**Web-services**

**Распределенная система** - система, в которой обработка информации сосредоточена не на одной вычислительной машине, а распределена между несколькими компьютерами.

**Существует шесть основных характеристик распределенных систем.**

1. **Совместное использование ресурсов***.* Распределенные системы допускают совместное использование как аппаратных (жестких дисков, принтеров), так и программных (файлов, компиляторов) ресурсов.
2. **Открытость.**Это возможность расширения системы путем добавления новых ресурсов.
3. **Параллельность***.*В распределенных системах несколько процессов могут одновременно выполнятся на разных компьютерах в сети. Эти процессы могут взаимодействовать во время их выполнения.
4. **Масштабируемость.**Под масштабируемостью понимается возможность добавления новых свойств и методов.
5. **Отказоустойчивость.**Наличие нескольких компьютеров позволяет дублирование информации и устойчивость к некоторым аппаратным и программным ошибкам. Распределенные системы в случае ошибки могут поддерживать частичную функциональность. Полный сбой в работе системы происходит только при сетевых ошибках.
6. **Прозрачность.**Пользователям предоставляется полный доступ к ресурсам в системе, в то же время от них скрыта информация о распределении ресурсов по системе.

**Распределенные системы обладают и рядом недостатков:**

1. **Сложность.** Намного труднее понять и оценить свойства распределенных систем в целом, их сложнее проектировать, тестировать и обслуживать. Также производительность системы зависит от скорости работы сети, а не отдельных процессоров. Перераспределение ресурсов может существенно изменить скорость работы системы.
2. **Безопасность.** Обычно доступ к системе можно получить с нескольких разных машин, сообщения в сети могут просматриваться и перехватываться. Поэтому в распределенной системе намного труднее поддерживать безопасность.
3. **Управляемость**. Система может состоять из разнотипных компьютеров, на которых могут быть установлены различные версии операционных систем. Ошибки на одной машине могут распространиться непредсказуемым образом на другие машины.
4. **Непредсказуемость**. Реакция распределенных систем на некоторые события непредсказуема и зависит от полной загрузки системы, ее организации и сетевой нагрузки. Так как эти параметры могут постоянно изменятся, поэтому время ответа на запрос может существенно отличаться от времени подачи запроса

**Выделяется три типа архитектур распределенных систем**.

1. **Архитектура клиент/сервер.** В этой модели систему можно представить как набор сервисов, предоставляемых серверами клиентам. В таких системах серверы и клиенты значительно отличаются друг от друга.
2. **Трехзвенная архитектура**. В этой модели сервер предоставляет клиентам сервисы не напрямую, а посредством сервера бизнес-логики.
3. **Архитектура распределенных объектов.** В этом случае между серверами и клиентами нет различий и систему можно представить как набор взаимодействующих объектов, местоположение которых не имеет особого значения. Между поставщиком сервисов и их пользователями не существует различий.

**Эта архитектура широко применяется в настоящее время и носит также название архитектуры веб-сервисов.**

**Веб-сервис** - это приложение, доступное через Internet и предоставляющее некоторые услуги, форма которых не зависит от поставщика (так как используется универсальный формат данных - XML) и платформы функционирования. В данное время существует три различные технологии, поддерживающие концепцию распределенных объектных систем. Это технологии EJB, CORBA и DCOM.

