УДК 004

Прогноз метеоэлементов с использованием Java

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Глаголев Владимир Александрович
Приамурский государственный университет имени Шолом-Алейхема
К.г.н., доцент, доцент кафедры информационных систем, математики и правовой информатики

Аннотация
Данная статья содержит материал для реализации парсинга данных метеоэлементов с web-ресурса visualcrossing. Программа написана на языке программирования Java, с использованием фреймворка Spring Boot. Результатом исследования станет готовый алгоритм для получения данных метеоэлементов за любой период времени.
Ключевые слова: Java, Spring Boot, погода.

Weather element forecast using Java

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Erovleva Regina Victorovna
Sholom-Aleichem Priamursky State University
Student

Glagolev Vladimir Aleksandrovich
Sholom-Aleichem Priamursky State University
Ph.D, Associate Professor, Associate Professor of the Department of Information Systems, Mathematics and Legal Informatics

Abstract
This article contains material for the implementation of weather element data parsing from the visualcrossing web resource. The program is written in the Java programming language using the Spring Boot framework. The result of the study will be a ready-made algorithm for obtaining weather element data for any period.

Keywords: Java, Spring Boot, weather.

1 Введение

1.1 Актуальность

В настоящее время повышение эффективности прогнозирования метеорологических условий остаётся актуальной задачей для науки. Данные о погоде и прогнозирование имеют существенное значение в ряде отраслей: экономика, авиация, строительство, сельское хозяйство и др., так как планирование и проведение различных видов мероприятий и работ во многом зависит от погодных условий.

1.2Обзор исследований

М.А. Потовиченко, М.В. Привалов и С.В. Корнев рассмотрели разработку программного продукта, который обеспечивает учет данных посещения занятий студентов, а также защиту их работ [1]. А.Д. Нарижный и Н.Е. Губенко провели сравнительный анализ технологий, которые имеют схожую функциональность и предназначены для одинаковых задач на технологии стеков JavaEE и Spring [2]. Д.В. Козырев, Л.А. Володченкова разработали программный интерфейс серверной части для облачного хранилища данных [3]. А.С. Волков и К.А. Волкова произвели краткий обзор элементов трехуровневой архитектуры для современных Web-приложений [4]. В.Л. Волушкова провела обзор технологий на примере языка java и привела примеры с подробным описанием для использования фреймворка SpringBoot [5].

1.3Цель исследования

Цель исследования – используя язык программирования Java и сервис visualcrossing спарсить и вывести данные для прогноза погоды.

2 Материалы и методы

Для создания программы используется язык программирования java и сервис visualcrossing.

3 Результаты и обсуждения

Для получения прогноза погоды на Java необходимо выполнить несколько шагов:

- Создать запрос RESTful для получения прогноза погоды.
- Получить данные с помощью запроса HTTP GET.
- Обработать данные как CSV или JSON.

Будем использовать «Visual Crossing Weather API», поэтому для начала нужно зарегистрироваться и получить ключ API (рис.1).

