

**Лабораторный ­практикум**

Курс:Службы приложений

Сентябрь 2015 года

Содержание

**No table of contents entries found.**

Обзор

Службы приложений – это службы без головы, которые могут инкапсулировать бизнес-логику, данные и транзакции относительно других приложений без потери производительности при запуске. Они обеспечивают функциональность, сходную с Веб-службами, но в контексте Магазина приложений. Некоторые из сценариев, включающие использование Служб приложений, включают извлечение и кэширование данных.

В случае извлечения данных вы можете сделать так, чтобы ваше приложение могло иметь доступ или обрабатывать данные, полученные из другого приложения без запуска самого приложения. Службы приложений, использующие фоновые задачи, являются эффективным способом получения доступа к таким данным.

Службы определения местоположения являются великолепным примером извлечения кэшированных данных с целью сохранения ресурсов. Вам может понадобиться ваше приложения для получения доступа к локальным данным карт при минимальном использовании данных. Если другое приложение, например, карты Bing, уже загрузило и кэшировало информацию, необходимую вашему приложению, вы можете использовать Службу приложения для извлечения такого рода информации без использования дополнительной пропускной способности.

В рамках данного курса вы создадите Службу для приложения по поиску сотрудников, которое считывает одни или несколько ID сотрудников в качестве входных данных и выдает соответствующие ФИО сотрудников. Затем вы научитесь создавать приложение для запроса службы и отображения результатов.

# Цели

* 1. Настоящий курс научит вас:
  + Создавать и регистрировать Службу приложения
  + Вызывать Службу приложения в фоне из другого приложения
  + Передавать данные обратно в вызывающее приложение
  + Отлаживать Службу приложения

# Системные требования

* 1. Для заверешения настоящего курса необходимы:
  + Microsoft Windows 10
  + Microsoft Visual Studio 2015

# Настройка

* 1. Вам следует выполнить следующие действия для подготовки компьютера:
  2. Установить Microsoft Windows 10.
  3. Установить Microsoft Visual Studio 2015.

# Упражнения

* 1. Настоящий Лабораторный практикум включает следующие упражнения:
  2. Создание и регистрация Службы приложения
  3. Вызов Службы приложения из другого приложения
  4. Расчетное время для завершения курса: **45-60 минут**.

Упражнение 1: Создание и регистрация Службы приложения

* 1. В рамках данного упражнения вы создадите решение для Службы приложения по поиску сотрудников. Решение будет содержать компонент Windows Runtime Component со Службой приложения, а также приложением, разработанным на основе шаблона Пустого приложения, с помощью которого вы сможете зарегистрировать Службу приложения. Мы не создадим Пользовательский интерфейс для самого приложения, но мы будем использовать наименование соответствующего пакета для семейства программных систем позже для того, чтобы другие приложения могли взаимодействовать со Службой приложения.

Задача 1 – Создать проект EmployeeLookupService

Создайте новое решение для проекта службы приложения.

1. В новой версии Visual Studio 2015 выберите **File (Файл) -> New (Новый) -> Project (Проект)**, чтобы открыть диалоговое окно New Project (Новый проект). Далее **Installed (Установленное) > Templates (Шаблоны) > Visual C# > Windows > Universal**, а затем выберите шаблон **Blank App приложения (Universal Windows)**.
2. Назовите свой проект **"EmployeeLookupService"**. Сохраните проект в папку, в которой вы храните свои работы по настоящему курсу.
3. Настройте Solution Configuration (Текущую конфигурацию решения) на **Debug (Отладку)** и Solution Platform (Платформу решений) в соответствии с **x86**. Выберите **Local Machine (Локальный компьютер)** из выпадающего меню Debug Target (Цели отладки).
4. Создайте и запустите свое приложение. Вы увидите окно Blank App со счетчиком частоты кадров, активированном по умолчанию для отладки.

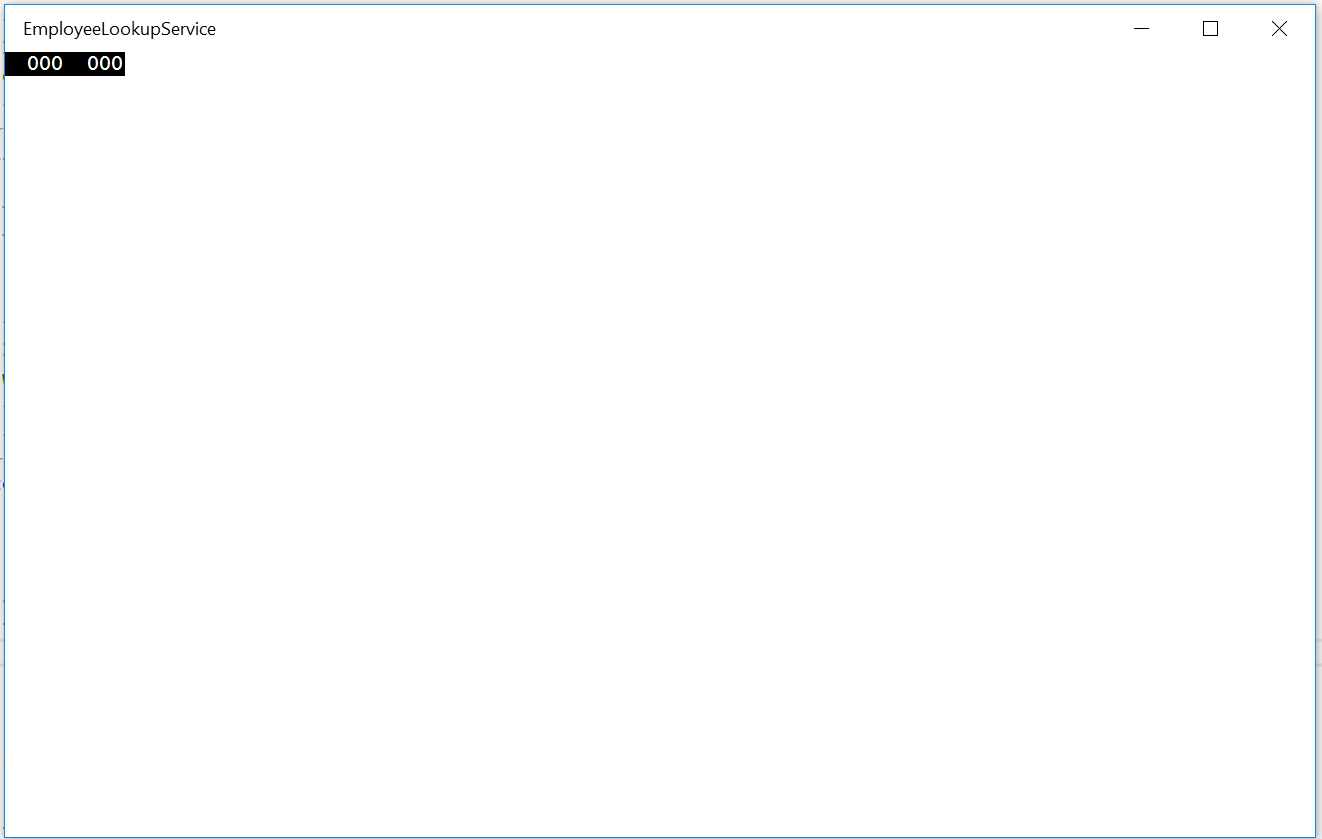


Рисунок 1

Приложение Blank universal выполняется в режиме Рабочего стола.

