

## Instalando o ODM Eclipse toolkit – Parte 1

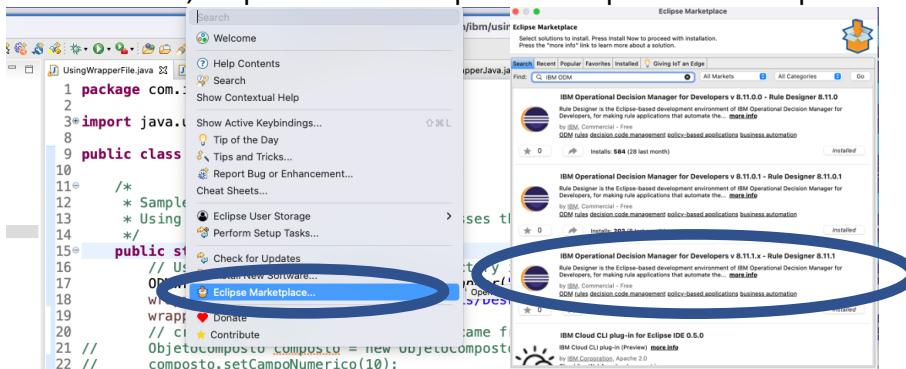
O ambiente de desenvolvimento do ODM se inicia a partir de um projeto Eclipse. Basta baixar o eclipse open source compatível com a versão do plugin. No momento da escrita deste artigo, a versão compatível é a 2020-06.



Neste link é possível encontrar a versão de Eclipse compatível com a implementação do plugin. <https://marketplace.eclipse.org/content/ibm-operational-decision-manager-developers-v-81100-rule-designer>

The screenshot shows the Eclipse Marketplace page for the 'IBM Operational Decision Manager for Developers v 8.11.0.0 - Rule Designer'. The page includes tabs for 'Details', 'Metrics', 'Errors', and 'External Install Button'. The 'Details' tab provides information about the plugin, including its key features like co-editing and debugging of Java code and rules, auto correction of rules, decision flow control, code-generation wizards, source code control integration, and conflict and redundancy detection. It also mentions that it is a collaborative work environment for developers to automate business policies for in-house or web-based use. The 'Additional Details' section highlights that the plugin is based on Eclipse 2020-06 (4.16). Other details include platform support for Windows, Mac, Linux/GTK, organization name (IBM), development status (Production/Stable), creation date (Thu, 2021-12-09 09:10), license (Commercial - Free), update date (Wed, 2022-09-21 09:20), and submitter (François Trible).

Instale o plugin. Você pode arrastar a partir da página anterior para o eclipse, quando estiver aberto, ou pode abrir o eclipse market place e buscar por “IBM ODM”



## Obtendo uma instância do Servidor para desenvolvimento – Derby – Parte 2.1

Para efeitos de desenvolvimento, é possível criar uma instância do ODM a partir de uma imagem Docker, com apenas uma linha de comando.

```
docker run -e LICENSE=accept -p 9060:9060 -p 9443:9443 -m 2048M --  
memory-reservation 2048M -e SAMPLE=false icr.io/cpopen/odm-  
k8s/odm:latest
```

Tenha certeza de que as portas 9060 e 9443 não estão sendo utilizadas na máquina. É possível alterar os valores em negrito para outros valores de portas disponíveis. Esta linha de comando inicia a execução e configura o servidor para execução em um Derby database. O banco de dados neste caso fica interno à imagem Docker e embedded no processo Java do Liberty.

## Obtendo uma instância do Servidor para desenvolvimento – Postgres – Parte 2.2

Também é possível criar uma instância de ODM utilizando um banco de dados externo Postgres. Neste caso, precisamos de duas imagens criadas, uma para o Postgres e outra para o servidor do ODM. **O recurso de permitir o acesso para a máquina hospedeira de dentro da imagem Docker do ODM é necessário para acessar o banco Postgres.** Este link a seguir fornece mais detalhes de como habilitar este acesso:

<https://medium.com/@TimvanBaarsen/how-to-connect-to-the-docker-host-from-inside-a-docker-container-112b4c71bc66>

As linhas de comando Docker que devem ser chamadas em sequência estão abaixo:

```
docker run -d -p 5432:5432 -e POSTGRES_DB=odmdb -e  
POSTGRES_USER=odmusr -e POSTGRES_PASSWORD=odmpwd postgres:9.5.10  
  
docker run -d -e LICENSE=accept -p 9061:9060 -p 9444:9443 -m 2048M --  
memory-reservation 2048M -e SAMPLE=false -e DB_TYPE=postgres -e  
DB_NAME=odmdb -e DB_USER=odmusr -e DB_PASSWORD=odmpwd -e  
DB_SERVER_NAME=host.docker.internal icr.io/cpopen/odm-k8s/odm:latest
```

Da mesma forma que o Derby, se certifique de que as portas do Postgres e ODM não estão em conflito. Neste caso, a imagem do Postgres deve ser criada antes da imagem do ODM.

Adicionalmente este GIT contém informações que podem ser úteis:

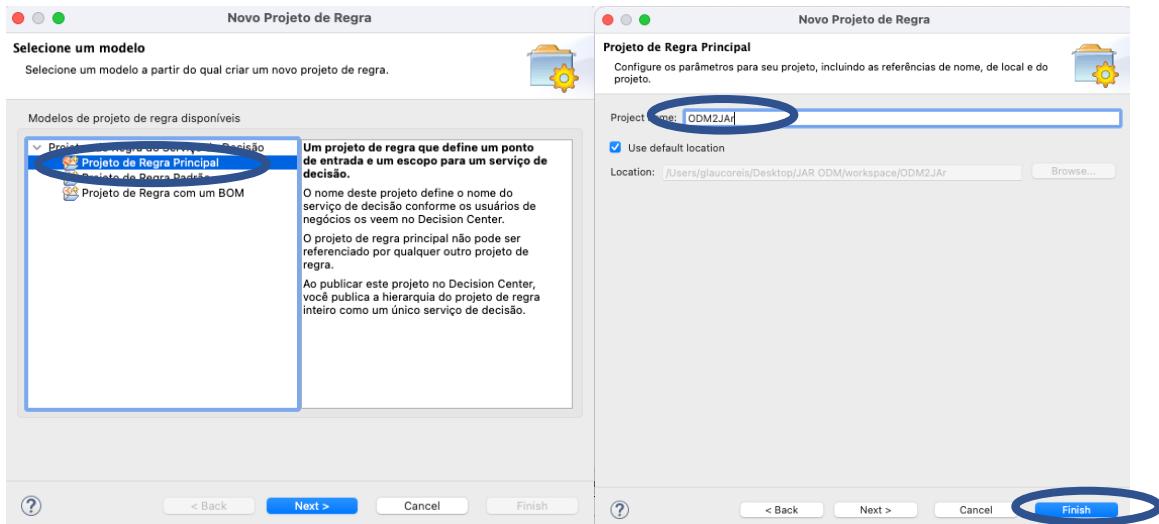
<https://github.com/DecisionsDev/odm-on-docker/blob/master/odm-standalone-postgres.yml>

## Criando um projeto no ODM – Parte 3

Com o Eclipse configurado, e com o servidor em execução na máquina, pode-se iniciar o projeto de regras no eclipse. A instalação do plugin é simples e foi apresentada na primeira sessão deste tutorial. Se instalado corretamente, deve ser possível abrir uma perspectiva Regras. Ela contém um passo a passo visual e interativo que auxilia no processo de criação do projeto, passo a passo. Ela tem esta aparência da foto :



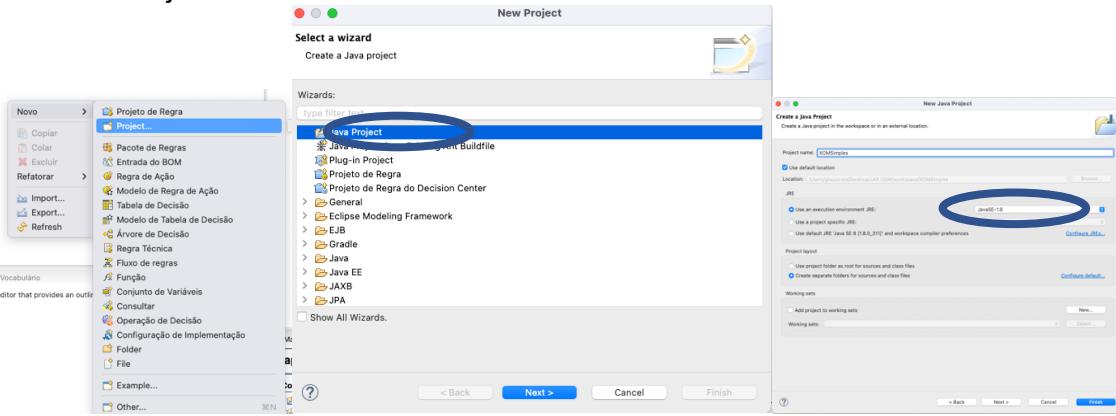
As etapas estão organizadas na ordem em que devem ser executadas para a construção do projeto. O primeiro passo é criar um “projeto de regra”. Ele pode ser um projeto principal, ou um projeto dependente. Como iremos criar um novo projeto, e portanto ele será um projeto principal.



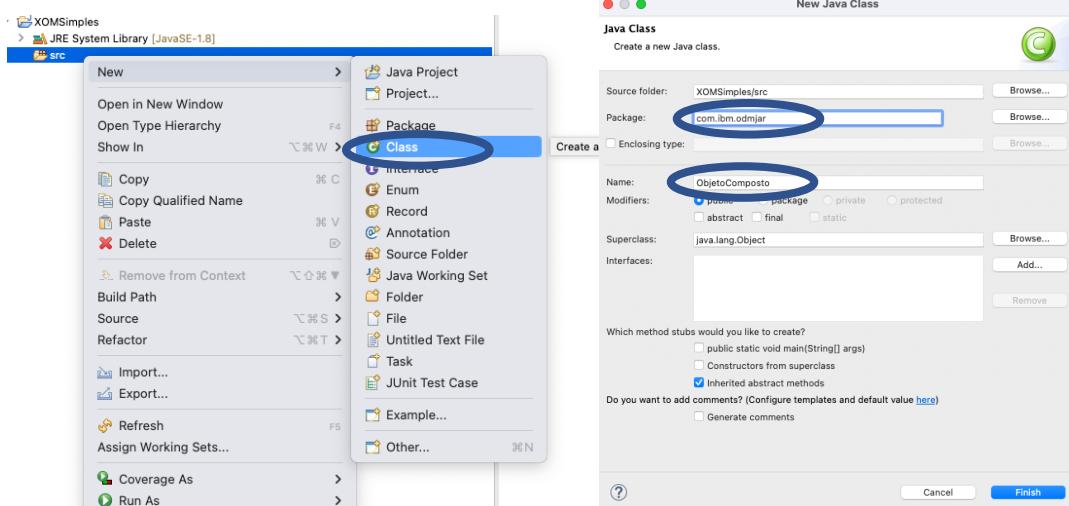
Se você observar na janela “Mapa do projeto de regra” verá que o item “Importar XOM” foi habilitado, após o projeto ter sido criado. Caso precise passar objetos complexos para a regra, elas devem vir dos códigos da TI. Eles podem ser fornecidos por uma classe Java, ou por uma especificação de XSD. No nosso caso, iremos criar uma classe Java com os campos que usaremos para a criação da regra.

Como um mecanismo auxiliar, no GIT de onde este documento foi baixado existe uma classe Java Simples que permite gerar um XOM a partir de um arquivo CSV. Consulte a pasta CSV2XOM para os fontes desta classe.

