

[dev](#), Инвентаризация, model, fields

Инвентаризация: DEV: Model Fields

Описание атрибутов модели делаем в одной функции:

```

/** @var string как называется один экземпляр модели (для страницы Create ->
Новый объект) */
public static $title='Объект';

/** @var string как называется список моделей (для страницы Index) */
public static $titles='Объекты';

/** @var string надпись на кнопке создания нового объекта в списке */
public static $addButtonText='Добавить';

/** @var null|string подсказка для кнопки создания нового объекта */
public static $addButtonHint=null;

/** @var string Префикс для страницы Create (Новый $title) */
public static $newItemPrefix='Новый';

public $parentAttr='parentService'; //имя атрибута в котором у нас объект-
предок (родительское помещение/сервис/отдел и т.д.) - нужен для рекурсивных функций
обхода дерева
public function attributeData() {
    return [
        'model_id' => [
            //Атрибуты отображаются в следующих основных сценариях:
            // - form (заполнение) поиск значений:label,hint
            // - view (просмотр) поиск
значений:viewLabel->label,viewHint->indexHint-> ''
            // - grid (заголовок в таблице) поиск
значений:indexLabel->viewLabel->label,indexHint->viewHint-> '' v
            // - search (заголовок в таблице с фильтром) значение:
indexHint()+(searchHint-> '')
            // - api (формирование документации) поиск
значений:apiLabel->label,apiHint->hint
            //метка атрибута
            //позже транслируется в attributeLabels, используется в
отображении атрибута в формах
            'label'=>'Модель ПК',
            //позже транслируется в attributeHints, используется для пояснения
атрибута в формах
            'hint'=>'Модель системного блока / ноутбука.
            .'Если нужная модель отсутствует в списке, то нужно сначала
завести в ее в соотв. категории оборудования',
            //метка атрибута для GRID
            //позже транслируется в attributeIndexLabels, используется в

