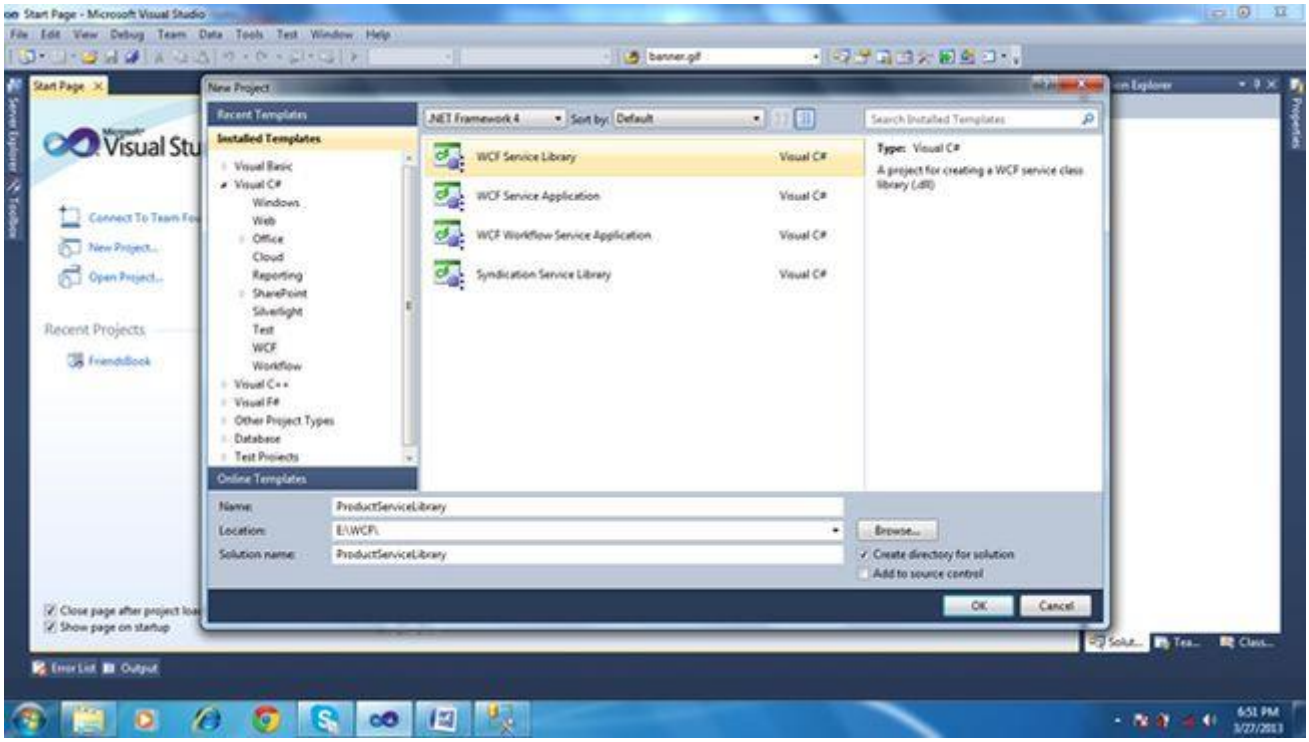
Creating WCF Service, IIS Hosting and Consuming

How to create WCF service with IIS hosting?

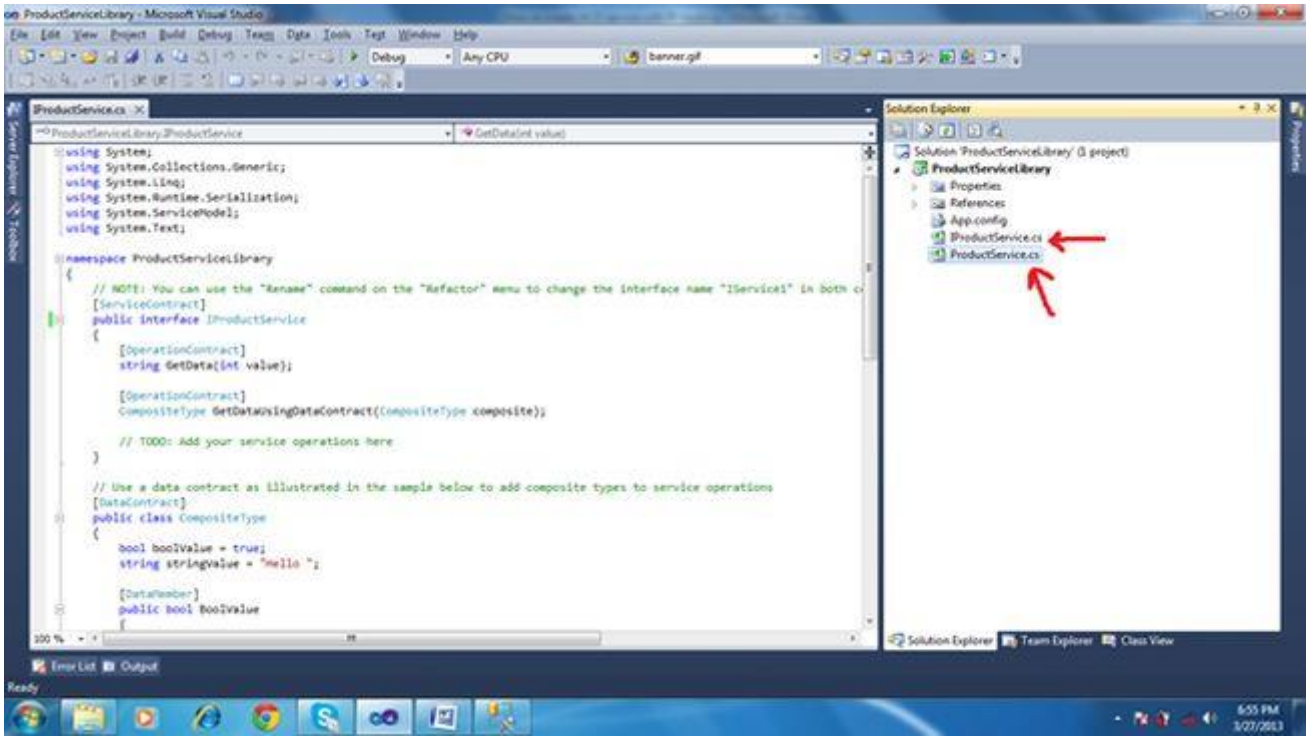
In Visual Studio go to "File" -> "New" -> "Project..." then select "C#" -> "WCF" -> "WCF Service Library".

Now set the name of the project as "ProductServiceLibrary" then click "OK".



In the next step change the name of the files:

From "IService.cs" to "IProductService.cs"
From "Service.cs" to "Product Service.cs"

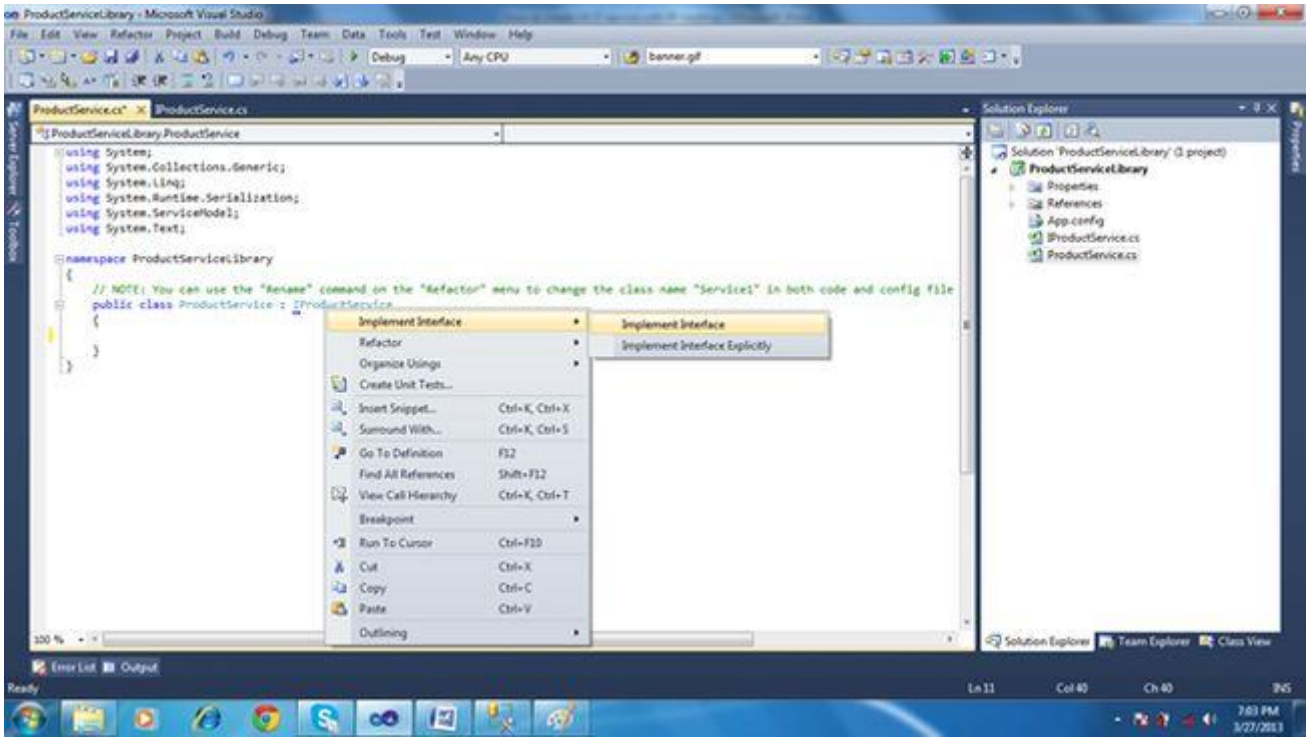


Now in "IProductService.cs" write the following code to implement it with the Service Class i.e. "ProductService.cs":

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace ProductServiceLibrary
{
    [ServiceContract]
    public interface IProductService
    {
        [OperationContract]
        List<Product> GetAllProduct();

        [OperationContract]
        Product GetProductByID(string ProductId);

    }


    [DataContract]
    public class Product
    {
        [DataMember]
        public int ProductID
        {
```

```
            get;
            set;
        }

        [DataMember]
        public string ProductName
        {
            get;
            set;
        }

        [DataMember]
        public string QuantityPerUnit
        {
            get;
            set;
        }

