



Получение данных с сервера в json

Последнее обновление: 19.08.2016



Одним из ключевых моментов работы многих мобильных приложений является возможность взаимодействия с сервером - отправка или получение данных. И в Xamarin Forms мы тоже можем взаимодействовать с сервером. Рассмотрим на примере получения данных в формате json.

В качестве примера возьмем бесплатный API от Yahoo для получения текущих котировок валют. Так, для

получения текущих значений для пары "рубль-доллар" в данном API необходимо обратиться по адресу `https://query.yahooapis.com/v1/public/yql?q=select+*+from+yahoo.finance.xchange+where+pair+=` И в качестве ответа мы получим данные в формате json примерно следующего вида:

```
{
  "query": {
    "count": 1,
    "created": "2016-08-19T15:46:31Z",
    "lang": "ru-RU",
    "results": {
      "rate": {
        "id": "USDRUB",
        "Name": "USD/RUB",
        "Rate": "64.0425",
        "Date": "8/19/2016",
        "Time": "12:53pm",
        "Ask": "64.0513",
        "Bid": "64.0425"
      }
    }
  }
}
```

Теперь посмотрим, как мы эти данные можем получать и отображать в приложении на Xamarin Forms.

Создадим новый проект и вначале добавим в него класс, который будет представлять загружаемые данные. Назовем этот класс `RateInfo`:

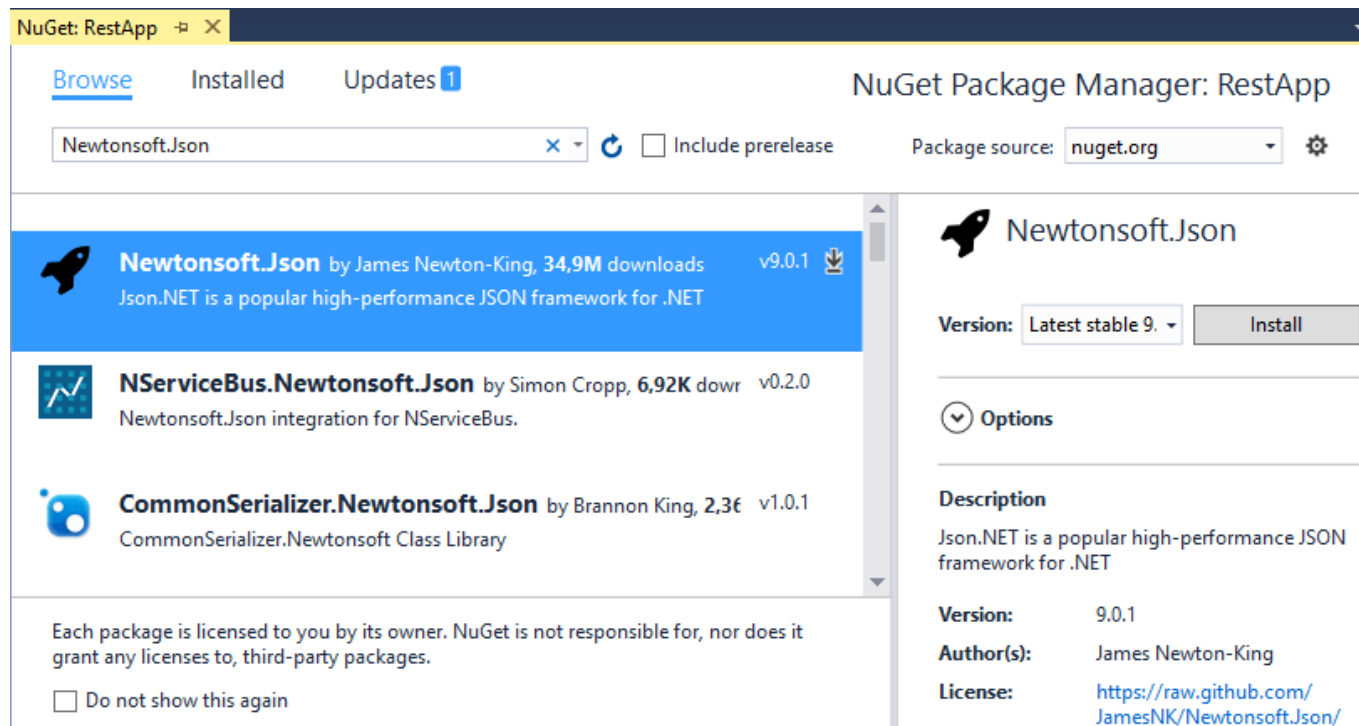
```
using System;
```

```
namespace RestApp
{
    public class RateInfo
    {
        public string Id { get; set; }
        public string Name { get; set; }
        public decimal Rate { get; set; }
        public DateTime Date { get; set; }
        public string Time { get; set; }
        public decimal Ask { get; set; }
        public decimal Bid { get; set; }
    }
}
```

