УДК 004.457

ПОДХОДЫ К ПРОЕКТИРОВАНИЮ АРХИТЕКТУРЫ СЕРВЕРНОЙ ЧАСТИ НА ПРИМЕРЕ ВЕБ-ПРИЛОЖЕНИЯ «ВЕБ-ПРИЛОЖЕНИЕ ПО НЕКОММЕРЧЕСКИМ ПЕРЕВОЗКАМ ПАССАЖИРОВ»

Н.А. БОБКОВ
(Представлено: Д.Ф. ПАСТУХОВ)

Анализируются технологии, используемые при реализации серверной части веб-приложений. Рассмотрен фрейморк Spring FrameWork. Показаны подходы к проектированию архитектуры серверной части на примере веб-приложения «веб-приложение по некоммерческим перевозкам пассажиров».

Стремительное развитие Интернет-технологий послужило толчком к появлению веб-приложений. Разработка веб-приложений – это не просто создание сайтов, это системы электронных платежей и интернет-банкинг, корпоративные порталы, включающие в себя документооборот, почту, календарь и множество других функций.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент – сервер». Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет ее клиенту по сети с использованием протокола HTTP.

Само веб-приложение может выступать в качестве клиента других служб, например, базы данных или другого веб-приложения, расположенного на другом сервере. Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки программирования, способные осуществлять вывод в стандартную консоль [1].

Серверная часть веб-приложения – это программа или скрипт на сервере, обрабатывающая запросы пользователя (точнее, запросы браузера). База данных – программное обеспечение на сервере, занимающееся хранением данных и их выдачей в нужный момент.

Разработка веб-приложений по системе front – end и back – end подразумевает иерархическое разделение процесса создания ресурса на две части: на разработку пользовательского интерфейса (фронтэнда) и его программно-административной части (бэкэнда).

Средства решения задачи

Рассмотрим основные средства реализации бэкэнда веб-приложения по некоммерческим перевозкам пассажиров.

Бэкэнд development – это процесс программирования веб-приложения и наполнения его функционалом. Создание ядра приложения, разработка платформы, наполнение его основным функционалом и создание административной зоны – это и есть бэкэнд разработка.

Бэкэнд производит обработку пользовательской информации, полученной из front-офиса, и возвращает front – end результат в понятной ему форме. Бэкэнд-программирование – это веб программирование, целью которого является реализация серверной стороны сайта, интеграция базы данных и связь ее с пользовательской (front-end) стороной. Разработка бэкэнда сайта также включает настройку и установку на сервер необходимого программного обеспечения. Проще говоря, фронтэнд передает информацию и команды от пользователя в бэкэнд, а тот, в свою очередь, производит их обработку. Или если уж совсем просто, то front – end создается для посетителя сайта, а back – end для его администратора [3].

В работе были использованы сервлеты. Сервлет является интерфейсом Java, реализация которого расширяет функциональные возможности сервера. Сервлет взаимодействует с клиентами посредством принципа запрос – ответ. Хотя сервлеты могут обслуживать любые запросы, они обычно используются для расширения веб-серверов. Для таких приложений технология Java Servlet определяет HTTP – специфичные сервлет классы.

Жизненный цикл сервлета состоит из следующих шагов:

1. В случае отсутствия сервлета в контейнере:

- класс сервлета загружается контейнером;
- контейнер создает экземпляр класса сервлета;
- контейнер вызывает метод init(). Этот метод инициализирует сервлет и вызывается, в первую очередь, до того, как сервлет сможет обслуживать запросы. За весь жизненный цикл метод init() вызывается только один раз.

2. Обслуживание клиентского запроса. Каждый запрос обрабатывается в своем отдельном потоке. Контейнер вызывает метод service() для каждого запроса. Этот метод определяет тип пришедшего запроса и распределяет его в соответствующий этому типу метод для обработки запроса. Разработчик сервлета

должен предоставить реализацию для этих методов. Если поступил запрос, метод для которого не реализован, вызывается метод родительского класса и обычно завершается возвращением ошибки инициатору запроса.

3. В случае если контейнеру необходимо удалить сервлет, он вызывает метод destroy(), который снимает сервлет из эксплуатации. Подобно методу init(), этот метод тоже вызывается единожды за весь цикл сервлета [1].

Также использовался контейнер сервлетов Apache Tomcat (в старых версиях – Catalina). Apache Tomcat (в старых версиях – Catalina) – контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces(JSF). Написан на языке Java.

Tomcat позволяет запускать веб-приложения, содержит ряд программ для самоконфигурирования. Он используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish [2].

Особо важной использованной технологией был фреймворк Spring, свободно распространяемый фреймворк, созданный Родом Джонсоном. Главная цель направлена на упрощение разработки приложений на языке Java. В своем устремлении на сложности, связанные с разработкой на языке Java, фреймворк Spring использует четыре ключевые стратегии:

- легковесность и ненасильственность благодаря применению простых Java-объектов (POJO);
- слабое связывание посредством внедрения зависимостей и ориентированности на интерфейсы;
- декларативное программирование через аспекты и общепринятые соглашения;
- уменьшение объема типового кода через аспекты и шаблоны.