        [DataMember]
        public decimal UnitPrice
        {
            get;
            set;
        }
    }
}
```

After creating the interface we must implement it and must write code for the service.



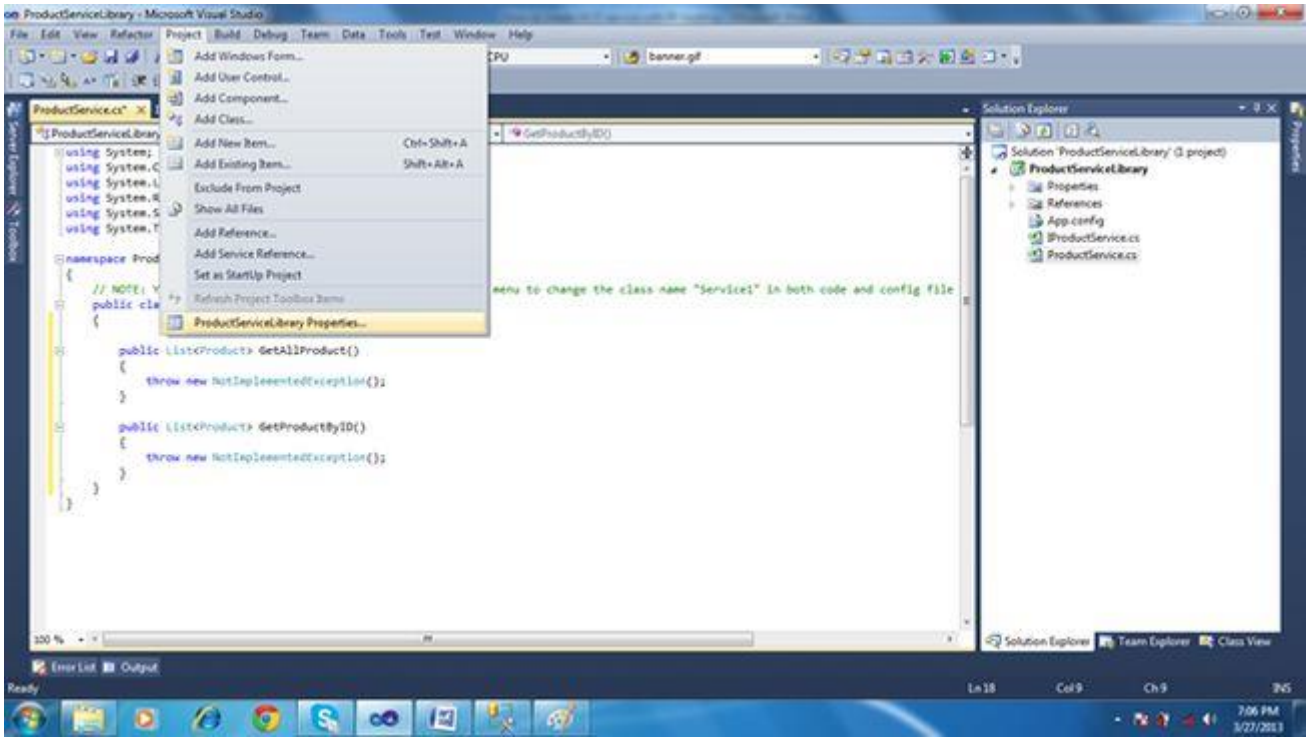

```
using System;
using System.Collections.Generic;
```

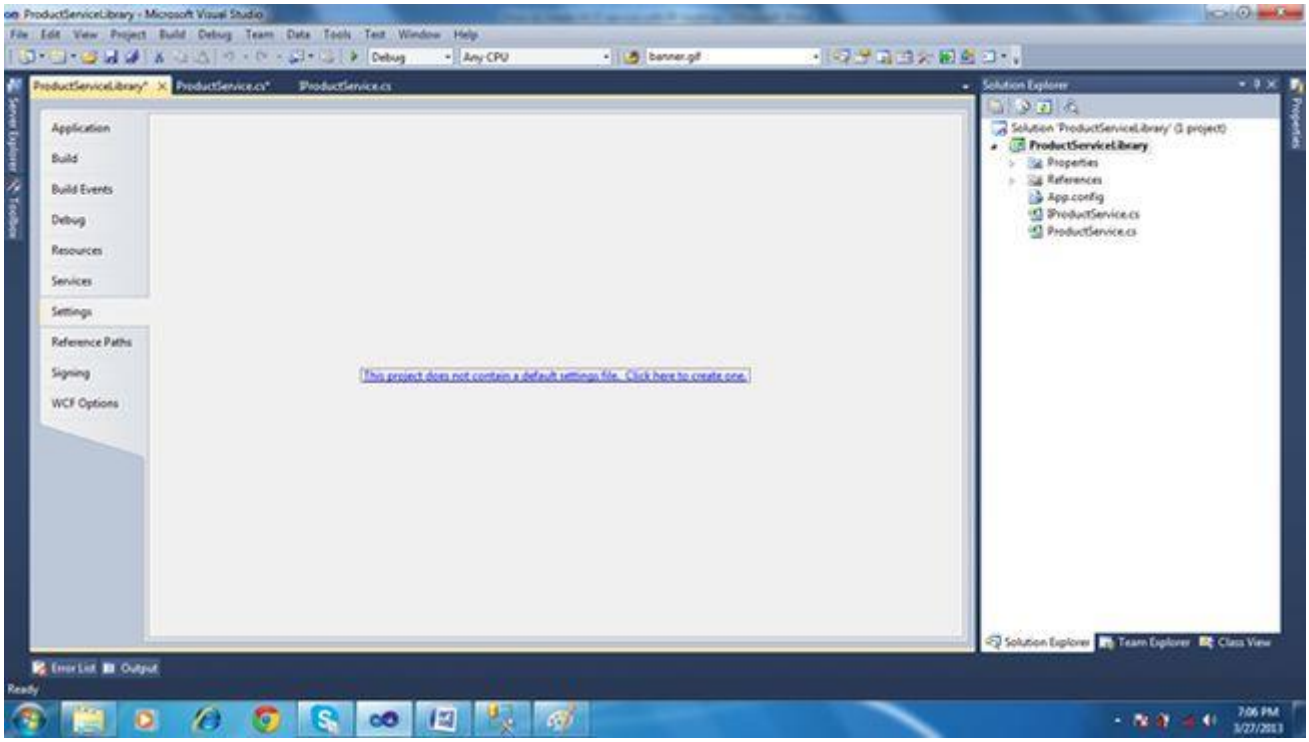
```
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace ProductServiceLibrary
{
    public class ProductService : IProductService
    {

        public List<Product> GetAllProduct()
        {
            throw new NotImplementedException();
        }

        public Product GetProductByID(string ProductId)
        {
            throw new NotImplementedException();
        }
    }
}
```

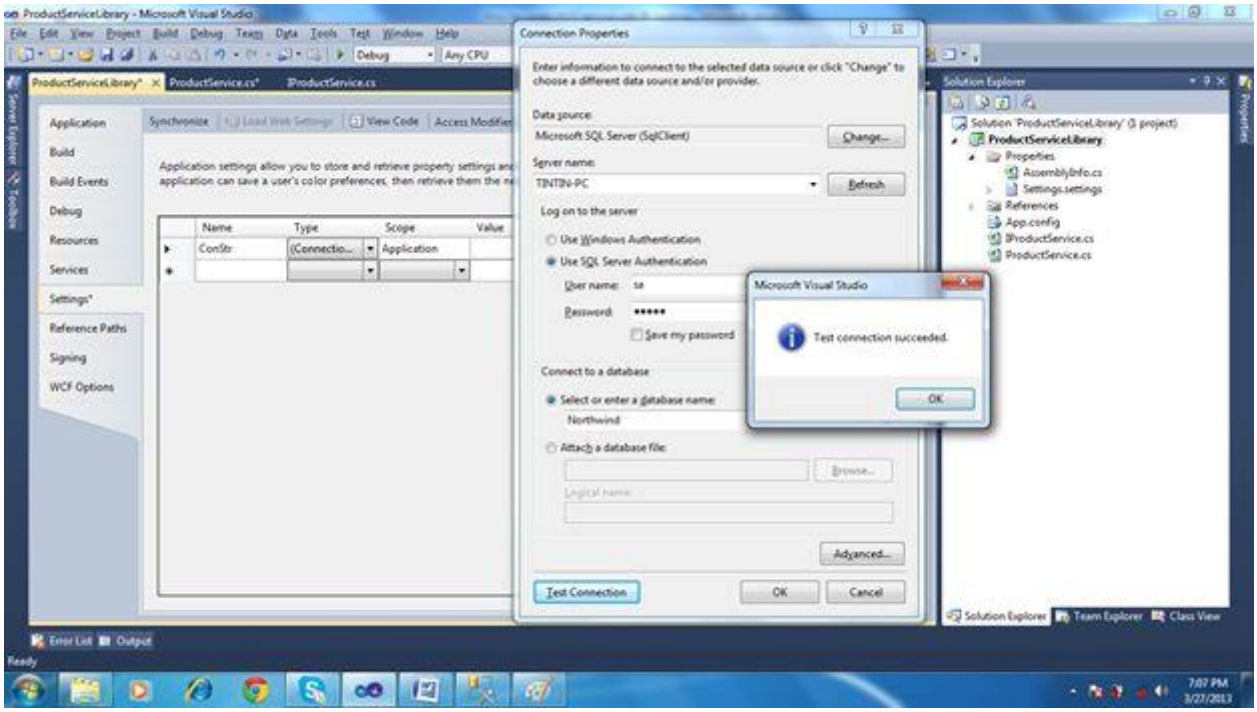
Now we have implemented the IProductService interface with the Service.

We must write the code to get all the products from the database and one product detail by product ID.

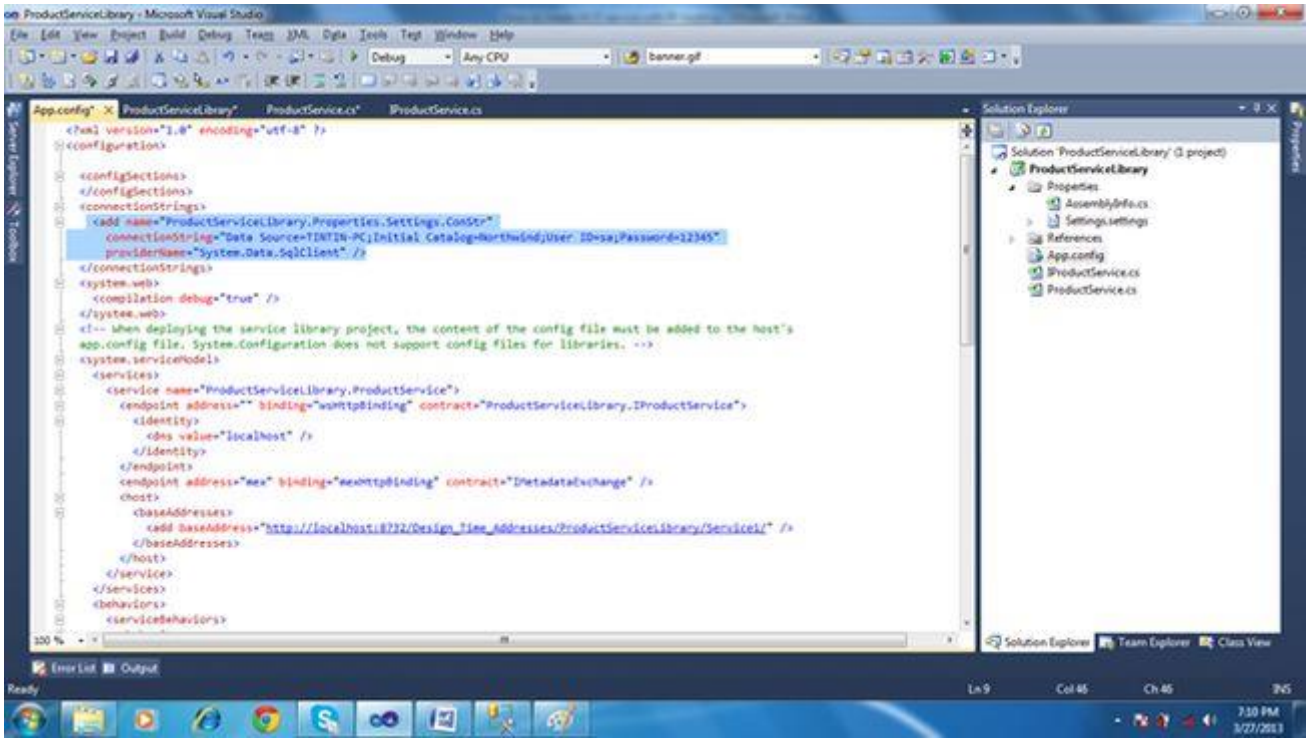
But before this we will set the database connection string in the Project property.







After clicking "Ok" open the "App.config" file and we will see the database connection string.



After setting the connection string in the project properties, now we will write code to access the data from the database in the methods we implemented.

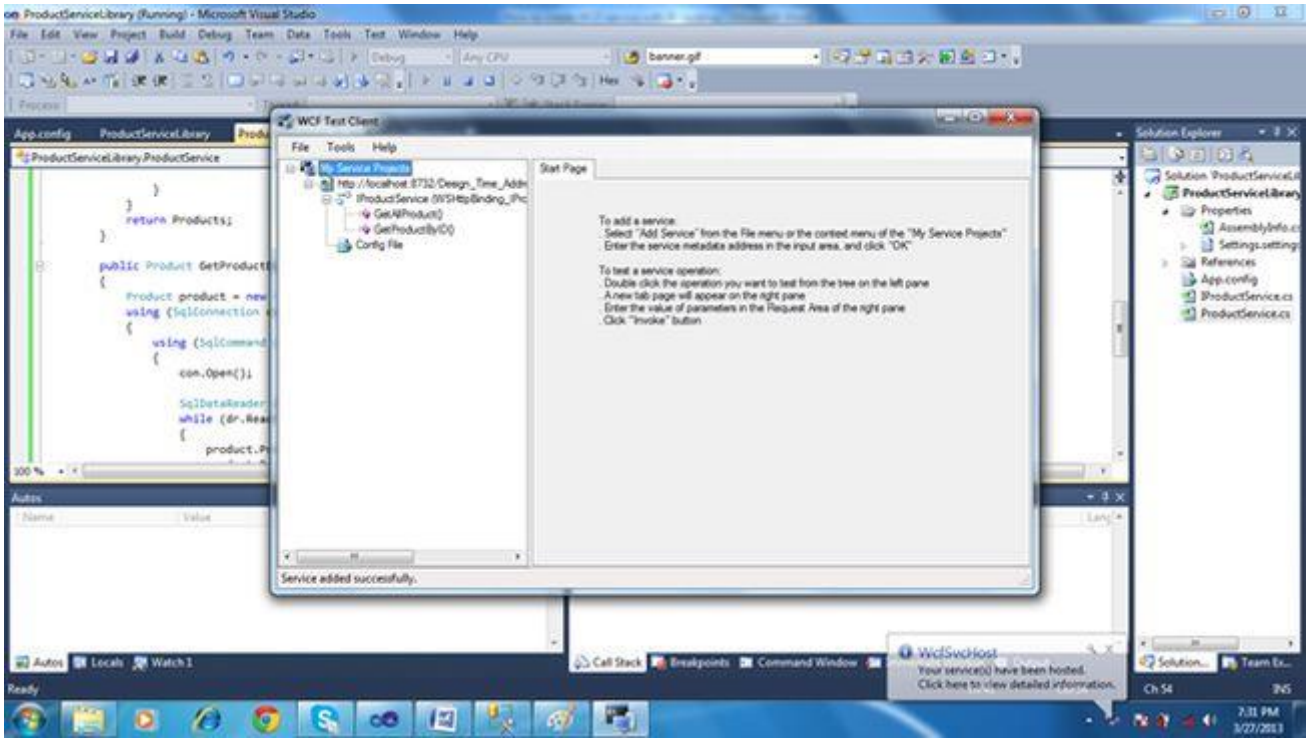
So here is the code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using System.Data.SqlClient;

namespace ProductServiceLibrary
{
   public class ProductService : IProductService
   {

      public List<Product> GetAllProduct()
      {
         List<Product> Products = new List<Product>();
         using (SqlConnection con = new SqlConnection(Properties.Settings.Default.ConStr))
         {
            using (SqlCommand cmd = new SqlCommand("Select
ProductId,ProductName,QuantityPerUnit,UnitPrice from Products", con))
            {
               con.Open();

               SqlDataReader dr = cmd.ExecuteReader();
               while (dr.Read())
               {
```

