Encapsulating and Reusing your User Interface Code

By Quentin “Q” Alldredge
Q Software Innovations

About Q

Feel Free to call me "Q"

- I got the nickname for both the Star Trek and James Bond references
- Work of the Aerospace and Defense Industry
  - First at ATK (Now Northrup Grumman)
  - Now at Hill Air Force Base
- LabVIEW Consulting as Q Software Innovations

Contact Q

Phone/Text: +1 (435)-730-1198

Email: q@qsoftwareinnovations.com

Website: www.qsoftwareinnovations.com

LinkedIn: www.linkedin.com/in/quentin-q-alldredge

LabVIEW**Wiki.org**: Q

Twitter: @QSI_Q

NI Community Forums: TheQ

LAVA Forums: The Q

stack**overflow**: TheQ

Overview

- Quick Recap About QControls
- Reuse
- Encapsulation
- Extensibility

QControl Recap

What is a QControl Again?

What is a **QControl?**

QControls are an object-oriented, extensible alternative to XControls.

- A QControl is LabVIEW Object-Oriented Class which:
  - Contains a Control Reference as part of its Private Data
  - Uses Properties and Methods of its Class for all manipulation of the Control
  - Can be reused to recreate the UI Logic wherever required
  - Could have an asynchronously called Event Hander that handles UI Logic
  - Is part of the QControl Class Hierarchy which mimics the VI Server Class Hierarchy

What is a **QControl?**

Why not use an XControl?

Property	QControls	XControls
Encapsulates UI Logic, Separates from Business Logic	X	X
Custom Properties and Methods	X	X
Inherits Properties and Methods from VI Server	X	
Separates UI logic from the Skin	X	
Extensible	X	
Use with Libraries	X	
Use with Packed Project Libraries (PPLs)	X	
Use with Object-Oriented Programming	X	
Use with Actor Framework/DQMH	X	

What is a **QControl?**

Why not use an XControl?

XControls have two fundamental problems caused because they are always running:

1. An XControl begins execution when it is loaded.
   - Init Ability fires on drop, load of VI, or on load of Library, Class, PPL, etc.
   - Uninit Ability fires on delete, VI leaving memory, or on close of Library, Class, PPL, etc.

2. An XControl is difficult to handle during edit time.
   - All edit time behavior has to be programmed
   - Properties and methods to the controls on the façade have to be recreated

Quick Recap About **QControls**

What is a QControl Again?

History

- First created and released to the NI Tools Network in 2015
- Presented at NIWeek 2018 (bit.ly/QControlsNIWeek2018)
- Won the NI Tools Network Community Contribution Award at NIWeek 2019

To learn more about QControls

- Download and Follow Tutorials: bit.ly/QControlsLVTool
- Become a member of the QControl Enthusiasts Group: bit.ly/QControlsTool
- Follow me on Twitter and/or LinkedIn

Quick Recap About **QControls**

Parts of QControl

A QControl extends rather than replaces controls

- Properties and Methods
  - Through inheritance all properties and methods are available
  - Properties are accessed via Property Nodes on the class wire
  - Methods are accessed via Quick Drop and/or MGI's Class Method Browser
- Event Handler
  - Handles all UI event code through dynamic events
  - Automatically launched at run-time by the QControl Toolkit
- State Data
  - Keeps data in sync between Properties and Methods running in the Main Application and the asynchronously running Event Handler

Reuse

Reuse

Why?

Code reuse aims to save time and resources and reduce redundancy by taking advantage of assets that have already been created in some form within the software product development process.

Reuse

How?

