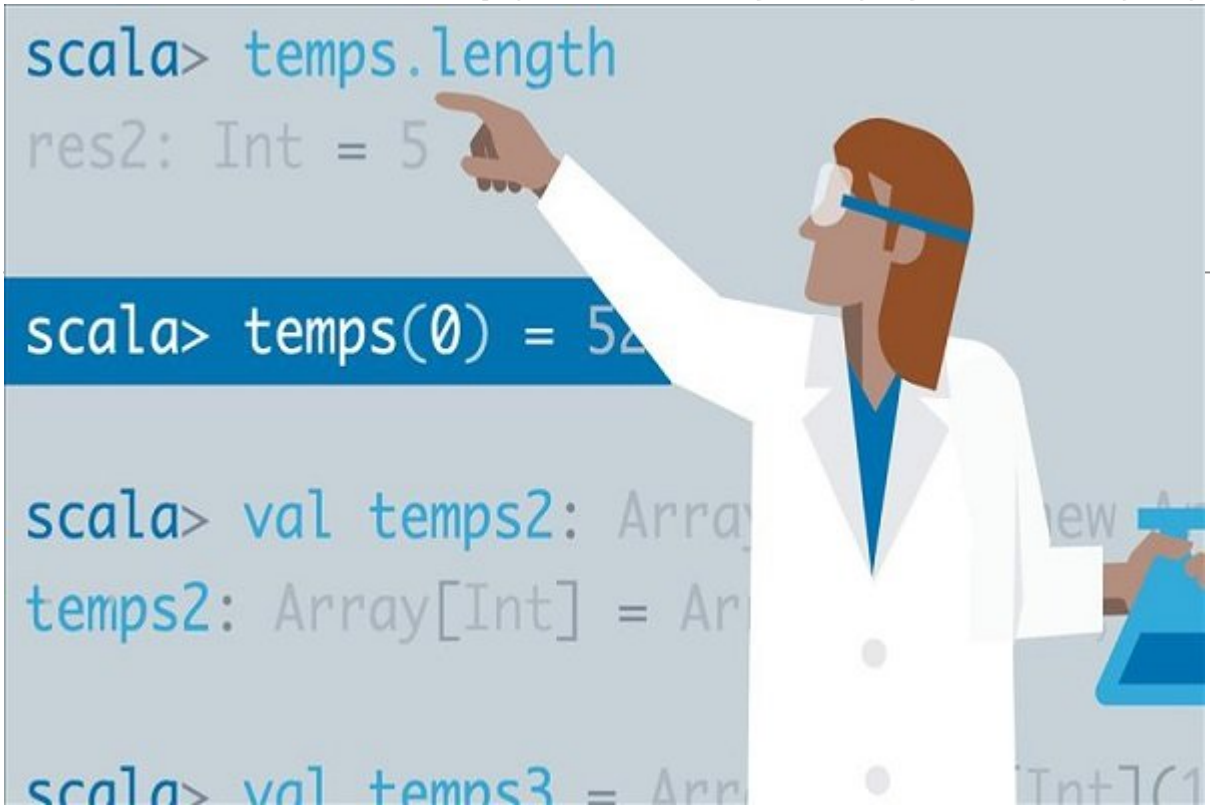
سرمایه‌گذاری برای آینده

چرا یادگیری زبان اسکالا و برنامه‌نویسی تابعی مهم است



تقریبا اکثر قریب به اتفاق برنامه‌نویسان به این حقیقت اذعان دارند که اسکالا یکی از رقبای اصلی و جدی زبان جاوا است. اسکالا زبانی است که به یک برگ برنده مهم که همانا چندپارادایمی است، تجهیز شده است. چندپارادایمی یکی از فاکتورهای مهمی است که باعث می‌شود برنامه‌نویسان جذب یک زبان برنامه‌نویسی شوند. در این مقاله با 10 دلیل مهمی آشنا خواهید شد که در چند سال اخیر باعث شده‌اند برنامه‌نویسان زبان جاوا به سمت زبان اسکالا متمایل شوند.

