week14

November 29, 2018

1 File I/O

- File input is reading the content of a file into memory (a variable)
- File output is writing the content of a variable or variables into a file
- File content can be text or binary
- We will be working only with text files

2 Reading a Files

- Download data1.txt and place it on the desktop
- Right click on the file then select properties
- Copy the file path

Note: File paths are different in different operating systems (e.g. mac)

3 Reading a Files

```
Dim data = My.Computer.FileSystem.ReadAllText(
                    "c:\path\to\file\data1.txt")
```

- This command will store all the data in the file in variable data
- **data** will be a string even though we did not mention that
- Try to display content of data
- **Important:** Remember to replace path\to\file with the correct path on your computer

4 Working With Lines

- Sometimes it is better to work with each line separatly
  - For example: We might store each record in a separate line

```
Dim txt = data.Split(vbNewLine) 'split the lines into a list
For Each x In txt
    'Work with each line
    MsgBox(x)
    'you can even split line and work with values
Next
```

Note: txt is now a List of Strings

5 Challenge

Try to display the list of strings in a ListBox

Look at the data, can you write a program to read the information of each student (name and grade)

Will it be easy?

What would you change about the organization of the data to make writing a program to read it easier?

6 Better Organized Data

- Look at names.txt and grades.txt
- Is it easier to write a program to read the data for each student?
- Add two lists to display the data
- Which grade belongs to which student?
  - **Important:** Data for the same student must be on the same line

7 Standard Ways of Organizing Textual Data

- This is called **file format**
- Most common file formats include XML, JSON, a CSV
  - List is probably endless

8 XML Example

```
<student>
    <name>mohammed</name>
    <grade>90</grade>
    <absence>1</absence>
</student>
<student>
    <name>Ahmed</name>
    <grade>80</grade>
    <absence>2</absence>
</student>
```

9 JSON Example

```
[
  {
        "name": "Mohammed",
        "grade": 90,
        "absence": 1
    },
    {
```

```
        "name": "Ali",
        "grade": 85,
        "absence": 2
    }
]
```

10 CSV Example

```
name,grade,absence
Mohammad,95,1
Ahmed,90,0
Khaled,85,3
Sara,80,2
```

11 Reading CSV Files

- Download the file data.csv
- Read it into variable **grades**
- Display each line in a separate MsgBox
- Instead of displaying MsgBox, we process the data line by line and extract the information
  - This is called **parsing** the text file

12 Parsing a CSV

```
Dim fileName as String = "c:\path\to\data.csv"
Dim data = My.Computer.FileSystem.ReadAllText(fileName)
Dim txt = data.Split(vbNewLine) ' split into list of string
For Each record In txt
        'split each line into its values
        Dim values = Record.split(",")
         For Each v in values
            MsgBox(v)' Here we examine the value
         Next
 Next
```

13 Parsing a CSV (another way)

```
Dim fileName as String = "c:\path\to\data.csv"
Dim data = My.Computer.FileSystem.ReadAllText(fileName)
Dim txt = data.Split(vbNewLine) ' split into list of string
For Each record In txt
        'split each line into its values
        Dim values = Record.split(",")
        MsgBox(values(0)) 'name
        MsgBox(values(1)) 'grade
```

```
        MsgBox(values(2)) 'absence
 Next
```

14 Better Way To Get Path To Special Folders

```
PathToDesktop = Environment.GetFolderPath(Environment.SpecialFolder.Desktop)
PathToUserDir = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile)
PathToMyDocuments = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)

' for path to directory on desktop
filePath = PathToDesktop & "\DirectoryName\filename"
```

15 Other Folder Tricks

```
' for file in current folder
filePath = ".\fileName"

' for file in a directory one level above current folder
filePath = "..\fileName"

' for file in a directory two levels above current folder
filePath = "..\..\fileName"
```

16 Challenge

Modify the previous program for reading data.csv to perform the following - Display 3 lists (use ListBoxes here), for names, grades, and absences - Add button to load data.csv into these lists - Add controls to allow the user to add another student record - Remember, student must have name, grade, and absence - Don't forget input validation - Add a button to save the list back into data.csv with any new data

17 Writing to A File

```
My.Computer.FileSystem.WriteAllText(filePath, data, False)
```

- **filePath** is the location of the file as a string (just like reading the file)
- **data** is a String variable that contains all your data
- **False** at the end means that you do not want to append to the file, but create a new one even if the data exists

18 Writing to A File

Create a button to perform the following action:

```
Dim filePath As String = "c:\path\to\file\myfile.txt"
Dim data = "Hello Files!"
My.Computer.FileSystem.WriteAllText(filePath, data, False)
```

Modify the filePath so that the new file is created on the desktop.
Examine the contents of the file, what do you see?
Click the button multiple times then see what happens to the file.
Replace False with True then click multiple times to see what happens to the file

19 How Do We Store CSV

Think about what we are trying to do: - We want the data for each student in a line - The line will contain name, grade, then absence - Data is separated by a comma - Order of data is important - Store it to a String variable - Store the content of the variable in a file

20 Writing To CSV

```
Dim dataList as new List(Of String)

' We need the index for all listboxes, use for loop not for each
' Add all the records to a list of string
For i = 0 to lstNames.Count() - 1
    Dim record = lstNames.items(i) & "," & lstGrades.items(i) & "," lstAbsence.items(0)
    csvData.add(record)
Next

' Combine each item in the list into a string and seperate with vbnewline
Dim dataString As String = String.Join(vbNewLine, dataList)
' Store the data
My.Computer.FileSystem.WriteAllText("c:\path\to\file\mydata.csv", dataString, False)
```

21 File Selector

- Use it as a way to let the user select the file path
- No need to remember what the path is or look for it

22 File Selector

```
' very important, added at the very top before the class
Imports Microsoft.Win32

'Create Dialog instance
Dim dlg As OpenFileDialog = New OpenFileDialog()

' If you want to save to a file, add the following line:
' dlg.CheckFileExists = False
```

```
' It allows you to select non-existent file so you can save to it

'Show the dialog
If dlg.ShowDialog Then
    'This stored the selected path by the user into fileName
    Dim fileName as String = dlg.FileName

    'You have the path, now you can read the data
    Dim data = My.Computer.FileSystem.ReadAllText(fileName)
End If
```

23 Challenge

Modify the programs you built this week to use the file selector

24 Challenge

- Create an application to allow the instructor to enter student grades only
- Calculate the statistics (min, max, average) for the grades
- Add the ability to store the data
- Add the ability to read the data
- Use file selector