

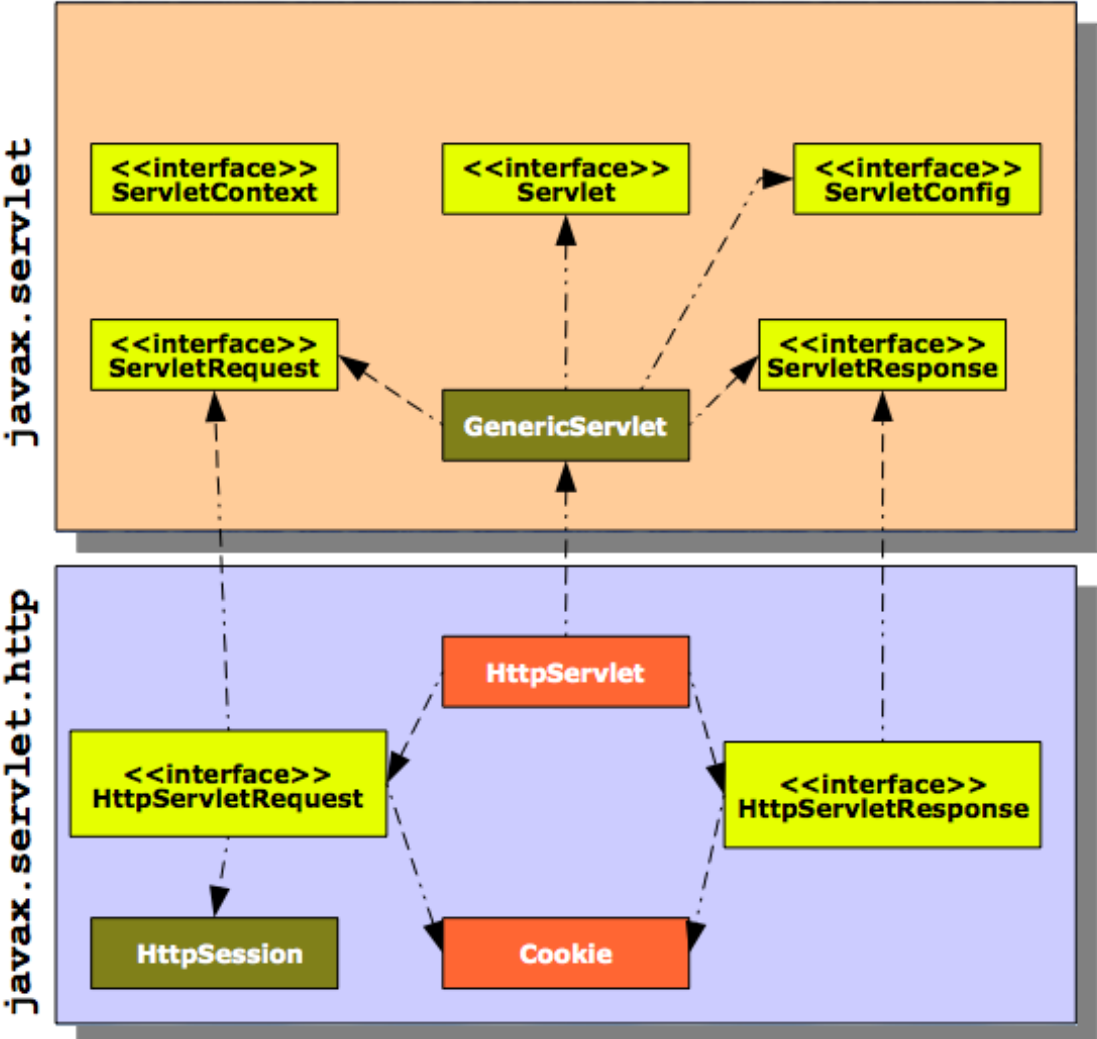


# **SERVLETS CICLO DE VIDA**

**José Ricardo da Silva Junior © 2011**

**[joserickardo.jr@gmail.com](mailto:joserickardo.jr@gmail.com)**

# API SERVLET



# CICLO DE VIDA

- O container é responsável pelo ciclo de vida de um **servlet**
- Ao receber uma requisição, o servidor a envia para o container, a qual é delegada a um **servlet**.
- Dessa forma, o container é responsável por:
  - Carrega a classe na memória (se já não tiver sido carregada)
  - Cria uma instância da classe do servlet (se não houver instancia)
  - Inicializa a instância chamando o método **init()**
- Após a criação do servlet, cada requisição é processada pelo método **service()** do servlet, a qual é chamada pelo container
  - O container cria um objeto de **requisição** (ServletRequest) e de **resposta** (ServletResponse) e depois chama **service()** passando os objetos como parâmetros
  - Quando a resposta é enviada, os objetos são **destruídos**
- Após a finalização do **servlet**, o método **destroy()** do servlet é invocado e o mesmo é removido da memória..

