[Exercise] Create 'Recursive Page Link List' Component

Goals

- To understand the power of FreeMarker macros.
- To understand how recursion may be achieved using a macro, with the example use case of traversing a node hierarchy.

Result of Exercise

- A component is created which:
  - Links to a page over the dialog.
  - Recursively traverses the page hierarchy.
  - Creates a web link to all traversed pages.

Tasks / Procedure

1. **Create** component **definition**: `recursivePageLinkList.yaml`
   Detailed steps

   Operate in the light module *'training-templating-additional'* in the file system.
   a. Navigate to the folder:
   `/training-templating-additional/templates/components`

   b. Create a component **template definition** named: `recursivePageLinkList.yaml` with this content:

```
title: Recursive Page Link List
renderType: freemarker
dialog: training-templating-additional:components/recursivePageLinkList
templateScript: /training-templating-additional/templates/components/recursivePageLinkList.ftl
```

2. **Create** the component **script**: `recursivePageLinkList.ftl`
   Detailed steps

   Operate in the light module *'training-templating-additional'* in the file system.
   a. Navigate to the folder:
   `/training-templating-additional/templates/components`

   b. Create the component **template script** named: `recursivePageLinkList.ftl` with this content:

```
[#if content.targetPageLink?has_content]
    [#assign targetPageNode = cmsfn.contentById(content.targetPageLink, "website")! /]

    [#if targetPageNode?has_content]
    <div>
        <a href="${cmsfn.link(targetPageNode)!}">${targetPageNode.title!targetPageNode.@name}</a>

        [#-- TODO 1
                - use a macro while looping over all the children of the selected node
                - pass to the macro following three things:
                -- the node to get the children from
                -- the depth of the targetPageNode -> startDepths
                -- the maxLevel -> read from the 'content' what the author defined
                - in the macro, for each child, call the macro again -> recursion
                - stop the recursion when the maxDepth is reached   ->
        --]
        [#list cmsfn.children(targetPageNode, "mgnl:page")]
        <ul>
            [#items as childNode]
                <li><a href="${cmsfn.link(childNode)!}">${childNode.title!childNode.@name}</a></li>
            [/#items]
        </ul>
        [/#list]
    </div>

    [#elseif cmsfn.editMode]
        <div>Target page node could not be found via the stored link.</div>
    [/#if]
[#elseif cmsfn.editMode]
    <div>No target page node defined.</div>
[/#if]
```

   ⓘ **Realize:** This code already contains a loop over all the children of the target page. What is left TODO is to create a macro which makes a recursive call during each loop iteration.

3. **Create** the component **dialog**: `recursivePageLinkList.yaml`
   Detailed steps

   Operate in the light module *'training-templating-additional'* in the file system.
   a. Navigate to the folder:
   `/training-templating-additional/dialogs/components`

   b. Create the component **dialog definition** named: `recursivePageLinkList.yaml` with this content:

```
label: Recursive Page Link List

form:
  properties:

    targetPageLink:
      $type: pageLinkField
      label: Page
      description: Choose a page to link to.
      required: true

    maxLevels:
      $type: comboBoxField
      label: Maximum levels
      description: Maximum of level of pages shown in this recursion.
      type: java.lang.Long
      datasource:
        $type: optionListDatasource
        options:
          one:
            value: 1
            label: One level down
          two:
            value: 2
            label: Two levels down
          three:
            value: 3
            label: Three levels down
            selected: true ## Wont work until MGNLUI-4184 is resolved
          four:
            value: 4
            label: Four levels down
          five:
            value: 5
```

```
label: Five levels down
```

> **Realize:** This dialog contains a form field *'targetPageLink'* for linking to a Page. And a second *'select'* field where the author can define how many levels down the recursion should traverse.

