# Meet the Guvnor

By now, you're probably keen to see if rule engines can live up to the hype. In the previous chapter, we set up all of the tools that we need. Now we're going to dive right in and write our first business rule.

Although we have a couple of choices of business rule editor, we will start writing our rules using the Guvnor editor (formerly known as the BRMS). This is a userfriendly web editor that's powerful enough to test our rules as we write them. Along the way, we'll explain some of these concepts:

* A quick tour of Guvnor
* Loading the samples
* Our first business rule—Hello world

## Taking a tour with the Guvnor

If you've ever been in London, most of the taxi drivers will call you Guv'nor. We'll avoid all play with words (and cockney rhyming slang) about 'taking a tour' and 'taxis' in this chapter, and just get on and see what the Guvnor screens can do.

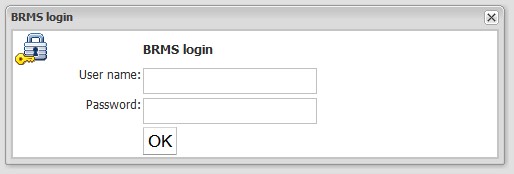
### Getting started

When we set up Guvnor in the previous chapter, we tested it by going to a web address similar to:

**http://someServerName:8080/drools-guvnor http://localhost:8080/drools-guvnor**

Open up the address in a web browser. The following screenshot uses Internet

Explorer (even if it only shows the web page), but Guvnor will also work with Firefox, Safari, Opera, and most other browsers. You will see a screen similar to the following one:



The screenshot shows the Guvnor login screen. By default, any username and password will be accepted, unless the version has been configured with extra security (for example, to use your Windows account details-ask whoever did the setup). Click on **OK**, and you'll be shown the welcome screen.

### General navigation

The first screen you will see is the search screen. This screen, like most screens in Guvnor, has the following components:

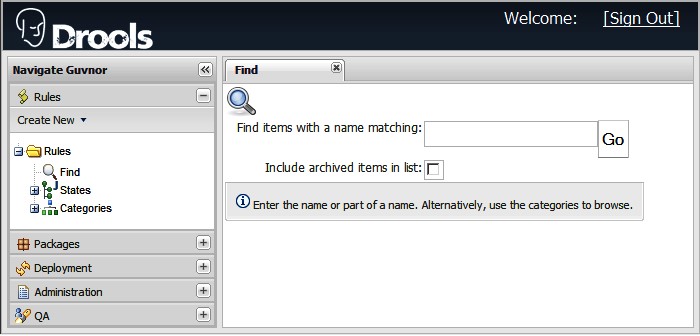
* The Drools logo on the upper-left of the screen.
* Details of who you are logged in as on the upper-right (this will be blank if you logged in using a blank username). This area also gives you the option to log out.
* The Guvnor Navigation sidebar (**Navigate Guvnor**) allows you to access all of the Guvnor functionality. In the search screen **Rules** | **Find** portion is shown.
* On the main, rightmost part of the screen is the functionality that we happen to have open (in this case the search screen). This part of the screen will change depending on what we are doing.

Navigating in Guvnor is fairly intuitive. Just click on the links (that's anything with a '+' sign next to it) on the lefthand side of the screen within the navigation toolbar to make them expand. Click on the displayed items to open them in the righthand side of the screen. Note that after you open a few screens in this way, you'll have a row of tabs across the top of the screen (similar to Excel). These allow you to switch easily between the most commonly used Guvnor screens. The following figure shows the tab bar in Guvnor:



#### The search screen

If you opened any other tab, you can return to the default search screen by clicking on the **Find** tab of the Guvnor tab bar. You will see something similar to the following screenshot:



The search screen (**Find**) works just like Google, allowing you to search for rules and other assets that you use within your rules. But what's the **Include archived items in list** option for? Remember we said that the Guvnor gave you version management? Nothing is ever deleted, just shuffled into an archive in case you need it again in the future—a bit like the 'Undo' feature in word, only much more powerful. Selecting the **Include archived items in list** checkbox allows you to search for older or deleted versions of rules and other assets.

What's this about assets? I thought this was a book about rules.

Rules can't work in isolation. They need a support team. Assets provide this support—things such as a data model (to get information to and from rules), packages (to organize the rules into folders), and more. Remember that all rules are assets, but not all assets are rules.

### Administration

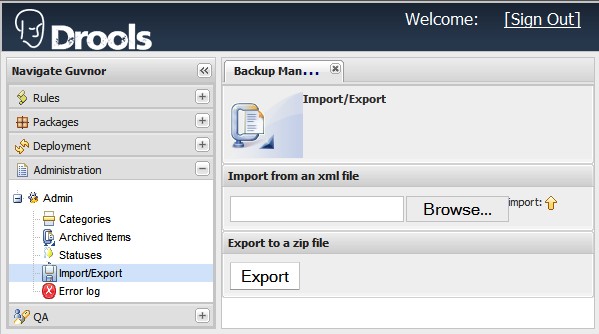
If you did a search on the previous screen, the chances are that no results were returned. By default, there is nothing to search for (unless samples or other rules are hanging around on your machine from a previous version). This is good when we write our rules (as we have a clean sheet), but not so good for our quick tour. To make this clearer, let's load some sample rules.

