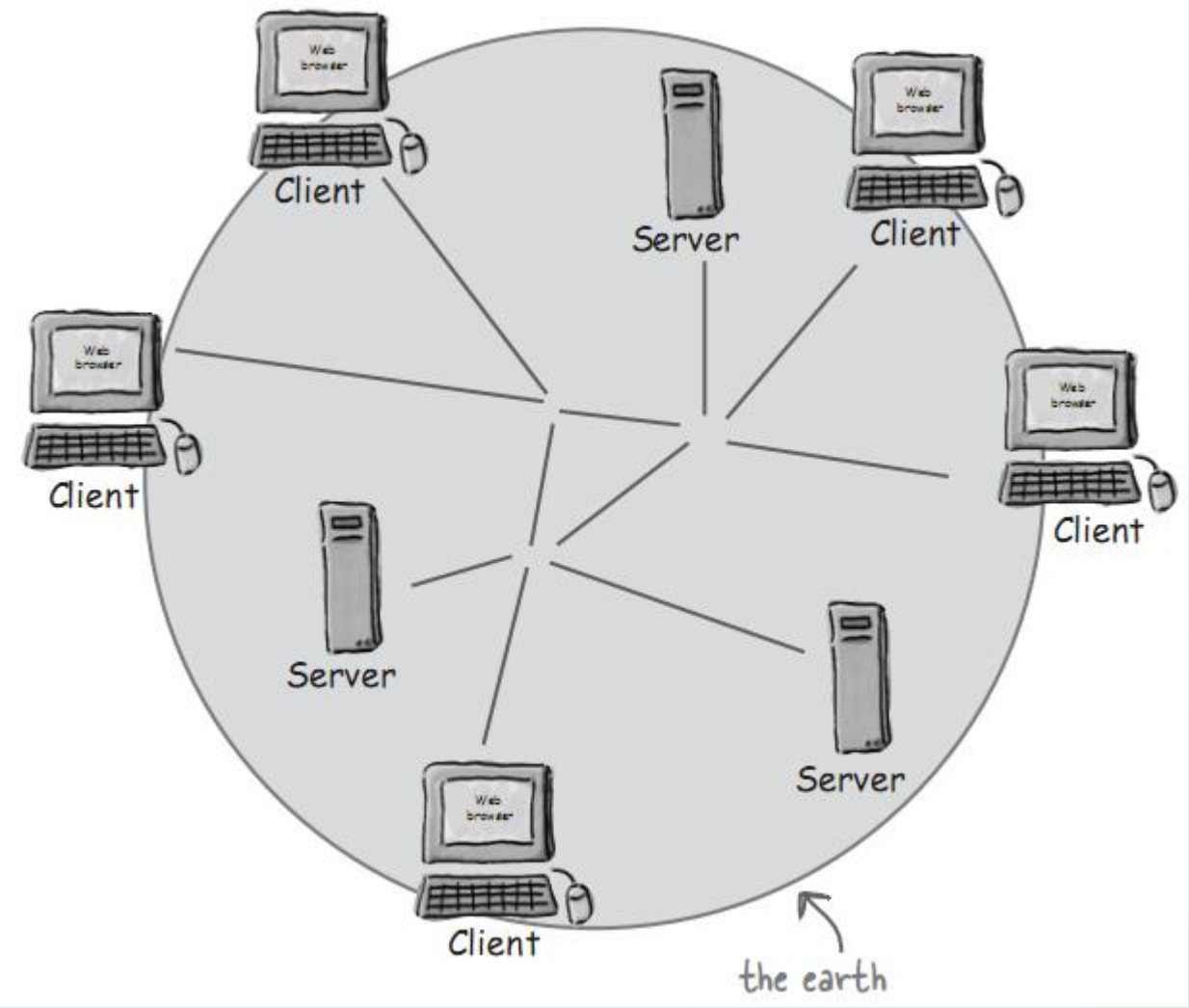
Conteúdo

- A Internet
- Cliente x Servidor
- HTTP
- GET x POST
- Mostrar a evolução da arquitetura para o desenvolvimento de aplicações WEB explorando o uso da Linguagem Java.

Internet

- Criada a ARPANET em 1970 com a finalidade de conectar departamentos de Pesquisa nos EUA
- Protocolo inicial *Network Control Protocol*(NCP)
- Em 1975 criação do TCP/IP
- Em 1990 a Internet passa a ter tendência comercial e não apenas pesquisa.
  - Criação de páginas estáticas
  - Criação de páginas dinâmicas

