# Practical 4: Implement 3-Tier Web Application (Part 1)

## Objectives

- To learn how to implement a 3-tier web application including database manipulation.
- To learn how to write classes that provides database manipulation.
- To learn how to consume web services that provides database manipulation.

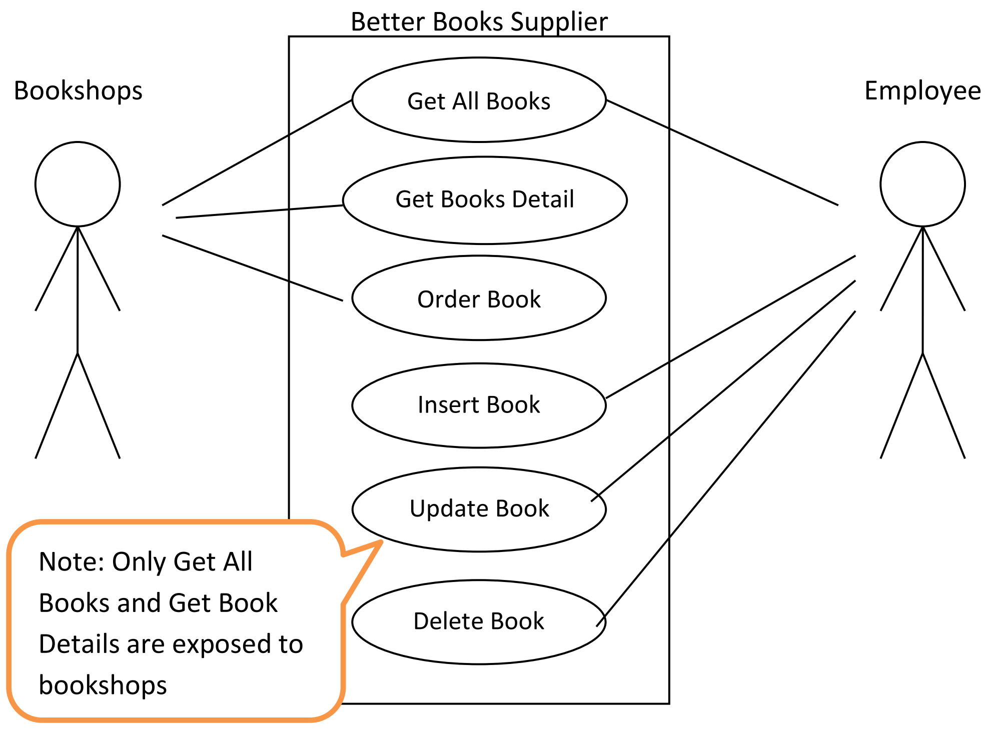
## Scenario

Better Books Supplier has a database of books that it supplies to bookshops. It provides following use cases :

- Insert, Update and Delete Books from the database for its employees.
- Get all books and Get book details to its customers(bookshop)

As each bookshop has its' own system, Better Books Supplier decided to expose get books' use cases as Web Services to the bookshops. In this practical we will create the Get All Books web service. The Get Books Detail and Order Book will be carried out in next practical.

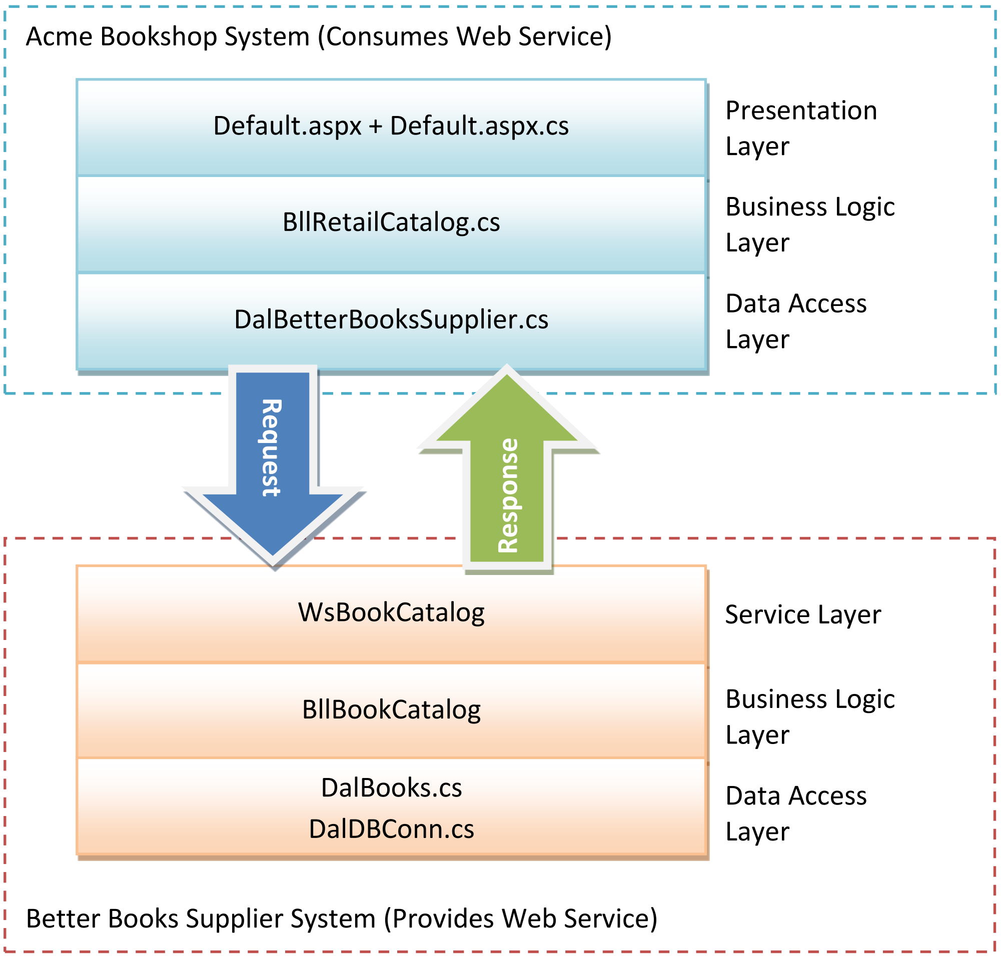
**Figure 1**



## System Architecture

Figure 2 is show the implementation of the two systems in the 3-tiers architecture. One system is the Better Books Supplier System and the other is the Acme Bookshop System.

Figure 2



Now it's time for us to implement the systems.

