

1

# Praćenje traga rada aplikacije

2016/17.12

## ➤ Motivacija

- Evidentirati pogreške tijekom rada (neovisno o dojavama korisnika)
- Detektirati uska grla aplikacije
- Prikupljanje ostalih informacija
  - npr. parametri pretrage, prijave korisnika
- ...

## ➤ Gdje evidentirati?

- Baza podataka
- Datoteka
- E-mail poruke, SMS
- Event Log na Windowsima
- ...

## ➤ Kako evidentirati?

- Na uniforman (centraliziran) način neovisan o odredištu traga

# Razine važnosti zapisa (1)

3

## ➡ Trace

- ➡ Informacija namijenjena programeru u cilju lakšeg rješavanja problema
  - ➡ npr. ispis svih parametara nekog postupka
- ➡ Bilježi male korake izvođenja programa
- ➡ Nije preporučljivo koristiti u produkciji
  - ➡ smije sadržavati osjetljive podatke
  - ➡ veća količina zapisa u odnosu na ostale razine

## ➡ Debug

- ➡ Informativna poruka u cilju otklanjanja pogrešaka
  - ➡ ispis određene vrijednosti kratkoročne koristi
- ➡ Slično kao Trace, ali rjeđe (i manje detaljno)
  - ➡ nije nužno namijenjena samo programeru
- ➡ Najčešće automatski isključeno iz produkcijske verzije

# Razine važnosti zapisa (2)

4

## ➤ Information

- Zapis trajnijeg karaktera koji služi za praćenje toka rada aplikacije
  - npr. informacija o posjetu određenoj stranici ili evidencija postavljenih kriterija pretrage

## ➤ Warning

- Za pogreške koje ne utječu na daljnji rad aplikacije, ali predstavljaju potencijalno opasne situacije te zahtijevaju naknadnu pažnju
  - npr. konfiguracijska datoteka ne sadrži traženu vrijednost, pa se koristi predodređena

# Razine važnosti zapisa (3)

5

## ➤ Error

- Služi za evidentiranje pogrešaka i iznimki koje se ne mogu obraditi
- Predstavljaju kritičnu pogrešku za određeni postupak, ali ne i za cijelu aplikaciju
  - npr. pogreška prilikom dodavanja novog podatka u bazu

## ➤ Critical (Fatal)

- Situacije koje uzrokuju prekid rada cijele aplikacije
  - npr. nedostatak prostora na disku, neispravne postavke za spajanje na bazu podataka, ...

➤ Trace < Debug < Information < Warning < Error < Critical

- Microsoft.Extensions.Logging
  - Podrška za različite pakete i odredišta zapisivanja tragova
  - Nekoliko ugrađenih implementacija
    - Npr. konzolni ispis
  - Moguće dodati vlastitu implementaciju
    - Vidi mapu Util/Logging iz oglednog primjera
- Zajedničko sučelje *ILogger* kao apstrakcija nad različitim implementacijama
- Koristi se tehnika *Dependency Injection*
  - Upravljači koji bilježe trag ovise o sučelju ILogger
  - ASP.NET stvara konkretni objekt temeljem podataka u Startup.cs


# Sučelje ILogger

7

- `IsEnabled(LogLevel logLevel)`
  - provjerava evidentira li konkretna implementacija zapise navedene razine
- `void Log<TState>(LogLevel logLevel, EventId eventId, TState state, Exception exception, Func<TState, Exception, string> formatter)`
  - evidentiranje zapisa određene razine i vrste događaja
    - Zapis (state) ne mora nužno biti string, već može biti bilo kojeg tipa
    - Vrsta događaja opisana strukturom `EventId`: Id i naziv
    - Uz zapis (state) može biti vezana određena iznimka (exception)
    - *formatter*: funkcija koja kreira string na osnovi zapisa i iznimke
- `IDisposable BeginScope<TState>(TState state)`
  - Služi za grupiranje više zapisa u jedan zajednički zapis
    - samo ako konkretna implementacija podržava


