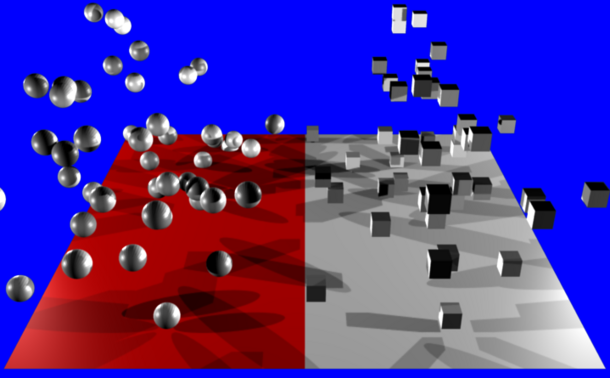
**Instancing and Concurrency in WCF**

****

**Rameshkartik. RS**

**Table of Contents**

[What is Concurrency? 2](#_Toc388879598)

[Difference between Instancing and Concurrency 5](#_Toc388879599)

[Instancing with the Concurrency 5](#_Toc388879600)

[Summary 18](#_Toc388879601)

**WHAT’s IN THIS ARTicle?**

* What is Concurrency?
* Difference between the instancing and the concurrency?
* Instancing with Concurrency
* Attachment
* Summary

# 

# What is Concurrency?

“ *Several computations are executing simultaneously*.”

I just try to explain you about the concurrency with the simple day to day activity. Assume that you are going for a shopping to get some provisional goods for your home,If you find only one shopkeeper in that shop obviously you must be in a queue in order to get your requirements done. Yes, the only shopkeeper serves the request one after the other. If the same shop has multiple shopkeepers,they will manage multiple buyers at the same time ,in result requests are processed very quickly.If we relate the same activity with the WCF I could say that, By forming multiple threads in the server instance, more requests from the client proxy would be addressed in a concurrent fashion. We can use the concurrency feature in three different ways,

* Single
* Multiple
* Reentrant

Single

Single request will be processed by a single thread on a server at any point of time, Client proxy sends request to the server,it process the request and takes the another request.At a time one request will be processed on the server side. Other requests have to wait until the request processed by the service is completed. Following is the declaration of Concurrency.Single

Con_Single.png

Multiple

Multiple requests will be processed by multiple threads on the server at any point of time, Client proxy sends multiple requests to the server, Requests are processed by the server by spawning multiple threads on the server object. Doing this you will get more throughput here. Following is the declaration of Concurrency.MultipleMultiple.png

Reentrant

The reentrant concurrency mode is almost like single concurrency mode. It is a single threaded service instance receives request from the client proxy and it unlocks the thread only after the reentrant service object calls the other service or can also call a WCF client through call back. Reentrant service calls the other service or call back so another client requests would be allowed. Following is the declaration of Concurrency.Reentrant

Conn_Rentrant.png

# Difference between Instancing and Concurrency

* Instancing is about how the objects are created in the server. I.e Single or multiple objects are created on the server side
* Concurrency is about how the requests are handled by the WCF object i.e Single or multiple threads are created on the server side to process the client requests

# Instancing with the Concurrency

Though WCF provides lots of facilities to build a distributed system, to write efficient services the developer has to think about the answer for following questions

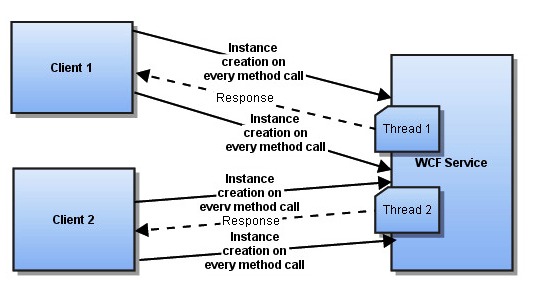
* In what frequent the service instances are to be created?
* How long the created instance,should exist?
* How about the service going to manage the incoming client requests? What level of concurrency (Single or Multiple) it is needed?
* Does your service require to maintain the state between calls?

Ok, lets discuss about the 9 different combinations of instancing and the concurrency.

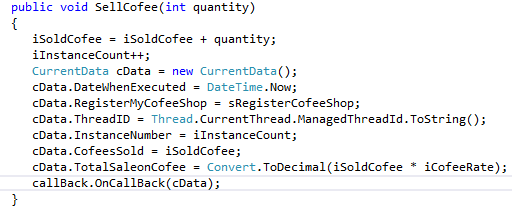
Instance Mode -> Per Call and Concurrency -> Single

*“Single thread for every client”*

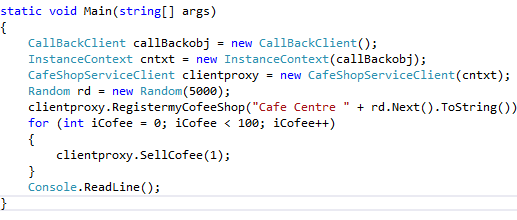
With the instance mode percall, new instances will be created for every service method call, With the concurrency mode is Single, one thread will be created in the server to manage all the requests from one client.



