成 绩:\_\_\_\_\_\_\_\_\_\_\_\_

教师签名:\_\_\_\_\_\_\_\_\_\_\_\_



**需求分析说明书**

**项目名称:** 基于访问权限控制的文件上传系统

教 师: 王欣

姓 名: 唐可寅

学 号: 201710253110

班 级: 17gb计算3班

日 期: 2020年07月15日

目录

[一、前言 1](#_Toc34954447)

[1、项目前言 1](#_Toc1216355098)

[2、系统的应用背景 1](#_Toc1383296293)

[3、系统的开发意义 1](#_Toc402834029)

[二、 可行性研究 2](#_Toc1563070059)

[1. 可行性研究的前提 2](#_Toc351027862)

[2. 管理可行性 2](#_Toc587698325)

[3. 经济可行性 2](#_Toc1168455722)

[4. 运行上可行性 2](#_Toc1644851486)

[5. 技术可行性 2](#_Toc461937371)

[6. 社会因素可行性分析 2](#_Toc628010492)

[三、系统需求分析 3](#_Toc90214039)

[1. 系统功能概述 3](#_Toc103898691)

[2. 系统功能需求 3](#_Toc321094626)

[3. 详细功能需求介绍 3](#_Toc10974271)

[4. 项目环境和依赖 4](#_Toc1908462702)

[四、系统设计 5](#_Toc716880922)

[1. 系统功能流程图 5](#_Toc1234396384)

[2. 系统设计的原则 5](#_Toc1807995868)

[3. 业务流程分析 6](#_Toc92948426)

[4. 数据库设计与实现 6](#_Toc963584413)

[五、 应用程序设计 12](#_Toc789047264)

[1. 前言 12](#_Toc805845823)

[2. 概要设计 12](#_Toc1818869179)

[3. 特定模块的详细介绍 12](#_Toc304576408)

[分层分析 12](#_Toc1562158455)

[详细设计 17](#_Toc62084963)

[六、附：页面前端程序代码 19](#_Toc1932404346)

[1. 用户登录页面 19](#_Toc1524649641)

[2. 文件上传页面 21](#_Toc1011640283)

[3. 远程数据库连接（web.config）代码片段： 23](#_Toc1010203082)

[4. 系统的测试 23](#_Toc477485992)

[七、结束语 25](#_Toc2108162352)

# 一、前言

## 1、项目前言

文件的上传和下载是网站最常用的功能。 以项目实战方式利用HTML语言（HTML）结合客户端 JavaScript 脚本动态产生控件。 并用C＃语言编写后台代码真正实现ASP.NET多文件上传和下载，为ASP.NET平台中文件上传和下载的处理提供了一个编程框架。

本次课程设计实现管理用户登录和文件上传的功能。本次设计首先对系统建设对各个层面进行了基本定义，并在其基础上进行了简单分析，阐述中给出了部分UML图以说明系统结构。本次设计以基于用户登录的文件上传系统为基准实行此次系统的开发。

## 2、系统的应用背景

开发的软件系统的名称：基于用户登录后的文件上传系统。

在 Web 应用系统开发中，经常须要上传文件保存到服务器上，因此文件的上传和下载功能在网站中是经常被需要使用到的功能。本次课程设计选用C#语言整合ASP.NET框架，数据库选用SQL Server进行数据管理。

Web客户端页面对文件的上传或者下载操作是ASP.NET开发过程中经常会用到的一项技术。这项技术可以给用户带来非常方便的文件操作, 可以直接在网页上进行文件的上传或者下载。通过Web网页上传或者下载的文件一般为除文件夹以外的其他文件, 例如压缩文件、Word文档、Excel文件、EXE应用文件等等。所以在文件进行上传时, 要先对文件进行前期的处理, 使之成为可以上传的非文件夹文件。经过处理后的文件才可以进行上传, 否则进行上传的文件将无法进行上传。

## 3、系统的开发意义

文件的上传和下载功能在网站中是被经常用到的功能，文件上传系统一旦使用基本下包含以下几个方面的意义：

1. 加快信息处理速度；
2. 提高工作效率；
3. 减轻劳动强度；
4. 实现对资源的共享作用；

除了上述列举的4个主要积极意义外，上传功能能缩短系统管理者和系统使用者的空间距离，是系统使用者能够快速的向系统管理者发布所需求的内容，文件上传系统不论是对系统管理员还是对社会与时俱进方面来说对于软件系统开发的研究来说都是十分重要的。

# 可行性研究

## 可行性研究的前提

* 1. 项目的基本功能要求

本系统主要包括四种操作：文件上传、文件下载、图片上传和回显。

在文件上传页中可以通过使用“添加文件”按钮来动态添加文件上传控件，通过使用文件上传控件可以添加需要上传文件的路径。文件下载需要先选择文件名才可以下载到本机。图片上传页面需要判断文件是否为图片オ可以上传。

* 1. 项目实现的目标

可以进行多文件同时上传，并且可以在下载页面下载所上传的文件，上传图片以后可以在另一页面查看。

## 管理可行性

随着时代的发展，人员素质己经逐步提高，不论是对于计算机系统的基本操作还是对于系统的维护都有了一定的基础。同时还可以配置专业的计算机维护人员来维护电脑，不必担心计算机故障问题。

## 经济可行性

系统建设不需要很大的投入。

## 运行上可行性

本系统只是网站开发的一部分，所耗费的资源非常的小。

## 技术可行性

目前, ASP.NET技术中主要通过两类方法实现文件的上传功能：第一类，利用第三方组件实现上传，但这类方法需要在服务器上注册；第二类, 无组件上传，当使用别人的服务器时，对方可能不允许安装第三方组件，这时就必须使用无组件上传。

## 社会因素可行性分析

从各种社会因素可行性分析：可提高文件的利用率，提高效率更有利于网友的交流会得到人们的一致同意的。

综上所述通过可行性分析认为新系统的开发方案切实可行，可进行开发。

# 三、系统需求分析

## 系统功能概述

信息化是当今世界经济和社会发展的大趋势。在信息化时代企业如何充分、快捷、有效地利用企业内外的大量信息，为企业的发展服务，提高企业的工作效率，是企业需要充分考虑的问题之一。随着企业规模的扩大以及服务内容和办公设施的现代化，这种情况日益突出同时，为了提高单位的日常办公效率，减轻手工劳动强度，使单位的日常运营纳入高效而快捷的轨道，就必须利用现代的信息手段。现代信息技术的优势在于对信息的处理高效而精确可以充分实现信息的共享和传输，及时而且透明地反映企业办公的正常活动过程与变动情况，完善办公调度，提高企业服务质量，进而实现企业经营管理的信息化、科学化。充分利用现代信息技术，是提高企业竞争力的必要手段。

