УДК 004.05:656.052.1

ПРОГРАММНЫЙ ПАТТЕРН ПРОЕКТИРОВАНИЯ АРХИТЕКТУРНОГО КАРКАСА MODEL-VIEW-CONTROLLER ПРИ РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЙ СИСТЕМ МОНИТОРИНГА СПЕЦТРАНСПОРТА АЭРОПОРТА

Б.П. ЕЛИСЕЕВ, А.В. ТАРАСЕНКО, О.А. ГОРБАЧЕВ, ЛЮ ДЖОНДА

В работе рассмотрены паттерны проектирования MODEL-VIEW-CONTROLLER (MVC), и MODEL-VIEW-PRESENTER (MVP), способы взаимодействия и роли каждого компонента архитектурных каркасов, возможности применения сервисов при разработке приложения. Приведены диаграммы пары http-запрос-ответ при использовании сервисов и без них. Описаны основные возможности фреймворка AngularJS. Показана структура взаимодействия компонентов веб-приложения при использовании фреймворка.

Ключевые слова: паттерн, веб-приложение, сервис, JavaScript, MVC, MVP, AngularJS.

ВВЕДЕНИЕ

Стремительный рост сети Интернет - это, по меньшей мере, частичная заслуга ее использования в качестве инструмента личной публикации. Изначально данные, которыми люди обменивались по сети, состояли в основном из статической информации, хранящейся в файлах. Файлы можно было редактировать, но действительно динамических информационных служб было мало.

Ситуация качественно изменилась с появлением динамической сети, являющейся результатом развития динамических служб. Появились новые сервисы - от CGI скриптов для поисковых машин до пакетов программ, соединявших веб-приложения с реляционными базами данных. Стало недостаточным создание веб-сайта - возникла необходимость проектировать веб-приложение [1].

Веб-приложение - особый вид программного продукта, который доступен пользователям по сети, использует браузер в качестве клиента и состоит из набора клиент-серверных сценариев, HTML-страниц и других ресурсов, которые могут быть распределены между несколькими серверами. Само приложение доступно пользователям по определенному пути внутри веб-сервера.

К основным преимуществам веб-приложений можно отнести кросс-платформенную совместимость и возможность обновлять и поддерживать приложение без установки программного обеспечения на клиентские устройства.

1. ПОСТАНОВКА ЗАДАЧИ

На начальной стадии разработки программных приложений стоит задача выбора архитектуры. Архитектура программного обеспечения в традиционном смысле включает определение всех модулей программ, их иерархии и сопряжения между ними и данными.

Основными критериями при выборе архитектуры веб-приложения системы мониторинга спецтранспорта аэропорта являются возможность модульной разработки, сопровождения, документации и структуризации кода.

2. ПАТТЕРН MVC

Разрабатываемая в МГТУ ГА система мониторинга спецтранспорта аэропорта - это динамическое приложение, которое использует JavaScript и AJAX для организации HTTP запро-

сов к веб-серверу. Веб-приложение, в отличие от статических HTML страниц, может динамически представлять контент с учетом параметров запроса, действия пользователя и настроек безопасности.

Как правило, веб-приложение состоит из нескольких подприложений. Та часть приложения системы мониторинга, которую видят обычные пользователи обладает достаточно нагруженным интерфейсом, состоящим из различных модулей. Среди основных можно отметить модуль для отображения транспортных средств на цифровой карте, модуль для создания и корректирования зон и коридоров, модуль формирования отчетов. Все эти модули обладают собственными данными и управляющей логикой.

При проектировании подобных приложений необходимо подобрать подходящие объекты, отнести их к различным классам, соблюдая разумную степень детализации, определить интерфейсы классов и установить существенные отношения между классами.

Паттерны или шаблоны проектирования дают возможность повторно использовать те решения, которые оказались удачными в прошлом и упрощают повторное использование удачных проектных и архитектурных решений. Представление прошедших проверку временем методик в виде паттернов проектирования облегчает доступ к ним со стороны разработчиков новых систем. Паттерны дают разработчику возможность быстрее найти «правильный путь» Здесь под паттернами проектирования понимается описание взаимодействия объектов и классов, адаптированных для решения общей задачи проектирования в конкретном контексте [2].

