МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федерально автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационных систем

Куркчи Ариф Эрнестович

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-31-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Веб-технологии»

на тему «Исследование возможностей хранения данных на стороне сервера. Работа с файлами. Работа с реляционными СУБД.»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. преподаватель   А. Л. Овчинников

(должность) (подпись) (инициалы, фамилия)

Севастополь 2017

1. ЦЕЛЬ РАБОТЫ

Изучить возможности хранения данных на стороне сервера: работу c файлами и СУБД MySQL из PHP, приобрести практические навыки организации хранения данных на стороне сервера в файлах, в базах данных MySQL, а также овладение навыками постраничного вывода данных.

2. ПОСТАНОВКА ЗАДАЧИ

2.1. Разработать базовый класс BaseActiveRecord для работы с базой данных, который реализует паттерн ActiveRecord (данный класс разместить в папке /my\_site/app/core). Данный класс должен реализовать следующие функции:

Таблица 2.1 – Составные компоненты главного AR класса

|  |  |  |
| --- | --- | --- |
| Метод | Возвращаемое значение | Описание |
| save() | Текущий AR (т.е. this) | Создает или обновляет соответствующую запись в БД. Создание происходит для новых записей. Если у записи уже есть $id то происходит выполнение SQL запроса ‘UPDATE’. |
| delete() | true | Удаляет запись из БД к которой привязан данный ActiveRecord объект. |
| find($id) | AR объект | Ищет в таблице, привязанной к текущему AR строку с id = $id и возвращает объект текущего класса, у которого все поля проставлены из колонок соответствующей таблицы. |
| findAll() | Массив AR объектов | Возвращает список всех строк таблицы. Каждая строка – это AR объект. |

2.2. Для всех таблиц, которые будут использоваться при выполнении данной лабораторной работы, создать дочерние AR классы. Для каждого из классов определить все поля и названия таблиц (данные классы необходимо разместить в папке /my\_site/app/models). Например, пусть у нас будут две дочерние таблицы guest\_book и blog.

2.3. Создать новую страницу "Гостевая книга". Страница должна содержать форму ввода (Фамилия, Имя, Отчество, E –mail, Текст отзыва), а также таблицу сообщений, оставленных пользователями. Сообщения в таблице должны располагаться в порядке убывания даты добавления сообщения.Для хранения сообщений пользователей использовать текстовый файл messages.inc, содержащий разделенные символом «;» данные: Дату сообщения, ФИО, E–mail и Текст отзыва. (Для получения текущей даты сервера возможно использовать PHP функцию *date('d.m.y')).*

2.4. Реализовать страницу "Загрузка сообщений гостевой книги", содержащую форму загрузки подготовленного заранее файла messages.inc на сервер.

2.5. Реализовать на странице "Тест по дисциплине" сохранение ответов пользователей и правильности ответов в разработанную таблицу базы данных MySQL, с возможностью просмотра сохраненных данных (дата, ФИО, ответы, верно/неверно).

2.6. Разработать страницу «Редактор Блога», позволяющую добавлять записи Блога. Страница должна содержать форму добавления записи Блога и список выдаваемых постранично записей, отсортированных в порядке убывания даты. Форма добавления должна содержать поля ввода:

* Тема сообщения – поле ввода однострочного текста (заполнение обязательно);
* Изображение – поле ввода файла (заполнение не обязательно);
* Текст сообщения (поле ввода многострочного текста);

Данные хранить в разработанной таблице базы данных MySQL. Валидацию данных осуществлять с использованием класса FormValidation, разработанного при выполнении ЛР №5.

2.7. Разработать страницу «Мой Блог», содержащую упорядоченные в порядке убывания даты добавления, выдаваемые постранично данные:

* Дата и время сообщения;
* Тема сообщения;
* Изображение;
* Текст сообщения;

Данные извлекать из таблицы базы данных MySQL.

Формат постраничного вывода определяется из Таблицы 2.2 в соответствии с вариантом задания.

