# Drooling over JBoss Rules

My grandfather was a docker in Belfast who loaded and unloaded ships. My father was an accountant who tracked and valued the items as they moved around the world. My job is to replace both of them with automated systems, and apply a complex set of business rules that determine what gets loaded first and how much it is worth. For items such as fresh produce, the answer is complex and often changes by the hour.

If you look at your family tree, you'll probably see a similar progression. We've gone from lifting bags of coal to being workers who move knowledge around. We may not have blisters on our hands, but we do get sore heads from the business (medical, legal, financial and similar) problems that we deal with.

How do we know if we're doing a good job? For our grandfather's generation, it was easy to see people and companies who moved the most 'stuff' around. For our generation things are different—We're moving knowledge that we can't see. We often need to apply unclear business rules to our job. To be successful we need to do our job quicker, faster, and better than anybody else.

JBoss Rules can help us become better and faster at managing our knowledge. To explain how, this chapter looks at:

* Who you are and what your problem is
* Life or death business rules
* Business rules in your organization
* Why existing solutions don't cut it
* How rule engines can come to the rescue
* An introduction to JBoss and JBoss Rules
* The bigger picture and parts of the solution
* How to write the rules
* When not to use a rule engine

**Who are you? What's your problem?**

If you are reading this, you probably know important information. You might be in the medical, legal, or accountancy professions. You may be the only person in the company who understands how to process refunds to tractor dealers in the Chicago area. You may be the most experienced underwriter in the mortgage application department. Or you may be the person who is most capable of talking to 'those guys in IT'. Perhaps you've taken part in one or more Business and IT projects, or maybe this is your first one.

Whoever you are, you've got a problem. Maybe your team is too busy for the workload it has, maybe you can't recruit enough people to work for you, or maybe you can get the people, but it takes too many resources to train them. Whatever the cause, there are not enough minds to go around, and costly or embarrassing mistakes happen as a result.

You've probably already joked about being able to clone your key people. Wouldn't it be great to leave your clone working at the desk while you get some time on the beach; or even just get to go home on time? Although JBoss Rules does not allow you to clone yourself, it does allow you to clone your mind that is, put your knowledge into a computer. Once in the computer, this knowledge can be copied, reviewed, and kept working even after you go home.

Your second thought after hearing the "put your knowledge into a computer" bit is probably, "If the computer knows what I know, will I be out of a job?". Maybe. Or more likely, you'll no longer do the routine 80% of your job that you hate—the rubbish that fills up your day. It means that you will spend more of your time doing the 20% that you enjoy, including talking to people, meeting customers, improving the process, planning your next golf (sorry, "business networking") trip, or whatever you find interesting.

This book is aimed at non-technical users, although it contains a lot of information for people who want to get under the covers of JBoss Rules. Don't worry even if the entire extent of your PC skills is limited to writing a couple of formulae in Excel. You're going to be OK.

**Does this sound like where you work?**

Everybody complains about his or her job from time to time. You probably have a mug saying "You don't have to be crazy to work here…but it helps". Just try out our 10-question pop quiz and see if it sounds somewhat like the place where you spend most of you working hours:

1. Is Bob, in the corner, the only person who knows how the system really works? Can the business scale only if we have an expert? Is critical knowledge lost when people like Bob leave?
2. If you're Bob (owning the knowledge), are you sick of people asking you stupid questions? Do you think: don't these people know that you've got a job to do?
3. Are your customers getting a different answer every time they call your company (and getting more than slightly irate about it)? Are you at risk of receiving a slap on the wrist (or worse, a fine) from a regulator or other standards body?
4. Do you find yourself working around, rather than with, your computer systems? Have you ever thought of pouring coffee into your computer keyboard in frustration? (Trust me, it doesn't help.)
5. Are things always done by the book, or is there a lot of informal knowledge that is just in people's heads?
6. Did you prepare for a quality (ISO 9001) audit and then leave the process documentation unused on a shelf? Is there anybody around who knows or wants to change this process? Is this the right balance between being too hard to change (and being stuck in a rut) and being too easy (resulting in chaos)? If a change is made, will people know about it and will they take any notice?
7. Does your business knowledge exist in some easily usable format? Is its format easy to update? Can everybody use it from one central location (so that copies are not 'out of sync')? Can you track changes made and roll them back if you get it wrong?
8. Do the right people (and only the right people) have access (both read and update) to this information? Does this access need to change depending on the context of what the user is doing at the time?
9. Do people in your organization work on projects? Do they come together to form goal-driven teams and then go back to their original jobs when the objectives have been achieved? Do you know how to document the outcome of these projects as rules so that they can be reused?
10. Are tasks carried out in isolation? How do we ensure that tasks and team members collaborate effectively? In the old days, everything was done in-house. Now the 'office as a factory' must also seamlessly interlink with other suppliers plugged in as part of the process.

If some of these problems seem familiar, then maybe, just maybe, business rules and JBoss Rules can help. But if you think you've got problems, consider the following example.