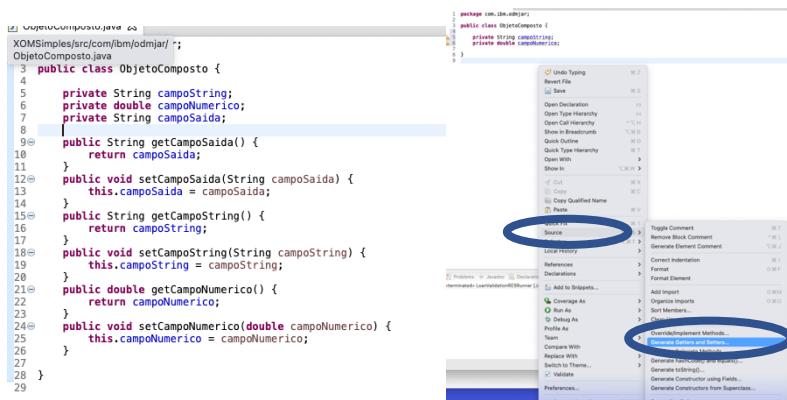
Para isto, crie um projeto Java. Com o botão da direita em “Package Explorer” selecione new Java Project :



Importante selecionar o JDK 1.8, se você tiver outras versões instaladas na máquina. Após isto, crie a classe com os campos necessários (ou utilize a classe auxiliar no GIT para gerar o Java a partir de um CSV). É recomendado em Java que cada classe fique em um package (não no default package).

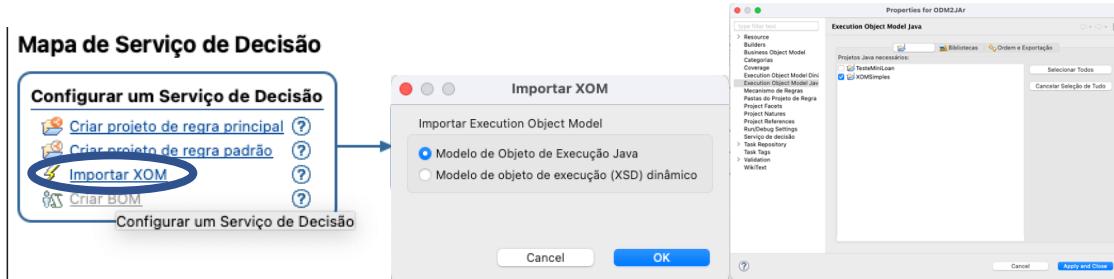


Coloque os atributos a serem expostos para a regra na classe. Eles devem espelhar os parâmetros que serão passados para a classe quando forem chamados externamente, por exemplo por um serviço REST ou SOAP. Também é importante que os getters e setters sejam gerados. Isto pode ser feito automaticamente pela ferramenta.

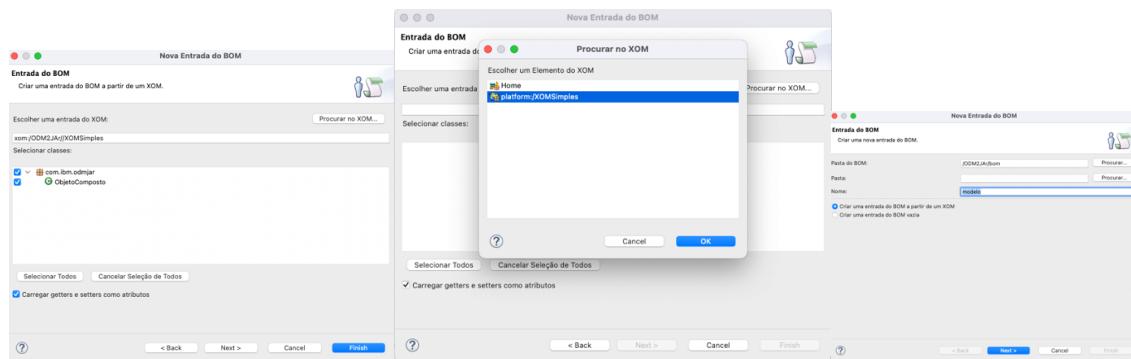


Importante ressaltar que não é mandatório criar um XOM. Se todos os tipos passados para a regra forem simples, como strings, números e datas, eles podem ser definidos diretamente dentro do Decision Center. Mas se os tipos forem complexos, como uma classe Java com atributos, eles precisam estar definidos dentro de um XOM.

Agora temos dois projetos, o de Regras criado no primeiro passo, e o projeto Java, com o XOM. Precisamos agora “conectar” o projeto Java ao projeto ODM. Isto é feito no menu importar o XOM.



Quando a importação for feita, o menu Criar BOM irá habilitar, e podemos agora verbalizar o XOM. O menu Criar BOM deve habilitar, e você deve clica-lo para criar a verbalização



A verbalização é uma etapa importante, que permite criar uma semântica sobre um objeto XOM. Ao invés de usar um linguajar de TI, a verbalização cria uma linguagem natural do negócio para a utilização da regra. Esta linguagem natural facilita a criação de regras pelo negócio e torna mais fácil encontrar problemas. Se abrir a pasta BOM até encontrar a classe, e abri-la com um clique duplo, encontrará os métodos e sua verbalização. Você pode criar a verbalização que achar mais conveniente.

The screenshot shows the ODM interface with two main windows. The left window is titled 'Membro campoSaida (classe: com.ibm.odmjar.IndexPath)' and displays the 'Informações Gerais' tab. It shows the member name 'campoSaida' and its class 'com.ibm.odmjar.IndexPath'. The right window is titled 'Classe IndexPath (pacote: com.ibm.odmjar)' and shows the 'Informações Gerais' tab with the class name 'IndexPath' and superclass 'java.lang.Object'. Both windows have sections for 'Verbalização de Membro' and 'Verbalização de Classe' with various configuration options like 'Remover a verbalização', 'Criar uma frase de navegação', and 'Criar uma frase de ação'. A navigation rule for 'campoSaida' is visible in the right window.

A partir deste ponto você tem duas alternativas. Pode continuar criando as regras a partir do Eclipse, com os outros menus em “[Mapa de Serviço de Decisão](#)” ou pode passar a criar as regras na interface Web do Decision Center. Para este tutorial, vamos criar as regras a partir do Decision Center. Precisamos portanto enviar o que criamos até agora para o servidor. Para isto, clique no botão da direita em Rule Explorer, com o projeto selecionado, e selecione conectar ao Decision Center.

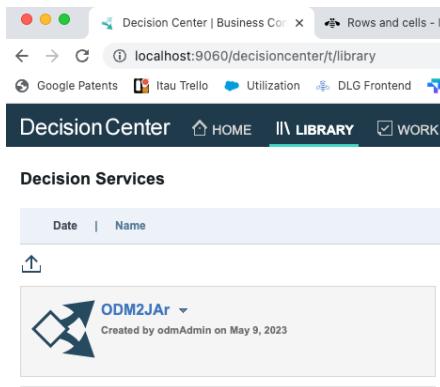
The screenshot shows the Eclipse IDE's Rule Explorer perspective. A context menu is open over a project named 'ODM2JAR'. The 'Decision Center' option is selected and highlighted with a blue oval. A confirmation dialog box is displayed at the bottom, asking 'Do you want to open this perspective now?' with 'Yes' and 'No' buttons. The 'Yes' button is circled with a blue oval.

Este servidor foi criado no passo 2, através da execução da imagem Docker. Se tiver dúvidas, abra um browser e digite localhost:9060 (user e pass odmAdmin como default).

The screenshot shows the 'Decision Center' interface in a web browser. The top navigation bar includes 'HOME', 'LIBRARY', 'WORK', and 'ADMINISTRATION'. The main area is titled 'Decision Services' and lists several items: 'RuleApp', 'ODM2JAR', 'Minican Rules', and 'Festa'. The 'RuleApp' item is highlighted with a blue oval.

## Criando uma regra dentro do Decision Center – Parte 4

Quando se conectar, ele irá perguntar se deseja chavear para a perspectiva de Sincronização. Ela permite saber o que está diferente entre o projeto no Eclipse e o projeto no Decision Center. Após sincronizado a primeira vez, no Decision center o projeto deve estar disponível.



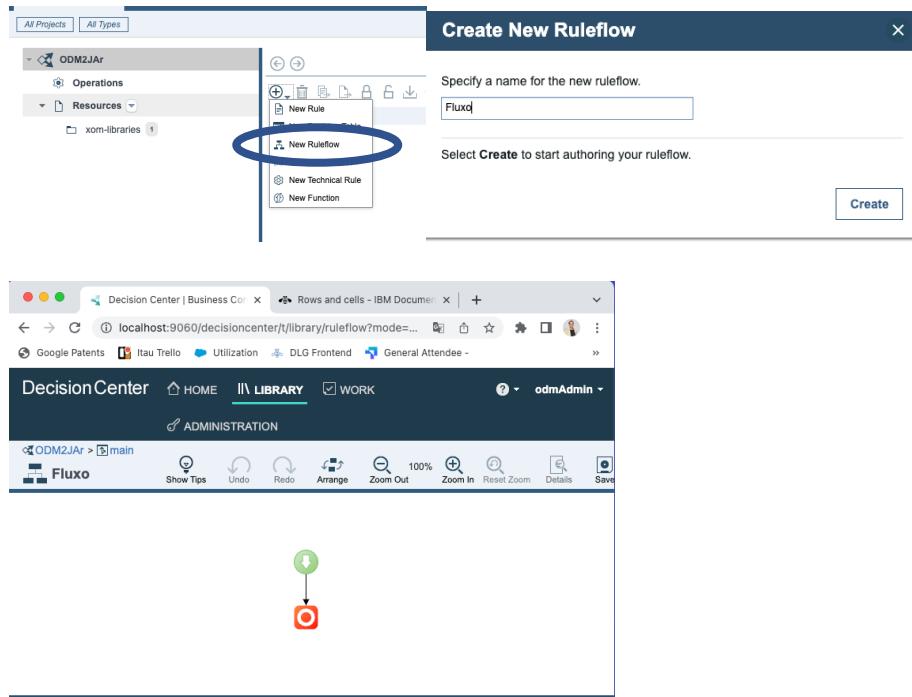
A partir deste ponto, caso nenhuma alteração for feita, o Decision center deve estar atualizado com o projeto do Eclipse. Podemos criar os outros artefatos da Regra na interface Web, ou podemos fazê-lo a partir do Eclipse, e depois sincronizar os dois. Selecione o projeto e a branch main, para entrar no editor Web. Se quiser validar se tudo está ok, clique no menu model. Ele mostrará em Vocabulary o Objeto Java que foi verbalizado anteriormente.

A screenshot of the 'Model' tab in the Decision Center. The top navigation bar includes 'LIBRARY' (selected) and 'ADMINISTRATION'. Below it, there are tabs for 'Decision Artifacts', 'Queries', 'Tests', 'Simulations', 'Deployments', 'Snapshots', and 'Model' (selected). Under 'Vocabulary', there are links for 'BOM Path' and 'Dynamic Domains'. A 'Filter:' input field and 'Collapse All'/'Expand All' buttons are present. The main area lists vocabulary entries under 'Name' and 'Categories'. One entry, 'objeto composto', is circled in red.

