

RunaWFE. Руководство разработчика.

Версия 3.0

© 2004-2011, ЗАО “Руна”. RunaWFE является системой с открытым кодом и распространяется в соответствии с LGPL лицензией (<http://www.gnu.org/licenses/lgpl.html>).

Введение

[См. о проекте ^[1]]

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 действий
- 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]).
2. **Среда разработки** - Eclipse от IBM (<http://www.eclipse.org> ^[3]).
3. **Средство генерации кода и дескрипторов** – xdoclet (<http://xdoclet.sourceforge.net> ^[4]).
4. **Система контроля версий** – subversion (<http://subversion.tigris.org/> ^[5]).
5. **Сборщик приложений** – ant (<http://ant.apache.org> ^[6]).
6. **Сервер баз данных** – поддерживаются сервера БД:
 - MS SQL Server (<http://www.microsoft.com/sql/evaluation/default.msp> ^[7])
 - MySQL (<http://www.mysql.com> ^[8])
 - HSQLDB (<http://hsqldb.org> ^[9])
 - Oracle (<http://www.oracle.com>) ^[10]
 - Postgres (<http://postgresql.com> ^[11])
7. **Операционная система** — поддерживаются следующие ОС:
 - Windows (Server 2000-2008, XP, Vista, 7)
 - ALTLinux
 - Mandriva Linux
 - Fedora
 - Debian/Ubuntu

Подробное описание текущей программной архитектуры системы

Проект разбит на несколько подпроектов, объединяющих логически связанные компоненты.

Структура подпроектов.

- wfe - основной подпроект системы RunaWFE
- customization – дополнительные элементы, которые могут разрабатывать пользователи системы
 - оргфункции
 - VarTags
 - FTLLTags
 - DecisionHandlers
 - Валидаторы
 - форматтеры переменных
 - ActionHandlers
- web – web-интерфейс системы
- bots – боты и бот-станции
- gpd – графический редактор бизнес-процессов
- rtn – клиент-оповещатель о поступивших заданиях
- abs – файлы необходимые для работы автосборки.
- os-specific- определение целей и свойств, необходимые для корректной сборки RunaWFE на разных ОС.
- docs - Документация по проекту.

Основные подсистемы.

- jbpwm - ядро workflow системы (Заимствовано у проекта JBOSS JBPM, в ядро внесено большое количество исправлений) (располагается в подпроекте wfe)
- af - подсистема авторизации и аутентификации. Не зависит от других подсистем, допускает независимое от RunaWFE использование. (располагается в подпроекте wfe)
- wf – собственно workflow подсистема. Зависит от af и jbpwm. (располагается в подпроекте wfe)
- web интерфейс (располагается в подпроекте web)

RunaWFE - Workflow-окружение

Основная функциональность RunaWFE сосредоточена в двух подсистемах:

- af - подсистема авторизации и аутентификации.
- wf – собственно workflow подсистема. Зависит от af.

Описание слоев архитектуры системы

Каждая подсистема включает в себя несколько логических компонентов, объединяющих взаимосвязанные действия. Каждый компонент состоит из нескольких уровней (слоев).

Список слоев:

- delegate
- service
- logic
- dao
- hibernate

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 – содержит методы для аутентификации пользователей в системе (с использованием логина/пароля, Kerberos и т.п.). Так же позволяет получить текущего пользователя из Subject'a.
 - AuthorizationServiceDelegate — содержит методы для работы с полномочиями пользователей (назначение полномочий, проверка на допустимость операции для пользователя и т.п.).
 - BotsServiceDelegate — содержит методы для работы с ботстанциями и ботами.
 - ExecutorServiceDelegate — содержит методы для работы с исполнителями. Позволяет создавать/удалять/изменять исполнителей а так же включать их в группы.
 - ProfileServiceDelegate — содержит методы для работы с профайлам пользователей.
 - SubstitutionServiceDelegate — содержит методы для управления правилами замещения.
 - SystemServiceDelegate — содержит методы для входа/выхода пользователя в систему
- wf

- `DefinitionServiceDelegate` — содержит методы для работы с определениями процессов.
- `ExecutionServiceDelegate` — содержит методы для работы с исполняемыми экземплярами процессов

Все существующие в данный момент Delegate-классы реализуют требуемую функциональность при помощи технологии EJB (stateless session beans).

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-технологии, однако в будущем возможны и другие реализации. Реализации классов Service являются Stateless Session Bean EJB, которые декларативно поддерживают транзакционность вызовов 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 RunaWFE ведется только через Hibernate.