## 系统功能需求

本系统包含如下基本功能：

1. 浏览器端页面
   1. 提供用户登录注册的功能；
   2. 提供实现单个或多个文件上传的功能；
   3. 提供对于用户上传文件大小限制的功能；
   4. 提供可视化的上传进度统计能力；
   5. 提供友好的用户界面；
2. 服务端
   1. 提供向所有使用者提供服务的能力；
   2. 提供用户登录注册的能力；
   3. 提供对服务器访问压力控制的能力；
   4. 提供自定义上传文件类型的能力；
   5. 提供指定目录上传的能力。

## 详细功能需求介绍

本系统由文件上传、文件下载、图片上传和回显四部分操作构成。

首先由用户访问本系统首页，本系统首页为系统功能介绍展示页。用户点击登录后跳转至登录页面，用户数据个人用户名和密码后，系统进行身份访问权限核验，核验通过后，系统将跳转至文件上传页。用户在文件上传页进行文件上传，用户提交的文件，服务器将进行处理并核验文件是否合规，合规的文件将被服务器存留于磁盘指定区域，不合规的文件将被服务器丢弃，并给出驳回上传请求的响应。用户上传成功后服务器将会跳转至文件回显页面，显示出用户上传的文件。至此本程序流程结束。用户可进行其他文件上传或用户退出等操作。

## 项目环境和依赖

* 开发环境
  + 开发语言：C#
  + 框架技术：ASP.NET
  + 开发工具：Visual Studio 2017 & JetBrains Rider 2020.1.4
  + 数据库：Microsoft SQL SERVER 2019(14003281) Linux
* 环境配置
  + 数据库访问路径：bj.kevinkda.cn:1433/university
  + 数据库类型：Microsoft SQL SERVER 2019 Linux
  + 服务端口号：1433
  + 数据库名：university
* 第三方库依赖
  + 无

# 四、系统设计

## 系统功能流程图

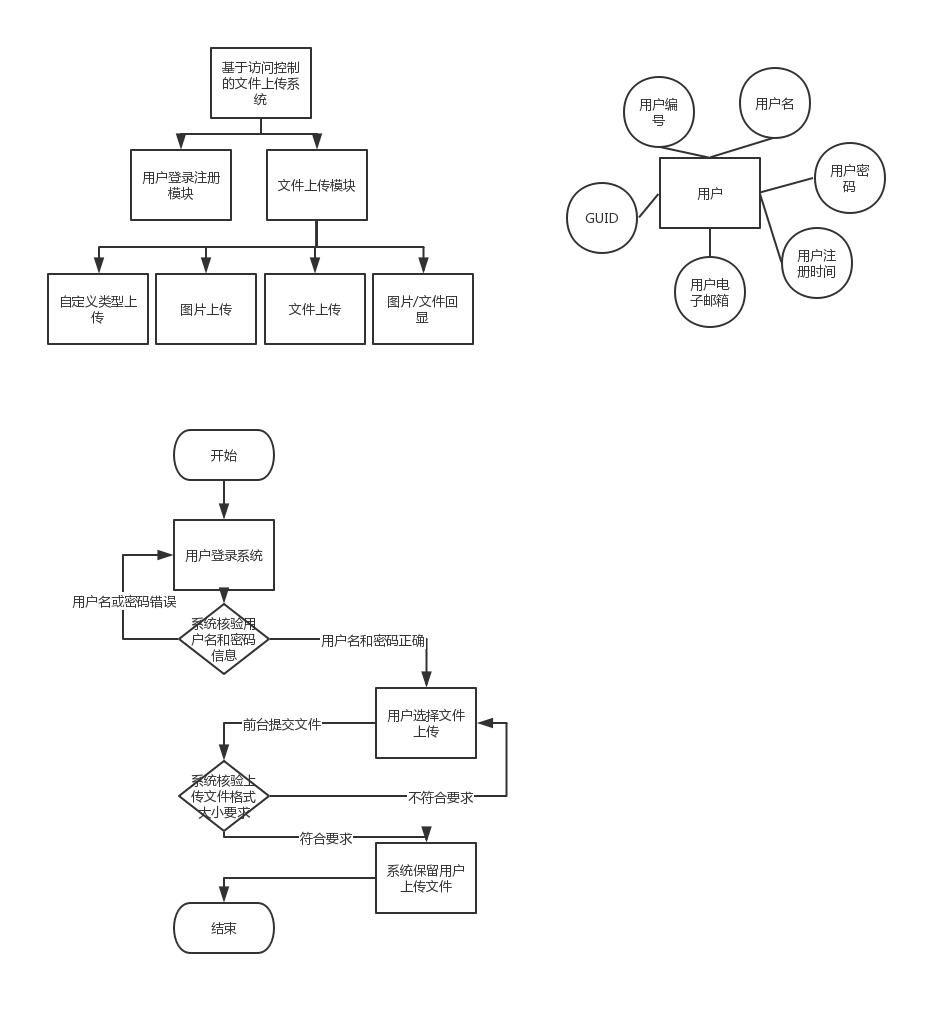


图 1系统功能流程图

## 系统设计的原则

ASP.NET设计基本原则

在开始设计任何web站点之前，回顾一下优秀站点设计的原则是很有益的。在网站设计中，应该记住三个基本概念：

* 尽量将信息与表示分隔开。例如，开发人员将设计页面的标题、布局和格式（表示）。在页面上，开发人员将放置配置好的控件以获取和显示信息（例如球队的球员列表）。通过将这些目标分隔开，开发人员可以分别对每个部分更新而不会影响到其他部分。例如，当球队增加了一名新球员时，关于该球员的信息将保存在数据库中，页面将自动显示新的球员列表而无需修改表示层。
* 尽量保证站点具有一致的外观。通过使用相同的色彩、图标和屏幕布局，可以为用户创建一种临场感（sense of presence)。忠实的球迷一看到这种色彩就感觉到像到了家一样。回访用户可以用利用已经掌握的经验使用该站点。
* 尽量简化站点导航。首先，所有页面上的菜单条将简化从站点的一个部跳转到另一个部分的导航。用户还将使用ASP.NET2.0工具显示当前正在查看的页面位于站点中的什么位置。