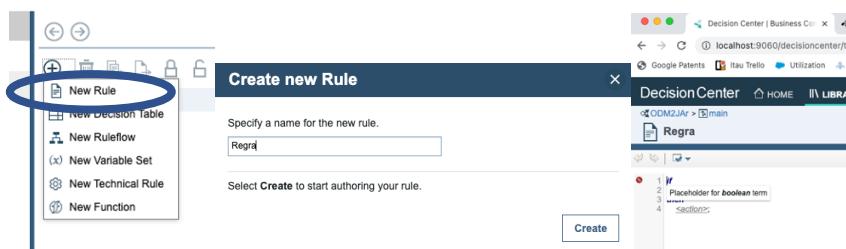
Cada um dos menus na parte superior apresentam um aspecto da ferramenta. Clicando em **Decision Artifacts** podemos criar novas regras. Clique em **Types** e selecione todos os artefatos, para limpar o filtro. Neste ponto, se clicar no nome do projeto e abrir a drop down, verá algo como:

A screenshot of the 'Decision Artifacts' tab in the Decision Center. The top navigation bar includes 'LIBRARY' (selected) and 'ADMINISTRATION'. Below it, there are tabs for 'Decision Artifacts', 'Queries', 'Tests', 'Simulations', and 'Deploy'. Under 'All Projects' and 'All Types', the project 'ODM2JAr' is selected. A dropdown menu for 'ODM2JAr' is open, showing options like 'Operations' and 'Resources'. A toolbar with icons for creating, deleting, and managing artifacts is visible.

Todo projeto de decisão é executado a partir de um Fluxo. Vamos criar nosso fluxo de decisão neste momento. Clicando na seta ao lado do sinal de mais, devem aparecer as opções. Iremos criar um RuleFlow.



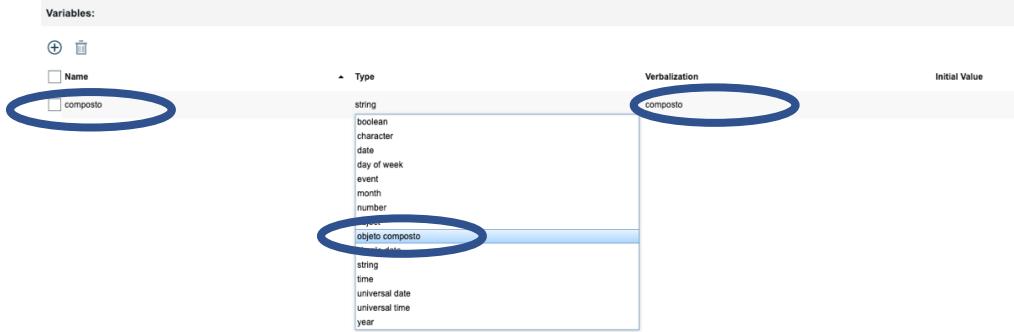
O Fluxo indica a sequência de etapas a executar. Para nosso exemplo, vamos criar uma regra simples, que valida se os parâmetros passados estão dentro de valores esperados. Salve o Fluxo e crie uma nova versão. Você deve retornar para a tela anterior. Clique novamente na seta e selecione New Rule. Dê um nome para ela, por exemplo regra.



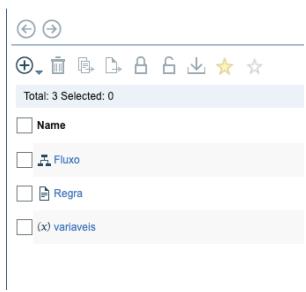
Salve novamente a regra sem alterá-la. Embora tenhamos criado o XOM e BOM, e eles apareçam em model, não definimos variáveis para serem utilizadas. O próximo passo é criar nossa variável para utilizar nos cálculos da regra. Novamente no menu, selecione “New variable set”.



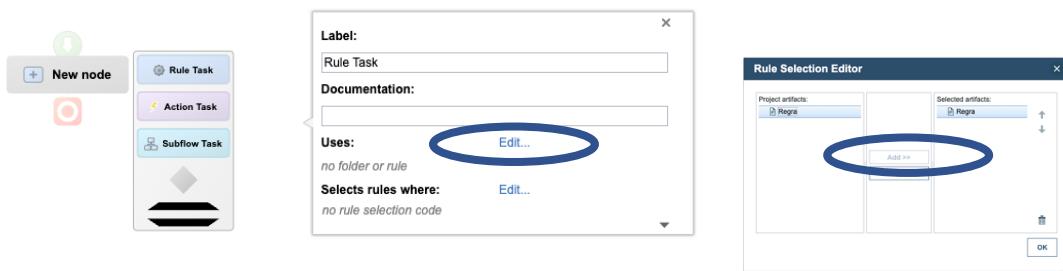
Dê um nome para ela, e crie uma variável com o nome composto. Selecione o tipo do XOM e salve esta variável criada. Preencha o campo nome. Perceba que é possível criar variáveis simples também, com vários tipos. Estas variáveis não dependem de um XOM, e podem ser utilizadas em qualquer cálculo.



Neste ponto, você tem um Fluxo, uma regra e uma variável criada. Nenhum deles se relaciona, e conectar todos será nosso próximo passo. O projeto deve estar desta forma aqui:

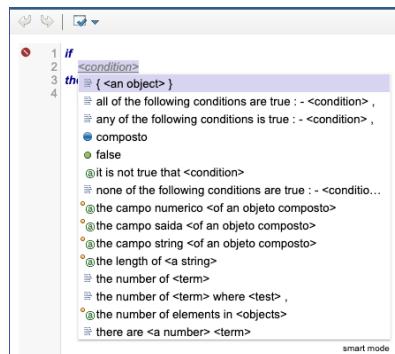


Abra o Fluxo, e clique em no lápis que aparece no canto superior direito. O Fluxo deve ocupar toda a tela, e está agora em modo edição. Clique na seta, e selecione new Rule Task.



Clique na regra, em Edit na frente de “Uses:” atribua a regra criada para este fluxo. Com isto, definimos que esta regra será executada toda vez que este fluxo for executado. Podemos ter vários fluxos e subfluxos, e dentro de cada Rule também podemos ter várias regras e decision tables selecionadas. Agora podemos editar nossa regra para fazer algum processamento. Pode clicar no projeto para mostrar a lista de itens, e clicar em Regra. Você precisa clicar no lápis no canto superior direito para editar a regra.

Cada linha tracejada quando clicada irá indicar sugestões de implementação. Clique por exemplo sobre <condition>, e um menu com sugestões aparecerá:



Conforme você clica em um item, novas linhas tracejadas aparecem, quando não aparecerem mais, você pode digitar Control+Space e novas opções devem aparecer.

Até agora a regra nos diz que o valor numérico deve ser no máximo 10. Faremos o mesmo com o valor String. Vá completando como explicado, até que tenhamos a regra:

Para esta regra, o valor numérico passado deve ser no máximo 10 e o valor String deverá ser igual a “umastring” para atribuirmos ao campo saída o valor acima definido. Perceba que o que aparece nesta tela é o que colocarmos na verbalização. Quanto melhor for a verbalização que criarmos, mais clara ficará a regra e mais fácil de detectar problemas se tornará no futuro, quando as implementações se tornarem complexas. Clique em Salvar. Assim criamos nossa regra. Os passos que fizemos até agora foram:

Criar fluxo e salvar

Criar Regra e salvar

Criar Variaveis e salvar

Atribuir Regra ao fluxo e salvar

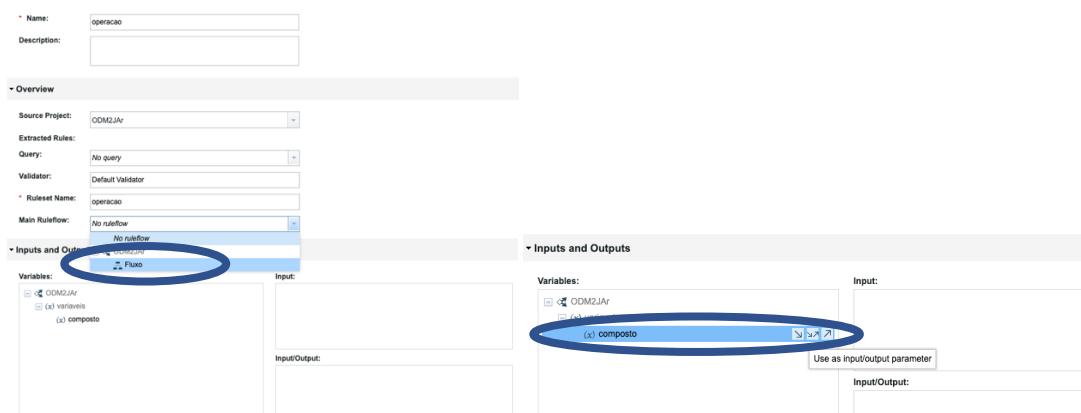
Editar regra e salvar

## Testando a regra dentro do Decision Center – Parte 5

Para testarmos a regra precisamos expô-la através de uma operação. No menu do projeto, clicamos em **Operations** e depois clicamos no sinal de soma dentro de um círculo.



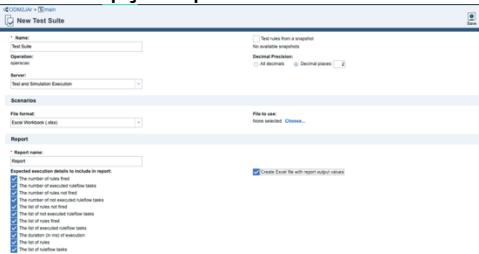
Os campos **Name** e **RuleSet Name** são obrigatórios e livres. Em Main RuleFlow selecionamos o fluxo que desejamos executar quando a operação for acionada.



Em variáveis, definimos as entradas e saídas. No fluxo somente definimos uma variável composta, e selecionamos ela como entrada e saída (o ícone do meio na frente de composto). Com a operação criada, podemos criar nosso teste. Muda para a ABA Tests, na parte superior do Decision Center, e clique no sinal de mais para criar um novo teste. O menu superior operação indica para qual operação desejamos criar um teste.



Outras opções podem ser selecionadas para o teste.



Podemos gerar um arquivo de testes padrão para ser preenchido. Selecionamos a regra, e clicamos na seta de download:

The screenshot shows the 'Test Suites' interface. At the top, there are tabs for 'Test Suites' and 'Reports'. Below the tabs, there is a message: 'Running tests against your system to make sure it works with what you expected. Scenario data and expected results are stored in scenario files.' There are several icons: a plus sign, a trash can, a download arrow (circled in blue), a file icon, and a refresh icon. Below these icons, it says 'Total: 1 Selected: 1'. Under the heading 'Recent', there is a table with two rows. The first row has a checked checkbox next to 'Name', the value 'operacao', and the status 'Recent'. The second row has a checked checkbox next to 'Test Suite', the value 'operacao', and the status 'Not run'.

Escolhemos a variável que será parte do teste, e então fazemos download:

The screenshot shows the 'Generate Scenario File' configuration page. At the top, there is a breadcrumb navigation: '< ODM2JAR > main'. Below it, there is a title 'Generate Scenario File' and a 'Download' button. The main area contains fields for 'Filename' (Scenario File - 2023-05-09\_04-52-02), 'Scenario file format' (Excel Workbook (.xlsx)), and 'Locale' (English (United States)). Under the heading 'Operation:', there is a single entry 'operacao'. Below this, there is a section titled 'Select the tests to include in the scenario file.' with a 'Tests:' heading. Two checkboxes are checked: 'Field' and 'composto'. A blue oval highlights the 'Tests:' section.

Deve ser gerada uma planilha Excel com o formato:

The screenshot shows a generated Excel spreadsheet. The first tab, 'scenarios', contains the following table:

Scenario ID	description	campo numerico	campo saida	campo string
Scenario 1		5		umastring

The second tab, 'Expected Results', contains the following table:

Scenario ID	the campo numerico composto equals	the campo saida composto equals	the campo string composto equals
Scenario 1		Menor que 10 e umastring	

A planilha define cenários. Cada linha na aba “scenarios” define os valores de entrada, e está relacionada com a aba “Expected Results”. Podem ser criadas várias linhas, bastando enumerá-las como indicado. Nossa teste deve ser positivo, porque passamos um valor menor que 10 e o valor esperado em campo string. Podemos retornar para a aba Tests, e clicar no lápis que aparece ao lado do teste, quando flutuamos o mouse na frente do nome. Em “File to use” escolhemos a planilha Excel que acabamos de preencher.

