LABS

Welcome to Level Three

Extend

Extend is the third Lab in a series of three intructables in the N+7 Project

If you haven’t already, download the PROIV Lab from the website homepage and watch the Lab One video for a guide on installation and the Developer Environment

Introduction

The previous exercises have shown how simply you can create and publish a function using PROIV rules technology.

Now what?

This Lab looks at updating the function such that it will create an excel spreadsheet with data on how it has been used.

As an example, this spreadsheet will simply contain the Date and Time of the generator being used, it will also log the input and output text.

There are many more possibilities to extend the framework using your Java skills.

In this PDF you will find:

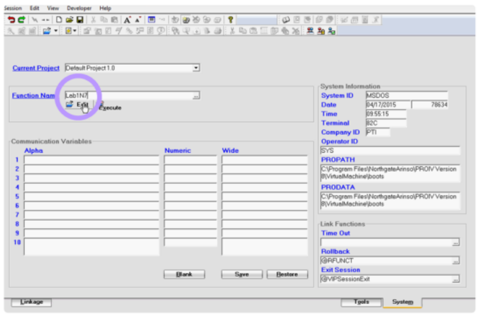
Adding an SSO to a Cycle

Create a Cycle for your SSO

Begin on the starting screen as with the two previous Labs

Type your Function Name **Lab1N7**

Click **Edit**

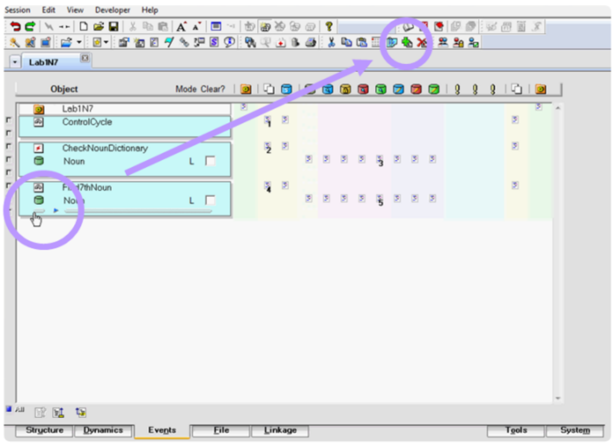


The **Structure** View will open

Go to the **Events** View

Select the Find7thNoun Cycle

Click the Add Icon



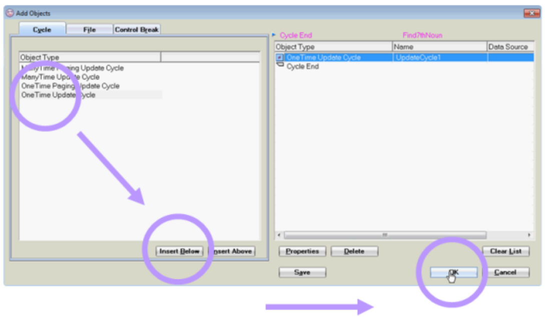
In the Add Objects window

This is a process that will happen once each use of the function

Select **OneTime Update Cycle**

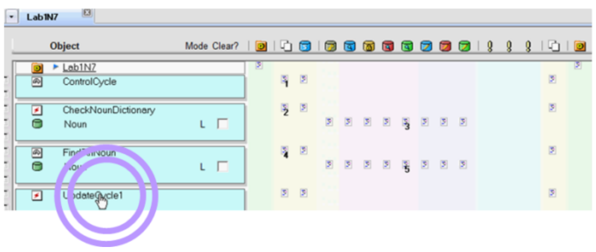
Click **Insert Below**

Click **OK**



A New Cycle will appear in your function.

