**Exercise**

ASP.NET Core – Custom middleware

Maart 2021

# Math middleware

Create an empty ASP.NET Core application “CustomMiddleWare”.

Add a new middleware component in the pipeline (right before the *app.UseRouting* component).

De middleware should react to requests to an url that matches the path “/math/*[operator]*”.

The [operator] should be a math operator like “add, multiply, faculty, …”.

When a math operator is found in the url, the arguments should be read from the body of the request. The body of the request should contain a list of one or more numbers separated by ‘,’, ‘;’ or ‘-’.

When an operator and the necessary arguments are present, the middleware should execute the math operation and write the string “Result: [result]” to the response ([result] being the result of the calculation).

If a necessary element is missing (operator, arguments) the middleware passes the request to the next component in the pipeline.

Also when there is no match in the path of the url the middleware should pass the request to the next component.

Add support for the following operators:

* /add (the number of arguments can vary from zero to infinity)
* /substract (the number of arguments can vary from one to infinity)
* /factorial (n!, there should be exactly one argument)

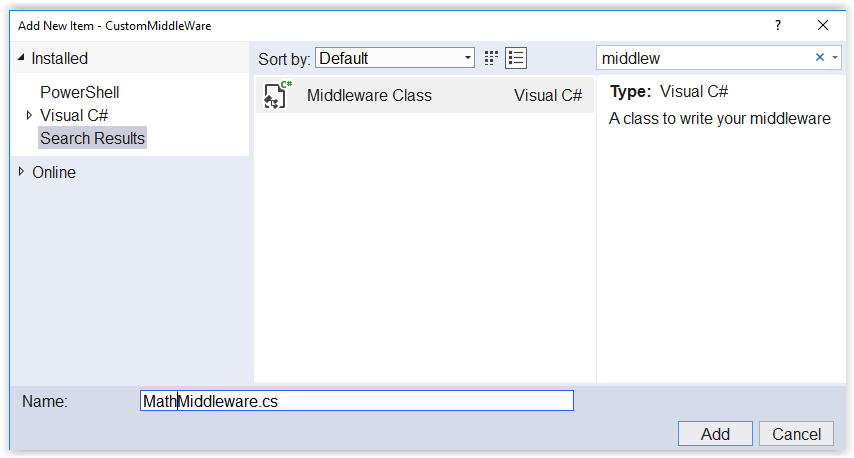
Tips:

* app.Use
* context.Request.Path.StartsWithSegments("/math", out var operatorPath)
* StreamReader
* ReadToEndAsync
* StringSplitOptions.RemoveEmptyEntries
* switch
* context.Response.WriteAsync
* Use Postman (<https://www.getpostman.com>) to send http requests with a body to your application.

# Middleware class

Refactor the code so that it becomes possible to add the middleware component by calling an extension method *app.UseMath()*:

Right click on the solution and select “Add -> New Item…”. Search for “middleware” and select “Middleware Class”.



Call the file “MathMiddleware.cs”.

Rename the “UseMathMiddleware” extension method to “UseMath”. Do a little research about extension methods if you don’t know what they are.

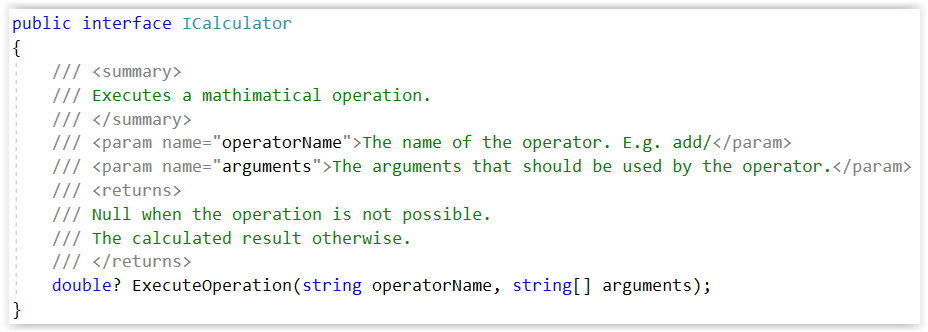
The “Invoke” method is the RequestDelegate value that will be called for each http request.

Try to move your code from StartUp.cs to this class.

In StartUp.cs you can now use the extension method.

# Dependency injection

Add an “ICalculator” interface to the application:



Refactor the middleware to use this interface instead of doing the calculation itself.

Tips:

* You can inject an ICalculator in the constructor of the middleware class