# Foreword

Drools was co-created by Bob McWhirter and I at Codehaus, a fantastic collaborative environment for open source development. Drools 2.0, the first official release of Drools, was a great volunteer effort from a range of people working tirelessly over evenings and weekends.

There is a long-running joke that much of Drools 2.0 was written while under the influence of beer—much to the chagrin of a CTO who was told this little anecdote just after telling us that transactions worth millions of dollars were processed with Drools 2.0.

The success of Drools 2.0 didn't go unnoticed. I was soon given the opportunity, by Marc Fleury, to work full time on Drools at Jboss—now a division of Red Hat. With the backing of JBoss, I was soon able to hire several of the key Drools community members, of Drools such as Michael Neale, Edson Tirelli, and Kris Verlaenen. Each of them has become an evil genius in his own right, making Drools what it is today. Bob is also employed at JBoss, but has since moved onto other Ruby-based endeavours.

I've known Paul for a number of years, throughout which he's been an active and valued community member. I remember the conversation two years back when Paul talked about his desire to write a Drools book. His initial goal was to write a small e-book, but I guess his ambitions and imagination got the better of him, prompting him to do something much more ambitious. So it's with great pleasure that I see Paul finally achieve this, his first Drools book. We didn't make things easy for him. Drools 5.0 was a continually changing platform during it's development cycle, on which he was trying to base this book. The main focus of this book is the web-based governance system, what other vendors call a BRMS. And luckily, most of the important aspects are here in all their glorious detail.

Previous Drools versions have challenged commercial vendors in this space, allowing people to state what they know and not have to waste time translating this knowledge into a machine-understandable format. Drools 5.0 is about to be released, and is a monumental peace of engineering over Drools 2.0, 3.0, and 4.0. It brings together rules, workflow, and event processing, along with an enterprise governance system, to form the foundations of the what I call a Business Logic Integration Platform.

I've always stated that end business users struggle to understand the differences between rules and processes, and more recently between rules and event processing. They just want to model it using some software. The traditional way of using two vendor offerings forces the business user to work with a process-oriented or rules-oriented approach. But this gets in the way, often with great confusion over which tool they should be using to model which bit. If you combine these technologies in the right way and take a behavioural modelling approach, you form something that is simpler and at the same time much more powerful. This allows the business user to work more naturally, where the full range of approaches is available to him or her, without the tools getting in the way. From being process-oriented to rule-oriented or shades of grey in the middle—whatever suits the problem being modelled at that time. We are taking this one step further and are adding event processing with Drools Fusion. Thus, we are creating a more holistic approach to software development. The term 'holistic' is used for emphasizing the importance of the whole and the interdependence of its parts.

The JBoss Drools community continues to grow at a fantastic rate and we are very lucky to get such great feedback and contributions. First-rate commercial support continues to be available from Red Hat, which helps us to continue what we are doing. Red Hat provides the branded version of Drools, JBoss Rules, which goes through additional QA and testing against the rest of the JBoss products, such as JBoss ESB and JBoss AS, and is available under long term support contracts. Thousands of sites worldwide have used Drools as a part of their solution.

So where does Drools go from 5.0? Our initial focus will now be on services and delivering codeless deployments. We will also focus on further enterprise-based governance enhancements as we continue to move up the stack from a simple embedded engine. In reality, we've only just started, and there's still much more of the vision to put into place. So all I can say is, you ain't seen nothing yet.

**Mark Proctor**

JBoss Rules Lead

# About the author

**Paul Browne**'s first job was selling computers in France and things went steadily downhill from there. He spent millons on behalf of a UK telephone company's procurement department and implemented direct marketing for a well-known Texan computer maker before joining the IT department of a company that builds bright red tractors and other seriously cool machines.

Paul then embraced his techie side (he was writing games in machine code from the age of 11) and started a consultancy that used IT to solve business problems for companies in the financial and public sectors in Ireland, UK , Belgium, and New Zealand. Eight years later, he now works with an Irish government agency that helps similar software companies to grow past their initial teething pains.

More formally, Paul has a bachelor's degree in Business and French from the University of Ulster, a master's degree in Advanced Software from UCD Dublin, a post-grad qualification in Procurement from the Chartered Institute of Procurement and Supply (UK), and will someday complete his ACCA financial exams.

Paul can be found on LinkedIn at **http://www.linkedin.com/in/paulbrowne**, and via the Red Piranha (Business knowledge) project at **http://code.google.com/ p/red-piranha/.**

I would like to thank my parents for the gift of learning; my wife and family for the constant encouragement to write this book; the work colleagues that I've had the pleasure to learn from and all the people behind the Drools and other outstanding open software projects.

