Wykorzystanie usług REST

PUM

Zagadnienia

- API (ang. *Application Programming Interface)* – interfejs programowania aplikacji, interfejs programistyczny aplikacji – zbiór reguł opisujących, w jaki sposób programy mogą komunikują się ze sobą.
- WebAPI to sposób w jaki programy komunikują się ze sobą przez sieć.
- RESTfull WebAPI to usługa sieciowa (WebAPI) zgodna z zasadami REST.
- Zasady REST.
- Tworzenie usługi na platformie NodeRED.
- Korzystanie z usługi.
- Biblioteka Volley – wysyłanie żądań https://developer.android.com/training/volley .
- JSON.

JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
     "streetAddress": "21 2nd Street",
     "city": "New York",
     "state": "NY",
     "postalCode": "10021"
   },
  "phoneNumber":  [
            {
              "type": "home",
              "number": "212 555-1234"
            },
            { "type": "fax",
              "number": "646 555-4567"
            }
   ],
  "gender": {
                "type": "male"
   }
}
```

Tworzenie obiektu JSON

```
Double z = 22.0;
Integer id;

JSONObject jsonObject = null;
try {
   jsonObject = new JSONObject();
   jsonObject.put("MeasurementId", id);
   jsonObject.put("x", "Tekst" );
   jsonObject.put("y", 123.1);
   jsonObject.put("z", z);
} catch (JSONException e) {
   e.printStackTrace();
}
```

Zezwolenie na dostęp do internetu

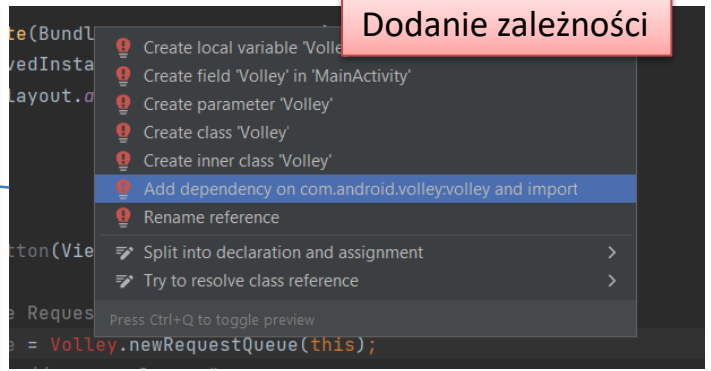
```
?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.webviewlocal">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.WebViewLocal"
        android:usesCleartextTraffic="true">
```

Żądanie tekstowe - StringRequest

Żądanie można wysłać Wieloma metodami, ale bez treści - kontentu

```
final TextView textView = (TextView) findViewById(R.id.text);
// ...

// Instantiate the RequestQueue.
RequestQueue queue = Volley.newRequestQueue(this);
String url ="https://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        // Display the first 500 characters of the response string.
        textView.setText("Response is: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        textView.setText("That didn't work!");
    }
});

// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Dodanie zależności



onRespons() – metoda wywoływana po otrzymaniu odpowiedzi, może nic nie robić

onErrorResponse() – metoda wywoływana po otrzymaniu błędu, może myć null

Żądanie tekstowe - StringRequest

```
RequestQueue queue;
String url = "http://lukan.sytes.net:1880/pum/cykl";

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    queue = Volley.newRequestQueue(this);
```

Stworzenie kolejki

```
StringRequest stringRequest = new StringRequest(Request.Method.GET, url + "?wiadomość=Ala ma kota",
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Toast.makeText(getApplicationContext(), response, Toast.LENGTH_LONG).show();
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH_LONG).show();
        }
    });
queue.add(stringRequest);
```

Parametr w adresie

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, url + "?wiadomość=Ala ma kota",
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                Toast.makeText(getApplicationContext(), response, Toast.LENGTH_LONG).show();
            }
        }, null );

queue.add(stringRequest);
```

onErrorResponse() – tu jest null

Żądanie JSON - JsonObjectRequest

```
String url = "http://my-json-feed";

JsonObjectRequest jsonObjectRequest = new JsonObjectRequest
      (Request.Method.GET, url, null, new Response.Listener<JSONObject>() {

