Tutorial 17

Programming exercise 1 – ConnectionString and DbManager

1. Make sure that WIUT.DAL project is selected in Solution Explorer. Go to Project => WIUT.DAL properties.
2. Select Settings tab, add new setting:
   1. Name: ConnectionString
   2. Type: (ConnectionString)
   3. Value: Data Source=WIUT.sdf
3. In WIUT.DAL project add new class. Name it DbManager and make it public

public class DbManager

1. Create a protected read-only property that will return a connection with correct connection string.

protected SqlCeConnection Connection

{

get

{

return new SqlCeConnection(Properties.Settings.Default.ConnectionString);

}

}

1. Update ApplicantManager and CourseManager to inherit from DbManager and update the methods with empty connection strings to acquire connection from the parent class property

var connection = Connection;

Programming exercise 2 – Collections

1. Add new class CourseList, make it public.
2. Add public function GetAllCourses() that will call corresponding function of manager class and return all courses stored in the database

public List<Course> GetAllCourses()

{

return new CourseManager().GetAll();

}

1. Do same for ApplicantList class

Programming exercise 3 – Updating settings

1. Create FormLoad handler for parent form
2. Find out how many courses we have by calling GetAllCourses() function of CourseList class

MessageBox.Show(new CourseList().GetAllCourses().Count.ToString());

1. Run the project and see the result
2. Rename WIUT.sdf to some other name and run the project again
3. Open app.config file in WIUT.DAL project and find the following section:

<connectionStrings>

<add name="WIUT.DAL.My.MySettings.ConnectionString" connectionString="Data Source=&quot;…&quot;"

providerName="Microsoft.SqlServerCe.Client.4.0" />

</connectionStrings>

1. Open app.config file in WIUT project. Replace its connectionStrings section with the one taken from WIUT.DAL’s app.config
2. Run the project to make all the files update.
3. Go to C:\...\WIUT\bin\Debug\ and check “WIUT.exe.config” file (general rule is {name of exe file}.config). Change the name of sdf file in file system and in config, see the result.

Programming exercise 4 – ByAttribute enumeration

1. Create new class in WIUT.DAL project. Name it ByAttribute
2. Create public enumeration, call it ByAttribute and add the following members
   1. Name
   2. Surname
   3. DoB
   4. Course

Programming exercise 5 – Sorting

1. Create Sort function in ApplicantList class to enable sorting using IComparer. Use ByAttribute enumeration for selection of sort options

public List<Applicant> Sort(ByAttribute attribute)

{

switch (attribute)

{

case ByAttribute.Name:

break;

case ByAttribute.Surname:

break;

case ByAttribute.DoB:

break;

case ByAttribute.Course:

break;

}

//if we are here - something went wrong

return null;

}

1. Add private class ByNameComparer inside ApplicantList class:

private class ByNameComparer : IComparer<Applicant>

{

public int Compare(Applicant x, Applicant y)

{

return string.Compare(x.Name, y.Name);

}

}

1. Populate corresponding clause in Sort function:

case ByAttribute.Name:

var result = GetAllApplicants();

result.Sort(new ByNameComparer());

return result;

1. Create corresponding comparers to fill in the remaining clauses

Programming exercise 6 – Sorting via LINQ

1. Create SortLinq function in ApplicantList class to do sorting using LINQ to objects. Compare the results.

public List<Applicant> SortLinq(ByAttribute attribute)

{

switch (attribute)

{

case ByAttribute.Name:

return GetAllApplicants().OrderBy(a => a.Name).ToList();

case ByAttribute.Surname:

var result = from a in GetAllApplicants() orderby a.Surname select a;

return result.ToList();

case ByAttribute.DoB:

break;

case ByAttribute.Course:

break;

}

//if we are here - something went wrong

return null;

}

1. Fill in the code for all clauses

Programming exercise 7 – Searching

Create Search function in ApplicantList class. Use ByAttribute enumeration to specify the attribute to search.

public List<Applicant> Search(string value, ByAttribute attribute)

{

switch (attribute)

{

case ByAttribute.Name:

return GetAllApplicants().Where(a => a.Name.Contains(value)).ToList();

case ByAttribute.Surname:

return GetAllApplicants().Where(a => a.Surname.Contains(value)).ToList();

}

//if we are here - something went wrong

return null;

}

Home work

Do research on Linked lists