CASE STUDY

HELIOCENTRIS

ZEROTURNAROUND

COMPANY
SCORE CARD

- **Size of development team:** 6
- **Technologies in use:** JBoss 7.x, Tomcat 6.x, Spring 3.2.x, Spring-Data-JPA, Aspect, AspectJ, Maven 3.x, Netty 4.x, GWT 2.5.x, PostgreSQL 9.x
- **Types of apps being built:** Scalable client-server application with data collection and data mining capabilities.
- **Build and redeploy time without JRebel:** 1min 34 seconds

Company Profile

Heliocentris is a 19 year old company, founded in Berlin, in 1995. Heliocentris specialises in autonomous energy supply and energy efficiency solutions with the aim of replacing diesel generators with “zero-emission” products. The company develops and markets systems and turnkey solutions for customers in industry and science and has three business areas: Energy Efficiency, Clean Energy and Didactic.

Heliocentris

THE REMOTE MANAGEMENT SYSTEM PROJECT

We talked to Miro Kopecky, the Software Research and Development Engineer, about a project he works on. Miro’s main focus is on back-end and middle tiers core architecture and support to front-end tier. The project calls the next generation of Remote Management System (server farm) for Energy Managers. Energy Manager is an embedded system which gathers huge amounts of data from fuel cells, rectifiers, solar-hybrids, various sensors, cameras etc. The data is transmitted back to Remote Management System servers over Energy Manager. This solution must be scalable enough to collect such amount of data from all units as well as mining significant data that should be displayed to the system user.

When Miro first joined the project, back in May 2013, it was already partly developed. When Miro got familiar with the general project requirements and its definition, he proposed to the team that they make general architectural changes, using other technologies to improve the development cycle, the final result and modularity.

The significant part of the new architecture includes usage of the Spring software stack with beans, ACID, security support, Aspect for bootstrapping, PostgreSQL and Netty as NIO server framework for communication with embedded systems over Energy Managers.

THE ARCHITECTURE

Let's look at the architecture in more depth: The project has currently more than 9 independent MAVEN modules and the final architecture has been divided into the tiers (back-end, middle and front-end) for code maintainability and reusability reasons.

The front-end module uses Spring MVC and GWT. The GWT technology is supported by 3rd party libraries as mvp4g, Sencha or dozer and etc. The API modules use Spring MVC to provide JSON/JAX-WS support.

In development, Miro mainly uses Tomcat 6 to test and partly JBoss 7. JBoss was required in the past to provide the full Java EE stack and old Netty 3.2 support, but as most of the JBoss and old Netty dependencies were replaced and removed, the project has become application server agnostic.

LET'S SEE SOME NUMBERS

Let's now take a look at some numbers to see how long typical development tasks take in Miro's environment using Tomcat 6, based on his observations:

Deployment metrics on Tomcat 6

Number of deployments/hour	20
Number of hours in a work day	8.00
Deployment time mm:ss	01:31
Time spent deploying each hour mm:ss	30:20
Redeployment time with JRebel mm:ss	00:03

Table 1 : Deployment metrics on Tomcat 6

One new that was recently added was the ability to send and receive data in a proprietary format over proprietary protocols. This specific development cycle included the design, implementation and testing of byte stream decoders and handlers together with core system functionality.

The data decoding and handling often needs to get the marker reader into the right position of the provided bytestream. So changes to code without JRebel would require a full redeployment, which we ran an measured for both Tomcat and JBoss. Here are the results:

Tomcat

Task	Command	Task
Build time	*mvn clean install*	0:01:11
Server restart time	*shutdown.sh && startup.sh*	0:00:26
Deploy time (DT)	*mvn tomcat:deploy*	0:01:31
Redeploy time	*mvn tomcat:redeploy*	0:01:38
Deploy + boostrap on	*mvn tomcat:deploy*	0:01:45

Table 2.1: Metrics using Tomcat v6.0.37

JBoss

Task	Command	Task
Deploy time	*mvn jboss-as:deploy*	0:01:40
Redeploy time	*mvn jboss-as:redeploy*	0:01:46
Server start time	*./standalone.sh*	0:00:40

Table 2.2: Metrics using JBoss 7.1

Introducing JRebel

Miro decided to try JRebel to see whether it is a reliable tool for the project and useful for the team and company. He downloaded the trial version and installed the plugin into IntelliJ and enabled his local Tomcat and JBoss servers. The project build system is Maven and Miro makes use of the last stable JRebel Maven plugin to automatically generate JRebel artifacts for every module during build.

For his development environment, Miro uses IntelliJ IDEA on a Mac while the rest of the team use Eclipse on a windows or linux machines. During his development iterations, Miro has continually upgraded to the newest versions of IntelliJ IDE 13.x and JRebel 5.x. Every update of both IntelliJ IDE and JRebel was smooth and working without any problems.