* 1. **Примечание:** В рамках данного курса мы будем использовать приложение EmployeeLookupService как контейнер для Службы приложения, выполняемого в качестве фоновой задачи. Отображение MainPage (Главной страницы) будет оставаться пустым.

1. Вернитесь к Visual Studio и отключите отладку.

Задача 2 – Создать компонент Windows Runtime Component

В рамках проекта служба приложения будет функционировать в качестве фоновой задачи с компонентом Windows Runtime Component. Мы начнем с создания компонента Windows Runtime Component.

1. Щелкните правой кнопкой мыши на решение EmployeeLookupService и выберите **Add (Добавить) > New Project (Новый проект)**. Добавьте проект, выполнив ряд действий: **Visual C# > Windows > Universal > Windows Runtime Component (Universal Windows).**. Назовите его **"EmployeeLookupService.Background"**.

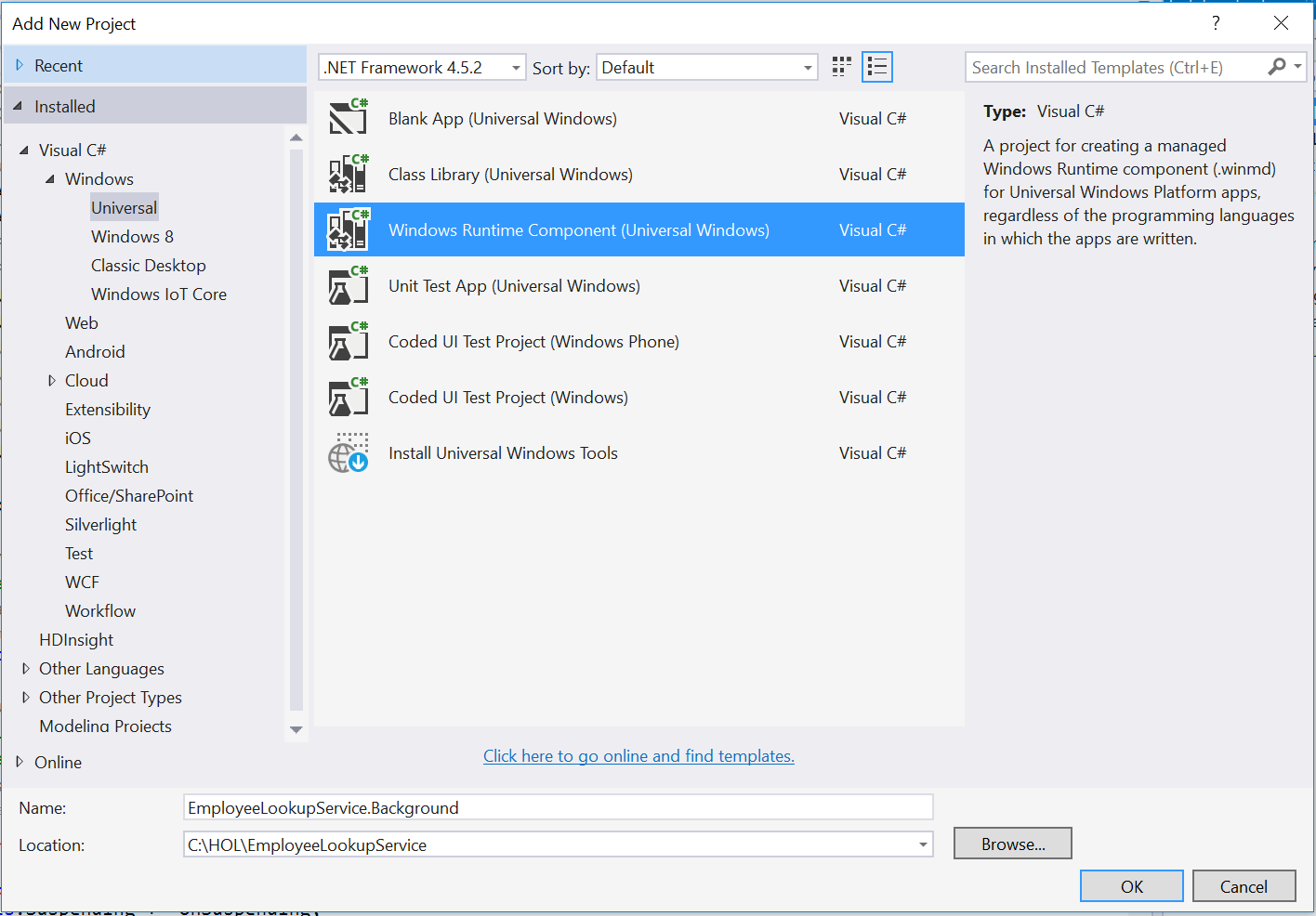


Рисунок 2

Добавьте компонент WinRT для Службы приложения.

1. Щелкните правой кнопкой мыши по Class1.cs в проекте EmployeeLookupService.Background и выберите Rename (Переименовать). Назовите проект **"EmployeeLookup.cs"**. Если необходимо выполнить переименование всех ссылок на "Class1", выберите **Yes (Да)**.

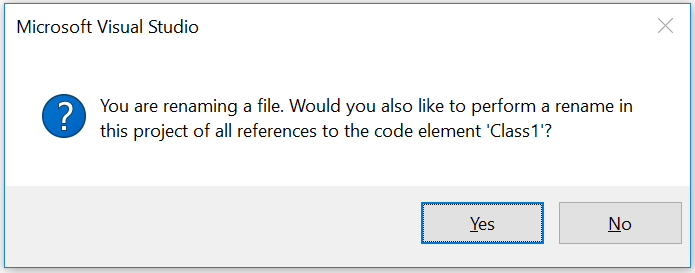


Рисунок 3

Переименуйте "Class1.cs" в "EmployeeLookup.cs".

Задача 3 – Реализовать интерфейс IBackgroundTask

* 1. Сейчас, когда ваш компонент WinRT подготовлен, вы можете реализовать базовую структуру фоновой задачи.
  2. **Примечание:** Ознакомьтесь с курсом **Background Tasks («Фоновые задачи»)** для более точного понимания фоновых задач и их структуры.

1. Откройте **EmployeeLookup.cs** и добавьте **пространство имен** Windows.ApplicationModel.Background**.**
   * 1. C#
   1. using Windows.ApplicationModel.Background;
2. Implement the IBackgroundTask interface.
   * 1. C#
   1. public sealed class EmployeeLookup : IBackgroundTask
   2. {
   3. public void Run(IBackgroundTaskInstance taskInstance)
   4. {
   5. throw new NotImplementedException();
   6. }
   7. }
3. Создайте задержку на уровне классов и откройте ее в методе **Run (Запуска)**. Подпишитесь на событие **taskInstance.Canceled**, чтобы закрыть задержку при отмене задачи.
   * 1. C#
   1. private BackgroundTaskDeferral deferral;
   2. public void Run(IBackgroundTaskInstance taskInstance)
   3. {
   4. this.deferral = taskInstance.GetDeferral();
   5. taskInstance.Canceled += TaskInstance\_Canceled;
   6. }
   7. private void TaskInstance\_Canceled(IBackgroundTaskInstance sender, BackgroundTaskCancellationReason reason)
   8. {
   9. if (this.deferral != null)
   10. {
   11. this.deferral.Complete();
   12. }
   13. }