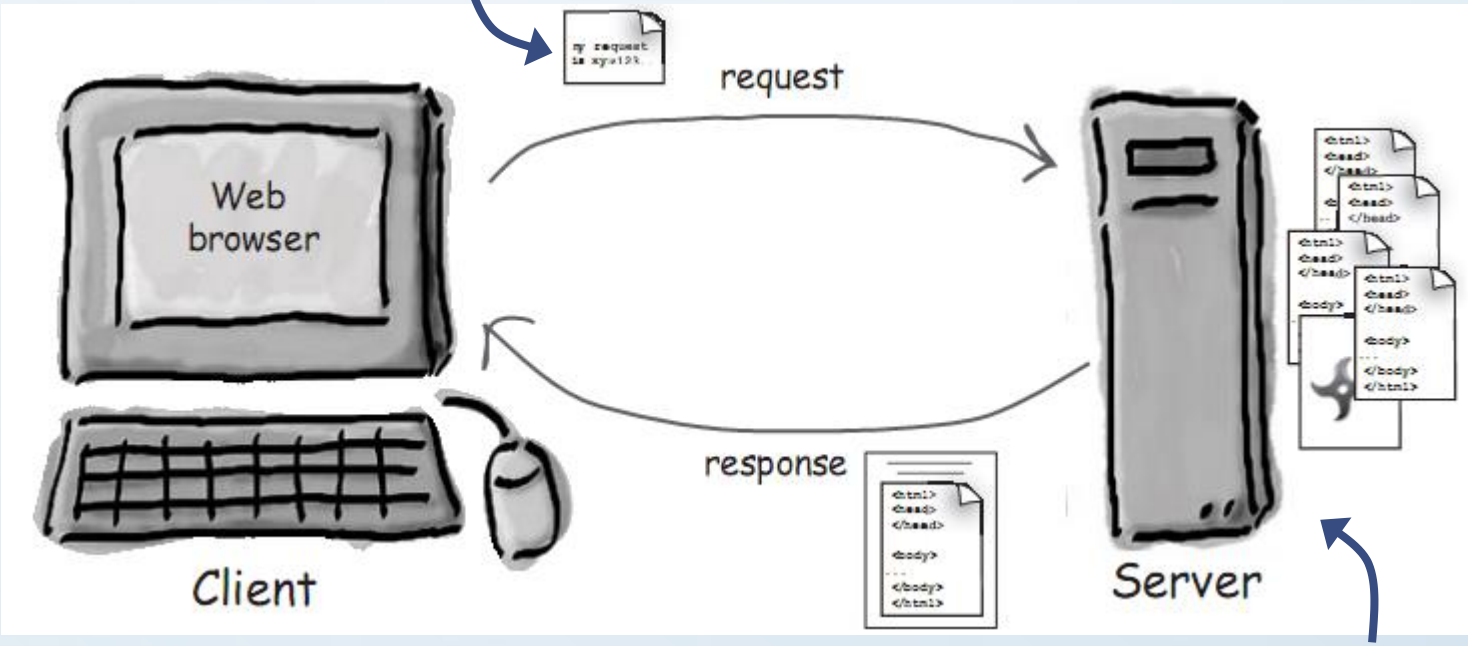
A Internet



A web consiste em bilhões de clientes (usando browsers como Mozilla ou IE) e servidores (rodando aplicações como o Apache), conectados através de redes com fio e wireless. Nosso objetivo é construir uma aplicação que os clientes ao redor do mundo possam acessar.

O que o servidor faz?

A solicitação do cliente contém o nome e endereço (a URL), daquilo que o cliente esta procurando.



Geralmente, o servidor tem muito conteúdo que pode ser mandado para o usuário. Este conteúdo pode ser páginas, JPEG e outros recursos.

A resposta do servidor contém o documento que o cliente solicitou (ou um código de erro se o pedido não puder ser processado).

Solicitação (*request*)

1

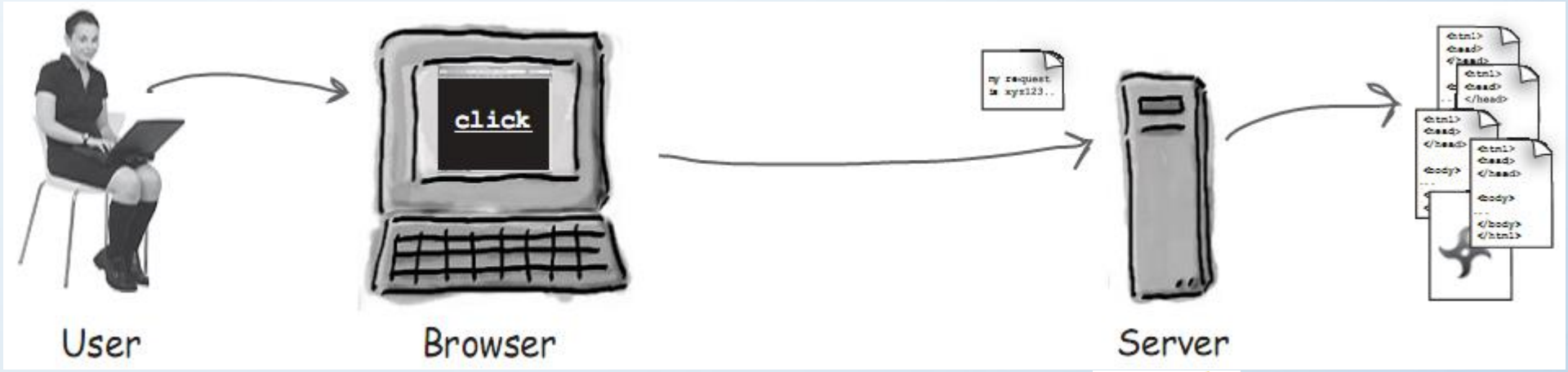
O usuário clica em um link no browser

2

O browser formata a solicitação e a envia para o servidor.

3

O servidor encontra a página solicitada.



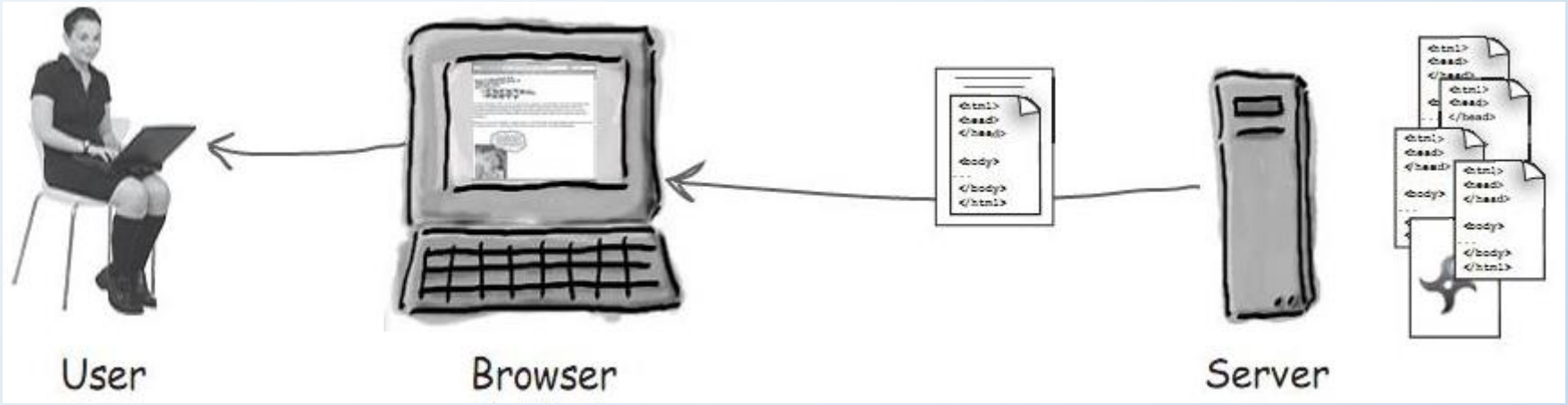
Resposta (*response*)

5

O browser consegue o HTML e o traduz em formato visual para o usuário.

