

# **.NET APPLICATIONS PRODUCTION PROFILING**

Rail Sabirov

# HOW WE ARE PROFILING ON DEV ENVIROMENT?

1. run some cool profiler!
2. make some load
3. analyze results in beautiful profiler UI

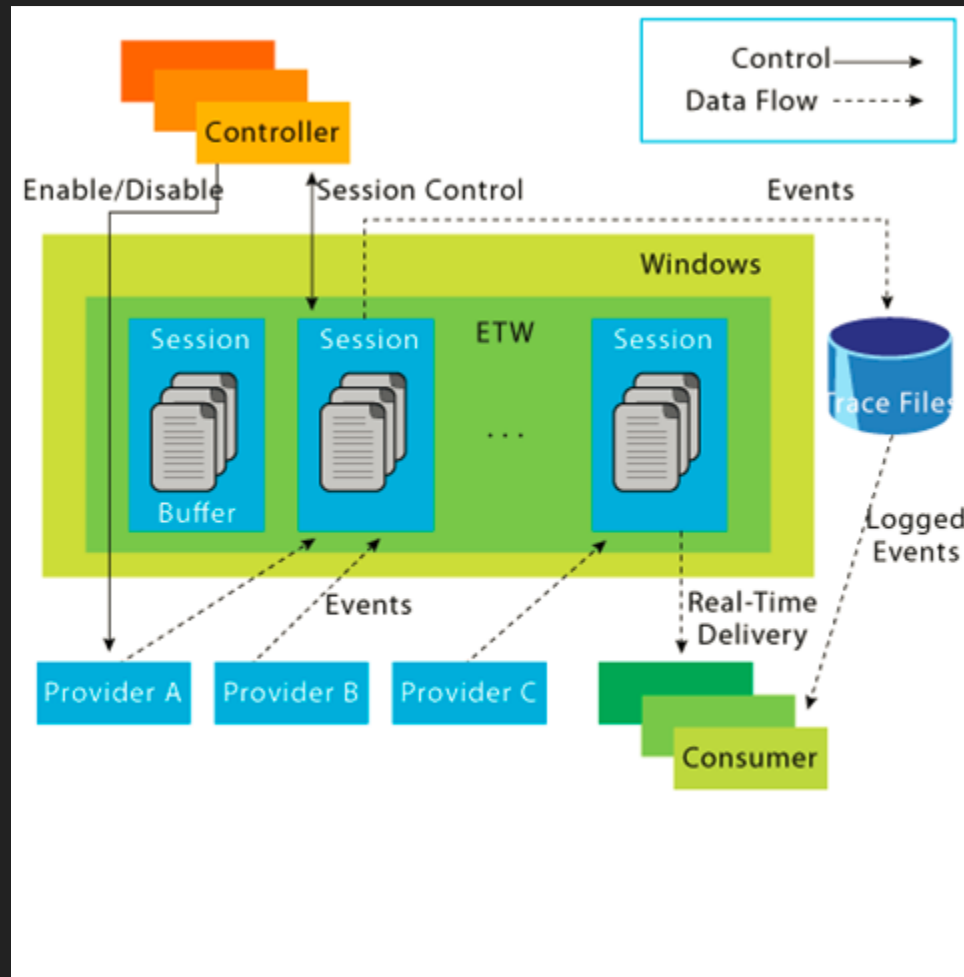
# WHY PRODUCTION PROFILING HARDER?

- we can not reproduce prod issue on dev environment
- we should not affect running instance (stop or slow down)
- we have to collect environmental information (cpu, memory, disk io, hard faults, network io...)
- we can not install profiler

# EVENT TRACING FOR WINDOWS (ETW)

- Symantic logging (each event and source has schema)
- Fast if enabled
- Doesn't affects performance if disabled
- Can drop events if "performance not enough"
- Used internally in Windows
- Used in .NET CLR (provides detailed info about CLR, JIT, GC...)
- Introduced on Windows 2000 (Windows Vista)

# ETW ARCHITECTURE OVERVIEW



# ETW COMPONENTS

## **event provider**

writes events to ETW sessions (it can be any user-mode application, managed application, driver etc)

## **event consumers**

application that reads log files or listens to a session for real time events and processes them

## **controller**

starts and stops ETW sessions and enables providers to them.

