File Input & Output

Saving Data and Loading Data

Topics

- File Paths
- File Types
- File I/O in .NET
- Writing CSV
- Reading CSV
- NuGet
- Serializing
- Deserializing

File Paths

File Paths

- The **full path** to the file.
- Example:

Continued on next slide…

File Paths

- Full Path
  EX: C: \ temp \ 2109 \ sample.txt

- Relative Path
  relative to the working directory
  EX: ..\..\Files\Config\sample.txt

- Current Directory
  the current directory of the application
  EX: sample.txt

File Types

File Types

- These are some of the more common file types you'll encounter.
- **CSV**: **C**omma-**S**eparated **V**alues
  - Does NOT technically have to be commas. Can use any **delimiter**.
- **XML**: e**X**tensible **M**arkup **L**anguage
- **JSON**: **J**ava**S**cript **O**bject **N**otation
  - Quickly replacing XML as the preferred file format for passing data on the web and in general.

File Types

CSV: **C**omma-**S**eparated **V**alues

- “Thor,Captain America,Iron Man”
- “Thor|Captain America|Iron Man”

File Types

JSON: **J**ava**S**cript **O**bject **N**otation

```
{
  "ArrogantSupers": [
    { "Name": "Thor" },
    { "Name": "Captain America" },
    { "Name": "Iron Man" }
  ]
}
```

File Types

XML: e**X**tensible **M**arkup **L**anguage

```
<?xml version="1.0"?>
<ArrogantSupers>
  <SuperHero Name="Thor" />
  <SuperHero Name="Captain America" />
  <SuperHero Name="Iron Man" />
</ArrogantSupers>
```

File I/O in .NET

System.IO Namespace

- **System.IO** namespace
  - **File class**
    - File.CreateText
    - File.ReadAllText
    - File.Exists
  - **Directory class**
    - Directory.CreateDirectory
    - Directory.EnumerateFiles, Directory.GetFiles
    - Directory.Exists
  - **Path class**
    - Path.GetExtension, Path.ChangeExtension, Path.HasExtension
    - Path.Combine

.NET File IO Operations

- For **text** and **binary**
  - StreamWriter and StreamReader for ASCII text I/O
  - BinaryWriter and BinaryReader for binary I/O
- For **XML**
  - XmlWriter and XmlReader
- For **JSON**
  - JSON.net should be used. https://www.newtonsoft.com/json

The Using Statement

3 Steps for working with files

- There are 3 steps for working with files:
  1. Open the file
  2. Read/Write the file
  3. CLOSE THE FILE!

The using statement

- Files that are opened **must be explicitly closed**.
- They are **system resources** and should only be opened for as little time as possible.
- To ensure that they will always be closed, you should be opening files in a using statement.

```
using (StreamWriter sw = new StreamWriter("outputFile.txt"))
{
        sw.WriteLine("Batman rules!");
        sw.Write("Lesser supes: Superman, Flash, etc. ");
}
```

Writing CSV

Write CSV

- Open a file with StreamWriter in a using statement
  NOTE: you need a using System.IO; in the usings at the top of the file

```
//without a path, it will save in the same directory as the EXE file
string filePath = "sample.txt";
using (StreamWriter sw = new StreamWriter(filePath))
{
```

- Write some comma-separated data to the file

```
    sw.WriteLine("A,B,C,D,E,F");
}
```

Writing CSV Challenge

1. Create a method called WriteData that takes a filePath as a parameter.
2. Create an List of ints (fill it with any ints)
3. Using StreamWriter, save the ints to the file in CSV format (make sure you do it in a using statement)
4. Call WriteData from Main and pass the name of the file.
5. Open Windows Explorer, navigate to the file and open it.

