# The VIVO log file


- What does a log message look like?
- What is the right level for a log message?
- Setting the output levels
    - Production settings
    - Developer settings
    - Changing levels while VIVO is running


The VIVO log file contains time-stamped statements intended to help you

- identify the configuration of VIVO,
- monitor the progress of the application, and
- diagnose problems that occur.

The log file is written to the `logs` directory of your Tomcat application. It is usually called `vivo.all.log`, but the name may vary, depending on how your VIVO was installed.

The log file can also be helpful during development and debugging. This is particularly true if the developer takes advantage of the different logging levels.

## What does a log message look like?

Here is an example of some code that writes to the log

```
private static final Log log = LogFactory.getLog(WebappDaoSetup.class);
...
log.info(elapsedSeconds + " seconds to set up models and DAO factories");
```

and here is the resulting line in the log:

```
2012-11-15 12:20:37,406 INFO [<span class="confluence-link">WebappDaoSetup</span>] 3 seconds to set up models
and DAO factories
```

The log holds the time that the statement was written, the severity level of the message,
the name of the Java class that wrote the statement, and the contents of the statement itself.

Writing exceptions to the log can be tricky: check out this page on Writing Exceptions to the Log

## What is the right level for a log message?

Each log message has an output level (sometimes known as a severity level).
The most common levels are DEBUG, INFO, WARN, ERROR.
Each level conveys a sense of how important the message is.

| | |
|---|---|
| **ERROR** | Serious errors which need to be addressed and may result in unstable state. |
| **WARN** | Runtime situations that are undesirable or unexpected, but not necessarily "wrong", especially if the system can compensate; "almost" errors. |
| **INFO** | Interesting runtime events; routine monitoring information. Commonly used to describe how the system starts up, or changes that are worth noting as the system runs. |
| **DEBUG** | Used by developers when debugging their code. These messages will not appear in the log unless specifically enabled (see below) |

The logging framework also supports the levels of `FATAL` for very serious errors,
and `TRACE` for verbose debugging messages, but these are much less commonly used.

## Setting the output levels

### Production settings

The output levels for VIVO are determined by a file called *`[vitro-core]`*`/webapp/config/log4j.properties`

This file sets the general output level to `INFO`, which means that messages at the `INFO` level or higher will be written to the log. Messages at `DEBUG` or lower will not be written to the log.

The file also sets higher output levels for some classes that are otherwise too chatty with their log messages. So for example, the `StartupStatus` class is assigned an output level of `WARN`. This means that messages at the `WARN` level or higher will be written to the log, and messages at `INFO` or lower will not.

## Developer settings

Developers can make temporary changes to these settings by creating a file called *`[vitro-core]`*`/webapp/config/debug.log4j.properties`

When VIVO is rebuilt, the settings in this file will be used instead of the settings in the default file. A developer will commonly change the output level of the classes or packages he is currently working on, using this file.

The debug settings file is ignored by Git. As a result it remains unique to the individual developer, and can be changed without concern.

The debug settings file should not be present in a VIVO that is being built for production use.

## Changing levels while VIVO is running

You can change the log levels for individual Java classes while VIVO is running.

Direct your browser to `[vivo]/admin/log4j.jsp` This page requires that you log in to VIVO as an administrator.

This page shows a list of all Java classes with active Logger components. Each class has a drop-down list that allows you to set the log output level for that class. Select the level(s) you want, and scroll to the bottom of the page to click the button labeled `Submit changes to logging levels`. The change is effective immediately.

This feature should be used with care. A log level of `DEBUG` can significantly slow down some Java classes, and can result in very large amounts of output to the log of a busy system.

*Note: The `log4j.jsp` page shows only the classes with **active** Loggers. This means that you can't set use this page to set the output level of a class prior to the first time it is used. Java loads classes dynamically, and until the class is loaded, it does not have an active Logger.*