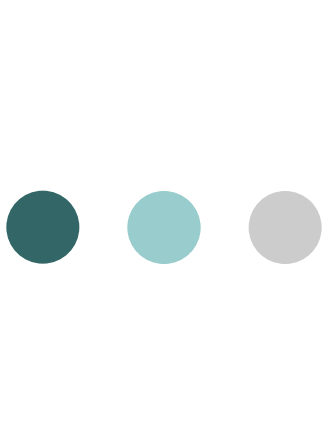
Из цикла лекций **«Internet-технологии разработки приложений»** для студентов 4-го курса кафедры Компьютерных технологий физического факультета Донецкого национального университета



Web-сервисы

Введение, протоколы, архитектура, создание Web-сервисов в среде Visual Studio .NET

проф. В.К.Толстых, www.tolstykh.com

Что это такое?

– отдельные независимые процедуры многократного использования, которые представляют свои функции через Web-интерфейс. Для связи с внешним миром и удаленного вызова процедур (RPC) используют протокол SOAP поверх протокола HTTP.

Любой клиент и любой сервер (потребители) могут использовать сервисы независимо от языка их реализации и устройства на котором они установлены. Web-сервисы основаны на открытых стандартах (используется XML), ими легко овладеть, и эти стандарты широко поддерживаются на всех платформах Unix и Windows.

Web-сервисы позволяют приложениям или другим Web-сервисам совместно использовать данные и функции таким способом, при котором не имеет значения, как именно эти приложения выполняются, какую платформу, операционную систему или устройство они используют.

Протоколы Web-сервиса

- **SOAP** (Simple Object Access Protocol) – простой протокол доступа к объектам. Основан на XML для дистанционного вызова процедур по Intranet и Internet. Определяет формат запроса и параметров, передаваемых в запросе.
- **WSDL** (Web Service Description Language) – протокол описания Web-секрвисов. Он позволяет предоставить описание и расположение всех методов Web-сервисов, а также их параметры на XML.
- **UDDI** (Universal Description, Discovery, and Integration) – универсальное описание, обнаружение и интеграция. Это открытый системный реестр, предназначенный для хранения информации о Web-сервисах. UDDI доступен по адресу www.uddi.org

Visual Studio, **Delphi** – это инструментальные средства, которые могут быть использованы для разработки Web-сервисов

Пример работы протокола SOAP

Сообщения между Web-сервисом и его пользователем пакуются в SOAP-конверты (SOAP envelopes). Вот как выглядит **простой SOAP-запрос**, который отправляется через HTPP к Web-сервису:

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope">
<env:Body>
<m:ValidatePostcode env:encodingStyle="http://www.w3.org/2001/06/soap-
  encoding" xmlns:m="http://www.somesite.com/Postcode">
<postcode>WC1A8GH</postcode>
<country>UK</country>
</m:ValidatePostcode>
</env:Body>
</env:Envelope>
```

Название Web-сервиса

Параметры запроса

Ключевые элементы SOAP-конверта узнать достаточно просто: это два параметра **<postcode>** ("почтовый индекс") и **<country>** ("страна"), которые содержатся внутри элемента под названием **<ValidatePostcode>**. Этот элемент является *названием Web-сервиса* к которой мы обращаемся с запросом: верный ли почтовый код для указанной страны?

Ответ:

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope" >
<env:Body>
<m:ValidatePostcodeResponse
env:encodingStyle="http://www.w3.org/2001/06/soap-encoding"
xmlns:m="http://www.somesite.com/Postcode">
<Valid>Yes</Valid>
</m:ValidatePostcodeResponse>
</env:Body>
</env:Envelope>
```

Ответ Web-сервиса

Параметры ответа

Элемент **<ValidatePostcode>** в запросе поменялся на элемент **<ValidatePostcodeResponse>** в ответе. В этом элементе содержится только один элемент **<Valid>**, значение которого обозначает, что почтовый индекс правильный.

Как это все работает

автор: Patrick Cooney и A List Apart

Представим себе, что я - разработчик сайта, и мой клиент попросил меня добавить к сайту новую функцию: необходимо добавить проверку правильности почтового индекса в регистрационной форме.

Для осуществления этой проверки мне понадобилось бы создавать базу данных всех почтовых индексов всех 30 стран, где наша компания ведет бизнес, а потом проверять при регистрации соответствие почтового индекса указанному в регистрации городу. Но у меня этих данных нет, и я думаю, что на сбор подобных данных придется потратить ощутимую сумму денег.

