SIMON MAPLE

ANDREW BIN-STOCK

# Java開発者への過去最大の調査

10,500人のJava開発者が、プロジェクト、ツール、そして自身について明かす

以下のレポートに示すデータは、10,500名を超えるアンケート対象者による回答に基づくものです。調査にご協力くださった皆様に対し、経験の共有のために貴重な時間を割いていただきましたことに、深くお礼申し上げます。この調査は、SnykとJava MagazineがJavaコミュニティ全体、Javaユーザー・グループ（JUG）、Java Magazineの購読者を対象に行ったものです。多くの方に回答していただくために、SnykとJava Magazineは、Devoxx4kidsへの寄付を約束しました。

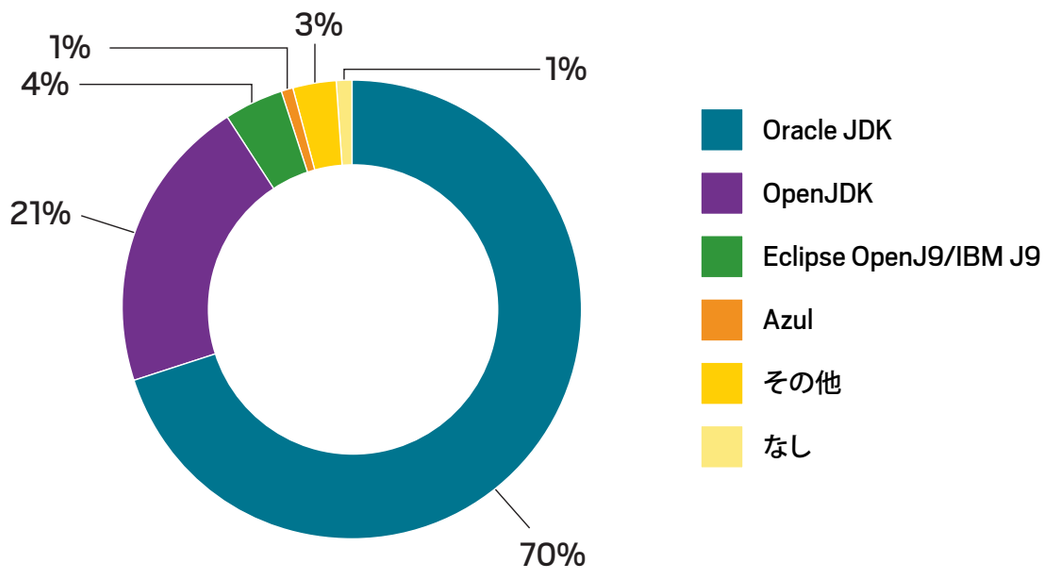
地域、所属企業の規模（該当する場合）、年齢、Javaの経験年数など、回答者に関する情報は、調査の最後に掲載しています。

## JDKについて

### 1.主要アプリケーションの本番環境で使っているのは、どのJavaベンダーのJDKですか。

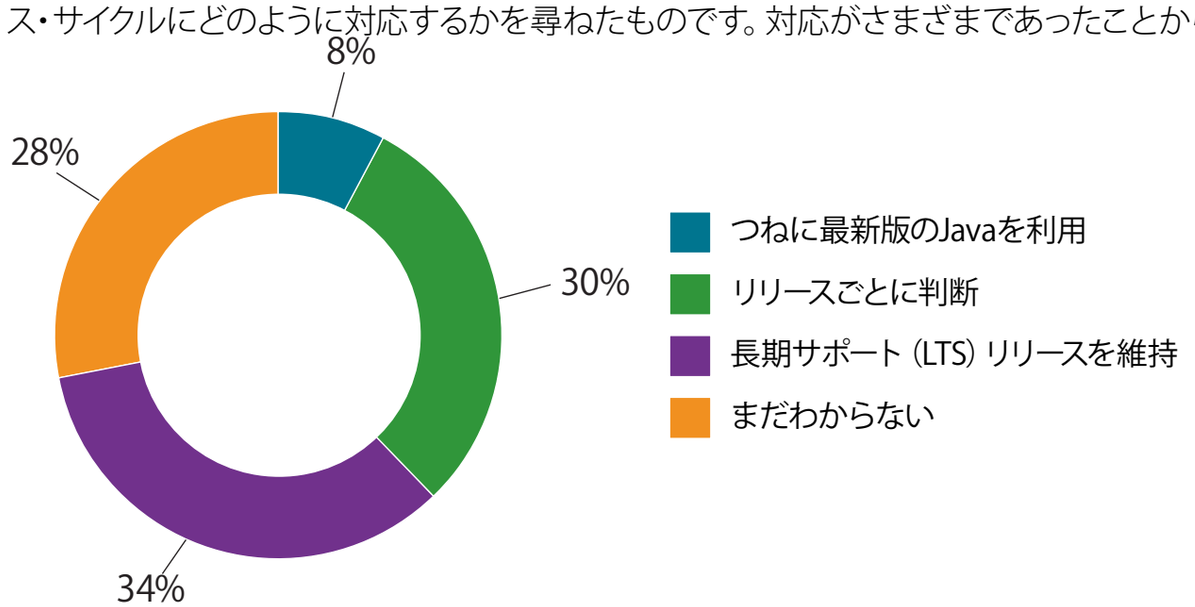
レポートは、核心的な質問から始めたいと思います。数多くのベンダーが独自のJDK実装を提供していますが、開発者が本番環境のアプリケーションで使っているのはどのプロダクトなのでしょうか。優勢なのは、Oracle JDKとOpenJDKであることがわかりますが、開発者の7割はOracle JDKの使用を、2割はOpenJDKの使用を選んでいることから、それほど激しい競争は起こっていないと言えるでしょう。ただし、今後のライセンスやサポートの変更により、これらの数字に変化が生じる可能性も考えられます。

写真（BINSTOCK）：BOB ADLER/STUDIO AT GETTY IMAGES FOR ORACLE




## 2.Javaの新しいリリース・サイクルにどう対応する予定ですか。

Java 9のリリースでは、アーキテクチャに大きな変更がいくつか行われましたが、同時に、半年ごとにJava SEのバージョンが提供されるという新たなリリース・サイクルも導入されています。2年から3年ごとに提供される長期サポート（LTS）リリースでは、セキュリティ更新などの長期サポートが行われます。この質問は、開発チームが新しいリリース・サイクルにどのように対応するかを尋ねたものです。対応がさまざまであったことから、明確に方向性が決まり

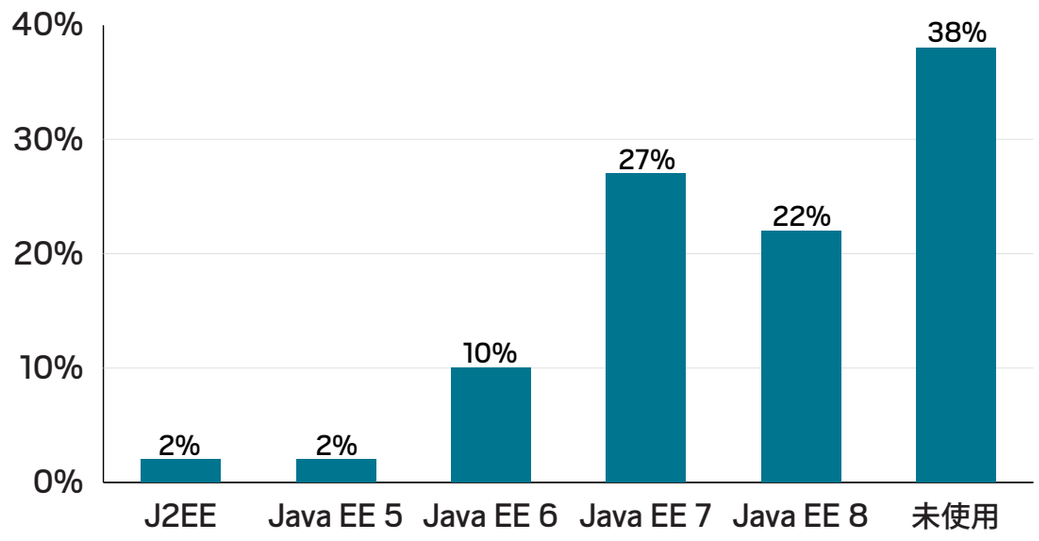


きっていないことがうかがえます。実際、3分の1近くの開発者が、新しいリリース・サイクルにどう対応するかまだわからないとしています。

おそらく、来年にはベスト・プラクティスが登場し、企業は好ましい移行サイクルに落ち着くものと予想されます。ただし、移行サイクルは業界によって大きく異なってくるでしょう。その結果、「まだわからない」の割合は減少すると思われますが、他のどの選択肢が増加するかは未知数です。

## 3.主要アプリケーションで使っているJava EEのバージョンはいくつですか。

回答者の4割近くは、エンタープライズJavaを使っていません。使っている場合、バージョン7が最多となっています。Java EE 8がリリースされたのは2017年9月であることから、リリースから1年未満で、もっとも多く使われているバージョンに迫っているというのは、前途有望です。Java SEとは違い、Java EEのリリースは実装を利用できるようになるまでに時間がかかるため、採用されるまでに長い時間が必要になります。さらに、アプリケーション・サーバーのベンダーは時間をかけて仕様を採用して実装する必要があるため、9か月もたたないうちにここまでユーザーに採用されているのは、前途有望な証だと言えます。



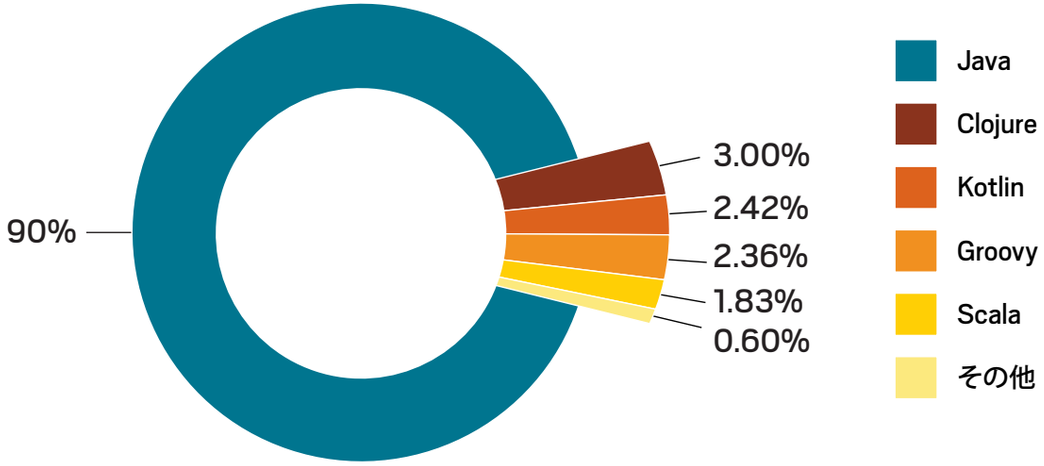

Jakarta EE 8の展開と採用が始まったときにこの数字がどのように変化するか、注目したいと思います。いまだにJ2EEを使っている2%の人のことも考えるべきでしょう。J2EEの最後のメジャー・バージョンのリリースは2003年でした。

