# Politechnika Świętokrzyska w Kielcach

## Wydział Elektrotechniki, Automatyki i Informatyki

Projekt: Technologie Obiektowe

| Ocena | Temat: Konwerter formatów | Grupa: **1ID22A** Mateusz Wójcik Jarosław Spyrka |
|---|---|---|

# Cel projektu

Zadaniem na zaliczenie projektu jest stworzenie aplikacji, która pozwala na konwersje formatów:

- JSON i CSV
- JSON i XML

# Teoria

**JavaScript Object Notation, JSON** – lekki format wymiany danych komputerowych. JSON jest formatem tekstowym, bazującym na podzbiorze języka JavaScript. Typ MIME dla formatu JSON to application/json

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

**CSV** (ang. comma-separated values, wartości rozdzielone przecinkiem) – format przechowywania danych w plikach tekstowych i odpowiadający mu typ MIME text/csv.

| Year | Make | Model | Description | Price |
|---|---|---|---|---|
| 1997 | Ford | E350 | ac, abs, moon | 3000.00 |
| 1999 | Chevy | Venture "Extended Edition" | | 4900.00 |
| 1999 | Chevy | Venture "Extended Edition, Very Large" | | 5000.00 |
| 1996 | Jeep | Grand Cherokee | MUST SELL!<br>air, moon roof, loaded | 4799.00 |

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

**XML** (ang. Extensible Markup Language, rozszerzalny język znaczników) – uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. Jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy heterogenicznymi (różnymi) systemami i znacząco przyczyniło się do popularności tego języka w dobie Internetu

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

# Architektura

Projekt został napisany w architekturze REST API

# Działanie aplikacji

Użytkownik na przeglądarce internetowej wybiera opcje w jaki sposób dostarczyć dane do formatowania. Do wyboru ma:

- wgranie pliku
- wpisanie w pole tekstowe danych

Następnie wybiera na jaki format aplikacja ma przekonwertować plik.

Możliwe jest pobranie przekonwertowanego pliku lub skopiowanie z pola tekstowego.

# Konwertery

## Konwerter CSV na JSON

```
public static String csvTojson(String content,String separator){

    StringBuilder sb=new StringBuilder("[\n");

    String csv = content;
    if(csv.contains("\"")){
        csv.replaceAll( regex: "\"", replacement: "");
    }
    csv = csv.replaceAll( regex: "\\r", replacement: "");
    String[] csvValues = csv.split( regex: "\n");
    String[] header = csvValues[0].split(separator);

    for(int i=1;i<csvValues.length;i++){
        sb.append("\t").append("{").append("\n");
        String[] tmp = csvValues[i].split(separator);
        for(int j=0;j<tmp.length;j++){
            sb.append("\t").append("\t\"").append(header[j]).append("\":\"").append(tmp[j]).append("\"");
            if(j<tmp.length-1){
                sb.append(",\n");
            }else{
                sb.append("\n");
            }
        }
        if(i<csvValues.length-1){
            sb.append("\t},\n");
        }
        else{
            sb.append("\t}\n");
        }
    }

    sb.append("]");


    return sb.toString();
}
```

Konwerter z formatu CSV na format JSON. Metoda przyjmuje parametry zawartości oraz separatora. Zawartość jest to format CSV zapisany w ciągu znaków, separator jest to znak którym oddzielane są wartości.

1. Tworzony jest obiekt klasy StringBuilder. Pozwala on na dynamiczne tworzenie ciągu znaków.
2. Do stringbuildera dodawane są odpowiednie znaki odpowiadające za rozpoczęcie pliku JSON, klamerki.
3. Z  pierwszego wiersza pliku CSV pobierane są zmienne do tablicy heads, jako klucze do pliku JSON
4. Pobierane są wartości oddzielone separatorem, jako wartości do pliku JSON
5. W stringbuilderze wstawiany jest klucz a następnie odpowiadająca mu wartość
6. Wstawiane są znaki zamykające, klamerki
7. Zwracany jest ciąg znaków