Вместо того, чтобы раскошеливаться на покупку базы данных, писать самому код, следить за целостностью и правильностью всех данных и отлаживать работу скриптов, я просто иду в каталог UDDI и ищу, нет ли там веб-сервиса, который мог бы сделать эту работу за меня. Придя на сайт www.uddi.org, я запускаю поиск и нахожу прекрасный сервис от компании XYZ Corp.

Я внимательно рассматриваю определение формата веб-сервиса (определение записано на языке WSDL, убеждаюсь, что сервис делает именно то, что мне нужно. Затем справляюсь у своих коллег о репутации компании XYZ Corp, узнаю, что она солидная, и затем обращаюсь к компании XYZ с вопросом о цене. Если цена на доступ к сервису доступна для моего бюджета, я пишу простую Web-страницу для своего сайта, которая вызывает веб-сервис компании XYZ Corp, и опля, на сайте появляется моментальная проверка почтового индекса.

Ещё пример

автор: Михаил Флёнов

Допустим, вы разрабатываете WEB магазин продажи книг. Можно составить каталог книг и никак не отслеживать наличие товара на складах и возможность поставок, но такое положение дел наверняка не устроит посетителей магазина.

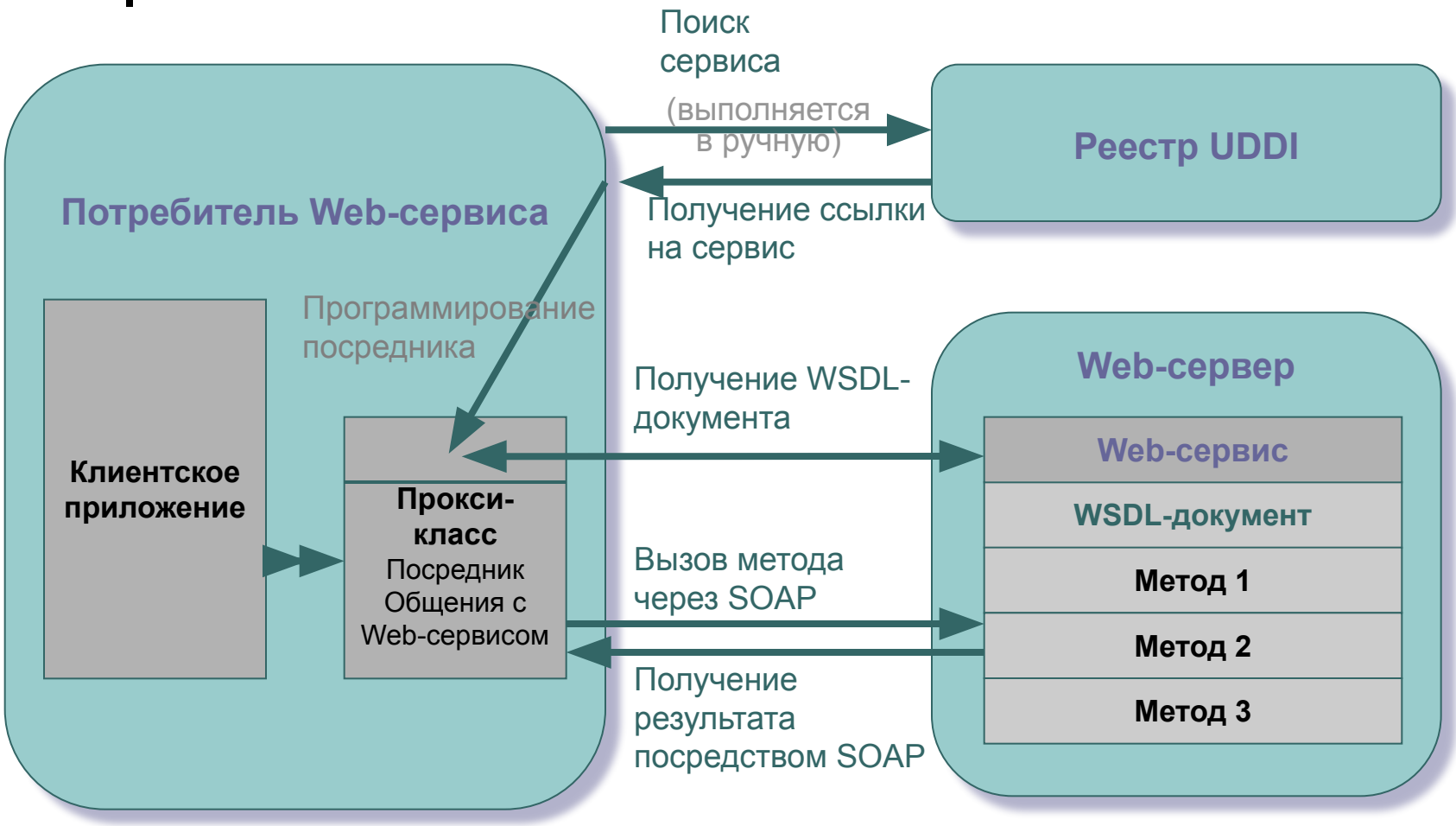
Чтобы отслеживать возможность поставки товара, вы можете сделать запрос к WEB-сервису оптовой компании, где закупаются книги. Он вернёт вам количество книг на оптовом складе, дату ближайшей поставки, если книги нет на складе. Эту информацию, вы можете легко показывать на своём сайте. С другой стороны, WEB сервис оптового магазина, для обработки запроса клиента, проверяет остатки собственной базы данных товара и запрашивает WEB сервис издателя книги, чтобы тот сообщил: а есть ли книга у издательства; если нет, когда будет переиздание; когда издательство может осуществить поставку оптовику и с какого склада;