# Exercise 1 – Create Web Service in Better Book Supplier's System

## Creating Better Book Supplier Project

1. Start Visual Studio to create a New Project.
    - Choose *Visual C#   ASP.NET Web Application*.
    - Name the project as **IT2605Prac05_BBS**
    - Click the *OK* button.
    - Select Empty template
2. Download P5_BBS_Resource file

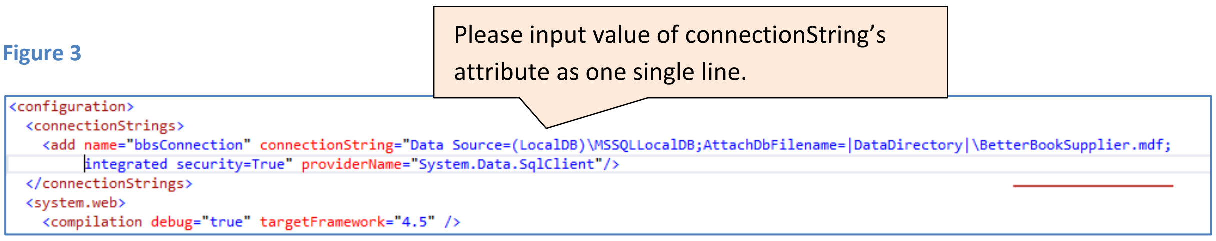
## Importing Database

3. Right click project IT2605Prac05_BBS, select Add **->**Add ASP.NET Folder **->**Add App_Data.
4. Add BetterBookSupplier.mdf file from resource file to the App_Data. This creates a SQL LocalDB.
5. Double click the mdf file in solution explorer to open the database in server explorer. If you encounter database compatible issue, modify the connection to reconnect.
6. The database contains 4 tables **books, authors, publisher** and **images**
7. Examining the database and discuss the relationship of the tables with your tutor.

## Creating a Data Access Layer

8. Open web.config file, add a child element named **<connectionString>** in **<configuration>** tag. Add connection string for BetterBookSupplier.mdf. The connection string can be obtained from P5_BBS_Resource

Figure 3



Please input value of connectionString's attribute as one single line.

```
<configuration>
  <connectionStrings>
    <add name="bbsConnection" connectionString="Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\BetterBookSupplier.mdf;
         integrated security=True" providerName="System.Data.SqlClient"/>
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
```

|DataDirectory| (enclosed in pipe symbols) is a substitution string that indicates the path to the database. It eliminates the need to hard-code the full path. By default, the |DataDirectory| variable for Web apps will be the App_Data folder.

The connection string in web.config defined the physical location of database server AND the credential to access the database.

This facilitates the deployment of application where the data base server is usually located at different location. You do not require to re-compile the code.

## Create data access layer to connect to database and manipulate the data in database

9. Right click **project** in Solution Explorer, click **create new folder** named as **DAL**. This folder holds data access layer.
10. For convenience, some code files have been prepared for this project resource folder, add DalDbConn, DalBooks.cs. Study the two class files.

    **DalDbConn** class contains the code necessary for connecting to the database.
    By creating this class, the rest of the project need not be concern with how to connect to the database. They do not need to worry about connection strings, what type of database is in the back end, etc.

    **DalBooks** class contains all the .NET code to create and execute the necessary SQL to manipulate the database of Better Book Supplier.

    **GetAll** function returns dataset that contain the book id, title, author name, and publisher name etc for all the books supplied by the supplier.

## Creating a Business Logic Layer

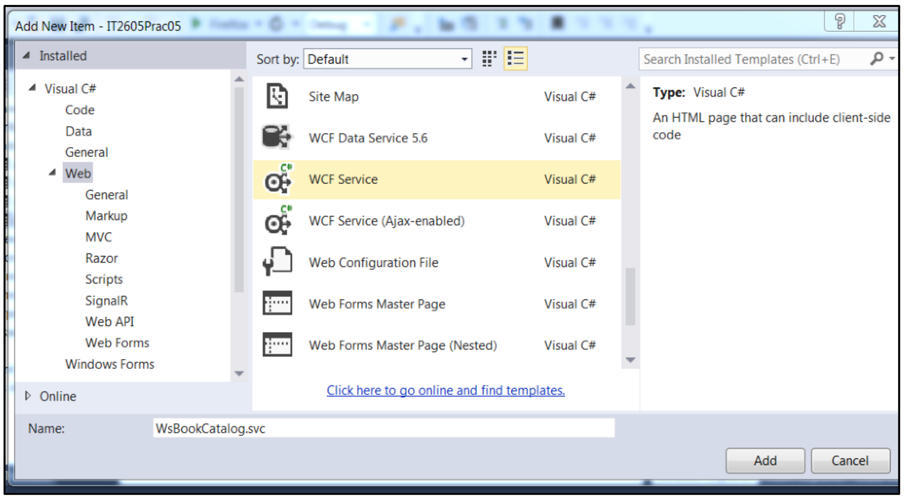
11. Create another new folder named as **BLL**. This folder holds *Business Logic Layer* class to access and process the data retrieved from database.
12. Select BLL folder, browse to your resource folder to add BLLBookCatalog.cs.
13. This class contains all the .NET code for business logic processes. In this practical, we just call the GetAll method in DalBooks, there is not much business logic yet, but we create the classes so that we can add the logic without much trouble at a later time.

## Create Web Service

Create the web service named as *WsBookCatalog.svc* to expose the book list to external customers, the bookshops.

14. *Right-click project in the Solution Explorer* and click *Add New Item*.
15. In the *Add New Item* dialog (See Figure 4):
    a. Select Visual C# under Installed Templates.
    b. Select ***WCF Service***. (Note: **NOT Web Service**.)
    c. Change the filename to ***WsBookCatalog.svc***.
    d. Click *Add* button.

Figure 4


Add New Item - IT2605Prac05
Installed
Visual C#
Code
Data
General
Web
General
Markup
MVC
Razor
Scripts
SignalR
Web API
Web Forms
Windows Forms
Online
Sort by: Default
Search Installed Templates (Ctrl+E)
Site Map Visual C#
WCF Data Service 5.6 Visual C#
WCF Service Visual C#
WCF Service (Ajax-enabled) Visual C#
Web Configuration File Visual C#
Web Forms Master Page Visual C#
Web Forms Master Page (Nested) Visual C#
Click here to go online and find templates.
Add
Cancel

Type: Visual C#

An HTML page that can include client-side code

Name: WsBookCatalog.svc

16. Two files are created.
    - WsBookCatalog.svc  - contains backend code to implement service operations
    - IWsBookCatalog.cs – create operation contract

17. Open WsBookCatalog.svc.cs,
    - delete ***DoWork()*** method and
    - add a ***GetBooks()*** method that return data set contains all the books from BLL as shown in Figure 5

Figure 5

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using System.Data;
using IT2605Prac05_BBS.BLL;
```

// (Change accordingly if your project name is different.)