Таблица 2.2

|  |  |  |
| --- | --- | --- |
| № | Вариант строки ссылок на страницы | Количество записей на странице |
| 4 | Всего страниц: 31  Предыдущая 21 22 23 24 25 Следующая | Постоянное, определяется константой |

2.8. Реализовать возможность добавления записей на страницу «Мой Блог» из файла формата CSV, содержащего следующие поля: title, message, author, created\_at. Например:

"тема 1","сообщение 1","Vasiliy","2014-01-01 14:00"

Для этого необходимо разработать страницу «Загрузка сообщений блога», содержащую форму загрузки файла формата CSV.

Добавление записей из файла в БД осуществлять с использованием подготавливаемых запросов (см. п. 2.4). Валидацию данных осуществлять с использованием класса FormValidation, разработанного при выполнении ЛР №5.

3. ИСХОДНЫЙ КОД

app/Core/Database

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | <?php | | 2 |  | | 3 | namespace App\Core; | | 4 |  | | 5 | /\*\* | | 6 | \* Class Database | | 7 | \* @package App\Core | | 8 | \* @inheritdoc \PDO | | 9 | \*/ | | 10 | *class* Database { | | 11 |  | | 12 | /\*\* | | 13 | \* @var \App\Core\Database | | 14 | \*/ | | 15 | protected static $\_instance; | | 16 | protected $\_pdo; | | 17 |  | | 18 | private *function* \_\_construct( *$host*, *$db*, *$username*, *$password* = '' ) { | | 19 | $dsn        = sprintf( 'mysql:host=%s;dbname=%s', $host, $db ); | | 20 | $this->\_pdo = new \*PDO*( $dsn, $username, $password ); | | 21 | } | | 22 |  | | 23 | public static *function* pdo(): \*PDO* { | | 24 | return static::instance()->\_pdo; | | 25 | } | | 26 |  | | 27 | public static *function* instance(): *Database* { | | 28 | if ( is\_null( static::$\_instance ) ) { | | 29 | static::$\_instance = new static( 'localhost', 'test', 'test', '12345678' ); // TODO move to config | | 30 | if ( ! function\_exists( 'pdo' ) ) { | | 31 | *function* pdo(): \*PDO* { | | 32 | return *Database*::pdo(); | | 33 | } | | 34 | } | | 35 | } | | 36 |  | | 37 | return static::$\_instance; | | 38 | } | | 39 |  | | 40 | public static *function* init() { | | 41 | static::instance(); | | 42 | } | | 43 |  | | 44 | public static *function* \_\_callStatic( *$name*, *$arguments* ) { | | 45 | return call\_user\_func\_array( [ static::instance(), $name ], $arguments ); | | 46 | } | | 47 |  | | 48 | public *function* \_\_call( *$name*, *$arguments* ) { | | 49 | if ( method\_exists( $this->\_pdo, $name ) ) { | | 50 | return call\_user\_func\_array( [ $this->\_pdo, $name ], $arguments ); | | 51 | } | | 52 |  | | 53 | return false; | | 54 | } | | 55 |  | | 56 | } | |  |

