

Kapitola 5

WCF, webové služby a mezidoménové zásady

Silverlight 2 přináší obrovské množství nástrojů, s jejichž pomocí lze vytvářet propracovaná uživatelská rozhraní. Může také využívat různé služby v síti nebo v Internetu. Díky schopnosti komunikovat s různými typy webových služeb několika způsoby (například prostřednictvím služeb RESTful, poskytování obsahu informačních kanálů a služeb SOAP) mohou aplikace Silverlightu využívat funkce těchto služeb. Klienti Silverlightu přináší uživatelům silný zážitek a díky schopnosti komunikovat s různými službami je Silverlight skvělou volbou pro interaktivní aplikace a aplikace založené na službách.

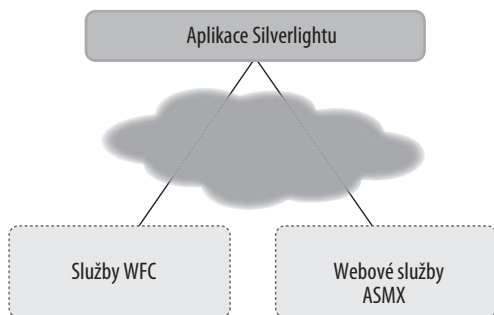
V této kapitole si ukážeme, jak lze vytvářet webové služby ASMX a WCF (Windows Communication Foundation) a jak s nimi komunikovat z klientských aplikací Silverlightu. Nejdříve vytvoříte službu ASMX založenou na SOAP, kterou využívá aplikace Silverlightu. Seznámíte se s několika aspekty komunikace založené na službách a dozvíte se, jak nastavit asynchronní komunikaci z klientů Silverlightu do služeb ASMX a WCF, jak implementovat obsluhy, využívat výsledky a zpracovávat výjimky. V této kapitole také probereme, jak mohou klienti Silverlightu vytvářet a využívat služby WCF a ukážeme si několik způsobů, jakými lze data předávat do služeb a ze služeb. Na obrázku 5.1 vidíte služby a jak s nimi Silverlight může komunikovat.

Všechny služby by se měly chránit proti tomu, aby je využívali klienti, kteří nemají oprávnění. Mezidoménové zásady řeší tuto úroveň ochrany pro služby ASMX a WCF, ale také pro služby, jež se samy nepopisují, včetně služeb založených na REST, RSS a Plain Old XML (POX). Flash i Silverlight nabízejí schéma souborů XML, jež lze navrhnout tak, aby byl povolen nebo zakázán přístup klientů k těmto typům služeb. V této kapitole se dozvíte, jak tyto soubory fungují a naleznete zde několik tipů k jejich vytváření.

Webové služby ASMX

Interakce s webovými službami ASMX je v současné době běžná. Umíte-li vytvářet a využívat webové služby ASMX, nemělo by být obtížné vytvořit webovou službu ASMX, kterou může

využívat klientská aplikace Silverlightu. Pokud byste například chtěli využívat webovou službu ASMX z klienta ASP.NET, bylo by nutné přidat odkaz na webovou službu ASMX, vytvořit pro ni proxy a synchronně nebo asynchronně volat službu. Jediný rozdíl spočívá v tom, že klient Silverlightu může volat webovou službu pouze asynchronně.



Obrázek 5.1 Webové služby a Silverlight



Typy služeb, jež lze využívat z klientských aplikací Silverlightu, mají určitá omezení. Například služby WCF musí používat vazbu `basicHttpBinding` a služby podporující vlastní hlavičky SOAP nejsou podporovány.

Chcete-li využívat webovou službu ASMX, je nutné ji nejdříve vytvořit (pokud ještě neexistuje). Poté definujete účel služby (vystavované akce nebo metody). Klientská aplikace Silverlightu musí odkazovat na službu. Jakmile bude aplikace Silverlightu obsahovat odkaz na službu, vytvoří proxy pro službu a obsluhu pro asynchronní volání jejích metod. Nakonec může aplikace Silverlightu vyvolat službu a využít její výsledky.

