山东理工大学

SHANDONGUNIVERSITY OF TECHNOLOGY

毕业设计说明书

基于 UWP 的山理校园通系统设计与实现

学　　院：计算机科学与技术

专　　业：计算机科学与技术

学生姓名：严修灏

学　　号：12110501101

指导教师：李盘靖

2016 年 06 月

摘 要

随着微软对 Windows 10 的投入力度的加强，已经有越来越多的设备运行了 Windows10 系统，目前这一数字为 3 亿，而微软的目标是 3 年内 10 亿，相信微软会持续加强对升级 Windows10 的工作力度。UWP 通用应用真正实现一次编写、一套业务逻辑和统一的用户界面。应用在统一的 Win10 商店中将只会有一个安装包，而它将适用于所有 Win10 设备。所以本次课题研究基于 UWP 的应用程序。

本课题的研究基于山东理工大学在校生的实际需要，满足学分查询、绩点查询、教务系统查询、往年四六级成绩查询等功能。本着简洁易用的原则，力求结构清晰、使功能一目了然。在本篇论文中，从开始先介绍了开发 UWP 所需的系统以及开发环境，然后介绍了本次项目所使用的主要控件、一些重要的方法。接着对每个页面分别进行分析讲述，对使用到的技术、注意事项进行说明。最后，讲述了如何将开发的应用上传至应用商店进行发布并进行后续维护与更新。

初步测试与试运行表明，本应用具有较好的稳定性和出色的易用性，用户能够轻松的查询到常用的信息，对山东理工大学的在校生提供了较大的帮助，实现了项目的初衷。

关键词：Windows10， UWP（Windows 通用应用平台），校园助手

Abstract

With Microsoft enhanced Windows 10 investment, has an increasing number of devices running Windows10 system, now that figure is 300 million, while Microsoft's goal is 1 billion within 3 years, I believe that Microsoft will continue to enhance efforts to upgrade Windows10. UWP General true write once, a set of unified user interface and business logic. In the unified Windows10 stores will only have an installation package, and it will apply to all Windows10 devices. So this research based on UWP.

The research based on the practical needs of undergraduates of Shandong University of Technology, meet the GPA queries, credit inquiries, educational system, CET4 and CET6 score queries in previous years and other functions. Based on user-friendly principles, seeks to clear structure and functions at a glance. In this paper, starting from first introduced the development required for the UWP system and development environment, and then introduced this item by using the main Controls, a number of important functions. Then analyze each page individually about, describes the technology used, and some troubles need to avoid. Finally, tells the story of how the application release uploaded to the app store and subsequent maintenance and updates.

Preliminary testing and trial run showed that this application has good stability and excellent ease of use, users can easily query to common information for undergraduates of Shandong University of Technology provides a great help to achieve the purpose of the project.

Key words: Windows10, UWP(Universal Windows Platform), Campus assistant

目录

第一章 引言

1.1 课题的背景和意义

Windows 10 是微软开发的最新一代操作系统，是 Windows 8 和 Windows 8.1 的后继者，微软在 2015 年 7 月 29 日正式面向全球发布了该操作系统。同时带来了 UWP（Universal Windows Platform）项目，即 Windows 通用应用平台，在 Windows 10 PC、平板、手机、Xbox、HoloLens 甚至是嵌入式设备运行。UWP 不同于传统的 PC 上的 x86 应用，也不是只适用于手机的 APP，UWP 不为特定的设备而设计，而是所有的安装有 Windows10 系统的终端上运行，实现了了一次编写，处处运行。这对于开发人员来说是一个巨大的机遇，简化了开发流程、降低了适配成本。而微软也在努力推进 Windows10 的全球覆盖率，目前全球已有超过 3 亿台设备运行 Windows10 系统，而且这个数字还在持续上升。所以在这个时期跟进 UWP 应用的开发，抓住前期的时机意义重大。

近年来，随着移动互联网的发展以及国内智能设备的迅速普及，基于校园信息系统的各类校园助手层出不穷。比较出名的有几年前的课程格子和超级课程表，而他们的创始人也都是在校大学生，这令人看到了大学生在做校园类应用时得天独厚的优势：更了解自身的需求，从而设计出更贴切本校学生使用习惯的程序，这对于开发此类应用意义重大。为此，本文对基于 UWP 的校园助手开发进行了研究。

1.2 当前课题的研究现状

纵观各类校园助手应用，无论是 Android 平台还是 IOS 平台都异常火爆，差不多每个高校都有几款自己特色的 APP，这一领域在这两个平台已是基本饱和。但是目光一转，来到 Windows 平台，此类应用的数量却与之前两大平台相去甚远，其中既有 Windows 平台在移动领域落后较多的因素，也有 Windows 10 的 UWP 发布日期不久，市场反应尚需一段时间的原因。不过，近期已陆续有开发者将适合自己学校的应用发布到了应用商店，并受到了本校同学的一致好评，被冠以良心开发者称号。考虑到 UWP 的优点以及当前理工大学的需要，课题应运而生。

1.3 主要研究内容与章节安排

本设计主要研究了在 Windows10 平台下 UWP 应用的开发特点。包括页面的布局方式、页面间的导航、资源的定义、数据绑定以及各种控件的组合使用。在此基础上初步实现了原定的基本功能。

在以下章节中，首先分析系统功能以及 UWP 开发的特点，在之后的章节中重点介绍了布局样式的实现、用于实现功能的主要控件。然后使用在前一章介绍的控件、布局来实现系统具体功能的实现。其次，在功能实现后进行了一下测试安排，保障软件的容错能力和健壮性。最后，进行本设计的经验总结，并展望以后的发展前景。

整篇文章的章节安排由点及面，先介绍基础后形成整体，力求文章通俗易懂，能对后来的读者形成一定的指导作用。在阅读本文时建议从前往后依次展开，这样有助于对整个项目的理解。

第二章 系统分析与相关技术

2.1 系统环境与工具

在这次设计中，由于目标是 Windows10 平台下的 UWP 应用开发，按照微软应用开发的要求，开发 Windows10 的 UWP 应用需要装有 Windows10 系统的 PC 机以及最新的集成开发环境 Visual Studio 2015。所以，在本次设计中，PC 端的开发过程使用 Windows 10 系统版本 1511（OS 内部版本 10586.318）；手机端测试时使用 Windows 10 移动版，版本 1511，OS 版本：10586.318；Visual Studio Community 2015 版本 14.0.24720.00 Update 1；Microsoft .Net Framework 版本为 4.6.01038。

在装有 Windows10 的 PC 机安装 Visual Studio，勾选“通用 Windows 应用开发工具”，如图 2.1 所示。

图 2.1 VS 安装时勾选项

Visual Studio 是微软公司推出的开发环境，Visual Studio 可以用来创建 Windows 平台下的 Windows 应用程序和网络应用程序，也可以用来创建网络服务、智能设备应用程序和 Office 插件。Visual Studio 功能强大但价格不菲。而 Visual Studio Community 2015 是微软推出的面向小型公司、创业公司、学术研究、教育以及开源软件开发者，用于开发非企业级项目的集成开发工具，可以用来开发桌面、移动、Web 以及云应用等全套

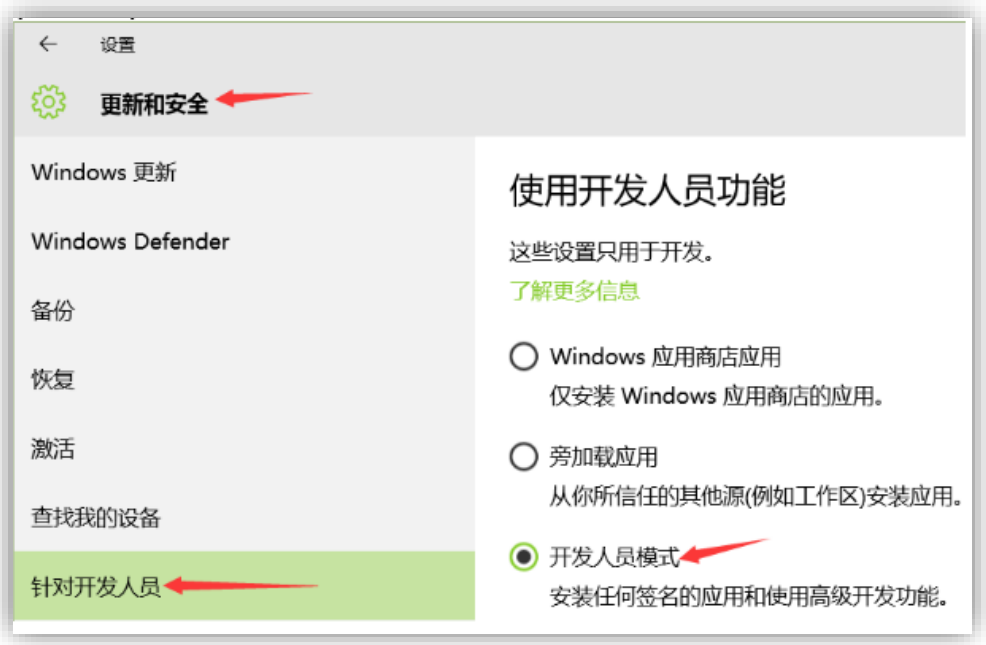
功能，最大的特点是免费的，这对于学生来说意义重大，这使得学生可以以最低的成本接触最新的 IDE，提高学习效率。因此，本次 IDE 选择 Visual Studio Community 2015。

2.2 开发者账户

在 Windows10 通用应用开发中，开发者若想要将自己的 APP 上传的微软商店供用户免费或付费下载，需要创建一个自己的开发者账户。得益于微软的 DreamSpark 计划，高校在校生可以免费申请属于自己的账户，为学生免费提供 Microsoft 专业级开发人员和设计人员工具，让学生可以追逐自己的梦想并实现技术上的新突破，或者为其职业开创一个良好的开端。（DreamSpark 是微软对教育的一个支持项目，通过允许处于学习、教学和研究的目的来访问 Microsoft 软件，从而为技术教育提供支持。详情链接 https://www.dreamspark.com/）

2.3 启用开发者模式

把电脑和用于调试的 Windows 10 手机的开发者模式打开，这样才能在设备上部署正在编写的程序并进行调试，开发者模式路径：设置—更新和安全—针对（面向）开发人员，如图 2.2。



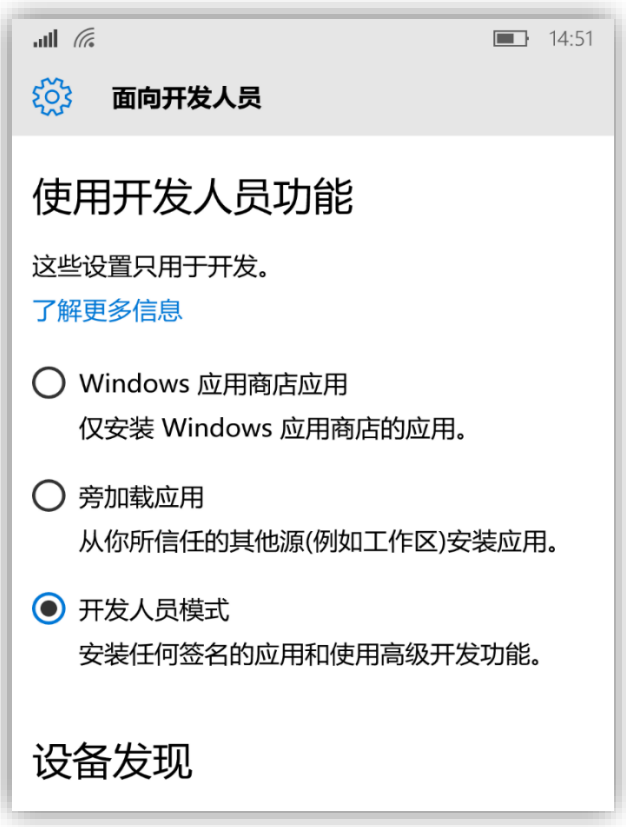


图 2.2 电脑和手机的开发者模式

2.4 创建通用程序项目

打开 Visual Studio Community 2015，点击新建，依次点击图示项目，在模板中选择 Visual C#—Windows—通用—空白应用（通用 Windows），添加名称，点击确定后创建第一个项目，如图 2.3 所示。

