Deployment & Configuration

@version@ (@date@)

by Dieter Wimberger

Table of contents

1. About

This section documents the deployment and configuration possibilities of this library.

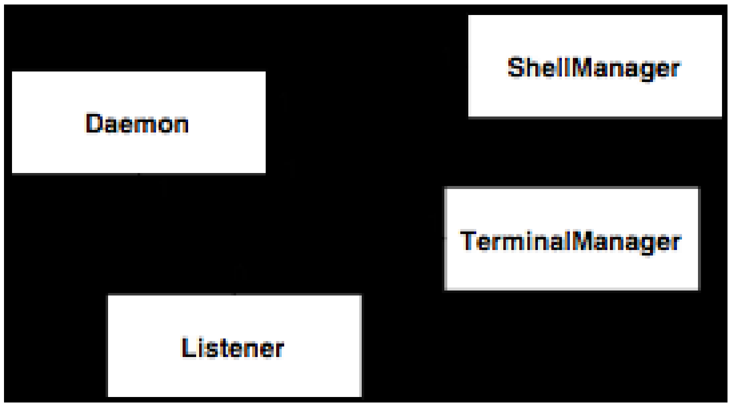
2. Overview

For using this library you will principally need to understand and work with two elements:

1. Configuration of important library provided elements
2. Development of an application specific shell class

3. Configuration

When deployed, telnetd will create the following set of important and configurable instances:



Runtime instances: Daemon, 1 ShellManager, 1 TerminalManager, n Listeners

3.1. The Daemon

TelnetD is a Singleton that will take care of loading and instantiating all required elements including logging.

The unified configuration file specifies all the configuration required for startup, except for logging. It is documented in more detail here (../deployment/daemon.html) .

3.2. The ShellManager

The ShellManager is a singleton that will take care of loading and providing shell instances.

The configuration file specifies a special section for the shells that are available as well as their implementing classes. Once you have written an implementation for a shell, this section needs to be updated.

3.3. The TerminalManager

The TerminalManager is a singleton that will take care of loading and providing instances of terminal type specific terminal implementations (e.g. ansi, vt100, xterm etc.).

The configuration file specifies in a special section the terminals that are available, their aliases as well as their implementing classes.

3.4. Listeners

Listeners will take care about accepting and managing connections.

For each listener defined in the configuration file there should be a related configuration section that specifies connection and connection management properties.

3.5. Logging Configuration

telnetd is now using the Jakarta Commons-Logging library and is delivered per default with log4j. The distribution contains a simple config file for console logging, but you might want to take a look at the Commons-Logging User Guide (http://jakarta.apache.org/commons/logging/commons-logging-1.0.4/docs/guide.html) .

4. Implementing a Shell class

The actual link between your application and the telnet daemon library is a shell implementation. The incoming connection will start with a configured login shell (see listeners configuration), that could for example authenticate the user.
A shell should basically take care about interpreting the user input (i.e. executing commands) and handle connection events (which allows your application to react on a broken connection etc.).

To help you with implementing the I/O, the library provides you with classes for doing I/O (including colors etc.). Probably you will want to take a look at the toolkit classes, which are somewhat inspired by (but not based on) NEWT (Not Erik's Windowing Toolkit), a library for building applications with text based interfaces ontop of Slang (if it does not ring a bell, it's the TUI we knew from old RedHat version's text installation).

I think it should be possible to implement reasonable TUI's even in Java. If you doubt, probably search Google (you'll find some way to use ncurses via JNI, jcurzez and CHARVA, which probably can somehow be adapted to work over the given I/O).

You should take a look at the shell implementation tutorial (../deployment/shell_tutorial.html) .

Many people have asked about cmd.exe and bash implementations. Well, probably you might want to look for Java shell implementations (which indeed exist) and adapt the code to run over the given I/O. Unfortunately most have never realized that it would be interesting to use their shells simply via a given InputStream and OutputStream (or Reader respectively Writer).

Note:
There will be a release of an extension package to telnetd2 soon, that will demonstrate a BeanShell wrapper.