10 Tips on Writing Reusable Code

1. Keep the code DRY. Dry means "Don't Repeat Yourself".
2. Make a class/method do just one thing.
3. Write unit tests for your classes AND make it easy to test classes.
4. Remove the business logic or main code away from any framework code.
5. Try to think more abstractly and use Interfaces and Abstract classes.
6. Code for extension. Write code that can easily be extended in the future.
7. Don't write code that isn't needed.
8. Try to reduce coupling.
9. Be more Modular
10. Write code like your code is an External API

From http://hoskinator.blogspot.com/2006/06/10-tips-on-writing-reusable-code.html

Encapsulation

Separate UI Code from Business Logic

Encapsulation

What?

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of bundling data and methods that work on that data within one unit, e.g., a class. This concept is also often used to hide the internal representation, or state, of an **object** from the outside.

Encapsulation

Why?

Encapsulation helps us in binding the data and the member functions (that work on the data) of a class. Encapsulation is also useful in hiding the data of a class from an illegal direct access.

Encapsulation

QControls encapsulates the UI code as self-contained, modular bundle, away from the business logic of the application and provides the application a clear API for use and reuse of the UI code.

Extension

Extensibility

What?

Extensibility is a principle that provides for future growth. Being extensible is a measure of the ability to extend a system and the level of effort required to implement the extension.

Extensions can be through the addition of new functionality or through modification of existing functionality. The principle is to be able to provide for enhancements without impairing existing system functions.

Reusability together with extensibility allows a code to be transferred to another project with less development and maintenance time, as well as enhanced reliability and consistency.

Extensibility

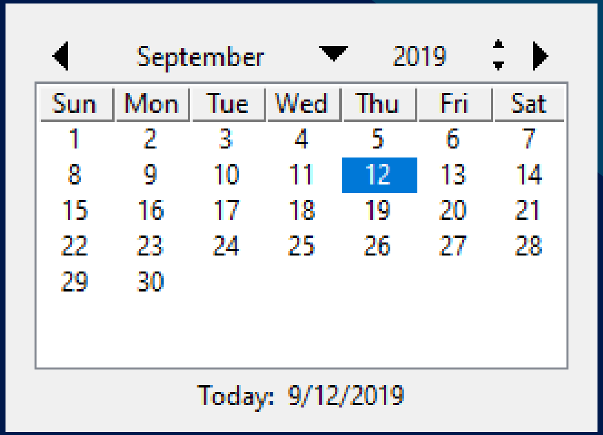
How are QControls extensible?

- Properties
- Overridable Properties
- Methods
- Overridable Methods
- Using a Class as a Property
- IDE Extension

Extensibility

Properties

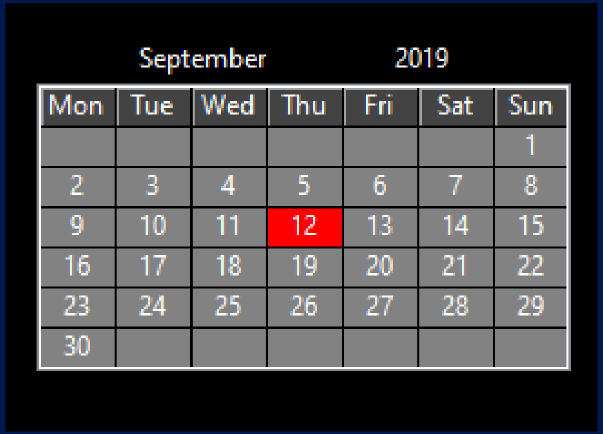
- **Properties** in QControls are just fancy Accessors
  - Can only have one input OR one output
  - Can have more code than bundle/unbundle of Class data
- Use Properties to add **custom** formatting



