عنوان مقاله: gRPC چیست؟ گامی فراتر از REST در ASP.NET Core

نویسنده مقاله: تیم فنی نیک‌آموز

تاریخ انتشار: ۱ اسفند ۱۴۰۲

منبع: https://nikamooz.com/what-is-grpc/

gRPC چیست ؟ در نگاه اول، این سؤال می‌تواند برای برخی افراد با ابهام همراه باشد. در جهان سیستم‌های توزیع‌شده، اپلیکیشن‌ها ترکیبی از سرویس‌های مستقل و خودکفا هستند و لازم است این سرویس‌ها بتوانند به‌طور کارآمد با یکدیگر تعامل داشته باشند تا کارایی و مقیاس‌پذیری اپلیکیشن حاصل شود. در این مقاله، ابتدا به این پرسش پاسخ دهیم که gRPC چیست و چگونه می‌توان آن را در ASP.NET Core استفاده کرد و در ادامه، به بررسی تفاوت آن با REST API می‌پردازیم. پیش از مطالعه این مطلب، پیشنهاد می‌شود **مقاله جامع آموزش میکروسرویس** را نیز مطالعه کنید تا با دلیل اهمیت سرویس‌ها و ساختار مستقل آن‌ها آشنا شوید.

gRPC چیست ؟

gRPC یک فریمورک متن‌باز و کارآمد است که قابلیت فراخوانی پروسیجر از راه دور (RPC) را بین اپلیکیشن‌های درحال اجرا روی سیستم‌های مختلف فراهم می‌کند. به بیان ساده، به کمک gRPC ، اپلیکیشن‌ها می‌توانند به گونه‌ای باهم ارتباط برقرار کنند که انگار در یک پردازش یکسان اجرا می‌شوند. این موضوع، در مواقعی که اپلیکیشن‌ها به‌صورت فیزیکی رو سیستم‌های مختلفی قرار داشته باشند نیز صادق است. شایان ذکر است که فریمورک gRPC از Google Remote Procedure Calls برگرفته شده و توسط شرکت گوگل توسعه داده شده است.

ویژگی های gRPC چیست ؟

بارزترین ویژگی های gRPC به شرح زیر است:

- **کارایی بالا:** استفاده از HTTP/2 به منظور انتقال کارآمد داده‌ها و همچنین پروتکل بافر (Protocol Buffers) برای سریال‌سازی Compact Data، باعث می‌شود این رویکرد در مقایسه با پروتکل‌های سنتی RPC (مانند REST)، سریع‌تر عمل کند.
- **پشتیبانی از Streaming:** در فریمورک gRPC ، قابلیت استفاده از الگوی ارتباطی Unary و الگوی ارتباطی Streaming وجود دارد و می‌توان آن‌ها را برای اپلیکیشن‌های گوناگون به کار برد.
- **کارآمد:** به‌واسطه استفاده از Protocol Buffers، نمایش داده‌ها به‌صورت کارآمدتری انجام می‌شود و به‌دنبال آن، میزان استفاده از پهنای باند (Bandwidth) کاهش می‌یابد.
- **عدم سوگیری زبانی (Language Neutrality):** به‌دلیل پشتیبانی از طیف وسیعی از زبان‌های برنامه‌نویسی، این فریمورک به Langauge Neutrality مشهور است. این موضوع، ارتباط میان سرویس‌های مختلف را تسهیل می‌دهد.
- **امنیت:** gRPC با مکانیزم‌های امنیتی خاصی مانند TLS یکپارچه‌سازی می‌شود تا بدین روش، ارتباط میان سرویس‌ها امن باشند.

صرف نظر از اینکه gRPC چیست و چه ویژگی‌هایی دارد، ممکن است برایتان جالب باشد که بدانید این فریمورک از کدام یک از انواع زبان‌های برنامه نویسی پشتیبانی می‌کند. در بخش بعدی، به این پرسش می‌پردازیم.

