

¿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)

AdictosAlTrabajo

Terrakas 1x03
¡¡Ya está en la web!!
terrakas.com



autentia
Soporte a desarrollo informático
Hosting patrocinado por
enredados

Entra en Adictos a través de  

E-mail

Contraseña

Entrar [Deseo registrarme](#)
[Olvidé mi contraseña](#)

[Inicio](#) [Quiénes somos](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#) [Más](#)

» Estás en: [Inicio](#) [Tutoriales](#) Revisión de jBPM5



Roberto Canales Mora

Creador y propietario de AdictosAlTrabajo.com, Director General de Autentia S.L., Ingeniero Técnico de Telecomunicaciones y Executive MBA por el Instituto de Empresa 2007.

Twitter: [Seguir a @rcanalesmora](#) 928 seguidores

Autor del Libro: Informática profesional, las reglas no escritas para triunfar en la empresa

Puedes consultar mi CV y alguna de mis primeras aplicaciones (de los 90) aquí



[Ver todos los tutoriales del autor](#)



Fecha de publicación del tutorial: 2009-02-26

Tutorial visitado 2 veces [Descargar en PDF](#)

Revisión de jBPM5

En los últimos días en Autentia ya acumulamos varias peticiones de proyecto basadas de jBPM y, aunque ya lo probé hace tiempo y usamos distintos motores BPM, me he decido a cacharrear personalmente en el estado del arte de la versión 5. El dirigir una empresa habiendo sido un friki técnico hay veces que produce mono ...

Si queréis entender un poquito de que va el BMP os recomiendo que miréis antes este articullilo.

Introducción a BPM
brought to you by Livescribe

jBPM nos puede servir como motor de Workflow interno para nuestras aplicaciones pero ya os adelanto que cualquier proyecto llevará muchas tareas de configuración y mejora (aunque eso pasa con cualquier producto más allá de una demo de preventa).

En este artículo solamente pretendo identificar los recursos disponibles e interpretarlos intentando que este trabajo le valga también al que se quiera iniciar.

Os paso las especificaciones de mi Mac, en las que va bastante bien:

Catálogo de servicios Autentia



Síguenos a través de:



Últimas Noticias

- » [Autentia colabora con la ONG Proyecto Ciclista Solidario](#)
- » [Curso de Kanban Core en Madrid con Masa K. Maeda](#)
- » [Failure demand](#)
- » [Comentando el Libro: Lean StartUp \(el método\)](#)
- » [VIII Autentia Cycling Day](#)

[Histórico de noticias](#)

Últimos Tutoriales

- » [Database MessageSource: obtener los literales de una base de datos](#)
- » [Comparando diferencias entre ficheros con java-diff-utils](#)
- » [Mejorando la calidad de nuestro código PHP](#)
- » [jQuery Waypoints: realizando acciones al llegar a un punto de la página con el scroll.](#)
- » [Invocar a procedimientos almacenados de Oracle usando Spring JDBC](#)

Últimos Tutoriales del

Impulsores Comunidad ¿Ayuda?

0 personas han traído clicks a esta página

sin clicks + + + + + + + +

powered by [karmacracy](#)



Nos vamos al Web de jboss a por el proyecto jBPM :

What does jBPM do?

A business process allows you to model your business goals by describing the steps that need to be executed to achieve that goal and the order, using a flow chart. This greatly improves the visibility and agility of your business logic, results in higher-level and domain-specific representations that can be understood by business users and is easier to monitor.

The core of jBPM is a light-weight, extensible workflow engine written in pure Java that allows you to execute business processes using the latest BPMN 2.0 specification. It can run in any Java environment, embedded in your application or as a service.

On top of the core engine, a lot of features and tools are offered to support business processes throughout their entire life cycle:

- Eclipse-based and web-based editor to support the graphical creation of your business processes (drag & drop)
- Pluggable persistence and transactions based on JPA / JTA
- Pluggable human task service based on WS-HumanTask for including tasks that need to be performed by human actors
- Management console supporting process instance management, task lists and task form management, and reporting
- Optional process repository to deploy your process (and other related knowledge)

Nos descargamos el instalador completo desde este enlace.

Son más de 400 Megs ... con la nueva fibra óptica en casa ... segundos, que maravilla.

Autor

- » Framework Scala liftweb
- » Primeros pasos con Scala
- » Dividir tu pantalla gigante en Mac con Divvy
- » Cómo alcanzar el éxito en el sector de la informática.
- » PMBOK (Project Management Body of Knowledge) v4.0

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.

Roberto Canales Mora
[rcanalesmora](#)

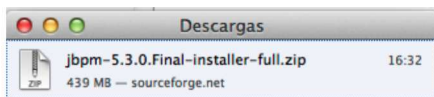
[rcanalesmora](#) Nuevos regalitos de Autentia ya en la ofi :-)) los frikis dominaremos el mundo :-p [pic.twitter.com/4JFdqHqZ](#)
4 hours ago · reply · retweet · favorite

[rcanalesmora](#) Terminado tutorial de jBPM5. Con mono de tocar código.. pero todavía tengo que acabar el segundo libro. Mañana lo subimos.
20 hours ago · reply · retweet · favorite

[rcanalesmora](#) "@diegomunozdiez: Genial conferencia en el IE sobre metodologías ágiles por parte de @rcanalesmora" muchas gracias, un placer
2 days ago · reply · retweet · favorite

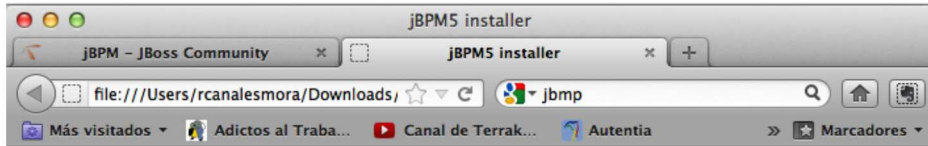
[rcanalesmora](#) Scrum vs Scrum but.. reciclando frase grandiosa que leí por

Join the conversation



Como siempre PFLLPD (Por Favor Leerse La P... Documentación) donde explican muy bien lo que hay que hacer. Es que normalmente vamos acelerados y queremos hacer las cosas intuitivamente.

Vemos que para que funcione correctamente el instalador hace falta una versión de Java superior a 1.5 y ant superior a 1.7.



Installation

This guide will assist you in installing and running a demo setup of the various components of the jBPM project. If you have any feedback on how to improve this guide, if you encounter problems, or if you want to help out, do not hesitate to contact the jBPM community as described in the "What to do if I encounter problems or have questions?" section.

Prerequisites

This script assumes you have Java JDK 1.5+ (set as JAVA_HOME), and Ant 1.7+ installed. If you don't, use the following links to download and install them:

Java: <http://java.sun.com/javase/downloads/index.jsp>

Ant: <http://ant.apache.org/bindownload.cgi>

Download the installer

First of all, you need to download the installer:

jbpm-{version}-installer.zip

You can for example find the latest release [here](#) or the latest snapshot release [here](#).

Demo setup

The easiest way to get started is to simply run the installation script to install the demo setup. Simply go into the install folder and run:

ant install.demo

This will:

- Download JBoss AS
- Download Eclipse
- Install Drools Guvnor into JBoss AS
- Install jBPM Designer into JBoss AS
- Install the jBPM Console into JBoss AS
- Install the jBPM Eclipse plugin
- Install the Drools Eclipse plugin



Primero verificamos que tenemos la versión de Java correcta. Abrimos un terminal.

```
javMacBookPro2RCanales:~ rcanalesmora$ java -version
java version "1.6.0_33"
Java(TM) SE Runtime Environment (build 1.6.0_33-b03-424-11M3720)
Java HotSpot(TM) 64-Bit Server VM (build 20.8-b03-424, mixed mode)
```

Java correcto: Ahora, que tenemos la versión de ant.

```
MacBookPro2RCanales:~ rcanalesmora$ ant -version
Apache Ant(TM) version 1.8.2 compiled on June 3 2011
MacBookPro2RCanales:~ rcanalesmora$
```

Ant correcto. Con esto parece que podemos seguir.

Ejecutamos el comando de instalación: ant install.demo

```
MacBookPro2RCanales:jbpmadictos rcanalesmora$ ant install.demo
Buildfile: /Users/rcanalesmora/jbpmadictos/build.xml
download.jboss.check: [echo] Checking JBoss AS download ...
```

```
download.jboss:
```

```
install.jboss: [unzip] Expanding: /Users/rcanalesmora/jbpmadictos/lib/jboss-as-7.0.2.Final.zip into /Users/rcanalesmora/jbpmadictos
download.drools.guvnor.check: [echo] Checking Drools Guvnor download ...
download.drools.guvnor:
```

Búscanos en Facebook

facebook



Roberto Canales
en Facebook

Me gusta

A 93 personas les gusta **Roberto Canales en Facebook**.



Jondarru



Sebastián



Jaime



JC



Cristián



Rodrigo Ant.



lodia y me



Madoño Mad.



Hector



Jona Pamon

Plug-in social de Facebook

```
check.jboss.version:
```

```
install.guvnor.into.jboss:
```

```
[mkdir] Created dir: /Users/rcanalesmora/jbpmadictos/target
```

```
[unzip] Expanding: /Users/rcanalesmora/jbpmadictos/lib/guvnor-distribution-wars-5.4.0-20120516.war into /Users/rcanalesmora/jbpmadictos/target
```

OJO: leer los mensajes de la consola. A mi me ha dado errores y he tenido que invocar varias veces el comando porque las descargas se quedaban incompletas (de eclipse) y daba error de extracción.

Reintentamos hasta que obtenemos el mensaje de que todo ha ido bien

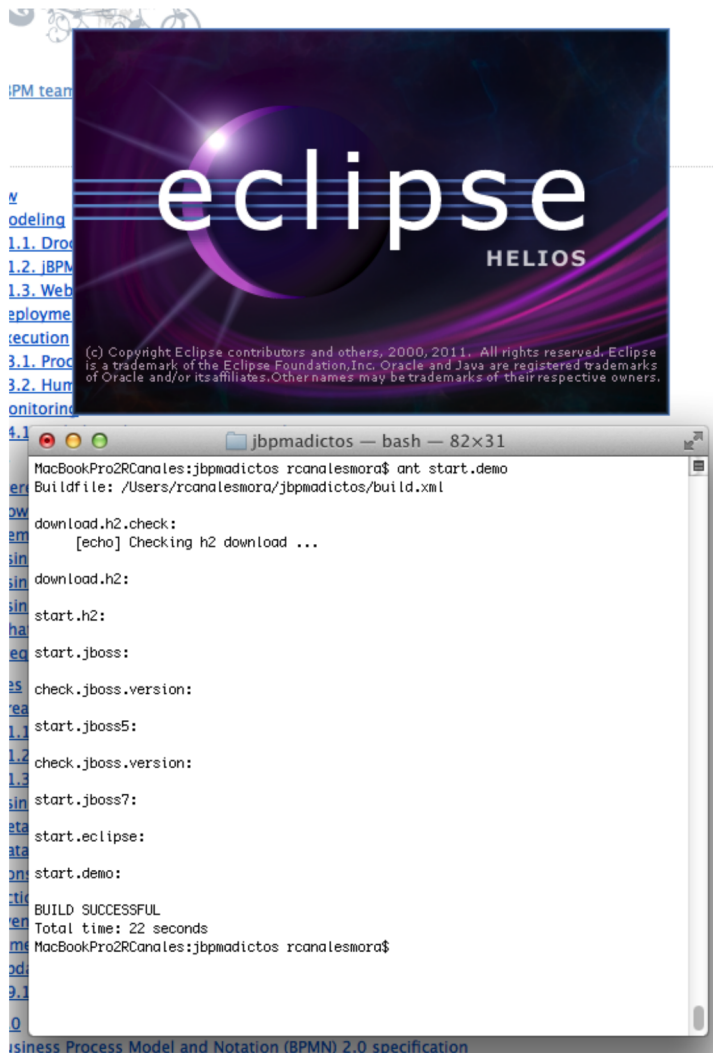
```
install.demo:
```

```
BUILD SUCCESSFUL
```

