Лабораторная работа №1. Распределенная архитектура

I

Теоретическая часть

С чего начинаем

http://en.wikipedia.org/wiki/Multitier_architecture
http://en.wikipedia.org/wiki/Middleware_(distributed_applications)
http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D1%91%D1%85%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0
http://ru.wikipedia.org/wiki/SaaS
http://ru.wikipedia.org/wiki/Windows_Communication_Foundation
http://web.archive.org/web/20140327102842/http://www.argc-argv.com/6_2007/article03.pdf

и далее по непонятным моментам изучаем самостоятельно.

Обращаем внимание:

Термины «Сервер приложений», «SaaS», «WCF», «хостинг WCF».

II

Практическая часть

Задача

Создать компоненты: библиотеку сервиса, сервер, клиент

1. Бизнес-логика веб-сервиса: библиотека сервиса

1.1. Создаем проект типа WCF-service library

1.2. Добавляем свою библиотеку предметной области в решение.

1.3. Создается проект, предназначенный для реализации бизнес-логики обслуживания вызовов с использованием веб (WCF). Внутри интерфейса (по умолчанию находится в IService1.cs), помеченного атрибутом ServiceContract, должны находится методы, помеченные OperationContract, которые будут доступны удаленно. Все сложные типы данных (рекурсивно вплоть до простых типов) должны быть помечены атрибутом DataContract.

1.4. Реализуется объявленный интерфейс (класс, реализующий интерфейс (по умолчанию Service1.cs))

1.5. VS предоставляет возможность тестирования сервиса (стандартное нажатие «Отладка»). Без хоста сервисом в рабочем режиме нельзя воспользоваться.

2. Сервер: обслуживающая инфраструктура сервиса

2.1. Создадим приложение-хост (это может быть и служба Windows): обычное оконное приложение, которое будет сервером приложения.

2.2. Добавляем Reference на .NET-библиотеку System.ServiceModel у проекта

2.3. Можно использовать два способа публикации сервиса: программно создавать описание серверной части и использовать файл конфигурации. Будем использовать первый способ.

2.4. Добавляем ссылки в проект сервера на пространство имен System.ServiceModel и на свою предметную библиотеку. Добавляем в оконную форму либо (предпочтительнее) в документ формы ссылку на используемое пространство имен (using System.ServiceModel; using System.ServiceModel.Description).

2.4. Добавляем к форме член класса

```
private ServiceHost m_pHost;
```

2.5. Добавляем в конструктор формы инициализацию объекта:

```
        string sUrlService = "http://127.0.0.1:8000/Service1";
        string sUrlServiceMeta = "http://127.0.0.1:8000/Service1/Meta";
        // Привязка для основного сервиса
        BasicHttpBinding pBinding = new BasicHttpBinding();
        pBinding.Security.Transport.ClientCredentialType =
HttpClientCredentialType.None;
        pBinding.Security.Mode = BasicHttpSecurityMode.None;
        // Поведение для публикации информации о сервисе
        ServiceMetadataBehavior smb = new ServiceMetadataBehavior();
        smb.HttpGetEnabled = true;
        smb.HttpGetUrl = new Uri(sUrlServiceMeta);
        smb.MetadataExporter.PolicyVersion = PolicyVersion.Policy15;

        // Создаем хост
        m_pHost = new ServiceHost(typeof(Service1));
        m_pHost.Description.Behaviors.Add(smb);
        // Добавляем обслуживание основного функционала
        m_pHost.AddServiceEndpoint(typeof(IService1), pBinding, sUrlService);
        // Добавляем обслуживание публикации информации о сервисе
        m_pHost.AddServiceEndpoint(typeof(IMetadataExchange),
MetadataExchangeBindings.CreateMexHttpBinding(), sUrlServiceMeta);
```

2.6. Добавляем кнопку старта и остановки сервера на форму. Устанавливаем на кнопки обработчик старта и остановки сервиса.

```
        m_pHost.Open();
```

и

```
        m_pHost.Close();
```

2.7. В настройках проекта библиотеки сервиса, созданного в п.1.1, в разделе WCF Options отключить автоматический запуск сервиса.

2.8. После запуска приложения открыть http://127.0.0.1:8000/Service1/Meta - откроется страница с описанием сервиса. Адрес работы сервера: http://127.0.0.1:8000/Service1

3. Клиентское приложение

3.1. Создадим клиентское приложение (например, оконное).

3.2. Для связи с сервисом можно использовать два способа: программно создавать описание клиентской части, использовать файл конфигурации либо создать клиентскую часть автоматически на основании сервиса. Воспользуемся последним вариантом. Add service reference на клиентском проекте, далее Discover. В результате создается клиентская обертка (Например, ServiceReference1.Service1Client). Этот класс можно использовать для взаимодействия с сервером.

