Создание интерфейса для работы с веб-сервисом

Последнее обновление: 17.09.2016

Продолжим проект из прошлой темы и создадим для него графический интерфейс. Для этого добавим две страницы XAML, которые назовем **FriendsListPage.xaml** (для вывода списка друзей) и **FriendPage.xaml** (для добавления/изменения друга).

И также добавим в проект класс ApplicationViewModel, через который будет идти взаимодействие графического интерфейса и веб-сервера:

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Windows.Input;
using Xamarin.Forms;
using System.Linq;
using System.Threading.Tasks;

namespace MobileClient
{
    public class ApplicationViewModel :
INotifyPropertyChanged
    {
        bool initialized = false;   // была ли начал
инициализация
        Friend selectedFriend;  // выбранный друг
        private bool isBusy;    // идет ли загрузка
сервера

        public ObservableCollection<Friend> Friends
get; set; }
        FriendsService friendsService = new
FriendsService();
        public event PropertyChangedEventHandler
PropertyChanged;

        public ICommand CreateFriendCommand { protec
set; get; }
        public ICommand DeleteFriendCommand { protec
set; get; }
        public ICommand SaveFriendCommand { protecte
set; get; }
        public ICommand BackCommand { protected set;
}

        public INavigation Navigation { get; set; }
```

```
        public bool IsBusy
        {
            get { return isBusy; }
            set
            {
                isBusy = value;
                OnPropertyChanged("IsBusy");
                OnPropertyChanged("IsLoaded");
            }
        }
        public bool IsLoaded
        {
            get { return !isBusy; }
        }

        public ApplicationViewModel()
        {
            Friends = new ObservableCollection<Frien
            IsBusy = false;
            CreateFriendCommand = new
Command(CreateFriend);
            DeleteFriendCommand = new
Command(DeleteFriend);
            SaveFriendCommand = new Command(SaveFrie
            BackCommand = new Command(Back);
        }

        public Friend SelectedFriend
        {
            get { return selectedFriend; }
            set
            {
                if (selectedFriend != value)
                {
                    Friend tempFriend = new Friend()
```

```
                    {
                        Id =value.Id,
                        Name = value.Name,
                        Email = value.Email,
                        Phone = value.Phone
                    };
                    selectedFriend = null;
                    OnPropertyChanged("SelectedFrien
                    Navigation.PushAsync(new
FriendPage(tempFriend, this));
                }
            }
        }
        protected void OnPropertyChanged(string prop
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new
PropertyChangedEventArgs(propName));
        }

        private async void CreateFriend()
        {
            await Navigation.PushAsync(new FriendPag
Friend(), this));
        }
        private void Back()
        {
            Navigation.PopAsync();
        }

        public async Task GetFriends()
        {
            if (initialized == true) return;
            IsBusy = true;
            IEnumerable<Friend> friends = await
friendsService.Get();
```

```
            // очищаем список
            //Friends.Clear();
            while (Friends.Any())
                Friends.RemoveAt(Friends.Count - 1);

            // добавляем загруженные данные
            foreach (Friend f in friends)
                Friends.Add(f);
            IsBusy = false;
            initialized = true;
        }
        private async void SaveFriend(object
friendObject)
        {
            Friend friend = friendObject as Friend;
            if (friend != null)
            {
                IsBusy = true;
                // редактирование
                if (friend.Id>0)
                {
                    Friend updatedFriend = await
friendsService.Update(friend);
                    // заменяем объект в списке на н
                    if (updatedFriend != null)
                    {
                        int pos =
Friends.IndexOf(updatedFriend);
                        Friends.RemoveAt(pos);
                        Friends.Insert(pos,
updatedFriend);
                    }
                }
                // добавление
                else
```

