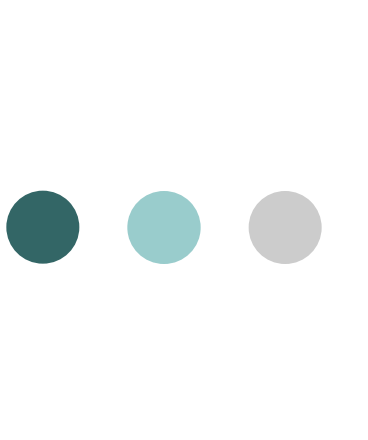


Web-сервисы

Понятие сервиса

- Сервисы в Web являются наследниками COM-объектов в обычных и распределенных приложениях.
- COM-объекты предоставляли любому приложению доступ к своим функциональным возможностям при помощи стандартизованных интерфейсов.
- Web-сервисы также обладают некоторой функциональностью, связываются с клиентской частью, передают результаты работы и получают команды от клиента по протоколу HTTP.

Понятие сервиса

- Web-сервис предоставляет клиентским приложениям стандартизованные интерфейсы.
- Использование независимого от платформы протокола HTTP позволяет создавать один Web-сервис и множество клиентских приложений на самых различных операционных системах.
- Web-сервисы могут принимать запросы, основанные на метолах GET и POST, а также запросы на языке SOAP.

Функционирование сервиса

- Сервис работает на сервере.
- Сервис принимает запросы, отправленные по протоколу HTTP.
- Сервис передает клиентскому приложению информацию о выполняемых сервисом действиях на языке XML.
- Клиентское приложение анализирует полученную информацию о функциональности сервиса и вызывает одну из его функций, пользуясь зыком XML и протоколом HTTP.

Функционирование сервиса

- Сервис выполняет затребованную клиентом функцию, а результаты ее выполнения передает клиенту.

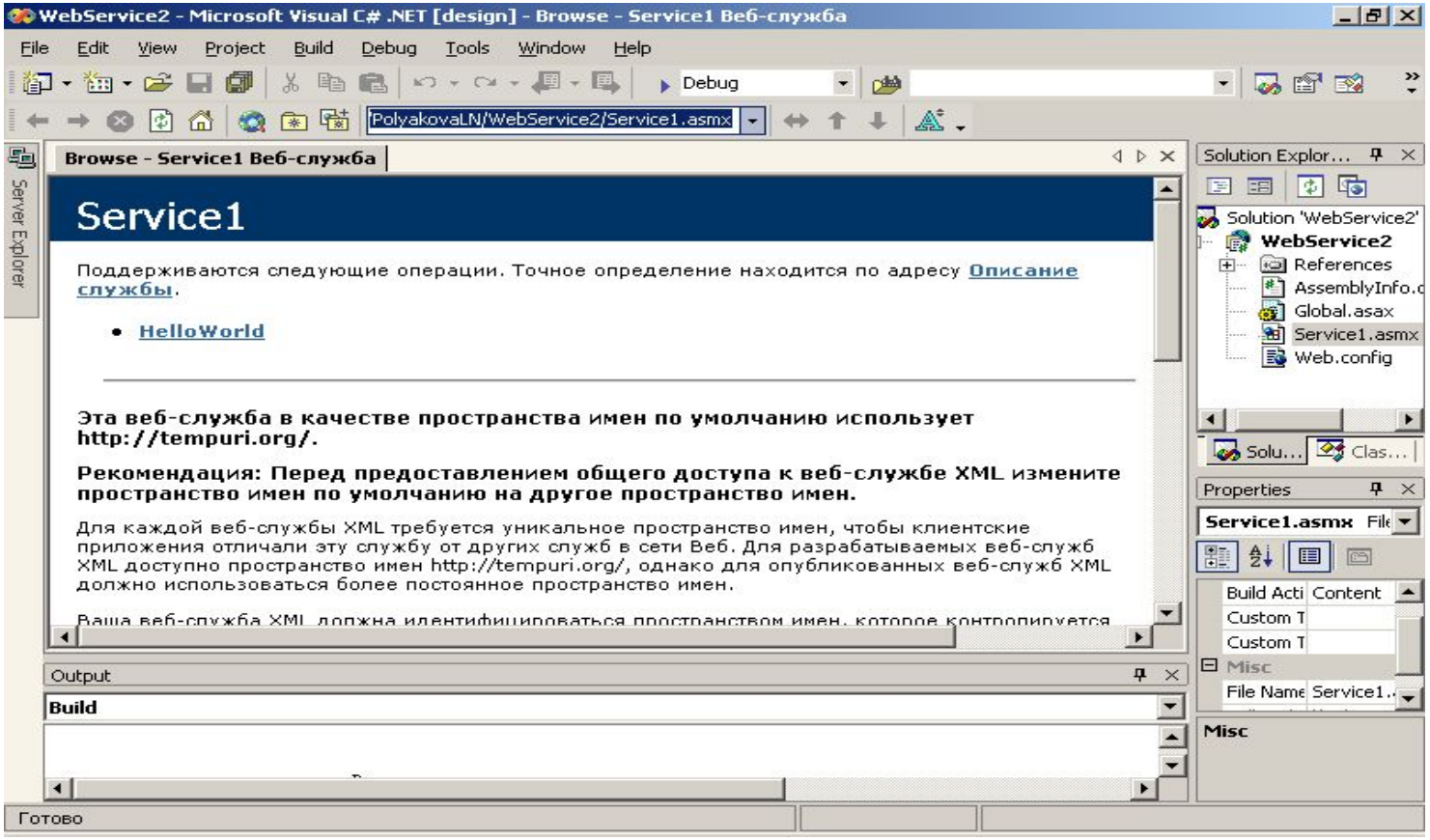
- Использование в качестве способа общения между сервисом и клиентом языка XML и общепринятого протокола HTTP позволяет создавать клиентские приложения практически на любой компьютерной платформе.

