MYTHBUSTERS: FIVE SCENARIOS WHERE JREBEL HELPS DEVELOPERS

create more, higher-quality Java code

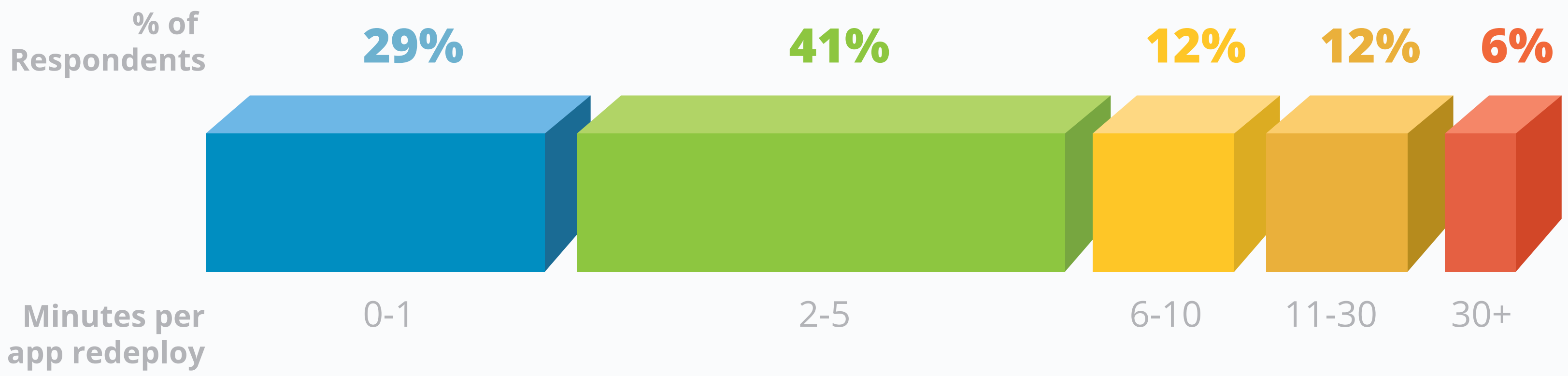
RogueWave SOFTWARE | JRebel

Redeploys: The hidden productivity killer

Simple and powerful, Java remains the default language for enterprise development everywhere. Not to say it doesn't have its downsides. Chief among them is the lengthy redeployment time after code changes.

Redeploys are a huge productivity killer. Studies have consistently shown that it takes multiple minutes for most code changes to take effect. When this is repeated several times an hour, it quickly adds up to ~20% of a developer's salaried time that is spent waiting for redeploys. Yet many developers — and their bosses — are resigned to waiting around for redeploys as an unfortunate fact of life.

Wasted development time spent on building and redeploying



Source: Java Tools and Technologies Landscape Report 2016

Redeploys: The hidden productivity killer

JRebel blows up that myth. JRebel fast-tracks the build-redeploy-test cycle for Java developers by skipping the time-consuming build and redeploy steps. Developers are more productive since they can view changes to code, classes, configuration, and assets in real time, all while maintaining application state and eliminating restart times.

JRebel doesn't just provide time savings. Instant code changes also make it easier for developers to pinpoint and resolve bugs quickly, for cleaner code. **This lets developers maintain their momentum and do higher-quality development, faster. And lets you bring products to market faster and gain a vital edge over the competition**.

You may still harbor doubts that JRebel can provide all these benefits, especially in your specific environments. In this eBook, we'll **bust those myths** by walking you through five popular use cases for Java developers, and demonstrating how JRebel can work in those environments to transform your development process for huge productivity gains and cleaner, better code.

MYTH #

1 Spring Boot is "fast enough" without JRebel

Spring Boot enables developers to create stand-alone, production-grade Spring-based Java applications that you can "just run". Spring Boot has gained strong traction over the last few years because of how easy and fast it is to get applications running, thanks to the ability to embed most commonly-used technologies into it with a single line of code.

One of the biggest misconceptions about Spring Boot is that the application restart time there is "fast enough" such that there are no benefits from using JRebel. This might be true for a fresh Spring Boot app, which might take only 10 seconds to restart. However, as soon as you start adding more classes, endpoints, logic, databases, etc., the restart time steadily increases as well. For many developers, that restart time grows beyond their acceptable pain-threshold at which they would benefit from a class-reloading technology such as JRebel.

We ask our JRebel users to report the restart/redeploy time for their app without JRebel, which we use for ROI calculations. For Spring Boot users, the average app restart time is 151 seconds, or 2.5 minutes. That's **far longer** than the 10 second redeploy time of a fresh/empty Spring Boot app. Clearly, there is room for improvement — which JRebel can help provide.

