ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ DOCUMINO.Архив

ФУНКЦИОНАЛЬНЫЕ ХАРАКТЕРИСТИКИ

Листов 47

2021

СОДЕРЖАНИЕ

1. ТЕРМИНЫ, СОКРАЩЕНИЯ И ОПРЕДЕЛЕНИЯ

В настоящем документе применены следующие термины, сокращения и их определения:

Термины/Сокращения	Определения
ACL (Access Control List)	Список управления доступом, который определяет, кто или что может получать доступ к объекту (программе, процессу или файлу), и какие именно операции разрешено или запрещено выполнять субъекту (пользователю, группе пользователей)
Active Directory (AD)	Службы каталогов корпорации Microsoft для операционных систем семейства Windows Server. Начиная с Windows Server 2008, включает возможности интеграции с другими службами авторизации, выполняя для них интегрирующую и объединяющую роль
API	Набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах
Jar -файл	Архив Java (сокращение от Java ARchive)
XQL (eXtended Query Language)	Текстовый язык доступа к данным ПО Documino.Архив. Представляет собой SQL-подобный язык, посредством которого клиентский код взаимодействует с платформой
Алгоритм SHA	Безопасный алгоритм хеширования
ОС	Операционная система
Плагин	Независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей
ПО Documino.Архив	Программное обеспечение Documino.Архив
Скрипт	Программа, которая автоматизирует некоторую задачу, которую без сценария пользователь делал бы вручную, используя интерфейс программы
СУБД	Система управления базами данных
Хэш	Результат преобразования массива входных данных произвольной длины в (выходную) битовую строку фиксированной длины, выполняемое определённым алгоритмом

2. ВВЕДЕНИЕ

2.2. Назначение документа

Данный документ содержит описание функциональных характеристик ПО Documino.Архив.

2.3. Краткое описание возможностей ПО Documino.Архив

ПО Documino.Архив обладает следующими основными возможностями:

- свой язык доступа к данным – XQL, выражения которого конвертируются в SQL запросы целевой СУБД;
- встроенная модель безопасности – поддержка пользователей, групп, ролей и прав доступа;
- версионность;
- хранение и доступ к контенту;
- возможность реализации на языке Java плагинов любой сложности, сделав их частью ПО Documino.Архив, расширяющие его функциональность;
- API для интеграции с внешними системами и обмена данными
- поддержка различных схем аутентификации.

2.4. Требование к эксплуатирующему персоналу

Для работы с платформой эксплуатирующий персонал (программист) должен обладать опытом:

- разработки на Java;
- работы с Jira, Maven, Git;
- создания Web-сервисов;
- работы с реляционными БД;
- работы с ОС Linux;

уверенными знаниями и владением:

- библиотеки классов Java 11 Standard Edition основных алгоритмов и шаблонов проектирования ПО;
- технического английского языка (чтение профессиональной литературы);
- технологий Web-разработки: HTML, JavaScript, CSS, JQuery.

3. ТЕХНИЧЕСКАЯ АРХИТЕКТУРА ПО DOCUMINO.АРХИВ

В основу архитектуры ПО Documino.Архив заложены следующие принципы:

- использование свободного программного обеспечения (СПО);
- минимальные возможности ядра, расширяющиеся за счёт плагинов;
- максимальное быстродействие между бизнес – логикой и СУБД, минимум логический архитектурных слоёв, что позволит работать с контент-ориентированными системами значительно быстрее аналогичных систем;
- безопасность – обеспечивается за счёт встроенной поддержки шифрования контента и различных схем аутентификации;
- гибкость – архитектура, основанная на плагинах, позволяет использовать любые движки полнотекстового поиска, бизнес-процессов и т.п.;
- универсальность – поддержка любой реляционной СУБД (на текущий момент реализована поддержка Postgres Pro).

3.2. XQL

XQL - eXtended Query Language - язык доступа к данным в ПО Documino.Архив. Представляет собой SQL-подобный язык, посредством которого клиентский код взаимодействует с платформой.

В ПО Documino.Архив используется LL синтаксический анализатор.

Результатом выполнения XQL-запроса всегда является коллекция - IDmCollection.

Набор полей в коллекции определяется типов запроса.

3.2.1. DDL

Выполнением DDL-запроса является коллекция, состоящая из одного поля result, логического типа.

3.2.2. DML

SELECT. Состав и тип данных коллекции select-запроса определяется самим запросом и колонками, указанными в выборке.

CREATE OBJECT. Возвращается коллекция, состоящая из одного поля result, строкового типа, который содержит идентификатор созданного объекта.

UPDATE OBJECTS. Возвращается коллекция, состоящая из одного поля result, целого типа, который содержит количество изменённых объектов.

DELETE OBJECTS. Возвращается коллекция, состоящая из одного поля result, целого типа, который содержит количество удалённых объектов.

GRANT PERMIT. Возвращается коллекция, состоящая из одного поля result, логического типа.

EXECUTE PLUGIN. Возвращается коллекция, состоящая из одного поля result, строкового типа, который содержит результат выполнения метода плагина.

Остальные. Возвращается коллекция, состоящая из одного поля result, логического типа.

3.2.3. Грамматика

Ниже описана синтаксическая структура языка.

statement

- ddl
- dml

3.2.3.1. ddl

- create_type
- alter_type
- drop_type

create_type

CREATE TYPE type_name '(' attribute_definition_list? ')'

type_name

WORD

attribute_definition_list

attribute_def (',' attribute_def)*

attribute_def

attribute_name attribute_data_type REPEATING? attribute_constraints?

attribute_constraints

'(' attribute_constraint (COMMA attribute_constraint)* ')'

attribute_constraint

NOT NULL

DEFAULT '=' default_value

READONLY

UNIQUE

REFERENCES type_name '(' attribute_name ')' (ON DELETE action)? (ON UPDATE action)?

action

CASCADE

RESTRICT

NO_ACTION

default_value

value

attribute_name

WORD

attribute_data_type

BOOLEAN

DOUBLE

INT

TIME

STRING '(' INT ')'

HASH '(' algorithm ',' INT ')'

CONTENT

algorithm

WORD

drop_type

DROP TYPE type_name

alter_type

ALTER TYPE type_name ADD attribute_definition_list

ALTER TYPE type_name DROP attribute_list

ALTER TYPE type_name SUPPORTS feature_list

ALTER TYPE type_name MODIFY attribute_name modify_data_type

ALTER TYPE type_name MODIFY attribute_name modify_constraint

attribute_list

attribute_name (COMMA attribute_name)*

feature_list

feature (',' feature)*

feature

WORD

modify_data_type

STRING '(' INT ')'

modify_constraint

set_constraint

drop_constraint

set_constraint

SET attribute_constraint

drop_constraint

DROP DEFAULT

DROP READONLY

DROP NOT NULL

DROP UNIQUE

3.2.3.2. dml

- select_root
- update_objects
- delete_objects
- create_object
- create_acl
- alter_group
- grant_permit
- execute_plugin
- create_trigger

select_root

parentheses_select

select

parentheses_select

' (' select_root ') '

select

select2 orderby_clause? limit_clause? offset_clause?

