1.1 Pollable Thread Manager Pattern

1.1.1 Name and Source

Pollable Thread Manager
Page 116-122 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 provide a framework for executing a long-running method and periodically check if it has completed.

1.1.5 Problem

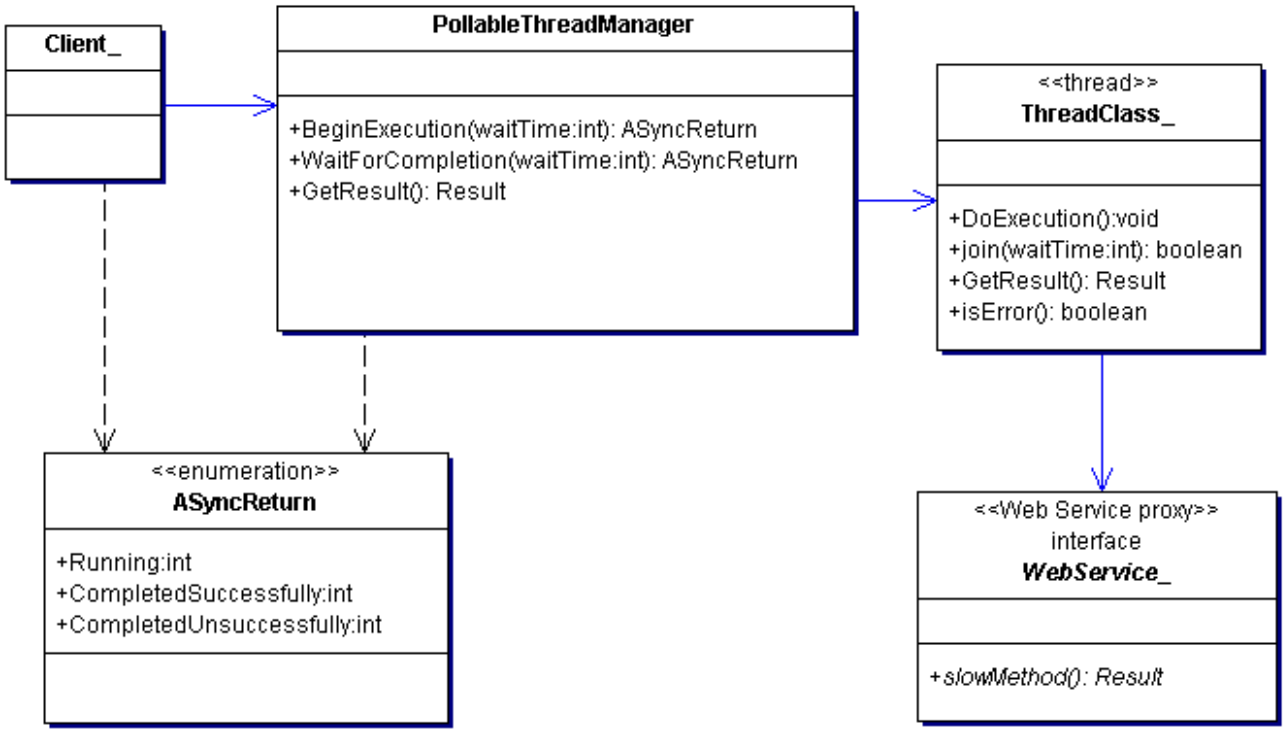
Essentially, the problem is very similar to the “Event Monitor Pattern”, so you may want to refer to that pattern for a description of the problem.