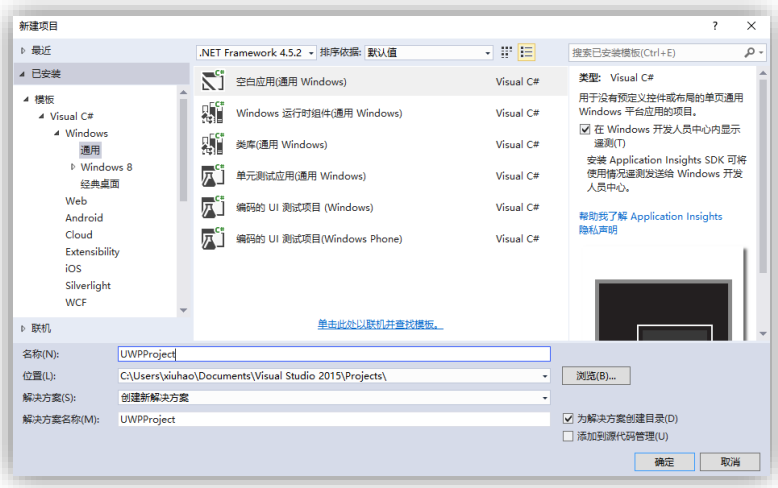


图 2.3 创建新的项目

确定后展现的便是 XAML 设计界面，左侧的 XAML 设计器，右上方解决方案管理器，右下方属性窗口，如果不经意间把窗口给关闭了，可以在 VS 顶部的窗口菜单中的重置窗口布局来回复默认的样式。设计界面如图 2.4 所示。

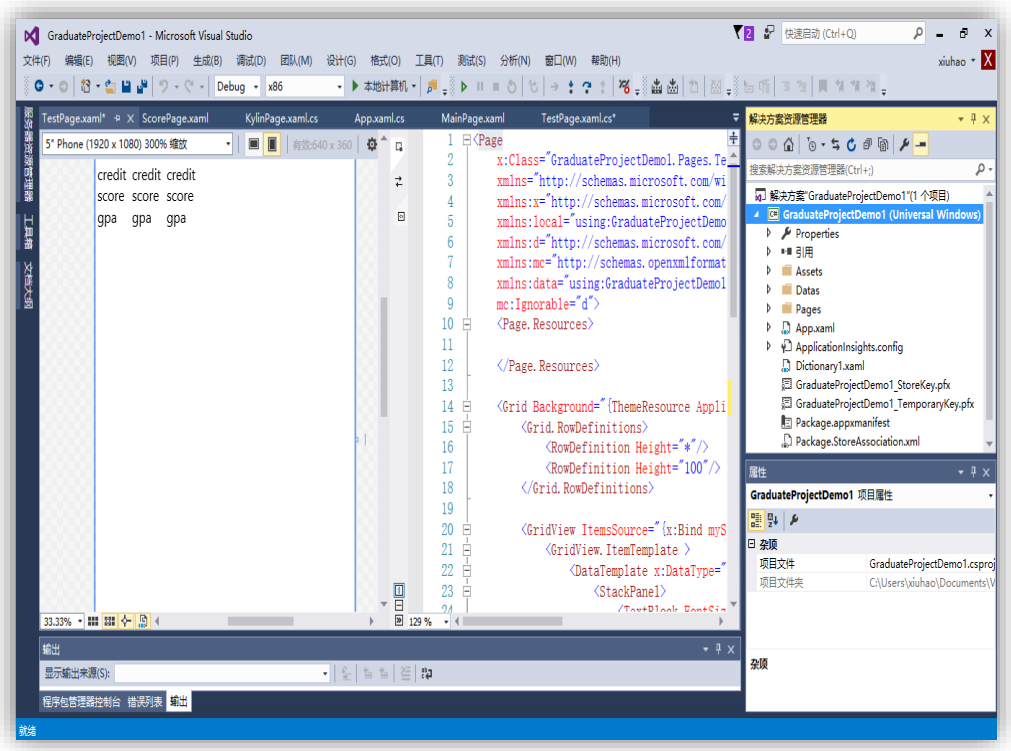


图 2.4　默认设计界面

2.5 XAML

2.5.1 XAML概述

UWP 应用的开发分为两部分：前台设计（XAML）和后台控制逻辑（C#）。XAML（eXtensible Application Markup Language）中文解释为：可扩展应用程序标记语言，它源于 XML，相当于是 XML 的变种，XML 和 HTML 属于同一个族系，所以语法、用法都有几分相似。但是 XAML 和 HTML 又有不同，XAML 在布局设计、交互逻辑等方面要优于 HTML，比起 JSP 更是简洁高效、一目了然，对于 XAML 的官方论述见 MSDN 官方支持文档：

https://msdn.microsoft.com/zh/cn/library/ms752059(v=vs.110).aspx#what_is_xaml

该语言基于 XML，但比 XML 要更加强大，XAML 可以定义更完整的寓意以及属性特性。主要用来构建应用程序的用户界面，由 XAML 构建的用户界面可以很轻易地将其与程序的业务逻辑部分分离。通过 XAML，开发人员和设计人员可以很方便地单独进行程序逻辑开发和界面设计，并且开发人员和设计人员的沟通更加方便快捷。

2.5.2 XAML设计

在 Visual Studio 中，提供了可视化的 XAML 编辑器，以及各种可以拖动的控件、布局。鉴于 XAML 编辑器的智能，在编写 XAML 代码的时候简洁优雅了许多。如图 2.5。

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="100"/>
    </Grid.RowDefinitions>

    <GridView ItemsSource="{x:Bind myScores}">
        <GridView.ItemTemplate >
            <DataTemplate x:DataType="data:Scores">
                <StackPanel>
                    <TextBlock FontSize="16" Text="{x:Bind id}"/>
                    <TextBlock FontSize="16" Text="{x:Bind name}"/>
                    <TextBlock FontSize="16" Text="{x:Bind credit}"/>
                    <TextBlock FontSize="16" Text="{x:Bind score}"/>
                    <TextBlock FontSize="16" Text="{x:Bind gpa}"/>
                </StackPanel>
            </DataTemplate>
        </GridView.ItemTemplate>
    </GridView>

    <StackPanel Grid.Row="1">
        <Button x:Name="bu"
                Content="test"
                Click="bu_Click"/>
    </StackPanel>
</Grid>
```

图 2.5　XAML 格式

2.5.3 XAML属性说明

鉴于 XAML 是 XML 的变种，所以 XAML 的语法以 XML 的语法为基础，但是相比较后者功能大大加强，可以更加简洁高效得地定义更加完善的语义和属性特征，常见用法如下：

声明一个对象：

在 C#中，我们通常需要以下语句来声明一个 TextBlock 对象：

```
TextBlock myTextBlock = new TextBlock();
```

但是在 XAML 中，我们只需要以下语句便可以创建一个 TextBlock，并且可以在可视化窗口中看到他的样式。

```
<TextBlock x:Name="MyTextBlock" />
```

XAML 的嵌套：

在 XAML 中，根据控件之间的继承关系，大多数 UserControl 控件都可以添加到布局控件之中。因为一般来说，大多数的布局控件都具有 Children.Add()方法，通过这个方法，可以把 UserControl 控件嵌套在自己的层级下边。例如，将上面提及的 TextBlock 放在一个 StackPanel 控件中时，只需在 XAML 编辑器中添加如下代码：

```
<StackPanel x:Name="myStackPanel">
    <TextBlock x:Name="MyTextBlock" />
</StackPanel>
```

而这相当于我的的 C#代码：

```
TextBlock myTextBlock = new TextBlock();
StackPanel newPanel = new StackPanel();
newPanel.Children.Add(myTextBlock);
```

设置属性：

XAML 的属性设置方法与 XML 基本一致，都是属性名称=“属性值”，属性名称写在元素名称之后，多个属性之间至少用一个空格隔开，例如下边这个例子，就是用 XAML 设置了 Button 的 Name、Height、Width、Content 属性：

```
<Button x:Name="button"
        Height="50"
        Width="50"
        Content="test" />
```

这相当于以下的 C#代码：

```
Button button = new Button();
button.Height = 50;
button.Width = 50;
```

这里有一点需要值得我们关注，假设我们在 XAML 里添加了一个名为 myButton 的 Button 控件，同时又在 C#代码里边声明了一个同样名为 myButton 的 Button 控件，那么 XAML 里边的这个控件将会被隐藏掉。

XAML 命名空间：

使用命名空间的目的在于消除在引用其它内容的时候产生的歧义，XAML 使用命名空间 xmlns（即 xml namespace），该属性是一个 URL，但是这个 URL 地址已经无法再浏览器中打开，只是表示一些定义好的规则，如图 2.6。