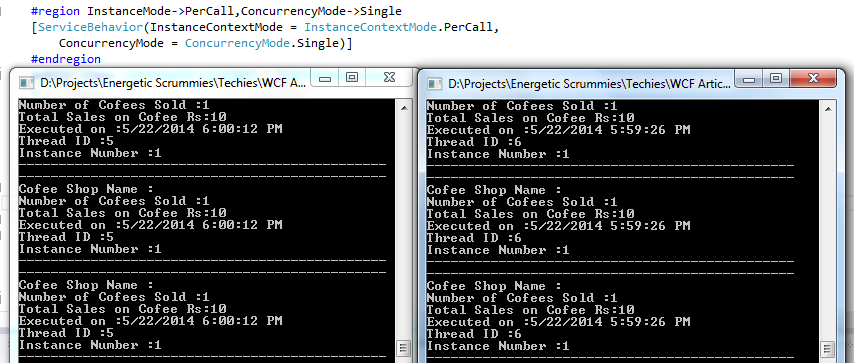
In the above diagram you can find that server instance will be created on every method call that client performs,The client will perform a method call by creating a new service instance, execute the service method and it release the memory. One thread sitting in the server for each client manages the request in one after the another.Lets discuss the same concept with the source code.



Above one is the exposed service method which actually maintains the number of coffees selling,Total sale made on coffee, For our demonstration purpose am just included the Thread Id to see how many threads actually serving the clients request.Here is the client portion where it triggers the server method SellCofee .



In the client am just trying to sell 100 cofees at a moment.With this example we will see how the instances are created? And how it manages the client requests.

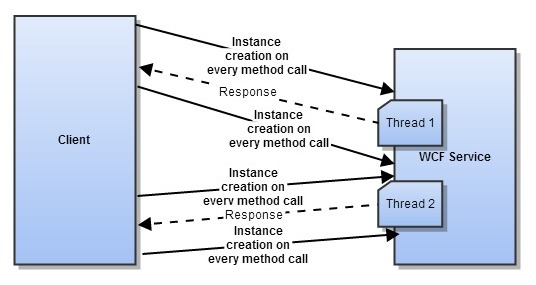


If you see the above snapshot you can find that two client applications were launched.As I told you earlier the instance will be creating on each and every server method call invocation.In our case Instance number will always displaying the value 1 across the clients, because of the existing information will be lost on new instance creation. Since the concurrency mode is single, you can see in the above picture that each client request will be managed by single thread(Thread Id 5 for Client 1) and(Thread Id 6 for Client 2). For further details please refer the source code attachment.

Instance Mode -> Per Call and Concurrency -> Multiple

*“Multiple Threads for every client”*

With the instance mode percall, new instances will be created for every service method call, With the concurrency mode is Multiple, Multiple threads will be created in the server to manage all the requests from each client.

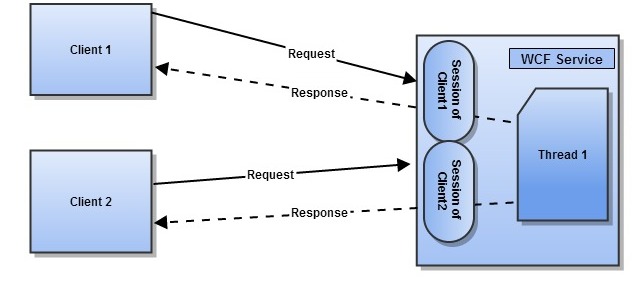
**

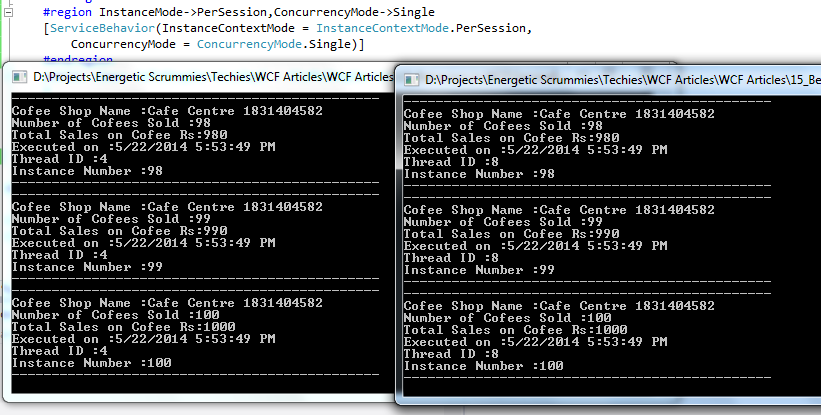
In the above diagram you can find that server instance will be created on every method call that client performs,The client will perform a method call by creating a new service instance, execute the service method and it’s release the memory. Multiple threads sitting in the server for each client to manage the request concurrently.Let’s discuss the same concept with the source code. For further details please refer the attached source code

