Пловдивски университет „Паисий Хилендарски”

Факултет по математика и информатика

Курсова работа

По дисциплина „Фреймуърк системи за уеб програмиране“

На тема: „SignalR“

|  |  |
| --- | --- |
| Изготвили: Генчо Драганов,  Александър Йончев,  Светозар Митров  Специалност: Софтуерни  технологии и дизайн | Проверил:  / гл. ас. д-р В. Кюркчиев / |

Какво е SignalR ?

SignalR е open-source библиотека, която опростява имплементирането на комуникация в реално време между сървър и клиент в едно приложение. Библиотеката използва Remote procedure calls (RPC), които се изпращат от сървъра и предизвикват изпълнението на функции, които се намират на клиента.

SignalR има две основни версии – SignalR и SignalR Core, като съответно първата използва .NET Framework и работи единствено върху Windows операционната система, а втората - .NET Core, таргетира и други високопопулярни операционни системи, като macOS и Linux. Технологията позволява на програмистите работещи с езици, които използват .NET платформата да създават приложения използващи двупосочна комуникация на живо.

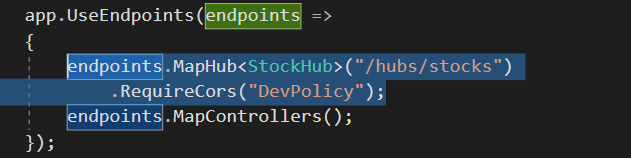
Фреймуъркът предоставя на програмистите една абстракция, която улеснява работата с установяване на връзката между клиент и сървър. SignalR използва различни похвати и технологии за постигане на двупосочна комуникация. Примери за това са WebSockets, Server-Sent Events и Long Polling. Именно в този ред е и приоритета за избор на технология осигуряваща връзката, тоест първо винаги се прави опит за осъществяване на комуникация чрез WebSockets, а като краен вариант се използва Long Polling. Библиотеката позволява на програмистите да се фокусират повече върху изграждането на желаната функционалност, като се грижи за избирането на най-ефективния метод за пренос на данни.

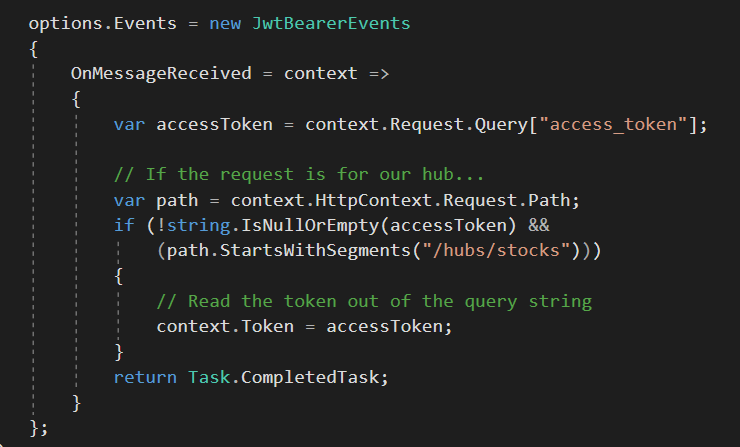
1. **Как можем да използваме SignalR ?**

SignalR разполага с две библиотеки – една за сървъра и една за клиента. Библиотеката за сървъра е добавена автоматично при създаването на нов ASP .NET Core 3.1 проект. За конфигурирането на SignalR в един проект за уеб приложения е нужно да добавим няколко команди. За успешното инжектиране в различните класове, където ще ни е нужна комуникация в реално време е нужно да добавим съответния програмен код в **ConfigureServices** метода на Startup класа.



Конфигурирането на входните точки за комуникация със сървъра се случва в **Configure** методана същия клас и е представена на снимката отдолу.



При нужда от оторизирана комуникация със сървъра, при която се използва JWT е нужно да добавим и съответния отрязък от код, който инструктира сървъра да вземе съответния тоукън от query параметрите на заявката.

Клиентската библиотека е нужно да бъде добавена ръчно към клиентското приложение. Официалната клиентска библиотека предоставена от Microsoft е [@microsoft/signalr@latest](https://www.npmjs.com/package/@microsoft/signalr).

Тя може да бъде добавена с npm, използвайки:

*npm install @microsoft/signalr*.

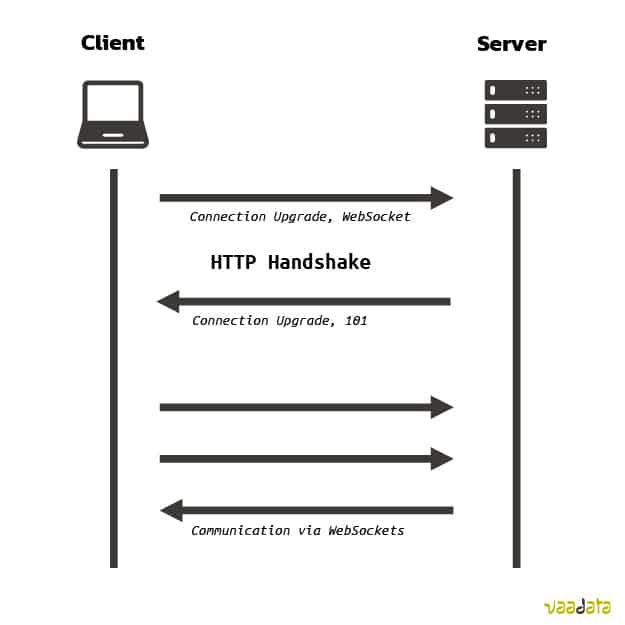
Или с LibMan използвайки:

libman install @microsoft/signalr@latest -p unpkg -d wwwroot/js/signalr --files dist/browser/signalr.js --files dist/browser/signalr.min.js

