**Концепция использования Activiti в CM5**

Целью интеграции Activiti в платформу CM4.5 является оценка возможности использования данного движка для решения задач маршрутизации и обработки доменных объектов в платформе CM5 и последующих, оценки удобства написания и понятности процессов используя нотацию BPMN2, оценки быстродействия. Данная цель будет достигаться путем внедрения движка Activiti в платформу CM4.5, написания обертки движка для удобства работы с ним и создания необходимой инфраструктуры для удобного взаимодействия движка Activiti и остальных сервисов платформы, написания тестовых процессов, которые должны охватывать максимальное количество механизмов, которые потребуются при реализации реальных бизнес процессов в последующих версиях платформы.

Перечень задач, которые будут решаться с помощью Activiti:

1. Маршрутизация и обработка доменных объектов, смена статуса, изменение атрибутов доменных объектов, создание и удаление доменных объектов.
2. Формирование задач в папке «Задачи» пользователей с возможностью сортировки и фильтрации.
3. Формирование элементов управления на карточках доменных объектов для выполнения действий доступных конкретному пользователю на данном этапе работы процесса.
4. Отправка уведомлений о поступление новой задачи по электронной почте.

Способ использования Activiti:

1. Движок Activity встраивается в ядро CM5
2. Данные Activiti хранятся в той же базе данных, что и данные CM5
3. Activiti использует информацию о пользователях и группах из CM5
4. Реализовываются spring бины для наиболее часто выполняемых операций, для упрощения процесса создания workflow
5. Доступные действия для каждого UserTask формируются с помощью заранее оговоренного имени переменной, в которой через запятую хранятся возможные действия. Имя переменной должно содержать имя активности, для исключения влияния друг на друга нескольких одновременно активных UserTask. Например NEGOTIATION\_ACTIONS или EXECUTION\_ACTIONS, где строка до нижнего подчеркивания имя активности, строка \_ ACTIONS константа, по которой находится нужная переменная. Данная переменная устанавливается в процессе в автоматической активности, предшествующей ручной активности. В результате действия пользователя устанавливается в true переменная, имя которой так же заранее оговорено, И содержит имя активности, например NEGOTIATION\_RESULT или EXECUTION\_RESULT. Значение этой переменной анализируется в gateway расположенном за ручной активностью.
6. Идентификатор связанного с процессом документа хранится в специально именованной переменной, например – DOCUMENT\_ID.
7. Для доступа из процесса к документу, привязанному к процессу, необходимо создать специальный спринг-бин, доступ к которому можно осуществлять из описания процесса с помощью выражений (expression). Например доступ к полю статус – «#{documentServise.getDocument(id).getAttribute(“status”)}», доступ к полю связанного документа - «#{documentServise.getDocument(DOCUMENT\_ID).getAttribute(“author.full\_name”)}».
8. Для облегчения работы с документом в начале процесса необходимо определить переменную «DOCUMENT» и инициализировать ее с помощью следующего выражения - #{documentServise.getDocument(DOCUMENT\_ID)}, далее в процессе доступ к документу можно осуществлять следующим выражением - DOCUMENT.getAttribute(“status”) или DOCUMENT.getAttribute(“author.full\_name”) в случае связанных полей.
9. Модификации документа производить в скриптовых активностях с помощью выражений DOCUMENT.setAttribute(“description”, “any\_string”); DOCUMENT.save();
10. К процессу могут быть привязаны не только документы, но и вспомогательные карточки, например карточка резолюции, карточка рассматривающего, карточка согласующего. Для процесса нет разницы какая карточка привязана, работа производится с доменным объектом, идентификатор которого указан в переменной DOCUMENT\_ID. Из GUI, представления задачи поднимается карточка привязанного к процессу документа, и на карточке формируются доступные действия, полученные из переменной ACTION\_NAME\_ACTIONS (необходимо продумать где хранить иконки и русские названия соответствующих элементов управления, я думаю что строка из ACTION\_NAME\_ACTIONS это имя соответствующих ресурсов в GUI).

