# MD5

MD5（Message Digest Algorithm）算法是一种散列（hash）算法（摘要算法，指纹算法），不是一种加密算法。任何长度的任意内容都可以用MD5计算出散列值。

主要作用：验明“真身”，确保信息传输过程中的完整性、一致性。

MD5的长度：由32位16进制数字字符串组成。MD5值的个数是有限的，但是源数据是无限的，因此存在不同的内容产生相同MD5值的概率。

## MD5算法的特点

* 长度固定：32位16进制数字字符串
* 单向性：不可逆向反推出原来的内容。
* 抗冲突性：理论上不重复
* “雪崩效应”：只要源发生一点变化，就会造成很大的变化。
* 相同的源产生的总是相同的MD5值。

### 补充：散列法

散列法（Hashing）或哈希法是一种将字符组成的字符串转换为固定长度（一般是更短长度）的数值或索引值的方法，称为散列法，也叫哈希法。由于通过更短的哈希值比用原始值进行数据库搜索更快，这种方法一般用来在数据库中建立索引并进行搜索，同时还用在各种解密算法中。

#### 定义

比如，在数据库中存储一些人名，排列方式可能是下面这样：

Abernathy, Sara

Epperdingle, Roscoe

Moore, Wilfred

Smith, David

所有名字均按字母排序......

可以利用这些名字本身来作为数据库的索引值。数据库搜索算法首先会逐个字符的进行名字的搜索，直到找到为止。但是如果利用散列法对每个名字进行了转换，就可能为数据库中的每一个名字产生一个四位的索引值，其中位数长度取决于数据库中到底有多少个人名，象下面这样：

7864 Abernathy, Sara

9802 Epperdingle, Roscoe

1990 Moore, Wilfred

8822 Smith, David

等等......

这样，下次搜索名字时，就先搜索哈希并对数据库中的每个值进行一一对应。通常来讲，寻找四位的数字比寻找未知长度的字符串要来得快得多。毕竟寻找数字时每一位只有10种可能，而名字的长度未定，且每一位都有26种可能。

#### 散列算法

散列算法也称为哈希函数——哈希的英文意思为“无用信息”，因此哈希函数一词的由来可能是因为最终形成的哈希表里面是各种看起来毫无意义的描述值的混合。除用来快速搜索数据外，散列法还用来完成签名的加密解密工作，这种签名可以用来对收发消息时的用户签名进行鉴权。先用哈希函数对数据签名进行转换，然后将数字签名本身和转换后的信息摘要分别独立的发送给接收人。通过利用和发送人一样的哈希函数，接收人可以从数字签名获得一个信息摘要，然后将此摘要同传送过来的摘要进行比较，这两个值相等则表示数字签名有效。

利用哈希函数对数据库中的原始值建立索引，以后每获取一次数据时都要利用哈希函数进行重新转换。因此，哈希函数始终是单向操作。没有必要通过分析哈希值来试图逆推哈希函数。实际上，一个典型的哈希函数是不可能逆推出来的。好的哈希函数还应该避免对于不同输入产生相同的哈希值的情况发生。如果产生了哈希值相同的情况，称为冲突。可接受的哈希函数应该将冲突情况的可能性降到非常小。

#### 一些相对简单的哈希函数

1）余数法：先估计整个哈希表中的表项目数目大小。然后用这个估计值作为除数去除每个原始值，得到商和余数。用余数作为哈希值。因为这种方法产生冲突的可能性相当大，因此任何搜索算法都应该能够判断冲突是否发生并提出取代算法。

2）折叠法：这种方法是针对原始值为数字时使用，将原始值分为若干部分，然后将各部分叠加，得到的最后四个数字（或者取其他位数的数字都可以）来作为哈希值。

3）基数转换法：当原始值是数字时，可以将原始值的数制基数转为一个不同的数字。例如，可以将十进制的原始值转为十六进制的哈希值。为了使哈希值的长度相同，可以省略高位数字。

4）数据重排法：这种方法只是简单的将原始值中的数据打乱排序。比如可以将第三位到第六位的数字逆序排列，然后利用重排后的数字作为哈希值。

哈希函数并不通用，比如在数据库中用能够获得很好效果的哈希函数，用在密码学或错误校验方面就未必可行。在密码学领域有几个著名的哈希函数。这些函数包括 MD2、MD4以及MD5，利用散列法将数字签名转换成的哈希值称为信息摘要（message－digest），另外还有安全散列算法（SHA），这是一种标准算法，能够生成更大的（60bit）的信息摘要，有点儿类似于MD4算法。

## 求字符串的MD5值

只需要两步：

* 创建MD5对象
* 用ComputerHash方法产生MD5.