```
x:Class="GraduateProjectDemo1.Pages.TestPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="using:GraduateProjectDemo1.Pages"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:data="using:GraduateProjectDemo1.Datas"
mc:Ignorable="d">
```

图 2.6　XAML 命名空间

再说一下大部分元素及控件元素的 Name 和 x:Name 的区别，虽然这两个属性都可以为控件命名，但两者却有显著的不同：Name 是一种依赖属性，而 x:Name 才是元素的属性。Name 和 x:Name 在有些情况下可以交替通用，但是如果在相同的元素上同时指定这两个属性就会出错了。除此之外，x:Name 的值是无法在后台的 C#代码中进行修改的，而 Name 可以

修改。不过这里有一点需要注意，如果 Name 的值被改变了，那样的话在用 Name 唯一表示的各个控件的使用就会出现错误。因此，在编写 XAML 代码的时候我们应该尽可能用 x:Name 而不是 Name 来给控件命名。

最后，说一下 XAML 的功能。XAML 在 UWP 的开发中起着举足轻重的作用，它可以将现有的控件模板化，可以使用资源字典定义自己想要的控件格式。除此之外，如果在 UWP、WinowsForm、ASP.Net 等开发中都是用 XAML 创建前端的用户界面，那么如果需要将应用进行各平台的移植时，这无疑将大大降低工作量，并且得益于所见即所得式的使用方法，设计人员在没有开发人员的帮助的情况下即可创建出想要的 UI。

第三章 XAML控件

3.1 控件简介

为实现本程序，控件是基础，故本章单独介绍这一重要部分。一个程序的 UI 是由许多个页面组成的，而组成这一个个页面的基本元素就是控件了，多个不同种类的控件通过有序的组合便构成了一个复杂页面，按照用途，常用的控件可分为布局类、按钮类、选择类等。

所有的控件继承自 FrameworkElement.FrameworkElement 为所有的控件和 UI 布局提供统一的 API 框架，同时又定义了与数据绑定、对象树和生命周期相关的一些 API。UIElement 是用户界面中具有可视外观并可以处理基本输入的大多数对象的基类。DependencyObject 表示参与依赖项属性系统的对象，是许多重要的 UI 相关类的直接基类。

3.2 常用控件

其中，Panel 和 Control 两个类是重要的控件基类，这两个类都在命名空间 Windows.UI.Xaml.Controls 里面。Windows.UI.Xaml.Controls 名称空间提供了支持现有和自定义控件的 UI 控件和类，通过 Panel 的子类可以对界面中的控件进行定位和排序，Control 的子类则包括了许多开发中的常用控件，例如文本输入框、按钮等。以下文字对本项目所使用的主要控件及其属性做一个初步的介绍。

3.2.1 Grid控件

Grid 控件，Grid 是一个布局控件，个人理解 Grid 好比 HTML 中的 Table 标签，可以自己定义行和列，也可以把 Grid 中的控件放在指定的行或列中。在创建一个空白页面时，XAML 设计器会自动添加一个 Grid。通常我们用 Grid 布局将网格划分成指定的行与列时，通过设置其子控件在第几行第几列的位置来摆放。特别是对位置的定位要求比较高且灵活的时候，我们使用 Grid 会带来很大的方便。

3.2.2 StackPanel控件

StackPanel 控件，也是一个布局控件，StackPanel 标签好比是 HTML 中的 div 表单。在默认情况下，其子控件会按照对齐方式依次填满这个面板，既不覆盖，也不留空隙，即第一个紧挨第二个，除非手动设置 margin，而不是像 Grid 那样重叠在一起。StackPanel 标签中有 Orientation 属性，用来控制标签中的元素是横向排列（Horizontal）还是纵向排列（Vertical）。StackPanel 不会自动换行，如果子控件达到了 StackPanel 的最大高度或宽度，也会继续显示，即使这有可能会被遮挡。

3.2.3 SPlitview控件

实现汉堡包菜单，最重要的一个控件就是 SplitView 控件，这是在 Visual Studio2015 的通用应用中新加入的一个控件。SplitVIew 可以让我们创造一个面板，在视图内来回切换。它有两个部分组成那个：左边的菜单（SplitView.Pane）部分和右边的主体（SplitView.Content）部分。

SplitView 在我们使用时常用属性主要有：

Name：所有控件的通用属性，控件的名字，用来在后台用 C#语言进行控件的唯一标识。

DisplayMode：控件的 Pane 的展示方式，又分为以下几种情况：1、OverLay:Pane 显示时会直接浮在 Content 的上面,点击非 Pane 区域则 Pane 隐藏，在隐藏时完全消失。 2、Inline：Pane 显示时会将 Content 挤开一段距离，即 OpenPaneLength，隐藏时完全消失，不点击则不会消失。3、CompactOverlay:根据 CompactPaneLength 的长度设置,会露出一段 Pane，其它行为与 Overlay 一致，适用于导航菜单由图标加文字组成，展开显示全部，隐藏时仅显示图标。4、CompactInline：类似于 CompactOverlay，根据 CompactPaneLength 的长度设置，会露出一段 Pane，其它行为与 Inline 一致，适用于导航菜单由图标加文字组成，展开显示全部，隐藏时仅显示图标。IsPaneOpen： Pane 是否默认展开，一般为否，即 IsPaneOpen=“False”;CompactPaneLength：在 CompactOverLay 和 CompactInline 模式下,Pane 关闭时所露出来的那一点。OpenPaneLength:Pane 打开时的宽度。

SplitView 的 Content 属性比较简单，往往会直接嵌套一个 Frame 用来显示其它的页面，并实现页面的转换。一个简单的 SplitView 的效果如图 3.1 所示。

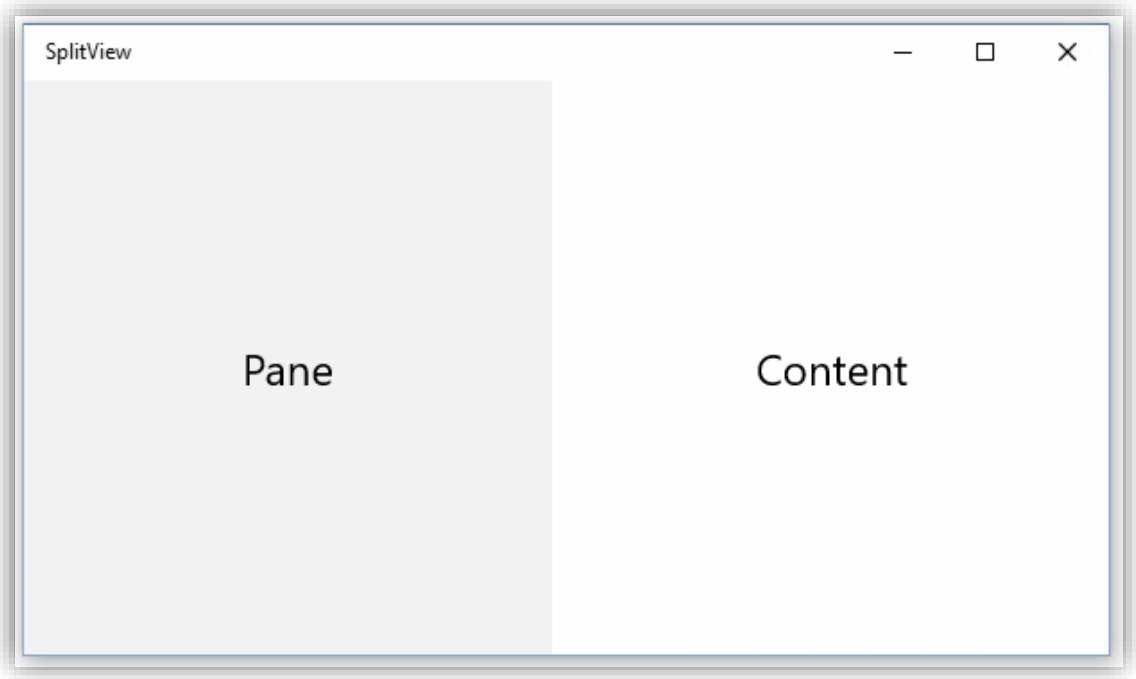


图 3.1　SPlitView 样式

3.2.4 ListBox控件

当我们把 Splitview 的框架搭好，有了完整的 Splitview.Pane 和 SplitView.Content 时，我们就可以往 Pane 里边添加菜单了，在这里，我们选用 ListBox 控件，ListBox 控件的常用属性有：

SelectionMode：值有 Single（默认）、Multiple（多选）、Extended（延伸），一般我们使用 Single 即可；

Selectionindex：默认选中的项的索引，从 0 开始。

ListBox 内有一个或多个 ListBoxItem 标签，这些标签构成了 ListBox 的内容，在 ListBox 里边可以嵌套 StackPanel 等控件，从而得到我们想要的效果。如大多数索引那样，ListBoxItem 的项目从第一条开始，索引属性为 Tag=“0”，第二条 Tag=“1”，以此类推。

ListBoxItem 中的内容，在这里选用 TextBlock，TextBlock 是基本的文本显示控件，类似于 WinForm 的 Label 标签，在这里用 TextBlock 显示图标，我们设置 TextBlock 的 FontFamily 属性为“Segoe MDL2 Assets”，这种字体是一种资源字体，是微软专门为 Win10 特点而改进的。Segoe MDL2 Assets 字体的详细说明见微软 MSDN 文档：

https://msdn.microsoft.com/zh-cn/library/windows/apps/jj841126.aspx

3.2.5 RelativePanel控件

RelativePanel 的本质是一种面板，也就是说它是一种容器，在 RelativePanel 中可以放其他的控件，比如 TextBlock、Button 等等。RelativePanel 的不同之处在于它提供了一些附加属性用于它的子元素，然后通过子元素的相对位置来实现布局，这样一来，不管当前的面板大小怎么变化，包含在 RelativePanel 中的控件的相对位置都会保持不变，这一点在 UWP 的跨平台特性上便显得尤为重要，因为考虑到不同设备类型导致的屏幕比例尺寸差异，通过使用 RelativePanel 便可以将各个控件的位置保持在我们想要的地方。相对位置常用的有两类：一是子元素之间的相对位置；二是子元素相对于面板的位置。其中，子元素相对于面板的位置的优先级要高于子元素之间的位置，比如：TextBlock 的 AlignTopWithPanel 的属性优先级就要高于 TextBlock 的 AlignTopWith 的属性。

3.2.6 AppBarButton以及Button控件

在以往，APPBarButton 是在屏幕的下方的，在 UWP 中可以把它放在上面了，当然，这个 APPBarButton 完全可以用 Button 代替，两者的功能并无差异，但是在实际运用中 Button 按下去之后有一个偏向一边的效果，不符合触摸设备开发的美观要求，于是用更加合适的 AppBarButton 来代替。APPBarButton 有个 APPBarButton.Icon 属性，这个属性是用来设置按钮图标的,注意 FontIcon 里的字符属性是 Glyph,FontFamily 还是选择 Segoe MDL2 Assets。APPBarButton 的点击事件和 Button 一样，都是 Click=“新建点击事件”,选中点击事件名称按下 F12 可跳转到事件对应的后台代码。

3.2.7 WebView控件

WebView 提供了一个展示互联网页面内容而不用跳转到浏览器的方法，这个控件的内容简单来说就是直接显示网页。在这里 Home 页面和 KylinPage 页面便是直接嵌套的一个 WebView 控件，如果单纯想要显示一个页面的内容，则需要声明一个 WebView 控件，然后在页面的构造方法中添加 WebView 的 Navigate（Uri）方法，使页面在启动的时候就加载这个地址的内容。

3.2.8 ProgressRing控件

这个控件是一个圆形的旋转的控件，这个控件的应用场景十分广泛，比如在后台加载处理一些事物或者提交等待反馈的时候，可以吸引用户注意力，用来提醒用户这个应用正在进行而不是卡顿或者卡死了。在视觉上起到了提示的作用。该控件的主要属性是 IsActive，有 True 和 False 两种取值，用来指定空间是否以活动的动画来显示。如果该控件没有被激活，那么它是不会显示出来的。此外，该控件只是起到一种指示作用，并不会触发任何事件。ProgressRing 的样子像这样，如图 3.2。

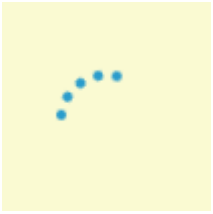


图 3.2 ProgressRing 样式

3.2.9 ListView与GridView控件

这两个控件都是用来显示内容或数据的集合的，可以根据需要显示想要的数量的数据。两者有属性方法大都相同，不同之处在于 ListView 是以类似于 ComoBox 那样在垂直方向上纵向显示一列数据，而 GridView 就更加灵活了，它取代了 ListVIew 单一的纵向排列方式，用横向的网格取而代之，而且当数据超过一行时会自动换到下一行，还会随着面板的大小而改变自己的排列方式，因此在显示大量数据的时候，我们有显示使用 GridView。两种控件都支持数据绑定，两种控件也都支持选中，在选中的同时选中项会多出一个标记表示该选项已被选中，但是如果我们只是想绑定数据源进行显示信息而不想在选中时是出现这个对号标记，那么可以设置 SelectionMode 属性的值为 None；同时，如果我们想支持多重选择，可以设置这个属性为 Multiple；最后，还可以添加一个 Click 事件，这时我们首先要设置 IsItemClickEnable 属性为 True,然后就可以添加 ItemClick 事件了。

3.2.10 Pivot控件

Pivot（枢轴）控件是微软在 WP8 开发时推出的具有 Windows8 风格的

一种控件，在 WP8 时代的地位可以说是相当于如今的 SplitView，也是一种非常经典的控件，它的主要作用就是左右滑动切换页面，非常符合触屏用户的使用习惯，Pivot 的基本层次结构如图 5.3。其中 PivotItem 是每一个分页项，Header 属性是分页项的标题。

```
<Pivot>
    <PivotItem Header="标题1">
        <StackPanel>
            <!--页面内容-->
        </StackPanel>
    </PivotItem>
    <PivotItem Header="标题2">
        <StackPanel>
            <!--页面内容-->
        </StackPanel>
    </PivotItem>
    <PivotItem Header="标题3">
        <StackPanel>
            <!--页面内容-->
        </StackPanel>
    </PivotItem>
</Pivot>
```

图 3.3　Pivot 基本层次结构

3.2.11 Image控件

这个控件的用法比较简单，就是在页面中显示一个图片，打上<Image>标签后，加一个 Source（路径）属性就可以显示图片了，例如 Source=“Assets/images/logo.png”。在这里需要注意，一般来说路径都是相对路径，所需要的图片不要直接复制到工作空间文件夹内，而是需要通过 VS 中的解决方案资源管理器粘贴到相应的文件夹才能生效，如果担心自己的路径拼写有错误，也可以直接从属性面板直接选择拷贝进来的照片，我在刚开始用这个空间的时候就因为路径问题迟迟不能显示图片，最后通过属性面板选择之后发现原因是自己的页面路径没有对相应的文件夹资源进行引用，所以当路径不正确的时候请及时通过属性面板解决，这样可以快速找到问题所在并及时解决。其中的 MaxWidth 属性是最大图片宽度，默认屏幕有多大，它就有多大，且比例与原图一样，我们使用这个属性来限制图片的显示大小。此外，还有一个 Strength 属性，用来规定图片的伸展性。

3.2.12 ScrollViewer控件

在把一个比较长的页面部署到设备上以后，发现由于内容太多，页面又不能滚动，导致内容不能完全显示。这时，ScrollViewer 便派上用场了，ScrollViewer 是一个支持滚动的控件，属性默认纵向滚动，常用属性还有ScrollBarVisibility，当值为 Auto 时显示滚动条，并在停止滚动时自动隐藏滚动条，值为 Hidden 时始终隐藏滚动条，Disable 时不滚动，Visible 的效果与 Auto 相同，嵌套结构如图 3.4。

