Расширение сущности Doctrine ORM

13 ноября 2017

Doctrine Symfony PHP SQL ORM

Иногда на практике бывает необходимо расширить сущности, передставляющие таблицы базы данных в коде. Например, PHP-фреймворк Symfony позволяет создавать сущности в так называемых вендорных бандлах. При необходимости в приложении, использующей этот бандл, можно расширить вендорную сущность свойствами, присущими предметной области данного приложения. В статье рассматриваются различные варианты расширения сущностей на примере Doctrine ORM с использованием языка PHP.

Наследование сущности без дополнительных аннотаций

Начнем с рассмотрения простейшего случая расширения простым наследованием. Для этого создадим сущность `ParentEntity` и расширим его сущностью `ChildEntity`. В данном примере не будем использовать специальные аннотации `Doctrine`, используемые для расширения сущностей.

```
<?php
// src/DataLayerBundle/Entity/ParentEntity.php

namespace DataLayerBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Table(name="parent_entity")
 */
class ParentEntity
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
```

```
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=255)
     */
    protected $name;
}
```

Сущность `ParentEntity` содержит идентификатор и свойсво `$name`, которое является текстовым.

```
<?php
// src/AppBundle/Entity/ChildEntity.php

namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;
use DataLayerBundle\Entity\ParentEntity;

/**
 * @ORM\Table(name="child_entity")
 */
class ChildEntity extends ParentEntity
{
    /**
     * @var int
     *
     * @ORM\Column(name="name", type="integer")
     */
    protected $name;

    /**
     * @var int
     *
     * @ORM\Column(name="some_int", type="integer")
     */
    protected $someInt;
}
```

В дочерней сущности свойство `$name` базового класса переопределено на целочисленное, также добавлено новое свойство `$someInt`. Создадим файл миграции, используя созданные сущности.

```
$ php app/console doctrine:migrations:diff
```

```
// ...
$this->addSql('CREATE TABLE child_entity (id INT NOT NULL, name
VARCHAR(255) NOT NULL, some_int INT NOT NULL, PRIMARY KEY(id))');
$this->addSql('CREATE TABLE parent_entity (id INT NOT NULL, name
VARCHAR(255) NOT NULL, PRIMARY KEY(id))');
// ...
```

Исходя из полученных запросов можно сделать следующие выводы:

1. под каждую сущность создается отдельная таблица;
2. сущность наследует свойства, определенные в родительской сущности;
3. для наследуемых свойств используются аннотации, определенные в родительском классе, даже если свойство было переопределено;
4. свойство, добавленное дочерней сущности, расширяет родительскую.

Наследование сущности с использованием аннотации

`@MappedSuperclass`

Известно, что *Doctrine* представляет функционал наследования сущностей с использованием специальных аннотаций. В следующем примере попробуем использование аннотации `@MappedSuperclass` в базовом классе.

```
<?php
// src/DataLayerBundle/Entity/ParentEntity.php

namespace DataLayerBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\MappedSuperclass
 */
class ParentEntity
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
```

```
    protected $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=255)
     */
    protected $name;
}
```

Дочерний класс `ChildEntity` оставим без изменений.

В результате в файле миграций будет запрос на создание только дочерней таблицы. Стоит обратть внимание, что поле `name` в запросе имеет тип `INT`, который был задан дочерним классом.

```
// ...
$this->addSql('CREATE TABLE child_entity (id INT NOT NULL, name INT
NOT NULL, some_int INT NOT NULL, PRIMARY KEY(id))');
// ...
```

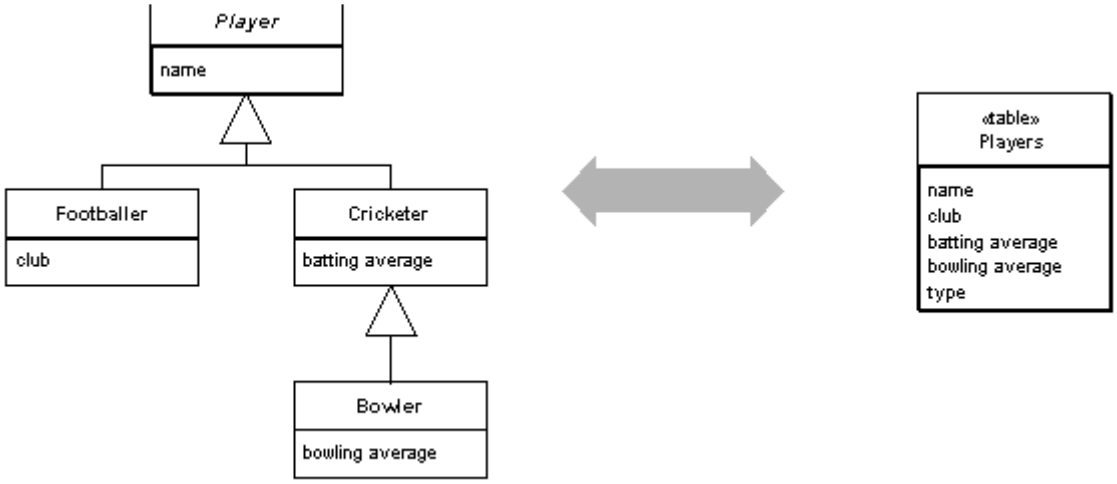
Напрашиваются следующие выводы:

1. создается только таблица для дочерней сущности;
2. сущность наследует свойства, определенные в базовом классе;
3. дочерняя сущность переопределяет свойства, определенные в базовом класс.

