

NaturalONE Tutorial

How to create a WebGUI with NaturalONE? - Working with tree controls -

Recommended Reading

- NaturalONE tutorial: How to set-up a NaturalONE project?
- NaturalONE tutorial: How to create a Web GUI with NaturalONE?
- Creating a new Layout page -

- 1 In this tutorial you will learn to use a tree control.

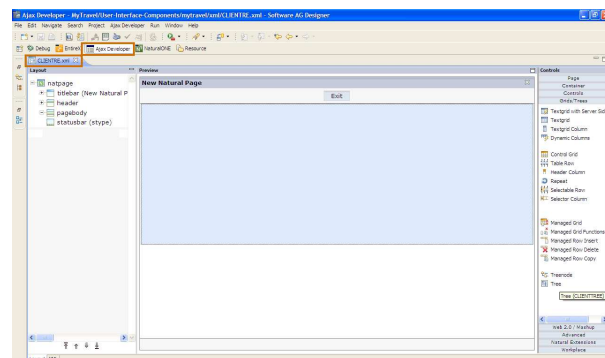
You will build a tree of Natural-Libraries, Natural object types and Natural objects.

The sample data will be defined as literals within the program.

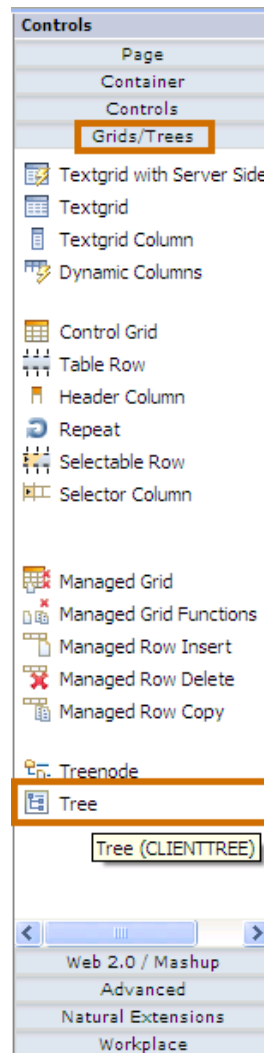
Experienced Natural programmers are encouraged to enrich this sample with real data (e.g. by using USR1054N - List libraries and USR1055N - List objects in a library from SYSEXT).

Open the **Ajax Developer** perspective.

Create a new page layout called: „**CLIENTRE**“.



- 2 Select **Grids/Trees** on the *Controls* palette and select the **Tree** control icon.

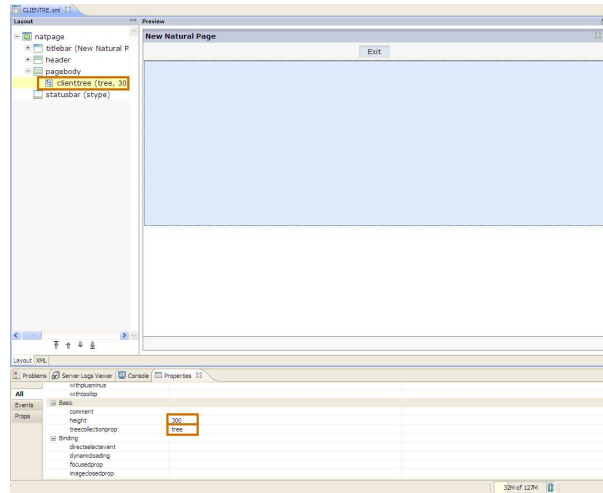


- 3 Drag and drop it onto the Preview pane.

On the *Layout* pane you can see the **CLIENTTREE** control as a sub-node of the **PAGEBODY** node.

Select the **clienttree** control and modify the following properties:

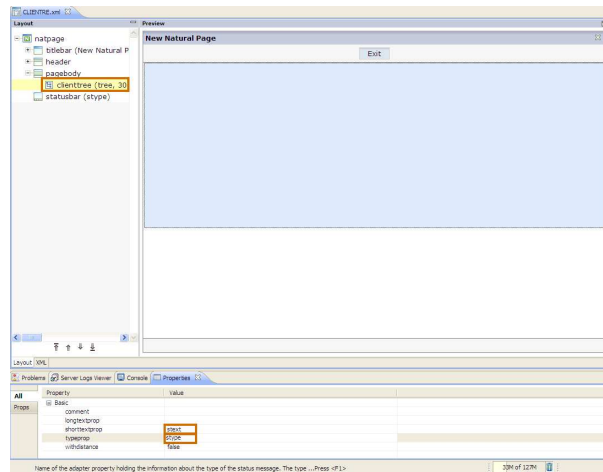
- height - **300**
- treecollectionprop - **tree** (the name of the array that will contain the data)



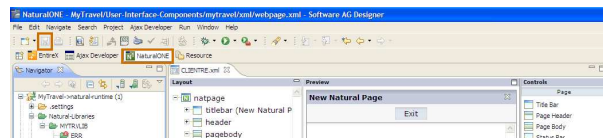
- 4 It's very handy to use the existing statusbar control in order to display some messages.