## Life or death business rules

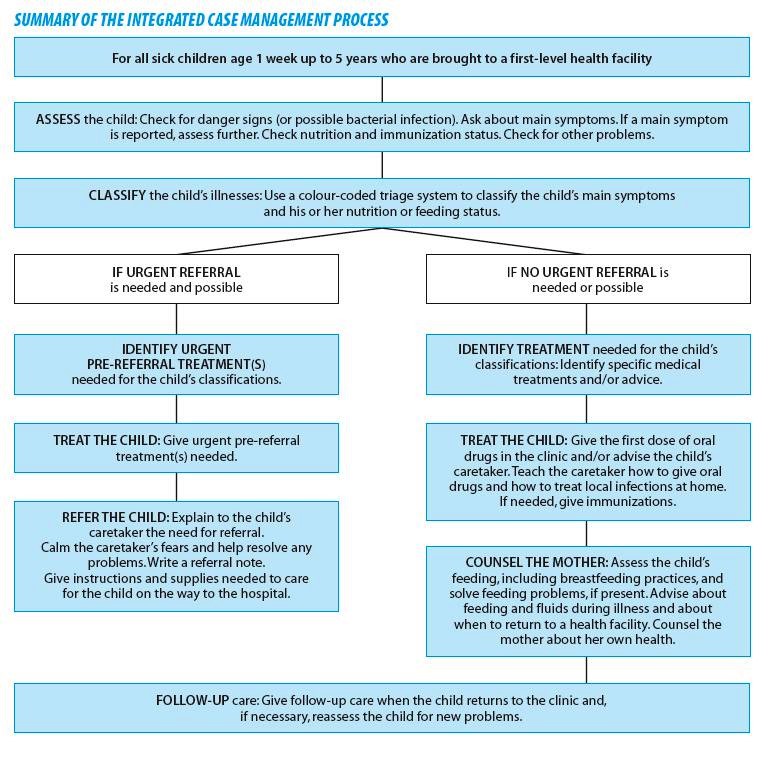
The health services in Bangladesh (like many elsewhere) can't get enough doctors. Training more doctors is not an answer. Those who do qualify tend to leave for higher rates of pay elsewhere. So, given the desperate need for trained medical staff in rural areas (for example, to curb child mortality rates), what are the health workers to do?

The more qualified a doctor is, the more likely he is to take a flight. The district hospital in Matlab, Bangladesh, boasts an operating table, lamp, oxygen cylinder, and anesthetic machine, all carrying the EU's gift tag. They gleam, partly because they are unused. Several surgeons and anesthetists have been trained, but none so far have been retained. "Other than holding a gun to their head, doctors do not stay here.", comments Shams Arifeen, a researcher in the **International** **Centre** for **Diarrhoeal** **Disease** **Research**, **Bangladesh** (**ICDDR**, **B**). Doubling their pay is not the answer because they can earn five or ten times as much in private practice. Besides, specialists want to educate their children in the capital Dhaka, not in Bangladesh's backwaters.

**What would you do?**

Imagine that you were standing in that clinic without medical training, when a mother asks you to look at her sick child. What will you do? The solution that the Bangladesh health workers came up with was **IMCI** or **Integrated Management of Childhood Illness**. IMCI takes the knowledge in a doctor's head and writes it down as a set of rules that health workers can follow. When a sick child is brought into the remote clinic, the health worker is able to follow the simple step-by-step instructions to make quite a sophisticated diagnosis.

The following figure shows IMCI:



Look at the boxes in the above diagram—it's a set of medical rules. (Source, World Health Organisation, **http://whqlibdoc.who.int/ publications/2005/9241546441.pdf**.) Using these rules a health care worker can quickly come to the following conclusions:

* When the child is under two years of age, then refer to doctor immediately
* When the child's skin does not bounce back when pinched, then the child is dehydrated
* When the child is dehydrated then give rehydration salts

This is a real life example of business rules. Although it is paper-based and useful to the medical profession, it's a good example of business rules all the same.

* Rules are 'when something is present, then do this'. And not just single rules, but many of them. Together, loads of simple rules allow you to come up with quite a sophisticated diagnosis.
* Ruleflow and Workflow allow you to group your rules and decide which should fire first. If you're a health worker with a sick child, you want to do the most important checks first. Depending on the outcome, you then apply the next set of medical rules.

Everybody, including the doctors, is happy that his or her knowledge has been translated into rules. The doctors are happy because they can (guiltlessly) move to better-paying jobs. The medical workers using the system are happy because they can help the sick children that they see every day. The children gain because the better availability of medical knowledge is literally the difference between life and death.

If you've used a computer language before, you might find the above example strange. 'Traditional' computer languages are more like a set of instructions: Do step 1, do step 2, repeat 5 times, and so on. Rules are different; they allow you to make many individual statements of what you know to be true and then let the computer decide if these rules apply

(or not) to the current situation. This is similar to the way in which the human mind works.

Look again at the example. We don't specify any order for our rules.

All of them, one of them, or none of them might apply in a given situation. A child under two who was dehydrated potentially could be referred to a doctor and given rehydration salts on the way in. This could be the outcome we want, or we may wish to rewrite our rules to be more precise—but always in the "when (something is true) then (do this)" format.

