

Programação Web

Módulo 2 - Java Servlets

Daniel Lucrédio
daniel.lucradio@ufscar.br

"Alo mundo"

Primeiros passos

Como instalar um servidor

Como fazer uma primeira aplicação

Como implantar a aplicação

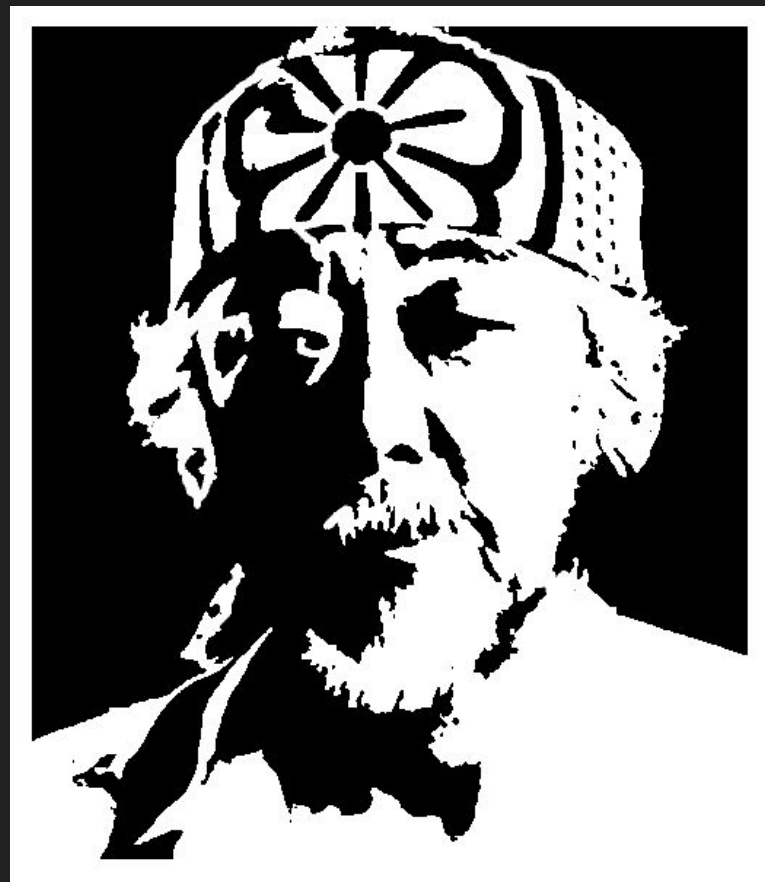
Instalação e execução Tomcat

Demonstração 3

Primeira aplicação

Utilizando linha de comando apenas

Depois veremos meios mais produtivos



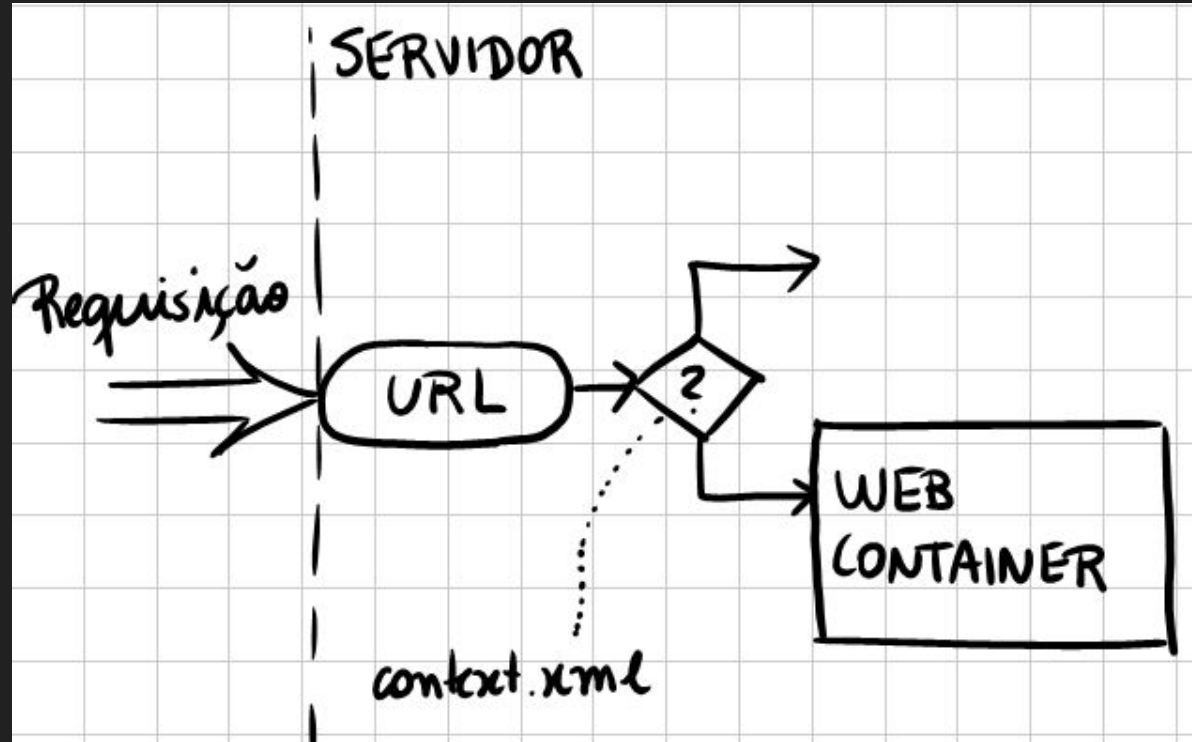
"Alo mundo"

