JAVA TOOLS AND TECHNOLOGIES LANDSCAPE FOR 2014

A GLOBAL SURVEY OF 2164 JAVA PROFESSIONALS

Oh yeah, and each survey donated to charity!

REBELLABS

TABLE OF CONTENTS

Click to go to section...

EDITOR'S NOTE

This survey rocked. Better-than-ever response rate, and each completed survey donated cash to charity. We love doing this.

Did you know that this is the 5th anniversary of ZeroTurnaround's Java development research reports? Break out the champagne! It actually started in **2009** with a survey about the speed of Java application server restarts, evolving over the years into a deeper look at the industry layout in our first research publication in **2012**.

In **2013**, we tackled two particular challenges that really matter to productive software organizations: software quality (got bugs?) and the predictability of delivery (last week or next year?). We learned a lot from that one!

But now we feel like it's time to revisit the broader tools & technologies landscape in Java these days, collect some data, crunch some numbers and see what's going on in the market at large. And what better way than to just throw into your face

a huge leaderboard of tools and technologies currently running the show as of May 2014... KA-CHOW!

N.B. Please take these findings at face value, and use at your own risk. The sampling error calculated by **DSS** is 2.1%, but since the sample is self-selected it is not truly random and can contain bias error that we cannot measure, though we do provide the audience data and it seems to be representative of the industry.

OLIVER WHITE
Head of RebelLabs

**DEVELOPER
PRODUCTIVITY
REPORT**
JAVA TOOLS, TECH, DEVS AND DATA
2012
Can you handle it?

REBELLABS

**Developer Productivity
Report 2012**

**DEVELOPER
PRODUCTIVITY
REPORT 2013**
HOW ENGINEERING TOOLS & PRACTICES IMPACT
SOFTWARE QUALITY & DELIVERY

REBELLABS

**Developer Productivity
Report 2013**

Introduction: What's going on in Java these days?

Since there is a very good chance you, dear reader, use one of these top technologies, let's look straight at top tools & technologies represented in each of the 14 categories we asked about.

As you can guess, in some categories multiple tools are often used in conjunction, so we allowed for multiple selections (denoted by *). For answers where a statistically significant portion (over 5%) of respondents selected "Do not use", the responses have been normalized (denoted by °) to exclude non-user groups.

It probably comes as no surprise that among the 2164 developers we surveyed, **Java SE 7 (65%)** is used by two-thirds of developers, but even more are using **JUnit (82.5%)**, the most-used single technology across the entire Java landscape. And a good thing too: unit testing is key for making sure your app gets out the door. Next is **Jenkins (70%)**, our favorite Lord of the Butlers, which is used by nearly 3 out of 4 developers that use Continuous Integration tools (1 in 5 does not). We've seen distributed VCS come a long way in recent years, and **Git (69%)** is now non-exclusively used by over two-thirds of developers - often alongside **Subversion (57%)**.

Taking in the next set of tech leaders really completes the Enterprise Java picture - **Hibernate (67.5%), Maven and Nexus (64%), Tomcat (50%)** and **Eclipse (48%)** gives you more or less a decent foundation of a basic, no frills enterprise development stack.

But don't think the last words have been had yet...because in this report we asked a few questions that directly highlight the feelings of developers towards certain technologies.

The 2014 Leaderboard
of Java Tools & Technologies

JUnit - 82.5%* - Top testing framework used by developers

Jenkins - 70%° - Most used CI server in the industry

Git - 69%* - #1 version control technology out there

Hibernate - 67.5%*/° - The top ORM framework used

Java 7 - 65% - The industry leader for SE development

Maven - 64% - Most used build tool in Java

Nexus - 64%° - The main repository used by developers

MongoDB - 56%° - The NoSQL technology of choice

FindBugs - 55%*/° - Most-used static code analysis tool in Java

Tomcat - 50%° - The most popular application server on the market

Java EE 6 - 49%° - Found in the most enterprise Java environments

Eclipse - 48% - The IDE used more than any other

Spring MVC - 40%*/° - The most commonly used web framework

MySQL - 32%° - The most popular SQL technology

THE 2014 LEADERBOARD
OF JAVA TOOLS & TECHNOLOGIES

REBELLABS

82.5%
JUnit*
TOP TESTING FRAMEWORK USED BY DEVELOPERS

70%
Jenkins°
MOST USED CI SERVER IN THE INDUSTRY

64%
Maven
MOST USED BUILD TOOL IN JAVA

64%
Nexus°
THE MAIN REPOSITORY USED BY DEVELOPERS

67.5%
Hibernate*/°
THE TOP ORM FRAMEWORK USED

65% Java 7
THE INDUSTRY LEADER FOR SE DEVELOPMENT

56%
MongoDB°
THE NOSQL TECHNOLOGY OF CHOICE

55%
FindBugs*/°
MOST-USED STATIC CODE ANALYSIS TOOL

50%
Tomcat°
THE MOST POPULAR APPLICATION SERVER

49%
Java EE 6°
FOUND IN THE MOST ENTERPRISES

69% Git*
#1 VERSION CONTROL TECHNOLOGY OUT THERE

48%
Eclipse
THE IDE USED MORE THAN ANY OTHER

40% **Spring MVC***/°
MOST COMMONLY USED WEB FRAMEWORK

32% **MySQL°**
THE MOST POPULAR SQL TECHNOLOGY

*Multiple selections possible   ° Normalized to exclude non-user base

Sample population of 2164 Java professionals, sample error 2.1%

What technologies are developers really interested in?

We asked which other JVM language they would be most interested in learning about (**Scala - 47%**), what IDE they'd rather use (**IntelliJ IDEA Ultimate - 49%**) and build tool they'd like to learn about (**Gradle - 58%**).

Java 8 is expected to be the #1 priority of 35% of respondents' companies over the next 2 years. Interestingly, responses regarding IDEs - one of the most used developer tools out there - betray the open-source nature of this tool group by clearly preferring the commercial version of IntelliJ IDEA.

Basically, there are great indications that these next 4 technologies have already become a force to be reckoned with (more on this later).

Look out for these technologies...

WHAT TECHNOLOGIES ARE DEVELOPERS REALLY INTERESTED IN?

Gradle - Almost 6 in 10 developers say they want to learn more about this build tool.

IntelliJ IDEA - Almost half of developers would rather use IntelliJ than any other IDE.

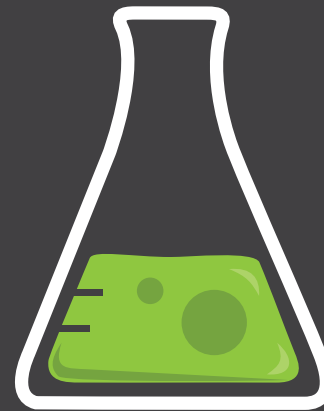**Scala** - 47% of developers would choose Scala as their next JVM language.

Java 8 - Over 1/3 of developers see getting familiar with Java 8 as their highest priority until 2015.
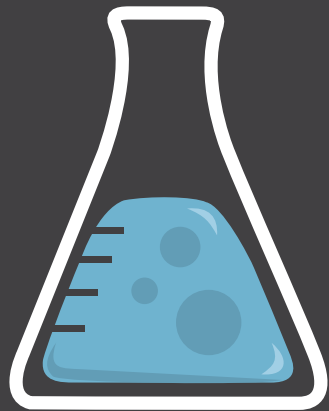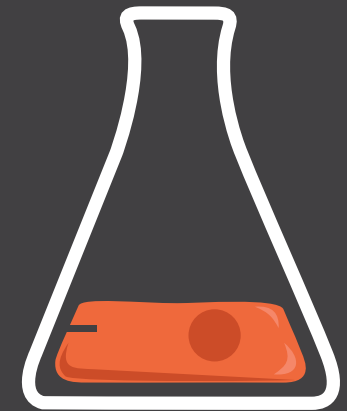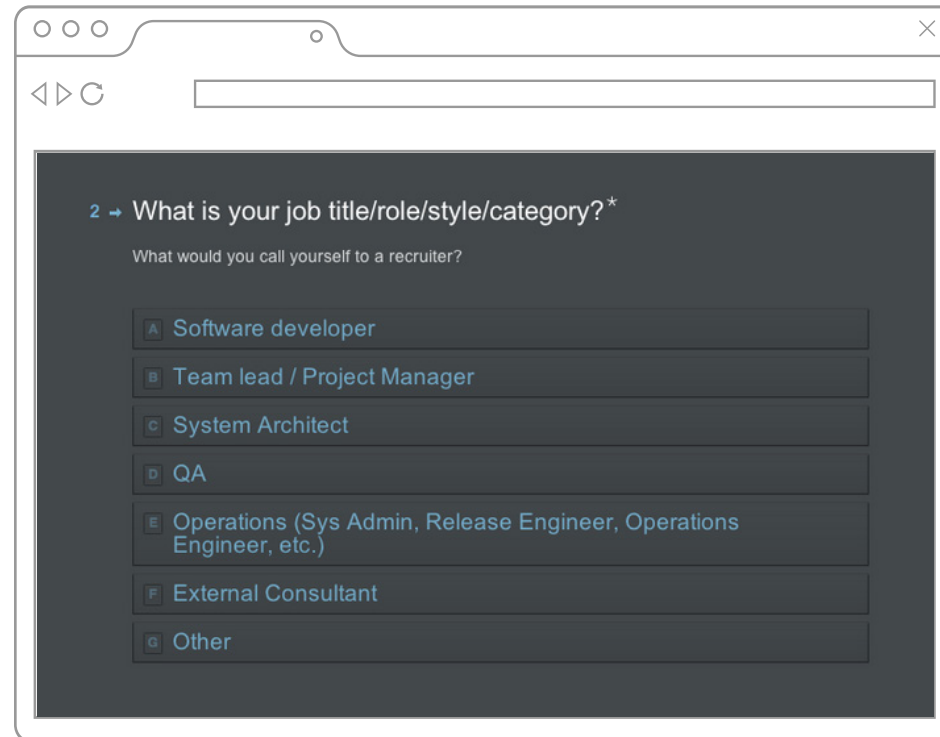
58%
Gradle

49%
IntelliJ IDEA

47%
Scala

35%
Java 8

REBELLABS

How we ran the survey ~~the survey~~

Before we look at the numbers, let's look at how we ran the survey, who replied and how we got them to share their information with us.

It started off with admitting that taking surveys is never really that thrilling, right? At best, they are just quick and don't ask you about something overly personal. For the first time, we were confident that this survey would get a better response than previous ones. Why? Because we fought to make it better, and focused on improving the UX by:

- **Finding the best audience to ask**
- **Introducing a meaningful incentive**
- **Using the best survey tool possible**

2 → What is your job title/role/style/category?*

What would you call yourself to a recruiter?

A	Software developer
B	Team lead / Project Manager
C	System Architect
D	QA
E	Operations (Sys Admin, Release Engineer, Operations Engineer, etc.)
F	External Consultant
G	Other

And the **audience says...**

RebelLabs has grown since our premier launch in Jan 2013, and that includes a sizable audience of engaging, intelligent geeks that obviously care about learning and sharing with others. In addition to asking RebelLabs subscribers, we asked Simon Maple, founder of Virtual JUG, to spread the word there and among the wider user group community - including within the helpful London Java Community (LJC). After that, we let friends and fans spread the word on Twitter, Facebook, Google +, LinkedIn and DZone...but it wouldn't have done so well, we think, without a proper incentive that actually matters...

Finding the **right incentive** to participate

Although many organizations continue to do this, giving away an iPad, Kindle or remote-controlled helicopter isn't the most effective way to engage geeks. Firstly, software developers are comparatively high-income earners, and geeks with money usually get themselves these toys whenever they want long before companies figure out that it might be cool to give away.

Second, the privileged that don't need or want expensive gifts are more reluctant to respond to actions for personal gain. They are good people, after all: so why not do something that requires geeks to be able to give something to others? In this case, we pledged that for every completed survey we would provide $0.50 USD to Child's Play, a charity that donates games and entertainment systems to children in hospitals. Boom.

Make it fun with **great survey software**

Another important part of this was to try to make the survey a bit FUN! With the tools we'd used before, like Survey Monkey and Poll Daddy, we saw great functionality and complex logic overcompensating for simply looking good and feeling easy to use or intuitive. When searching for something else, we quickly came across Typeform, a survey tech startup based in Spain, that told us we could make a survey great-looking, and customize it quickly with our own graphics and color scheme.

Within 5 minutes of trying it, we felt that we would better engage respondents respondents simply by offering them a good-looking form to fill out: the survey flows well, looks great on mobile devices, lets you use keystrokes to select options, etc.

Anyway, 2164 software professionals completed the survey, generating over $1082 for a charity we are happy to support. So let's see who they are, and and what tools and technologies they use.

