2. *Goldshtejn S., Zurbalev D., Flatov I.* Optimizaciya prilozhenij na platforme .Net. 2-e izd. M.: DMK Press, 2014. 524 s.
3. *Dzhepiks F., Troelsen E.* Yazyk programmirovaniya C# 7 i platformy .NET i .NET Core. 8-e izd. M.: Vil'yams, 2018. 1328 s.
4. *Lippman S.B., Lazhoje Zh., Mu B.E.* Yazyk programmirovaniya C++. Bazovyj kurs. 5-e izd. M.: Vil'yams, 2017. 1118 s.
5. *Lutc M.* Izuchaem Python. 4-e izd. M.: Simvol-Plyus, 2011. 1280 s.
6. *Makkonnell S.* Sovershennyj kod. Master-klass. M.: BKhV-Peterburg, 2017. 896 s.
7. *Mejers S.* Effektivnyj i sovremennyj S++. 42 rekomendacii po ispol'zovaniyu C++11 i C++14. M.: Vil'yams, 2018. 304 s.
8. *N'yuland K., Evans B.* Java. Optimizaciya programm. Prakticheskie metody povysheniya proizvoditel'nosti prilozhenij v JVM. M.: Vil'yams, 2019. 448 s.
9. *Ramal'o L.* Python. K vershinam masterstva. M.: DMK Press, 2016. 768 s.
10. Uskorenie koda na Python sredstvami samogo yazyka / Khabr [Elektronnyj resurs]. – URL: https://habr.com/ru/post/124388/ (data obrashcheniya 11.11.2020).

DOI: 10.25586/RNU.V9187.21.01.P.165 УДК 004.4

С.В. Бондаренко, Т.В. Жукова, Э.М. Вихтенко

РАЗРАБОТКА ОБЪЕКТНО-ОРИЕНТИРОВАННОГО КОМПОНЕНТНОГО ФРЕЙМВОРКА, РЕАЛИЗУЮЩЕГО ПАРАДИГМУ MVC

Разработан фреймворк на языке PHP, поддерживающий парадигму разделения данных приложения. Реализована возможность подключения к нескольким видам баз данных благодаря использованию PDO (PHP Data Objects). Описаны реакции приложения на действия пользователя. Разработанное программное средство использует для работы с базами данных классы ActiveRecord, выполняет перехват и обработку ошибок, обеспечивает ввод и валидацию форм, аутентификацию и авторизацию пользователей, позволяет генерировать исходный PHP-код для CRUD-операций, а также поддерживает шаблоны оформления веб-страниц.
Ключевые слова: MVC-фреймворк, язык PHP, веб-сайт, CRUD-операции, генерация кода, виджет.

S.V. Bondarenko, T.V. Zhukova, E.M. Vikhtenko

DEVELOPMENT OF OBJECT-ORIENTED COMPONENT FRAME IMPLEMENTING THE MVC PARADIGM

A framework in PHP has been developed that supports separation of data paradigm. Implemented the ability to connect to several types of databases using PDO (PHP Data Objects). The application's reactions to user actions are described. The developed software tool uses the ActiveRecord classes to work with databases, intercepts and handles errors, provides form input and validation, user authentication and authorization, allows generating PHP source code for CRUD operations, and also supports web page design templates.
Keywords: MVC framework, PHP language, website, CRUD operations, code generation, widget.

Современные тенденции развития компьютерных и информационных технологий

Введение

Число предпринимателей, желающих представить свой бизнес в интернете, с каждым днем увеличивается. Особенно усилилась эта тенденция в связи с введением ограничительных мер на личные встречи из-за распространения коронавирусной инфекции. Одним из условий успешного существования в сети является разработка сайта компании. В связи с этим спрос на создание сайтов в последние годы постоянно растет. Перед разработчиком в первую очередь возникает проблема выбора инструментария для создания сайта. В большинстве случаев приходится делать выбор между написанием сайта с нуля, использованием CMS (система управления контентом) или применением фреймворка [7]. Использование профессионально разработанного фреймворка существенно облегчает создание информационной системы. Фреймворки содержат в себе библиотеки функций, позволяющие выстраивать каркас будущей системы, объединять разные компоненты программного проекта в единое целое.

В данной работе описана реализация фреймворка, поддерживающего парадигму разделения данных приложения (MVC-фреймворк) на языке PHP [4]. Выбор языка PHP обусловлен тем, что он поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяемых для создания динамических веб-сайтов [2].

Разрабатываемый фреймворк должен:

- удовлетворять парадигме Model – View – Controller;
- использовать ActiveRecord для работы с базами данных;
- выполнять перехват и обработку ошибок;
- обеспечивать ввод и валидацию форм;
- проводить аутентификацию и авторизацию пользователей;
- генерировать исходный PHP-код для CRUD-операций;
- поддерживать шаблоны оформления для их легкой смены.

