売上集計

見本

概要

Chapter2のサンプルアプリケーションでは，顧客データをもとに，「性別」や「年代」別データの集計を行いました．しかし，プログラミングの現場では，いろいろな種類のデータを扱います．商品データや，売上データ，人事データ，営業日報やスケジュールのデータなどです．

そこで，今度は，Chapter2で見てきた処理を応用して，使用頻度の高そうな，商品データを例にとり，金額の集計を行ってみましょう．製品データの「単価」と「数量」を乗算して「合計金額」をもとめ，「合計金額」を加算して「総合計」を算出します．データのソートや，詳細情報の表示機能も追加してみます．

ここでは，既に後述のproduct.xml(p.222)のようなXMLファイルが作成されているものとします．XMLファイルの生成プログラムの考えかたは，基本的に，「新規データ入力」(p.84)に同じですから，トライしてみてください．

表示

ページが表示されると，既存のXMLファイル(ここでは，product.xml，p.222参照)が読み込まれ，品番と品名毎の集計が実行されます(**図1**)．この画面で，ソートや詳細情報の表示が可能となります(**図2**，**図3**)．

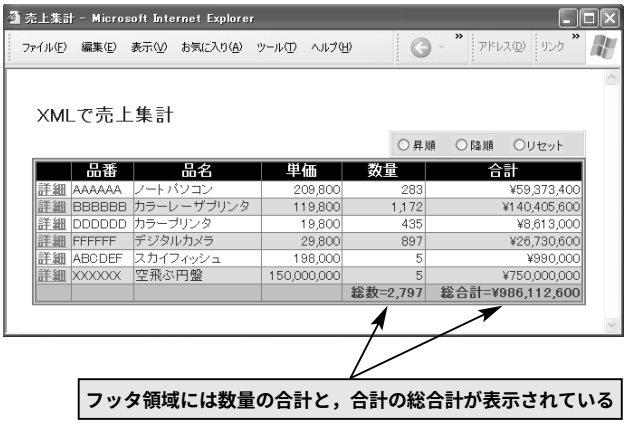


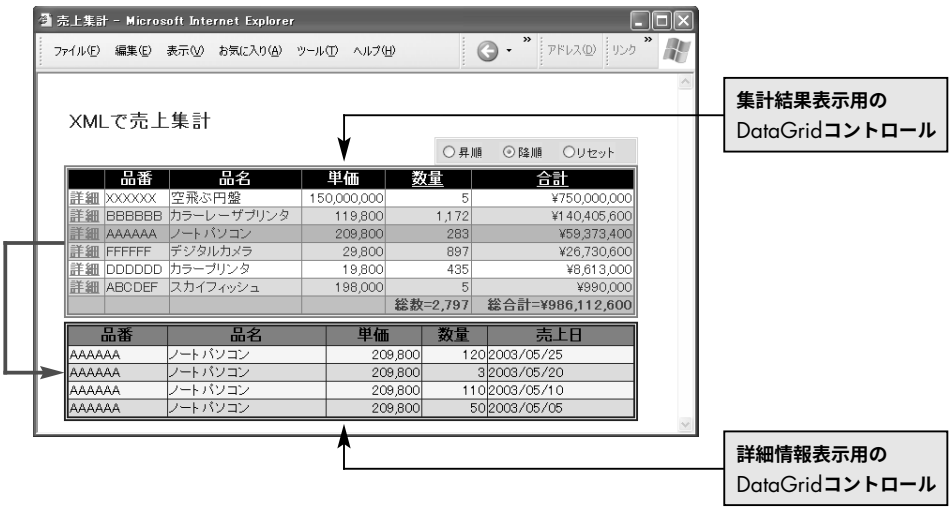
図1

品番と品名毎に，数量と合計が集計されて表示されています．また，フッタ領域には数量の「総数」と，「総合計」が表示されています．「降順」のソートを選択すると，図2のように「数量」と「合計」のヘッダテキストに，リンクのアンカーであることを示す下線が表示されます．この文字リンクスイッチをクリックすると，選択した項目をソートキーとして「降順」で並び替えが実行されます．

図2 [降順]ソートを選択した場合の表示です．合計のリンクスイッチをクリックすることにより，合計欄をキーとして，「降順」ソートが実行されます．[リセット]を選択すると，「合計」欄は元の並びに戻り，「数量」と「合計」の下線が消え，初期の状態に戻ります．
[詳細]という文字リンクスイッチをクリックすると，図3のように，選択した項目の集計前の詳細情報が表示されます．

