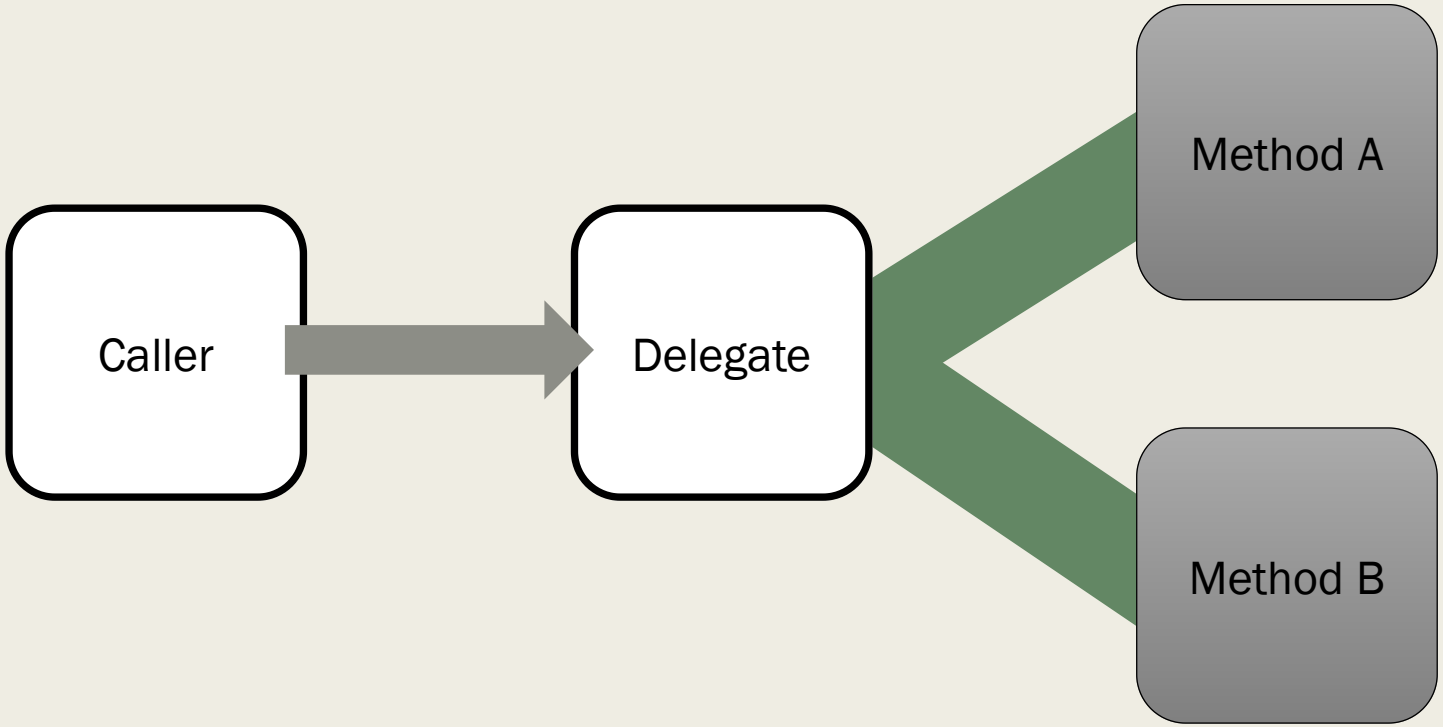
.NET BACK TO BASICS

Using Delegates, Events, Anonymous Methods and Lambdas for Application Extensibility

