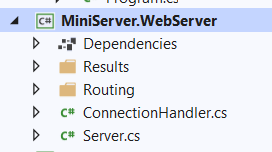
# Mini HTTP Server - Упражнение

## SIS.WebServer Архитектура

"**WebServer**" проекта ще съдържа информация за класовете, които изграждат връзка с "**TCP**". Тези класове ще комуникират с класовете от "**HTTP** **Project**". Проекта ще изнася няколко класа, които ще служат за "**външния**" свят, за да създаваме приложния.

Създайте следните папки и класове:



### Results папка

"Results" папката ще съдържа няколко класа, които са наследени от "HttpResponse**"** класа. Тези класове, ще използват за имплементираме прости приложения с "MiniServer". Трябва да създадем три класа вътре - "TextResult", "HtmlResult" и "RedirectResult".

#### TextResult

Създаден е така, че да връща текст, като отговор. Трябва да има "Content-Type" и header – "text/plain"



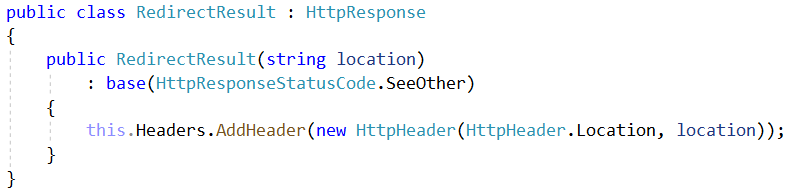
#### HtmlResult

Създаваме този клас, да връща HTML в себе си. Така чрез този клас, ние можем да върнем HTML или просто съобщение. Трябва да има "Content-Type" и header – "text/html"



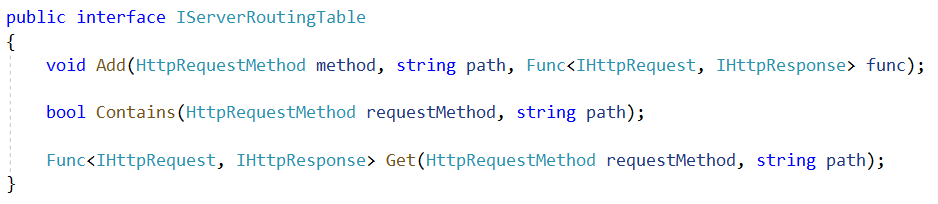
#### RedirectResult

Този клас, не трябва да има Content. Единствената задача е да бъде пренасочен. Този "**Response"** има локация. Статуса трябва да бъде – "SeeOther".

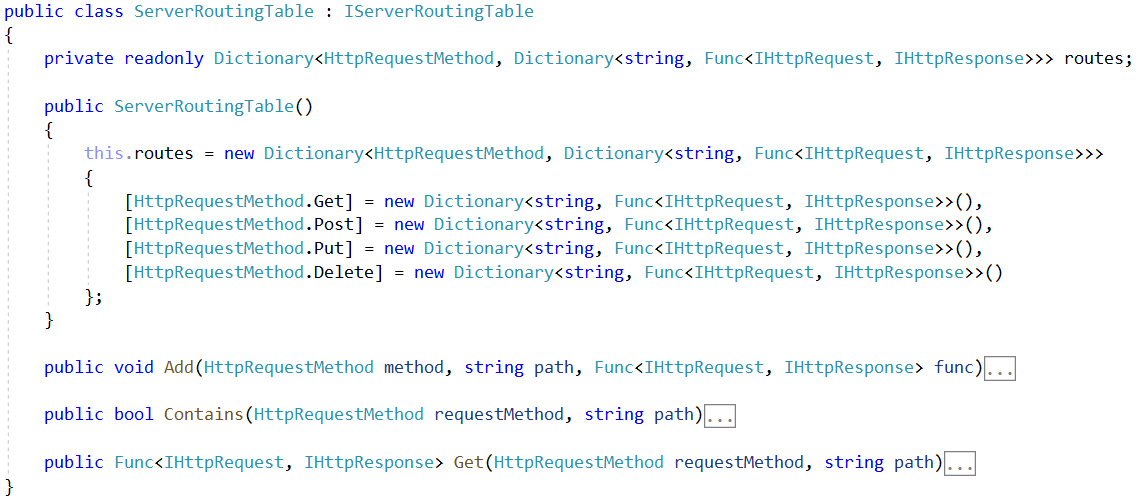


### Routing папка

В папка, ще съдържа логиката за рутиране и конфигурация на сървъра. Ще съдържа един интерфейс и един клас - "IServerRoutingTable" and "ServerRoutingTable"



Този клас съдържа големи колекции от насложени асоциативни масиви, които ще се използват за рутиране.