	品番	品名	単価	数量	合計
詳細	XXXXXX	空飛ぶ円盤	150,000,000	5	¥750,000,000
詳細	BBBBBB	カラーレーザプリンタ	119,800	1,172	¥140,405,600
詳細	AAAAAA	ノートパソコン	209,800	283	¥59,373,400
詳細	FFFFFF	デジタルカメラ	29,800	897	¥26,730,600
詳細	DDDDDD	カラープリンタ	19,800	435	¥8,613,000
詳細	ABCDEF	スカイフィッシュ	198,000	5	¥990,000
				総数=2,797	総合計=¥986,112,600

図3 品番が「AAAAAA」のレコードを選択して，集計される前のデータが表示されています．売上日も表示され，日付は「降順」でソートされています．



	品番	品名	単価	数量	合計
詳細	XXXXXX	空飛ぶ円盤	150,000,000	5	¥750,000,000
詳細	BBBBBB	カラーレーザプリンタ	119,800	1,172	¥140,405,600
詳細	AAAAAA	ノートパソコン	209,800	283	¥59,373,400
詳細	FFFFFF	デジタルカメラ	29,800	897	¥26,730,600
詳細	DDDDDD	カラープリンタ	19,800	435	¥8,613,000
詳細	ABCDEF	スカイフィッシュ	198,000	5	¥990,000
				総数=2,797	総合計=¥986,112,600

品番	品名	単価	数量	売上日
AAAAAA	ノートパソコン	209,800	120	2003/05/25
AAAAAA	ノートパソコン	209,800	3	2003/05/20
AAAAAA	ノートパソコン	209,800	110	2003/05/10
AAAAAA	ノートパソコン	209,800	50	2003/05/05

コード解説

●集計用のサンプルXMLファイル(product.xml)

CD-ROM⇒cq_xml_IO/data/product.xml

このデータ集計プログラムで使うサンプルデータです．

ルート要素は<売上管理>で，<売上管理>の中には<商品情報>が複数回出現します．<商品情報>には，この商品が販売された日を記録するための"売上日"属性があります．<商品情報>の中には<品番><品名><単価><数量>が存在しています．このXMLファイル中のデー

タには，同じ品番で同じ品名の商品が複数存在します．ここの品番と品名別に，データを集計します．

【1. 集計用のサンプルXMLファイルの構造】

product.xml

```
<?xml version="1.0" encoding="Shift_JIS"?>
<売上管理>
  <商品情報 売上日="2003/5/10">
    <品番>AAAAAA</品番>
    <品名>ノートパソコン</品名>
    <単価>209800</単価>
    <数量>110</数量>
  </商品情報>
  <商品情報 売上日="2003/5/14">
    <品番>BBBBBB</品番>
    <品名>カラーレーザプリンタ</品名>
    <単価>119800</単価>
    <数量>245</数量>
  </商品情報>
～中略～
  <商品情報 売上日="2003/5/5">
    <品番>AAAAAA</品番>
    <品名>ノートパソコン</品名>
    <単価>209800</単価>
    <数量>50</数量>
  </商品情報>
<商品情報>～</商品情報>繰り返し
</売上管理>
```

売上日の異なる同一商品が複数存在する

●応用データ集計プログラム（productManagement.aspx）

CD-ROM⇒cq_xml_IO/Chapter3/productManagement.aspx

【1. @Pageディレクティブの設定】

Language属性に，使用言語としてVB.NETを指定します．Strict属性にはTrueを指定して，あいまいな型変換を禁じています．変数の宣言には全てAs句が必要になります．

```
<%@ Page Language="VB" Strict="True"%>
```

【2. 名前空間のインポート】

@ImportディレクティブのNamespace属性を使って，プログラムで使用する名前空間を指定します．XMLを扱うためのSystem.Xml名前空間をインポートし，データアクセスのためのSystem.Data名前空間をインポートしています．

①は，このサンプルで使用しているStringBuilderクラスの属する名前空間です．

StringBuilderクラスは，文字列を構築するクラスで，非常に高速な処理を提供してくれます．例えば，ある文字列の連結処理を，数十万回以上繰り返す場合を想定すると，

```
myStr=myStr & "A"
```

