



RabbitMQ With ASP.NET Core - Microservice Communication With MassTransit



Amit Naik

Updated date Jun 21, 2021

35.3k

3

5

[Download Free .NET & JAVA Files API](#)

In this article, we will see Microservice Communication using RabbitMQ with ASP.NET Core. We will learn how to enable communication between Microservices using RabbitMQ and MassTransit.

We will be using MassTransit Helpers to publish/receive messages from our RabbitMQ server,

- [Download source code from GitHub](#)

What is Message Broker

Before going to the topic RabbitMQ, we will see about Message Broker. Message Broker's main responsibility is to broker messages between publisher and subscribers.

Once a message is received by a message broker from the producer, it routes the message to a subscriber. The message broker pattern is one of the most useful patterns when it comes to decoupling microservices.

- **Producer:** An application responsible for sending messages.
- **Consumer:** An application responsible for messages.
- **Queue:** Storage where messages are stored

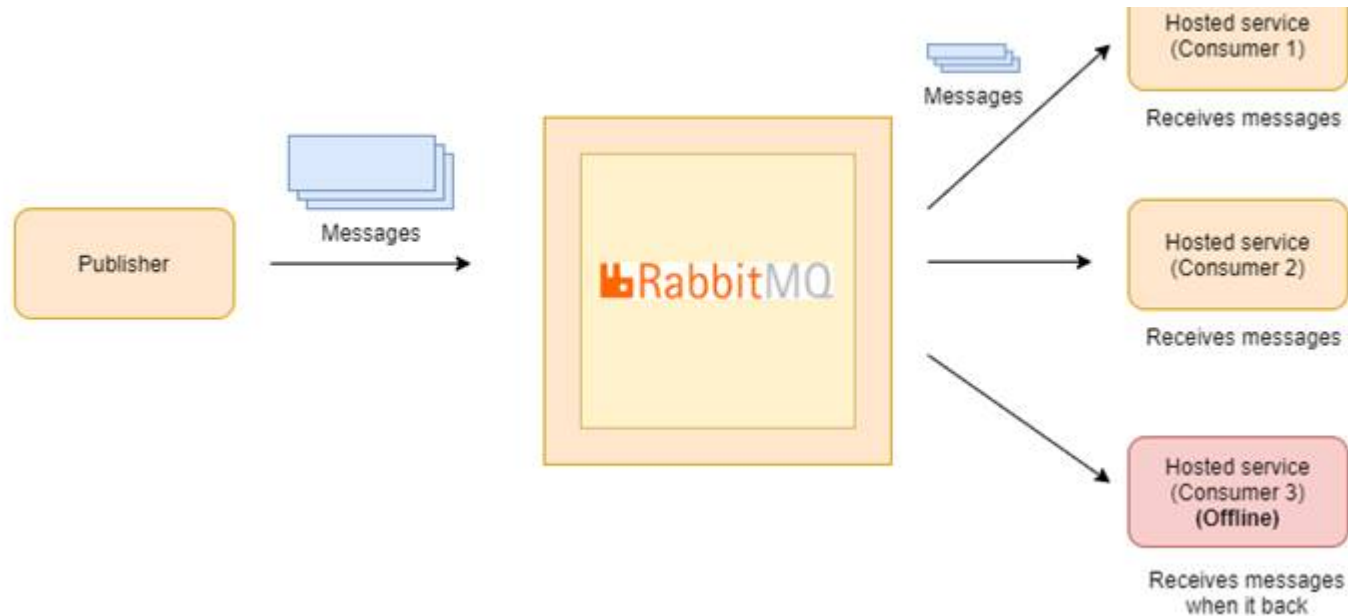
RabbitMQ is one of the most widely used open-source message Broker services. It basically gives your applications a common platform for sending and receiving messages. This ensures that our messages (data) are never lost and successfully received by consumers and it supports various messaging protocols.

Advantages Of RabbitMQ

There are some reasons why using a queue instead of directly sending data is better,

- Higher availability and better error handling
- Better scalability
- Extremely lightweight and very easy to deploy
- Share data with whoever wants/needs it
- Better user experience due to asynchronous processing

Demonstration of RabbitMQ setup



A simple demonstration of RabbitMQ setup. In case, any of the consumers is offline for some time, messages are still in RabbitMQ waiting for consumers to come online and receive messages.