## 4.主要アプリケーションで使っているおもなJVM言語は何ですか。

ちょうど9割のJVMユーザーは、主要アプリケーションにJavaを使っています。現在、多くのプロジェクトが多言語化されていますが、JVMの役割はおもにJavaを実行することです。

しかし、Javaは非常に好まれているものの、JVMベースの開発者は一貫して他のJVM言語にも強い興味を示してきています。その裏付けとして、Java Magazineや主要プログラミング・サイトで、Java以外のJVM言語に関する記事の人気が高いことが挙げられます。ここ数年間で登場した「ホット」なJVM言語は、JetBrainsのKotlinです。Kotlinは、着実に進展しており、Android開発がサポートされたことに加え、Gradleビルド・ツールのスクリプトではGroovyに次いで2番目の主要言語になっています。こういった採用が続いていることから、KotlinはScalaを超え、今回の調査ではGroovyも追い抜きました。

3.0%がClojureと答えているのは、非常に高い数値です。関数型プログラミングへの関心が続いている証でしょう（少なくとも私たちはそう捉えています）。多くのJava 8機能が関数型指向になっていることから、Java自体にも関数型プログラミングが組み入れられていることがわかります。

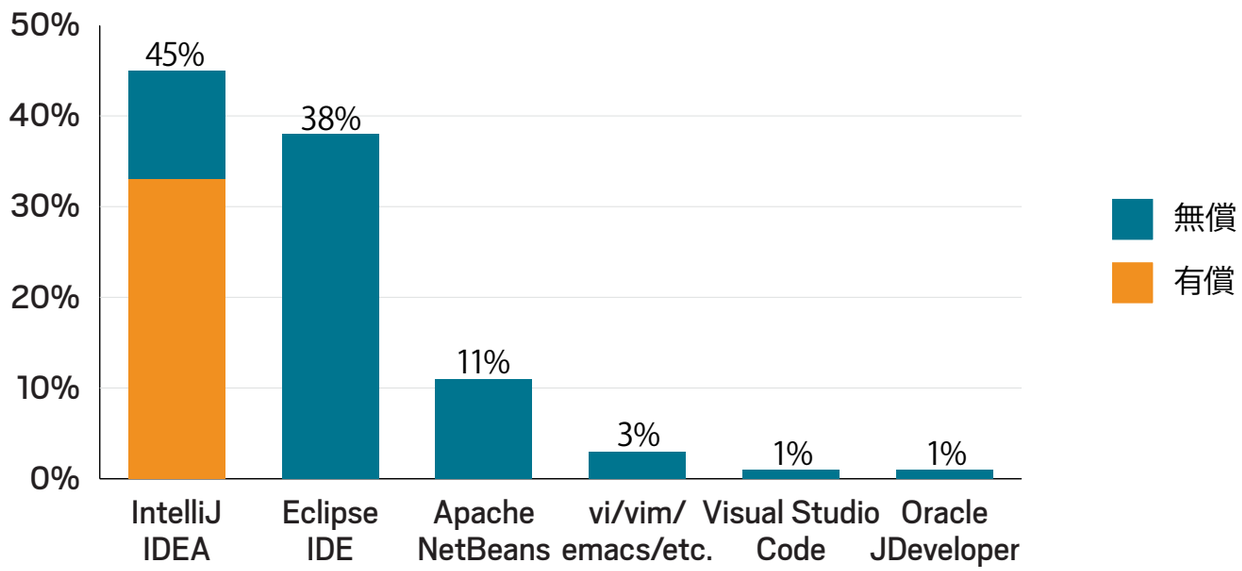


## ツールについて

### 5.どのIDEで開発を行っていますか。

次のグラフは、最近行われた他の調査とも一致しています。ここ1年から2年ほどでIntelliJがEclipseを超えましたが、Apache NetBeansはマーケットの約10%を維持しています。IntelliJ全体では45%ですが、そのうち32%はIntelliJ IDEA Ultimate Edition（有償版）、11%はIntelliJ Community Edition（無償版）、2%はAndroid Studioのユーザーです。Eclipseのカテゴリには、Eclipse STS、JBossツール、Rational Application DeveloperなどのEclipseベースのツールが含まれています。

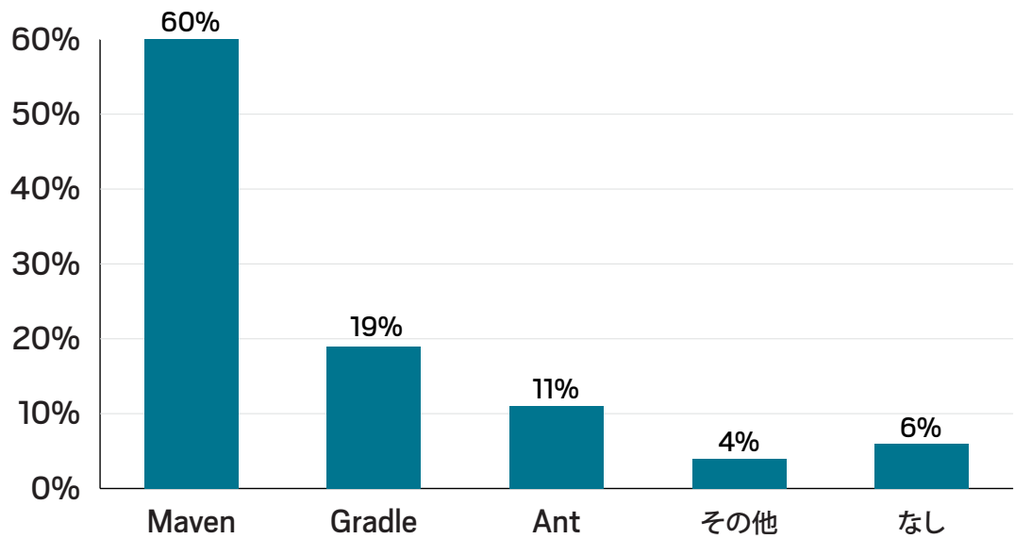
Apache NetBeansのユーザー数はさほど変化していません。このことは、オラクルからApache Software Foundationへの移管が、ユーザー・ベースには影響していないことを示唆しています。まだ1%にすぎませんが、Visual Studio Codeが登場してJavaコミュニティで頭角を現し始めたことも特筆すべきでしょう。また、おそらくこのレポートをタブレット（石板のことです）で読んでいる「vi/vim/emacsなど」のグループには敬意を表します。




### 6.主要プロジェクトで使っているビルド・ツールはどれですか。

この数を吟味するにあたっては、使っているおもなビルド・ツールを尋ねたという点が重要になります。私たちが知りたいのは、主要プロジェクトで使っているビルド・ツールです。チームがすべてのプロジェクトで使っているビルド・ツールを聞くのも興味深いかもしれませんが、回答が増え過ぎると、その有用性が低下してしまうことになりま

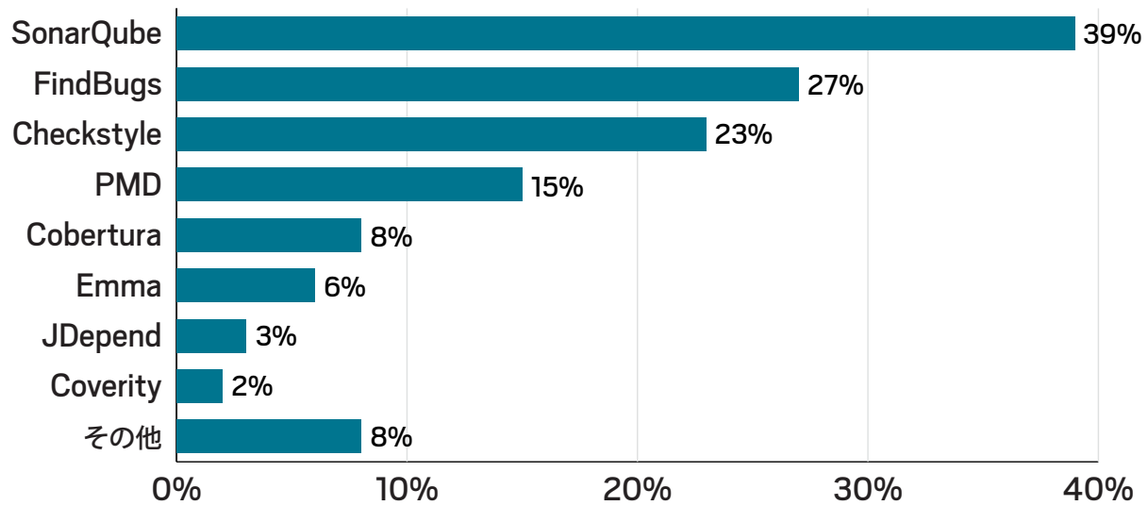
す。結果からわかるように、Mavenは3対1の比率でライバルのGradleを明らかに圧倒しています。開発者の10人に1人はいまだにAntを使っていますが、なんと20人に1人は何も使っていないと答えています。




RebelLabsが2016年に行った同様の調査（回答者数2,000）では、Mavenが68%、Gradleが16%でした。Gradleの採用率が増えているのはおそらく、スクリプト言語としてKotlinのサポートが追加されたためでしょう。また、GradleはAndroidプロジェクトのデフォルト・ビルド・エンジンでもあります。しかし、Mavenに迫るスピードは遅いと言ってよいでしょう。

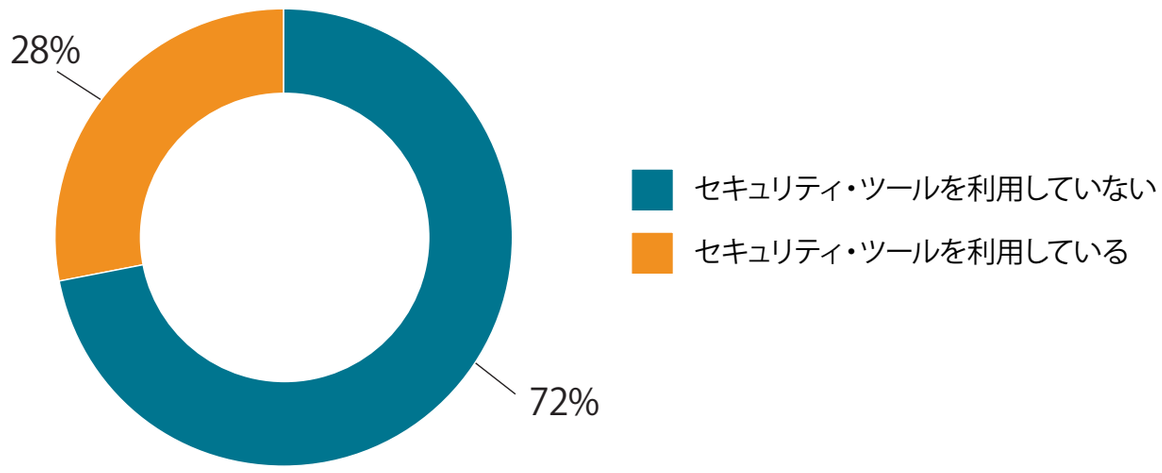
## 7.使っている静的品質ツールはどれですか。

情報を訂正させようと電子メールやツイートを送る前に、この質問は複数回答可能であるため、数値を合計しても100%にはならないという点に留意してください。このデータからわかることの1つは、ほんの一握りの静的品質ツールが広く使われていることです。上位につけているSonarQube、FindBugs、Checkstyleを見ても、特に驚きはありません。驚いたのは、回答者の36%がどの静的品質ツールも使っていないことです。私たちは、これらのツールを使うことは当たり前だと考えていました。




