

.NET에서 비동기 프로그래밍 배우기 at 20Q1

문성원 at Planetarium

@longfin / swen@planetariumhq.com

들어가기 전에

이 발표는 .NET Conf 2019 서울에서 발표한 ".NET에서 비동기 프로그래밍 배우기"를 판 올림한 것입니다. 😊



😊 Set status

Swen Mun

longfin

Edit profile

@planetarium

South Korea, Seoul

longfinfunnel@gmail.com

longfin.github.com

Organizations



Overview

Repositories 72

Projects 0

Packages 0

Stars 64

Followers 37

Following 10

Pinned

Customize your pins

planetarium/libplanet

Blockchain core in C#/.NET for persistent peer-to-peer online games

C# ★ 129 🍴 28

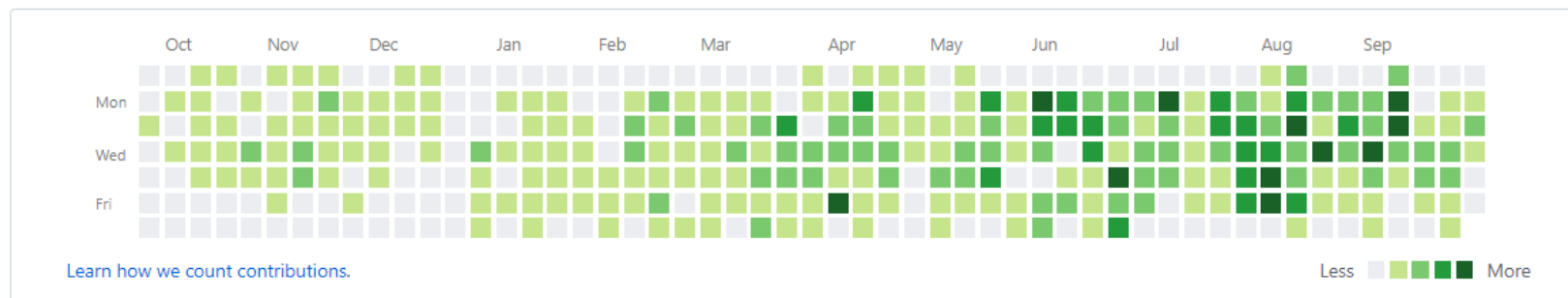
planetarium/libplanet-explorer

Explorer blocks, transactions, and addresses on your Libplanet-powered distributed games

C# ★ 6 🍴 12

2,214 contributions in the last year

Contribution settings



Contribution activity

October 2019



Created 5 commits in 2 repositories

planetarium/libplanet-explorer-frontend 3 commits

2019

2018

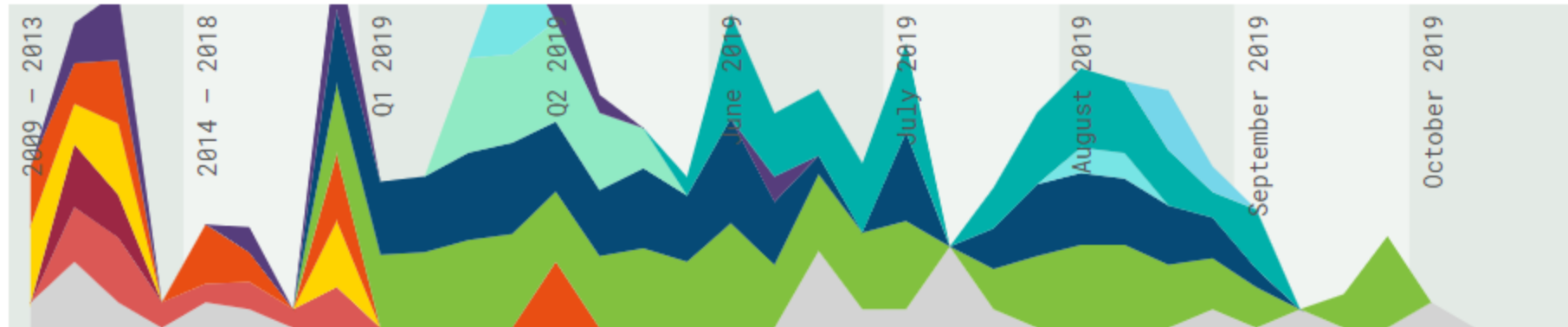
2017

Overview

31 repos

Last updated: 2019/10/04 - 02:10:00

2



Clojure



Scheme



JavaScript



HTML



C#



xunit



Travis CI



ASP.NET



CSS



LiteDB



Benchmark...



