IBM WEBSPHERE AND JREBEL

ALL THE JAVA EE GOODNESS WITHOUT THE WAIT

ZEROTURNAROUND

Why Companies Like WebSphere

At ZeroTurnaround, we get to see a LOT of different development environments. There are many debates on the merits and failures of various application servers, like ours from earlier this year: The Great Application Server Debate.

One of the appservers that we see quite often is **IBM WebSphere (WAS)**. WAS has been around for quite a while, **since 1998**, and it boasts an impressive feature set. It's scalable, it's got solid and supported implementations of the Java EE JSRs and Enterprise OSGi, and it's an industry standard, so it's reusable information in future jobs.

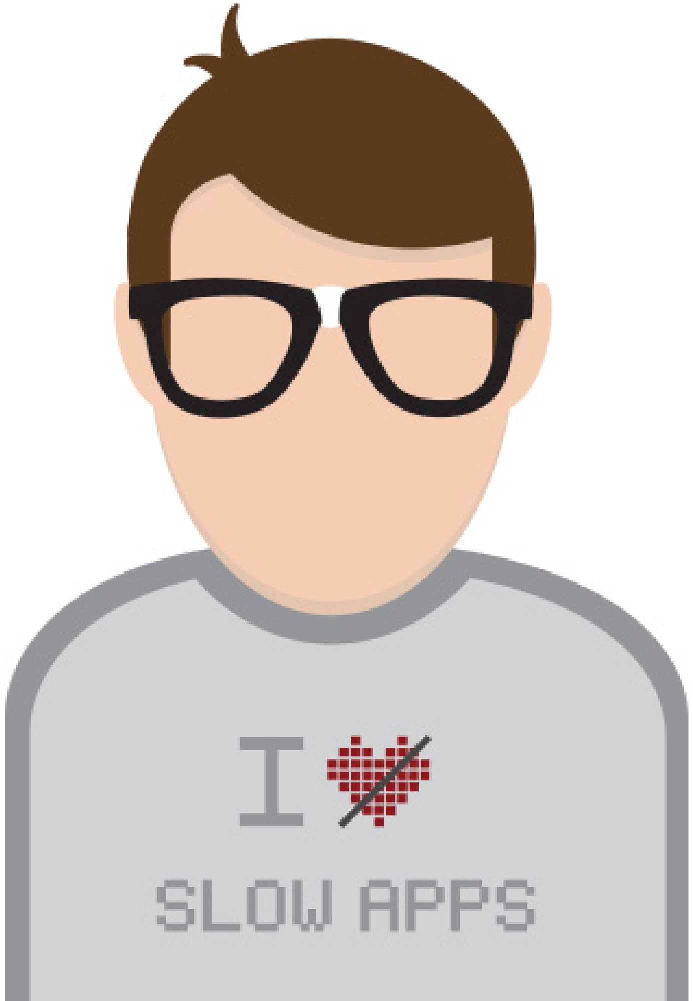
From a production standpoint, WAS is a great choice and is the standard amongst most high availability (due to deployment manager architecture), high throughput industries where uptime and performance are the driving forces - such as finance, governments, healthcare, and Software As A Service providers, especially as the backbone to many of the WebSphere sub-offerings like Portal and Commerce.

since 1998

Why Devs Dislike WebSphere

However, on the development side, the user experience leaves much to be desired. WAS is kind of slow for the normal developer use case, and Java EE applications can be large and unwieldy. Most of the features that make WAS so great for production environments often go unused or even misconfigured in development.
Also, typically the type of critical applications that run on WAS are larger than average. Developers like to write software, not fight with their tools.

When I meet WAS users, a daily occurrence, and ask them if they enjoy using WebSphere, I'm met with with hemming and hawing about how it's the most capable application server, but it is really slow for them in their daily work, and they're willing to either go to great lengths to speed it up or have trained themselves to work their application server redeploys into their schedule
(the **"2:30pm I'm redeploying coffee run"**,
for example).



Just how slow is it?

ZeroTurnaround has collected quite a bit of data from WAS users on their redeploy habits, such as:

> **how many times** per hour they redeploy their application

> **how long** each redeploy takes

> and what percentage of their precious **coding time is wasted** on redeploys.

We gather this information in a survey that is filled out by all of our prospective customers. The survey also asks several questions relating to frameworks and application servers used, and general coding habits of the individual and organization. Through this survey, we have been able to gain some insight into our customers as well as their tooling. From the survey, we have harvested some hard data regarding the redeployment habits and times of developers using WAS, from **version 6.1 all the way to 8.5.5** and **Liberty Profile**.