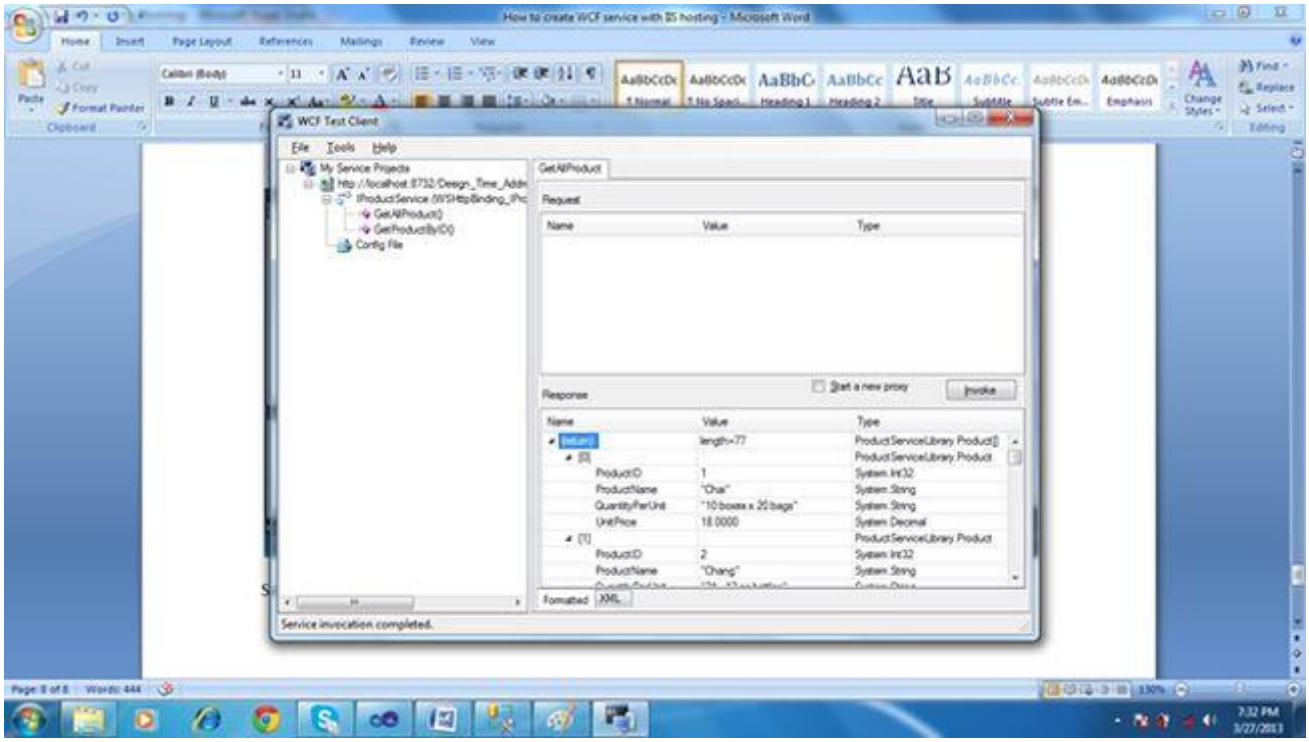
```
                    Product product = new Product();
                    product.ProductID = dr.GetInt32(0);
                    product.ProductName = dr.GetString(1);
                    product.QuantityPerUnit = dr.GetString(2);
                    product.UnitPrice = dr.GetDecimal(3);

                    Products.Add(product);
                }

            }
        }
        return Products;
    }

    public Product GetProductByID(string ProductId)
    {
        Product product = new Product();
        using (SqlConnection con = new SqlConnection(Properties.Settings.Default.ConStr))
        {
            using (SqlCommand cmd = new SqlCommand("Select
ProductId,ProductName,QuantityPerUnit,UnitPrice from Products", con))
            {
                con.Open();

                SqlDataReader dr = cmd.ExecuteReader();
                while (dr.Read())
                {
                    product.ProductID = dr.GetInt32(0);
                    product.ProductName = dr.GetString(1);
                    product.QuantityPerUnit = dr.GetString(2);
                    product.UnitPrice = dr.GetDecimal(3);

                }

            }
        }
        return product;
    }
  }
}
```

Now our first WCF has been created. It is time to test our WCF Service Application; just press the F5 button.

The WCF Test Client will run.



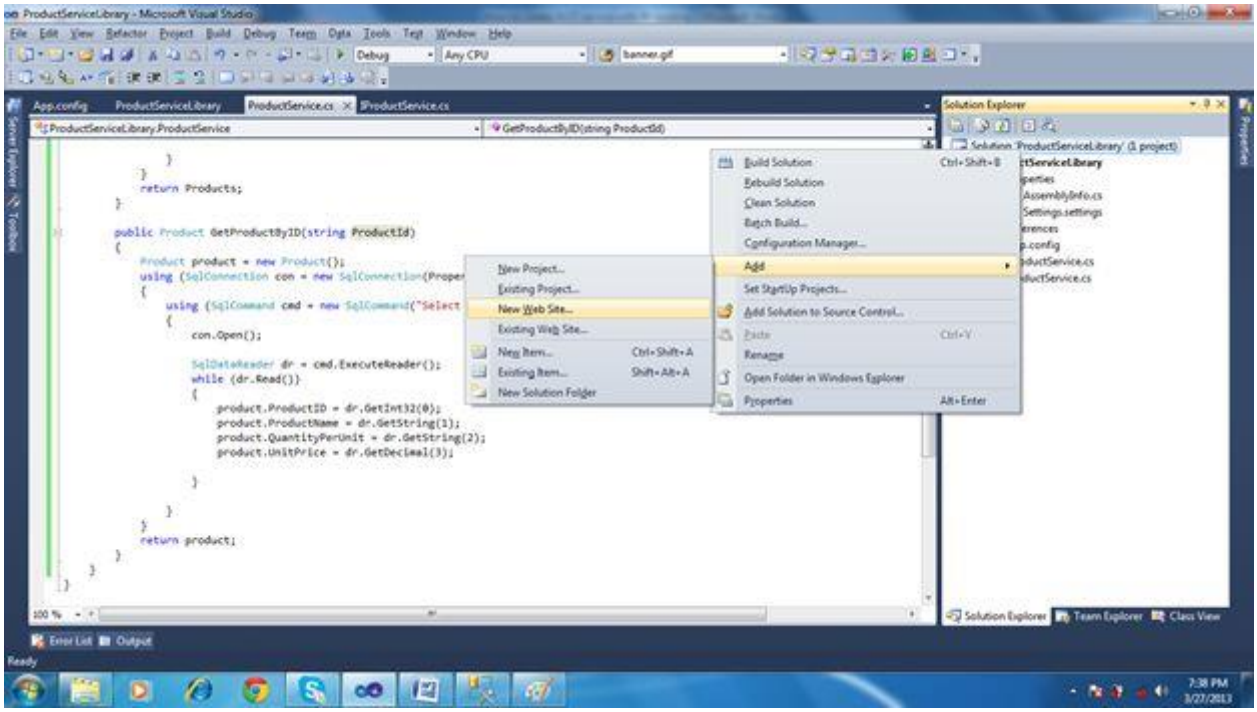
Select the method "GetAllProduct" and click "Invoke".

