Grid Elements

Extension Key: **gridelements**

Copyright 2011-2012

Jo Hasenau

info@cybercraft.de

Rocco Georgi

rocco@pavingways.com

Table of Contents

Grid Elements introduction

What does it do?

Grid View

Since version 4.5 the TYPO3 core offers the so called **grid view**, a feature developed during the user experience week, that gives backend users some nice options to get a more **user friendly backend layout**. You can create your own table based backend layout records, fill in as many columns as you like with either a wizard or a *TSconfig* like code and arrange these columns to match your desired layout, so backend users will easily recognize where to put their content. Each record can get an icon that will be used as with the layout selector box.

Grid Elements are pushing these features to the next level, namely content elements.

You will get pretty much the same backend layout records, again created with a wizard or by hand. By assigning such a layout to a Grid Element, you can enable a table based structure for this element, which is becoming a container this way. This container is offering different **cells for your content elements**, which can of course be Grid Elements as well. Setting up **nested structures is a breeze** this way. Each record can get a second icon that will be used for the detailed description within the new content element wizard.

Additionally CE backend layouts can contain a **flexform** to add lots of different features, like form based select boxes and switches to control the frontend output of a grid elements based on this layout later on.

Another usability improvement of Grid Elements is the new **drag and drop behavior** added to the page module. You can drag elements between different columns within the page or element grid. Drop an element to move it or make a copy by pressing the CTRL-key while dropping. You can drag in new content elements from a new content element wizard overlay, that can be activated by the add new content element button on top of the page module. You can create references to content elements on the same or another page with icons appearing on top of each column as soon as an element is available from the normal clipboard. And of course you can have the so called *unused elements* as well, by simply adding a column with colPos -2 to your page grid.

A short roundup of the features and advantages

- Completely TypoScript based backend layout
- Comfortable point and click wizard to create backend layout structures
- Completely XML- and CSV-less normalized relations between containers and elements
- Flexforms used for configurational stuff only, can be derived from existing data structures
- Original colPos and sorting fields still working
- Top level layouts to exclude certain types of Grid Elements from being used within other Grid Elements
- Drag & drop move and copy actions for the page module
- New content element wizard overlay to drag in new content elements
- Paste icons for pasting copies and references into grid columns
- Completely TypoScript based frontend output
- Flexform field values automatically added to the data set
- No need for HTML templates to get a backend layout and/or frontend output
- Completely based on hooks without XCLASSing

Some of you might be used to similar features of TemplaVoila and ask themselves why they should be using grid elements instead. If you want to know more details, check out the **FAQ** section to find some answers

How to use

Step by step:

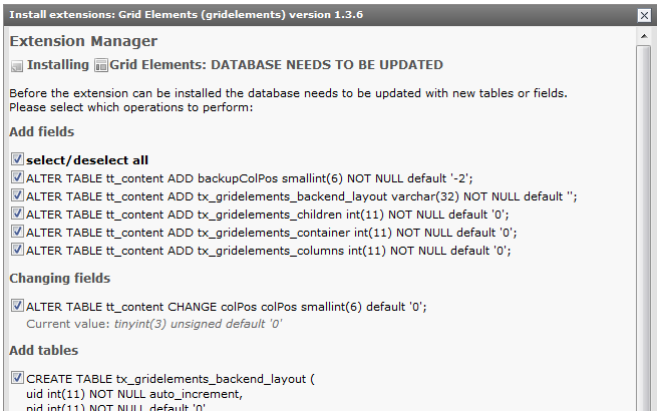
Download or import the Grid Elements extension

First you should download the extension either as a T3X file, that you can import with the Extension Manager, or directly from the TER by using the repository import features of the Extension Manager.

Install the Grid Elements extension

After download and/or import have been completed, you have to install Grid Elements with the Extension Manager to make use of it. There are no dependencies, so you should be able to do so without problems. The only extension, that will have conflicts, is TemplaVoila, since it uses some of the hooks used by Grid Elements as well. You can still install Grid Elements by ignoring the warning of the Extension Manager though, so advanced users might have both extension active while migrating from one to the other. But TemplaVoila features will be disabled then.

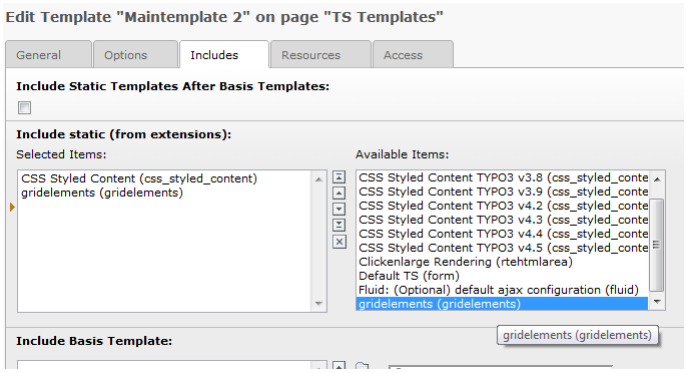
When the Extension Manager is asking you for permission to add and/or modify some of the database tables, you should give your OK, so the necessary fields will be available.



Important Note: The colPos field of the tt_content table will be changed from unsigned int(3) to smallint(6) to enable the usage of negative values. This should **never be reverted** by any upgrade script later on! Otherwise any child element will be moved from it's parent container to the default page column. Some people ran into problems during automatic upgrades done by their providers. So be sure to make a backup of your content before upgrading!

