

Componenti dinamici

All'interno della libreria **cwbLibHtml** sono presenti dei metodi per la generazione di componenti dinamici a run-time.

Pulsanti dinamici nella buttonbar

Metodo: **pulsantiDinamiciButtonBar**(\$formName, \$divName, \$pulsanti)

Parametri:

- **\$formName**: nome della form
- **\$divName**: nome del div dove iniettare i componenti
- **\$pulsanti**: array di dati che rappresenta i pulsanti da aggiungere.

Ogni pulsante è rappresentato da un array associativo, con le seguenti chiavi:

- **id**: Id pulsante
- **icon**: icona pulsante (per vedere le icone utilizzabili, vedere il seguente link: <https://api.jqueryui.com/theming/icons/>)
- **newline**: indica se dopo il pulsante occorre andare a capo riga
- **properties**: array di proprietà da associare al pulsante

Esempio di utilizzo:

```
cwbLibHtml::pulsantiDinamiciButtonBar($this->nameForm,
'divPulsantiDinamici', array(
    array(
        'id' => 'btnDinamico1',
        'icon' => 'ui-icon-search',
        'newline' => 1,
        'properties' => array(
            'style' => 'width:140px;',
            'value' => 'Pulsante 1',
        )
    ),
    array(
        'id' => 'btnDinamico2',
        'icon' => 'ui-icon-search',
        'newline' => 1,
        'properties' => array(
            'style' => 'width:140px;',
            'value' => 'Pulsante 2',
        )
    ),
));
```

Componenti dinamici Su Grid

Metodo: **addGridComponent**(\$formName, \$components)

Parametri:

- **\$formName**: nome della form
- **\$components**: array di dati che rappresenta i componenti da aggiungere

Componenti attualmente gestiti:

ita-edit

L'array \$components deve contenere:

- **type**: 'ita-edit'
- **id**: id componente
- **onChangeEvent**: true se si vuole abilitare l'evento onChange, false altrimenti (evento 'afterSaveCell')
- **model**: il nome della form su cui deve rientrare (solo se richiesta gestione eventi 'onChangeEvent')
- **rowKey**: identificativo riga selezionata in griglia
- **additionalClass**: da passare se si vogliono agganciare delle classi custom al componente
- **properties**: array di proprietà da associare al componente(es. array('value' ⇒ 'valore') per settare il value del componente)

ita-edit-date

L'array \$components deve contenere:

- **type**: 'ita-edit-date'
- **id**: id componente
- **onChangeEvent**: true se si vuole abilitare l'evento onChange, false altrimenti (evento 'afterSaveCell')
- **model**: il nome della form su cui deve rientrare (solo se richiesta gestione eventi 'onChangeEvent')
- **rowKey**: identificativo riga selezionata in griglia
- **additionalClass**: da passare se si vogliono agganciare delle classi custom al componente
- **properties**: array di proprietà da associare al componente(es. array('value' ⇒ 'valore') per settare il value del componente)
- **formatter**: formatter da usare per ottenere il valore sulla post (se non passato di default viene messo quello standard, passare 1 solo valore).

ita-select

L'array `$components` deve contenere:

- **type**: 'ita-edit-date'
- **id**: id componente
- **onChangeEvent**: true se si vuole abilitare l'evento `onChange`, false altrimenti (evento 'afterSaveCell')
- **model**: il nome della form su cui deve rientrare (solo se richiesta gestione eventi 'onChangeEvent')
- **rowKey**: identificativo riga selezionata in griglia
- **additionalClass**: da passare se si vogliono agganciare delle classi custom al componente
- **properties**: array di proprietà da associare al componente(es. array('value' ⇒ 'valore') per settare il value del componente)
- **options**: array di opzioni della select contenente: id,value,selected,text (opzionale, se non c'è viene messo value)

Componenti dinamici Su Pagine

Metodo: **componentiDinamici**(\$formName, \$divName, \$components)

Parametri:

- **\$formName**: nome della form
- **\$divName**: nome del div dove iniettare i componenti
- **\$components**: array di dati che rappresenta i componenti da aggiungere

Componenti attualmente gestiti:

div

E' un componente che contiene al suo interno altri componenti.