#### Loading the samples

The Guvnor samples, which provide the business rules for an extremely shady car insurance company, are available from the Book Samples web site **http://code.google.com/p/red-piranha/.**

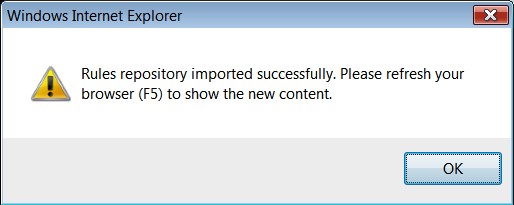
Download the **droolsbook\_chapter3\_sample.zip** file and extract it to a temporary folder. Find the **repository\_export.xml** file. We'll need this in a minute.

Within Guvnor, click on the **Admin** tab, then click on **Import / Export**. (You may need to expand the **Admin** tab by clicking on the '**+**' sign next to it.) You will see the **Backup Manager** tab, as per the following screenshot:



Click on the **Browse** button and select the **repository\_export.xml** file that we found earlier. Then click on **Import** (the button with the up arrow next to it).

After clicking on **OK** to confirm that you want to import the file (remember, this will wipe out anything that you have done up to this date), the system will churn away for a few minutes. Then it will display a message indicating that the import is successful.



Now go back to the search screen (click on the F**ind** tab, or on the left sidebar click on the **Rules tab,** and then select **Rules | Find**) and look for items with the word 'insurance'. You will now get plenty of search results.

Go ahead, and play around with Guvnor. You can always clear everything by redeploying Guvnor. To do this, first stop JBoss (you may need to press *Ctrl+C*). Then, in the folder that you installed JBoss into (for example, **C:\software\JBoss**), there will be a folder called **repository**—this is where Guvnor stores all of the rules. If you delete the folder and restart JBoss, you have a new, clean, version of Drools Guvnor. Be careful though, as this will wipe everything (Rules and other assests) that you've created in this version of Guvnor!

**What did we just do?**

In short, you saved yourself a lot of typing.

By now, you will be getting the feeling that the Guvnor is more than just a web page. It is an industry-standard repository for your rules. A repository is useless without being able to import and export information. We just imported the standard Drools insurance sample into the Guvnor. It makes the web pages that we're going to view a lot clearer when we see some real life examples.

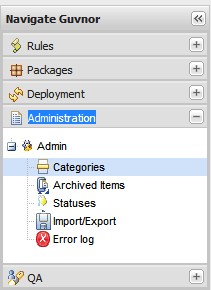
The Drools repository is based on Apache Jackrabbit. This has its own storage, but can be configured to use an industry-standard database such as Oracle, Microsoft SQL Server, or MySQL. No matter how good the database is, it's still reassuring if we can import to it and export from it. For this,

Guvnor uses and XML-based format. If you're curious, open **repository\_ export.xml** in your favorite text editor and have a look at it.

#### More on the admin page

The administration page can do more than just import rules. This page gives you functionality not directly related to rules editing, but vital for managing the system—for example archiving items, managing rule categories, and so on.

To see what else the **Admin** tab can do, open it (if you haven't already done so) by clicking on **Administration | Admin | Categories**. On the lefthand side you'll see the following options:

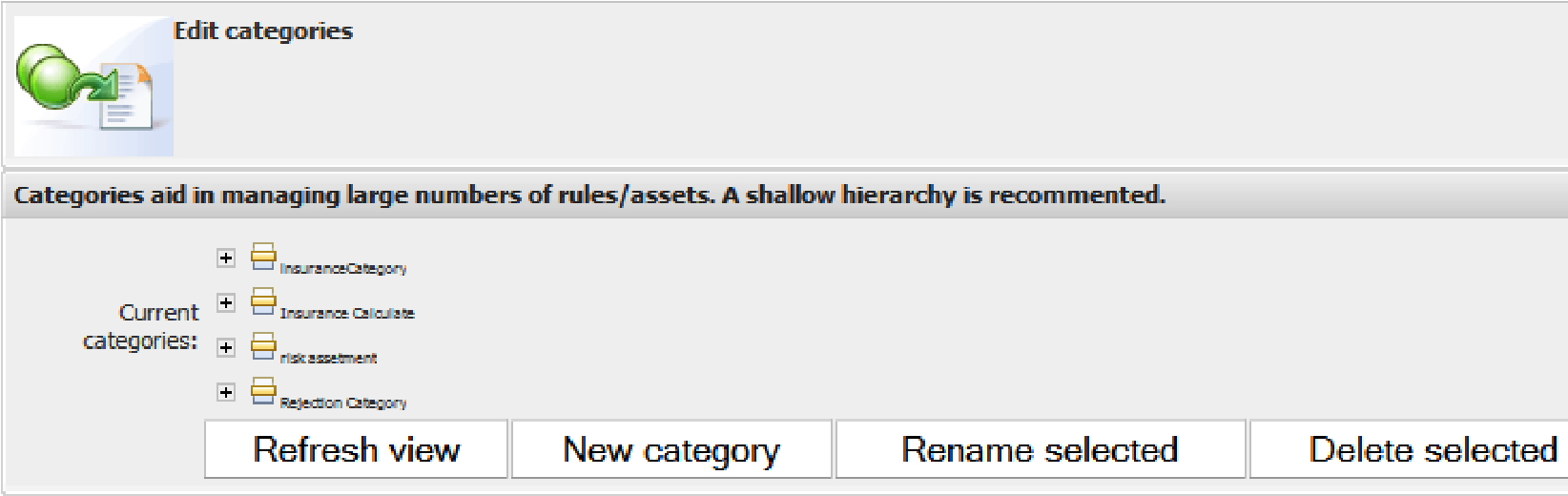


Let's run through these:

1. **Categories**: Every rule lives in a folder (or package) similar to what you might find on your computer's hard disk. But we can also 'tag' the rule or asset with a category name (such as 'sales' or 'accounts'). This screen gives the ability to change these categories and tags.
2. **Archived Items**: Archiving is like deleting (except that nothing really gets lost). If we archive a rule or asset, it normally 'disappears' from that screen. Otherwise the screens would get cluttered with older rules. This option lets us find archived items if we need them again.
3. **Statuses**:Rules and assets don't get written in one sitting. Often they pass through various states such as 'draft', '2nd draft', 'review' and 'production'. The various states will differ depending on the process your organization follows. Luckily, this screen allows us to set various asset states (the default ones being draft and production).
4. **Import/Export**: We've already seen how to import rules into a system. Export is the process in reverse (extracting rules from our system so that we can import them on another computer and/or back into our own system at a later date). Note that the file exported is a compressed ZIP file. The **repository\_export.xml** file needed for importing it later is stored inside it.
5. **Error log**:What happens when something goes wrong? The error log gives you more details (over and above the usual error message that you will see on the screen).

What's the difference between a category and a status? Although both are used to describe a rule, they do different things. A status may change over the lifetime of a rule (for example, moving from draft to production), while a rule will normally stay in the same category (for example, sales) during its lifetime. In addition, a rule can have only one status, but can have many categories assigned to it.

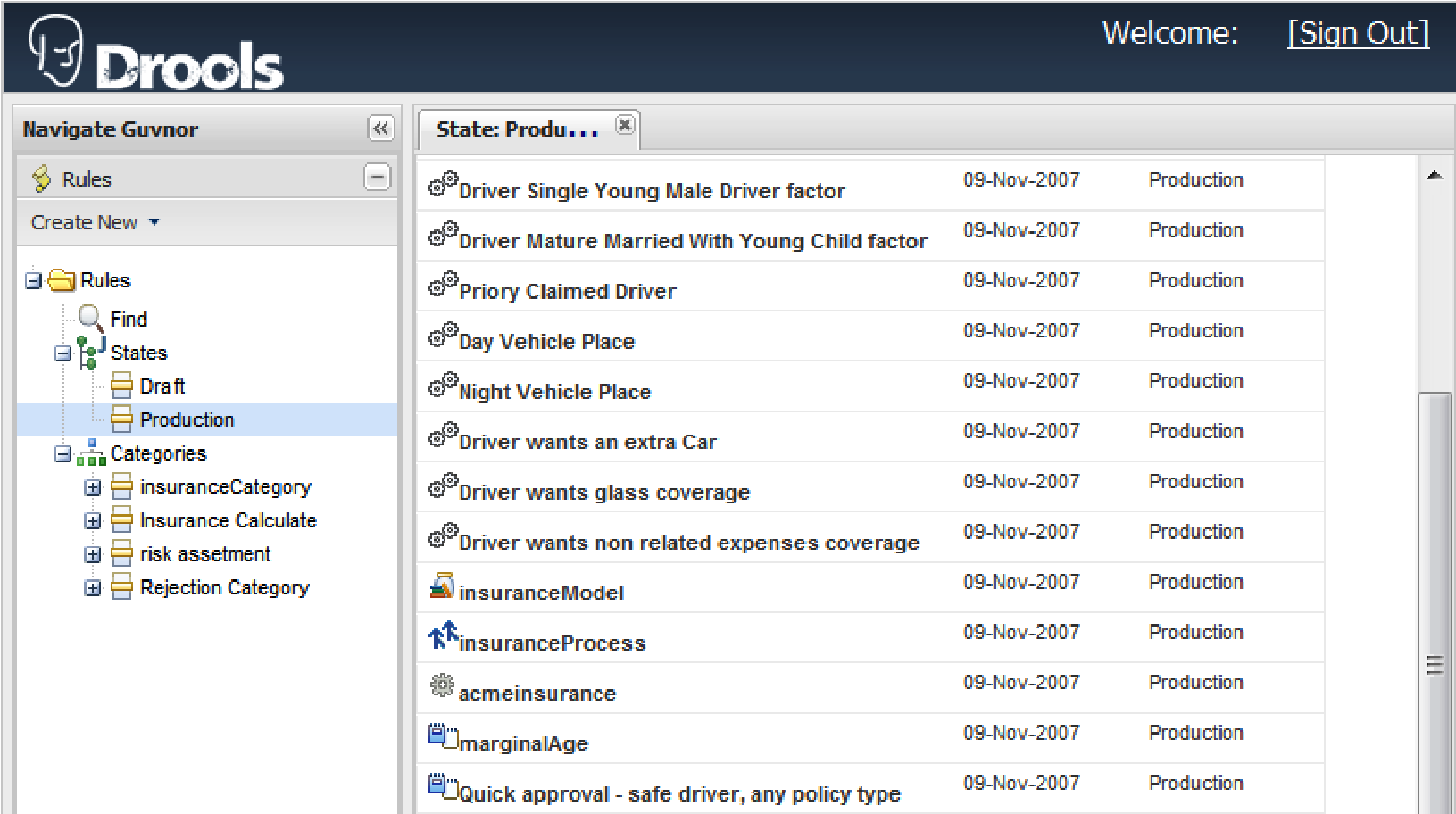
Categories are important—every rule and every asset must have a least one of them. So let's take a minute to look at the following **Edit categories** screen. The screen shows the insurance categories that we imported in the previous step. If you want, you can add, rename, or delete category names here. We can even nest categories inside categories. But don't get carried away, as often the complexity isn't needed!