Example:

```
//without a path, it will save in the same directory as the EXE file
string filePath = "sample.txt";
using (StreamWriter sw = new StreamWriter(filePath))
{
    sw.WriteLine("A,B,C,D,E,F");
}
```

LINKS

Using

StreamWriter

VIDEOS

Reading CSV

Read CSV

- Open a file with StreamReader in a using statement
  NOTE: you need a using System.IO; in the usings at the top of the file

```
//without a path, it will look for the file
//in the same directory as the EXE file
string filePath = "sample.txt";
using (StreamReader sr = new StreamReader(filePath))
{
```

- Read until you reach the end of the file

```
    string line;
    while ((line = sr.ReadLine()) != null)
    {
        //the data is in csv format
        //so need to split the string to get the data
        string[] lineData = line.Split(',');
    }
}
```

Read file in 1 line

- Open, Read, Close a file with File.ReadAllText

```
//reads all the contents into 1 string
string fileText = File.ReadAllText(filePath);
```

Reading CSV Challenge

1. Create a method called ReadData that takes a string parameter for the filepath.

2. Use StreamReader in a using statement to load the array of ints from the csv file.

   The data is csv data so you'll need to split it to get the individual items.

3. After splitting, convert the array to a List. Then print the list.

4. Call ReadData from Main and pass the name of the file to read.

Example:

```
//in the same directory as the EXE file
string filePath = "sample.txt";
using (StreamReader sr = new StreamReader(filePath))
{
    string line;
    while ((line = sr.ReadLine()) != null)
    {
        //the data is in csv format
        //so need to split the string to get the data
        string[] lineData = line.Split(',');
    }
}
```

