现代操作系统应用开发实验报告

姓名：　谷雨

学号：　16341005

实验名称：lab2TodoList

一、参考资料

1. 《win10 uwp 读取保存 WriteableBitmap 、BitmapImage》
   https://blog.csdn.net/lindexi_gd/article/details/54612553
由于需要挂起时保存图片，查了许多资料。这篇文章相对全面。不过主要的作用是让我理解了，ImageSource,WriteableBitmap,BitmapImage 之间的关系。
2. *Creating tiles (XAML) --Microsoft Documents*
   https://docs.microsoft.com/en-us/previous-versions/windows/apps/hh868260(v=win.10)
微软官方文档少有的示例细致完整的教程
3. 《共享数据》-- *Microsoft Documents*
   https://docs.microsoft.com/zh-cn/windows/uwp/app-to-app/share-data
4. 《C# 中 SQLite 使用介绍》
   https://blog.csdn.net/kasama1953/article/details/52658118

二、实验步骤

第四周：

1. 实现挂起和恢复，包括页面路由状态，页面内的临时数据。
2. 实现图片的本地存储。由于挂起事件不足以完成存储，所以采取即刻存储图片的方式

第五周：

1. 实现创建磁贴，动态磁贴，动态图片。

2. 实现应用之间分享数据，动态的图片分享。

3.另外尝试了文档存储数据，而不仅仅是挂起。

第六周：

1. SQLite 数据库的连接和操作

2. 项目逻辑有点混乱，重构。

三、关键步骤截图

第四周：

保存图片，通过 item 的 id

```
public static async void SaveBitmapToFileAsync(WriteableBitmap image, string userId)
{
    StorageFolder pictureFolder = await ApplicationData.Current.LocalFolder.CreateFolderAsync("ProfilePictures", CreationCollisionOption.OpenIfExists);
    var file = await pictureFolder.CreateFileAsync(userId + ".jpg", CreationCollisionOption.ReplaceExisting);

    using (var stream = await file.OpenStreamForWriteAsync())
    {
        BitmapEncoder encoder = await BitmapEncoder.CreateAsync(BitmapEncoder.JpegEncoderId, stream.AsRandomAccessStream());
        var pixelStream = image.PixelBuffer.AsStream();
        byte[] pixels = new byte[image.PixelBuffer.Length];

        await pixelStream.ReadAsync(pixels, 0, pixels.Length);

        encoder.SetPixelData(BitmapPixelFormat.Bgra8, BitmapAlphaMode.Ignore, (uint)image.PixelWidth, (uint)image.PixelHeight, 100, 100, pixels);

        await encoder.FlushAsync();
    }
}
```

读取图片，同样通过 item 的 id

```
public static async Task<WriteableBitmap> GetProfilePictureAsync(string userId)
{

    StorageFolder pictureFolder = await ApplicationData.Current.LocalFolder.GetFolderAsync("ProfilePictures");
    StorageFile pictureFile = await pictureFolder.GetFileAsync(userId + ".jpg");

    using (IRandomAccessStream stream = await pictureFile.OpenAsync(FileAccessMode.Read))
    {
        BitmapDecoder decoder = await BitmapDecoder.CreateAsync(stream);
        WriteableBitmap bmp = new WriteableBitmap((int)decoder.PixelWidth, (int)decoder.PixelHeight);
        await bmp.SetSourceAsync(stream);
        return bmp;

    }
}
```

删除图片，依然通过 id

```
public static async void DeleteProfilePictureAsync(string userId)
{

    StorageFolder pictureFolder = await ApplicationData.Current.LocalFolder.GetFolderAsync("ProfilePictures");
    StorageFile pictureFile = await pictureFolder.GetFileAsync(userId + ".jpg");
    await pictureFile.DeleteAsync();


}
```