Как можно заметить, данный класс соответствует части ответа от сервера в формате json:

```
"rate":{
  "id":"USDRUB",
  "Name":"USD/RUB",
  "Rate":"64.0425",
  "Date":"8/19/2016",
  "Time":"12:53pm",
  "Ask":"64.0513",
  "Bid":"64.0425"
}
```

Затем поскольку мы загружаем данные в формате json, то нам надо будет десериализовать эти данные. Для этого будем использовать библиотеку **Newtonsoft.Json**, которую добавим в проект через Nuget:



Далее добавим в проект класс RateViewModel, который будет представлять ViewModel, через которую будет идти загрузка и отображение данных на странице:

```
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using System;
using System.ComponentModel;
using System.Net.Http;
using System.Windows.Input;
using Xamarin.Forms;

namespace RestApp
{
    public class RateViewModel : INotifyPropertyChanged
    {
        private decimal rate;
        private decimal ask;
        private decimal bid;

        public decimal Rate
```

```
{
    get { return rate; }
    private set
    {
        rate = value;
        OnPropertyChanged("Rate");
    }
}

public decimal Ask
{
    get { return ask; }
    private set
    {
        ask = value;
        OnPropertyChanged("Ask");
    }
}

public decimal Bid
{
    get { return bid; }
    private set
    {
        bid = value;
        OnPropertyChanged("Bid");
    }
}

public ICommand LoadDataCommand { protected

public RateViewModel()
{
    this.LoadDataCommand = new Command(LoadD
}

private async void LoadData()
```

```
{
    string url = "https://query.yahooapis.co
q=select+*+from+yahoo.finance.xchange+where+pair+=+%

    try
    {
        HttpClient client = new HttpClient()
        client.BaseAddress = new Uri(url);
        var response = await client.GetAsync
        response.EnsureSuccessStatusCode();

        // десериализация ответа в формате j
        var content = await response.Content
        JObject o = JObject.Parse(content);

        var str = o.SelectToken("@$.query.re
        var rateInfo = JsonConvert.Deseriali

        this.Rate = rateInfo.Rate;
        this.Ask = rateInfo.Ask;
        this.Bid = rateInfo.Bid;
    }
    catch(Exception ex)
    { }
}

public event PropertyChangedEventHandler Pro
public void OnPropertyChanged(string prop =
{
    if (PropertyChanged != null)
        PropertyChanged(this, new PropertyCh
    }
}
}
```

Прежде всего надо отметить, что данный класс реализует интерфейс `INotifyPropertyChanged`. В классе определены свойства `Rate`, `Bid`, `Ask`, которые аналогичны соответствующим свойствам из класса `Rate` и данные которых будут выводиться на страницу.

Для загрузки данных определено свойство-команда `LoadDataCommand`. Эта команда при выполнении будет вызывать метод `LoadData()`.

Для взаимодействия с сервером применяется класс **`HttpClient`**. Он определяет ряд методов, через которые можно отправлять серверу данные или, наоборот, получать от него данные. В данном случае мы задействуем метод `client.GetAsync()` для получения ответа по указанному адресу.

С помощью метода `response.EnsureSuccessStatusCode()` указываем, что в случае неудачного выполнения запроса будет выброшено исключение, которое мы можем обработать.

В итоге весь код загрузки данных по определенному адресу выглядеть следующим образом:

```
HttpClient client = new HttpClient();
client.BaseAddress = new Uri(url);
var response = await
client.GetAsync(client.BaseAddress);
response.EnsureSuccessStatusCode(); // выброс
исключения, если произошла ошибка
```

После получения ответа его надо привести в тот вид, который позволит нам манипулировать содержащимися в нем данными. Для этого вначале получим текст ответа с помощью метода `response.Content.ReadAsStringAsync()`.

Поскольку текст ответа представляет данные в формате `json`, то нам их надо распарсить и привести к объекту `RateInfo`. Для этого получаем `JsonObject` для упрощения работы с данными:

```
JsonObject o = JsonObject.Parse(content);
```

Поскольку нас интересуют не все абсолютно данные, а только их часть (элемент `rate` в полученном ответе), то нам их надо извлечь:

```
var str = o.SelectToken("@$.query.results.rate");
```