The average developer using WAS redeploys and interactively tests their application in their development environments **3.2 times per hour**. The average build and deploy phase takes **6.6 minutes**, which tells us that approximately **21 minutes of every hour** are wasted waiting to interactively validate the results of the code change. Given an 8 hour workday, this means that about **25% of the day** is wasted on this redeploy phase. From my own personal experience, this is most likely skewed given the dynamic level of coding activity through the duration of the sprint or life cycle of the project. Most likely very little of the time is wasted in the beginning, since that time is spent planning out the project or tasks for the sprint. However as deadlines approach, more and more time is spent rapidly coding to achieve goals and meet deadlines. What that really translates to is more time spent working overtime as deadlines approach, unhappy and stressed developers, and potentially poorly written code.

3.2 per hour
redeploys & tests

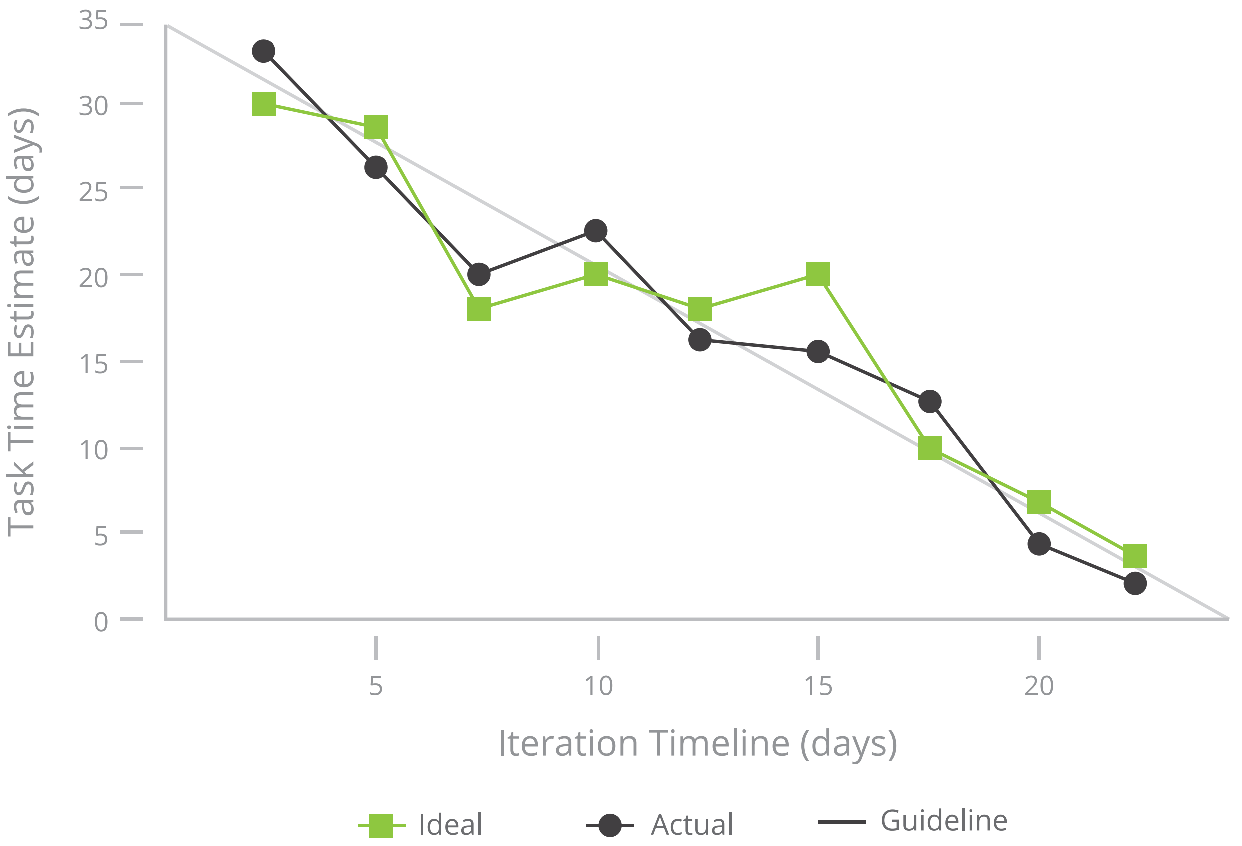
6.6 minutes
build & deploy

21 minutes
wasted per hour

25% of day
on redeploys

Most likely very little of the time is wasted in the beginning, since that time is spent planning out the project or tasks for the sprint. However as deadlines approach, more and more time is spent rapidly coding to achieve goals and meet deadlines. What that really translates to is more time spent working overtime as deadlines approach, unhappy and stressed developers, and potentially poorly written code.

