Enrolment Application

The aim of this small app is to revise the purpose of UML and start creating multi layered applications. For this application we will NOT be using any sqlDataSource controls. We will instead use an object data source control and code in the code behind to access our business objects.

Analysis

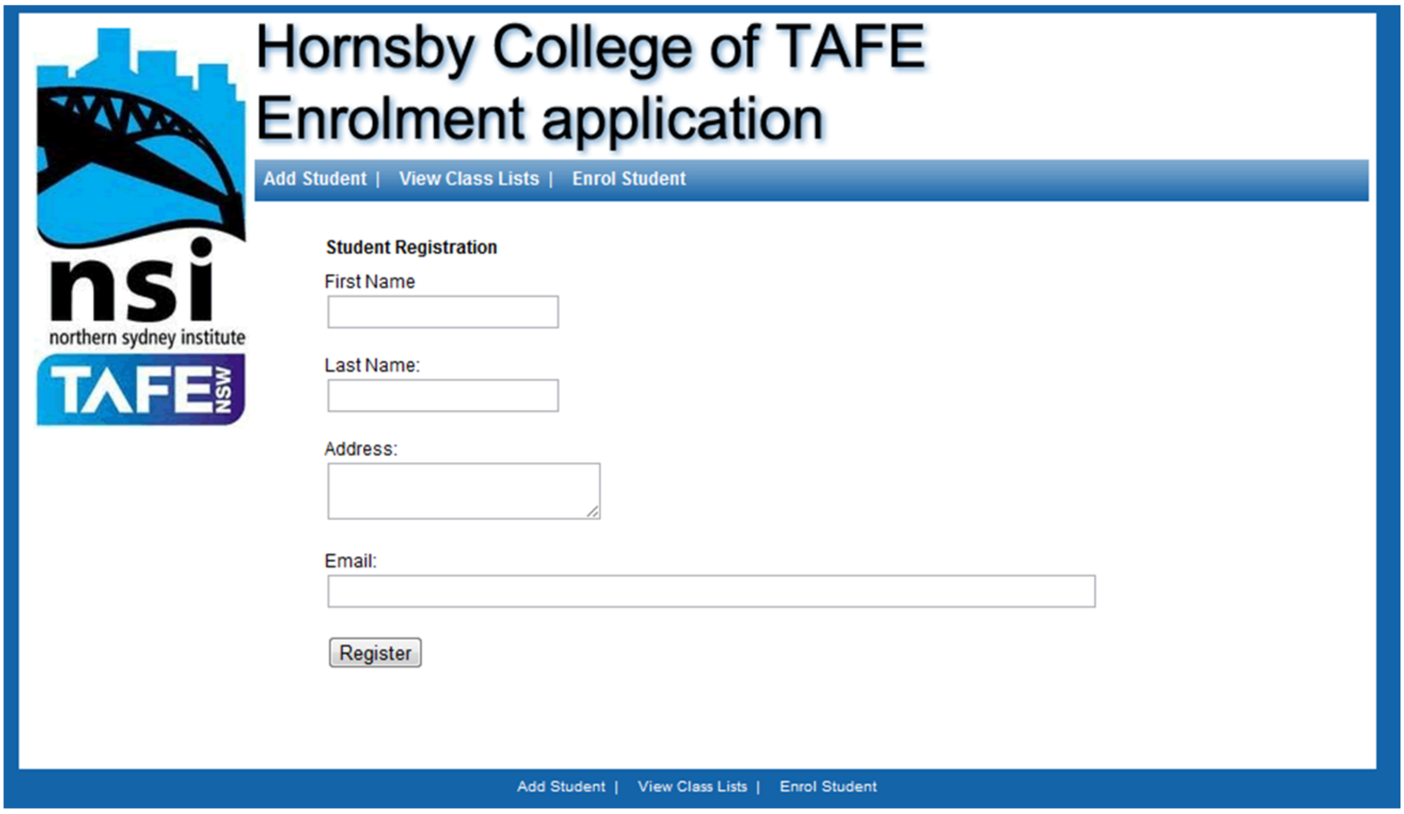
TAFE requires an enrolment application. This application will be web based and will only be available through the local network (intranet). It will:

- Provide capability for students to register as a new student by entering their details. They will be required to provide their first and last name, email and address
- Provide the capability for the enrolment officer to view course class list.
- Provide the capability for the enrolment officer to enrol a student into a course.

The analysis phase of an application helps identify the requirements this is where you would create the wireframes and use cases.

Wireframes

Register as a new student



nsi
northern sydney institute
TAFE NSW
Add Student | View Class Lists | Enrol Student
Register
Add Student | View Class Lists | Enrol Student

