

DESENVOLVIMENTO DE SOFTWARE PARA A WEB 1

Módulo 2 Java Servlets

Delano Medeiros Beder
delano@dc.ufscar.br

Java Servlet

- Um *Servlet* é uma classe Java que segue a especificação API **Java Servlet** (do pacote **javax.servlet**) a qual proporciona ao desenvolvedor a possibilidade de adicionar conteúdo dinâmico (acesso a banco de dados etc) em um servidor Web usando a plataforma Java.
- O nome *Servlet* vem da ideia de um pequeno servidor cujo objetivo é receber chamadas HTTP, processá-las e devolver uma resposta ao cliente.
- Para que *Servlets* sejam executados, é necessária a utilização de um **container de servlets**, que é um componente de um servidor Web o qual interage com Java *Servlets*.
 - Tomcat – software livre desenvolvido pela Apache Software Foundation

"Alô mundo"

Primeiros passos

Como instalar um servidor Tomcat

Como fazer uma primeira aplicação

Como implantar a aplicação



Instalação e execução Tomcat

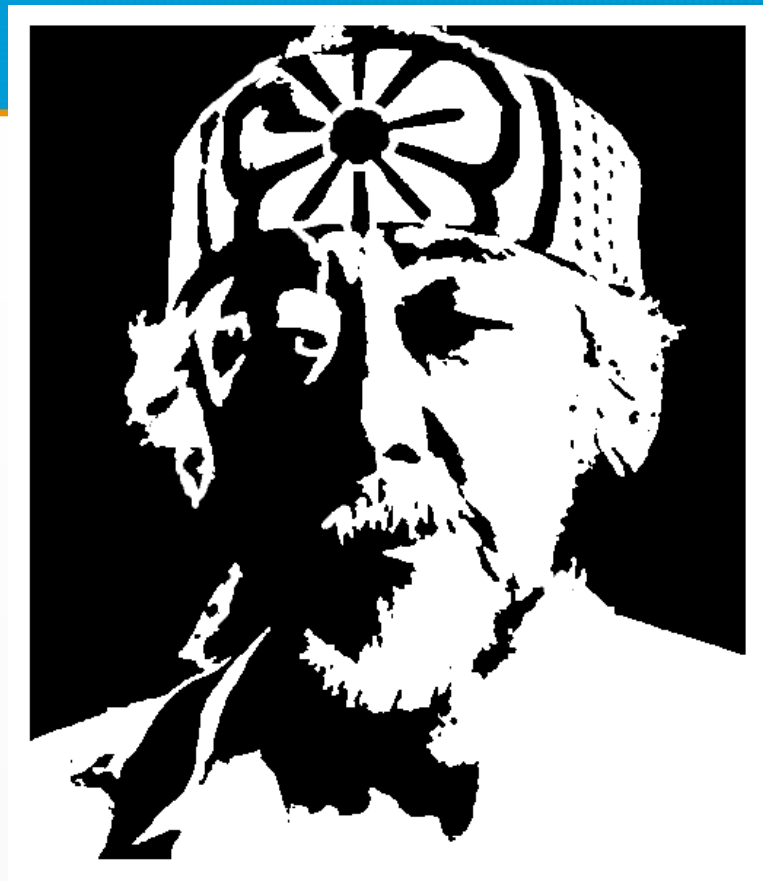
<http://tomcat.apache.org/>

Demonstração 1

Primeira aplicação

Utilizando linha de comando apenas

Depois veremos meios mais produtivos



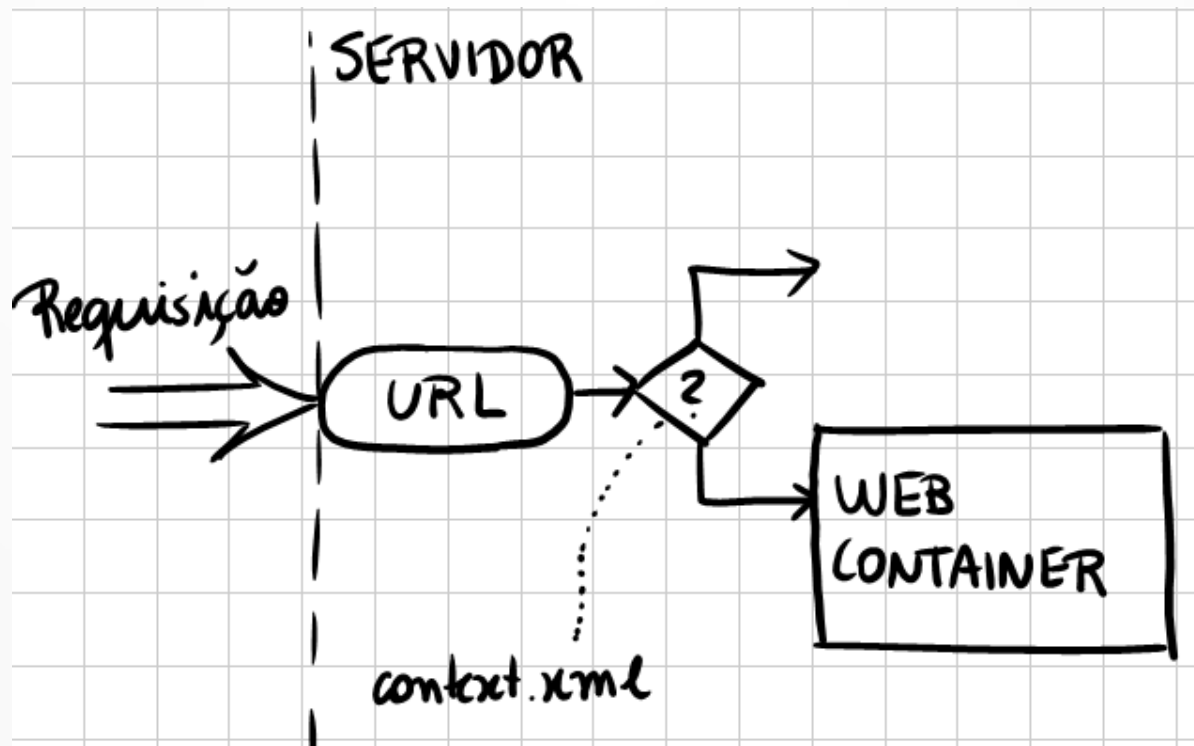
"Alô mundo"

