



The Guide to .NET Profilers

WHAT YOU NEED TO KNOW

At Stackify, we do a lot of things around application performance, and have actually written a couple profilers ourselves. We know a lot about code profiling for .NET — and after reading this guide, you will too.



Build**better** / by /



Stackify

Table of Contents

1. Introduction

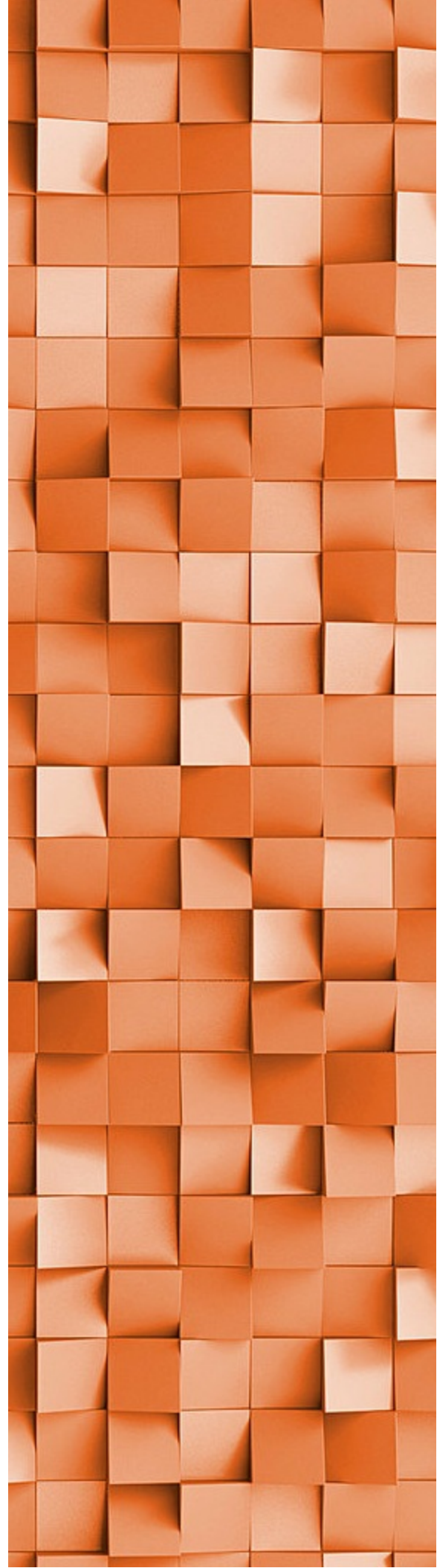
2. The 3 Different Types of .NET Profilers

3. Traditional .NET Profilers

4. Lightweight .NET Profiler/Transaction Tracing Tools

5. Application Performance Management (APM) Tools

6. Retrace what your code is doing...



Introduction

.NET profilers are a developer's best friend when it comes to optimizing application performance, and are especially critical when doing low level CPU and memory optimizations. They are just one of the tools a developer needs to have in their toolbox.

What are profilers?

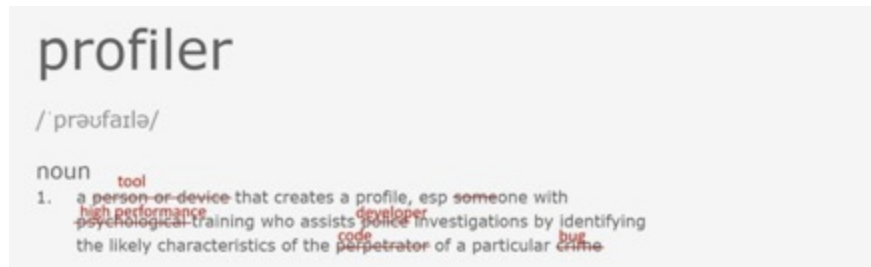
Here's what the Internet says:

"A person or device that creates a profile, especially someone with psychological training to assist police investigations with identifying the likely characteristics of the perpetrator of a particular crime."

It's kind of a weird definition, but with a few changes it works perfectly:

All right, well, it sort of works.

In short, profilers are used by developers to help them find performance problems, without requiring a change in code.



