Из цикла лекций **«Internet-технологии разработки приложений»** для студентов 4-го курса кафедры Компьютерных технологий физического факультета Донецкого национального университета
проф. В. К. Толстых

WCF-службы:
Надёжность,
Управление экземплярами (сеансы)

проф. В.К.Толстых,
www.tolstykh.com

Надёжность транспорта и сообщений

Надёжность транспорта обеспечивает гарантированную доставку на уровне сетевых *пакетов*, включая их последовательность. Естественно, надёжность транспорта не срабатывает при разрывах коммуникаций.

Надёжность сообщений обеспечивает гарантированную доставку* и порядок *сообщений*, не зависимо от количества пакетов, посредников и сетевых переходов (хопов). Сообщения могут обрабатываться в порядке их отправки (включено по умолчанию) или в порядке получения. В WCF надежность сообщений настраивается на уровне привязки:

Привязка	Поддержка надёжности	Состояние по умолч.	Поддержка упорядоченной доставки	Состояние по умолч.
BasicHttpBinding	Нет		Нет	
NetTcpBinding	Да	Выкл	Да	Вкл
NetPeerTcpBinding	Нет		Нет	
NetNamedPipeBinding	Нет		Нет	
wsHttpBinding	Да	Выкл	Да	Вкл
wsDualHttpBinding	Да	Вкл	Да	Вкл

* - В случае неудачной передачи сообщений организуется повторная передача (по умолчанию – 8 попыток). Для определения времени повторной передачи использует алгоритм с экспоненциальной задержкой. Задержка перед первой попыткой повторной передачи составляет 1 секунду, для каждой последующей попытки время задержки удваивается. Таким образом, временной интервал 8 попыток составляет около 8,5 минут.

Настройка надёжности связи

Включение надёжности должно осуществляться как на стороне службы, так и на стороне клиента. Пример включения надёжности в файле конфигурации для привязок TCP:

```
<endpoint address="net.tcp://localhost:8000/MyService/"
    binding="NetTcpBinding"
    contract="IMyContract"
        bindingConfiguration="MyNetTCP" >
        ...
<bindings>
   <netTcpBinding>
      <binding name="MyNetTCP">
       <reliableSession enabled = "true"  ordered="true" />
      </binding name="MyNetTCP">
   </netTcpBinding>
</bindings>
```

Уточнение настроек для привязок типа **netTcpBinding**

Настройка привязки конечной точки с bindingConfiguration="**MyNetTCP**"

Включение надёжности

Упорядочение доставки (по умолчанию – true)

Установить упорядоченную доставку, например, для всех конечных точек службы можно и при помощи атрибута у контракта службы:

```
[DeliveryRequirements(RequireOrderedDelivery = true)]
Class MyService: IMyContract
{...}
```

Управление экземплярами

При подключении клиента к службе, у клиента создаётся необходимый экземпляр посредника (прокси), который организует канал связи и подключение к службе, а на стороне службы создаётся экземпляр службы для обработки запросов клиента. Далее клиент по каналам посредника делает запрос к операциям экземпляра службы.

Для каждого запроса к операциям службы может создаваться отдельный экземпляр службы. В этом случае все операции в службе оказываются независимыми, если искусственно не поддерживается состояние операций, например, при помощи БД.

Службы уровня вызова – PerCall: для каждого клиентского запроса создаётся (и в последствии уничтожается) новый экземпляр службы.

Службы уровня сеанса – PerSession: создаётся один экземпляр службы* для каждого подключения одного и того же клиента. Принято по умолчанию.

Синглетные службы – Single: все подключения клиентов обслуживаются одним экземпляром службы.

Отдавайте предпочтение службам уровня вызова. Избегайте сеансовых и тем более синглетных служб. Используйте упорядоченную доставку в сеансовых службах.

*- в классических Web-сервисах создаётся сеанс с методом, а не со всем сервисом, что не позволяет создавать один сеанс для всех методов.

Службы уровня вызова

Настроить службу как службу уровня вызова можно при помощи атрибута поведения у контракта службы:

```
[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerCall)]
Class MyService: IMyContract
{...}
```

Теперь для каждого вызова *одного и того же метода* (операции) клиента будет создаваться новый экземпляр службы при одном и том же подключении посредника. Клиент не знает, что он каждый раз имеет дело *с новым экземпляром службы*. При необходимости, такие службы должны быть службами *с контролем состояния*.

Преимущества:

1. Объект выделяется клиенту только на время обработки вызова.
2. Многократное создание и уничтожение экземпляра службы на стороне службы *без разрыва связи с клиентом* (с посредником клиента) обходится гораздо дешевле, чем в классической клиент-серверной модели (создание экземпляра + подключение).
3. Службы уровня вызова хорошо масштабируются. При проектировании таких служб рекомендуется делать 10-кратный запас по нагрузке «на будущее».
4. Службы уровня вызова хорошо укладываются в транзакционную модель программирования независимо от системной нагрузки.

Службы уровня сеанса

Сеансы WCF являются «клиентскими» и имеют следующие основные особенности:

- Они явным образом инициируются и завершаются вызвавшим приложением, т.е. – клиентом, а не сервером как в ASP.NET.
- Клиентский сеанс определяется на уровне конкретной рабочей точки, т.е. один клиент может иметь несколько сеансов, организованных через разные точки.
- Нет общего хранилища данных, связанного с сеансом WCF в отличие от ASP.NET.

