
@Description：Report of Servlet Basic, Part 1

@Date：2018.5.24

@Auther：Sujin Guo

@Reference: [1] https://blog.csdn.net/qq_26676207/article/details/51023327

[2] https://www.w3cschool.cn/servlet/servlet-intro.html

[3]《Servlet & Jsp &Spring MVC 初学指南》

Servlet

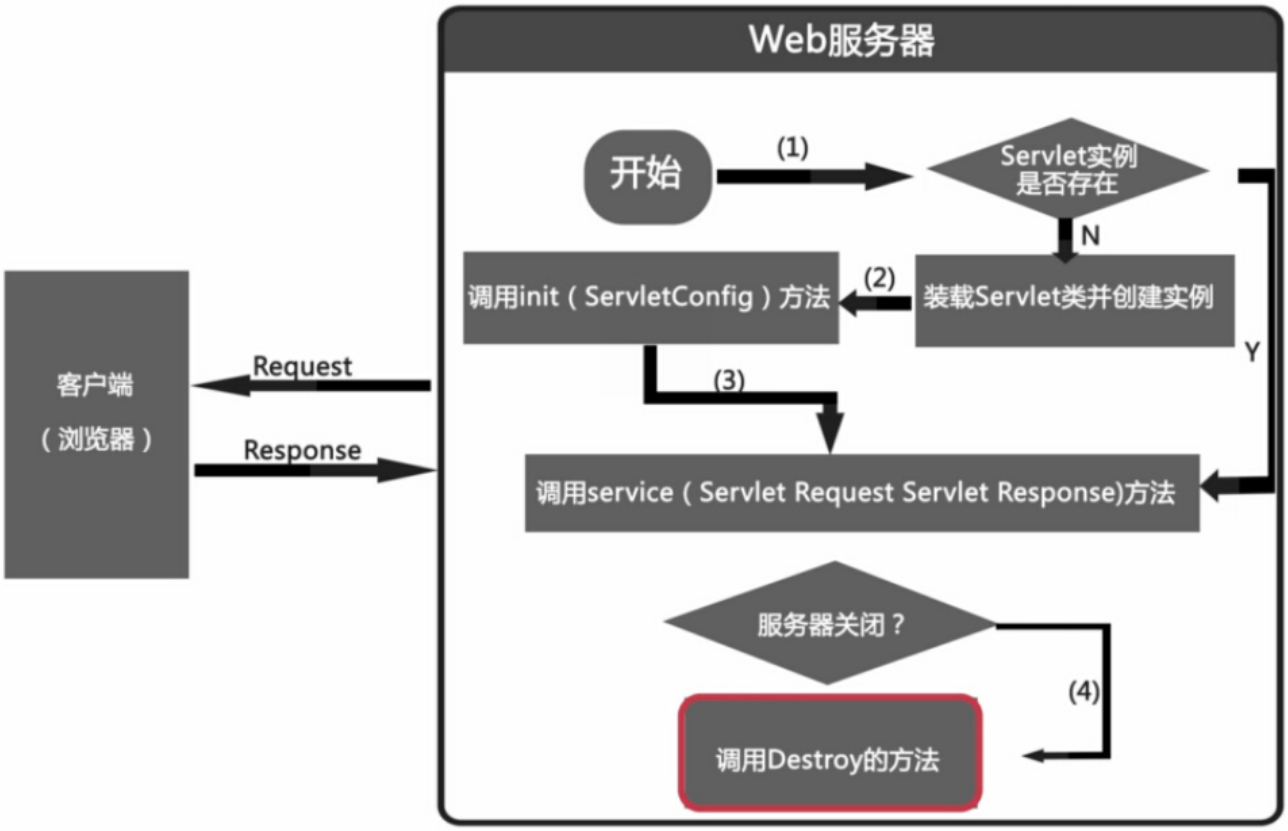
一、Servlet 原理

Servlet 接口定义了一套 Java 处理网络请求的规范，即如何处理 HTTP 请求，并返回结果。

Servlet 不能直接运行，要放在web容器中才能使用。与客户端打交道的是web容器（如tomcat），容器接收Http请求报文，将请求内容转化为 HttpRequest 的Java类，传入Servlet 进行逻辑操作，再将封装Http响应内容的HttpResponse 转换为 Http 响应报文返回给客户端。