## 8.使っている静的セキュリティ・ツールはどれですか。

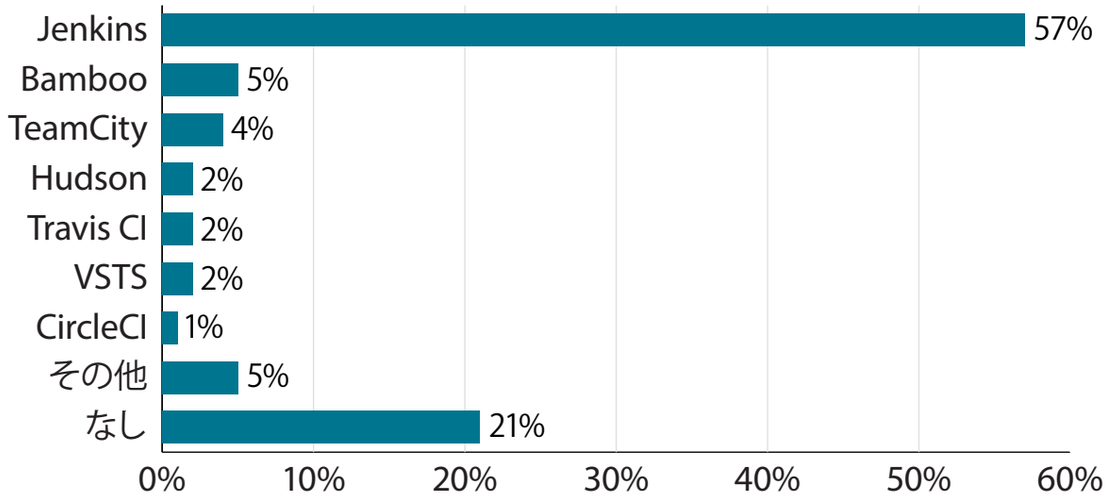
ほとんどの現場では、静的セキュリティ・ツールを使っていません。セキュリティ問題のコストが広く知られていることを考えると、全体的な採用率の低さは驚きです。ツールを使っている現場では、古いツールが普及しており、SonatypeとFortifyがマーケットを牽引しています。今回の調査で初登場したSnykは、多くのエコシステムで全自動修正を提供しています。今後の調査では、セキュリティ・ツールの採用が広がっているという結果が得られることを期待しています。



## 9.使っているCIサーバーはどれですか。

ほとんどの開発者の予想どおりだと思いますが、57%のマーケット・シェアを獲得して圧倒的勝利を収めたCIサーバーはJenkinsです。次点となったのは「なし」の21%で、Jenkins以外のCIサーバーをすべて足し合わせた数（22%）とほぼ一致します。Jenkins以外のCIサーバーは、いずれも5%未満のマーケット・シェアしかありません。Jenkinsにとって年配の親戚と言えるHudsonは、2%となっています。一般的にJava/JVMの領域では選択肢とならないと思われるVSTS（Microsoft Visual Studio Team Server）が2%を記録しているのは、特筆に値することでしょう。

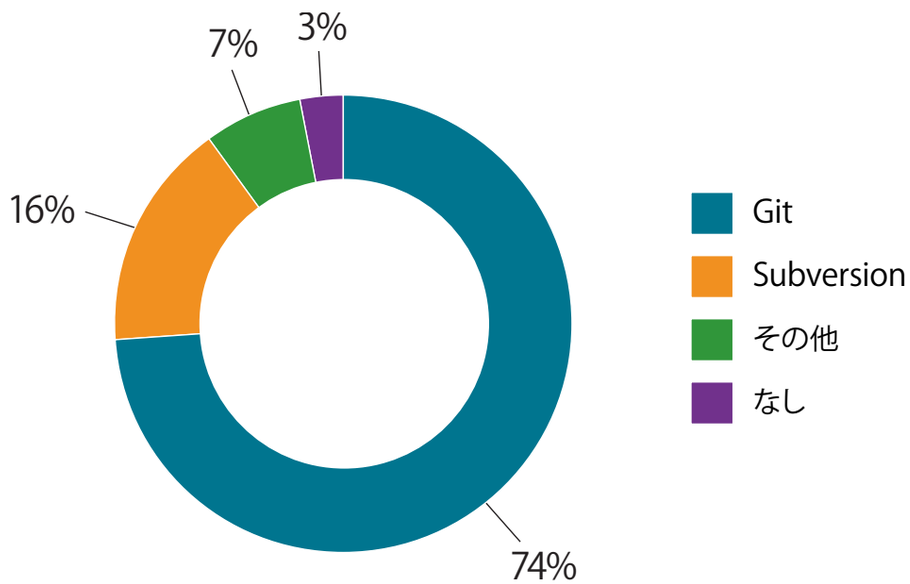
現在、ほとんどの開発者は、ほぼすべての現場で継続的インテグレーションを使うことを想定しています。そのため、5分の1のアプリケーションで何も使っていないことがわかったのは驚きです。現在では、個人的なプロジェクトでさえもCI（Travis CIやCircleCIなど）を使っています。こういったCIツールは、BitbucketやGitHubなどの公開プロジェクト・ホスティング・サイトで公開されています。もし皆さんがプロジェクトでCIを使っていない21%に含まれるのであれば、その理由をぜひ伺いたいと思っています。




## 10.チームが主要プロジェクトで使っているソース・コード管理プラットフォームはどれですか。

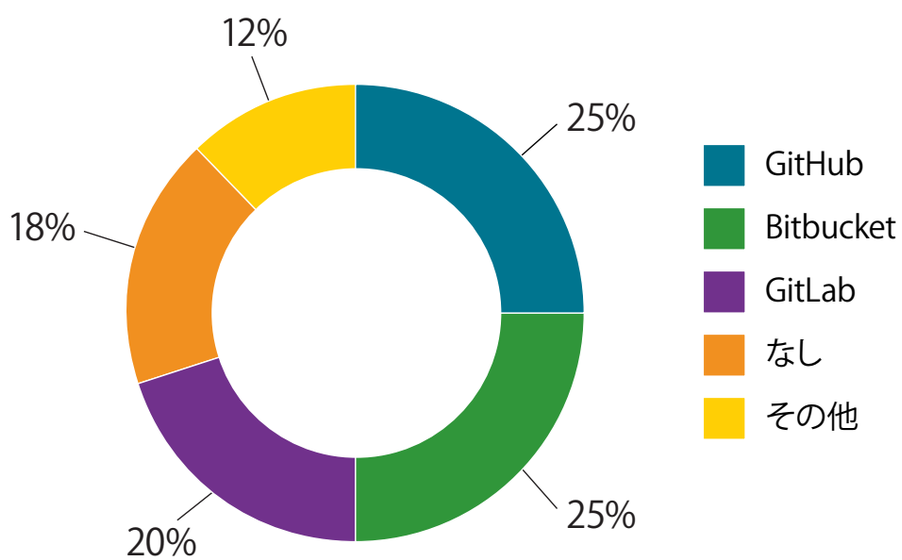
予想どおり、ソース・コード管理のレースでは、Gitが圧倒的勝利を収めています。その普及度合いに疑問を抱いてい

る方に向けて言えば、回答者の4分の3近くが、Gitでコードベースを管理しているチームで作業しています。その他の回答者の過半数は、Subversionと答えています。どういうわけか、2018年の現在でも、3%の人が何のソース・コード管理も使っていません。これには言葉を失ってしまいます。




## 11.主要プロジェクトで使っているコード・リポジトリはどれですか。

コード・リポジトリはソース・コード管理とはまったく違い、回答がはるかに分散しています。GitHibとBitbucketがそれぞれ25%で接戦を繰り広げており、GitLabが20%でその後ろにつけています。これらは、プロジェクト・ホスティン

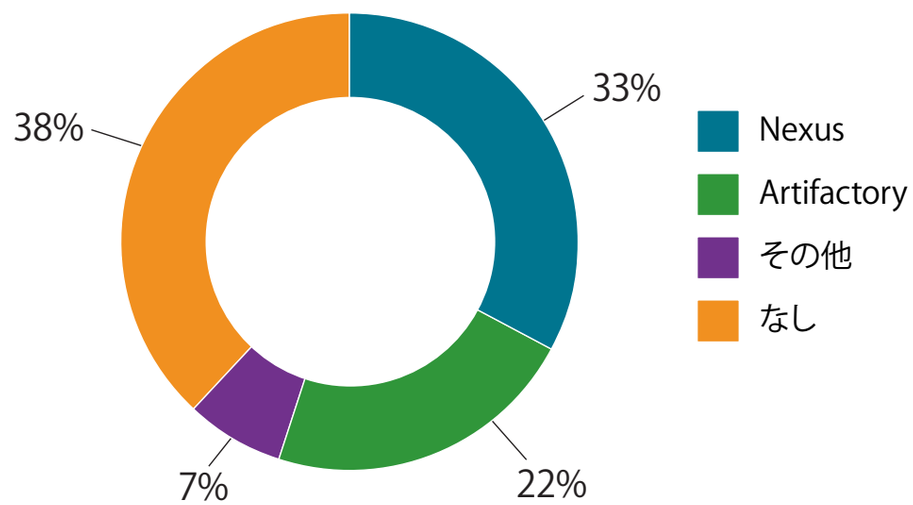


グの「ビッグ3」と呼べそうです。なお、この質問はパブリック・プロジェクトのみを対象としたものではなく（その場合、GitHubが相当優勢になると見込まれます）、プライベートのプロジェクト・ホスティングも対象としています。

最近MicrosoftがGitHubを買収したことは、今後の採用率に影響するかもしれません。この点は今後の調査で明らかになることでしょう。GitHubと答えた25%の回答者のうち、半数強（52%）はパブリック・バージョンを、残り（48%）はオンプレミスのプライベートなGitHub Enterpriseを使っています。「その他」のうち、VSTSが2%を占めています。