Обращение интернет магазина к сервису оптовой компании и далее к сервису издательства, называется b2b (business to business). Это избавит вас от перебоя в поставках при минимальных затратах на хранение.

А если сервисы будут отображать и изменения цены, то магазин сможет автоматически устанавливать свою розничную цену как:

цена оптовой компании + цена поставки + N% прибыли.

Архитектура Web-сервисов

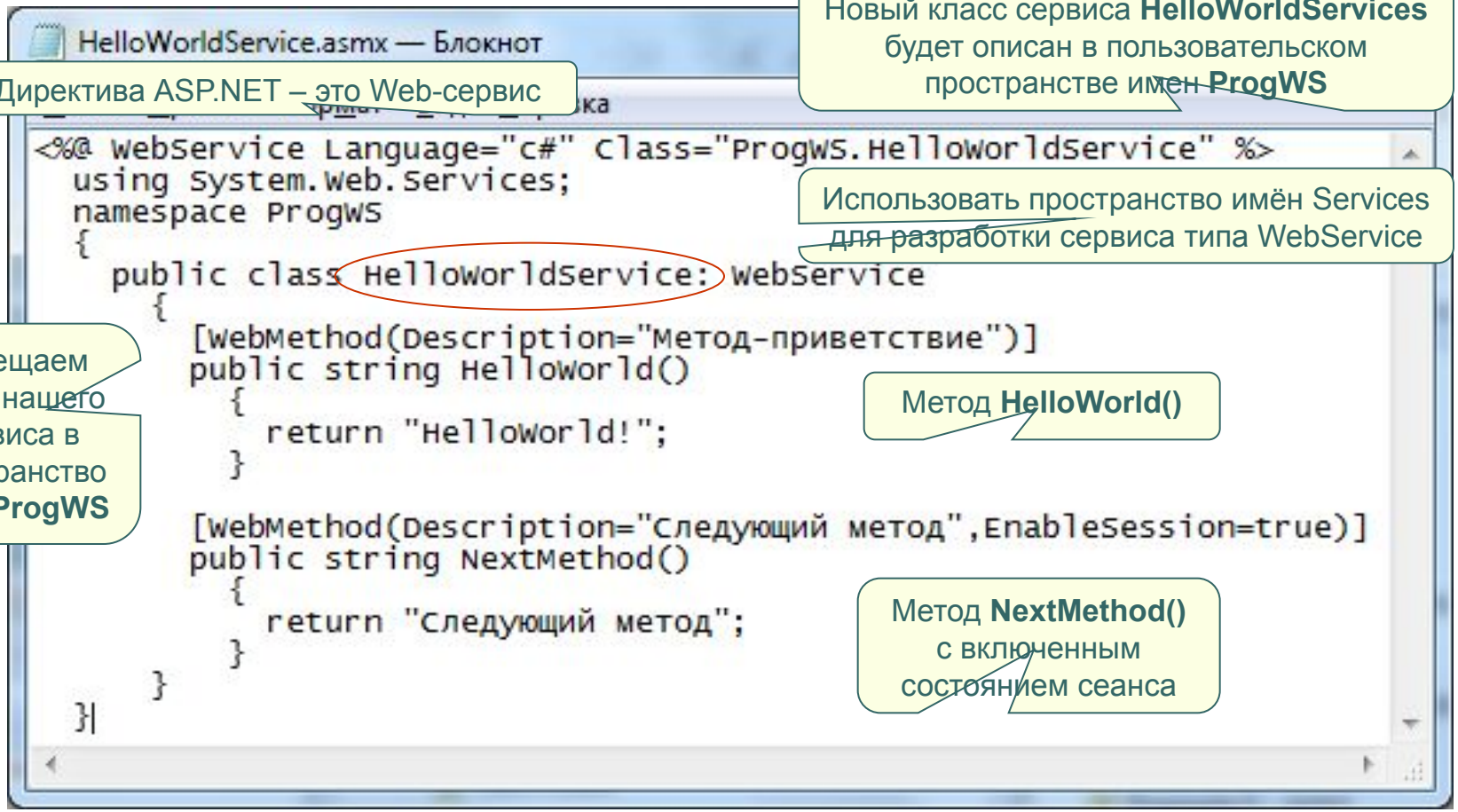


Реализация Web-сервисов для .NET приложений

1. Вы разрабатываете web-сервис как **.NET**-класс из System.Web.Services с директивой ASP.NET «<%@ WebService», которые идентифицируют такой класс как web-сервис с некоторыми функциями.
2. В среде **.NET** автоматически создается документ WSDL, где описывается, как клиент должен взаимодействовать с web-сервисом.
3. Потребитель находит ваш web-сервис и, решив воспользоваться ею, добавляет соответствующую web-ссылку в проект **Delphi .NET** или **Visual Studio .NET** и т.п. (или запускает утилиту wsdl.exe).
