Лекция 24. Windows Communication Foundation

# РАЗРАБОТКА ПРИЛОЖЕНИЙ НА ПЛАТФОРМЕ .NET

# API распределенных систем

- DCOM – Distributed COM. Вызов COM объектов на удаленной машине.
  - Используется реестр Window.
  - Windows-зависимая архитектура.
- COM+ – добавлен способ управления транзакциями.
  - Windows-зависимая архитектура.
  - В .NET можно взаимодействовать с COM+ с помощью библиотеки System.EnterpriseServices
- MSMQ – Microsoft Message Queuing – очередь сообщения Microsoft.
  - Отказоустойчивый механизм взаимодействия
  - В .NET можно взаимодействовать с MSMQ с помощью библиотеки System.EnterpriseServices
- .NET Remoting
  - Взаимодействие .NET приложений на разных компьютерах
  - Возможность передавать данные в двоичном виде
  - Единая система типов – CTS
  - Частично кроссплатформенная (Mono – на Unix)
  - Использование конфигурационных файлов для настройки сервиса
  - В .NET это пространство имен - System.Runtime.Remoting

# API распределенных систем

- Web-службы XML
  - Не зависит от платформы. Позволяет взаимодействовать программах на абсолютно разных языках и разных ОС
  - Используются только открытые технологии
    - Используется протокол HTTP
    - Данные передаются посредством XML (SOAP)
  - World Wide Web Consortium (W3C) и Web Services Interoperability Organization (WS-I) – разработка единых спецификаций WS-*
  - Недостаток – плохая производительность.
  - Пространство имен в .NET - System.Web.Services
- RESTfull Web-службы
  - Не зависит от платформы. Позволяет взаимодействовать программах на абсолютно разных языках и разных ОС
  - Используют возможности URI, и протокола HTTP
  - Данные передаются посредством персонализированного (custom) XML
  - Пространство имен в .NET - System.Web.Services
- Именованные каналы (pipe)
  - Взаимодействие между процессами
  - Очень быстрое взаимодействие
  - В .NET пространство имен System.IO.Pipe
- Одноранговые Pear-to-Pear (P2P)
  - Пространство имен в .NET – System.Net.PearToPear

# Windows Communication Foundation (WCF)

- До WCF
  - Разные технологии – трудный выбор нужной
  - Функциональность многих пересекается, что еще затрудняет выбор
  - Для каждой технологии свой API, свои инструменты для работы с ними
  - Очень тяжелый переход от одной технологии к другой. Фактически необходимо переписать огромные куски кода.
- WCF
  - Интегрирует все ранее разработанные технологии распределенного взаимодействия в единый стройный API-интерфейс.
  - Единый стиль построения сервиса для всех технологий
  - Для использования другого нижележащего API не нужно переписывать код и даже не нужно перекомпилировать приложение. Достаточно изменить несколько строк в конфигурационном файле.
  - Один и тот же сервис (контракт) может вещаться одновременно используя несколько нижележащих API.

