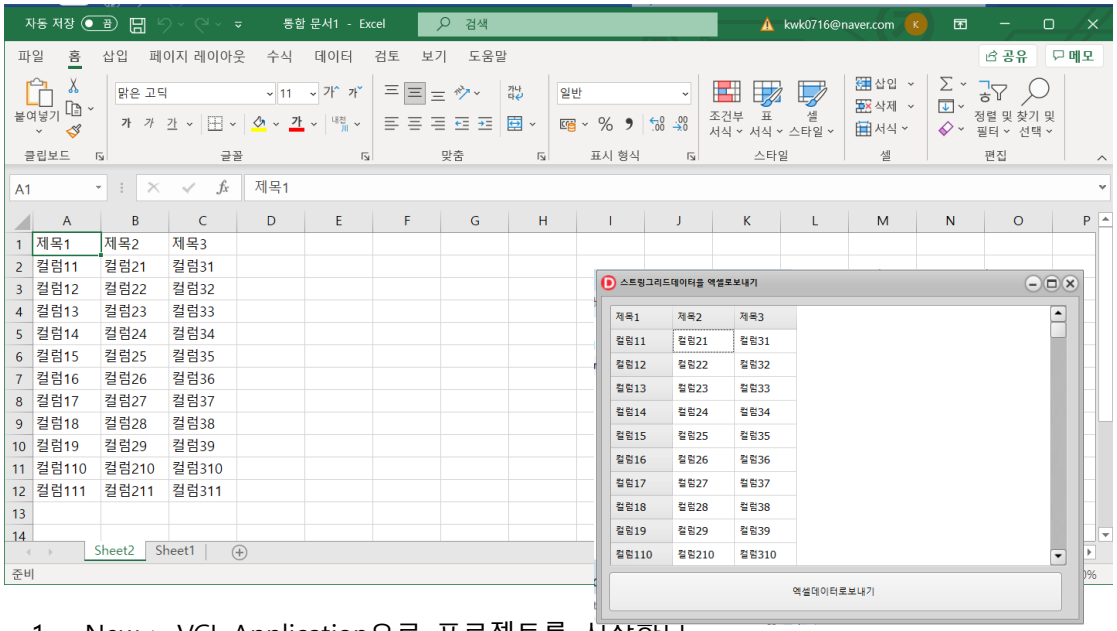
[데이터 트랙 - 향상] 엑셀 데이터를 활용하기

마이크로소프트 엑셀(MS Excel)이 설치된 환경에서는 델파이로 MS Excel Automation 기술을 이용해 엑셀파일을 직접 열어서 읽기/쓰기 작업을 할 수 있다.

델파이에서 엑셀을 다루는 방법은 두 가지 방법이 있다. 천 번째는 엑셀을 직접 컨트롤하는 방법과 두 번째는 엑셀관련 컴포넌트를 사용하는 경우이다.

[따라하기] 엑셀 직접 띄우기



1. New > VCL Application으로 프로젝트를 시작한다.
2. 폼에 StringGrid 컴포넌트를 내려놓고 Button 컴포넌트도 추가한다.
3. Uses 절에 Comobj를 추가한다.
4. Var 부분에 다음의 변수들을 선언한다.

```
var
  Excel: OleVariant;
  WorkBook: OleVariant;
  WorkSheet: OleVariant;
```

5. Form의 OnCreate 부분에 다음과 같이 코드를 구현한다.

```
procedure TForm1.FormCreate(Sender: TObject);
var
    i:integer;

begin
      StringGrid1.colcount := 3;
```

```
        StringGrid1.rowCount := 11;
        with StringGrid1 do
        begin
          Cells[0,0] := '제목1';
          Cells[1,0] := '제목2';
          Cells[2,0] := '제목3';
          for i := 1 to RowCount do
          begin
               Cells[0,i] := '컬럼1' + inttostr(i);
               Cells[1,i] := '컬럼2' + inttostr(i);
               Cells[2,i] := '컬럼3' + inttostr(i);
          end;
        end;
end;
```