## Обще используемые классы, реализующие общие операции в автоматических активностях.

Перечень классов, и их описание.

|  |  |  |
| --- | --- | --- |
| **Имя класса** | **Описание** | **Поля** |
| *InitProcess* | класс, необходимый для инициализации общих переменных во всех процессах. Инициализирует переменную DOCUMENT | **variables** – Поле класса, инициализируемого списком переменных, устанавливаемых в процессе. Формат поля: var\_name1=var\_value1; var\_name2=var\_value2;…; var\_nameN=var\_valueN; |
| *SetStatus* | Класс, устанавливающий статус документа | **status** – Новый статус документа |
|  |  |  |
|  |  |  |

## Этапы интеграции движка Activiti в платформу CM4.5

Этап 1. Глубокое изучения движка

На данном этапе необходимо произвести глубокое изучение движка, интеграцию его в платформу CM4.5, написание внутреннего API и тестовых процессов. На данном этапе необходимо провести функциональные и нагрузочные испытания, выявить слабые стороны и пути их обхода или исправления.

При встраивание движка необходимо максимально облегчить работу с ним для конечных пользователей – разработчиков процессов. Для этого всю рутинную работу желательно вынести за пределы шаблона процесса, а так же при возможности обеспечить возможность взаимодействия с остальными сервисами платформы на понятном, лаконичном языке формул и выражений, работа с которыми не требует квалификации программиста.

Перечень наиболее предпочтительных функций окружения движка Activity в платформе:

1. При запуске процесса инициализировать переменную DOMAIN\_OBJECT по идентификатору, который передается в функцию запуска процесса. Объект, на который ссылается переменная DOMAIN\_OBJECT позволяет получать атрибуты доменного объекта и менять их.
2. В выражениях желательно использовать упрощенный синтаксис обращения к доменному объекту. Например, чтобы получить наименование вместо DOMAIN\_OBJECT.getAttribute(“name”), желательно использовать выражение DOMAIN\_OBJECT.name. Для реализации данного функционала изучить возможности интеграции в Activiti спринг библиотеки SpEL.

### Этап 2. Доводка движка до стадии полноценного инструмента.

На данном этапе необходимо исправить все ошибки интеграции, доработать при необходимости API и обертку жвижка внутри платформы, доработать обертку для более удобного и понятного использования, при возникновение проблем с производительностью найти пути по решению и исправить данную проблему. Задокументировать итоговый вариант.

## Вопросы, по использованию Activiti

|  |  |  |
| --- | --- | --- |
| **Вопрос** | **Предположительный ответ** | **Окончательный ответ** |
| Есть ли необходимость создания доменного объекта задача | Да, необходимость есть. Наличие такого объекта позволит стандартными способами вести историю, настраивать коллекции, назначать права, создавать карточки в интерфейсе. | Да |
| Есть ли необходимость создания доменного объекта задача для каждого типа задачи | Нет. Набор полей в карточке задачи фиксирован и фиксация в документации имен полей задачи движка и соответствующих им имен полей в доменном объекте позволит формировать практические любые интерфейсы в GUI. | Нет. |
| Каким образом на интерфейсе карточки задачи или карточки документа отображается список доступных для данной задачи (задачи по данному документу) действий и каким образом в процесс передается результат этого действия. | Вариант 1. В объекта задача движка фиксируется имя поля, в котором через запятую перечисляются возможные действия для данной задачи. В интерфейсе данное поле зачитывается и формируются элементы управления в виде кнопок или выпадающего списка с вариантами действий. При выборе одного из действий имя этого действия сохраняется в другое предопределенное поле задачи движка Activiti. Далее это поле анализируется в процессе.  Вариант 2. На уровне конфигурации GUI к карточке задачи привязываются действия (Action). Состав этих действий зависит от значения определенных полей доменного объекта задача и его статуса. При выборе одного из действий имя этого действия сохраняется в предопределенное поле задачи движка Activiti. Далее это поле анализируется в процессе.  Вариант 3. К процессу так или иначе привязан дленный объект (например объект рассмотрение). При формирование списка задач основным идентификатором является идентификатор доменного объекта. Действия привязываются на уровне конфигурации к данному доменному объекту. При выполнение тех или иных действий меняется поля данного доменного объекта, и затем эти поля анализируются в процессе. При этом в настройке действий их доступность определяется статусом доменного объекта. | Выбираем вариант 1 с небольшими дополнениями. Список действий это не произвольный набор строк, а перечисление через запятую имена действий в конфигурации Actions. Фиксируется 3 поля:  ATTACHMENT\_ID – идентификатор прикрепленного к процессу основного доменного обьекта  ACTIONS – список имен действий  TASK\_RESULT – выбранное пользователем действие |
| Каким образом на карточке задачи формируется список присоединенных к процессу документов или как карточка документа определяет список задач по этому документу | Вариант 1. В объекте задача движка Activiti формируется поле с предопределенным названием, содержащее идентификатор основного документа привязанного к процессу. Это может быть идентификатор входящего документа или идентификатор карточки рассмотрения. | Да. Поле называется ATTACHMENT\_ID. |

### Описание интеграции Activiti в ядро.

Активити интегрировано в ядро как набор спринговых бинов. Необходимые таблицы в базе данных создаются автоматически при страте сервера приложений.

