REST in .Net

This exercise introduces you to the development of simple Web Services in .NET.

For this exercise we you will need

- Windows 2008 Server, Windows 2008 Server R2, Windows Vista or Windows 7 (IIS must be installed)
- Internet Information Server 7
- Visual Studio 2008 or Visual Studio 2010
- WSCF.blue from thinktecture

You can either install the software on your own computer (faster, more responsive) or use Windows 2008 R2 server `sandstorm.cs.ut.ee`. Windows operating system and Visual Studio can be acquired through MSDN AA download site (more information at http://www.math.ut.ee/199208). We will also use Microsoft SQL Server in future practices. WSCF.blue can be downloaded from http://wscfblue.codeplex.com/ .

Part A – Turning SOAP Web Service into a simple REST Web Service

- Complete tutorial 5 (SOAP Web Services in .NET) part B.
- Change the interface definition (IITempConverter.cs) as following:
  - WCF 3.5 can automatically only extract strings from URI and string or integer values from parameters section. The original method definitions used floats, thus they need to be changed. Change the method definitions so that they would take only strings or integers as arguments.

You can use `float f = float.Parse(string s, …)` *to get float value from string.*

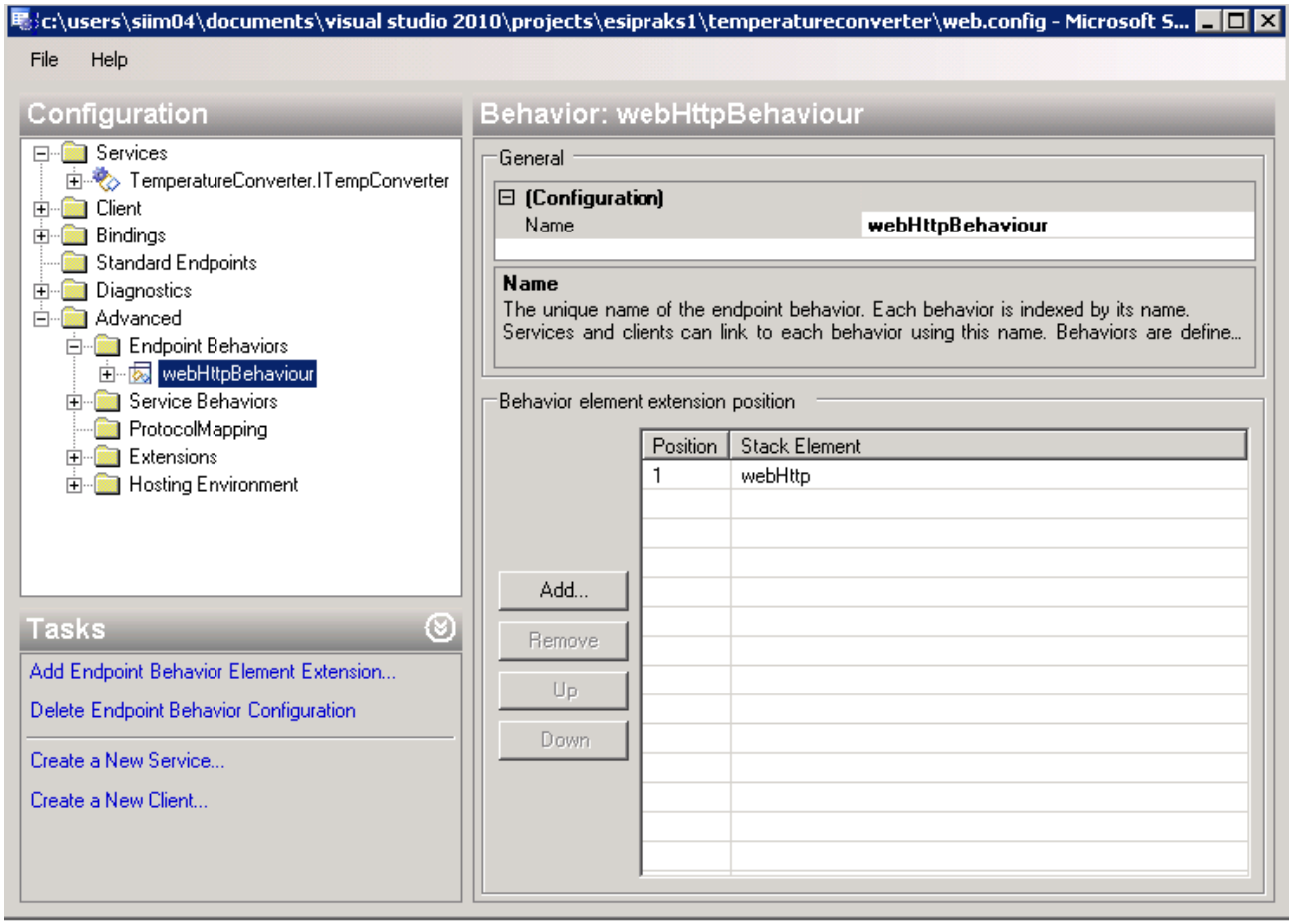
  - Add attribute WebGet to the methods.