The screenshot shows the 'Test Suite (v1.0)' configuration screen. In the 'File to use' section, there is a dropdown menu set to 'Scenario File - 2023-05-09\_04-52-02.xlsx' and a 'Choose...' button. A blue oval highlights this area.

Agora basta clicar em **Save and Run** para o teste ser executado. Será apresentada uma tela com os resultados, regras que foram executadas e outras informações.

The screenshot shows the 'Report - 2023-05-09\_05-02-21-450' page. It includes a summary section with a green circle indicating 100% success rate, and a 'Results' table with two rows highlighted by blue ovals. The first row has 'Result' and 'Expected' columns both containing 'Menor que 10 e umastring'. The second row also has 'Result' and 'Expected' columns both containing 'Menor que 10 e umastring'.

Se clicar em “**Details**” em “**The List of Rules Fired**”, indicará detalhes da regra executada:

The screenshot shows the 'The list of rules fired' dialog. It displays the 'Execution sequence' tab, which lists 'Fluxo' and 'rule\_0' under 'Regra'. Below this, the 'Preview Regra' section shows the rule definition:

```

If
the campo numerico of composto is at most 10
and the campo string of composto is "umastring"
then
atribuir o campo saida of composto a "Menor que 10 e umastring";

```

Agora que nossa regra está operacional, e funcionando no ambiente de testes do Decision Center, podemos publicá-la no servidor do Decision Server. Por enquanto apenas testes locais foram feitos com ela.

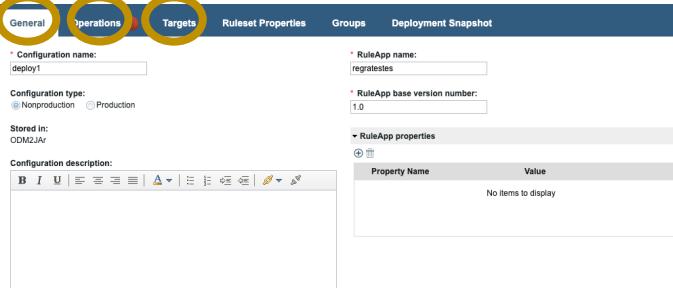
## Publicando a regra no Decision Server – Parte 6

O ODM está dividido em dois ambientes, um ambiente de construção e testes (**Decision Center**) e um ambiente de execução real (**Decision Server**), composto por um servidor. Na aba superior, temos um item **Deployments**. Vamos criar um deploy em um servidor:



The screenshot shows the ODM main interface with the 'Deployments' tab highlighted. Below the tabs, there's a message: 'To deploy this branch, use the listed configurations.' A table lists deployment configurations with columns for Name, Type, Operations, Target Servers, and Groups. A blue circle highlights the '+' icon in the first column.

Cada Deploy é formado por uma sequência de passos que podem ser preenchidos para definir os vários aspectos da execução. Quando clicado em publicação, uma nova sequência de abas precisa ser preenchida:



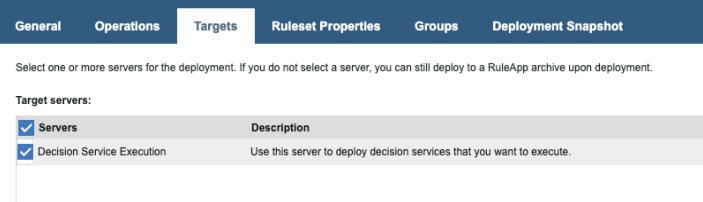
The screenshot shows the 'New Deployment Configuration' dialog with the 'General' tab selected. Other tabs like 'Operations' and 'Targets' are circled in yellow. The General tab contains fields for Configuration name (deploy1), Configuration type (Nonproduction), and RuleApp properties.

Na aba General precisamos definir o nome da configuração e o nome da APP a executar. A versão também pode ser definida neste momento. Na sequência definimos a operação a executar:



The screenshot shows the 'New Deployment Configuration' dialog with the 'Operations' tab selected. It displays a list of operations: 'Operations' and 'operacao', with 'Operations' checked.

Em Targets, escolhemos o servidor para onde a regra será publicada.



The screenshot shows the 'New Deployment Configuration' dialog with the 'Targets' tab selected. It displays a list of target servers: 'Servers' and 'Decision Service Execution', both of which are checked.

Iremos neste momento publicar a regra em um servidor.

Specify how to deploy the branch.

Deployment configuration:  
deploy1 (v1.0)

Target:  
 Server  
Decision Service Execution  
 RuleApp archive

Ruleset versions:  
Increment minor ruleset version numbers.  
RuleApp: regatesetes/1.0

Rulesets	Base Version	Latest Deployed Version	New Version
operacao	1.0	N/A	1.0

Include debug information

Uma vez selecionada as opções, podemos salvar, mas ainda é necessário publicar para o servidor. O ícone de “anzol” permite fazer o deploy.

To deploy this branch, use the listed configurations.

Total: 1 Selected: 0

Name	Type
deploy1	Nonproduction

Configurations | Reports

This table lists the deployment reports and the current deployments for this branch

Total: 1 Selected: 0

Name	Status	Run By
Report 2023-05-09_1732...	<input checked="" type="checkbox"/>	odmAdmin

Podemos fazer testes com a regra no ambiente servidor, o que será feito no próximo passo.

## Testando a regra no Servidor – Parte 7

O console do Servidor está acessível em outra URL.

<http://localhost:9060/res/>

Este console será utilizado pelo time de infraestrutura, para controlar o servidor e gerenciar sua execução.



### Welcome to the Rule Execution Server console

- |                    |  |
|--------------------|--|
| Explorer           | Use the Explorer to deploy, browse, and modify RuleApps.   |
| Decision Warehouse | Search and view decision traces.   |
| Diagnostics        | Run the server diagnostics to verify installation.   |
| Server Info        | View server configuration information and logged events.   |
| REST API           | Access the test tool for the resource management REST API. Use this tool to format and send requests, and to view responses. |

Se clicarmos em Explorer, podemos verificar as regras publicadas. Em RuleApps encontramos nossa aplicação, e em Libraries estão publicados nossos XOMs. Estes dois itens são importantes para a execução, e serão necessários quando formos executar a regra a partir de um programa Java externo.

A screenshot of the Ruleset View page. The left sidebar shows a tree structure with RuleApps, Resources, and Libraries. The "operacao/1.0" node under RuleApps is selected and highlighted with a blue oval. The main content area displays details for the "/regratestes/1.0/operacao/1.0" ruleset, including its name, version, creation date, display name, description, rule engine, status, and debug settings. A "Ruleset Parameters" table is also shown below.

A screenshot of the Library View page. The left sidebar shows a tree structure with RuleApps, Resources, and Libraries. The "regrestes\_library/1.0" node under Libraries is selected and highlighted with a blue oval. The main content area displays details for the "regratestes\_library/1.0" library, including its name, version, creation date, and URI. Below this, a table lists "Referenced Internal Resource(s)" with one entry: "XOMSimples.zip".

Iremos agora executar um teste com a regra no servidor. Selecionando RuleApps->regatestes->operação, que foram os nomes dados no momento do deploy, clicamos no item do menu “Retrieve HTDS Description File”.

### Ruleset View

The screenshot shows the 'Ruleset View' interface. At the top, there is a navigation bar with several icons and links: 'Test Ruleset', 'View Statistics', 'View Execution Units', 'Upload Ruleset Archive', 'Add Managed URI', 'Add Property', 'Edit', and 'Retrieve HTDS Description File'. The 'Retrieve HTDS Description File' button is circled in blue. Below the navigation bar, the URL is displayed as '/regatestes/1.0/operacao/1.0'.

Nesta tela podemos exportar a definição da regra para ser utilizada com um programa externo, como o Postman. Podem ser exportados em formato SOAP ou REST, e um Swagger também está disponível. No momento, iremos testar, e portanto clicamos na opção Test.

The dialog box has a title 'Retrieve HTDS Description File' and a URL '/regatestes/1.0/operacao/1.0'. It contains a 'Service protocol type' section with 'REST' selected. Below it is a 'Format' dropdown set to 'WADL'. There are four checkboxes: 'Latest ruleset version', 'Latest RuleApp version', 'Decision trace information', and 'Inline types in separate XSD files'. At the bottom are buttons for 'Cancel', 'View', 'Download', and 'Test', with 'Test' highlighted by a blue oval.

Será apresentada uma tela composta por uma área de entrada, preenchida automaticamente, e uma área de saída, com a resposta da execução.

#### Decision Service : /regatestes/1.0/operacao/1.0 REST Service

The screenshot shows the 'Decision Service' interface. At the top, it says 'Execution Request: XML' with a dropdown arrow. Below it is a large text area containing an XML request:

```

<par:Request xmlns:par="http://www.ibm.com/rules/decisionservice/Regatestes/Operacao/param">
  <!--Optional:-->
  <par:DecisionID>string</par:DecisionID>
  <!--Optional:-->
  <par:composto>
    <!--Optional:-->
    <campoNumerico>5</campoNumerico>
    <!--Optional:-->
    <campoSaida>string</campoSaida>
    <!--Optional:-->
    <campoString>umastring</campoString>
  </par:composto>
</par:Request>

```

Below the request is a 'Validation results:' section with a 'Execute Request' button. At the bottom, it shows a 'Server Response' XML output:

```

<?xml version="1.0" encoding="UTF-8"?><par:Response xmlns:par="http://www.ibm.com/rules/decisionservice/Regatestes/Operacao/param">
<par:DecisionID>string</par:DecisionID>
<par:composto>
  <campoNumerico>5.0</campoNumerico>
  <campoSaida>Menor que 10 e umastring</campoSaida>
  <campoString>umastring</campoString>
</par:composto>
</par:Response>

```

Podemos colocar valores conhecidos para o campo numérico e o campo String e então clicamos em Execute Request. Veremos a resposta enviada, no formato solicitado.

De forma similar, podemos testar as requisições em formato JSON, selecionando a opção YAML na tela anterior.

