RUNA WFE. Руководство разработчика.
Версия 2.1

Оглавление

RUNA WFE - открытая, масштабируемая, ориентированной на конечного пользователя система управления бизнес-процессами для средних и крупных предприятий, полностью разработанная на Java. Система является open source решением, основано на популярном workflow ядре JBOSS-JBPM.

Характеристики системы:
- возможность интеграции существующих разнородных приложений предприятия
- удобный веб интерфейс пользователя
- боты для выполнения автоматических заданий
- гибкая система определения исполнителей на основе ролей
- простая интеграция с существующими реляционными базами данных
- система безопасности позволяющая интеграцию с LDAP/MS Active Directory
- локализация на английский, французский, немецкий и русский языки
- поддержка операционных систем Windows, Linux, Solaris, FreeBSD

Общее описание

Для WF–системы была выбрана следующая общая архитектура (в целом соответствует архитектуре, предлагаемой коалицией WfMC):

Компоненты системы:

- Ядро системы. (На основе JBOSS JBPM)
  - Содержит набор определений бизнес-процессов
  - Содержит набор выполняющихся экземпляров бизнес–процессов
- Компонент, «назначающий» исполнителей для Activity
- Клиент
  - Task list. (Набор графических форм, содержит очереди поступивших работ, сортировки и фильтры)
  - Проигрыватель форм. (Визуализирует формы, разработанные в редакторе процессов)
  - Административный интерфейс
    - Показывает состояния процессов, позволяет фильтровать и останавливать процессы
    - Позволяет загружать-выгружать процессы
    - Позволяет заводить-удалять пользователей
    - Позволяет задавать различные права
  - Редактор назначения заместителей.
- Графический редактор процессов.
- Конструктор графических форм.
- Бот-станции, содержащие ботов (Боты - приложения специального вида, которые также как и обычные пользователи могут выполнять задания)
- Подсистема управления правами доступа (авторизация и аутентификация)

В проекте использованы следующие технологии:

- EJB 2.0 (stateless session beans) – интерфейс взаимодействия с серверной частью и декларативная транзакционность
- JSP 2.0, Servlet 2.3, Struts 1.2 – построение тонкого пользовательского интерфейса
- ORM (Hibernate 2.1) – организация доступа к данным
- Eclipse RCP – платформа, на которой разработан графический редактор процессов
- JAAS - аутентификация пользователей

В проекте использовано ORM Hibernate, поэтому система легко перенастраивается на различные СУБД.

Платформа программирования и используемые программные средства.

В качестве платформы программирования используется J2EE.

Используемые программные средства:

1. **Сервер приложений** - JBOSS (http://www.jboss.org).
2. **Среда разработки -** Eclipse от IBM (http://www.eclipse.org).
3. **Средство генерации кода и дескрипторов** – xdoclet (http://xdoclet.sourceforge.net).
4. **Система контроля версий** – subversion (http://subversion.tigris.org/).
5. **Сборщик приложений** – ant (http://ant.apache.org).
6. **Сервер баз данных** – поддерживаются сервера БД:
   - ✓ MS SQL Server (http://www.microsoft.com/sql/evaluation/default.mspx)
   - ✓ MySQL (http://www.mysql.com)
   - ✓ HSSQLDB (http://hsqldb.org)
   - ✓ Oracle (http://www.oracle.com)

Подробное описание текущей программной архитектуры системы

Структура папок проекта.

- wfe - основной подпроект системы RUNA WFE
- bots – боты и бот-станции
- customization – дополнительные элементы, которые могут разрабатывать пользователи системы
  - оргфункции
  - VarTags
  - DecisionHandlers
  - Валидаторы
  - форматтеры переменных
  - ActionHandlers
- gpd – графический редактор бизнес-процессов
- rtn – клиент-оповещатель о поступивших заданиях (для Windows)
- web – web-интерфейс системы

Система состоит из нескольких подсистем.
Основные подсистемы:

- jbpm - ядро workflow системы (Заимствовано у проекта JBOSS JBPM, в ядро внесено большое количество исправлений)
- af - подсистема авторизации и аутентификации. Не зависит он других подсистем,

допускает независимое от Runa WFE использование.
- wf – собственно workflow подсистема. Зависит от af и jbpm.

Описание JBOSS JBPM 2.0 beta3 core с патчем для Runa WFE

Ядро системы JBOSS JBPM размещено в jar-файле: jbpm.core.jar
Ядро зависит от библиотек, находящихся в следующих папках дистрибутива JBOSS JBPM:

- lib/commons/*.jar
- lib/hibernate/*.jar

Ядро системы JBOSS JBPM состоит из следующих пакетов:

- org.jbpm
- org.jbpm.ant
- org.jbpm.delegation
- org.jbpm.impl
- org.jbpm.model
- org.jbpm.par
- org.jbpm.persistence
- org.jbpm.sheduler

Пакет org.jbpm

Основной пакет ядра, содержит интерфейсы верхнего уровня для работы с ядром.
Этот интерфейс состоит их трех Java-интерфейсов:

- DefinitionService – сервис для работы с определениями бизнес-процессов в ядре системы
- ExecutionService – сервис для работы с выполняющимися экземплярами бизнес-процессов в ядре системы.

Для работы с этими интерфейсами предусмотрен класс-фабрика ServiceFactory.

Более подробное описание интерфейсов и классов пакета:
Интерфейс DefinitionService
Методы:

- deployProcessArchive(JarInputStream processArchiveStream) – загружает определение бизнес-процесса в ядро системы
- undeployProcessArchive(String name) – удаляет определение и все экземпляры бизнес-процесса из ядра системы

Интерфейс ExecutionService (наследует интерфейс ExecutionReadService)
Основные Методы:

- startProcessInstance(long definitionId, Map variables) – создает экземпляр бизнес-процесса, запускает его на выполнение и передает ему начальные значения параметров
- endOfState(long tokenId, Map variables) – сообщает экземпляру бизнес-процесса, что задание выполнено и передает экземпляру бизнес-процесса новые значения параметров.

- reassign(long tokenId, String newActorId) - реинициализирует роль-дорожку
- cancelProcessInstance(long processInstanceId) – отменяет выполнение экземпляра бизнес-процесса
- cancelToken(long flowId) – отменяет выполнение потока

Замечание. Реализация ExecutionService активно использует классы пакета delegation, в частности ExecutionContextImpl (реализует интерфейсы пакета delegation: ExecutionContext, AssignmentContext, ForkContext, JoinContext, ProcessInvocationContext)
Интерфейс ExecutionReadService (его наследует интерфейс ExecutionService)
Основные методы:

- getTaskList(String targetActorId)
- getToken(long tokenId)
- getProcessInstance(long processInstanceId)
- getVariables(long tokenId)
- getDefinition(long definitionId)
- getLatestDefinitions()
- getFile(long processDefinitionId, String fileName)
- getAllDefinitions()
- findActiveTokensInState(long stateId)
- findByQuery(JbpmQuery jbpmQuery)
- findByCriteria(Class clazz, Criterion[] criterions, Order[] orders)

Замечание. Данные методы реализуются в системе при помощи классов пакета org.jbpm.persistence.

Менее важные классы и интерфейсы пакета:
Класс JbpmConfiguration – содержит конфигурационные настройки ядра системы.
Класс JbpmQuery – позволяет посылать запросы ядру
Интерфейс SchedulerService – интерфейс для составления расписаний выполнения и отмен заданий

Пакет org.jbpm.impl

В основном пакет содержит классы, реализующие интерфейсы пакета org.jbpm.
Основные классы пакета:

- DefinitionServiceImpl
- ExecutionServiceImpl
- SchedulerServiceImpl

Пакет org.jbpm.model

Пакет содержит другие пакеты, которые содержат интерфейсы описывающие объекты предметной области: определения и экземпляры бизнес-процессов, а также объекты типа Job.

Пакет org.jbpm.model.definition

Содержит интерфейсы описывающие определение бизнес-процесса.
Основные интерфейсы пакета:

- Definition – определение бизнес-процесса
- State – узел-действие
- Decision – исключающий выбор – простое соединение
- Fork – параллельное расщепление
- Join – синхронизация
- ProcessState – узел-подпроцесс
- StartState – точка старта процесса
- EndState – точка завершения процесса
- Node - узел – базовый класс
- Swimlane – роль-дорожка
- Transition – переход
- Variable - переменная

Краткое резюме по набору интерфейсов пакета

Набор классов соответствует стандартному способу реализации направленного графа в виде набора объектов. Для этого применяются объекты двух видов:

- Узлы
- Связи

Для класса узел разработан набор классов-наследников (в соответствии с различными видами узлов):

- Начальный узел графа
- Конечный узел графа
- Начало ветвления процесса (на две параллельные ветви)
- Конец ветвления процесса («встреча» параллельных ветвей)
- Узел – подпроцесс
- Развилка - слияние процесса
- Обычный узел – узел, связанный с ответом пользователя или другого процесса

Основные элементы, из которых состоит Definition:

- Имя (унаследовано из Element)
- Начальный узел
- Конечный узел
- Множество узлов
- Множество ролей-дорожек
- Множество переменных
- Номер версии определения

Замечание: здесь в явном виде не присутствуют связи. Они определены отдельно.