Unattribut...



GitHub 밖에선 이렇게합니다. 😅


- PHP로 프로그래밍을 처음 배워서
- Java로 웹 개발을 이어서 하다
- 스타트업에서 Python/TypeScript로 이것저것 만지다가
- 지금은 C#으로 블록체인을 만들고 있습니다.



c# async programming




 전체

 동영상

 이미지

 뉴스

 쇼핑

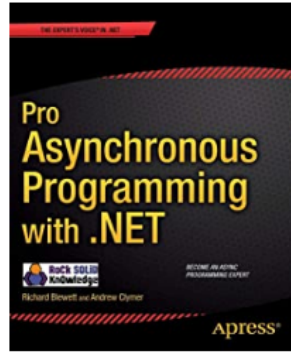
 더보기

설정

도구

검색결과 약 2,530,000개 (0.47초)

 ...



Pro Asynchronous Programming with .NET
by Richard Blewett, Andrew Clymer, et al.

★★★★☆ ~ 12

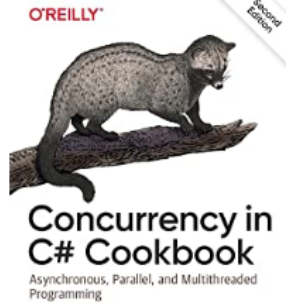
Paperback

\$30⁶⁸ to rent
\$47.75 to buy

More Buying Choices
\$42.03 (40 used & new offers)

Other format: Kindle

Best Seller



Stephen Cleary

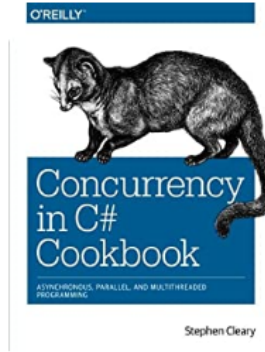
Concurrency in C# Cookbook:
Asynchronous, Parallel, and
Multithreaded Programming
by Stephen Cleary

Paperback

\$38¹⁵ \$49.99

More Buying Choices
\$28.77 (22 used & new offers)

Other format: Kindle



Stephen Cleary

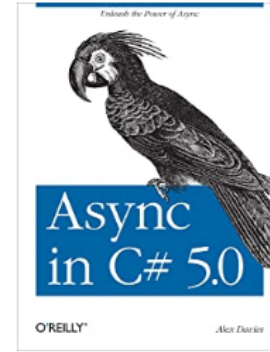
Concurrency in C# Cookbook
by Stephen Cleary

★★★★☆ ~ 34

Paperback

\$38²⁶

More Buying Choices
\$27.21 (28 used & new offers)



Alex Davies

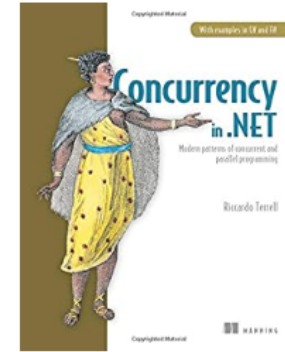
Async in C# 5.0: Unleash the Power
of Async
by Alex Davies

★★★★☆ ~ 21

Kindle

\$9⁵⁶ \$9.99

Other format: Paperback



Riccardo Terrell

Concurrency in .NET: Modern
patterns of concurrent and parallel
programming
by Riccardo Terrell

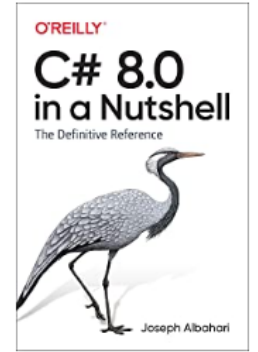
★★★★☆ ~ 7

Paperback

\$46⁶⁰ \$59.99

Only 17 left in stock (more on the way).

More Buying Choices
\$41.67 (23 used & new offers)



Joseph Albahari

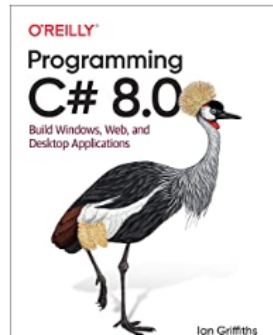
C# 8.0 in a Nutshell: The Definitive
Reference
by Joseph Albahari and Eric Johannsen

Paperback

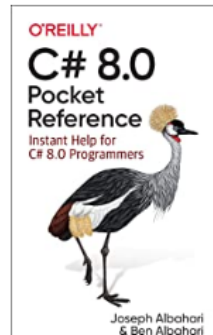
\$75⁹⁹ \$79.99

Pre-order Price Guarantee.