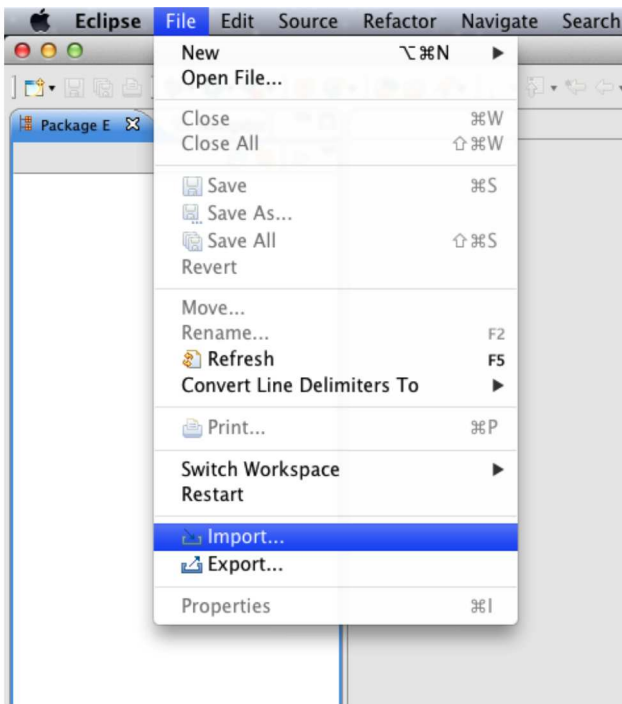
```
Total time: 5 minutes 18 seconds
```

```
MacBookPro2RCanales:jbpmadictos rcanalesmora$
```

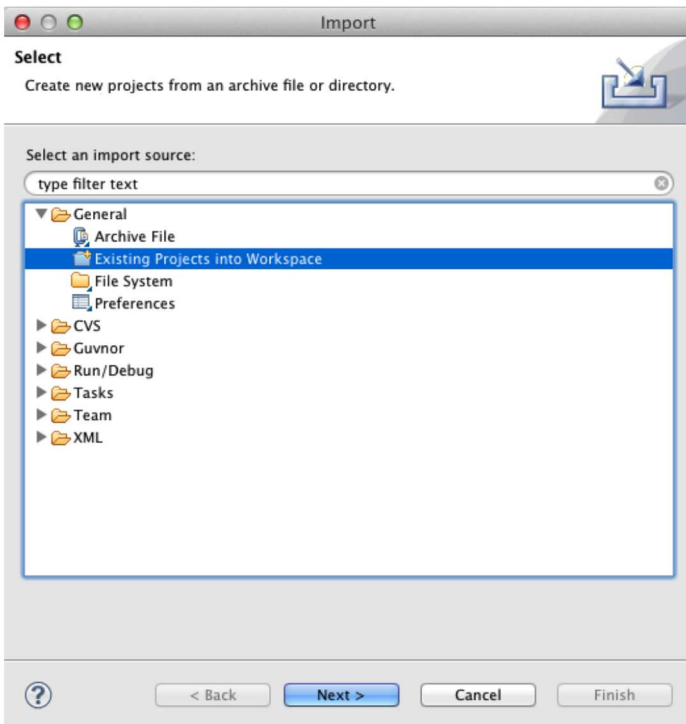
Se nos instalan todos los componentes. Ahora solo tenemos que lanzar todos los servicios y eclipse ... con el comando **ant start.demo**



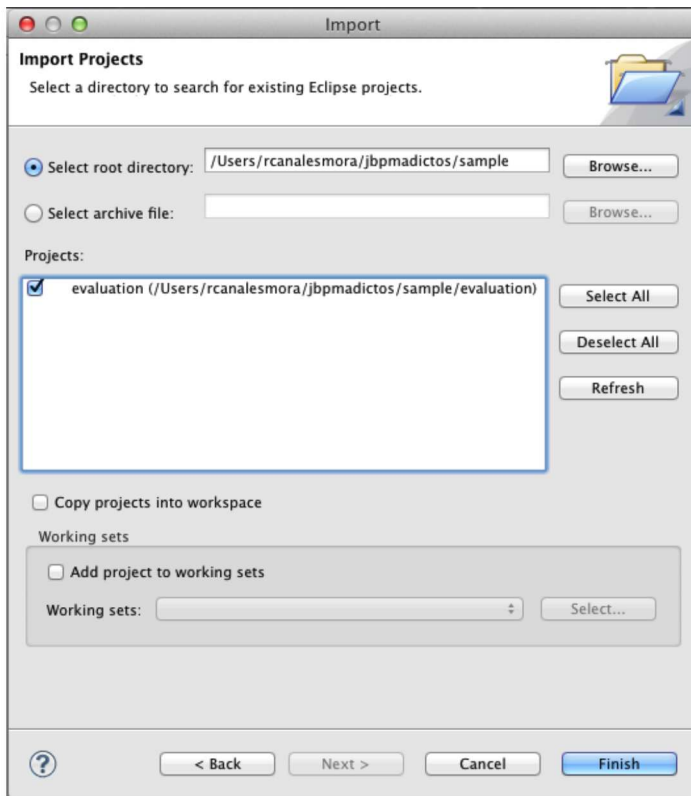
Se nos arranca automáticamente eclipse. Vamos a importar el proyecto ejemplo y tratar de identificar algunas cosas importantes. Importamos el proyecto ejemplo: Damos a File->Import



... elegimos proyectos existentes



Vamos al directorio donde hemos descomprimido los ficheros descargados y elegimos **sample**

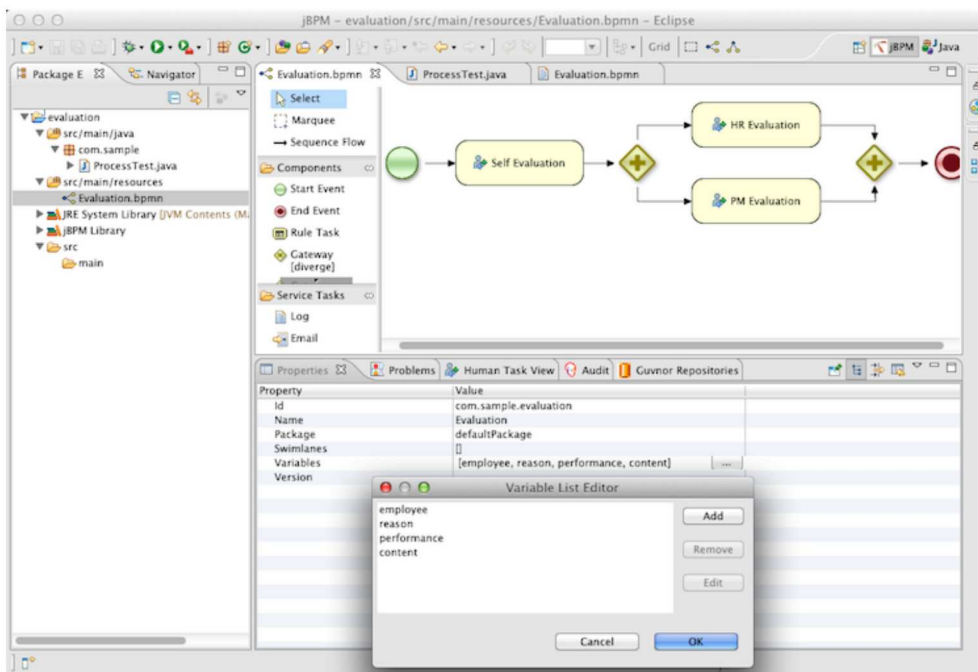


Ya tenemos en pantalla el proyecto que consta de dos ficheros: el fuente java y el diagrama BPMN 2.0.

Si hacemos doble clic sobre el diagrama nos aparecerá el editor gráfico. Sin seleccionar sobre ningún elemento, podemos ver las propiedades del diagrama/proceso BPM que estamos modelando.

Nos interesa el identificador: **com.sample.evaluation**

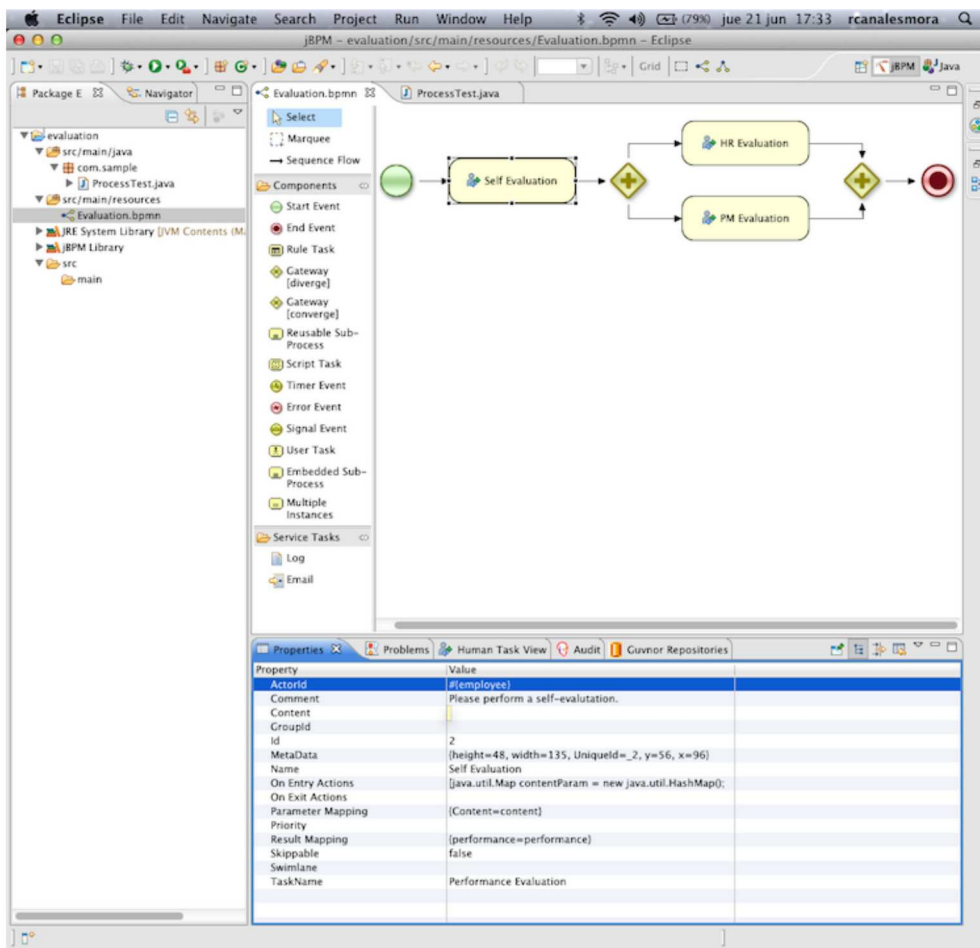
Un proceso arrastra variables que se utilizarán a lo largo de los mismos, es lo que se denomina en algunas herramientas UDAs (atributos definidos por el usuario). Vemos que están definidas 4.



Sobre cada una de las tareas del proceso podemos ver el **ActorId** que será el usuario que recibirá la tarea una vez arrancado el proceso y llegado a ese punto.

Puede estar asignado directamente como: **mary** (es decir, el usuario que recibirá la tarea directamente), o hacer referencia a un UDA: **#{employee}** (es decir, que se asignará a uno u otro dependiendo de un atributo arrastrado en el proceso)

En este segundo caso, en la tareas de **Self Evaluation**, se asignará al valor actual de la variable.



Podemos ver el código que crea una sesión contra el motor jBPM y crea una copia del proceso.