Instance Mode -> Per Session and Concurrency -> Single

*“Single Thread for Every Client”*

With the Instance Mode is PerSession,instance will be created for every client session,With the concurrency mode is Single, one thread will be created in the server to manage all the requests from one client.

**

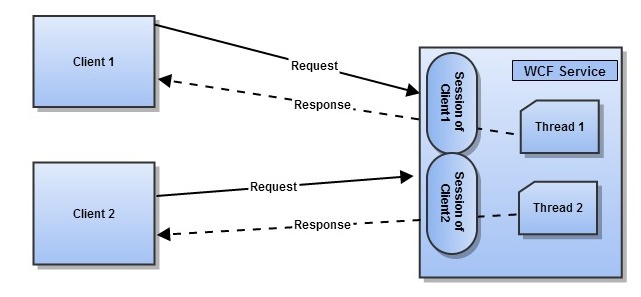


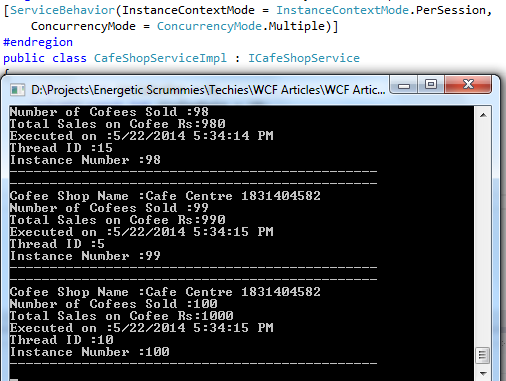
If you see the above snapshot, you can find that one server instance will be created for every client session. Instance number is incremented and it is maintained for one client. If you look the instance number in the another client again incremented starting from the 0. Threads are created in server for every client session. Thread number 4 created for the client 1, and the thread number 8 are created for the client 2. For further details please refer the source code attachment.

Instance Mode -> Per Session and Concurrency -> Multiple

*“Multiple threads for every request”*

With the Instance Mode is PerSession, instance will be created for every client session , With the concurrency mode is multiple, multiple threads will be created on the server to manage all the requests from one client call.

**

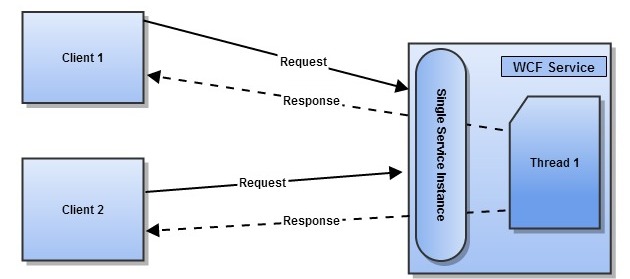
**

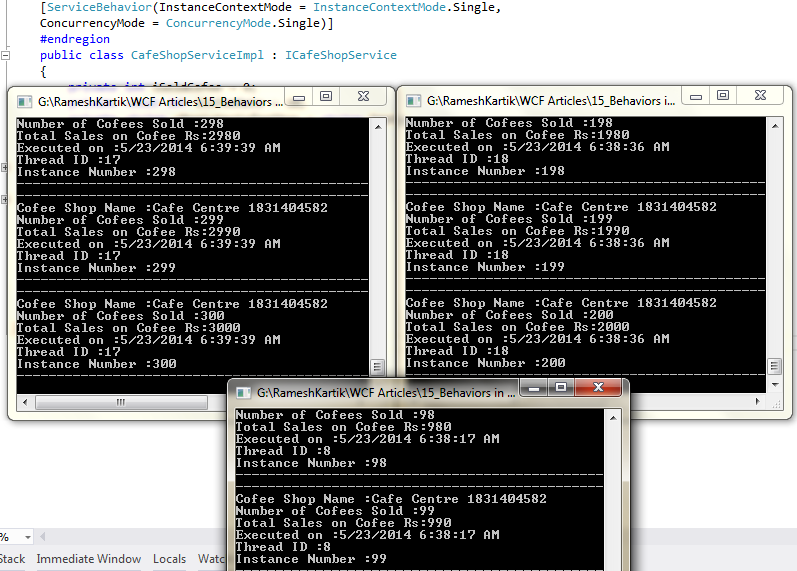
In the above snapshot you can find that multiple threads are created for every client session.On every client session the server spawns multiple threads to process each request of particular client. In our case Thread number 5,10 and 15 are created in the server side. For further details please refer the source code attachment.

