# Лабораторная работа 5

# ORM Readbean

**ORM** или **Object-relational mapping** (рус. Объектно-реляционное отображение) — это технология программирования, которая позволяет преобразовывать несовместимые типы моделей в ООП, в частности, между хранилищем данных и объектами программирования. ORM используется для упрощения процесса сохранения объектов в реляционную базу данных и их извлечения, при этом ORM сама заботится о преобразовании данных между двумя несовместимыми состояниями.

RedBeanPHP был создан для значительного облегчения жизни программистов в процессе работы с базами данных. RedBeanPHP требует версию PHP >= 5.3.4.

В RedBeanPHP каждый объект записи называется бином. Эти бины можно воспринимать как самые обычные объекты, свойства которых представляют собой записи в Ваших таблицах. То есть одна запись это один бин, а его свойства это поля Вашей записи. Работать с бином можно точно также как с обычным массивом.

Скачать библиотеку RedBeanPHP можно на официальном сайте.

**Подключение библиотеки.**

```
1. require 'libs/rb.php';
```

Для подключения к базе данных в RedBeanPHP есть статичный метод setup, который принимает 4 опциональных аргумента. Опциональными они являются, потому что Вы можете никакой аргумент не задать и тогда RedBeanPHP создаст временную базу данных в формате SQLite в Вашей временной директории. Вызывается метод setup для MySQL следующим образом:

```
1. R::setup('mysql:host=localhost; dbname=dbname','login','password');
2. if  (!R::testConnection() )
3. {
4. exit ('Нет соединения с базой данных');
5. }
```

Метод testConnection проверяет есть ли у нас фактическое подключение к базе.

Закрыть соединение с базой данных Вы можете при помощи метода close. Вызывается он вот так:

```
1. R::close();
```

**CRUD: Create (Создание записи)**

```
1. // Указываем, что будем работать с таблицей book
2. $book = R::dispense('book');
3. // Заполняем объект свойствами
```

```
$book->title = 'Призрак победы';
$book->price = 199;
// Можно обращаться как к массиву
$book['author'] = 'Макс Глебов';
// Сохраняем объект
R::store($book);
```

**CRUD: Read (Чтение)**

Если нужно получить данные без каких-либо условий, то легче это сделать методами load() и loadAll()

```
// Получаем все записи, ID которых указаны в массиве ids
$ids = [1,2,3];
$books = R::loadAll('book', $ids);
foreach ($books as $book){
  echo $book->title.'<br>';
}
 // Получаем одну запись по её ID
$id = 1;
$book = R::load('book', $id);
echo $book->title;
```

Если по каким-то причинам вам понадобится именно массив данных, то на этот случай есть метод export():

```
$id = 1;
$book = R::load('book', $id);
$book = $book->export();
echo $book['title'];
```

**CRUD: Update (Обновление записи)**

```
$id = 1;
// Загружаем объект с ID = 1
$book = R::load('book', $id);
// Обращаемся к свойству объекта и назначаем ему новое значение
$book->price = 210;
// Сохраняем объект
R::store($book);
```

**CRUD: Delete (Удаление)**

Удалить запись с ID = 5

```
$id = 5;
$book = R::load('book', $id);
```

```
R::trash($book);
```

Удалить записи с ID = 6, 7

```
$ids = [6, 7];
$book = R::loadAll('book', $ids);
R::trashAll($book);

// Начиная с версии 5.1 данную задачу лучше выполнить методом R::trashBatch(). В таком случае нет необходимости создавать (получать) бин - объект RedBeanPHP
$ids = [6, 7];
R::trashBatch('book', $ids);

// Удаление записи с ID = 3
$id = 3;
R::hunt('book', 'id = ?', [$id]);
```

Метод R::wipe() полностью очищает указанную таблицу:

```
R::wipe('book');
```

Метод R::nuke() полностью очищает всю базу данных. Режим заморозки должен быть выключен:

```
R::freeze(false);
R::nuke();
```

**Поиск данных: find(), findOne(), findAll()**

Если вы не знаете идентификатор бина, вы можете искать бины, используя метод find():

```
$min_price = 250;
$books = R::find('book', 'price > ?', [$min_price]);

$search = 'строка';
$books = R::find('book', 'author LIKE ?', ["%$search%"]);

$id = 1;
$min_price = 300;
$books = R::find('book', 'id > :id AND price < :price', [':price' => $min_price, ':id' => $id]);

$ids = [1, 3, 5];
$books = R::find('book', 'id IN (' . R::genSlots($ids) . ')', $ids);
```

Если необходимо получить только одну запись, используем метод findOne():

```
$id = 1;
```

```php
$book = R::findOne('book', 'id = ?', [$id]);

$title = 'гостья из будущего';
$book = R::findOne('book', 'title = ?', [$title]);
```

Если необходимо получить все данные без особых условий, используем метод findAll():

```php
$books = R::findAll('book');
$limit = 5;
$books = R::findAll('book', 'ORDER BY id ASC LIMIT ?', [$limit]);
```

**Метод findLike()**

Данный метод предназначен для поиска по записям.

```php
$search_1 = 'Джон Пристли';
$search_2 = 'Сергей Тармашев';

$books = R::findLike('book',
  ['author' => [$search_1, $search_2]],
  'ORDER BY title ASC'
);
```