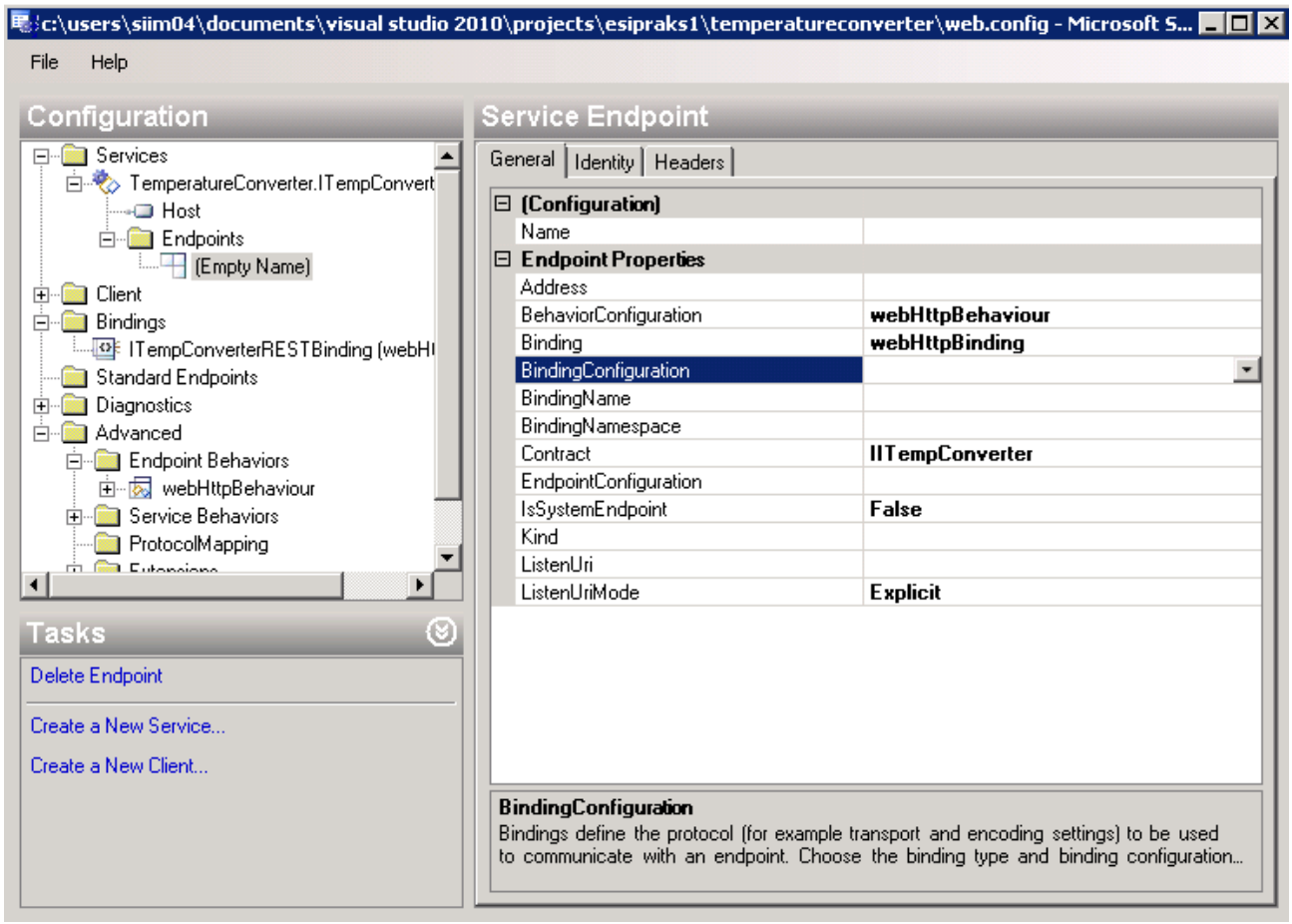
The attribute definition for CtoF method should look like this:

```
[System.ServiceModel.Web.WebGet(
       UriTemplate="/CtoF/temp={temp}",
       BodyStyle= System.ServiceModel.Web.WebMessageBodyStyle.Bare)]
```