Основные элементы, из которых состоит узел:

- Имя (унаследовано из Element)
- Множество исходящих переходов

Основные элементы, из которых состоит переход:

- Имя (унаследовано из Element)
- Узел «из»
- Узел «в»

Замечание: таким образом, при помощи данной конструкции - через множество узлов и множество пар узлов (связей) - можно задать граф любой структуры. Однако, дополнительно, каждый узел содержит множество исходящих переходов – это является

избыточным и теоретически может привести к нарушению целостности данных (если в множестве исходящих переходов будут находиться переходы, не являющиеся для данного узла исходящими). По-видимому данное архитектурное решение было «унаследовано» из спецификаций коалиции WfMC,

Пакет org.jbpm.model.definition.impl

Содержит классы-реализации интерфейсов пакета org.jbpm.model.definition

Пакет org.jbpm.model.execution

Содержит интерфейсы, описывающие экземпляры бизнес-процесса.
Интерфейсы пакета:

- ProcessInstance – экземпляр бизнес-процесса
- Token – поток управления

Пакет org.jbpm.model.execution.impl

Содержит классы, реализующие интерфейсы пакта org.jbpm.model.execution, а также класс VariableInstanceImpl, который предназначен для работы с экземпляром переменной бизнес-процесса.

Краткое резюме по набору классов пакета

Для каждого экземпляра бизнес–процесса создается экземпляр класса ProcessInstanceImpl, реализующий интерфейс ProcessInstance.

Этот класс содержит ссылку на определение бизнес процесса (класс, реализующий интерфейс ProcessDefinition). Также бизнес – процесс содержит ссылку на текущий поток выполнения бизнес-процесса (интерфейс Flow), а также ссылку на поток управления «родительского» процесса.

Каждый поток содержит набор переменных (типа VariableInstanceImpl) и ссылку на узел графа определения бизнес–процесса, в котором в текущий момент времени находится «управление потоком».

Замечание. Моделирование бизнес-процессов классы пакета осуществляют совместно с классами пакета delegation, в частности, с ExecutionContextImpl (реализует интерфейсы пакета delegation: ExecutionContext, AssignmentContext, ForkContext, JoinContext, ProcessInvocationContext)

Реализация выполнения бизнес-процессов в JBOSS JBPM оказалась достаточно запутанной. Эту функциональность в основном реализуют следующие классы:

- org.jbpm.impl.ExecutionServiceImpl
  Параметры конструктора - actorId, persistenceSession, serviceFactory
  Содержит следующие основные методы
  - startProcessInstance(definitionName, variables)
  - endOfState(tokenId, variables)

- org.jbpm.model.execution.impl.ProcessInstanceImpl – соответствует экземпляру бизнес-процесса

Основные переменные
- definition – ссылка на определение бизнес-процесса
- root – ссылка на «корневой» поток управления данного бизнес-процесса

- org.jbpm.model.execution.impl.TokenImpl – соответствует потоку управления
  Основные переменные
  - actorId – идентификатор пользователя – исполнителя текущего узла-действия
  - processInstance – ссылка на экземпляр бизнес-процесса, к которому относится данный поток управления
  - state – узел связанного с экземпляром данного бизнес-процесса определения процесса, в котором в текущий момент времени находится управление потока
  - parent – «родительский» поток управления
  - children – множество «дочерних» потоков управления
  - variableInstances – множество связанных с потоком переменных бизнес-процесса

  Замечание. «Дочерние» потоки управления используются для реализации расщеплений и слияний потока управления. После того, как управление потока попадает в Fork, этот поток приостанавливается, у него создаются и запускаются «дочерние» потоки – по одному потоку на каждый исходящий переход. (Реализации delegation по умолчанию, однако, устроены таким образом, что переменные содержит только «корневой» поток управления бизнес-процесса.). После того, как «дочерний» поток управления приходит в соответствующий Join, он завершается. После прихода в Join последнего «дочернего» потока, «родительский» поток реактивируется.

- org.jbpm.impl.ExecutionContextImpl – контекст потока управления. Класс используется только локально для облегчения реализации различных связанных с бизнес-процессом событий.
  Параметры конструктора - actorId, executionService, token
  Основные переменные
  - actorId – идентификатор пользователя – исполнителя текущего узла-действия
  - definition – ссылка на определение бизнес-процесса
  - processInstance – ссылка на экземпляр бизнес-процесса
  - token – ссылка на рассматриваемый поток управления
  - node - узел, в котором находится управление (текущий узел-действие)
  - abstractService – ссылка на ExecutionServiceImpl

  Замечание. Данный класс является несколько избыточным, все его перечисленные выше переменные уже определены в других классах. Используется для упрощения реализации методов других классов.

Замечание. Реализации различных типов узлов бизнес-процесса предусматривают использование механизма delegation. В этих случаях в процессе выполнения бизнес-процесса будет выполнен код соответствующих классов, реализующих интерфейсы пакета org.jbpm.delegation. Эти классы можно отдельно разрабатывать для конкретных бизнес-процессов, размещать в специальной папке архива бизнес-процесса и подгружать в ядро во время deployment'а бизнес-процесса.

Пакет org.jbpm.par

Содержит классы для работы с файлом-архивом бизнес-процесса
Основные классы пакета:

- DefinitionParser – производит разборку файла-архива бизнес-процесса в класс, реализующий Definition

Пакет org.jbpm.persistence

Основным интерфейсом пакета является интерфейс PersistenceSession.

Основные методы интерфейса:
Методы, относящиеся к бизнес-процессам.

- findLatestDefinition(String name) – возвращает последнее размещенное в ядре определение бизнес-процесса
- findTokensByActor(String actorId) – возвращает все потоки управления, текущие задания которых предназначены данному Пользователю
- findByQuery(JbpmQuery jbpmQuery) – возвращает результат пользовательского запроса к ядру
- findActiveTokensInState(long[] stateIds) – возвращает все потоки управления, находящиеся в данном узле-действии
- findAllDefinitions(String name) – возвращает все версии определений бизнес-процесса с данным именем
- removeDefinition(long id) – удаляет определение бизнес-процесса с данным id

Методы, относящиеся к транзакциям и сессии.

- beginTransaction();
- commitTransaction();
- rollbackTransaction();
- close() – закрывает сессию.

Пакет org.jbpm.persistence.hibernate

Содержит классы, реализующие интерфейсы пакта org.jbpm.persistence при помощи Object Relational Mapping (ORM) средства hibernate. Основным классом пакета является класс HibernateSession, Реализующий интерфейс PersistenceSession.

Hibernate – средство для отображения Java объектов в реляционную базу данных.

Пакет org.jbpm.delegation

В пакет собраны интерфейсы, для которых можно переписать реализации в случае специализированной системы на основе jbpm. Это потребуется, если default'ные реализации этих элементов чем-либо не удовлетворяют пользователей специализированной workflow-системы. Основные интерфейсы пакета.

«Контекст»-интерфейсы:

- ExecutionContext – интерфейс для доступа к контексту выполнения
- ForkContext – контекст расщепления потоков управления
- JoinContext – контекст слияния потоков управления
- ProcessInvocationContext – контекст запуспа подпроцессов

«Handler»-интерфейсы:

- ActionHandler – интерфейс, дающий возможность определять собственные реализации классов Action
- AssignmentHandler – интерфейс, дающий возможность реализовывать собственные алгоритмы назначения роли-дорожки
- DecisionHandler – интерфейс, дающий возможность реализовывать собственные алгоритмы для определения поведения элемента Decision
- ForkHandler – интерфейс, дающий возможность реализовывать собственные алгоритмы для определения поведения элемента Fork
- JoinHandler – интерфейс, дающий возможность реализовывать собственные алгоритмы для определения поведения элемента Join.
- ProcessInvocationHandler – интерфейс, дающий возможность реализовывать собственные алгоритмы для запуска подпроцесса.

Остальные пакеты

Остальные классы и пакеты имеют скорее второстепенное значение.

Описание внесенных в ядро изменений

Еще не готово. Будет сделано позже.

RunaWFE - Workflow-окружение

Система RunaWFE состоит из нескольких подсистем.
Основные подсистемы:

- af - подсистема авторизации и аутентификации. Не зависит он других подсистем, допускает независимое от Runa WFE использование.
- wf – собственно workflow подсистема. Зависит от af.

