**“C++程序设计与训练”课程大作业**

**项目报告**

**项目名称：超市商品管理系统**

**姓名：** 张为先

**学号：** 2013011542

**班级：** 自36

**日期：** 2014年9月

目 录

[1 系统功能设计1](#_1系统功能设计_1)

1.1 概述1

1.2 功能点1

[2 系统总体结构3](#_2系统总体结构)

2.1 概要设计3

2.2 小组分工4

[3 本人工作内容5](#_3本人工作内容)

3.1 系统整体架构5

3.2 GUI 制作5

3.3 逻辑层7

3.4 数据库相关11

3.5 打包与发布11

[4 项目总结12](#_4项目总结)

[5 相关问题的说明13](#_5相关问题的说明)

# 1系统功能设计

* 1. 概述

实现了大作业要求中的**全部基本功能**，还实现了“用户管理系统”、“权限管理系统”、“PDF导出”等**拓展功能**。

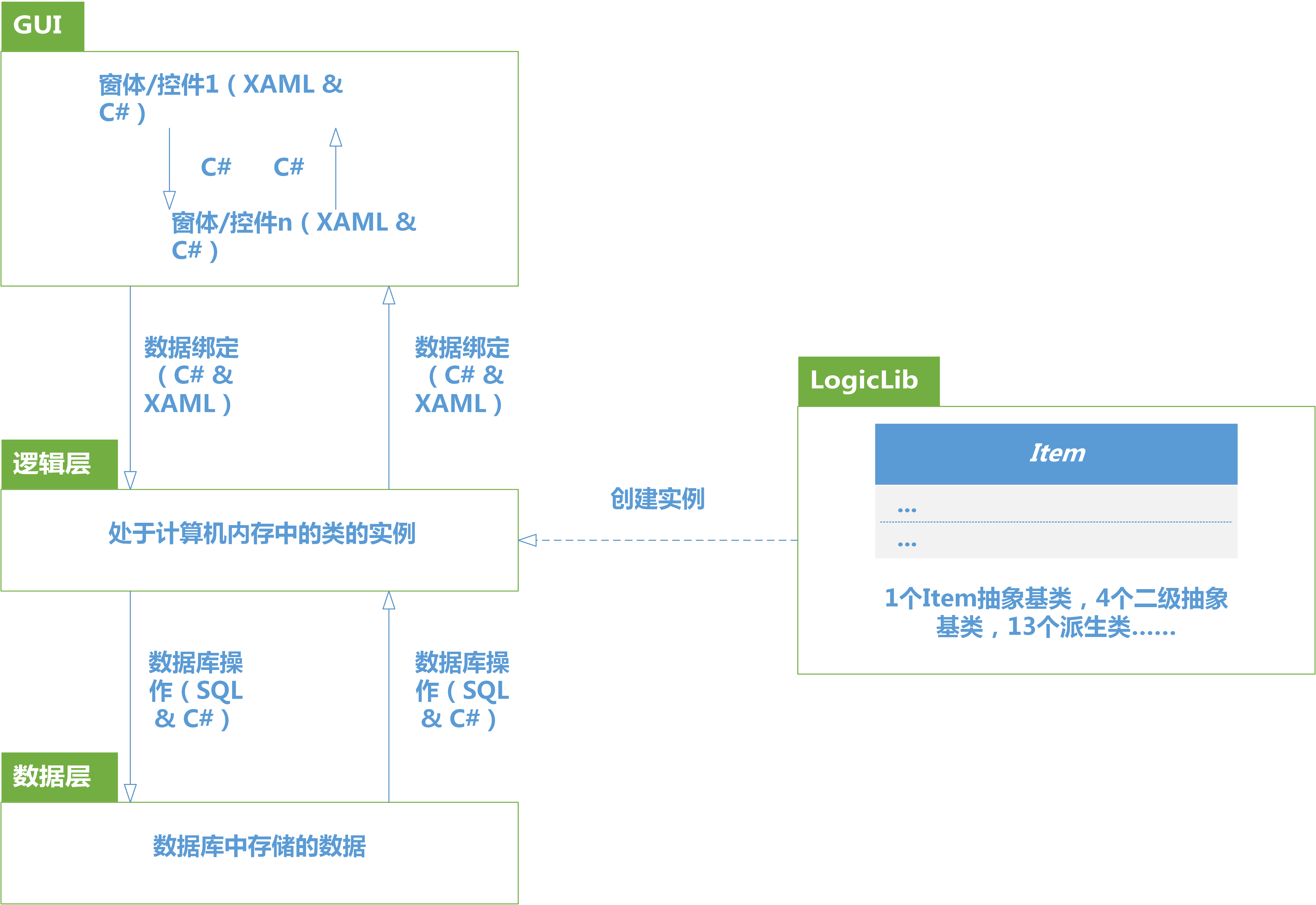
* 1. 功能点（表格）

|  |  |  |  |
| --- | --- | --- | --- |
| **功能类型** | **实现的功能点** | **实现方式**   1. **自己编写C#代码** 2. **使用C#标准库** 3. **使用第三方库** 4. **使用SQL语句** | **备注** |
| 初级功能 | 信息的读写 | 自己编写C#代码 | 所有数据均被全部读到相应的类对象中，在内存变量中完成处理过程，得到最终结果后才存储至数据库或文件。  **没有**在数据库中**直接操作**相应数据。 |
| 信息增加 | 自己编写C#代码 | 点击“操作”——“管理商品”，选择“新建商品（进货）”选项卡。  能根据用户选择的一级类别、二级类别来**动态确定**“信息增加”操作界面。 |
| 信息删除 | 自己编写C#代码 | 点击“操作”——“管理商品”，选择“删除商品”选项卡。 |
| 基本功能 | 状态变更 | 自己编写C#代码 | 点击“操作”——“管理商品”，选择“修改商品信息（上架/销售/退货）”选项卡。   1. 实现了对商品状态信息的修改； 2. 大大拓展了可修改范围，让用户能够修改商品名称、编码等基本信息。 |
| 记录退回原因 | 自己编写C#代码 | 状态为“退回”时会**强制用户输入退货原因**。 |
| 简单查询 | 自己编写C#代码 | 点击“查询”——“单条商品信息查询”。 |
| 数据可存储至文件或数据库 | 使用C#标准库（用以连接数据库） | 使用Access数据库。 |
| 容错特性 | 自己编写C#代码 | 考虑到了每一个非法操作可能会引发的问题。  比如说某项关键信息输入为空、商品编码中掺杂了特殊字符、生成统计图表时未选择需要哪一种图表、**商品状态为“退货”却未输入退货原因**等**数十种**非法操作。  **每次出现非法操作时，都会弹出提示窗口并终止非法操作**。 |