Выражение `"$.query.results.rate"` представляет синтаксис языка запросов в JSON - `JsonPath`, с помощью которого мы можем обратиться к определенной порции данных. В данном случае мы получаем узел `rate`, который находится в узле `results`, который в свою очередь находится в узле `query`. То есть по факту мы получим следующий узел:

```
"rate":{  
  "id": "USDRUB",  
  "Name": "USD/RUB",  
  "Rate": "64.0425",  
  "Date": "8/19/2016",  
}
```



```
"Time": "12:53pm",  
"Ask": "64.0513",  
"Bid": "64.0425"  
}
```

И в конце десериализуем этот узел в объект RateInfo (десериализация идет на основании соответствия свойств класса и свойств в json):

```
JsonConvert.DeserializeObject<RateInfo>  
(str.ToString())
```

После этого происходит установка полученных данных:

```
this.Rate = rateInfo.Rate;  
this.Ask = rateInfo.Ask;  
this.Bid = rateInfo.Bid;
```

И в конце определим главную страницу MainPage. В коде C# определим в качестве контекста страницы объект RateViewModel:

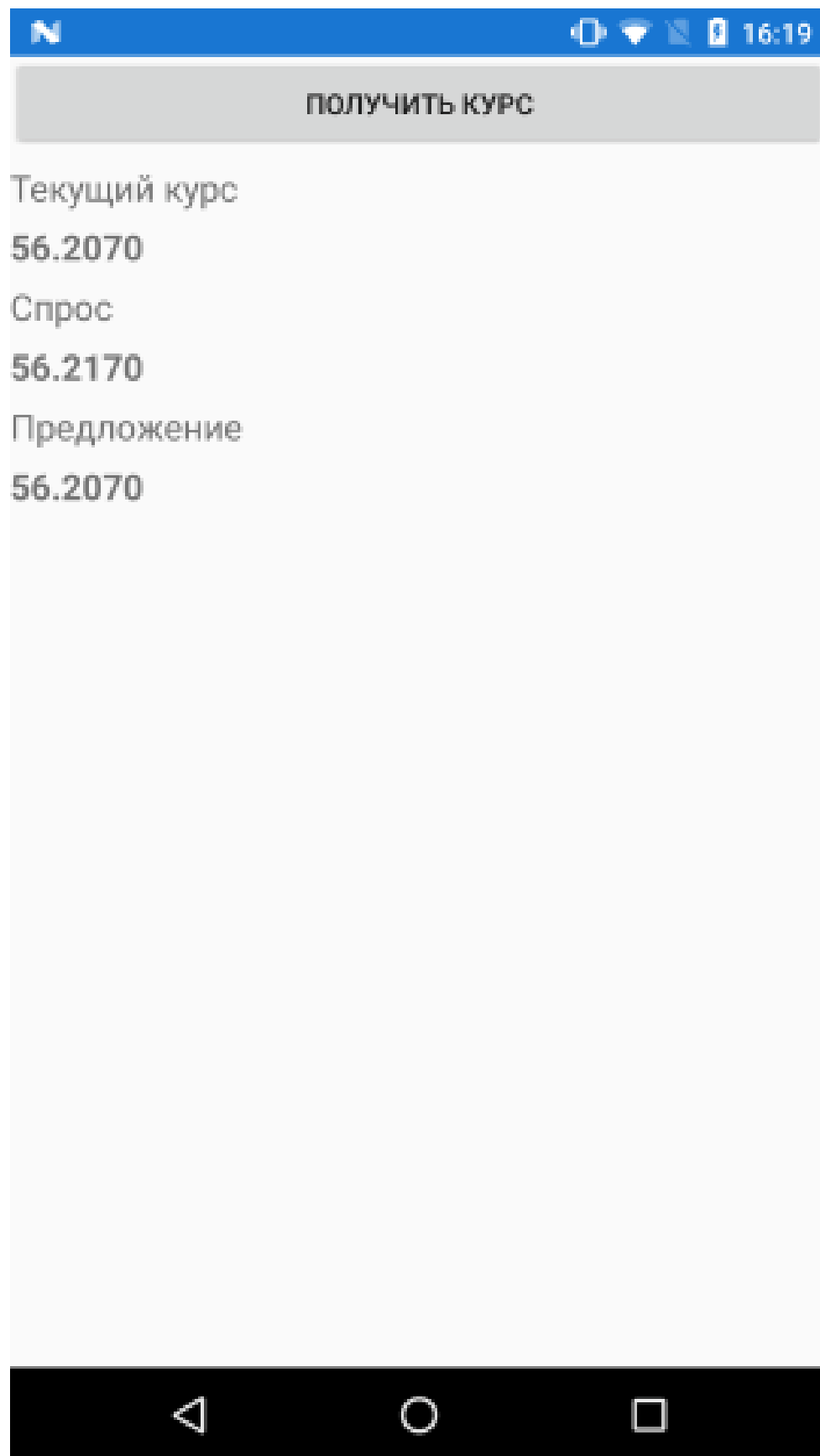
```
using Xamarin.Forms;  
  
namespace RestApp  
{  
    public partial class MainPage : ContentPage  
    {  
        RateViewModel viewModel;  
        public MainPage()  
        {  
            InitializeComponent();  
  
            viewModel = new RateViewModel();  
        }  
    }  
}
```

```
// установка контекста данных
this.BindingContext = viewModel;
    }
}
}
```