Описание слоев архитектуры системы

Список слоев:

- delegate
- service
- logic
- dao
- hibernate

Подробное описание слоев

Слой Delegate

Delegate-интерфейсы и реализующие их DelegateImpl-классы – это интерфейсы реализующие pattern проектирования BusinessDelegate, упрощающие доступ к серверному API workflow системы. Клиентское приложение (или бот) взаимодействуют с workflow системой только через Delegate-классы. Ссылки на интерфейсы Delegate получаются посредством запроса фабрики DelegateFactory которая в зависимости от конфигурации

возвращает нужную реализацию. Классы Delegate являются частью клиентского API.

Список основных Delegate-интерфейсов:

- af
  - AuthenticationServiceDelegate
    Основные методы
    - authenticate(...) - производит аутентификацию клиента, и в случае успеха возвращает javax.security.auth.Subject, в противном случае генерирует AuthenticationExceptio. На данный момент реализовано несколько перегруженных методов authenticate() позволяющие атуентифицировать клинета по имени/паролю, NTLM, Kerberos.
    - getActor(Subject subject) возвращает класс Actor, соответствующий переданному Subject.
  - AuthorizationServiceDelegate
    Основные методы
    - isAllowed(Subject subject, Permission permission, Identifiable identifiable) – возвращает имеет субъект, соответствующий Subject, данное право (permission) на объект identifiable.
    - setPermissions(Subject subject, Executor performer, Permission[] permissions, Identifiable identifiable) – устанавливает заданный набор прав на объект identifiable субъекту, соответствующему Subject.
    - getPermissions(Subject subject, Executor performer, Identifiable identifiable) – возвращает набор прав (собственных и унаследованных), которые имеет субъект, соответствующий Subject, на объект Identifiable.
    - getOwnPermissions(Subject subject, Executor performer, Identifiable identifiable) – возвращает набор прав (собственных), которые имеет субъект, соответствующий Subject, на объект Identifiable.
  - ExecutorServiceDelegate
    Основные методы
    - create(Subject subject, Actor actor) – создать пользователя
    - create(Subject subject, Group group)) – создать группу пользователей
    - remove(Subject subject, Executor[] executors) – удалить пользователей/группы пользователей
    - update(Subject subject, Actor actor, Actor newActor) – обновить детали пользователя
    - update(Subject subject, Group group, Group newGroup) – обновить детали группы пользователей
    - setPassword(Subject subject, Actor actor, String password) – установить пароль для пользователя
    - addExecutorToGroups(Subject subject, Executor executor, Group[] groups) – добавить пользователя в группы
    - removeExecutorFromGroups(Subject subject, Executor executor, Group[] groups) – исключить пользователя из групп
    - getActor(Subject subject, String name) – вернуть пользователя по его имени
    - getGroup(Subject subject, String name) – вернуть группу по ее имени
  - SystemServiceDelegate

Основные методы
- login(Subject subject, ASystem system) – войти в систему
- logout(Subject subject, ASystem system) – выйти из системы

- wf
  - DefinitionServiceDelegate
    Основные методы
    - deployProcessDefinition(Subject subject, byte[] process) – загрузить бизнес-процесс в систему
    - undeployProcessDefinition(Subject subject, String definitionName) – выгрузить бизнес-процесс из системы
  - ExecutionServiceDelegate
    Основные методы
    - startProcessInstance(Subject subject, String definitionName) – запустить бизнес-процесс
    - cancelProcessInstance(Subject subject, long processInstanceId) – принудительно остановить бизнес-процесс
    - getTasks(Subject subject, BatchPresentation batchPresentation) – возвращает список заданий пользователю (с учетом фильтров, настроек и т.д.)
    - compliteTask(Subject subject, long taskId, Map variables) – сообщить системе о том, что задание (task) выполнено и передать экземпляру бизнес-процесса новые значения измененных в данном узле-действии переменных.
    - getVariableNames(Subject subject, long taskId) – возвращает список имен переменных по Id задания
    - getVariable (Subject subject, long taskId, String variableName) – возвращает переменную по ее имени и Id задания
    - getSwimlanes(Subject subject, long instanceId) – возвращает список ролей-Дорожек по id экземпляра бизнес-процесса

В настоящее время их реализуют следующие Delegate-классы:
- af
  - AuthenticationServiceDelegateLocalImpl
  - AuthenticationServiceDelegateRemoteImpl
  - AuthorizationServiceDelegateLocalImpl
  - AuthorizationServiceDelegateRemoteImpl
  - ExecutorServiceDelegateLocalImpl
  - ExecutorServiceDelegateRemoteImpl
  - SystemServiceDelegateLocalImpl
  - SystemServiceDelegateRemoteImpl
- wf
  - DefinitionServiceDelegateLocalImpl
  - DefinitionServiceDelegateRemoteImpl
  - ExecutionServiceDelegateLocalImpl
  - ExecutionServiceDelegateRemoteImpl

Все существующие в данный момент Delegate-классы реализуют требуемую функциональность при помощи технологии EJB (stateless session beans), однако в будущем возможны и другие реализации, например, на основе технологии web-сервисов.

Слой Service

Слой service – это серверное API доступа к системе. Реализации Delegate интерфейсов обращаются именно к этому слою. Каждый Delegate работает с одним соответствующим классом Service. В настоящее время все разработанные service классы и интерфейсы ориентированы на EJB-технологию, однако в будущем возможны и другие реализации. Реализации классов Service являются Stateless Session Bean EJB, которые декларативно поддерживают транзакционность вызывов и запрашивают соответствующие классы из слоя Logic.
Таким образом, классы Delegate – Service - Logic образуют как бы «транспорт» между клиентом и сервером.

Слой Logic

Слой Logic – это реализация бизнес логики работы системы. Слой работает с интерфейсами ядра JBOSS JBPM и классами слоя DAO для доступа к постоянному хранилищу.

Слой Dao

Слой Dao – это интерфейсы и классы, обеспечивающие доступ к данным, находящимся в постоянном хранилище (базе данных). Слой реализован только для af-системы, в случае wf-системы доступ производится по-другому (через интерфейсы JBOSS JBPM). В настоящее время все Dao-классы системы реализованы при помощи ORM-средства Hibernate.

Использование Hibernate

Hibernate – ORM (Object/Relational Mapping) средство. Отображает объектную архитектуру на реляционную структуру данных. Допускает настройку (не меняя разработанного кода) на большинство существующих серверов реляционных баз данных:

- MySql
- HSQLDB
- Oracle
- MS SQL Server
- и т.д.

Поддерживает работу с распределенными транзакциями, автоматически создает таблицы для новых классов и т.д. Вся работа с данными внутри Runa WFE ведется только через Hibernate.

Физическое размещение компонент

Список всех размещаемых модулей проекта:

- runa-common.jar
- af.core.jar
- af.logic.jar
- af.service.jar
- af.delegate.jar
- wf.core.jar
- wf.logic.jar

- wf.service.jar
- wf.delegate.jar
- wfe.war
- runa-specific.jar
- wfe-bot.jar
- jbpmdelegation.jar

Дополнительные модули (библиотека с «нашими» патчами)

- jbpm2.core.jar

Более подробное описание модулей.

- runa-common.jar
  Классы, используемые всеми другими компонентами
  Зависимости: нет

  Система авторизации и аутентификации (af):

- af.core.jar
  Базовые классы af-системы (Actor, Group и т.д.)
  Зависимости:
  - runa.commons.jar

- af.logic.jar
  Реализует уровни логики и DAO для af-системы
  Зависимости:
  - runa.commons.jar
  - af.core.jar

- af.service.jar
  Реализует уровень сервисов для af-системы
  Зависимости:
  - runa.commons.jar
  - af.core.jar
  - af.logic.jar

- af.delegate.jar
  Реализует уровень Delegate для af-системы
  Зависимости:
  - runa.commons.jar
  - af.core.jar

  Workflow подсистема (wf):

- wf.core.jar
  Базовые классы для wf
  Зависимости:
  - runa.commons.jar

- wf.logic.jar
  Реализует уровни логики для wf

Зависимости:
- runa.commons.jar
- wf.core.jar
- af.logic.jar
- jbpm2.core.jar

- wf.service.jar
  Реализует уровень сервисов для wf
  Зависимости:
  - runa.commons.jar
  - wf.core.jar
  - wf.logic.jar

- wf.delegate.jar
  Реализует уровень Delegate для wf
  Зависимости:
  - runa.commons.jar
  - wf.core.jar

Web интерфейс:
- wfe.war
  Реализует теги, графические формы и т.д.
  Зависимости:
  - runa.commons.jar
  - af.core.jar
  - wf.core.jar
  - af.delegate.jar
  - wf.delegate.jar

Боты:
- wfe-bot.jar
  Реализует функциональность, связанную с ботами (TaskHandler, BotInvoker и т.д.)
  Зависимости:
  - runa.commons.jar
  - af.core.jar
  - wf.core.jar
  - af.delegate.jar
  - wf.delegate.jar

jbpm-delegation:
- jbpmdelegation.jar
  Реализует механизмы jbpm-delegation, общие для всей обязки (см. RUNA WFE. Руководство разработчика бизнес-процессов).
  Зависимости:
  - runa.commons.jar
  - af.core.jar
  - wf.core.jar
  - af.delegate.jar
  - wf.delegate.jar

jbpm:

- jbpm2.core.jar
  jbpm – это workflow ядро проекта jBoss jBPM. Используется в качестве библиотеки. Релизы Runa WFE содержат откомпилированный код ядра и diff-файл к jbpm 2.0 beta3 (релиз jbpm используемый как основа)
  Зависимости: нет

Описание папок проекта

Папки верхнего уровня.