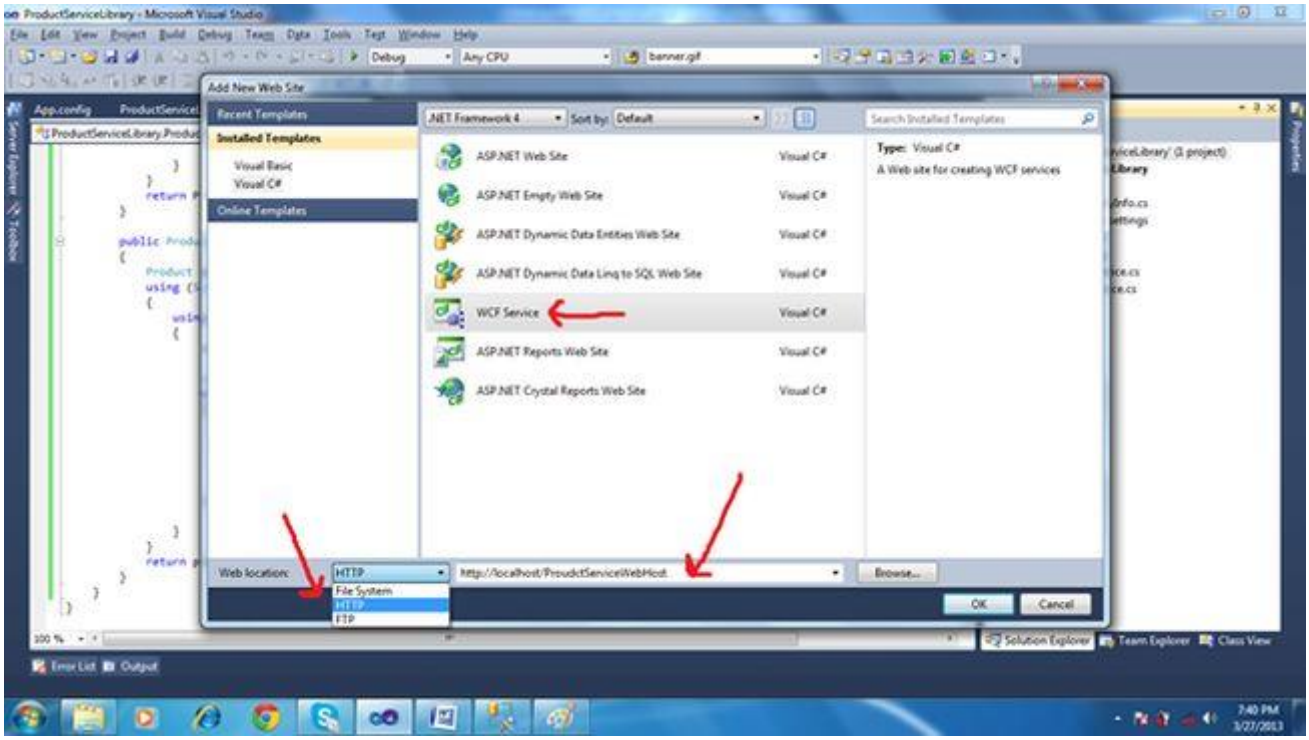


The preceding screen is the indication that the f WCF Service was created successfully.

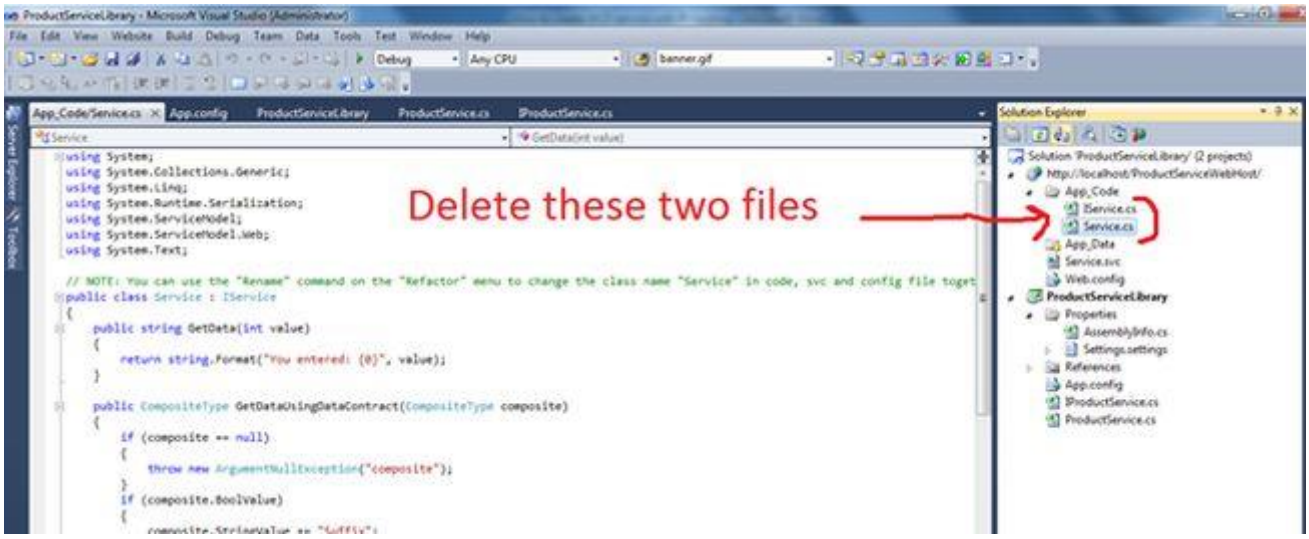
Now our second step is just to host this WCF Application using IIS.

Click on the solution then right-click then select "Add" -> "New Website".

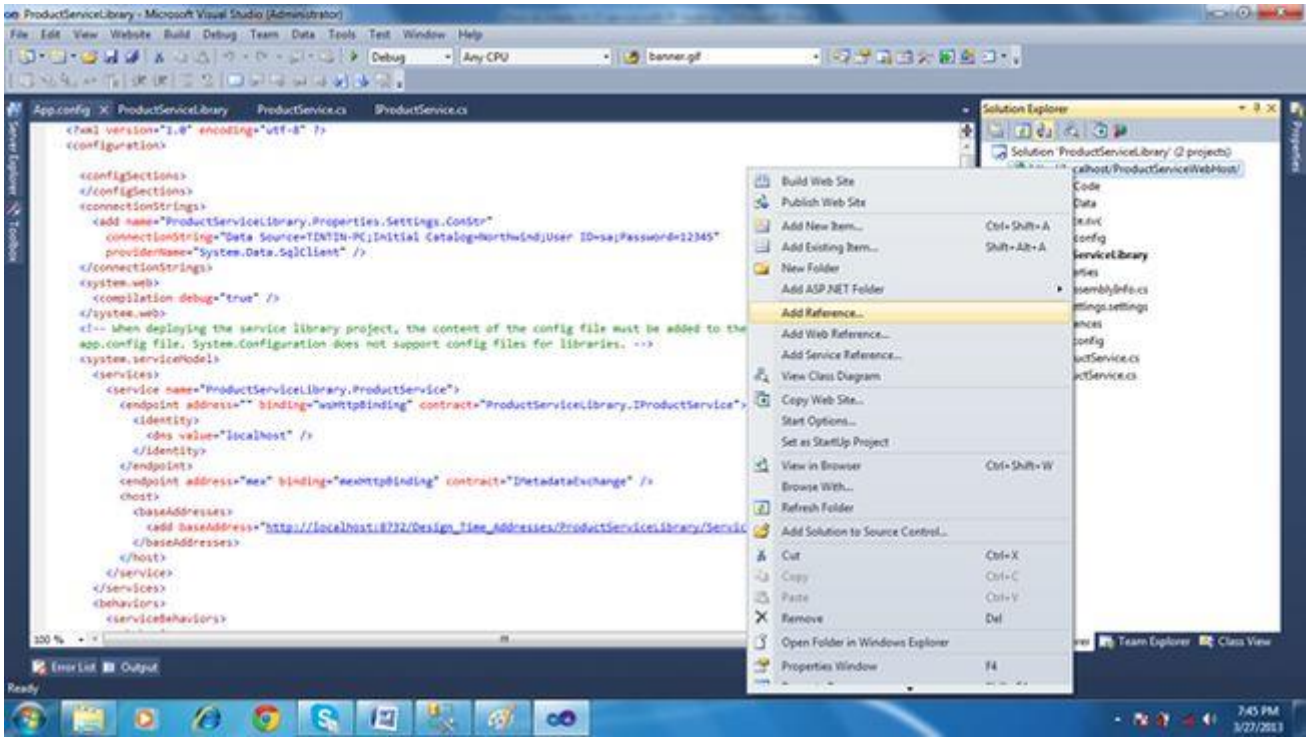


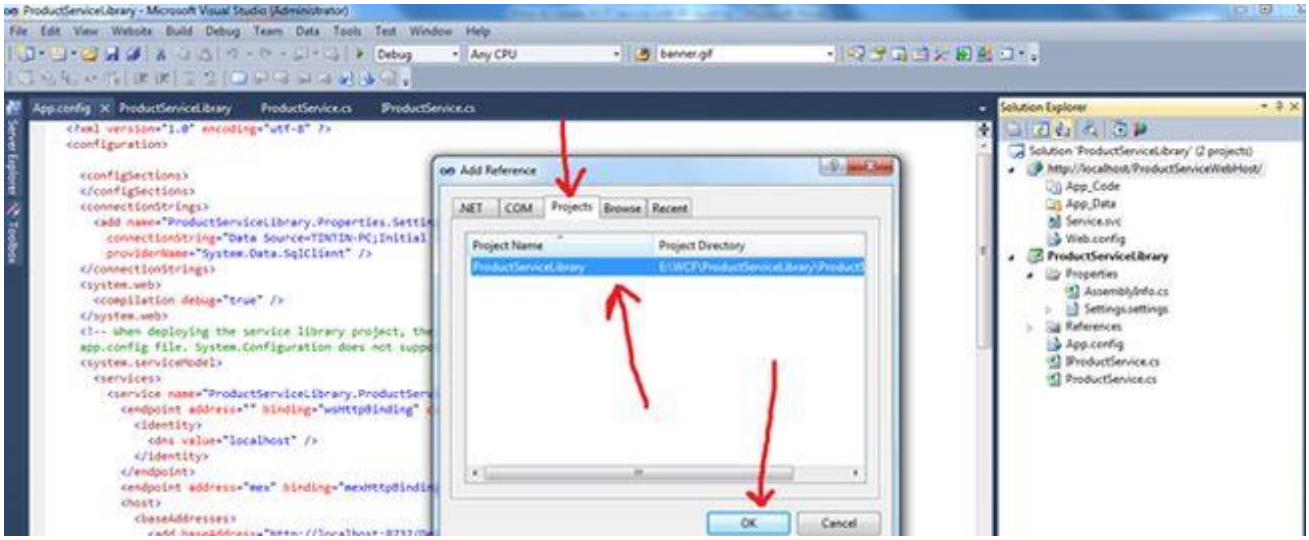


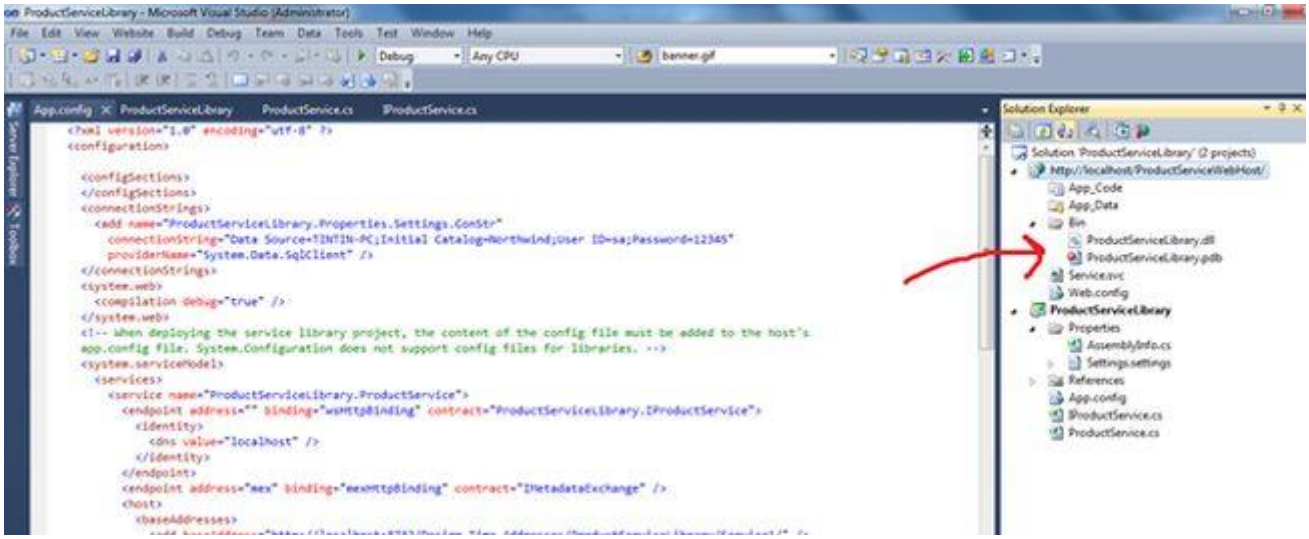
After clicking "OK" the Website application will be added to the solution. Then after it has been added it, two extra files will get generated in the project, i.e. "IService.cs" & "Service.cs". We have, however, already created these file in the WCF Library, so now we will delete them and add the reference to "ProductServiceLibrary.dll".