Hornsby College of TAFE
Enrolment application

Student Registration

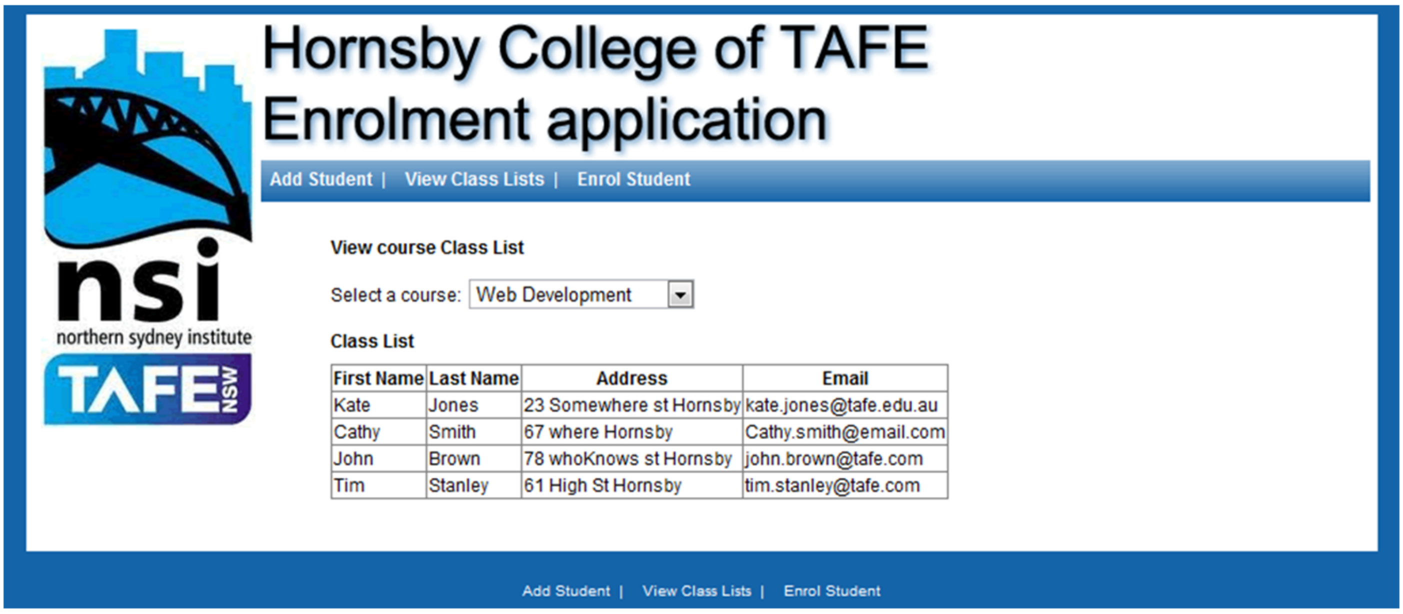
First Name

Last Name:

Address:

Email:

View course class list


Hornsby College of TAFE
Enrolment application
Add Student | View Class Lists | Enrol Student
nsi
northern sydney institute
TAFE NSW

View course Class List

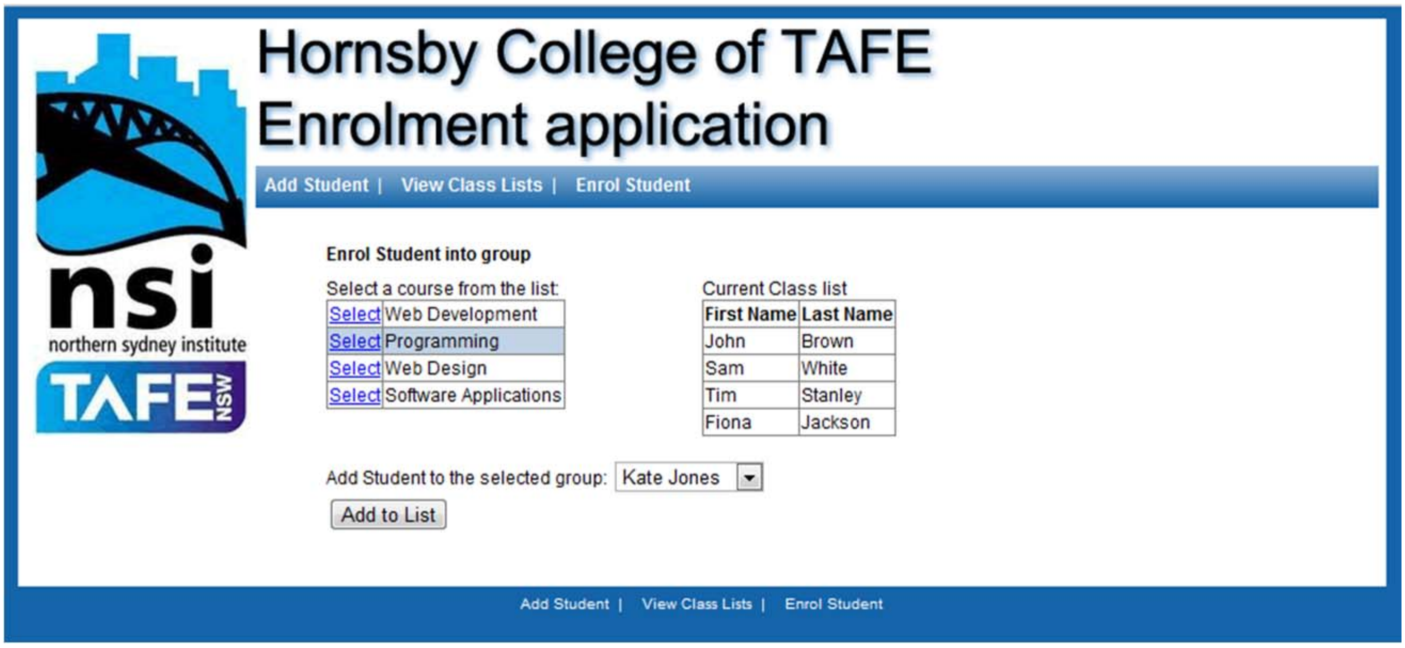
Select a course: Web Development

Class List

First Name	Last Name	Address	Email
Kate	Jones	23 Somewhere st Hornsby	kate.jones@tafe.edu.au
Cathy	Smith	67 where Hornsby	Cathy.smith@email.com
John	Brown	78 whoKnows st Hornsby	john.brown@tafe.com
Tim	Stanley	61 High St Hornsby	tim.stanley@tafe.com

Add Student | View Class Lists | Enrol Student

Enrolling a student into a course class


Hornsby College of TAFE
Enrolment application
Add Student | View Class Lists | Enrol Student
nsi
northern sydney institute
TAFE NSW

Enrol Student into group

Select a course from the list:

Select	Web Development
Select	Programming
Select	Web Design
Select	Software Applications

Current Class list

First Name	Last Name
John	Brown
Sam	White
Tim	Stanley
Fiona	Jackson

Add Student to the selected group: Kate Jones

Add to List
Add Student | View Class Lists | Enrol Student

Activity 1. Your first task is to create a use case diagram and description for this system

The purpose of the use case diagram and descriptions is to list out what functionality will be provided by the system and who or what will access that functionality.

Design

As we will be building a multi layered application we need to know which objects need to be included in our application. We will be using a database so we need a class that will provide the data access functionality. For the other classes start by reading through the use case diagram and descriptions and pick out all the nouns they may become some of your classes.