```
<PivotItem Header="标题2">
    <ScrollViewer VerticalScrollMode="Auto"
                  VerticalScrollBarVisibility="Auto">
        <StackPanel>
            <!--页面内容-->
        </StackPanel>
    </ScrollViewer>
```

图 3.4　ScrollViewer 嵌套

3.2.13 ToggleSwitch 控件

该控件只有两种状态 On（打开状态）和 Off（关闭状态）。可以在 XAML 代码中设置处于 On 和 Off 状态下的文字以及标题的文字。此外，还可以添加 Toggled 事件用于监听状态的转换并处理这种转换。在本软件中的应用如图 3.5。

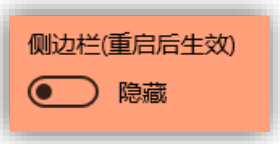


图 3.5　ToggleSwitch 样式

第四章 系统设计与实现

4.1 页面导航设计

4.1.1 页面导航简述

大多数应用都不只有一个页面，当有两个以上的页面时，我们就得结局不同页面间的跳转问题，如果有一些页面用户访问不到，那么这个产品的设计可以说是不成功的甚至是失败的了。

UWP 应用的结构如图 4.1 所示，Application、Window、Frame、Page 层层嵌套：分别代表应用程序、窗口、框架、页面。而默认生成的 MainPage 就是一个 Page 页面。

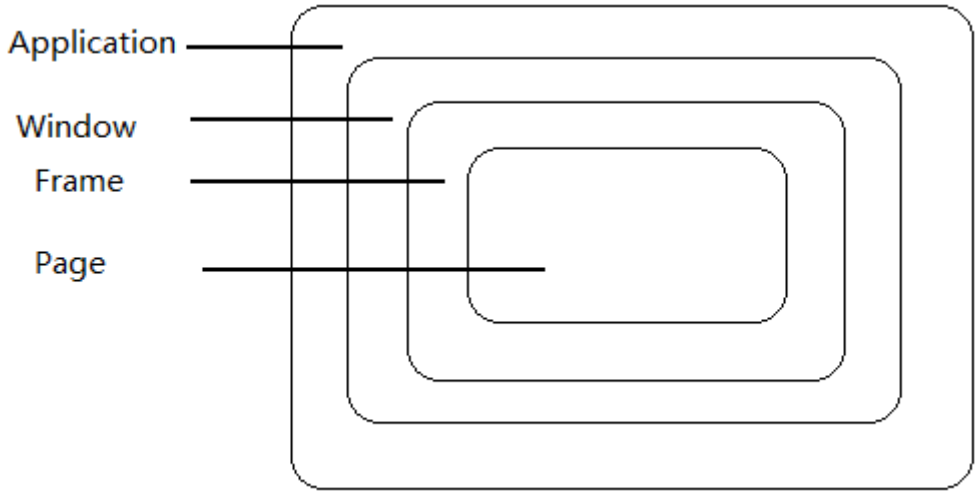


图 4.1 应用结构

如果把 Application 比作一个容器的话，那么 Window 负责应用程序的 UI 表现，当程序启动后，系统会自动初始化 Application 的 Window。

应用程序的导航体系则是由 Window 内的 Frame 支撑的，Frame 是多个 Page 的容器，我们需要手动创建一个 Frame，并把它嵌入到 Window 中。

一个应用程序至少有一个 Frame，一个 Page，通常情况下会有多个 Page，但是很少有多个 Frame 的情况。Frame 类负责在多个页面间的而导航和数据传递。

4.1.2 Frame类

Frame 类可以说是一个 APP 的导航体系的核心了，它的本质是一个 Content 控件，它不仅能够显示 Page，最重要的是在程序中的不同 Page 之间提供了相互导航的支持，在有多个页面的程序中，Frame 控件是不可或缺的。Frame 类会维护一个 Page 栈，用来接收用户前进或后退事件的请求，这点有些类似于在常见的各种网络浏览器中的前进和后退功能，下面说一下 Frame 类的主要方法、属性、以及事件：

属性：

CanGoBack：该属性可以判断在 Frame 的导航栈中有没有存在“上一个页面”可以让用户导航到上一个页面。

CanGoGorward：该属性可以判断在 Frame 的导航栈中有没有存在“下一个页面”可以让用户导航到下一个页面。

Frame 控件比较特别，它没有其他控件那样的所具有的一些触发事件，从而让我们对这些触发操作进行响应，但是 Frame 控件提供了以下这些方法作为补充，从而供开发者进行调用来实现页面间的相互导航。

方法：

Navigate（）导航至另一个页面，这个方法需要用 typeof（TargetPage）来作为传入参数，除此之外，它还可以传递一个 Object 类型的参数。这个方法有两个重载。第一个重载方法的参数接收一个具体的页面类型。第二个重载方法除了接受一个具体的类型外，还接收一个 object 参数，通过这个参数可以给将要导航至的页面传递参数。

GoBack（），如果有上一个页面的话，调用此方法可以使页面导航到导航栈里的上一个页面。但是导航栈中如果不存在这个页面，那么导航会失败并且抛出异常，所以我们在导航的之前应当先用 CanGoBack 属性判断一下是否存在上一个页面。

GoForward（），如果有下一个页面的话，调用此方法可以使页面导航到导航栈里的下一个页面。但是导航栈中如果不存在这个页面，那么导航会失败并且抛出异常，所以我们在导航的之前应当先用 CanGoForward 属性判断一下是否存在下一个页面。

事件：

Navigating：当要发生页面跳转时触发此事件。

Navigated：当目标页面已找到且可用时触发此事件。

当从第一个页面导航到第二个页面时，首先触发 Frame 的 Navigating 事件，然后触发第一个页面的 OnNavigatingFrom 事件，随后触发 Frame 的 Navigated 事件，再触发第二个页面的 OnNavigatedTo 事件。

4.2 页面内容设计

4.2.1 页面简述

掌握了页面导航的方法，下一步就可以放心大胆的进行页面的设计了。依次点击 VS 菜单栏的项目—添加新项—Visual C#—空白页，给要新建的页面取一个比较容易理解的名字，点击添加，这样便可以创建多个页面了。这里一定要注意在新建页面的时候选择“空白页”而不是“XAML”视图，如果选择“XAML 视图”那么将不会在 Frame 中显示。我在刚开始些项目的时候就因为这点儿问题没有注意到，致使长时间没有解决页面导航，耽误了工程进度，非常令人恼火。接下来重点介绍各个页面的设计过程。整个资源如图 4.2 所示。

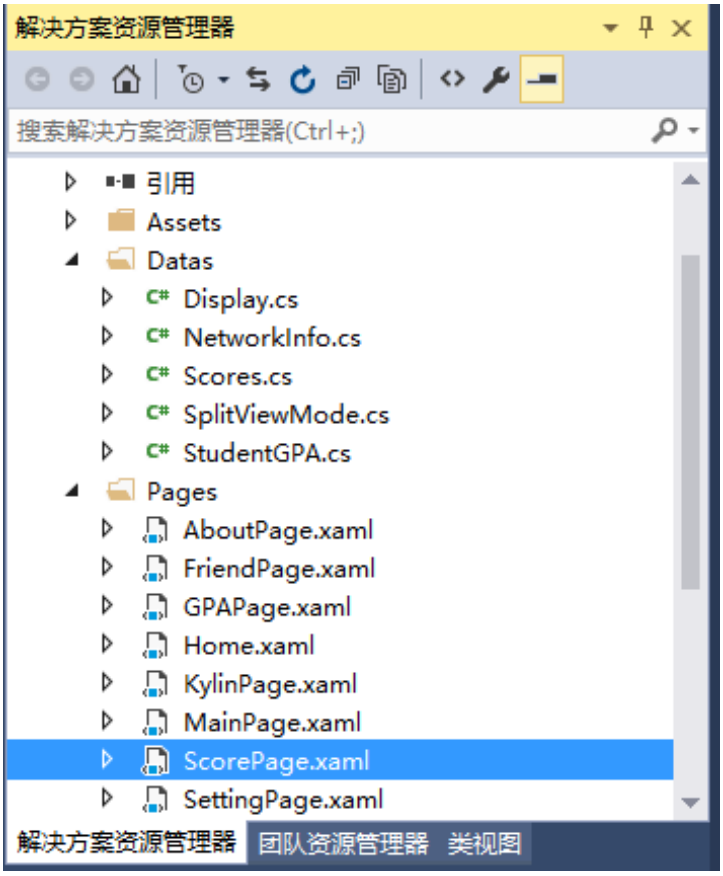


图 4.2 资源项目示意

4.2.2 资源定义

在 UI 设计时，我们可能会重复使用一些属性相同的元素，比如多个 Button，我们需要的宽、高、字体都一致，如果重复定义则显得比较繁琐，而且会使得代码比较冗余，可读性下降。因此，我们定义一组资源，在需要的时候直接引用，如果是全局的属性，在整个 APP 中都需要引用的时候，我们将资源定义在 App.xaml 中，如下代码：

```
<Application.Resources>
        <Style TargetType="Button" x:Key="ImageButtonStyle">
            <Setter Property="Height" Value="50" />
            <Setter Property="FontFamily" Value="Segoe MDL2 Assets" />
            <Setter Property="FontSize" Value="36" />
        </Style>
        <Style TargetType="ProgressRing" x:Key="ProgressRingStyle">
            <Setter Property="HorizontalAlignment" Value="Center" />
            <Setter Property="VerticalAlignment" Value="Center" />
            <Setter Property="Height" Value="50" />
            <Setter Property="Width" Value="50" />
        </Style>
    </Application.Resources>
```

以上代码定义了一个 Button 的样式、一个 ProgressRing 的属性。在 Button 属性中的设置中，给这个样式取名为：ImageButtonStyle，然后分别设置了 Height、FontFamily、FontSize 属性。在需要的时候，只要这句代码 Style=“{StaticResource ImageButton}”即可，其中，双引号内的{}表示绑定，这也是在应用开发中比较重要的概念。

```
<Button x:Name="SearchButton"
                        Style="{StaticResource ImageButtonStyle}"
                        Content="&#xE71E;"
                        RelativePanel.AlignRightWithPanel="True"
                        Click="SearchButton_Click"/>
```

4.2.3 自适应触发器

自适应触发器的作用是根据应用外观的大小来调整布局，也就是响应式布局。自适应触发器一般在 XAML 页面定义，格式如下：

```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup x:Name="MyVisualStateGroup">
        <VisualState x:Name="Phone">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="0"></AdaptiveTrigger>
            </VisualState.StateTriggers>
            <VisualState.Setters>
                <Setter Target="MySplitView.IsPaneOpen" Value="False"></Setter>
                <Setter Target="MySplitView.DisplayMode" Value="Overlay"></Setter>
            </VisualState.Setters>
        </VisualState>
        <VisualState x:Name="Tablet"...>
        <VisualState x:Name="PC">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="800"></AdaptiveTrigger>
            </VisualState.StateTriggers>
            <VisualState.Setters>
                <Setter Target="MySplitView.IsPaneOpen" Value="False"></Setter>
                <Setter Target="MySplitView.DisplayMode" Value="CompactOverlay"></Setter>
            </VisualState.Setters>
        </VisualState>
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

自适应触发器有两个主要部分组成：Trigger 和 Setter。Trigger 是触发条件，Setter 是出发后的操作，在上述代码中，设置了三组状态，以最后一组为例子，触发条件是窗口宽度大于 800，此时对应的操作是设置 Pane 的展示状态为 CompactOverlay，Pane 默认不展开。

4.2.4 MainPage页面

MainPage 是创建工程后默认生成的一个页面，也是 APP 启动后的默认载入页面，那我就直接把这个作为主页面，在里边搭好框架，设置一个 Frame，在这个 Frame 中对各个页面进行来回切换导航，其主要的 XAML 结构如图 4.3。

如图所示，整个页面<Page>内包含了一个<Grid>，<Grid>又通过<RowDefinition>定义了两个行，相当于 HTML 中的 table 的两个行。在这两个行里边，分别放入一个 RelativePanel 和一个 Grid。

其中 RelativePanel 里边包含了三个 Button、一个 TextBlock，三个 Button 分别是 Hamburger Button（汉堡包菜单按钮）、Back Button（返回

键按钮）、Setting Button（设置菜单按钮），TextBlock 居中用于显示当前页面的信息。这里边有些控件使用了的附加属性，比如 RelativePanel.AlignRightWithPanel="True"这条 XAML 语句表示当前控件的右端始终于面板保持对齐，无论面板如何变换，这个控件将始终在面板在最右边。

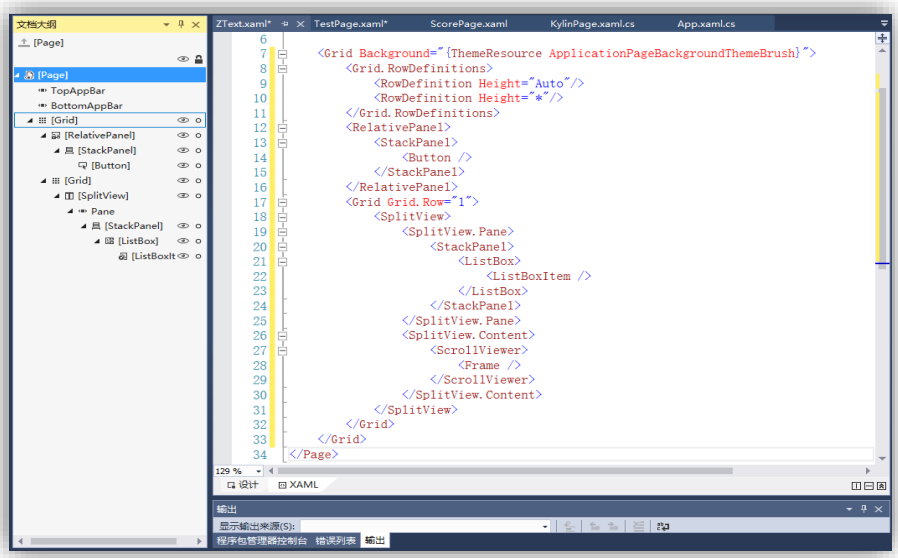


图 4.3 XAML 结构

下边的 Grid 有个属性 Grid.Row=1,表示这个表格位于第二行中，我们在这个 Grid 中又嵌套了一个 SplitView，而这个 SplitView 实现了整个 APP 的导航，可以说是整个项目的关键。Splitview 的关键部分在于 SPlitview.Pane，先选取 Pane 的一部分代码展示如图 4.4。