And what about Spring Boot Devtools? It’s an open-source developer package used by many Spring Boot users. It’s similar to JRebel in that it tries to speed up development. However, its application restart approach has many limitations. Application state is not preserved and must be reproduced with a full restart. Moreover, Pivotal, the creator of Spring Boot Devtools, recommends JRebel, stating **“If you find that restarts are not quick enough for your applications or you encounter class loading issues, you could consider reloading technologies such as JRebel.”** Finally, Spring Boot Devtools is unable to measure how much developer time it saves and thus validate its time-saving claims.

Contrast this with JRebel, which, in addition to Spring Boot, supports 100+ Java frameworks. JRebel can automatically rewire and update framework configurations, and it maintains application state during code changes with ease.

As a class reloading solution, JRebel ensures that only the code that is changed is acted on, foregoing the need for unnecessary restarts. JRebel also measures the developer time it saves and the redeploys it prevented, letting you know just how many developer-hours — and dollars — you are saving.

All in all, JRebel provides great benefits to users of Spring Boot. Besides making Spring Boot developers much more productive, it also provides much more features and support than Devtools. So it won’t just be your developers that thank you — so will your users and business leaders.

MYTH #

2 Your IBM WebSphere and Oracle WebLogic applications are too mature for JRebel

It's no secret: the bigger the application, the longer it takes to redeploy. And established enterprise applications, the kind using WebSphere or WebLogic as application servers, can be huge. That can sap the productivity of your developer team.

According to JRebel users working with WebSphere apps, it takes an average of 6.6 minutes for an app restart. For the average WebLogic user, it's 7.5 minutes. With a reported average of 3.1 redeployments and tests per hour, the average WebLogic user spends 23 minutes per hour, or 38% of their 8-hour workday, on redeployment (it's only 25% of the workday for WebSphere users). With a team of just 10 developers, your annual cost from redeploys could easily **run between $200,000 and $400,000** in lost productivity.

Clearly, there's a huge cost in time and money from redeployments. However, many developers have been using WebSphere or WebLogic for so long without JRebel that they've gotten used to the enforced delays, the mid-afternoon-redeployment-coffee-runs, the dubious hacks such as running a lighter-weight alternative application server during development that creates hard-to-track-down bugs when they try to move the code into production.

WebSphere and WebLogic users shouldn't have to put up with the status quo. With JRebel, they can still enjoy the enterprise scale and stability of developing on WebSphere or WebLogic, but without the missed deadlines and stress resulting from sluggish development cycles. With JRebel, your developers can test their code while they're writing it and make changes without losing context or focus. Besides creating better-quality code, this also avoids time-wasting forced redeploys.

JRebel is as easy to install as adding a single JVM argument to your startup script, or install an IDE plugin if you're using a mainstream IDE like IntelliJ IDEA, Eclipse, MyEclipse, RAD, or NetBeans.

For about $1.50 a day with JRebel, you can enjoy huge productivity gains, write better software, and enjoy doing it. JRebel is so powerful that IBM and Oracle, the makers of WebSphere and WebLogic, are both customers, and we speak at WebSphere conferences. What better proof is there that JRebel can help WebSphere and WebLogic users?

MYTH #

3 JRebel doesn't integrate with SAP Hybris

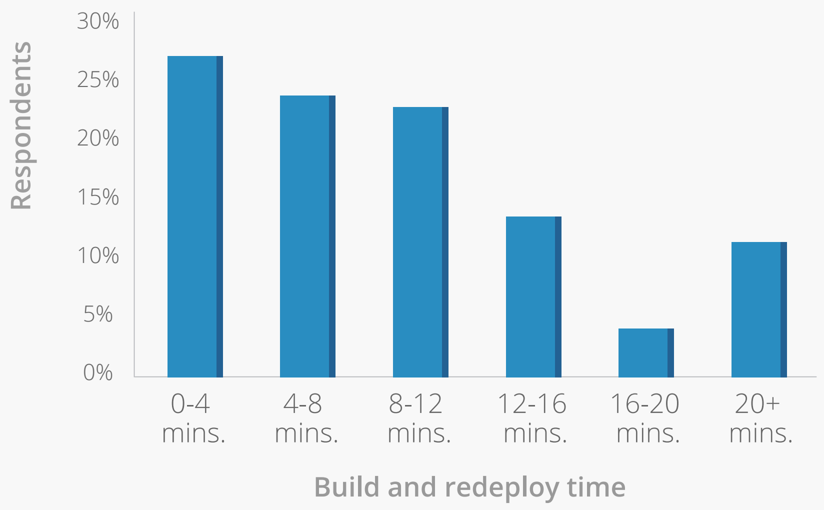
SAP Hybris

SAP Hybris is a B2B and B2C e-commerce platform long used by large enterprises, especially financial services and e-commerce companies. It includes many components — commerce, marketing, sales, service and billing — while the core commerce application runs on top of a Java application server.

