ЗАНЯТИЕ 2.2

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Тема: Работа с сетью в Java и Android приложениях

Упражнение №1

В данном упражнении предстоит изучить особенности работы с классом HttpUrlConnection для выполнения запросов к серверу и обработки его ответов. Код одинаково будет в Java и Android приложениях, поэтому его можно использовать в обоих случаях. Обязательное требование к Android приложениях – работа с сетью в отдельном потоке.

Создайте новое Java-приложение. Добавьте в него новый класс HTTPRunnable, реализующий интерфейс Runnable. Данный класс будет отвечать за выполнение запросов к серверу в отдельном потоке. Создайте в классе переменные: address (адрес сервера), requestBody, responseBody. Создайте конструктор с двумя параметрами и геттер для responseBody.

```
public class HTTPRunnable implements Runnable{

    private String address;
    private HashMap<String,String> requestBody;
    private String responseBody;

    public HTTPRunnable(String address, HashMap<String, String> requestBody) {
        this.address = address;
        this.requestBody = requestBody;
    }

    public String getResponseBody() {
        return responseBody;
    }

    @Override
    public void run() {

    }
}
```

Рисунок 1

В методе run необходимо реализовать работу по отправке запроса на сервер. Для получений сущности URLConnection нужно использовать объект класса java.net.URL, его конструктор принимает тип String где помимо всего должен быть указан протокол – в данном случае http или https.

После получения сущности URL, вызывается метод openConnection, который возвратит сущность HttpsUrlConnection. При этом нужно обработать или пробросить IOException.

После этого переменная httpConnection будет хранить ссылку на объект HttpsUrlConnectionImpl. По умолчанию будет формироваться GET-запрос. Также тип запроса можно указать методом setRequestMethod. Если нужно добавить поля в header запроса используется метод setRequestProperty,

который принимает key и value. Например, можно установить Content-Type в application/x-www-form-urlencoded.

```
@Override
public void run() {
    if(this.address != null && !this.address.isEmpty()){
        try{
            URL url = new URL( string:this.address);
            URLConnection connection = url.openConnection();
            HttpURLConnection httpConnecton = (HttpURLConnection)connection;
            httpConnecton.setRequestMethod( method:"GET");
            InputStreamReader isr = new InputStreamReader( in:httpConnecton.getInputStream());
            BufferedReader br = new BufferedReader( reader:isr);
            String currentLine;
            StringBuilder sbResponse = new StringBuilder();
            while((currentLine = br.readLine()) != null){
                sbResponse.append( str:currentLine);
            }
            responseBody = sbResponse.toString();
            }catch(IOException ex){
                System.out.println("Error: " + ex.getMessage());
            }
    }
}
```

Рисунок 2

В методе main главного класса программы создайте новый поток и запустите его выполнение. В качестве адреса сервера, на который будет отправляться запрос используйте https://www.mirea.ru/. Так как параметры запроса отсутствуют, то для инициализации переменной requestBody в конструктор можно передать значение null.

```
public class Practical9 {

    public static void main(String[] args) {
        System.out.println( x:"Start program!");
        String server = "https://www.mirea.ru/";
        HTTPRunnable hTTPRunnable = new HTTPRunnable( url:server,  requestBody:null);
        Thread th = new Thread( r:hTTPRunnable);
        th.start();
        try {
            th.join();
        } catch (InterruptedException ex) {

        }finally{
             System.out.println("Response from server:" + server);
             System.out.println( x:hTTPRunnable.getResponseBody());
        }
    }
}
```

Рисунок 3

Запустите программу, убедитесь, что данные с сервера (HTML-страница) загружается успешно.

Реализуйте сохранение данных ответа сервера в файл.

```
public static void main(String[] args) {
    System.out.println( x:"Start program!");
    String server = "https://www.mirea.ru/";
    HTTPRunnable httpRunnable = new HTTPRunnable( url:server,  requestBody:null);
    Thread th = new Thread( r:httpRunnable);
    th.start();
    try {
        th.join();
    } catch (InterruptedException ex) {

    }finally{
        try {
            FileWriter fw = new FileWriter( string:"D:\\Work\\MIREA\\resp.html");
            fw.write( str:httpRunnable.getResponseBody());
            fw.close();
            System.out.println("Success save response from server:" + server);
        } catch (IOException ex) {
            System.out.println("Error response saving : " + ex.getMessage());
        }
    }
}
```

Рисунок 4

Для сохранения текстовых данных в файл можно использовать класс FileWriter. УБЕДИТЕСЬ, что каталог, куда вы пытаетесь сохранить файл **СУЩЕСТВУЕТ** на вашем компьютере. Имя файла может быть любое.

