

METANIT.COM



Сайт о программировании



Класс HttpClient и отправка запросов

Последнее обновление: 17.09.2016



Learn PowerShell scrip

Download our free guide and become a Powe
veeam.com/Free-PowerShell-Guide

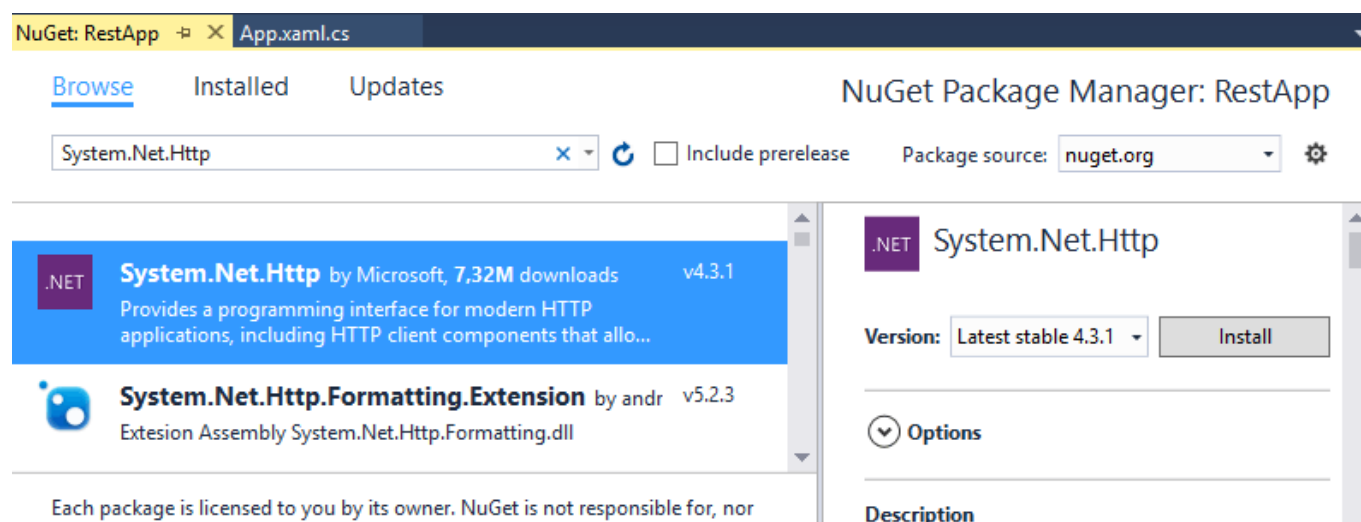
Для взаимодействия с веб-сервисами в .NET используется класс **HttpClient** из пространства имен **System.Net.Http**. С помощью его методов можно посылать определенный запрос к серверу.

Добавление HttpClient

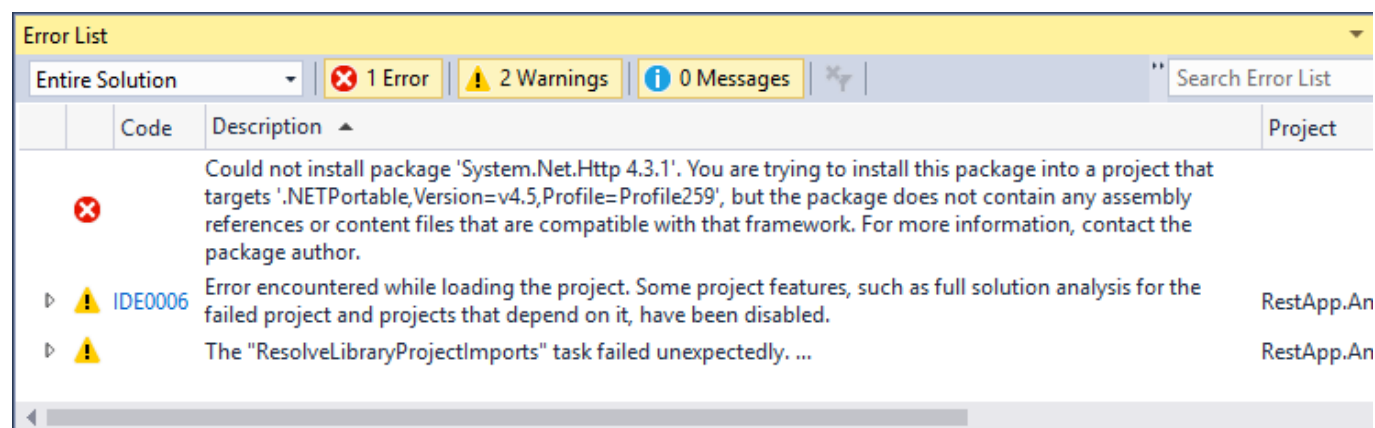
По умолчанию проект Xamarin.Forms не содержит функционала HttpClient, поэтому его надо вручную