## 业务流程分析

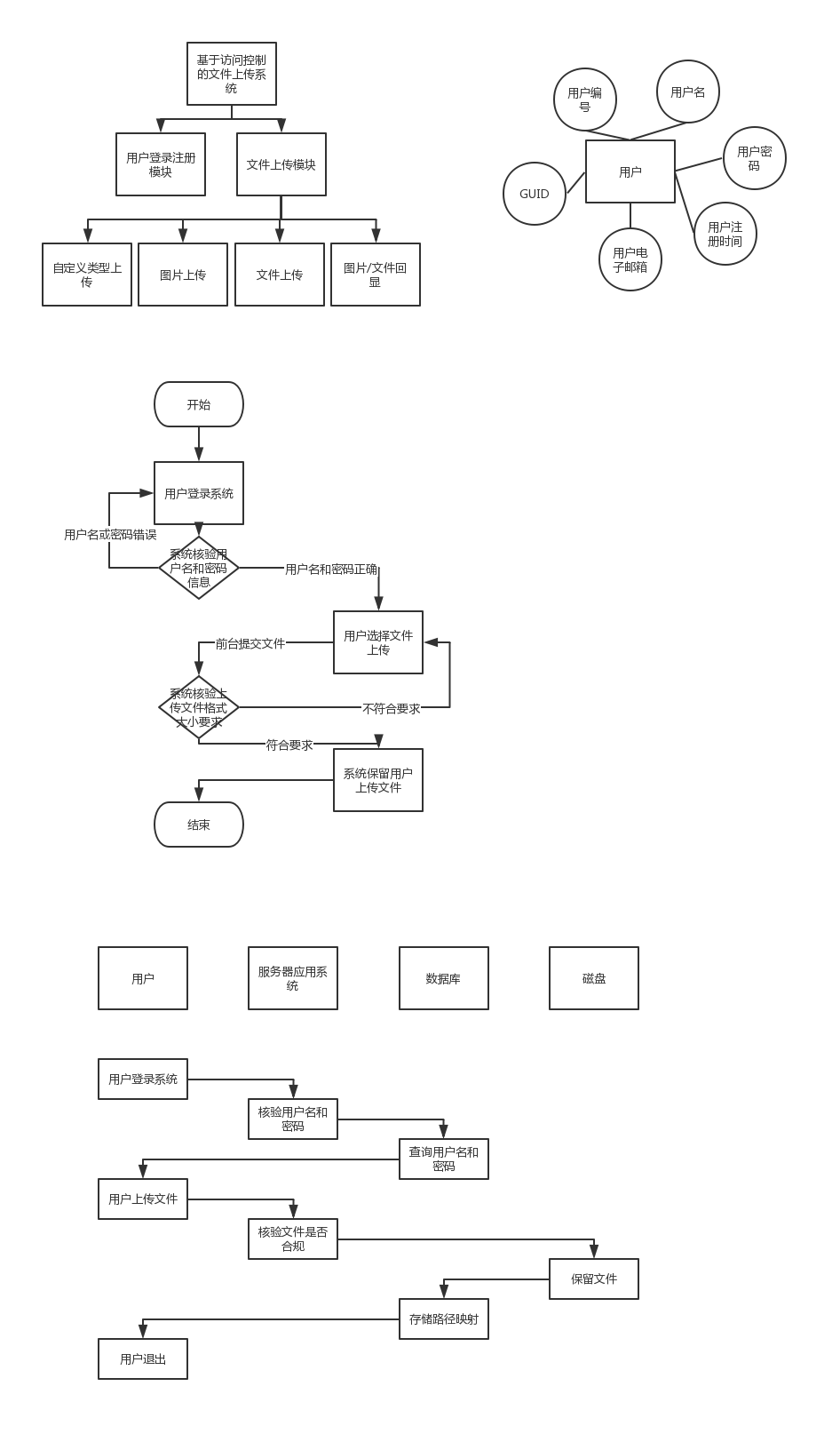


图 2业务流程分析

## 数据库设计与实现

* 1. SQL Server数据库

SQL Server 是作为一个服务器的数据库系统使用的，并非桌面系统，比如 MS Access。

SQL Server 被设计为在中央服务器上运行（或服务器），使多个使用者可以同时访问相同的数据；用户通常通过应用程序访问数据库。

例如，一个网页可以存储在数据库中的所有内容。当访问者浏览的文章，他们从数据库中检索数据。一个网站服务的对象达到了数百，甚至数千的访客。同时，还得满足其他用户可以更新他们的个人资料，会员区，还能订阅新闻简报或其他任何网站的用户操作。所以有很多用户都是同时读取和更新数据库的，那么一个良好的，强大的数据库系统满足这种类型的用法。毕竟，你不希望你的数据库锁定，因为太多用户试图访问它，或者更糟的是，你不会希望在你的数据库，由于电线损坏而致数据获取不到。

通常，这是是由网站的应用程序提供的功能，以这些访问者（例如，可以使用如，ColdFusion，HTML和JavaScript来构建网站）。它使用数据库存储数据，并使其可用。但是，SQL Server不包括一些有用的功能，可帮助应用程序提供的功能。