- af – соответствует формируемым модулям
  - af.core.jar
  - af.logic.jar
  - af.service.jar
  - af.delegate.jar
- bot – соответствует формируемым модулям
  - wfe-bot.jar
- common – соответствует формируемым модулям
  - runa-common.jar
- runa – соответствует формируемым модулям
  - runa-specific.jar
- wf – соответствует формируемым модулям
  - wf.core.jar
  - wf.logic.jar
  - wf.service.jar
  - wf.delegate.jar
  - jbpmdelegation.jar
  - wfe.war
- generated
  Генерируемые файлы

Подробное описание папок проекта.

af/core

Папка соответствует формируемому модулю af.core.jar
Структура папки:

- ru.runa.af
  - organizationfunction
  - presentation

Основные классы и интерфейсы

ru.runa.af

- Identifiable – объект системы безопасности, на который можно дать права
- Permission – разрешение (тип прав на Identifiable)
- Actor – пользователь

- ActorPermission – разрешение на пользователя
- ASystem – специальный объект системы безопасности «Система»
- SystemPermission – разрешение на объект «Система»
- Executor – исполнитель (пользователь или группа пользователей)
- ExecutorPermission – разрешение на исполнителя
- Group – группа пользователей
- GroupPermission – разрешения на группу (дополнительные к наследуемым от Исполнителя)

af/delegate

Папка соответствует формируемому модулю af.delegate.jar
Структура папки:

- ru.runa.af.delegate
  - impl

af/logic

Папка соответствует формируемому модулю af.logic.jar
Структура папки:

- ru.runa.af
  - authentication
  - dao
    - impl
  - logic
  - organizationfunction

Основные классы папки ru.runa.af.authentication

- InternalDBPasswordLoginModule логин-модуль для аутентификации во внутренней базе данных (устанавливается «по умолчанию» в дистрибутиве системы)
- AbstractLdapLoginModule – «заготовка» для аутентификации в LDAP
- ADPasswordLoginModule логин-модуль для аутентификации в AD
- KerberosLoginModule логин-модуль для аутентификации через Kerberos
- NTLMLoginModule логин-модуль для аутентификации через NTLM

ru.runa.af.organizationfunction

- OrganizationFunction – интерфейс, который должны реализовывать конкретные функции над оргструктурой
- OrganizationFunctionConfiguration интерфейс конфигурации для функции над оргструктурой (см механизм delegation в документе RUNA WFE. Руководство разработчика бизнес-процессов).

af/service

Папка соответствует формируемому модулю af.service.jar
Структура папки:

- ru.runa.af.service
  - impl.ejb

af/test

Папка соответствует Cactus-тестам, разработанным для af-подсистемы
Структура папки:

- ru.runa
  - af
    - delegate
    - service
  - junit

wf/core

Папка соответствует формируемому модулю wf.core.jar
Структура папки:

- ru.runa.wf
  - form

wf/delegate

Папка соответствует формируемому модулю wf.delegate.jar
Структура папки:

- ru.runa.wf.delegate
  - impl

wf/jbpmdelegation

Папка соответствует формируемому модулю jbpmdelegation.jar
Структура папки:

- ru.runa
  - af.organizationfunction
  - wf.jbpm.delegation
    - action
    - assignment
    - decision
    - serializer

wf/logic

Папка соответствует формируемому модулю wf.logic.jar
Структура папки:

- ru.runa.wf
  - forms
  - logic

wf/service

Папка соответствует формируемому модулю wf.service.jar
Структура папки:

- ru.runa.wf. service
  - impl.ejb

wf/web

Папка соответствует формируемому модулю wfe.war
Структура папки:

- org.jbpm.web.formgen.format
- ru.runa
  - af.web
    - action
    - form
    - html
    - tag
  - common.web
    - action
    - form
    - html
    - tag
  - wf.web
    - action
    - form
    - forms.format
    - html
      - vartag
    - servlet
    - tag

wf/test

Папка соответствует Cactus-тестам для подсистемы wf
Структура папки:

- ru.runa
- af
  - organizationfunction.impl
  - web.action
  - batchpresentation
  - commons.web.action
- wf
  - delegate
  - jbpm
  - jpdl
  - service
  - web
  - action
  - forms
  - html

bot

Папка соответствует формируемому модулю wfe-bot.jar

Структура папки:

- ru.runa
  - af
    - delegate
      - impl
    - logic
    - service.impl.ejb
  - wf
    - jbpm.delegation.action
    - logic

common

Папка соответствует формируемому модулю runa-common.jar
Структура папки:

- ru.runa. commons
  - hibernate
  - util
  - xml

generated

Автоматически генерируемые файлы, соответствуют используемым в механизме EJB классам для доступа к бинам.

Описание основных ant'овских task'ов

Ant'овские task'и сгруппированы в следующие пакеты

- Основные task'и
  - Корень проекта - SourceForge RUNA WFE (для стандартной версии системы)
  - Common components (вспомогательные операции)
  - Authorization Framework (для af-подсистемы)
  - WF Part of Runa WFE (для wf-подсистемы)

- Специфические для конкретных проектов task'и
  - Runa Specific Part of Runa WFE (вспомогательные операции для проекта КГ Руна)
  - Runa Specific Runa WFE (операции для системы для КГ Руна)
  - Bot Part of Runa WFE (операции, связанные с ботами системы для КГ Руна)

Интерес представляют следующие task'и

- Основные task'и
  - WFE
    - build – собирает все дистрибутивы
    - install.wfe – собирает серверную часть системы и устанавливает ее в jboss
    - dist – создает дистрибутив системы RUNA WFE
    - deploy – собирает из исходных файлов и размещает систему в контейнере Application сервера (удобно использовать при работе через IDE)
    - install.demo.db – заполняет БД demo-данными (устанавливает demo-конфигурацию системы) – заводит demo-пользователей, группы, бизнес-процессы, раздает права и т.д.

    - Authorization Framework
        - test – запускает Cactus-тесты для af-подсистемы
        - report.test – запускает Cactus-тесты и формирует соответствующий отчет для af-подсистемы
    - WF Part of Runa WFE
        - test – запускает Cactus-тесты для wf-подсистемы
        - report.test – запускает Cactus-тесты и формирует соответствующий отчет для wf -подсистемы
- Специфические для конкретных проектов task'и
    - Runa Specific Part of Runa WFE
        - import.db – импортирует пользователей (для проекта КГ Руна)
        - synchronize.db – синхронизирует список пользователей с внешней для WF-системы КИС (для проекта КГ Руна)
    - Runa Specific Runa WFE
        - dist – создает дистрибутив кастомизированной для КГ Руна системы RUNA WFE
        - deploy – собирает из исходных файлов и размещает кастомизированную для КГ Руна систему в контейнере Application сервера (удобно использовать при работе через IDE)
    - Bot Part of Runa WFE
        - start.bots – запускает ботов в кастомизированной для КГ Руна системе
        - stop.bots – останавливает ботов в кастомизированной для КГ Руна системе

Описание ресурсов

Структура папок ресурсов:

- `af` – настройки для подсистемы авторизации и аутентификации
- `bot` – описывает ботов как пользователей системы
- `merge`
- `runa`
    - `bot`
        - `handler` – содержит конфигурации конкретных классов-обработчиков для ботов КГ Руна
    - `orgfunction` – содержит конфигурации оргфункций для Руны
- web
    - `WEB-INF` – содержит jsp-страницы проекта и настройки для Struts
        - `af`
        - `classes – localization bundles`
        - `common`
        - `wf`
    - `calendar` – содержит код и настройки используемого графического календаря
    - `images` – содержит используемые в проекте иконки
- `wf`
    - `orgfunction` - содержит конфигурации настроек функций над оргструктурой
- `wfescript` – содержит командные файлы административных скриптов и скрипты для нескольких демо-конфигураций (подробнее см. документ «RUNA WFE.