主页面挂起和恢复挂起状态时的操作，保存和恢复被选择的 item 以及子页面的路由状态

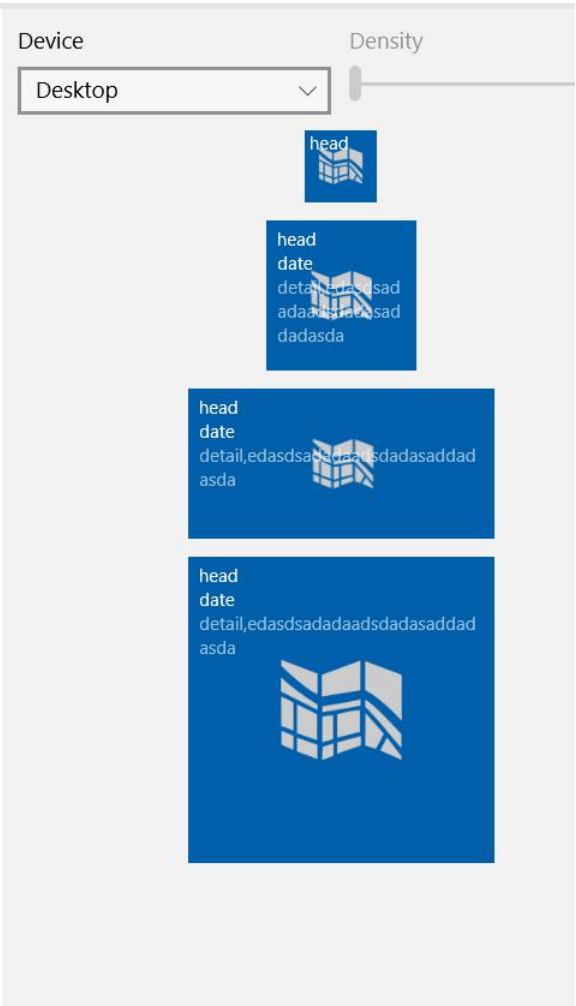
```
ApplicationDataCompositeValue composite = new ApplicationDataCompositeValue();
if(PageView != null&&PageView.Content!=null)
{
    ApplicationData.Current.LocalSettings.Values["pageNavigationState"] = PageView.GetNavigationState();

}

if (ViewModel.selectedItem != null)
{
    ApplicationData.Current.LocalSettings.Values["selected"] = ViewModel.selectedItem.id;
}
```

```
if (e.NavigationMode != NavigationMode.New)
{
    if (ApplicationData.Current.LocalSettings.Values.ContainsKey("selected"))
    {
        foreach (var item in ViewModel.AllItems)
        {
            if (item.id == (string)ApplicationData.Current.LocalSettings.Values["selected"])
            {
                ViewModel.selectedItem = item;
            }
        }

    }
    if (ApplicationData.Current.LocalSettings.Values.ContainsKey("pageNavigationState"))
    {
        PageView.SetNavigationState((string)ApplicationData.Current.LocalSettings.Values["pageNavigationState"]);

    }
    ApplicationData.Current.LocalSettings.Values.Remove("selected");
    ApplicationData.Current.LocalSettings.Values.Remove("pageNavigationState");
```

第五周：

Tile 使用 Notification Visualizer

```
<tile>
  <visual>

    <binding template="TileSmall">
      <image
src="Assets/Apps/Maps/TileLogoSmall.png"
placement="background"/>
<group>
<subgroup>
<text hint-style="caption" hint-
wrap="true" >head</text>
</subgroup>
</group>
    </binding>

    <binding template="TileMedium">
      <image
src="Assets/Apps/Maps/TileLogoMedium.png"
placement="background"/>
<group>
<subgroup>
<text hint-style="caption" hint-
wrap="true" >head</text>
<text hint-style="caption">date</text>
<text hint-style="captionsubtle"  hint-
wrap="true" >
detail,edasdsadadaadsdadasaddadasda</text>
</subgroup>
</group>
    </binding>

    <binding template="TileWide">
      <image
src="Assets/Apps/Maps/TileLogoWide.png"
placement="background"/>
<group>
<subgroup>
<text hint-style="caption" hint-
wrap="true" >head</text>
<text hint-style="caption">date</text>
```




```
{
    string dateText = "" + todoitem.date.Year + '-' + todoitem.date.Month + '-' + todoitem.date.Day;

    XDocument xDoc = new XDocument(
        new XElement("tile", new XAttribute("version", 3),
            new XElement("visual",
                // Small Tile
                new XElement("binding", new XAttribute("displayName", todoitem.title), new XAttribute("template", "TileSmall"),
                    new XElement("imge", new XAttribute("placement", "background"), new XAttribute("src", "ms-appdata:///local/ProfilePictures/" + todoi
                    new XElement("group",
                        new XElement("subgroup",
                            new XElement("text", todoitem.title, new XAttribute("hint-style", "caption"), new XAttribute("hint-wrap", "true"))
                        //  new XElement("text", dateText, new XAttribute("hint-style", "caption")),
                        // new XElement("text", todoitem.description, new XAttribute("hint-style", "captionsubtle"), new XAttribute("hint-wrap", true),
                        )
                    )
                ),

                // Medium Tile
                new XElement("binding", new XAttribute("displayName", todoitem.title), new XAttribute("template", "TileMedium"),
                    new XElement("image", new XAttribute("placement", "background"), new XAttribute("src", "ms-appdata:///local/ProfilePictures/" + todo
```

