**CS 555**

**HOMEWORK 3**

**Name:** Ivan Sangines Escrig

**ID#:** 968606

**Instructor:** Dr. Mahmood

**Date:** October 26, 2018

**TABLE OF CONTENT**

[**INTRODUCTION**](#_Toc446970371) 3

[**SCREENSHOTS:**](#_Toc446970373) 4

[**SOURCE CODE:**](#_Toc446970374) 9

[**CONCLUSION:**](#_Toc446970375) 55

**INTRODUCTION**

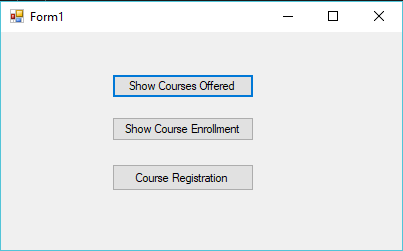
The porpuse of this assignment is to get used to and understand better the structure and concepts about Windows and Web aplications. The first thing we needed to do is to recreate the Database in order to have our data ready for our aplications.

The first part of the assignment consists on recreating the Windows Application in the handout. This will help us to understand better the structure of a Win App and how the different layers are connected to each other.

The last part of the assignment consists in converting the Win App to a Web Application. Then, we will have to do some changes in order to achieve a loose coupling relatinship between the different layers. This is achieved by creating constructors in the different layers. The benefits of a loose coupling is that if tomorrow we want to obtain the same data diferently (Oracle data for example), we won’t need to change our whole program. It can be achieved by just changing one of the layers in our program (Data Layer) and then passing the data object to the business layer, with no change needed on the rest of the layers. Once this is achieved, we were asked to create a new capability to the application and I decided to show all the available instructors. The last part was to protect our Web app using sessions. I created a log in page and protected all the oder pages,. We can not acces to any of them if we are not first logged in. In order to logg in, we just need the student NAME and ID.

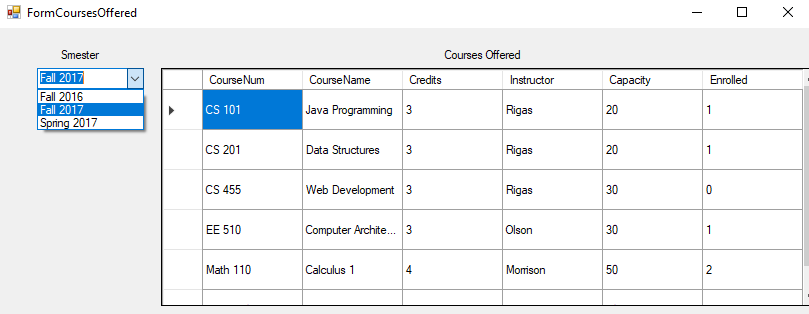
**SCREEN SHOTS:**

**Windows Application:**

****

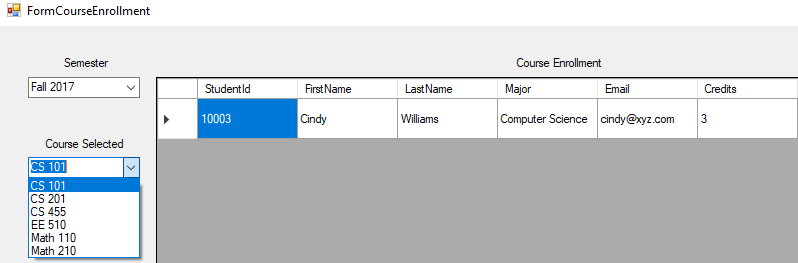
Show Courses:

This form has a dropdown with all the available semesters. Once a semester is selected, it will show us all the available courses for that semester.

****

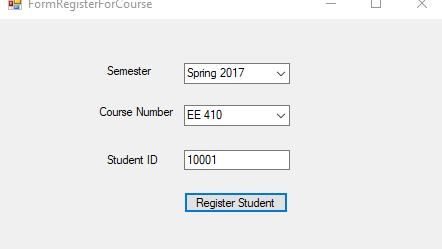
Show Courses Enrollment:

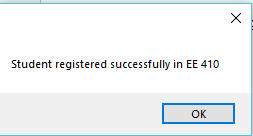
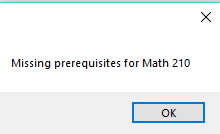
Here we can see the show courses form where we have two dropdowns. The first drop down is automatically fetched with all the available semesters. The second dropdown is fetched with all the available courses once the semester is selected (no courses for 2016). Then once the course is selected, the data is displayed.

****

Course Registration:

This form is used to register students. However, our web app will first check if the student meets the prerequisites to take such course. If the student has the requirements, a successful msg will show up, otherwise the message will tell us that the student does not meet the prerequisites and it will not add it to the course (insert it to DB).

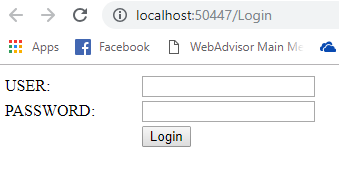


**Web Application:**

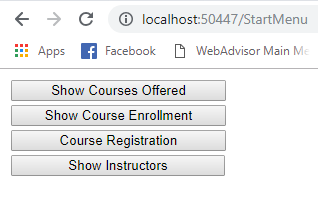
Login Page:

This is the initial page where we can log in with the student name and id number. Once we login, we will create a session object and we will have access to all the other pages. If we did not log in and we try to access to any other page, we will be redirected to the log in page.



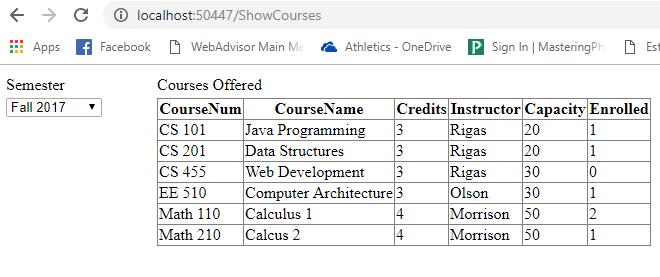
StartMenu:

Once we log in, we will get to the star menu where we can decide to what page we want to go.



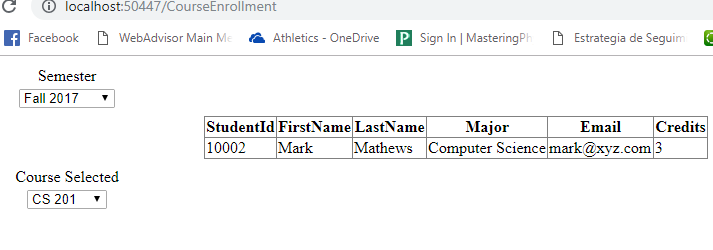
Show Courses Offered:

Once we click on the show courses offered button, a new page with a dropdown with all the semesters available will show up. Once we choose the semester, all the courses offered for that particular semester will show up in a table.