Администраторские скрипты»).

Подробное описание наиболее интересных для разработчика ресурсов:

Папка af

`af_logic.properties` – default логин и пароль администратора

`af_delegate.properties` – настройка на удаленные или локальные интерфейсы для слоя Delegate

`login_module.properties` – настройки обязательности-необязательности для login modules подсистемы аутентификации

`ntlm_support.properties` – настройки для NTLM-аутентификации

`orgfunction.properties` – настройки для специального системного пользователя, который используется при вызове оргфункций (будет изменено в дальнейшем)

Папка bot

`bots.xml` – содержит описания и настройки используемых ботов

`bots.xsd` – XML-схема, в соответствии с которой формируется файл `bots.xml`

Краткое описание тегов, используемых для настройки ботов:

Тег <bot> - описывает бота как пользователя WF-системы

Атрибуты тега:

- логин
- пароль

Вложенные элементы:

- тег task

Тег <task> - описывает имена заданий (совпадающих с именами Узлов-Действий бизнес-процесса), которые может выполнять бот. Тег связывает имя задания с классом, который должен обрабатывать задание, также в теге указывается имя конфигурации, содержащей используемые классом данные.

Атрибуты тега:

- name – имя Узла-Действия
- handler – имя класса – обработчика задания
- configuration – имя файла, содержащего настроечные данные для класса – обработчика

Папка runa

Содержит XML-схемы для конфигураций для зданий отмены бизнес-процесса и зданий SQL-боту, а также параметры для utility import_db – импорт WF-пользователей из базы 1С КГ Руна.

Папка runa/bot/handler

Содержит конфигурации конкретных классов-обработчиков для ботов КГ Руна – файлы следующих типов:

- *.xml – файлы
- *.properties – файлы

Конфигурация для SQL – бота (*.xml – файлы). Основные теги:

Тег <database-tasks> - сообщает, что группа заданий предназначена для SQL – бота.

Вложенные элементы:

- тег task

Тег <task> - содержит описание задания для SQL – бота.

Атрибуты тега:

- url – url базы данных (должен содержать собственно url-адрес базы, логин, пароль пользователя, имя используемого jdbc-драйвера)

Вложенные элементы:

- тег queries

Тег <queries> - содержит описание набора SQL запросов.

Вложенные элементы:

- тег query

Тег <queriy> - содержит описание SQL запроса.

Атрибуты тега:

- sql – строка, содержащая параметризованное описание SQL-запроса (вместо значений используется символ “?”

Вложенные элементы:

- тег param
- тег swimlane-param

Тег <param> - содержит описание параметра SQL запроса.

Атрибуты тега:

- placeholder – номер параметра (по порядку) в SQL запросе
- var – имя переменной бизнес-процесса, значение которой будет использовано в качестве значения параметра SQL запроса.

Тег <swimlane-param> - содержит описание параметра SQL запроса, являющегося одним из свойств Роли-Дорожки.

Атрибуты тега:

- placeholder – номер параметра (по порядку) в SQL запросе
- var – имя переменной – Роли-Дорожки бизнес-процесса
- field – имя свойства Роли-Дорожки, значение которого будет использовано в качестве значения параметра SQL запроса.

Конфигурация для SMTP – бота (*.properties – файлы). Настройки:

- smtp.server= - имя почтового сервера
- from= - от кого отправлено
- subject= - тема сообщения, допускает использование тега <customtag> и механизма delegation (т.е. предоставляет возможность использовать значения переменных бизнес-процесса)
- to= - кому (адрес получателя сообщения)
- reply.to= кому отвечать (адрес, по которому надо отправлять ответ на сообщение)
- CC= - копия (адрес получателя копии сообщения)
- BCC= – скрытая копия (адрес получателя скрытой копии сообщения)

Конфигурация для Excel – бота (*.properties – файлы). Настройки:

Для работы бота необходимо, чтобы на компьютере, на котором он будет запущен, библиотека jcom.dll (или соответственно jacob.dll) находилась в папке файловой системы, на которую настроена системная переменная path.

Excel-бот настраивается:
в файле resources/bot/excel.properties (значение
excel.provider.class надо установить равным ru.runa.wfe.bp.commons.JcomExcelProvider (или соответственно провайдеру для jacob-моста между com и Java)

Существующие конфигурации для excel-бота и их параметры:

- resources/runa/bot/handler/overtimes_report.properties – отчет о сверхурочных
- resources/runa/bot/handler/timings_report.properties – личный отчет по учету времени
- resources/runa/bot/handler/worktime_report.properties – табель учета рабочего времени

Параметры конфигураций:

- `excel.template.file= файл, содержащий Excel – макрос (слеши должны быть удвоены – пример D:\\Temp\\Excel\\pattern.xls`
- `excel.function.name – имя макроса`
- `excel.function.variables – список переменных (через запятую) бизнес-процесса, значения которых будут переданы на excel лист 2, начиная с ячейки A1 и вниз (в первый столбец)`
- `excel.report.name=имя создаваемого отчета – xls-файла (например-Worktime.xls)`
- `excel.report.variable – имя переменной бизнес-процесса, в которую будет передан результат`

Папка runa/orgfunction

Содержит конфигурации настроек функций над оргструктурой, специфичных для КГ Руна

Папка wf

- `wf_jbpm.properties` – файл, в котором задается сервер баз данных – **информационное хранилище для данных ядра системы**, а также некоторые настройки hibernate (замечание: информационное хранилище для обвязки задается в файле `$(DIST_ROOT)/hibernate_build.properties`)
- `wf_delegate.properties` – файл, в котором устанавливается – локально или удаленно работает слой delegate (удаленная настройка требуется для **установки удаленных ботов**)
- `wf_logic.properties` – файл, в котором задаются некоторые настройки слоя логики
- `forms.xsd` – XML-схема для элементов форм (которые умеет проигрывать forms player)
- `workflowScript.xsd` - XML-схема для администраторских скриптов (подробнее см. документ «RUNA WFE. Администраторские скрипты»)

Описание tools

Структура папок tools:

- patches – описание патчей – автоматических преобразований таблиц при переходе к новым версиям
    - base64toVarbinary
    - wfe1.0.1to2.0
    - wfe2.0batch_presentation_cleanup
- wfescript
    - conf
    - lib

Также tools содержит скрипты:

- Импорт базы данных для Руны
- Синхронизация базы данных для Руны
- Запуск ботов
- Остановка ботов
- Логин с подстановкой текущего имени пользователя Windows

Как настроить хранилище данных

Хранилище данных для ядра (jboss jbpm).

Настройка на сервер баз данных производится в файле:
<root folder>/resources/wf/wf_jbpm.properties

В этом файле надо определить два свойства:

- hibernate.connection.datasource
- hibernate.dialect

Пример настройки на MS SQL сервер:
hibernate.connection.datasource=java:/MSSQLDS
hibernate.dialect=net.sf.hibernate.dialect.SQLServerDialect

(MSSQLDS – имя конкретного источника данных)

Пример настройки на HSQL сервер:
hibernate.connection.datasource=java:/DefaultDS
hibernate.dialect=net.sf.hibernate.dialect.HSQLDialect

(DefaultDS – имя конкретного источника данных)

Настройка должна производиться перед build'ом системы.

Хранилище данных для обвязки (runa wfe).

Настройка на сервер баз данных производится в файле:

<root folder>/hibernate_build.properties

В этом файле надо определить два свойства:

- `hibernate.connection.datasource`
- `hibernate.dialect`

Пример настройки на MS SQL сервер:

```
hibernate.connection.datasource=java:/MSSQLDS
hibernate.dialect=net.sf.hibernate.dialect.SQLServerDialect
```

(`MSSQLDS – имя конкретного источника данных`)

Пример настройки на HSQL сервер:

```
hibernate.connection.datasource=java:/DefaultDS
hibernate.dialect=net.sf.hibernate.dialect.HSQLDialect
```

(DefaultDS – имя конкретного источника данных)

Настройка должна производиться перед build'ом системы.

Примеры настройки источника данных (для jboss)

MS SQL сервер:
Настройка производится в файле <root folder>/server/default/deploy/mssql-ds.xml, имя источника данных задается в теле тега <jndi-name>, также в файле указывается имя и пароль пользователя, от имени которого будут производиться обращения к базе данных (подробнее см. в прилагаемых к дистрибутиву примерах конфигураций).

HSQL сервер:
Настройка производится в файле <root folder>/server/default/deploy/hsqldb-ds.xml, имя источника данных задается в теле тега <jndi-name>, также в файле указывается имя и пароль пользователя, от имени которого будут производиться обращения к базе данных (подробнее см. в прилагаемых к дистрибутиву примерах конфигураций).

