Developing **MyNet** using Visual Studio and Windows Azure

A step by step guide

Content

Prerequisites	3
Introduction MyNet Application	4 4
Creating a Windows Azure Database	6
Creating the Database Scheme	12
Developing and Testing a Web Service	
Deploying a Web Service into Windows Azure	29
Testing the Windows Azure Service Locally Windows Azure Service Deployment	32 36
Developing a Web User Interface based on a Windows Azure Service	39

Prerequisites

- 1. Activation of the Windows Azure Code http://www.WindowsAzurePass.com/azureu
- 2. Visual Studio 2012 Professional Edition
- 3. Windows Azure SDK <u>http://www.windowsazure.com/en-us/develop/downloads/</u>
- 4. Microsoft Web Platform Installer http://www.microsoft.com/web/downloads/platform.aspx

Introduction

The objective of this guide is to give you an overview of the required steps for developing a cloud-based application using Visual Studio and Windows Azure. For this purpose the guide describes the implementation of a basic (but functional) version of the application **MyNet** in button-up fashion. Just follow the steps.

During the course you will be asked to extend the functionality of **MyNet** multiple times. Consider this guide as a memento of how to accomplish a particular task.

MyNet Application

As shown in figure 1 **MyNet** is a cloud application that integrates the social networks' information of a particular user in one place.



Figure 1 MyNet application

The main components of the application are following:

- 1. **Data Management** in charge of data storage according to a data model (e.g., the relational model).
- Business Logic offering set of operations for accessing and modifying the database through a program (e.g., functions for importing and inserting data from Facebook and Twitter into the DB). These operations are exported by web services.
- 3. **Presentation** for data visualization and operation invocation based on a HTML client. This layer is also responsible for managing users (e.g., login, registering).

As a user you will interact with the presentation component through the **Web UI** using a web browser. Nevertheless note that it is possible to interact directly with the Web Services offered by **MyNet**.

Creating a Windows Azure Database

Creating a database in Windows Azure requires first the creation of a **database server**. Both steps can be accomplished by using the **Windows Azure Management Portal**.

In the web browser:

- 1. Go to http://windows.azure.com and sign in into the portal
- 2. Click **+NEW** at the bottom of the portal



3. Click Data Services > SQL Database > Custom Create

NEW							×
F	COMPUTE	В	SQL DATABASE	F	QUICK CREATE	Simply specify a name to	
x	DATA SERVICES		STORAGE	Ŷ	CUSTOM CREATE	then decide whether to use	
¢°°	APP SERVICES			5	IMPORT	an existing server or create a new one.	
\odot	NETWORKS						

- 4. In the creation form enter **MyNetDB** in the field *Name*. This will be the name of the new database.
- 5. Leave the default values for the *Edition*, *Limit Database Size*, and *Collation* fields.
- 6. Select **New SQL Database Server** in the **Server** field and verify that your form looks like the one shown below.

MyNetDB	×	
MER ROZINE22		
IMIT DATABASE SIZE (MAX SIZE)		
COLLATION		
SQL_Latin1_General_CP1_CI_AS	✓ 2	
SERVER		
New SQL Database Server	\checkmark	

Database Settings

Remark

- SQL Database provides two database editions: the Web Edition that grows up to a size of 5 GB and the Business Edition that grows up to a size of 50 GB.
- For each SQL database created on Windows Azure, there are actually three replicas of that database. This is done to ensure high availability. Also, failover is transparent and part of the service.
- The <u>Service Level Agreement</u> provides 99.9% uptime for SQL Database.
- 7. Click the *arrow* to go on to the next page where you will specify the settings of your **database server**.
- 8. Enter an authentication login name and provide a strong password.
- 9. Choose your nearest geographical location as Region.
- 10. Leave selected the *Allow Windows Azure Services to access the server* option and verify that your form looks like the one shown below.

espinosa	2	
•	•	
LOGIN PASSWORD	_	
LOGIN PASSWORD CONFIRMATION	٦	
••••••		
North Europe]	
Allow Windows Azure services to access the server	9	



Remark

- Region determines the geographical location of the server. Choose a location that is closest to you. Keeping your Windows Azure application and database in the same region saves you on egress bandwidth cost and data latency.
- Notice that you did not specify a server name. SQL Database *auto-generates the server name* to ensure there are *no duplicate DNS entries*. The server name is a tencharacter alphanumeric string. You cannot change the name of your SQL Database server.

- The Allow Windows Azure Services to access the server option is important since it allows the administration of the database using the Preview Management Portal.
- 11. Click the **checkmark** at the bottom of the page and wait for the creation of both the *database server* and the *database instance*.

🕂 Windows Azure		PREV	IEW Q		~
ALL ITEMS	sql databases				
MOBILE SERVICES	DATABASES SERVERS				
	vx2lf0ikby	→ ✓ Started	North Europe	SUBSCRIPTION	
DB SOL DATABASES					
STORAGE					



12. Click **SQL Databases** > **Servers** > the **arrow** next to the name of your database server for opening the Dashboard of your database server.

V	Vindows Azure			PREVIEW
	ALL ITEMS	sql databases		
	MOBILE SERVICES	DATABASES SERVERS	STATUS	
••••	0	yx2lf0ikbv	→ ✓ Started	
DB	SQL DATABASES			
	STORAGE 0			

13. In the Dashboard click **Configure** for configuring the database server **firewall**.

W	indows Azure	PREVIEW	
•	$\langle \epsilon \rangle$	42 DASHBOARD DATABASES CONFIGURE HISTORY	
	yx2lf0ikbv		
		usage overview	
DB		USED AVAILABLE	
		1 Database(s)	
₿Ъ		QUOTA	1% of 149 DATABASES

14. Add your **current IP address** to the list of *allowed IPs addresses* and click **Save** at the bottom of the page.

allowed ip addresses					0
CURRENT CLIENT IP ADDRESS		(add to allow	VED IP ADDRESSES	
@ Home	12.23	3.34.45		12.23.34.45	×
RULE NAME	STAR	T IP ADDRESS		END IP ADDRESS	
allowed services windows azure services	S NO				0
	آئ ې		5		

Remark

• If you are connecting from a network, this is the **IP address that your router or proxy server** is listening on. SQL Database detects the IP address used by the current connection so that you can create a firewall rule to accept connection requests from this device. You now have a database instance and a database server accepting inbound connections from your IP address. Now you will retrieve the necessary information for connecting to the database.