4. В среде **.NET** осуществляется автоматическая проверка документа WSDL и генерируется прокси-класс, который позволяет потребителю взаимодействовать с web-сервисом.
5. Потребитель вызывает один из методов вашего класса web-сервиса. С его точки зрения этот вызов не отличается от вызова метода любого другого класса, но в действительности потребитель взаимодействует с прокси-классом, а не с web-сервисом.
6. Прокси-класс преобразует переданные параметры в сообщение SOAP и отправляет его web-сервису.
7. Вскоре прокси-класс получает SOAP-ответ, преобразует таковой в соответствующий тип данных и возвращает его как обычный тип данных **.NET**.
8. Потребитель использует возвращенную ему информацию.

Разработка Web-сервиса с помощью текстового редактора

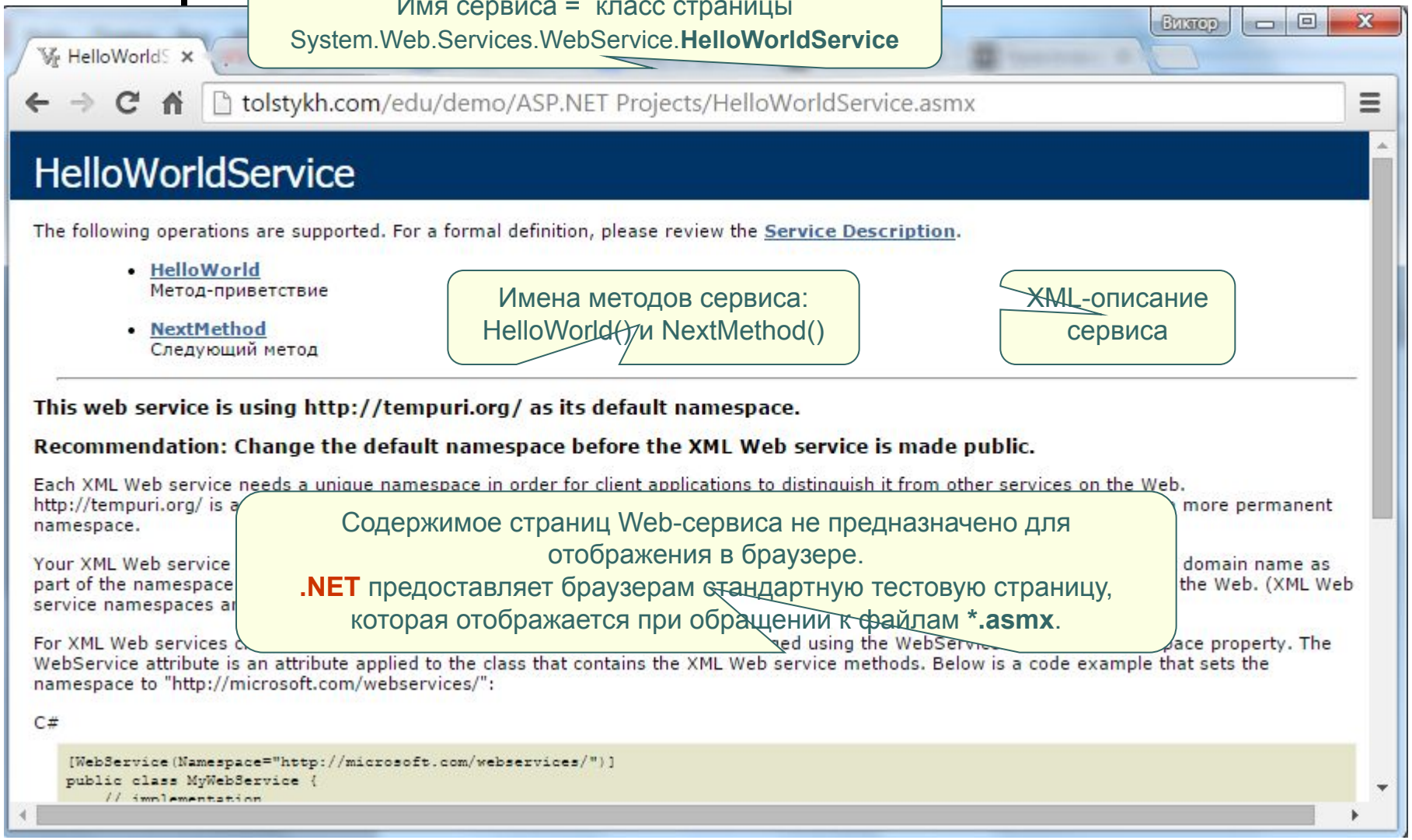
Код Web-сервиса HelloWorldService на языке C#:



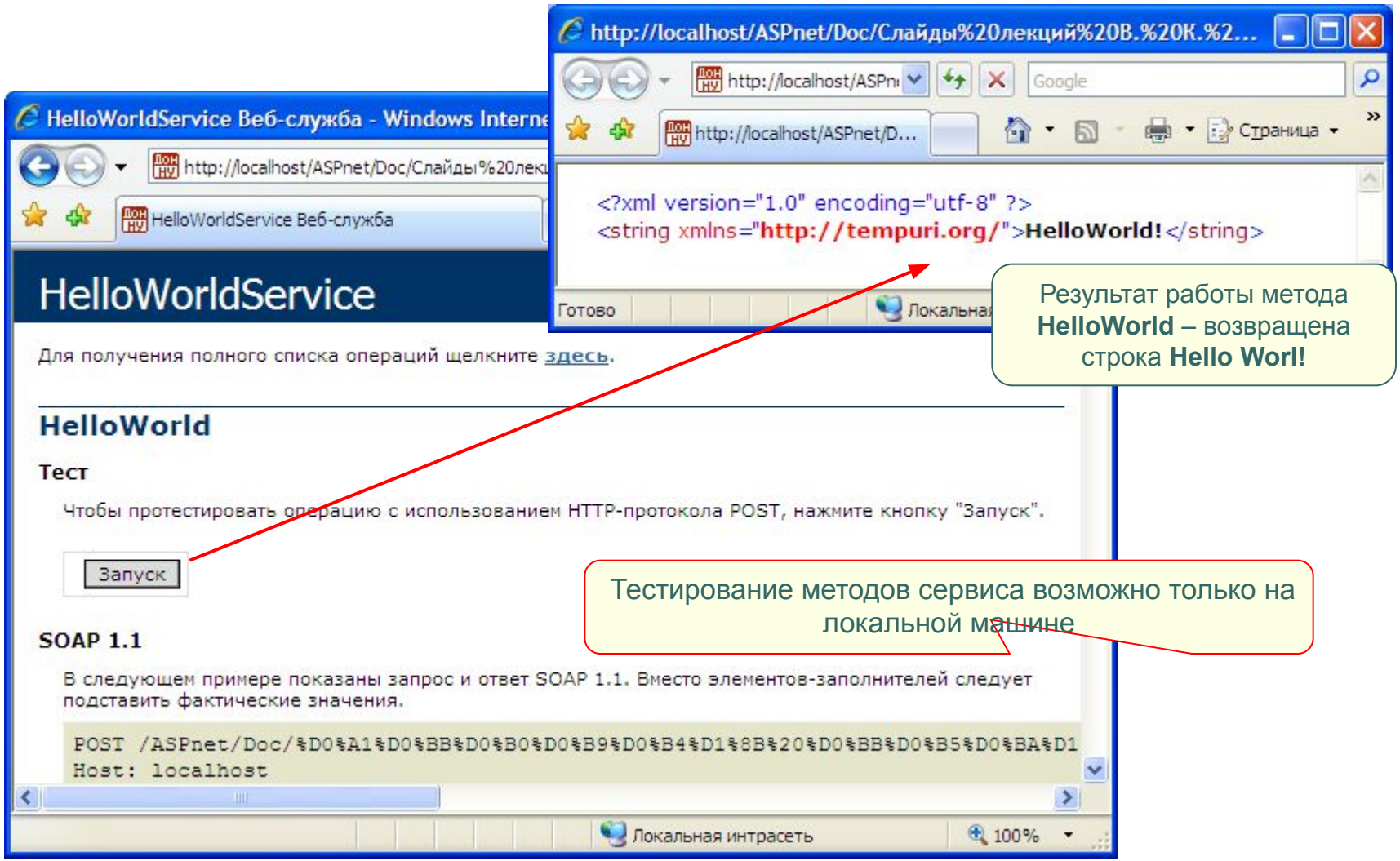
Файл сервиса **HelloWorldService.asmx** помещаем в область видимости **IIS**

