SKILL UP!

将来にわたって使えるスキルを身につけるために

オブジェクト指向ではじめるプログラミング

日向 俊二
HYUGA, Shunji

第9回

C#でXML文書を扱う（その3）

Technology Tools

- ❑ Visual Basic .NET
- ☑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ❑ ASP.NET
- ❑ Internet Information Services
- ☑ Other:
  .NET Framework SDK

Level

★ ★ ★ ★ ★

Samples

・この記事で取り上げたソースコードおよびサンプルプログラムは、付録CD-ROMの¥DOTNET¥CSHARPディレクトリに収録しています。

¥COMSAX
MSXMLを使ったプログラム

¥DOMSAX
擬似SAXプログラム

※この記事では.NET Framework SDKを利用して解説していますが、C#Builder、SharpDevelopでも利用できます（C#コンパイラならすべて利用できます）。

はじめに

先月号では、DOM（Document Object Model：ドキュメントオブジェクトモデル）を利用してXML文書を扱うC#プログラムについて説明しました。DOMは、今回説明するSAXと共に、XML文書を扱うプログラムのためのアプリケーションインターフェイスです。

今回は、XML文書を操作するプログラムが使う標準インターフェイスのひとつであるSAXを.NET Frameworkプログラムで利用する方法について説明します。

SAXとは？

XML文書を利用したり操作するための標準化されたインターフェイスで一般的によく使われるものとして、先月号で解説したDOMのほかに、SAX（The Simple API for XML）があります。

DOMの場合は、XML文書全体をメモリにロードして、XML文書のツリーをメモリ上に保存します。そして、そのメモリ上のツリーの各ノードの情報を取り出したり追加したりします。

一方、SAXはXML文書を読み込みながら、文書（ドキュメント）の読み込み開始、要素（エレメント）の読み込み開始などのイベントをプログラムに通知します。XML文書を読み込みながら処理を行なうことができるので、ネットワークなどで転送されるデータを扱うのに適し、また、反応が早く、メモリ消費量の少ないプログラムを作成することが可能です。

SAXの仕様には、SAX1とSAX2があります（表1）。SAX2はSAX1にいくつかの機能が追加されているほかに、インターフェイスやクラスの名前やメンバの一部がSAX1と違います。これから作るプログラムには、SAX2を使ってください。SAX2-Extensionは、SAX2の含められる予定だった機能のうち、最終的にSAX2には取り入れられずSAX2の拡張として定義された仕様です。

なお、DOMはW3Cで策定されて勧

表1：SAXの仕様

仕様	内容
SAX1	Perserオブジェクトを使ってXML文書を解析するインターフェイスを定義する
SAX2	XMLReaderを使ってXML文書を解析するインターフェイスを定義する
SAX2-Extension	拡張ハンドラ（LexicalHandler、DeclHandler）インターフェイスを定義する

＊）SAX2対応のXMLパーサーであっても、SAX2-Extensionに含まれるインターフェイスをサポートしていないものもあります。

告されているW3Cの仕様ですが、SAXはW3Cの仕様ではありません。

SAXを使ったプログラミング

DOMと同様に、SAXもプログラムでXML文書を扱うときに使うインターフェイスの仕様です。SAXそのものは仕様なので、実際のプログラミングでは、SAXをサポートするXMLパーサー（XMLプロセッサ、SAXドライバなどともいいます）か、それをクラスにラップした（包み込んだ）一連のクラスを使います。XMLパーサーの機能詳細はXMLパーサーの種類によって異なりますが、SAXの仕様に従っている限り、いずれのXMLパーサーを使っても、XML文書の操作の方法は同じです。

簡単にいえば、まず、XMLパーサーが生成するイベントを処理するための一連のイベントハンドラ（イベント処理メソッドまたはイベント処理関数）を作ります。イベントハンドラさえ作れば、あとはXMLパーサーオブジェクトを作成してXML文書をパース（解析、分析）するだけです。

Windows環境では、次の2種類のプログラミング方法があります。

①.NET Frameworkより前から使われていたMicrosoftのXMLパーサーである「MSXML」を使う

②.NET Frameworkのクラスだけを使ってSAXのような処理を擬似的に行なう

この記事では、SAX2をサポートするMSXMLを使ったプログラム例と、.NET Frameworkのクラスだけを使ってSAXに似た動作を実現するプログラム例をそれぞれ説明します。

SAXのイベントとインターフェイス

すでに説明したように、SAXはXML文書を読み込みながら、イベントを検出してプログラムに通知します。たとえば、ドキュメントの読み込みを開始したときにはイベントハンドラ「startDocument」が呼び出されます。要素を読み込み始めたときには「startElement」が呼び出されます。エラーが発生したときには「error」や「fatalError」が呼び出されます。ですから、プログラムを作る前に、SAXのイベントを知っておく必要があります。SAX2の主なイベントハンドラを表2に示します。

これらのイベントハンドラは、SAXの仕様ではそれぞれ特定のインターフェイスに定義されています。たとえば、先ほど紹介したドキュメント読み込み開始イベントハンドラstartDocumentや要素読み込み開始イベントハンドラstartElementは、ContentHandlerインターフェイスに定義されています。エラーが発生したときに呼び出されるエラーイベントハンドラはErrorHandlerインターフェイスに定義されています。そのほか、属性やDTDのようなXML文書の主な構成要素を扱うためのインタ

表2：SAX2の主なイベントハンドラ

種類	名前	説明
内容イベントハンドラ	characters	要素の中のキャラクタデータを受け取ったことを通知
	endDocument	ドキュメントの終わりを受け取ったことを通知
	endElement	要素の終わりを受け取ったことを通知
	ignorableWhitespace	要素の中の無視できるホワイトスペースを受け取ったことを通知
	notationDecl	注釈（Notation）宣言を受け取ったことを通知
	processingInstruction	処理インストラクションを受け取ったことを通知
	startDocument	ドキュメントの開始を通知
	startElement	要素の開始を通知
エラーイベントハンドラ	error	リカバー可能なパーサーエラーを受け取ったことを通知
	fatalError	致命的なパーサーエラーを受け取ったことを通知
	warning	パーサーの警告を通知