* 1. 数据库设计——物理模型

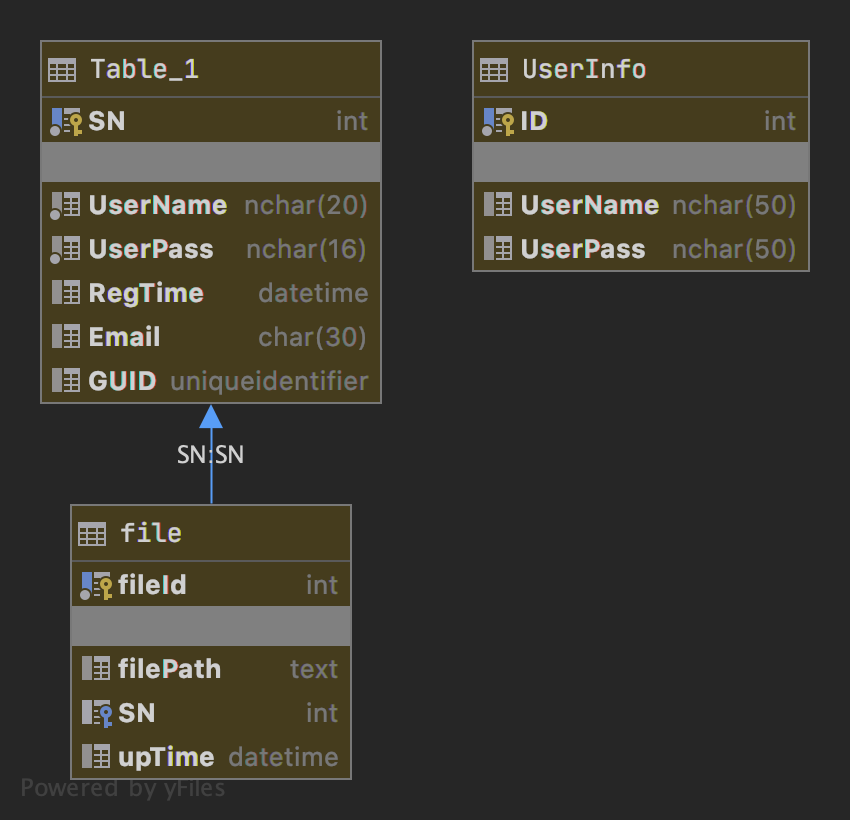


图 3数据库设计——物理模型

* 1. 数据库表字段
     1. 用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 约束 | 注释 |
| SN | int | 11 | 主键，自增长 | 用户ID |
| UserName | varchar | 255 | 非空 | 用户名 |
| UserPass | varchar | 255 | 非空 | 密码 |
| RegTime | datetime |  | getdate() | 注册时间 |
| Email | varchar | 255 |  | 电子邮箱 |
| GUID | varchar | 32 | 非空 | UUID |

* + 1. 文件存储信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 约束 | 注释 |
| fileId | int | 11 | 主键，自增长 | 文件ID |
| filePath | text | 255 | 非空 | 文件路径 |
| SN | varchar | 255 | 非空 | 用户ID |
| upTime | datetime |  | getdate() | 上传时间 |

* 1. 数据库实现代码

/\*

Navicat Premium Data Transfer

Source Server : Aliyun ECS SQL Server

Source Server Type : SQL Server

Source Server Version : 14003281

Source Host : bj.kevinkda.cn:1433

Source Catalog : university

Source Schema : dbo

Target Server Type : SQL Server

Target Server Version : 14003281

File Encoding : 65001

Date: 14/07/2020 21:36:54

\*/

-- ----------------------------

-- Table structure for file

-- ----------------------------

IF EXISTS (SELECT \* FROM sys.all\_objects WHERE object\_id = OBJECT\_ID(N'[dbo].[file]') AND type IN ('U'))

DROP TABLE [dbo].[file]

GO

CREATE TABLE [dbo].[file] (

[fileId] int IDENTITY(1,1) NOT NULL,

[filePath] text COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS NULL,

[SN] int NULL,

[upTime] datetime DEFAULT (getdate()) NULL

)

GO

ALTER TABLE [dbo].[file] SET (LOCK\_ESCALATION = TABLE)

GO

EXEC sp\_addextendedproperty

'MS\_Description', N'文件ID',

'SCHEMA', N'dbo',

'TABLE', N'file',

'COLUMN', N'fileId'

GO

EXEC sp\_addextendedproperty

'MS\_Description', N'文件路径',

'SCHEMA', N'dbo',

'TABLE', N'file',

'COLUMN', N'filePath'

GO

EXEC sp\_addextendedproperty

'MS\_Description', N'上传用户ID',

'SCHEMA', N'dbo',

'TABLE', N'file',

'COLUMN', N'SN'

GO

EXEC sp\_addextendedproperty

'MS\_Description', N'文件上传时间',

'SCHEMA', N'dbo',

'TABLE', N'file',

'COLUMN', N'upTime'

GO

-- ----------------------------

-- Records of file

-- ----------------------------

BEGIN TRANSACTION

GO

SET IDENTITY\_INSERT [dbo].[file] ON

GO

SET IDENTITY\_INSERT [dbo].[file] OFF

GO

COMMIT

GO

-- ----------------------------

-- Table structure for UserInfo

-- ----------------------------

IF EXISTS (SELECT \* FROM sys.all\_objects WHERE object\_id = OBJECT\_ID(N'[dbo].[UserInfo]') AND type IN ('U'))

DROP TABLE [dbo].[UserInfo]

GO

CREATE TABLE [dbo].[UserInfo] (

[SN] int IDENTITY(1,1) NOT NULL,

[UserName] nchar(20) COLLATE Chinese\_PRC\_CI\_AS NOT NULL,

[UserPass] nchar(16) COLLATE Chinese\_PRC\_CI\_AS NOT NULL,

[RegTime] datetime DEFAULT (getdate()) NULL,

[Email] char(30) COLLATE Chinese\_PRC\_CI\_AS NULL,

[GUID] uniqueidentifier DEFAULT (newid()) NULL

)

GO

ALTER TABLE [dbo].[UserInfo] SET (LOCK\_ESCALATION = TABLE)

GO

-- ----------------------------

-- Records of UserInfo

-- ----------------------------

BEGIN TRANSACTION

GO

SET IDENTITY\_INSERT [dbo].[UserInfo] ON

GO

INSERT INTO [dbo].[UserInfo] ([SN], [UserName], [UserPass], [RegTime], [Email], [GUID]) VALUES (N'1', N'Kevin ', N'123 ', N'2019-09-19 15:50:58.093', N'135@126.com ', N'922B79CA-F59F-4FD2-B5D2-F56D2FA88413')

GO

INSERT INTO [dbo].[UserInfo] ([SN], [UserName], [UserPass], [RegTime], [Email], [GUID]) VALUES (N'2', N'123 ', N'14564 ', N'2019-09-19 15:45:42.803', N'156@.1 ', N'190CE20F-68B1-48BC-8976-AAC1936A93CE')

GO

INSERT INTO [dbo].[UserInfo] ([SN], [UserName], [UserPass], [RegTime], [Email], [GUID]) VALUES (N'3', N'15616 ', N'1261 ', N'2019-09-19 16:06:43.183', N'1216 ', N'CBD09E84-6870-4287-8D66-F4FC40AE426F')

GO

SET IDENTITY\_INSERT [dbo].[UserInfo] OFF

GO

COMMIT

GO

-- ----------------------------

-- Auto increment value for file

-- ----------------------------

DBCC CHECKIDENT ('[dbo].[file]', RESEED, 1)

GO

-- ----------------------------

-- Primary Key structure for table file

-- ----------------------------

ALTER TABLE [dbo].[file] ADD CONSTRAINT [PK\_\_file\_\_C2C6FFDCE40DAA97] PRIMARY KEY CLUSTERED ([fileId])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