Задача 4 – Осуществить управление соединением со Службой приложения

* 1. Служба приложения, которую мы создаем, считает ID сотрудника в качестве входных данных и выдаст соответствующее ФИО сотрудника. В рамках данной задачи вы сможете создать простой справочник с указанием информации о сотрудниках, выявлять входящее соединение со Службой приложения, а также возвращать сообщение с запрашиваемой информацией.
  2. **Примечание:** AppServiceConnection – это асинхронное соединение с точкой входа Службы приложения, открытой за счет вызова приложения. Вызывающее приложение может отправить сообщение в Службу приложения, которая может обработать сообщение и возвратить ответ. Вы научитесь открывать соединение в следующем упражнении. Для ознакомления с дополнительной информацией о AppServiceConnections посетите страницу: <https://msdn.microsoft.com/en-us/library/windows/apps/windows.applicationmodel.appservice.appserviceconnection.aspx>

1. Создайте закрытый справочник на уровне классов для хранения данных о поиске сотрудников. Возможно предоставление информации одного из четырех сотрудников.
   * 1. C#
   1. public sealed class EmployeeLookup : IBackgroundTask
   2. {
   3. private BackgroundTaskDeferral deferral;
   4. private Dictionary<string, string> employees = new Dictionary<string,
   5. string> { { "0", "John Smith" }, { "1", "Mary Echo" }, { "2", "Jane
   6. Doe" }, { "3", "Bob Harvey" } };
2. Добавьте пространтсво имен **Windows.ApplicationModel.AppService** в **EmployeeLookup.cs**.
   * 1. C#
   1. using Windows.ApplicationModel.AppService;
3. Добавьте переменную **appServiceConnection** на уровень классов. При реализации метода **Run (Запуска)** установите **appServiceConnection** на входящее соединение со Службой приложения относительно задачи.
   * 1. C#
   1. public sealed class EmployeeLookup : IBackgroundTask
   2. {
   3. private BackgroundTaskDeferral deferral;
   4. private AppServiceConnection appServiceConnection;
   5. private Dictionary<string, string> employees = new Dictionary<string,
   6. string> { { "0", "John Smith" }, { "1", "Mary Echo" }, { "2", "Jane
   7. Doe" }, { "3", "Bob Harvey" } };
   9. public void Run(IBackgroundTaskInstance taskInstance)
   10. {
   11. this.deferral = taskInstance.GetDeferral();
   12. taskInstance.Canceled += TaskInstance\_Canceled;
   13. var trigger = taskInstance.TriggerDetails as AppServiceTriggerDetails;
   14. this.appServiceConnection = trigger.AppServiceConnection;
   15. }
4. Подпишитесь на событие подключения к Службе приложения **RequestReceived** по окончании метода **Run**. После обработки запроса задержка будет считаться завершенной.
   * 1. C#
   1. this.appServiceConnection = trigger.AppServiceConnection;
   2. this.appServiceConnection.RequestReceived +=
   3. AppServiceConnection\_RequestReceived;
   4. }
   5. private void AppServiceConnection\_RequestReceived(AppServiceConnection sender, AppServiceRequestReceivedEventArgs args)
   6. {
   7. var deferral = args.GetDeferral();
   8. deferral.Complete();
   9. }
5. Добавьте пространство имен **Windows.Foundation.Collections** к классу EmployeeLookup**.**
   * 1. C#
   1. using Windows.Foundation.Collections;
6. Любое приложение, которое вызывает службу поиска сотрудников, должно будет отправить сообщение с ID сотрудника для последующего поиска. Простой объект может быть передан в фоновую задачу, использующую ValueSet. В обработчике событий AppServiceConnection\_RequestReceived получите входящее сообщение и возвратите ValueSet в ответ, содержащий результаты поиска сотрудников. ID сотрудника будет являться ключом для ответа, а его или ее ФИО будет являться значением.
   * 1. C#
   1. private async void AppServiceConnection\_RequestReceived(AppServiceConnection sender, AppServiceRequestReceivedEventArgs args)
   2. {
   3. var deferral = args.GetDeferral();
   4. var requestMessage = args.Request.Message;
   5. var responseMessage = new ValueSet();
   6. foreach (var item in requestMessage)
   7. {
   8. if (employees.ContainsKey(item.Value.ToString()))
   9. {
   10. responseMessage.Add(item.Value.ToString(),
   11. employees[item.Value.ToString()]);
   12. }
   13. }
   14. await args.Request.SendResponseAsync(responseMessage);
   15. deferral.Complete();
   16. }
   17. **Примечание:** ValueSet представляет собой карту с ключом типа строки и значением типа объекта. Только сериализуемые типы могут быть представлены на карте. Для ознакомления с дополнительной информацией о ValueSets посетите страницу: <https://msdn.microsoft.com/en-us/library/windows/apps/windows.foundation.collections.valueset.aspx>

Задача 5 – Зарегистрировать Службу приложения

Служба приложения должна быть зарегистрирована таким образом, чтобы быть доступной. Мы зарегистрируем Службу приложения EmployeeLookup в соответствующем приложении и определим точку входа в манифесте.

1. Откройте пакет **package.appxmanifest** приложения EmployeeLookupService в редакторе манифеста.
2. На вкладке **Declarations (Объявления)** добавьте объявление **App Service (Служба приложения)**. Присвойте ему имя **"EmployeeLookupService"** и установите точку входа на **EmployeeLookupService.Background.EmployeeLookup**. Сохраните манифест.