Просмотр Web-сервиса

См. пример **HelloWorldService.asmx**



XML-описание и тестирование метода **HelloWorld**



Использование свойств атрибута **WebMethod**

Можно удаленно вызывать через сеть только те методы сервиса, которые имеют атрибут WebMethod. Данный атрибут имеет следующие свойства:

- BufferResponse - по умолчанию **true,** т.е. – ответ Web-службы перед отправкой на запрос клиента полностью формируется в буфере.
- CacheDuration - определяет промежуток времени в секундах на который кэшируется Web-служба. По умолчанию он равен 0, т.е. кэширование отключено;
- Description - описание метода, которое выводится на страницу службы под ссылкой на страницу метода;
- EnableSession - включает поддержку сеансов: **[WebMethod(EnableSession=true)]**. По умолчанию поддержка сеансов в методах Web-служб отключена, т.е. после выполнения каждого метода связь со службой разрывается;
- MessageName - идентифицирует перегруженные методы с помощью псевдонимов;
- TransactionOption - позволяет создать новую транзакцию. (см. Понятие транзакции, слайды автора)

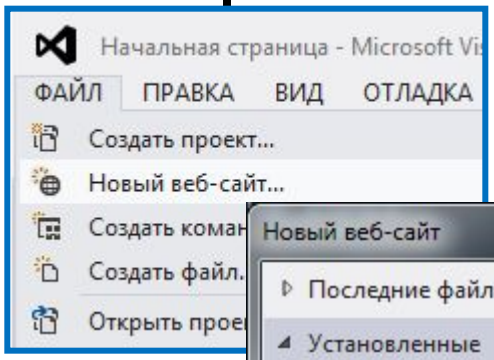
Свойство **EnableSession** атрибута WebMethod

Включает состояние сеанса для метода Web-сервиса. По умолчанию EnableSession=**false**. Во время сеанса после ответа метода связь с ним не разрывается, поэтому не требуются сетевые подключения при последующих обращениях к методу. Не используйте сеансы где в них нет острой необходимости.