While developers enjoy Hybris's scale and unified environment, its massive size also makes "ant all" scripts to restart Hybris applications very sluggish. Of all the platforms out there, Hybris is widely considered to **impact its developers the most**, by the time it takes to reload class and configuration changes, restart servers, and restore application states.

According to our users, the median Hybris application takes just over 8 minutes to redeploy without JRebel. That's comparable with the lengthy restart times for WebSphere and WebLogic applications.

How long does your build + redeploy take without JRebel?



70% of users report times higher than 5 minutes

50% of users report times higher than 8 minutes

Source: Product metrics from JRebel Hybris users

SAP Hybris applications, like WebSphere and WebLogic ones, have often been around a long time. Developers have learned ways to get by. And the sheer scale of the Hybris platform may have many developers doubting that JRebel could possibly integrate well with it.

That's not true at all. In fact, JRebel can reload almost all code and configuration changes you make with Hybris, allowing you to skip the "ant all" process while preserving application state. JRebel also has comprehensive debugger support, provides plug-ins for all the major IDEs to help with setup, configuration and usage, integrates with key frameworks commonly used in Hybris such as Spring MVC, ZK, and JSF, and supports remote servers, to enable class reloading if you are running a virtualized Hybris environment.

.

All in all, JRebel improves Hybris developer efficiency by 33%. The financial ROI is huge. Drawing on figures collected from our existing customers, **JRebel can save a 10-developer Hybris team $267K per year**. To calculate the ROI for your Java development team, visit jrebel.com/pricing.

Let's not forget all the deadlines you'll meet early, the better-quality code you'll produce, and the stress you'll no longer have when you use JRebel to speed up your Hybris development.

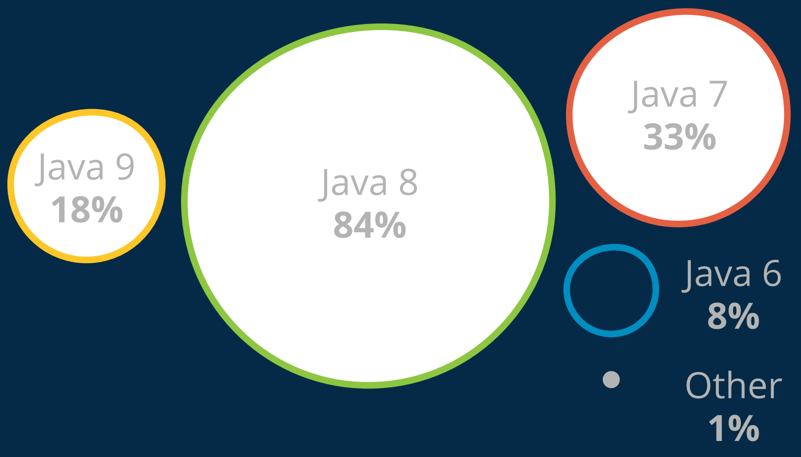
MYTH #

4 JRebel doesn't work in complex environments and custom architectures

First released publicly in 1996, Java continues to rule the enterprise development scene. There are many new Java applications being created, as well as older applications — some nearly as old as the language itself — still being constantly updated.

Maybe you oversee one of those dot-com-era applications. Or maybe your Java environment happens to be incredibly intricate and complex, full of customized code, legacy plug-ins, and obscure frameworks. Or maybe you're extremely skeptical that any plug-in like JRebel that promises to accelerate development times could actually work in your non-standard environment.

Which versions of Java do you use regularly?



Source: The State of Developer Ecosystem Survey in 2018

Well: challenge ACCEPTED. Because however custom or non-standard your environment may be, there’s a very good chance that JRebel will integrate with it. JRebel supports the majority of Java frameworks, IDEs, and application servers out there, including 100+ Java frameworks, plugins for all the major IDEs, and all of the main JVMs and application servers in use, including many older versions. Moreover, JRebel can be set up with complex modular builds, whether they are based on build automation tools such as Maven, Gradle, or something else.

Bottom line: don’t assume you can’t enjoy the benefits of JRebel due to your complex or customized enterprise architecture. View our list of supported frameworks, IDEs, and application servers. Even if one of your components isn’t listed, JRebel probably works out of the box. Contact us at support@jrebel.com if you have specific questions about your environment.

MYTH #

5 Remote development on virtual machines or cloud environments is too much for JRebel to handle

Cloud environments and virtual machines (VMs) have many advantages — efficient use of resources, low start-up costs, easier scalability, (often) less complexity, and more. No wonder many enterprises have moved their Java applications using these technologies or are doing so today.

