## Servlets

Da necessidade de gerar conteúdo dinâmico surgiram os programas de CGI (*Common Gatway Interface*). Através de linguagens como C, C++, Perl, ASP, PHP, Cobol, Delphi, Shell etc., foi possível gerar conteúdo que permite ao usuário acesso a diversas funcionalidades através de páginas HTML (WALRATH, 2002).

Para melhorar o desempenho do CGI, o que viria a ser uma *servlet*, uma nova forma de trabalhar com requisições de clientes via web que economiza o tempo de processamento de uma chamada e a memória que seria gasta para tal processo, além de ser desenvolvida em Java e possuir todas as vantagens e facilidades de orientação a objeto (KURNIAWAN, 2002).

Um *servlet* funciona como um pequeno servidor (servidorzinho em inglês) que recebe chamadas de diversos clientes. Cada *servlet* é um objeto que recebe tais chamadas e retorna algo, como por exemplo, uma página html ou uma imagem do formato jpeg.

Diversas requisições podem ser feitas a uma mesma *servlet* ao mesmo tempo, em um servidor, por isso ela é mais rápida que um programa CGI comum.

Novos dados não precisam ser criados para que o servidor execute sua nova tarefa. A Figura 03 mostra três clientes acessando o mesmo servidor *web/contêiner* de *servlets* através do protocolo http.

A página a ser retornada pela *servlet* pode ser um jpeg, um gif, um arquivo html, arquivos de texto ou simplesmente binários.



Figura 03 – Acesso de três clientes ao mesmo servidor web/contêiner de servlets através do protocolo http

## Benefícios

Dentre os vários benefícios desta tecnologia, (KURNIAWAN, 2002) destaca:

* **Desempenho**: Cada solicitação é gerenciada pelo processo contentor de *servlet*. Depois que um *servlet* termina de processar uma solicitação, ele permanece na memória, aguardando por uma outra solicitação.
* **Portabilidade**: análogo a outras tecnologias Java, os aplicativos *servlet* são portáteis. Você pode movê-los para outros sistemas operacionais sem dificuldades.
* **Rápido ciclo de desenvolvimento**: Como uma tecnologia Java, *servlets* têm acesso à rica biblioteca Java, que ajuda a apressar o processo de desenvolvimento.
* **Robustez**: *Servlets* são gerenciados pela JVM. Como tal, você não precisa se preocupar com galha de memória ou coleta de resíduos, o que lhe ajuda a criar aplicativos robustos.
* **Aceitação difundida**: Java é uma tecnologia largamente aceita. O que significa que muitos fabricantes trabalham com tecnologias baseadas em Java, o que torna isso uma das grandes vantagens, pois pode ser encontrados com facilidade e comprados componentes, que se ajustem às necessidades do usuário, o que poupa tempo de desenvolvimento.

## Operação de um Servlet

Ao receber pela primeira vez uma requisição para uma aplicação, o servidor de aplicação carrega a classe Java, ou *servlet*, que implementa aquela aplicação, e que será responsável por atender à requisição. O *servlet* então recebe a requisição e a processa, enviando a resposta ao servidor de aplicação que a encaminha a quem solicitou. Ao término do processamento da requisição, o *servlet* permanece em memória, esperando por outras requisições.

Há duas situações que fazem com que o *servlet* seja recarregado: falta de memória identificada pelo servidor de aplicação ou a versão do *servlet* em memória não é mais a versão atual. A figura 04 representa o funcionamento de um *servlet*.

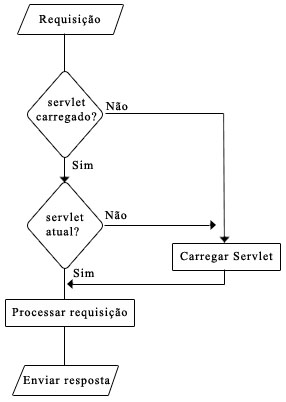


Figura 04 - Funcionamento de um Servlet

FONTE: (KURNIAWAN, 2002)

O ciclo de vida é controlado pelo contêiner no qual o *servlet* foi implantado. Todo *servlet* deve implementar a interface javax.servlet.Servlet, que contém os três métodos que controlam o ciclo de vida de um *servlet*: init(), service() e destroy(), caracterizado pelos seguintes estágios (WALRATH, 2002):

* **O método init():** o método init() é chamado pelo contêiner após o *servlet* ter sido instanciado; esse método é chamado uma vez e sua execução deve ser completada com sucesso para que o *servlet* possa receber solicitação.
* **O método service():** o método service() é chamado pelo contêiner depois do sucesso na execução do método init(); o contêiner passa como argumento um objeto ServletRequest e um objeto ServletResponse, responsáveis respectivamente por conter as requisições do cliente e as respostas do *servlet*, tornando possível uma personalização de como o *servlet* atenderá às requisições do cliente.
* **O método destroy():** o método destroy() é invocado quando o contêiner determina que um *servlet* deve ser removido do serviço, ou por falta de memória ou pelo fato de o contêiner estar sendo finalizado; esse método deve liberar os recursos utilizados pelo *servlet* e que foram inicializados no método init().

REFERÊNCIAS BIBLIOGRÁFICAS

KURNIAWAN, Budi. Java para a Web com Servlets, JSP e EJB. Rio de janeiro: Ciência Moderna, 2002.

WALRATH, Kathy. Tutorial do J2EE. São Paulo: Editora Campus, 2002.