**Лекція 13. Основи технології WCF**

Технологія WCF є засобом для того, щоб корінним чином замінити техніку реалізації розподіленого програмування для розробників, що використовують рішення Microsoft® .NET Framework NET. WCF об'єднує той, що існує засоби розподіленого програмування в .NET у єдину модель програмування, яка надає нові рівні функціональності і функціональної сумісності, і виправдовує всі надії користувачів в цій області . Ця стаття представляє WCF, програмування, і показує, з чого слід почати.

Як випливає з її назви, технологія WCF забезпечує використання .NET у якості підставою для розробки програмного коду, який дозволяє встановлювати зв'язки між компонентами, застосуваннями, і системами. Технологія WCF проектувалася згідно з принципами функціонування служби сервісів в .NET. Сервіс в .NET - частину коду використовуючи яку організовується обмін повідомленнями. Сервіси пасивні. Вони чекають вхідних повідомлень перед виконанням деякої дії . Клієнти - ініціатори. Клієнти відправляють сервісам повідомлення щоб, щоб запитати деяку дію.

Сервисы предоставляют возможность отправки сообщения одной или нескольким конечным точкам. Каждая оконечная точка (точки коммуникации) состоит(определяется ) из адреса, соединения, и контракта (см. Рис. **1)**. Обозначения конечной точки:

* **АПК**( адрес, привязка(соединение), контракт
* **ABC** (address, binding, contract);

Адрес конкретизирует, куда послать надо сообщение чтобы его получила определенная оконечная точка. Соединение описывает способ передачи (канал) используемый для пересылки сообщения. Контракт описывает набор состав сообщения и функций, предоставляемых сервисом конечной точки. Клиентам нужно узнать эту информацию перед тем обратиться к сервису. Описанные в контракте операции отображаются на методы класса, реализующего оконечную точку, и включают в частности типы параметров, передаваемых каждому методу и получаемых от него.



Рис 1 Сервисы и конечные точки

Сервисы могут упаковать описания конечных точек , для предоставления их распределенным клиентам , используя ( как правило) язык описания Веб- сервисов ( Services Description Language- WSDL). Клиенты могут использовать созданное описание, для того чтобы генерировать код в пределах своей среды, способный к рассылке и получению сообщений

(see **Figure 2**).

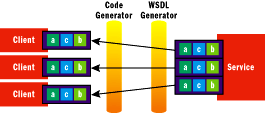


Рис 2**Разделяемое описание конечных точек (точек соединений)**

Технология WCF предоставляет новую библиотеку классов, находящихся в пространстве имен System.ServiceModel которая реализует сервисно - ориентированную модель. К этому пространству обращаются все WCF – ориентированные программы

**WCF Модель программирования**

В WCF программист является либо разработчиком сервиса предоставляющего клиентам конечные либо разработчиком клиентов, которые взаимодействую с оконечными точками. Таким образом, конечные точки являются центральным звеном модели программирования WCF и её инфраструктуры. Связь оконечных точек модели WCF с используемыми классами и интерфейсами .NET приведена в таблице.

|  |  |
| --- | --- |
| **ЭЛЕМЕНТ** | **Используемый класс или интерфейс** .NET |
| Конечная точка  Endpoint | System.ServiceModel.ServiceEndpoint |
| Адрес  Address | System.Uri |
| Соединение  Binding | System.ServiceModel.Binding |
| контракт  Contract | Interfaces annotated with System.ServiceModel attributes |

clear  Формируя сервис WCF, вы обычно начинаете с определения .NET интерфейса, чтобы сформировать контракт(набор методов) сервиса. Затем вы осуществляете имплементацию контракта сервиса в .NET классе, известном, как тип севиса, и формируете его поведение. Далее, вы определяете конечные точки, предоставляемые сервисом, конкретизируете адрес, соединение, и контракт для каждого сервиса. Наконец, вы размещаете тип службы в приложении, использующем инфраструктуры для хостинга( размещения в рабочем состоянии) WCF. Как только тип службы размещён, клиенты могут отыскать описания его оконечной точки и начать взаимодействие с ним.

Формируя клиента WCF, необходимо: во первых получить описание целевой конечной точки, к которой требуется обратиться. Описание конечной точки используется, для того чтобы динамически создать требуемый типизированный прокси. WCF предоставляет утилиту SvcUtil.exe для автоматизации этого процесса. Затем можно разработать код обращения к полученному прокси, для отсылки соответствующих сообщений целевой оконечной точке.

**Контракт службы и диспетчеризация поведения**

Для реализации WCF\_сервиса требуется написать класс на одном из языков .NET, а затем снабдить его атрибутами из пространства имен System.ServiceModel. Это пространство имен устанавливается вместе с .NET 3.0 и содержит большую часть кода WCF.