```
namespace IT2605Prac05_BBS
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to
change the class name "WsBookCatalog" in code, svc and config file
together.
    // NOTE: In order to launch WCF Test Client for testing this service,
please select WsBookCatalog.svc or WsBookCatalog.svc.cs at the Solution
Explorer and start debugging.
    public class WsBookCatalog : IWsBookCatalog
    {
        public DataSet GetBooks()
        {
            IT2605Prac05_BBS.BLL.BLLBookCatalog bizLayerBooks;
            bizLayerBooks = new IT2605Prac05_BBS.BLL.BLLBookCatalog();
            return bizLayerBooks.GetBooks();
        }
    }
}
```

18. In *IWsBookCatalog.cs*,
    - delete the ***DoWork*** operation (including "[OperationContract]")
    - add ***GetBooks*** operation (including "[OperationContract]") as in Figure 6.

Figure 6

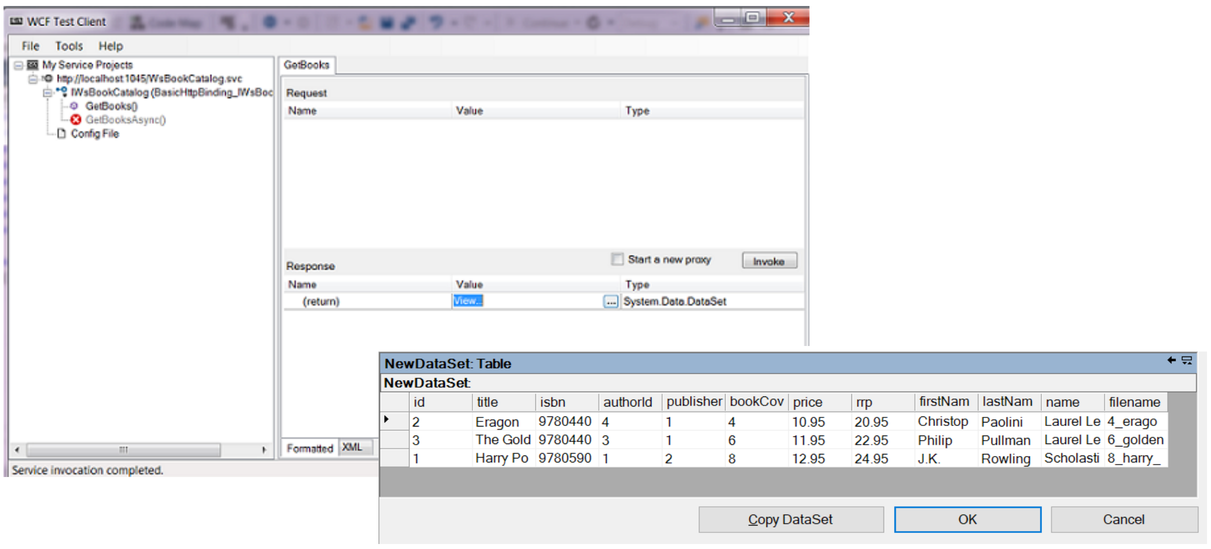
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using System.Data;

namespace IT2605Prac05_BBS
{
    // NOTE: You can use the "Rename" command on the
"Refactor" menu to change the interface name
"IWsBookCatalog" in both code and config file together.
    [ServiceContract]
    public interface IWsBookCatalog
    {
        [OperationContract]
        DataSet GetBooks();
    }
}
```

Add this using statement in order to use dataset in the code

We completed the code implementation. In this web service, we only expose the *GetBooks()* operations. The external customers can only access this operation. They are not able to access other methods such as Insert, Update and Delete books which are present in the Data Access Layer.

19. Select WsBookCatalog.svc in Solution Explorer, right click to set as start page.
20. Press F5 to test the service. Click on GetBooks at the left panel, click the invoke button.
21. At the formatted view of Response panel as in **Error! Reference source not found.**, bring the cursor to view, then click on ellipsis button to obtain the record-set returned by GetBooks method. You can also view the response in XML format by clicking on the XML tag.

Figure 7



| Name | Value | Type |
|---|---|---|
| (return) | View... | System.Data.DataSet |

Service invocation completed.

NewDataSet: Table
NewDataSet:

| id | title | isbn | authorId | publisher | bookCov | price | rrp | firstNam | lastNam | name | filename |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | Eragon | 9780440 | 4 | 1 | 4 | 10.95 | 20.95 | Christop | Paolini | Laurel Le | 4_erago |
| 3 | The Gold | 9780440 | 3 | 1 | 6 | 11.95 | 22.95 | Philip | Pullman | Laurel Le | 6_golden |
| 1 | Harry Po | 9780590 | 1 | 2 | 8 | 12.95 | 24.95 | J.K. | Rowling | Scholasti | 8_harry_ |

We have completed the web service creation for Better Book Supplier project.

## Exercise 2 – Create a web client for Acme Bookshop's System

In this section, we will create the Acme Bookshop System as a **separate** project.

### Create Acme Bookshop Project

1. Download and unzip the *P5_Acme_Resource.zip* file.
2. The resource file contains a solution **IT2605Prac05_ACME.sln**
3. **Launch another Visual Studio**, select *Open* ➔ *Project/Solution*
4. Browse to the resource folder to open **IT2605Prac05_ACME.sln**
5. The Project contains, a master page, css and Default web form.

### Adding a Service Reference to the Better Book Supplier Web Service

As we will be consuming the web service provided by *Better Books Supplier*, we will need to add a *Service Reference* to it.

6. Before adding service reference, ***ensure that WsBookCatalog.svc is hosted in internet***. If you have closed the project IT2605Prac05_BBS, reopen it and run the WsBookCatalog.svc.

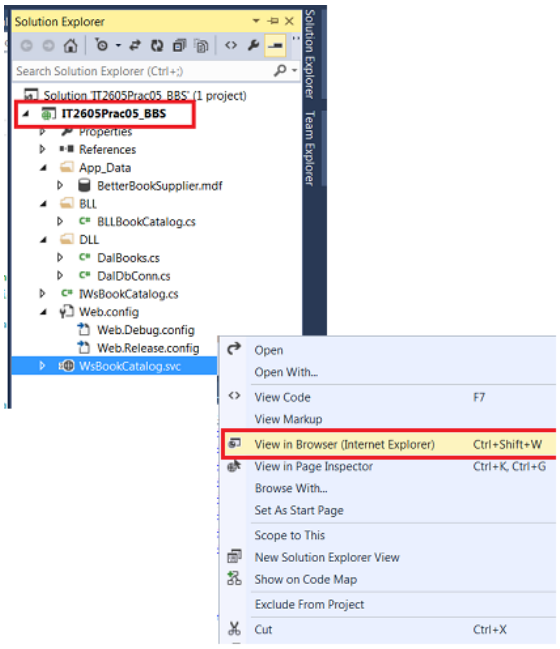
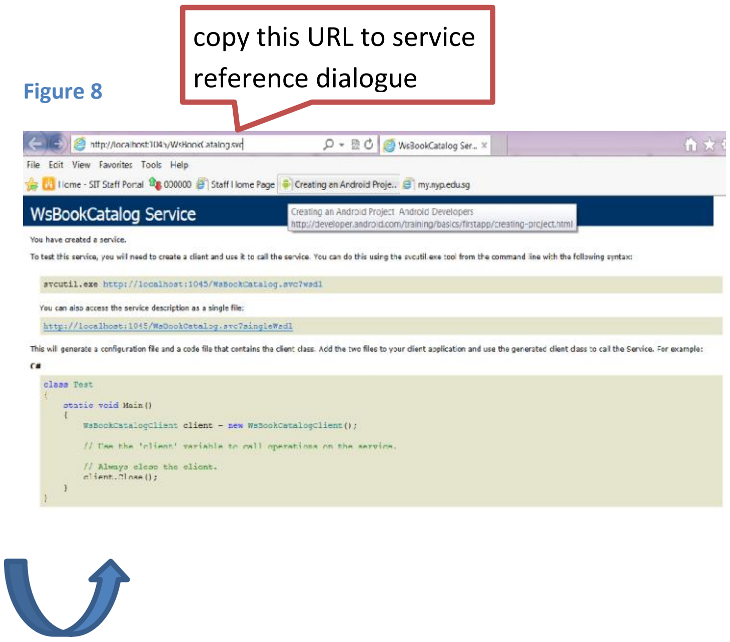


