1.1 Notifying Thread Manager Pattern

1.1.1 Name and Source

Notifying Thread Manager
Page 108-116 in the book ".NET Patterns: Architecture, Design, and Process " [NET 03]

1.1.2 Also Known As

-

1.1.3 Type

1.1.4 Intent

To let a client invoke a long-running web service call without blocking the client while waiting for the response, even if the server does not provide a notification mechanism.

1.1.5 Problem

Essentially, after generalizing the originally described problem that only covers “.NET Windows Forms”, the problem is very similar to the “Event Monitor Pattern”, so you may want to refer to that pattern for a description of the problem.

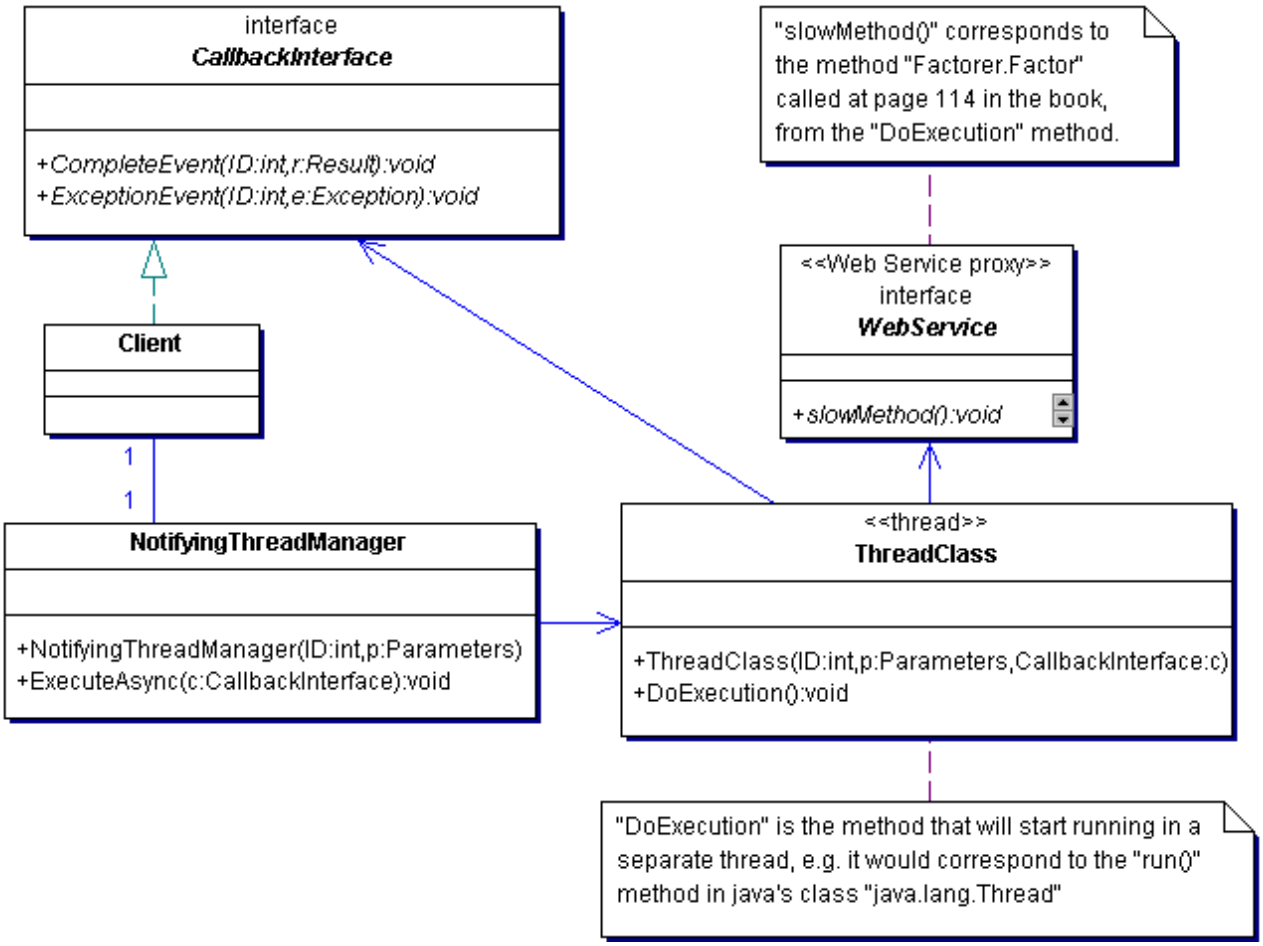
How can a web service client invoke a long-running web service call without getting the client application blocked while waiting for the response ?