Protocol supported

RabbitMQ supports multiple protocols,

- AMQP 0-9-1: RabbitMQ was originally developed to AMQP 0-9-1. AMQP 0-9-1 is a binary protocol and defines quite strong messaging semantics.
- STOMP: is a text-based messaging protocol.
- MQTT: Binary protocol focusing mainly on Publish/Subscribe scenarios.
- AMQP 1.0
- HTTP and WebSocket: While HTTP is not really a messaging protocol, RabbitMQ can transmit messages over HTTP

What is MassTransit

message bus concept.

Setting Up The Environment

For installing RabbitMQ, there are multiple approaches, I would recommend Approach 1 i.e., by installing via docker images,

Approach 1

Install ErLang and RabbitMQ in your local machine,

- ErLang
- RabbitMQ

Approach 2

Or run docker rabbitMQ with management web console.

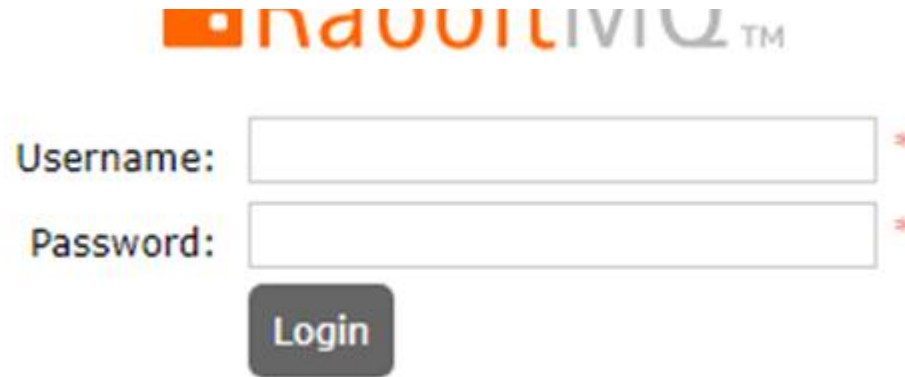
```
docker run -d --hostname my-rabbit --name some-rabbit -p 15672:15672 -p 5672:5672 rabbitmq:3-management
```

To enable RabbitMQ managementt plugin- Dashboard

To activate the RabbitMQ management dashboard, run the below command in the command prompt with the administrator.

```
cd C:\Program Files\RabbitMQ Server\rabbitmq_server-3.8.17\sbin
rabbitmq-plugins enable rabbitmq_management
net stop RabbitMQ
net start RabbitMQ
```

Now navigate to <http://localhost:15672>, where you can find the management dashboard of RabbitMQ running.

The image shows a login interface for MassTransit. At the top, the MassTransit logo is displayed in orange and grey. Below the logo, there are two input fields. The first field is labeled 'Username:' and the second is labeled 'Password:'. Both fields have a red asterisk to their right, indicating they are required. Below the password field is a dark grey button with the word 'Login' in white text.

Username: *

Password: *

Login

The default user id and password for management is guest/guest.