## 12.使っているプライベート・バイナリ/アーティファクト・リポジトリはどれですか。

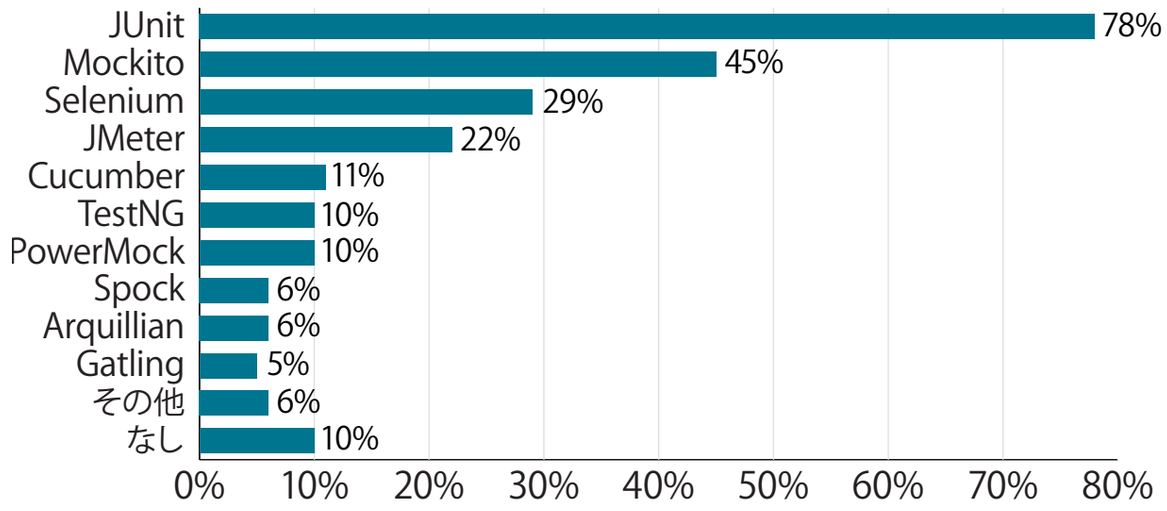
ほとんどの現場では、パッケージ・アーティファクト・リポジトリを使っていません。理論上は、使う必要がないからです。この便利さを活用したいという方は、定評があり、JVMエコシステムを非常に重視しているNexusを選択しています。Nexusに続いているのは、どちらかと言うと多言語エコシステムに人気の高いJFrogのArtifactoryです。




## 13.使っているテスト・テクノロジーはどれですか。

5分の4（近く）の人がJUnitを使っているという驚くべき結果と、さらに10%の人がTestNGを使っていることを考えると、JVMエコシステムでは、ユニット・テストが他を引き離してもっとも優勢なテスト手法であることは明白です

（回答者は複数の回答を選択できるため、合計は100%を超えます）。モッキングについて言えば、好まれるモッキング・フレームワークとして、Mockitoが台頭していることが明らかとなっています。

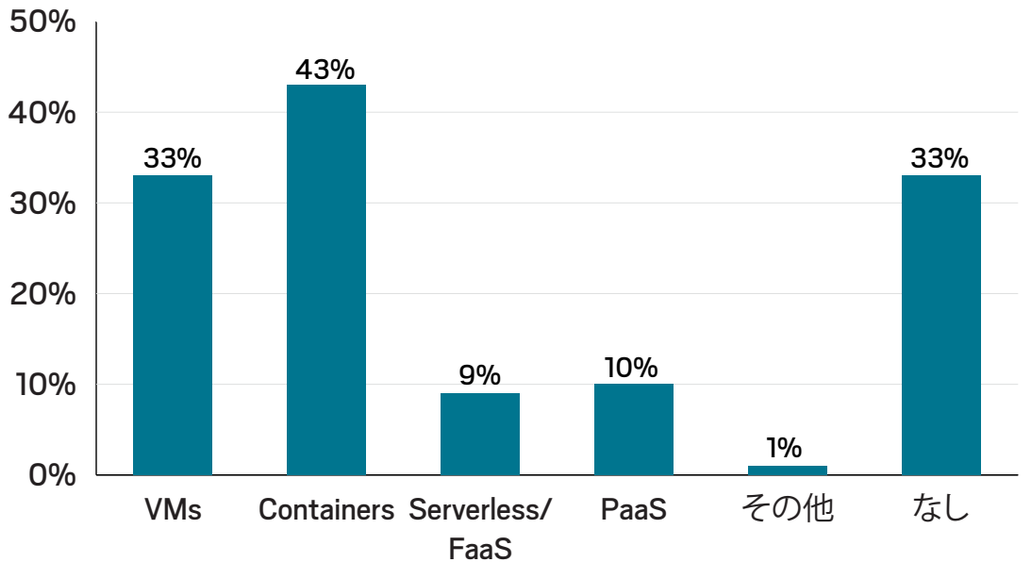



4分の1近くの回答者がJMeterを、5%の回答者がGatlingを使っていることから、パフォーマンス・テストの必要性が重視されるようになってきていることがわかります。Seleniumが29%となっている点も注目すべきです。静的品質ツールの回答とは違い、何のテスト・ツールも使っていないと答えた回答者はわずか10%です。ちょっと待ってください。1割の人はテスト・ツールを使っていないのでしょうか。人類への希望を失う前に、次に進むことにします。

## クラウドについて

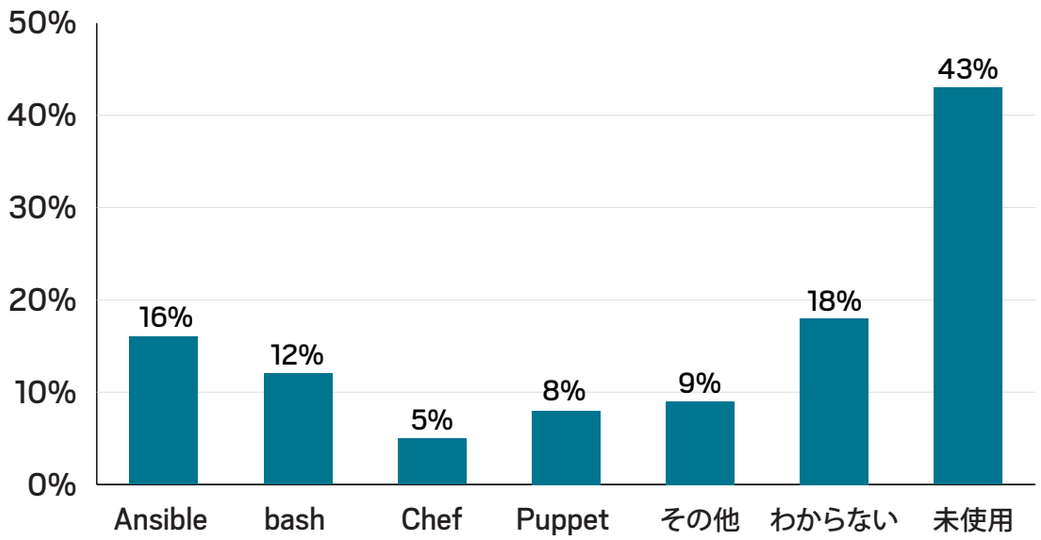
### 14.採用しているクラウドへのアプローチはどれですか。

コンテナが43%で先頭を走っていますが、VMも33%で後を追っています。比較的新しいテクノロジーとして、サーバーレス/FaaSが堂々たる登場を果たしており、1割弱の回答者がこのアプローチを採用しています。もう少し歴史の長いPaaSも10%となっており、ほぼ同じ割合です。3分の1の回答者は、いかなる形でもクラウドへのアプローチを行っていません。




**15.使っている継続的デプロイ・ツールまたはリリース自動化ツールはどれですか。**

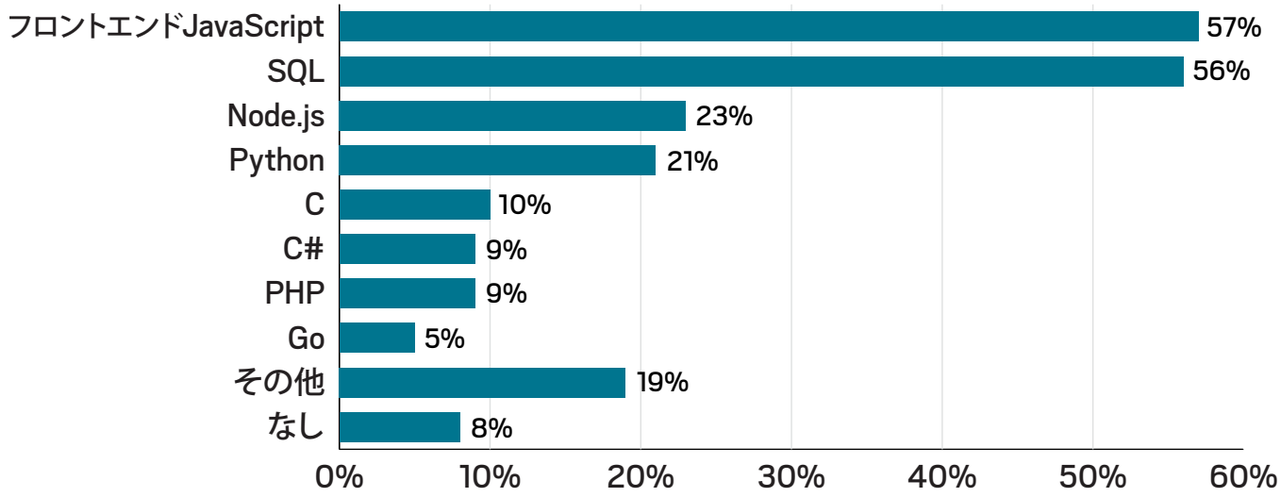
半数近くの回答者は、何の継続的デプロイ・ツールもリリース自動化ツールも使っていません。しかし、5分の1近くの回答者が、CDまたはリリース自動化にどのツールを使っているかわからないと答えているため、何のツールも使っていない人の割合は、実際にはさらに高くなる可能性もあります。ChefやPuppetよりもbashの方が使われているというのは、多少驚くことです。冗談でも何でもなく、bashは皆に愛されているのですから。CDツールの筆頭はAnsibleで、割合は16%となっています。