Spring Framework имеет довольно широкую функциональность и активно используется при разработке сложных бизнес-приложений. Spring Framework может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании:

- inversion of Control контейнер: конфигурирование компонент приложений и управление жизненным циклом Java объектов;
- фреймворк аспектно-ориентированного программирования: работает с функциональностью, которая не может быть реализована возможностями объектно-ориентированного программирования на Java без каких-либо потерь;
- фреймворк доступа к данным: работает с системами управления реляционными базами данных на Java платформе;
- некоторые другие фреймворки, рассмотрение которых отложим до лучших времен.

С целью помочь в решении всех проблем при создании приложения был разработан веб-фреймворк, входящий в состав Spring. Опираясь на шаблон модель – представление – контроллер (Model – View – Controller, MVC), фреймворк Spring MVC помогает строить веб-приложения, столь гибкие и слабо связанные, как сам фреймворк Spring Framework.

Каждый раз, когда пользователь щелкает на ссылке или отправляет форму в веб-браузере, запрос отправляется на работу. Наш запрос работает курьером, перенося информацию из одного места в другое. С момента, когда он покинет браузер, и до момента, когда вернется ответ, запрос сделает несколько остановок, каждый раз сбрасывая часть информации и подбирая что-то взамен [1; 2].

Первой остановкой на пути запроса является DispatcherServlet. Как и большинство веб-фреймворков на языке Java, фреймворк Spring MVC пропускает все входящие запросы через единственный сервлет входного контроллера. Входной контроллер (front controller) является типичным шаблоном проектирования веб-приложений, где единственный сервлет берет на себя ответственность за передачу всех запросов остальным компонентам приложения, выполняющим фактическую их обработку. В Spring MVC входным контроллером является DispatcherServlet.

Задача контроллера Dispatcher Servlet состоит в том, чтобы передать запрос контроллеру Spring MVC. Контроллер – это компонент Spring, обрабатывающий запрос. Но приложение может иметь несколько контроллеров, и входному контроллеру Dispatcher Servlet требуется помощь, чтобы определить, какому контроллеру передать запрос. Поэтому контроллер Dispatcher Servlet консультируется с одним или несколькими механизмами отображения и выясняет, где будет следующая остановка запроса. При принятии решения механизм отображения, в первую очередь, руководствуется адресом URL в запросе.

Как только будет выбран соответствующий контроллер, Dispatcher Servlet отправляет запрос в путь к выбранному контроллеру. Достигнув контроллера, запрос отдаст часть своего груза (информацию, отправленную пользователем) и терпеливо будет ждать, пока контроллер обработает эту информацию. (На самом деле хорошо спроектированный контроллер сам почти не занимается обработкой информации, вместо этого он делегирует ответственность за обработку одному или нескольким служебным объектам.)

В результате работы контроллера часто появляется некоторая информация, которая должна быть передана назад пользователю и отображена в браузере. Эта информация называется моделью. Но отправки обратно необработанной информации недостаточно, перед отправкой ее следует представить в удобном для пользователя формате, обычно в HTML. Для этого информация должна быть передана в одно из представлений, которыми обычно являются JSP.

Последнее, что должен сделать контроллер, – упаковать вместе модель и имя представления для отображения результатов в браузере. Затем он отсылает запрос вместе с моделью и именем представления обратно входному контроллеру Dispatcher Servlet [1].

Методы решения задач. В соответствии с выполняемыми функциями, веб-приложение можно разбить на несколько модулей.

