

# ¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.  
 Ese apoyo que siempre quiso tener...

## 1. Desarrollo de componentes y proyectos a medida



## 2. Auditoría de código y recomendaciones de mejora

## 3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



## 4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,  
 HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)  
 Gestor de contenidos (Alfresco)  
 Aplicaciones híbridas

Tareas programadas (Quartz)  
 Gestor documental (Alfresco)  
 Inversión de control (Spring)

Control de autenticación y  
 acceso (Spring Security)  
 UDDI  
 Web Services  
 Rest Services  
 Social SSO  
 SSO (Cas)

JPA-Hibernate, MyBatis  
 Motor de búsqueda empresarial (Solr)  
 ETL (Talend)

Dirección de Proyectos Informáticos.  
 Metodologías ágiles  
 Patrones de diseño  
 TDD

BPM (jBPM o Bonita)  
 Generación de informes (JasperReport)  
 ESB (Open ESB)



» Estás en: Inicio » Tutoriales » Introducción a Guvnor



[Jose Manuel Sánchez Suárez](#)

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

 [Ver todos los tutoriales del autor](#)

**Catálogo de servicios Autentia**



Fecha de publicación del tutorial: 2013-03-01

Tutorial visitado 6 veces [Descargar en PDF](#)

## Introducción a Guvnor

### 0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Instalación.
- 4. Primeros pasos.
  - 4.1. Definición de procesos.
  - 4.2. Definición de reglas.
- 5. Cómo hacer uso de recursos de guvnor.
- 6. Referencias.
- 7. Conclusiones.

### 1. Introducción

JBoss Guvnor es el nombre del proyecto open source sobre el cual se libera la versión empresarial del BRMS de Jboss. Como ya vimos en el tutorial de [introducción a drools](#), Guvnor es básicamente un repositorio centralizado de reglas de negocio que permite gestionarlas con una interface web; si bien, dentro de la suite de productos de jBPM, Guvnor se convierte en un repositorio central de conocimiento en el que, además de almacenar reglas de negocio, podemos:

- crear reglas guiadas con un wizard,
- crear tablas de decisión de una forma visual,
- crear, editar y almacenar modelos de negocio, bien creados en lenguaje DSL o realizando un upload de POJOS, para que puedan usarse como modelo de las reglas de negocio,
- crear, editar y almacenar procesos, pudiendo editarlos mediante una interfaz web,
- crear nuestra propia definición tipos de tareas dentro de un proceso de negocio,
- crear los formularios asociados a tareas del proceso de negocio,
- simular escenarios de prueba sobre los procesos de negocio,
- definir contextos de spring para la inicialización de drools conforme a una serie establecida de reglas y procesos, que luego estarán disponibles vía http,
- categorizar los contenidos,
- llevar un histórico de cambio de versiones de todos los recursos que almacena, y
- permite el acceso a los contenidos del repositorio a través de webdav y servicios REST. También podemos conectar con los recursos del repositorio desde Eclipse, para llevar a cabo su descarga y, del mismo modo, actualización o subida

En este tutorial vamos a hacer uso de Guvnor fuera de la demo de instalación de jBPM y a realizar una vista rápida de las posibilidades de la herramienta.

El objetivo es disponer de los proyectos necesarios, para ejecutar guvnor en cualquier servidor de aplicaciones o contenedor de servlets, mavenizándolos.

### 2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.4 GHz Intel Core i7, 8GB DDR3 SDRAM).
- Sistema Operativo: Mac OS X Lion 10.7.4
- Drools Guvnor 5.5.0.Final
- Apache Tomcat 7.0.35



**Síguenos a través de:**



**Últimas Noticias**

- » [Vendedor: Soy inseguro, filtra o elige por mi: si quieres que te compre.](#)
- » [Comentando el libro: El arte de pensar, de Rolf Dobelli](#)
- » [Ya está a la venta mi segundo libro: Planifica tu éxito, de aprendiz a empresario](#)
- » [Ya esta disponible en eBook mi primer libro: Informática Profesional](#)
- » [Comentando el libro: La inteligencia reformada, las inteligencias múltiples en el siglo XXI de Howard Gardner](#)

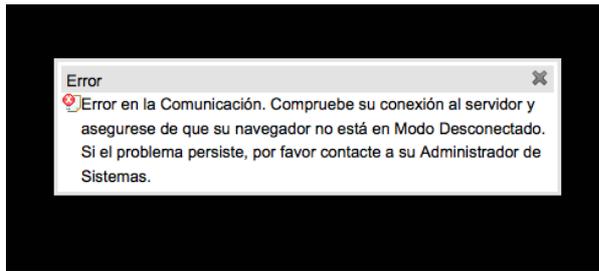
[Histórico de noticias](#)

**Últimos Tutoriales**

- » [Gestión de expedientes en el ámbito de las Administraciones Públicas \(IV\): buscando en las forjas una solución.](#)
- » [Gestión de expedientes en el ámbito de las](#)

### 3. Instalación.

En la instalación que viene por defecto con jBPM 5.4, podemos tener problemas de comunicaciones...



