DECORATOR PATTERN IN WEB APPLICATION

Viliam Malcher

Faculty of Management, Department of Information Systems, Comenius University, 820 05 Bratislava 25, Odbojárov 10, P.O. Box 95, Slovak Republic, Europe

viliam.malcher@fm.uniba.sk

ABSTRACT

Design patterns refer to reusable program solutions that aim to solve similar design problem during development process. The good practise of using patterns is encouraged and efforts have to be taken to knowledge on design patterns. We use a decorator pattern to improving a web application. Web application is implemented using HTML language, ASP.NET, and C#.

KEYWORDS

Design Patterns, Decorator Pattern, Web Application

1. INTRODUCTION

Developers use their own design for given software requirements, which may be new each time for developing of applications. Such method makes software maintenance more difficult and adapting of the software application to future will be problematic and consumes more time. A solution of this problem could be design pattern concept. Design patterns are software solutions optimized to the programming problems that we encounter every day. A design pattern is not a library that we can simply plug into our program. It is a template that to be implemented in the correct situation. Other words, the design patterns refer to solutions that aim to solve similar design problems during a development process.

The design patterns were introduced in 1994 in a book that specifies and describes 23 patterns [1]. J. Bishop has used in [2] examples of the patterns which might be used in programming of a modern computer system based on the novel features of the C# language. There are three basic kinds of design patterns: structural, creational and behavioural. Structural patterns generally deal with relationships between entities making it easier for these entities to work together.

V. K. Kerji in [3] has used abstract factory and singleton design to create decorator pattern objects in a web application. Next of his work [4], decorator pattern combined with XML data to add the additional responsibility to the user web page. P. L.Thung et al. in [5], adopted design patterns to improving a web application and they analysed two architectural patterns that are Model-View-Controller and Presentation-Abstraction Control. Adopting the design patterns with designing of web applications could be promoted reusability and more maintainable design. However, choosing the most suitable design patterns in the content of web application is no so easy.

In this work we have proposed a decorator pattern in a web application based using the HTML language, C# language and ASP.NET platform.

2. DECORATOR PATTERN

A decorator pattern expands the functionality of an instance of class without changing the class code. It provides a way to attach a new state and behaviour to an object dynamically. It means that the decorator pattern adds responsibilities to individual objects and the object does not know it is being decorated. We can add functionality to an object extending that class at compile time. Decorator is an object will wraps an object which needs to be added with new functionality at run time.

To better understand that, let us take into account an example. Consider the theoretical code of the decorator pattern published by J. Bishop [2] in a shorter version as it was published:

```
using System;

 interface IComponent {

    string Operation();

 }


 class Component : IComponent {

    public string Operation () {

      return "I am walking ";

    }

 }


 class Decorator: IComponent {

    IComponent component;

    public Decorator (IComponent c) {

       component = c;

    }

    public string Operation() {

      string s = component.Operation();

      s += "and listening my music";

      return s;

    }

  }
  class Client {

     static void Main() {

       IComponent component = new Component();

       Console.WriteLine(component.Operation());

       IComponent c =  new Decorator(component);

       Console.WriteLine(c.Operation());

    }

  }
```

/* Output:

I am walking

I am walking and listening my music

Listing 1: Decorator pattern class

The code starts with the `IComponent` interface and a simple `Component` class implements it. There is only one decorator that implements the interface - it includes a declaration of an `IComponent,` which is the object will decorate.

3. Implementation and Sample Code

In this work we have created a web based application using the HTML language with the decorator pattern concept. This web application allows different types of users (for simplicity we discuss only two users, User and Admin) to login and operate. The user page has functionality similar across all type of users. The admin page has content same as the user page but involves more elements. It means the user page is decorated and we get the admin page.

According the above design code we have proposed the following program for the web application:

```
<%@ Page Language="C#" Debug="true" %>


<script runat="server">
   interface IComponent {
      string Render();
   }
   class Component: IComponent {
      public string Render(){
          string s ="<table width='350' height='10' border='1'>"+
          "<tr><td><form method=post><input type=submit value='Send1'>"+
                      "<input type=submit value='Send2'>"+
                  "</form></td></tr></table>";
          return s;
      }
   }
   class Decorator : IComponent {
      IComponent component;
      public Decorator(IComponent c) {
         component = c;
      }
      public string Render() {
         string s = component.Render();
         s +="<table width='350' height='10' border='1'>"+
             "<tr><td><form method=post><input type=submit value='Admin1'>"+
                 "<input type=submit value='Admin2'></form></td>"+
                  "</tr></table>";
         return s;
     }
   }
   void User(Object sender, EventArgs e){
       IComponent component = new Component();
       Response.Write(component.Render());
    }
    void Admin(Object sender, EventArgs e) {
       IComponent component = new Component();
       IComponent c = new Decorator(component);
       Response.Write(c.Render());
    }
  </script>

<html><body>
<form id="form1" runat="server">
  <table width="350" height="100" cellspacing="0" cellpadding="0" border="1">
     <tr><td valign="top" rowspan="2"><a
href="http://www.google.com">Google</a></td>
         <td align="center" colspan="2"> Company</td>
     </tr>
    <tr>
        <td valign="top">Text, text,text, text, text, text, text, text </td>
         <td align="center">
           <asp:button id="btn1" Text="User" width="60" OnClick="User"
runat="server"/>
```

```
           <asp:button id="Btn2" Text="Admin" width="60" OnClick="Admin"
runat="server"/></td>
    </tr>
   </table>
 </form></body></html>
```

