Java Instrumentation

What is Instrumentation

Instrumentation is the act or the process of adding bytecode to your existing Java bytecode during runtime for the purpose of gathering data.

Now who wants to do this and why is this useful?

Many tools like Jacoco for analyzing code path execution for code coverage, profilers so that you can see the hotspots or memory leaks within your application, or even realtime monitoring tools like Datadog or NewRelic. **These tools need to add the bytecode to your application code without touching your source code because they do not have access to them in order to provide the functionality.**

> By using instrumentation it can provide you a way to transform bytecode of classes during runtime as they are being executed by the JVM. Developers can intercept, modify, and analyze the bytecode to give it extra functionality during runtime without modifying the source code.

What is Java Agent

A Java Agent is a Java program that can be attached to a running Java Application to modify its behavior, and it the agent uses the Instrumentation API to intercept and transform classes. They are packaged and attached by adding the `-javaagent` commandline argument.

We are not going to be diving into how to create a Java Agent, however, if you would like to learn more about it you can visit Link to learn more about it!

What is Telemetry

Telemetry (noun) is the process of collecting and analyzing data from sources to get insights about its performance.

In the IT space, telemetry means collecting data from deployed softwares and help you get deeper insights about how your application is performing in realtime.

Telemetry and monitoring are sometimes used interchangeably but they have a slight differences:

- **Monitoring**: A more narrower scope, measures metrics in order to detect potential issues and take proper action to avoid impacts
- **Telemetry**: Collects and analyses data for a broader range of purposes, which can include monitoring but can be used more than that

What is NewRelic

NewRelic is a platform that you can send your telemetry data to to provide you monitoring / alerting / and querying tool on your data.

NewRelic provides a Java agent that can be use to instrument your application code. You can download it and then just package it along with your application and then when you run your Java application use the `-javaagent` commandline argument for it.

What is Open Telemetry

Okay what is Open Telemetry? **It is an open source vendor-agnostic observability tool** that you can use to basically instrument your application much like a normal agent does.

It standarizes the telemetry data colletion and exports. Provides an unified set of APIs, libraries, agents, and instrumentation to capture traces, metrics, and logs from your application and infrastructure.

If you use OTEL you're free to choose the backend vendor that you send the data to, it doesn't have to be NewRelic, it could be DataDog or another SaaS.

Information about Observability

What does observability mean? Having observability means how well the internal working of the system can be inferred by looking at the output of the system. "You know how well an application is working by looking at the outputs without knowing the intricate inner working of every line of code".

Three Pillars of Observability

The three pillars of observability are metrics, logs, and traces (MELT).

Logs

They are produced by the application, these are great for tracing a single event situation. Logs are useful for unplanned researches and unique situations. However they are costly to store as your application grows.

Metrics

They are numerical data produced by the application, realtime example would be the speedometer in a car. Metrics can be used to build graphs and monitoring alerts for your application. They are efficient to store and cost effetive compared to logs. However, coming up with the right metrics to be produced by your system is a design challenge of it's own.

> Initially, logs and metrics were used predominantly as the means of observability. For example for monitoring a car, you can see the speedometer for speed, the oil gauge for the amount of fuel leftover, and that can be observed through metrics. However, if your car broke down and you send it to a mechanics, the mechanics is probably going to ask you couple questions as to what happend to the car before diagnosing further.

Traces

Distributed Tracing

Distributed tracing tracks request across different complex systems, it provides you with end to end visibility of a request as it moves through microservices / API / other components. It help teams by identifying the issue in real time.

Distributed tracing is especially useful in architectures built on top of microservices.

Context Propagation

The trace ID which can help correlate request across different system is pass through the request header as it makes its way through the network.

Revision #6
Created 2025-09-10 03:03:14 UTC by Tamarine
Updated 2025-09-11 05:12:14 UTC by Tamarine