What happens when over 1000 Java developers compare their development environment?

Last year, ZeroTurnaround OÜ published a report on turnaround time, tools and application containers in the Java ecosystem. Over 1300 Java developers ended up sharing info about their development environment, and over 40,000 people found these results helpful.

This year we expanded on that, asking more and better questions to give you better and more accurate information. The survey itself provided data from over 1000+ Java EE developers and those responses were analyzed to create this report. We set out to discover:

- What is more popular, Ant or Maven?
- Which Java IDE is the most popular?
- What Java EE standards are used the most?
- Which Java Framework is most prevalent?
- What Java Container / App Server is used the most?
- How much development time is spent redeploying Java containers?

We asked the people we met at JavaOne 2010, emailed previous survey respondents and promoted the survey in the media. They were sent an email with the link to the survey and told they would be notified again once the results were made available. Participants were offered a chance to win free JRebel licenses and one recipient will receive an iPad once the final results are published.

A total of 1027 responses were analyzed to create this report. Of these, 28% are JRebel users - they were requested to provide data from before they were purchasers. The raw data including all calculations is available for download.
Build Tools and IDEs – Setting the scene

What is more popular, Ant or Maven?

We asked everyone to mark the tools they use for building their application:

As you can see, Maven and Ant are used almost equally and some of the respondents use both. Clearly both of them are useful in some way. Tools like Gradle, Ivy and SBT were mentioned, but none of them gathered even 5 votes.
Which Java IDE is the most popular?

The IDE results show a much wider distribution of preferences:

In terms of user base, Eclipse has emerged here as the clear leader, with only one-third of respondents using IntelliJ IDEA (#2) and Netbeans (#3). Perhaps including MyEclipse separately and not doing the same for other Eclipse distributions like RAD was not the best idea, that’s something we’ll do better next year. It is interesting that according to this chart Oracle’s [ex]-flagship IDE, JDeveloper, is even more marginal than the small independent player MyEclipse.
Servers & Frameworks – Who are the popular kids at school?

Which Java Containers / App Servers are used the most?

Container Popularity

- Tomcat 33%
- JBoss 26%
- Weblogic 10%
- Jetty 8%
- Glassfish 8%
- Websphere 7%
- Other 8%

Compared to the last year’s chart\(^\text{iv}\) we see % gains for the open source containers. We can also see that Jetty and Tomcat have a bigger share than last year, while Glassfish is sliding a bit. The results from these 1000+ developers shows that Oracle WebLogic and IBM WebSphere have lost a total of 8% of the market to the open source containers compared to last year.

We asked people to choose only one primary container in this question, even though in many companies several containers are in use. Since we were interested first of all, in what container do people spend most of their time in development it just made more sense to phrase it like that. So don't make any assumptions as to how this applies to the production deployment, but it's a good estimate for the situation in development.
What Java EE standards are used the most?

As far as Java EE standards are concerned we have the following picture:

Here the market penetration is more important than the comparison aspect. JPA is used almost as much as the venerable JSP, with 37% of market penetration. EJB versions altogether have 39%, which would make them the most popular standard, but there’s likely some overlap between EJB2 and EJB3 users, so the actual total is likely a bit smaller. Up-and-coming CDI standard has gained 6% so far, will be very interesting to see how much this will change next year. We’ll cover JSF in the next section.
Which Java Framework is most prevalent?

Let's take a look at the framework chart:

![Web framework popularity chart]

Spring and Hibernate are by far the most popular and in fact are still used more than standards. However, as far as web frameworks go, JSF looks like a popular choice with 24% of answers.

Unfortunately, we haven’t separated Spring MVC from Spring, but we assume (interpret this as you must) that it is at least as popular as JSF. The rest of the frameworks hold a share of 10% or less (GWT is barely over that and is the third most popular).

The third most popular framework in our survey was GWT, followed closely by Struts 1. We asked [Matt Raible](http://mattrabile.com) if these results aligned with his recent controversial presentation [Comparing JVM Web Frameworks presentation from Devoxx 2010](http://mattrabile.com/presentations/2010/jvm-web-frameworks) and here’s what he had to say:

"It's surprising to see that framework popularity closely aligns with the JVM Web Framework Matrix results I calculated at Devoxx. I had Spring MVC and GWT listed as the top JVM Frameworks, along with Ruby on Rails (running on JRuby of course)."
"While this matrix was the result of much controversy, I think it gives developers a decent technique for choosing a web framework to use. Of course, the way to choose a web framework is to pick a few you like, prototype with them and see which one satisfies your needs to best.

"More than anything, make sure the developers on your team like developing with it, as that's likely to be key in their productivity using it.

"I'm surprised to see so many folks using JSF, but I do understand why many companies choose it because it's such a "safe" choice as a standard. In fact, you could say that Struts 1 and Spring are pseudo-standards based on their popularity.

However, just because frameworks are popular doesn't mean you should automatically use them. There's other great component-based frameworks in Java (GWT, Wicket and Tapestry come to mind) that are easier to use than JSF."

