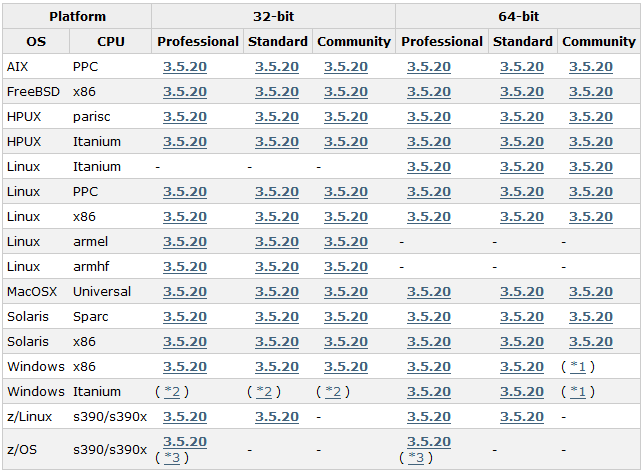
[**Java Service Wrapper简介与使用**](http://www.cnblogs.com/zcy_soft/p/3738947.html)

   在实际开发过程中很多模块需要独立运行，他们并不会以web形式发布，传统的做法是将其压缩为jar包独立运行，这种形式简单易行也比较利于维护，但是 一旦服务器重启或出现异常时，程序往往无法自行修复或重启。解决服务器重启的传统做法是编写一段shell脚本随服务器启动而运行，但是这样做只是治标， 那么我们想寻求一种“治本”的方式该怎么办呢？  
        Java Service Wrapper就轻松而简单的为我们解决了这些问题。"Java Service Wrapper"顾名思义，将我们的Java程序包装成系统服务，这样就可以随着系统的运行而自动运行，当然Java Service Wrapper(下面简称Wrapper)的功能绝不仅于此。

        Wrapper下载地址：<http://wrapper.tanukisoftware.com/doc/english/download.jsp>



        通过下载页面我们可以看到Wrapper几乎支持所有的系统环境，说明Wrapper在这方面还是很下工夫的，目前最新版本为3.5.20，我们选择Linux x86版本下载，解压后目录组成如下图所示：



        为了更直观的了解Wrapper的目录及文件结构，可以通过"tree"命令列出Wrapper的所有文件树，cmd控制台下输入命令：

1. tree /f
3. 显示目录结构如下：
4. wrapper-linux-x86.
5. │  jdoc.tar.gz             //javadoc文件
6. │  README\_de.txt           //说明
7. │  README\_en.txt           //说明
8. │  README\_es.txt           //说明
9. │  README\_ja.txt           //说明
10. │
11. ├─bin                      //执行文件目录
12. │      demoapp             //示例程序
13. │      testwrapper         //测试程序
14. │      ★wrapper           //主程序(重要)
15. │
16. ├─conf                     //配置文件目录
17. │      demoapp.conf        //示例配置文件
18. │      ★wrapper.conf      //主配置文件(重要，文件名可修改)
19. │
20. ├─doc                      //说明文档目录
21. │      index.html          //首页
22. │      revisions.txt       //版本说明
23. │      wrapper-community-license-1.1.txt  //许可协议
24. │
25. ├─jdoc                     //javadoc文档目录
26. │      index.html          //首页
27. │
28. ├─lib                      //依赖类库目录
29. │      ★libwrapper.so     //wrapper linux文件(.so：用户层的动态库)
30. │      ★wrapper.jar       //wrapper主程序(重要)
31. │      wrapperdemo.jar     //示例程序
32. │      wrappertest.jar     //测试程序
33. │
34. ├─logs                     //日志目录
35. │      wrapper.log         //日志文件
36. │
37. └─src                      //源代码目录
38. ├─bin                  //执行程序目录
39. │      ★sh.script.in  //shell脚本源代码(重要)
40. └─conf                 //配置目录
41. wrapper.conf.in    //原始配置