فریمورک gRPC از چه زبان هایی پشتیبانی می کنند؟

در زمان نوشتار این مقاله، gRPC از زبان‌های برنامه‌نویسی مختلفی ازجمله Node.js ، Dart، Objective-C، سی شارپ (C#)، سی (C)، سی پلاس پلاس (C++)، جاوا (Java)، پایتون (Python) و روبی (Ruby) پشتیبانی می‌کند. پشتیبانی قدرتمند از مشهورترین زبان‌ها، برقراری ارتباط میان سرویس‌های ساخته‌شده توسط آن‌ها را ساده‌سازی می‌کند. در شرایطی که gRPC از زبان برنامه‌نویسی موردنظرتان پشتیبانی نمی‌کند، می‌توانید از کتابخانه‌های شخص ثالث (Third-Party Libraries) یا پیاده‌سازی راه‌حل‌های سفارشی به‌عنوان رویکرد جایگزین بهره‌مند شوید.

ساختار gRPC چگونه است؟

gRPC دارای یک ساختار مشخص است و به کمک آن، تعامل میان اپلیکیشن‌ها تسهیل داده می‌شود. سؤال مهم این است که منظور از مهم‌ترین کامپونت‌های gRPC چیست ؟ موارد زیر، به‌عنوان اجزای کلیدی این فریمورک به حساب می‌آیند:

۱- Protocol Buffers

Protobuf یا همان Protocol Buffers، به‌عنوان بخش بنیادی برای ارتباط gRPC عمل می‌کند و به کمک آن، سرویس‌ها و ساختمان‌های داده (Data Structures) در قالب Languages Agnostic تعریف می‌شوند. به این ترتیب، تضمین می‌شود که این موارد برروی انواع زبان‌های برنامه‌نویسی سازگار باشند. علاوه‌بر این، هر زبان حاوی کامپایلر یا Generator خاص خود است که به‌واسطه آن، تعاریف Protobuf به کدهای مخصوص زبان برنامه‌نویسی مذکور ترجمه می‌شوند. این کد تولیدشده به اموری مانند سریال سازی و Deserialization داده‌ها رسیدگی کرده و تبادل سریع داده‌ها بین سرویس‌ها را امکان‌پذیر خواهد کرد.

۲- سرویس ها

اهمیت سرویس ها در gRPC چیست ؟ در پاسخ به این سؤال می‌توان گفت که در این کامپوننت، تعریف‌های RPC و پیاده‌سازی‌های سرور حائز اهمیت هستند. درحقیقت، سرویس‌ها متدهایی را تعریف می‌کنند که مشابه زبان‌های سنتی، امکان فراخوانی (Invoke) آن‌ها به‌صورت از راه دور فراهم باشد. هر متد دارای یک نام، پارامترهای ورودی و نوع خروجی است که تمامی‌شان در تعریف Protobuf تعیین شده‌اند. سرور، متدهای تعریف‌شده سرویس را برای رسیدگی به درخواست‌های دریافتی و تولید پاسخ پیاده‌سازی می‌کند.

۳- کلاینت ها (Clients)

کلاینت‌ها از Stub های تولیدشده برای ارائه متدهایی استفاده می‌کنند که با متدهای سرویس از راه دور مرتبط هستند. به کمک Stub ها، جزئیات ارتباطات، مانند جمع‌آوری داده‌ها در قالب Protobuf، ارسال درخواست‌ها روی شبکه و دریافت پاسخ، بررسی می‌شوند. علاوه‌بر این، Client ها Invoke کردن متدها را روی Stub انجام می‌دهد؛ به این طریق، پارامترهای ورودی مهم ارائه می‌شوند. Stub وظیفه ارسال درخواست به سرور و دریافت پاسخ را برعهده دارد و یک رابط آشنا به منطق اپلیکیشن نشان می‌دهد. به‌طور کلی، می‌توان این کامپوننت را به دو مرحله مهم، یعنی Stub Generation و Method Invocation دسته‌بندی کرد.

