**каналы**

Канал - отвечает за подготовку и доставку сообщений еди нообразным способом

Стек каналов представляет собой последовательность каналов, сконфигури рованных с помощью элементов привязки.

Назначение стека каналов состоит в том, чтобы преобразовать передаваемое по сети сообщение в формат, совместимый с получателем и отправителем

Транспортный канал всегда находится внизу стека и отвечает за транспортировку сообщений в соответствии с транспортным про токолом. WCF предоставляет широкое разнообразие транспортных каналов, включая HTTP, TCP, MSMQ, пиринговые (peertopeer) и именованные каналы (named pipe). Протокольные каналы располагаются поверх транспортных и про чих каналов; они трансформируют и модифицируют сообщения. WCF поддержи вает много разных протокольных каналов, например, для обеспечения безопасно сти, транзакционной целостности и надежной передачи.

Фабрика каналов создает канал для отправки сообщений и становится владельцем созданных каналов.

**привязки**

Привязкой называется заранее сконфигурированный стек каналов. Привязки описывают соглашения между клиентом и сервером о порядке передачи данных по сети. В привязке задается способ транспортировки, кодирование и протоколы, участвующие в коммуникации.

Привязка описывает конфигурацию стека и знает о том, как создать его на этапе выполнения. Каждая привязка составляется из набора элементов привязки, которые обычно представляют отдельные каналы в стеке.

**поведения**

Поведения – это классы, которые влияют на работу WCF на этапе выполнения. Они вызываются, когда инициализируется исполняющая среда WCF на стороне клиента или сервера, а также в процессе передачи сообщений между ними.

Примеры: ConcurrencyMode.Single, Reentrant, Multiple и

InstanceContextMode.PerSession, Single, PerCall

IServiceBehavior, IEndpointBehavior, IOperationBehavior или IContractBehavior.

<behaviors>

<serviceBehaviors>

<behavior name="throttling">

<serviceThrottling maxConcurrentSessions="5"/>

</behavior>

</serviceBehaviors>

</behaviors>

Область действия поведений оконечной точки ограничена одной оконечной точкой службы.

На стороне клиента поведения могут выполнять три функции:

 Инспекция параметров. Осмотреть и/или изменить данные в .NETпред ставлении перед тем, как они преобразуются в формат XML.  Форматирование сообщения. Осмотреть и/или изменить данные в про цессе преобразования из типа .NET в XML.

 Инспекция сообщения. Осмотреть и/или изменить данные в XMLпред ставлении перед тем, как преобразовать его в типы .NET.

На стороне сервера поведения позволяют реализовать два дополнительных сценария:

 Выбор операции. На уровне службы можно проинспектировать входящее сообщение и решить, какую операцию вызывать.

 Вызов операции. На уровне операции вызвать тот или иной метод класса

Transaction behavior:

[ServiceContract(SessionMode=SessionMode.Allowed)]

[OperationBehavior (TransactionScopeRequired=true)] - В результате перед тем, как передать управление вызванному методу, WCF создаст новую транзакцию и включит в нее поток, в котором выполняется операция. Если в ходе выполнения операции возникнет ошибка, все частичные обновления транзакционных ресурсов будут откачены.

TransactionScopeRequired=false (это умолчание), то операция выполняется без заведения транзакции и, следовательно, не поддерживает свойства ACID. Если такая операция обновит одну таблицу, а при обновлении другой возникнет ошибка, то первая таблица останется обновленной, то есть свойство атомарности нарушается

[OperationBehavior(TransactionAutoComplete=true)], то операция неявно считается завершенной, если она не сообщила об ошибке.

[OperationBehavior (TransactionAutoComplete =false)] и перед возвратом управления явно вызвать метод OperationContext.Current.SetTransactionComplete()

TransactionFlowOption. Allowed, позволяющий транзакции пересекать границы вызовов(нескольких сервисов). Так же клиенту тога надо использовать TransactionScope  
+ <binding name="transactions"

transactionFlow="true" >

</binding>

**Заметки**

Конттракты служб, контракты данных , контракты сообщений

ограничения пропускной способности (throttling)

В основе своей служба – это множество оконечных точек (endpoints)

На самом нижнем уровне элемент привязки – это транспортный механизм, обеспечивающий доставку сооб щений по сети. В WCF встроены следующие транспорты: HTTP, TCP, Named Pipes, PerChannel и MSMQ.

Контракт определяет набор функ ций, предоставляемых оконечной точкой, то есть операции, которые она может выполнять, и форматы сообщений для этих операций.

В случае дуплек сного контракта клиент не становится явной WCFслужбой и не может свободно выбирать привязки или объявлять свои оконечные точки. Адрес, привязка и кон тракт, определяющие клиентскую оконечную точку, назначаются фабрикой каналов в момент инициирования дуплексного сеанса связи клиентом.

public void DivideBy(double n)

{

result = result / n;

Callback.Result(result);

}

//...

ICalculatorDuplexCallback Callback

{

get

{

return OperationContext.Current.GetCallbackChannel<ICalculatorDuplexCallback>();

}

}