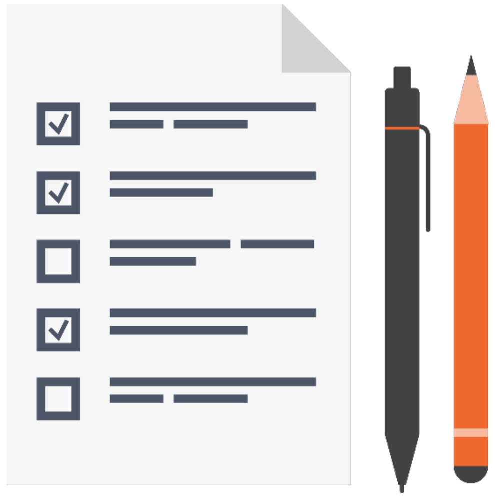
Building Reactive Concurrent WPF Applications with Akka.NET

Introduction

Jason Roberts

@robertsjason | dontcodetired.com

Overview



Prerequisites

Why reactive Akka.NET WPF applications

Overview of reactive systems and the Reactive Manifesto

ReactiveUI

Overview of the demo application

Get started in Visual Studio

Course Prerequisites

- Akka.NET
  - Actors and actor references
  - Messages
  - Supervision hierarchies
  - “Building Concurrent Applications with the Actor Model in Akka.NET” course
- Dependency Injection
  - General understanding of DI (e.g. via constructor parameters)
  - DI in Akka.NET
  - “Implementing Logging and Dependency Injection in Akka.NET” course

Why Reactive Akka.NET WPF Applications?

- Take advantage of the power of multiple cores
- Easier to reason about concurrent operations
- UI asynchronously tells the actor model to do something
- UI reacts to (messages) being sent by actors in the actor model
- Fault tolerance and self-healing built in to the Actor Model
- Potential to use remote actors

Reactive Systems and the Reactive Manifesto

"responds in a timely manner if at all possible"

Responsive

"stays responsive under varying workload"

Elastic

Resilient

"stays responsive in the face of failure"

Message Driven

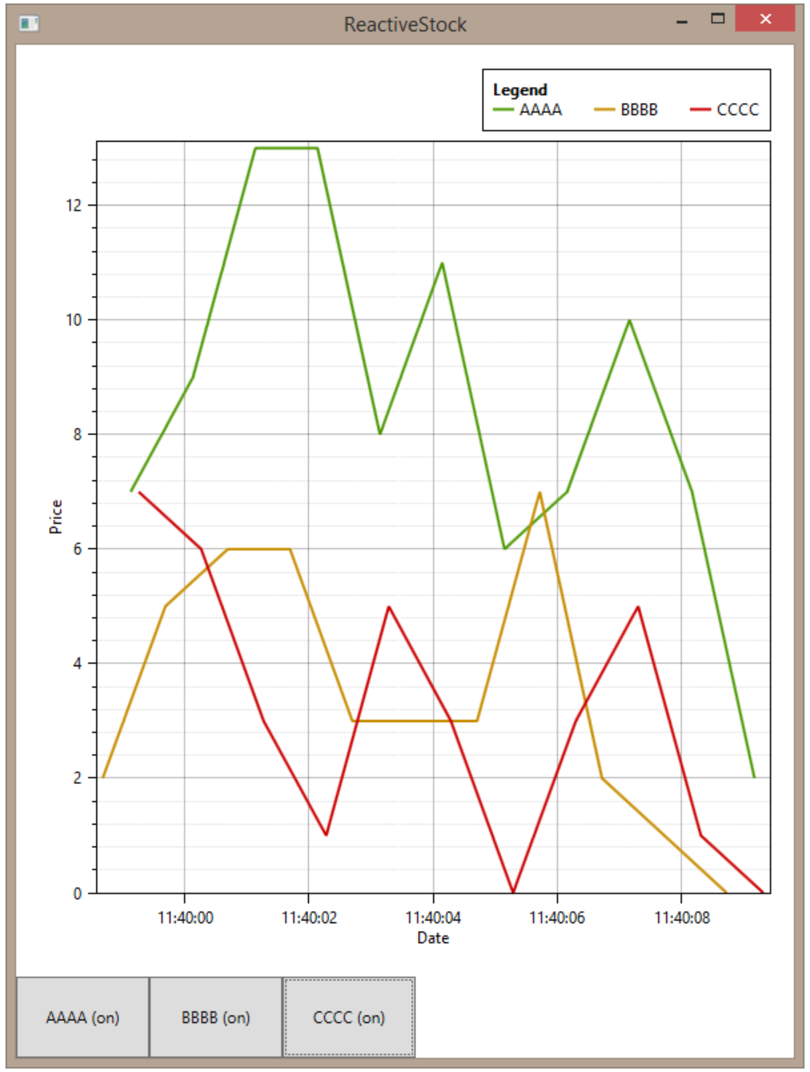
"asynchronous message-passing to establish a boundary between components that ensures loose coupling"