Select the **statusbar** node and set:

- shorttextprop - **stext** (contains any text to be displayed)
- typeprop - **styp** (can be S-success/W-warning/E-error)

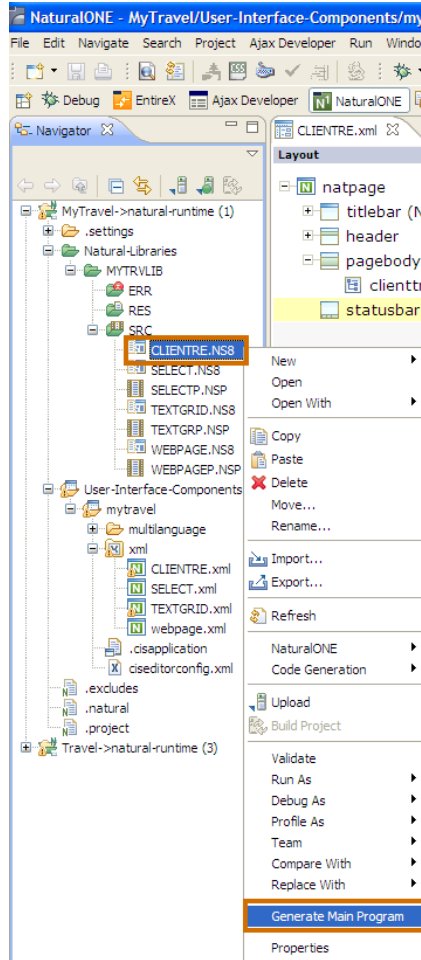


- 5 Save the layout and select the **NaturalONE** perspective.

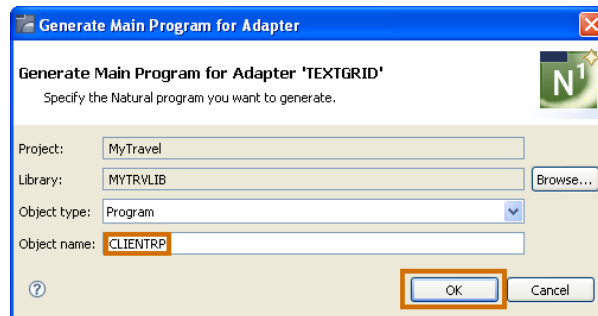


- 6 In the *Navigator* view you can see the newly created adapter **CLIENTRE.NS8**.

Select the adapter.
Choose **Generate Main Program** from the context menu.



- 7 Type „**CLIENTRP**“ as *Object name* and press **OK**.



- 8 You will now add some program logic: Counters, predefined Library names and object types:

After “2 TEXT (A) DYNAMIC” type:

```
1 #I(I2)
1 #J(I2)
1 #K(I2)
1 #L(I2)
```

```
1 #LIB(A/1:3) DYNAMIC INIT
<'TRAVEL', 'TRINBND', 'TROUBND'>
```

```
1 #TYPE(A/1:5) DYNAMIC INIT
<'Parameter Data Areas', 'Global Data Areas', 'Local Data Areas',
'Subprograms', 'Programs'>
```

```

1# * ->Natural Source Reader 0000000
50* ----- GENERATED BY NATURAL FOR AJAX TOOLS
6 * Library .....: MYIRVLIB
7 * Source Name ..: CLIENTRE
8 * -----
90 DEFINE DATA LOCAL
10 1 STXT (A) DYNAMIC
11 1 STYF (A) DYNAMIC
120 1 TREE (I-*)
13 2 LEVEL (I4)
14 2 OPENED (I4)
15 2 SELECTED (L)
16 2 TEXT (A) DYNAMIC
17 **
18 1 #I(I2)
19 1 #J(I2)
20 1 #K(I2)
21 1 #L(I2)
22
23 1 #LIB(A/1:3) DYNAMIC INIT <'TRAVEL', 'TRINBND', 'TROUBND'>
24 1 #TYPE(A/1:5) DYNAMIC INIT <'Parameter Data Areas', 'Global Data Areas',
'Subprograms', 'Programs'>
25
26
27 END-DEFINE
28 EXPAND ARRAY TREE TO (1:100)
29 FOR #I = 1 TO 3 /* 3
30 ADD 1 TO #I
31 TREE.LEVEL (#I) := 1
32 TREE.OPENED (#I) := 1 /* Opened
33 TREE.TEXT (#I) := #LIB(#I)
34 FOR #J = 1 TO 5 /* 5
35 ADD 1 TO #J
36 TREE.LEVEL (#I) := 2
37 TREE.OPENED (#I) := 0 /* Closed
38 TREE.TEXT (#I) := #TYPE(#J)
39 FOR #K = 1 TO 5 /* 5
40 ADD 1 TO #L

