

## The Design Behind jpf-shell

### The Need

When JPF is used on large complicated systems often the completely text based configuration and reporting of JPF can become quite large and unruly. This was often combated by creating IDE configurations that would help to simplify running JPF but of course these configurations were IDE specific. The main problem was that JPF had no official way to hook onto or create a GUI.

The need for JPF have a GUI isn't just limited to configuration issues. JPF's results from verifying complicated programs are often long and cumbersome. A GUI can be used to help ease navigating through JPF's output.

### The Requirements

- A GUI that can exist for configuration and reporting needs
- Backwards compatible with configurations that have already been created
- Must be able to interface with the IDE while still maintaining portability

### The Solution

- Implement a UI mechanism within JPF
  - JPF can be configured to launch a swing application instead verifying the System Under Test.
    - Swing application determined by the "shell=<Class of Shell>" property
    - The Swing app is in the same memory space (process) as JPF so it can still run and configure JPF as needed.
- Simple IDE Specific Plugins to launch JPF in a separate process
  - Eliminates the need to create IDE specific launch configurations.
  - The IDE plugins have open a socket to communicate with JPF (or more specifically a shell) allowing for communication between the two.