SAMPLE POPULATION
DEMOGRAPHICS

" *Yep, it falls to me to present the least memorable of all these stats: The majority of our 2164 respondents are software developers, working on web-based applications, using Android phones and Windows. Who would have thought? :-D* "
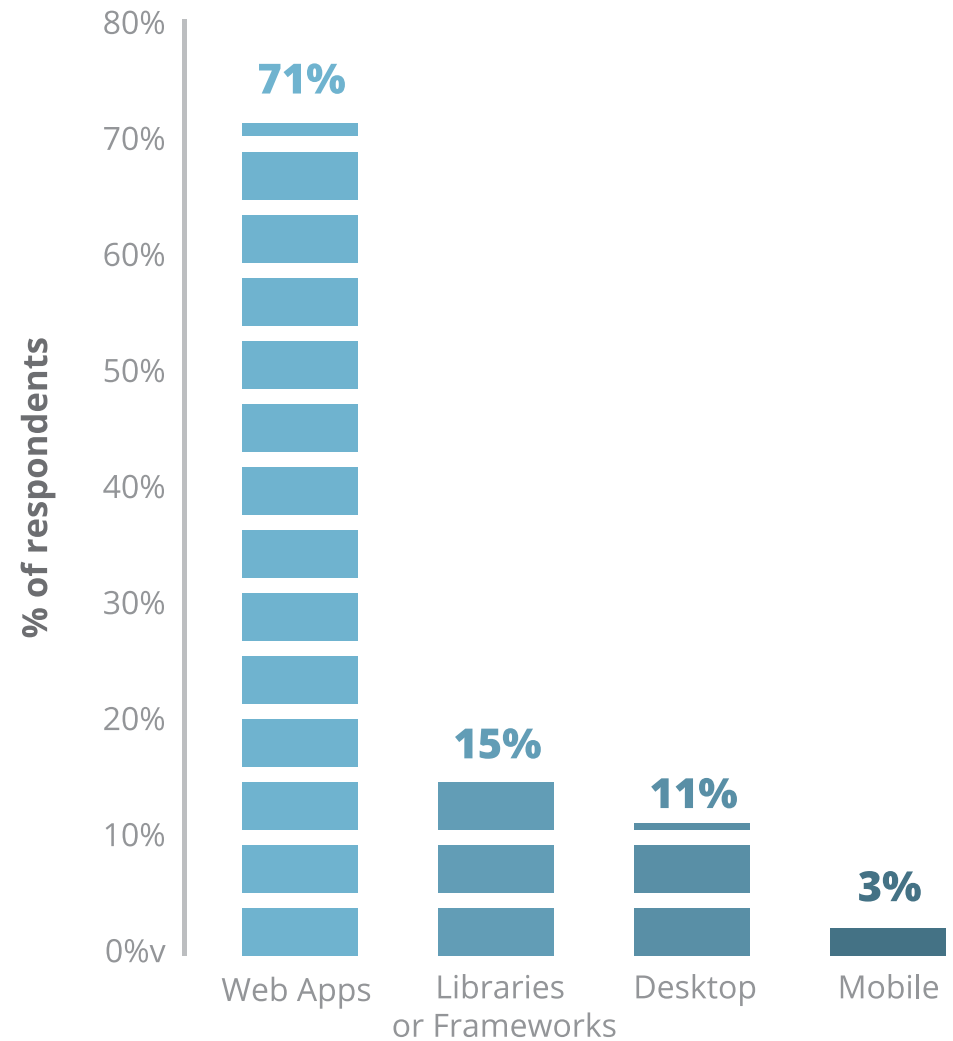
OLIVER WHITE,
Head of RebelLabs

Before we determine what you're using out there, it's good - or at least polite - to find out a little bit about who you are and what you're building.
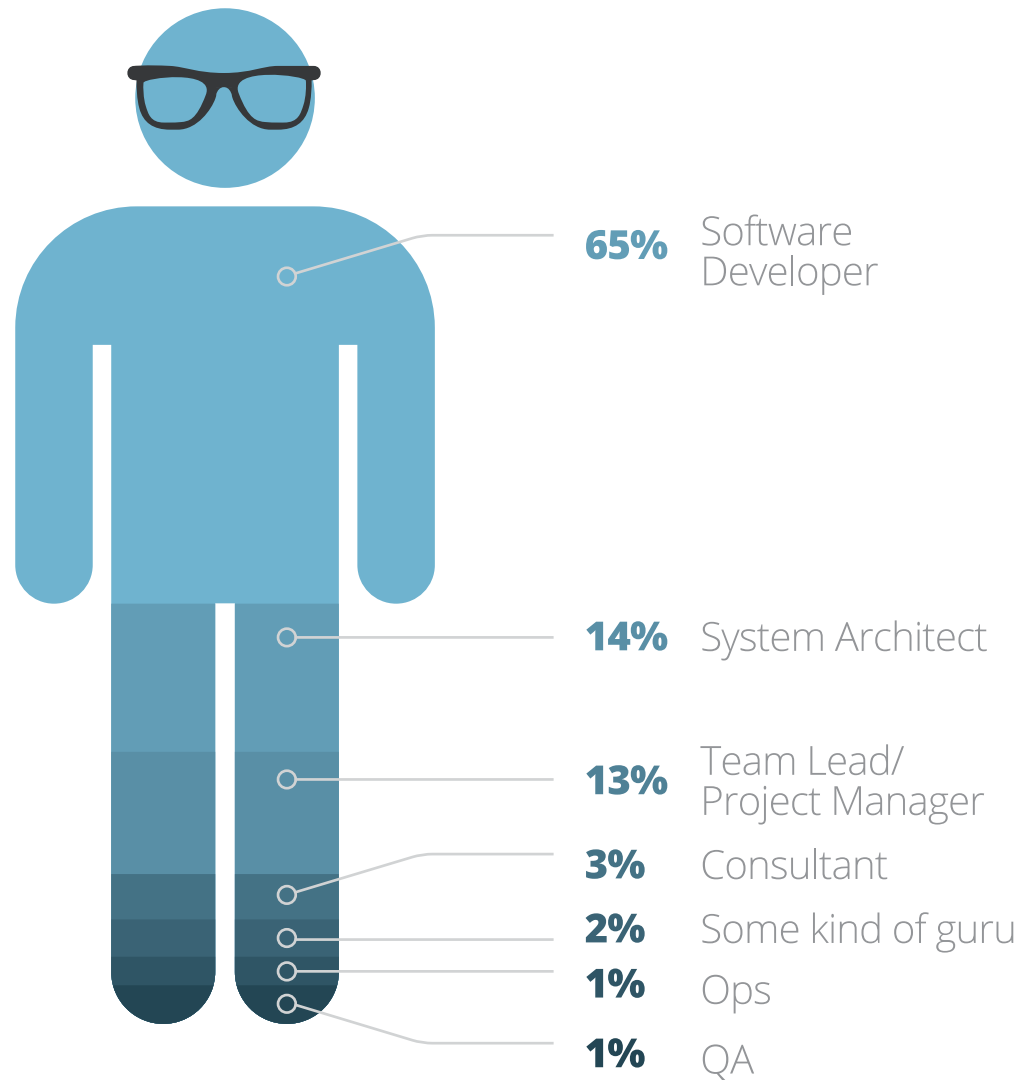
Over a one-month period, a total of 2164 software professionals responded to our survey. We tried digging in to see if there were any significant correlations hidden in the demographics - such as application type or mobile device OS variance based on job profile, but the numbers didn't betray anything truly interesting.

It's not surprising to see the dominance of **web applications (71%)** in development these days, since that's pretty much aligned with visible trends. But if you step back and think about it, who the heck is making all these **libraries and frameworks (15%)**? That's 1 in 7 developers. Weird. **Desktop apps (11%)** continue to decrease in abundance as easier-to-try, no-download alternatives in the cloud continue to propagate. Lastly, there are a few out there creating **mobile applications (3%)**. Presumably, the majority of mobile app developers are selecting not Java ME (SE embedded), but rather Dalvik (Android) or iOS.

What are you building?

% of respondents

Web Apps	71%
Libraries or Frameworks	15%
Desktop	11%
Mobile	3%

REBELLABS

What do they call you?

65% **Software Developer**

14% System Architect

13% Team Lead/ Project Manager

3% Consultant

2% Some kind of guru

1% Ops

1% QA

In terms of job role/position/title, **software developer (65%)** take up nearly two-thirds of respondents. More than a quarter of respondents are in more senior positions as either **system architects (14%)** or **team leads / project managers (13%)**, and the rest are consultants (i.e. likely independent contractors), Ops, QA or other (some kind of specialist guru, "DevOps Architect", "Chief Code Officer" or something else equally colorful).

REBELLABS

What Operating Systems do you use?

OS	%
Microsoft Windows	50%
Linux/Unix	26%
Apple OS X	24%

0% 10% 20% 30% 40% 50% 60% 70%

OS	%
Google Android	66%
Apple iOS	27%
Other	7%

70% 60% 50% 40% 30% 20% 10% 0%
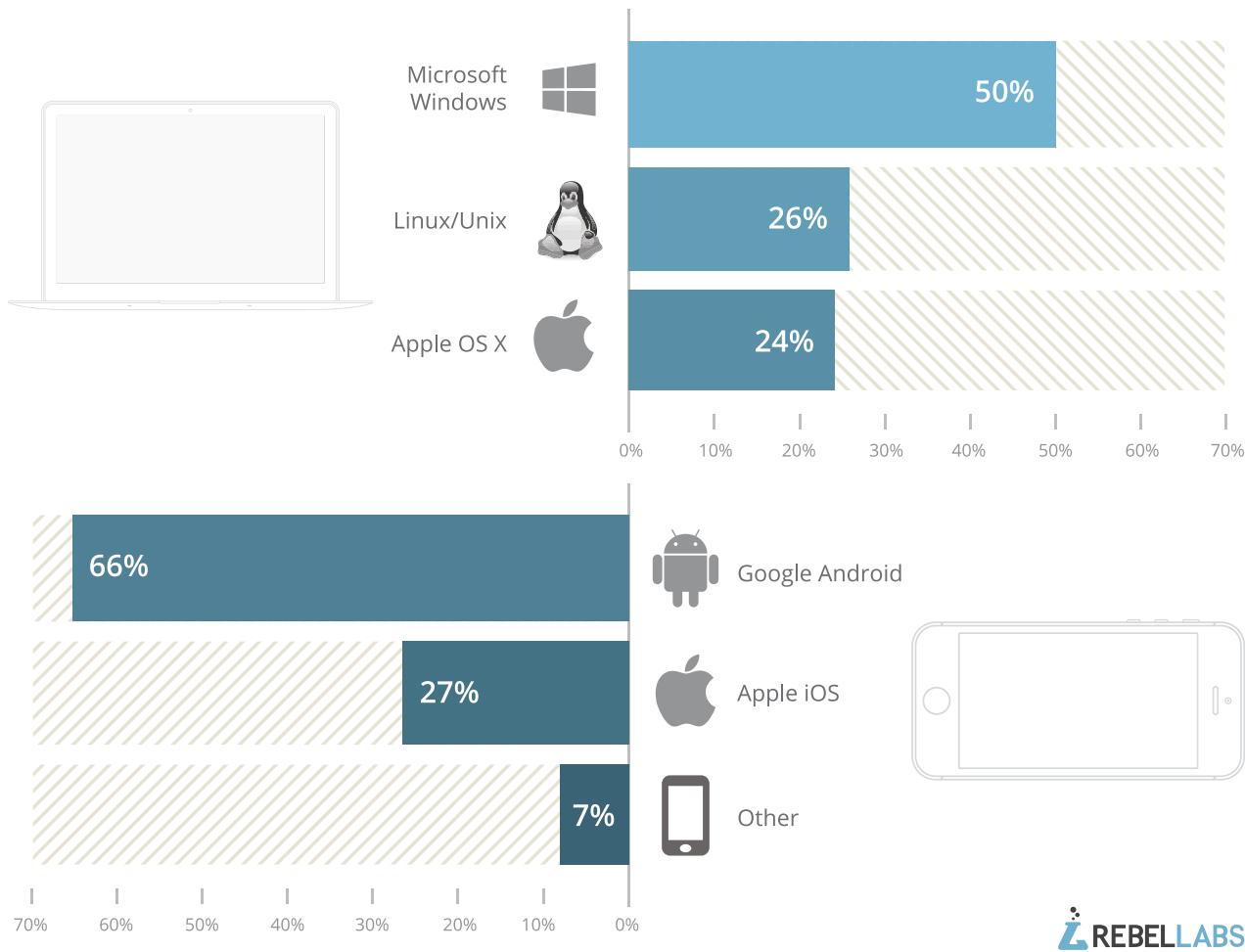
The questions that asked about operating system for both respondents' workstations and mobile phones was simply for fun - there is nothing ground-breaking here: **Microsoft Windows (50%)** is used by half of respondents, and **Linux/Unix (26%)** and **Apple OS X (24%)** more or less evenly split the remainder although it's curious to speculate what will happen to workstations in the next 5 years. For mobile OS, **Google Android (66%)** commands two-thirds of the market, while the priorietary and more expensive **Apple iOS (27%)** is used by slightly more than 1 in 4. **Other mobile OS (7%)**, meaning Blackberry and Windows Mobile, make up the rest, but apparently we don't really care. ;-)

NOTE:
*We asked about open source project committers out there, and really surprisingly high **31% of developers** reported that they work on open source projects, which says more about bias towards these type of exceptional software professionals with whom we are able to cooperate rather than a general indication of the industry...although it IS a promising indicator nonetheless! :-)*

JAVA VERSIONS

" *Java EE 6 was a good solid release that gave parity with the Spring suite, I'd expect to see many enterprises sticking with it unless they really want standardised web socket & JSON support. The number of developers not using Java EE doesn't surprise me, there are a large class of applications and architectures (e.g. low resource, horizontally scaling micro services) that Java EE simply can't serve.* "

MARTIJN VERBURG,
CEO of jClarity

WHO IS USING JAVA EE?

68% Java EE (+SE)
32% Java SE only

Java EE versions in use*

- **49%** Java EE 6
- **35%** Java EE 7
- **11%** Java EE 5
- **5%** J2EE

Java SE versions in use

- **65%** Java 7
- **26%** Java 6
- **7%** Java 8
- **2%** Java 5
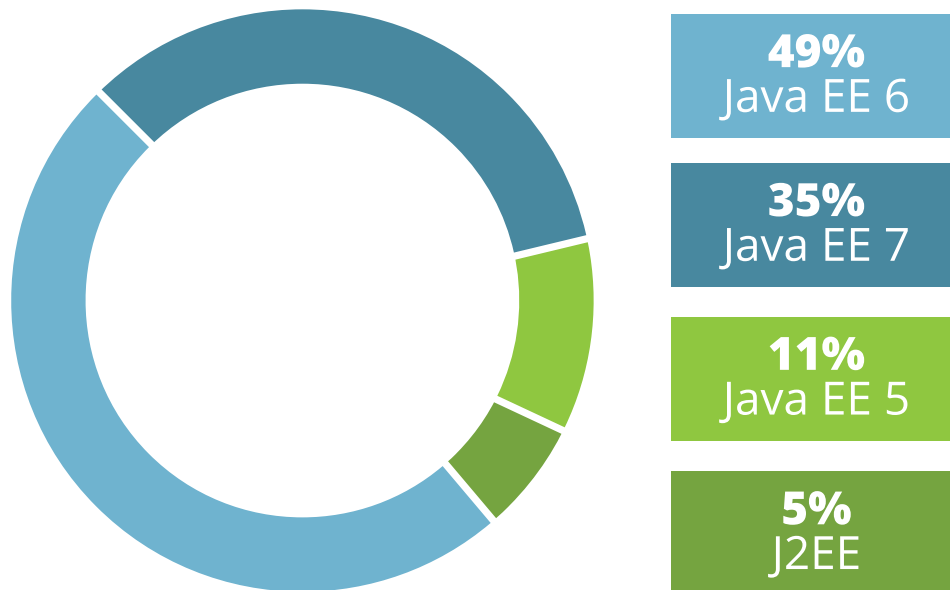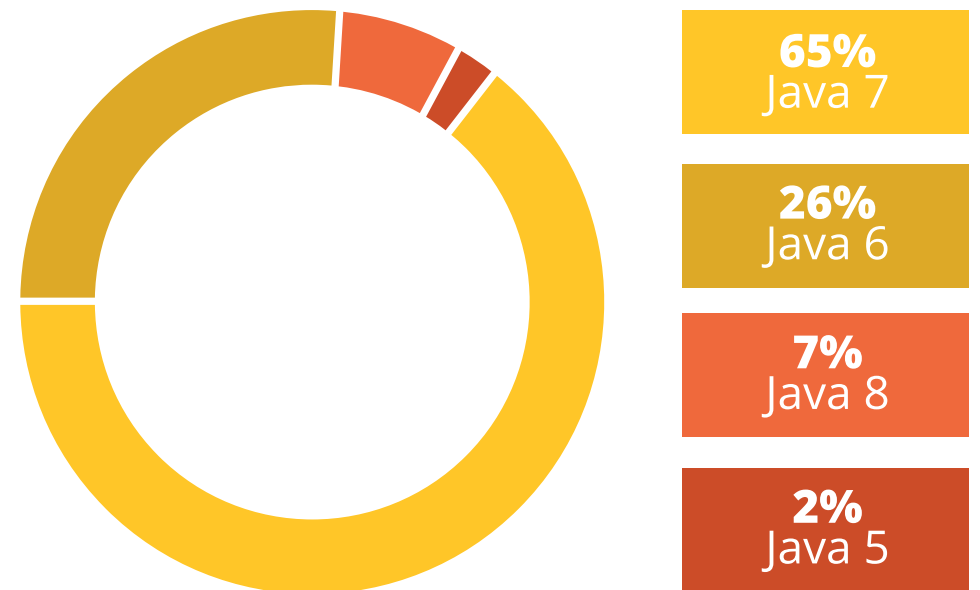
* The results were normalized to exclude non-users

