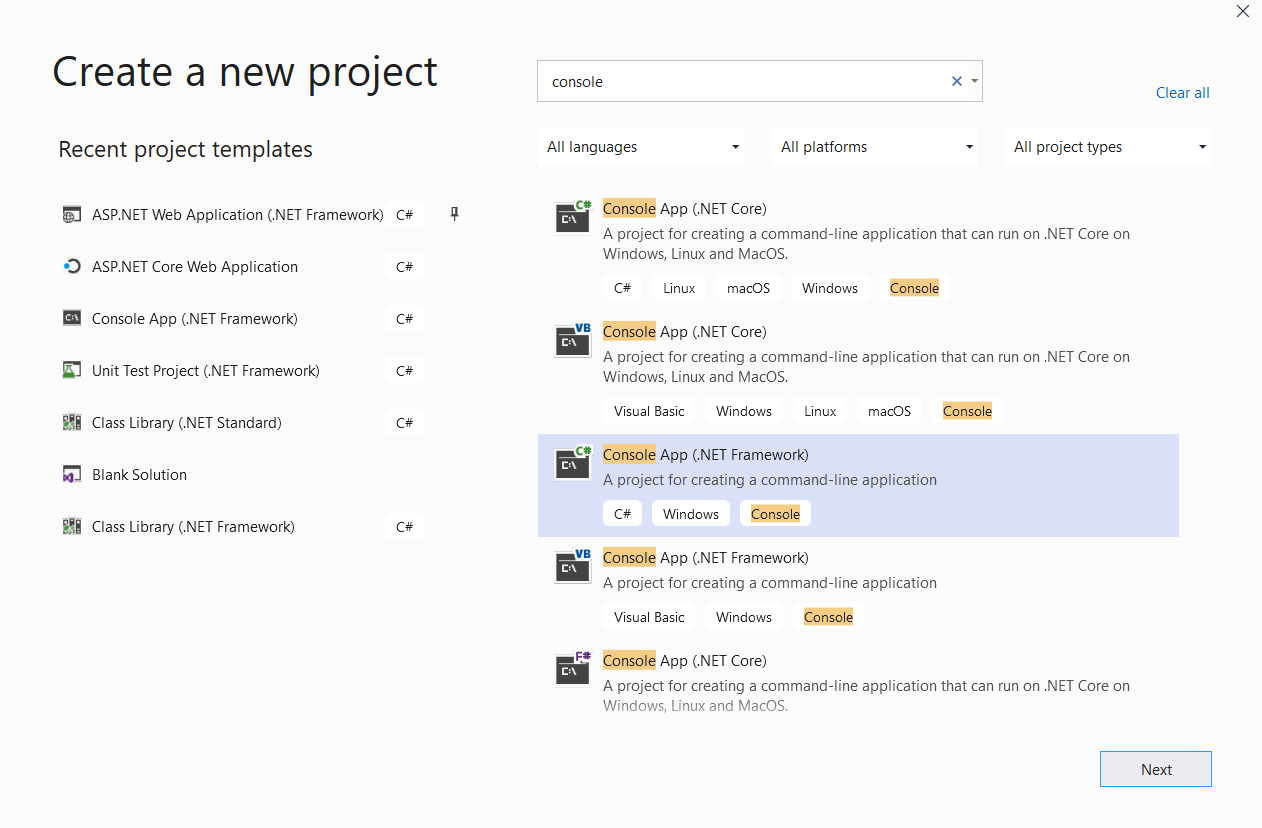
**Create JSON Objects [Serialize & Deserialize]**