Архитектурный паттерн Model-view-controller (MVC) предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Контроллер – так, что модификация каждого компонента может осуществляться независимо.

Впервые понятие MVC возникло в языке Smalltalk-80 при подходе к разработке многоуровневых графических пользовательских интерфейсов в конце 1970-х, начале 1980-х. В дальнейшем паттерн зарекомендовал себя как удачная архитектура программного обеспечения.

В настоящее время паттерн MVC доступен для многих языков программирования и хорошо проявил себя в таких фреймворках, как Apache Struts для Java, Maypole для Perl и Rails для Ruby.

Хотя большинство современных фреймворков пытаются развивать парадигмы MVC, для лучшего соответствия потребностям развития веб-приложений, остается один фреймвор, который придерживается классического паттерна, описанного в Smalltalk-80 - Maria.js Питера Мишо.

Модель, как первый компонент шаблона, обрабатывает состояние приложения и может содержать в себе любые данные, например, массив координат. Данный компонент не имеет информации о состоянии HTML- или веб-серверов, его задача - обеспечить возможность запроса состояния объекта и определить пути его изменения.

Второй компонент - Представление, которое отображает пользовательский интерфейс. В приложении, как правило, используется несколько Представлений. Основная функция Представления - запрос данных из Модели, без возможности изменения её состояния. В веб-приложениях на базе MVC Представление создается с использованием HTML, которое в конечном итоге либо отображается как полноценная страница, либо подставляется в требуемое место на странице.

Все действия пользователя, которые он производит в Представлении обрабатываются третьим компонентом MVC - Контроллером. Он получает запрос пользователя (HTTP-запрос), обрабатывает его и переводит в последовательность действий, которые должна исполнить Модель. Кроме того, контроллер выбирает подходящий режим для обработки ответа Модели.

Следующая функция Контроллера - выбор соответствующей страницы (Представления), в зависимости от статуса, который возвращает Модель. Ведь в зависимости от того, имеет ли Модель запрашиваемые данные или нет, пользователю необходимо либо отобразить их, либо уведомить об ошибке. Эта функция полезна не только при разработке сложных приложений с большим количеством страниц, но и для простых одностраничных приложений. С помощью Контроллера можно обрабатывать, например, такие действия, как аутентификации и управления сессиями.

3. ПАТТЕРН MVP

Архитектурный паттерн Model-View-Presenter (MVP) является производным от MVC, который фокусируется на улучшении логики представления. Паттерн возник в компании Taligent в начале 1990-х годов, когда они работали над библиотеками CommonPoint для C ++. Несмотря на то, что и MVC и MVP разделяют задачи между несколькими компонентами, существуют некоторые фундаментальные различия между ними.

Компонент Presenter содержит бизнес-логику пользовательского интерфейса. В отличие от MVC, вызовы от Представления делегированы Presenter, который извлекает данные из Модели и форматирует их для отображения в Представлении. Наиболее распространенной реализацией MVP является использование пассивного Представления, практически не содержащего логики. Модели MVP почти идентичны MVC моделям, и те, и другие обрабатывают данные приложения. Presenter выступает в роли посредника между моделями, при их полной изоляции относительно друг друга. Он эффективно связывает модели Представлением, то есть берет те функции, которые ранее выполняли контроллеры в MVC. Как следствие Presenter является центральным звеном паттерна MVP, реализующим логику Представлений, кроме того, он отвечает за управление событиями пользовательского интерфейса (например, mouseDown, keyDown).

Вызываемый через Представление, Presenter выполняет любую работу с запросами пользователя и передачей данных обратно. Работа подразумевает извлечение, манипулирование и определение того, как данные должны быть отображены в Представлении. В некоторых реализациях, Presenter взаимодействует со службами сохранения данных (Модели). Модели могут вызвать события, на которые Presenter может подписываться, для последующего обновления Представления.

