Configuration

AARHUS UNIVERSITY
AARHUS UNIVERSITY SCHOOL OF ENGINEERING
MICHAEL LOFT
ML@ASE.AU.DK

Agenda

Why do we need configuration files?

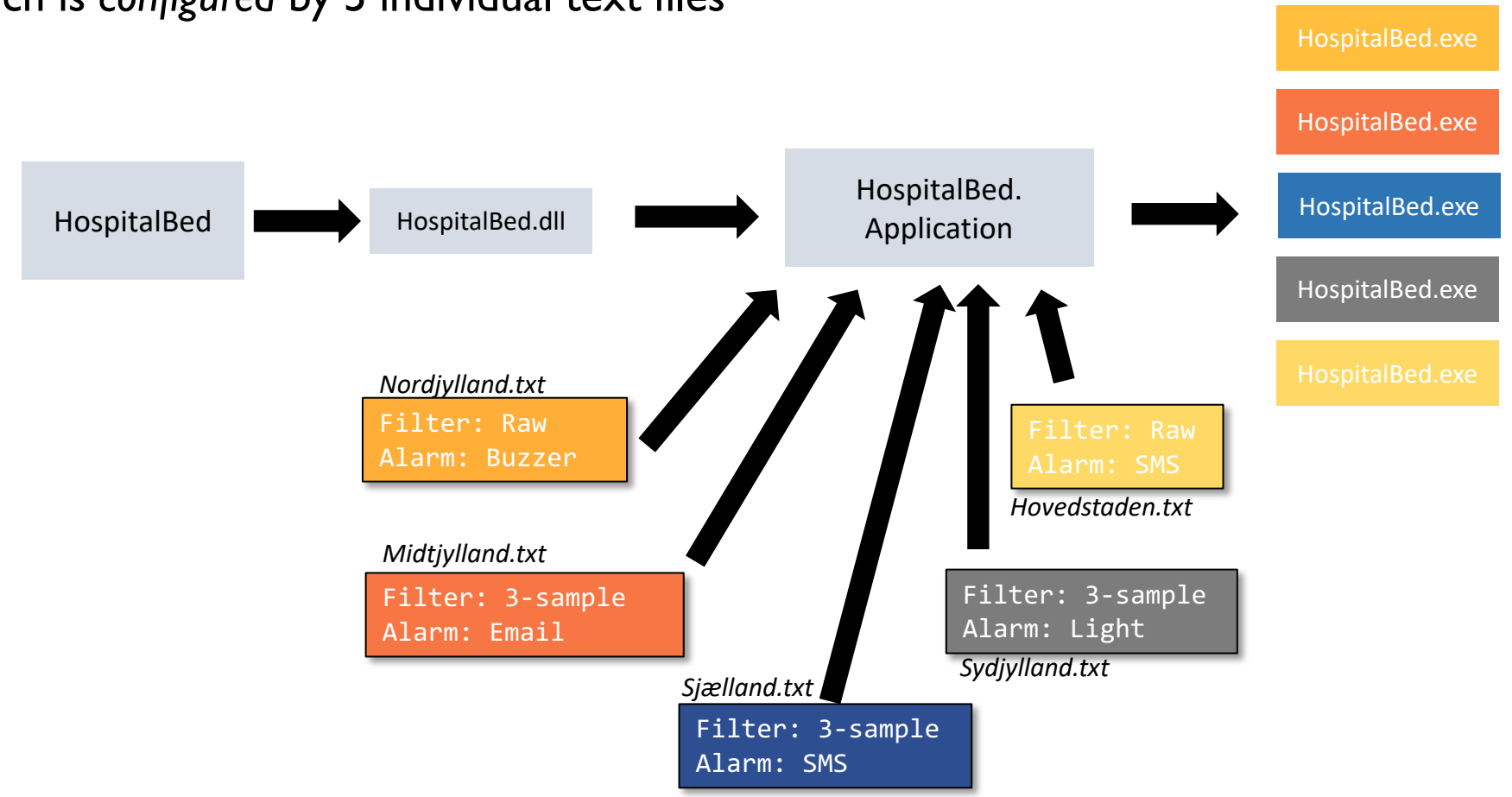
How to read and write configuration files

Default configuration

Why do we need configuration files?