Activity 2. Create sequence diagrams

Create a sequence diagram for each of the following functions:

- Provide capability for students to register as a new student by entering their details. They will be required to provide their first and last name, email and address
- Provide the capability for the enrolment officer to view course class list.
- Provide the capability for the enrolment officer to enrol a student into a course.

Activity 3. Create the class diagram listing out the methods

Database diagram

At this point you should have a good idea of the database requirements. You should be able to create a database diagram and create the database.

For the purpose of this exercise the database has been supplied to you.

Implementation

Now we are ready to start creating the application.

Add new student into application

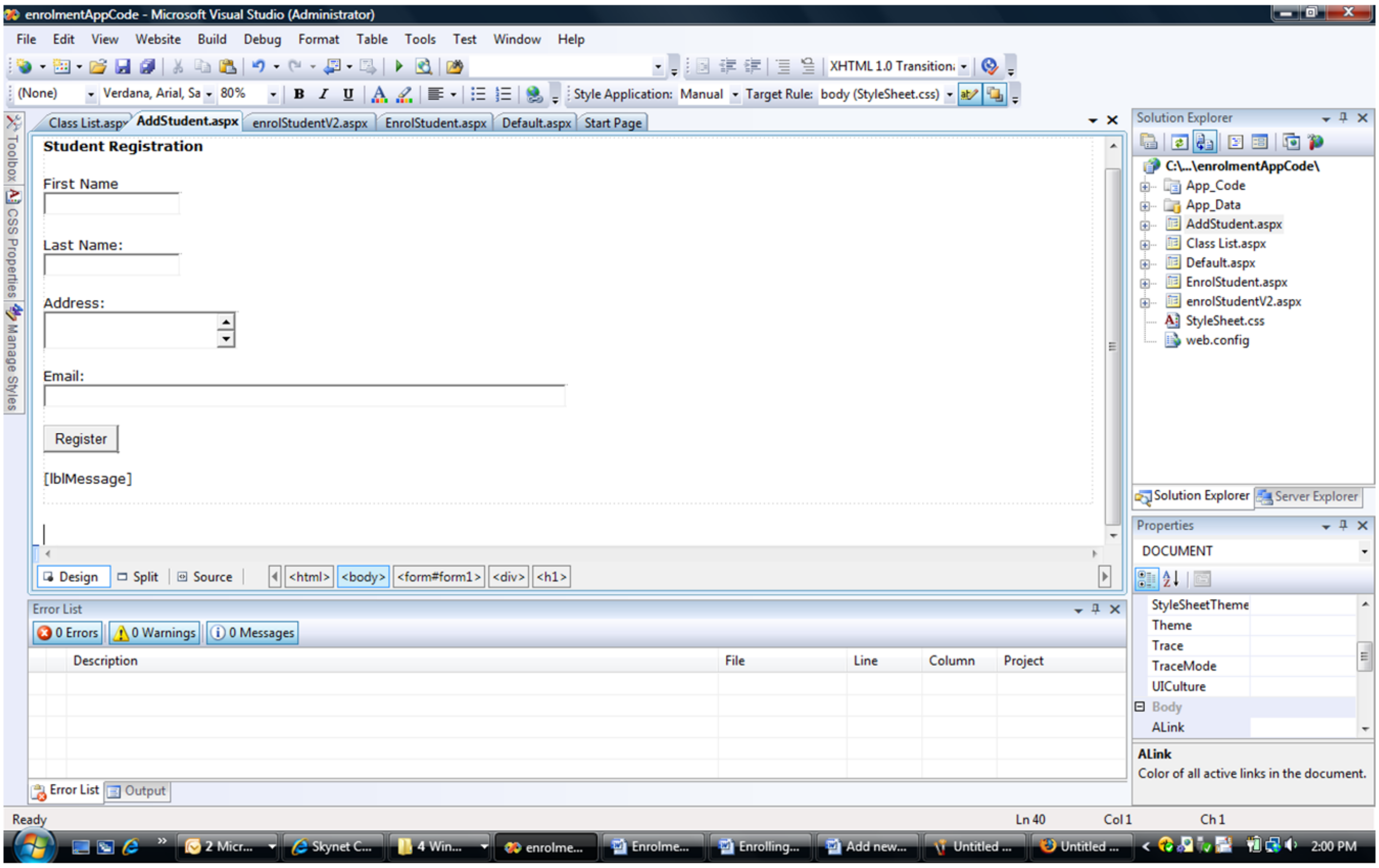
1. Create stored procedure

```
CREATE PROCEDURE dbo.uspAddStudent
      (
      @firstName varchar(50),
      @lastName varchar(50),
      @address varchar(200),
      @email varchar(100)
      )
AS
      insert into tblStudent(firstName, lastName, address, email)
      values(@firstname, @lastName, @address, @email)
      RETURN
```

2. Create method in student class

```
using System;
using ...
using System.Data.SqlClient;
using System.Data;
using System.Configuration;

/// <summary>
/// Summary description for Student
/// </summary>
public class Student
{
    private SqlHelper _objDAL;

      public Student()
      {
            //create an instance of SQL Helper
         _objDAL = new SqlHelper();
      }

    public int AddStudent(string firstName, string lastName, string address,
string email)
    {
            string strSQL = "uspAddStudent";

            //populate parameters
            SqlParameter[] objParams;
            objParams = new SqlParameter[4];
            objParams[0] = new SqlParameter("@lastName", SqlDbType.VarChar,
50);
            objParams[0].Value = lastName;
            objParams[1] = new SqlParameter("@firstName", SqlDbType.VarChar,
50);
            objParams[1].Value = firstName;
            objParams[2] = new SqlParameter("@address", SqlDbType.VarChar,
200);
            objParams[2].Value = address;
            objParams[3] = new SqlParameter("@email", SqlDbType.VarChar, 100);
            objParams[3].Value = email;

            return _objDAL.NonQuerySQL(strSQL, objParams);
    }
}
```