# Pokrate za postupak Log

8

- Statički razred Microsoft.Extensions.Logging.Logger s proširenjima za sučelje ILogger
- Nekoliko preopterećenih postupaka:
  - LogTrace, LogDebug, LogInformation, LogError, LogCritical
- Pojednostavljuje zapisivanje traga
  - Primjer:  Web \ Firma.Mvc \ Controllers \ DrzavaController.cs

```
public IActionResult Create(Drzava drzava){  
    logger.LogTrace(JsonConvert.SerializeObject(drzava));  
    ...  
    catch (Exception exc) {  
        logger.LogError(  
            "Pogreška prilikom dodavanje nove države: {0}",  
            exc.CompleteExceptionMessage()  
        );  
    }
```



- NLog kao jedan od alata za praćenje traga kompatibilan s Microsoft.Extensions.Logging
  - <http://nlog-project.org/>
- Omogućava više vrsta (različitih) spremišta i formata ovisno o razini zapisa
- Upute za dodavanje u ASP.NET Core projekt
  - [https://github.com/NLog/NLog.Web/wiki/Getting-started-with-ASP.NET-Core-\(csproj---vs2017\)](https://github.com/NLog/NLog.Web/wiki/Getting-started-with-ASP.NET-Core-(csproj---vs2017))
  - Umjesto apsolutne putanje u konfiguracijskoj datoteci za NLog koristiti relativnu putanju
    - Primjer:  Web \ Firma.Mvc \ nlog.config

```
<target xsi:type="File" name="allfile"
  fileName="logs\nlog-all-${shortdate}.log"
  layout="${longdate} | ${event-
properties:item=EventId.Id} | ${logger} | ${uppercase:${level}} | ${message}
${exception}" />
```

# Konfiguracijska datoteka za NLog

10

- Moguće definirati više različitih odredišta, a zatim pravila koja određuju gdje će sve pojedini zapis biti evidentiran

➤ Primjer:  Web \ Firma.Mvc \ nlog.config

```
<targets>
  <target xsi:type="File" name="allfile"
    fileName="logs\nlog-all-${shortdate}.log" ... />
  <target xsi:type="File" name="ownFile-web"
    fileName="logs\nlog-own-${shortdate}.log" ... />
  <!-- write to the void aka just remove -->
  <target xsi:type="Null" name="blackhole" />
</targets>
<rules>
  <logger name="*" minlevel="Trace" writeTo="allfile" />
  <!--Skip Microsoft logs and so log only own logs-->
  <logger name="Microsoft.*" minlevel="Trace"
    writeTo="blackhole" final="true" />
  <logger name="*" minlevel="Trace" writeTo="ownFile-web" />
</rules>
```

# Aktivacija NLoga

11

- U konstruktoru razreda Startup potrebno uključiti konfiguracijsku datoteku (potrebno NLogu radi vlastitih postavki)
- U postupku Configure koristeći na objektu tipa ILoggerFactory pozvati proširenje kojim se među postojeće *loggere* dodaje i NLog
- Primjer:  Web \ Firma.Mvc \ Startup.cs

```
public class Startup {  
    public Startup(IHostingEnvironment env) {  
        ...  
        env.ConfigureNLog("nlog.config");  
    }  
    public void Configure(IApplicationBuilder app,  
                          IHostingEnvironment env,  
                          ILoggerFactory loggerFactory,  
                          IServiceProvider serviceProvider) {  
  
        loggerFactory.AddNLog();  
        app.AddNLogWeb();  
    }  
}
```

# Stranica za prikaz svih zapisa

12

- U ogleđnoj aplikaciji evidentiraju se poruke vezane za dodavanje i brisanje države.
- Izrađena stranica koja za odabrani datum pronalazi datoteku s korisničkim poruka i ispisuje ih na ekranu

→ localhost:50051/ViewLog/Show?dan=2017-05-19

## Zapisi u log datoteci za dan 19.05.2017.

Show  entries Search:

Vrijeme	Id	Upravljač	Level	Message	Url	Action
02:44:35	0	Firma.Mvc.Controllers.DrzavaController	TRACE	{"OznDrzave":"AD","NazDrzave":"Test","Iso3drzave":"123","SifDrzave":123,"Mjesto":[]}>	http://localhost/Drzava/Create	Create
02:44:36	0	Firma.Mvc.Controllers.DrzavaController	ERROR	Pogreška prilikom dodavanje nove države: An error occurred while updating the entries. See the inner exception for details.Violation of PRIMARY KEY constraint 'pk_Drzava'. Cannot insert duplicate key in object 'dbo.Drzava'. The duplicate key value is (AD).The statement has been terminated. >	http://localhost/Drzava/Create	Create
02:44:43	0	Firma.Mvc.Controllers.DrzavaController	TRACE	{"OznDrzave":"A","NazDrzave":"Test","Iso3drzave":"123","SifDrzave":123,"Mjesto":[]}>	http://localhost/Drzava/Create	Create
02:44:43	0	Firma.Mvc.Controllers.DrzavaController	INFO	Država Test dodana. >	http://localhost/Drzava/Create	Create
02:44:55	0	Firma.Mvc.Controllers.DrzavaController	INFO	Država Test1 uspješno obrisana >	http://localhost/Drzava/Delete	Delete

Showing 1 to 5 of 5 entries Previous 1 Next

# Odabir datuma za prikaz zapisa

13

## ➤ Odabir datuma korištenjem jQueryUI datepickera

➤ Odabrani datum u formatu koji omogućava automatsko povezivanje u datumski tip u akciji Show

➤ GET varijanta kako bi datum bio dio adrese, a ne tijela zahtjeva

➤ Primjer:  Web \ Firma.Mvc \ Views \ ViewLog \ Index.cshtml

```
<form asp-action="Show" method="get">
...
    <input name="dan" class="form-control datum" />
...
    <button type="submit"
        class="btn btn-xs btn-primary save" ...
</form>
@section scripts{
    ...
    <script>
        $(".datum").datepicker({
            dateFormat: "yy-mm-dd"
        });
    </script>
}
```

# Provjera postoji li datoteka za odabrani dan

14

► Primjer:  Web \ Firma.Mvc \ Controllers \ ViewLogController.cs

```
public async Task<IActionResult> Show(DateTime dan) {  
    ViewBag.Dan = dan;  
    List<LogEntry> list = new List<LogEntry>();  
    string format = dan.ToString("yyyy-MM-dd");  
    string filename = $"logs/nlog-own-{format}.log";  
    if (System.IO.File.Exists(filename))  
    {  
        ...  
    }  
}
```

# Čitanje datoteke po retcima

15

- Veza na datoteku korištenjem razreda FileStream koji se zatim pretvara u StreamReader

- Omogućava čitanje redak po redak

- Primjer:  Web \ Firma.Mvc \ Controllers \ ViewLogController.cs

```
public async Task<IActionResult> Show(DateTime dan) {  
    ...  
    String previousEntry = string.Empty;  
    using (FileStream fileStream = new FileStream(filename,  
        FileMode.Open, FileAccess.Read)) {  
  
        using (StreamReader reader = new StreamReader(fileStream)) {  
            string line;  
            while ((line = await reader.ReadLineAsync()) != null) {  
                ...  
            }  
        }  
    }  
}
```

- Poruka nekog zapisa mogla se nalaziti u više redaka

- Novi zapis prepoznaje se tako da počinje s datumom te se u tom trenutku stari dodaje u listu (vidi ostatak programskog koda)

# Model za prikaz zapisa

16

► Primjer:  Web \ Firma.Mvc \ ViewModels \ LogEntry.cs

```
public class LogEntry {
    public DateTime Time { get; set; }
    public int Id { get; set; }
    public string Controller { get; set; }
    public string Level { get; set; }
    public string Message { get; set; }
    public string Url { get; set; }
    public string Action { get; set; }
    internal static LogEntry FromString(string text) {
        string[] arr = text.Split('|');
        LogEntry entry = new LogEntry();
        entry.Time = DateTime.ParseExact(
            arr[0], "yyyy-MM-dd HH:mm:ss.ffff", null);
        entry.Id = int.Parse(arr[1]);
        entry.Controller = arr[2];
        entry.Level = arr[3];
        entry.Message = arr[4];
        entry.Url = arr[5].Substring(5); //url:
        entry.Action = arr[6].Substring(8); //action:
        return entry;
    }
}
```



