Logger类提供了多种方法来处理日志活动。 Logger类不允许实例化一个新的记录器实例，但它提供了两个静态方法获得一个 Logger 对象：

- **public static Logger getRootLogger();**
- **public static Logger getLogger(String name);**

此处两种方法的第一个返回应用程序实例根记录器并没有名字。任何其他命名的Logger对象实例是通过第二种方法通过记录器的名称获得。记录器名称是可以传递任何字符串，通常是类或包的名称，因为我们已经使用在最后一章。

```
static Logger log = Logger.getLogger(log4jExample.class.getName());
```

Logging 方法:

我们得到了一个名为记录器的实例之后，可以使用记录的几种方法来记录消息。 Logger类有专门用于打印日志信息下面的方法如下。

SN	方法及描述
1	**public void debug(Object message)** 这种方法打印使用 Level.DEBUG 消息级别
2	**public void error(Object message)** 这种方法打印使用 Level.ERROR 消息级别
3	**public void fatal(Object message);** 这种方法打印使用 Level.FATAL 消息级别
4	**public void info(Object message);** 这种方法打印使用 Level.INFO 消息级别
5	**public void warn(Object message);** 这种方法打印使用 Level.WARN 消息级别
6	**public void trace(Object message);** 这种方法打印使用Level.TRACE消息级别

所有的级别定义在org.apache.log4j.Level类中，并且任何上述方法都可以调用如下：

```
import org.apache.log4j.Logger;

public class LogClass {
   private static org.apache.log4j.Logger log = Logger
                                    .getLogger(LogClass.class);
   public static void main(String[] args) {
      log.trace("Trace Message!");
      log.debug("Debug Message!");
      log.info("Info Message!");
      log.warn("Warn Message!");
      log.error("Error Message!");
      log.fatal("Fatal Message!");
   }
}
```

当编译并运行LogClass程序会产生以下结果：

Debug Message!
Info Message!
Warn Message!
Error Message!
Fatal Message!
所有的调试消息更有意义，当它们在级别组合使用。级别将在下一章介绍，那么在下一节会有一个很好的理解及如何使用这些方法在不同的级别调试。