1、Servlet 生命周期

- Servlet 通过调用 **init ()** 方法进行初始化。初始化时，会创建ServletConfig对象，传入Servlet的成员变量。
- Servlet 调用 **service()** 方法来处理客户端的请求。service() 方法检查 HTTP 请求类型（GET、POST、PUT、DELETE 等），并调用 doGet、doPost、doPut，doDelete 等方法。
- Servlet 通过调用 **destroy()** 方法终止（结束）。



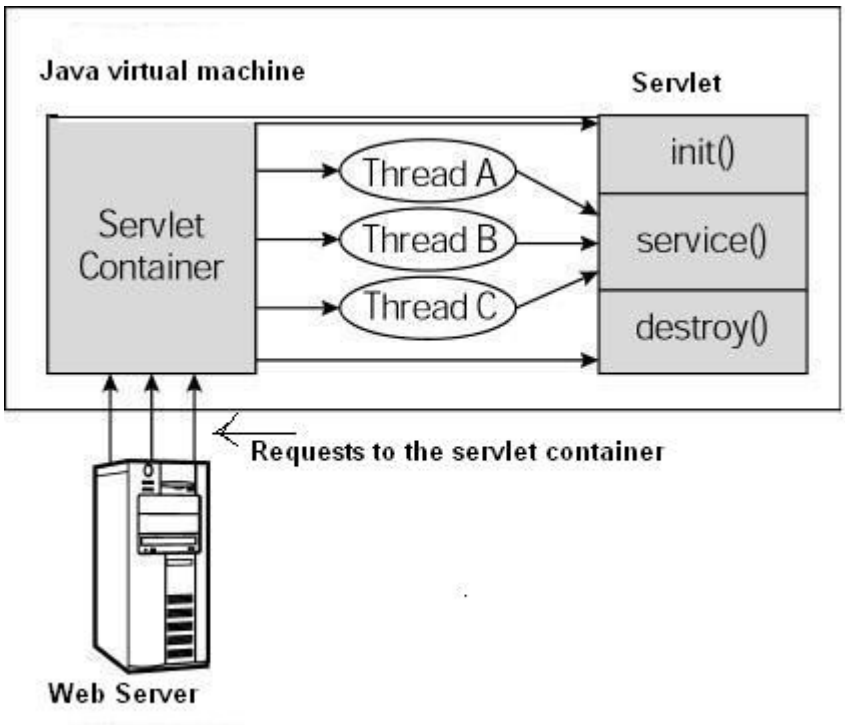
2、Servlet 单实例多线程

初始化：

- 每个Servlet 只会实例化一次
- 容器启动时实例化：在 web.xml 中通过标签进行设定，参数为数字，表示了启动的顺序，数字越小优先级越高。
- 第一次访问时实例化：默认不配置该参数的情况下，Servlet 只有在被访问时才会实例化。