Интерфейсы, доступные в качестве web-сервисов

В настоящее время через web сервисы доступна только ограниченная функциональность RunaWFE достаточная для реализации основного взаимодействия с системой. В будущем предполагается реализовать интерфейс системы через web сервисы в полном объеме.

WSDL web сервисов доступен по адресам `/wf.webservice/ExecutionBean?wsdl` и `/af.webservice/AuthenticationBean?wsdl`

Физическое размещение компонент

Список всех размещаемых модулей проекта:

- runa-common.jar
- af.core.jar
- af.logic.jar
- af.service.jar
- af.delegate.jar
- af.webservice.jar
- wf.core.jar
- wf.logic.jar
- wf.service.jar
- wf.delegate.jar
- wf.webservice.jar
- wfe.war
- wfe-bot.jar
- wfe-botstation.jar
- jbpmdelagation.jar

Дополнительные модули (библиотека jbrpm с «нашими» патчами)

- jbrpm.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.common.jar
- af.core.jar
- af.webservice.jar

Реализует web сервисы для af-системы

Зависимости:

- runa.common.jar
- af.core.jar
- af.logic.jar

Workflow подсистема (wf):

- wf.core.jar

Базовые классы для wf

Зависимости:

- runa.common.jar
- wf.logic.jar

Реализует уровни логики для wf

Зависимости:

- runa.common.jar
- wf.core.jar
- af.logic.jar
- jbp2.core.jar
- wf.service.jar

Реализует уровень сервисов для wf

Зависимости:

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

Реализует уровень Delegate для wf

Зависимости:

- runa.common.jar
- wf.core.jar
- af.webservice.jar

Реализует web сервисы для wf-системы

Зависимости:

- runa.common.jar
- wf.core.jar
- wf.logic.jar
- wfe-botstation.jar

Реализует функциональность необходимую для поддержки ботов

Зависимости:

- runa.common.jar
-

- af.core.jar
- wf.core.jar
- af.delegate.jar
- wf.delegate.jar

Web интерфейс:

- wfe.war

Реализует теги, графические формы и т.д.

Зависимости:

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

jbpmm-delegation:

- jbpmmdelegation.jar

Реализует механизмы jbpmm-delegation, общие для всей обвязки (см. RunaWFE. Руководство разработчика бизнес-процессов).

Зависимости:

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

jbpmm:

- jbpmm.core.jar

jbpmm — это workflow ядро проекта jBoss jBPM. Используется в качестве библиотеки. Релизы RunaWFE содержат откомпилированный код ядра.

Зависимости: нет

Описание основных ant'овских task'ов

Наиболее важные и используемые задания определены в корне:

- `install.wfe` — собирает и устанавливает RunaWFE.
- `install.simulation` — собирает и устанавливает симулятор RunaWFE.
- `install.remote.bots` — собирает и устанавливает удаленную ботстанцию RunaWFE.
- `test.wfe` — тестирует RunaWFE прогоняя тесты из `wfe` и `web`. Перед началом тестирования `jboss` будет запущен заданием самостоятельно.

Основные задания в подпроектах.

1 wfe

- `dist` — собирает библиотеки подпроекта `wfe`.
- `deploy` — копирует библиотеки RunaWFE и настройки в `jboss`. При необходимости библиотеки собираются.
- `install` — копирует библиотеки RunaWFE и настройки в `jboss`. Так же копируются используемые сторонние библиотеки и настраиваются файлы `jboss` для работы RunaWFE.
- `test` — тестирует систему, прогоняя тесты из `af.build` и `wf.build`. При запуске тестирования `jboss` с RunaWFE должен быть запущен.

2 web

- `copy.libs` — синхронизирует библиотеки RunaWFE необходимые для сборки подпроекта с содержимым `jboss`.
- `dist` — собирает библиотеки подпроекта `web`
- `deploy` — копирует библиотеки RunaWFE и настройки в `jboss`. При необходимости библиотеки собираются.
- `test` — тестирует систему, прогоняя тесты из `af.build` и `wf.build`. При запуске тестирования `jboss` с RunaWFE должен быть запущен.

3 bots

- `copy.libs` — синхронизирует библиотеки RunaWFE необходимые для сборки подпроекта с содержимым `jboss`.
- `dist` — собирает библиотеки подпроекта `web`
- `deploy` — копирует библиотеки RunaWFE и настройки в `jboss`. При необходимости библиотеки собираются.