۴- پروتکل های ارتباطی

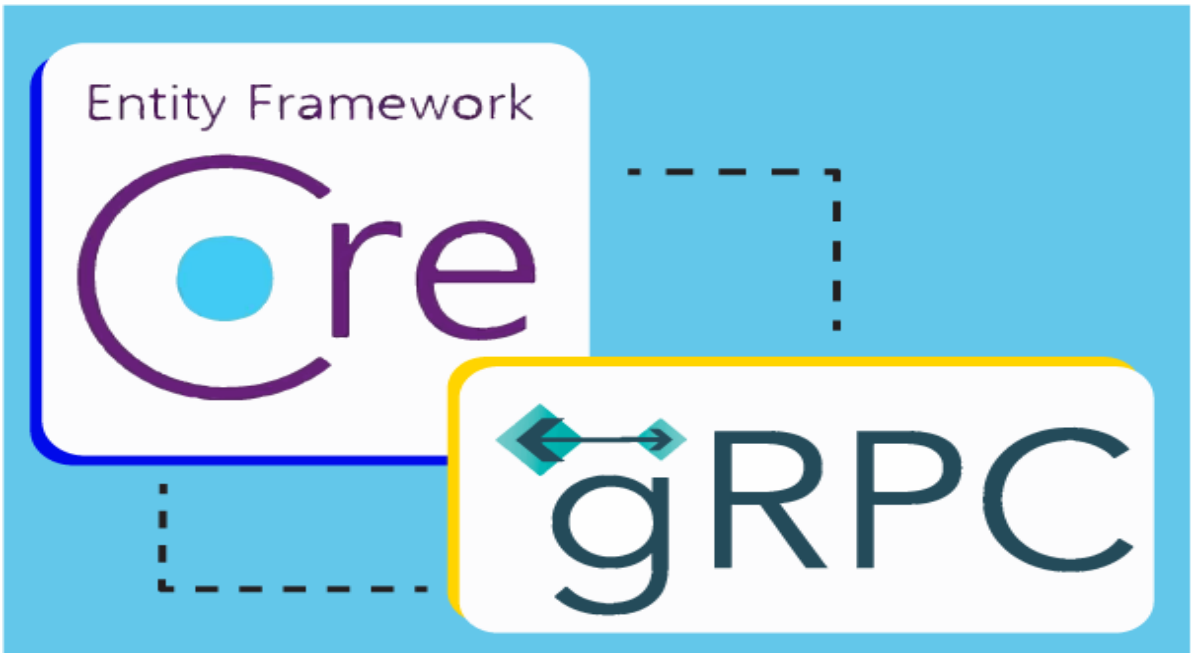
پیش‌تر در قسمت «ویژگی های gRPC چیست ؟» اشاره کردیم که این فریمورک از HTTP/2 برای انتقال کارآمد داده‌ها استفاده می‌کند. پروتکل HTTP/2 مزیت‌های گوناگونی شامل فشرده‌سازی Header، انتقال چندتایی پیغام‌ها (Multiplexing) و Server Push را دارا است. به همین دلیل، در مقایسه با HTTP/1.1 سرعت ارتباط بالاتر خواهد بود.

۵- امنیت

می‌توان gRPC را با مکانیزم‌های امنیتی خاصی مانند TLS یکپارچه‌سازی کرد تا خیالتان از بابت امن بودن ارتباط میان سرویس‌ها راحت باشد. این امر از محرمانه‌بودن و یکپارچگی داده‌ها در حین انتقال محافظت می‌کند.

شروع کار با gRPC در ASP.NET Core

شما می‌توانید gRPC را در اپلیکیشن‌های ASP.NET Core استفاده کنید و از مزیت‌های ساخت میکروسرویس‌های مقیاس‌پذیر و کارآمد بهره‌مند شوید. مراحل زیر به شما کمک می‌کنند تا بتوانید در مسیر استفاده از gRPC قرار بگیرید.



