XML is a very popular format—a very popular format—which makes it a bit surprising that it has often been more than a bit of a pain for developers to deal with.

Rockin’ It “Old School”

The first version of the .NET base classes provided two ways of parsing XML. The simple way to deal with XML was to use the XML DOM, which allows the developer to easily find elements in a document, validate them against a schema, and do that sort of thing. It is fairly simple to use,¹ but it is a fairly big stick; it takes a bit of time to load and parse XML and consumes quite a bit of memory.

The opposite end of the spectrum is represented by the XmlReader and XmlWriter classes. They are very fast at parsing XML, but they don’t provide a lot of help; the developer has to write complex code that ends up being difficult to understand and hard to change. But they are fast.

Creating XML had the same choices: the big DOM, the XmlWriter class, and the always-popular WriteLine() approach.²

Soon after, the XPathDocument class showed up, which provided a middle ground for parsing; it was faster than the DOM and much easier than the XmlReader class to use. But there was no improved way to create XML documents.

Linq to XML

Linq to XML is a bit of a misnomer. Under this umbrella are the following:

- A new syntax to create XML
- A nicer way to parse XML
- Smooth interfacing with other Linq abstractions

¹Assuming you are better at remembering how namespaces work than I am.
²Purist may cringe, but there is a lot of code written this way.
Creating XML

For a bit of context, here is some code that uses the DOM to create some XML:

```csharp
static public string CreateXmlExample1DOM()
{
    XmlDocument xmlDocument = new XmlDocument();
    XmlNode xmlBooksNode = xmlDocument.CreateElement("books");
    xmlDocument.AppendChild(xmlBooksNode);
    xmlBooksNode.AppendChild(xmlBookNode);
    XmlNode xmlNameNode = xmlDocument.CreateElement("name");
    xmlNameNode.InnerText = "Fox in socks";
    xmlBookNode.AppendChild(xmlNameNode);
    XmlNode xmlPriceNode = xmlDocument.CreateElement("price");
    xmlPriceNode.InnerText = "35.99";
    xmlBookNode.AppendChild(xmlPriceNode);
    return xmlDocument.OuterXml;
}
```

Take a look at that code, and write down the XML that it generates.
My guess is that it took you a bit of time to do that, and a similar effort is required whenever that code is read. One of the major goals of Linq to XML is to make XML creation easier to write and understand. Here is the XML that it generated:

```xml
<books>
  <book>
    <name>Fox in socks</name>
    <price>35.99</price>
  </book>
</books>
```

You will generate the same XML using the XElement class. You'll start with one of the inner elements.

```csharp
XElement element = new XElement("name", "Fox in socks");
```

---

3All the XML examples in this chapter are formatted to be easy to read. The actual generated XML may differ in whitespace.