Simple Factory – an example

1 application library containing all code, referenced by 1 application project which is *configured* by 5 individual text files



Or simply user settings

Ways to work with configuration in .NET

System.Configuration

Read and write key/value pairs in Web.Config or App.Config (xml files)

XMLSerializer

Serialize and de-serialize objects to XML
data is organized as a tree

JsonSerializer

Serialize and de-serialize objects to JSON
data is organized as objects

Microsoft.Extensions.Configuration

File format can be JSON, XML, ini or custom
all configuration is read as keys/value pairs
can't write to files... ☹

Structuring the configuration

```
[Information]
AppName = Configuration Demo ini

[UserSettings]
TimeFormat = 24
```

INI files:

Single level of grouping.

```
{
  “Information” : {
     "AppName" : "Configuration Demo Json“
  },
  “UserSettings” : {
     “TimeFormat” : 24
  }
}
```

Json:

A tree, in the form of object nested in objects.

```
<root>
  <Information>
    <AppName>Conf Demo XML</AppName>
  </Information>
  <UserSettings>
    <TimeFormat>24</TimeFormat>
  </UserSettings>
</root>
```

XML:

A tree, in the form of xml tags inside xml tags.

JsonSerializer

JsonSerializer – Configuration object

```
public class AlarmSettings
{
    public enum TimeFormatEnum
    {
        H12,
        H24
    }

    public TimeFormatEnum TimeFormat { get; set; }
    public bool AlarmIsOn { get; set; }
    public string Locale { get; set; }
}
```

The XMLSerializer can be used to save/load an object graph.

Create an object with public properties.

There **must** be a parameterless constructor. But you can add other constructors.

```
{
  "TimeFormat":0,
  "AlarmIsOn":false,
  "Locale":"DK”
}
```

JsonSerializer – Save the configuration

```
public void Save(AlarmSettings alarmSettings,
                 string path)
{
    string json =
        JsonSerializer.Serialize(alarmSettings);
    File.WriteAllText(path, json);
}
```

Use the DataContractJsonSerializer to save the object as Json.

```
{
  "TimeFormat":0,
  "AlarmIsOn":false,
  "Locale":"DK”
}
```

JsonSerializer – Load the configuration

```
public AlarmSettings Load(string path)
{
    string text = File.ReadAllText(filename);
    AlarmSettings alarmSettings =
      JsonSerializer.Deserialize<AlarmSettings>(text);
    return alarmSettings;
}
```

Use the JsonSerializer to load the object from Json.

```
{
 "AlarmIsOn":true,
 "Locale":"US",
 "TimeFormat":2
}
```

JsonSerializer – Lists

```
public class NestedConfigurations
{
    ...
    public List<AlarmSettings> alarms { get; set; }
    ...
}
```

Lists can also be serialized.

```
[
 {
  "TimeFormat":0,
  "AlarmIsOn":false,
  "Locale":"DK”
 },
 {
  "TimeFormat":1,
  "AlarmIsOn":true,
  "Locale":"DK”
 }
]
```

JsonSerializer - Attributes

```
public class AlarmSettings
{
    public enum TimeFormatEnum
    {
        H12,
        H24
    }

    public TimeFormatEnum TimeFormat { get; set; }
    [JsonPropertyName("IsOn")]
    public bool AlarmIsOn { get; set; }
    public string Locale { get; set; }
    [JsonIgnore]
    public bool snoozed {get; set; }
}
```

Attributes can be used to change the Json element name or ignore a class field.

```
{
 "IsOn":true,
 "Locale":"US",
 "TimeFormat":2
}
```

https://docs.microsoft.com/en-us/dotnet/standard/serialization/controlling-xml-serialization-using-attributes

JsonSerializer – Error handling

The JsonSerializer throws an exception, if the Json can't be de-serialized to the object graph.

You need to handle this in the code.

A good way:

Provide the user with an error message, create a valid configuration file, which the user can modify and exit the program.

Don't overwrite the bad configuration file.

Create something like "**hospitalbed.config.default**" instead.

XMLSerializer

XMLSerializer – Save the configuration

```
public static void Save(Configuration configuration,
                        string path)
{
    FileStream fs = new FileStream(path, FileMode.Create);

    XmlSerializer serializer =
                new XmlSerializer(typeof(Configuration));

    serializer.Serialize(fs, configuration);
    fs.Close();
}
```

Use the XMLSerializer to save the object as XML.

```
<?xml version="1.0"?>
<Configuration
xmlns:xsi="http://www.w3.org/2001/XMLS
chema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLS
chema">
  <TimeFormat>H24</TimeFormat>
  <AlarmIsOn>false</AlarmIsOn>
  <Locale>US</Locale>
</Configuration>
```