Aunque en realidad es un error de pérdida de conectividad entre el cliente y el servidor que se da frecuentemente.

De ahí que optemos por realizar una instalación personalizada, con el soporte de maven, teniendo más control sobre la generación del war, su despliegue y nos permita:

- llevar a cabo una personalización de la interfaz de usuario,
- modificar la estrategia por defecto de autenticación y autorización, o
- modificar la estrategia por defecto de persistencia de contenidos.

Guvnor hace uso de las siguientes tecnologías:

- Apache Jackrabbit como repositorio de contenidos, pero se puede cambiar la configuración para que se persistan en una base de datos relacional,
- Jboss Seam 3 en la capa de control, negocio y para la gestión de seguridad. La autenticación por defecto está configurada para un único usuario administrador pero se puede modificar para que haga uso de una autenticación tipo JAAS o enganche con el SSO de CAS, y
- GWT en la capa de presentación.

Además, para hacer uso del modelador BPMN2 debemos hacer uso de una segunda aplicación, el designer.

Lo primero que vamos a hacer es crear un proyecto parent, que nos permitirá establecer las propiedades comunes al resto y el repositorio desde el cuál realizar las descargas.

A continuación el pom.xml del proyecto parent:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-NS"
3     <modelVersion>4.0.0</modelVersion>
4
5     <groupId>com.autentia.bpm.repository</groupId>
6     <artifactId>tnt-jbpm</artifactId>
7     <version>0.0.1-SNAPSHOT</version>
8     <packaging>pom</packaging>
9
10    <name>tnt-jbpm</name>
11    <inceptionYear>2013</inceptionYear>
12
13    <properties>
14        <java.version>1.6</java.version>
15        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
16        <drools.version>5.5.0.Final</drools.version>
17        <jbpm.version>5.4.2.Final</jbpm.version>
18    </properties>
19
20    <repositories>
21        <repository>
22            <id>jboss-public-repository-group</id>
23            <name>JBoss Public Maven Repository Group</name>
24            <url>https://repository.jboss.org/nexus/content/groups/public-jboss</url>
25            <layout>default</layout>
26            <releases>
27                <enabled>>true</enabled>
28                <updatePolicy>never</updatePolicy>
29            </releases>
30        </repository>
31    </repositories>
32
33    <modules>
34        <module>tnt-guvnor</module>
35        <module>tnt-designer</module>
36    </modules>
37 </project>

```

Los módulos serán dos:

- tnt-guvnor: el módulo que contiene la aplicación guvnor, que proporciona una interfaz web del repositorio de contenidos, y
- tnt-designer: la aplicación web que permitirá crear procesos BPMN2 desde guvnor.

#### 3.1. tnt-guvnor.

El pom.xml del proyecto tnt-guvnor es el siguiente:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-NS"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

```

Administraciones Públicas (III): BPM y la gestión de procesos de negocio.

» jBPM5: usando nuestra propia base de datos.

» AngularJS: primeros pasos.

» Gestión de expedientes en el ámbito de las Administraciones Públicas (II): requisitos.

#### Últimos Tutoriales del Autor

» Gestión de expedientes en el ámbito de las Administraciones Públicas (IV): buscando en las forjas una solución.

» Gestión de expedientes en el ámbito de las Administraciones Públicas (III): BPM y la gestión de procesos de negocio.

» jBPM5: usando nuestra propia base de datos.

» Gestión de expedientes en el ámbito de las Administraciones Públicas (II): requisitos.

» Introducción a OpenLayers: un visor de mapas javascript.

#### Últimas ofertas de empleo

2011-09-08

Comercial - Ventas - MADRID.

2011-09-03

Comercial - Ventas - VALENCIA.

2011-08-19

Comercial - Compras - ALICANTE.

2011-07-12

Otras Sin catalogar - MADRID.

2011-07-06

Otras Sin catalogar - LUGO.