Demonstração 4

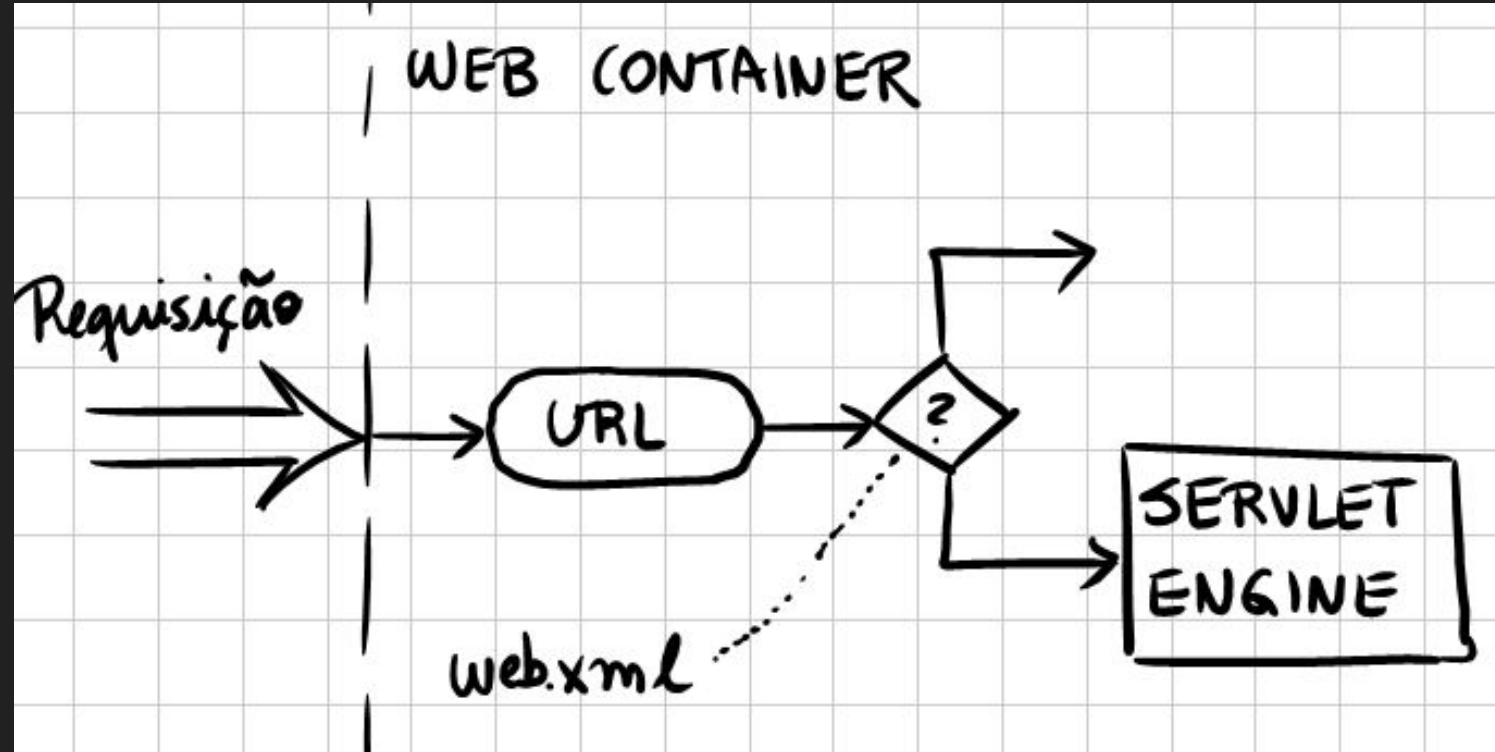
"Alo mundo" - em uma IDE

Demonstração 5

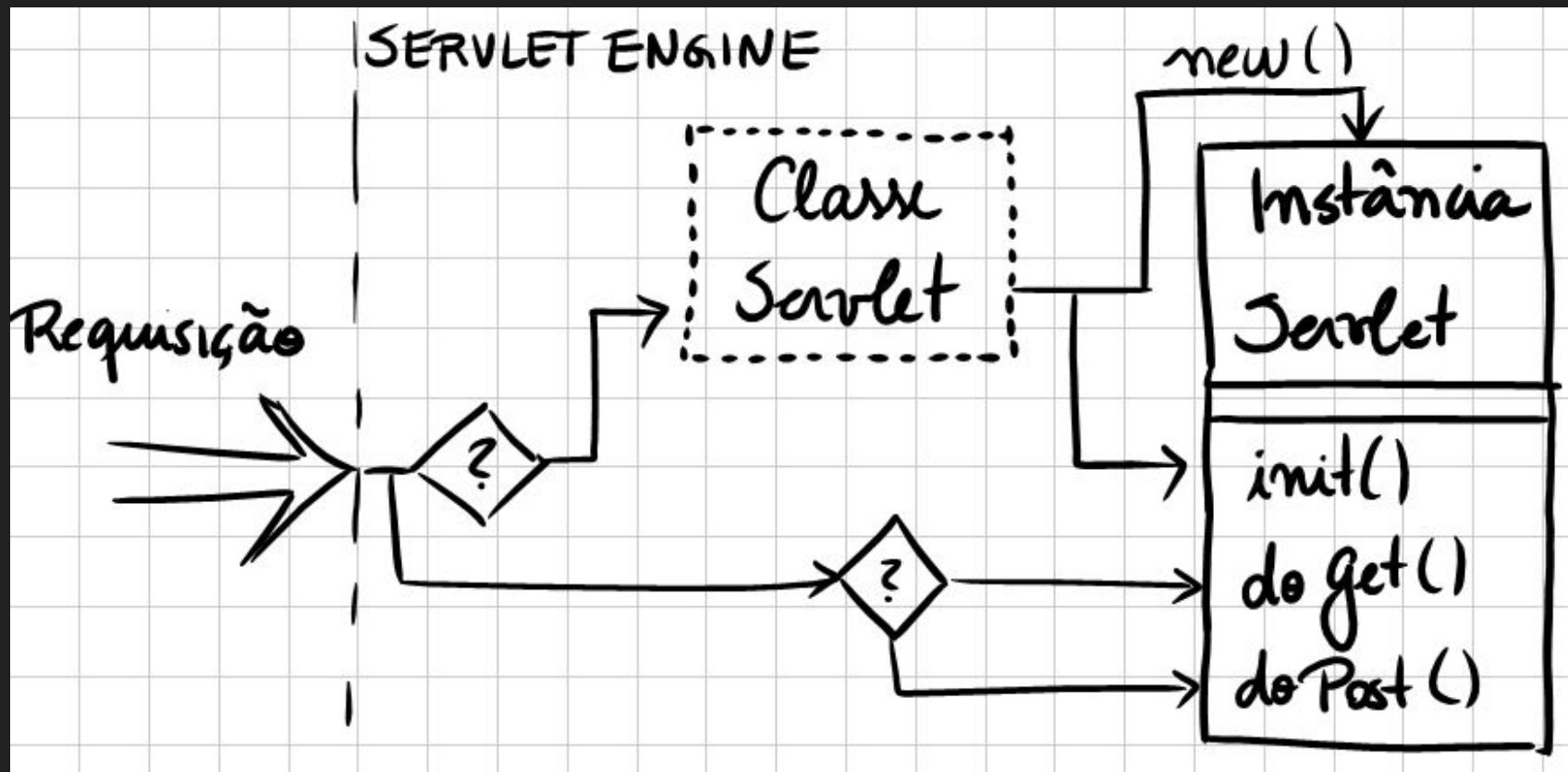
Ciclo de vida de um Servlet



Ciclo de vida de um Servlet



Ciclo de vida de um Servlet



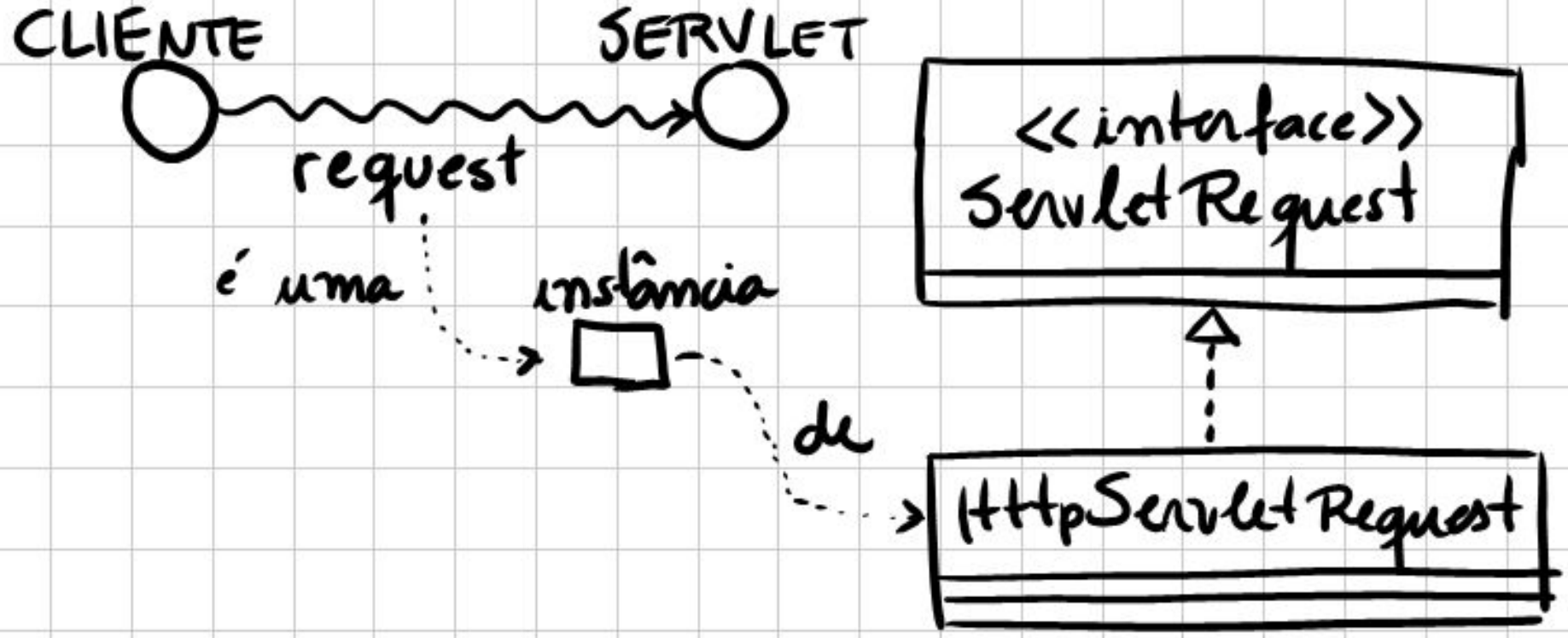
Principais métodos

- **init()**
 - Tarefas de inicialização
 - “construtor” do servlet
- **doGet()**
 - Trata requisições do tipo HTTP GET
- **doPost()**
 - Trata requisições do tipo HTTP POST
- **Outros serviços (pouco utilizados)**
 - doDelete(), doOptions(), doPut(), doTrace()

Tratamento de uma requisição

- É feito utilizando um objeto do tipo ServletRequest (HttpServletRequest)
