Электронный научный журнал "Математическое моделирование, компьютерный и натурный эксперимент в естественных науках" http://mathmod.esrae.ru/
URL статьи: mathmod.esrae.ru/16-58
Ссылка для цитирования этой статьи:
Успенский К.Е. Создание сервисно-ориентированного приложения Windows // Математическое моделирование, компьютерный и натурный эксперимент в естественных науках. 2017. №4

УДК 519.683.8

СОЗДАНИЕ СЕРВИСНО-ОРИЕНТИРОВАННОГО ПРИЛОЖЕНИЯ WINDOWS

Успенский К.Е.[1]
[1]Поволжский институт управления имени П.А. Столыпина (филиал РАНХиГС), Россия, Саратов, uspenskiyke@mail.ru

THE CREATION OF SERVICE-ORIENTED APPLICATION FOR WINDOWS

Uspenskiy K.E.[1]
[1]Volga Region Institute of Management, named by P.A. Stolypin (a Branch of Russian Academy of National Economy and Public Administration), Russia, Saratov, uspenskiyke@mail.ru

Аннотация. Рассматриваются некоторые возможности платформы разработки универсальных приложений Windows на примере создания сервисно-ориентированного приложения на языке программирования C#

Ключевые слова: универсальные приложения Windows, программирование, сервис, .NET Framework, JSON.

Abstract. Some features of the universal Windows platform are considered on the example of creating a service-oriented application using the C# programming language

Keywords: universal Windows applications, programming, service, .NET Framework, JSON.

Операционная система Windows [1] имеет многолетнюю историю. Со времени своего появления в 1985 году, она претерпела множество изменений. Различные преобразования коснулись как пользовательского интерфейса и самой архитектуры системы, так и инструментов разработки приложений. На заре своего существования данная операционная система использовала технологию компонентной модели объектов (COM) [2]. Приложения согласно этой модели представляются как совокупность взаимодействующих объектов, каждый из которых может обновляться независимо от остальных. На основе этой технологии позже был разработан стандарт внедрения и связывания объектов (OLE) [3], с помощью которого были созданы графические элементы управления. Дальнейшим развитием технологии COM было появление

технологий DCOM [4] и COM+ [5]. Основным преимуществом первой из них была возможность работы на основе сетевого взаимодействия, а второй – работа с пулом потоков, контекстами компонентов и транзакциями. Система Windows имеет собственные функции уровня ядра для доступа к компонентам COM – Windows API [6]. Затем, в 2002 году была создана платформа .NET Framework [7]. Это было огромным шагом вперед. Чтобы убедиться в этом, достаточно обратить внимание на её структуру. Она содержит общеязыковую среду выполнения (CLR), общую систему типов (CTS), средства компиляции по запросу (JIT), средства «сборки мусора» и систему пространств имён. Всё это позволяет, во-первых, чётко организовать структуру классов библиотеки .NET, во-вторых, использовать любой из многочисленных языков программирования, официально поддерживаемых в .NET. Кроме того, появилась возможность использовать встроенные в .NET механизмы обеспечения безопасности. С другой стороны, было предложено усовершенствовать низкоуровневую подсистему Windows – Windows API. Дело в том, что практически все возможности этой операционной системы реализованы в виде объектов. Почему бы не использовать тот же подход и в реализации функций Windows API? Таким образом, эти функции стали объектами COM и получили название Windows Runtime [8]. Возможности Windows стали расширяемыми на уровне Windows Runtime. Эта технология поддерживается в операционной системе, начиная с версии Windows 8. Далее наступает эпоха «Интернета вещей» (IoT) [9], когда одной из главных идей при разработке любого электронного устройства станет возможность обмена данными с соседними устройствами. Это означает, что как приложения, так и сама операционная система должны иметь возможность работать на разных устройствах. Платформа универсальных приложений Windows (UWP), появившаяся в Windows 10, как раз и решает эту задачу. Один и тот же код, написанный разработчиком, сможет работать на десктопном компьютере, планшете или смартфоне. Сохраняется возможность взаимодействия и с другими устройствами, если их разработчики предоставят соответствующие средства для разработки (SDK). Эта работа посвящена краткому рассмотрению некоторых особенностей разработки приложений с использованием UWP.

В качестве примера рассмотрим приложение, получающее информацию от онлайн-сервисов. Для разработки будем использовать язык программирования C#, поскольку он создан специально для .NET Framework и имеет хорошо знакомый по языку C синтаксис основных алгоритмических конструкций. Итак, приложение должно принять данные от пользователя, передать их сервису, получить ответ, осуществить его разбор и отобразить требуемые данные с помощью элементов пользовательского интерфейса. Под сервисом будем понимать web-приложение стороннего разработчика, которое может предоставлять свои функции удаленно другим разработчикам для использования в их программных продуктах. Поскольку в этом случае решение прикладной задачи достигается при помощи приложения и сервиса, то

