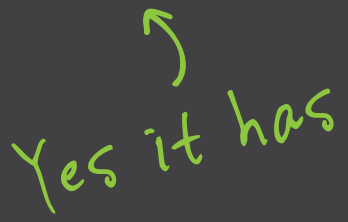
CODING WITH JREBEL

JAVA FOREVER CHANGED

By Arnel Pällo



ZEROTURNAROUND

Abstract

For those with only few minutes to spare, this paper addresses the ways in which JRebel, Java's famed redeploy killer, has made an impact on how developers spend their day coding. With JRebel, developers get to reload code changes instantly, fine-tune their code with incremental changes, debug, explore their code with ease (both locally and remotely), and ultimately spend more time learning and communicating with colleagues instead of waiting for the dreaded application redeploy to finish.

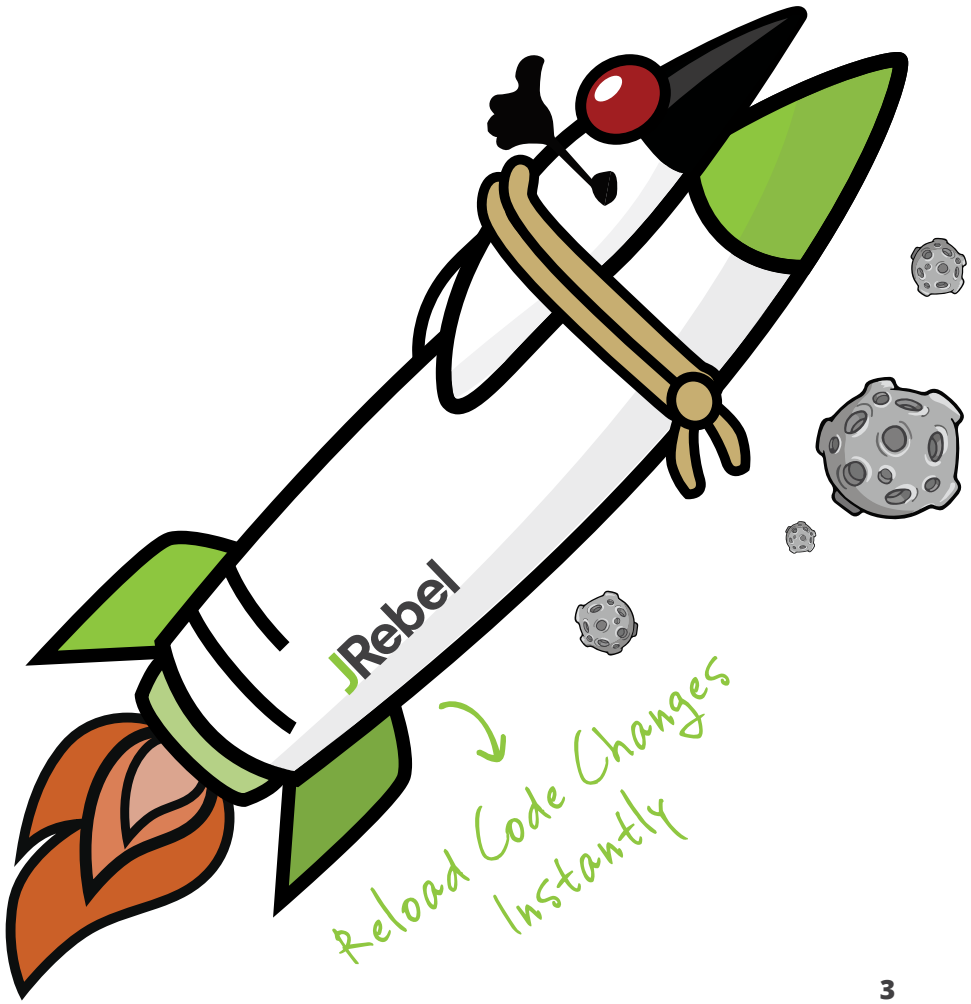
ZeroTurnaround is transforming how software is created, enabling development teams to build better software faster. Powered by award-winning Hotpatching Technology, JRebel and XRebel are revolutionizing the way development teams work with Java. Thousands of engineers use JRebel to turn the laborious build/test/deploy phase of application development into a light-speed cycle as fast and iterative as Python or PHP, while XRebel, the lightweight Java profiler, helps developers get insight into their application's performance.