А в коде XAML определим привязку элементов к свойствам RateViewModel:

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/
    xmlns:x="http://schemas.microsoft.com/w
    x:Class="RestApp.MainPage">
    <StackLayout>
        <Button Text="Получить курс" Command="{Binding
LoadDataCommand}" />
        <Label FontSize="Medium" Text="Текущий курс" />
        <Label FontSize="Medium" FontAttributes="Bold" T
Rate}" />
        <Label FontSize="Medium" Text="Спрос" />
        <Label FontSize="Medium" FontAttributes="Bold"
Ask}" />
        <Label FontSize="Medium" Text="Предложение" />
        <Label FontSize="Medium" FontAttributes="Bold" T
Bid}" />
    </StackLayout>
</ContentPage>
```

Запустим приложение и нажмем на кнопку, и на странице отобразятся загруженные с сервера данные:



Free Guide on PowerS

A quick start guide for those who who want
PowerShell! veeam.com/Free-PowerShell-Gu

[Назад](#) [Содержание](#) [Вперед](#)





ОБРАЗОВАТЕЛЬНЫЙ
ЦЕНТР ПВТ

КУРС **iOS**
РАЗРАБОТЧИКА



Курс
Java

всего за
1 год!

Подробнее



[Комментарии](#)[Сообщество](#)[Войти](#) ▼[❤️ Рекомендовать](#)[🔗 Поделиться](#)[Лучшее в начале](#) ▼

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS



Balalamb • 3 месяца назад

Добрый день! Не получается передать данные:

Отправка:

```
string log= "qwdqwd";
```

```
HttpContent content = new StringContent(log);
```

```
HttpClient client = new HttpClient();
```

```
HttpRequestMessage request = new
```

```
HttpRequestMessage
```

```
{
```

```
RequestUri = new
```

```
request = new
```

```
Uri("http://192.168.1.107:52905/Mobile/Login"),  
Method = HttpMethod.Post,  
Content = content
```

```
};
```

```
HttpResponseMessage response = await  
client.SendAsync(request);
```

Получение на сервере:

```
[HttpPost]
```

```
public JsonResult Login(string log)
```

```
{
```

```
User us = new User() { Station_ID = 100,
```

```
uname = "wdawd", Id=12, Pwd="passw" };
```

```
return Json(us);
```

```
}
```

string log всегда null

^ | v • Ответить • Поделиться ›



Олег Волков • 3 месяца назад

Пробую использовать код примера, просто чтобы запустить.

Все время отваливается с ошибкой

Skipped 795 frames! The application may be doing too much work on its main thread.

В чем проблема? Как вообще может что-то блокироваться, если все операции

асинхронные?

ассигнованы :

^ | v • Ответить • Поделиться ›



Серега • год назад

Хотел бы задать такой вопрос есть ли у вас урок по получению данных с бд, через интернет, и отрисовка пунктов которые есть в этой бд

если нет, то вообще возможно ли это? и как, заранее спасибо

^ | v • Ответить • Поделиться ›



Metanit Модератор ➔ Серега

• год назад

в следующей теме создается веб-сервис, через который приложение на Xamarin взаимодействует с базой данных

^ | v • Ответить • Поделиться ›



Серега ➔ Metanit • год назад

А в какой теме говорится про динамическую отрисовку, ну во время работы приложения

^ | v • Ответить • Поделиться ›



Metanit Модератор ➔

Серега • год назад

тут -

<https://metanit.com/sharp/>

и тут -

<https://metanit.com/sharp/>

^ | v • Ответить •

Поделиться ›



Сергеа ➔ Metanit

• год назад

Большое вам спасибо)


^ | v • Ответить •

Поделиться ›



Майк Элеван • год назад

Ох лол, в общем имя свойств объекта для сериализации должны совпадать с таковыми полями в запросе, т.е. нам возвращают [{cid:88, title: "kort"}] - свойства в объекте cid (get; set;) и title(get; set;) соотв. иначе не будет работать + для получения массива объектов просто указать в типе массив var

AdChoices 

DC/DC
switching
regulators
designed for
ease of use
from the



[Вконтакте](#) | [Телеграм](#) | [Twitter](#) | [Google+](#) |
[Youtube](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2012-2018. Все права защищены.