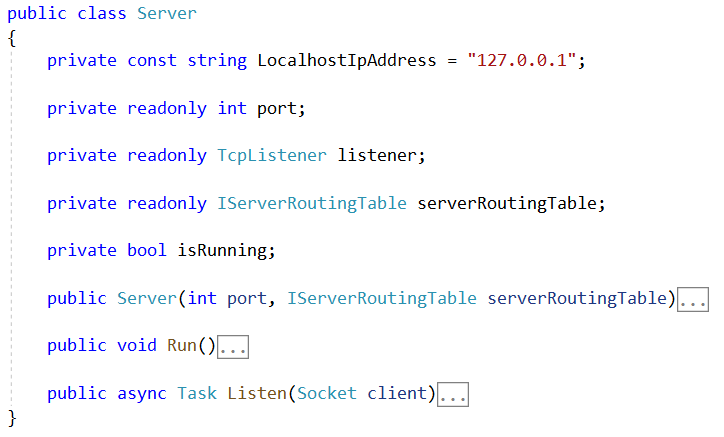
Това е главният алгоритъм за "**Request Handling**". "**Request Handler**" се конфигурира, като настройва "**Request Method**"и "**Path**"на заявката. "**Handler**"сам по себе си е функция, която приема"**Request**" параметър и генерира "**Response**"параметър.

<Method, <Path, Func>>

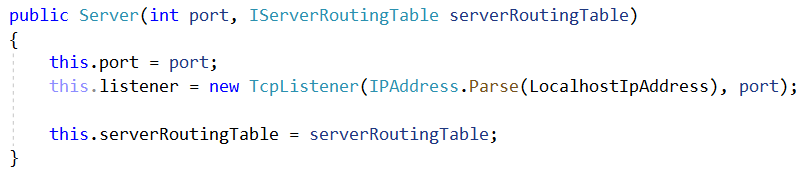
Ще видим по-надолу в примерите как работи.

### Server клас

Server класа е обвивка за **TCP connection**. Използва TcpListener , за да запише връзката с клиента и да я подаде на ConnectionHandler, която го изпълнява.



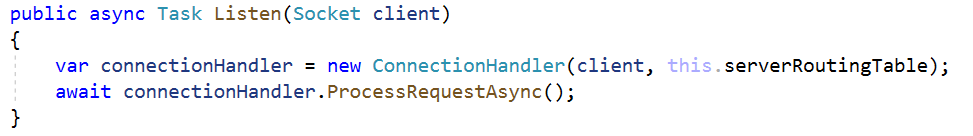
Конструкторът се използва, за да бъде инициализиран Listener и RoutingTable



Този метод се използва за процеса на слушане. Процесът трябва да бъде асинхронен, за да подсигури функционалността, когато двама клиенти изпратят заявка.



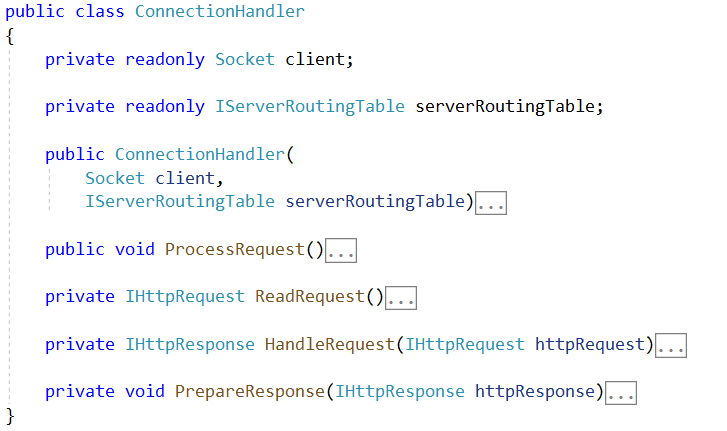
Listen() метода е главният процес при свързване с клиента.