Demonstração 2

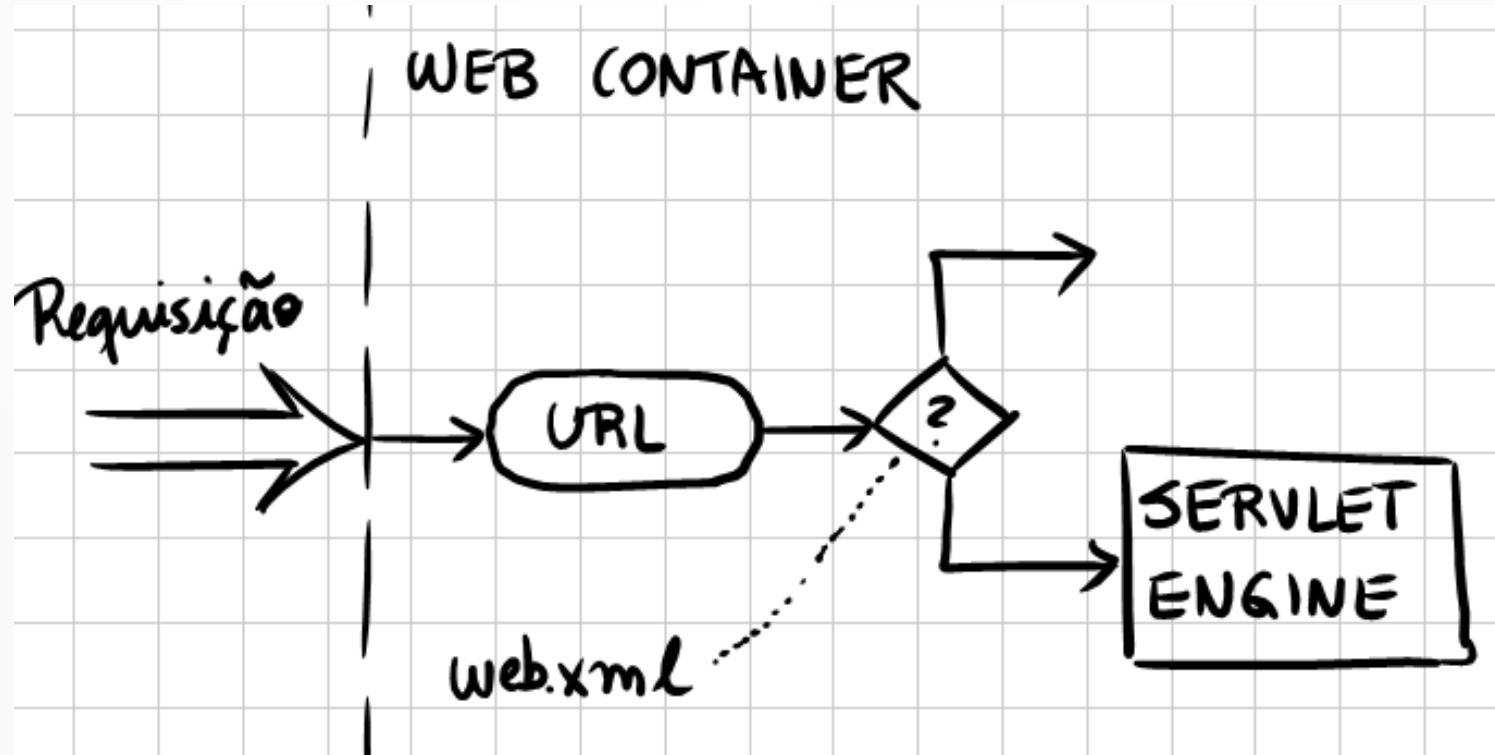
"Alô mundo" - Maven

Demonstração 3

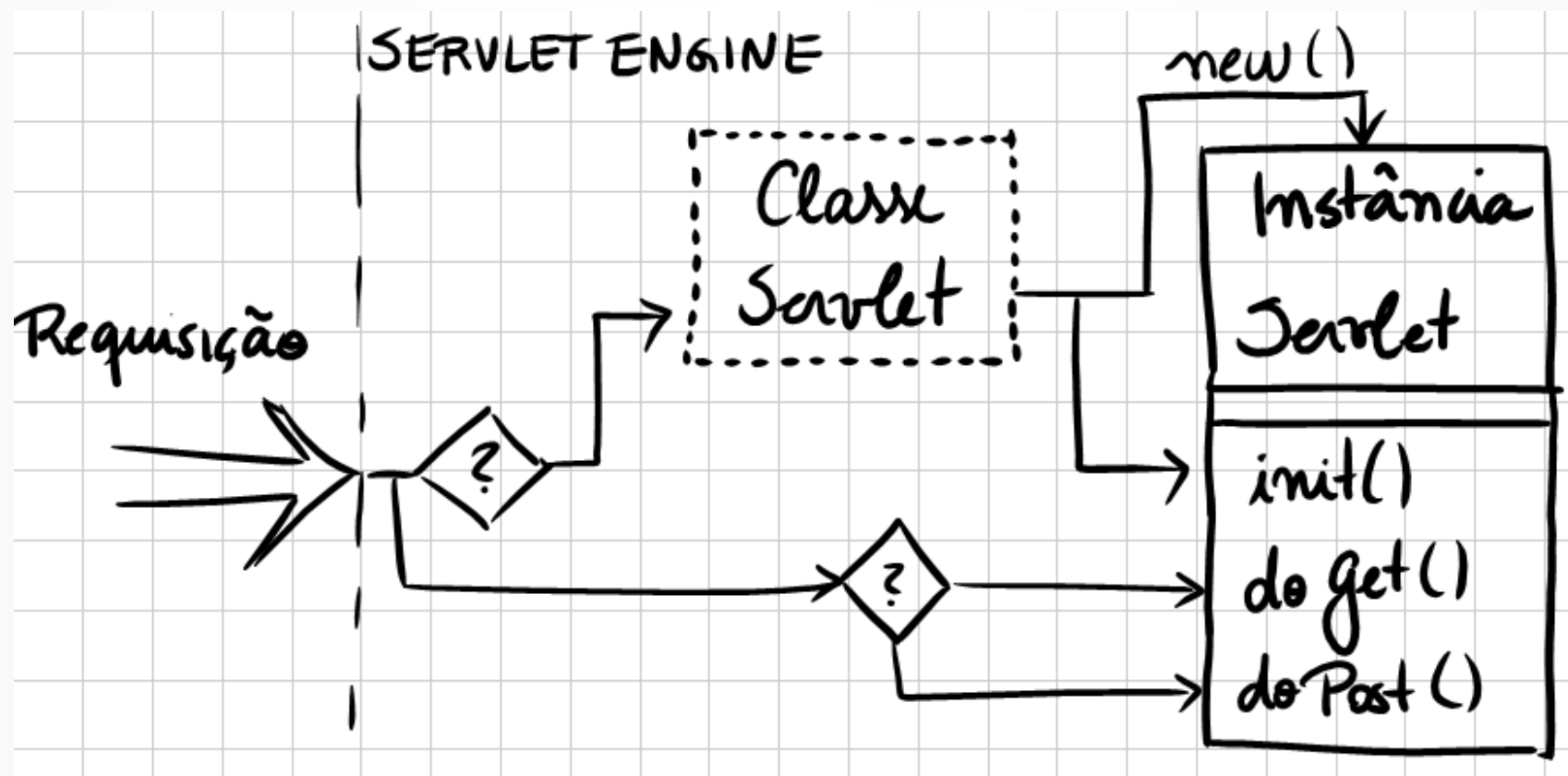
Ciclo de vida de um Servlet



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`)