добавлять в проект. Но здесь мы можем столкнуться с некоторыми трудностями, поэтому подробнее рассмотрим данный момент.

Прежде всего нам надо добавить через NuGet пакет **System.Net.Http**:

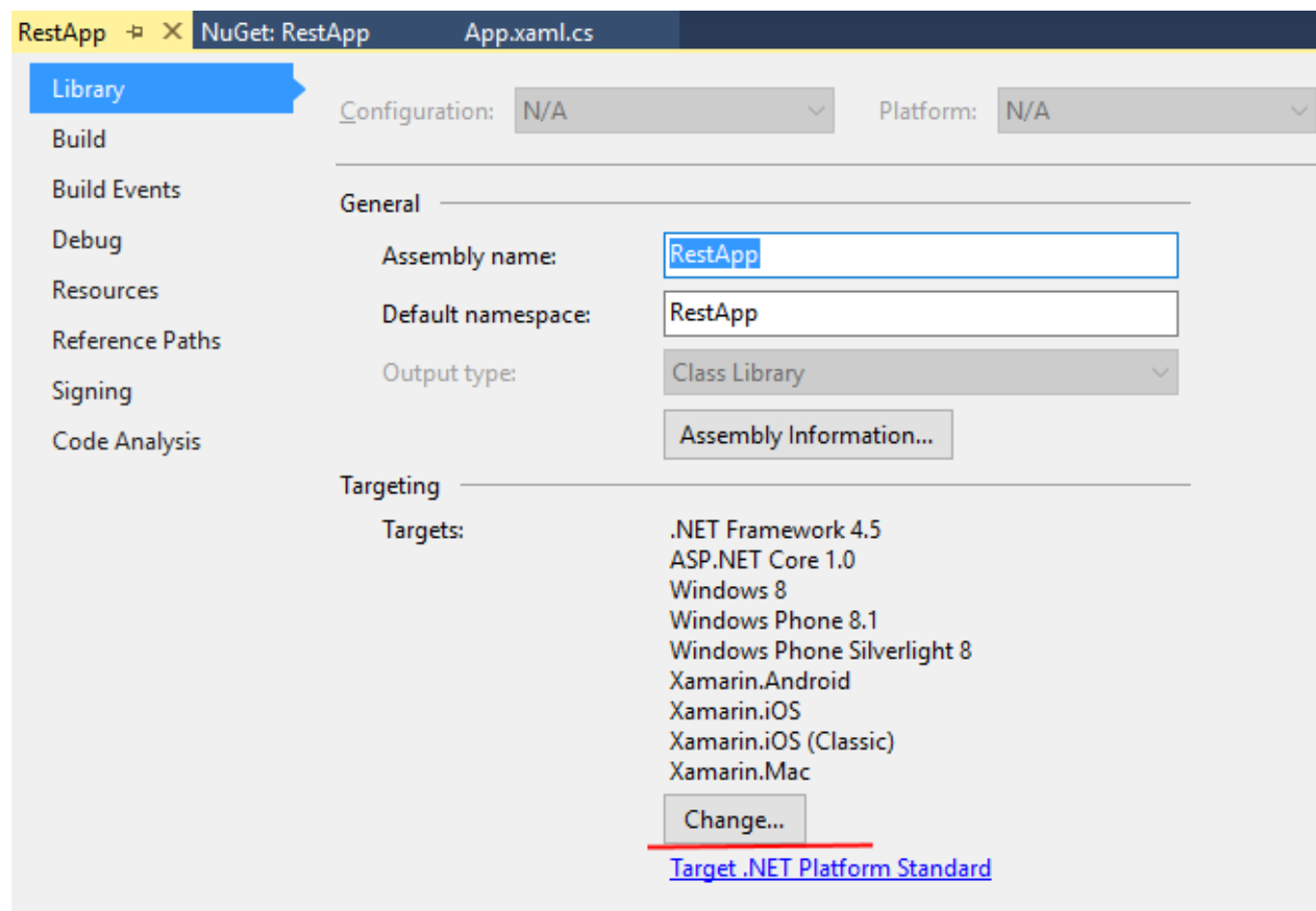


Однако в процессе добавления можно столкнуться с проблемой:



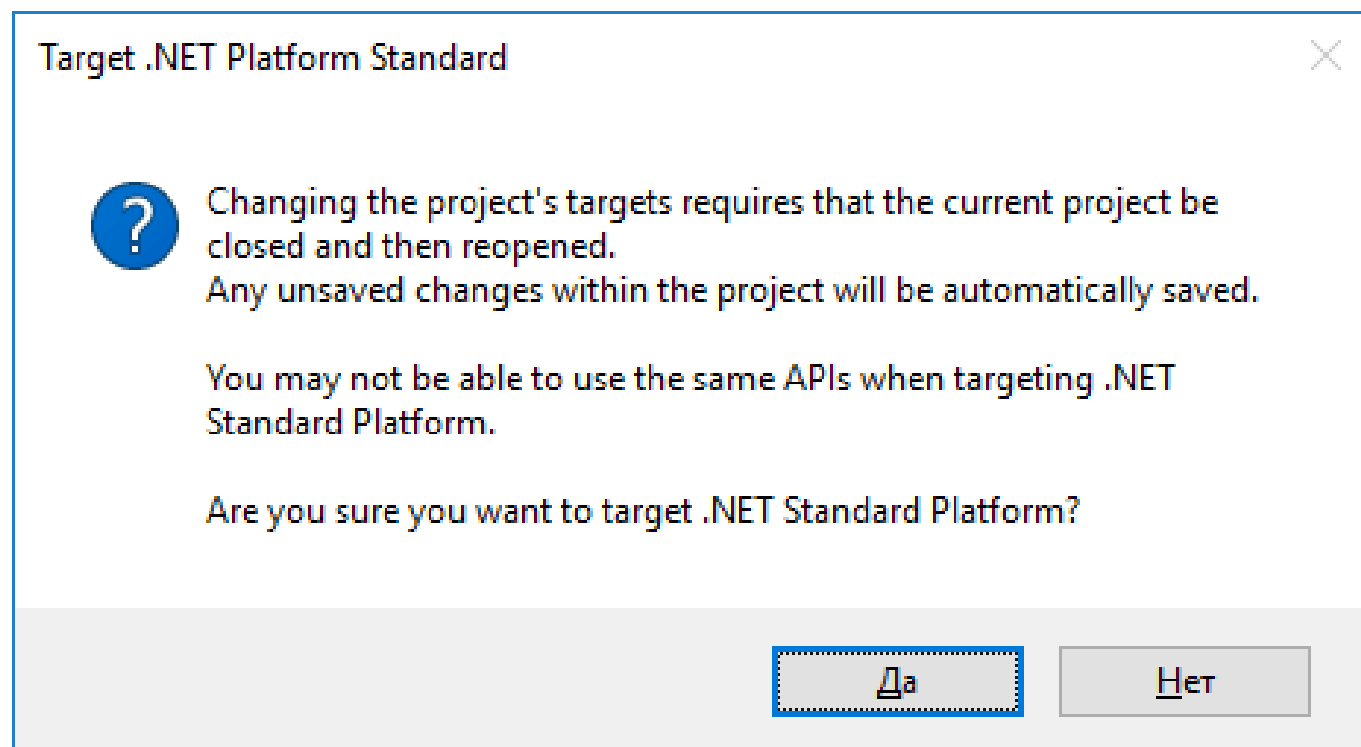
Ошибка говорит о невозможности добавления пакета System.Net.Http в проект, так как профиль проекта "Profile259", а для System.Net.Http требуется профиль "Profile111". Что делать в данном случае?