Decision Service : /regratestes/1.0/operacao/1.0 REST Service

Execution Request:

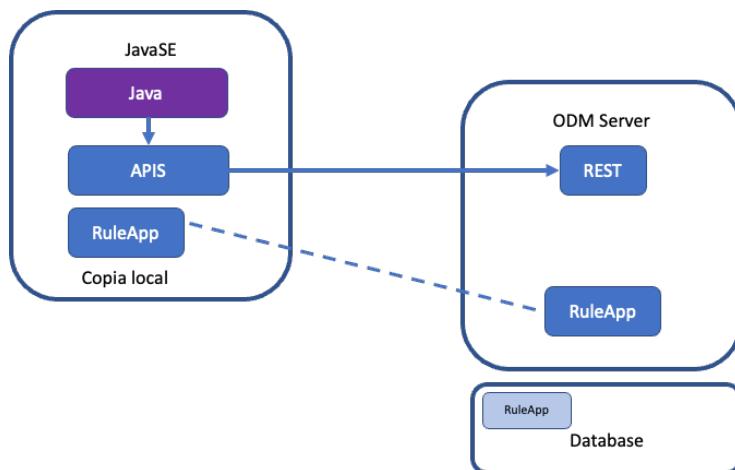
```
1 {  
2   "__DecisionID__": "string",  
3   "composto": {  
4     "campoString": "umastring",  
5     "campoNumerico": 7,  
6     "campoSaida": "string"  
7   }  
8 }
```

Server Response:

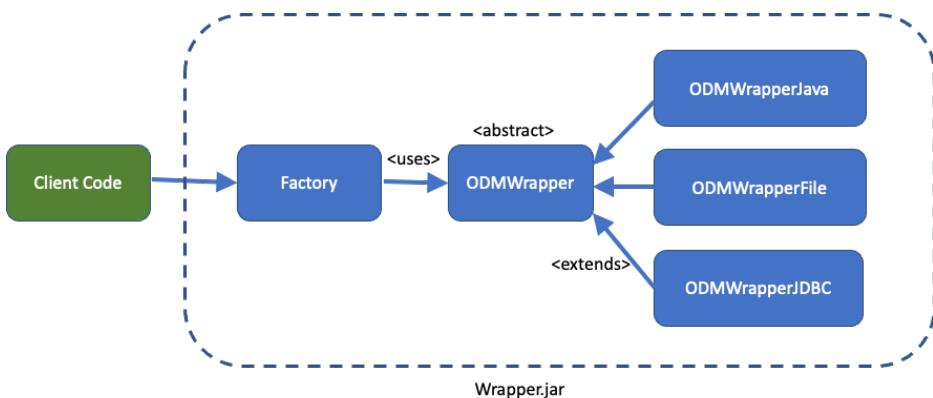
```
{  
  "composto": {  
    "campoString": "umastring",  
    "campoNumerico": 7,  
    "campoSaida": "Menor que 10 e umastring"  
  },  
  "__DecisionID__": "string"  
}
```

## Executando regras em um programa Java externo – Parte 8

Embora o servidor seja extremamente conveniente, permitindo administrar de forma visual as regras, algumas vezes desejamos ter a máxima performance. Desejamos executar regras como uma biblioteca diretamente a partir do Java, sem todo Workload de uma requisição REST ou SOAP. Toda implementação da regra pode ser exportada e executada em um programa Java externo. Como a chamada à regra é direta e não depende de REST ou SOAP, tornamos as execuções muito mais rápidas. Existem três formas de executar uma regra como um programa externo. Exploraremos aqui estas três opções. Toda implementação aqui apresentada foi encapsulada em um conjunto de classes Wrapper, que simplificam o uso e tornam a explicação mais simples. As regras podem ser exportadas do Decision Center ou do Decision Server. O formato de export é um JAR, que encapsula as regras e pode ser executado pelas bibliotecas do ODM.



O processo consiste em criar um programa Java, utilizar um dos mecanismos de acesso à regra, preparar as entradas de dados e executar a regra. Como existe alguma complexidade no acesso às APIs, foram criadas classes “wrapper” que simplificam o acesso às regras. Os códigos fontes estão disponíveis na pasta “wrapper”, e o JAR disponível na raiz do GIT de onde este documento foi baixado (<GIT-address>). Um desenho mínimo de arquitetura de uso das classes está apresentado a seguir.



## Executando regras em um programa Java como parte do classpath – Parte 8.1

Nesta modalidade, as regras devem ser exportadas do Decision Center ou Decision Server e colocadas no classpath para que sejam executadas. Utilizaremos as classes Wrapper para um pequeno exemplo de uso. Baixe a biblioteca wrapper.jar do GIT <GIT-address>. Acesse a imagem Docker criada no passo 1, e baixe os arquivos:

- j2ee\_connector-1\_5-fr.jar
- jrules-res-8.11.1.0-execution.jar
- jrules-engine-8.11.1.0.jar

Com a linha de comando “**Docker ps**” é possível obter o ID da imagem do ODM em execução:

```
glauco@Glauco-MacBook-Pro-5 ~ % docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
786442d1a7f0        icr.io/open/odm-k8s/odm:latest   "/script/runserver.s..."   3 hours ago       Up 3 hours          9080/tcp, 9453/tcp, 0.0.0.0:9861->9860/tcp, 0.0.0.0:9444->9443/tcp
8ec00244a55b        postgres:9.5.14      "docker-entrypoint.s..."   3 hours ago       Up 3 hours          0.0.0.0:5432->5432/tcp
```

Podemos copiar arquivos de dentro da imagem Docker para o exterior, e utilizarmos no classpath do programa java que criaremos. As linhas de comando a seguir indicam os comandos de cópia:

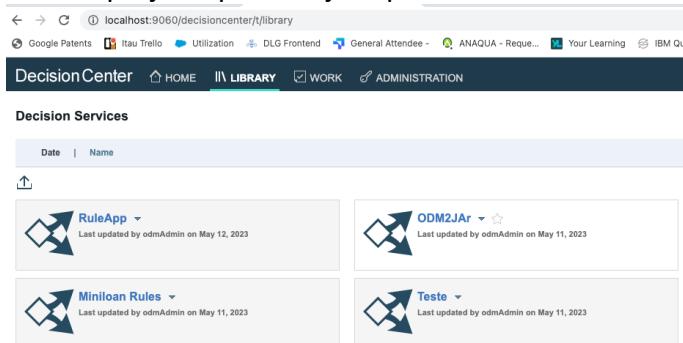
```
glauco@Glauco-MacBook-Pro-5 ~ % docker cp 786442d1a7f0:/opt/ibm/wlp/usr/servers/defaultServer/apps/decisioncenter.war/WEB-INF/lib/jrules-engine-8.11.1.0.jar ~/Desktop
glauco@Glauco-MacBook-Pro-5 ~ % docker cp 786442d1a7f0:/opt/ibm/wlp/usr/servers/defaultServer/apps/res.war/WEB-INF/lib/j2ee_connector-1_5-fr.jar ~/Desktop
glauco@Glauco-MacBook-Pro-5 ~ % docker cp 786442d1a7f0:/opt/ibm/wlp/usr/servers/defaultServer/apps/DecisionService.war/WEB-INF/lib/jrules-res-8.11.1.0-execution.jar ~/Desktop
glauco@Glauco-MacBook-Pro-5 ~ %
```

Caso algum dos comandos apresentar erro, é possível que alguma versão de biblioteca esteja em um diretório diferente do apontado. Neste caso, digite o comando

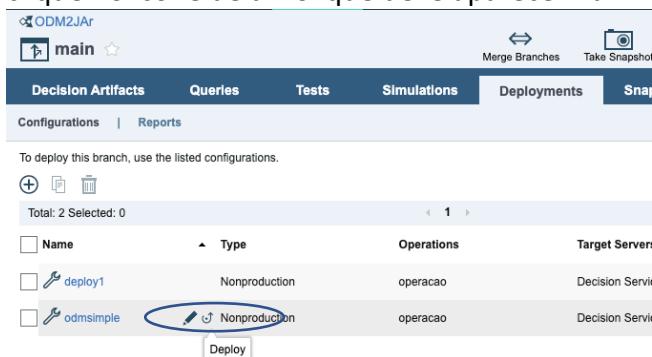
```
docker exec -it 786442d1a7f0 /bin/bash
find . -name jrules-res-8.11.1.0-execution.jar
```

A partir deste ponto você deve ter os três arquivos da lista acima, e o arquivo ODMwrapper.jar do GIT. Ainda precisa do arquivo de regras e do XOM. Vamos obtê-los do Decision Center ou do Decision Server.

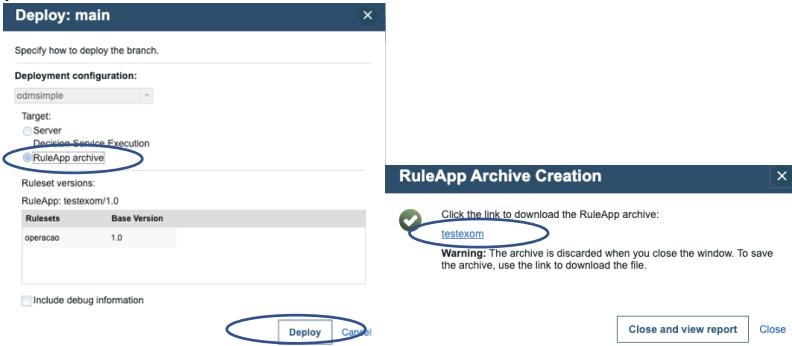
Abra o projeto que deseja exportar no Decision Center:



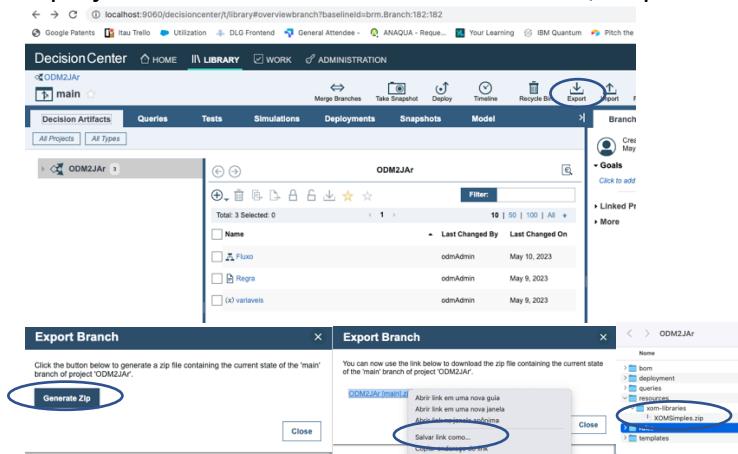
Clique na ABA Deployments e com o mouse na frente do projeto que deseja explorar, clique no ícone de anzol que deve aparecer na linha



Indique que deseja exportar para um RuleApp archive e clique em deploy. Um link para o download das regras deve aparecer e você pode baixa-lo para seu diretório de preferência:



O XOM, que contém as classes de Objetos complexos (criada no projeto Java da Etapa 3 deste Doc) pode ser obtida de dentro do Projeto. De dentro do Decision Center, com o projeto aberto na aba “Decision Artifacts”, clique no ícone Export:

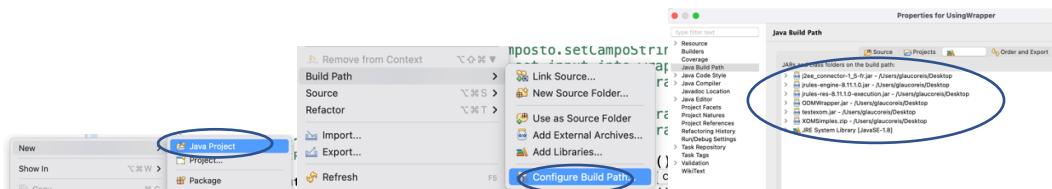


Ele deve baixar um arquivo ZIP, que uma vez expandido deve conter um diretório resources -> xom-libraries e dentro dele um ZIP com as classes.