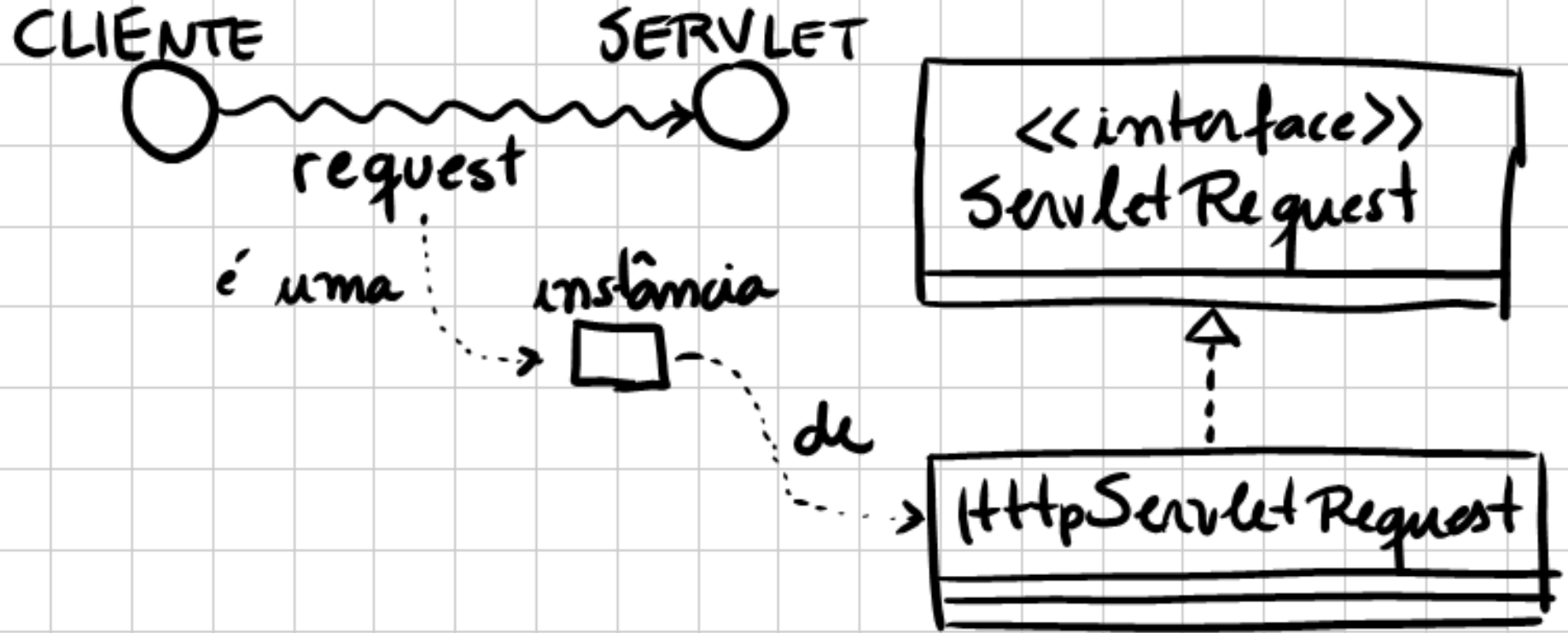
Procedimento correto:

- 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



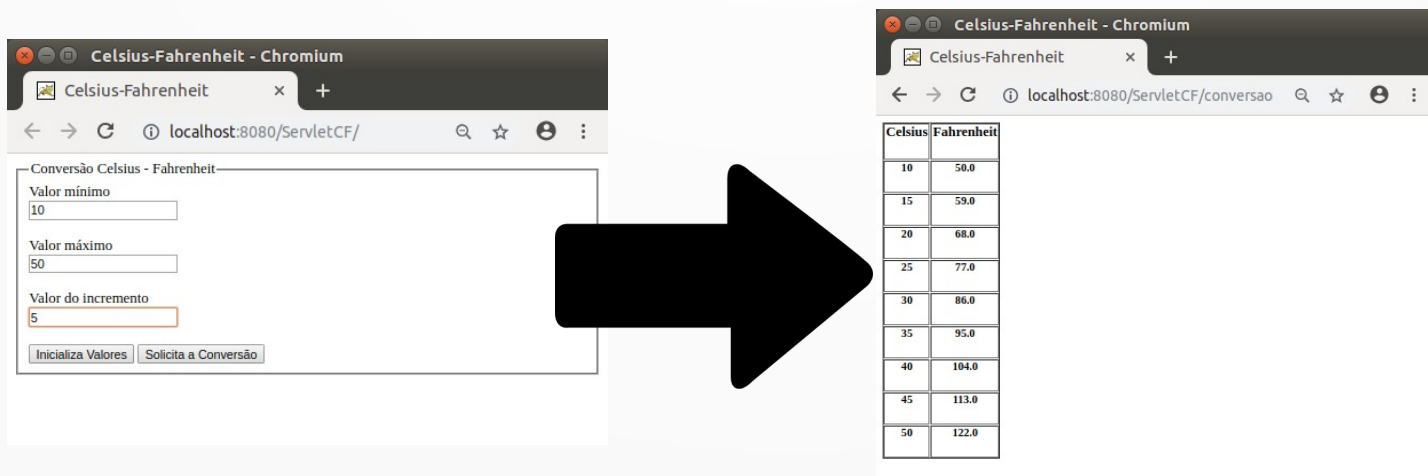
Tratamento de uma requisição

Demonstração 4

Exercício (1)

- Escreva uma aplicação web, utilizando a tecnologia Java Servlet, que apresente uma tabela de conversão de Graus Celsius (°C) para Graus Fahrenheit (°F). A faixa de valores (mínimo e máximo), bem como o incremento entre valores consecutivos na tabela, podem ser passados via formulário HTML. Caso os parâmetros não sejam passados, considere os seguintes valores padrões (mínimo = -100, máximo = 100 e incremento = 5).

Utilize a seguinte fórmula para a conversão: $^{\circ}\text{F} = ^{\circ}\text{C} * 1.80 + 32$

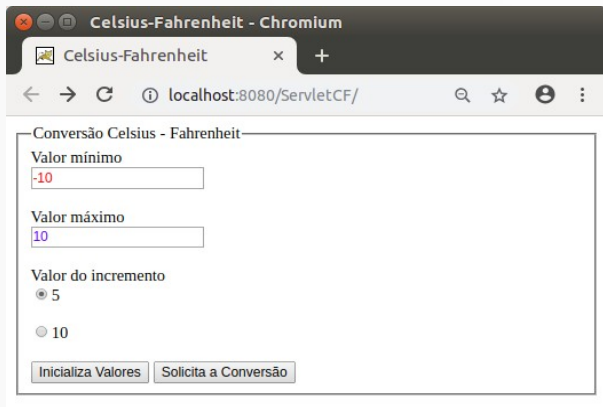


The screenshot shows a web browser window titled "Celsius-Fahrenheit - Chromium". The address bar shows "localhost:8080/ServletCF/". The page content is divided into two main sections. The left section is a form titled "Conversão Celsius - Fahrenheit" with three input fields: "Valor mínimo" (10), "Valor máximo" (50), and "Valor do incremento" (5). Below these fields are two buttons: "Inicializa Valores" and "Solicita a Conversão". A large black arrow points from this form to the right section. The right section displays a table with two columns: "Celsius" and "Fahrenheit". The table contains 11 rows of data, starting from 10 Celsius and increasing by 5 units up to 50 Celsius.