**Построение запросов (Querying)**

При использовании RedBeanPHP (как и любой другой ORM) не всегда можно ограничится простыми методами поиска (Finding). Часто существует необходимость сделать более сложный запрос, который сделать простыми методами крайне проблематично. **Важно!** Рассмотренные выше методы Finding необходимо применять, если требуется сделать простой запрос, без каких-либо сложных условий. В рассмотренных ниже примерах всегда возвращается массив данных (а не объекты-бины), поэтому это тоже является плюсом ☺

**Метод exec()**

Метод для произвольного SQL запроса (чаще всего применяется для добавления, изменения и удаления):

```php
$id = 3;
$title = 'New title';

R::exec('UPDATE `book` SET `title` = :title WHERE id = :id', [
  'id' => $id,
  'title' => $title
]);
```

**Метод getAll()**

Вернёт массив данных (все записи/несколько по условию) из указанной таблицы:

```
//$books = R::getAll('SELECT `title` FROM `book`');
$id = 1;
$books = R::getAll('SELECT `title` FROM `book` WHERE `id` > ?', [$id]);

foreach ($books as $book){
   echo $book['title'].'<br>';
}
```

**Метод getRow()**

Вернёт все записи, но выводит только одну. Рекомендуется добавлять LIMIT 1, чтобы и запрашивалась тоже только одна запись:

```
$search = 'поворот';
$book = R::getRow('SELECT * FROM `book` WHERE `author` LIKE :search LIMIT 1', [
   'search' => "%$search%"
]);
```

**Метод getCol()**

Вернёт колонку:

```
// Выбрать все названия всех книг
$books = R::getCol( 'SELECT `title` FROM book' );
```

**Метод getCell()**

Вернёт ячейку одной записи:

```
$id = 5;
$title = R::getCell('SELECT `title` FROM book WHERE `id` = ? LIMIT 1', [$id]);
```

**Метод getAssoc()**

Чтобы получить ассоциативный массив с указанным столбцом ключа и значения, используйте:

```
R::getAssoc('SELECT id, title FROM book');
```

**Метод getInsertID()**

Вернёт ID последней вставленной записи:

```
$res = R::exec("INSERT INTO book (title, author, price) VALUES (?,?,?)", ['New Book', 'New Author', 10]);
$id = R::getInsertID();
```

**Методы convertToBean() и convertToBeans()**

Конвертация массива записей в бины или один бин (convertToBean())

```
$books = R::getAll("SELECT * FROM book");
```

```php
$books = R::convertToBeans('book', $books);

$book = R::getRow("SELECT * FROM book WHERE `id` = ?", [1]);
$book = R::convertToBean('book', $book);
```

**Работа с Базами Данных и их таблицами**

Метод inspect() возвращает названия таблиц в БД. Если параметром передать название таблицы, то он вернёт все поля этой таблицы:

```php
// Какие таблицы есть в БД
$tables = R::inspect();

// Какие поля есть в указанной таблице
$fields = R::inspect('book');
```

**Связи (отношения) в RedBeanPHP**

One-to-many (связь **один ко многим**). Достанем из БД все книги, у которых category_id = 1

```php
$category_id = 1;
$category = R::load('category', $category_id);
$books = $category->ownBookList;

// Сортировка и лимит
$books = $category->with('ORDER BY `title` ASC LIMIT 3')->ownBookList;

// Но более предпочтительным способом является метод withCondition()
$status = 1;
$limit = 3;
$books = $category
  ->withCondition('status = ? ORDER BY title ASC LIMIT ?', [$status, $limit])
  ->ownBookList;

foreach ($books as $book){
  echo $book->title.'<br>';
}
```

Many-to-one (связь **Многие к одному**). Достанет из базы название категории, с которой связана книга

```php
$book = R::load('book', 1);
$category = $book->category->title;
```

Many-to-many (связь **Многие к одному**). Достанет из базы (из связующей таблицы) все книги этой категории:

```php
$category = R::load('category', 1);
$books = $category->sharedBookList;

print_r($books);
```

**Методы подсчёта (Counting)**

Простой подсчёт элементов:

```php
// Сколько записей (элементов) в таблице book
$books = R::count( 'book' );

// Сколько записей (элементов) в таблице book, у которых поле status = 1
$books = R::count( 'book', 'status = ?', [1] );
```

Подсчёт элементов связанных таблиц:

```php
// Сколько записей (элементов) в таблице book, связанных с категорией с ID = 1
$category = R::load('category', 1);
$numBook = $category->countOwn('book');
```

## Задания к лабораторной работе

С использованием Readbean выполните основные операции с базой данных библиотеки:

1. Выполните вставку данных формы в таблицу автор.
2. Выполните редактирование информации об авторе.
3. Удалите выбранный жанр.
4. Выполните запрос, позволяющий сформировать таблицу вида Жанр-Количество книг в жанре.
5. Выполните запрос на поиск книги по ключевому слову, введенному в форму поиска. Выведите всю информацию об этой книге.
6. Найдите книги, написанные за последнее десятилетие. Выведите их в виде маркированного списка в виде Название книги (год написания)