```
                {
                    Friend addedFriend = await
friendsService.Add(friend);
                    if(addedFriend!=null)
                        Friends.Add(addedFriend);
                }
                IsBusy = false;
            }
            Back();
        }
        private async void DeleteFriend(object
friendObject)
        {
            Friend friend = friendObject as Friend;
            if (friend != null)
            {
                IsBusy = true;
                Friend deletedFriend = await
friendsService.Delete(friend.Id);
                if(deletedFriend!=null)
                {
                    Friends.Remove(deletedFriend);
                }
                IsBusy = false;
            }
            Back();
        }
    }
}
```

В этом классе определяется коллекция Friends, где будут храниться все друзья, а также переменная friendsService для взаимодействия с сервером.

Также в классе определяется ряд команд и свойство IsBusy, которое будет указывать, находятся ли данные в процессе загрузки.

В конструкторе определяем пустой список Friends и команды

Начальные данные в список Friends загружаются дс помощью метода `GetFriends()`. В этом методе `GetFriends()` идет обращение к серверу через метод `Get()` объекта friendsService. Но в данном случае мы будем загружать данные только один раз - при старте приложения. И для этого в классе определен флаг initialized, который приобретает значение true после начальной загрузки.

Через свойство `SelectedFriend` мы будем отслеживать выделенный элемент в списке. Когда пользователь будет выбирать какой-то объект в списке, то будет происходить изменение этого свойства, будет создаваться копия выделенного друга, который будет передаваться для редактирования/удаления на страницу FriendPage:

```
if (selectedFriend != value)
{
    Friend tempFriend = new Friend()
    {
        Id =value.Id,
        Name = value.Name,
        Email = value.Email,
```

```
            Phone = value.Phone
        };
        selectedFriend = null;
        OnPropertyChanged("SelectedFriend");
        Navigation.PushAsync(new FriendPage(tempFriend,
this));
}
```

Команда CreateFriendCommand будет вызывать метод CreateFriend(), который будет передавать новый объект на страницу FriendPage:

```
private async void CreateFriend()
{
    await Navigation.PushAsync(new FriendPage(new
Friend(), this));
}
```

Команда сохранения SaveFriendCommand будет получать созданный/отредактированный объект Friend и в зависимости от режима (добавление или редактирование) использовать один из методов класса FriendService для отправки запроса.

Так как наш веб-сервис, который был создан в позапрошлой теме, при добавлении или изменении возвращает добавленный/измененный объект, то, получв этот объект от сервера, мы можем его добавить в коллекцию Friends или изменить элемент в этой коллекции.

Изменение:

```
Friend updatedFriend = await
friendsService.Update(friend);
// заменяем объект в списке на новый
if (updatedFriend != null)
{
    int pos = Friends.IndexOf(updatedFriend);
    Friends.RemoveAt(pos);
    Friends.Insert(pos, updatedFriend);
}
```

Добавление:

```
Friend addedFriend = await
friendsService.Add(friend);
if(addedFriend!=null)
    Friends.Add(addedFriend);
```

Команда DeleteFriendCommand выполняет метод DeleteFriend, который во многом аналогичен редактированию объекта:

```
Friend deletedFriend = await
friendsService.Delete(friend.Id);
if(deletedFriend!=null)
{
    Friends.Remove(deletedFriend);
}
```

Это основные моменты класса ApplicationViewModel, но надо сказать, что многое зависит от сервера и его API. В данном случае веб-сервис делаем мы сами, и возвращаение при добавлении/редактировании/удалении измененного объекта нам очень сильно помогает. Во-первых, для взаимодействия с сервером

нам необходим только один запрос: в запросе посылаем объект на добавление и в ответе получаем добавленный объект. Потом этот объект добавляем в коллекцию. Это экономичнее, чем в одном запросе отправлять добавляемый объект, а в другом запросе получать все объекты с сервера с учетом добавления. Во-вторых, в данном случае экономится мобильный траффик.

Однако используемый нами веб-сервис может не представлять подобной функциональности, как возвращение измененного объекта, либо к данным на сервере могут одновременно обращаться несколько пользователей и соответственно изменять эти данные. И в этом случае надежнее будет заново загружать данные после каждой произведенной операции.

Далее изменим страницу FriendPage. В коде Xaml определим простейший интерфейс:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/
             xmlns:x="http://schemas.microsoft.com/w
             x:Class="MobileClient.FriendPage" Title
друге">
  <StackLayout>
    <StackLayout>
      <Label Text="Имя" />
      <Entry Text="{Binding Path=Model.Name}" FontSi
      <Label Text="Электронная почта" />
      <Entry Text="{Binding Path=Model.Email}" FontS
      <Label Text="Телефон" />
```

```
        <Entry Text="{Binding Path=Model.Phone}" FontS
    </StackLayout>
    <StackLayout Orientation="Horizontal"
HorizontalOptions="CenterAndExpand">
        <Button Text="Сохранить" Command="{Binding
ViewModel.SaveFriendCommand}" CommandParameter="{Bin
        <Button Text="Удалить" Command="{Binding
ViewModel.DeleteFriendCommand}" CommandParameter="{B
/>
        <Button Text="Назад" Command="{Binding
Path=ViewModel.BackCommand}" />
    </StackLayout>
  </StackLayout>
</ContentPage>
```