   @Override
   public void onResponse(JSONObject response) {
      textView.setText("Response: " + response.toString());
   }
}, new Response.ErrorListener() {

   @Override
   public void onErrorResponse(VolleyError error) {
      // TODO: Handle error

   }
});

// Access the RequestQueue through your singleton class.
MySingleton.getInstance(this).addToRequestQueue(jsonObjectRequest);
```

Żądanie można wysłać wieloma metodami, treść to obiekt JSONObject.
W przykładzie JSONObject jest null.

Żądanie JSON - JsonObjectRequest

Stworzenie kolejki

```
RequestQueue queue;
String url = "http://lukan.sytes.net:1880/pum/cykl";

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    queue = Volley.newRequestQueue(this);
```

Tworzenie obiektu klasy JSONObject,
Operacja niebezpieczna i musi być otoczona try ... catch

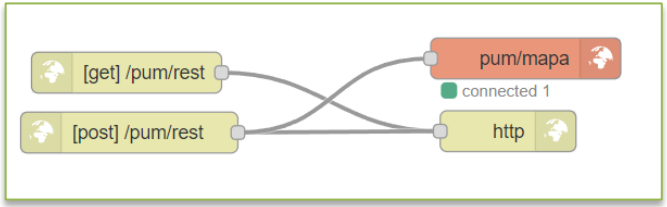
Wysłanie żądania

```
Double lat = 54.0, lon = 18.0;
JSONObject jsonObject = null;
try {
    jsonObject = new JSONObject();
    jsonObject.put("name","Andrzej");
    jsonObject.put("lat",lat);
    jsonObject.put("lon",lon);

} catch (JSONException e) {
    e.printStackTrace();
}

JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.POST, url,  jsonObject , new
Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {

    }
},null);

queue.add(jsonObjectRequest);
```

Wysyłanie pozycji do wyświetlenia na mapie

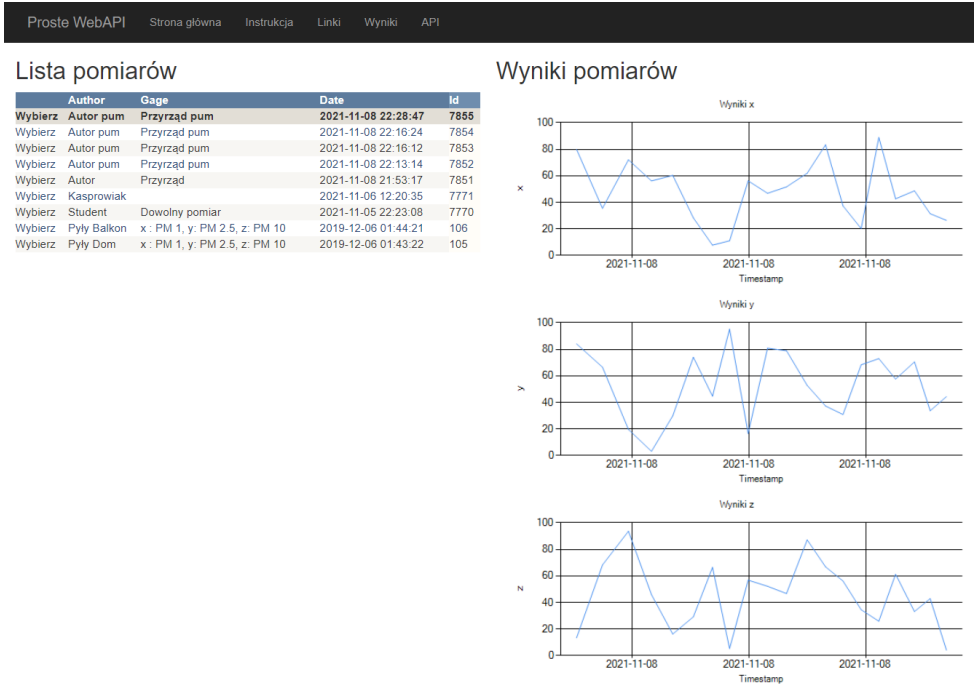
```
findViewById(R.id.button3).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Double lat = 54.0, lon = 18.0;
        JSONObject jsonObject = null;
        try {
            jsonObject = new JSONObject();
            jsonObject.put("name","Andrzej");
            jsonObject.put("lat",lat);
            jsonObject.put("lon",lon);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST,
                url,
                jsonObject ,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                    }
                },
                null);
        Toast.makeText(getApplicationContext(), jsonObject.toString(), Toast.LENGTH_LONG).show();
        queue.add(jsonObjectRequest);
    }
});
```

