# Working with JSON

JSON data providers provide just that. There is no schema with that data.Why do we need schema, some might ask?

Schema for data is like building columns and corners in Autocad Drawings. Based on those columns and corner points,as an example in AutoCad, one can recreate the plan for all floor, with one click of a computer button.

Computer and software web services allow different objects, located in different applications, on a different hardware devices, such as computer and smartphones and tablets, to talk to one another!

Is JSON considered a web service? In my personal opinion, it is a shadow of web services. Why do we use web services? We use them, so that one consumer, sitting in Canada, can consume the same web service, in a same way, and use that data, as the other consumer, sitting in Australia, be it a novice computer software developer, a merchant, a marketing agent, a marketing business analyst, or a senior programmer analyst, who wants to impress his boss, who closes the deals.

As an example, let's take a database of cars,that has information about cars, trucks,motorcycles, and everything that has wheels. It contains details about cars, for example, including car's details,such as make, model,color,year,new,used,factory id,and so on.

My computer screen has 1280 x 800 = 1,024,000 pixels. A car has more than 1,024,000 parts. All those parts' details are stored in databases of one sort or another. SQL Server, is an example of a database ( a base for data ).

A car data provider, who wants to share cars' details with car dealers,individual people,marketing agencies,and so on,offers all this data in JSON format. As an example,this is JSON format:`

```
{
  "Car": "Four Wheel Drive",
  "Truck": "Racing Truck",
  "TwoWheels": "BMX" ,
  "ThreeWheels": "Three Wheel Scooter",
  "FourWheels": 700,
  "CanBeDriven": true
}
```

As a result, we need to consume it. Wouldn't it  be great if we could do something like this and get the details we want,like:

MachineType.Car

MachineType.Truck

MachineType.Bicycle

Or, in a mass marketing email that we are going to send before holidays, we need to include the type of a car:

"Dear Customer,

As the holiday season is approaching, the <% MachineType.Car %> cars that we have,will have a special price tag on them . . ."

Or,

"Our shop offers <%  MachineType.Car, MachineType.Truck, MachineType.TwoWheels  %>, and <% MachineType.FourWheels - 200 + " Cars"  %>".

To do that, and based on the JSON data that we have above, we need to create a class, a C# class. I am using Visual Studio 2012, and .NET Framework 4.0, but it is not necessary. We can use VB, and .NET any version.

The class that I am going to create, will have to mimic the exact data types and structure as our JSON array mentioned above.

It will look like this:

```
public class MachineType
 {
    public string Car { get; set; }
    public string Truck { get; set; }
    public string TwoWheels { get; set; }
    public string ThreeWheels { get; set; }
    public int FourWheels { get; set; }
    public bool CanBeDriven { get; set }
 }
```

And before jumping into retreiving and manipulating our static JSON data,let us also consider this another JSON format. The same as our first one but a bit more:

```
{
 "Car": "Four Wheel Drive",
 "Truck": "Racing Truck",
 "TwoWheels": "BMX" ,
 "ThreeWheels": "Three Wheel Scooter",
 "FourWheels": 700,
 "CanBeDriven": true,
 "MachineTypeDetails": [
     {
          "Interior": "Beige",
          "Exterior": "White",
          "Seating": 4,
          "Doors": 2
     },
     {
          "Interior": "Black",
          "Exterior": "Black",
          "Seating": 4,
          "Doors": 2
```

```
      }
   ]
}
```

The "MachineTypeDetails" field that was added to our initial array is an array itself, which contains its own array of data items: Interior,Exterior,Seating,and doors, and there could be as many as you want "MachineItemDetails".

So now we have two JSON arrays, one nested within another array. The little trick and tip here is that for each JSON array, we need to create a C# class.

So our schema for JSON array will look as follows:

```
public class MachineType
  {
     public string Car { get; set; }
     public string Truck { get; set; }
     public string TwoWheels { get; set; }
     public string ThreeWheels { get; set; }
     public int FourWheels { get; set; }
     public bool CanBeDriven { get; set }
     public MachineTypeDetails [ ] MachineDetails { get; set; }
  }

public class MachineTypeDetails
  {
     public string Interior { get; set; }
     public string Exterior { get; set; }
     public int Seating { get; set; }
     public int Doors { get; set; }
  }
```

As you can see, `public MachineTypeDetails` is a type in MachineType, which we will refer when accessing JSON sub arrays.

Do we need a schema for JSON? Well, to be honest, no, but if you do not want your compiler or interpreter to get lost or get tired, in such a way you will give it orientation, and clear direction (schema) where to look for data, if there are millions of JSON array records.

Now what if we have three nested JSON arrays as follows:

```
{
 "Car": "Four Wheel Drive",
 "Truck": "Racing Truck",
 "TwoWheels": "BMX" ,
 "ThreeWheels": "Three Wheel Scooter",
 "FourWheels": 700,
 "CanBeDriven": true,
 "MachineTypeDetails": [
     {
          "Interior": "Beige",
          "Exterior": "White",
          "Seating": 4,
          "Doors": 2,
          "Windows": [
          {
               "Date":"2014-11-08T21:00:00.008514-03:00",
               "InteriorWindow":"clear",
               "ExteriorWindow":"scratched"
          }
          {
               "Date":"2014-10-08T20:00:00.008314-12:00",
               "InteriorWindow":"clear",
               "ExteriorWindow":"clear"
          }
     ]
     },
     {
```

```
            "Interior": "Black",
            "Exterior": "Black",
            "Seating": 4,
            "Doors": 2
        }
    ]
}
```

So our schema for this JSON array will look as follows:

```
public class MachineType
 {
    public string Car { get; set; }
    public string Truck { get; set; }
    public string TwoWheels { get; set; }
    public string ThreeWheels { get; set; }
    public int FourWheels { get; set; }
    public bool CanBeDriven { get; set }
    public MachineTypeDetails [ ] MachineDetails { get; set; }
 }

public class MachineTypeDetails
 {
    public string Interior { get; set; }
    public string Exterior { get; set; }
    public int Seating { get; set; }
    public int Doors { get; set; }
    public Windows [ ] Windows { get; set;}

 }

public class Windows
 {
    public DateTime Date { get; set; }
    public string InteriorWindow{ get; set; }
    public string ExteriorWindow { get; set; }
 }
```

And so on and so forth for all the nested JSON arrays, wherever they are nested, it does not matter where, and at what level, first nested level, second, third, or tenth nested level. The application is the same, and very simple. Object oriented approach helps solve many problems.

And before we can do this:

**MyMachineType.MachineTypeDetails[1].Windows[0].InteriorWindow;**

to get the state of an interior window for a parked in car, we need to utilize the powerful performance of .NET 4.0 by reading JSON request by HTTP as follows:

```
var request = (HttpWebRequest)WebRequest.Create("http://sobdomain.getjsondata.com/");
var response = (HttpWebResponse)request.GetResponse();
var JsonString = new StreamReader(response.GetResponseStream()).ReadToEnd();

MachineType MyMachineType =
Newtonsoft.Json.JsonConvert.DeserializeObject<MachineType >(JsonString);
```

My choice is to include a reference to Newtonsoft.Json Dynamic Link Library for Deserialization of JSON string arrays.

Thank you,

Jinan Kordab
softuniversum.com