

Современные платформы прикладной разработки

GRPC

gRPC

ОБЩАЯ ИНФОРМАЦИЯ

Общая информация

gRPC — это не зависящая от языка высокопроизводительная платформа удаленного вызова процедур (RPC)

Общая информация

Платформа была разработана
компанией Google в 2015 году.

Общая информация

В качестве транспорта
используется HTTP/2.

В качестве языка описания
интерфейса — Protocol Buffers

Преимущества gRPC

Современная высокопроизводительная упрощенная платформа RPC.

Разработка API по модели "сначала контракт" с использованием механизма Protocol Buffers по умолчанию, что позволяет выпускать не зависящие от языка реализации.

Преимущества gRPC

Доступные для многих языков инструменты, предназначенные для создания строго типизированных серверов и клиентов.

Поддержка клиентских, серверных и двунаправленных потоковых вызовов.

Снижение уровня использования сети за счет двоичной сериализации Protobuf.



Файл PROTO

GRPC

Файл PROTO

Для разработки API в gRPC используется подход, при котором сначала создается контракт. Буферы протоколов (protobuf) по умолчанию используются в качестве языка IDL.

Файл PROTO

Файл       содержит следующие компоненты:

- ☐ Определение службы gRPC.
- ☐ Сообщения, передаваемые между клиентами и серверами.

Файл PROTO

```
syntax = "proto3";  
import "google/protobuf/wrappers.proto";  
option csharp_namespace = "GrpcServer";  
  
service Menu {  
    rpc GetDishInfo (DishInfoRequest)  
        returns (DishResponse);  
    rpc GetDishesList (DishListRequest)  
        returns (stream DishResponse);  
}
```

Файл PROTO

```
message DishInfoRequest{  
    int32 Id = 1;  
}
```

```
message DishListRequest{}
```

```
message DishResponse{  
    int32 Id=1;  
    string Name=2;  
    string Description=3;  
    int32 Calories =4;  
    string Image=5;  
    int32 CategoryId=6;  
}
```

Server gRPC

Server gRPC

- ❑ Загрузить NuGet пакет Grpc.AspNetCore
- ❑ Создать файл *.proto
- ❑ Добавить файл *.proto в группу <Protobuf>:

```
<ItemGroup>  
  <Protobuf Include="Protos\dishes.proto" GrpcServices="Server"/>  
</ItemGroup>
```

- ❑ Скомпилировать проект (будут созданы классы для описания сервера на основе файла *.proto)

Server gRPC

Реализовать класс сервиса:

```
public class MenuService : Menu.MenuBase
{
    private readonly AppDbContext _context;

    public MenuService(AppDbContext context)
    {
        _context = context;
    }

    public override async Task<DishResponse> GetDishInfo(DishInfoRequest request,
                                                         ServerCallContext context)
    {
        . . .
    }
    public override async Task GetDishesList(DishListRequest request,
                                              IServerStreamWriter<DishResponse> responseStream,
                                              ServerCallContext context)
    {
        . . .
    }
}
```

Server gRPC

```
public override async Task<DishResponse> GetDishInfo(DishInfoRequest request,
                                                    ServerCallContext context)
{
    var dish = await _context.Dishes.FindAsync(request.Id);
    var response = new DishResponse();
    if (dish != null)
    {
        response.Id = dish.Id;
        response.Name = dish.Name;
        response.Description = dish.Description;
        response.Calories = dish.Calories;
        response.CategoryId = dish.CategoryId;
    }
    return response;
}
```


Server gRPC

```
public override async Task GetDishesList(DishListRequest request,
    IServerStreamWriter<DishResponse> responseStream,
    ServerCallContext context)
{
    foreach (var dish in _context.Dishes)
    {
        var response = new DishResponse();
        response.Id = dish.Id;
        response.Name = dish.Name;
        response.Description = dish.Description;
        response.Calories = dish.Calories;
        response.CategoryId = dish.CategoryId;
        await responseStream.WriteAsync(response);
    }
}
```

Server gRPC

В классе Program:

```
builder.Services.AddGrpc();
```

```
...
```

```
app.MapGrpcService<MenuService>();
```

Server gRPC

В файле appsettings.json добавить:

```
"Kestrel": {  
  "EndpointDefaults": {  
    "Protocols": "Http2"  
  }  
}
```

Server gRPC

The screenshot displays a gRPC client interface. At the top, the text 'gRPC' is followed by the endpoint 'loc: /Menu/GetDishInfo'. To the right of this are icons for refreshing and saving, a 'Send' button, and a status indicator '0 OK'. Below the endpoint, there are two tabs: 'Unary' (selected) and 'Headers'. The main area is divided into 'Body' and 'Response 1'. In the 'Body' section, a single request is shown: `1 {"Id": "3"}`. In the 'Response 1' section, the corresponding JSON response is displayed with line numbers: `1 {`, `2 "Id": 3,`, `3 "Name": "Котлета пожарская",`, `4 "Description": "Хлеб - 80%, Морковь - 20%",`, `5 "Calories": 635,`, `6 "Image": "",`, `7 "CategoryId": 4`, `8 }`.

