Kie Integration scenariis

David THIBAU – 2019

david.thibau@gmail.com

Agenda

Considerations
Architecture
Maven knowledge repositories
Externalizing knowledge services

Integration Kie with applications

Considerations
Architectures
Maven repositories
Kie Server and Kie Workbench

Engine/application communiation

Some mechanisms to communicate between Drools and the rest of the application :

- Global variables may contain domain objects
- Entry points : Stream of facts and event
- Channels to send information to the outside world
- Listeners or marshallers

Synchronous

- If we need to execute our business rules in a synchronous manner
  - Using stateless Kie Sessions and create as many sessions as requests
  - Use global variables to store specific rule execution information

Asynchronous

- Kie Sessions may be shared between different threads, because some of them may insert new information and others might take care of firing rules
- Register special listeners that take care of notifying other components about special situations detected by the rules
- Entry points become a very useful component when multiple sources will be inserting information into a single Kie Session

Patterns of integration

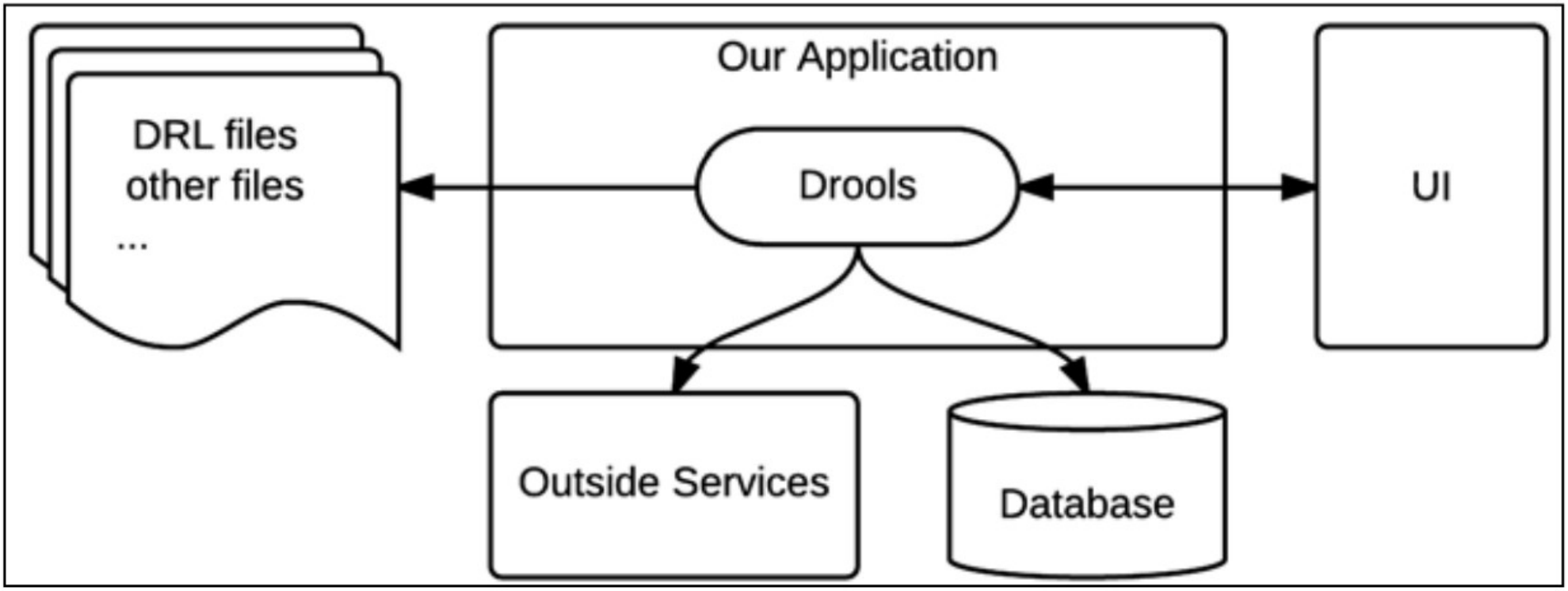
Scenarios of integration

Different ways of integration can be considered :

- Embed Drools runtime in your application
- Externalize rules in a separate repository
- Knowledge as a service

For these scenariis, Drools offers supports for CDI, Spring and Camel

Architecture



Drools Usage

Drools an interact with any and all layers of our application, depending on what we expect to accomplish with it

- It could interact with the UI to provide complex form validations
- It could interact with data sources to load persisted data when a rule evaluation determines it is needed
- It could interact with outside services, to either load complex information into the rules or send messages to outside services about the outcome of the rules execution

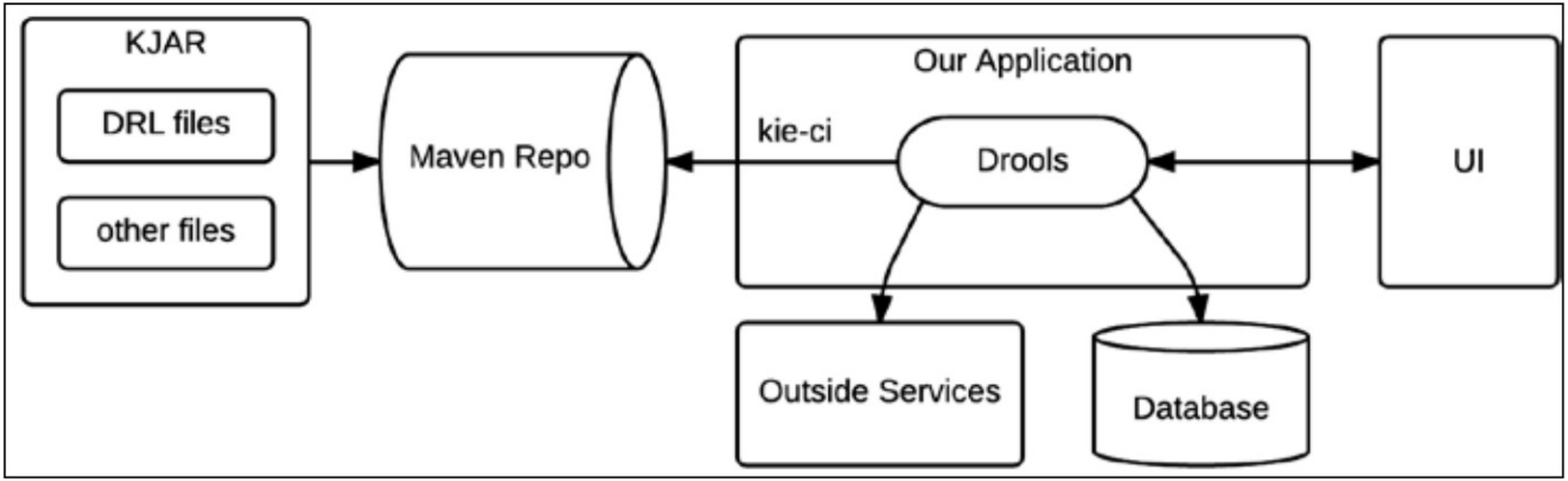
Maven Repo

Another approach is to separate the rules from the rest of the application by using a Maven repository :

- business rules are deployed in the Maven repository as a KJAR
- Components Drools runtime can dynamically load the rules

=> The rules can be developed, deployed, and managed as many times as needed without having to redeploy the applications.