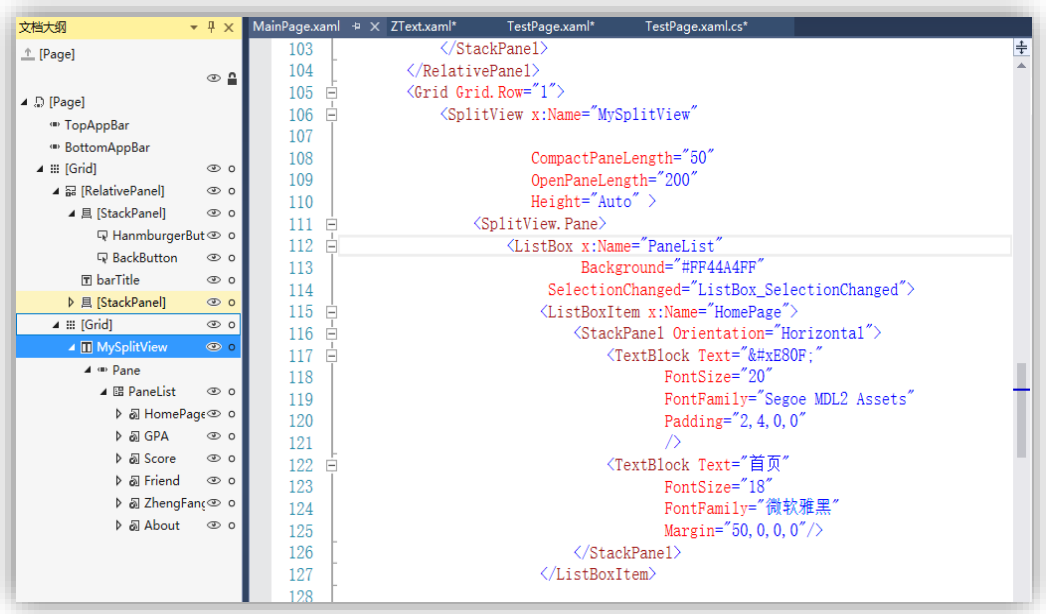


图 4.4 SPlitView.Pane 结构

在文件大纲的帮助下可以看到整个 Pane 的逻辑结构，Pane 里边嵌套了一 ListBox，ListBoxItem 里边嵌套了一个 StackPanel，StackPanel 里边又嵌套了两个 TextBlock，一层套一层。这里说一下为何要在 ListBoxItem 中嵌套 StackPanel，因为 ListBoxItem 的 Content 属性只能设置一次，但是我们需要两个 TextBlock 分别显示图标和文字，所以需要一个 StackPanel 把他们套在一起。所以说，灵活掌握各种控件的嵌套关系对于设计好一个 UI 是至关重要的。

SPlitview.Content 内先嵌套了一个 ScrollViewer，这样页面就能在需要的时候实现滚动了，然后又里边又嵌套了一个 Frame，并取名为 MyFrame。接下来，我们的页面导航都要靠这个 Frame 来完成了，虽然只有短短的一个声明代码，却发挥了中枢的作用。

转到后台代码，首先，便是 Hamburger 菜单控制 SPlitVIew 的 Pane 部分开关的实现，这里通过取反运算符实现了 Pane 的开关，如下代码所示：

```
private void HanmburgerButton_Click(object sender, RoutedEventArgs e)
        {
            MySplitView.IsPaneOpen = !MySplitView.IsPaneOpen;
        }
```

当我们点击 Pane 中的 ListBoxItem 时，需要导航至我们想要去的页面，这时我们在 ListBox 上添加一个 SelectionChanged 事件，每次选项改变时触发，主要代码如下：

```
private void ListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if(HomePage.IsSelected)
            {
                MyFrame.Navigate(typeof(Home));
                barTitle.Text = "首页";
                BackButton.Visibility = Visibility.Collapsed;
            }
            else if(ZhengFang.IsSelected)
            {
                MyFrame.Navigate(typeof(KylinPage));
                barTitle.Text = "万能的麒麟";
                BackButton.Visibility = Visibility.Visible;
            }
```

MyFrame 中要加载的 Page 在构造函数中定义，这样当程序加载的时

候便可以实现对页面的导航，代码如下：

```
public MainPage()
        {
            this.InitializeComponent();

            MyFrame.Navigate(typeof(Home));
        }
```

返回键和设置键的事件则比较简单，就是调用一次 Frame 的 Navigate 方法，然后导航至要达到的页面。对于 Frame 的前进后退需要先做一下判断，这个在前边已经提到，代码如下：

```
private void BackButton_Click(object sender, RoutedEventArgs e)
        {
            if(MyFrame.CanGoBack)
            {
                MyFrame.GoBack();
            }
        }
```

整个 MainPage 的效果如图 4.5 所示。

图 4.5 MainPage 页面样式

4.2.5 Home以及KylinPage页面

这两个页面比较简单,就是嵌套了一个 WebView 和一个 ProgressRing，代码如下：

```
<Grid Background="LightPink">
        <WebView x:Name="sdut"
                 NavigationCompleted="sdut_NavigationCompleted" />
        <ProgressRing x:Name="MyProgressRing"
                      IsActive="True"
                      Style="{StaticResource ProgressRingStyle}"/>
</Grid>
```

说下 ProgressRing，这里有个需要注意的地方，一是 IsActive 属性在一开始设置为 True，这样当开始加载这个页面的时候便有一个旋转的环的效果。然后，我在 WebView 中添加了一个 NavigationCompleted 事件，而它的作用就是在页面加载完成后将 ProgressRing 的 IsActive 属性设为 False，这样进度环变消失了，留下干净的页面。

转到后台代码，发现在构造函数中调用了一个 IsConnectedToInternet 方法，该方法在 Datas 文件夹中的 NetworkInfo 类中定义，所以先要添加对这个类的引用。这个类的定义如下代码所示：

```
public class NetworkInfo
    {
        public static bool IsConnectedToInternet()
        {
            bool connected = false;
            ConnectionProfile cp = NetworkInformation.GetInternetConnectionProfile();
            if(cp!=null)
            {
                NetworkConnectivityLevel cl = cp.GetNetworkConnectivityLevel();
                connected = cl == NetworkConnectivityLevel.InternetAccess;
            }
            return connected;
        }
    }
```

这个类的定义用到了 Windows.Networking.Connectivity 命名空间，需要在使用前进行引用。用来判断设备连接到网络，避免在没有网络时出现

不必要的异常。网络连接的状态信息可以用 NetworkInformation 类提供的方法进行判断，其中的 GetInternetConnectionProfile 方法可以获取网络连接的配置信息（ConnectionProfile），ConnectionProfile 类中的 GetNetworkConnectivityLevel 方法能够返回当前系统连接级别（NetworkConnectivityLevel），当级别为 InternetAcess=3 的时候，表示连接到本地和 Internet。上边所示代码就是对是否有 Internet 连接，如果链接到互联网返回 true，否则返回 false。

我们在页面的构造函数中对该方法进行引用，如果无网络连接，则弹出提示框并不进行加载，否则加载页面。代码如下：

```
string source = "http://youth.sdut.edu.cn";
        string msg= "咦，发生了神马￣△￣";
        public Home()
        {
            this.InitializeComponent();

            if(!NetworkInfo.IsConnectedToInternet())
            {
                msg = "木有网啊，请检查网络连接(°ー°〃)";
                Display.ShowMessageDialog(msg);
            }
            else
            {
                NavigateWebView(source);
            }
        }
```

其中，ShowMessage 方法也是定义在 Datas 中的，在 Display 类中。

KylinPage 的设计与 Home 相同，在这里就不做过多介绍了。

4.2.6 ScorePage、GPAPage和FriendPage页面

ScorePage 的整体效果如图 4.6 所示。

在搜索框中点击搜索后，搜到的结果在下方的 GridView 中显示，点击每一个 GridViewItem 则会显示点击成绩的详情。这个功能的实现依赖于 ScorePage，ScorePage 整体是一个分为两行的 Grid，在 Grid 的第一行放一个 RelativePanel，RelativePanel 中又嵌入一个 TextBox 作为搜索框，一个

Button 作为搜索按钮。Grid 的第二行则先嵌套一个 ScrollViewer，然后再嵌套一个 GridView，显示查询到的成绩。由于没能获取到数据来源网站的接口信息，所以该功能的实现主要依靠对网页信息的抓取以及后续处理。

图 4.6 ScorePage 页面样式

转到后台代码，首先是 button 的 click 事件代码如下：

```
private void SearchButton_Click(object sender, RoutedEventArgs e)
        {
            if(NetworkInfo.IsConnectedToInternet())
            {
                gpaProgressRing.IsActive = true;
                string target = url;
                target += SearchBox.Text;
                target += ".lg";

                displayGPA.Items.Clear();
                GetHtml(target);
            }
            else
            {
                msg = "木有网啊，请检查网络连接(°ー° 〃)";
                Display.ShowMessageDialog(msg);
            }

        }
```

在点击搜索按钮后，首先判断是否有 Internet 级别的网络连接，如果有连接，则首先激活 ProgressRing，产生 Ring 旋转的动态效果，然后产生要请求的页面的完整 URL，随后调用 GetHtml 方法，得到想要获得的成绩信息。故这个 GetHtml 方法是获取目标信息的关键，我们看一下 GetHtml 方法的主要代码：

```
public async void GetHtml(string url)
        {
            try
            {

                WebRequest request = HttpWebRequest.Create(url);

                WebResponse response = await request.GetResponseAsync();
                Stream stream = response.GetResponseStream();
                var result = "";
                using (StreamReader sr = new StreamReader(stream))
                {
                    result = sr.ReadToEnd();
                }
                HtmlDocument doc = new HtmlDocument();
                doc.LoadHtml(result);

                var nodes = doc.DocumentNode.SelectNodes("//p/small");

                if (nodes==null)
                {
                    msg = "请检查输入的学号无误~~~~(> _ <)~~~~";
                    gpaProgressRing.IsActive = false;
                    Display.ShowMessageDialog(msg);
                }
                else
                {
                    gpaProgressRing.IsActive = false;

                    foreach (var item in nodes)
                    {
                        ListViewItem lv = new ListViewItem();
                        lv.Content = item.InnerText;
                        displayGPA.Items.Add(lv);

                    }
                }
            }
```

在该方法中，首先是使用 WebRequest 和 WebResponse 获取到目标页面的源码信息，然后把这些信息转换为一个 HTML 文档信息类型的数据，随后，使用 HtmlAgilityPack 这个解析 html 以及采集页面信息的神器进行对页面信息的提取。以下对本功能的核心部分 HtmlAgilityPack 做一些说明。

HtmlAgilityPack 是一个开源的、基于.Net 的、第三方的解析 HTML 元素的微型类库，。在使用这个类库之前，解析 HTML 文档一般都采用正则表达式，然而正则表达式语法复杂晦涩，不利于短期高效的开发。在使用了 HtmlAhilityPack 后，解析 HTML 变得更加高效和准确，提高了开发效率和应用性能。

由于这是一个第三方的类库，所以在使用前需要添加对该类库的引用，右击解决方案管理器中的引用，管理 NuGet 程序包，打开 NuGet 包管理器，搜索 XML.XPath.XDocument，点击安装。以后有类似的引用，都可通过此方法添加。如图 4.7。

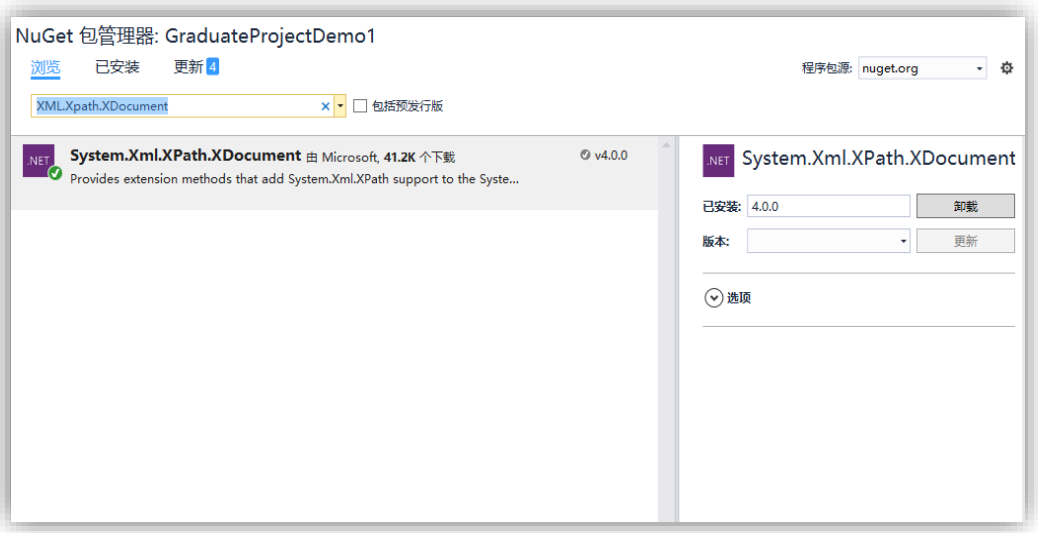


图 4.7 NuGet 包管理器

除此之外，还需引用一个动态链接库文件，即 HtmlAgilityPack.Win10.dll，这个库文件可以在网上下载到，添加方式略有不同，分别是引用—添加引用—浏览—本地文件确定引用。添加完引用后，在 C#类文件的开头添加引用命名空间 using HtmlAgilityPack；就可以在后续处理中使用 HtmlAgilityPack 了。

