

# .Net 开发中 MVC 模式的应用

孙健伟,高 岭,王晓晔,吕颜兴

(西北大学 信息科学与技术学院,陕西 西安 710127)

**摘 要:**介绍了如何利用 .NET 平台结合 MVC 模式开发安全、可扩展、易维护的系统。分析了 MVC 设计模式,设计了一个基于 MVC 模式的 .NET 框架下的应用模型,并结合该实例,探讨了具体的实现方法,并给出了每个模块的代码实现。该系统实现了应用程序模块化,具有良好的安全性、可扩展性和易维护性。

**关键词:** .Net; MVC 设计模型; BLL; DAL

**中图分类号:** TP311

**文献标识码:** A

**文章编号:** 1673-629X(2007)11-0008-03

## Application of .Net Development with MVC Model

SUN Jian-wei, GAO Ling, WANG Xiao-ye, LÜ Yan-xing

(College of Information Science and Technology, Northwest University, Xi'an 710127, China)

**Abstract:** In this paper, how to make use of MVC model to develop a safe, dependable and easily maintainable system is introduced. At first, model design and MVC design model are analyzed, and then the application of these patterns and architecture based on Microsoft .Net framework by an example system are discussed. This is a safe, flexible and easily maintainable system which is modularized.

**Key words:** .Net; MVC model; BLL; DAL

## 0 引 言

随着 Internet 的普及,基于 B/S 结构的大型 Web 应用越来越多,这些应用几乎都是以 .Net 和 Java 为开发平台。目前关于 J2EE 的设计模式,人们已经提出了很多,但是对于 .Net 的设计模式,研究的并不很多。

随着时间的推移,Web 开发当中令人头痛的问题逐渐暴露出来,如开发周期漫长、客户需求变化频繁、维护成本高等。以传统设计模型开发软件时,由于模型的柔性差,使得项目进展缓慢,软件更新困难,甚至一个处于维护期的产品,当用户提出新需求时,要从头做起进行新的开发工作。软件设计中 MVC 模型的使用在一定程度上缓解了这一问题。MVC 模型提倡视图(View)与控制器(Controller)相分离,允许一个开发者将一个好的面向对象的设计与用户接口隔离开来,在同样的模型中容易地使用多个接口,并且允许在实现阶段对接口作大的修改而不需要对相应的模型进行修改。使得在用户界面很少变化的情况下,隐式地

改变网站的事务处理,同时支持多个用户界面对应相同的事务处理。大大简化了面向对象软件的设计开发过程,开发出的软件便于修改,具有良好的柔性。

文中以成功应用 MVC 结构的 .Net 应用为基础,介绍了一种 MVC 模块的划分以及各层之间的关系,并给出了每一个模块的具体设计。

## 1 MVC 原理

MVC(Model-View-Controller)设计模式是一种最早应用于 Smalltalk 80 的设计范式。MVC 中,Model 表示用户与之交互的对象,通常包括业务数据和业务逻辑,View 从 Model 中抽取数据并将其以用户易于理解和交互的方式显示,Controller 则对用户的请求进行解释,并将必要的控制信息交给 Model 和 View。Controller 和 View 对 Model 有充分了解,而 Model 对 Controller 和 View 除了知道它们存在外知之甚少。因此,对 Controller 和 View 的改变不会影响 Model<sup>[1]</sup>。

MVC 设计模式因其模块划分清晰、责任明确、易重用、易伸缩和易维护,已成为当今设计交互式应用的事实上的标准,但是作为一种基本的设计思想,它还需要详细的设计规划,实现并不十分容易。同时由于将应用分为三层,这就意味着代码文件增多,因此,对于文件的管理就需要费些心思。

收稿日期:2007-01-02

基金项目:陕西省自然科学基金项目(2005f36)

作者简介:孙健伟(1981-),男,河北人,硕士研究生,研究方向为计算机网络及其应用;高 岭,教授,博士,研究方向为计算机络性能分析、服务质量及其应用研究。

## 2 .Net 平台应用

在 Java 中, MVC 设计模式已有了比较成熟和广泛的应用设计方案, 比如 struts, 而在 .Net Web 应用中, 由于 Microsoft 推荐使用“Code Behind”代码来处理每一页面的用户交互, 怎样应用 MVC 一直令人感到困惑。

.Net 采用“Code - behind”代码来处理显示页面大量用户交互。即每个 aspx 文件都对应一个“Code - behind”源代码文件。通过“Code - behind”实现逻辑控制代码与脚本语言, HTML 等代码的分离<sup>[2]</sup>。