15. In the *Management Portal*, click **SQL Databases** > **Databases** > the **arrow** next to the **MyNetDB** database.

Windows Azure			PREVIEW
ALL ITEMS	sql data	abases	
	DATABASES	SERVERS	
	NAME	STATUS	LOCATION
0	FaceNet	🕞 🗸 Online	North Europe
B SQL DATABASES			
STORAGE			

16. Click **Show Connection Strings**.

quick glance
(i) Show connection strings
i Learn how to troubleshoot connections
i Learn about backup and restore
NAME FaceNet
STATUS Online
SERVER yx2lf0ikbv
COLLATION SQL_Latin1_General_CP1_CI_AS
EDITION Web

17. Copy the string in the **ADO.NET** section. You will use it in your code for connecting to the database.



Remark

- For security reasons the connection string does not include your password. You will need to type it into the string before using it.
- The Fully Qualified Name is a URL includes the server address and the port.

Congratulations! Now you are ready to go.

Creating the Database Scheme

The first step in the development of MyNet is the definition of the database scheme. This section will show you how to connect to your *database server* for creating a database table.

Start by preparing Visual Studio (VS).

1. Run VS12 as administrator



2. Click File > New > Project



- 3. Click Web > .NET Framework 4 > ASP.NET Empty Web Application
- 4. Name the solution **MyNet** and the project **MyNet.WebServices**
- 5. Click OK

	New Project	? ×
▷ Recent	.NET Framework 4 Sort by: Default Search Installed Templates	s (Ctrl+E) 🛛 🔎 👻
▲ Installed	ASP.NET Empty Web Application Visual C# Type: Visual C#	
 Templates Visual C# Windows Store 	An empty project for cree ASP.NET Web Forms Application Visual C#	ating an Iser interface
Windows	ASP.NET MVC 3 Web Application Visual C#	
Cloud	ASP.NET MVC 4 Web Application Visual C#	
Silverlight Test	ASP.NET Dynamic Data Entities Web Application Visual C#	
WCF Windows Phone	ASP.NET Dynamic Data Ling to SQL Web Application Visual C#	
Workflow ▷ Other Languages	ASP.NET AJAX Server Control Visual C#	
 Other Project Types Samples 	ASP.NET AJAX Server Control Extender Visual C#	
▶ Online	ASP.NET Server Control Visual C#	
Name: MyNet.We	Services	
Location: C:\Users\J	Desktop\cdm\	
Solution name: MyNet	✓ Create directory for solut Add to source control	ion
	ОК	Cancel

Remark

For the time of writing Windows Azure only supports .NET Framework 4, so be sure that you select the right framework.

Next, in the menu of Visual Studio:

6. Click SQL > Transact-SQL Editor > New Query



7. Enter your **database server** information, your **username** and your **password**

J.	Connect to Server	×
SQL Se	rver 2012	
Server type:	Database Engine	~
Server name:	yx2lf0ikbv.database.windows.net,1433	~
Authentication:	SQL Server Authentication	~
Login:	espinosa@yx2lf0ikbv	¥
Password:	*****	
	Remember password	
Conne	Cancel Help Options	>>

- 8. Click **Options**
- 9. Choose **Browse Server** in the *Connect to database* list of the *Connection Properties* tab

J Connect to Server								
SQL Server 2012								
Login Connection Properties	Additional Connection Parameters							
Type or select the name of the	Type or select the name of the database for the connection.							
Connect to database:	<default></default>							
Network	<default></default>							
Network protocol:	 default>							
Network packet size:	4096 🜩 bytes							
Connection								
Connection time-out:	15 🚖 seconds							
Execution time-out:	0 🗢 seconds							
Encrypt connection								
Use custom color:	Select							
	Reset All							
Connect	Cancel Help Options <<							

- 10. Click Yes and wait a few moments
- 11. Select MyNetDB and click OK

Browse Server for Database						
Browse the server and select the database to connect to.						
 yx2lf0ikbv.database.windows.net,1433 yx2lf0ikbv.databases ystem Databases yester yester Databases yester Databases yester Databases 						
OK Cancel						

12. Finally click **Connect**

At this point you have established a connection to your database server and all the queries you execute in the *SQL Editor* will be directed to the database **MyNetDB**. As an example, enter the following lines in the editor and execute them by clicking in the VS menu **SQL** > **Transact-SQL Editor** > **Execute**.

```
CREATE SCHEMA mynet
GO
CREATE TABLE mynet.Contact (
ID INT IDENTITY (1, 1) NOT NULL,
Lastname NVARCHAR(MAX) NOT NULL,
Firstname NVARCHAR(MAX) NOT NULL,
Society VARCHAR (20) NULL,
CONSTRAINT pkContact PRIMARY KEY CLUSTERED (ID DESC)
);
GO
```

As shown in the following figure, after the execution of the query your database will contain the table **mynet.Contact**.



You will conclude this section by saving the previous query in a file.

