Санкт-Петербургский Государственный Университет
2014 г.

Проектирование и разработка библиотеки ORM на PHP

Медведев Антон Алексеевич
Научный руководитель: к. ф.-м. н. Сысоев Сергей Сергеевич
Рецензент: генеральный директор ООО «ПетроМС», Велюхов Юрий Геннадьевич

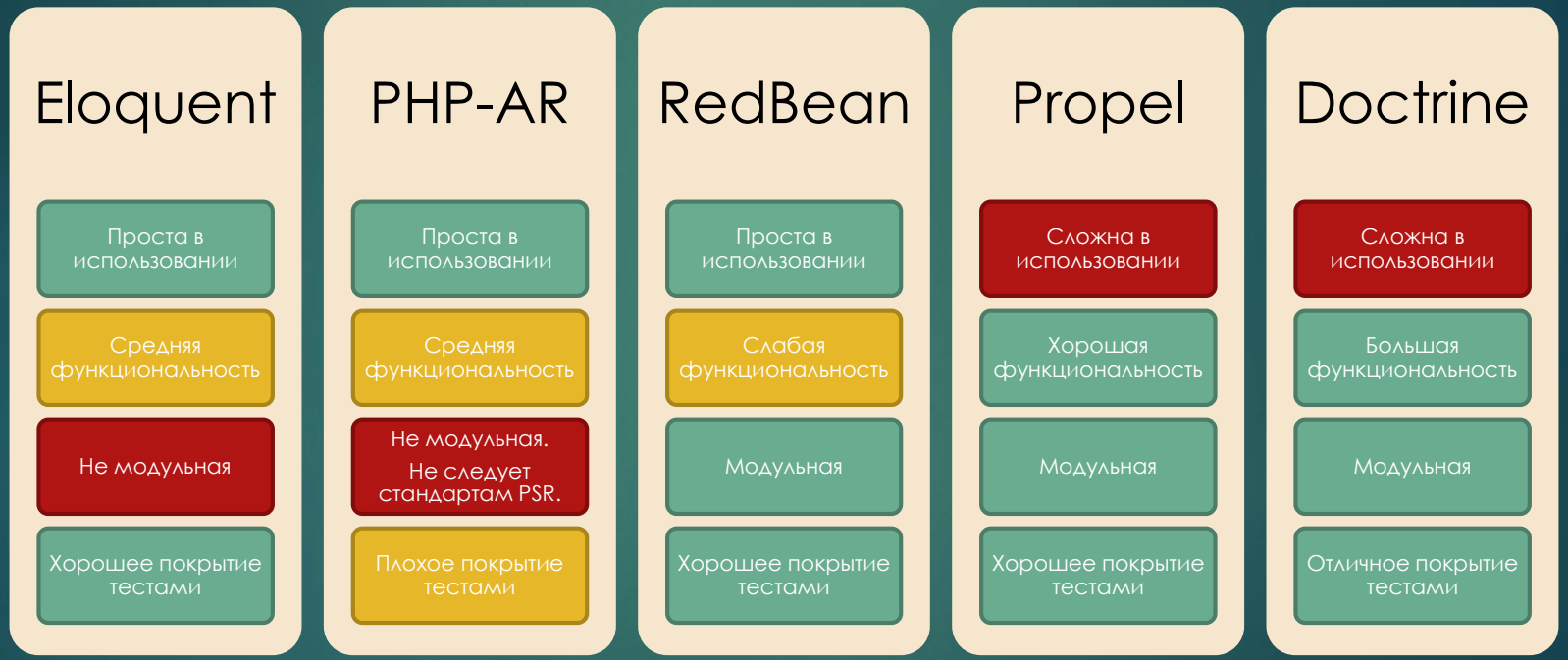
Понятие ORM

ORM — Object-Relation Mapping (рус. Объектно-реляционное отображение)

