УДК 004.457

ПОДХОДЫ К РАЗРАБОТКЕ HTTP-СЕРВЕРА ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ АНДРОИД С НЕБОЛЬШИМ ПОТРЕБЛЕНИЕМ РЕСУРСОВ

А.А. ЮШКЕВИЧ
(Представлено: Е.Р. СУХАРЕВ)

Рассматривается подход к проектированию серверной части андроид-приложения для взаимодействия с компьютером по протоколу HTTP. Рассмотрена легковесная библиотека NanoHTTPD для создания HTTP-сервера, а также описаны важные моменты проектирования архитектуры приложения.

В настоящее время компьютеры получили широкое распространение. Причём у большинства людей есть как персональные компьютеры или ноутбуки, так и различные смартфоны, планшеты, использующие операционную систему Android. И здесь появляется вопрос, как можно легко осуществить какое-либо взаимодействие (передать файлы, экспортировать контакты и т.д.) между различными устройствами. В данной ситуации существует огромное множество способов, которыми можно решить поставленную задачу. Например, данные можно передать с помощью технологии bluetooth, которую поддерживает большинство современных смартфонов. Также необходимая информация может быть перенесена с помощью специального кабеля. Такой способ передачи обеспечит максимальную скорость.

Ещё файлы можно переписать с помощью протокола FTP или специальных программ, которые включают в себя клиентскую и серверную часть.

Все эти способы способны решить поставленную задачу. Однако под рукой может оказаться кабель с неподходящим разъёмом, bluetooth слишком медленный, а с помощью FTP можно передавать только файлы и при попытке расширения функционала можно столкнуться с проблемами. Поэтому стоит задуматься об универсальном аналоге, который сможет решать задачу быстро, без дополнительного оборудования и помимо передачи файлов выполнять другие функции.

Средства решения задач. Широкое распространение таких приложений, как браузеры, позволяет говорить о веб-приложениях как о кроссплатформенных. Например, данный подход можно использовать с целью создания интерфейса между компьютером и мобильным телефоном на базе андроид. Однако в связи с тем, что на телефоне ресурс существенно ограничен, попытка переноса полноценного сервера не всегда может оказаться эффективной. Поэтому в данной работе описана такая небольшая библиотека, как NanoHTTPD. Её главным отличием от многих других является минимализм, что важно при разработке под мобильные платформы. Библиотека распространяется в виде открытых исходных кодов. Причём лицензия позволяет её свободно изменять. Это означает, что она может быть оптимизирована под какую-либо конкретную задачу, что может стать огромным плюсом при разработке под устройство, обладающее небольшим количеством ресурсов.

Для использования библиотеки, можно воспользоваться исходными кодами с github, которые распространяются согласно модифицированной лицензии BSD.

Чтобы запустить сам сервер, необходимо создать класс и унаследовать его от NanoHTTPD и вызвать метод start. Для обработки запросов, необходимо переопределить метод *public Response serve (IHTTPSession session).*

IHTTPSession – интерфейс, который включает в себя методы для работы с cookie, получения параметров запроса, заголовка запроса. Также присутствует метод для разбора тела запроса. В библиотеке уже есть готовая реализация данного интерфейса, которая позволяет передавать и разбирать заголовки, параметры. Причём последние могут быть как в закодированном, так и в раскодированном виде. Кроме этого присутствует возможность передавать файлы.

При обработке POST-запросов следует учесть, что разбор параметров происходит не автоматически, а в результате явного выхода метода *void parse Body (Map<String, String> files) throws IOException, Response Exception.*

Как можно было заметить, метод serve возвращает такой класс, как Response. Он предназначен для передачи ответа клиенту и может быть создан различными способами, в зависимости от передаваемого содержимого. Чтобы создать класс, можно воспользоваться одним из его статических методов. Например, если необходимо передать поток данных указанного типа, можно воспользоваться методом *static Response new Chunk end Response (IStatus status, String mime Type, Input Stream data).* Это может оказаться эффективным для передачи файлов или иных данных неизвестного размера. Если последний известен,

можно воспользоваться методом *static Response new Fixed Length Response (IStatus status, String mime Type, byte [] data)* для передачи массива байтов, либо *static Response new Fixed Length Response (IStatus status, String mimeType, Input Stream data, long total Bytes)* для пересылки данных из потока.

Если необходимо переслать строку, можно воспользоваться методом *static Response new Fixed Length Response (IStatus status, String mime Type, String txt)* для передачи данных произвольного типа, либо *static Response new Fixed Length Response (String msg)* для отправки информации, которая будет интерпретирована как html-содержимое.

Из важных особенностей NanoHTTPD следует отметить, что данная технология также позволяет работать с cookie. Для того чтобы их получить, необходимо вызвать метод *CookieHandler get Cookies()* у класса, реализующего интерфейс IHTTPSession. Как видно, метод возвращает CookieHandler, который реализует интерфейс Iterable, а сделовательно все куки можно проитерировать с помощью конструкции foreach. Но итерироваться будут имена. Чтобы прочитать значение, необходимо воспользоваться методом *String read (String name).* Для удаления используется метод *void delete (String name),* а для добавления или изменения значения, метод *void set (Cookie cookie),* либо *void set (String name, String value, int expires)* [1].