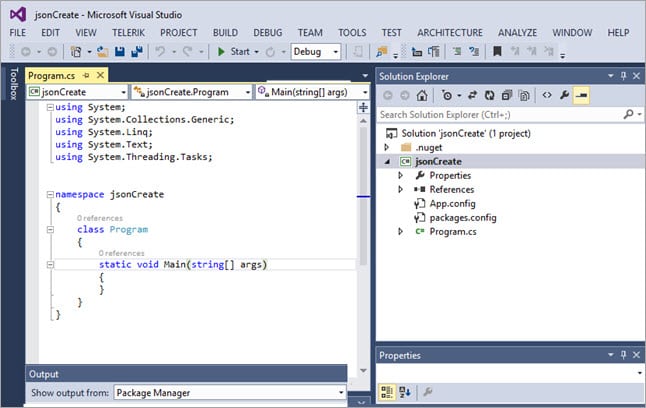
Give a proper meaningful name to your project and provide the location. Here, as we are going to write a simple program to create a JSON, I have given it a name like **“jsonCreate”**. You can provide any name that you are comfortable with or which is easier for you to identify your program.

1. **Create a new project**

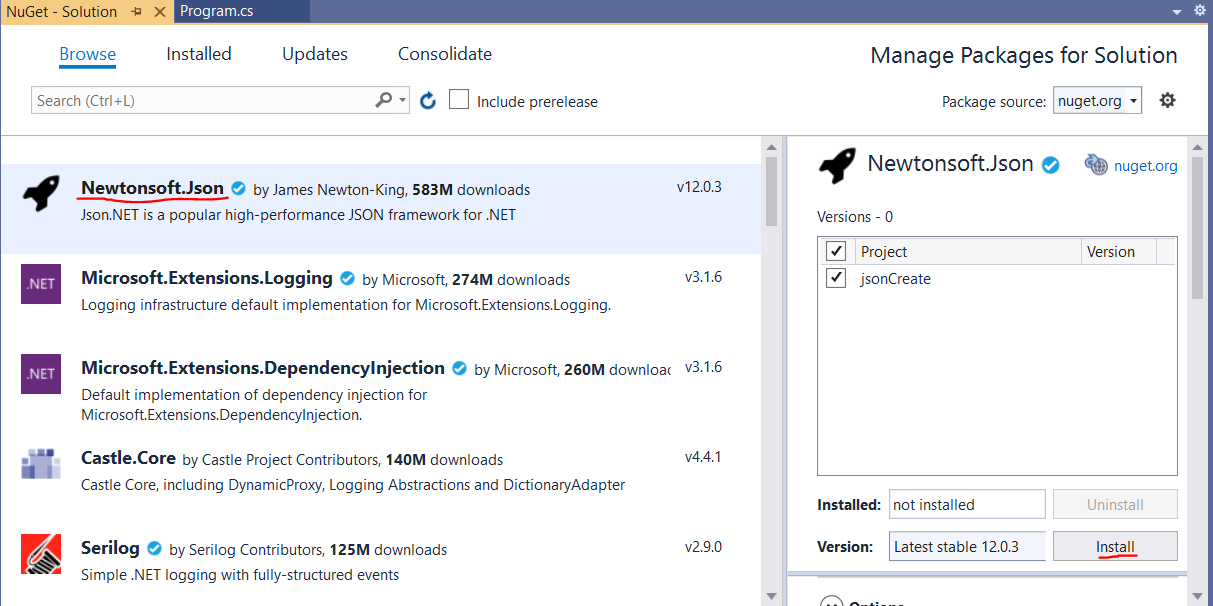


Once everything is set **click on ok** button.

**A new project will be created and it will look as the image given below:**

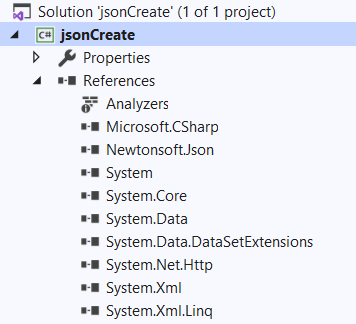
[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/2-JSON-New-Project.jpg)

Once the project has been created, we will add **json.net** reference to the project. To add the reference, right click on the solution in the right panel and click on the **“Manage NuGet Packages”** option from the menu list.