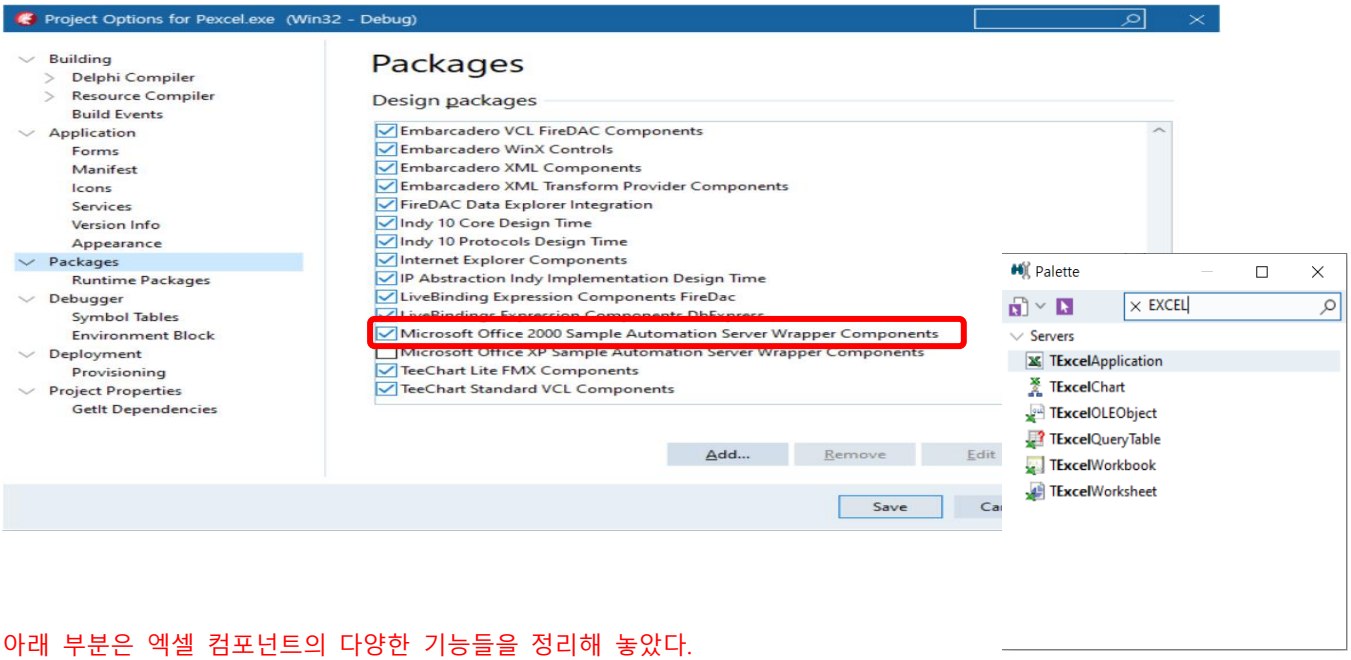
6. 엑셀보내기 버튼의 이벤트 핸들러를 다음과 같이 작성합니다.

```
    procedure TForm1.Button1Click(Sender: TObject);
    var
      i:integer;
    begin
     Excel := CreateOleObject('Excel.Application');
        Excel.Visible := True;
      // 워크북 추가
       Excel.WorkBooks.Add;
       WorkBook := Excel.ActiveWorkBook;
      // 워크시트 추가
       Workbook.sheets.add;
      try
        // 작업할  워크시트 선택
        WorkSheet := WorkBook.WorkSheets[1];
         for I := 0 to StringGrid1.RowCount do
         begin
           WorkSheet.Cells[i+1,1].value := StringGrid1.Cells[0,i];
           WorkSheet.Cells[i+1,2].value := StringGrid1.Cells[1,i];
           WorkSheet.Cells[i+1,3].value := StringGrid1.Cells[2,i];
         end;
      finally
        // 워크북 닫기
        WorkBook.close;
        WorkBook:=unAssigned;
```

```
        WorkSheet:=unAssigned;
        // 엑셀 종료
        Excel.Quit;
        Excel:=unAssigned;
      end ;
  end;
```

[팁] 다음 방법은 엑셀 관련 컴포넌트를 사용하는 방법입니다.

Project > Options > Packages 에서 아래와 같이 Microsoft office 2000을 선택하고 Save 버튼 누르면 툴 팔레트에 Excel 관련 컴포넌트들이 표시된다.



아래 부분은 엑셀 컴포넌트의 다양한 기능들을 정리해 놓았다.

LCID?

LCID는 엑셀(Excel)에서 "xlListDataTypeCurrency"는 화폐(Currency) 심볼을 가리키며, 엑셀 컬럼타입에 데이터 타입이 지정되지 않았다면 항상 0을 리턴한다.