ON [PRIMARY]

GO

-- ----------------------------

-- Auto increment value for UserInfo

-- ----------------------------

DBCC CHECKIDENT ('[dbo].[UserInfo]', RESEED, 49)

GO

-- ----------------------------

-- Primary Key structure for table UserInfo

-- ----------------------------

ALTER TABLE [dbo].[UserInfo] ADD CONSTRAINT [PK\_Table\_1] PRIMARY KEY CLUSTERED ([SN])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

ON [PRIMARY]

GO

-- ----------------------------

-- Foreign Keys structure for table file

-- ----------------------------

ALTER TABLE [dbo].[file] ADD CONSTRAINT [FK\_\_file\_\_SN\_\_5441852A] FOREIGN KEY ([SN]) REFERENCES [dbo].[UserInfo] ([SN]) ON DELETE NO ACTION ON UPDATE NO ACTION

GO

# 应用程序设计

## 前言

在 Web 应用系统开发中，经常须要上传文件保存到服务器上，因此文件的上传和下载功能在网站中是经常被需要使用到的功能。本次课程设计选用C#语言整合ASP.NET框架，数据库选用SQL Server进行数据管理。

Web客户端页面对文件的上传或者下载操作是ASP.NET开发过程中经常会用到的一项技术。这项技术可以给用户带来非常方便的文件操作, 可以直接在网页上进行文件的上传或者下载。通过Web网页上传或者下载的文件一般为除文件夹以外的其他文件, 例如压缩文件、Word文档、Excel文件、EXE应用文件等等。所以在文件进行上传时, 要先对文件进行前期的处理, 使之成为可以上传的非文件夹文件。经过处理后的文件才可以进行上传, 否则进行上传的文件将无法进行上传。

本系统主要利用ASP.NET技术进行开发，有很强的逻辑性、可扩展性，便于维护。物业管理信息系统设计主要使用了CSS进行聊天界面的美化。其次本系统设计还使用了 SQL Server 、CSS、JavaScript、jQuery、AJAX等技术。

## 概要设计

1. 基于访问控制的文件上传系统简要功能介绍

本次课程设计提供的简易文件上传管理信息系统是可以适用于各类系统系统的一个子功能模块，能基本实现管理员用户登录和文件上传及回显的功能。

1. 用户登录注册

功能描述：面向用户提供的登录注册功能；

1. 文件上传管理

功能描述：面向已登录用户提供的文件上传及上传成功后显示的功能；

## 特定模块的详细介绍

### 分层分析

1. 驱动和中间介层
   1. C#与SQL Server数据库连接

对于不同的.NET数据提供者，ADO.NET采用不同的Connection对象连接数据库。这些Connection对象为我们屏蔽了具体的实现细节，并提供了一种统一的实现方法。

Connection类有四种：SqlConnection，OleDbConnection，OdbcConnection和OracleConnection。

SqlConnection类的对象连接SQL Server数据库；OracleConnection 类的对象连接Oracle数据库；OleDbConnection类的对象连接支持OLE DB的数据库，如Access；而OdbcConnection类的对象连接任何支持ODBC的数据库。与数据库的所有通讯最终都是通过Connection对象来完成的。

SqlConnection类

Connection 用于与数据库“对话”，并由特定提供程序的类（如 SqlConnection）表示。尽管SqlConnection类是针对Sql Server的，但是这个类的许多属性、方法与事件和OleDbConnection及OdbcConnection等类相似。

注意：使用不同的Connection对象需要导入不同的命名空间。OleDbConnection的命名空间为System.Data.OleDb。SqlConnection的命名空间为System.Data.SqlClient。OdbcConnection的命名空间为System.Data.Odbc。OracleConnection的命名空间为System.Data.OracleClinet。

我们就可以使用如下两种方式连接数据库，即采用集成的Windows验证和使用Sql Server身份验证进行数据库的登录。

* + 1. 集成的Windows身份验证语法范例

string constr = "server=.;database=myschool;integrated security=SSPI";

说明：程序代码中，设置了一个针对Sql Server数据库的连接字符串。其中server表示运行Sql Server的计算机名，由于程序和数据库系统是位于同一台计算机的，所以我们可以用.(或localhost)取代当前的计算机名。database表示所使用的数据库名(myschool)。由于我们希望采用集成的Windows验证方式，所以设置 integrated security为SSPI即可。

* + 1. Sql Server 2005中的Windows身份验证模式如下：

string constr = "server=.;database=myschool;uid=sa;pwd=sa";

说明：程序代码中，采用了使用已知的用户名和密码验证进行数据库的登录。数据库连接字符串是不区分大小写的。uid为指定的数据库用户名，pwd为指定的用户口令。为了安全起见，一般不要在代码中包括用户名和口令，你可以采用前面的集成的Windows验证方式或者对Web.Config文件中的连接字符串加密的方式提高程序的安全性。

* + 1. Sql Server 2005中的Sql Server身份验证模式如下：

string constr = "data source=.;initial catalog=myschool;user id=sa;pwd=sa";

说明：程序代码中data source 表示运行数据库对应的计算机名，initial catalog表示所使用的数据库名。uid为指定的数据库用户名，pwd为指定的用户口令。

* + 1. Access数据库的连接字符串的形式如下：

string connectionString =@"provider=Microsoft.Jet.OLEDB.4.0;data source=c:\DataSource\myschool.mdb";

* + 1. Web.config 配置

<?xml version="1.0"?>

<configuration>

<connectionStrings>

<add name="myschool" connectionString="Server=localhost;Integrated Security=True;Database=myschool;Persist Security Info=True" providerName="System.Data.SqlClient" />

</connectionStrings>

<system.web>

<pages styleSheetTheme="Default"/>

</system.web>

</configuration>

在ASP.NET 2.0中，使用了一种在运行时解析为连接字符串值的新的声明性表达式语法，按名称引用数据库连接字符串。连接字符串本身存储在 Web.config 文件中的 ＜connectionStrings＞配置节下面，以便易于在单个位置为应用程序中的所有页进行维护。

我们也可以用下面的方式从配置文件直接读取数据库连接字符串。首先我们需要引用using System.Web.Configuration命名空间，该命名空间包含用于设置 ASP.NET 配置的类。string connectionString =ConfigurationManager.ConnectionStrings["myschool"].ConnectionString;