September 2019

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Today: 9/12/2019



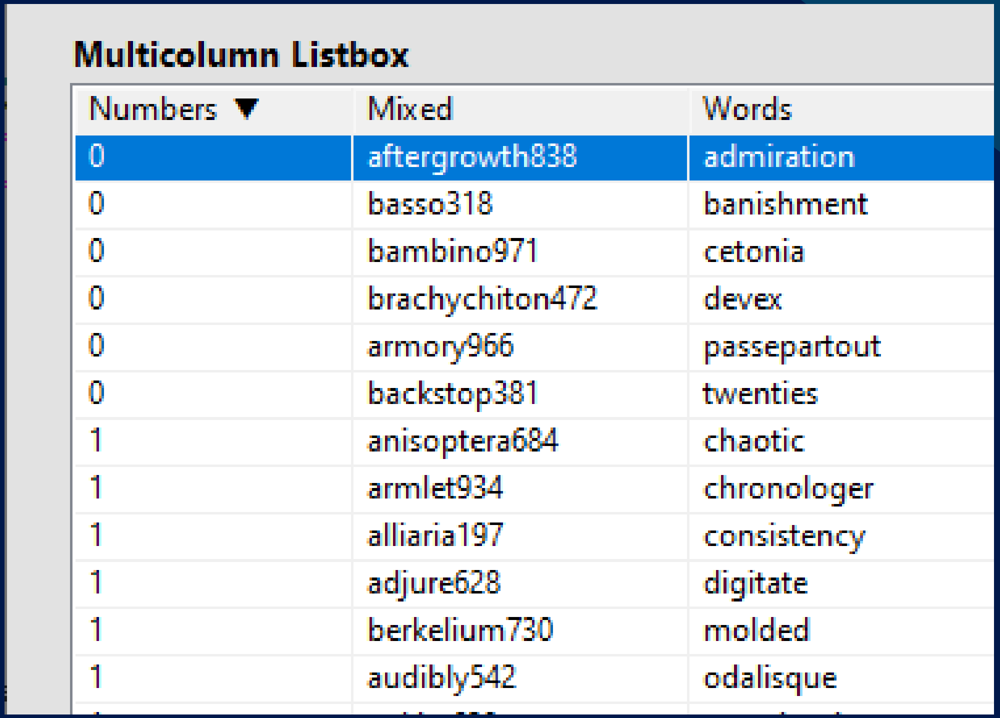
September 2019

Mon	Tue	Wed	Thu	Fri	Sat	Sun
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

Extensibility

Overridable Properties

- Make the Properties Dynamic Dispatch
  - Behavior of the Property can be extended
  - Make sure it still acts as the developer would expect
- All built-in Properties are **Overridable**