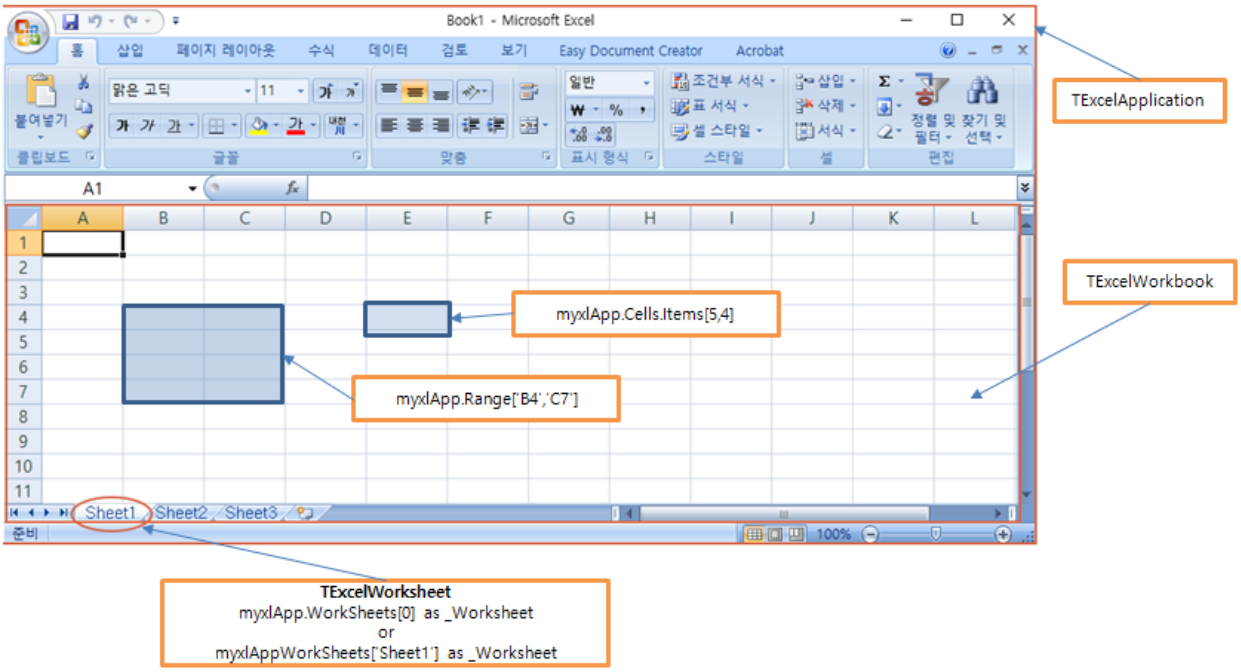
LCID를 얻는 방법은 다음과 같다

```
// Global 변수로 설정해서 초기화 부분에서 획득해 주면 된다.
LCID := GetUserDefaultLCID;
```

Excel문서 구조 및 변수 선언

엑셀를 이용해 불러오기 위해서는 다음과 같이 변수를 선언해야 한다.

```
private
    // 용도에 따라서 전역(Global) 또는 지역(Local) 변수로 선언하면 된다.
    ExcelApp: TExcelApplication;
    myxlBook: TExcelWorkbook;
    myxlSheetLegacy: TExcelWorksheet;    //기존 시트 읽기용
    myxlSheetNew: TExcelWorksheet;       //신규 시트 추가용
```



Excel 연결을 위한 Application 객체 생성

```
myxlApp := TExcelApplication.Create(Nil);
myxlApp.Connect;
myxlApp.Visible[LCID] := True; // 엑셀을 연결하고 화면에 노출되도록 한다
```

엑셀 연결 종료 및 해제

```
myxlApp.Disconnect;
myxlApp.Quit;
FreeAndNil(myxlApp);
```

WorkBook 추가/연결

```
//엑셀에 새로운 WorkBook을 추가하고 활성화된 워크북을 연결한다.
myxlApp.Workbooks.Add(EmptyParam, LCID);
myxlBook := TExcelWorkbook.Create(myxlApp);
myxlBook.ConnectTo(myxlApp.ActiveWorkbook);
```

WorkBook 연결끊기

```
//엑셀문서를 닫기전에 반드시 WorkBook의 연결을 끊어야 한다.
//연결을 끊기 전에 편집된 문서를 저장한다.
myxlBook.Close(True,'C:₩ExcelTest₩ExcelTest1.xlsx');
myxlBook.Disconnect;
FreeAndNil(myxlBook);
```

