

# JSON Server

v3.1

# Wprowadzenie

# Czym jest JSON Server?

## Problem

Bardzo często pisząc strony czy też aplikacje okazuje się, że potrzebujemy dopisać komunikację z serwerem czy też zasymulować wygląd strony gdy już będą realni użytkownicy.

Przy większych projektach może zdarzyć się tak, że strona back-endowa aplikacji nie jest jeszcze gotowa a my nie możemy ruszyć dalej.

## Rozwiązanie

Z pomocą przychodzi nam **JSON Server**. Jest to paczka NPMowa, dzięki której, jesteśmy w stanie bardzo szybko stworzyć prostą bazę danych wraz z działającym Restowym API.

## Zalety JSON Serer:

- szybka konfiguracja,
- prostota działania,
- w pełni działający CRUD.

# Instalacja JSON Server

## Instalacja

Aby korzystać z **JSON Server**, najpierw musimy zainstalować go globalnie:

```
npm install -g json-server
```

Następnie tworzymy plik **db.json**, w którym będziemy przechowywać nasze informacje.

Nasz plik **db.json** będzie przechowywał informacje o samochodach.

## db.json

```
{
  "cars": [
    {
      "id": 1,
      "name": "Gallardo",
      "brand": "Lamborghini"
    }
  ]
}
```

# JSON Server

## Uruchomienie

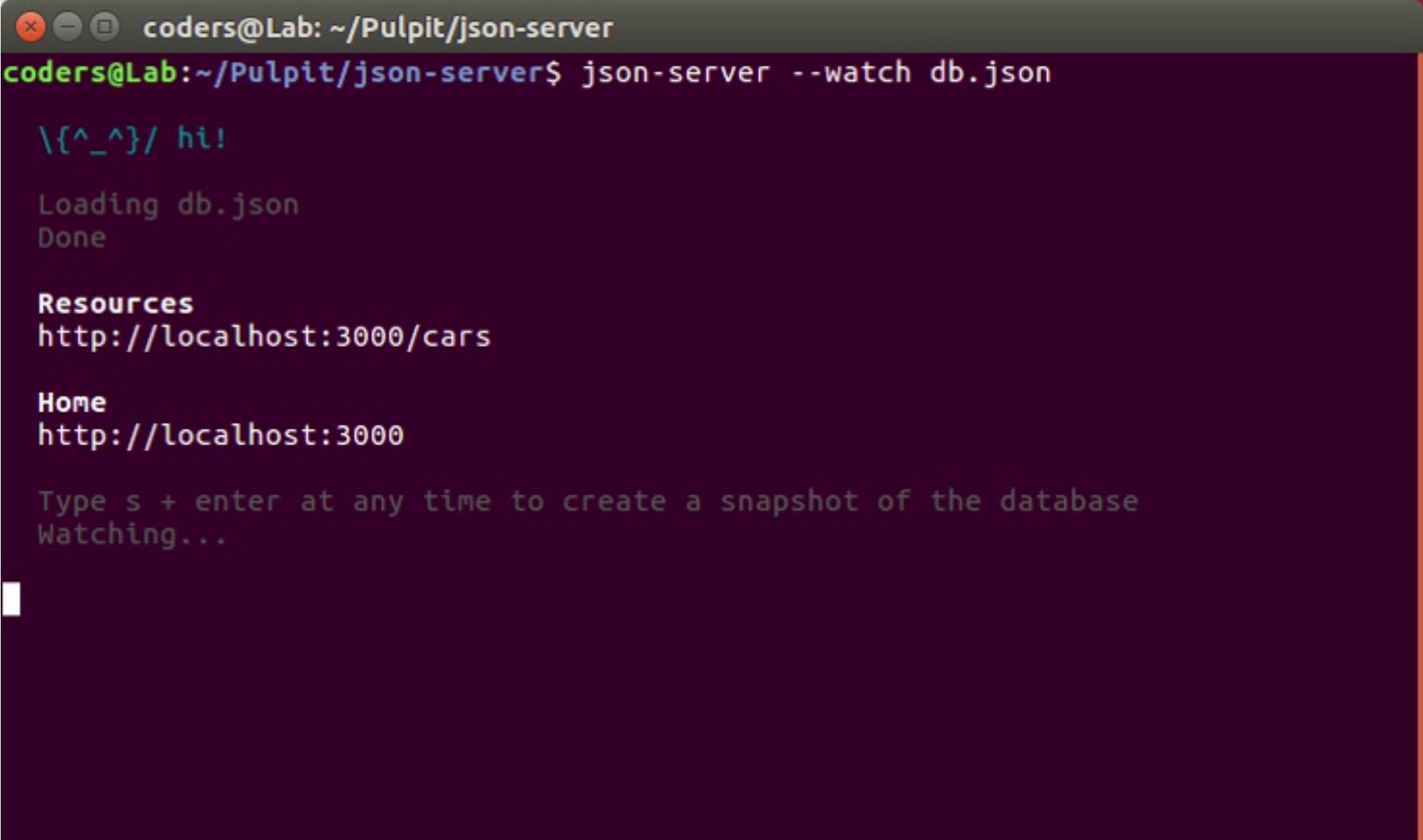
Gdy już mamy plik z danymi, możemy uruchomić nasz **JSON Server**. W katalogu, w którym jest plik db.json, wykonujemy następujące polecenie:

```
json-server --watch db.json
```

