プログラムによるコンテンツ生成

DocumentTools パッケージに新しいサブパッケージおよびコマンドが追加され、完全なプログラミング機能を使ってワークシートとドキュメントの内容を作成し、活用できるようになりました。プログラミングできるコンテンツは以下のとおりです。

- カスタマイズされた表
- 埋め込みコンポーネントとコード編集領域を含んだアプリケーション
- セクションとサブセクション、ドキュメントブロックと実行グループ、カスタマイズされたテキストと入出力
- 現在のドキュメントへの挿入、新規ドキュメントとしての起動と保存

Explore、ImageTools:-Embed、および新しい Tabulate コマンドなどの Maple コマンドでは、この機能がすでに採用されています。

Layout エレメント

DocumentTools:-Layout サブパッケージには、ドキュメントまたはワークシートの基本要素を作成するコマンドが含まれています。

現在使用できるコマンドは、以下のとおり、サブパッケージを読み込むためのコマンドに示されています。各コマンドにはヘルプトピックも用意されています。

```
> with(DocumentTools:-Layout)
```

[*Cell*, *Column*, *DocumentBlock*, *Equation*, *Font*, *Group*, *Image*, *InlinePlot*, *Input*, *Output*, *Row*, *Section*, *Table*, *Textfield*, *Title*, *Worksheet*] **(1.1)**

これらのコマンドを使用すると、ネスト構造の Maple 関数コールを構築できます。この関数コールには、ドキュメント全体またはワークシート全体が XML 形式で記述されます。作成されたドキュメントの内容は、挿入、起動、保存することが可能です。

以下に、コンテンツの作成および挿入について簡単な例を示します。ここで使用されるコンストラクタコマンドは、Worksheet、Table、Row、Textfield、および Equation です。

```
> E1 := Equation('sum(i', i = 1 .. n), style = TwoDimInput):
  E2 := Equation(factor(sum(i, i = 1 .. n)), style = TwoDimInput, typesetting = extended):
  T := Textfield("The sum of the first n positive integers ", E1, " is equal to ", E2, alignment = centered):
  xml := Worksheet(Table(Row(T), alignment = center, width = 50)):
```

InsertContent コマンドを使用すると、このコンテンツを現在のドキュメントに直接挿入できます。

```
> DocumentTools:-InsertContent(xml):
```

The sum of the first n positive integers $\sum_{i=1}^{n} i$

is equal to $\frac{n\,(n+1)}{2}$

埋め込みコンポーネントとアプリケーションの作成

DocumentTools:-Components サブパッケージには、埋め込みコンポーネントをコンテンツとして作成するためのコマンドが含まれています。

```
> with(DocumentTools:-Components)
```

[*Button*, *CheckBox*, *CodeEditRegion*, *ComboBox*, *DataTable*, *Dial*, *Label*, *ListBox*, *MathContainer*, *Meter*, *Microphone*, *Plot*, *RadioButton*, *RotaryGauge*, *Shortcut*, *Slider*, *Speaker*, *TextArea*, *ToggleButton*, *VideoPlayer*, *VolumeGauge*] (2.1)

対話型のアプリケーションをコンテンツとしてプログラムで作成できます。

以下の例では、固有の動作コードを持つ Plot コンポーネントおよび Button コンポーネントを作成しています。マウスで Plot コンポーネントをクリックするたびに、グローバル名 L に割り当てられたリストに新しいデータ点が追加され、データの最小二乗法による線形近似がプロットされます。Button をクリックすると、プロットをクリアします。

最初の例として、現在のドキュメントにアプリケーションを挿入します。コンポーネントの内容の作成には、Plot および Button コマンドを使用します。

この例では、2 つのプロシージャ update および clear も使用します。挿入されたアプリケーションが機能するためには、これらを定義する (および定義を保持する) 必要があります。これらのプロシージャは、各コンポーネントがクリックされるたびに呼び出されます。

```
L
```
(2.2)

```
> update := proc(comp)
    local cx, cy, v;
    global L;
      use DocumentTools, plots, CurveFitting in
        cx, cy := Do(comp(clickx)), Do(comp(clicky));
        if type(L, listlist) then
            L := [L[], [cx, cy]]
        else
            L := [[cx, cy]]
        end if;
        Do(comp = display(plot(L, style = point, symbol = solidcircle, symbolsize = 15, color = blue, axes
         = normal), plot(LeastSquares(L, v), v = 0..10), view = [0..10, 0..10]));
      end use:
  end proc:

> clear := proc(comp)
    DocumentTools:-Do(comp = plot([[0, 0]], color = white, axes = normal, view = [0..10, 0..10]))
  end proc:
```