分享

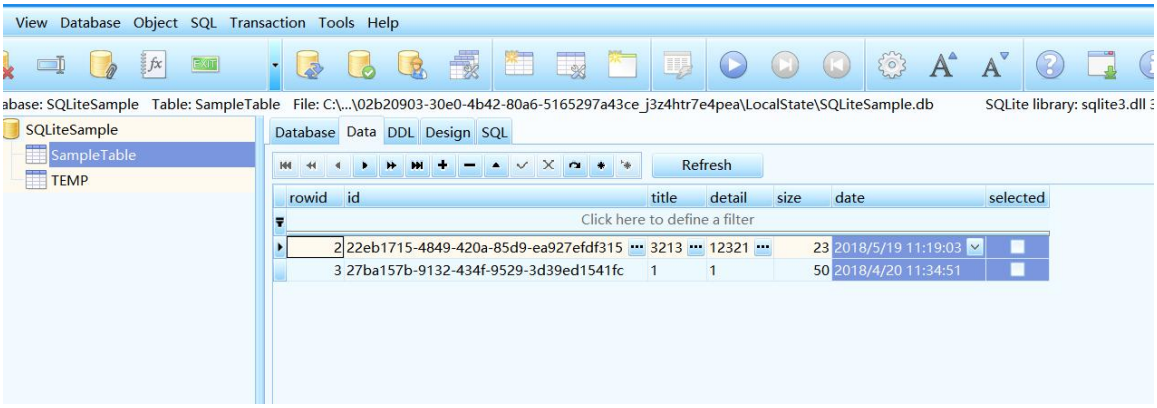
```
private void ShareButton_Click(object sender, RoutedEventArgs e)
{

    dynamic ori = e.OriginalSource;
    ViewModel.selectedItem = (Models.TodoItem)ori.DataContext;
    DataTransferManager dataTransferManager = DataTransferManager.GetForCurrentView();
    dataTransferManager.DataRequested += DataTransferManager_DataRequested;
    DataTransferManager.ShowShareUI();
}
async void DataTransferManager_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)
{
    DataRequest request = args.Request;
    var deferral = args.Request.GetDeferral();
    StorageFolder pictureFolder = await ApplicationData.Current.LocalFolder.GetFolderAsync("ProfilePictures");
    StorageFile pictureFile = await pictureFolder.GetFileAsync(ViewModel.selectedItem.id + ".jpg");

    request.Data.Properties.Title = ViewModel.selectedItem.title;
    string str = ViewModel.selectedItem.description;
    request.Data.SetText(str);
    request.Data.SetStorageItems(new List<StorageFile> { pictureFile });
    deferral.Complete();

}
```

第六周:

数据库

```
    public static DatabaseService GetInstance()...
    public static SQLiteConnection _connection;
    private static String DB_NAME = "SQLiteSample.db";
    private static String TABLE_NAME = "SampleTable";
    public DatabaseService()...
    public void CreateTable()...
    public void Insert(TodoItem todoItem)...
    public void Delete(string id)...
    public void Update(string id, string title, string detail, double size, DateTime date)...
    public string Query(string text)...
    public void Complete(string id, bool IsCompleted)...
    public void LoadData()
    {
        try
        {
            var sql = "SELECT id, title, detail,size, date, selected FROM "+TABLE_NAME;
            using (var statement = _connection.Prepare(sql))
            {
                while (SQLiteResult.ROW == statement.Step())
                {
                    string id = (string)statement[0];
                    string title = (string)statement[1];
                    string detail = (string)statement[2];
```