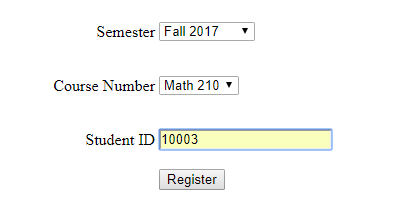
Show Courses Enrollment:

In the show courses enrollment page we will have two dropdowns. The first one will have all the semesters available. Once we choose a semester, all the courses for that semester will be fetched to the second dropdown. Once we choose a course, all the students enrolled for that course will show up.



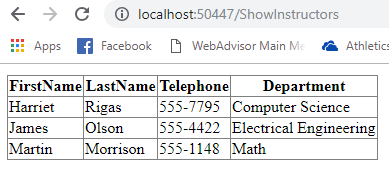
Course Registration:

Same as on the Win app, in this page we will be able to register a student for a course. However, some validations are needed (check if the students have the prerequisites). If the registration is valid, a successful msg will show up and if the registration is invalid, an error msg will show up.



Show Instructors:

In this new page, we will have all the advisors available in our Web App.



**SOURCE CODE:**

**Windows Application:**

DATA LAYER:

DataAccess:

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.DataLayer

{

class DataAccess

{

public static string ConnectionString = ConfigurationManager.ConnectionStrings["STDB2017"].ConnectionString;

public static object GetSingleAnswer(string sql, List<SqlParameter> PList,

SqlConnection connc = null, SqlTransaction sqtr =null, bool IsStoredProc = false)

{

SqlConnection conn = null;

if (sqtr == null)

conn = new SqlConnection(ConnectionString);

else

conn = connc;

object obj = null;

try

{

if (sqtr == null)

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

if (sqtr != null)

cmd.Transaction = sqtr;

if (IsStoredProc == true)

cmd.CommandType = CommandType.StoredProcedure;

if (PList != null)

{

foreach (SqlParameter param in PList)

cmd.Parameters.Add(param);

}

obj = cmd.ExecuteScalar();

}

catch(Exception)

{

throw; // send the error back to caller

}

finally

{

if (sqtr == null)

conn.Close();

}

return obj;

}

public static DataTable GetManyRowsCols(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false)

{

SqlConnection conn = null;

if (sqtr == null)

conn = new SqlConnection(ConnectionString);

else

conn = connc;

DataTable dt = new DataTable();

try

{

if (sqtr == null)

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

if (sqtr != null)

cmd.Transaction = sqtr;

if (IsStoredProc == true)

cmd.CommandType = CommandType.StoredProcedure;

if (PList != null)

{

foreach (SqlParameter param in PList)

cmd.Parameters.Add(param);

}

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = cmd;

da.Fill(dt);

}

catch(Exception)

{

throw;

}

finally

{

if (sqtr == null)

conn.Close();

}

return dt;

}

public static int InsertUpdateDelete(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false)

{

SqlConnection conn = null;

if (sqtr == null)

conn = new SqlConnection(ConnectionString);

else

conn = connc;

int rowsModified = 0;

try

{

if (sqtr == null)

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

if (sqtr != null)

cmd.Transaction = sqtr;

if (IsStoredProc == true)

cmd.CommandType = CommandType.StoredProcedure;

if (PList != null)

{

foreach (SqlParameter param in PList)

cmd.Parameters.Add(param);

}

rowsModified = cmd.ExecuteNonQuery();

}

catch(Exception)

{

throw;

}

finally

{

if (sqtr == null)

conn.Close();

}

return rowsModified;

}

}

}

DBHelper:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.DataLayer

{

class DBHelper

{

public static void AddSqlParam(List<SqlParameter> PList, string paramName,

SqlDbType paramType, object paramValue, int size=0)

{

SqlParameter p = null;

if (size == 0)

p = new SqlParameter(paramName, paramType);

else

p = new SqlParameter(paramName, paramType, size);

p.Value = paramValue;

PList.Add(p);

}

}

}

DBList:

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.DataLayer

{

class DBList

{

public static List<T> GetList<T>(DataTable dt)

where T : IEntity, new()

{ // will do conversion from dt to List<> for reference types

List<T> TList = new List<T>();

foreach (DataRow dr in dt.Rows)

{

T t1 = new T();

// populate the columns from dr into the fields of the t1

// delegate the work of populating t1 to the class T

t1.PopulateFields(dr);

TList.Add(t1);

}

return TList;

}

public static List<T> GetListValueType<T>(DataTable dt, string colname)

where T : IConvertible // will do conversion from dt to List<>for value types including string

{

List<T> TList = new List<T>();

foreach (DataRow dr in dt.Rows)

TList.Add((T)dr[colname]);

return TList;

}

}

}

IEntity:

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.DataLayer

{

interface IEntity

{ // any class that is ivolved in conversion from a DataTable will provide this method

void PopulateFields(DataRow dr);

}

}

RepositoryCourses:

using StudentWinApp.Models;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.DataLayer

{

class RepositoryCourses // purpose of repository is to serve data

{ // if data is being obtained from a Database Server, then the repository

// assembles SQL and its parameters for a given task

public int GetEnrollmentCount(string semester, string courseNum)

{

int enrollCount = 0;

try

{

string sql = "select count(\*) from CourseEnrollments where SemesterId=@SemesterId and " +

"CourseNum=@CourseNum";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

object objCount = DataAccess.GetSingleAnswer(sql, PList);

if (objCount != null)

enrollCount = int.Parse(objCount.ToString());

}

catch(Exception)

{

throw;

}

return enrollCount;

}

public List<string> GetSemesters()

{

List<string> SList = new List<string>();

try

{

string sql = "select SemesterId from Semesters";

DataTable dt = DataAccess.GetManyRowsCols(sql, null);

// convert datatable to List<string>

foreach (DataRow dr in dt.Rows)

SList.Add(dr["SemesterId"].ToString());

}

catch (Exception)

{

throw;

}

return SList;

}

public List<CourseOfferedVM> GetCoursesOffered(string semester)

{

List<CourseOfferedVM> CList = new List<CourseOfferedVM>();

try

{

string sql = "select co.CourseNum, c.CourseName, c.Credits, co.Capacity," +

"i.LastName as Instructor from CoursesOffered co " +

"inner join Courses c on co.CourseNum=c.CourseNum " +

"inner join Instructors i on co.InstructorId=i.InstructorId where " +

"co.SemesterId=@SemesterId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DataTable dt = DataAccess.GetManyRowsCols(sql, PList);