Burndown Chart



How do devs fix this now?

Programmers are always looking for hacks to speed up their development environments and the systems they code against. In the last few years, the forced downtime of using WebSphere and a full Java EE stack during the redeploy phase has become the new target for optimization by developers and their organizations. There are several mitigation tactics. Some of them are ingenious and others are almost insane, and we have seen them all. One of the most common tactics is to use a different, faster, lighter-weight application server in development than production. This generally involves developers running Tomcat or Jetty on their development machines and just packaging the IBM implementations of the Java EE specifications they're coding against. As far as strategies go, this is fairly reasonable, but invariably there is the merge phase where developers try to run their code against a real instance of WAS and experience hard-to-track-down bugs due to differences in implementations.

At the other end of the spectrum, we see dev teams spend so much time trying to inline and pre-build resources and artifacts that they spend as much time working on their build systems as they do developing their products. This practice, while well-meaning, leads to the build system becoming "black magic" and the more junior developers having no idea how their application is actually built. The vast majority of developers on WebSphere just deal with their downtime and try to find creative things to do while they wait.

Enter JRebel

There is a way to keep using Java EE and keep using WebSphere, but completely eliminate the downtime from redeploys. JRebel is a productivity tool that allows developers to more interactively test their code without sitting through the hurry-up-and-wait rebuild/redeploy cycle for their applications and application servers. There are currently **over 65,000 active users** of JRebel, and many of our users use WebSphere. JRebel is as easy to install as a simple jar drop-in into your JVM's arguments, or an IDE plugin if you're using any of the mainstream **IDEs like Eclipse, MyEclipse, RAD, IntelliJ IDEA, or NetBeans**. With JRebel enabled in your environment, you can keep your current tech stack and not resort to any of the compromises previously mentioned. Based on the numbers we have from our surveys, JRebel enables WAS users to be approximately **35% more productive** and finish their projects about **18% faster**.

65,000 JRebel users

The current trend in application server development is moving toward faster and more lightweight application servers, often with smaller feature sets. However even appserver developers acknowledge that no matter how fast the appserver itself is, the application initialization is out of their hands. IBM is making great strides with Liberty Profile, an instance of WAS that boots in **less than 5 seconds**. It's a welcome addition to the appserver landscape, but if your application still takes **2 to 3 minutes to initialize**, you are still wasting a significant portion of your day. JRebel uses bytecode instrumentation to allow unlimited class and resource reloading throughout your application, even in multi-module projects. JRebel also hotpatches frameworks and their configurations. **Over 100 frameworks** are currently supported including the most used portions of the Java EE spec: CDI, EJB, JPA, JSF, all of the JAX components, and more. JRebel also supports making changes to application servers running remotely. A common WAS scenario is to have a VM running WAS and a developer pushing code to the remote instance. JRebel eliminates that downtime as well, and can push the code delta to the remote WAS instance and hotpatch the code without any loss of productivity.

With JRebel, you can test your code while you're writing it and make changes without losing context or focus. This results in vastly better code, as it allows for developers to make quick experiments, see the results, and revert back immediately if needed. For around **$1 per day**, you can make huge productivity gains, write better software, and enjoy doing it. JRebel improves the user experience for the average WAS developer so much, that even IBM is a customer, and invites us to speak at their conferences (IBM Impact 2013 and 2014 - Rapid Application Development with IBM WebSphere Application Server Liberty Profile & JRebel).
JRebel is available as a **free 14 day trial** from our website. Become a rockstar in your organization, and join the rebellion to fight redeploys!

CAUTION
DEVELOPERS AT PLAY
CHANGE CODE
ADD FEATURES, FIX BUGS
BUILD / COMPILE
MY CODE'S COMPILING!
SEE CHANGES
SH*T A TYPO!
Hello wolrd?
DEPLOY
I'M SO SICK OF REDEPLOYING

CONTACT
ZEROTURNAROUND

say hi! :)

Boston
North American HQ
Sales & Marketing

399 Boylston St.,
Suite 300, Boston,
MA 02116
USA
Phone: +1 857 221 9900

Tartu
European HQ
Product Development
& Support

Ülikooli 2, 4th fl
51003 Tartu
ESTONIA
Phone: +372 653 6099

Tallinn
Product Development
& Support

Roosikrantsi 11, 3rd fl
10119 Tallinn
ESTONIA
Phone: +372 653 6099

Prague
Sales & Marketing

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

For more information or to schedule a briefing
go to our website
www.zeroturnaround.com
or send us an email at
marketing@zeroturnaround.com