This title has not yet been released.



Ion Griffiths



Joseph Albahari
& Ben Albahari

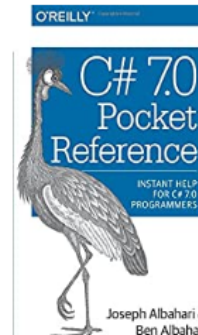


O'REILLY

Joseph Albahari
& Ben Albahari



Jon Skeet



Joseph Albahari
& Ben Albahari



Joseph Albahari & Ben Albahari

그래서 오늘 이야기 할 것들

- 배경
- 배운 것들 & 해 본 것들
- 유용한 라이브러리들
- Q&A

배경

Planetarium is an open source blockchain platform set to transform the longevity and the profit model of games. Rather than simply introducing blockchain technology into games, we intend to make the games run forever through decentralization and pioneer a new form of community-powered games.



GitHub

Planetarium is an open source **blockchain** platform set to transform the longevity and the profit model of games. Rather than simply introducing blockchain technology into games, we

Unity에서 쓸 수 있는 **블록체인 코어** 라이브러리를 만듭니다.

블록체인의 구성 요소

- 저장소 (Storage)
- 합의 방식 (Consensus)
- 네트워크 (Network)

네트워크 없이 만들었던 PoC

- 저장소: 파일 기반
- 합의 방식: Hashcash PoW(Proof of Work)
- 네트워크: 드롭박스 공유 기능을 통해 파일을 통째로 공유

A blockchain TCG (sort of)

Edit

[Manage topics](#)

🕒 20 commits

🔗 1 branch

📦 0 packages

🏷 0 releases

👤 1 contributor

📄 GPL-3.0

Branch: master ▾

New pull request

Create new file

Upload files

Find File

Clone or download ▾



dahlia Merge pull request #3 from kanghyojun/change-states ...

Latest commit 861dc58 on Nov 12, 2018

📁 cardchain	States should be libplanet.action.States	11 months ago
📁 hooks	Initial commit	last year
📄 .gitignore	Initial commit	last year
📄 LICENSE	Initial commit	last year
📄 README.rst	Reverse search received transactions of an account	11 months ago
📄 setup.cfg	Use whitespace around parameter equals with annotation	11 months ago
📄 setup.py	Initial commit	last year

📖 README.rst



블록체인은 이제 대강 알 것 같은데, 슬슬 네트워크 만들어야 하지 않나요?

P2P 블록체인 네트워크?

- 서버가 없다 ≠ Listen이 없다.
- 의외로 TCP 서버와 크게 다를 것도 없긴 한데,
 - 웹서버도 직접 짜본 적이 없더라고요. 😊
- 피어에 연결하고 있는 중에도 다른 피어로부터 연결을 처리해야 합니다.



c# asynchronous programming



C#의 비동기 프로그래밍 | Microsoft Docs

<https://docs.microsoft.com> > ko-kr > programming-guide > concepts > async ▼

2019. 3. 17. - async, await 및 Task를 사용하여 비동기 프로그래밍을 지원하는 **C#** 언어에 대해 간략히 설명합니다.

C#/.NET에서 지원하는 비동기 프로그래밍 패턴들

- Event-based Asynchronous Pattern (EAP)
- Asynchronous Programming Pattern (APM)
- **Task-based Asynchronous Pattern (TAP)**
 - MS에서 권장하는 방법이라니 이걸로 해보기로

첫 인상

```
var httpResult = await client.GetAsync("https://google.com");
```

- 편해보인다. & 있어보인다.
- 다른 언어(e.g. Python, TypeScript)랑 크게 다르지 않네?
 - 사실 이 쪽이 원조라고 합니다. 🧑
- 근데 저러면 매번 스레드가 생겨서 기다리는 건가?
 - 다행히 그렇지는 않았습니다. 😊

그러던 와중에...

h**g: ZeroMQ 란걸 쓰면 소켓 프로그래밍을 안전하게 할 수 있다던데요.