数据库内容可视化工具的使用



四、亮点与改进（可选）

1. 完成了每一次的 bonus：

图片存储挂起和恢复，可变磁贴背景图片，动态分享图片。由于据说数据库一般不推荐存图片，所以只选择了存图片的路径，也是经过对应的 id 作为图片名称访问的。

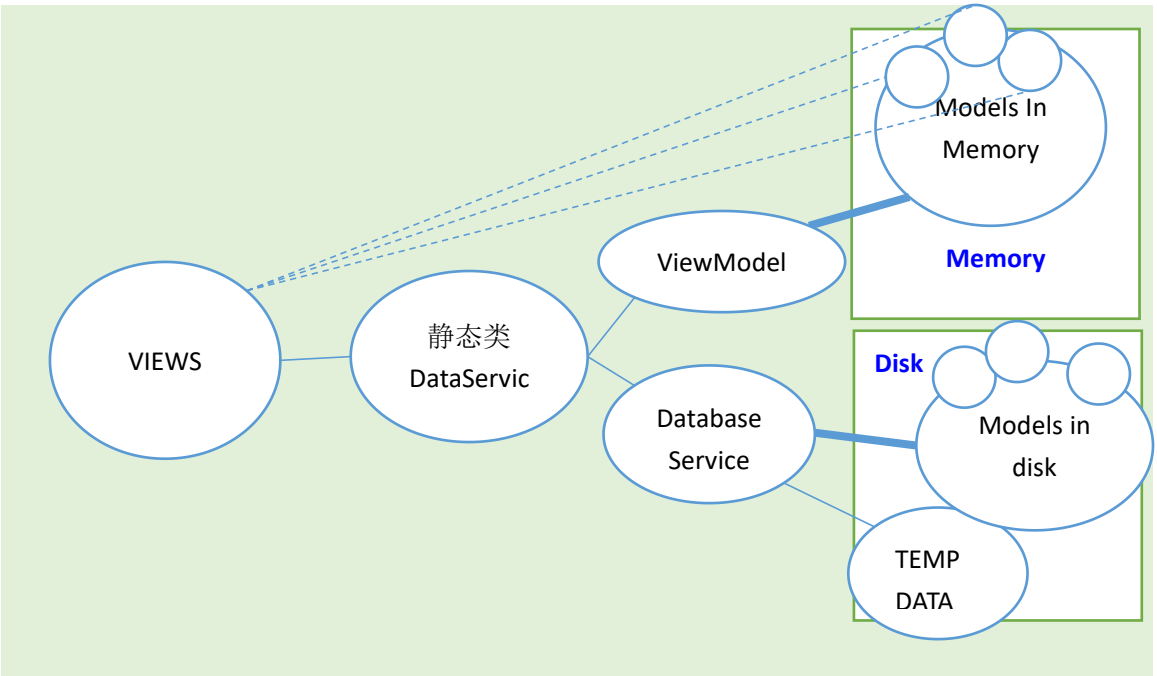
2. 重新设计了整个项目的结构：

加入数据库之后，程序运行时数据就存在了两份，一份是内存中的 Model,另一份在数据库中的元组。

对于内存中 models 主要通过 ViewModel 进行插入删除 update 等操作，对于数据库，即硬盘中的数据元组主要通过 DatabaseService 进行操作。因为两者都

有很多方法是同名且功能逻辑相似就通过加入一个 DataService 静态类对二者的方法合并。

Views 模块基本只需要通过 DataService 的静态方法就可以完成跟数据层的交互。各个模块之间的耦合性较低，后续的升级和修改也应该会比较容易。因为之前的各个部分逻辑分离做得还好所以这次大改虽然基本算是重写但花费时间也并不算太久。



五、遇到的问题

第四周：

这一周主要实现挂起和关闭，基本没有难度。参考的 ppt 中实现方式都有。我面临的难点有二：

1：关于挂起和恢复：我的双页面视图实现方法采用嵌入页面实现，故挂起和恢复不能直接参考 TA 的代码。网上也没有寻找到嵌入式页面的恢复方法。

学习了一下相关文档后。发现我应该做的是保存嵌入的 Frame 的路由状态。于是我尝试在主页面中保存和恢复子页面的路由状态，成功。

2：图片保存的 bonus 实现：文件存取还有访问权限没有学习，有一定难度。但这不是这个过程中的困难之处。困难主要在于，挂起留给程序的时间不足以异步完成图片的保存。存下来的图片只有 0 字节。但是异常确是显示没有找到。对 Debug 造成了很大困扰。

和同学交流之后，尝试使用图片即刻存储的方式，成功实现图片的挂起和恢复。

第五周：

3. 实现创建磁贴

由于对磁贴这种形式一头雾水，我开始几天查了许多文章，并做了一些方向错误的尝试。比如 BackgroundTask 的实现。一些尝试似乎比磁贴本身还要麻烦和困难。

在找到正确的方向后，我很快就实现了磁贴。bonus 中的添加动态图片背景，由于之前实现的 bonus 图片保存，也很容易。

第六周：

4. 项目重新架构

因为主要任务耗时不多，又感到添加数据库功能时的困惑，不知写在哪个部分，并且由于对 MVVM 模式的理解困难，所以这一次将结构重写。完成了感觉舒服一点的架构。示意图见“四、亮点与改进”的结尾。

六、思考与总结

对 UWP 和 c#有了一定的了解，有了一定查找文档和博客的经验。寻求问题答案时高效了很多。并且有了一定 Debug 经验，对 UWP 应用的运行模式有了初步了解

设计时应该注意各个部分的逻辑分离，方便未来的功能升级。

合理的实现方法也许可以为开发提供许多便利。

关于 LocalFolder RoamingFolder 的作用：

LocalFolder 是本地存储的位置，会被备份，重新安装时可以还原。

RoamingFolder 可以跨设备漫游同步，但有大小限制严重。

关于 StringBuilder 的作用：

在 C#中，string 类型虽然可以自由的使用+等操作改变内容，但其实质是重新申请一块内存再更改指向位置。因此如果频繁的更改 string 类型数据的值，会耗费大量的空间。直到启动内存回收机制，但会拖慢运行速度。所以产生了有容量的 stringBuilder 类型，可以方便地进行字符串拼接等操作。