## **event trace session**

actual logging and buffering on separate kernel thread per session

**DEMO 0**

**PERFVIEW TOOL**

**OVERVIEW**

# HOW TO CREATE EVENTS SOURCE?

just create 100 line xml schema

```
<instrumentationManifest
  xmlns="http://schemas.microsoft.com/win/2004/08/events"
  xmlns:win="http://manifests.microsoft.com/win/2004/08/windows/events"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
>

  <instrumentation>
    <events>
      <provider ...>
        <channels>
          <importChannel .../>
          <channel .../>
        </channels>
        <levels>
          <level .../>
        </levels>
        <tasks>
          <task .../>
        </tasks>
        <opcodes>
          <opcode .../>
        </opcodes>
        <keywords>
          <keyword .../>
        </keywords>
        <filters>
          <filter .../>
        </filters>
        <maps>
          <valueMap ...>
            <map .../>
          </valueMap>
          <hitMan ...>
```



# EASIER WAY FOR CREATING EVENTS SOURCE

```
class MyEventSource : EventSource
{
    public void Message(string message)
    {
        WriteEvent(1, message);
    }

    public void OrderAccepted(int orderId)
    {
        WriteEvent(2, orderId);
    }
}
```

**DEMO 1, 2, ...**

# BEST PRACTICES

- Use primitive types if possible
- Use IsEnabled function if calculation of values for event takes memory/cpu
- Create separate EventSource for highly detailed logging

# BEST PRACTICES

Check ConstructionException at least on Debug configuration

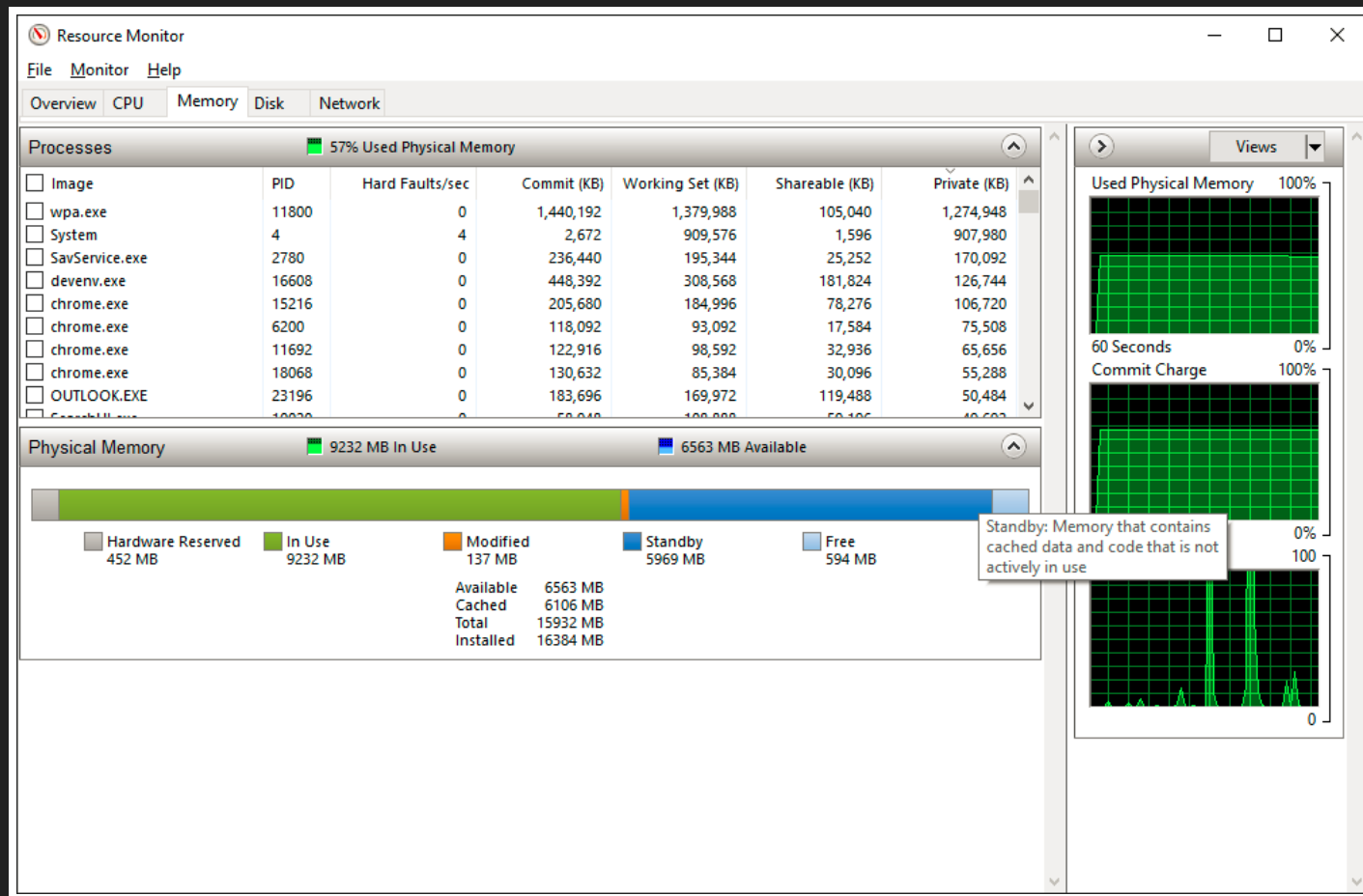
```
#if DEBUG
    if (MinimalEventSource.Log.ConstructionException != null)
        throw MinimalEventSource.Log.ConstructionException;
#endif
```

