GPars - Groovy Parallel Systems

Jon Kerridge

Version 1.0, 2015-10-29

Table of Contents

Anonymous Network Channels: A Print Server 1
Operational Details 1
Multi-User Operations 1
A Diagram Sums It Up 1

Anonymous Network Channels: A Print Server

A print server is probably the simplest service used by users of a networked service. It provides a means whereby a user can send a file for printing using a printer shared among the users of the network. In this implementation, a print service will be constructed that accepts print lines, a line at a time, from a user. The print service will accept print lines from a number of users, in parallel, up to some limit set when the print server is installed.

Operational Details

Once the user has sent all the lines of text to be printed; the print server will then output those lines in a single printed output. The printed output will be preceded by a job number that can be recognised by the user of the service. The user of the service will be informed both when their job has been accepted and when it has completed. The user will be unaware that the print service is dealing with other user requests. The print service should run in the background and always be ready to accept requests from a user, that is, the user processes should start asynchronously with the print service process. The order in which the respective processes start should have no bearing on the operation of the system.

From the foregoing, it is obvious that users need to request that their lines of output are sent to the print service and subsequently on completion of their output, the user needs to indicate that the lines of text can be printed. To this end, the print service provides two named channels by which the user can request and subsequently release their use of the print service.

Multi-User Operations

In addition, if the print service is going to manage print operations from more than one user in parallel then some means of telling the user which of the services to use will be required. The user also needs to be able to send lines to be printed to the print service. These connections will change with each print job and thus the corresponding network channels will be created dynamically and anonymously.

The architecture of the system is shown in `Figure 16-1`. The **PrintSpooler** process provides the print service using two named `Net2One` channels called `request` and `release`. The network connections are indicated by the dashed lines, one for each channel. Each **PrintUser** process can dynamically connect to the request and release channels by defining them as `Any2Net` when their node is created. In order to avoid multiple communications on the request and release channels, only one communication will be permitted on each channel for each print job.

A Diagram Sums It Up

The diagram shows the state when the **PrintSpooler** is willing to accept print lines from up to two **PrintUser** processes in parallel. These have been given names for clarity but in reality are anonymous. The `useChannel` is used by **PrintSpooler** to tell the **PrintUser** the location of the `printChannel`. It is used to send the lines to the **PrintSpooler**. The `printChannel` is used to send the lines to be printed to the

PrintSpooler.

Want to read more of this chapter? Download this chapter's PDF here.