# **Syntax-Highlight mit DTSyntax**

Spracherweiterung für Syntax-Highlight im Listings-paket.\*

Daniel Töpel

24. September 2009

<sup>\*</sup>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

#### 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**

# 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();
5
6
       % X-Achse anpassen
7
8
       x1 = min(x);
                                         % Minimum
       x2 = (ceil(max(x)/pi)) * pi;
                                        % Maximum
9
       n = ceil(max(x)/pi) + 1;
                                        % Anzahl von Unterteilungen
10
11
       XTick = linspace(x1, x2, n);
12
13
       plot(x,y); grid on;
14
       % Achsbeschriftungen
15
       xlabel('x','fontsize',10);
16
       ylabel('sin(x)','fontsize',10);
17
18
       legend( 'sin(x)');
19
       % Formatoptionen fuer Diagramm
20
21
       set(gca,'fontsize',10, ...
                'XColor', [0.45, 0.45, 0.45],...
22
                'YColor', [0.45, 0.45, 0.45],...
23
                'XTick', XTick,...
24
                'XLim', [XTick(1) XTick(end)]);
25
26
       % Ausgabe der PNG-Datei
27
       set(gcf, 'PaperPosition', [1 1 12 8]);
28
       print( f, '-dpng', '-r150', 'sinx');
29
```

Listing 1: Bsp. Matlab: bspMatlab.m

#### 4.2 Modelica

```
1 //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];
5
   end GravityInterface;
  function PointMassGravity
       extends GravityInterface;
       input Modelica.SIunits.Mass m;
10
11
       acceleration:= -Modelica.Constants.g*m*position/(position*position)^1.5;
12
   end PointMassGravity;
13
14
15
  model Body
       Modelica.Mechanics.MultiBody.Interface.Frame_a frame_a;
16
       replaceable function gravity=GravityInterface;
17
   equation
18
       frame_a.f = gravity(frame_a.r0); // or gravity(position=frame_a.r0);
19
       frame_a.t = zeros(3);
20
21
   end Body;
22
  model PlanetSimulation
23
       function sunGravity = PointMassGravity(m=2e30);
24
25
           Body planet1(redeclare function gravity=sunGravity);
           Body planet2(redeclare function gravity=PointMassGravity(m=2e30));
26
   end PlanetSimulation;
```

Listing 2: Bsp. Modelica: bspModelica.mo

# 4.3 Pearl

```
MODULE(hallo); !!!! PEARL-Testprogramm
3
  SYSTEM;
    ausgabe: STDOUT;
4
5
  PROBLEM;
6
    DCL AUSGABE DATION OUT ALPHIC DIM(*, 80) CREATED(ausgabe);
7
        ! Merke: PEARL unterscheidet bei Bezeichnern
8
        ! Gross- und Kleinbuchstaben
9
10
    Task_1: TASK MAIN;
11
12
      OPEN AUSGABE;
      PUT '!!!!!!! Hallo !!!!!!! TO AUSGABE BY SKIP, X(5), A, (3) SKIP;
13
      ACTIVATE Task_2 PRIO 200;
14
      AFTER 2 SEC ALL 2 SEC ACTIVATE Task_4;
15
      AFTER 5 SEC RESUME;
16
      PREVENT Task_4;
17
      PUT TO AUSGABE BY (2) SKIP;
18
      CLOSE AUSGABE;
19
     END:
20
21
    Task_2: TASK PRIO 100;
22
     PUT 'PEARL Installation ist ' TO AUSGABE BY X(5), A, SKIP;
23
      ACTIVATE Task_3;
24
25
      PUT DATE, NOW TO AUSGABE BY X(5), A, X(4), T(8), X(3);
     END;
26
27
     Task_3: TASK PRIO 100;
28
      PUT '----- O K -----' TO AUSGABE BY X(5), A, (2) SKIP;
29
30
31
     Task_4: TASK PRIO 200;
32
33
      PUT ' ! ' TO AUSGABE BY A;
34
     END;
  MODEND;
```

Listing 3: Bsp. Pearl: bspPearl.prl

## **4.4 VHDL**

```
-- Berechnung SIN(X)
3
  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;
10
  use std.textio.all;
11
   use WORK.func.all;
12
13
14
15
   entity sinx is
       Port ( xValue : in std_logic_vector(31 downto 0);
16
               xResult : out std_logic_vector(31 downto 0));
17
   end sinx;
18
19
   architecture berechnung of sinx is
20
21
  begin
22
23
   -- Berechnung sinus(xValue)
24
      process(xValue)
25
           variable temp_x : real;
26
           variable ergebnis : real;
27
           variable x_bit : Bit_Vector(31 downto 0);
28
           variable x : std_logic_vector(31 downto 0);
29
       begin
30
31
           -- slv2real, sin_x, real2slv im package func
32
           temp_x := slv2real(xValue);
33
           ergebnis := sin_x(temp_x);
34
           xResult <= real2slv(ergebnis);</pre>
35
36
       end process;
37
38
   end berechnung;
```

Listing 4: Bsp. VHDL: bspVHDL.vhd