REBELLABS

> Java has been called "dead" for years. But it's vital and new language versions get adopted with the industry-typical restrained uptake. For both Java SE and Java EE, nearly 2/3 of the participants are on the latest two releases. The faster uptake on the EE side clearly expresses how important developer productivity and ease of use are today.
>
> – MARKUS EISELE,
> *Java EE evangelist and Java champion*

Java is dead. Long live Java.

Java **SE**

Java SE 7 (65%) leads the way here, which is no surprise, and although **Java SE 6 (26%)** has dropped from 88% usage in 2012, however it's unnerving that 1 in 4 developers still clings to it - end of life from Oracle's side was February 2013.

Java SE 8 (7%) penetration, even with all the lambda madness, is surprisingly high - considering only a few app servers support it and big technologies like Hibernate and Javassist aren't fully supporting Java 8 as of the time of this writing.

Java **EE**

The majority of respondents (68%) are using Java EE, so we are confident that they represent the "enterprise Java development" segment. **Java EE 6 (49%)** perhaps unsurprisingly is used by 1 in 2 developers, and **Java EE 7 (35%)** usage seems to have grown well since its release, where only particular app servers support it - JBoss and GlassFish namely - and true to that, we see a surging 2x increase in GlassFish usage by Java EE 7 developers alone (JBoss doesn't change much).

We note that 1 in 6 developers is still stuck with **Java EE 5 (11%)** or its predecessor **J2EE (5%)** - given how difficult it can be to migrate legacy apps away from these veteran specs, this is not a complete shock.

*For further reading, we recommend Java 8 Revealed: Lambdas, Default Methods and Bulk Data Operations **by Anton Arhipov.***

MOST INTERESTING ALTERNATIVE
JVM LANGUAGES

" *It's exciting to see Scala topping the list of JVM languages people want to learn more about. Scala really is the language with the most similarities to Java (static typing, object-model, etc.) that has the most significant improvements (pattern matching, traits, first class functions); this means learning Scala for the JVM can be a very enriching experience and I think a lot of developers are drawn to that in their hunger to grow.* "

JOSH SUERETH,
Geek at Typesafe

The next JVM language to learn

2% Xtend

2% Ceylon

2% Kotlin

4% Others

12% Clojure

47% Scala

31% Groovy

REBELLABS

For further reading, check out *The Adventurous Developer's Guide to JVM Languages.*

"If you had to choose just one additional JVM language to learn about, it would probably be..?" was a fun question to ask, since the results don't significantly affect terribly much in the world of Java. That said, some trends are visible: after Java, the real JVM-based contenders out there have always been Scala and Groovy, and this year we see these two pop up once again.

Statically-typed **Scala (47%)** and dynamic **Groovy (31%)** might do things quite differently, but both remain interesting to Java developers. Considering the professional support and community strength behind Scala by Typesafe, it's easy to see the Scala ecosystem continue making gains into enterprises as a serious alternative to Java. Groovy is also a very popular choice among JVM developers - coincidentally Groovy is the DSL used in Gradle, another popular technology among those we asked.

Clojure (12%), the functional language built by the elusive Lisp creator Rich Hickey, remains a fascination for Java developers as well. It's not even in the top 100 programming languages according to the TIOBE index, yet 1 in 8 Java developers would choose it as their next language to learn. **Kotlin (2%)**, **Ceylon (2%)** and **Xtend (2%)** each have small followings, but nothing to rival the leaders.
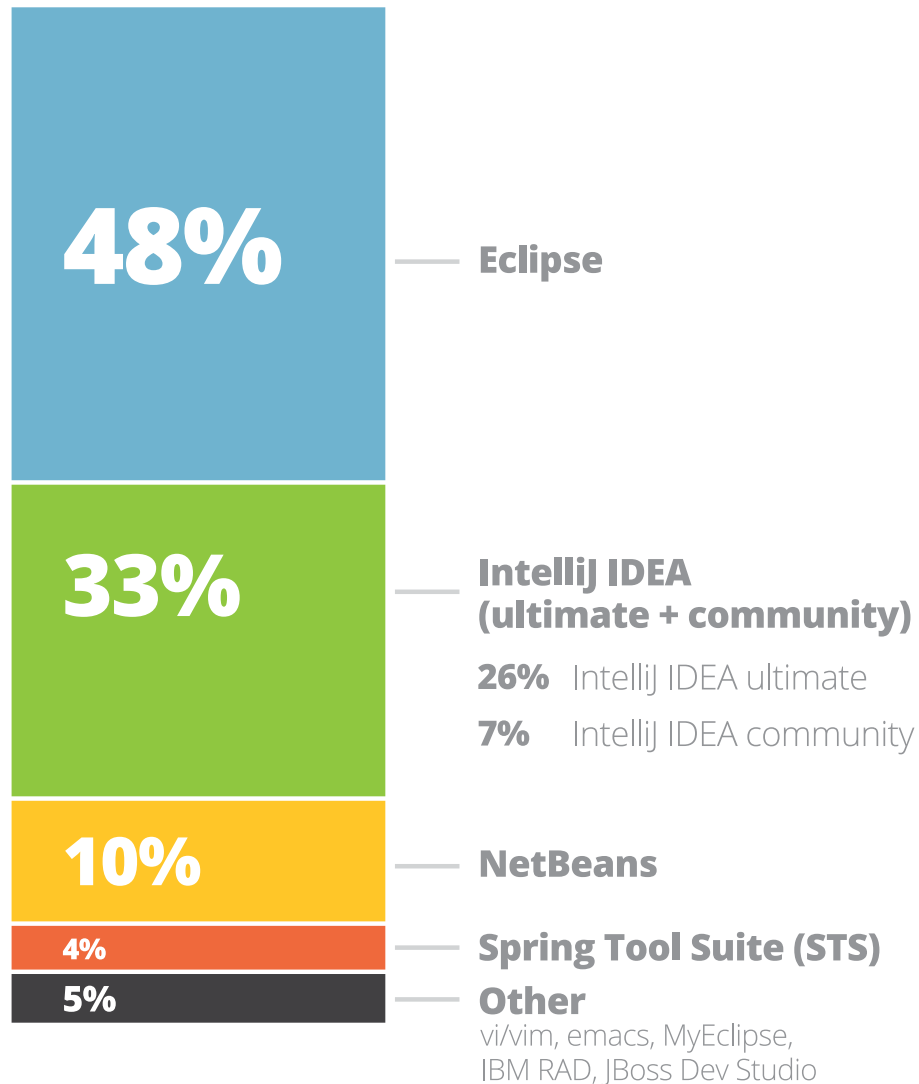
IDES

"

As a developer tools vendor, one of our major challenges today is overcoming a common misconception that IDEs are bloated and packed with ridiculous features that creative developers should only use when dealing with large projects full of legacy code. Contrary to that, we believe that smart tools enable creative work with any project, no matter how big or small it is, be it a fresh startup or something that's been in development for a long time.

"

MAX SHAFIROV,
CEO of JetBrains

IDE used most often

48% — Eclipse

33% — IntelliJ IDEA (ultimate + community)

26% IntelliJ IDEA ultimate

7% IntelliJ IDEA community

10% — NetBeans

4% — Spring Tool Suite (STS)

5% — Other
vi/vim, emacs, MyEclipse, IBM RAD, JBoss Dev Studio

In last year's Developer Productivity Report 2013, 97% of respondents reported using IDEs (and 3% don't...really, WAT?), which makes the IDE market quite representative of development environments. This is a very mature market, with a strong commercial presence following the leading open source option, something that we could consider a reasonable indicator of market maturity.

So we aren't surprised at **Eclipse (48%)** retaining the dominant market position, which has been incredibly stable over the years - last time we asked, over two-thirds of developers used Eclipse on at least some projects; however, in 2012 the survey choices were non-exclusive, indicating that at least some developers use multiple IDEs. Eclipse's market dominance increases if you combine the other tools out there that are based on the Eclipse platform as a different distribution - e.g. Spring Tool Suite (STS), MyEclipse, IBM RAD and JBoss Developer Studio - then we can add another 7% to the score, coming in with over half the IDE market (55%) for Eclipse.

IntelliJ IDEA (33%) continues to make gains, with their Ultimate (26%) and Community editions (7%) representing the flagship JetBrains product. After years of anecdotal evidence, we also took the opportunity to ask "which IDE would you rather use or test for development?", listing all the same IDEs as in the previous question.

Yep, some out there still use vi/vim...

REBELLABS

Which IDE would you rather use or test for development?

49%	51%

IntelliJ IDEA
(ultimate + community)

Any Other IDE

REBEL**LABS**

Interestingly, between Ultimate and Community editions, IntelliJ IDEA is preferred by nearly half of respondents (49%), compared to "any other IDE". Commercial tool makers, you can take this as good news if you operate in a space dominated by open-source alternatives.

NetBeans (10%) maintains a stable 3rd-place as in previous years, despite the fact that releases since version 7.2 have received generally positive feedback and the recent NetBeans 8 shipped as Java 8-ready. Whether a small, stable market share for NetBeans is a strategic goal of Oracle or not, we hope that any business' preference for commercial products over open source wouldn't influence the level of support/promotion for a product in question.

Finally, for the first time, we saw **Spring Tool Suite (STS)** appear on the radar with 4%, displacing MyEclipse and IBM RAD in 4th place in the 2012 report.

For further reading, we recommend How to Get Started with IntelliJ IDEA as an Eclipse User and Using Eclipse for Java development.
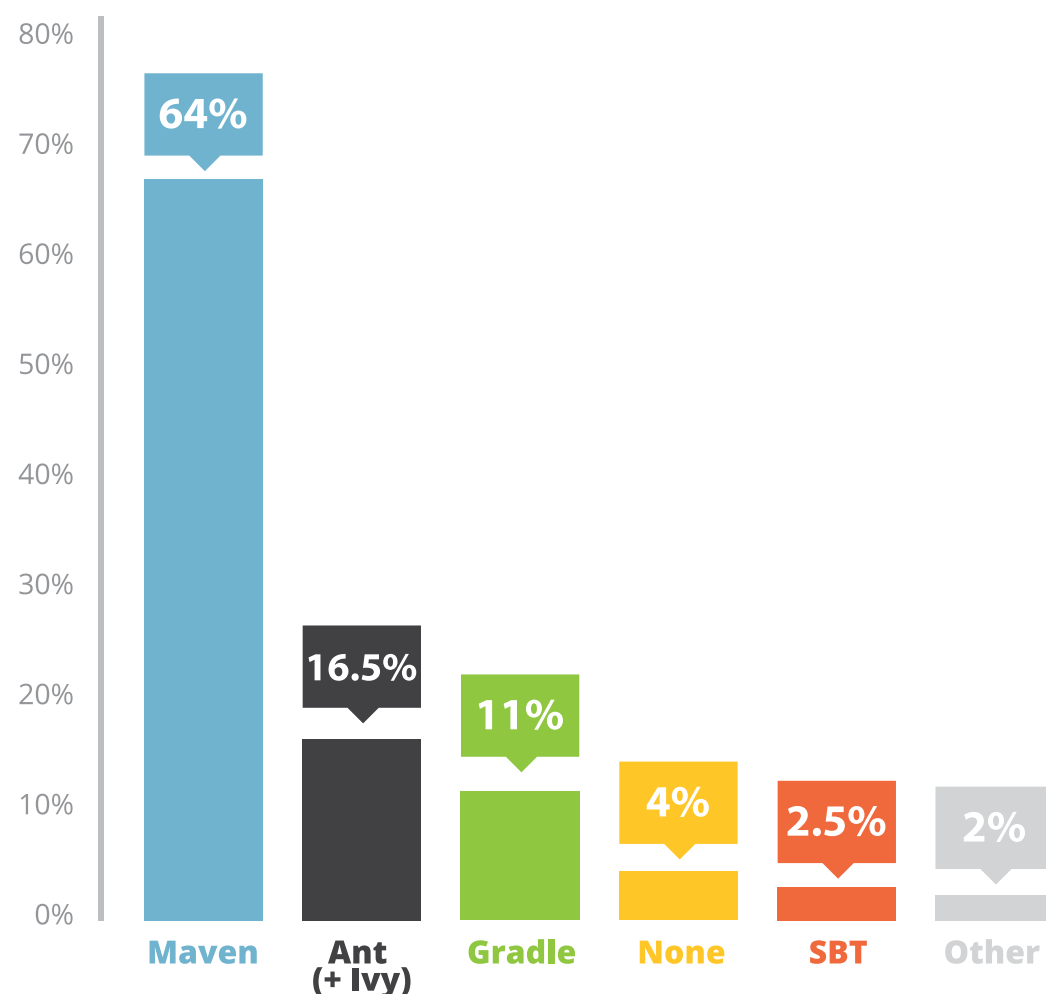
BUILD TOOLS

" *Our mission is to be the next generation build system for all major software platforms. This survey by RebelLabs is important evidence that points to how well Gradle is on its way. Gradle's momentum is enormous across Java, Android and C/C++. The overall number of Gradle downloads has surpassed 3 million, with more than 330,000 downloads just in April 2014. Exciting times for all of us.* "

HANS DOCKTER,
Founder of Gradle & CEO Gradleware

Build tool used most often

80%

70%

60%

50%

40%

30%

20%

10%

0%

64%

16.5%

11%

4%

2.5%

2%

Maven   **Ant (+ Ivy)**   **Gradle**   **None**   **SBT**   **Other**

REBELLABS

The build tools segment of the industry is quite mature, having been more or less stable for the last few years. In the past, **Maven (64%)** and **Ant + Ivy (16.5%)** have been more or less neck and neck. Indeed, we believe that many developers often use both in different ways on complex projects - in 2012, 67% of respondents used Maven and 48% used Ant.