Jose Manuel Sánchez  
sanchezsuaresj

adictosaltrabaj Gestión de expedientes en el ámbito de las Administraciones Públicas (IV): buscando en las forjas una solución. - [kcy.me/g0de](http://kcy.me/g0de)  
3 days ago · reply · retweet · favorite

adictosaltrabaj Gestión de expedientes en el ámbito de las Administraciones Públicas (III): BPM y la gestión de procesos de negocio - [kcy.me/fy2p](http://kcy.me/fy2p)  
4 days ago · reply · retweet · favorite

adictosaltrabaj jBPM5 y como hacer uso de nuestra propia base de datos en vez de H2. - [kcy.me/ftab](http://kcy.me/ftab)  
7 days ago · reply · retweet · favorite

sanchezsuaresj If you can't



```

4 <modelVersion>4.0.0</modelVersion>
5
6 <parent>
7 <groupId>com.autentia.bpm.repository</groupId>
8 <artifactId>tnt-jbpm</artifactId>
9 <version>0.0.1-SNAPSHOT</version>
10 </parent>
11
12 <groupId>com.autentia.bpm.repository</groupId>
13 <artifactId>tnt-guvnor</artifactId>
14 <version>0.0.1-SNAPSHOT</version>
15 <packaging>war</packaging>
16
17 <inceptionYear>2013</inceptionYear>
18
19 <build>
20 <finalName>drools-guvnor</finalName>
21 <plugins>
22 <plugin>
23 <groupId>org.apache.maven.plugins</groupId>
24 <artifactId>maven-compiler-plugin</artifactId>
25 <version>2.5.1</version>
26 <configuration>
27 <source>${java.version}</source>
28 <target>${java.version}</target>
29 </configuration>
30 </plugin>
31 <plugin>
32 <groupId>org.apache.maven.plugins</groupId>
33 <artifactId>maven-surefire-plugin</artifactId>
34 <version>2.4.2</version>
35 <configuration>
36 <skipTests>>true</skipTests>
37 </configuration>
38 </plugin>
39 <plugin>
40 <groupId>org.apache.maven.plugins</groupId>
41 <artifactId>maven-war-plugin</artifactId>
42 <version>2.3</version>
43 </plugin>
44 <plugin>
45 <groupId>org.mortbay.jetty</groupId>
46 <artifactId>maven-jetty-plugin</artifactId>
47 <version>6.1.26</version>
48 <configuration>
49 <scanIntervalSeconds>10</scanIntervalSeconds>
50 <stopKey>foo</stopKey>
51 <stopPort>9999</stopPort>
52 <stopPort>9966</stopPort>
53 <connectors>
54 <connector implementation="org.mortbay.jetty.nio.SelectChannelC
55 <port>9080</port>
56 <maxIdleTime>60000</maxIdleTime>
57 </connector>
58 </connectors>
59 <webAppConfig>
60 <contextPath>/drools-guvnor</contextPath>
61 </webAppConfig>
62 </configuration>
63 </plugin>
64 </plugins>
65 </build>
66
67 <dependencies>
68 <dependency>
69 <groupId>org.drools</groupId>
70 <artifactId>guvnor-webapp-drools</artifactId>
71 <version>${drools.version}</version>
72 <type>war</type>
73 </dependency>
74 </dependencies>
75 </project>

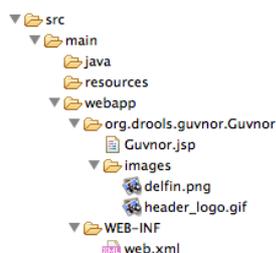
```

Configuramos el plugin de jetty para poder levantar la aplicación en un jetty de forma embebida:

```
1 mvn package jetty:run-exploded
```

Con el soporte de war overlay de maven, generaremos un proyecto war con el mismo contenido que el proyecto guvnor-webapp-drools, sobrescribiendo aquellos ficheros que nos interese modificar el proyecto original.

Como ejemplo, vamos a incluir nuestras imágenes corporativas para personalizar la interfaz de usuario de guvnor.



Y podemos modificar el layout para adaptar los estilos que necesitemos; tomaremos el fichero Guvnor.jsp del directorio de despliegue y lo incluimos en el path src/main/webapp/org.drools.guvnor.Guvnor/; en nuestro caso hemos modificado los iconos:

```

1 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
2 <meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7" />

```

```

3 <meta name="gwt:property" content="locale=<%=request.getLocale().toString()%>">
4 <title>JBoss Guvnor</title>
5 <link rel="shortcut icon" href="images/delfin.png" type="image/png" />
6 <link rel="icon" href="images/delfin.png" type="image/png" />