3.3. Добавляем в модуль клиентского приложения ссылки на System.ServiceModel

3.4. Добавляем в клиентское приложение функции взаимодействия с сервисом. Добавить кнопку. На обработчик кнопки повесить обращение к сервису.

```
        string sUrlService = "http://127.0.0.1:8000/Service1";
        BasicHttpContextBinding pBinding = new BasicHttpContextBinding();
        EndpointAddress pEndpointAddress = new EndpointAddress(sUrlService);
        Service1Client pClient = new Service1Client(pBinding, pEndpointAddress);
        MessageBox.Show(pClient.GetData(1));
```

3.5. Запускаем приложение-сервер и приложение-клиент (Debug -> Start new instance) либо Свойства решения -> тип запуска: несколько проектов.

3.6. Нажимаем пуск у сервера. Проверяем из клиента работоспособность.

Заключение и выводы:

Создали сервис, обслуживающий запросы. Отделили приложение от бизнес-логики.

Лабораторная работа №2. Применение WCF-сервиса

I

Теоретическая часть

См. Лабораторную работу №1.

II

Практическая часть

Материалы в электронном виде:

https://github.com/sergeyverevkin/pi19/tree/master/pi19_02/L1_WCF

Предметная область

Электронная энциклопедия содержит список словарных статей с одной опциональной заглавной иллюстрацией, рассортированных по категориям (разделам).

Задача

Создать прикладное оконное приложение, использующее данные с сервиса для отображения. Создать сервис (приложение-сервер с хостингом WCF) по заданному контракту.

Детализировать предметную область, уточнив предметную область и контракт взаимодействия.

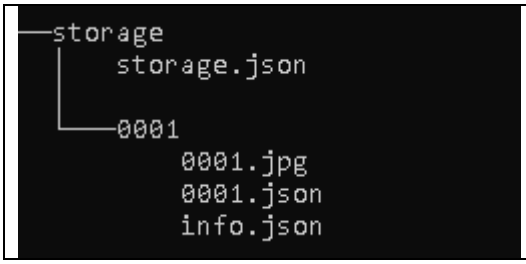
Расширить функции:

1. Добавить в контракт один или больше методов, расширяющих бизнес-логику из перечня либо придумать самостоятельно:

— получение списка избранных статей (визуализация) по логину пользователя;
— отметка статьи избранной по логину пользователя;
— добавление категории;
— добавление статьи (по логину пользователя и содержимому);
— добавление/замена картинки в статье;
— комментарии к статье: добавление и просмотр;
— ограничить доступ по логину/паролю;
— сделать локализацию (перевод) статей (поддержка нескольких языков), изменив структуру хранения;

2. Расширить предметную область (продумать самостоятельно, в соответствии с дополнительными методами)

Данные хранить на сервере. Предлагается это делать в файловой структуре. Формат хранения произвольный (json, xml, txt). Примерная структура хранения:

	Папка хранения на сервере Файл описания энциклопедии со списком разделов Раздел 0001 Титульный рисунок словарной статьи Файл описания словарной статьи Файл описания раздела

Контракт взаимодействия

```
/// <summary>
/// Сервис "Электронная энциклопедия"
/// </summary>
[ServiceContract]
```

```
public interface IEncyclopediaService
{
  /// <summary>
  /// Получение списка категорий и информации о энциклопедии
  /// </summary>
  /// <returns></returns>
  [OperationContract]
  EncyclopediaType GetInfo();

  /// <summary>
  /// Получить информацию по разделу энциклопедии
  /// </summary>
  /// <param name="sCode"></param>
  /// <returns></returns>
  [OperationContract]
  EncyclopediaPartType GetPart(string sCode);

  /// <summary>
  /// Получить полную словарную статью
  /// </summary>
  /// <param name="sCode"></param>
  /// <returns></returns>
  [OperationContract]
  EncyclopediaArticleType GetArticle(string sCode);
}

/// <summary>
/// Энциклопедия
/// </summary>
[DataContract]
public class EncyclopediaType
{
  /// <summary>
  /// Название энциклопедии
  /// </summary>
  [DataMember]
  public string Title { get; set; }

  /// <summary>
  /// Список разделов энциклопедии
  /// </summary>
  [DataMember]
  public EncyclopediaPartType[] PartList { get; set; }

  // TODO
}

/// <summary>
/// Раздел энциклопедии
/// </summary>
public class EncyclopediaPartType
{
  /// <summary>
  /// Список разделов энциклопедии
  /// </summary>
  [DataMember]
  public EncyclopediaArticleInfoType[] ArticleInfoList { get; set; }

  // TODO
}

/// <summary>
/// Краткая информация о статье энциклопедии
/// </summary>
public class EncyclopediaArticleInfoType
{
  // TODO
}

/// <summary>
/// Полная статья энциклопедии с иллюстрацией
/// </summary>
public class EncyclopediaArticleType
{
  // TODO
}
```