Vba create new worksheet with name from cell

Koen has a worksheet that has a list of names in column A. He must create a worksheet for each name in the list and have the worksheet named according to that name. Koen suspects a macro will be needed, but he doesn't know how to do such a task. This task is relatively easy to do if you're using a macro, and there are a number of ways you could go about it. An easy way is to select the worksheet name list, and then run the following macro. Under AddWorksheetsFromSelection() Dim CurSheet As MeasfoieDim Source As Range Dim c As Range Set CurSheet = ActiveSheet Set Source = Selection.Cells Application.ScreenUpdating = False For Each c In Source sName = Trim(c.Text) If Len(s Name) > 0 Then Worksheets. Add After:=Worksheets(Worksheets. Count) ActiveSheet.Name = sName End If Next c CurSheet.Activate Application.ScreenUpdating = True End Under Macro essentially grabs each cell in the selection, creates a new worksheet, and then renames that worksheet based on what was in the cell. The macro checks to make sure that a particular cell actually contains something (you can't rename a worksheet if there's no name in the cell), but it's still not as robust as it might be. There may be other flaws in the worksheet name list that may result in errors while running the macro. For example, what happens if the list contains duplicates? Or does it contain names that Excel doesn't allow? These (and any number of other errors) could be anticipated and the code changed to manage such situations. While using a macro to create worksheets is quick and easy, keep in mind that you don't need to use a macro. In fact, you can use Excel's PivotTable capabilities to create the worksheets you want. Let's say, for the sake of this example, that the worksheet name you want is in column A of a worksheet and that cell A1 contains a heading for the column (such as Name or Worksheets). What you want to do is create a PivotTable that is based on these names. Follow these steps: Select any worksheet name in the column. Show the Insert ribbon tab. Click the PivotTable tool on the left side of the ribbon. Excel displays the Create PivotTable dialog box, with the name range of the worksheets already specified. (See Figure 1.) Figure 1. Create PivotTable dialog box. Click OK. Excel creates PivotTable and displays the PivotTable Fields pane to the right of the screen. In the PivotTable Fields pane, click the check box next to the field used for the worksheet list. (It should be something like Name or Worksheets.) Excel adjusts the PivotTable Report. Drag the name of the check-in field (Name or Worksheets) to the Filters area in the PivotTable Fields panel. (See 2.) Figure 2. PivotTable Fields panel with a set of filters. Make sure the Ribbon Analysis tab is displayed. (It should have been displayed by default after you created the PivotTable.) Click the down arrow below PivotTable PivotTable on the left side of the ribbon. Excel shows some options you can make. Click the down arrow to the right of the Options option. (Don't click the Options option itself; which displays a dialog box. You just want to click the down arrow.) Choose the Show Report Filter Pages option. Excel displays the Show Report Filter Pages dialog box. Click OK. Excel creates a worksheet for each worksheet name in the list. It is important to realize that at this point each of the new worksheets contains a small PivotTable. To get rid of these PivotTables, you might think you can create a selection set for new worksheets (click the first tab of the worksheet, and then hold down Shift while you click the last tab of the worksheet), and then press the Delete key. However, in my testing, this doesn't work - Excel won't allow you to make changes to the PivotTable in group editing mode. Instead, you'll need to display each worksheet in turn and delete PivotTable Tables. This may seem like a lot of work, but if you just need to create all these worksheets once, it can be a relatively quick way to do this without the need to invoke a macro. ExcelTips is your source. This tip (13463) applies to Microsoft Excel 2007, 2010, 2013 and 2016. In this post, I will share a short snippet of VBA code that creates new worksheets based on names stored in cell values. This can be useful when you have a list of departments and you want to create a worksheet for each department. Or a list of accounts, or employees, or regions, and so on. This post is the result of an anonymous question I received. So for whoever asked for it... thank you for your question and I hope this post helps! Goal Before we get too far, let's take a look at our goal. We have a few cell values in a regular range, like this: We would like to be able to select the range and have Excel create and name a new sheet for each item in our range. Since I'm aware of any built-in way to achieve this in the version of Excel I'm using as I write this, we'll turn to VBA. VBA VBA (Visual Basic for Applications) is a programming language that we can use to write macros. The good news is I've already written the code so you can swipe it. If you're not concerned about understanding how the macro