Po wykonaniu tej komendy, uruchomi się serwer lokalny, dzięki któremu będziemy mieć własne REST API.

Serwer jest dostępny pod adresem:

**http://localhost:3000**



```
coders@Lab: ~/Pulpit/json-server
coders@Lab:~/Pulpit/json-server$ json-server --watch db.json

\{^_^}/ hi!

Loading db.json
Done

Resources
http://localhost:3000/cars

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
Watching...
```

# JSON Server & jQuery

## Pierwsze użycie

Gdy już mamy działający JSON Server, możemy już z niego skorzystać. Do komunikacji z serwerem przy pomocy REST API użyjemy jQuery i metody AJAX.

Przykładowy kod, który znajduje się po prawej stronie, pobierze wszystkie dane z bazy i wyświetli je w konsoli.

Ścieżka **http://localhost:3000/db** zwróci nam całą bazę danych.

## app.js

```
var url = "http://localhost:3000";
$.ajax({
  method: "GET",
  url: url + "/db",
  dataType: "json"
}).done(function(response) {
  console.log(response);
});
```

# REST API Metody



# GET

## Pobieranie danych

Aby pobrać dane, musimy użyć metody **GET** oraz wiedzieć, jakie dane potrzebujemy mieć.

Wróćmy do naszej bazy zapisanej w **db.json**. Załóżmy, że chcemy sprawdzić, jaki samochód znajdujący się w bazie ma **ID 1**.

```
GET> http://localhost:3000/cars/1
```

Pobierać możemy samochody (dokumenty) po jednym, możemy również pobrać zbiór wszystkich samochodów (kolekcję).

## Przykładowy kod

```
var url = "http://localhost:3000";
$.ajax({
  method: "GET",
  url: url + "/cars/1",
  dataType: "json"
}).done(function(response) {
  console.log(response);
});
```



# POST

## Wysyłanie danych

Aby wysłać dane, musimy użyć metody **POST** oraz posiadać dane do wysłania.

Do naszej bazy dodamy nowy samochód (dokument). Marka samochodu to Daewoo a model to Polonez.

```
POST> http://localhost:3000/cars
```

Gdy nie określimy właściwości **ID**, zostanie ona dodana automatycznie a wartością będzie liczba o jeden większa niż ID o najwyższej wartości w bazie.

## Przykładowy kod

```
var url = "http://localhost:3000";
var car = {
  name: "Polonez",
  brand: "Daewoo"
};
$.ajax({
  method: "POST",
  url: url + "/cars",
  dataType: "json",
  data: car
}).done(function(response) {
  console.log(response);
});
```

# PATCH i PUT

## Modyfikacja danych

Aby modyfikować dane, musimy użyć metody **PUT** lub **PATCH**.

Jeśli będziemy modyfikować jedną właściwość – użyjemy metody **PATCH**, jeśli cały dokument – wtedy użyjemy metody **PUT**.

## Przykładowy kod

```
var url = "http://localhost:3000";
var car = {
  name: "Polonez Caro"
};
$.ajax({
  method: "PATCH",
  url: url + "/cars/2",
  dataType: "json",
  data: car
})
.done(function(response) {
  console.log(response);
});
```

# DELETE

## Usuwanie danych

Aby usunąć dane, musimy użyć metody **DELETE** oraz wiedzieć, które dane chcemy usunąć.

Usuniemy samochód, który posiada ID 2 (Polonez Caro).

```
DELETE> http://localhost:3000/cars/2
```

## Przykładowy kod

```
var url = "http://localhost:3000";
$.ajax({
  method: "DELETE",
  url: url + "/cars/2",
  dataType: "json"
}).done(function(response) {
  console.log(response);
});
```

# Dokumentacja

Cała dokumentacja dostępna jest na GitHubie:

<https://github.com/typicode/json-server>