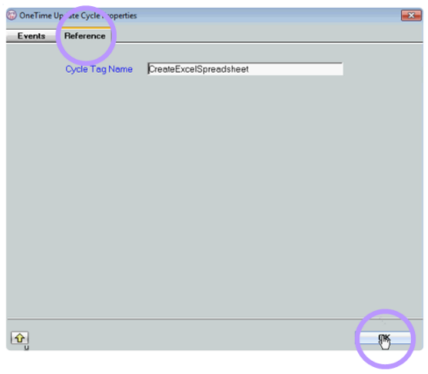
To rename the cycle, double click



Go to the **Reference** Tab

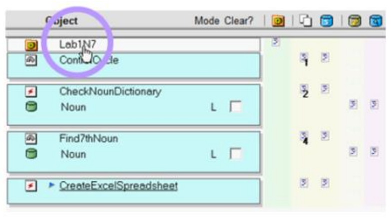
Name your Function **CreateExcelSpreadsheet**

Click **OK**



Cycle Tag Name: CreateExcelSpreadsheet

Now you have returned to the full view of the function



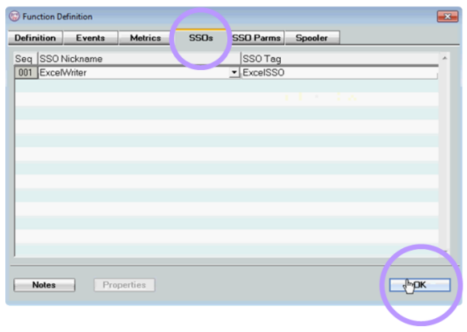
Click **Lab1N7** at the top of your cycles to open the Function Definition

In the Function Definition window

Select the **SSO** tab

In the dropdown, select the SSO we have added, @**ExcelWriter**

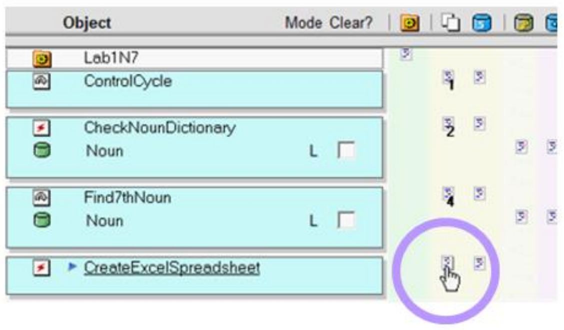
Add a name to refer to it in this function we have called it, **ExcelSSO**



Seq	SSO Nickname	SSO Tag
001	ExcelWriter	ExcelSSO

With this in place, go to the CreateExcelSpreadsheet cycle

Click on the Entry to the cycle



Insert the Rules for the Server Side Object

Include the following by Copy & Paste

```
//Initialise Variables
$OutputFile = 'C:\Temp\proivLab.xls'          //location and name of the Excel File (Client Side)
$WorksheetName = 'N + 7 Generator'            //Worksheet Name
$Alignment = 'L'                              //Default alignment is Left justified
//Excel SSO (begin)
ExcelSSO.setXlsFileName(fileName)              //Set Excel File Name
ExcelSSO.setWorksheetName(sheetName)  //Set Excel WorkSheet Name
ExcelSSO.openExcelFile()                       //Open Excel File To Write (New File)
//Create Date column Headings
ExcelSSO.addNewExcelRow()
$ColumnText = 'Date'
#Column = 0
#Width = 10
ExcelSSO.setColumnWidth(Width)
ExcelSSO.writeExcelCell(Column)
//Create Time column Headings
$ColumnText = 'Time'
#Column = 1
#Width = 10
ExcelSSO.setColumnWidth(Width)
ExcelSSO.writeExcelCell(Column)
//Create Input column Headings
$ColumnText = 'Input Text'
#Column = 2
#Width = 80
ExcelSSO.setColumnWidth(Width)
ExcelSSO.writeExcelCell(Column)
//Create output column Headings
$ColumnText = 'Output Text'
#Column = 3
#Width = 80
ExcelSSO.setColumnWidth(Width)
ExcelSSO.writeExcelCell(Column)
//Create Number of Nouns column Headings
$ColumnText = 'Number of Nouns'
#Column = 4
#Width = 20
$ColumnText = conv(#NounsReplaced)
ExcelSSO.setColumnWidth(Width)
ExcelSSO.writeExcelCell(Column)
// add a new row and populate columns
ExcelSSO.addNewExcelRow()                     // Add New Row (Row is 1 Now, 0 Index Based)
#Column = 0
$ColumnText = CDATE(@DATE,'MM/DD/YYYY')
ExcelSSO.writeExcelCell(Column)
#Column = 1
$ColumnText = @TIME(1,2) + ':' + @TIME(3,4) + ':' + @TIME(5,6)
ExcelSSO.writeExcelCell(Column)
#Column = 2
$ColumnText = $ReceivedTextREST
//set font for Column 2
$Font = 'A'                                 //Set Font To Arial
#Size = 12
$Bold = 'N'
$Regular = 'N'
$Colour = 'g'                                          //Set Colour To Green
ExcelSSO.setFont(CellFont)
ExcelSSO.writeExcelCell(Column)
//set font for Column 3
$Font = 'A'                                            //Set Font To Arial
#Size = 12
```

Line 02

This line directs to the location you would like the excel sheet to be saved in.