## アプリケーションについて

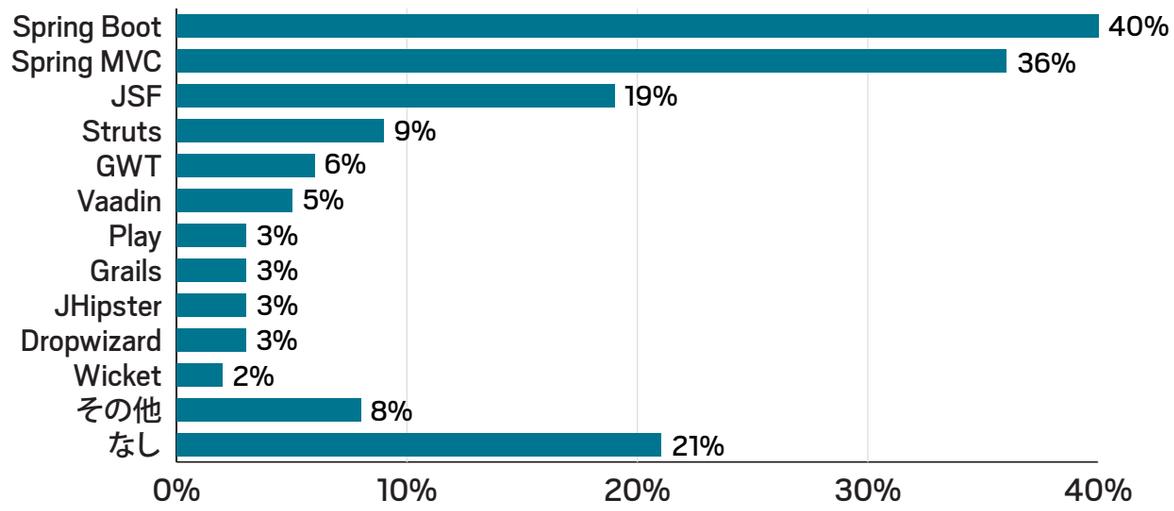
### 16.アプリケーションで使っている（JVM以外の）他の言語はどれですか。

現在の多言語の世界で、JVMアプリで使われるのはJVM言語のみだと考えるのは甘すぎるかもしれません。実際、半数を超えるJVMアプリケーションでフロントエンドJavaScriptが、5分の1でPythonが、4分の1近くでNode.jsが使われています。予想どおりかと思いますが、SQLも多くのプロジェクトで使われています。



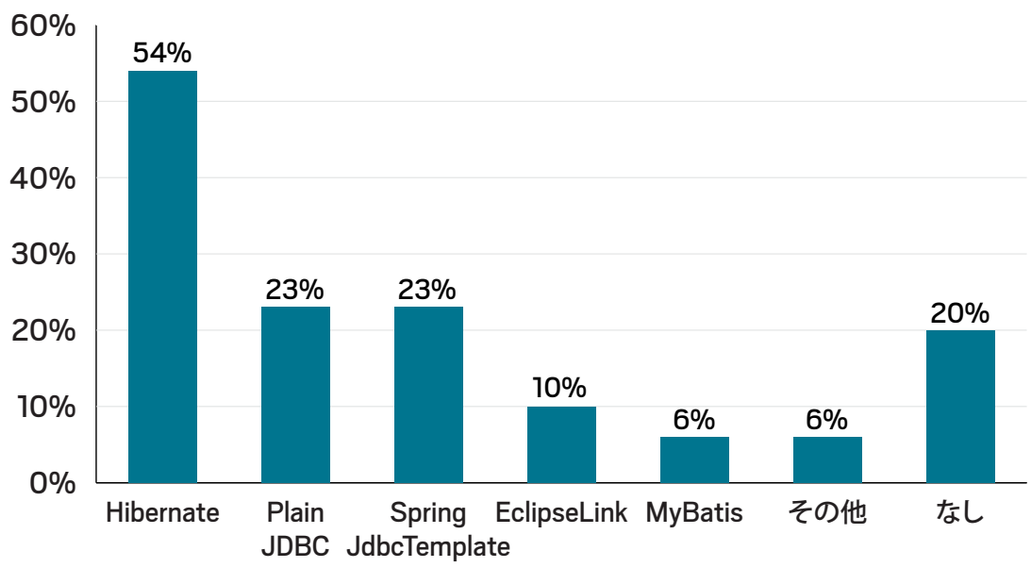

### 17.使っているWebフレームワークはどれですか。

SpringがJavaエコシステムを支配していることをこのグラフより雄弁に語れるものはほとんどないでしょう。4割の開発者がアプリケーションでSpring Bootを使っています。Spring Bootが初めてSpring MVCフレームワークを追い抜いたのは、興味深いことです。これらのフレームワークの次につけているのは、19%と健闘しているJSFです。リモート・コード実行の脆弱性が頻繁にニュースになっているにもかかわらず、Strutsが4位につけており、1割弱の開発者に採用されています。2割強の開発者はおそらく、Webフレームワークをまったく必要としないほど小さいアプリケーションであることを誇っています。



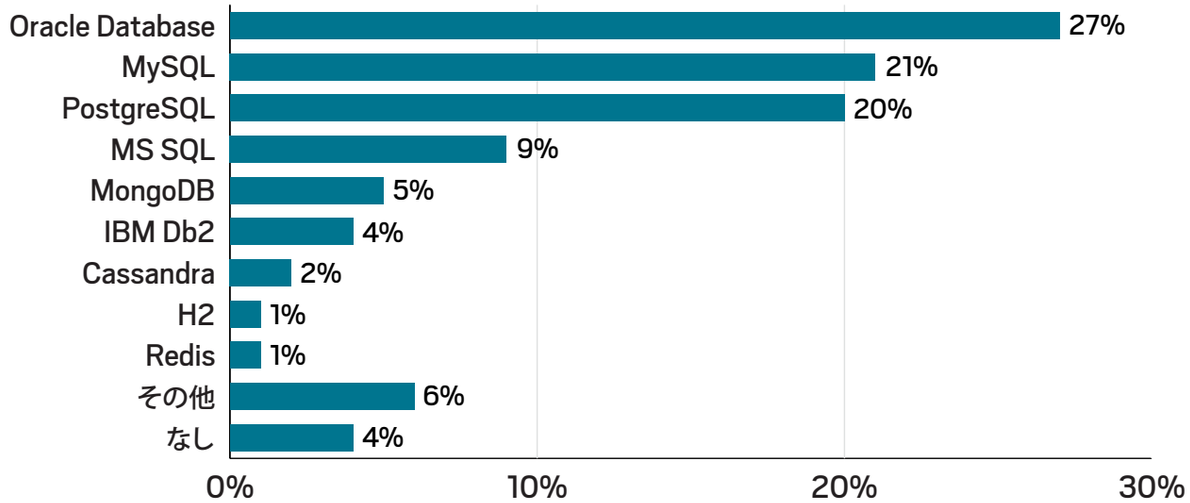

## 18.使っているオブジェクト・リレーショナル・マッピング（ORM）フレームワークはどれですか。

半数を超える開発者が、アプリケーションでHibernateを使っています。4分の1近くの開発者は従来のJDBCで満足しています。Spring開発者にとって、23%が使っているSpring JdbcTemplateは自然な選択肢となります。5分の1の開発者は、何のORMフレームワークも使わずにデータにアクセスしています（開発者は複数の回答を選択できるため、合計は100%になりません）。



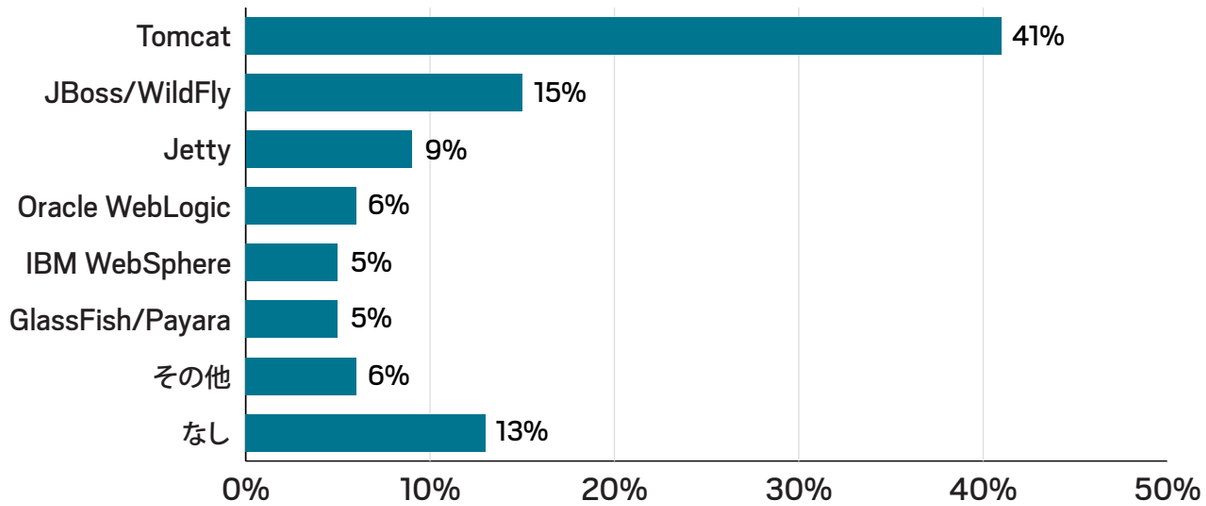
## 19.本番環境で使っているのは、どのデータベースですか。

ここでもOracle Databaseが首位に立っており、3割近くのアプリケーションの本番環境に使用されています。MySQLとPostgreSQLはその強力なライバルで、それぞれ21%と20%となっています。もっともよく使われているNoSQLデータベースはMongoDBで、5%という結果でした。




## 20.主要アプリケーションの本番環境で使っているのは、どのアプリケーション・サーバーですか。

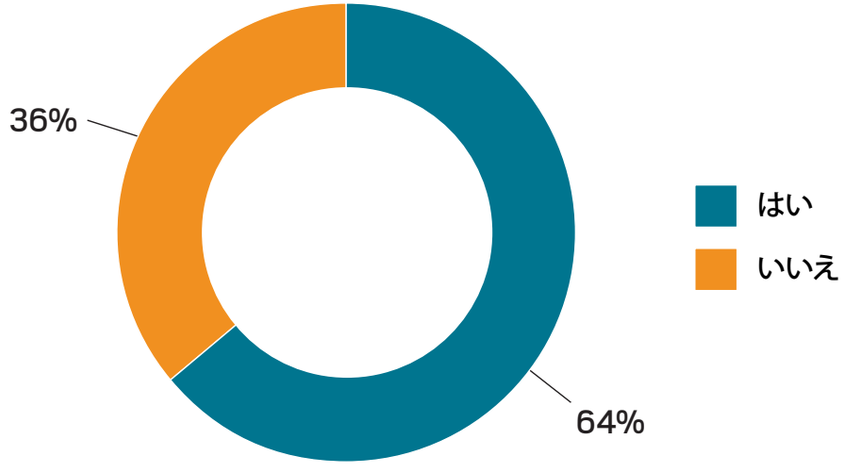
4割を超える回答者が、アプリケーション・サーバーとしてApache Tomcatを選択しています。このオープンソース・アプリケーション・サーバーは、高速、軽量でコミュニティにも愛されています。長いこと首位を走っており、こ