## Konwerter JSON na CSV

```
private static void flatten(JSONArray obj, Map<String, String> flatJson, String prefix) {
    int length = obj.length();

    for (int i = 0; i < length; i++) {
        if (obj.get(i).getClass() == JSON_ARRAY) {
            JSONArray jsonArray = (JSONArray) obj.get(i);

            // jsonArray is empty
            if (jsonArray.length() < 1) {
                continue;
            }

            flatten(jsonArray, flatJson, prefix: prefix + "[" + i + "]");
        } else if (obj.get(i).getClass() == JSON_OBJECT) {
            JSONObject jsonObject = (JSONObject) obj.get(i);
            flatten(jsonObject, flatJson, prefix: prefix + "[" + (i + 1) + "]");
        } else {
            String value = obj.get(i).toString();

            if (value != null) {
                flatJson.put(prefix + "[" + (i + 1) + "]", value);
            }
        }
    }
}
```

W celu przekonwertowania pliku w formacie JSON na CSV najpierw spłaszczamy zawartość poprzez usunięcie klamer i wyciągnięciu danych do mapy.

1. Pobierany jest obiekt
2. Pętla przechodzi przez cały obiekt i uruchamia funkcje dla poszczególnych elementów
3. Z elementów pobierane są wartości i zapisywane do nowego ciągu znaków bez znaków pliku JSON.
4. Sprawdzane jest czy zmienne posiadają tablice

5. Jeżeli zmienne nie posiadają tablicy to dodawane są do stringbuildera
6. Jeżeli zmienne posiadają tablice to wstawiany jest prefix w postaci kropki i dodawane kolejne wartości

```
private static void flatten(JSONObject obj, Map<String, String> flatJson, String prefix) {
    Iterator<?> iterator = obj.keys();
    String _prefix = prefix != "" ? prefix + "." : "";

    while (iterator.hasNext()) {
        String key = iterator.next().toString();

        if (obj.get(key).getClass() == JSON_OBJECT) {
            JSONObject jsonObject = (JSONObject) obj.get(key);
            flatten(jsonObject, flatJson,  prefix: _prefix + key);
        } else if (obj.get(key).getClass() == JSON_ARRAY) {
            JSONArray jsonArray = (JSONArray) obj.get(key);

            if (jsonArray.length() < 1) {
                continue;
            }

            flatten(jsonArray, flatJson,  prefix: _prefix + key);
        } else {
            String value = obj.get(key).toString();

            if (value != null && !value.equals("null")) {
                flatJson.put(_prefix + key, value);
            }
        }
    }

}
```

Następnym etapem jest przetworzenie spłaszczonego JSONA na format CSV.

1. Pobierane są elementy z pierwszego wiersza i oznaczane jako nagłówki
2. Następnie wstawiane są między te wartości separatory
3. W kolejnej kolumnie wstawiane są wartości odpowiadające nagłówką
4. Wartości oddzielane są separatorami

```java
public static String getCSV(List<Map<String, String>> flatJson, String separator) {
    Set<String> headers = collectHeaders(flatJson);
    String csvString = StringUtils.join(headers.toArray(), separator) + "\n";

    for (Map<String, String> map : flatJson) {
        csvString = csvString + getSeparatedColumns(headers, map, separator) + "\n";
    }

    return csvString;
}

private static Set<String> collectHeaders(List<Map<String, String>> flatJson) {
    Set<String> headers = new LinkedHashSet<~>();

    for (Map<String, String> map : flatJson) {
        headers.addAll(map.keySet());
    }

    return headers;
}
```

## Konwerter XML na JSON

```java
public static String convertToJson(File file){

    DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    StringBuilder sb = new StringBuilder();
    Document doc = db.parse(file);
    Element root = doc.getDocumentElement();
    sb.append("{\"").append(root.getNodeName()).append("\":");
    root.normalize();
    if(root.hasChildNodes()){
        int depth = 0;
        sb.append("{");
        getNode(root.getChildNodes(), depth, sb);
        sb.append("}");
    }
    sb.append("}");

    return sb.toString();
}
```