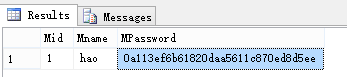
|  |
| --- |
| static void Main(string[] args)  {  while (true)  {  *Console*.*WriteLine*("输入字符串");  string str = *Console*.*ReadLine*();  string md5 = getMd5FromString(str);  *Console*.*WriteLine*(md5);  }  }  /// <summary>  /// 获取字符串的MD5值  /// </summary>  /// <param name="str"></param>  /// <returns></returns>  private static string getMd5FromString(string str)  {  //字符串转换为字节时的编码时UTF8  byte[] buffer = *Encoding*.*UTF8*.*GetBytes*(str);  byte[] md5Byte;  *StringBuilder* sb = new *StringBuilder*();  //1.创建一个用来计算Md5的类对象  using (*MD5* md5 = *MD5*.*Create*())  {  //2.计算给定字符串的MD5值。  md5Byte = md5.*ComputeHash*(buffer);  }  //返回的MD5值以16位展示。  for (int i = 0; i < md5Byte.*Length*; i++)  {  sb.*Append*(md5Byte[i].*ToString*("X2"));  }  return sb.*ToString*();  } |

## 求文件的MD5值

|  |
| --- |
| static void Main(string[] args)  {  string filepath = @"D:\Baidu Down loader\35----2013.11.04--三层\35\_20111104三层\简单三层资料\MD5\CalcMD5\CalcMD5.exe";  string str = getMd5FromFile(filepath);  *Console*.*WriteLine*(str);  *Console*.*ReadLine*();  }  private static string getMd5FromFile(string filepath)  {  byte[] Md5Byte;  *StringBuilder* sb = new *StringBuilder*();  using (*MD5* md5 = *MD5*.*Create*())  {  using (*FileStream* reader = *File*.*OpenRead*(filepath))  {  //使用ComputerHash的重载方法获取MD5值  Md5Byte = md5.*ComputeHash*(reader);  }  }  foreach (var item in Md5Byte)  {  sb.*Append*(item.*ToString*("X2"));  }  return sb.*ToString*();  } |

## MD5登录

首先设计数据库



### 设计HyfDictionary类

|  |
| --- |
| public class HyfDictionary:*Dictionary*<string,string>  {    } |

### 设计SqlHelper类

|  |
| --- |
| public static class HyfSqlHelper  {  /// <summary>  /// 连接数据库  /// </summary>  /// <param name="connStrname"></param>  /// <returns></returns>  private static *SqlConnection* getConn(string connStrname)  {  string connStr = *System*.*Configuration*.*ConfigurationManager*.*ConnectionStrings*[connStrname].*ConnectionString*;  return new *SqlConnection*(connStr);  }  //执行sql语句，返回首行首列。  public static object ExecuteScalar(string connStrname, string sql, params HyfDictionary[] hdict)  {  object obj = null;  using (*SqlConnection* conn = getConn(connStrname))  {  using (*SqlCommand* cmd = new *SqlCommand*(sql, conn))  {  if (hdict.*Length* > 0)  {  for (int i = 0; i < hdict.*Length*; i++)  {  foreach (var item in hdict[i])  {  *SqlParameter* sp = new *SqlParameter*(item.*Key*, item.*Value*);  cmd.*Parameters*.*Add*(sp);  }  }  }  conn.*Open*();  obj = cmd.ExecuteScalar();  }  }  return obj;  }  /// <summary>  /// 判断用户名是否存在。  /// </summary>  /// <param name="connStrname"></param>  /// <param name="Tablename"></param>  /// <param name="name"></param>  /// <returns></returns>  public static bool IsUserExist(string connStrname, string Tablename, string name)  {  bool isExist = false;  string sql = "select Count(\*) from " + Tablename + " where Mname = @name";  HyfDictionary hdic = new HyfDictionary();  hdic.*Add*("@name", name);  object obj = ExecuteScalar(connStrname, sql, hdic);  if ((int)obj != 0)  {  isExist = true;  }  return isExist;  }  } |

### 点击按钮，进行登录

界面如下：



代码如下：

|  |
| --- |
| private void button1\_Click(object sender, *EventArgs* e)  {  string name = txt\_name.*Text*.*Trim*();  string pwd = txt\_pwd.*Text*;  string connStraname = "Md5";  string tableName = "T\_Md5User";  pwd = GetMD5(pwd);  if (LoginStat(name, pwd, connStraname, tableName))  {  *MessageBox*.*Show*("登录成功！");  }  else  {  *MessageBox*.*Show*("密码错误！");  }  }  private static bool LoginStat(string name, string pwd, string connStraname, string tableName)  {  bool LoginStatus = false;  if (HyfSqlHelper.IsUserExist(connStraname, tableName, name))  {  string sql = "select MPassword from " + tableName + " where Mname=@name";  HyfDictionary hdic = new HyfDictionary();  hdic.*Add*("@name", name);  object obj = HyfSqlHelper.ExecuteScalar(connStraname, sql, hdic);  string Tpwd = obj.*ToString*();  if (Tpwd == pwd)  {  LoginStatus = true;  }  }  else  {  *MessageBox*.*Show*("用户名不存在");  }  return LoginStatus;  }  private static string GetMD5(string pwd)  {  byte[] bytesBuffer;  *StringBuilder* sb = new *StringBuilder*();  using (*MD5* m5 = *MD5*.*Create*())  {  bytesBuffer = *Encoding*.*Default*.*GetBytes*(pwd);  bytesBuffer = m5.*ComputeHash*(bytesBuffer);  }  for (int i = 0; i < bytesBuffer.*Length*; i++)  {  sb.*Append*(bytesBuffer[i].*ToString*("x2"));  }  return sb.*ToString*().*ToLower*();  } |