LINKS

Using

StreamReader

Split

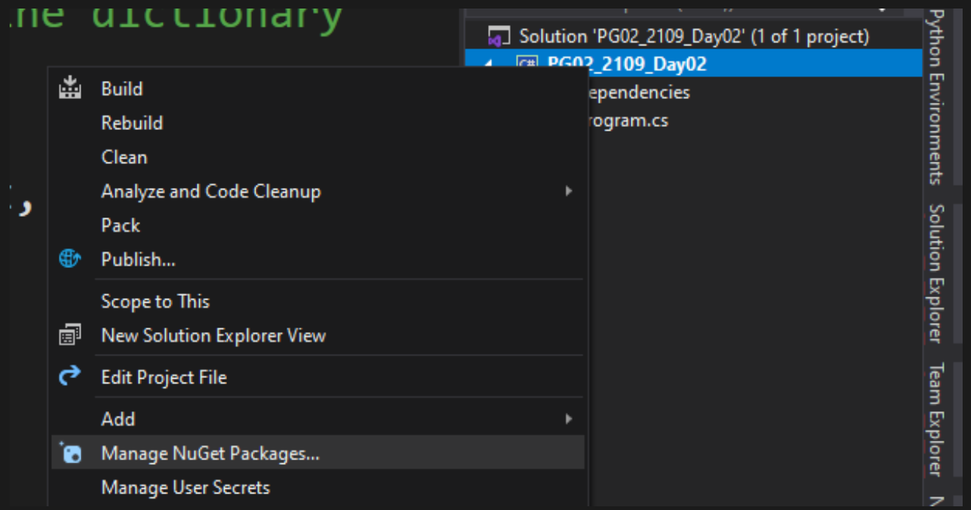
ToList

VIDEOS

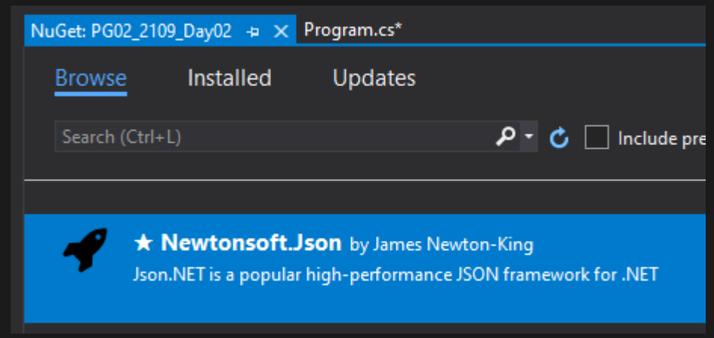
NuGet and Json.NET

Add Newtonsoft.Json

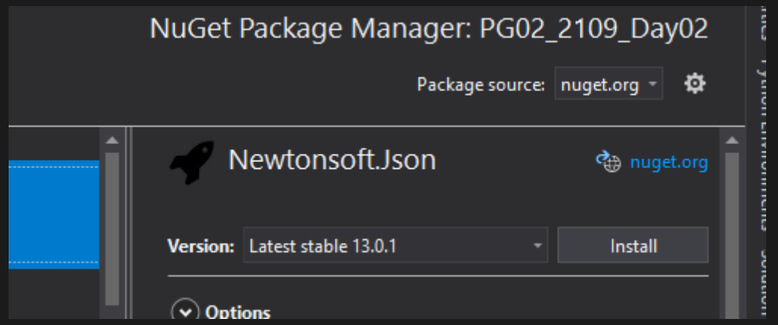
1. Right-click the project and select “Manage NuGet Packages…”



2. Select the Browse option and select the Newtonsoft.Json library



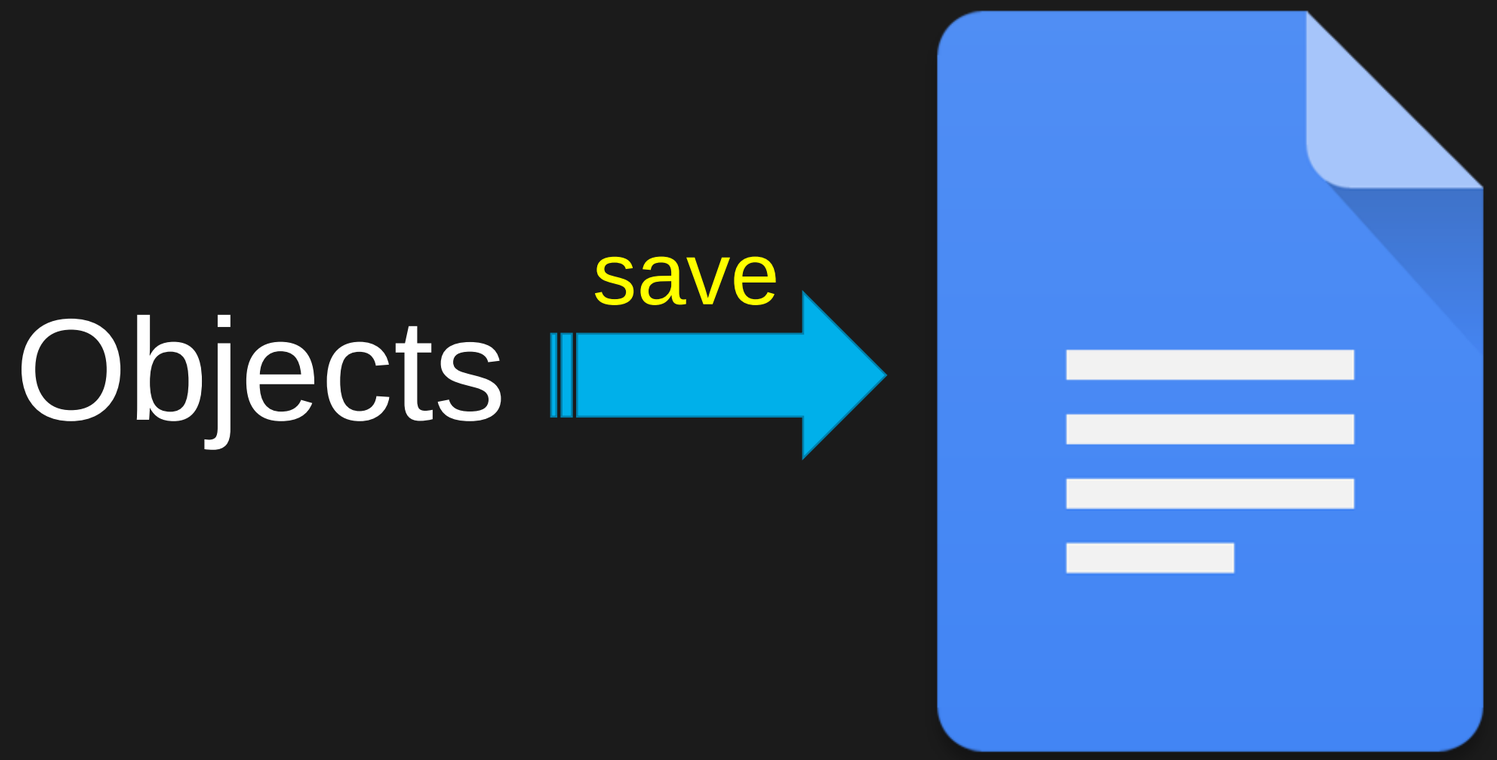
3. Click the Install button on the right of the screen.