Пример полной программной реализации службы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Collections.Generic;

using System.Runtime.Serialization;

using System.ServiceModel;

namespace EssentialWCF

{

[ServiceContract]

public interface IStockService

{

[OperationContract]

double GetPrice(string ticket);

}

public class StockService:IStockService

{

public double GetPrice(string ticket)

{

Console.WriteLine("Запрос"+ ticket);

if (ticket == "tic1") { return 100.25;}

if (ticket == "tic2") { return 125.25;}

return 100;

}

}

public class Service

{

public static void Main()

{

ServiceHost serviceHost = new ServiceHost(typeof(StockService), new Uri("http://localhost:8000/EssentialWCF"));

serviceHost.AddServiceEndpoint(typeof(IStockService), new BasicHttpBinding(), "");

Console.WriteLine("Активизируем сервис");

serviceHost.Open();

Console.WriteLine("Для завершения нажмте Ввод");

Console.ReadLine();

serviceHost.Close();

}

}

}

В данном примере выполняются следующие действия:

* **Определение контракта**. Пишется класс, который делает нечто полезное, после чего он снабжается атрибутами WCF. Атрибут [ServiceContract] помечает интерфйс, как контракт. В терминах языка WSDL [ServiceContract] определяет тип порта PortType. Атрибут [OperationContract] определяет методы класса, которые можно вызывать через интерфейс службы. Одновременно он определяет, какие сообщения можно передать этим методам и получить от них. С точки зрения WSDL, этот атрибут соответствует разделам Operations и Messages. В листинге определен класс StockService, содержащий единственный метод GetPrice.
* **Определяется оконечная точка**. В определении оконечной точки адрес, привязка и контракт задаются с помощью метода AddServiceEndpoint класса ServiceHost. Адрес мы оставляем пустым, это означает, что адрес оконечной точки такой же, как адрес самой службы. В качестве привязки указывается basicHttpBinding, совместимая со спецификацией WS\_I BP 1.1 и обеспечивающая интероперабельность с большинством систем, в которых реализованы Web\_службы на базе XML.
* Размещается сервисв процессе EssentialWCF, чтобы она могла прослушивать входящие запросы. В примере служба размещается в консольном приложении с помощью класса ServiceHost.

Сервис ожидает поступления запросов на адрес <http://localhost:8000/EssentialWCF>.

**Реализация службы с помощью кода и конфигурационных файлов**

В WCF имеется развитая поддержка для определения атрибутов службы в конфигурационных файлах. Кодировать алгоритм работы службы все равно придется, но задание адресов, привязок и поведений оконечных точек можно перенести из кода в конфигурационные файлы. Задание оконечных точек и поведения в конфигурационных файлах обеспечивает большую гибкость по сравнению с заданием в коде. Предположим, например, что оконечная точка была написана для работы с клиентами по протоколу HTTP. В листинге 1.1 это реализуется за счет того, что методу AddServiceEndPoint передается объект типа BasicHttpBinding. А теперь допустим, что вы решили изменить привязку на WSHttpBinding, поскольку она обеспечивает повышенную безопасность, добавляя к защите на уровне транспортного протокола еще и защиту на уровне отдельных сообщений. В таком случае пришлось бы изменить и перекомпилировать программу. Если же перенести выбор привязки из кода в конфигурацию, то можно будет обойтись без перекомпиляции. А если вы захотите распространить

контракт на оба протокола, достаточно будет определить две оконечные точки: одну для базового протокола HTTP, а другую для протокола WS\_Security; ничего изменять в коде при этом не придется. За счет этого сопровождение программы упрощается.

В листинге 1.2 приведен полный код WCF\_службы, размещаемой в консольном приложении. Для нее требуется конфигурационный файл, в котором содержится информация о поведении и оконечной точке. В данном примере выполняется следующее:

* **Определяем контракт**. Пишем класс, выполняющий полезную работу, и снабжаем его атрибутами WCF. Код самой службы одинаков что при программном определении, что при описании в конфигурационных файлах. В листинге 1.2 класс называется StockService, как и в листинге 1.1

**Размещаем службу в процессе операционной системы.**Это делается для того чтобы клиенты могли обращаться к ней из сети. Для этого создается объект класса ServiceHost, определенного в пространстве имен System.ServiceModel, и вызывается его метод Open точно так же, как в листинге 1.1.

* **Создаем конфигурационный файл**

В нем определяется базовый адрес службы и АПК ее оконечной точки. **Отметим,** что код в листинге 1.2 никак не ссылается на конфигурационный файл. При вызове метода ServiceHost.Open WCF ищет в конфигурационном файле приложения (app.config или web.config) секцию <serviceModel>, из которой читает конфигурационные параметры.

**Листинг 1.2. Реализация службы с помощью кода**

**и конфигурационного файла**

using System;

using System.ServiceModel;

namespace EssentialWCF;

{

[ServiceContract]

public interface IStockService

{

[OperationContract]

double GetPrice(string ticker);

}

public class StockService : IStockService

{

public double GetPrice(string ticker)

{

return 94.85;

}

}

public class Service

{

public static void Main()

{

ServiceHost serviceHost = new ServiceHost(typeof(StockService));

serviceHost.Open();

Console.WriteLine("Для завершения нажмите <ENTER>.\n\n");

Console.ReadLine();

serviceHost.Close();

}

}

}

В листинге 1.3 приведен конфигурационный файл, используемый совместно с программой из листинга 1.2. В секции <serviceModel> определена конечная точка. Для каждой оконечной точки задаются адрес, привязка и контракт. Адрес в данном случае пуст, это означает, что нужно использовать тот же адрес, что для самой службы. Если у службы несколько оконечных точек, то у каждой должен быть уникальный адрес. В качестве привязки мы указали basicHttpBinding, а в качестве имени контракта – имя класса, определенного в коде, – EssentialWCF.StockService.

**Листинг 1.3. Конфигурационный файл для службы,**

**реализованной в листинге 1.2**

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<system.serviceModel>

<services>

<service name="EssentialWCF.StockService">

<host>

<baseAddresses>

<add baseAddress="http://localhost:8080/EssentialWCF"/>

</baseAddresses>

</host>

<endpoint address=""

binding="basicHttpBinding"

contract="EssentialWCF.IStockService" />

</service>

</services>

</system.serviceModel>

</configuration>