namespace tomogochi\_brain

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.AddCors(options =>

{

options.AddPolicy("CorsApi",

builder => builder.WithOrigins("http://localhost:8080", "http://rusakovam.su:8080")

.AllowAnyHeader()

.AllowAnyMethod());

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

app.UseCors("CorsApi");

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

namespace tomogochi\_brain.Controllers

{

[EnableCors("CorsApi")]

[Route("api/[controller]")]

[ApiController]

public class MessageController : ControllerBase

{

// GET api/<Message>/5

[HttpGet("{id}")]

public string Get(int id)

{

string json = "not found";

if ((id < ms.messages.Count) && (id>=0))

{

json = JsonSerializer.Serialize(ms.messages.ElementAt(id));

}

return json.ToString();

}

// POST api/<Message>

[HttpPost]

public void Post([FromBody] message msg)

{

ms.Add(msg.username,msg.text);

}

}

}

namespace tomogochi\_brain

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

webBuilder.UseUrls("http://localhost:5000");

//webBuilder.UseUrls("http://192.168.51.7:5000", "http://localhost:5000");

//"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --disable-web-security --user-data-dir="d:\temp"

});

}

}

[Serializable]

public class message

{

public string username { get; set; }

public string text { get; set; }

public DateTime timestamp { get; set; }

public message()

{

this.username = "Server";

this.text = "Server is running...";

this.timestamp = DateTime.UtcNow;

}

}

public message(string username, string text)

{

this.username = username;

this.text = text;

this.timestamp = DateTime.UtcNow;

}

}

[Serializable]

public class MessagesClass

{

public List<message> messages = new List<message>();

public void Add(message ms)

{

ms.timestamp = DateTime.UtcNow;

messages.Add(ms);

Console.WriteLine(messages.Count);

}

public void Add(string username, string text)

{

message msg = new message(username, text);

messages.Add(msg);

Console.WriteLine(messages.Count);

}

public MessagesClass()

{

messages.Clear();

message ms = new message();

//for (int i = 0; i < 10; i++)

messages.Add(ms);

}

public MessagesClass(List<message> messages)

{

messages.Clear();

message ms = new message();

//for(int i=0; i<10;i++)

messages.Add(ms);

}

public override string ToString()

{

return messages.ToString();

}

}