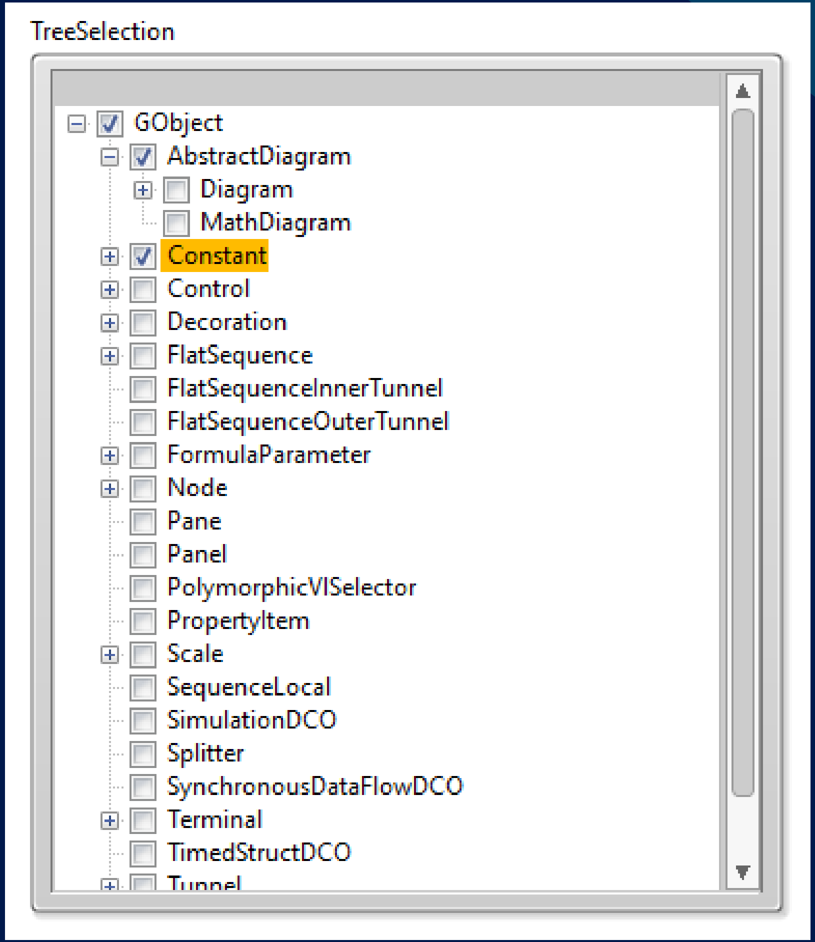
Multicolumn Listbox

Numbers ▼	Mixed	Words
0	aftergrowth838	admiration
0	basso318	banishment
0	bambino971	cetonia
0	brachychiton472	devex
0	armory966	passepartout
0	backstop381	twenties
1	anisoptera684	chaotic
1	armlet934	chronologer
1	alliaria197	consistency
1	adjure628	digitate
1	berkelium730	molded
1	audibly542	odalisque

Extensibility

Methods

- **Methods** in QControls are regular VIs
    - Can be public, protected, private, etc.
    - Use to establish API for inheritance
- Use Methods to **extend** functionality
    - Add checkboxes by methods controlling symbols



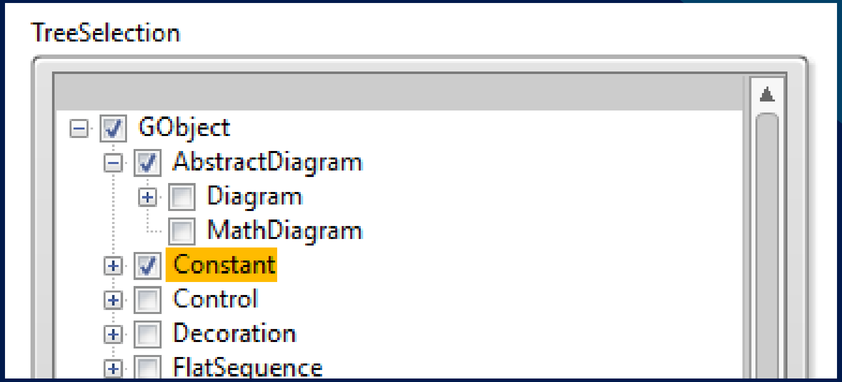
TreeSelection

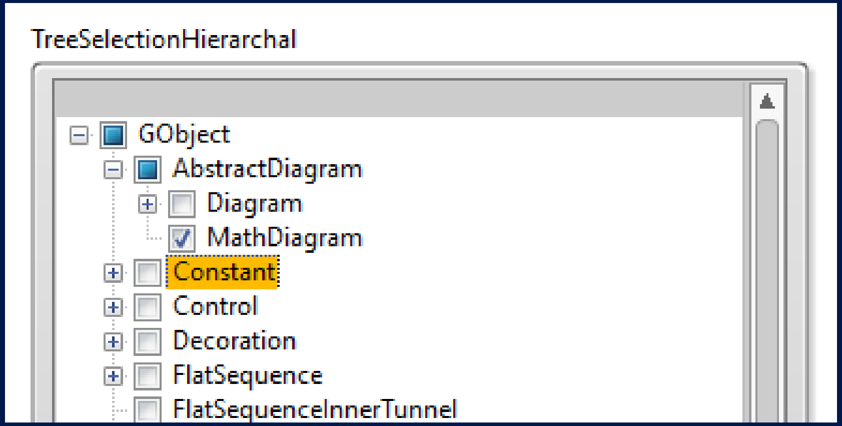
- [✓] GObject
    - [✓] AbstractDiagram
        - [ ] Diagram
        - [ ] MathDiagram
    - [✓] Constant
    - [ ] Control
    - [ ] Decoration
    - [ ] FlatSequence
    - [ ] FlatSequenceInnerTunnel
    - [ ] FlatSequenceOuterTunnel
    - [ ] FormulaParameter
    - [ ] Node
    - [ ] Pane
    - [ ] Panel
    - [ ] PolymorphicVISelector
    - [ ] PropertyItem
    - [ ] Scale
    - [ ] SequenceLocal
    - [ ] SimulationDCO
    - [ ] Splitter
    - [ ] SynchronousDataFlowDCO
    - [ ] Terminal
    - [ ] TimedStructDCO
    - [ ] Tunnel