1. **Какви технологии и принципи използва SignalR ?**

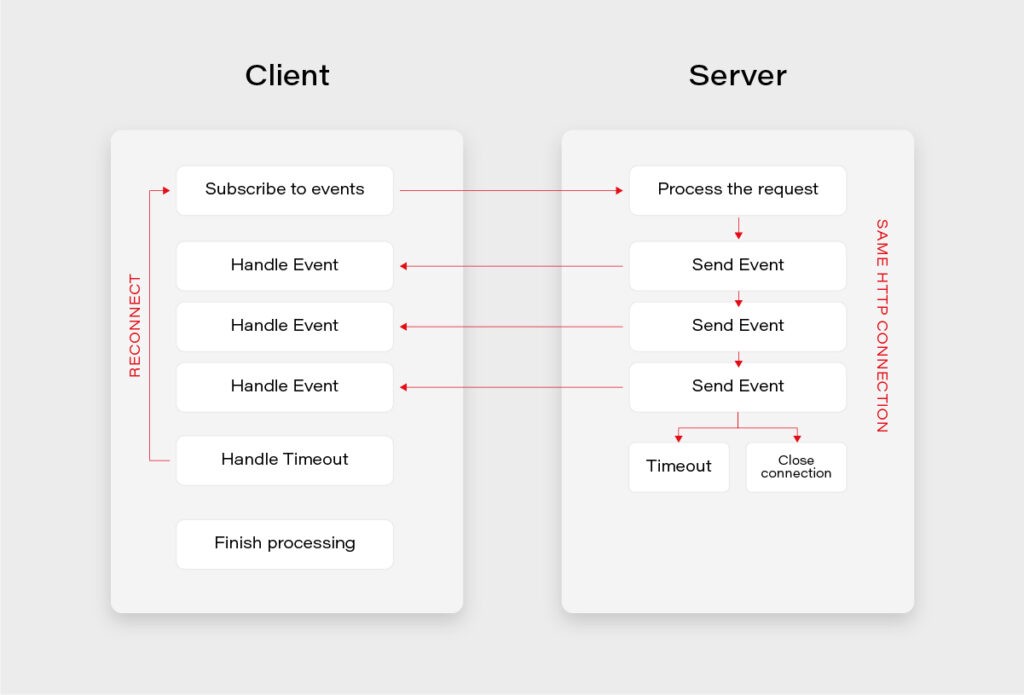
За да предостави комуникация в реално време, SignalR може да използва един от три възможни транспортни протокола за информацията:

1. **WebSocket**



За да се установи WebSocket връзка, клиентът изпраща WebSocket handshake request, което е HTTP заявка и сървърът отговаря с HTTP 101 Switching Protocols response. След това комуникацията преминава в двупосочен бинарен протокол, което позволява информация да бъде пращана както от клиента към сървъра така и обратно без установяването на допълнителни връзки.

