1. Что нужно сделать.
2. Сущности.

Сущности кладем в Application\Models, соответственно пространство имен назначаем такое же. Наследуются они у нас от \System\Orm\DomainObject. У сущностей есть ряд параметров, которые мы задаем set-ами и получаем get-ами. Так вот, если наша сущность содержит коллекцию объектов, “getter” должен иметь примерно следующий вид:

*function getAuthors(){*

*if (! isset($this->authors)){*

*$this->authors= $this->getCollection($this->targetClass(),$this->getId());*

*}*

*return $this->authors;*

*}*

В данном случае мы просим коллекцию авторов. В общем, кроме имени нашей сущности все остается неименным.

Каждый “setter” должен содержать в конце такое обращение *$this->markDirty();*

И наконец не забываем определить метод targetClass(), который вернет нам имя нашей сущности. Возможно от него я избавлюсь в дальнейшем, но пока определяйте.

1. Классы

Реализуем наследников классов Collection, IdentityObject, DomainObjectFactory, UpdateFactory, DeferredAuthorCollection в пространстве имен Application\Orm, естественно лежать они должны по тому же пути.

Сейчас пока покажу на примере автора, что и как нужно сделать. Фактически, вам просто нужно внести некоторые изменения, делая по аналогии с уже существующими сущностями.

В принципе что-то подобное я уже описывала в первом гайде.

Не забываем в каждом классе о namespace Application\Orm;

1. class AuthorCollection extends \System\Orm\Collection

Реализуем метод targetClass()- уже давно всем знакомый.

1. class DeferredAuthorCollection extends AuthorCollection

Изменения самые незначительные- вообще никаких, просто пока копируем все, что там есть, только сущность( вместо автора будет что-то другое) поменяется.

1. class AuthorIdentityObject extends \System\Orm\IdentityObject

Как и было сказано ранее, необходимо определить такой же конструктор:

function \_\_construct($field=null){

parent::\_\_construct($field, array('id','name','family','patronymic','paper\_id'));

}

Изменив только массив допустимых полей БД. В основном это названия полей БД, откуда дергается вся инфа для объекта, плюс некоторые поля таблиц-связок. Это значит, что по ходу дела кое-что придется в этот список добавить, если скрипт начнет ругаться и отказываться составить запрос к БД.

Ошибки типа

***Fatal error****: Uncaught exception 'Exception' with message 'Поле name не является корректным полем id,family,patronymic,paper\_id'* не должны вас пугать) Это значит добавьте в конструктор еще одно полу и все.

1. class AuthorDomainObjectFactory extends \System\Orm\DomainObjectFactory

Главный метод

*function doCreateObject(array $array){*

*$obj= new \Application\Models\Author();*

*$obj->setName($array['name']);*

*$obj->setFamily($array['family']);*

*$obj->setPatronymic($array['patronymic']);*

*$obj->setId($array['id']);*

*return $obj;*

*}*

Что тут нужно изменить в случае другой сущности? Создать эту самую другую сущность и наполнить объект необходимыми именно для этой сущности данными.

И метод targetClass().

Что необходимо делать в случае сложного объекта расскажу чуть позже.

1. class AuthorUpdateFactory extends \System\Orm\UpdateFactory

Определяем подобный метод

*function newUpdate(\app\models\DomainObject $obj){*

*$id= $obj->getId();*

*$cond=null;*

*$values['name']=$obj->getName();*

*$values['family']=$obj->getFamily();*

*$values['patronymic']= $obj->getPatronymic();*

*if ($id >-1){*

*$cond['id']=$id;*

*}*

*return $this->buildStatement('author',$values,$cond);*

*}*

Поменяется в зависимости от сущности наполнение массива $values и только.

1. Команды

Как это все использовать в командах? Примерно, опять же на примере автора.

1. ShowProfile

*$factory= \System\Orm\PersistenceFactory::getFactory('Author');*

*$finder= new \System\Orm\DomainObjectAssembler($factory);*

*$idobj=$factory->getIndentityObject()->field('id')->eq($id);*

*$author=$finder->findOne($idobj);*

$id, отмечено красным- откуда его брать- ваша задача. После проделанных манипуляций у нас есть объект автор. Можем смело брать из него все, что нужно и отображать как пожелаете.

1. SaveProfile

*$insrt= new \Application\Models\Author();*

*$insrt->setName($name);*

*$insrt->setFamily($family);*

*$insrt->setPatronymic($patronymic);*

*ObjectWatcher::instance()->performOperations();*

В общем получили из формы, создали объект и добавили в БД. Элементарно (пока объект простой)

1. EditProfile

Тут по сути объединяются две предыдущих команды. Так как обновить мы можем объект, который у нас уже лежит в БД, нам необходим его $id, ну и сам объект конечно.

*$factory= \System\Orm\PersistenceFactory::getFactory('Author');*

*$finder= new \System\Orm\DomainObjectAssembler($factory);*

*$idobj=$factory->getIndentityObject()->field('id')->eq($id);*

*$author=$finder->findOne($idobj);*

Теперь мы берем данные из формы и как бы переопределяем наш объект.

*$author ->setName($name);*

*$author ->setFamily($family);*

*$author ->setPatronymic($patronymic);*

И самое главное, переходим непосредственно к обновлению:

*ObjectWatcher::instance()->performOperations();*

Заключение. Гайд будет дополнен примерами сложного объекта, я не буду заморачиваться и возьму статью. Как работать с IdentityObject, то есть как создавать запросы- смотрите плиз в старом гайде.