For more information on this real life example, read *The Economist* magazine's online article at: **http://www.economist.com/research/articlesBySubject/ PrinterFriendly.cfm?story\_id=9440765**.

## Business rules in your organization

You're going to be hearing a lot about 'business rules' over the next couple of pages, so it might be helpful to clarify what they are. We use the term 'business rule' to show that rules are non-technical. They could also be called 'medical rules', 'financial rules', 'insurance rules', 'benefit payment rules', and so forth. It all depends on the organization you work for, and the particular niche that it finds itself in.

A business rule is any bit of knowledge that can be expressed in the following format:

**When** 'something' is true, **Then** do 'this'.

Er, that's it. Nothing more complicated than that.

You do have knowledge like that in your organization, don't you? All companies and organizations have business rules, even if they are implied (that's, unwritten) or buried (as code) in existing systems (for example the ones with black screens and green text that you see in Hollywood movies).

Examples of these rules are:

* When a football team wins a game, jump up and down and shout loudly • When a staff member gets promoted, give them a pay rise of 10%
* When a person's salary is less than 30,000 dollars, apply a tax rate of 20%
* When somebody leaves the office before 4 pm, make sarcastic comment about 'taking a half-day vacation'

This book uses the "when...then" format for business rules. In practical terms, these are very similar to "if...then" statements you may have seen in any computer language.

The key reason for using 'when' is to underline that business rules will 'fire' whenever the condition is true. Traditional 'if' statements will only fire if we happen to be at that step in the process at the time.

Business rules themselves tend to be simple. Their power comes from the fact that there are many of them (tens, hundreds, or even thousands). Just as you have many rules in your head (when you see a bear, run away), the trick is knowing when to apply them (what happens when you see a bear in the zoo?). Later we'll look at writing rules clearly and testing them to ensure that they do what you want.

Business rules should be written as clearly as possible (in English, or your human language of choice). While this makes your life easier when writing the rules, more importantly, it allows other people to review your rules in the future. Various estimates are that 95% of all work on a computer system is in this 'review and update' phase long after the original team has left. So clarity is one of the biggest advantages of using rule engines.

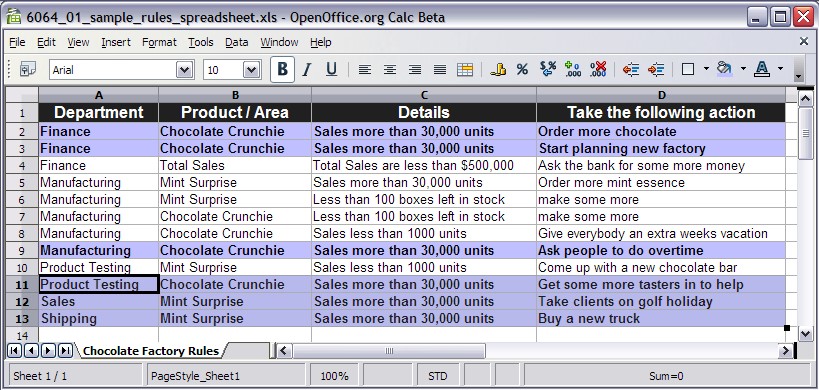
### Exercise — rules in your organization

As a simple exercise, take 10 minutes to list some of the business rules in your organization. Don't worry if they are simple or difficult. Just write them out in the "when…then" format.

Here's one I did earlier.

### The chocolate factory

The following figure shows sample business rules for a chocolate factory:



Since I was a child, I've always wanted to work in a chocolate factory, and here's my chance. The figure shows the sample rules that I came up with. The first three columns are the 'when' part of our business rule or the Left-Hand Side (LHS). The last (fourth column) is the 'then' part, also known as the Right-Hand Side (RHS).

For example, the first record in the table says:

When the **Finance** department sees that we've sold more than **30,000** **Chocolate** **Crunchie** bars, then they should **Order more chocolate**.

You'll notice that there are a couple of departments other than **Finance** (**Manufacturing**, **Product Testing**, **Sales**, and **Shipping**). Although I made these rules up, often the rules that get deployed are a combination of rules from various teams.

When I'm reading books such as this, I wonder how I take this concept or theory and actually use these in a 'real system'. Well, I don't want to spoil the surprise, but the actual business rules we will be writing and deploying are very similar to the 'clear English' examples discussed earlier.

There's a lot of clever stuff happening under the covers to keep our business rules simple, but that's how it should be ,with the machines, rather than people, doing the hardwork.

### Build your own rule engine in Excel

You are probably wondering why some of the rules in the figure are highlighted. These are the rules that will 'fire' whenever we sell more than **30,000 Chocolate Crunchie** bars. You could imagine another set of rules highlighting whenever we have less than 100 boxes of **Mint Surprise** left in stock.

That's all a rule engine does. It selects the rules that are correct for the current situation and then carries out whatever they say. If you're good at Excel, you could probably mimic this behavior using auto filters or conditional formatting so that the colors would change automatically.

