Class Guidelines

MASTERS 2020
NOW A DIGITAL EVENT

- **This class, *MSS-501/MSS-502* will be recorded.**
- **The video & the PowerPoint will be posted to Crestron Online Help ID 2015.**

Crestron Masters 2020 presentations and videos

- **Please use the lecture-qa channel for general questions about the class**
- **Please use the masters-socializing channel for non class related conversations.**
- **Please use each lab/assignment channel for questions relating to the lab/assignment.**
- **You are welcome to stay after the presentation to continue with Questions and Answers not covered during the course.**

1 CRESTRON

1

Crestron Masters 2020 – MSS-501 / 502

2 2 CRESTRON

2

Anatomy of a SIMPL#Pro Program

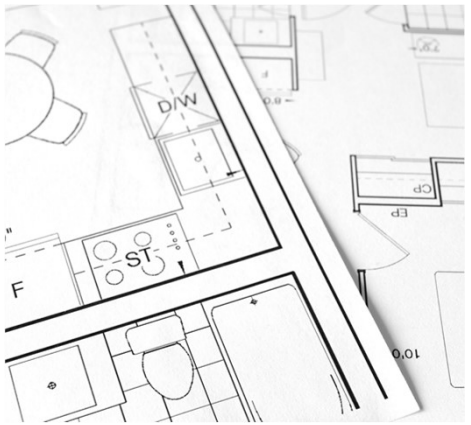
- **Inherit from CrestronControlSystem**
- Constructor
- InitializeSystem
- Event Handlers
- Registering Devices
- Handling Data

3 CRESTRON

3

Constructor

- **Initialize the max number of threads (no more than 400)**
- **Cannot Send/Receive data**
- **Make sure it's in a try/catch**
- **Has to exit in a timely fashion (20s)**
- **Use it to:**
- Register Devices
- Register EventHandlers
- Add Console Commands



4 CRESTRON

4

InitializeSystem()

- **Think of this as the first solution in logic**
- **Make sure it's in a try/catch**
- **Has to exit in a timely fashion (20s)**
- **Use it to:**
- Start threads
- Configure COM and Versiports
- Start / Initialize socket connections
- Send Initial device configurations

5

CRESTRON

5

System Event Handlers

- **SystemEventHandler**
- DiskInserted, DiskRemoved, Rebooting
- **ProgramStatusEventHandler**
- Stopping, Paused, Resumed
- **EthernetEventHandler**
- LinkUp, LinkDown

6

CRESTRON

6

Event Handlers

- **Incoming events from devices**
- **Incoming joins from touch screens / remotes / EISC**
- **Exit out of event handler quickly, as it won't trigger again until exited**
- If needed, start a thread to process and exit eventhandler

7

CRESTRON

7

Registering Devices

To use a device, registration is needed:

Exception:

- System Monitor
- Internal EX Gateway

Ports on control system:

- IR: Register the slot
- Everything else: Register individual port

EX Gateway

- Internal: register each device
- External: add device, register gateway

DM

- Add all cards and endpoints
- Register switch

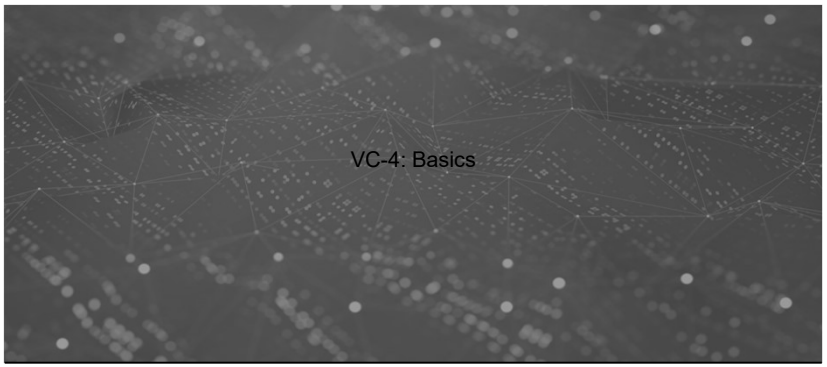
Always check results!
&
Unregister when the device is no longer needed!

8

CRESTRON

8

Crestron Masters 2020 – MSS-501 / 502



9

CRESTRON

9

VC-4 Basics

Demo

10

CRESTRON

10

Explanation of the Lab

[Yuri]

11

CRESTRON

11

Crestron Masters 2020 – MSS-501 / 502

12

CRESTRON

12

Monday's Lab

13

CRESTRON

13

Threads

A thread in computer science is short for a thread of execution. Threads are a way for a program to divide (termed "split") itself into two or more simultaneously (or pseudo-simultaneously) running tasks.

https://simple.wikipedia.org/wiki/Thread_(computer_science)

14

CRESTRON

14

Threads

In computer science, a thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system. The implementation of threads and processes differs between operating systems, but in most cases a thread is a component of a process. Multiple threads can exist within one process, executing concurrently and sharing resources such as memory, while different processes do not share these resources. In particular, the threads of a process share its executable code and the values of its dynamically allocated variables and non-thread-local global variables at any given time.

https://en.wikipedia.org/wiki/Thread_(computing)

15

CRESTRON

15

Threads

Different Operating System, Different Behavior:

- 3-Series: Priorities
- 4-Series / VC-4: Fairness
- 3-Series: Single core
- 4-Series / VC-4: Multi core

Threading is Challenging:

- Increased complexity
  - Programming
  - Debugging
  - Testing
- Contention: Sharing resources between threads

Threading is Necessary!