**Click on the install** button to install, Newtonsoft.Json. It will begin downloading the Newtonsoft.Json package. Once the download is complete it will be installed and a green down arrow will appear on Newtonsoft.Json.

Go to reference in the solution explorer, where you will find that a reference for Newtonsoft.json has already been added there.



So, with the creation of a project and addition of the Newtonsoft.json our setup is complete. Now, we can start writing the code for creating JSON.

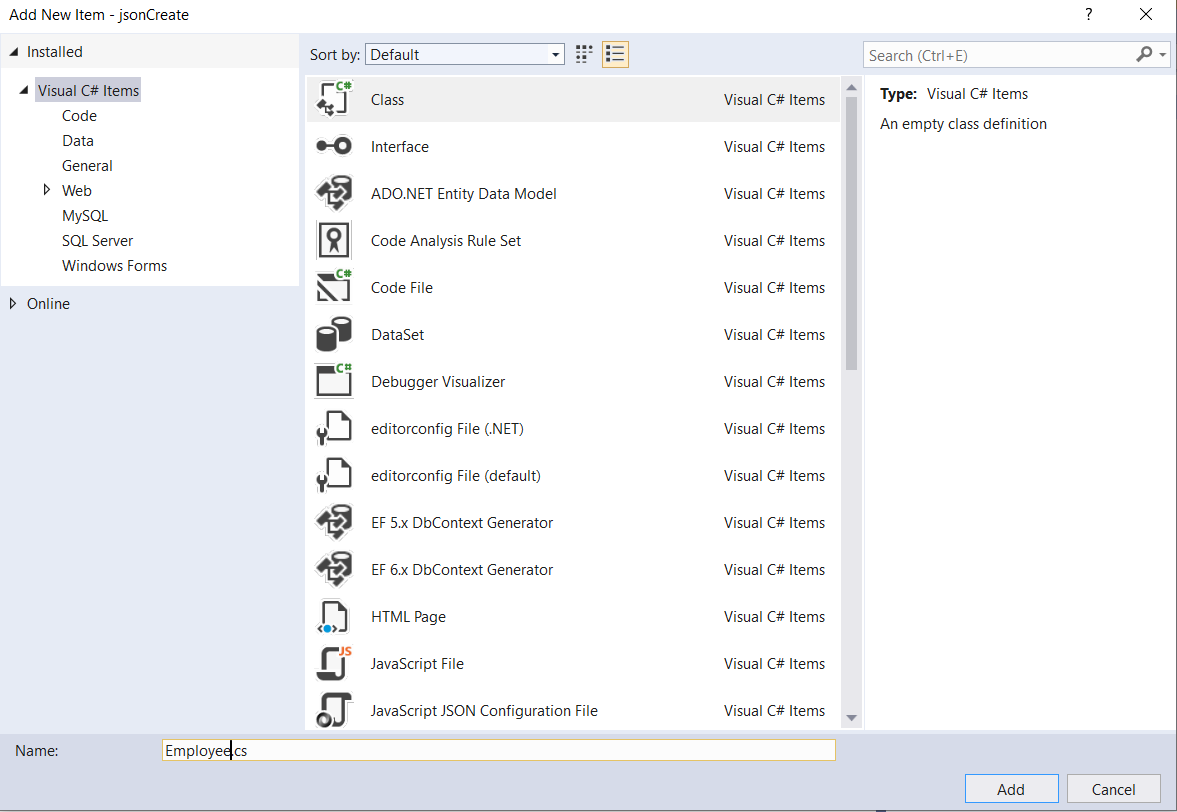
1. **Writing the Code for your First JSON**

We have already added the reference for the Newtonsoft to our solution. Now, we can start working on our very first code to serialize and create a JSON. We will start with a simple JSON structure and later let's gradually move towards more complex structures while discussing each line of the code and its functionality in details. So let’s begin!