Celsius	Fahrenheit
10	50.0
15	59.0
20	68.0
25	77.0
30	86.0
35	95.0
40	104.0
45	113.0
50	122.0

Exercício (2)

- Refatore a aplicação web do exercício anterior para atender as seguintes restrições:
 - Valor mínimo deve ser negativo e valor máximo deve ser positivo
Dica: Use Javascript para validar os dados do formulário antes do envio.
 - Valor do incremento deve ser selecionado entre 2 opções: 5 e 10
 - Personalize a aparência da tabela através de um estilo CSS



Celsius-Fahrenheit - Chromium

Celsius-Fahrenheit

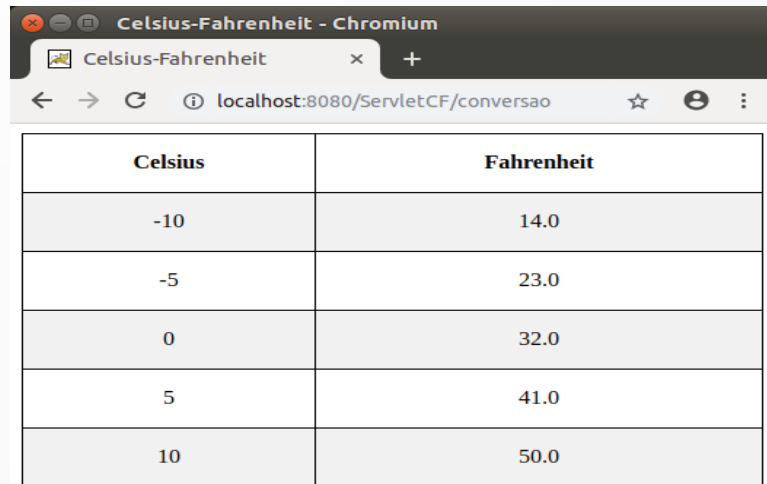
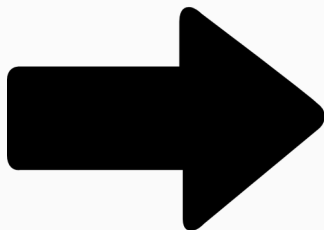
localhost:8080/ServletCF/

