Programming in C# Jump Start

Jerry Nixon | Microsoft Developer Evangelist
Daren May | President & Co-founder, Crank211

07 | Advanced C#, Part 3

Jerry Nixon | Microsoft Developer Evangelist
Daren May | President & Co-founder, Crank211

Module Overview

- Interacting with the File System
- Working with REST Services

Interacting with the File System

MVA Microsoft Virtual Academy

Microsoft

Why read or write to the file system?

- Show existing data to user
- Integrate user-provided data
- Serialize objects out of memory
- Persist data across sessions
- Determine environment configuration

How do we write to files? (027)

- This is simplified with *Framework methods*; open / shut
  - File.WriteAllText / ReadAllText
- Open *for reading* to keep open and keep writing
- Open *as stream* for large payloads and realtime processing

```
var dir = System.IO.Directory.GetCurrentDirectory();
var file = System.IO.Path.Combine(dir, "File.txt");
var content = "how now brown cow?";

// write
System.IO.File.WriteAllText(file, content);

// read
var read = System.IO.File.ReadAllText(file);
Trace.Assert(read.Equals(content));
```

How do we find files? (027)

- Get Windows folder with *Environment Special Folders*
- Get the Current folder with File.IO *Get Current Directory*()
- Use *Isolated Storage* dedicated to the current application
- *Anything Else*. Caveat: Windows Store App development

```
// special folders
var docs = Environment.SpecialFolder.MyDocuments;
var app = Environment.SpecialFolder.CommonApplicationData;
var prog = Environment.SpecialFolder.ProgramFiles;
var desk = Environment.SpecialFolder.Desktop;

// application folder
var dir = System.IO.Directory.GetCurrentDirectory();

// isolated storage folder(s)
var iso = IsolatedStorageFile
    .GetStore(IsolatedStorageScope.Assembly, "Demo")
    .GetDirectoryNames("*");

// manual path
var temp = new System.IO.DirectoryInfo("c:\temp");
```

How do we modify files? (027)

- Iterate through files using GetFiles()
- Rename / Move with System.IO methods
- Get File Info with Syetem.UI.FileInfo

```csharp
// files
foreach (var item in System.IO.Directory.GetFiles(dir))
    Console.WriteLine(System.IO.Path.GetFileName(item));

// rename / move
var path1 = "c:\temp\file1.txt";
var path2 = "c:\temp\file2.txt";
System.IO.File.Move(path1, path2);

// file info
var info = new System.IO.FileInfo(path1);
Console.WriteLine("{0}kb", info.Length / 1000);
```

Leveraging Web Services

MVA Microsoft Virtual Academy

Microsoft

What are Web Services?

- Web Services encapsulate implementation
- Web Services expose to disparate system
- Web Services allow client systems to communicate servers
  - Web protocols (HTTP, GET, POST, etc)
- Web Services are important to Service Oriented Architecture
  - With and without metadata
  - Loose coupling

What is SOAP?

- SOAP is a standard for returning structured data from a Web Service as XML
  - Envelope
    - Header
    - Body
- SOAP handling is a built-in feature of Visual Studio

```xml
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

What is REST?

- REST is becoming a common, industry standard
- REST does not require XML parsing
- REST does not require a message header
- REST is generally human-readable
- REST uses less bandwidth thank SOAP
- REST services typically return XML or JSON
- JSON is JavaScript Object notation
  - JSON is becoming a common, industry standard
  - JSON is generally a lighter payload than XML (or SOAP)

```
// fetch data (as JSON string)
var url = new Uri("http://localhost:1234/MyService.svc/json/4");
var client = new System.Net.WebClient();
var json = await client.DownloadStringTaskAsync(url);

// deserialize JSON into objects
var serializer = new JavaScriptSerializer();
var data = serializer.Deserialize<JSONSAMPLE.Data>(json);

// use the objects
Console.WriteLine(data.Number);
foreach (var item in data.Multiples)
    Console.Write("{0}, ", item);
```

DEMO

Microsoft

REST Services and Serialization

Asynchronous Programming

MVA Microsoft Virtual Academy

Microsoft

What is asynchronous programming?

- Asynchronous maximizes resources on multicore systems, by allowing units of work to be separated and completed.
- Asynchronous programming frees up the calling system, especially a user interface, as to not wait for long operations.

What is the C# ASYNC/AWAIT keywords?

- *Async* and *await* simplify asynchronous programming.
- *Async* and *await* allow asynchronous code to resemble the structure of synchronous code.
- Methods marked with *async* may return *Task*<*T*>.
- The *async* keyword instructs the compiler to allow *await*.
- The *await* keyword instructs the method to return.
- The *await* keyword instructs the compiler to resume execution within the same context after the operation is complete.

```csharp
public event EventHandler<DownloadStringCompletedEventArgs> Completed;

void GetHtml(string url)
{
    var client = new WebClient { };
    client.DownloadStringCompleted += client_DownloadStringCompleted;
    client.DownloadStringAsync(new Uri(url));
}

void client_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
{
    if (Completed != null)
        Completed(this, e);
}
```

```
async Task<string> GetHtml(string url)
{
    var client = new WebClient { };
    var html = await client.DownloadStringTaskAsync(url);
    return html;
}
```

Module Recap

- Interacting with the File System
- Working with REST Services

Microsoft