.NET Enterprise Services

Venkat Subramaniam
venkats@agiledeveloper.com

June 2003

Presentation and examples can be downloaded from
http://www.agiledeveloper.com/download.aspx

Abstract

Abstract .NET Enterprise Services provides a number of services like object pooling, events, queued components, and distributed transactions. In this presentation we discuss the Enterprise Services in general and focus on the distributed transaction facilities. New features in 2003 version (Framework 1.1) are presented as well. The objective of this presentation is for the attendees to understand the benefits of using .NET Enterprise Services and leave with enough information to start utilizing it in their applications.

Speaker Dr. Venkat Subramaniam is an agile developer who teaches and mentors software developers. He has significant experience in architecture, design and development of distributed object systems. Venkat has trained more than 2500 software professionals around the world. He is also an adjunct professor at University of Houston and teaches the Professional Software Developer Series at Rice University's Technology Education Center.

Examples Any page with a  has an example attached
Download from http://www.agiledeveloper.com/download.aspx

.NET Enterprise Services

- **Why and What?**
- Object Pooling
- JIT Activation
- Queued Components
- Transactions
- Controlling Object Creation
- Managing DB Access
- V 1.1 Enhancements
- Conclusion

Before we start

- Things to know
- **AppDomain** provides isolation within a .NET process
- Part of an assembly's identity is its name, version number, culture and **strong name**
- In addition to the keywords, capabilities of a class may be extended by using **attributes**
- A COM accessible class has a **GUID or CLSID**
- For COM accessibility, you need to register the **type library** for the component

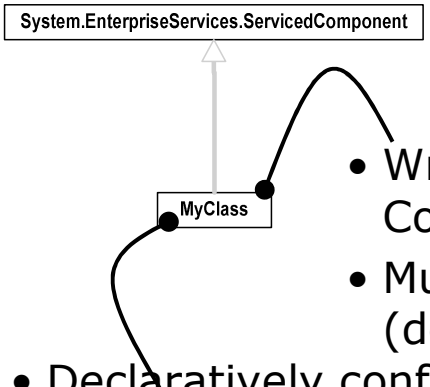
Why use Enterprise Services?

- Benefit from Infrastructural facilities
  - reduces your code size
  - easier to implement complex constructs
  - faster development
  - Scalability
  - more robust
- If you had a need to use COM+, you would need this as well
- OK, what is it?
  - **Integration of COM+ services into .NET**

Serviced Components

- A Serviced Component allows COM+ services to be available to .NET Framework classes
- Features supported
  - JIT Activation
  - Object Pooling
  - Queued Components
  - Transactions
  - (and other COM+ services)
- **System.EnterpriseServices** namespace provides facilities to access these

Writing a Serviced Component



- Written in a CLS-Compliant Language
- Must have no-argument (default) constructor
- Declaratively configured using Attributes
- COM+ Catalog uses this information
- COM+ provides the context for execution
- Registered *dynamically* or manually

Dynamic vs. Manual Registration

- Dynamic registration
  - On first request from a managed client, a ServicedComponent object is registered automatically by the runtime
  - assembly not placed in GAC
  - Manual registration required if client is COM
- Manual registration
  - use regsvcs.exe to register
  - required for unmanaged clients to access
  - useful to perform design time tests

Activation Types

- ActivationOption.Library
  - ServicedComponent created in caller's process
- ActivationOption.Server
  - ServicedComponent created in a new process
  - Assembly must be in GAC
  - Types of parameters passed to methods must either be Serializable or inherit from MarshalByRefObject

.NET Enterprise Services

- Why and What?
- **Object Pooling**
- JIT Activation
- Queued Components
- Transactions
- Controlling Object Creation
- Managing DB Access
- V 1.1 Enhancements
- Conclusion

Object Pooling

- Allows reusing an object among method calls from different clients
- Useful if object creation overhead needs to be eliminated
- You can set the
  - minimum # of objects in pool
  - maximum # of objects in pool
  - creation timeout
- *Client must call DisposeObject()*
  - to signal that object may be returned to pool
- Should ActivationOption be Server or Library?

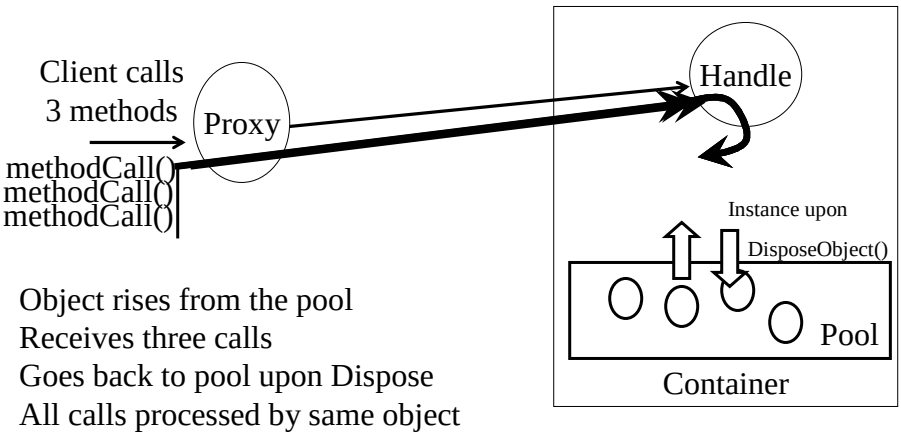
ActivationOption in Object Pooling

- Server
  - Out of process clients will use the objects that are created in the Default Domain
  - Seems obvious, isn't it?
- Library
  - This is a mess?!
  - Behavior depends on the OS
  - In 2000, behaves like Server option
  - In XP and 2003, one pool ***per application domain*** is maintained