Neste momento temos seis arquivos disponíveis:

- j2ee\_connector-1\_5-fr.jar (extraido do Docker)
- jrules-res-8.11.1.0-execution.jar (extraido do Docker)
- jrules-engine-8.11.1.0.jar (extraido do Docker)
- testexom.jar (extraido do Decision Center)
- XOMSimple.zip (extraido do ZIP do projeto, baixado do Decision Center)
- ODMWrapper.jar (baixado do GIT deste projeto)

No Eclipse, crie um novo projeto Java e coloque todos os seis arquivos no build path:



Agora basta criar uma Classe Java que faça a chamada às classes Wrapper.

```
1 package com.ibm.usingwrapper;
2 import com.ibm.odmjar.ObjetoComposto;
3 import com.ibm.odmwrapper.FactoryODM;
4 import com.ibm.odmwrapper.ODMWrapper;
5
6 public class UsingWrapperJava {
7
8     /*
9      * Sample using JAR to run ODM Rule
10     * Using ODMWrapper, a utility set on Classes that hide some of complexity of ODM usage with Java
11     */
12    public static void main(String[] args) {
13        /*
14         * Use factory to return Wrapper - Factory is a singleton
15         * ODMWrapper wrapper = FactoryODM.getWrapper("Java", "/testexom/operacao");
16         * wrapper.initialize();
17         * ObjetoComposto composto = new ObjetoComposto();
18         * composto.setCampoNumerico(10);
19         * composto.setCampoString("umastring");
20         * wrapper.putInputParameter("composto", composto);
21         * try {
22         *     wrapper.execute();
23         * } catch (Exception e) {
24         *     wrapper.getError().printStackTrace();
25         * }
26         * System.out.println("Rules fired " + wrapper.getNumberFiredRules());
27         * System.out.println(composto.getCampoSaida());
28     }
29 }
```

No Git também se encontra um JavaDOC para uso das classes Wrapper. O processo básico para seu uso consiste de :

- Chamada ao método `FactoryODM.getWrapper("java", "testexom/operação")`. A string Java indica que a carga será feita pelo Classloader local. A String da regra pode ser obtida no Decision Server, na aba Explorer, ou abrindo-se o arquivo JAR com as regras e obtendo-se o caminho de dentro dele



- A chamada `initialize()` estabelece a comunicação e inicialização do motor de regras
- ObjetoComposto foi a classe criada no momento do projeto Java, e os valores são atribuídos como qualquer programa Java. Elas estão dentro do arquivo ZIP exportado.
- O método `putInputParameter()` permite inserir parâmetros de entrada
- O método `execute()` estimula o motor de regras. Este método pode ser chamado diversas vezes para tornar o processamento ágil
- Podemos obter parâmetros de saída com o método `wrapper.getInputParameter()`

Outra forma de fazer download dos arquivos de regras e de XOM é através do console do Decision Server. Na aba Explorer, podemos selecionar a RuleApp e clicar em download.

**RuleApp View**

**/regretates/1.0**

Name	Version	Creation Date	Display Name	Description
regretates	1.0	May 9, 2023, 12:32:50 PM GMT-3		

**1 Ruleset(s)**

Name	Version	Ruleset Path	Creation Date
operacao	1.0	/regretates/1.0/operacao/1.0	May 9, 2023, 12:32:50 PM GMT-3

**Download**

O XOM também pode ser baixado do Decision Server, em Libraries, logo abaixo do link para baixar a implementação da regra.

**Library View**

**/regretates\_library/1.0**

Name	Version	Creation Date	URI
regretates_library	1.0	May 9, 2023, 12:32:47 PM GMT-3	restfb://regretates_library/1.0

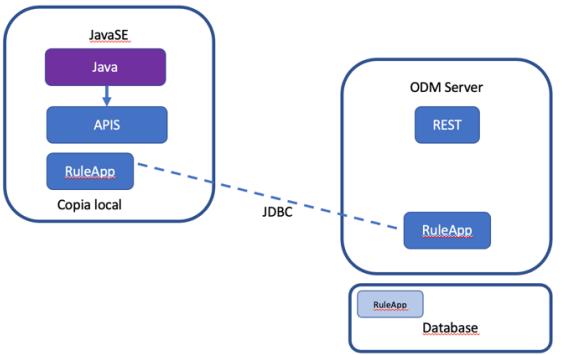
**Referenced Internal Resource(s)**

Index	Name	Version	Creation Date	Checksum
1	XOMSimple.zip	1.0	May 9, 2023, 12:03:01 PM GMT-3	16b76600943091c87a7cdafbd524ecdd4f6ca28

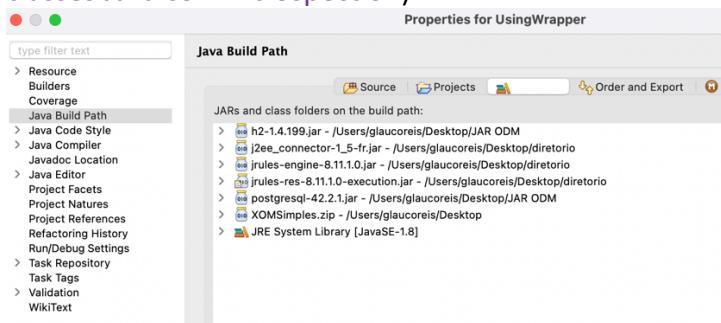
**Download Resource**

## Executando regras baixadas do banco de dados – Parte 8.2

Outra modalidade de execução de regras é buscando as regras e os XOMs diretamente do banco de dados do motor de regras. Nesta modalidade estes dois arquivos serão baixados por um Classloader interno do ODM, que traz os arquivos diretamente do banco e os carrega no Classloader do Java, no momento da execução.



Para esta modalidade, não precisamos apontar o JAR e XOM no classpath ([lembrando que caso o XOM não seja apontado, é necessário se utilizar de carga dinâmicas de classes Java com Introspection](#)).



O uso das classes wrapper podem ser feitos de forma muito parecida, somente mudando o parâmetro para JDBC.

```
1 package com.ibm.usingwrapper;
2 import com.ibm.odmjar.ObjetoComposto;
3 import com.ibm.odmwrapper.FactoryODM;
4 import com.ibm.odmwrapper.ODMWrapper;
5
6 public class UsingWrapperJDBC {
7=   public static void main(String[] args) {
8=     // Use factory to return Wrapper - Factory is a singleton
9=     ODMWrapper wrapper = FactoryODM.getWrapper("JDBC", "/testexom/operacao");
10    // set parameters for Database Access
11    wrapper.setJDBCDriverClassName("org.h2.Driver");
12    wrapper.setJDBCURL("jdbc:h2://Users/glaucoreis/Desktop//JAR ODM//dbdata//resdb;" +
13      "MODE=HSQldb;LOCK_TIMEOUT=40000;auto_server=true");
14    wrapper.setJDBCUser("res");
15    wrapper.setPassword("res");
16
17    // Create your input and set values (came from XOM - should be in classpath)
18    ObjetoComposto Composto = new ObjetoComposto();
19    Composto.setCampoNumerico(10);
20    Composto.setCampoString("umestring");
21    wrapper.putInputParameter("composto", Composto);
22
23    try {
24      wrapper.execute();
25    } catch (Exception e) {
26      wrapper.getError().printStackTrace();
27    }
28    // If ok, you can get the number of rules fired
29    System.out.println("Rules fired " + wrapper.getNumberFiredRules());
30    System.out.println(Composto.getCampoSaida());
31
32  }
33}
```

Alguns parâmetros adicionais precisam ser configurados, como o driver do banco de dados, a URL de conexão, usuário e senha do banco. Para o banco Derby, como fica armazenado em arquivos do sistema operacional, precisam ser baixados da imagem Docker. Isto pode ser feito através do comando:

```
docker cp 8d91b0d54ad7:/opt/ibm/wlp/usr/servers/defaultServer/dbdata/ ~/Desktop
```

Com o derby baixado, modifique o parâmetro de setJDBCURL para o caminho da pasta para onde o banco do ODM foi baixado.

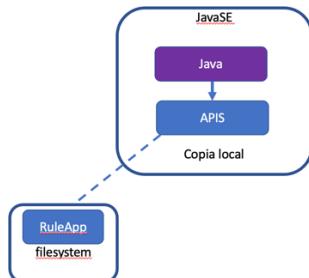
```
INFORMAÇÕES: Conjunto de repositórios XUM no modo de persistência de banco de dados: H2 1.4.199 (2019-03-13)
mai 15, 2023 5:39:08 PM com.ibm.rules.res.logging.internal.RESLogger log
INFORMAÇÕES: Solicitado para /testexom/1.0/operacao/1.0 ClassLoader (Último ClassLoader do Cliente):
\-\-/testexom/1.0/operacao/1.0
\-\-/reslib://testexom_library/1.0
\-\-/resuri://XOMSimple.zip/1.0
.
mai 15, 2023 5:39:08 PM com.ibm.rules.engine.load.XUEngineDynamicLoaderBuilder buildFromClassLoader
INFORMAÇÕES: Build XUEngineDynamicLoaderBuilder from class loader
mai 15, 2023 5:39:09 PM com.ibm.rules.engine.load.XUEngineDynamicLoaderImpl createEngineDefinitionFromDSAR
INFORMAÇÕES: Create engine definition from DSAR
Rules fired 1
Menor que 10 e uma string
```

Se o banco de dados for do tipo postgres, não é necessário baixar o banco, basta apontar para a URL e porta do banco (configurado na etapa 2.2 deste tutorial)

```
1  /*
2   * Sample using JAR to run ODM Rule
3   * Using ODMWrapper, a utility set up Classes that hide some of complexity of ODM usage with
4   */
5  public static void main(String[] args) {
6      // Use factory to return Wrapper - Factory is a singleton
7      ODMWrapper wrapper = FactoryODM.getWrapper("JDBC", "/SimpleIssue/operation");
8      // set parameters for Database Access
9      wrapper.setJDBCDriverClassName("org.h2.Driver");
10     wrapper.setJDBCURL("jdbc:h2://Users/caiqueudutrasantos/Downloads//jar-file-jdbc//dbda");
11     wrapper.setJDBCUser("res");
12     wrapper.setPassword("res");
13
14
15     wrapper.setJDBCDriverClassName("org.postgresql.Driver");
16     wrapper.setJDBCURL("jdbc:postgresql://localhost:5432/odmdb");
17     wrapper.setJDBCUser("odmusr");
18     wrapper.setPassword("odmpwd");
19
20     wrapper.initialize();
21     // create your input and set values (came from XOM - should be in classpath)
22     ObjetoComposto composto = new ObjetoComposto();
23     composto.setCampoNumerico(10);
24     composto.setCampoString("umastring");
25     // set input into wrapper
26     //wrapper.putInputParameter("composto", composto);
```

## Executando regras carregadas de um diretório local – Parte 8.2

A terceira modalidade de execução carrega as regras e XOMs de um diretório local na máquina.



Da mesma forma que as duas anteriores, é necessário se obter os arquivos:

- j2ee\_connector-1\_5-fr.jar (extraído do Docker)
- jrules-res-8.11.1.0-execution.jar (extraído do Docker)
- jrules-engine-8.11.1.0.jar (extraído do Docker)
- ODMWrapper.jar (baixado do GIT deste projeto)

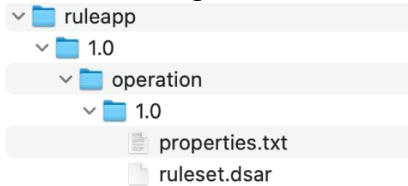
Não é necessário o driver de bancos de dados, e os arquivos:

- testexom.jar (extraído do Decision Center)
- XOMSimple.zip (extraído do ZIP do projeto, baixado do Decision Center)

Podem ser colocados em um diretório de livre escolha, que será apontado pelo programa na hora da carga.

```
1 package com.ibm.usingwrapper;
2
3 import com.ibm.odmjar.ObjetoComposto;
4 import com.ibm.odmwrapper.FactoryODM;
5 import com.ibm.odmwrapper.ODMWrapper;
6
7 public class UsingWrapperFile {
8     public static void main(String[] args) {
9         // Use factory to return Wrapper - Factory is a singleton
10        ODMWrapper wrapper = FactoryODM.getWrapper("file", "/regratestes/operacao");
11        wrapper.setDirectory("/Users/glaucoreis/Desktop/diretorio/");
12        wrapper.initialize();
13        ObjetoComposto composto = new ObjetoComposto();
14        composto.setCampoNumerico(10);
15        composto.setCampoString("umestring");
16        wrapper.putInputParameter("composto", composto);
17        try {
18            wrapper.execute();
19        } catch (Exception e) {
20            wrapper.getError().printStackTrace();
21        }
22        // If ok, you can get the number of rules fired
23        System.out.println("Rules fired " + wrapper.getNumberFiredRules());
24        if (wrapper.getNumberFiredRules() == 0)
25            System.out.println(wrapper.getError().getMessage());
26        System.out.println(composto.getCampoSaida());
27    }
28
29 }
```