Sometimes things can be as simple as they seem. If you're coming from a business department, it makes sense that all the rules should be applied, often all at once. Why shouldn't the manufacturing and finance people carry out their actions simultaneously?

Technical people (like me, for the first couple of months) might miss this point. We're used to telling computers to do things one at a time and have to 'unlearn' our years of experience that says that computers must do things exactly one step at a time.

We did say that this book is for everybody. Items like this just help level the playing field.

Speaking of technical people…

**Why can't the tech guys write the rules for me?**

If you're a good business manager, you've probably been taught to delegate. Until now, for anything technical (such as computer systems) you've probably been delegating to those 'techie guys'. So why don't the tech guys write the business rules for you?

The answer is that the tech guys can write the rules on your behalf, but it's a bit like booking a flight through a travel agent rather than over the Internet. Sometimes it's a much better idea to do it yourself.

* Have you ever turned up at the airport and found that the travel agent got it wrong? Doing it yourself means that there is one less link in the chain that can go wrong. Booking your own flight (and writing your own rules) is quicker and easier.
* Have you understood what all the hieroglyphic codes on the paper ticket meant? (I'm showing my age—most airlines phased out paper tickets years ago). The chances are that if you give a technical person the rules to write, he or she is probably going to do it in a computer language such as C#, Java, or VB. There is nothing wrong with that; it's just that they might as well write it in Egyptian hieroglyphics for all that you will able to understand it—there is no way you will be able to check if they got it right. Business rules solve this problem.

For simple flights (for example, Dublin-London return), booking online (that is, doing it yourself) is fine. For multi-stop round-the-world tickets, getting advice from a travel agent is often a good idea. Likewise for rules: write most of the simple ones yourself and then get some help with writing the complex ones.

## Why existing solutions don't cut it

Computers have been around for a long time and we're not the first people to use them to solve these kinds of problems for business people. In general, these business systems do three things:

1. Capture information, for example, via a web interface (presentation layer).
2. Apply business knowledge to this information (business layer).
3. Store or forward this information (service or data layer).

It is the business layer that we are most concerned with. The presentation and service layers, while not trivial, are known problems that lend themselves to some degree of standardization. In contrast, the business layer will be unique to each organization, as it reflects the processes and knowledge of the organization.

In some ways, the business layer is the 'learned memory of the organization'. Despite (or perhaps because of) years of implementing **EIS** (**Executive** **Information** **Systems**), many of them suffer from the following problems:

* All three layers tend to be tightly interlinked, so it is not easy to extract the business logic and use it elsewhere.
* Business knowledge and rules are often hidden in code. This is difficult to audit and can lead to discrepancies between the documentation and the actual implementation.
* It is hard for the domain experts (the guys with the business knowledge) and the technical experts to collaborate as they (literally) speak different languages.
* The business layer can be difficult to update, both in implementation and for fear of undesirable side effects.

Although theory states that these functions should be separated, the fact that the business tier is often expressed in a programming language like Java means that other functions (for example, database access) often creep in over time. Even worse, there is no clearly delineated place to put the business logic, which is why it can become scattered throughout the system, making it hard to reuse.

Given that we've had these problems for many years, how can we do any better?

## Rule engines to the rescue

A rule engine can solve these problems—at least to some extent. Instead of having technical, spaghetti code, it allows us to keep our business rules as simple as possible, just like the examples we saw earlier. A rule engine allows us to 'run' these business rules into the rest of our bigger computer system so that we can get our values from a web page, save the results into a database, or anything else we need to do with our information. At the same time, our business rules stay in a 'clear English' format so that we are able to review them later.

So, what is a **rule** **engine**? Very simply, it is a place in which we can evaluate our business rules. Without it, our rules would be stuck 'on paper' and we'd have no way of feeding them into our system.

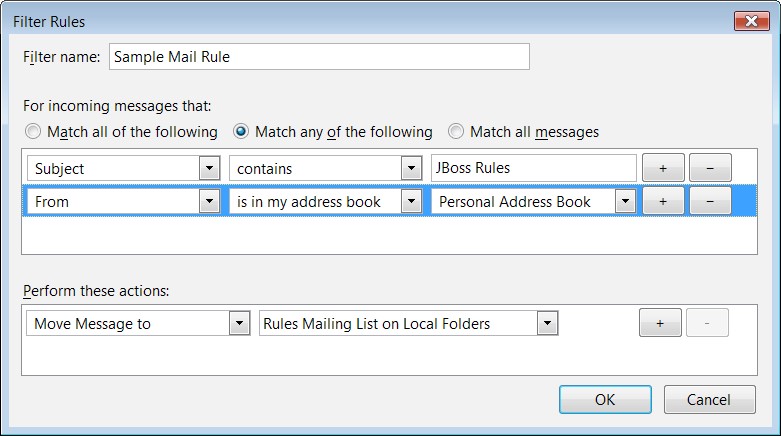
Here are a couple of points that explain why rule engines are better than most computer systems:

* Rule engines allow you to say "what to do", not "how to do it". This means that you can focus on what you know to be true, and allow the machine do the heavy lifting of figuring out the consequences of all of these combined truths'.
* Logic and data separation: You probably already have a database to store information. It's a good place for data, but a bad home for your business rules. Having a rule engine gives your rules a natural home where you can manage your (entire) business knowledge properly.
* Speed and scalability: The way rule engines work (based on the Rete algorithm, if you're interested) has been mathematically proven to be faster and more scalable than most traditional handcoded 'if…then' solutions.
* Powerful tools: For developers, as well as for business analysts, tools provide easy ways to edit and manage rules. More importantly, they give immediate feedback—no more slogging through 10 web screens to reproduce a 'bug' in the business logic.
* Auditing: Rule systems provide an explanation facility allowing you to audit how and why a particular decision was made.

### Other rules (Microsoft Outlook)

If you're a power user of Microsoft Outlook, then you probably have mail filters set up that say something like:

* **When** a mail comes in that looks like spam, **then** put it into the trash can
* **When** a mail comes in saying 'Jboss Rules', **then** put it in the folder marked 'rules'



The figure shows business rules hiding in your mailbox. In this figure can you recognize the 'when...then' format? You've already been using a rule engine without even knowing it! But the rule engine in Outlook is limited to email sorting and we need something more powerful to meet our business problem. Enter JBoss Rules.

## Meet JBoss Rules

Your boss or somebody from the IT department or a consultant has mentioned Drools as part of the solution. After having a good laugh at the name (it's a long story) you want to find out more. We'll look at this in two parts—Who is JBoss, and what is the Drools/JBoss Rules team.

JBoss is a division of Red Hat (NYSE:RHT). This means that Drools is backed by an industry-leading company. The support from this company is available whenever you need it.

Even better, a key part of JBoss and Drools is open source. To put this in quality terms, both the JBoss and Drools teams are confident enough about their product to let you poke inside it. It's a bit like getting a tour of the Mercedes car factory.

Open source also shows the confidence that the team has in the quality of the product and of their support, If you don't think the support is good enough, you are free (and able!) to get third parties to do the job to your satisfaction. Because the bulk of how JBoss/Red Hat makes their money is service related, they're pretty confident that that option won't be needed.

You may be confused by the naming of the project. Is it **Drools** or **JBoss Rules**? Officially it is now the latter, although it started out as 'Drools' and the name is still in common usage. This book tends to use 'Drools' as it is shorter to type and read. Both terms refer to the same thing—the business rules product from Red Hat and JBoss.

Drools is an advanced rule engine (and a lot more besides, as we shall see later). It allows you to state things that you know to be correct (for example, if the expenses claim is above $5000, then a senior manager needs to sign it off). As somebody who has knowledge of business rules, you'll be able to feed the rule engine with what you know.

### A bit more on open source

A few years ago, if you searched for the words 'open source business' on the internet, you would have found people describing it as a little bit 'hippy', or maybe old style communists resurfacing in another form. Those critics (including Microsoft—for example,  **http://port25.technet.com/**) have now happily embraced open source as a part of their business model.

Imagine buying a car with the bonnet welded shut so that only the car's manufacturers could service it. Would you be happy with that? (Audi almost did this with their smaller A2 model). Most closed source traditional software are like that—you are at the mercy of the one supplier for bug fixes and improvements. What happens if that supplier goes bust?

Now, I know next to nothing about car engineering, but I still find it comforting that I could choose almost any mechanic to fix my car. Likewise, with open source software you're unlikely to change the software yourself, but it's comforting to know that you could hire somebody to do it for you if required.

All of JBoss Rules is available as open source under the Apache open source license. That's Apache, the web server that powers most of the web sites you read every day.

The Apache license is particularly business-friendly, and you can take the code and use it in pretty much any way you want, as long as you acknowledge that your product was 'built using Drools'. You don't have to publish your changes or additions (as another famous open source license, the GPL, requires you to do). Nor do you have to pay any license fee for using their product, even as part of a commercial deployment.

Of course, you'd want to confirm the exact details with your lawyer. But the chances are that he or she will tell you the same thing and charge you a lot more for doing so.

The next question is where do Red Hat and JBoss make their money if they're not selling a product? The answer is through a combination of training and consulting services, as well as selling cross-tested 'stable' versions guaranteed to work with most standard server configurations.

All of the software we use in the book is available for free from the JBoss community web sites. And it's also available as an enterprise product with full Red Hat support, if that's important to your organization.

### The JBoss Rules community

As JBoss develops its rules' code 'in the open', it's easy to get in touch with the developers to get help.

#### Where to get help

Check around the sites listed as follows before firing off your 'please help me' email.

* The **Product** **home** **page** is the official home page, tailored more to a business audience. If you're trying to sell a Drools **BRMS** (**Business** **Rules** **Management** **System**) to your boss, this is the place to go.

**http://www.jboss.com/products/rules**

* The **Community home page** is a slightly more detailed resource. This provides links to a lot of useful resources, including the Drools technical documentation. The information on this page tends to be more 'bleeding edge', including stuff that may not yet have made it into the official enterprise versions.