XMLSerializer – Load the configuration

```
public static Configuration Load(string path)
{
    FileStream fs = new FileStream(path,
                                   FileMode.Open);

    XmlSerializer serializer =
       new XmlSerializer(typeof(Configuration));

    Configuration configuration =
       (Configuration)serializer.Deserialize(fs);

    fs.close();
    return configuration;
}
```

Use the XMLSerializer to load the object from XML.

```
<?xml version="1.0"?>
<Configuration
xmlns:xsi="http://www.w3.org/2001/XMLS
chema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLS
chema">
  <TimeFormat>H24</TimeFormat>
  <AlarmIsOn>false</AlarmIsOn>
  <Locale>US</Locale>
</Configuration>
```

XMLSerializer – Lists

```
public class NestedConfiguration
{
    ...
  public List<AlarmConfiguration> Alarms { get; set; }
                    = new List<AlarmConfiguration>();
    ...
}

public class AlarmConfiguration
{
  public AlarmConfiguration()
  {
  }

  public int Hour { get; set; }
  public int Minute { get; set; }
  public bool Active { get; set; }
}
```

Lists can also be serialized.

```
<?xml version="1.0"?>
<NestedConfiguration
xmlns:xsi="http://www.w3.org/2001/XMLS
chema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLS
chema">
  <Alarms>
    <AlarmConfiguration>
      <Hour>6</Hour>
      <Minute>20</Minute>
      <Active>true</Active>
    </AlarmConfiguration>
    <AlarmConfiguration>
      <Hour>6</Hour>
      <Minute>30</Minute>
      <Active>false</Active>
    </AlarmConfiguration>
  </Alarms>
</NestedConfiguration>
```

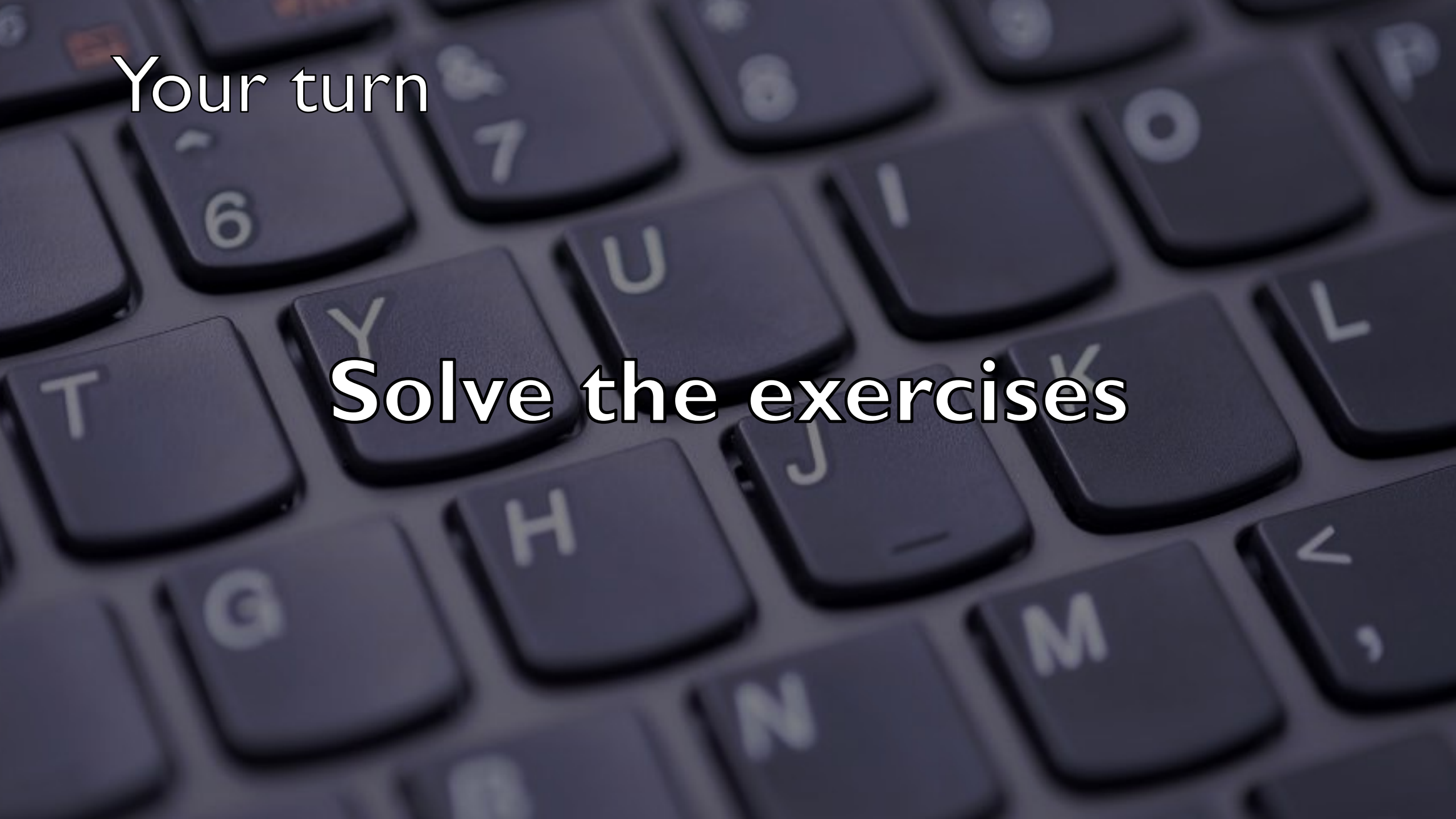
XMLSerializer - Attributes

```
public class TaxRates{

    [XmlElement(ElementName = "TaxRate")]
    public decimal ReturnTaxRate;

    // The XmlSerializer ignores this field.
    [XmlIgnore]
    public string Comment;
}
```