Include static template(s)

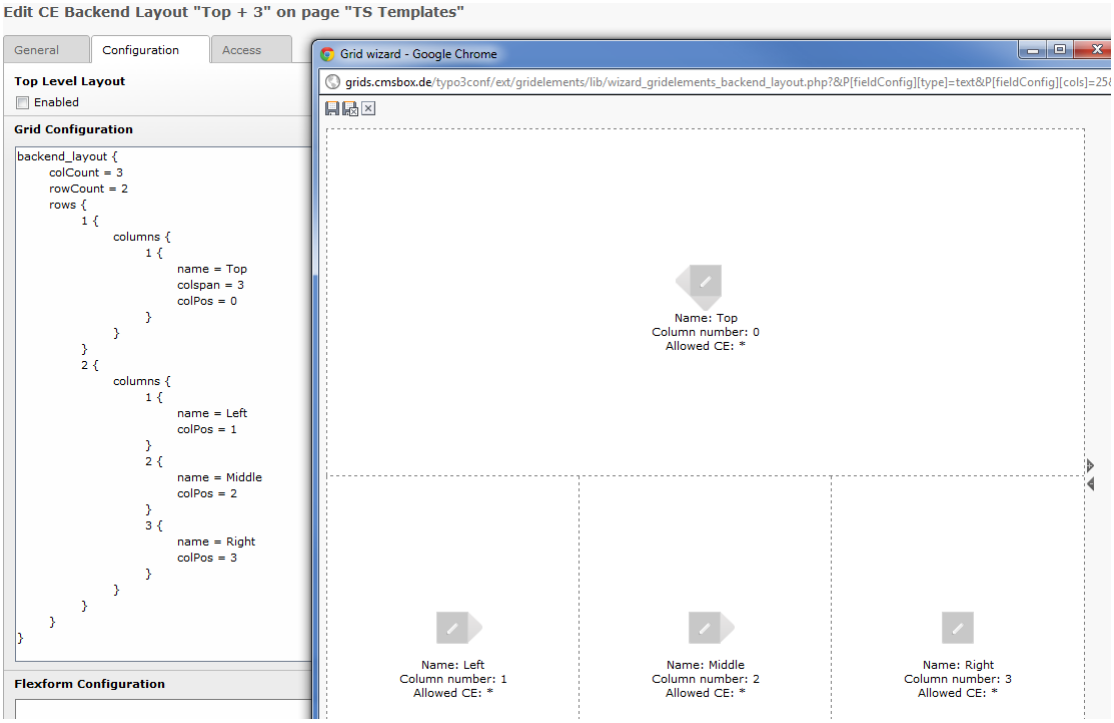
In most of the cases you want a working frontend output as well. So you should go to the Template module, edit your main TypoScript template record and include the static template there. This will be providing the basic TypoScript setup for the Grid Elements frontend plugin. You will find it in the select box at the *Include static (from extensions)* section of the *Includes* tab. Currently there is only one static template *gridelements(gridelements)* available. There might be more in future releases.



Create some CE backend layouts

To make use of any backend layout within content elements you have to create some *CE backend layout* records first. The process is similar to the one you might already know from the page backend layouts provided by the TYPO3 core.

- Switch to the list module and select the page, that you want to use as the container for your backend layouts
- If you are using a so called *General Record Storage Page*, i.e. for *tt_news*, you must place your backend layouts there as well. And since you can define a storage page for your backend layout records by *TSconfig*, you should select the page you have defined there, if any.
- Click on the *Create new record* button and select *CE backend layout* in the *Grid Elements* section.
- Give your element a title and description, upload an icon to be used in the layout selector box later on and select one of the available colors if you want to use a colored frame for your grid.
- Now you can either manually enter the TypoScript setup for your layout, or have it created with the Grid Wizard. Go to the appropriate chapters to find out how to do so.

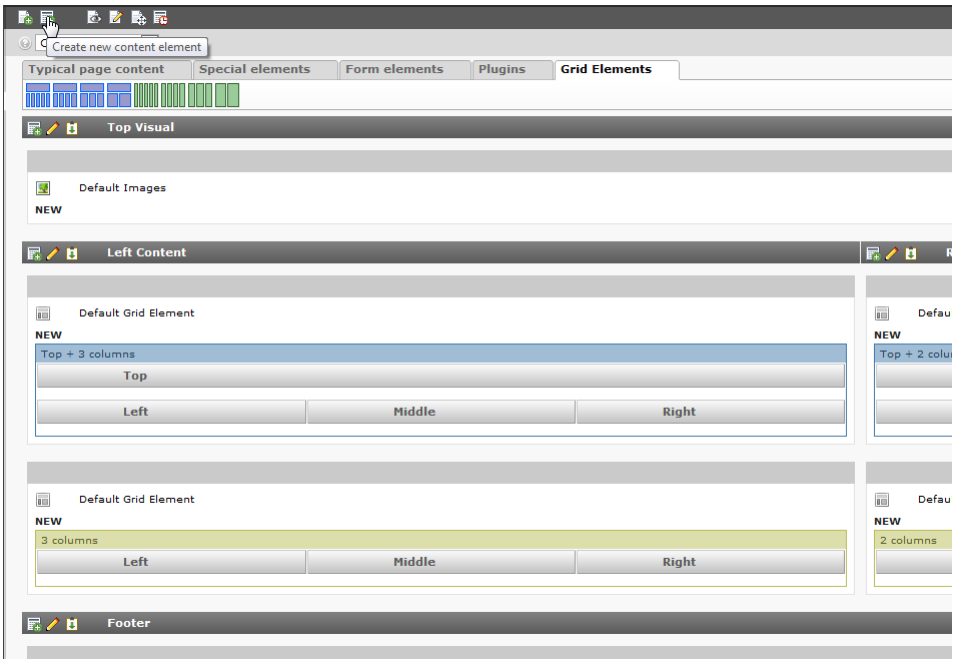


- If necessary you can fill in a flexform datastructure to provide additional settings within your grid element. Values of these flexforms will be available within the data set of the Grid Element during frontend output later on. Go to the flexform chapter to find out how to do that.

Now save the record and create some more layouts if you like.