**http://www.jboss.org/drools/**

* The **Wiki** is a much more rough-and-ready resource. It has guides of varying quality, dealing with specific issues (for example, deploying the rules engine on non-JBoss web/application servers). Wikis are writable as well as readable, so if you're doing something that doesn't appear to be documented here, think about adding it. The chances are that the solution is technical and generic enough to be sharable.

**http://wiki.jboss.org/wiki/JBossRules**

* The **mailing lists** are where you can see previous questions asked by Drools users and developers. This is where you can ask for help. But read the next section on how to ask for help, or I can guarantee that your pleas for assistance will go unanswered.

**http://www.jboss.org/drools/lists.html**

* The **Bugs and feature requests** page shows you what the Drools development team is currently working on. Yes, when we said the project was open, we meant it. You may get far too much information, but better that than too little. If you feel something is missing from the current version, checking here might show that it's on its way. And if you talk to the guys on the mailing lists (they really appreciate end-user feedback), you might be able to persuade them to add your feature here. **http://jira.jboss.com/jira/browse/JBRULES**

If there is a bug that you need fixing or feature that you need implementing as soon as possible, one way of getting it done quickly might be to offer to sponsor development. That is, pay for the JBoss Rules team to add it on your behalf.

Often this works best if the feature is already on Drools team's 'todo' list, but may be 18 months from development due to other priorities. Although the feature will be open sourced, it will get built faster and better than any other alternative—these guys know the product inside out. Once inside the product, it will continue to work in future releases and maybe get further improvements 'for free'! It's an effective, if non-traditional way, of getting you what you need.

#### How to ask for help

If you got locked out of your house, how would you ask your neighbor for help? Would you be arrogant, and demand the he/she helps you straight away (ignoring that he might be doing something important themselves), or would you ask nicely, explaining your problem, what you've done to try and sort it out, and then ask him or her if they can help you? Which approach is most likely to succeed?

Asking for help in an open source project is somewhat similar. Any open source team is busy—the core guys also have bosses and deadlines to meet. So if you've got a problem with Drools, you can increase the chances of getting help from the JBoss Rules guys by doing the following:

1. To start out, assume that the problem is due to a mistake that you've made. I'd consider myself experienced with computers, but you'd be amazed at some of the 'duh!' errors that I still make. Check spellings. Check the instructions. Check that you're connected to the network. Then check again.
2. Read the manual or search the Web. Then read it again. Unless you're pushing the boundaries of what Drools does, the chances are that somebody has seen this problem before. Google is great for this. Put in the error message that you're getting and you'll get back plenty suggestions of areas to investigate.
3. When you search the Web/mailing lists, look at problems that are similar but not exactly the same as your own. Often, the solution will be similar (if not exactly the same).
4. Ask a colleague for a sanity check, even if he or she may not be familiar with the product. Two pairs of eyes are better than one. Often, while you're walking through the sample, you'll see the basic mistake you've made.

Have you done all of this once, twice, thrice? Now you're ready to ask a question from the mailing lists. If your question is clear, has enough (relevant) detail, and you have put in lot of effort to solve the challenging problem yourself, the greater are your chances of getting a quick reply. Before you type your email, read the classic article *How To Ask Questions The Smart Way* at **http://catb.org/~esr/faqs/ smart-questions.html**

This section might make the Drools team appear unfriendly, but they're not. They're very approachable and down-to-earth guys. You also have a direct line to them, unlike most commercial software projects. But, like all open source projects, they are asked a lot of lazy I-can't-be-botheredto-read-the-manual type of questions. So spend an extra 10 minutes to compose your email and you'll be rewarded with support worth hundreds of dollars. If you want to know what really irks the Drools guys, read this blog post: **http://blog.athico.com/2007/11/ drools-user-mailing-list-growth.html**.

After all that preparation, send your email to the user mailing list at **http://www.jboss.org/drools/lists.html**. Then wait. Do not re-send it. Remember that Drools is an open source project and you may never get your question answered, or get it answered only after a couple of days' delay. If you need guaranteed support, consider buying a subscription from Red Hat.

Don't be surprised that the answer, if and when you get one, is along the lines of 'have you considered trying X, Y, or Z?' Don't expect a complete solution, but just good suggestions as to areas that you can try to resolve the problem.

When you do find the solution, post the answer to the mailing lists. Keep it technical, with nothing confidential to your organization. Drools users who will follow in your footsteps will be eternally grateful. It will also earn you major kudos with the team, which will benefit you when you ask another question in the future.

## The bigger picture

You're unlikely to go through the trouble of putting your knowledge into a rules system and leave it at that. You've a problem that you're trying to solve. For that, you're going to use rules as part of a bigger system.

Here's the five-minute guide to almost any computer system. It takes information from users (these days, mainly via a web page), does something with it, and then stores it somewhere (normally in a database). You may recognize some database brand names such as Oracle, SQL Server, or MySql. Think of the database as a very big version of Excel. Sometimes the flow of information goes the other way—access information in the database, and then show it on the web page. That's it! So what are you paying all these IT consultants for?