Getting Started

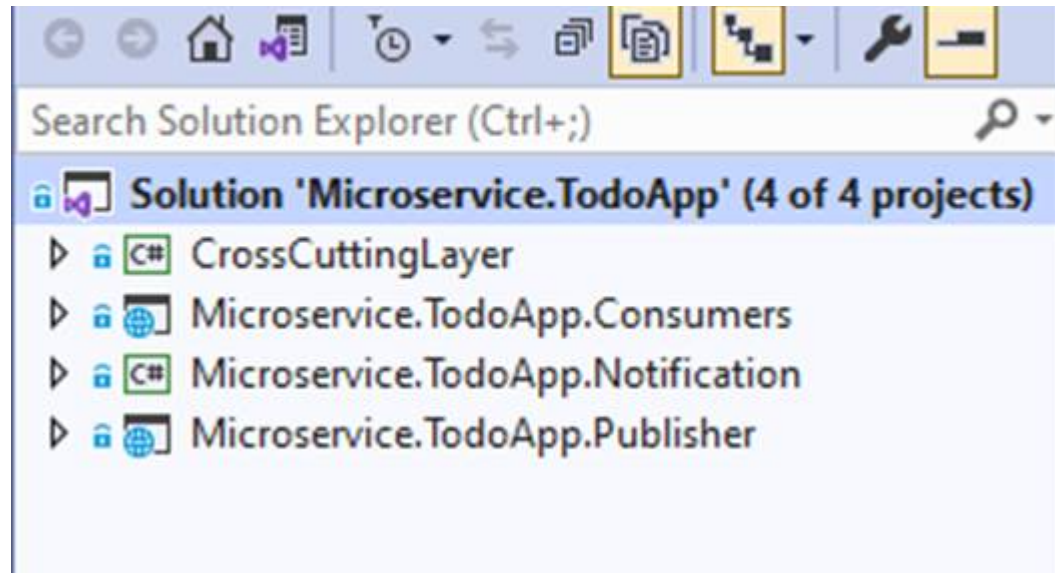
Nuget library used in the application.

Install-Package MassTransit

Install-Package MassTransit.AspNetCore

Install-Package MassTransit.RabbitMQ

Clone [Source code](#) and you can see the project structure via solution explorer. We walk through the code.



CrossCuttingLayer

This is a common class library for all the applications. Here we have Model (i.e., *Todo* class) and Constant (For RabbitMQ configuration).

Below is the code snippet for *Todo* and *Constant* class.

Todo.cs

```
1 using System;
2
3 namespace CrossCuttingLayer
4 {
5     public class Todo
6     {
7         public string Id { get; set; }
8         public DateTime CreatedTime { get; set; } = DateTime.UtcNow;
```

```
11     }  
12  
13 }
```

RabbitMqConsts.cs

```
1 public class RabbitMqConsts  
2 {  
3     public const string RabbitMqRootUri = "rabbitmq://localhost";  
4     public const string RabbitMqUri = "rabbitmq://localhost/todoQueue";  
5     public const string Username = "guest";  
6     public const string Password = "guest";  
7     public const string NotificationServiceQueue = "notification.service";  
8 }
```

Publisher class library

We will configure MassTransit in a startup.cs file in our Asp.net Core Web API application.

Startup.cs

```
1 public void ConfigureServices(IServiceCollection services)  
2 {  
3  
4     services.AddMassTransit(x =>  
5     {  
6         x.AddBus(provider => Bus.Factory.CreateUsingRabbitMq(config =>  
7         {  
8             config.Host(new Uri(RabbitMqConsts.RabbitMqRootUri), h =>  
9                 {
```

```

12         });
13     }));
14 });
15 services.AddMassTransitHostedService();
16 services.AddControllers();
17 services.AddSwaggerGen(c =>
18 {
19     c.SwaggerDoc("v1", new OpenApiInfo { Title = "Microservice.TODO.Publisher", Version
20 });
21 }

```

And in our publisher controller, we will create a post method to publish a message.

TodoController.cs

```

1  [HttpPost]
2  public async Task<IActionResult> CreateTicket(Todo todoModel)
3  {
4      if (todoModel is not null)
5      {
6          Uri uri = new Uri(RabbitMqConsts.RabbitMqUri);
7          var endPoint = await _bus.GetSendEndpoint(uri);
8          await endPoint.Send(todoModel);
9          return Ok();
10     }
11     return BadRequest();
12 }

```

Notification class library


```
1 static void Main(string[] args)
2 {
3     Console.Title = "Notification";
4     var bus = Bus.Factory.CreateUsingRabbitMq(cfg =>
5     {
6         cfg.Host(new Uri(RabbitMqConsts.RabbitMqRootUri), h =>
7         {
8             h.Username(RabbitMqConsts.UserName);
9             h.Password(RabbitMqConsts.Password);
10        });
11        cfg.ReceiveEndpoint("todoQueue", ep =>
12        {
13            ep.PrefetchCount = 16;
14            ep.UseMessageRetry(r => r.Interval(2, 100));
15            ep.Consumer<TodoConsumerNotification>();
16        });
17    });
18
19    bus.StartAsync();
20    Console.WriteLine("Listening for Todo registered events.. Press enter to exit");
21    Console.ReadLine();
22    bus.StopAsync();
23
24 }
```