- Procedimento correto:
 - 1: Obter informações da requisição
 - 2: Obter um fluxo de saída da resposta
 - 3: Preencher os cabeçalhos de resposta
 - 4: Escrever o corpo da resposta
- Essa ordem é importante
 - Preencher os cabeçalhos após o corpo ter sido enviado é inútil

Tratamento de uma requisição



Tratamento de uma requisição

Demonstração 6

Navegação entre Servlets

- **Cooperação**

- Servlet/Servlet
- Servlet/Outro recurso web

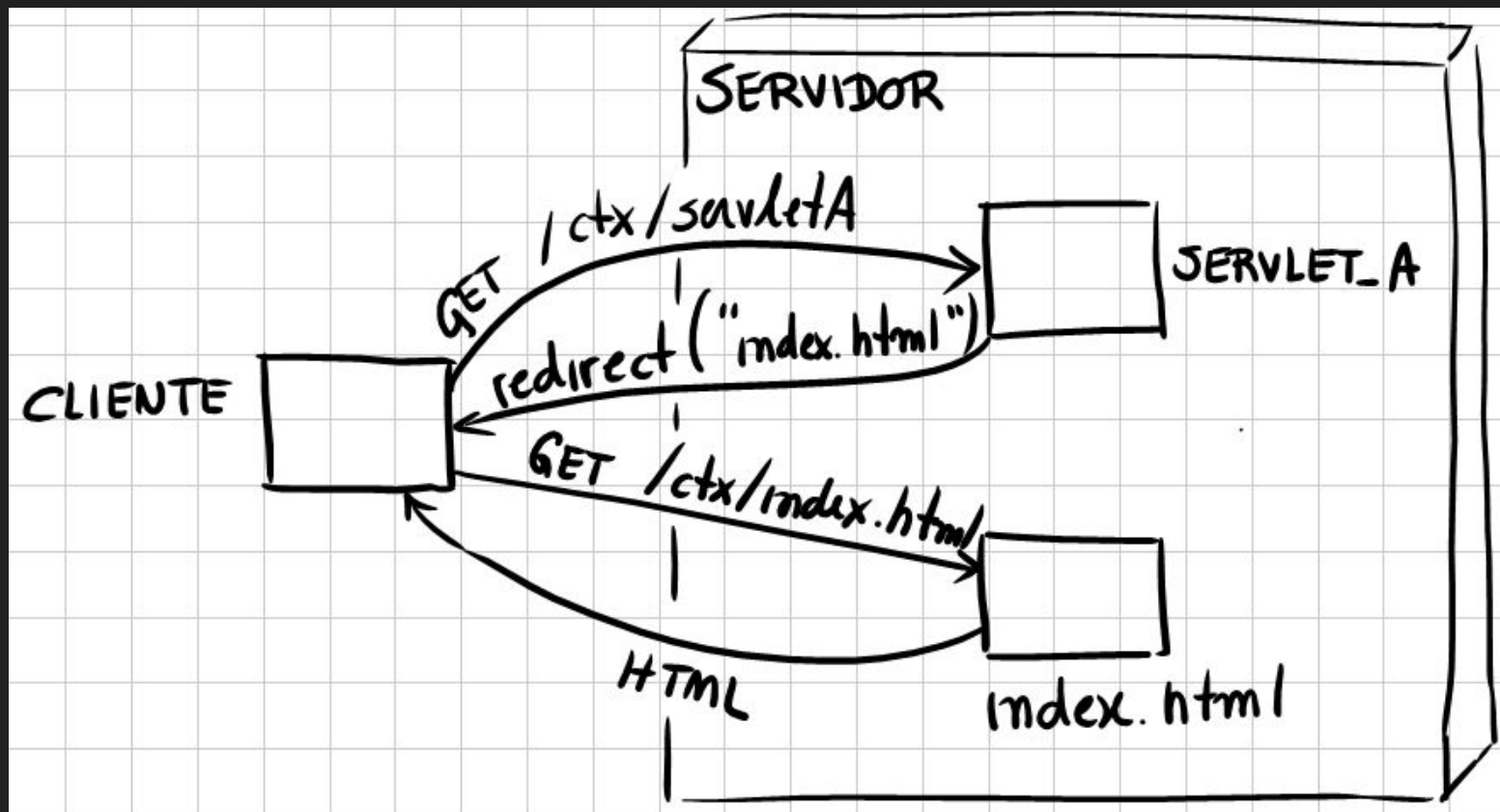
- **Exemplos:**

- Um servlet faz validação e outro faz processamento
- Um servlet faz o processamento e um HTML mostra o resultado

