Разработка веб-интерфейса для сервера уведомлений

Левин Ю. В.

***Аннотация* — В силу быстрого развития мобильного интернета одним из популярных способов распространения контента в сети стали сервисы push-уведомлений. Для каждой мобильной платформы разработаны свои службы push-уведомлений (Apple Push Notification Service (APNS), Google Cloud Messaging for Android (GCM) и др.). Основываясь на данных службах, разрабатываются сервисы рассылок сообщений с более сложной организацией взаимодействия между клиентом и сервером. Например, сервис может предоставлять удобный веб-интерфейс для работы с многоуровневым контентом рассылок, собирать, обрабатывать и отображать статистику о мобильных платформах и их пользователях и т. д.**

***Ключевые слова* —AJAX, Design patterns, JSON, MVC.**

I. Введение

Веб-интерфейс — это совокупность средств, при помощи которых пользователь взаимодействует с веб-сайтом или любым другим приложением через браузер [1]. Основным преимуществом веб-интерфейсов является возможность доступа к ним посредством сети (интернет, VPN-сети и т.д.) используя любой браузер. Например, для редактирования базы данных MySQL широко используется веб-интерфейс phpMyAdmin.

Современные веб-интерфейсы все чаще используют подход, основанный на обновлении лишь той части интерфейса, которая требует обновления. Этот подход получил название AJAX (Asynchronous JavaScript and XML). Частичное обновление делает интерфейс более быстрым и интерактивным. Впервые термин AJAX был употреблен в статье «Ajax: A New Approach to Web Application». Особенно популярен данный подход стал после использования его компанией Google в своих сервисах (Gmail, Google Maps и др.). Изначально, в подходе AJAX применялся XML, но из-за больших издержек связанных с этим форматом (при работе с данными небольшого объема и простой структуры доля разметки в общем объеме достаточно велика) сегодня все чаще используется формат JSON (JavaScript Object Notation). JSON - это текстовый формат обмена данными. С его помощью можно описать любую структуру данных. Кроме этого, доля разметки JSON в общем объеме данных мала [2].

Левин Юрий Владимирович – студент 3-го курса Второго Высшего Образования факультета вычислительной математики и кибернетики Московского Государственного Университета имени М. В. Ломоносова.

Для решения задач проектирования используют шаблоны (*design patterns*). Впервые понятие шаблон было введено Кристофером Александером для отображения наиболее распространенных архитектурных решений. Согласно его определению шаблон это: «…периодически возникающая задача и описание основных принципов ее решения…». Позднее этот термин применили и к программному обеспечению, а через несколько лет сфера применения данного термина расширилась за счет области проектирования пользовательских интерфейсов [3].

При проектировании веб-интерфейсов в качестве архитектурного решения часто применяется шаблон Model-View-Controller (MVC). Он впервые был описан в 1979 году Трюгве Реенскауг при разработке языка Smalltalk [4]. MVC подразумевает разделение всей программы на три независимых объекта:

- Модель - предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
- Представление (вид) - отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма с графическими элементами.
- Контроллер - обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

Важной особенностью данного подхода является невозможность прямого взаимодействия модели и представления. Любое действие пользователя, обрабатывается контроллером, который в свою очередь взаимодействует с моделью.

Схема MVC разделяет эти объекты в пользовательских интерфейсах, что повышает гибкость и повторное использование программы. MVC отделяет вид от модели, устанавливая между ними протокол взаимодействия. Вид должен гарантировать, что внешнее представление отражает состояние модели. При изменении внутренних данных, модель оповещает все зависящие от нее виды, в результате чего, вид обновляет себя. Это позволяет создавать несколько видов для модели, не изменяя ее [5].

Целью данной статьи является проектирование архитектуры легко масштабируемого веб-интерфейса

сервера уведомлений, а также описание основных протоколов взаимодействия между объектами.

II. Разработка веб-интерфейса

A. *Представление*

Представление веб-интерфейса реализуется как HTML страница (доступная в браузере), построенная с помощью технологии Java Server Pages. JSP строит каркас веб-интерфейса, а вся необходимая информация подгружается с помощью AJAX запросов. Более подробно AJAX запросы рассмотрены далее в пункте «Контроллер» данной главы.