Architecture



Loading the rules

With CDI

```
@Inject @KSession
@KReleaseId(groupId = "org.drools.devguide",artifactId =
  "chapter-11-kjar",
version = "0.1-SNAPSHOT")
KieSession kSession;
```

With a scanner

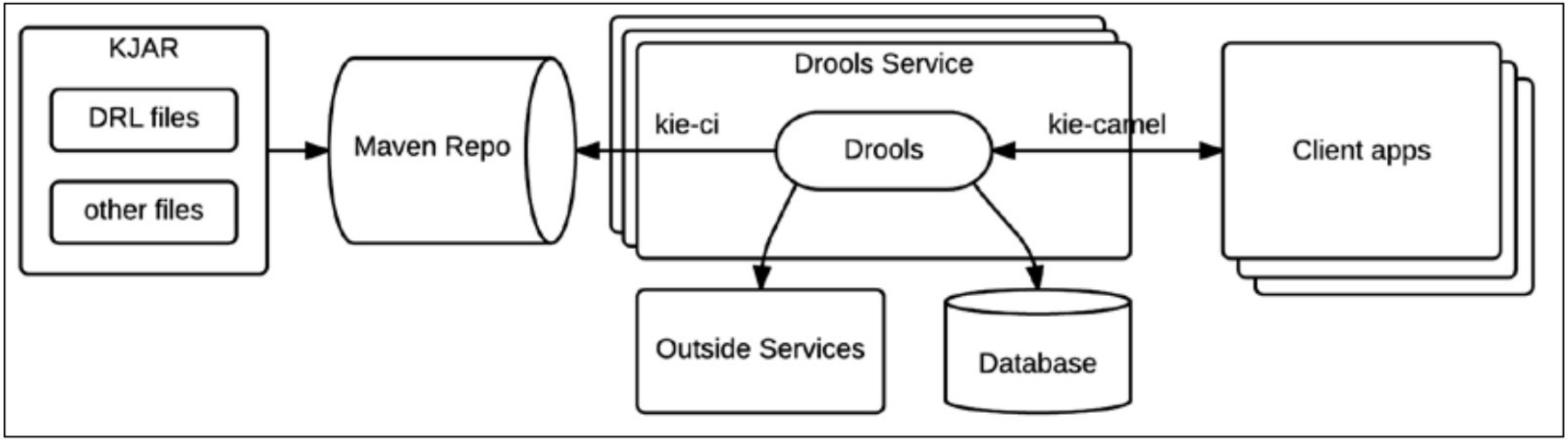
```
KieServices ks = KieServices.Factory.get();
KieContainer kContainer = ks.newKieContainer(
ks.newReleaseId("org.drools.devguide",
"chapter-11-kjar", "0.1-SNAPSHOT"));
KieScanner kScanner = ks.newKieScanner(kContainer);
kScanner.start(10_000);
KieSession kSession = kContainer.newKieSession();
```

Knowledge as a service

The last approach is to externalize the Drools runtime in an external services.

- This service expose une API (REST or other)
- Multiple-applications can access the service
- The service may be replicated if needed
- Life cycle of rules are completely independent

Architecture



Maven knowledge repositories

Building a kjar
Business resources update

Introduction

One of the primary goals when using a rule/process engine is to decorrelate the business rules/processes from the application that uses it.

Kie projects relies on Maven to deploy business rules/processes in a repository.

Applications can then:

- Either declare a Maven dependency to the artifact of containing the business resources
- The Maven repository can be local or remote

Building *kjar* with Maven

- A "Kie resources" project is typically a Maven project plus a *kmodule.xml* file configuring the various knowledge bases
- The build cycle is enriched with the Kie plugin that precompiles the rules and ensures their validity

```
<packaging>kjar</packaging>
...
<build>
  <plugins>
    <plugin>
      <groupId>org.kie</groupId>
      <artifactId>kie-maven-plugin</artifactId>
      <version>${project.version}</version>
      <extensions>true</extensions>
    </plugin>
  </plugins>
</build>
```

Default configuration

kmodule.xml defines knowledge bases and sessions that can be used in the project

An empty file gives a default configuration:

- A single knowledge base including all rules / processes / etc. found in the *resources* directory of jar
- A stateful session and a stateless session

Another *kmodule.xml*

```
<kmodule xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance " xmlns="http://jboss.org/kie/6.0.0/kmodule">
  <kbase name="KBase1" default="true" eventProcessingMode="cloud" equalsBehavior="equality">
    <ksession name="KSession2_1" type="stateful" default="true/">
    <ksession name="KSession2_1" type="stateless" default="false" beliefSystem="jtms"/>
</kbase>
<kbase name=""KBase2" default="false" eventProcessingMode="stream" equalsBehavior="equality"
 declarativeAgenda="enabled"
packages="org.domain.pkg2,org.domain.pkg3" includes="KBase1">
<ksession name="KSession2_1" type="stateful" default="false" clockType="realtime">
  <fileLogger file="drools.log" threaded="true" interval="10"/>
  <workItemHandlers>
    <workItemHandler name="name" type="org.domain.WorkItemHandler"/>
  </workItemHandlers>
  <listeners>
    <ruleRuntimeEventListener type="org.domain.RuleRuntimeListener"/>
    <agendaEventListener type="org.domain.FirstAgendaListener"/>
    <agendaEventListener type="org.domain.SecondAgendaListener"/>
    <processEventListener type="org.domain.ProcessListener"/>
  </listeners>
</ksession>
</kbase>
</kmodule>
```

Attributes of *kbase*

nom	défaut	autorisé	description
name	none		Mandatory. Used to retreive the base from the container
includes	none	Liste d'item séparés par des virgules	Knowledge bases to include
packages	all	Liste d'item séparés par des virgules	List the resources to compile
default	false	true/false	You do not have to specify the name for the default knowledge base
equalsBehaviour	identity	identity/equality	Test of presence of a fact in the knowledge base
eventProcessingMode	cloud	cloud/stream	Stream for drools-fusion
declarativeAgenda	disabled	disabled/enabled	Declarative agenda activated or not. experimental, the rules can act directly on the activations present in the agenda

Attributes of *ksession*

nom	défaut	autorisé	description
name	none		Mandatory. Name used to retrieve a container session
type	stateful	stateless/ stateful	
default	false	true/false	It is not necessary to specify the name for the default session
clockType	realtime	realtime/ pseudo	Indicates whether the timestamp of events is provided by the system or by the application
beliefSystem	simple	simple/jtms/ defeasible	System of fact management

Elements of *ksession*

A <*ksession*> element can also define different sub-elements:

- A **logger** is used to define a trace file that will record all Drools events in a file
- **WorkItemHandlers** allows you to define task managers that are associated with specific Drools-flow nodes (ex: Human task)
- **Event listeners**: *ruleRuntimeEventListener*, *agendaEventListener*, or *processEventListener*

Classpath Container (jbpm6)

Once the application has access to *kmodule.xml*, it can retreive the knowledge bases and create sessions via their name.

```
KieServices kieServices = KieServices.Factory.get();
KieContainer kContainer = kieServices.getKieClasspathContainer();
KieBase kBase1 = kContainer.getKieBase("KBase1");
KieSession kieSession1 = kContainer.newKieSession("KSession2_1");
StatelessKieSession kieSession2 = kContainer.newStatelessKieSession("KSession2_2");
```

Maven and Release ID (jbpm6)

If the KIE project is a Maven project, it is possible to use the Maven coordinates to instantiate the container.

```
KieServices kieServices = KieServices.Factory.get();
ReleaseId releaseId = kieServices.newReleaseId( "org.acme", "project", "1.0" );
KieContainer kieContainer = kieServices.newKieContainer( releaseId ) ;

KieSession kSession = kContainer.newKieSession("ksession1");
```

Jbpm7 style

If we prefer to use *RuntimeManager*, the *RuntimeEnvironmentBuilderFactory* interface provides methods based on classpath or ReleaseId :

```
// Based on classpath
public RuntimeEnvironmentBuilder
  newClasspathKmoduleDefaultBuilder();
// Based on Maven repository
public RuntimeEnvironmentBuilder newDefaultBuilder(String
  groupId, String artifactId, String version);
public RuntimeEnvironmentBuilder
  newDefaultBuilder(ReleaseId releaseId);
```

Sample

```
ReleaseId releaseId = ks.newReleaseId("org.jbpm", "helloworld", "1.0");
builder = RuntimeEnvironmentBuilder.Factory.get()
          .newDefaultBuilder(releaseId)
            .entityManagerFactory(emf)
            .registerableItemsFactory(new DefaultRegisterableItemsFactory() {
                 public Map<String, WorkItemHandler> getWorkItemHandlers(RuntimeEngine runtime) {
                     return myhandlers;
                 }
              public List<ProcessEventListener> getProcessEventListeners(RuntimeEngine runtime) {
                     return myProcessListeners;
                 }
              public List<AgendaEventListener> getAgendaEventListeners( RuntimeEngine runtime) {
                     return myAgendaLlisteners;
                   }
              public List<TaskLifeCycleEventListener> getTaskListeners() {
                     return myTaskListener;
              }
            })
            .userGroupCallback(userGroupCallback)
```

Update of business resources

Deploymenent

Deployment consist of updating a Maven repository.

The client application can then :

- Be manually updated to use a new version of the *kjar*
- Be dependent of the latest version
- Be dependent of a family of version

KieScanner

KieScanner allows to constantly scan the Maven repository to check if a new release of the project is available

In this case, the new release is deployed in the *KieContainer* and the new rules are taken into account

The use of *KieScanner* requires the presence of ***kie-ci.jar*** in the classpath

Scanner registration

```
KieServices kieServices = KieServices.Factory.get();

ReleaseId releaseId = kieServices.newReleaseId( "org.acme",
   "myartifact", "1.0-SNAPSHOT" );

KieContainer kContainer =
   kieServices.newKieContainer( releaseId );

KieScanner kScanner =
   kieServices.newKieScanner( kContainer );

// Start KieScanner

// which scans the repository every 10 seconds

kScanner.start( 10000L );
```

Upgrade

Automatic update is effective if:

- the version of the artifact is suffixed with SNAPSHOT, LATEST RELEASE, or a version interval
- KieScanner finds an update in the Maven repository

The update consists of downloading the new version and performing an incremental build.

KieBases and *KieSessions* controlled by *KieContainer* are automatically updated and all new *KieBases* or *KieSessions* use the new version of the project.

Externalizing knowledge services

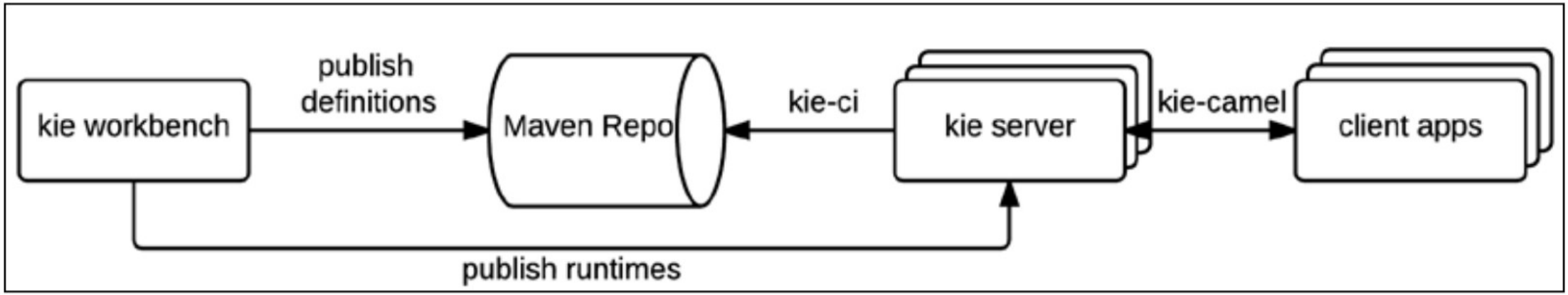
Kie Server
Kie Workbench

Introduction

KIE provides a module called ***kie-server*** , which can be used to configure similar environments that are only responsible for running Drools rules and processes, taking the kJAR from outside sources

It provides also ***kie-workbench*** which can manage rules and processes : edit, build deploy

Architecture



Kie Server

- It is a modular, standalone server component that can be used to execute rules and processes, configured as a WAR file
- It is available for web containers and JEE6 and JEE7 application containers
- It can be easily deployed in cloud environments
- Each instance of the Kie Server can manage many Kie Containers
- Its functionality can be extended through Kie Server Extensions

Configuration

The distributions of Kie Server : archive zip or Docker image use a jBoss wildfly server in the full configuration.

