Visual J++

Adrian Kosmaczewski

2021-02-12

Once upon a time, there was a programming environment made by Microsoft called Visual J++. It was their attempt to do with Java what they had done with JScript before, and to be honest, it was quite cool. You could compile and run Java code on Windows with a very good IDE - this was 5 years before IntelliJ released IDEA! It generated much faster binaries than what the official Java compiler from Sun produced. Developers could access functionality inside of packages starting with the `microsoft.` name, but that of course that kind of broke the whole point of Java which is to make cross-platform stuff that you only write once and then you run everywhere.

So at some point Sun got fed up and sued Microsoft, who promptly threw Visual J++ away and came up with something more interesting, which was C# and .NET, which was essentially like Java but with one major difference: it could only run in Windows. Which in 2000 was still an understandable proposition.

Both C# and Java have very similar "Hello, World!" implementations:

```java
public static void main(String[] args) {
    System.out.println("Hello, World!");
}
```

```csharp
public static void Main(string[] args)
{
    Console.WriteLine("Hello, World!");
}
```

And we all know that "Hello, World!" is the golden standard for programming language comparison. Show me your "Hello, World!" and I will extrapolate enough facts for a whole subreddit.

From a historical point of view, it is quite obvious that the major difference between them stem from just one person: Anders Hejlsberg, who was also the creator of Turbo Pascal, Delphi, Visual J++, and would later bring TypeScript to the world. Talk about a resumé. And since in Pascal one uses `PascalCased` entities pervasively, not only for classes but also for methods, so C# came to be.

Well, there is also the position of the opening curly bracket. That is the second major difference.

The dichotomy between Java and .NET was so strong, that even a popular website such as TheServerSide.com begat TheServerSide.net as a way to control the clash of matter and antimatter.

After the release of .NET, Microsoft came up with a successor of Visual J++, called J# which as the name suggests was a Java compiler for the .NET runtime – but it was discontinued a few years later.

See, the only thing that everyone wanted Microsoft to do with .NET in 2002 was to make Visual Basic 6 compatible with Visual Basic.NET. And *that* is precisely the one feature that enterprise developers never got. Even worse, both languages were largely incompatible with each other, to the extent that even copy-pasting code from one to the other did not work. Microsoft was very vocal in that you could call COM components from within both languages, but the truth is that nobody made COM components anyway; most apps were written in VBScript or Visual Basic 6, and neither of these had a decent migration path towards .NET. That is what I call spitting on the face of your biggest customers.

Or, as Ballmer would say, developers, developers, developers.

So the beginning of the 21st century was marked by yet another Great Rewriting™®©. It happens every so often. Well, not that often for COBOL or Fortran, but still.

But to be fair, starting with the second version of C# in 2005, Java and C# started to diverge a lot. Which is an euphemism to say that Java stagnated and C# evolved. So much that many Java developers started to use anything but Java (the language) in their Java (the runtime) applications; Clojure for functional programming purists, Scala for backend services, Kotlin for Android applications, JRuby for scripting and Groovy for build systems. Heck, at some point you could even run PHP inside the JVM if you really wanted to.

Of course there is a team of developers working overtime in the basement of a bank in Zurich, still maintaining the Java applets forming the core of an intranet only compatible with Internet Explorer 6 that, for some weird reason, refuses to die in dignity. If you are one of those developers, reading this note during your coffee break, please shout. We will come to rescue you.

Java got lots of backlash in the past 15 years, starting with Steve Yegge, who has been very vocal against it, and now apparently loves Kotlin. But just like in fashion, Java is kind of cool again lately, in particular thanks to Quarkus, which is basically a Kubernetes-ready Java runtime.

And by putting the word “Java” next to “Kubernetes”, Red Hat won the buzzword war. Which is precisely what IBM expected for the 34 billion USD price tag.