一个 Web 页面对应一个 aspx 文件, 它的设计可以通过 HTML 设计工具编写, 最后将其后缀修改为 .aspx。另外, 在 .Net 开发环境中设计视图也非常便捷。Net 在视图设计器的工具箱中提供了数据、Web 窗体、组件、HTML 等各种控件, 同时还提供了用户控件。用户控件是一种直观的可重用的模型, 扩展名位 .ascx。它可以是最简单的 HTML 控件、服务器控件或多个控件嵌套构成的 Web 自定义控件。可以像开发 Windows 界面一样方便地通过所见即所得的方式进行可视化设计<sup>[3]</sup>。

与 aspx 对应的 Code - Behind 源代码文件, 后缀为 cs。通过“Code - Behind”实现了逻辑控制代码和 HTML 代码的分离。这些 cs 文件代码继承自 System. UI. Web. Page 类。它包括了各种初始化和控制函数。当加载 aspx 页面时将调用 Page. Load 时间, 当 aspx 页面从内存中被卸载时将调用 Page. Unload 事件。如果某个控件触发页面以使其被重新加载则将调用 Control Event 事件<sup>[2]</sup>。

由此可以看出, .Net 很好地契合了 MVC 架构, 也即所有 aspx 文件组成的 Web 层对应了 View, 所有 cs 文件组成了控制层。由 aspx 文件确定数据的显示方式, 由 cs 文件执行页面传入的操作并为其绑定数据。

## 3 设计案例

该项目是以教学平台为主, 包括论坛和博客的大型 Web 应用。系统架构如图 1 所示。

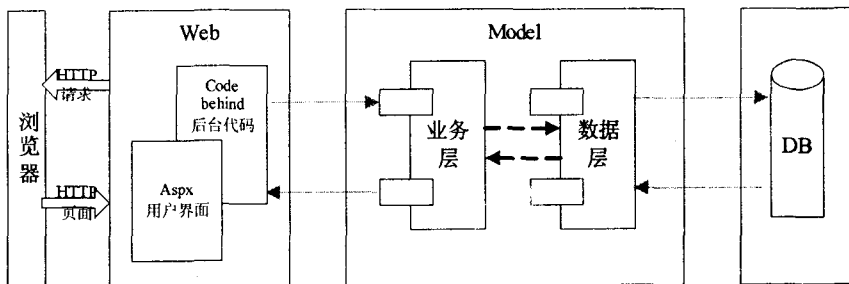


图 1 三层结构图

根据图 1 的设计, MVC 中的 Controller 和 View 位于三层架构中的显示层, 即 Web 层, 而 MVC 中的 Model 则涵盖了三层架构中的业务逻辑层, 即 BLL 和 DAL。而图中的数据表示层是为业务逻辑层或表示层提供数据服务, 对原始数据进行操作<sup>[4]</sup>。

下面针对各层给出具体的设计。

### 3.1 Web 层的设计

aspx 文件确定了数据显示的方式, 并由 cs 文件根据接收 aspx 页面传来的请求决定要显示的下一个页面, 并为其绑定数据。下面以平台中的博客显示页面为例:

```

public class blogindex : System. Web. UI. Page
{
    private void Page. Load(object sender, System. EventArgs)
    {
        .....
        blogbind(); //完成博客数据绑定
        .....
    }
    public void blogbind()
    {
        .....
        BLL. Blog Bblog = new BLL. Blog();
        ds = Bblog. GetBlogs(); //以上两行为代码(2)
        .....
        this. blog. DataSource = ds; //ds 是数据源, blog 为页面控件
        this. blog. DataBind(); //这两行完成数据绑定
        .....
    }
}
  
```

在 Web 应用中, 安全性(访问控制)是构建安全页面的第一个也是最重要的一个因素。它包括验证和授权, 前者往往是通过登陆页面来实现, 通过输入正确的用户名和密码以确认它的身份, 后者是用权限控制用户能够访问的资源<sup>[2]</sup>。

在本案例中使用的是 .Net 环境提供的基于角色的访问控制, 以让 Web 应用同时进行用户的验证和用户的权限验证。一种角色对应多个权限。每一个权限

对应一个页面, 只有具有页面权限的用户才能浏览该页面。具体做法如下: 提供登陆页面, 将成功登录的用户角色名存入 session, 并借此判断登录与否。在用户打开某一页面时, 对应这一 aspx 文件的 cs 文件获得浏览该 Web 页面的权限名, 并以此为参数, 调用相关函