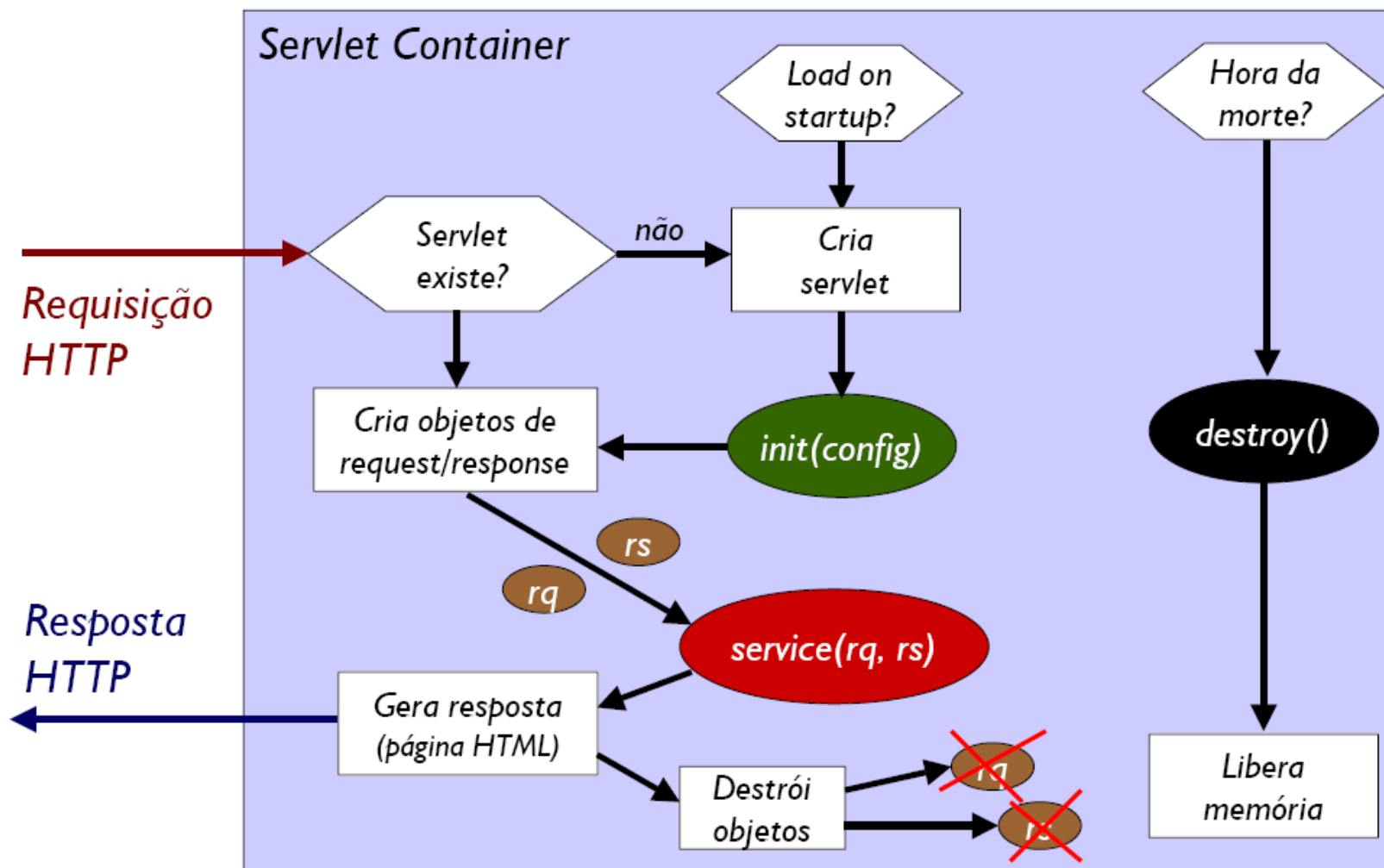


## CICLO DE VIDA

- Método `init()`: chamado pelo container do servlet no momento de sua criação
- Método `service()`: chamado a cada requisição do cliente
- Método `destroy()`: chamado quando um servlet é destruído pelo container.



# CICLO DE VIDA



# INICIALIZAÇÃO DE UM SERVLET

- Deve-se sobrepor `init(config)` com instruções que serão realizadas para inicializar um servlet
  - Carregar parâmetros de inicialização, dados de configuração
  - Obter outros recursos (abrir banco de dados, etc)
- Falha na inicialização deve provocar `UnavailableException` (subclasse de `ServletException`)

```
public void init(ServletConfig config)
    throws ServletException {

    String contador = config.getInitParameter("contador");
    if (contador == null) {
        throw new UnavailableException(
            "Configuração incorreta!");
    }
}
```



# FINALIZAÇÃO

- Durante a remoção do servlet da memória, é chamado o método `destroy()` do servlet.
  - Geralmente é feita a “limpeza no sistema”
- O servlet geralmente só é destruído quando todos os seus métodos `service()` terminaram (ou depois de um timeout)
  - Se sua aplicação tem métodos `service()` que demoram para terminar, você deve garantir um shutdown limpo.

```
public void destroy() {  
    banco.close();  
    banco = null;  
}
```



# UTILIZANDO O ECLIPSE

- Configurando o Eclipse
- Criando um webapp
- Criando um servlet



# EXERCÍCIOS

- Criar um servlet que exiba a mensagem “Hello World”.
- Criar um servlet contador, o qual exibe o número de requisições efetuadas.
- Criar um servlet que possa receber várias requisições e seja capaz de imprimir o endereço IP do cliente que o está acessando. Para isto, utilize o método `getRemoteAddr()` de `request`, que retorna uma string com o endereço do cliente.
- Estender o exercício anterior armazenando o número de vezes que o ip acessou o servlet (dica: utilizar `hashtable`)