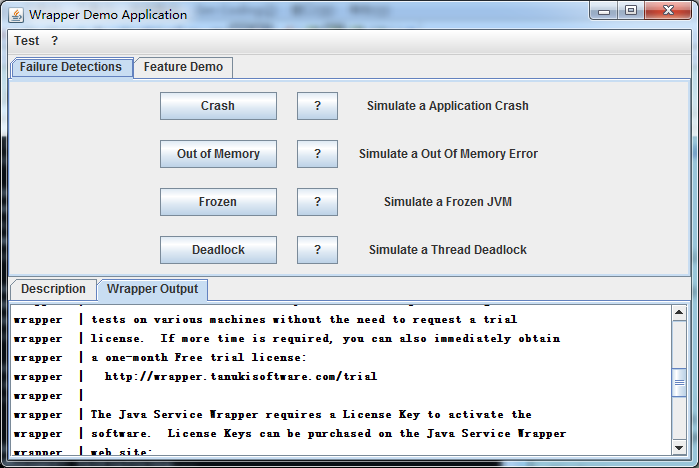
        以下是官方给出的一些Wrapper的优点：

        (1) 使用我们的产品无须在你的程序中添加任何额外的代码。  
        (2) 当你的程序或JVM出现问题时会自动响应事先定制的策略。  
        (3) 当出现问题时会及时进行通知。  
        (4) 完善的日志记录功能可以更好为您提供支持。  
        (5) 在不同的系统上你可以指定一个标准的流程相同流程，也就是说相同的程序可以不必修改即运行于不同系统。  
        (6) 可以将你的应用安装成windows或unix的服务或守护进程。

        看到Wrapper有这么多好处，那么我们就通过Wrapper自带的示例程序来进一步了解Wrapper吧：  
        1.创建服务工作目录，以操作系统为Linux，目录结构为usr/local/wrapper为例，按照上面的目录结构，在其下创建"bin","conf","lib","logs"这四个相同名称的文件夹。  
        2.将配置及程序文件复制至相应目录（也就是上面标★的文件）；  
        (1)bin 目录下的wrapper 文件复制到usr/local/wrapper/bin下。  
        (2)src\bin 目录下的sh.script.in 文件复制到usr/local/wrapper/bin下，并将.in后缀名删除并修改名称，修改后为javaService.script。  
        (3)conf 目录下的wrapper.conf 文件复制到usr/local/wrapper/conf下。  
        (4)lib 目录下的wrapper.jar 和libwrapper.so 文件复制到usr/local/wrapper/lib下。  
        注：以上是正式环境所需文件的配置方式，这里我们需要运行Wrapper自带的demo程序，所以需要将demoapp,demoapp.conf,wrapperdemo.jar 这三个文件复制到相应目录。  
        3.进入bin目录执行以下命令：

1. ./demoapp start

        接下来会显示很多提示，最终显示如下页面：



        出现此页面证明你的程序已经运行成功了，恭喜！  
        如果启动失败，我们可以查看logs日志内容，如下：

1. STATUS | wrapper  | 2013/07/30 11:22:47 | --> Wrapper Started as Daemon
2. STATUS | wrapper  | 2013/07/30 11:22:47 | Java Service Wrapper Community Edition 64-bit 3.5.20
3. STATUS | wrapper  | 2013/07/30 11:22:47 |   Copyright (C) 1999-2013 Tanuki Software, Ltd. All Rights Reserved.
4. STATUS | wrapper  | 2013/07/30 11:22:47 |     http://wrapper.tanukisoftware.com
5. STATUS | wrapper  | 2013/07/30 11:22:47 |
6. STATUS | wrapper  | 2013/07/30 11:22:47 | Launching a JVM...
7. INFO   | jvm 1    | 2013/07/30 11:22:47 | WrapperManager: Initializing...
8. INFO   | jvm 1    | 2013/07/30 11:22:47 | DemoApp: Initializing...
9. INFO   | jvm 1    | 2013/07/30 11:22:47 | Demo: start()
10. INFO   | jvm 1    | 2013/07/30 11:22:47 | Demo: Showing dialog...
11. INFO   | jvm 1    | 2013/07/30 11:22:47 | Demo:
12. INFO   | jvm 1    | 2013/07/30 11:22:47 | Demo: ERROR - Unable to display the GUI:
13. INFO   | jvm 1    | 2013/07/30 11:22:47 | Demo:           java.awt.HeadlessException:
14. INFO   | jvm 1    | 2013/07/30 11:22:47 | No X11 DISPLAY variable was set, but this program performed an operation which requires it.
15. INFO   | jvm 1    | 2013/07/30 11:22:47 | Demo:
16. INFO   | jvm 1    | 2013/07/30 11:22:47 | Demo: This demo requires a display to show its GUI.  Exiting...
17. INFO   | jvm 1    | 2013/07/30 11:22:48 | Demo: stop(0)
18. STATUS | wrapper  | 2013/07/30 11:22:49 | <-- Wrapper Stopped