```

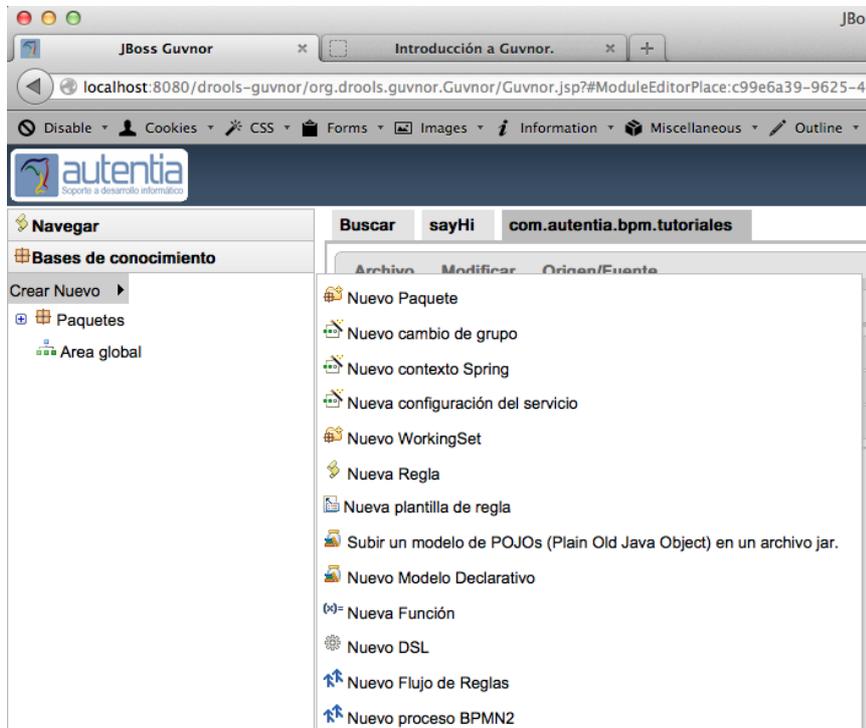
Por último, si queremos desplegar en Apache Tomcat 7, debemos "traernos" también el descriptor de despliegue **web.xml** para solucionar un bug de Jboss Seam 3, una NPE, NullPointerException. Debemos incluir el siguiente parámetro de contexto.

```

1 <context-param>
2   <param-name>org.jboss.seam.transaction.disableListener</param-name>
3   <param-value>true</param-value>
4 </context-param>

```

Con todo ello, bien en jetty o Tomcat, podríamos desplegar y tendríamos la siguiente interfaz de usuario accediendo a la siguiente url: <http://localhost:9080|8080/drools-guvnor>.



También podríamos modificar la política de autenticación sobrescribiendo el fichero beans.xml, pero esto queda fuera del objetivo de este primer tutorial.

### 3.2. tnt-designer.

El pom.xml del proyecto tnt-designer es el siguiente:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5
6   <parent>
7     <groupId>com.autentia.bpm.repository</groupId>
8     <artifactId>tnt-jbpm</artifactId>
9     <version>0.0.1-SNAPSHOT</version>
10  </parent>
11
12  <groupId>com.autentia.bpm.repository</groupId>
13  <artifactId>tnt-designer</artifactId>
14  <version>0.0.1-SNAPSHOT</version>
15  <packaging>war</packaging>
16
17  <inceptionYear>2013</inceptionYear>
18
19  <build>
20    <finalName>designer</finalName>
21    <plugins>
22      <plugin>
23        <groupId>org.apache.maven.plugins</groupId>
24        <artifactId>maven-compiler-plugin</artifactId>
25        <version>2.5.1</version>
26        <configuration>
27          <source>${java.version}</source>
28          <target>${java.version}</target>
29        </configuration>
30      </plugin>
31      <plugin>
32        <groupId>org.apache.maven.plugins</groupId>
33        <artifactId>maven-surefire-plugin</artifactId>
34        <version>2.4.2</version>
35        <configuration>
36          <skipTests>true</skipTests>

```