Парадигма MVC

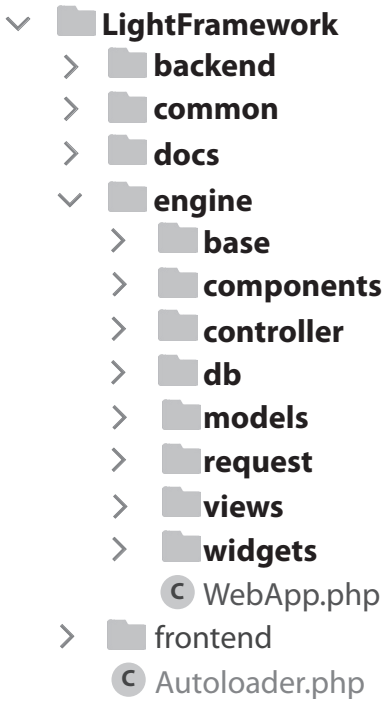
MVC (Model – View – Controller) – схема разделения данных приложения, предназначенная для отделения бизнес-логики от пользовательского интерфейса. Этот подход позволяет разработчикам легко изменять отдельные части приложения, не затрагивая другие [3, 5].

Компонент «Модель» предоставляет данные и реагирует на команды компонента «Контроллер», изменяя свое состояние. Все запросы пользователя приходят на контроллер. Специальная часть URL-адреса указывает на метод, который требуется выполнить. Контроллер проверяет наличие запрошенного метода, выполняет проверку прав доступа, создает модель и передает ее в представление. Операции просмотра, создания, изменения и удаления записей в базе данных (БД) контроллер реализует по умолчанию. Компонент «Представление» вызывает функции модели для получения данных и выводит их в виде html-кода [6]. Представление предоставляет возможность изменения данных модели через html-формы. Пользователь изменяет данные в форме, и они в виде post-запроса отправляются на адрес контроллера. Контроллер принимает данные, производит проверку на наличие CSRF-атаки, позволяющей злоумышленнику выполнить на незащищенном сайте различные действия от имени зарегистрированных пользователей, и в случае ее от-

сутствия загружает измененные данные в модель [1]. Далее модель выполняет валидацию и сохранение данных в БД.

Реализация фреймворка

На рисунке представлена структура каталогов разработанного фреймворка.



Структура каталогов фреймворка

В корневой папке фреймворка (engine) расположены следующие компоненты:

- *base* – классы базовых операций (содержит класс Asset для подключения стилей и js-скриптов, классы Exceptions для обработки исключений, классы Helpers для вывода html-тегов, класс Logger для выполнения логирования, класс Validator для проверки корректности значений и класс User для работы с пользователями);
- *components* – компоненты приложения (генератор кода Gsc, URLManager, AccessManager);
- *controller* – базовый класс-контроллер;
- *db* – классы для работы с БД (содержит набор классов DataProvider для выполнения выборок из БД, родительский класс AR моделей ActiveRecord, класс DataBase для соединения с БД);
- *models* – базовый класс моделей;
- *request* – обработчик http-запросов;
- *views* – базовые представления;
- *widgets* – набор виджетов (включает виджет ActiveForm для создания html-форм,

виджет GridView для формирования таблиц на основе моделей, виджет Breadcrumbs для вывода маршрута до текущего местоположения, виджет Sidebar для вывода бокового меню);

- *WebApp* – основной класс приложения, отвечающий за его инициализацию.

Базовый класс контроллера находится в пространстве имен engine\controller под именем Controller. В классе реализованы функции selectAction, render, redirect.

Функция selectAction находит запрашиваемое действие и проверяет соответствие запрашиваемых и переданных аргументов. При обнаружении несоответствия во время проверки аргументов генерируется исключение ArgumentNotFoundException. При нахождении действия функция выполняет проверку на наличие прав доступа у текущего пользователя для выполнения этого действия.

В таблице приведены возможные ситуации и реакция приложения.

Реакция приложения на действие пользователя

Ситуации	Реакция приложения
Пользователь авторизован, права на доступ имеются	Действие выполняется
Пользователь не авторизован, права на доступ имеются	Действие выполняется
Пользователь авторизован, прав на доступ нет	Перенаправление на страницу ошибки с кодом 403 и текстом ошибки об отсутствии доступа
Пользователь не авторизован, прав на доступ нет	Перенаправление пользователя на страницу авторизации

Функция render выполняет формирование представления на основе шаблона и файла представления. Создается объект класса View, который непосредственно выполняет загрузку файла представления. Представление является php-скриптом. Подключение выполняется функцией require_once, что позволяет получать значения переменных, установленных внутри представления, а также получать доступ к переменным вне представления.