# Обзор средств WCF

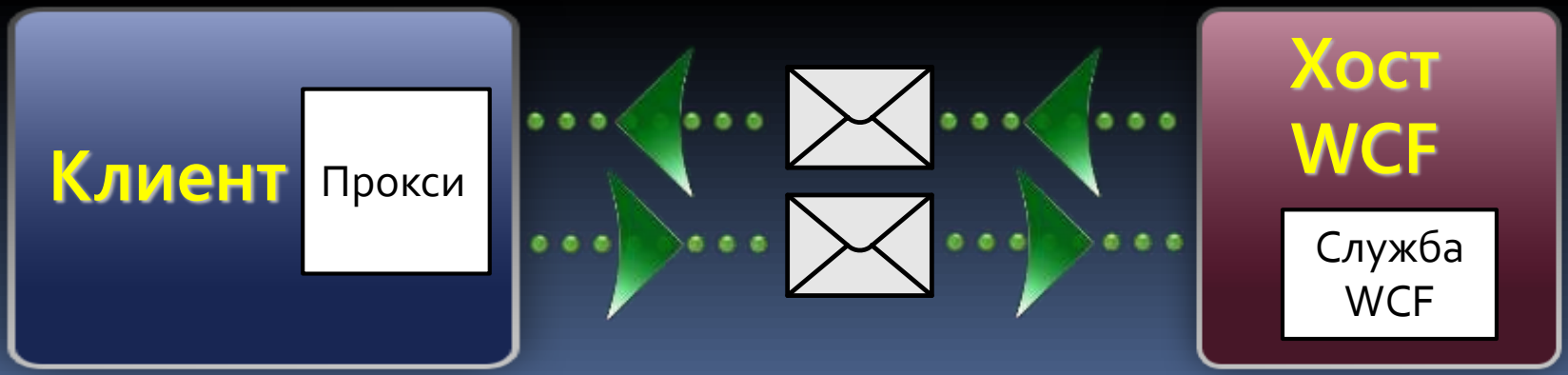
- Взаимодействие и интеграция различных API-интерфейсов
- Поддержка как строго типизированных, так и не типизированных сообщений
- Поддержка нескольких привязок (HTTP, TCP, MSMQ, Pipe)
- Поддержка спецификаций веб-служб WS-*
- Полностью интегрированная модель безопасности. Поддерживаются как встроенные Windows/.NET, так и нейтральные технологии защиты, построенные на стандартах веб-служб.
- Поддержка технологий хранения состояния сеансов, а также поддержка однонаправленных сообщений без состояния
- Базируется на принципах дизайна архитектуры, ориентированной на службы (Service-oriented architecture - SOA)

# Принципы SOA

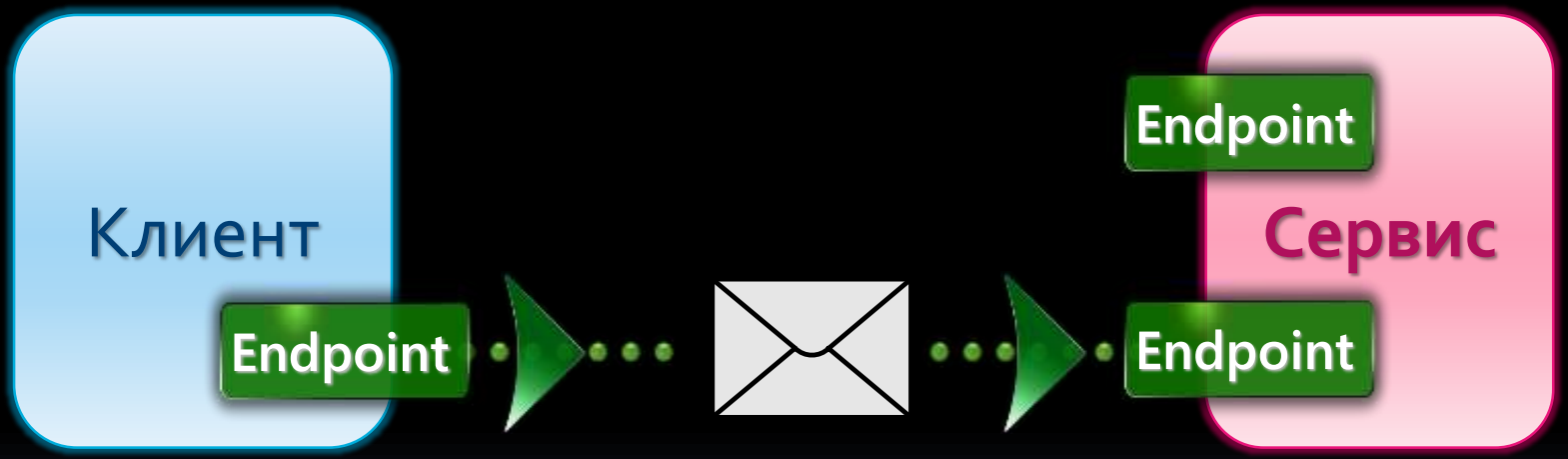
| | |
|---|---|
| Явные границы | Функциональность службы определяется через четко определенный интерфейс. Взаимодействие со службой только через интерфейс |
| Сервисы автономны | Сервисы и клиенты разрабатываются и развертываются полностью независимо. |
| Используют контракты, а не реализацию | Детали реализации службы не касаются вызывающей стороны. Взаимодействие исключительно через контракт |
| Совместимость основана на политиках | Возможности и требования определяются схемой; она используется для установления совместимости сервисов. |

