Responding to User Interactions with Commanding

Brian Lagunas
INFRAGISTICS – MICROSOFT MVP

@brianlagunas | https://brianlagunas.com

Agenda

Understanding ICommand

Using the DelegateCommand

Raising Change Notifications

Using the CompositeCommand

ICommand

Binds a UI gesture to an action

ICommand

- Execute(object parameter)
- CanExecute(object parameter)

Enables/Disables element

DelegateCommand

Doesn’t require an event handler

Uses delegate methods

Defined within a ViewModel

Creating DelegateCommand

```
DelegateCommand SomeCommand = new DelegateCommand(Execute);
```

Creating DelegateCommand

```
DelegateCommand SomeCommand = new DelegateCommand(Execute);

private void Execute()

{

    //do something

}
```

Creating DelegateCommand

```
DelegateCommand SomeCommand = new DelegateCommand(Execute, CanExecute);

private void Execute()

{

    //do something

}
```

Creating DelegateCommand

```
DelegateCommand SomeCommand = new DelegateCommand(Execute, CanExecute);
private void Execute()
{
    //do something
}
private bool CanExecute()
{
    return true;
}
```

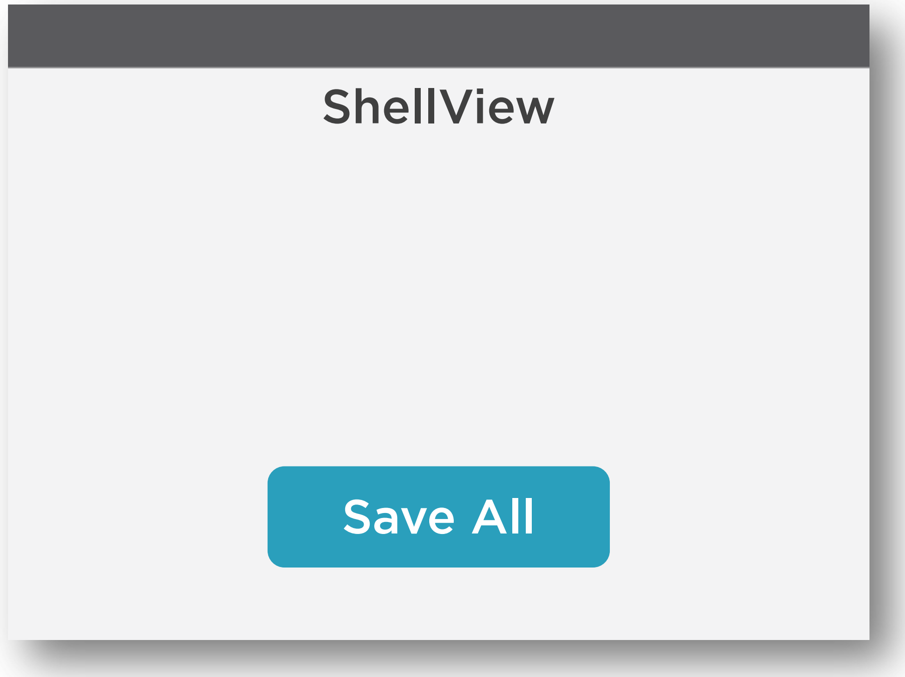
DelegateCommand<T>

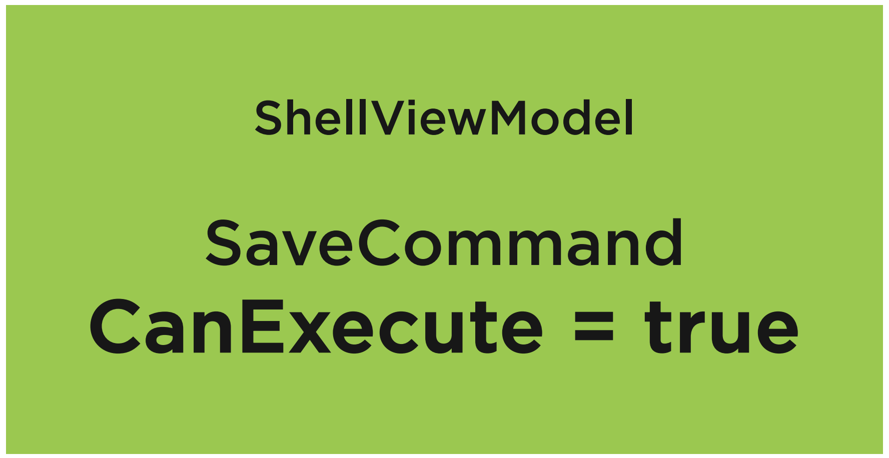
```
DelegateCommand SomeCommand = new DelegateCommand<string>(Execute, CanExecute);

private void Execute(string param)

{

     //do something

}

private bool CanExecute(string param)

{

    return true;

}
```