Представление содержит html-код, который в обычном режиме сразу выводится при подключении файла. Данное поведение не позволяет выводить содержимое представления внутри шаблона. Для реализации этой возможности используется буферизация вывода.

Буферизация вывода перенаправляет поток вывода в переменную $content, при этом позволяет выполнить php-код внутри представления. Далее выполняется очистка буфера вывода. В результате имеется переменная $content, которая содержит требуемый код представления. Переменная передается в файл шаблона и выводится в блоке для контента страницы.

В веб-приложении все модели наследуются от класса ActiveRecord, который находится в пространстве имен engine\db. В данном классе реализованы методы, выполняющие операции добавления, обновления и удаления записей базы данных. Для каждой операции генерируется свой sql-запрос на основе полей и данных конкретной модели-наследника.

Операции добавления и обновления выполняются методом класса save, который выбирает действие на основе флага isNew. Сначала выполняется валидация данных модели в

соответствии с правилами в массиве ActiveRecord::$attribute Labels. Если данные не проходят валидацию, генерируется исключение соответствующего типа. Далее проверяется флаг isNew. Если флаг равен true, то генерируется INSERT-запрос, иначе – UPDATE-запрос. После проверки флага выполняется метод generateQuery, который возвращает строку для запроса. Далее полученная строка передается в функцию PDO->prepare, в случае успеха возвращается объект PDO Statement. В конце выполняется метод Statement->execute ($data). Если ошибок не обнаружено, то управление возвращается контроллеру, в противном случае генерируется исключение типа DatabaseException.

Для создания html-форм реализован виджет ActiveForm, расположенный в пространстве имен engine\widgets\ActiveForm.

Он позволяет генерировать html-форму следующим образом.

1. Вызывается статический метод класса ActiveForm::begin(), который создает открывающий тег <form> с атрибутами и скрытое поле с _csrf-токеном. Метод begin возвращает объект формы.

2. У объекта формы вызывается метод field, который принимает аргумент объекта модели, название поля и набор атрибутов поля и возвращает объект класса engine\widgets\Field.

3. У полученного объекта вызывается один из методов для завершения создания полей:

- *textInput* – создает и заполняет текстовое поле type = 'text';
- *textarea* – создает и заполняет элемент <textarea>;
- *hiddenInput* – создает и заполняет скрытое поле;
- *passwordInput* – создает и заполняет поле type = 'password'.

4. Вызывается метод ActiveForm::end(), который выводит закрывающий тег формы.

Далее данные отправляются на сервер; перед сохранением данных модели сравнивается значение поля _csrf со значением токена в WebApp::$user->token. Если значения не совпадают, значит, была попытка совершения CSRF-атаки.

Значение свойства WebApp::$user->token вычисляется при каждой авторизации пользователя на сайте, что снижает риск проведения атаки данного вида.

Помимо валидации формы браузером перед отправкой производится проверка на корректность данных при вызове метода $model->load(). Если хотя бы одно из полей не удалось загрузить, функция возвращает false. Также проверка корректности производится при вызове метода $model->save(). Если одно из полей содержит некорректные данные, генерируется исключение, соответствующее обнаруженной проблеме. Это не позволяет выполнять запись в базу некорректных данных.

При использовании данного фреймворка разработчик может указать в модели правила валидации формы, которые будут переданы в HTML как атрибуты, что поможет исключить большое количество ошибок ввода данных и обеспечит корректную работу web-приложения.

Разработанный фреймворк позволяет выполнять аутентификацию и авторизацию пользователя. Пользователи добавляются администратором сайта путем включения данных о пользователе в таблицу users и назначением ролей доступа в таблице access_roles. Пароль пользователя шифруется функцией password_hash, что гарантирует криптостойкость пароля. В таблице users сохраняются данные пользователя и хеш-пароля.

При аутентификации пользователя выполняется поиск пользователя с полученным логином, в случае успеха проверяется корректность пароля. В случае успеха выполняется старт сессии, и объекту WebApp::$users присваивается найденный в базе пользователь. Если пользователь не найден или получен некорректный пароль, отправляется соответствующее сообщение.

Для перехвата и обработки ошибок весь код функции WebApp::run() обернут в конструкцию try{…} catch(){…}. Это позволяет выполнить обработку исключений и ошибок, возникших на любом уровне вложенности функций. Каждое исключение имеет определенный тип. Реализованы следующие классы исключений:

- *ActionNotFoundException* – запрошенное действие не существует;
- *ArgumentNotFoundException* – не найден аргумент;
- *CSRFDetectedException* – обнаружена CSRF-атака;
- *DataBaseException* – ошибка при работе с БД;
- *EmptyRequiredFieldException* – отсутствуют данные в обязательном поле;
- *FileExistsException* – файл уже существует;
- *FileNotFoundException* – файл не найден;
- *ForbiddenException* – доступ запрещен;
- *InvalidDataException* – данные имеют неверный формат;
- *ModelNotFoundException* – модель не найдена;
- *ParameterNotFoundException* – параметр не найден;
- *PropertyNotFoundException* – не найдено свойство объекта;
- *URLNotFoundException* – запрашиваемый адрес не найден.

