苏州大学文正学院

毕业设计（论文）开题报告

题　　目　个人照片管理系统设计与实现

系　　科　计算机工程系

专　　业　计算机科学与技术

学生姓名　徐福扬　**学号**　1717249062

指导教师　姚望舒　**职称**　副教授

2019 年　5 月 1　日

1、引言

Windows Presentation Foundation（WPF）是美国微软公司推出.NET Framework 3.0 及以后版本的组成部分之一，它是一套基于 XML、.NET Framework、向量绘图技术的展示层开发框架，微软视其为下一代用户界面技术，广泛被用于 Windows Vista 的界面开发。WPF 使用一种新的 XAML（eXtensible Application Markup Language）语言来开发界面，这将把界面开发以及后台逻辑很好的分开，降低了耦合度，使用户界面设计师与程序开发者能更好的合作，降低维护和更新的成本。

2、概述

本系统是基于.net 平台下的 C#语言编写的一款照片管理系统，除了基本的 C#语法之外还使用了 wpf 技术，解决了传统的 winform 界面的界面不美观。借助了第三工具读取照片中的地理信息。开发工具为 visual studio 2017,.net framework 版本为 4.7.2，该版本号实现了.net .NETStandard2.0 规范。

3、关键技术

3.1 C#技术

C# 语法高度重视表达，但学习起来也很简单轻松。 任何熟悉 C、C++ 或 Java 的人都可以立即认出 C# 的大括号语法。 通常情况下，了解上述任何一种语言的开发者可以在很短的时间内就开始使用 C# 高效工作。 C# 语法简化了 C++ 的许多复杂操作，并提供强大功能，如可以为 null 的值类型、枚举、委托、lambda 表达式和直接内存访问，而 Java 并不提供这些功能。 C# 不仅支持泛型方法和类型，提升了类型安全性和性能，还支持迭代器，以便集合类的实现者可以定义方便客户端代码使用的自定义迭代行为。 语言集成查询 (LINQ) 表达式让强类型查询成为最高级的语言构造。

作为面向对象的语言，C# 支持封装、继承和多形性这些概念。 所有变量和方法（包括作为应用程序入口点的 Main 方法）都封装在类定义中。 虽然类可能会直接继承一个父类，但可以实现任意数量的接口。若要用方法重写父类中的虚方法，必须使用 override 关键字，以免发生意外重定义。 在 C# 中，结构就像是轻量级类，是可以实现接口但不支持继承的堆栈分配类型。

3.2 WPF

wpf(windows presentation foundation)是用于windows的现代图形显示系统。与win32，

mfc 相比，wpf 使用的是 c#，而不是 C/C++。另外主要引入 了“内置硬件加速”，“更高级的 api”和“分辨率无关”等创新功能。在 wpf 问世之前的 15 年里，windows 平台的开发人员一直使用着本质相同的显示技术，主要是 windows 应用程序都依靠 User32 和 GDI/GDI+来创建用户界面。尽管从 win32 到 mfc 到.net，与底层这两块交互的 api 变得越来越简单，更加高效，但这些底层系统组件当初在设计时的限制却是一直无法突破的。

直到后来微软推出了 DirectX，经过数年的发展 DirectX 已经越来越强大，但 DirectX 具有很高的复杂性，导致少有商业软件使用 DirectX 开发，似乎 DirectX 已经成了游戏开发的专有。

而 Wpf 的出现彻底改变了 windows 平台应用开发的这种局面。wpf 底层的显示技术不是 GDI/GDI+，而直接是 DirectX。这样不管是复杂的三维图形还是简单的几个文本，wpf 都是通过 DirectX 管线完成绘图。即使在普通的桌面软件上也可以带来炫酷的效果。

利用 DPI(dot per inch)的概念，wpf 根据系统 dpi 进行缩放，并不根据物理显示系统的 dpi 进行缩放，灵活得放大/缩小显示内容，以使其适合所用的显示器和显示选择，做到分辨率无关性。

[物理单位尺寸] = [设备无关单位尺寸] x [系统 DPI]

3.2.1 数据绑定

Windows Presentation Foundation (WPF) 数据绑定为应用程序呈现数据并与数据交互提供了一种简单且一致的方式。 元素能够以 公共语言运行时 (CLR) 对象和 XML 形式绑定到来自各种数据源的数据。 ContentControl 如 sButton 并 ItemsControl 如 sListBox 和 ListView 具有内置功能，使灵活的样式设置单个数据项的集合。 可基于数据生成排序、筛选和分组视图。