А теперь попробуем удалить дочерний класс, оставив при этом аннотацию `@MappedSuperclass` в базовом классе. Посмотрим, что будет в файле миграции? Как и следовало ожидать, там нет запросов.

Наследование с единой таблицей (Single Table Inheritance)

Паттерн «Наследование с единой таблицей» описан Мартиным Фаулером в книге Patterns of Enterprise Application Architecture.



Суть шаблона заключается в расположении полей нескольких классов в единой таблице СУБД. К примеру, это способствует уменьшению количества JOIN'ов при выборке данных из базы данных. Для реализации этого подхода нужно создать родительский класс и аннотировать его следующими аннотациями:

- `@InheritanceType` - указывает тип наследования
- `@DiscriminatorColumn` (опционально) - указывает столбец в таблице базы данных, в котором хранится информация о типе строки относительно иерархии классов
- `@DiscriminatorMap` (опционально) - указывает, какой записью в столбце `@DiscriminatorColumn` идентифицировать отпределнный тип

В исходном коде аннотации `@InheritanceType` видно, что она может принимать значения: `NONE`, `JOINED`, `SINGLE_TABLE`, `TABLE_PER_CLASS`.

```php
<?php

namespace Doctrine\ORM\Mapping;

/**
 * @Annotation
 * @Target("CLASS")
 */
final class InheritanceType implements Annotation
{
    /**
     * The inheritance type used by the class and its subclasses.
     *
     * @var string
     *
     * @Enum({"NONE", "JOINED", "SINGLE_TABLE", "TABLE_PER_CLASS"})
     */
    public $value;
}
```

Для реализации данного паттерна следует использовать `SINGLE_TABLE`.

Создадим необходимые классы.

```
<?php
// src/DataLayerBundle/Entity/ParentEntity.php

namespace DataLayerBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\InheritanceType("SINGLE_TABLE")
 * @ORM\DiscriminatorColumn(name = "discr", type = "string")
 * @ORM\DiscriminatorMap({"parent_entity" = "ParentEntity",
"child_entity" = "AppBundle\Entity\ChildEntity"})
 */
class ParentEntity
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=255)
     */
    protected $name;
}
```

Дочерний класс `ChildEntity` оставим без изменений. В результате в файле миграций появится запрос на создание только одной таблицы `parent_entity`.

```
// ...
$this->addSql('CREATE TABLE parent_entity (id INT NOT NULL, name
VARCHAR(255) NOT NULL, discr VARCHAR(255) NOT NULL, some_int INT
DEFAULT NULL, PRIMARY KEY(id))');
// ...
```

Видно, что:

1. создается только таблица для родительской сущности;
2. в таблицу попадают свойства, определенные в дочернем классе;
3. дочерняя сущность не переопределяет свойства, определенные в базовом класс.

Попробуем выполнить миграцию и выполнить ввод данных через наши сущности, например, в контроллере.

```
// ...
$em = $this->get('doctrine.orm.entity_manager');

$parent = new ParentEntity();
$parent->setName('parent name');

$child = new ChildEntity();
$child->setSomeInt(9999999);
$child->setName('child name');

$em->persist($parent);
$em->persist($child);
$em->flush();
// ...
```

В результате в таблице появятся следующие записи:

```
SELECT * FROM parent_entity;
```

id	name	discr	some_int
1	parent name	parent_entity	
2	child name	child_entity	9999999