.NET Enterprise Services

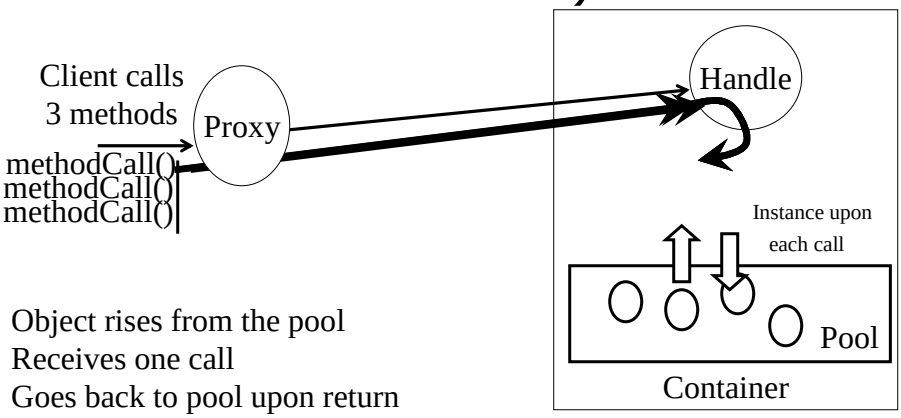
- Why and What?
- Object Pooling
- **JIT Activation**
- Queued Components
- Transactions
- Controlling Object Creation
- Managing DB Access
- V 1.1 Enhancements
- Conclusion

Instance Pooling vs. Swapping Pooling (Object Pooling)



Object rises from the pool
Receives three calls
Goes back to pool upon Dispose
All calls processed by same object
object may process request from multiple clients over time

Instance Pooling vs. Swapping Swapping (Object Pooling + JIT Activation)



Object rises from the pool
Receives one call
Goes back to pool upon return
Each call may be processed by different server side objects

JIT Activation

- While the client holds reference, the object is disposed
- The context is saved however
- Next call from client to object will be handled by another instance
- Similar to how session beans behave in EJB
  - instance swapping

JIT vs. Object Pooling

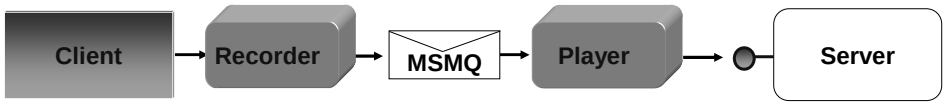
- If you intend to make one call only on the pooled object from each client, then combine JIT Activation with Object Pooling
- If you intend to make multiple calls on the object, use only object pooling

.NET Enterprise Services

- Why and What?
- Object Pooling
- JIT Activation
- **Queued Components**
- Transactions
- Controlling Object Creation
- Managing DB Access
- V 1.1 Enhancements
- Conclusion

Queued Components

- Provides for loose coupling
- Asynchronous processing of messages



- Works
  - disconnected
  - when server is slow or has failed

Quiz Time

1. Which class should your component inherit from?
   ServicedComponent
2. What are the two activation types?
   Library, Server
3. What's the effect of Library activated object pooling on XP?
   One pool per AppDomain of client
4. What's the different between JIT Activation and object pooling?
   Object deactivated right after a method call in JIT
5. When should you use JIT Activation and when to use object pooling
   Clients typically make single call?, use JIT Activation
6. What methods you must override for object pooling?
   Activate, Deactivate, CanBePooled
7. What method is the client required to call when done with a pooled object
   DisposeObject
8. From what class do you obtain the interface for a queued component
   System.Runtime.InteropServices.Marshal
9. When is a ServicedComponent registered with COM+
   Manually, or automatically @ 1st .NET client invocation
10. What tool is used to manually register the component
    regsvcs.exe

.NET Enterprise Services

- Why and What?
- Object Pooling
- JIT Activation
- Queued Components
- **Transactions**
- Controlling Object Creation
- Managing DB Access
- V 1.1 Enhancements
- Conclusion

Transaction Integrity

- Set of related tasks that either succeed or fail as a unit - atomic
- Establishing and managing a transaction boundary may be challenging
- Requires close examination of code
- All related objects must participate in it
- Need to make sure all operations fall into that one transaction
- Find a timely place to commit or rollback
- Each participant needs to vote towards commit or rollback
- Managing all this may be tedious and error prone

Transaction-less Collector

- We have small time collectors and big time collectors (what ever they collect)
- Small time collector will be interested in any quantity
- Big time collector will only accept larger quantities (1000 or more)
- Lets see how this works with ASP.NET
  - Fails transaction integrity
- How to fix it?
  - You can create a transaction object and pass it around?!