Когда клиент создаёт нового посредника для службы, которая настроена как сеансовая служба, то клиент получает новый специализированный экземпляр службы, не зависящий от всех остальных экземпляров той же службы. Экземпляр продолжает существовать до тех пор, пока не станет ненужным клиенту. При этом экземпляр службы может использоваться для хранения состояния, а модель программирования оказывается близкой к клиент-серверной модели.

По умолчанию службы поддерживают не более 10 параллельных сеансов. Служба принимает подключения новых клиентов до достижения этого числа. После этого служба отклоняет подключения новых клиентов, пока не будет закрыт один из текущих сеансов. Поддержку большего количества клиентов можно обеспечить двумя способами: не использовать сеансовую привязку; увеличить ограничение сеансов в поведении службы – MaxConcurrentSessions. Поддержка большого количества сеансов требует больших затрат ресурсов, связанных с каждым специализированным экземпляром службы.

Настройка сеансов

Возможность поддержки сеансов определяется привязкой, контрактом и поведением службы, например (принято по умолчанию):

```
[ServiceBehavior(InstanceContextMode=InstanceContextMode.PerSession)]
Class MyService: IMyContract
{...}

[ServiceContract(SessionMode=SessionMode.Allowed)]
Interface: IMyContract
{...}
```

Свойство **SessionMode.Allowed** разрешает сеансы, но не делает их обязательными,
SessionMode.Required устанавливает сеансы если они возможны,
SessionMode.NotAllowed запрещает сеансы, получаем службу уровня вызова.

Сеанс между клиентом и экземпляром службы надёжен лишь в той степени, в какой надёжен транспортный сеанс.

Привязка	Сеансовый режим со службой
BasicHttpBinding	Не поддерживается
NetTcpBinding, **NetNamedPipeBinding**	Поддерживается на транспортном уровне
wsHttpBinding, **wsDualHttpBinding**	Поддерживается логическим идентификатором сеанса в заголовке HTTP

Завершение сеанса

Сеанс открывается при первом обращении клиента к службе, а завершается при закрытии клиентом своего посредника, или при неактивности клиента в течение тайм-аута 10 мин (по умолчанию), или при неудачной передаче сообщений при включенной надёжной доставке.

Значение тайм-аута можно задавать через свойство ReliableSession соответствующей привязки. Например, для WS-привязки клиента установка тайм-аут 25 минут на программном уровне может иметь вид:

```
System.ServiceModel.WSHttpBinding MyWsBinding =
    new System.ServiceModel.WSHttpBinding();
MyWsBinding.ReliableSession.Enabled = true;
MyWsBinding.ReliableSession.InactivityTimeout =
    TimeSpan.FromMinutes(25);
```

Или в конфигурационном файле при включенной надёжной доставке:

```
<wsHttpBinding>
   <binding name="..." />
     <reliableSession enabled="true" inactivityTimeout="00:25:00" />
```

Если клиент или служба задают разные тайм-ауты, то используется более короткий интервал

Согласование привязки, контракта и поведения службы

Привязка	Контракт	Поведение	Режим управления экземплярами
Basic	Allowed/NotAllowed	PerCall/PerSession	PerCall
TCP, **IPC**	Allowed/Required	PerCall	PerCall
TCP, **IPC**	Allowed/Required	PerSession	PerSession
WS (без надёжности и безопасности)	NotAllowed/Allowed	PerCall/PerSession	PerCall
WS (с безопасностью или надёжностью)	Allowed/Required	PerSession	PerSession
WS (с безопасностью или надёжностью)	NotAllowed	PerCall/PerSession	PerCall

Синглетные службы

Все клиенты независимо друг от друга подключаются к единственному экземпляру службы, обслуживающему все конечные точки. Срок существования синглетной службы не ограничен, и она уничтожается только при завершении хоста. Единственный экземпляр синглетной службы создаётся ровно один раз – при создании хоста.

Чтобы установить службу на синглетный режим необходимо сделать следующие настройки поведения:

```
[ServiceBehavior(InstanceContextMode=InstanceContextMode.Single)]
Class MyService: IMyContract
{...}
```

Синглетные службы плохо масштабируются. В любой момент времени с синглетом может работать только один клиент. Такое ограничение снижает производитель-ность, скорость отклика и доступа. Например, если операция с синглетом занимает 0.1 секунды, служба сможет обслуживать только 10 клиентов в секунду. При большем количестве клиентов (20 или 100) производительность системы становится неприемлемой.

Регулирование нагрузки

WCF позволяет регулировать нагрузку службы в её поведении. Например:

```
<behaviors>
      <serviceBehaviors>
        <behavior name="…">
           <serviceThrottling
             maxConcurrentCalls=" 12"
             maxConcurrentSessions="34"
             maxConcurrentInstances="56" />
        </behavior>
      </serviceBehaviors>
      <endpointBehaviors>
         <behavior name="PointBehavior">
    …
         </behavior>
      </endpointBehaviors>
</behaviors>
```

Максимальное число вызовов, обрабатываемых в данный момент всеми экземплярами служб уровня вызова (по умолчанию - 10).

Максимальное количество клиентов, использующих сеансы (по умолчанию - 16)

Максимальное число одновременно существующих экземпляров. Для служб уровня сеанса оно совпадает с **maxConcurrentSessions**. Для служб уровня вызова – берётся минимальное из указанного и **maxConcurrentCalls**.
Для синглетных служб - игнорируется

Источники

- Джувел Лёве. Создание служб Windows Communication Foundation. – СПб.: Питер, 2008 . – 592 с.: ил.

- http://msdn.microsoft.com