4 rtn

- `copy.libs` — синхронизирует библиотеки RunaWFE необходимые для сборки подпроекта с содержимым `jboss`.
- `build` — собирает `rtn`.

5 gpd

- `copy.libs` — синхронизирует библиотеки RunaWFE необходимые для сборки подпроекта с содержимым `jboss`.
- `build` — собирает `gpd`. Сборка происходит сразу для нескольких платформ с использованием `delta-pack`.

6 abs

- `build` — запускает автосборку.

Описание подпроектов

Подпроект wfe

Подпроект содержит ядро workflow. Основные компоненты содержащиеся в подпроекте:

- общеиспользуемые файлы
- подсистема авторизации (af)
- поддержка ботов и ботстанций
- jbpm с необходимыми изменениями
- workflow подсистема (wf)

Основные папки подпроекта:

- resources — содержит ресурсы подпроекта
- samples — содержит демонстрационные процессы
- src — содержит исходные файлы.
- lib — содержит необходимые для сборки и работы библиотеки

Структура исходных файлов

- af — подсистема авторизации. Компилируется в af.core.jar, af.logic.jar, af.service.jar, af.delegate.jar, af.weservice.jar. Также содержит тесты для подсистемы авторизации (подпапка test).
- bot — поддержка ботов и ботстанций. Компилируется в wfe-botstation.jar.
- common — общеиспользуемые файлы. Компилируется в runa-common.jar.
- jbpm — ядро системы (jbpm). Компилируется в jpm.core.jar.
- wf — workflow подсистема. Компилируется в wf.core.jar, wf.logic.jar, wf.service.jar, wf.delegate.jar, wf.weservice.jar. Также содержит тесты для workflow подсистемы.

Структура ресурсов (описаны только наиболее интересные настройки)

- adminkit — административные скрипты, которые при сборке помещаются в JBOSS_HOME/adminkit.
- af — настройки для подсистемы авторизации
- af_delegate.properties — настройка делегатов для подсистемы авторизации.
- af_logic.properties — задает свойства для системного администратора по умолчанию а также список действий, которые необходимо выполнить при входе исполнителя в систему и изменению статуса исполнителя.
- kerberos_module.properties — настройки для kerberos аутентификации по RMI.
- login_module.properties — настройки модулей логирования. Разрешает или запрещает аутентификацию с использованием какого-либо модуля.
- ntlm_support.properties — настройки для ntlm аутентификации по RMI.
- bot — настройки ботстанции.
- bot_invoker.properties — задает периодичность вызова ботстанции и класс для вызова ботов.
- distr-build — файлы, используемые при сборке различных дистрибутивов.
- jboss-configuration — файлы, используемые при настройке конфигураций jboss.
- remote-bots — файлы используемые при сборке удаленной ботстанции.
- wf — настройки для подсистемы workflow
- graph.properties — настройки генерации графа процессов.
- runa_loading.properties — порядок, в котором библиотеки RunaWFE грузятся при старте системы.
- wf_logic.properties — основным параметром является период, с которым логи для законченных процессов переносятся в отдельные таблицы.

Все компоненты, предоставляющие клиентское API построены с использованием описанных выше слоев. В целях оптимизации в некоторых компонентах применяется кэширование, которое может встречаться на уровне logic или DAO. Основная функциональность по управлению кэшами находится в классах

CachingLogic, WFRunaHibernateInterceptor и классах соответствующих кешей. Изменения в кешируемых объектах перехватываются в WFRunaHibernateInterceptor и сообщаются посредством CachingLogic всем подписавшимся на изменения классам кешей. Транзакция, изменяющая кеш, считается закрытой по окончании обработки ejb вызова.

При внесении изменений в систему и её развитии иногда приходится менять схему базы данных. Для таких случаев предусмотрена функциональность по применению патчей над базой данных. Патчи применяются при старте системы и в общем случае могут не только менять структуру базы данных, но и делать какие-либо другие действия. Для добавления патча необходимо:

1. Написать патч, реализующий интерфейс `ru.runa.common.dbpatch.DBPatch`. В методе `apply` патч должен проделать всю работу и вернуть `true` в случае если патч успешно применен к системе.
2. В классе `ru.runa.af.logic.InitializerLogic` добавить написанный патч в конец массива `dbPatches`.