The screenshot shows the Eclipse IDE with the 'ProcessTest.java' file open. The code is a Java class that demonstrates how to create a jBPM session and start a process. The code is as follows:

```
package com.sample;

import java.util.HashMap;

/**
 * This is a sample file to launch a process.
 */
public class ProcessTest {

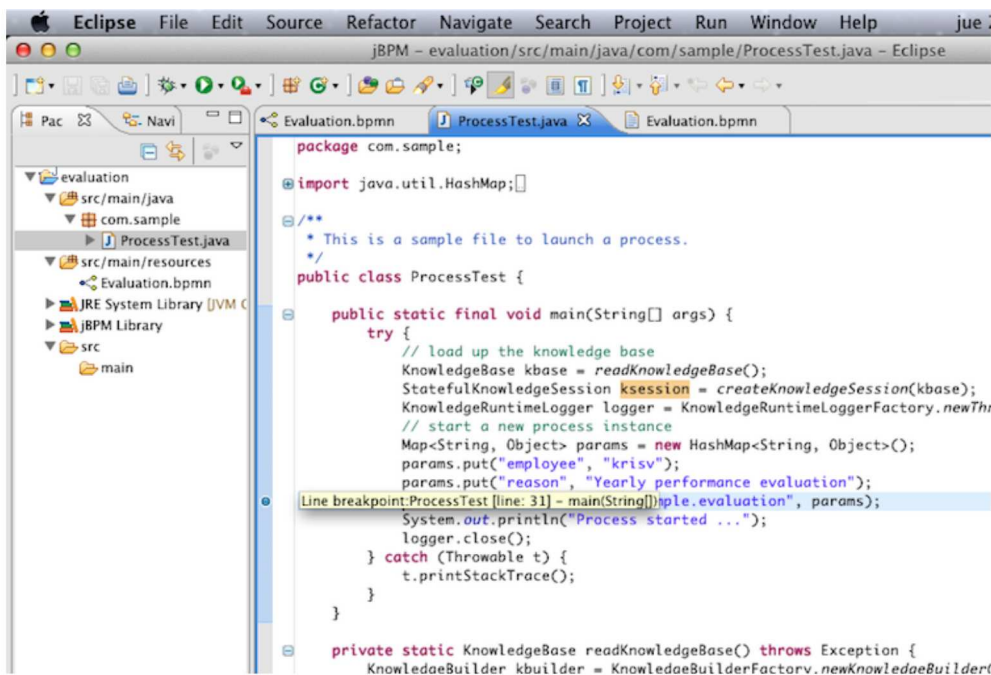
    public static final void main(String[] args) {
        try {
            // load up the knowledge base
            KnowledgeBase kbase = readKnowledgeBase();
            StatefulKnowledgeSession ksession = createKnowledgeSession(kbase);
            KnowledgeRuntimeLogger logger = KnowledgeRuntimeLoggerFactory.newThreadedFileLogger(ksession, "t");
            // start a new process instance
            Map<String, Object> params = new HashMap<String, Object>();
            params.put("employee", "krisv");
            params.put("reason", "Yearly performance evaluation");
            ksession.startProcess("com.sample.evaluation", params);
            System.out.println("Process started ...");
            logger.close();
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }

    private static KnowledgeBase readKnowledgeBase() throws Exception {
        KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
        kbuilder.add(ResourceFactory.newClassPathResource("Evaluation.bpmn"), ResourceType.BPMN2);
        return kbuilder.newKnowledgeBase();
    }

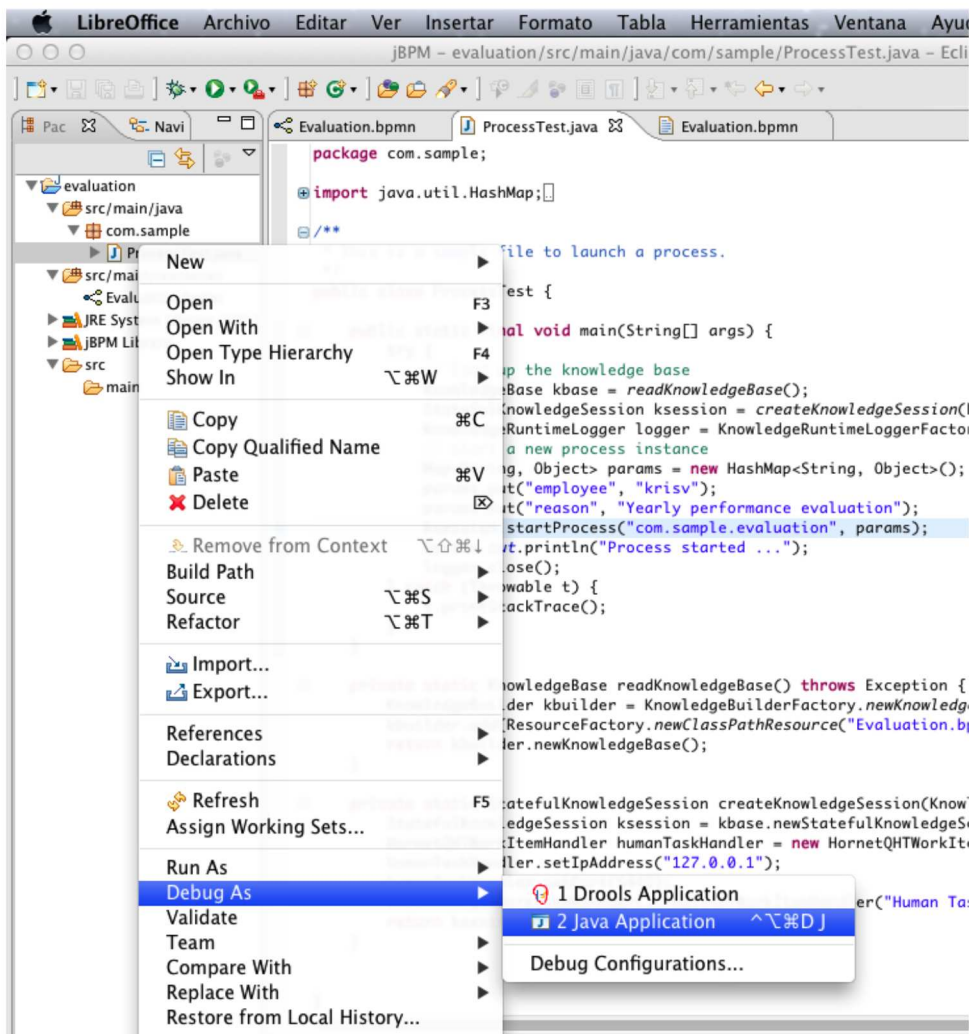
    private static StatefulKnowledgeSession createKnowledgeSession(KnowledgeBase kbase) {
        StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();
        HornetQHTWorkItemHandler humanTaskHandler = new HornetQHTWorkItemHandler(ksession);
        humanTaskHandler.setIpAddress("127.0.0.1");
        humanTaskHandler.setPort(5445);
        ksession.getWorkItemManager().registerWorkItemHandler("Human Task", humanTaskHandler);
        return ksession;
    }
}
```

Podemos arrancar la aplicación pero, para tener una sesión válida contra el motor de jBPM, vamos a poner un punto de parada en el código java y usar la perspectiva y vista especial que está integrada en eclipse para ver lo que está pasando en el motor.

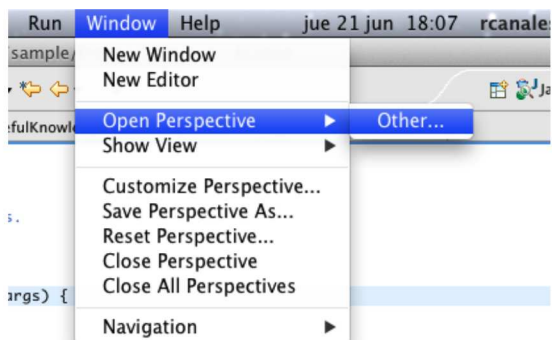
Pinchando en la barra lateral izquierda, al lado del código, aparecerá un punto de parada (bola azul).



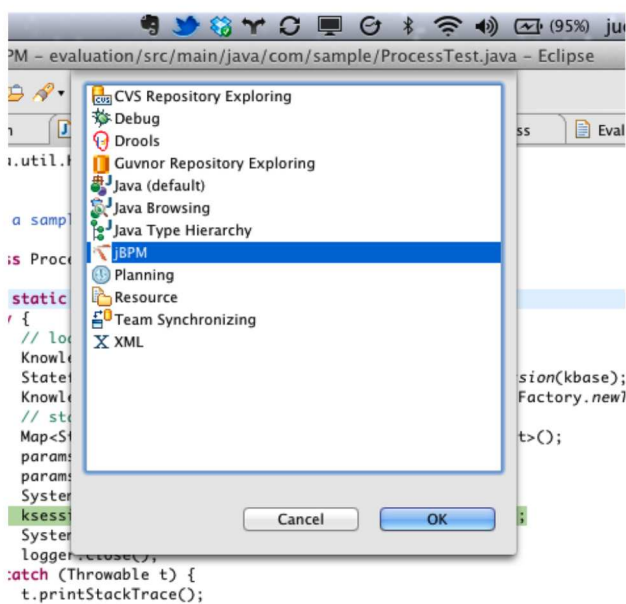
Ahora pulsamos el botón derecho sobre **ProcessTest.java** y elegimos la opción sobre **debug as -> java application**



Y ya hemos arrancado hasta el punto de parada. Nos aparecerá una pantalla diciendo que si queremos ir a la perspectiva de depuración. Lo podemos hacer ahora o cambiar cuando queramos de perspectiva. Vemos como se hace:



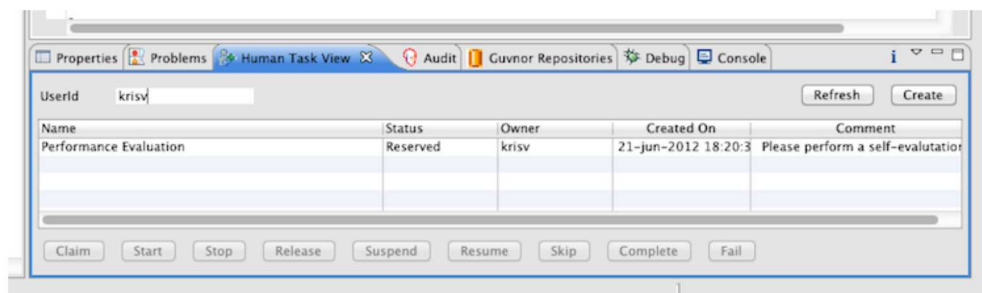
Yo elijo la perspectiva jBPM



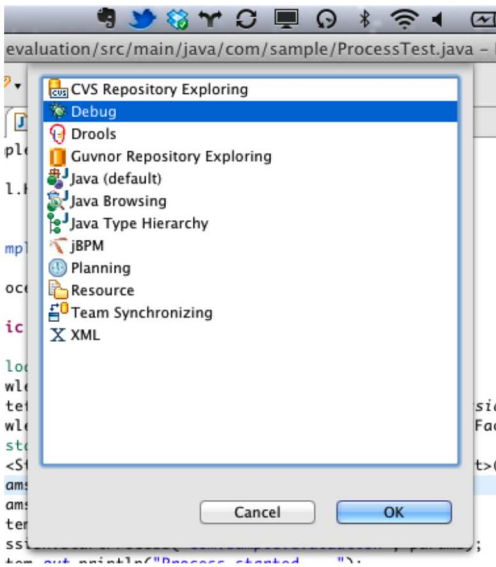
Y aparece una vista especial llamada **Human Task View** donde podemos interrogar al motor de jBPM (recordad que con una sesión válida activa) sobre las bandejas de tareas de cada usuario.

Podemos ver, en el código java, que podemos arrancar un proceso con el nombre de actor que queramos:

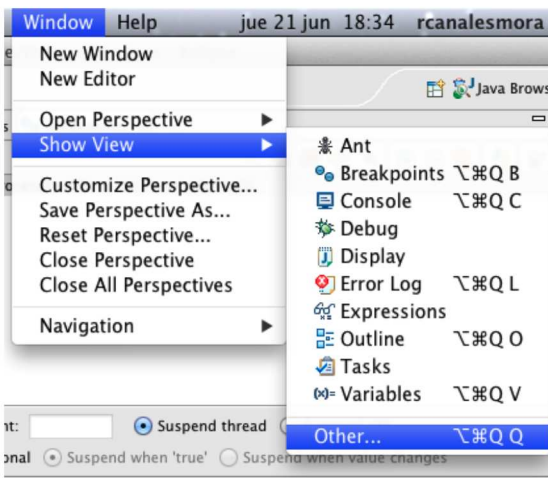
```
view plain print ?
01. params.put("employee", "Roberto");
02. params.put("reason", "Yearly performance evaluation");
```



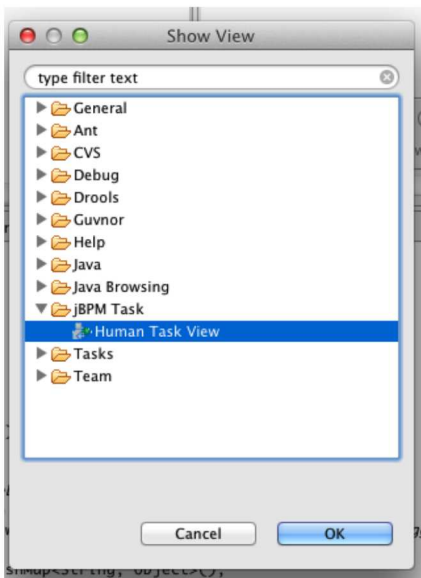
Si estamos en la perspectiva de depuración