# Основы WCF

- Служба WCF. Обычно это dll сборка. Содержит контракты и их реализацию
- Хост службы WCF. Публикует службу WCF, организует взаимодействие. В роли хоста может выступать любой тип приложения (консольное, WinForms, WPF, Служба Windows), IIS
- Клиент WCF. Клиентское приложение

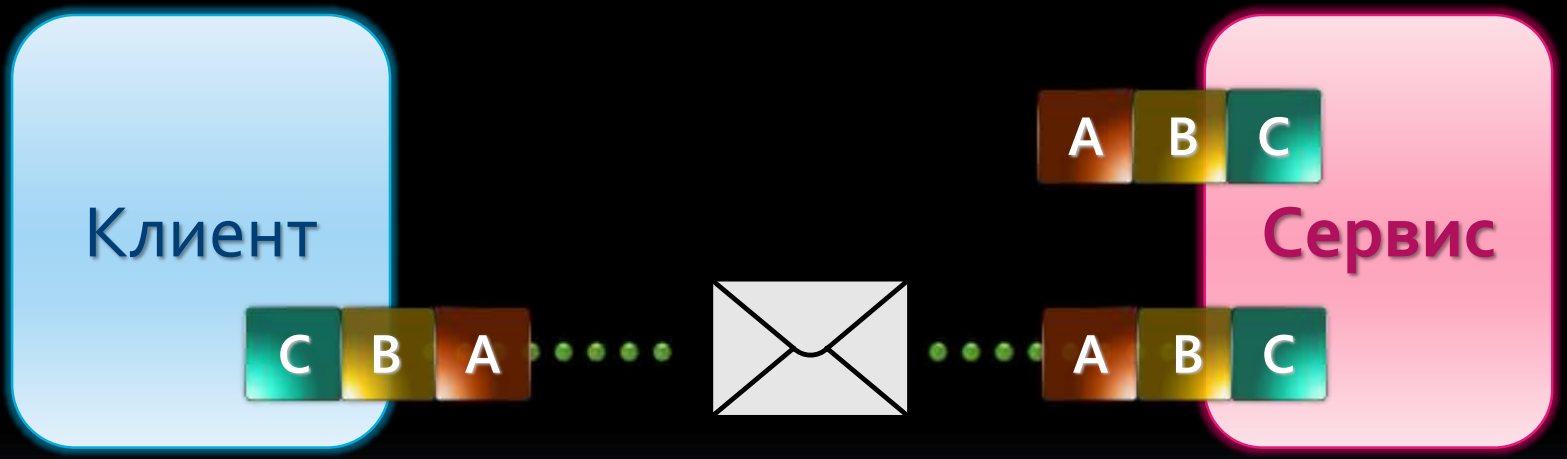


# Конечный точки




Обычно информация о Конечных точках хранится в конфигурационных файлах, но может быть жестко закодирована в Хосте службы и в прокси клиента

# ABC в WCF




**Address** *(Где)*

**Binding** *(Как)*

**Contract** *(Что)*

# Контракт

- Интерфейсы – контракты служб WCF
- Интерфейсы для работы с WCF помечаются атрибутом [ServiceContract]
- Каждый метод в интерфейсе помечается атрибутом [OperationContract]
- Классы, реализующий контракты служб – типы служб

```
[ServiceContract]
public interface ICalculator
{
   [OperationContract]
   Result SolveProblem (ComplexProblem p);
}
```