JBPM все действия над процессами логирует, записывая логи в таблицу `JBPM_LOG`. Со временем таблица логов быстро растет, что приводит к сильному замедлению работы (в основном операций, влекущих за собой вставку в таблицу логов). Для исправления ситуации в RunaWFE добавлена функциональность по переносу логов завершенных процессов в отдельные таблицы для уменьшения размера `JBPM_LOG`. Периодичность, с которой система проверяет на наличие логов, требующих переноса задается параметром `ru.runa.wf.logrotation.period` в файле `wf_logic.properties`. В случае, если параметр отрицательный, автоматического переноса логов не происходит (В этом случае желательно организовать перенос логов другим способом).

В подпроекте так-же содержится реализация части клиентского API RunaWFE в виде web-сервисов. Реализация располагается в папках `src/af/webservice` и `src/wf/webservice`. В настоящее время размер части клиентского API RunaWFE, реализованного через web-сервисы продолжает увеличиваться.

Подпроект *customization*

Содержит расширения RunaWFE. Основные компоненты содержащиеся в подпроекте:

- `actionHandler`'ы — обработчики, которые используются в бизнес-процессах.
- `decisionHandler`'ы — компоненты, принимающие решения и использующиеся в элементе `decision`.
- форматтеры — используются для преобразования значений к текстовому представлению и обратно.
- Орг-функции — используются при назначении ролей и в правилах замещения для определения заместителя.
- валидаторы — используются для проверки переменных в формах.
- `varTags` — генерирует html код для отображения переменной в web интерфейсе.
- `ftlTags` - генерирует html код для отображения переменной в web интерфейсе.

Основные папки подпроекта:

- `resources` — содержит ресурсы подпроекта.
- `src` — содержит исходные файлы. Структура исходных файлов соответствует компонентам.
- `lib` — содержит необходимые для сборки и работы библиотеки.

Структура ресурсов (описаны только наиболее интересные настройки)

- `ActionHandlers` — папка содержит настройки для различных обработчиков (`ActionHandler`'ы).
 - `autoShowForm.properties` — если после выполнения действия в процессе существует форма, которую надо показать текущему пользователю, то она будет немедленно показана вместо перенаправления к списку заданий (в случае установки `auto.show.form` в `true`).
 - `confirmationPopup.properties` — настройки для диалогов подтверждения действий.
 - `emailTaskNotifier.properties` — настройки для оповещения пользователей о новых заданиях по email.
-

Подпроект web

Содержит web интерфейс системы RunaWFE.

Web интерфейс построен на основе технологий JSP и Struts.

Основные папки подпроекта:

- resources — содержит ресурсы подпроекта.
- src — содержит исходные файлы.
- lib — содержит необходимые для сборки и работы библиотеки.

Структура ресурсов (описаны только наиболее интересные настройки)

- merge tooltip web — используются при создании war файла
- portlet — содержат файлы специфичные для сборки web интерфейса в виде портлетов
- kerberos_web_support.properties ntlm_support.properties — настройки для kerberos и ntlm аутентификации через браузер.

Web интерфейс написан таким образом, что бы сборка версии в виде портлетов происходила из тех же исходников, что и обычный web интерфейс. Для поддержания исходных текстов в корректном для сбора портлетов состоянии необходимо:

- URL'и в jsp страницах формировать вызовом `html:rewrite` (См. `main_layout.jsp`).
- Для формирования URL в java коде использовать методы из класса `ru.runa.common.web.Commons`.

Для установки RunaWFE с портлетами необходимо:

1. В качестве jboss взять jboss-portal
2. Установить RunaWFE (`ant install.wfe` в корне проекта)
3. Собрать подпроект web с заданием `portlet.dist` (`ant portlet.dist`).
4. Заменить `wfe.war` файл находящийся в `JOSS_HOME/server/default/deploy` на файл, содержащийся в папке `deploy` (собранный на предыдущем шаге).
5. Скопировать `jbpm.core.jar` из `server/default/deploy` в `server/default/lib`
6. Удалить `jbpm-*.jar` из `server/default/deploy/jboss-portal.sar`

Подпроект bots

Содержит ботов для системы RunaWFE и стандартные исполнители ботов.

Основные папки подпроекта:

- resources — содержит ресурсы подпроекта.
- src — содержит исходные файлы.
- lib — содержит необходимые для сборки и работы библиотеки.

Структура ресурсов (описаны только наиболее интересные настройки).