| 数据库支持 | 数据库读写 | 使用SQL语句； | **只使用**了INSERT、SELECT（未使用ORDER BY等）、UPDATE、DELETE四种**基本SQL语句**。  排序操作全部在类的方法中完成。 |
| 复杂查询功能 | 排序 | 自己编写C#代码 | 点击“查询”——“商品数量查询”。  或：点击“查询”——“商品库存/上架时间查询”。  这两个功能中的结果呈现均用到了排序。 |
| 查询指定年、月入库的商品 | 自己编写C#代码 | 点击“查询”——“商品入库时间查询”。 |
| 统计功能 | 统计四个季度的销售情况并制图 | 使用第三方库（ComponentOne） | 单击菜单栏中的“统计”，选择对应统计功能。  由于该第三方库的正版至少要400多美元，我们用的是其试用版。  使用时会弹出提示窗口，点提示击“OK”或直接关闭弹窗即可，不影响统计功能的正常使用。 |
| 统计四个大类的商品的销售情况并制图 | 使用第三方库（ComponentOne） |
| 统计销量最大的六种商品并制图 | 使用第三方库（ComponentOne） |
| 其它功能 | 用户登录与验证 | 自己编写C#代码；  使用SQL语句 | 由于超市管理系统属于商用产品，封闭度较高，故不实行开放注册的机制。取而代之的是管理者在员工管理系统中创建账号并进行授权、分发的模式（参考清华大学信息门户新生密码封的机制）。  **默认管理员工号：0003，密码：121212** |
| 密码修改 | 自己编写C#代码；  使用SQL语句 | 点击“系统”——“修改密码” |
| 新建用户资料 | 自己编写C#代码 | 点击“操作”——“管理员工”，选择“新建员工资料”选项卡。  我们参考了INFO的模式，先由管理人员创建好账号、设定初始密码，再将工号和初始密码告知用户，让用户自行修改密码，**故在此处为了方便管理人员确定初始密码，未采用掩码。** |
| 修改用户资料 | 自己编写C#代码 | 点击“操作”——“管理员工”，选择“修改员工资料”选项卡。 |
| 删除用户资料 | 自己编写C#代码 | 点击“操作”——“管理员工”，选择“删除员工资料”选项卡。 |
| 设定用户权限 | 自己编写C#代码 | 点击“操作”——“管理员工”，选择“新建员工资料”选项卡。  或：点击“操作”——“管理员工”，选择“修改员工资料”选项卡。  在里面选择对应的权限即可。 |
| 导出到PDF | 使用第三方库（iTextSharp） | 点击“查询”——“商品入库时间查询”。  选择年份、月份后单击“查询”按钮即可（未选择时间时会弹出提示并终止非法操作）。  单击“导出到PDF”并选择路径即可导出PDF文件。 |
| 信息栏 | 自己编写C#代码 | **页面左侧**的信息栏通过显示当前登录用户信息、当前在软件中所处位置、系统时间，**方便用户了解当前系统状态**。 |