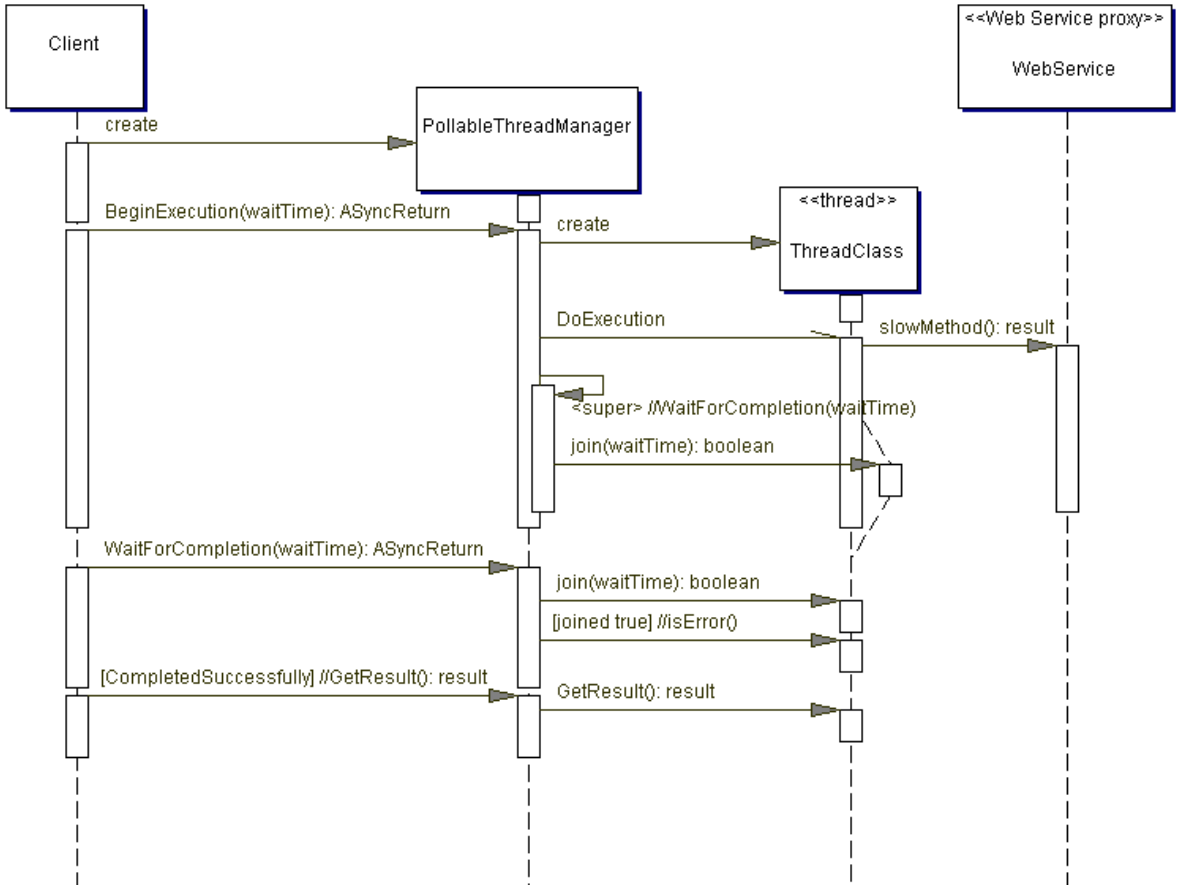
How can a web service client invoke a long-running web service call without getting the client application blocked while waiting for the response ?

1.1.6 Forces

1.1.7 Solution

Create a Pollable Thread Manager class that will invoke an **as**ynchronous call to a thread class running in a separate thread. This thread class will then invoke a **s**ynchronous Web Service method and will save the result in an attribute that can be returned later to the Client through the Pollable Thread Manager. The Client then has to continuously invoke the method “WaitForCompletion” of the Pollable Thread Manager that each time waits for a while to see if the thread will complete. When that method returns “AsyncReturn.CompletedSuccessfully” the Client can invoke “GetResult” to finally get the result from the long-running web service.


Client_
PollableThreadManager
+BeginExecution(waitTime:int): ASyncReturn
+WaitForCompletion(waitTime:int): ASyncReturn
+GetResult(): Result
<<thread>>
ThreadClass_
+DoExecution():void
+join(waitTime:int): boolean
+GetResult(): Result
+isError(): boolean
<<enumeration>>
ASyncReturn
+Running:int
+CompletedSuccessfully:int
+CompletedUnsuccessfully:int
<<Web Service proxy>>
interface
WebService_
+slowMethod(): Result


Client
<<Web Service proxy>>
WebService
create
PollableThreadManager
BeginExecution(waitTime): ASyncReturn
create
<<thread>>
ThreadClass
DoExecution
slowMethod(): result
<super> //WaitForCompletion(waitTime)
join(waitTime): boolean
WaitForCompletion(waitTime): ASyncReturn
join(waitTime): boolean
[joined true] //isError()
[CompletedSuccessfully] //GetResult(): result
GetResult(): result

[the occurrence of "<super>" in the diagram is not intentional and will be removed in the final version]

Client – An object that periodically checks for completion. Maybe the programming language or class library you are using provides some kind of timer feature that can be set to regularly invoke the method “PollableThreadManager.WaitForCompletion”.

PollableThreadManager – An object that creates and starts the thread where the web service invocation will be done.

ThreadClass – A thread object with a method “DoExecution” that runs in a separate thread, and the code will probably run within a try/catch clause if the programming language supports that kind of error handling. If an error is catched then the “isError” method will return true when it will be invoked from the Pollable Thread Manager (The public and asynchronous message “DoExecution” to the thread class corresponds to the “start” method in a java thread which is implemented with the “run” method and invoked by the start method).

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

ASyncReturn – An enumeration, or a class with constants, that defines status values, e.g. one value for successful completion and one value for failure. As an option to this class or enumeration, the method “PollableThreadManager.WaitForCompletion” might instead return an integer between 0 and 100 that represents an estimation of how many percent of the “WebService.slowMethod” that has executed. However, such an estimation can probably not be done easily from the client without some help method from the web service itself.

1.1.8 Consequences

1.1.9 Related patterns

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

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