www.reactivemanifesto.org

ReactiveUI

- MVVM framework
- Built on Reactive Extensions (Rx) for .NET
- Easier multithreaded programming at the UI level
- Uses observable collections with asynchronous events
- Combine streams of events
- Not an Actor Model framework
  - Supervision hierarchies of actors
  - Location transparency / remote actors

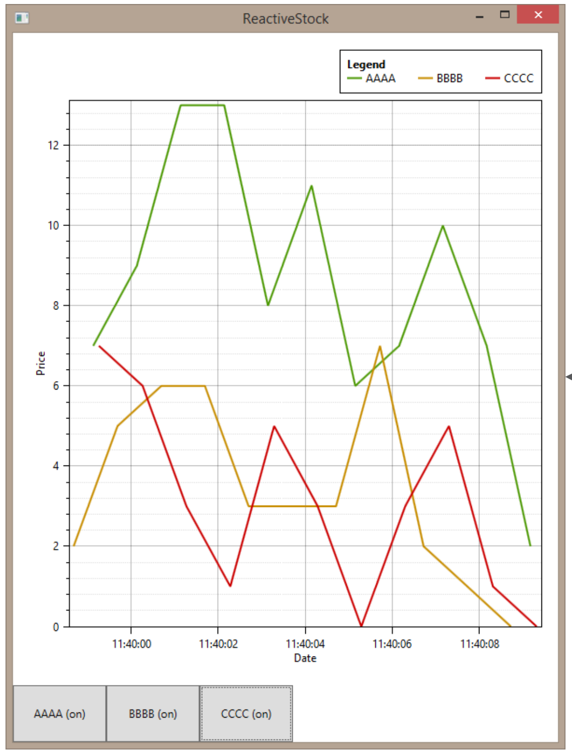
Demo Application Overview



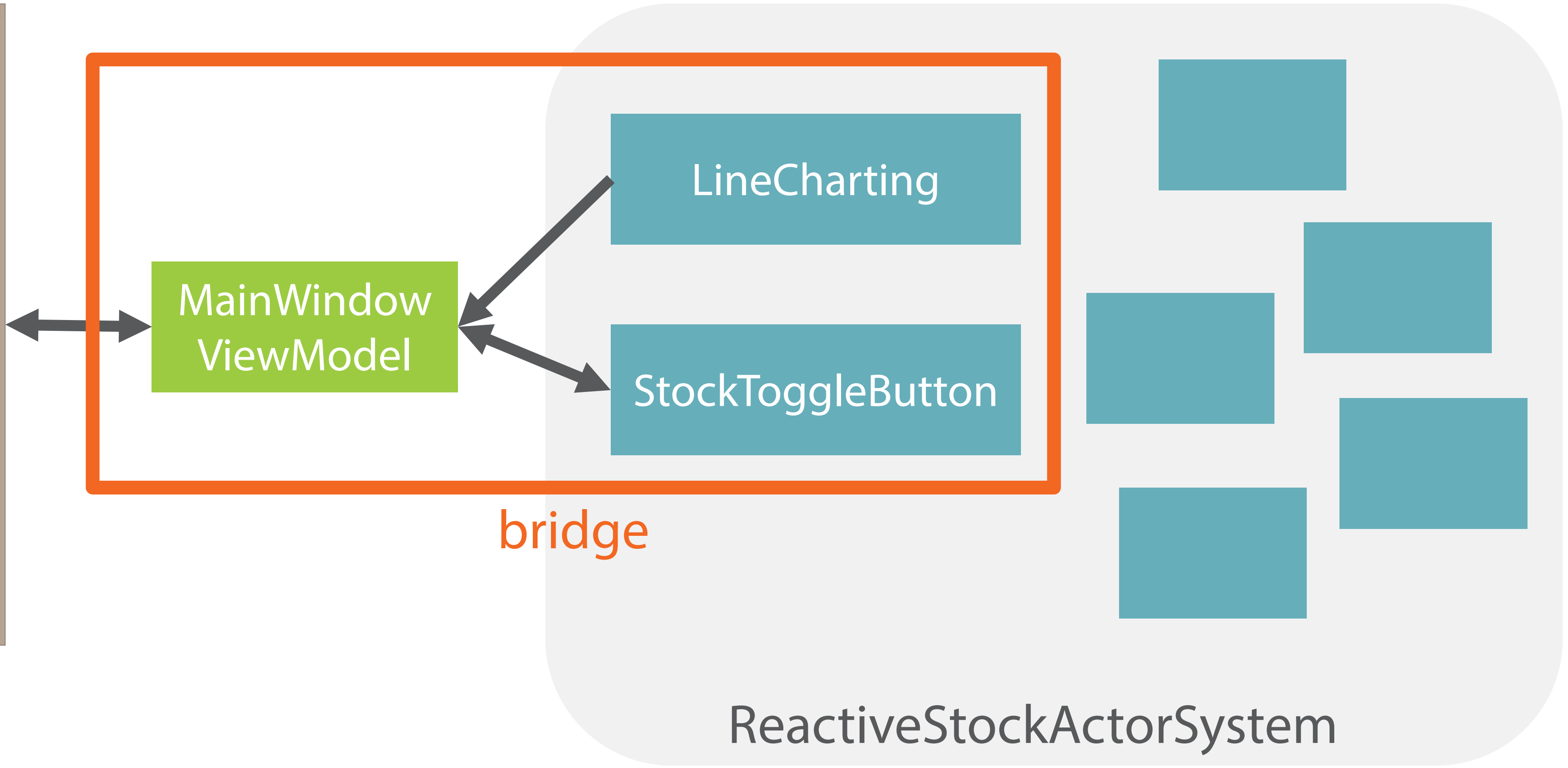
Tools and libraries:

- Akka.NET
- Ninject (DI)
- MVVM Light
- OxyPlot (line chart)

Demo Application Overview



MainWindow.xaml



Getting Started

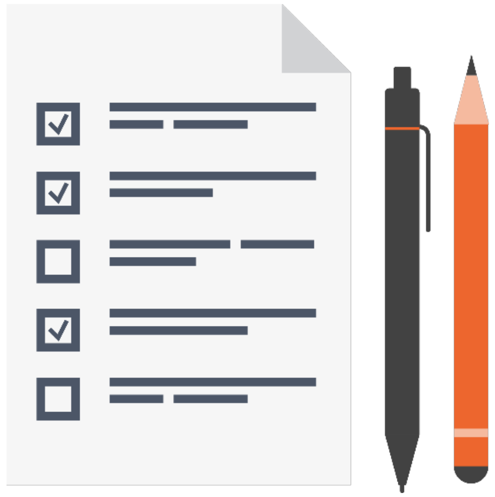
Start with new WPF project

Install required NuGet packages

Create actor system instance

Configure Ninject DI

Summary



Prerequisites

Why reactive Akka.NET WPF applications

Overview of reactive systems and the Reactive Manifesto

ReactiveUI

Overview of the demo application

Got started in Visual Studio

Next:

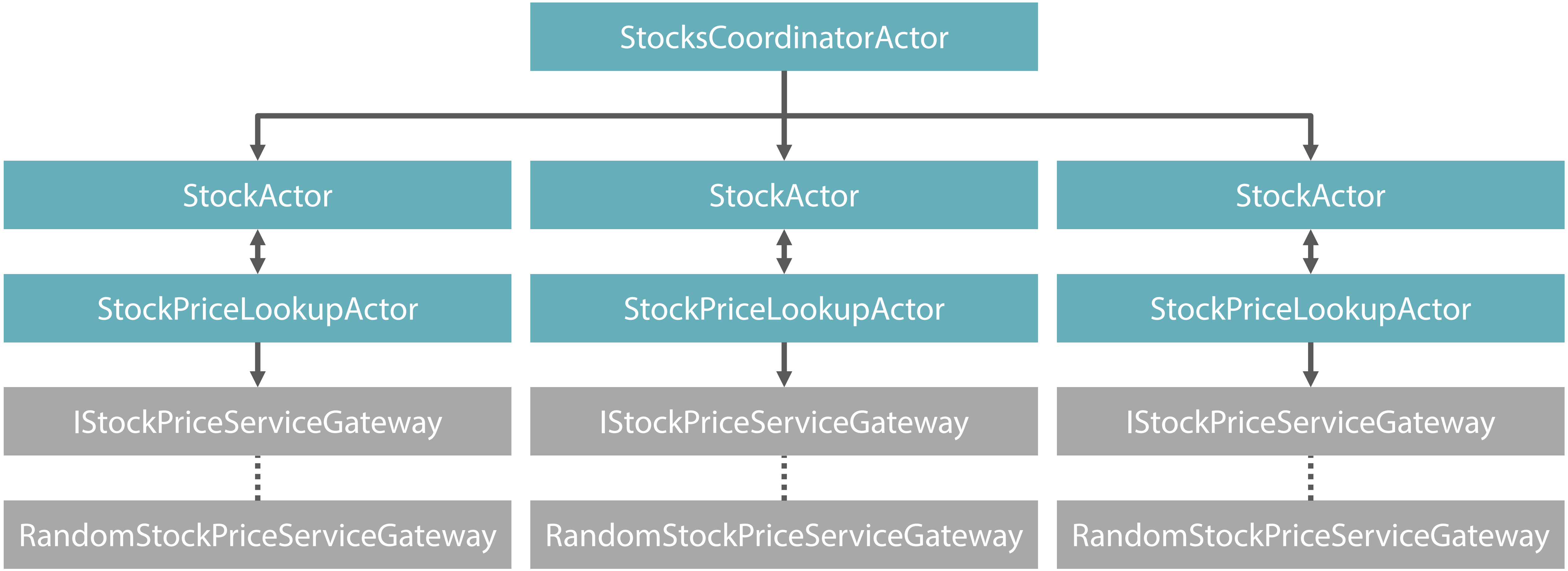
Building the Stock Price Watching Actors

Building the Stock Price Watching Actors

Jason Roberts

@robertsjason | dontcodetired.com

Overview


StocksCoordinatorActor
StockActor
StockActor
StockActor
StockPriceLookupActor
StockPriceLookupActor
StockPriceLookupActor
IStockPriceServiceGateway
IStockPriceServiceGateway
IStockPriceServiceGateway
RandomStockPriceServiceGateway
RandomStockPriceServiceGateway
RandomStockPriceServiceGateway

Creating the StockPriceServiceGateway

IStockPriceServiceGateway

RandomStockPriceServiceGateway

Configure Ninject

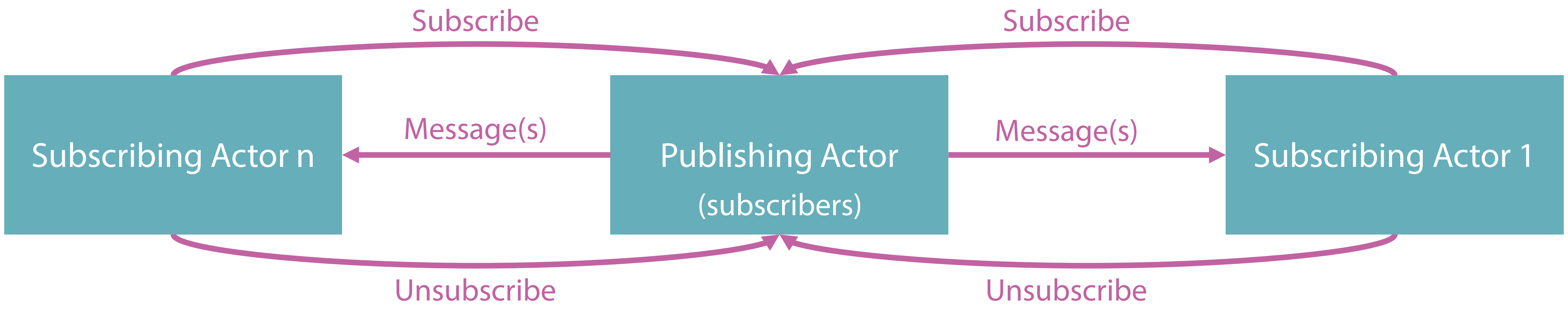
Writing the First Actor

StockPriceLookupActor

RefreshStockPriceMessage

UpdatedStockPriceMessage

The Publish-Subscribe Pattern Between Actors



Creating the Publishing StockActor

SubscribeToNewStockPrices

UnSubscribeFromNewStockPrices

StockActor

Getting New Prices in the StockActor

Child StockPriceLookupActor

Delegate RefreshStockPriceMessage

Create StockPriceMessage

Handle UpdatedStockPriceMessage

Send StockPriceMessage to subscribers

Scheduling StockActor Updates

Repeat RefreshStockPriceMessage send

Create ICancelable field

Create schedule in PreStart()

