УДК 004.89

ПРИМЕНЕНИЕ ASP.NET ДЛЯ СОЗДАНИЯ WEB- СЕРВИСОВ

В.Д.БОРЩЕВА, С.К.АБДИЕВА
E.mail. ksucta@elcat.kg

Microsoft ASP.NET (WebMethods) Web – сервистери методдору Web – сервистерди түзүүгө карата жогорку өндүрүмдүү ыкманы камсыздандырат. WebMethods болсо HTTP, XML, XML Schema, SOAP жана WSDL ./1/ди колдоп турган Web – сервис операциялары сымал Microsoft .NET салтык методдорун ачып бере алат.

Методы Web-сервисов Microsoft ASP.NET (WebMethods) обеспечивают высокопроизводительный подход к построению Web-сервисов. WebMethods могут раскрывать традиционные методы Microsoft .NET, такие как операции Web-сервиса, которые поддерживают HTTP, XML, XML Schema, SOAP и WSDL /1/.

Method Web-services Microsoft ASP.NET (WebMethods) provide the high-efficiency approach to construction Web-services. WebMethods can open traditional methods Microsoft.NET, such as operations Web-service which support HTTP, XML, XML Schema, SOAP and WSDL. [1]

Web-сервисы – это код, к которому можно обратиться через HTTP. Так как все это реализовано через HTTP, то язык, на котором будет строиться Web-сервис, не имеет значения. В качестве клиентов используются как Windows, так и Web-приложения. Для вызова методов Web-сервиса можно воспользоваться HTTP или XML и SOAP /1/. Web-сервис строится из тех же компонентов .NET (классы, интерфейсы, сборки) и выполняет роль «черного ящика» для клиента, возвращая ответ на запрос. Для того, чтобы Windows и консольные приложения могли работать с Web-сервисами, в .NET существуют средства для генерации проксисборок, которые перенаправляют запросы от приложений на Web-сервис через HTTP и SOAP /3/.

Сегодня существует два фундаментально различных способа реализации Web-сервисов, основанных на HTTP, в Microsoft .NET. Первой и наиболее низкоуровневой техникой является написание специального класса IHttpHandler, который вставляется в цепочку .NET HTTP pipeline. Этот подход требует использования System.Web API для обработки входящих HTTP сообщений и System. Xml API – для обработки конверта SOAP, находящегося в теле HTTP. При написании специального обработчика также требуется создать вручную документ WSDL, который точно описывает вашу реализацию. Чтобы сделать все это правильно, необходимо глубокое понимание спецификаций XML, XSD, SOAP и WSDL.

Более продуктивным способом реализации Web-сервисов является использование WebMethods оболочки Microsoft ASP.NET. С ASP.NET поставляется специальный класс IHttpHandler для .asmx (называемых WebServiceHandler), который обеспечивает набор необходимых функциональных возможностей XML, XSD, SOAP и WSDL.

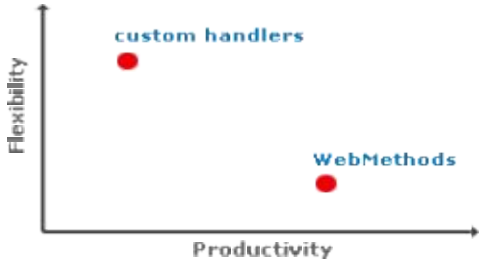

Рис. 1. Соотношение выгод и потерь между гибкостью и продуктивностью

Выбор между техниками реализации приводит к общему сравнению выгод и потерь между гибкостью и продуктивностью, как показано на рис. 1. Создание специального IHttpHandler дает неограниченную гибкость, но также требует большего времени на написание, тестирование и отладку кода. Оболочка WebMethods облегчает организацию Web-сервиса и быстроту разработки, но ограничивает рамками оболочки. Однако существует возможность расширения оболочки WebMethods, добавляя собственные дополнительные функциональные возможности /2/.