の傾向はしばらくの間変わりそうにありません。しかし、JBossおよびWildFlyも15%となっており、さほど引き離されているわけではありません。それよりも大規模なエンタープライズ・アプリケーション・サーバーでは、Oracle WebLogicが僅差でWebSphereを上回っています。「その他」の中でもっとも使われているのはApache TomEEとLiberty Profileで、それぞれ1%ずつとなっています。

## 21.本番環境で使っているアプリケーション・サーバーと同じものを使って開発していますか。

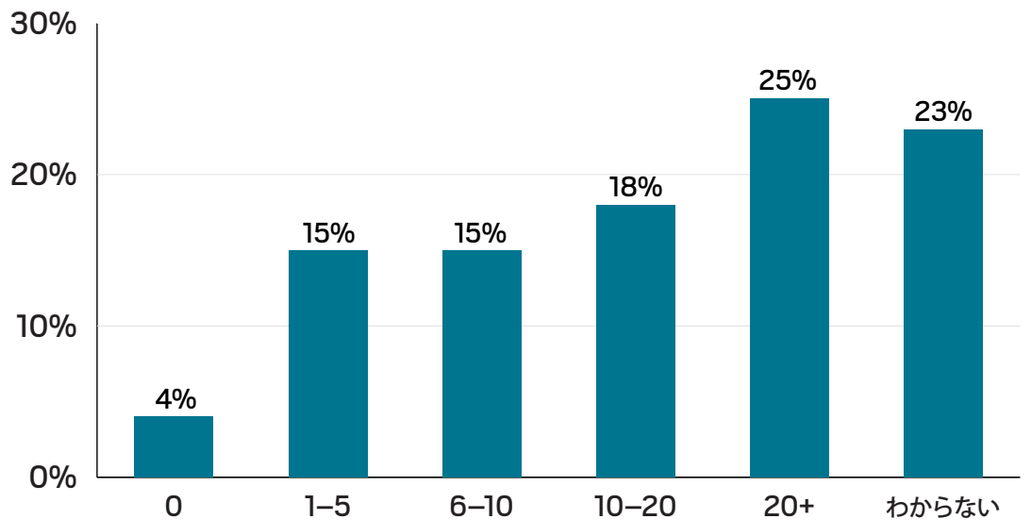
明らかなリスクがあるにもかかわらず、3分の1を超える回答者が本番環境とは違うサーバーで開発し、便利さと引き替えに、起き得る問題のコストを背負うことを選んでいます。驚くべきことに、開発環境で別のアプリケーション・サーバーを使っている（または何も使っていない）と答えた開発者は、本番環境で実に多様なアプリケーションやサーバーを使っています。ほとんどは開発者がローカルで使うのが難しい大規模なモノリシック・アプリケーション・サーバーではないかと考えていましたが、比率はほぼ同等でした。




## 22.主要なアプリケーションにオープンソースへの（直接）依存性はいくつありますか。

アプリケーションにいくつの直接依存性があるかを確認するために、チェックしなければならなかった人がどのくらいいたのかも気になるところです。おそらく、大半の人がそうだったことでしょう。直接依存性と推移的依存性の両方を尋ねなかったのは、よい選択だったと思います。実際、4分の1近くの回答者は、依存性の数がわからないと率直に答えています。これは、アプリケーションが複雑なビルド・システムに分散されていることが理由かもしれま

せん。結果から、オープンソースへの依存性がない回答者は20分の1未満ですが、72%という大半の回答者には、何らかの依存性が存在することがわかります。「わからない」という回答を除けば、95%、すなわち20人中19人の回答者がアプリケーションでオープンソースの依存性を使用していることもわかります。これは、オープンソースの採用がどれほど広がっているかを示しています。同時に、開発者はこういったサードパーティ製ライブラリによって、アプリケーションで求められるセキュリティ、品質、可用性が提供されることを確認する必要があることも示しています。

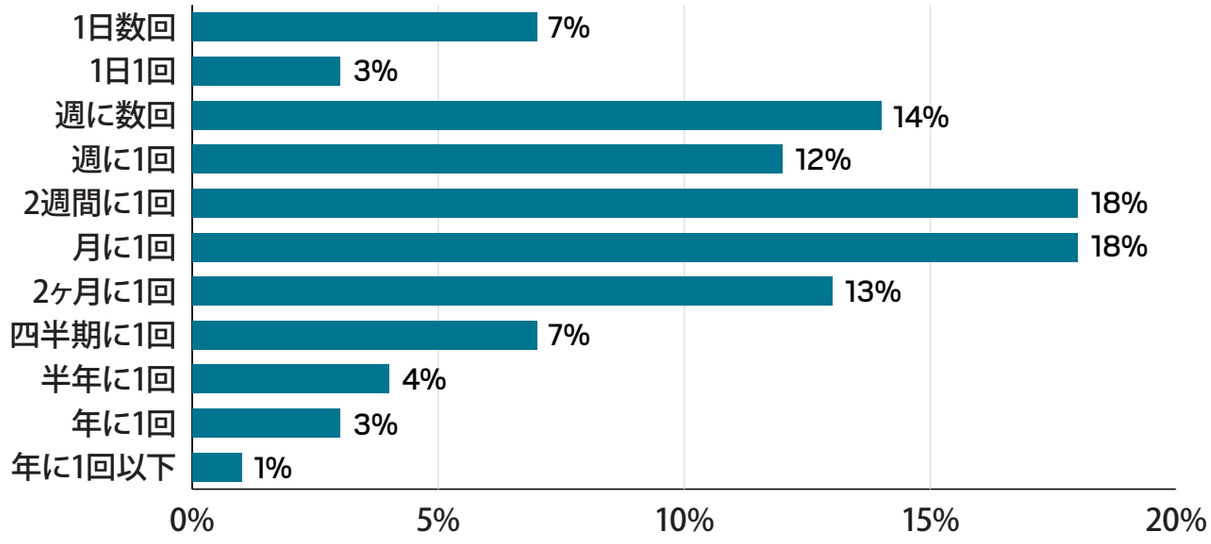



## プロセスについて

### 23.コードの新バージョンをリリースする頻度はどのくらいですか。

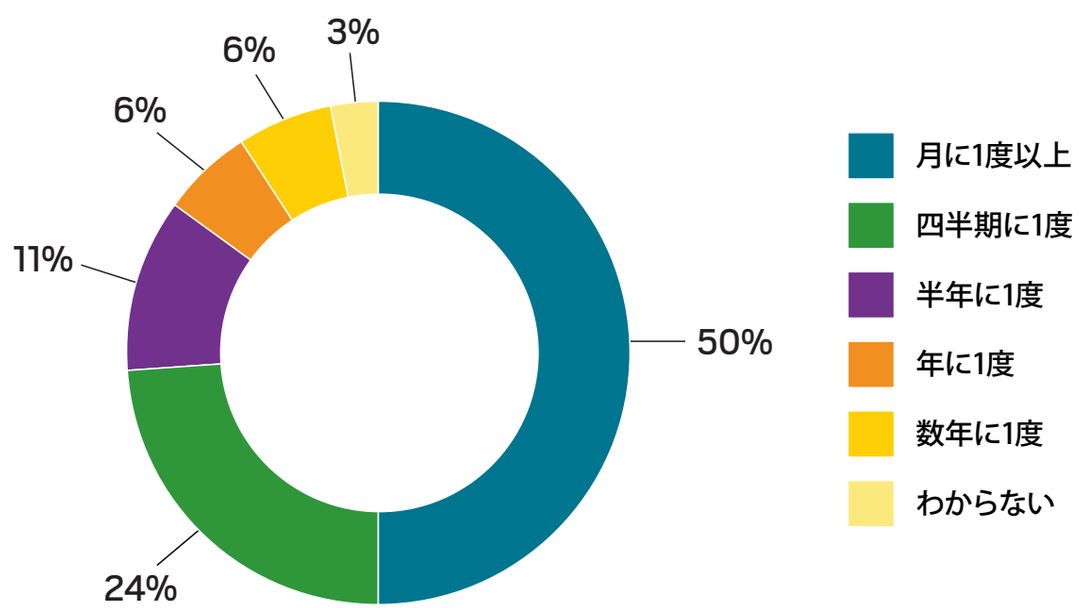
このグラフには示していませんが、4分の1近くの回答者（24%）が、コードがリリースされる頻度を知らないと回答しています。この点も、アプリケーションの複雑さが理由であるのかもしれません。また、サービスによってリリースのタイミングが異なるからかもしれません。

1割近くの回答者は、1日に複数回リリースしていると答えている強者たちです。しかし、別の観点から見た場合、この頻度でリリースすれば、大胆さは堅実さと見なされることでしょう。数週間から数か月に1回リリースしている回答者が多数派となっています。リリース間隔が半年以上である回答者は、1割未満です。




**24.コードの監査を行う頻度はどのくらいですか。**

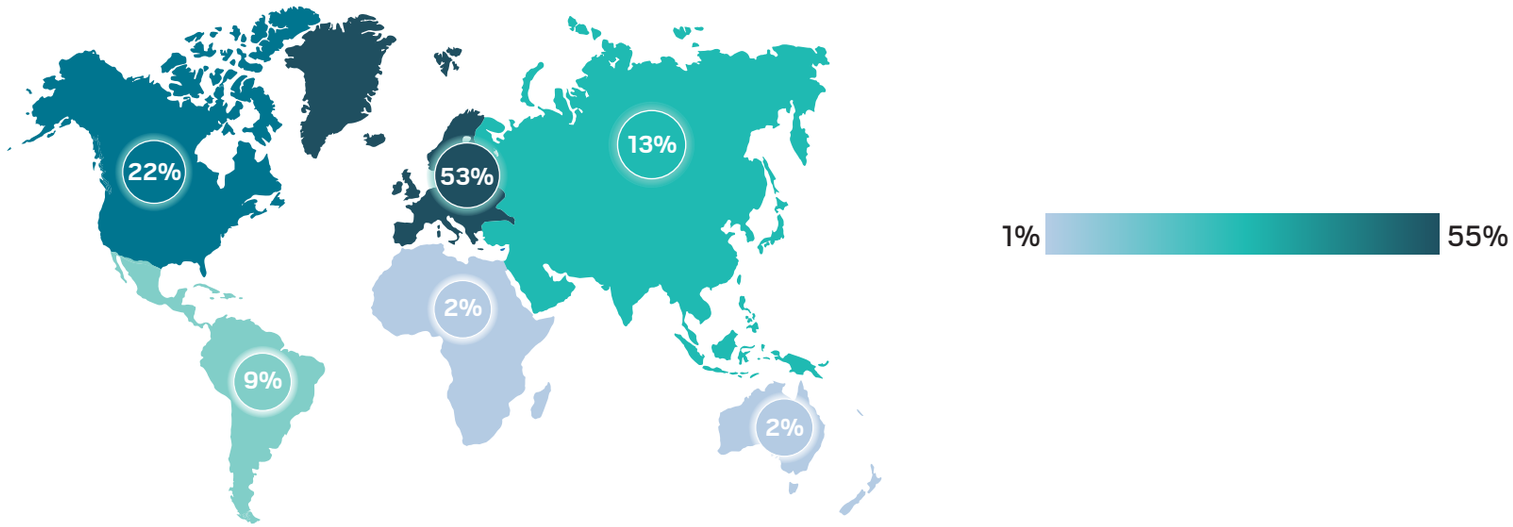
現場の約半分で、コードの監査が行われています。監査間隔が四半期に1回以上であるのは、4分の1にすぎません。監査がセキュリティ、パフォーマンス、品質のいずれのためであろうと、このような棚卸しをするのはよいことです。回答者の半分は、コードの監査を何も行っていません。そのようなコードベースの中には、どんなバグが潜んでいることでしょうか。