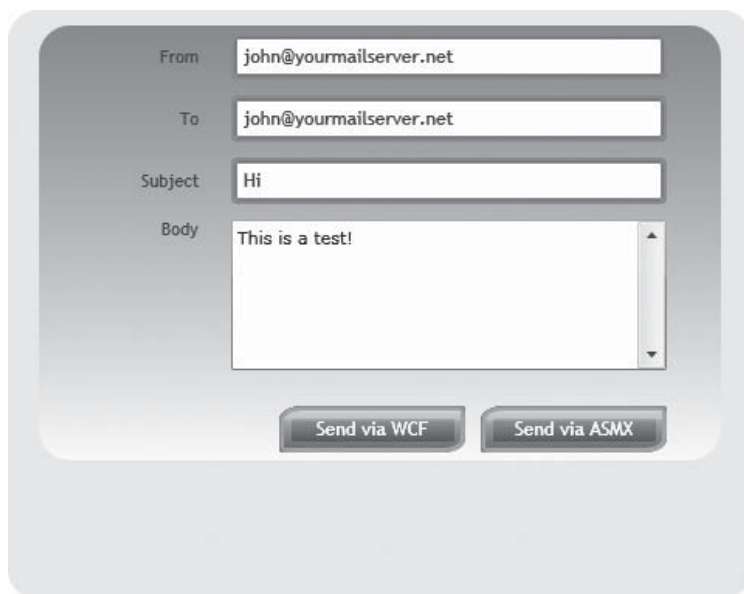
Vytvoření webové služby ASMX

Vývojáři budou obecně komunikovat buď s existující webovou službou, nebo vytvoří webovou službu pro vlastní účely. V této kapitole si ukážeme několik variací webových služeb a způsoby, jak s nimi lze komunikovat z klientů Silverlightu. Nejdříve se dozvíte, co potřebujete k vytvoření webové služby ASMX a ke komunikaci z klientské aplikace Silverlightu.

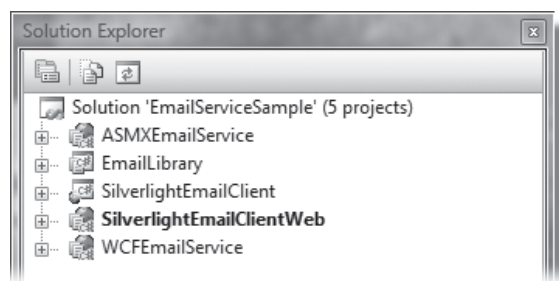
Zkušební aplikace Silverlightu, kterou vytvoříme, umožní uživateli odesílat e-mailové zprávy. Protože klient Silverlightu nemá přístup ke jmennému prostoru `System.Net.Mail`, musí komunikovat se serverem, jenž mu umožní odesílat e-mailové zprávy. Klient by jinak mohl vyvolat výchozí poštovní aplikaci a použít ji k odeslání zprávy. Řešení založené na serveru je však často výhodnější, protože umožňuje mnohem více úprav.

V aplikaci Silverlightu na obrázku 5.2 může uživatel jednoduše zadat informace pro e-mailovou zprávu a klikne na tlačítko Send via ASMX, aplikace Silverlightu vytvoří proxy pro webovou službu ASMX, přidá obsluhu události pro zpracování vráceného volání webové služby ASMX

a asynchronně vyvolá webovou službu. Webové službě jsou předány informace z klientské aplikace Silverlightu, jež pak služba použije k odeslání e-mailu příjemci (příjemcům).



Obrázek 5.2 Poštovní klient Silverlightu



Obrázek 5.3 Řešení EmailServiceSample

Začněme příkladem

Řešení příkladu se nachází ve složce s kódem pro kapitolu 5. Řešení má název `EmailServiceSample` a obsahuje pět projektů, jejichž názvy vidíte na obrázku 5.3. Projekt `SilverlightEmailClient` je projekt Silverlightu s klientem, který je znázorněn na obrázku 5.2. Projekt `SilverlightEmailClientWeb` je webová aplikace, jež hostí aplikaci Silverlightu; použijete ji k otestování klienta Silverlightu.

Projekt `ASMXEmailService` obsahuje webovou službu `ASMX`, jež bude volána z projektu `SilverlightEmailClient`. Projekt `EmailLibrary` obsahuje jednoduchou třídu `.NET` s názvem `Mail`, kterou zkušební kód používá k odeslání e-mailové zprávy prostřednictvím třídy `MailMessage` jmenného prostoru `System.Net.Mail`.

Posledním projektem je `WCFEmailService`. Tento projekt obsahuje webovou službu `WCF`, jež nabízí stejnou funkčnost jako webová služba `ASMXEmailService`, ale místo `ASMX` využívá `WCF`. Projekt `WCFEmailService` si popíšeme dále v této kapitole.