ZeroMQ

- <https://zeromq.org/>
- POSIX 소켓과 굉장히 비슷한 인터페이스를 제공하지만 사실은 준 프레임워크
- 다양한 편의 기능 제공
 - 다양한 비동기 프로그래밍 패턴 제공
 - 길이 제한이 없는 멀티 파트 메시지
 - Listen / Connect 의 순서 제약이 없음
 - 자동 재연결 처리

사족 🙄

- 알아보니 nanomsg라는 게 있었고...
 - 또 다시 알아보니 nng라는 게 있었고...
 - 이걸 C#/.NET 어떻게 쓰나 보니 [nng.NET](#) 이라는 게 있었는데...
 - 기껏 다 만들었더니 Windows 에서만 겨우 겨우 돌아갔습니다.
- 이식성이 좋은 라이브러리를 찾아봅시다.

NetMQ

- <https://github.com/zeromq/netmq>
- C#/.NET에서 바로 쓸 수 있는 구현체
- 바인딩이 아닌 순수 C#으로 포팅 되었고
- 나름대로 C# 스타일로 작성 되어 있습니다.

Peer discovery #30

Edit

Merged longfin merged 5 commits into `planetarium:master` from `longfin:net` on Jan 14

Conversation 41

Commits 5

Checks 1

Files changed 12

+1,375 -1



longfin commented on Jan 7 • edited

Member + 😊 ...

This PR is completed version of #21. (it's based on [planetarium/libplanet-python#3](#))

- Due to compatibility issues with `.NET` bindings, I used `NetMQ` instead of `nng`.
 - `NetMQ` doesn't have concept like bus, so I used `ROUTER-DELAER` patterns. Please let me know if you have an awkward part because I'm new to zmq/NetMQ.
- Added `Serilog` as logging library to debug async operations.



Peer type

9b9effb



longfin force-pushed the `longfin:net` branch 8 times, most recently from `dccacd0` to `67d7823` on Jan 7



longfin requested review from [dahlia](#), [kijun](#) and [ipdae](#) on Jan 9



longfin changed the title ~~[WIP] Peer discovery~~ Peer discovery on Jan 9



longfin referenced this pull request on Jan 9

WIP: Peer discovery #21



dahlia approved these changes on Jan 9

View changes

Reviewers



ipdae



kijun



dahlia



Assignees



No one—assign yourself

Labels



None yet

Projects



None yet

Milestone



0.1.0

Notifications

Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

4 participants

Closed

배운 것들 & 해 본 것들

Swarm<T>

- 네트워크 파사드 (Facade)
- 서버 / 클라이언트 역할을 동시에 수행...
 - 한다곤 하지만 TCP 서버에 가깝습니다.
- 메시지 송수신 이외에 주기적으로 수행하는 특수한 작업들도 있습니다.
 - 트랜잭션 전파
 - 라우팅 테이블 갱신

→ 오늘 자세히 다루진 않습니다. 😊

Task.WhenAll() 은 이럴 때 많이 쓰시죠?

```
var tasks = new Task[]
{
    GetAsync("https://github.com/planetarium/libplanet"),
    GetAsync("https://github.com/planetarium/libplanet-explorer"),
    GetAsync("https://github.com/planetarium/libplanet-explorer-frontend")
};

await Task.WhenAll(tasks);
```

이런 일에도 쓸 수 있습니다.

```
var tasks = new Task[]  
{  
    ReceiveMessage(),  
    BroadcastTxs(),  
    RefreshTable(),  
};  
  
await Task.WhenAll(tasks);
```

예외 처리를 해볼까요. 🌟

```
var tasks = new Task[]
{
    ReceiveMessage(),
    BroadcastTx(),
    RefreshTable(),
};

try
{
    await Task.WhenAll(tasks);
}
catch (RefreshTableException)
{
    RebuildTable();
}
```

하지만 RefreshTableException 은 발생하지 않습니다.

를 봅시다.

Creates a task that will complete when **all** of the `Task` objects in an enumerable collection have completed.

<https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenall?view=netstandard-2.0>

Task.WhenAny() 를 써보죠.

```
var tasks = new Task[]
{
    ReceiveMessage(),
    BroadcastTx(),
    RefreshTable(),
};

try
{
    await Task.WhenAny(tasks);
}
catch (RefreshTableException)
{
    RebuildTable();
}
```

하지만 여전히 RefreshTableException 은 발생하지 않습니다.

다시 를 봅시다.

Creates a task that will complete when any of the supplied tasks have completed.

[https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenany?
view=netstandard-2.0](https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenany?view=netstandard-2.0)

별 문제 없는 것 같은데요?

다시 를 (끝까지) 봅시다.

Returns

`Task<Task>`

A task that represents the completion of one of the supplied tasks. The return task's Result is the task that completed.

<https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenany?view=netstandard-2.0>

완료된 `Task` 그대로 반환하기 때문에 아무런 예외가 발생하지 않습니다.