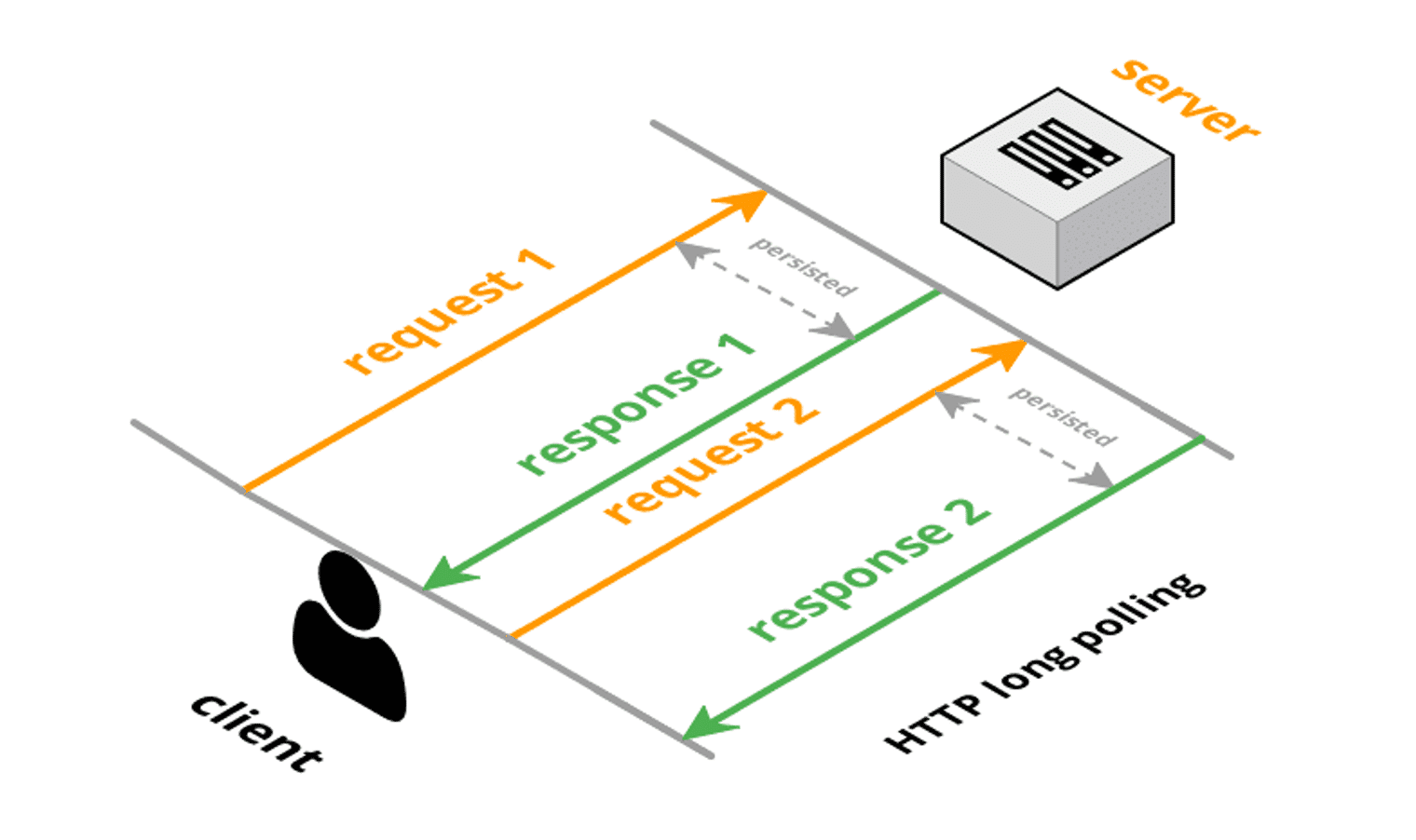
1. **Server-Sent Events**



Когато не е възможна комуникация чрез WebSocket , единия възможен вариант е чрез Server-Sent Events. При този метод на комуникация, клиента изпраща HTTP заявка към сървъра, а той отговаря чрез отваряне на връзка между клиента и сървъра наречена EventSource. Тази връзка е еднопосочна и служи единствено за изпращане на информация от сървъра към клиента.

Когато клиента трябва да изпрати информация към сървъра, той трябва да използва традиционна HTTP заявка.

1. **Long Polling**



Последния вариант, когато не е възможно да се използва нито WebSocket връзка, нито Server-Sent Events, остава Long Polling. При този метод на комуникация клиентът изпраща HTTP заявка към сървъра и изчаква отговор. При получаването на отговор или при автоматичното прекратяване на връзката (стандартното време за изчакване на отговор е 2 минути) се изпраща нова заявка, която отваря нова връзка към сървъра и отново изчаква за отговор.

1. **Какви компоненти предоставя SignalR ?**

SignalR използва т.нар. Hub , като входна точка на клиентите към приложението. Hub представлява един канал , който е на по-високо ниво на абстракция и позволява на сървъра да извиква методи на клиента и обратния вариант. Съобщенията, които се изпращат посредством един Hub могат да в два формата в зависимост от протокола, който се използва.

SignalR предоставя два протокола за съобщенията: текстов, който предоставя информацията в JSON формат и бинарен, който е базиран на MessagePack протокола. Разликата между двата е, че MessagePack създава съобщения с по-малък рамер.

1. **Каква функционалност предоставя Bull Market ?**

Целта на проекта, който сме разработили е да покаже част от възможностите на SignalR.

Той съдържа клиент, който е Angular приложение и представлява User Interface, чрез който потребителите могат да влязат или регистрират и да получават информация за определени акции в реално време, да оставят коментари за всяка акция и да разглеждат по-подробна информация за нея.

Angular клиента комуникира с ASP .NET Core 3.1 Web приложение, което се състои както от REST API за предоставяне на детайлната информация за всяка акция, така и от SignalR Hubs, които организират WebSocket връзки, чрез които те предоставят на клиентите информация в реално време за цените на акциите. Примери за двата вида комуникация са: информацията в реално време за цените на акциите и коментарите към всяка акция (WebSocket) , детайлната информация за всяка акция (HTTP).

.NET Core приложението взема информация за текущите цени на акциите посредством [Alpaca Streaming API](https://alpaca.markets/docs/api-documentation/) , като отваря WebSocket връзка с Alpaca API при стартирането на приложението и я поддържа отворена докато не бъде спряно.

Информацията за акциите, коментарите и потребителите са съхраняват в SQL база данни, като за комуникация с нея .NET Core приложението използва EF Core 3.1.