# Kreiranje vlastitog Loggera

17


## ➤ Napisati vlastitu implementaciju sučelja ILogger

### ➤ Konstruktor kreirati po želji tako da prima sve potrebne parametre

➤ Npr. Connection string, service provider za DI, funkcija kojom se određuje evidentira li se pojedina razina...

### ➤ Primjer: Web \ Firma.Mvc \ Util \ Logging \ FirmaLogger.cs


```
public class FirmaLogger : ILogger {
    private IServiceProvider serviceProvider;
    private Func<LogLevel, bool> filter;
    public FirmaLogger(IServiceProvider serviceProvider,
                       Func<LogLevel, bool> filter) {
        this.filter = filter;
        this.serviceProvider = serviceProvider;
    }
    public IDisposable BeginScope<TState>(TState state) {
        return null; //ne podržava scope...
    }
    public bool IsEnabled(LogLevel logLevel) { ... }
    public void Log<TState>(LogLevel logLevel, eventId, TState
state, Exception exception, Func<TState, Exception, string> formatter)
```

- Vlastitu implementaciju ILoggera aktivira vlastita implementacija sučelja ILoggerProvider
  - Konstruktor po želji s parametrima potrebnim za instanciranje implementacije ILoggera
  - Primjer:  Web \ Firma.Mvc \ Util \ Logging \ FirmaLoggerProvider.cs

```
public class FirmaLoggerProvider : ILoggerProvider {  
    private IServiceProvider serviceProvider;  
    private Func<LogLevel, bool> filter;  
    public FirmaLoggerProvider(IServiceProvider serviceProvider,  
                               Func<LogLevel, bool> filter) {  
        this.filter = filter;  
        this.serviceProvider = serviceProvider;  
    }  
    public ILogger CreateLogger(string categoryName) {  
        return new FirmaLogger(serviceProvider, filter);  
    }  
  
    public void Dispose() { }  
}
```

# Aktiviranje vlastite implementacije praćenje traga (1)


19

- Po uzoru na ostale implementacije napraviti proširenje nad sučeljem *ILoggerFactory*
  - Parametri proširenja po želji
    - Sve što je potrebno vlastitom *ILoggerProvideru*
  - Dodaje vlastiti *ILoggerProvider* u skup postojećih
  - Vraća referencu na primljeni *ILoggerFactory*
  - Primjer:  Web \ Firma.Mvc \ Util \ Logging \ FirmaLoggerExtensions.cs

```
public static class FirmaLoggerExtensions {  
    public static ILoggerFactory AddFirmaLogger(  
        this ILoggerFactory factory,  
        IServiceProvider serviceProvider,  
        Func<LogLevel, bool> filter = null)  
    {  
        factory.AddProvider(  
            new FirmaLoggerProvider(serviceProvider, filter));  
        return factory;  
    }  
}
```

# Aktiviranje vlastite implementacije praćenje traga (2)

20

- ➡ Može postojati više istovremenih *logger*a
- ➡ Aktivirati vlastiti korištenjem prethodno napisanog proširenja
- ➡ Primjer:  Web \ Firma.Mvc \ Startup.cs

```
public class Startup {  
    public void Configure(IApplicationBuilder app,  
        IHostingEnvironment env,  
        ILoggerFactory loggerFactory,  
        IServiceProvider serviceProvider) {  
  
        loggerFactory.AddNLog();  
        app.AddNLogWeb();  
  
        loggerFactory.AddFirmaLogger(serviceProvider,  
            level => level >= LogLevel.Warning);  
        ...  
    }  
}
```