Demo

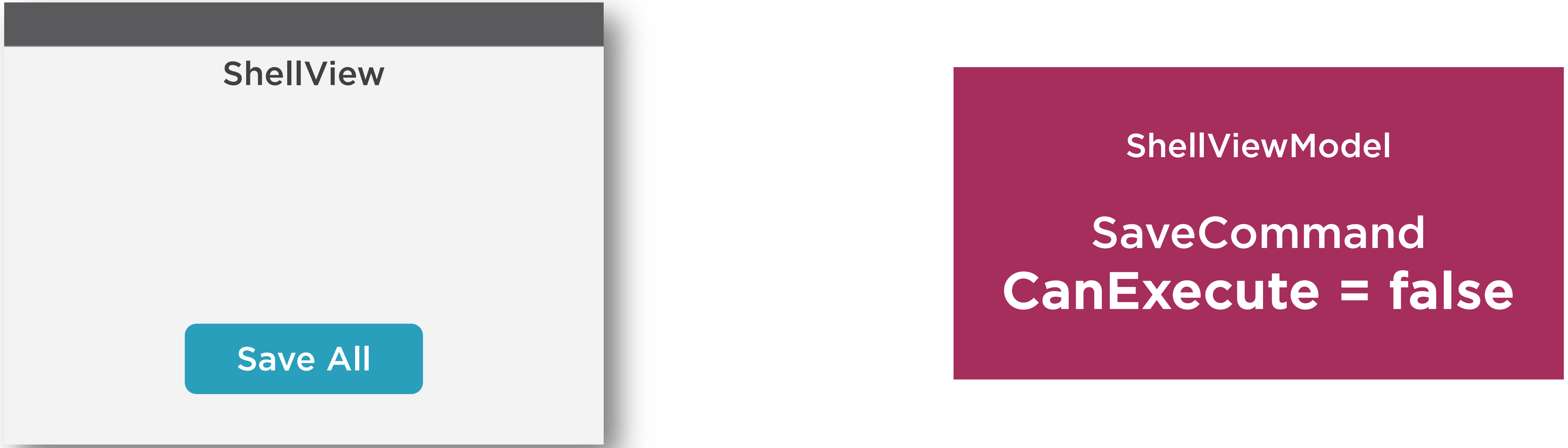
Using the DelegateCommand

Raising Change Notifications

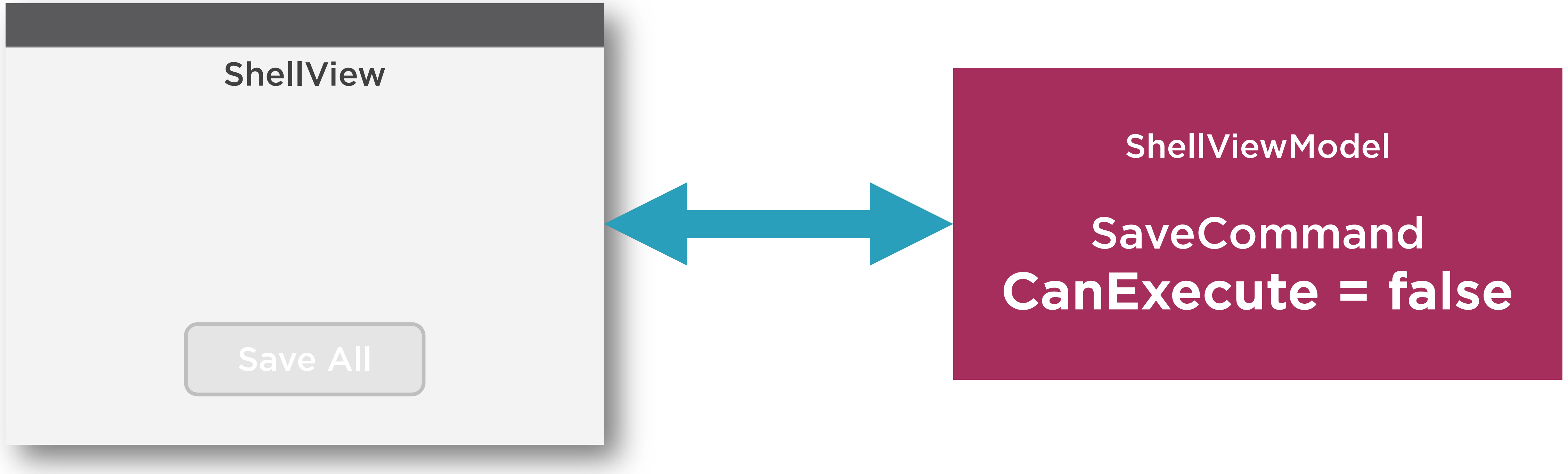




Raising Change Notifications



Raising Change Notifications



Raising Change Notifications

RaiseCanExecuteChanged

ObservesProperty

ObservesCanExecute

Manually RaiseCanExecuteChanged

```
DelegateCommand SomeCommand = new DelegateCommand(Execute, CanExecute);

SomeCommand.RaiseCanExecuteChanged();
```

ObservesProperty

```
DelegateCommand SomeCommand = new DelegateCommand(Execute, CanExecute)

    .ObservesProperty(() => MyProperty);
```

Chaining ObservesProperty

```
DelegateCommand SomeCommand = new DelegateCommand(Execute, CanExecute)
    .ObservesProperty(() => MyProperty)
    .ObservesProperty(() => MyOtherProperty);
```

ObservesCanExecute

```
DelegateCommand SomeCommand = new DelegateCommand(Execute)

        .ObservesCanExecute(() => CanEdit);
```