#### Rules

Now that we understand categories, we're ready to look at the rules pages. We already saw the search/find rules screen as soon as we opened Guvnor. To see the other rules pages, click on the **Rules** tab, then expand both the **States** and **Categories** submenus (by clicking on the '**+**' sign). Double-clicking on the **Production** state will allow you to see the insurance rules, as shown in the following screenshot.

If you're clicking through **States** and **Categories**, you'll see that many of the rules/ assets are repeated. That's OK—a rule will have a state and one or more categories. Think of it as many different ways of finding the same thing.



There are a lot of entries here (on the righthand side), but they all belong to a few simple types.

##### Process

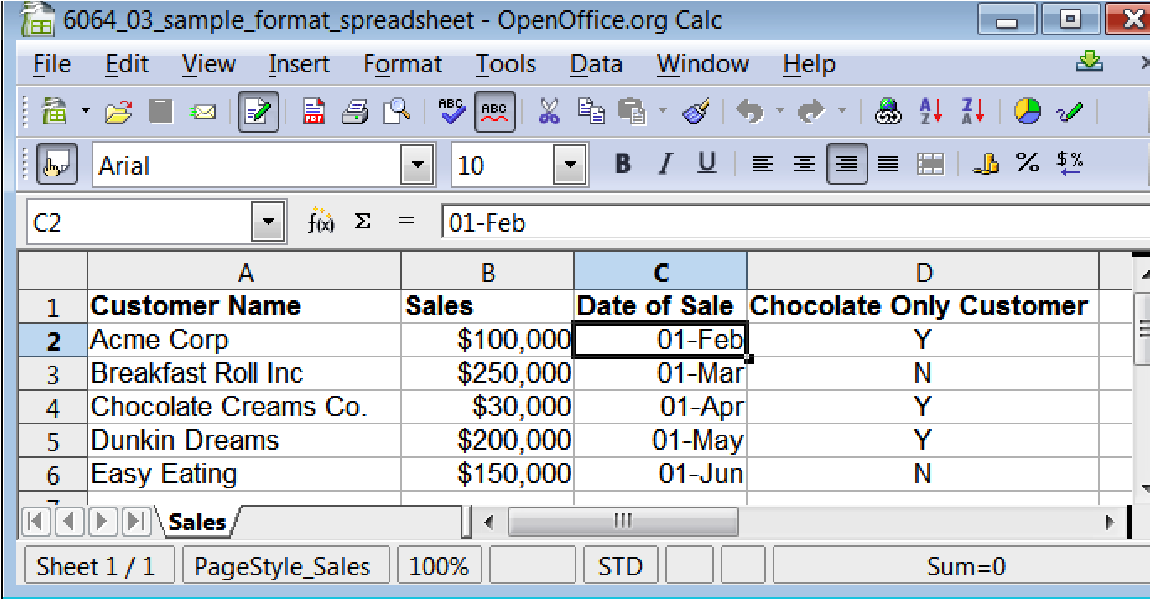
You are probably familiar with workflow diagrams. JBoss Rules allows you to draw workflow diagrams in the Eclipse IDE (but not yet in Guvnor). This allows you to have more control over the order in which groups of rules fire. The **insuranceProcess** item in the above screenshot is an example of a process.

##### The model

We mentioned earlier that rules are deployed as a part of a larger system. So we need a way of getting information into and out of the rule engine. The model lets us do this.

To explain this in more detail, think of how you might email sales information around our imaginary chocolate company from the previous chapter. You'd probably send the information in a spreadsheet similar to one that follows, showing customer sales for the month of February.