1. Dane z XML zapisywane są w postaci Node`ów czyli powiązanych ze sobą kolekcji
2. Wstawiamy klamry otwierające następnie wchodzimy głębiej do zależności
3. Pobieramy nazwę node
4. Sprawdzamy czy następny element jest tablicą
5. Jeżeli jest tablicą to wstawiamy odpowiednia klamre i wypisujemy zawartość dziecka rodzica oraz ich rodzeństwo czyli elementy z tablicy
6. Jeżeli nie jest tablicą to wypisujemy zawartość dziecka

# Konwerter JSON na XML

```
        } else if (value instanceof JSONArray) {
            ja = (JSONArray)value;
            length = ja.length();

            for(i = 0; i < length; ++i) {
                value = ja.get(i);
                if (value instanceof JSONArray) {
                    sb.append('<');
                    sb.append(key);
                    sb.append('>');
                    sb.append(toString(value));
                    sb.append("</");
                    sb.append(key);
                    sb.append('>');
                } else {
                    sb.append(toString(value, key));
                }
            }
        } else if ("".equals(value)) {
            sb.append('<');
            sb.append(key);
            sb.append("/>");
        } else {
            sb.append(toString(value, key));
        }
```

1. Pobieramy wartości ze stringa
2. Przechodzimy po wszystkich elementach
3. Sprawdzamy, czym jest dany element
4. Wstawiamy w odpowiednie miejsce klucz oraz wartość i przechodzimy do następnego elementu

# API

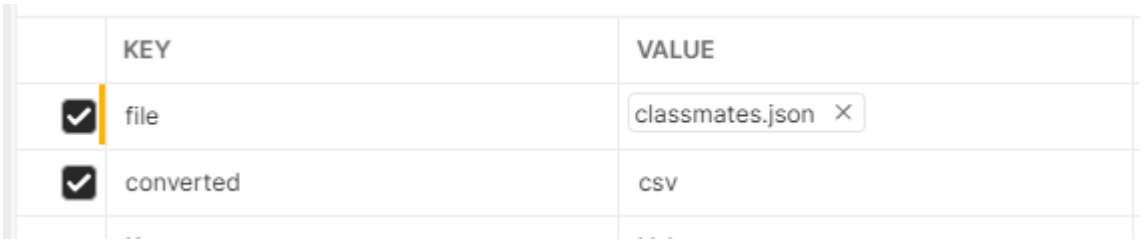
W celu połączenia się przeglądarki z serwerem wystawione są endpoint`y do wysyłania danych.

```
@PostMapping("v2/files/upload")
public ResponseEntity<ResponseMessage> fileConversion(@RequestParam("file") MultipartFile file
        ,@RequestParam String converted) {
```

W celu przesłania danych do przetworzenia należy wpisać URL:

*http://localhost:8080/api/v2/files/upload*

a w body należy wstawić odpowiednie klucze z wartościami:




Klucz file przyjmuje plik do przetworzenia a klucz converted przyjmuje Stringa z informacją na jaki format ma zostać plik przetworzony.

Następnie uruchamiane są odpowiednie metody i tworzony jest nowy plik oraz zwracana jest odpowiedź z nazwą przetworzonego pliku.

```
@GetMapping("/file/{filename:.+}")
@ResponseBody
public ResponseEntity<Resource> getFile(@PathVariable("filename") String filename){
```