4

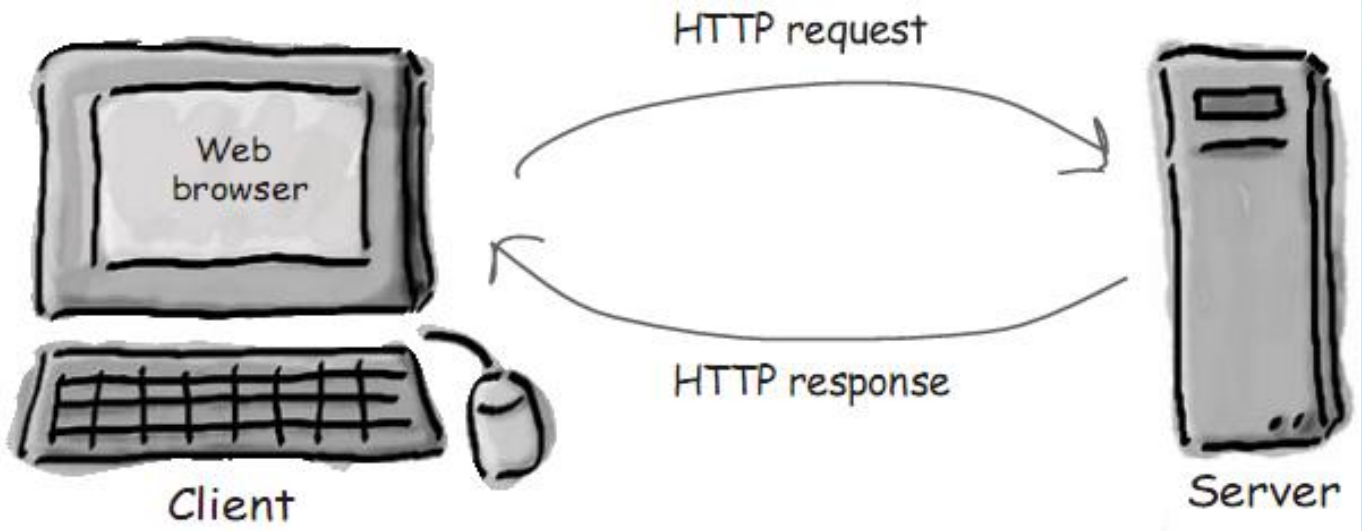
O servidor formata a resposta e a envia para o cliente (browser)



Protocolo HTTP

Principais elementos do fluxo de solicitação:

- O método HTTP (a ação de ser executada).
- A página que será acessada (uma URL)
- O parâmetros do formulário (como argumentos para um método).



Principais elementos do fluxo de resposta:

- Um código de status (no caso de uma solicitação bem sucedida)
- Tipo de conteúdo (texto, imagem, HTML, etc.)
- O conteúdo (o HTML real, a imagem, etc.)

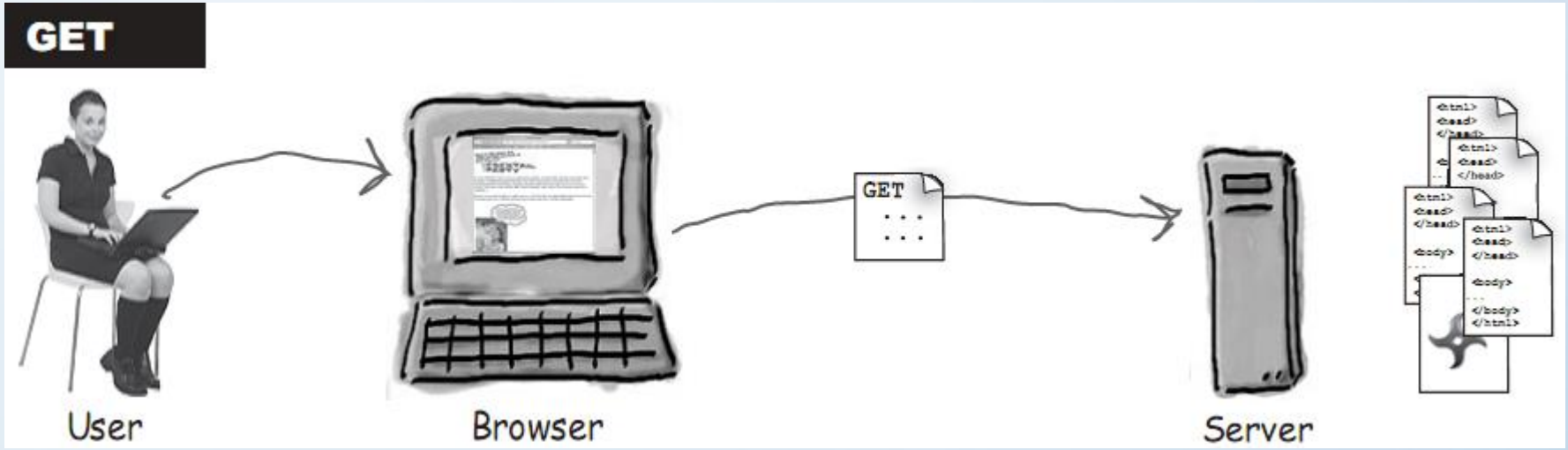
HTTP GET

1

O usuário clica em um link para uma nova página.

2

O browser envia um HTTP GET ao servidor, pedindo ao servidor que consiga a página



Solicitação GET

A linha de solicitação

O caminho para o recurso no servidor.