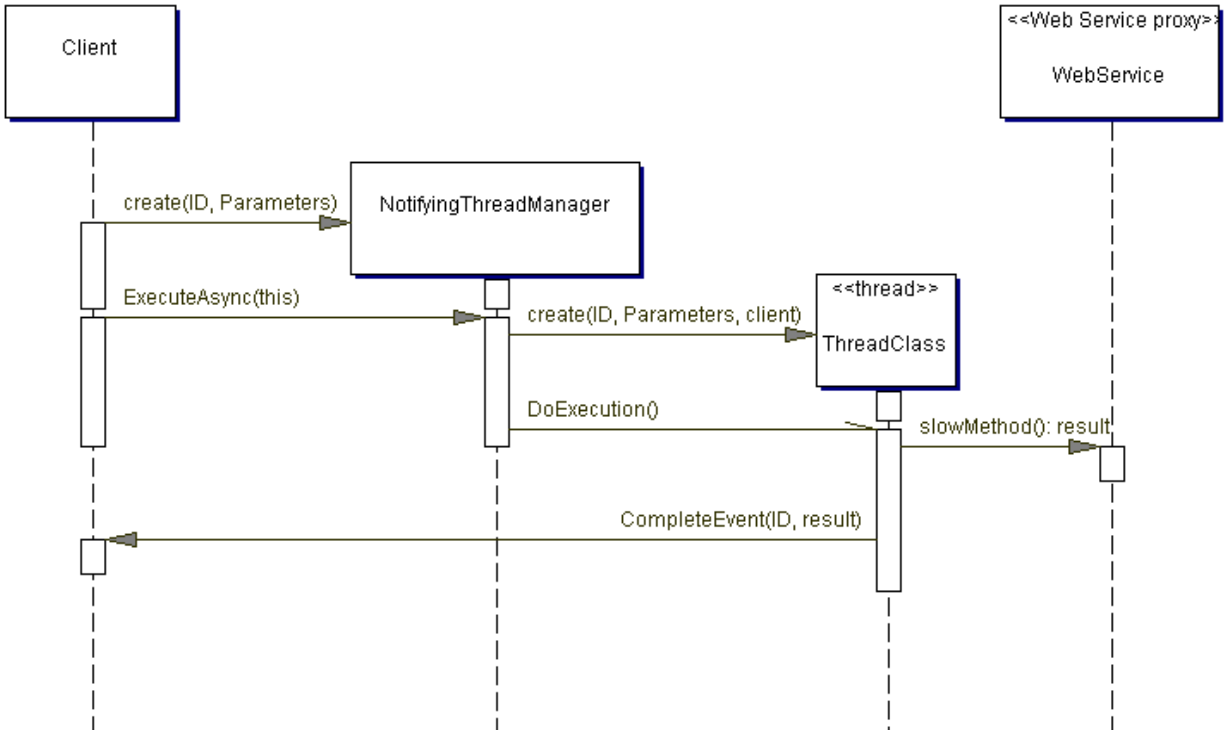
1.1.6 Forces

1.1.7 Solution

Let the client implement a callback interface that will be used by another thread at the client application when the long-running web service method has finished.

The book where this pattern was found is very .NET focused and the pattern discusses how to notify a .NET “System.Windows.Forms.Control” about when a long-running web service is finished. Further, the solution to the pattern also includes so called “delegates” which is similar to a function pointer in some other languages such as C or C++, but a delegate is object-oriented and thus can not only reference “functions” (i.e. static methods) but also instance methods. Delegates may be an interesting feature for .NET programmers, but to make the pattern more generally interesting, i.e. also for a programming language such as java, the delegates and the Windows control to be notified have been replaced with a solution based on a more generically applicable callback interface.


interface
CallbackInterface
+CompleteEvent(ID:int,r:Result):void
+ExceptionEvent(ID:int,e:Exception):void
"slowMethod()" corresponds to the method "Factorer.Factor" called at page 114 in the book, from the "DoExecution" method.
<<Web Service proxy>>
interface
WebService
+slowMethod():void
Client
1
1
NotifyingThreadManager
+NotifyingThreadManager(ID:int,p:Parameters)
+ExecuteAsync(c:CallbackInterface):void
<<thread>>
ThreadClass
+ThreadClass(ID:int,p:Parameters,CallbackInterface:c)
+DoExecution():void
"DoExecution" is the method that will start running in a separate thread, e.g. it would correspond to the "run()" method in java's class "java.lang.Thread"



Client – A consumer of a long-running web service. It does not invoke the thread nor uses the web service proxy itself, but it only uses the NotifyingThreadManager. The parameters may include a unique ID that in the notification later will enable the Client to determine which invocation has finished, since the Client might trigger many invocations that will execute in parallel threads.

NotifyingThreadManager – A helper object that provides an API to the Client. It creates the thread and delegates the parameters from the Client, including a reference to the Client object. Then it starts the thread with an asynchronous call and returns control to the client without blocking it as it would do with a synchronous call to a long-running web service.

ThreadClass – A thread that invokes a synchronous call to a long-running web service. When the web service has finished the thread notifies the client that implements the callback interface. This notification includes not only the result but also an ID that lets the Client identify which web service invocation has finished. If the execution of the web service finishes successful then the “CompleteEvent” method will be invoked, as illustrated in the sequence diagram, but otherwise an “ExceptionEvent” method will be used instead.

WebService – A proxy object for the long-running web service.

1.1.8 Consequences

If multiple threads can complete and send callbacks at the same time you may need to take care of synchronization issues. In the original pattern this is done with the thread safe .NET method “System.Windows.Forms.Control.BeginInvoke(Delegate, Object[])”.

1.1.9 Related patterns

"Event Monitor pattern" and the other notification or polling patterns:

- "Observer pattern for Web Services"
- "Publish/subscribe Pattern"
- "Pollable Thread Manager"
- "Multisync Thread Manager"