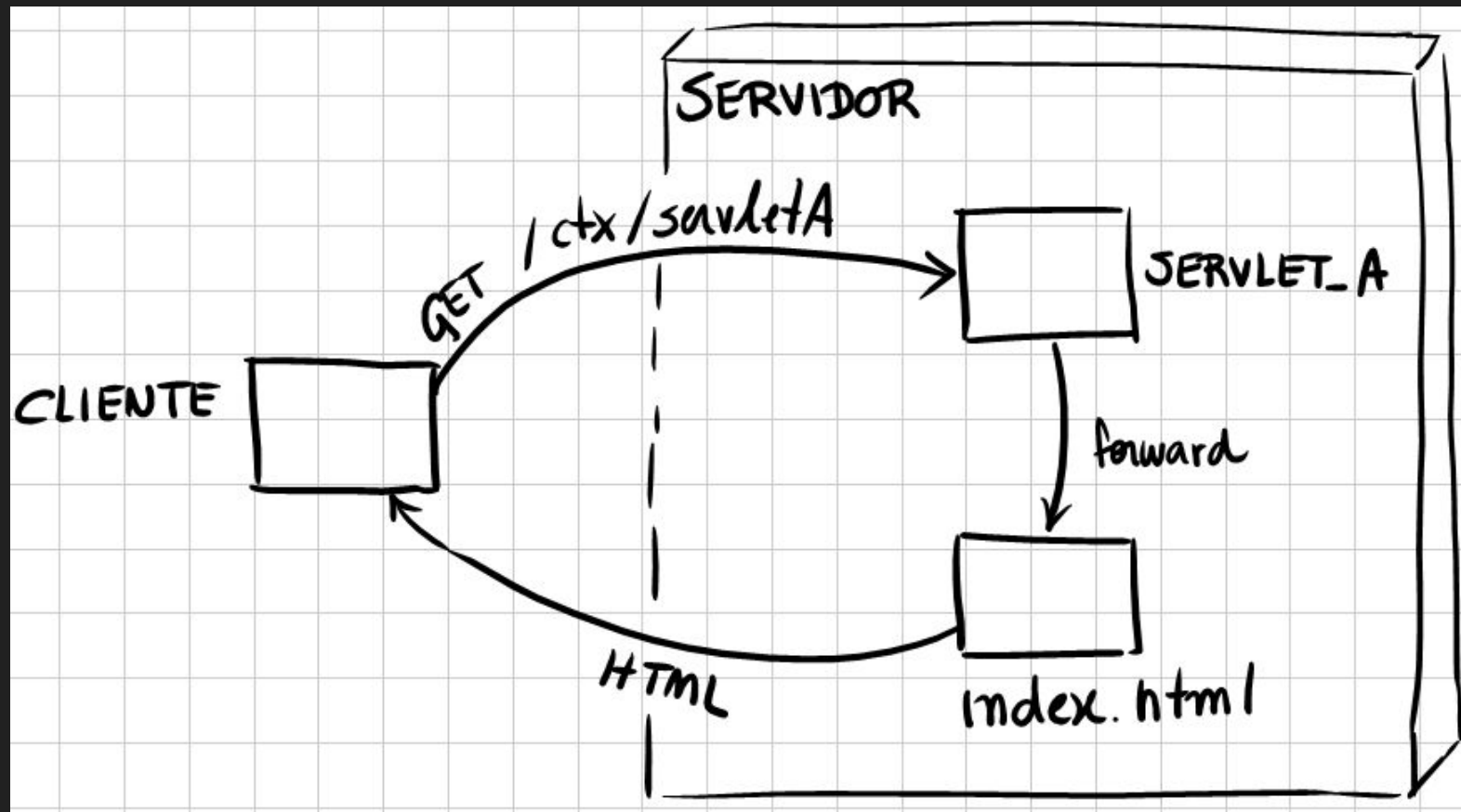
Navegação

- 3 modos:
- **Redirecionamento**
 - O servlet responde, pedindo para que o cliente faça outra requisição
- **Encaminhamento**
 - O servlet não responde, passando a bola para que outro recurso o faça
- **Inclusão**
 - Um servlet gera a resposta incluindo outros recursos como parte da resposta