```

- 9 Add the following code to initialize the array „TREE“ with appropriate data:

Formatted: English (U.K.)

After “END-DEFINE” type:

```
EXPAND ARRAY TREE TO
(1:100)
FOR #I = 1 TO 3 /* 3
  ADD 1 TO #L
  TREE.LEVEL (#L) := 1
  TREE.OPENED (#L) := 1
/* Opened
  TREE.TEXT (#L) := #LIB(#I)
  FOR #J = 1 TO 5 /* 5
    ADD 1 TO #L
    TREE.LEVEL (#L) := 2
    TREE.OPENED (#L) := 0
/* Closed
    TREE.TEXT (#L) :=
#TYPE(#J)
    FOR #K = 1 TO 5 /* 5
      ADD 1 TO #L
      TREE.LEVEL (#L) := 3
      TREE.OPENED (#L) := 2
/* End node
      COMPRESS 'Object-' #K TO
TREE.TEXT (#L) LEAVING NO
    END-FOR
  END-FOR
END-FOR
```

The screenshot shows a code editor window with the following code:

```
18 1 #I(I2)
19 1 #O(I2)
20 1 #K(I2)
21 1 #L(I2)
22
23 1 #LIB(A/1:5) DYNAMIC INIT <'TRAVEL','TRINBND','TROUBND'>
24 1 #TYPE(A/1:5) DYNAMIC INIT <'Parameter Data Areas','Global Data Areas','Local Data Areas',
25   'Subprograms','Programs'>
26
27 END-DEFINE
28 EXPAND ARRAY TREE TO (1:100)
29 FOR #I = 1 TO 3 /* 3
30   ADD 1 TO #L
31   TREE.LEVEL (#L) := 1
32   TREE.OPENED (#L) := 1 /* Opened
33   TREE.TEXT (#L) := #LIB(#I)
34   FOR #J = 1 TO 5 /* 5
35     ADD 1 TO #L
36     TREE.LEVEL (#L) := 2
37     TREE.OPENED (#L) := 0 /* Closed
38     TREE.TEXT (#L) := #TYPE(#J)
39     FOR #K = 1 TO 5 /* 5
40       ADD 1 TO #L
41       TREE.LEVEL (#L) := 3
42       TREE.OPENED (#L) := 2 /* End node
43       COMPRESS 'Object-' #K TO TREE.TEXT (#L) LEAVING NO
44     END-FOR
45   END-FOR
46 END-FOR
47 END-FOR
48 *
49 PROCESS PAGE USING "CLIENTRE"
50 *
51@ DECIDE ON FIRST *PAGE-EVENT
52@ VALUE U"natipage.end"
53 /* Page closed.
54 *
```

The most relevant properties to be filled are:

TREE.LEVEL - in our case:

- 1 - for Library name,
- 2 - for Object type,
- 3 - for the object itself.

TREE.TEXT - the displayed node label.

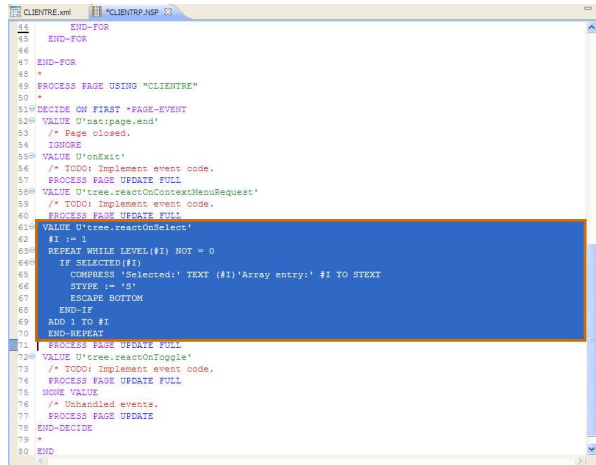
TREE.OPENED - node state which accepts 0- for closed node, 1- opened node and 2- end node (leaf).