Transaction Aware Collector

- Let's make Collector Transaction aware
- Make it a ServicedComponent
- Mark the class with
  [Transaction(TransactionOption.Required)]
  - this makes state of component part of TXN
- Mark the methods with
- [AutoComplete]
  - this sets the vote to commit or abort (if an exception is thrown)
  - Alternately you may use ContextUtil.SetComplete() or SetAbort()

Problem with AutoComplete

- If no exception is thrown, ContextUtil.SetComplete() is called
- If exception is thrown, ContextUtil.SetAbort() is called
- Sounds good, what's your problem?
- Both these methods have a side effect
- Also sets ContextUtil.DeactivateOnReturn = true
- So, your component is gone upon return!
- What's the fix?
  - Do not use AutoComplete or SetComplete/SetAbort
  - set *ContextUtil.MyTransactionVote* to Abort on entry
  - If success, set **ContextUtil.MyTransactionVote to commit**

Transaction Options

- Disabled
  - Is not transaction aware. Ignores TXN in current context
- NotSupported
  - Component is created in a context with no governing TXN
- Required
  - If caller is in a TXN, participates in it. Otherwise, creates a new one
- RequiresNew
  - Component created with its own TXN
- Supported
  - Participates in a TXN of caller if one exists

.NET Enterprise Services

- Why and What?
- Object Pooling
- JIT Activation
- Queued Components
- Transactions
- **Controlling Object Creation**
- Managing DB Access
- V 1.1 Enhancements
- Conclusion

Controlling the use of Component

- In my original code, I had the constructor of Collector as protected internal
- This make it hard for others to create object of my class
- However, EnterpriseServices requires that I have a public default constructor!
- What if in the group of developers, some one tries to create an object of my class instead of using the factory?
- A simple trick comes in handy

Obsolete Attribute!

Mark constructor as obsolete

```
[Obsolete("Please use load method instead", true)]
public Collector() {}
```

…

To create an object within my project

```
ctr =
 Activator.CreateInstance(typeof(Collector))
   as Collector;
```

…

If users of my class use new to create object, they get a compilation error

.NET Enterprise Services

- Why and What?
- Object Pooling
- JIT Activation
- Queued Components
- Transactions
- Controlling Object Creation
- **Managing DB Access**
- V 1.1 Enhancements
- Conclusion

Optimizing the Save

- Notice that Collector's save is not exposed.
- However, each time sell or buy is called, DB is accessed
- How about reducing the number of calls to DB?

- We may keep a dirty flag
- Upon call to Deactivate, we could save automatically

.NET Enterprise Services

- Why and What?
- Object Pooling
- JIT Activation
- Queued Components
- Transactions
- Controlling Object Creation
- Managing DB Access
- **V 1.1 Enhancements**
- Conclusion

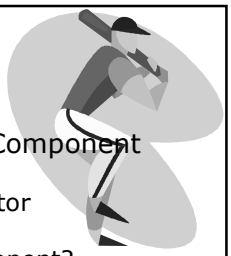
Down side of Component Model

- .NET allows single implementation inheritance
- If you want your component to support Enterprise Services, you must inherit from ServicedComponent
- What if you want your component to inherit from another class?
- Transaction is simply an aspect (AOP)
- It is a cross cutting concern
- So, why not make it just that?
- This is where **ServiceDomain** class comes in

ServiceDomain

- Supported only on Windows 2003
- Has Enter and Leave methods
- Call to Enter creates a new context and pushes it onto Context Stack
- Leave emulates the behavior of returning from a call to another context
- More efficient, less overhead since no thread marshalling is involved
- ServiceConfig class is used to provide context specific information

Quiz Time



1. Three reasons to use Enterprise Services?
   Scalability, Rapid development, Efficient
2. Things you should follow in developing a Serviced Component
   public default constructor, CLI compliant
3. How can you make sure no one uses this public constructor
   Mark it as obsolete
4. How do you express the transaction needs of your component?
   [Transaction(TransactionOption.Required)]
5. What attribute helps with a method voting automatically based on success
   [AutoComplete]
6. What is the side effect of calling SetAbort or SetCompleted
   deactivates the component
7. What class allows you to voice your components transaction vote
   ContextUtil
8. What property of ContextUtil is better to use than AutoComplete
   MyTransactionVote
9. What method you may depend upon to persist your data
   Deactivate
10. What 1.1 classes eliminate the need for inheriting from ServicedComponent
    ServiceDomain and ServiceConfig

.NET Enterprise Services

- Why and What?
- Object Pooling
- JIT Activation
- Queued Components
- Transactions
- Managing DB Access
- Controlling Object Creation
- V 1.1 Enhancements
- **Conclusion**

Conclusion

- Enterprise Services will help us
  - develop faster
  - more robust and scalable
  - lesser code to write and test
- Has some challenges
- Requires a bit more discipline in developing the components
- A direct .NET access to COM+
- Consider it if you manage your own TXN or other facilities provided

References

- MSDN Documentation
- Examples, slides are for your download at

http://www.agiledeveloper.com/download.aspx

Thanks!