Преимущества и недостатки веб-сервисов К преимуществам веб-сервисов можно отнести следующее: • веб-сервисы позволяют компании интегрировать свои бизнеспроцессы с бизнес-процессами бизнес-партнеров и клиентов при меньшей стоимости, чем с использованием других интеграционных технологий. Стоимость подобных решений на основе веб-сервисов доступна даже для SMB (Small and Medium Business), что откроет для таких компаний новые перспективы развития; • поскольку веб-сервисы организуются в публичные реестры (UDDI-реестры, ebXML-реестры или иные), доступные заинтересованным лицам по всему миру, порог выхода компаний на новые рынки снижается, возможности же для наращивания клиентской базы, напротив, возрастают; • веб-сервисы обеспечивают преемственность в отношении уже имеющихся в информационной системе компании, т. е. можно сказать, что веб-сервисы надстраиваются над существующими ИС, но не вместо них. Таким образом, обеспечивается сохранность уже сделанных инвестиций в IT-инфраструктуру и не идет увеличения требуемых, поскольку нет необходимости в радикальных изменениях; • построение новых корпоративных решений с применением веб-сервисов реализуется быстрее и совокупно дешевле, поскольку основное внимание сосредотачивается на создании бизнес-логики решения, программирование самих веб-сервисов лишь по необходимости «обрамляет» этот процесс, не требуя больших трудозатрат 37 за счет эффективного применения повторно используемого кода и адаптированных средств разработки (IDE и SDK). К недостаткам (минусам) веб-сервисов можно отнести: • стандарты интеграции бизнес-процессов, вопросы управления транзакциями и выработка единых бизнес- и IT-политик, взаимодействующих посредством веб-сервисов компаний, находятся пока на стадии разработки корпорациями IBM, XLANG и Microsoft, и спецификации WS-Coordination и WS-Transaction – результат сотрудничества IBM, Microsoft и BEA. Очевидно, без их четкой формализации и опубликования построение ИС на основе веб-сервисов может идти лишь с переменным успехом; • динамическое использование информации бизнес-реестров веб-сервисов, вызов веб-сервисов требует решения вопросов доверительности отношений между различными бизнес-реестрами. Кроме того, есть трудности в совместном использовании бизнесреестров различных форматов (например, задача поиска определенного веб-сервиса в UDDI-реестре и ebXML-реестре требует различных подходов в силу различия XML-документов, описывающих один и тот же веб-сервис в каждом из этих реестров). Надо отметить, что есть попытки решить эту проблему созданием единого браузера реестров. Например, графическая утилита Registry Browser корпорации Sun Microsystems, реализующая набор интерфейсов JAXR (Java API for XML Registries); • добавление к функциям сервера приложений функциональности провайдера веб-сервисов в силу новизны технологий может представлять определенную трудность; • вопросы безопасности функционирования информационной системы на основе веб-сервисов пока не урегулированы до конца. Спецификация WS-Security – продукт деятельности корпораций IBM и Microsoft – в настоящее время достаточно молода, не «устоялась» и частично все еще дорабатывается. Однако в силу общности положений спецификации WS-Security уже готовится к выпуску следующий слой спецификаций, посвященных вопросам безопасности: Web Services Policy Assertions, Web Services Policy Attachments, Web Services Policy Framework, Web Services Trust, Web Services Secure Conversation, Web Services Federation. Итак, преимущества представляют собой стратегические бизнес-преимущества компаний, а недостатки имеют технологический характер и обусловлены новизной технологий, решение этих проблем лишь вопрос времени. 38 Определение Веб-сервиса Сервисом (service) называется ресурс, реализующий бизнесфункцию, обладающий следующими свойствами: • является повторно используемым; • определяется одним или несколькими явными технологически-независимыми интерфейсами; • слабо связан с другими подобными ресурсами и может быть вызван посредством коммуникационных протоколов, обеспечивающих возможность взаимодействия ресурсов между собой. Веб-сервисом называется программная система, идентифицируемая строкой URL, чьи интерфейсы и привязки определены и описаны посредством XML. Описание этой программной системы может быть найдено другими программными системами, которые могут взаимодействовать с ней согласно этому описанию посредством сообщений, основанных на XML и передаваемых с помощью интернет-протоколов.

Достоинства

1. Взаимодействие между разными технологиями(.net & java)
2. Отказоустойчивость
3. Масштабирование ???
4. Любое изменение в монолитной системе приводит к пересборке и развертыванию новой версии серверной части приложения – Здесь – нет .
5. Изменение логики одного модуля не влияет на другие модули.

WCF

**каналы**

Канал - отвечает за подготовку и доставку сообщений единообразным способом

Стек каналов представляет собой последовательность каналов, сконфигурированных с помощью элементов привязки.

