void Tdsmainform::NextGrid1CellFormating(TObject\* Sender, int ACol, int ARow, TColor& TextColor, TFontStyles& FontStyle, TCellState CellState)

{

if (NextGrid1->Cells[4][ARow] == "Yes")

{

TextColor = clGreen;

NextGrid1->Cells[ACol][ARow].Color = clYellow;

}

else

{

TextColor = clRed;

NextGrid1->Cells[ACol][ARow].Color = clBlack;

}

}

//---------- besser trennen

void \_\_fastcall Tdsmainform::NextGrid1CellColoring(TObject \*Sender, int ACol, int ARow, TGridDrawState State, TColor &Color)

{

if (NextGrid1->Cells[4][ARow] == "Yes")

{

Color = clYellow;

}

else

{

Color = clBlack;

}

}

//-------

void \_\_fastcall Tdsmainform::NextGrid1CellFormating(TObject \*Sender, int ACol, int ARow, TFontStyles &FontStyle, TColor &FontColor, Grids::TGridDrawState State, bool &DefaultDrawing)

{

if (NextGrid1->Cells[4][ARow] == "Yes")

{

FontColor = clGreen;

}

else

{

FontColor = clRed;

}

}

-----------

procedure TForm1.NextDBGrid1CellColoring(Sender: TObject; ACol, ARow: Integer; var CellColor, GridColor: TColor; CellState: TCellState); begin if (ARow < NextDBGrid1.RowCount) then begin if (not (csSelected in CellState)) and (ARow mod 2 = 0) then CellColor := RGB(245, 245, 245); end; end;" Was ist mit dem not (csEmpty in CellState) nutzen ?. Bitte in C++ den Code

Copilot

****das web****wird verwendet

Sicher, Sie können **csEmpty** in **CellState** verwenden, um zu überprüfen, ob eine Zelle leer ist. Hier ist, wie Sie den gegebenen Delphi-Code in C++ für C++ Builder übersetzen könnten:

void \_\_fastcall TForm1::NextDBGrid1CellColoring(TObject \*Sender, int ACol, int ARow, TColor &CellColor, TColor &GridColor, TGridDrawState CellState)

{

if (ARow < NextDBGrid1->RowCount)

{

if (!(CellState.Contains(gdSelected)) && !(CellState.Contains(gdEmpty)) && ARow % 2 == 0)

CellColor = RGB(245, 245, 245);

}

}

\_\_\_

void \_\_fastcall TForm1::NextDBGrid1DrawColumnCell(TObject \*Sender, const TRect &Rect, int DataCol, TColumn \*Column, TGridDrawState State)

{

// Überprüfen Sie, ob dies die Zeile ist, die Sie einfärben möchten

if (ARow == DieZeileDieSieEinfärbenMöchten)

{

// Setzen Sie die Pinsel- und Schriftfarbe des Canvas

NextDBGrid1->Canvas->Brush->Color = DieFarbeDieSieVerwendenMöchten;

NextDBGrid1->Canvas->Font->Color = clWhite;

// Füllen Sie das Rechteck der Zelle mit der Pinsel- und Schriftfarbe

NextDBGrid1->Canvas->FillRect(Rect);

// Zeichnen Sie den Text der Zelle

NextDBGrid1->Canvas->TextOut(Rect.Left, Rect.Top, Column->Field->AsString);

}

}

\_\_\_

Ja, das ist korrekt. Im Kontext von **TNextDBGrid** und ähnlichen Steuerelementen in C++ Builder und Delphi bezieht sich das **Canvas** auf die Zeichenfläche des Steuerelements, auf der Sie zeichnen können. Wenn Sie in der **OnDrawColumnCell**-Ereignisbehandlungsroutine arbeiten, bezieht sich das **Canvas** auf die Zeichenfläche der aktuellen Zelle.

Wenn Sie jedoch die gleiche Farbe und den gleichen Stil für alle Zellen in einer Zeile verwenden, wird der Effekt so aussehen, als ob Sie auf die gesamte Zeile gezeichnet hätten. Das liegt daran, dass die **OnDrawColumnCell**-Ereignisbehandlungsroutine für jede Zelle in der Zeile aufgerufen wird, und wenn Sie für jede Zelle die gleiche Formatierung anwenden, wird die gesamte Zeile einheitlich formatiert erscheinen. Ich hoffe, das hilft! Wenn Sie weitere Fragen haben, lassen Sie es mich bitte wissen. Ich bin hier, um zu helfen.