WPF 中的数据绑定功能与传统模型相比具有几个优点，包括本质上支持数据绑定的大量属性、灵活的 UI 数据 UI 表示形式以及业务逻辑与 UI 的完全分离。

数据绑定是在应用程序 UI 和业务逻辑之间建立连接的过程。 如果绑定具有正确的设置，并且数据提供适当的通知，则在数据更改其值时，绑定到该数据的元素会自动反映更改。数据绑定还意味着，如果元素中数据的外部表示形式发生更改，则基础数据可以自动进行更新以反映更改。 例如，如果用户编辑中的值 TextBox 元素，则基础数据值自动更新以反映所做的更改。

数据绑定的典型用法是将服务器或本地配置数据放置到窗体或其他 UI 控件中。 在 WPF 中，此概念得到扩展，包括将大量属性绑定到各种数据源。 在 WPF 中，元素的依赖属性可以绑定到 CLR 对象（包括 ADO.NET 对象或与 Web 服务和 Web 属性关联的对象）和 XML 数据。

3.2.2 数据绑定基本概念

在建立绑定时，需要将绑定目标绑定到绑定源。 例如，如果您要显示某些基础 XML 中的数据 ListBox 使用数据绑定时，要绑定您 ListBox 到 XML 数据。

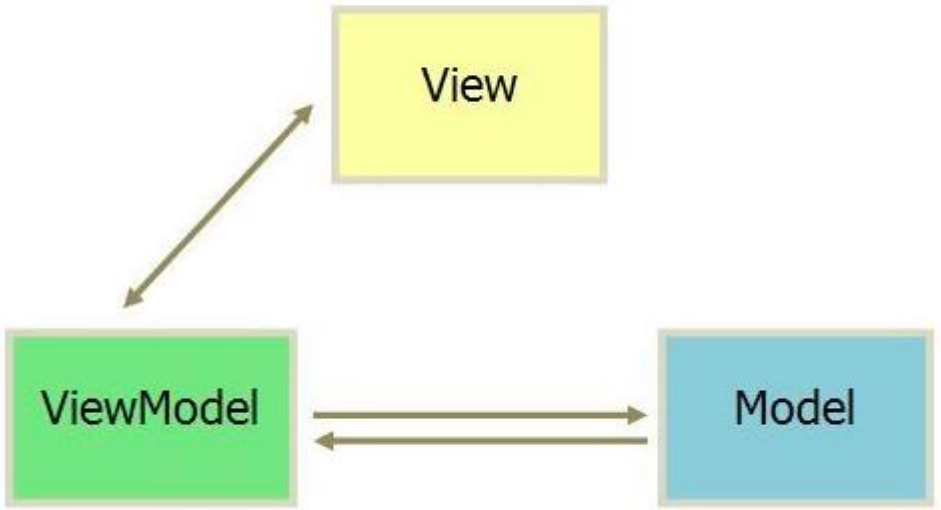
若要建立绑定，请使用 Binding 对象。 本主题的其余部分将讨论的许多相关的概念和一些属性和使用情况的 Binding 对象。

数据流的方向

正如前面提到的以及在上图中箭头所示，绑定的数据流可以进入从绑定目标绑定源 (例

如，源值发生更改时用户编辑的值 TextBox) 和/或从绑定源到绑定目标 (例如，你 TextBox 内容更新的是绑定源中的更改) 如果绑定源提供适当的通知。

可能希望应用程序允许用户更改数据，然后将该数据传播回源对象。 或者，可能不希望允许用户更新源数据。 您可以通过设置控制这 Mode 属性在 Binding 对象。 下图演示了不同类型的数据流：



- OneWay 绑定导致更改源属性自动更新目标属性，但对目标属性的更改不会传播回源属性。 如果绑定的控件为隐式只读，则此类型的绑定适用。 例如，可能绑定到如股票行情自动收录器这样的源，或许目标属性没有用于进行更改的控件接口（如表的数据绑定背景色）。 如果无需监视目标属性的更改，则使用 OneWay 绑定模式可避免 TwoWay 绑定模式的系统开销。
- TwoWay 绑定导致更改源属性或自动更新另的目标属性。 此类型的绑定适用于可编辑窗体或其他完整交互式的 UI 方案。 大多数属性默认为 OneWay 绑定，但某些依赖关系属性 (通常是用户可编辑控件的属性 Text 属性 TextBox 并 IsChecked 属性 CheckBox) 默认为 TwoWay 绑定。 确定依赖属性绑定在默认情况下是单向还是双向的编程方法是：使用 GetMetadata 获取属性的属性元数据，然后检查 BindsTwoWayByDefault 属性的布尔值。
- OneWayToSource 相反 OneWay 绑定；它的源属性更改时，更新目标属性。一个示例方案是只需要从 UI 重新计算源值的情况。
- 不在图中所示是 OneTime 绑定，这将导致源属性传入以初始化目标属性，但不是传播后续更改。 这意味着，如果数据上下文发生了更改，或者数据上下文中的对象发生了更改，该更改不会反映在目标属性中。 如果你在适合使用当前状态的快照或数据实际为静态数据的位置使用数据，则此类型的绑定适合。 如果你想使用源属性中的某个值来初始化目标属性，且提前不知道数据上下文，则此类型的绑定也有用。 这是实质上是 OneWay 绑定的一种简化形式，它在源值不更改的情况下提供更好的性能。

