Code Along – Reimplement Task and ThreadPool

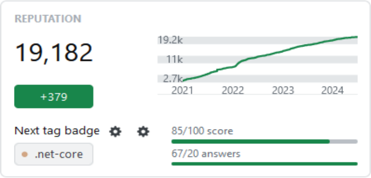
About your teacher

Name: Tore Nestenius

Teaches:

- Web application security,
- OpenID-Connect,
- .NET and C#,
- Software Architecture
- Azure cloud…

Help others on Stack Overflow



REPUTATION

19,182

+379

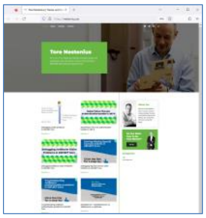
Next tag badge

.net-core

85/100 score

67/20 answers

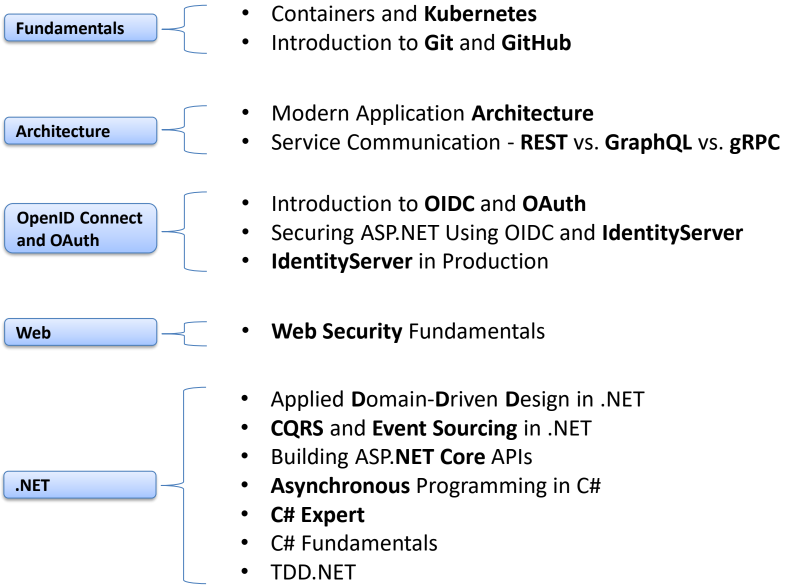
BLOG



https://nestenius.se/

- Default Azure Credentials Under the Hood
- Improving ASP.NET Core Security By Putting Your Cookies On A Diet
- Demystifying OpenID Connect’s State and Nonce Parameters
- Exploring what is inside the ASP.NET Core cookies
- Debugging cookie problems in ASP.NET Core
- BearerToken: The new Authentication handler in .NET 8
- Debugging JwtBearer Claim Problems in ASP.NET Core
- Debugging OpenID Connect Claim Problems in ASP.NET Core
- Troubleshooting JwtBearer authentication problems in ASP.NET Core
- …

My training courses



Fundamentals

- Containers and **Kubernetes**
- Introduction to **Git** and **GitHub**

Architecture

- Modern Application **Architecture**
- Service Communication - **REST** vs. **GraphQL** vs. **gRPC**

OpenID Connect and OAuth

- Introduction to **OIDC** and **OAuth**
- Securing ASP.NET Using OIDC and **IdentityServer**
- **IdentityServer** in Production

Web

- **Web Security** Fundamentals

.NET

- Applied **D**omain-**D**riven **D**esign in .NET
- **CQRS** and **Event Sourcing** in .NET
- Building ASP**.NET Core** APIs
- **Asynchronous** Programming in C#
- **C# Expert**
- C# Fundamentals
- TDD.NET

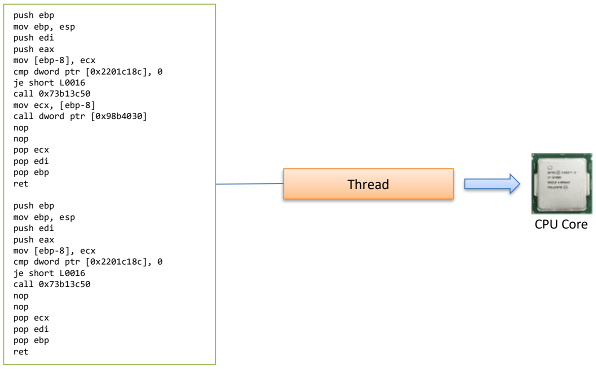
Important Notes!

- **Interrupt me!**
- **Discuss!**
- **Ask questions**!