# WHO ELSE USES ETW?

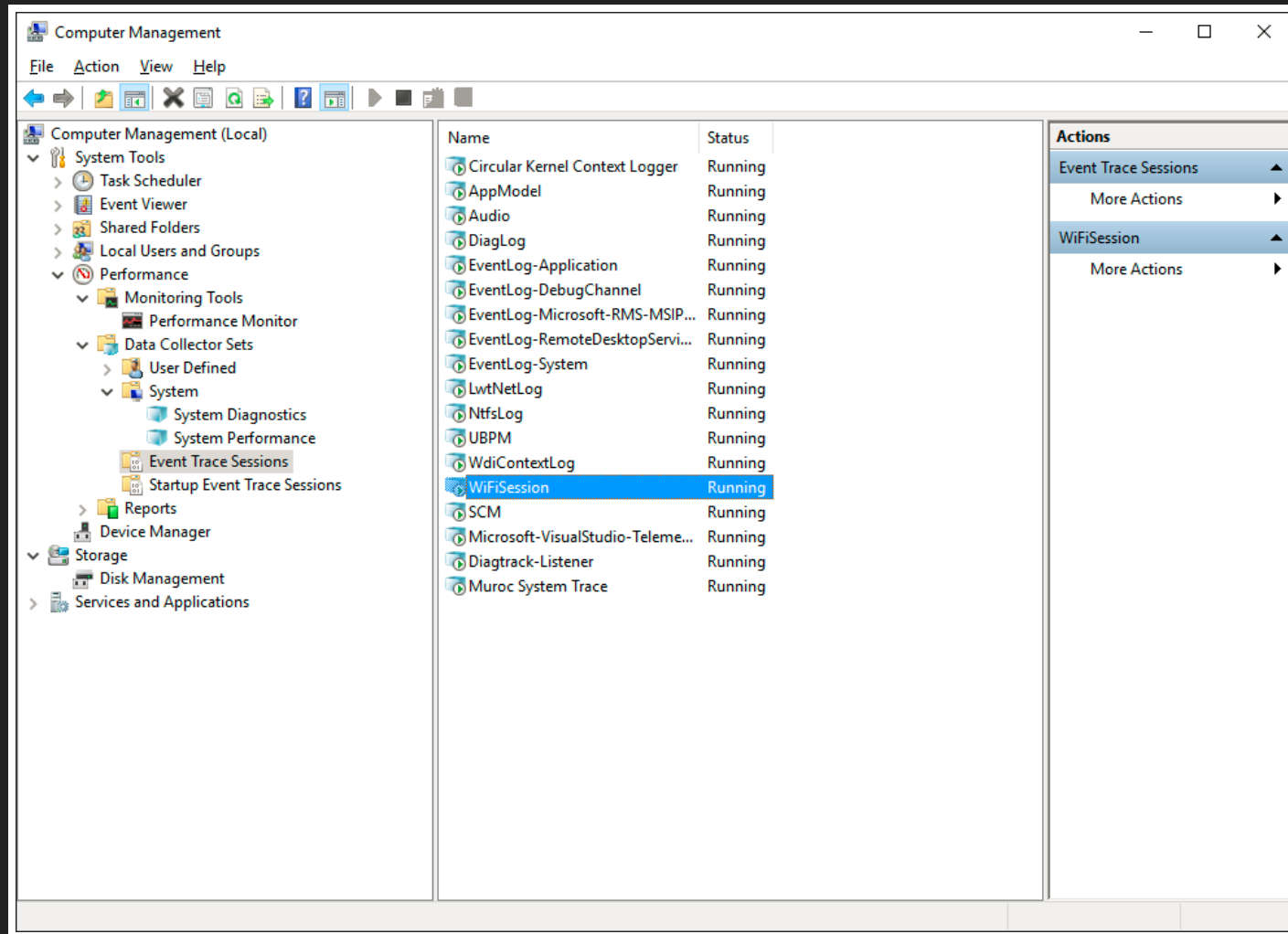
Process Monitor - Sysinternals: www.sysinternals.com						
File Edit Event Filter Tools Options Help						
Time of Day	Process Name	PID	Operation	Path	Result	Detail
10:34:07.5485459 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: HandleTags, HandleTags: 0x0
10:34:07.5485784 PM	Explorer.EXE	8936	RegOpenKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Desired Access: Maximum Allowed, Granted Access: Read
10:34:07.5486195 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: Name
10:34:07.5486575 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: HandleTags, HandleTags: 0x0
10:34:07.5486977 PM	Explorer.EXE	8936	RegOpenKey	HKCU\Software\Classes\TypeLib\{EA39B853-...	NAME NOT FOUND	Desired Access: Maximum Allowed
10:34:07.5487294 PM	Explorer.EXE	8936	RegEnumKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Index: 0, Name: 0
10:34:07.5487580 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: Name
10:34:07.5487905 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: HandleTags, HandleTags: 0x0
10:34:07.5488299 PM	Explorer.EXE	8936	RegOpenKey	HKCU\Software\Classes\TypeLib\{EA39B853-...	NAME NOT FOUND	Desired Access: Maximum Allowed
10:34:07.5488542 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: HandleTags, HandleTags: 0x0
10:34:07.5488919 PM	Explorer.EXE	8936	RegOpenKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Desired Access: Maximum Allowed, Granted Access: Read
10:34:07.5489184 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: Name
10:34:07.5489487 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: HandleTags, HandleTags: 0x0
10:34:07.5489945 PM	Explorer.EXE	8936	RegOpenKey	HKCU\Software\Classes\TypeLib\{EA39B853-...	NAME NOT FOUND	Desired Access: Maximum Allowed
10:34:07.5490176 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: HandleTags, HandleTags: 0x0
10:34:07.5490497 PM	Explorer.EXE	8936	RegOpenKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Desired Access: Maximum Allowed, Granted Access: Read
10:34:07.5490757 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: Name
10:34:07.5491027 PM	Explorer.EXE	8936	RegQueryKey	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Query: HandleTags, HandleTags: 0x0
