

## Practical 07: Web Service Security

### Objectives

- Add message authentication and authorisation control in Better Book Supplier Web Service
- Manage message exception in ACME client application

### Scenario

The Better Book Supplier Web Service does not have any security. This means that anybody, including their competitors, can access the Web Service and see the company's pricing and other information. These are information that the company would probably want to keep secure from unauthorised viewing.

Today we will add security to the web service. We will code our web service such that it requires user credentials (information to authenticate the user such as user name and password) to be included in the Web Service call. The credentials will be in the header of the SOAP request. (Recall from our lectures that web services use the SOAP protocol to communicate.)

### Better Book Supplier's Web Service Security

#### Preparation

1. Download and unzip **P07\_Resource.zip**.
2. Open the IT2605Prac05\_BBS solution.
3. Open the BBS database.
4. A new table Retailer is found in the database. This table contains the record of retailer credential information with user id, password and user role.

#### Exercise 1: Adding Authentication and Authorisation control to the Web Service

SOAP base web service use **Message Contracts** to specifically control the layout of the SOAP messages to provide extra layer of security.

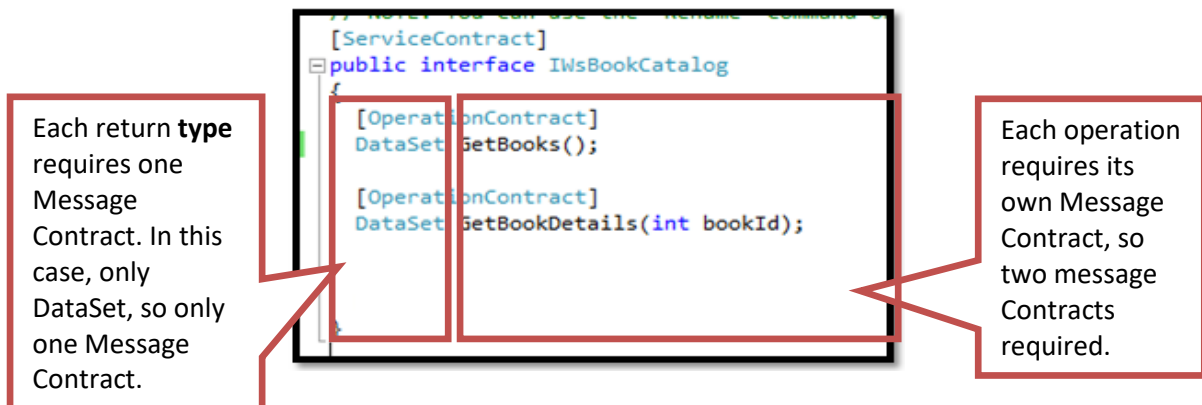
Rules for working with message contract.

- Only **one parameter can be used in Operation**.
- Service operation either **should return MessageContract** type **or it should not return any value**.

This exercise will modify the Web Service to include Authentication and Authorisation. We are going to transmit the user name and password via the *SOAP Message Headers*. In this practical, you will customise the *SOAP Messages* to include authentication information in the message header, and any parameters in the message body. To do that, we need to use *Message Contracts* to define the format of the *SOAP Messages* that our web service uses.