Figure 8


copy this URL to service reference dialogue
http://localhost:1045/WsBookCatalog.svc
WsBookCatalog Ser...
File Edit View Favorites Tools Help
Home - SIT Staff Portal
000000
Staff Home Page
Creating an Android Proje...
my.nyp.edu.sg
Creating an Android Project Android Developers
http://developer.android.com/training/basics/firstapp/creating-project.html

WsBookCatalog Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http://localhost:1045/WsBookCatalog.svc?wsdl
```

You can also access the service description as a single file:

```
http://localhost:1045/WsBookCatalog.svc?singleWsdl
```

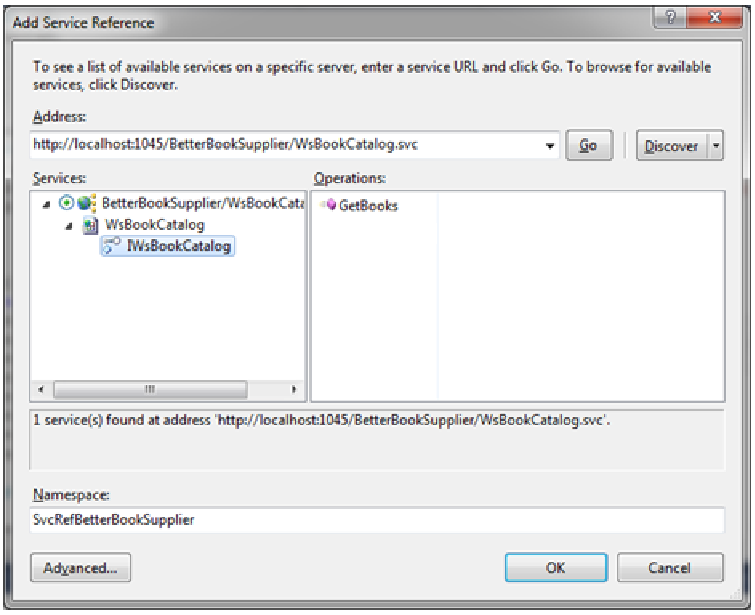
This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        WsBookCatalogClient client = new WsBookCatalogClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

7. In *Solution Explorer*, right-click the **IT2605Prac05_ACME** and select *Add Service Reference….*
8. In the *Add Service Reference* dialog, copy the URL in Figure 8, click GO button, WSBookCatalog is shown (See Figure 9Figure 9) .

Figure 9


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:1045/BetterBookSupplier/WsBookCatalog.svc

Services: BetterBookSupplier/WsBookCata / WsBookCatalog / IWsBookCatalog

Operations: GetBooks

1 service(s) found at address 'http://localhost:1045/BetterBookSupplier/WsBookCatalog.svc'.

Namespace: SvcRefBetterBookSupplier

9. Change the *Namespace* to ***SvcRefBetterBookSupplier*** and click the *OK* button.

## Adding DAL and BLL Folders to Acme Bookshop Project

10. In *Solution Explorer*, right-click IT2605Prac05_ACME to *add new folders*. Named the folder as DAL and BLL

## Creating a Data Access Layer to Access the Better Book Supplier Web Service

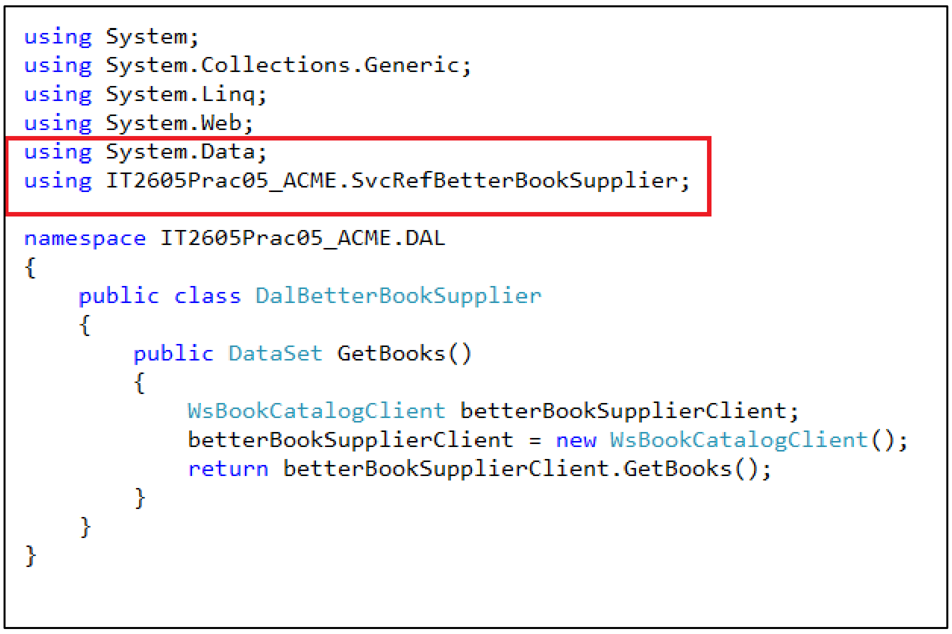
What the web service provides is data about *Better Book Supplier's* books, hence we will create a Data Access Layer class that has methods to call the Web Methods and get the data.

**Important:** In the real world, *Acme Bookshop* might have many book suppliers. Each supplier will probably provide the same data (i.e.: book title, author, publisher, ISBN code, prices, etc) but each bookshop may format, organise or name the data differently. In such a situation, the Data Access Layer objects can help to transform the received data into a common format for use within the *Acme Bookshop* system. In the simplified scenario for this project, we will ignore that step.

11. Add a new *Visual C#* class to **DAL** folder and name the class *DalBetterBookSupplier.cs*.
In *DalBetterBookSupplier*.cs*,* add the code shown in

12. to invoke GetBooks method from Better Book Supplier's web service

Figure 10



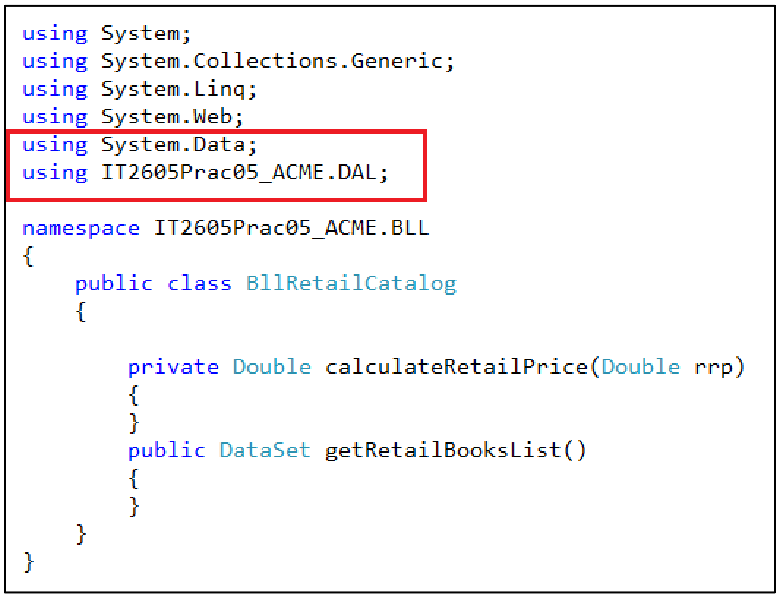
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data;
using IT2605Prac05_ACME.SvcRefBetterBookSupplier;

namespace IT2605Prac05_ACME.DAL
{
    public class DalBetterBookSupplier
    {
        public DataSet GetBooks()
        {
            WsBookCatalogClient betterBookSupplierClient;
            betterBookSupplierClient = new WsBookCatalogClient();
            return betterBookSupplierClient.GetBooks();
        }
    }
}
```