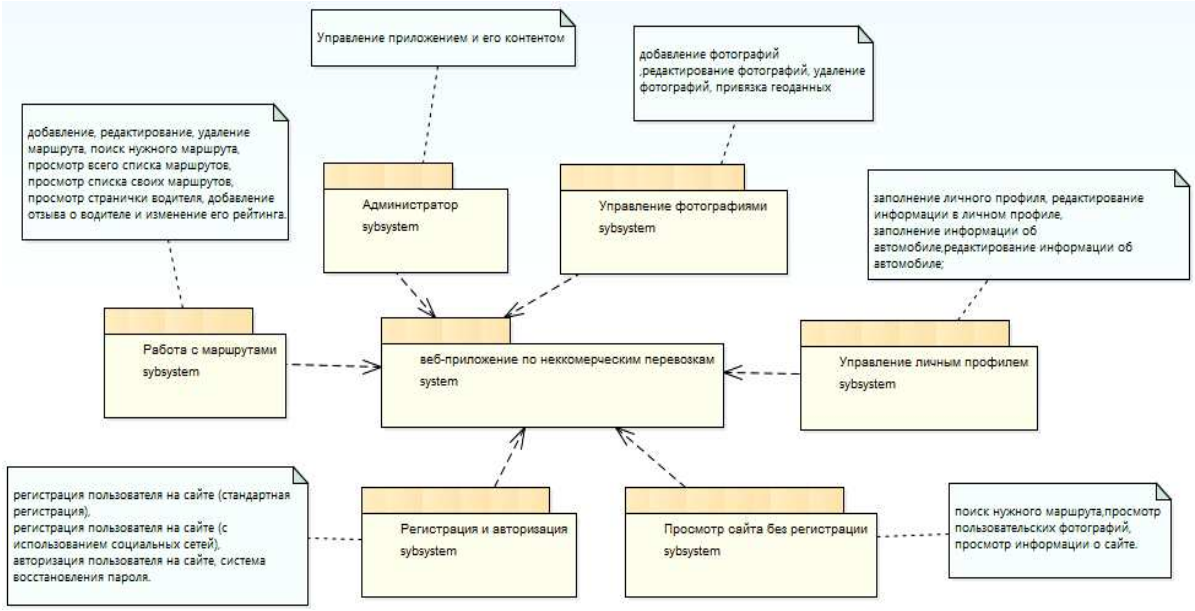
1. Модуль просмотра сайта без регистрации:
   - поиск нужного маршрута;
   - просмотр пользовательских фотографий;
   - просмотр информации о сайте.
2. Модуль регистрации и авторизации:
   - регистрация пользователя на сайте (стандартная регистрация);
   - регистрация пользователя на сайте (с использованием социальных сетей);
   - авторизация пользователя на сайте;
   - система восстановления пароля.
3. Модуль работы с личными данными:
   - заполнение личного профиля;
   - редактирование информации в личном профиле;
   - заполнение информации об автомобиле;
   - редактирование информации об автомобиле;
   - просмотр списка отзывов.
4. Модуль работы с маршрутами:
   - добавление маршрута;
   - редактирование маршрута;
   - удаление маршрута;
   - поиск нужного маршрута;
   - просмотр всего списка маршрутов;
   - просмотр списка своих маршрутов;
   - просмотр странички водителя;
   - добавление отзыва о водителе и изменение его рейтинга.
5. Модуль работы с фотографиями:
   - добавление и загрузка фотографии;
   - редактирование фотографии;
   - удаление фотографии;
   - прикрепление геоданных к фотографии;
   - просмотр всего списка фотографий;
   - просмотр выбранной фотографии.
6. Модуль администратора:
   - все вышеперечисленные функции;
   - удаление отзывов;
   - редактирование отзывов;
   - удаление пользовательских профилей;

Функциональная структура системы представлена на рисунке.

Веб-приложение содержит следующие классы:

- Admin Controller содержит методы, необходимые для работы администратора в приложении;
- Login Controller содержит методы, необходимые для авторизации в приложении;
- Rersonal Controller содержит методы, необходимые для реализации функциональности в личном кабинете пользователя;
- Photo Controller содержит методы, для загрузки, редактирования и удаления фотографий;
- Recalls Controller содержит методы, необходимые для функционирования системы отзывов;
- Registration Controller содержит методы, необходимые для регистрации в приложении;
- Routes Controller содержит методы, необходимые для регистрации в приложении;
- Vk Service Impl содержит методы, необходимые для регистрации в приложении через социальную сеть «Вконтакте»;

- Fb Service Impl содержит методы, необходимые для регистрации в приложении через социальную сеть «Facebook»;
- Fb Service содержит методы и поля, необходимые для получения данных из социальной сети «Facebook»;
- Vk Service содержит методы и поля, необходимые для получения данных из социальной сети «Вконтакте»;
- Fotos Service Impl содержит методы, необходимые для корректного отображения фотографий в приложении;
- Recalls Service Impl содержит методы, необходимые для успешного функционирования системы отзывов и системы рейтинга водителей;
- User Service Impl содержит методы, необходимые для реализации функциональности, отвечающей за деятельность пользователя в приложении;
- Routes Service Impl содержит методы, необходимые для реализации функциональности, отвечающей за корректное отображение информации о маршрутах;
- Valid Email – класс, проверяющий корректность электронной почты пользователя;
- Sec Security Config – класс, содержащий информацию о кодировании пользовательских паролей.



Функциональная структура системы

Заключение. Проектирование и разработка программного средства должны осуществляться с учетом современных технологий, поставленных требований и правил к веб-приложениям. Существенное преимущество построения веб-приложений для поддержки стандартных функций браузера заключается в том, что функции должны выполняться независимо от операционной системы данного клиента. Вместо того чтобы писать различные версии для Microsoft Windows, Mac OS X, GNU/Linux и других операционных систем, приложение создается один раз для произвольно выбранной платформы и на ней разворачивается.

При разработке серверной части веб-приложения необходимо выбирать такие средства и методы, которые позволят в полной мере реализовать необходимый функционал. Использование сервлетов и контейнеров для их хранения, различных фреймворков, таких как Spring и не только, а также разработка веб-приложения по шаблону MVC значительно ускоряет и упрощает процесс разработки серверной части веб-приложения без потери функциональности.

ЛИТЕРАТУРА

1. Википедия – свободная энциклопедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/. – Дата доступа: 23.09.2016.
2. Metanit – сайт о программировании [Электронный ресурс]. – Режим доступа: http://metanit.com. – Дата доступа: 23.09.2016.
3. Hinex – компания, специализирующаяся в веб-разработке и Интернет-рекламе [Электронный ресурс]. – Режим доступа: http://hinex.ru/. – Дата доступа: 23.09.2016.