Instancing and Concurrency

Miguel A. Castro
@miguelcastro67

Service Instancing

- **Determines when the host will new-up a new instance of service**
  - Determines which instance services the call
- **Major difference between WCF and Web API**
  - Web API is always stateless
- **3 modes of instantiation**
  - PerCall
  - PerSession *(default)*
  - Single

PerCall Instancing

- **Each call from a proxy (in any client) is serviced by a brand-new instance of the service**
- **The host news-up a service instance, performs the operation, and disposes the instance (if IDisposable)**
- **This is the case even if the client proxy stays open**
- **Most scalable solution**
- **Nothing left open and in memory**
- **Typical in-and-out SOA calls**
- **Service CANNOT hold in-memory state**
  - Class-scoped variables

PerSession Instancing

- **Host news-up service instance on first call from proxy**
  - Or if proxy manually opened (explained in upcoming module)
- **All calls from the same proxy are serviced by same instance**
- **Service can hold in-memory state**
  - Class-scoped variables
  - If doing updates to state, must consider locking
- **When proxy is closed, service instance is disposed**
  - On a final infrastructure-call
- **Transport session must be present *(remember this?)***
  - TCP Binding
  - IPC Binding (named pipe)
  - WS-Http Binding with Reliability or Security turned ON
  - Otherwise downgrade to per-call

Single Instancing

- **Host news-up service instance when opened**
- **All calls from proxies, all clients, all countries, all planets, are serviced by same instance**
- **Service can hold in-memory state**
  - Class-scoped variables
  - If doing updates to state, must consider locking
- **When host is closed, service instance is disposed**
- **Host can be tied to an existing service instance**
  - Manually newed-up
  - Good for pre-hosting initialization

Setting Instance Mode

```
[ServiceBehavior(InstanceContextMode=
       InstanceContextMode.{enum-value})]
public class MyService : IMyService
{
   …

enum-value:
   PerCall
   PerSession
   Single
```

Demo Time

Demarcation

- **Service Contract can control who can start/end session**
- **IsInitiating / IsTerminating properties**
  - OperationContract attribute
- **IsInitiating defaults to true**
- **IsTerminating defaults to false**
- **Controls the creation and termination of transport session**

Demo Time

Session Mode

- **Configures the requirement-mode of a Transport Session**
  - Remember Transport Session?
  - TCP, IPC, WS with Reliability or Security
  - No Basic HTTP
- **Allowed (default)**
  - Allows one but does not require it
  - No error thrown
  - **PerSession** services will behave like **PerCall** if no Transport Session
- **Required**
  - Requires a Transport Session binding setup
- **NotAllowed**
  - Does not allow a Transport Session binding setup

Demo Time

Concurrency

- **Determines how a service handles locking during multiple, concurrent calls**
- **Three modes**
  - Single
  - Multiple
  - Reentrant

Single Concurrency

- **Service instance allows one call in at a time from each proxy**
  - Remember this is per-instance – instance mode is relevant
- **PerCall**
  - Each call serviced by new instance
  - Client can use same proxy on multiple threads (still new instances)
  - Service will still only allow one call at a time
- **PerSession**
  - Each proxy serviced by new instance
  - Client can make proxy calls on multiple threads (same instance)
  - Service will allow only one call at a time
- **Single**
  - Every call (from everywhere) serviced by same instance
  - Service will allow only one call at a time

Multiple Concurrency

- **WCF stays out of the way and provides NO locking**
- **Depending on scenario, you must provide your own locking**
- **PerCall**
  - Each call serviced by new instance
  - Client can use same proxy on multiple threads (still new instances)
  - All calls processed concurrently (but still new instance, no locking needed)
- **PerSession**
  - Each proxy serviced by new instance
  - Client can make proxy calls on multiple threads (same instance)
  - All calls allowed in (even concurrently) so locking rules apply
- **Single**
  - Every call (from everywhere) serviced by same instance
  - All calls allowed in (even from multiple clients) so locking rules apply

Reentrant Concurrency

- **Similar to Single Concurrency**
- **WCF disregards lock when client is same one that placed it originally**
  - I know, confusing
- **Necessary only in case of callbacks**
  - Covered in full later in course

Demo Time