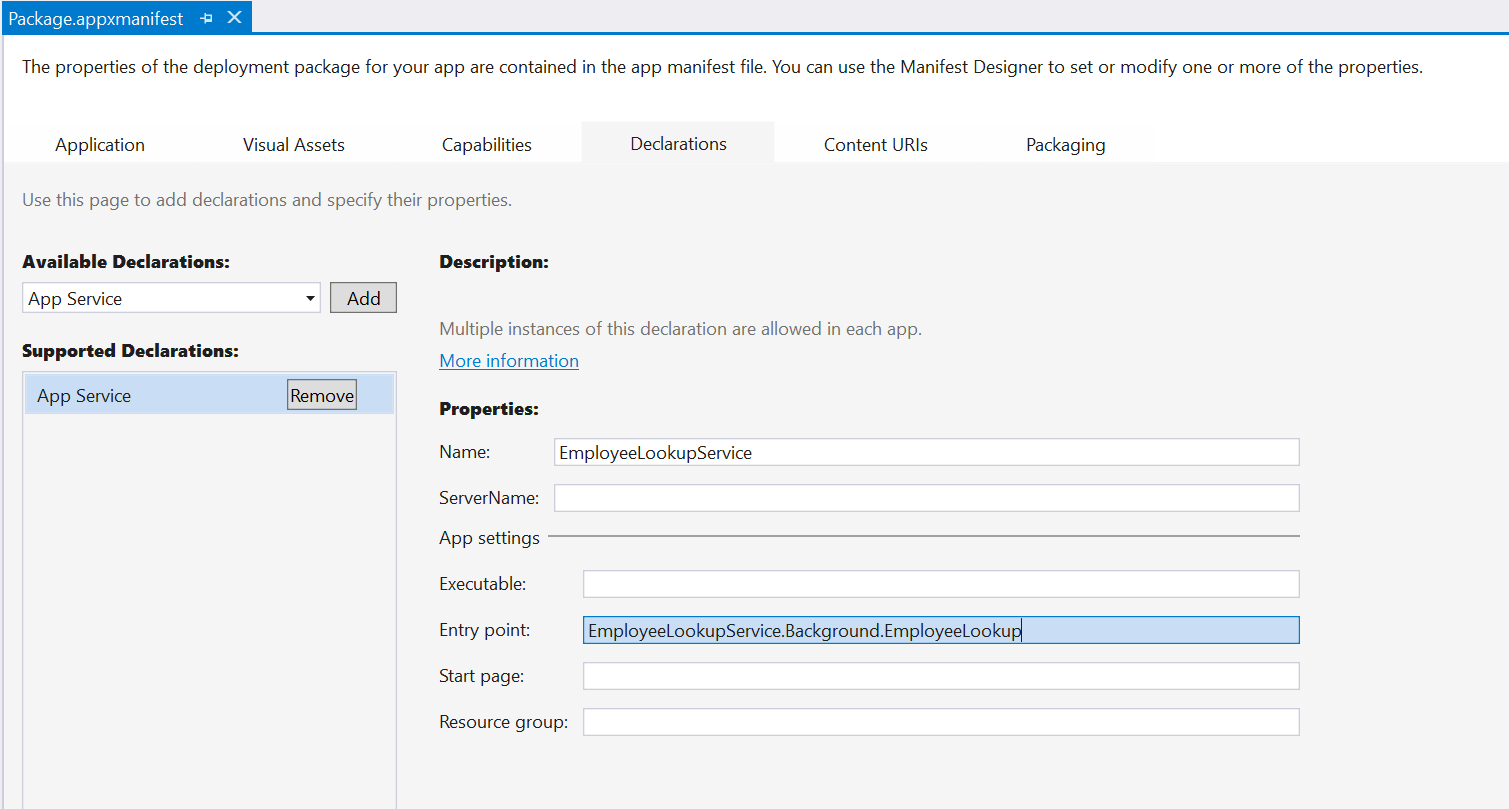


Рисунок 4

Зарегистрируйте Службу приложения в манифесте EmployeeLookupService.

1. Щелкните правой кнопкой мыши на папку **References (Ссылки)** в приложении EmployeeLookupService, а также выберите **Add (Добавить) > Reference (Ссылка)**. Добавьте ссылку в проект **EmployeeLookupService.Background**.

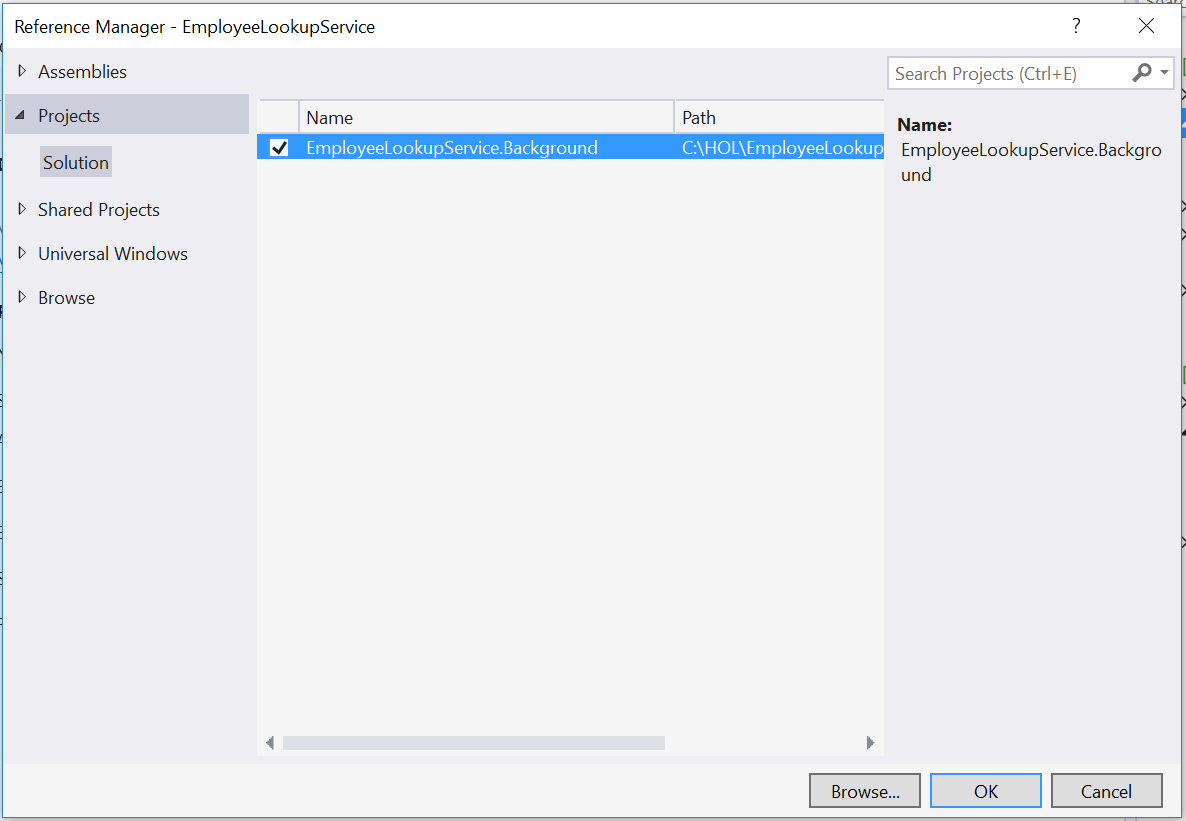


Рисунок 5

Добавьте компонент WinRT в качестве ссылки.

1. Создайте и запустите приложение EmployeeLookupService, чтобы развернуть свою фоновую задачу. Главная страница приложения будет оставаться пустой, так как мы не реализовали Пользовательский интерфейс для этого приложения.
   1. **Примечание:** Мы будет рассматривать способы отладки фоновой задачи в следующем упражнении.
2. Отключите отладку и вернитесь в Visual Studio.

Упражнение 2: Вызов службы приложения из другого приложения

1. Сейчас, когда вы уже создали и зарегистрировали Службу приложения, вы можете вызвать ее из другого приложения. В этом упражнении вы создадите приложение, в которое вы сможете ввести ID сотрудников, вызвать службу EmployeeLookup и получить результаты с соответствующим ФИО сотрудника.

Задача 1 – Создать пустое приложение Universal Windows

Мы начнем с создания проекта на основе шаблона Blank App (Пустого приложения).

1. В новой версии Visual Studio 2015 выберите **File (Файл) -> New (Новый) -> Project (Проект)**, чтобы открыть диалоговое окно New Project (Новый проект). Далее **Installed (Установленное) > Templates (Шаблоны) > Visual C# > Windows > Universal**, а затем выберите шаблон **Blank App приложения (Universal Windows)**.
2. Назовите свой проект **"AppServices"** и выберите местоположение файловой системы, в которое будет осуществлено сохранение результатов прохождения Лабораторного практикума.

Не изменяйте настройки, установленные для **Create new solution (Создания нового решения)** и **Create directory for solution (Создания папки для решения)**. Вы можете снять галочки как с **Add to source control (Добавить в систему контроля версий)**, так и **Show telemetry in the Windows Dev Center (Отобразить телеметрию в Windows Dev Center)**, если не хотите обновлять версию своей работы или использовать инструмент Application Insights. Нажмите **OK** для создания проекта.