select2

parentheses_select2

select_union

parentheses_select2

' (' select2 ') '

select_union

single_select (UNION single_select)*

single_select

parentheses_single_select

SELECT DISTINCT? select_list FROM table_references where_clause? groupby_clause? having_clause? limit_clause? offset_clause?

parentheses_single_select

' (' single_select ') '

select_list

displayed_column (',' displayed_column)*

' * '

displayed_column

column_spec (alias)?

operand (alias)?

column_spec

(table_alias '.')? attribute_name ('[' INT ']')?

table_alias

WORD

alias

AS WORD

table_references

table_reference (',' table_reference)*

table_reference

type_name ('(' ALL ')')? (table_alias)?

groupby_clause

GROUP BY groupby_item (',' groupby_item)*

groupby_item

column_spec

having_clause

HAVING expression

limit_clause

LIMIT INT

offset_clause

OFFSET INT

orderby_clause

ORDER BY orderby_item (',' orderby_item)*

orderby_item

column_spec (ASC | DESC)?

INT (ASC | DESC)?

where_clause

WHERE expression

expression

and_condition (OR and_condition)*

and_condition

condition (AND condition)*

condition

operand

operand NOT? IN '(' select_root ')'

operand NOT? IN '(' value_list ')'

operand IS NOT? NULL

operand NOT? LIKE operand

operand compare_operator operand

NOT '(' condition ')'

EXISTS '(' select_root ')'

value_list

value (',' value)*

operand

factor (('+' | '-') factor)?

factor

term (('*' | '/') term)?

term

value

column_spec

'(' expression ')'

function

param

compare_operator

'=' | '!=' | '>' | '<' | '>=' | '<='

value

string_value

numeric_value

time_value

boolean_value

file_value

text_value

url_value

NULL

string_value

' ANYTHING '

array

'[' elements? ']'

elements

value (',' value)*

file_value

FILE '(' STRING ')'

FILE '(' STRING ',' STRING ')'

text_value

TEXT '(' STRING ')'

TEXT '(' STRING ',' STRING ')'

url_value

URL '(' STRING ')'

URL '(' STRING ',' STRING ')'

numeric_value

('+' | '-')? INT

('+' | '-')? DOUBLE

time_value

DATE '(' time_string ',' format_string ')'

time_string

STRING

format_string

STRING

boolean_value

T | F

function

COUNT '(' expression ')'

COUNT '(' '*' ')'

SUM '(' expression ')'

AVG '(' expression ')'

MIN '(' expression ')'

MAX '(' expression ')'

param

?

update_objects

UPDATE type_name OBJECTS update_list where_clause?

update_list

update_value (update_value)*

update_value

SET attribute_name ('[' INT ']')? '=' rvalue

rvalue

value

'(' select')'

param

delete_objects

DELETE type_name OBJECTS where_clause?

create_object

CREATE type_name OBJECT update_list

alter_group

ALTER GROUP group_name ADD member_list

ALTER GROUP group_name DROP member_list

member_list

member_name (',' member_name)*

member_name

WORD

STRING

grant_permit

GRANT permit TO USER user_name ON id TYPE type_name

GRANT permit TO GROUP group_name ON id TYPE type_name

permit

1 | 2 | 3 | 4

user_name

WORD

group_name

WORD

STRING

execute_plugin

EXECUTE plugin_name '.' method_name '(' argument_list ')'

plugin_name

WORD

method_name

WORD

argument_list

argument (',' argument)*

argument

value

array

create_trigger

CREATE TRIGGER trigger_name ON type_name event_list '(' action_list ')'

trigger_name

WORD

event_list

event_name (',' event_name)*

event_name

ONINSERT

ONUPDATE

action_list

action_definition (',' action_definition)*

action_definition

copy_action

copy_action

COPY '(' copy_attribute_list ')' TO type_name '(' attribute_list ')' PK '(' attribute_name')'

copy_attribute_list

copy_attribute (',' copy_attribute)*

copy_attribute

attribute_name

string_value

3.3. Модель данных ПО Documino.Архив

Модель данных представлена в графическом виде в приложении (Приложение 1.
Схема Модели данных).

3.4. Обязательные атрибуты

Все типы ПО Documino.Архив (встроенные и кастомные) обязательно имеют следующие атрибуты:

Атрибут	Тип данных	Длина	Множ. знач.	READONLY	NOT NULL	DEFAULT	Описание	Ограничение	Комментарий
r_object_id	ID	16	Нет	Да	Да	FALSE	Идентификатор объекта	Уникальный	За исключением типов dm_type, dm_type_attribute и dm_type_feature
r_creator_name	STRING	64	Нет	Да	Да	FALSE	Создатель объекта	Внешний ключ на dm_user.dss_name	
r_creation_date	TIME	-	Нет	Да	Да	FALSE	Время создания объекта		
r_modifier_name	STRING	64	Нет	Нет	Нет	FALSE	Пользователь, изменивший объект последний раз	Внешний ключ на dm_user.dss_name	
r_modify_date	TIME	-	Нет	Нет	Нет	FALSE	Дата последнего изменения		

							объекта		

3.4.1. dm_type

Хранит все зарегистрированные типы платформы:

Атрибут	Тип данных	Длина	Множ. знач.	Редактирование	NOT NULL	DEFAULT	Описание
dss_name	STRING	50	Нет	Да	Да	NULL	Системное имя типа
dsb_immutable_type	BOOLEAN	-	Нет	Да	Да	FALSE	Можно ли изменять тип
dsb_immutable_object	BOOLEAN	-	Нет	Да	Да	FALSE	Можно ли изменять объекты типа
r_creator_name	STRING	64	Нет	Да	Да	FALSE	Создатель объекта
r_creation_date	TIME	-	Нет	Да	Да	FALSE	Время создания объекта
r_modifier_name	STRING	64	Нет	Нет	Нет	FALSE	Пользователь, изменивший объект последний раз
r_modify_date	TIME	-	Нет	Нет	Нет	FALSE	Дата последнего изменения объекта

3.4.2. dm_type_attribute

Хранит информацию об атрибутах зарегистрированных типов:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
dss_type_name	STRING	50	Нет	Системное имя типа
dss_attr_name	STRING	50	Нет	Системное имя атрибута
dsi_attr_type	INT	-	Нет	Тип данных атрибута
dsi_attr_length	INT	-	Нет	Длина атрибута
dsb_attr_repeating	BOOLEAN	-	Нет	Множественный атрибут или нет
dss_info	STRING	50	Нет	Дополнительная информация об атрибуте
dsb_readonly	BOOLEAN	-	Нет	Атрибут только для чтения
dsb_not_null	BOOLEAN	-	Нет	Атрибут не может быть NULL
dss_default_value	STRING	50	Нет	Значение атрибута по умолчанию
r_creator_name	STRING	64	Нет	Создатель атрибута
r_creation_date	TIME	-	Нет	Время создания атрибута
r_modifier_name	STRING	64	Нет	Пользователь, изменивший атрибут последний раз
r_modify_date	TIME	-	Нет	Время последнего изменения атрибута