# NOPI

## Excel基础概念

工作簿（Workbook）：用来存储和处理数据的文件，其扩展名为.xlsx。

工作表（sheet）：用于存储各种数据的表格，包含若干行和列。

行（row）：工作表中横向上的数据组成行。

单元格（cell）：工作表中行与列形成的方格。

## Excel处理的技术

### OLE Automation

程序启动一个Excel进程，然后和Excel进程进行通讯来进行Excel的操作。

优点：强大，能够使用Excel的所有功能，要求装Excel，微软最推荐这种用法，因为可以促进Excel的销量。会启动Excel进程，不适合于服务器（比如Asp.Net网站，安全性、效率）。参考资料<http://topic.csdn.net/t/20031204/09/2525334.html>。

### Microsoft.Jet.OleDb

把Excel当成数据库，使用Microsoft.Jet.OleDb访问访问Excel ，参考资料 http://tieba.baidu.com/f?kz=331569890 只适合于完全二维结构，功能最弱，很少用。不用装Exce

### OpenXML

微软提供的读写Excel的技术，优点和NPOI差不多，不过只能处理xlsx格式文件。docx、pptx。

下载微软Open XML SDK for Microsoft Office。

### NPOI

NPOI、MyXls等，NPOI能够分析Excel文件的格式，能够进行常用Excel操作，不依赖于Excel，节省资源，没有安全性、性能的问题，在ASP.net中用最合适。只能处理xls格式文件、不能处理xlsx这样的新版本Excel文件格式。处理xlsx还要用OpenXML

## 写入Excel

|  |
| --- |
| private void 加载ExcelToolStripMenuItem\_Click(object sender, *EventArgs* e)  {  *OpenFileDialog* ofd = new *OpenFileDialog*();  ofd.*Filter* = "Excel03文件（\*.xls）|\*.xls|Excel07文件（\*.xlsx）|\*.xlsx";  ofd.*FilterIndex* = 1;  *DialogResult* dr = ofd.*ShowDialog*();  *StringBuilder* sb = new *StringBuilder*();  sb.*Clear*();  if (dr == *DialogResult*.*OK*)  {  //创建文件流，指向excel文件  using (*FileStream* fs = *File*.*OpenRead*(ofd.*FileName*))  {  //1. 根据文件流创建工作簿对象 。WorkbookFactory自动根据文件系统的类型返回HSSFWorkbook或XSSFWorkbook，  IWorkbook wb = WorkbookFactory.Create(fs);  //2. 获取工作表的个数：NumberOfSheets  for (int i = 0; i < wb.NumberOfSheets; i++)  {  //根据索引得到当前工作表  ISheet sheet = wb.GetSheetAt(i);  sb.*AppendLine*("======工作表： " + sheet.SheetName + "======");  //3. 获取当前工作表行的个数：LastRowNum  for (int j = 0; j <= sheet.LastRowNum; j++)  {  //获取当前行  IRow row = sheet.*GetRow*(j);  //如果行为空，终止当前工作表的循环  if (row == null)  {  break;  }  else  { //4. 遍历单元格，获得当前单元格的内容  for (int k = 0; k < row.LastCellNum; k++)  {  ICell cell = row.GetCell(k);  sb.*AppendFormat*("{0}\t", cell);  }  //单元格换行  sb.*AppendLine*();  }  }  }  }  }  txt\_show.*Text* = sb.*ToString*();  } |

## 创建excel

|  |
| --- |
| private void button1\_Click(object sender, *EventArgs* e)  {  *List*<Person> p = new *List*<Person>(){  new Person(){Name="秦始皇",Age=18,Email="bmy@163.com"},  new Person(){Name="汉武帝",Age=20,Email="hwd@h.com"},  new Person(){Name="唐太宗",Age=16,Email="ttz@t.com"}  };  //1. 创建XSSFWorkbook对象。即xlsx对象文件  IWorkbook wb = new *XSSFWorkbook*();  //2. 创建工作表  *ISheet* sheet = wb.CreateSheet("my1");  //3.创建行  for (int i = 0; i < 3; i++)  {  *IRow* row = sheet.*CreateRow*(i);  //4. 设置单元格的内容  row.*CreateCell*(0).*SetCellValue*(p[i].Name);  row.*CreateCell*(1).*SetCellValue*(p[i].Age);  row.*CreateCell*(2).*SetCellValue*(p[i].Email);  }  //5. 保存文件  using (*FileStream* writer = *File*.*OpenWrite*("1.xlsx"))  {  wb.Write(writer);  }  }  }  public class Person  {  public string Name { get; set; }  public int Age { get; set; }  public string Email { get; set; }  } |