In the Solution Explorer:

- 13. Click right button **MyNet.WebServices** > **Add** > **New Folder**
- 14. Name it **DataModel**

			Solution Explorer				
			◎ ◎ ☆ ◎ • ≈ ◎ ◎ ● ▶				
			Sea	rch Solution Explorer (C	trl+\$)		
			N	Solution 'MyNet' (1 p	roject)		
.*.	Ruild		14	MyNet.WebServi	ices		
	Rebuild		L .	Properties			
	Clean		L .	C# IMyNetDbServi	ice.cs		
6	Publich		L	▷ :⊕ MyNetDbServi	ce.svc		
	Run Code Analysis		L .	Veb.config			
តា	View in Browser (Internet Evolorer)		L				
9-	Convert to Web Application		L .				
ak	View in Page Inspector	Ctrl+K Ctrl+G	L .				
C.	Check Accessibility	carriç carro	L .				
Ŭ	Scone to This						
-	New Solution Explorer View		L .				
	Coloridad Code Matrice		Sol	ution Explorer Team Ex	plorer		
	Calculate Code Metrics		Dro	portion			
	Add Windows Azure Cloud Service Project			perties			
	Add	•	°D	New Item	Ctrl+Shift+A		
	Add Reference		* 0	Existing Item	Shift+Alt+A		
	Add Service Reference		1	New Folder			
畄	Manage NuGet Packages			Add ASP.NET Folder	•		
* \$	View Class Diagram			Web Form			
ø	Set as StartUp Project			Web User Control			
	Debug	+		JavaScript File			
*7	Add Solution to Source Control			Style Sheet			

Back in the *Transact-SQL Editor*:

- 15. Press **CTRL + S** for saving the query
- 16. Name it **MyNetDB.sql** and save it inside the **DataModel** folder.
- 17. Right click **MyNet.WebServices** project > **Add** > **Existing item** for integrating the file into VS for future use

		.	Soluti	on Explo	orer			Ŧ	Ψ×
			0 0	۵ <u>۵</u>	"o - ≓	00	6	6 0	
		+	Search	n Solutio	on Explorer	(Ctrl+\$)			ρ-
	Build Rebuild Clean Publish to Windows Azure Publish to Windows Azure Publish Run Code Analysis View in Browser (Internet Explorer) Convert to Web Application View in Page Inspector Check Accessibility Check Accessibility Scope to This New Solution Explorer View Calculate Code Metrics Project Dependencies Project Build Order	Ctrl+K, Ctrl+G		Solution	ı 'MyNet' (: let.WebSen let.WebUl	3 projects vices rsAzure)		
	Add	•	°`	New I	tem		Ctrl+Sh	ift+A	
-	Add Kererence			Existin	ig Item		Shift+A	lt+A	
ă	Manage NuGet Packages		, <u>-</u>	New F Add A	SP.NET Fo	lder			٠

Developing and Testing a Web Service

As described in the **Introduction** of this document, the business logic of the application *MyNet* is exposed as a set of web services. For instance, the web service **MyNetDbService** offers CRUD operations (Create, Read, Update, Delete) for manipulating the database MyNetDB. This section will show you how to develop a web service in VS by implementing a simplified version of the service **MyNetDbService**. The service will offer the operations shown in the class diagram of Figure 2.



Figure 2 MyNetDbService Operations

Note that the operations offered by the service receive and return objects of type **Contact**. You will start by creating the class **Contact**. Then you will create the service **MyNetDbService**.

In the Solution Explorer:

18. Right click **DataModel** folder > **Add** > **Class**

			Solu	ition Explorer		- ų ×
			G	୍ର 🟠 🧿 - 🤃 🖸	d 🕞 🖊 🔂	
			Sear	ch Solution Explorer (Ctrl	+\$)	- <i>م</i>
			2	Solution 'MyNet' (1 proj	ect)	
5 6 6 8	Build Rebuild Clean Publish Run Code Analysis View in Browser (Internet Explorer) Convert to Web Application View in Page Inspector	Ctrl+K, Ctrl+G		MyNet.WebService ✓ Properties Interferences C" INJNetDbService MyNetDbService Q Web.config	s cs svc	
G	Check Accessibility		L .			
	Scope to This New Solution Explorer View Calculate Code Metrics Add Windows Azure Cloud Service Project		Solu	ition Explorer Team Expl	orer	- ₫ ×
	Add	•	°D.	New Item	Ctrl+Shift+A	
ă	Add Reference Add Service Reference Manage NuGet Packages		†0 *2	Existing Item New Folder Add ASP.NET Folder	Shift+Alt+A	
49	View Class Diagram			Web Form		es.csproj
Φ	Set as StartUp Project Debug	•		Web User Control JavaScript File		⊳\cdm\MyNet\
49	Add Solution to Source Control			Style Sheet		
ж	Cut	Ctrl+X		WCF Service		404/
£	Paste	Ctrl+V	**	Class	Shift+Alt+C	
-			1			

19. Name it **Contact.cs**

20. Click **Add**

	Add New Item - MyNet.We	bServices	? ×
▲ Installed	Sort by: Default	Search Installed Tem	plates (Ctrl+E) 🛛 🔎 👻
✓ Visual C# Code Data	Web Form	Visual C# An empty class dec	laration
General Web Windows Forms	Web User Control	Visual C#	
Reporting	Class	Visual C#	
Silverlight Workflow	Master Page	Visual C#	
▶ Online	Nested Master Page	Visual C#	
	HTML Page	Visual C#	
	Style Sheet	Visual C#	
Name: Contact.cs			.dd Cancel

This will create the file **Contact.cs**. The file already contains the following code:

```
namespace MyNet.WebServices.DataModel {
    public class Contact { }
}
```

Remark

Note that VS automatically constructs the namespace based on the project name and the path to the file.

Complete the class with the following code:

```
public class Contact {
    public int Id;
    public String Firstname;
    public String Lastname;
}
```

Once the type **Contact** defined, you can continue with the definition of the service **MyNetDbService**.