A construção deste diretório pode ser feita abrindo-se o arquivo JAR de regras dentro do diretório de carga. Ele acabará ficando no formato:



O arquivo **properties.txt** pode ser manualmente preenchido com os valores:

```
ruleset.engine=de
```

O resultado para esta execução será:

```
INFORMAÇÕES: Build XUEngineDynamicLoaderBuilder from class loader
mai 15, 2023 6:13:10 PM com.ibm.rules.engine.load.XUEngineDynamicLoaderImpl createEngineDefinitionFromDSAR
INFORMAÇÕES: Create engine definition from DSAR
Rules fired 1
Menor que 10 e umastring
```

E outras opções podem ser configuradas neste arquivo de configuração:

```
ruleset.engine=de
#ruleset.debug.enabled=false
#ruleset.shareable=true
#ruleset.trace.enabled=false
#rulesetSEQUENTIAL.trace.enabled=false
#rulesetSEQUENTIAL.trace.tasks=
#ruleset.status=enabled
#rulesetxmlDocumentDriverPool.reserveTimeout=30000
#rulesetxmlDocumentDriverPool.maxSize=1
#rulesetoptimization.enabled=false
#rulesetbom.enabled=true
#rulesetmaxIdleTime-1
#rulesetmanagedxom.uris=\rule\set
#dataconnector.factory.class
#rulesetdecisionEngine.maxRunningTime
#rulesetengine.version
#monitoring.enabled=true
#rulesetmaxIdleTime=
#rulesetoptimization.enabled=true
```

## Apendice A - Construção do Workspace do Eclipse a partir de um Library no Decision Center

O objetivo deste documento é apresentar o passo a passo para reconstruir um projeto no Eclipse com o plugin do ODM, a partir de informações do Decision Center e de XOMs fornecidos externamente. Iniciaremos com o projeto disponibilizado no Decision Center :

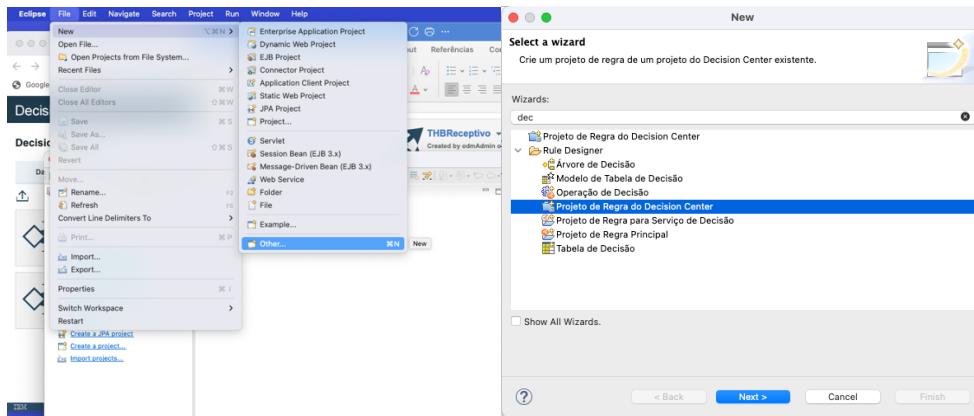
The screenshot shows the 'LIBRARY' section of the Decision Center interface. It lists three projects:

- Miniloan Rules**: Created by odmAdmin on Apr 26, 2023.
- Teste**: Last updated by odmAdmin on Apr 25, 2023.
- THBReceptivo**: Created by odmAdmin on Apr 17, 2023.

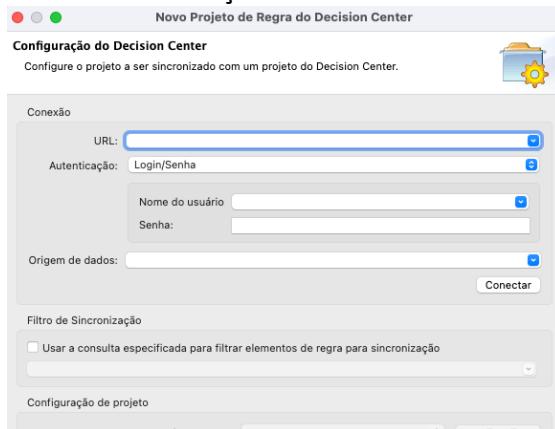
Abriremos um Eclipse, e apontaremos para um Workspace zerado :

The screenshot shows a clean Eclipse IDE workspace titled 'odmworkspace1 - Eclipse IDE'. The 'Project Explorer' view indicates there are no projects in the workspace. The 'Outline' view also shows no active editor. The 'Markers' view is empty.

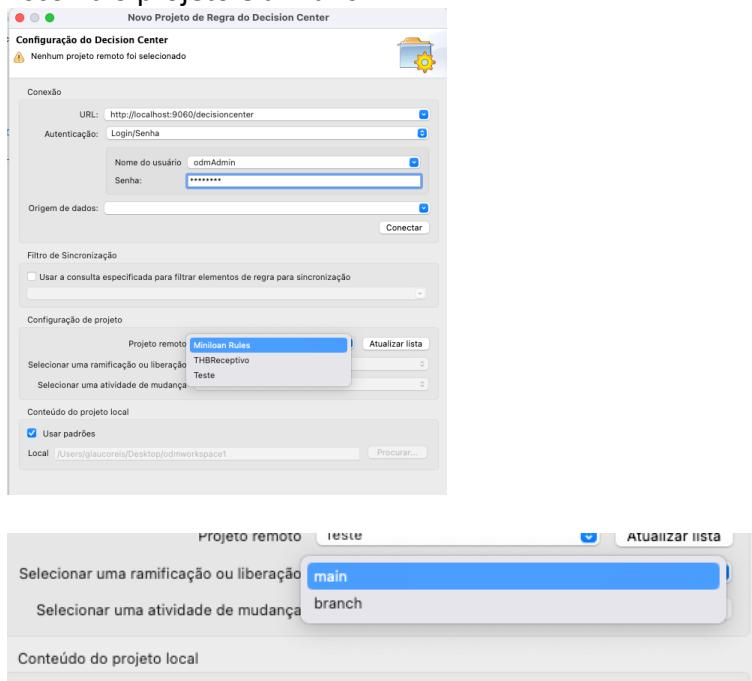
Crie um novo projeto a partir do Decision Center (new Project -> Other) :



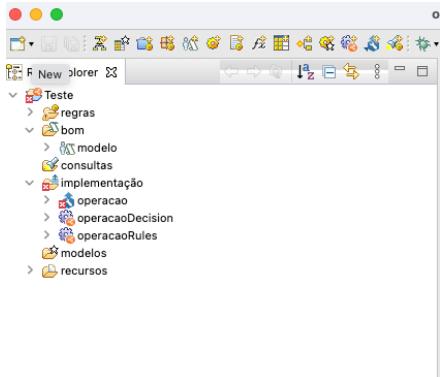
Insira as informações de conexão com o Servidor :



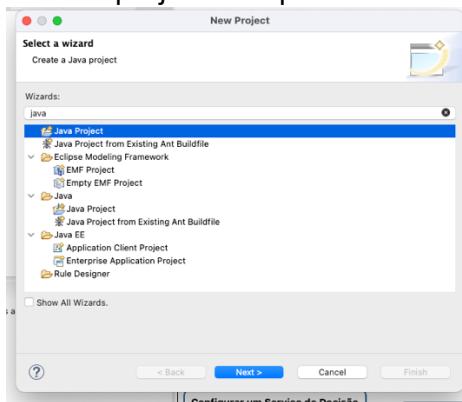
Escolha o projeto e a Branch :



Na perspectiva regra devem aparecer todos os artefatos que estavam no Decision Center. Vale lembrar que o fonte JAVA dos XOMs não é enviado para o Decision, precisamos reconstruí-lo :



Crie um projeto Java para receber o XOM :

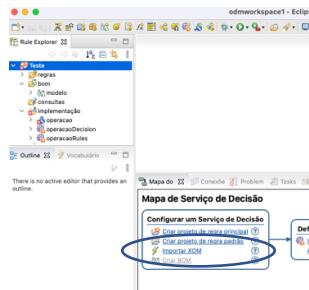


Crie o package e copie a classe java para o projeto, se certifique de que ela não contém erros. Faça o mesmo com outras classes que sejam necessárias para o projeto.

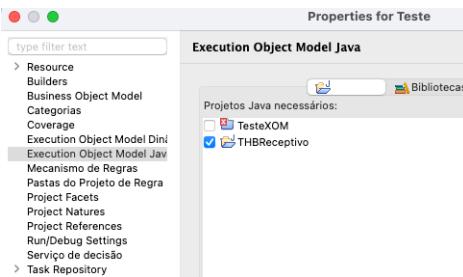
```

1 package com.ibm.odmjar;
2
3 public class ObjetoComposto {
4
5     private String campoString;
6     private double campoNumerico;
7     private String campoSaida;
8
9     public String getCampoSaida() {
10         return campoSaida;
11     }
12     public void setCampoSaida(String campoSaida) {
13         this.campoSaida = campoSaida;
14     }
15     public String getCampoString() {
16         return cam String com.ibm.odmjar.ObjetoComposto.getCampoString()
17     }
18     public void se
19         this.campoString = campoString;
20     }
21     public double getCampoNumerico() {
22         return campoNumerico;
23     }
24     public void setCampoNumerico(double campoNumerico) {
25         this.campoNumerico = campoNumerico;
26     }
27 }
28 }
```

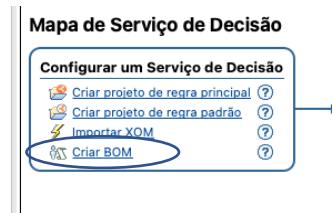
Clique sobre o projeto que importou do Decision Center. Se certifique de que está na perspectiva Regras. Clique em importar XOM :



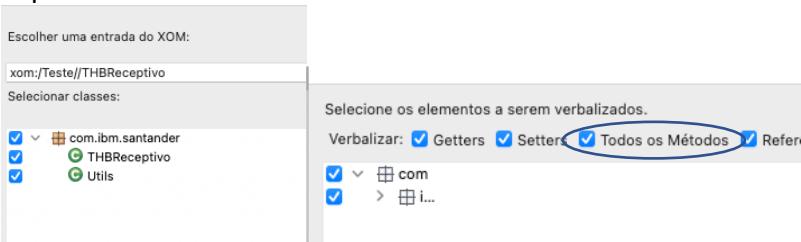
Conekte o projeto Java recém criado com o projeto importado a partir do Decision Center :



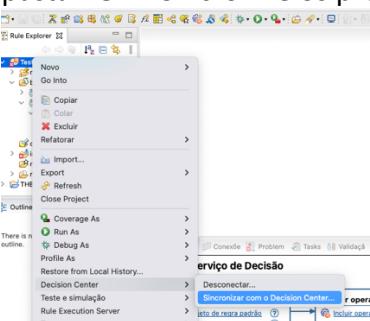
No mapa de Serviços de decisão, o item Criar BOM deve habilitar, permitindo a criação dos BOMs :