3.4.3. dm_type_feature

Хранит информацию об особенностях типов:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
dss_type_name	STRING	50	Нет	Системное имя типа
dss_feature_name	STRING	50	Нет	Имя особенности
r_creator_name	STRING	64	Нет	Создатель особенности
r_creation_date	TIME	-	Нет	Время создания особенности
r_modifier_name	STRING	64	Нет	Пользователь, изменивший особенность последний раз
r_modify_date	TIME	-	Нет	Время последнего изменения особенности

3.4.4. dm_folder

Представляет собой папку:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор объекта
dss_name	STRING	64	Нет	Имя папки
dsid_folder	ID	16	Нет	Идентификатор родительской папки
r_creator_name	STRING	64	Нет	Создатель папки
r_creation_date	TIME	-	Нет	Время создания папки
r_modifier_name	STRING	64	Нет	Пользователь, последний раз изменивший папку
r_modify_date	TIME	-	Нет	Время последнего изменения папки

3.4.5. dm_user

Учётная запись:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор объекта
dss_name	STRING	64	Нет	Логин пользователя
dss_password	HASH	512	Нет	Хэш пароля пользователя
dss_last_name	STRING	128	Нет	Фамилия
dss_first_name	STRING	128	Нет	Имя
dss_middle_name	STRING	128	Нет	Отчество
dss_email	STRING	50	Нет	Почтовый адрес
dsi_state	INT	-	Нет	Состояние пользователя

dsi_authentication	INT	-	Нет	Схема аутентификации
dsid_folder	ID	16	Нет	Идентификатор папки пользователя - личный ящик
r_creator_name	STRING	64	Нет	Создатель пользователя
r_creation_date	TIME	-	Нет	Время создания
r_modifier_name	STRING	64	Нет	Пользователь, последний раз изменивший этого пользователя
r_modify_date	TIME	-	Нет	Время последнего изменения

3.4.6. dm_group

Группа пользователей:

Атрибут	Тип данных	Длина	Множ. знач.	
r_object_id	ID	16	Нет	Идентификатор
dss_name	STRING	64	Нет	Системное имя группы
r_creator_name	STRING	64	Нет	Создатель группы
r_creation_date	TIME	-	Нет	Время создания группы
r_modifier_name	STRING	64	Нет	пользователь, изменявший группу
r_modify_date	TIME	-	Нет	Время последнего изменения группы

3.4.7. dm_group_users

Пользователи группы:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор
dss_group_name	STRING	64	Нет	Системное имя группы
dss_user_name	STRING	64	Нет	Системное имя пользователя
r_creator_name	STRING	64	Нет	Кто включил пользователя в группу
r_creation_date	TIME	-	Нет	Время включения пользователя в группу
r_modifier_name	STRING	64	Нет	
r_modify_date	TIME	-	Нет	

3.4.8. dm_acl

Набор прав доступа:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор
dss_name	STRING	32	Нет	Системное имя
dsb_immutable	BOOLEAN	-	Нет	Изменяемый набор прав доступа или нет
r_creator_name	STRING	64	Нет	Создатель
r_creation_date	TIME	-	Нет	Время создания
r_modifier_name	STRING	64	Нет	Изменивший пользователь
r_modify_date	TIME	-	Нет	Время последнего изменения

3.4.9. dm_user_permit

Права доступа пользователя:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор
dss_acl_name	STRING	32	Нет	Имя набора прав доступа
dss_accessor_name	STRING	64	Нет	Имя пользователя
dsi_permit	INT	-	Нет	Права
r_creator_name	STRING	64	Нет	Создатель
r_creation_date	TIME	-	Нет	Время создания
r_modifier_name	STRING	64	Нет	Изменивший пользователь
r_modify_date	TIME	-	Нет	Время изменения

3.4.10. dm_group_permit

Права доступа группы:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор
dss_acl_name	STRING	32	Нет	Имя набора прав доступа
dss_accessor_name	STRING	64	Нет	Имя группы
dsi_permit	INT	-	Нет	Права
r_creator_name	STRING	64	Нет	Создатель
r_creation_date	TIME	-	Нет	Время создания
r_modifier_name	STRING	64	Нет	Изменивший пользователь
r_modify_date	TIME	-	Нет	Время изменения

3.4.11. dm_content

Содержимое:

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор
r_mime_type	STRING	16	Нет	MIME-type
r_content_size	INT	-	Нет	Размер содержимого

r_creator_name	STRING	64	Нет	Создатель
r_creation_date	TIME	-	Нет	Время создания
r_modifier_name	STRING	64	Нет	Изменивший пользователь
r_modify_date	TIME	-	Нет	Время изменения

3.4.12. dm_plugin

Загружаемый плагин**:**

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор
dss_name	STRING	64	Нет	Имя
dss_implementation	STRING	64	Нет	Реализация - полное имя класса, реализующего функциональность плагина
dsc_jar	CONTENT	16	Нет	JAR-файл, содержащий класс-реализацию и все зависимости
r_creator_name	STRING	64	Нет	Создатель
r_creation_date	TIME	-	Нет	Время создания
r_modifier_name	STRING	64	Нет	Изменивший пользователь
r_modify_date	TIME	-	Нет	Время изменения

3.4.13. dm_property

Свойство**:**

Атрибут	Тип данных	Длина	Множ. знач.	Описание
r_object_id	ID	16	Нет	Идентификатор
dss_name	STRING	64	Нет	Имя
dss_implementation	STRING	64	Нет	Реализация - полное имя класса, реализующего функциональность плагина
dsc_jar	CONTENT	16	Нет	JAR-файл, содержащий класс-реализацию и все зависимости
r_creator_name	STRING	64	Нет	Создатель
r_creation_date	TIME	-	Нет	Время создания
r_modifier_name	STRING	64	Нет	Изменивший пользователь
r_modify_date	TIME	-	Нет	Время изменения

3.5. API

API ПО представляет собой небольшой набор интерфейсов, позволяющих:

- выполнять запросы XQL;
- управлять транзакциями;
- вызывать методы загружаемых плагинов.

На текущий момент сессия ПО Documino.Архив является тонкой обёрткой вокруг jdbc-соединения с СУБД.

