

## Progettazione

- [Struttura](#)
  - [Guida](#)
  - [Codice nel Model](#)
  - [Codici in menLib](#)
    - [Funzione GetMenu](#)
    - [Funzine caricaTreeLegami](#)
- 

### Struttura

Come abbiamo già visto il treeGridModel utilizzato è quello Adjacency.

E' importante definire, nella parte di programmazione, che comportamento dovranno assumere i vari campi.

Le possibili configurazioni sono:

- **Level** : Che assume valori integer, e indica il livello di profondità
- **Parent** : Indica ,nell'array, l'indice del padre. Quale livello è padre. (Assume anche esso valori integer)
- **isLeaf** :
  - **true**: Indica se è o meno una foglia. Nota. La foglia è l'ultimo elemento dell'albero, quindi non è chiaramente espandibile.
  - **false**: Se impostato indica che l'elemento non è una foglia.
- **loaded** :
  - **true**: Se impostato appena viene caricata la grid, vengono anche visualizzate le foglie.
  - **false**: Se impostato appena viene caricata la grid, non verranno visualizzate le varie foglie del parent, quindi sarà necessario un click per visualizzare le varie foglie.
- **expanded** :
  - **true**: Se impostato si avrà l'effetto grid ad albero.
  - **false**: Se impostato false , la tabella non visualizzerà nulla, perchè ,come definito da programma utilizzando l'Adjacency Model, il programma si aspetta un effetto ad "albero".

#### Nota.

Se si vuole ottenere un effetto ad albero, dove vengono visualizzate automaticamente i nodi e le loro foglie, è necessario impostare sia **Expanded** che **Loaded** in "True". Se almeno uno dei 2 risulterà falso l'effetto sarà lo stesso: verrà visualizzato solo il nodo principale.

## Guida

### Model

## Esempio:

Nel Programma andremo a chiamare la funzione **getMenu** dalla libreria **menLib**.

Il risultato chiaramente poi finirà nell'array per costruire la Grid.

```
.....
$this->tree = $this->menLib->getMenu($voceMenu, $only_menu = false,
$gruppo, $return_model = 'adjacency', $filtro = false);
$arr = array('arrayTable' => $this->tree,
            'rowIndex' => 'idx');

$griglia = new TableView($this->tableId, $arr);
$griglia->setPageNum(1);
$griglia->setPageRows('1000');
.....
```

---

## Libreria: menLib

Nella libreria menLib troveremo chiaramente molte funzioni, ma ce ne interessano 2 in particolare:

- [Funzione getMenu](#)
- [Funzione caricaTreeLegami](#)

## Funzione getMenu

```
public function getMenu($root = 'TI_MEN', $only_menu = false, $gruppo = '',
$return_model = 'adjacency', $filtro = true) {
    $inc = 0;
    $albero = array();
    $albero[$inc]['INDICE'] = $inc;
```

Nella funzione getMenu possiamo notare che:

**\$inc** viene settato a **0**

Viene dichiarato l'array **\$albero** e gli viene attribuito il valore **\$inc** nel campo **'INDICE'** di **\$inc**.

In questo caso avremo nell'array:

- [0]
  - ↳ INDICE = 0

Successivamente dovremo impostare alcuni campi obbligatori:

**'level','parent','isLeaf','expanded','loaded'**

```

.....
    $albero[$inc]['INDICE'] = $inc;
    $albero[$inc]['level'] = 0; //Imposta il livello 0
    $albero[$inc]['parent'] = NULL;           // Imposta il parent
nullo perchè è il primo nodo
    $albero[$inc]['isLeaf'] = 'false';       // Non è una foglia,
essendo il primo nodo che avremo
    $albero[$inc]['expanded'] = 'true';     //Rende l'albero
espandibile
    $albero[$inc]['loaded'] = 'true';       // Carica le foglie
insieme all'albero

```

Dopo di che possono essere impostati anche gli altri campi all'interno dell'array :

```

.....
    $albero[$inc]['loaded'] = 'true';       // Carica le foglie
insieme all'albero
    $albero[$inc]['pm_voce'] = $root;
    $albero[$inc]['me_id'] = $chiave;
    $albero[$inc]['pm_id'] = $pm_id;
    $albero[$inc]['pm_descrizione'] = $pm_descrizione;
    $albero[$inc]['pm_sequenza'] = 0;
    $save_count = count($albero);

```

Alla fine dell'impostazione dei campi, possiamo notare **“\$save\_count = count(\$albero);”**.

La variabile **\$save\_count** viene quindi settata con il conteggio totale delle righe in **\$albero**, che servirà più avanti come controllo.

A questo punto ci servirà richiamare un'altra funzione, per definire i vari legami tra i livelli; la funzione **creaTreeLegami** contenuta sempre nella libreria menLib.

```

    $albero = $this->caricaTreeLegami($chiave, $albero, 1, $inc,
$only_menu, $filtro);
    if ($save_count == count($albero)) {
        $albero[$inc]['isLeaf'] = 'true';
    }
    return $albero;
}

```

In questo esempio possiamo notare che nell'assegnazione dei valori della funzione **caricaTreeLegami**:

- al campo **\$level** assegna il valore 1 (1 perchè in questo esempio è il livello successivo al primo parent (0), ma è possibile inserire anche un autoincrementazione )
- al campo **\$parent** assegna il valore **\$inc**, (\$inc perchè vi è contenuto il valore del parent iniziale, che in questo caso è 0)

La condizione ci serve per capire se abbiamo un risultato dalla funzione. Infatti potrebbe essere che non ci sia nessun sottolivello e che quindi il primo livello (0 in questo caso) sia solo una foglia, viene quindi impostato in `isLeaf = True`.