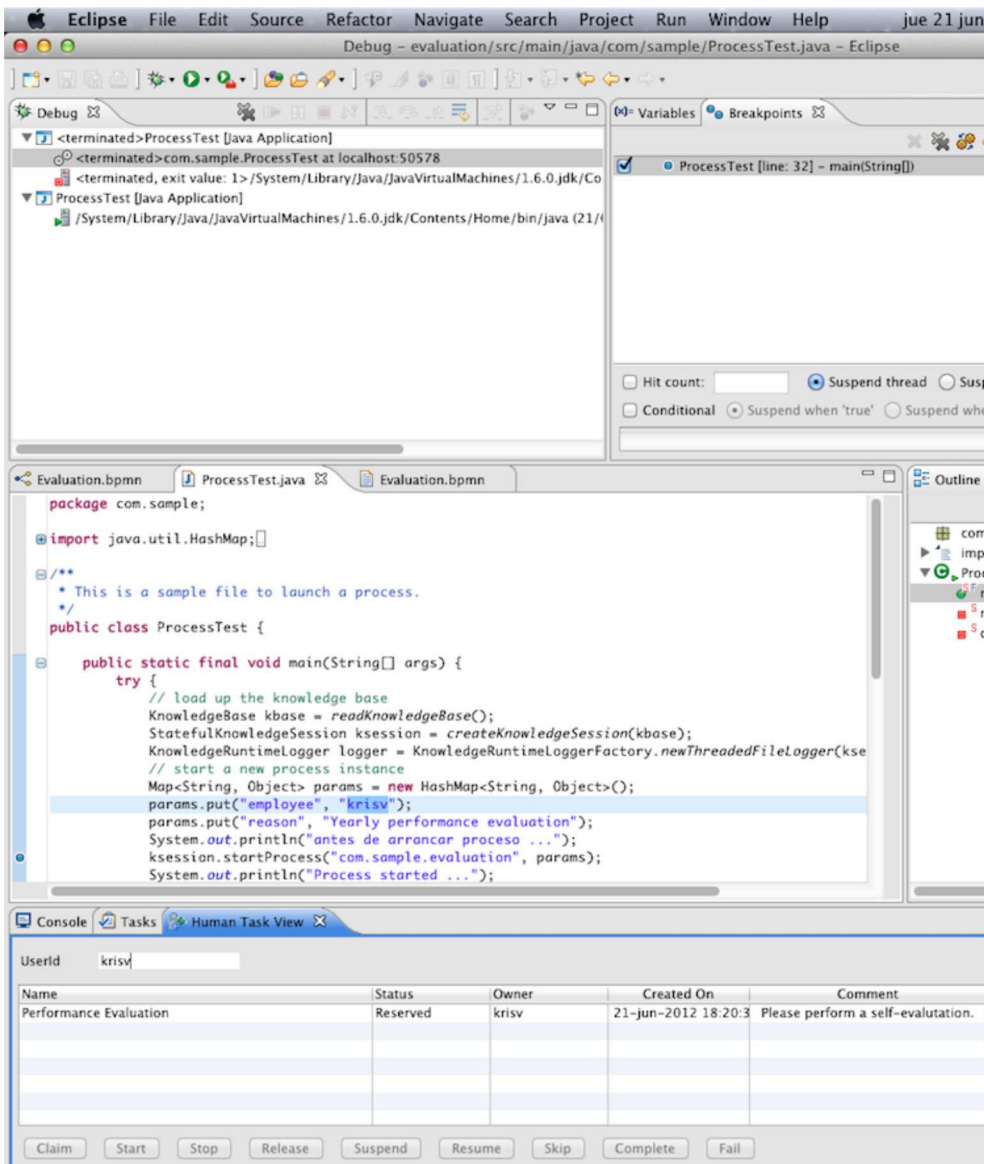
Podremos añadir la vista **Human Task View** manualmente.



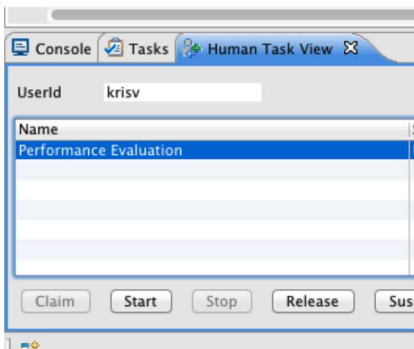
Y aparecerá junto a la de propiedades



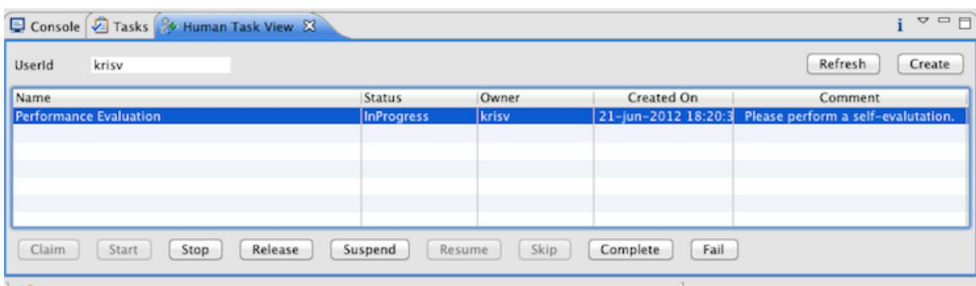
Este es el aspecto de nuestra perspectiva de depuración



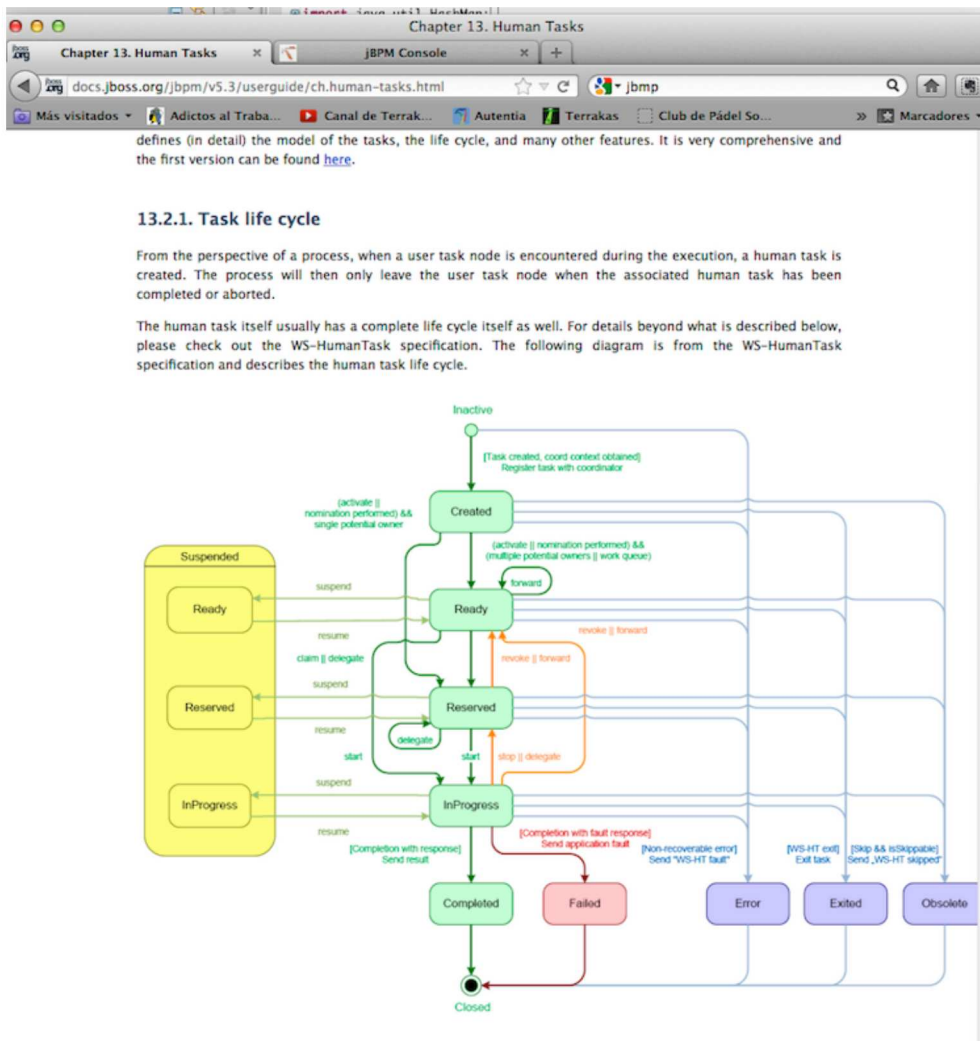
Cuando creamos una copia del proceso (uso esta nomenclatura para diferenciar lo que es la plantilla del proceso, de una instancia concreta) esta activa, podemos interrogar al sistema sobre las tareas disponibles para un usuario. El ejemplo por defecto se las asigna a **krisv**



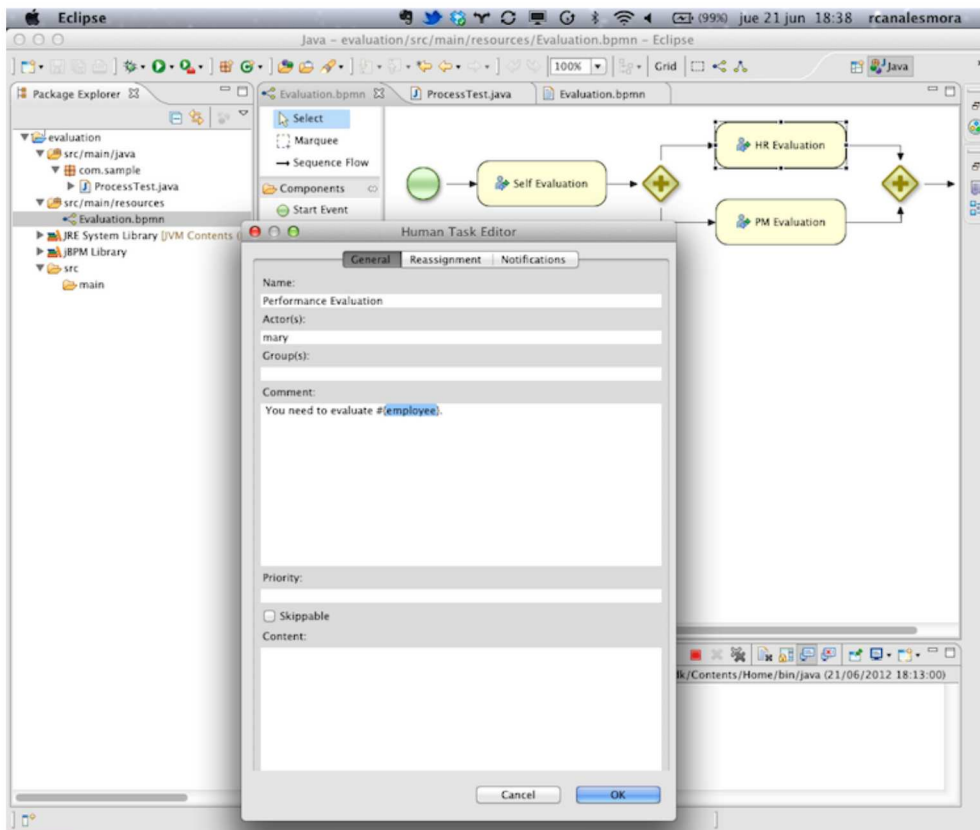
Deberemos entender que una tarea con intervención humana tiene un ciclo de vida. Primero aceptarla, arrancándola, podríamos escalarla, suspenderla, completarla o declararla como fallida.



Os recomiendo revisar la documentación para que veáis los posibles estados.