Drools helps you with the middle 'do something with the information' bit. Here you apply the business knowledge (the stuff that's currently in your head) to the information that is passing through.

We recommend Drools, as one of the other options is to put your brain into a glass jar (think of a mad scientist lab with rows of brains suspended in bubbling liquid) and somehow wire it in to the system. Drools is a much less painful option.

### Members of your team

Unless you're a business user by day and techie by night, we don't expect you to build the entire web system by yourself. In general, as a business user, you'll supply two bits of information to the IT team. The rest should be considered 'plumbing'—stuff that should be done according to industry standards and best practices, but that otherwise will be hidden from you (the user) and should 'just work' (like water coming out of the tap).

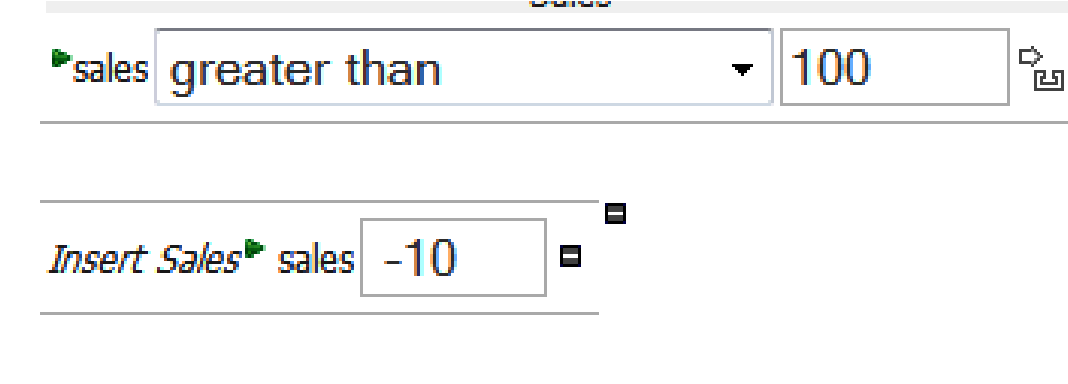
The two sorts of information you'll generally need to provide are:

* The user's interactions with the completed system. For example, the web page that the user uses to log in, the first screen they see after they log in, and what the various buttons on this screen do. Entire books have been written on this subject and so we won't get into those details.
* The actual business rules. Unlike the screens, this is 'behind the scenes' stuff. This is your knowledge applied to the data that's being captured on the web pages. Even if you don't use Drools or any another rule engine, you'll still need to do this step. Otherwise, how will the system know to pay for prescriptions for Viagra,and not for aspirin? (or whatever your business rules actually are).

## How do I write the rules

So, you want to get right in and start wiring up the rules. You've got four choices of editors for rule-writing:

* You can use the **Business** **Rules** **Management** **System** (**BRMS**) from Drools, which is called **Guvnor**. This is a web-based application that's aimed at people like you. Not only is it easy to use, but it can be set up once for the entire team to use via Internet Explorer, Firefox, or your favorite web browser. In general, this is the editor that we recommend, unless you need a feature that is only available in one of the other editors. The following screenshot gives an idea of what a business rule looks like in Guvnor— there is more information on this in Chapter 4.



* You could write the rules via a simple text editor such as Notepad. This a bit masochistic and dull, staring at black and white text with no help as to what is expected. We mention it here only to show that there is nothing special about the rules format; it's just a plain text file.
* You can write rules in Microsoft Excel or any spreadsheet that can output Excel-like spreadsheets (for example, Sun's Open Office). You have to follow a certain template (it's not that difficult once you see it). The Excel format lends itself to rules that repeat themselves a lot (the sample Drools for Decision Tables has lots of different categories of car insurance claims).
* Use the Drools IDE, which is based on Eclipse. **IDE** stands for **Integrated**

**Development** **Environment**, so Eclipse is a bit like 'Microsoft Office for Techies'. The chances are that your technical team is using it anyway (to write in a language called Java, although it can be used with other computer languages). The Drools IDE bit adds plug-ins to Eclipse to allow rule editing and debugging.

The IDE is more powerful, but also more complex. We'll talk about its extra features later, but most of the commonly used ones are already in the BRMS (and over time, the remainder will be implemented). It's possible to easily switch between IDE and BRMS.

Whichever way you choose, the rules that get fed into the rule engine are pretty much the same. In fact, the BRMS allows you to import and manage rules written as text/Decision Tables via the IDE. So, for now, following the BRMS is a good choice.

## Introducing the BRMS (Guvnor)

The BRMS is a web page that you open in Internet Explorer, Firefox, or your favorite Internet browser. You've seen web pages before, right? The BRMS allows you to enter your knowledge as business rules via a web page.

BRMS or Guvnor? The web-based rule editor that we will talk about in this section started out as the Business Rules Management System or BRMS. Unfortunately, other rule engine vendors use the term BRMS to refer to something completely different (not just the editor, but the core rule engine as well). Hence the renaming of the Drools BRMS to Guvnor, which also reflects that this web-based application can also be used to manage other things such as deployment, testing, and processes.