Create new Grid Elements

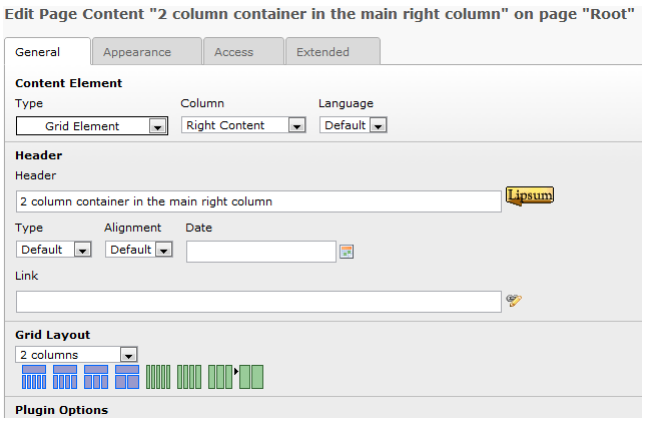
Now that you have some *CE backend layouts* available, you can easily use them to create new grid elements. There is a feature called *New Content Element Wizard Overlay*. So go to the page module now and activate it by clicking on the *Create new record* button on top of the module. Now you can drag any kind of content element from this wizard into any of the visible and active columns of the current page. Select one of the available Grid Elements and while you drag it, some light orange drop zones will appear to let you drop it into the desired column. After a few seconds the spinner symbol will disappear and show your newly created grid element. Drag in as many elements as necessary for the desired page layout.



Note: Of course you can drag elements into the columns of a Grid Element as well, as soon as you got at least one of them on your page. So nesting can be done with the drag in wizard within just a few seconds as well.

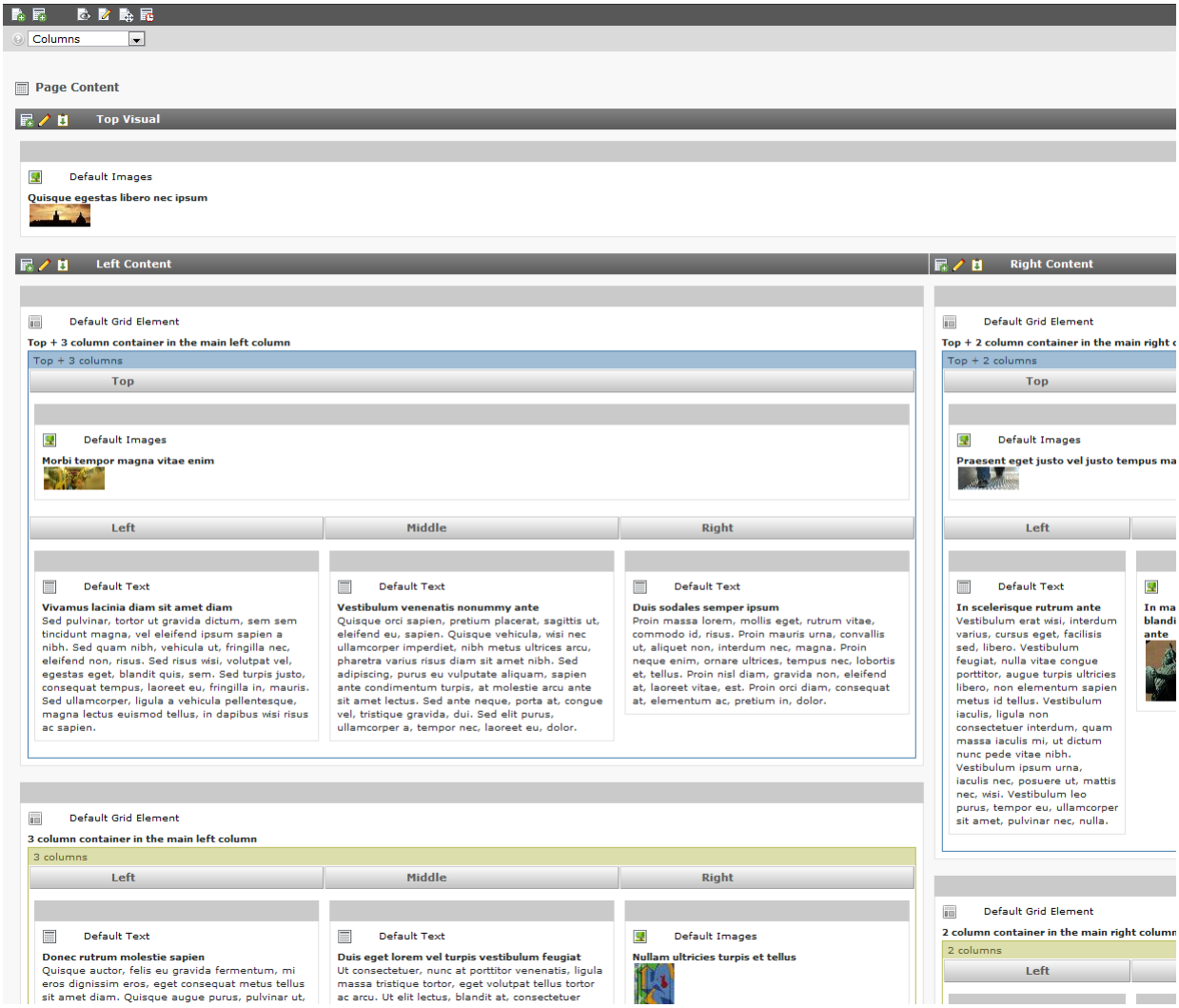
Change existing elements into Grid Elements

If you want to change existing elements into Grid Elements you can do so in the content editing form. Just edit the desired content element and change the type to *Grid Element*. The editing form will change and show you the appropriate fields. GO to the *Grid Layout* section of the *General* tab and select one of the backend layouts you have created before. Now save the record and close, and you should see the new Grid Element in your page module.



Fill your grid with content elements

Now that you have created all the necessary grids you can fill them with content elements. You can either use the same drag in wizard as you have been using while creating the grids or you can use the Create new record on top of this column button to add new content elements without dragging. Of course you can copy and/or move existing elements into the columns of your newly created grids and even creating references to elements in the clipboard is possible.



Include your own TypoScript

The default template of the plugin will just provide the most basic functionality. It will create the content of any column within a grid container as a serialized chain of child elements. So each of the children will be put into a div container, that will again be put into a div container of the column it has been taken from, that will finally be put into a div container of the parent Grid Element. If you want to provide more sophisticated stuff, go to the TypoScript section of this manual and see what is possible.

Grid TS Syntax

The syntax we use to store information about the grid structure within the page and CE backend layout records is basically *TypoScript*. Both grid view and grid elements are using the internal TS parser of the core to transform this syntax into an array, which is then used by the different methods we attached to the hooks provided by the core.

