Log và logger

# Log là gì?

Log là hành động ghi lại tất cả các sự kiện xảy ra trong hệ điều hành, trong một ứng dụng nào đó hoặc trong trình duyệt vào các tập tin log. Các tập tin log có thể đơn giản chỉ là một tập tin văn bản thông thường. Tập tin log cung cấp các minh chứng về những gì bạn đã làm, bạn đã mở những trang web nào, gửi mail cho ai, down những tập tin nào.

Những tập tin log này có thể được tìm thấy trong:

* Hệ điều hành
* Trình duyệt web (dưới dạng các tập tin cache)
* Các ứng dụng (dưới dạng các tập tin backup)
* Email

Ví dụ về một tập tin log:

*1/1/2009 - New year*

*1/2/2009 - Date formatted log.*

*2/8/2009 - More log information*

*5/12/2009 - Done with log*

# Ý nghĩa của log

* Log sẽ giúp các nhà phát triển biết và tìm ra cách sửa chữa khi một ứng dụng của họ bị lỗi hoặc bị crash trên máy của người dùng.
* Log giúp người dùng có thể khôi phục lại trạng thái của chương trình nếu họ lỡ tay xóa một cái gì đó hoặc chương trình bị lỗi.
* Log còn giúp người dùng đồng bộ thông tin trên một chiếc điện thoại thông qua phương thức “Backup and restore”.
* Kết hợp với phân tích thống kê, log sẽ giúp các nhà phát triển ứng dụng biết được nhu cầu của người dùng từ đó nâng cao chất lượng sản phẩm tốt hơn, điển hình nhất là tập đoàn Google.
* Hoặc đơn giản log sẽ cho người dùng biết được lịch sử duyệt web của họ, những trang họ đã mở, những tin nhắn họ đã gửi.
* Tuy nhiên, log có thể sử dụng với mục đích xấu là đánh cắp thông tin người dùng bằng cách ghi lại các sự kiện trên máy tính của họ và gửi đến cho các tin tặc

# Log trong Java

Log trong Java chỉ đơn giản là một quá trình ghi lại các log messages trong suốt quá trình thực thi của chương trình về một nơi cố định. Việc ghi log này sẽ cho phép bạn báo cáo lỗi, các warning messages cũng như info messages mà sau này có thể được lấy ra và phân tích.

Đối tượng thực hiện việc ghi log trong ứng dụng thì được gọi là Logger

# Logger

Lớp java.util.Logger là một lớp chính để truy cập vào Java logging API. Cách tạo:

Logger logger = Logger.getLogger("myLogger");

Chuỗi truyền vào phương thức getLogger() sẽ là tên của Logger được tạo. Bạn có thể đặt tên thoải mái nhưng nó sẽ quyết định Logger nằm ở đâu trong Hệ thống Logger (Logger Hierarchy). Mỗi dấu chấm (.) trong tên bạn đặt sẽ là một nhánh trong Hệ thống Logger. Hãy nhìn ví dụ dưới đây:

myApp

myApp.user

myApp.admin

Tất cả các tên trên đều sử dụng được. Chúng tạo thành một hệ thống. Tên “myApp” nằm ở đỉnh của hệ thống. Hai tên “myApp.user” và “myApp.admin” là con của tên “myApp”

Nếu bạn muốn biết tên của Logger bạn có thể gọi phương thức getName():

String name = logger.getName();

Người ta thường dùng tên lớp của lớp tạo ra Logger, bao gồm cả package name, để đặt tên cho Logger.Ví dụ:

Logger logger = Logger.getLogger( MyClass.class.getName());

## Các phương thức log()

Nhóm phương thức log() sẽ ghi lại message tại một mức log (log Level) cố định. Mức log sẽ được truyền vào dưới dạng tham số. Các mức log này có các hằng số cố định.

Một số phương thức log() có thể dùng đối tượng làm tham số. Những đối tượng này sẽ được đưa vào log message trước khi được ghi xuống. Việc đưa đối tượng này vào log message chỉ diễn ra nếu như message không được lọc. Một số ví dụ:

Logger logger = Logger.getLogger("myLogger");

logger.log(Level.SEVERE, "Hello logging");

Đây là những gì được ghi vào console (nơi ghi log mặc định):

08-01-2012 14:10:43 logging.LoggingExamples main

SEVERE: Hello logging

Còn đây là ví dụ khi truyền một tham số vào message: [1]

logger.log(Level.SEVERE, "Hello logging: {0}, {1}",

new Object[] {"P1", "P2"});

Đây là những gì được ghi xuống:

08-01-2012 14:45:12 logging.LoggingExamples main

SEVERE: Hello logging: P1, P2

Còn đây là một ví dụ về một đối tượng Throwable:

logger.log(Level.SEVERE, "Hello logging",

new RuntimeException("Error"));

Và được ghi xuống như sau:

08-01-2012 14:54:29 logging.LoggingExamples main

SEVERE: Hello logging

java.lang.RuntimeException: Error

at logging.LoggingExamples.main(LoggingExamples.java:18)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39) at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)

at java.lang.reflect.Method.invoke(Method.java:597)

at com.intellij.rt.execution.application.AppMain.main(AppMain.java:120)

## Các phương thức logp()

Các phương thức logp() hoạt động giống với các phương thức log() nhưng mỗi phương thức có thêm hai tham số: sourceClass và sourceMethod.