- **type**: 'div'
- **id**: id componente
- **children**: array di sottocomponenti

ita-button

- **type**: 'ita-button'
- **id**: id componente
- **icon**: icona pulsante (per vedere le icone utilizzabili, vedere il seguente link: <https://api.jqueryui.com/theming/icons/>)
- **newline**: indica se dopo il pulsante occorre andare a capo riga
- **properties**: array di proprietà da associare al pulsante

ita-edit

- **type**: 'ita-edit'

- **id**: id componente
- **newline**: indica se dopo il pulsante occorre andare a capo riga
- **properties**: array di proprietà da associare al pulsante
- **label**: oggetto label associato al componente
- **additionalClass**: da passare se si vogliono agganciare delle classi custom al componente
- **onChangeEvent**: true se si vuole abilitare l'evento onChange, false altrimenti (evento 'afterSaveCell')
- **model**: il nome della form su cui deve rientrare (solo se richiesta gestione eventi 'onChangeEvent')

ita-edit-lookup

- **type**: 'ita-edit-lookup'
- **id**: id componente
- **newline**: indica se dopo il pulsante occorre andare a capo riga
- **properties**: array di proprietà da associare al pulsante
- **label**: oggetto label associato al componente

ita-readonly

- **type**: 'ita-readonly'
- **id**: id componente
- **newline**: indica se dopo il pulsante occorre andare a capo riga
- **properties**: array di proprietà da associare al pulsante
- **label**: oggetto label associato al componente

ita-checkbox

- **type**: 'ita-checkbox'
- **id**: id componente
- **newline**: indica se dopo il pulsante occorre andare a capo riga
- **properties**: array di proprietà da associare al pulsante
- **label**: oggetto label associato al componente

ita-select

- **type**: 'ita-select'
- **id**: id componente
- **newline**: indica se dopo il pulsante occorre andare a capo riga
- **properties**: array di proprietà da associare al pulsante
- **label**: oggetto label associato al componente
- **options**: array di valori che può assumere il componente
- **onChangeEvent**: true se si vuole abilitare l'evento onChange, false altrimenti (evento 'afterSaveCell')
- **model**: il nome della form su cui deve rientrare (solo se richiesta gestione eventi 'onChangeEvent')

- **additionalClass**: da passare se si vogliono agganciare delle classi custom al componente

label

- **text**: testo della label
- **position**: posizione label ('sx', 'dx')
- **style**: stile css

ita-select option

- **id**: id elemento
- **value**: Valore da mostrare a video
- **selected**: se indicato, e se valorizzato a 1, indica si assume come valore di default

Esempio di utilizzo:

```
private function componentiDinamici() {
    $componenti = array(
        array(
            'type' => 'div',
            'id' => 'divInfoAggiuntive',
            'children' => array(
                array(
                    'type' => 'ita-edit',
                    'id' => 'txtDemo1',
                    'newline' => 1,
                    'properties' => array(
                        'maxlength' => 10,
                        'size' => 7,
                        'style' => 'text-align:right;'
                    ),
                    'label' => array(
                        'text' => 'label comp. 1',
                        'position' => 'sx',
                        'style' => 'width:120px;'
                    )
                ),
                array(
                    'type' => 'ita-edit',
                    'id' => 'txtDemo2',
                    'newline' => 1,
                    'properties' => array(
                        'maxlength' => 20,
                        'size' => 14,
                        'style' => 'text-align:left;'
                    ),
                    'label' => array(
                        'text' => 'label comp. 2',
                        'position' => 'sx',
```

```
        'style' => 'width:120px;'
    )
),
array(
    'type' => 'ita-edit-lookup',
    'id' => 'txtLookup1',
    'newline' => 0,
    'size' => 200,
    'properties' => array(
        'maxlength' => 2,
        'size' => 5,
        'style' => 'text-align:right;'
    ),
    'label' => array(
        'text' => 'label lookup',
        'position' => 'sx',
        'style' => 'width:120px;'
    )
),
array(
    'type' => 'ita-readonly',
    'id' => 'txtLookup1_decod',
    'newline' => 1,
    'properties' => array(
        'size' => 50
    )
),
array(
    'type' => 'ita-checkbox',
    'id' => 'txtCheckTest',
    'newline' => 1,
    'label' => array(
        'text' => 'label check',
        'position' => 'sx',
        'style' => 'width:120px;'
    )
),
array(
    'type' => 'ita-select',
    'id' => 'txtSelect1',
    'newline' => 1,
    'label' => array(
        'text' => 'label select',
        'position' => 'sx',
        'style' => 'width:120px;'
    ),
    'options' => array(
        array(
            'id' => '01',
```

```

        'value' => 'EUROPA',
        'selected' => 1
    ),
    array(
        'id' => '02',
        'value' => 'ASIA'
    )
)
),
array(
    'type' => 'ita-button',
    'id' => 'btnSub1',
    'icon' => 'ui-icon-search',
    'newline' => 1,
    'properties' => array(
        'style' => 'width:140px;',
        'value' => 'Pulsante 1',
    )
),
)
);
    cwLibHtml::componentiDinamici($this->nameForm, "divCampiDinamici",
$componenti);
}