Once a user with the role *kie-server* configured

The REST API is available at :
http://localhost:8080/kie-server/services/rest/server/

A swagger documentation is avaiable

http://localhost:8080/kie-server/docs/

Sample Response


```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<response type="SUCCESS" msg="Kie Server info">
    <kie-server-info>
        <capabilities>KieServer</capabilities>
        <capabilities>BRM</capabilities>
        <capabilities>BPM</capabilities>
        <capabilities>CaseMgmt</capabilities>
        <capabilities>BPM-UI</capabilities>
        <capabilities>BRP</capabilities>
        <capabilities>DMN</capabilities>
        <capabilities>Swagger</capabilities>
        <location>http://localhost:8230/kie-server/services/rest/server</location>
        ...
    </kie-server-info>
</response>
```

Native REST client for Execution Server

- Interaction with the kie-server can be performed by any REST client (curl, postman, Spring restTemplate, ...)
- But KIE offers a native rest client suited for this kind of interaction. It is available with the org.kie:kie-server-client dependency

Client request configuration

Configuration of the client includes :

- Credentials of the kie-server user
- KIE Server location, such as http://localhost:8080/kie-server/services/rest/server
- Marshalling format for API requests and responses (JSON, JAXB, or XSTREAM)
- A KieServicesConfiguration object and a KieServicesClient object, which serve as the entry point for starting the server communication using the Java client API
- A KieServicesFactory object defining REST protocol and user access
- Any other client services used, such as RuleServicesClient, ProcessServicesClient, or QueryServicesClient

Types of client

There are different type of client each is specialized in an aspect of KIE :

- ***RuleServicesClient*** : Used to insert/rectract facts and fire rules
- ***ProcessServicesClient*** : Used to start, signal, and abort processes or work items
- ***QueryServicesClient***: Used to query processes, process nodes, and process variables
- ***UserTaskServicesClient*** : Used to perform all user-task operations, such as starting, claiming, or canceling a tas
- ...

Sample (1)

```
public static void main(String[] args) {
        initializeKieServerClient();
        initializeDroolsServiceClients();
        initializeJbpmServiceClients();
    }
```

Sample (2)

```
public static void initializeKieServerClient() {
        conf = KieServicesFactory.newRestConfiguration(URL, USER,
   PASSWORD);
        conf.setMarshallingFormat(FORMAT);
        kieServicesClient =
   KieServicesFactory.newKieServicesClient(conf);
    }


    public static void initializeDroolsServiceClients() {
        ruleClient =
   kieServicesClient.getServicesClient(RuleServicesClient.class);
        dmnClient =
   kieServicesClient.getServicesClient(DMNServicesClient.class);
    }
```

Sample (3)

```
public static void initializeJbpmServiceClients() {
        caseClient =
    kieServicesClient.getServicesClient(CaseServicesClient.class);
        documentClient =
    kieServicesClient.getServicesClient(DocumentServicesClient.class);
        jobClient =
    kieServicesClient.getServicesClient(JobServicesClient.class);
        processClient =