Запустите приложение, убедитесь, что файл успешно создается. Проверьте его содержимое.

Попробуйте выполнить запрос на другие интернет-ресурсы. Ознакомьтесь с ответами сервера. Продемонстрируйте сохраненные ответы преподавателю.

Упражнение №2

Продолжайте работать в данном приложении. Для выполнения следующего задания понадобится подключение дополнительной зависимости в проект. После отправки POST-запроса сервер будет возвращать данные в формате JSON, для обработки которого понадобится пакет, не входящий в состав JDK (в Android SDK он добавлен) – org.json.

JSON (англ. JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript. Но при этом формат независим от JS и может использоваться в любом языке программирования. В качестве значений в JSON могут быть использованы: JSON-объект, массив, число (целое или вещественное), литералы true (логическое значение «истина») или false (логическое значение «ложь»), строка.

Добавление новой зависимости необходимо осуществить с помощью системы Maven через файл конфигурации проекта pom.xml.

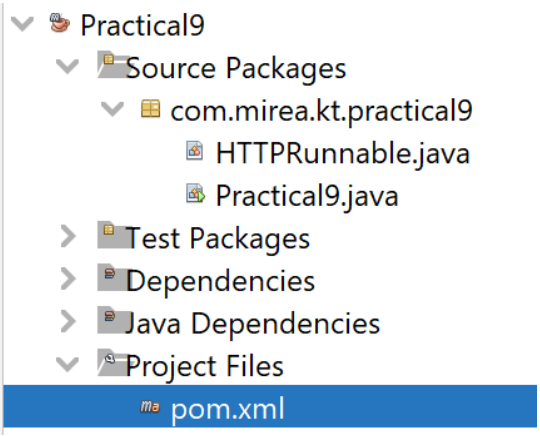

Рисунок 5

В нем необходимо добавить поле <dependencies>, включающее в себя описания всех зависимостей проекта. Внутри тега dependencies добавьте новую зависимость (org.json):

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.o
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mirea.kt</groupId>
    <artifactId>Practical9</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <exec.mainClass>com.mirea.kt.practical9.Practical9</exec.mainClass>
    </properties>
    <dependencies>
    <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20211205</version>
    </dependency>
    </dependencies>
</project>
```

Рисунок 5

После этого заново соберите проект, зависимость загрузится из репозитория Maven автоматически.

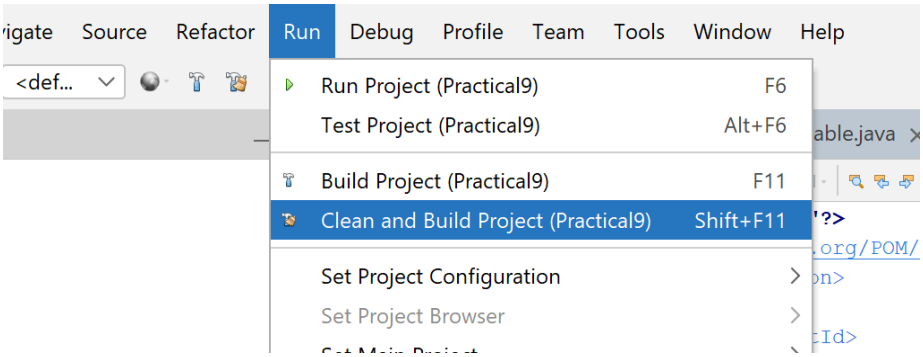

Рисунок 6

Измените код метода run класса HTTPRunnable, чтобы на сервер отправлялся POST-запрос, а также данные в теле запроса в формате fieldname1=value1&fieldname2=value2&fieldname3=value3.

```
@Override
public void run() {
    if(this.address != null && !this.address.isEmpty()){
        try{
            URL url = new URL( string:this.address);
            URLConnection connection = url.openConnection();
            HttpURLConnection httpConnecton = (HttpURLConnection)connection;
            httpConnecton.setRequestMethod( method:"POST");
            httpConnecton.setDoOutput( dooutput:true);
            OutputStreamWriter osw = new OutputStreamWriter( out:httpConnecton.getOutputStream());
            osw.write( str:generateStringBody());
            osw.flush();
            int responseCode = httpConnecton.getResponseCode();
            System.out.println("Response Code : " + responseCode);
            if(responseCode == 200){
                InputStreamReader isr = new InputStreamReader( in:httpConnecton.getInputStream());
                BufferedReader br = new BufferedReader( reader:isr);
                String currentLine;
                StringBuilder sbResponse = new StringBuilder();
                while((currentLine = br.readLine()) != null){
                    sbResponse.append( str:currentLine);
                }
                responseBody = sbResponse.toString();
            }else{
                System.out.println( x:"Error! Bad response code!");
            }
            }catch(IOException ex){
                System.out.println("Error: " + ex.getMessage());
            }
    }
}
```

Рисунок 7