3. Create web form


enrolmentAppCode - Microsoft Visual Studio (Administrator)
File Edit View Website Build Debug Format Table Tools Test Window Help
XHTML 1.0 Transition
(None) Verdana, Arial, Sa 80%
Style Application: Manual Target Rule: body (StyleSheet.css)
Class List.aspx AddStudent.aspx enrolStudentV2.aspx EnrolStudent.aspx Default.aspx Start Page
Toolbox CSS Properties Manage Styles
Student Registration
First Name
Last Name:
Address:
Email:
Register
[lblMessage]
Design Split Source
<html> <body> <form#form1> <div> <h1>
Error List
0 Errors 0 Warnings 0 Messages
Description File Line Column Project
Error List Output
Solution Explorer
C:\...\enrolmentAppCode\
App_Code
App_Data
AddStudent.aspx
Class List.aspx
Default.aspx
EnrolStudent.aspx
enrolStudentV2.aspx
StyleSheet.css
web.config
Solution Explorer Server Explorer
Properties
DOCUMENT
StyleSheetTheme
Theme
Trace
TraceMode
UICulture
Body
ALink
ALink
Color of all active links in the document.
Ready Ln 40 Col 1 Ch 1
2 Micr... Skynet C... 4 Win... enrolme... Enrolme... Enrolling... Add new... Untitled ... Untitled ... 2:00 PM

Add code to code behind

```
protected void btnRegister_Click(object sender, EventArgs e)
    {
        //create instance of student
        Student objStudent = new Student();

        objStudent.AddStudent(txtFirstName.Text, txtLastName.Text,
txtAddress.Text, txtEmail.Text);

        lblMessage.Text = "New Student added";
    }
```

View course class list

1. Create stored procedures

This functionality requires 2 stored procedures. One to get a listing of the courses offered and another to list all the students enrolled in that course.

```
CREATE PROCEDURE dbo.uspRetrieveCourses

AS
	Select * from tblCourse

	RETURN
```

```
CREATE PROCEDURE dbo.uspRetrieveList

	(
	@courseID int

	)

AS
	select firstName, lastName, address, email
	from tblstudent
	inner join tblEnrol on tblStudent.studentID = tblEnrol.studentID
	where courseID = @courseID
	RETURN
```

2. Create method in course and student class

Course class:

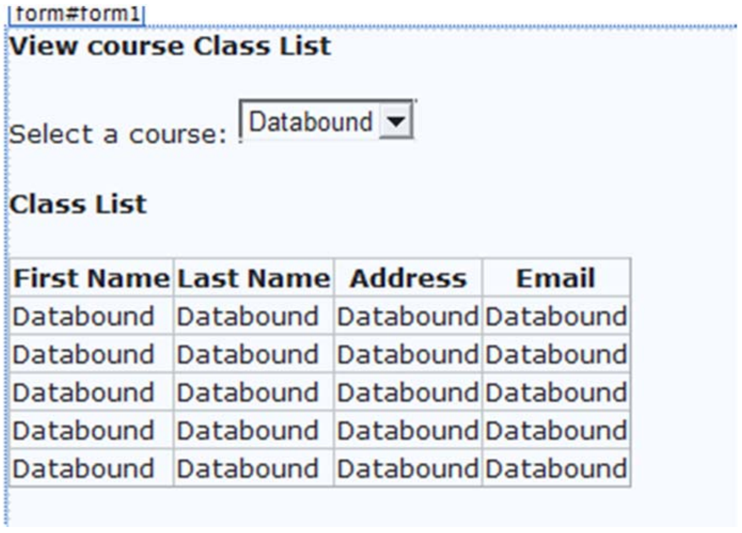
```
using ...
using System.Data.SqlClient;

public class Course
{

    //private variables
    private SqlHelper _objDAL;

      public Course()
      {
        //create instance of SQLHelper class
        _objDAL = new SqlHelper();
      }

    public SqlDataReader retrieveCourses()
    {
        string strSQL = "uspRetrieveCourses";

        return _objDAL.executeSQL(strSQL);
    }

}
```

Student class:

We already have a student class add the following method to it:

```
public SqlDataReader retrieveList(int courseID)
{
        string strSQL = "uspRetreiveList";

        //populate parameters
        SqlParameter[] objParams;
        objParams = new SqlParameter[1];
        objParams[0] = new SqlParameter("@courseID", SqlDbType.Int);
        objParams[0].Value = courseID;

        return _objDAL.executeSQL(strSQL, objParams);
}
```

3. Create web form


form#form1

View course Class List

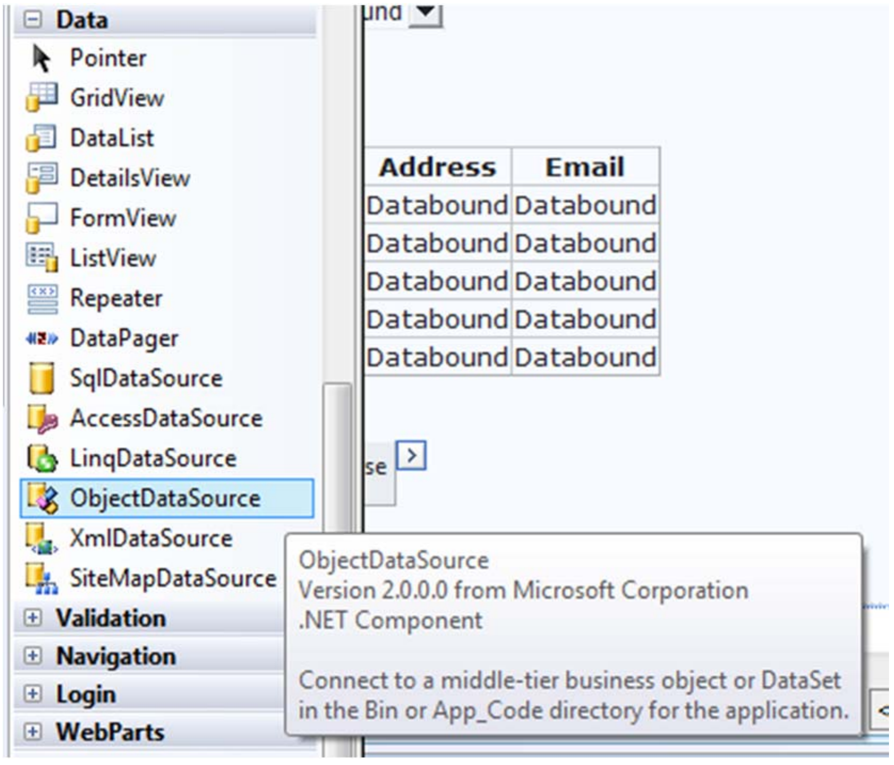
Select a course: Databound

Class List

First Name	Last Name	Address	Email
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound
Databound	Databound	Databound	Databound

The data for the drop down and gridview will come from object data sourse controls. These controls allow us to retrieve the data from objects instead of directly accessing the database.