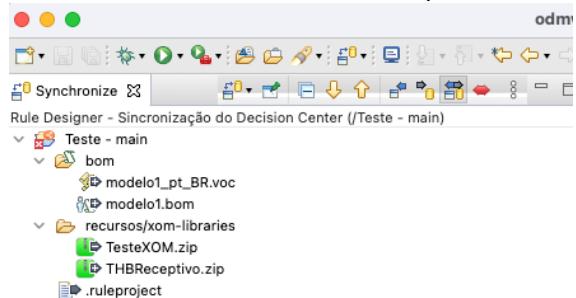
Clique em criar BOM e selecione as classes que deseja verbalizar. Se a classe necessitar expor outros métodos, que não sejam getters ou setters, marque a checkbox para exportar todos os métodos :



Agora o projeto contém os BOMs para as classes verbalizadas. Elas devem aparecer na pasta BOM. Sincronize os projetos :



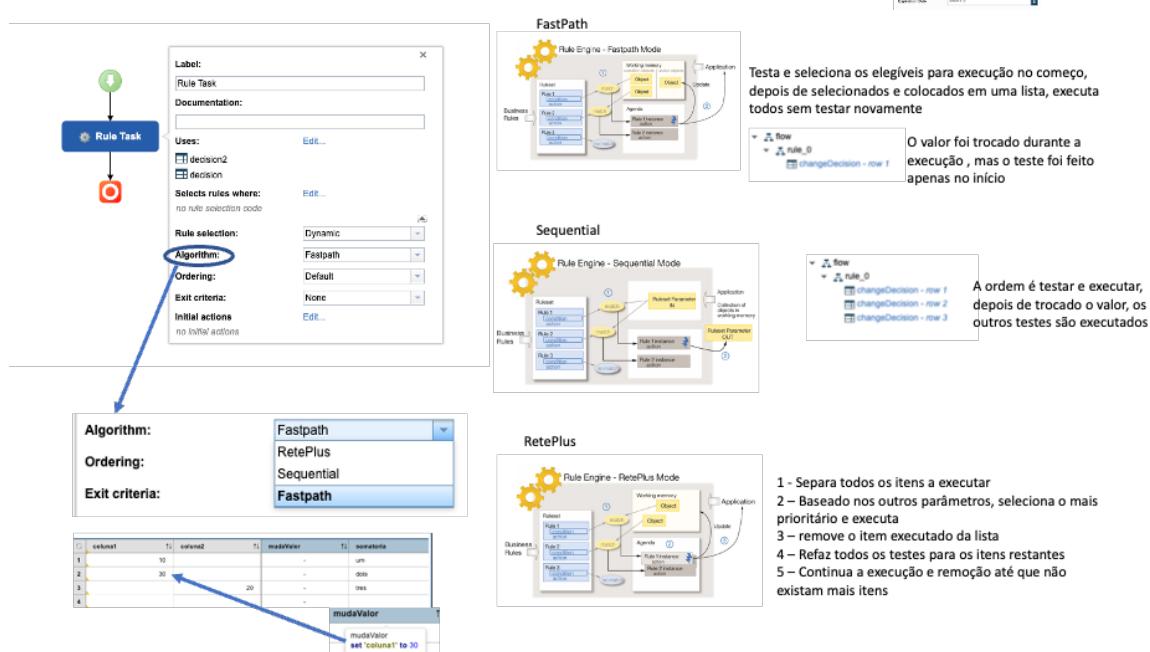
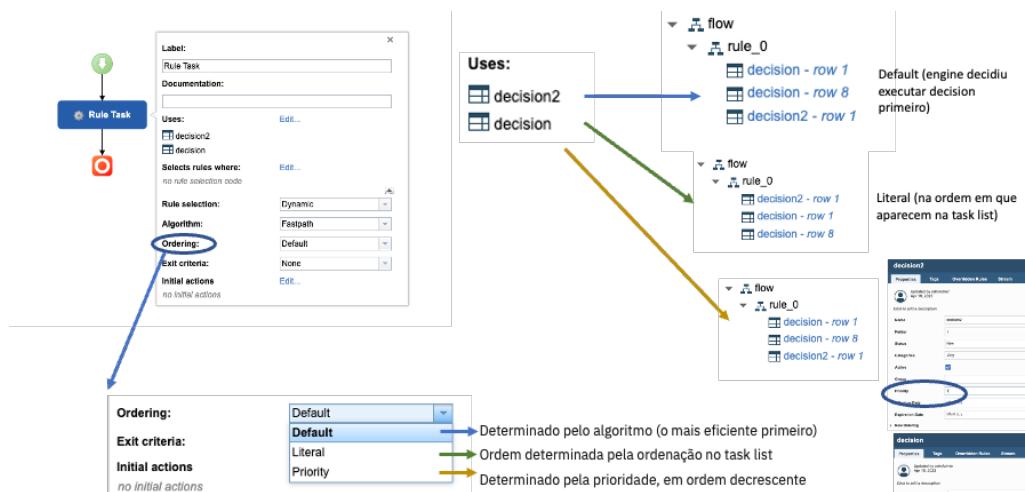
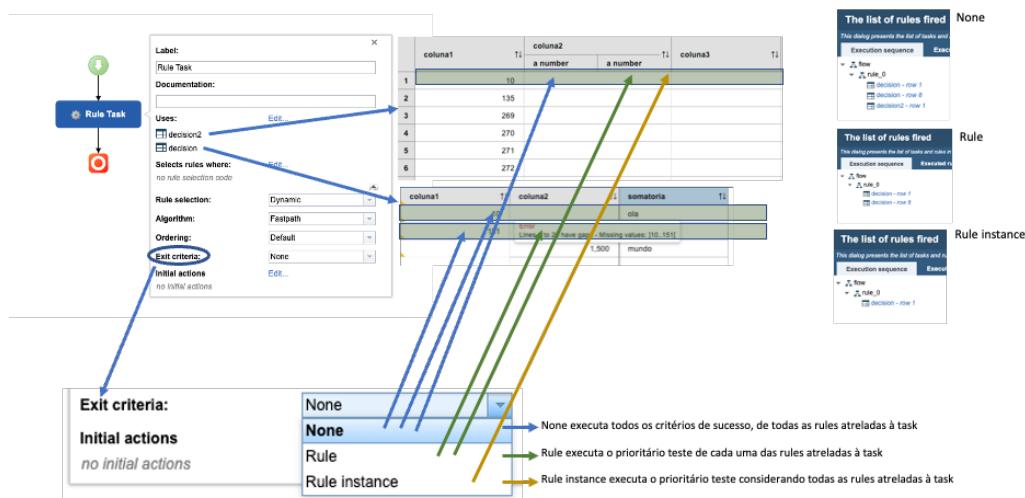
Se perguntar se deseja chavear para a perspectiva Team Syncronizing, clique em Yes. Você verá os itens que devem ser enviados e recebidos. Eles são representados com sinais de + e -, dentro de ícones pretos.

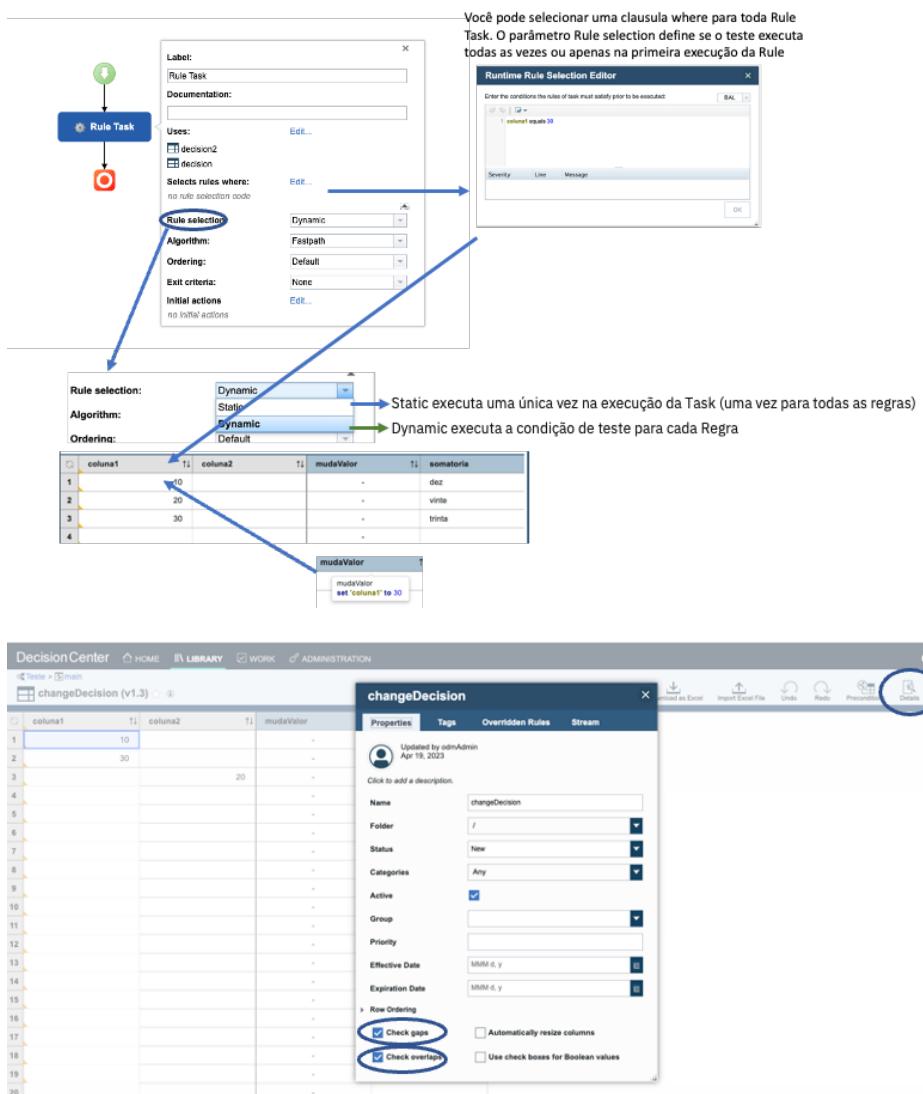


Clique em “Substituir e atualizar”, a partir do menu clicando com o botão direito do mouse sobre o projeto que veio do Decision Center. Agora, se você for até o Decision Center, verá o projeto atualizado, com os XOMs e BOMs configurados (na aba Models) :

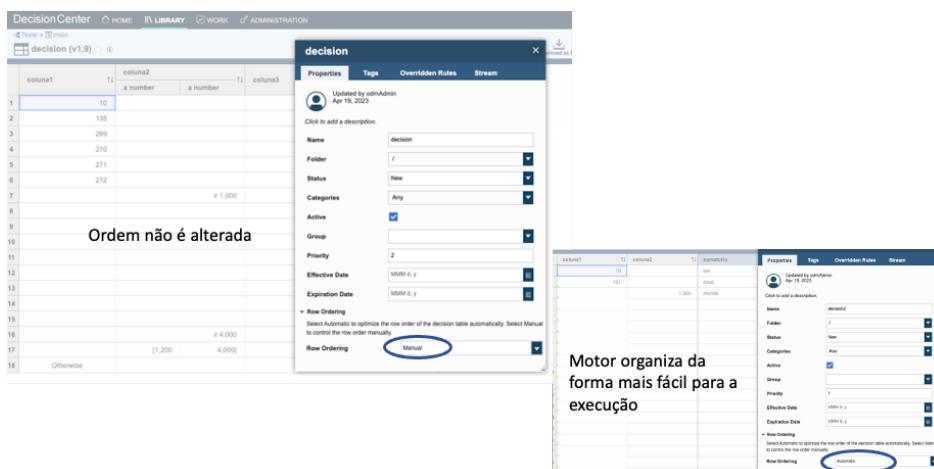


## Apendice B – Opções de configuração para Flows





Pode-se ignorar a verificação de gaps e de overlaps (importante se for um requisito da decision table)



Pode-se ignorar a verificação de gaps e de overlaps (importante se for um requisito da decision table)