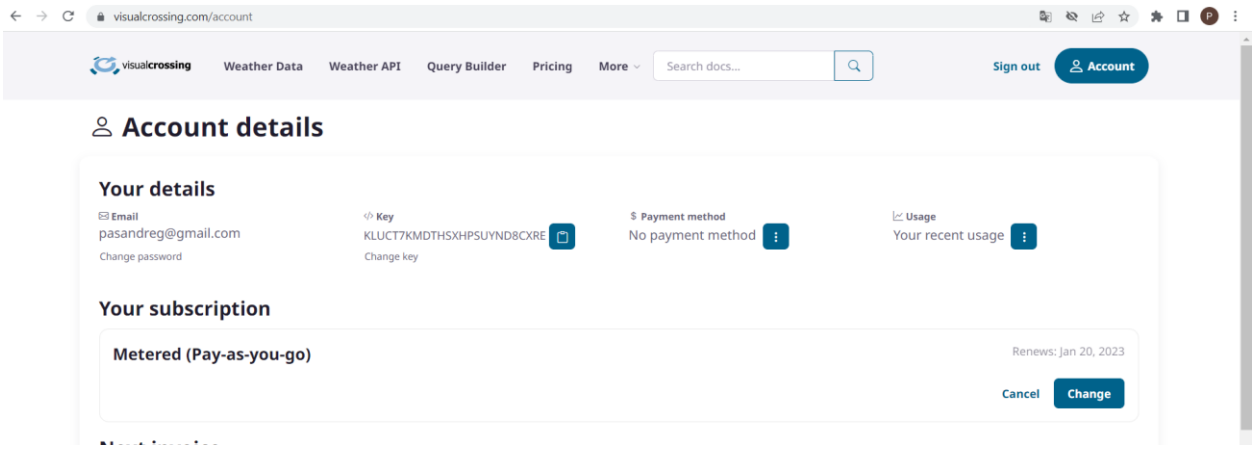

Рисунок 1 – сервис Visual Crossing

Также потребуется использовать библиотеку HttpComponents для отправки HTTP-запроса. Это можно сделать с помощью необработанных коммуникационных пакетов Java, но использование такой библиотеки, как HttpComponents, позволяет сосредоточиться на построении запроса Weather API и обработке результата.

Чтобы включить HttpComponents, добавим зависимость Maven (рис.2).

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.12</version>
</dependency>

<dependency>
    <groupId>com.mopano</groupId>
    <artifactId>hibernate-json-org-contributor</artifactId>
    <version>1.0</version>
</dependency>
```

Рисунок 2 – Добавление зависимости

Напишем код для определения используемых параметров. А определение точки для показа данных погоды зададим в теле класса (рис.3).

```
public static void timelineRequestHttpClient(String locations) throws Exception {

    String apiEndPoint="https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/";
    String startDate="2022-12-30";
    String endDate="2023-01-01";

    String unitGroup="metric";
    String apiKey="KLUCT7KMDTHSXHPSUYND8CXRE";
```

Рисунок 3 – Определение параметров

Будем использовать следующие параметры:

- «unitGroup» – нужны метрические единицы измерения: градусы Цельсия для температуры и миллиметры для осадков.
- «key»— уникальный идентификатор запроса.
- «startDate и endDate» - значение «null» выдаст прогноз на 1 неделю. Установим значение дат на промежуток с 30 декабря по 1 января.
- «apiEndPoint» - ссылка куда будет отправляться запрос.

Теперь можем собрать все параметры в url (рис.4.)

```
String requestBuilder = apiEndPoint + URLEncoder.encode(locations, StandardCharsets.UTF_8) + "/" + startDate + "/" + endDate;

URIBuilder builder = new URIBuilder(requestBuilder);

builder.setParameter( param: "unitGroup", unitGroup)
        .setParameter( param: "key", apiKey);

HttpGet get = new HttpGet(builder.build());

CloseableHttpClient httpclient = HttpClients.createDefault();

CloseableHttpResponse response = httpclient.execute(get);
```

Рисунок 4 – Сборка url

Далее напишем код, который будет отправляет запрос в Weather API. Методы здесь основаны на основах HttpClient (рис.5).

```
String rawResult=null;
try {
    if (response.getStatusLine().getStatusCode() != HttpStatus.SC_OK) {
        System.out.printf("Ошибка запроса, возвращен код:%d%n", response.getStatusLine().getStatusCode());
        return;
    }
    HttpEntity entity = response.getEntity();
    if (entity != null) {
        rawResult=EntityUtils.toString(entity, StandardCharsets.UTF_8);
    }
} finally {
    response.close();
}

parseTimelineJson(rawResult);
```

Рисунок 5 – Отправка запроса

Здесь не будет выполняться обработка исключений для ясности ошибки, главное, убедиться, что ответ http закрыт, когда он используется.

Когда запрос выполняется успешно, в ответе приходит прогноз погоды, заполненный в формате JSON в переменной «rawResult». Также можно запросить данные в формате CSV, если этот формат больше подходит для приложения.

Предыдущий пример кода извлекает данные прогноза погоды. Теперь нужно создать вывод данных из этой таблицы, проанализировав ответ JSON (рис.6).

```
private static void parseTimelineJson(String rawResult) {

    if (rawResult==null || rawResult.isEmpty()) {
        System.out.printf("Нет данных для обработки%n");
        return;
    }

    JSONObject timelineResponse = new JSONObject(rawResult);

    ZoneId zoneId=ZoneId.of(timelineResponse.getString( key: "timezone"));

    System.out.printf("Данные о погоде для: %s%n", timelineResponse.getString( key: "resolvedAddress"));
```

Рисунок 6 – Обработка полученных данных

В данной фрагменте переменная с данными «rawResult» анализируется как Json, для этого определяем JSONObject.

После анализа текста извлекаем часовой пояс для запрошенного местоположения. Это возвращается в JSON как свойство «timezone». Используем результат для создания экземпляра Java «ZoneId», который будем использовать для создания экземпляров «ZoneDateTime» для данных о погоде.

Чтобы использовать данные, создадим на консоли простой результат, ограниченный вкладкой, используя цикл вокруг свойства «day». Свойство дней представляет собой массив дат в результате (рис.7).