By Kevin Hazzard

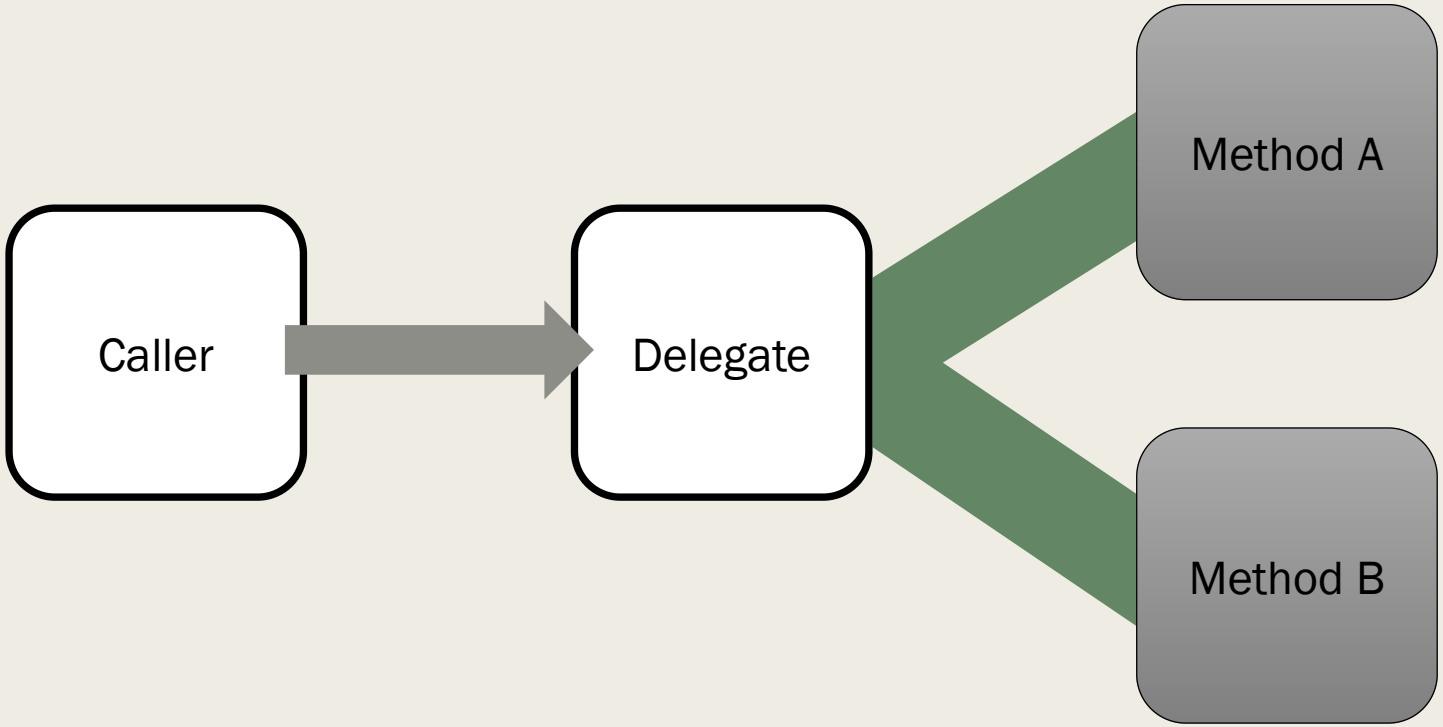
Why is a delegate called a delegate?



Example 1 – Runtime Method Selection Using a v1 Delegate

- Simple binding of a delegate to a method for dynamic invocation

The Multicast Delegate



Multicast Delegate Notes

- The target methods are called in the order they are added.
- Unhandled exceptions within any invoked method breaks the multicasting and returns the exception to the caller.
- You may use +, +=. – and -= to add and remove delegates from the list.
- When subtracting a delegate, if multiples of the target exist, only the last one added will be removed.
- Base type System.Delegate has some interesting methods:
  - *GetInvocationList*
  - *GetMethodInfo (extension)*

Example 2 –Using a v1 Delegate to Queue Operations

- Simple binding ordered delegates for dynamic invocation of a series of operations.

Example 3 –Using a v1 Delegate to Queue and Remove Operations

- Simple binding ordered delegates for dynamic invocation of a series of operations.
- Removal of targets

Anonymous Methods in .NET v2

- Version 2 of .NET (2005) introduced anonymous methods.
- These are really delegates with no names from our perspective.
- Since these methods are instantiated on demand and have an internal naming mechanism, there’s no way to subtract them from a multicast delegate except by reference to a specific instance.

Example 4 –Runtime Method Selection Using a v2 Anonymous Method

- Simple binding of an anonymous method for dynamic invocation

Events for Pub / Sub Messaging

- Built into the CLR and the Framework Class Library
  - *EventHandler*
  - *EventArgs*
- Windows Forms and Web applications use this pattern extensively.
- The event implementer is called the publisher.
- The event consumer is called the subscriber.
- Very much like multicast delegates but constrained by arguments and method signatures in the FCL.
- You can extend to use your own delegate types, too!

Example 5 – Publishing and Subscribing in Windows Forms

- Subscribe to a button press at runtime.
- Inspect the Click event metadata.
- Try dynamically subtracting and adding delegates to the event.

Lambda Functions

- Introduced in .NET 3 to support Language Integrated Query
- Anonymous methods on performance-enhancing drugs.
- Helps us to begin thinking of code like data.

Example 6 – Runtime Method Selection Using a v3 Lambda Function

- Simple binding of lambda expressions for dynamic invocation

Example 7 – Runtime Method Selection Using v3 Lambda Functions as Objects

- Simple binding of lambda expressions from pre-defined objects for dynamic invocation
- Look at statement lambdas (as opposed to expression lambdas).

Generic Delegates
Func<T, TResult> and Action<T>

- You can create generic delegates that are based on specific parameters.
- If no return is required, Action<T> will do.
- If a return is requires, use Func<T, TResult>
  - *Replace T with any number of types (up to eight) required for the parameters*
  - *The last type parameter is for the result*

Example 8 – Use Action<T> Generic Delegate

- Simple binding of lambda expressions from pre-defined objects based on Action<T> for dynamic invocation

Example 9 – Use Action<T> Generic Delegate from Compiled Lambda

- Simple binding of lambda expressions from pre-defined objects based on Action<T> created from a compiled lambda expression for dynamic invocation

Expression Trees

- Going all the way: CODE IS DATA.

Example 10 – Building an Expression Tree on the Fly to Compile and Execute

- Let’s examine a common mathematical function as a function delegate built from scratch for runtime compilation and execution.