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

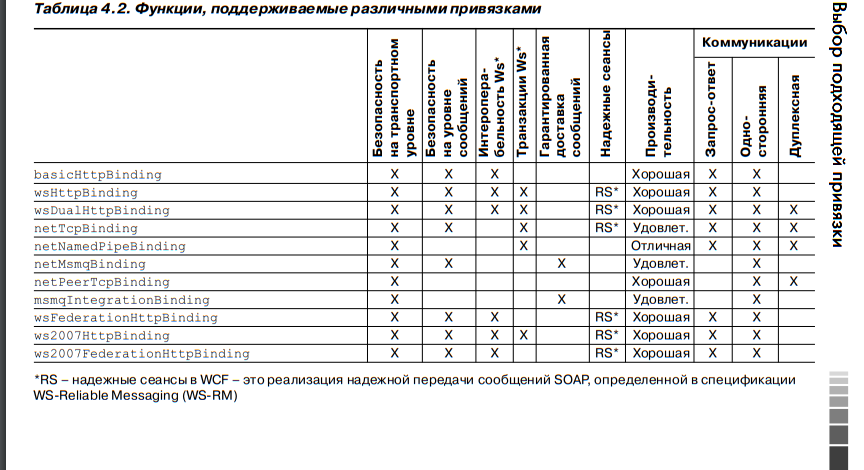
Транспортный канал всегда находится внизу стека и отвечает за транспортировку сообщений в соответствии с транспортным протоколом. WCF предоставляет широкое разнообразие транспортных каналов, включая HTTP, TCP, MSMQ, пиринговые (peertopeer) и именованные каналы (named pipe). Протокольные каналы располагаются поверх транспортных и про чих каналов; они трансформируют и модифицируют сообщения. WCF поддерживает много разных протокольных каналов, например, для обеспечения безопасности, 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 .NET представлении перед тем, как они преобразуются в формат XML.  Форматирование сообщения. Осмотреть и/или изменить данные в процессе преобразования из типа .NET в XML.

 Инспекция сообщения. Осмотреть и/или изменить данные в XML представлении 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>

Контракты

-о службе(методы класса .NET на описания служб, типов портов и операций на языке WSDL.)-

-о данных(типы CLR на определения на языке XML Schema Definitions (XSD) и определя ет, как их следует сериализовывать и десериализовывать.)

- о сообщениях (отображает типы CLR на сообщения протокола SOAP и описывает формат последних, что находит отражение в определе ниях сообщений на языках WSDL и XSD)

-об отказах

**Заметки**

Контракты 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 сообщений по сети. В WCF встроены следующие транспорты: HTTP, TCP, Named Pipes, PerChannel и MSMQ.

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

В случае дуплексного контракта клиент не становится явной WCFслужбой и не может свободно выбирать привязки или объявлять свои оконечные точки. Адрес, привязка и контракт, 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>();

}

}

хотя в описании оконечной точки разрешается указать лишь один контракт, с помощью агрегирования интерфейсов единственный контракт может раскрывать несколько интерфейсов. Кроме того, по одному и тому же адресу могут располагаться несколько оконечных точек с одинаковой привязкой, но разными контрактами; тем самым создается иллюзия, будто единственная оконечная точка реализует более одного контракта. Объявляя для нескольких оконечных точек службы один и тот же контракт, вы можете сделать его доступным через разные привязки.

типы, помеченные атрибутами [DataContract] и [DataMember] - Если класс не снабжен атрибутом [DataContract], то он и не будет включен в WSDL описание. [DataMember] - включить в XSDсхему.

**Exceptions**

**При простой обработке исключений информация об самом исключении не посылается и отказ посылается в канал + сеанс уничтожается и при следующем обращении получаем** CommunicationObjectFaultedException которое мы можем проверить \_serviceProxy.State == CommunicationState.Faulted.

Передача информации об исключении:

[ServiceBehavior(IncludeExceptionDetailInFaults=true)] - Передача информации об исключениях

Конструктору объекта FaultException можно передать также аргументы FaultReason(for use different languages) и FaultCode(Код Sender означает ошибку в сообщении, которое отправил клиент, Код Receiver означа ет, что обработку не удалось завершить изза проблем, с которыми столкнулась служба)

[OperationContract]

[FaultContract(typeof(TrackedFault))]

public bool ApproveInvoice(int invoiceId)

catch (InvoiceNotFoundException exp)

{ TrackedFault tf = new TrackedFault( Guid.NewGuid(), invoiceId + " sometxt”, DateTime.Now);

throw new FaultException(tf)

}