// convert data table to List<CourseOfferedVM>

foreach (DataRow dr in dt.Rows)

{

CourseOfferedVM covm = new CourseOfferedVM();

covm.CourseNum = dr["CourseNum"].ToString();

covm.CourseName = dr["CourseName"].ToString();

covm.Credits = (int)dr["Credits"];

covm.Capacity = (int)dr["Capacity"];

covm.Instructor = dr["Instructor"].ToString();

covm.Enrolled = GetEnrollmentCount(semester, covm.CourseNum);

CList.Add(covm);

}

}

catch (Exception)

{

throw;

}

return CList;

}

public List<CourseEnrollmentVM> GetCourseEnrollment(string semester, string courseNum)

{

List<CourseEnrollmentVM> CList = new List<CourseEnrollmentVM>();

try

{

string sql = "select s.StudentId, s.FirstName, s.LastName, s.Email, c.Credits," +

"d.DepartmentName as Major from Students s " +

"inner join CourseEnrollments ce on s.StudentId=ce.StudentId " +

"inner join Courses c on ce.CourseNum=c.CourseNum " +

"inner join StudentDepartments sd on s.StudentId=sd.StudentId " +

"inner join Departments d on sd.DepartmentId=d.DepartmentId " +

"where ce.SemesterId=@SemesterId and ce.CourseNum=@CourseNum";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

DataTable dt = DataAccess.GetManyRowsCols(sql, PList);

// convert datatable to List<CurseEnrollmentVM>

CList = DBList.GetList<CourseEnrollmentVM>(dt);

}

catch(Exception)

{

throw;

}

return CList;

}

public List<string> GetCoursesOfferedForASemester(string semester)

{

List<string> CList = new List<string>();

try

{

string sql = "select CourseNum from CoursesOffered where SemesterId=@SemesterId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DataTable dt = DataAccess.GetManyRowsCols(sql, PList);

// convert dt to List<string>

//foreach (DataRow dr in dt.Rows)

// CList.Add(dr["CourseNum"].ToString());

CList = DBList.GetListValueType<string>(dt,"CourseNum");

}

catch(Exception)

{

throw;

}

return CList;

}

public List<string> GetPreRequisiteCourses(string courseNum)

{

List<string> CList = new List<string>();

try

{

string sql = "select PreReqCourseNum from CoursePreRequisites where CourseNum=@CourseNum";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

DataTable dt = DataLayer.DataAccess.GetManyRowsCols(sql, PList);

CList = DBList.GetListValueType<string>(dt, "PreReqCourseNum");

}

catch (Exception)

{

throw;

}

return CList;

}

public bool IsThereRoomInTheCourse(string semester, string courseNum)

{

bool ret = false;

try

{

string sql = "select Capacity from CoursesOffered where CourseNum=@CourseNum and " +

"SemesterId=@SemesterId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

object objCap = DataLayer.DataAccess.GetSingleAnswer(sql, PList);

if (objCap != null)

{

int capacity = int.Parse(objCap.ToString());

int enrollCount = GetEnrollmentCount(semester, courseNum);

if (enrollCount < capacity)

ret = true;

}

}

catch (Exception)

{

throw;

}

return ret;

}

}

}

RepositoryStudents:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.DataLayer

{

class RepositoryStudents

{

public bool DoesStudentExist(long studentId)

{

bool ret = false;

try

{

string sql = "select StudentId from Students where StudentId=@StudentId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@StudentId", SqlDbType.BigInt, studentId);

object objId = DataAccess.GetSingleAnswer(sql, PList);

if (objId != null)

ret = true;

}

catch (Exception)

{

throw;

}

return ret;

}

public float? GetGradeForACourse(long studentId, string courseNum)

{

float? grade = null;

try

{

string sql = "select grade from CoursesCompleted where " +

"CourseNum=@CourseNum and StudentId=@StudentId ";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@StudentId", SqlDbType.BigInt, studentId);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

object objGrade = DataAccess.GetSingleAnswer(sql, PList);

if (objGrade != null)

grade = float.Parse(objGrade.ToString());

}

catch (Exception)

{

throw;

}

return grade;

}

public bool HasStudentTakenPreRequisteCourses(long studentId, string courseNum, float minGrade)

{

bool ret = true;

try

{

RepositoryCourses repCourses = new RepositoryCourses();

List<string> PreReqList = repCourses.GetPreRequisiteCourses(courseNum);

// determine the grade and see if each course is completed with minGrade or better

foreach (string cnum in PreReqList)

{

float? grade = GetGradeForACourse(studentId, cnum);

if ((grade == null) || (grade < minGrade))

{

ret = false;

break;

}

}

}

catch (Exception)

{

throw;

}

return ret;

}

public bool RegisterStudent(long studentId, string semester, string courseNum)

{

bool ret = false;

try

{

string sql = "Insert into CourseEnrollments(StudentId,SemesterId,CourseNum) values" +

"(@StudentId,@SemesterId,@CourseNum)";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@StudentId", SqlDbType.BigInt, studentId);

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

int rows = DataAccess.InsertUpdateDelete(sql, PList);

if (rows > 0)

ret = true;

}

catch (Exception)

{

throw;

}

return ret;

}

}

}

BUSINESS LAYER:

BusinessCourses:

using StudentWinApp.DataLayer;

using StudentWinApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.BusinessLayer

{

class BusinessCourses

{

RepositoryCourses \_rep = new RepositoryCourses();

// the UI layer always communicates with the business layer

// even if no code is needed, the business layer forwards the request to repository in data layer

public List<string> GetSemesters()

{

return \_rep.GetSemesters();

}

public List<CourseOfferedVM> GetCoursesOffered(string semester)

{

return \_rep.GetCoursesOffered(semester);

}

public List<CourseEnrollmentVM> GetCourseEnrollment(string semester, string courseNum)

{

return \_rep.GetCourseEnrollment(semester, courseNum);

}

public List<string> GetCoursesOfferedForASemester(string semester)

{

return \_rep.GetCoursesOfferedForASemester(semester);

}

}

}

BusinessStudents:

using StudentWinApp.DataLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.BusinessLayer

{

class BusinessStudents

{

RepositoryCourses repCourses = new RepositoryCourses();

RepositoryStudents repStudents = new RepositoryStudents();

public bool RegisterStudent(long studentId, string semester, string courseNum)

{

bool ret = false;

try

{

if (repStudents.DoesStudentExist(studentId))

{

if (repStudents.HasStudentTakenPreRequisteCourses(studentId, courseNum, 2.0f))

{

if (repCourses.IsThereRoomInTheCourse(semester, courseNum))

ret = repStudents.RegisterStudent(studentId, semester, courseNum);

else

throw new Exception("Course capacity exceeded..");

}

else

throw new Exception("Missing prerequisites for " + courseNum);

