## Назначение подсистемы AuditLog

Подсистема AuditLog предназначена для хранения всех изменений доменных объектов с целью определения пользователя, который внес изменения в доменный объект, состав и дату этих изменений, а так же возможности восстановления доменного объекта на определенный момент времени.

### Способ реализации

Для хранения всех версий доменных объектов в базе данных создается структура хранения, аналогичная структуре для хранения самих доменных объектов с добавлением дополнительных полей.

На каждый тип доменных объектов создается набор таблиц с суффиксом «\_log» для базового типа и для всех дочерних типов. Состав атрибутов аналогичен основным таблицам, за исключением:

1. Добавляется поля для хранения идентификатора доменного объекта domain\_object\_id и domain\_object\_type.
2. Не строится внешний ключ на основные таблицы хранения доменных объектов. Это необходимо для того, чтобы была возможность сохранить аудит после удаления основного доменного объекта.
3. Добавляются поля для хранения дополнительной информации о версии:
   1. component – идентификатор компоненты из accessTocken
   2. ip\_address – IP адрес удаленной машины выполнившей изменения
   3. info – дополнительная произвольная информация (зарезервировано)
   4. operation – выполненное действие (1-создание, 2-изменение, 3-удаление)

Поле id таблицы хранения версий содержит уникальный идентификатор версии, который используется для однозначной идентификации версии в системе.

В конфигурацию добавляется раздел верхнего уровня <global-settings>, тэг может быть в xml только один раз. Внутри добавляется тэг <audit-log default-enable=”true|false”/> конфигурирующий поведение аудит лога по умолчанию.

Для детальной конфигурации AuditLog для конкретного типа в конфигурации доменных объектов добавляется атрибут audit-log в теге domain-object-type. Атрибут не обязательный. Если атрибут не указан то берется значение из глобальной конфигурации из тэга audit-log атрибута enable, если в глобальной конфигурации отсутствует тэг audit-log то значение принимается false.

Добавление записей в AuditLog производится внутри domainObjectDao бине и не доступно извне.

Для работы с аудитом создаются спринг бин AuditLogServiceDao и EJB AuditLogService:

**package** ru.intertrust.cm.core.dao.api;

**import** java.util.List;

**import** ru.intertrust.cm.core.business.api.dto.DomainObjectVersion;

**import** ru.intertrust.cm.core.business.api.dto.Id;

/\*\*

\* Интерфейс сервиса работы с Audit логом

\* **@author** larin

\*

\*/

**public** **interface** AuditLogServiceDao {

/\*\*

\* Получение всех версий доменного объекта

\* **@param** domainObjectId

\* **@return**

\*/

List<DomainObjectVersion> findAllVersions(Id domainObjectId);

/\*\*

\* Получение конкретной версии по известному идентификатору

\* **@param** versionId

\* **@return**

\*/

DomainObjectVersion findVersion(Id versionId);

/\*\*

\* Очистка аудита доменного объекта.

\* **@param** domainObjectId

\*/

**void** clean(Id domainObjectId);

}

**package** ru.intertrust.cm.core.business.impl;

**import** java.util.List;

**import** ru.intertrust.cm.core.business.api.dto.DomainObjectVersion;

**import** ru.intertrust.cm.core.business.api.dto.Id;

**import** ru.intertrust.cm.core.business.api.dto.VersionComparisonResult;

/\*\*

\* Интерфейс сервиса работы с Audit логом

\* **@author** larin

\*

\*/

**public** **interface** AuditService {

/\*\*

\* Получение всех версий доменного объекта

\* **@param** domainObjectId

\* **@return**

\*/

List<DomainObjectVersion> findAllVersions(Id domainObjectId);

/\*\*

\* Получение конкретной версии по известному идентификатору

\* **@param** versionId

\* **@return**

\*/

DomainObjectVersion findVersion(Id versionId);

/\*\*

\* Очистка аудита доменного объекта.

\* **@param** domainObjectId

\*/

**void** clean(Id domainObjectId);

/\*\*

\* Получение информации об изменениях между текущей версией доменного объекта и версией с переданным идентификатором

\* **@param** baseVersionId

\* **@return**

\*/

VersionComparisonResult compare(Id baseVersionId);

/\*\*

\* Получение информации об изменениях в двух разных версиях доменного объекта по известным идентификаторам версий

\* **@param** baseVersionId

\* **@param** comparedVersionId

\* **@return**

\*/

VersionComparisonResult compare(Id baseVersionId, Id comparedVersionId);

}

## Описание моделей, используемых при работе с AuditLog.

Класс, описывающий версию - DomainObjectVersion.