```

37     </configuration>
38 </plugin>
39 <plugin>
40   <groupId>org.apache.maven.plugins</groupId>
41   <artifactId>maven-war-plugin</artifactId>
42   <version>2.3</version>
43 </plugin>
44 <plugin>
45   <groupId>org.mortbay.jetty</groupId>
46   <artifactId>maven-jetty-plugin</artifactId>
47   <version>6.1.26</version>
48   <configuration>
49     <scanIntervalSeconds>10</scanIntervalSeconds>
50     <stopKey>foo</stopKey>
51     <stopPort>9999</stopPort>
52     <stopPort>9966</stopPort>
53     <connectors>
54       <connector implementation="org.mortbay.jetty.nio.SelectChannelCo
55         <port>9080</port>
56         <maxIdleTime>60000</maxIdleTime>
57       </connector>
58     </connectors>
59     <webAppConfig>
60       <contextPath>/designer</contextPath>
61     </webAppConfig>
62   </configuration>
63 </plugin>
64 </plugins>
65 </build>
66
67 <dependencies>
68   <dependency>
69     <groupId>org.jboss.drools</groupId>
70     <artifactId>jbpm-designer</artifactId>
71     <version>2.4.0-SNAPSHOT</version>
72     <type>war</type>
73   </dependency>
74 </dependencies>
75 </project>

```

El proyecto no requiere de más configuración y, tras desplegar tendremos accesible la aplicación a través de la siguiente url: <http://localhost:8080/designer>

Ahora ya podremos desplegar ambas aplicaciones en el mismo servidor y crear un proceso BPMN2 desde Guvnor.

La recomendación sería redimensionar la configuración de memoria de arranque de la JVM, para evitar problemas de falta de memoria, puesto que el despliegue de ambas aplicaciones puede comenzar a ser pesado.

#### 4. Primeros pasos.

Vamos a hacer una visita rápida; crearemos un proceso y una regla de negocio, para poder hacer uso después de los mismos desde una clase de prueba.

##### 4.1. Definición de procesos.

Ya hemos visto que tenemos accesible el editor visual de procesos BPMN directamente desde la interfaz de Guvnor, para que funcione correctamente debemos respetar el nombre original de los contextos de aplicación puesto que son dos proyectos distintos y en ellos se comunican a través de los servicios rest.

Cuando entremos más en profundidad tendremos que configurar ambos proyectos para que sepan hablar entre ellos fuera del dominio local y haciendo uso de la política de autenticación definida; ahora, por defecto, se buscan en localhost y con usuario admin.

Para crear un proceso nuevo no hay más que posicionarse en un paquete y pulsar sobre Crear nuevo > Nuevo Proceso BPMN2; se mostrará la interfaz que hemos visto con anterioridad y en ella disponemos de una paleta de formas, un área de edición y una paleta de propiedades de cada uno de los objetos del diagrama.

Se hace importante la opción de validar "Validate Process" del menú para comprobar que efectivamente nuestro diagrama

cumple con la especificación y las validaciones mínimas.

Cuando terminemos con el proceso debemos empaquetarlo y lo ideal es asignar un número de versión, para poder añadir a posteriori modificaciones en el proceso sin perder el histórico.

Una vez hecho esto, sobre la vista de paquete, en la pestaña de "Modificar", podemos "Construir el paquete" para poder hacer uso del proceso empaquetado.

Debemos comprobar que disponemos de la imagen del proceso que se usará para indicar el paso en el que se encuentra un proceso desde la consola. Deberíamos tener en el listado de recursos del paquete algo similar a esto:

<input type="checkbox"/>			sayHi-image	Draft	26/02/2013 12:31:58	<a href="#">Abrir</a>
<input type="checkbox"/>			sayHi-taskform	Draft	26/02/2013 12:30:58	<a href="#">Abrir</a>

1-5 of 5

#### 4.2. Definición de reglas.

Para crear una nueva regla de negocio no tenemos más que seleccionar la opción correspondiente del menú "Crear Nuevo"; pero más que crear una nueva regla de negocio vamos a reutilizar las reglas que ya creamos de la mano de mi compi Miguel en el primer tutorial de drools.

Para ello vamos a generar un jar con los objetos de negocio y los vamos a subir como un modelo de POJOs; tras seleccionar la opción correspondiente del menú se mostrará una interfaz como la que sigue en la que podremos asignar un nombre:

✕
Nuevo Modelos

**Nuevo Modelos**

Crear nuevo:  
 Importar activo del área global:

Nombre:

Paquete Create in: com.autentia.bpm.tutoriales  
 Crear en el área global

Descripción inicial:

Tras pulsar OK se mostrará una interfaz como la que sigue desde la que podremos importar el jar.

Archivo
Modificar

Atributos:
Modificar

**OrderEntities**

Subir nueva versión:

Descargar versión actual:

Este es un archivo JAR que contiene objetos POJO (del inglés Plain Old Java Objects) que definen un modelo de dominio.

Tras la importación tendremos en el listado de modelos una entrada como esta:

Modelos					
Formato	Válido	Nombre	Estado	Ultimo modificado	Abrir
☐	✔	OrderEntities	Draft	26/02/2013 08:39:23	<input type="button" value="Abrir"/>

1 of 1

Lo siguiente será, en vez de crear una regla desde cero, importar el contenido de la regla que ya teníamos, para ello crearemos un archivo simple con extensión drl, "Crear nuevo" > "Crear un archivo".

✕
Nuevo Otros recursos, documentación

**Nuevo Otros recursos, documentación**

Crear nuevo:  
 Importar activo del área global:

Nombre:

Extensión de archivo (tipo/formato):

Paquete Create in: com.autentia.bpm.tutoriales  
 Crear en el área global

Descripción inicial:

Al incluir la extensión drl, Guvnor entiende que es una regla de negocio y nos propone una pantalla para comenzar a trabajar, en la que podemos pegar el contenido textual de la regla que ya teníamos del [tutorial de Miguel](#).