Для исправления ситуации переходим к свойствам главного проекта PCL. Здесь мы можем увидеть ряд целевых платформ, на которые по умолчанию нацелен наш проект:

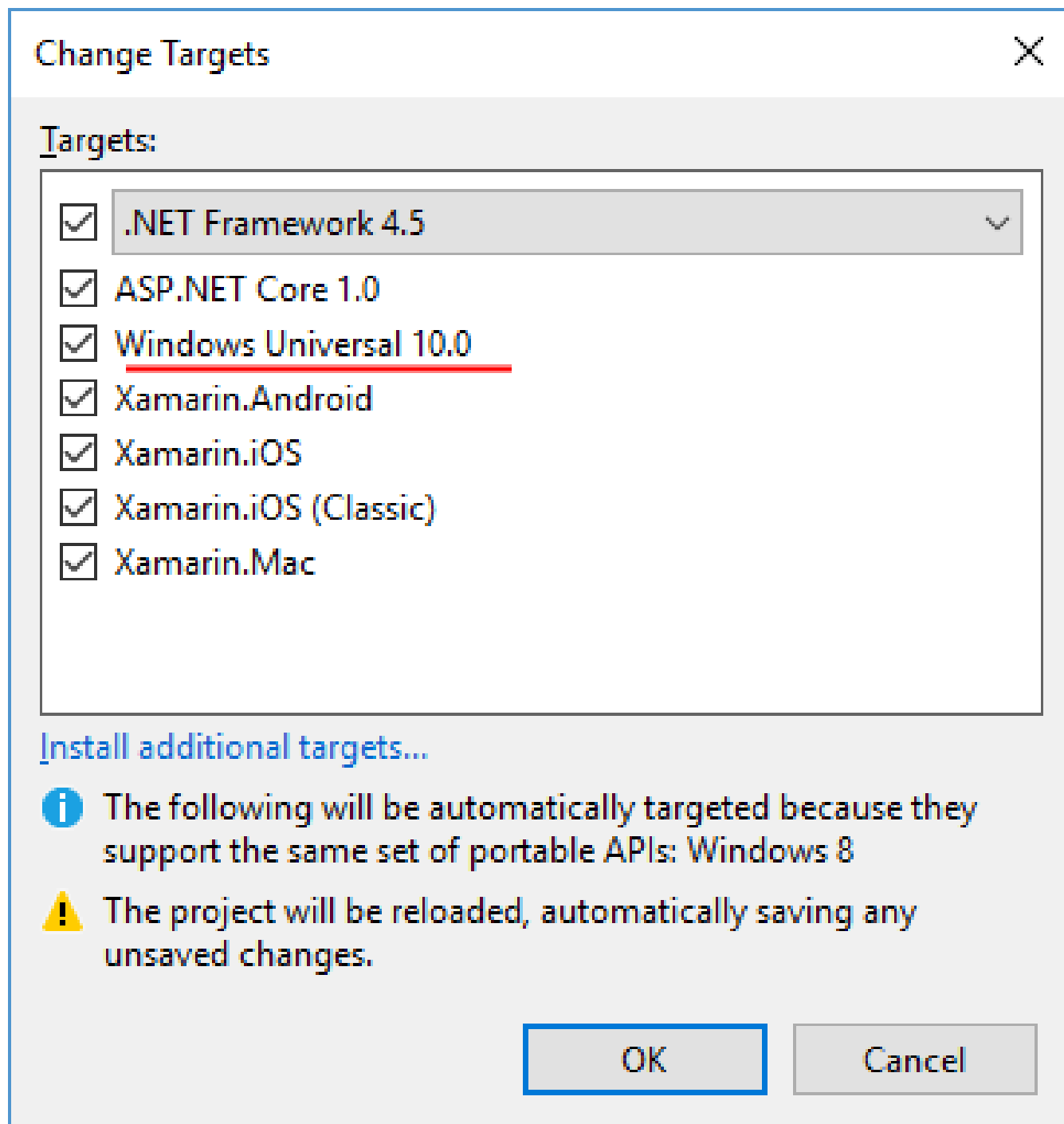


В данном случае нам надо из списка сред удалить **Windows Phone Silverlight 8**. Но перед изменением состава целевых платформ нам надо из главного проекта PCL удалить все добавленные в него NuGet-пакеты. Как правило, проект имеет только один пакет - Xamarin.Forms. Поэтому его надо удалить, а после смены целевых платформ его можно заново добавить. NuGet-пакеты нужно удалять только из главного проекта PCL.

После удаления всех NuGet-пакетов на странице целевых платформ проекта нажмем на кнопку **Change**. После этого нам отобразится окошко для подтверждения действия:



Нажмем на кнопку **Да** для подтверждения. Затем нам откроется окно с выбором целевых сред:



Укажем здесь пункт **Windows Universal 10.0** и нажмем на кнопку OK.