Os headers da solicitação

```
GET /paginas/pagina.jsp HTTP/1.1
Host: www.grupointegrado.br
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv; 1.4)
Gecko/20030624 Netscape/7.1
Accept: text/xml, application/xml, application/xhtml+xml, text/html;
q=0.9, text/plain; q=0.8, video/x-mng, image/png, image/jpeg, image/gif;
q=0.2, */*; q=0.1
Accept-Language: en-us,en; q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *; q=0.7
Keep-Alive: 300
Connection : keep-alive
```

Espere um momento...
Eu poderia jurar que vi solicitações GET que enviavam alguns dados por parâmetros ao servidor.

Solicitação GET

Em uma solicitação GET, os parâmetros (se existir algum) serão Anexados à primeira parte da solicitação URL, iniciando-se por Uma "?". Os parâmetros são separados usando-se o "&".

A linha de solicitação

Os headers da solicitação

```
GET /paginas/pagina.jsp?id=1&aluno=Douglas HTTP/1.1
Host: www.grupointegrado.br
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv; 1.4)
Gecko/20030624 Netscape/7.1
Accept: text/xml, application/xml, application/xhtml+xml, text/html;
q=0.9, text/plain; q=0.8, video/x-mng, image/png, image/jpeg, image/gif;
q=0.2, */*; q=0.1
Accept-Language: en-us,en; q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *; q=0.7
Keep-Alive: 300
Connection : keep-alive
```

Solicitação POST

A linha de solicitação

O caminho para o recurso no servidor.

Os headers da solicitação

O corpo da mensagem

```
POST /paginas/recebeDados HTTP/1.1
Host: www.grupointegrado.br
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv; 1.4)
Gecko/20030624 Netscape/7.1
Accept: text/xml, application/xml, application/xhtml+xml, text/html;
q=0.9, text/plain; q=0.8, video/x-mng, image/png, image/jpeg, image/gif;
q=0.2, */*; q=0.1
Accept-Language: en-us,en; q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *; q=0.7
Keep-Alive: 300
Connection : keep-alive

aluno=Claudio&idade=17
```

Praticando

- Um usuário digitando um login e uma senha.
- Um usuário solicitando uma nova página via hyperlink
- Um usuário em uma sala de bate-papo enviando uma resposta.
- Um usuário clica no botão "next" para ver a próxima página.
- Um usuário clica no botão de "logout" num site seguro de um banco.
- Um usuário clica em "Voltar" no browser.
- Um usuário envia em formulário com nome e endereço para o servidor.
- Um usuário faz uma escolha em um botão de seleção.

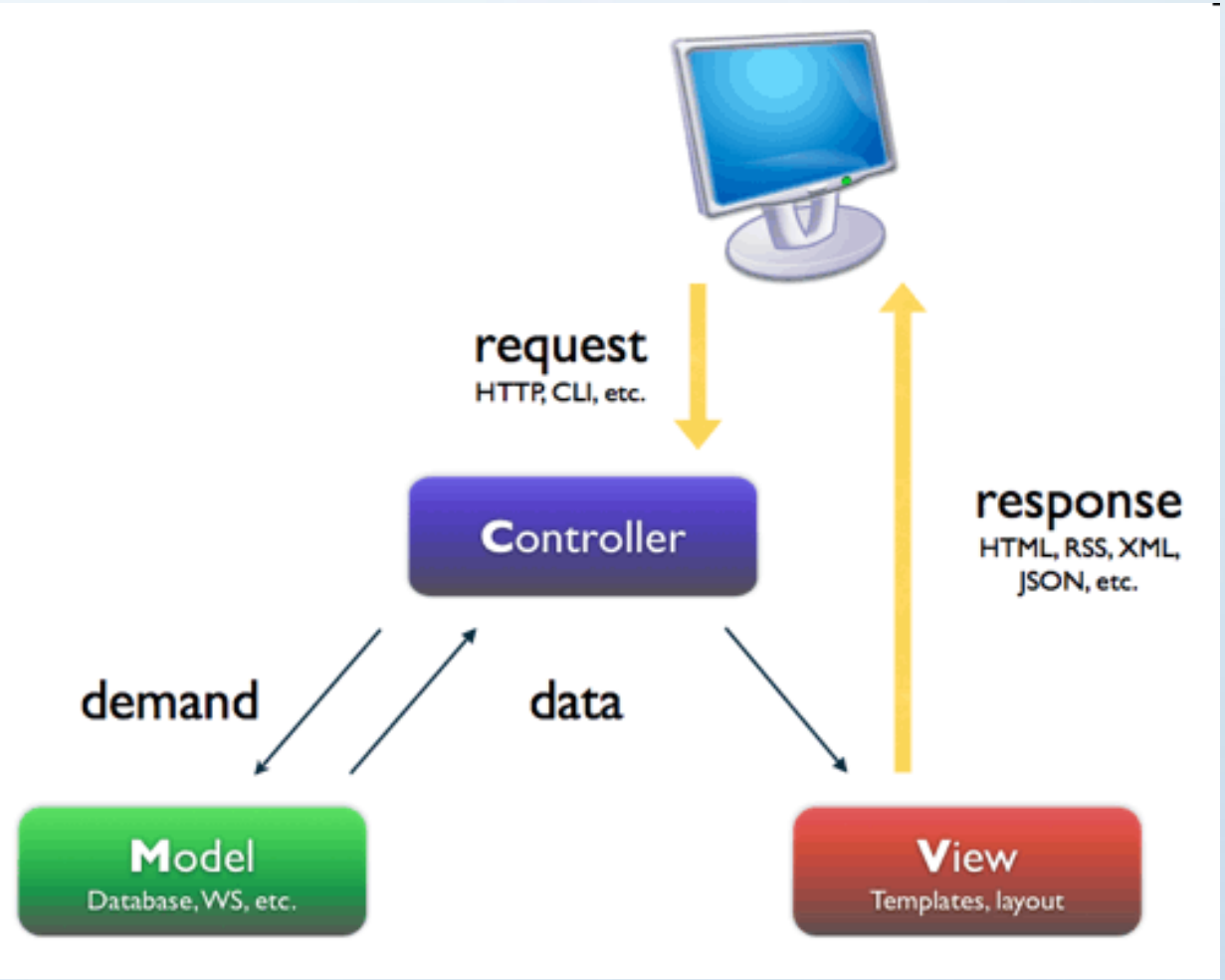
Características de uma Aplicação Web

- Sujeito a interrupções estranhas ao fluxo normal da aplicação.
- Vulneráveis à irregularidade de fluxo.
- Desenvolvimento em equipe.