# 2系统总体结构

2.1 概要设计

现行的软件设计思路推荐将UI层与逻辑层分离开来，这正是WPF技术诞生的原因。这次大作业的系统也采用了这种思路。如下是系统的总体结构：



2.1.1 GUI层

使用XAML语言与C# 制作界面外观，并使用C# 完成窗口间、控件间的信息交互。

2.1.2 逻辑层

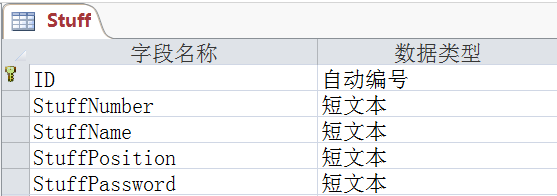
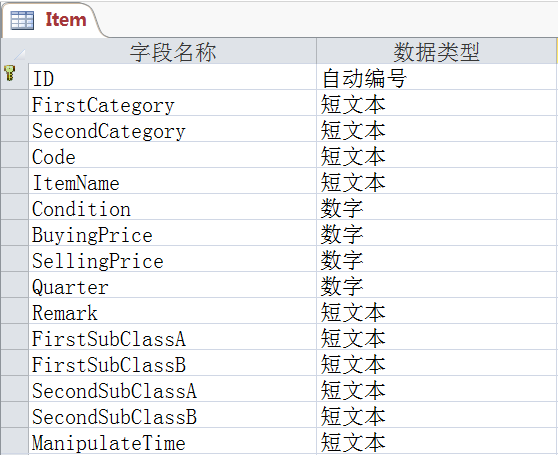
处于内存中的类的实例、结构体，以及实例、结构体的数组或动态数组。它们是排序等算法操作的对象。

2.1.3 系统与数据库的交互

考虑到系统的数据规模与兼容性等问题，我们选择Access作为我们的数据库。

考虑到我们的作品有两个比较独立的需要与数据库交互的模块（商品管理系统、员工管理系统），我们设计了两个数据表来储存信息：Item数据表和Stuff数据表；设计了两个类来实现对数据库的操作：ItemAccess类和StuffAccess类。

数据库设计附图如下（Item表和Stuff表）：



2.1.4 LogicLib

解决方案中的一个类库项目，包括1个为所有商品提供抽象基类的Item类，4个分别为电器、百货、食品、书籍提供抽象基类的二级父类，以及13个最派生类。此外还有1个员工类、2个枚举。

2.2 小组分工

|  |  |  |  |
| --- | --- | --- | --- |
| 小组成员姓名 | 小组成员班级 | 小组成员学号 | 小组成员分工 |
| 刘柏 | 自36 | 2013011548 | 前端：   1. 设计客户端整体功能结构； 2. 选择界面配色方案； 3. 编写用户手册。   逻辑层：   1. 实现系统容错功能，为每一处数据输入接口添加容错性检查功能； 2. 设计并实现了ItemAccess类，用于操作Item数据库； 3. 设计并实现了StuffAccess类，用于操作Stuff数据库； 4. 完成“复杂功能查询”中的排序； 5. 完成“操作”模块中从前端获取用户输入的数据并存入对象、在数据库中存储或更新或删除数据、在逻辑层处理数据、在前端显示数据的功能； 6. 完成“查询”模块中从数据库读取数据、在逻辑层的类中处理数据、在前端显示数据的功能； 7. 完成“统计”模块中从数据库读取数据、在逻辑层的类中处理数据、在前端显示数据的功能；； 8. 完成用户登录验证的功能；   后端：   1. 建立Market数据库，设计并创建Item和Stuff数据表； 2. 导入原始数据并增添操作时间等新属性； 3. 建立系统与数据库的连接，将数据库的操作进行封装，实现系统和数据库的相对独立。 |
| 张为先 | 自36 | 2013011542 | 前端：   1. 完成GUI的绘制与交互逻辑； 2. 完成窗后或控件之间的数据传递；   逻辑层：   1. 将各种商品抽象成类，建立了1个Item抽象基类、4个一级分类、13个二级分类的继承体系； 2. 为每个类添加字段、属性、方法（即建立了系统结构图中的LogicLib项目）； 3. 通过数据绑定（Binding）实现GUI与内存中对象的信息交互； 4. 利用第三方控件（ComponentOne）实现绘制统计图的功能； 5. 利用第三方程序集实现导出PDF文件的功能； 6. 设计实现了操作权限系统；   后端：   1. 实现修改密码功能；   其它：   1. 提出总体架构设计； 2. 使用Installshield完成程序的打包、发布。 |

