

# The JPF Logging API

## [TracNav](#)

- [JPFWiki - Welcome Page](#)

[Introduction...](#)

[Installing JPF...](#)

[User Guide...](#)

[Developer Guide](#)

- [Design](#)
- [Choice Generator](#)
- [Partial Order Reduction](#)
- [Attributes](#)
- [Listener](#)

[MJI...](#)

- [Bytecode Factory](#)
- [Logging](#)
- [Report](#)
- [Embedded](#)
- [JPF tests](#)
- [JPF project layout](#)
- [Create a JPF project](#)
- [Coding Conventions](#)
- [Hosting update site](#)

[Projects...](#)

- [Summer Projects](#)
- [External Projects](#)
- [Change\(B\)log](#)

[About...](#)

- [Events](#)
- [Presentations](#)
- [Papers](#)
- [FAQ](#)
- [History?](#)
- [Support](#)
- [People?](#)
- [Playground](#)
- [Table of Context](#)

There is one simple rule:

## **Tips:**

don't use System.out or System.err for any permanent logging!

Of course we all do this temporarily during debugging, but it really shouldn't stay in the code. The logging infrastructure is quite easy to use. Just declare a static logger instance with an appropriate id (either package or logging topic) at the top of your class, and then use the Logger API to create output:

```

...
import java.util.logging.Level;
import java.util.logging.Logger;

package x.y.z;

class MyClass .. {
    static Logger log = JPF.getLogger("x.y.z");
    ...
    log.severe("there was an error");
    ...
    log.warning("there was a problem");
    ...
    log.info("something FYI");
    ...
    if (log.isLoggable(Level.FINE)){      // (1) don't create garbage
        log.fine("this is some detailed info about: " + something);
    }
    ...
}

```

Note that there is only one instance for each Logger ID, i.e. you can have a corresponding static field in all your relevant classes, and don't have to share the fields. Another aspect that is mostly important for the lower log levels (e.g. FINE) is that you should't concatenate log messages in operations that occur frequently (1), since the corresponding *StringBuilder* instances can cause performance degradations even if the log level is not set (the arguments still get evaluated). In this case, encapsulate the logging in *log.isLoggable(level){..}* blocks.