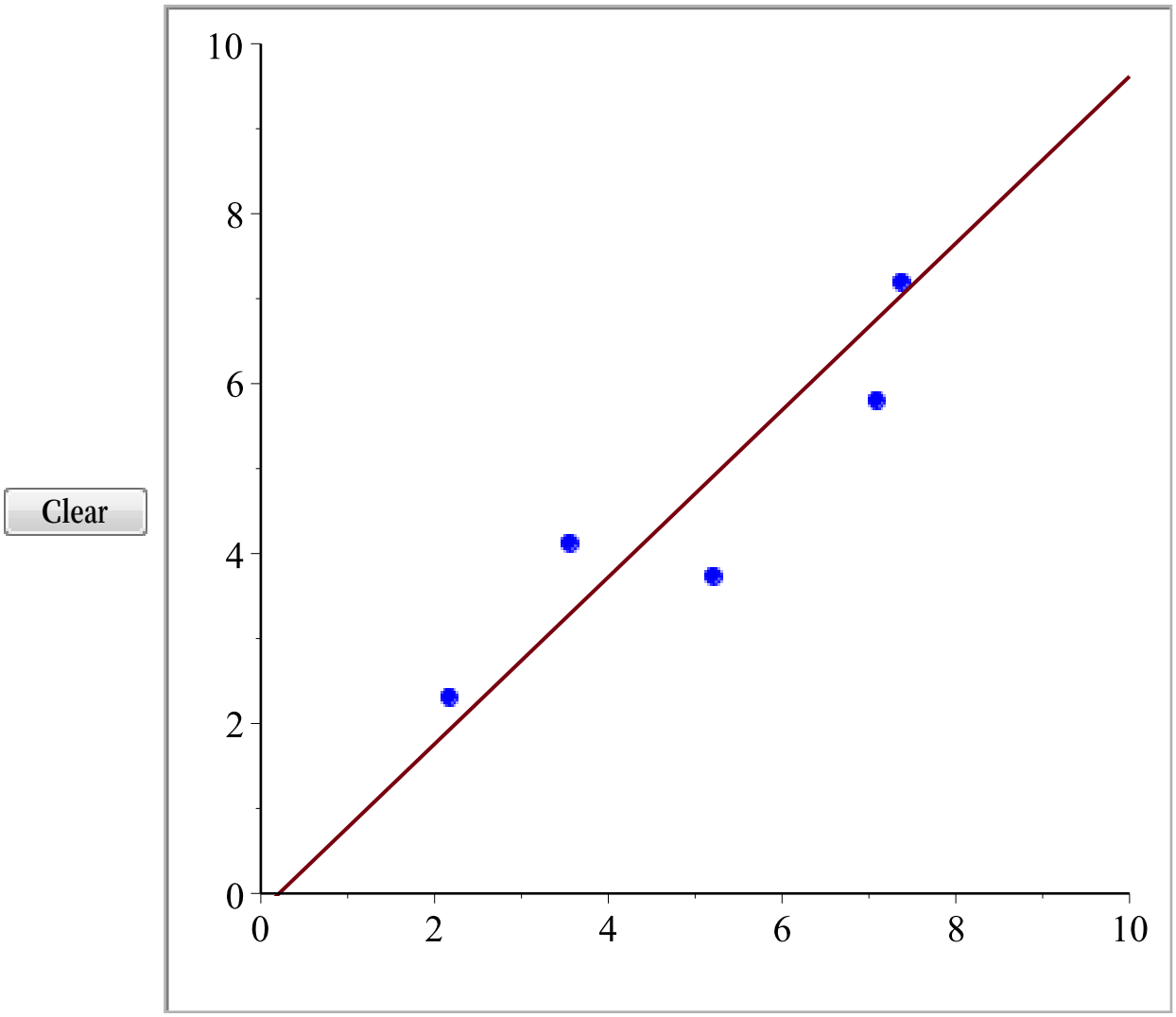
次に、2 つのコンポーネントを作成します。各コンポーネントの動作コードは、コンポーネントがクリックされ

るたびに実行され、前述のプロシージャを呼び出します。

```
> P := Plot( plot( [ [0, 0 ] ], color = white, axes = normal, view = [ 0 ..10, 0 ..10 ] ), identity = "Plot0", clickdefault,
      clickaction = "update(%Plot0);" ) :
  B := Button( "Clear", identity = "Button0", action = "unassign(':-L'); clear(%Plot0);" ) :
```