Add the reference to the ProductServiceLibrary.dll in this project.





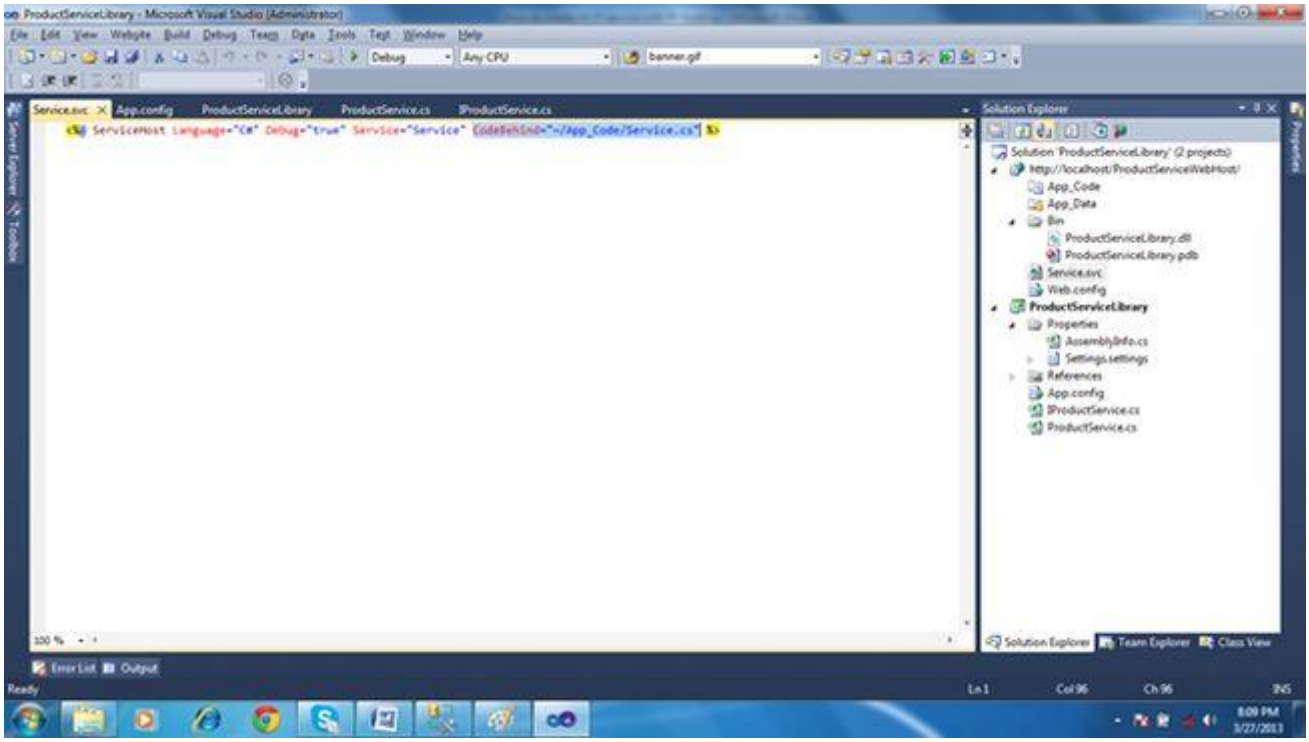


Now open the Service.svc file.

Delete the selected code (in the following image) and the reason is very simple, because in our ProductServiceLibrary.dll we created the ProductService.cs and in the current project we have deleted the Service.cs file from the App_code folder.

And replace the current line with this line of code:

```
<%@ ServiceHost Language="C#" Debug="true" Service="ProductServiceLibrary.ProductService
" %>
```



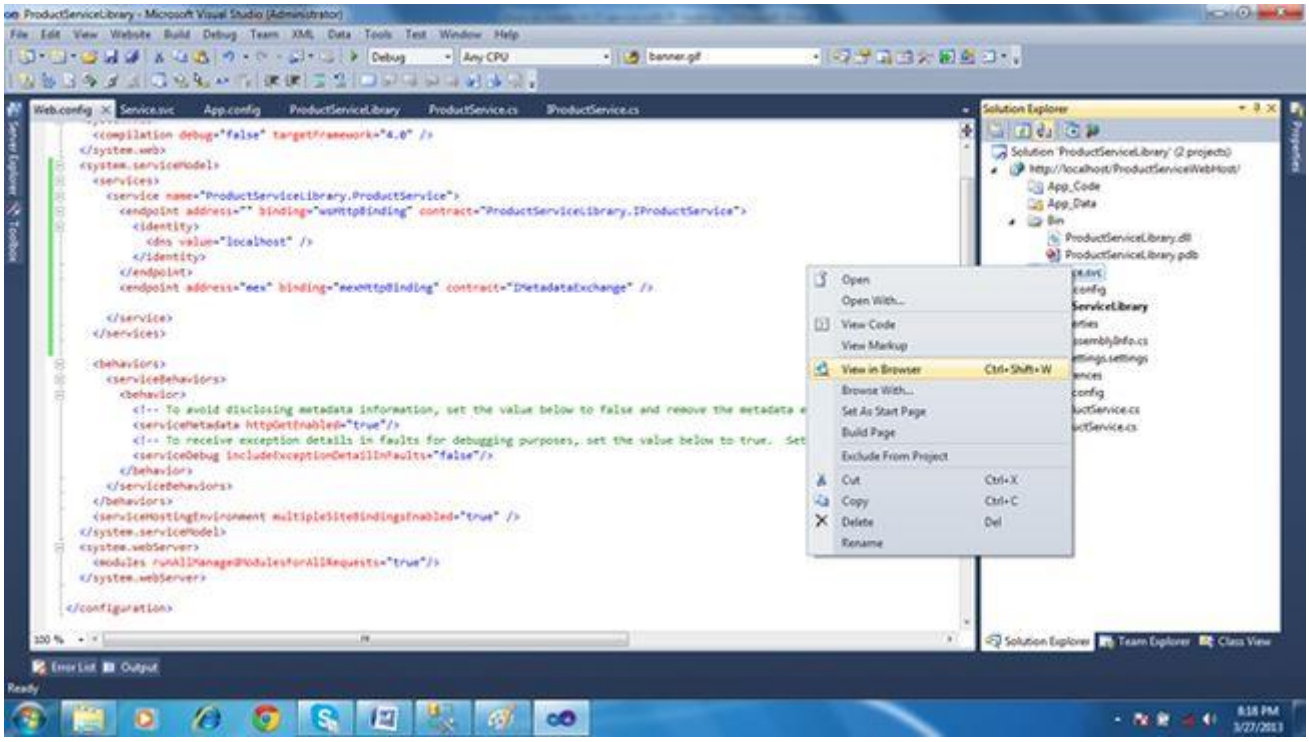
We have altered the .svc file and it is time to add the End Points to Web.Config, as in:

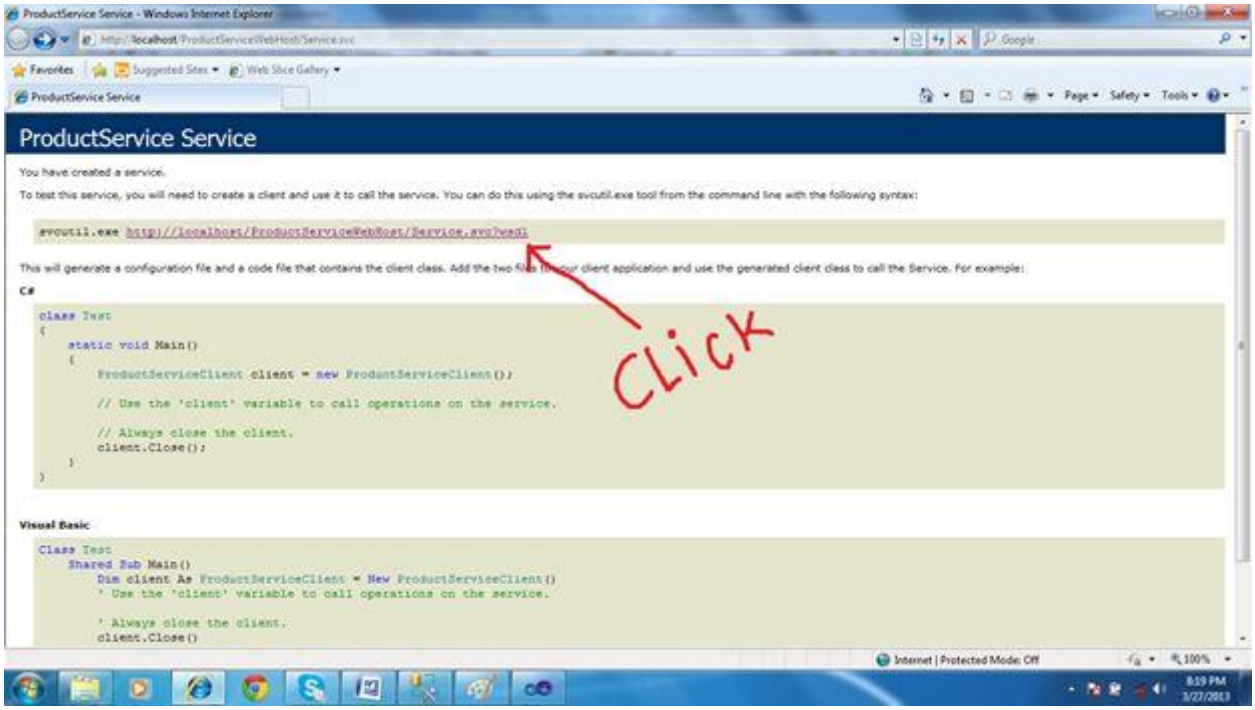
```
<?xml version="1.0"?>
<configuration>

  <system.web>
    <compilation debug="false" targetFramework="4.0" />
  </system.web>
  <system.serviceModel>
    <services>
      <service name="ProductServiceLibrary.ProductService">
        <endpoint address="" binding="wsHttpBinding" contract="ProductServiceLibrary.IProduct
Service">
          <identity>
            <dns value="localhost" />
          </identity>
        </endpoint>
        <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

      </service>
    </services>

    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- To avoid disclosing metadata information, set the value below to false and remove
the metadata endpoint above before deployment -->
          <serviceMetadata httpGetEnabled="true"/>
          <!-- To receive exception details in faults for debugging purposes, set the value below to
true.  Set to false before deployment to avoid disclosing exception information -->
          <serviceDebug includeExceptionDetailInFaults="false"/>
```