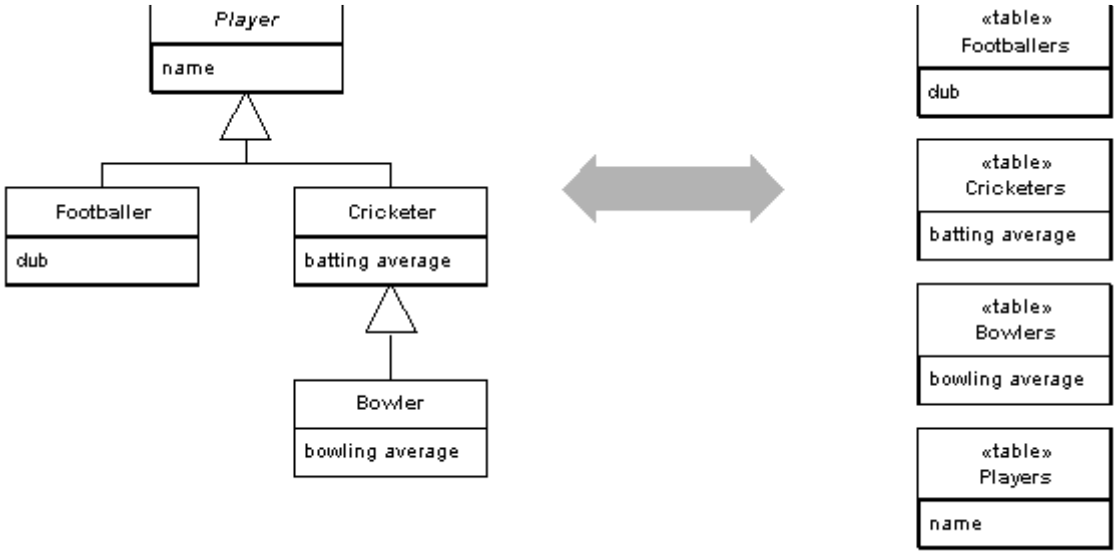
Если не писать аннотацию `@ORM\DiscriminatorMap({"parent_entity" = "ParentEntity", "child_entity" = "AppBundle\Entity\ChildEntity"})` в родительском классе, то значения в столбеце `discr` будут следующими:

```
SELECT id, discr FROM parent_entity;
```

id	discr
1	parententity
2	childentity

Наследование с таблицами классов (Class Table Inheritance)

Паттерн «Наследование с таблицами классов» также описан Мартиным Фаулером в книге Patterns of Enterprise Application Architecture.



В отличии от паттерна «Наследование с единой таблицей» в данном подходе используется одна таблица на один класс в иерархии наследования. Релизация отличается лишь значением аннотации `@InheritanceType`, которым в данном случае является `JOINED`. Родительский класс будет выглядеть следующим образом.

```
<?php
// src/DataLayerBundle/Entity/ParentEntity.php

namespace DataLayerBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity
 * @ORM\InheritanceType("JOINED")
 * @ORM\DiscriminatorColumn(name = "discr", type = "string")
 * @ORM\DiscriminatorMap({"parent_entity" = "ParentEntity",
"child_entity" = "AppBundle\Entity\ChildEntity"})
```

```
 */
class ParentEntity
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=255)
     */
    protected $name;
}
```

Дочерний класс `ChildEntity` снова оставим без изменений. В результате создадутся следующие запросы в файле миграций.

```
// ...
$this->addSql('CREATE TABLE child_entity (id INT NOT NULL, some_int
INT NOT NULL, PRIMARY KEY(id))');
$this->addSql('CREATE TABLE parent_entity (id INT NOT NULL, name
VARCHAR(255) NOT NULL, discr VARCHAR(255) NOT NULL, PRIMARY
KEY(id))');
$this->addSql('ALTER TABLE child_entity ADD CONSTRAINT
FK_677D8034BF396750 FOREIGN KEY (id) REFERENCES parent_entity (id) ON
DELETE CASCADE NOT DEFERRABLE INITIALLY IMMEDIATE');
// ...
```

Таким образом, создаются две таблицы, по одрной на каждую сущность из иерархии классов. В каждой таблице находятся только поля, объявленные в соответсвующем классе сущности. Стоит обратить внимание а то, что создается внешний ключ `child_entity.id -> parent_entity.id`. Произведем ввод данных, аналогичный предыдущему, и сделаем выборку данных из таблиц.

```
SELECT * FROM parent_entity;
```

id	name	discr

id	name	discr
1	parent name	parent_entity
2	child name	child_entity

```
SELECT * FROM child_entity;
```

id	some_int
2	9999999

Тип наследования TABLE_PER_CLASS

В аннотации `@InheritanceType` используем значение `TABLE_PER_CLASS`.

Выполняем команду для создания файла миграций и получаем `Doctrine\ORM\ORMException`:

```
This behaviour is (currently) not supported by Doctrine 2
```

В документации *Doctrine* об этом типе ничего не сказано, но если обратиться к документации Hibernate, то можно сделать вывод, что должны создаваться таблицы для каждой сущности. Причем каждая таблица содержит как свойства конкретной сущности, так и свойства всех родителей. Судя по всему, тоже самое мы получим при аннотировании `@InheritanceType("NONE")`.

Тип наследования NONE

```
$this->addSql('CREATE TABLE child_entity (id INT NOT NULL, name
VARCHAR(255) NOT NULL, some_int INT NOT NULL, PRIMARY KEY(id))');
$this->addSql('CREATE TABLE parent_entity (id INT NOT NULL, name
VARCHAR(255) NOT NULL, PRIMARY KEY(id))');
```

Далее

Повторная обработка сообщений в RabbitMQ с различным временным интервалом ожидания между повторами	28 декабря 2017
Повторное выполнение задач в RabbitMQ	12 ноября 2017