Conversão Celsius - Fahrenheit

Valor mínimo
-10

Valor máximo
10

Valor do incremento
☒ 5
☐ 10



Celsius-Fahrenheit - Chromium

Celsius-Fahrenheit

localhost:8080/ServletCF/conversao

Celsius	Fahrenheit
-10	14.0
-5	23.0
0	32.0
5	41.0
10	50.0

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

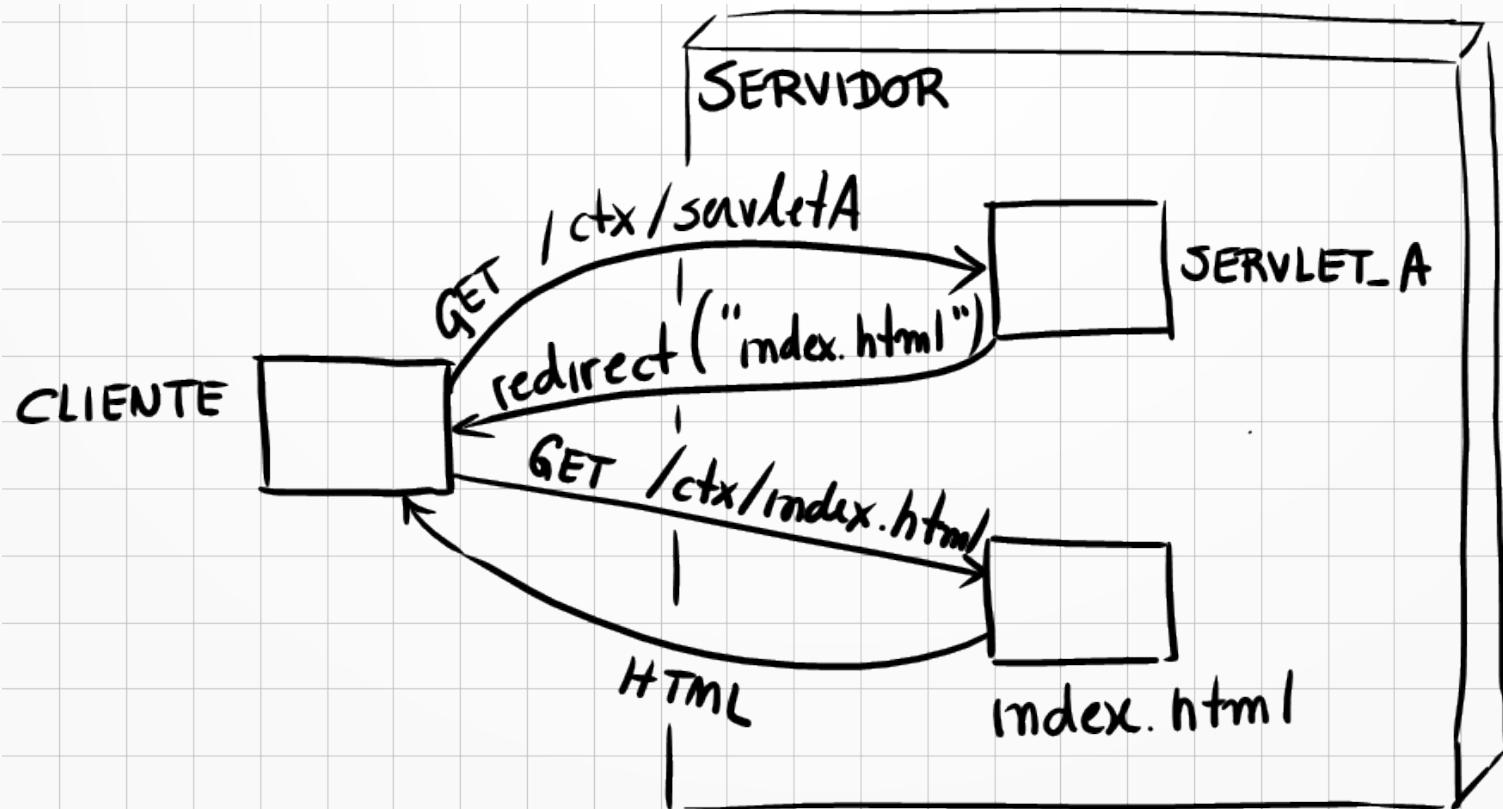
Navegação

- Redirecionamento x Encaminhamento

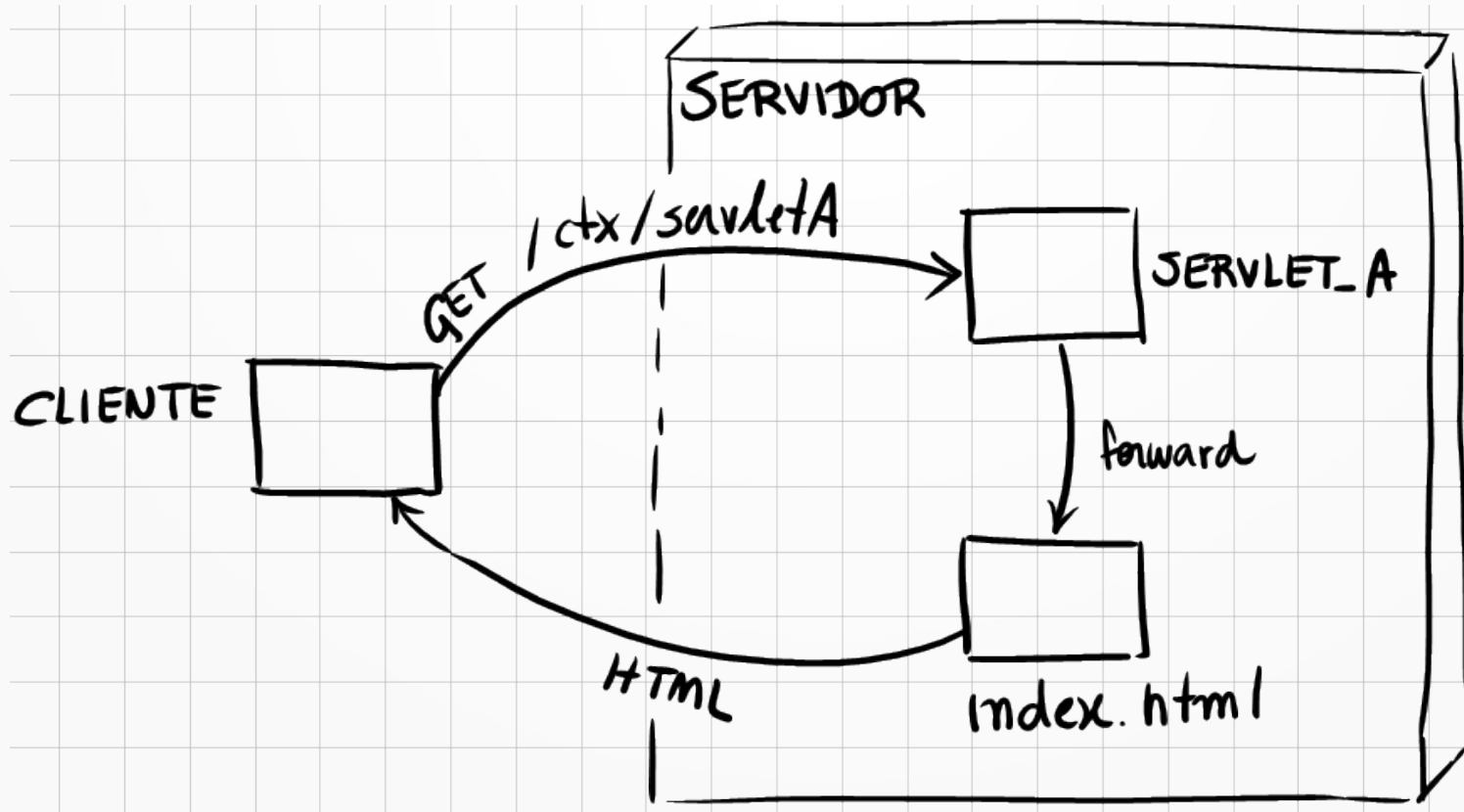
No primeiro caso (Redirect), o cliente receberá uma resposta http em cujo header haverá a informação de que ele deve requisitar outra página, e o browser fará esta requisição. Ou seja, o redirecionamento ocorre no lado no cliente.

No segundo caso (Forward), no lado do server a requisição do usuário será encaminhada para ser atendida por outro recurso (outro servlet). Este outro servlet eventualmente devolverá outra página para o usuário.

