“LINQ In Breadth” Bart De Smet

May 22, 2009
[MS Techdays 2009] Summary “LINQ In Breadth” Bart De Smet

LINQ

Language Integrated Query or LINQ keeps on growing and is providing support for a lot of sources and allowing integration with several languages (see LINQ fan-in and fan-out below).

LINQ brings together some parts introduced in the earlier versions of the .NET Framework like Lamba Expressions, Type Inference and Extension Methods and can be expressed in several syntaxes.

I started off trying to summarize for you everything Bart said but his presentation contained some many things going from Lambda expressions to the mathematical theory of Monads (Category Theory used to explain similarities between query operators like *IEnumarble* and *IQueryable*) that I just gave up. Summarizing it would have taken me about fifty pages of text and would probably confuse you more than help you. Moreover, I would have massacred a lot of Bart? ideas.

So I?l just direct you to his blog: http://community.bartdesmet.net/blogs/bart/Default.asp. Just have a look at the posts about *Isoteric LINQ bindings*, *The construction of ExceLINQ* and *LINQ to Objects Debugging*.

His blogs contains a lot of other interesting material if you?e more into the mathematic theories behind software development and programming.

A few highlights from his presentation

The evolution in C#

- C# 2.0 iterators
  - yield return
  - On-demand fetch
  - E.g. LINQ to Objects
- C# 3.0 lambdas
  - () => compute
  - On-demand evaluation
  - E.g. TPL futures

Expression translation cheat sheet:

C# 3.0 query expression translation cheat sheet

`from T x in e`	`from x in ( e ) . Cast < T > ( )`
`join T x in e on k1 equals k2`	`join x in ( e ) . Cast < T > ( ) on k1 equals k2`

`from x1 in e1`
`from x2 in e2`
`select v`
	`( e1 ) . SelectMany ( x1 => e2 , ( x1 , x2 ) => v )`

`from x1 in e1`
`from x2 in e2`
`…`
	`from * in ( e1 ) . SelectMany ( x1 => e2 , ( x1 , x2 ) => new { x1 , x2 } )`
	`…`

`from x in e where f …`	`from x in ( e ) . Where ( x => f ) …`
`from x in e select x`	`( e ) . Select ( x => x )`
`from x in e select v`	`( e ) . Select ( x => v )`
`from x in e let y = f …`	`from * in ( e ) . Select ( x => new { x , y = f } ) …`

`from x1 in e1`
`join x2 in e2 on k1 equals k2`
`select v`
	`( e1 ) . Join ( e2 , x1 => k1 , x2 => k2 , ( x1 , x2 ) => v )`

`from x1 in e1`
`join x2 in e2 on k1 equals k2`
`…`
	`from * in ( e1 ) . Join ( e2 , x1 => k1 , ( x1 , x2 ) => new { x1 , x2 } )`
	`…`

`from x1 in e1`
`join x2 in e2 on k1 equals k2 into g`
`select v`
	`( e1 ) . GroupJoin ( e2 , x1 => k1 , x2 => k2 , ( x1 , g ) => v )`

`from x1 in e1`
`join x2 in e2 on k1 equals k2 into g`
`…`
	`from * in ( e1 ) . GroupJoin ( e2 , x1 => k1 , ( x1 , g ) => new { x1 , g } )`
	`…`

`from x in e`
`orderby k1 , k2 , … , kn`
`…`
	`from x in ( e ) . OrderBy ( x => k1 ) . ThenBy ( x => k2 ) . … . ThenBy ( x => kn )`
	`…`

`from x in e group v by k`	`( e ) . GroupBy ( x => k , k => v )`
`from …1 into x …2`	`from x in ( from …1 ) …2`