А в коде FriendPage.xaml.cs пропишем передачу на страницу объектов Friend и ApplicationViewModel и установку контекста:

```
using Xamarin.Forms;

namespace MobileClient
{
    public partial class FriendPage : ContentPage
    {
        public Friend Model { get; private set; }
        public ApplicationViewModel ViewModel { get
private set; }
        public FriendPage(Friend model,
ApplicationViewModel viewModel)
        {
            InitializeComponent();
            Model = model;
            ViewModel = viewModel;
```

```
            this.BindingContext = this;
        }
    }
}
```

Далее на странице FriendsListPage.xaml определим вывод списка друзей:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/
             xmlns:x="http://schemas.microsoft.com/w
             x:Class="MobileClient.FriendsListPage"
друзей">
  <StackLayout>
    <Button Text="Добавить" Command="{Binding Create
IsEnabled="{Binding IsLoaded}" />
    <ListView x:Name="friendsList" ItemsSource="{Bin
              SelectedItem="{Binding SelectedFriend,
HasUnevenRows="True">
      <ListView.ItemTemplate>
        <DataTemplate>
          <ViewCell>
            <ViewCell.View>
              <StackLayout>
                <Label Text="{Binding Name}" FontSiz
                <Label Text="{Binding Email}" FontSi
                <Label Text="{Binding Phone}" FontSi
              </StackLayout>
            </ViewCell.View>
        </ViewCell>
        </DataTemplate>
      </ListView.ItemTemplate>
    </ListView>
    <StackLayout IsVisible="{Binding IsBusy}"
                 HorizontalOptions="Center"
VerticalOptions="CenterAndExpand" Padding="20">
```