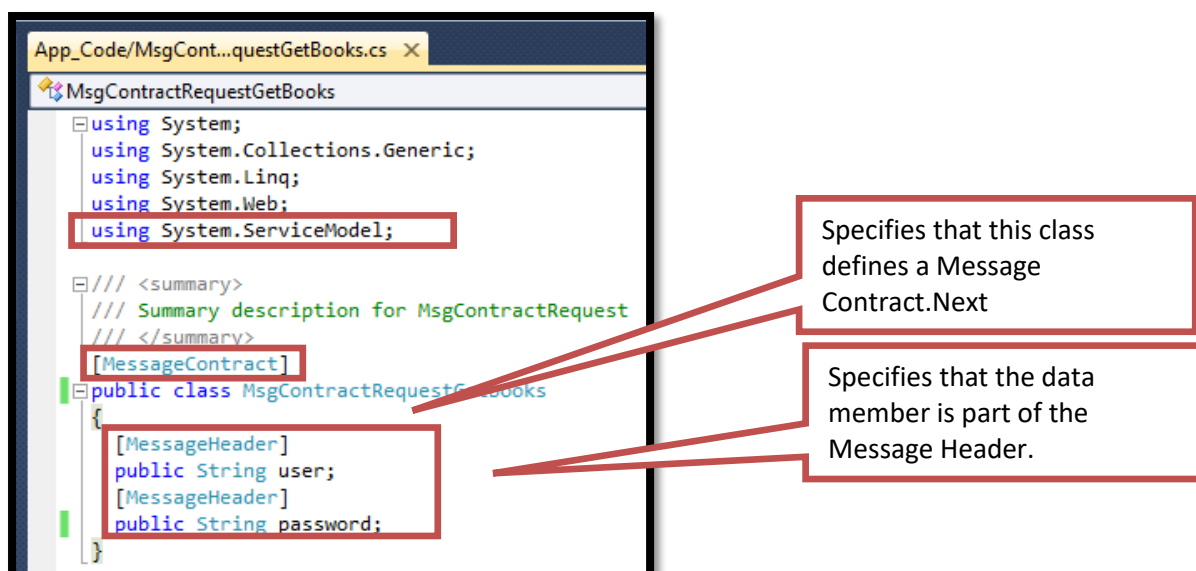
5. Due to time constraint, we will only secure *GetBooks()* and *GetBookDetails()*.
6. Examine *IWsBookCatalog.cs*, you will realise that three message contracts are required (see Figure 1).
  - a. Message Contract for input request of *GetBooks()*
  - b. Message Contract for input request of *GetBookDetails*
  - c. Message Contract for return data for both operations

Figure 1



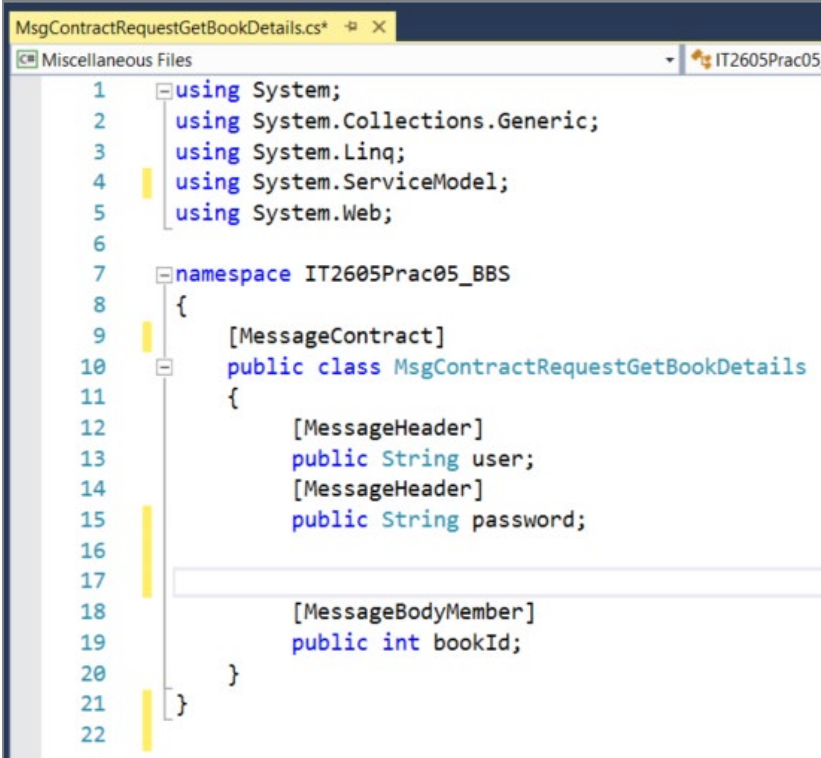
7. From Solution Explorer, add a new class named ***MsgContractRequestGetBooks.cs***. This class will define a *Message Contract* for the client to invoke the *GetBooks()* operation. Notice that we name the classes after the web operations.
8. We want the client to include its' user id and password in the header of the SOAP Message when it invokes the web service operations, we use the *[MessageHeader]* modifier to specify user id and password as *SOAP Message Header*.