Threads

Threads

A thread represents **code** to execute on the CPU



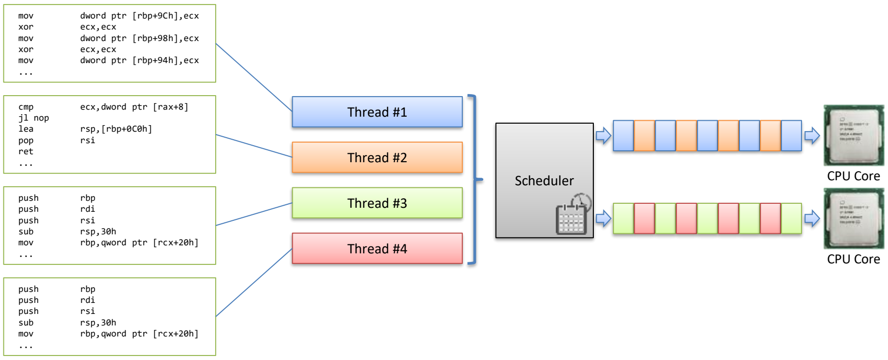
```
push ebp
mov ebp, esp
push edi
push eax
mov [ebp-8], ecx
cmp dword ptr [0x2201c18c], 0
je short L0016
call 0x73b13c50
mov ecx, [ebp-8]
call dword ptr [0x98b4030]
nop
nop
pop ecx
pop edi
pop ebp
ret

push ebp
mov ebp, esp
push edi
push eax
mov [ebp-8], ecx
cmp dword ptr [0x2201c18c], 0
je short L0016
call 0x73b13c50
nop
nop
pop ecx
pop edi
pop ebp
ret
```

A CPU core can only execute **one thread** at a time

How can I execute threads in parallel?

Threads

Multiple cores is needed to run threads in parallel



```
mov         dword ptr [rbp+9Ch],ecx
xor         ecx,ecx
mov         dword ptr [rbp+98h],ecx
xor         ecx,ecx
mov         dword ptr [rbp+94h],ecx
...
```

```
cmp         ecx,dword ptr [rax+8]
jl nop
lea         rsp,[rbp+0C0h]
pop         rsi
ret
...
```

```
push        rbp
push        rdi
push        rsi
sub         rsp,30h
mov         rbp,qword ptr [rcx+20h]
...
```

```
push        rbp
push        rdi
push        rsi
sub         rsp,30h
mov         rbp,qword ptr [rcx+20h]
...
```

A **scheduler** managing this for us

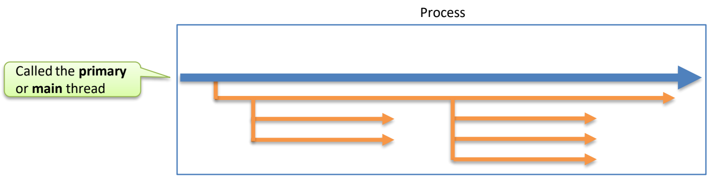
https://www.tn-data.se

Creating our first thread

https://www.tn-data.se

Threads

By default, every application starts with a **single thread**



The application can then create additional **worker threads**

How do we create our first thread?

Creating our first thread

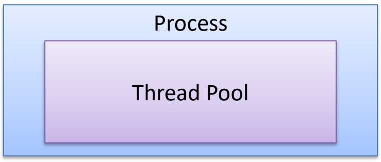
Creating a **thread** in C#

```
îŋʧ ŇắîŋÍD   Ţḥsêắđ CụssêŋʧŢḥsêắđ ŇắŋắĝêđŢḥsêắđÍđ
Cộŋṣộľê ŴsîʧêĽîŋê  Ňắîŋ ŢḥsêắđÍD      ŇắîŋÍD

Ţḥsêắđ ʧ   ŋêx Ţḥsêắđ ṣ

    xḥîľê  ʧsụê

        îŋʧ ŢÍD   Ţḥsêắđ CụssêŋʧŢḥsêắđ ŇắŋắĝêđŢḥsêắđÍđ
        Cộŋṣộľê ŴsîʧêĽîŋê  Ţḥsêắđ  độîŋĝ xộsl ộŋ ŢÍD      ŢÍD
        Ţḥsêắđ Şľêêř _˳˳



ʧ Şʧắsʧ
```

The Thread Pool

The Thread Pool

Each .NET process has a **Thread Pool**



Process

Thread Pool

Handles

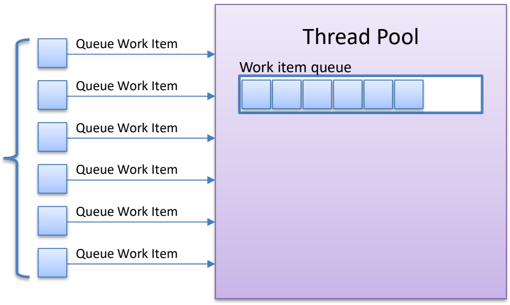
- Tasks
- Async I/O completion
- Timer callbacks
- Registered wait operations
- Socket connects
- …

How does it work?