16

CRESTRON

16

Crestron Threads

Namespace: Crestron.SimplSharpPro.CrestronThread
Class: Thread
Constructor: Thread(<callback function>, <object>)
Constructor: Thread(<callback function>, <object>, <option>)
Options: Running or Suspended
Properties:
- Name
- Priority
- ThreadState

17 CRESTRON

17

Crestron Threads

Namespace: Crestron.SimplSharpPro.CrestronThread
Class: Thread

Methods:
- Abort()
- Join()
- Start()

18 CRESTRON

18

Contention

In order to prevent contention to resources between threads, you need to manage it:
- Prevent access to resources from other threads
  - Crestron Critical Section: CCriticalSection
  - Crestron Mutex: CMutex
  - Crestron Named Mutex: CNamedMutex
  - Object.Lock
  - System.Threading.Mutex
  - System.Threading.Semaphore

19 CRESTRON

19

Signaling between Threads

Sometimes you will need to signal between threads:
- Events (CEvent)
- Named Event: (CNamedEvent)

20 CRESTRON

20

Thread Synchronization

Thread.Join()

- Make one thread wait on another thread to finish.
- From a main thread, create a worker thread that processes one part independently while the main thread continues to process another part. However, before the main thread can return the full results, it has to wait for the worker thread to finish.

21

CRESTRON

21

Files

Accessing Files

- 3-Series: \ and case **insensitive**
- 4-Series and VC-4: / and case **sensitive**

Folder Path

- `Crestron.SimplSharp.CrestronIO.Directory`
  - `GetApplicationDirectory()`
    - On 3-Series: \SIMPL\APPXX
    - On 4-Series: /simpl/appxx
    - On VC-4: RunningPrograms/[RoomID]/App
  - `GetApplicationRootDirectory()`
    - On 3-Series: \
    - On 4-Series: /
    - On VC-4: RunningPrograms/[RoomID]/

22

CRESTRON

22

In, or out of the sandbox?

- **In the sandbox:**
- Crestron handles capitalization of system folders in the path
- Works on 3- and 4-Series
- Crestron simplifies file access
- **Out of the sandbox:**
- File paths are not handled
- Plenty of examples around
- Use 3rd party libraries (although, you may have to deal with paths)

23

CRESTRON

23

Read Data

- Easy:
  - ```
    String x = File.ReadToEnd(
                Directory.GetApplicationRootDirectory() +
                "/User/toine.txt", Encoding.ASCII);
    ```
- More Advanced:
  - ```
    FileStream stream = File.OpenRead(
                            Directory.GetApplicationRootDirectory() +
                            "/User/toine.txt", Encoding.ASCII);
    StreamReader reader = new StreamReader(stream);
    String x = reader.ReadToEnd();
    reader.Close();
    reader.Dispose();
    ```

24

CRESTRON

24

Write Data

```
• StreamWriter writer = File.AppendText(
                          Directory.GetApplicationRootDirectory() +
                          "/User/toine.txt", Encoding.ASCII);
  writer.WriteLine("Masters 2020 is awesome");
  writer.Close();
  writer.Dispose();
```


Using using

```
Using (StreamWriter writer = File.AppendText(filePath))
{
     writer.WriteLine(data);
}
```


Checking if a File Exists

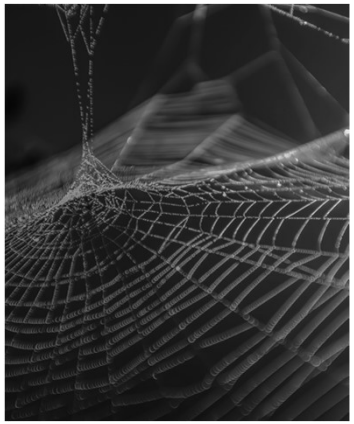
```
if (!File.Exists(filePath))
{
     ErrorLog.Error(LogHeader + "Config file doesn't exist");
}
```


Best Practice

- **Always put file operations in try / catch blocks**
- **If multiple threads are accessing a file, use locks, critical sections, etc. to prevent concurrent access.**

Crestron Web Scripting

- **Make interactive configuration pages, use CWS as the back-end script**
- **Expose a REST API for your application**



29

CRESTRON

29

Server

- **HttpCwsServer**
- HttpCwsServer(<path>);
- ReceivedRequestEvent
- HttpRequestHandler
- Register()

The HttpCwsServer is the entry into CWS scripting, use a specific path.