Figure 2



9. Next, add a new class named ***MsgContractRequestGetBookDetails.cs***. This class defines a *Message Contract* to invoke *GetBookDetails()* operation.
10. Complete the code in Figure 3.  
**[Hint]** This time in addition to the user and password, we also need to transmit the book ID of the book, hence **the book ID should be in the Message Body**, we use a *[MessageBodyMember]* modifier.

Figure 3



```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.ServiceModel;
5  using System.Web;
6
7  namespace IT2605Prac05_BBS
8  {
9      [MessageContract]
10     public class MsgContractRequestGetBookDetails
11     {
12         [MessageHeader]
13         public String user;
14         [MessageHeader]
15         public String password;
16
17
18         [MessageBodyMember]
19         public int bookId;
20     }
21 }
22
```

11. Add another class named ***MsgContractResponseDataSet.cs*** for responses message. This class declare message contract contain **message body** that return **data set**. Note that it is not necessary to declare header for this response message. Complete the code in Figure 4.

Figure 4

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ServiceModel;
using System.Data;

namespace IT2605Prac05_BBS
{
    [MessageContract]
    public class MsgContractResponseDataSet
    {
        [MessageBodyMember]
        public DataSet results;
    }
}
```

12. Now we can modify our web service operations. Open *IWsBookCatalog.cs*.
13. We will modify the methods to accept the respective *Message Contracts* as inputs and return data as *MsgContractResponseDataSet*. Modify the code as shown in Figure 5.

Figure 5

```
[ServiceContract]
public interface IWsBookCatalog
{
    [OperationContract]
    MsgContractResponseDataSet GetBooks(MsgContractRequestGetBooks req);

    [OperationContract]
    MsgContractResponseDataSet GetBookDetails(MsgContractRequestGetBookDetails req);

    [OperationContract]
    int CreateOrders(String customerid, List<Order> bookOrder);
}
```

14. Open *WsBookCatalog.cs* and modify the code as shown in Figure 6.

Figure 6



**Adding data layer code to authenticate user**

15. Let us add the code to authenticate consumers against the provider's user database.
16. Right click **DLL folder**, add an existing item 'DalRetailer.cs' from unzip folder.
17. Open the file and look at **isValidUser** method. It selects record from **retailer** table in database that matched user id and password in input parameter.
18. Complete the code in Figure 7 such that if the record exist, the method will return true.

Figure 7

```
public Boolean isValidUser(String userid, String userpw)
{
    StringBuilder sql;
    SqlDataAdapter da;
    DataSet retailer_ds = new DataSet();

    SqlConnection conn = dbConn.GetConnection();

    sql = new StringBuilder();
    sql.AppendLine("SELECT userId, userPW ");
    sql.AppendLine("FROM retailer");
    sql.AppendLine("WHERE userId = @paraId");
    sql.AppendLine("AND userPW = @paraPw");
    try
    {
        da = new SqlDataAdapter(sql.ToString(), conn);
        da.SelectCommand.Parameters.AddWithValue("@paraId", userid);
        da.SelectCommand.Parameters.AddWithValue("@paraPw", userpw);

        da.Fill(retailer_ds);
    }
    catch (Exception ex)
    {
        errMsg = ex.Message;
    }
    finally
    {
        conn.Close();
    }
}
```

Complete the code to return true if record is found that match the select statement.

[Hint] Use the record count returned from the SqlDataAdapter.

**Adding data layer code to verify authorised user**

19. Look at the method *isAppUser*.
20. This method required input parameter for user id in String data type, and return true if the user role is 'appuser'.
21. Formulate SQL command to retrieve record from Retailer table where user role is equal to 'appuser'

**Figure 8**