Listing 2: User and Admin interfaces

The program output reads:

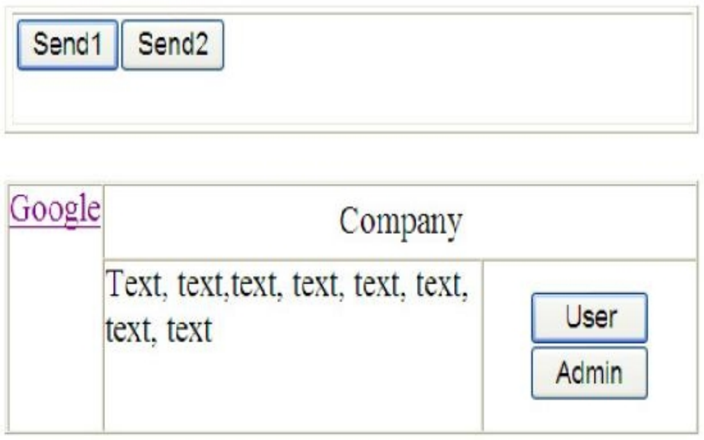

Figure 1. The output of an user page

When you click on the Admin button the web page is changed, see Figure 2.

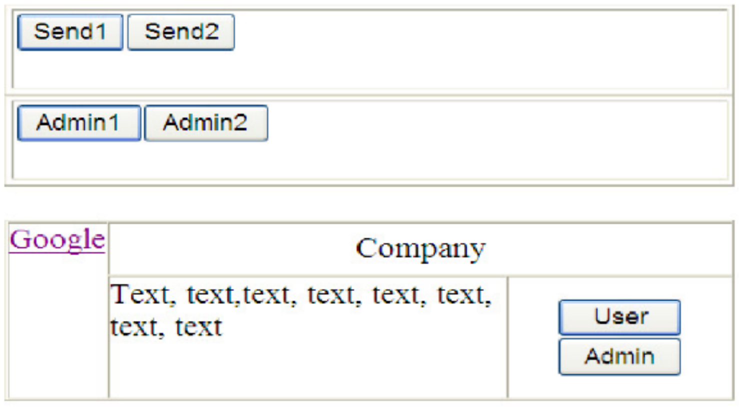

Figure 2. The Admin page is decorated by the User one

This is the Admin user interface. We have obtained this page from User one and the software adds the new elements to the admin functionality.

4. CONCLUSIONS

Software applications used with the design patterns enhance performance and maintenance. The design patterns make the software design more flexible to the changes in requirements and improve the application performance. We have applied the decorator pattern to the web applications based on HTML and C# language. Using the decorator we can add a new state behaviour to an object in runtime. This provides us the flexibility of creating an instance of decorators on an as-needed basis. This would provide a real advantage in scenarios where the additional responsibility increases the use of memory.

Design patterns refer to reusable or repeatable solutions that aim to solve similar design problems during development process as we have written above. Software architects and developers use them to build high quality robust applications. The new version of HTML5 and CSS3 [4] give us to increase a quality for development of web application based on HTML language and it could be better to support the maintenance of the software together with the decorator pattern concept.

REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, J. M. Vlissides, (1995) *Design Patterns: Elements of Reusable Object Oriented Software,* Addison-Wesley, Boston, MA.

[2] J. Bishop, (2008) *C# 3.0 Design Patterns*, O‘ Reilly.

[3] V. K. Kerji, (2011), ”Decorator Pattern with XML in web application”, *Electronics Computer Technology* (*ICECT*)*, 3rd International Conference*, Vol. 5, No. 5, pp. 304-308.

[4] V. K. Kerji, (2011) ”Abstract Factory And Singleton Design Patterns To Create Decorator Pattern Objects in Web Application”, *International Journal of Advanced Information Technology* (*IJAIT*), Vol. 1, No. 5, October.

[5] Phek Lan Thung, Chu Jian Ng, Swee Jing Thung, Sulaiman S. (2010), "Improving a web application using design patterns: A case study" *Information Technology (ITSim),* 2010, International Symposium, Vol. 1, pp. 1-6, 15-17 June.

[6] B. P. Hogan, (2011) *Pragmatic Programmers*, LLC.

AUTHOR

Viliam Malcher (PhD) born in Bratislava, Slovak Republic. He graduated from the Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, in 1981. From 1983 he was engaged in the industry as a researcher. After he received PhD degree in 1995 worked in Computer Centre at Comenius University and from 2001 in Analytical Laboratories as IT worker. In this time he joined the Faculty of Management, Department of Information Systems at Comenius University. He is a teacher and his research interests include programming, mainly on the .NET platform, object oriented analysis, and quantum computing.