}

else

throw new Exception("Invalid student Id..");

}

catch (Exception)

{

throw;

}

return ret;

}

}

}

MODELS:

CourseEnrollmentVM:

using StudentWinApp.DataLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

namespace StudentWinApp.Models

{

class CourseEnrollmentVM : MyEntityBase//IEntity

{

public long StudentId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Major { get; set; }

public string Email { get; set; }

public int Credits { get; set; }

}

}

CourseOferedVM:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWinApp.Models

{

class CourseOfferedVM

{

public string CourseNum { get; set; }

public string CourseName { get; set; }

public int Credits { get; set; }

public string Instructor { get; set; }

public int Capacity { get; set; }

public int Enrolled { get; set; }

}

}

MyEntityBase:

using StudentWinApp.DataLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Reflection;

namespace StudentWinApp.Models

{

class MyEntityBase : IEntity

{

public void PopulateFields(DataRow dr)

{

// use reflection to populate the fields of this class from DataRow

Type tp = this.GetType();

foreach (PropertyInfo pi in tp.GetProperties())

pi.SetValue(this, dr[pi.Name]);

}

}

}

FORMS:

Form1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace StudentWinApp

{

public partial class Form1 : Form

{

FormCoursesOffered frmCoursesOffered = null;

FormCourseEnrollment frmCourseEnrollment = null;

FormRegisterForCourse frmRegisterForACourse = null;

public Form1()

{

InitializeComponent();

}

private void btnShowCoursesOffered\_Click(object sender, EventArgs e)

{

if (frmCoursesOffered == null)

{

frmCoursesOffered = new FormCoursesOffered();

frmCoursesOffered.Show(); // modeless display

}

else

{

if (frmCoursesOffered.IsDisposed)

{

frmCoursesOffered = new FormCoursesOffered();

frmCoursesOffered.Show(); // modeless display

}

else

frmCoursesOffered.WindowState = FormWindowState.Normal;

}

}

private void btnShowCourseEnrollment\_Click(object sender, EventArgs e)

{

if (frmCourseEnrollment == null)

{

frmCourseEnrollment = new FormCourseEnrollment();

frmCourseEnrollment.Show(); // modeless display

}

else

{

if (frmCourseEnrollment.IsDisposed)

{

frmCourseEnrollment = new FormCourseEnrollment();

frmCourseEnrollment.Show(); // modeless display

}

else

frmCourseEnrollment.WindowState = FormWindowState.Normal;

}

}

private void btnShowRegisterForACourse\_Click(object sender, EventArgs e)

{

if (frmRegisterForACourse == null)

{

frmRegisterForACourse = new FormRegisterForCourse();

frmRegisterForACourse.Show(); // modeless display

}

else

{

if (frmRegisterForACourse.IsDisposed)

{

frmRegisterForACourse = new FormRegisterForCourse();

frmRegisterForACourse.Show(); // modeless display

}

else

frmRegisterForACourse.WindowState = FormWindowState.Normal;

}

}

}

}

FormCourseEnrollment:

using StudentWinApp.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace StudentWinApp

{

public partial class FormCourseEnrollment : Form

{

BusinessLayer.BusinessCourses \_busCourses = new BusinessLayer.BusinessCourses();

public FormCourseEnrollment()

{

InitializeComponent();

}

private void FormCourseEnrollment\_Load(object sender, EventArgs e)

{

try

{

List<string> SList = \_busCourses.GetSemesters();

cmbSemesters.DataSource = SList;

cmbSemesters.Refresh();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void cmbSemesters\_SelectedIndexChanged(object sender, EventArgs e)

{

string semester = cmbSemesters.Text;

// cmbCourses.Items.Clear();

if (semester.IndexOf("DataRowView") < 0)

{

try

{

List<string> CList = \_busCourses.GetCoursesOfferedForASemester(semester);

cmbCourses.DataSource = CList;

cmbCourses.Refresh();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

private void cmbCourses\_SelectedIndexChanged(object sender, EventArgs e)

{

string semester = cmbSemesters.Text;

string courseNum = cmbCourses.Text;

if (courseNum.IndexOf("DataRowView") < 0)

{

try

{

List<CourseEnrollmentVM> EList = \_busCourses.GetCourseEnrollment(semester, courseNum);

dgEnrollment.DataSource = EList;

dgEnrollment.Refresh();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

}

FormCoursesOffered:

using StudentWinApp.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace StudentWinApp

{

public partial class FormCoursesOffered : Form

{

BusinessLayer.BusinessCourses \_busCourses = new BusinessLayer.BusinessCourses();

public FormCoursesOffered()

{

InitializeComponent();

}

private void FormCoursesOffered\_Load(object sender, EventArgs e)

{

try

{

List<string> SList = \_busCourses.GetSemesters();

// databinding : assign the datasource and call refresh

cmbSemesters.DataSource = SList;

cmbSemesters.Refresh();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void cmbSemesters\_SelectedIndexChanged(object sender, EventArgs e)

{

string semester = cmbSemesters.Text;

if (semester.IndexOf("DataRowView") < 0)

{

try

{

List<CourseOfferedVM> CList = \_busCourses.GetCoursesOffered(semester);

dgCoursesOffered.DataSource = CList;

dgCoursesOffered.Refresh();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

}

FormRegisterForCourses:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace StudentWinApp

{

public partial class FormRegisterForCourse : Form

{

BusinessLayer.BusinessCourses \_busCourses = new BusinessLayer.BusinessCourses();

BusinessLayer.BusinessStudents \_busStudents = new BusinessLayer.BusinessStudents();

public FormRegisterForCourse()

{

InitializeComponent();

}

private void FormRegisterForCourse\_Load(object sender, EventArgs e)

{

try

{

List<string> SList = \_busCourses.GetSemesters();

cmbSemesters.DataSource = SList;

cmbSemesters.Refresh();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void cmbSemesters\_SelectedIndexChanged(object sender, EventArgs e)

{

string semester = cmbSemesters.Text;

if (semester.IndexOf("DataRowView") < 0)

{

try

{

// populate the cmbCoursesOffered with the courses being offered for the selected semester

List<string> CList = \_busCourses.GetCoursesOfferedForASemester(semester);

cmbCoursesOffered.DataSource = CList;

cmbCoursesOffered.Refresh();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

private void btnRegisterForCourse\_Click(object sender, EventArgs e)

{

string semester = cmbSemesters.Text;

string courseNum = cmbCoursesOffered.Text;

try

{

bool ret = \_busStudents.RegisterStudent(long.Parse(txtStudentId.Text), semester, courseNum);

if (ret == true)

MessageBox.Show("Student registered successfully in " + courseNum);

else

MessageBox.Show("Problem in registratio..");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