چندپارادایمی بودن در زبان اسکالا به چه معنا است؟

در دنیای امروز، برنامه‌نویسان برای آن‌که بتوانند سطح مهارت‌های برنامه‌نویسی خود را بهبود بخشند، مجبور هستند دست کم به یک زبان چندپارادایمی تسلط پیدا کنند. به لطف JVM برنامه‌هایی که با اسکالا نوشته می‌شوند کارایی بالایی دارند. همین **کارایی بالا به اسکالا** اجازه داده کوتلین و Groovy را پشت سر گذاشته و به انتخاب اصلی افرادی تبدیل شود که بیشتر عمر خود را صرف کدنویسی می‌کنند. اسکالا چند قابلیت بسیار مهم دارد که چندپارادایمی بودن از شاخص‌ترین آن‌ها است. برنامه‌نویسی تابعی قابلیتی در اختیار توسعه‌دهندگان قرار می‌دهد تا به بهترین شکل بتوانند از معماری چندهسته‌ای پردازنده مرکزی استفاده کنند. در این میان، اسکالا توانسته ترکیب خوبی میان این قابلیت و برنامه‌نویسی شی‌گرایی به وجود آورد. اسکالا نه تنها قادر است بسیاری از مشکلات رایج جاوا را حل کند، بلکه قادر است الگوهای طراحی قدرتمندی در اختیار برنامه‌نویسان قرار دهد.

با وجود انعطاف‌پذیری و قدرتمندی زبان‌های پویایی شبیه به پایتون، این زبان هنوز هم در برخی موارد با محدودیت‌هایی روبه‌رو است. بهره‌مندی از دو پارادایم متفاوت نه تنها به برنامه‌نویسان اجازه می‌دهد از طریق یادگیری یک زبان با دو پارادایم مهم آشنا شوند، بلکه اجازه می‌دهد از هر دو سبک برنامه‌نویسی توامان با یکدیگر استفاده کنند، بدون آن‌که مشکل خاصی در این زمینه داشته باشند. شبیه به سایر زبان‌های برنامه‌نویسی تابعی در اسکالا توابع به صورت First-class هستند به این معنا که شما قادر هستید همانند سایر مقادیر آن‌ها را به‌عنوان آرگومان ورودی به سایر توابع انتقال دهید. اسکالا همچنین از توابع بدون نام (Anonymous) و Currying پشتیبانی می‌کند. ترکیب این ویژگی‌ها با یکدیگر باعث شده تا کدهای نوشته شده کاملا مختصر و زیبا باشند. چندپارادایمی یکی از قوی‌ترین **ویژگی‌های اسکالا** است. البته لازم به توضیح است که جاوا نیز در نسخه 8 خود و در تلاش برای رسیدن به چنین توانمندیی قابلیت Lambda expressions را معرفی کرد. اضافه شدن عبارت‌های لامبدا از آن جهت حائز اهمیت بود که جاوا در مسیر پیوستن به جمع زبان‌های برنامه‌نویسی تابعی قرار دارد. قابلیت یادشده به برنامه‌نویسان اجازه می‌دهد تا یک تابع را به‌عنوان پارامتری برای تابع دیگری ارسال کنند. به عقیده برخی از برنامه‌نویسان اسکالا در این زمینه کارآمدتر از جاوا عمل می‌کند.