В основном в ресурсах содержатся настройки для ботов по умолчанию. Особый интерес представляют файлы:

- `bot_invoker.properties` — настройки вызывателя ботов (класс вызывателя и период вызова ботов)
 - `botstation.xml` — настройки ботстанции. Пользователь под которым должна работать ботстанция и количество потоков, в которых выполнять задания.
-

Подпроект **rtn**

Содержит клиент-оповещатель системы RunaWFE.

Основные папки подпроекта:

- `resources` — содержит ресурсы подпроекта.
- `src` — содержит исходные файлы.
- `lib` — содержит необходимые для сборки и работы библиотеки.

Подпроект **gpd**

Содержит графический редактор процессов. Документацию по разработке `gpd` см. в `Process-editor_Developer_guide`

Подпроект **os-specific**

Содержит ant'кие свойства и задания специфичные для различных платформ.

Подпроект **abs**

Содержит файлы автосборки.

Обработчики («Handler»-интерфейсы)

В системе RunaWFE есть возможность вызвать выполнение java-кода при наступлении определенных событий в бизнес-процессе, например, проход точки управления по определенному переходу. Java код должен быть реализован в методе `execute()` класса, реализующего соответствующий Handler-интерфейс. Этот класс должен быть загружен в систему.

Список Handler-интерфейсов:

- `ActionHandler` — интерфейс, дающий возможность определять собственные реализации классов `Action`
- `TaskHandler` - интерфейс, дающий возможность реализовывать алгоритм для автоматического выполнения задачи ботом
- `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.

Замечание про `BSHActionHandler`.

BSHActionHandler используется для пересчета значений переменных БП.

Конфигурацией является корректный код `BeanShell` (www.beanshell.org ^[12]), который по синтаксису очень похож на `Java` и из которого можно делать вызов в `Java`.

Пример конфигурации:

```
My_date = new java.util.Date();  
My_rnd = new java.util.Random(1000).nextInt();  
My_time = java.lang.System.currentTimeMillis();
```

Измененные в процессе исполнения скрипта переменные изменяют свое значение в БП. Будут изменены лишь переменные, которые до этого уже существовали в БП, создание новых не поддерживается.

Как написать обработчик (ActionHandler)

Обработчик (ActionHandler) — компонент системы RunaWFE использующиеся для выполнения каких либо действий при возникновении определенных событий в системе (проход по определенному переходу в бизнес-процессе, приход или уход управления из узла и т.п.). Все входящие в поставку RunaWFE обработчики находятся в подпроекте customization.

Обработчик должен реализовывать интерфейс `org.jbpm.graph.def.ActionHandler`. Создание экземпляра обработчика в системе RunaWFE происходит с использованием конструктора по умолчанию и передачей параметров обработчику вызовом метода `setConfiguration(String)`.

При написании нового обработчика рекомендуется наследовать его от абстрактного класса `BaseActionHandler` (находящегося в том же подпроекте customization). В методе `setConfiguration` обработчик должен запомнить свои параметры (если обработчик их ожидает), а в методе `execute` происходит непосредственная обработка и выполнение требуемых действий.

В случае, если при выполнении обработчика произошла ошибка, произойдет откат транзакции и состояние процесса будет возвращено к заданию, исполнение которого повлекло вызов обработчика.

Как написать обработчик для Decision (ветвление)

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 ветвления).

В настоящее время существуют следующие реализации интерфейса `DecisionHandler`:

- `ru.runa.wf.jbpm.delegation.decision.BSFDDecisionHandler` — обработчик, на основе BSF — скрипта. (находится в подпроекте customization)
- `org.jbpm.delegation.decision.ExpressionDecisionHandler` — незаконченный класс проекта JBOSS jBpm

Классы могут быть сконфигурированы с использованием конструктора принимающего строку, либо с использованием конструктора по умолчанию и метода `setConfiguration`. Выбор способа конфигурирования зависит от исполняемого бизнес-процесса и рекомендуется реализовывать оба метода.

В бизнес-процессе (файл `processdefinition.XML`) обработчик для Decision указывается в теле `<decision>` при помощи подэлемента `handler`. В качестве параметра `class` тега `<delegation>` указывается класс-обработчик для Decision, параметр `config-type` указывает на способ создания и конфигурирования класса (по умолчанию через `setConfiguration`), а внутри тега `<handler>` содержится конфигурация для этого класса.

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 jpdI для RunaWFE (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 (все, кроме 2-х последних находятся в подпроекте customization):

- `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 написать функцию над организационной структурой

Функции над организационной структурой в системе RunaWFE применяются, например, при инициализации Ролей-Дорожек и в правилах замещения.