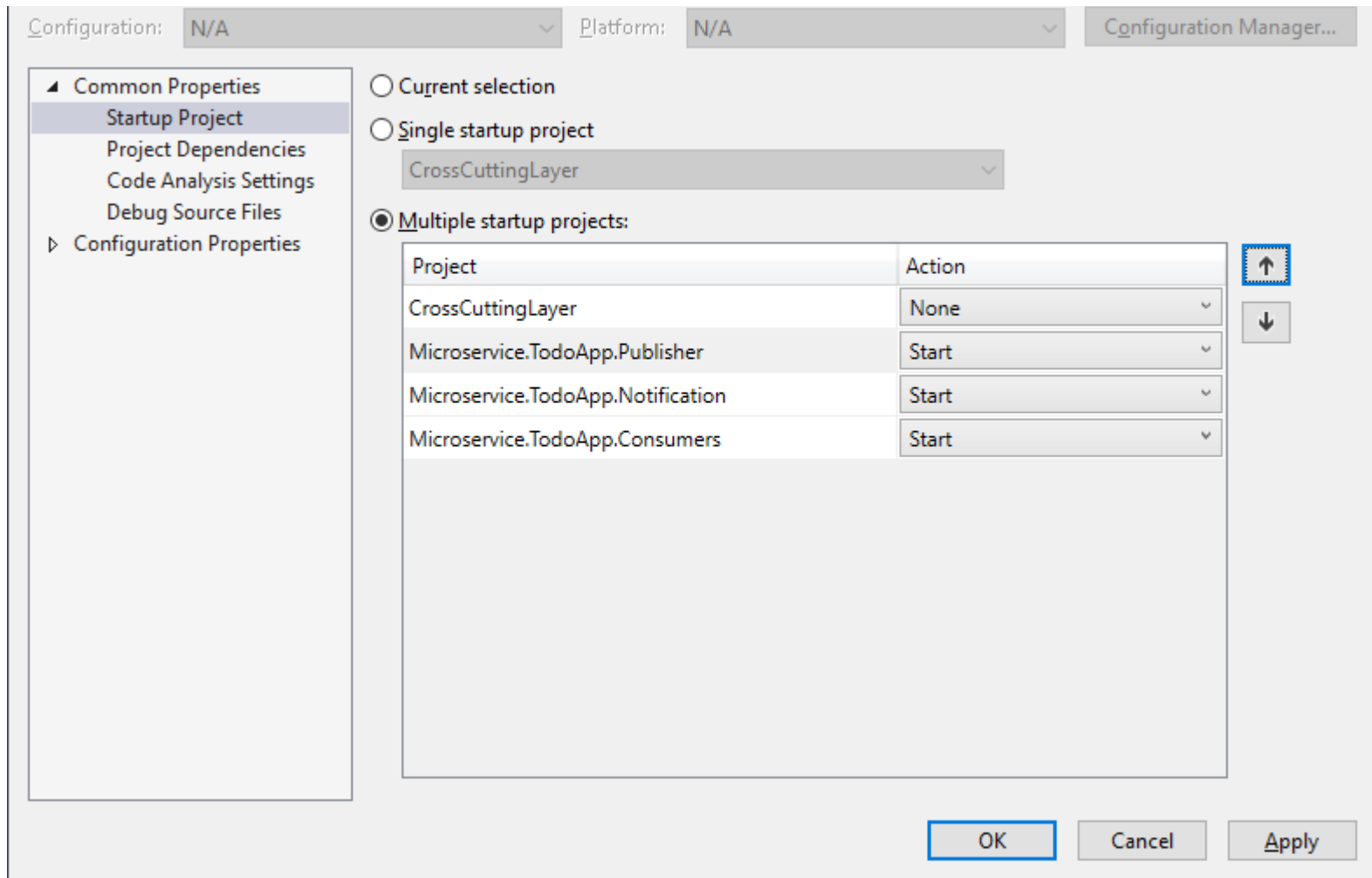
```
1 public class TodoConsumerNotification : IConsumer<Todo>
2 {
3     public async Task Consume(ConsumeContext<Todo> context)
```

```
6      }  
7  }
```

Testing our application

Scenario 1

Keeping Publisher and Consumer/Notification running as Multiple startup projects.