WebMethods поставляет основные сервисы, которые необходимы большинству конечных Web-сервисов, а также некоторые интересные возможности расширения, которые позволяют привести оболочку в соответствие вашим конкретным надобностям. Исходя из этого, далее в статье обсуждаются внутренние механизмы работы WebMethods /4/.

Создание проекта

Необходимое программное обеспечение

1. IIS (входит в состав Windows).
2. Visual Studio 2010.

Создаем в Visual Studio 2010 проект ASP.NET Empty Web Application под названием MyFirstWebService . Solution Explorer примет следующий вид:

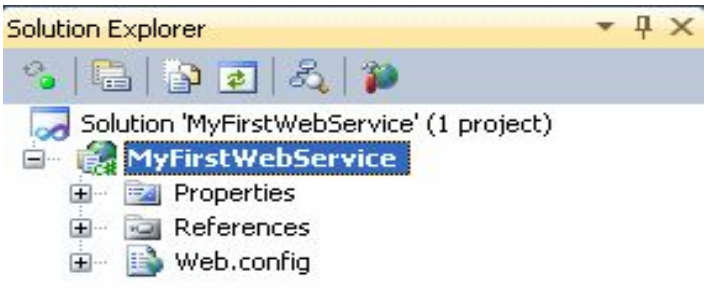

Рис. 2. Solution Explorer (Обозреватель Решения)

Проект имеет тип Empty Web Application, необходимо добавить файл типа WebService с расширением .asmx /2/.

Оболочка WebMethods

Оболочка WebMethods занимается преобразованиями SOAP сообщений в методы класса .NET. Это делается, прежде всего, путем аннотирования методов, атрибутом [WebMethod], находящимся в пространстве имен System.Web.Services. Например, следующий класс .NET WebService.amx содержит четыре метода, два из которых аннотированы атрибутом [WebMethod] /5/:

```
[WebMethod]
  public double Add(double x, double y) {
    return x + y;
  }
  [WebMethod]
  public double Subtract(double x, double y) {
    return x - y;
  }
  public double Multiply(double x, double y) {
    return x * y;
  }
  public double Divide(double x, double y) {
    return x / y;
```

}

Чтобы использовать этот класс с оболочкой WebMethods, нужно запустить проект. В этом примере методы Add и Subtract могут быть раскрыты как операции Web-сервиса, в то время как с методами Multiply и Divide этого сделать нельзя (так как они не были отмечены атрибутом [WebMethod]).

Раскрываются Add и Subtract как операции Web-сервиса через .asmx файл.
При запуске можно увидеть следующее:

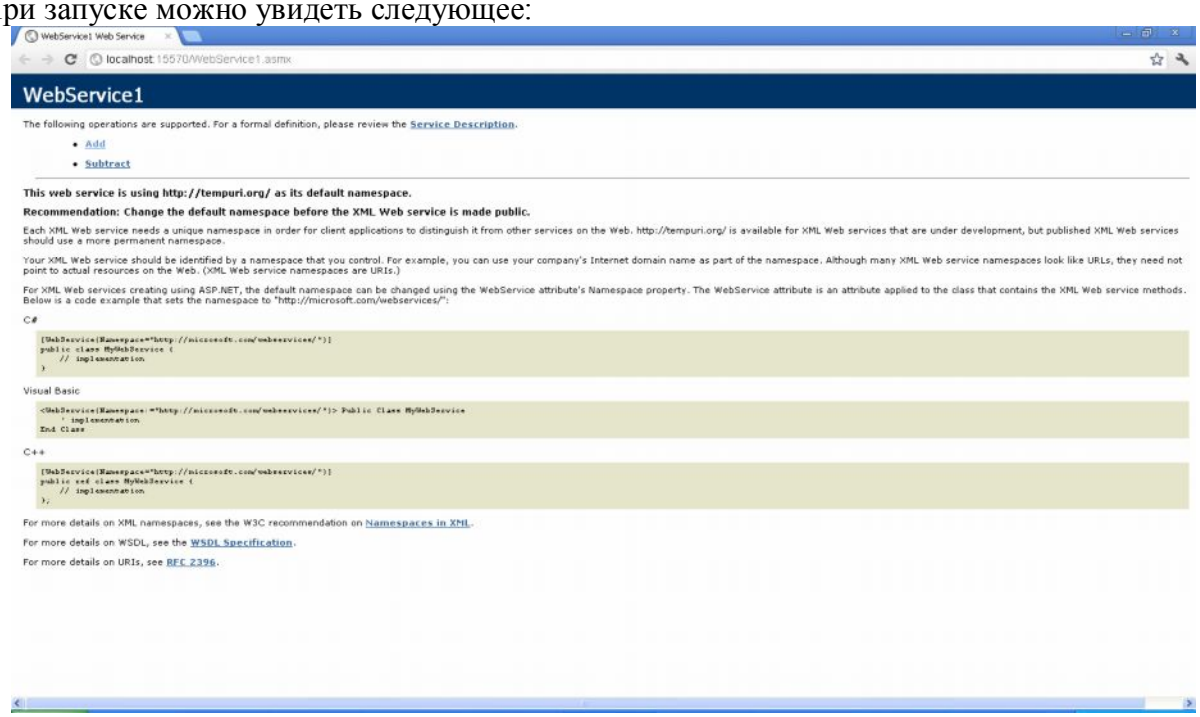

Рис. 3. Запущенный сервис

При нажатии на кнопку выходит следующее сгенерированное окно.

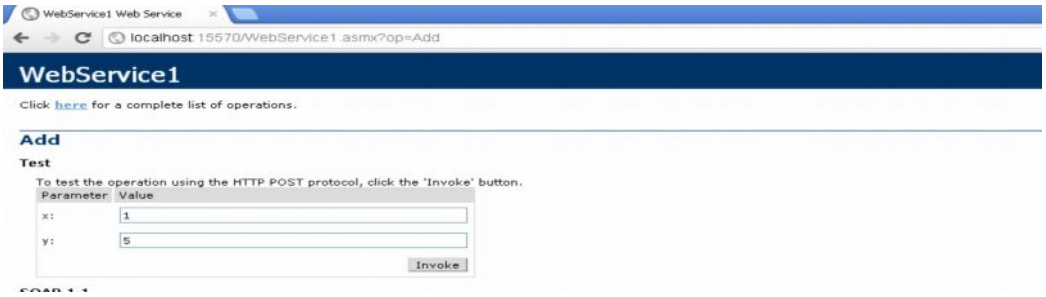

Рис. 4. Работа функции Add, введенные значения 1, 5

У проекта меняются свойства Propertis:

- На закладке Web нужно установить:
  Specific - WebService1.asmx
- Use Local IIS Web server нажать кнопку Create Virtual Directory

Строим Клиент – Серверное приложение.

В Solution Explorer нужно добавить проект Windows Form Application с именем Form.

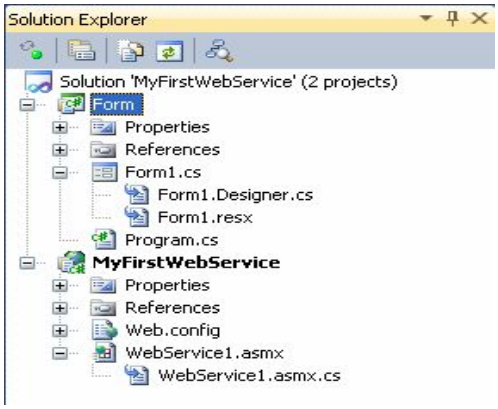

Рис. 5. Структура проекта

В данный проект следует сделать ссылку на Web Service – Add Web Service Peference. В поле адрес вводим http://localhost/MyFirstWebService/WebService1.asmx и нажимаем на кнопку Go. Задать Namespace – Service1 и нажать на кнопку Advanced – Add Web Reference /5/.