그럼 어떻게 하죠?

```
var tasks = new Task[]
{
    ReceiveMessage(),
    BroadcastTx(),
    RefreshTable(),
};

try
{
    await await Task.WhenAny(tasks);
}
catch (RefreshTableException)
{
    RebuildTable();
}
```

Dispose()에선 await 못하나요?

```
class Swarm<T> : IDisposable
{
    public void Dispose()
    {
        await SayGoodByeAsync(); // CS4033
    }
}
```

`.Wait()` 라는게 있다던데...

```
class Swarm<T> : IDisposable
{
    public void Dispose()
    {
        SayGoodByeAsync().Wait();
    }
}
```

에러는 없어졌네요. 그치만...

`.Wait()` 라는게 있다던데...

```
class Swarm<T> : IDisposable
{
    public void Dispose()
    {
        SayGoodByeAsync().Wait();
    }
}
```

`Swarm<T>.Dispose()` 가 `SynchronizationContext` 를 갖는 스레드에서 실행하면 데드락이 발생합니다.

IAsyncDisposable

- <https://docs.microsoft.com/ko-kr/dotnet/api/system.iasyncdisposable?view=netcore-3.0>
- `public System.Threading.Tasks.ValueTask DisposeAsync();`
- 하지만 .netstandard 2.0 (& Unity)에서는 쓸 수가 없네요. 😞

적당히 타협하기

```
class Swarm<T>
{
    public Task StopAsync()
    {
        await SayGoodByeAsync();
    }
}
```

- Libplanet(라이브러리)에서는 `Dispose()` 대신 `async StopAsync()` 만을 제공하고
- 게임에서는 일정 시간 동안만 이를 기다리고 종료하게끔 작성합니다.

Unity에서 잘 돌아갈까요?

- Unity에서 사용되는 C#/.NET 런타임은 Mono
 - 정규 버전이 아닌 Bleeding Edge
- NetMQ는 .NET Framework와 .NET Core를 모두 지원하긴 하니... 괜찮지 않을까요?



netmq unity



전체

지도

동영상

이미지

쇼핑

더보기

설정

도구

검색결과 약 3,020개 (0.33초)

도움말: **한국어** 검색결과만 검색합니다. 환경설정에서 검색 언어를 지정할 수 있습니다.


NetMQ + Unity3D, am I wasting my time? · Issue #631 ... - GitHub

<https://github.com> › [zeromq](#) › [netmq](#) › [issues](#) ▾ 이 페이지 번역하기

2016. 12. 3. - or, is there a localized problem only when attaching Mono Develop debugger to
Unity3D + NetMQ (i.e. **NetMQ** is good for production but I ...

NetMQ + Unity3D, am I wasting my time? #631

New issue

 Open Sohojoe opened this issue on 4 Dec 2016 · 28 comments



Sohojoe commented on 4 Dec 2016

+ 😊 ...

I have been trying to implement a basic REQ/REP model with Unity3D (Mac OS X) acting as the server and python as the client. It works OK as a standalone Unity program. However Unity will soft-lock after 10-40 frames if i run with the Mono Develop debugger attached (requiring a Force Quit)

I have read through the numerous open and closed issues with regards to problems with NetMQ and Unity3D;

1. Is there a fundamental problem with Unity3D + NetMQ (i.e. I wasting my time with NetMQ and should look for an alternative)
2. or, is there a localized problem only when attaching Mono Develop debugger to Unity3D + NetMQ (i.e. NetMQ is good for production but I should look for another solution for debugging)
3. or, it's something hookey with what I'm trying to do (so either we debug that or I change my architecture)

Things I have tried

- building NetMQ and AsyncIO from master
- adding AsyncIO.ForceDotNet.Force ();
- adding NetMQConfig.ManualTerminationTakeOver();
- adding NetMQConfig.ContextCreate(true);
- using RouterSocket instead of ResponseSocket
- running server code on a seperate thread (per one of the example in the Issues

Environment

NetMQ Version:

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

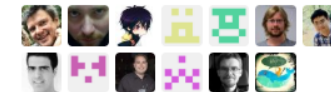
Notifications

Customize

 Subscribe

You're not receiving notifications from this thread.

