CHAPTER 20

Logging

Jini 2.0 introduced use of the `java.util.logging` package. This package can be used for auditing, or possibly for debugging, and it allows an application to write to named logs. Typically, the name of a log is a package name. For example, the log for `LookupDiscovery` objects is `net.jini.discovery.LookupDiscovery`. Each object will write a record of its activities to the class log. These messages will have levels such as `SEVERE`, `INFO`, and so on down to `FINEST`. For example, `LookupDiscovery` objects will write these messages, among others:

- `SEVERE` when a network interface is bad or not configured for multicast
- `INFO` when any exception other than an `InterruptedIOException` occurs while attempting unicast discovery
- `FINEST` when a discovered, discarded, or changed event is sent

To see what is put into a log, a `Handler` must be added to the log. There are several supplied handlers, including the following:

- `ConsoleHandler`: This handler writes all log messages to the console that have `INFO` level or above. The format is given by the `SimpleFormatter` object, and just gives brief readable messages.
- `FileHandler`: This handler writes all messages to a file, using an XML format. If you only want simple messages, this can be set to be a `SimpleFormatter`.

In this chapter, we'll discuss how this is used in a typical Jini object and how you can log events from such an object.

Logging LookupDiscovery

A program to log activities of a `LookupDiscovery` object is given by this program, which just adds logging to the earlier `MulticastRegister` program:

```
package basic;
import net.jini.discovery.LookupDiscovery;
import net.jini.discovery.DiscoveryListener;
import net.jini.discovery.DiscoveryEvent;
import net.jini.core.lookup.ServiceRegistrar;
import java.lang.reflect.*;
import java.util.logging.*;
```

```
/**
 * MulticastRegisterLogger.java
 */
public class MulticastRegisterLogger implements DiscoveryListener {
    static final String DISCOVERY_LOG = "net.jini.discovery.LookupDiscovery";
    static final Logger logger = Logger.getLogger(DISCOVERY_LOG);
    private static FileHandler fh;

    static public void main(String argv[]) {
        new MulticastRegisterLogger();
        // stay around long enough to receive replies
        try {
            Thread.currentThread().sleep(10000L);
        } catch(java.lang.InterruptedException e) {
            // do nothing
        }
    }

    public MulticastRegisterLogger() {
        try {
            // this handler will save ALL log messages in the file
            fh = new FileHandler("mylog.txt");
            // the format is simple rather than XML
            fh.setFormatter(new SimpleFormatter());
            logger.addHandler(fh);
        } catch(Exception e) {
            e.printStackTrace();
        }
        // this handler will write all INFO and
        // above messages to the console
        logger.addHandler(new ConsoleHandler());
        System.setSecurityManager(new java.rmi.RMISecurityManager());
        LookupDiscovery discover = null;
        try {
            discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);
        } catch(Exception e) {
            System.err.println(e.toString());
            e.printStackTrace();
            System.exit(1);
        }
        discover.addDiscoveryListener(this);
    }

    public void discovered(DiscoveryEvent evt) {
        ServiceRegistrar[] registrars = evt.getRegistrars();
        for (int n = 0; n < registrars.length; n++) {
            ServiceRegistrar registrar = registrars[n];
```

```
            // the code takes separate routes from here for client or service
            System.out.println("found a service locator");
          }
    }
    public void discarded(DiscoveryEvent evt) {
    }
} // MulticastRegister
```

When this program is run, a few messages will be printed to the console. A great deal more will be written to the `mylog.txt` file, including a line like this whenever a lookup locator is found:

```
FINEST:     discovered locator  = jini://jannote.jan.edu.au/
```

Summary

This short chapter has looked at the logging package and its use in the core Jini classes. It has shown how logging information collected by a Jini object can be used by a program to give extra information about what the Jini object is doing.