Vytváření webové služby ASMX

Projekt `ASMXEmailService` je projekt webové aplikace `ASP.NET`, jenž pro služby `ASMX` využívá `IIS`. Mohlo by se jednat i o webový server, pro tento příklad je však využít webový server `Cassini`. Projekt obsahuje jednu webovou službu `ASMX` s názvem `EmailService.asmx`. Příklad 5.1 ukazuje obsah třídy `EmailService`, jež jednoduše definuje metodu webové služby `SendMessage` přijímající požadované informace o zprávě. Webová služba `EmailService` definuje svůj jmenný prostor jako `http://www.silverlight-data.com` a přidává atribut `WebMethod` do metody `SendMessage`. Tento atribut deklaruje, že metoda bude přístupná z klientů, kteří odkazují na webovou službu `EmailService`.



Třída `EmailLibrary.Mail` je v samostatném projektu, protože bude volána také webovou službou `WCF` dále v této kapitole.

Příklad 5.1 Třída `EmailService`

```

using System.Web.Services;
using EmailLibrary;

namespace ASMXEmailService
{
    [WebService(Namespace = "http://www.silverlight-data.com/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    public class EmailService : System.Web.Services.WebService
    {
        [WebMethod]
        public void SendMessage(string from, string to,
                               string subject, string body)
        {
            new Mail().SendMessage(from, to, subject, body);
        }
    }
}

```

```

}
VB Imports System.Web.Services
Imports EmailLibrary

Namespace ASMXEmailService
    <WebService(Namespace := "http://www.silverlight-data.com/"), _
        WebServiceBinding(ConformsTo := WsiProfiles.BasicProfile1_1), _
        System.ComponentModel.ToolboxItem(False)> _
    Public Class EmailService
        Inherits System.Web.Services.WebService
        <WebMethod> _
        Public Sub SendMailMessage(ByVal from As String, _
            ByVal [to] As String, ByVal subject As String, _
            ByVal body As String)
            CType(New Mail(), Mail).SendMessage(From, [to], subject, body)
        End Sub
    End Class
End Namespace

```



Všimněte si, že tento kód vyžaduje, aby byl port SMTP a hostitel příslušně nakonfigurován pro server SMTP, jenž bude použit. Změny na správná nastavení je nutné provést před spuštěním tohoto zkušebního kódu.

Metoda `SendMailMessage` volá metodu `SendMessage` třídy `EmailLibrary.Mail`. Příklad 5.2 ukazuje obsah třídy `Mail` a její metodu `SendMessage`, jež odesílá zprávu seznamu příjemců. V kódu chybí zpracování výjimek a určitá propracovanost, na to se však v této kapitole nezaměřujeme.

Příklad 5.2 Třída Mail

```

C# using System;
using System.Net.Mail;

namespace EmailLibrary
{
    public class Mail
    {
        private readonly char[] separator = { ';' };
        private readonly string host = "mail.yoursmtpserver.net";
        private readonly int port = 25;

        public void SendMessage(string from, string to,
            string subject, string body)
        {

```

```

using (MailMessage message =
new MailMessage { From = new MailAddress(from),
                  Body = body, Subject = subject })
{
    string[] addresses = to.Split(separator,
        StringSplitOptions.RemoveEmptyEntries);
    foreach (string t in addresses)
        message.To.Add(new MailAddress(t));
    SmtplibClient smtpClient = new SmtplibClient(host, port);
    smtpClient.Send(message);
}
}
}
}

```

VB Imports System

Imports System.Net.Mail

Namespace EmailLibrary

Public Class Mail

Private ReadOnly separator() As Char = { ";", "c" }

Private ReadOnly host As String = "mail.yourmailserver.net"

Private ReadOnly port As Integer = 25

Public Sub SendMessage(ByVal from As String, ByVal [to] As String, _
 ByVal subject As String, ByVal body As String)

Using message As MailMessage = New MailMessage _
 With {.From = New MailAddress(From), .Body = body, _
 .Subject = subject}

Dim addresses() As String = [to].Split(separator, _
 StringSplitOptions.RemoveEmptyEntries)

For Each t As String In addresses
 message.To.Add(New MailAddress(t))

Next t

Dim smtpClient As New SmtplibClient(host, port)
 smtpClient.Send(message)

End Using

End Sub

End Class

End Namespace

To je vše, co je třeba pro vytvoření webové služby ASMX, již může využívat aplikace Silverlightu. Dalším krokem je přidání odkazu na službu.