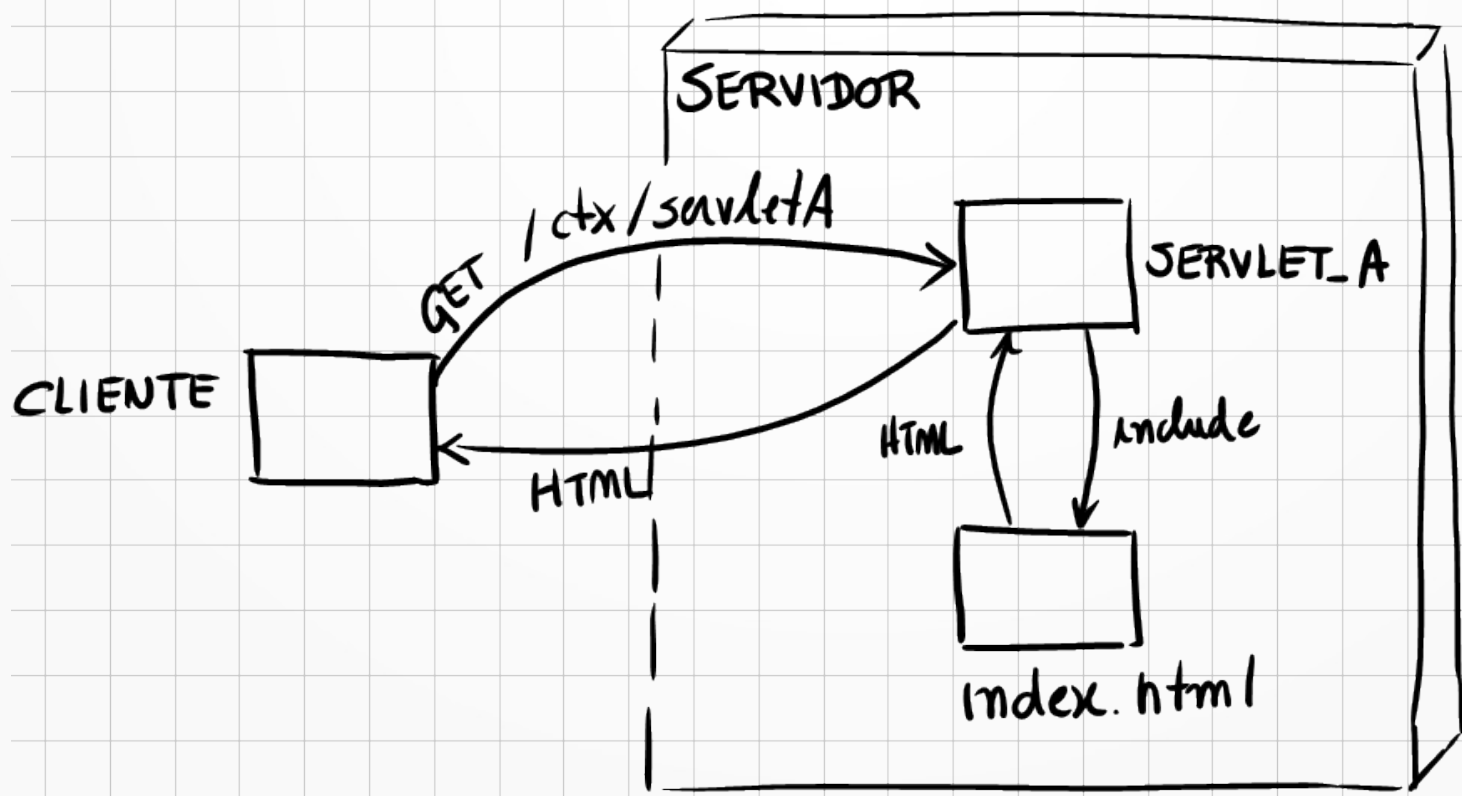
Redireccionamiento



Encaminhamento



Inclusão



Navegação

Demonstração 5

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

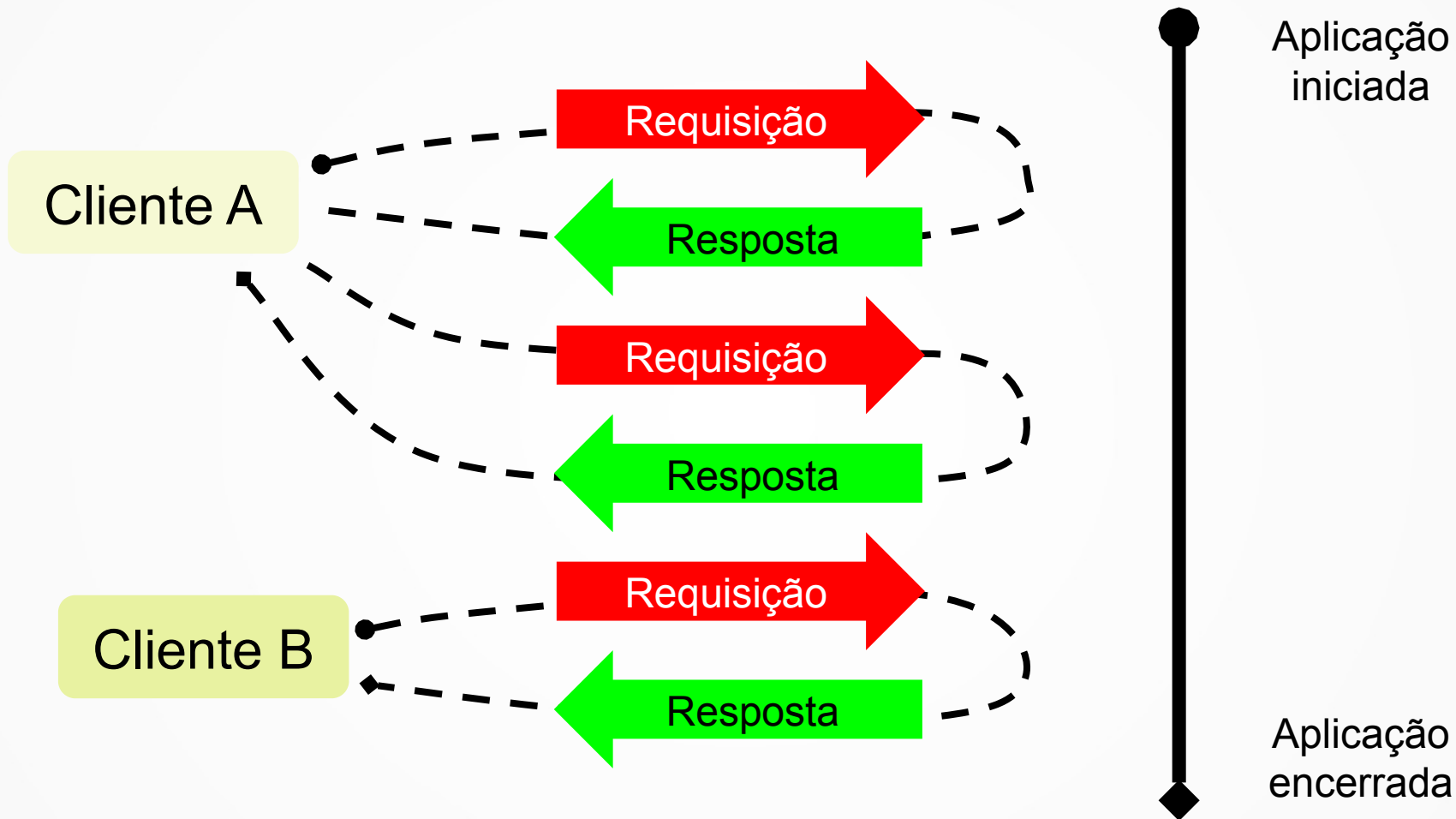
Requisição < Sessão < Aplicação



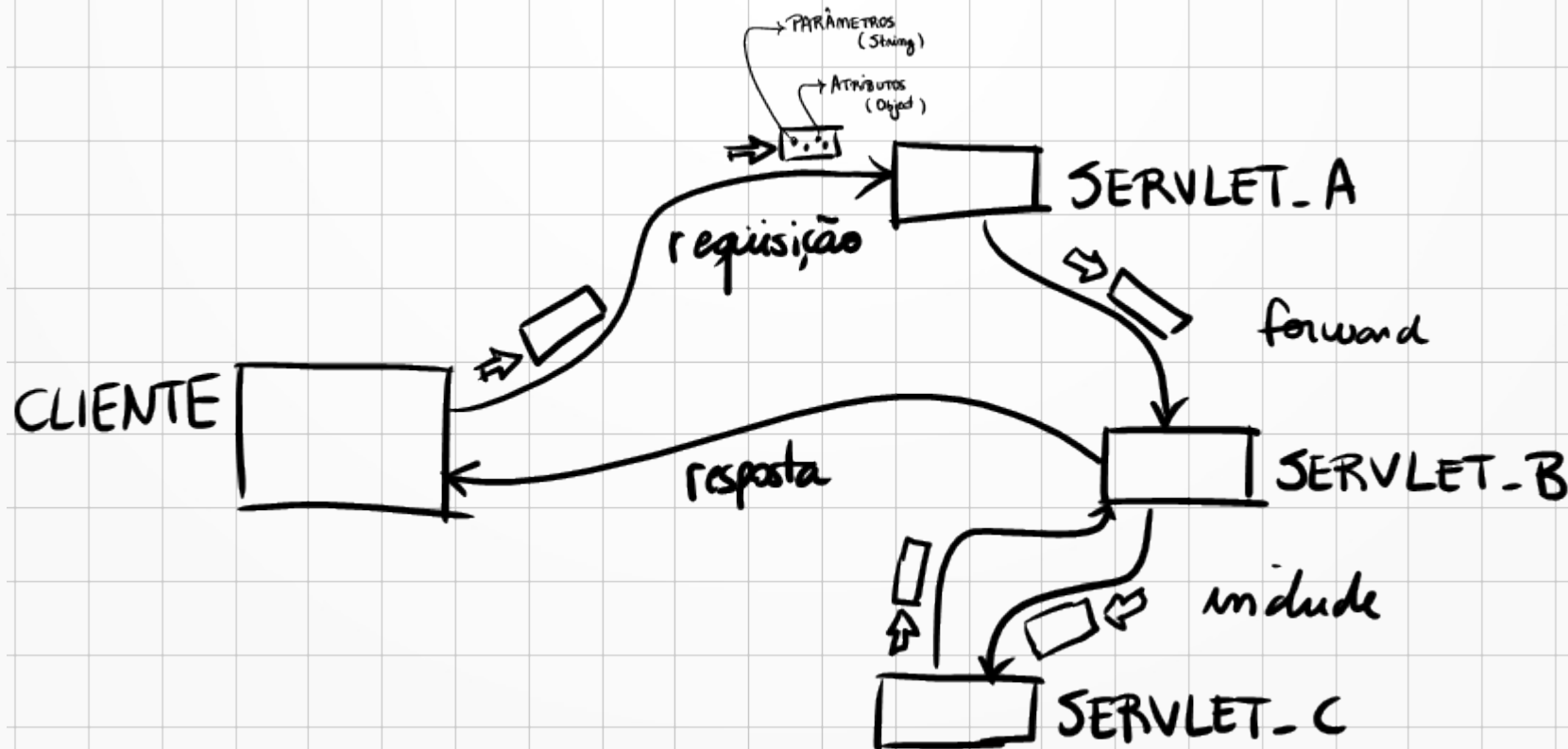
A diagram illustrating a request-response cycle. It features two large, thick arrows. The top arrow is red and points to the right, containing the word 'Requisição' in white. The bottom arrow is green and points to the left, containing the word 'Resposta' in black. The arrows are positioned such that they appear to be part of a continuous loop, with the red arrow starting from the left and the green arrow returning to the left. The background is a light gray with a faint, circular, dashed-line pattern.