Cancel schedule in PostStop()

Creating the StocksCoordinatorActor

WatchStockMessage

UnWatchStockMessage

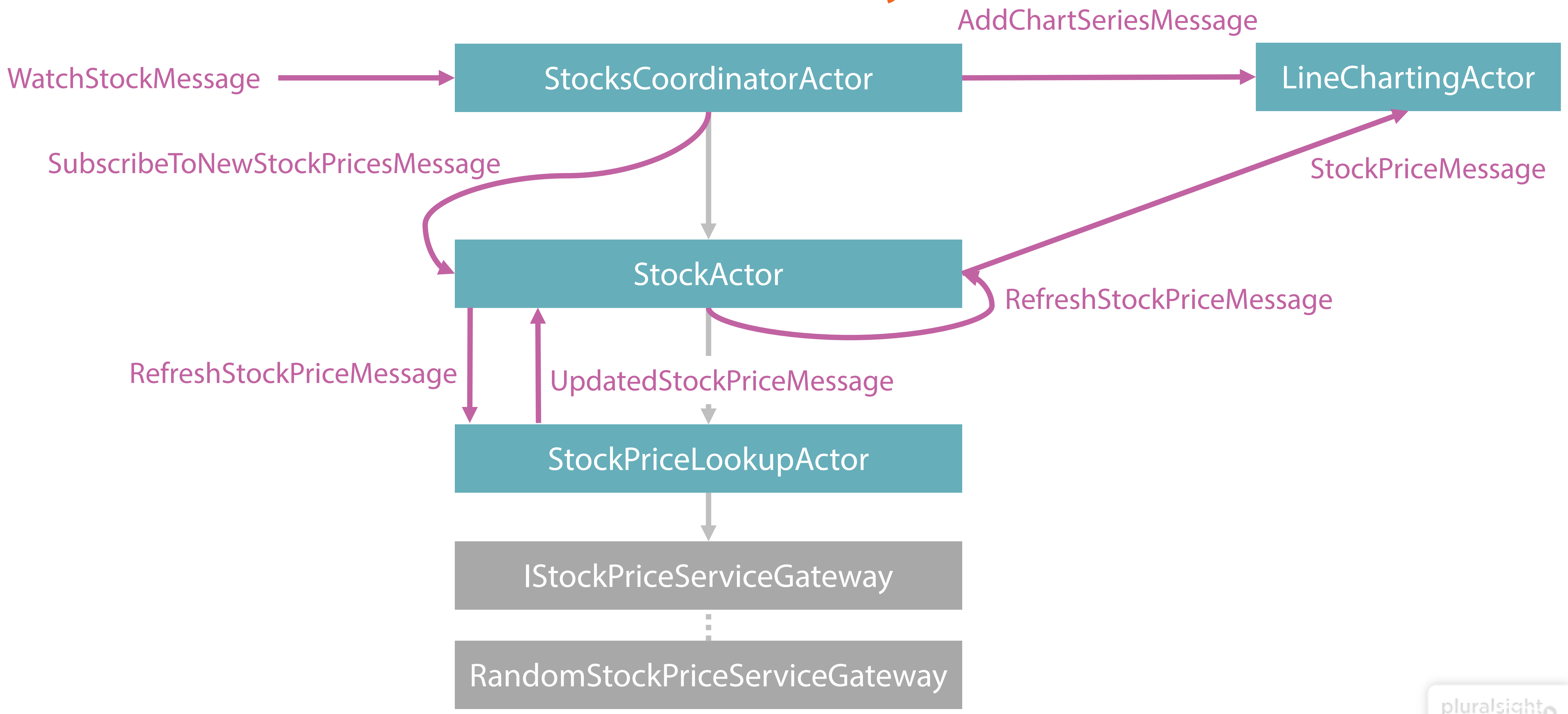
AddChartSeriesMessage

RemoveChartSeriesMessage

Create child StockActors

(un)subscribe the chart actor to new stock prices

Summary


AddChartSeriesMessage
WatchStockMessage
StocksCoordinatorActor
LineChartingActor
SubscribeToNewStockPricesMessage
StockPriceMessage
StockActor
RefreshStockPriceMessage
RefreshStockPriceMessage
UpdatedStockPriceMessage
StockPriceLookupActor
IStockPriceServiceGateway
RandomStockPriceServiceGateway