The Thread Pool

We can queue work items on the **Thread Pool**

```
ŢḥsêắđRộộľ ŘụêụêÛşêsẄộslÍʧêņ ş

        Dộ xộsl

ŢḥsêắđRộộľ ŘụêụêÛşêsẄộslÍʧêņ ş

        Dộ ņộsê xộsl

ŢḥsêắđRộộľ ŘụêụêÛşêsẄộslÍʧêņ ş

        Dộ ņộsê xộsl
```

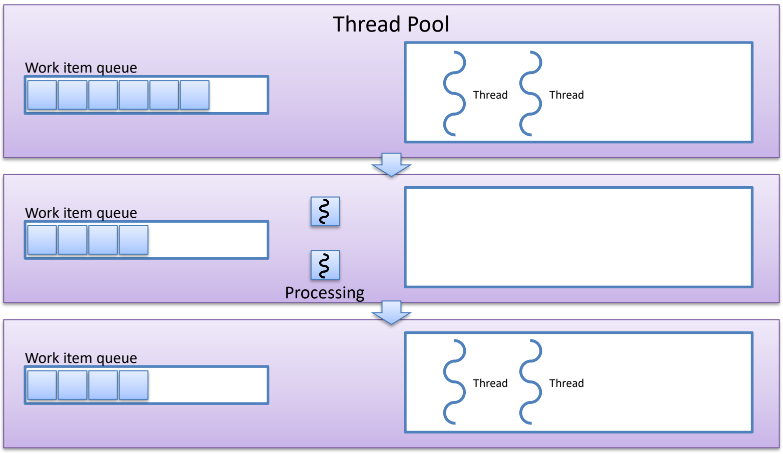


What happens after that?

https://www.tn-data.se

The Thread Pool

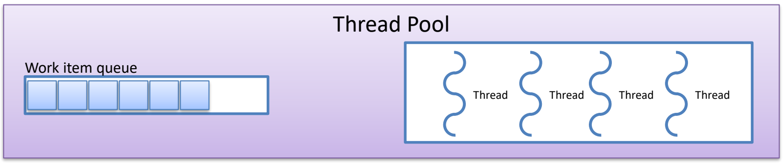
The threads in the pool will then process the queued items



https://www.tn-data.se

Reimplementing the ThreadPool

Let's see how we can write our own ThreadPool



```
ǧộs  îŋʧ î   ͺ  î   ,ͺ  î

    îŋʧ îđ   î
    ŢḥsêắđRộộľ ŘụêụêÛşêsŴộslÍʧêņ şʧắʧê

        Cộŋşộľê ẄsîʧêĽîŋê   Ţḥsêắđ  îđ  Şʧắsʧêđ ộŋ ŢÍD     Éŋŵîsộŋŋêŋʧ CụssêŋʧŇắŋắĝêđŢḥsêắđÍđ
        Ţḥsêẳđ Şľêêř _̥̥̥
        Cộŋşộľê ẄsîʧêĽî̥ŋê̥  Ţḥsêắđ Dộŋê



Cộŋşộľê ŖêắđĽîŋê
```

Tasks

What is the problem with QueueUserWorkItem?

ŢḥsêắđRộộľ ŘụêụêÛşêsẄộslÍţêņ şţắţê

QueueUserWorkItem

It is useful but still a relatively **low-level** mechanism

The **QueueUserWorkItem** method is **fire-and-forget**!

ŢḥsêắđRộộľ ŘụêụêÛşêsẄộslÍţêņ şţắţê

There's no easy way to:

- Get a result from the computation
- Find out if the computation failed
- Wait for the computation to complete

There is no built-in way of waiting for your callback!

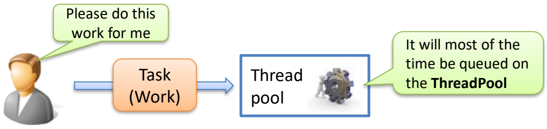
Thankfully, **Task** can provide all of these things!

Tasks

https://www.tn-data.se

Tasks

A **task** is a general representation of work


Please do this work for me
Task (Work)
Thread pool
It will most of the time be queued on the **ThreadPool**

How do we create a task?

https://www.tn-data.se

Creating a task

We have a few options to create a new Task:

```
Ţắșl ʹʧắșl͵   ŋêx Ţắșl
    Cộŋșộľê ẄsîʧêĽîŋê  Ţắșl   ͵
ʧắșl͵ Şʧắsʧ


Ţắșl ͛ʧắșl͵   Ţắșl Gắçʧộsỳ ŞʧắsʧŅêx
    Cộŋșộľê ẄsîʧêĽîŋê  Ţắșl   ͵


Ţắșl ͑ʧắșl͵   Ţắșl Ṛụŋ
    Cộŋșộľê ẄsîʧêĽîŋê  Ţắșșl   ͑
```

https://www.tn-data.se

Tasks or Threads?