# 3本人工作内容

3.1 系统整体架构

提出“UI、逻辑、数据”三层分离的想法，一方面便于代码管理，另一方面使团队成员分工明确。

3.2 GUI制作

GUI的风格上使用了扁平化的设计，力求友好、清晰、简明。

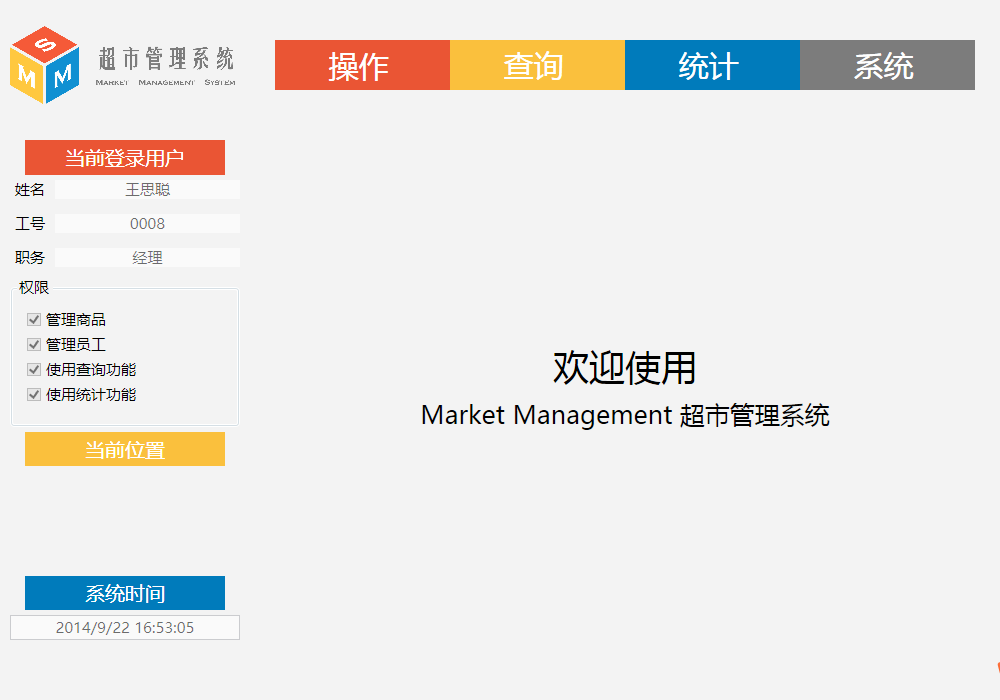
3.2.1 登录窗口



其中输入密码采用了PasswordBox控件，以起到保护密码的作用。

“登录”按钮被按下后，会在数据库中验证工号、密码，并获取用户姓名、职务、权限等信息，将这些信息以参数的形式传递给主窗体的构造函数，以供后续使用（见3.2.2（2））。

3.2.2 主窗口



1. 菜单栏



使用Menu控件制作，并更改了其样式。

1. 侧边栏



利用从登录窗口传来的参数（一个bool[]），显示当前用户的信息，其中关于权限的参数在权限管理功能中还有用途（见3.3.4）。



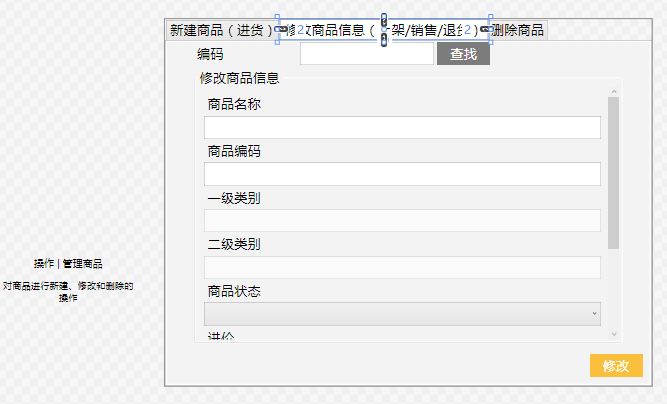
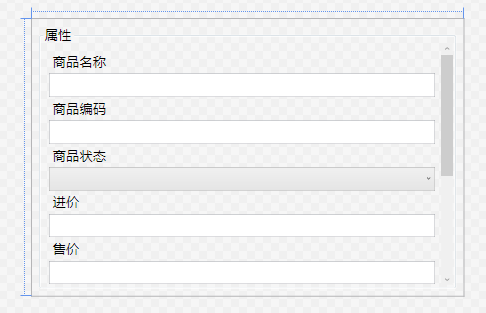
这里利用了一个DispatcherTimer（独立线程计时器）类的实例timer，这个timer每10000毫微秒溢出一次，溢出时触发Tick事件，订阅此事件的OnTimerTick()方法会读取当前系统时间并在TextBox中显示。