Next:

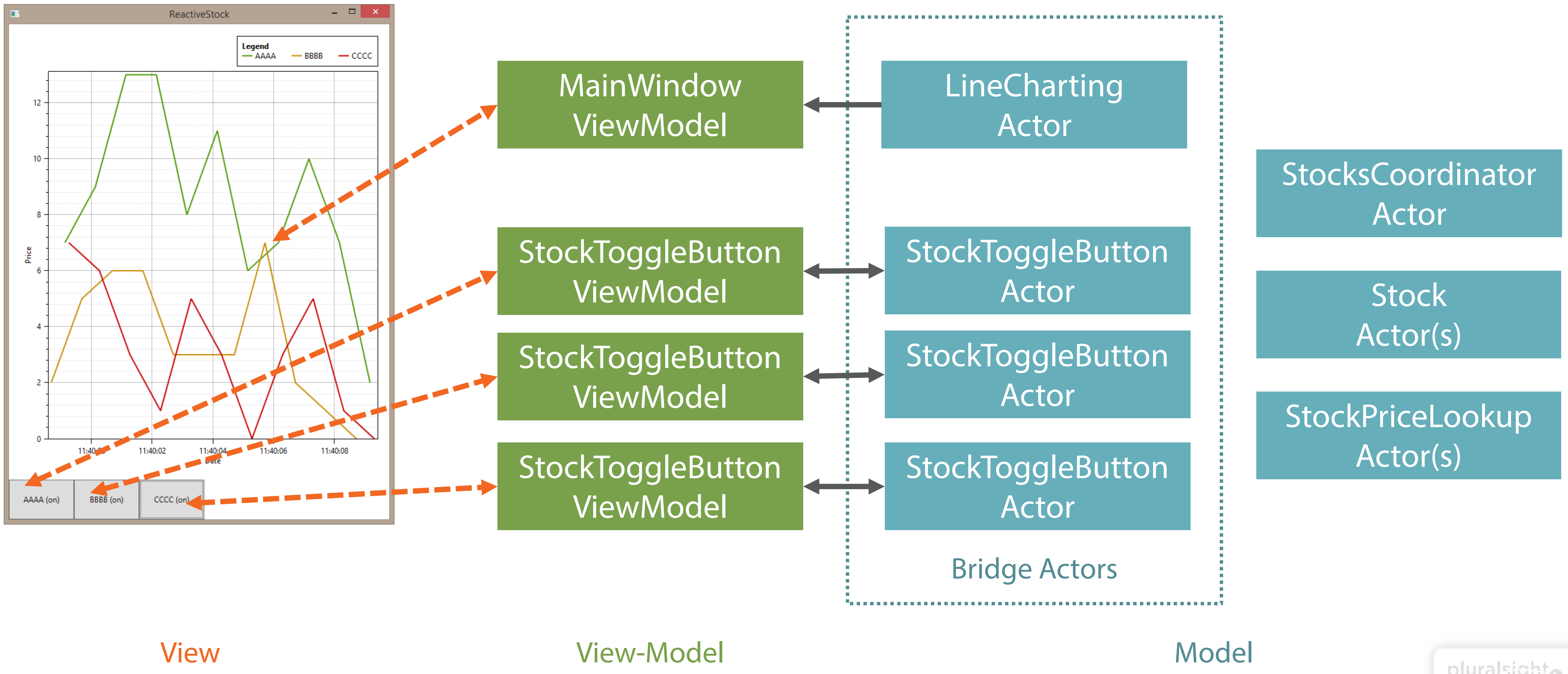
Creating the User Interface Actors

Creating the User Interface Actors

Jason Roberts

@robertsjason | dontcodetired.com

Overview


ReactiveStock
Legend
AAAA
BBBB
CCCC
Price
Date
11:40:02
11:40:04
11:40:06
11:40:08
AAAA (on)
BBBB (on)
CCCC (on)
MainWindow ViewModel
StockToggleButton ViewModel
StockToggleButton ViewModel
StockToggleButton ViewModel
LineCharting Actor
StockToggleButton Actor
StockToggleButton Actor
StockToggleButton Actor
Bridge Actors
StocksCoordinator Actor
Stock Actor(s)
StockPriceLookup Actor(s)
View
View-Model
Model