Представление веб-интерфейса состоит из следующих частей:

- Авторизация клиента - HTML форма для ввода уникальной пары логин/пароль.
- Основной вид компаний рассылки - содержит все компании и все темы заказчика, все управляющие элементы для создания и редактирования компаний, тем, сообщений и для отображения статистических данных.
- Система оверлеев - используется для добавления и редактирования компаний, тем и сообщений, а также для отображения статистики. Оверлей (overlay) – это облегченная версия всплывающего окна. Основными отличиями от классических всплывающих окон являются экономия ресурсов компьютера и отсутствие лишних навигационных элементов всплывающих окон. Благодаря возможностям технологии AJAX оверлей можно разместить прямо в основном виде компаний рассылки [5].

Основной вид компаний рассылки представлен на рис. 1.

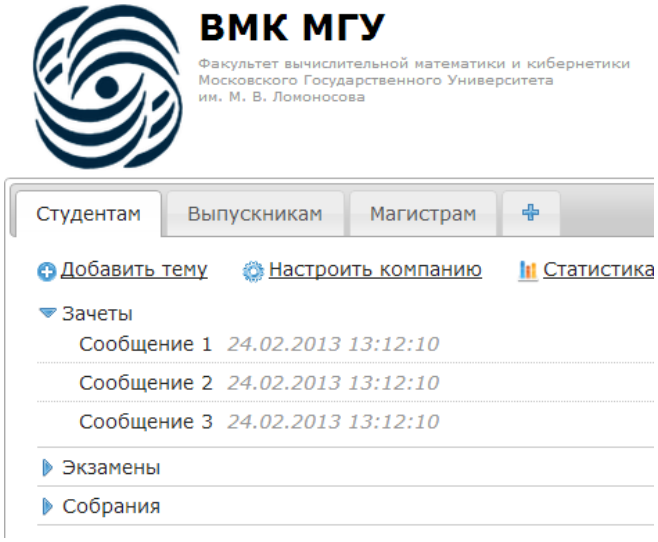

Рис. 1.

При первом попадании клиента на страницу веб-интерфейса, система показывает вид авторизации. Пользователь заполняет HTML форму (логин/пароль) и отправляет ее. Система, проверив логин/пароль, либо разрешает доступ к веб-интерфейсу, показывая основной вид компаний, либо запрещает доступ перенаправляя на вид авторизации клиента. В случае разрешения доступа между клиентом (браузером) и сервером создается сессия, действующая фиксированное время.

В качестве примера заказчиком выбран факультет ВМК МГУ, который ведет различные рассылки по следующим компаниям: студенты, выпускники и магистры. У каждой компании есть темы (здесь показаны темы компании «Студентам»): зачеты, экзамены и собрания. В каждой теме содержатся сообщения рассылки.

В рамках проектируемого веб-интерфейса оверлеи используются для добавления, редактирования контента и для отображения статистики по компаниям/группам. Оверлеи для добавления и редактирования представляют собой HTML формы. Оверлеи для отображения статистики имеют более сложную структуру, состоящую из диаграмм и графиков. Они генерируются с помощью JavaScript. На рис. 2 представлен пример оверлея для добавления темы.

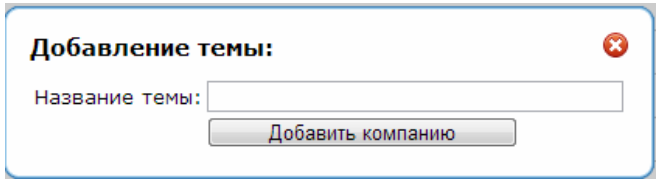

Рис. 2.

При нажатии на управляющий элемент добавления или редактирования контента (на рис. 1 это ссылки «Добавить тему» и «Настроить компанию») срабатывает JavaScript код, строящий оверлей и выполняющий AJAX запрос. Каркас веб-интерфейса представляет собой динамически сгенерированную с помощью JSP HTML страницу, содержащую минимум контента. Минимизирование контента каркаса интерфейса делает его построение менее ресурсоемким, а загрузку более быстрой. Кроме того, это позволяет, при необходимости, изменить формат выдачи контента, поменяв лишь ответы сервера на AJAX запросы. При этом представление не меняется. Ниже представлен приблизительный HTML код представления.

```
<html>
<head>
<!-- ... -->
<link type="text/css" rel="stylesheet" href="style.css">
    <!-- javascript -->
</head>
<body>
<img id="user-logo" class="logo" alt=""
    src="/img/no-photo.png"/>
<div class="client-name">
</div>
<div class="clear"></div>
<div id="tabs">
</div>
</body>
</html>
```

Рис. 3.