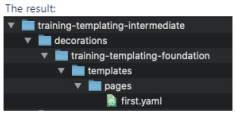
4. **Make** the *'recursivePageLinkList'* component **available** on the *'first'* **page**.
   - Detailed steps

   Operate on the light module 'training-templating-additional' in the file system.
   a. Create this folder structure:
   `/training-templating-additional/decorations/training-templating-freemarker/templates/pages/`

   b. Create the decoration file: `first.yaml`:

```yaml
areas:
  main:
    availableComponents:
      recursivePageLinkList:
        id: training-templating-additional:components/recursivePageLinkList
```

   The result:

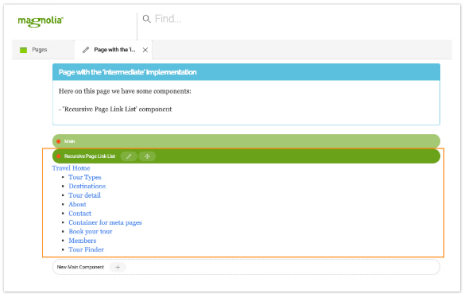


> **Remember:** An area is basically a component container; that's why components get added into areas.

> Forgetting or skipping this step means that the component can't be added anywhere and can't be tested.

5. **Create page** and **test** the behavior.
   - Detailed steps

   Operate in the *'Pages'* app.
   a. Create a new page of type *'first'* and render it.
   b. Add a *'recursivePageLinkList'* component into the page.
   The result:



6. Resolve **TODO 1** in: `recursivePageLinkList.ftl`
   - Detailed steps

   Operate in the light module *'training-templating-additional'* in the file system.
   a. Navigate to the folder page script:
   `/training-templating-additional/templates/components/recursivePageLinkList.ftl`

   b. **Try to resolve** the *'TODO 1'*:
      - Expand for the code

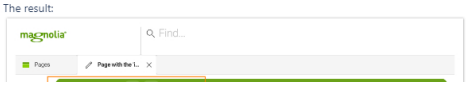
      Read the comment for coding instructions.

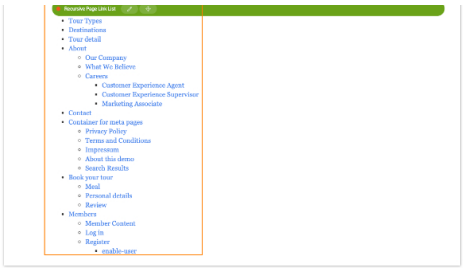
```
[#-- TODO 1
     Do: - delegate the list directive and its children loop to a macro.
         - pass to the macro following three things:
         -- node to get children from
         -- depth of the targetPageNode -> startDepths
         -- maxLevel -> read from the 'content' what the author defined
         - call within the macro on each child the macro itself again -> recursion
         - stop the recursion when the maxDepth is reached   ->
--]
[#list cmsfn.children(targetPageNode, "mgnl:page")]
<ul>
    [#items as childNode]
        <li><a href="${cmsfn.link(childNode)!}">${childNode.title!childNode.@name}</a></li>
    [/#items]
</ul>
[/#list]
```

      - Cheat for the code - BUT first try it your self :)

```
[#assign maxLevels = content.maxLevels!3]
[#assign targetPageNode = cmsfn.contentById(content.targetPageLink, "website")! /]
[#assign targetPageNodeDepth = targetPageNode.@depth!1]


[@linkChildren targetPageNode targetPageNodeDepth maxLevels /]


[#--  The macro getting all the childNode nodes and calling itself recursive. --]
[#macro linkChildren node startNodeDepth maxLevels]
    [#list cmsfn.children(node, "mgnl:page")]
    <ul>
        [#items as childNode]
            <li><a href="${cmsfn.link(childNode)!}">${childNode.title!childNode.@name}</a></li>

            [#assign childNodeDepth = childNode.@depth!2]
            [#assign doRecursion = (childNodeDepth - startNodeDepth) < maxLevels]
            [#if doRecursion!false]
                [#--  The recursive call on the childNode again. --]
                [@linkChildren childNode startNodeDepth maxLevels /]
            [/#if]
        [/#items]
    </ul>
    [/#list]
[/#macro]
```

   c. The result:




Navigation Page Link List
Tour Types
Destinations
Tour detail
About
Our Company
What We Believe
Careers
Customer Experience Agent
Customer Experience Supervisor
Marketing Associate
Contact
Container for meta pages
Privacy Policy
Terms and Conditions
Impressum
About this demo
Search Results
Book your tour
Meal
Personal details
Review
Members
Member Content
Log in
Register
enable-user