1. 主功能区



主功能区（图中蓝框）实际上是个ContentControl控件，根据Menu中选择项的不同，会将这个ContentControl控件的Content属性设置成不同的值。这些值都是已经预先绘制好的自定义控件，而不是逐控件地重绘界面。

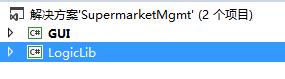
1. 自定义控件

预先在自定义控件中绘制好需要的TextBox、ComboBox等控件，在选择功能（如：点击操作 -> 管理商品）后，即将所需的自定义控件（如：ItemManagement.xaml）绘制在ContentControl中，这些控件位于GUI.FunctionBlock命名空间下。此外，一些自定义控件中还完成了绑定操作（见3.3.3），这些自定义控件主要是因为不同的类具有不同的字段，需要填写或显示的内容有所不同。它们都位于GUI.SecondClassSettings命名空间下。

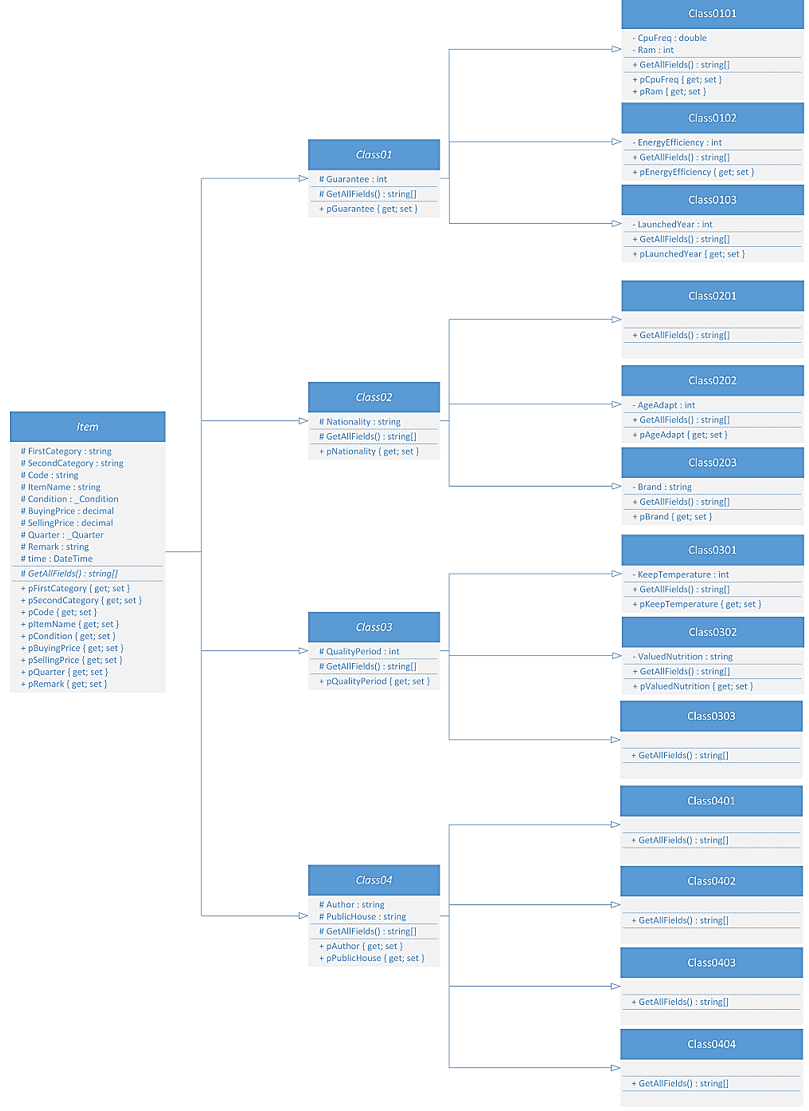
3.3 逻辑层

3.3.1 LogicLib项目



LogicLib是独立于GUI之外的项目，为解决方案提供逻辑支持。其中包含19个类和2个枚举。

1. 商品类

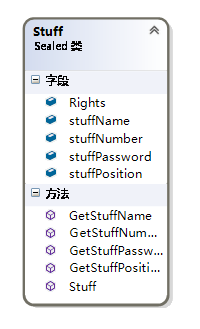


其中Item是所有商品的抽象基类，包含商品编码（Code）、商品名（Name）、进价（BuyingPrice）、售价（SellingPrice）等商品的基础信息；Class01 ~ Class04分别是电器类、百货类、食品类、书籍类商品的抽象基类，添加了一些某类商品特有的字段；Class0101 ~ Class0404是为电脑、家用电器等商品设计的类，又添加了一些更特殊的字段，这些类被设为密封（Sealed），因为它们不会再派生出新的类。

此外，每个类中有一个名为GetAllFields()的方法，它返回一个string[]，其中包含了该类商品的所有字段（全部被转换为string），以供数据库操作。这个方法采用逐级传递信息的方式运作，例如，一个Class0101类的实例computer调用了GetAllFields()方法，这个方法会通过base关键字调用父类（Class01）中的同名方法，父类方法又调用了Item类中的同名方法，逐级向string[]中添加信息。

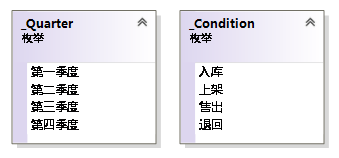