```
/*
Метод формирования тела запроса из HashMap
*/
private String generateStringBody(){
    StringBuilder sbParams = new StringBuilder();
    if(this.requestBody != null && !requestBody.isEmpty()){
        int i = 0;
        for (String key : this.requestBody.keySet()){
            try {
                if (i != 0){
                    sbParams.append( str:"&");
                }
                sbParams.append( str:key).append( str:"=")
                        .append( str:URLEncoder.encode( s:this.requestBody.get(key),  enc:"UTF-8"));
            } catch (UnsupportedEncodingException e){
                e.printStackTrace();
            }
        i++;
        }
    }
    return sbParams.toString();
}
```

Рисунок 8

В главном классе в методе main сформируйте HashMap с параметрами, которые будут отправляться на сервер (фамилия, номер группы). Также необходимо поменять адрес сервера.

```
public static void main(String[] args) {
    System.out.println( x:"Start program!");
    String server = "https://android-for-students.ru";
    String serverPath = "/materials/practical/hello.php";
    HashMap<String,String> map = new HashMap();
    map.put( key:"name",  value:"Ivanov"); // впишите сюда вашу фамилию
    map.put( key:"group",  value:"RIBO-00-21"); // впишите сюда вашу группу
    HTTPRunnable httpRunnable = new HTTPRunnable(server + serverPath,  requestBody:map);
    Thread th = new Thread( r:httpRunnable);
    th.start();
    try {
        th.join();
    } catch (InterruptedException ex) {

    }finally{
        System.out.println( x:httpRunnable.getResponseBody());
    }

}
```

Рисунок 9

Обратите внимание на переменную **serverPath** – в ней отдельно выделен путь к php-скрипту, который выполнится на сервере. Данный скрипт будет обрабатывать данные POST-запроса. Запустите программу. Ознакомьтесь с ответом сервера.

В данном случае сервер будет возвращаться данные в формате JSON. Для более удобной работы с этим форматом необходимо использовать подключенную ранее библиотеку. Весь текст, который находится в фигурных скобках можно представить как JSONObject.

JSONObject представляет из себя структуру типа HashMap (ключ – значение). Если в JSONObject содержится JSONArray, то следует обратить внимание, что тип элемента такого массив по умолчанию Object.

```
{                                                                    JSONOBJECT
    "result_code": 1,
    "message_type": "data",
    "message_text": "Student Ivanov...blah-blah-blah",
    "task_list": [                                                   JSONARRAY
        "develop a Java application for editing files",
        "drink coffee",
        "develop a mobile application for communication with Elon Musk"
    ]
}
```

Рисунок 10

Т.е. чтобы получить значения поля result (которое в данном случае имеет тип int) нам необходимо выполнить вызов метода jsonObject.getInt("result_code") (получить значение типа int по ключу (названию поля) result_code). Сделайте то же самое для каждого поля.

Добавьте код по обработке ответа сервера в блок finally метода main:

```
}finally{
    JSONObject jSONObject = new JSONObject( source:httpRunnable.getResponseBody());
    int result = jSONObject.getInt( key:"result_code");
    System.out.println("Result: " + result);
    System.out.println("Type: " + jSONObject.getString( key:"message_type"));
    System.out.println("Text: " + jSONObject.getString( key:"message_text"));
    switch (result) {
        case 1: // если сервер вернул валидные данные
            JSONArray jSONArray = jSONObject.getJSONArray( key:"task_list");
            System.out.println( x:"Task list:");
            for(int i = 0;i < jSONArray.length();i++){
                System.out.println((i + 1) + ") " + jSONArray.get( index:i));
            }
            break;
        case 0: // если сервер вернул ошибку
            // какая-то обработка в случае оишбочных значений
            break;
        default: // во всех странных остальных случаях
            break;
    }
}
```

Рисунок 10

Попробуйте ввести ошибочные значения в поле name или group (например, оставьте поле пустым). Убедитесь, что ваша программа обрабатывает ответ-ошибку.