数查询该用户是否拥有此权限,如果有则显示页面,否则给予错误提示。

### 3.2 Model 层设计

Model 中封装了应用所需要的数据以及操纵这些数据的方法。应用所需要的数据包括描述 Model 状态变更的用户会话数据,以及对应于数据库中物理存储的实体数据。这些数据一般位于 BLL 中,而 DAL 则负责将 BLL 中的实体数据映射到数据库中。

#### 3.2.1 BLL 的设计

BLL 是对业务逻辑的实现,它从 Controller 接收请求,执行业务逻辑处理,并将处理结果返回给 Controller,以供 View 显示,而 BLL 的业务处理一般直接调用 DAL 的某个方法来处理,如以下代码所示:

```
using CidaSolution.DAL;
using CidaSolution.Model;
public class BLL_Blog
{
    public T_Blog Tblog = new T_Blog();
    public BLL_Blog(int blog_ID)
    {
        DAL_Blog Dblog = new DAL_Blog();
        Tblog = Dblog.GetBlogByBlogId(blog_ID);
    }
    .....
}
```

如上面代码所示,定义了一组实体类,每一个实体类对应数据库中的一个表,用于接收调用 DAL 处理方法后返回的单条数据,该类只包括字段和对应于所有字段的属性。代码如下:

```
public class T_Blog
{
    private int _blog_id;
    private string _blog_name;
    public int blog_ID
    {
        set { _blog_id = value; }
        get { return _blog_id; }
    }
    public string blog_Name
    {
        set { _blog_name = value; }
        get { return _blog_name; }
    }
    .....
}
```

而数据集则有 DataSet 类定义的对象接收,最后

BLL 将这些数据返回给 Controller,以确定是刷新当前页面还是跳转到其他页面<sup>[5]</sup>。

#### 3.2.2 DAL 设计

DAL 主要是为 BLL 操作数据提供支持。如图 2 所示。

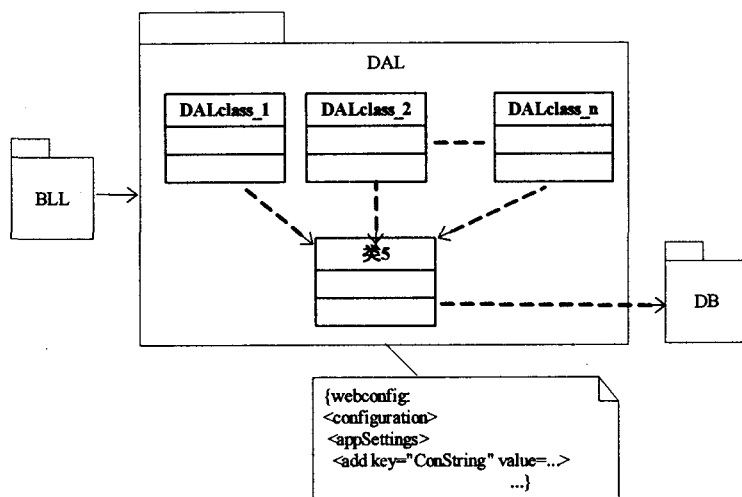


图 2 DAL 层设计图

DAL 由多个对数据库操作类和一个 DBManageSQL 类组成,其中 DBManageSQL 类是核心类,该类负责完成系统中所有对数据库的访问。DAL 的所有其它类则负责构造数据访问参数,并依赖于 DBManageSQL 类访问数据库。DBManageSQL 设计如图 3 所示。

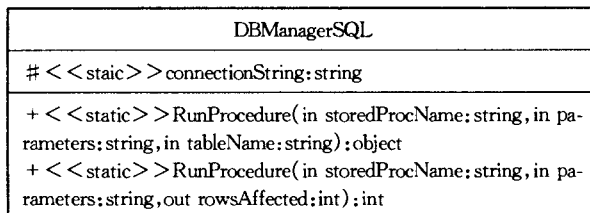


图 3 数据库操作类设计图

```
public DataSet GetBlogArticleByBlogIdAndTopic(int blog_Id, string
article_Topic)
{
    SqlParameter[] parameters =
    {
        new SqlParameter("@article_BlogId", SqlDbType.Int, 4),
        new SqlParameter("@article_Topic", SqlDbType.VarChar,
100)
    };
    parameters[0].Value = blog_Id;
    parameters[1].Value = article_Topic;
    DataSet ds = new DataSet();
    return
    DbManagerSQL.RunProcedure("UP_Get Blog Article By
```

(下转第 141 页)

志,  $ID_{m+1}$  是  $C_{m+1}$  身份信息。

3) B 用密钥  $K_1$  解密  $J$  获得时间  $t'$ ,  $ID_{m+1}$ , 并核对时间  $t'$  的有效性。如果没有问题, B 通知  $C_{m+1}$ 。

