

Articles

by Nagaraj

<http://nbende.wordpress.com>

Article of the week: Working on n-tier architecture and its features

N-tier application helps us distribute the overall functionality into various tiers or layers.

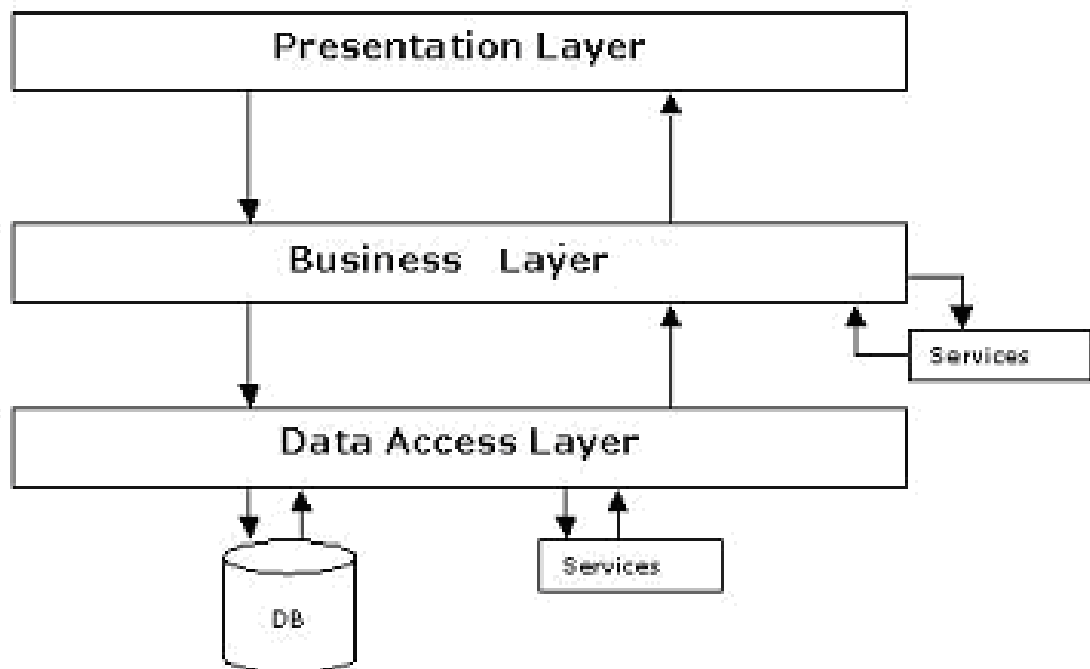
They are as follows:

- 1) Presentation Layer
- 2) Business Layer
- 3) Data Access Layer
- 4) Database/Data store

In certain scenarios some of the layers mentioned above may be split further into one or more sub layers.

Each Layer can be developed independently of the other provided that it adheres to the standards and communicates with the other layers as per the specifications.

The Logical Building Blocks



These layers are described below:

1) The Presentation Layer:

It is also called as the client layer or Presentation Layer or Graphical User Interface (GUI) layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

2) The Business Layer:

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business layer can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others.

3) The Data Access Layer:

This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not effect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer. The database is not accessed directly from any other layer/component. Hence the table names, field names are not hard coded anywhere else. This layer may also access any other services that may provide it with data, for instance Active Directory, Services etc. Having this layer also provides an additional layer of security for the database. As the other layers do not need to know the database credentials, connect strings and so on.

4) The Database Layer:

This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc.

In an n-tier application, the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

Many packaged n-tier Applications are created so that they can work the same with SQL Server, Oracle, UDB and so on.

FEATURES:

The N-tier Applications provide specific advantages that are vital to the business continuity of the enterprise. Typical features of a real life n-tier may include the following:

1) Security:

Application has Appropriate Authentication, logging and monitoring mechanisms

2) Availability and Scalability:

Application should be reliable and should have sufficient fail-over mechanisms (redundancy) such as fail-over clusters

3) Manageability:

Application should be designed to have the capability to Deploy, Monitor and troubleshoot. Methodology to handle Errors, log Errors and provide useful information for problem resolution

4) Easy Maintenance:

This is generally achieved by adopting coding standards, deployment standards, modular application design, 'data abstraction' and application frameworks.

5) Data Abstraction:

Easily make changes to the functionality, business rules with the least amount of impact to the entire applications.