## Creating a Business Logic Layer

The Data Access Layer merely provides us a convenient way to access the web service. ACME may have other business logic to apply to the data. So now we create a BLL class.

13. Add a new *Visual C#* class to **BLL** folder and name the class ***BllRetailCatalog.cs*.**
14. In *BllRetailCatalog*.cs*,* add the code shown in Figure 11. You will get the red error lines below the method names because we have not added code to return a value.

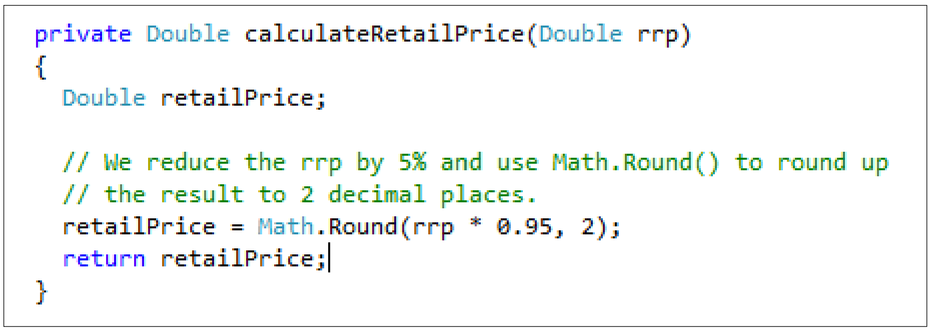
Figure 11



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data;
using IT2605Prac05_ACME.DAL;

namespace IT2605Prac05_ACME.BLL
{
    public class BllRetailCatalog
    {

        private Double calculateRetailPrice(Double rrp)
        {
        }
        public DataSet getRetailBooksList()
        {
        }
    }
}
```

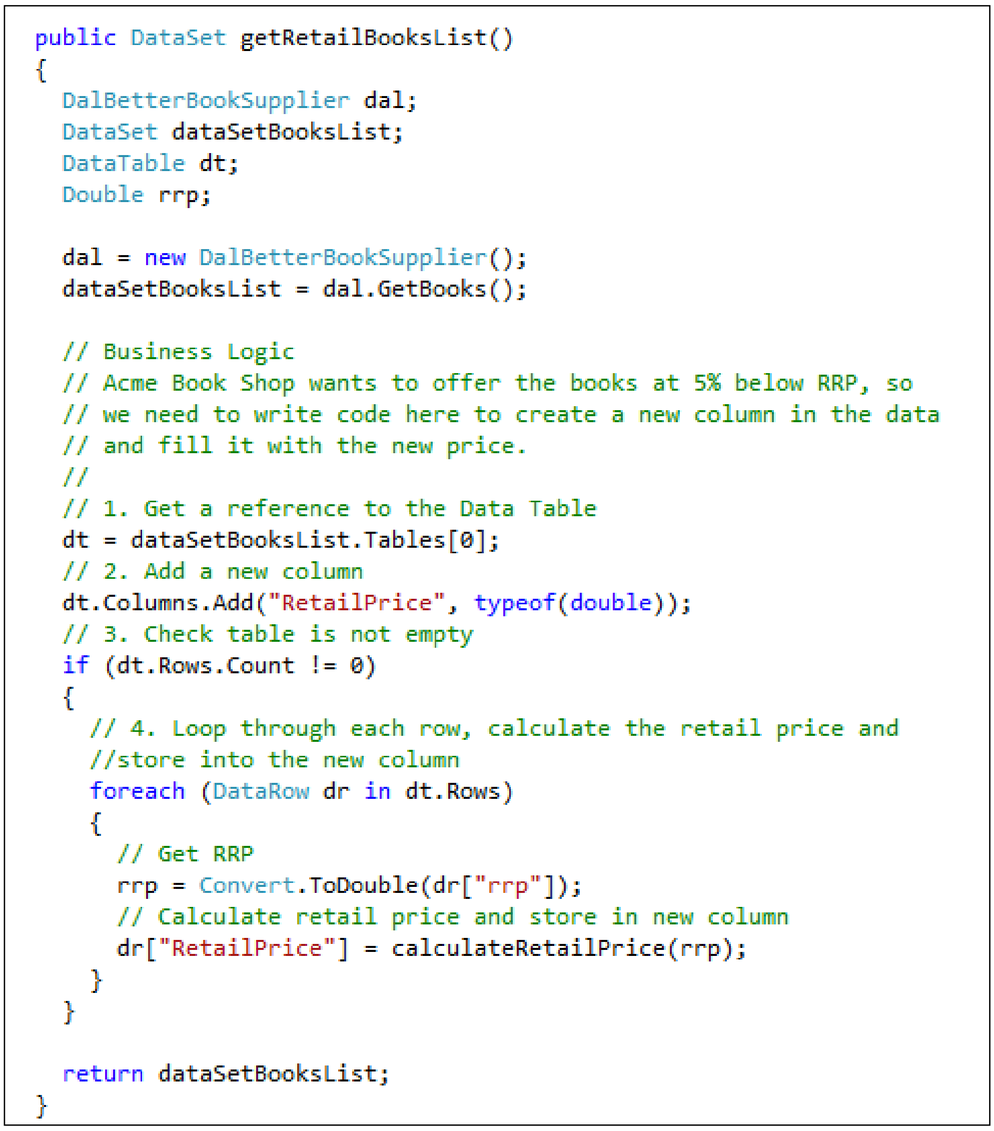
15. ACME bookshop derives their retail price based on the supplier's offer. Add the code to the respective methods as shown in Figure 12 and Figure 13. Make sure to copy the comments as well, so that you understand what the code does.

Figure 12



```
private Double calculateRetailPrice(Double rrp)
{
  Double retailPrice;

