Database Communication in Visual Studio/C# using 3-tier Architecture

The examples uses ADO.NET

Hans-Petter Halvorsen, M.Sc.
The database-centric style. Typically, the clients communicate directly with the database.

A three-tier style, in which clients do not connect directly to the database.

Web Services, etc.
3-tier/layer Architecture

Note! The different layers can be on the same computer (Logic Layers) or on different Computers in a network (Physical Layers)

Presentation Tier

Business Logic Tier

Data Access Tier

Data Tier - DL

Logic Tier

Data Source

PL

BL

DAL
Why 3-Tier (N-Tier Architecture?)

• Flexible applications
• Reusable code
  – Code once, use many times
• Modularized
  – You need only to change part of the code
  – You can deploy only one part
  – You can Test only one part
  – Multiple Developers
• Different parts (Tiers) can be stored on different computers
• Different Platforms and Languages can be used
• etc.
Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

http://en.wikipedia.org/wiki/Multitier_architecture
3-tier/layer Architecture

Presentation Tier

- This is the topmost level of the application.
- The presentation tier displays information related to such services as browsing merchandise, purchasing and shopping cart contents.
- It communicates with other tiers by which it puts out the results to the browser/client tier and all other tiers in the network.
- In simple terms it is a layer which users can access directly such as a web page, or an operating systems GUI

Application tier (business logic, logic tier, data access tier, or middle tier)

- The logical tier is pulled out from the presentation tier and, as its own layer.
- It controls an application’s functionality by performing detailed processing.

Data tier

- This tier consists of database servers. Here information is stored and retrieved.
- This tier keeps data neutral and independent from application servers or business logic.
- Giving data its own tier also improves scalability and performance.

http://en.wikipedia.org/wiki/Multitier_architecture
3-tier Architecture

Different Devices can share the same Business and Data Access Code

The different Tiers can be physical or logical
3-tier + WebService Architecture - Example

Team Foundation Server

Installed on one or more Windows Servers in your LAN or in the Cloud

Web Server

WebService

Architecture

Example

Presentation Tier

TFS Client

Team Foundation Server

Web Services

Business/Data Logic Tier

Stored Procedures

Data Source

Data Tier

Visual Studio
Creating the Data Tier

Hans-Petter Halvorsen, M.Sc.
Data Tier

We are going to create the Database / Data Layer/Tier, including:

1. Tables
2. Views
3. Stored Procedures
4. Triggers
5. Script for some “Dummy” Data

Download Zip Files with Tables, Views, Stored Procedures and Triggerse in order to create the Data Tier in SQL Server (The ZIP File is located on the same place as this File)

Note! Install them in this order
Data Tier

- Triggers
- Stored Procedures
- Views
- Tables

SQL Server
Execute the different Scripts inside SQL Server Management Studio
You are finished with the Exercise
Creating the Logic Tier

Hans-Petter Halvorsen, M.Sc.
Purpose:
• All the Apps should/could share the same Logic Tier
• To make your Apps easier to maintain and extend
• etc.
Create an Empty (Blank) Solution in Visual Studio

New Project

- **Recent**
  - .NET Framework 4.5
  - Sort by: Default

- **Installed**
  - Blank Solution
  - Blank Solution

- **Templates**
  - Visual C#
    - Windows Store
    - Windows
  - Web
    - Visual Studio 2012
    - Cloud
    - Reporting
    - Silverlight
    - Test
    - WCF
    - Workflow
    - TypeScript
  - Other Languages
  - Other Project Types
    - Setup and Deployment
    - Visual Studio Solutions
  - Samples

- **Online**
  - Click here to go online and find templates.

- **Name:** MySoftware
- **Location:** C:\Work\Development\TFS\Development\ tier Architecture\ 
- **Solution name:** MySoftware

- Create directory for solution
- Add to source control

**OK** | **Cancel**
Add **Project** for Logic Tier (Data Access)

Select a “**Class Library**” Project

```
LogicTier
```
Add a New **Class** to the Project (“StudentData.cs”)
Create the **Code**, e.g., like this (“StudentData.cs”):

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Data;

namespace Tuc.School.LogicTier
{
    public class StudentData
    {
        public DataSet GetStudentDB(string connectionString)
        {
            string selectSQL = "select StudentName, StudentNumber, SchoolName, ClassName, Grade from StudentData order by StudentName";

            // Define the ADO.NET objects.
            SqlConnection con = new SqlConnection(connectionString);
            SqlDataAdapter da = new SqlDataAdapter(selectSQL, con);
            DataSet ds = new DataSet();
            da.Fill(ds);
            return ds;
        }
    }
}
```

**Create your own Namespace**

**A View that collects data from several tables**

**Improvements: Use Try... Catch ...**
You should test the SQL Query in the SQL Server Management Studio first.