在 HtmlAgilityPack 中，节点的定位方法大多依赖于 XPath 表达式。

对于 XPath 的官方解释如下：“XPath 使用 路径表达式在 XML 文档中进行导航”，也就是说 XPath 语法使用路径表达式来选取 XML 或 HTML 文档中的节点或节点集，而节点又是通过沿着路径或者 step 来选取的。若想获取想要得到的节点信息，则需要了解 XPath 的语法， XPath 语法最常用的路径表达式如图 4.8 所示。

表达式	描述
nodename	选取此节点的所有子节点。
/	从根节点选取。
//	从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。
.	选取当前节点。
..	选取当前节点的父节点。
@	选取属性。

图 4.8 XPath 常用语法

在这个页面中，通过分析抓取的网页源，发现所有的成绩信息都放在节点<table>—<tbody>—<tr>中，如图 4.9 所示，这样我们可以写一个 XPath 表达式抓取这个节点的信息，然后保存下来。

```
<table class="table table-bordered">
  <thead>…</thead>
  <tbody>
    <tr>
      <td>1</td>
      <td style="width:30%">
        <small>体育(A)Ⅰ</small>
      </td>
      <td>…</td>
      <td>…</td>
      <td>…</td>
    </tr>
```

图 4.9 页面结构

选取该节点信息含有的 XPath 语法的语句如下：

```
var nodes = doc.DocumentNode.SelectNodes("//table/tbody/tr");
```

这样我们就选中了<tr>节点的数据，然后对所获取的节点进行一下判断，如果节点内容为空，则说明没有该同学所对应的成绩信息，或者学号输入有误，这时弹出窗口进行提示。如果节点内容非空，则对于每一个节

点的信息，提取它的文本输出到前端安排好的 GridVIew 中，就这样完成了成绩的查询功能，代码如下：

```
foreach (var item in nodes)
                        {
                            ListViewItem lv = new ListViewItem();
                            lv.Content = item.InnerText;
                            displayGPA.Items.Add(lv);

                        }
```

之前提到过点击 GridView 的 Item 会弹出关于这条成绩的详细信息，这个比较容易实现，就是在 ItemClick 事件中获取点击 Item 的文本信息，然后用消息弹出框的形式把这些信息显示出来，代码如下：

```
private void GridDisplay_ItemClick(object sender, ItemClickEventArgs e)
        {
            string score = (string)e.ClickedItem;
            ShowMessageDialog(score);

        }
        public static async void ShowMessageDialog(String msg)
        {
            var msgDialog = new Windows.UI.Popups.MessageDialog(msg) { Title = "详细信息" };

            await msgDialog.ShowAsync();
        }
```

关于异步编程：在这一部分，代码中存在大量的 Async 和 await 关键字，说明这是异步编程的用法。C#5.0 正式引入两个关键字 Async 和 await，把异步编程的易用度又推到了一个新的高度，在这种方式下，可以通过类似同步方式编写异步代码，极大简化了异步编程模型，成为了 C#5.0 最重要的改进。

说到异步编程，不得不提同步代码，我们平时所写的代码大都是同步的，也就是逻辑下边的代码要等前边的执行完了以后才能继续执行。而这种代码有一个很严重的问题，比如当我们使用同步代码发送一份请求到 Web 服务请以后，在得到服务器的响应信息之前整个程序都会处于一种等待状态，用户无法对应用进行别的操作了，只能等待当前进程执行完毕，

就行应用卡死了一样，这对用户体验来说是灾难性的、不可接受的。比如以下代码：

```
1 WebClient wc = new WebClient(); 2 string s= wc.DownloadString("你的URL");
```

改代码在运行时，必须等 DownloadString 完成，才能进行其余的操作。

为了解决类似问题，.Net Framework 早在.Net 1.0 中就开始就提供了异步特性，直到.Net 4.5 中 Async 和 await 的引进，令这种问题的解决更加优雅。使用 await 关键字之前需要使用 Async 修饰声明的方法，如下代码：

```
public async void GetHtml(string url)
{
    try
    {
        WebRequest request = HttpWebRequest.Create(url);
        WebResponse response = await request.GetResponseAsync();
```

说一下使用 GridView 的好处，它可以根据当前窗体的大小自动调整行的排列方式，这一点非常适合 UWP 跨平台开发的需要。GPAPage 原理与 ScorePage 大致相同，只是展示时用的是 ListView，在这里不做过多说明。更改窗体大小后视图变为图 4.10 这样：

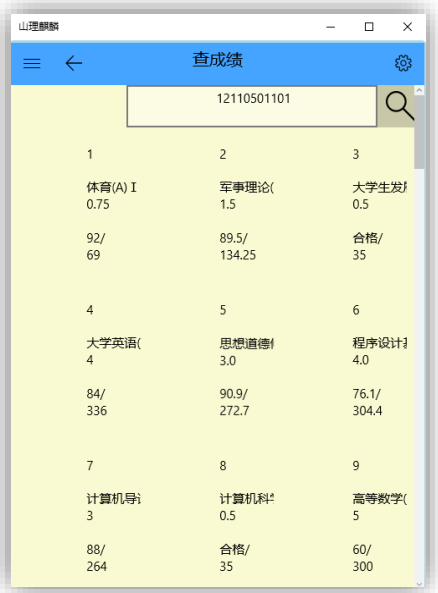


图 4.10 更改大小后的页面

除此之外还有一个 FriendPage 用来查询本校的同名同学，这个页面与上述两个页面的不同之处在于需要将输入的字符转换下编码，这里多增加了一个 UrlEncode，将输入框中得到的 UTF-8 编码的姓名转换成 URL 编码，然后附加到目标 URL 发送到服务器从而获取相应的页面信息，代码如下：

```
public static string UrlEncode(string str)
        {
            StringBuilder sb = new StringBuilder();
            StringBuilder res = new StringBuilder();
            byte[] byStr = System.Text.Encoding.UTF8.GetBytes(str);
            for (int i = 0; i < byStr.Length; i++)
            {
                sb.Append(@"%" + Convert.ToString(byStr[i], 16));
            }
            for (int j = 0; j < sb.Length; j++)
            {
                if (!sb[j].Equals(" "))
                {
                    res.Append(sb[j]);
                }
            }
            return (res.ToString().ToUpper());
        }
```

4.2.7 SettingPage页面

SettingPage 是设置界面，该页面主要由两部分组成，系统设置部分和反馈部分，该页面嵌套了一个 PivotView，两部分分别是两个 PivotViewItem，在第一个 Item 中，设置了是否显示侧边栏，XAML 代码如下：

```
<ToggleSwitch x:Name="setPane"
                                        Header="侧边栏(重启后生效)"
                                    OffContent="隐藏"
                                    OnContent="显示"
                                    Margin="0,10"
                                    Toggled="setPane_Toggled" />
```

系统设置：

侧边栏是否显示的开关使用了 ToggleSwitch 这个控件，其 Toggled 事件用来保存选定的显示方式，当值为 1 时，面板显示，值为 0 时，面板隐藏。首先，定义一个用来保存数据的方法：

```
public void savePaneState(int x)
        {
            ApplicationData.Current.LocalSettings.Values["DisplayMode"] = x;
        }
```

这句代码的作用就是把数据存入应用内，这个方法的使用需要导入命名空间 using Windows.Storage；起作用的那句代码中，“DisplayMode”是自己取的键名，x 是想要保存的值，类型不限，string、int、bool、byte 都可以。

当 Toggled 事件触发后，调用该方法对数据进行保存：

```
private void setPane_Toggled(object sender, RoutedEventArgs e)
        {
            //判断 ToggleSwitch 的开关状态，并进行相关处理
            var toggle = sender as ToggleSwitch;
            if(toggle.IsOn)
            {
                savePaneState(1);//显示
            }
            else
            {
                savePaneState(0);//隐藏
            }
        }
```

数据存下后相对应的就是读取了，读取同样需要导入上述命名空间，定义的

```
public static int readPanelState()
        {
          if (ApplicationData.Current.LocalSettings.Values.ContainsKey("DisplayMode"))
            {
            return    (int)ApplicationData.Current.LocalSettings.Values["DisplayMode"];
            }
            else
                return 0;
        }
```

读取方法：

在读取的时候先判断下要读取的键名是否存在，如果存在就读取出来并返回存储的值，否则返回值 0，默认面板处于关闭状态。读取状态在两个地方得到体现：一是设置界面 Toggle 的显示状态，在初始化页面时便读取这个值，从而决定开关是处于 On 还是 Off 模式，代码：

```
if(SplitViewMode.readPanelState()==1)
                {
                    setPane.IsOn = true;
                }
                else
                {
                    setPane.IsOn = false;
                }
```

另一处读取数据在 MainPage 里应用，判断状态代码，从而决定是否展示 Pane：

```
public MainPage()
        {
            this.InitializeComponent();

            MyFrame.Navigate(typeof(Home));

            BackButton.Visibility = Visibility.Collapsed;

            devicetype = DeviceType();
            if(devicetype==1)
            {
                MySplitView.DisplayMode = SplitViewDisplayMode.CompactOverlay;// 显
示
            }
            if(SplitViewMode.readPanelState()==1)
            {
                MySplitView.DisplayMode = SplitViewDisplayMode.CompactOverlay;// 显
示
            }
            else
            {
                MySplitView.DisplayMode = SplitViewDisplayMode.Overlay;//隐藏
            }
```

反馈页面主要是发送邮件，当软件使用者在发现软件存在的问题或者有改进建议时，可以通过该渠道及时与开发者进行沟通。当用户提交反馈

时，软件会调动系统邮箱应用进行邮件发送，我们需要做的是获取用户的反馈内容并将这些发送信息传递给邮箱，完成邮件发送。

为了使发送格式满足邮件发送的需求，在后台代码加了一些判断，只有所要求的条件全部满足时才能完成提交并调用系统邮件，代码如下：

```
if (questionDescribe.Text == "")
                {
                    msg = "问题描述不能为空_(:з」∠)_";
                    Display.ShowMessageDialog(msg);
                }
                else if (passResult.Text=="")
                {
                    msg = "请输入验证码~~(ˇ ﹏ ˋ)b";
                    Display.ShowMessageDialog(msg);
                }
                else if(passResult.Text!="")
                {
                    temp=int.Parse(passResult.Text);
                    if (temp != key)
                    {
                        msg = "验证码错误~~~~(>_<)~~~~";
                        Display.ShowMessageDialog(msg);
                    }
                    else
                        isTrue = 1;
                }
                if(isTrue==1)
                {
                    await OpenEmailComposeAsync(address, mailSbject, describe);
                }
```

最后，当所有状态验证正确，调用 OpenEmailComposeAsync 方法：

```
private static async Task OpenEmailComposeAsync(string toAddress,string subject,string body)
        {
            var uri = new Uri($"mailto:{toAddress}?subject={subject}&body={body}", UriKind.Absolute);
            await Launcher.LaunchUriAsync(uri);
        }
```

