**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](https://redbeanphp.com/index.php?p=/download).

**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
4. $book->title = 'Призрак победы';
5. $book->price = 199;
6. // Можно обращаться как к массиву
7. $book['author'] = 'Макс Глебов';
8. // Сохраняем объект
9. R::store($book);

**CRUD: Read (Чтение)**

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

1. // Получаем все записи, ID которых указаны в массиве ids
2. $ids = [1,2,3];
3. $books = R::loadAll('book', $ids);
4. foreach ($books as $book){
5. echo $book->title.'<br>';
6. }
7. // Получаем одну запись по её ID
8. $id = 1;
9. $book = R::load('book', $id);
10. echo $book->title;

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

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

**CRUD: Update (Обновление записи)**

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

**CRUD: Delete (Удаление)**

Удалить запись с ID = 5

1. $id = 5;
2. $book = R::load('book', $id);
3. R::trash($book);

Удалить записи с ID = 6, 7

1. $ids = [6, 7];
2. $book = R::loadAll('book', $ids);
3. R::trashAll($book);
5. // Начиная с версии 5.1 данную задачу лучше выполнить методом R::trashBatch(). В таком случае нет необходимости создавать (получать) бин - объект RedBeanPHP
6. $ids = [6, 7];
7. R::trashBatch('book', $ids);
9. // Удаление записи с ID = 3
10. $id = 3;
11. R::hunt('book', 'id = ?', [$id]);

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

1. R::wipe('book');

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

1. R::freeze(false);
2. R::nuke();

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

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

1. $min\_price = 250;
2. $books = R::find('book', 'price > ?', [$min\_price]);
4. $search = 'строка';
5. $books = R::find('book', 'author LIKE ?', ["%$search%"]);
7. $id = 1;
8. $min\_price = 300;
9. $books = R::find('book', 'id > :id AND price < :price', [':price' => $min\_price, ':id' => $id]);
11. $ids = [1, 3, 5];
12. $books = R::find('book', 'id IN (' . R::genSlots($ids) . ')', $ids);

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

1. $id = 1;
2. $book = R::findOne('book', 'id = ?', [$id]);
4. $title = 'гостья из будущего';
5. $book = R::findOne('book', 'title = ?', [$title]);

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

1. $books = R::findAll('book');
2. $limit = 5;
3. $books = R::findAll('book', 'ORDER BY id ASC LIMIT ?', [$limit]);

**Метод findLike()**

Данный метод предназначен для поиска по записям.

1. $search\_1 = 'Джон Пристли';
2. $search\_2 = 'Сергей Тармашев';
4. $books = R::findLike('book',
5. ['author' => [$search\_1, $search\_2]],
6. 'ORDER BY title ASC'
7. );

**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):

1. $id = 3;
2. $title = 'New title';
4. R::exec('UPDATE `book` SET `title` = :title WHERE id = :id', [
5. 'id' => $id,
6. 'title' => $title
7. ]);

**Метод getAll()**

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

1. //$books = R::getAll('SELECT `title` FROM `book`');
2. $id = 1;
3. $books = R::getAll('SELECT `title` FROM `book` WHERE `id` > ?', [$id]);
5. foreach ($books as $book){
6. echo $book['title'].'<br>';
7. }

**Метод getRow()**

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

1. $search = 'поворот';
2. $book = R::getRow('SELECT \* FROM `book` WHERE `author` LIKE :search LIMIT 1', [
3. 'search' => "%$search%"
4. ]);

**Метод getCol()**

Вернёт колонку:

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

**Метод getCell()**

Вернёт ячейку одной записи:

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

**Метод getAssoc()**

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

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

**Метод getInsertID()**

Вернёт ID последней вставленной записи:

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

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

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

1. $books = R::getAll("SELECT \* FROM book");
2. $books = R::convertToBeans('book', $books);
4. $book = R::getRow("SELECT \* FROM book WHERE `id` = ?", [1]);
5. $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:

1. // Какие таблицы есть в БД
2. $tables = R::inspect();
4. // Какие поля есть в указанной таблице
5. $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

1. $category\_id = 1;
2. $category = R::load('category', $category\_id);
3. $books = $category->ownBookList;
5. // Сортировка и лимит
6. $books = $category->with('ORDER BY `title` ASC LIMIT 3')->ownBookList;
8. // Но более предпочтительным способом является метод withCondition()
9. $status = 1;
10. $limit = 3;
11. $books = $category
12. ->withCondition('status = ? ORDER BY title ASC LIMIT ?', [$status, $limit])
13. ->ownBookList;
15. foreach ($books as $book){
16. echo $book->title.'<br>';
17. }

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

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

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

1. $category = R::load('category', 1);
2. $books = $category->sharedBookList;
4. print\_r($books);

**Методы подсчёта (Counting)**

Простой подсчёт элементов:

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

Подсчёт элементов связанных таблиц:

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

**Задания к лабораторной работе**

C 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)