Syntax-Highlight mit DTSyntax

Spracherweiterung für Syntax-Highlight im Listings-paket.*

Daniel Töpel

24. September 2009

*Version 1.01; zusätzliche bzw. erweiterte Sprachen: Matlab, Modelica, Pearl, VHDL.

1 Einbinden von DTSyntax

Dieses Paket wird im Deklarationsteil eingebunden.

```
\usepackage[options]{dtsyntax}
```

Optionen können sein: **framed** und **numbered**

2 Verwendung im Text

2.1 Matlab

Eingabe: `\mcode{function plotSinX(x,y)}`
Ausgabe: `function plotSinX(x,y)`

2.2 Modelica

Eingabe: `\modelica{import "work/Motor.mo";}`
Ausgabe: `import "work/Motor.mo";`

2.3 Pearl

Eingabe: `\pearl{PUT ''\\1B3D2632\\'', TerminalStr TO Terminal BY A;}`
Ausgabe: `PUT ''\1B3D2632\'', TerminalStr TO Terminal BY A;`

2.4 VHDL

Eingabe: `\vhdl{variable x : std_logic_vector(31 downto 0);}`
Ausgabe: `variable x : std_logic_vector(31 downto 0);`

3 Umgebungen

3.1 Matlab

```
\begin{lstlisting}[language=mcode]
% X-Achse anpassen
    x1 = min(x);                        % Minimum
    x2 = (ceil(max(x)/pi)) * pi;        % Maximum
    n = ceil(max(x)/pi) + 1;            % Anzahl von Unterteilungen
    XTick = linspace(x1,x2,n);
\end{lstlisting}
```

```
% X-Achse anpassen
    x1 = min(x);                        % Minimum
    x2 = (ceil(max(x)/pi)) * pi;        % Maximum
    n = ceil(max(x)/pi) + 1;            % Anzahl von Unterteilungen
    XTick = linspace(x1,x2,n);
```

3.2 Modelica

```
\begin{lstlisting}[language=modelica]
import "work/Motor.mo";

model TestMotor
  Motor Testmotor annotation (extent=[-20, 14; 0, 34]);
  Modelica.Blocks.Sources.Step Step1 annotation (extent=[-50, 18; -30, 38]);
  annotation (Diagram);
\end{lstlisting}
```

```
import "work/Motor.mo";

model TestMotor
  Motor Testmotor annotation (extent=[-20, 14; 0, 34]);
  Modelica.Blocks.Sources.Step Step1 annotation (extent=[-50, 18; -30, 38]);
  annotation (Diagram);
```

3.3 Pearl

```
\begin{lstlisting}[language=pearl]
  Task_1: TASK MAIN;
    OPEN AUSGABE;
    PUT '!!!!!!!! Hallo !!!!!!!' TO AUSGABE BY SKIP, X(5), A, (3)SKIP;
    ACTIVATE Task_2 PRIO 200;
    AFTER 2 SEC ALL 2 SEC ACTIVATE Task_4;
    AFTER 5 SEC RESUME;
    PREVENT Task_4;
    PUT TO AUSGABE BY (2)SKIP;
    CLOSE AUSGABE;
  END;
\end{lstlisting}
```

```
Task_1: TASK MAIN;
   OPEN AUSGABE;
   PUT '!!!!!!!! Hallo !!!!!!!' TO AUSGABE BY SKIP, X(5), A, (3)SKIP;
   ACTIVATE Task_2 PRIO 200;
   AFTER 2 SEC ALL 2 SEC ACTIVATE Task_4;
   AFTER 5 SEC RESUME;
   PREVENT Task_4;
   PUT TO AUSGABE BY (2)SKIP;
   CLOSE AUSGABE;
END;
```

3.4 VHDL

```
\begin{lstlisting}[language=vhdl]
entity sinx is
    Port ( xValue : in  std_logic_vector(31 downto 0);
           xResult : out  std_logic_vector(31 downto 0));
end sinx;
\end{lstlisting}
```

```
entity sinx is
    Port ( xValue : in  std_logic_vector(31 downto 0);
           xResult : out  std_logic_vector(31 downto 0));
end sinx;
```

4 Einbinden von Dateien

```
\lstinputlisting[
    caption={Hinweis zum Quelltext},
    captionpos=b|t,
    frame=none,
    language=mcode|modelica|pearl|vhdl
]{datei.m|mo|prl|vhd}
```