    kieServicesClient.getServicesClient(ProcessServicesClient.class);
        queryClient =
    kieServicesClient.getServicesClient(QueryServicesClient.class);
        uiClient = kieServicesClient.getServicesClient(UIServicesClient.class);
        userTaskClient =
    kieServicesClient.getServicesClient(UserTaskServicesClient.class);
     }
```

Sample Use of RuleService

```
public void executeCommands() {

  String containerId = "hello";
  System.out.println("== Sending commands to the server ==");
  RuleServicesClient rulesClient = kieServicesClient.getServicesClient(RuleServicesClient.class);
  KieCommands commandsFactory = KieServices.Factory.get().getCommands();

  Command<?> insert = commandsFactory.newInsert("Some String OBJ");
  Command<?> fireAllRules = commandsFactory.newFireAllRules();
  Command<?> batchCommand = commandsFactory.newBatchExecution(Arrays.asList(insert, fireAllRules));

  ServiceResponse<String> executeResponse = rulesClient.executeCommands(containerId, batchCommand);

  if(executeResponse.getType() == ResponseType.SUCCESS) {
    System.out.println("Commands executed with success! Response: ");
    System.out.println(executeResponse.getResult());
  } else {
    System.out.println("Error executing rules. Message: ");
    System.out.println(executeResponse.getMsg());
  }
}
```

JBPM Commands

StartProcessCommand

SignalEventCommand

CompleteWorkItemCommand

AbortWorkItemCommand

QueryCommand

Kie workbenches

Workbenches

Kie projects provide a set of usable workbench tools that allow to create, build, and deploy rule and process definitions in any Kie Server

These workbenches are build with the framework *UberFire* .

They are web application deployed in a application server (Jboss wildfly)

Installation

Download the distribution which is a war
kie-wb-*.war

Deploy it in the application server

- Drools 7 is only available for Wildfly 14

Workbench is available at :
http://<server>:<port>/

Roles and users

A WB use the following roles:

- ***admin*** : Administrator, manages users, repositories, ...
- ***developer*** : Manage assets (rules, models, processes, forms. Create build and deploy projects
- ***analyst*** : Idem developer with restricted right (no deployment for example)
- ***user*** : Participate in business processes and perform tasks
- ***manager*** : Access to reporting

Getting started

Typical actions to start:

1) Create a user with the admin rôle
2) Add a repository
3) Add a project
4) Provide the business model
5) Create the rules
6) Build and deploy

These actions can be done through the web interface, an online command tool (kie-config-cli.sh) or via a REST interface

Packages

- Package configuration is usually done only once by someone with expertise with rules and templates
- All assets belong to a single package that acts as a namespace
- A package is created by specifying its name

Formats of rules

The workbench supports multiple rule formats and editors:

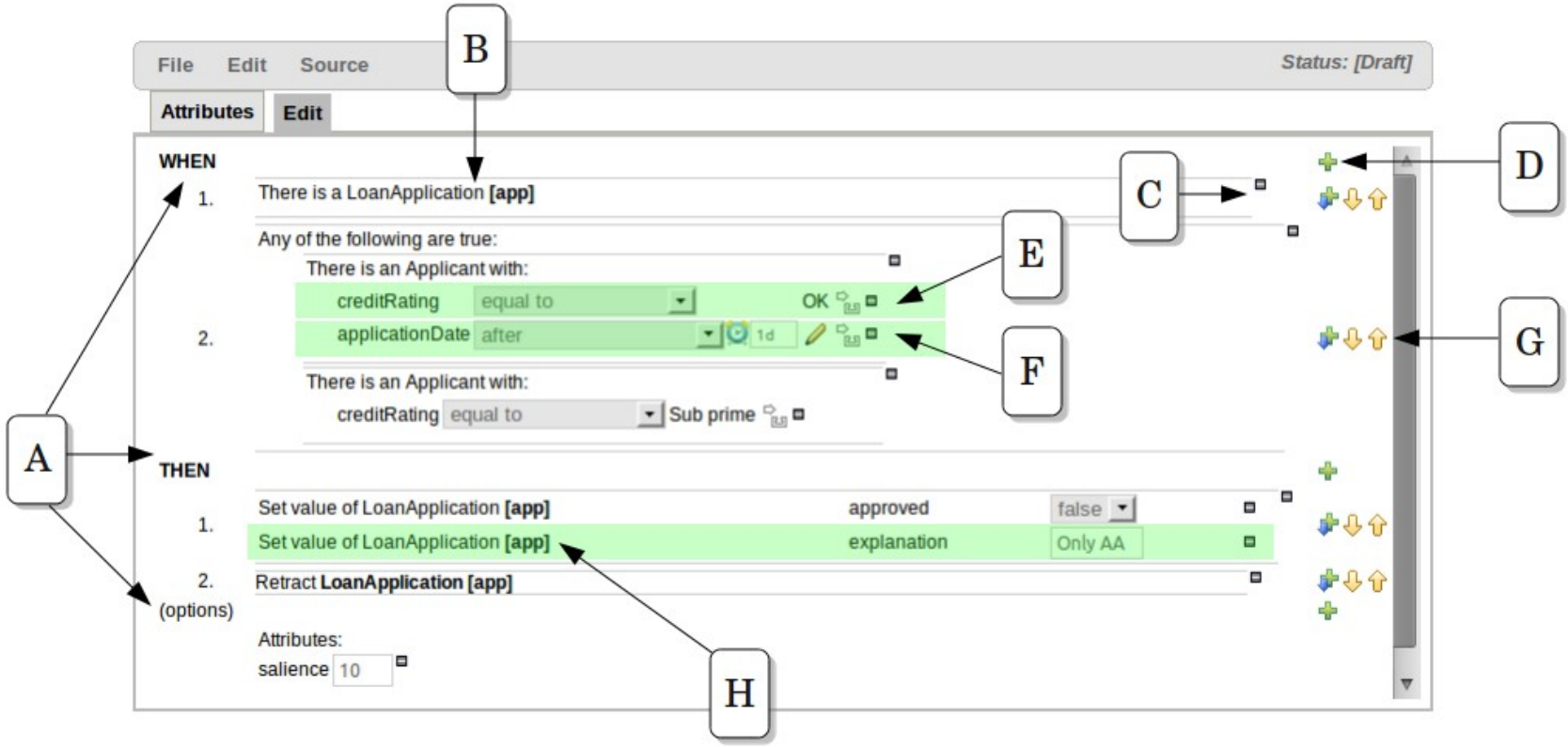
- BRL Format with BRL Editor (also present in Eclipse)
- DSL
- Decision tables in file to upload
- Decision tables edited directly in the web interface
- Rule Flow process to upload
- DRL
- functions
- Configuring lists of values (Enum)
- Rules template powered by data tables