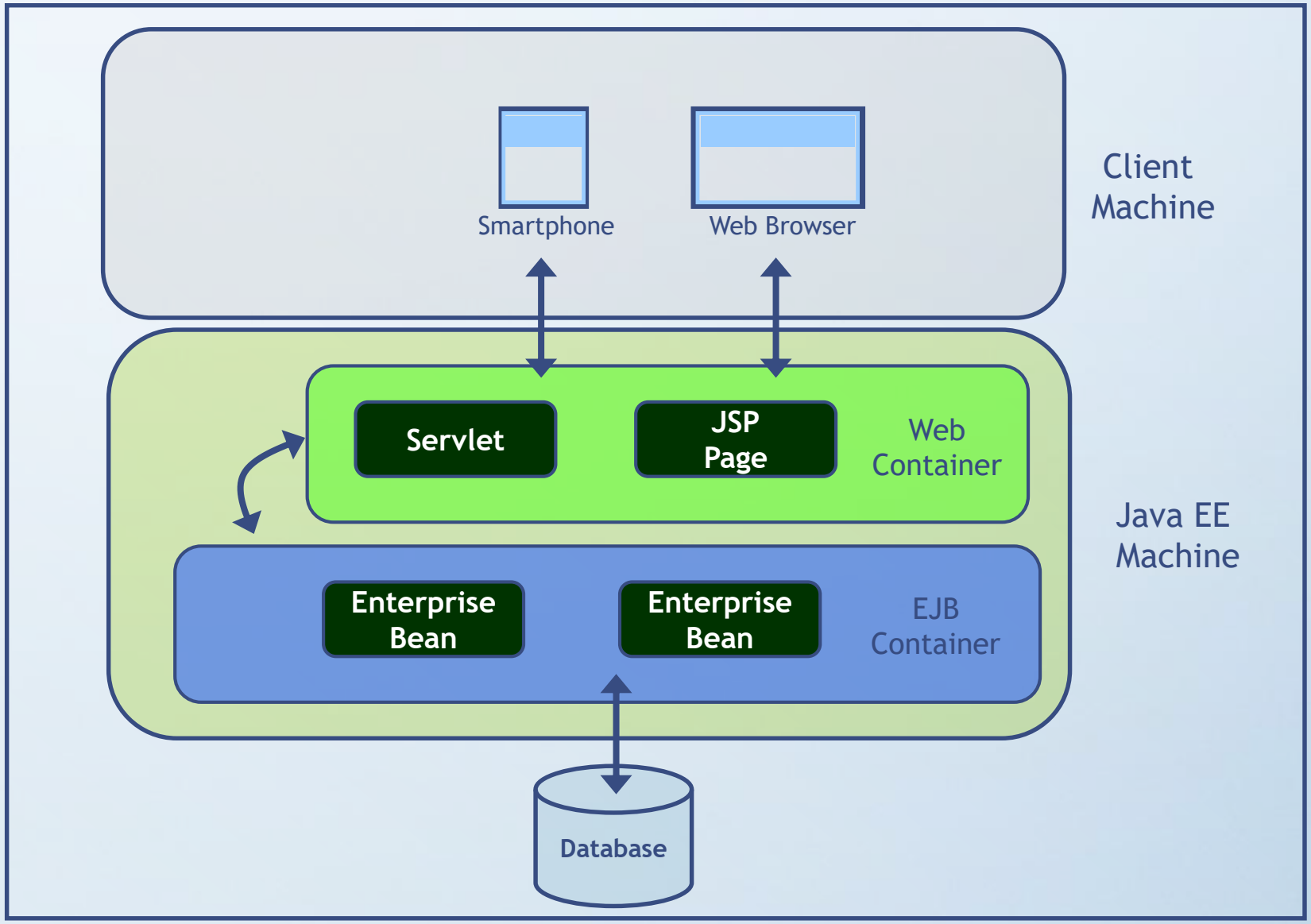
Características de uma Aplicação Web

- Protocolo HTTP – Solicitação / Resposta;
- Sem informação de estado.