类中的每个字段都有对应的属性，以供数据绑定时使用（见3.3.3）。

1. 员工类（Stuff）



员工类中包含了员工的权限、姓名、工号、密码、职务等信息，以及获取相应信息的方法。

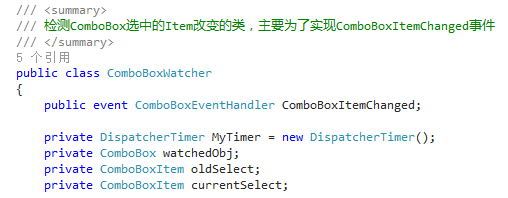
1. \_Condition枚举和\_Quarter枚举



提供商品状态和季度的枚举。由于在绑定操作完成后，枚举值会直接在控件中显示出来，故使用中文作为枚举值（见3.3.3）。

3.3.2 ComboBox选中项监视事件

在操作 -> 管理商品功能中，需要动态地根据一级类别的选项生成二级类别的选项，再动态地根据二级类别的选项在下方的ContentControl中显示相应的自定义控件（因为不同商品类别具有的属性不同，需要填写的字段也不同）。这就需要实时监视ComboBox的当前选中项。虽然.Net框架提供了SelectionChanged这一事件，但为了实现一些自定义的扩展功能，需要自己来重新实现这一事件。

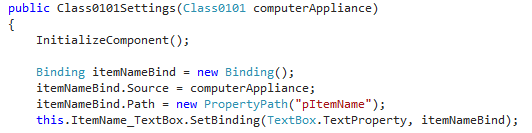


为此定义了一个GUI.Miscellaneous.ComboBoxWatcher类，主要是为了实现其中的ComboBoxItemChanged事件。在这个类被实例化后，会建立一个DispatcherTimer（独立线程计时器）对象，这个计时器每1000毫微秒溢出一次，溢出时调用一个函数，检查当前ComboBox被选中项并记录下来。下次溢出时与上次的记录值对比，若不同，则ComboBoxItemChanged事件发生，同时传出新的被选中项。订阅事件的函数可通过传出的被选中项决定生成哪些二级类别选项，或在ContentControl中显示哪个控件。

3.3.3 设置绑定

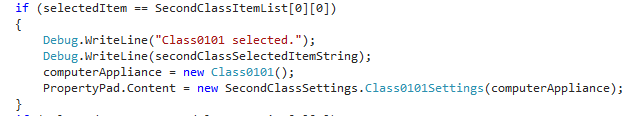
WPF中的绑定（Binding）技术，可以将UI层上某控件的一个属性同逻辑层中某对象的一个属性联系起来。这两者中任意一者发生了改变，另一者就会立即随之改变。这一技术为商品的新建、单条信息查询提供了方便。

绑定过程均于GUI.SecondClassSettings命名空间下各类的构造函数中完成。构造函数带有一个参数，它是某个二级商品分类的对象，作为绑定的源。下图代码实现的就是将商品名称TextBox的文本属性与Class0101类对象的pItemName属性绑定起来。