Add an object data source control. This can be found in the data section of the toolbox:

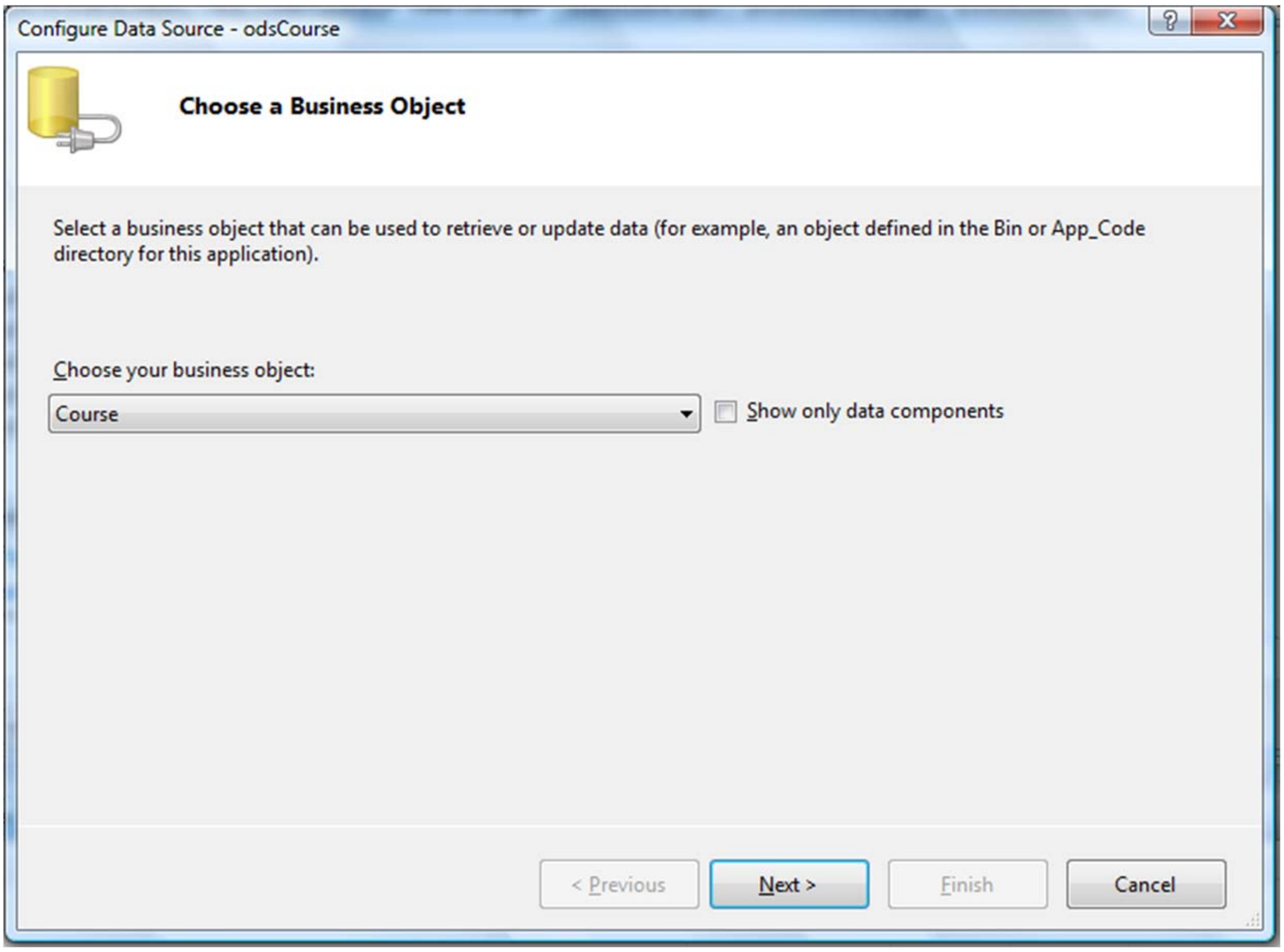


Address	Email
Databound	Databound
Databound	Databound
Databound	Databound
Databound	Databound
Databound	Databound

ObjectDataSource
Version 2.0.0.0 from Microsoft Corporation
.NET Component

Connect to a middle-tier business object or DataSet in the Bin or App_Code directory for the application.

Name it odsCourse. Click on the arrow and select “configure data source”. Set it up to access the course class:


Configure Data Source - odsCourse

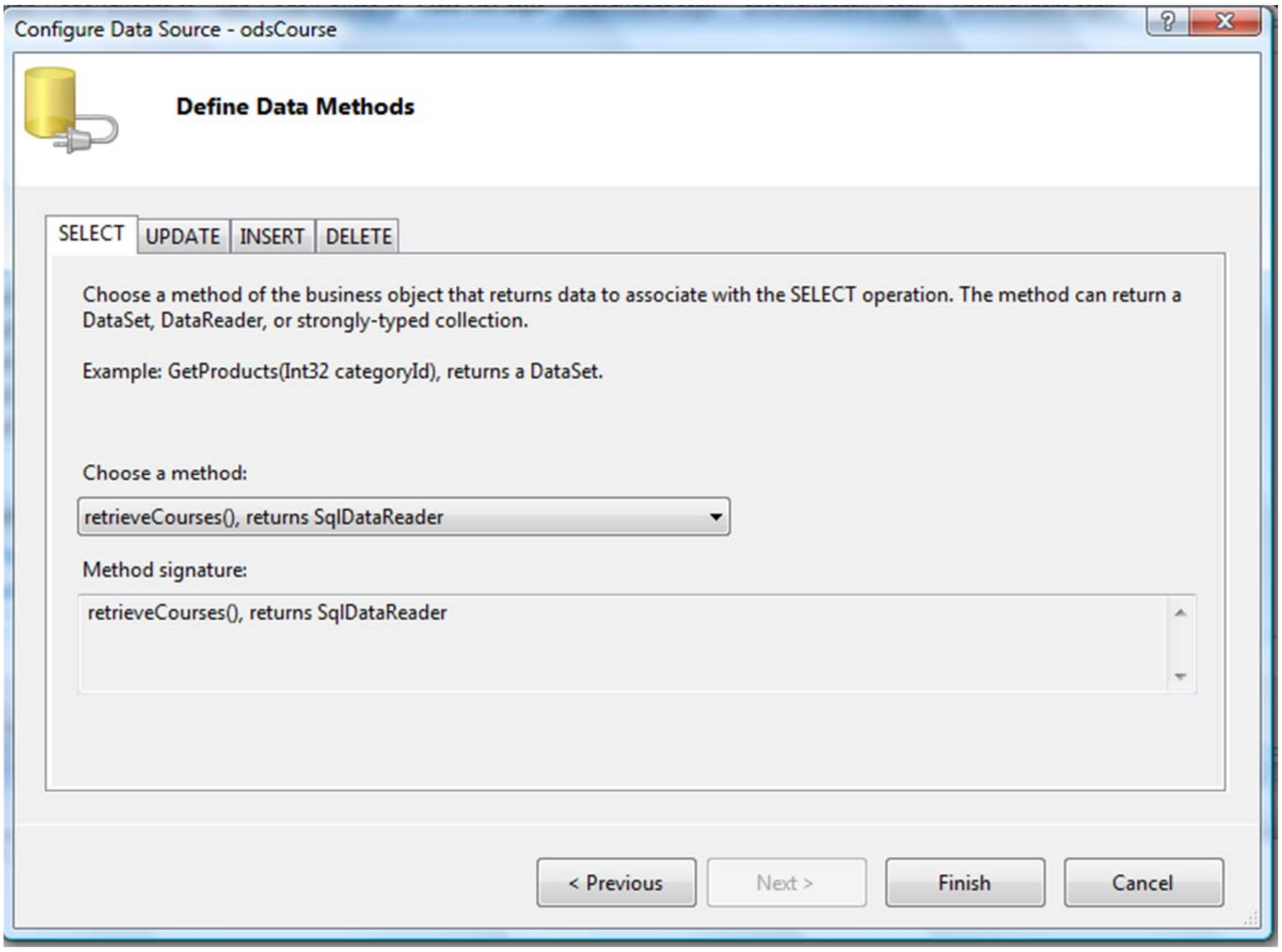
Choose a Business Object

Select a business object that can be used to retrieve or update data (for example, an object defined in the Bin or App_Code directory for this application).

Choose your business object: Course

☐ Show only data components

Click on next. Select "retrieveCourses()"


Configure Data Source - odsCourse

Define Data Methods

SELECT | UPDATE | INSERT | DELETE

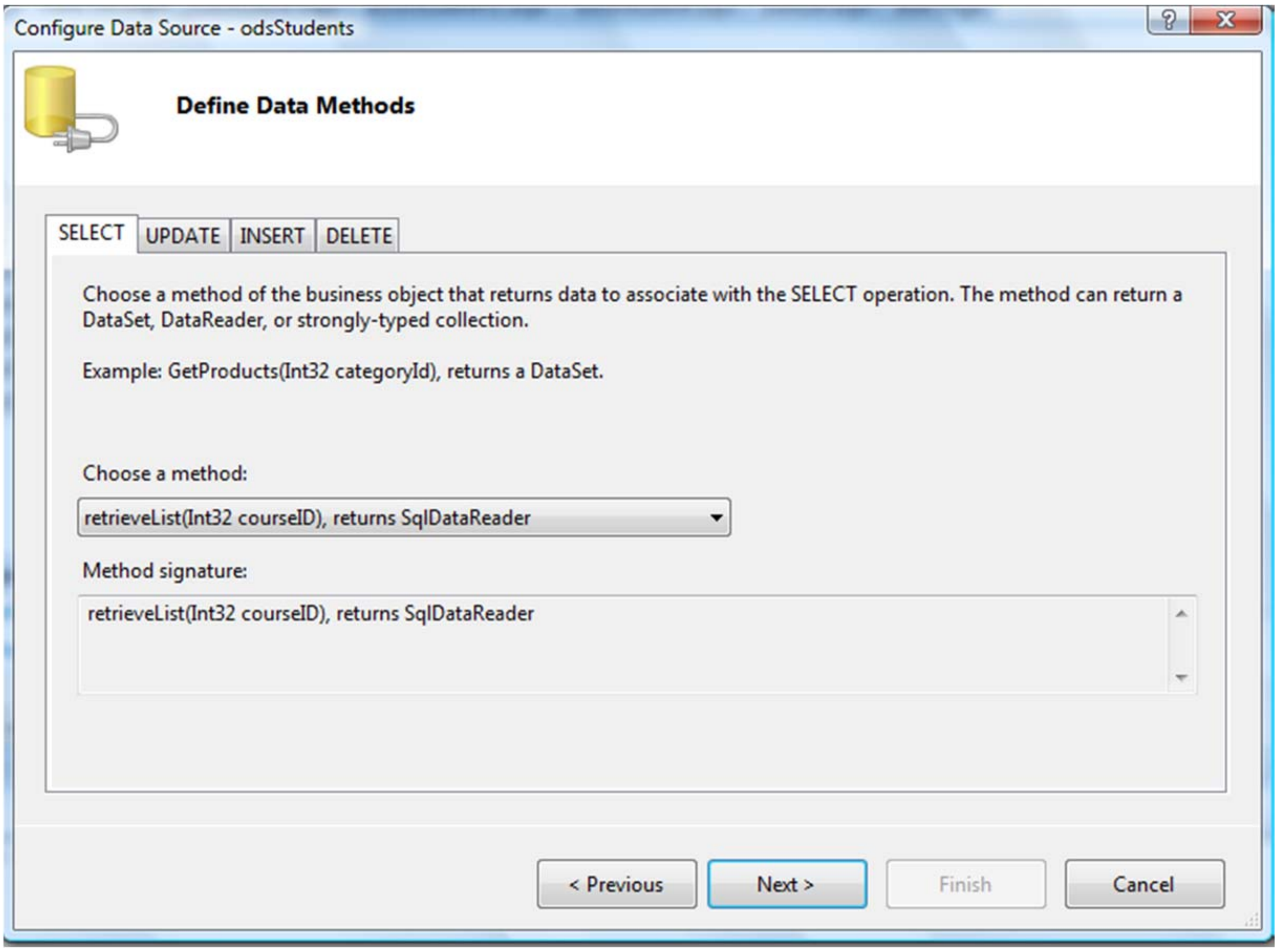
Choose a method of the business object that returns data to associate with the SELECT operation. The method can return a DataSet, DataReader, or strongly-typed collection.