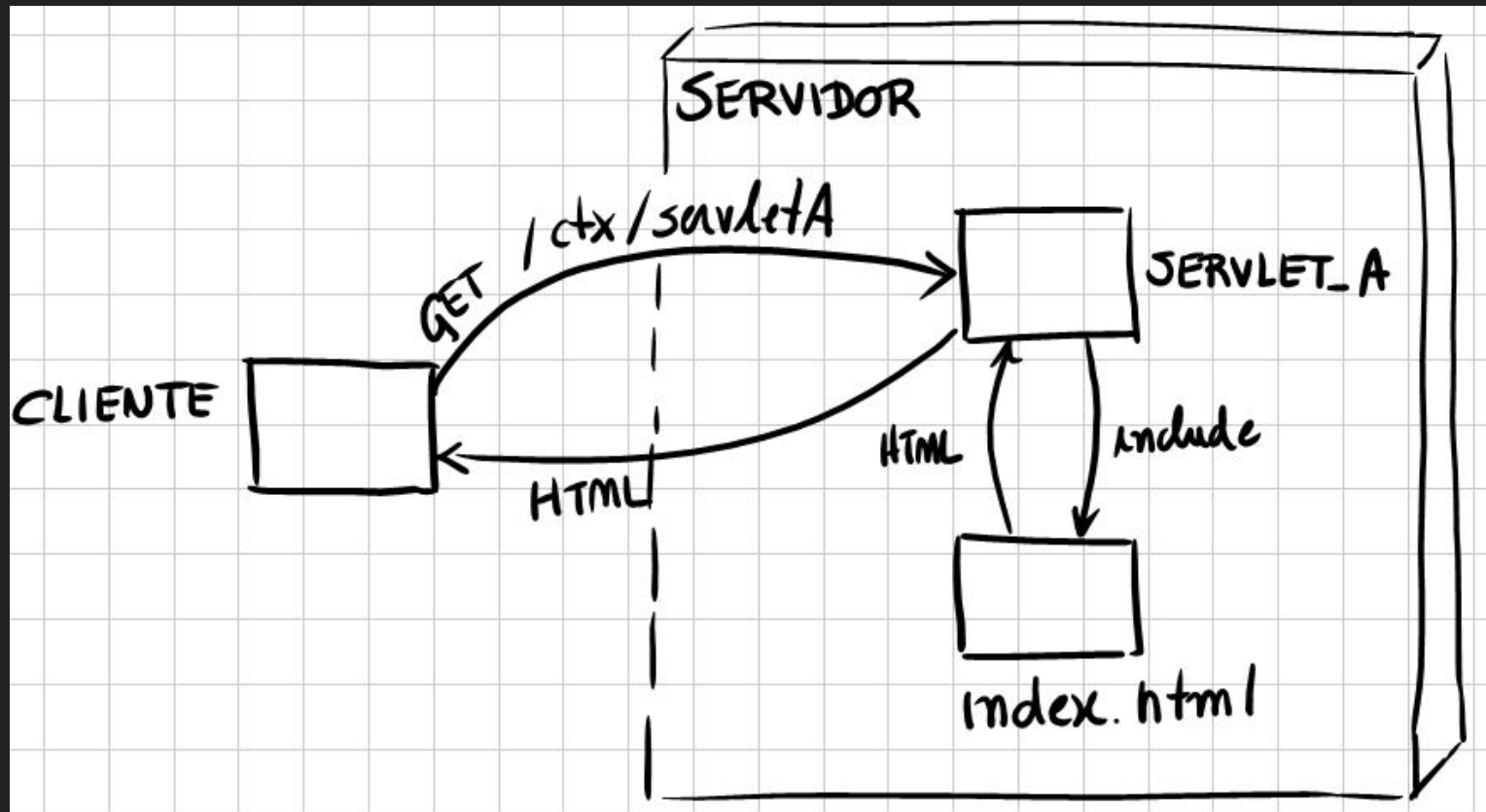
Redireccionamiento



Encaminhamento



Inclusão



Navegação

Demonstração 7

Compartilhamento de informações

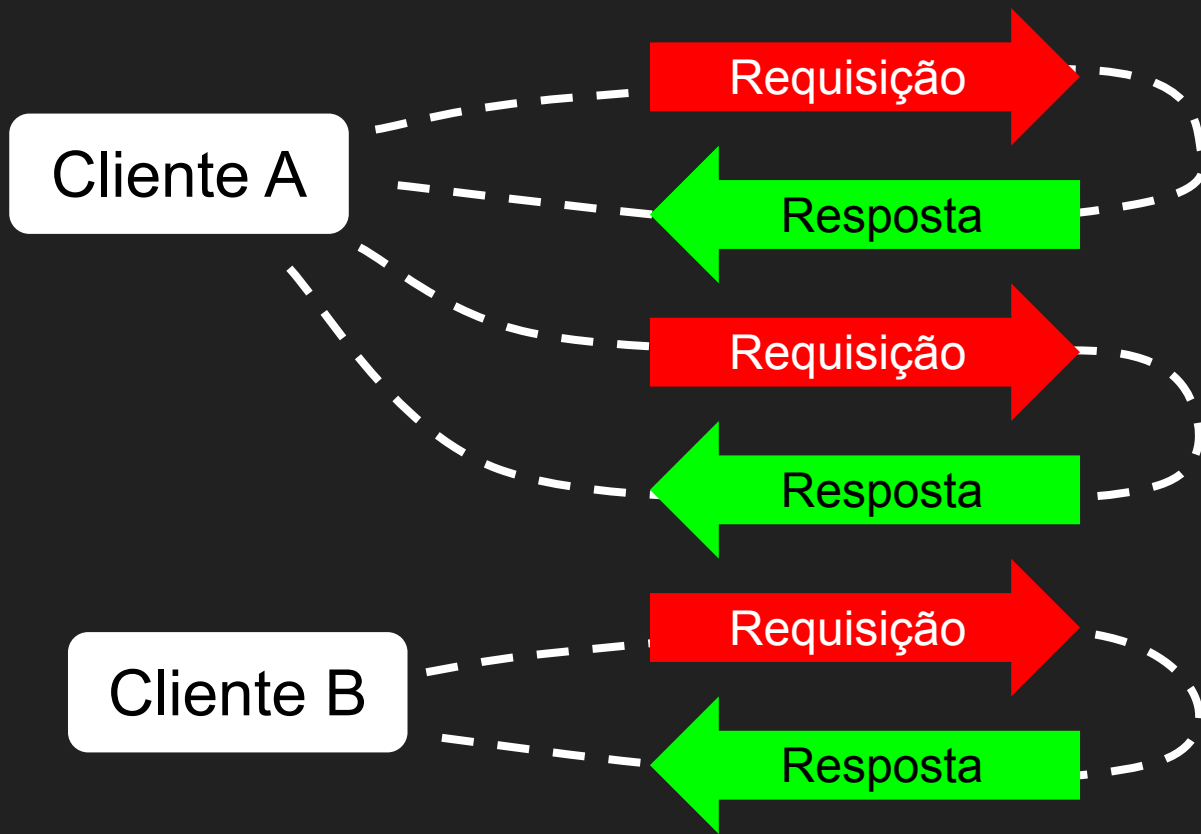
- É possível compartilhar informações entre servlets que colaboram
- **Várias situações:**
 - Um servlet pode realizar algum cálculo e deixar o valor disponível para os demais
 - Um servlet pode armazenar um valor temporário
 - Um servlet pode passar um valor para outro servlet específico
- Exemplos: usuário logado, carrinho de compras, mensagens de erro, etc.

