

Creazione Nuovo Menu Terzo Livello (leafItem)

In Generale

In Geoweb, un menu di terzo livello, è una particolare tipologia di menu. Più menu di terzo livello (da ora in poi TLM, *Third Level Menu*) sono collegati gerarchicamente ad un unico menu di secondo livello (da ora in poi SLM, *Second Level Menu*).

// TODO ADD IMAGE

In Geoweb il TLM si esplicita tramite il componente **leafItem** (*elemento foglia, di un ipotetico menu ad albero distribuito su 3 livelli*). Questo di norma è rappresentato come un elemento HTML dotato di icona e label (configurabili). Per visualizzare un TLM nel layout di Progetto, bisogna aggiungere un tag *leafItem*, di uno specifico *type*, dentro un tag *accordionPanelItem*.

Esempio:

```
<accordionPaneItem name="slm1" label="Second Level Menu Item" image=""
type="leafItemContainer">
  <leafItem name="t1m1" label="Third Level Menu Item" image=""
type="gwClassList">
  <parameter name="gwClassName" value="class_name"></parameter>
</leafItem>
</accordionPaneItem>
```

In Geoweb sono definiti molteplici tipi di *leafItem*, tra cui.

- *gwClassList*
- *gwPrefilteredList*
- *gwReportsSheet*
- *gwHtmlReport*
- *gwMap*
- *gwItemDetail*
- *leafItemListFilters*
- *leafItemLayoutChange*
- *leafItemClassificationMenu*
- *leafItemHierarchicalFilterMenu*
- *gwAction*
- *gwHtml*
- *leafItemSeparator*.

Altre tipologie di *leafItem* sono definite dai vari *plugin* di Geoweb, ed è previsto un meccanismo per poter facilmente crearne di nuove.

Nuovo LeafItem

Per definire una nuova tipologia di *leafItem* semplicemente popolare l'attributo *type* del tag *leafItem* con un valore univoco. Esempio per un ipotetico *leafItem* di *type gwPluginType*:

```
<accordionPaneItem name="slm1" label="Second Level Menu Item" image=""
type="leafItemContainer">
    <leafItem name="t1m1" label="Third Level Menu Item" image=""
type="gwPluginType">
    </leafItem>
</accordionPaneItem>
```

In risposta al click su un leafItem, di norma viene aperta una specifica scheda (fanno eccezione solo particolari tipi di leafItem, gestiti ad hoc, come gwAction). Nello specifico al click su un leafItem di tipo gwPluginType Geoweb proverà ad aprire una scheda tab che ha:

- per **id** una concatenazione di gwPluginType e del name del leafItem (generata usando la function js createTabId(gwPluginType , name))
- per **titolo** il contenuto dell'attributo label del leafItem
- per **contenuto** la pagina jsp del ModelAndView Spring ritornato dal controller mappato come nell'esempio sotto. I parametri projectName e projectType vengono aggiunti in automatico. Il controller Wrapper non rimanda alla jsp con il contenuto vero e proprio, ma ad un contenitore, specifico per ogni projectType, che a sua volta si occuperà di recuperare il contenuto della scheda tramite wrappedPageAddress (corredato da eventuali parametri di POST), che è mappato da Spring MVC al controller sottostante gwPluginType. Questo modo operandi prevede che ci siano per ogni tipologia di leafItem che apre un tab, un file .jsp wrapper per ogni tipologia di progetto (attualmente GeoManger e GeoExplorer). Ci sono apposite cartelle di tipo progetto sotto jsp/ per ospitare questi wrapper.

```
@RequestMapping(value =
"{projectType}/{projectName}/gwPluginTypeWrapper", method =
RequestMethod.POST, produces = "text/html; charset=utf-8")
@ResponseBody
public ModelAndView gwPluginTypeWrapper(
@PathVariable String projectType,
@PathVariable String projectName,
@RequestBody HashMap<String, Object> hashMap,
Locale locale
){
    ModelAndView modelAndView = null;
    try{
        String projectTypePath = projectType.toLowerCase();
        modelAndView = new ModelAndView(projectTypePath+"/gwPluginTypeWrapper");
        String param1= (String) hashMap.get("param1");
        String wrappedPageAddress =
projectType+"/"+projectName+"/gwPluginType.html";
        modelAndView.addObject("wrappedPageAddress", wrappedPageAddress);
        modelAndView.addObject("param1", param1);
    }catch(Exception e){
        log.error(e.getMessage(), e);
        //throw e;
        //return a message to client
        modelAndView = new
ModelAndView("commons/emptyPageWithDialogErrorMessage");
        String dialogMessage = e.getMessage();
```

```

    dialogMessage = GwUtils.javascriptAndHtmlEscape(dialogMessage);
    modelAndView.addObject("dialogMessage", dialogMessage);
}
return modelAndView;
}
@RequestMapping(value = "{projectType}/{projectName}/gwPluginType",
method = RequestMethod.POST, produces = "text/html; charset=utf-8")
@ResponseBody
public ModelAndView gwPluginType(
@PathVariable String projectType,
@PathVariable String projectName,
@RequestBody HashMap<String, Object> hashMap,
Locale locale
){
...
}

```

Un eventuale plugin di Geoweb che faccia uso di uno specifico type di leafItem, dovrà quindi anche preoccuparsi di esporre un congruo metodo (mappato appropriatamente tramite @RequestMapping) dentro una sua classe controller.

Handler Function

Accade che, in taluni casi, non si voglia aprire un tab al click sul leafItem. Geoweb permette di sovrascrivere questo comportamento. Infatti prima di tentare l'apertura di una scheda, si cerca nel namespace globale window se esiste una function con un name che segue questo pattern:

```

var handlerFunctionName =
'on'+firstLetterUpperCase(tabWidgetType)+'LeafItemClickHandler';

```

Nel nostro caso di esempio tabWidgetType è 'gwPluginType'. Se tale function esiste, essa viene invocata e riceve in ingresso come parametro un oggetto contenente l'intero set di parameter del leafItem. Questa modalità è spesso usata nei plugin di Geoweb, eventualmente definendo la function in qualche risorsa js messa a disposizione dal plugin.

Es: una eventuale function *onGwPluginTypeLeafItemClickHandler(parametersMap){..}* sarebbe mappata ed eseguita per leafItem di tipo *gwPluginType*.

LeafItem jsp Path

Volendo intervenire ancora più a monte è possibile bypassare questo meccanismo impostando una specifica .jsp che può essere usata per determinare il come verrà mostrato il LeafItem. Esiste una .jsp di default (.../leafItemSimple.jsp) che viene usata in automatico (e lavora come descritto sopra). Se necessario se ne può utilizzare una specializzata per rispondere in modo customizzato all'interazione utente. La .jsp per il leafItem si imposta così (nel gwRegistry o nei vari *InitializerService.class dei plugin).

```

gwRegistry.setLeafItemJspPath("gwPluginType", "/jsp/commons/pathToSpecific/file.jsp");

```

Di default varrebbe così:

```
gwRegistry.setLeafItemJspPath("gwPluginType",  
"/jsp/commons/accordionPaneBar/accordionPane/leafItem/leafItemSimple.jsp")
```

Naturalmente la jsp mappata deve essere resa disponibile nel plugin o in qualche altra parte del framework.

From:
<https://wiki.geowebframework.com/> - **GeowebFramework**

Permanent link:
https://wiki.geowebframework.com/doku.php?id=custom:development_creazione_nuovo_menu_terzolivello_leafitem&rev=1678440682

Last update: **2023/03/10 10:31**