This year we asked about the build tool used most often, and forced a single selection. Considering Maven's ubiquitousness and activity in Maven Central, for example, and these numbers aren't too surprising. What is interesting is the relatively rapid disappearance of Ant, which is a powerful tool until Ivy slowly creeps into the situation for dependency management and makes everything seem a lot more leisurely. Regardless, Ant with or without Ivy is only one-third as commonplace as it was reported in 2012.

The rapid increase of **Gradle (11%)** is also interesting to witness. Since 2012, Gradle use has more than doubled, which is probably due to its being chosen by Google as the future official build tool of the Android OS (which two-thirds of you out there are using, according to this report).

Gradleware, the enterprise solutions company led by the founder of Gradle, has helped along a uniquely active community for this market - indeed, when we asked: "Which build tool would you like to learn more about", **58% of developers selected Gradle**, which appears to be the most interesting single tool we asked about.

Incidentally, **SBT (2.5%)**, was selected by 1 in 10 respondents as the tool to learn more about. This reflects the popularity and interest in Scala we saw previously, even though we are reminded by Typesafe that the tool can be used for Scala and Java projects alike.

For further reading, check out Java Build Tools Part 1 as well as Part 2, which go deeper into using and benefitting from Maven, Ant & Ivy and Gradle

Which build tool would you like to learn more about?

58%
Gradle

42%
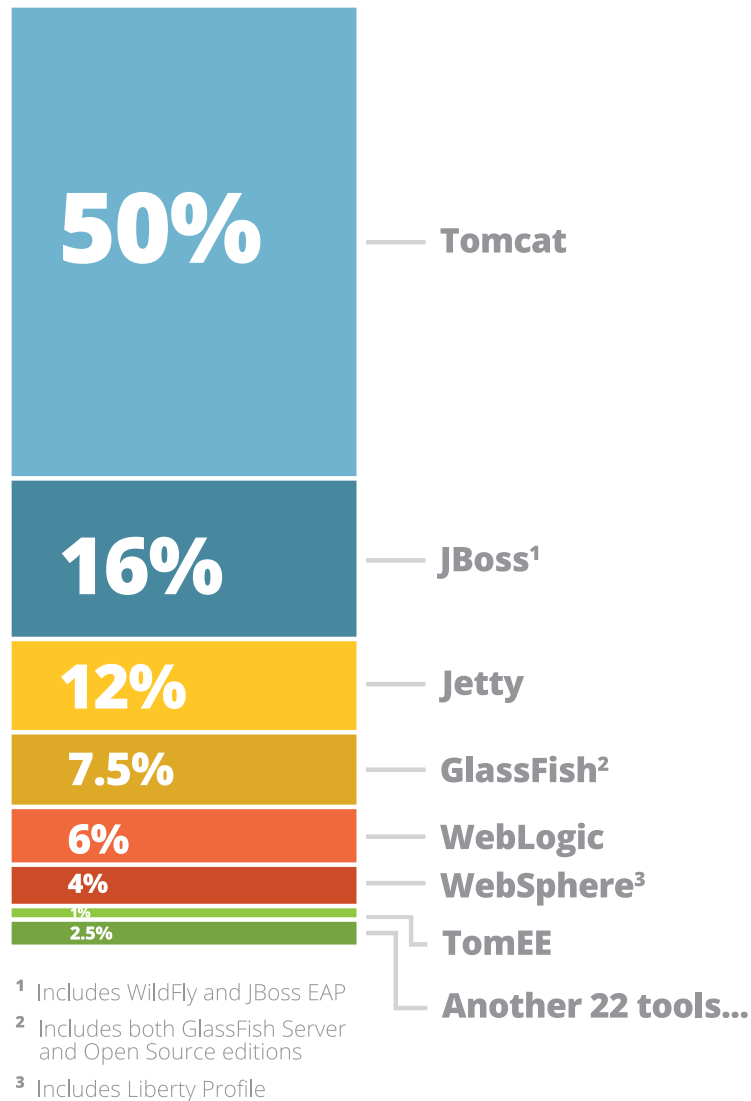**Any Other Build Tool**

REBELLABS

APPLICATION SERVERS

> "Its always pleasing to know that JBoss is the top commercially supported Java EE application server for such a large sample. This number is all the more relevant given that 81% use the same server in production. JBoss and WildFly have always been appealing to developers and we are always looking for feedback on how to improve."

ARUN GUPTA,
Director of Dev Advocacy at Red Hat

App Server most often used*

Percentage	Server
50%	Tomcat
16%	JBoss[1]
12%	Jetty
7.5%	GlassFish[2]
6%	WebLogic
4%	WebSphere[3]
1%	TomEE
2.5%	Another 22 tools...

[1] Includes WildFly and JBoss EAP

[2] Includes both GlassFish Server and Open Source editions

[3] Includes Liberty Profile

The market for Java application servers is quite mature, with a plethora of offerings from both open source and commercial providers alike. Our results showed that 8% of respondents are not using an app server at all (the scores have been normalized). Additionally, there are loads of tools available to choose from - the bottom 2.5% of users made up over 20 different technologies!

Tomcat (50%) remains the dominant application server (also open source), consistent with the non-exclusive results from the 2012 survey, which showed Tomcat in use by 59% of respondents at that time. Otherwise, the results pretty much fall in line with what we saw in 2012: including **JBoss (16%)** in 2nd place and **Jetty (12%)** in 3rd place. **GlassFish (7.5%)**, which was behind **Oracle WebLogic (6%)** in 2012, has since risen above just to have commercial support cut from it in late 2013. **IBM WebSphere (4%)** along with Liberty Profile, still contain about 4% of users we surveyed.

One newcomer has made a small but meaningful entrance into the market: **TomEE (1%)** supported by Tomitribe is a Java EE 7 web profile certified app server based on Tomcat, which might be a good way for enterprise Tomcat users to go.

REBEL LABS

* The results were normalized to exclude non-users

Same App Server
in Production & Development?

81% ▶ Yes

19% ▶ No
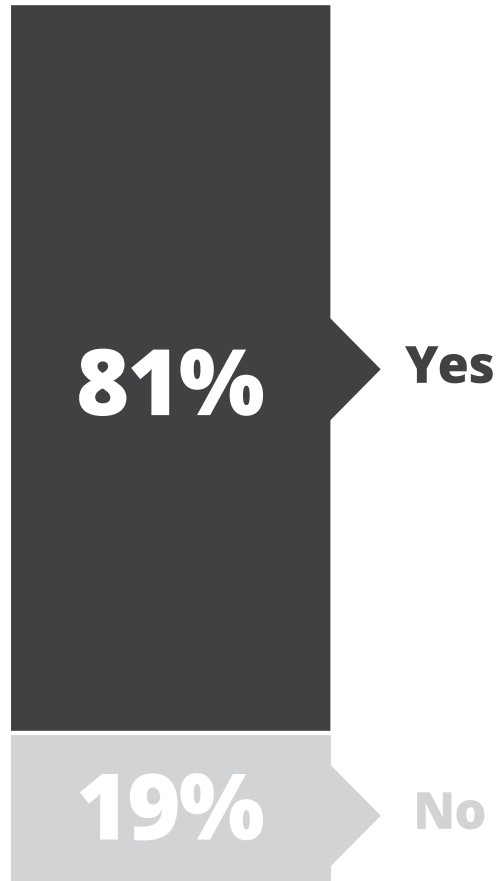
We also asked whether developers are using the same server in development as they are using in production - in most cases, the app server used in production is determined first, then teams can often decide to use the same in development or not. We see that 81% of respondents use the same application in development as in production, while 19% do not.

We ran some pivot analysis to know whether the relative market share changes based on if the **same application server used in production is selected for use in development**.

We reasoned that:

1. If market share rises for any given application server when used in development only, then the app server is popular with developers, since a choice of app server is implied.

2. However, if the market share decreases for any given application server when used in development only, this means that these developers would otherwise not choose the same app server for development if it were up to them (most likely not Java EE developers).

There were 3 application servers whose market share increased when used in development only. Again, we're trying to detect developer

preferences in this - i.e. when given the choice to freely decide which application server to use, which do they pick?

Our analysis shows that Jetty, TomEE and Tomcat are popular with developers who don't need to think about production. Below, the **% in bold** shows the overall market usage of each server, followed by the % of use in both production and development as well as in development only.

App Servers that
developers choose
outside of production

1. **Jetty (12%):** Falls to 9% when used in both environments, increases massively to 28% when used in development only. Jetty is very popular with developers.

2. **TomEE (1%):** Stays the same at 1% when used in both environments, but increases significantly to 3% when used for development only. TomEE is popular with developers.

3. **Tomcat (50%):** Drops to 48% market share when used in both environments, increases to 51% market share when used in development only. Tomcat is popular with developers.

Our analysis also showed that, when given the choice, respondents will not as often choose an application server that is not required for use in production as well. JBoss and GlassFish both fall into this category despite a not-incorrect perception of popularity among developers, but the numbers are what they are. The truly unpopular choices are, not surprisingly, the pricey and designed-for-production app servers WebSphere and WebLogic (the latter has just one single solitary developer using it when it's not used in production).

Some app servers are more popular with developers, some more for businesses

App servers used more often **in production than development**

1.   **Oracle WebLogic (6%):** Increases to 7% when used in both environments, but drops to about 0.005% share when used in development only. Literally, 1 respondent reported WebLogic in development when they don't need it in production. Extremely unpopular with developers.

2.   **IBM WebSphere (4%):** Increases to 5% when used in both production and development environments, drops to 2% when used in development only - and this lack of popularity is most likely saved only by IBM's lightweight Liberty Profile shining through, rather than the full-blown, multiple-gigabytes WebSphere package.

3.   **JBoss (16%):** Increases to 17% market share when used in both environments, falls to 7% when used in development only. JBoss is a popular choice of production server, not as popular with developers.

4.   **GlassFish (7.5%):** No change in GlassFish usage when used in both environments, falls to 5% when used in development only. It seems that GlassFish is the most popular of the unpopular app servers to use in development only.

For additional reading, we recommend
The Great Application Server Debate with Tomcat, JBoss, GlassFish, Jetty and IBM Liberty Profile.
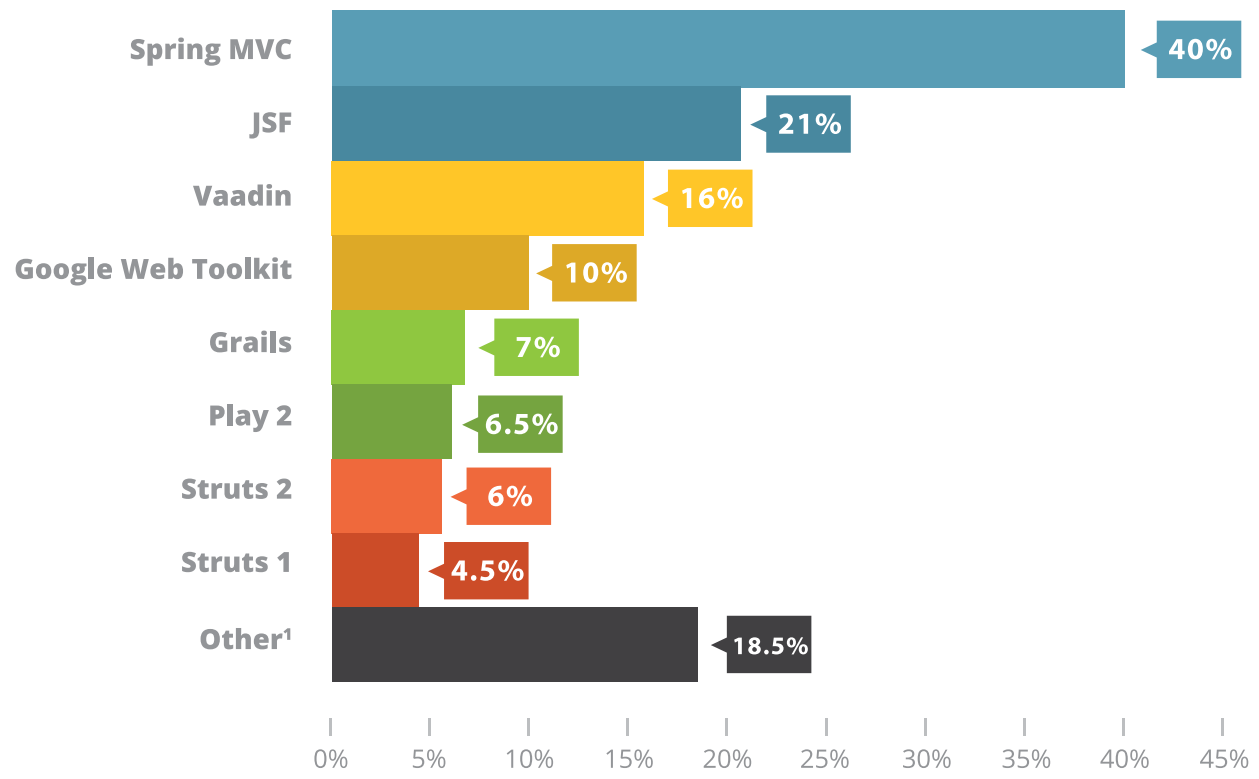
WEB FRAMEWORKS

> " *It's not surprising that Spring MVC is the most popular web framework among Java developers. Not only was it better than Struts when it was first released in 2004, but it has continued to innovate over the years, keeping up with trends like Ajax, REST and API development. It is surprising to see that "none" is a popular choice, but I'm guessing that's because people are developing APIs (for JavaScript and mobile clients) instead of server-side HTML-based UIs.* "

MATT RAIBLE,
Founder of Raible Designs

Web frameworks in use *

Framework	Percentage
Spring MVC	40%
JSF	21%
Vaadin	16%
Google Web Toolkit	10%
Grails	7%
Play 2	6.5%
Struts 2	6%
Struts 1	4.5%
Other[1]	18.5%

This is a fragmented, very mature market where **Spring MVC (40%)** is used by many developers out there, and still enjoys a sizable lead over **JSF (21%)** (including all the *Faces) in 2nd place. Notable changes since 2012 are the increase in usage of **Vaadin (16%)**, whereas **Struts** (both 1 & 2 together) has fallen from 17% in 2012 to just over 10% usage nowadays. **GWT (10%)** retains some market share, but less than before, we suspect, as security issues around JavaScript, whether valid or not, has led to announcements by Google. And since we never really asked, we needed to normalize these results because one in six developers (around 17%) doesn't use any real framework at all, just JSPs and servlets.