HTML код, показанный на рис. 3, строится средствами JSP. В разделе body созданы: логотип заказчика (img id=”user-logo”), название заказчика и дополнительная информация (блок client-name), блок контента (div id=”tabs”).

Дальнейшее построение страницы возлагается на JavaScript и AJAX. После загрузки страницы в браузере срабатывает JavaScript код, блокирующий доступ пользователя к странице веб-интерфейса и реализующий AJAX запросы к контроллеру на предоставление данных о пользователе (блоки “user-logo”, “client-name”) и данных о его компаниях и темах (блок “tabs”). Контроллер возвращает запрошенную информацию адресанту. Затем информация, полученная с помощью JavaScript, обрабатывается и записывается в соответствующие блоки на странице веб-интерфейса (информация о компаниях записывается в блок “tabs” и др.). После этого веб-интерфейс становится заполненным актуальной информацией, а пользователь получает доступ к управляющим элементам. Остановимся более подробно на обработке полученного ответа средствами JavaScript. JavaScript код получает ответ от сервера в формате JSON. Далее для преобразования данных формата JSON в объект JavaScript используется функция eval. Функция eval, в качестве единственного параметра, принимает строку и преобразует данную строку в объект. Например, eval(“{"result":true,"count":1}”); создаст 2 объекта: result со значением true и count равное 1.

Управляющие элементы – это элементы интерфейса, с помощью которых происходит взаимодействие с веб-интерфейсом. Как правило, это HTML ссылки или кнопки отправки форм. Любое нажатие на управляющий элемент запускает соответствующий JavaScript код.

Для основного вида компании определены следующие управляющие элементы:

- добавить компанию;
- настроить компанию;
- добавить тему;
- редактировать тему;
- добавить сообщение;
- редактировать сообщение;
- статистика по компании;
- статистика по теме;
- кнопки отправки форм.

Обработка управляющего элемента требует отправки AJAX запроса и внесение изменений в содержимое страницы согласно полученному ответу на данный запрос. Для обработки взаимодействия с управляющими элементами (для всех, кроме кнопок отправки форм) из содержимого ответа строится оверлей.

Для повышения скорости работы с веб-интерфейсом используются клавиши быстрого доступа. Это разновидность взаимодействия со страницей веб-интерфейса представляющая собой нажатие сочетаний клавиш на клавиатуре, которым назначено некое действие — команды, исполняемые веб-интерфейсом [6]. Клавиши быстрого доступа дублируют функции веб-интерфейса. Ниже представлен список клавиш быстрого доступа с описанием функционала:

- Ctrl + Shift + C – дублирует функцию «Добавить компанию»;
- Ctrl + Shift + T – дублирует функцию «Добавить темы»;
- Ctrl + Shift + E – дублирует функцию «Настроить компании»;
- Ctrl + Shift + S – дублирует функцию «Статистика по компании».

Кроме клавиш быстрого доступа, для редактирования текстовой информации применяется шаблон непосредственного редактирования (in-page editing) [3]. Данный шаблон проектирования применяется для быстрого редактирования текстовой информации прямо на странице, что обеспечивает более быструю работу с интерфейсом. Непосредственное редактирование текста применимо для редактирования названия заказчика, его описания, названия тем и названия сообщений. При двойном нажатии на текстовую строку срабатывает JavaScript код, переводящий строку в режим редактирования. То есть строка заменяется полем для ввода, заполненным редактируемым текстом. Ниже этого поля показываются две кнопки: «Сохранить» и «Отменить». При нажатии на кнопку «Сохранить» происходит сохранение текста (посылается AJAX-запрос на редактирование информации), а при нажатии на «Отмена» - отмена редактирования. Поскольку редактирование текстовой информации возможно через управляющие элементы, описанные выше, шаблон непосредственного редактирования является дублирующим. Непосредственное редактирование имеет несколько недостатков. Самым весомым из них является то, что данный шаблон является незаметным в веб-интерфейсе. Для того, чтобы пользователь узнал об этой возможности, можно использовать одну из следующих подсказок:

- всплывающая при наведении на строку текстовая подсказка;
- подсветка области с возможностью редактирования;
- превращение указателя мыши в текстовый курсор.