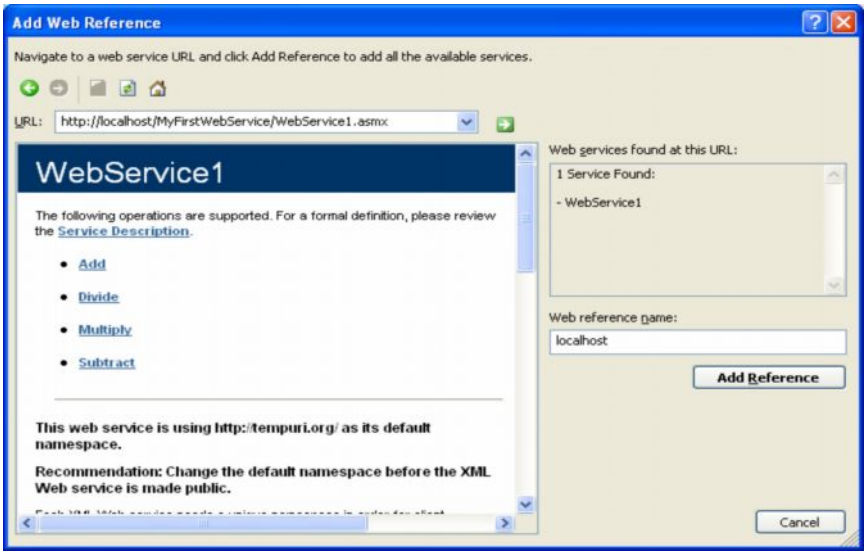

Рис. 6. Добавления ссылки на Web-сервис

Следует добавить в проект файл конфигурации Web-сервиса – XML Configuration File с именем WebService.Config.

Код:

```
<?xml version="1.0" encoding="utf-8" ?>
<WebServicesUrls>
  <PrototypeServiceUrl
TimeOut="600000">http://localhost/MyFirstWebService/WebService1.asmx
</PrototypeServiceUrl>
</WebServicesUrls>
```

В этом коде прописывается период ожидания обращения к сервису. В закладке Properties данного файла необходимо поменять Cope to Output Directory на Copy always. Это необходимо для того, чтобы наш конфигурационный файл всегда копировался в выходную директорию, и проект мог его увидеть.

Необходимо добавить в проект два класса WebServiceConfig.cs и WebServiceFactory.cs.

Код WebServiceConfig.cs:

```
using System.Xml;
   public class WebServiceConfig
```

```
    {
        private XmlDocument xmlDocument = new XmlDocument();
        public WebServiceConfig(string configFileName)
        {
            xmlDocument.Load(configFileName);
        }
        public string GetUrl(string service)
        {
            var urlElement = xmlDocument.GetElementsByTagName(service)[0];

            return urlElement.InnerText;
        }
        public int GetTimeOut(string service)
        {
            var urlElement = xmlDocument.GetElementsByTagName(service)[0];

            return int.Parse(urlElement.Attributes["TimeOut"].Value);
        }
    }
```

Код WebServiceFactory.cs

```
using System;
using System.Net;
using System.Web.Services.Protocols;
using System.Windows.Forms;
using WindowsFormsApplication1.localhost;
    public class WebServiceFactory
    {
        private static readonly CookieContainer globalCookieContainer = new CookieContainer();
        private static readonly WebServiceConfig webServiceConfig = new
WebServiceConfig(Application.StartupPath + "\\WebService.config");
        private static WebService1 prototypeService;
        private static SoapHttpClientProtocol CreateService(Type serviceType, string
configKeyName)
        {
            var protocol = (SoapHttpClientProtocol) Activator.CreateInstance(serviceType);
            protocol.Url = webServiceConfig.GetUrl(configKeyName);
            protocol.Timeout = webServiceConfig.GetTimeOut(configKeyName);
            protocol.EnableDecompression = false;
            protocol.CookieContainer = globalCookieContainer;
            return protocol;
        }
        public static WebService1 CreatePrototypeService()
        {
            if (prototypeService == null)
            {
                prototypeService = (WebService1)CreateService(typeof(WebService1),
"PrototypeServiceUrl");
                prototypeService.UseDefaultCredentials = true;
                prototypeService.PreAuthenticate = true;
                prototypeService.Credentials = CredentialCache.DefaultCredentials;
            }
            return prototypeService;
```

```
    }
}[6]
```