Implementing n-tier architecture in Practical: Steps for creating:

- 1) Click on start button-> programs->Microsoft Visual Studio2005-> Microsoft Visual Studio2005.
- 2) Start Page will appear then click on create Projects->projects Types select any language like Visual C# -> Select a template like Windows Application ->ok. By default a form will appear.
- 3) Design the form.
- 4) Double click on add button and write the following code:

```
private void button1_Click(object sender, EventArgs e)
{
    blUsers obj = new blUsers();
    obj.Username = textBox1.Text;
    obj.Password1 = textBox2.Text;
    obj.Role = textBox3.Text;
    obj.adddata();
    MessageBox.Show("rec inserted succ");
}
```

```
}
```

5)Steps for Business Layer:

- a)Go to solution explorer (Ctrl+Alt+L).
- b)Right click on windows Application -> Add -> new item-> blUsers.cs
- c)Write the code for business layer.

Code for Business layer:

```
namespace WindowsApplication1
{
    class blUsers
    {
        String username, Password, role;
        public String Username
        {
            get { return username; }
            set { username = value; }
        }
        public String Password1
        {
            get { return Password; }
            set { Password = value; }
        }
        public String Role
        {
            get { return role; }
            set { role = value; }
        }
        public void adddata()
        {
            DALUsers.adddata(username, Password, role);
        }
    }
}
```

6)Steps for Data Access Layer:

- a) Go to solution explorer (Ctrl+Alt+L).
- b) Right click on windows Application -> Add -> new item->DALUser.cs
- c)Write the code for Data Access layer.

Code for Data Access Layer:

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

```

using System.Data;

namespace WindowsApplication1
{
    class DALUsers
    {
        static SqlConnection cn;
        static SqlCommand cmd;
    public static void adddata(string a, string b, string c)
    {
        SqlParameter parm1;
        cn = new SqlConnection("user id=sa;database=useraccess;data source=server");
        cn.Open();
        cmd = new SqlCommand("spInsertData", cn);
        cmd.CommandType = CommandType.StoredProcedure;
        parm1 = new SqlParameter("@username", SqlDbType.VarChar, 30);
        parm1.Value = a;
        cmd.Parameters.Add(parm1);
        parm1 = new SqlParameter("@password", SqlDbType.VarChar, 30);
        parm1.Value = b;
        cmd.Parameters.Add(parm1);
        parm1 = new SqlParameter("@role", SqlDbType.VarChar, 30);
        parm1.Value = c;
        cmd.Parameters.Add(parm1);
        cmd.ExecuteNonQuery();
    }
}
}

```

Nagaraj NES

7) Creating DATABASE: Create Table:

Username	Password	Role
Kamala	Star111	Manager
Raju	Fivestar	Programmer

Procedure for Insert:

```

create PROCEDURE dbo.spInsertData
(
    @username varchar(30),
    @password varchar(30),
    @role varchar(30)
)

```

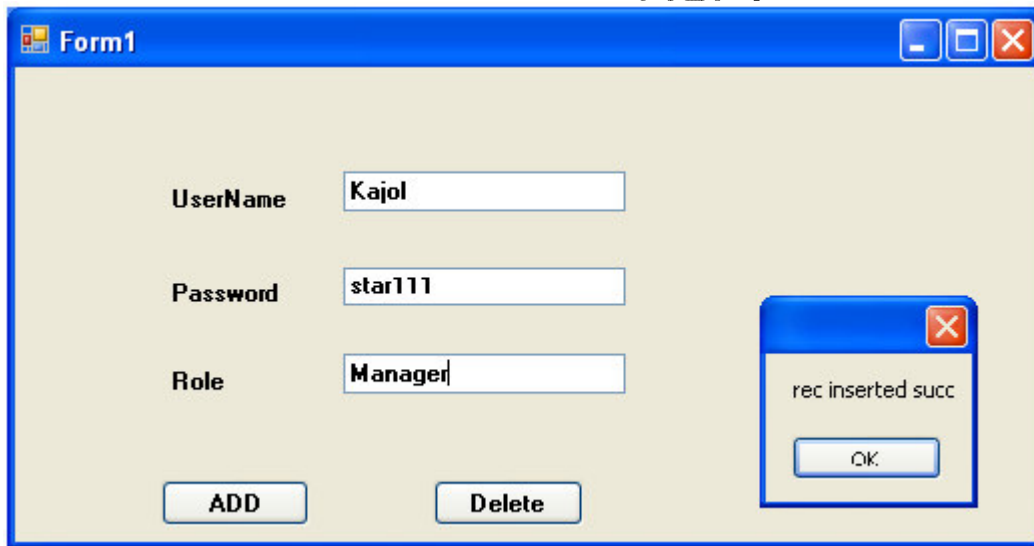
```
)  
AS  
insert into users values(@username,@password,@role)  
RETURN
```

Procedure for Delete:

```
Create PROCEDURE dbo.spDelete  
(@username varchar(50))  
AS  
delete from users where username =@username  
RETURN
```

8) Run the entire project.

Output:



Nagaraj.NET