10:34:07.5491527 PM	Explorer.EXE	8936	RegOpenKey	HKCU\Software\Classes\TypeLib\{EA39B853-...	NAME NOT FOUND	Desired Access: Maximum Allowed
10:34:07.5491792 PM	Explorer.EXE	8936	RegQueryValue	HKCR\TypeLib\{EA39B853-5769-4937-8ECE-...	SUCCESS	Type: REG_SZ, Length: 88, Data: C:\Windows\System32\UIAutomationCore\F
10:34:07.5493443 PM	Explorer.EXE	8936	CreateFile	C:\Windows\System32\UIAutomationCoreRes...	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: Synchronous IO N
10:34:07.5494482 PM	Explorer.EXE	8936	QueryNetwork...	C:\Windows\System32\UIAutomationCoreRes...	SUCCESS	CreationTime: 11-Feb-16 10:17:40 PM, LastAccessTime: 11-Feb-16 10:17:40 F
10:34:07.5495329 PM	Explorer.EXE	8936	ReadFile	C:\Windows\System32\UIAutomationCoreRes...	SUCCESS	Offset: 0, Length: 64, Priority: Normal
10:34:07.5495863 PM	Explorer.EXE	8936	ReadFile	C:\Windows\System32\UIAutomationCoreRes...	SUCCESS	Offset: 192, Length: 4
10:34:07.5496223 PM	Explorer.EXE	8936	ReadFile	C:\Windows\System32\UIAutomationCoreRes...	SUCCESS	Offset: 196, Length: 20
10:34:07.5496552 PM	Explorer.EXE	8936	ReadFile	C:\Windows\System32\UIAutomationCoreRes...	SUCCESS	Offset: 456, Length: 40

Showing 292,275 of 949,144 events (30%)      Backed by virtual memory

# WHO ELSE USES ETW?



# WHO ELSE USES ETW?



# TOOLS

- PerfView (<http://aka.ms/PerfView>)
- Logman.exe - command-line controller ([Technet article](#))
- tracerpt.exe - a general consumer tool ([Technet article](#))
- Windows Performance Analyzer ([WPA](#)) is a tool that creates graphs and data tables of ETW
- [Xperf](#) actions are trace processing components that collate event information to produce text reports
- Even google has own ETW tool :)

<https://github.com/google/UIforETW>

There are lot of tools around ETW. Collectors and event analyzers.



# USEFULL LINKS

- [Channel9 PerfView tutorial](#)
- [Blog posts about ETW](#)
- [Microsoft.Diagnostics.Tracing.Logging repo](#)
- [The TraceEvent Library Programmers Guide](#)
- [Summary of ETW support in .NET](#)
- [ETW Events in the Common Language Runtime](#)

# WHAT REMAINS OUT OF SCOPE OF THIS PRESENTATION?

- Memory profiling
- Controlling sessions using .NET
- Collecting events using .NET
- Cool tools around ETW

# QUESTIONS?