Замечание. В папке …/server/default/lib сервера приложений JBOSS должен находиться jdbc-драйвер для использующегося сервера баз данных.

Как запрограммировать бота

Что такое боты.
Боты это workflow роботы (специальные программные сущности), заменяющие людей - пользователей системы. С каждым ботом должен быть связан обычный пользователь системы (с логином, паролем, описанием, полномочиями и т.д.). Каждый бот регистрируется в специальном приложении (BotInvokerServiceBean), которое периодически активизирует всех зарегистрированных ботов. Бот получает от WF-системы предназначенные связанному с ним пользователю задания и передает их на выполнение сопоставленным им классам-обработчикам заданий. После выполнения задания бот сообщает об этом WF-системе и передает значения соответствующих переменных бизнес-процесса.
Замечание. Регистрация бота собственно и является его определением.

Как регистрируется бот (в проекте, с последующим build'ом бинарной версии).
Бот регистрируется при помощи XML-тегов в файле /resources/bot/bots.xml, находящемся в

папке ресурсов. Соответствующая XML-схема находится в файле /resources/bot/bots.xsd.

Как регистрируется бот (в бинарной версии системы).
Бот регистрируется в файле $(DIST_ROOT)/server/default/conf/bots.xml

Приведем краткое описание основных тегов, определяющих ботов.

Тег bots

Описание тега: Основной тег описания ботов. Содержит список ботов

Вложенные элементы:

Элемент	***Краткое описание***
Bot	Задает конкретный бот

пример тега:

```
<bots                                                xmlns="http://runa.ru/xml"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://runa.ru/xml bots.xsd">

      <bot login="SMTPbot" ...>
            ...
      </bot>

      <bot login=" ExcelBot " ...>
            ...
      </bot>

      <bot login=" HelloBot " ...>
            ...
      </bot>
</bots>
```

Тег bot

Описание тега: Описывает бота в виде специальной сущности

Атрибуты элемента:

Атрибут	***Краткое описание***
login	Логин. Должен совпадать с логином одного из существующих пользователей WF-системы.
password	Пароль. Должен совпадать с паролем этого пользователя.

Вложенные элементы:

Элемент	***Краткое описание***
task	Тип задания, который “умеет” выполнять бот.

пример тега:

```
      <bot login="ExcelBot" password="123">
            <task name = "overtimes report" .../>
            <task name = "timings report " .../>
            <task name = "worktime report " .../>
      </bot>
```

Тег task

Описание тега: Описывает тип задания для бота, ставит ему в соответствие класс-обработчик задания и его конфигурацию.

Атрибуты элемента:

Атрибут	***Краткое описание***
name	Имя задания. Должно совпадать с именем Узла-Действия бизнес-процесса, в котором данный бот должен выполнить задание.
handler	Java класс – обработчик задания, который бот вызовет для выполнения задания. Этот класс должен быть загружен в систему.
configuration	Ссылка на конфигурационный файл, находящийся в /resources/…, файл содержит настройки, которые читает класс – обработчик задания. При помощи изменения настроек можно изменять поведение бота, не меняя класс – обработчик задания. Конфигурационные файлы могут быть файлами двух типов: XML-файлами и *.properties файлами. Эти файлы должны быть загружены в систему.

примеры тега:

```
<task name = "change data shift" handler =
    "ru.runa.wfe.bp.timing.bl.DatabaseTaskHandler" configuration =
    "/bot/handler/shift_insert.xml" />


<task name = "overtimes report" handler =
    "ru.runa.wfe.bp.commons.ExcelTaskHandler" configuration =
    "bot.handler.overtimes_report" />
```

Что такое классы-обработчики заданий.

Каждый бот умеет выполнять некоторый набор типов заданий. Каждый тип задания должен быть связан с java классом – обработчиком задания, который, собственно и будет выполнять поступившее боту задание. Класс-обработчик заданий для бота должен реализовывать интерфейс ru.runa.wf.logic.TaskHandler. Интерфейс содержит два метода:

- handle(Subject subject, TaskStub taskStub) – обработка задания
- configure(String configurationName) – чтение конфигурации

В системе Runa WFE уже разработаны следующие классы-обработчики заданий:

- ru.runa.wfe.bp.timing.bl.DatabaseTaskHandler – операции с таблицами баз данных
- ru.runa.wfe.bp.timing.bl.cancelprocess.CancelProcessTaskHandler – отмена бизнес-процесса и соответствующее изменение связанных с этим процессом внешних данных
- ru.runa.wfe.bp.commons.ExcelTaskHandler – операции с MS Excel
- ru.runa.wfe.bp.commons.HelloTaskHandler – простейший образец обработчика задания
- ru.runa.wf.PrintTaskHandler – помещает все параметры задания в log
- ru.runa.wfe.bp.commons.TextEmailTaskHandler – обработчик, посылающий текстовое сообщение через почтовый сервер

Конфигурации для ботов, работающих с БД задаются XML файлами, соответствующими XML схеме database-tasks.xsd. Аттрибуты элемента <task> этой схемы (driver и url) определяют имя класса JDBC-драйвера для подключения к БД и URL подключения (в соответствии с форматом, поддерживаемым драйвером).
Замечание: URL должно обязательно содержать имя пользователя, под которым осуществляется подключение к БД и его пароль.

Конфигурации ботов, отсылающих e-mail сообщения, задаются файлами свойств (.properties) имеющие кодировку Latin1. Свойство smtp.server указывает SMTP сервер, через который осуществляется отсылка сообщений, свойства from, to, reply.to и subject соответствуют одноименным полям заголовка SMTP.

После загрузки в систему файлы конфигурации обработчиков ботов находятся в директории $(DIST_ROOT)/server/default/conf/bot/handler. Каждый файл конфигурации соответствует записи в файле конфигурации ботов ($(DIST_ROOT)/server/default/conf/bots.xml).

Пример файла свойств:

```
smtp.server=rm_exchange.runa.ru
from=bot@runa.ru
subject=<customtag                                          var="employee"
delegation="ru.runa.wf.web.html.vartag.ActorFullNameDisplayVarTag" />
to=absenceInOffice@runa.ru
```

Пример XML-файла:

```
<?xml version="1.0" encoding="UTF-8"?>
   <database-tasks                              xmlns="http://runa.ru/xml"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://runa.ru/xml database-tasks.xsd">
         <task
            url="jdbc:jtds:sqlserver://RM_SQL;DatabaseName=bp_timing_dem
            o;User=bp_timing_testing_ad;Password=pwd"
            driver="net.sourceforge.jtds.jdbc.Driver">
               <queries>
                  <query sql="update TIMINGS set CONFIRMATION_DATE = ?
                     where INSTANCE_ID = ?">
                        <param  var="currentDate" />
                        <param  var="instanceId" />
                  </query>
               </queries>
         </task>
   </database-tasks>
```

Как осуществить запуск ботов так, чтобы они работали с удаленным сервером (Система Runa WFE размещена на удаленном сервере, боты размещены на текущем серевере, запускаются с текущего сервера).

1. В ресурсах в файле resources/wf/wf_delegate.properties установить `ru.runa.wf.delegate.interface.type=Remote`
2. В ресурсах в файле resources/wf/wfescript_delegate.properties установить `ru.runa.wfescript.delegate.remote.provider.url=jnp://server_name:1099`
3. В ресурсах в файле resources/af/af_delegate.properties установить `ru.runa.af.delegate.interface.type=Remote`
4. В ресурсах в файле resources/af/af_delegate.properties установить `ru.runa.af.delegate.remote.provider.url=jnp://server_name:1099`
5. Запустить ant task «Install»
6. Запустить dist/adminkit/bot-invoker.bat –start (или ant task «Bot Part of Runa WFE / start.bots»

Принудительный вызов ботов с помощью BotInvoker Action Handler.

Для повышения скорости выполнения задач ботами можно явно их вызвать (по умолчанию боты периодически обнаруживают задачи готовые к выполнению с периодом порядка нескольких минут).
Пример принудительного вызова (определяется при разработке бизнес-процесса, находится в файле processdefinition.xml):

```
<transition name="tr1" to="Положить записку об окончании процесса">
  <action event-type="transition">
    <delegation class="ru.runa.wf.jbpm.delegation.action.BotInvokerActionHandler" configuration=”rm:8809”>
    </delegation>
  </action>
 </transition>
```

Обратим внимание на ActionHandler. Название класса – всегда ru.runa.wf.jbpm.delegation.action.BotInvokerActionHandler, атрибут configuration означает адрес Bot station.
Адрес Bot station задается в формате remote_bot_server:bot_rmi_port, где remote_bot_server – название компьютера Bot station (DNS имя), bot_rmi_port – RMI порт Bot station (по умолчанию = 1099, в этом случае указывается только название компьютера)
Заметим, что атрибут configuration не обязателен, в этом случае будет задействована Bot station по умолчанию.