app/Core/ActiveRecord

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | <?php | | 2 |  | | 3 | namespace App\Core; | | 4 |  | | 5 | *class* ActiveRecord { | | 6 | protected static $table = ''; | | 7 | protected static $id\_field = 'id'; | | 8 | protected static $fields = [ | | 9 | 'id', | | 10 | ]; | | 11 |  | | 12 | protected $\_attributes = []; | | 13 | protected $\_original = []; | | 14 | protected $\_inserted = false; | | 15 | protected $\_lastError = false; | | 16 |  | | 17 | public *function* \_\_construct( *array* *$attributes* = [], *bool* *$inserted* = false ) { | | 18 | $this->\_attributes = $attributes; | | 19 | if ( $inserted ) { | | 20 | $this->\_inserted = $inserted; | | 21 | $this->\_original = $this->\_attributes; | | 22 | } | | 23 | } | | 24 |  | | 25 | public static *function* create( *array* *$attributes* ) { | | 26 | $instance = new static( $attributes ); | | 27 | $instance->save(); | | 28 |  | | 29 | return $instance; | | 30 | } | | 31 |  | | 32 | public *function* save() { | | 33 | $table    = static::$table; | | 34 | $prepares = []; | | 35 | $values   = []; | | 36 | $fields   = []; | | 37 |  | | 38 | if ( $this->\_inserted ) { | | 39 | $id\_field              = static::$id\_field; | | 40 | $id\_prepare            = ":\_\_$id\_field"; | | 41 | $values[ $id\_prepare ] = $this->\_original[ static::$id\_field ]; | | 42 | foreach ( $this->\_attributes as $key => $value ) { | | 43 | if ( $this->\_original[ $key ] !== $value ) { | | 44 | $prepares[]           = "`$key` = :$key"; | | 45 | $values[ ':' . $key ] = $value; | | 46 | } | | 47 | } | | 48 | if ( ! empty( $prepares ) ) { | | 49 | $keys    = implode( ', ', $prepares ); | | 50 | $prepare = pdo()->prepare( "UPDATE `$table` SET $keys WHERE $id\_field = $id\_prepare" ); | | 51 | if ( $prepare->execute( $values ) ) { | | 52 | $this->\_original = array\_merge( $this->\_original, $this->\_attributes ); | | 53 |  | | 54 | return true; | | 55 | } else { | | 56 | $this->\_lastError = [ | | 57 | 'code' => $prepare->errorCode(), | | 58 | 'info' => $prepare->errorInfo(), | | 59 | ]; | | 60 |  | | 61 | return false; | | 62 | } | | 63 | } | | 64 |  | | 65 | return true; | | 66 | } else { | | 67 | foreach ( $this->\_attributes as $key => $value ) { | | 68 | $fields[]             = "`$key`"; | | 69 | $prepares[]           = ":$key"; | | 70 | $values[ ':' . $key ] = $value; | | 71 | } | | 72 | if ( ! empty( $prepares ) ) { | | 73 | $fields  = implode( ', ', $fields ); | | 74 | $keys    = implode( ', ', $prepares ); | | 75 | $prepare = pdo()->prepare( "INSERT INTO `$table`($fields) VALUES($keys)" ); | | 76 | if ( $prepare->execute( $values ) ) { | | 77 | if ( empty( $this->\_attributes[ static::$id\_field ] ) ) { | | 78 | $this->\_attributes[ static::$id\_field ] = pdo()->lastInsertId(); | | 79 | } | | 80 | $this->\_original = array\_merge( $this->\_original, $this->\_attributes ); | | 81 | $this->\_inserted = true; | | 82 |  | | 83 | return true; | | 84 | } else { | | 85 | $this->\_lastError = [ | | 86 | 'code' => $prepare->errorCode(), | | 87 | 'info' => $prepare->errorInfo(), | | 88 | ]; | | 89 |  | | 90 | return false; | | 91 | } | | 92 | } | | 93 |  | | 94 | return false; | | 95 | } | | 96 | } | | 97 |  | | 98 | public static *function* find( *$id* = false ) { | | 99 | $table = static::$table; | | 100 | if ( $id !