Example: GetProducts(Int32 categoryId), returns a DataSet.

Choose a method: retrieveCourses(), returns SqlDataReader

Method signature: retrieveCourses(), returns SqlDataReader

Add another object data source control on to your web page set it up to use the student class and select the "retrieveList" method.



Configure Data Source - odsStudents

Define Data Methods

Choose a method of the business object that returns data to associate with the SELECT operation. The method can return a DataSet, DataReader, or strongly-typed collection.

Example: GetProducts(Int32 categoryId), returns a DataSet.

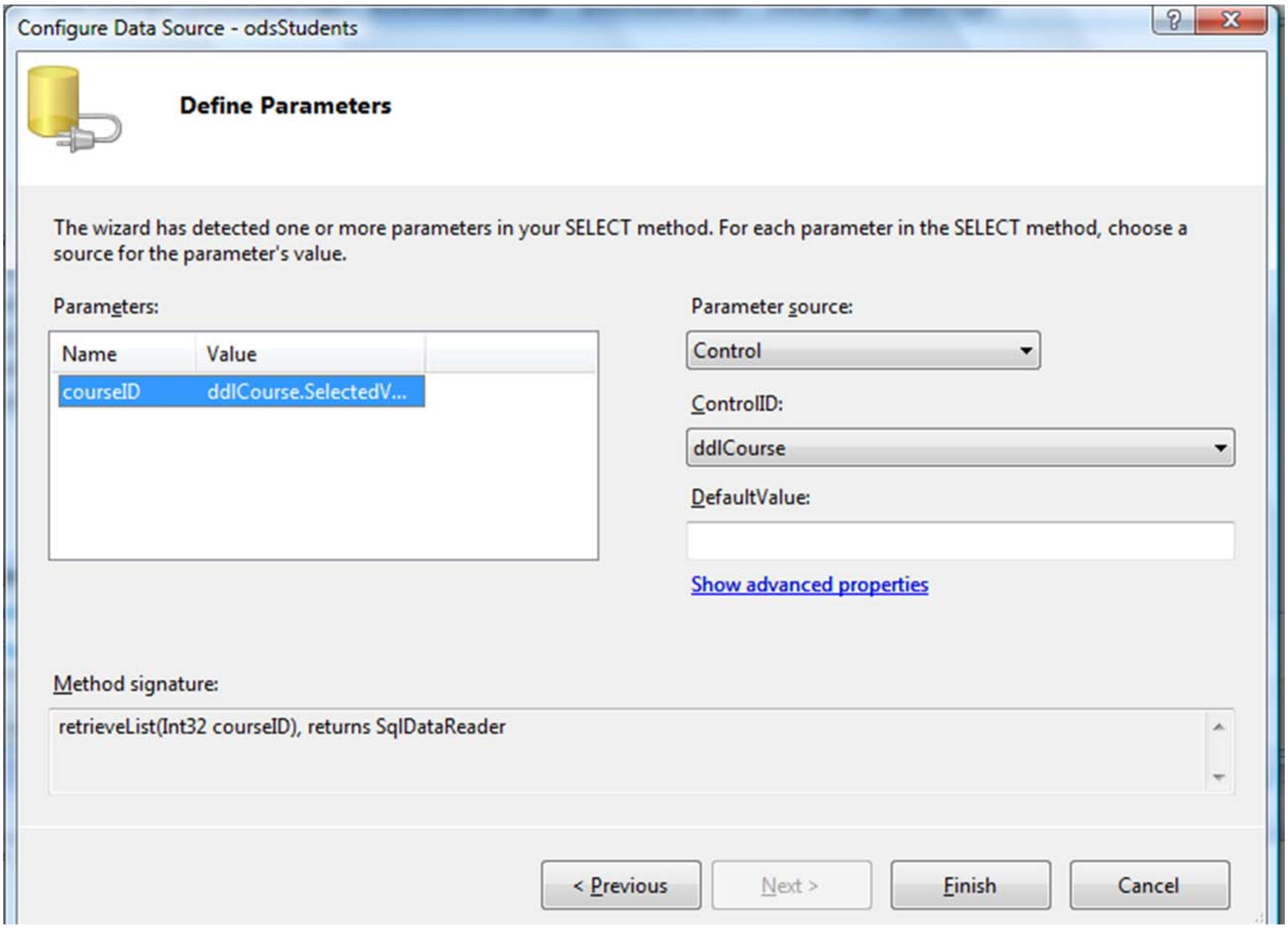
Choose a method:

retrieveList(Int32 courseID), returns SqlDataReader

Method signature:

retrieveList(Int32 courseID), returns SqlDataReader

Click on next. Just like the sql data source control you can specify where the data will come from. In our example we are wanted to select the course ID from the drop down list.


Configure Data Source - odsStudents

Define Parameters

The wizard has detected one or more parameters in your SELECT method. For each parameter in the SELECT method, choose a source for the parameter's value.

Parameters:

Name	Value
courseID	ddlCourse.SelectedV...