Requisição

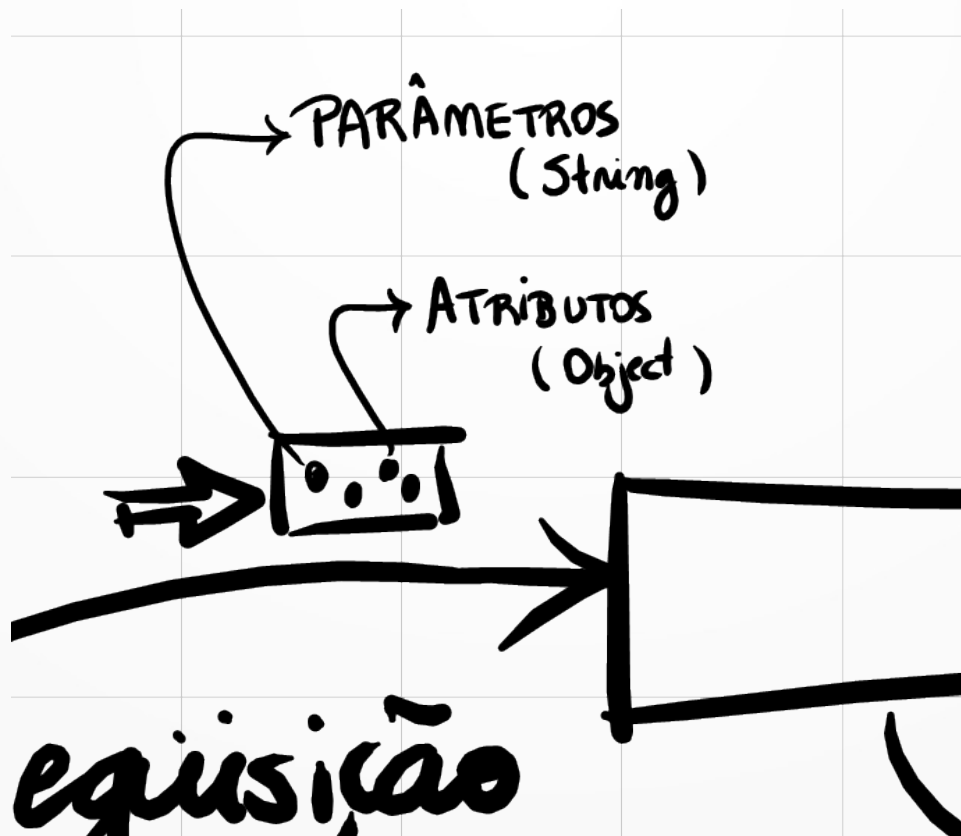
Resposta



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

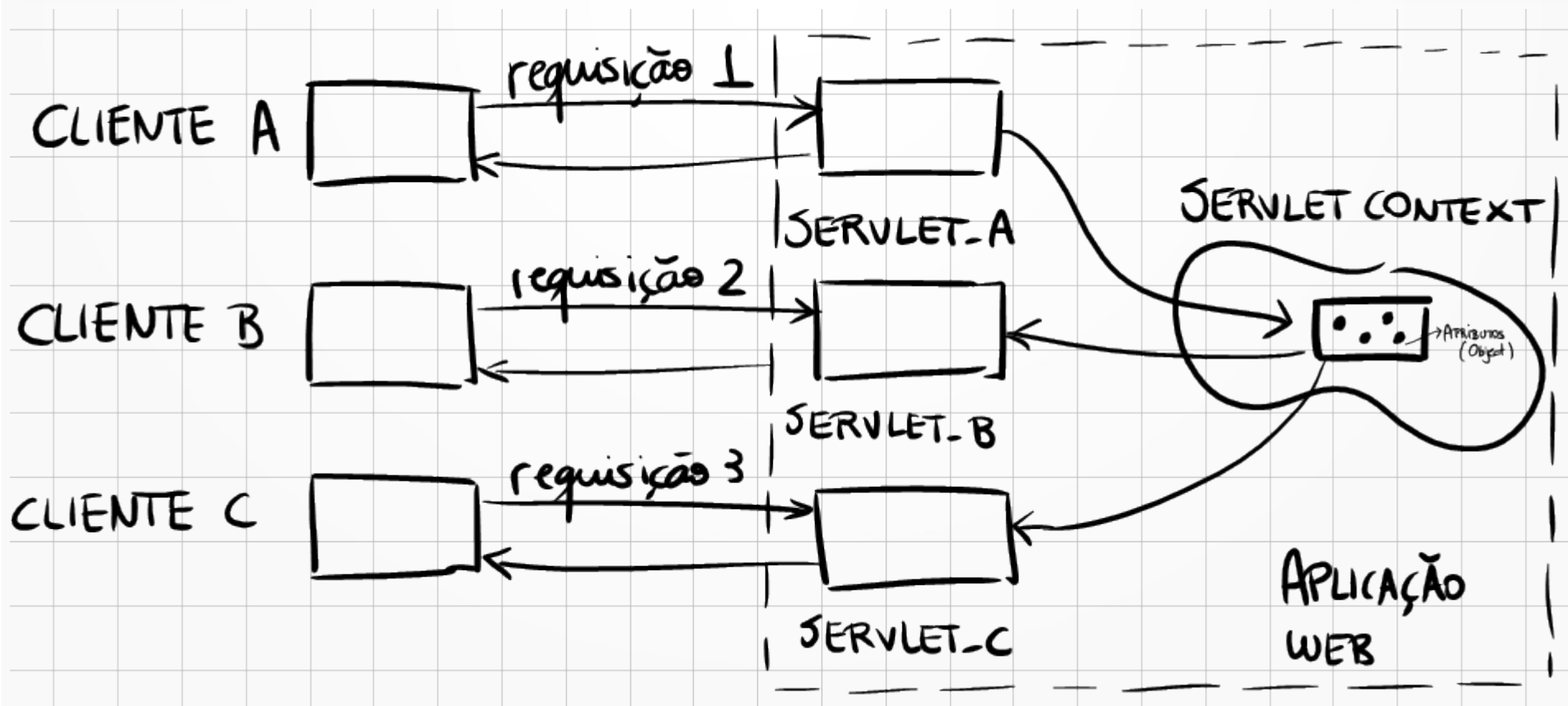
Escopo de requisição

- 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

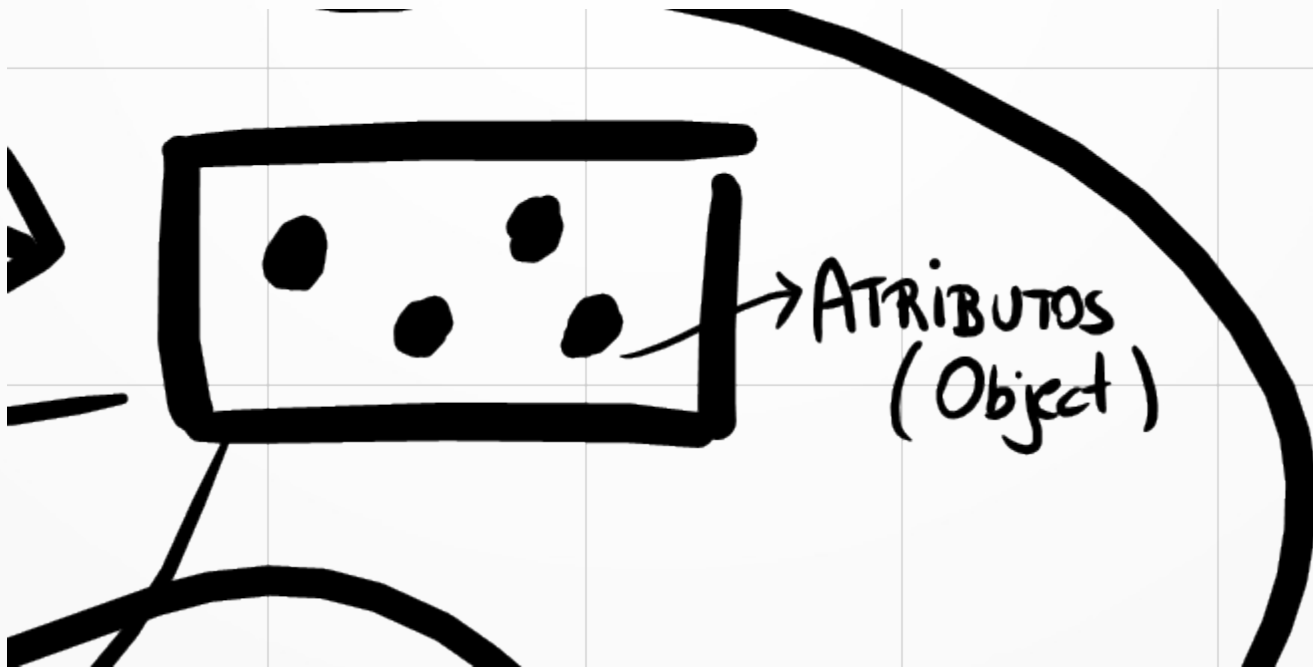