首先你应该区分Windows验证与SQL自身的验证的区别。

Windows验证就是SQL Server服务器使用Windows自带的验证系统，如果你指定SQL Server内Windows的一个组有访问的权限，那么加入此组的Windows用户都有访问数据库的权限。此验证有个缺点，就是如果不是在域模式下，无法加入远程计算机的用户，所以如果使用C/S方式写程序的话，使用Windows验证无法使本地计算机的Windows帐户访问远程数据库服务器。

SQL验证就简单多了，就是使用SQL Server的企业管理器中自己定义由SQL控制的用户，指定用户权限等。这个帐户信息是由SQL Server自己维护的，所以SQL Server更换计算机后信息不会丢失，不用重新设定。

所以如果你的项目使用在一个比较大的网络中，而且对安全要求比较高，那么应该建立域，使用Windows验证，而且要与系统管理员配合详细设定可以访问SQL Server的Windows帐户。如果使用一个小网络，而且此网络仅用来使用项目，对安全没有高要求，那么使用SQL Server验证，而且更新，升级等都方便。

Windows验证与SQL Server验证的数据库联接字符串是不同的。

* 1. SQL Server数据库

SQL Server 是作为一个服务器的数据库系统使用的，并非桌面系统，比如 MS Access。

SQL Server 被设计为在中央服务器上运行（或服务器），使多个使用者可以同时访问相同的数据；用户通常通过应用程序访问数据库。

例如，一个网页可以存储在数据库中的所有内容。当访问者浏览的文章，他们从数据库中检索数据。一个网站服务的对象达到了数百，甚至数千的访客。同时，还得满足其他用户可以更新他们的个人资料，会员区，还能订阅新闻简报或其他任何网站的用户操作。所以有很多用户都是同时读取和更新数据库的，那么一个良好的，强大的数据库系统满足这种类型的用法。毕竟，你不希望你的数据库锁定，因为太多用户试图访问它，或者更糟的是，你不会希望在你的数据库，由于电线损坏而致数据获取不到。

通常，这是是由网站的应用程序提供的功能，以这些访问者（例如，可以使用如，ColdFusion，HTML和JavaScript来构建网站）。它使用数据库存储数据，并使其可用。但是，SQL Server不包括一些有用的功能，可帮助应用程序提供的功能。

1. ASP.NET 框架层

ASP.NET 是.NET FrameWork的一部分，是一项微软公司的技术，是一种使嵌入网页中的脚本可由因特网服务器执行的服务器端脚本技术，它可以在通过HTTP请求文档时再在Web服务器上动态创建它们。 指 Active Server Pages(动态服务器页面) ，运行于 IIS(Internet Information Server 服务，是Windows开发的Web服务器)之中的程序 。

1. 开发工具软件
   1. Visual Studio IDE

Visual Studio 集成开发环境是一种创新启动板，可用于编辑、调试并生成代码，然后发布应用。集成开发环境 (IDE) 是一个功能丰富的程序，可用于软件开发的许多方面。除了大多数 IDE 提供的标准编辑器和调试器之外，Visual Studio 还包括编译器、代码完成工具、图形设计器和许多其他功能，以简化软件开发过程。

* 1. JetBrains Rider

JetBrains Rider是一款快速强大的C#编辑器，也就是俗称的IDE。有2500多项的智能代码检查与重构，更加迅速编写和零误差的代码。完美集成与Unity的双向通信，无需离开Rider编辑器，就可以开始、暂停、逐帧运行，允许与Unity的API交互测试，Logs工具窗口，有指向堆栈的链接，可以浏览事件的消息、警告、错误，可以轻松导航到源文件、类、方法、属性。提供有C#代码分析，包括Unity的代码检查和修复。让我们一起来学习一下吧。

1. Model和View层
   1. Model和View层的关系

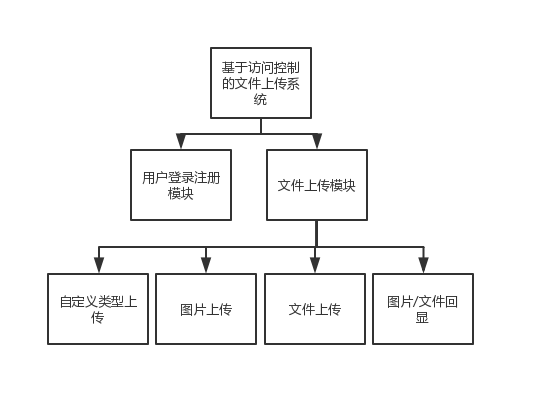


图 4 Model和View层的关系E-R图

1. Constant层

该层根据阿里巴巴代码设计在应用模块包下建立constant包，建立ConstConfig程序类存放应用级共享常量，这将可以消除编码过程中可能出现的绝大多数魔法值，使应用返回信息统一，实现一处修改多处同步，使程序复杂度降低。