Either alter this line to the location of your choice, or create the directory **C:\Temp**

```
$Bold = 'N'
$Regular = 'N'
$Colour = 'R'                                    //Set Colour To Red
ExcelSSO.setFont(CellFont)
#Column = 3
$ColumnText = $ReturnText
ExcelSSO.writeExcelCell(Column)
#Column = 4
$ColumnText = conv(#NounsReplaced)
ExcelSSO.writeExcelCellAsInteger(Column)
ExcelSSO.writeExcelFile()                       //Write Excel File
ExcelSSO.closeExcelFile()                       //Close Excel File
```

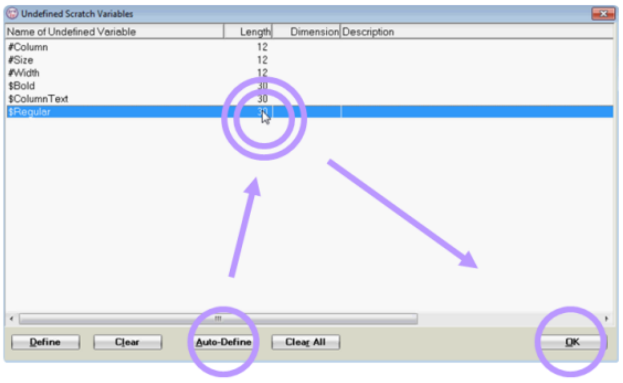
When you are prompted for the Length Variables

Click **Auto-Define**

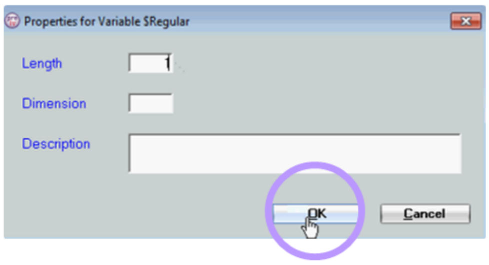
Alter the length of $Regular to **1**
Click **OK**

Alter the length of $Bold to **1**
Click **OK**

Click **OK** to close



Name of Undefined Variable	Length	Dimension	Description
#Column	12		
#Size	12		
#Width	12		
$Bold	30		
$ColumnText	30		
$Regular	30		

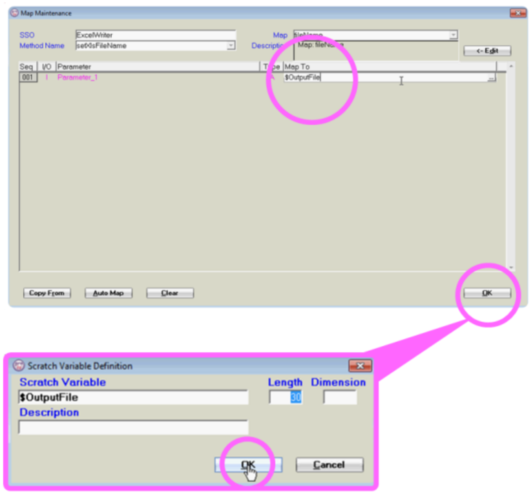


Length: 1
Dimension:
Description:

Adding & Changing Parameters

This is done to format the Excel spreadsheet that PROIV will create

Scroll to the top of the entered rules
Click between the brackets of a parameter.
The first is **(filename)**



Map Maintenance

SSO: ExcelWriter
Method Name: setXlsFileName
Map: fileName
Description: Map: fileName

Seq	I/O	Parameter	Type	Map To
001	I	Parameter_1	A	$OutputFile

Scratch Variable Definition

Scratch Variable: $OutputFile
Length: 30
Dimension:
Description:

Select the Interface Mapping Icon

In ‘Map To’ insert **$OutputFile**

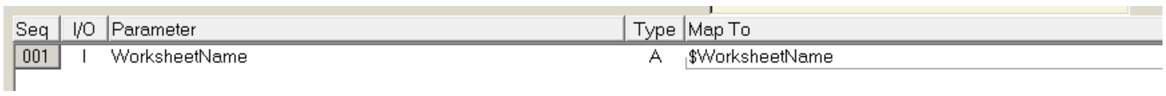
Click **OK**

Click **OK** for length at 30

Click **OK** to close

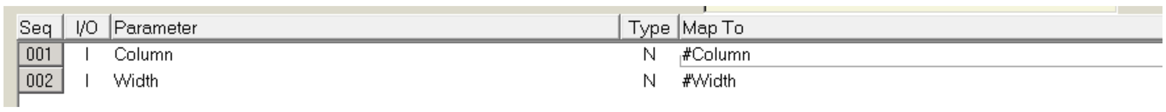
For each unique parameter this need only be performed once

(sheetname)



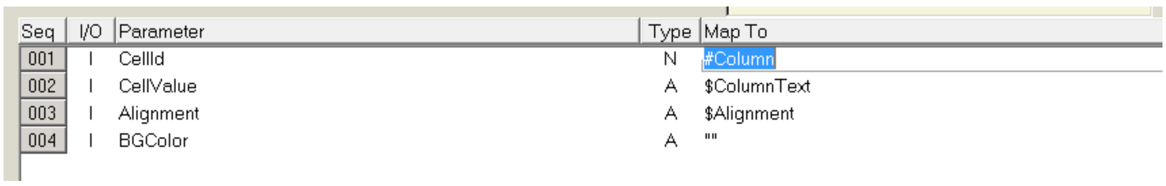
Seq	I/O	Parameter	Type	Map To
001	I	WorksheetName	A	$WorksheetName

(Width)



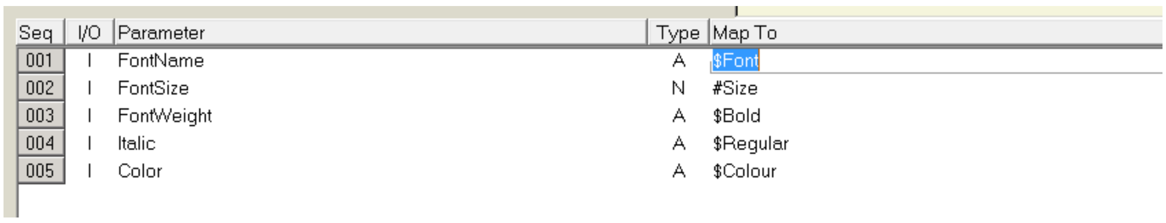
Seq	I/O	Parameter	Type	Map To
001	I	Column	N	#Column
002	I	Width	N	#Width

(Column)



Seq	I/O	Parameter	Type	Map To
001	I	CellId	N	#Column
002	I	CellValue	A	$ColumnText
003	I	Alignment	A	$Alignment
004	I	BGColor	A	""

(Cell Font)



Seq	I/O	Parameter	Type	Map To
001	I	FontName	A	$Font
002	I	FontSize	N	#Size
003	I	FontWeight	A	$Bold
004	I	Italic	A	$Regular
005	I	Color	A	$Colour

When prompted, set **$Font**, and **$Colour** length to 1

Now all of the parameters have been set,

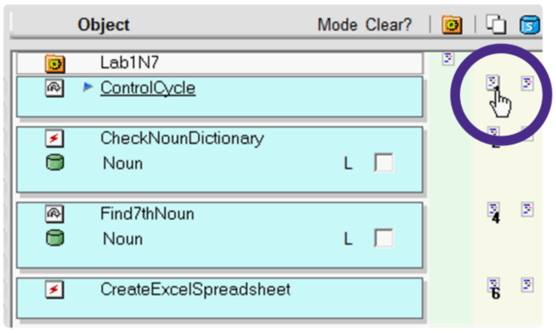
Click **Save**

Create a Read-Only Copy of Data

To store the generator data in excel, we need to make a copy before the input text is changed by the function