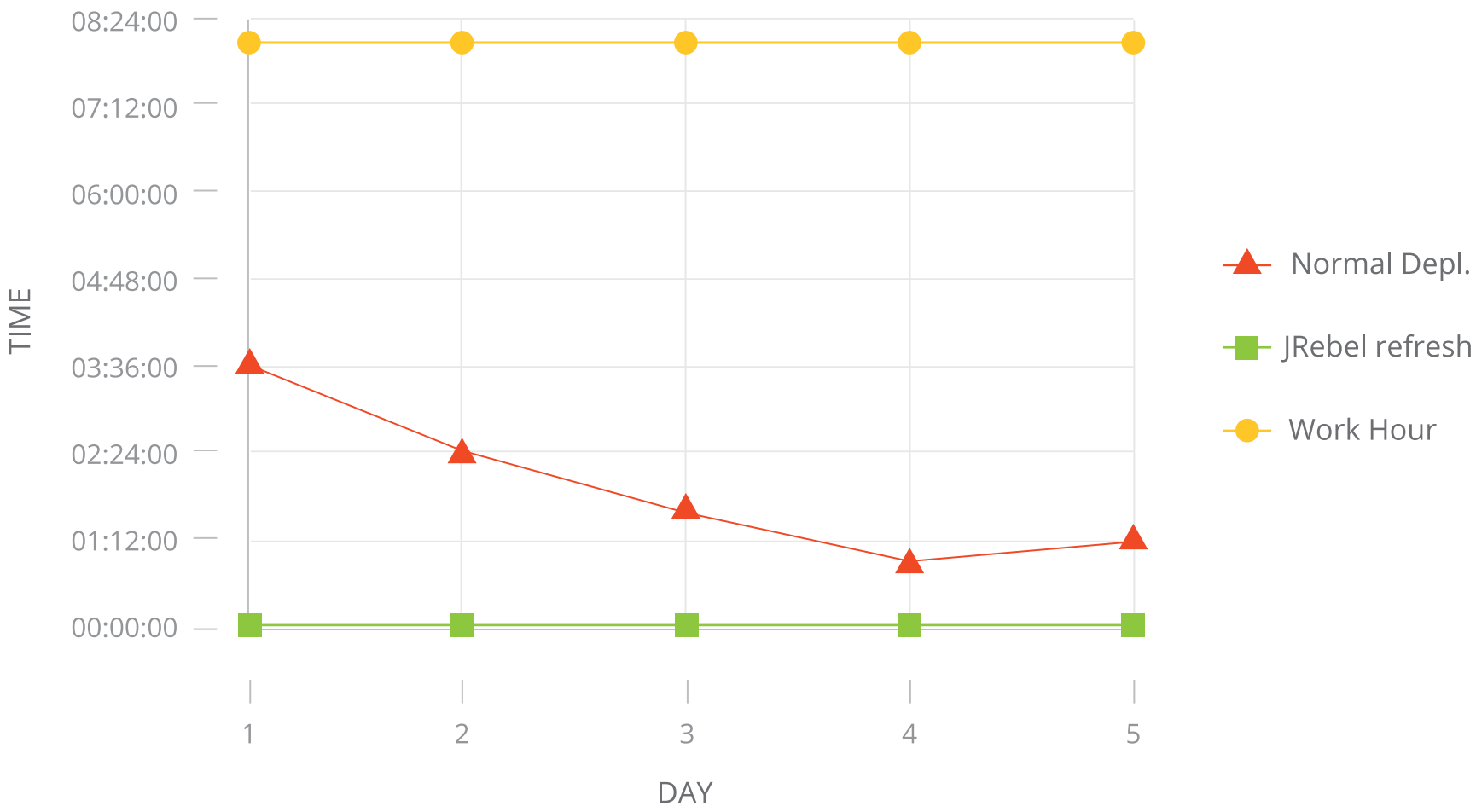
At the time of writing, Miro is testing the next generation Remote Management System on local and remote Tomcat/JBoss server instances. In the future, the team would like to push their test environment out to a remote machines and use JRebel Remoting to provide the same functionality, but currently this is all done locally.

Day	Total Deployment time hh:mm:ss	App refresh time using JRebel
1	3:38:24	0:07:12
2	2:25:36	0:04:48
3	1:37:04	0:03:12
4	0:48:32	0:01:36
5	1:12:48	0:02:24
	9:42:24	**0:19:12**

Tab. 3: Time it takes to view application code changes with and without JRebel on Tomcat 6.x.

Work Week

Img. 1: Graphical display of the time consumption (Tab. 3) over the work week with JRebel employment into the development process.



Summary

Miro's testing with and without JRebel showed the substantial gains which JRebel brought to the development environment. Over the 5 days which Miro measured, Over one of those days was simply wasted, which would be saved using JRebel.

Miro says:

> Once JRebel was downloaded and set up, I was able to continually develop and test my backend code. We aim eliminate wasted time in our short development cycles, every minute counts - JRebel is simply the right tool for the job.

Miro used JRebel in conjunction with other frameworks and libraries. Like the average developer, Miro doesn't just use Java so relies on JRebel framework and library integration to ensure that other changes he makes are also reflected in the live runtime instantly.

Miro's experiences with JRebel, Netty and Spring integration:

> Netty and Spring together with JRebel is real pleasure to work with. The value of time saved is over 50% of my development time.

State is also very important to Miro:

> Almost of all my tasks include redesigning byte stream decoders and handlers according to the proprietary communication protocols. Such redesign massively increases redeployment necessity to see the result of byte stream handling.

Miro's general experiences with JRebel:

> JRebel has had an incredibly positive and valuable impact on the whole project development (backend and middle tier mainly). Personally for me is JRebel the tool that gives me the wings to make a complex project more relaxed and see its simplicity!

Contact Us

Twitter: @JRebel

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
Osadní 35 - Building B
Prague, Czech Republic 170 00
Phone: +372 740 4533