We could have used serialized objects or arrays as well, but decided to go for TypoScript, since this can easily be written by advanced integrators. For those, who are not familiar with TypoScript or just prefer the usability of a point and click interface, there is a comfortable Grid Wizard that will help to create the TypoScript code. Later on it might be more convenient to modify the structures by hand, especially when backend layouts that are based on a similar structure haven't got too many differences.

Step by step:

Start with the number of columns and rows

Go to the **Configuration** tab of the layout record and edit the **Grid Configuration** there. The wrapper for the whole block is the same as for pages: **backend_layout** – Use the keys **colCount** and **rowCount** to create the basic grid structure. Both values should be at least the lowest common multiple of the column sizes you want to create. They represent the actual grid behind the cell structure. The calculation should take into account that you might be using colspan and rowspan as well.

```
backend_layout {
  colCount = 4
  rowCount = 3
}
```

Fill in the rows

The array of rows does not offer any specialties. It is just a simple array with numeric keys. You will need a key for each possible row, even though it might stay empty later on.

```
backend_layout {
  colCount = 4
  rowCount = 3
  rows {
    1 {
    }
    2 {
    }
    3 {
    }
  }
}
```

Create the cells

Each of the cells comes with up to 5 different keys: **name**, **colPos**, **colSpan**, **rowSpan** and **allowed**. There must be at least the **name** and if you want to use the column as somehting else than a placeholder, there must be a value for **colPos** as well. Otherwise the cell will be marked as *inactive* in the page module.

The values for **colSpan**, **rowSpan** and **allowed** are optional. The **allowed** feature is not implemented yet, but planned to determine those content element types the user will be allowed to use within this column in future releases. The **colPos** value will be used while fetching the content elements from the database, since grid view and grid elements are using normalized relations to relate columns and content elements with each other.

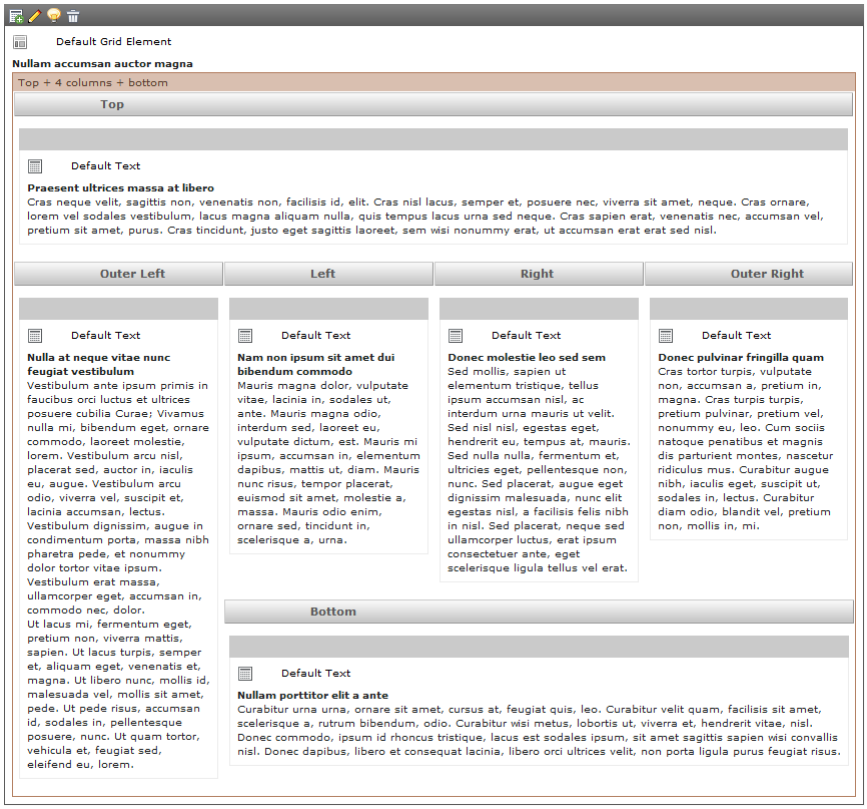
The following example will create a cell for a larger top column:

```
columns {
  1 {
    Name = Top
    colSpan = 4
    colPos = 0
  }
}
```

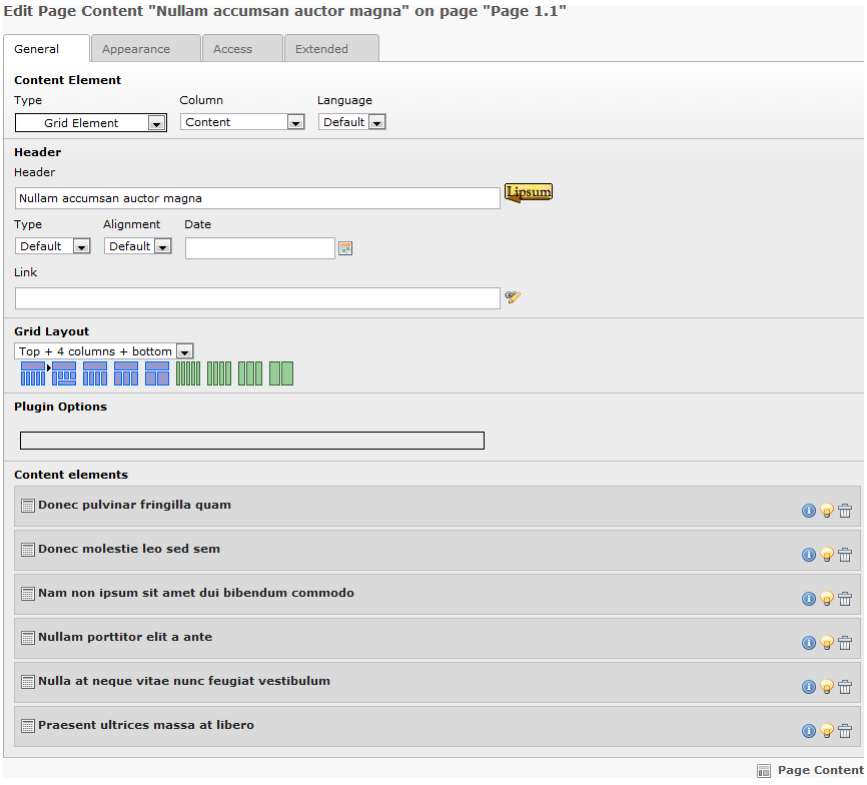
This is the complete example code

```
backend_layout {
  colCount = 4
  rowCount = 3
  rows {
    1 {
      columns {
        1 {
          name = Top
          colspan = 4
          colPos = 0
        }
      }
    }
    2 {
      columns {
        1 {
          name = Outer Left
          rowspan = 2
          colPos = 1
        }
        2 {
          name = Left
          colPos = 2
        }
        3 {
          name = Right
          colPos = 3
        }
        4 {
          name = Outer Right
          colPos = 4
        }
      }
    }
    3 {
      columns {
        1 {
          name = Bottom
          colspan = 4
          colPos = 5
        }
      }
    }
  }
}
```