In the Events view,

Click on the Entry to the Control Cycle



Amending Logic for SSOExcel

Include the following **bold purple text** above and below your existing rules

```
//Text is received from the global parameter $RecievedText
$pName =  "text"
$ReceivedTextREST = RestfulRequestSSO.getParameter(paramMap)
$RecievedText = $ReceivedTextREST
// initialise temporary variables
#ReceivedTextLength = LEN($ReceivedText)
#startpositon = 0
#nextspace = 0
//Read individual words from the paragraph to find the nouns
for #loop = 1 to #ReceivedTextLength
   #nextspace = index($ReceivedText,' ')
   if #nextspace = 0 then loopexit;
   $word = $ReceivedText(#startpositon,#nextspace -1)
   //Providing a word is found go check if this is a noun
   $NounFound – 'N'
   if($word # '') then
   lscall("CheckNounDictionary")
   endif
   if $NounFound = "Y" then
     lscall("Find7thNoun")
     #NounsReplaced += 1
   endif
   //Build return text by appending the word
   if $ReturnText = "" then
         $ReturnText = $word
   else
         $ReturnText = $ReturnText + " " + $word
   endif
   $ReceivedText = $ReceivedText(#nextspace + 1, #ReceivedTextLength)
Endfor
```

This provides the copy of input text for the spreadsheet

This will count the number of nouns processed

```
Iscall('CreateExcelSpreadsheet')
//Return updated information to using parameter $ReturnText
$pName = "/body/text"
$pValue = $ReturnText
RestfulResponseSSO.setString(paramSet)
$pName = "/header/mimeType"
$pValue = "application/json"
RestfulResponseSSO.setString(paramSet)
```

Before the function exits this cycle, it must call the CreateExcelSpreadsheet

Once you have entered these amendments,

Click **Save**

And, in the top left corner,

Click **Compile**

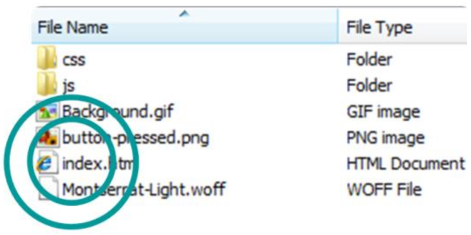
Congratulations!

Now we can test to be sure we have been successful!

As you did at the end of the
second Lab, go to:

>C: Drive
>Program Files (Folder)
>Northgate Arinso (Folder)
>PROIV Version 8 (Folder)
In the Labs folder you created,

Double Click **index.html**



File Name	File Type
css	Folder
js	Folder
Background.gif	GIF image
button-pressed.png	PNG image
index.htm	HTML Document
Montserrat-Light.woff	WOFF File

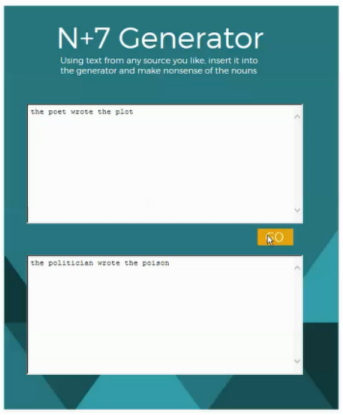
Now your Generator has opened,

Run a phrase through the generator:

The poet wrote the plot

should become

The politician wrote the poison


N+7 Generator
Using text from any source you like, insert it into the generator and make nonsense of the nouns
GO

```
the poet wrote the plot
```

```
the politician wrote the poison
```

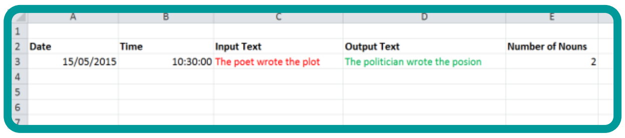
Once you have run this test, an excel file
will be created and saved to your computer

The file will be in the location specified in the rules on page 7

If you left this unaltered, the file will be saved to:

C:\Temp\proivLab.xls

Go and see your results!



	A	B	C	D	E
1					
2	Date	Time	Input Text	Output Text	Number of Nouns
3	15/05/2015	10:30:00	The poet wrote the plot	The politician wrote the posion	2
4					
5					
6					
7					

How did it go?
Tell us how it went via the Homepage contact form, or email **proiv@ngahr.com**