Сергеев Павел Андреевич
студент
ФГБОУ ВО «Государственный университет морского
и речного флота им. адмирала С.О. Макарова»
г. Санкт-Петербург

ИСПОЛЬЗОВАНИЕ JSON API В ANDROID-ПРИЛОЖЕНИИ

Аннотация*: статья посвящена использованию JSON API на основе Android-приложения по мониторингу рынка криптовалют «CoinData». Рассматриваются также общие аспекты работы с форматом JSON и сервисами API. Используемый при разработке язык программирования – Java.*

Ключевые слова*: JSON, API, Java, разработка Android-приложений.*

Введение. Огромное количество мобильных приложений, которыми люди пользуются каждый день, используют в своей работе сеть Интернет для получения или передачи необходимой информации посредством обмена данными с серверами. Данная архитектура называется клиент-серверной, где обе стороны диалога представляют собой программное обеспечение, обменивающееся сообщениями через вычислительную сеть посредством сетевых протоколов.

В данной статье будет рассматриваться получение информации стороной клиента (мобильным приложением) средствами JSON от сервисов (серверов), которые предоставляют данные через взаимодействие с собственными API.

JSON (JavaScript Object Notation) – формат обмена данными, который имеет вид понятного человеческому глазу текста. В основу JSON были положены две структуры данных: собрание пар имя/значение (во многих языках понимается как объект) и последовательный список значений (во многих языках понимается как массив) [1]. Пример сообщения в формате JSON представлен на рисунке 1.

```
{
  "ticker": {
    "base": "BTC",
    "target": "USD",
    "price": "5328.20307033",
    "volume": "51671.65780775",
    "change": "3.80047258"
  },
  "timestamp": 1556649004,
  "success": true,
  "error": ""
}
```


Рис. 1

Данное сообщение представляет собой объект JSON, который содержит 4 предмета – 1 объект («ticker», который в свою очередь содержит 5 объектов) и 3 пары имя/значение («timestamp», «success», «error»).

API («Application programming interface», или «интерфейс прикладного программирования») – совокупность подпрограмм, коммуникационных протоколов и инструментов для создания программного обеспечения. В целом, это набор четко определенных методов связи между различными компонентами [2].

API разнятся в зависимости от способа их применения. В данной статье рассматривается использование Web API, при котором используется протокол HTTP (Hypertext Transfer Protocol) и запросы обрабатываются при обращении по конкретному адресу URL (Uniform Resource Locator) в сети Интернет.

Выбор API. В зависимости от того, какая информация требуется клиенту, необходимо выбрать подходящий Web API. Формат клиент-серверной связи по Web API происходит единообразно вне зависимости от тематики сервисов.

Данная статья написана на основе созданного приложения по мониторингу рынка криптовалют «CoinData» (https://play.google.com/store/apps/details?id=app.pavel.coindata), которое получает информацию от бесплатных API следующих интернет-порталов: www.coinlore.com, www.coingecko.com, www.cryptonator.com, www.cryptocompare.com, www.hitbtc.com. Криптовалюта – разновидность цифровой валюты, создание и контроль за которой базируется на криптографических методах [3].

Процесс отправки запроса и получения ответа от сервера

Описанные ниже процессы работы с получаемыми и обрабатываемыми данными располагаются в разных Java классах для упрощения прочтения программного кода.

Перед отправкой запроса выбранному сервису необходимо правильно сформировать строку URL (Uniform Resource Locator).

В языке программирования Java существует отдельный класс URL для создания данных объектов и взаимодействия с ними.

Сначала формируется строка запроса, значение которой передается объекту класса URL. Затем, из вызова метода «openConnection» с объекта класса URL, создается экземпляр класса HttpURLConnection, который может быть использован для единичного запроса по протоколу HTTP.

Далее создается объект класса BufferedReader, отвечающий за чтение поступающего потока информации, который берет свое значение из ранее созданного экземпляра класса HttpURLConnection с применением метода «getInputStream» через создание объекта класса InputStreamReader, который байты входящего потока данных переводит в символы (буквы, цифры, специальные символы etc.).

Затем значение потока входящей информации построчно записываются в объект класса StringBuffer (объект данного класса автоматически расширяем).

Далее, когда поток входящей информации прочитывается до конца, к объекту класса InputStreamReader применяется метод «close», который говорит о завершении работы с потоком входящей информации и высвобождении системных ресурсов, затраченных на работу с данным потоком.

Для иллюстрации процесса отправки запроса и получения ответа от сервера представлен рисунок 2, на котором изображен код, выполняющий вышеописанные действия.

```
URL url = new URL(String.format(OPEN_COIN_API));
HttpURLConnection connection = (HttpURLConnection)url.openConnection();

BufferedReader reader = new BufferedReader(
        new InputStreamReader(connection.getInputStream()));

StringBuffer json = new StringBuffer(1024);
String tmp = "";
while ((tmp = reader.readLine()) != null)
    json.append(tmp).append("\n");
reader.close();

JSONObject data = new JSONObject(json.toString());
```

Рис. 2

Процесс обработки полученного от сервера ответа

Полученный и правильно прочитанный ответ от сервера необходимо обработать для дальнейшего использования.

Как было сказано выше, JSON формат предполагает две структуры хранения данных: объект и массив.

Обработка исключений в работе с данными в формате JSON

Исключения в Java представляют собой в общем виде ошибку, возникающую во время выполнения программы, выражающуюся в остановке программы и выводе сообщения об исключении в консоли среды разработки или лог-файлах исполнения устройства. Все исключения существуют в виде объектов, то есть могут быть созданы в исполняемом коде. Варианты возникновения исключений различны, например, деление целочисленной переменной на 0.

При работе с исключениями используются следующие ключевые слова: try, catch, throw, throws, finally.

Слова try, catch и finally формируют структуру, состоящую из нескольких блоков, позволяющую обрабатывать исключения. В блок try помещаются те операторы программы, которые необходимо отследить на появление исключений. Исполняемый программный код может перехватить исключение в блоке catch и обработать. Код, который должен быть выполнен после блока try помещается в блок finally.

Слово throw используется для передачи исключения вручную. Слово throws используется для обозначения исключения, которое может быть создано или

передано внутри какого-либо метода. Данное слово должно быть указано в интерфейсе данного метода.

При обработке полученных данных в формате JSON возникает возможность выброса виртуальной машиной Java (JVM) исключения JSONException. К основным вариантам возникновения данного исключения относятся [5]:

– попытка изъять для обработки искаженные документы;

– использование слова «null» в качестве имени;

– использование числовых типов (значений), недоступных для использования в формате JSON, таких как NaN или Infinite;

– поиск объекта с использованием индекса, находящегося вне допустимого диапазона или поиска объекта с именем, несуществующем в исследуемой структуре;

– несоответствие типов при поиске.

Список литературы

1. Introducing JSON [Электронный ресурс]. – Режим доступа: https://www.json.org/

2. API-fication [Электронный ресурс]. – Режим доступа: www.hcltech.com

3. Мащенко П.Л. Технология Блокчейн и ее практическое применение / П.Л. Мащенко, М.О. Пилипенко // Наука, техника, образование. – Олимп, 2017. – №32. – С. 61–64.

4. API specification for the Java™ Platform, Standard Edition [Электронный ресурс]. – Режим доступа: https://docs.oracle.com

5. Documentation for app developers [Электронный ресурс]. – Режим доступа: https://developer.android.com/