```

Archivo  Modificar  Origen/Fuente  Estado: D
Atributos:  Modificar
Mostrar tipos de hechos
package com.autentia.bpm.tutoriales
import com.autentia.tutorial.drools.data.*;

// Sumamos el importe total de los productos
rule "Initial rule"
  salience 20
  when
    order : Order ();
    totalPrice : Double() from accumulate (
      Product( productPrice : price) from order.getProducts, |
      init (double total = 0);
      action (total += productPrice);
      result (new Double(total))
    );
  then
    order.setTotalPrice(totalPrice);
  end

// Comprobamos si el cliente es SILVER, si es así aplicamos un 5% de descuento
rule "SILVER customer rule"
  salience 15
  when
    order : Order ();
    customer : Customer ( status == Customer.SILVER_CUSTOMER) from order.getCustomer();
  then
    order.setTotalPrice(order.getTotalPrice() * (1 - (5 / 100d) ));
  end
    
```

Tras guardar tendremos en el listado de reglas técnicas una entrada como la que sigue:

Recurso de regla técnica						
Formato	Válido	Nombre	Estado	Ultimo modificado	Abrir	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	OrderRule	Draft	26/02/2013 01:39:39	<input type="button" value="Abrir"/>	

Con ello, ya tenemos contenido en nuestro repositorio que consumir.

### 5. Cómo hacer uso de recursos de guvnor.

Los recursos de Guvnor los podemos importar en el espacio de trabajo con el plugin de conexión a Guvnor que incorpora jBPM para Eclipse y hacer uso de una copia local de los mismos. Del mismo modo podemos incorporar a Guvnor recursos desde el espacio de trabajo de Eclipse al repositorio de contenidos.

Lo ideal, sobre todo para el proceso de promoción entre entornos, sería disponer de esos recursos versionados en un control de versiones y generar un empaquetado automatizado de los mismos para importarlos en Guvnor (sería la filosofía de los recursos amp de Alfresco).

En este punto, omitiendo dicha configuración, vamos a proponer consumir los recursos de Guvnor mediante la exposición de los mismos en un contexto de Spring. Guvnor permite crear contextos de Spring, accesibles vía url, publicando en el mismo el contenido de la última versión de un paquete de recursos. Con ello, el negocio de las reglas en Guvnor podría variar en el tiempo y para nuestras aplicaciones sería transparente.

Para crear un nuevo contexto de Spring no tenemos más que pulsar en "Crear Nuevo" > "Nuevo contexto de Spring" y se mostrará una ventana como esta:

**Nuevo SpringContext**

Crear nuevo:  
 Importar activo del área global:

Nombre:

Paquete Create in:   
 Crear en el área global

Descripción inicial:

Tras pulsar OK, nos redirige al área de edición del contenido del fichero desde la cuál podremos añadir configuraciones de paquetes desde el árbol inferior izquierdo:

Archivo Modificar Origen/Fuente Estado: 'Draft'

Atributos: Modificar

Palette

- KSession
- KBase
- Node
- KAgent
- Spring Bean
- Paquetes
  - globalArea
  - com.autentia.bpm
    - LATEST
    - defaultPackage

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:drools="http://drools.org/schema/drools-spring"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org
/schema/beans/spring-beans-2.5.xsd
  http://www.springframework.org/schema/context http://www.springframework.org/schema/context
  /spring-context-2.5.xsd
  http://drools.org/schema/drools-spring https://github.com/droolsjbpm/droolsjbpm/raw/master/drools-
container/drools-spring/src/main/resources/org/drools/container/spring/drools-spring-1.2.0.xsd"
  default-autowire="byName">
  <drools:kbase id="kbase1">
    <drools:resources>
      <drools:resource type="PKG" source="http://localhost:8080/drools-guvnor
/orig.drools.guvnor.Guvnor/package/com.autentia.bpm.tutoriales/LATEST" basicAuthentication="enabled"
username='admin' password='admin' />
    </drools:resources>
  </drools:kbase>
  <drools:ksession id="ksession1" type="stateful" kbase="kbase1"/>
</beans>
```

Lo que hará es añadir configuración al xml, en función de la selección y, cuando hayamos finalizado lo ideal es validar el contenido del xml con la opción "Origen/Fuente" > validar. En este punto podemos tener un error como el que sigue:

Validando resultados

Validando resultados

[applicationContext-order.xml] schema\_reference:4: Fallo al leer el documento de esquema 'https://github.com/droolsjbpm/droolsjbpm/raw/master/drools-container/drools-spring/src/main/resources/org/drools-container/spring/drools-spring-1.2.0.xsd': porque 1) no se ha encontrado el documento; 2) no se ha podido leer el documento; 3) el elemento raíz del documento no es <xsd:schema>.