New Rule → name, category and format

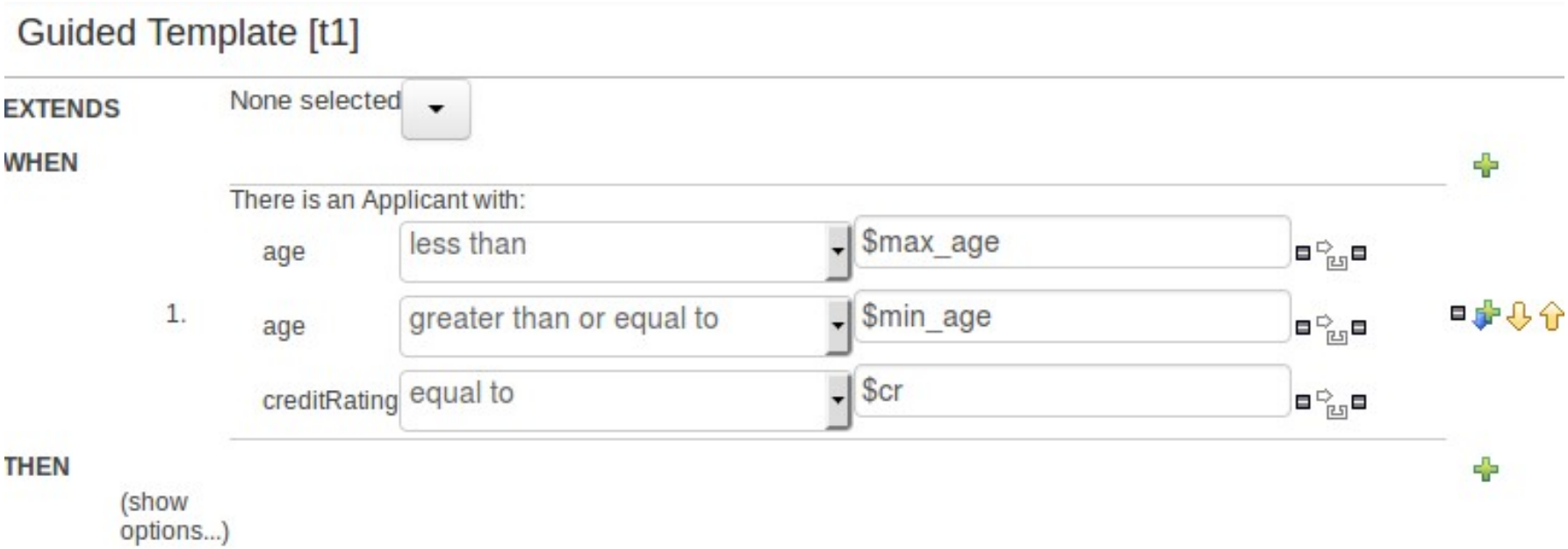
=> A wizard starts

It is always possible to consult the source in drl

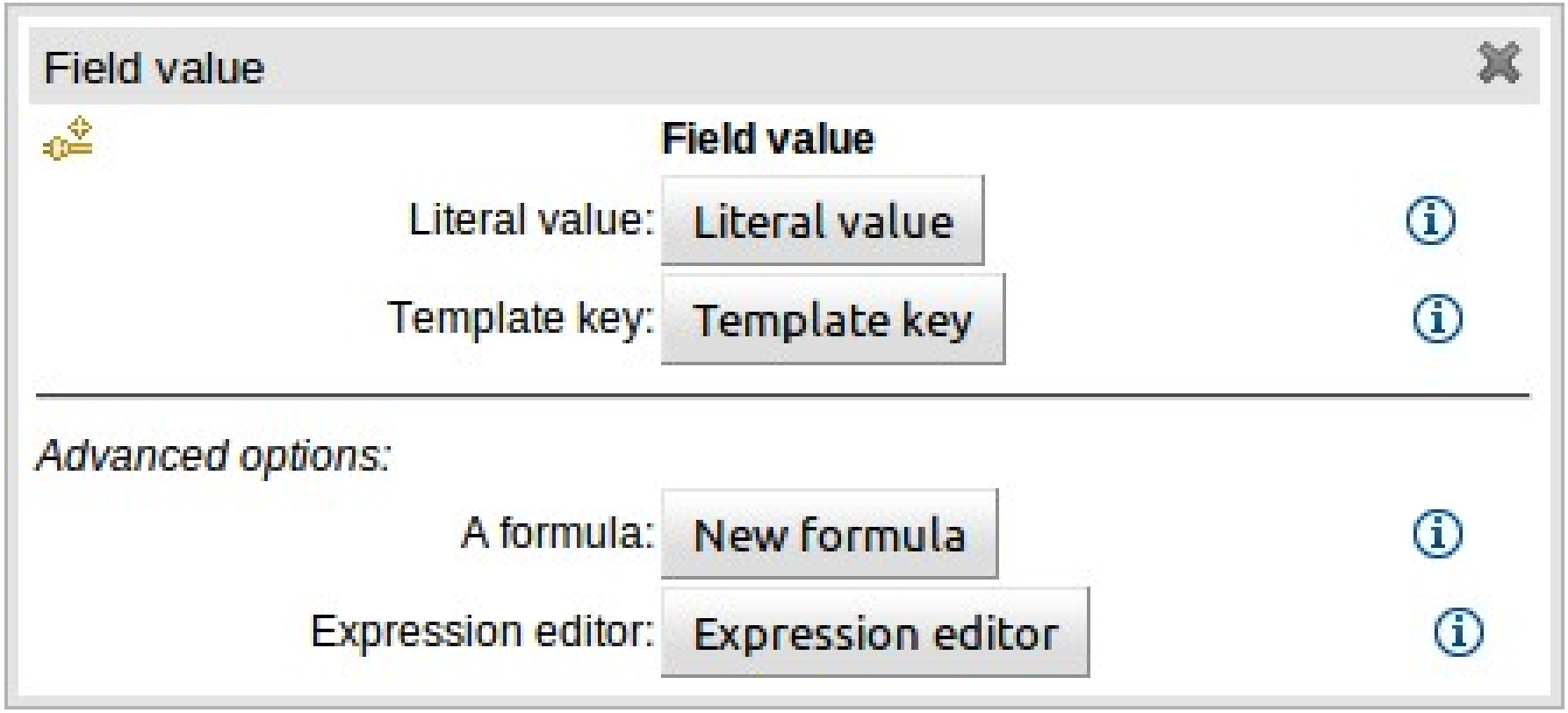
Guided Business Rules Editor



Guided Template Editor



Keys of templates



Template data

Guided Template [t1]

Add row...

$max_age	$min_age	$cr
25	20	AA
		OK
		Sub prime
35	25	AA
		OK
		Sub prime
45	35	AA
		OK
		Sub prime

Decision Table Editor

Drools-WB offers an editor for decision tables.

The editor proposes facts and fields available in the context of the project

2 types of tables can be created:

- **Extended Entries**: Column definitions do not specify a value. The values are then indicated in the body of the table. However, they can be restricted by an interval.
- **Limited Entries**: The column definitions specify a value. The body of the table contains checkboxes

Extended table

Decision table

#	Description	Age	Make	Premium
		Applicant [$a]	Vehicle [$v]	
		age [<]	make [==]	
1		35	BMW	1000
2		35	Audi	1000

Limited table

Decision table

#	Description	Age < 35	BMW	Audi	Premium 1000
		Applicant [$a]	Vehicle [$v]		
		age [<35]	make [==BMW]	make [==Audi]	
1		☑	☑	☑	☐
2		☐	☑	☑	☐
3		☑	☐	☑	☐
4		☐	☐	☑	☐
5		☑	☑	☐	☐
6		☐	☑	☐	☐
7		☑	☐	☐	☐
8		☐	☐	☐	☐

Test scenario

The test scenarios validate the operation of the rules and avoid regression bugs.

A test case defines several sections:

- **Given** list the test facts
- **Expected** lists the expected changes and actions

Test scenario

Test Scenario [Good credit history only]

Save Delete Rename Copy

Run scenario

GIVEN

Insert 'LoanApplication'[app]

approved: false

Insert 'IncomeSource'[incomeSource]

Add a field

Insert 'Applicant'[a]

creditRating: OK

CALL METHOD

Add input data and expectations here.

EXPECT

Use real date and time

LoanApplication 'app' has values:

approved: equals false

More...

(configuration) All rules may fire

(globals)

Test Scenario Config Metadata All Test Scenarios

Reporting

Success

Text

...

DSL editor

File Edit Source

Attributes Edit

```
[when]When the credit rating is {rating:ENUM:Applicant.creditRating} = applicant:Applicant(creditRating=="{rating}")

[when]When the applicant dates is after {dos:DATE:default} = applicant:Applicant(applicationDate>"{dos}")

[when]When the applicant approval is {bool:BOOLEAN:checked} = applicant:Applicant(approved=={bool})

[when]When the ages is less than {num:1?[0-9]?[0-9]} = applicant:Applicant(age<{num})

[then]Approve the loan = applicant.setApproved(true);

[then]Set applicant name to {name} = applicant.setName("{name}");
```

List of values

Enum Editor [credit ratings]