REBELLABS

* *Multiple selections were possible and the results were normalized to exclude non-users*
[1] *Including Wicket, Seam, Tapestry, Play 1, ZK framework, VRaptor and about 40 others*

> " While thin server UI frameworks continue to generate a lot of buzz and interest, the fact is that the Java world still has significant existing and new investment in server-side UI technologies. I'm happy to see that JSF, now in its tenth year of continuous improvement, still connects with its users to deliver value.

> – ED BURNS,
> *JSF Spec Lead at Oracle*

These frameworks like to mix it up and party together...

In a modern development environment, using multiple web frameworks is not only expected, but even recommended for some. Combining the strengths of multiple frameworks, like Spring and JSF or GWT can be beneficial - indeed, when we checked it out, 25% of Spring MVC users are also using JSF!

Drilling deeper into the data revealed more parallel use of several frameworks at once, which is something we can get into later. Developers often use multiple frameworks in conjunction, combining the benefits of Spring, JSF, Vaadin, Wicket, etc, so we aren't surprised at these results.

For further reading, we would like to share The Curious Coder's Java Web Frameworks Comparison as well as The 2014 Decision Maker's Guide to Java Web Frameworks (Parts 1 and 2 on the topic).

OBJECT-RELATIONAL MAPPING (ORM) FRAMEWORKS

> " *Looking at these stats, we can see that RDBMS are still the top data storage technology (85%), and ORMs (53%) are the slightly leading means of database access before SQL (32%). At Data Geekery, we believe that those who prefer SQL over ORMs deserve better, typesafe, embedded SQL – which is why jOOQ is gaining momentum in that segment.* "
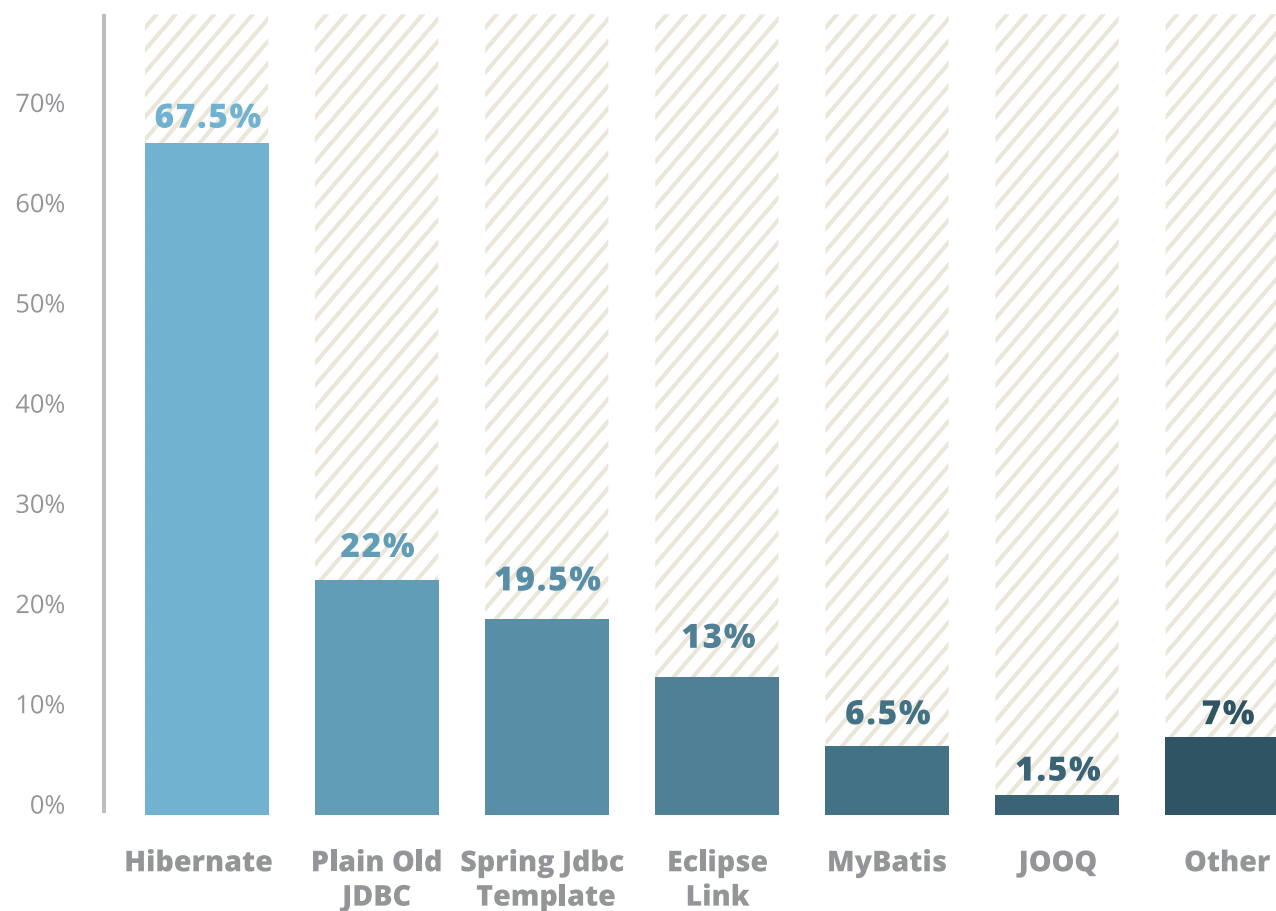
LUKAS EDER
CEO of Data Geekery GmbH

ORM framework(s) in use*

Framework	Percentage
Hibernate	67.5%
Plain Old JDBC	22%
Spring Jdbc Template	19.5%
Eclipse Link	13%
MyBatis	6.5%
JOOQ	1.5%
Other	7%

This is the first year we specifically asked about ORM frameworks for helping you with your data management. In the past, we'd lumped the dominant **Hibernate (67.5%)** into a more generic "Application Frameworks" segment, where in 2012 it was reported being used by 54% of respondents. Approximately 10% of respondents do not use this technology, so the results have been normalized.

In this semi-mature, changing market, we find plain old **JDBC (22%)** and **Spring JdbcTemplate (19.5%)**, two popular flavors of more or less the same tech, taking over 40% of the market when combined. We were a bit surprised at the high level of use of **EclipseLink (13%)** in light of these more popular ORM frameworks, but then again it is integrated with JPA, commonly used by developers. **MyBatis (6.5%)** and **jOOQ (1.5%)** also appear on our radar as well, serving some niche audiences.

REBELLABS   * Multiple selections were possible and the results were normalized to exclude non-users
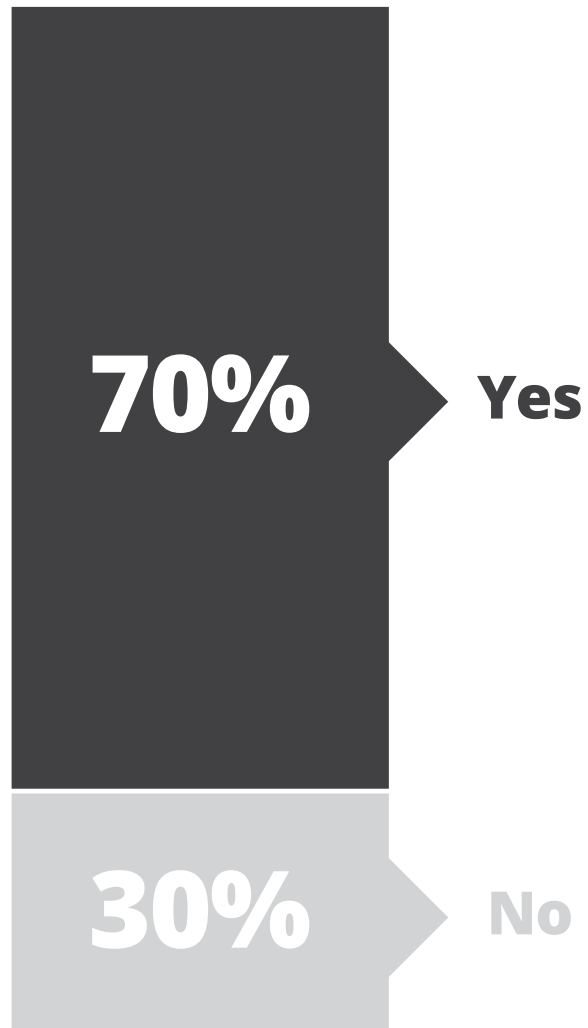
CODE ANALYSIS TOOLS

" *When initially implementing systems like FindBugs or Checkstyle, it can be a massive hump to get over. The long term cost, however, approaches zero (for me) because the tools actually serve as a bit of a training tool. In time I find myself writing code according to the style policies subconsciously. In addition, it's also a nice, reasonably automated way to onboard new/junior developers since the tools will guide those new to a project in the "correct" way to write code freeing up any mentoring to focus on actual architectural discussions.* "

JUSTIN LEE,
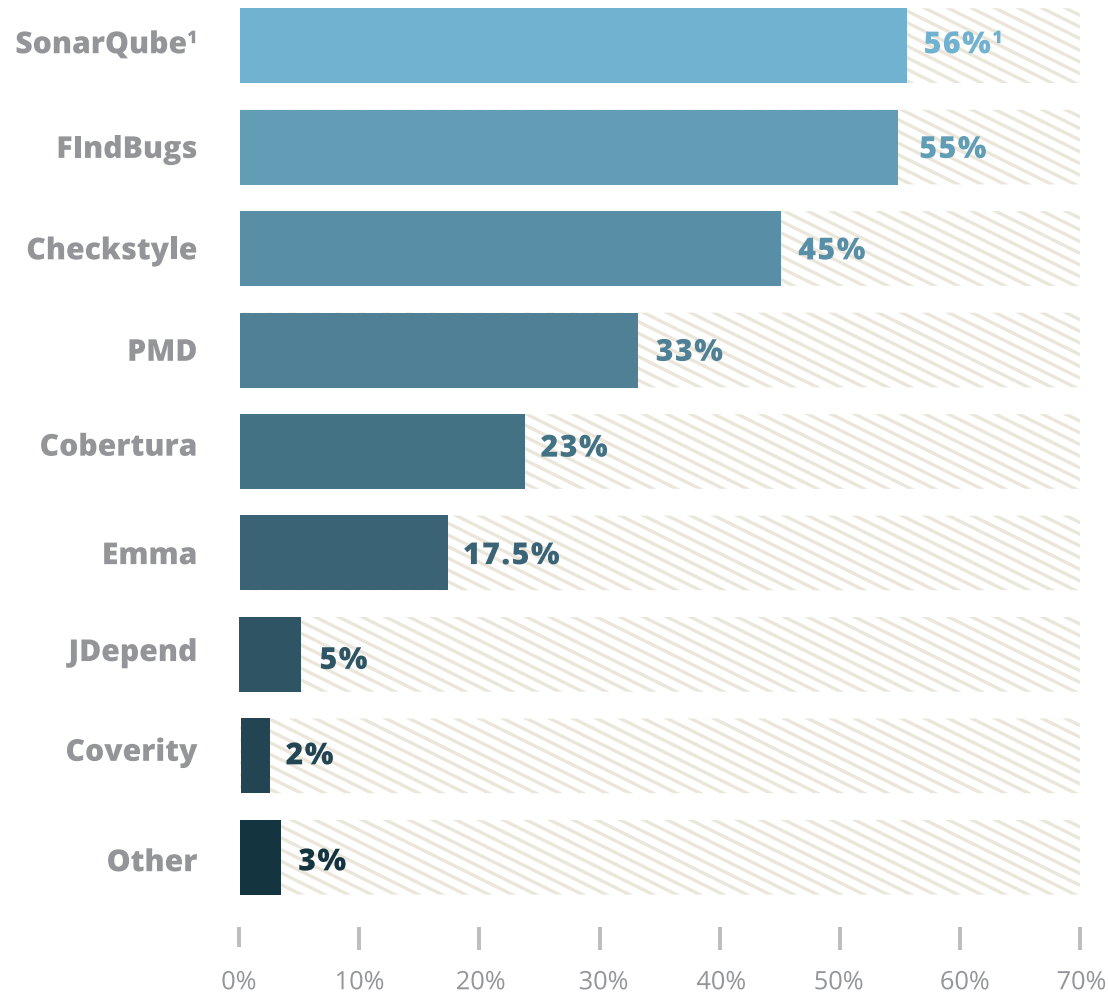Member of Technical Staff at MongoDB

Do you use one or more
code analysis tools?

In 2013, we saw that using code analysis tools has a significant effect on the quality of software (i.e. less bugs) and developers' ability to predictably deliver a final version. At the time, only 49% of developers we asked "Do you monitor and fix code quality problems?" replied affirmatively, so in this 2014 survey we specifically asked about static code analysis tools and provided a list of options. This proved that more teams are in fact using code analysis tools (70%), and the results are normalized to reflect that about 1 in 3 developers do not use anything.

We know you aren't using these tools...but you should!

70% Yes

30% No

REBELLABS

Code analysis tools in use *

SonarQube[1]   56%[1]
FIndBugs   55%
Checkstyle   45%
PMD   33%
Cobertura   23%
Emma   17.5%
JDepend   5%
Coverity   2%
Other   3%

0% 10% 20% 30% 40% 50% 60% 70%

* Multiple selections were possible and the results were normalized to exclude non-users
[1] SonarQube is an integration platform for hosting analysis tools

This market could be described as emerging, and there is still some maturity to be gained. **SonarQube (56%)**, aka Sonar or SonarSource, is a intelligent platform for integrating other code analysis tools, like **FindBugs (55%)**, a University of Maryland project and the most popular single technology. But developers also enrich their experience by using **Checkstyle (45%), PMD (33%), Cobertura (23%)** and **Emma (17.5%)**.