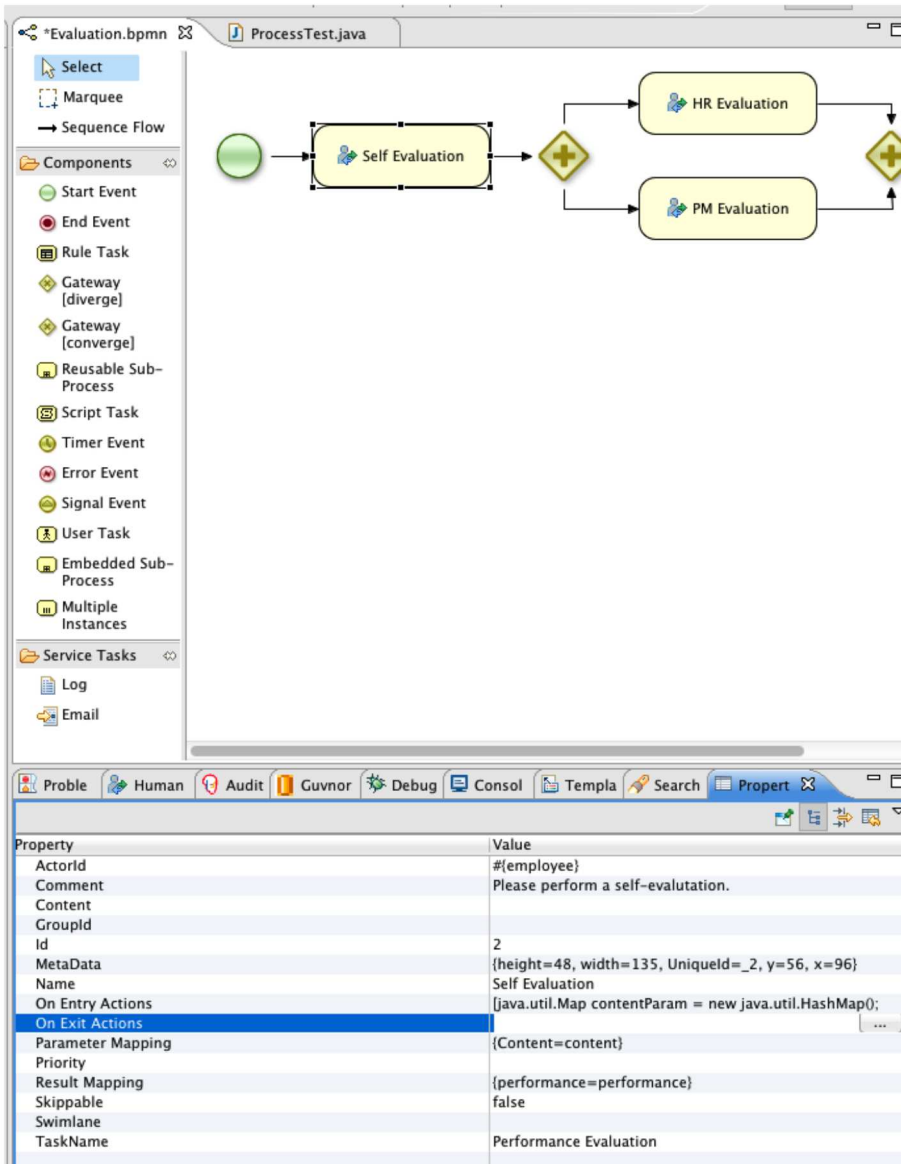


Una vez que nuestro actor **krisv** ha completado su tarea, esta estaría en la bandeja tanto de **john** como de **mary**, porque el proceso se ha bifurcado. Podemos verlo en las propiedades dentro del diagrama.



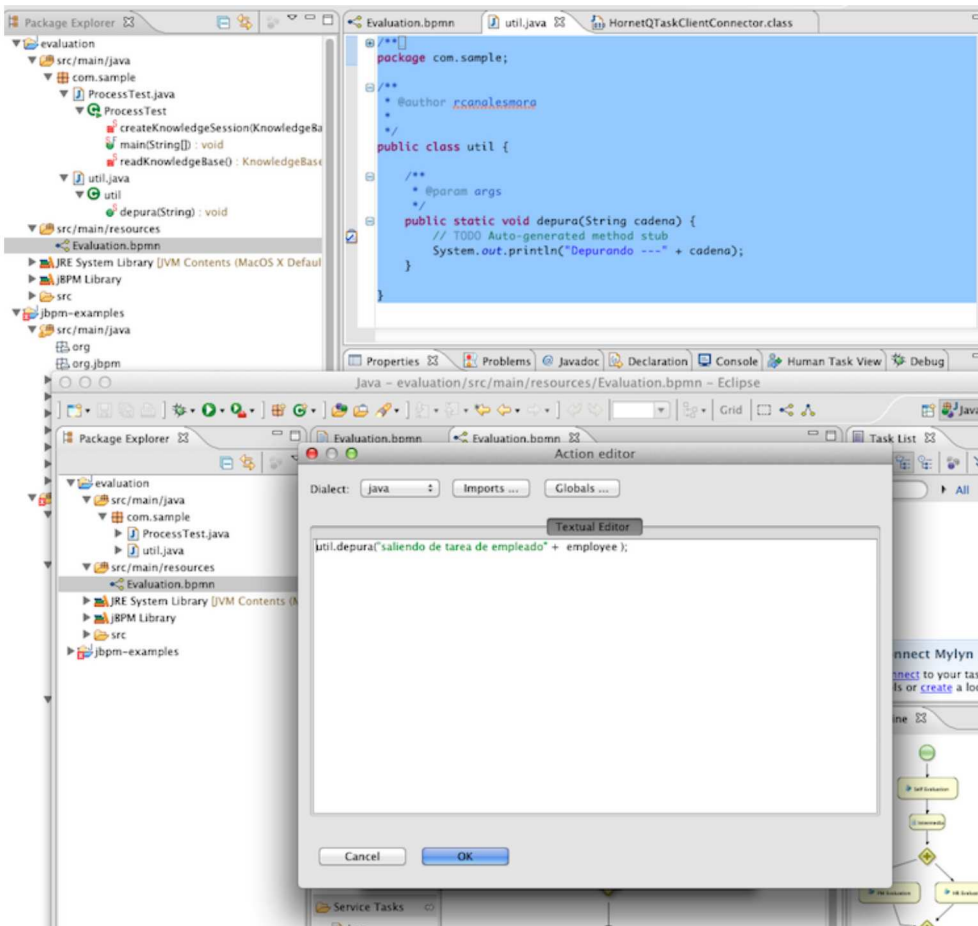
El proceso en si mismo simplemente va avanzando, por lo que es tirando a poco útil. Habría que ejecutar en cada paso alguna acción, como guardar los datos, actualizar alguna tabla, comunicarnos con un proceso externo, etc. La idea es que el flujo BPM invoque a servicios externos reutilizables y que dentro del motor se mantenga el mínimo de lógica posible.

Esta llamada a código o invocación a servicios la podríamos hacer de varios modos, siendo el más sencillo añadiendo código directamente a las acciones OnExit u OnEntry que se ejecutan cuando se entra o sale respectivamente del estado. Podemos ver en las propiedades el área para editar.



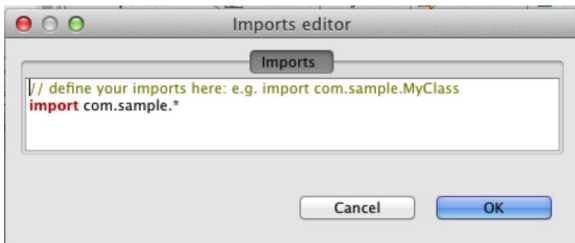
Haremos algo sencillo como poner un mensaje. Si podemos hacer esto podremos hacer cualquier cosa. En esto otras herramientas BPM proporcionan muchas más ayudas visuales permitiendo mapear datos a bases de datos, acceder a servicios Web y mapeando UDAs y datos de servicio visualmente, etc. Eso en principio parece una gran ventaja pero ... si queremos tener una lógica lo menos acoplada al motor de BPM en sí, tampoco parece una gran desventaja. Eso sí, es más laborioso.

Añadir código es sencillo aunque hay que tener un par de precauciones. Pinchamos en los puntos suspensivos de la acción:

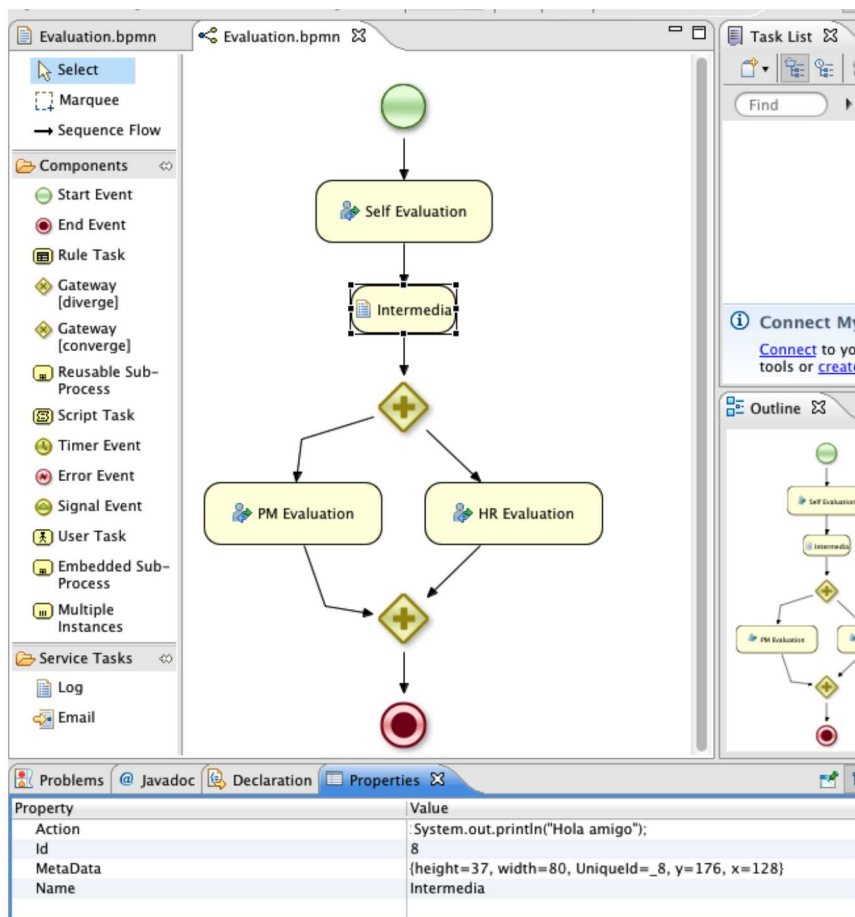


Para que funcione, no se os olvide indicar que el lenguaje es java (porque por defecto aparece MVEL).

Tampoco se os olvide, en la sección imports, declarar las librerías a importar. Como hemos creado una clase util, debemos declarar donde está.



También, para el mismo fin, podemos crear tareas intermedias de Script.



Hasta ahora hemos visto cómo se pinta un proceso donde hay tareas a realizar por un usuario, cómo invocar nuestro código y cómo ver lo que está pasando dentro del motor pero ¿cómo integramos esto con nuestras aplicaciones?

En un entorno real, pongamos en un portal Web, un usuario externo entraría en una página pública y tras hacer algo (pongamos un pedido) el portal arrancaría un proceso contra el motor de jBPM aportando los datos (tal como hemos visto en el código Java) y, posteriormente, usuarios internos de la organización irían recibiendo tareas. Pongamos que una persona tiene que mirar si el pedido es válido y que pase a almacén. ¿cómo y donde hace esto?

Otra vez hay varios modos. Un modo sencillo, sería disponer de una aplicación genérica que permita logarse a esos usuarios internos, ver las tareas que tienen en el motor de Workflow y, desde ese mismo punto, asignárselas, arrancarlas, que le apareciera una pantalla para ver la información (solo la necesaria) del proceso y que pudiera aportar nuevos datos y darla por completada. En nuestro ejemplo del pedido, podría ver quien lo pide, el nivel de crédito consultando el histórico de compras, la página web del cliente o comprando algún informe de riesgo, etc. y dejar pasar el pedido o matarlo en ese punto.

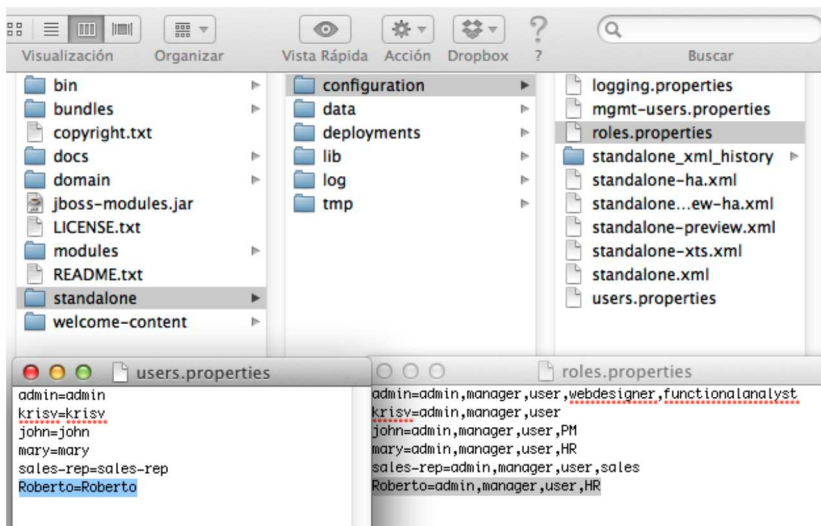
Esta aplicación genérica se suele llamar la consola.

Como tenemos arrancada la aplicación, sólo tenemos que ir a la siguiente URL para acceder a la aplicación. Esta consola yo diría que es casi un prototipo para que veas el código y tengáis ejemplos para hacer vuestras propias integraciones en vuestras aplicaciones, que es el segundo modo de hacer el trabajo.