Save | Delete | Rename | Copy | Validate

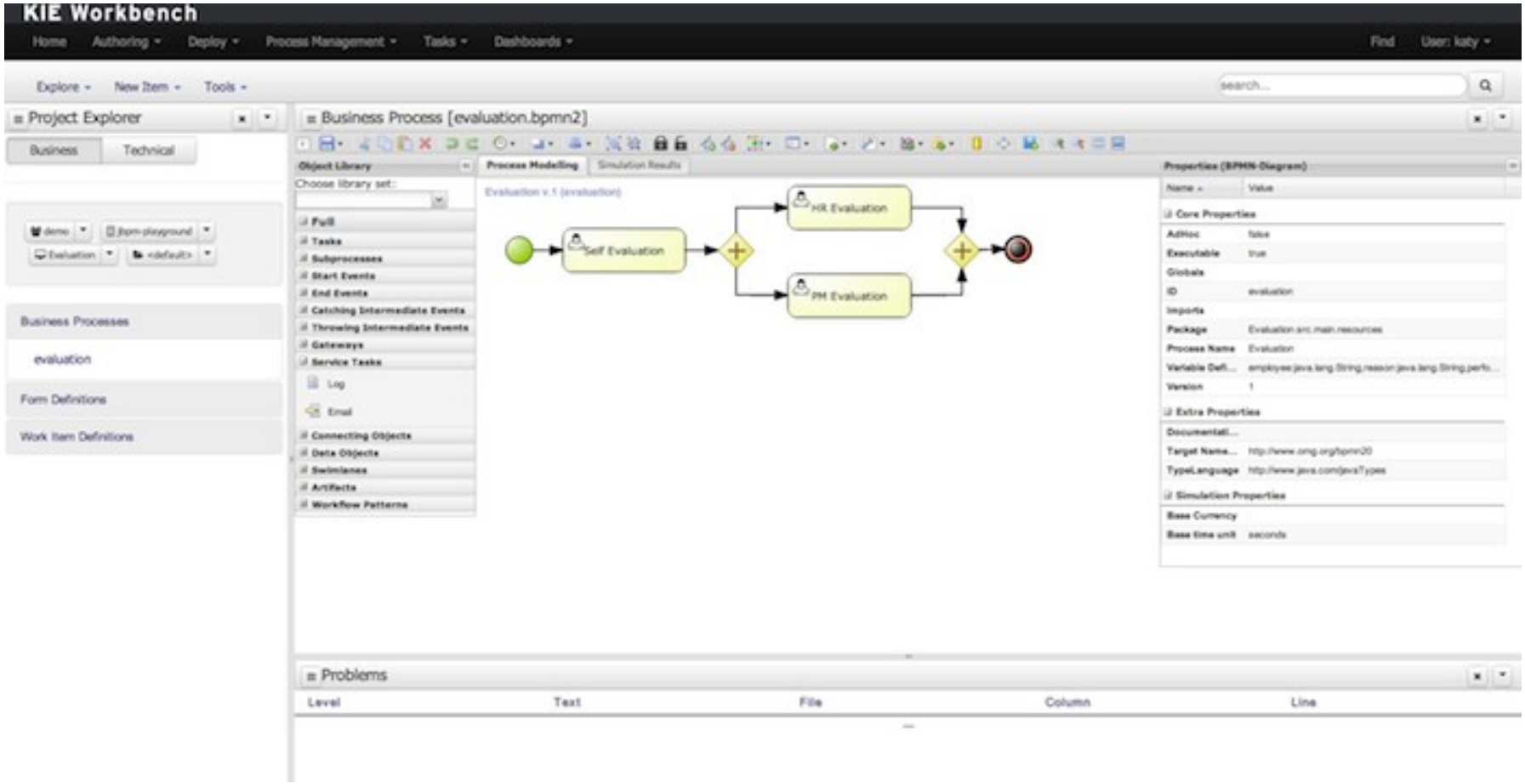
Add enum

	Fact	Field	Context
	Applicant	creditRating	['AA', 'OK', 'Sub prime']
	Person	age	['20','25','30','35']

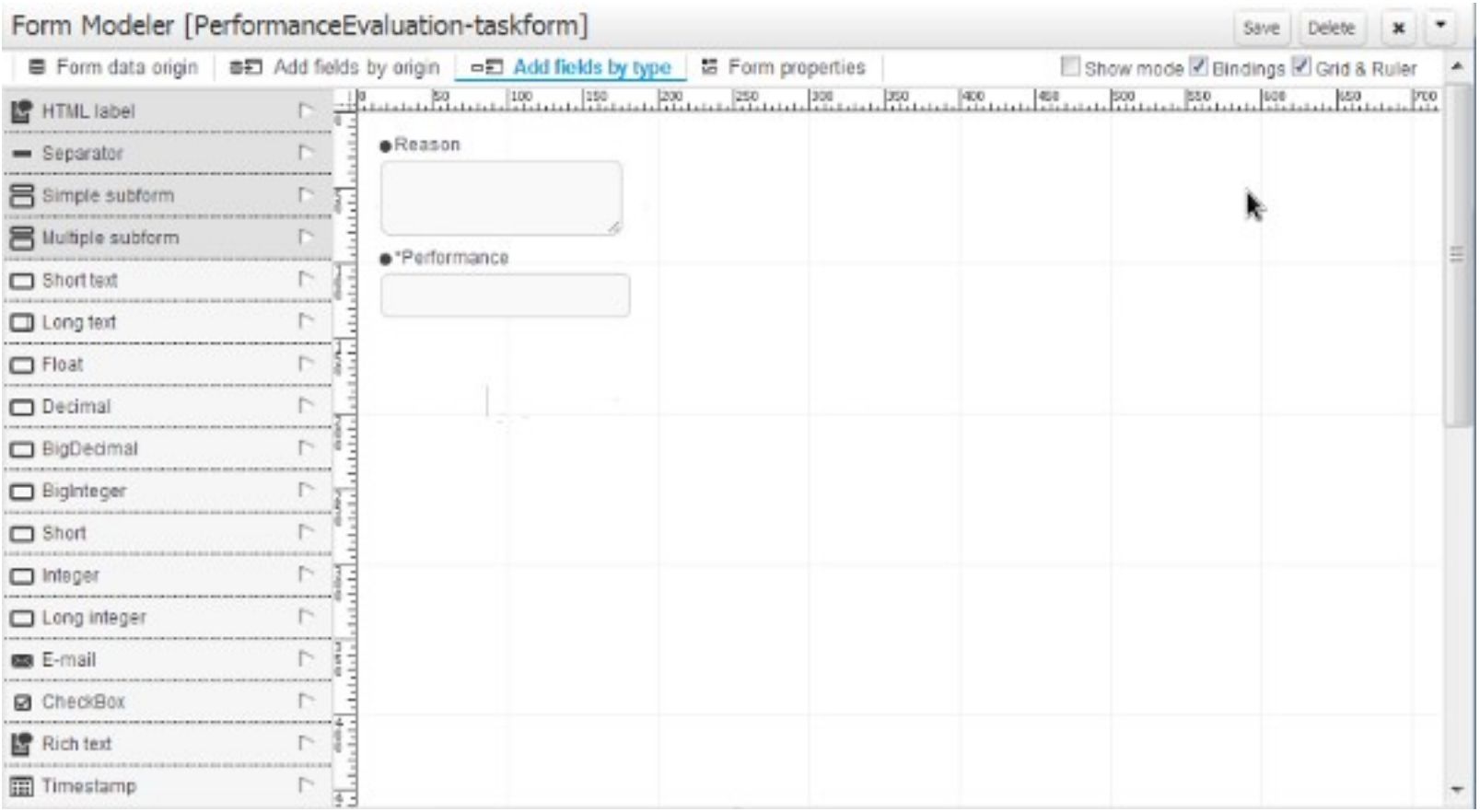
Access to the Workbench database

```
public class MainKieTest {

    public static void main(String[] args) {

        // works even without -SNAPSHOT versions
        String url = "http://localhost:8080/kie-drools/maven2/de/test/Test/1.2.3/Test-1.2.3.jar";

        // make sure you use "LATEST" here!
        ReleaseIdImpl releaseId = new ReleaseIdImpl("de.test", "Test", "LATEST");

        KieServices ks = KieServices.Factory.get();

        ks.getResources().newUrlResource(url);

        KieContainer kieContainer = ks.newKieContainer(releaseId);

        // check every 5 seconds if there is a new version at the URL
        KieScanner kieScanner = ks.newKieScanner(kieContainer);
        kieScanner.start(5000L);
        // alternatively:
        // kieScanner.scanNow();

        Scanner scanner = new Scanner(System.in);
        while (true) {
            runRule(kieContainer);
            System.out.println("Press enter in order to run the test again....");
            scanner.nextLine();
        }
    }

    private static void runRule(KieContainer kieKontainer) {
        StatelessKieSession kSession = kieKontainer.newStatelessKieSession("testSession");
        kSession.setGlobal("out", System.out);
        kSession.execute("testRuleAgain");
    }
}
```