## Funzine caricaTreeLegami

La funzione `caricaTreeLegami` è una funzione ricorsiva che:

1. ispeziona un gruppo di dati,
2. scorre la tabella estratta,
3. si ripete se trova altri sottolivelli.

In breve stabilisce quali record nell'array sono dei "Rami" e quali le "foglie".

```
public function caricaTreeLegami($chiave, $albero, $level, $indice,
$only_menu = false, $filtro = true) {
    if ($level == 10) { // Impostato a 10 perchè è
praticamente impossibile che si arrivi ad avere più di 10 livelli.
        return $albero;
    }

    $sql = "SELECT * FROM ita_puntimenu WHERE me_id = '" . $chiave . "'
ORDER BY pm_sequenza";
    $Ita_puntimenu_tab = ItaDB::DBSQLSelect($this->ITALSOFT_DB, $sql,
true);
    if ($Ita_puntimenu_tab) {
        foreach ($Ita_puntimenu_tab as $i => $Ita_puntimenu_rec) {
            if ($only_menu && $Ita_puntimenu_rec['pm_categoria'] !=
'ME') {
                continue;
            }
        }
    }
}
```

In questa prima parte di codice, possiamo notare un primo controllo: **if (\$level == 10)**, per evitare che la ricorrenza avvenga all'infinito.

In questo esempio si vedono chiaramente anche una **SELECT** e un `foreach`.

All'interno di quest'ultimo è possibile notare una `if`, dove viene controllato se il campo **\$only\_menu** e il campo **'pm\_categoria'** nel record **ita\_puntimenu\_rec** non sono di tipo **ME** (Menu).

Nell'esempio questo controllo è utilizzato per capire se il record che sta passando è o meno una Foglia.

All'interno del **foreach** continueremo con l'impostazione dei campi obbligatori, per quanto riguarda l'impostazione delle foglie:

```
.....
        continue;
```

```

    }
    $inc = count($albero);
    $albero[$inc] = $Ita_puntimenu_rec;
    $albero[$inc]['INDICE'] = $inc;
    $albero[$inc]['level'] = $level;
    $albero[$inc]['parent'] = $indice;
    $albero[$inc]['expanded'] = 'false';
    $albero[$inc]['loaded'] = 'false';
    $albero[$inc]['isLeaf'] = 'true';

```

Possiamo notare che la variabile **\$inc** viene settata con il conteggio totale dei record in **\$albero**.

Successivamente, sempre all'interno del **foreach**, verrà controllato se il record che sta passando è o meno un menu :

```

        .....
        $albero[$inc]['isLeaf'] = 'true';

        if ($Ita_puntimenu_rec['pm_categoria'] == 'ME') {
            $albero[$inc]['isLeaf'] = 'false';
            $sql = "SELECT * FROM ita_menu WHERE me_menu = '" .
$Ita_puntimenu_rec['pm_voce'] . "'";
            $Ita_menu_giu_rec =
ItaDB::DBSQLSelect($this->ITALSOFT_DB, $sql, false);
            $me_id = $Ita_menu_giu_rec['me_id'];

```

Nel caso in cui si tratti di un **Menu**, possiamo subito notare l'impostazione del campo obbligatorio **'isLeaf'** definita **false**, in quanto si tratta di un Nodo e non una foglia.

E' inoltre presente una **Select**, che assegnerà alla variabile **\$me\_id** il valore **'me\_id'** risultante dalla query.

Dopo di che verrà salvato il conteggio totale in **\$save\_count** dei record contenuti in **\$albero** e la funzione avrà un "autorichiamo":

```

        ....
        $me_id = $Ita_menu_giu_rec['me_id'];
        $save_count = count($albero);

        $albero = $this->caricaTreeLegami($me_id, $albero, $level +
1, $inc, $only_menu, $filtro);
        if ($save_count == count($albero)) {
            $albero[$inc]['isLeaf'] = 'true';
        }
    }
}
}
return $albero;
}

```

Possiamo notare che nell'assegnazione della funzione ricorsiva i valori **\$chiave**, **\$level** e **\$parent** subiscono delle variazioni:

- alla variabile **\$chiave** viene assegnato il valore **\$me\_id**
- la variabile **\$level** viene incrementata di **1**
- e alla variabile **\$parent** viene assegnato il valore contenuto in **\$inc**

Viene inoltre riutilizzata la condizione **if (\$save\_count == count(\$albero))** che come prima ha il compito di controllare se il risultato è differente( quindi nuovo livello) o non è variato (quindi foglia).

Infine vi è il ritorno di \$albero : **return \$albero;**

Per concludere:

```
.....
$this->tree = $this->menLib->getMenu($voceMenu, $only_menu = false,
$gruppo, $return_model = 'adjacency', $filtro = false);
$arr = array('arrayTable' => $this->tree,
            'rowIndex' => 'idx');

$griglia = new TableView($this->tableId, $arr);
$griglia->setPageNum(1);
$griglia->setPageRows('1000');
.....
```

**tree** a questo punto contiene tutto il nostro albero.

Come è già stato detto in precedenza nel Generetor il nostro elemento è stato definito treeGrid, e attraverso la selezione del 'arrayTable' e la funzione TableView , il nostro albero verrà inserito nella grid.

From:

<https://wiki.nuvolaitalsoft.it/> - **wiki**

Permanent link:

<https://wiki.nuvolaitalsoft.it/doku.php?id=sviluppo:proggridalbero&rev=1351259219>

Last update: **2018/03/19 10:45**