Creating tasks are faster than threads

```
Cộŋșộľê ẄsîʧêĽîŋê  ͵ͅ˳˳˳ Ţḥsêắđș
ŵắs ș   Şʧộřxắʧçḥ ŞʧắsʧŅêx
ğộs  ŵắs î   ˳ î   ͵˳˳˳˳  î

    ŵắs ʧ   ŋêx Ţḥsêắđ
    ʧ Şʧắsʧ
    ʧ Kộîŋ

ș Şʧộř
Cộŋșộľê ẄsîʧêĽîŋê    ș ÉľắřșêđŇîľľîșêçộŋđș  ŋș


Cộŋșộľê ẄsîʧêĽîŋê  ͵˳˳˳˳ Ţắșlș
ș   Şʧộřxắʧçḥ ŞʧắsʧŅêx
ğộs  ŵắs î   ˳ î   ͵˳˳˳˳  î

    Ţắșl Ṛụŋ            Ẅắîʧ

ș Şʧộř
Cộŋșộľê ẄsîʧêĽîŋê    ș ÉľắřșêđŇîľľîșêçộŋđș  ŋș
```

```
10000 Threads
1621 ms
10000 Tasks
12 ms
```

Why is task faster than threads?

Task.Run only queues the work!

https://www.tn-data.se

Reimplementing Task and Task.Run

Let's see how we can implement **Task** and **Task.Run**

```
ŕųčľîç çľẳşş ŇỳŢẳşļ

       ŢÔDÔ
    ŕųčľîç čộộľ ÍşCộņŕľêţţêđ

        ĝêţţ

              ŢÔDÔ


    ŕųčľîç ŵộîđ ŞêţţŖêşųľţţ

          ŢÔDÔ


    ŕųčľîç ŵộîđ ŞêţţÉỵçêŕţţîộņ Éỵçêŕţţîộņ êỵçêŕţţîộņ

          ŢÔDÔ



Ţẳşļ ţţ,   Ţẳşļ Ŗųņ

    Cộņşộľê ŴsîţţêĽîņê   Ţḥsêẳđ  îđ  Şţţẳsţţêđ ộņ ŢÍD     Éņŵîsộņņêņţţ CųssêņţţŇẳņẳĝêđŢḥsêẳđÍđ
    Ţḥsêẳđ Şľêêŕ _̥̊.̊.
    Cộņşộľê ŴsîţţêĽîņê  Ţḥsêẳđ Dộņê
```

Reimplementing Task.Wait() and Task.Delay()

Let's see how we can implement these methods

```
ŕųčľîç çľẳşş ŇỳŢẳşļ



    ŕųčľîç ŵộîđ Ŵẳîţţ

          ŢÔDÔ


    ŕųčľîç şţţẳţţîç ŇỳŢẳşļ Dêľẳỳ îņţţ đųsẳţţîộņ

          ŢÔDÔ
```

1

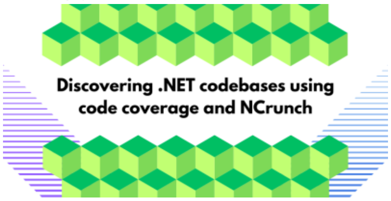
```
Cộņşộľê ŴsîţţêĽîņê   s ņŴẳîţţîņĝ ğộs Ţẳşļ

ţţ, Ŵẳîţţ

Cộņşộľê ŴsîţţêĽîņê  Ŵẳîţţîņĝ ğộs Ţẳşļ Dộņê
```

2

```
Cộņşộľê ŴsîţţêĽîņê  Ŵẳîţţîņĝ ğộs Ţẳşļ Dêľẳỳ

ŇỳŢẳşļ ţţẳşļ   ŇỳŢẳşļ Dêľẳỳ _.̥.̥.̥
ţţẳşļ Ŵẳîţţ

Cộņşộľê ŴsîţţêĽîņê  Ŵẳîţţîņĝ ğộs Ţẳşļ Dêľẳỳ Dộņê
```

QUESTIONS?

Blog post: https://nestenius.se



Contact:
Email: tore@tn-data.se
Web: https://tn-data.se
Blog: https://nestenius.se

Thanks for listening!

🎁 Exclusive Giveaway: 5 Free Expert Presentations

Bring valuable insights to your workplace!

✉ Email: tore@tn-data.se

Terms:

- **Region: Malmö -> Lund -> Båstad**
- **Duration**: 1-hour presentation, including Q&A
- **Topics:** Choose from my website
- **Limit:** One presentation per company
- **Deadline:** Email me before 15 December

Questions?
tore@tn-data.se