13 participants



<https://github.com/zeromq/netmq/issues/631>

범인은 AsyncIO

- <https://github.com/somdoron/AsyncIO>
- NetMQ에서 비동기 소켓이 필요할땐 .NET 소켓 대신 이것을 사용하는데...
- 실행하는 플랫폼에 따라 IOCP / .NET Completion Port를 구분합니다.
- Unity + Windows에선 IOCP가 제대로 동작하지 않습니다. 😞

May the **Force** be with you

```
static Swarm()  
{  
    if (!(Type.GetType("Mono.Runtime") is null))  
    {  
        ForceDotNet.Force();  
    }  
}
```

<https://github.com/planetarium/libplanet/blob/master/Libplanet/Net/Swarm.cs#L76-L82>

동기 메소드를 비동기로 만들기

- NetMQ(<4.0.0.239-pre)의 API는 모두 동기 메소드
- 아무리 겉의 인터페이스를 TAP로 작성해도 소켓은 블록될 수 밖에 없는 구조

```
public async SendMessage(Peer peer)
{
    Dealer socket = CreateSocket(peer);
    socket.Connect(); // Blocking!
    socket.Send("hello"); // and, Blocking!
    socket.Receive();
}
```

```

internal static class NetMQSocketExtension
{
    public static async Task SendFrameAsync(
        this IOutgoingSocket socket,
        byte[] data,
        TimeSpan? timeout = null,
        TimeSpan? delay = null,
        CancellationToken cancellationToken = default(CancellationToken))
    {
        TimeSpan delayNotNull = delay ?? TimeSpan.FromMilliseconds(100);
        TimeSpan elapsed = TimeSpan.Zero;

        while (!socket.TrySendFrame(data))
        {
            await Task.Delay(delayNotNull, cancellationToken);

            elapsed += delayNotNull;
            if (elapsed > timeout)
            {
                throw new TimeoutException(
                    "The operation exceeded the specified time: " +
                    $"{timeout}.");
            }
        }
    }
}

```


근본적인 문제가 남아 있습니다.

- ZeroMQ / NetMQ의 모든 소켓은 스레드-안전(Thread-safe)하지 않습니다.
 - TMI) 요건 버그가 아니라 설계 철학에 해당하는 부분이라고 하네요.
- 실행 문맥이 보존되는 환경이라면 괜찮지만, 그렇지 않은 환경에서는 오작동의 원인이 될 수 있습니다.



Doron Somech

@somdoron



#NetMQ now supports async/await
[netmq.readthedocs.io/en/latest/asyn...](https://netmq.readthedocs.io/en/latest/async-await.html)

[트윗 번역하기](#)

오후 7:38 · 2019년 7월 7일 · [Twitter Web Client](#)

3 리트윗 **7** 마음에 들어요

Async Await

Since version (4.0.0.239-pre) NetMQ support async/await.

To use async/await feature you need to create a `NetMQRuntime`.

Example:

```

async Task ServerAsync()
{
    using (var server = new RouterSocket("inproc://async"))
    {
        for (int i = 0; i < 1000; i++)
        {
            var (routingKey, more) = await server.ReceiveRoutingKeyAsync();
            var (message, _) = await server.ReceiveFrameStringAsync();

            // TODO: process message

            await Task.Delay(100);
            server.SendMoreFrame(routingKey);
            server.SendFrame("Welcome");
        }
    }
}

async Task ClientAsync()
{
    using (var client = new RouterSocket("inproc://async"))
    {

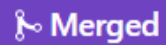
```

아이디어는 이렇습니다.

- `async / await` 를 사용하는 환경(`Task`)를 구분하는 실행 단위(`NetMQRuntime`)을 만들고
- `NetMQPoller` 로 단일 스레드를 붙여 둔 다음
- `SynchronizationContext.SetSynchronizationContext()` 를 통해 실행 단위 안에서 동기화 문맥을 고정합니다.

Problem: NetMQ cannot be used with async/await pattern

#802



somdoron merged 1 commit into `zeromq:master` from `somdoron:master` on Jul 7

💬 Conversation 6

🔗 Commits 1

📄 Checks 0

📄 Files changed 10

+437 -51



somdoron commented on Jul 7

Member



Solution: Add ReceiveAsync (and some overloads) to socket. Also, a NetMQRuntime must be created and run in order to signal the completion of the tasks



somdoron requested a review from drewnoakes on Jul 7



somdoron force-pushed the `somdoron:master` branch 2 times, most recently from `1a28192` to `05a96ab` on Jul 7