3.3、.net framefowk

.NET Framework 是 Windows 的托管执行环境，可为其运行的应用提供各种服务。 它包括两个主要组件: 公共语言运行时 (CLR), 它是处理运行应用的执行引擎; .NET Framework 类库，它提供开发人员可从其自己的应用中调用的已测试、可重用代码库。 .NET Framework 提供的用于运行应用的服务包括:

- 内存管理。 在许多编程语言中，程序员负责分配和释放内存并处理对象生存期。 在 .NET Framework 应用中，CLR 代表应用提供这些服务。
- 常规类型系统。 在传统编程语言中，基本类型由编译器定义，这将使跨语言互操作性复杂化。 在 .NET Framework 中，基本类型由 .NET Framework 类型系统定义，并且是面向 .NET Framework 的所有语言所共有的。
- 一个全面的类库。处理常见的低级编程操作时, 程序员可通过 .NET Framework 类库使用类型及其成员的易访问库，而不必编写大量代码。
- 开发框架和技术。.NET Framework 包括用于特定区域应用开发的库, 如用于 Web 应用的 ASP.NET，用于数据访问的 ADO.NET，用于面向服务的应用的 Windows Communication Foundation 以及用于 Windows 桌面应用的 Windows Presentation Foundation。
- 语言互操作性。 面向 .NET Framework 的语言编译器发出名为公共中间语言 (CIL) 的中间代码，反过来，通过公共语言运行时在运行时进行编译。 借助此功能，使用某种语言编写的例程可由另一种语言访问，程序员可以专注于使用其首选语言创建应用。
- 版本兼容性。 除少数例外，使用特定版本的 .NET Framework 开发的应用无需在更高版本中修改即可运行。
- 并行执行。 通过允许同一台计算机上存在公共语言运行时的多个版本，.NET Framework 可帮助解决版本冲突。 这意味着应用的多个版本可以共存，并且应用可在构建它的 .NET Framework 版本上运行。 并行执行适用于 .NET Framework 版本组 1.0/1.1, 2.0/3.0/3.5, and 4/4.5.x/4.6.x/4.7.x。
- 多定向。 通过面向 .NET Standard，开发人员可创建适用于该标准版本支持的多种 .NET Framework 平台的类库。 例如，面向 .NET Framework 4.6.1、NET Core 2.0 和 UWP 10.0.16299 的应用可以使用面向 .NET Standard 2.0 的库。

.NET Framework 包括公共语言运行时 (CLR) 和 .NET Framework 类库。 公共语言运行时是 .NET Framework 的基础。 可将运行时看作一个在执行时管理代码的代理，它提供内存管理、线程管理和远程处理等核心服务，并且还强制实施严格的类型安全以及可提高安全性和可靠性的其他形式的代码准确性。 事实上，代码管理的概念是运行时的基本原则。 以运行时为目标的代码称为托管代码，而不以运行时为目标的代码称为非托管代码。 类库是一个综合性的面向对象的可重用类型集合，可使用它来开发多种应用，这些应用程序包括传统的命令行或图形用户界面 (GUI) 应用，还包括基于 ASP.NET 提供的最新创新的应用（如 Web 窗体和 XML Web Services）。

.NET Framework 可由非托管组件承载，这些组件将公共语言运行时加载到它们的进程中并启动托管代码的执行，从而创建一个同时利用托管和非托管功能的软件环境。 .NET Framework 不但提供若干个运行时主机，而且还支持第三方运行时主机的开发。

例如，ASP.NET 承载运行时以为托管代码提供可伸缩的服务器端环境。 ASP.NET 直接使用运行时以启用 ASP.NET 应用和 XML Web Services（本主题稍后将对这两者进行讨论）。 Internet Explorer 是承载运行时（以 MIME 类型扩展的形式）的非托管应用的一个示例。 使