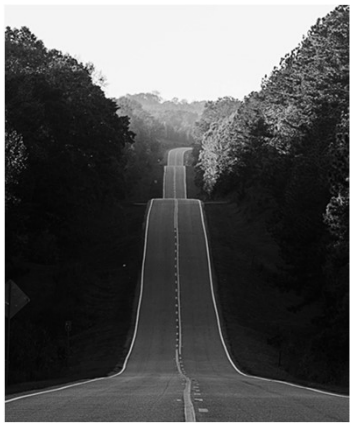
30

CRESTRON

30

Route

- **HttpCwsRoute**
- HttpCwsRoute(<url>)
- Name
- RouteHandler



31

CRESTRON

31

Handler

- **IHttpCwsHandler**
- ProcessRequest(context)
- **Order of Events:**
- Server.ReceivedRequestEvent (always)
- Server.HttpRequestHandler (For routes without their own handler)
- Route.HttpRequestHandler (For routes with handler)

32

CRESTRON

32

Examples

```
try
{
    myServer = new HttpCwsServer("api");  //instantiate the server
    myServer.ReceivedRequestEvent += MyServerOnReceivedRequestEvent; // register the generic event handler for myServer
    myServer.HttpRequestHandler = new ServerHandler(); // register the IHttpCwsHandler class for myServer

    myRoute = new HttpCwsRoute( url: "test2"); // create a route
    myRoute.Name = "TEST2"; // give the route the name TEST2
    myRoute.RouteHandler = new RouteHandler(); // register the IHttpCwsHandler class for this route
    myServer.AddRoute(myRoute); // add the route to myServer

    // add a new route, with its own IHttpCwsHandler to myServer in a single line
    myServer.AddRoute(new HttpCwsRoute( url: "test") {Name = "TEST", RouteHandler = new RouteHandler2()});

    // add a new route, utilizing the server's handlers to myServer
    myServer.AddRoute(new HttpCwsRoute( url: "test/{parm}") {Name = "TESTWITHPARM"});

    // register myServer. Without this, the routes are not live.
    myServer.Register();
}
```


ReceivedRequestEvent Example

```
1 usage
private void MyServerOnReceivedRequestEvent(object sender, HttpCwsRequestEventArgs args)
{
    // this event triggers for all requests to the server, for any URL under the base URL.
    // If you use this, you don't have to user the myServer.HttpRequestHandler
    // even if routes have their own handlers, this event will trigger. It's really the catch-all.

    ErrorLog.Notice( message: "URL is: {0}", args.Context.Request.Url);
}
```


HttpRequestHandler Example for the Server

```
1 usage
public class ServerHandler: IHttpCwsHandler
{
    public void ProcessRequest(HttpCwsContext context)
    {
        // this method gets triggered for any request to the server that isn't caught by a route's specific HttpRequestHandler
        // the main difference with the ReceivedRequestEvent, is that the args are replaced by just the context. There is no
        // difference in available data between the two
        if (context.Request.RouteData != null && context.Request.RouteData.Route != null)
            ErrorLog.Error( message: "[TCL] - ServerHandler: Route: {0}, Url: {1}",  params args: context.Request.RouteData.Route.Name,
                context.Request.Url);
        else
        {
            ErrorLog.Error( message: "[TCL] - ServerHandler: Route: NULL, Url: {0}", context.Request.Url);
        }
    }
}
```


HttpRequestHandler Example for the Route

```
1 usage
public class RouteHandler : IHttpCwsHandler
{
    public void ProcessRequest(HttpCwsContext context)
    {
        // this method gets triggered for any request that matches the exact route path. This means that if you define a route as
        // "test", it will just route api/test, not api/test/test.txt
        // if you define your route as "/test/{id}" it will match api/test/text but not api/test/test.txt
        // anything not caught by this handler will be passed to the server's handler.
        if (context.Request.RouteData != null && context.Request.RouteData.Route != null)
            ErrorLog.Error( message: "[TCL] - RouteHandler: Route: {0}, Url: {1}",  params args: context.Request.RouteData.Route.Name,
                context.Request.Url);
        else
        {
            ErrorLog.Error( message: "[TCL] - Route Handler: Route: NULL, Url: {0}", context.Request.Url);
        }
    }
}
```

HTTP Methods

HTTP uses verbs to indicate intent:

```
if (context.Request.HttpMethod == "GET")
    ErrorLog.Info( message: "Retrieving Data");
if (context.Request.HttpMethod == "POST")
    ErrorLog.Info( message: "Creating Data");
if (context.Request.HttpMethod == "PUT")
    ErrorLog.Info( message: "Updating Data");
if (context.Request.HttpMethod == "DELETE")
    ErrorLog.Info( message: "Deleting Data");
```

37

CRESTRON

37

Responding to Requests

Use the context.Response Object:

```
context.Response.AppendHeader( name: "Header Name",  value: "Header Value");
context.Response.ContentType = "application/json";
context.Response.StatusCode = 200;
context.Response.StatusDescription = "OK";
context.Response.Write( s: @"{""field"": ""value""",  final: true);
```

38

CRESTRON

38