۱- نصب پکیج های NuGet ضروری

- **Grpc.AspNetCore:** این پکیج، کارکردهای اصلی برای یکپارچه‌سازی ASP.NET Core، شامل Middleware برای رسیدگی به درخواست‌ها و پاسخ‌های gRPC را فراهم می‌کند.
- **Grpc.Core:** این Package، پایه‌های لازم برای ارتباط gRBC ازجمله تعاریف سرویس، پیاده‌سازی‌های سرور و کلاینت و همچنین، پیام‌های Serialization/Deserialization را ارائه می‌کند.
- **Grpc.Tools:** با نصب این پکیج، تولید کد از فایل‌های proto. تسهیل می‌یابد؛ به طوری که ساخت کدهای سرور و کلاینت روی تعاریف سرویس شما، خودکارسازی خواهد شد.

۲- تعریف قراردادهای سرویس

- **فایل‌های proto.:** این فایل‌های متنی، ساختار داده‌ها و متدهایی را تعریف می‌کنند که توسط سرویس gRPC نمایش داده می‌شوند. درواقع، آن‌ها همانند قرارداد میان سرور و کلاینت هستند.
- **ساختمان داده‌ها:** ازطریق کلیدواژه‌هایی مانند message، field و نوع‌های داده مانند int32 ، string و bool، نوع پیام‌ها را تعریف کنید. این پیام‌ها به‌منظور نمایش داده‌های تبادل‌شده میان کلاینت و سرور استفاده می‌شوند.
- **متدهای سرویس:** با استفاده از کلیدواژه rpc، متدهای سرویس را تعریف کنید تا نوع پیغام‌های درخواست و پاسخ تعیین شوند.

۳- تولید کد کلاینت و سرور

- **تولید کد:** ابزارهای Grpc.Tools را به‌همراه فایل‌های proto. به‌عنوان ورودی اجرا کنید. این موضوع، کد کلاینت و سرور را براساس ترجیح شما تولید می‌کند.
- **کد سرور:** معمولاً کد سرور تولیدشده، یک کلاس سرویس ارث‌بری‌شده از Grpc.ServiceBase و پیاده‌سازی‌های متدهای سرویس تعریف‌شده را شامل می‌شود.
- **کد کلاینت:** کد کلاینت تولیدشده، متدهای لازم برای Invoke متدها روی سرویس gRPC ریموت را ارائه می‌کنند.

۴- پیاده سازی منطق سمت سرور

- **ساخت کلاس سرویس gRPC:** یک کلاس جدید ایجاد کنید که از Grpc.ServiceBase به ارث برسد. این کلاس، جایی برای پیاده‌سازی‌های متدهای سرویس شما خواهد بود.
- **پیاده‌سازی متدهای سرویس:** منطق هر متد سرویس تعریف‌شده در فایل‌های Proto. را پیاده‌سازی کنید.

۵- پیکربندی و هاستینگ gRPC Service

- **هاستینگ ASP.NET Core:** از مکانیزم‌های ASP.NET Core برای هاست کردن سرویس gRPC به‌همراه وب‌اپلیکیشن یا یک سرویس مجزا استفاده کنید.
- **پیکربندی Middleware:** با راه‌اندازی Middleware، پشتیبانی از gRPC-web را ممکن کنید. بدین طریق، ارتباط در مرورگر و ازطریق Websocket ها مجاز خواهد شد.

۶ -توسعه اپلیکیشن های کلاینت

- **استفاده از کد کلاینت تولیدشده:** با استفاده Client Code در اپلیکیشن‌های کلاینت، می‌توانید متدها را روی سرویس gRPC از راه دور Invoke کنید.
- **پیاده‌سازی کلاینت:** معمولاً کد کلاینت مواردی همچون ساخت نمونه کلاینت، پیام‌های درخواست، فراخوانی متدهای سرویس و رسیدگی به پیام‌های پاسخ را دربرمی‌گیرد.
- **انعطاف زبان:** می‌توان کلاینت را با استفاده از زبان‌های مختلف و براساس کد تولیدشده نوشت. این ویژگی، استفاده مجدد از کد در پلتفرم‌های مختلف را ترویج می‌دهد.

با دنبال کردن موارد فوق و در نظر داشتن معیار جانبی مؤثر در این مسیر، امکان یکپارچه‌سازی gRPC در ASP.NET Core برای شما فراهم خواهد شد.