枚举类型和ComboBox都具有有限种选择情况，把它们绑定起来是非常自然的想法。然而，自定义的枚举类型没有实现INotifyCollectionChanged接口，无法实现动态绑定，因此需要另一种方式完成它们之间的绑定——将枚举值都添加到一个List中，然后将这个List设为ComboBox的选项来源。这就是枚举值是中文的原因。

绑定技术的使用意味着在实例化UI层控件之前，应该有一个逻辑层中的商品对象先被实例化。以新建一个电脑类商品为例，确定选中的二级分类（0101）后，会在后台实例化一个Class0101类对象computerAppliance，并将其作为参数传递给UI层控件的构造函数，然后，将这个控件在ContentControl中显示出来。此时，用户在UI层上的改动就会立即引起computerAppliance对象的相应变动。

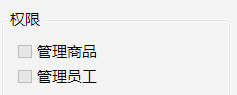


单条查询商品信息时，使用的是与新建商品相同的UI自定义控件。与新建不同的是，逻辑层对象在被构造并设置绑定后会被从数据库中提取的信息赋值，UI层上会同步地体现这一变化——商品信息会被显示出来。

3.3.4 权限系统

权限控制的实现比较简单，在主窗口的构造函数MainWindow()的参数中有一个bool[]，它除了控制侧边栏“权限”框的显示外，还决定了主窗口上某些菜单项是否可用。

例如，用一个没有管理商品、管理员工权限的工号登录，“权限”框中的前两项的IsChecked属性会被置为false，同时操作菜单下的两个MenuItem的IsEnabled属性也被置为false，无法访问。



3.3.5 绘制统计图

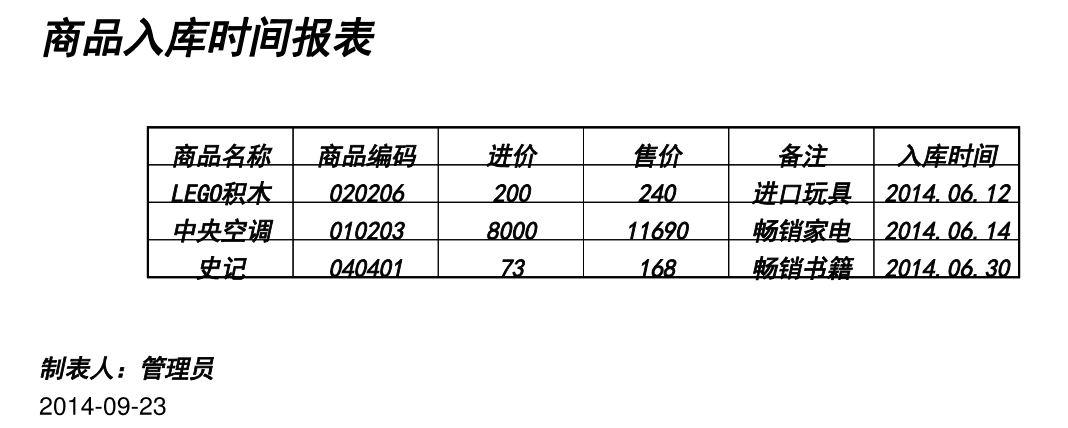
统计图功能使用第三方控件（Component One）实现。当“统计”按钮被按下时，即从数据库中提取信息并处理，然后将处理好的信息赋给统计图控件数据集的ValuesSource（数值）和ItemNames（统计项名称）两个属性。统计结果的呈现方式则依据ComboBox中的选项设置统计图控件的ChartType属性。

3.3.6 导出PDF**[[1]](#footnote-1)**

导出PDF的功能可以通过I/O流来实现，但过于麻烦，且几乎无法控制格式，难以保证导出文件的质量。所以采用了第三方组件iTextSharp与ICSharpCode.SharpZipLib来实现导出PDF的功能。

导出PDF的功能于“商品入库时间查询”中实现。点击“导出PDF”按钮后，第三方组件提供的Document类被实例化，然后一个SaveFileDialog类窗口被打开，让用户选择保存文件的路径。接下来，向PDF文件中添加标题、表头，并逐行添加表格信息。最后，利用主窗口接收到的员工信息参数，添加制表人姓名；并获取当前系统日期，添加为制表时间。

这其中有一点问题：写入中文字符时，必须指定字体，否则中文字符无法正常显示。为获取一种中文字体，使用了System.Environment.GetEnvironmentVarialbe()方法，获取到了WINDIR环境变量的值（通常是C:\Windows），然后访问其中的Fonts文件夹，使用绝大多数计算机上都会安装的黑体（simhei.ttf）作为PDF文件中的字体。