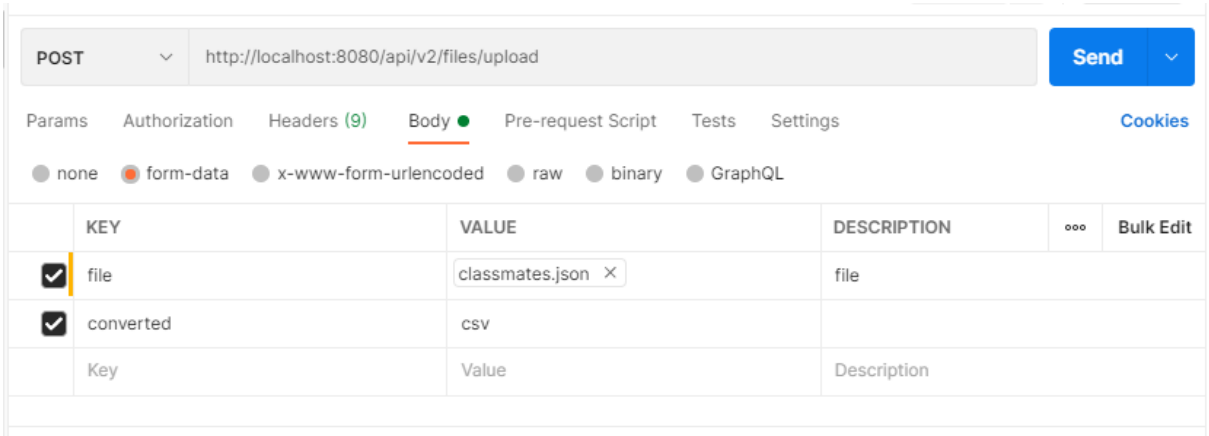

```
@GetMapping("/file/{filename:.+}")
@ResponseBody
public ResponseEntity<Resource> getFile(@PathVariable("filename") String filename){
```


W celu pobrania zawartości z serwera należy użyć:

- http://localhost:8080/api/content/”nazwa_pliku” w celu pobrania zawartości
- http://localhost:8080/api/file/”nazwa_pliku” w celu pobrania pliku

Serwer wyszukuje następnie pliku o danej nazwie i uruchamiana jest odpowiednia metoda.

## Działanie aplikacji po stronie back-end

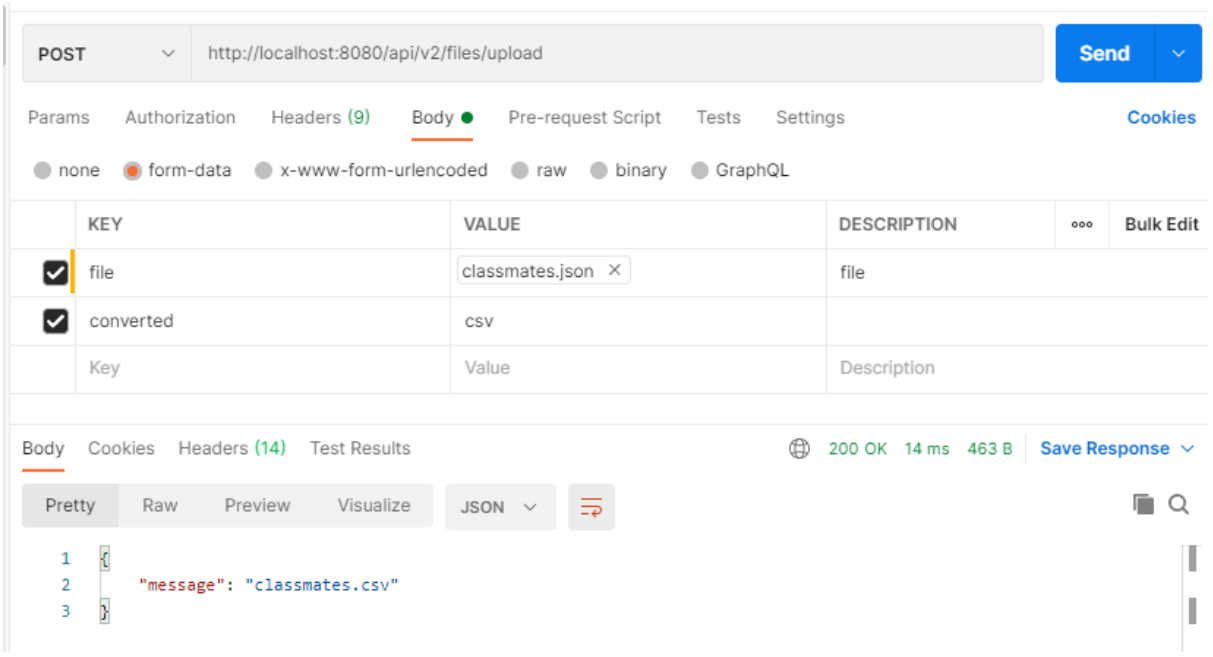


Przesyłamy metodę POST z plikiem oraz nazwą na jaki format ma zostać przekonwertowany.