Getting Started with the MainWindowViewModel

Initial MainWindowViewModel

Configure ViewModelLocator

Set XAML DataContext

Creating the StockToggleButtonViewModel

Bind a Button’s Text (Content)

Bind a Button’s Command

StockToggleButtonActorRef

.Tell(new FlipToggleMessage())

Creating the StockToggleButtonActor Bridge

FlipToggleMessage

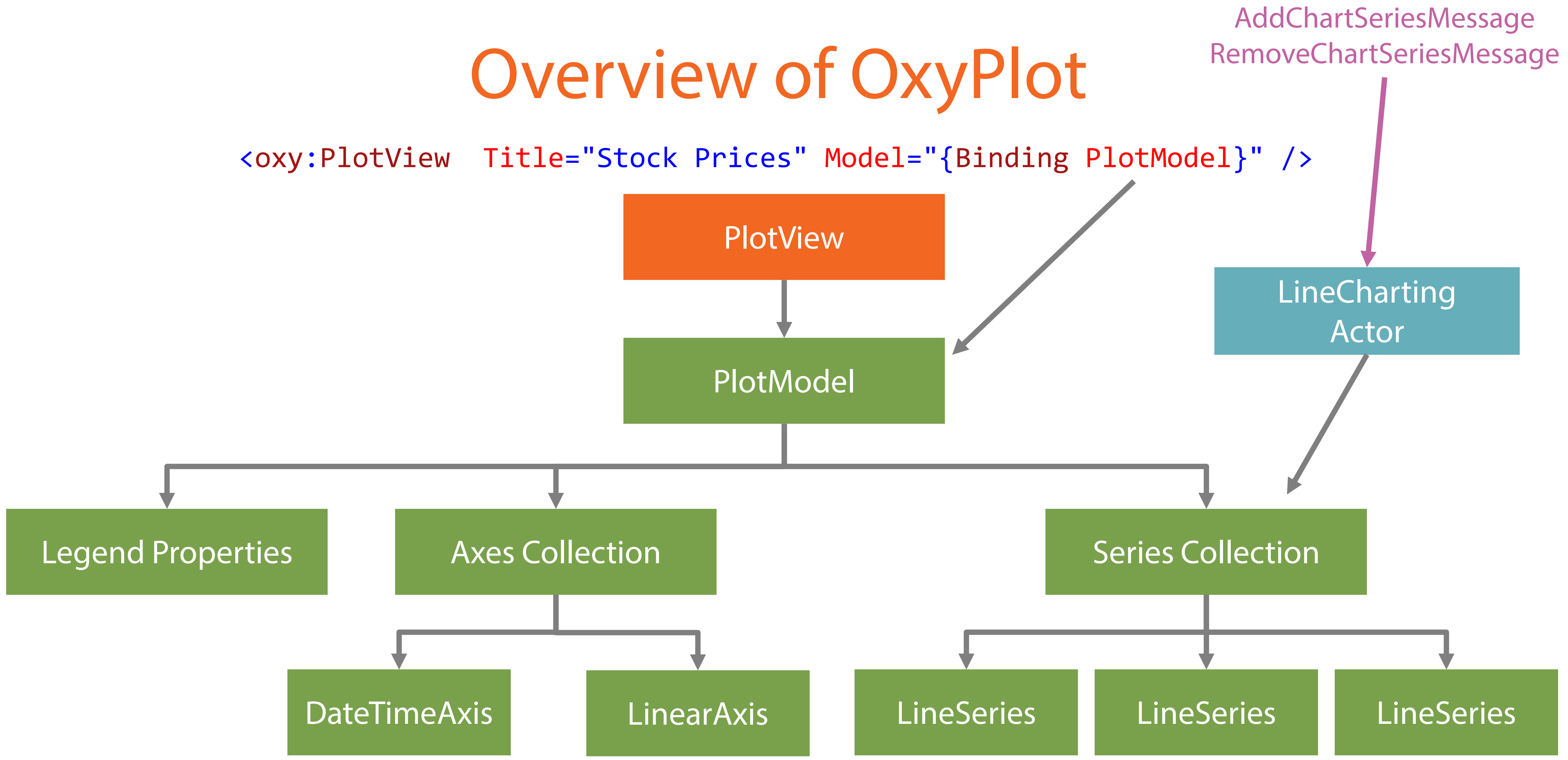
ToggledOff and ToggledOn states

_viewModel

_coordinatorActor

Overview of OxyPlot

```
<oxy:PlotView  Title="Stock Prices" Model="{Binding PlotModel}" />
```