Для взаимодействия с activity создан EJB - ru.intertrust.cm.core.business.api.ProcessService. Ниже следует описание методов интерфейса:

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| void | [**completeTask**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#completeTask(ru.intertrust.cm.core.business.api.dto.Id, java.util.List, java.lang.String))([**Id**](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/Id.html?is-external=true) taskId, [**List**](http://download.oracle.com/javase/6/docs/api/java/util/List.html?is-external=true)<**[ProcessVariable](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/ProcessVariable.html?is-external=true" \o "class or interface in ru.intertrust.cm.core.business.api.dto)**> variables, [**String**](http://download.oracle.com/javase/6/docs/api/java/lang/String.html?is-external=true) action)  Завершить задачу |
| [**String**](http://download.oracle.com/javase/6/docs/api/java/lang/String.html?is-external=true) | [**deployProcess**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#deployProcess(byte[], java.lang.String))(byte[] processDefinition, [**String**](http://download.oracle.com/javase/6/docs/api/java/lang/String.html?is-external=true) processName)  Установка шаблона процесса |
| [**List**](http://download.oracle.com/javase/6/docs/api/java/util/List.html?is-external=true)<**[DeployedProcess](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/DeployedProcess.html?is-external=true" \o "class or interface in ru.intertrust.cm.core.business.api.dto)**> | [**getDeployedProcesses**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#getDeployedProcesses())()  Получение информации об установленных процессах |
| [**List**](http://download.oracle.com/javase/6/docs/api/java/util/List.html?is-external=true)<**[DomainObject](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/DomainObject.html?is-external=true" \o "class or interface in ru.intertrust.cm.core.business.api.dto)**> | [**getUserDomainObjectTasks**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#getUserDomainObjectTasks(ru.intertrust.cm.core.business.api.dto.Id))(**[Id](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/Id.html?is-external=true" \o "class or interface in ru.intertrust.cm.core.business.api.dto)** attachedObjectId)  Получение доступных задач для юзера и определенного доменного обьекта |
| [**List**](http://download.oracle.com/javase/6/docs/api/java/util/List.html?is-external=true)<**[DomainObject](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/DomainObject.html?is-external=true" \o "class or interface in ru.intertrust.cm.core.business.api.dto)**> | [**getUserTasks**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#getUserTasks())()  Получение доступных задач для юзера |
| [**String**](http://download.oracle.com/javase/6/docs/api/java/lang/String.html?is-external=true) | [**startProcess**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#startProcess(java.lang.String, ru.intertrust.cm.core.business.api.dto.Id, java.util.List))([**String**](http://download.oracle.com/javase/6/docs/api/java/lang/String.html?is-external=true) processName, [**Id**](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/Id.html?is-external=true) attachedObjectId, [**List**](http://download.oracle.com/javase/6/docs/api/java/util/List.html?is-external=true)<**[ProcessVariable](https://sup.inttrust.ru:8446/prjdocs/cm5/core/model/apidocs/ru/intertrust/cm/core/business/api/dto/ProcessVariable.html?is-external=true" \o "class or interface in ru.intertrust.cm.core.business.api.dto)**> variables)  Запуск процесса |
| void | [**terminateProcess**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#terminateProcess(java.lang.String))([**String**](http://download.oracle.com/javase/6/docs/api/java/lang/String.html?is-external=true) processId)  Остановка процесса |
| void | [**undeployProcess**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#undeployProcess(java.lang.String, boolean))([**String**](http://download.oracle.com/javase/6/docs/api/java/lang/String.html?is-external=true) processDefinitionId, boolean cascade)  Удаление шаблона процесса(каскадно или нет). |

Установка процесса производится с помощью вызова метода [**deployProcess**](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#deployProcess(byte[], java.lang.String))**.** Методу передается массив байт описания процесса и его имя. Имя может быть произвольным, но должно заканчивается символами .bpm.