次に、コンテンツ作成を完了させ、ワークシート全体を XML 形式に変換します。これを現在のワークシートに挿入します。

```
> xmlapp := Worksheet( Group( Input( Textfield( B, P ) ) ) ) :
  DocumentTools:-InsertContent( xmlapp ) :
```



2 つ目の例として、同一のアプリケーションを再生成します。その際、補助的な update および clear プロシージャは生成後のアプリケーションの内部に定義します。これにより、新しいワークシートとして起動または保存しても、アプリケーションを機能させることができます。

この例では、プロシージャの定義が修正され、2 つのコンポーネントの動作コードに含められています。別の方法として、コンテンツにコード編集領域を追加して、そこにプロシージャ定義を含めることもできます。

```
> P2 := Plot( plot( [ [0, 0 ] ], color = white, axes = normal, view = [ 0 ..10, 0 ..10 ] ),
                identity = "Plot0", clickdefault,
                clickaction = cat( sprintf ( "update:=%a;", eval( update ) ), "update(%Plot0);" ) ) :

> B2 := Button( "Clear", identity = "Button0",
                  action = cat( sprintf ( "clear:=%a;", eval( clear ) ), "unassign(':-L'); clear(%Plot0);" ) ) :
```

```
> xmlapp2 := Worksheet(Group(Input(Textfield(B2, P2)))):
```

ContentToString コマンドを使用すると、実際のワークシートに表示する文字列を作成できます。この文字列は、新規ウィンドウで直接起動したり、ワークシートファイルとして保存することができます。

```
> appstring := DocumentTools:-ContentToString(xmlapp2):
```

このワークシート文字列を新規ウィンドウで直接起動するには、Worksheet パッケージの Display コマンドを使用します。このような新規ウィンドウでは、現在のワークシート内の定義とは無関係にアプリケーションが機能します。

以下の呼び出しでは、グローバルな名前構文 :-Worksheet を使用することで、この名前でパッケージを参照しています。前に読み込んだ Layout パッケージの名前でエクスポートする必要はありません。

```
> :-Worksheet:-Display(appstring)
```

ワークシートのテスト文字列もワークシートファイルとして保存できます。

```
> fname := FileTools:-TemporaryFile( ):
  FileTools:-Text:-WriteFile( fname, appstring):
```

保存されたファイルは、現在のワークシートの定義とは無関係に機能するアプリケーションとなります。

```
> :-Worksheet:-DisplayFile( fname)
```

Tabulate コマンド

Tabulate コマンドを使用すると、リスト、ベクトル、行列、配列などのインデックス付き構造の形式を変更し、その要素をセルの入力とした表 GUI 要素を挿入できます。

```
> M := LinearAlgebra:-RandomMatrix(6, 2, generator = 0..100)
```

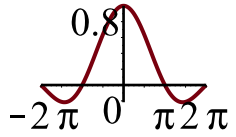
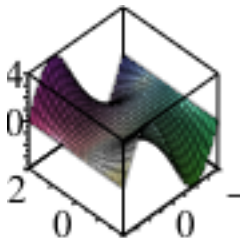
$$M := \begin{bmatrix} 99 & 91 \\ 30 & 63 \\ 22 & 89 \\ 55 & 86 \\ 77 & 76 \\ 82 & 49 \end{bmatrix} \tag{3.1}$$

```
> DocumentTools:-Tabulate(M, widthmode = pixels, width = 100)
```

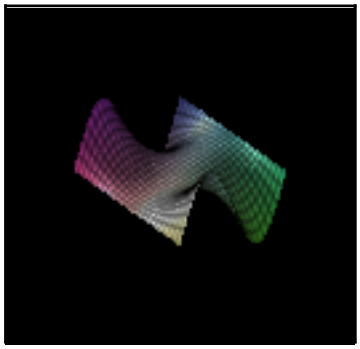
99	91
30	63
22	89
55	86
77	76
82	49

> $A := \left[plot\left(\frac{\sin(x)}{x} \right), \sum_{i=1}^{N} f(i), plot3d\left(y^2 \sin(x), x = -\pi..\pi, y = -2..2\right) \right]:$

> $DocumentTools:-Tabulate(A, width = 60)$

	$\sum_{i=1}^{N} f(i)$	

> $DocumentTools:-Tabulate\left([[plot3d\left(y^2 \sin(x), x = -\pi..\pi, y = -2..2, axes = none\right)]], fillcolor = \text{"black"}, width = 40\right)$



>

Pages That Link to This Page

updates/Maple2015/EmbeddedComponents, updates/Maple2015/Finance, updates/Maple2015/StatisticsEducation