From Data Handling to Presentation of Data: Encapsulating the App.config in .NET Applications through Design Patterns

Swapna S Kolhatkar
Assistant Professor,
MMCOE,
Karvenagar, Pune 411052

ABSTRACT
In todays world, most of the business has become dynamic and design patterns provide the necessary solutions to recurring design problems. The .NET environment provides for quick application building as well as services. This article is about studying the .NET based application development and the implementation of the design patterns.

The article further shows how certain database related processes can be considered as regular patterns in order to simplify the application development and project management and concentrate more on logic and presentation layer than the actual handling of data. We look into the possibility of developing a service for processes which can be treated as routine. The overall discussion is divided into 3 parts: - introduction to design patterns and the .NET environment. Second part discusses the common problems of data handling and data operations. Next, we see the possibility of a pattern or service to manage any type of data for any data files and lastly, the conclusions, keeping in mind the advantages and disadvantages of such an application.

General Terms
This article is concerned with the usage of few design patterns to understand the application development in .NET applications.

Keywords
Design Patterns, .NET Applications, app.config, application settings, inserting values in database, service, business logic.

1. INTRODUCTION
1.1 Design Patterns
Design Patterns are commonly defined as time-tested solutions to recurring design problems [Ref 1, 2]. There are 23 design patterns proposed by Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides (collectively known as the Gang of Four, or GoF for short. The present Software Engineering scenario has evolved and now got introduced to many patterns like .Net patterns, Sun's J2EE patterns, game design patterns, architectural patterns, JSP patterns, VB patterns, implementation patterns, analysis patterns, .Net patterns etc which discuss problems and their solutions and thus benefit the process of software design and development.

1.2 .NET Environment
To understand the .NET Environment [Ref 4] according to our topic of interest, let us consider the following points ie from 1 to 4 as shown in fig 1.

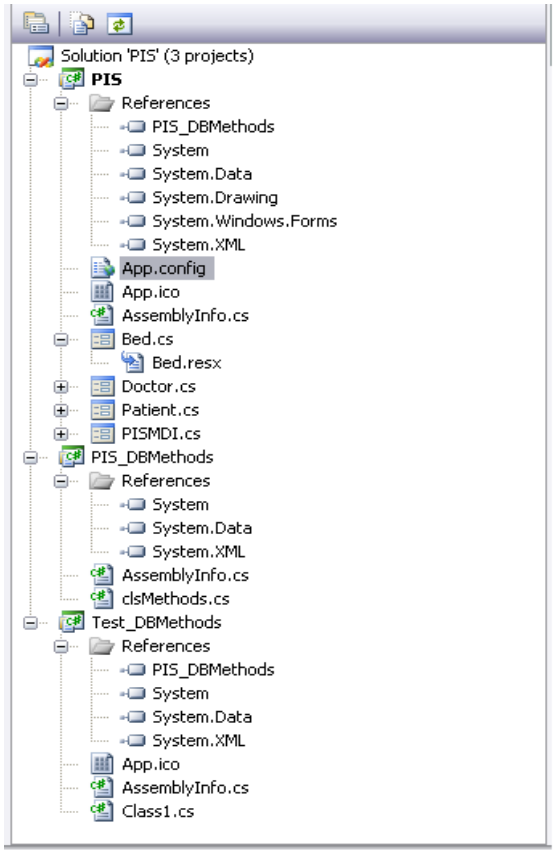

Fig 1. .NET Solution Explorer

1.2.1 Solution Explorer
In case of .NET Environment, there is a concept of solution under which we can have different projects which are classified as per their purpose. Generally, the business logic is put under

one project and user interface in some other project so that it implements loose coupling.

1.2.2 Project Explorer

Within a project, there can be few given files which are either C#.NET, VB.NET or ASP.NET based. Other files like xml, config etc can also be put under a project.

1.2.3 Language Selection

For web based applications, organizations use either VB or C# as the language with ASP.Net.

1.2.4 Database Connectivity

All the situations described above will require database connectivity considering an average case. The application configuration ie app.config file maintains the database connection string as given in the code below [Ref 6,11]. A developer has to get the string settings defined in the file and carry out the database related operations. Sample contents of app.config file are as shown below.

App.config file :

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<appSettings>
<add key = "Connection_PIS" value = "workstation id=HP11841122072; packet size=4096; integrated security=SSPI; data source= HP11841122072; persist security info=False; initial catalog=DB_PIS">
</add>
</appSettings>
</configuration>
```

2. PROBLEMS IN THE ABOVE SITUATION

2.1 Establishing Connection

The developer has to make use of some common statements as mentioned below in order to carry out the database related operations [Ref 7, 12].

```
{
Get connection string …….
Create a command object ……
Open connection ……..
Execute the command ……..
Get confirmation of operation ……
Close the connection …….
}
```

2.2 Executing DML Statements

For a given application development, the nature of Database Manipulation Language (DML) related statements is repetitive in the sense that these statements are mostly about insert, update, delete and select.

2.3 Query Builders

For simple queries, the developer hardly makes use of the query builders.

2.4 Data Insertion Errors

Data manipulation requires the data in proper format as per the database design. Consider for example, special characters: names or addresses can have a single quote character as data. This character and the SQL statements, cause incorrect data to get inserted into the database. Functions are available for taking care of such scenario, but they need to be explicitly called.

2.5 Incorrect Data Values

The incidence of committing a syntax error while writing the data manipulation language is high. Values may not be written as per the specifications. This leads to a considerable amount of time being lost in the coding and debugging process.

The end result that is seen:

"Focus gets shifted from business or presentation to the handling of data."

3. POSSIBILITY OF A DESIGN PATTERN / SERVICE

After the introduction of Object Oriented concepts, systems were developed keeping in mind not only the data and its behavior but also laying stress on the object oriented analysis and design methodologies. Currently, there is a need to have a pattern or a service based approach to most of the software tools and applications to introduce loose coupling and ease of maintenance.

In the above mentioned problems related to .NET and database related operations, there can be a possibility of a pattern or a service which handles the data. This service can be thought of as an entity above DAL (Data Access Layer), to relieve the developer of syntax errors and the data format errors. The service is as follows:

Service:

```
public bool insertValues(string [ ]ColumnName, Type [ ] ColumnType, object [ ] ColumnValues, string TableName)
{
try
{
strCon = openConnection();
con = new SqlConnection(strCon);
con.Open();
if((ColumnName.Length==ColumnType.Length)&&
(ColumnName.Length == ColumnValues.Length))
{
string strCmd = "insert into " + TableName + "(";
for (int i = 0; i < ColumnName.Length; i++)
{
strCmd += ColumnName[i] + ",";
```

```
}//end of for
strCmd = strCmd.Remove(strCmd.Length-1,1);
strCmd += ") values (";
for (int i = 0; i < ColumnType.Length; i++)
{

if(ColumnType[i].ToString() == "System.String")
{
//code goes here
}
else
{
//code goes her
}
}//end of for
strCmd += "'" + ColumnValues[i].ToString() + "',";
strCmd += ColumnValues[i].ToString() + ",";
strCmd = strCmd.Remove(strCmd.Length-1,1) + ")";
Console.WriteLine(strCmd);
cmd = new SqlCommand(strCmd, con);
cmd.ExecuteNonQuery();
con.Close();
}//end of if else
{
throw new Exception("Number of Arguments do not match.");
}//end of else
}//end of try
catch (Exception ex)
{
Console.WriteLine(ex.Message);
}//end of catch return true;
}//end of function insertValues
```

Amongst the basic categories of design patterns like structural, behavioral and creational, there are 23 design patterns which are used to provide the developer to think more on presentation than technical matters. Still, there exists a possibility of enhancing the patterns to move further towards business operations and presentation rather than the technical complexities.

3.1 Design Patterns Used

There are 4 design patterns that are of interest in this case. They are : Adapter, Command, Façade and Iterator [Ref 1,2,5].

3.2 Need for a Service

Combining the features of above mentioned patterns, there could be a possibility of a more advanced pattern/service which concentrates more on business operations than data. Suppose we call this service as the DMLOperation service or Transaction Service. The functionality of this service is to encapsulate the basic design patterns for handling of data.

3.3 Abstract Service

The proposed Transaction Service will be abstract by nature in the sense that the implementation will lie purely with the developer. The implementation can be done by calling the public method of the mentioned service.

3.4 App.config Encapsulated

The Transaction Service will have its own class definition in a different project which will get referenced from the developer's project so as to ensure that the developer is not bothered with the app.config file and SQL statements.

This approach can handle situations related to insert, delete, select (by using the iterator pattern) and update (by mentioning the 'where' condition). The command pattern can be used to connect ADO.Net to the Database and the façade pattern will ensure that right data is getting handled.

There could be a few drawbacks to this line of thought.

3.5 Memory Overhead

For developing any new application, the Transaction Service project needs to be referenced always which may prove a deterrent for applications that rely heavily on memory availability.

3.6 Performance of Database Extensive Applications

Referencing the Transaction Pattern in database extensive applications needs to be seen. Its efficiency in such a scenario needs to be thoroughly looked into.

3.7 Network Traffic

In case of n tier architecture, where there are different servers as per the structure, communication on the network with respect to ADO.NET also needs to be investigated.

4. CONCLUSION

The application of design patterns in the application development is not new but will continue to evolve. Based on previous work, the developments will lead us to the applicability of such patterns in .NET as well as many other software tools. We can say that there could be a possibility of a new service design to handle such recurring database related operations.

The management of project and its technical details need to be simplified for the experienced as well as inexperienced programmers. And when most of the focus desired by the IT businesses is on the presentation and business components, the above discussion of encapsulating the application configuration file by providing a new service, will benefit the business.

5. ACKNOWLEDGMENTS

I am thankful to the experts who have contributed towards development of the design patterns and continue to be more evolving.

6. REFERENCES

[1] Design Patterns in C# and VB.NET - Gang of Four (GOF) www.dofactory.com/Patterns/Patterns.aspx.

[2] Design Patterns for .Net - www.designpatternsfor.net

[3] ieeexplore.ieee.org/iel4/5805/15482/00713578.pdf

[4] Enterprise Solution Patterns Using Microsoft .NET msdn.microsoft.com/en-us/library/ms998469.

[5] Structural Design Patterns and .NET Framework 2.0 whitepapers.techrepublic.com.com

[6] .Net Architecture and Design Pattern- www.theserverside.net

[7] Design Patterns: From Analysis to Implementation www.netobjectives.com/download/dpmatrix.pdf

[8] Design pattern Resources TechRepublic search.techrepublic.com.com/search/design+pattern.html

[9] uidesign.net white papers index- www.uidesign.net/newpapers_index.html

[10] portal.acm.org/citation.cfm?id=1008038

[11] Storing database connection settings in .NET articles.techrepublic.com.com/5100-10878_11-5779516

[12] CodeGuru: Create a Microsoft Access Database Using ADOX and Visual Studio www.codeguru.com/vb/gen/vb_database/microsoftaccess/article.php/c5149/