Старт процесса производится методом [startProcess](file:///D:\MyDocuments\MyProject\CM\CM5SRC\CM5\Core\service-api\target\site\apidocs\ru\intertrust\cm\core\business\api\ProcessService.html#startProcess(java.lang.String, ru.intertrust.cm.core.business.api.dto.Id, java.util.List)). Методу передается имя процесса, которое определено в шаблоне процесса (не путать с processName которое передается при инсталляции). Так же процессу передается идентификатор основного приаттаченного доменного объекта (опционально) и набор произвольных атрибутов (так же опционально). При передаче идентификатора основного доменного объекта в процессом создается переменная MAIN\_ATTACHMENT\_ID, и в нее записывается идентификатор процесса. Так же при передаче идентификатора основного приаттаченного объекта, в процессе создается переменная MAIN\_ATTACHMENT инициализированная объектом типа ru.intertrust.cm.core.tools.DomainObjectAccessor. Эта переменная служит для более удобного доступа к атрибутам доменного объекта, возможности получить атрибуты, установить атрибуты а так же выполнить сохранение. Например, получить атрибут можно в активности типа скрипт следующей строкой MAIN\_ATTACHMENT.get(“Name”). Описание всех методов класса ru.intertrust.cm.core.tools.DomainObjectAccessor будет приведено ниже. Так же в каждом экземпляре процесса создается переменная SESSION типа ru.intertrust.cm.core.tools.Session. Данный объект позволяет более удобно работать с сервисами из скриптовых активностей. Например для создания доменного объекта определенного типа необходимо вызвать метод SESSION.create(“type\_name”).

При формирование задачи пользователю в activity, автоматически создается доменный объект типа Person\_Task. Дальнейшая работа с задачами производится с помощью существующих сервисов. Адресат задачи сохраняется в специальных доменных объектах типа Assignee\_Person и Assignee\_Group в зависимости пользователь или группа указаны в поле assignee задачи.

При формирование задачи в шаблоне процесса можно определить действия, которые в дальнейшем могут отобразиться. Действия определяются в поле формы ACTIONS, привязанной к задаче. При формирование доменного объекта возможные действия копируются в поле Actions в формате name1=label1;… nameN=labelN.

При завершение задачи в метод completeTask надо передать идентификатор доменного объекта задача и имя действия из поля Actions. Далее идентификатор выполненного действия копируется в переменную, указанную в атрибуте variable поля формы ACTIONS. Пример описания задачи с действиями.

<userTask id="usertask3" name="Проверка возможности маршрутизации с помошью акшенов" activiti:assignee="#{personManagement.getPersonId(&quot;admin&quot;).toStringRepresentation()}">

<documentation>Получить список действий и выполнить одно из них.</documentation>

<extensionElements>

<activiti:formProperty id="ACTIONS" name="Действие" type="enum" variable="RESULT">

<activiti:value id="YES" name="Да"></activiti:value>

<activiti:value id="NO" name="Нет"></activiti:value>

</activiti:formProperty>

</extensionElements>

</userTask>

В данном примере доступные действия YES и NO и после завершения задачи выполненное действие (значение параметра action метода completeTask) скопируется в переменную процесса RESULT, и в дальнейшем эта переменная может использоваться в написание логики процесса, например в условии потока. Например:

<sequenceFlow id=*"flow-now"* sourceRef=*"exclusivegateway1"* targetRef=*"scripttask2"*>

<conditionExpression xsi:type=*"tFormalExpression"*><![CDATA[${RESULT.equals("NO")}]]></conditionExpression>

</sequenceFlow>

Особенность текущей реализации activity такова, что не работает механизм автоматического создания переменных процесса, при их объявлении в javaScript. Для избежания данных ошибок необходимо при формирование ScriptTask устанавливать атрибут autoStoreVariables в false. Пример SctiptTask:

<scriptTask id="scripttask2" name="Установка данных по действию НЕТ" scriptFormat="javascript" activiti:autoStoreVariables="false">

<script>var test =MAIN\_ATTACHMENT.get("test\_text");

test += " Результат маршрутизации НЕТ.";

MAIN\_ATTACHMENT.set("test\_text", test);

MAIN\_ATTACHMENT.save();

</script>

</scriptTask>