Composição de uma Aplicação



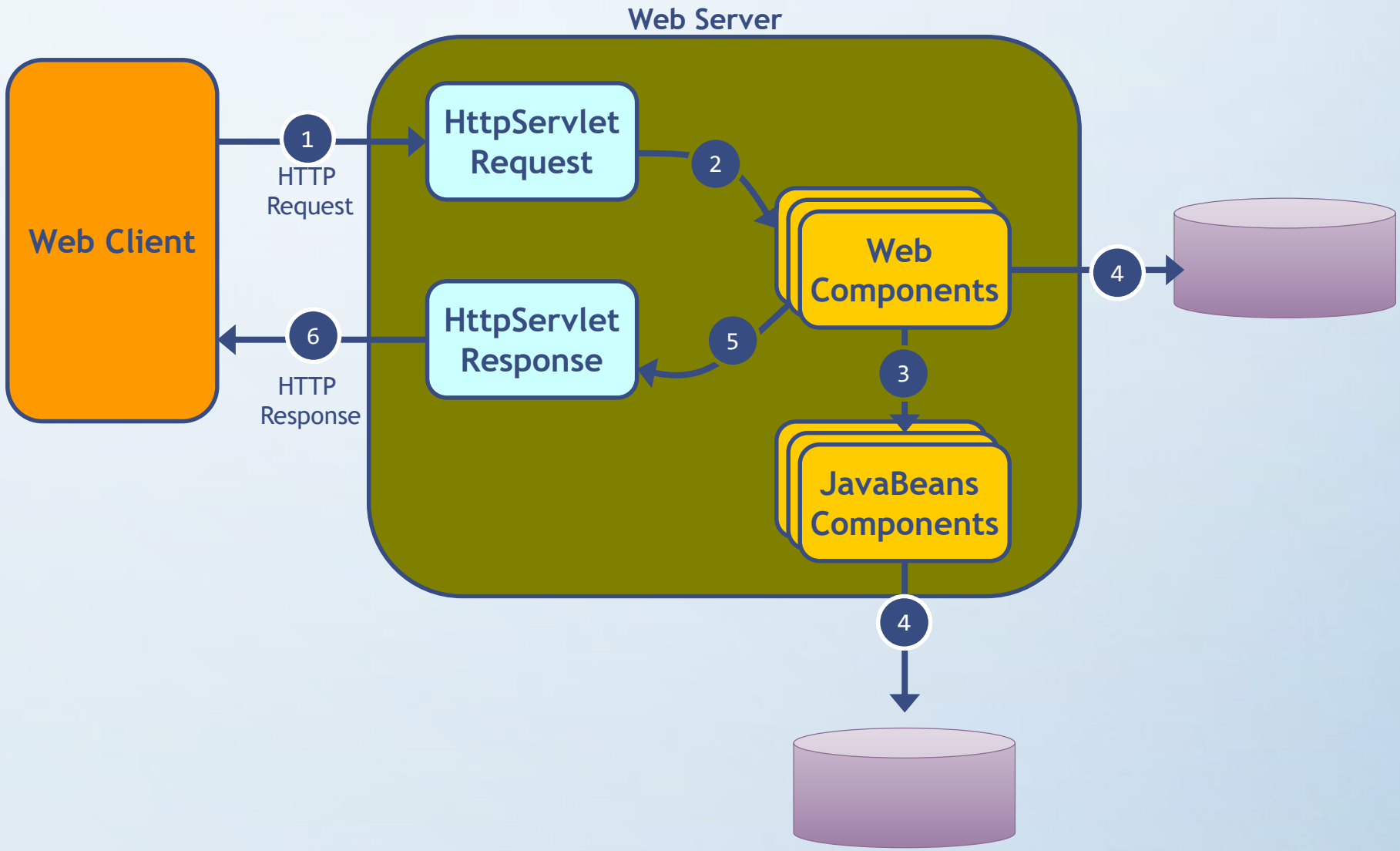
Estrutura da Aplicação Web



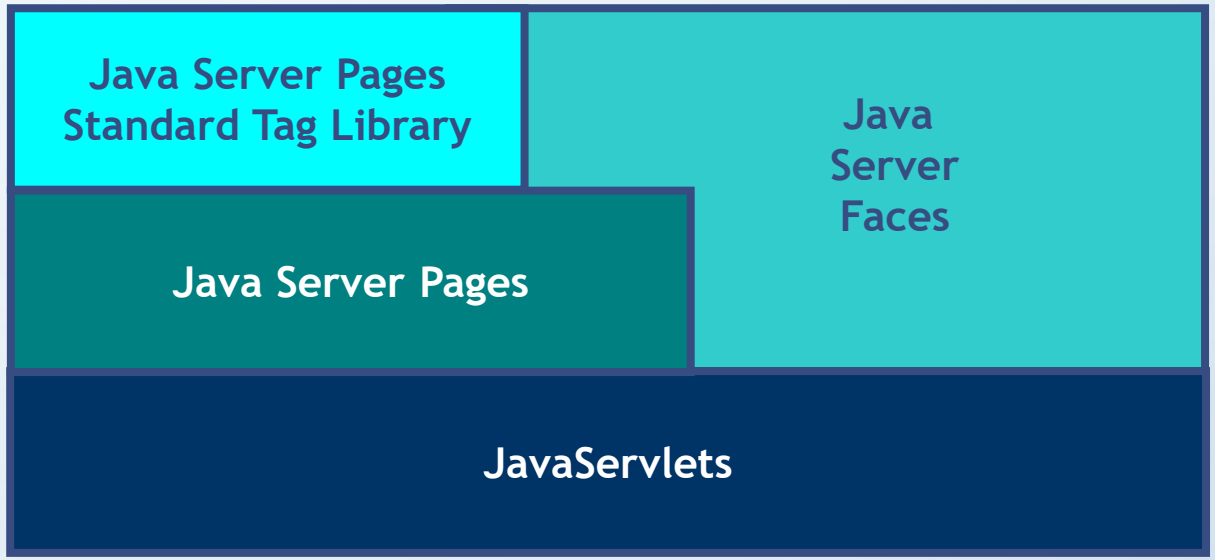
Estrutura da Aplicação Web

- Componentes Web (*Web Components*): JSP, Servlets, Tags e JavaBeans;
- Componentes Web rodam em Containers Web: Todo servidor JavaEE fornece o Container web;
- Containers fornecem serviços para Componentes Web (autenticação, encaminhamento de requisições, segurança, administração do ciclo de vida);
- Uma Aplicação Web é uma estrutura de diretórios que contêm JSP, Servlets, web.xml, taglibs, classes, bibliotecas, etc.
- Aplicações Web podem ser distribuídas utilizando-se arquivos do tipo **`war`**.