と記述するよりも，StringBuilderクラスのAppendメソッドを使って，

```
StringBuilder.Append("A")
```

と記述するほうが圧倒的に高速に処理されます．

②の名前空間は，DataGridコントロールに，詳細データを表示させる際に使用するクラスの含まれる名前空間です．W3C仕様のXPath1.0をサポートするクラス群が含まれます．

```
<%@ Import Namespace="System.Xml"%>
<%@ Import Namespace="System.Data"%>
<%@ Import Namespace="System.Text"%> ―――――――――――――――――――― ①
<%@ Import Namespace="System.Xml.XPath"%> ―――――――――――――――――― ②
<HTML>
<HEAD><meta http-equiv="content-type" content="text/html;charset=Shift_JIS">
<title>売上集計</title>
<script runat="server">
```

【3. 変数の宣言】

①各クラス用オブジェクト変数を宣言します．

XmlDocumentはXML文書を表すクラスです．

DataTableは，メモリ内の1個のテーブルを表すクラスです．このプログラムでは，集計結果の格納と，詳細データの格納のために2個のDataTableを用意しておきます．

DataViewはDataTableのビューを表し，このサンプルではソートの際に使用しています．これもDataTable同様，2個のDataViewを用意しておきます．

```
    Dim filePath As String
    Dim xmldoc As XmlDocument ――――――――――――――――――――――――
    Dim dt As DataTable
```

```
Dim dt2 As DataTable
Dim dv As DataView
Dim dv2 As DataView
```
①

【4. ページが読み込まれた時に発生するLoadイベントの処理】

①新しいXmlDocumentオブジェクトを生成し，LoadメソッドでXMLファイル(product.xml)を読み込みます．SelectNodesメソッドで，指定したXPath式に合致する各要素のノードリストを取得しておきます．

②新規DataTableオブジェクトを生成し，DataTableの各フィールドの項目名とデータ型を定義します．DataTableのColumnsプロパティで，DataColumnCollectionを取得します．項目名とデータ型で定義されたDataColumnオブジェクトを生成し，DataColumnCollectionクラスのAddメソッドで，DataColumnCollectionに追加していきます．この処理で，品番と品名ごとに集計された結果がバインドされる，DataGrid1というIDを持つDataGrid(p.238，Aの部分)の項目名とデータ型が定義されます．

※DataColumnCollectionはDataTable内のDataColumnオブジェクトの集合です．
※DataColumnクラスはDataTable内の列の定義を表すクラスです．

③先の②と同様の処理です．この処理で，［詳細］という文字リンクスイッチがクリックされた時に表示される，DataGrid2というIDを持つDataGridコントロール(p.239，Bの部分)の項目名とデータ型が定義されます．

④DataRowクラス用オブジェクト変数を宣言します．

※DataRowはDataTableの行を表すクラスで，System.Data名前空間に属しています．

⑤新規StringBuilderオブジェクトを生成します．StringBuilderクラスについては，【2. 名前空間のインポート】①(p.223)を参照して下さい．

⑥合計，総合計，数量の合計を格納するための変数値を0で初期化しています．

⑦①で取得した<品番>要素の，ノードリスト内のノードの数だけ反復処理を実行します．ノード数は，Countプロパティで取得することができます．

リスト1のproduct.xmlには，同じ商品が複数含まれています．これらを商品毎にまとめて集計するには，まず，商品毎の数量の合計をもとめ，その数量の合計に単価を掛けて，その商品の合計金額をもとめる必要があります．そこで，はじめに，品番毎の数量の合計を算出する処理を行います．

⑧SelectNodesメソッドで，各品番に合致する<数量>要素のノードリストを取得します．

```
Dim selectNoNode As XmlNodeList = xmldoc.SelectNodes("売上管理/商品
情報[品番='" & productNoNode(i).InnerText & "']/数量")
```

と記述して，<品番>要素の内容が，productNoNode(i).InnerTextに合致する<数量>要素のノードリストを取得しています．①で取得しておいた，XmlNodeListのproductNoNodeには，product.xmlの<品番>要素の内容が入っています．<品番>がAAAAAAで

ある<数量>のノードリストを取得し，次に <品番>がBBBBBBである<数量>のノードリストを取得する，といったように順番に各品番ごとの<数量>要素のノードリストを取得するようになります．取得したノードリストにノードが存在すれば，そのノードの内容テキストを数値に変換して加算していきます．この処理で，各品番ごとの数量の合計をもとめ，合計値を変数pieceに格納しておきます．