Reviewers



drewnoakes



Assignees

No one assigned

Labels

None yet

Projects

```
async Task ServerAsync()
{
    using (var server = new RouterSocket("inproc://async"))
    {
        for (int i = 0; i < 1000; i++)
        {
            var (routingKey, more) = await server.ReceiveRoutingKeyAsync();
            var (message, _) = await server.ReceiveFrameStringAsync();

            // TODO: process message

            await Task.Delay(100);
            server.SendMoreFrame(routingKey);
            server.SendFrame("Welcome");
        }
    }
}
```

```
async Task ClientAsync()
{
    using (var client = new DealerSocket("inproc://async"))
    {
        for (int i = 0; i < 1000; i++)
        {
            client.SendFrame("Hello");
            var (message, more) = await client.ReceiveFrameStringAsync();

            // TODO: process reply

            await Task.Delay(100);
        }
    }
}

static void Main(string[] args)
{
    using (var runtime = new NetMQRuntime())
    {
        runtime.Run(ServerAsync(), ClientAsync());
    }
}
```

그런데 빠진게 있더라고요.

- 추가된 비동기 메소드들이 취소 토큰(`CancellationToken`)을 받지 않았습니다. 😞
 - 덕분에 타임 아웃도 구현할 수 없었죠.
- 저희가 애용하던 멀티 파트 메세지 지원도 없었습니다. 😞
- 덤으로 테스트도 깨지고 있었습니다. 😞

제멋대로 오픈 소스 철학

- 상용 코드의 퀄리티 \geq 오픈 소스 코드의 퀄리티
- 혼자 구현한 코드의 퀄리티 $<$ 같이 구현한 코드의 퀄리티

그런고로 직접 고쳐보기로 했습니다.

```
public static async Task<NetMQMessage> ReceiveMultipartMessageAsync(  
    [NotNull] this NetMQSocket socket,  
    int expectedFrameCount = 4,  
    CancellationToken cancellationToken = default(CancellationToken))  
{  
    var message = new NetMQMessage(expectedFrameCount);  
  
    while (true)  
    {  
        (byte[] bytes, bool more) = await socket.ReceiveFrameBytesAsync(cancellationToken);  
        message.Append(bytes);  
  
        if (!more)  
        {  
            break;  
        }  
    }  
  
    return message;  
}
```

```

public static Task<(byte[], bool)> ReceiveFrameBytesAsync(
    [NotNull] this NetMQSocket socket
)
{
    if (NetMQRuntime.Current == null)
        throw new InvalidOperationException("NetMQRuntime must be created before calling async functions");
    socket.AttachToRuntime();
    var msg = new Msg();
    msg.InitEmpty();
    if (socket.TryReceive(ref msg, TimeSpan.Zero))
    {
        var data = msg.CloneData();
        bool more = msg.HasMore;
        msg.Close();
        return Task.FromResult((data, more));
    }

    TaskCompletionSource<(byte[], bool)> source = new TaskCompletionSource<(byte[], bool)>();

    void Listener(object sender, NetMQSocketEventArgs args)
    {
        if (socket.TryReceive(ref msg, TimeSpan.Zero))
        {
            var data = msg.CloneData();
            bool more = msg.HasMore;
            msg.Close();
            socket.ReceiveReady -= Listener;
            source.SetResult((data, more));
        }
    }
    socket.ReceiveReady += Listener;
    return source.Task;
}

```

```
public static Task<(byte[], bool)> ReceiveFrameBytesAsync(
    [NotNull] this NetMQSocket socket,
    CancellationToken cancellationToken = default(CancellationToken)
)
{
    // ...

    TaskCompletionSource<(byte[], bool)> source = new TaskCompletionSource<(byte[], bool)>();
    cancellationToken.Register(() => source.SetCanceled());


    // ...
    return source.Task;
}
```

다행히 업스트림에도 머지되었습니다. 😊

Merged

Enhence AsyncReceiveExtension #816

somdoron merged 4 commits into `zeromq:master` from `longfin:feature/enhance-async` 29 days ago




longfin commented 29 days ago

Contributor + 😊 ...



(This PR is cherry-picked version of [planetarium#1](#))

This PR adds some feature to `NetMQ.AsyncReceiveExtensions` as below.