```
        <Label Text="Загрузка данных..." TextColor="Gr
HorizontalOptions="Center" />
        <ActivityIndicator IsRunning="{Binding IsBusy}
>
        </ActivityIndicator>
    </StackLayout>
  </StackLayout>
</ContentPage>
```

А в коде FriendsListPage.xaml.cs пропишем установку контекста страницы и загрузку начальных данных:

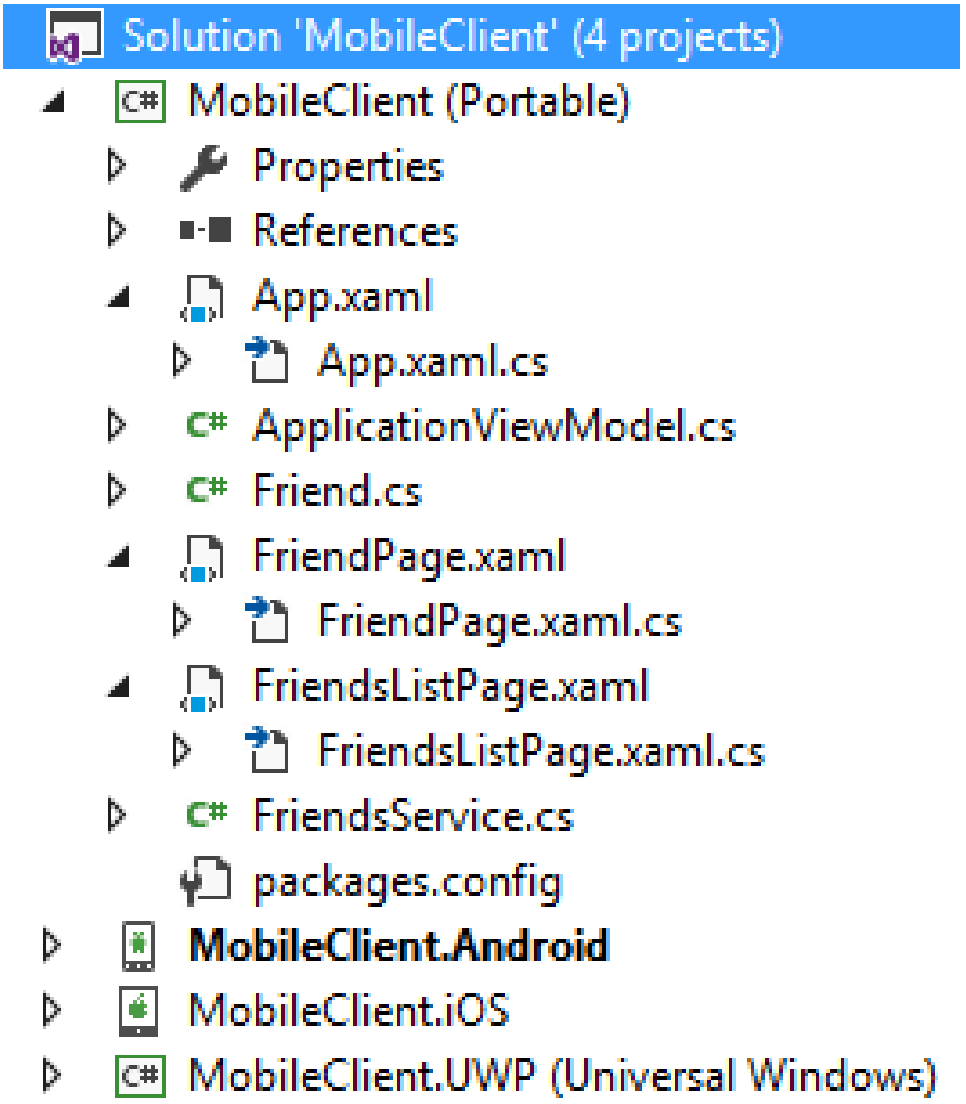
```
using Xamarin.Forms;

namespace MobileClient
{
    public partial class FriendsListPage :
ContentPage
    {
        ApplicationViewModel viewModel;
        public FriendsListPage()
        {
            InitializeComponent();
            viewModel = new ApplicationViewModel() {
Navigation = this.Navigation };
            BindingContext = viewModel;
        }

        protected override async void OnAppearing()
        {
            await viewModel.GetFriends();
            base.OnAppearing();
        }
    }
```

```
}
```

И в конце установим страницу FriendsListPage в качестве главной в классе App:

```
using Xamarin.Forms;

namespace MobileClient
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();