Слайд 6. Создание сервисов

Проект WebService1

```
[WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }
[WebMethod]
    public bool Validate(string t)
    {
        if (t.Equals(“text"))
            return true;
        else
            return false;
```

Слайд 7. Просмотр сервиса в браузере по адресу http://adm-119-1/PolyakovaLN/WebService2/Service1.asmx



- Результатом работы созданного Web-сервиса является XML-документ.
- Клиентское приложение получит этот XML-документ , само обработает его и выдаст результат пользователю.
- Таким образом, был создан сервис, который сам документирует себя, принимает данные тремя различными способами по протоколу HTTP и возвращает результат работы на языке XML.

Самодокументирование Web-сервисов

- При создании Web-сервиса компилятор одновременно создает WSDL-файл, в котором описана структура сервиса. Именно на основе этого файла клиенты получают информацию о функциональности сервиса.
- Файл WSDL (Web Service Descriptor Language) является обычным XML-файлом и доступ к нему можно получить из браузера.

Слайд 10. Использование сервисов

Проект WebApplication1

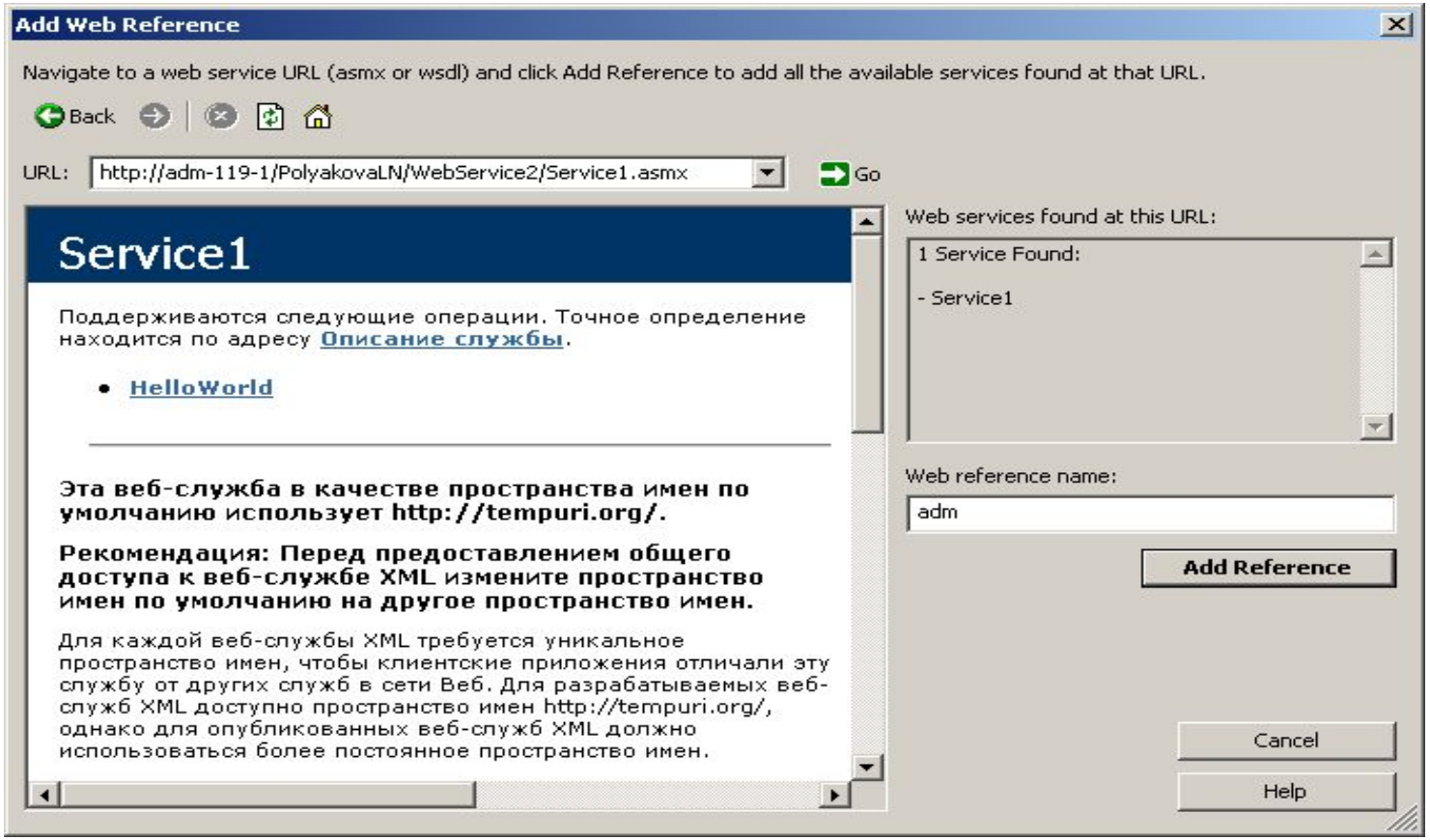
1. Подключить сервис в Solution Explorer

ПК ADD Add Web References, указав адрес сервиса