⑨この処理では，同一の「品番」と「品名」をまとめて数量を算出する必要があります(品番と品名が同じ商品については単価も同じです)．重複する「品番」と「品名」をまとめていくために，まず，InStr関数で，⑩で生成した文字列(変数checkString)に格納された値の中に，①で取得した<品番>と<品名>の連結された文字列が含まれているかどうかをチェックしています．InStr関数は，指定した文字列の中から，指定した文字列を検索し，最初に見つかった文字位置を返してくれます．書式は，

戻り値は`Integer=InStr(`開始位置，文字列1，文字列2，文字列の比較タイプ`)`

のように書き，「開始位置」には検索位置の数式を指定します．省略した場合は，先頭から検索されます．

「文字列1」に検索対象となる文字列を指定します(必須)．

「文字列2」に検索する文字列を指定します(必須)．

「文字列の比較タイプ」にはText(1)またはBinary(0)を指定します．この引数は任意で省略可能です．

⑩StringBuilderクラスのAppendメソッドで<品番><品名>と「,(カンマ)」を連結してStringBuilderオブジェクトの末尾に追加していきます．StringBuilderクラスについては【3. 変数の宣言】①(p.223)を参照して下さい．

※Appendは，指定した文字列をStringBuilderオブジェクトの末尾に追加していくメソッドです．

この処理によって，どのような文字列が連結されているかを，次ページの**図4**に示します．これらの文字列の中に，①で取得した<品番>と<品名>の連結された文字列が含まれているかをチェックします．文字列の追加されたStringBuilderオブジェクトを，順次ToStringメソッドで文字列に変換して，変数checkStringに格納しています．

⑪この処理では，集計結果を格納するDataTableの行を生成します．⑨のcheckResultの値を確認すると**図5**のように表示されます．戻り値が0の時の値を取り出すと，重複しない項目を取り出すことができます．そこで，InStr関数の戻り値が0の時の，<単価>の内容テキストを取得して変数priceに格納し，「単価」と「数量」を掛けて合計をもとめ，変数totalに格納します．このtotalを集計して「総合計」をもとめ，変数sumに格納します．数量の計も表示する必要があるので，⑧で取得した数量を集計して変数pieceSumに格納しています．<品番>と<品名>要素の内容テキストも取り出します．このように，①で取得した<品番>と<品名>の連結された文字列が，⑩で連結された文字列の中に含まれていない「品番」と「品名」だけを取り出していくことで，重複しない「品番」と「品名」を取得することができます．

DataRowクラス用オブジェクト変数を宣言します．NewRowメソッドを使用して複数のDataRowオブジェクトを生成し，インデックスで指定された行に値をセットします．

図4 CD-ROM⇒cq_xml_IO/chapter3/Stringbuilder.aspx参照

「品番」と「品名」と「,(カンマ)」が，StringBuilderオブジェクトの末尾に順番に追加されています．

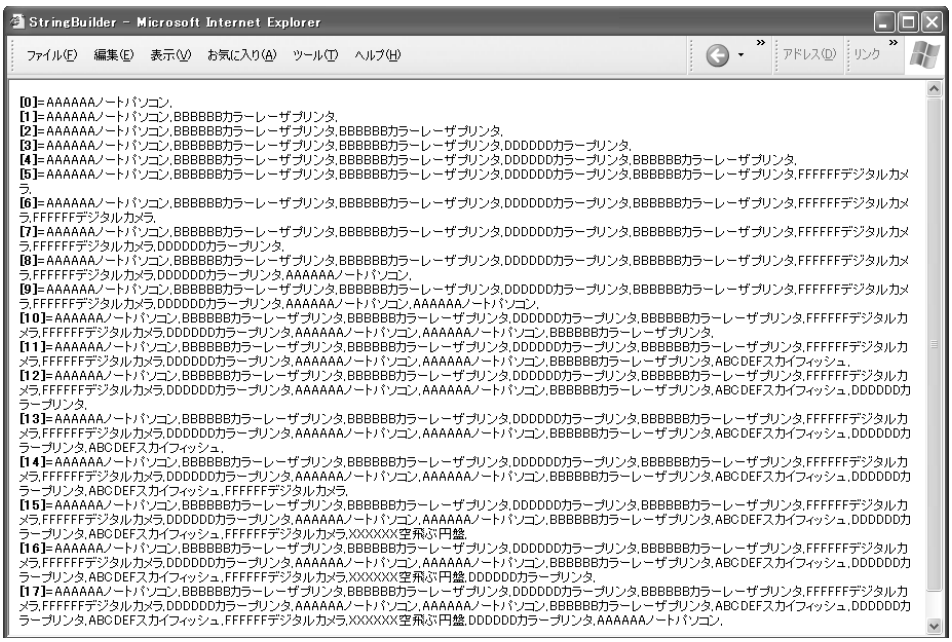
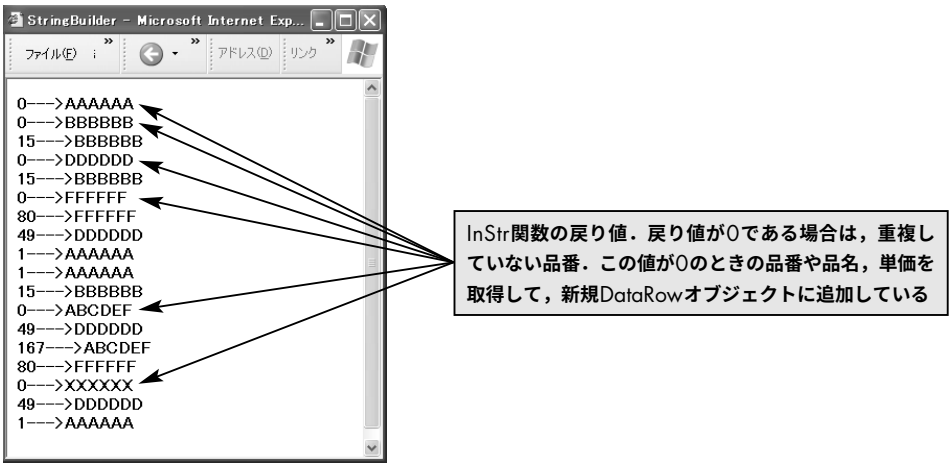