**Web Application:**

DATA LAYER:

Authentication:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

namespace StudentWebApp.DataLayer

{

class Authentication:IAuthentication

{

IDataAccess \_idac = null;

public Authentication()

{

\_idac = new DataAccess();

}

public Authentication(IDataAccess idac)

{

\_idac = idac;

}

public bool VerifyLogin(string username, string password)

{

bool ret = false;

try

{

string sql = "select Username from Users where Username= @username and Password=@password";

List<SqlParameter> AList = new List<SqlParameter>();

DBHelper.AddSqlParam(AList, "@username", SqlDbType.VarChar, username, 30);

DBHelper.AddSqlParam(AList, "@password", SqlDbType.VarChar, password, 30);

object obj = \_idac.GetSingleAnswer(sql, AList);

if (obj != null)

ret = true;

}

catch (Exception)

{

throw;

}

return ret;

}

}

}

DataAccess:

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

class DataAccess: IDataAccess

{

public static string ConnectionString = ConfigurationManager.ConnectionStrings["STDB2017"].ConnectionString;

//Implementing Interface

public object GetSingleAnswer(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false)

{

SqlConnection conn = null;

if (sqtr == null)

conn = new SqlConnection(ConnectionString);

else

conn = connc;

object obj = null;

try

{

if (sqtr == null)

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

if (sqtr != null)

cmd.Transaction = sqtr;

if (IsStoredProc == true)

cmd.CommandType = CommandType.StoredProcedure;

if (PList != null)

{

foreach (SqlParameter param in PList)

cmd.Parameters.Add(param);

}

obj = cmd.ExecuteScalar();

}

catch (Exception)

{

throw; // send the error back to caller

}

finally

{

conn.Close();

}

return obj;

}

public DataTable GetManyRowsCols(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false)

{

SqlConnection conn = null;

if (sqtr == null)

conn = new SqlConnection(ConnectionString);

else

conn = connc;

DataTable dt = new DataTable();

try

{

if (sqtr == null)

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

if (sqtr != null)

cmd.Transaction = sqtr;

if (IsStoredProc == true)

cmd.CommandType = CommandType.StoredProcedure;

if (PList != null)

{

foreach (SqlParameter param in PList)

cmd.Parameters.Add(param);

}

SqlDataAdapter da = new SqlDataAdapter();

da.SelectCommand = cmd;

da.Fill(dt);

}

catch (Exception)

{

throw;

}

finally

{

if (sqtr == null)

conn.Close();

}

return dt;

}

public int InsertUpdateDelete(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false)

{

SqlConnection conn = null;

if (sqtr == null)

conn = new SqlConnection(ConnectionString);

else

conn = connc;

int rowsModified = 0;

try

{

if (sqtr == null)

conn.Open();

SqlCommand cmd = new SqlCommand(sql, conn);

if (sqtr != null)

cmd.Transaction = sqtr;

if (IsStoredProc == true)

cmd.CommandType = CommandType.StoredProcedure;

if (PList != null)

{

foreach (SqlParameter param in PList)

cmd.Parameters.Add(param);

}

rowsModified = cmd.ExecuteNonQuery();

}

catch (Exception)

{

throw;

}

finally

{

if (sqtr == null)

conn.Close();

}

return rowsModified;

}

}

}

DBHelper:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

class DBHelper

{

public static void AddSqlParam(List<SqlParameter> PList, string paramName,

SqlDbType paramType, object paramValue, int size=0)

{

SqlParameter p = null;

if (size == 0)

p = new SqlParameter(paramName, paramType);

else

p = new SqlParameter(paramName, paramType, size);

p.Value = paramValue;

PList.Add(p);

}

}

}

DBList:

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

class DBList

{

public static List<T> GetList<T>(DataTable dt)

where T : IEntity, new()

{ // will do conversion from dt to List<> for reference types

List<T> TList = new List<T>();

foreach (DataRow dr in dt.Rows)

{

T t1 = new T();

// populate the columns from dr into the fields of the t1

// delegate the work of populating t1 to the class T

t1.PopulateFields(dr);

TList.Add(t1);

}

return TList;

}

public static List<T> GetListValueType<T>(DataTable dt, string colname)

where T : IConvertible // will do conversion from dt to List<>for value types including string

{

List<T> TList = new List<T>();

foreach (DataRow dr in dt.Rows)

TList.Add((T)dr[colname]);

return TList;

}

}

}

IAuthentication:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

interface IAuthentication

{

bool VerifyLogin(string username, string password);

}

}

IDataAccess:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

interface IDataAccess

{

object GetSingleAnswer(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false);

DataTable GetManyRowsCols(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false);

int InsertUpdateDelete(string sql, List<SqlParameter> PList, SqlConnection connc = null, SqlTransaction sqtr = null, bool IsStoredProc = false);

}

}

IEntity:

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

interface IEntity

{ // any class that is ivolved in conversion from a DataTable will provide this method

void PopulateFields(DataRow dr);

}

}

IRepositoryCourses:

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

interface IRepositoryCourses

{

int GetEnrollmentCount(string semester, string courseNum);

List<string> GetSemesters();

List<CourseOfferedVM> GetCoursesOffered(string semester);

List<CourseEnrollmentVM> GetCourseEnrollment(string semester, string courseNum);

List<string> GetCoursesOfferedForASemester(string semester);

List<string> GetPreRequisiteCourses(string courseNum);

bool IsThereRoomInTheCourse(string semester, string courseNum);

}

}

IRepositoryInstructors:

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

interface IRepositoryInstructors

{

List<Instructors> GetInstructors();

}

}

IRepositoryStudents:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

interface IRepositoryStudents

{

bool DoesStudentExist(long studentId);

float? GetGradeForACourse(long studentId, string courseNum);

bool HasStudentTakenPreRequisteCourses(long studentId, string courseNum, float minGrade);

bool RegisterStudent(long studentId, string semester, string courseNum);

}

}

RepositoryCourses:

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

class RepositoryCourses: IRepositoryCourses // purpose of repository is to serve data

{ // if data is being obtained from a Database Server, then the repository

// assembles SQL and its parameters for a given task

IDataAccess \_idac = null;

public RepositoryCourses()

{

\_idac = new DataAccess();

}

public RepositoryCourses(IDataAccess idac)

{

\_idac = idac;

}

public int GetEnrollmentCount(string semester, string courseNum)

{

int enrollCount = 0;

try

{

string sql = "select count(\*) from CourseEnrollments where SemesterId=@SemesterId and " +

"CourseNum=@CourseNum";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

//object objCount = DataAccess.GetSingleAnswer(sql, PList);

object objCount = \_idac.GetSingleAnswer(sql, PList);

if (objCount != null)

enrollCount = int.Parse(objCount.ToString());

