EXERCISE 12, WORKING WITH FILES

FUNDAMENTAL

# Exercise 12, Working with Files

## Objective

The objective of this practical session is for you to become familiar with a number of classes found in the IO namespace and practice reading and writing to files.

## Part 1 – File I/O Practice

1. Create a new C# Console Application called Lab12
2. Add three pre-supplied text files to your project.   
   course.txt, trainer.txt and courseTrainer.txt

**Tip:** to add these right mouse click on the project name and then select Add > ‘Existing item’ and select the text files.

1. Create a method in the Program class with following signature

**private static string[] ReadCSV(string inFile)**

1. Write code in the **ReadCSV** method to read one line of the **inFile** parameter and split the resulting string using "**,**" as delimiter. Return the resulting string[].

**Tip**: use the String's **Split(',')** method and return the resulting string[ ].

1. Back in Main(), call the ReadCSV() method and pass it the "**../../course.txt**" as file name and capture the result in a variable called **courses**.

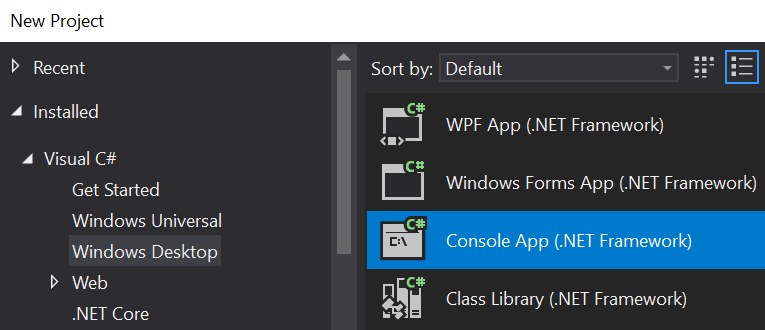
Do the same for "**../../trainer.txt**" and capture the result in variable called **trainers**.

The reason for the strange file path is because the compiled code is stored in **/bin/debug/Lab12.exe**, therefore you'll need to look up twice (**../..**), to get to the root of the project where you copied the files.

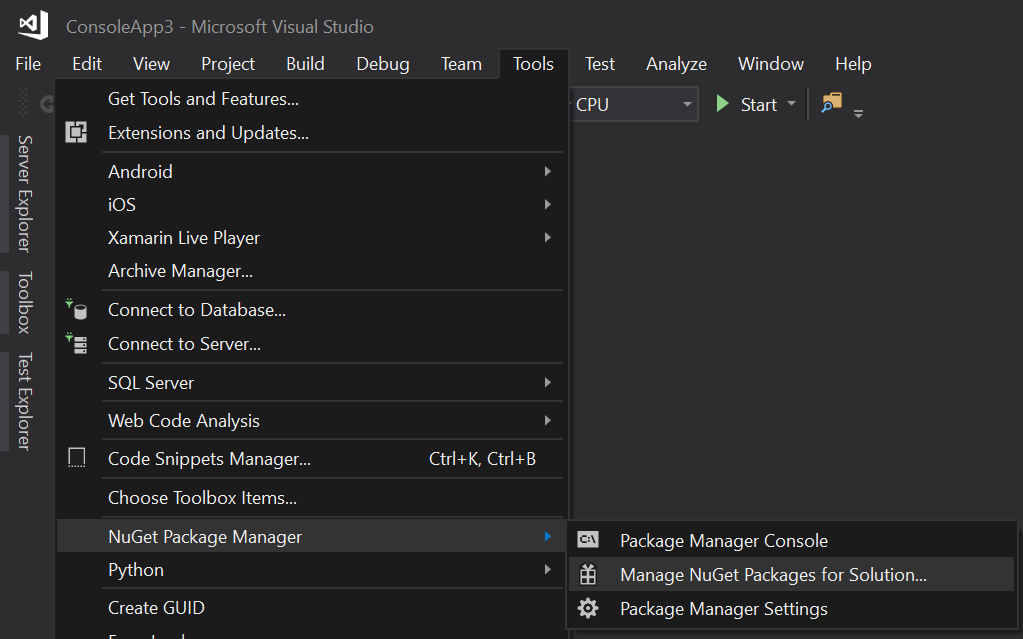
1. Create a file called trainersAndCourses.txt and append the courses and trainers arrays to this file.
2. Run your code and make sure it works.