Но это также не гарантирует то, что пользователь поймет как использовать данную функцию.

B. Модель

В рамках веб-интерфейса модель представляет собой базу данных, хранящую в себе всю необходимую информацию. Перечислим основные данные, хранящиеся в базе данных, в рамках веб-интерфейса:

- Данные необходимые для аутентификации пользователей в системе веб-интерфейса. Каждому пользователю (заказчику рассылки) присваивается уникальная пара

логин/пароль, которая однозначно определяет его в веб-интерфейсе. Заметим, что для каждого заказчика может быть создано несколько уникальных пар логин/пароль с различными правами доступа к функциям интерфейса.

- Дополнительная информация о заказчике – хранит данные, необходимые для визуализации веб-интерфейса. Например, логотип, название, описание проекта заказчика.
- Сообщения - содержит текст рассылаемого сообщения.
- Темы рассылок - разделяют рассылаемые сообщения на темы. Служит для логической группировки всех сообщений в группы сообщений. Кроме этого, на эти темы подписывается конечный получатель рассылок.
- Компании рассылок. У каждого заказчика могут быть несколько компаний рассылок, каждая из которых содержит подгруппы - темы рассылок. Компания – логическое объединение нескольких тем рассылок в группы по какому-то критерию.
- Графики рассылки сообщений. Для тем рассылок предусмотрена возможность отправки сообщения в определенные дни недели и временной интервал (по будням, с 9 до 18).
- Статистические данные. В базе данных хранится собранная информация о подписанных на различные темы пользователях, статистические данные по темам и компаниям (сколько клиентов подписаны на данную тему, аудитория темы/компании и т.д.).

В веб-интерфейсе для сервера уведомлений в качестве базы данных используется объектно-реляционная система управления базами данных PostgreSQL. Данный выбор обусловлен надежностью данной СУБД. По результатам автоматизированного исследования в исходном коде СУБД PostgreSQL выявлено 20 проблемных мест на 775000 строк (примерно одна ошибка на 39000 строк кода) [7].

Для взаимодействия контроллера с моделью используется промышленный стандарт взаимодействия Java приложений с СУБД, называемый JDBC (Java Data Base Connectivity — соединение с базами данных на Java) [8].

C. Контроллер

Веб-интерфейс в качестве контроллера использует набор функций для работы с базой данных. Каждая функция определяется уникальным URI (Uniform Resource Identifier – унифицированный идентификатор ресурса). В качестве входных параметров функция получает данные по HTTP протоколу. Данные передаются с помощью метода GET или POST. Таким образом, на языке Java данный набор функций может быть реализован с помощью интерфейса сервлетов (servltes). Взаимодействие представления и контроллера осуществляется с помощью механизма асинхронных запросов (AJAX). В качестве формата обмена данных в AJAX применяется форма JSON. Ответ контроллера состоит из двух переменных: success – код результата выполнения операции и html – HTML разметка. Код результата выполнения операции содержит значение 0 – если операция успешно выполнена и код ошибки в противном случае. Если в ходе выполнения операции произошла ошибка, в переменную html записывается сообщение об этой ошибке. Например, {“success”: 1; “html”:”Неверный формат запроса.”}.

Все необходимые для правильной работы функции параметры проверяются (проходят валидацию). Например, проверяется наличие всех обязательных параметров и правильность их заполнения. При отсутствии ошибок функция, основываясь на параметрах, формирует ответ.

Так как функция однозначно определяется своим URI, будем обозначать функцию ее адресом. Ниже представлены основные функции контроллера с описанием параметров (символом * помечены обязательные для заполнения параметры). Перед названием функции стоит метод, указываемый в HTTP пакете.

GET company – функция получения компаний конкретного пользователя. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации.

POST company – функция для добавления, редактирования. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации;
- company_id – уникальный номер компании в базе;
- company_name* – имя компании;
- company_descr – описание компании;
- company_image – логотип компании;
- company_person – лицо, отвечающее за данную компанию (служебная информация);
- company_phone – контактный телефон.