Интерфейсы ПО Documino.Архив API находятся в пакете ru.idmt.documino.client.api (jar-файл documino-api). Также понадобятся классы пакета ru.idmt.documino.client.commons (jar-файл documino-commons). Входной точкой для работы с ПО, является интерфейс клиента ПО Documino.Архив:

```
package ru.idmt.documino.client.api;
 import ru.idmt.documino.client.api.session.IDmResourceManager;
import ru.idmt.documino.client.api.session.IDmSession;
import ru.idmt.documino.client.api.session.IDmSessionManager;
import ru.idmt.documino.client.api.util.DmException;
import ru.idmt.documino.client.api.util.IDmLoginInfo;
public interface IDmClient {
   String RESOURCE_IO2 = "io2";
   void authenticate(IDmLoginInfo info) throws DmException;
    IDmSessionManager newSessionManager(IDmLoginInfo info, int size) throws DmException;
    IDmSession newSession(IDmLoginInfo info)  throws DmException;
    void putResourceManager(String name, IDmResourceManager resourceManager);
    void removeResourceManager(String name);
    interface IDmListener {
      void onExec(String sql);
   }
}
```

Для полноценной работы с Documino необходимо создать экземпляр IDmClient, который специфичен для различных СУБД. Вся дальнейшая работа идёт с корректно установленным объектом клиента.

3.5.1. Работа с ресурсами

Ресурсы - это объекты, реализующие интерфейс ru.idmt.documino.client.api.session.IDmResource, время жизни которых совпадает с временем жизни сессии.

3.5.2. Логин-пароль

```
package ru.idmt.documino.client.example;

import ru.idmt.documino.client.api.IDmClient;
import ru.idmt.documino.client.api.session.IDmSession;
import ru.idmt.documino.client.api.util.DmException;
import ru.idmt.documino.client.commons.session.DmLoginInfo;
```

```
IDmClient client = ...;

        IDmSession session = null;
        try {
           session = client.newSession(new DmLoginInfo("user1", "1234"));

           //работа с платформой
        } finally {
           if(session != null) {
              session.disconnect();
           }
        }
```

Как обычно, после работы с системой - необходимо закрыть соединение с СУБД, используя метод disconnect().

3.5.3. Выполнение запросов

В Documino существует только один способ работы с данными - использование языка XQL. Для выполнения запросов необходимо аутентифицироваться в системе и иметь объект сессии IDmSession.

```
package ru.idmt.documino.client.example;

import ru.idmt.documino.client.api.IDmClient;
import ru.idmt.documino.client.api.query.IDmCollection;
import ru.idmt.documino.client.api.query.IDmData;
import ru.idmt.documino.client.api.query.IDmQuery;
import ru.idmt.documino.client.api.session.IDmSession;
import ru.idmt.documino.client.api.util.DmException;
import ru.idmt.documino.client.commons.session.DmLoginInfo;
 String xql = "SELECT COUNT(r_object_id) AS c FROM dm_user";
IDmQuery query = session.newQuery(xql);
IDmCollection collection = null;
try {
   collection = query.execute();
   if(collection.next()) {
      IDmData dmData = collection.get();
      System.out.println(dmData.getInt("c"));
   }
} finally {
   if (collection != null) {
      collection.close();
```

```
    }
}
```

3.5.4. Типы данных

ПО Documino.Архив поддерживает следующие типы данных:

- BOOLEAN - логический тип;
- INTEGER - целое число;
- STRING - строка с ограничением по длину;
- TIME - время;
- DOUBLE - число с плавающей точкой;
- CONTENT - контент, файл;
- HASH - хэш. Представляет собой атрибут строкового типа, но при записи значения в объект, значение хэшируется в соответствии с алгоритмом, указанным в определении типа атрибута.

Для работы с указанными типами существует интерфейс IDmData, который является абстракцией одной строки результата запроса и предоставляет соответствующие методы:

```
package ru.idmt.documino.client.api.query;
import ru.idmt.documino.client.api.util.DmException;
import java.util.Date;
public interface IDmData {
    int DM_BOOLEAN = 0;
    int DM_INTEGER = 1;
    int DM_STRING = 2;
    int DM_ID = 3;
    int DM_TIME = 4;
    int DM_DOUBLE = 5;
    int DM_CONTENT = 6;
    int DM_HASH = 7;
    int DM_UNDEFINED = 8;
    boolean getBoolean(String name) throws DmException;
    int getInt(String name) throws DmException;
    String getString(String name) throws DmException;
    IDmId getId(String name) throws DmException;
    Date getTime(String name) throws DmException;
    double getDouble(String name) throws DmException;
    IDmContent getContent(String name) throws DmException;
    IDmArray getArray(String name) throws DmException;
    IDmValue getValue(String name) throws DmException;
}
```

3.5.5. Работа с контентом

Если атрибут имеет контентный тип, то для получения контента, необходимо использовать метод getContent интерфейса IDmData. Данный метод возвращает объект типа IDmContent:

```
package ru.idmt.documino.client.api.query;

import java.io.IOException;
import java.io.InputStream;

public interface IDmContent {
   InputStream getInput() throws IOException;
}
```

Метод getInput() возвращает входной поток байтов, который позволяет прочитать содержимое.

Например, необходимо выкачать во временную директорию весь контент документов типа dm_document:

```
package ru.idmt.documino.client.commons;
import ru.idmt.documino.client.api.IDmClient;
import ru.idmt.documino.client.api.query.IDmCollection;
import ru.idmt.documino.client.api.query.IDmData;
import ru.idmt.documino.client.api.query.IDmQuery;
import ru.idmt.documino.client.api.session.IDmSession;
import ru.idmt.documino.client.api.util.DmException;
import ru.idmt.documino.client.commons.session.DmLoginInfo;
import ru.idmt.documino.client.commons.util.DmIO;
import java.io.File;
import java.io.IOException;
String xql = "SELECT dsc_file FROM dm_document";
IDmQuery query = session.newQuery(xql);
IDmCollection collection = null;
try {
   collection = query.execute(session);
   if(collection.next()) {
      IDmData dmData = collection.get();
      IDmContent content = dmData.getContent("dsc_file");
      File file = File.createTempFile("dm_document", ".bin");
      DmIO.copy(content, new FileOutputStream(file));
   }
} finally {
   if (collection != null) {
```

```
            collection.close();
        }
    }
```

В одном типе может быть сколько угодно атрибутов контентного типа. Создание таких атрибутов и их заполнение осуществляется с помощью языка XQL.

3.6. Версионность

Версионность - это возможность создавать версии карточки объекта. В этом случае объект представлен не одной карточкой, а деревом, узлы которого являются версиями данного объекта (карточками).