然后跳转到系统邮件，完成发送。

第五章 系统部署与测试

5.1 系统测试

在完成项目主体后，需要进行项目测试从而找出 Bug 并改进。在测试中首先是对于输入错误进行测试，当输入有误时会有弹出框提示，如图 5.1

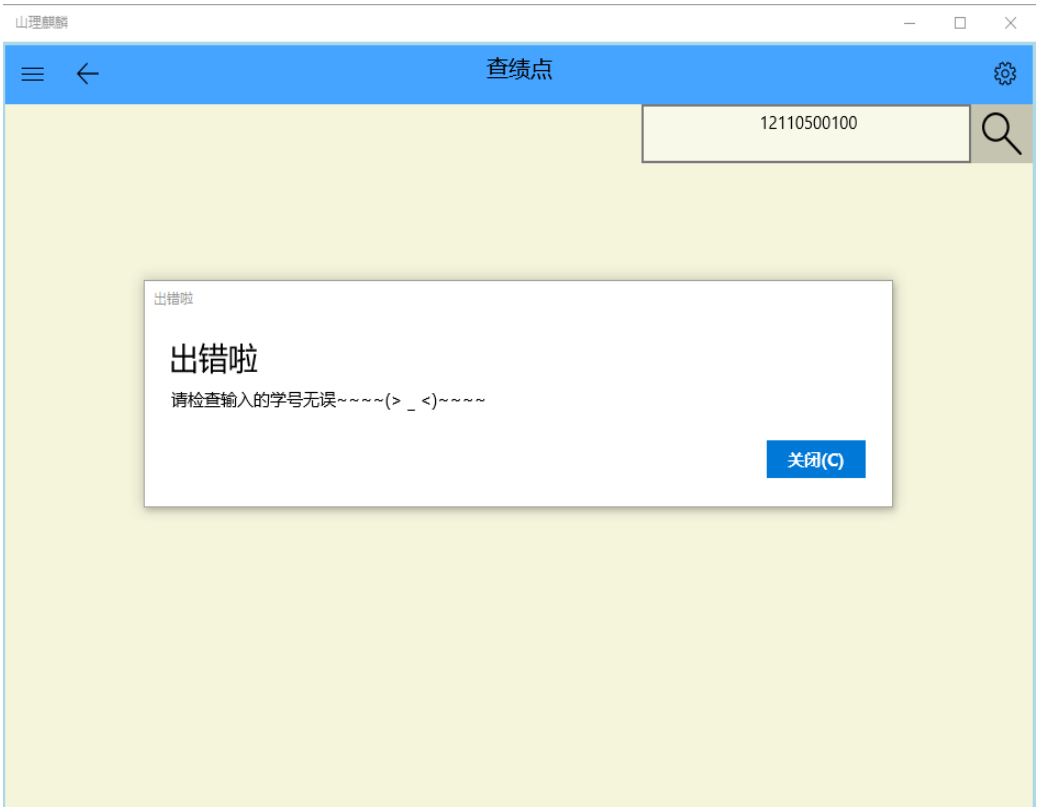


图 5.1 错误提示

邮件的发送页面也进行了相应的测试，在输错验证码或没有填写详情描述时都会有消息提示框。如图 5.2

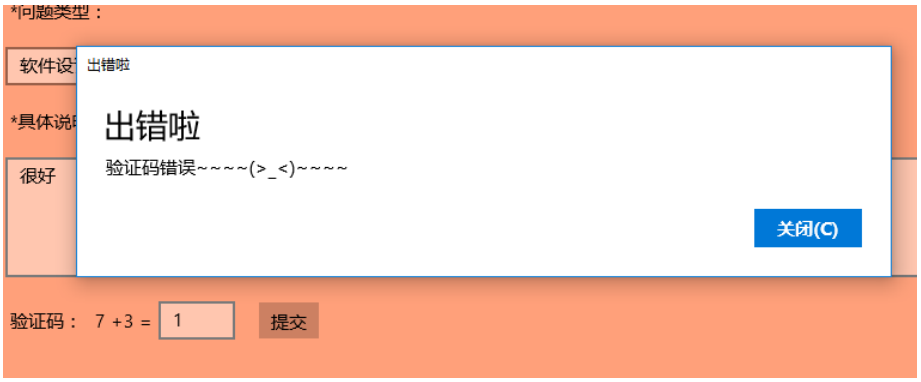


图 5.2 验证码错误提示

5.2 提交说明

经过测试排除一些 Bug，现在应用已经完工了，便可以提交到应用商店了，审核应用一般在 16 小时左右完成。鉴于目前应用商店中应用的数量比较少，微软方面自然想尽快加强生态的建设，巴不得尽快尽早提交更多的应用才是。

简单来说，微软商店的应用审核时间比较短，提交完成后大约一到两个小时便能审核通过，不过从审核通过到发布到应用商店并且能在应用商店中搜索到则需要多达 16 个小时甚至更久，这个时候能做的就只有等待了，如图 5.3。

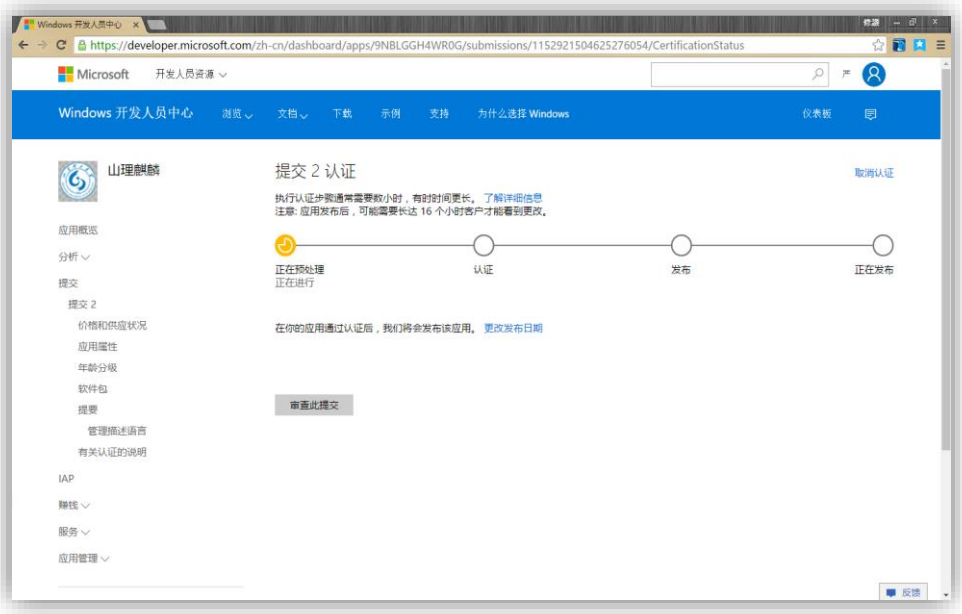


图 5.3 等待认证页面

5.3 保留应用名称

保留应用名称的目的是为了不让你苦心孤诣想出来的好的应用的名称被别人给前抢先霸占了，本着先到先得的原则，我们可以先保留自己喜欢的应用名称，然后在正式提交应用之前将应用名称保留到开打人员中心。除此之外，还可以登录 Microsoft 开发人员中心，打开仪表盘，在这里创建新应用，保留应用名称即可。

5.4 打包

提交到应用商店的内容是以应用程序包的形式，这需要自己手动将项目打包好。打包有两种方式，以是否提交到应用商店为区分：如果仅仅是用作自己内测或者为别人调试为目的的话就不需要登录开发者账号了。如果要提交至应用商店，那么必须登录自己的开发者账号。

打包时右击解决方案资源管理器中的项目名，依次选择应用商店-创建应用程序包，在弹出来的对话框中选择是否提交到应用商店，选择是，点下一步，如图 5.4。

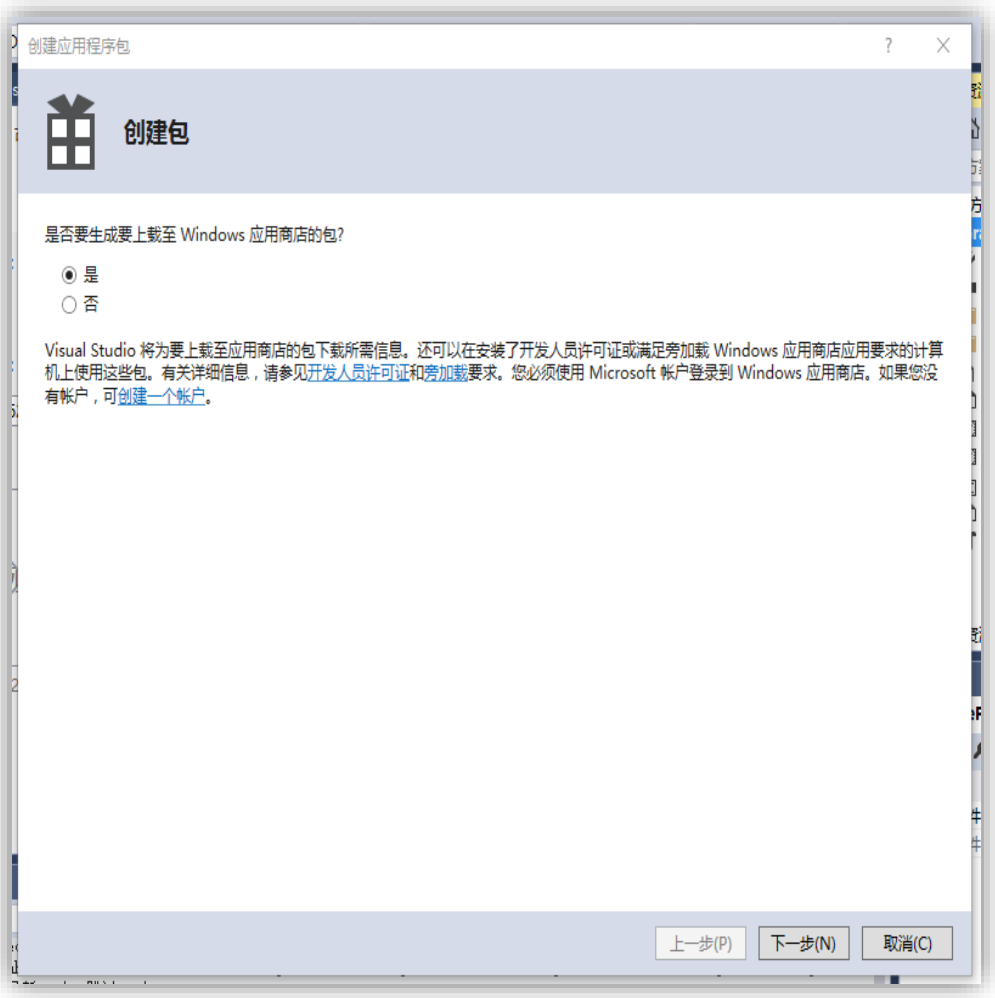


图 5.4 创建包页面

接下来登录自己的开发者账户，然后选择或者创建一个自己保留的应用名称，点击下一步，在出现的选择支持的平台上若只想在移动端可用就只勾选 ARM，在 PC 端安装就选择 x86 和 x64，如果想跨平台安装就全部勾选，如图 5.5。

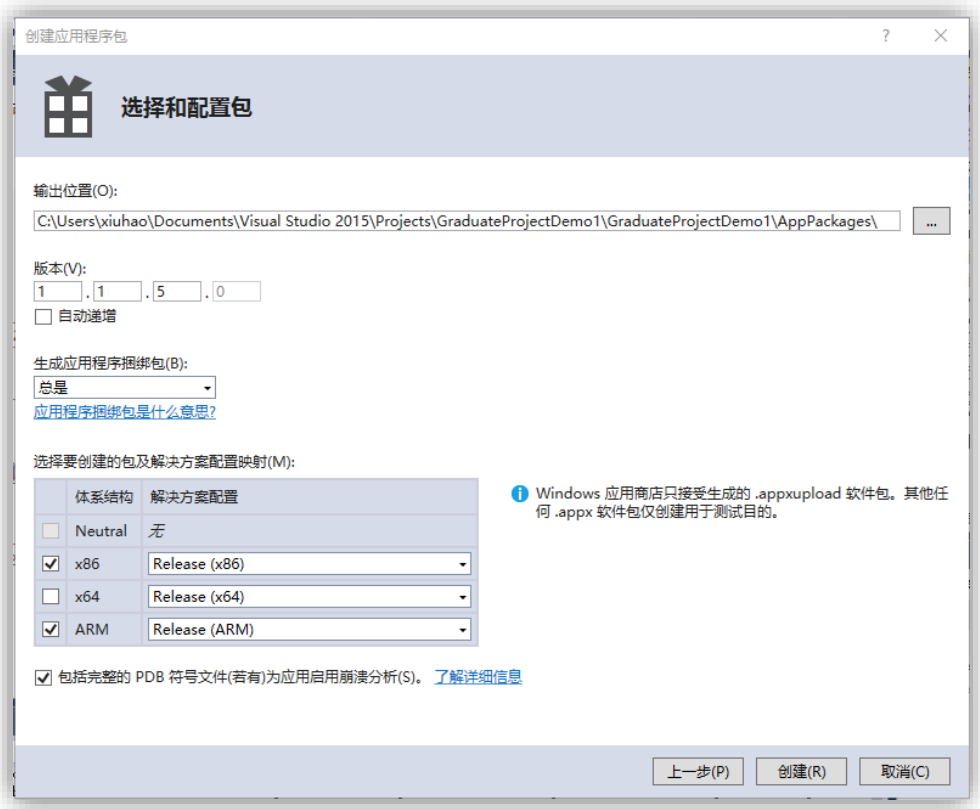


图 5.5 配置生成选项

选择包的生成位置，一般来说选择默认就好，最后编译通过生成一个文件夹和对应的一个 appxupload 格式的文件，这个格式的文件就是我们即将要提交到应用商店的文件。

5.5 提交

登录到 Windows 开发人员中心，点击仪表盘，选择预保留的应用程序名称，开始提交应用，按照所列出的要求一次完善提交信息，全部填写完毕后会显示绿色对号，这说明填写正确符合提交要求，如图 5.6。然后点击最下方的提交到应用商店，此时完成应用提交，微软会在 16 小时左右完成审核，如果审核通过，将发布到应用商店，这时就可以在应用商店搜索和下载应用了。