Hai tham số này sẽ cho biết lớp nào và phương thức nào mà log message đã được tạo ra.

## Các phương thức logrb()

Các phương thức logrb() cũng giống các phương thức log(), nhưng ngoài ra chúng có thể lấy log message từ một resource bundle. Một resource bundle là một tập hợp chứa các cặp khóa và giá trị. Các khóa và giá trị này không đổi bất chấp ngôn ngữ nào.

Ví dụ về logrb():

logger.logrb(Level.SEVERE, "logging.LoggingExamples", "main",

"resources.myresources", "key1");

Trong ví dụ này chúng ta tìm trong resource bundle có tên là resources.myresources với khóa là “key 1”. Nếu trong resource bundle không có chứa khóa nào tên là “key 1” thì giá trị ghi xuống log sẽ là “key 1”, còn nếu tìm thấy nó sẽ lấy giá trị tương ứng với khóa “key 1” trong resource bundle và ghi xuống. Đây là những gì trong resource bundle:

key1 : This is message 1

key2 : this is message 2

Và kết quả ghi xuống console là:

08-01-2012 17:14:39 logging.LoggingExamples main

SEVERE: This is message 1

## Những phương thức log khác

Logger còn có các phương thức log sau đây:

entering(String sourceClass, String sourceMethod);

entering(String sourceClass, String sourceMethod, Object param1);

entering(String sourceClass, String sourceMethod, Object[] params);

exiting (String sourceClass, String sourceMethod);

exiting (String sourceClass, String sourceMethod, Object result);

fine (String message);

finer (String message);

finest (String message);

config (String message);

info (String message);

warning (String message);

severe (String message);

Mỗi phương thức tương ứng với một mức log khác nhau cùng tên với nó. Sử dụng những phương thức nà để ghi log tương ứng với sử dụng phương thức log().

# Logger Hierarchy

Phần này chúng ta sẽ nói kĩ hơn về việc tổ chức hệ thống Logger/Cây Logger

Xem xét ví dụ sau:

Logger logger = Logger.getLogger("com.jenkov.web");

Trong ví dụ này tên của Logger được tạo ra là com.jenkov.web

Tên của Logger sẽ chỉ ra một hệ thống Loggers. Mỗi dấu chấm (.) sẽ đánh dấu một mức trong hệ thống. Với tên com.jenkov.web thì Logger có 3 Logger cha là:

""

"com"

"com.jenkov"

Nếu bạn gọi getParent() của Logger vừa được tạo trên, kết quả sẽ trả về com.jenkov. Và bạn gọi tiếp getParent() ở Logger đó sẽ nhận được kết quả là com. Đỉnh của hệ thống sẽ là Logger với tên rỗng (“”).

***Chú ý:***

Bạn phải tạo hệ thống cây logger theo thứ tự từ trên xuống. Nếu bạn tạo như sau:

Logger logger = Logger.getLogger("com.jenkov.web");

..khi gọi phương thức getParent() bạn sẽ nhận được Logger với tên rỗng (“”). Lý do là tại những Logger trung gian chưa được tạo ra. Bạn phải tạo hệ thống cây Logger như sau:

Logger logger = Logger.getLogger("");

Logger logger1 = Logger.getLogger("com");

Logger logger2 = Logger.getLogger("com.jenkov");

Logger logger3 = Logger.getLogger("com.jenkov.web");

# References

|  |  |
| --- | --- |
| [1] | C. Hope, "ComputerHope.com," [Online]. Available: http://www.computerhope.com/jargon/l/log.htm. [Accessed 21 03 2015]. |
| **[2]** | **Jenkov, "Java-logging," [Online]. Available: http://tutorials.jenkov.com/java-logging/index.html. [Accessed 21 03 2015].** |
| [3] | K. Bonsor, "How stuff works," HowStuffWorks, [Online]. Available: http://computer.howstuffworks.com/workplace-surveillance4.htm. [Accessed 21 03 2015]. |
| [4] | Vogella, "Vogella," Vogella, [Online]. Available: http://www.vogella.com/tutorials/Logging/article.html. [Accessed 21 03 2015]. |