**Part 2 – Read and parse JSON data.**In this part you’ll read a JSON file and parse the data into a List of Customers

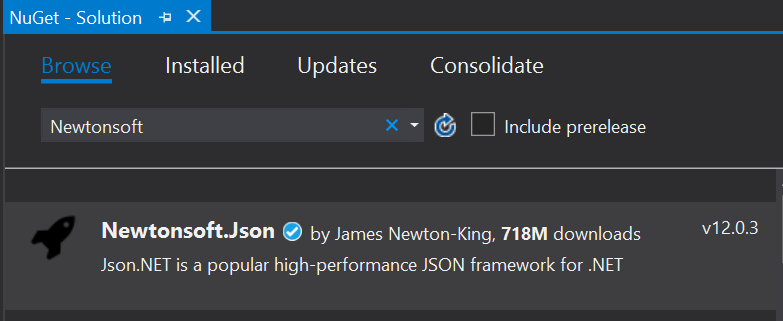
1. Create a new Console application and choose a suitable name for your project

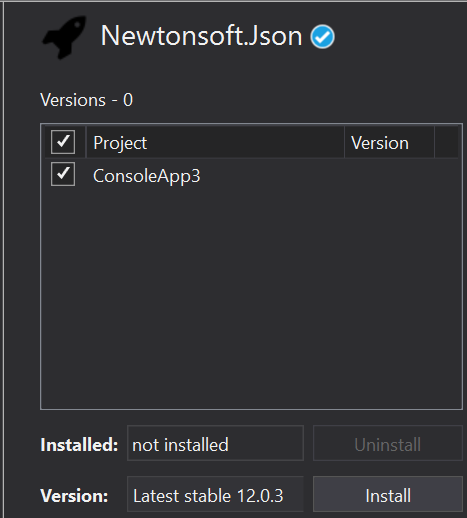


1. Use the NuGet GUI tool to import the NewtonSoft library for processing JSON file.



1. Select the Browes option and then type “Newtonsoft” in the search box.



Select the **Newtonsoft** and then add it to your project  


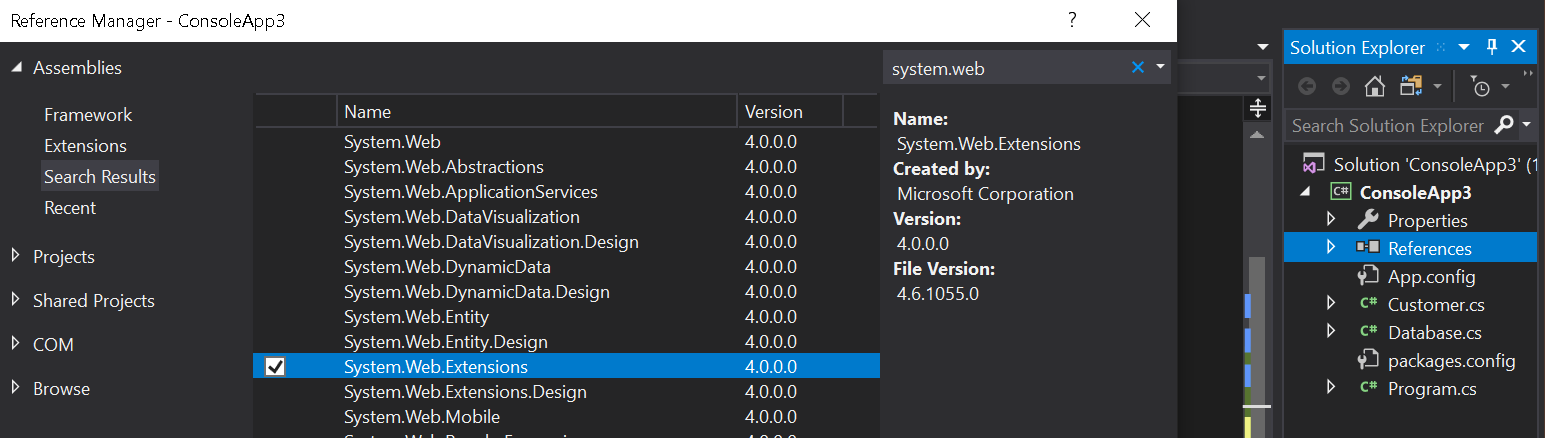
1. You’ll soon read customer data and therefore you’ll need to define a class than matches the customer found in the provided customers.json
2. Create a new Class called Customer as:  
   **public class Customer  
   {  
    public string CustomerID { get; set; }  
    public string CompanyName { get; set; }  
    public string ContactName { get; set; }  
    public string ContactTitle { get; set; }  
    public string Address { get; set; }  
    public string City { get; set; }  
    public string Region { get; set; }  
    public string PostalCode { get; set; }  
    public string Country { get; set; }  
   }**
3. Create a method to read data from the customers.json as:  
   (the location of your customers.json may differ)  
   **public static List<Customer> getCustomers()  
   {  
    using (StreamReader r = new StreamReader(@"c:\qa\customers.json"))  
    {  
    return JsonConvert.DeserializeObject<List<Customer>>(r.ReadToEnd());  
    }  
   }**