На форму выносим следующие компоненты:

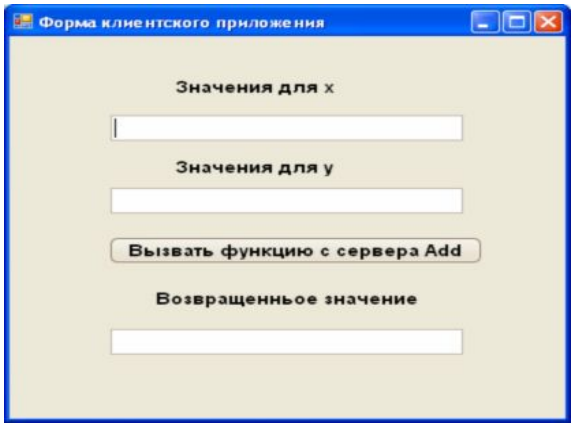

Рис. 7. Форма клиентского приложения

На кнопку Вызвать функцию с сервера Add добавляется следующий код:

```
double x = Convert.ToDouble(textEdit1.Text);
double y = Convert.ToDouble(textEdit2.Text);
double i = WebServiceFactory.CreatePrototypeService().Add(x, y);
textEdit4.Text = i.ToString();
```

В строке WebServiceFactory.CreatePrototypeService().Add(x, y); необходимо обратиться к классу WebServiceFactory и открыть соединение с сервером /5/.

Заключение

Оболочка WebMethods ASP.NET обеспечивает высокопроизводительный подход к построению Web-сервисов. WebMethods сделали возможным использовать традиционные методы .NET в качестве операций Web-сервиса, которые поддерживают HTTP, XML, XML Schema, SOAP и WSDL. Обработчик WebMethod (*.asmx*) автоматически определяет, как отправить входящие SOAP сообщения в соответствующий метод, в какой точке он автоматически сериализует входящие XML элементы в соответствующие объекты .NET. И упрощая жизнь клиента, обработчик *.asmx* также обеспечивает автоматическую поддержку для генерирования документации для человека (HTML) и для машины (WSDL).

По сравнению со специальным **IHttpHandlers** оболочка WebMethods может выглядеть несколько ограниченной, она обеспечивает мощную модель расширяемости, известную как оболочка SOAP расширения. Расширения SOAP обеспечивают возможность для удовлетворения ваших конкретных нужд ввести дополнительные функциональные возможности. Например, Microsoft выпустила Web Services Enhancements 1.0 для Microsoft .NET (WSE), которые просто предоставляют класс **SoapExtension**, вводящий в оболочку WebMethods поддержку для нескольких GXA спецификаций.

Список литературы

1. Как работают Web-сервисы ASP.NET http://www.vbnet.ru/articles/showarticle.aspx?id=208 Автор: Aaron Skonnard, под переводом: Шатохина Надежда
2. Web-страницы и Web-сервисы (инструкции по C#) http://msdn.microsoft.com/ru-ru/library/ms186209(v=vs.90).aspx

3. Web-Сервисы http://simple-cs.ru/web-service.aspx
4. Подключение к Web-сервисам NAV из C# с помощью Web Reference http://naviart.ru/nav-web-service-connect-from-csharp-webref
5. Дэвид С. Плат. Знакомство с MICROSOFT .NET. – М., 2001. – С. 154, 155.