```
public Boolean isAppUser(String userid)
{
    StringBuilder sql;
    SqlDataAdapter da;
    DataSet retailerds = new DataSet();
    SqlConnection conn = dbConn.GetConnection();

    try
    {
        da = new SqlDataAdapter(sql.ToString(), conn);
        da.SelectCommand.Parameters.AddWithValue("@paraid", userid);
        da.Fill(retailerds);
    }
    catch (Exception ex)
    {
        errMsg = ex.Message;
    }
    finally
    {
        conn.Close();
    }
    if (retailerds.Tables[0].Rows.Count == 0)
        return false;
    else
        return true;
}
```

Formulate SQL command to retrieve record from Retailer table where user role equal to 'appuser' for the selected user id.

### Adding business layer code for user authentication and authorisation

22. Right click **BLL folder**, add an existing item 'ValidateRetailer.cs' from unzip folder.
23. Return to *WsBookCatalog.cs* and add the code shown in Figure 9.

Figure 9



The code performs two levels of security verification.

- A. Authentication verification by calling **isValidUser()**, if the user id and password are invalid
- B. Authentication verification by calling **isAppUser()** with designated user role

If both authentication and authorisation checked is passed, the service provider will then grant access to GetBooks operation.

Both verification will throw fault exception with appropriate fault detail and fault code.

24. The program purposely throw exception, this will cause the debugger to stop execution at the point of exception, hence it is necessary to uncheck the 2 following options from the Tools->Option menu of IDE.

TOOLS→Option → Debugging →General →

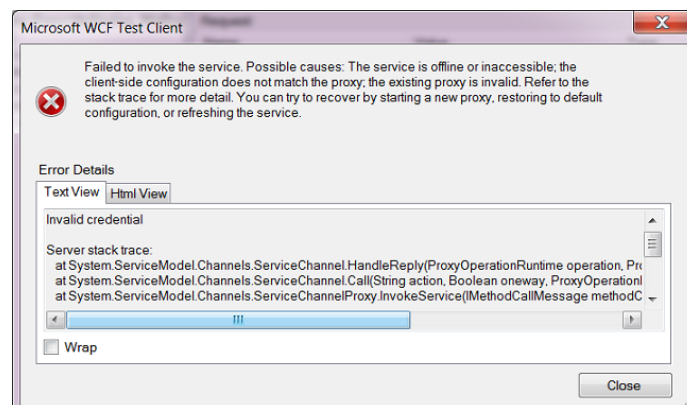
- Uncheck **Enable the exception assistance** and
- Uncheck **Enable just my code**



25. Build the solution and test the result using following test cases.

User id	User Password	Test result
POP	pa\$\$word	Passed.
XYZ	pa\$\$word	Authentication Fail
PANPAN	pa\$\$word	Authorisation Fail
ACMEadm	pa\$\$word	Passed.

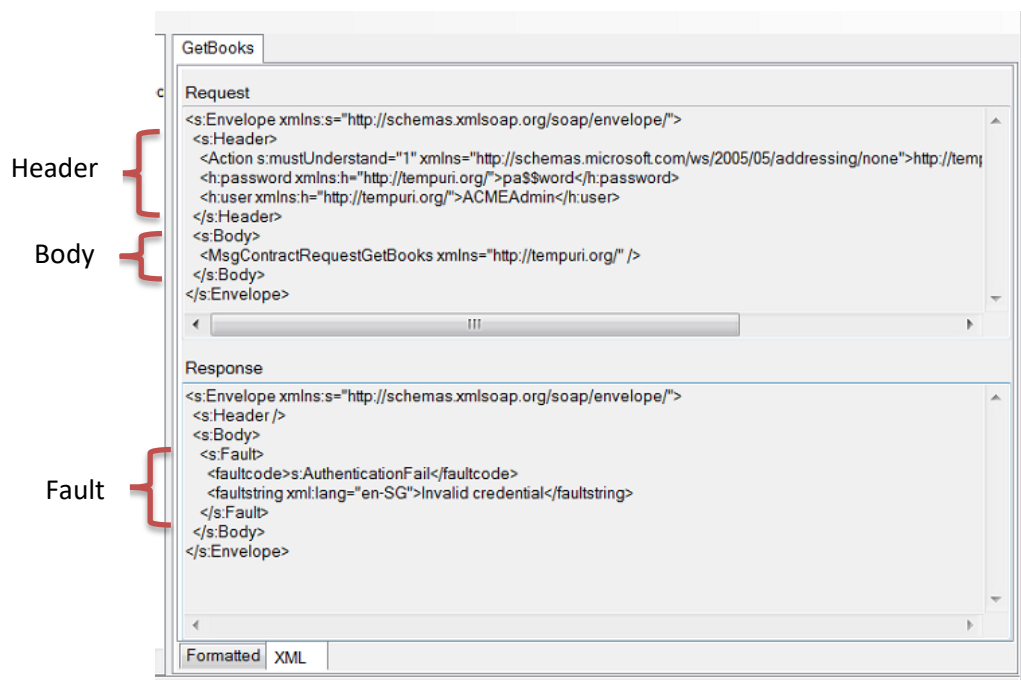
Please not to be alarmed when you receive the following error page in WCF Test Client when the authentication and authorisation fail. This is due to the fact that the system throws the exception in Figure 9.



