Writing Apps, Drivers, and Scouts for the LoT Hub

David Lachut

February 19, 2014

Agenda

First Things

Make an App

Make a Driver

Make a Scout

RTFM

Dude, you're not Dr Banerjee ...

- My name is David
- PhD student since 2011
- Working on HomeOS/LoT Hub since pre-Alpha
- dlachut1@umbc.edu

Terms

Port	An access point for services
Role	An agreed upon description of what a service can do
Module	A software addin to Hub that accesses Ports
App	A module that consumes services from Ports
Driver	A module that provides services via Ports
Scout	A helper program to scan for hardware and activate a driver
Platform	The control layer between Apps, Drivers, and Scouts

- Does everyone have their Dev environment set-up with dependencies installed?

The rest of the lecture

- 1h 15m is not enough time to thoroughly cover all the necessary details
- Even if it were, you probably wouldn't learn it without working along with me
- This talk will be an overview to make it easier when you read the docs

Agenda

First Things

Make an App

Make a Driver

Make a Scout

RTFM

Overview

- Create a VS Project
- Set Names, Properties, and References
- Implement the Abstract Methods

Create a VS Project

- In Apps section of the solution
- Add a C# Class Library
- Place in "Apps" directory
- Override the Abstract Methods

Set Names, Properties, and References

- Source: Rename the class
- Source: Rename the Namespace
- Source: Add ‘using’ declarations
- Source: Add property declaration
- Source: Inherit from ModuleBase
- Source: Generate Stubs for Abstract Methods

Set Names, Properties, and References

- References: Add ‘Common’, ‘Views’, and ‘DataStore’
- References: Add ‘AddIn’, ‘ServiceModel’, and ‘ServiceModel.Web’

Set Names, Properties, and References

- Properties: Set the output path
- Properties: Set the Assembly Name
- Properties: Set the Default namespace

Override the Abstract Methods

- `Start()` Entry point for the App
- `Stop()` Clean up after the App
- `PortRegistered(VPort port)`
  Called when a port is instantiated somewhere on the hub
- `PortDeregistered(VPort port)`
  Called when a port is removed from the hub
- `OnNotification(string Rolename, string OpName, IList<VParamType> retVals, VPort senderPort)`
  Called when a notification is issued from a subscribed port

Agenda

First Things

Make an App

Make a Driver

Make a Scout

RTFM

Overview

- Create a VS Project
- Set Names, Properties, and References
- Implement the Abstract Methods

Create a VS Project

- In Drivers section of the solution
- Add a C# Class Library
- Place in “Drivers” directory
- Override the Abstract Methods

Set Names, Properties, and References

- Source: Rename the class
- Source: Rename the Namespace
- Source: Add ‘using’ declarations
- Source: Add property declaration
- Source: Inherit from ModuleBase
- Source: Implement abstract Methods
- References: Add ‘Common’, ‘Views’, and ‘DataStore’
- References: Add ‘AddIn’, ‘ServiceModel’, and ‘ServiceModel.Web’
- Properties: Set the output path
- Properties: Set the Assembly Name
- Properties: Set the Default namespace

Override the Abstract Methods

- `Start()` Entry point for the App
- `Stop()` Clean up after the App
- `PortRegistered(VPort port)`
  Called when a port is instantiated somewhere on the hub
- `PortDeregistered(VPort port)`
  Called when a port is removed from the hub
- `OnNotification(string Rolename, string OpName, IList<VParamType> retVals, VPort senderPort)`
  Called when a notification is issued from a subscribed port

Override the Abstract Methods

- `PortDeregistered(VPort port)`
- `PortRegistered(VPort port)`
- `OnNotification(string Rolename, string OpName, IList<VParamType> retVals, VPort senderPort)`
- A driver can usually have these methods do nothing, because it usually doesn't care about other ports

Agenda

First Things

Make an App

Make a Driver

Make a Scout

RTFM

Overview

- Create a VS Project
- Set Names, Properties, and References
- Implement the Abstract Methods

Create a VS Project

- In Scouts section of the solution
- Add a C# Class Library
- Place in “Scouts” directory
- Override the Abstract Methods

Set Names, Properties, and References

- Source: Rename the class
- Source: Rename the Namespace
- Source: Add ‘using’ declarations
- Source: Don’t need property declaration
- Source: Inherit from IScout
- Source: Implement abstract Methods
- References: Add ‘Common’, ‘Views’, and ‘DeviceScout’
- References: Add ‘AddIn’, ‘ServiceModel’, and ‘ServiceModel.Web’
- Properties: Set the output path
- Properties: Set the Assembly Name
- Properties: Set the Default namespace

Set Names, Properties, and References

- Easiest way to implement the Scout is to copy the DummyScout and change the names as appropriate
- Only substantive differences will be in the `GetDevices()` method

Agenda

First Things

Make an App

Make a Driver

Make a Scout

RTFM

RTFM

- I hope this has been a decent orientation.
- You'll probably need to
  RTFM: Read The *Fine* Manuals at
  `https://labofthings.codeplex.com/downloads/get/764896`
  [PDF WARNING]
- Look here, too:
  `http://labofthings.codeplex.com/documentation`
- You can learn a lot from the example code you found here:
  `http://labofthings.codeplex.com/SourceControl/latest`
- These slides are at:
  `https://github.com/dslachut/LoT`