The **insuranceModel** item in the above screenshot is an example of this.



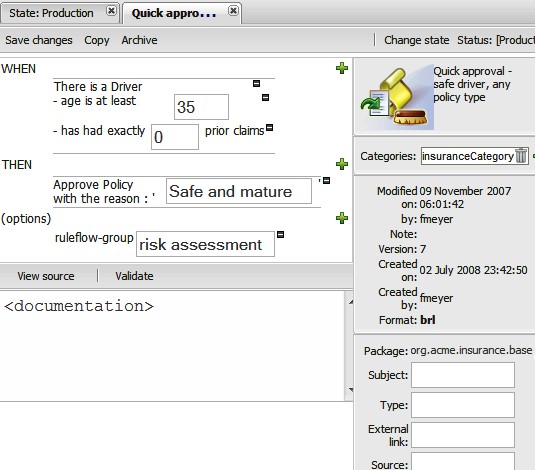
In the above screenshot, we have four columns showing **Customer Name**, **Sales**, **Date of Sale**,and an indication of whether we've sold items other than chocolate to the customer. Imagine the spreadsheet for March—the customer details might change, but the structure of the spreadsheet (including the columns) remains the same. The structure of the spreadsheet (but not the contents) is the model. For JBoss Rules, it's written in Java.

Clicking on a model item (in the case of the rules in our Guvnor example, the **insuranceModel** item) is a shortcut to the screen to upload or download the model files that we created in Java.

##### Guided rules

The aim of the Guvnor is to allow the easy editing of business rules. The guided rules screen (available by double-clicking on the **Quick approval - safe driver,** **any policy type** rule in the previous screen) allows you to do this. This is a guided editor that makes intelligent suggestions to help you write your rules. With this editor you won't be typing any text, but using dropdowns and clicking on icons to create business rules.

The following screenshot is the guided rule editor for the **Quick approval** rule:



The key features of the lefthand side of this screen are:

* Buttons at the top left to **Save changes**, **Copy**, and **Archive** the rule
* The **When** and **Then** sections of the business rules on the upper-left and middle left
* Three green '**+**' icons next to the **When**, **Then**, and **Options** sections to add more constraints or consequences
* Multiple '**-**' icons (next to the textboxes) allowing you to delete existing constraints or consequences
* Options that describe the rule; for example, is it part of a process flow, is it enabled, and the date that it is effective from
* Buttons (near the bottom), used to view the source (**View source**) and validate the rule (**Validate**)
* A space (at the bottom of the screen) to allow (optional) information about the rule to be specified

On the righthand side of the screen, you will find the following features:

* Options on the upper-right,used to change the **Status** and **Categories**
* Notes on the package to which the rule belongs
* **Version** history and other metadata about the rule, such as who created it and when

Everything that you do on this screen actually creates the technical rule behind the scenes (and hides the complexity from you). The **View Source** button allows you to see (but not edit) the rule that is created for you. If you want the additional power of editing technical rules, you can always use the technical rules screen.

##### Technical rules

Most of the assets in the insurance sample are actually technical rules (for example, the **Quick approval** rule. If you open the **Driver Glass Coverage** rule (by doubleclicking on it in the list of rules) you'll see a similar screen, but with text instead of the guided editor.

Note that this text-based rule follows the same "when…then" format.  **when**

**$driver : Driver ( driverID : id )**

**$supple : SupplementalInfo ( driverId == driverID, glassCoverage == true)**

**then**

**$driver.updateInsuranceFactor( 1.05 );**

**System.out.println("Driver wants glass coverage: " +**

**$driver.getInsuranceFactor());**

Or, in plain English:

**When**

**There is a Driver**

**And that Driver has requested glass coverage**

**Then**

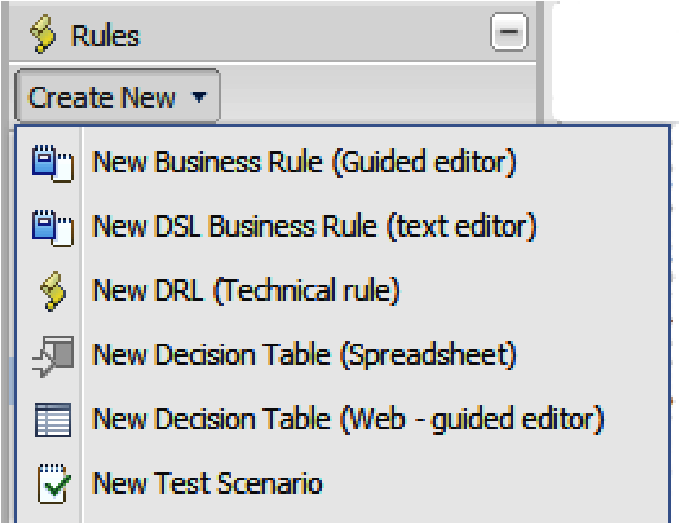
**Update the driver's insurance factor**

**Print a message saying that the driver wants coverage**

The technical rules follow a predictable pattern, so it gets easier to understand and even to write. We've to leave something for the next few chapters!