Vamos a esta URL: <http://localhost:8080/jbpm-console/app.html>

Usamos el usuario y contraseña **krisv**

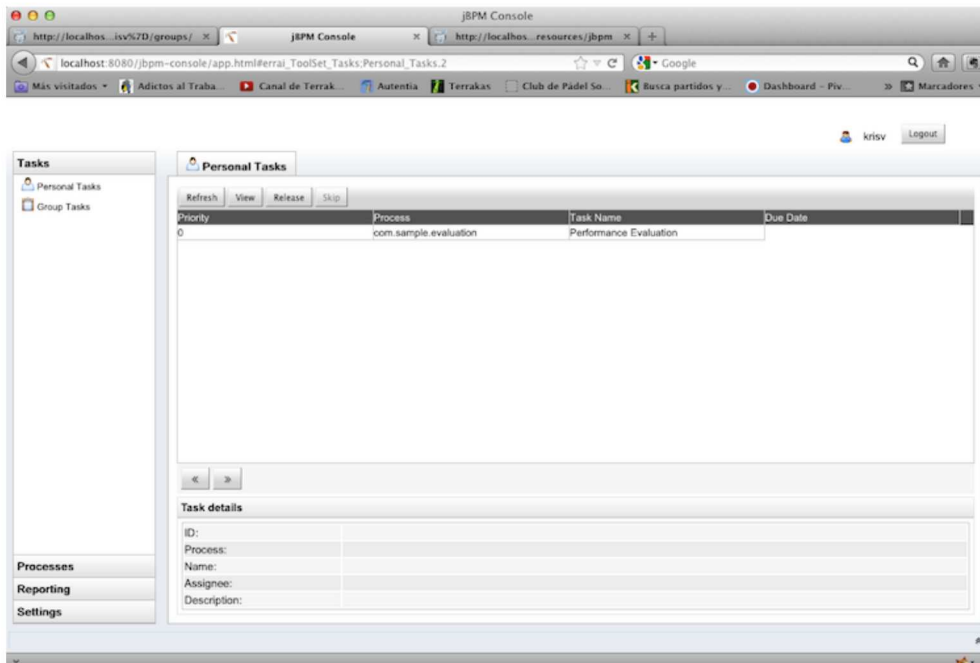
Si tenéis curiosidad de saber donde se registran estos usuarios, debéis saber que inicialmente el ejemplo de jBPM viene configurado con autenticación del servidor Jboss en el que corre. Si queréis crear un nuevo usuario, tendréis que ir a los ficheros: user y roles en `/Users/tuUsuario/tuDirectorio/jboss-as-7.0.2.Final/standalone/configuration`



Igual que en cualquier otra aplicación Web donde el servidor de aplicaciones se encarga de la identificación, se pueden usar otras opciones como LDAP, bases de datos, etc.

Ya en la consola, podemos ver los menús a la izquierda: Task, process, reports, setting.

Lo primero que hacemos es ver las tareas que tiene nuestro usuario. Como hemos lanzado nuestro programa java, **krisv** tendrá una tarea asignada.



Si pulsamos **view**, ejecutaremos la tarea. Si pulsamos complete, ya la tenemos terminada y la recibirán las bandejas de los siguientes usuarios en el flujo.

Task Form: Performance Evaluation

Employee evaluation

Please perform a self-evaluation.

Reason:MarshaledContentWrapper{content=[B@231f9881, marshaller=org.drools.marshalling.impl.SerializablePlaceholderResolverStrategy, type=class java.lang.String}

Please fill in the following evaluation form:

Rate the overall performance: Outstanding

Check any that apply:

☐ Displaying initiative
 ☐ Thriving on change
 ☐ Good communication skills

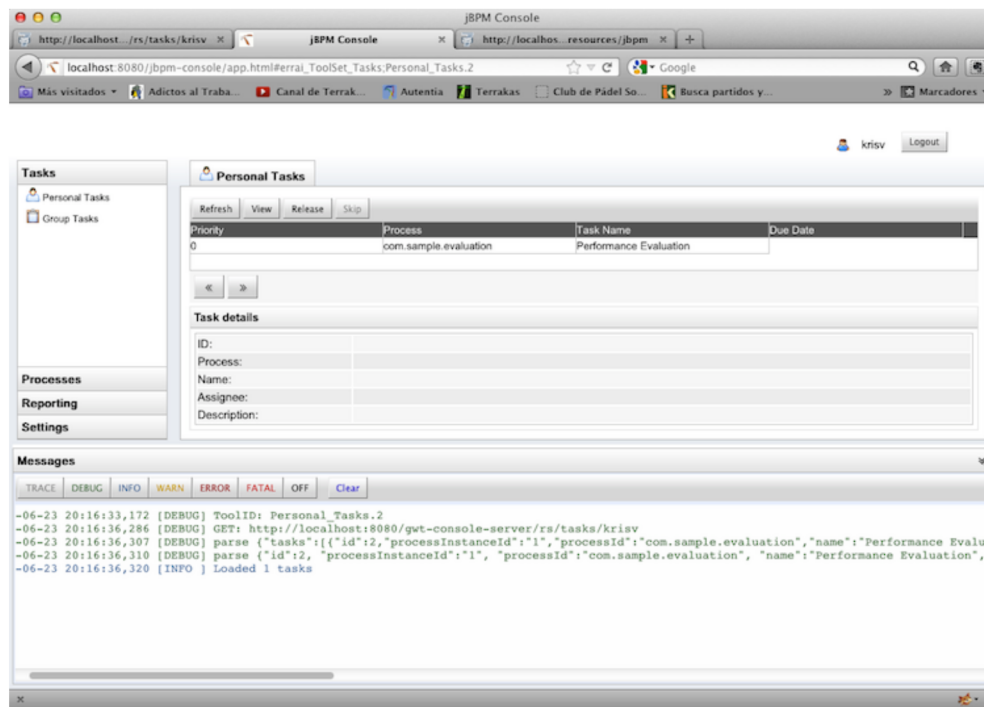
Complete

Si nos fijamos, en la parte inferior derecha hay unas flechas que despliegan una ventana con las trazas del sistema.

Lo realmente interesante es ver, en las trazas de consola jBPM, los comandos REST que está invocando y cómo se está parseando la respuesta.

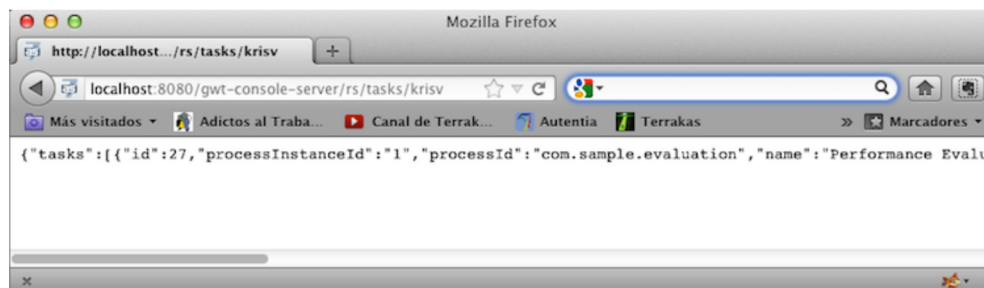
La consola puede ser válida para una pequeña empresa que se tenga que apañar tal cual pero es bastante probable que sea insuficiente para una gran organización que tenga muchas más aplicaciones. O bien la tendrá que adaptar y bien empotrar en su portal o integrar por medios externos. Esta integración se hará o bien a través de las APIs Java o mediante los servicios REST.

Además, es posible que otras aplicaciones tengan que mandar notificaciones a nuestros flujos, aunque lo lógico sería proporcionar unos servicios Web para ello.



Si queréis ver la lista de todos los comandos REST de jBPM podéis acceder a este enlace. <http://localhost:8080/gwt-console-server/rs/server/resources/jbpm>. Los comandos GET los podéis invocar desde un navegador. Para los POST se puede hacer a mano con `HttpURLConnection` o cualquier librería java-REST.

Veamos un ejemplo de invocación manual.



Igual que podemos ver la lista de tareas, podemos arrancar un proceso cualquiera de todos los disponibles ...

Pinchamos sobre **Process Overview** y podemos ver que existe el nuestro de Evaluation.

The screenshot shows the jBPM Console web application. The browser address bar indicates the URL is `localhost:8080/jbpm-console/app.html#errai_ToolSet_Pr`. The user is logged in as 'john'. The main content area is titled 'Process Overview' and contains a table of process instances. The table has columns for 'Process', 'v.', 'Instance', 'State', and 'Start Date'. The 'Process' column shows 'Evaluation' with a value of '0'. The 'Instance' column shows two instances: '3' and '4', both in a 'RUNNING' state. The 'Start Date' column shows '2012-06-21 19:30:18' for instance 3 and '2012-06-21 19:31:01' for instance 4. Below the table, there is an 'Execution details' panel for instance 4, showing its ID, state (RUNNING), start date, and activity (Self Evaluation). The left sidebar contains links for 'Tasks', 'Processes', 'Reporting', and 'Settings'.

Process	v.	Instance	State	Start Date
Evaluation	0	3	RUNNING	2012-06-21 19:30:18
		4	RUNNING	2012-06-21 19:31:01

Execution details for instance 4:

- ID: 4
- State: RUNNING
- Start Date: 2012-06-21 19:31:01
- Activity: Self Evaluation

Pulsamos start y arrancamos uno nuevo.

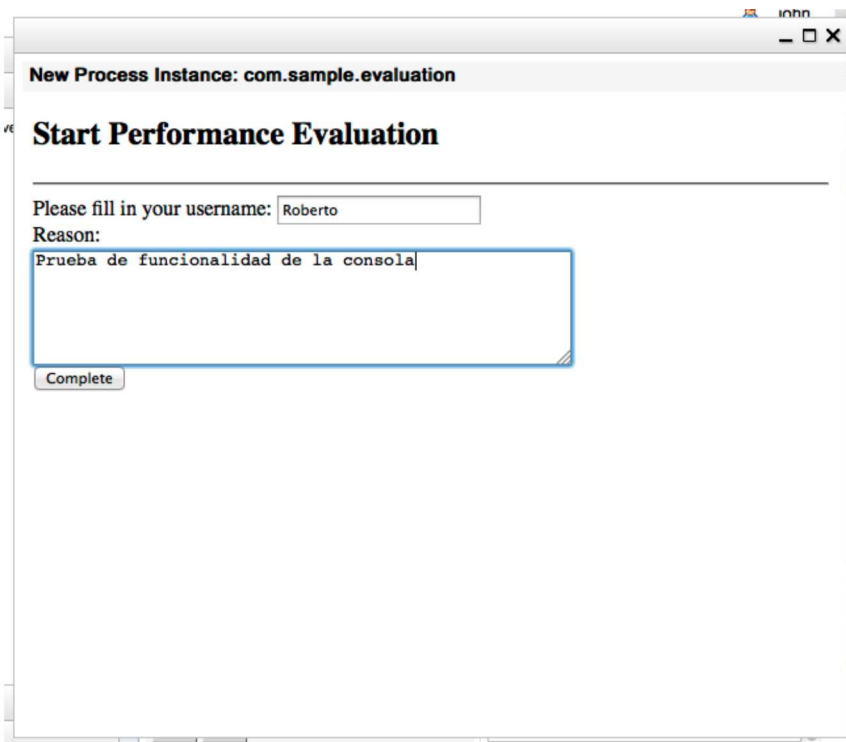
The screenshot shows the same jBPM Console web application, but with a 'Start new execution' dialog box open. The dialog box has a title bar with a close button (X) and a question mark icon. The text inside the dialog box asks: 'Do you want to start a new execution of this process?'. There are 'Cancel' and 'OK' buttons at the bottom of the dialog box. The background shows the 'Process Overview' page with the same table of process instances as in the previous screenshot.

Start new execution

Do you want to start a new execution of this process?

Cancel OK

Nos aparece una plantilla de formulario asociada al proceso, para lanzarlo.