Click the Close button, toggle to XML tag in test form.

In SOAP request, you will find Header message contain child elements user and password

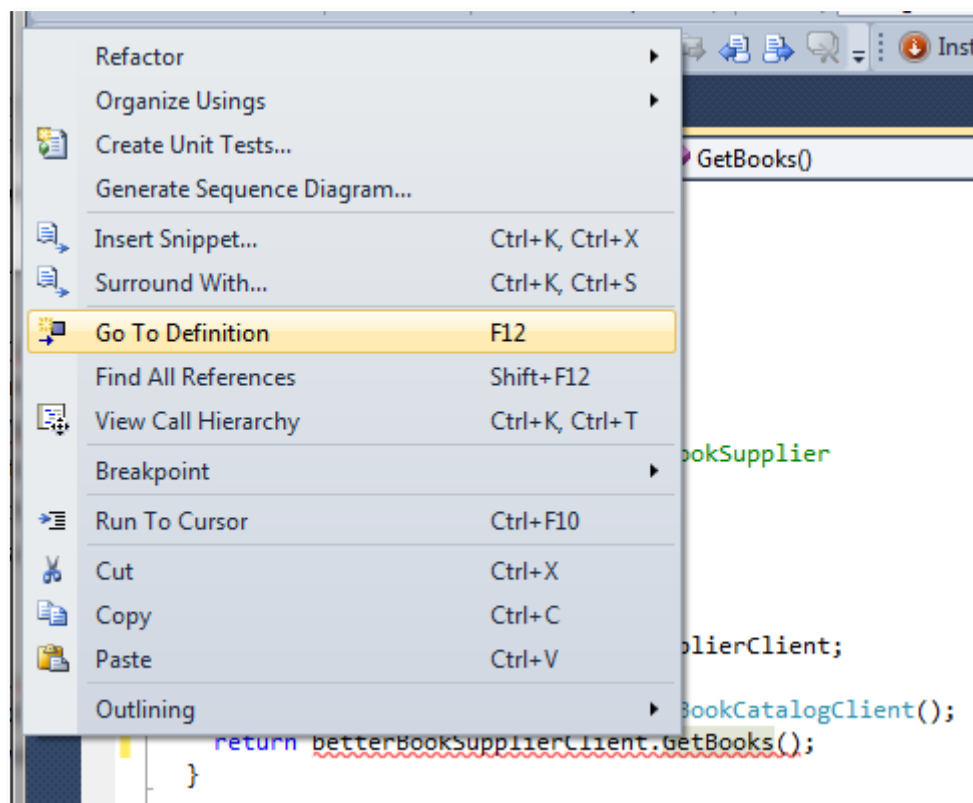
In SOAP response, you will find the Fault element that provide error message returned by the code



## Exercise 2: Manage SOAP exception with Web Service Client

26. Before we add code to authenticate the consumer, let us make sure *AcmeBookShop* can still work with the changes we just made.
27. Open *IT2605Prac05\_ACME solution*
28. Open *DalBetterBookSupplier.cs*
29. In the *Solution Explorer*, right-click on *SvcRefBetterBookSupplier* in **Service References**, and select *Update Service References*. Give Visual Studio a minute to complete the task. (Make sure your BSS is running)
30. Check *DalBetterBookSupplier.cs* again. Notice that there are now errors in the code? This is because the update has captured the changes we made and Visual Studio realises that the old code does not match up any more.
31. Hover your mouse over the line:  
`return betterBookSupplierClient.GetBooks();`
32. The error message is "No overload for method GetBooks() takes 0 arguments." This means that the *GetBooks()* operation from the web service requires 1 or more parameters.
33. Let us see what the new parameters are. Right-click on *GetBooks()* in the line:  
`return betterBookSupplierClient.GetBooks();`  
and select *Go To Definition* (see Figure 10).

Figure 10



34. Visual Studio will open an auto-generated file named *WsBookCatalogClient*. Look in the code for the *GetBooks()* method. Take note of the parameters required (see Figure 11). Interestingly, we are not required to use *MsgContractRequestGetBooks*, as defined for *GetBooks()* in

*IWsBookCatalog.cs* and *WsBookCatalog.cs*. Instead Visual Studio has mapped it to the relevant primitive types. Similarly, instead of having to handle an *MsgContractResponseDataSet*, it is just a *DataSet*. Take note of the arrangement of the parameters of each method; password first, user is second and then other parameters.

**Important:** Always use *Goto Definition* to find out how the new web service operations are defined before you code, so that you will use them correctly. Or pay attention to the hints that Visual Studio displays when you type your code.

Figure 11