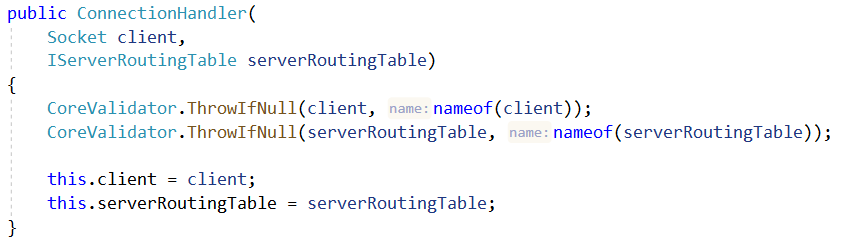
Както виждате, ние създаваме нов ConnectionHandler, за всяка нова връзка и я подаваме на ConnectionHandler, заедно с **routing table**, така че заявката да бъде изпълнена.

### ConnectionHandler клас

ConnectionHandler е клас, който произвежда връзката с клиента. Приема връзката, изважда заявката, като низ и минава процес през **routing table**, като я изпраща обратно на "Response" в байт формат, чрез **TCP link.**



Конструктора се използва, за да се инициализира **socket** и **routing table**.



ProcessRequestAsync() метода е асинхронен метод, който съдържа главната функционалност на класа. Използва и други методи да чете **заявки**, да ги **обработва** и да създава **Response**, Който да бъде върнат на клиента и най-накрая да затвори връзката.



ReadRequest() метода е асинхронен метод, който чете байт данни, от връзката с клиента, изважда низа от заявката и след това го обръща в HttpRequest обект.

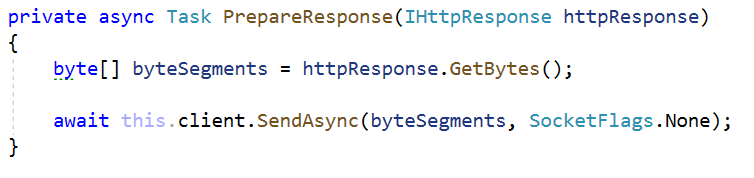


HandleRequest() метода проверява ако **routing table** има **handler** за дадената заявка, като използва **Request’s** **Method** и **Path**

* Ако няма такъв **handler** “Not Found” отговор е върнат.
* Ако има такъв **handler**, функцията е извикана и резултата е върнат.



PrepareResponse() метода изважда байт данни от отговора и ги изпраща на клиента.



Това е финалният вид на нашия ConnectionHandler и **WebServer** проект.

## Hello, World!

Създайте трети проект, който да се казва MiniServer.Demo. Той трябва да реферира и двата проекта MiniServer.HTTP и MiniServer.WebServer.

Създайте следните класове:

### HomeController клас

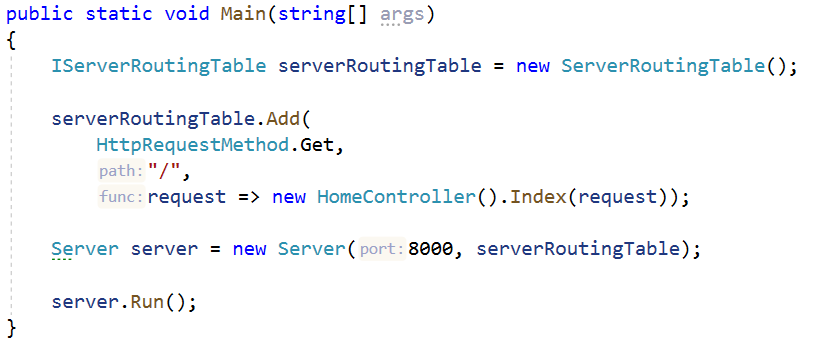
HomeController класа трябва да има един метод – Index(), който да изглежда по този начин:



### Launcher клас

Launcher класа трябва да съдържа Main метода, който инстанциира Server и го конфигурира да се справя със заявките, като използва ServerRoutingTable.

Конфигурирайте само пътя “/”, като използва ламбда функция, която извиква HomeController.Index метода.



Сега стартирайте **MiniServer.Demo** проекта и трябва да видите това на конзолата, ако всичко е наред:



Отворете браузъра и отидете на localhost:8000 и трябва да видите това:



Поздравления! Завършихте първото си приложение с **MiniServer**.