## 读取数据库数据到excel

需要设置每个单元格的数据类型。

|  |
| --- |
| using *System*;  using *System*.*Collections*.*Generic*;  using *System*.*Linq*;  using *System*.*Text*;  using *System*.*Threading*.*Tasks*;  using HyfSqlHelper;  using *System*.*Data*;  using *System*.*IO*;  using NPOI.SS.UserModel;  using NPOI.XSSF.UserModel;  namespace test  {  class Program  {  static void Main(string[] args)  {  //1.读取数据  SqlHelper sh = new MSSqlHelper("test");  string sql = "select \* from t\_userinfo";  SqlParamsDictionary spdic = new SqlParamsDictionary();  *DataTable* dt = sh.GetDataTable(sql);  using (*FileStream* fswriter = *File*.*OpenWrite*("test.xlsx"))  {  *IWorkbook* wb = new *XSSFWorkbook*();  *ISheet* sheet = wb.*CreateSheet*();  for (int i = 0; i < dt.*Rows*.*Count*; i++)  {  //转换为确定的数据类型  *IRow* row = sheet.*CreateRow*(i);  int c0 = (int)dt.*Rows*[i][0];  string c1 = (string)dt.*Rows*[i][1];  string c2 = (string)dt.*Rows*[i][2];  int c3 = (int)dt.*Rows*[i][3];  //可以为空的值  *DateTime*? c4 = dt.*Rows*[i][4] == null ? null : (*DateTime*?)dt.*Rows*[i][4];  row.*CreateCell*(0).*SetCellValue*(c0);  row.*CreateCell*(1).*SetCellValue*(c1);  row.*CreateCell*(2).*SetCellValue*(c2);  row.*CreateCell*(3).*SetCellValue*(c3);  *ICell* cell4 = row.*CreateCell*(4);  if (c4 == null)  {  cell4.*SetCellType*(*CellType*.*Blank*);  }  else  {  cell4.*SetCellValue*((*DateTime*)c4);  //设置单元格格式  ICellStyle cellStyle = wb.*CreateCellStyle*();  IDataFormat format = wb.*CreateDataFormat*();  //设置数据格式  cellStyle.DataFormat = format.GetFormat("yyyy-mm-dd h:mm");  //赋值  cell4.*CellStyle* = cellStyle;  }  }  wb.*Write*(fswriter);  }  *Console*.*WriteLine*("读取成功");  *Console*.*Read*();  }  }  } |

## 读取Excel数据到数据库

|  |
| --- |
| using (*FileStream* fsreader = *File*.*OpenRead*("test.xlsx"))  {  //1.创建工作簿对象  *IWorkbook* wb = *WorkbookFactory*.*Create*(fsreader);  //2. 获取工作表  *ISheet* sheet = wb.*GetSheetAt*(0);  string sql = "insert into T\_UserInfo values(@name,@password,@Errortimes,@LastErrorTime)";  SqlParamsDictionary spdic = new SqlParamsDictionary();  SqlHelper sqlhelper = new MSSqlHelper("test");  //3. 遍历行  for (int i = 0; i <= sheet.*LastRowNum*; i++)  {  *IRow* row = sheet.*GetRow*(i);  //获取当前行单元格的值  string name = row.*GetCell*(1).*StringCellValue*;  string password = row.*GetCell*(2).*StringCellValue*;  int Errortimes = (int)row.*GetCell*(3).*NumericCellValue*;  *ICell* cell = row.*GetCell*(4);  //空值的设定  double? LastErrorTime = null;  //单元格为空或值为null  if (cell != null && cell.*CellType* != *CellType*.*Blank*)  {  LastErrorTime = row.*GetCell*(4).*NumericCellValue*;  }  //sql语句参数的构造  spdic.*Clear*();  spdic.*Add*("@name", name);  spdic.*Add*("@password", password);  spdic.*Add*("@Errortimes", Errortimes.*ToString*());  spdic.*Add*("@LastErrorTime", LastErrorTime == null ? *DBNull*.*Value*.*ToString*() : *DateTime*.*FromOADate*((double)LastErrorTime).*ToString*());  sqlhelper.ExecuteNonQuery(sql, spdic);  }  } |

# 汉语转拼音

这篇文章主要介绍了C#中实现输入汉字获取其拼音（汉字转拼音）的2种方法,本文分别给出了使用微软语言包、手动编码实现两种实现方式,需要的朋友可以参考下

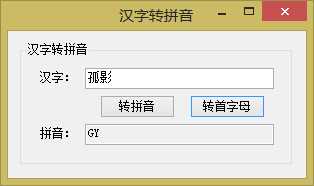
前不久看到有的朋友实现对商品名称拼音的录入，发现他的实现方式是手动输入的，—\_—#、同志们，福利来了！

本文刚发布时，只写了一个实现方式，使用的是微软的语言包，但是对多音字的效果不怎么理想，甚至个别字会出现很诡异的错误，因此，现在扩展另一个方法，手动实现。

方式一、使用微软语言包

微软为了开发者实现国际化语言的互转，提供了Microsoft Visual Studio International Pack，这个扩展包里面有中文、日文、韩文、英语等各国语言包，并提供方法实现互转、获取拼音、获取字数、甚至获取笔画数等等。  
[这种方式对多音字的效果不怎么理想，但是，这个方法比较简单，直接导入包就可以了，因此，对于那些只需要个别语句进行处理的或者不注重多音字的，可以使用这种方式，毕竟简便嘛。]

在这里示例讲的是输入汉字，获取其拼音，获取拼音和获取拼音首字母实现效果分别如下：



首先，去微软官网下载Microsoft Visual Studio International Pack语言包，下载地址分别如下：

[Microsoft Visual Studio International Pack 1.0 SR1](http://www.microsoft.com/zh-cn/download/details.aspx?id=15251)、[Microsoft Visual Studio International Feature Pack 2.0](http://%20http//www.microsoft.com/zh-cn/download/details.aspx?id=18970" \t "_blank)