После этого мы сможем спокойно добавить в проект System.Net.Http, ну и кроме того, надо будет добавить пакеты, которые были удалены, в частности, Xamarin.Forms.

Работа с HttpClient

HttpClient предоставляет следующие методы:

- **GetAsync()**: отправляет запрос типа Get
- **DeleteAsync()**: отправляет запрос типа Delete
- **PostAsync()**: отправляет запрос типа Post
- **PutAsync()**: отправляет запрос типа Put
- **SendAsync()**: отправляет запрос любого типа в зависимости от настроек заголовков запроса. Выше определенные методы являются частными случаями данного метода
- **GetByteArrayAsync()**: отправляет массив байтов в запросе типа Get
- **GetStreamAsync()**: отправляет поток Stream в запросе типа Get
- **GetStringAsync()**: отправляет строку в запросе типа Get

При отправке запроса для его настройки мы можем применять объект класса **HttpRequestMessage**. В частности, для конфигурации запроса можно использовать следующие свойства класса:

- **Content**: отправляемые данные

- **Headers:** заголовки http
- **Method:** метод отправки(Get/Post/Put/Delete)
- **RequestUri:** адрес сервера

Например:

```
HttpClient client = new HttpClient();
HttpRequestMessage request = new
HttpRequestMessage();
request.RequestUri = new Uri("http://somesite.com");
request.Method = HttpMethod.Get;
request.Headers.Add("Accept", "application/json");
await client.SendAsync(request);
```

После выполнения запроса мы можем получить ответ в виде объекта класса **HttpResponseMessage**. Для обработки ответа мы можем воспользоваться следующими свойствами данного класса:

- **Content:** полученные данные
- **Headers:** заголовки ответа, полученные от сервера
- **StatusCode:** статусный код ответа

Например:

```
HttpClient client = new HttpClient();
HttpRequestMessage request = new
HttpRequestMessage();
request.RequestUri = new Uri("http://somesite.com");
```

```
request.Method = HttpMethod.Get;
request.Headers.Add("Accept", "application/json");

HttpResponseMessage response = await
client.SendAsync(request);
if(response.StatusCode==HttpStatusCode.OK)
{
    HttpContent responseContent = response.Content;
    var json = await
responseContent.ReadAsStringAsync();
}
```

Если сервер отправляет нам данные в виде json, то, получив данные в виде строки с помощью метода `responseContent.ReadAsStringAsync()`, мы можем их десериализовать из json с помощью специальных инструментов или библиотек, как `Json.NET`.

Используя методы

`GetAsync()/PostAsync()/PutAsync()/DeleteAsync()`
соответственно уже не надо указывать метод запроса.
Например:

```
HttpClient client = new HttpClient();
HttpResponseMessage response = await
client.GetAsync("http://somesite.com");
if(response.StatusCode==HttpStatusCode.OK)
{
    HttpContent responseContent = response.Content;
    var json = await
responseContent.ReadAsStringAsync();
}
```


Если сервер посылает ответ в виде простой строки, массива байт или файла, то в данном случае проще использовать методы

`GetStringAsync()/GetByteArrayAsync()/GetStreamAsync()`.

Например, получим от сервера строку:

```
HttpClient client = new HttpClient();  
string response = await  
client.GetStringAsync("http://somesite.com");
```

В то же время в данном случае мы не сможем отследить статусный код ответа от сервера.

Отправка и получение данных

Данные запроса и ответа представлены в виде объекта класса **HttpContent**. Этот класс является абстрактным, и фактически мы будем работать с его производными классами:

- **StringContent**: применяется, если отправка или получение данных происходит в виде строки
- **ByteArrayContent**: применяется, если отправка или получение данных происходит в виде массива байтов
- **StreamContent**: применяется, если отправка или получение данных происходит, как правило, в виде файла, например, изображения

Если мы хотим получить данные в виде строки, массива байт или файла, то соответственно мы можем использовать один из методов `GetStringAsync()/GetByteArrayAsync()/GetStreamAsync()`, либо можно обрабатывать свойство `Content` объекта `HttpResponseMessage`.