قابلیت همکاری متقابل با زبان جاوا

بدون شک قابلیت همکاری با جاوا یکی دیگر از نقاط مثبت **زبان اسکالا** است. برنامه‌نویسان می‌توانند کدهایی را که به **زبان اسکالا** نوشته‌اند با ماشین مجازی جاوا (JVM) اجرا کنند. توسعه‌دهندگان همچنین این قابلیت را در اختیار دارند تا از کدهای جاوا در اسکالا استفاده کنند. به عبارت دقیق‌تر، توسعه‌دهندگان این قابلیت را در اختیار دارند از کتابخانه‌های موجود در زبان جاوا به شکل مستقیم در توسعه نرم‌افزارها و کدهای اسکالا بهره ببرند. با توجه به این‌که بسیاری از توسعه‌دهندگان از زبان جاوا به سمت اسکالا می‌آیند، این یک راهکار عالی است که به آن‌ها اجازه می‌دهد از سال‌ها تجربه کاری که به دست آورده‌اند به شکل مستقیم در زبان دیگری استفاده کنند. همچنین امکان فراخوانی کدهای اسکالا از درون جاوا امکان‌پذیر است. در نتیجه توسعه‌دهندگان می‌توانند بخشی از کدنویسی پروژه خود را به زبان اسکالا انجام داده و بخش دیگری را با زبان جاوا بنویسند. به‌طور خلاصه، قابلیت همکاری با جاوا باعث شده تا زیرساخت اسکالا به شکل گسترده‌ای به یکی از زبان‌های برنامه‌نویسی اصلی کسب‌وکارهای مختلف تبدیل شود. فراموش نکنید بخش مهمی از برنامه‌های بزرگ سازمانی امروزی به زبان جاوا هستند و همین موضوع به شکل غیرمستقیم به پیشرفت زبان اسکالا کمک فراوانی می‌کند.

بهترین راهکارها و الگوهای از پیش ساخته شده در اسکالا

یکی از نکات تقریبا پنهان **زبان اسکالا** که شاید بسیاری در مورد آن اطلاعی نداشته باشند به نحوه ساخت این زبان باز می‌گردد.
پایه‌های شکل‌گیری زبان اسکالا در ابتدا در دانشگاه EPFL سوئیس کلید خورد. پژوهشگران این دانشگاه به دنبال آن بودند تا یک نوآوری جدید در حوزه زبان‌های برنامه‌نویسی به وجود آورند. آن‌ها به دنبال شکل خاصی از نوآوری بودند که در نهایت بتواند قشر کثیری از برنامه‌نویسان را به سمت خود جذب کند. مشابه با رویکردی که زبان جاوا در آن موفق ظاهر شد. در زبان‌های برنامه‌نویسی یکسری الگوها و شیوه‌های عملی خاص که در اصطلاح تخصصی به آن‌ها Best Practice گفته می‌شود، ساخته می‌شوند. در ارتباط با زبان اسکالا می‌توان به مواردی همچون تعریف متغیرهای غیرقابل تغییر در سطح بالا از طریق کلیدواژه var اشاره کرد. قابلیتی که در اصل نسخه بهینه‌سازی شده‌ای از کلیدواژه final در زبان جاوا یا const/read-only در زبان سی‌شارپ است. در هر دو زبان، برنامه‌نویسان با یکسری قواعد و دستورالعمل‌های عجیب و غریب سروکار دارند. اسکالا همچنین قابلیتی با عنوان case classes را معرفی کرده که به برنامه‌نویسان اجازه می‌دهد کلاس‌های غیرقابل تغییر در این زبان تعریف کنند. Closure قابلیت ارزشمند دیگری است که اسکالا به آن تجهیز شده است. قابلیتی که پیش‌تر در زبان‌های برنامه‌نویسی پویایی شبیه به پایتون و روبی در جهت دستیابی به پارادایم برنامه‌نویسی تابعی به‌کار گرفته می‌شود. قابلیت Closure در اصل یک تابع بدون نام است که به توابع دیگر این توانایی را می‌دهد تا به متغیرهایی خارج از محدوده تعریف شده برای یک تابع دسترسی پیدا کنند و مقداری را بر مبنای تعداد متغیرهایی که به‌عنوان پارامتر ورودی دریافت می‌کنند، بازگردانند.