**package** ru.intertrust.cm.core.business.api.dto;

**import** java.util.Date;

/\*\*

\* Интерфейс версии доменного объекта

\* **@author** larin

\*

\*/

**public** **interface** DomainObjectVersion **extends** IdentifiableObject {

/\*\*

\* Идентификатор доменного объекта

\* **@return**

\*/

Id getDomainObjectId();

/\*\*

\* Получение дополнительной информации о версии (зарезервировано)

\* **@return**

\*/

String getVersionInfo();

/\*\*

\* Получение информации о компоненте, производившей изменения. Информация

\* берется из systemAccessToken

\* **@return**

\*/

String getComponent();

/\*\*

\* Получение IP адреса хоста, с которого выполнялась работа при выполнении

\* изменений

\* **@return**

\*/

String getIpAddress();

/\*\*

\* Идентификатор персоны (тип Person) выполнившей изменение

\* **@return**

\*/

Id getModifier();

/\*\*

\* Возвращает дату модификации данного доменного объекта

\*

\* **@return** дату модификации данного доменного объекта

\*/

Date getModifiedDate();

}

Класс описывающий разницу между двумя версиями VersionComparisonResult

**package** ru.intertrust.cm.core.business.api.dto;

**import** java.util.Date;

**import** java.util.List;

/\*\*

\* Иньерфейс описывающий разницу в версиях доменных объектов

\* **@author** larin

\*

\*/

**public** **interface** VersionComparisonResult {

/\*\*

\* Возвращает идентификатор версии, с которой производится сравнение

\* **@return**

\*/

Id getBaseVersionId();

/\*\*

\* Возвращает идентификатор версии, которую сравниваю с базовой

\* **@return**

\*/

Id getComparedVersionId();

/\*\*

\* Возвращает идентификатор доменного объекта

\* **@return**

\*/

Id getDomainObjectId();

/\*\*

\* Получение идентификатора персоны, выполнившей изменения. в случае если

\* изменения производились от имени системы то null

\* **@return**

\*/

Id getModifier();

/\*\*

\* Дата сохранения изменений

\* **@return**

\*/

Date getModifiedDate();

/\*\*

\* Получение дополнительной информации о версии (зарезервировано)

\* **@return**

\*/

String getVersionInfo();

/\*\*

\* Получение информации о компоненте, производившей изменения. Информация

\* берется из systemAccessToken

\* **@return**

\*/

String getComponent();

/\*\*

\* Получение IP адреса хоста, с которого выполнялась работа при выполнении

\* изменений

\* **@return**

\*/

String geyIpAddress();

/\*\*

\* Информация о изменившихся атрибутах

\* **@return**

\*/

List<FieldModification> getModifiedFields();

}

Класс описывающий изменение одного атрибута ModifiedAttribute

**package** ru.intertrust.cm.core.business.api.dto;

/\*\*

\* Информация о изменившемся поле доменного объекта

\* **@author** larin

\*

\*/

**public** **interface** FieldModification {

/\*\*

\* Получение имени атрибута

\* **@return**

\*/

String getName();

/\*\*

\* Получение значение атрибута базовой версии

\* **@return**

\*/

<T **extends** Value> T getBaseValue();

/\*\*

\* Получение значения атрибута сравниваемой версии

\* **@return**

\*/

<T **extends** Value> T getComparedValue();

}

## Необходимые доработки в ядре и других сервисах

1. Необходимо создать метод getModifier() в доменном объекте и соответствующее поле в базе данных. Поле возвращает идентификатор персоны, выполнившей изменение. В случае если изменение было выполнено системным процессом (джобом или workflow) в поле сохраняется null.
2. Не хватает информации о текущем пользователе внутри spring бинов или иных классов, где нет доступа к SessionContext. Необходимо продумать механизм создания RequestContext или ThreadContext с этой информацией.
3. Не хватает информации о адресе машины, инициирующей запрос к серверу внутри spring бинов или иных классов. Необходимо продумать механизм создания RequestContext или ThreadContext с этой информацией.