Фреймворк предоставляет возможность генерации PHP-кода. При разработке сайта часто требуется создавать большое количество типовых классов, таких как контроллеры, модели, представления. Так как данные классы имеют типовую структуру, можно выделить шаблоны для каждого типа файлов, которые будут заполняться данными, полученными от пользователя. В данной работе реализованы функции для генерации моделей, контроллеров, представлений для выполнения CRUD-операций над таблицами. Для этой цели создан компонент Gsc. Он позволяет генерировать классы моделей, контроллеров и представлений на основе шаблонов.

Имеются шаблоны класса контроллера, класса модели, класса searchModel для выполнения различных выборок данных из БД, шаблон представления формы для редактирования модели и др.

В генераторе кода предусмотрены два режима – генерация моделей и генерация CRUD-операций. CRUD-операциями называют такие функции манипулирования данными, как create (создание), read (чтение), update (модификация), delete (удаление).

Генерация моделей происходит по следующему сценарию.

1. Получение данных от пользователя (таблица, имя класса модели, пространство имен, имя класса родителя, шаблон кода и др.).

2. Получение списка полей таблицы из базы данных.

3. Заполнение шаблона данными.

4. Сохранение полученного кода в файл.

Генерация CRUD-операций происходит по схожему сценарию.

Заключение

В работе использована технология создания объектно ориентированного компонентного MVC-фреймворка на языке программирования PHP. Создано программное средство, соответствующее поставленным задачам. Реализована возможность подключения к нескольким видам базам данных благодаря использованию PDO (PHP Data Objects), что сильно снижает риск возникновения sql-инъекций (атака, позволяющая внедрять в данные вредоносный sql-код). Ошибки и исключения, возникающие при реализации приложений на базе фреймворка, успешно перехватываются и выводятся на специальной странице приложения. Все данные, переданные пользователем, проходят проверку на допустимость значений.

Литература

1. Безопасное программирование на PHP / Web-технологии [Электронный ресурс]. – URL: https://htmlweb.ru/php/php_protect.php (дата обращения: 01.11.20).
2. Дронов В.А. HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов. СПб.: БХВ-Петербург, 2014. 416 с.
3. Лучшие практики MVC / Yii PHP Framework [Электронный ресурс]. – URL: https://www.yiiframework.com/doc/guide/1.1/ru/basics.best-practices (дата обращения: 01.11.20).
4. Никсон Р. Создаем динамическиее веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд. СПб.: Питер, 2013. 560 с.
5. Сафронов М. Разработка веб-приложений в Yii 2. М.: ДМК Пресс, 2015. 392 с**.**
6. DOM Enlightenment [Электронный ресурс]. – URL: http://domenlightenment.com (дата обращения: 01.11.20).
7. PHP MVC Framework Tutorial: CodeIgniter Example / Guru 99 [Электронный ресурс]. – URL: https://www.guru99.com/php-mvc-frameworks.html (дата обращения: 01.11.20).

Literatura

1. Bezopasnoe programmirovanie na PHP / Web-tekhnologii [Elektronnyj resurs]. – URL: https://htmlweb.ru/php/php_protect.php (data obrashcheniya: 01.11.20).
2. Dronov V.A. HTML 5, CSS 3 i Web 2.0. Razrabotka sovremennykh Web-sajtov. SPb.: BKHV-Peterburg, 2014. 416 s.
3. Luchshie praktiki MVC / Yii PHP Framework [Elektronnyj resurs]. – URL: https://www.yiiframework.com/doc/guide/1.1/ru/basics.best-practices (data obrashcheniya: 01.11.20).
4. *Nikson R.* Sozdaem dinamicheskiee veb-sajty s pomoshch'yu PHP, MySQL, JavaScript i CSS. 2-e izd. SPb.: Piter, 2013. 560 s.
5. *Safronov M.* Razrabotka veb-prilozhenij v Yii 2. M.: DMK Press, 2015. 392 s.
6. DOM Enlightenment [Elektronnyj resurs]. – URL: http://domenlightenment.com (data obrashcheniya: 01.11.20).
7. PHP MVC Framework Tutorial: CodeIgniter Example / Guru 99 [Elektronnyj resurs]. – URL: https://www.guru99.com/php-mvc-frameworks.html (data obrashcheniya: 01.11.20).