Extensibility

Overridable Methods

- **Override Methods** in QControls to
  - Can only have one input OR one output
  - Can have more code than bundle/unbundle of Class data
- Use Methods to **extend** functionality
- All built-in Methods are **Overridable**

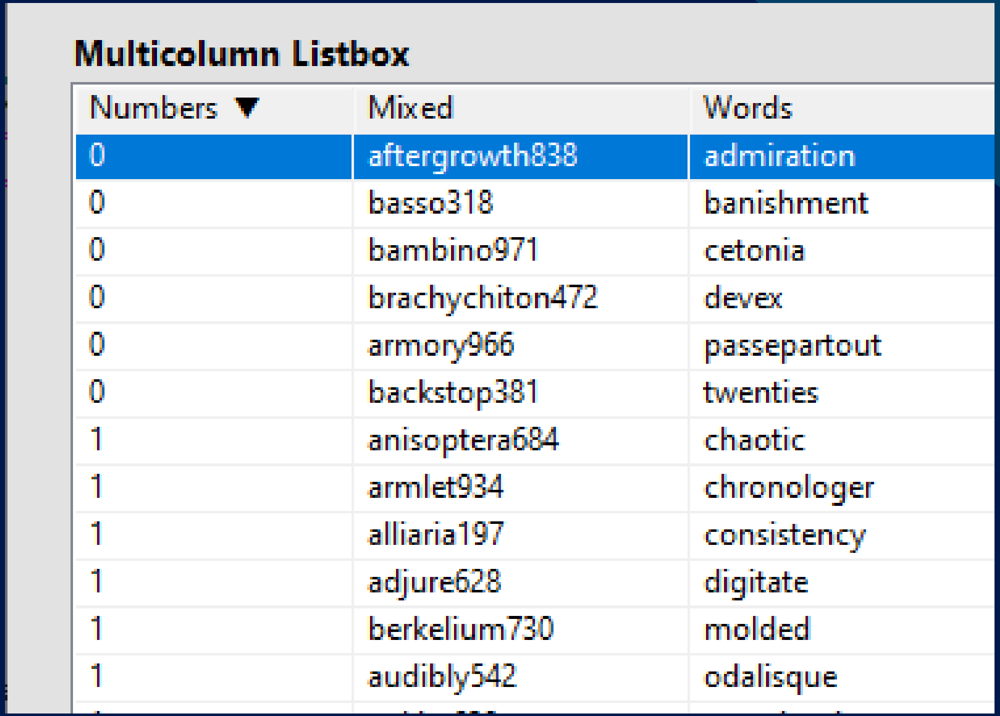




Extensibility

Using Class as a Property

- **Objects can be passed to a QControl as a Property**
  - Define the Property with the Parent Class
  - Use the Parent Class to define the API for others to extend