        从日志内容可以查看程序及服务的运行状态，Wrapper日志采用此种格式：类型 | 拥有者 | 时间 | 具体内容

        日志内容显示我们的Linux系统没有安装图形界面或者根本没有显卡。

        注：这里需要说明一下，Wrapper运行首先需要Java运行环境支持，所以在使用Wrapper前请先确认已安装好了Java

        下面我们来尝试一下无参数调用服务的方式，如：

1. ./testwrapper
2. ./demoapp

        两者的提示相同，都为：

1. Usage: ./程序 名 [ console {JavaAppArgs} | start {JavaAppArgs} | stop | restart {JavaAppArgs} | condrestart {JavaAppArgs} | status | install | remove | dump ]
3. Commands:
4. console      Launch in the current console.
5. start        Start in the background as a daemon process.
6. stop         Stop if running as a daemon or in another console.
7. restart      Stop if running and then start.
8. condrestart  Restart only if already running.
9. status       Query the current status.
10. install      Install to start automatically when system boots.
11. remove       Uninstall.
12. dump         Request a Java thread dump if running.
14. JavaAppArgs: Zero or more arguments which will be passed to the Java application.

        原来Wrapper提供了很多种参数的选择，如：start为启动，stop为停止。下面为参数的详细解释：

1. Commands:
2. console      启动并显示控制台信息
3. start        作为一个守护进程后台启动
4. stop         停止程序
5. restart      重启程序
6. condrestart  重启已经运行的程序，与前者区别是程序必须已经在运行
7. status       查看该程序状态
8. **install      将程序安装为自启动服务，即随系统启动而启动**
9. remove       卸载自启动服务
10. dump         报告运行时的Java thread dump(thread dump百度百科：http://baike.baidu.com/view/5111187.htm)

        我们还发现单独运行wrapper命令时的提示内容与前面两者不同，如下所示：

1. Java Service Wrapper Community Edition 64-bit 3.5.20
2. Copyright (C) 1999-2013 Tanuki Software, Ltd. All Rights Reserved.
3. http://wrapper.tanukisoftware.com
5. Usage:
6. ./wrapper <command> <configuration file> [configuration properties] [...]
7. ./wrapper <configuration file> [configuration properties] [...]
8. (<command> implicitly '-c')
9. ./wrapper <command>
10. (<configuration file> implicitly 'wrapper.conf')
11. ./wrapper
12. (<command> implicitly '-c' and <configuration file> 'wrapper.conf')
14. where <command> can be one of:
15. -c  --console run as a Console application
16. -v  --version print the wrapper's version information.
17. -?  --help    print this help message
18. -- <args>     mark the end of Wrapper arguments.  All arguments after the
19. '--' will be passed through unmodified to the java application.
21. <configuration file> is the wrapper.conf to use.  Name must be absolute or relative
22. to the location of ./wrapper
24. [configuration properties] are configuration name-value pairs which override values
25. in wrapper.conf.  For example:
26. wrapper.debug=true
28. Please note that any file references must be absolute or relative to the location
29. of the Wrapper executable.

        因为wrapper是Wrapper运行的主程序也是核心，他无法单独运行需要通过src/bin中的sh.script.in这个shell脚本调用，这个文件的使用我们之后会讲到。  
        运行wrapper可以按如上提示添加参数，如：./wrapper -c wrapper.properties  
  
        以上就是对Wrapper的一个整体认识，希望此文可以帮助大家更快的上手并使用Wrapper，之后几篇文章会详细讲解Wrapper的配置及定制自己的应用。