Server gRPC

The screenshot shows the gRPC client interface. At the top, the URL bar displays 'gRPC lo /Menu/GetDishesList'. To the right of the URL bar are icons for refresh, save, and a 'Start' button. Further right is a green button labeled '0 OK'. Below the URL bar, there are two tabs: 'Server Streaming' and 'Headers'. The 'Server Streaming' tab is active, and it shows a 'Body' section with a single entry '1 {}'. To the right of the tabs, there is a red-bordered box containing five tabs labeled 'Response 1', 'Response 2', 'Response 3', 'Response 4', and 'Response 5'. The 'Response 1' tab is selected, and it displays a JSON object with the following fields: 'Id': 3, 'Name': 'Котлета пожарская', 'Description': 'Хлеб - 80%, Морковь - 20%', 'Calories': 635, 'Image': '', and 'CategoryId': 4.

```
gRPC lo /Menu/GetDishesList Start 0 OK
```

Server Streaming Headers

Body

```
1 {}
```

Response 1 Response 2 Response 3 Response 4 Response 5

```
1 {
2   "Id": 3,
3   "Name": "Котлета пожарская",
4   "Description": "Хлеб - 80%, Морковь - 20%",
5   "Calories": 635,
6   "Image": "",
7   "CategoryId": 4
8 }
```

Клиент gRPC

Клиент gRPC

☐ Добавить NuGet пакеты:

- Grpc.Net.Client
- Google.Protobuf
- Grpc.Tools

☐ Скопировать файл *.proto из серверного приложения

☐ Добавить в проект ссылку на файл *.proto:

```
<ItemGroup>
```

```
  <Protobuf Include="Protos\Dishes.proto" GrpcService="Client" />
```

```
</ItemGroup>
```

☐ Скомпилировать проект

Клиент gRPC

```
Console.OutputEncoding = System.Text.Encoding.UTF8;
var channel = GrpcChannel.ForAddress("http://localhost:5085");
var client = new Menu.MenuClient(channel);

var response = client.GetDishesList(new DishListRequest());

while(await response.ResponseStream.MoveNext())
{
    var data = response.ResponseStream.Current;
    Console.WriteLine($"{data.Name}: {data.Description} -
{data.Calories} calories");
}

Console.ReadLine();
```


Клиент gRPC

```
Суп-харчо: Очень острый, невкусный - 200 calories  
Борщ: Много сала, без сметаны - 330 calories  
Котлета пожарская: Хлеб - 80%, Морковь - 20% - 635 calories  
Макароны по-флотски: С охотничьей колбаской - 524 calories  
Компот: Быстро растворимый, 2 литра - 180 calories  
new dish: new description 1 - 222 calories
```

Взаимодействие с сервером из браузера

GRPC

Службу gRPC нельзя вызвать напрямую из браузера. gRPC использует функции HTTP/2, и ни один браузер не предоставляет необходимый уровень контроля над веб-запросами для поддержки клиента gRPC

gRPC в ASP.NET Core предлагает
два решения, совместимые с
браузером: gRPC-Web и
перекодирование gRPC JSON.

gRPC-Web

gRPC-Web позволяет браузерным приложениям вызывать службы gRPC с помощью клиента gRPC-Web и Protobuf.

Это похоже на обычный gRPC, но используется немного другой протокол, что обеспечивает совместимость с HTTP/1.1 и браузерами.

Подробнее – см.

<https://learn.microsoft.com/ru-ru/aspnet/core/grpc/grpcweb?view=aspnetcore-7.0>

Перекодирование gRPC JSON

Перекодирование gRPC JSON позволяет браузерным приложениям вызывать службы gRPC, как обычные RESTful API с поддержкой формата JSON.

Для перекодирования gRPC JSON требуется .NET 7 или более поздней версии.

Подробнее – см.

<https://learn.microsoft.com/ru-ru/aspnet/core/grpc/json-transcoding?view=aspnetcore-7.0>

Применение gRPC

gRPC

Применение gRPC

Функция	gRPC	API HTTP с JSON
Контракт	Обязательное значение (.proto)	Необязательно (OpenAPI)
Протокол	HTTP/2	HTTP
Payload	Protobuf (малый, двоичный)	JSON (большой, удобочитаемый)
Обязательность	Строгая спецификация	Нестрого. Допустимы все HTTP-протоколы.
Потоковые операторы	Клиент, сервер, оба направления	Клиент, сервер
Поддержка браузеров	Нет (требуется grpc-web)	Да
Безопасность	Транспорт (TLS)	Транспорт (TLS)
Создание кода клиента	Да	OpenAPI + сторонние средства

<https://learn.microsoft.com/ru-ru/aspnet/core/grpc/comparison?view=aspnetcore-7.0#high-level-comparison>

Применение gRPC

- ☐ Микросервисы
- ☐ Взаимодействие между точками в реальном времени
- ☐ Среды с разными языками
- ☐ Среды с ограниченными ресурсами сети
- ☐ Межпроцессное взаимодействие (IPC)

<https://learn.microsoft.com/ru-ru/aspnet/core/grpc/comparison?view=aspnetcore-7.0#grpc-recommended-scenarios>

Недостатки gRPC

Ограниченная поддержка веб-браузера

Сейчас невозможно напрямую вызвать службу gRPC из браузера. gRPC активно использует функции HTTP/2, и ни один браузер не предоставляет необходимый уровень контроля над веб-запросами для поддержки клиента gRPC.

Недостатки gRPC

Недоступно для чтения человеком

По умолчанию сообщения gRPC кодируются с помощью Protobuf. Хотя Protobuf является эффективным методом отправки и получения, его двоичный формат не читается человеком. Для Protobuf требуется описание интерфейса сообщения, указанное в файле **.proto** для правильной десериализации. Для анализа полезных данных Protobuf на канале передачи и создания запросов вручную требуются дополнительные средства.