```

```
отображении атрибута в списках
    //рекомендуется использовать более короткое название атрибута чем
'label', т.к. должен умещаться в ширину колонки
    'indexLabel'=>'Модель ПК',
    //позже транслируется в attributeIndexHints, используется для
пояснения атрибута в списках
    //если отсутствует, то используется значение из hint
    //макрос {same} будет заменен на значение в hint
    'indexHint'=>'Модель системного блока / ноутбука.',
    //подсказка для поиска. добавляется к indexHint если в GRID включен
поиск
    'searchHint' => 'Имена модели можно перечислить через верт. черту
" | ",
    //метка атрибута для API документации
    //если отсутствует, то используется indexLabel
    'apiLabel' => 'Модель ПК',
    //подсказка атрибута для API документации
    //макрос {same} будет заменен на значение в hint
    //если отсутствует, то используется indexHint
    'apiHint' => 'Модель системного блока / ноутбука.',
    //тип Input этого атрибута в форме редактирования
    // - boolean - да / нет (чекбокс)
    // - toggle - это как boolean, но только для 0 и 1 есть свои
названия, напр сервис/услуга
    // - radios - это как toggle, только значений может быть больше
2
    // - list - это как radios, только значений прям много и
оформляется в виде dropdown
    // - ntext - textarea (простой текст без форматирования с
конвертацией nl -> <br> при рендре)
    // - text - текст с форматированием (какой формат прописывается в
параметрах)
    // - date - дата
    // - datetime - дата/время
    // - ips - список IP
    // - macs - список MAC
    // - urls - список URL
    // - link - одиночная ссылка
    // - number - число (ввод через обычный input/поиск числовой)
    // - string - обычный текст (по умолчанию)
    'type' => 'toggle',
    //список констант-значений атрибута (для toggle/radios/list)
    'fieldList' => ['Услуга', 'Сервис'],
    //чем заполнить селектор в форме, если значение не введено
    'placeholder' => 'Модель ПК не выбрана',
    //поле наследуемое (если не задано в этом объекте, то значение берется
из родителя)
    'is_inheritable'=>true,
    //атрибут только для чтения (updated_at, раскрытие ссылки) -> для
API документации
```

```

        'readOnly'=>true,
        //атрибут только для записи (пароль) -> для API документации
        'writeOnly'=>true,
        //при вызове функции absorb это поле нужно поглощать из переданного
объекта
        // - false - нет (по умолчанию для полей - "не обратных ссылок"
        // - 'isEmpty' - если локальное значение отсутствует, то принимать
с абсорбируемого объекта
        // - true - да (по умолчанию для полей, являющихся обратными
ссылками)
        'absorb'=>true,
        //что нужно заджойнить при отображении этого атрибута в списке (для
жадной загрузки + поиск)
        'join'=>['techModel', 'techType']
    ],
    'model'=>['alias'=>'model_id'], //значения параметров атрибута брать
из model_id
]
}

/**
 * @return array[] Ссылками на объекты каких классов являются атрибуты
 * $linksSchema:[
 *     'services_ids'=>[
 *         Service::class,           //на какой класс ссылаемся
 *         'acls_ids',              //если там есть обратная ссылка, то в каком
аттрибуте
 *             'updater' => ['class' =>
ManyToManySmartUpdater::class,], //Если запись в many-2-many таблицу делается
кастомным способом
 *
 *                 //Передается в behaviors()
 *
//https://github.com/voskobovich/yii2-linker-behavior?tab=readme-ov-file#custom-junction-table-values
 *             'loader'=>'servicesList',//как загрузчик этого объекта называется,
если он не формируется автоматически из названия ссылки
 *             'deleteable'=>true,      //можно ли удалять объект с такими
ссылками (если ссылки удаляются в beforeDelete)
 *
 ],
 *
 */
public $linksSchema=[
    'depends_ids' => [Services::class, 'dependants_ids'],
    'comps_ids' => [Comps::class, 'services_ids'],
    'techs_ids' => [Techs::class, 'services_ids'],
    'maintenance_reqs_ids'=>
[MaintenanceReqs::class, 'services_ids'],
    'maintenance_jobs_ids'=>
[MaintenanceJobs::class, 'services_ids'],
    'support_ids' =>
[Users::class, 'support_services_ids', 'loader'=>'support'],

```

```
'infrastructure_support_ids' =>
[Users::class,'infrastructure_support_services_ids','loader'=>'infrastructureSupport'],
    'contracts_ids' => [Contracts::class,'services_ids'],
    'acls_ids' => [Acls::class,'services_id'],

    'responsible_id' => [Users::class,'services_ids'],
    'infrastructure_user_id' =>
[Users::class,'infrastructure_services_ids','loader'=>'infrastructureResponsible'],
        'providing_schedule_id' =>
[Schedules::class,'providing_services_ids'],
        'support_schedule_id' =>
[Schedules::class,'support_services_ids'],
            'segment_id' => [Segments::class,'services_ids'],
            'parent_id' =>
[Services::class,'children_ids','loader'=>'parentService'],
            'partners_id' => [Partners::class,'services_ids'],
            'places_id' => [Places::class,'services_ids'],
            'currency_id' => Currency::class,
];

/***
 * @inheritDoc
 */
public function attributeLabels()
{
    $labels=[];
    foreach ($this->attributeData() as $key=>$data)
        if (isset($data['label']))
            $labels[$key]=$data['label'];
    return $labels;
}

/***
 * @inheritDoc
 */
public function attributeHints()
{
    $hints=[];
    foreach ($this->attributeData() as $key=>$data)
        if (isset($data['hint']))
            $hints[$key]=$data['hint'];
    return $hints;
}
```

From:
<http://wiki.reviakin.net/> - Wiki

Permanent link:
<http://wiki.reviakin.net/%D0%BB%D0%BD%D0%B2%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%BB%D0%B7%D0%B0%D1%86%D0%BB%D1%8F:dev:model:fields?rev=1758440238>

Last update: 2025/09/21 07:37