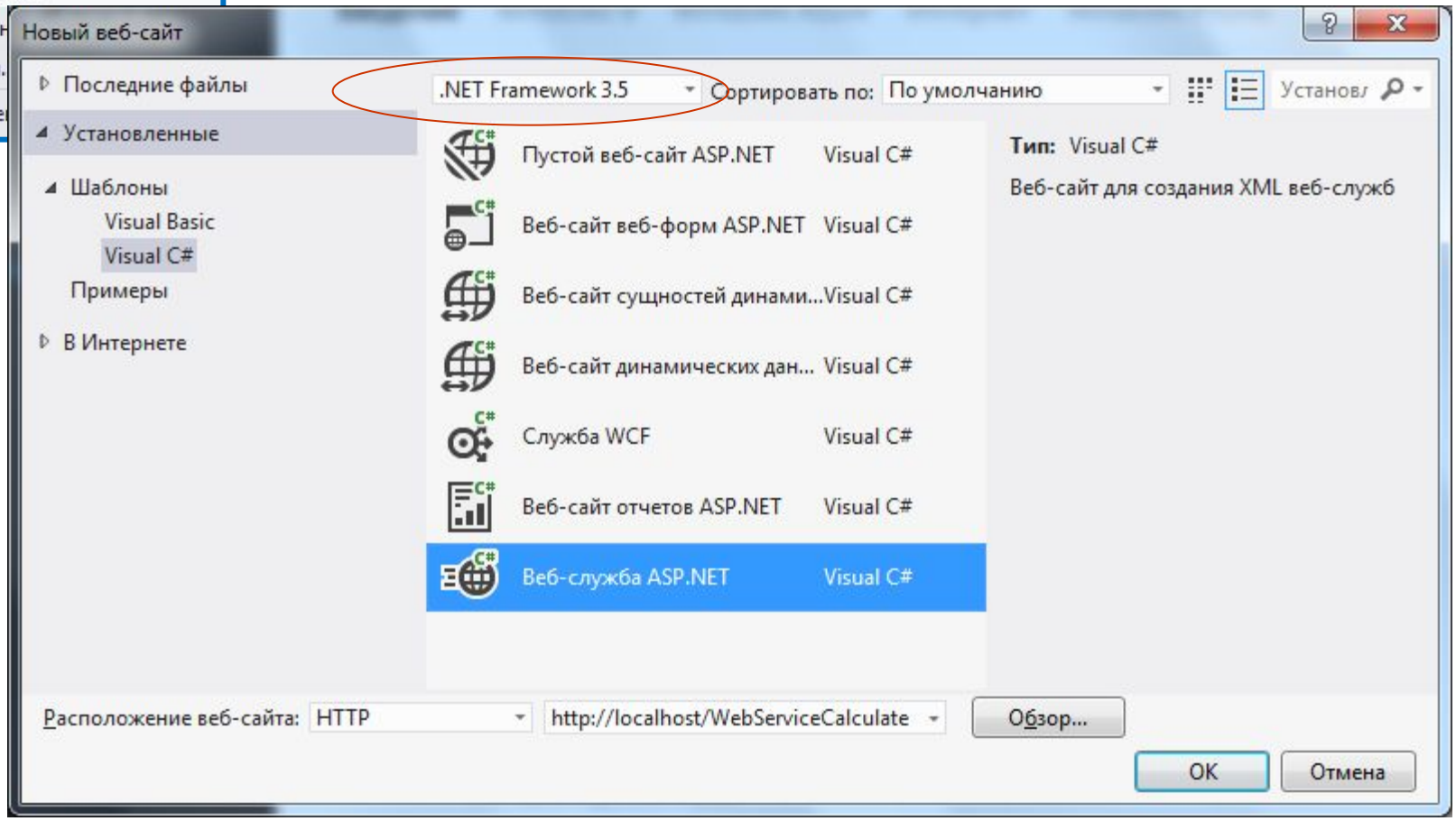
Пример счётчика доступа к методу ***SessionHitCounter*** Web-сервиса за один сеанс:

```
<%@ WebService Language="C#" Class="Util" %>
using System.Web.Services;
public class Util: WebService {
  [ WebMethod(Description="Per session Hit Counter",EnableSession=true)]
  public int SessionHitCounter() {
    if (Session["HitCounter"] == null) {
       Session["HitCounter"] = 1;
    } else {
       Session["HitCounter"] = ((int) Session["HitCounter"]) + 1;
            } return ((int) Session["HitCounter"]);
} }
```