}

catch (Exception)

{

throw;

}

return enrollCount;

}

public List<string> GetSemesters()

{

List<string> SList = new List<string>();

try

{

string sql = "select SemesterId from Semesters";

//DataTable dt = DataAccess.GetManyRowsCols(sql, null);

DataTable dt = \_idac.GetManyRowsCols(sql, null);

// convert datatable to List<string>

SList = DBList.GetListValueType<string>(dt, "SemesterId");

//foreach (DataRow dr in dt.Rows)

//SList.Add(dr["SemesterId"].ToString());

}

catch (Exception)

{

throw;

}

return SList;

}

public List<CourseOfferedVM> GetCoursesOffered(string semester)

{

List<CourseOfferedVM> CList = new List<CourseOfferedVM>();

try

{

string sql = "select co.CourseNum, c.CourseName, c.Credits, co.Capacity," +

"i.LastName as Instructor from CoursesOffered co " +

"inner join Courses c on co.CourseNum=c.CourseNum " +

"inner join Instructors i on co.InstructorId=i.InstructorId where " +

"co.SemesterId=@SemesterId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

//DataTable dt = DataAccess.GetManyRowsCols(sql, PList);

DataTable dt = \_idac.GetManyRowsCols(sql, PList);

//convert data table to List<CourseOfferedVM> using MyWntityBase

//CList = DBList.GetList<CourseOfferedVM>(dt);

// convert data table to List<CourseOfferedVM>

foreach (DataRow dr in dt.Rows)

{

CourseOfferedVM covm = new CourseOfferedVM();

covm.CourseNum = dr["CourseNum"].ToString();

covm.CourseName = dr["CourseName"].ToString();

covm.Credits = (int)dr["Credits"];

covm.Capacity = (int)dr["Capacity"];

covm.Instructor = dr["Instructor"].ToString();

covm.Enrolled = GetEnrollmentCount(semester, covm.CourseNum);

CList.Add(covm);

}

}

catch (Exception)

{

throw;

}

return CList;

}

public List<CourseEnrollmentVM> GetCourseEnrollment(string semester, string courseNum)

{

List<CourseEnrollmentVM> CList = new List<CourseEnrollmentVM>();

try

{

string sql = "select s.StudentId, s.FirstName, s.LastName, s.Email, c.Credits," +

"d.DepartmentName as Major from Students s " +

"inner join CourseEnrollments ce on s.StudentId=ce.StudentId " +

"inner join Courses c on ce.CourseNum=c.CourseNum " +

"inner join StudentDepartments sd on s.StudentId=sd.StudentId " +

"inner join Departments d on sd.DepartmentId=d.DepartmentId " +

"where ce.SemesterId=@SemesterId and ce.CourseNum=@CourseNum";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

//DataTable dt = DataAccess.GetManyRowsCols(sql, PList);

DataTable dt = \_idac.GetManyRowsCols(sql, PList);

// convert datatable to List<CurseEnrollmentVM>

CList = DBList.GetList<CourseEnrollmentVM>(dt);

}

catch (Exception)

{

throw;

}

return CList;

}

public List<string> GetCoursesOfferedForASemester(string semester)

{

List<string> CList = new List<string>();

try

{

string sql = "select CourseNum from CoursesOffered where SemesterId=@SemesterId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

//DataTable dt = DataAccess.GetManyRowsCols(sql, PList);

DataTable dt = \_idac.GetManyRowsCols(sql, PList);

// convert dt to List<string>

//foreach (DataRow dr in dt.Rows)

// CList.Add(dr["CourseNum"].ToString());

CList = DBList.GetListValueType<string>(dt, "CourseNum");

}

catch (Exception)

{

throw;

}

return CList;

}

public List<string> GetPreRequisiteCourses(string courseNum)

{

List<string> CList = new List<string>();

try

{

string sql = "select PreReqCourseNum from CoursePreRequisites where CourseNum=@CourseNum";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

//DataTable dt = DataLayer.DataAccess.GetManyRowsCols(sql, PList);

DataTable dt = \_idac.GetManyRowsCols(sql, PList);

CList = DBList.GetListValueType<string>(dt, "PreReqCourseNum");

}

catch (Exception)

{

throw;

}

return CList;

}

public bool IsThereRoomInTheCourse(string semester, string courseNum)

{

bool ret = false;

try

{

string sql = "select Capacity from CoursesOffered where CourseNum=@CourseNum and " +

"SemesterId=@SemesterId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

//object objCap = DataLayer.DataAccess.GetSingleAnswer(sql, PList);

object objCap = \_idac.GetSingleAnswer(sql, PList);

if (objCap != null)

{

int capacity = int.Parse(objCap.ToString());

int enrollCount = GetEnrollmentCount(semester, courseNum);

if (enrollCount < capacity)

ret = true;

}

}

catch (Exception)

{

throw;

}

return ret;

}

}

}

RepositoryInstructors:

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Web;

namespace StudentWebApp.DataLayer

{

class RepositoryInstructors : IRepositoryInstructors

{

IDataAccess \_idac = null;

public RepositoryInstructors()

{

\_idac = new DataAccess();

}

public RepositoryInstructors(DataAccess idac)

{

\_idac = idac;

}

public List<Instructors> GetInstructors()

{

List<Instructors> IList = new List<Instructors>();

try

{

string sql = "select i.FirstName, i.LastName, i.Telephone, d.DepartmentName as Department from Instructors i " +

"inner join Departments d on i.DepartmentId=d.DepartmentID";

DataTable dt = \_idac.GetManyRowsCols(sql, null);

IList = DBList.GetList<Instructors>(dt);

}

catch (Exception)

{

throw;

}

return IList;

}

}

}

RepositoryStudents:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.DataLayer

{

class RepositoryStudents: IRepositoryStudents

{

IDataAccess \_idac = null;

public RepositoryStudents()

{

\_idac = new DataAccess();

}

public RepositoryStudents(IDataAccess idac)

{

\_idac = idac;

}

public bool DoesStudentExist(long studentId)

{

bool ret = false;

try

{

string sql = "select StudentId from Students where StudentId=@StudentId";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@StudentId", SqlDbType.BigInt, studentId);

object objId = \_idac.GetSingleAnswer(sql, PList);

if (objId != null)

ret = true;

}

catch (Exception)

{

throw;

}

return ret;

}

public float? GetGradeForACourse(long studentId, string courseNum)

{

float? grade = null;

try

{

string sql = "select grade from CoursesCompleted where " +

"CourseNum=@CourseNum and StudentId=@StudentId ";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@StudentId", SqlDbType.BigInt, studentId);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

object objGrade = \_idac.GetSingleAnswer(sql, PList);

if (objGrade != null)

grade = float.Parse(objGrade.ToString());