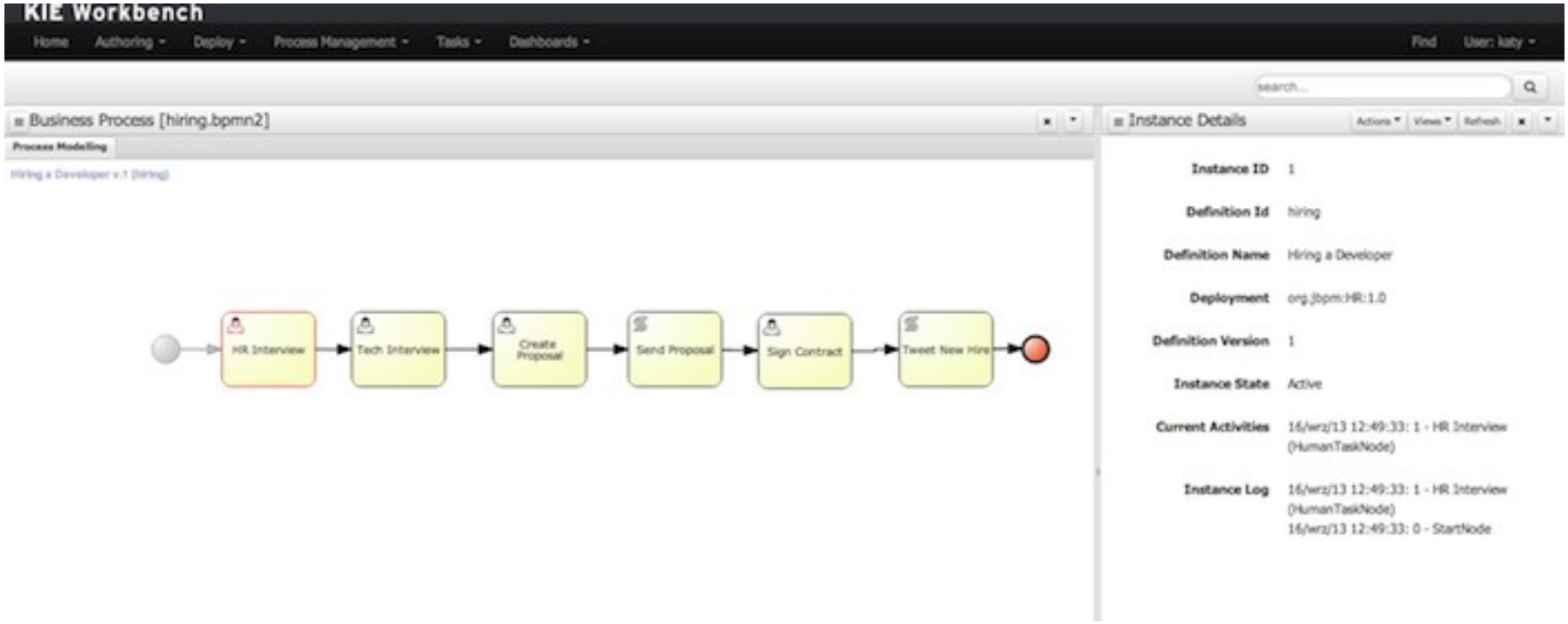
Web Editor


KIE Workbench
Home
Authoring
Deploy
Process Management
Tasks
Dashboards
Find
User: katy
Explore
New Item
Tools
Project Explorer
Business
Technical
Business Processes
evaluation
Form Definitions
Work Item Definitions
Business Process [evaluation.bpmn2]
Object Library
Process Modeling
Simulation Results
Self Evaluation
HR Evaluation
PM Evaluation
Properties (BPMN Diagram)
Core Properties
Extra Properties
Simulation Properties
Problems
Level
Text
File
Column
Line

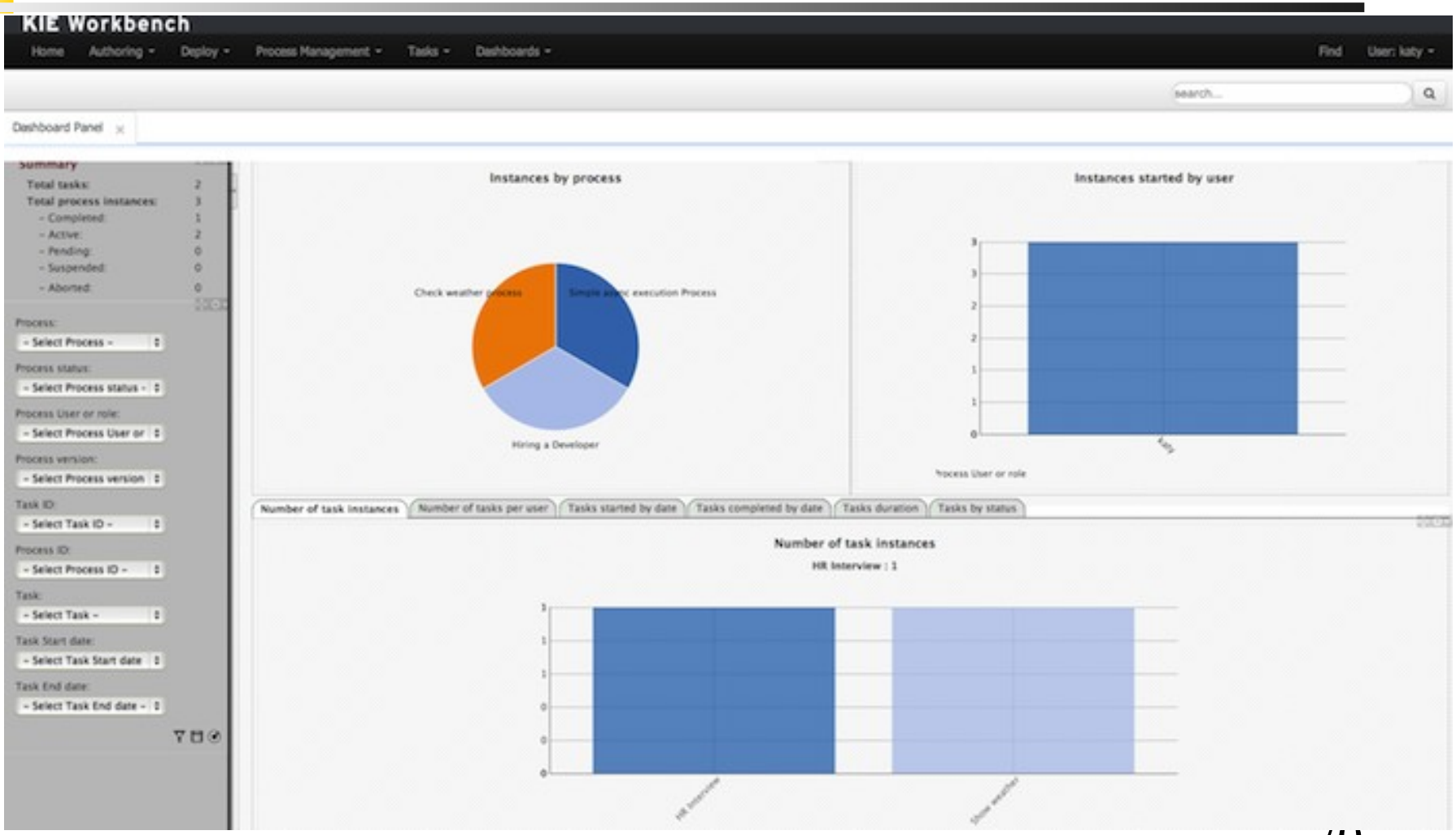
Data modelling



Process Instances Management



BAM Reporting (Birt)



Thank U!!!

- THANK YOU FOR YOUR ATTENTION