In the Solution Explorer:

21. Click right button **MyNet.WebServices** > **Add** > **New Item**

_			Solut	tion Exp	lorer					
			G	0 🏠	ō	- 2	<u>وې</u>	đ	à 🕨 🗖	
			Sear	ch Solut	tion Exp	olorer	(Ctrl	+\$)		
			N.	Solutio	n 'MyN	Vet' (1	proj	ect)		
			4		Net.W	ebSer	vice	s		
÷.	Build		L	2	Proper	ties				
	Rebuild				Referen	nces				
	Clean			r AT	web.co	onnig				
ø	Publish									
	Run Code Analysis									
6	View in Browser (Internet Explorer)									
	Convert to Web Application									
	View in Page Inspector	Ctrl+K, Ctrl+G								
C	Check Accessibility		L .							
	Scope to This		L							
Ð	New Solution Explorer View				_					
	Calculate Code Metrics		Solu	tion Exp	lorer	Team	Expl	orer		
	Add Windows Azure Cloud Service Project		Prop	erties						
	Add Mindows Azare cloud Service Hojeet		-			-	•			
	Add Peference			New	tem			(Ctrl+Shift+A	
	Add Reference			Existin	ig Item			5	shift+Alt+A	
	Add Service Reference		1-20	New F	older					
	Manage NuGet Packages			Add A	SP.NET	Fold	er			•
14	View Class Diagram			Web F	orm					5.
Ø	Set as StartUp Project			Web l	Jser Co	ntrol				10
	Debug	•		JavaSo	cript Fil	e				
÷9	Add Solution to Source Control			Style S	Sheet					

- 22. Click **Web** > **WCF Service**
- 23. Name it **MyNetDbService.svc**
- 24. Click **Add**

	Add New Item - MyNet.WebService	25	? ×
✓ Installed	Sort by: Default		Search Installed Templates (Ctrl+E)
✓ Visual C# Code	Dynamic Data Field	Visual C#	 Type: Visual C# A class for creating a WCF service
Data General	Generic Handler	Visual C#	-
Web Windows Forms	Global Application Class	Visual C#	
WPF Reporting	Site Map	Visual C#	
Silverlight Workflow	Skin File	Visual C#	
▶ Online	WCF Data Service	Visual C#	
	WCF Service	Visual C#	
	Web Configuration File	Visual C#	
	I Web Service	Visual C#	-
Name: MyNetDbServic	e.svc		
			Add Cancel

The previous steps will create the following files:

- **IMyNetDbService.cs** containing an interface defining the operations that will be exported by the service.
- **MyNetDbService.svc** containing the class implementing the operations defined in the service interface.

Open the file **IMyNetDbService.cs** and change the auto-generated code with the following code:

```
namespace MyNet.WebServices {
    using MyNet.WebServices.DataModel;
    [ServiceContract]
    public interface IMyNetDbService {
        [OperationContract]
        void addContact( Contact contact );
        [OperationContract]
        List<Contact> getContacts( );
    }
}
```

The new code specifies the methods addContact and getContacts as the operations exported by the service.

Note the use of the namespace MyNet.WebServices.DataModel. Recall that it denotes the address of the class **Contact** that the compiler will use for locating it. Note also that the interface and the methods are respectively decorated with the attributes [ServiceContract] and [OperationContract]. At compilation time these attributes and the information in **Web.config** are used by the *middleware* (WCF) and the *web server* (IIS) for preparing all the plumbing that will host your service instances.

Remark

Do not forget to declare the interface and the interface methods with the attributes [ServiceContract] and [OperationContract]. Without them the service will not work.

Once the interface has been specified, continue implementing the service operations. For this, open **MyNetDbService.svc** and complete the auto-generated code with the following code:

```
using System.Data.SqlClient;
namespace MyNet.WebServices {
    using MyNet.WebServices.DataModel;
    public class MyNetDbService : IMyNetDbService {
        static String databaseServer = "ENTER_YOUR_AZURE_DB-SERVER_SETTINGS_HERE";
```

```
static String databaseName = " ENTER YOUR AZURE DB-SERVER SETTINGS HERE";
static String username = " ENTER_YOUR_AZURE_DB-SERVER_SETTINGS_HERE";
static String password = " ENTER_YOUR_AZURE_DB-SERVER_SETTINGS_HERE";
static String MyNetDbConnectionString;
public MyNetDbService( ) {
    // Create DB Connection String
    var connectionString = new SqlConnectionStringBuilder {
         DataSource = databaseServer,
         InitialCatalog = databaseName,
         Encrypt = true,
         TrustServerCertificate = false,
         UserID = username,
         Password = password
    };
    MyNetDbConnectionString = connectionString.ToString();
} // Constructor
/// Add Contact
public void addContact( Contact contact ) {
    // Connect to MyNetDB
    using ( var dbConnection = new SqlConnection( MyNetDbConnectionString ) ) {
         dbConnection.Open();
         // Execute SQL statements
         using ( SqlCommand sqlCommand = dbConnection.CreateCommand() ) {
             // Firstname and Lastname may contain apostrophes '
             // They are replaced with double apostrophes '' before insertion in DB
var firstname = contact.Firstname.Replace( "'", "''" );
var lastname = contact.Lastname.Replace( "'", "''" );
             var sqlStatementTemplate =
                  'INSERT INTO mynet.Contact ( Firstname, Lastname ) VALUES ( N'{0}', N'{1}')";
             var sqlStatement = String.Format( sqlStatementTemplate, firstname, lastname );
             // Insert Contact in DB
             sqlCommand.CommandText = sqlStatement;
             sqlCommand.ExecuteNonQuery();
         } // SqlCommand
         dbConnection.Close();
    } // dbConnection
} // Method
/// Get All Contacts
public List<Contact> getContacts( ) {
    List<Contact> contacts = null;
    // Connect to MyNetDB
    using ( var dbConnection = new SqlConnection( MyNetDbConnectionString ) ) {
         dbConnection.Open();
         // Execute SQL statements
         using ( SqlCommand sqlCommand = dbConnection.CreateCommand() ) {
             // Query: Select ALL Contacts
             sqlCommand.CommandText = "SELECT * FROM mynet.Contact";
             SqlDataReader reader = sqlCommand.ExecuteReader();
```

```
// Create Contact object for every contact in MyNetDB
                    contacts = new List<Contact>();
                    while ( reader.Read() ) {
                     var contact = new Contact {
                            Id = (int) reader["Id"],
                            Firstname = (String) reader["Firstname"],
                          Lastname = (String) reader["Lastname"]
                       };
                contacts.Add( contact );
            } // While
                } // SqlCommand
                dbConnection.Close();
            } // dbConnection
            return contacts;
        } // Method
    } // Class
}
```

The line public class MyNetDbService : IMyNetDbService specifies that the class implements a service exporting the operations defined in the IMyNetDbService interface. These operations will use the information related to your *database server* for establishing a connection with your DB in order to add contacts and retrieve the existing ones.