This is the visible result of the example code



When you now edit this grid element, you can see how the child elements are connected to their parent grid via the core functions provided by Inline Relational Record Editing (IRRE). You will even be able to edit any element within a possible tree of nested grids and their children without having to deal with the whole page module, but of course you will lose the structured view of the grid this way. Sorting by D&D or clicking on the sorting arrows will be disabled as well.

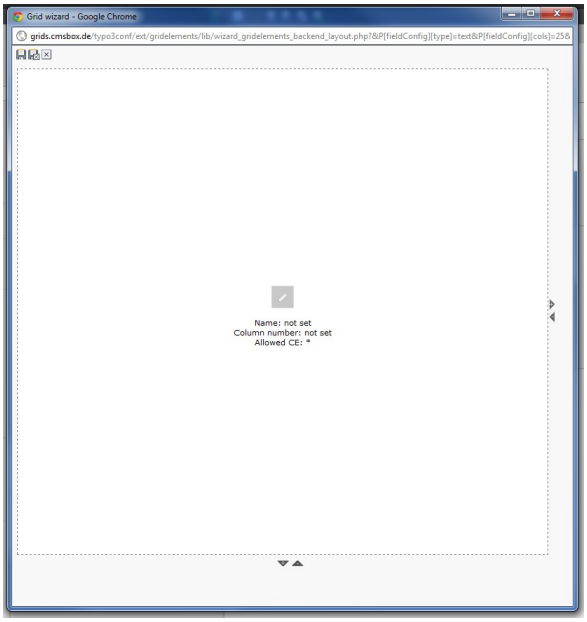


Grid Wizard

For those, who are not familiar with TypoScript or just prefer the usability of a point and click interface, there is a comfortable Grid Wizard that will help to create the TypoScript code.

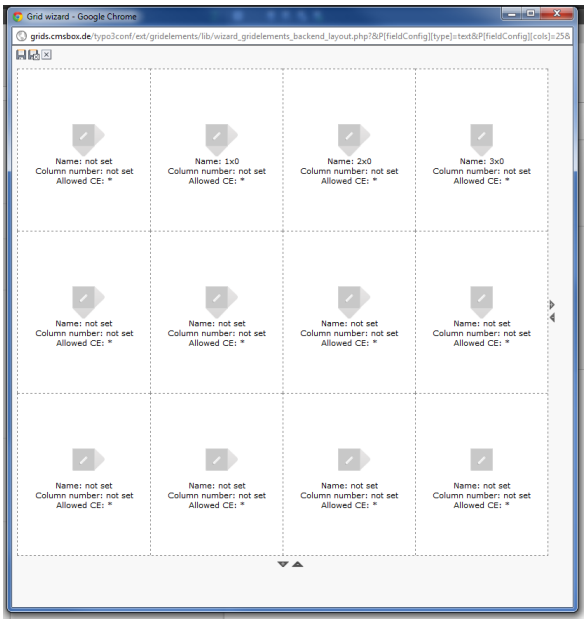
Creating the basic grid structure

When you want to use this wizard just go to the **Configuration** tab of the layout record, click on the **icon with the pencil** to the right of the of the **Grid Configuration** area and wait for the popup window to open. When this is a newly created record, the wizard will look like this:



Otherwise it will show a visible representation of the structure provided in the textarea.

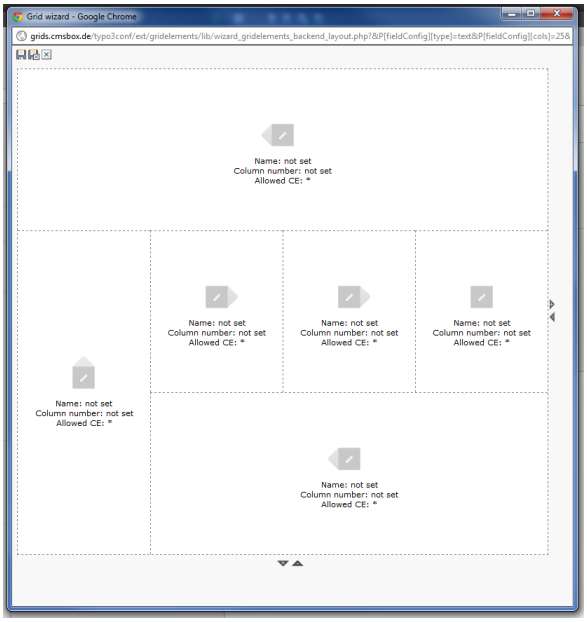
Now you can click on the **small triangles** at the right and at the bottom to create the basic grid structure. + will increase the number of columns and/or rows, - will decrease it. To get the example we have been using for the Grid TS Structure, the basic grid would be looking like this:



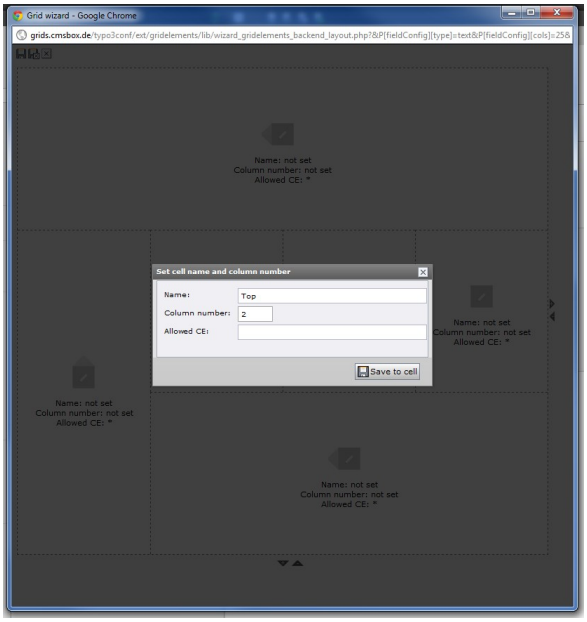
Spanning, naming and assigning cells

Now you can deal with the cells that should be **spanning multiple columns and/or rows**. Therefor you just have to click on the **triangle symbols beside the cells** you want to enlarge. You can span **right and down only**, since this resembles the way cells are spanned in the HTML table used within the page module. Only when you spanned a cell over at least one column and/or row there will be **additional triangles pointing to the left and up**, so that you can **remove** the spanning by clicking on them.

To create the structure of the Grid TS example, you first should click on the right triangle of the upper left cell until it spans the whole row. Then you should click on the bottom triangle of the first cell of the second row to have it span two rows. Finally you should click on the right triangle of the second cell of the last row until it spans the remaining three columns of the last row. Now the result should be looking like this:

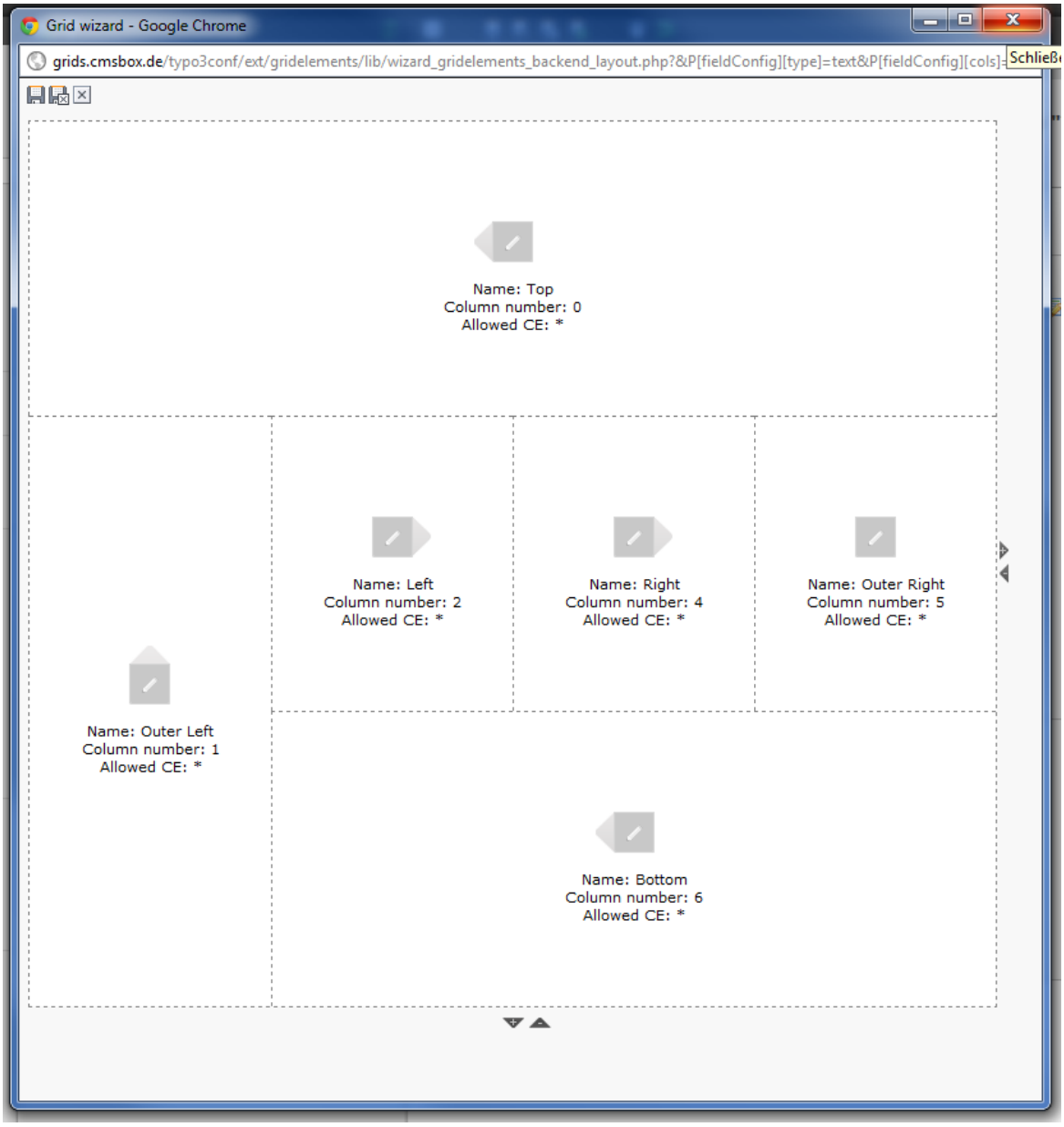


Finally you should give the cells a **name** and a number to be used as the value for the internal colPos within a grid element using this layout. If you don't set the **column number**, the cell will be a placeholder that can not contain any element later on. To edit the values for each cell, just click on the **pencil within the square** in the middle of each cell, fill in the values and save them by clicking on the **disk symbol**.



Saving the layout to the CE backend layout record

Now that you have named and assigned each cell, the layout should be looking like this:



You can save it by clicking on the **disk symbol at the upper left corner** of the popup window. Depending on the names and column values you have been using the result should be close the example we have been using in the Grid TS structure section. When you open the wizard the next time, it will come up in the same state.