WorkSheet 추가 및 연결

```
myxlSheetLegacy := TExcelWorksheet.Create(myxlBook);
myxlSheetLegacy.ConnectTo(myxlBook.ActiveSheet as _worksheet);
//현재 활성화된 워크시트 연결
myxlSheetLegacy.Name := 'LegacySheet1';
//컴포넌트 이름을 변경해준다, 컴포넌트 이름이 중복되면 안되기 때문
```

워크시트(WorkSheet) 연결끊기

```
myxlSheetLegacy.Disconnect;
FreeAndNil(myxlSheetLegacy);
```

워크북(WorkBook)에 새 시트(WorkSheet) 추가하기

```
myxlBook.Worksheets.Add(EmptyParam, EmptyParam, EmptyParam, EmptyParam, LCID);
myxlSheetNew := TExcelWorksheet.Create(myxlBook);
myxlSheetNew.ConnectTo(myxlBook.ActiveSheet as _worksheet);
myxlSheetNew.Name := 'NewSheet1';
```

시트이름 또는 인덱스로 시트(WorkSheet) 접근하기

```
(myxlApp.Worksheets[0] as _Worksheet).Activate(LCID);
// Or
(myxlApp.Worksheets['Sheet1'] as _Worksheet).Activate(LCID);
```

Cell과 Range를 이용한 값(Value) 입력

```
//열과 컬럼을 이용한 값 할당(현재 Active Sheet에)
myxlApp.Cells.Item[1,1] := 'Value 1';
//셀이름을 이용한 범위지정을 통한 값 할당(현재 Active Sheet에)
myxlApp.Range['A3','A3'].Value := 'value 2';
//열과 컬럼을 이용한 값 할당(지정된 Sheet에)
myxlSheetLegacy.Cells.Item[1,5] := 'JITENDRA';
//셀이름을 이용한 범위지정을 통한 값 할당(지정된 Sheet에)
myxlSheetLegacy.Range['E3','E3'].Value := '7834911261';
```

범위를 지정해 서식 변경하기

```
with myxlSheetLegacy.Range['A1', 'B3'] do
begin
   Font.Name := 'Verdana';
   Font.Size := 15;
   Font.Bold := True;
   Font.Strikethrough := True;
   Font.Color := clRed;
end;
```

범위를 지정해 셀의 배경색 변경하기

```
with myxlSheetLegacy.Range['A1', 'A1'].Interior.Color := clYellow;
// 지정된 범위의 셀을 병합하고 컬러를 변경
myxlSheetLegacy.Range['A5', 'D7'].Merge(False);
myxlSheetLegacy.Range['A5', 'D7'].Interior.Color := clRed;
```

범위를 지정해 셀 합치기

```
//Merge 파라미터가 True인 경우에는 행별로 셀을 병합
```

```
myxlSheetLegacy.Range['A5', 'D7'].Merge(False);
myxlSheetLegacy.Range['A5', 'D7'].Value := 'Merged data';
```

범위를 지정해 셀의 높이와 길이 변경하기

```
myxlSheetLegacy.Range['B5', 'B5'].ColumnWidth := 5;      //단일컬럼 폭 변경
myxlSheetLegacy.Range['J5', 'L8'].ColumnWidth := 15;     //다중컬럼 폭 변경
myxlSheetLegacy.Range['B5', 'B5'].RowHeight := 50;       //단일열 높이 변경
myxlSheetLegacy.Range['J10', 'J15'].RowHeight := 50;     //다중열 높이 변경
```

이미 존재하는 워크북(WorkBook) 열기

```
myxlApp.Workbooks.Open ( 'C:₩ExcelTest₩ExcelTest1.xlsx'
EmptyParam , EmptyParam , EmptyParam , EmptyParam ,
EmptyParam , EmptyParam , EmptyParam , EmptyParam ,
EmptyParam , EmptyParam , EmptyParam , EmptyParam , 0 );
```

동일시트(Sheet)내에서 셀의 복사와 붙여넣기

```
// #1
myxlSheetLegacy.UsedRange[LCID].Copy(myxlSheetLegacy.Range['J10', 'J10']);
myxlSheetLegacy.Range['A5', 'D7'].Copy(myxlSheetLegacy.Range['J10', 'J10']);

// #2
myxlSheetLegacy.UsedRange[LCID].Copy(EmptyParam);
myxlSheetLegacy.Range['J10', 'J10'].PasteSpecial(xlPasteAll, xlPasteSpecialOperationNone,
EmptyParam, EmptyParam);

myxlSheetLegacy.Range['A5', 'D7'].Copy(EmptyParam);
myxlSheetLegacy.Range['J10', 'J10'].PasteSpecial(xlPasteAll, xlPasteSpecialOperationNone,
EmptyParam, EmptyParam);
```