3.6.1. Включение версионности

ALTER TYPE dm_document SUPPORTS VERSIONS

При включении функции VERSIONS для типа происходят следующие действия:

- проверяется, что данный тип существует. В противном случае возникает ошибка;
- проверяется, что данный тип является изменяемым типом. В противном случае возникает ошибка;
- проверяется, что для данного типа функция VERSIONS не включена. В противном случае происходит ошибка;
- в тип добавляется атрибут i_chronicle_id STRING(16) для хранения идентификатора текущей версии;
- в тип добавляется атрибут r_version_label STRING(512) - уникальная в дереве версий метка;
- в тип добавляется атрибут r_is_current BOOLEAN - признак текущей версии в дереве объектов;
- в dm_type_feature сохраняется информация о том, что функция включена для данного типа;
- для всех существующих объектов данного типа атрибут i_chronicle_id заполняется значением r_object_id;
- для всех существующих объектов данного типа атрибут r_version_label заполняется значением '0.0.0'.

3.6.2. Создание версии

Для создания новой текущей версии, используется выражение:
CREATE dm_document VERSION ('1.0.0') FROM '0001112345678012' SET
При создании новой текущей версии обязательно проставление атрибутов r_version_label. Остальные атрибуты версии можно менять произвольно, используя оператор SET.

3.6.3. Выборка данных

При выборке объектов версионируемого типа в результаты выборки по умолчанию попадают только текущие версии объектов (то есть те, для которых выполняется условие r_object_id = i_current_id).

Для того чтобы увидеть все версии объектов (убрать условие r_object_id = i_current_id), необходимо использовать специальную версию SELECT-выражения:

SELECT * FROM dm_document (ALL).

3.6.4. Изменение данных

Для изменения текущей версии объекта используется обычное UPDATE-выражение. Не текущие версии объекта изменять нельзя.

3.6.5. Удаление данных

Для удаления текущей версии объекта используется обычное DELETE-выражение. Не текущие версии объекта удалить нельзя.

3.6.6. Принцип работы

В основе механизма лежит добавление дополнительного условия, связанного оставляющего по умолчанию лишь текущие версии, при конвертации XQL-выражения в SQL-запрос целевой СУБД.

Примеры

Выражение:
*SELECT * FROM ddt_document* превратится в *SELECT * FROM ddt_document WHERE (r_is_current = TRUE).*
Выражение:
*UPDATE ddt_document OBJECTS SET dss_name = 'TES*T' превратится в *UPDATE ddt_document SET dss_name = 'TEST' WHERE (r_is_current = TRUE).*
Выражение:
DELETE ddt_document OBJECTS WHERE dss_name = 'TEST' превратится в *DELETE FROM ddt_document WHERE (r_is_current = TRUE).*
Для того чтобы увидеть не только текущие версии объектов, используется следующий синтаксис:
*SELECT * FROM ddt_document (ALL*) В данном случае sql-запрос будет выглядеть как *SELECT * FROM ddt_document.*

Аналогичного синтаксиса для UPDATE и DELETE-выражений не существует.

3.6.7. Заполнение атрибутов при создании версии

Версия создаётся XQL-выражением, типа:

CREATE ddt_document VERSION('1.0.0') FROM '0001112345678012' SET dss_name = 'TEST',

где 0001112345678012 - идентификатор текущей версии объекта.

В результате этого выражения:

- создается копия объекта '0001112345678012' (тип ddt_document);
- для атрибута r_version_label созданной копии устанавливается значение '1.0.0';
- для атрибута i_chronicle_id созданной копии устанавливается значение атрибута i_chronicle_id объекта '0001112345678012';
- для атрибута r_is_current созданной копии устанавливается значение TRUE;
- для атрибута r_is_current объекта '0001112345678012' устанавливается значение FALSE;
- для остальных атрибутов (типа dss_name) значения переносятся из выражения CREATE-VERSION.

После этого созданная копия становится текущей версией объекта.

3.6.8. Версионность и ссылочная целостность

При установке связи между объектами, в зависимости от задачи, можно устанавливать связь на r_object_id (если требуется связь с конкретной версией) или на i_current_id (если требуется ссылка на текущую версию).

3.7. Права доступа

Права доступа - это одна из функций ПО Documino.Архив, которая позволяет ограничивать доступ к объектам типа в соответствии с назначенными на данный объект правами.

3.7.1. Включение

ALTER TYPE dm_document SUPPORTS ACL

- при включении функции ACL для типа происходят следующие действия:
- проверяется, что данный тип существует. В противном случае возникает ошибка;
- проверяется, что данный тип является изменяемым типом. В противном случае возникает ошибка;
- проверяется, что для данного типа функция ACL не включена. В противном случае происходит ошибка;

- в тип добавляется атрибут i_owner_name STRING(64) для хранения владельца объекта
- в тип добавляется атрибут i_acl_name STRING(64), который ссылается на атрибут dss_name типа dm_acl;
- в dm_type_feature сохраняется информация о том, что функция включена для данного типа.

3.7.2. Работа с данными

3.7.2.1. Права доступа

ПО Documino.Архив поддерживает следующие права доступа:

- NONE = 1 - нет прав на объект;
- READ = 2 - можно читать атрибуты объекта;
- WRITE = 3 - можно изменять значения атрибутов объекта;
- DELETE = 4 - объект можно удалить.

3.7.2.2. Раздача прав

Для изменения прав доступа на объект существует конструкция grant_permit.

Для того чтобы дать пользователю u1 право на чтение (READ) объекта типа ddt_document с идентификатором '00000000000000fj', нужно выполнить XQL-запрос:

GRANT 2 TO USER u1 ON '00000000000000fj' TYPE ddt_document

Аналогично, чтобы дать группе g1 право на запись (WRITE) объекта типа ddt_document с идентификатором '00000000000000fj', нужно выполнить следующий XQL-запрос:

GRANT 3 TO GROUP g1 ON '00000000000000fj' TYPE ddt_document

Для назначения прав доступа на объект всем пользователям системы, нужно использовать имя системного пользователя dm_world:

GRANT 2 TO USER dm_world ON '00000000000000fj' TYPE ddt_document

После выполнения этого запроса, объект ddt_document[00000000000000fj] будет доступен на чтение всем пользователям системы.

3.7.2.3. Выборка данных

Для выборки используется стандартное SELECT-выражение, никакого специального синтаксиса при этом не используется, например:

SELECT * FROM ddt_document

Если тип ddt_document поддерживает ACL, то выборка будет производиться с фильтрацией по правам доступа.

Объект попадает в выборку, если:

- для соединения с СУБД используется специальный экземпляр IDmClient, который не ограничивает объекты по правам доступа;

или

- текущий пользователь является владельцем объекта;

или

- текущий пользователь входит в группу, которая является владельцем этого объекта;

или

- пользователю на этот объект назначено право >= READ;

или

- пользователь входит в группу, которой на этот объект назначено право >= READ;

или

- на данный объект пользователю dm_world назначено право >= READ.