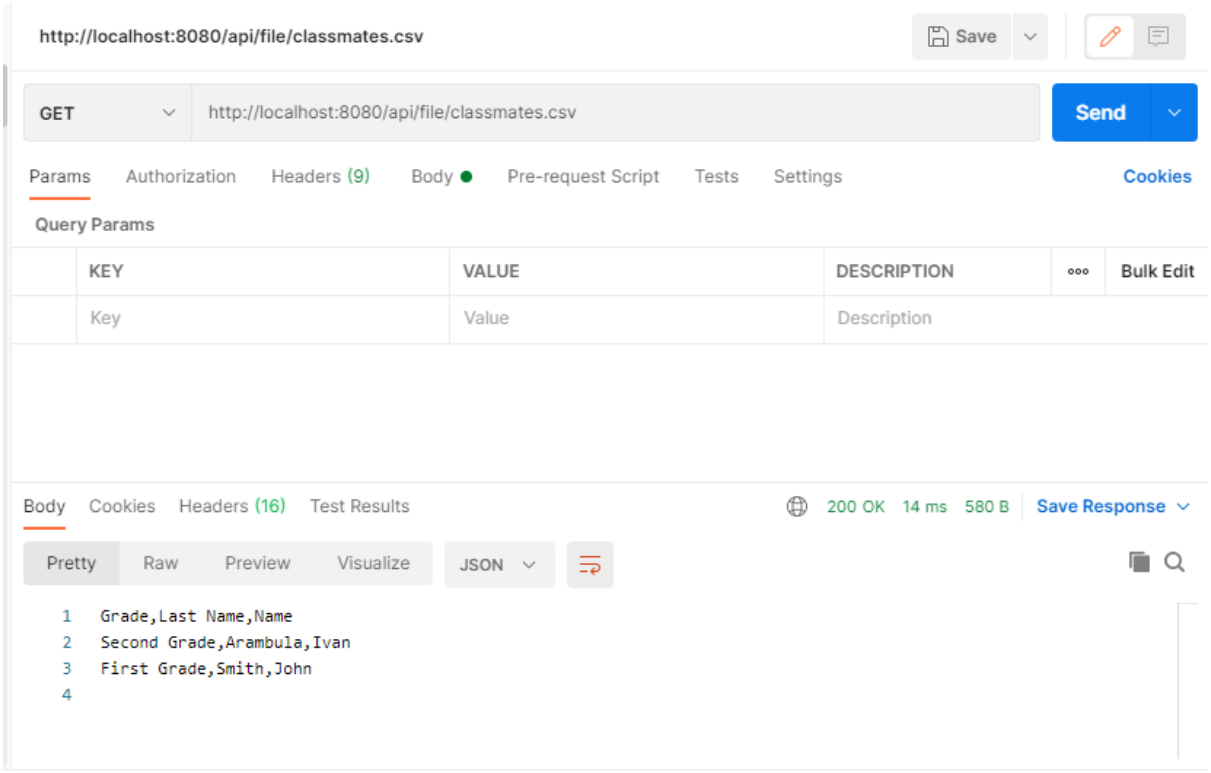
Przesłany plik JSON:


```
[{
        "Name": "Ivan",
        "Last Name": "Arambula",
        "Grade": "Second Grade"
    },
    {
        "Name": "John",
        "Last Name": "Smith",
        "Grade": "First Grade"
    }
]
```