}

catch (Exception)

{

throw;

}

return grade;

}

public bool HasStudentTakenPreRequisteCourses(long studentId, string courseNum, float minGrade)

{

bool ret = true;

try

{

//RepositoryCourses repCourses = new RepositoryCourses();

IRepositoryCourses repCourses = new RepositoryCourses();

List<string> PreReqList = repCourses.GetPreRequisiteCourses(courseNum);

// determine the grade and see if each course is completed with minGrade or better

foreach (string cnum in PreReqList)

{

float? grade = GetGradeForACourse(studentId, cnum);

if ((grade == null) || (grade < minGrade))

{

ret = false;

break;

}

}

}

catch (Exception)

{

throw;

}

return ret;

}

public bool RegisterStudent(long studentId, string semester, string courseNum)

{

bool ret = false;

try

{

string sql = "Insert into CourseEnrollments(StudentId,SemesterId,CourseNum) values" +

"(@StudentId,@SemesterId,@CourseNum)";

List<SqlParameter> PList = new List<SqlParameter>();

DBHelper.AddSqlParam(PList, "@StudentId", SqlDbType.BigInt, studentId);

DBHelper.AddSqlParam(PList, "@SemesterId", SqlDbType.VarChar, semester, 20);

DBHelper.AddSqlParam(PList, "@CourseNum", SqlDbType.VarChar, courseNum, 50);

int rows = \_idac.InsertUpdateDelete(sql, PList);

if (rows > 0)

ret = true;

}

catch (Exception)

{

throw;

}

return ret;

}

}

}

BUSINESS LAYER:

BusinessAuthentication:

using StudentWebApp.DataLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace StudentWebApp.BusinessLayer

{

class BusinessAuthentication : IBusinessAuthentication

{

IAuthentication \_iauth = null;

public BusinessAuthentication()

{

\_iauth = new Authentication();

}

public bool VerifyLogin(string username, string password)

{

return \_iauth.VerifyLogin(username, password);

}

}

}

BusinessCourses:

using StudentWebApp.DataLayer;

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.BusinessLayer

{

class BusinessCourses : IBusinessCourses

{

//RepositoryCourses \_rep = new RepositoryCourses();

IRepositoryCourses \_irc = null;

public BusinessCourses()

{

\_irc = new RepositoryCourses();

}

public BusinessCourses(IRepositoryCourses irc)

{

\_irc = irc;

}

// the UI layer always communicates with the business layer

// even if no code is needed, the business layer forwards the request to repository in data layer

public List<string> GetSemesters()

{

return \_irc.GetSemesters();

}

public List<CourseOfferedVM> GetCoursesOffered(string semester)

{

return \_irc.GetCoursesOffered(semester);

}

public List<CourseEnrollmentVM> GetCourseEnrollment(string semester, string courseNum)

{

return \_irc.GetCourseEnrollment(semester, courseNum);

}

public List<string> GetCoursesOfferedForASemester(string semester)

{

return \_irc.GetCoursesOfferedForASemester(semester);

}

}

}

BusinessInstructors:

using StudentWebApp.DataLayer;

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace StudentWebApp.BusinessLayer

{

class BusinessInstructors:IBusinessInstructors

{

IRepositoryInstructors \_iri = null;

public BusinessInstructors()

{

\_iri = new RepositoryInstructors();

}

public BusinessInstructors(RepositoryInstructors iri)

{

\_iri = iri;

}

public List<Instructors> GetInstructors()

{

return \_iri.GetInstructors();

}

}

}

BusinessStudents:

using StudentWebApp.DataLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.BusinessLayer

{

class BusinessStudents : IBusinessStudents

{

/\*

RepositoryCourses repCourses = new RepositoryCourses();

RepositoryStudents repStudents = new RepositoryStudents();

\*/

IRepositoryCourses \_irc = null;

IRepositoryStudents \_irs = null;

public BusinessStudents()

{

\_irc = new RepositoryCourses();

\_irs = new RepositoryStudents();

}

public BusinessStudents(IRepositoryCourses irc, IRepositoryStudents irs)

{

\_irc = irc;

\_irs = irs;

}

public BusinessStudents(IRepositoryCourses irc)

{

\_irc = irc;

\_irs = new RepositoryStudents();

}

public BusinessStudents(IRepositoryStudents irs)

{

\_irc = new RepositoryCourses();

\_irs = irs;

}

public bool RegisterStudent(long studentId, string semester, string courseNum)

{

bool ret = false;

try

{

if (\_irs.DoesStudentExist(studentId))

{

if (\_irs.HasStudentTakenPreRequisteCourses(studentId, courseNum, 2.0f))

{

if (\_irc.IsThereRoomInTheCourse(semester, courseNum))

ret = \_irs.RegisterStudent(studentId, semester, courseNum);

else

throw new Exception("Course capacity exceeded..");

}

else

throw new Exception("Missing prerequisites for " + courseNum);

}

else

throw new Exception("Invalid student Id..");

}

catch (Exception)

{

throw;

}

return ret;

}

}

}

IBusinessAuthentication:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.BusinessLayer

{

interface IBusinessAuthentication

{

bool VerifyLogin(string username, string password);

}

}

IBusinessCourses:

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.BusinessLayer

{

interface IBusinessCourses

{

List<string> GetSemesters();

List<CourseOfferedVM> GetCoursesOffered(string semester);

List<CourseEnrollmentVM> GetCourseEnrollment(string semester, string courseNum);

List<string> GetCoursesOfferedForASemester(string semester);

}

}

IBusinessInstructors

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.BusinessLayer

{

interface IBusinessInstructors

{

List<Instructors> GetInstructors();

}

}

IBusinessStudents:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.BusinessLayer

{

interface IBusinessStudents

{

bool RegisterStudent(long studentId, string semester, string courseNum);

}

}

MODELS:

CoursesEnrollmentVM:

using StudentWebApp.DataLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

namespace StudentWebApp.Models

{

class CourseEnrollmentVM : MyEntityBase//IEntity

{

public long StudentId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Major { get; set; }

public string Email { get; set; }

public int Credits { get; set; }

}

}

CoursesOfferedVM:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace StudentWebApp.Models

{

class CourseOfferedVM : MyEntityBase

{

public string CourseNum { get; set; }

public string CourseName { get; set; }

public int Credits { get; set; }

public string Instructor { get; set; }

public int Capacity { get; set; }

public int Enrolled { get; set; }

}

}

Instructors:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace StudentWebApp.Models

{

class Instructors : MyEntityBase

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string Telephone { get; set; }

public string Department { get; set; }

}

}

MyEntityBase:

using StudentWebApp.DataLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Reflection;

namespace StudentWebApp.Models

{

class MyEntityBase : IEntity

{

public void PopulateFields(DataRow dr)