The rest of the code should be self-explanatory.

Remark

Do not forget to **change the values** of the variables containing your Windows Azure *database settings*.

As stated before, the file **Web.config** contains information about your services in the form of xml. You can always edit this file by hand however, for the sake of clarity, you will use the WCF Configuration Tool for defining your service graphically.

Press **F6** to compile the project. Then in the Service Explorer:

25. Click right button Web.config > Edit WCF Configuration



- 26. Click Services > Create a New Service
- 27. Click Browse > bin Folder > MyNet.WebServices.dll > MyNet.WebServices.MyNetDbService
- 28. Click **Open** > **Next**
- 29. Click **HTTP** > **Next**
- 30. Click Basic Web Service Interoperability > Next
- 31. Delete the content of Address
- 32. Click **Next** > **Yes**
- 33. Click Finish

e Help	
onfiguration	Service: MyNet.WebServices.MyNetDbService
Services MyNet.WebServices.MyNetDbService MyNet.WebServices.MyNetDbService Gradient Standard Endpoints Diagnostics Advanced	(General) BehaviorConfiguration Name MyNet.WebServices.MyNetDbService
asks 🛞 elete Service eate a New Service	
eate a New Client	Name The type of the service. The service type is the class that implements your service.

34. Close the editor and Save the changes

This will insert the following xml lines in the **Web.config** file:



The more relevant information is inside the <service> element. The element specifies that

- The service is implemented by the class MyNetDbService
- The operations exposed by the service are those defined in the interface **IMyNetDbService**.
- The **URL** used for locating the service is **relative** to the URL where you will deploy the project

At this point your service is ready to be tested. For this purpose you will use the *WCF Test Client* developer tool that is a *generic web service client* available in VS. In order to use it you need to configure your project first for launching automatically the WCF Test Client.

In the Solution Explorer:

35. Click right button **MyNet.WebServices** > **Properties**

				Solution	n Explorer		- ↓ ×		
				00	∰ ™o - ≓ ©	a 🗿 🗡 🔂			
				Search	Solution Explorer (Ctrl	+\$)	ρ-		
				Jan So	olution 'MyNet' (1 proj	ect)			
١.				4 🖲	MyNet.WebService	s			
		Build		Þ	🔑 Properties				
		Rebuild		Þ	References				
		Clean			DataModel	4			
	€	Publish		⊳	C# IMvNetDbService	,cs			
		Run Code Analysis	Code Analysis			svc			
	0	View in Browser (Internet Explorer)			▶ 🚹 MyNetDbServ	ice.svc.cs			
		Convert to Web Application		Þ	🖓 Web.config				
	€*	View in Page Inspector	Ctrl+K, Ctrl+G						
	C	Check Accessibility							
		Scope to This							
		New Solution Explorer View		L					
		Calculate Code Metrics		Solutio	n Explorer Team Expl	orer			
		Add Windows Azure Cloud Service Project		Propert	ies				
		Add	•	MyNet	.WebServices Projec	t Properties	•		
		Add Reference		8 24	عر ،				
		Add Service Reference		Alwa	ays Start When Debugg	jii True			
	Ħ	Manage NuGet Packages		Ano	nymous Authenticatio	n Enabled			
	***	View Class Diagram		Man	aged Pipeline Mode	Integrated			
	8	Set as Startlin Project		Proje	ect Folder	C:\Users\J\Deskto	p\cdm\MvNet\		
ł.	~	Debug		SSL E	Enabled	False			
	* 3		,	SSL	URL				
	7	Add Solution to Source Control		URL		http://localhost:4	9454/		
ne 1	¥	Cut	Ctrl+X	Wind	dows Authentication	Disabled			
		Paste	Ctrl+V						
de	×	Remove	Del						
de	1.	Rename							
		Unload Project		Always Start When Debugging					
1	0	Open Folder in File Explorer		Start th	e local Web server eve	n when not the star	up project		

- 36. Click **Web** > **Specific Page**
- 37. Type in **MyNetDbService.svc**

MyNet.WebServices* 🕫 🗙							
Application Build	Configuration: N/A	✓ Platform: N/A ✓					
Web*	Start Action						
Package/Publish Web Package/Publish SQL	O Current Page						
Silverlight Applications	Specific Page	MyNetDbService.svc					
Build Events	 Start external program 						
Resources	Command line arguments						
Settings	Working directory						
Reference Paths							
Signing	Start URL						
Code Analysis	○ Don't open a page. Wait f	or a request from an external application.					

38. Save (CTRL + S) and close the tool

With this last configuration done you can test your service by right clicking **MyNet.WebServices** > **Debug** > **Start new instance** in the *Solution Explorer*.

		Sol	lution	Explorer		- ‡ ×
		G	0	₲ ७ - २ 🗇 🗡	33	
		Sea	arch So	olution Explorer (Ctrl+\$)		- م
	Build] Soli	ution 'MyNet' (3 projects)		
	Rebuild			MyNet.WebServices		
	Clean			MyNet.WebUI MyNet WindowsAzure		
	Publish to Windows Azure		ľ	MyNet.WindowsAzure		
6	Publish					
	Run Code Analysis					
6	View in Browser (Internet Explorer)					
	Convert to Web Application					
⊕*	View in Page Inspector	Ctrl+K, Ctrl+G				
G	Check Accessibility					
	Scope to This					
Ē	New Solution Explorer View					
	Calculate Code Metrics					
	Project Dependencies					
	Project Build Order					
	Add	•				
	Add Reference					
	Add Service Reference					
ă	Manage NuGet Packages					
13	View Class Diagram					
Φ	Set as StartUp Project					
	Debug	•	•	Start new instance		
*7	Add Solution to Source Control		ς.	Step Into new instance		
ж	Cut	Ctrl+X	-			
£	Paste	Ctrl+V				

Then follow the instructions presented in the *WCT Test Client* for testing the operations exported by your service.