необходимо предварительно принять условия лицензионного соглашения, предложенного владельцем сервиса. Также возможно взимание платы с разработчика за использование сервиса. В качестве целевой платформы разработки выберем сборку операционной системы Windows 10 Redstone 3. Поскольку при разработке используется UWP, то пользовательский интерфейс будет реализован на основе Windows Presentation Foundation (WPF) [10]. Эта технология была реализована с целью возможности получения прямого доступа к графической подсистеме DirectX [11]. Кроме того, в связи с высокой популярностью веб-разработки, модель WPF предусматривает использование web-компоновки при размещении элементов управления. Работая с ней, разработчик имеет дело с двумя языками – одним из многочисленных .NET языков и специализированным декларативным языком разметки пользовательского интерфейса – XAML. Последний основан на технологии XML [12], и поэтому сильно напоминает HTML по своей структуре. С его помощью могут быть описаны различные графические эффекты, в том числе трёхмерные. В условиях промышленной разработки это позволяет создавать пользовательский интерфейс независимо от логики приложения.

Необходимо выбрать, в каком формате приложение и сервис будут обмениваться данными. Наиболее часто применяются два формата: XML и JSON [13]. По-видимому, предпочтительнее использовать второй в связи с его более удобным восприятием (хотя XML поддерживается гораздо большим количеством технологий разработки). Далее выбираем сервис, с которым будет работать приложение. Различных сервисов в Интернете достаточно много. Но естественно выбрать один из широко известных. Поэтому было решено использовать сервис Google Maps Directions. Перед началом использования сервиса необходимо зарегистрироваться и получить токен (уникальный ключ) на его официальном сайте. Затем, ознакомившись со структурой запроса и ответа в формате JSON, приступаем к написанию кода. При разработке использовалась интегрированная среда Visual Studio 2017 Community Edition.

Так как функционал у приложения небольшой, оно имеет однооконный интерфейс. Его основной задачей является получение от сервиса информации о расстоянии между двумя точками на карте, которые задаются названиями населенных пунктов или адресами в их пределах. Также приложение показывает время пути на автотранспорте между заданными точками. Информация отображается в текстовом виде. В классе приложения имеется единственный метод, в котором задействована технология асинхронного программирования. Такая необходимость продиктована тем, что приложение ожидает получения данных из сети и всё это время должно быть доступным пользователю. Для осуществления разбора JSON файла с использованием консоли NuGet был установлен и подключен к проекту бесплатный пакет Newtonsoft.json [14].

Как показало тестирование, приложение успешно справилось со своей основной задачей. Время получения данных от сервиса не превышало минуты.

Ясно, что быстродействие данного приложения, в основном, обусловлено скоростью передачи данных по сети и степенью загруженности серверов, на которых работает сервис. Но такая ситуация имеет место лишь до тех пор, пока приложение не возьмёт на себя значительную нагрузку по обработке большого количества полученных данных. Для этого случая платформа UWP также предоставляет решение. Это компиляция в машинный код. Что касается отображения приложения на экранах мобильных устройств, то эту возможность обеспечивает адаптивность пользовательского интерфейса. Разработчику не нужно об этом беспокоиться. Отрисовка интерфейса в приложениях UWP происходит с учётом возможностей устройства.

В данной работе были коротко рассмотрены основные шаги реализации приложения, взаимодействующего с сервисом. Разработка сервисно-ориентированных приложений сегодня является весьма популярной. А учитывая тот факт, что уже сегодня существует кроссплатформенная сокращенная версия .NET, можно не сомневаться, что в недалеком будущем технологии разработки приложений для Windows станут доступны для большинства операционных систем.

Литература

1. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows. СПб: Питер 2013. 800 с.
2. Swanke John E. COM Programming by Example: Using MFC, ActiveX, ATL, ADO, and COM+. CMP Books 2000. 358 с.
3. Microsoft Application Programming Interfaces: Directx, Activex Data Objects, Windows Api, Object Linking and Embedding, Directdraw 2010. 602 с.
4. Thuan Thai. Learning DCOM. O'Reilly Media. 2011. 504 с.
5. Роберт Дж. Оберг. Технология COM+. Основы и программирование. М: Вильямс. 2000. 480с.
6. Щупак Ю. Win32 API. Разработка приложений для Windows. СПб: Питер. 592 с.
7. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. СПб: Питер. 2017. 896 с.
8. Рихтер Дж, Мартен ван де Боспурт. WinRT. Программирование на C# для профессионалов. М: Вильямс. 2014 368 с.
9. Грингард С. Интернет вещей. Будущее уже здесь. М: Альпина-Паблишер. 2017. 188 с.
10. Макдональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов. М: Вильямс. 2013. 1024 с.
11. Попов А.А. DirectX 10 – это просто. Программируем графику на C++. СПб: БХВ-Петербург. 2008. 464 с.
12. Хантер Д и др. XML. Базовый курс. М: Вильямс. 2018. 1344 с.
13. Smith B. Beginning JSON. Apress. 2015. 353 с.

14. Документация на официальном сайте Newtonsoft. URL: https://www.newtonsoft.com/json/help/html/Introduction.htm