Alternative JVM Langu for JA Pr

ISTOCKPHOTO

Wiring the 783rd Spring Bean. Creating the 2,162nd getter-and-setter JavaBean pair. Using the collection utility class to filter a list and losing count of all the curly brackets, angle brackets, and parentheses. Some days, life on a long-lived Java project can be repetitive and tiresome. This is because some concepts in Java require a lot of code and syntax to express, earning the language a reputation as being verbose.

The good news is that in 2011 there are several languages that run on the Java Virtual Machine (JVM), work with existing Java libraries, and have a successful history of deployment in production. This means these languages are stable enough for businesses to trust them for important systems, just like Java. Some of the most widely used alternative JVM languages are Groovy, Scala, JRuby, and Clojure. Each of these languages has commercial support options and a growing community of talented developers, which means you won't be alone if you pick a language and need help. Companies such as Akamai, Foursquare, JBoss, LinkedIn, Netflix, SAP, and Twitter are using these languages.

But choosing another language isn't just about finding some variety in your daily work. Some languages let you express certain concepts with less code than Java requires. For example, all the alternative languages listed in this article offer closures and functions as first-class objects. These require far less code than Java's equivalent, anonymous inner classes. Other languages offer radically different approaches to problems like concurrency. Scala and Clojure provide programming models using actors or software transactional memory that can be simpler to reason about for some use cases than typical Java shared-state multithreaded code. For example, to maintain a variable that may be accessed and changed by several different threads, a safe Java solution requires a careful analysis and synchronization strategy to avoid deadlocks or logic errors. However, with a software transactional memory, programmers access and modify variables inside atomic transactions (similar to how databases are modified) and let the transaction manager handle the complexities of concurrent access.

Here are some stories about how developers have integrated alternative JVM languages in their Java projects.

ages
VA
ojects

by Daniel Wellman

Groovy

Groovy is a dynamic language that offers a gentle transition from Java. Most Java code is valid Groovy code, which means a Java programmer can ease his way into the more dynamic features of the language. Runtime metaprogramming, or writing code that can write code (imagine an object representing a database record that can connect to the database at runtime, inspect its target table schema, then generate methods to access each column), is one of the most powerful dynamic features of Groovy, yet it cannot be emulated in Java—all Java methods must exist at compile time. The popular web application framework Grails borrows much of the convention-over-configuration philosophy (designing libraries by preferring intuitive, common defaults versus extensive explicit configuration files) from Ruby on Rails, yet is built on established Java components like Spring and Hibernate.

Moss Collum, a developer at Cyrus Innovation, was working with his team on a five-year-old Java project. The team had started to feel frustration at how much code it took to express certain concepts in Java, especially for manipulating collections of objects (for example, finding the largest item

The Java Virtual Machine

The Java Virtual Machine (JVM) is the runtime platform designed for applications written in the Java language. The JVM has been ported to several operating systems and hardware devices. It also includes a mature garbage collector and just-in-time compiler that boosts runtime performance. The wide deployment and runtime optimizations of the JVM make it an appealing target for alternative languages beyond Java.

Functions as First-class Objects, Closures, and Lambdas

In Java, objects can be passed as arguments to method calls, assigned to variables, and returned as a result of a method call. This is not possible with functions. That is, functions are not first-class objects in Java.

A closure is a function that may refer to the variables that were in scope when the function was created. Lambdas are functions without a name—that is, they are anonymous. These language features enable creating flexible, reusable control structures and succinct libraries.

in a set or transforming a list of objects by applying a function to each value.) They experimented by starting to write new controllers and model objects in Groovy. "Groovy's very Java-like syntax was helpful with the large existing codebase as it minimized the cost of context switching between Java and Groovy code," said Collum. "It also meant that porting Java to Groovy was dead simple; since most Java is also valid Groovy, we got into a rhythm of moving the Java code into a Groovy file and then removing the excess Java noise. Groovy's closures, easy code reflection, and concision addressed our biggest Java complaints. We've found several gotchas and a few bugs, mainly in IDE support and Java interoperability, but even so the gains are so great that we wouldn't go back to Java."

Scala

Scala offers a fusion of object-oriented and functional language programming models. This means that in Scala, programs can be composed in terms of both objects (the bread and butter of Java programs) and functions (something that can only be simulated rather verbosely in Java using anonymous inner classes). Scala is the only statically typed alternative language mentioned in this article, but it offers flexible syntax and type inference that can give Scala programs the feel of a dynamic language.

Using a new language for implementing tests is one of the most common ways to introduce a new language into a Java project. Kris Nuttycombe, cofounder of ReportGrid.com, got started with Scala at a previous job by writing tests for a Java application.

"I picked Scala because my code had been moving towards a functional style for a long time, and a stint with Ruby made me want to use lambdas (anonymous functions)," Nuttycombe said. "I started using it just for testing, but I was writing better code for the tests than in the project, so 'adding Scala' eventually transitioned to 'replacing Java code with Scala.' We started running Scala in production a month after starting to use it for testing, but migrating the full codebase was a longer process; we would basically port code from Java to Scala as the opportunity arose in the refactoring process. That codebase is still in production use, continues to be developed, and handles tens of millions of dollars in transactions a year."