## 回答者自身について

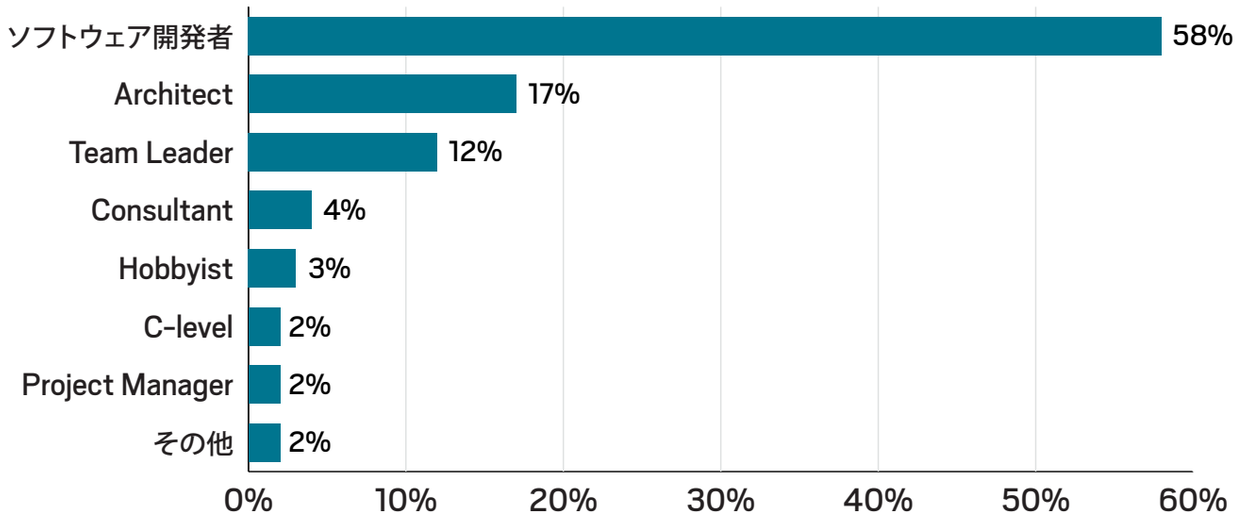
### 25.開発作業を行っている場所はどこですか。

北米からの回答者がもっと多いと予測していたこと以外、ここで述べることは特段ありません。



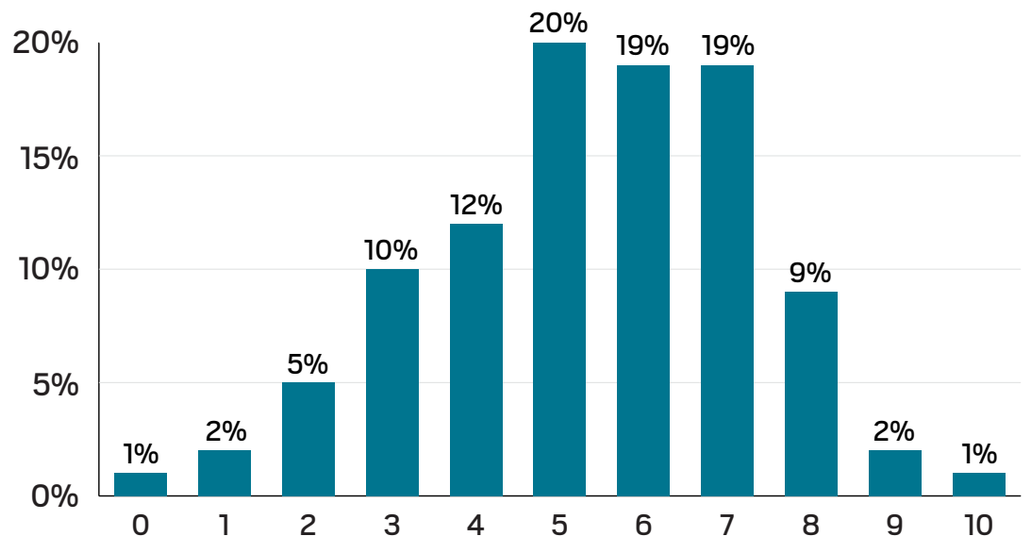

### 26.どのような職種に就いていますか。

回答者のほとんどは技術職で、87%は開発者、チーム・リーダー、アーキテクトのいずれかです。ソフトウェア開発者が半数を超え、回答者の2%がこの調査のためにわざわざ時間をとってくれた経営幹部レベルの方々です。



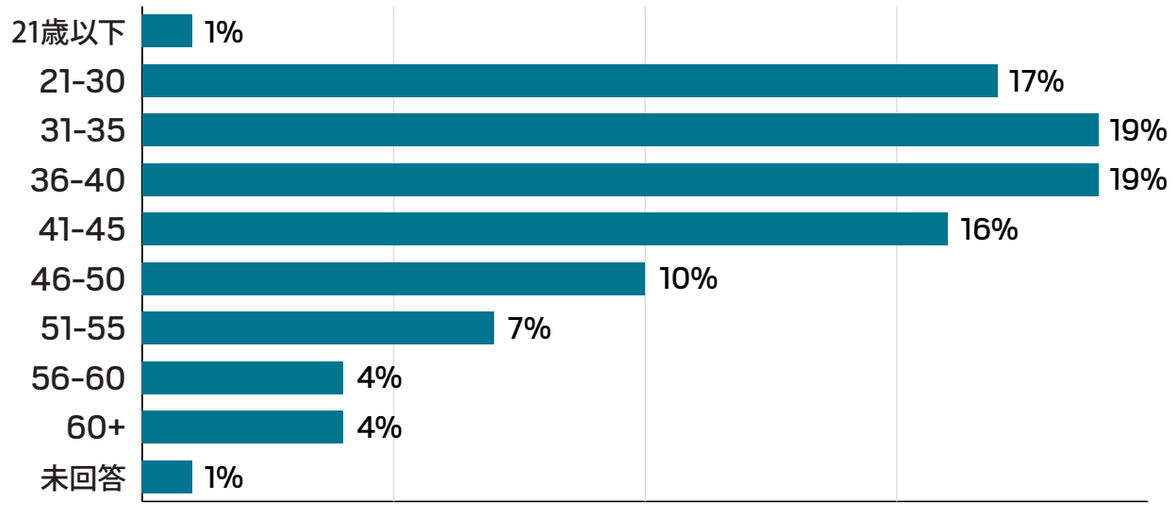
## 27.ご自身のセキュリティの専門度をどのように評価しますか。

セキュリティは、黒魔術のようなものと見なされることがよくあります。多くの開発者は、その場をしのいで担当する機能を提供するために「ちょうど足りる」くらいの知識しかありません。つまり、本物のセキュリティ専門家とは、セキュリティ専門のキャリアを積んでいる人々です。アプリケーション・セキュリティに関して開発者が担う責任は増しており、これが注目の話題にもなっています。今回の調査において、回答者の1%は、まったくセキュリティの知識を持たず、ただ楽しくアプリケーションを書いているだけだと答えています。10段階で10と答えた真のセキュリティ専門家も同じ割合でした。回答者の過半数、すなわち6割は5から7と回答し、専門家ではないが初心者でもないと答えています。



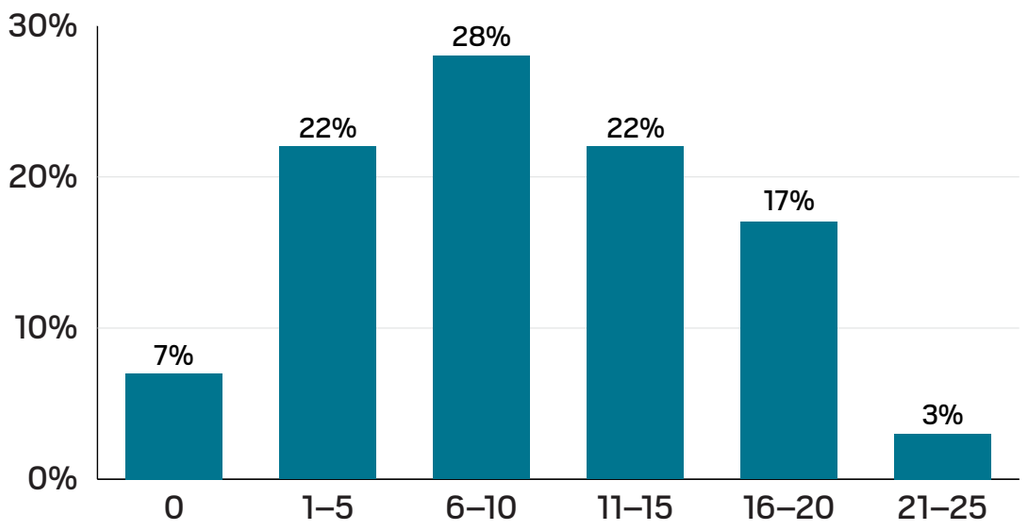

## 28.年齢はいくつですか。

プログラミングが、若者から中年前期の職業と見なされている点は変わっていません。調査回答者の37%が35歳以下、35%が36歳から45歳、25%が46歳以上でした。調査データから、責任の重い役職と年齢との間に相関関係があることがわかります。中年以降の割合が低いのは、プログラマーが管理職に移ることも一因でしょう。