== false ) { | | 101 | $id\_field   = static::$id\_field; | | 102 | $id\_prepare = ":\_\_$id\_field"; | | 103 | $prepare    = pdo()->prepare( "SELECT \* FROM `$table` WHERE `$id\_field` = $id\_prepare" ); | | 104 | $prepare->bindValue( $id\_prepare, $id ); | | 105 | } else { | | 106 | $prepare = pdo()->prepare( "SELECT \* FROM `$table`" ); | | 107 | } | | 108 | if ( $prepare->execute() ) { | | 109 | if ( $id !== false ) { | | 110 | return ( $row = $prepare->fetch( \*PDO*::FETCH\_ASSOC ) ) !== false ? new static( $row, true ) : false; | | 111 | } else { | | 112 | $result = []; | | 113 | foreach ( $prepare->fetchAll( \*PDO*::FETCH\_ASSOC ) as $data ) { | | 114 | $result[] = new static( $data, true ); | | 115 | } | | 116 |  | | 117 | return $result; | | 118 | } | | 119 | } else { | | 120 | return false; | | 121 | } | | 122 | } | | 123 |  | | 124 | public static *function* paginate( *$offset* = 0, *$limit* = 0 ) { | | 125 | $table   = static::$table; | | 126 | $prepare = pdo()->prepare( "SELECT \* FROM `$table` ORDER BY `created\_at` DESC LIMIT :offset, :limit" ); | | 127 | $prepare->bindParam( ':offset', $offset, \*PDO*::PARAM\_INT ); | | 128 | $prepare->bindParam( ':limit', $limit, \*PDO*::PARAM\_INT ); | | 129 | if ( $prepare->execute() ) { | | 130 | $result = []; | | 131 | foreach ( $prepare->fetchAll( \*PDO*::FETCH\_ASSOC ) as $data ) { | | 132 | $result[] = new static( $data, true ); | | 133 | } | | 134 |  | | 135 | return $result; | | 136 | } else { | | 137 | return false; | | 138 | } | | 139 | } | | 140 |  | | 141 | public static *function* count() { | | 142 | $table    = static::$table; | | 143 | $id\_field = static::$id\_field; | | 144 | $prepare  = pdo()->prepare( "SELECT COUNT(`$id\_field`) FROM `$table`" ); | | 145 | if ( $prepare->execute() ) { | | 146 | return $prepare->fetchColumn(); | | 147 | } else { | | 148 | return 0; | | 149 | } | | 150 | } | | 151 |  | | 152 | public static *function* deleteAll() { | | 153 | $table = static::$table; | | 154 |  | | 155 | return pdo()->prepare( "DELETE FROM `$table`" )->execute(); | | 156 |  | | 157 | } | | 158 |  | | 159 | public *function* delete() { | | 160 | if ( $this->\_inserted ) { | | 161 | $table      = static::$table; | | 162 | $id\_field   = static::$id\_field; | | 163 | $id\_prepare = ":\_\_$id\_field"; | | 164 | $prepare    = pdo()->prepare( "DELETE FROM `$table` WHERE `$id\_field` = $id\_prepare" ); | | 165 | $prepare->bindValue( $id\_prepare, $this->\_original[ $id\_field ] ); | | 166 |  | | 167 | if ( $prepare->execute() ) { | | 168 | $this->\_inserted = false; | | 169 |  | | 170 | return true; | | 171 | } else { | | 172 | $this->\_lastError = [ | | 173 | 'code' => $prepare->errorCode(), | | 174 | 'info' => $prepare->errorInfo(), | | 175 | ]; | | 176 |  | | 177 | return $this; | | 178 | } | | 179 | } | | 180 |  | | 181 | return $this; | | 182 | } | | 183 |  | | 184 | public *function* \_\_get( *$name* ) { | | 185 | if ( array\_key\_exists( $name, $this->\_attributes ) ) { | | 186 | return $this->\_attributes[ $name ]; | | 187 | } | | 188 |  | | 189 | return null; | | 190 | } | | 191 |  | | 192 | public *function* \_\_set( *$name*, *$value* ) { | | 193 | if ( in\_array( $name, static::$fields ) ) { | | 194 | $this->\_attributes[ $name ] = $value; | | 195 | } | | 196 | } | | 197 |  | | 198 | } | |  |