В зависимости от установленного метода протокола HTTP (GET или POST) контроллер формирует различный ответ. Если установлен метод GET, функция формирует HTML код для компаний и тем данного пользователя. Если POST, то, в случае успешного выполнения запроса, в переменную html запишется сообщение «Компания успешно добавлена». Если заполнен параметр company_id, происходит редактирование компании. Однако, если компании с таким значением company_id не существует, то формируется сообщение об ошибке.

GET theme – функция получения тем для данной компании. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации;
- theme_id* – уникальный номер темы в базе.

POST theme – функция для добавления, редактирования тем. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации;
- theme_id* – уникальный номер темы в базе;
- theme_name* – имя темы.

GET message – функция для получения сообщений для данной темы. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации;
- theme_id – уникальный номер темы в базе данных, которой принадлежит сообщение.

POST message – функция для добавления, редактирования сообщения. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации;
- message_id* – уникальный номер сообщения в базе данных, которой принадлежит сообщение;
- message_name* – имя компании;
- message_text* –текст сообщения.

Функция действует по тому же принципу, что и функция company.

GET stats/company – функция для получения статистики по компании. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации;
- company_id* – уникальный номер компании в базе.

GET stats/theme – функция для получения статистики по теме. Принимает параметры:

- user_id* – уникальный номер пользователя, выданный при успешной авторизации;
- theme_id* – уникальный номер темы в базе.

Функция company и message использует метод в HTTP пакете как один из параметров запроса. В остальных функциях метод HTTP запроса не используется и может быть как POST, так и GET.

III. Заключение

В данной статье была рассмотрена архитектура веб-интерфейса сервера уведомлений. В основу архитектуры был заложен подход Model View Controller (MVC). Важной особенностью данного подхода – является отсутствие прямой связи между моделью и представлением. Модель связана с контроллером с помощью промышленного стандарта JDBC. Представление связано с контроллером с помощью подхода AJAX, используя формат обмена данными JSON. Данный формат по сравнению с XML содержит значительно меньшую долю разметки, что уменьшает размер пересылаемых сообщений. Меньший размер пересылаемых сообщений требует меньшего времени загрузки, что увеличивает скорость работы веб-интерфейса. Также для ускорения работы интерфейса используются шаблоны проектирования. Оверлеи позволяют ускорить процесс построения всплывающих окон. Из-за дополнительных элементов интерфейса всплывающие окна расходуют больше памяти и процессорного времени. Оверлеи строятся прямо на странице без лишних элементов интерфейса.

Также для проектирования быстрого веб-интерфейса были применены шаблон непосредственного редактирования и клавиши быстрого доступа. Оба метода дублируют основной функционал, но позволяют ускорить взаимодействие пользователя с интерфейсом. Непосредственное редактирование текста позволяет пользователю оставаться в контексте страницы.

В качестве контроллера был спроектирован набор функций, реализующий методы взаимодействия с базой данных. Используя язык программирования Java функции контроллера реализуются через интерфейс сервлетов (Servltes).

Далее полученную информацию планируется использовать для написания веб-интерфейса сервера уведомлений. Также планируется развитие данной архитектуры и внедрение API для внешних приложений, таких как приложение для мобильных платформ. Это позволит сделать мобильный веб-интерфейс подписки на темы рассылок.

Библиография

[1] http://ru.wikipedia.org/wiki/%C2%E5%E1-%E8%ED%F2%E5%F0%F4%E5%E9%F1
[2] http://ru.wikipedia.org/wiki/JSON
[3] Б. Скотт, Т. Нейл Проектирование веб-интерфейсов, - Пер. С англ. – Спб.: Символ-Плюс, 2010 г.352с., ил.
[4] http://ru.wikipedia.org/wiki/Model-View-Controller
[5] Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования, Питер, 2007 г.
[6] http://ru.wikipedia.org/wiki/%D1%EE%F7%E5%F2%E0%ED%E8%E5_%EA%EB%E0%E2%E8%F8
[7] http://www.postgresql.org/about/news/363/
[8] http://ru.wikipedia.org/wiki/Java_Database_Connectivity
[9] http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.HTML
[10] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications
[11] http://java.cnam.fr/iagl/biblio/spec/jdbc-3_0-fr-spec.pdf