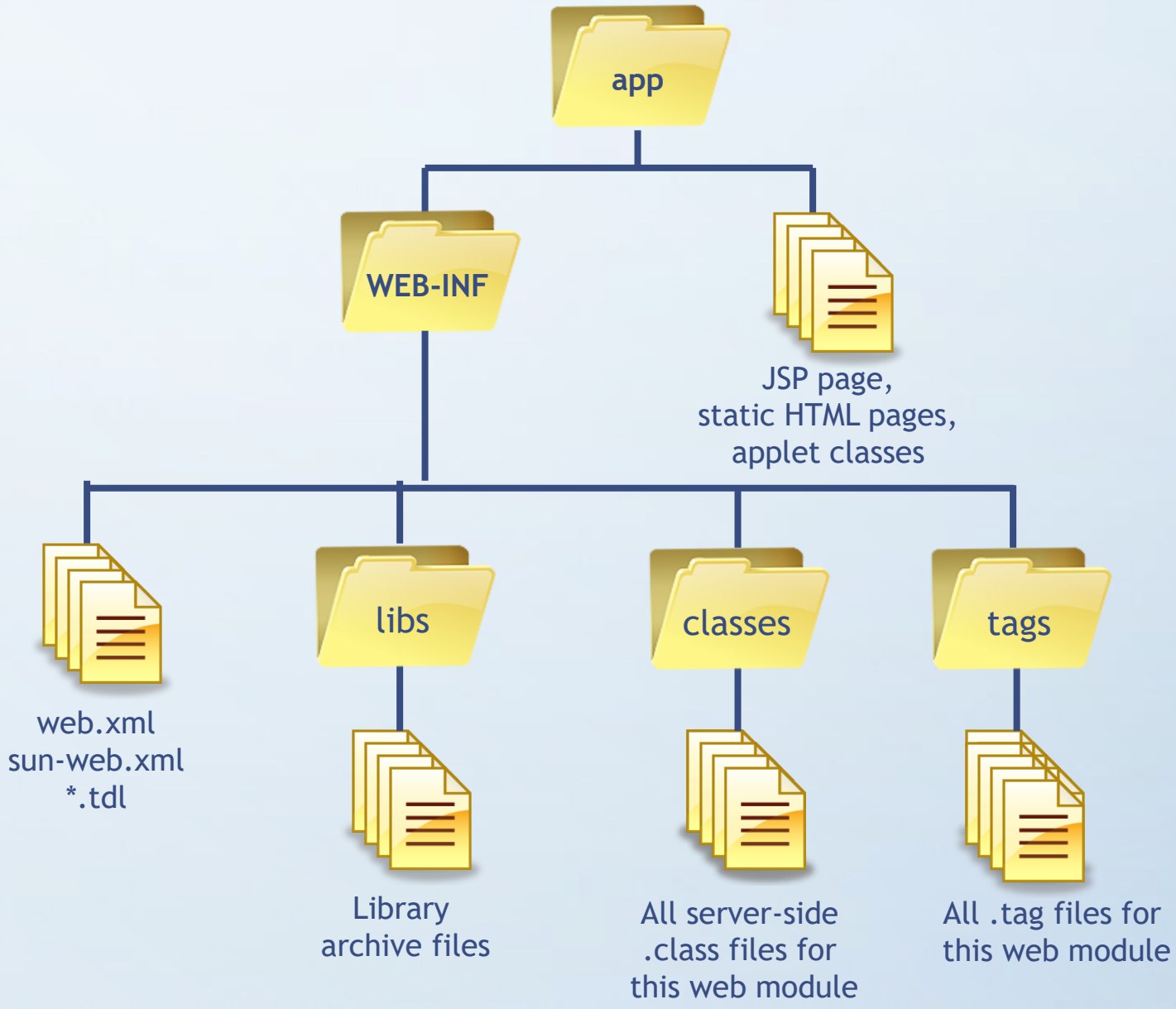
Estrutura da Aplicação Web



Estrutura da Aplicação Web



Estrutura da Aplicação Web



Arquitetura de Aplicações Web

- Desenvolvimento Java para Web:
  - Design centrado em Servlets;
  - Design centrado em JSP.

Design centrado em JSP

- Características:
  - Páginas inter-relacionadas;
  - JSPs solicitadas diretamente pelos usuários;
  - Realiza controle/lógica/apresentação.

Design centrado em JSP

listaAgencia.jsp

```
<% @page session="true"  import="java.util.*" %>
<html>
   <head><title>Bancos do Mensalao</title></head>
<body>
   <h1>Lista de Agencias</h1>
   <%
      AgenciasAdm agencias = new AgenciasAdm();
      Agencia agencia;
      ArrayList listaAgencias = agencias.listaTodos();
      for (int i=0; i < listaAgencias.size(); i++ ) {
         agencia = (Agencia) listaAgencias.get(i);
    %>
     <a href="<%=Agencia.getId()%>"><%=agencia.getNome()%></a><br>
    <% } %>
</body>
</html>
```

Design centrado em JSP

- Limitações
  - Baixa manutenibilidade;
  - Baixa interoperabilidade;
  - Baixa reusabilidade;
  - Sem controle de fluxo;
  - Cada JSP deve validar as solicitações vindas dos usuários;
  - Mistura HTML e código Java.

Design centrado em Servlets

- Características
  - Controla as solicitações dos clientes;
  - Executa a lógica da aplicação;
  - Atribui a apresentação à páginas JSPs.