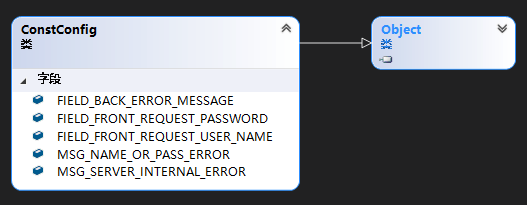


图 5 Constant层ConstConfig类图

1. DAO层

用户信息表：用来存储用户信息，如用户ID、用户账户、用户密码、用户注册时间、身份证号码、用户姓名、用户性别等。

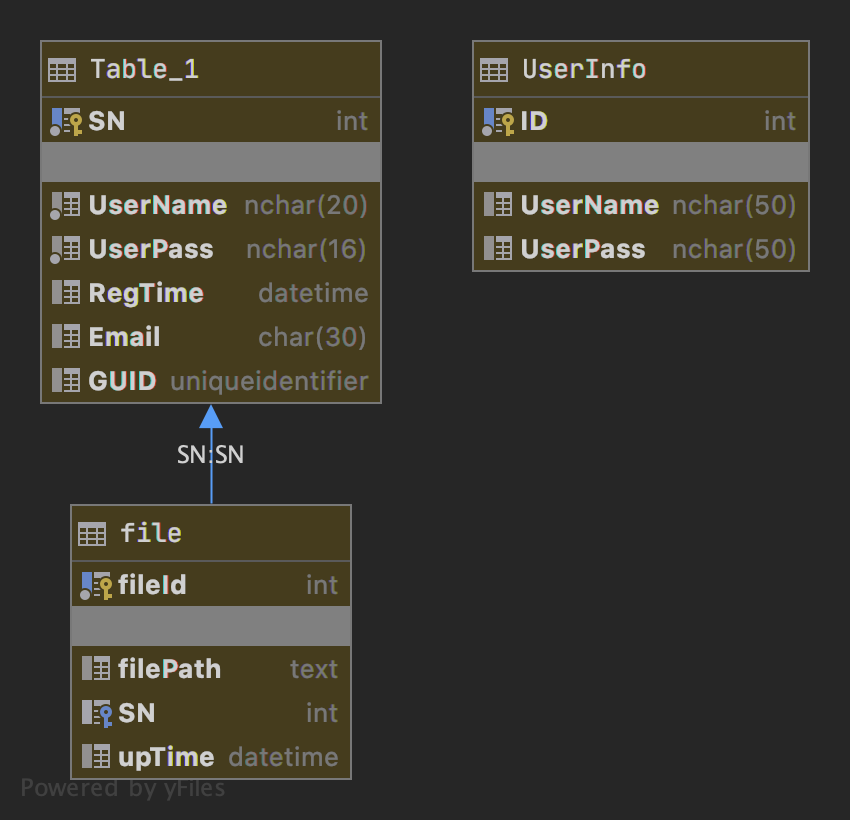


图 6 Entity实体类

1. E-R图

以上表都是独立的实体，这些实体的 E-R 图设计为：

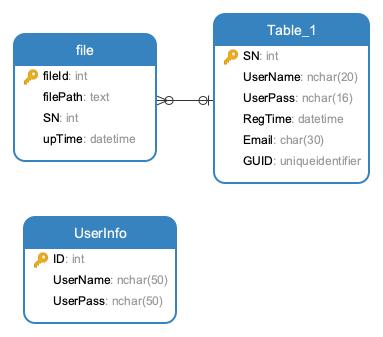


图 7实体E-R图

### 详细设计

1. 服务端设计
   1. 应用级常量
      1. ConstConfig.java
         1. 前台回传/请求数据字段（input标签name属性名）名

|  |  |
| --- | --- |
| 常量名 | 常量值 |
| FIELD\_FRONT\_REQUEST\_USER\_NAME | "userName" |
| FIELD\_FRONT\_REQUEST\_PASSWORD | "userPass" |
| FIELD\_BACK\_ERROR\_MESSAGE | "errMsg" |

* + - 1. 后台向前台发送的回显消息

|  |  |
| --- | --- |
| 常量名 | 常量值 |
| MSG\_NAME\_OR\_PASS\_ERROR | "用户名或密码错误" |
| MSG\_SERVER\_INTERNAL\_ERROR | "服务器内部错误" |

* 1. Path image

此路径是用户上传文件的服务器存储路径，用户上传的文件都会存储于此。

* 1. Path web
     1. Login.ashx
        1. 功能描述

此一般处理程序用于接受来自Login.html页面提交的参数，并查询数据库与数据库中的数据进行比对，核验用户身份信息，核验通过放行，用户页面跳转至UpFile.html页面；核验不通过用户页面返回Login.html页面，并显示错误提示信息。

* + 1. UpFile.ashx
       1. 功能描述

此一般处理程序用于接受来自用户上传的文件，并进行文件合规性检查，合规的文件将被服务器保留，存储于本课程设计 --> 详细设计 --> Path image中指定的路径；不合规的文件将被服务器丢弃。上述操作完成后，程序将给出返回参数信息，并交由back.html页面回显，服务器给出的提示，成功的请求将会在back.html页面显示上传的内容，失败的请求将会提示失败信息。

* 1. 前段HTML页面
     1. Login.html
        1. 功能描述

用户在此页面输入用户名和密码，此页面将提交给Login.ashx进行用户身份信息核验，核验通过放行，用户页面跳转至UpFile.html页面；核验不通过用户页面返回Login.html页面，并显示错误提示信息。

* + - 1. 运行截图

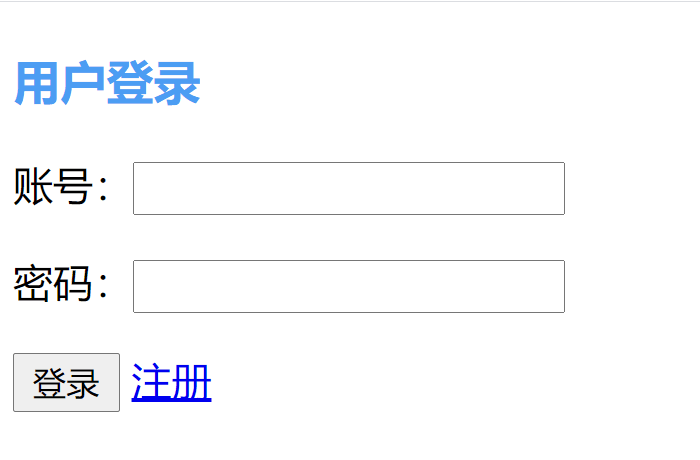


图 8 Login.html页面运行截图

* + 1. UpFile.html
       1. 功能描述

用户在此页面上传文件，此页面将提交给UpFile.ashx进行文件合规性核验，核验通过的将被服务器存留，用户页面跳转至back.html页面，并回显用户上传内容；核验不通过用户页面重定向back.html页面，并显示错误提示信息。

* + - 1. 运行截图

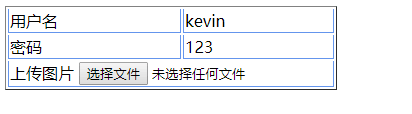


图 9 UpFile.html页面运行截图

# 六、附：页面前端程序代码

## 用户登录页面

Login.html页面代码：

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title></title>

</head>

<body>

<h1>User Login System</h1>

<form action="Login.ashx" method="post">

<p><label>账号：<input type="text" name="userName" value="" required /></label></p>

<p><label>密码：<input type="password" name="userPass" value="" required /></label></p>

<p><input type="submit" value=" Go！" style="color:red;border: dashed;"/></p>

<p><a href="Register.html">点我注册←</a></p>

</form>

</body>

</html>

用户登录一般处理程序（Login.ashx）代码：

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Web;

namespace WebApplication202007.App\_Code

{

/// <summary>

/// Login 的摘要说明

/// </summary>

public class Login : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{

context.Response.ContentType = "text/html";

int count = 0;

string connStr = System.Configuration.ConfigurationManager.ConnectionStrings["connStr"].ConnectionString;//找个配置文件中的数据库

using (SqlConnection conn = new SqlConnection(connStr))//打开数据库链接

{

using (SqlCommand cmd = new SqlCommand())