##### Creating a new rule

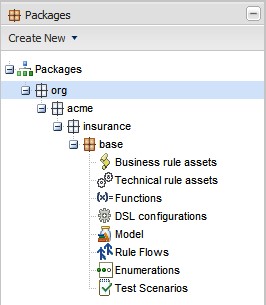
Clicking on the **Create New** button just underneath the **Rules** section on the navigation bar brings up the following menu. This menu allows you to create new technical and business rules (like the one's we've just seen). It also allows you to create DSL-based rules (a way of writing near-English business rules), Decision Tables (an Excel-like format for business rules), and test scenarios (to make sure that your rules work the way you intend them to).



#### Packages

Packages are like folders. They are a way of organising rules and assets. The difference between packages and directories comes at deployment time, when everything in one package gets deployed at the same time. Opening the package (**org | acme | insurance | base**) shows you all of the assets available in the package.

Most of these concepts (business rules, technical rules, DSL, models, rule flows, and test scenarios) are familiar, but there are two new items: **Functions** and **Enumerations**. We might want to call **Functions**, which are useful for calculations and the like, from the rules. **Enumerations** are lists of values that we can use in our rules.

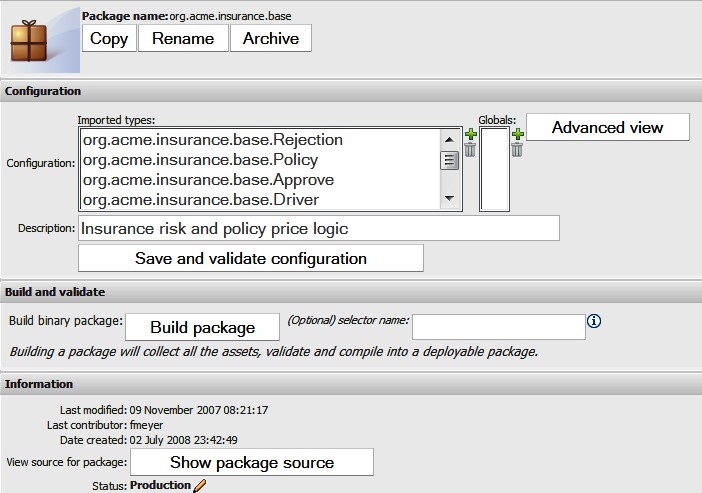


The **Create New** button (just below the **Packages** tab) allows you to create new ones of each of these items.

If you select the main package (**base**), you will be able to see a summary of the package details.

Java recommends a standard notation for packages names (that's where the name **org.acme.insurance.base** comes from). Although you don't have to follow this convention, there is no harm in doing so (especially when we start talking about the Java-based rule model later). The package name looks a little bit like an Internet web address (although it doesn't actually link to anything). The format is: **companyurl.projecturl.subproject. anyotherdivisionsrequired**.

Most of the packages that you create will only need three or four levels. However, there can be as many subdivisions as you need. You will see more of these subdivisions when we talk about Java code (Fact models).



Some of the features on the package details screen (from top to bottom) are:

* The buttons **Copy**, **Rename**, and **Archive** at the top of the screen, used to to copy, rename, and archive (delete but save a copy) respectively.
* Next, just below the **Configuration** heading are the statements that import the fact model into the package. These are normally generated automatically when we add the (Java) fact model.
* An optional **Description** of the package.
* A **Save and validate configuration** button, used to to save (and validate) the configuration.
* A **Build package** button, used to to build the package and put it into a deployable condition (or let you know of any problems).
* A **Show package source** button, used to show the package source (for example, the technical rules language that has been written by Guvnor on your behalf). This shows the entire package (including imports and functions), and not just single rules.

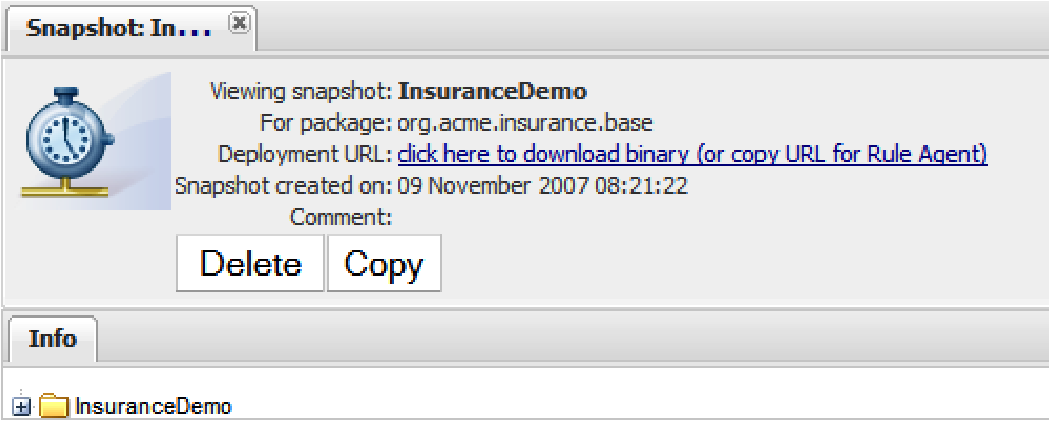
The button used to build the package is important, as building the package is the step before deployment (that is, using our rules in a real-life production system).