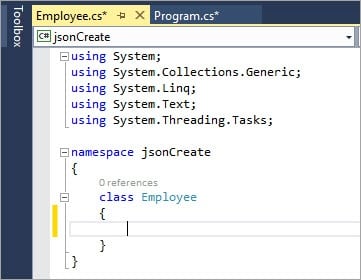
Let’s say we want to create an employee JSON with the following employee data.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/Creating-Emplayee-JSON.jpg)

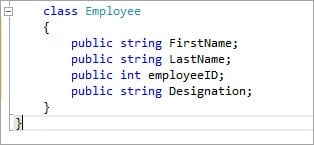
**For structuring a JSON, let’s first add a new class to our project.**



I am calling this class as **“Employee”**, you can give any relevant name for your class. Once you have created the class, it will get added inside the current namespace.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/7-Added-New-Class.jpg)

Once the class has been created, let’s define the variable objects in the new class.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/8-Defining-Variable.jpg)

Here, we have assigned public access to our objects. This will ensure that we can access these objects from any other class inside the namespace. This will be quite helpful while we use JSON serialize.

Further keeping a similar set of data in a single class makes it easier for the user to change the data on the go or perform any operations on the data. This will also help in maintaining the data integrity as any changes in the objects in any class will be restricted to that class only. The user will not have to make changes to the project.

We have also assigned the data type for each of the variables that we have defined here. Now, let’s go back to our main method.

First, we will define the employee class as an object in our main method.

Employee emp = new Employee();

Next, we will serialize the class object that we defined into JSON using **JsonConvert.SerializeObject**. Let’s store the serialized data inside a string variable.

string JSON result = JsonConvert.SerializeObject(emp);

Now, we have serialized the data into JSON structure, but we will need to save the data somewhere, so we will provide a path. To make it simpler we will store the location path into a string variable to use it later.

string path = @"C:\json\employee.json";

Now, to save the JSON at the given location we will use **StreamWriter** to save the .JSON file at the given path.

using (var tw = new StreamWriter(path, true))

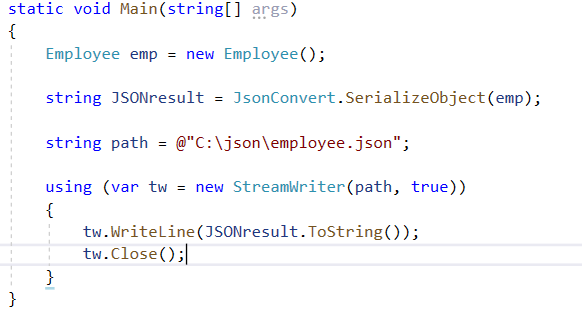
{

tw.WriteLine(JSONresult.ToString());

tw.Close();

}

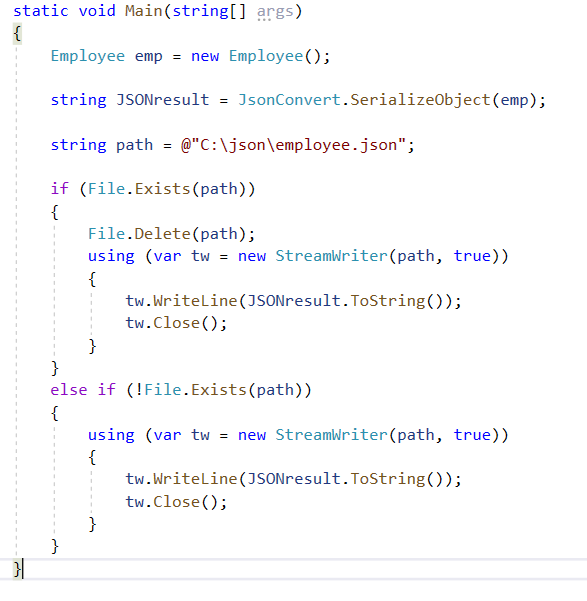
**Overall code structure for the main method will look like this:**



As shown the StreamWriter will keep on placing the newly created file at the given location. But, if the location already contains a file with the same name then what will happen? So, to handle this kind of situation we will write a simple condition to check whether the given file already exists at the particular location, if yes then we will first delete it then save a new file.