下载后分别是“vsintlpack1.zip”、“Vsintlpack2.msi”、双击“Vsintlpack2.msi”安装、路径随意、但是要记得、因为一会要引用的、

　安装“Vsintlpack2.msi”之后、解压“vsintlpack1.zip”、里面包含七个语言包、  
　例如中文转拼音“CHSPinYinConv.msi”、简体繁体互转“CHTCHSConv.msi”等等。。

　在这里我们用到的是“CHSPinYinConv.msi”、双击安装成功后、打开Visual Studio、新建一个WinForm项目、窗体布局如上图所示、

首先：添加刚刚安装的语言包引用：

复制代码代码如下:

“D:\Program Files (x86)\Microsoft Visual Studio International Pack\Simplified Chinese Pin-Yin Conversion Library\ChnCharInfo.dll”

默认是C盘、在这里我安装在D盘了，然后添加using引用：

1 using Microsoft.International.Converters.PinYinConverter;//导入拼音相关

创建获取拼音的方法：

复制代码代码如下:

/// <summary>   
/// 汉字转化为拼音  
/// </summary>   
/// <param name="str">汉字</param>   
/// <returns>全拼</returns>   
public static string GetPinyin(string str)  
{  
    string r = string.Empty;  
    foreach (char obj in str)  
    {  
        try  
        {  
            ChineseChar chineseChar = new ChineseChar(obj);  
            string t = chineseChar.Pinyins[0].ToString();  
            r += t.Substring(0, t.Length - 1);  
        }  
        catch  
        {  
            r += obj.ToString();  
        }  
    }  
    return r;  
}

创建获取汉字拼音首字母的方法：

复制代码代码如下:

/// <summary>   
/// 汉字转化为拼音首字母  
/// </summary>   
/// <param name="str">汉字</param>   
/// <returns>首字母</returns>   
public static string GetFirstPinyin(string str)  
{  
    string r = string.Empty;  
    foreach (char obj in str)  
    {  
        try  
        {  
            ChineseChar chineseChar = new ChineseChar(obj);  
            string t = chineseChar.Pinyins[0].ToString();  
            r += t.Substring(0, 1);  
        }  
        catch  
        {  
            r += obj.ToString();  
        }  
    }  
    return r;  
}

然后在“转拼音”按钮的点击事件中调用上述方法：

复制代码代码如下:

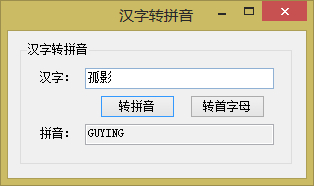
// 汉字转拼音  
private void btn\_One\_Click(object sender, EventArgs e)  
{  
    string source = this.txt\_ChineseCharacter\_One.Text.Trim();  // 得到输入的源字符  
    string result = GetPinyin(source);  // 调用方法，获取拼音  
    this.txt\_Pinyin\_One.Text = result;  
}

在“转首字母”按钮点击事件中调用上述方法：

复制代码代码如下:

// 转首字母  
private void btn\_Two\_Click(object sender, EventArgs e)  
{  
    string source = this.txt\_ChineseCharacter\_One.Text.Trim();  // 得到输入的源字符  
    string result = GetFirstPinyin(source);  // 调用方法，获取拼音  
    this.txt\_Pinyin\_One.Text = result;  
}

到此，已经完成了80%，运行程序，你会发现，当点击“转拼音”的时候，结果是这样子的：



并不是我开始说的那种“Gu Ying”的效果啊、这是因为我在获取拼音的时候简单的处理了一下：

复制代码代码如下:

// 汉字转拼音  
private void btn\_One\_Click(object sender, EventArgs e)  
{  
    string source = this.txt\_ChineseCharacter\_One.Text.Trim();  // 得到输入的源字符

    string result = string.Empty;   // 转拼音的结果  
    string temp = string.Empty; // 下面foreach用到的临时变量  
    foreach (char item in source)   // 遍历每个源字符  
    {  
        temp = GetPinyin(item.ToString());  // 将每个字符转拼音  
        // 处理：获取首字母大写、其余字母小写  
        result += (String.Format("{0}{1} ", temp.Substring(0, 1).ToUpper(), temp.Substring(1).ToLower()));  
    }

    //string result = GetPinyin(source);  // 调用方法，获取拼音  
    this.txt\_Pinyin\_One.Text = result;  
}

　OK、到此、这个功能已经实现完成了，还有其余的语言包功能，和此类似，大家可以百度“Microsoft Visual Studio International Pack使用”、各种语言之间的互转及功能示例就出来了。

方式二、手动编码实现

这种方式其实也不困难，说白了就是根据Unicode编码值，定义对应的拼音数组或集合，然后实现此效果。

首先定义拼音区编码数组：

复制代码代码如下:

//定义拼音区编码数组  
private static int[] getValue = new int[]  
    {  
        -20319,-20317,-20304,-20295,-20292,-20283,-20265,-20257,-20242,-20230,-20051,-20036,  
        -20032,-20026,-20002,-19990,-19986,-19982,-19976,-19805,-19784,-19775,-19774,-19763,  
        -19756,-19751,-19746,-19741,-19739,-19728,-19725,-19715,-19540,-19531,-19525,-19515,  
        -19500,-19484,-19479,-19467,-19289,-19288,-19281,-19275,-19270,-19263,-19261,-19249,  
        -19243,-19242,-19238,-19235,-19227,-19224,-19218,-19212,-19038,-19023,-19018,-19006,  
        -19003,-18996,-18977,-18961,-18952,-18783,-18774,-18773,-18763,-18756,-18741,-18735,  
        -18731,-18722,-18710,-18697,-18696,-18526,-18518,-18501,-18490,-18478,-18463,-18448,  
        -18447,-18446,-18239,-18237,-18231,-18220,-18211,-18201,-18184,-18183, -18181,-18012,  
        -17997,-17988,-17970,-17964,-17961,-17950,-17947,-17931,-17928,-17922,-17759,-17752,  
        -17733,-17730,-17721,-17703,-17701,-17697,-17692,-17683,-17676,-17496,-17487,-17482,  
        -17468,-17454,-17433,-17427,-17417,-17202,-17185,-16983,-16970,-16942,-16915,-16733,  
        -16708,-16706,-16689,-16664,-16657,-16647,-16474,-16470,-16465,-16459,-16452,-16448,  
        -16433,-16429,-16427,-16423,-16419,-16412,-16407,-16403,-16401,-16393,-16220,-16216,  
        -16212,-16205,-16202,-16187,-16180,-16171,-16169,-16158,-16155,-15959,-15958,-15944,  
        -15933,-15920,-15915,-15903,-15889,-15878,-15707,-15701,-15681,-15667,-15661,-15659,  
        -15652,-15640,-15631,-15625,-15454,-15448,-15436,-15435,-15419,-15416,-15408,-15394,  
        -15385,-15377,-15375,-15369,-15363,-15362,-15183,-15180,-15165,-15158,-15153,-15150,  
        -15149,-15144,-15143,-15141,-15140,-15139,-15128,-15121,-15119,-15117,-15110,-15109,  
        -14941,-14937,-14933,-14930,-14929,-14928,-14926,-14922,-14921,-14914,-14908,-14902,  
        -14894,-14889,-14882,-14873,-14871,-14857,-14678,-14674,-14670,-14668,-14663,-14654,  
        -14645,-14630,-14594,-14429,-14407,-14399,-14384,-14379,-14368,-14355,-14353,-14345,  
        -14170,-14159,-14151,-14149,-14145,-14140,-14137,-14135,-14125,-14123,-14122,-14112,  
        -14109,-14099,-14097,-14094,-14092,-14090,-14087,-14083,-13917,-13914,-13910,-13907,  
        -13906,-13905,-13896,-13894,-13878,-13870,-13859,-13847,-13831,-13658,-13611,-13601,  
        -13406,-13404,-13400,-13398,-13395,-13391,-13387,-13383,-13367,-13359,-13356,-13343,  
        -13340,-13329,-13326,-13318,-13147,-13138,-13120,-13107,-13096,-13095,-13091,-13076,  
        -13068,-13063,-13060,-12888,-12875,-12871,-12860,-12858,-12852,-12849,-12838,-12831,  
        -12829,-12812,-12802,-12607,-12597,-12594,-12585,-12556,-12359,-12346,-12320,-12300,  
        -12120,-12099,-12089,-12074,-12067,-12058,-12039,-11867,-11861,-11847,-11831,-11798,  
        -11781,-11604,-11589,-11536,-11358,-11340,-11339,-11324,-11303,-11097,-11077,-11067,  
        -11055,-11052,-11045,-11041,-11038,-11024,-11020,-11019,-11018,-11014,-10838,-10832,  
        -10815,-10800,-10790,-10780,-10764,-10587,-10544,-10533,-10519,-10331,-10329,-10328,  
        -10322,-10315,-10309,-10307,-10296,-10281,-10274,-10270,-10262,-10260,-10256,-10254  
    };

然后定义拼音数组：

复制代码代码如下:

//定义拼音数组  
private static string[] getName = new string[]  
    {  
        "A","Ai","An","Ang","Ao","Ba","Bai","Ban","Bang","Bao","Bei","Ben",  
        "Beng","Bi","Bian","Biao","Bie","Bin","Bing","Bo","Bu","Ba","Cai","Can",  
        "Cang","Cao","Ce","Ceng","Cha","Chai","Chan","Chang","Chao","Che","Chen","Cheng",  
        "Chi","Chong","Chou","Chu","Chuai","Chuan","Chuang","Chui","Chun","Chuo","Ci","Cong",  
        "Cou","Cu","Cuan","Cui","Cun","Cuo","Da","Dai","Dan","Dang","Dao","De",  
        "Deng","Di","Dian","Diao","Die","Ding","Diu","Dong","Dou","Du","Duan","Dui",  
        "Dun","Duo","E","En","Er","Fa","Fan","Fang","Fei","Fen","Feng","Fo",  
        "Fou","Fu","Ga","Gai","Gan","Gang","Gao","Ge","Gei","Gen","Geng","Gong",  
        "Gou","Gu","Gua","Guai","Guan","Guang","Gui","Gun","Guo","Ha","Hai","Han",  
        "Hang","Hao","He","Hei","Hen","Heng","Hong","Hou","Hu","Hua","Huai","Huan",  
        "Huang","Hui","Hun","Huo","Ji","Jia","Jian","Jiang","Jiao","Jie","Jin","Jing",  
        "Jiong","Jiu","Ju","Juan","Jue","Jun","Ka","Kai","Kan","Kang","Kao","Ke",  
        "Ken","Keng","Kong","Kou","Ku","Kua","Kuai","Kuan","Kuang","Kui","Kun","Kuo",  
        "La","Lai","Lan","Lang","Lao","Le","Lei","Leng","Li","Lia","Lian","Liang",  
        "Liao","Lie","Lin","Ling","Liu","Long","Lou","Lu","Lv","Luan","Lue","Lun",  
        "Luo","Ma","Mai","Man","Mang","Mao","Me","Mei","Men","Meng","Mi","Mian",  
        "Miao","Mie","Min","Ming","Miu","Mo","Mou","Mu","Na","Nai","Nan","Nang",  
        "Nao","Ne","Nei","Nen","Neng","Ni","Nian","Niang","Niao","Nie","Nin","Ning",  
        "Niu","Nong","Nu","Nv","Nuan","Nue","Nuo","O","Ou","Pa","Pai","Pan",  
        "Pang","Pao","Pei","Pen","Peng","Pi","Pian","Piao","Pie","Pin","Ping","Po",  
        "Pu","Qi","Qia","Qian","Qiang","Qiao","Qie","Qin","Qing","Qiong","Qiu","Qu",  
        "Quan","Que","Qun","Ran","Rang","Rao","Re","Ren","Reng","Ri","Rong","Rou",  
        "Ru","Ruan","Rui","Run","Ruo","Sa","Sai","San","Sang","Sao","Se","Sen",  
        "Seng","Sha","Shai","Shan","Shang","Shao","She","Shen","Sheng","Shi","Shou","Shu",  
        "Shua","Shuai","Shuan","Shuang","Shui","Shun","Shuo","Si","Song","Sou","Su","Suan",  
        "Sui","Sun","Suo","Ta","Tai","Tan","Tang","Tao","Te","Teng","Ti","Tian",  
        "Tiao","Tie","Ting","Tong","Tou","Tu","Tuan","Tui","Tun","Tuo","Wa","Wai",  
        "Wan","Wang","Wei","Wen","Weng","Wo","Wu","Xi","Xia","Xian","Xiang","Xiao",  
        "Xie","Xin","Xing","Xiong","Xiu","Xu","Xuan","Xue","Xun","Ya","Yan","Yang",  
        "Yao","Ye","Yi","Yin","Ying","Yo","Yong","You","Yu","Yuan","Yue","Yun",  
        "Za", "Zai","Zan","Zang","Zao","Ze","Zei","Zen","Zeng","Zha","Zhai","Zhan",  
        "Zhang","Zhao","Zhe","Zhen","Zheng","Zhi","Zhong","Zhou","Zhu","Zhua","Zhuai","Zhuan",  
        "Zhuang","Zhui","Zhun","Zhuo","Zi","Zong","Zou","Zu","Zuan","Zui","Zun","Zuo"  
   };

然后定义转换字符串的方法：

复制代码代码如下:

/// <summary>汉字转换成全拼的拼音</summary>  
        /// <param name="Chstr">汉字字符串</param>  
        /// <returns>转换后的拼音字符串</returns>   
        public string StrConvertToPinyin(string Chstr)  
        {  
            Regex reg = new Regex("^[\u4e00-\u9fa5]$");//验证是否输入汉字  
            byte[] arr = new byte[2];  
            string pystr = "";  
            int asc = 0, M1 = 0, M2 = 0;  
            char[] mChar = Chstr.ToCharArray();//获取汉字对应的字符数组  
            for (int j = 0; j < mChar.Length; j++)  
            {  
                //如果输入的是汉字  
                if (reg.IsMatch(mChar[j].ToString()))  
                {  
                    arr = System.Text.Encoding.Default.GetBytes(mChar[j].ToString());  
                    M1 = (short)(arr[0]);  
                    M2 = (short)(arr[1]);  
                    asc = M1 \* 256 + M2 - 65536;  
                    if (asc > 0 && asc < 160)  
                    {  
                        pystr += mChar[j];  
                    }  
                    else  
                    {  
                        switch (asc)  
                        {  
                            case -9254:  
                                pystr += "Zhen"; break;  
                            case -8985:  
                                pystr += "Qian"; break;  
                            case -5463:  
                                pystr += "Jia"; break;  
                            case -8274:  
                                pystr += "Ge"; break;  
                            case -5448:  
                                pystr += "Ga"; break;  
                            case -5447:  
                                pystr += "La"; break;  
                            case -4649:  
                                pystr += "Chen"; break;  
                            case -5436:  
                                pystr += "Mao"; break;  
                            case -5213:  
                                pystr += "Mao"; break;  
                            case -3597:  
                                pystr += "Die"; break;  
                            case -5659:  
                                pystr += "Tian"; break;  
                            default:  
                                for (int i = (getValue.Length - 1); i >= 0; i--)  
                                {  
                                    if (getValue[i] <= asc) //判断汉字的拼音区编码是否在指定范围内  
                                    {  
                                        pystr += getName[i];//如果不超出范围则获取对应的拼音  
                                        break;  
                                    }  
                                }  
                                break;  
                        }  
                    }  
                }  
                else    //如果不是汉字  
                {  
                    pystr += mChar[j].ToString();//如果不是汉字则返回  
                }  
            }  
            return pystr;//返回获取到的汉字拼音  
        }