Odkazování na webovou službu ASMX

Než můžete začít službu používat, je nutné přidat do klientské aplikace odkaz na webovou službu ASMX. Přejděte do průzkumníka projektu (Project Explorer), klikněte pravým tlačítkem myši na uzel Service Reference pro projekt `SilverlightEmailClient` a v místní nabídce zvolte příkaz Add Service Reference. Otevře se okno, v němž lze vyhledat službu dostupnou na počítači. Případně lze zadat adresu URL na konkrétní webovou službu. Protože služba `EmailService.asmx` je na stejném počítači jako toto řešení, můžete kliknout na tlačítko Discover. Po chvíli klientská aplikace Silverlightu vyhledá službu `EmailService.asmx` a vy ji můžete vybrat. V tomto okně bude vyhledána jakákoli služba, jež vystavuje jazyk WSDL (Web Services Description Language), díky němuž ji lze najít. Webové služby WCF i ASMX podporují SOAP 1.1 a lze je najít prostřednictvím tohoto okna.

Jako odkaz na službu z aplikace Silverlightu lze přidat jakoukoli službu SOAP, kterou lze nalézt, a jež podporuje základní profil SOAP 1.1. Vyhledatelné služby podporují WSDL. Jak si ukážeme v této kapitole, patří sem služby WCF i ASMX. Přístup ke službám, jež se samy nepopisují (jako je například REST, PX a RSS), lze získat prostřednictvím `WebClient` a `HttpWebRequest`, což si ukážeme v dalších kapitolách.



Všimněte si, že adresa webové služby na obrázku 5.4 ukazuje, že využívá webový server Cassini na portu 8242. Cassini je Vývojový webový server ASP.NET. Vytváříte-li webovou aplikaci s pomocí serveru Cassini, může se číslo portu lišit.

`ASMXEmailService` je webová aplikace, jež běží na serveru Cassini a automaticky přiřazuje porty. Pokud byla služba vytvořena pro webový server, běžela by na IIS a ne na Cassini.

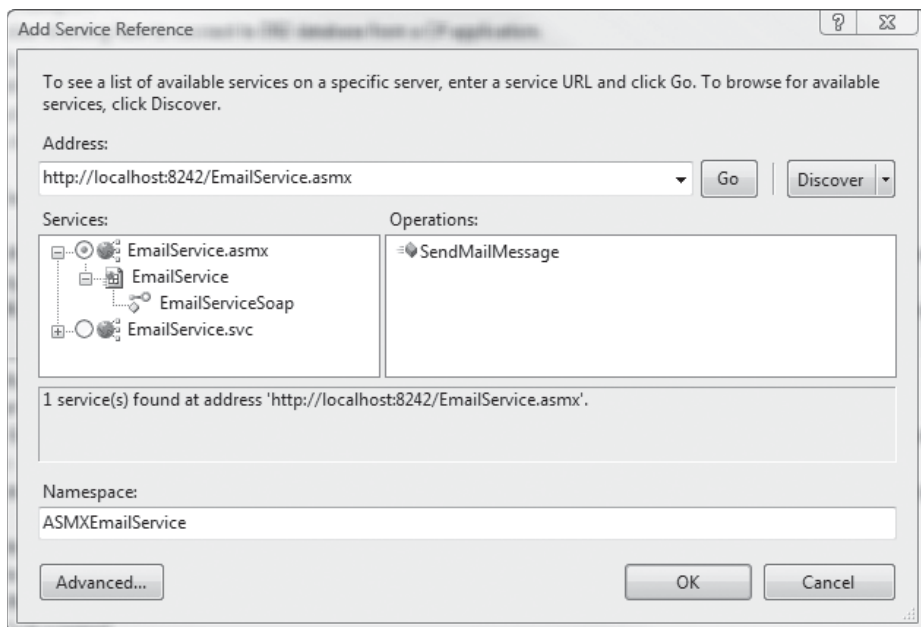
Na obrázku 5.4 je vybraná služba `EmailService.asmx` a v rozbaleném seznamu jejích služeb a metod je uvedena webová metoda `SendMailMessage`, která byla právě vytvořena. Služba je přejmenována na `ASMXEmailService`, což bude jmenný prostor pro třídu proxy, kterou vytvoříme pro vyvolání webové služby ASMX.

