数据存取

中山大学　郑贵锋

通过这一节课，可以使学生了解windows8应用里面文件的存取，SQLite数据库等数据存取的形式

大纲

1. 应用程序的数据存储
2. 用户库操作
3. 文件访问方式
4. SQLite数据库

应用程序的数据存储

- 从应用商店下载并安装应用程序到本地设备后，系统会为其分配独立的空间进行储存和管理。
- 我们可以通过相关的API访问并合理使用这些数据
- 应用程序卸载时，相关的数据也会被删除。
- 一般将用户信息和不可替代的数据保存在“用户库”或者是“SkyDrive”里面

本地数据存储

- 最常用

- 数据不会因为系统的挂起、退出或者关机等操作而丢失

- 如果应用被卸载，数据也被清空

漫游数据存储

. 需要用户设备处于联网的状态

. 及时获得同步数据更新

. 从云端（网络服务器存储）

临时数据存储

. 系统会根据资源的使用情况对临时数据存储空间中的数据进行清理

. 只适合存储本次用户使用的临时数据

通过使用ApplicationData类的LocalFolder、RoamingFolder、TemporaryFolder属性可以分别获取本地、漫游、临时存储空间的根文件夹。

创建、删除文件和文件夹。

写入和读取文件内容。

MainPage.xaml.cs

```
StorageFolder localFolder = ApplicationData.Current.LocalFolder;

StorageFolder folder = null;

StorageFile file = null;
```

“localFolder”用来保存本地应用存储空间的根文件夹。

“folder”用来保存新建的文件夹。

“file”用来保存新建的文件。

创建文件夹

```
async private void CreateFolder_Click(object sender, RoutedEventArgs
e)
        {
            folder = await localFolder.CreateFolderAsync("myFolder",
CreationCollisionOption.ReplaceExisting);
        }
```

删除文件夹

```
async private void DeleteFolder_Click(object sender, RoutedEventArgs
e)
        {
            if (folder != null)
            {
                await folder.DeleteAsync();
            }
        }
```

创建文件

```
async private void CreateFile_Click(object sender, RoutedEventArgs e)
        {
            file = await localFolder.CreateFileAsync("myFile",
CreationCollisionOption.ReplaceExisting);
        }
```

删除文件

```
async private void DeleteFile_Click(object sender, RoutedEventArgs e)
        {
            if (file != null)
            {
                await file.DeleteAsync();
            }
        }
```

写入文件

```
 async private void WriteFile_Click(object sender, RoutedEventArgs e)
        {
            file = await localFolder.CreateFileAsync("myFile",
CreationCollisionOption.ReplaceExisting);
            await FileIO.WriteTextAsync(file, "hello");
        }
```

读取文件

```
async private void ReadFile_Click(object sender, RoutedEventArgs e)
        {
            if (file != null)
            {
                String text = await FileIO.ReadTextAsync(file);
            }
        }
```

应用设置存储

本地存储、漫游存储

获取以及更改存储中的应用程序设置信息

应用设置容器获取

系统使用应用设置容器ApplicationDataContainer来保存应用的设置信息。

获取应用设置容器是获得应用设置信息的第一步。

```
Windows.Storage.ApplicationDataContainer localSettings =
Windows.Storage.ApplicationData.Current.LocalSetting;
```

应用设置容器操作

容器中的设置信息是以键值对的形式存储的。

包含简单键值对和复合键值对。

♦简单键值对

```
//创建名为“test”的键值
String name = “test”;

//将这个键赋值为“localSettingsTest!”
localSeetings.Values[name] = “localSettingsTest!”;

localSettings.Values.Remove(“test”);
```

应用设置容器操作

♦复合键值对

新建ApplicationDataCompositeValue类的对象；
在这个对象中存储复合键值对设置信息；
将对象作为一个整体保存到应用设置容器中。

```
ApplicationDataCompositeValue compsite = new ApplicationDataCompositeValue();
```