**The using statement creates a StreamReader object and automatically dispose of it within its block (which also closes it). This is good practice in C#.**

**Please use the above method to read the data and display it within your console.**

Writing objects as JSON:  
**How would you write a single object to a file?   
How about a List of objects?   
Here is a sample code to get you started:**  
 You can use the **NewtonSoft** package (see code below) or use a built-in library. The following example code shows you how to use both but let’s start with the **System.Web.Script.Serialization** reference.  
  
Let’s start by setting a reference to the **System.Web.Extensions.dll**

* Right mouse click on the References node in the solution explorer
* Select Assemblies on the left menu
* Type System.web in the search window
* Select System.Web.Extensions from the list (as seen below)



The following code example shows how to serialise and reserialize objects using this library. The resulting string can also be written to a file or return from a web site.

**using System.Web.Script.Serialization;**

**class Program**

**{**

**static void Main(string[] args) {**

**JavaScriptSerializer ser = new JavaScriptSerializer();**

**int[] nos = { 1, 3, 5, 7, 9 };**

**string res = ser.Serialize(nos);**

**Console.WriteLine(res);**

**int[] z = ser.Deserialize<int[]>(res);**

**Student stu = new Student("Bob", 21);**

**res = ser.Serialize(stu);**

**Student stu1 = ser.Deserialize<Student>(res);**

**Console.WriteLine(stu1.name);**

**List<Student> students = new List<Student>**

**{**

**stu,**

**new Student("Linda",23),**

**new Student("Mike",30)**

**};**

**res = ser.Serialize(students);**

**Console.WriteLine(res);**

**List<Student> myStudents = ser.Deserialize<List<Student>>(res);**

**foreach (var st in myStudents) {**

**Console.WriteLine(st.name);**

**}**

**}**

**}**

//-----------------------------------

**public class Student**

**{**

**public string name;**

**public int age;**

**public Student(string name, int age)**

**{**

**this.name = name;**

**this.age = age;**

**}**

**// a default constructor needed for serializer to work**

**public Student()**

**{**

**}**

**}**

### **Note:**

You can also use the **NewtonSoft** package to achieve the same task by using the **JsonConvert.SerializeObject()** method such as   
  
string res = Newtonsoft.Json.JsonConvert.SerializeObject(myStudents);

## Part 3 – Relate courses and trainers (optional and only if you've time)

1. Add a pre-supplied text file called **trainerCourse.txt** to the project.

This file contains rows of two **","** separated integers.

The two integers point to trainer numbers (the first value) and course titles (second number).   
Trainer and course numbers being the order in which trainers and courses are stored in the files.

For example, **1,2** indicates   
"*Mike Baradaran*" (**1st** row of trainer.txt), teaching  
"*Systems Development Basics*" (**2nd** row of course.txt).

Convert all the numbers to trainer names and course titles in order to produce a report displaying the trainer's name and the course title.

\*\* End \*\*