```
String url = "http://lukan.sytes.net:1880/pum/rest";
```



Program Rest_Json

http://argo.umg.edu.pl/www/RestWyniki/Pages/WebForm1



Dodawanie nowego pomiaru

```
RequestQueue queue;
String urlMeasurements = "http://argo.umg.edu.pl/www/RestWebApi/Api/Measurements";
String urlResults = "http://argo.umg.edu.pl/www/RestWebApi/Api/Results";
Integer id;
```

```
queue = Volley.newRequestQueue(this);
```

```
findViewById(R.id.buttonREST).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String author = ((TextView)findViewById(R.id.editTextTextAuthor)).getText().toString();
        String gage = ((TextView)findViewById(R.id.editTextTextGage)).getText().toString();
        JSONObject jsonObject = null;
        try {
            jsonObject = new JSONObject();
            jsonObject.put("Author", author);
            jsonObject.put("Gage", gage);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlMeasurements, jsonObject, new
Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                try {
                    id = response.getInt("MeasurementId");
                } catch (JSONException e) {
                    e.printStackTrace();
                }
                Toast.makeText(getApplicationContext(), response.toString(), Toast.LENGTH_LONG).show();
                ((TextView)findViewById(R.id.textViewId)).setText(id.toString());
            }
        }, null);
        queue.add(jsonObjectRequest);
    }
});
```

Dodawanie nowego wyniku

```
RequestQueue queue;
String urlMeasurements = "http://argo.umg.edu.pl/www/RestWebApi/Api/Measurements";
String urlResults = "http://argo.umg.edu.pl/www/RestWebApi/Api/Results";
Integer id;
```

```
queue = Volley.newRequestQueue(this);
```

```
findViewById(R.id.buttonResults).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Double x = 10.0, y = 13.0, z = 22.0;
        Random random = new Random();
        x = random.nextDouble()*100.0;
        y = random.nextDouble()*100.0;
        z = random.nextDouble()*100.0;
        JSONObject jsonObject = null;
        try {
            jsonObject = new JSONObject();
            jsonObject.put("MeasurementId", id);
            jsonObject.put("x", x);
            jsonObject.put("y", y);
            jsonObject.put("z", z);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlResults, jsonObject, new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                Toast.makeText(getApplicationContext(), response.toString(), Toast.LENGTH_LONG).show();
            }
        }, null);
        queue.add(jsonObjectRequest);
    }
});
```

Dodawanie nowego wyniku

```
findViewById(R.id.buttonResults).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Double x = 10.0, y = 13.0, z = 22.0;
        Random random = new Random();
        x = random.nextDouble()*100.0;
        y = random.nextDouble()*100.0;
        z = random.nextDouble()*100.0;
        JSONObject jsonObject = null;
        try {
            jsonObject = new JSONObject();
            jsonObject.put("MeasurementId", id);
            jsonObject.put("x", x);
            jsonObject.put("y", y);
            jsonObject.put("z", z);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlResults, jsonObject, new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                Toast.makeText(getApplicationContext(), response.toString(), Toast.LENGTH_LONG).show();
            }
        }, null);
        queue.add(jsonObjectRequest);
    }
});
```

Sprawozdanie

1. Żądanie GET na adres http://lukan.sytes.net:1880/pum/cykl
   - Ekran urządzenia przenośnego
   - Ekran z serwera
2. Żądanie POST z pozycją na adres http://lukan.sytes.net:1880/pum/cykl
   - Ekran urządzenia przenośnego
   - Ekran z mapą z serwera
3. Program wysyłania symulowanych danych pomiarowych do usługi REST http://argo.umg.edu.pl/www/RestWyniki/
   - Wpis na stronie z wynikami http://argo.umg.edu.pl/www/RestWyniki/Pages/WebForm1
4. Przykład samodzielnego wykorzystania usługi REST, np. odczyt pogody
   - https://openweathermap.org/api