Кроме описанных возможностей, библиотека может также создавать соединение как по протоколу HTTP, так и по защищенному HTTPS [2].

Во избежание переполнения оперативной памяти, запросы большого размера кэшируются в память устройства. При передаче файлов в виде multipart data запрос кэшируется в память устройства, затем выполняется его разбор, после чего формируется временный файл, который позже можно переместить туда, где он нужен.

Этот момент реализации может оказаться не совсем удачным при использовании мобильной операционной системы потому, что память на устройстве ограничена и создавать много копий файлов не стоит. В связи с этим при реализации мобильного приложения было решено внести изменения в исходный код библиотеки. В частности, модифицирован код, касающийся загрузки и разбора запроса. Теперь файл вместо того, чтобы записываться во временные, записывается сразу по необходимому адресу.

Методы решения задач. Очевидно, что при реализации приложения важно сформировать некоторую архитектуру. Здесь отлично подходит модель MVC. Она представляет собой 3 элемента: модель, представление и контроллер [3].

Модель является некоторыми данными, которые описывают предметную область и над ними могут быть выполнены некоторые операции. Также эта информация может быть предоставлена пользователю на графическом интерфейсе.

Представление является этим самым графическим интерфейсом.

Контроллер реализует некоторые алгоритмы для получения данных с представления, обработки их и отправки обратно.

Для работы с моделью удобным может оказаться создание некоторого базового класса или интерфейса, что и было сделано при разработке андроид-приложения. В классы, выполняющие функции моделей, также были помещены методы для работы с JSON [4]. Таким образом, JSON-содержимое может быть легко преобразовано в класс-модель и наоборот.

Для реализации представления используется язык разметки html, содержимое на котором генерируется с помощью javascript и ajax [5].

Учитывая, что для каждой реализуемой функции приложения необходимо разработать свой собственный контроллер, необходимо создать фабрику контроллеров. Входным параметром может быть URI-запроса. Таким образом, контроллер – это класс, наследующий некоторый базовый класс, отвечающий за контроллеры. И все классы-контроллеры лежат в коллекции Map, ключами которой являются URI.

Однако в сами контроллеры помещать весь функционал не стоит, потому что это может привести к загромождению кода, а следовательно и к ухудшению его читабельности. Поэтому при разработке приложения был выделен уровень сервисов, который должен отвечать за выполнение операций на телефоне, таких как получение сведений, модификация и др. Контроллеры же только преобразуют данные для взаимодействия с клиентом и вызывают необходимый метод из сервиса [6].

Исходя из всего сказанного, краткий алгоритм работы разработанного алгоритма работы приложения можно описать следующим образом:

- в методе NanoHTTPD.serve захватывается запрос;
- для get-запроса выполняется попытка найти и отправить подходящий файл;
- для всех остальных запросов выполняется поиск контроллера по URI;
- передача запроса в контроллер;
- разбор запроса;
- преобразование данных из JSON в класс модель;

- выполнение действия, за которое отвечает контроллер;
- преобразование выходных данных из класса-модели в формат JSON;
- создание ответа;
- отправка ответа клиенту.

Заключение. Библиотека NanoHTTPD зарекомендовала себя как минималистичная, легковесная, экономная в плане расходования ресурсов устройства, что важно при разработке программных продуктов для мобильных устройств. К тому же объектно-ориентированный подход к программированию может существенно упростить процесс разработки. В частности, он позволяет избегать повторов в коде. В разработанном приложении это видно в механизмах работы с моделями и контроллерами. Кроме всего прочего, улучшить читабельность кода, а значит облегчить разработку и исправление ошибок, позволяет использование модели MVC, а также паттерна service layer.

ЛИТЕРАТУРА

1. NanoHttpd/nanohttpd: Tiny, easily embeddable HTTP server in Java / Jarno Elonen [Electronic resource]. – 2010. – Mode of access: https://github.com/NanoHttpd/nanohttpd. – Date of access: 21.09.2016.
2. Rescorla, E. HTTP Over TLS / E. Rescorla // RTFM Inc. [Electronic resource]. – 2010. – Mode of access: https://tools. ietf.org/ html/rfc7159. – Date of access: 21.09.2016.
3. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер. – Москва, Санкт-Петербург, Киев, 2006. – 544 с.
4. Bray, T. The JavaScript Object Notation (JSON) Data Interchange Format / Bray T. // Google Inc. [Electronic resource]. – 2014. – Mode of access: https://tools.ietf.org/html/rfc7159. – Date of access: 21.09.2016.
5. Мархвид, И.В. Создание WEB-страниц: HTML, CSS, JavaScript / Мархвид И.В. – Минск : ООО «Новое знание», 2002. – 350 с.
6. Хоп, Г. Шаблоны интеграции корпоративных приложений / Г. Хоп, Б.Б. Вульф. – М. : Вильямс, 2006. – 672 с.