When we looked into it, more than 50% of all respondents reported using at 2 or more of these tools, and hundreds reported using as many as five different static code analysis technologies. At the very bottom, we see **Coverity (2%)**, the only commercial code analysis tool we see with any real numbers, albeit small digits to be sure. As this industry matures, we expect to see more formalized offerings with support made available for organizations that care about how many bugs make it into their releases.

For additional reading, check out this recently-published report: The Wise Developer's Guide to Static Code Analysis.

REBELLABS
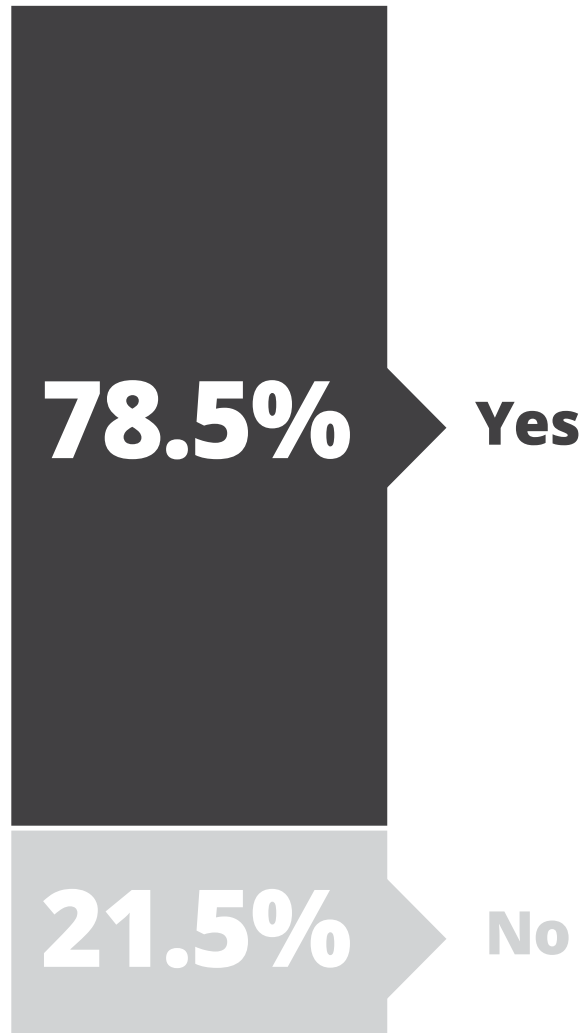
CONTINUOUS INTEGRATION
SERVERS

" *Automation is the foundation of productivity gains in software development nowadays, and the diversity in the ways in which we develop software calls for an open-source platform that can fit any environment. The survey result shows that Jenkins continues to lead this space with its ecosystem, extensibility, and diversity.* "

KOHSUKE KAWAGUCHI,
Jenkins project founder,
CloudBees CTO

Do you use
Continuous Integration?

78.5% Yes

21.5% No

In 2013, only 68% of developers reported using Continuous Integration technology, so it's good to see that a more representative number from this year's findings has raised it nearly to 80%. Only about 1 in 5 developers do not use CI technologies whatsoever.

However, more than half of developers (among all respondents) not only use CI, but use Jenkins specifically. When normalized, **Jenkins (70%)** settles in a comfortably dominant position, possibly amused that it was once considered a rival of **Hudson (8%)**. This market is still maturing, with the majority of developers either using Jenkins or nothing at all. At least the discussion over Jenkins vs Hudson can be put to rest!

Yeah, Jenkins won. Get over it.

REBELLABS

Other signs of progression in this area are present, with commercial tools **Bamboo (9%)** and **TeamCity (7.5%)** each grabbing some of the CI market, likely enjoying some small usage based on the popularity of Atlassian's and JetBrains' tools for developers and software organizations. We saw **Travis CI (2%)** appear in the past, which offers both open source and professional CI options, and **CruiseControl (1%)**, which had 4% of the market in 2012, seems to be sinking peacefully into oblivion.

For additional resources, please review *Release Management for Enteprises* **plus** *Why Devs <3 CI: A Guide to Loving Continuous Integration* **as well as** *Jenkins CI: The Origins of Butlers, Build Masters and Bowties*.

Continuous Integration (CI) server used*

70% — Jenkins
9% — Bamboo
8% — Hudson
7.5% — TeamCity
2% — Travis CI
1% — CruiseControl
2.5% — Other
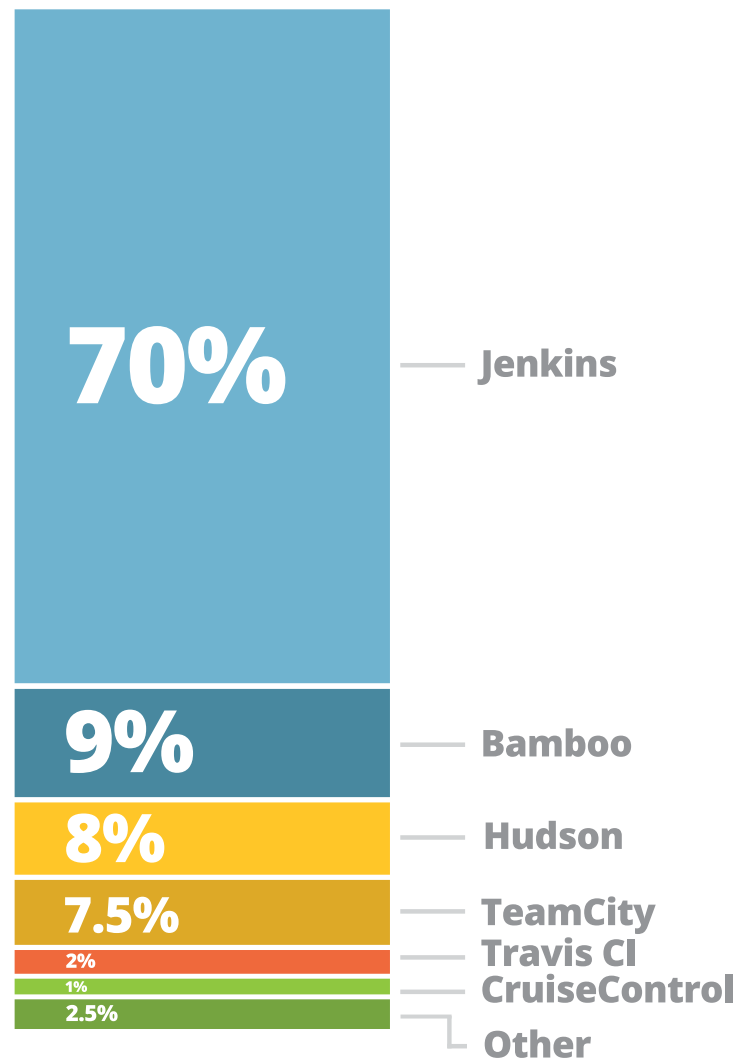
DATABASES: SQL & NOSQL

" *Some years ago, no one would have thought that the Oracle / MySQL / SQL Server leadership could be challenged by a newcomer, but we at Data Geekery think that PostgreSQL is the most promising database of this decade, elegantly combining relational and many non-relational / NoSQL concepts.* "

LUKAS EDER,
CEO of Data Geekery GmbH

SQL, NoSQL or both?

39%	**53%**	4%	4%
Use Both	**Use only SQL**	**Use only NoSQL**	**Use Nothing**

In the world of databases, opinions can run hot. The SQL vs NoSQL vs Something Even Better debate continues to rage at some level, and when we looked at the results we found that a large minority of the 2164 respondents **use both (39%)**, seeing advantages of using SQL and NoSQL technologies in conjunction:

as our own engineers will tell you, the high-performance SQL is great at querying websites and other real-time services, whereas NoSQL is better at handling frequently-changing data structures, namely in regards to reporting and configuration.

While 4 out of 10 engineers are using both technologies in peace, the rest are divided into camps: **SQL only (53%), NoSQL only (4%) and None (4%)**.

Primary SQL technology*

19% PostgreSQL

30% Oracle DB

32% MySQL

8% MS SQL Server

4% DB2

2% H2

5% Other[1]

We normalized the numbers to exclude non-users (8%) in total, and what we see is a pretty mature market layout with several strong contenders of both open-source and commercial flavors.

MySQL (32%), the original relational database management system (RDBMS) folks, were bought by Sun Microsystems in 2008 and probably became one of many compelling factors for the acquisition of Sun by Oracle in 2010. So, if you combine this with the market share of **Oracle DB (30%)**, Oracle effectively maintains nearly two-thirds of the SQL market.

The third-place upcomer is **PostgreSQL (19%)**, an object-relational object-relationship database management system (ORDMBS), and is run by a consortium of volunteers as the PostgreSQL Global Development Group. The final one-fifth of the market is led by **MS SQL (8%), DB2 (4%), H2 (2%)** and a dozen more.

* *The results were normalized to exclude non-users*
[1] *Including Derby/Java DB, Sybase, SQLite and a dozen others*

Primary NoSQL technology*

> **"**
> We find that developers are using NoSQL databases like MongoDB more and more, for the advantages they provide over traditional relational databases: agility & scalability. Developers love that flexible schemas allow them to develop rapidly and adapt easily to changes in the data model; and the high availability and replication out of the box suits DevOps nicely.
>
> – TRISHA GEE,
> *Developer / Evangelist at MongoDB* **"**

9%
Redis

10%
Apache Cassandra

56%
MongoDB

5%
Hazelcast

3%
Neo4J

17%
Other[1]

REBELLABS

* *The results were normalized to exclude non-users*
[1] *Including Memcache, Riak and a dozen others*

In the NoSQL world - remember, that stands for "Not Only SQL", not "No SQL" - things are a bit more fragmented. This area is less mature than the SQL landscape, and among the developers using NoSQL, which is less than half of our respondents, **MongoDB (56%)** is clearly leading the pack. The rest of the bunch are relatively new and some are supported by powerful organizations like Apache and Pivotal.

Here we have **Apache Cassandra (10%)**, **Redis (9%)**, **Hazelcast (5%)** and **Neo4J (3%)**, and a couple dozen more minor projects out there make up the final **Other (17%)** section.
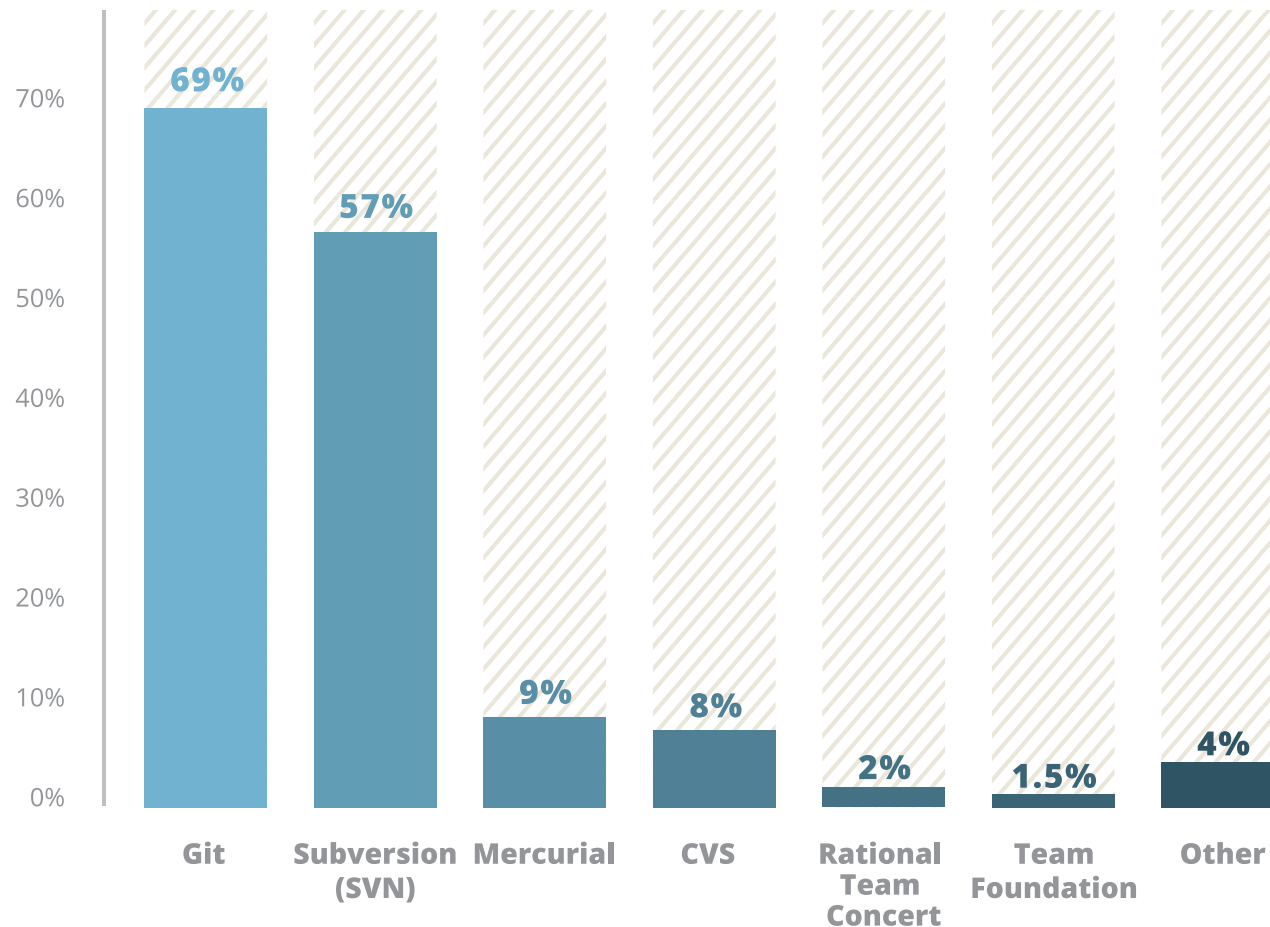
VERSION CONTROL SYSTEMS: (VCS)

> " The last twelve months have seen Git become the dominant version control system for leading edge teams and more conservative software development shops alike. This is due in part to industry publications mentioning Git collaboration platforms like GitHub, the maturing build tool support of Git, and an ever-increasing number of IDEs that use Git as their default VCS. "

MATTHEW MCCULLOUGH,
Trainer at GitHub

VCS technologies used*

- Git: 69%
- Subversion (SVN): 57%
- Mercurial: 9%
- CVS: 8%
- Rational Team Concert: 2%
- Team Foundation: 1.5%
- Other: 4%

In our 2013 report, we found that using Version Control gave a 9% gain in predictability for releases - one of the best tool categories out there. For the first time, we see **Subversion (57%)** finally displaced by **Git (69%)**, although both systems are being used in conjunction: 60% of Subversion users also use Git, and 49% of Git users also use Subversion.