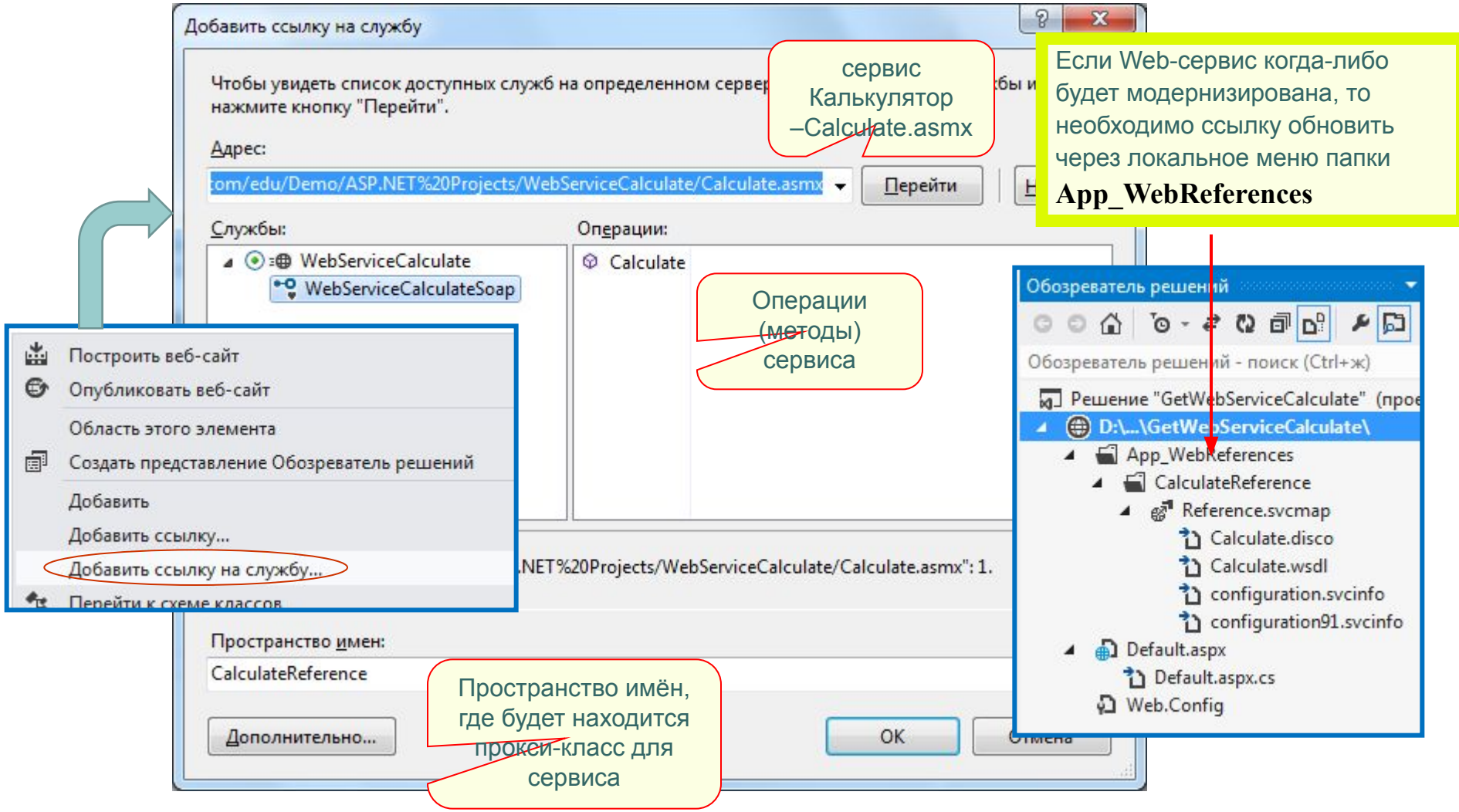
Создание Web-сервиса в среде Visual Studio .NET 2012



Смотрите пример на слайдах - **Web-служба Калькулятор**



Построение прокси-класса для подключения к Web-сервису



Вызов операций (методов) сервиса

Идентификатор экземпляра прокси-класса Web-сервиса

```
//Создание экземпляра прокси для связи с Web-сервисом:
  CalculateReference.WebServiceCalculateSoapClient wsCalculate =
    new CalculateReference.WebServiceCalculateSoapClient();
//Вызов метода Calculate в Web-сервисе:
  TextBox3.Text = wsCalculate.Calculate(n1,n2,signum);
```

Смотрите пример работы приложения – **GetWebServiceCalculate**

Приложения, требующие длительного времени обработки (Web-службы, БД), должны организовываться в виде ***асинхронных*** страниц. В момент ожидания ответа от других серверов они не занимают пул приложений и не прерывают связь с клиентом с сообщением **503 «Server too busy»**.

В директиве `Page` файла **.aspx** должно быть `Async="true"`.

Вызов операций сервиса в приложении PHP

Для использования SOAP в php необходимо подключить модуль SOAP – дописать в **php.ini**: `extension=php_soap.dll.` Не забудьте перезапустить сервер, если php у вас установлен как модуль.

Пример (см. Хабрахабр):

```
<?php
// Создание SOAP-клиента по WSDL-документу
$client = new SoapClient("http://...Service.wsdl");

// Посылка SOAP-запроса и получение результата от метода getRate()
$result = $client->getRate("us", "russia");

echo 'Текущий курс доллара: ', $result, ' рублей';
?>
```

...проблемы веб-сервисов

К сожалению, за великий потенциал веб-сервисов приходится платить определенную цену:

- Использование XML в качестве формата передачи данных приводит к тому, что ваши сообщения могут быть очень большими. Хотя, при сеансовой работе скорость доступа к операциям сервиса может быть даже быстрее чем при работе со сжатыми форматами REST.
- Так как мы используем удаленные компьютеры для выполнения определенных функций, мы полностью полагаемся на Интернет, что создает много ненадежных звеньев в цепи между нашим приложением, веб-сервером и веб-сервисом.
- Уровень доверия к качеству и надёжности, лицензированию и оплате услуг для «чужих» удалённых разработчиков может подвергаться сомнению.