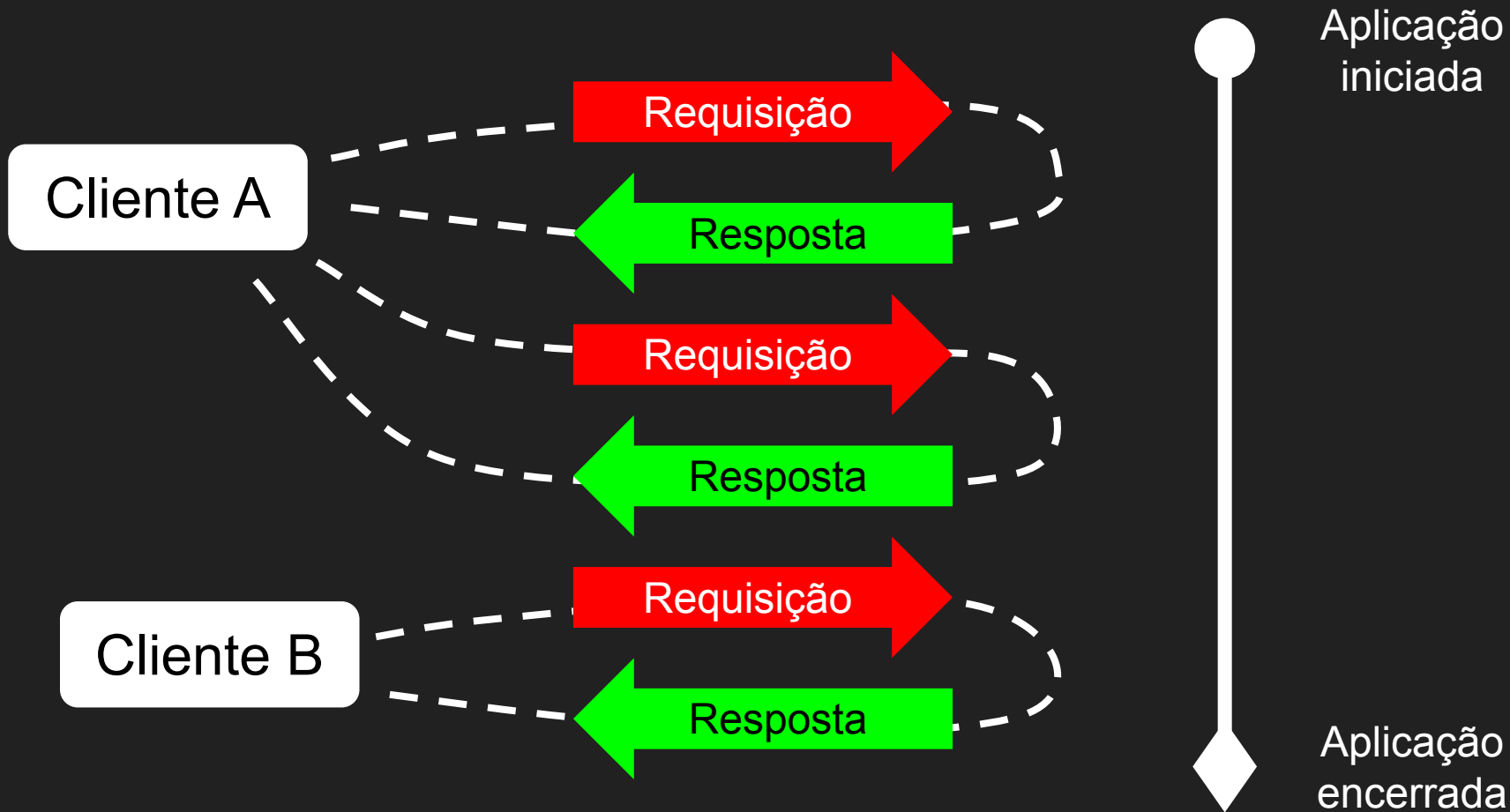
Objetos de escopo

- Existem 3 diferentes escopos para compartilhamento de informação:
 - Requisição
 - Sessão
 - Aplicação / contexto

