使用示例： java -jar selenium-server.jar [-interactive] [options]

* **-port <nnnn>:** selenium服务器使用的端口号（默认 4444）
* **-timeout <nnnn>:** 我们放弃前（超时）所等待的秒数
* **-interactive:** 进入交互模式。参考[教程](http://wiki.javascud.org/pages/viewpage.action?pageId=5213)获取更多信息
* **-multiWindow:** 进入被测试网站都在单独窗口打开的模式，并且selenium支持frame
* **-forcedBrowserMode <browser>:** 设置浏览器模式（例如，所有的会话都使用"\*iexplore"，不管给getNewBrowserSession传递什么参数）
* **-userExtensions <file>:** 指定一个被载入到selenium的JavaScript文件
* **-browserSessionReuse:** 停止在测试间重新初始化和替换浏览器。
* **-alwaysProxy:** 默认情况下，我们尽量少的进行代理；设置这个标志将会强制所有的浏览器通讯都通过代理
* **-firefoxProfileTemplate <dir>:** 一般情况，我们在每次启动之前都生成一个干净的Firefox设置。您可以指定一个目录来让我们将您的设置拷贝过来，代替我们生成的。
* **-debug:** 进入debug模式，会有更多的跟踪调试信息
* **-htmlSuite <browser> <startURL> <suiteFile> <resultFile>:** 使用指定的浏览器（例如"\*firefox"）在指定的URL（例如"http://www.google.com"），运行一个单独的HTML Selenese (Selenium Core)测试套件然后立即退出。您需要指定HTML测试套件的绝对路径还有我们将会生成的HTML测试结果文件的路径。
* **-proxyInjectionMode:** 进入代理注入模式，这个模式中selenium服务器作为进入测试程序的所有内容的代理服务器。在这个模式下，可以跨多个域访问，并且还支持如下附加参数：
  + **-dontInjectRegex <regex>:** 附加的正则表达式，代理注入模式能够使用它决定是否进行注入
  + **-userJsInjection <file>:** 指定一个JavaScript文件，将它注入到所有页面中
  + **-userContentTransformation <regex> <replacement>:** 一个正则表达式，对所有被测HTML内容进行匹配；第二个string将会对替换所有匹配的内容。这个标志能够使用多次。一个简单的适合使用这个参数的例子：如果你添加"-userContentTransformation https http"那么测试应用程序的HTML中的所有"https"字符串都会被替换为"http"。

我们还支持两种Java系统属性：**-Dhttp.proxyHost 和 -Dhttp.proxyPort**。使用Selenium服务器作为代理服务器，Selenium RC一般重载你的代理服务器配置。使用这个参数适合在使用Selenium服务器代理的同时使用你自己的代理服务器。使用代理服务器时这样配置：

java -Dhttp.proxyHost=myproxy.com -Dhttp.proxyPort=1234 -jar selenium-server.jar

如果你的HTTP代理服务器需要验证，你还可以在http.proxyHost和http.proxyPort后面设置-**Dhttp.proxyUser** 和 -**Dhttp.proxyPassword**。

java -Dhttp.proxyHost=myproxy.com -Dhttp.proxyPort=1234 -Dhttp.proxyUser=joe -Dhttp.proxyPassword=example -jar selenium-server.jar