            MainPage = new NavigationPage(new
FriendsListPage());
        }

        protected override void OnStart() { }

        protected override void OnSleep() { }

        protected override void OnResume() { }
    }
}
```

В итоге весь проект будет выглядеть следующим образом:



Запустим веб-сервис и мобильное приложение и нажмем на кнопку добавления. Введем какие-нибудь данные:

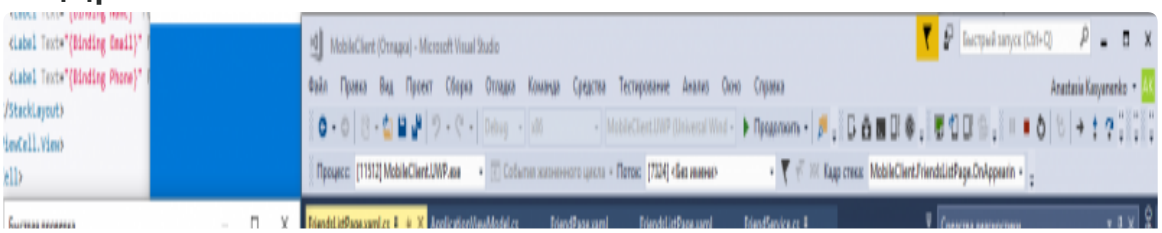
И после добавления новый объект отобразится на главной странице:

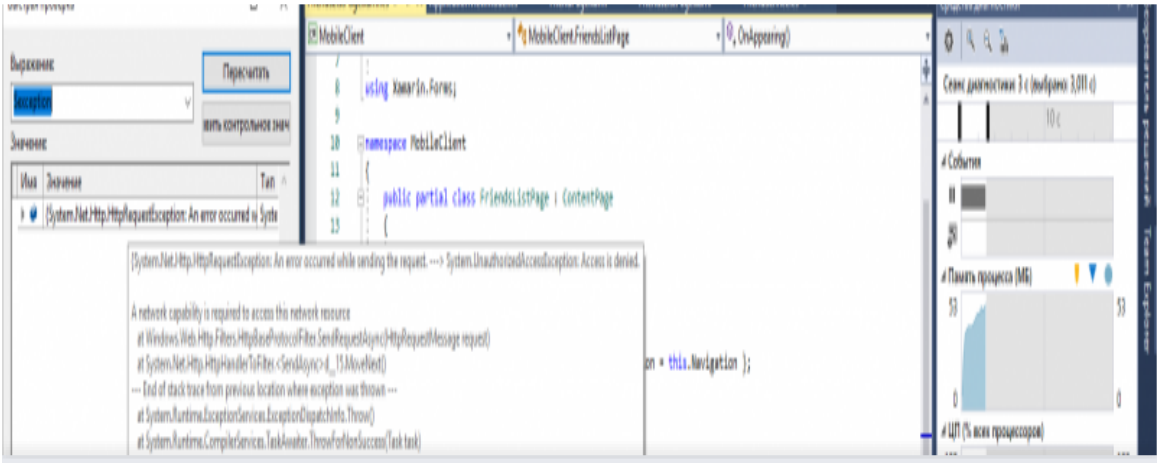

19:14
Список друзей
ДОБАВИТЬ
Иван Иванов
ican@gmail.com
+78976543210

Назад **Содержание** **Вперед**

Настюнчик Касьяненко • месяц назад

Здравствуйте, большое спасибо за данный ресурс, он мне очень помогает! Только вот на этом уроке у меня возникла проблема: я не могу запустить
приложение, вылазит ошибка: System.Net.Http.HttpRequestException: "An error occurred while sending the request.". На первой картинке видны подробности этой ошибки.





показать больше

• Ответить • Поделиться ›

Metanit Модератор ➚ Настюнчик Касьяненко • месяц назад

прежде всего на мобильном устройстве необходимо включить возможность работы с сетью. То есть на мобильном устройстве не должно быть никаких ограничений. Во-вторых, при создании приложения в файле манифеста надо прописать разрешения, как описано в статье https://metanit.com/sharp/x...
По поводу веб-сервиса, по данной картинке сложно сказать, что неправильно. Посмотрите последний по дате комментарий в статье https://metanit.com/sharp/m... Возможно поможет

• Ответить • Поделиться ›

Настюнчик Касьяненко
Metanit • месяц назад

Большое спасибо! Все заработало!