63		WCF Test Client	- 🗆 ×
File Tools Help Image: Service Projects Image: Service Projects	Start Page	To add a service: . Select "Add Service" from the File menu or the context menu o . Enter the service metadata address in the input area, and click To test a service operation: . Double click the operation you want to test from the tree on th . A new tab page will appear on the right pane . Enter the value of parameters in the Request Area of the right . Click "Invoke" button	of the "My Service Projects" k "OK" e left pane pane
Service added successfully.			

Remark

When testing your web server locally the **URL** of the service has the following structure

http://localhost:<**YOUR_PORT**>/<**YOUR_SERVICE_NAME**>.svc

Deploying a Web Service into Windows Azure

In the previous section you developed the web service **MyNetDbService**. In this section you will learn how to deploy the service (or any web service) into Windows Azure. You will start by defining a **Windows Azure Project** that will contain the information used by Windows Azure for creating and configuring the Virtual Machines (VM) that will host your services.

In the Solution Explorer:

- 4 × Solution Explorer ○ ○ 습 '⊙ - ≠ 司 🖌 🔂 + Search Solution Explorer (Ctrl+\$) ρ. Build Solution F6 MyNet.WebServices **Rebuild Solution** MyNet.WebUI Properties Þ Clean Solution ▶ ■ References Alt+F11 Run Code Analysis on Solution Account Batch Build... App_Data Configuration Manager... 4 App_Start C# AuthConfig.cs Manage NuGet Packages for Solution... Þ Content Enable NuGet Package Restore 👂 デ Images New Solution Explorer View Þ Scripts Þ About.aspx Calculate Code Metrics Þ Contact.aspx Project Dependencies... Þ Default.aspx Project Build Order... favicon.ico Add • New Project... Set StartUp Projects... Existing Project... New Web Site... Add Solution to Source Control... Existing Web Site... A Paste I Rename Dew Item... Ctrl+Shift+A to Existing Item... Shift+Alt+A Copen Folder in File Explorer a New Solution Folder Properties Alt+Enter
- 1. Right click **MyNet** Solution > **Add** > **New Project**

- 2. Click Cloud > Windows Azure Cloud Service
- 3. Name it **MyNet.WindowsAzure**

		Add New I	Project	? 🗙
▷ Recent		.NET Framework 4 • Sort by: Default	• # E	Search Installed Templates (Ctrl+E)
✓ Installed ✓ Visual C [≠] Windows Store Windows ✓ Cloud Reporting Silverlight Test WCF Windows Phon Workflow ▶ Other Languages	e v	Windows Azure Cloud Service	Visual C#	Type: Visual C# A project for creating a scalable service that runs on Windows Azure.
▷ Online				
Name: Location:	MyNet.Windows	Azure op\cdm\MyNet		Browse OK Cancel

4. Click OK

5. Leave the list of Windows Azure Cloud Service Solutions empty and click OK

New Windows Azure Cloud Service						×
.NET Fra	mework 4 roles:			Windows Azure Cloud Service solution:		
🔿 Vis	ual C#	^				
<mark>لک</mark>	ASP.NET Web Role Service with a web user interface					
ل ے	ASP.NET MVC 4 Web Role Service with a web user interface usin					
ت	ASP.NET MVC 3 Web Role Service with a web user interface usin		>			
o.	WCF Service Web Role Web role for WCF services		<			
ل والح الت	Worker Role Background processing service					
ل والح الت	Cache Worker Role Background processing service that h					
-5*	Worker Role with Service Bus Queue	\sim				
				ОК	Cance	el

The previous steps will create a new *Windows Azure Project* that contains no references to a web service. In the next step you will associate the web service MyNetDbService to your Windows Azure project through the concept of **Roles**.

In the Solution Explorer:

6. Right click Roles inside the MyNet.WindowsAzure project



- 7. Click Add > Web Role Project in solution
- 8. Select **MyNet.WebServices** and click **OK**

2	Associate with Role Project ? ×
Projects	
Project Name	Project Directory
MyNet.WebService	s C:\Users\J\Desktop\cdm\MyNet\MyNet
MyNet.WebUI	C:\Users\J\Desktop\cdm\MyNet\MyNet
<	OK Cancel

This will produce a new role called *MyNet.WebServices* in your Windows Azure project.

As stated before roles contain information about the VMs hosting your service. Open the role **MyNet.WebServices** and see the default configuration for your service. For instance, note that the default configuration specifies that only one VM will be created.

Remark

For this exercise 1 instance of a VM is ok. However you can increment the number of VMs (up to three with your subscription) in order have multiple instances of your service. Some advantages of doing these are: fault tolerance, load balancing, SLA etc.

Also note that, under the **Endpoint** section, the default **Public Port** assigned to your service is **80**. As you may recall this port is usually reserved to web applications like *MyNet Web UI* so you have to change the default value.

Back in the open **MyNet.WebServices** role file:

- 9. Click **Endpoints**
- 10. Change **Public Port** to **8080**

	1			· ·				
ī	MyNet.WebServices [Role]*	+ ×						
5								
ox	Configuration	Service Configuration: All Configurations						
	Settings							
	Endpoints	ΆÅ	凸 Add Endpoint 🗙 Remove Endpoint					
	· · · ·							
	Local Storage	Conf	igure the endpoints	for this role. Selec	t the certifi	icate to use fo	r each HTTPS endpoint when the Windo	ws Azure Cloud Service project is deployed to
		Wind	lows Azure (not app	licable when runni	ng on the l	ocal Window	Azure compute emulator).	
	Certificates							
			Manaa	Turne	Destand	Dublis Dast	Definition Dent	SSI Cartificate Name
	Virtual Network		Name	туре	Protocol	Public Port	Private Port	SSL Certificate Name
	The second		Endpoint1	Input Y	http ~			(not applicable)
	Caching			,				

11. Save (CTRL + S) and close the file

Testing the Windows Azure Service Locally

At this point your service should be ready to be tested locally. For this purpose you will use the **Windows Azure Emulator**.

Remark

The Windows Azure Emulator is one of the multiple tools belonging to the *Windows Azure Tools* that you installed as prerequisites.