4)  $C_{m+1}$  和 B 建立一个保密密钥  $K_{m+1}$ , 用保密密钥建立方案。

5) B 发送如下信息:

$$I_1 = E_K(ID_B)$$

$$I_2 = E_K(t' \parallel L)$$

$$I_3 = (A_{m+1}, E_{K_{m+1}}(K))$$

其中  $t'$  是时间标志,  $A_{m+1}$  是  $C_{m+1}$  的化名。

6)  $C_{m+1}$  从  $I_3$  中获得密钥  $K$ , 与其他会议员一样确认它的真实性, 然后加入会议。

### 3 安全性分析

许多无线通信会议分配方案都被证明为不安全的。文献[2]中协议设计是动态的, 因此, 开会中会议员应该知道所有的会议员, 但只有 NC 知道这些。NC 认证每个参与者, 两个参与者不能通过会话密钥 CK 来鉴别对方是否为合法的参与者, 所以这个方案对模仿和偷听是不安全的。方案<sup>[2-5]</sup>被证明在抵抗主动攻击和被动攻击时不安全的。主动攻击是一组人共谋实施的, 他们首先建立一个会议通路, 通过主动攻击获得网络中心的会话密钥, 邀请其他人加入。当他们逐渐离开会议后, 最后, 会议由一个完全不同的一组人继续进行, 没有任何一个攻击者参加, 但攻击者仍保留着 NC 通话密钥, 能解密通话会议。换句话说, 会议密钥的更新对攻击者是透明的。被动攻击在会议员人数很多时起作用。当有千个会议员时攻击可能成功, 因为

(上接第 10 页)

Topic", parameters, "ds");

}

### 4 试验结果

该系统将 MVC 架构应用与 .Net 开发, 使得整个软件开发呈现出系统性, 成功实现了软件开发的分工。实现了应用程序的模块化。在满足需求的基础上, 是一个具有良好安全性、可扩展性和易维护的交互式系统。

### 5 结论

MVC 架构是一种非常先进的设计思想, 已经被广泛应用于软件开发之中。文中讨论了 MVC 的原理, 以实例方式给出了一种具体的 MVC 设计方案, 并给

攻击者不需要参加会议, 所以攻击是被动的, 他只需要截取通信和进行计算操作。只要会议员的数目大, 会议密钥的不安全概率就高。

### 4 结 语

给出了基于模平方根技术移动密钥分配方案, MSR 求解的困难性依赖于大整数分解的困难性, 基于 MSR 的密钥分配方案是安全的。文中对原有方案进行了分析与改进, 使得可以更有效地抵抗重用攻击, 并且没有增加通信与计算负担。

### 参考文献:

- [1] Hwang Min - Shang, Pang Wei. Conference Key distribution protocols for digital mobile communication systems[J]. IEEE journal on Selected areas in communications, 1995, 13(2): 416 - 420.
- [2] Hwang Min - Shang. Dynamic Participation in a Secure Conference Scheme for Mobile Communications[J]. IEEE transactions on vehicular technology, 1999, 48(5): 1469 - 1471.
- [3] Hwang Kuo - Feng, Chang Chin - Chen. A Self - Encryption Mechanism for Authentication of Roaming and Teleconference Services[J]. IEEE transactions on wireless communications, 2003, 5(3): 400 - 407.
- [4] Yi Xun, Siew Chee Kheong, Tan Chik How. A Secure and Efficient Conference Scheme for Mobile Communications[J]. IEEE transactions on vehicular technology, 2006, 52(54): 784 - 793.
- [5] Bao Feng. Analysis of a Secure Conference Scheme for Mobile Communication[J]. IEEE transactions on wireless communications, 2006, 5(8): 1984 - 1987.

出了每一模块的具体设计。这种设计方案围在 .Net 环境下开发具有较高安全性、易于维护和易于扩展的 Web 应用提供了可行方案。

### 参考文献:

- [1] 姚慧广, 赵岳松. Web 编程中 MVC 模型的应用[J]. 微机发展, 2002, 12(3): 9 - 10.
- [2] Prosise J. Programming Microsoft. Net[M]. 北京: 清华大学出版社, 2005.
- [3] 庞开放, 李龙澍. 基于 .NET 框架的 web 应用与实现[J]. 微机发展, 2005, 15(3): 85 - 89.
- [4] 卫索琪. 基于 MVC 模式的一种 Web 应用框架[D]. 北京: 北京工业大学, 2003.
- [5] 刘 克. MVC 架构及其在 Web 应用开发中的应用[J]. 计算机应用与软件, 2006, 23(7): 57 - 59.