They can:

- Answer questions like how many times each method in your code is called and how long each of those methods take
- Track things like memory allocations and garbage collection
- Track key methods in your code so you can understand how often SQL statements and web services are called
- Track web requests and then train those transactions to understand the performance of transactions within your code
- Track all the way down to each individual line of code or the CPU instructions

The 3 Different Types of .NET Profilers

Profilers are powerful tools for measuring and improving the performance of your apps in development and production. All are very valuable but serve relatively different purposes and do different types of performance profiling. You should get familiar with all three types.

Traditional .NET Profilers

These track process memory usage, time spent per line of code, and frequency of method calls. They include products like Visual Studio .NET profiler, [Red Gate ANTS](#), [dotTrace](#), [JustTrace](#), SciTech, and YourKit.

These tools are a lifesaver when you need them, but they are very resource intensive and definitely slow you down.

The vast majority of developers have never (or very rarely) used these types of profilers. These simply aren't needed day-to-day for apps that a lot of developers create.

Traditional usage scenarios for a .NET profiler:

- **High memory usage.** Profilers are extremely powerful when it comes to tracking down memory leaks and optimizing memory usage.
- **CPU usage is out of control.** If your server CPU is extremely high and you have no idea why, a profiler may be your last resort for figuring out why.
- **Proactive performance tuning.** Optimizing CPU usage for some apps is a never ending job.

A standard .NET profiler works by using the .NET CLR profiling interface. This allows profiling of the .NET MSIL bytecode at a low level in order to understand each operation your code performs. It shows you the “hot path” within your code to see which methods are using the most CPU. You can then typically drill down to see which specific lines of code in your app are using the most CPU. This can be a huge help when you need to urgently find a problem.

We proactively use the Visual Studio Profiler and ANTS to tune the performance of our Windows monitoring agent. Our goal is to add as little overhead as possible on the servers of our customers. We have also had to use them to chase down some weird memory leaks.

Lightweight .NET Profiler/Transaction Tracing Tools

Lightweight profilers are geared more toward tracking the high level performance of your app. They help you understand total page load time, which database calls were executed, etc. These tools are designed to help developers every single day. They don't have a huge performance impact on your code, so they can always be on.

There are three primary tools available to .NET developers that all work as **ASP.NET profilers**. They are all very different in how they are implemented, how they work, and the types of information they can provide.

Glimpse

- Installed into your app and requires many configuration changes and NuGet packages
- Open source project now led by Microsoft
- Does not use the .NET CLR profiler
- Utilizes an extension and packages framework to add support for various app dependencies and technologies
- Requires some code changes
- Only works with web apps

MiniProfiler

- Installed into your app as a lightweight tracing tool
- Does not use the .NET CLR profiler
- Database calls can be tracked by changing your code to wrap your SQL connections. You can also change your code to report additional steps to include in the pseudo profile traces
- Only works with web apps
- Requires a lot of code changes

Stackify Prefix

- Installs on a developer's workstation outside of your app
- Based on the .NET CLR profiler and uses the same technology that powers Stackify's APM product for monitoring server apps
- Automatically tracks the performance of 30+ common .NET frameworks and libraries
- Can be used to view exceptions, logs, and much more
- Can be extended to profile any method in your code
- Works with non-web apps
- Requires no code or configuration changes to work



Lightweight profilers are **designed around individual web requests or transaction tracing**. This makes them very useful for tracking how long specific web requests take and why. They can save a ton of time compared to writing a bunch of custom logging or debugging your code, and they can put a lot of good information at your fingertips. **Every developer should have one of these tools in their toolbox**

Prefix is an amazing free tool. It is extremely easy to install and **it works** with no headaches, code changes, or config changes. It is very lightweight and designed to be used every day. We have many users who leave it open on their second monitor all the time.

Application Performance Management (APM) Tools

APM is largely an industry- or vendor-created term for anything that has to do with managing or monitoring the performance of your code, application dependencies, transaction times, and overall user experiences.

One of the great things about APM products is they collect enough details to quickly identify and resolve most common application problems. Products like Stackify APM can show the exact database queries, logging, exceptions, web service calls, and so much more to understand how to fix bugs or improve application performance.