Flexform

Each time you want a Grid Element to be more than just a structured multicolumn container with lots of different content elements, you can make use of the **Flexform Configuration** field of the CE backend layout record.

We thought that even though it is no good idea to store relations as CSV lists in XML structures within a database field, flexforms are still very useful when it comes to configurational stuff.

Anything that has nothing to do with the relation to the actual content elements can be put here using the same syntax as in any other flexform field:

- Checkboxes and/or radio buttons to enable or disable certain behaviours
- Selectors to get different variants of the Grid Element in the frontend
- Input fields for additional information besides the usual content elements
- Textareas for internal notes to the editors

You could even copy and paste TemplaVoila data structures here, which might be helpful during a migration process from FCEs to Grid Elements.

Anything defined in the configuration will show up in the **Plugin Options** of the Grid Element's editing form.

Currently any type of form field value will be transferred to the data set of the Grid Element record while rendering it in the frontend. They will be prefixed with "**flexform_**" to make sure that they don't override any other field with the same name, but still can be accessed via the usual TypoScript functions.

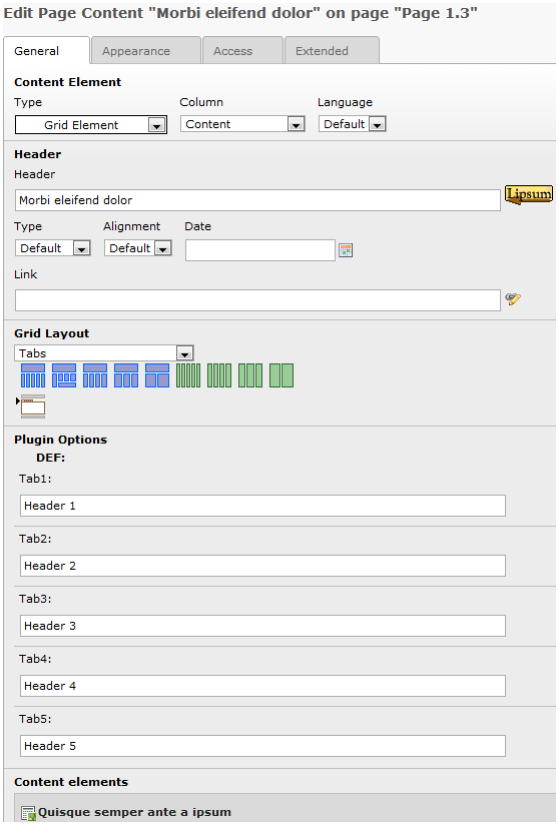
Sections are not supported by the frontend rendering process yet.

Some examples

This setup will add some simple input fields to the form of a Grid Element. This element will create some jQuery based tabs in the frontend output of the webpage later on:

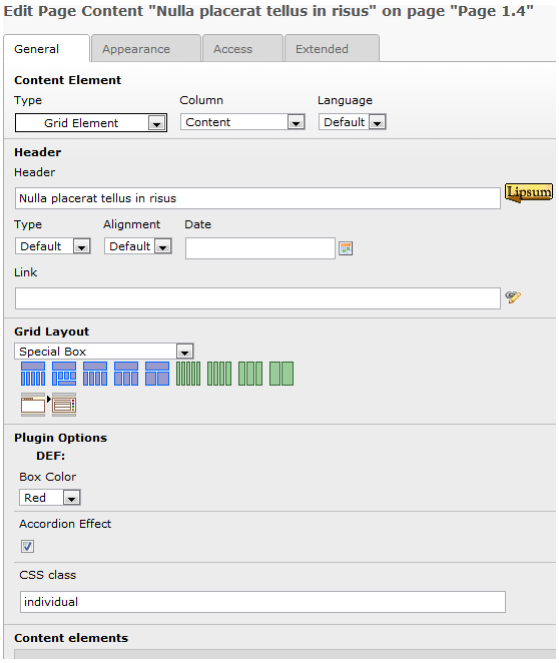
```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<T3DataStructure>
  <ROOT type="array">
    <type>array</type>
    <el type="array">
      <tabheader_1 type="array">
        <TCEforms type="array">
          <config type="array">
            <type>input</type>
            <size>48</size>
            <eval>trim</eval>
          </config>
          <label>Tab1: </label>
        </TCEforms>
      </tabheader_1>
        ...
      <tabheader_5 type="array">
        <TCEforms type="array">
          <config type="array">
            <type>input</type>
            <size>48</size>
            <eval>trim</eval>
          </config>
          <label>Tab5: </label>
        </TCEforms>
      </tabheader_5>
    </el>
  </ROOT>
</T3DataStructure>
```

The backend form of this setup will be looking like this:



Take a look at the TypoScript section to find out how this would be rendered in the frontend.

And to get this kind of structure for a Grid Element box with special features …



… you will need this setup. It will create some special boxes in the frontend, that can have frames in different selectable colors and an additional jQuery based accordion that can be enabled by the user:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<T3DataStructure>
  <ROOT type="array">
    <type>array</type>
    <el type="array">
      <color type="array">
        <TCEforms type="array">
          <label>Box Color</label>
          <config type="array">
            <type>select</type>
            <items type="array">
              <numIndex index="0" type="array">
                <numIndex index="0">Red</numIndex>
                <numIndex index="1">1</numIndex>
              </numIndex>
              <numIndex index="1" type="array">
                <numIndex index="0">Green</numIndex>
                <numIndex index="1">2</numIndex>
              </numIndex>
            </items>
          </config>
        </TCEforms>
      </color>
      <accordion type="array">
        <TCEforms type="array">
          <label>Accordion Effect</label>
          <config type="array">
            <type>check</type>
            <default>0</default>
          </config>
        </TCEforms>
      </accordion>
      <class type="array">
        <TCEforms type="array">
          <config type="array">
            <type>input</type>
            <size>48</size>
            <eval>trim</eval>
          </config>
          <label>CSS class</label>
        </TCEforms>
      </class>
    </el>
  </ROOT>
</T3DataStructure>
```

TSconfig

It is necessary to prevent children of Grid Elements from being visible in the list view, due to serious problems that might occur when using the up and down arrows to move an element.

Since the TYPO3 Core just knows two ways of moving “move after (an element)” or “move into (the top of a column)”, you will run into problems when moving a child element of a grid container after a child element of another container. In this case the moved element will get another parent container, which is in most of the use cases not what we want.

So currently the most important TSconfig setting used by Grid Elements is:

```
TCEFORM.tt_content.tx_gridelements_backend_layout {
  removeChildrenFromList = 1
}
```

TypoScript

FAQ

ToDo

Find a way to handle move actions triggered by the up and down arrows of the list module correctly.

Notes

Important note about the colPos field!

The colPos field of the tt_content table will be changed from unsigned int(3) to smallint(6) to enable the usage of negative values. This should **never be reverted** by any upgrade script later on! Otherwise any child element will be moved from it's parent container to the default page column. Some people ran into problems during automatic upgrades done by their providers. So be sure to make a backup of your content before upgrading!

Important note about the list module!

There might be problems when trying to sort child elements with the up and down arrows in the list module. See the Tsconfig section to find out how to prevent child elements from being visible in the list as a workaround.

Changelog

1.3.12 => 2012-04-27

fixed broken D&D move actions introduced with 1.3.10

1.3.11 => 2012-04-25

fixed dependency issues

fixed manual

fixed coding guideline issues

1.3.10 => 2012-04-18

fixed issue #35959 Localize labels completely (Thanks to Marc von Schalscha)

fixed issue #36144 Move in list-view: moved elements are still there

1.3.7 => 2012-04-16

added manual.sxw and manual.pdf - TypoScript and TSconfig sections still missing

prepared tce_main hook functions for an upcoming core patch that will fix problems while importing T3D packages containing grid elements

1.3.6 => 2012-03-30

fixed problems with Grid Container column being "not allowed" in case people were using no backend layout but their own TSconfig for colPos items

1.3.5 => 2012-03-28

fixed problems while moving records introduced with the last fix

1.3.4 => 2012-03-23

fixed problems while moving records

fixed problems while moving records between pages

1.3.3 => 2012-03-13

fixed issue #34934 Does not scroll during drag

fixed issue #34719 Removed TS for non existing HTML template

fixed issue #34785 user setting (hideColumnheaders) does not work

fixed issue #34868 wrong tabindex of wizard input fields

fixed issue #34810 copy/cut/paste problems with multiple clicks on the copy/cut links of the same element

fixed issue #34810 copy/paste problem across pages (insert on copy source page)

some trailing whitespace cleanup

1.3.2 => 2012-03-06

fixed missing class instantiation

1.3.1 => 2012-03-06

Removed console log and debug output

1.3.0 => 2012-03-05

Fixed Bug #34045 Bug when no translation exists

Fixed Bug #34109 PHP Warning: strcmp() expects parameter 1 to be string

Fixed Bug #33364 Drag and drop initiation breaks with "cannot call hasClass on null"

Fixed Bug #34045 Bug when no translation exists

Feature #32354 Show Grid Layout

Fixed Bug #32355 Grid Columns lost

Fixed Bug #33388 Drop zone - sorting related

Fixed Bug #32388 Sorting of CE

Feature #33389 Add new content element within container

Fixed Bug #33490 Attribute name of "icon(s)" in class.tx_gridelements_layoutsetup.php

Fixed Bug #33402 Column layout broken if page languages overview is selected

1.2.3

Fixed: Issue #33319 by reverting the particular file to version 1.1.0

1.2.2

Fixed: Issue #33301 getLL has been replaced with a non existing function name

1.2.1

Fixed: Empty columns show their numbers in the FE

1.2.0

Added: Issue #32835 Missing language overlay and versioning preview support

Fixed: Removal of unused variables and code cleanup

Fixed: Sorting problem with nested grid tables that don't use ascending column order

Fixed: Issue #32241 bug in class.tx_gridelements_tcemainhook.php

Fixed: Issue #32355 Grid columns lost

Fixed: Issue #31804 Error while saving container

Added: Issue #30830 Localized column name

Fixed: Issue #30923 Wrong column wrapping

Added: Paste reference after for click menus

Added: Issue #32510 Parent data access

Added: Paste reference into for page and grid columns

Added: Usage of large icons for the drag in wizard overlays

1.1.0 => 2011-11-07

Fixed: $BACK_PATH issue with symlinked filesystem, now there's a $BACK_PATH_ABS variable - only used for PHP requires

Fixed: backend layout wizard was linked to typo3/ext for some reason, path edited to use t3lib_extMgm::extRelPath($_EXTKEY)

Fixed: handling of "unused elements" in colPos -2 is wrong

Fixed: number of child elements is not updated while creating new children

Added: server time for future reload-less DnD (page edit time will be compared against this)

Added: templates for future reload-less DnD of new content elements

Added: options to switch reloads and templates off for future reload-less DnD

Added: exclude current parent of dragged element from targets (drop now has no effect/does not reload)

1.0.0 => 2011-10-10

Fixed different bugs with D&D in IE8

Fixed the handling of the layout wizard BACK_PATH for global and local installs of the extension

Fixed bug: Dragging in of new elements after elements outside of grid containers does not work

0.6.0 => 2011-10-09

Drag In of new content elements complete

Just activate the drag in wizard by clicking on the "create new content element" icon on top of the page

0.5.0 => 2011-10-09

lots of improvements to the D&D feature

Drag & Drop works properly now including copying with CTRL-key pressed

Grid containers are updated and logged as well during actions of a child element between one or two container(s)

Basic concept for Drag In of new elements is already in the code base but still deactivated

Still some cosmetical things to do

0.4.0 => 2011-09-19

added basic JS for drag-and-drop

0.3.0 => 2011-09-16

Just a double upload during the security fix phase

0.2.0 => 2011-09-06

Security fix

0.1.0 => 2011-09-06

Initial upload. Grid view for CE works completely.

D&D, FE output via TS/plugin and manual still pending.

Can be installed together with TV but will disable some of the TV hooks, which is the reason for the message during the install.

So you can still migrate from TV to the new structures before removing TV.

Additional Links