# Привязка

- Описывает
  - Транспортный уровень. Протокол передачи данных (HTTP, MSMQ, именованный каналы, TCP/IP)
  - Тип канала (однонаправленный, запрос-ответ, дуплексный)
  - Механизм кодирования (двоичный, XML, SOAP)
  - Поддерживаемые протоколы Web-служб, если разрешены (WS-Security, WS-Transaction, и т.д.)

# Привязка


Привязка
HTTP
Text
Security
RM
TX
Транспортные протоколы
TCP
HTTP
MSMQ
IPC
?
Кодировки
Text
Binary
?
Дополнительные Протоколы
Security
RM
TX
.NET
?

# Основные привязки

- Привязки на основе HTTP. Веб службы XML
  - BasicHttpBinding – соответствует спецификациям WS-I Basic Profile 1.1
  - WSHttpBinding – поддержка множества стандартов WS-* (безопасность, транзакции и др.)
  - WSDualHttpBinding – двухсторонний обмен сообщениями
  - WSFederationBinding – главное безопасность. Поддержка WS-Trust, WS-Security, WS-Federation
- Привязки на основе именованных каналов. Передача данных в двоичном виде
  - NetNamedPipeBinding – взаимодействие между приложениями на одном компьютере

# Основные привязки

- Привязки на основе TCP. Передача данных в двоичном виде
  - NetTcpBinding – оптимизированная передача параметров между .NET приложениями на разных компьютерах
  - NetPeerTcpBinding – безопасная привязка на основе P2P
- Привязки на основе MSMQ. Передача данных используя очереди сообщений Microsoft.
  - NetMsmqBinding – передача сообщений между .NET приложениями на разных компьютерах
  - MsmqIntegrationTcpBinding – передача сообщений между приложениями  использующих разные технологии (COM, C++ и т.д.)  на разных компьютерах

# Стандартные привязки

| | Интероп | Защита | Сессия | Транзакции | Дуплекс | Поток | |
|---|---|---|---|---|---|---|---|
| BasicHttpBinding | BP | T | | | | | |
| WsHttpBinding | WS | TS | ✓ | ✓ | | | |
| WsDualHttpBinding | WS | TS | ✓ | ✓ | ✓ | | |
| NetTcpBinding | | TS | ✓ | ✓ | ✓ | O | |
| NetNamedPipesBinding | | TS | ✓ | ✓ | ✓ | O | |
| NetMsmqBinding | | TS | ✓ | ✓ | | | |
| NetPeerTcpBinding | | TS | | | | ✓ | |

T = Защита на транспорте | S = WS-Security | O = One-Way Only

# Адрес

- Задается типом System.Uri или в *.config файле
- Зависит от выбранной привязки
- В общем случае должны быть заданы
  - scheme://MachineName[:port]/Path
  - Схема. Транспортный протокол. HPPP, TCP и т.д.
  - Имя машины.DNS имя или IP адрес или др. в зависимость от схемы
  - Порт. Номер порта. Некоторый протоколы имеют порт по умолчанию и он может быть опущен
  - Путь. Путь к службе WCF
- Примеры
  - http://localhost:8080/MyWCFService
  - net.tcp://localhost:8080/MyWCFService
  - net.pipe://localhost/MyWCFService
  - net.msmq://localhost/private$/MyPrivateQuery

# Хостинг службы WCF

- Публикует службу WCF, организует взаимодействие.
- В роли хоста может выступать:
  - любой тип приложения (консольное, WinForms, WPF)
  - Служба Windows
  - IIS (Internet Information Service)

# Хост службы WCF – приложение

Необходимы сборка System.ServiceModel и такое же пространство имен.