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 бизнес-процессе (файл `processdefinition.XML`) обработчик функции над оргструктурой используется при определении Ролей-Дорожек: указывается внутри тега `swimlane`, в теле тега `<assignment>`. В качестве параметра `class` тега `<assignment>` указывается класс-обработчик для инициализации `swimlane`: `ru.runa.wf.jbpm.delegation.assignment.AssignmentHandler`

Внутри тега `<assignment>` 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 реализовать свой графический элемент для ввода или отображения данных в форме.

Классы, реализующие интерфейс `ru.runa.wf.web.html.VarTag`, в системе RunaWFE 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 RunaWFE оформляется в виде тега, «расширяющего» HTML для используемых в RunaWFE 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`:

- ActorFullNameDisplayVarTag - класс для визуализации Actor'a (ФИО)
- ActorNameDisplayVarTag - класс для визуализации Actor'a (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'a
- HiddenDateTimeInputVarTag
- TimeValueDisplayVarTag
- VariableValueDisplayVarTag - класс для визуализации строковых переменных

Как реализовать FreeMarker тег для формы.

Классы, расширяющие абстрактный класс `ru.runa.wf.web.ftl.freemarker.Tag`, в системе RunaWFE применяются при разработке FTL форм для бизнес-процессов.

Нужно реализовать 2 метода:

Нужно вернуть кол-во аргументов требуемых для работы тега

protected abstract int getParametersCount();

Исполнение тега:

protected abstract Object executeTag() **throws** TemplateModelException;

Во время исполнения тега имеется доступ к проинициализированным переменным типа Subject, PageContext и списку переменных формы. Возвращаемое значение может быть любым, с этим объектом далее можно проделывать любые операции с помощью freemarker expression language.

Пример использования:

```
public class GroupMembersTag extends FreemarkerTag {

    @Override
    protected Object executeTag() throws TemplateModelException {
        String varName = getParameterAs(String.class, 0);
        String groupVarName = getParameterAs(String.class, 1);
        String view = getParameterAs(String.class, 2);
        String groupName = (String) variables.get(groupVarName);

        List<Actor> actors = getActors(subject, groupName);
        if ("all".equals(view)) {
            return getHtml(actors, varName);
        } else if ("raw".equals(view)) {
            return actors;
        } else {
            throw new TemplateModelException("Unexpected value of VIEW parameter: " + view);
        }
    }
}
```

```
    }  
}  
  
@Override  
protected int getParametersCount() {  
    return 3;  
}  
  
}
```

Описание системы аутентификации - авторизации

Аутентификация.

Аутентификация в системе RunaWFE 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 RunaWFE `LoginModule`-классы находятся в папке `ru.runa.af.authentication`. 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 «RunaWFE. Руководство администратора».

Реализация системы авторизации находится в папке `ru.runa.af`

Инициализация ролей-дорожек и система заместителей

(Этот раздел будет перенесен в руководство администратора, немножко в руководство пользователя и здесь останется только ссылка.)

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 или группам, в которые входит пользователь
- Задания другим пользователям, перенаправленные данному пользователю по замещению

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

Если роль-дорожка проинициализирована группой пользователей, то выполнить такое задание может любой член группы. После того, как какой-либо из Пользователей, входящих в группу, выполнит задание, эта роль-дорожка реинициализируется данным пользователем и он в дальнейшем должен будет выполнять все задания, присвоенные рол-дорожке. Если необходимо что бы каждый раз при приходе управления в задание, роль-дорожка нициализировалась заново (что бы выполнение задания мог выполнять не только пользователь, выполнивший предыдущее задание, с указанной роль-дорожкой), то при разработке бизнес-процесса для необходимо отметить галочку 'реинициализировать роль-дорожку'.

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).

Примечания

- [1] http://wf.runa.ru/rus/doc/RunaWFE._%D0%9E%D0%B1%D1%89%D0%B5%D0%B5_%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5.
- [2] <http://www.jboss.org/>
- [3] <http://www.eclipse.org/>
- [4] <http://xdoclet.sourceforge.net/>
- [5] <http://www.cvshome.org/>
- [6] <http://ant.apache.org/>
- [7] <http://www.microsoft.com/sql/evaluation/default.msp>x
- [8] <http://www.mysql.com/>
- [9] <http://hsqldb.org/>
- [10] <http://www.oracle.com/>
- [11] <http://postgresql.com/>
- [12] <http://www.beanshell.org/>