Resulta que la xsd ha dejado de estar disponible en la URL que propone Guvnor. Buscando una solución podemos sustituir la ruta de la xsd de drools-spring por esta:

1 | <http://drools.org/schema/drools-spring> <http://anonsvn.jboss.org/repos/labs/labs/jbossru>

Y, con ello, tener el siguiente contenido:

Archivo Modificar Origen/Fuente Estado: 'Draft'

Atributos: Modificar

Palette

- KSession
- KBase
- Node
- KAgent
- Spring Bean
- Paquetes
  - globalArea
  - com.autentia.bpm
    - LATEST
    - defaultPackage

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:drools="http://drools.org/schema/drools-spring"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org
/schema/beans/spring-beans-2.5.xsd
  http://www.springframework.org/schema/context http://www.springframework.org/schema/context
  /spring-context-2.5.xsd
  http://drools.org/schema/drools-spring http://anonsvn.jboss.org/repos/labs/labs/jbossrules/trunk
/drools-container/drools-spring/src/main/resources/org/drools/container/spring/drools-spring-
1.2.0.xsd"
  default-autowire="byName">
  <drools:kbase id="kbase1">
    <drools:resources>
      <drools:resource type="PKG" source="http://localhost:8080/drools-guvnor
/orig.drools.guvnor.Guvnor/package/com.autentia.bpm.tutoriales/LATEST" username='admin'
password='admin' />
    </drools:resources>
  </drools:kbase>
  <drools:ksession id="ksession1" type="stateful" kbase="kbase1"/>
</beans>
```

Llegados a este punto, deberíamos disponer de un acceso público a la configuración de spring vía url; a nivel de paquete en la pestaña de "Modificar", deberíamos tener un contenido similar a este:

✓ Paquete construido correctamente. Tue Feb 26 01:35:34 GMT+100 2013

[Descargar paquete binario](#)

Tomar snapshot:

URL para la documentación del <http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/com.autentia.bpm.tutoriales/LATEST/documentation.pdf>

paquete: <http://localhost:8080/drools-guvnor/rest/packages/com.autentia.bpm.tutoriales/source>

URL de la fuente del paquete: <http://localhost:8080/drools-guvnor/rest/packages/com.autentia.bpm.tutoriales/binary>

URL del paquete binario: <http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/com.autentia.bpm.tutoriales/LATEST/SCENARIOS>

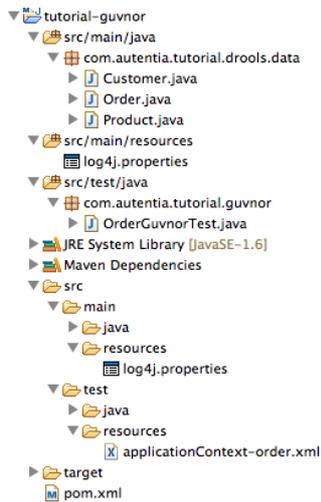
URL para correr pruebas: <http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/com.autentia.bpm.tutoriales/LATEST/ChangeSet.xml>

Change Set: <http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/com.autentia.bpm.tutoriales/LATEST/MODEL>

Modelos: <http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/com.autentia.bpm.tutoriales/LATEST/SpringContext/application-order.xml>

SpringContext:

Con ello, ya podríamos jugar a levantar el contexto de Spring con esta configuración para poder acceder a los recursos del paquete en Guvnor; si bien como nuestra intención es realizar un test de jUnit, la URL de acceso al contexto de Spring está securizada y no queremos aún configurar la autenticación y autorización en Guvnor, para realizar la prueba, en vez hacer uso de la url, vamos a tomcat el contenido del fichero y crear una copia local en la carpeta de recursos de test dentro de nuestro workspace.



El contenido del applicationContext-order.xml sería el siguiente:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xmlns:drools="http://drools.org/schema/drools-spring"
6     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springfra
7     http://www.springframework.org/schema/context http://www.springframework.org/schema/
8     http://drools.org/schema/drools-spring http://anonsvn.jboss.org/repos/labs/labs/jbos
9     default-autowire="byName">
10
11     <drools:kbase id="kbase1">
12
13         <drools:resources>
14
15             <drools:resource type="PKG" source="http://localhost:8080/drools-guvnor/org.droc
16                 basic-authentication="enabled"
17                 username='admin' password='admin' />
18
19         </drools:resources>
20
21     </drools:kbase>
22
23
24     <drools:ksession id="ksession" type="stateful" kbase="kbase1" />
25
26 </beans>
27

```

Y vamos a escribir un test como el que sigue, con el soporte de Spring, para arrancar un proceso de negocio:

```

1 package com.autentia.tutorial.guvnor;
2
3 import static org.junit.Assert.assertEquals;
4
5 import javax.annotation.Resource;
6
7 import org.drools.runtime.StatefulKnowledgeSession;
8 import org.drools.runtime.process.ProcessInstance;
9 import org.junit.Test;
10 import org.junit.runner.RunWith;
11 import org.springframework.test.context.ContextConfiguration;
12 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
13
14 @RunWith(SpringJUnit4ClassRunner.class)
15 @ContextConfiguration({"classpath:applicationContext-order.xml"})
16 public class OrderGuvnorTest {
17
18     @Resource
19     private StatefulKnowledgeSession ksession;
20
21     @Test
22     public void shouldStartGuvnorProcess() {
23         final ProcessInstance processInstance = ksession.startProcess("sayHi");
24         assertEquals(ProcessInstance.STATE_COMPLETED, processInstance.getState());
25     }
26
27 }

```

La referencia a la factoría de sesiones la obtenemos por inyección con el soporte de Spring y podemos arrancar un proceso por id.