Хост службы WCF – всегда класс ServiceHost

```
ServiceHost host =
    new ServiceHost(typeof(MyWCFService));
host.Open();
.....
host.Close();
```

# Конфигурация конечных точек

```
<configuration>
  <system.serviceModel>
    <services>
      <service type="CalculatorService">
        <endpoint address="http://localhost/calculator"
                  binding="basicHttpBinding"
                  contractType="ICalculator" />
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

# Конфигурация конечных точек

Можно задать базовый адрес

```
<configuration>
  <system.serviceModel>
    <services>
      <service type="CalculatorService">
        <endpoint address=“”
                  binding="basicHttpBinding"
                  contractType="ICalculator" />
        <host>
          <baseAddresses>
            <add baseAddress="http://localhost:8732/calculator/" />
          </baseAddresses>
        </host>
      </service>
    </services>
  </system.serviceModel>
</configuration>
```

# Конфигурация привязок

```
<configuration>
  <system.serviceModel>
    <services>
      <service type="CalculatorService">
          <endpoint address="http://localhost/calculator"
                    binding="basicHttpBinding"
                    bindingConfiguration="Binding1"
                    contractType="ICalculator" />
      </service>
    </services>
    <bindings>
      <basicHttpBinding>
        <binding configurationName="Binding1"
                 hostNameComparisonMode="StrongWildcard"
                 sendTimeout="00:10:00"
                 maxMessageSize="65536"
                 messageEncoding="Text"
                 textEncoding="utf-8"
        </binding>
      </basicHttpBinding>
    </bindings>
  </system.serviceModel>
</configuration>
```

# Пользовательские привязки

