Dlouho běžící požadavky
jako zabiják výkonu web serveru

Michal Altair Valášek
development & security consultant

michal.valasek@altairis.cz
www.altair.blog

MVP Microsoft® Most Valuable Professional

altairis

altair.blog

altairis

Jak funguje web server?

- Aplikace běží ve worker procesu
  - Tradičně `w3wp.exe`
- Ten je různě šikanován web serverem
  - Určení identity procesu
  - Možnost omezení paměti, CPU time…
- Má omezený počet vláken
  - Vychází z počtu procesorových jader

altairis

Dlouho běžící požadavky

- Doba zpracování požadavku pokud možno do 100 ms
- Delší blokují dostupná vlákna
  - Další požadavky se řadí do fronty
- Mohou být ukončeny web serverem
  - Např. pokud si myslí, že běží příliš dlouho
- Nad celým procesem nemáme kontrolu

➔ **Je tedy lepší tyto požadavky zpracovávat asynchronně**

Asynchronní programování versus asynchronní architektura

altairis

Asynchronní programování

- Použití `async` a `await` v C#
- Typicky pro komunikaci s pomalými zdroji, jako například:
  - Disk
  - Databáze
  - Síť
- Rozhodně ho používejte, kdekoliv to jde
  - Od .NET Core jsou relevantní API async
  - Ta nová jsou už jenom asynchronní

altairis

Asynchronní architektura

- Někdy `async` a `await` nestačí, je třeba správně navrhnout celou architekturu aplikace
  - Frontend přijme job
  - Zařadí ho do fronty
  - Frontend zobrazuje stav jobu
  - Backend ho vyřídí, až na něj dojde řada
  - Frontend zobrazí výsledek
- Cokoliv trvá řádově sekundy musí být řešeno takto!

altairis

Frontend

- Standardní webová aplikace
- Přijímá požadavky od uživatele
- Sleduje stav fronty
  - Obyčejný autorefresh
  - JS callback
  - Web Sockets, SignalR apod.
- Zobrazí výsledek až je k dispozici

altairis

Backend

- Může běžet kdekoliv
  - V rámci procesu web serveru
  - V samostatném procesu na tomtéž serveru
  - Na samostatném serveru
  - V cloudu
    - Azure Web Jobs
    - Azure Functions
- Máme nad ním plnou kontrolu
  - Paralelizace požadavků
  - Prioritizace požadavků

Komunikace v rámci asynchronní architektury

altairis

Komunikace

- Typicky probíhá formou fronty (Queue)
- Výhody
  - Volné propojení (i různých platforem)
  - Možnost škálování
    - Vertikální (worker na samostatném serveru)
    - Horizontální (více workerů běží paralelně)
    - Podle zátěže, zejména v cloudu
  - Odolnost proti výpadku
  - Brání přetížení
- Nevýhody
  - Náročnější implementace
  - Závislost na QMS

altairis

Jak udělat frontu

- Databáze
- Soubory na disku
- Specializovaný software
  - MSMQ
  - RabbitMQ
  - Redis
- Cloudové služby
  - Azure Storage Queues
  - Azure Service Bus
  - Amazon SQS
- Event streaming
  - Azure Event Hubs
  - Apache Kafka

Background workers v ASP.NET Core

altairis

Backround Workers

- K dispozici od .NET Core 2.1
- Namespace `Microsoft.Extensions.Hosting`
  - Interface `IHostedService`
  - Třída `BackgroundService`
- Od .NET Core 3.0 je pro ně přímá podpora a šablona
  - Worker Service template

https://docs.microsoft.com/aspnet/core/fundamentals/host/hosted-services

altairis

Hosting background workerů

- V rámci procesu web serveru
  - Stačí zaregistrovat do IoC/DI kontejneru
- V samostatném procesu
  - Běžná konzolová aplikace
  - Windows služba
    - NuGet `Microsoft.Extensions.Hosting.WindowsServices`
  - Linux daemon
    - NuGet `Microsoft.Extensions.Hosting.SystemD`

demo

Background workers v ASP.NET Core

michal.valasek@altairis.cz

facebook.com/**rider.cz**
github.com/**ridercz**
twitter.com/**ridercz**
linkedin.com/in/**ridercz**

www.**altairis.cz**
www.**rider.cz**
www.**altair.blog**