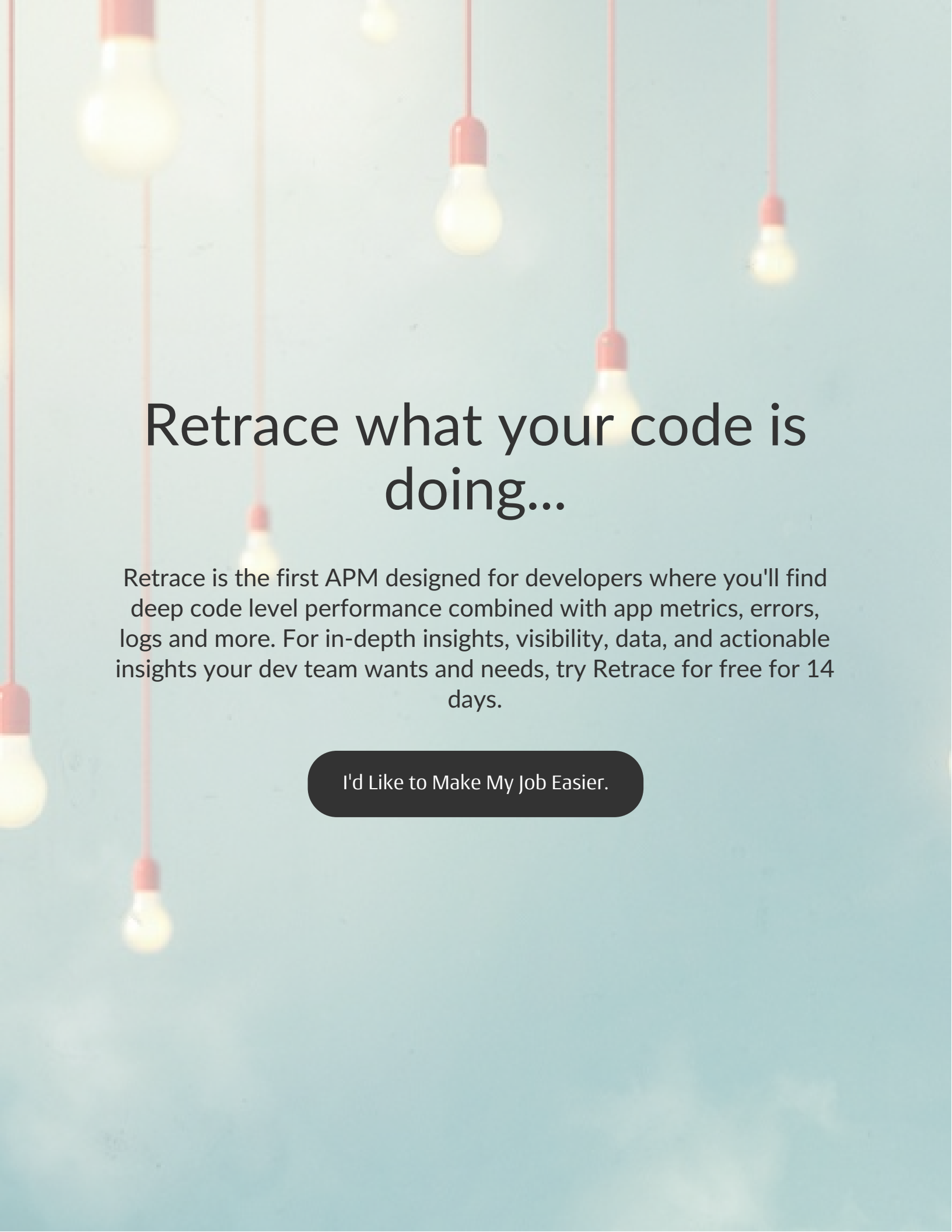
APM solutions can help identify common application problems quickly:

- Track overall application usage to understand spikes in traffic
- Find slowness or connection problems with application dependencies
- Identify slow SQL queries
- Find highest volume and slowest web pages or transactions

Some other tools monitor based on server and application metrics, not code level performance. These are sometimes referred to as application performance monitoring solutions. Knowing your server CPU or the average response of your webserver is important and helpful, but APM tools aim to go way deeper.



Most APM solutions are very complex to configure and use - they end up being expensive traffic lights and dashboards. Some vendors have put a huge focus on **making their products affordable** and very easy to use so they can be available to development and operations teams of all sizes.

The background of the entire page is a light blue wall with several glowing yellow light bulbs hanging from red cords. The bulbs are at various heights and positions, creating a soft, ambient lighting effect.

Retrace what your code is doing...

Retrace is the first APM designed for developers where you'll find deep code level performance combined with app metrics, errors, logs and more. For in-depth insights, visibility, data, and actionable insights your dev team wants and needs, try Retrace for free for 14 days.

I'd Like to Make My Job Easier.