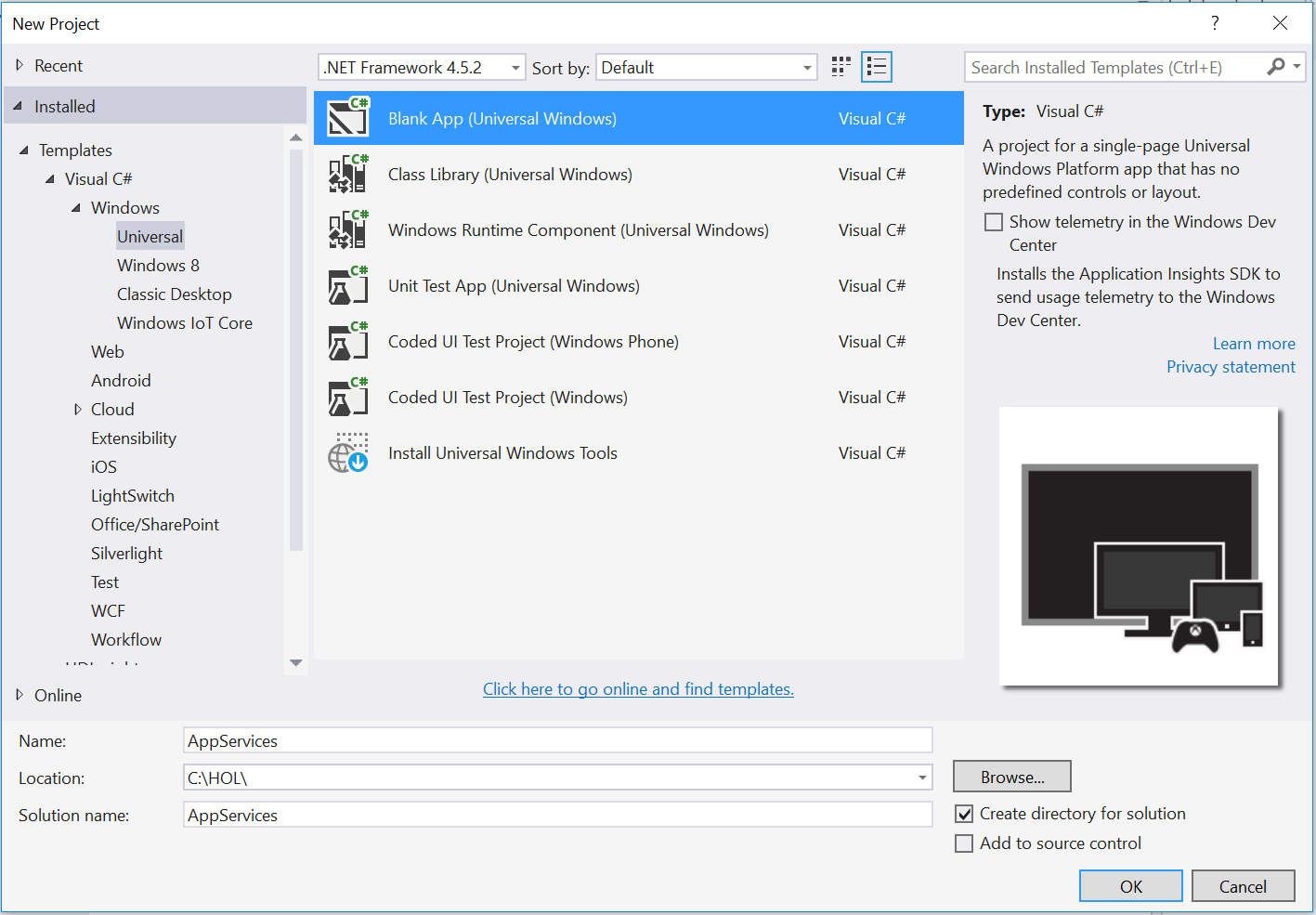


Рисунок 6

Создайте новый проект Blank App (Пустое приложение) в Visual Studio 2015.

1. Настройте Solution Configuration (Текущую конфигурацию решения) на **Debug (Отладку)** и Solution Platform (Платформу решений) в соответствии с **x86**. Выберите **Local Machine (Локальный компьютер)** из выпадающего меню Debug Target (Цели отладки).

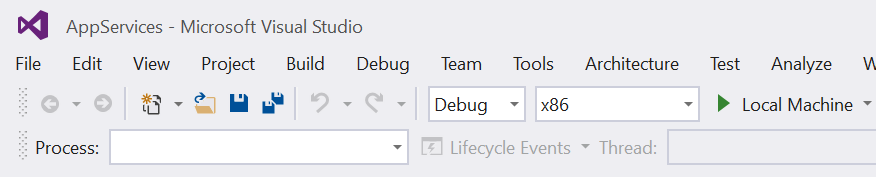


Рисунок 7

* + 1. Сконфигурируйте свое приложение, которое должно выполниться на Локальном компьютере.

1. Создайте и запустите свое приложение. Вы увидите окно Blank App со счетчиком частоты кадров, активированном по умолчанию для отладки.

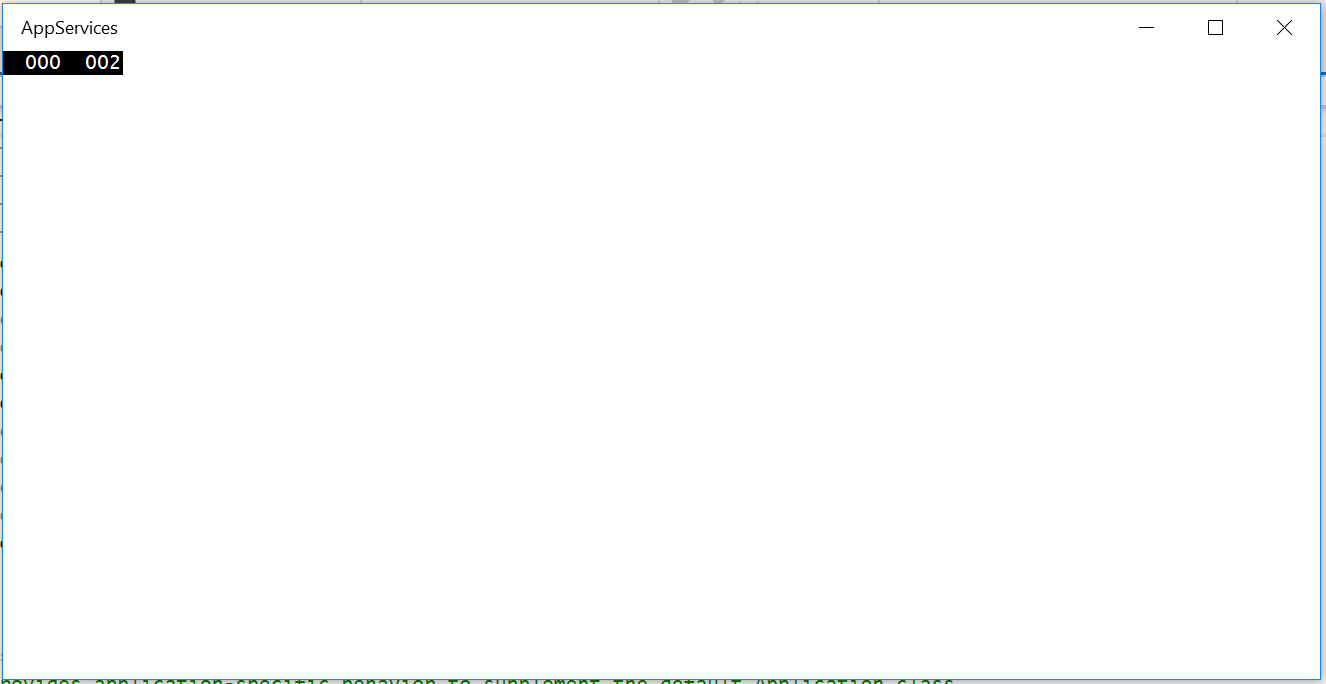


Рисунок 8

Приложение Blank universal выполняется в режиме Рабочего стола.