  // We reduce the rrp by 5% and use Math.Round() to round up
  // the result to 2 decimal places.
  retailPrice = Math.Round(rrp * 0.95, 2);
  return retailPrice;
}
```

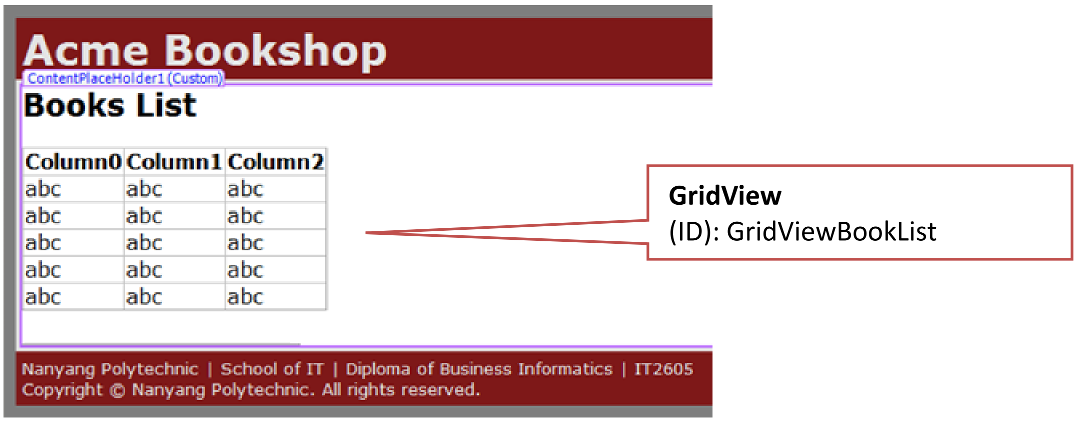
Figure 13



```
public DataSet getRetailBooksList()
{
  DalBetterBookSupplier dal;
  DataSet dataSetBooksList;
  DataTable dt;
  Double rrp;

  dal = new DalBetterBookSupplier();
  dataSetBooksList = dal.GetBooks();

  // Business Logic
  // Acme Book Shop wants to offer the books at 5% below RRP, so
  // we need to write code here to create a new column in the data
  // and fill it with the new price.
  //
  // 1. Get a reference to the Data Table
  dt = dataSetBooksList.Tables[0];
  // 2. Add a new column
  dt.Columns.Add("RetailPrice", typeof(double));
  // 3. Check table is not empty
  if (dt.Rows.Count != 0)
  {
    // 4. Loop through each row, calculate the retail price and
    //store into the new column
    foreach (DataRow dr in dt.Rows)
    {
      // Get RRP
      rrp = Convert.ToDouble(dr["rrp"]);
      // Calculate retail price and store in new column
      dr["RetailPrice"] = calculateRetailPrice(rrp);
    }
  }

  return dataSetBooksList;
}
```

## Create a Web Page to Display the Book Catalogue

16. Open *Default.aspx in Design view as* in Figure 14 Figure 14


Acme Bookshop
ContentPlaceHolder1 (Custom)
Books List

| Column0 | Column1 | Column2 |
|---|---|---|
| abc | abc | abc |
| abc | abc | abc |
| abc | abc | abc |
| abc | abc | abc |
| abc | abc | abc |

**GridView**
(ID): GridViewBookList

Nanyang Polytechnic | School of IT | Diploma of Business Informatics | IT2605
Copyright © Nanyang Polytechnic. All rights reserved.

## Displaying the Books in the Grid View

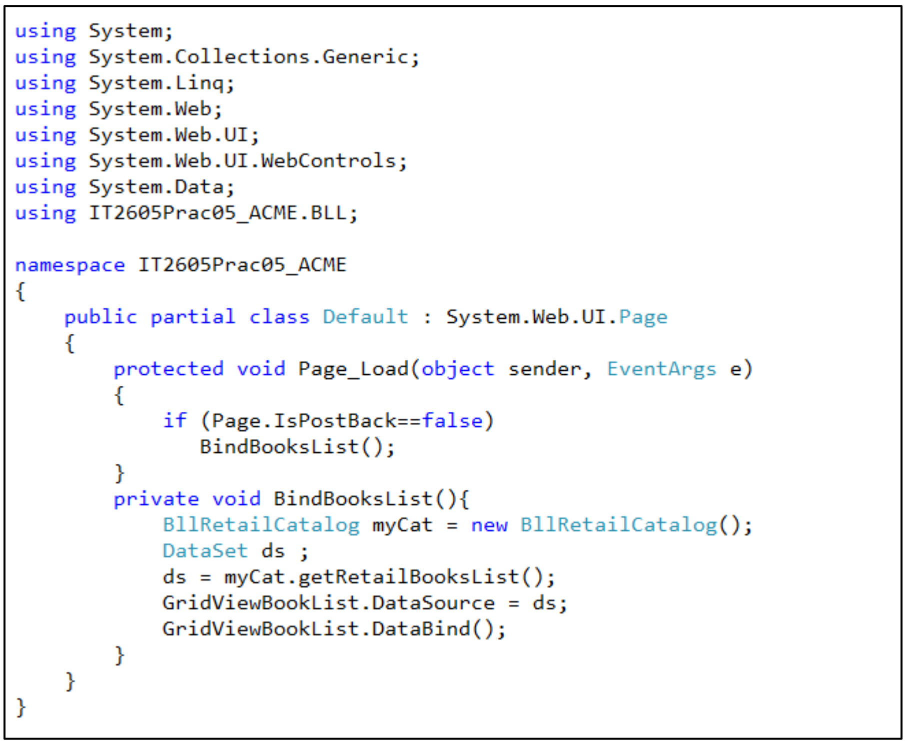
17. Toggle to code behind.

    Add the code to the respective methods as shown in Figure 15

    Note that when the page first loads, a check is made to see if **IsPostback** is false. If so, it means that the web form is loaded for very first time, and we bind the controls using BindBooksList method.

    BindBooksList method instantiates BllRetailCatalog class to call getRetailBooksList method. We will use the return dataset from getRetailBooksList method to bind to data source of Grid view.

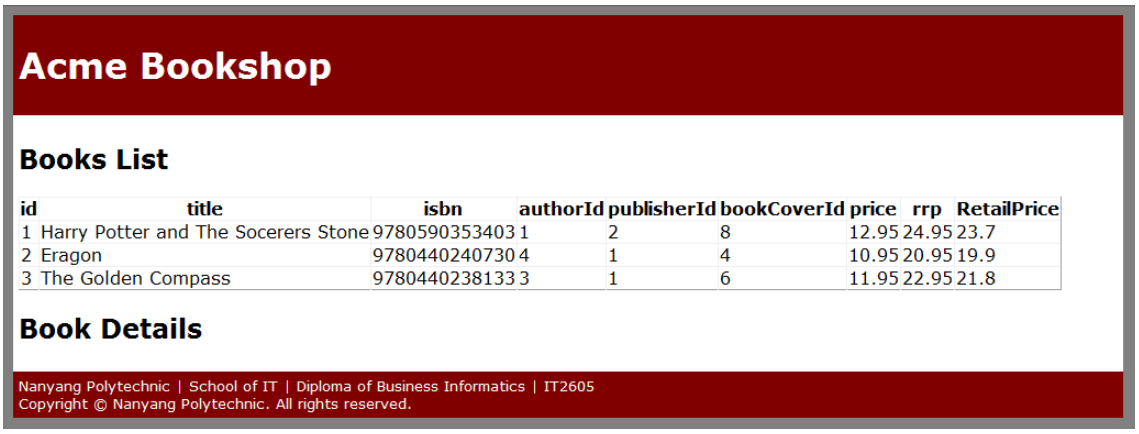
Figure 15



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using IT2605Prac05_ACME.BLL;

namespace IT2605Prac05_ACME
{
    public partial class Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Page.IsPostBack==false)
               BindBooksList();
        }
        private void BindBooksList(){
            BllRetailCatalog myCat = new BllRetailCatalog();
            DataSet ds ;
            ds = myCat.getRetailBooksList();
            GridViewBookList.DataSource = ds;
            GridViewBookList.DataBind();
        }
    }
}
```