At the start you can enter rules via the guided editor (a similar idea to the helpful 'wizards' that you might have come across in Windows). Later, as you get more used to the rules syntax, you might want to edit the rules directly in the text editor.

There are a couple of other things that the BRMS gives you over and above basic 'rules editing', such as:

* Team editing
* Version management of rules and related assets
* Asset management
* A deployment mechanism
* Security (Login)
* Import and export of data

## Parts of the solution

By now you should have understood the basic concept that a rule engine allows you to capture your knowledge and integrate it into an enterprise web system. However, a rule engine isn't just a black box. There are a couple of parts to it that are useful to know about. (I don't know much about car mechanics, but I can check the oil and tyre pressure. We'll keep the Drools technical bits at that level.)

### Rules editor

This is the choice of BRMS, IDE, Decision Table, or plain text file. All produce a similar underlying rule language. The mechanism for deploying these rules (RuleAgent or some other equivalent) is similar.

### Rules compiler

Something needs to translate the near-English rules language into something the rules engine can understand—this is what the compiler does. Your main awareness of the compiler (as a BRMS user) is when it complains that it does not understand the way that you are phrasing your rules.

### Runtime

As the information flows through your system, something has to be applied to the (compiled) rules. This is where the Drools runtime comes in. In general, you don't worry about the technical aspects of this. You just care that there is something applying the business rules that you have written in the live/production system.

### Fact model

So we have a working system with information flowing from the Web, modified by the rules, and then saved in the database. Obviously, when writing our rules, we need to know the form that this information will be in. (Will we ask the user for salary before or after tax? Will we ask the user what country he or she lives in or just the post/zip code?) The information has to be in a certain format. (Think of an Excel spreadsheet. We need to know which column the salary information is stored in, and if this is before or after tax.) The description of the information we need and the format it is stored in is known as the fact model.

### Java

Rather than writing the fact model in Excel, it's mostly written in Java. Don't worry, at the level we're working (specifying the names of the information that we're collecting and if it's a number, piece of text, and so on), it's not that complicated. Remember that if you can handle Excel, you can do this. We've two approaches to building the fact model:

* For most of this guide, we'll assume that somebody else has done the analysis and that all the information you need when writing the rules will magically be there. Realistically, you're going to find things that are missing when you start writing your rules. The Drools technical guide has more information on how to build the fact model using Java.
* It's probably not beyond your ability to modify the fact model (just follow the recipe even if you don't fully understand the low-level details). The main reason you won't update it is that other parts of the system, such as the web screens and the database, also use the fact model as it's a key part of how the system is linked together. So, change a bit here without talking to the other guys and you risk breaking things for them.

An important note is that the BRMS helps you edit the rules, and typically does not form a part of the production system that the end users will see. That task is left to the core rules engine.

### Rule repository

Rules are important and you're going to spend a substantial amount of time writing them. While you can store them on your PC's hard disk, can you guarantee that they're not going to get corrupted? If you store them on a shared network drive (with backups) how to you manage the different versions (for example, you want to see the business rules as of July 4th last year)? How do you allow collaborative editing and track changes made by different people?

A rule repository solves these problems for us. Luckily, there is one built into the BRMS/Guvnor. But we've a couple of other options (for example, Subversion) should be wish to tie into the rest of the system.

### Rest of the system

Remember that a rule engine will not run in isolation, but be embedded in a wider system. What the 'rest of the system' will be will depend on your project, but the rules will pass data back and forward to it by means of the fact classes.

## When not to use a rule engine

This may seem strange for a book about (JBoss) business rules, but there are times when you should not use a rule engine, even if it initially appears to be a good idea. A couple of things you should consider before using a rule engine, are:

* Don't use a rule engine if your application doesn't have much complexity.

A lot of applications are just web pages that save information in a database. Even if there are a couple of checks for business logic, is there enough to justify the complexity of a rule engine? However, applications tend to increase in complexity over time. So keep this in mind when you're making your decision.

* Don't use a rule engine for the first time on a project that has strict deadlines or is high-profile. Like all new technologies (to your organization), either prototype the solution or gain the skills on a smaller project first.
* Don't use a rule engine when it's the wrong technology. What you may be looking for is workflow, or doing things in a strict sequence. Or you may just need a web page management solution such as Spring Webflow.

There are many places where you can use a rule engine. This is especially true when:

* The business logic changes often
* There are people who understand the business problem in great detail, but may not have the technical IT skills
* The problem may be too fiddly or complex for other solutions
* You need an agile solution—rule engines allow you to easily change the business logic in an iterative manner

## Summary

This chapter has given us a good platform for understanding business rules and JBoss Rules. We saw the problems that you might have. We looked at what business rule engines are, and how they can evaluate business rules that appear very simple, yet when multiple rules are combined are extremely powerful.

In the next chapter, we'll use this platform to dive into hands-on business rules. We'll start with learning more about the Business Rules Management System (Guvnor).