* 1. **Примечание:** Счетчик частоты кадров является инструментом, используемым в процессе отладки, который помогает следить за производительностью вашего приложения. Он полезен для тех приложений, которые требуют интенсивной графической обработки, однако не подходит для простых приложений, которые будут создаваться вами на данный момент.
  2. В шаблоне Blank App директива препроцессора активирует или отключает счетчик частоты кадров посредством **App.xaml.cs**. Счетчик частоты кадров может перекрывать или скрывать контент вашего приложения, если не свернуть его. При выполнении данных работ вы можете отключить его, отметив **this.DebugSettings.EnableFrameRateCounter** как **False (Ложное)**.

1. Вернитесь к Visual Studio и отключите отладку.

Задача 2 – Создать Пользовательский интерфейс и простую модель

* 1. Мы создадим простой класс Employee (Сотрудников) со свойствами ID и ФИО для упрощения процедуры отображения результатов поиска сотрудника на основе Службы поиска. Затем мы создадим простой Пользовательский интерфейс, чтобы разрешить пользователям вводить ID и запрашивать Службу приложения.

1. Щелкните правой кнопкой мыши по проекту AppServices и выберите **Add (Добавить) > Class (Класс)**. Называйте класс **"Employee.cs"**.
2. Сделайте класс Employee (Сотрудников) **открытым** и присвойте ему строковые свойства **ID** и **ФИО**.
   * 1. C#
   1. public class Employee
   2. {
   3. public string Id { get; set; }
   4. public string Name { get; set; }
   5. }
3. Сохраните и закройте класс Employee (Сотрудники).
4. Добавьте панель **StackPanel**, содержащую элементы управления **TextBlock (Текстовый блок)**, **TextBox (Текстовое окно)** и **Button (Кнопку)**, к элементам Grid для MainPage (Главной страницы) приложения AppServices.
   * 1. XAML
   1. <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
   2. <StackPanel>
   3. <TextBlock Text="Enter one employee ID per line." Margin="12" />
   4. <TextBox x:Name="EmployeeId" AcceptsReturn="True"
   5. Margin="12,0,12,12"/>
   6. <Button Content="Look up employee(s)" Margin="12,0,0,0"/>
   7. </StackPanel>
   8. </Grid>
5. Добавьте событие Click кнопки, чтобы вызвать обработчик **GetEmployeeById**. Вы создадите обработчик на следующем этапе.
   * 1. XAML
   1. <Button Content="Look up employee(s)" Click="GetEmployeeById"
   2. Margin="12,0,0,0"/>
6. Создайте обработчик **GetEmployeeById** в вспомогательном коде MainPage.
   * 1. C#
   1. public MainPage()
   2. {
   3. this.InitializeComponent();
   4. }
   5. private void GetEmployeeById(object sender, RoutedEventArgs e)
   6. {

}

Задача 3 – Открыть соединение со Службой приложения

Служба приложения, созданная вами в Упражнении 1, настроена на получение входящего соединения со Службой приложения. В рамках данной задачи вы откроете соединение со Службой приложения и отправите сообщение в Службу приложения с запросом на поиск сотрудников.

1. Добавьте пространство имен **Windows.ApplicationModel.AppService** к вспомогательному коду MainPage.
   * 1. C#
   1. using Windows.ApplicationModel.AppService;
2. Создайте соединение со Службой приложения на уровне классов MainPage.
   * 1. C#
   1. public sealed partial class MainPage : Страница
   2. {
   3. private AppServiceConnection appServiceConnection;
3. В отдельный версии Visual Studio вы можете загрузить проект **EmployeeLookupService** и открыть манифест приложения в редакторе манифеста. Под вкладкой **Packaging (Пакет)** поместите **Package family name (Наименование пакета для семейства программных систем)** и скопируйте данное наименование в свой буфер обмена.

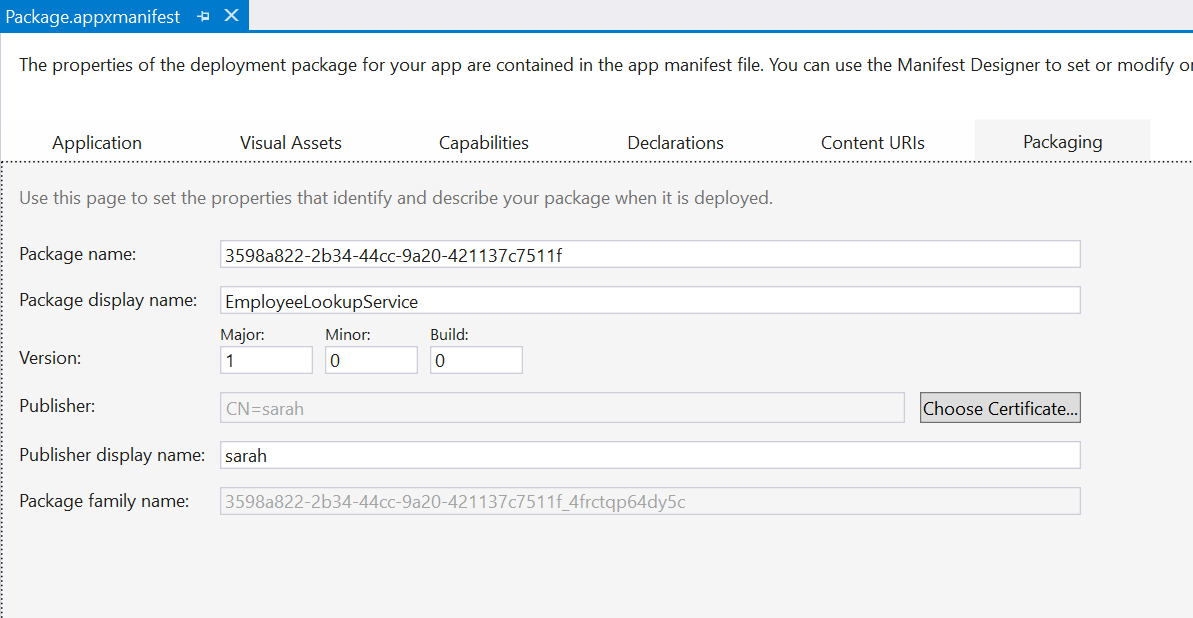


Рисунок 9

Скопируйте Package family name (Наименование пакета для семейства программных систем) из манифеста приложения EmployeeLookupService.