Every change has its implications, however. For developers, uploading code changes to remote, distributed applications can take longer than locally-hosted ones. Similarly, VMs and containers add another layer of infrastructure that may need to be restarted for changes to code residing inside of them to take effect.

JRebel handles modern technologies such as cloud and VMs with ease. Say your Java application is residing on multiple remote application servers. With JRebel, you can push your code changes out to all the servers at once with one click. You just change your code, have your IDE compile it, click the button, and JRebel takes care of the rest (the uploads and updates to the server).

Aside from making rollouts of code changes to remote servers easy, JRebel still delivers its core value: the ability to skip virtually all time-wasting redeployments. Say you're building a huge WAR file of all the affected development files every time you change your code, which you then need to wait on while it's uploaded to the remote server, right? Not with JRebel. JRebel keeps track of what you've changed, so you only need to upload and reload the delta.

JRebel also enables you to locally host cloud files and resources outside of the hosted virtual images, so you don't have to do a full restart in the VM and/or on the cloud. The ability to view and test code changes in real time on the local app server helps developers improve application quality. And JRebel supports popular containers like Docker and popular cloud PaaS (Platforms-as-a-Service) including AWS (Amazon Web Services), IBM BlueMix, and Cloud Foundry.

Enjoy all the benefits of containers in the cloud and none of the potential latency by using JRebel.

Conclusion:
Accelerate development velocity — with JRebel

Sometimes we get asked about class reloading technologies such as HotSwap or DCEVM that also purport to save time for Java developers. Our view is that compared to JRebel, both HotSwap and DCEVM are extremely limited in terms of the Java versions, JVMs, application servers, and enterprise Java frameworks that they support. Neither HotSwap nor DCEVM supports the same breadth of change types as JRebel. And HotSwap also lacks the ability to track your time savings, something JRebel has well-covered. In other words, JRebel is a true enterprise tool — HotSwap and DCEVM are not.

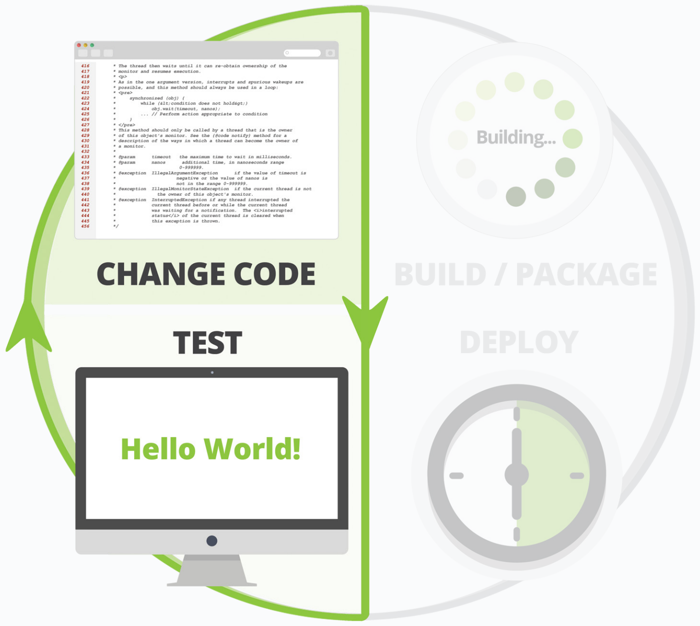
The sprawling Java ecosystem is less a landscape than an ocean. There are always new Java features, frameworks, application servers, build systems, and more emerging from beneath the waves. At the same time, many other technologies are receding or sinking under the waters. As an enterprise development leader, you need to stay afloat by keeping track of all these changes, while your developers keep up with product features and bug fixes.

The greater challenge is this: **how do you ensure that your team can tackle both ordinary and extraordinary technical changes while maintaining a foundation of quality, security, and release velocity that’s vital to your business?**

Conclusion:
Accelerate development velocity — with JRebel

JRebel can insulate your team from change at the code and platform levels. JRebel eliminates the hidden costs in critical parts of the development process. That translates into massive time savings and productivity gains that preserve your budget. No matter what Java environment your applications run in, or however customized, complicated, or virtualized your architecture is, JRebel will likely integrate into it, saving your teams time and money. JRebel also helps teams bring their products to markets and internal users faster, allowing enterprises to gain a competitive advantage that will translate to a better bottom line.

Take finer control of your development velocity, enable your developers to do more, and gain a business edge with JRebel.



Evaluate JRebel for free
jrebel.com/pricing

RogueWave SOFTWARE | JRebel