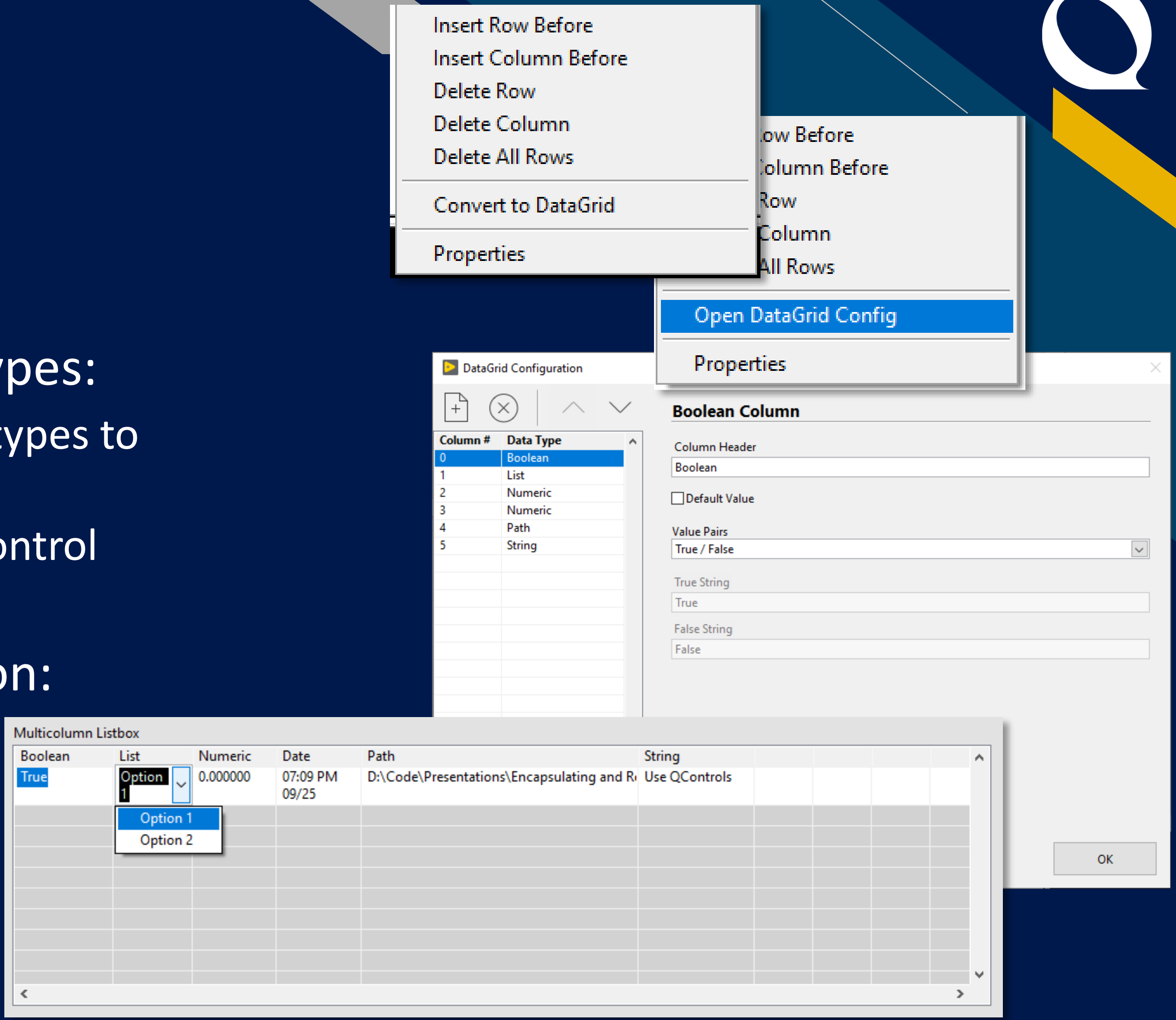
Multicolumn Listbox

Numbers ▼	Mixed	Words
0	aftergrowth838	admiration
0	basso318	banishment
0	bambino971	cetonia
0	brachychiton472	devex
0	armory966	passepartout
0	backstop381	twenties
1	anisoptera684	chaotic
1	armlet934	chronologer
1	alliaria197	consistency
1	adjure628	digitate
1	berkelium730	molded
1	audibly542	odalisque