# About the reviewer

**Peter Johnson** started his computer career in August of 1980, in Burroughs. He programmed mainframes in COBOL and Algol. He started working in Java in 1998, and was the lead designer on projects such as a JDBC driver for the DMSII (Unisys Data Management System II) database that runs on Unisys mainframes. For the past several years he has been the chief architect in a team that does performance analysis of Java applications on large-scale Intel-based machines (8 to 32 CPUs), and evaluates various open source software for enterprise readiness. In addition, Peter is a JBoss committer and is a co-author of the book *JBoss In Action*, published by Manning. Peter often speaks on Java performance and various open source topics at industry conferences such JBossWorld and the annual Computer Measurement Group International Conference.

# Preface

In business, a lot of actions are trigged by rules: "Order more ice cream when the stock is below 100 units and temperature is above 25° C", "Approve credit card application when the credit background check is OK, past relationship with the customer is profitable, and identity is confirmed", and so on. Traditional computer programming languages make it difficult to translate this "natural language" into a software program. But JBoss Rules (also known as Drools) enables anybody with basic IT skills and an understanding of the business to turn statements such as these into running computer code.

This book will teach you to specify business rules using JBoss Drools, and then put them into action in your business. You will be able to create rules that trigger actions and decisions, based on data that comes from a variety of sources and departments right across your business. Regardless of the size of your business, you can make your processes more effective and manageable by adopting JBoss Rules.

Banks use business rules to process your mortgage (home loan) application, and to manage the process through each step (initial indication of amount available, actual application, approval of the total according to strict rules regarding the amount of income, house value, previous repayment record, swapping title deeds, and so on).

Countries such as Australia apply business rules to visa applications (when you want to go and live there)—you get points for your age, whether you have a degree or masters, your occupation, any family members in the country, and a variety of other factors.

Supermarkets apply business rules to what stock they should have on their shelves and where—this depends upon analyzing factors such as how much shelf space there is, what location the supermarket is in, what people have bought the week before, the weather forecast for next week (for example, ice cream in hot weather), and what discounts the manufacturers are giving.

This book shows how you can use similar rules and processes in your business or organization. It begins with a detailed, clear explanation of business rules and how JBoss Rules supports them.

You will then see how to install and get to grips with the essential software required to use JBoss Rules. Once you have mastered the basic tools, you will learn how to build practical and effective of the business rule systems.

The book provides clear explanations of business rule jargon. You will learn how to work with Decision Tables, **Domain-Specific** **Languages** (**DSL**)s, the Guvnor and JBoss **Integrated** **Development** **Environment** (**IDE**), workflow and much more.

By the end of the book you will know exactly how to harness the power of JBoss Rules in your business.

## What this book covers

*Chapter 1*: This chapter gives you a good platform to understand business rules and JBoss rules. We look at the problems that you might have (and why you're probably reading this book). We look at what business rule engines are, and how they evaluate business rules that appear very simple and how they become powerful when multiple rules are combined.

*Chapter 2*: This chapter explains setting up Java, setting up **Business Rule**

**Management System** (**BRMS**)/Guvnor running on the JBoss App Server, setting up Eclipse, and installing the Drools Plug-in. It also details the installation of the Drools examples for this book and the Maven to build them.

*Chapter 3*: Guvnor is the user-friendly web editor that's also powerful enough to test our rules as we write them. We take up an example to make things easier. Then we look at the various Guvnor screens, and see that it can not only write rules (using both guided and advanced editors), but that it can also organize rules and other assets in packages, and also allow us to test and deploy those packages. Finally, we write our very first business rule—the traditional 'Hello World' message announcing to everyone that we are now business rule authors.

*Chapter 4*: This chapter shows how to use the Guvnor rule editor to write some more sophisticated rules. It also shows how to get information in and out of our rules, and demonstrates how to create the fact model needed to do this. We import our new fact model into the Guvnor and then build a guided rule around it. Finally we test our rule as a way of making sure that it runs correctly.

*Preface*

*Chapter 5*: This chapter pushes the boundries of what we can do with the Guvnor rule editor, and then brings in the JBoss IDE as an even more powerful way of writing rules. We start by using variables in our rules example. Then we discuss rule attributes (such as salience) to stop our rules from making changes that cause them to fire again and again. After testing this successfully, we look at text-based rules, in both the Guvnor and the JBoss IDE, for running 'Hello World' in the new environment.