```

IT2605Prac05_ACME.SvcRefBetterBookSupplier.WsBookCatalogClient
0 references
188 public WsBookCatalogClient(System.ServiceModel.Channels.Binding binding, System.ServiceModel.EndpointAddress remoteAddress) {
189     base(binding, remoteAddress) {
190 }
191
192 [System.ComponentModel.EditorBrowsableAttribute(System.ComponentModel.EditorBrowsableState.Advanced)]
3 references
193 IT2605Prac05_ACME.SvcRefBetterBookSupplier.MsgContractResponseDataSet IT2605Prac05_ACME.SvcRefBetterBookSupplier.Channel.GetBooks(request);
194 return base.Channel.GetBooks(request);
195 }
196
197 public System.Data.DataSet GetBooks(string password, string user) {
198     IT2605Prac05_ACME.SvcRefBetterBookSupplier.MsgContractRequestGetBooks inValue = new IT2605Prac05_ACME.SvcRefBetterBookSupplier.MsgContractRequestGetBooks();
199     inValue.password = password;
200     inValue.user = user;
201     IT2605Prac05_ACME.SvcRefBetterBookSupplier.MsgContractResponseDataSet retVal = ((IT2605Prac05_ACME.SvcRefBetterBookSupplier.Channel)base.Channel).GetBooks(inValue);
202     return retVal.results;
203 }
    
```

35. Return to *DalBetterBookSupplier.cs* and modify the code as shown in Figure 12.

Figure 12

```

public DataSet GetBooks()
{
    WsBookCatalogClient betterBookSupplierClient;

    betterBookSupplierClient = new WsBookCatalogClient();
    return betterBookSupplierClient.GetBooks("pa$$word", "acme123");
}

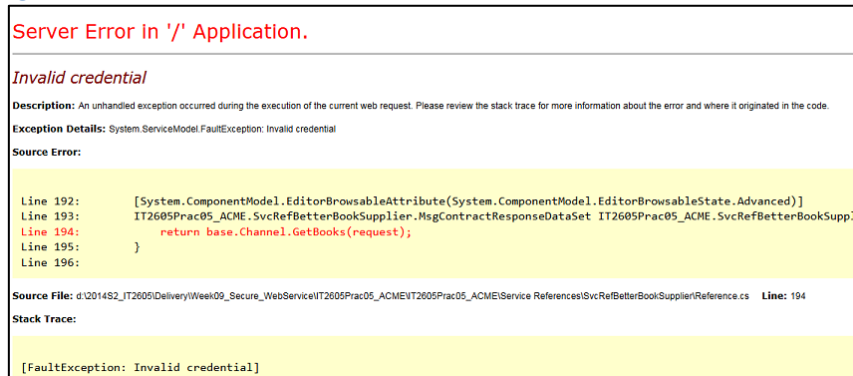
public DataSet GetBookDetails(int bookId)
{
    WsBookCatalogClient betterBookSupplierClient;

    betterBookSupplierClient = new WsBookCatalogClient();
    return betterBookSupplierClient.GetBookDetails("pa$$word", "acme123", bookId);
}
    