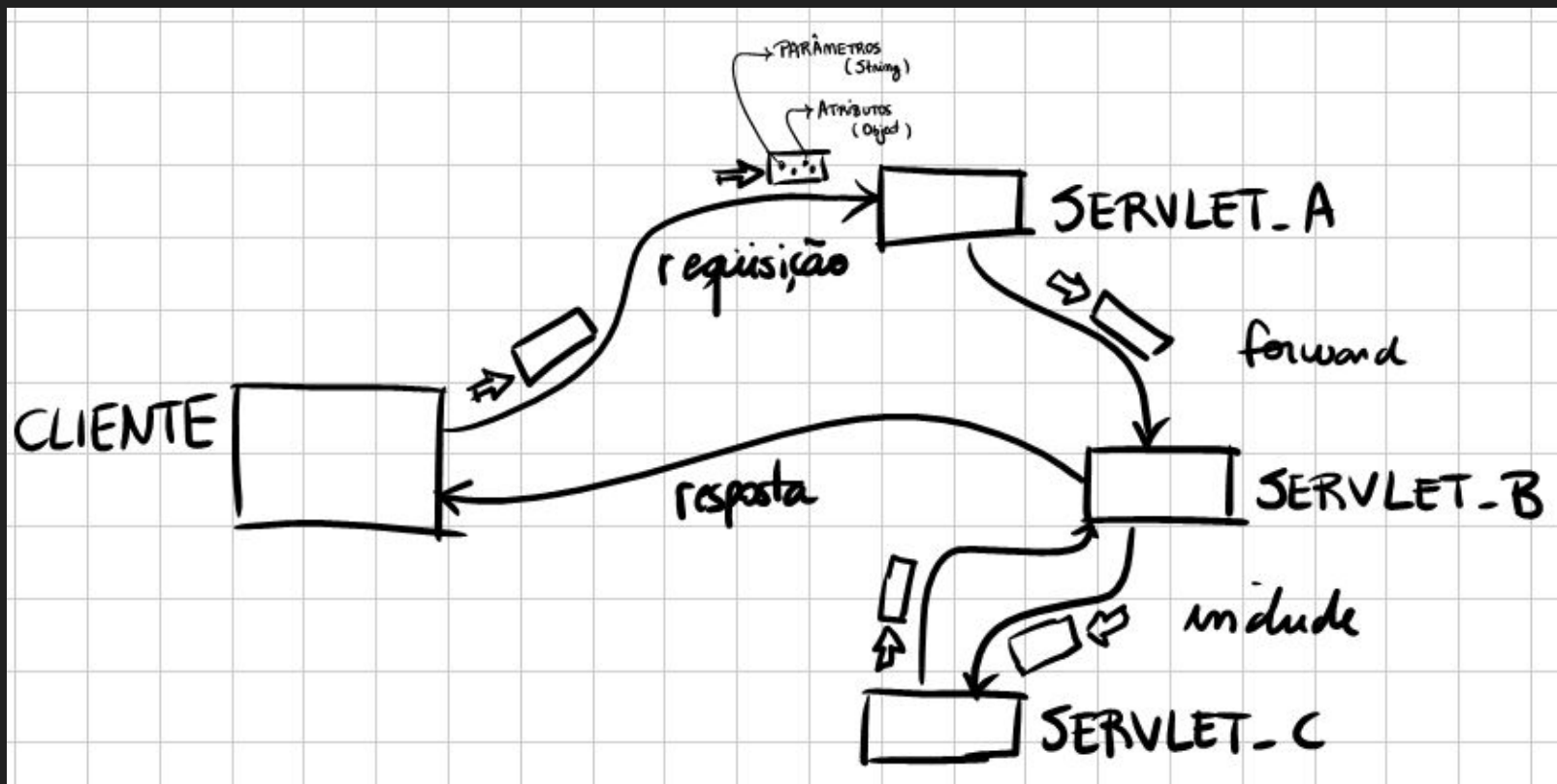
Requisição < Sessão < Aplicação



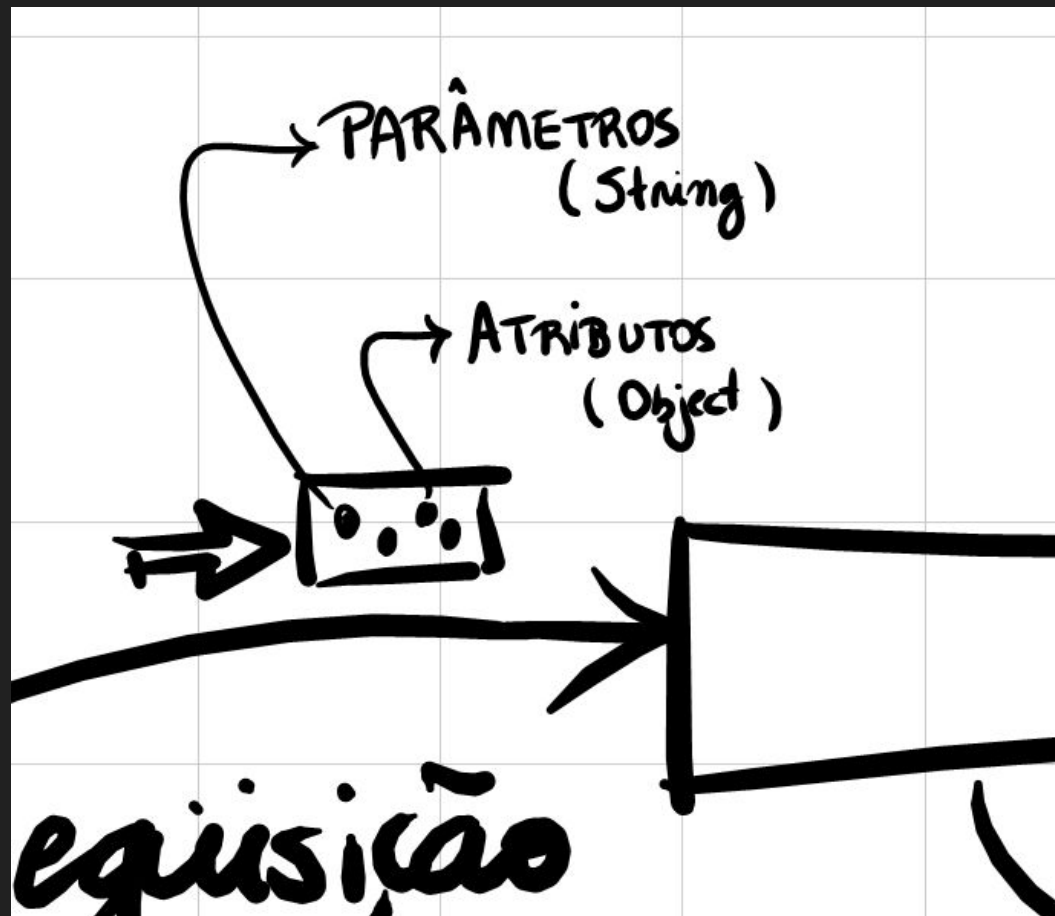




Requisição



Requisição



Escopo de requisição

- `String request.getParameter(String);`
 - Os parâmetros vindos da requisição original são levados por toda a cadeia de tratadores
- `void request.setAttribute(String name, Object o);`
 - Armazena um objeto na requisição, associado a um nome
- `Object request.getAttribute(String name);`
 - Recupera um objeto da requisição, pelo nome
 - Observe que é necessário fazer “casting”
- `void request.removeAttribute(String name);`
 - Remove um objeto da requisição, pelo nome

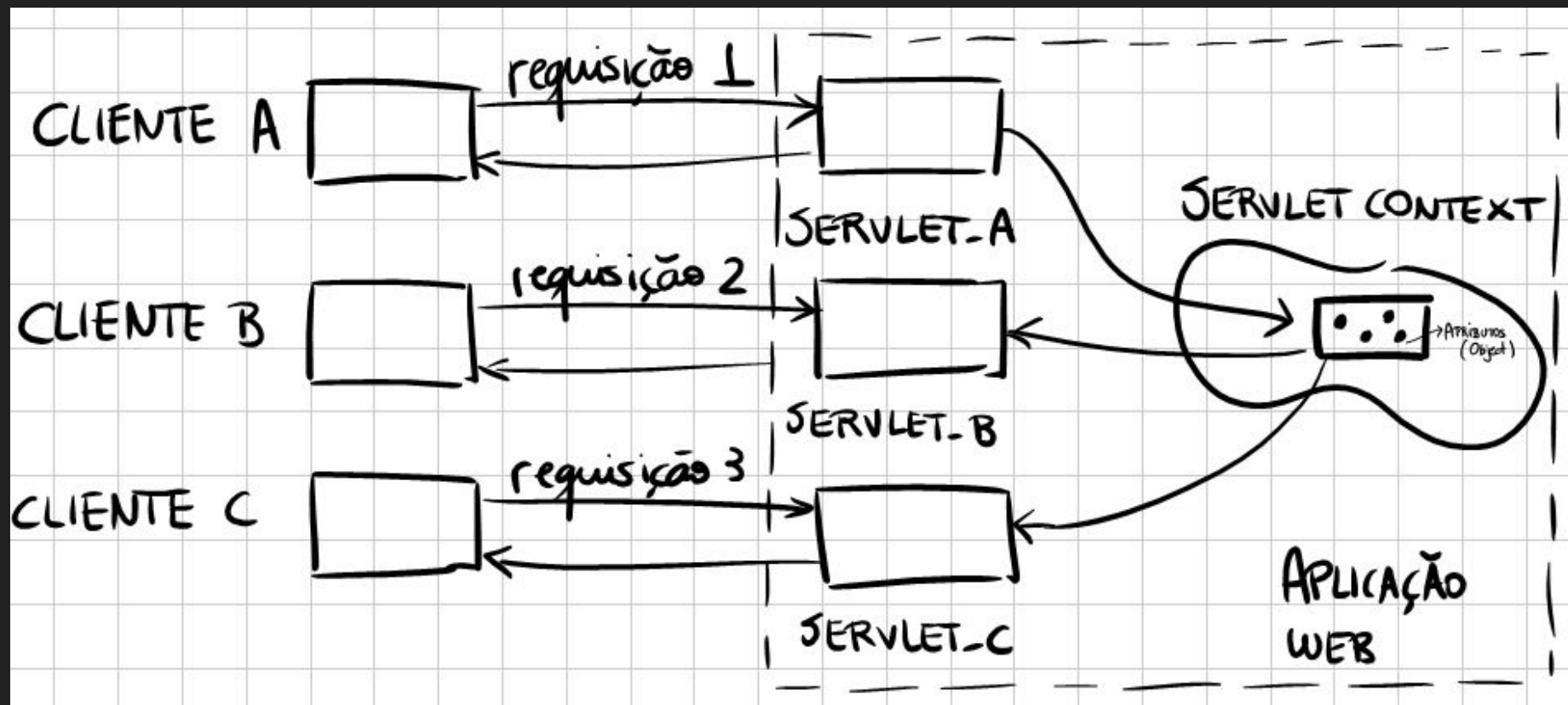
Escopo de requisição

- Atributos e parâmetros são apagados entre diferentes requisições
- São usados junto com forward e include
- Caso seja feito um redirect, atributos e parâmetros são perdidos