Call post method API/todo via Swagger like below image,

POST /api/ToDo

Parameters

No parameters

Request body

application/json

```
{
  "id": "FirstTask",
  "createdTime": "2021-06-20T06:48:18.536Z",
  "taskDescription": "First task for testing keeping publisher and consumer up",
  "isCompleted": true
}
```

Output

 RabbitMQ With ASP.NET Core - Microservice Communication With MassTransit

Scenario 2

Keeping Publisher set as start-up project and Consumer/Notification running as offline.

Call post method API/todo via Swagger like the below image,

 RabbitMQ With ASP.NET Core - Microservice Communication With MassTransit

You can see that our Queue has 1 new Message Ready which does not deliver yet. It will keep the messages in memory till a consumer is connected.

 RabbitMQ With ASP.NET Core - Microservice Communication With MassTransit

Let's bring our Consumer online now, and we can see the message is delivered and Queue is 0.

 RabbitMQ With ASP.NET Core - Microservice Communication With MassTransit

 RabbitMQ With ASP.NET Core - Microservice Communication With MassTransit

Summary

In this article, we have gone through Message Brokers, RabbitMQ, Advantages, Integrating RabbitMQ with ASP.NET Core using MassTransit. We also build a small prototype application to send data over the RabbitMQ Server. [Download source code from GitHub](#)

ASP.NET Core

Microservice Communication With MassTransit

RabbitMQ

RabbitMQ With ASP.NET Core

Next Recommended Reading

[Consuming RabbitMQ Messages In ASP.NET Core](#)

OUR BOOKS





Amit Naik *TOP 1000*

Technical Lead | Full Stack Developer | Design pattern, Asp.net Core, .Net Core, C#, Sqlserver, Angular, React, MSBI, Webcomponent

877

216.6k

1

5

3



Type your comment here and press Enter Key (Minimum 10 characters)



I have "HTTP Error 500.0 - ANCM In-Process Handler Load Failure" error. What to do to fix this?

fg

2132 9 0

Oct 14, 2021

0 0 Reply



How to return consume data to Controller?

sky Kumar

2000 141 0

Jul 23, 2021

0 0 Reply



How to return data from Consume

sky Kumar

2000 141 0

Jul 23, 2021

0 0 Reply

FEATURED ARTICLES

Simple QRCode Generator Using SPFx

Microservices Async Communication Using Ocelot Gateway, RabbitMQ, Docker, And Angular 14

Build A Blazor Hybrid App with .NET MAUI for Cross-Platform Application

Git - Comparing Visual Studio 2022 With MeGit/EGit And SourceTree

TRENDING UP

01 Row_Number(),Rank(),Dense_Rank(),Lead(),Lag() Function In SQL

02 Introducing Carbon - Google's New Programming Language

03 Using Tuples Data Structure In Python 😊

04 Python Constants

05 How To Receive Real-Time Data In .NET 6 Client Application Using SignalR

06 RabbitMQ Message Queue Using .NET Core 6 Web API

07 Getting Started With .NET MAUI For Cross Platform Application Development

08 Build A Blazor Hybrid App with .NET MAUI for Cross-Platform Application

09 Create QR Code Using Google Charts API In VB.Net

10 Filtering In Datagridview In Vb.Net And Also In C#



Angular 8 in 10 Days

CHALLENGE YOURSELF



Angular 13

GET CERTIFIED



JavaScript Developer

[About Us](#) [Contact Us](#) [Privacy Policy](#) [Terms](#) [Media Kit](#) [Sitemap](#) [Report a Bug](#) [FAQ](#) [Partners](#)

[C# Tutorials](#) [Common Interview Questions](#) [Stories](#) [Consultants](#) [Ideas](#) [Certifications](#)

©2022 C# Corner. All contents are copyright of their authors.