Преимущество вышеописанного отличия от MVC состоит в том, что оно увеличивает тестируемость приложения и обеспечивает более четкое разделение между Представлением и Моделью. Но отсутствие привязки данных в структуре паттерна означает то, что это становится отдельно задачей, о которой стоит позаботиться при разработке приложения.

Как правило, MVP чаще используется в приложениях корпоративного уровня, где необходимо как можно больше использовать логику Presenter. Может оказаться, что MVC не совсем подходит для приложений с очень сложными пользовательским интерфейсом, предполагающим взаимодействие нескольких Представлений, так как организация взаимодействия ложится на несколько контроллеров. В MVP вся подобная сложная логика может быть инкапсулирована в Presenter, который может значительно проще в написании и сопровождении.

В зависимости от реализации MVP может быть более легким при автоматическом модульном тестировании, чем MVC. Presenter может содержать в себе сразу весь интерфейс и поэтому его можно протестировать независимо от других компонентов. Но все это зависит от выбранного языка, на котором осуществляется реализация MVP.

В целом, основные проблемы MVC скорее всего справедливы для MVP, учитывая, что различия между ними, в основном, семантические. И если при разработке при-

ложения аккуратно отделять проблемы при написании Модели, Представления и Контроллера или Presenter можно смело полагаться на те преимущества, которые дают оба паттерна.

4. MVC ДЛЯ ВЕБ-ПРИЛОЖЕНИЙ

Чтобы применить MVC для веб-приложений, авторы использовали комбинацию скриптов, серверных компонентов и обычных объектов для реализации различных компонентов в рамках приложения.

На рис. 1 представлена временная диаграмма пары запрос-ответ на примере запроса списка активного спеацавтотранспорта.

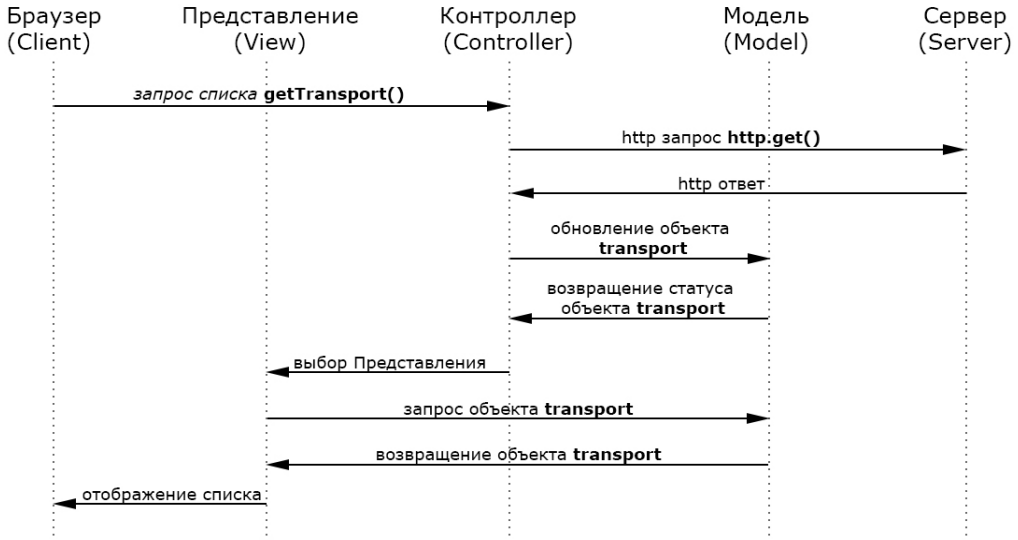

Рис. 1. Временная диаграмма пары запрос-ответ

5. СЕРВИСЫ

При разработке веб-приложения возникают ситуации, когда Модель и Контроллер взаимодействуют достаточно тесно и практически невозможно разделить код на два компонента. Для исключения этого эффекта используют схему, при которой Контроллер предоставляет минимальные действия, например, только обработку HTTP-запросов и передачу результата в Модель. В этом случае Модель может содержать в себе необходимые методы обработки результатов и форматирования выходных данных.