http://adm-119-1/PolyakovaLN/WebService1/Service1.asmx.

И присвоив ему имя adm.

Слайд 11. Подключение Web-сервиса в Web-приложении



Слайд 12. Подключение Web-сервисов в программном коде

2. В объявлении

```
WebApplication1.adm.Service1 my=new
    WebService1.adm.Service1();
```

3. В методе Page_Load установка пользовательских реквизитов

```
    my.PreAuthenticate = true;

    my.Credentials = System.Net.CredentialCache.
    DefaultCredentials;
```

Или (пример открытой аутентификации)

```
    my.Credentials=new
    NetworkCredential("PolyakovaLN",“password");
```

Слайд 13. Аутентификация Web-сервиса

Аутентификация — это возможность доказать, что объект — например, пользователь или компьютер — является именно тем, за кого он себя выдает. Можно подтвердить подлинность, если иметь в наличии объект, который предоставляет данные для аутентификации. Данные аутентификации часто задаются в форме имени пользователя и пароля. Необходимо отметить, что некоторые протоколы аутентификации более защищены, чем другие. Поэтому нужно точно убедиться, что идентификационные данные действительно получены от корректного пользователя, а не от хакера.

Слайд 14. Протоколы аутентификации

Web-сервис, выполняющийся на верхнем уровне *IIS*, имеет большое число доступных ему протоколов аутентификации.