In the Solution Explorer:

12. Right click MyNet.WindowsAzure > Debug > Start new instance



This will launch the Windows Azure Emulator (visible in the Windows Taskbar) and will open your default web browser pointing to the **address where your service is hosted**.





By default cloud projects do not launch the **WCF Test Client** so you will need to launch it manually.

In the Windows Application Menu:

13. Search for **Developer Command Tools for VS 2012** using the *search tool* and launch it



14. In the command tool enter wcftestclient



This will open the *WCF Test Client*. To using it and test your service you need to specify the address to your service.

In the WCF Test Client:

15. Click **File** > **Add Service**

•		WCF Test Client
File Tools Help		
Add Service Ctrl+Shift+A	Start Page	
Recent Services		
Exit		To add a service: . Select "Add Service" from the File menu or the context menu of th . Enter the service metadata address in the input area, and click "(
		To test a service operation: . Double click the operation you want to test from the tree on the le . A new tab page will appear on the right pane . Enter the value of parameters in the Request Area of the right par

16. Enter the **address** to your service e.g. *http://localhost:<PORT>/MyNetDbService.svc*

Add Service	×
Please enter the endpoint address:	ОК
	Cancel
http://localhost:49454/MyNetDbService.svc	~

Remark

The *WCF Test Client* is a handful tool for testing local and remote web services. Try to remember these steps. You will use them during the next section.

Windows Azure Service Deployment

The easiest way to deploy a web service into Windows Azure is using the VS tools.

In the Service Explorer:

17. Right click **MyNet.WindowsAzure** > **Publish**



18. Click Sign In To Download Credentials and follow the instructions in the browser

	Publish Windows Azure Application	?	×
Windows	Azure Publish Sign In		
Sign in Settings Summary	Credentials are required to connect to your Windows Azure subscriptions Sign in to download credentials Choose your subscription: Azdem181H41764X v	Import.	
	Online privacy statement < Previous Next > Publish	Cance	el 🛛

The previous steps will generate a file containing your **Windows Azure Credentials**. Save the file in your computer since for using it in the next step.

Back in Visual Studio:

- 19. Import file containing your credentials to populate the subscriptions list
- 20. Choose a subscription and click Next

- 21. In Common Settings, in Cloud Service, select Create New
- 22. Name it **MyNetProject< <u>ProjectID</u>** and set your geographical **location**

	Publish Windows Azure Application ? ×
Windows	s Azure Publish Settings
Sign in Settings Summary	Common Settings Advanced Settings Cloud Service: MyNetProject (North Europe) v Environment:
	Production Build configuration: Release Service configuration: Cloud Enable Remote Desktop for all roles Settings Enable Remote Desktop for all roles Settings
	Online privacy statement < Previous Next > Publish Cancel

Remark

The < **ProjectID** > inside the name will be assigned to you by the professor.

23. In Advanced Settings, in Storage Account, select Create New

24. Name it **MyNetStorage**< **ProjectID** > and set your geographical **location**

	Publish Windows Azure Application ? ×
Windows	s Azure Publish Settings
Sign in	Common Settings Advanced Settings
Summary	Deployment label: MyNet.WindowsAzure Image: Append current date and time Storage account: MyNetStorage (North Europe) Image: Delete deployment on failure Image: Delete deployment on failure Image: Delete deployment update Settings Image: Enable profiling Settings
	Online privacy statement < Previous Next > Publish Cancel

- 25. Click Next
- 26. Click the **diskette icon** to save the configuration in a **profile**
- 27. Click **Publish** and wait a couple of minutes

	Publish Windows	Azure Application ? ×
Windows A	zure Publish Summ	ary
Sign in	Target profile: MyNetProjec	tProduction v
Settings	Deployment update:	Enabled
Summary	Subscription:	Azdem181H41764X
	Cloud Service:	MyNetProject (North Europe)
	Environment:	Production
	Build configuration:	Release
	Service configuration:	Cloud
	Remote Desktop:	Disabled
	Web Deploy:	Disabled
	Deployment label:	MyNet.WindowsAzure - 07/11/2012 19:21:27
	Storage account:	MyNetStorage (North Europe)
	Online privacy statement	< Previous Next > Publish Cancel

Once the deployment is finished, look for the URL of your Windows Azure Service in VS. Then use the **WCT Test Client** for testing your service running in the cloud (cf. the steps presented in the previous subsection).

Windows Azure A Deployment	ctivity Log	뚇 Remove all completed		- ₫ ×
Descriptio	n		Status	Start Time (UTC)
🔺 🕑 Deploying	MyNet.WindowsAzure to MyNetProjec	:t - Production	Completed	07/11/2012 19:37:56
Production	Website URL http://mynetproject.cloudapp.net/ Deployment ID c40fd27f37a5406f8e6e530e5a760883 Open in Server Explorer	History 19:42:17 - Created Deployment ID: c40fd27f37a5406f8e6e530e5a760883. 19:42:17 - Starting 19:42:35 - Initializing 19:42:35 - Created Website URL: http://mynetproject.cloudapp.net/ 19:42:35 - Complete.		×
4				Þ

Remark

Note that the **URL** assigned to a cloud application is composed of **the name of the windows azure service** (i.e. the name you specified in the Publishing wizard) and the domain **cloudapp.net**. For instance,

http://mynetproject<mark><ProjectID></mark>.clouddapp.net.

Developing a Web User Interface based on a Windows Azure Service

As you may recall from the introduction, the users of the application MyNet interact with the web services offered by MyNet by using a Web User Interface (Web UI). For instance, using the buttons in the Web UI a user can get all the contacts in the database. In this section you will learn how to develop a simple Web UI in Windows Azure for the application MyNet.

In the Solution Explorer:

- 1. Click right button MyNet Solution> New Project
- 2. Click Web > .NET Framework 4 > ASP.NET Web Forms Application
- 3. Name it MyNet.WebUI
- 4. Click **OK**



This will create a new web application project ready to be used. If you right click **MyNet.WebUI** > **View in Browser** you will see the web application **home page** in your default web browser.



The content of the home page resides in the file **MyNet.WebUI** > **Default.aspx**. You will change the content of this file in order to call the operations of the cloud service **MyNetDbService**.