Introduction

Java's bane has always been a reliance on tools. If all I had for writing code was Notepad and a command line compiler, I would have gladly considered a worthwhile career in street-cleansing, fruit-picking or subway-guitar playing industries.

But we have a handful of modern IDEs. We also have a huge ecosystem that provides us with amazing OSS libraries and application servers. And since we have JRebel, gone are the days of reading through the whole series of Harry Potter novels in one week while waiting for the application to redeploy itself after fixing that spelling typo, only to discover there's one more.

This post reviews some of the great advantages of using JRebel in Java development, and how eradicating the daily, time-consuming process of redeploying your application to see your changes has long-lasting effects on the way that you will write code in the future.



Java changed forever #1

SEE ALL CHANGES IMMEDIATELY

Does your code smell? Find out really quickly and run your own mini-tests throughout the day. Remove bad code before it goes to QA and they make fun of you.

With JRebel, you get to reload the your changes immediately after compiling them. When the IDE builds classes automatically, the process is as fast as using a scripting language or testing your latest HTML changes. In fact, I urge you to try compiling the following code snippet, start it up with JRebel enabled, and change the "Hello World" part to something else while the program is running.

```
public class TestLoop {
 public static void main(String... args) throws Exception {
   while (true) {
     saySomething();
     Thread.sleep(1000);
   }
 }
 private static void saySomething() {
   System.out.println("Hello World");
 }
}
```

For me, the console output following the JRebel banner was (yes, I write fast):

```
Hello World
Hello World
Hello World
JRebel: Reloading class 'TestLoop'.
Goodbye Redeploys
Goodbye Redeploys
```

This opens up a world of possibilities that have never before been available for developers. It combines the power of Java with the fast turnaround of dynamic scripting languages. You don't have to queue up a gazillion changes before redeploying. You can code by making small incremental changes and alt-tabbing to your browser to verify the results.

Java changed forever #2

FINE-TUNING THROUGH INCREMENTAL CHANGES

This works best when you have to implement some logic that is tricky to get just right. Imagine you have to add validation to your form. With JRebel, you can add the rules field by field and play around with the way errors should be displayed until satisfied by the results. You can even add some new fields to your controller or validator with JRebel as well. By the way, JRebel preserves all application state, which means it does not matter how you reached the page you're modifying.

JRebel actually goes way beyond "simple" class reloading. Applications today take advantage of frameworks and Java EE technologies, such as EJBs, Hibernate, Spring Framework, JSF, Seam, Struts, etc. The thing with frameworks and technology implementations is that they initialize themselves when the application starts up and generate lots of metadata. It is usually a good thing since nobody wants Hibernate to re- initialize itself for every HTTP request (that's what PHP programs have to do, by the way, even with opcode caching enabled).

JRebel has addressed this by having explicit support for almost 90+ popular frameworks and application containers. This means when you make a change to a "special" resource such as adding a new EJB, changing JPA mappings (either via xml or annotations), or add new Spring Beans (again, either via xml or annotations), JRebel updates the metadata to reflect those changes. Your job is to press F5 in the browser.

Java changed forever #3

DEBUG REMOTELY FOR FUN AND PROFIT

JRebel makes remote debugging more useful than ever. Imagine you are using it to hunt down some difficult bug. Now you can hit a breakpoint, investigate what's going on, make a change, resume execution, and see the result almost immediately! The new code kicks in right after compiling, next time when the affected method is invoked. You can use this approach to fix discovered bugs immediately or to quickly add some debugging statements in the code (just remember to avoid committing them). You can even drop to a parent frame and check the effect immediately!

Java changed forever #4

EXPLORE THE CODE WITH EASE

One of the benefits of small turnaround is that you can test the framework features interactively. You can write a small method to test some functionality of a new framework and learn hands on how it works. Python and Scala even have interactive shells.

The problem here is the applicability in real world. While exploring the API of Java libraries is easy (even if they lack documentation), it can be ridiculously tedious in some other languages, especially when they are dynamic in nature, lack documentation and/or miss out on good IDE support.

Even if you can indeed enjoy the fast turnaround with PHP and the like, it has little net value when you have to dig into some big and obscure framework (like Joomla CMS, which I actually tried that a few years ago, and it wasn't pleasant) to see how it works. The ability to ctrl-click and dive into the code, as well as enjoy the auto- completion would be great, if only they were there. Granted, there is some support for it in modern IDE's, but it is nowhere nearly as useful. The difference is in the order of magnitudes.