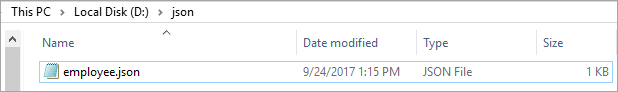
To do this we will simply enclose the StreamWriter with an i**f condition**. We will use **File. Exists** on the path that we provided earlier to validate if the file is already present at the given location. If it is present then our code will delete the first one and then it will create a new one.

If the condition is not true, i.e. the file is not present then it will directly create the file at the given path.

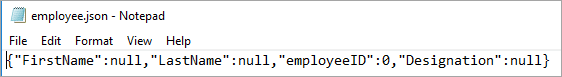


So, everything is set now. Let's build our project first. Once the build is complete and we don’t have any compilation errors remaining then we are good to go. Just **click on the Start button** at the top and the program will be executed. The program will create our first **.json** at the given location.

Now, we will navigate to the location we have provided in the code and we can see an **employee .json** file presents there.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/11-Json-File.png)

**Open the JSON file to view the content.**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/12-Json-Input-File.png)

All the keys that we provided in the employee class are present in the JSON but the values are null for string and it's “0” for integer.

Let’s now try to add values to the keys in the JSON.

There are numerous ways by which a value can be assigned to its key using the code but as we have just pitched into the initial stage of the JSON creation we will directly add the values to the variables in the employee class itself.

Go to the employee class and assign values directly to the variables. This will allow the class object that we created in the main method to pick both the key and values together directly from the class.

|  |
| --- |
| class Employee  {  public string FirstName = "Sam";  public string LastName = "Jackson";  public int employeeID = 5698523;  public string Designation = "Manager";  } |

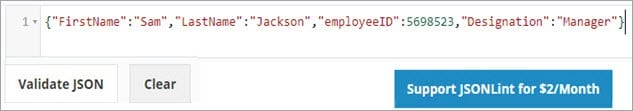
Now, we will save the project and build it again. Once the build is complete we will run the project. Now let's navigate to the path where the JSON is being saved, we will find that a new JSON has been created at the location.

Open the new file. It will now have all the key-value pairs as assigned in our code.

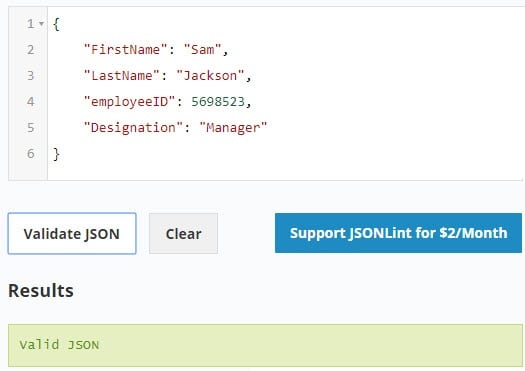
[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/13-Open-New-File.jpg)

Finally, we have created a JSON file but let’s validate if the JSON that we have created has a valid structure or not. To validate this we will go [here](https://jsonlint.com/).

Just copy the data from the JSON file and paste it into the text area of the site.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/14-Json-File.jpg)

After pasting the data click on the **“Validate JSON”** button. This will arrange the data and validate if the JSON we have provided is valid or not.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/12/15-Validate-JSON.jpg)

Congrats we have created our first valid JSON file programmatically.

**Exercises**

**1. Create a Student JSON:**

Create a Student JSON with the following keys: Name, Class, Subjects, and Roll No.

The name is a string, Class. And “Roll No”, will be integer. And Subject will be an Array or a list.

Pass the appropriate values to each key.

**2. Read the JSON file into a C# Object:**

Now try to read you file or simply copy what is in you student.json in a variable and make an student Object in C#

Hint: [Link](https://www.codementor.io/@andrewbuchan/how-to-parse-json-into-a-c-object-4ui1o0bx8)