آیا gRPC جای REST API را می گیرد؟

gRPC در سناریوهای خاص مزیت‌های قابل توجهی از خود به نمایش گذاشته است؛ با این وجود، بعید است که gRPC جایگزین همه‌منظوره‌ای برای Rest API محسوب شود. هر دوی آن‌ها جایگاه مخصوص به خود را در دنیای API دارا هستند. در عمل، gRPC در سناریوهای بلادرنگ و پرفورمنس‌محور تجلی می‌کند و REST API برای مواقعی مناسب است که سادگی و سازگاری گسترده موردنیاز است.

تفاوت gRPC و REST API چیست؟

gRPC مبتنی‌بر RPC است و اپلیکیشن‌ها در آن، به‌صورت مستقیم امکان فرخوانی متدها روی سرویس‌های ریموت را دارند؛ این عمل به گونه‌ای انجام می‌شود که انگار متدها توابع محلی هستند. در صورتی که در REST API، داده‌ها مانند منابعی درنظر گرفته می‌شوند که می‌توان اموری همچون دسترسی و دستکاری آن‌ها را انجام داد. در این فرآیند از متدهای استاندارد HTTP، شامل GET ، POST، PUT و DELETE استفاده می‌شود.

به‌طور کلی، gRPC و REST API، هردو نقاط ضعف و قوت خود را دارند و می‌توان آن‌ها را در سناریوهای متفاوت و برای ساخت API به کار برد. در شرایطی که کارایی و عملکرد، اولویت بالاتری نسبت به سایر معیارها داشته باشد، gRPC انتخاب مناسبی است؛ زیرا به‌واسطه وجود ویژگی‌های همچون خنثی بودن به زبان، RPC و قابلیت Streaming، می‌توان آن را برای اپلیکیشن‌های بلادرنگ و میکروسرویس (Microservice) به بهترین شکل استفاده کرد. ازسوی دیگر، REST API برای مواقعی مناسب است که سادگی و انعطاف‌پذیری بالا مدنظر است؛ به همین دلیل است که REST API برای API های مناسب کاربرد عمومی و سازگاری وسیع‌تری مطلوب است.

نحوه تست gRPC در Postman

Postman یک اپلیکیشن نرم‌افزاری مشهور است که برای توسعه، تست و مدیریت API ها به کار می‌رود. این نرم‌افزار، به‌دلیل دارابودن رابط کاربرپسند، برای توسعه‌دهندگان ابزار مناسبی محسوب می‌شود. در ادامه، گام‌های لازم برای تست gRPC در Postman لیست شده‌اند:

۱. ایجاد یک درخواست gRPC جدید
۲. تعریف پیام درخواست
۳. ارسال درخواست و مشاهده پاسخ
٤. امکان ویرایش Header های درخواست و پاسخ (انتخابی)
٥. ذخیره‌سازی مقادیر با قابلیت استفاده مجدد (مانند توکن‌های احراز هویت و URL های سرور)
٦. سازماندهی درخواست‌های gRPC در مجموعه‌ها و فولدرها (انتخابی)

توجه کنید که سه مورد پایانی از لیست فوق، موارد جوانبی هستند و افراد علاقه‌مند می‌توانند آن‌ها را به کار ببرند.

جمع بندی gRPC: چیست؟

در این مقاله به بررسی gRPC و نحوه استفاده از آن در ASP.NET Core پرداخته شد. استفاده از این فریمورک، مزیت‌هایی همچون بهبود کارایی، تعامل بلادرنگ و انعطاف‌پذیری در زبان را به‌همراه دارد. ضمن اینکه می‌توان آن را در حوزه‌های مختلف، شامل توسعه اپلیکیشن‌های موبایل و IOT، ارتباطات میکروسرویس‌ها، توسعه API و جریان داده بلادرنگ استفاده کرد. بنابراین، یادگیری کار با آن به‌عنوان یک مهارت به شما توصیه می‌شود.