В качестве альтернативы данному решению предполагается отделить от связки Модель-Контроллер так называемый Сервис.

Сервисы, выступают в роли библиотек многократного использования кода для других компонентов приложения. Они обеспечивают приложение сквозным функционалом, например, логированием, обработкой ошибок, проверкой прав доступа, обменом сообщений и кэшированием. Сервисы могут содержать код для запроса и хранения данных с внешних серверов. Они также могут включать в себя функциональные возможности для сортировки, фильтрации и преобразования данных в различные форматы по мере необходимости [3].

На рис. 2 представлена временная диаграмма пары запрос-ответ на примере запроса списка активного спеацтранспорта с использованием сервиса для работы с данными по транспорту.

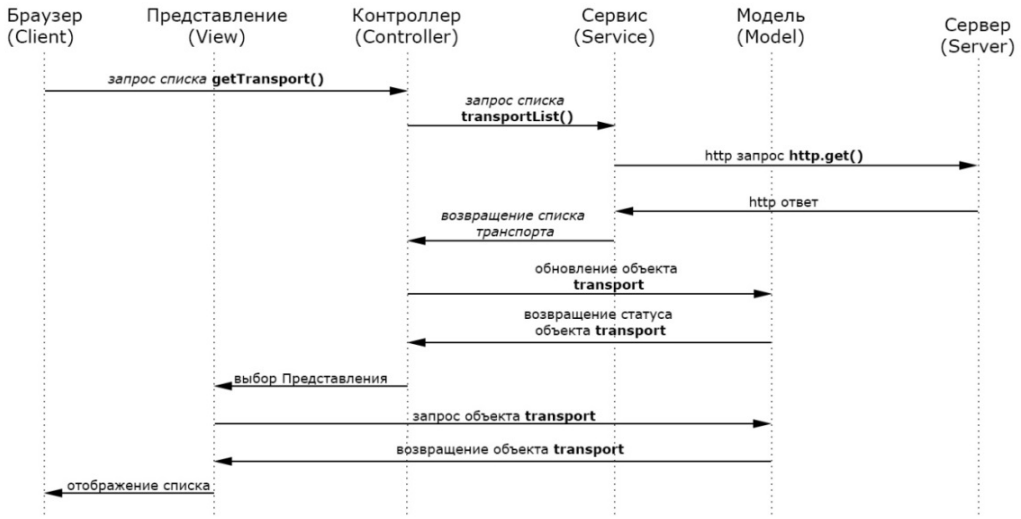

Рис. 2. Временная диаграмма пары запрос-ответ при наличии Сервиса

6. ФРЕЙМВОРК ANGULARJS

Реализация веб-приложения системы мониторинга спецтранпорта аэропорта с использованием паттерна MVC возможна в следующей связке: в качестве реализации компонента Представление возможно применение языка разметки HTML, компоненты Модель и Контроллер реализуются на языке JavaScript, общий функционал паттерна обеспечивает каркас или так называемый фреймворк AngularJS.

AngularJS - это клиентский фреймворк, написанный на JavaScript и выполняемый в браузере. Основная цель фреймворка - расширение браузерных приложений на основе MVC шаблона, а также упрощение тестирования и разработки. AngularJS адаптирует и расширяет традиционный HTML, чтобы обеспечить двустороннюю привязку данных для динамического контента, что позволяет автоматически синхронизировать Модель и Представление. Механизм построения Представлений не требует явно обновлять дерево DOM, так как фреймворк способен следить за действиями пользователя, событиями браузера и изменениями в Модели и вовремя обнаруживать, когда и какое Представление требуется обновить [4].

AngularJS имеет в запасе механизм внедрения зависимостей (Dependency Injection), который существенно упрощает сборку веб-приложений из небольших, надежно протестированных служб.