{

// use reflection to populate the fields of this class from DataRow

Type tp = this.GetType();

foreach (PropertyInfo pi in tp.GetProperties())

pi.SetValue(this, dr[pi.Name]);

}

}

}

WEB FORMS:

CoursesEnrollment:

using System;

using System.Collections.Generic;

using StudentWebApp.Models;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using StudentWebApp.BusinessLayer;

namespace StudentWebApp

{

public partial class CourseEnrollment1 : System.Web.UI.Page

{

//BusinessLayer.BusinessCourses bCourses = new BusinessLayer.BusinessCourses();

IBusinessCourses ibCourses = new BusinessCourses();

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["LOGGEDIN"] == null)

Response.Redirect("Login");

else

{

try

{

if (!IsPostBack)

{

List<string> SList = ibCourses.GetSemesters();

semesterOptions.DataSource = SList;

semesterOptions.DataBind();

}

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

protected void semesterOptions\_SelectedIndexChanged(object sender, EventArgs e)

{

string choice = semesterOptions.Text;

if (choice.IndexOf("DataRowView") < 0)

{

try

{

List<string> CList = ibCourses.GetCoursesOfferedForASemester(choice);

coursesList.DataSource = CList;

coursesList.DataBind();

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

protected void coursesList\_SelectedIndexChanged(object sender, EventArgs e)

{

string semester = semesterOptions.Text;

string course = coursesList.Text;

if (course.IndexOf("DataRowView") < 0)

{

try

{

List<CourseEnrollmentVM> CList = ibCourses.GetCourseEnrollment(semester, course);

gv1.DataSource = CList;

gv1.DataBind();

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

}

}

Login:

using StudentWebApp.BusinessLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace StudentWebApp

{

public partial class Login : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

try

{

IBusinessAuthentication iba = new BusinessAuthentication();

bool ret = iba.VerifyLogin(txtUser.Text, txtPassword.Text);

if (ret == true)

{

Session["LOGGEDIN"] = true;

Response.Redirect("StartMenu");

}

else

if(IsPostBack)

lblStatus.Text = "Invalid Login";

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

}

RegisterForCourses:

using StudentWebApp.BusinessLayer;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace StudentWebApp

{

public partial class RegisterForCourses : System.Web.UI.Page

{

//BusinessLayer.BusinessCourses \_busCourses = new BusinessLayer.BusinessCourses();

//BusinessLayer.BusinessStudents \_busStudents = new BusinessLayer.BusinessStudents();

IBusinessCourses ibCourses = new BusinessCourses();

IBusinessStudents ibStudents = new BusinessStudents();

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["LOGGEDIN"] == null)

Response.Redirect("Login");

else

{

if (!IsPostBack)

{

try

{

List<string> SemesterList = ibCourses.GetSemesters();

ddSemester.DataSource = SemesterList;

ddSemester.DataBind();

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

}

protected void ddSemester\_SelectedIndexChanged(object sender, EventArgs e)

{

//We need to populate the courses drop down with the semester selected

string semester = ddSemester.Text;

if (semester.IndexOf("DataRowView") < 0)

{

try

{

List<string> CoursesList = ibCourses.GetCoursesOfferedForASemester(semester);

ddCourse.DataSource = CoursesList;

ddCourse.DataBind();

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

protected void btnRegister\_Click(object sender, EventArgs e)

{

string semester = ddSemester.Text;

string course = ddCourse.Text;

try

{

bool accepted = ibStudents.RegisterStudent(long.Parse(txtStudentID.Text), semester, course);

if (accepted == true)

lblStatus.Text = "Student registered in " + course;

else

lblStatus.Text = "Registration Failed...";

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

}

ShowCourses:

using System;

using System.Collections.Generic;

using StudentWebApp.Models;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using StudentWebApp.BusinessLayer;

namespace StudentWebApp

{

public partial class CourseEnrollment : System.Web.UI.Page

{

//BusinessLayer.BusinessCourses bCourses = new BusinessLayer.BusinessCourses();

IBusinessCourses ibCourses = new BusinessCourses();

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["LOGGEDIN"] == null)

Response.Redirect("Login");

else

{

try

{

if (!IsPostBack)

{

List<string> SList = ibCourses.GetSemesters();

DropDownSemesters.DataSource = SList;

DropDownSemesters.DataBind();

}

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

protected void DropDownSemesters\_SelectedIndexChanged(object sender, EventArgs e)

{

string semester = DropDownSemesters.SelectedItem.Text;

if (semester.IndexOf("DataRowView") < 0)

{

try

{

List<CourseOfferedVM> CList = ibCourses.GetCoursesOffered(semester);

dgCourses.DataSource = CList;

dgCourses.DataBind();

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

}

}

ShowInstructors:

using StudentWebApp.DataLayer;

using StudentWebApp.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace StudentWebApp

{

public partial class ShowInstructors : System.Web.UI.Page

{

IRepositoryInstructors irep = new RepositoryInstructors();

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["LOGGEDIN"] == null)

Response.Redirect("Login");

else

{

try

{

List<Instructors> CList = irep.GetInstructors();

gv1.DataSource = CList;

gv1.DataBind();

}

catch (Exception ex)

{

lblStatus.Text = ex.Message;

}

}

}

}

}

StartMenu:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace StudentWebApp

{

public partial class StartMenu : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["LOGGEDIN"] == null)

Response.Redirect("Login");

}

protected void btnCourses\_Click(object sender, EventArgs e)

{

Response.Redirect("ShowCourses.aspx");

}

protected void btnEnrollment\_Click(object sender, EventArgs e)

{

Response.Redirect("CourseEnrollment.aspx");

}

protected void btnRegistration\_Click(object sender, EventArgs e)

{

Response.Redirect("RegisterForCourses.aspx");

}

protected void btnInstructors\_Click(object sender, EventArgs e)

{

Response.Redirect("ShowInstructors.aspx");

}

}

}

**Conclusion:**

After doing this assignment I was able to understand better how applications work. I understood the need of each element and how to connect the different contents of a web app. I now know that in order to have a good architecture for our Web App, we need to divide it in different layers such as Data Layer, Business Layer and the WebForm in order to display. Then, inside the Data Layer we need a Data Access in order to communicate with the DB and Repositories which will connect the business layers with the Data Layer. Also, we need a Models folder which will contain different classes in order to create an object with the data selected for future uses (display, validations etc.). As mentioned before, the relationship between different layers needs to be loose coupling in order to avoid difficult changes in our application in the future (in case we want to obtain data using Oracle, do different actions with the data etc.). Also, I got to learn how sessions work and how to protect the different subpages of an application if the user is not logged in (by creating session objects).