Creating the LineChartingActor Bridge

PlotModel _chartModel

AddChartSeriesMessage

RemoveChartSeriesMessage

StockPriceMessage

Completing the MainWindowViewModel

StockButtonViewModels property

PlotModel property

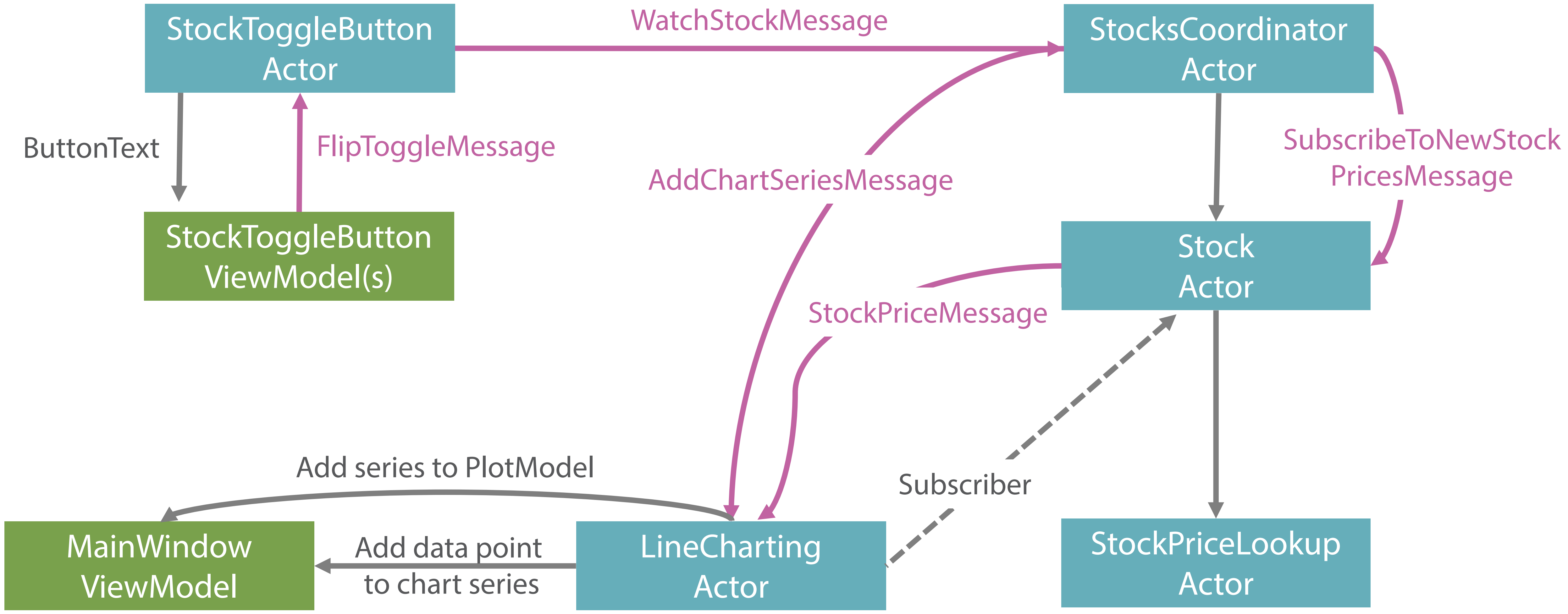
SetUpChartModel()

InitializeActors()

CreateStockButtonViewModels()

Creating the MainWindow XAML

Summary


StockToggleButton Actor
WatchStockMessage
StocksCoordinator Actor
ButtonText
FlipToggleMessage
AddChartSeriesMessage
SubscribeToNewStock PricesMessage
StockToggleButton ViewModel(s)
Stock Actor
StockPriceMessage
Add series to PlotModel
Subscriber
MainWindow ViewModel
Add data point to chart series
LineCharting Actor
StockPriceLookup Actor