```
<bindings>
    <customBinding>
      <binding configurationName="Binding1">
            <reliableSession bufferedMessagesQuota="32"
                inactivityTimeout="00:10:00"
                maxRetryCount="8"
                ordered="true" />
            <textMessageEncoding maxReadPoolSize="64"
                                 maxWritePoolSize="16"
                                 messageVersion="Default"
                                 writeEncoding="utf-8" />
            <httpsTransport manualAddressing="false"
                maxMessageSize="65536"
                hostNameComparisonMode="StrongWildcard"/>
      </binding>
    </customBinding>
</bindings>
```

# Класс ServiceHost

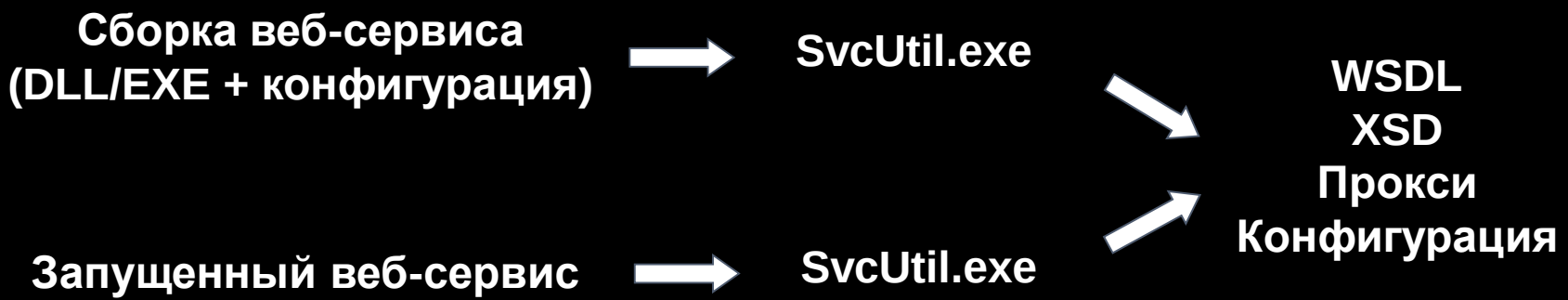
- Конструктор принимает набор адресов
- Свойства и методы
  - BaseAddresses – зарегистрированные адреса для службы
  - AddServiceEndpoint() – программное добавление конечной точки
  - AddDefaultEndpoints() – программное управление конечными точками по умолчанию
  - Open(), BeginOpen(), EndOpen() – синхронное, асинхронное открытие сервиса
  - Close(), BeginClose(), EndClose() – синхронное, асинхронное закрытие сервиса
  - State – состояние сервиса (открыт, закрыт, создан)

# Построение клиента

- Необходимы
  - общий контракт
  - привязка
  - адрес
- Статический импорт метаданных
  - Add Service Reference в Visual Studio
  - Утилита SvcUtil
  - Настроить руками. Сложно, но возможно
- Динамический импорт метаданных
  - Add Service Reference в Visual Studio для запущенного сервиса

# Автоматическая генерация контракта

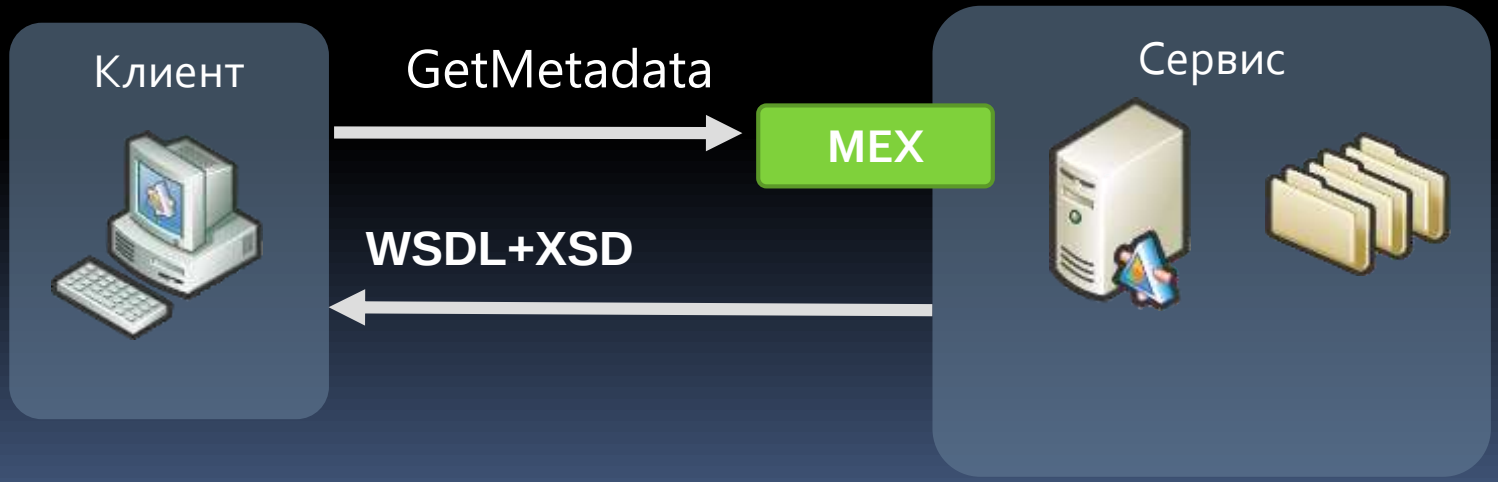
Во время разработки:



# Автоматическая генерация контракта

Во время выполнения:

*WS-MetaDataExchange* (MEX)



# Обнаружение WCF-сервисов

- Должно быть настроено в конфигурации

```
<endpoint address="mex"
                binding="mexHttpBinding"
                contract="IMetadataExchange" />

<service behaviorConfiguration="mybeh" ...>
...
<behavior name="mybeh" >
        <serviceMetadata httpGetEnabled="true" />
</behavior>
```

# Использование на клиенте

- Создан класс прокси MyServiceClient
- Класс прокси можно использовать как обычный класс. Но выполняться будет на сервисе

```
MyServiceClient service = new MyServiceClient();
        int x = service.MyMethod(4, 5);
        ...
service.Close();
```