{ //新建储存对象

conn.Open();

cmd.Connection = conn;

cmd.CommandText = "select count (\*) from admin where UserName=@userName and UserPass=@userPass";

cmd.Parameters.Add(new SqlParameter("@userName", context.Request.Form["userName"]));

cmd.Parameters.Add(new SqlParameter("@userPass", context.Request.Form["userPass"]));

count = (int)cmd.ExecuteScalar();

if (count > 0)

{

//context.Response.Write("Landing successfully");

context.Response.Redirect("UpFile.html");

}

else

{

context.Response.Write("Login failed, wrong user name or password, or user does not exist");

}

}

}

}

public bool IsReusable

{

get

{

return false;

}

}

}

}

## 文件上传页面

UpFile.html页面代码：

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title></title>

</head>

<body>

<form action="UpFile.ashx" method="post"enctype="multipart/form-data">

<input type="file"value="" name="file" />

<input type="submit" value="Submit" />

</form>

</body>

</html>

文件上传一般处理程序（UpFile.ashx）代码：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.IO;

using System.Drawing;

namespace WebApplication202007

{

/// <summary>

/// UpFile 的摘要说明

/// </summary>

public class UpFile : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{

context.Response.ContentType = "text/plain";

HttpPostedFile file = context.Request.Files["file"];

if (file == null)

{

context.Response.Write("请选择要上传的文件");

}

else

{

string filename = Path.GetFileName(file.FileName);

string Extension = Path.GetExtension(filename);

if (Extension == ".jpg" || Extension == ".gif" || Extension == ".png")

{

//file.SaveAs(context.Request.MapPath("/Image/" + filename));

string dir = "/image/" + DateTime.Now.Year + "/" + DateTime.Now.Month + "/" + DateTime.Now.Day + "/";

Directory.CreateDirectory(Path.GetDirectoryName(context.Request.MapPath(dir)));

string guid = Guid.NewGuid().ToString();

string fileNewName = dir + guid + Extension;

//file.SaveAs(context.Request.MapPath(fileNewName));

//context.Response.Write("上传成功");

/// GDI+

using (Image img = Image.FromStream(file.InputStream))

{

using (Bitmap map = new Bitmap(img.Width, img.Height))

{

using (Graphics g = Graphics.FromImage(map))

{

g.DrawImage(img, new Rectangle(0, 0, map.Width, map.Height));

g.DrawString("湖北工业大学", new Font("微软雅黑", 36.0f, FontStyle.Italic), Brushes.Green, new PointF(map.Width \* 0.5f, map.Height \* 0.8f));

map.Save(context.Request.MapPath(fileNewName));

context.Response.Write("上传成功");

}

}

}

}

else

{

context.Response.Write("请选择图片类型上传");

}

}

}

public bool IsReusable

{

get

{

return false;

}

}

}

}

## 远程数据库连接（web.config）代码片段：

<connectionStrings>

<add name="connStr" connectionString="server=bj.kevinkda.cn;uid=kevin;pwd=Karry0921;database=WebProData"/>

</connectionStrings>

## 系统的测试

软件测试的目的和原则

在开发软件系统的过程中，我们需要面对错综复杂的问题，不可避免的在每个生产阶段都会产生一些错误，我们力求进行严格的技术审查在每个阶段任务结束前尽可能多的发现和纠正软件中的错误。而软件测试的目的就是在软件投入生产环境前尽可能的修正软件中的错误，目前软件测试仍然是软件质量控制的最后防线。

软件测试分以下三种类型：

1. 单元测试：单元测试的用例从单元详细设计中导出。在单元测试中可以采用功能性测试和结构性测试两种；
2. 集成测试：在这个测试步骤中所发现的往往是概要设计的错误；
3. 系统测试：在这个测试步骤中所发现的往往是需求规格说明的错误。一般来说，系统测试是功能性测试，不是结构性测试。

本次设计进行了如下调试和测试，并提出相应的解决方案：

1. 访问失败

检查浏览器访问的路径是否为 http://127.0.0.1:8097/estate/ ；

1. 前端脚本失效

检查是不是有语法错误导致后面的脚本全被屏蔽了；

1. 数据库连接出错

检查 SQL Server版本；检査数据库服务监听是否成功开启；检查数据库 URL、用户名密码等是否正确；

1. 配置文件加载失败

为web.config配置的内容是否保存，可能配置文件书写过程中是否出现语法错误；

1. C#类和数据库 Type 问题

需要查询这三种 type 的对应关系，印象最深的就是不同数据库和C#类的对应关系不一样。

# 七、结束语

这次ASP.NET应用课程设计是一门非常不错的课程，在此之前我写服务器基本都是自己造轮子自己写，但是这次课设使得我有机会使用现在还在使用ASP.NET架构进行开发，同时，这次实验对我来说是一个巨大的挑战。因为之前只是对ASP.NET框架有所耳闻，原先用 HTML的时候也是自己从GET和 POST处理函数为入口自己写处理，所以一开始对于ASP.NET的写法还不是很熟练。但是，经过持续约一个月坚持不懈的努力，我终于完成了本次的实验任务。

当然在开发的过程中也遇到了很多问题。老师曾说，对于我们现在接触到的困难，之前肯定有人接触过，所以要好好利用互联网。课程设计，从安装 SQL Server，到配置数据库，到使用程序代码进行数据的增删改查，每步都曾遇到很棘手的问题，多亏了老师的帮助，也感谢在社区和论坛上帮我解决问题的朋友们。

采用B/S结构与ASP.NET框架。动态 WEB 中，程序依然使用客户端和服务端，客户端依然使用浏览器（Chrome、Safari、Firefox 等），通过网络（Network）连接到服务器上，使用 HTTP 协议发起请求（Request），都交由Internet Information Services（IIS，互联网信息服务）来处理，应用程序开发人员只用专心设计程序代码，不用处理IIS的内部响应。

当然，这次实验的另外一个体会就是，时间安排非常重要。老师在很早的时候就把任务书发了下来，但是当时我没在意，所以在之后做得非常紧。由此了解到，做好一个项目，每一阶段的时间都必须规划好。因为我们学生现阶段采用开发模式基本都是瀑布开发模式，就必须保证每一步的正确性和及时性。总之，本次设计锻炼了实践能力，増长了我发现问题解决问题的能力，大大丰富了自己的专业知识和技能。由于各种原因，系统的功能还没有得到进一步的完善，以后会继续努力，在今后的学习过程中，不断地提高自己。