So, often times these tools are complementary in developers' stacks, although some out there wonder why, with distributed VCS options like Git and **Mercurial (9%)** available, anyone uses Subversion, or **CVS (8%)**, which hasn't released anything since 2008. It would be interesting to see the ratio between new vs. legacy projects for each VCS. Bottom line: Git, with the assistance of the popular enterprise provider GitHub, has announced its dominance.
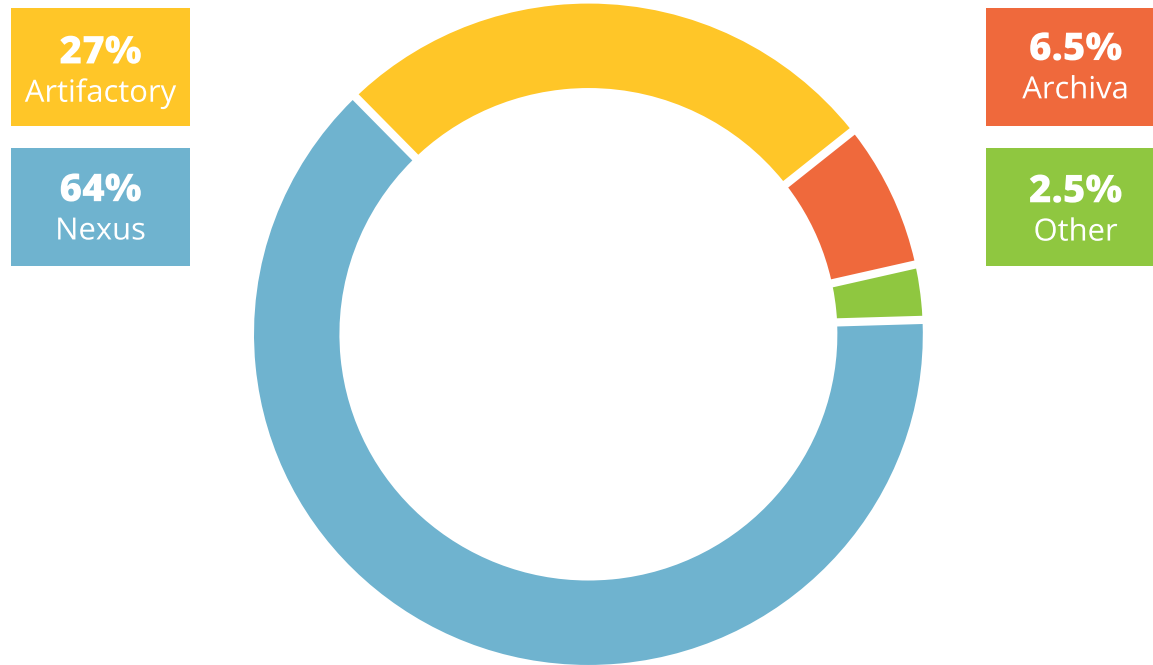
REBELLABS

Multiple selections were possible

BINARY AND ARTIFACT
REPOSITORIES

" *I'm continually amazed that many development teams and enterprises do not use an artifact repository.* "

COLLABNET BLOG

Binary/artifact repository used*

27%
Artifactory

64%
Nexus

6.5%
Archiva

2.5%
Other

In past reports, we didn't consider looking too deeply into binary and artifact repositories. Possibly the cause is that quite a large minority of developers - nearly 40% in this survey - simply don't use one.

In the past, the real enterprise choice has been **Nexus (64%)** from Sonatype, but recent developments in this tool category, led mainly by JFrog's **Artifactory (27%)**, have turned attention back to developer needs for versioning artifacts - from JARs and WARs to complete applications and libraries. Apache's **Archiva (6.5%)** represents a small group of users, but in this relatively immature marketplace there is a lot of room for changes and growth.

REBELLABS

The results were normalized to exclude a significant population of non-users
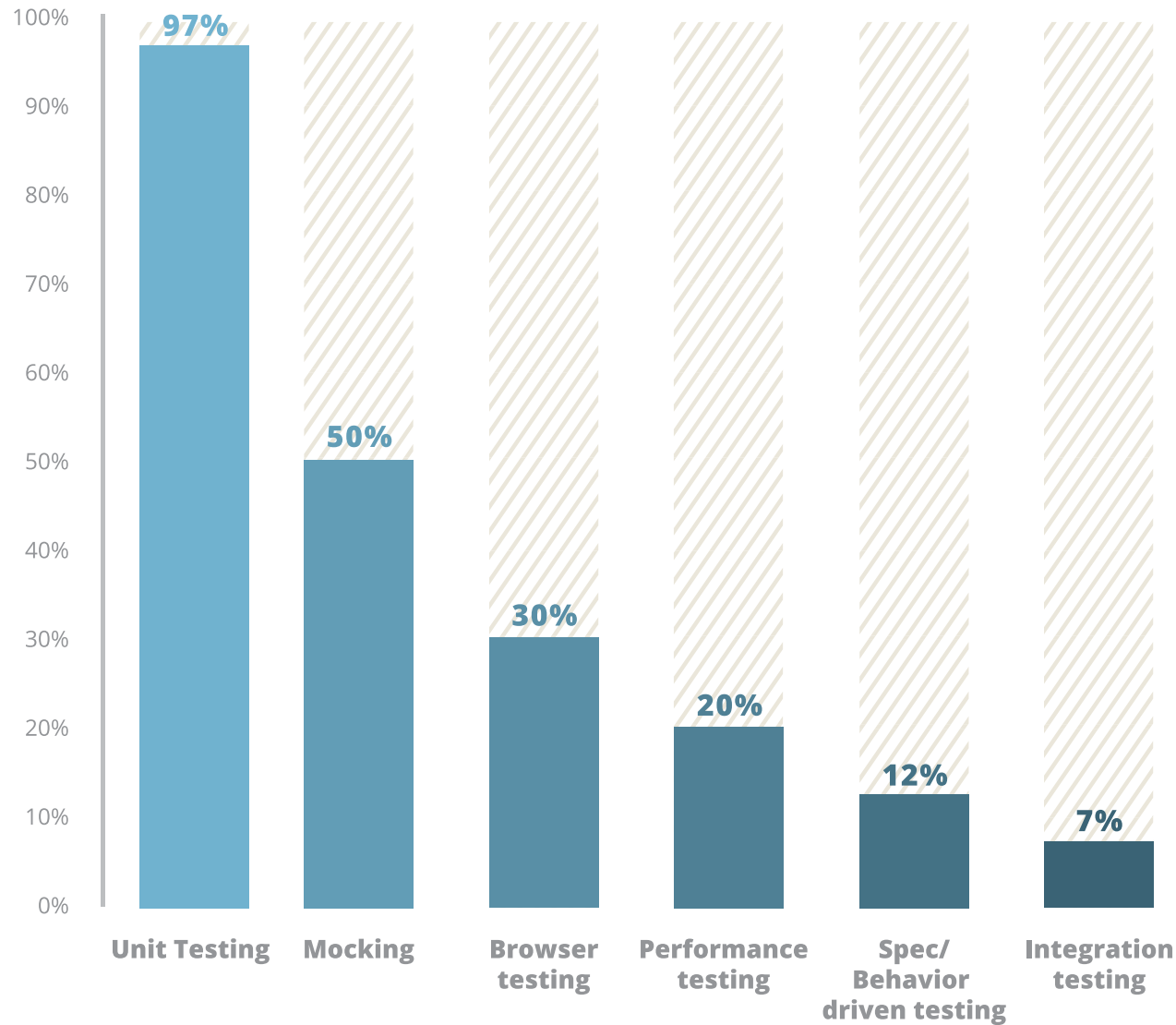
TESTING FRAMEWORKS

"" *It's great to see such high use of unit testing with mocking. UI testing and 20% performance and load testing is extremely positive to read. Now I want to see an increase in the quality of software I use daily!* ""

TOOMAS RÖMER,
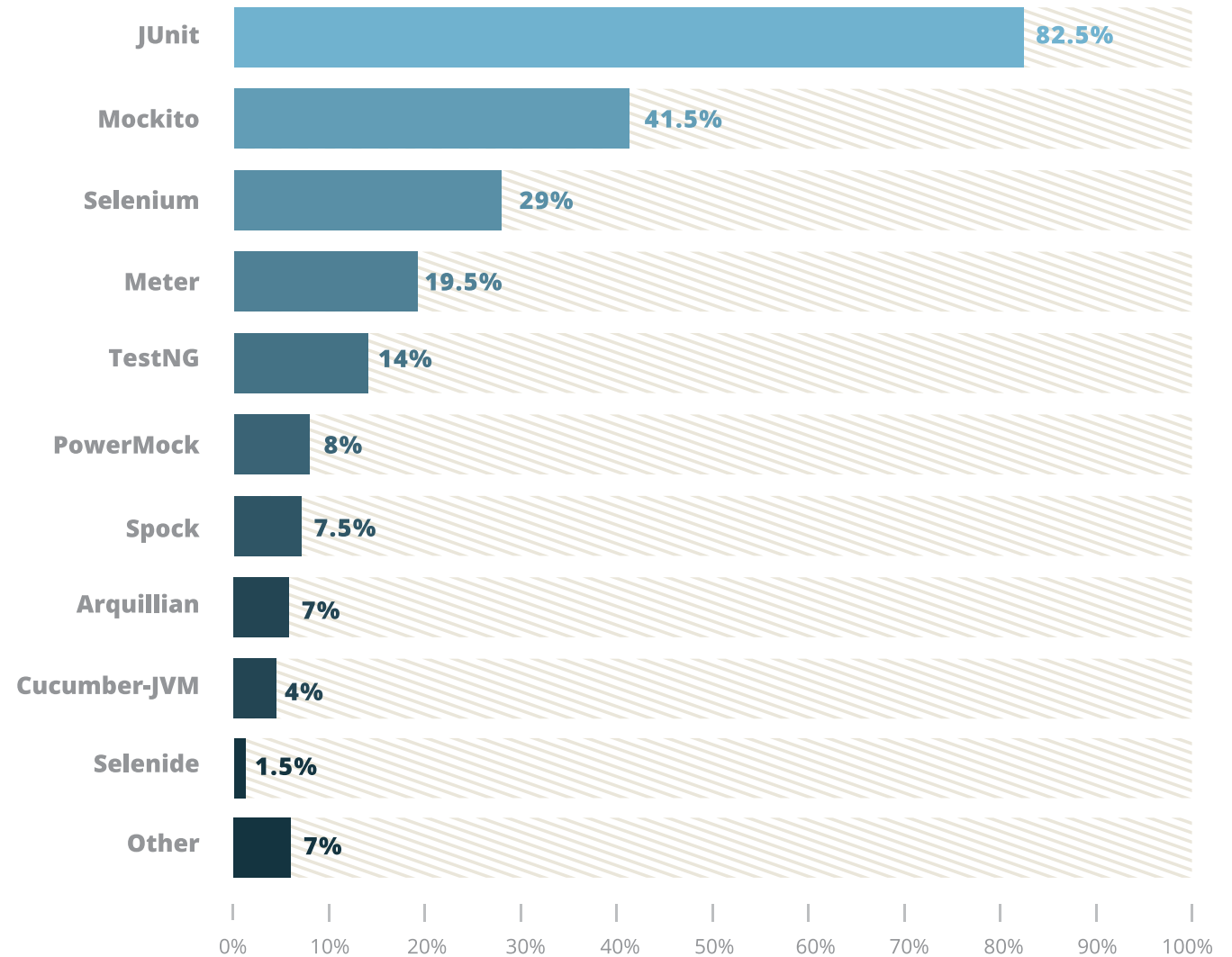Co-Founder of ZeroTurnaround

Types of tests being performed

As software producers (and users) it's relieving to see that the vast majority of developers - nearly 97% - do commit to unit testing their apps (i.e. with JUnit & TestNG), and half them apply some kind of mocking framework as well (i.e. Mockito / PowerMock). About 1 in every 3 developers also automate their browser testing (Selenium / Selenide) and 1/5th practise performance and load testing. Integration testing and spec-based or behavior driven testing are not incredibly common still.

Testing technologies in use*

Testing frameworks are in a maturing technology segment, where dominant technologies like JUnit, Mockito and Selenium are delivering solid-enough performance to raise eyebrows among a larger population, thus carving the way for small alternatives to build upon foundation laid out by others. The top four most used testing frameworks - **JUnit (82.5%)**, **Mockito (41.5%)**, **Selenium (29%)** and **JMeter (19.5%)** - are fully complementary to one another and cover entirely different areas of testing.

For more about unit testing and mocking, we invite you to check out Go Away Bugs! Keeping your code safe with JUnit, TestNG and Mockito.