Demo

Raising change notifications

CompositeCommand

Acts as a Parent command

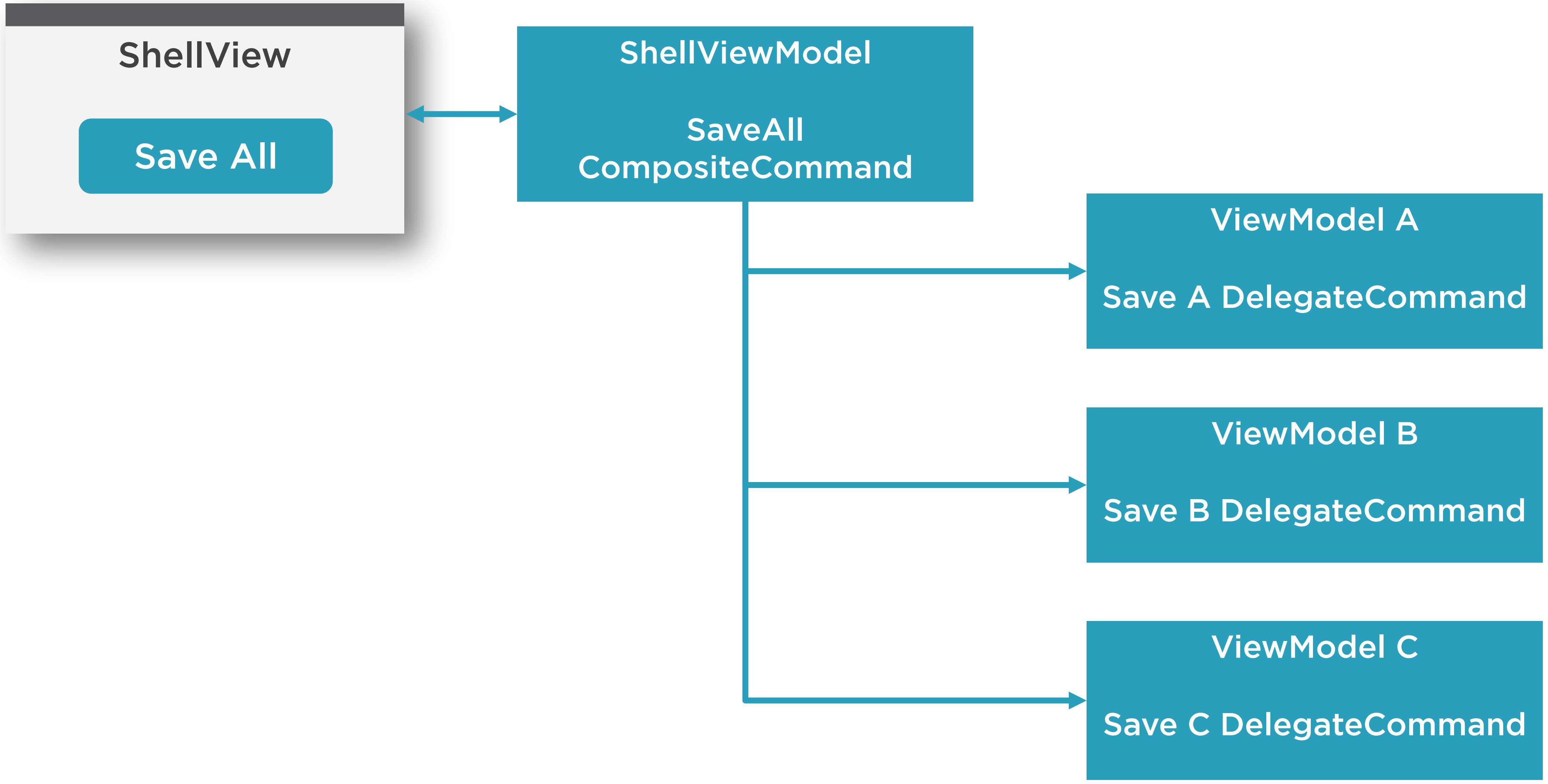
- Multiple child commands

Usually global

Local commands are registered

When invoked, all registered commands invoked

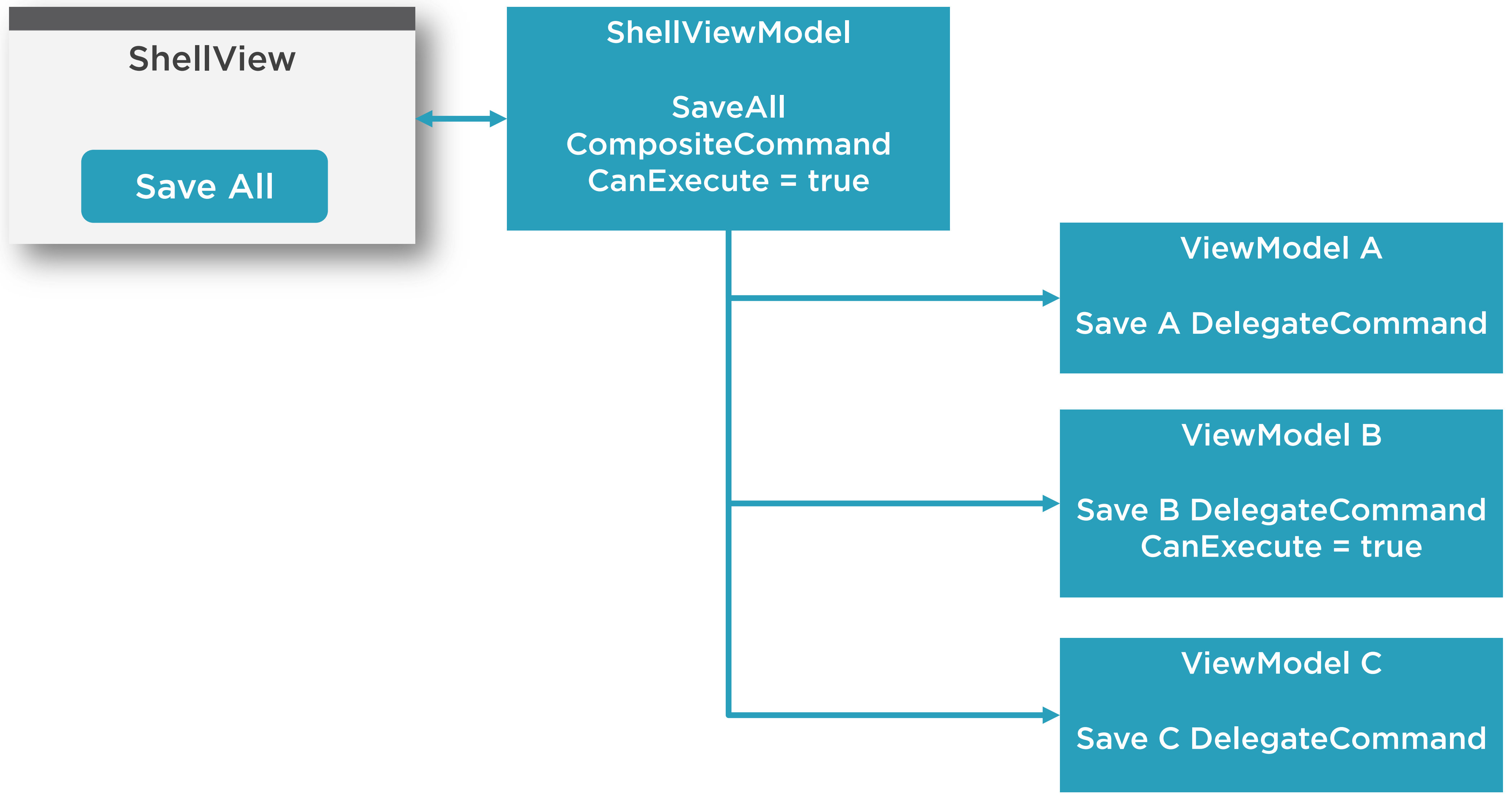
How a CompositeCommand Works



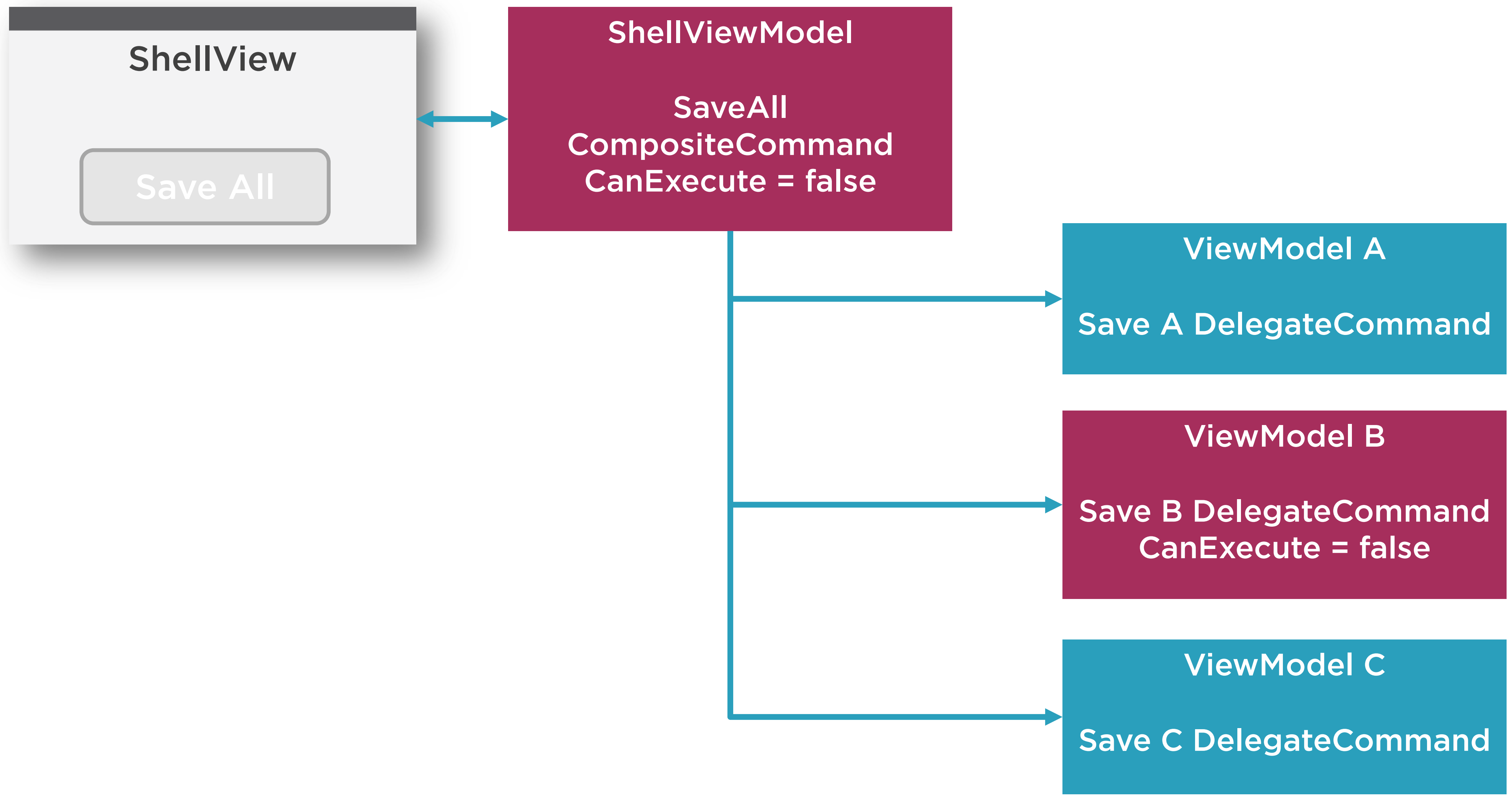
CompositeCommand

- Acts as a Parent command
  - Multiple child commands
- Usually global
- Local commands are registered
- When invoked, all registered commands invoked
- Supports enablement

CompositeCommand CanExecute



CompositeCommand CanExecute



Creating a CompositeCommand

```
CompositeCommand _saveCommand = new CompositeCommand();
public CompositeCommand SaveCommand
{
        get { return _saveCommand; }
}
```

Register/Unregister a CompositeCommand

```
//register
SaveCommand.RegisterCommand(delegateCommand);
```

Register/Unregister a CompositeCommand

```
//register
SaveCommand.RegisterCommand(delegateCommand);

//unregister
SaveCommand.UnregisterCommand(delegateCommand);
```

Demo

Using a CompositeCommand

Summary

Understanding ICommand

Using the DelegateCommand

Raising Change Notifications

Using the CompositeCommand