You can also use WebInvoke attribute instead WebGet attribute, but then you need to specify access method.

- Modify the implementing class accordingly.
- Edit WCF configuration in your web.config file (You can use WCF Service Configuration Editor for doing that).
  - Create a new endpoint behaviour called webHttpBehaviour with webHttp element.
  - Set the endpoint Binding to be webHttpBinding and BehaviorConfiguration to webHttpBehavior.





- Save and test your service. You can test the service by sending GET requests to http://localhost:????/ITempConverter.svc/CtoF?temp=XXX and

http://localhost:????/ITempConverter.svc/FtoC?temp=XXX where ???? is your testing port number and XXX the temperature you want to be converted.

Part B – Creating a REST consumer

- Pick a public and free weather data provider API (check out API catalogue at http://www.programmableweb.com/apis/directory/1?apicat=Weather&protocol=REST ; for example, Weather Underground offers public service: http://wiki.wunderground.com/index.php/API_-_XML).
- Create an application, that implements following workflow:
  - The application asks user for a location (e.g. Town and Country names)
    - A general tutorial for creating a WPF application can be found at http://wpftutorial.net/HelloWPF.html , a general tutorial for creating a ASP.NET Web Application can be found at http://www.asp.net/get-started .
    - Creating a REST consumer is accomplished similar to creating a SOAP consumer. That is, you need to:
      - Create an interface with contract and WebGet or WebInvoke attributes.
      - Define a new behaviour and a new **client endpoint** using the interface as the contract (similar to the way you created the REST service).
      - Use ChannelFactory (http://msdn.microsoft.com/en-us/library/bb908674.aspx) to create a proxy/client for the REST service.
      - Use the proxy/client to call the service.
    - An introduction to WCF REST programming model is available at http://msdn.microsoft.com/en-us/library/bb472530.aspx. WCF support is limited in the Client and Compact frameworks. So make sure you target full .NET Framework instead of Client framework when creating a desktop application (can be set in project properties).
  - The application asks for the weather at the user specified location from the weather data provider.
    - You can process XML returned by the weather service either using LINQ (XML datasource) http://msdn.microsoft.com/en-us/library/bb308960.aspx or XPath http://support.microsoft.com/kb/308333 . The latter is preferable for this simple case. There are lots of other options available as well, but XPath is the simplest for accessing single values in large documents. XPath documentation is available at http://www.w3.org/standards/techs/xpath#w3c_all .
  - The application calls the service you created in part A to convert the temperatures returned by the weather API. If the weather service gives you both vales, compare the values you got with calculation with the weather service's values.
  - The application displays the user the weather data both in Celsius and Fahrenheit.

Optional – Creating a REST data manipulation service

- Implement a database manipulation service (e.g. modify application created in tutorial 5 (SOAP Web Services in .NET) Part C) in REST (you need to use WebInvoke attribute for DELETE, PUT, and POST methods).

Optional – Publish your SQL Server database through SOAP or REST interface

- See tutorial at http://www.netrostar.com/Tutorials-66-How%20to%20create%20HTTP%20Endpoint%20in%20SQL%20Server%202005 .

Additional resources

- **eLearning**: http://learning.microsoft.com
  - Collection 6261: Developing Rich Experiences using Microsoft .NET Framework 3.5 & Visual Studio 2008
- **Free eBook „Introduction to LINQ“:** http://introducinglinq.com/
- **Microsoft IT Academy:** https://itacademy.microsoftelearning.com
  - Collection 6461: Visual Studio 2008: Windows Communication Foundation
  - Collection 6463: Visual Studio 2008 ASP.NET 3.5
  - Collection 6464: Visual Studio 2008 ADO.NET 3.5