Java Servlets

A tecnologia Java Servlets utiliza a plataforma Java para criar programas, classes que estedem uma específica super classe, que rodam no servidor de aplicações web, recebendo requisições via GET e POST, do cliente na camada de apresentação, e podendo gerar respostas em HTML e/ou XML. Podem ainda acessar outros recursos como banco de dados ou servir uma página web (HTML, JSP, PHP, entre outros formatos) como resposta.

Servlets só serão executados se hospedados em específicos servidores de aplicações web que suportam servlets. Dentre esses servidores estão o Apache Tomcat, IBM WebSphere Application Server, Oracle WebLogic, JBOSS, GlassFish, entre outros menos conhecidos.

Exemplo de um servlet:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

   public void doGet(HttpServletRequest request, HttpServletResponse response) throws
          ServletException, IOException {
                   PrintWriter saida = response.getWriter();
                   saida.println("Hello World!");
   }
}
```

Ciclo de vida de um servlet

Ao ser invocado pela primeira vez depois que o servidor web entrou no ar, o servlet é carregado no "contenedor de servlet" – *servlet container*. Ao ser carregado no container, o servlet executa automaticamente o método **`init()`** uma única vez e em seguida executa o método **`service()`**. A partir daí, as próximas requisições ao servlet, será executado apenas o método **`service()`**. O método **`service()`** é quem chama os métodos doGet() ou doPost() ou os outros doXXX(), dependendo do tipo de requisição enviada ao servidor web.

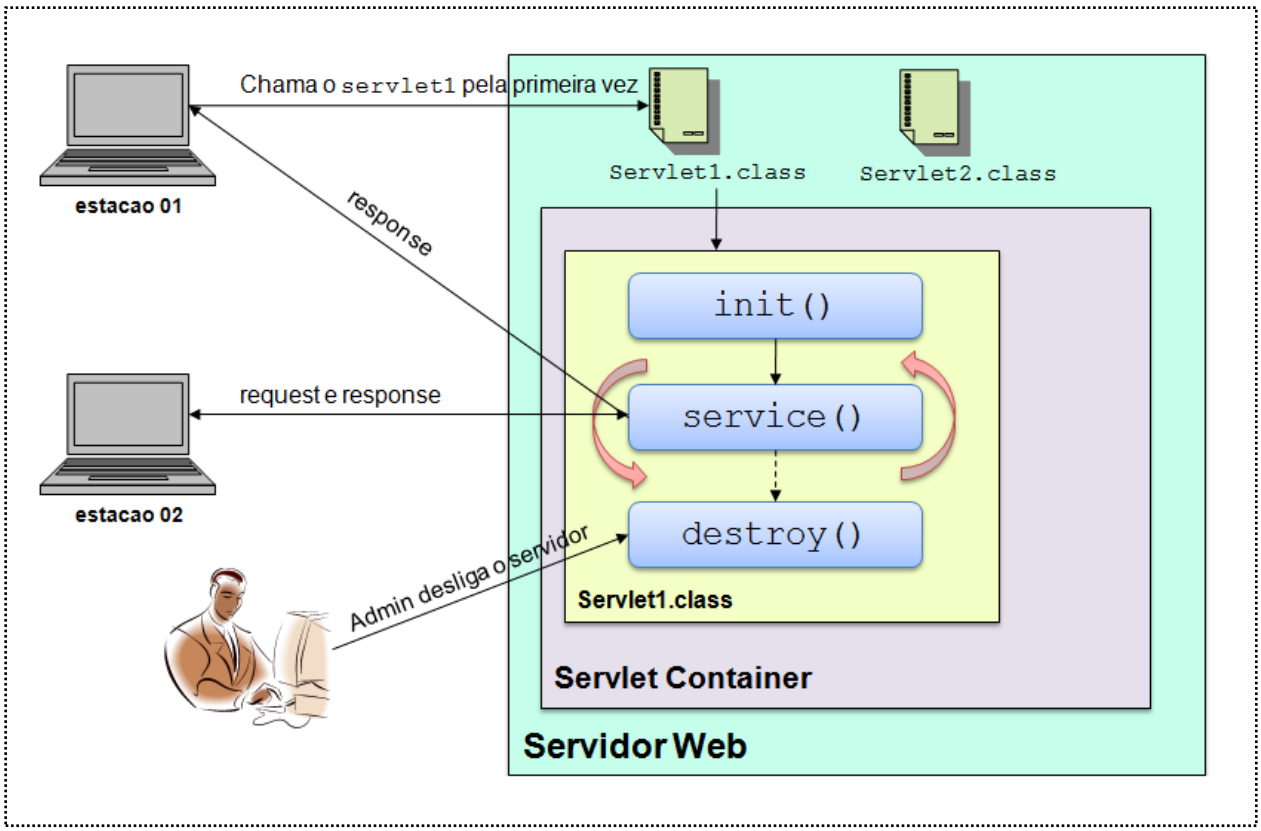

Figura 1: Ciclo de vida de um servlet

Mapeando uma URL para o servlet através de Anotações (*Annotations*)

Para que um servlet possa ser invocado através de uma requisição web, é necessário configurar um mapeamento de URL para o servlet. Para isso utilizaremos a classe de *annotations* chamada `javax.servlet.annotation.WebServlet`, a qual necessita como parâmetro a descrição da URL que acessará o servlet.

```
@WebServlet("/ola")
public class HelloWorld extends HttpServlet {

	protected void doGet(HttpServletRequest request, HttpServletResponse response)
                                                throws ServletException, IOException {

		PrintWriter saida = response.getWriter();
		saida.println("Hello World!");

	}
}
```

Nesse caso, apesar da classe de servlet se chamar `HelloWorld`, esse servlet será invocado na URL com o nome de `ola`.

Veja com ficaria a URL completa da invocação desse servlet:

`http://127.0.0.1:8080/nomedoprojeto/ola`

Você pode mapear mais de um nome para o mesmo servlet, para isso você deve ajustar o parâmetro do `@WebServlet` como mostra o exemplo abaixo:

```
@WebServlet({"/HelloWorld","/ola","/hello"})
```

Nesse caso, qualquer uma das partes de URL, `HelloWorld`, `ola` e `hello`, irá invocar o servlet HelloWorld.

Instalando o servlet-api.jar

Todo servlet estende a classe `javax.servlet.http.HttpServlet` e utiliza outras classe necessárias para que o servlet possa de fato ser chamado de servlet.

Essas classes essenciais para os servlets se encontram compactadas em um arquivo chamado **`servlet-api.jar`** que se encontra no `%caminho_de_instalação%\xampp\tomcat\lib`.

Copie o arquivo servlet-api.jar para a pasta `WebContent\WEB-INF\lib` do seu projeto no Eclipse.