1. Настройте соединение со Службой приложения в обработчике **GetEmployeeById()**, чтобы иметь доступ к своей Службе приложения **EmployeeLookup**. Вам понадобится наименование службы, заданной вами в манифесте Службы приложения, а также Наименование пакета для семейства программных систем, которое вы сохранили на последнем этапе.
   * 1. C#
   1. private async void GetEmployeeById(object sender, RoutedEventArgs e)
   2. {
   3. appServiceConnection = new AppServiceConnection
   4. {
   5. // Replace the AppServiceName and PackageFamilyName with those from
   6. // your EmployeeLookupServiceApp.
   7. AppServiceName = "EmployeeLookup",
   8. PackageFamilyName =
   9. "3598a822-2b34-44cc-9a20-421137c7511f\_4frctqp64dy5c"
   10. };
2. Сделайте обработчик **GetEmployeeById()** асинхронным, а также откройте соединение со Службой приложения. На основе статуса соединения мы уведомим пользователя о наличии любых ошибок. Вы создадите метод **LogError** на следующем этапе.
   * 1. C#
   1. private async void GetEmployeeById(object sender, RoutedEventArgs e)
   2. {
   3. appServiceConnection = new AppServiceConnection
   4. {
   5. AppServiceName = "EmployeeLookup",
   6. PackageFamilyName =
   7. "3598a822-2b34-44cc-9a20-421137c7511f\_4frctqp64dy5c"
   8. };
   9. var status = await appServiceConnection.OpenAsync();
   10. switch (status)
   11. {
   12. case AppServiceConnectionStatus.AppNotInstalled:
   13. await LogError("The EmployeeLookup application is not installed.
   14. Please install it and try again.");
   15. return;
   16. case AppServiceConnectionStatus.AppServiceUnavailable:
   17. await LogError("The EmployeeLookup application does not have the
   18. available feature");
   19. return;
   20. case AppServiceConnectionStatus.AppUnavailable:
   21. await LogError("The package for the app service is unavailable.
   22. Have you added the WinRT component as a reference
   23. to the EmployeeLookupService app?");
   24. return;
   25. case AppServiceConnectionStatus.Unknown:
   26. await LogError("Unknown Error.");
   27. return;
   28. }
3. Добавьте пространства имен **System.Threading.Tasks** и **Windows.UI.Popups** к вспомогательному коду MainPage.
   * 1. C#
   1. using System.Threading.Tasks;
   2. using Windows.UI.Popups;
4. Добавьте метод **LogError** во вспомогательный код для обработки статуса сообщений, созданных вами во время Этапа 5. Настоящий метод откроет диалоговое окно сообщения с указанием статуса соединения со Службой приложения, если оно изначально не будет успешно открыто.
   * 1. C#
   1. private async Task LogError(string errorMessage)
   2. {
   3. await new MessageDialog(errorMessage).ShowAsync();
   4. }
5. В обработчике **GetEmployeeById()** выделите ID сотрудников из **входных данных** типа EmployeeId, созданных ранее, для отображения на MainPage (Главной странице). Вы можете указать атрибут **AcceptsReturn="True"** в данном TextBox (Текстовом окне). Мы позволим пользователям запрашивать службу поиска сотрудников среди огромного количества сотрудников и их ID до тех пор, пока у каждого ID не появится собственная строка.
   * 1. C#
   1. case AppServiceConnectionStatus.Unknown:
   2. await LogError("Unknown Error.");
   3. return;
   4. }

var items = this.EmployeeId.Text.Split(new string[] { Environment.NewLine },

* 1. StringSplitOptions.RemoveEmptyEntries);

1. Создайте сообщение, состоящее из ValueSet(), чтобы отправить его в Службу приложения. Добавьте каждый отобранный ID в сообщение.
   * 1. C#

var items = this.EmployeeId.Text.Split(new string[] { Environment.NewLine },

* 1. StringSplitOptions.RemoveEmptyEntries);
  2. var message = new ValueSet();
  3. for (int i = 0; i<items.Length; i++)
  4. {
  5. message.Add(i.ToString(), items[i]);
  6. }

1. Отправьте сообщение в Службу приложения и дожидайтесь ответа. Создайте переключатель для обработки потенциального предоставления статуса ответа.
   * 1. C#
   1. for (int i = 0; i<items.Length; i++)
   2. {
   3. message.Add(i.ToString(), items[i]);
   4. }
   5. var response = await appServiceConnection.SendMessageAsync(message);
   6. switch (response.Status)
   7. {
   8. case AppServiceResponseStatus.ResourceLimitsExceeded:
   9. await LogError("Insufficient resources. The app service has been shut
   10. down.");
   11. return;
   12. case AppServiceResponseStatus.Failure:
   13. await LogError("Failed to receive response.");
   14. return;
   15. case AppServiceResponseStatus.Unknown:
   16. await LogError("Unknown error.");
   17. return;

}

Задача 4 – Отобразить результаты и отладить фоновую задачу

Ваш Пользовательский интерфейс может обработать входные данные, а ваша Служба приложения уже готова к запросам. Последний шаг заключается в отображении результатов запроса и отладке фоновой задачи, если это необходимо. В рамках этой задачи вы сможете добавить ListView в Пользовательский интерфейс, чтобы отобразить один или несколько результатов запроса.

1. К **MainPage.xaml.cs** добавьте следующее пространство имен: System.Collections.ObjectModel.
   * 1. C#
   1. using System.Collections.ObjectModel;
2. Создайте новую версию **ObservableCollection**, которая будет содержать элементы информации, касающиеся типа **сотрудника**. Вы создали простой класс Employee (Сотрудников) в Задаче 2 настоящего упражнения.
   * 1. C#
   1. public sealed partial class MainPage : Страница
   2. {
   3. private AppServiceConnection appServiceConnection;
   4. public ObservableCollection<Employee> Items { get; set; } = new
   5. ObservableCollection<Employee>();
3. В обработчике **GetEmployeeById** добавьте каждый элемент ответа от Службы приложения в коллекцию Items (Элементов).
   * 1. C#
   1. switch (response.Status)
   2. {
   3. case AppServiceResponseStatus.ResourceLimitsExceeded:
   4. await LogError("Insufficient resources. The app service has been shut
   5. down.");
   6. return;
   7. case AppServiceResponseStatus.Failure:
   8. await LogError("Failed to receive response.");
   9. return;
   10. case AppServiceResponseStatus.Unknown:
   11. await LogError("Unknown error.");
   12. return;
   13. }
   14. foreach (var item in response.Message)
   15. {
   16. this.Items.Add(new Employee
   17. {
   18. Id = item.Key,
   19. Name = item.Value.ToString()
   20. });
   21. }
4. Добавьте **ListView** к панели **StackPanel** в MainPage.xaml. ListView отобразит ID и ФИО сотрудника при проведении запроса.
   * 1. XAML
   1. <StackPanel>
   2. <TextBlock Text="Enter one employee ID per line." Margin="12" />
   3. <TextBox x:Name="EmployeeId" AcceptsReturn="True" Margin="12,0,12,12"/>
   4. <Button Content="Look up employee(s)" Click="GetEmployeeById"
   5. Margin="12,0,0,0"/>
   6. <ListView ItemsSource="{x:Bind Items}" Grid.Column="1">
   7. <ListView.ItemTemplate>
   8. <DataTemplate x:DataType="local:Employee">
   9. <StackPanel>
   10. <StackPanel Orientation="Horizontal" Margin="12">
   11. <TextBlock Text="{x:Bind Id}" Margin="0,0,12,0" />
   12. <TextBlock Text="{x:Bind Name}" />
   13. </StackPanel>
   14. </StackPanel>
   15. </DataTemplate>
   16. </ListView.ItemTemplate>
   17. </ListView>
   18. </StackPanel>
   19. **Примечание:** Мы используем x:Bind для того, чтобы задать контекст данных для **Items (Элементов)** ListView. Для ознакомления с дополнительной информацией о привязке данных прочтите курс **Data Binding («Привязка данных»)**.
5. Создайте и запустите приложение **AppServices**. Введите ID сотрудника в текстовое окно и используйте кнопку для поиска сотрудников **Look up employee(s)**, чтобы запросить Службу поиска сотрудников.
6. Поздравляем, если ваша служба приложений отображает ФИО сотрудника при ответе на запрос! Результат будет подобен представленному ниже:

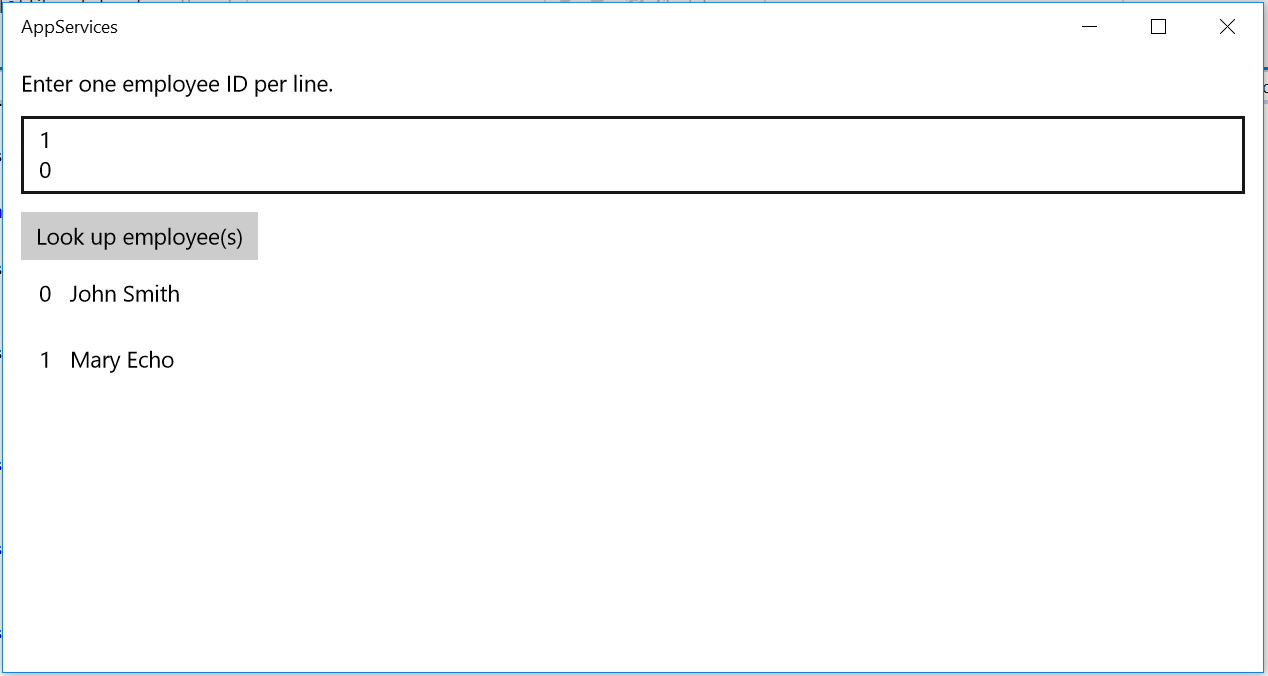


Рисунок 10

Результаты многострочного поиска сотрудников возвращаются от Службы приложения.

1. Отключите отладку и вернитесь в Visual Studio.
2. В случае возникновения ошибки при проведении поиска, давайте исправим имеющиеся неисправности. Если вы получаете ошибку статуса **AppUnavailable**, возможно вы забыли добавить проект EmployeeLookupService.Background как ссылку на приложение EmployeeLookupService.

Также ошибка статуса **AppUnavailable** может указывать на то, что объявленная вами входная точка Службы приложения не соответствует наименованию класса в классе **EmployeeLookup.cs**.

1. Для отладки самой фоновой задачи, откройте решение EmployeeLookupService.
2. Откройте свойства проекта **EmployeeLookupService** и перейдите во вкладку **Debug (Отладка)**. Отметьте пункт **"Do not launch, but debug my code when it starts" («Не запускать, но произвести отладку моего кода в качестве начального шага»)**, а затем сохраните свойства файла.

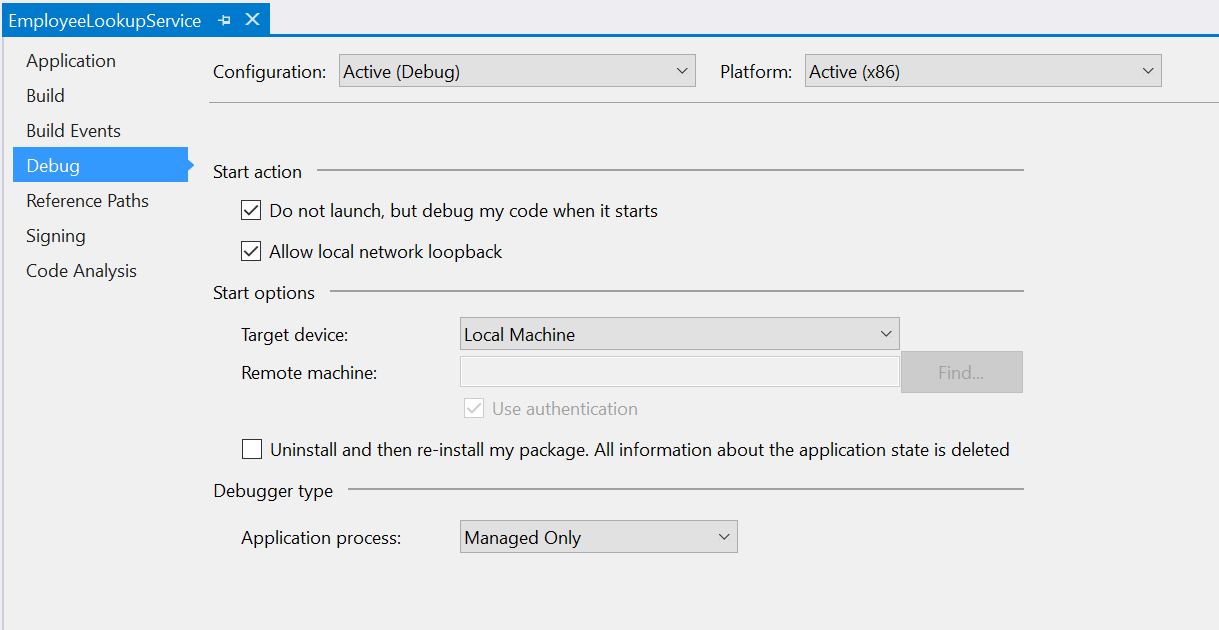


Рисунок 11

Настройте опции отладки таким образом, чтобы можно было произвести отладку фоновой задачи.

1. Задайте точку прерывания в своей фоновой задаче и используйте кнопку **Start Debugging (Начать отладку)**, чтобы подключить отладчик. Приложение EmployeeLookupService не запускается.
2. Создайте и запустите приложение **AppServices**, а также запросите Службу поиска. Вы пройдете точку прерывания в своей фоновой задаче.

Краткий обзор

* 1. В рамках настоящего курса вы научились создавать и регистрировать Службу приложения, а также запрашивать ее асинхронно из другого приложения, использующего соединение со Службой приложения.
  2. В случае с корпорацией вам может понадобиться форматирование своей Службы приложения в SDK с последующим предоставлением доступа через библиотеку классов. Для этих целей, лучше всего использовать архитектурный стиль REST с целью обновления собственной Службы приложения и создания новых входных точек для изменений в прерывании.