18. Save your files and test the page. You should be able to get something like Figure 16.

Figure 16



Acme Bookshop

Books List

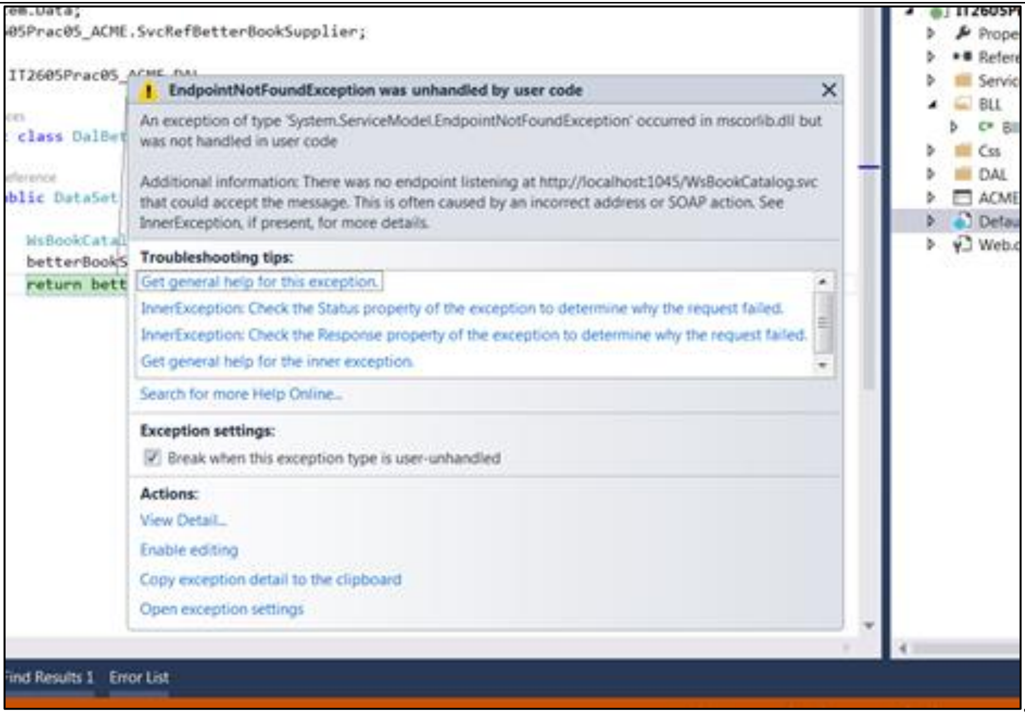
| id | title | isbn | authorId | publisherId | bookCoverId | price | rrp | RetailPrice |
|---|---|---|---|---|---|---|---|---|
| 1 | Harry Potter and The Socerers Stone | 9780590353403 | 1 | 2 | 8 | 12.95 | 24.95 | 23.7 |
| 2 | Eragon | 9780440240730 | 4 | 1 | 4 | 10.95 | 20.95 | 19.9 |
| 3 | The Golden Compass | 9780440238133 | 3 | 1 | 6 | 11.95 | 22.95 | 21.8 |

Book Details

Nanyang Polytechnic | School of IT | Diploma of Business Informatics | IT2605
Copyright © Nanyang Polytechnic. All rights reserved.

19. Close the browser and return to *Visual Studio*.
20. In case you encountered error "**EndpointNotFoundException**" during run time, the cause could be that
    a. BBS web service is not run
    b. Changes in BBS web service are not refreshed or out of sync.

    To resolve this error, closed WsBookCatalog.svc from your browser, rebuild the solution of WsBookCatalog and update the service reference in Acme Book Shop. Refer to explanation in Exercise 1 step **Error! Reference source not found.**.



**EndpointNotFoundException was unhandled by user code**

An exception of type 'System.ServiceModel.EndpointNotFoundException' occurred in mscorlib.dll but was not handled in user code

Additional information: There was no endpoint listening at http://localhost:1045/WsBookCatalog.svc that could accept the message. This is often caused by an incorrect address or SOAP action. See InnerException, if present, for more details.

**Troubleshooting tips:**

Get general help for this exception.

InnerException: Check the Status property of the exception to determine why the request failed.

InnerException: Check the Response property of the exception to determine why the request failed.

Get general help for the inner exception.

Search for more Help Online...

**Exception settings:**

☑ Break when this exception type is user-unhandled

**Actions:**

View Detail...

Enable editing

Copy exception detail to the clipboard

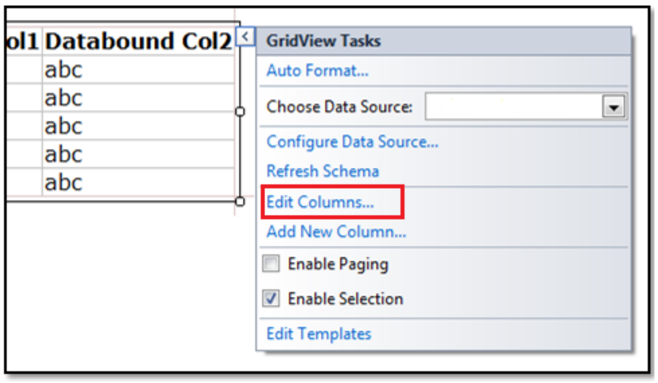
Open exception settings

## Customising GridView

So far, all the columns retrieved by BllRetailCatalog are displayed by default in Grid View. We can customise data to display only columns we need.