Java, on the other hand, suffers from the fact that while the 3rd-party code is easily navigable, writing the stubs and running them can be tedious or time consuming. Writing stand alone tests requires effort to set up and are not exactly a substitute for a real-world use case.

Embedding a test code into an existing application is simple and easy, but restarting the server or redeploying the app to see the changes takes time...unless you're using JRebel to eliminate the need to restart.

Java changed forever #5

DEPLOYING CODE LOCALLY OR REMOTELY

It's great when a developer can check out a code from a repository, compile it, and deploy the thing in his own machine. Sometimes this is not possible.

It can be due to performance reasons, or the whole system is simply too complex. Sometimes the system has to run in a different network than the developer's, which cannot be easily simulated. Some developers may be using low-powered laptops. Whatever the reason, sometimes running the app in the same machine is not an option.

I can almost hear developers cry (myself included) when that is the case. JRebel can help even in this case and the capability to make a system deployable on a developer's machine is not crucial anymore.

JRebel Remoting was introduced in version 4.6, which lets you push changes from a developer's machine to a remote server. It works over HTTP on the same port as the server is serving the application itself, therefore needs close to zero configuration to set up and no changes to firewall settings. We have examples on our website where we show JRebel Remoting in action when deploying to Amazon EC2, so if it can work well in the cloud far away, it will definitely work in your LAN or on the other side of the city.

You can even let multiple developers loose on a single application instance on a remote machine. By default, JRebel Remoting reverts back to the initial codebase when the app server is restarted. You can set up an application on the remote machine, have a developer have his way with it, restart it (ok, there's one redeploy, sorry), and allow another developer do his magic on the app. This is the case when both developers are making changes to the same application. When there are several apps deployed to the remote server, developers can bombard them without stepping on each other's toes.

Java changed forever #6

NO RESTARTS = MORE TIME FOR BETTER STUFF

No restarts means no forced interruptions, which means devs have more time for doing better things. Sometimes it's hard to maintain concentration on a single task for more than a few minutes, and that's when the level of focus is totally up to the devs themselves. If there was a circus clown jumping out from behind a door every 6 minutes to frighten me, I couldn't imagine working properly throughout the day.

With JRebel, you get more time to meet your deadlines, research & learn and *gasp* communicate with people. According to a recent report on Developer Stress, it came out that developers really do care about making deadlines and their level of expertise, wishing they had more time to spend on education.

In a related post, we also discovered that for every 1 hour developers spend writing code, they spend 30 minutes dealing with communication overhead (i.e. meetings, chats, reporting/timesheets, etc). JRebel's instantly class reloading magic simply frees up more time for doing the things that developers are already spending time on.

Final words

JRebel will give you the best of both worlds – the performance, scalability and robustness of Java, as well as the FTL-turnaround and the feeling of lightness, which are typically associated with 100 line guestbook applications, not with big mature systems. JRebel behaves as if it's in Rome – you don't have to make changes to your code, architecture or sysops. It adapts to your environment and requirements. All you have to do is start using it. You get to keep all the effort and investments already made to your infrastructure.

We've heard from thousands of developers directly about how JRebel has forever changed not only how productive they work – saving more than 5 full work weeks of time otherwise lost to Java restarts each year – but the way in which they actually write the code, how they discover faults and test iteratively in order to minimize service requests and bugs.

The joy we provide is achieved in a sensible way, not via some dark magic of something "on rails". Nothing in your entire app (frameworks included) is vague, ambiguous or hidden from you. All we do is some good engineering with JVMs and popular OSS frameworks. And only within the walls of Hogwarts (i.e. during development). The delivered software has no runtime penalties or lingering dependencies.

Arnel Pällo is a Java engineer at ZeroTurnaround.
He works on adding support for web-service frameworks into JRebel and is a major fanboy of pretty code. He plays Bioware games and hacks his Android phone on weekends. He also likes salsa dancing and can make a great Mojito.
You can connect with Arnel on Facebook.

[R]EVOLUTIONARY
DEVELOPER TOOLS

Contact Us

Twitter: @zeroturnaround
Web: http://zeroturnaround.com
Email: info@zeroturnaround.com

Estonia
Ülikooli 2, 4th floor
Tartu, Estonia, 51003
Phone: +372 653 6099

USA
399 Boylston Street,
Suite 300, Boston,
MA, USA, 02116
Phone: 1(857)277-1199

Czech Republic
Jankovcova 1037/49
Building C, 5th floor,
170 00 Prague 7, Czech Republic
Phone:+420 227 020 130