Http

Работа с API

.NET MAUI

Работа с API

REST API сервис предоставляет доступ к данным

Работа с API

Данные передаются в теле сообщения Http в формате XML или JSON

Работа с API

Чтобы проверить сетевое подключение в приложении .NET MAUI, используется класс **Connectivity**.

Этот класс предоставляет свойство с именем **Current.NetworkAccess** и событие с именем **ConnectivityChanged**. Эти члены можно использовать для обнаружения изменений в сети.

Работа с API

Свойство **NetworkAccess** возвращает значение из перечисления:

- **ConstrainedInternet**,
- **Internet,**
- **Local,**
- **None** и
- **Unknown**

Если свойство **NetworkAccess** возвращает значение **NetworkAccess.None**, значит, у вас нет подключения к Интернету.

Этот механизм переносим на разные платформы.

Работа с API

```
if (Connectivity.Current.NetworkAccess ==
                        NetworkAccess.None)
    {
      . . .
    }
```

Работа с API

```
Connectivity.Current.ConnectivityChanged +=
                    Connectivity_ConnectivityChanged;


void Connectivity_ConnectivityChanged(object sender,
ConnectivityChangedEventArgs e)
        {
            bool stillConnected =
e.NetworkAccess.Equals(NetworkAccess.Internet);
        }
```

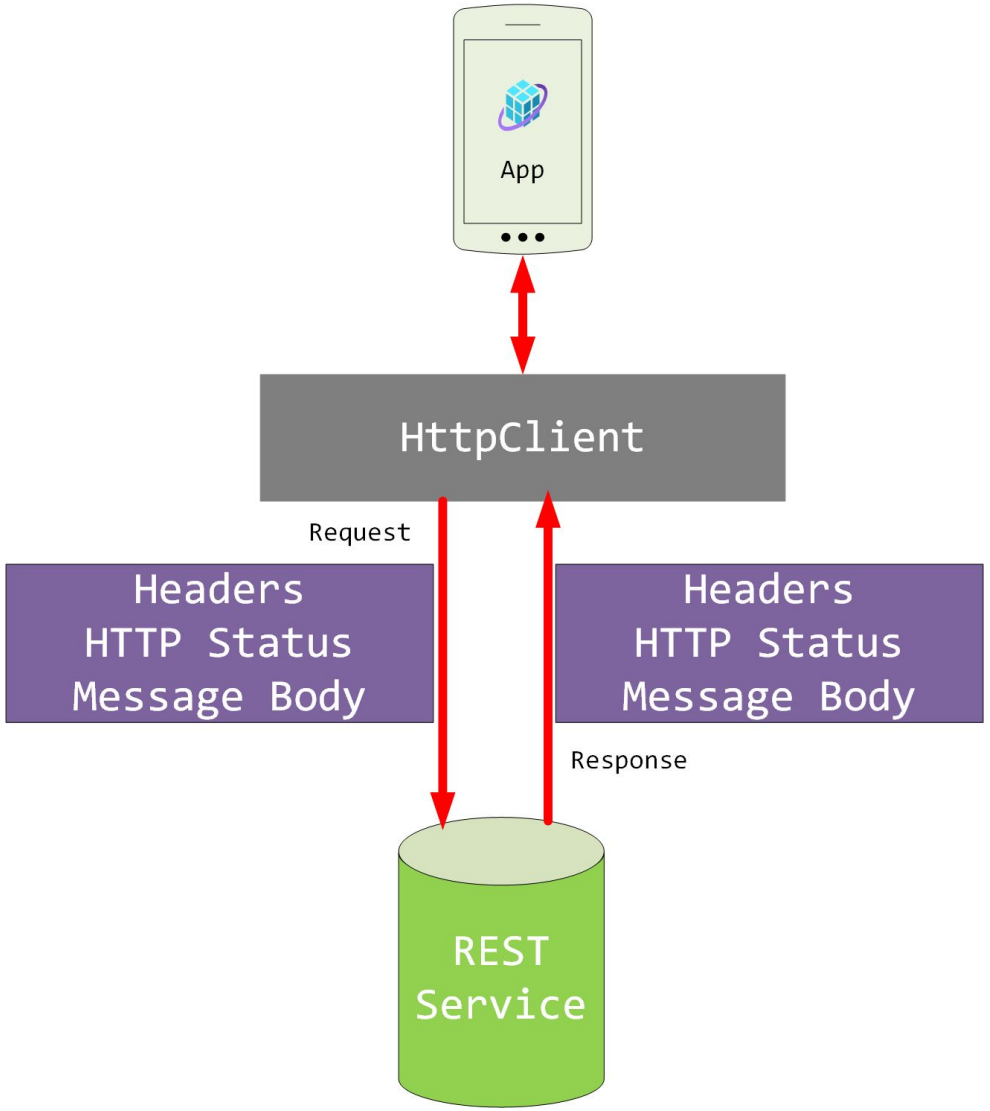
Работа с API

HttpClient — это класс .NET, который используется для отправки **HTTP-запросов** и получения **HTTP-ответов**.