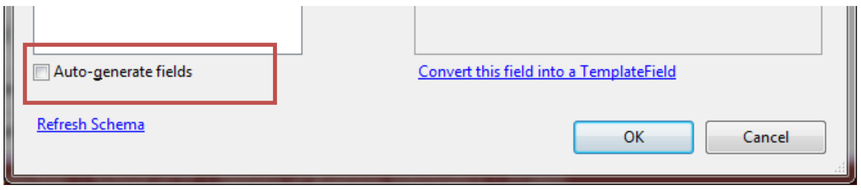
21. Expand the *Smart Tag* of the *Grid View* and select *Edit Columns…* (Figure 7)

Figure 17



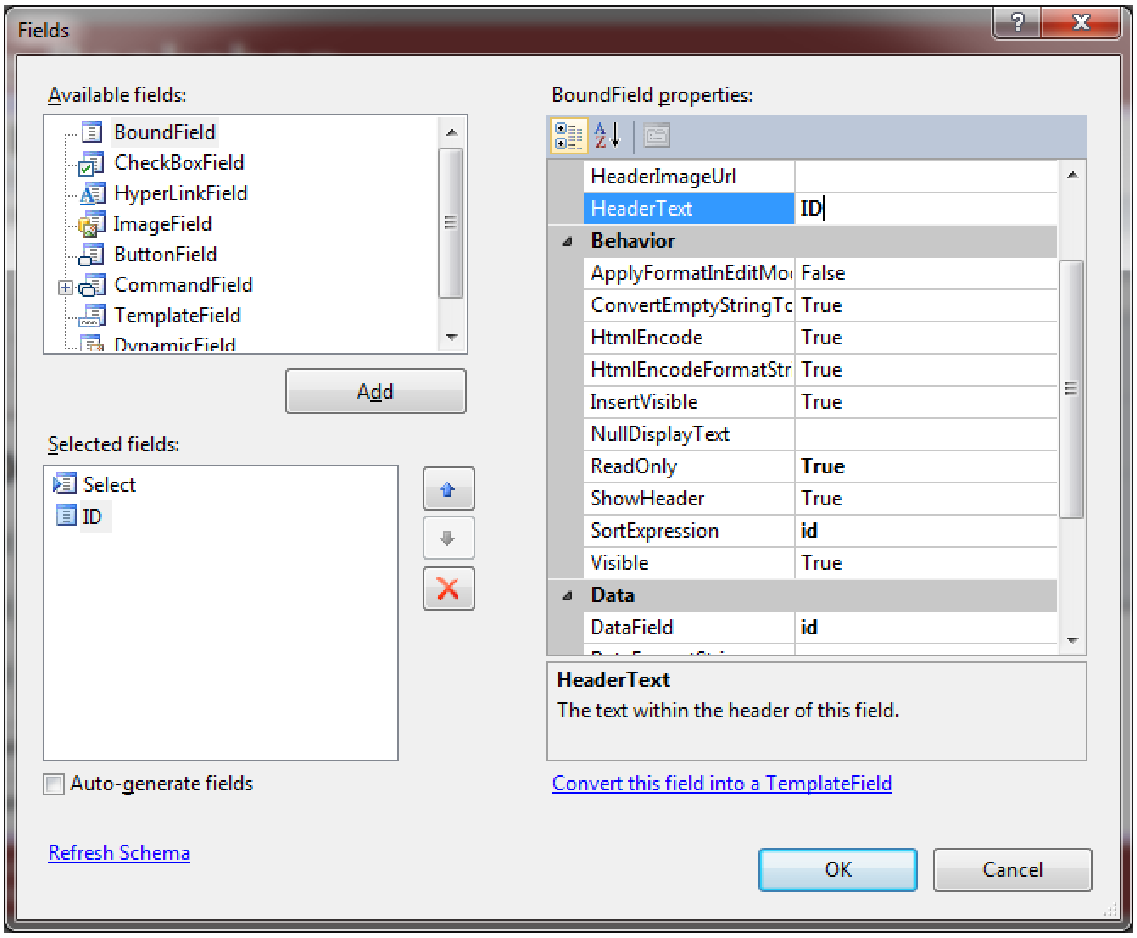
22. In the *Fields* dialog, un-check *Auto-generate fields* (see Figure 18).

Figure 18



23. Select the *BoundField* in the *Selected fields* list and click on Add button then set the following properties (see Figure 19):
    a. *HeaderText*: ID
    b. *ReadOnly*: True
    c. *SortExpression*: id
    d. *DataField*: id

**Figure 19**



| | |
|---|---|
| HeaderImageUrl | |
| HeaderText | ID |
| **Behavior** | |
| ApplyFormatInEditMo | False |
| ConvertEmptyStringTc | True |
| HtmlEncode | True |
| HtmlEncodeFormatStr | True |
| InsertVisible | True |
| NullDisplayText | |
| ReadOnly | **True** |
| ShowHeader | True |
| SortExpression | **id** |
| Visible | True |
| **Data** | |
| DataField | **id** |

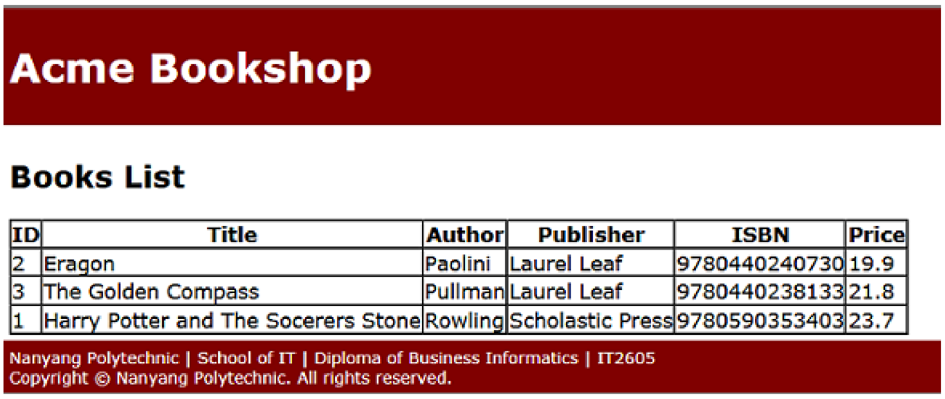
**HeaderText**
The text within the header of this field.

24. Repeat the steps above to add more bound fields shown in Table 1 with the respective settings. note that the DataField is corresponded to the column name of the DataTable of the web service in Figure 17 except RetailPrice which is derived from ACME business logic layer

Table 1

| **HeaderText** | **DataField & SortExpression (case sensitive)** | **Data Formating string** | **Remark** |
|---|---|---|---|
| Title | title | | |
| Author | lastName | | |
| Publisher | name | | |
| ISBN | isbn | | |
| Price | RetailPrice | {0:C} | To display currency |

25. After the last entry of field item, click the *OK* button to save your settings
26. Build the solution and test it again. You should get something like


Acme Bookshop

Books List

| ID | Title | Author | Publisher | ISBN | Price |
|---|---|---|---|---|---|
| 2 | Eragon | Paolini | Laurel Leaf | 9780440240730 | 19.9 |
| 3 | The Golden Compass | Pullman | Laurel Leaf | 9780440238133 | 21.8 |
| 1 | Harry Potter and The Socerers Stone | Rowling | Scholastic Press | 9780590353403 | 23.7 |

Nanyang Polytechnic | School of IT | Diploma of Business Informatics | IT2605
Copyright © Nanyang Polytechnic. All rights reserved.

**~ End ~**

## Checkpoint

By the end of this practical, you should be able to:

- ✓ Able to create a WCF web service that return message containing data from service requestor database
- ✓ Able to consume the web service and gain access of data generated from service requestor database.