"I think the biggest thing that component-based frameworks need to work on is their REST support. I think more and more components are being developed on the client (with frameworks like jQuery, ExtJS and SproutCore) and web frameworks should try to embrace that. I especially like frameworks that allow emitting JSON and HTML from the same classes. This allows for easy web page and API development at the same time."
Turnaround Time - How different J2EE environments compare

This year we asked questions about the time it takes to redeploy the application and number of redeploys per hour. We also asked those who don't have to redeploy to comment on why this happens. We specifically asked everyone to provide the answers **without using JRebel**. We were glad to see the development in this area since last year:

- "I wait till I have enough changes to warrant a redeploy" -- we've got a few comments like this, it's definitely one of the ways around the problem, but it can lead to a very artificial way of coding with little incremental feedback.
- "We use tests (unit tests, integration tests, etc)" -- if you have enough tests (and especially if you are not doing UI), it's a great way to both avoid redeploys and ensure that the delivered system is stable and correct. Kudos to those who do this!
- "We use Eclipse/Sysdeo/Intellij IDEA/etc HotDeploy" -- for applications that deploy quickly this is a great way to improve turnaround, by making the IDE republish changes every time they occur. Unfortunately for bigger applications it saves clicks, but not minutes.
- "We use HotSwap/Eclipse Debugger Session/etc" -- The built-in class replace mechanism is a great way to make small changes, but limited only to method bodies. Also, if you can’t use exploded deployment, you still have to redeploy for any change in JSP/HTML/etc.
- "We use OSGi" -- only a couple of answers like that, but in a well designed OSGi application redeploy can be very quick, as only the changed bundle needs to be reinitialized.
- "We use Grails, Play! or Tapestry 5" -- a few answers for each of those, great to see that these extremely productive frameworks are being used.
- "We use JRebel" -- the most popular answer, but not surprising considering that 28% of respondents were JRebel users :)

We originally did not include a “30sec” answer to the question “How long does take to redeploy your application?” However, we have since received a few comments about that, and fixed it halfway through the responses. To compensate for that, we will use a “48sec” (0.8 min) value for the “1 minute” answer.
How long does it take to restart your container and redeploy your app?

This was the first question we asked. The answers are distributed below:

We didn't ask with such accuracy last year, which makes it harder to compare, but it seems the trends have stayed the same. The average redeploy time is 3.1 minutes, but the standard deviation is 2.8, which means that the redeploy time varies greatly.

It can be noted that a statistically significant segment of respondents (just over 1 in every 10 developers) responded that it takes over 10 minutes to fully redeploy.
In an hour of coding, how many times do you redeploy?

The average frequency is **4 times an hour** with the standard deviation of **3.2**.

With these two questions answered, we can make a reliable estimate for the total time spent redeploying in an hour. We began by removing those who don't redeploy at all and those that reported redeploying more than 60 minutes an hour (ummm....what?)
How much time per hour do you spend redeploying?

The average respondent spends about **10.5 minutes an hour** redeploying with a standard deviation of **9.8**. This is about **17.5%** of total coding time. If you consider 5 hours out of each day as "coding time", and assume 4 weeks out of each year are vacation time, then on average **5.3 full**, 40-hour workweeks per year are spent exclusively redeploying and restarting.
Redeploy time per Java EE Container (as a % of overall coding time)

Finally, we can also see how the choice of container correlates to the time spent redeploying. Note: This doesn't mean that bigger containers are that much slower, rather bigger apps influence the choice of container:

![Percent of coding time spent redeploying](image)

Not many changes from last year here, although it's nice to see the improvements in Glassfish v3.
Conclusions?

Interpreting data is always a hard and dangerous task. A sample of 1027 developers is reasonably large. But some bias is possible, as it's likely the more active developers bothered to answer.

The data gives a fair overview of what the Java development ecosystem looks like. Probably the most interesting thing is that, despite what some say, the world of open source and commercial software manage to coexist quite well.

Of course with companies like JBoss and SpringSource it's becoming increasingly difficult to distinguish the two, but commercial interests are present on every single graph (if you include Sonatype backing Maven). It will be interesting to see how this balance develops in the upcoming years:

- Whether it will be the vendor-backed standards or community-backed frameworks that will find the most use.
- How much the Open Core will be adopted by projects like Jetty, Tomcat and GlassFish and their commercial backers.
- Will the closed-source containers and tools survive in the ecosystem? Will their role diminish or increase?

Come back next year, let's see what happens!
Footnotes and URLs

i http://www.zeroturnaround.com/blog/java-ee-container-redeploy-restart-turnaround-report/

ii http://www.zeroturnaround.com/survey2010/


v http://raibledesigns.com/

vi http://raibledesigns.com/rd/entry/my_comparing_jvm_web_frameworks

vii http://raibledesigns.com/rd/entry/how_i_calculated_ratings_for