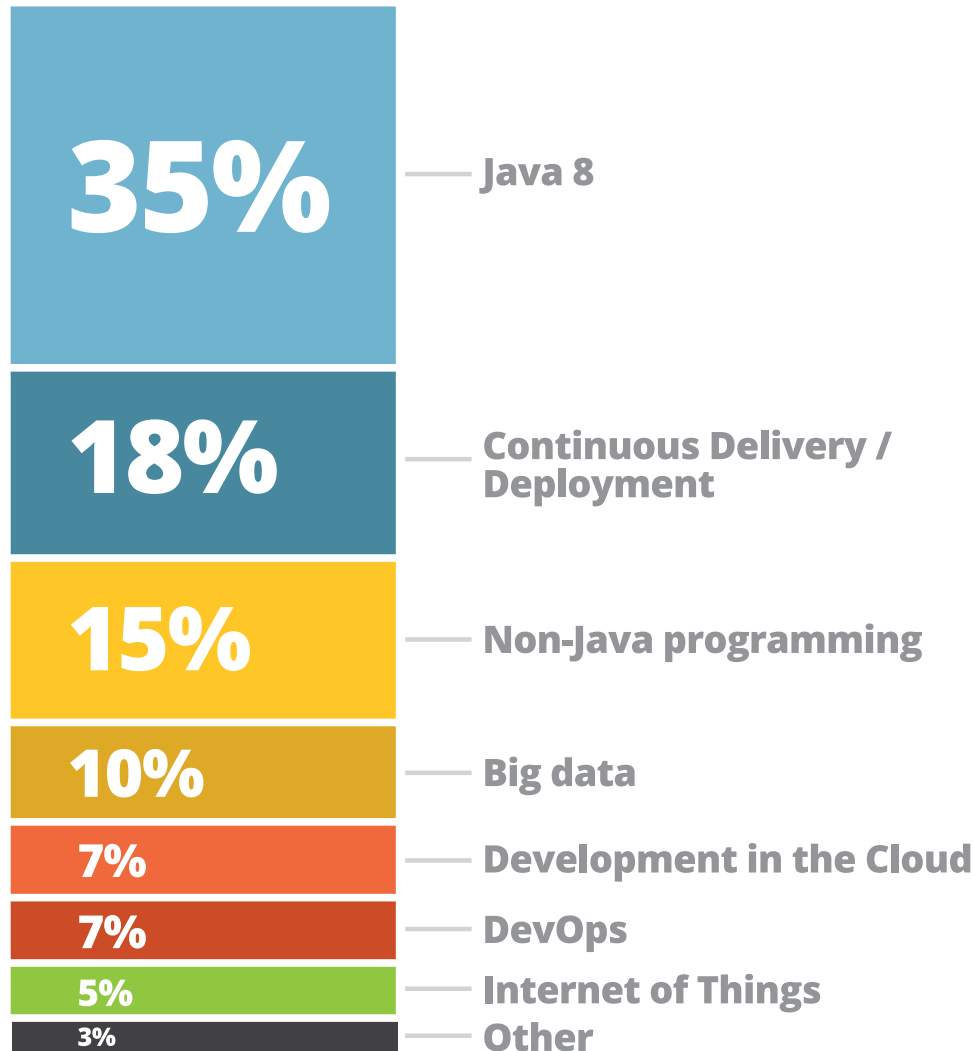| Technology	%
JUnit	82.5%
Mockito	41.5%
Selenium	29%
Meter	19.5%
TestNG	14%
PowerMock	8%
Spock	7.5%
Arquillian	7%
Cucumber-JVM	4%
Selenide	1.5%
Other	7%

REBEL LABS

LOOKING FORWARD
INTO 2015

" *I love seeing Continuous Delivery/Deployment taking hold in people's minds, as I believe that the way the industry currently releases applications needs help - harmonizing build server usage with release server usage would help.* "

DR. JEVGENI KABANOV,
CEO and Founder of ZeroTurnaround

Priorities for 2014-2015

35% —— Java 8

18% —— Continuous Delivery / Deployment

15% —— Non-Java programming

10% —— Big data

7% —— Development in the Cloud

7% —— DevOps

5% —— Internet of Things

3% —— Other

It's always good to look into other developer sentiments, and asking about what you (or your organization) believe to be high-priority is a good way to determine that. Not surprisingly, **Java 8 (35%)** is on the mind of over 1/3 of developers - with some long-awaited changes, like lambda functions, finally available, there seems to have been a breath of new life inserted into this "dead" language.

Interesting, **Continuous Delivery / Deployment (18%)** is receiving some renewed interest in light of not only some excellent successes for those organizations that embrace this, but for the concept's roots in automation and tooling to get the job done - something you'd likely approve of highly. Also, manual application deployments are becoming unpopular due to high risk and poor results.

From looking at earlier results, we see a reasonable interest in embracing **Non-Java programming (15%)**, as Scala and Groovy grow in enterprise adoption. **Big data (10%)** certainly is big, but still not necessarily grabbing too much attention, and **Development in the Cloud (7%)** is an interesting experiment whose outcome will require some waiting-and-seeing.

REBELLABS

But what happened with **DevOps (7%)**? It's scratching the bottom of the barrel compared to what we've all seen and heard in the last few years. Even with the patient love and energy of Patrick Debois and the burgeoning global DevOps community, along with pronouncements of support from companies like ThoughtWorks, PuppetLabs and ZeroTurnaround, DevOps seems to be struggling to find a crisp and clear mission statement - it's not always clear what is the key problem they are solving. Perhaps, like with some community movements, a major backer with enough force to initiate the movement and provide confidence for others to join along in the spirit of competitiveness, could make a difference. Netflix, we're looking at you ;-)

And as for the **Internet of Things (5%)**, it's too soon to say much, but we like that it's a great way to appreciate the way technology plays an increasingly larger role in the way we all live (just like in the movies!)

Looking at the next section, we conclude with a summary of our findings and a brief look at the maturity of different tool categories.

Searching for a DevOps champion

LAST WORDS

It's time to review the main takeaways from each section - anything that was too small to matter much right away can be revisited in a year, and anything that was too big to ignore is what we'd like to glance at.

The TL;DR version for efficient readers

- **Java versions** - Adoption since 2012 of the newest Java versions has been strong, namely growth in Java SE 7 and Java EE 6…the fact that certain older version of Java SE and EE still command decent minorities is a bit odd, but nothing to fear.

- **IDEs** - Eclipse is a very stable market leader, but increasing favor and interest in IntelliJ IDEA may eventually equalize things.

- **Alternative JVM languages** - Scala is the most interesting, enterprise-ready language on the JVM, and Groovy & Clojure play a great counterpart to the non-Java coding movement.

- **Build tools** - Maven's dominance is now unquestioned with Ant's decline from previous years, although fast-growing Gradle is extremely interesting to respondents.

- **Application servers** - Open-source players like Tomcat, JBoss, Jetty and GlassFish dominate both production and development environments, which use the same technology for 81% of respondents, although it's clear that development-only app servers in the highest favor are Jetty, Tomcat & TomEE.

- **Web frameworks** - A mature and fragmented market, over 1/3 of developers use more than 1 web framework, and here Spring MVC is still king in this area, with stable JSF and growing Vaadin following.

- **Object-relational mapping frameworks** - When it comes to ORM, Hibernate takes the cake, although other technologies are available and in common use in parallel.

- **Code analysis tools** - Although nearly 1/3 of developers don't use these tools (big mistake), the market is rich with complementary technologies like FindBugs, CheckStyle and a platform to bring it all together, SonarQube.

- **Continuous Integration (CI) servers** - Jenkins' dominance here is stronger than ever, yet 1 in 5 developers still don't use CI as a practice.

- **Databases: SQL and NoSQL** - The mature SQL is dominated by Oracle, but the market has a good mix of free/open-source and proprietary offerings from long-time players. NoSQL is a maturing segment that is mainly driven by MongoDB.

keep reading…..

- **Version Control Systems (VCS)** - Git, supported by the headline-worthy GitHub, is finally reigning supreme over Mercurial as a distributed VCS, and is often used in parallel with the legacy Subversion (SVN).

- **Repositories** - A maturing market that still needs a lot of adoption by developers, the legacy market leader Nexus is losing ground to JFrog's Artifactory.

- **Testing frameworks** - In this area, complementary technologies work better together rather than competing, with unit testing, mocking and browser testing all highly practiced using strong tools like JUnit, Mockito and Selenium respectively.

- **Priorities for 2015** - Java 8 (obviously), Continous Delivery / Deployment and Non-Java programming using alternative JVM languages are the 3 top priorities for the next year according to respondents.
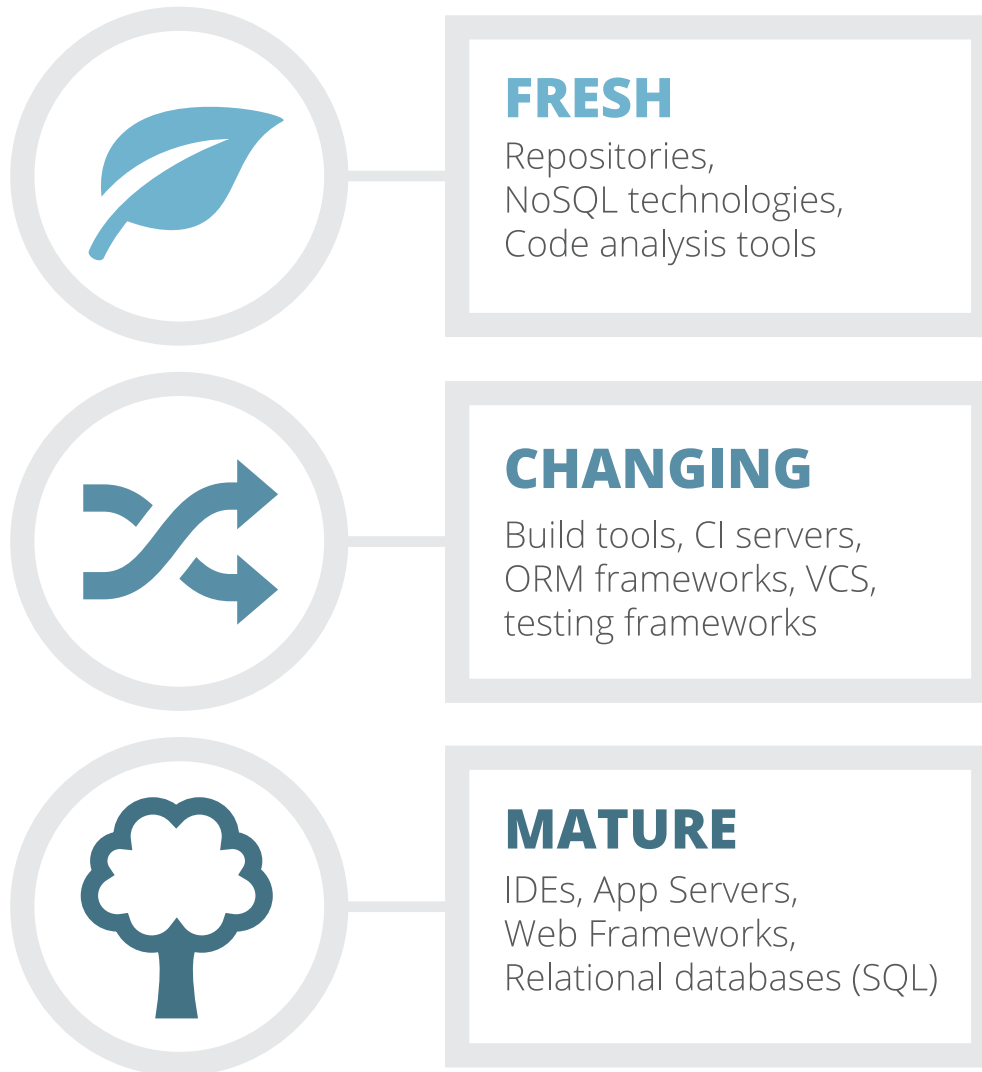
A note on the maturity of the technologies

To conclude, we have some short comments on what we could call "market maturity".

Maturity may not be terribly simple to define, but we believe that a couple reasonable indicators could be:

1. **Size of non-user base** - i.e. people who simply don't use to a particular type of tool for whatever reason, although frequent barriers are ignorance and lack of time, all of which affect market maturity.

2. **Level and type of market fragmentation** - i.e. looking at how many tools are represented overall in the category, what is representation of commercial vs. open source (or simply free) tools, existence of SMP (significant market power) players like Jenkins, Maven, Tomcat and Eclipse, and other factors influence market maturity.

Maturity of technology segments

FRESH
Repositories,
NoSQL technologies,
Code analysis tools

CHANGING
Build tools, CI servers,
ORM frameworks, VCS,
testing frameworks

MATURE
IDEs, App Servers,
Web Frameworks,
Relational databases (SQL)

REBELLABS

Based on these factors, we took a stab at defining three levels of market maturity, which goes like this:

Fresh technologies - Repositories, NoSQL technologies and code analysis tools all have large non-user groups, and it's this underuse that makes it difficult to tell which tools will emerge supreme at this point. More adoption in these spaces is needed.

Changing technologies - ORM frameworks, version control systems (VCS), continuous integration servers (CI), testing frameworks and build tools are changing based on evidence of things like the existence of a significant market power in the segment, a reasonably high level of adoption among the sample population, and signs of a changing landscape (i.e. the comparatively rapid growth of an emerging player).

Mature technologies - IDEs, App Servers, Web Frameworks, relational DBs (SQL) are mature, with comparatively little change year-over-year. With the exception of a developer "sea change" in interest and adoption towards IntelliJ IDEA, others like Tomcat, Spring and MySQL/OracleDB are all placed in what Gartner would call the "Plateau of Productivity" (although coders might have a different POV on that...)
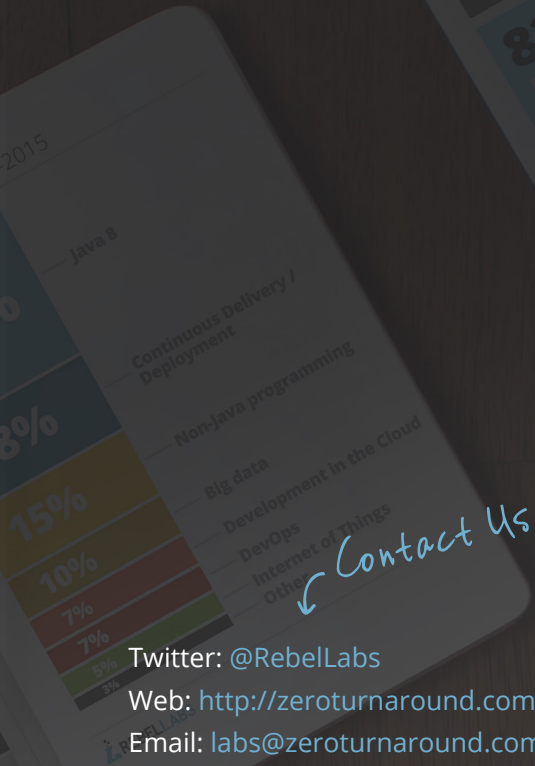
THANKS FOR READING!

*We would like to thank the subject matter experts (SMEs) quoted
in this report, the authors, editors & reviewers from ZeroTurnaround,
our sponsor that made it possible to create this report, and finally our readers
and the 2164 engineers+ out there that spent 5 minutes to help sick kids
in hospitals have a bit of imaginative gameplay. We hope our next survey
gets even more participants and goes to another worthy cause!
Stay tuned for more from RebelLabs, and tell us what you think on
Twitter @RebelLabs.*

Check out all our reports here: http://rebellabs.org