　这种方法虽然也会对多音字的识别也不是很理想，但是这种方式毕竟是手动实现的，可以手动控制，比如，“家长”，我们想得到的结果是“Jia Zhang”，但是生成的结果却是“Jia Chang”、

　对于这种包含多音字的词组，我们可以另行控制，比如，定义一个多音字数组和其对应的不同词组组合，在我们进行转换拼音的时候，判断，如果是多音字，那么在其词组中找到对应的拼音即可。

　类似于我们做Web开发时的关键字过滤。

# 复合查询

|  |
| --- |
| private void button1\_Click(object sender, *EventArgs* e)  {  string sql = "select \* from T\_bookbiao";  //用来拼接sql语句  *List*<string> lSqllike = new *List*<string>();  //用来存放sql参数  *List*<*SqlParameter*> lSPar = new *List*<*SqlParameter*>();  //内容不为空  if (txt\_author.*Text*.*Trim*().*Length* > 0)  {  //添加sql字符串  lSqllike.*Add*(" Author like @author");  //添加SqlParameter参数。参数化查询为模糊查询  lSPar.*Add*(new *SqlParameter*("@author", "%" + txt\_author.*Text*.*Trim*() + "%"));  }  if (txt\_bookname.*Text*.*Trim*().*Length* > 0)  {  lSqllike.*Add*("Bookname like @bookname");  lSPar.*Add*(new *SqlParameter*("@bookname", "%" + txt\_bookname.*Text*.*Trim*() + "%"));  }  if (txt\_isbn.*Text*.*Trim*().*Length* > 0)  {  lSqllike.*Add*("isbn like @isbn");  lSPar.*Add*(new *SqlParameter*("@isbn", "%" + txt\_isbn.*Text*.*Trim*() + "%"));  }  //如果list不为空，则表示有输入  if (lSqllike.*Count* > 0)  {  //拼接sql语句。  sql += " where " + string.*Join*(" and ", lSqllike.*ToArray*());  }  *MessageBox*.*Show*(sql);  } |

# 递归（recursion）

函数自己调用自己

|  |
| --- |
| static void Main(string[] args)  {  int i = 3;  M1(i);  *Console*.*ReadLine*();  }  static void M1(int i)  {    *Console*.*WriteLine*("A"+i);  i--;  if(i>0)  {  M1(i);  }  *Console*.*WriteLine*("B"+i);  } |

输出结果：  


## 树形结构：省市信息

|  |
| --- |
| private void Form1\_Load(object sender, *EventArgs* e)  {  int Pid = 0;  LoadAreaTree(Pid, treeView1.*Nodes*);  }  private void LoadAreaTree(int Pid, *TreeNodeCollection* treeNodeCollection)  {  string sql = "select AreaId,AreaName from TblArea where AreaPId = @pid";  SqlHelper sqlHelper = new MSSqlHelper("CCDB");  SqlParamsDictionary spdic = new SqlParamsDictionary();  spdic.*Add*("@pid", Pid);  *DataTable* dt = sqlHelper.ExecuteDataTable(sql, spdic);  foreach (*DataRow* item in dt.*Rows*)  {  // MessageBox.Show(item[1].ToString());  *TreeNode* node = treeNodeCollection.*Add*(item[1].*ToString*());  Pid = (int)item[0];  //递归  LoadAreaTree(Pid, node.*Nodes*);  }  } |

## 递归删除

|  |
| --- |
| private void DeleteTreeNode(int Pid)  {  string sql = "update TblArea set IsDeleted=1 where AreaId = @pid ";  SqlHelper sqlhelper = new MSSqlHelper("CCDB");  SqlParamsDictionary spdic = new SqlParamsDictionary();  spdic.*Add*("@pid", Pid);  sqlhelper.ExecuteNonQuery(sql, spdic);  }  private void DeleteTreeRecursion(int pid)  {  string sql = "select AreaId,AreaName from TblArea where AreaPId = @pid and IsDeleted=0";  SqlHelper sqlHelper = new MSSqlHelper("CCDB");  SqlParamsDictionary spdic = new SqlParamsDictionary();  spdic.*Add*("@pid", pid);  *DataTable* dt = sqlHelper.ExecuteDataTable(sql, spdic);  int pid1 = -1;  foreach (*DataRow* item in dt.*Rows*)  {  pid1 = (int)item[0];  DeleteTreeRecursion(pid1);  }  DeleteTreeNode(pid);  }  private void treeView1\_MouseDoubleClick(object sender, *MouseEventArgs* e)  {  *TreeNode* node = treeView1.*SelectedNode*;  DeleteTreeRecursion((int)node.*Tag*);  node.*Remove*();  } |