Для отправки данных мы можем задействовать свойство `Content` объекта `HttpRequestMessage`:

```
var friend = new Friend { Id = 1, Name = "Иван Иванов" };  
// сериализация объекта с помощью Json.NET  
string json = JsonConvert.SerializeObject(friend);  
HttpContent content = new StringContent(json);  
  
HttpClient client = new HttpClient();  
HttpRequestMessage request = new  
HttpRequestMessage();  
request.RequestUri = new Uri("http://somesite.com");  
request.Method = HttpMethod.Post;  
request.Content = content;  
HttpResponseMessage response = await  
client.SendAsync(request);
```

Либо в качестве альтернативы для отправки данных можно использовать специальные методы `PostAsync()/PutAsync()`:

```
var friend = new Friend { Id = 1, Name = "Иван Иванов" };  
// сериализация объекта с помощью Json.NET  
string json = JsonConvert.SerializeObject(friend);
```

```
HttpContent content = new StringContent(json);  
  
HttpClient client = new HttpClient();  
HttpResponseMessage response = await  
client.PostAsync("http://somesite.com", content);
```

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






ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР ПВТ

КУРС **iOS** РАЗРАБОТЧИКА

Высокая востребованность
на рынке IT!



РУЧНОЕ И АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ ПО

☆ Дата старта — 12 февраля

ЗАПИСАТЬСЯ

[Комментарии](#)[Сообщество](#)[Войти](#)

 Рекомендовать Поделиться

Лучшее в начале ▼

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS 

Имя

**Богдан Рудницкий** • 10 месяцев назад

Добрый день!

Подскажите, пожалуйста, а можно ли использовать напрямую SqlClient и сразу обращаться к удаленному серверу?

9 ^ | v • Ответить • Поделиться ›

**Dmitry Vasilyuk (Just3F)** • 6 месяцев назад



Здравствуйте. Сделал по примеру, однако при попытке отправить запрос с андроид эмулятора выдает `System.Net.WebExceptionStatus.ConnectFailure` эксепшн. Такое ощущение что не дает просто обратиться к серверу с эмулятора. Возможно знате в чем может быть проблема?

^ | v • Ответить • Поделиться ›



Metanit Модератор ➔ Dmitry Vasilyuk
(Just3F) • 6 месяцев назад

убедитесь, что серверное приложение доступно из локальной сети, например, попробуйте обратиться к нему с телефона. Также попробуйте использовать вместо эмулятора реальное устройство.

^ | v • Ответить • Поделиться ›



Dmitry Vasilyuk (Just3F) ➔
Metanit • 6 месяцев назад

Я пробовал обращаться через стороннее приложение "PostMan". Как я понял проблема в том что я указываю в адресе Localhost а надо ip адрес. Через стороннее все работает хорошо. Я прочитал что к локалхост через эмулятор не хочет работать вроде. но это не точно :)

^ | v • Ответить • Поделиться ›



Metanit Модератор ➔
Dmitry Vasilyuk (Just3F)
• 6 месяцев назад

надо не через localhost

обращаться, а через
внутрисетевой адрес
192.168.x.xx:xxxx

подробнее -

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

^ | v • Ответить •

Поделиться ›



Дмитрий • год назад

Добрый день! Можно ли использовать
HttpClient для загрузки файлов примерно
так?:

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

```
var stream = await
```

```
client.GetStreamAsync(UriFile);
```

и далее копировать поток в файл или лучше
использовать WebClient через dependency
service?

^ | v • Ответить • Поделиться ›



Igor Kravchenko • год назад

Добрый день.

У меня не получается добавить

System.Net.Http в главный проект с помощью
Nuget. В Android-проект добавляется. А сюда
не могу. Неужели я должен для каждой
платформы реализовать всё отдельно?



^ | v • Ответить • Поделиться ›



Metanit Модератор ➔ Igor Kravchenko

• год назад

попробуйте обновить пакет Xamarin Forms в главном проекте.

Реализовать для каждой платформы не надо. Код пишется один раз только в главном проекте

Посмотрел, в новых проектах по умолчанию такая проблема появляется. Чтобы ее решить, надо перейти в свойства проекта, на вкладке Library нажать на кнопку Change и отключить Windows Phone Silverlight 8.1 и сохранить изменения

^ | v • Ответить • Поделиться ›



Metanit Модератор ➔ Metanit

• год назад



ПЕТІО

ваш

производитель
программируемы
х **умных розеток**

230В,

управляемых по

LAN/WiFi

ПОДРОБНЕЕ ►





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

Контакты для связи: metanit22@mail.ru

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