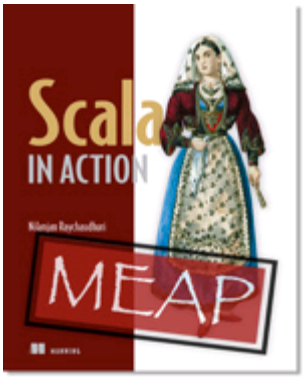
MANNING PUBLICATIONS

Scala does more with less code

An article from



[Scala in Action](#)
EARLY ACCESS EDITION

Nilanjan Raychaudhuri
MEAP Release: March 2010
Softbound print: Early 2011 | 525 pages
ISBN: 9781935182757

This article is taken from the book Scala in Action*. The author shows how Scala is capable of accomplishing the same goals as Java with fewer lines of code.*

"If I were to pick a language to use today other than Java, it would be Scala."

James Gosling

Java was first released in May 1995 by Sun Microsystems. Since then, Java has come a long way as a programming language. When Java came to the programming language scene, it brought some good ideas like a platform-independent programming environment (write once, run anywhere), automated garbage collection, and object-oriented programming. Java made object-oriented programming easier for developers when compared with C/C++ and got quickly adopted into the industry.

Over the years Java is becoming a bloated language. Every new feature added to the language brings with it more boilerplate code for the programmer. Even small programs can become bloated with annotations, templates, and type information. As Java developers, we're always looking for new ways to improve our productivity using third-party libraries and tools. But is that the answer to our problem? Why not have a more productive programming language?

Today, developers' needs are very different than they used to be. In the world of Web 2.0 and agile development, we understand how important it is to have flexibility and extensibility in our programming environment. We need a language that can scale and grow with us. If you're from Java, then Scala is that language. It will make you productive and it will allow you to do more with less code and without the boilerplate code.

To demonstrate the succinctness of Scala, we have to dive into the code. Let's pick up a simple example of finding an uppercase character in a given string and compare Scala and Java code (see listing 1).

Listing 1 Finding an uppercase character in a string using Java

```
boolean hasUpperCase = false;
for(int i = 0; i < name.length(); i++) {
    if(Character.isUpperCase(name.charAt(i))) {
        hasUpperCase = true;
        break;
    }
}
```

In this code, you're iterating through each character in the given string name and checking whether the character is uppercase. If it's uppercase, you set the `hasUpperCase` flag to `true` and exit the loop. Now let's see how we could do it in Scala (listing 2).

Listing 2 Finding an uppercase character in a string using Scala

```
val hasUpperCase = name.exists(_.isUpper)
```

In Scala you could solve this problem with a single line of code. Even though it's doing the same amount of work, most of the boilerplate code is taken out of the programmer's hands. In this, case we're calling a function called `exists` on `name`, which is a string, by passing a predicate that checks whether the character is true, and that character is represented by `_`. Here I wanted to demonstrate the brevity of the Scala language without losing the readability, but now let's look at another example (listing 3), where we create a class called `Programmer` with the properties `name`, `language`, and `favDrink`.

Listing 3 Defining a `Programmer` class in Java

```
public class Programmer {

    private String name;
    private String language;
    private String favDrink;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getLanguage() {
        return language;
    }
    public void setLanguage(String language) {
        this.language = language;
    }
    public String getFavDrink() {
        return favDrink;
    }
    public void setFavDrink(String favDrink) {
        this.favDrink = favDrink;
    }
}
```

This is a simple plain old Java object (POJO) with three properties—nothing to it. In Scala, we could create a similar class in a single line, as shown in listing 4.

Listing 4 Defining a `Programmer` class in Scala

```
class Programmer(
  var name:String,
  var language:String,
```

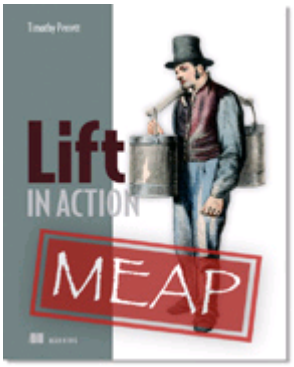
```
    var favDrink:String
    )
```

In this example, we're creating a similar class, called `Programmer` in Scala, but with something called a primary constructor (similar to the default constructor in Java) that takes three arguments. Yes, in Scala you could define a constructor along with the class declaration—another example of succinctness in Scala. The `var` prefix to each parameter will make the Scala compiler generate a getter and a setter for each field in the class. That's impressive, right? It's clear that, with Scala, we could do more with fewer lines of code. You could argue that IDE will automatically generate some of this boilerplate code, and that's not a problem. But I'd argue that you'll still have to maintain the generated code. In Java and Scala code comparisons, the same feature requires 3 to 10 times fewer lines in Scala than Java.

Summary

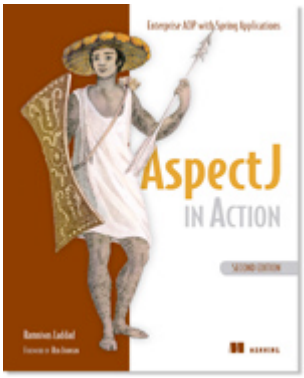
You learned why you should consider learning Scala as your next programming language. Scala's extensible and scalable features make it a language that you could fit into small to large programming problems. Its multi-paradigm model provides programmers with the power of abstractions from both functional and object-oriented programming models.

Here are some other Manning titles you might be interested in:



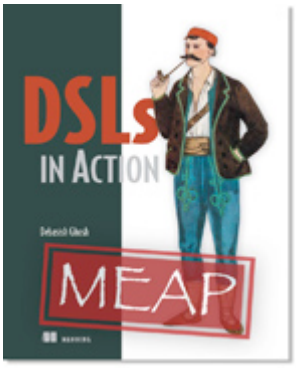
Lift in Action
EARLY ACCESS EDITION

Timothy Perrett
MEAP Release: April 2010
Softbound print: Early 2011 | 450 pages
ISBN: 9781935182801



AspectJ in Action, Second Edition
IN PRINT

Ramnivas Laddad
MEAP Release: October 2009
September 2009 | 568 pages
ISBN: 1933988053



DSLs in Action
EARLY ACCESS EDITION

Debasish Ghosh
MEAP Began: October 2009
Softbound print: Fall 2010 | 375 pages
ISBN: 9781935182450