*Chapter 6*: This chapter looks again at the structure of a rule file. At the end of this chapter, we look at some more advanced rules that we can write and run in the IDE.

*Chapter 7*: This chapter explains how testing is not a standalone activity, but part of an ongoing cycle. In this chapter we see how to test our rules, not only in the Guvnor, but also using FIT for rule testing against requirements documents. This chapter also explains Unit Testing using JUnit.

*Chapter 8:* This chapterexplains how to use Excel Spreadsheets (cells and ranges) as our fact model to hold information, instead of the write-your-own-JavaBean approach we took earlier. Then we use Excel spreadsheets to hold Decision tables, to make repetitive rules easier to write.

*Chapter 9*: This chapter aims to make our rules both easier to use, and more powerful. We start with DSLs—Domain-Specific Languages. This chapter follows on from the 'easy to write rules' theme from the previous chapter and also discusses both ruleflow and workflow.. It would be great to draw a workflow diagram to see/control what (groups of) rules should fire and when. Rule flow gives us this sort of control.

*Chapter 10*: This chapter shows you how to deploy your business rules into the real world. We look at the pieces that make up an entire web application, and where rules fit into it. We see the various options to deploy rules as part of our application, and the team involved in doing so. Once they are deployed, we look at the code that would load and run the rules—both home-grown and using the standard RuleAgent. Finally we see how to combine this into a web project using the framework of your choice.

*Chapter 11*: This chapter looks at what happens under the cover by opening up the internals of the Drools rule engine to understand concepts such as truth maintenance, conflict resolution, pattern matching, and the rules agenda. In this chapter, we explore the Rete algorithm and discuss why it makes rules run faster than most comparable business logic. Finally we see the working memory audit log and the rules debug capabilities of the Drools IDE.

*Chapter 12*: This chapter deals with the other advanced Drools features that have not yet been covered. This includes Smooks to bulk load data, Complex Event Processing, and Drools solver to provide solutions where traditional techniques would take too long.

## What you need for this book

We need four pieces of software for this book. All of these are open source, can be downloaded easily from the Internet, and are available under a business-friendly license.

We need Java as this is the core computer language upon which all of the other tools are built. We need BRMS/Guvnor and JBoss App Server to provide a web-based rules editor aimed at business users. We also need to install Maven, a build tool that takes the various Java scripts (source) and transforms them into a package that we can deploy on a web server. We need Eclipse and the Drools plug-in to edit the Java files that we will use for transporting information around the system. We also need to download Drools examples for this book which are available at **http://code.google.com/p/red-piranha.**

## Who this is book for

If you are a business analyst—somebody involved with enterprise IT at a high level, who understands problems and planning solutions, rather than coding in-depth implementations—then this book is for you.

If you are a business user who needs to write rules, or a technical person who needs to support rules, this book is for you.

If you are looking for an introduction to rule engine technology, this book will satisfy your needs.

If you are a business user and want to write rules using Guvnor or the JBoss IDE, this book will be suitable for you.

This book will also suit your need if you are a business user and who wants to understand what Drools can do and how it works, but would rather leave the implementation to a developer.

## Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "We'll see that one file contains two rules:

**Hello World** and **GoodBye**."

*Preface* A block of code will be set as follows:

**public void setChocolateOnlyCustomer**

**(boolean choclateOnlyCustomer) { this.chocolateOnlyCustomer = chocolateOnlyCustomer;**

When we wish to draw your attention to a particular part of a code block, the relevant lines or items will be made bold:

**for ( int i = 0; i < rules.length; i++ ) {**

**String ruleFile = rules[i]; log.info( "Loading file: " + ruleFile );**

Any command-line input and output is written as follows:

**cd C:\projects\drools-book-examples**

**New terms** and **important words** are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes for example, appear in our text like this: "When Eclipse opens (and you've selected the workspace), select **File | New Project** from the menu".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

## Reader Feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an email to **feedback@packtpub.com**, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on **www.packtpub.com** or email **suggest@packtpub.com**.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on **www.packtpub.com/authors**.

## Customer Support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

## Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in text or code—we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help to improve subsequent versions of this book. If you find any errata, report them by visiting **http://www.packtpub. com/support**, selecting your book, clicking on the **let us know** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata added to the list of existing errata. The existing errata can be viewed by selecting your title from **http://www.packtpub.com/support**.

## Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide the location address or website name immediately so we can pursue a remedy.

Please contact us at **copyright@packtpub.com** with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

## Questions

You can contact us at **questions@packtpub.com** if you are having a problem with some aspect of the book, and we will do our best to address it.