Transforming Python to C# – read a file

Python

```python
@staticmethod
def read_beers(filename):
    beers = []
    fo = open(filename)
    fo.readline()  # first line = title row -> ignore
    line = fo.readline()
    while (line != ""):
        line = line.rstrip('\n')
        delen = line.split(";")

        try:
            beer = Beer(delen[1], delen[2], float(delen[4].replace(',', '.')), delen[3])
            beers.append(beer)
        except ValueError as e:
            print("The following line was not processed: " + line)

        # read next line
        line = fo.readline()

    fo.close()
    return beers
```

C#

```
public static List<Beer> GetBeers()
{
    List<Beer> beerlist = new List<Beer>();

    var assembly = typeof(Beer).GetTypeInfo().Assembly;
    Stream stream = assembly.GetManifestResourceStream("LAB01_DEMO.Assets.beerlist.csv");

    using (var reader = new System.IO.StreamReader(stream))
    {
        reader.ReadLine(); //ignore first line (title row)
        string line = reader.ReadLine();

        while (line != null)
        {
            string[] parts = line.Split(';');

            try
            {
                beerlist.Add(new Beer(parts[0], parts[1], Convert.ToDouble(parts[2]), parts[3]));
            }
            catch (Exception)
            {
                Debug.WriteLine("error processing line: " + line);
            }
            line = reader.ReadLine();
        }
    }

    return beerlist;
}
```

- To get the code above to be compiled, we need to add two extra using statements:

```
using System.IO;
using System.Reflection;
```

  - ✓ System.IO stands for **I**nput/**O**utput, and is required to use the StreamReader object.
  - ✓ System.Reflection enables code to be compiled at runtime, and is needed to get the assembly info.
  - ✓

```
var assembly = typeof(Beer).GetTypeInfo().Assembly;
Stream stream = assembly.GetManifestResourceStream("LAB01_DEMO.Assets.beerlist.csv");
```

    For now, all you have to understand from these lines of code, is that:
  - ✓ **typeof(…)** needs the `<classname>`, and
  - ✓ **GetManifestResourceStream(…)** needs a string of the following format:
    "`<project_name>.<foldername>.<filename_with_extension>`"
    Eg. in this case:

<project_name>	LAB01_DEMO
<foldername>	the csv file is in a folder named Assets
<filename_with_ext>	beerlist.csv

- The '**using**' block makes sure the file is seized on by the program, and is automatically closed / released at the end of that block.

Xamarin specific: display using layout

We will display the list of beers by making use of a **ListView** control. This is a control that's used to display a list of objects.

MainPage.xaml

In your xaml file (= visual layout), you can find the ListView control that was added:

```
<ListView x:Name="lstBeers" />
```

- The ListView gets a name to make it accessible from code

MainPage.xaml.cs

The code behind file will now use the name of the list to fill it with data

```
List<Beer> allBeers = Beer.GetBeers();
lstBeers.ItemsSource = allBeers;
```

- The **ItemsSource** property contains the collection of items the ListView should display

The result

If you start the application, you can see a list of beers. The details of the visualization (such as colors) can vary among the different operation systems and are based on the settings of the device.