```
JSONArray values=timelineResponse.getJSONArray( key: "days");

System.out.printf("Дата\t\t|Макс. Темп\t\t|Мин. Темп\t|Осадки%n");
for (int i = 0; i < values.length(); i++) {
    JSONObject dayValue = values.getJSONObject(i);

    ZonedDateTime datetime=ZonedDateTime.ofInstant(Instant.ofEpochSecond(dayValue.getLong( key: "datetimeEpoch")), zoneId);

    double maxtemp=dayValue.getDouble( key: "tempmax");
    double mintemp=dayValue.getDouble( key: "tempmin");
    double pop=dayValue.getDouble( key: "precip");
    System.out.printf("%s\t|%.1f\t\t\t|%.1f\t\t|%s%n", datetime.format(DateTimeFormatter.ISO_LOCAL_DATE), maxtemp, mintemp, pop);
}
```

Рисунок 7 – Оформление вывода данных

Дата может быть получена из свойства «datetimeEpoch», которое представляет количество секунд, прошедших с эпохи UNIX. Используя ZoneId, созданный ранее, создадим экземпляр ZonedDateTime.

Конечным результатом является простой вывод данных о погоде на консоль для городов, указанных в теле вызова класса (рис.8-9).

```
public static void main(String[] args)  throws Exception {
    TimelineApiForecastSample.timelineRequestHttpClient( locations: "Birobidzhan");
    System.out.println("_____");
    TimelineApiForecastSample.timelineRequestHttpClient( locations: "Khabarovsk");
    System.out.println("_____");
    TimelineApiForecastSample.timelineRequestHttpClient( locations: "Novosibirsk");
}
```

Рисунок 8 – Указание городов для вывода данных

```
"C:\Program Files\Java\jdk-11.0.3\bin\java.exe" ...
Данные о погоде для: Биробиджан, Дальневосточный федеральный округ, Россия
Дата        |Макс. Темп     |Мин. Темп  |Осадки
2022-12-30  |-17,7          |-24,7      |0.0
2022-12-31  |-18,5          |-26,2      |0.0
2023-01-01  |-17,8          |-27,5      |0.0

-------------------------------------------------------------------------------------------
Данные о погоде для: Хабаровск, Дальневосточный федеральный округ, Россия
Дата        |Макс. Темп     |Мин. Темп  |Осадки
2022-12-30  |-18,3          |-26,3      |0.1
2022-12-31  |-19,7          |-27,4      |0.1
2023-01-01  |-18,7          |-28,6      |0.0

-------------------------------------------------------------------------------------------
Данные о погоде для: Новосибирск, Сибирский федеральный округ, Россия
Дата        |Макс. Темп     |Мин. Темп  |Осадки
2022-12-30  |-2,4           |-15,9      |2.8
2022-12-31  |-2,6           |-17,3      |2.2
2023-01-01  |-18,0          |-27,0      |0.5

Process finished with exit code 0
```

Рисунок 9 – Вывод данных о погоде

Получившееся приложение можно легко расширить, включив в него исторические данные о погоде, поскольку API-интерфейс Timeline Weather объединяет исторические и прогнозные данные в одном вызове API.

Также данное приложения можно доработать и создать информационную систему, либо мобильное приложение.

В данной статье был рассмотрен процесс для реализации парсинга данных метеоэлементов с web-ресурса visualcrossing.

Библиографический список

1. Потовиченко М.А., Привалов М.В., Корнев С.В. Компьютеризированная подсистема учета текущей успеваемости студента в условиях вуза // Информатика, управляющие системы, математическое и компьютерное моделирование. 2019. № 2. С. 71-75.
2. Нарижный А.Д., Губенко Н.Е. Сравнительный анализ стеков технологий spring и javaee (jakartaee) для разработки enterprise приложений // Информатика, управляющие системы, математическое и компьютерное моделирование. 2020. № 17. С. 459-462.
3. Володченкова Л.А., Козырев Д.В. Разработка серверной части программного приложения для удаленного хранения данных// Математические структуры и моделирование. 2020. № 1 (53). С. 108-138.

4. Волков А.С., Волкова К.А. Обзор архитектурных компонентов современного веб-приложения // Аллея науки. 2019. № 1(28). С. 958-961.
5. Волушкова В.Л. Архитектурные решения java для доступа к данным// Тверской государственный университет. 2019. С. 137-140.