4.1 Matlab

```
%% Ausgabe der Sinusfunktion
% Minimalbeispiel zur weiteren Anpassung
function plotSinX(x,y)

    f = figure();

    % X-Achse anpassen
    x1 = min(x);                    % Minimum
    x2 = (ceil(max(x)/pi)) * pi;    % Maximum
    n = ceil(max(x)/pi) + 1;        % Anzahl von Unterteilungen
    XTick = linspace(x1,x2,n);

    plot(x,y); grid on;

    % Achsbeschriftungen
    xlabel('x','fontsize',10);
    ylabel('sin(x)','fontsize',10);
    legend( 'sin(x)' );

    % Formatoptionen fuer Diagramm
    set(gca,'fontsize',10, ...
            'XColor',[0.45,0.45,0.45],...
            'YColor',[0.45,0.45,0.45],...
            'XTick',XTick,...
            'XLim',[XTick(1) XTick(end)]);

    % Ausgabe der PNG-Datei
    set(gcf, 'PaperPosition', [1 1 12 8]);
    print( f, '-dpng', '-r150', 'sinx');
```

Listing 1: Bsp. Matlab: bspMatlab.m

4.2 Modelica

```
//Example: Demonstrating a redeclaration using a function-compatible function
//from: http://www.modelica.org/documents/ModelicaSpec30.pdf
function GravityInterface
    input Modelica.SIunits.Position position[3];
    output Modelica.SIunits.Acceleration acceleration[3];
end GravityInterface;

function PointMassGravity
    extends GravityInterface;
    input Modelica.SIunits.Mass m;
algorithm
    acceleration:= -Modelica.Constants.g*m*position/(position*position)^1.5;
end PointMassGravity;

model Body
    Modelica.Mechanics.MultiBody.Interface.Frame_a frame_a;
    replaceable function gravity=GravityInterface;
equation
    frame_a.f = gravity(frame_a.r0); // or gravity(position=frame_a.r0);
    frame_a.t = zeros(3);
end Body;

model PlanetSimulation
    function sunGravity = PointMassGravity(m=2e30);
        Body planet1(redeclare function gravity=sunGravity);
        Body planet2(redeclare function gravity=PointMassGravity(m=2e30));
end PlanetSimulation;
```

Listing 2: Bsp. Modelica: bspModelica.mo

4.3 Pearl

```
MODULE(hallo);  !!!!  PEARL-Testprogramm

SYSTEM;
  ausgabe: STDOUT;

PROBLEM;
  DCL AUSGABE DATION OUT ALPHIC DIM(*, 80) CREATED(ausgabe);
      ! Merke: PEARL unterscheidet bei Bezeichnern
      ! Gross- und Kleinbuchstaben

  Task_1: TASK MAIN;
    OPEN AUSGABE;
    PUT '!!!!!!!! Hallo !!!!!!!' TO AUSGABE BY SKIP, X(5), A, (3)SKIP;
    ACTIVATE Task_2 PRIO 200;
    AFTER 2 SEC ALL 2 SEC ACTIVATE Task_4;
    AFTER 5 SEC RESUME;
    PREVENT Task_4;
    PUT TO AUSGABE BY (2)SKIP;
    CLOSE AUSGABE;
  END;

  Task_2: TASK PRIO 100;
    PUT 'PEARL Installation ist ' TO AUSGABE BY X(5), A, SKIP;
    ACTIVATE Task_3;
    PUT DATE, NOW  TO AUSGABE BY X(5), A, X(4), T(8), X(3);
  END;

  Task_3: TASK PRIO 100;
    PUT '--------- O K --------' TO AUSGABE BY X(5), A, (2)SKIP;
  END;

  Task_4: TASK PRIO 200;
    PUT ' ! ' TO AUSGABE BY A;
  END;
MODEND;
```

Listing 3: Bsp. Pearl: bspPearl.prl

4.4 VHDL

```vhdl
-- ------------------------------------------------------------
-- Berechnung SIN(X)
-- ------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_textio.all;
use IEEE.numeric_bit.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;
use IEEE.math_real.all;
use std.textio.all;
use WORK.func.all;


entity sinx is
    Port ( xValue : in  std_logic_vector(31 downto 0);
           xResult : out  std_logic_vector(31 downto 0));
end sinx;

architecture berechnung of sinx is

begin

-- Berechnung sinus(xValue)
    process(xValue)
        variable temp_x : real;
        variable ergebnis : real;
        variable x_bit : Bit_Vector(31 downto 0);
        variable x : std_logic_vector(31 downto 0);
    begin

        -- slv2real, sin_x, real2slv im package func
        temp_x := slv2real(xValue);
        ergebnis := sin_x(temp_x);
        xResult <= real2slv(ergebnis);

    end process;

end berechnung;
```

Listing 4: Bsp. VHDL: bspVHDL.vhd