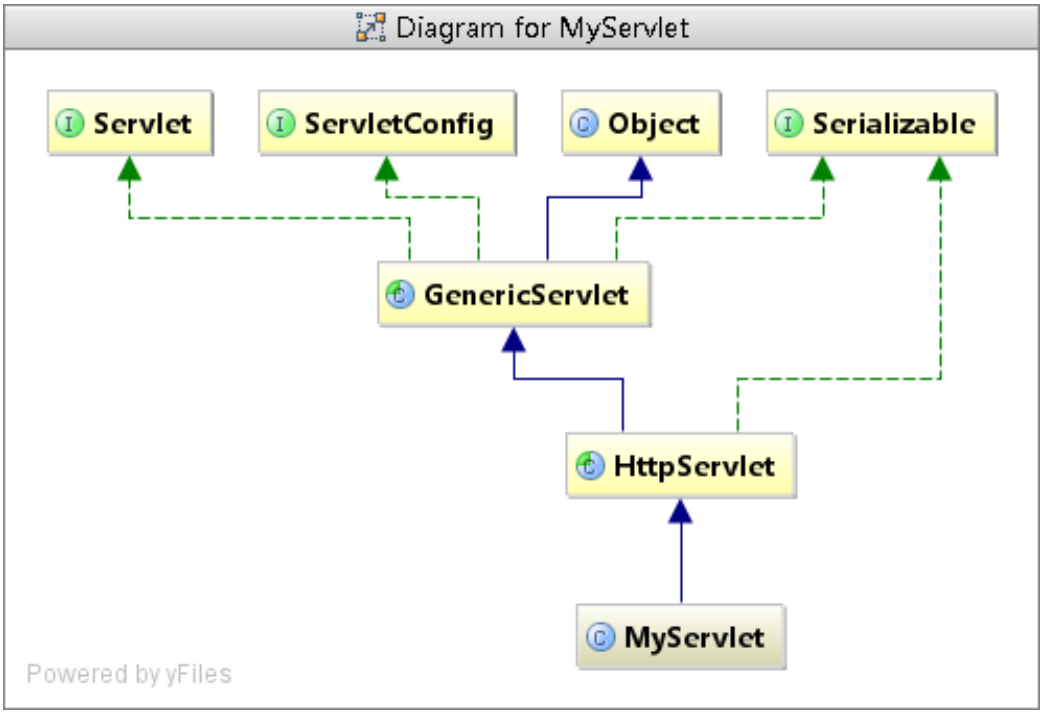
[03] Servlet 继承关系和生命周期

1、Servlet 的继承关系

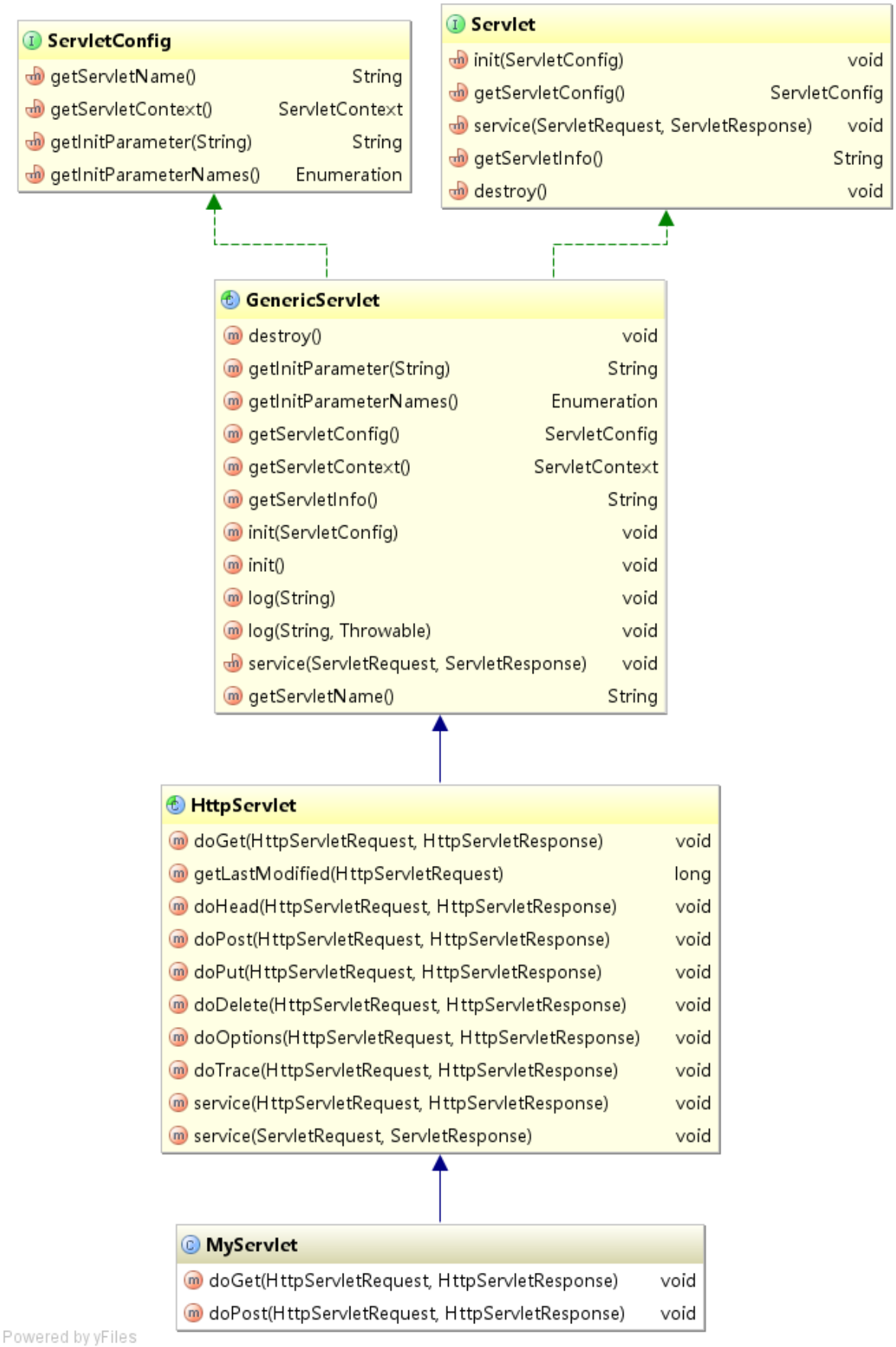
假如现有我们自定义的一个 Servlet，继承 HttpServlet，那么实际上它的继承链如下图：



可以看到，核心的部分在于：

- 两个顶级接口
  - Servlet
  - ServletConfig
- 接口的实现类
  - GenericServlet
- 基于 HTTP 协议的实现类
  - HttpServlet

我们剥离一下，把不需要看的去掉，再把方法显示一下，可清晰知道 Servlet 的整体继承关系如下：



其中重点摘录部分进行说明：

- **ServletConfig** Servlet 的配置信息，常用来在 Servlet 初始化时进行信息传递
  - getServletContext() 获取 Servlet 运行的上下文环境对象，可以获取对应信息（如 Servlet 路径），存取容量级的变量
  - getInitParameter(String name) 获取初始化参数（web.xml 中配置的 init-param）

- **GenericServlet** 一般的 Servlet，实现了 Servlet 和 ServletConfig 接口
  - init(ServletConfig config) 初始化方法，方法中调用了 init()
  - init() 初始化方法，方法体为空，主要用于自定义 Servlet 的覆盖
  - service(ServletRequest request, ServletResponse response) 抽象方法 service，要求继承类实现
  - destory() Servlet 销毁前要执行的方法

- **HttpServlet** 基于 HTTP 协议的实现类
  - service(ServletRequest request, ServletResponse response) 实现了 GenericServlet 的抽象方法，调用了 service(HttpServletRequest, HttpServletResponse)
  - service(HttpServletRequest request, HttpServletResponse response) 根据请求的不同调用了 doGet 或 doPost 方法
  - doGet() 处理 GET 方式的请求
  - doPost() 处理 POST 方式的请求

其中稍微提一下，在 GenericServlet 中有个 init(ServletConfig config)方法，调用了 init()方法，但是 init()方法体却为空，为什么？

```
public void init(ServletConfig config) throws ServletException {
    this.config = config;
    this.init();
}
public void init() throws ServletException {

}
```

首先，为了方便能够在其他地方也能直接使用 ServletConfig 对象，而不仅仅局限在 init(ServletConfig config)方法中，所以创建一个私有的成员变量 config，在 init(ServletConfig config)方法中就将其赋值给 config，这样一来，GenericServlet 和其子类都可以调用其 getServletConfig()方法来获取 ServletConfig 对象了。