用 Internet Explorer 承载运行时使您能够在 HTML 文档中嵌入托管组件或 Windows 窗体控件。 以这种方式承载运行时可使托管移动代码成为可能，不过它需要进行只有托管代码才能提供的重大改进（如不完全受信任的执行和独立的文件存储）。

下面的插图显示公共语言运行时和类库与应用之间以及与整个系统之间的关系。 该插图还显示托管代码如何在更大的结构内运行。

3.3.1 公共语言运行时的功能

公共语言运行时管理内存、线程执行、代码执行、代码安全验证、编译以及其他系统服务。 这些功能是在公共语言运行时上运行的托管代码所固有的。

至于安全性，取决于包括托管组件的来源（如 Internet、企业网络或本地计算机）在内的一些因素，托管组件被赋予不同程度的信任。 这意味着即使用在同一活动应用中，托管组件既可能能够执行文件访问操作、注册表访问操作或其他须小心使用的功能，也可能不能够执行这些功能。

运行时还通过实现称为常规类型系统 (CTS) 的严格类型验证和代码验证基础结构来加强代码可靠性。 CTS 确保所有托管代码都是可以自我描述的。 各种 Microsoft 编译器和第三方语言编译器都可生成符合 CTS 的托管代码。 这意味着托管代码可在严格实施类型保真和类型安全的同时使用其他托管类型和实例。

此外，运行时的托管环境还消除了许多常见的软件问题。 例如，运行时自动处理对象布局并管理对对象的引用，在不再使用它们时将它们释放。 这种自动内存管理解决了两个最常见的应用错误：内存泄漏和无效内存引用。

运行时还提高了开发人员的工作效率。例如，程序员用他们选择的开发语言编写应用，却仍能充分利用其他开发人员用其他语言编写的运行时、类库和组件。 任何选择以运行时为目标的编译器供应商都可以这样做。 以 .NET Framework 为目标的语言编译器使得用该语言编写的现有代码可以使用 .NET Framework 的功能，这大大减轻了现有应用的迁移过程的工作负担。

尽管运行时是为未来的软件设计的，但是它也支持现在和以前的软件。 托管和非托管代码之间的互操作性使开发人员能够继续使用所需的 COM 组件和 DLL。

运行时旨在增强性能。 尽管公共语言运行时提供许多标准运行时服务，但是它从不解释托管代码。 一种称为实时 (JIT) 编译的功能使所有托管代码能够以它在其上执行的系统的本机语言运行。 同时，内存管理器排除了出现零碎内存的可能性，并增大了内存引用区域以进一步提高性能。

最后，运行时可由高性能的服务器端应用（如 Microsoft SQL Server 和 Internet Information Services (IIS)）承载。 此基础结构使您在享受支持运行时承载的行业最佳企业服务器的优越性能的同时，能够使用托管代码编写业务逻辑。

3.2.2 .NET Framework Class Library — .NET Framework 类库

.NET Framework 类库是一个与公共语言运行时紧密集成的可重用的类型集合。该类库是面向对象的，并提供某些类型，可供你自己的托管代码从中派生功能。 这不但使 .NET Framework 类型易于使用，而且还减少了学习 .NET Framework 的新功能所需要的时间。

此外，第三方组件与 .NET Framework 中的类无缝集成。

例如，.NET Framework 集合类实现一组用于开发自己的集合类的接口。 你的集合类与 .NET Framework 中的类无缝地混合。

正如您对面向对象的类库所希望的那样，.NET Framework 类型使您能够完成一系列常见编程任务（包括诸如字符串管理、数据收集、数据库连接以及文件访问等任务）。 除这些常规任务之外，类库还包括支持多种专用开发方案的类型。

4、总结

当今技术基于 winform 的架构越来越多越来越成熟，从之前的 win32 无 ui 设计界面到 mfc 基于 c++的设计架构，再到如今的 wpf。Wpf 简洁而高效。更加容易编写出漂亮的界面。本系统采用的是 wpf 架构。

5、参考文献

[1] 微软文档官网 https://docs.microsoft.com/zh-cn/
[2] Jeffrey Richter， via C#第四版，清华大学出版社，2014
[3] 刘铁猛，深入浅出 WPF，中国水利水电出版社，2010
[4] (美)内格尔 (Nagel,C.) 等著，C#高级编程(第 9 版)，清华大学出版社，2014
[5] 博客园，https://www.cnblogs.com/