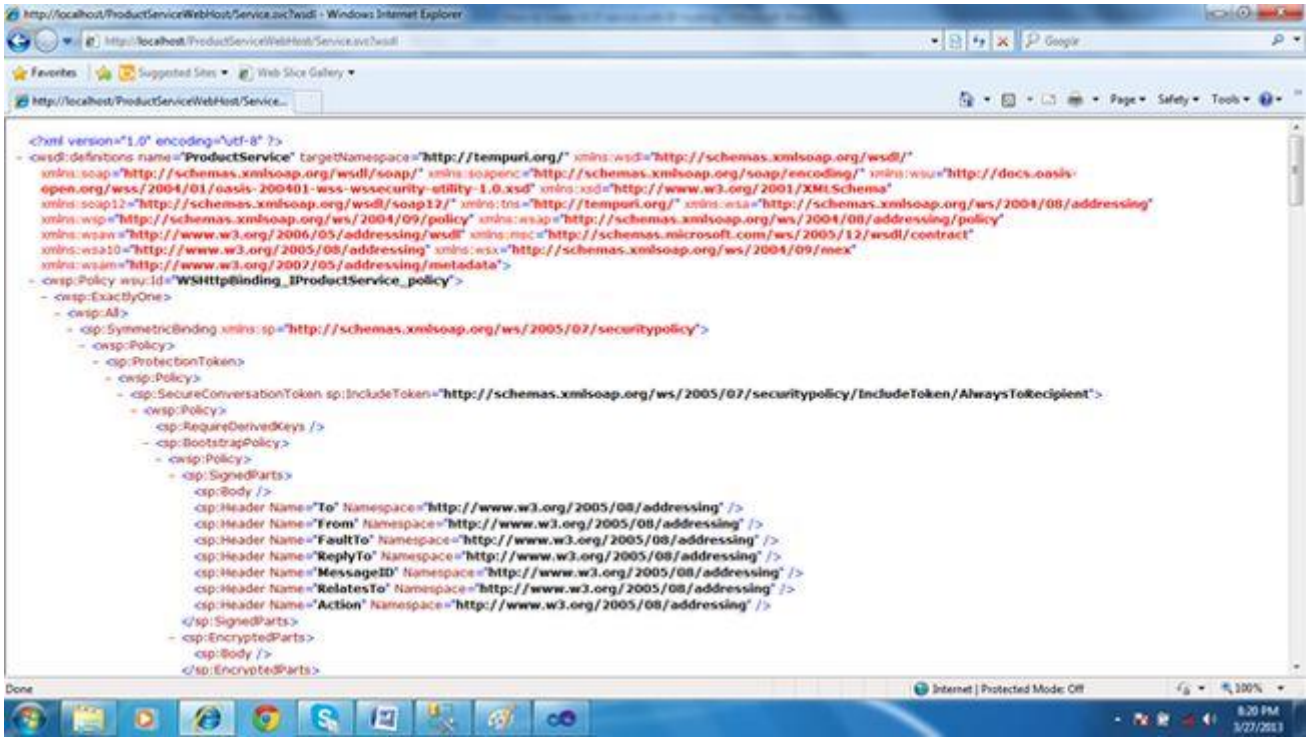
```
      </behavior>
    </serviceBehaviors>
  </behaviors>
  <serviceHostingEnvironment multipleSiteBindingsEnabled="true" />
</system.serviceModel>
<system.webServer>
  <modules runAllManagedModulesForAllRequests="true"/>
</system.webServer>

</configuration>
```

After adding the End Points in the web.config file, our hosting task is finished. Now it is time to test the Hosting Application.

For that please use the following steps in the following image:



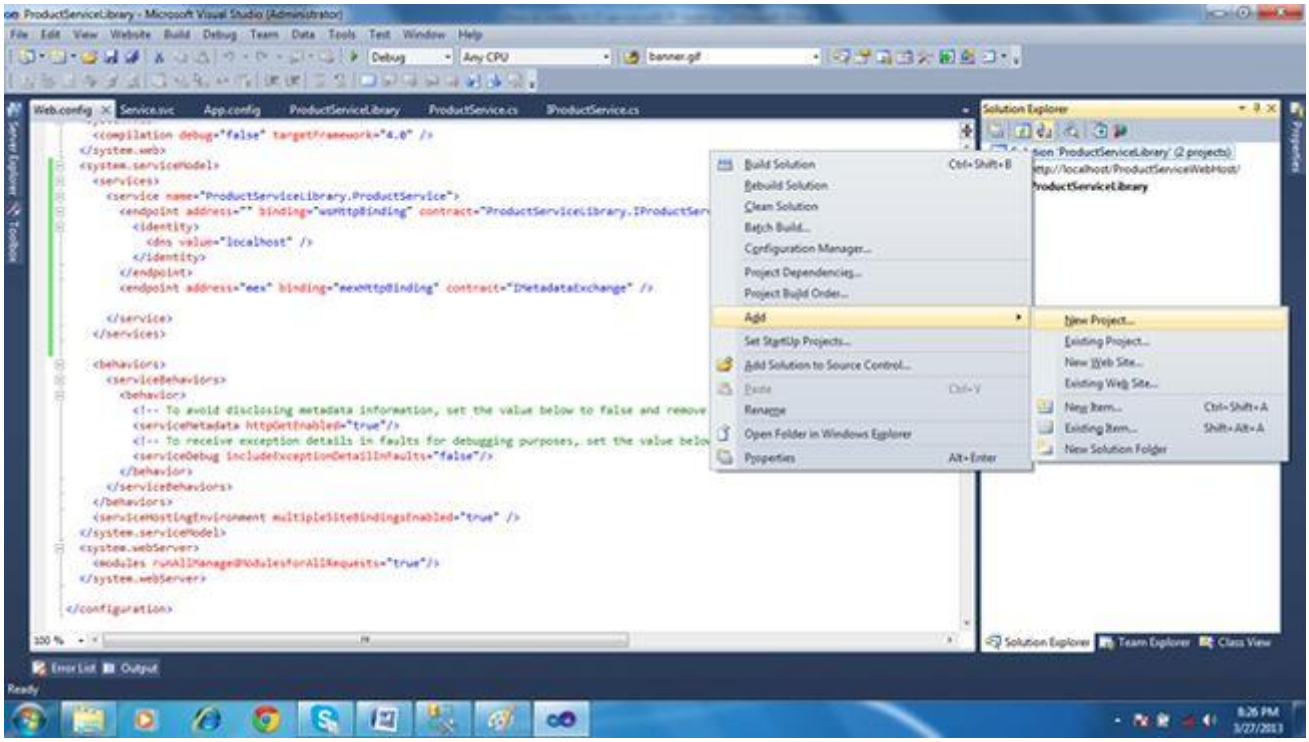


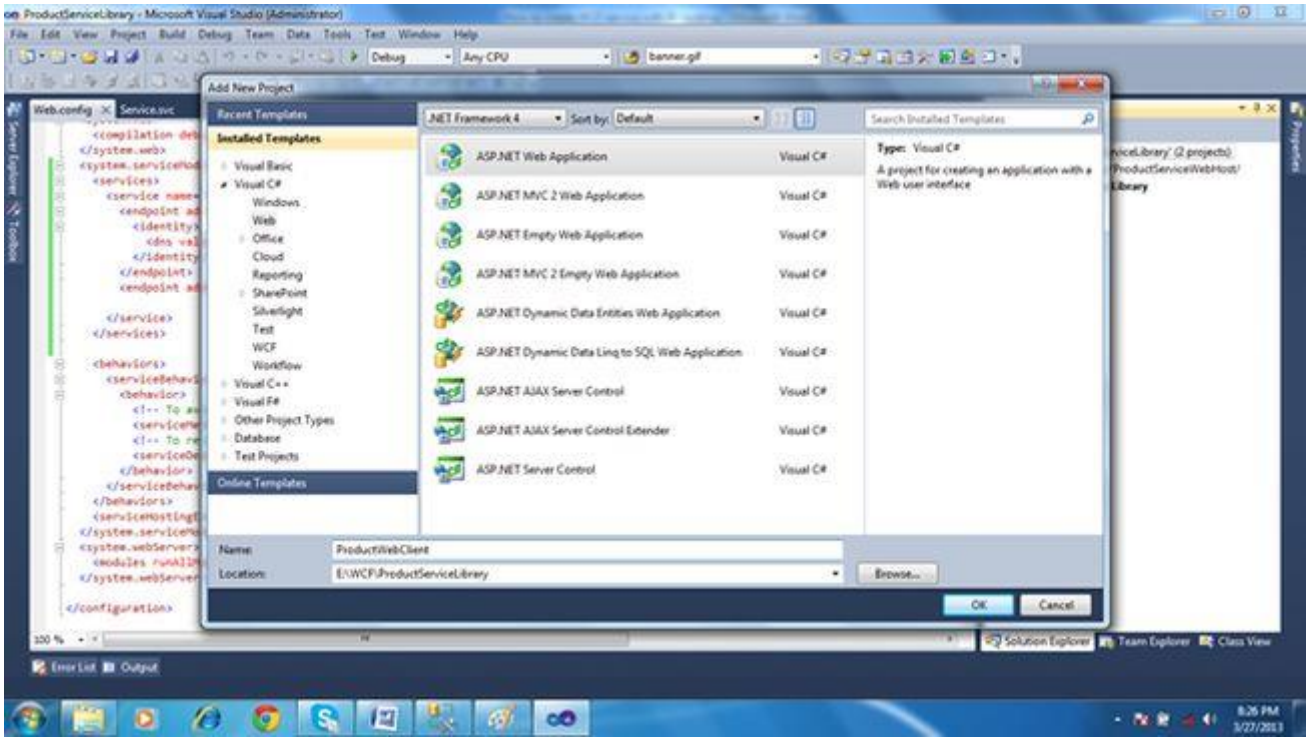


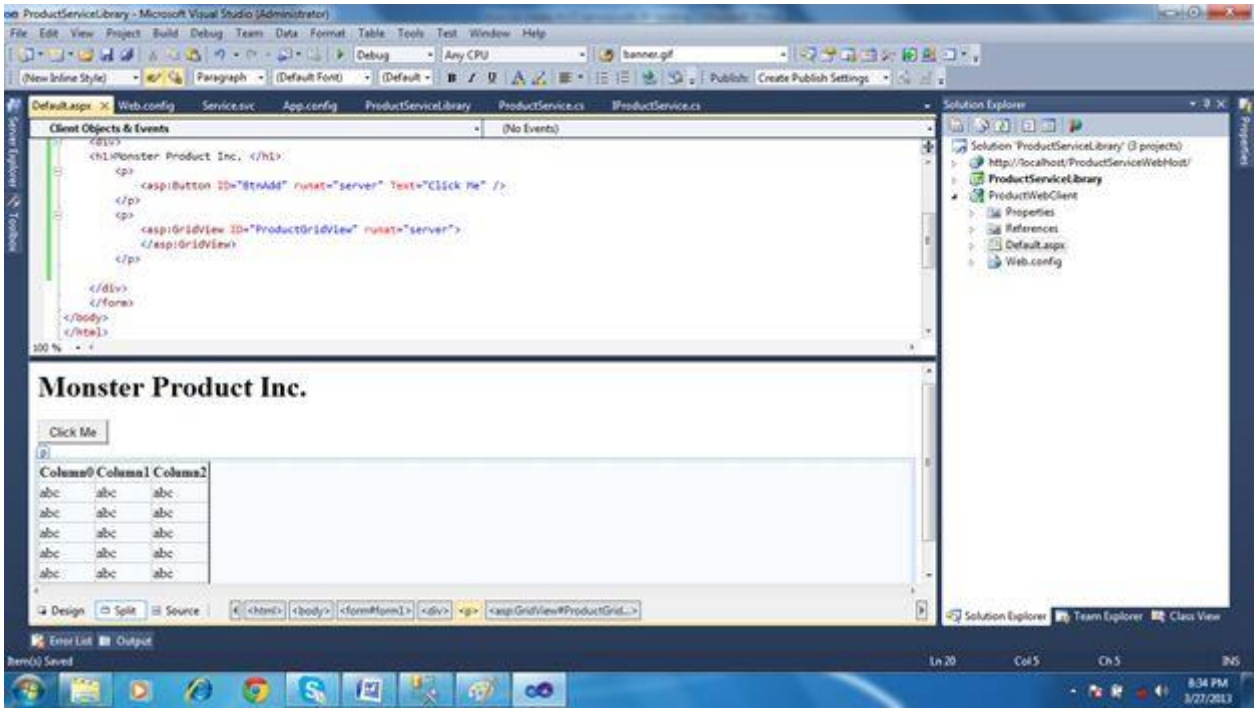
The preceding color full page is the indication of successfully hosting WCF in IIS.

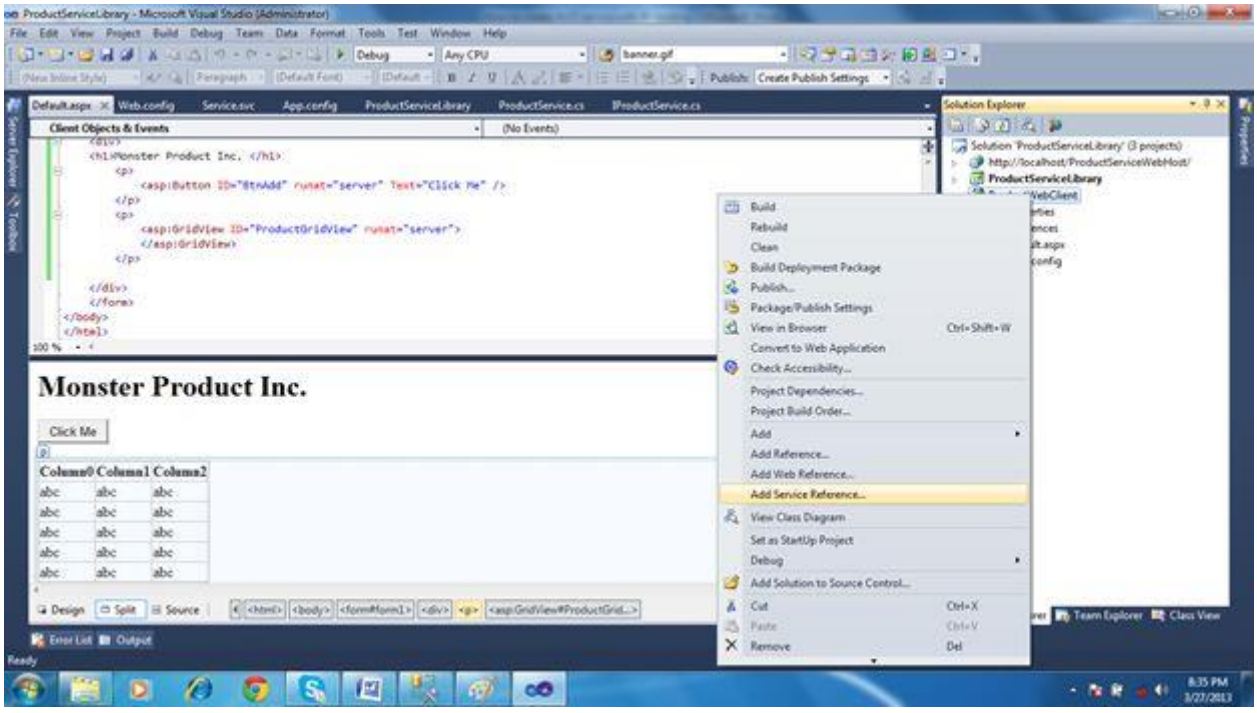
Now our next step is to consume this WCF IIS hosted service in a client application and this is a very easy task.

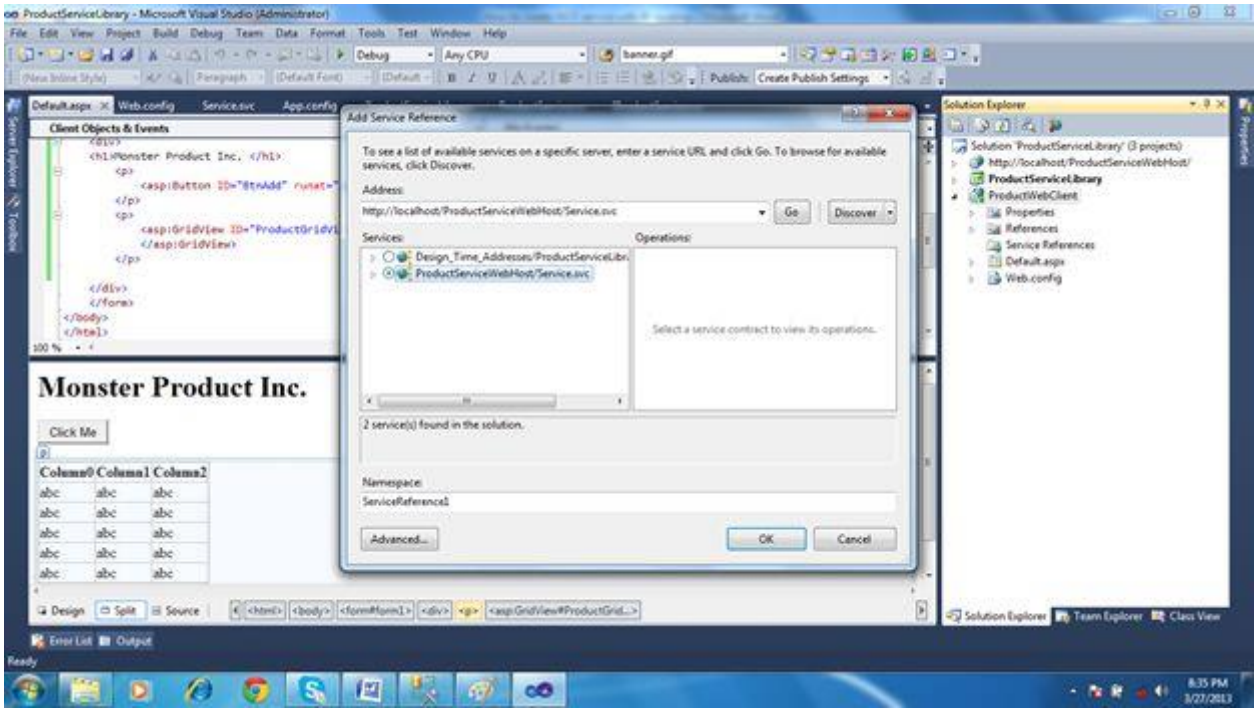
So let's proceed to our last step.

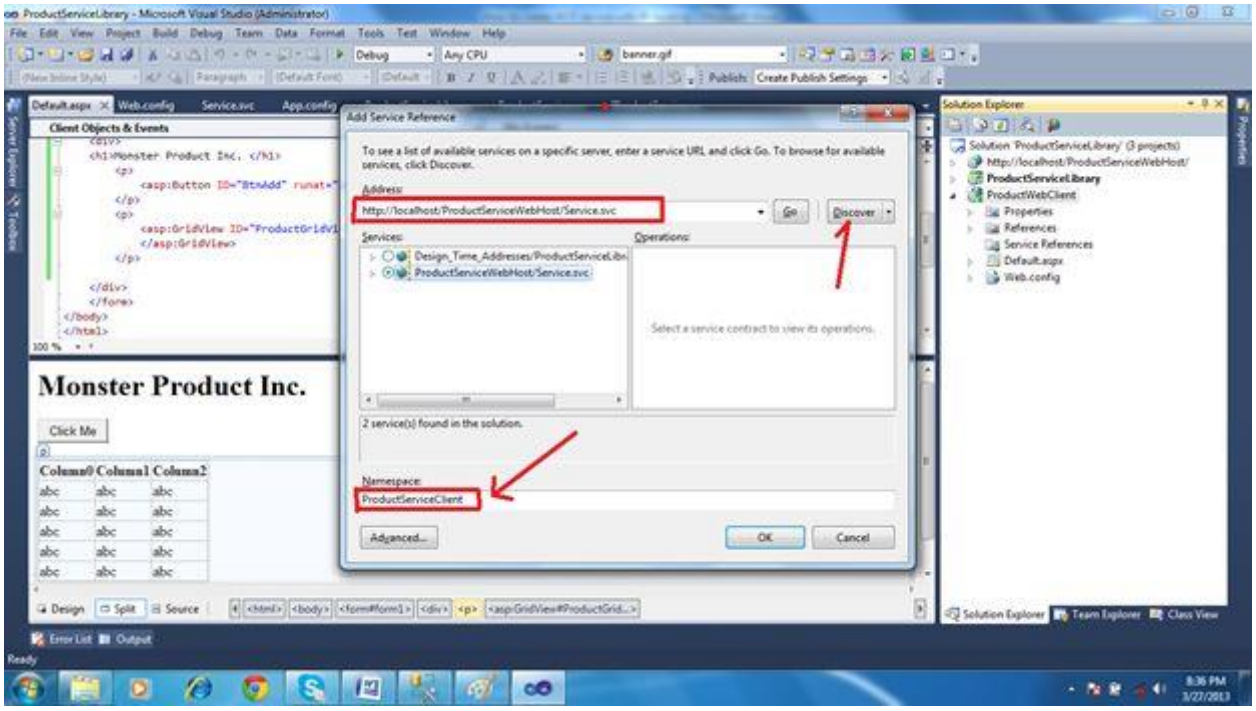

ProductServiceLibrary - Microsoft Visual Studio (Administrator)
Web.config
Service.svc
App.config
ProductServiceLibrary
ProductService.cs
IProductService.cs
Solution Explorer
Build Solution
Rebuild Solution
Clean Solution
Batch Build...
Configuration Manager...
Project Dependencies...
Project Build Order...
Add
Set StartUp Projects...
Add Solution to Source Control...
Paste
Rename
Open Folder in Windows Explorer
Properties
New Project...
Existing Project...
New Web Site...
Existing Web Site...
New Item...
Existing Item...
New Solution Folder
Error List
Output
Ready