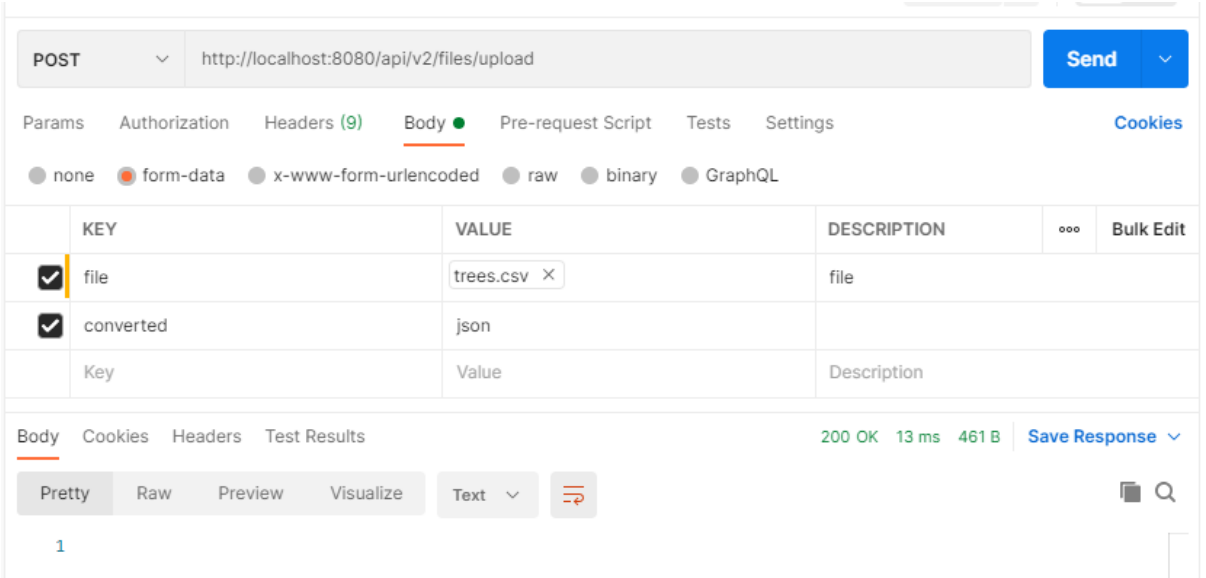

Serwer zwraca w response body nazwę pliku do pobrania z serwera.