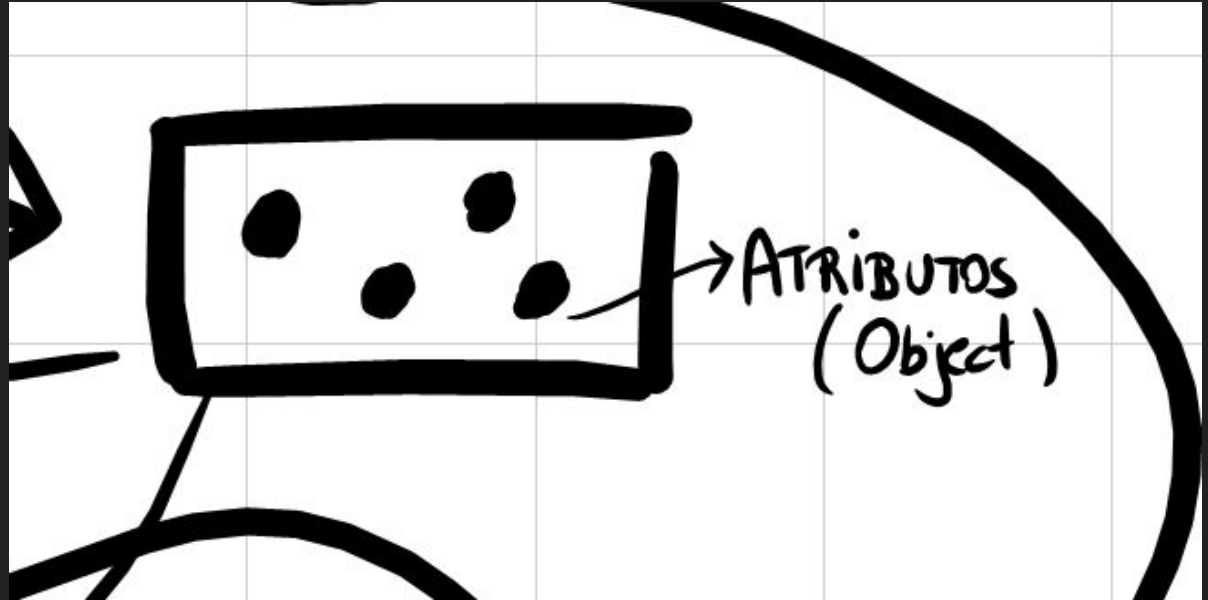
Escopo de requisição

Demonstração 8

Escopo de aplicação



Escopo de aplicação



Escopo de aplicação

- `ServletContext ctx = getServletContext();`
 - Recupera o contexto de um servlet (this)
- `void ctx.setAttribute(String name, Object o);`
 - Armazena um objeto no contexto, associado a um nome
- `Object ctx.getAttribute(String name);`
 - Recupera um objeto do contexto, pelo nome
 - Observe que é necessário fazer “casting”
- `void ctx.removeAttribute(String name);`
 - Remove um objeto do contexto, pelo nome

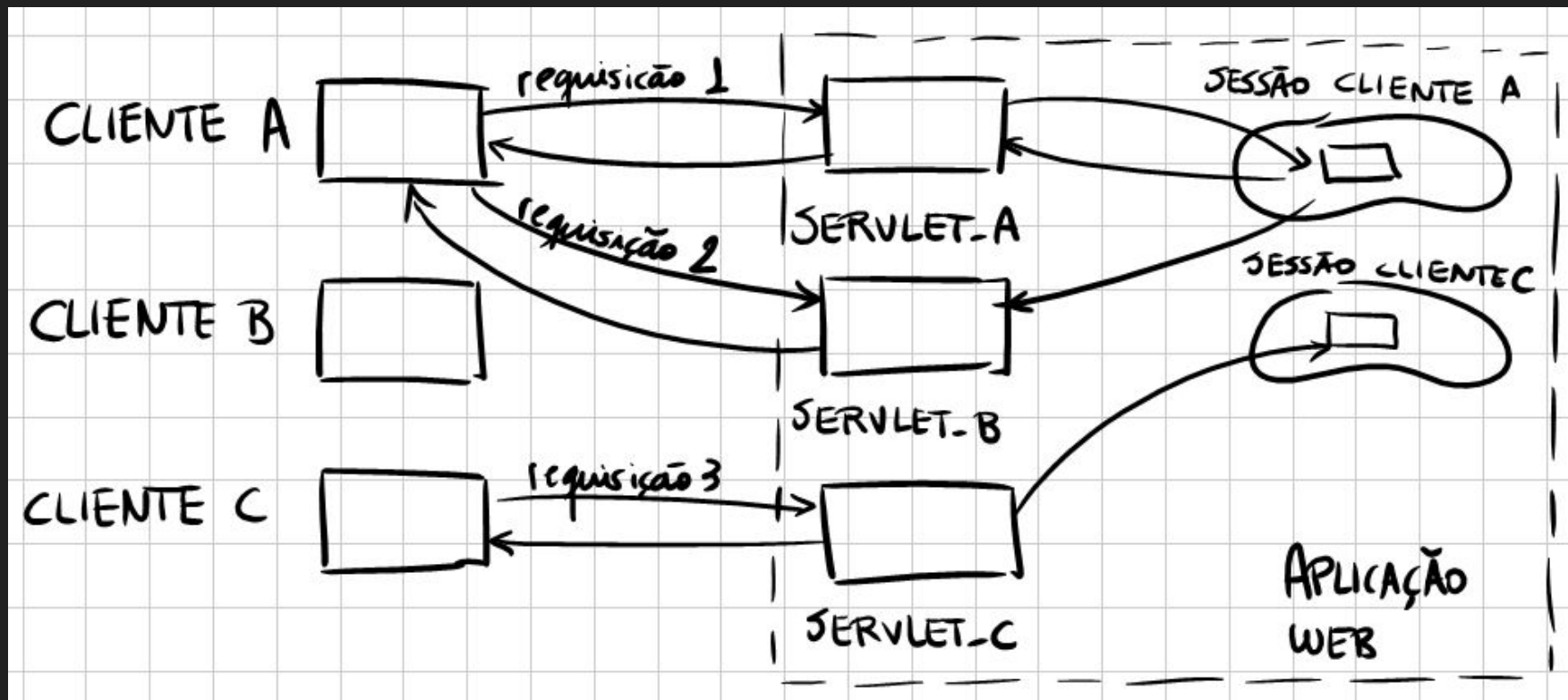
Escopo de aplicação

- Atributos são mantidos entre diferentes requisições, e compartilhados entre diferentes clientes
- Atributos são mantidos enquanto a aplicação estiver rodando
- Normalmente são usados para configurações globais da aplicação ou para implementar o padrão Singleton

Escopo de aplicação

Demonstração 9

Escopo de sessão



Escopo de sessão

- `HttpSession session = request.getSession();`
 - Recupera a sessão associada a uma requisição
- `void session.setAttribute(String name, Object o);`
 - Armazena um objeto na sessão, associado a um nome
- `Object session.getAttribute(String name);`
 - Recupera um objeto da sessão, pelo nome
 - Observe que é necessário fazer “casting”
- `void session.removeAttribute(String name);`
 - Remove um objeto da sessão, pelo nome

Escopo de sessão

- Objetos ficam armazenados em uma área que é específica para cada cliente
 - Permite manter ou “lembrar” o estado do cliente
- É usado para armazenar informações personalizadas e melhorar a interatividade
- Exemplos típicos:
 - Usuário logado
 - Carrinho de compras

Escopo de sessão

- Sessões expiram depois de um certo tempo de inatividade
- Pode ser configurado no arquivo web.xml

```
<session-config>  
  <session-timeout>  
    30  
  </session-timeout>  
</session-config>
```

- Nesse exemplo, após 30 minutos de inatividade, a sessão expira, e os atributos armazenados são perdidos

Escopo de sessão

Demonstração 10

Fim