3.7.2.4. Изменение данных

Для изменения используется UPDATE-выражение, никакого специального синтаксиса при этом не используется, например:

UPDATE ddt_document OBJECTS SET dss_name = 'TEST'

Если тип ddt_document поддерживает ACL, то изменение затронет объекты, если:

Для соединения с СУБД используется специальный экземпляр IDmClient, который не ограничивает объекты по правам доступа;

или

- текущий пользователь является владельцем объекта;

или

- текущий пользователь входит в группу, которая является владельцем этого объекта;

или

- пользователю на этот объект назначено право >= WRITE;

или

- пользователь входит в группу, которой на этот объект назначено право >= WRITE;

или

- на данный объект пользователю dm_world назначено право >= WRITE.

3.7.2.5. Удаление данных

Для удаления используется DELETE-выражение, никакого специального синтаксиса при этом не используется, например:

DELETE ddt_document OBJECTS

Если тип ddt_document поддерживает ACL, то объект будет удалён, если:

- для соединения с СУБД используется специальный экземпляр IDmClient, который не ограничивает объекты по правам доступа;

или

- текущий пользователь является владельцем объекта;

или

- текущий пользователь входит в группу, которая является владельцем этого объекта;

или

- пользователю на этот объект назначено право >= DELETE;

или

- пользователь входит в группу, которой на этот объект назначено право >= DELETE;

или

- на данный объект пользователю dm_world назначено право >= DELETE.

3.7.3. Принцип работы

В основе механизма лежит добавление дополнительного условия, связанного с правами доступа, при конвертации XQL-выражения в SQL-запрос целевой СУБД.

Например, выражение:

*SELECT * FROM ddt_document*

превратится в

*SELECT *FROM ddt_document*

WHERE (((i_owner_name = 'u1' OR i_owner_name IN (SELECT dss_group_name

FROM dm_group_users

WHERE dss_user_name = 'u1') OR EXISTS(SELECT 1

FROM dm_acl a, dm_user_permit p

WHERE i_acl_name = a.dss_name AND

p.dss_acl_name = a.dss_name

AND p.dss_accessor_name IN

('u1', 'dm_world') AND

p.dsi_permit >= 2) OR

EXISTS(SELECT 1

FROM dm_acl a, dm_group_permit p

WHERE i_acl_name = a.dss_name AND p.dss_acl_name = a.dss_name AND

p.dss_accessor_name IN (SELECT dss_group_name

FROM dm_group_users

WHERE dss_user_name = 'u1') AND p.dsi_permit >= 2)))),

где u1 - имя пользователя, выполняющего запрос.

Выражение:

UPDATE ddt_document OBJECTS SET dss_name = 'TEST'

превратится в

UPDATE ddt_document SET dss_name = 'TEST'

WHERE (((i_owner_name = 'u1' OR i_owner_name IN (SELECT dss_group_name

FROM dm_group_users

WHERE dss_user_name = 'u1') OR EXISTS(SELECT 1

FROM dm_acl a, dm_user_permit p

WHERE i_acl_name = a.dss_name AND

p.dss_acl_name = a.dss_name

AND p.dss_accessor_name IN

('u1', 'dm_world') AND

p.dsi_permit >= 3) OR

EXISTS(SELECT 1

FROM dm_acl a, dm_group_permit p

WHERE i_acl_name = a.dss_name AND p.dss_acl_name = a.dss_name AND

p.dss_accessor_name IN (SELECT dss_group_name

FROM dm_group_users

WHERE dss_user_name = 'u1') AND p.dsi_permit >= 3))))

и, наконец, выражение:

DELETE ddt_document OBJECTS WHERE dss_name = 'TEST'

превратится в

DELETE FROM ddt_document

WHERE (((i_owner_name = 'u1' OR i_owner_name IN (SELECT dss_group_name

FROM dm_group_users

WHERE dss_user_name = 'u1') OR EXISTS(SELECT 1

FROM dm_acl a, dm_user_permit p

WHERE i_acl_name = a.dss_name AND

p.dss_acl_name = a.dss_name

AND p.dss_accessor_name IN

('u1', 'dm_world') AND

p.dsi_permit >= 4) OR

EXISTS(SELECT 1

FROM dm_acl a, dm_group_permit p

WHERE i_acl_name = a.dss_name AND p.dss_acl_name = a.dss_name AND

p.dss_accessor_name IN (SELECT dss_group_name

FROM dm_group_users

WHERE dss_user_name = 'u1') AND p.dsi_permit >= 4))) AND

(((dss_name = 'TEST'))))

Видно, что фильтрация используется с использованием системных типов dm_acl, dm_user_permit, dm_group_permit. Объекты типа dm_acl имеют имя (dss_name) и признак неизменяемости (dsb_immutable).

Объекты с поддержкой прав доступа ссылаются на ACL через атрибут i_acl_name, в котором хранится имя объекта dm_acl. Непосредственно права пользователей хранятся в объекте dm_user_permit, который имеет атрибуты dss_acl_name (имя dm_acl), dss_accessor_name (имя пользователя) и dsi_permit (значение права доступа). Аналогично, права групп хранятся в типе dm_group_permit, который имеет атрибуты dss_acl_name (имя dm_acl), dss_accessor_name (имя группы) и dsi_permit (значение права доступа).

При выполнении операции GRANT происходит следующее:

1. **Если i_acl_name != null**

   1.1. Если dm_acl.dsb_immutable = TRUE - создаётся новый объект dm_acl (dss_name = 'dm_идентификатор', dsb_immutable = FALSE), объекты dm_user_permit и dm_group_permit копируются у старого объекта dm_acl и привязываются к новому. Имя нового объекта dm_acl прописывается в атрибут i_acl_name объекта. После этого для нового набора прав доступа создаётся новый объект dm_user_permit (dm_group_permit) если такому пользователю (или группе) ещё не раздавали права на этот объект или изменяется существующий dm_user_permit (dm_group_permit) если такому пользователю (или группе) уже раздавали права на этот объект.

   1.2. Если dm_acl.dsb_immutable = FALSE - для этого набора прав доступа создаётся новый объект dm_user_permit (dm_group_permit), если такому пользователю (или группе) еще не раздавали права на этот объект, или изменяется существующий dm_user_permit (dm_group_permit), если такому пользователю (или группе) уже раздавали права на этот объект.

   1.3. Если i_acl_name = null, то создаётся новый объект dm_acl (dss_name = 'dm_идентификатор', dsb_immutable = FALSE), объекты dm_user_permit и dm_group_permit копируются у старого объекта dm_acl и привязываются к новому. Имя нового объекта dm_acl прописывается в атрибут i_acl_name объекта. После этого для нового набора прав доступа создается новый объект dm_user_permit (dm_group_permit), если такому пользователю (или группе) ещё не раздавали права на этот объект, или изменяется существующий dm_user_permit (dm_group_permit), если такому пользователю (или группе) уже раздавали права на этот объект.

3.7.4. Заполнение атрибутов i_acl_name и i_owner_name при создании объекта

При создании объекта i_owner_name заполняется именем пользователя текущей сессии ПО Documino.Архив. i_acl_name заполняется значением атрибута по умолчанию. Если значение по умолчанию не установлено, то i_acl_name = null.