Extensibility

IDE Extension for UI Extension

- **Class inheritance** for different data types:
  - Plugin architecture allows for new data types to be added
  - API defines how data type reacts with control and configuration dialog
- **Right-Click Extension** for Configuration:
  - Allows developer to set configuration at develop time
  - Configuration saves with the VI



Column #	Data Type
0	Boolean
1	List
2	Numeric
3	Numeric
4	Path
5	String

Boolean Column

Column Header: Boolean

☐ Default Value

Value Pairs: True / False

True String: True

False String: False

Multicolumn Listbox

Boolean	List	Numeric	Date	Path	String
True	Option 1	0.000000	07:09 PM 09/25	D:\Code\Presentations\Encapsulating and R	Use QControls

Summary

Almost Done

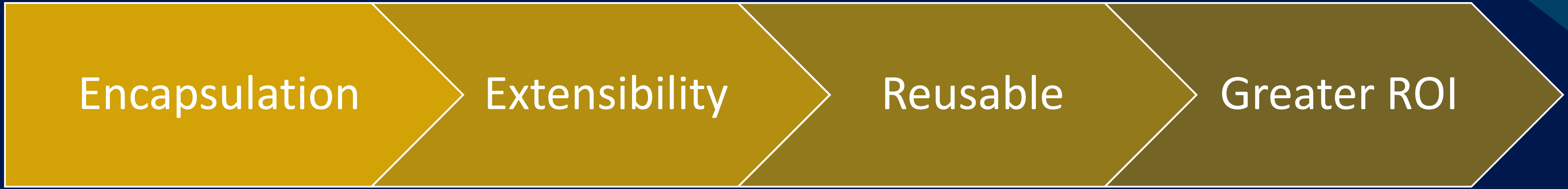
Summary

- Reuse saves time and resources
- Encapsulation facilitates reuse by make code modular
- Extensibility facilitates reuse by allowing functionality to be extendable

- **QControls are the best* way to reuse UI code and are extensible by:**
  - Properties – add formatting
  - Property Overrides – customize formatting
  - Methods – extend functionality
  - Method Overrides – improve/change functionality
  - Class Object as Property – allow for future added capability
  - IDE Extension – improve developer interaction

* There might be some bias to this statement.

Summary



Summary of Links

Where do I go again?

QControl Demo Code

https://gpackage.io/ GPM

• **Calendar**	@qsi/calendar-qcontrol
• **ColumnSortMulticolumnListbox**	@qsi/columnsortmulticolumnlistbox-qcontrol
• **DataGrid**	**Coming Soon to GPM** Source at: https://gitlab.com/QSI_Shared_Code/SharedQControls/DataGrid.git
• **TreeControl Selection**	Distributes with the QControl Toolkit from the NI Tools Network

Summary of Links

Where do I go again?

QControl Links

• **bit.ly/QControlsTool**	QControl Enthusiasts LabVIEW Community Group
• **bit.ly/QControlsNIWeek2018**	NIWeek 2018 Presentation on QControls (CoE Site)
• **bit.ly/QControlsNIToolsNet**	NI Tools Network QControl Toolkit
• **bit.ly/QControlsIdea**	Add QControls to LabVIEW Core on Idea Exchange
• **bit.ly/QControlsLVWiki**	QControls on the LabVIEW Wiki

Summary of Links

Where do I go again?

G Community Links

• **www.gcentral.org**	**Independent** Source to find G Libraries
• **www.lavag.org**	**Independent** Source for G Discussion
• **www.labviewwiki.org**	**Independent** G Knowledge Base
• **www.gpackage.io**	**Independent** G Packager/Repository
• **www.gdevcon.com**	**Independent** Graphical Programming Conference