Escopo de requisição

Demonstração 6

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

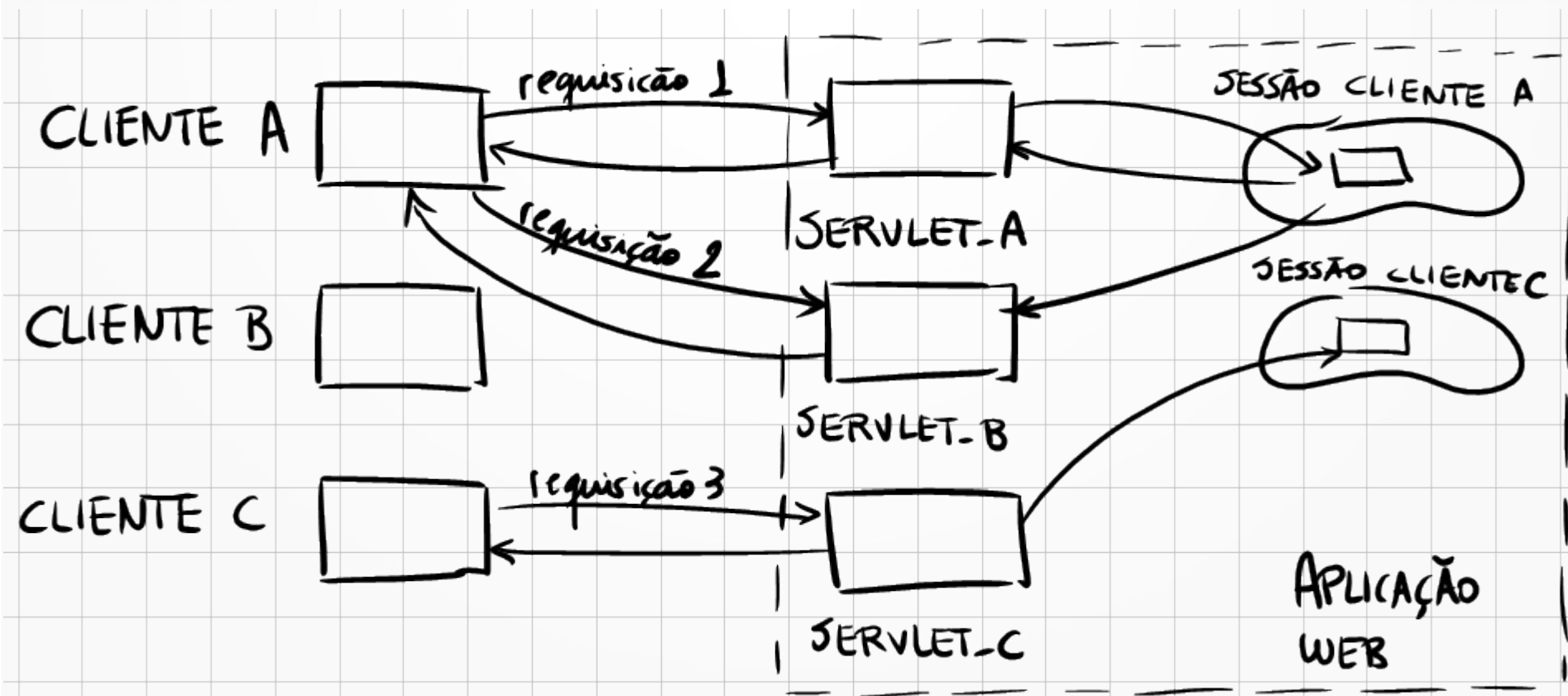
Escopo de aplicação

- 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 7

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 8

FIM