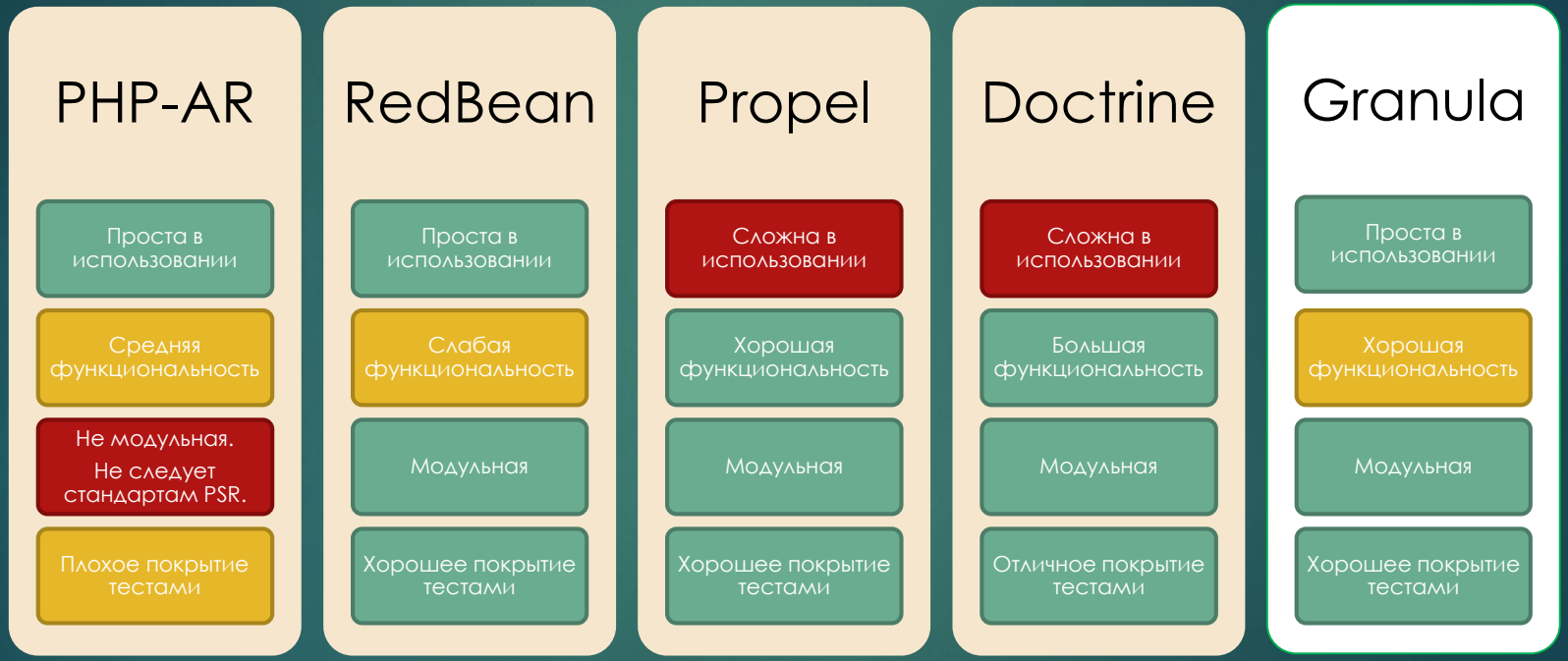
Цели ORM

1. Снижение количества ошибок.
2. Разделение логики приложения и логики хранения данных.
3. Независимость кода приложения от конкретной СУБД.
4. Снижение рисков нарушения целостности данных.
5. Повышение безопасности работы с данными.
6. Уменьшение размеров кода приложения.

Существующие решения



Потребность в новом



Разработка ORM

Возникающие проблемы

1. Степень детализации
2. Наследование
3. Идентичность
4. Ассоциации
5. Доступ к данным

Разработка ORM

Возможные подходы

1. Table Gateway
2. Row Gateway
3. Active Record
4. Data Mapper

[1] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley, 2002.

Разработка ORM

Выбранный подход

1. Table Gateway
2. Row Gateway
3. Active Record
4. Data Mapper

Разработка ORM

Объектная модель


```php
/**
 * Class User
 * @property int $id
 * @property string $name
 * @property string $password
 * @property string $email
 * @property string $avatar
 * @property \Entity\Profile $profile
 * @property \Entity\User $friend
 * @property \DateTime $birthDate
 */
class User
{
    use ActiveRecord;
    use Setter;

    private $id;
    private $name;
    private $password;
    private $email;
    private $avatar;
    private $profile;
    private $friend;
    private $birthDate;
}
```

```php
/**
 * Class Profile
 * @property int $id
 * @property string $address
 * @property int $age
 * @property array $tags
 * @property User $user
 */
class Profile
{
    use ActiveRecord;
    use Setter;

    private $id;
    private $address;
    private $age;
    private $tags;
    private $user;
}
```

```php
/**
 * Class Address
 * @property int $id
 * @property string $address
 * @property int $age
 * @property array $tags
 * @property User $user
 */
class Address
{
    use ActiveRecord;
    use Setter;

    private $id;
    private $country;
    private $city;
    private $street;
}
```

Разработка ORM

Автоматическая генерация SQL

MySQL

```
CREATE TABLE users (id INT AUTO_INCREMENT NOT NULL, profile INT DEFAULT NULL, friend INT DEFAULT NULL, name VARCHAR(255) NOT NULL, password VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL, avatar VARCHAR(255) DEFAULT '', birthDate DATETIME NOT NULL, UNIQUE INDEX UNIQ_1483A5E9E7927C74 (email), INDEX name_email_index (name, email), INDEX IDX_1483A5E98157AA0F (profile), INDEX IDX_1483A5E955EEAC61 (friend), PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci ENGINE = InnoDB
CREATE TABLE profile (id INT AUTO_INCREMENT NOT NULL, address INT DEFAULT NULL, user INT DEFAULT NULL, age INT NOT NULL, tags LONGTEXT NOT NULL COMMENT '(DC2Type:simple_array)', INDEX IDX_8157AA0FD4E6F81 (address), INDEX IDX_8157AA0F8D93D649 (user), PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci ENGINE = InnoDB
CREATE TABLE address (id INT AUTO_INCREMENT NOT NULL, country VARCHAR(255) NOT NULL, city VARCHAR(255) NOT NULL, street VARCHAR(255) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci ENGINE = InnoDB
ALTER TABLE users ADD CONSTRAINT FK_1483A5E98157AA0F FOREIGN KEY (profile) REFERENCES profile (id) ON UPDATE CASCADE
ALTER TABLE users ADD CONSTRAINT FK_1483A5E955EEAC61 FOREIGN KEY (friend) REFERENCES users (id) ON UPDATE CASCADE
ALTER TABLE profile ADD CONSTRAINT FK_8157AA0FD4E6F81 FOREIGN KEY (address) REFERENCES address (id) ON UPDATE CASCADE
ALTER TABLE profile ADD CONSTRAINT FK_8157AA0F8D93D649 FOREIGN KEY (user) REFERENCES users (id) ON UPDATE CASCADE
```