Vrácení kolekce `ObservableCollection<T>`

Tlačítko Advanced (v levé dolní části na obrázku 5.4) umožňuje upravit odkaz na webovou službu. Obrázek 5.5 ukazuje, jak lze díky této možnosti upravit typ `Collection`, který použijeme pro vrácené hodnoty založené na kolekci. To znamená, že pokud webová služba vrátí `List<T>`, můžete nakonfigurovat odkaz na službu tak, aby byl `List<T>` automaticky převeden na `ObservableCollection<T>`. Nakonfigurovat lze několik možností, včetně `Array`, `List`, `Collection` a `ObservableCollection`. To je důležité, protože se tak usnadňuje využívání služeb, jež vracejí seznamy entit s funkcemi datových vazeb kolekce `ObservableCollection`.

Využívání webové služby ASMX

Jakmile má aplikace Silverlightu odkaz na webovou službu ASMX, může ji vyvolat prostřednictvím objektu proxy. Pro vyvolání webové služby je nutné přidat odkaz na službu, vytvořit instanci objektu proxy, přidat obsluhu události pro událost dokončení a asynchronně vyvolat webovou službu.



Obrázek 5.4 Přidání odkazu na e-mailovou službu ASMX

Znovu ty vazby

Aplikace Silverlightu obsahuje čtyři ovládací prvky `TextBox`, jež představují adresu To (komu), adresu From (od), Subject (předmět) a Body (text zprávy). Adresa To může obsahovat sadu e-mailových adres oddělených středníkem.

Výchozí hodnoty lze ručně načíst do ovládacích prvků `TextBox` nastavením jednotlivých hodnot vlastnosti `Text` prvku `TextBox`. Poté, co uživatel zadá hodnoty, můžete tyto hodnoty získat z ovládacích prvků `TextBox` a předat je do webové metody ve webové službě ručním získáním hodnot z ovládacích prvků a odesláním do místních proměnných.

Ačkoli tento postup funguje, další možností je využití techniky vazeb založených na XAML. Třída `MessageInfo` je v kódu příkladu uvedena za účelem uložení hodnot z těchto polí a působí jako prostředník pro ovládací prvky `TextBox`. Příklad 5.3 ukazuje kód pro třídu `MessageInfo`. Je vytvořena instance třídy `MessageInfo` a výchozí hodnoty jsou inicializovány v konstruktoru

ovládacího prvku `EmailClient` Silverlightu. Příklad 5.4 ukazuje část kódu XAML ovládacího prvku a jeho vazeb.

Příklad 5.3 Třída `MessageInfo`

```
C# public class MessageInfo
{
    public string FromAddress { get; set; }
    public string ToAddress { get; set; }
    public string Subject { get; set; }
    public string Body { get; set; }

    public string BodyWithDateTag
    {
        get { return string.Format("{0}\n\nSent on: {1}",
            this.Body, DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss")); }
    }
}
```

```
VB Public Class MessageInfo
    Private privateFromAddress As String
    Public Property FromAddress () As String
        Get
            Return privateFromAddress
        End Get
        Set(ByVal value As String)
            privateFromAddress = value
        End Set
    End Property
    Private privateToAddress As String
    Public Property ToAddress() As String
        Get
            Return privateToAddress
        End Get
        Set(ByVal value As String)
            privateToAddress = value
        End Set
    End Property
    Private privateSubject As String
    Public Property Subject() As String
        Get
            Return privateSubject
        End Get
        Set(ByVal value As String)
            privateSubject = value
        End Set
    End Property
End Class
```

```

        End Set
    End Property
    Private privateBody As String
    Public Property Body() As String
        Get
            Return privateBody
        End Get
        Set(ByVal value As String)
            privateBody = value
        End Set
    End Property
    Public ReadOnly Property BodyWithDateTag() As String
        Get
            Return String.Format("{0}" & Constants.vbLf + Constants.vbLf & _
                "Sent on: {1}", Me.Body, _
                DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"))
        End Get
    End Property
End Class

```

Jakmile je instance vytvořena, je nastavena na vlastnost `DataContext` panelu rozložení `Grid`, jenž obsahuje čtyři ovládací prvky `TextBox`. Vlastnost `Text` každého ovládacího prvku `TextBox` je svázána s odpovídající vlastností instance `MessageInfo` s využitím vazby `TwoWay`. Díky tomu není nutné ručně odesílat hodnoty do ovládacích prvků a ručně je z nich získávat. Hodnoty zachytíte nebo nastavíte proměnnou instance `messageInfo`.