JRuby

JRuby is an implementation of the Ruby language on the JVM. Ruby's popularity exploded with the Ruby on Rails web framework, known for its high productivity. JRuby can run Ruby on Rails applications in an existing Java infrastructure.

Najati Imam was working for Cyrus Innovation at a large financial services client with an investment in Java and Unix infrastructure. When the team needed to add a feature to a legacy Perl CGI script, they realized they could rewrite the application faster and make it easier to change in Ruby on Rails than they could by modifying the Perl code. But getting a Ruby production environment approved would have taken several months.

"We picked JRuby so that we could run Ruby on Rails in a Java application server," said Imam. "We demonstrated that JRuby was a well-trusted, open source Java library like so many others the client was already using."

Their investment paid off; the team was able to deliver new features on time and at a pace that far exceeded the client's expectations.

"JRuby ended up being a Trojan horse that allowed us to sneak adaptive, rapid feature development in to the enterprise."

Sometimes, combining an alternative language with Java can yield something unique, pulling together the best parts of each. Bob McWhirter of JBoss is the founder of the TorqueBox project, which lets developers write JRuby applications using frameworks like Rails, Sinatra, and Rack that are served on the JBoss Application Server. This means developers get the flexibility and conciseness of Ruby with the high-performance messaging, asynchronous task processing, and scheduling facilities of a robust Java application server.

"I fell in love with Ruby," said McWhirter, "and thus

Concision and Reflection

Java tends to be verbose for certain operations (see the sidebar on anonymous inner classes). Other operations like reflection, or inspecting and invoking code at runtime, also require a fair amount of code. The alternative languages listed in this article have simpler reflection features, requiring just one line of code where several would be required in Java.

Java's Anonymous Inner Classes, Collections, and Concision

Java does not provide first-class functions or lambdas. Instead, they must be simulated using anonymous inner classes, which can be verbose.

Imagine you had a list of numbers and wanted the list of only those numbers that were less than five. This is how you might write the code to filter that list in Java:

```
Collection<Integer> filtered =
  Collections2.filter(numbers,
    new Predicate<Integer>() {
      public boolean apply(Integer input) {
        return input < 5;
    }
  });
```

Note that the Predicate class is being used to simulate a function that takes one argument and returns true or false. The Collections2 class is a utility class and is required since the original Java collections libraries did not include support for operations involving predicates. And here's how the same code would look in Scala, which includes lambdas and a library of lists and collections that use lambdas:

```
numbers.filter { x => x < 5 }
```

This defines an anonymous function that takes a parameter x and returns the result of the expression x < 5. Note also that the Scala version does not need to specify the types of the function arguments; the Scala compiler can infer them using a process known as type inference. Scala, like Java, is a statically typed language, which means that argument types are checked at compile time.

Here is a similar implementation in Ruby, a dynamic language:

```
numbers.select { |x| x < 5 }
```

Note that in the Ruby version, the types are also omitted. This is because Ruby is a dynamically typed language, where the parameter and variable types are not declared in the source code. Groovy and Clojure are also dynamically typed languages and share the same concision.

needed some way to tie that back in to the Java-centric company for which I worked. The JRuby interpreter had really been doing well, and I figured JBoss had a rather large bag of enterprise-grade technology. Mix the two, and you have TorqueBox."

Clojure

Clojure is a dialect of Lisp, and unlike the other languages listed here, is not object-oriented. It features a radically different approach to managing program state; everything is immutable (unmodifiable after initialization) by default, and any changes must be coordinated through Clojure's software transactional memory or other concurrency libraries.

Dan Chamberlain of Chamberlain Consulting was working with a team building a new feature that required implementing computationally expensive, recursive search algorithms. Due to the size of the data set and the performance requirements, the searches needed to be run in parallel. When the team realized just how complex and how much code would be needed to implement these searches in parallel in Java, they started looking for implementation alternatives. Chamberlain's team chose to build this feature in Clojure.

"It really came down to immutability, recursion, and parallelism," said Chamberlain. "You can do these things in Java, but they're dead simple in Clojure."

"Things didn't always go so smoothly," continued Chamberlain. "The learning curve was pretty steep since we didn't have a background in a Lisp programming language. There have also been some issues with some of the tools and features of the language that have made us change course a couple of times. But, overall, it has been a pleasant experience."

Growing Beyond Java

The JVM has become a great platform for trying out new languages. It offers a diverse collection of alternatives, each providing a unique opportunity to learn something new. And by using languages that run on the JVM, you get the benefits of experimentation in addition to being able to use these languages in your existing Java projects.

Perhaps one of the greatest benefits of learning a new language is in self-improvement. Andrew Hunt and David Thomas write in *The Pragmatic Programmer*, "Learn at least one new language every year. Different languages solve the same problems in different ways. By learning several different approaches, you can help broaden your thinking and avoid getting stuck in a rut." **{end}**

dan@danielwellman.com

Sticky Notes

For more on the following topic go to www.StickyMinds.com/bettersoftware or scan the QR code below.

- Supplemental materials