Active Objects in Symbian OS

This article provides an overview of the Symbian C++ Active Object Framework. For a more complete explanation see the canonical reference: Fundamentals of Symbian C++/Active Objects.

Introduction

Active Object provides a support for asynchronous processing by encapsulating the service request and service completion response.

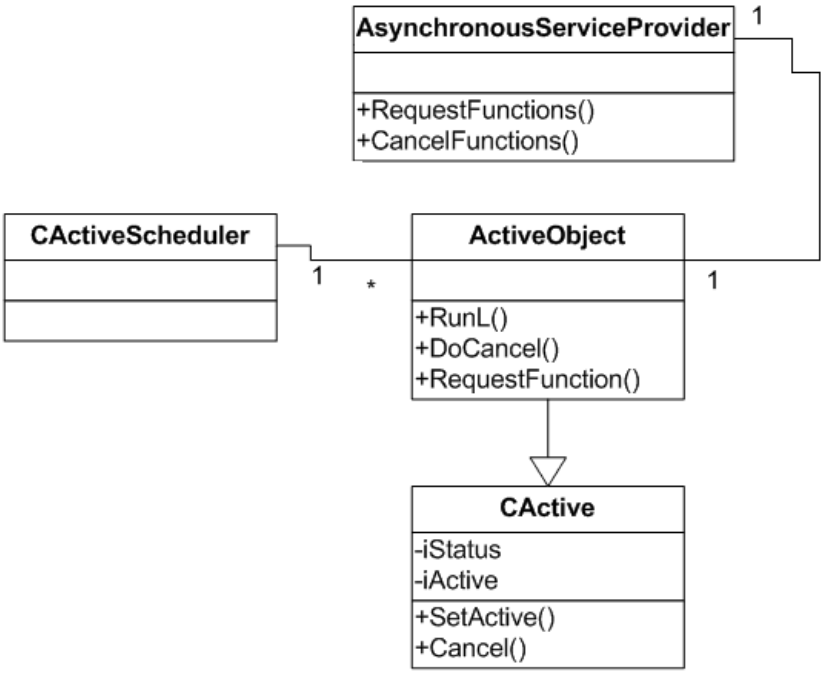
Framework

The pattern is used in the context of client-server architectures, when the server's request processing time is long enough, and therefore the client may utilise this time for performing other tasks asynchronously with the server.

In asynchronous processing, a client requests the services of a server by issuing asynchronous requests, i.e. the requesting function does not block after issuing a request. The server informs the client when the service is complete.

Writing the code implementing asynchronous requests, one needs to address a large number of low-level details. This makes the code more difficult to understand and reuse, and may result in hidden programming errors. To address the limitations of mobile terminals, the implementation should also have small memory footprint.

Active Object encapsulates asynchronous requests. The client makes requests by calling specific function of Active Object. Upon the completion of the request, the Active Object's respective function (RunL) is called. The structure of the pattern is illustrated below:



UML class diagram of Active Object framework

Each ActiveObject is associated with an AsynchronousServiceProvider which offers a request function taking as a parameter a status variable. When ActiveObject's request function is called, the service request is passed to AsynchronousServiceProvider. At the same time, the status variable is updated to reflect pending status. When the service provider changes the status variable according the results of processing (e.g. KErrNone) and signals the requesting thread's CActiveScheduler about the completion. It is the responsibility of scheduler to identify ActiveObject to be resumed.

The active objects and the active scheduler are situated within a same thread (however different from the tread of AsynchronousServiceProvider) and work together to provide 'co-operative multitasking' with a thread.

Active Object runs in a same thread as the application, thus, reducing the memory requirements and eliminating additional overhead of context-switching, which would have been incurred if the Active Object was implemented in separate thread or in separate thread or in separate process. The single-thread implementation also eliminates the problem of thread synchronisation, and simplifies debugging.

The liability of Active Object is the fact that it is non-preemptive, i.e. only one event can be processed by RunL at a time.

(C) Copyright Nokia 2014. All rights reserved.

Therefore, RunL must finish quickly; this is sometimes difficult to accomplish.

Representation

Active Objects represent asynchronous service request, encapsulating:

- A data member representing the status of the request(iStatus).
- A handle on the asynchronous service provider(usually an R-Class Object).
- Connection to the service provider during construction.
- The function to issue(or reissue) the asynchronous request (user-defined)
- The handler function to be invoked by the active scheduler when the request completes(RunL()).
- The function to cancel an outstanding request(Cancel()).

Conclusion

Active Objects are often used in Symbian C++ programming, where an Active Object can be implemented by deriving from the CActive class and implementing necessary methods. The Active Scheduler need not be implemented by the programmer as it is provided by the Symbian OS application architecture.

Other Related Links

- Active Object Example
- Splitting long running tasks with active objects
- Simple Timer implementation Using Active Objects
- Responsibilities of Asynchronous Service Provider in Active Objects
- How to add Active Object to the Active Scheduler

External Links

- Active Objects

(C) Copyright Nokia 2014. All rights reserved.