При разработке бизнес-процесса в графическом редакторе нужно проделать следующее:
1. Выделить переход (transition), который ведет к задаче бота.
2. В окне Outline (иерархический вид) по правому клике мыши на нем «Добавить действие»
3. В свойствах созданного узла-действия выбрать класс ru.runa.wf.jbpm.delegation.action.BotInvokerActionHandler и указать (если необходимо) конфигурацию (адрес Bot station)

Особенности тестирования бизнес-процессов до ввода в эксплуатацию на реальные Bot

station.
Чтобы при разработке указывать реальные адреса Bot station, но в то же время не задействовать эти компьютеры предлагается локально прописать алиасы для имен компьютеров Bot station в файле ${windir}\system32\drivers\etc\hosts (/etc/bin/drivers/hosts для Unix).

Особенности обслуживания Bot station.
Т.к. определения бизнес-процессов явно содержат адреса Bot station, то при промышленном использовании системы необходимо эти DNS имена компьютеров зарезервировать для Bot station.
В случае отказа компьютеров с Bot station или при их плановом обслуживании можно изменять на DNS сервере IP-адреса используемых компьютеров и переводить (временно или постоянно) ботов на другие (резервные) Bot station.

TextReport Bot

Полный пример можно найти в проекте WFE (/samples/TextReport/Bot)
Text Report bot работает следующим образом:
На вход он принимает шаблон документа, заполняет его значениями переменный бизнес-процесса и кладет результат как переменную типа файл в бизнес-процесс.

Чтобы вставить значение переменной в шаблоне нужно написать ${MyVariable}. И тогда, в процессе выполнения бот возьмет значение переменной MyVariable и вставит ее на это же место.
Замечание: если запрошенная переменная (например MyVariable) не существует в бизнес-процессе или ее значение равно null – то возникнет исключение `TaskHandlerException`.
Замечание: При желании можно задать формат выводимых переменных. Для этого используйте конструкцию

```
<formatters>
  <variable
      variableName="testDate"
      formatClass="java.text.SimpleDateFormat"
      pattern="dd-MM-yy" />
</formatters>
```

```
, где
  variableName означает название переменной,
  formatClass означает класс формата и
  pattern – параметры, переданные в конструктор при создании формата.
```

```
Конструкция
<replacements xmlFormat="false" applyToRegexp="false" />
в конфигурации определяет алгоритм подмены, где
xmlFormat – является ли шаблон XML документом, это имеет значение для спец.
символов формата XML;
applyToRegexp – применять ли замены к регулярному выражению
"\\$\\{(.*?[^\\\\])\\}".
```

Как реализовать класс-формат для переменных бизнес-процесса

Класс должен наследовать абстрактный класс java.text.Format
Основной метод интерфейса:

- Object parseObject(String source)

То есть класс-формат для переменных бизнес-процесса должен преобразовывать полученную текстовую строку в экземпляр Java-класса (который собственно и будет переменной бизнес-процесса)

Классы-форматы для переменных бизнес-процесса используются в языке jpdl для Runa WFE (в теге <variable> файла forms.xml архива бизнес-процесса) для определения переменных бизнес-процесса.

В настоящее время в RunaWFE существуют следующие классы-форматы для переменных бизнес-процесса:

- ru.runa.wf.web.forms.format.BooleanFormat – формат для логических переменных
- ru.runa.wf.web.forms.format.DateFormat – формат для переменных типа Дата
- ru.runa.wf.web.forms.format.DateTimeFormat – формат для переменных типа Дата - Время
- ru.runa.wf.web.forms.format.TimeFormat – формат для типа Время
- org.jbpm.web.formgen.format.DefaultFormat – формат по умолчанию
- org.jbpm.web.formgen.format.DoubleFormat – формат для числа с плавающей точкой

Как реализовать свой графический элемент для ввода или отображения данных в форме.

Классы, реализующие интерфейс ru.runa.wf.web.html.VarTag, в системе Runa WFE применяются при разработке форм для бизнес-процессов.

Интерфейс содержит одну функцию getHtml(Subject subject, String varName, Object varValue, PageContext pageContext). Здесь varName – имя переменной бизнес-процесса, varValue – значение переменной. В реализующем интерфейс VarTag классе эта функция вернет HTML код, который будет вставлен в соответствующее место HTML-формы для бизнес-процесса.

Графический элемент для ввода или отображения данных в форме бизнес-процесса для системы Runa WFE оформляется в виде тега, «расширяющего» HTML для используемых в Runa WFE форм для бизнес-процессов: <customtag>
Параметры тега:

- var – имя переменной бизнес-процесса
- delegation – имя класса, реализующего интерфейс ru.runa.wf.web.html.VarTag

В настоящее время существуют следующие реализации интерфейса VarTag:

- AbstractActorVarTag – абстрактный класс для визуализации «Руновских» сотрудников
- ActorFullNameDisplayVarTag - класс для визуализации Actor'а (ФИО)
- ActorNameDisplayVarTag - класс для визуализации Actor'а (Имя)
- SubordinateAutoCompletingComboboxVarTag – класс для визуализации иерархии подчиненных
- AbstractDateTimeInputVarTag – абстрактный класс для времени
- DateInputVarTag – класс для ввода даты
- DateTimeInputVarTag – класс для ввода даты и времени

- ComboBoxVarTag - абстрактный класс для выбора Actor'ов из списка
- ActorComboboxVarTag - класс для выбора Actor'ов из списка
- AutoCompletionComboBoxVarTag
- GroupMembersComboboxVarTag - класс для выбора из списка Actor'ов – членов определенной группы
- DateTimeValueDisplayVarTag - класс для визуализации даты и времени
- DateValueDisplayVarTag - класс для визуализации даты
- FileVariableValueDownloadVarTag - класс для визуализации имени файла и возможного download'а
- HiddenDateTimeInputVarTag
- TimeValueDisplayVarTag
- VariableValueDisplayVarTag - класс для визуализации строковых переменных

Как написать функцию над организационной структурой

Функции над организационной структурой в системе Runa WFE применяются, например, при инициализации Ролей-Дорожек.

Класс, соответствующий функции над организационной структурой, должен реализовывать интерфейс ru.runa.af.organizationfunction.OrganizationFunction.

Интерфейс содержит одну функцию long[] getExecutorIds(Object[] parameters). Эта функция должна возвращать массив идентификаторов Исполнителей.

После того, как функция написана и загружена в систему, ее можно использовать при разработке бизнес-процессов.

В настоящее время существуют следующие реализации интерфейса OrganizationFunction:
- ru.runa.af.organizationfunction.ChiefFunction
- ru.runa.af.organizationfunction.ChiefRecursiveFunction
- ru.runa.af.organizationfunction.DemoChiefFunction
- ru.runa.af.organizationfunction.ExecutorByNameFunction
- ru.runa.af.organizationfunction.GroupFunction
- ru.runa.af.organizationfunction.SQLFunction
- ru.runa.af.organizationfunction.SubordinateFunction
- ru.runa.af.organizationfunction.SubordinateRecursiveFunction

В бизнес-процессе (файл processdefinition.XML) обработчик функции над оргструктурой используются при определении Ролей-Дорожек: указывается внутри тега swimlane, в теле тега <delegation>.В качестве параметра class тега <delegation> указывается класс-обработчик для swimlane assignment
ru.runa.wf.jbpm.delegation.assignment.AssignmentHandler
В соответствии с механизмом delegation, в данном случае внутри тега <delegation> содержится конфигурация для этого класса, которая представляет собой имя класса-наследника интерфейса OrganizationFunction и список параметров в скобках. В качестве параметра может выступать конкретное значение или имя переменной бизнес процесса в фигурных скобках, перед которым стоит значок $ (в этом случае в функцию в качестве параметра будет передано значение этой переменной)

Как написать обработчик для Decision

Обработчик для Decision является java-классом, реализующим интерфейс org.jbpm.delegation.DecisionHandler. Интерфейс содержит один метод decide(ExecutionContext executionContext), который возвращает имя выбранного перехода (одного из выходящих из данного Decision'а).

В настоящее время существуют следующие реализации интерфейса DecisionHandler:

- ru.runa.wf.jbpm.delegation.decision.BSFDecisionHandler – обработчик, на основе BSF – скрипта.
- org.jbpm.delegation.decision.ExpressionDecisionHandler – незаконченный класс проекта JBOSS jBpm

Оба класса также являются реализациями интерфейса Configurable, т.е. для них предусмотрена возможность конфигурирования. Интерфейс Configurable содержит один метод configure(String configuration), при помощи которого класс получает свою конфигурацию.