Siguiendo el código del test de reglas que ya teníamos del primer tutorial podríamos escribir un segundo test con el siguiente código:

```

1 package com.autentia.tutorial.guvnor;
2
3 import java.util.Arrays;
4 import java.util.List;
5
6 import javax.annotation.Resource;
7
8 import org.drools.runtime.StatefulKnowledgeSession;

```

```

 9  import org.junit.Test;
10  import org.junit.runner.RunWith;
11  import org.springframework.test.context.ContextConfiguration;
12  import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
13
14  import com.autentia.tutorial.drools.data.Customer;
15  import com.autentia.tutorial.drools.data.Order;
16  import com.autentia.tutorial.drools.data.Product;
17
18  @RunWith(SpringJUnit4ClassRunner.class)
19  @ContextConfiguration({"classpath:applicationContext-order.xml"})
20  public class OrderGuvnorTest {
21
22      @Resource
23      private StatefulKnowledgeSession ksession;
24
25
26      @Test
27      public void shouldUseGuvnorRules() {
28
29          final List<Order> orders = Arrays.asList(getOrderWithDefaultCustomer(), getOrder
30              getOrderWithGoldCustomer(), getOrderWithGoldCustomerAndTenProducts());
31          for (Order order : orders) {
32              ksession.insert(order);
33          }
34          ksession.fireAllRules();
35
36          showResults(orders);
37      }
38
39      private static Order getOrderWithDefaultCustomer() {
40          final Order order = new Order(getDefaultCustomer());
41          order.addProduct(getProduct1());
42          return order;
43      }
44
45      private static Order getOrderWithSilverCustomer() {
46          final Order order = new Order(getSilverCustomer());
47          order.addProduct(getProduct1());
48          return order;
49      }
50
51      private static Order getOrderWithGoldCustomer() {
52          final Order order = new Order(getGoldCustomer());
53          order.addProduct(getProduct1());
54          return order;
55      }
56
57      private static Order getOrderWithGoldCustomerAndTenProducts() {
58          final Order order = new Order(getSilverCustomer());
59          for (int i = 0; i < 10; i++) {
60              order.addProduct(getProduct1());
61          }
62          return order;
63      }
64
65      private static Customer getDefaultCustomer() {
66          return new Customer(Customer.DEFAULT_CUSTOMER, "Cliente estandar");
67      }
68
69      private static Customer getSilverCustomer() {
70          return new Customer(Customer.SILVER_CUSTOMER, "Cliente SILVER");
71      }
72
73      private static Customer getGoldCustomer() {
74          return new Customer(Customer.GOLD_CUSTOMER, "Cliente GOLD");
75      }
76
77      private static Product getProduct1() {
78          return new Product(1, "Producto 1", 100d);
79      }
80
81      private static void showResults(List<Order> orders) {
82          for (Order order : orders) {
83              System.out.println("Cliente " + order.getCustomer() + " productos: " + order
84                  + " Precio total: " + order.getTotalPrice());
85          }
86      }
87  }

```

Lo único que varía es la obtención de la factoría mediante el soporte de inyección de Spring, que además es in singleton.

Podéis descargar aquí los fuentes usados en el tutorial. Se trata de un fichero comprimido con dos proyectos maven:

- tnt-jbpm: el proyecto parent que permite desplegar guvnor y el designer, y
- tutorial-guvnor: el proyecto que contiene los tests.

## 6. Referencias.

- <http://docs.jboss.org/drools/release/5.4.0.Final/drools-guvnor-docs/pdf/guvnor-docs.pdf>
- <http://kverlaen.blogspot.com.es/2011/10/introducing-service-repository.html>
- <http://vimeo.com/28473009>
- <http://vimeo.com/21445110>

## 7. Conclusiones.

Aún nos quedan cuestiones por resolver en relación a Guvnor, sobre todo:

- el tema de la autenticación y autorización de usuarios,
- la parametrización de la comunicación entre el designer y guvnor, para que no esté acoplado con el dominio local, y
- el versionado y la promoción entre entornos de los recursos.

Pero ya podemos pensar en él como una buena solución para la gestión centralizada de reglas y de procesos dentro de nuestro ecosistema BPM, con jBPM.

Un saludo.

Jose

[jmsanchez@autentia.com](mailto:jmsanchez@autentia.com)

### A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)



### Por favor, vota +1 o compártelo si te pareció interesante



Ánimate y coméntanos lo que pienses sobre este **TUTORIAL**:

» [Regístrate](#) y accede a esta y otras ventajas «



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)