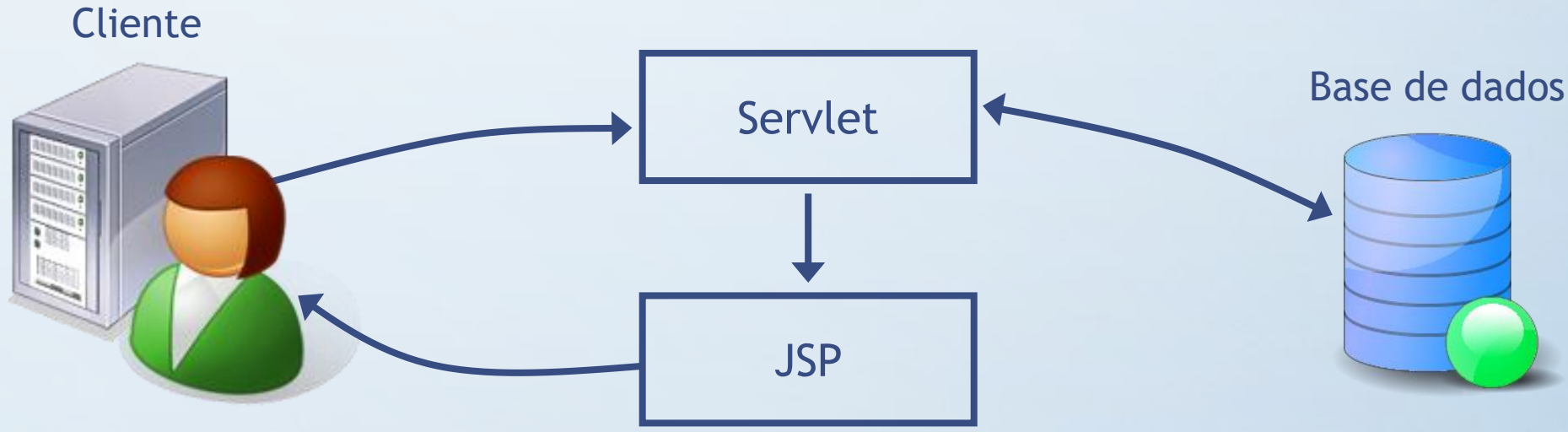
Design centrado em Servlets

ContasUsuario.java

```
import ....;

public class ContasUsuario extends HttpServlet {

  public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String cpf = request.getParameter("cpj");
        Arraylist lista = listar(cpf);
        request.setAttribute("ListadeContas", lista);
        response.sendRedirect("/banco/contasdoUsuario.jsp");
   }

  public Arraylist listar(String cpf){....}
  public void Salvar(String cpf, String nome) {...}
  public Connection getConexao(){...}
}
```

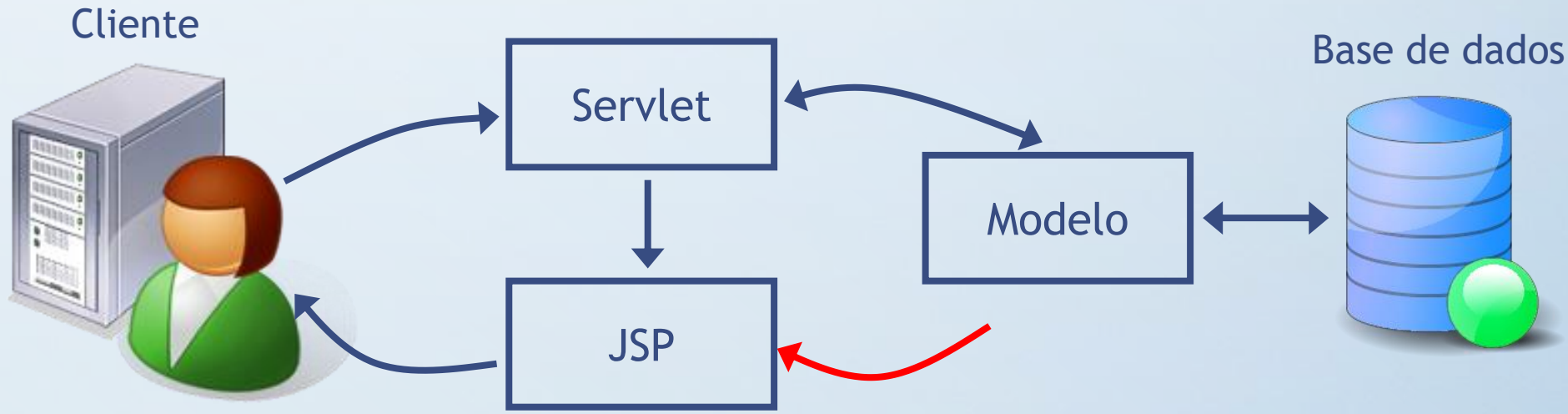
Design centrado em Servlets

- Limitações
  - Controle de solicitações e lógica da aplicação juntas
  - Manutenibilidade, Interoperabilidade e Reusabilidade, ainda, reduzida



MVC – Modelo, Visão e Controle

- Arquitetura que disciplina o desenvolvimento Web:
  - Modelo – Classes Java
  - Visão - JSPs
  - Controle - Servlets



MVC – Modelo, Visão e Controle

ControllerServlet.java

```
public class ControllerServlet extends HttpServlet {
  public void doGet(... request,... response) throws .... {
     String cpf = request.getParameter("cpf");
     String nome = request.getParameter("nome");
     String endereco = request.getParameter("endereco");
     String operacao = request.getParameter("opc");
     if(operacao.equals("salvar")){
        Correntista corr = new Correntista(nome,email,endereco);
        CorrentistaAdm corrAdm = new CorrentistaAdm();
        corrAdm.salvar(corr);
        request.setAttribute("correntista", corr);
        response.sendRedirect("/banco/resposta.jsp");
     }else{
        if(operacao.equals("listar"))
        .....
     }
  }
```

Controle

MVC – Modelo, Visão e Controle

Correntista.java

```
public class Correntista {
   private String cpf;
   private String nome;
   private String endereco
   public Correntista() {  }
   gets/sets .....
}
```

Modelo

MVC – Modelo, Visão e Controle

CorrentistaAdm.java

```
imports ....
public class CorrentistaAdm {
   public void salva (Correntista corr){
   try{
      Connection con = Banco.getConexao();
      PreparedStatement pstmt = con.prepareStatement("INSERT INTO
      TB_CORRENTISTA(CPF,NOME, ENDERECO) VALUES (?,?,?)");
      pstmt.setString(1,corr.getCPF());
      pstmt.setString(2,corr.getNome());
      pstmt.setString(3,corr.getEndereco());
      pstmt.executeUpdate();
  }catch(SQLException e){
      System.out.println("Problemas ao abrir a conexão com o BD");
  }finally{
      Banco.closeConexao(con,pstmt);
  }
  public void deleta(Correntista corr){....}
  public void listaAll(){....} public void listaAll(){....}
}
```

Modelo

MVC – Modelo, Visão e Controle

Resultado.jsp

```
<html>
 <head>
  <title> Cadastro de  Correntista</title>
 </head>
 <body>
  <h1>Cadastro Realizado com Sucesso</h1>
  <jsp:useBean id="Correntista" scope="request"
class="modelo.Correntista"/>
  Correntista: <jsp:getProperty name="Correntista" property="nome"/>
 </body>
</html>
```

Visão

Vantagens do MVC

- Vantagens do MVC:
  - Controle de fluxo centralizado
  - Robustez
  - Fácil manutenção
  - Melhor reuso

Dúvidas

?