SQLite

```
CREATE TABLE users (id INTEGER NOT NULL, profile INTEGER DEFAULT NULL, friend INTEGER DEFAULT NULL, name VARCHAR(255) NOT NULL, password VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL, avatar VARCHAR(255) DEFAULT '', birthDate DATETIME NOT NULL, PRIMARY KEY(id), CONSTRAINT FK_1483A5E98157AA0F FOREIGN KEY (profile) REFERENCES profile (id) NOT DEFERRABLE INITIALLY IMMEDIATE, CONSTRAINT FK_1483A5E955EEAC61 FOREIGN KEY (friend) REFERENCES users (id) NOT DEFERRABLE INITIALLY IMMEDIATE)
CREATE UNIQUE INDEX UNIQ_1483A5E9E7927C74 ON users (email)
CREATE INDEX name_email_index ON users (name, email)
CREATE INDEX IDX_1483A5E98157AA0F ON users (profile)
CREATE INDEX IDX_1483A5E955EEAC61 ON users (friend)
CREATE TABLE profile (id INTEGER NOT NULL, address INTEGER DEFAULT NULL, user INTEGER DEFAULT NULL, age INTEGER NOT NULL, tags CLOB NOT NULL, PRIMARY KEY(id), CONSTRAINT FK_8157AA0FD4E6F81 FOREIGN KEY (address) REFERENCES address (id) NOT DEFERRABLE INITIALLY IMMEDIATE, CONSTRAINT FK_8157AA0F8D93D649 FOREIGN KEY (user) REFERENCES users (id) NOT DEFERRABLE INITIALLY IMMEDIATE)
CREATE INDEX IDX_8157AA0FD4E6F81 ON profile (address)
CREATE INDEX IDX_8157AA0F8D93D649 ON profile (user)
CREATE TABLE address (id INTEGER NOT NULL, country VARCHAR(255) NOT NULL, city VARCHAR(255) NOT NULL, street VARCHAR(255) NOT NULL, PRIMARY KEY(id))
```

Разработка ORM

Описание схемы

DocBlock

```
/**
 * Class User
 * @property int $id
 * @property string $name
 * @property string $password
 * @property string $email
 * @property string $avatar
 * @property \Entity\Profile $profile
 * @property \Entity\User $friend
 * @property \DateTime $birthDate
 */
```

PHP

```
$meta->table('users');
$meta->field('id', 'integer')->primary()->options(['autoincrement' => true]);
$meta->field('name', 'string');
$meta->field('password', 'string');
$meta->field('email', 'string')->unique()->options(['notnull' => true]);
$meta->field('avatar', 'string')->options(['notnull' => false, 'default' => '']);
$meta->field('profile', 'entity')->entity(Profile::class);
$meta->field('friend', 'entity')->entity(User::class);
$meta->field('birthDate', 'datetime');
$meta->index(['name', 'email'], 'name_email_index');
```

Сценарии использования

Конфигурация и подключение

```php
$params = [
    'dev' => true,

    'driver' => 'pdo_mysql',
    'host' => 'localhost',
    'user' => 'root',
    'password' => '',
    'dbname' => 'granula',
];

$em = new EntityManager($params, [
    User::class,
    Profile::class,
    Address::class,
]);
```

Сценарии использования

Создание объектов

```php
$profile = new Profile();
$profile->age = 21;
$profile->tags = ['one', 'two'];
$profile->birth = new DateTime();
$profile->city = 'Saint Petersburg';
$profile->create();

$user = new User();
$user->name = 'Anton';
$user->avatar = null;
$user->profile = $profile;
$user->friend = User::lazy($friendId = 12);
$user->create();
```

Сценарии использования

Выполнение пользовательских SQL запросов

```php
$result = User::query('SELECT * FROM users u WHERE u.id IN (?)', [[1, 2]], [Connection::PARAM_INT_ARRAY]);

foreach ($result as $user) {
    // Работа с экземпляром $user.
}
```

Результаты

- Простота использования
- Быстрота изучения
- Достаточно широкая функциональность
- Следование стандартам

- Код созданной ORM библиотеки выложен на GitHub: https://github.com/elfet/granula
- Данная библиотека уже используется в разрабатываемом продукте.