- Added `.RecevieMultipartMessageAsync()` to receive a multi part message easily.
- Added `CancellationTok` argument to async extension methods in `AsyncReceiveExtensions`. (resolves [#813](#))
- Fixed `AsyncTest` to indicate assertion properly.
- Removed state check on `NetMQPoller.Stop()` to prevent freezing during unit testing.





longfin added 4 commits on Aug 28

 Fix unittest hanging ...



da7044e

 Fix AsyncTest


516c8a8

 Add cancellation support to AsyncExtension

c13d912

 Add ReceiveMultipartMessageAsync

✖ 659f69e



somdoron merged commit `4bf9df0` into `zeromq:master` 29 days ago

1 of 3 checks passed

View details

Revert

Reviewers

No reviews

Assignees

No one assigned

Labels

None yet


Projects

None yet

Milestone



No milestone

Notifications

 Uns

You're receiving not you authored the th

2 participants



유용한 라이브러리들

AsyncEx

- <https://github.com/StephenCleary/AsyncEx>
- .NET에서 제공하는 기능/동기 API의 비동기 버전 제공
 - `lock` → `AsyncLock`
 - `BlockingCollection<T>` → `AsyncCollection<T>`
 - `AutoResetEvent` → `AsyncAutoResetEvent`
- 여러가지 유틸리티
 - `AsyncContext`
 - `AsyncLazy`
 - `CancellationTokenHelpers`

AsyncLock

```
var _mutex = new object();  
lock (_mutex)  
{  
    await RunAsync(); // CS1996 Cannot await in the body of a lock statement  
}
```

```
var _mutex = new AsyncLock();  
using (await _mutex.LockAsync())  
{  
    await RunAsync();  
}
```


AsyncCollection<T>

```
var requests = new AsyncCollection<Message>();

// Add
await requests.AddAsync(new Message("Hello"));

// receive
while (true)
{
    var request = await requests.TakeAsync(cancellationToken);
}
```

AsyncAutoResetEvent

```
class Swarm
{
    public Blockchain Blockchain { get; }
    public AsyncAutoResetEvent BlockReceived { get; }

    private void ReceiveBlock()
    {
        // ...
        BlockReceived.Set();
    }
}

// Unittest
[Fact]
public async Task BlockReceiving()
{
    var swarm = new Swarm();
    await swarm.RunAsync();

    // ...

    await swarm.BlockReceived;
    Assert.Contains(swarm.BlockChain, block);
}
```

AsyncEnumerable

- <https://github.com/Dasync/AsyncEnumerable>
- C# 8.0 에 추가되는 Async Streams등을 따라 이전 버전에서도 써 볼 수 있는 라이브러리

Streams

```
// Producer
public IEnumerable<byte[]> GetStreams()
{
    foreach (byte[] bytes in GetData())
    {
        yield return bytes;
    }
}

// Consumer
foreach (byte[] chunk in GetStreams())
{
    Console.WriteLine($"Received data size = {chunk.Count}");
}
```

Async Streams

```
// Producer
public async IEnumerable<byte[]> GetStreamsAsync()
{
    foreach (byte[] bytes in await GetDataAsync())
    {
        yield return bytes;
    }
}

// Consumer
await foreach (var chunk in GetStreamsAsync())
{
    Console.WriteLine($"Received data size = {chunk.Count}");
}
```

Async Streams (AsyncEnumerable)

```
using Dasync.Collections;

// Producer
public IAsyncEnumerable<byte[]> GetStreamsAsync()
{
    return new AsyncEnumerable(async yield =>
    {
        foreach (byte[] bytes in await GetDataAsync())
        {
            await yield.ReturnAsync(bytes);
        }
    });
}

// Consumer
await asyncStreams.ForEachAsync(chunk =>
{
    Console.WriteLine($"Received data count = {chunk.Count}");
});
```

ParallelForeachAsync()

```
await uris.ParallelForEachAsync(  
    async uri =>  
    {  
        var str = await httpClient.GetStringAsync(  
            uri,  
            cancellationToken  
        );  
        result.Add(str);  
    },  
    maxDegreeOfParallelism: 5,  
    cancellationToken: cancellationToken  
);
```

Microsoft.Bcl.AsyncInterfaces

- <https://www.nuget.org/packages/Microsoft.Bcl.AsyncInterfaces/>
- 단순한 인터페이스 정의지만... 🎮
- 놀랍게도, C# 8의 많은 기능이 예전 런타임(Unity마저!)에서도 동작합니다.
- 자세한 건 <https://stu.dev/csharp8-doing-unsupported-things/> 에서


```
// Producer
public async IEnumerable<byte[]> GetStreamsAsync()
{
    foreach (byte[] bytes in await GetDataAsync())
    {
        yield return yield.ReturnAsync(bytes);
    }
}

// Consumer
await foreach (byte bytes in asyncStream)
{
    Console.WriteLine($"Received data count = {chunk.Count}");
};
```

함께 하실 분을 찾고 있습니다.

<https://bit.ly/join-planet>

Q&A