3.7.5. Именованные права доступа

Documino при выполнении операции GRANT всегда создаёт лишь изменяемые (dsb_immutable = FALSE) и неименованные (dss_name = 'dm_идентификатор') наборы прав доступа. Признак неизменности dsb_immutable в наборах прав доступа используется для полноценного использования именованных прав доступа.

Например, есть тип ddt_document, который поддерживает права доступа. Необходимо, чтобы все объекты этого типа при создании были доступны группе ddt_document_readers на чтение. В этом случае необходимо создать именованный набор прав доступа для этого:

```
CREATE dm_acl OBJECT
SET dss_name = 'acl_DEFAULT_DOCUMENT_READERS'
SET dsb_immutable = T;
CREATE dm_group_permit OBJECT
SET dss_acl_name = 'acl_DEFAULT_DOCUMENT_READERS'
SET dss_accessor_name = 'ddt_document_readers'
SET dsi_permit = 2;
```

Далее необходимо создать имя этого набора прав доступа значением по умолчанию для i_acl_name:

```
ALTER TYPE dm_method MODIFY i_acl_name SET DEFAULT =
'acl_DEFAULT_DOCUMENT_READERS'
```

Теперь объекты по умолчанию будут доступны группе ddt_document_readers. Если же выполнить для такого объекта операцию GRANT, то acl_DEFAULT_DOCUMENT_READERS останется неизменным, создастся системный набор прав доступа на основе acl_DEFAULT_DOCUMENT_READERS, в который вносятся изменения, требуемые операцией GRANT и имя которого сохранится в атрибуте i_acl_name.

3.7.6. Результат раздачи прав

Чтобы детально представить процесс раздачи прав, ниже описаны возможные результаты выполнения этой операции и условия, которые к ним приводят.

3.7.6.1. Раздача прав пользователю

GRANT 2 TO USER u1 ON '00000000000000fj' TYPE ddt_document

1. Тип существует.

   1.1. корректное значение прав (1-4):
   1.1.1. пользователь u1 существует:
   1.1.1.1 у текущего пользователя права WRITE на документ:
   1.1.1.1.1 у документа есть ACL:
   - изменяемая ACL;
   ✓ пользователь u1 присутствует в ACL;
   ▪ изменяется значение прав доступа в dm_user_permit для пользователя u1 ACL;
   ✓ пользователь u1 отсутствует в ACL;
   ▪ создаётся запись с нужными правами доступа в dm_user_permit для пользователя u1 ACL;
   - неизменяемая ACL (Клонируем ACL в изменяемую);
   ✓ пользователь u1 присутствует в ACL;
   ▪ изменяется значение прав доступа в dm_user_permit для пользователя u1 в склонированной ACL;
   ✓ пользователь u1 отсутствует в ACL;
   ▪ создаётся запись с нужными правами доступа в dm_user_permit для пользователя u1 в склонированной ACL;

   1.1.1.1.2. у документа НЕТ ACL:
   - создаётся ACL и сохраняется её имя в i_acl_name документа;
   - создаётся запись с нужными правами доступа в dm_user_permit для пользователя u1 в созданной ACL.

   1.1.1.2. у текущего пользователя НЕТ права WRITE на документ: **ОШИБКА;**
   1.1.2. пользователь u1 НЕ существует: **ОШИБКА;**
   1.2. некорректное значение прав (1-4): **ОШИБКА** (Например: GRANT 45 TO USER u1 ON '00000000000000fj' TYPE ddt_document);

2. Тип НЕ существует: **ОШИБКА.**

3.7.6.2. Раздача прав группе

GRANT 2 TO GROUP g1 ON '00000000000000fj' TYPE ddt_document

3. Тип существует.

1.1. корректное значение прав (1-4):
1.1.1. группа g1 существует:
1.1.1.1 у текущей группы права WRITE на документ:
1.1.1.1.1 у документа есть ACL:
- изменяемая ACL;
- ✓ группа g1 присутствует в ACL;
- ▪ изменяется значение прав доступа в dm_user_permit для группы g1 ACL;
- ✓ группа g1 отсутствует в ACL;
- ▪ создаётся запись с нужными правами доступа в dm_user_permit для группы g1 ACL;
- неизменяемая ACL (Клонируем ACL в изменяемую);
- ✓ группа g1 присутствует в ACL;
- ▪ изменяется значение прав доступа в dm_user_permit для группы g1 в склонированной ACL;
- ✓ группа g1 отсутствует в ACL;
- ▪ создаётся запись с нужными правами доступа в dm_user_permit для группы g1 в склонированной ACL;

1.1.1.1.2. у документа НЕТ ACL:
- создаётся ACL и сохраняется её имя в i_acl_name документа;
- создаётся запись с нужными правами доступа в dm_user_permit для группы g1 в созданной ACL.

1.1.1.2. у текущего пользователя НЕТ права WRITE на документ: **ОШИБКА;**
1.1.2. группа g1 НЕ существует: **ОШИБКА;**
1.2. некорректное значение прав (1-4): **ОШИБКА** (Например: GRANT 45 TO GROUP g1 ON '00000000000000fj' TYPE ddt_document);

4. Тип НЕ существует: **ОШИБКА.**

3.8. Плагины

ПО Documino.Архив -плагин - это завершённый программный модуль, загружаемый в систему в виду объекта dm_plugin и инкапсулирующий в себе некую логику.

Плагины - это основное средство расширения возможностей платформы.

3.8.1. Атрибуты

Ниже дано описание атрибутов типа dm_plugin.

3.8.1.1. dss_name

Уникальное ненулевое имя плагина. Используется для вызова плагина посредством XQL.

3.8.1.2. dss_implementation

Полное имя класса реализации плагина. Данный класс должен удовлетворять следующим требованиям:

- реализовать интерфейс ru.idmt.documino.client.api.plugin.IDmPlugin;
- иметь публичный конструктор без аргументов.

3.8.1.3. dsc_jar

Jar -файл, содержащий класс реализации плагина и все необходимые зависимости.

3.8.2. Создание плагина

Для начала необходимо создать интерфейс плагина. Данный интерфейс будет использоваться для доступа к методам плагина.

```
package ru.idmt.documino.plugin.example.api;
import ru.idmt.documino.client.api.plugin.IDmPlugin;
public interface IDmExample extends IDmPlugin {
    String echo(String value);
    String execute(String value);
    int execute(int val1, int val2);
}
Теперь создадим класс реализации:
package ru.idmt.documino.plugin.example.impl;
import ru.idmt.documino.plugin.example.api.IDmExample;
public class DmExampleImpl implements IDmExample {
    @Override
    public String echo(String value) {
        return value;
    }
    @Override
    public String execute(String value) {
        return value;
    }
    @Override
    public int execute(int val1, int val2) {
```

```
        return val1 + val2;
    }
}
```

Запаковав все необходимые для работы плагина классы в jar-файл (который содержит в себе класс реализации, класс интерфейса и класс IDmPlugin) , например, он лежит по пути /u01/temp/example-impl.jar, можно загрузить плагин в систему:

```
CREATE dm_plugin OBJECT
SET dss_name = 'example'
SET dss_implementation =
'ru.idmt.documino.plugin.example.impl.DmExampleImpl'
SET dsc_jar = FILE('/u01/temp/example-impl.jar')
```

Теперь плагин загружен в систему и готов к работе.