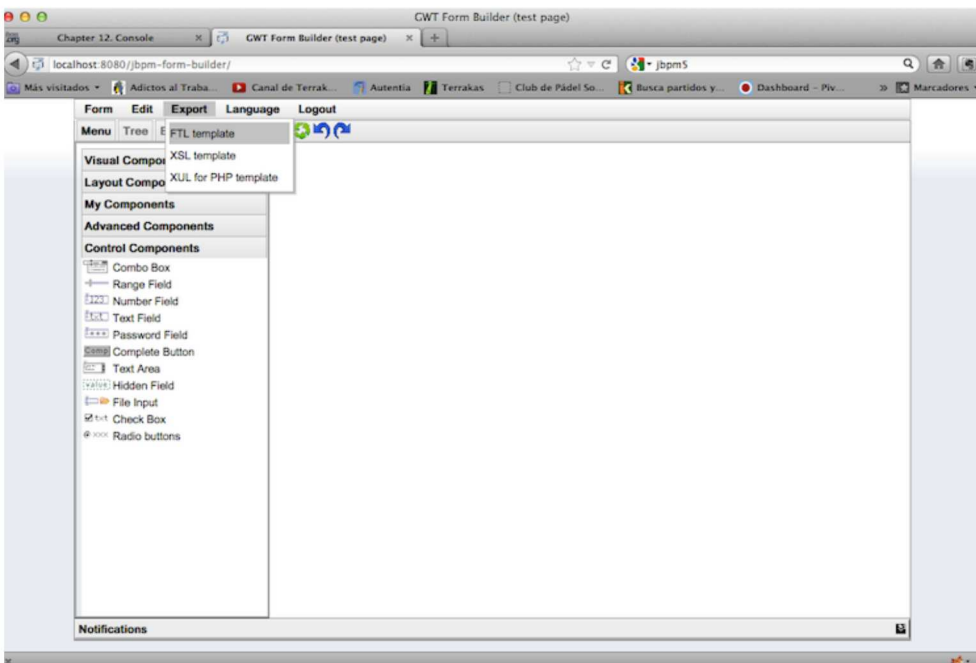


Si no queréis volveros locos para explicaros por qué la pantalla tiene ese aspecto, os recomiendo que busquéis los ficheros ftl, que se encuentran en el siguiente directorio:

```
/Users/rcanalesmora/jbpmadictos/jboss-as-7.0.2.Final/standalone/deployments/jbpm-gwt-console-server.jar >> en >> /WEB-INF  
/lib/jbpm-gwt-form-5.3.0.Final/com.sample.evaluation.ftl
```

Podréis ver cómo lo hace internamente la propia consola e invocar directamente a los formularios desde cualquier otra aplicación incluso no Web. (<https://github.com/droolsjbpm/jbpm/tree/master/jbpm-gwt/jbpm-gwt-form/src/main/java/org/jbpm/integration/console/forms>)

De echo, sería una buena idea echar un vistazo a la aplicación que también se despliega en nuestra instalación para editar los ficheros FTL: <http://localhost:8080/jbpm-form-builder/>



La consola nos proporciona mucha más información. Abajo a la derecha, podemos ver, de una copia de proceso, el diagrama indicando en que punto se encuentra. Como le hemos dicho que es para Roberto, se encontrará en la primera tarea asignado al usuario Roberto.

The screenshot shows the jBPM Console web application. The top navigation bar includes a user profile for 'john' and a 'Logout' button. The main interface is divided into several sections:

- Tasks:** A sidebar on the left with 'Processes' and 'Process Overview' options.
- Process Overview:** A table showing process instances. The 'Evaluation' process has 5 instances.
- Process Instance Activity:** A diagram for instance 5 showing a flow from 'Self Evaluation' to a parallel split, then to 'HR Evaluation' and 'PM Evaluation', followed by a parallel join, and finally to an end state.
- Execution details:** A panel at the bottom right showing details for instance 5: ID: 5, State: RUNNING, Start Date: 2012-06-21 19:36:54, Activity: Self Evaluation.

Process	V.	Instance	State	Start Date
Evaluation	0	3	RUNNING	2012-06-21 19:30:18
		4	RUNNING	2012-06-21 19:31:01
		5	RUNNING	2012-06-21 19:36:54

Process Instance Activity for Instance: 5

```
graph LR; Start(( )) --> SelfEval[Self Evaluation]; SelfEval --> Split{+}; Split --> HREval[HR Evaluation]; Split --> PMEval[PM Evaluation]; HREval --> Join{+}; PMEval --> Join; Join --> End((( )));
```

Execution details for Instance: 5

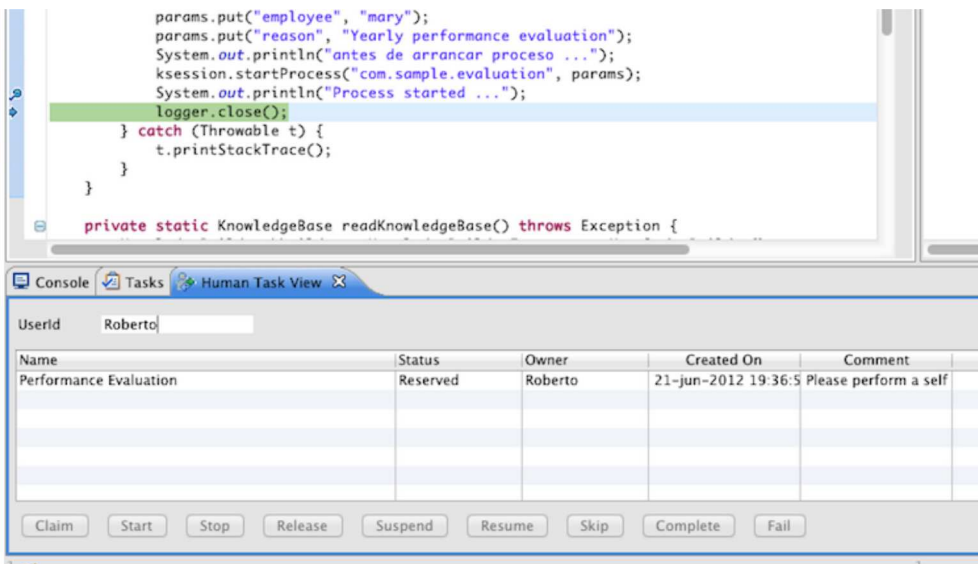
ID:	5
State:	RUNNING
Start Date:	2012-06-21 19:36:54
Activity:	Self Evaluation

Podemos ver también los datos de la instancia.

Process Instance Data: 5

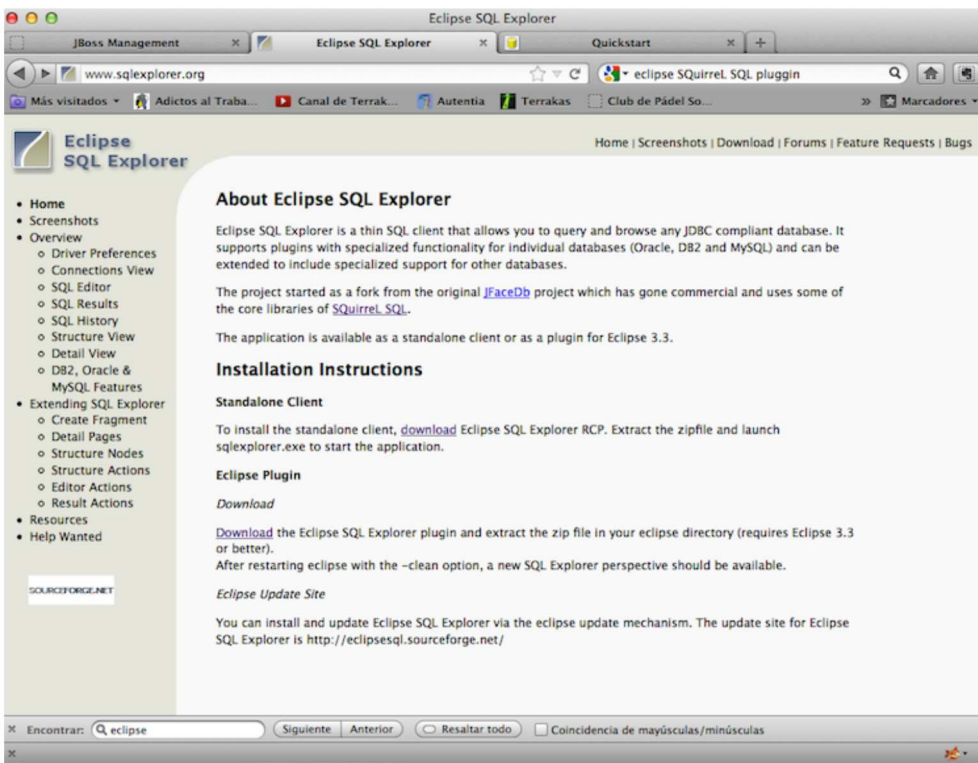
Key	XSD Type	Java Type	Value
content	hashMap	java.util.HashMap	n/a
reason	xs:string	java.lang.String	Prueba de funcionalidad de la c
employee	xs:string	java.lang.String	Roberto

Como inicialmente no hay un usuario con el que me pueda logar llamado Roberto, para ver que es verdad, me voy a Eclipse, donde todavía tengo el proceso en depuración y por lo tanto con sesión válida, y desde el visor de tareas humanas compruebo la tarea asignada a Roberto.



Para entender que está pasando dentro del sistema, lo mejor es conectarse a la base de datos e interrogar el modelo.

La demo viene configurada por defecto con una base de datos **h2**. Podríamos utilizar cualquier editor **jdbc**. Para recorrerla sin salir de eclipse me voy a descargar un plugin llamado Eclipse SQL Explorer.



Nos bajamos el fichero de sourceforge.org (SQL Explorer Plugin) y lo expandimos en la carpeta deseada.

Podemos arrancarlo como aplicación independiente ejecutando sqlexplorer.app o bien copiar los ficheros de las carpetas **features** y **plugin** a esas carpetas de nuestra instalación de eclipse, donde nos aparecerá una nueva perspectiva al rearrancarlo.

Nombre	Fecha de modificación
installgoogledrive.dmg	18/06/2012 13:25
jbpm-5.3.0.Final-installer-full.zip	ayer 16:32
jbpm-installer	ayer 16:52
SQLExplorer	hoy 19:07
artifacts.xml	17/11/2010 16:19
change_log.txt	17/11/2010 16:57
configuration	17/11/2010 16:57
features	17/11/2010 16:19
license.txt	17/11/2010 16:57
p2	17/11/2010 16:19
plugins	17/11/2010 16:19
README.txt	17/11/2010 16:57
sqlexplorer.app	17/11/2010 16:19
sqlexplorer_rcp-3.6.1.macosx.cocoa.x86.tgz	hoy 11:45

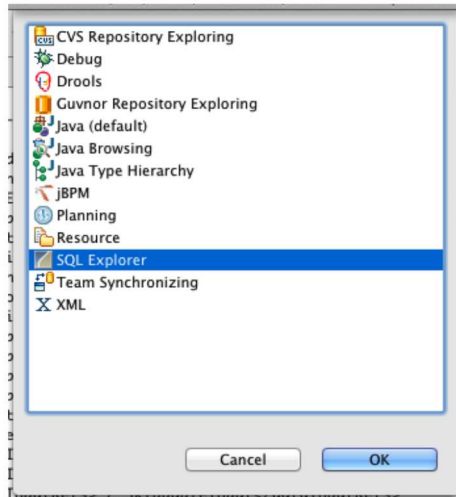
Looking for the latest version? [Download sqlexplorer_rcp-3.6.1.macosx.cocoa.x86.tgz \(44.1 MB\)](#)