مطلب پیشنهادی

تقابل علاقه و اجبار: بر اساس یافته‌های سایت استک‌اورفلو
10 زبانی که برنامه‌نویسان علاقه دارند و 15 زبانی که بیزارند

صریح و شفاف بودن زبان اسکالا

زمانی که اسکالا را با جاوا قیاس می‌کنیم، مشاهده می‌کنیم کدهای اسکالا به شکل ساده‌تری درک می‌شوند که خود برگ برنده بزرگ دیگری در مقایسه با جاوا است. اسکالا به شکل ذاتی بر شفاف بودن تاکید دارد. کدهای نوشته شده به زبان اسکالا زیباتر و کاربردی‌تر هستند. دو فاکتور مهمی که باعث شده برنامه‌نویسان جاوا مجذوب این زبان شوند. برای آن‌که دید روشنی در خصوص این تفاوت‌ها به دست آورید پیشنهاد می‌کنیم به آدرس زیر مراجعه کنید.

http://www.java67.com/2015/10/java-program-to-find-repeated-words-and-co...

در این آدرس سورس کدی را مشاهده می‌کنید که برای پیدا کردن تعداد تکرار یک واژه در یک فایل متنی به زبان جاوا نوشته شده است. در حالی که اسکالا به شکل ساده‌تری قادر به انجام این کار است. البته نسخه 8 زبان جاوا

با یکسری تغییرات همراه بود و یکسری ویژگی‌های موجود در آن بهبود پیدا کردند. با وجود این باید بگوییم اسکالا در زمینه کدنویسی مختصر و شفاف یک سروگردن بالاتر از زبان جاوا است.

بازار کار بهتر

برنامه‌نویسی را سراغ دارید که به دنبال یک فرصت شغلی نباشد یا نرم‌افزاری را طراحی کند که هیچ‌گاه آن‌را به بازار عرضه نکند؟

برنامه‌نویسان همواره به دنبال یادگیری یک فناوری یا چارچوب جدید هستند، به دلیل این‌که می‌دانند همواره بازار کار خوب و پیشرفت خوب به واسطه تسلط بر یک فناوری جدید پیش روی آن‌ها قرار می‌گیرد. به همین دلیل است که یادگیری زبان اسکالا و تسلط بر آن، این اطمینان خاطر را به توسعه‌دهندگان می‌دهد تا در آینده مشکلی از بابت پیدا کردن فرصت‌های شغلی جدید نداشته باشند. در مقطع فعلی شرکت‌هایی همچون توییتر، لینکدین، Foursquare و Quora از این زبان استفاده کرده یا در حال عزیمت به سمت این زبان هستند. با توجه به بازاریابی درست و هوشمندانه انجام شده در ارتباط با گسترش‌پذیری اسکالا، جای تعجب نیست که مشاهده می‌کنیم بانک‌ها و سازمان‌های مالی بزرگ ترغیب شده‌اند در حوزه کاری خود از اسکالا استفاده کنند. در همین ارتباط توییتر بهترین شیوه‌های توسعه برنامه‌های کاربری با زبان اسکالا را به‌نام Effective Scala منتشر کرده است. بنیان‌گذاران سایت Quora نیز چند ماهی است فرآیند ساخت یک چارچوب اختصاصی را برای استفاده در زبان اسکالا آغاز کرده‌اند. مارتین اودراسکای مردی که در پس‌زمینه طراحی زبان اسکالا قرار داشت، اکنون به شکل جدی در پروژه Typesafe متمرکز شده است. پروژه‌ای که به دنبال آن است تا نشان دهد از اسکالا در زمینه‌های تجاری به‌خوبی می‌توان استفاده کرد. با کمی تحقیق در این زمینه متوجه می‌شویم که زبان اسکالا در مسیر کاملا درست پیشرفت قرار گرفته و این پتانسیل را دارد تا جایگزین اصلی جاوا شود. جالب آن‌که جاوا از جانب زبان کوتلین نیز در تعقیب است. زبانی که در نظر دارد در زمینه توسعه برنامه‌های اندرویدی جای جاوا را بگیرد.