В бизнес-процессе (файл processdefinition.XML) обработчик для Decision указывается в теле <decision> при помощи механизма delegation.В качестве параметра class тега <delegation> указывается класс-обработчик для Decision, внутри тега <delegation> содержится конфигурация для этого класса.

Настройка «толстого» клиента для Windows

«Толстый» клиент – это отдельное приложение, разработанное на базе платформы Eclipse. Приложение содержит «внутри» себя MS Internet Explorer, который связывается с workflow системой по определенному адресу по NTLM протоколу. Параллельно и независимо само толстое приложение связывается с workflow системой по Kerberos протоколу.

Задачи «толстого» клиента:

- Явным образом сообщать пользователю о приходе новых заданий
- Попадать в systray при закрытии и «разворачиваться» оттуда (а также сигнализировать о приходе нового задания, находясь в systray'е)

Приложение - «толстый» клиент настраивается следующим образом:

- В проекте TasksNotifier, файле ./resources/application.properties надо установить в переменную start.browser.url значение URL стартовой страницы WF-системы Runa WFE (пример строки: start.browser.url=http://localhost:8080/wfe/)
- В файле папку ./lib необходимо скопировать библиотеки af.delegate.jar и wf.delegate.jar из WF-проекта, «на который» будет настроено толстое приложение
- В архиве af.delegate.jar надо отредактировать файл af_delegate.properties:
    - Установить ru.runa.af.delegate.interface.type=Remote
    - Установить в переменную ru.runa.af.delegate.remote.provider.url ссылку на WF систему Runa WFE (пример строки:

ru.runa.af.delegate.remote.provider.url=jnp://localhost:1099)

- В архиве wf.delegate.jar надо отредактировать файл wf_delegate.properties:
- Установить ru.runa.wf.delegate.interface.type=Remote
- Далее надо запустить ant'овский task "build" проекта TasksNotifier. В папке ./build/deploy/ будут сгенерированы следующие файлы:
  - rtn.jar
  - runa_tasks.exe
  - swt-win32-3138.dll
- Для запуска «толстого» клиента необходимо поместить все эти файлы в любую папку файловой системы и запустить runa_tasks.exe

Для корректной работы системы также надо проделать действия, описанные в разделе «Настройка Kerberos аутентификации (для толстого клиента)» документа «Установка и эксплуатация системы RUNA WFE в КГ Руна. Инструкция по конфигурированию системы. Версия 2.1»

Описание системы аутентификации - авторизации

Аутентификация.
Аутентификация в системе Runa WFE основана на JAAS (авторизация – нет, авторизация – «собственная»). В соответствии с JAAS для каждого вида должен быть разработан специальный класс, реализующий интерфейс javax.security.auth.spi.LoginModule.
Разработанные в системе Runa WFE LoginModule-классы находятся в папке ru.runa.af.authentication (см. описание соответствующей папки проекта). Разработаны следующие логин-модули:

- для аутентификации во внутренней базе данных (устанавливается «по умолчанию» в дистрибутиве системы)
- «заготовка» для аутентификации в LDAP
- для аутентификации в AD
- для аутентификации через Kerberos
- для аутентификации через NTLM

Авторизация.
Описание логического механизма системы авторизации изложено в документе «RUNA WFE. Руководство администратора».

Реализация системы авторизации находится в папке ru.runa.af (см. описание папки проекта)

Настройка поддержки аутентификации через AD/LDAP

Для настройки AD аутентификации необходимо сконфигурировать следующие файлы:

В бинарной версии:

- runawfe-2.0RC3_root_folder\server\default\conf\login_module.properties
- runawfe-2.0RC3_root_folder\server\default\conf\ad_password_login_module.properties

(В проекте они находятся в ./resources/af папке)

Конфигурирование состоит в следующем:

- Раскомментируется строка ru.runa.af.authenticaion.ADPasswordLoginModule=SUFFICIENT в login_module.properties файле.
- Вводится имя AD/LDAP сервера и домена соответственно в переменные ru.runa.af.active.directory.server.url и ru.runa.af.active.directory.damain.name properties в файле ad_password_login_module.properties

В бинарной версии после этого необходимо перезапустить jBoss сервер. В проекте запустить build системы.

Далее в системе Runa WFE надо завести пользователей с такими же логинами, как и в AD/LDAP и дать им права на login (пароль для них задавать не надо). После этого при входе в систему под этими пользователями пароль будет проверяться в AD/LDAP.

Настройка поддержки NTLM аутентификации

Для настройки NTLM аутентификации необходимо сконфигурировать следующие файлы:

В бинарной версии:

- runawfe-2.0RC3_root_folder\server\default\conf\login_module.properties
- runawfe-2.0RC3_root_folder\server\default\conf\ntlm_support.properties

(В проекте они находятся в ./resources/af папке)

Конфигурирование состоит в следующем:

- Раскомментируется строка ru.runa.af.authenticaion.NTLMLoginModule=SUFFICIENT в login_module.properties файле.
- Вводится имя домена в переменную domain, в переменную ntlm_supported вводится значение true (в файле ntlm_support.properties)

В бинарной версии после этого необходимо перезапустить jBoss сервер. В проекте запустить build системы.

Для NTLM аутентификации в системе Runa WFE разработан LoginModule - класс ru.runa.af.authenticaion.NTLMLoginModule.

Инициализация ролей-дорожек и система заместителей

В системе существует возможность в некоторых случаях (например, если исполнитель текущего задания болеет) перенаправлять задания другим исполнителям. Реализовано это при помощи правил замещения и статуса пользователя.

Роли-Дорожки и их инициализация

В бизнес-процессе определен набор специальных локальных переменных Ролей-Дорожек (см. Глоссарий), с каждой Ролью-Дорожкой связан специальный оператор - Инициализатор.

Каждому Действию бизнес-процесса поставлена в соответствие одна из Ролей-Дорожек.

Инициализация Роли-Дорожки состоит в том, что Инициализатор ставит ей в соответствие множество Пользователей.

Сужение Роли-Дорожки состоит в том, что из множества Пользователей, поставленных ей в соответствие, исключаются все Пользователи, кроме одного.

Алгоритм работы Инициализатора определяется Формулой инициализации над Исполнителями, переменными бизнес-процесса и Функциями над Исполнителями.

Формула инициализации представляет собой оператор присваивания, в левой части которого находится имя Роли-Дорожки, а в правой Исполнитель, переменная бизнес-процесса или Функция над Исполнителями с набором фактических параметров.

Списки заданий.

Для каждого Пользователя формируются задания, состоящие из:

- Задания данному пользователю
- Задания другим пользователям, перенаправленные данному пользователю по замещению

Списки перенаправленных заданий определены в разделе «Правила назначения преемника».

Если роль-дорожка проинициализирована группой пользователей, то выполнить такое задание может любой член группы. После того, как какой-либо Пользователей, входящих в группу, выполнит задание, эта роль-дорожка реинициализируется данным пользователем.

Статус пользователя.

У Пользователя может быть один из следующих статусов:

- Активен (присутствует)
- Не активен (отсутствует)

Правила назначения заместителя.

В некоторых случаях задание может быть перенаправлено другому Пользователю (заместителю).

Заместитель назначается при помощи набора правил замещения Пользователя. Набор является упорядоченным списком правил.

В общем случае правило является функцией над оргструктурой, которая возвращает заместителя.

Предусмотрен также стандартный тип правила, параметры которого можно задать через графический интерфейс системы. Список параметров этого типа правила:

- Замещаемый Пользователь (Пользователь)
- Заместитель (Пользователь, Функция над оргструктурой, возвращающая Пользователя)
- Применимо ли правило (Логическая функция)

Пример правила назначения заместителя:

- Иванов
- Петров
- (Роль-Дорожка = «инспектораКадровойСлужбы») & (Бизнес-процесс= «больничный»)

Порядок применения правил замещения Пользователя.

В случае, если Пользователь имеет статус «Не активен» (задание инициализировано обязательно пользователем, а не группой, в которую входит пользователь), то из списка правил будут выбраны все правила замещения, относящиеся к данному Пользователю. Далее из этих правил будет выбрано первое по порядку правило, которому соответствуют характеристики задания и заместитель по которому имеет статус «Активен». В списке заданий этого Пользователя (заместителя) и будет показано данное задание.

Замечание. Возможны ситуации, в которых у Пользователя не будет заместителя.

Замечание. Когда пользователь получает статус «Не активен», никакие значения Ролей-Дорожек не изменяются. Задания этого пользователя появляются в соответствующих списках заданий заместителя.

Замечание. Роль-Дорожка, поставленная в соответствие Действию, указывается на диаграмме бизнес-процесса (в верхней части Действия, в круглых скобках).

Замечание. Роль-Дорожка, соответствующая пользователю, запустившему процесс, указывается на диаграмме бизнес-процесса (непосредственно над точкой старта, в круглых скобках).