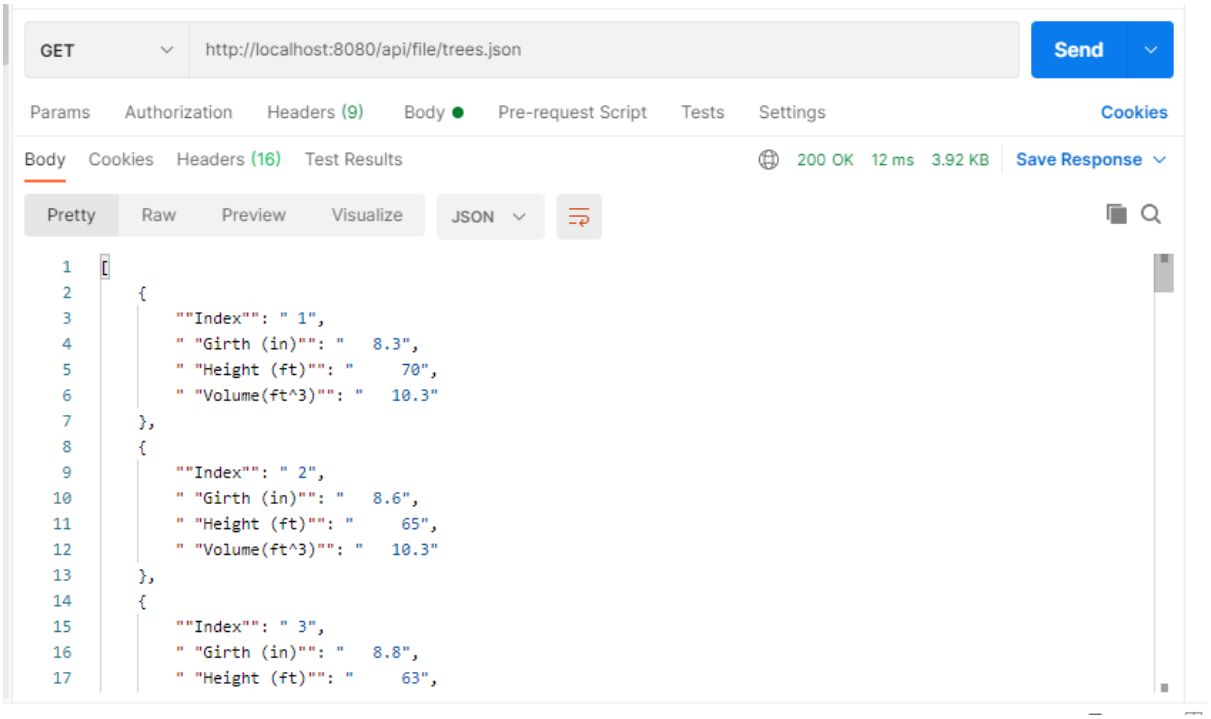



Po odebraniu pliku przy odpowiednim URL otrzymujemy plik do pobrania lub tekst do wyświetlenia:




Tak samo działa w przypadku konwersji CSV na JSON.






Skonwertowany plik:

| Index, "Girth (in)", "Height (ft)", "Volume(ft^3)" |
|---|
| 1, 8.3, 70, 10.3 |
| 2, 8.6, 65, 10.3 |
| 3, 8.8, 63, 10.2 |
| 4, 10.5, 72, 16.4 |
| 5, 10.7, 81, 18.8 |
| 6, 10.8, 83, 19.7 |
| 7, 11.0, 66, 15.6 |
| 8, 11.0, 75, 18.2 |
| 9, 11.1, 80, 22.6 |
| 10, 11.2, 75, 19.9 |
| 11, 11.3, 79, 24.2 |
| 12, 11.4, 76, 21.0 |
| 13, 11.4, 76, 21.4 |
| 14, 11.7, 69, 21.3 |
| 15, 12.0, 75, 19.1 |
| 16, 12.9, 74, 22.2 |
| 17, 12.9, 85, 33.8 |
| 18, 13.3, 86, 27.4 |
| 19, 13.7, 71, 25.7 |
| 20, 13.8, 64, 24.9 |
| 21, 14.0, 78, 34.5 |
| 22, 14.2, 80, 31.7 |
| 23, 14.5, 74, 36.3 |
| 24, 16.0, 72, 38.3 |
| 25, 16.3, 77, 42.6 |

```
[
    {
        ""Index"":" 1",
        " "Girth (in)"":"   8.3",
        " "Height (ft)"":"     70",
        " "Volume(ft^3)"":"   10.3"
    },
    {
        ""Index"":" 2",
        " "Girth (in)"":"   8.6",
        " "Height (ft)"":"     65",
        " "Volume(ft^3)"":"   10.3"
    },
    {
        ""Index"":" 3",
        " "Girth (in)"":"   8.8",
        " "Height (ft)"":"     63",
        " "Volume(ft^3)"":"   10.2"
    },
    {
        ""Index"":" 4",
        " "Girth (in)"":"  10.5",
        " "Height (ft)"":"     72",
        " "Volume(ft^3)"":"   16.4"
    },
    {
        ""Index"":" 5",
        " "Girth (in)"":"  10.7",
        " "Height (ft)"":"     81",
        " "Volume(ft^3)"":"   18.8"
    },
    {
        ""Index"":" 6",
        " "Girth (in)"":"  10.8",
        " "Height (ft)"":"     83",
        " "Volume(ft^3)"":"   19.7"
    },
    {
        ""Index"":" 7",
        " "Girth (in)"":"  11.0",
        " "Height (ft)"":"     66",
```


Konwersja w przypadku JSON na XML:


```
{"note":
{"to":["Tove","Jerry"],
"from":"Jani",
"heading":"Reminder",
"body":"Don't forget me this weekend!"
}
}
```