Ресурсы, предоставляемые веб-службой, идентифицируются набором **URI**.

URI объединяет адрес веб-службы с именем ресурса, доступного по этому адресу.

Работа с API


App
HttpClient
Request
Headers
HTTP Status
Message Body
Headers
HTTP Status
Message Body
Response
REST
Service

Работа с API

Класс **HttpClient** доступен в пространстве имен **System.Net.Http**.

Приложение может создать объект **HttpClient** с помощью конструктора по умолчанию:

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

Работа с API

Веб-служба REST позволяет клиенту выполнять операции с данными с помощью набора HTTP-команд.

Задача HTTP-глагола — указать желаемое действие, которое необходимо выполнить с ресурсом. Наиболее распространенными являются четыре: **POST, GET, PUT** и **DELETE**.

Работа с API

- POST указывает, что вы хотите создать новый ресурс.
- GET указывает, что вы хотите получить ресурс.
- PUT указывает, что вы хотите обновить ресурс.
- DELETE указывает, что вы хотите удалить ресурс.

Работа с API (добавление объекта)

```
HttpClient client = new HttpClient();

HttpRequestMessage message =
        new HttpRequestMessage(HttpMethod.Post, url);
 message.Content = JsonContent.Create<Book>(book);

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

Работа с API (добавление объекта)

```
try
{
   string bookJson = JsonSerializer.Serialize(book, _jsonSerializerOptions);
   StringContent content = new StringContent(bookJson, Encoding.UTF8, "application/json");
   HttpResponseMessage response = await _httpClient.PostAsync($"{_url}/book", content);
   if(response.IsSuccessStatusCode)
    {….}
   else
    {...}
}
catch (Exception ex)
{…}
```

Работа с API (получение объекта)

```
var client = new HttpClient();

var message = new HttpRequestMessage(HttpMethod.Get, uri);
message.Headers.Add("Accept", "application/json");
//client.DefaultRequestHeaders.Accept
//.Add(new MediaTypeWithQualityHeaderValue("application/json"));

var response = await client.SendAsync(message);
if (!response.IsSuccessStatusCode) return null;

return await JsonSerializer.DeserializeAsync<Book>(
                        response.Content.ReadAsStream());
```

Использование HttpClientFactory

Использование HttpClientFActory

1. Установить NuGet пакет **Microsoft.Extensions.Http**

Использование HttpClientFActory

2. Зарегистрировать сервис (MauiProgram.cs):

```
var baseAddress = DeviceInfo.Platform == DevicePlatform.Android
                ? "http://10.0.2.2:5091"
                : "https://localhost:7091";

builder.Services.AddHttpClient<IRestDataService, RestDataService>(opt =>
        opt.BaseAddress = new Uri(baseAddress));
```

Использование HttpClientFActory

3. Внедрить HttpClient

```
public RestDataService(HttpClient httpClient)
{
        //_httpClient = new HttpClient();
        _httpClient = httpClient;
 }
```

Подключение к небезопасному соединению

iOS

App Transport Security (ATS) — это функция iOS, которая требует, чтобы каждое сетевое взаимодействие, осуществляемое через собственный сетевой стек HTTP, использовало TLS 1.2 или выше.

iOS

Чтобы отказаться от **App Transport Security**, добавьте новый ключ в файл **Info.plist** с именем **NSAppTransportSecurity** (файл **iOS/Platforms/ Info.plist**)

iOS

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSExceptionDomains</key>
  <dict>
    <key>dotnet.microsoft.com</key>
    <dict>
      <key>NSExceptionMinimumTLSVersion</key>
        <string>TLSv1.0</string>
      <key>NSExceptionAllowsInsecureHTTPLoads</key>
        <true/>
    </dict>
  </dict>
</dict>
```

Android

Открытый текстовый (не HTTPS) трафик отключен по умолчанию, если ваше приложение предназначено для **Android 9 (API уровня 28)** или выше.

Android

Чтобы разрешить обычный текстовый трафик, создайте новый файл **AML** в папке **Platforms/Android/Resources/xml** с именем **network_security_config.xml**

В этот файл добавьте элемент **network-security-config** с дочерним элементом **domain-config**.

Android

```
<?xml version="1.0" encoding="utf-8" ?>
<network-security-config>
<domain-config cleartextTrafficPermitted="true">
<domain includeSubdomains="true">10.0.2.2</domain>
<!-- Debug port -->
<domain includeSubdomains="true">microsoft.com</domain>
</domain-config>
</network-security-config>
```

Android

В файле **AndroidManifest.xml** добавьте

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
<application android:networkSecurityConfig="@xml/network_security_config" . . . ></application>
. . .
</manifest>
```