図5 CD-ROM⇒cq_xml_IO/chapter3/Stringbuilder.aspx参照




```
dr(0)= productNoNode(i).InnerText
```

のように，インデックスで指定しているDataRowには，次のように列名を指定してもかまいません．

```
dr("品番")= productNoNode(i).InnerText
```

次に，DataTableオブジェクトのRowsプロパティでDataRowCollectionを取得し，Addメソッドを使用して，これらの値がセットされたDataRowオブジェクトをDataRowCollectionに追加していきます．

※DataRowはDataTable内の行を表し，DataRowCollectionは，DataTableオブジェクト内の行の集合を表すクラスです．

⑫ViewState("pieceSum")プロパティに「数量の合計」を，ViewState("sum")プロパティに「総合計」の値を保持させておきます．この値は【6．DataGridコントロールにデータがバインドされた後に発生する処理】(p.230)で使用します．

⑬DataTableの新しいビューを生成します．集計結果表示用のDataTableと，詳細データ表示用のDataTableの，2個のビューを生成しています．この時点ではまだ表示されていませんが，詳細データ表示用のDataGridコントロールの「売上日」フィールドの値を降順でソートさせています．ソートするには，DataViewクラスのSortプロパティを使って，

```
dv2.Sort = DataGrid2.Columns(4).SortExpression
```

と記述します．Columnsのインデックスは0から始まるため，Columns(4)は「売上日」のフィールドになります(**図3**参照)．

SortExpressionプロパティは，ソートが実行される際に渡されるフィールドの名前や式を取得するプロパティで，このサンプルでは， p.239のHTML内の(C)の，次の式の値("売上日 DESC")になります．

```
<asp:BoundColumn  DataField="売 上 日 "  HeaderText="売 上 日 "
DataFormatString="{0:d}" SortExpression="売上日 DESC"></asp:
BoundColumn>
```

※DESC(Descending)は降順，ASC(Ascending)は昇順を表します．

ソートは複数列に対しても行うことができます．p.243，カコミ記事「複数列をソートする」を参照して下さい．

ページが初めて読み込まれている場合にのみ，データソースにデータをバインドするMyDataBindプロシージャを実行します．

```
Private Sub Page_Load(sender As Object,e As EventArgs)
   filePath = Server.MapPath("../")
   xmldoc = New XmlDocument()
   xmldoc.Load(filePath & "data/product.xml")
   Dim productNoNode As XmlNodeList = xmldoc.SelectNodes("//品番")
   Dim productNameNode As XmlNodeList = xmldoc.SelectNodes("//品名")
   Dim priceNode As XmlNodeList = xmldoc.SelectNodes("//単価")
   Dim pieceNode As XmlNodeList = xmldoc.SelectNodes("//数量")
```

①