Attributes can be used to change the XML element name or ignore a class field.

https://docs.microsoft.com/en-us/dotnet/standard/serialization/controlling-xml-serialization-using-attributes

Your turn



Solve the exercises

Extra stuff – just for informational purpose!

Microsoft.Extensions.Configuration

Microsoft.Extensions.Configuration

Installation through NuGet package manager.

```
Install-Package Microsoft.Extensions.Configuration -Version 5.0.0
Install-Package Microsoft.Extensions.Configuration.FileExtensions -Version 5.0.0
Install-Package Microsoft.Extensions.Configuration.Json -Version 5.0.0
Install-Package Microsoft.Extensions.Configuration.Xml -Version 5.0.0
Install-Package Microsoft.Extensions.Configuration.Ini -Version 5.0.0
Install-Package Microsoft.Extensions.Configuration.Binder -Version 5.0.0
```

Key/Value access

```
var applicationName = configuration["AppName"];

Console.WriteLine("Application name is {0}", applicationName);
```

Reading a JSON file

```
class JsonConfiguration
{
    public void ReadConfiguration()
    {
        var configurationBuilder = new ConfigurationBuilder();
        var currentDirectory = Directory.GetCurrentDirectory();
        string configPath = Path.Combine(currentDirectory,
                                "ConfigurationDemo.json.config");

        IConfigurationRoot configuration = configurationBuilder
            .AddJsonFile(configPath, optional: false)
            .Build();

        var applicationName = configuration["AppName"];

        Console.WriteLine("Application name is {0}",
                          applicationName);
    }
}
```

```
{
 "AppName" : "Configuration Demo Json"
}
```

Reading an XML file

```
class XmlConfiguration
{
    public void ReadConfiguration()
    {
        var configurationBuilder = new ConfigurationBuilder();
        var currentDirectory = Directory.GetCurrentDirectory();
        string configPath = Path.Combine(currentDirectory,
                                "ConfigurationDemo.xml.config");

        IConfigurationRoot configuration = configurationBuilder
            .AddXmlFile(configPath, optional: false)
            .Build();

        var applicationName = configuration["AppName"];

        Console.WriteLine("Application name is {0}",
                          applicationName);
    }
}
```

```
<root>
  <AppName>Conf Demo XML</AppName>
</root>
```

Reading an INI file

```
class IniConfiguration
{
    public void ReadConfiguration()
    {
        var configurationBuilder = new ConfigurationBuilder();
        var currentDirectory = Directory.GetCurrentDirectory();
        string configPath = Path.Combine(currentDirectory,
                                "ConfigurationDemo.ini.config");

        IConfigurationRoot configuration = configurationBuilder
            .AddIniFile(configPath, optional: false)
            .Build();

        var applicationName = configuration["AppName"];

        Console.WriteLine("Application name is {0}",
                          applicationName);
    }
}
```

```
AppName = Configuration Demo ini
```

References and image sources

Images:

Alarm clock: https://upload.wikimedia.org/wikipedia/commons/e/eb/Digital-clock-alarm.jpg

Computer keyboard: http://stockmedia.cc/computing_technology/slides/DSD_8790.jpg

Bonus: http://wjreviews.com/reviews-cta/bonus.png

AARHUS UNIVERSITY