DevDays
2008
Bring your
ideas to life
LINQ TO XML

Mike Taulsty
Developer & Platform Group
Microsoft UK
Mike.Taulty@microsoft.com
http://www.miketaulty.com
Agenda

• Do we need LINQ to XML?
  – Can we make a better XML API?

• Tour of LINQ to XML
  – Creating, Querying, Modifying

• More “advanced” topics
  – Working with Schema, XPath and Large Documents

• Possible Futures
  – The “LINQ to XSD” Alpha Preview
<?xml version="1.0" encoding="utf-8" ?>

<numbers>
  <number value="1" squared="1" />
  <number value="2" squared="4" />
  <number value="3" squared="9" />
  <number value="4" squared="16" />
  <number value="5" squared="25" />
  <number value="6" squared="36" />
  <number value="7" squared="49" />
  <number value="8" squared="64" />
  <number value="9" squared="81" />
  <number value="10" squared="100" />
</numbers>
DEMO

Can we make a better XML API? Yes, we can 😊
LINQ to XML – Basic Facts

• An XML API implemented in assembly
  – System.Xml.Linq.dll

• Namespaces
  – System.Xml.Linq
  – System.Xml.Schema
  – System.Xml.XPath

• Integrates with Language INtegrated Query

• Released with .NET Framework V3.5 in Visual Studio 2008
Key Classes in System.Xml.Linq

- **System.Xml.Linq** is a “DOM like” API
  - Manipulates an XML tree in memory
- **Naturally work with both “full documents” and “fragments”**
- **The two key classes in System.Xml.Linq**

![Diagram showing key classes XElement and XAttribute]

- XElement
  - Class
  - + XContainer
- XAttribute
  - Class
  - + XObject
DEMO

Creating XML with XElement & XAttribute
System.Xml.Linq – More Classes

- XDeclaration
- XDocumentType
- XProcessingInstruction
- XComment

- IXmSerializable
- XDocument
- XElement
- XAttribute

- XText
- XCData
DEMO

Creating a More Complete Document
Xml Namespaces

- Important to make namespace support easy
- Very natural syntax for expressing names

```
XmlElement element =
    new XElement("{urn:mynamespace-com}myElement");
```

- More control over this given by two additional classes
DEMO
Working with Namespaces
Loading Xml Content

- **Loading Xml** is performed with;
  - XElement.Load
  - XDocument.Load
- **Both support loading from**
  - URI, XmlReader, TextReader

```csharp
XmlReader reader = XmlReader.Create("myData.xml");
XmlElement element = XElement.Load(reader);
```
Querying XML Content

- **XElement** has “navigation” methods
  - Descendants()
  - Ancestors()
  - etc.

- **These methods return**;
  - `IEnumerable<T>`
How does LINQ fit in here?

The *query expression* pattern in LINQ

```csharp
from itemName in srcExpr
join itemName in srcExpr on keyExpr equals keyExpr
    (into itemName)?
let itemName = selExpr
where predExpr
orderby (keyExpr (ascending | descending)?)*
select selExpr
group selExpr by keyExpr
into itemName query-body
```

Works with IQueryable<T> and IEnumerable<T>
**IEnumerable** & **IQueryable**

- **IEnumerable** – query executed piece by piece

- **IQueryable** – query executed in one go
DEMO
Loading & Querying XML
Modifying XML

- XML tree exposed by XElement and friends is not read-only
- Modifications through methods such as;
  - XElement.Add(), XElement.Remove(), etc.
- Modified tree can be persisted via
  - XElement.Save(), XDocument.Save()
  - Both supporting filename, TextWriter, XmlWriter.

```csharp
XElement element = new XElement("foo");
element.Save(@"c:\temp\foo.xml");
```
DEMO
Modifying & Saving XML
Detour

- The age-old struggle of good versus evil 😊
Working with schema

- Validation of an XML tree contained in XElement, XDocument can be done via the Validate method

```csharp
XDocument element = XDocument.Load("mydata.xml");
XmlSchemaSet schema = new XmlSchemaSet(new NameTable());

// (loading schema missed out)
element.Validate(schema, null, true);
```

- Can optionally populate the tree with the Post-Schema Validation InfoSet
  - Allows for querying via the GetSchemaInfo method
  - Means default values from the schema are now in place
DEMO

Working with Schema
Working with XPath

- **XPath available within the context of LINQ to XML**
  - Remember the System.Xml.XPath namespace
- **Extension methods**
  - XPathEvaluate
  - XPathSelectElement
  - XPathSelectElements
- **Returned data is one or more XElements – not an XPathNavigator**
DEMO

Working with XPath
Working with large XML files

• **DOM like API’s not usually suited to processing large XML files**
  – Memory usage of the DOM relates to the size of the file

• **Streaming input files**
  – No generic solution to this in LINQ to XML
  – Recommended pattern around using C# iterators to build your own *axis function* based on `XmlReader`

• **Streaming output files**
  – `XStreamingElement` class assists in this case
  – Does not build the XML tree from the query – captures it and executes it at serialisation time
DEMO
Working with Large Files
Possible Futures

• LINQ to XML code still contains quite a lot of casts and strings

```csharp
var query = from x in element.Descendants("customer")
            where (string)x.Attribute("country") == "UK"
            select (int)x.Attribute("age");
```

• LINQ to XSD (0.2 Alpha)
  – Generates strongly typed classes from XSD
  – Derived from XElement, XDocument, etc.
DEMO
LINQ to XSD
Summary

• New XML API
• Works with or without LINQ
  – A lot nicer with LINQ 😊
• Additional language support in VB 9
• Start using it today with Visual Studio 2008 & .NET Framework V3.5
• Shows up again in Silverlight 2
LINQ TO XML

Mike Taulty
Developer & Platform Group
Microsoft UK
Mike.Taulty@microsoft.com
http://www.miketaulty.com