Я в первый раз создавала веб-сервис, поэтому у меня возник вопрос по поводу его запуска. Для того чтобы запустить созданное приложение MobileClient мне необходимо сначала запустить веб-сервис, то есть я открываю одновременно два окна Visual Studio: одно - MobileClient, другое веб-сервис. Запускаю на отладку веб-сервис, в командной строке, потом делаю его доступным по локальной подсети и далее запускаю приложение MobileClient. Как можно еще запустить веб-сервис без открытия и отладки его в Visual Studio?

• Ответить • Поделиться ›

Анатолий Бучнев

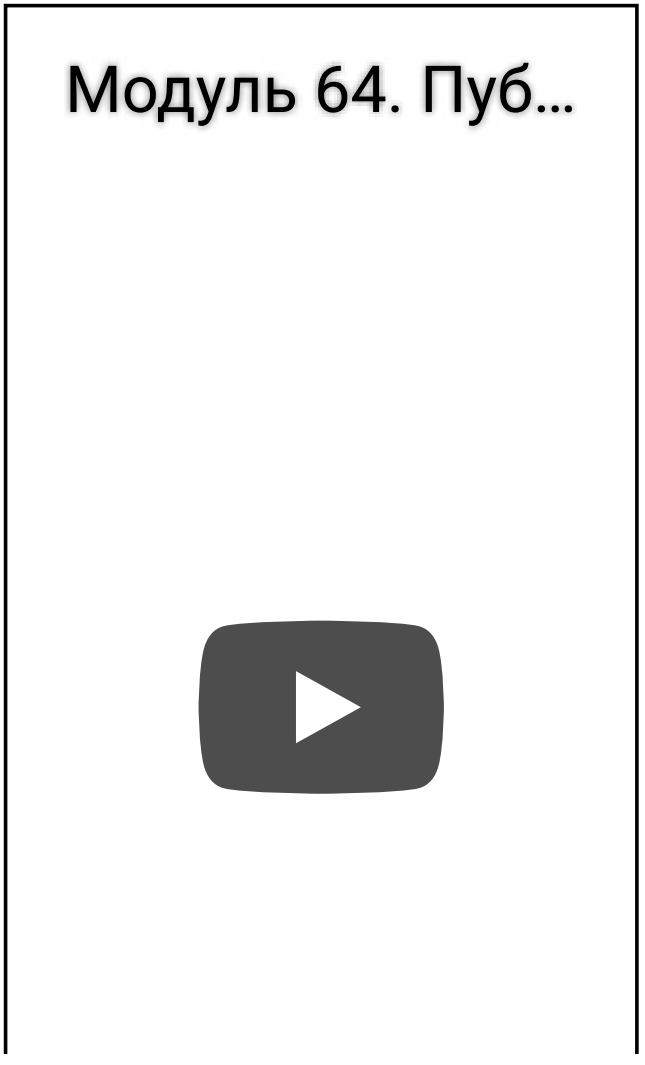
Настюнчик Касьяненко
• месяц назад

А из какого вы города?)

^ v • Ответить • Поделиться ›

Metanit Модератор
Настюнчик Касьяненко
• месяц назад

надо развернуть его на каком-нибудь хостинге. Например, у меня есть видео, как закинуть приложение на один из хостингов -



Ответить • Поделиться ›

Настюнчик Касьяненко
Metanit • месяц назад

Спасибо! Буду пробовать.

Ответить • Поделиться ›

Guest • 2 месяца назад

При попытке добавления нового элемента на сервер выводит ошибку: System.FieldAccessException: "Методу "Newtonsoft.Json.JsonSerializer..ctor()" не удалось получить доступ к полю "Newtonsoft.Json.JsonSerializerSettings.Default В чем может быть причина?

Ответить • Поделиться ›

Metanit Модератор → Guest
• 2 месяца назад

что-то неправильно сериализуете

• Ответить • Поделиться ›

Boris • 4 месяца назад

Здравствуйте! Все выполнил как описано в ваших уроках, при тестировании UWP проекта всё работает, приложение подключается к серверу, все запросы выполняются. Но при запуске Android приложения, возникает ошибка при выполнении запроса ("an error occurred while sending the request"). Появляется ошибка на строке string result = await client.GetStringAsync(Url); в методе public async Task<ienumerable<friend>> Get() класса FriendsService.

• Ответить • Поделиться ›

Metanit Модератор Boris
• 4 месяца назад