Общая структура взаимодействия компонентов MVC в рамках модуля приложения для формирования отчетов с использованием данного фреймворка показана на рис. 3.

Рис. 3. Общая структура взаимодействия компонентов MVC в AngularJS

Немаловажным фактором является декларативный подход AngularJS к конструированию пользовательского интерфейса. С практической точки зрения это означает, что Представления описывают желаемый эффект, а не способы его достижения. Возможность декларативного описания Представлений позволяет быстро создавать сложное и интерактивные интерфейсы. А принятие решений о том, когда и как изменять элементы DOM берет на себя фреймворк.

ЗАКЛЮЧЕНИЕ

В работе развивается подход к реализации программного паттерна проектирования MVC относительно веб-приложений. На основе изложенного анализа можно сделать вывод, о целесообразности применения паттерна MVC при разработке веб-приложений систем мониторинга спецтранспорта аэропорта. Данный подход позволит значительно упростить модульную разработку и дальнейшее сопровождение программного продукта.

ЛИТЕРАТУРА

1. **Шкляр Л., Розен Р.** Архитектура веб-приложений. – М.: Эксмо, 2011.
2. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.** Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2014.
3. **Lavin J.** *AngularJS Services*. Pack Publishung Ltd., Aug. 2014, ch. 1, pp. 8-11.
4. **Козловский П., Дарвин П**. Разработки веб-приложений с использованием AngularJS. – М.: ДМК-Пресс, 2014.

MVC SOFTWARE ARCHITECTURE PATTERN FOR WEB-APPLICATION OF SYSTEM OF MONITORING OF AIRPORT TRANSPORT

Eliseev B.P., Tarasenko A.V., Gorbachev O.A., Lyu Dzhonda

The article describes MVC and MVP patterns, ways of interaction and the role of each component of the pattern, the possibility of using services when developing an application. The diagrams of http-request-response pairs are presented when using services and without them. The main features of the AngularJS framework are described.

Keywords: pattern, web-application, service, JavaScript, MVC, MVP, AngularJS.

REFERENCES

1. **Shklyar L., Rozen R.** *Arhitektura web-prilozheni* (Web-application architecture), Moscow, Eksmo, 2011, 640 p.
2. **Gamma E., Helm R., Dzhonson R., Vlissides Dzh.** *Priemyi ob'ektno-orientirovannogo proektirovaniya. Patternyi proektirovaniya* (Design Patterns. Elements of Reusable Object-Oriented Software), Saint Petersburg, Piter, 2014. 368 p.
3. **Lavin J.** AngularJS Services. Pack Publishung Ltd., Aug. 2014, ch. 1, pp. 8-11.
4. **Kozlovskiy P., Darvin P**. *Razrabotki veb-prilozheniy s ispolzovaniem AngularJS* (Mastering web application development with AngularJS), Moscow, DMK-Press, 2014, 394 p.

Сведения об авторах

Елисеев Борис Петрович, 1957 г.р., окончил Дальневосточный государственный университет (1982), профессор, доктор юридических наук, доктор технических наук, заслуженный юрист РФ, ректор МГТУ ГА, почетный профессор Технологического университета Нингбо, автор более 150 научных работ, область научных интересов – государственное управление, административное, финансовое, воздушное право.

Тарасенко Алексей Вячеславович, 1987 г.р. окончил МГТУ ГА (2014), начальник отдела развития информационных сервисов ООО «Интеллектуальные телематические системы для транспорта», область научных исследований - проектирование пользовательских интерфейсов.

Горбачев Олег Анатольевич, 1959 г.р., окончил ИГУ (1982), доктор технических наук, профессор, директор Иркутского филиала МГТУ ГА, автор 56 научных работ, область научных интересов – радиофизика, спутниковые системы навигации.

Лю Джонда, 1963 г.р., окончил университет в г. Чанъань, доктор наук по специальности: Мосты и тоннели инженерные, почетный профессор МГТУ ГА, ректор Технологического университета Нингбо, автор более 30 научных работ, область научных интересов – транспортное образование, транспортная инфраструктура.