works, go to the end of the post and download the Excel file (which contains the VBA code). But if you want to analyze and understand the code, here is a quick run down. Fragment of code Let's start by taking a look at the code: Under Insert_Sheet_Names() For each c in the selection with sheets. Add(After:=ActiveSheet) . = c.Value End With Next c End Sub Now let's talk through it. Under and below the related end lines define the procedure name (Insert_Sheet_Names) and the body of the code. The lines For each and the next related define a collection loop. A loop is a segment of code that can be repeated several times. In our case, it will be repeated once for each cell (c) in the selected range (Selection). The With and Cut lines that Excel says add a new worksheet after the active sheet, and then name it with the value in the cell. Use A possible way to use this code is to follow these steps: Download and extract the zip file below Open the workbook enabled for InsertSheets.xlsm whenever you need to perform this task Select a range of cells from any open workbook And then run the macro (View > Macros > View Macros or Alt+F8) : The macro can work on any open workbook, and you don't need to copy the VBA code to the workbook that contains the sheet name range. As with other modes in Excel, there are other ways to use this code, including adding it to your personal macro workbook. Conclusion VBA is a great language, and there are many other snippets of code that will accomplish the same thing. Additionally, there are several ways to improve the code to manage errors, for example, if the value in the cell does not meet the specifications for a sheet name, or if the current selection is not a range of cells and is instead a chart or other object. If you have any alternatives or improvements to the code, please share by posting a comment below... Thank! Example file: InsertSheets.zip Note: For security reasons, the macro-enabled workbook was entered in a zip folder. For use, download the zip file and extract the InsertSheets.xlsm file. Author: Oscar Cronquist Article last updated on January 15, 2020 This article demonstrates a macro that inserts new name-based worksheets into a range of cells. The range of cells can have multiple columns if you want. This macro allows you to create new worksheets very quickly. this animated image macro below shows this macro works. Press Alt + F8 to open the Macro dialog box. Select the CreateSheets macro. Click the Run button. An input box appears that requires a range of cells. Select a range of cells and click the OK button. VBA Macro 'Macro Name Under CreateSheets() 'Dimension Variables and Declares Data Types Dim rng As Range Dim Cell As Range 'Enable error handling On Error GoTo Errorhandling 'Shows Enter user and requests a set rng cell range = Application.InputBox(Prompt:=Select celle range:, _ Title:=Create sheets, _ Default:=Selection.Address, Type:=8) 'Iterate through cells in selected cell range For Each sell In rng 'Check if cell is not empty If cell <> Then 'Insert measier thought and name and name the send on lel olity Sheets.Add.Name = cell End If 'Continue with next cellage age in cergo littery 'Go here if oror handling: 'Stop macro End Sub Where to thenity to put the code Copy over VBA code. Press Alt + F11 to open the Visual Basic Editor. Click the workbook in Project Explorer. Click Insert from menu. Click Mode. Paste the VBA code into the code window, see the image above. Explaining the code Creating procedures in Excel is easy. Easy. Visual Basic Editor using one of these instructions: Press Alt+F11 Go to the Developer tab and click the Visual Basic button Create macro procedures in a module. First create a module. Right-click the workbook in the project explorer. Click Insert | Module. Under CreateSheets() Type: Under CreateSheets() in mode. CreateSheets() is the name of the macro. Dim rng As dim interval cell as range These lines declare rng and cell as range objects. A range object can contain a single cell, multiple cells, column, or row. Read more about declaring variables. At Goto Errorhandling Error If the user selects something other than a range of cells, there would be a chart, this line makes the procedure go to Errorhandling. Set rng = Application.InputBox(Prompt:=Select Cell Range:, _ Title:=Create Sheets, _ Default:=Selection.Address, Type:=8) The input box requires the user to have a range of cells. The range of cells is stored in the range object rng field. For each cell in rng It stores each cell value from the rng range object to the cell object, one by one. If cell <> Then verify that the cell variable is NOT empty. If the cell variable is empty, the procedure goes to the End If line. We can't create a sheet without a name. Sheets.Add.Name = cell Creates a new named sheet with the value stored in the cell variable. Ends if the If statement ends. The next cell Return to the For Each statement and store a new cell in the cell object. Errorhandling: The procedure goes to this line if a line returns an error. Termination under all procedures must end with this line. Recommended Read List of all open workbooks and corresponding sheets (vba) (vba)