```

Aggiunta di una subform

Per includere dinamicamente un model (disegnato con il generator) all'interno di un div, utilizzare il metodo: **includiFinestra**(\$src, \$dest, \$containerName)

Parametri:

- **src**: Nome finestra da includere
- **dest**: Nome finestra di destinazione
- **containerName**: Nome container dove iniettare la finestra

In questo caso la finestra inclusa sarà vista come un'estensione della finestra ospitante, e tutti gli eventi della finestra inclusa saranno gestiti da quest'ultima.

Esempio:

```

cwLibHtml::includiFinestra('cwBtaNazion', $this->nameForm,
'divContainer');

```

Aggiunta di componenti all'interno di una jqGrid

Per aggiungere dei componenti all'interno di una jqGrid, devono verificarsi innanzitutto le seguenti

condizioni:

- I dati devono essere presenti all'interno del model (array)
- I dati devono essere caricati tutti in memoria
- I dati devono essere serializzati in sessione

Il funzionamento è il seguente:

- Viene effettuato il caricamento dei dati in un array, poi serializzato in sessione
- I dati vengono elaborati: viene prodotto l'html per il render dei componenti, a seconda del tipo di dato da gestire
- Gestione degli eventi: i dati in memoria devono essere sincronizzati

Metodo: **addGridComponent**

Parametri:

- **component**: Array che rappresenta i dati del componente

Questo metodo funge da wrapper per i componenti specifici, ad esempio, per 'ita-select', viene chiamato il seguente metodo: **addGridComponentItaSelect**

Parametri:

- **type**: Tipo componente ('ita-select')
- **model**: Nome della form (corrisponde con `$this->nameForm`)
- **name**: Nome del componente (serve per costruire l'id in questo modo: `[model]_[name]_[rowKey]`)
- **rowKey**: Chiave che identifica il record
- **options**: Array dei valori con cui popolare il componente 'ita-select'
- **additionalData**: Array di valori da restituire nel gestore degli eventi

All'evento 'change', vengono restituiti i seguenti valori nella `$_POST`:

- **id**: Id del componente html
- **name**: Nome componente (id, senza rowKey)
- **model**: Nome della form
- **value**: Nuovo valore

Inoltre, verranno passati tutti i valori previsti in 'additionalData' (direttamente nella `$_POST`).

Esempio di utilizzo (all'interno di un metodo di elaborazione del record):

```
private function elaboraRecordsTest($Result_tab) {
    $numRiga = 0;
    foreach ($Result_tab as $key => $Result_rec) {
        $numRiga++;
        $html = cwLibHtml::addGridComponent(array(
            'type' => 'ita-select',
            'model' => $this->nameForm,
            'name' => 'selectIndirizzo',
```



```
'rowKey' => $Result_tab[$key]['ID'],
'options' => array(
    array(
        'id' => 1,
        'value' => 'vai milano 1, Jesi(AN)'
    ),
    array(
        'id' => 2,
        'value' => 'vai ascoli piceno 2, Apiro(MC)'
    )
),
'additionalData' => array(
    'numRiga' => $numRiga
)
));
$Result_tab[$key]['INDIRIZZO'] = $html;
}
return $Result_tab;
}
```

Note

E' necessario dopo aver creato a runtime dei componenti particolari (date-picker ecc) associare tramite la funzione "parseHtmlContainer" lo script js per far funzionare il componente. Ad esempio se creo a runtime una griglia editabile è necessario per far funzionare tutti i suoi elementi (date-picker, formater del data-edit ecc) lanciare la function sul componente padre che si occuperà di propagare tale function ai figli

```
Out::codice("parseHtmlContainer($('##" . $this->nameForm . '_' .
[NOMEGRIGLIA] . "'))");");
```

From:

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

Permanent link:

https://wiki.nuvolaitalsoft.it/doku.php?id=sviluppo:cityware_componenti_dinamici&rev=1505383847

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