#### Deployment

To open the deployment screen, click on the **Deployment** bar in the lefthand navigation section (available on all screens). This screen plays a very important role. It gets your rules and assets from the Guvnor editor and puts them into the production systems.

When you edit your rules, they don't get deployed (to the live or real world system) immediately. Can you imagine being in the middle of writing the second of three new rules and having the incomplete rule set deployed? The **Deployment** tab allows you to control when your rules are released to the end users. It also allows you to view previous deployments.

Clicking on the **Deployment** tab displays the following screenshot. The list of available snapshots comes from the packages built by the **Build package** button under the **Package** tab.



In addition to the **Delete** and **Copy** buttons, there is a web link (URL). You can use this in two ways. You can click on it here to download the package (if you wish to copy and deploy it manually to the target system). Or you can right-click on the link and copy the URL. Drools provides a component (called the RuleAgent) that we can deploy into our production system. The RuleAgent can check for updates to the package (via the URL) and deploy them to production automatically.

The architects of your target system will probably have a specific deployment plan (generally it's not a good idea to deploy rules directly to production). Guvnor gives you a couple of options, but this default one will help you to get your system up and running quickly.

#### QA—Quality Analysis

Here's a problem: You write rules and you check them to make sure they do what you intend them to do. They work OK. Then you change a rule. So you have to test them again. Still OK this time. What if it's the 60th time you've made a tiny change? Are you tempted to skip the testing yet? Or maybe you'll test, but not as well as you should. What if you've 600 rules? Do you test all of them?

##### Automating testing in Guvnor

The solution: You automate the testing. This is what the Guvnor Quality Analysis page allows you to do.

This page is pretty simple. You know the inputs to your rules (for example, the insurance application form for your typical first-time 21-year old driver) and the outputs from the rules (the insurance premium that they should pay). The inputs and outputs should be the same every time, which makes them ripe for automation.

The testing framework alerts you if the test results differ from what you expect. That way, you spend less time testing and more time playing golf (or whatever it is that you do). And you end up with better quality tests.

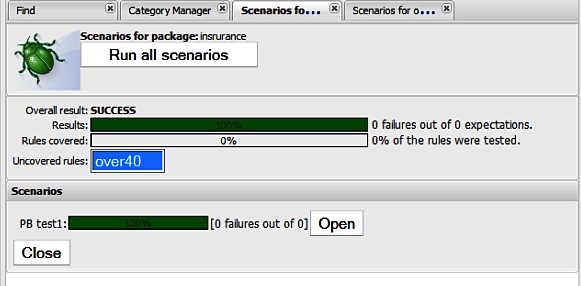
We'll create a simple test scenario when we'll write our 'Hello World' sample in the next section. But as you can see from the following screenshot, this screen allows you to run all of the tests.

Many people (me included) recommend **Test-Driven Development** or

**TDD** that is, you write your test **before** you write your business rule (write test, write rule, verify test, write test, and so on). The reason for testing first is that, as a human, you may be tempted to "forget" to write your test if the business rule appears to work OK.

Over time, these single tests build up to give you a 'safety net' that dramatically reduces the cost of things going wrong. It's much easier and cheaper to fix something that you've just written (when your tests fail) rather than three months later when you go live and your airline is giving away free transatlantic flights by accident on it's web site.

The test scenario screen allows us to run all of the tests in our package simultaneously, which is a useful sanity check for our rules, before deploying them.



##### The analysis page

The power of business rules comes from writing many simple rules that cover most business scenarios. For example, insurance rates for drivers under 20 years of age, insurance for the people above 40 years, insurance for the people above 60 years. But what if we leave a gap (in this example, insurance for drivers in the age range of 20-40 years)?

The analysis page carries attempts to catch these gaps. It's not perfect (especially for more complicated scenarios), but for the items that it catches, you'll be glad that you used it.

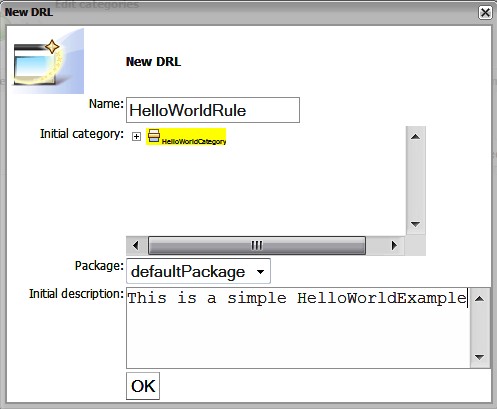
#### Hello World example

It's traditional to show the simplest possible example—a rule that just says 'Hello World' when it is fired during a test scenario.