Instance Mode -> Single and Concurrency -> Single

*“Single Thread for all Clients”*

With the instance mode is single, The single server instance will be created in the server side,using the same and the only instance, all the clients will be communicated.Here the concurrencymode is also single, so the single thread in the server manages all the requests from all the clients. Comparing all other combinations this one might be relatively slow,because the only thread available to process the requests from all the clients.



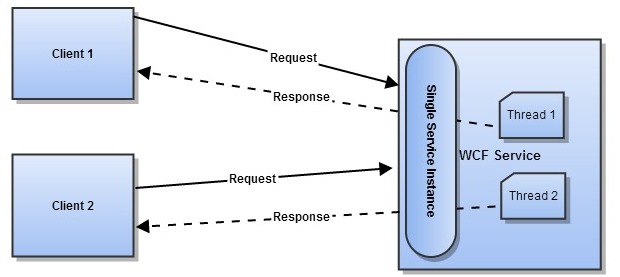


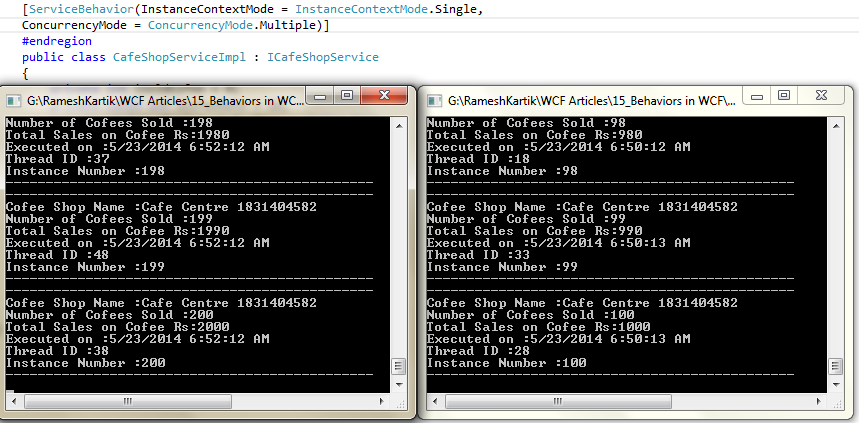
Since the same server instance are using across clients you can see the information(Instance number) are maintainted in the server. Also single thread is dedicated for each client. Thread number 17 is managing the client 1 requests, where as the Thread 18 is managing the client 2 requests, and Thread 8 is managing the client 3 requests. For further details please refer the source code attachment.

Instance Mode -> Single and Concurrency -> Multiple

*“Multiple Threads for all the clients”*

With the instance mode is single, The single server instance will be created in the server side,using the same and the only instance, all the clients will be communicated.Here the concurrencymode is Multiple, so the Multiple threads in the server manages all the requests.



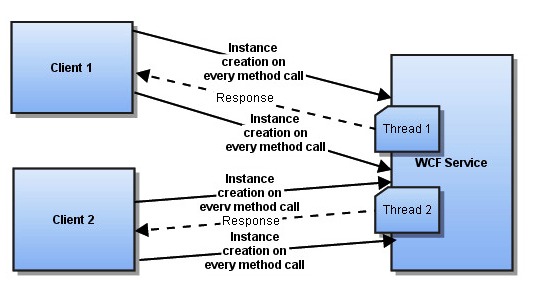


Since the same server instance are using across clients you can see the information(Instance number) are maintainted in the server. Also Multiple threads are created for each client request. In the above snapshot you can see that different threads are created for each request. For further details please refer the source code attachment.

Reentrant

Only one thread in server to serve all requests. If your WCF service makes an outbound call to other WCF service, or makes a client call, it releases the thread lock. In other words, until the outbound call is completed, other WCF clients can not make calls

*CHANGE*



# Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Instance ContextMode** | **Concurrency Mode** | | |
| **Single** | **Multiple** | **Reentrant** |
| Single | Single Thread for all the clients | Multiple threads for all the clients | Single Thread for all clients.Lock will be released when the outbound call to other services are completed |
| PerSession | Single Thread for every client | Multiple threads for every request | Single Thread for all clients.Lock will be released when the outbound call to other services are completed |
| PerCall | Single Thread for every Client | Multiple threads for each request | Single Thread for all clients.Lock will be released when the outbound call to other services are completed |