Statically Typed

به‌طور کلی، یک زبان ایستا شبیه به جاوا مانع از آن می‌شود تا برنامه‌نویسان در زمان کدنویسی دچار اشتباهات ریز اما تاثیرگذار شوند. در حالی‌که در یک زبان برنامه‌نویسی پویا شبیه به پایتون خطاهای برنامه تنها در هنگام اجرای یک برنامه قابل تشخیص هستند. اسکالا هر دو ویژگی یاد شده را در خود جای داده است. به عبارت دیگر، اسکالا در حالی‌که یک زبان پویا است، در عین حال یک زبان ایستا نیز به شمار می‌رود. کامپایلر اسکالا واقعا هوشمند بوده و از

مکانیزم استنتاج نوع به شکل دقیقی استفاده کرده و با توجه به مقداردهی اولیه‌ای که برای یک متغیر انجام می‌شود، قادر است نوع آن را تشخیص دهد. اسکالا از این مکانیزم در ارتباط با توابع نیز استفاده می‌کند. جالب آن‌که قابلیت استنتاج نوع متغیرها و توابع در اسکالا بهتر از زبان‌های سی‌شارپ و جاوا عمل می‌کند.

مطلب پیشنهادی

برنامه‌نویسان از شنیدن چه جملاتی بیزار هستند؟
کدنویسان با شنیدن این هفت جمله به سرعت عصبانی می‌شوند

چارچوب‌های در حال رشد اسکالا

پویایی و رشد بدون وقفه یکی از نقاط مثبت اکوسیستم اسکالا به شمار می‌رود. کتابخانه‌ها و چارچوب‌های بسیار خوبی برای اسکالا طراحی شده است. شرکت‌هایی که از این زبان در مناسبات تجاری خود استفاده می‌کنند، در عین حال به رشد اسکالا کمک فراوانی کرده‌اند. از چارچوب‌های مطرح وبی که برای اسکالا تعریف شده‌اند به Lift و Play می‌توان اشاره کرد. Akka چارچوب همزمانی قدرتمند دیگری است که بر مبنای اسکالا ارائه شده است. این چارچوب در قالب یک ابزار و محیط زمان اجرا برای ساخت برنامه‌هایی با هم‌زمانی بالا، توزیع شده و برنامه‌هایی که آستانه تحمل خطای آن‌ها بالا است، به کار گرفته می‌شود. اسکالا همچنین در زمینه بزرگ داده‌ها همراه با آپاچی اسپارک به کار گرفته می‌شود. توسعه‌دهندگان جاوایی که در زمینه بزرگ داده‌ها به فعالیت اشتغال دارند این حقیقت را پذیرفته‌اند که اسکالا در این زمینه عملکرد خوبی دارد.

جامعه روبه‌رشد

این تنها **زبان اسکالا** و چارچوب‌های پیرامون آن نیست که رشد بالایی دارند. جامعه برنامه‌نویسان این زبان نیز به سرعت در حال رشد هستند، به‌گونه‌ای که برخی از توسعه‌دهندگان جاوا نیز در حال پیوستن به جامعه برنامه‌نویسان اسکالا هستند. همچنین محیط‌های توسعه یکپارچه بیشتر و بیشتری از **ترکیب نحوی اسکالا** پشتیبانی می‌کنند. Eclipse و IntelliJ از جمله این موارد هستند. ابزارهای قدرتمند دیگری همچون Maven، SBT و Ant نیز در این زمینه در اختیار برنامه‌نویسان قرار دارد. نظرسنجی‌های مختلف انجام شده در خصوص اسکالا نشان می‌دهد، بسیاری از توسعه‌دهندگان بر این باور هستند که اسکالا جایگزین شماره یک برای جاوا است. با توجه به این‌که سازمان‌های بزرگی همچون توییتر از اسکالا استفاده می‌کنند، جای تعجب نخواهد بود که جامعه اسکالا روزبه‌روز بزرگ و بزرگ‌تر شود.

