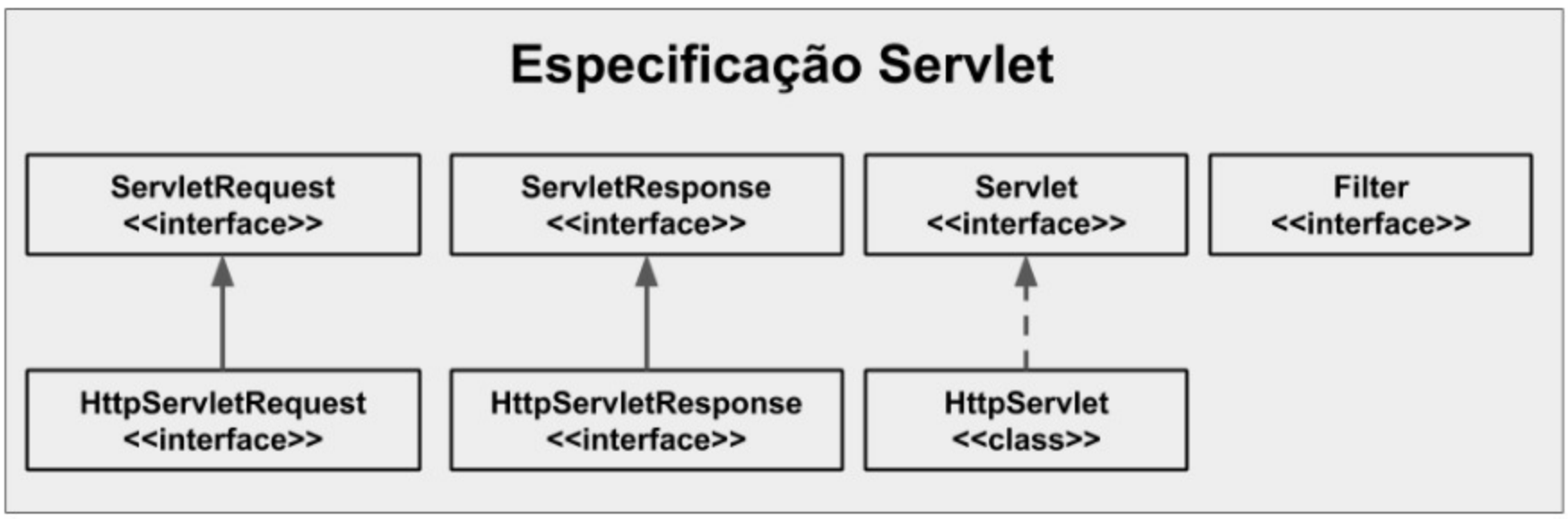
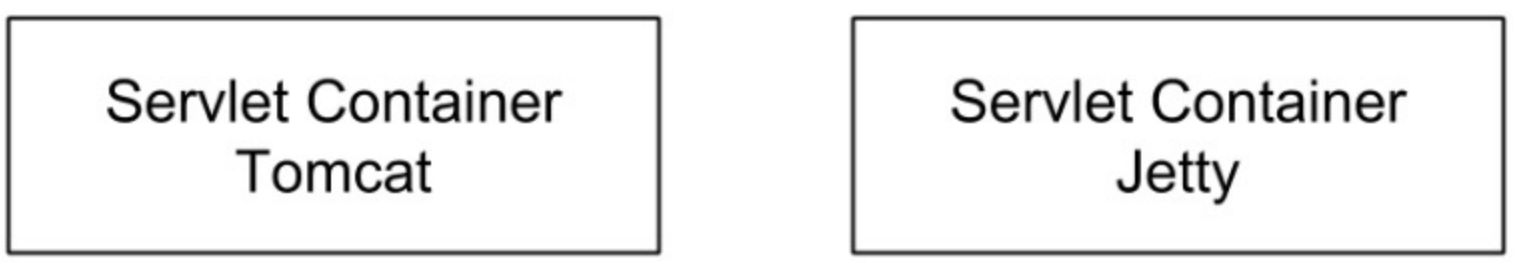
Falamos sobre a especificação Servlet, e assim a garantia que podemos rodar o nosso projeto em diferentes *Servlet Containers*. Apresentei a imagem abaixo, que mostra alguns tipos importantes da especificação, como `javax.servlet.Servlet`, `javax.servlet.http.HttpServlet`, `javax.servlet.ServletRequest`, `javax.servlet.http.HttpServletRequest`, entre outros:





Repare que existem tipos mais genéricos e tipos mais específicos, focados no protocolo HTTP, por exemplo:

- `javax.servlet.Servlet` --> `javax.servlet.http.HttpServlet`
- `javax.servlet.ServletRequest` --> `javax.servlet.http.HttpServletRequest`
- `javax.servlet.ServletResponse` --> `javax.servlet.http.HttpServletResponse`

Tirando a interface `Filter`, sempre existe um tipo mais especifico do mundo HTTP. Por quê?

A ideia inicial era que o mundo Servlets suportasse outros protocolos como FTP ou SMTP. Ou seja, as servlets e os servlet containers poderiam trabalhar com outros protocolos além do HTTP. Por isso existem essas interfaces genéricas (sem `Http` no nome), para estender e atender novos protocolos.

Por exemplo, poderia existir um *FTP Servlet Container*, que atenderia o protocolo FTP e assim estender os tipos genéricos, para criar um `FtpServlet`, ou `FtpServletRequest`. No final, não existem essas implementações, e o protocolo HTTP é o único que as servlets atendem. Isso também se dá por causa da onipresença do protocolo HTTP no dia a dia, e da baixa relevância dos outros protocolos.