app/Controller/BlogController

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | <?php | | 2 |  | | 3 | namespace App\Controller; | | 4 |  | | 5 | use App\Core\*Controller*; | | 6 | use App\Core\*Form*; | | 7 | use App\Core\*Validation*; | | 8 | use App\Model\*Post*; | | 9 | use *function* App\Core\view; | | 10 |  | | 11 | *class* BlogController extends *Controller* { | | 12 |  | | 13 | protected $per\_page = 10; | | 14 | protected $\_rules = [ | | 15 | 'title'   => 'required', | | 16 | 'message' => 'required', | | 17 | ]; | | 18 | protected $\_messages = [ | | 19 | 'title'   => [ | | 20 | 'required' => 'Поле обязательное', | | 21 | 'words'    => 'Введите заголовок', | | 22 | ], | | 23 | 'message' => [ | | 24 | 'required' => 'Поле обязательное', | | 25 | 'email'    => 'Введите содержимое поста', | | 26 | ], | | 27 | ]; | | 28 |  | | 29 | *function* index() { | | 30 | $this->page( 1 ); | | 31 | } | | 32 |  | | 33 | *function* page( *$n* ) { | | 34 | $total = (*int*) ceil( *Post*::count() \* 1.0 / $this->per\_page ); | | 35 | if ( $n <= 0 || $n > $total ) { | | 36 | $n = 1; | | 37 | } | | 38 | $posts    = *Post*::paginate( $this->per\_page \* ( $n - 1 ), $this->per\_page ) ?: []; | | 39 | $count    = count( $posts ); | | 40 | $paginate = [ | | 41 | 'per\_page'   => $this->per\_page, | | 42 | 'on\_page'    => $count, | | 43 | 'is\_empty'   => count( $posts ) === 0, | | 44 | 'current'    => $n, | | 45 | 'total'      => $total, | | 46 | 'is\_last'    => $n >= $total, | | 47 | 'is\_first'   => $n == 1, | | 48 | 'left\_tail'  => max( 1, $n - 2 ), | | 49 | 'right\_tail' => min( $total, $n + 2 ), | | 50 | 'prev'       => max( 1, $n - 1 ), | | 51 | 'next'       => min( $total, $n + 1 ), | | 52 | ]; | | 53 |  | | 54 | view( 'blog.index', compact( 'posts', 'paginate' ) ); | | 55 | } | | 56 |  | | 57 | *function* create() { | | 58 | $form = new *Form*( $\_REQUEST, *Validation*::run( $this->\_rules, $this->\_messages ) ); | | 59 | if ( $form->success() ) { | | 60 | $image = ''; | | 61 | if ( ! empty( $\_FILES['image'] ) ) { | | 62 | $file  = $\_FILES['image']; | | 63 | $image = sprintf( | | 64 | '/uploads/%s.%s', | | 65 | hash( 'sha256', $file['name'] . time() ), | | 66 | pathinfo( $file['name'], PATHINFO\_EXTENSION ) | | 67 | ); | | 68 | $img   = new \*Imagick*( realpath( $file['tmp\_name'] ) ); | | 69 | $img->scaleImage( 300, 300, true ); | | 70 | $img->writeImage( \_\_APP\_ROOT\_\_ . 'public' . $image ); | | 71 | $img->destroy(); | | 72 | } | | 73 | *Post*::create( [ | | 74 | 'title'   => $form->val( 'title' ), | | 75 | 'message' => $form->val( 'message' ), | | 76 | 'image'   => $image, | | 77 | ] ); | | 78 | header( 'Location: /blog' ); | | 79 | exit; | | 80 | } | | 81 |  | | 82 | view( 'blog.create', compact( 'form' ) ); | | 83 | } | | 84 |  | | 85 | *function* import() { | | 86 | $result = false; | | 87 | if ( count( $\_REQUEST ) && ! empty( $\_FILES['file'] ) ) { | | 88 | $rewrite = ! empty( $\_REQUEST['rewrite'] ); | | 89 | $file    = $\_FILES['file']; | | 90 | if ( substr\_compare( $file['name'], '.csv', - 4, 4, true ) !== 0 ) { | | 91 | $result = [ | | 92 | 'type'    => 'danger', | | 93 | 'message' => 'Расширение файла должно быть .csv', | | 94 | ]; | | 95 | } else { | | 96 | $data   = file\_get\_contents( $file['tmp\_name'] ); | | 97 | $count  = $this->import\_posts( $data, $rewrite ); | | 98 | $result = [ | | 99 | 'type'    => 'success', | | 100 | 'message' => sprintf( 'Успешно загружено %d записей', $count ), | | 101 | ]; | | 102 | } | | 103 | } | | 104 |  | | 105 | view( 'blog.import', compact( 'result' ) ); | | 106 | } | | 107 |  | | 108 | protected *function* import\_posts( *$data*, *$rewrite* ) { | | 109 | $imported = 0; | | 110 | if ( $rewrite ) { | | 111 | *Post*::deleteAll(); | | 112 | } | | 113 | foreach ( explode( PHP\_EOL, $data ) as $tmp ) { | | 114 | if ( empty( trim( $tmp ) ) ) { | | 115 | continue; | | 116 | } | | 117 | $tmp = explode( '","', substr( trim( $tmp ), 1, - 1 ) ); | | 118 | if ( count( $tmp ) !== 5 ) { | | 119 | continue; | | 120 | } | | 121 | $created\_at = date\_create\_from\_format( 'h:i:s d.m.Y', $tmp[3] ); | | 122 | $updated\_at = date\_create\_from\_format( 'h:i:s d.m.Y', $tmp[4] ); | | 123 | if ( $created\_at === false || $updated\_at === false ) { | | 124 | continue; | | 125 | } | | 126 | *Post*::create( [ | | 127 | 'title'      => trim( $tmp[0] ), | | 128 | 'message'    => trim( $tmp[1] ), | | 129 | 'image'      => trim( $tmp[2] ), | | 130 | 'created\_at' => $created\_at, | | 131 | 'updated\_au' => $updated\_at, | | 132 | ] ); | | 133 | $imported ++; | | 134 | } | | 135 |  | | 136 | return $imported; | | 137 | } | | 138 |  | | 139 | } | |  |