Parameter source: Control

ControlID: ddlCourse

DefaultValue:

Show advanced properties

Method signature:

retrieveList(Int32 courseID), returns SqlDataReader

There is no code in the code behind file.

You will need to set the following properties to the drop down list:

dataValueField: courseId

dataTextField: courseName

Enrol student into course

1. Create stored procedures

For this functionality we need 2 new stored procedures:

```
CREATE PROCEDURE dbo.uspEnrolStudent

	(
	@studentID int,
	@courseID int,
	@year varchar(50)
	)

AS
	insert into tblEnrol(studentID, courseID, yearEnrolled)
	values(@studentID, @courseID, @year)
	RETURN
```

```
CREATE PROCEDURE dbo.uspStudentsNotInCourse
(
	@courseID int
)

AS
	select firstName + ' ' + lastName as studentName, tblStudent.studentID
```

```
      from tblStudent
      where studentID not in (select studentID from tblenrol where courseID =
@courseID);

RETURN
```

2. Create methods

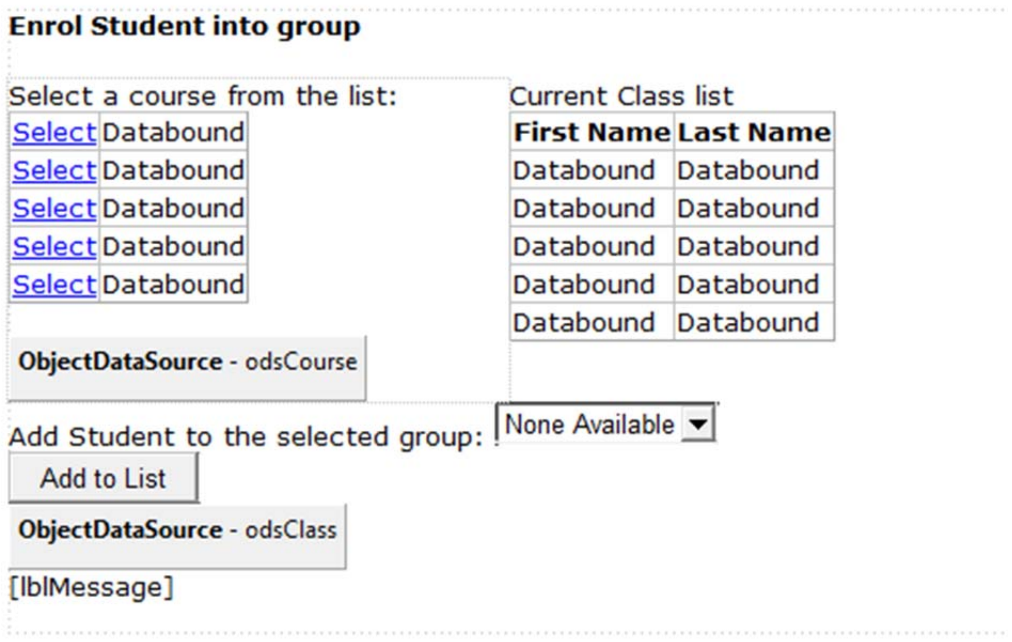
Add this method to the student class:

```
public SqlDataReader retrieveNotInCourse(int courseID)
{
        string strSQL = "uspStudentsNotInCourse";

        //populate parameters
        SqlParameter[] objParams;
        objParams = new SqlParameter[1];
        objParams[0] = new SqlParameter("@courseID", SqlDbType.Int);
        objParams[0].Value = courseID;

        return _objDAL.executeSQL(strSQL, objParams);
}
```

Add this method to the student class:

```
public int EnrolStudent(int courseID, int studentID, string year)
{
        string strSQL = "uspEnrolStudent";

        //populate parameters
        SqlParameter[] objParams;
        objParams = new SqlParameter[3];
        objParams[0] = new SqlParameter("@courseID", SqlDbType.Int);
        objParams[0].Value = courseID;
        objParams[1] = new SqlParameter("@studentID", SqlDbType.Int);
        objParams[1].Value = studentID;
        objParams[2] = new SqlParameter("@year", SqlDbType.VarChar, 50);
        objParams[2].Value = year;

        return _objDAL.NonQuerySQL(strSQL, objParams);
}
```

3. Create web form



Enrol Student into group

Select a course from the list:

Select	Databound
Select	Databound
Select	Databound
Select	Databound
Select	Databound

Current Class list

First Name	Last Name
Databound	Databound
Databound	Databound
Databound	Databound
Databound	Databound
Databound	Databound

ObjectDataSource - odsCourse

Add Student to the selected group: None Available

ObjectDataSource - odsClass

[lblMessage]

The first gridview “gvCourse” displays all the courses. Selection is enabled. The object data source control is set up to use the course class. The method is set to “retrieveCourse” this is the same method as used in a previous file.

The second gridview “gvClass” displays the students currently enrolled in the selected course. The object data source control is set up to use the student class. The method is set to “retrieveList” this is the same method as used in a previous file.

Code behind file:

```
public partial class enrolStudentV2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void gvCourse_SelectedIndexChanged(object sender, EventArgs e)
    {
        displayStudents();

    }
    protected void btnAdd_Click(object sender, EventArgs e)
    {
        //create new instance of student class
        Student objStudent = new Student();

        objStudent.EnrolStudent((int)(gvCourse.SelectedValue),
int.Parse(ddlStudents.SelectedValue), DateTime.Today.Year.ToString());

        lblMessage.Text = "Student added";

        gvClass.DataBind();
        displayStudents();
    }

    private void displayStudents()
    {
        //display students
        if (gvCourse.SelectedIndex >= 0)
        {
            //clear out the drop down list
            ddlStudents.Items.Clear();

            //a course was selected retreive all students not enrolled in that
course
            Student objStudent = new Student();

            ddlStudents.DataSource =
objStudent.retreiveNotInCourse((int)(gvCourse.SelectedValue));
            ddlStudents.DataBind();

            //if drop down list is empty add "no students" message
            if (ddlStudents.Items.Count < 1)
            {
                ddlStudents.Items.Add("No Students Available");
            }
        }
    }
}
```