DEV, Инвентаризация, history

Инвентаризация: DEV: Model History

Как нам вести историю изменения моделей.

Первоначальная задумка

заводим таблицу, которая

- содержит те же поля что и исходная таблица
- master_id - ID в мастер таблице где хранится актуальная версия модели
- строковые поля, содержащие id ссылок через запятую для many-2-many realtions
- changed_at - время начала периода действия записи (unix epoch)
- changed_by - автор изменений (string login)
- changed_comment - комментарий к изменениям
- changed_attributes - список изменившихся атрибутов (кроме ['id','updated_at','updated_by','changed_attributes','updated_comment'])

после изменения модели добавляем запись в таблицу истории с новыми значениями и автором Возможно получится это залепить через Behaviour (некогда разбираться) Также очевидно, что изменение полей относящихся к many2many связям приводит к «изменению» связанных объектов. Почему в кавычках? Потому что в самой таблице модели изменений нет, но если посмотреть many2many атрибуты (из других таблиц), то изменения есть. Таким образом надо внести в класс истории модели

- функцию вытаскивания many2many связанных объектов
- список полей которые надо будет прописывать в журнал связанных объектов из инициатора изменений: udated_at,updated_by,updated_comment придется брать из журнала инициатора, т.к. связанный объект явно не изменялся
- функцию журналирования состояния с возможностью указания инициатора изменений

Кандидаты на аудит

- ~~Сервисы (важный узел, нужно видеть кто что меняет)~~
- ~~Документы - видеть кто косячит при заведении~~
- ~~Оборудование - косяки при заведении~~
- ~~Модели оборудования - косяки при заведении~~
- ~~ACL - чувствительные данные~~

Как добавить историю модели

Таблицы

В самой модели должны быть поля

- updated_at (timestamp()) - время обновления модели
- updated_by (string(32)) - логин обновившего (т.к. логин может скакать между пользователями, которые на самом деле сотрудники, а не пользователи)

```
$this->addColumnIfNotExist('aces','updated_at',$this->timestamp(),true);
$this->addColumnIfNotExist('aces','updated_by',$this->string(32),true);
```

Создаем таблицу для модели истории Там должны быть поля

- master_id (integer()) - ссылка на оригинальную модель **+ индекс**
- updated_at (timestamp()) - отметка о времени записи изменения **+ индекс**
- updated_by (string(32)) - логин изменившего **+ индекс**
- (опционально) updated_comment (string()) - можно комментировать каждое изменение
- changed_attributes (text()) - список полей изменившихся относительно прошлой записи (через запятую)

Все поля оригинальной модели нужно также добавить в том же виде (если не добавить, то их история не будет вестись) Также (если) нужно добавить поля для many-2-many и обратных ссылок в виде (text()) полей. Туда будут складываться ID объектов связанных с моделью. Например:

- users_ids (text())
- comps_ids (text())

```
$this->createTable('aces_history',[
    'id'=>$this->primaryKey(),
    'master_id'=>$this->integer(),
    'updated_at'=>$this->timestamp(),
    'updated_by'=>$this->string(32),
    'updated_comment'=>$this->string(),
    'changed_attributes'=>$this->text(),

    'comment'=>$this->string(),
    'notepad'=>$this->text(),
    'acls_id'=>$this->integer(),
    'users_ids'=>$this->text(),
    'comps_ids'=>$this->text(),
    'access_types_ids'=>$this->text(),
    'ips'=>$this->text(),
]);
$this->createIndex('aces_history-master_id','aces_history','master_id');
$this->createIndex('aces_history-updated_at','aces_history','updated_at');
$this->createIndex('aces_history-updated_by','aces_history','updated_by');
```

Модели

Исходная

Должна наследоваться от ArmsModel, тогда в процессе afterSave будет проверяться наличие

класса истории, и если он есть, то история изменений будет сохранена

Атрибуты Ссылки

надо, чтобы в оригинальной модели были доступны атрибуты с ID объектов с many-2-many и one-2-many обратными ссылками. Если это many-2-many ссылки, то они там уже объявлены через Behaviour. Для обратных ссылок one-2-many можно использовать тот же механизм, прописав эти поля в Behaviour many-2-many

```
/**
 * В списке поведений прикручиваем many-to-many ссылки
 * @return array
 */
public function behaviors()
{
    return [
        [
            'class' => \voskobovich\linker\LinkerBehavior::className(),
            'relations' => [
                'users_ids' => 'users',
                'comps_ids' => 'comps',
                'access_types_ids' => 'accessTypes',
            ]
        ]
    ];
}
```

Атрибут Name

Должен быть, если ссылки на этот объект есть в истории других объектов сойдет даже вычисляемый getName()