```
SELECT StudentName, StudentNumber, SchoolName, ClassName, Grade
FROM StudentData
ORDER BY StudentName;
```
```csharp
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Data;

namespace Tuc.School.LogicTier
{
    public class StudentData
    {
        public DataSet GetStudentDB(string connectionString)
        {
            string selectSQL = "select StudentName, StudentNumber, SchoolName, ClassName, Grade from StudentData order by StudentName";

            // Define the ADO.NET objects.
            SqlConnection con = new SqlConnection(connectionString);

            SqlDataAdapter da = new SqlDataAdapter(selectSQL, con);

            DataSet ds = new DataSet();
            da.Fill(ds);

            return ds;
        }
    }
}
```
Create a proper name for the **Assembly** (.dll File)

Right-click on the Project in the Solution Explorer and select Properties

- Enter the name **Tuc.School.LogicTier** in the Assembly name field.
- Ensure the default namespace also reads **Tuc.School.LogicTier**.

Then Build your Project (hopefully with no errors)

This will be the Assembly for your Logic Tier, that can be imported and used in other projects. Create once – use it many times!!
You are finished with the Exercise
Creating the Presentation Tier

Hans-Petter Halvorsen, M.Sc.
Presentation Layer
Desktop App: WinForms

We assume the App will be used only in the local LAN (or local on the same computer where the database is located) and that we have direct access to the Database

![Image of a WinForms application with a label and a DataGridView control]
Add a **WinForm Project**
Add a New Class (“StudentWinForm.cs”)
Add Code in Class

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using Tuc.School.LogicTier;

namespace Tuc.School.WinFormApp
{
    public class StudentWinForm
    {
        public DataSet GetStudent(string connectionString)
        {
            StudentData studentData = new StudentData();
            return studentData.GetStudentDB(connectionString);
        }
    }
}
```

Add a Reference to the Assembly in the Logic Tier
using System.Data; Since we are using the DataSet Class
using Tuc.School.LogicTier; Reference to our Logic Tier
namespace Tuc.School.WinFormApp
{
    class StudentWinForm
    {
        public DataSet GetStudent(string connectionString)
        {
            StudentData studentData = new StudentData();
            return studentData.GetStudentDB(connectionString);
        }
    }
}

Our Database Method in our Logic Tier
Create Form

Student Information

DataGridView
Create Form Code

```csharp
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;
using System.Configuration;
using Tuc.School.WinFormsApp;
	namespace WinFormApp
{
    public partial class Form1 : Form
    {
        private string connectionString = ConfigurationManager.ConnectionStrings["SCHOOLConnectionString"].ConnectionString;
        
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            FillStudentGrid();
        }

        private void FillStudentGrid()
        {
            DataSet ds = new DataSet();
            StudentWinForm studentList = new StudentWinForm();
            ds = studentList.GetStudent(connectionString);
            dataGridViewStudentInformation.DataSource = ds.Tables[0];
        }
    }
}
```
using System.Configuration;
using Tuc.School.WinFormApp;

namespace WinFormApp
{
    public partial class Form1 : Form
    {
        private string connectionString =
            ConfigurationManager.ConnectionStrings["SCHOOLConnectionString"].ConnectionString;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            FillStudentGrid();
        }

        private void FillStudentGrid()
        {
            DataSet ds = new DataSet();

            StudentWinForm studentList = new StudentWinForm();

            ds = studentList.GetStudent(connectionString);

            dataGridViewStudentInformation.DataSource = ds.Tables[0];
        }
    }
}
Note! Add `System.Configuration` Reference
Create DB ConnectionString in App.config

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>

  <connectionStrings>
    <add name="SCHOOLConnectionString" connectionString="Data Source=macwin8;Initial Catalog=SCHOOL;Persist Security Info=True;User ID=sa;Password=xxxxxx" providerName="System.Data.SqlClient" />
  </connectionStrings>

</configuration>
```
Test it

It works!!!
You are finished with the Exercise
Recommended Literature

• Tutorial: Introduction to Database Systems
  http://home.hit.no/~hansha/?tutorial=database

• Tutorial: Structured Query Language (SQL)
  http://home.hit.no/~hansha/?tutorial=sql

• Tutorial: Using SQL Server in C#

• Tutorial: Introduction to Visual Studio and C#
  http://home.hit.no/~hansha/?tutorial=csharp
Hans-Petter Halvorsen, M.Sc.
Telemark University College
Faculty of Technology
Department of Electrical Engineering, Information Technology and Cybernetics

E-mail: hans.p.halvorsen@hit.no
Blog: http://home.hit.no/~hansha/