- 10 The only event you will handle will be the: **U'tree.reactOnSelect** event.

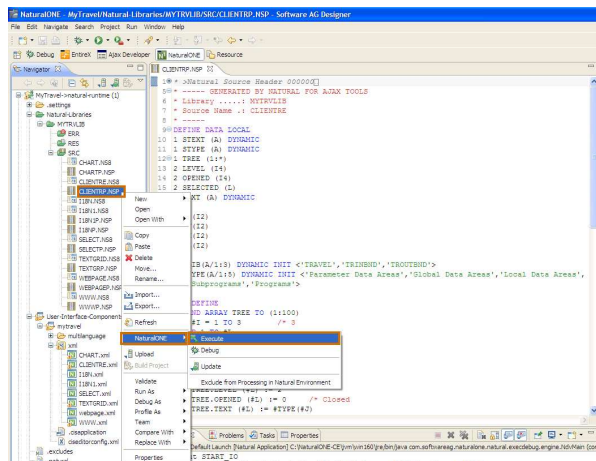
This default event is sent to the server each time a node is selected; when an entry with TREE.SELECTED = TRUE is encountered, certain details will be displayed on the status line.

After “VALUE U'tree.reactOnSelect” type:

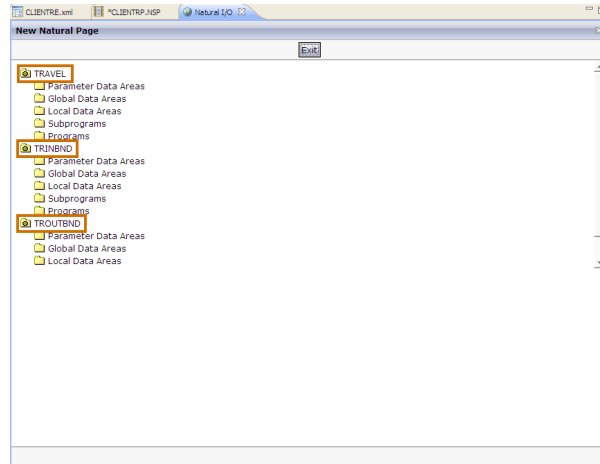
```
VALUE U'tree.reactOnSelect'
  #I := 1
  REPEAT WHILE LEVEL(#I)
NOT = 0
    IF SELECTED(#I)
      COMPRESS 'Selected:'
TEXT (#I)'Array entry:' #I
TO STEXT
      STYPE := 'S'
      ESCAPE BOTTOM
    END-IF
  ADD 1 TO #I
END-REPEAT
```



- 11 Save the program. Execute to check the result.

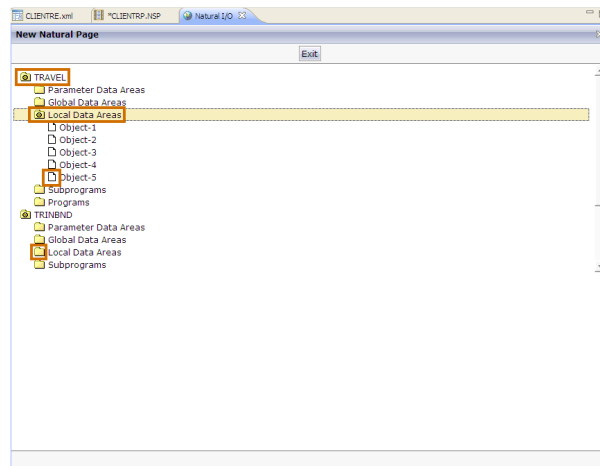


- 12 Notice that the nodes representing the Natural Libraries are expanded. (defined as `TREE.OPENED = 1`) and their subtrees are collapsed.



- 13 Select „**Local Data Area**“ node. Notice that it expands all sub-nodes (objects) which are of type Local Data Area, within the **TRAVEL** Library.

You may also notice the different icons for expanded and collapsed nodes, as well as for leaves (i.e. end nodes).



- 14 Select the leaf „**Object-3**“ and notice that its details are displayed on the status line.