Serializing

Serializing

- **Serializing** is the process of storing object instances to a file or stream. It stores the state of the object – the values of the properties are saved.



Serialize JSON

- Serializing saves objects so you'll need an object to serialize.
  In this example, we'll serialize the rankings object.

```
Dictionary<int, string> rankings = new Dictionary<int, string>()
{
    { 1,"Dolphins" }
};
rankings.Add(2, "Bills");
rankings[3] = "Patriots";
rankings[32] = "Jets";
```

Continued on next slide…

Serialize JSON

- Now open a StreamWriter and a JsonTextWriter
  NOTE: you'll need using Newtonsoft.Json; in the usings

```
string filePath = "sample.json"; //note the json extension
using (StreamWriter sw = new StreamWriter(filePath))
{
    using (JsonTextWriter jsonWriter = new JsonTextWriter(sw))
    {
```

- Then serialize the rankings object using the JsonSerializer.

```
        JsonSerializer serializer = new JsonSerializer();
        serializer.Serialize(jsonWriter, rankings);
    }
}
```

Serializing Challenge

1. Create a method called WriteJson that takes a filePath as a parameter.
2. Change the extension on your file path to .json
3. Using a JsonTextWriter, serialize a List of ints to the file
4. Call WriteJson from Main.

Example:

```
string filePath = "sample.json"; //note the json extension
using (StreamWriter sw = new StreamWriter(filePath))
{
    using (JsonTextWriter jsonWriter = new JsonTextWriter(sw))
    {
        JsonSerializer serializer = new JsonSerializer();
        serializer.Serialize(jsonWriter, rankings);
    }
}
```