You will start by generating a client for your service.

In the Solution Explorer:

1. Right click MyNet.WebUI > Add Service Reference



- 2. Specify the **Address** of your service. For instance, <u>http://mynetprojectt####.cloudapp.net:8080/MyNetDbService.svc</u>
- 3. Click GO and wait for a few seconds

4. Set Namespace to MyNetDbService

	Add Service Reference	? ×		
To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.				
Address:				
http://mynetproject.cloudapp.net:8080/MyNetDbService.svc v Go Discover v				
Services:	Operations:			
	Select a service contract to view its opera	itions.		
1 service(s) found at address 'http://	'mynetproject.cloudapp.net:8080/MyNetDbService.svc'.			
Namespace:				
MyNetDbService				
Advanced	ОК	Cancel		

5. Click Advanced

6. In *Data Type > Collection Type* select **System.Collection.Generic.List** and click **OK**

Service Reference Settings			k	
Client			-	
Access level for generated classes:	Public	Ý		
Allow generation of asynchronous operations				
Generate task-based operations				
Generate asynchronous operation:	5			
Data Type			.	
Always generate message contracts				
Collection type:	System.Collections.Generic.List	~		
Dictionary collection type:	System.Array		Н	
Reuse types in referenced assemblies	System.Collections.ArrayList System.Collections.Generic.LinkedList			
Reuse types in referenced assemblies	System.Collections.Generic.List			
Keuse types in an referenced assert	System.Collections.ObjectModel.ObservableCollection			
Reuse types in specified referenced System.ComponentModel.BindingList				
🗌 🗖 AspNet.ScriptManager.jQ	luery	^]	
AspNet.ScriptManager.jQuery.UI.Combined				
DotNetOpenAuth.AspNet				
DotNetOpenAuth.Core				
DotNetOpenAuth.OAuth				
DotNetOpenAuth.OAuth.Consumer				
DotNetOpenAuth.OpenIc	I	~		
Compatibility				
Add a Web Reference instead of a Service Web Services technology.	Reference. This will generate code based on .NET Framew	/ork 2.0		
Add Web Reference				
	ОКС	ancel		

7. Click **OK**

The previous steps caused a lot of work to be done by VS. First, VS contacted your cloud service for retrieving its *metadata*. Examples of the information contained in the metadata are i) the name of the operations exported by the service, ii) the types of operation parameters iii) the communication protocol used by the service. Then VS used this metadata to automatically generate the code that will communicate with your cloud service (i.e. the service reference). You can see the generated classes by **double clicking** the reference to **MyNetDbService** inside the folder **Service References** of the project **MyNet.WebUI**. Then, simple inspect the content of the namespace **MyNet.WebUI.MyNetDbService**.



Once the service reference is generated, you can proceed with the development of the web page that will contain your Web UI. For this you have to open the **Default.aspx** file and replace its content with the following lines.

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"</p>
CodeBehind="Default.aspx.cs" Inherits="MyNet.WebUI._Default" %>
<asp:Content runat="server" ID="BodyContent" ContentPlaceHolderID="MainContent">
    <h2>List of Contacts</h2>
    <asp:GridView ID="ContactsGridView" runat="server"</pre>
        DataSourceID="MyNetDbServiceClient"
        AutoGenerateColumns="False" >
        <Columns>
            <asp:BoundField DataField="Id" HeaderText="Id" />
            <asp:BoundField DataField="Firstname" HeaderText="Firstname" />
            <asp:BoundField DataField="Lastname" HeaderText="Lastname" />
        </Columns>
    </asp:GridView>
    <asp:ObjectDataSource ID="MyNetDbServiceClient" runat="server"
        TypeName="MyNet.WebUI.MyNetDbService.MyNetDbServiceClient"
        SelectMethod="getContacts" >
    </asp:ObjectDataSource>
    <h2>Add New Contact</h2>
```



</asp:Content>

The more relevant elements in these lines are <asp:GridView>, <asp:ObjectDataSource> and <asp:Button>. The first one will generate a table containing the **Contacts** of the database **MyNetDB**. The table will contain 3 columns, each column bound to an attribute of the **Contact** type i.e., *Id*, *Firstname* and *Lastname*. The second one will create an object that will be used for filling the table. In this case the object is an instance of

MyNetDbServiceClient and the operation to be called is **getContacts()** that, as you may recall, produces a list of **Contact** instances. The last one will generate a button used for adding new contacts to the database.

The previous code defines the structure of the interface but does not define its behaviour. For defining the behaviour open the file **Default.aspx.cs** that resides inside **Default.aspx**.



Now replace the auto-generated code with the following code.

```
public partial class _Default : Page {
   protected void Page_Load ( object sender, EventArgs e ) { } // Method
   protected void AddContactButton_Click ( object sender, EventArgs e ) {
        // Service client
       var myNetDbService = new MyNetDbService.MyNetDbServiceClient();
       // Create Contact object based on info in Text fields
       var contact = new MyNetDbService.Contact {
            Firstname = FirstnameTextBox.Text,
            Lastname = LastnameTextBox.Text
        };
        // Send the New Contact to the service for insertion in DB
       myNetDbService.addContact ( contact );
        // Update the interface
        ContactsGridView.DataBind ();
        FirstnameTextBox.Text = "";
       LastnameTextBox.Text = "";
   } // Method
} // Class
```

The code contains the behaviour of the button when clicked. Basically it creates a **Contact** object with the information in the textboxs of the page and it send the contact object to the service.

At this point you should be able to test your Web UI locally. For this right click **MyNet.WebUI** > **Debug** > **Start new instance**.



After local testing, you can continue with the deployment of the Web UI on Windows Azure. Follow the instructions of the previous section for adding a new role to the project **MyNet.WindowsAzure**. The role has to be associated with the project **MyNet.WebUI**. Just be sure that the port assigned to the new role is **80** before publishing into Windows Azure. Finally use your web browser to interact with your Web UI deployed in Windows Azure. The URL that you will have to use is similar to

http://mynetproject<mark><ProjectID></mark>.cloudapp.net.