之所以有空的 init()，实际上就是为了后续的扩展和重写，有需要的情况下去覆盖 init()而不是去覆盖 init(ServletConfig config)，因为后者一旦覆盖，就无法通过上述的方法在其他地方便捷地调用 getServletConfig 方法获取 ServletConfig 对象了。

实际上，Servlet 继承关系文字描述起来反而复杂，结合继承图和源码查看，能更加清晰明了，这里进行了大概的阐述，更多细节的话需要查看源码或者 API 了，此处不再详细展开。

2、Servlet 的生命周期

Servlet 之间的启动是有先后顺序的，这可以在 web.xml 中通过<load-on-startup>标签进行设定，参数为数字，表示了启动的顺序。启动顺序的默认值是 0：

- load-on-startup --> 0：Servlet 被访问时才进行实例化
- load-on-startup --> other：在容器启动时进行 Servlet 实例化

即是说，默认不配置该参数的情况下，Servlet 只有再被访问时才会实例化；配置了参数以后，根据参数按顺序在容器启动时就将 Servlet 实例化。

然后，Servlet 的生命周期，其实说来也简单：

- 容器在加载 Servlet 的时候，会执行其 init()方法
- 当接收请求的时候，会调用 service(ServletReqeust request, ServletResponse response)方法，继而调用 doGet 或 doPost 方法
- 在服务器关闭之前，会调用 Servlet 的 destory()方法

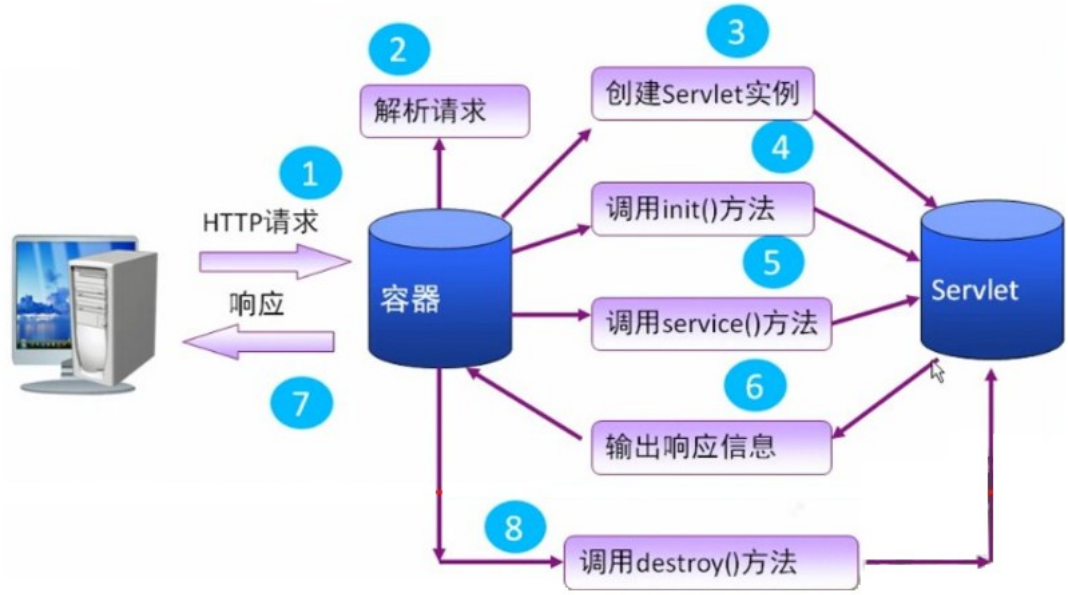
即：

- init()
- doGet() / doPost()
- destory()

实际上，完整一点来说：

- 初始化
  - 构造方法
  - init(ServletConfig config)
  - init()
- 提供服务
  - service(ServletRequest request, ServletResponse response)
  - doGet() / doPost()
- 销毁
  - destory()

（写得有点啰嗦..



最后简单总结下要点：

- Servlet 只初始化一次，它是单例的，只有一个实例，通过多线程访问。即 Servlet 是多线程单实例的
- 实例化过程中，先调用构造方法，再调用 init 方法，所以初始化操作可以覆盖写到 init 方法中
- 请求方式不同会调用 doGet()或 doPost()方法
- 根据实际情况在 Servlet 销毁前调用其 destroy()方法