## 29.報酬を得るプロフェッショナルとしてのJavaの経験年数はどれほどですか。

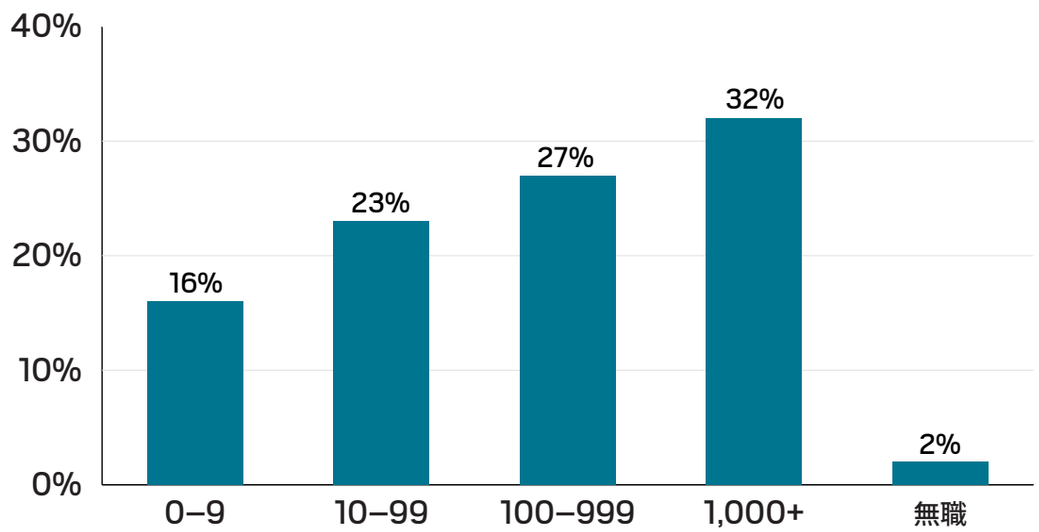
グラフからわかるように、回答者の経験は広範囲にわたっていました。しかし、私たちは、昇進する一般的な年齢を知るために、質問26にある職種ごとにそれぞれの年齢の中央値も知りたいと考えました。結果はとても興味深く、中央値は、開発者では10年、チーム・リーダーは11年、アーキテクトは14年、経営幹部レベルは12年の経験がありました。




## 30.会社の規模はどのくらいですか。

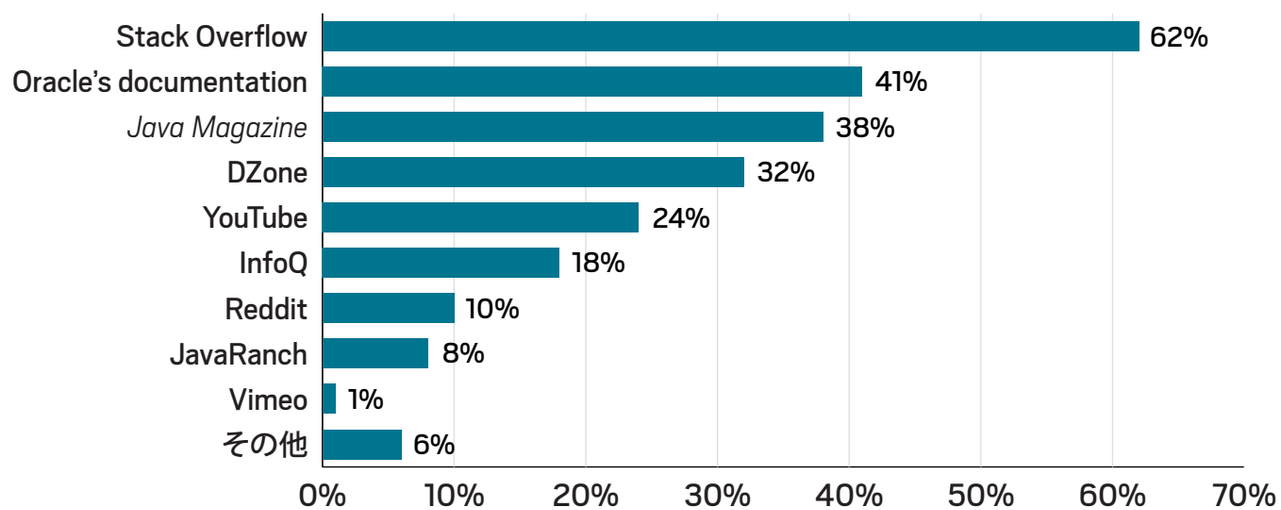
40%近くの回答者が従業員100人未満の会社で働いていることから、Javaが引き続き新興企業や中小企業で重要な役割を果たしていることがわかります。この点は、Javaがエンタープライズ・アプリケーション用の言語であると

認識されていることと矛盾します。これはもちろんそのとおりですが、エンタープライズ・アプリケーション用だけではないということも確かです。



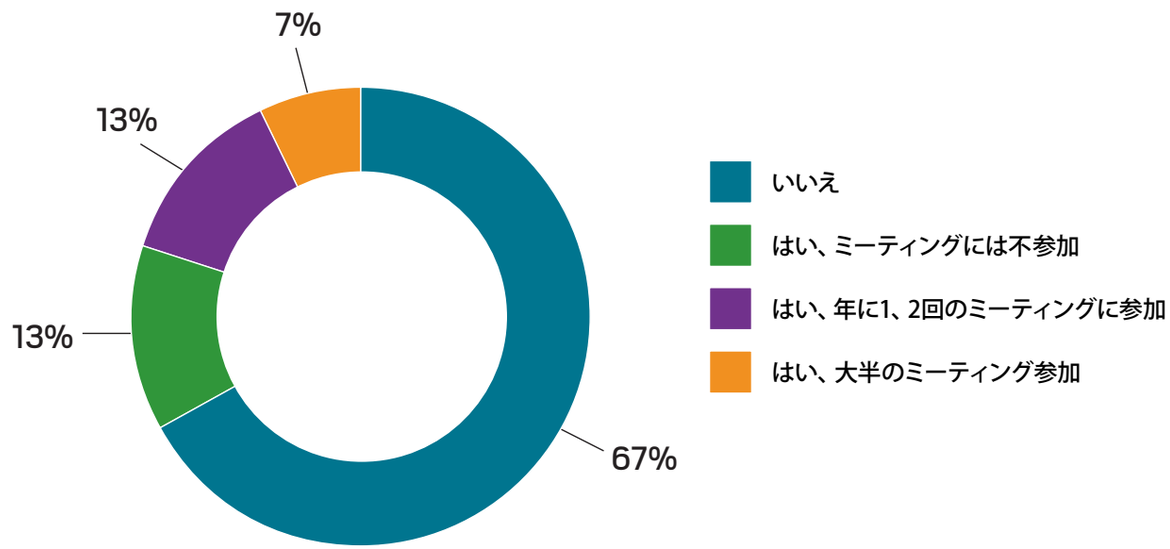

## 31.オンラインでは、Javaの情報をおもにどこから入手していますか。

単発で質問することや、以前の回答を検索して有用な情報を探すことができるフォーラムとして、Stack Overflowが好まれている点は変わりません。オラクルのすばらしいドキュメントが参照されているのも当然でしょう。Java Magazineでは、取り上げる話題について完全に理解したいという読者の方に向けて、長文で詳細な記事を提供しています。YouTubeでも、動画形式で同じようなことが行われています。DZoneもこうしたすべての領域をカバーしており、InfoQについてもある程度そう言えます（回答者は複数の回答を選択できます）。



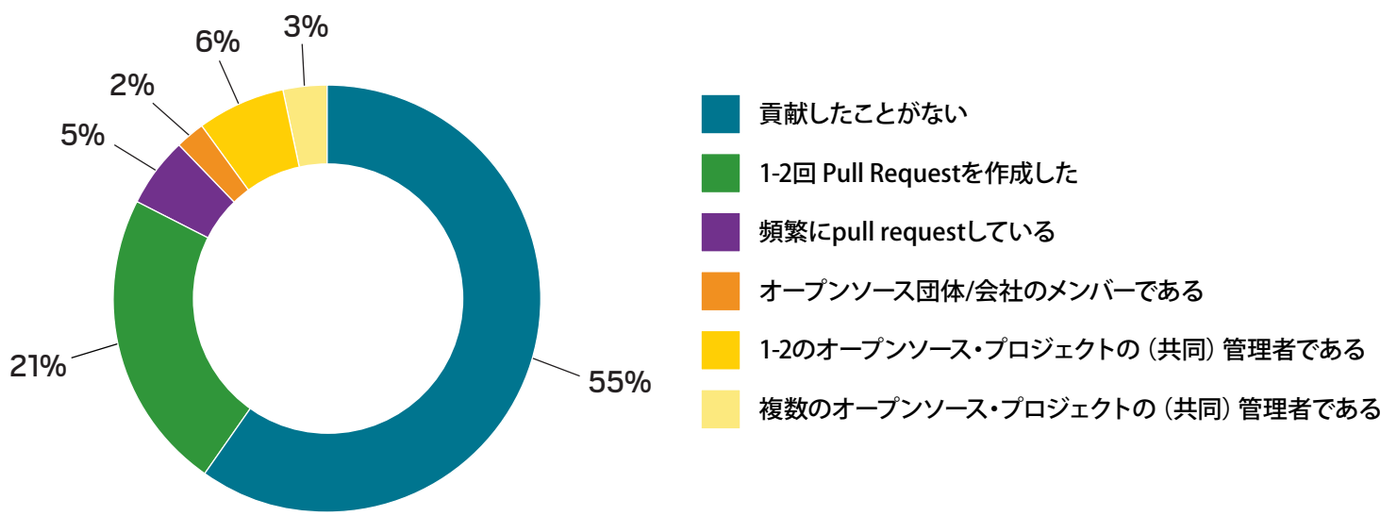
## 32.Javaユーザー・グループ（JUG）のメンバーですか。

1年に1回以上ミーティングに参加している開発者は、5分の1にすぎず、JUGがコミュニティであまり活用されていないリソースである点は変わりません。多数を占める3分の2の回答者は、どのJUGにも参加していません。その理由が地理的制約によるものという開発者の方には、Virtual Java User Groupがお勧めです。




## 33.どのくらいオープンソースに貢献していますか。

オープンソースは、登場したころの状態とさほど変わってはいません。すなわち、ごく一部の献身的な開発者による世界です。GitHubなど、簡単にアクセスできるコード・リポジトリによって開発者はコードに貢献しやすくなっています。しかし、オープンソース・プロジェクトに貢献したことがない開発者が半数を超えており、やるべきことはまだ多いと言えます。ぜひ貢献することを考えてみてください。そうすればいつの日か、オープンソースを使う、20人中19人の開発者の1人ではなく、オープンソースに貢献する、20人中の19人の開発者の1人になることができます。</article>




**Simon Maple**（@sjmaple）：Java Champion。プロジェクト依存性の脆弱性を検出して修正する製品を擁する会社、Snykでデベロッパー・アドボカシー担当ディレクターを務める。Virtual Java User Group（vJUG）の創設者でもある。以前はZeroTurnaroundでデベロッパー・リレーション担当ディレクターを務め、その前にはIBMで同様の職に就いていた。

**Andrew Binstock**（@platypusguy）：Java Magazine編集長。以前は、開発者向けの有名な雑誌およびWebサイトだったDr. Dobb'sを率いていた。その前はUNIX Review誌で、さらにその前にはC Gazetteで編集長を務めた。プログラミングに関する複数の書籍を執筆するとともに、オープンソースという用語が登場する前から、オープンソース・プロジェクトへの貢献を続けている。