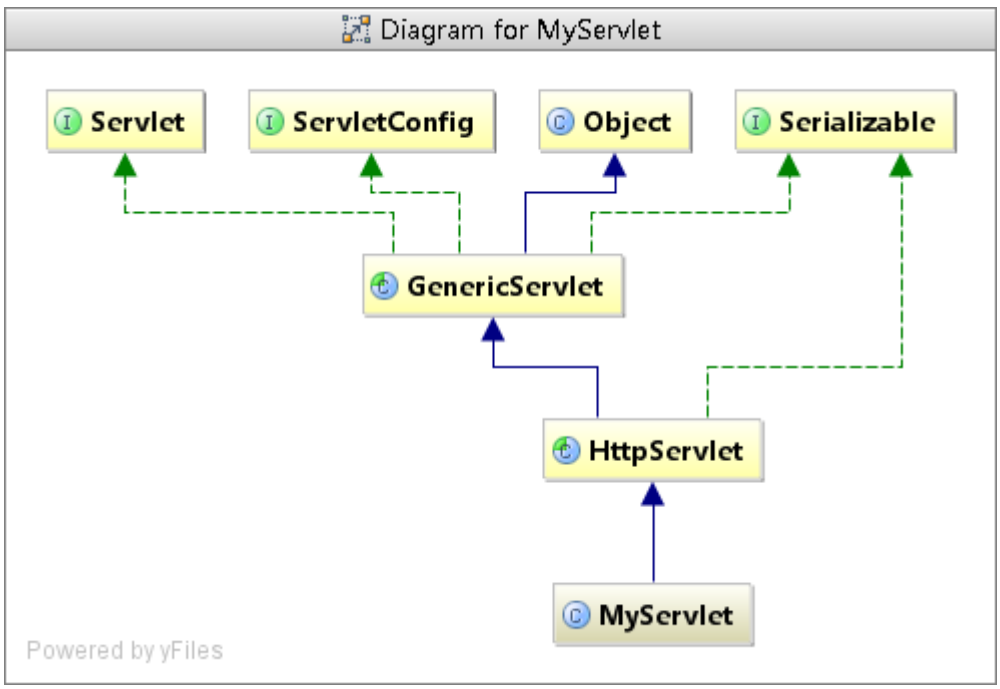
多线程：

- 每次服务器接收到一个 Servlet 请求时，服务器会产生一个新的线程并调用服务。



3、Servlet 继承关系

- **ServletConfig** Servlet的配置信息，常用来在Servlet初始化时进行信息传递
- **Servlet** 定义Servlet生命周期调用的方法：init()、service()、destroy()
- **GenericServlet** 与协议无关的Servlet，实现了Servlet和ServletConfig接口
- **HttpServlet** 基于HTTP协议的实现类



二、Servlet 使用

1、编写 Servlet 流程

编写一个Servlet程序大体上需要3个步骤：继承HttpServlet-->重写doGet（）或者doPost（）方法-->在web.xml中注册Servlet。

新语法中，可以用 @WebServlet(name = "MyServlet", urlPatterns = { "/my" }) 这样的注解来注册servlet。

2、Servlet 自定义初始化

Servlet 的自定义初始化可以通过配置 web.xml 文件，或者重写Servlet的init()方法实现。

3、常用类

（1）ServletConfig

通过Servlet 的 getServletContext 方法获得。

当Servlet容器初始化Servlet时，Servlet容器会给Servlet的init方法传入一个ServletConfig。ServletConfig中存放着Servlet的初始化信息，可通过@WebServlet或web.xml配置。

```
//示例
@WebServlet(name = "ServletConfigDemoServlet",
  urlPatterns = { "/servletConfigDemo" },
  initParams = {
    @WebInitParam(name="admin", value="Harry Taciak"),
    @WebInitParam(name="email", value="admin@example.com")
  }
)
```

方法名	说明
getInitParameter()	根据参数名获取指定初始化参数
getInitParameterNames()	获取所有初始化参数名

（2）ServletContext

通过 ServletConfig 的 getServletContext 方法获得

ServletContext 表示 Servlet 应用程序。每个Web应用程序只有一个上下文。在将一个应用程序同时部署到多个容器的分布式环境中，每台Java虚拟机上的Web应用都会有一个ServletContext对象。

通过在ServletConfig中调用getServletContext方法，可以获得ServletContext。ServletContext可以共享从应用程序中访问到的信息，并且可以动态注册Web对象。

```
java.lang.Object getAttribute(java.lang.String name)
java.util.Enumeration<java.lang.String> getAttributeNames()
void setAttribute(String name, Object object)
void removeAttribute(String name)
```