Если реализация плагина изменится, ее можно обновить в любой момент (без рестартов компонентов системы):

```
UPDATE dm_plugin OBJECTS
SET dsc_jar = FILE('/u01/temp/example-impl.jar')
WHERE dss_name = 'example'
```

3.8.3. Вызов плагина

С плагином можно работать двумя способами: программно и через XQL.

Для программного доступа к методам плагина нужно воспользоваться методом newPlugin(String name) объекта IDmSessionЭ, где name - это значение атрибута dss_name объекта dm_plugin в системе.

```
IDmSession session = ...

IDmExample example = session.newPlugin("example");

String value = example.echo("TEST")
```

Для работы с плагином посредством языка XQL, необходимо воспользоваться конструкцией XQL#execute_plugin.

EXECUTE example.execute(1, 2) вызовет метод

```
 @Override

    public int execute(IDmSession session, int val1, int val2) {
```

```
    return val1 + val2;
  }
```

А выражение:

EXECUTE example.execute('TEST') приведёт к вызову метода

```
 @Override
   public String execute(IDmSession session, String value) {
      return value;
   }
```

Первым аргументом метода всегда является сессия, в контексте которой вызывается плагин. Результат вызова метода через XQL всегда приводится к строковому типу.

Если метода с необходимой сигнатурой не найдено, то выполнение такого XQL-выражения завершится ошибкой java.lang.NoSuchMethodException

3.9. Аутентификация

Для корректной работы с СУБД, хранилищем содержимого и получения доступа к возможностям ПО Documino.Архив необходимо установить сессию.

При установлении сессии ПО Documino.Архив клиентская сторона должна пройти процедуру аутентификации. Информация о пользователе, прошедшем процедуру аутентификации, будет использоваться платформой в файлах журналирования, сохраняться на уровне СУБД в качестве создателя объектов, фиксироваться в СУБД при изменении данных и т.д.

Для поддержки механизмов аутентификации в ПО Documino.Архив используется справочник dm_user, который содержит информацию об учётных записях системы.

В общем случае запись в dm_user представляет собой сущность (не обязательно человек, это может быть сервис, сторонняя система или др.), которая может аутентифицироваться и установить сессию ПО Documino.Архив.

ПО Documino.Архив поддерживает 2 типа программных клиентов:

- Административный:

При использовании этого типа клиента аутентификация не осуществляется. Все действия в рамках этой сессии считаются произведёнными от имени системного пользователя master.

- Пользовательский:

Предварительные условия успешной пользовательской аутентификации:

- ✓ Учетная запись должна существовать в dm_user.
- ✓ Учетная запись должна быть активна (dsi_state == 0).

3.9.1. Аутентификация по логину и паролю

Если dsi_authentication = 0, то используется аутентификация по логину и паролю, хранящемуся в СУБД. Пароль в СУБД хранится в виде хэша.

Алгоритм хэширования хранится в справочнике dm_type_attribute. По умолчанию, для хэширования паролей учётных записей используется алгоритм SHA.

При аутентификации пароль, используемый клиентской стороной, хэшируется и сравнивается с хэшом пароля, который хранится в СУБД в атрибуте dss_password учётной записи. Если эти хэши одинаковы, аутентификация считается пройдённой успешно.

3.9.2. Доменная аутентификация

Если dsi_authentication = 1, то используется аутентификация с использованием AD. Параметры доступа к AD хранятся в типе dm_ldap_config. Поскольку конфигурация к AD может быть несколько, используется первая активная (dsb_active = TRUE) конфигурация.

Логин и его пароль используется для аутентификации в AD. Если аутентификация пройдена успешно, то пользователь считается успешно аутентифицированным.

3.10. Идентификатор ПО Documino.Архив

Идентификатор ПО Documino.Архив представляет собой строку из 16 символов. Каждый символ - один из следующего списка:

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

3.10.1. Назначение

Идентификатор служит первичным ключом всех объектов ПО Documino.Архив. При создании объектов платформа автоматически заполняет существующий системный атрибут r_object_id уникальным идентификатором.

3.10.2. Алгоритм формирования

Идентификаторы всех объектов, созданных средствами ПО Documino.Архив должны быть уникальны. Для того чтобы обеспечить эту уникальность - сервис, генерирующий идентификаторы должен каждый раз выдавать новое значение. По сути, идентификатор - это 16-разрядное число, записанное в позиционной системе отсчёта с основанием 62.

Сервис генерации идентификаторов может выдавать строки, представляющие собой последовательно увеличивающиеся числа, записанные в указанном формате. Ниже описаны значения символов, используемых при записи числа (слева - символ, справа - значение в десятичной системе счисления):

0 - 0; 1 - 1; 2 - 2; 3 - 3; 4 - 4; 5 - 5; 6 - 6;

7 - 7; 8 - 8; 9 - 9; a - 10; b - 11; c - 12; d - 13;

e - 14; f - 15; g - 16; h - 17; i - 18; j - 19; k - 20;

l - 21; m - 22; n - 23; o - 24; p - 25; q - 26; r - 27; s - 28; t - 29;

u - 30; v - 31; w - 32; x - 33; y - 34; z - 35; A - 36; B - 37;

C - 38; D - 39; E - 40; F - 41; G - 42; H - 43;

I - 44; J - 45; K - 46; L - 47; M - 48; N - 49; O - 50; P - 51; Q - 52;

R - 53; S - 54; T - 55; U - 56; V - 57; W - 58; X - 59; Y - 60; Z - 61;

Таким образом идентификаторы при последовательной генерации будут иметь следующие значения:

000000000000000

000000000000001

000000000000002

000000000000003

000000000000004

000000000000005

000000000000006

000000000000007

000000000000008

000000000000009

00000000000000a

00000000000000b

...

00000000000000x

00000000000000y

00000000000000z

00000000000000A

00000000000000B

00000000000000C

...

00000000000000X

00000000000000Y

00000000000000Z

000000000000010

000000000000011

000000000000012

000000000000013

...

00000000000001x

00000000000001y

00000000000001z

...

4. НЕШТАТНЫЕ СИТУАЦИИ

Для решения ошибок в работе ПО Documino.Архив необходимо обращаться в компанию «АйДи –Технологии управления» по адресу: mail@id-mt.ru.

ПРИЛОЖЕНИЕ 1.
СХЕМА МОДЕЛИ ДАННЫХ ПО DOCUMINO.АРХИВ