Наиболее значимые:

- *анонимная аутентификация (Anonymous authentication);*
- *открытая аутентификация (Basic authentication);*
- *краткая аутентификация (Digest authentication);*
- *Windows-аутентификация;*
- *аутентификация, основанная на сертификатах;*
- *аутентификация, основанная на формах;*

Слайд 15. Открытая аутентификация

В серверах *IIS версий 5 и 6* учетные записи, предназначенные для открытой аутентификации, должны быть допустимыми учетными записями Windows. Однако когда открытая аутентификация применяется в ASP.NET или, допустим, в Web-сервисе, написанном с помощью ASP.NET, то можно использовать подход, применяемый в базах данных, для определения корректности параметров идентификации. Можно установить открытую аутентификацию и с помощью средств администрирования IIS. Следует отметить, что открытая аутентификация чрезвычайно ***ненадежна***, особенно без применения технологий защиты канала связи SSL/TLS.

Слайд 16. Создание аутентифицированной связи

Для создания аутентифицированной связи с Web-сервисом можно использовать такой код (этот код будет работать для открытой, краткой и Windows-аутентификаций):

```
using System;
using System.Net;
using System.Web.Services.Protocols;

ClientApp.localhostService s=new
ClientApp.localhostService();
s.Credentials=new NetworkCredential(username,
password, domain);
string shipped = s.GetShippingStatus(" 10001");
```

Слайд 17. Windows-аутентификация

Имя вызываемого пользователя можно получить, используя в Web-сервисе следующий код:

```
System.Security.Principal.WindowsIdentity wi =
    WindowsIdentity.GetCurrent();
string name = wi.Name;
```

В приложении имя пользователя можно прочитать:

```
this.Response.Write("name="+name+"<br>");
```

В результате получим:

name=NT AUTHORITY\NETWORK SERVICE

Слайд 18. Использование Web-сервиса в приложении

```
private void Button1_Click(object sender,
    System.EventArgs e)
     {
          this.Label1.Text=my.HelloWorld();

           if(my.Validate(TextBox1.Text))
            this.Label1.Text="YES";
          else
            this.Label1.Text="NO";
     }
```

Слайд 19. Параметры безопасности Web-сервисов

При создании корпоративного XML Web-сервиса обычно требуется обеспечить проверку прав пользователя. Существует два набора параметров для обеспечения безопасности в XML Web-сервисах.

- Параметры встроенной безопасности IIS. Их достоинство заключается в том, что не требуется дополнительная база данных пользователей, однако создание записи доменного пользователя для каждого клиента вряд ли можно считать идеальным решением для приложений Internet.
- Второй набор параметров предусматривает один из видов специальной аутентификации:

Слайд 20. Специальная аутентификация

- Прием и пересылка имени и пароля пользователя как параметров вызова метода;
- Создание некоторого метода Login, который вызывается раньше других методов. В дальнейшем для проверки правомочности пользователя достаточно cookie-файлов;
- Применение заголовков или тела SOAP-сообщений для хранения пользовательских реквизитов;
- Создание специального заголовка HTTP для передачи входных данных.

Слайд 21. Прием и пересылка имени и пароля пользователя как параметров вызова метода

- Первый из перечисленных параметров аутентификации самый простой и меньше всех подвержен сбоям по причине неудачной конфигурации клиента. При вызове метода имя и пароль пользователя передаются в виде параметров. Далее они сравниваются со сведениями, хранящимися в базе данных.

Слайд 22. Создание таблицы в среде MS SQL Server

```
create table tab_user
(id_user int    identity(1,1) primary key,
 name_user   varchar(20),
 login_user    varchar(20),
 pas_user      varchar(10))

insert into tab_user
    values('Полякова','PolyakovaLN',
‘password')
```

Слайд 23. Создание хранимой процедуры в среде MS SQL Server

```
create proc proc_user
@us varchar(20),
@pw varchar(10),
@fam varchar(20) output
as
select @fam=name_user
from tab_user
Where login_user=@us and pas_user=@pw
```

Слайд 24. Проверка процедуры

```
declare @t varchar(20)
exec proc_user 'PolyakovaLN',
‘password',
@t output

select @t
```