三、Servlet 程序示例

- 第一个 Servlet（编写 Servlet 流程）
- Servlet 生命周期（展示 init、destroy、service 函数的作用）

- 获取 Servlet 初始化参数（配置Servlet初始化参数并获取）

Request & Response

一、Request & Response API

1、ServletRequest

对于每一个HTTP请求，Servlet容器都会创建一个 ServletRequest 实例，并将它传给Servlet的Service方法。ServletRequest 封装了关于这个请求的信息。

getParameter() 是最常用的方法，可以获取表单(form)的参数值。

方法名	说明
getContentLength()	返回请求主体的字节数。如果不知道字节长度，这个方法就会返回−1。
getContentType()	返回请求主体的MIME类型，如果不知道类型，则返回null
getParameter(String name)	返回指定请求参数的值。
getProtocol()	返回这个HTTP请求的协议名称和版本。

2、HttpServletRequest

HttpServletRequest 继承了 ServletRequest

方法名	说明
getContextPath()	返回表示请求上下文的请求URI部分
getCookies()	返回一个Cookie对象数组
getMethod()	返回生成这个请求的HTTP方法名称。
getQueryString()	返回请求URL中的查询字符串
getSession()	返回与这个请求相关的会话对象。如果没有，将创建一个新的会话对象。
getSession(boolean create)	返回与这个请求相关的会话对象。如果有，并且create参数为True，将创建一个新的会话对象。

3、ServletResponse

ServletResponse接口表示一个Servlet响应。在调用Servlet的Service方法前，Servlet容器首先创建一个ServletResponse，并将它作为第二个参数传给Service方法。ServletResponse隐藏了向浏览器发送响应 的复杂过程。

方法名	说明
getWriter	获取字符输出流
setContentType	设置方法体内容格式，如：text/html
getOutputStream	获取二进制输出流

4、HttpServletResponse

HttpServletResponse 继承了 ServletResponse

方法名	说明
addCookie(Cookie cookie)	给这个响应对象添加一个cookie
addHeader(String name, String value)	给这个响应对象添加一个header
sendRedirect(String location)	发送一条响应码，将浏览器跳转到指定的位置

二、Requset & Response 的使用

1、转发和重定向

网页跳转有两种方式重定向和服务器内部转发。Forward 和 Redirect 的区别。

Forward:

```
request.getRequestDispatcher("/somePage.jsp").forward(request, response);
```

1. servlet将请求交给Web应用的另一部分RequestDispatcher.forward(request,response) 方法的调它们属于同一个访问请求和响应过程
2. 浏览器以正常方式得到响应，把它显示给用户（在浏览器的地址栏上没有变化，用户不知道发生过请求分派）

Redirect:

```
response.sendRedirect(String location);
```

1. servlet在响应上请求sendRedirect(aString)
2. HTTP响应有一个状态码“301”，和一个“Location”首部，这个首部值是一个URL
3. 浏览器得到响应，发现“301”状态码，并寻找“Location”首部，由此建立一个新的请求（在浏览器的地址栏里URL改变了）

区别：

1. 重定向跳转过程发生在客户端，客户端发送了两次请求。转发跳转过程发生在服务端，客户端不能察觉到转发过程。
2. 由于重定向要经过两次请求来回，所以转发速度更快，并且可以继续使用 request、response 对象。

3. 重定向可以将页面转到外网。

2、相对路径和绝对路径

URL的写法也有2种方式：绝对路径和相对路径。相对路径使用..即可，而绝对路径重定向需要依赖 request.getContextPath() 方法取得上下文环境，而服务器内部转发中的斜线就表示项目的根目录。

3、获取请求信息

获取 http 请求头信息

获取 http 表单信息：request.getParameter(String name)

4、设置响应信息

设置和获取 http 响应头信息：response.addHeader(String name, String value)

设置响应内容：response.getWriter()

三、Requset & Response 程序示例

- Servlet 表单提交与展示（获取请求信息、设置相应信息、转发 Forward）
- Servlet 路径跳转（绝对路径和相对路径）