app/Controller/GuestController

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1 | <?php | | 2 |  | | 3 | namespace App\Controller; | | 4 |  | | 5 | use App\Core\*Controller*; | | 6 | use App\Core\*Form*; | | 7 | use App\Core\*Validation*; | | 8 | use *function* App\Core\view; | | 9 |  | | 10 | *class* GuestController extends *Controller* | | 11 | { | | 12 | public $\_ajaxOnly = ['validate']; | | 13 | protected $\_rules = [ | | 14 | 'name'  => 'required,words:2-3', | | 15 | 'email' => 'required,email', | | 16 | 'text'  => 'required', | | 17 | ]; | | 18 | protected $\_messages = [ | | 19 | 'name'  => [ | | 20 | 'required' => 'Поле обязательное', | | 21 | 'words'    => 'Введите в формате: Фамилия Имя Отчество', | | 22 | ], | | 23 | 'email' => [ | | 24 | 'required' => 'Поле обязательное', | | 25 | 'email'    => 'Введите действительную почту', | | 26 | ], | | 27 | 'text'  => 'Поле обязательное', | | 28 | ]; | | 29 |  | | 30 | protected $storage = 'storage/messages.inc'; | | 31 |  | | 32 | *function* index() | | 33 | { | | 34 | $form = new *Form*($\_REQUEST, *Validation*::run($this->\_rules, $this->\_messages)); | | 35 | if ($form->success()) { | | 36 | $matches = mb\_split('\s+', $form->val('name')); | | 37 | $message = [ | | 38 | 'first\_name'  => $matches[1], | | 39 | 'last\_name'   => $matches[0], | | 40 | 'middle\_name' => empty($matches[2]) ? '' : $matches[2], | | 41 | 'email'       => $form->val('email'), | | 42 | 'text'        => $form->val('text'), | | 43 | ]; | | 44 | $this->add\_message($message); | | 45 | $form->clear(); | | 46 | } | | 47 | $messages = $this->get\_messages(); | | 48 |  | | 49 | view('guest.index', compact('messages', 'form')); | | 50 | } | | 51 |  | | 52 | private *function* add\_message(*$message*) | | 53 | { | | 54 | $message['date'] = date('h:i:s d.m.Y', time()); | | 55 | $file = fopen($this->storage, 'a'); | | 56 | fputs($file, implode(';', $message) . PHP\_EOL); | | 57 | fclose($file); | | 58 | } | | 59 |  | | 60 | private *function* get\_messages() | | 61 | { | | 62 | $messages = []; | | 63 | if (file\_exists($this->storage)) { | | 64 | $file = fopen($this->storage, 'r'); | | 65 |  | | 66 | while ($tmp = fgets($file)) { | | 67 | $tmp = explode(';', trim($tmp)); | | 68 | $messages[] = [ | | 69 | 'first\_name'  => $tmp[0], | | 70 | 'last\_name'   => $tmp[1], | | 71 | 'middle\_name' => $tmp[2], | | 72 | 'email'       => $tmp[3], | | 73 | 'text'        => $tmp[4], | | 74 | 'date'        => date\_create\_from\_format('h:i:s d.m.Y', $tmp[5]), | | 75 | ]; | | 76 | } | | 77 | fclose($file); | | 78 | } | | 79 |  | | 80 | return $messages; | | 81 | } | | 82 |  | | 83 | *function* import() | | 84 | { | | 85 | $result = false; | | 86 | if (count($\_REQUEST) && !empty($\_FILES['file'])) { | | 87 | $rewrite = !empty($\_REQUEST['rewrite']); | | 88 | $file = $\_FILES['file']; | | 89 | if (substr\_compare($file['name'], '.inc', -4, 4, true) !