用户库操作

. 用户库分为图片库、音乐库、文档库和视频库。

. 通过使用相应的API可以向这些库里面添加新的文件、文件夹或者对文件进行重新组织。

KnownFolders类

DocumentsLibrary	获取文档库
MusicLibrary	获取音乐库
PicturesLibrary	获取图片库
VideosLibrary	获取视频库

StorageFolder类

CreateFileAsync(String)	在文件夹中新建文件
CreateFileAsync(String,CreationCollisionOption)	在当前文件夹中新建文件
GetFilesAsync()	从当前文件夹获取子文件的列表
GetFolderAsync()	从当前文件夹中获取一个子文件
GetFoldersAsync()	从当前文件夹中获取其子文件夹的列表

StorageFile类

OpenAsync()	打开一个指定的文件，并返回随机访问流
OpenReadAsync()	在当前文件中打开一个随机访问流，以读取文件中的内容
DeleteAsync()	删除当前文件

FileIO类

WriteTextAsync(IStorageFile,string)	向文件中写入文本数据
WriteTextAsync(IStorageFile,String,UnicodeEncoding)	向文件中写入文本数据,并指定其字符编码
ReadTextAsync(IStorageFile)	读取文件的文本数据
ReadTextAsync(IStorageFile，UnicodeEncoding)	读取文件内容，并指定其字符编码
ReadBufferAsync()	读取指定文件内容，并返回一个缓冲区
WriteBufferAsync()	将缓冲区的数据写到指定文件中

获取用户库位置

- 获取文档库位置

- `StorageFolder documentFolder = KnownFolders.DocumentsLibrary;`

文件读写

- 写入文本
- `await Windows.Storage.FileIO.WriteTextAsync(file, “内容字符串”);`

读取文本

- `string text = await Windows.Storage.FileIO.ReadTextAsync(file);`

创建子文件夹

```
StorageFolder folder = await folders.CreateFolderAsync( “hello” );
```

删除子文件夹

```
await subFolder.DeleteAsync();
```

文件访问方式

. 用户可以访问应用程序内部文件，还可以通过文件选取器对其他文件进行选取

. 使用文件URL直接进行检索

文件选取器

. 是应用与系统进行交互的一个接口

. 文件打开选取器：FileOpenPicker

. 文件保存选取器：FileSavePicker

文件打开选取器属性

ViewMode	展示文件或文件夹的视图模式，List:列表模式；Thumbnail:缩略图模式
SuggestedStartLocation	显示的初始位置
FileTypeFilter	显示的文件类型集合

创建文件打开器

```
//创建文件选取器
    FileOpenPicker openPicker = new FileOpenPicker();
//设置ViewMode属性
   openPicker.ViewMode = PickerViewMode.Thumbnail;

//将开始位置设置在图片库
   openPicker.SuggestedStartLocation =
PickerLocationId.PicturesLibrary;

//设置可以选取的文件格式
   openPicker.FileTypeFilter.Add( “.jpg” );
   openPicker.FileTypeFilter.Add( “.png” );
```

SQLite数据库

插入数据

```
//获得数据库连接
    SQLiteConnection db = CreateSQLiteConnection();

//插入数据
   db的Insert方法；
//关闭连接
    db.Close();
```

删除数据

```
//获得数据库连接
    SQLiteConnection db = CreateSQLiteConnection();

//插入数据
   db的Delete方法

//关闭连接
    db.Close();
```

修改数据

```
//获得数据库连接
    SQLiteConnection db = CreateSQLiteConnection();

//插入数据
    db的Update方法；

//关闭连接
    db.Close();
```

查询数据

```
//获得数据库连接
    SQLiteConnection db = CreateSQLiteConnection();

//插入数据
   对象的Where的方法；

//关闭连接
    db.Close();
```

总结

- 应用程序存储方式
- 用户库API
- 文件选择器
- SQLite数据库使用

作业

实现个人头像的存取功能：

- 头像的存取
- 读取的图片显示在界面上

Optional：

- 合理的布局（参考第三周的作业）
- 其他类型数据的存取，如文本等
- 使用SQLITE进行数据存取。

Q&A