다른 시트로 복사/붙여넣기

```
// #1
myxlSheetLegacy.UsedRange[LCID].Copy(myxlSheetNew.Range['J10', 'J10']);
```

```
myxlSheetLegacy.Range['A5', 'D7'].Copy(myxlSheetNew.Range['J10', 'J10']);

// #2
myxlSheetLegacy.UsedRange[LCID].Copy(EmptyParam);
myxlSheetNew.Range['J10', 'J10'].PasteSpecial(xlPasteAll,
xlPasteSpecialOperationNone,EmptyParam, EmptyParam);

myxlSheetLegacy.Range['A5', 'D7'].Copy(EmptyParam);
myxlSheetNew.Range['J10', 'J10'].PasteSpecial(xlPasteAll,
xlPasteSpecialOperationNone,EmptyParam, EmptyParam);
```

선택된 셀(범위) 초기화 하기

```
myxlSheetLegacy.Range['b3', 'b10'].ClearContents; //내용을 지우고
myxlSheetLegacy.Range['b3', 'b10'].ClearFormats;  //서식을 지운다
```

지정된 범위 자동 채우기

```
myxlSheetLegacy.Range['p1', 'p1'].Value := 1;
myxlSheetLegacy.Range['p2', 'p2'].Value := 2;
myxlSheetLegacy.Range['p1', 'p1'].AutoFill(myxlSheetLegacy.Range['p1', 'p10'], xlFillSeries);
{   Other fill options
    xlFillCopy
    xlFillDays
    xlFillDefault
    xlFillFormats
    xlFillMonths
    xlFillSeries
    xlFillValues
    xlFillWeekdays
    xlFillYears
    xlGrowthTrend
    xlLinearTrend
}
```

워크시트/워크북 인쇄 미리보기 /

```
myxlSheetLegacy.PrintPreview;
```

```
myxlSheetLegacy.PrintOut;
myxlBook.PrintPreview;
myxlBook.PrintOut;
```

워크시트(WorkSheet) / 워크북(WorkBook) 저장

```
myxlSheetLegacy.SaveAs('Filename');
myxlBook.Save;
```

지정된 범위의 테두리 변경하기

```
 myxlSheetLegacy.Range['p3', 'p4'].Borders.Color := clRed;
myxlSheetLegacy.Range['p3', 'p4'].Borders.LineStyle := xlDouble;
myxlSheetLegacy.Range['p3', 'p4'].Borders.Item[xlEdgeLeft].Color := clBlue;
myxlSheetLegacy.Range['p3', 'p4'].Borders.Item[xlEdgeRight].Color := clBlue;
{ Other line styles.
    xlContinuous
    xlDash
    xlDashDot
    xlDashDotDot
    xlDot
    xlDouble
    xlSlantDashDot
    xlLineStyleNone
}
```

지정된 범위의 채움 스타일 변경하기

```
myxlSheetLegacy.Range['p3', 'p4'].Interior.Pattern := xlPatternCrissCross;
myxlSheetLegacy.Range['p3', 'p4'].Interior.PatternColor := clBlue;
{   Other pattern styles
      xlPatternAutomatic
      xlPatternChecker
      xlPatternCrissCross
      xlPatternDown
      xlPatternGray16
      xlPatternGray25
```

```
    xlPatternGray50
    xlPatternGray75
    xlPatternGray8
    xlPatternGrid
    xlPatternHorizontal
    xlPatternLightDown
    xlPatternLightHorizontal
    xlPatternLightUp
    xlPatternLightVertical
    xlPatternNone
    xlPatternSemiGray75
    xlPatternSolid
    xlPatternUp
    xlPatternVertical
}
```

지정된 범위 셀들의 집계(SUM/AVG/MAX/COUNT)등 처리

```
myxlSheetLegacy.Range['k1', 'k1'].Formula := '=Sum(p3:p8)';
myxlSheetLegacy.Range['k3', 'k3'].Formula := '=Avg(p3:p8)';
myxlSheetLegacy.Range['k5', 'k5'].Formula := '=Max(p3:p8)';
myxlSheetLegacy.Range['k7', 'k7'].Formula := '=Count(p3:p8)';
```