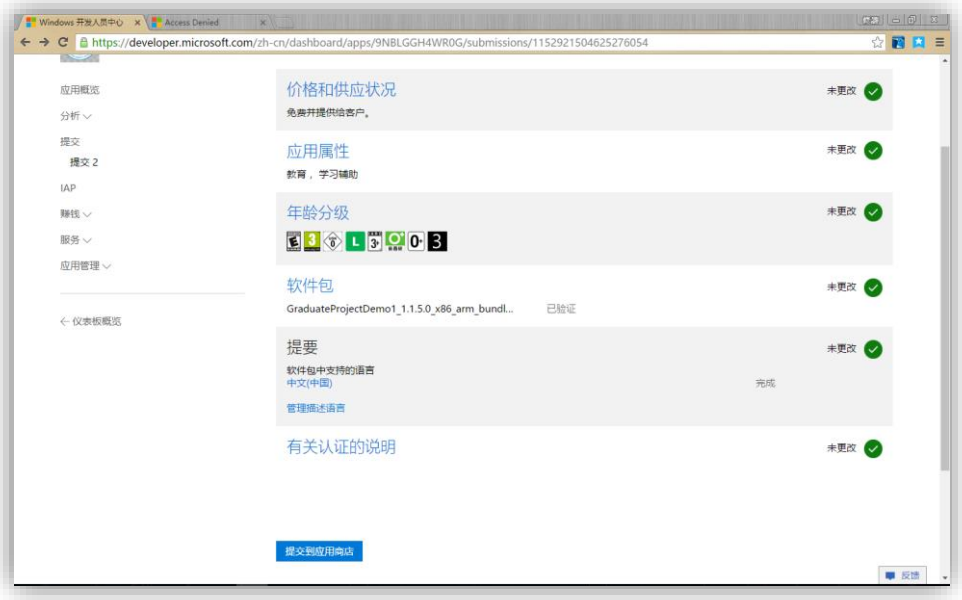


图 5.6 填写提交信息

5.6 遇到的问题

打包的时候会莫名其妙的遇到各种问题导致打包失败，此时一般是 Package.appxmanifest 的问题，大多是改了一些不该修改的配置，这时一般会重启 VS。除了打包错误，别的一些莫名其妙的错误也可以通过重启 VS 或者重启设备试着解决。

在第一次提交到应用商店后，审核了十几个小时微软给出的最终回复是应用违反了一些规定不能上架，这让我感到困惑，仔细看了下邮件觉得可能是因为提交的应用的素材清单有问题，因为里边有些素材是微软默认的没有做修改，于是尝试着替换了所有的图片素材，再次提交至应用商店，这次终于审核通过了，并且在二十多个小时候可以在应用商店中搜索到了。

还有刚开始的时候在商店中可能会搜索不到应用，这有两种可能性，已是在提交的时候不小心选择类在应用商店内不可见，第二是微软商店自己的问题。这时可以试着把应用程序的配置文件、素材库都是这修改下以更新的形式再次提交到应用商店，如果没有填错信息，待微软审核通过了就可以在应用商店内搜索到了，如图 5.7，点击下载即可使用，如图 5.8。

至此，应用开发流程已基本结束，转入后期维护阶段，在此阶段，如果要更新在商店中的应用，可登陆开发人员中心，选择项目并提交所要更新的包，等审核通过后用户便会受到商店的更新通知。

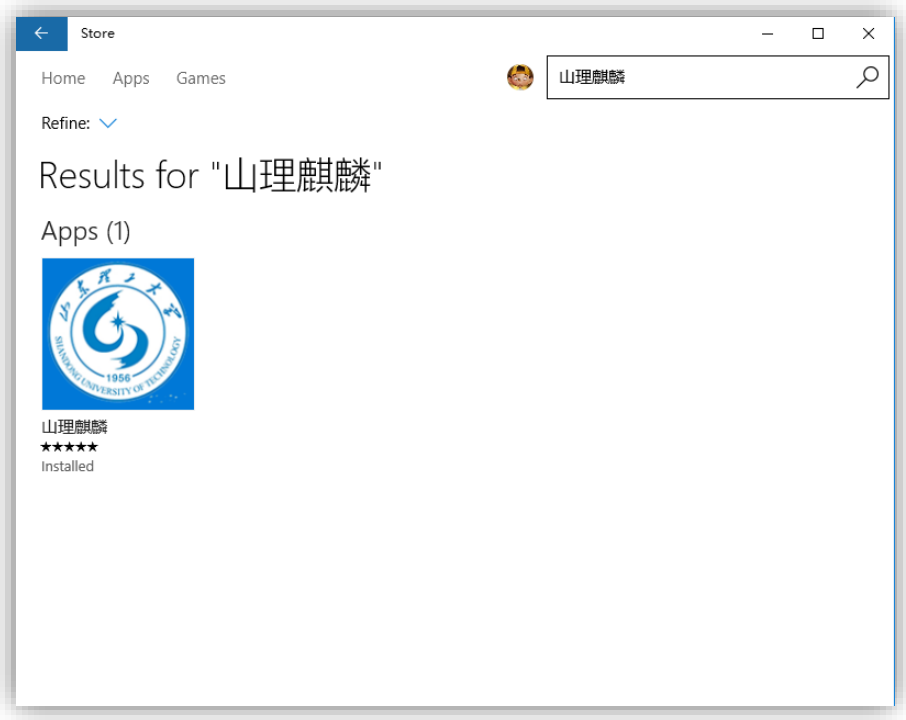


图 5.7 搜索到应用

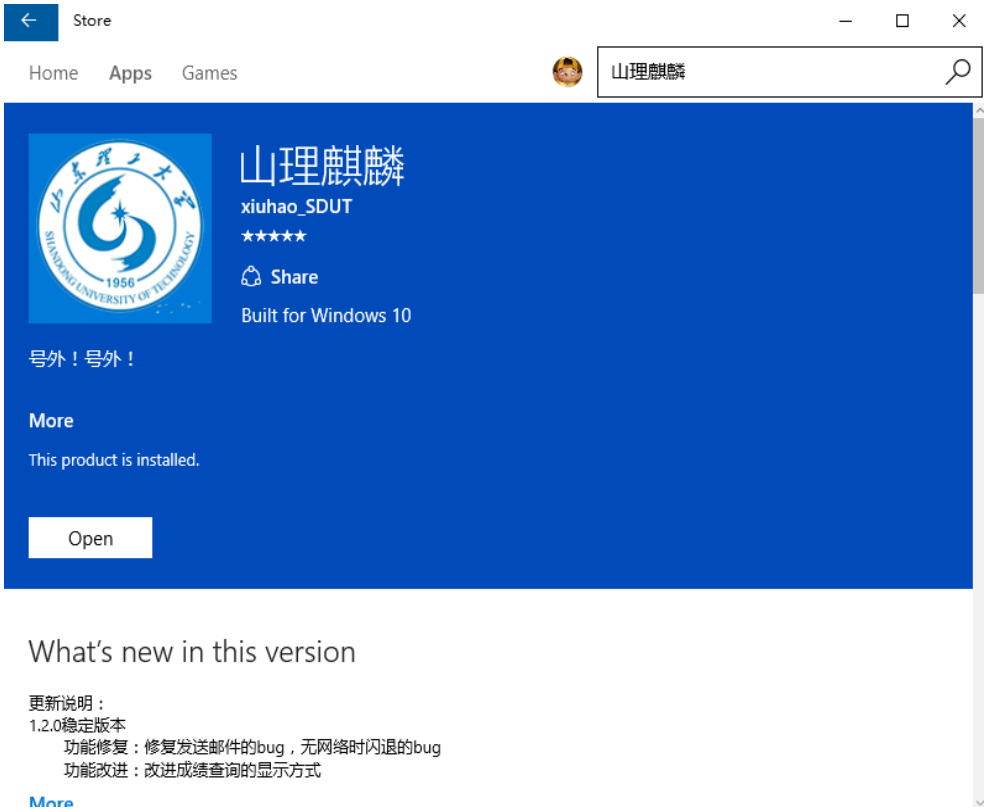


图 5.8 下载页面

第六章 总结与展望

在开始毕业设计之前，我就开始关注 Windows10 的发展，也许是因为自己是 Windows 手机的使用者，故对当前应用商店数量短缺、质量低下的状况比较无奈。加之 Windows 平台缺少一个适用于自己学校特色的校园助手应用，所以最终决定自己写一个符合本校的实际情况的 UWP 应用作为自己的毕业设计项目。

最开始，先在微软的商店搜索一些其他已经发布的本校的应用，研究它们的结构、提供的功能作为自己应用的借鉴。比如“掌控校园”、“南昌航空航天大学”、“掌上重邮-重邮小帮手”，它们都是与自己的学校实情紧密结合，功能上的设置也比较贴切，从而获得了普遍的好评。受此启示，本项目的功能主要有绩点、成绩、四六级，并根据需求进行迭代更新。

开始着手做这个项目后，发现情况并没有想象之中的那么容易。首先是相关资料的匮乏，在图书馆只能找到C#、ASP.Net等这种传统技术书籍，而 Windows10 开发的书籍却一本也没有，最后找到了为数不多的几本相关的 Windows8 开发书籍进行参照。网络资源同样如此，在如今移动开发中，Android 和 IOS 占据了半壁江山，关于 Windows10 的开发资源、博客却是寥寥无几，有时想查找一个控件的用法，却发现大都是 Android 和 IOS 开发的。还好有 MSDN 的官方文案，虽然比较晦涩但是准确，再结合几个专门开发 Windows10 的博主的文章，终于可以边研究边进行编写了。

在此次开发过程中，由于缺乏独立开发经验，所以走了不少弯路。此外，由于本项目几乎是从零开始，所以目前该项目还不够完善，也存在一些缺陷，因此特地设计了反馈渠道。在以后的版本中进行持续维护与更新。

虽然目前 UWP 平台的应用数量、质量较 Android 和 IOS 还有不小的差距，但是随着 Windows10 装机量的持续增长，这一平台正呈现出巨大的发展潜力。而且该平台用户付费习惯较好，只要开发出高质量的应用那么自然不愁获得丰厚的回报。

当本次项目的功能基本完善并发布到商店后，还有感觉小有成就感的。我想这次设计最大的收益就是这对自己的一次检验、对自己实际能力的一次提升，这会令我今后的学习工作受益匪浅。随着微软全球推广 Windows10 的力度的进一步加强，相信会有越来越多的开发者投身 UWP 的开发，也希望我的这篇文章能对那些希望入门 UWP 开发的同学有所帮助。

参考文献

[1] 胡学刚. C#应用开发与实践.北京.人民邮电出版社.2012.12

[2] 周家安.Windows10 应用开发实战.北京.清华大学出版社.2015.09

[3] (美)沃森（Watson,K）.C#入门经典(第 6 版)(齐立波，黄俊伟 译).北京.清华大学出版社.2014.01

[4] 童明.Windows8 应用开发实战.北京.电子工业出版社.2013.12

[5] 周智勋.Windows8 开发实战体验.北京.海洋出版社.2013.08

[6] (匈)诺瓦克.Windows8 应用开发入门经典(杨小东译).北京.清华大学出版社.2014.04

[7] 林政.深入浅出：Windows10 通用应用开发.北京.清华大学出版社.2015.11

[8] 林政.深入浅出：Windows Phone8.1 应用开发.北京.清华大学出版社.2014.10

[9] 严蔚敏，吴伟民.数据结构：C 语言版.北京.清华大学出版社.1997.04

[10] (美) Christian Nagel，Jay Glynn，Morgan Skinner 著.C#高级编程(第 9 版)(李铭译).北京.清华大学出版.2015.01

[11] (美) Jeffrey Richter.CLR via C#(周靖 译).北京.清华大学出版社.2014.11

[12] 邱鹏，陈吉，潘晓明.移动 App 测试实战：互联网企业软件测试和质量提升实践.北京.机械工业出版社.2015.07

[13] (美) Theresa Neil. 移动应用 UI 设计模式(田原 译).北京.人民邮电出版社.2015.01

[14] (美) Joe Fawcett.XML 入门经典（第 5 版）(刘云鹏，王超 译).北京.清华大学出版社.2013.12

[15] 石怡. WPF 使用 XAML 实现对 SQL Server 数据绑定的方法[J]. 电脑开发与应用,2011,10:70-71+74.

[16] 琚彬. 基于 WPF 平台的自定义控件开发[D].西安电子科技大学,2008

[17] 朱斐,任勇. 一种先进的面向对象的开发语言——C~#[J]. 计算机与现代化,2002,11:14-16

致　谢

本论文是在我的导师李盘靖副教授的悉心指导下完成的，无论是论文的选题、框架的设计还是修改定稿，李老师都给予了我详细的指导和不懈的支持。李老师的知识见解与理论素质让我开阔了眼界，而他平易近人、求真务实的生活态度也将深深地影响我。在这里对李老师表示衷心的感谢。

感谢三年来各位老师对我的教导，他们广博的学识和严谨的治学态度将使我受益匪浅，他们的言传身教将使我终生受益。

感谢 402 软件孵化实验室老师和同学多年来的关心和支持！在这次毕业设计中，更进一步加深了彼此的友谊。虽然我们做的项目不尽相同，但我们交流各自的经验，而且临近毕业，难得有机会与同学在一起做自己喜欢的项目，等以后再次相逢，一起挈阔谈宴，这必定是以后的珍贵而又美好的回忆。