LINKS

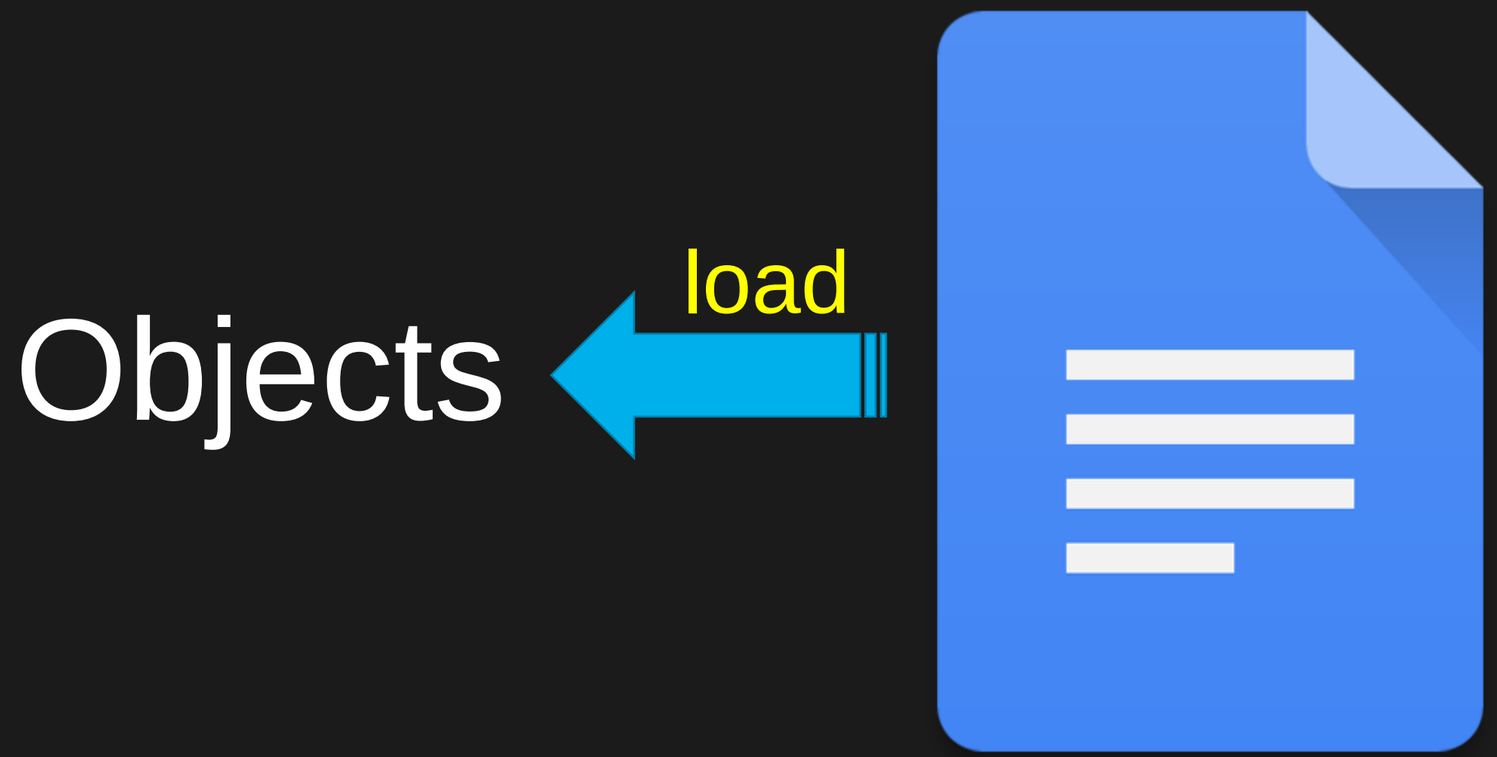
ChangeExtension

VIDEOS

Deserializing

Deserializing

- **Deserializing** is the process of reading the object state from the file and re-creating the object.



Deserialize JSON

- Before reading the file, be sure to check if the file exists.

```
if (File.Exists(filePath))
{
```

- Read the file into a string variable using File.ReadAllText

```
string jsonText = File.ReadAllText(filePath);
```

Continued on next slide…

Deserialize JSON

- Because deserializing can throw exceptions, make sure you put this code inside a try-catch.

  Deserialize using the JsonConvert object.
  NOTE: the type you pass to DeserializeObject is the type you serialized.

```
try
{
    //var saves us from typing a long type
    var savedRankings = JsonConvert.DeserializeObject<Dictionary<int, string>>(jsonText);
}
catch (Exception)
{
```

Change the extension

- To *properly* change the extension on a file path, use the Path.ChangeExtension method.

```
filePath = "sample.txt";
//to change the extension to .json
filePath = Path.ChangeExtension(filePath, ".json");
```

Deserializing Challenge

1. Create a method called ReadJson that takes a filePath as a parameter and returns a list of ints.
2. Read the contents of the file with File.ReadAllText. Make sure you check if the file exists before reading the file.
3. Using JsonConvert, deserialize the contents of the file to a List of ints. Make sure you use a try-catch to handle exceptions.
4. Print the list.
5. Call ReadJson from Main and print the list that is returned.

Example:

```
string jsonText = File.ReadAllText(filePath);
```

```
try
{
    //var saves us from typing a long type
    var savedRankings = JsonConvert.DeserializeObject<Dictionary<int, string>>(jsonText);
}
catch (Exception)
{
```

LINKS

File.Exists

File.ReadAllText

Try-catch

VIDEOS