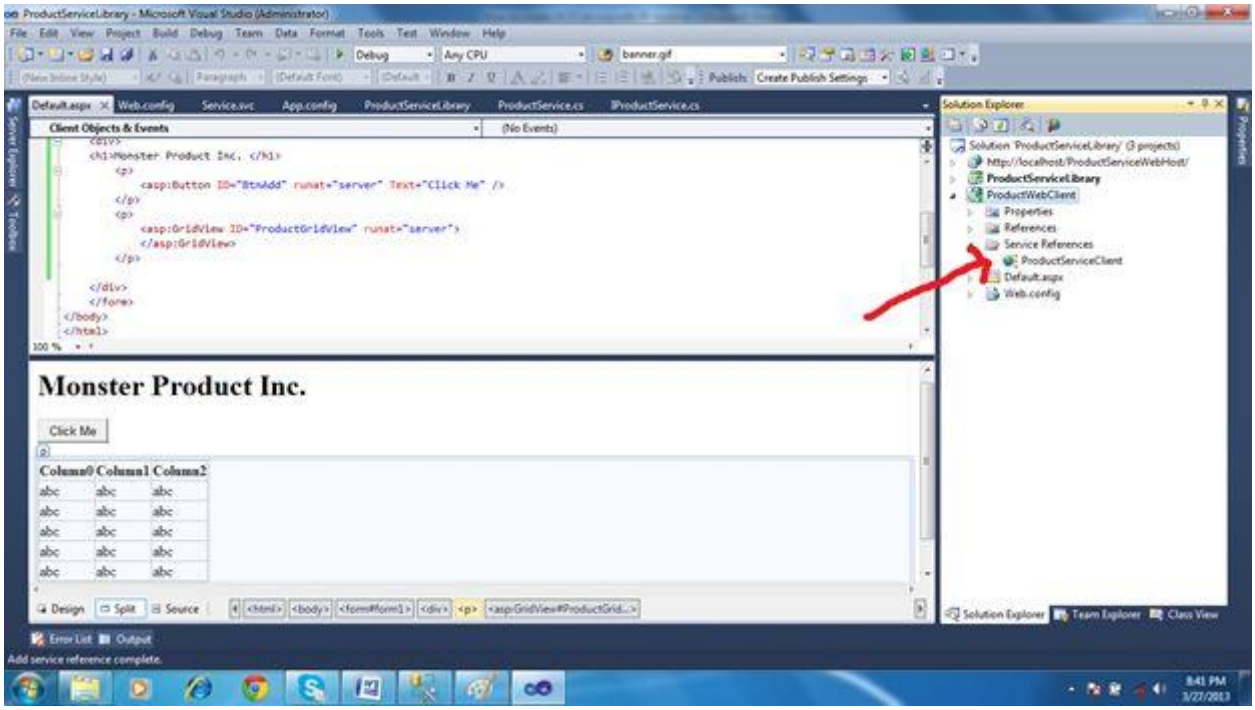

ProductServiceLibrary - Microsoft Visual Studio (Administrator)
Add New Project
Recent Templates
Installed Templates
Visual Basic
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
Other Project Types
Database
Test Projects
Online Templates
.NET Framework 4
Sort by: Default
ASP.NET Web Application
ASP.NET MVC 2 Web Application
ASP.NET Empty Web Application
ASP.NET MVC 2 Empty Web Application
ASP.NET Dynamic Data Entities Web Application
ASP.NET Dynamic Data Linq to SQL Web Application
ASP.NET AJAX Server Control
ASP.NET AJAX Server Control Extender
ASP.NET Server Control
Visual C#
Type: Visual C#
A project for creating an application with a Web user interface
Name: ProductWebClient
Location: E:\WCF\ProductServiceLibrary
Browse...
OK
Cancel


Monster Product Inc.
Click Me
Column0 Column1 Column2
abc abc abc
abc abc abc
abc abc abc
abc abc abc
abc abc abc


Monster Product Inc.
Click Me
Column0 Column1 Column2
abc abc abc
abc abc abc
abc abc abc
abc abc abc
abc abc abc
Build
Rebuild
Clean
Build Deployment Package
Publish...
Package/Publish Settings
View in Browser Ctrl+Shift+W
Convert to Web Application
Check Accessibility...
Project Dependencies...
Project Build Order...
Add
Add Reference...
Add Web Reference...
Add Service Reference...
View Class Diagram
Set as StartUp Project
Debug
Add Solution to Source Control...
Cut Ctrl+X
Paste Ctrl+V
Remove Del


ProductServiceLibrary - Microsoft Visual Studio (Administrator)
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://localhost/ProductServiceWebHost/Service.svc
Go
Discover
Services:
Operations:
Design_Time_Addresses/ProductServiceLibr
ProductServiceWebHost/Service.svc
Select a service contract to view its operations.
2 service(s) found in the solution.
Namespace:
ServiceReference1
Advanced...
OK
Cancel
Monster Product Inc.
Click Me
Column0 Column1 Column2
abc abc abc
Solution Explorer
Team Explorer
Class View
Error List
Output
Ready

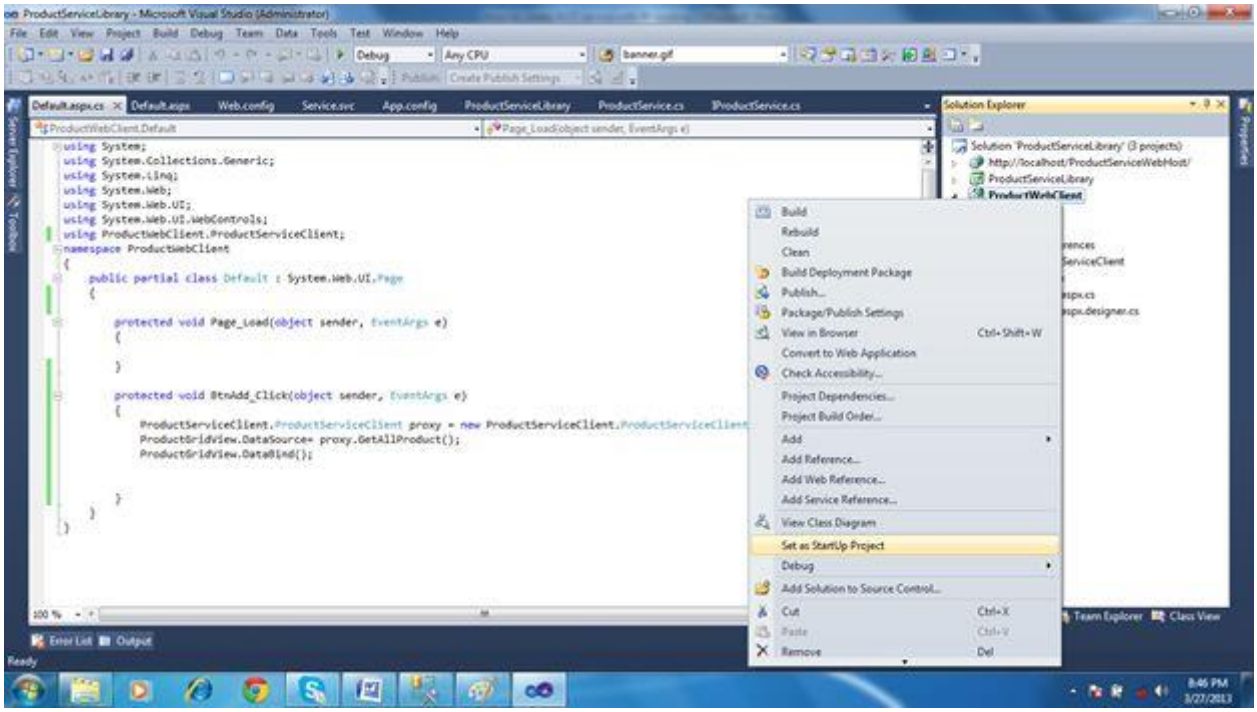

ProductServiceLibrary - Microsoft Visual Studio (Administrator)
Add Service Reference
To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.
Address:
http://localhost/ProductServiceWebHost/Service.svc
Go
Discover
Services:
Operations:
Design_Time_Addresses/ProductServiceLibr
ProductServiceWebHost/Service.svc
Select a service contract to view its operations.
2 service(s) found in the solution.
Namespace:
ProductServiceClient
Advanced...
OK
Cancel
Monster Product Inc.
Click Me
Column0 Column1 Column2
abc abc abc
Solution Explorer
Team Explorer
Class View
Error List
Output
Ready

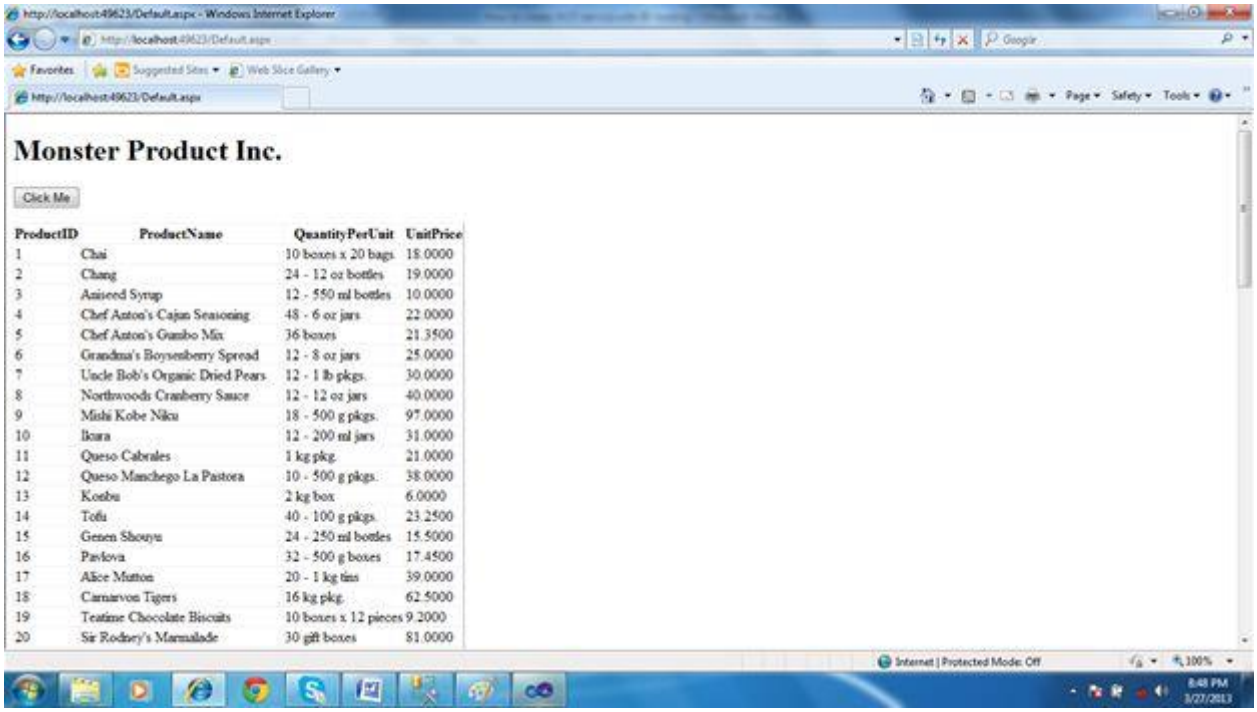


On the Default.aspx.cs page write the code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using ProductWebClient.ProductServiceClient;
namespace ProductWebClient
{
   public partial class Default : System.Web.UI.Page
   {

      protected void Page_Load(object sender, EventArgs e)
      {

      }

      protected void BtnAdd_Click(object sender, EventArgs e)
      {
         ProductServiceClient.ProductServiceClient proxy
= new ProductServiceClient.ProductServiceClient();
         ProductGridView.DataSource= proxy.GetAllProduct();
         ProductGridView.DataBind();


      }
   }
}
```

Now run the application, except before doing that do one more last task.





Set the ProductWebClient as "Set as StartUp Project" and press F5 and here is the result.