用同样的方法可以为其它查询菜单下的其它项设计导出PDF功能。

3.4 数据库相关

数据库操作是刘柏同学的工作，在他的工作的基础上，利用查找、修改函数，实现了修改密码的功能。



“确定”按钮被按下后，修改密码窗口中三个PasswordBox的Password属性被获取，首先在数据库中查找当前登录用户的旧密码，若不匹配则弹出MessageBox，并终止操作；然后对比下面两个PasswordBox的Password属性是否相同。最后调用数据库修改函数，在数据库中修改密码。

3.5 打包与发布

使用Installshield，将可执行文件与必要的动态链接库打包。由于连接数据库时，数据库路径被设为..\Database\Market.mdb，因此数据库文件被放在Database文件夹中一同打包。

# 4项目总结

4.1 合作开发与版本控制

这次大作业是首次体验较复杂软件的开发，最初原本打算用tortoise或Visual Studio自带的团队管理工具进行合作开发的管理和版本控制，却由于某些原因未能使用。这给合作开发带来了一些不便——我们使用文本比较软件来比较、更新各自写的代码；一个版本的代码经更新替换后就无法恢复。这给项目开发也带来了一些风险。版本控制是一项在复杂工程中非常必要的技术，以后应加强这方面的学习。

4.2 合理的架构体系与合理分工的重要性

由于采用了“UI、逻辑、数据”三层结构，我和刘柏同学得以合理、明确地分工——我主要负责UI的绘制，以及UI和逻辑层之间的交互；刘柏同学主要负责数据库的搭建，以及数据库和逻辑层的交互。这样的分工使我们的工作相互独立，一个人对代码的修改对另一个人的工作几乎没有影响。也正是得益于这种合理的分工，才使我们的项目在没有使用合作开发工具的情况下不至于出现混乱。

4.3 用户体验与UI设计

虽然用户体验为表，程序功能为里，但在这个程序功能十分强大复杂的时代中，UI设计和用户体验也应得到充分的重视。只有清晰简明的UI，合理的提示、引导，才能使程序最大化地发挥功能。项目中，我们尽可能地从实际出发，把这个项目考虑成一个实际的工程，尽可能考虑用户的需求；而非仅仅是一个大作业。

4.4 关于大作业命题的质疑与数据库结构的思考

大作业要求中，提出了“实现商品信息的新建、修改、删除”的要求，这使得我们将数据库中的一条信息设计成“一件商品”而非“一项操作”，当商品状态发生变化时，我们修改了数据库中相应的值，从而丢失了商品上一条状态的纪录。整个数据库中，仅能查询到商品现在的状态。比如一件售出的商品，它是何时进货、何时上架的，不得而知。

而在现实生活中，超市管理方不可能也不应该去修改或删除商品的信息。当商品状态发生变化时，我们做的应该是建立一条新的数据，来纪录这一操作，而不是把原有的纪录更改或覆盖掉。只有这样，才能记录下一件商品从入库、上架到售出或者退回的完整信息——这样的记录才是客观的、实用的。

4.5 代码复用性与编程艺术

在项目中，我尝试过一些提升代码复用性和可维护性的做法，但都没成功；为了项目的正常推进，最终采用了最简单最朴素的方法，导致项目中存在较多的机械性重复代码。例如GUI. SecondClassSettings命名空间下那13个控件，最初的设想是通过反射（Reflection）技术遍历一个类的成员，并在UI层绘制相应的控件——例如检测到一个名为“ItemName” 的成员，就在UI上绘制一个TextBox用于填写商品名称。如果能以这种方式完成，将使程序的代码复用性、可维护性得到很大的提高，因为对LogicLib中类定义的改动会自动导致UI控件的相应改动，而不需要开发者自己去重做控件。但由于个人技术欠缺，没有实现这一设想。提高代码复用性、优化程序结构与代码风格不仅只是一种观念，它需要扎实的编程基础与高超的编程技巧的支持。

# 5相关问题的说明

5.1 开发、测试环境

开发环境：Visual Studio 2013 Ultimate、Visual Studio 2012 Premium

测试环境：Windows 8.1、Windows 7

5.2 开发语言

C# & XAML

5.3 试用版控件说明

绘制图表使用的Component One控件是付费控件，且价格昂贵，因此项目中使用的是试用版控件。对于在软件使用过程中可能会弹出激活窗口，关闭即可，不影响正常使用。

1. 此功能的实现参考了来自网络的资源：苟安廷 .《PDF文件制作全攻略》 [↑](#footnote-ref-1)