```

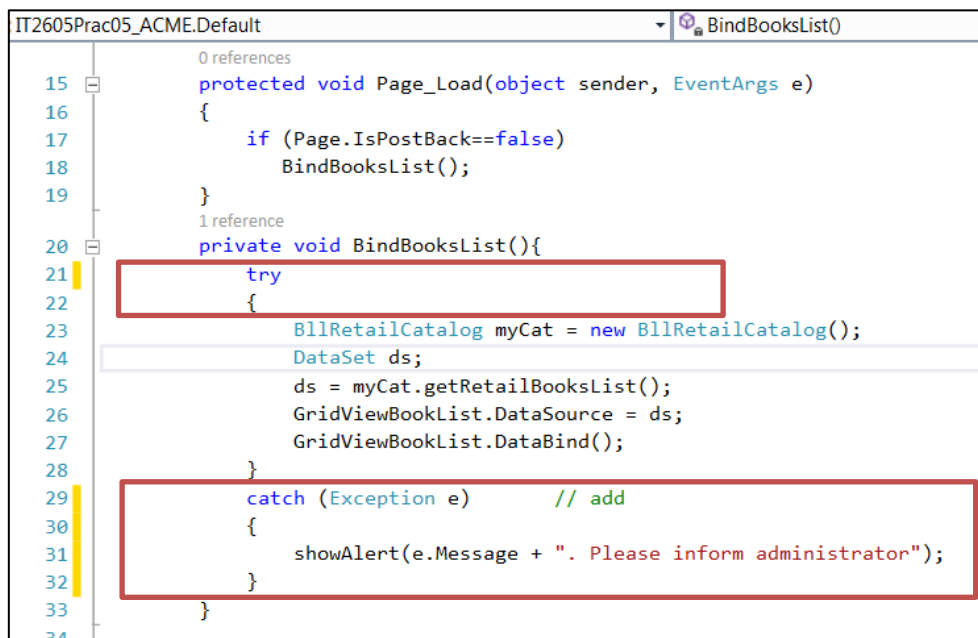
36. Save all your files, build solution and fix all errors before continuing.

37. View *Default.aspx* again. Did you get an error similar to Figure 13?

Figure 13



38. We should probably manage the exception message. Add the **try.... catch** block to catch the exception.

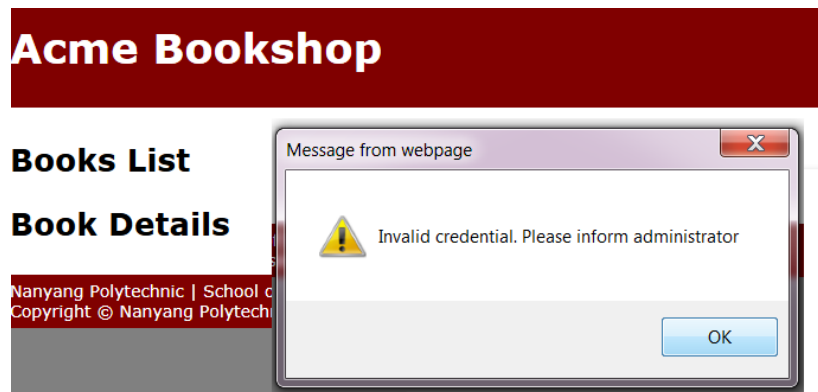


The `e.Message` returns the detail fault exception message from the service provider.

39. Add the **showAlert** method in *Default.aspx*. This method displays error using javascript alert box.

```
private void showAlert(string excepMsg)
{
    StringBuilder sb = new StringBuilder();
    sb.Append("<script type = 'text/javascript'>");
    sb.Append("window.onload=function(){");
    sb.Append("alert('");
    sb.Append(excepMsg);
    sb.Append("');");
    sb.Append("</script>");
    ClientScript.RegisterClientScriptBlock(this.GetType(), "alert", sb.ToString());
}
```

40. Press F5 to run Default.aspx, alert message is prompted.



41. The program exposes too detail information. Change the alert message to minimize the security risk by restricting the display of exception message.

```
showAlert("Invalid credential, please inform administrator");
```

42. Back to step 35 and check. Edit the code to use the correct user id from test case in step 25 and view *Default.aspx* again. It should work this time.

~ End ~