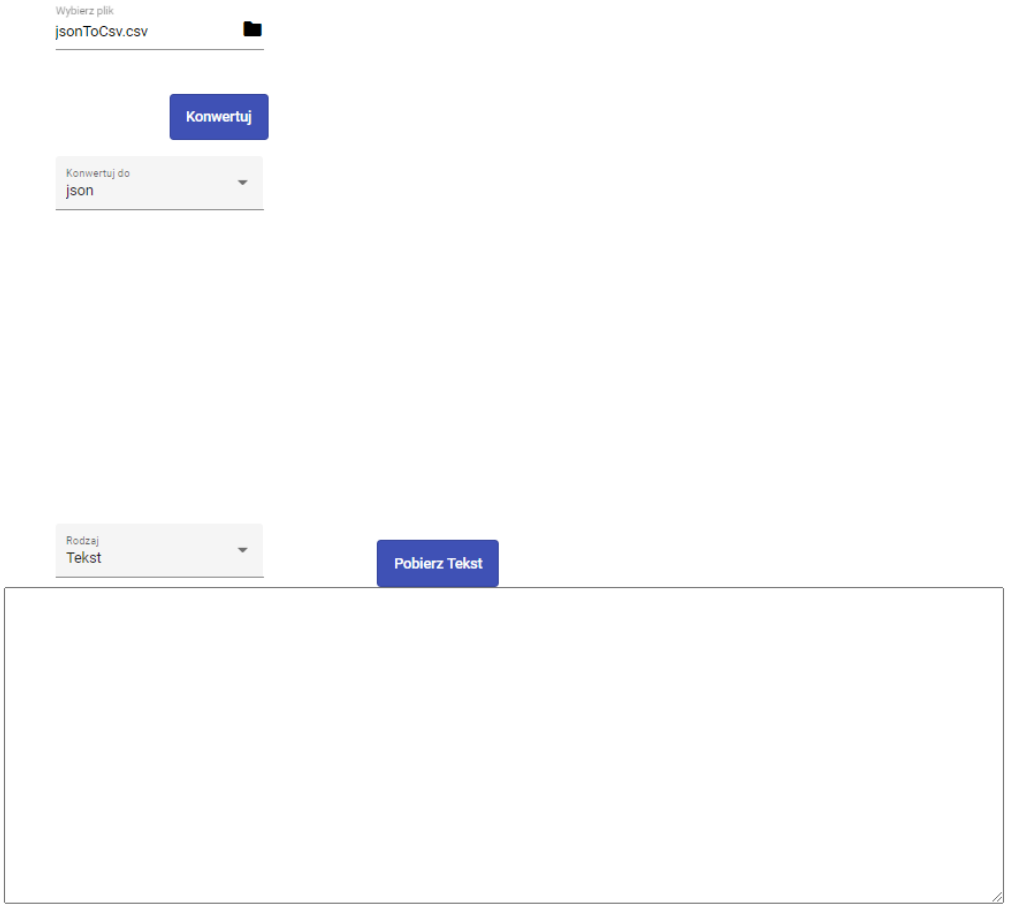
```xml
<note>
    <heading>Reminder</heading>
    <from>Jani</from>
    <to>Tove</to>
    <to>Jerry</to>
    <body>Don't forget me this weekend!</body>
</note>
```

Oraz z XML na JSON

```xml
<book>
  <name>A Song of Ice and Fire</name>
  <author>George R. R. Martin</author>
  <language>English</language>
  <genre>Epic fantasy</genre>
</book>
```

```json
{"book":
{"name":"A Song of Ice and Fire",
  "author":"George R. R. Martin",
  "language":"English",
  "genre":"Epic fantasy"}
}
```

# Prezentacja front-endu:




Na stronie można wgrać plik który chcemy konwertować.

Następnie wybieramy format i konwertujemy przyciskiem “Konwertuj”

Kolejnym krokiem jest wybranie w jaki sposób chcemy otrzymać dane. Do wyboru mamy plik do pobrania lub tekst który będzie wyświetlony w polu poniżej.