Слайд 25. Использование хранимой процедуры в Web-сервисе или приложении

```
public class Form1 : System.Windows.Forms.Form
    {
     private System.Data.SqlClient.SqlConnection
    sqlConnection1;
     private System.Data.SqlClient.SqlCommand
    sqlCommand1;

…
```

Слайд 26.

```
private void InitializeComponent()
    {
        this.sqlConnection1 = new
    System.Data.SqlClient.SqlConnection
    ();
        this.sqlCommand1 = new
    System.Data.SqlClient.SqlCommand()
    ;
```

Слайд 27. Настройка строки соединения

```
//
// sqlConnection1
//
this.sqlConnection1.ConnectionString = “…";
```

Слайд 28. Настройка команды

```
//
// sqlCommand1
//
this.sqlCommand1.CommandText = "proc_user";
this.sqlCommand1.CommandType =
    System.Data.CommandType.StoredProcedure;
this.sqlCommand1.Connection = this.sqlConnection1;
this.sqlCommand1.Parameters.Add(new
    System.Data.SqlClient.SqlParameter("@us",
    System.Data.SqlDbType.VarChar, 20));
```

Слайд 29.

```
this.sqlCommand1.Parameters.Add(new
    System.Data.SqlClient.SqlParameter("@pw",
    System.Data.SqlDbType.VarChar, 10));

this.sqlCommand1.Parameters.Add(new
    System.Data.SqlClient.SqlParameter("@fam",
    System.Data.SqlDbType.VarChar, 20,
    System.Data.ParameterDirection.Output, false,
    ((System.Byte)(0)), ((System.Byte)(0)), "",
    System.Data.DataRowVersion.Current, null));
```

Слайд 30. Вызов процедуры из Web-сервиса

```
try
{
    this.sqlConnection1.Open();
    this.sqlCommand1.Parameters["@us"].Value="PolyakovaLN";
    this.sqlCommand1.Parameters["@pw"].Value="passworw";
    this.sqlCommand1.ExecuteNonQuery();

    string
    res=this.sqlCommand1.Parameters["@fam"].Value.ToString();
```

Слайд 31.

```
if (res!="")
    this.label1.Text=res;
else
    this.label1.Text="Доступа нет";
}
```

Слайд 32.

```
catch(Exception ex)
{
      this.label1.Text=ex.Message;
}
```

Слайд 33. Пример сервисов для получения различной информации

```
[WebMethod]
public string method1()
{
    return "User"+this.User.ToString();
}
[WebMethod]
public string method2()
{
    return "Context
    Server"+this.Context.Server.ToString();
}
[WebMethod]
public string method3()
{
    return "Server"+this.Server.ToString();
}
```

Слайд 34. Пример подключения сервисов в приложении

```
this.Response.Write(my.method1()+"<br>");
this.Response.Write(my.method2()+"<br>");
this.Response.Write(my.method3()+"<br>");

this.Response.Write(my.Url.ToString()+"<br>");
this.Response.Write(my.UserAgent.ToString()+"<br>");

this.Response.Write(my.ClientCertificates.ToString()+
"<br>");
```

Слайд 35. Результат работы приложения

- UserSystem.Security.Principal.WindowsPrincipal
- Context ServerSystem.Web.HttpServerUtility
- ServerSystem.Web.HttpServerUtility
- http://adm-119-1/PolyakovaLN/WebService2/Service1.asmx
- Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol 1.1.4322.2300)
- System.Security.Cryptography.X509Certificates.X509CertificateCollection

1 вариант защиты сервиса–файл Web.config

```
<authentication mode="Windows" />
  <identity impersonate ="true" />


  <authorization>
   <deny users ="3CASE\PolyakovaLN" />
   <allow users="*" />
</authorization>
```

2 вариант защиты метода в сервисе

```
[WebMethod]
    public string method1 ()
    {
    WindowsIdentity wi=
    WindowsIdentity.GetCurrent();
    if (wi.Name=="3CASE\\YakimkinVV")  return
    "DENY";
    return wi.Name;
    }
```

Самодокументирование сервисов