Příklad 5.4 XAML pro ovládací prvek Silverlightu `EmailClient`

```

<TextBox Grid.Row="0" Grid.Column="1" Margin="11, 5, 45, 5"
    Style="{StaticResource TextBoxStyle}" x:Name="tbFrom"
    HorizontalAlignment="Stretch"
    Text="{Binding Mode=TwoWay, Path=FromAddress}"/>

<TextBox Grid.Row="1" Grid.Column="1" Margin="11, 5, 45, 5"
    Style="{StaticResource TextBoxStyle}"
    x:Name="tbTo" HorizontalAlignment="Stretch"
    Text="{Binding Mode=TwoWay, Path=ToAddress}"/>

<TextBox Grid.Row="2" Grid.Column="1" Margin="11, 5, 45, 5"
    Style="{StaticResource TextBoxStyle}" x:Name="tbSubject"
    HorizontalAlignment="Stretch"
    Text="{Binding Mode=TwoWay, Path=Subject}"/>

<TextBox Grid.Row="3" Grid.Column="1" Margin="11, 5, 45, 5"

```

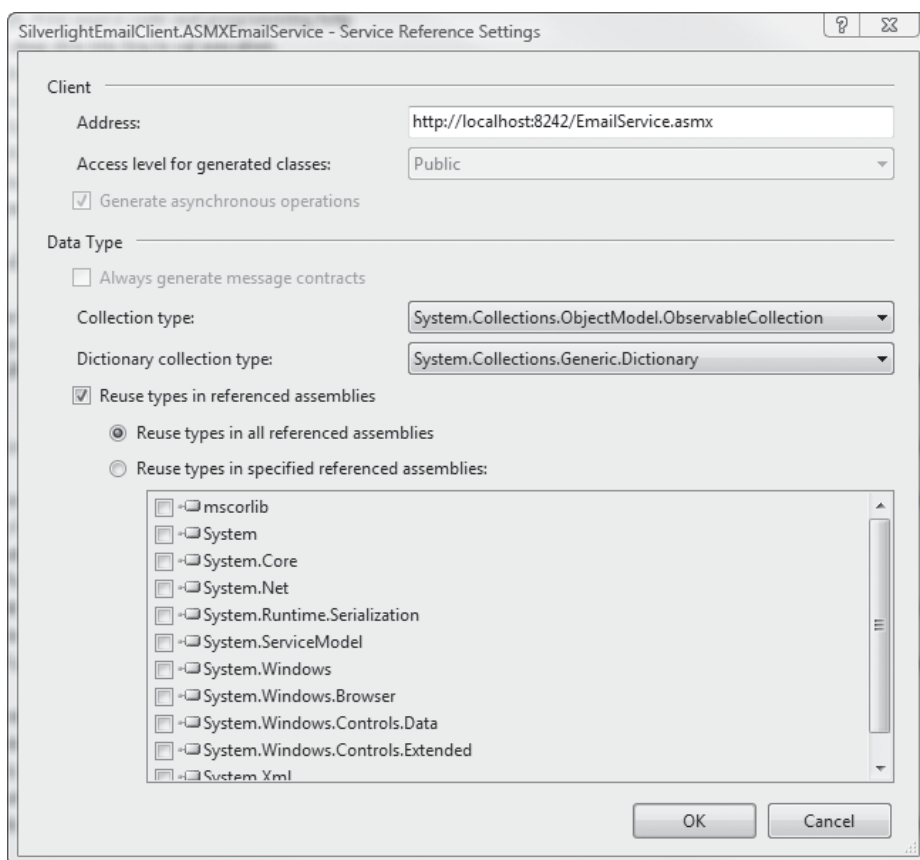
```

Style="{StaticResource TextBoxStyle}" x:Name="tbBody"
HorizontalAlignment="Stretch" VerticalAlignment="Top" Height="100"
VerticalScrollBarVisibility="Visible" AcceptsReturn="True" TextWrapping="Wrap"
Text="{Binding Mode=TwoWay, Path=Body}"/>

```



Třída **MessageInfo** obsahuje veřejnou vlastnost pouze pro čtení s názvem **BodyWithDateTag**, jež neodkazuje přímo na pomocnou vlastnost. Vlastnost **BodyWithDateTag** vrátí vlastnost zprávy **Body** spolu s datem a časovým razítkem, především pro účely ladění.



Obrázek 5.5 Nastavení odkazu na službu

Konstruktor **EmailClient** (viz příklad 5.5) také přidává obsluhu události **Click** pro prvek **Button** s názvem **btnSendViaASMX**. Metoda **btnSendViaASMX** obsluhuje událost kliknutí, když je uživatel připraven odeslat zprávu. Tato metoda vytvoří proxy a bude volat webovou službu.