ترکیب نحوی دقیق اسکالا

درست است که ترکیب نحوی زبان جاوا به‌خوبی قابل درک است، اما جاوا یک مشکل بزرگ دارد. شما برای انجام یک کار کوچک باید کدهای فراوانی را بنویسید، در شرایطی که **زبان اسکالا** یک بنچ‌مارک جدید ارائه کرده تا ترکیب نحوی این زبان تا حد ممکن کوتاه و خوانا شود. کامپایلر اسکالا که به‌نام scalac معروف است، فراتر از آن چیزی که تصور می‌کنید قدرتمند است. این کامپایلر می‌تواند توابعی همچون hasCode ,equals(), toString() و سایر توابع را به شکل هوشمندانه برای برنامه‌نویسان تولید کند. در فهرست یک و دو با دو کلاس یکسان نوشته شده با زبان‌های جاوا و اسکالا آشنا می‌شوید. قضاوت در مورد خوانایی و کوتاه بودن را به شما واگذار می‌کنیم.

```
public class Book {
    private final String name;
    private final double price;
    public Star(String name, double price) {
        this.name = name;
        this.price = price;
    }
```

```
    @Override
    public int hashCode() {
        int hash = 7;
        hash = 23 * hash + Objects.hashCode(this.name);
        return hash;
    }
    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Test other = (Test) obj;
        if (!Objects.equals(this.name, other.name)) {
            return false;
        }
        if (Double.doubleToLongBits(this.price) != Double.doubleToLongBits(other.price)) {
            return false;
        }
        return true;
    }
    @Override
    public String toString() {
        return "Test{" + "name=" + name + ", price=" + price + '}';
    }
}
```

□□ □□□□□

شما همین کار را از طریق case class در اسکالا به شیوه بهتر و البته کوتاه‌تری می‌توانید انجام دهید.

```
case class Book(name: String, price: double)
```

□□ □□□□□

شما همچنین می‌توانید از کتابخانه Lombok یکی از کتابخانه‌های مهمی که از سوی برنامه‌نویسان جاوا به کار گرفته می‌شود برای حذف کدهای تکراری مرتبط با getters, setters, equals, hashCode و toString استفاده کنید.
این کتابخانه به شکل خودکار قادر است توابع یادشده را برای شما تولید کند.

مطلب پیشنهادی

یادگیری الگوی طراحی، انتخاب یا ضرورت
چند راهکار برای حل مشکلات رایج برنامه‌نویسی

یادگیری به نسبت ساده اسکالا

برای یک توسعه‌دهنده جاوا، یادگیری یک زبان برنامه‌نویسی تابعی کلاسیک شبیه به Haskell یا OCaml در مقایسه با اسکالا سخت‌تر است. به عبارت دیگر، **یادگیری اسکالا** به لطف شی‌گرایی بودن ساده‌تر است. اگر یک توسعه‌دهنده جاوا هستید، به‌جای آن‌که وقت خود را صرف یادگیری یک زبان برنامه‌نویسی تابعی کنید، این شانس را دارید تا از مهارت‌های خود در اسکالا با اتکا بر آموخته‌های قبلی خود در زمینه شی‌گرایی استفاده کرده و در مدت زمان کوتاهی بر ترکیب نحوی اسکالا تسلط پیدا کنید. فراموش نکنید اسکالا نیز همانند جاوا از کتابخانه‌های قدرتمند، مستندات آنلاین غنی و جامعه برنامه‌نویسان گسترده برخوردار است. بدون شک یادگیری اسکالا سرمایه‌گذاری برای حال و آینده است. پس اگر به دنبال یک زبان برنامه‌نویسی شی‌گرایی هستید که قابلیت‌های کاربردی زبان‌های پویایی همچون پایتون را در خود جای داده باشد پیشنهاد ما به شما اسکالا است.

تاریخ انتشار:

17 مرداد 1397

نشانی منبع:
https://www.shabakeh-mag.com/workshop/13608/%DA%86%D8%B1%D8%A7-%DB%8C%D8%A7%D8%AF%DA%AF%DB%8C%D8%B1%DB%8C-%D8%B2%D8%A8%D8%A7%D9%86-%D8%A7%D8%B3%DA%A9%D8%A7%D9%84%D8%A7-%D9%88-%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87%E2%80%8C%D9%86%D9%88%DB%8C%D8%B3%DB%8C-%D8%AA%D8%A7%D8%A8%D8%B9%DB%8C-%D9%85%D9%87%D9%85-%D8%A7%D8%B3%D8%AA