Home

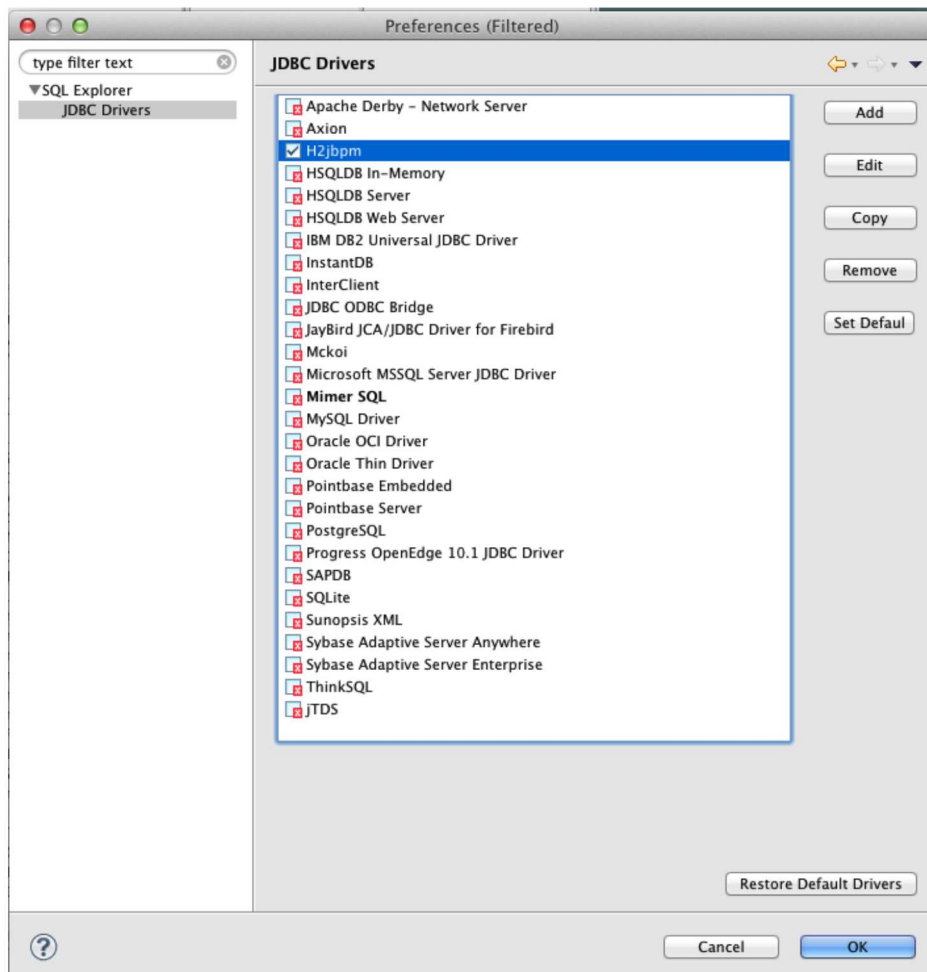
Name ▾	Modified ▾	Size ▾
SQL Explorer RCP (inc JRE)	2010-11-17	
SQL Explorer RCP (exc JRE)	2010-11-17	
SQL Explorer Plugin	2010-11-17	

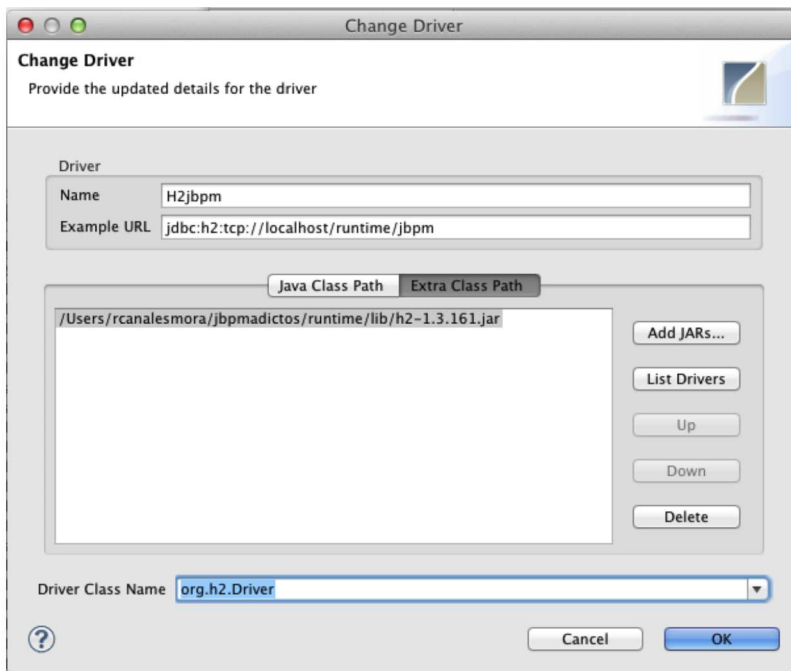
Totals: 3 Items

Cerramos Eclipse, lo volvemos a abrir y cambiamos a la perspectiva SQL Explorer.

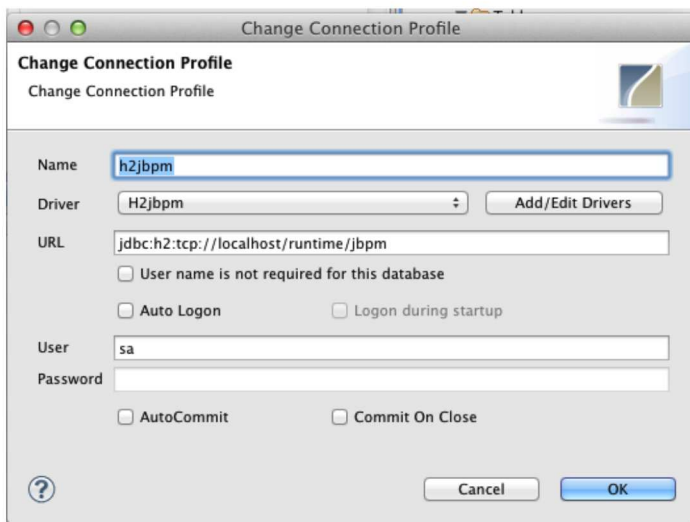


En la pantalla principal tenemos que definir la conexión que queremos utilizar. Vienen muchas de ejemplo: simplemente reciclamos una.





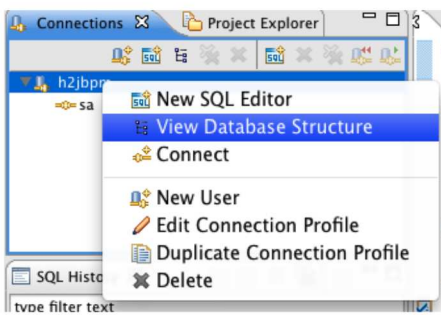
Quedará tal como esto:



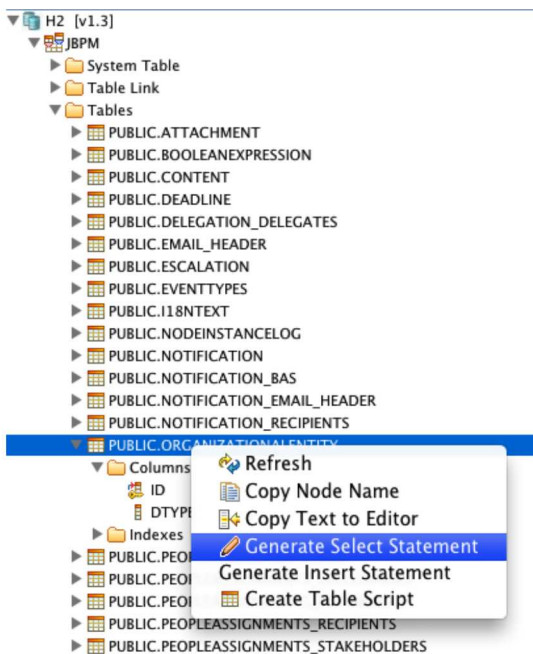
Para saber los parámetros que tenemos que poner lo mejor es espiar la definición del pool de conexiones del servidor de aplicaciones ya que la consola de jBPM se conecta a la misma base de datos. En mi instalación es /Users/rcanalesmora/jbpmadictos/jboss-as-7.0.2.Final/standalone/configuration/standalone.xml

```
standalone.xml
<datasource jndi-name="java:jboss/datasources/jbpmDS" pool-name="H2DS"
enabled="true" jta="true" use-java-context="true" use-ccm="true">
  <connection-url>
    jdbc:h2:tcp://localhost/runtime/jbpm
  </connection-url>
  <driver>
    h2
  </driver>
  <pool>
    <min-pool-size>
      1
    </min-pool-size>
    <max-pool-size>
      4
    </max-pool-size>
    <prefill>
      false
    </prefill>
    <use-strict-min>
      false
    </use-strict-min>
    <flush-strategy>
      FailingConnectionOnly
    </flush-strategy>
  </pool>
  <security>
    <user-name>
      sa
    </user-name>
    <password>
    </password>
  </security>
  <validation>
    <check-valid-connection-sql>
      SELECT 1
    </check-valid-connection-sql>
    <validate-on-match>
      false
    </validate-on-match>
    <background-validation>
      false
    </background-validation>
    <use-fast-fail>
      false
    </use-fast-fail>
  </validation>
</datasource>
<drivers>
  <driver name="h2" module="com.h2database.h2">
    <xa-datasource-class>
      org.h2.jdbcx.JdbcDataSource
    </xa-datasource-class>
  </driver>
</drivers>
</datasources>
</subsystem>
```

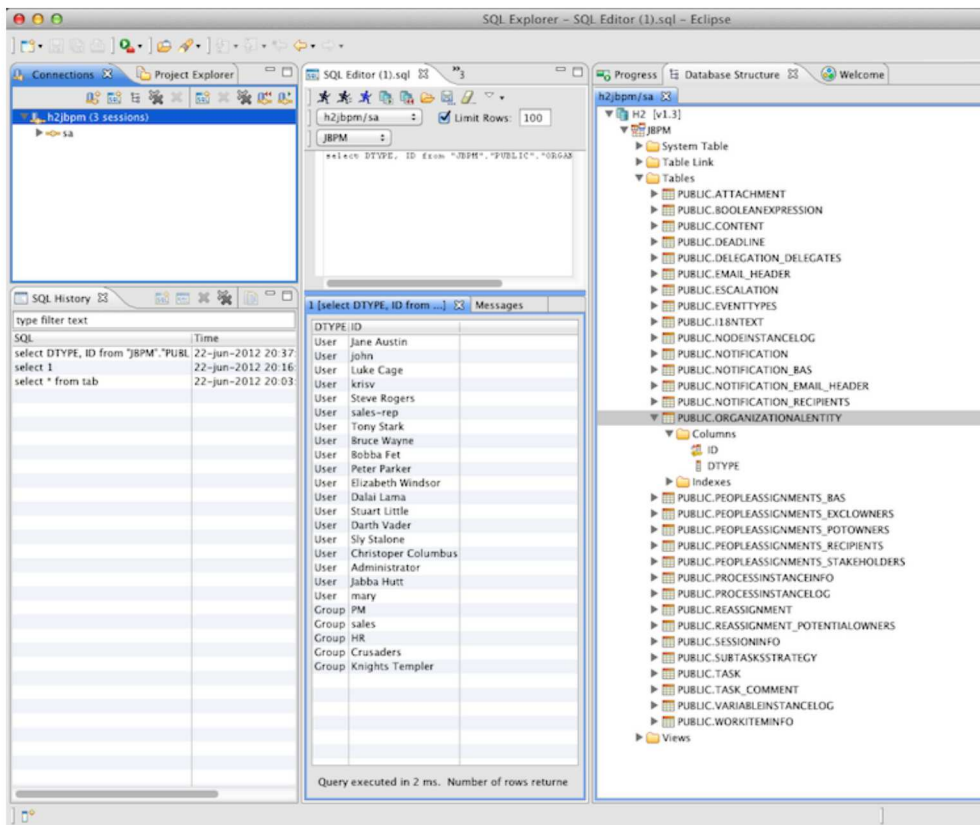
Ahora, sólo tenemos que pulsar conectar y View Database Structure



Podemos generar automáticamente la consulta a la tabla de jBPM deseada.



Y ya tenemos las tablas internas accesibles. Podemos ver donde se encuentran registrados los usuarios y los grupos.

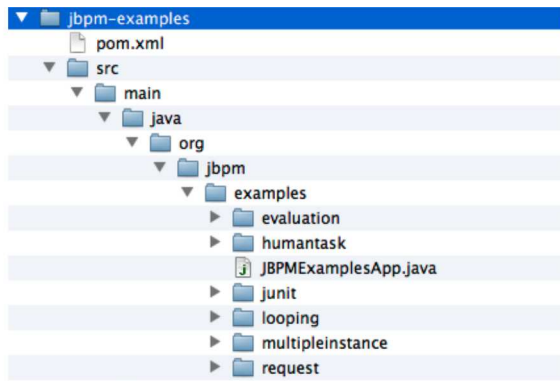


Obviamente no es conveniente acceder al modelo de datos directamente porque cualquier cambio de versión del producto o de su modelo de datos tirará por tierra nuestros desarrollos, por lo que es mejor utilizar el API adecuado. De todos modos, no está de más saber siempre que está pasando, sobre todo para tuning del sistema.

Bueno, hasta aquí ya hemos visto un montón de cosas para un simple artículo. Nos quedaría revisar el resto de los ejemplos para entrar en más detalle con las distintas posibilidades de BPM:

- Temporización de tareas.
- Invocación del motor de reglas.
- Envío y recepción de eventos.
- Creación de sub-procesos.

Vamos a [descargarlos](#).



Para instalarlos podemos importarlos en el workspace o copiarlos directamente en el proyecto activo.

También hace falta revisar muchas otras cosas como el versionado de los procesos, la persistencia de los datos, creación de tareas propias, etc. Vamos, que tenemos para semanas entretenidas.

A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)

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

[Share](#) |

0

Animáte 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](#)

Copyright 2003-2012 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