##### Writing the rule

The easiest way to do this is to create a new technical rule. In Guvnor, select the **Rules** tab, click on the **Create new** dropdown, and then select **New DRL** (technical rule). A new screen will appear.

We enter a name (**HelloWorldRule**), a category (that can be created using the **Admin** tab we saw earlier), and use the **defaultPackage** and (optionally) enter a description of the rule.



Clicking on **OK** will take us to the (technical) rule editor, similar to the one we saw on our tour of the Guvnor (although at this point the rule is blank). In the main part of the screen (the blank part) enter the following text: **rule "Hello World" when eval(true) then**

**System.out.println("Hello world"); end**

The important thing about this rule is the **when… eval(true)** statement. This means that the rule will always try to fire and carry out the **then** part— that is, print the 'Hello World' message.

Normally, our business rules would be much more choosy about 'when they fire' (and have a lot more conditions in the 'when' part). But for our simple sample, this suits us fine.

Next, save the changes by clicking on the **Save changes** button. You should be asked for an optional 'check in' comment after clicking this button. Congratulations, you've written your first business rule!

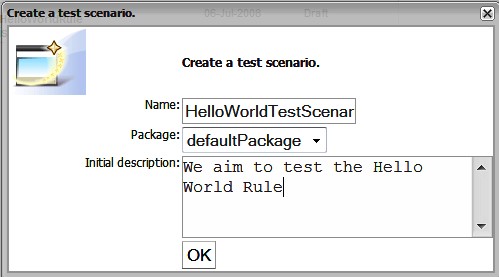
##### Firing the rule

Now we've a problem. Unlike most computer languages (for example, Java or C#), we can't 'run' a set of business rules—after all they have no single start point!

So how do we test our shiny new 'Hello World' rule? The answer is that we contrive a scenario that we know should cause our rule to fire. In Guvnor, this is relatively easy to do using the QA screen that we saw above. We'll use the QA screen to create a new test scenario where we can exercise our business rule.

This point about not being able to run rules is important and may be a major change from what you are used to. Remember that with rules, you say 'do this when this is true' and leave the rule engine to carry out that instruction when it finds itself in that scenario.

Let's create a new test scenario in Guvnor. A little bit strangely, this is done via the **Packages** tab. Select **Packages | Create new | New test scenario**, and a screen similar to the one shown in the following screenshot will be displayed:



Similar to what you did for creating a new business rule, enter a **Name**, **Package**, **Initial description** and then press **OK**. The scenario-editing screen will then be displayed.

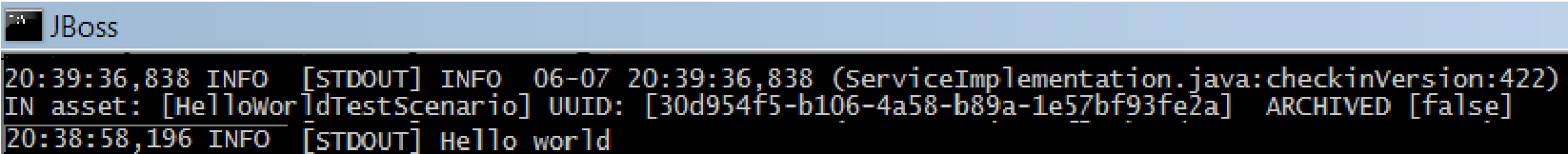
This screen has three green '**+**' signs that we use to set up our scenario. From top to bottom these are:

* **GIVEN** — allows us to set our inputs (that is, create a scenario in which we know a business rule will fire)
* **EXPECT** — allows us to inspect the output after the rule has fired, to make sure that it has worked correctly
* **globals** — allows us to pass in environmental variables that the rule may need



Fortunately for us, as our rule is set to fire every time, we don't need to set these up. All we have to do is click the **Run scenario** button. When we do this, two things will happen:

1. We will see a message appear on the web page—**1 rules fired in 0ms**. Pressing the **Show rules fired** next to this message shows that the 'Hello World' rule was activated.
2. We will see a 'Hello World' message in the web server (JBoss) log, similar to the following:



**What just happened?**

In the first part we created a rule that essentially said: **rule "Hello World"**

**when**

**Anytime the rules are run**

**then**

**Print a message to the console ("Hello world"); end**

The next step (firing the rules) was to contrive a situation where the preconditions in the **when** part were met so that the rule would fire. For such a simple rule, this was easy. In fact, we had nothing to add in this case.

Finally, when we ran this scenario, the rule was activated, and the **Hello world** message was successfully printed to the console.

## Summary

In this chapter we did three things. We loaded the Drools insurance sample into the Guvnor editing tool to give us some very good examples. Then we looked at the various Guvnor screens and saw that Guvnor can not only write rules (using both guided and advanced editors), but can also organize them and other assets into packages. The Guvnor screens also allow us to test and deploy these rules. Finally, we wrote our very first business rule—the traditional 'Hello World' message, announcing to everyone that we are now business rule authors.

We will use all of these skills in the next chapter. In that chapter, we will start on more sophisticated business rules, using both Guvnor and more advanced editing options.