== 0) { | | 90 | $result = [ | | 91 | 'type'    => 'danger', | | 92 | 'message' => 'Расширение файла должно быть .inc', | | 93 | ]; | | 94 | } else { | | 95 | $data = file\_get\_contents($file['tmp\_name']); | | 96 | $count = $this->import\_messages($data, $rewrite); | | 97 | $result = [ | | 98 | 'type'    => 'success', | | 99 | 'message' => sprintf('Успешно загружено %d записей', $count), | | 100 | ]; | | 101 | } | | 102 | } | | 103 |  | | 104 | view('guest.import', compact('result')); | | 105 | } | | 106 |  | | 107 | private *function* import\_messages(*$data*, *$rewrite* = false) | | 108 | { | | 109 | $messages = $rewrite ? [] : $this->get\_messages(); | | 110 | $imported = 0; | | 111 | foreach (explode(PHP\_EOL, $data) as $tmp) { | | 112 | if (empty(trim($tmp))) { | | 113 | continue; | | 114 | } | | 115 | $tmp = explode(';', trim($tmp)); | | 116 | if (count($tmp) !== 6) { | | 117 | continue; | | 118 | } | | 119 | $date = date\_create\_from\_format('h:i:s d.m.Y', $tmp[5]); | | 120 | if($date === false) { | | 121 | continue; | | 122 | } | | 123 | $messages[] = [ | | 124 | 'first\_name'  => $tmp[0], | | 125 | 'last\_name'   => $tmp[1], | | 126 | 'middle\_name' => $tmp[2], | | 127 | 'email'       => $tmp[3], | | 128 | 'text'        => $tmp[4], | | 129 | 'date'        => date\_create\_from\_format('h:i:s d.m.Y', $tmp[5]), | | 130 | ]; | | 131 | $imported++; | | 132 | } | | 133 | usort($messages, *function* (*$a*, *$b*) { | | 134 | return $a['date']->getTimestamp() - | | 135 | $b['date']->getTimestamp(); | | 136 | }); | | 137 | foreach($messages as &$message) { | | 138 | $message['date'] = $message['date']->format('h:i:s d.m.Y'); | | 139 | } | | 140 |  | | 141 | $file = fopen($this->storage, 'w'); | | 142 | foreach ($messages as $message) { | | 143 | fputs($file, implode(';', $message) . PHP\_EOL); | | 144 | } | | 145 | fclose($file); | | 146 |  | | 147 | return $imported; | | 148 | } | | 149 |  | | 150 | *function* validate() | | 151 | { | | 152 | echo json\_encode(*Validation*::run($this->\_rules, $this->\_messages)); | | 153 | } | | 154 |  | | 155 | } | |  |

Вывод

В ходе выполнения лабораторной работы были изучены возможности хранения данных на стороне сервера, работа с файлами и СУБД MySQL из PHP, приобретены практические навыки организации хранения данных на стороне сервера в файла, в базах данных MySQL, а также был освоен навык постраничного вывода данных.