Grid Elements

This project aims to integrate the grid layout concept also to regular content elements - the grid elements.
This approach is an alternative to TemplaVoila storing relations normalized in the database without using XML and offers a lot of comfortable features for an improved experience of the backend user.

It's development has been financed by T-Systems, Cybercraft and via a crowd funding project @ http://www.startnext.de/en/typo3-grid-elements-2-0 with sponsoring of 99 Supporters

Version: 2.1.0, 2014-07-28

Extension Key: gridelements

Language: en

Keywords: gridelements

Copyright: 2014, Jo Hasenau, Cybercraft Media Manufactory, <info@cybercraft.de>

Copyright for contributions: Take a look at the team at http://forge.typo3.org/projects/typo3v4-gridelements

The content of this document is related to TYPO3

- a GNU/GPL CMS/Framework available from www.typo3.org

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 and assign columns and allowed content types

- 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

- Grid and backend layout aware list module with top level sorting that can list children of containers as nested list via AJAX

- 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

- Get me a copy from another page icon so you won't have to switch pages when fetching content

- New content element wizard overlay to drag in new content elements

- Paste icons for pasting copies and references into grid columns

- References to complete pages can be used within the reference element

- Referenced content visible in the preview section of the reference element

- Completely TypoScript based frontend output

- Flexform field values automatically added to the data set

- Prerendered content and arrays added to the data set to be used even with other templating engines like Smarty or Fluid

- No need for HTML templates and mapping to get a backend layout and/or frontend output

- Completely based on hooks without XCLASSing (only exception is the list module due to missing hooks)

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.
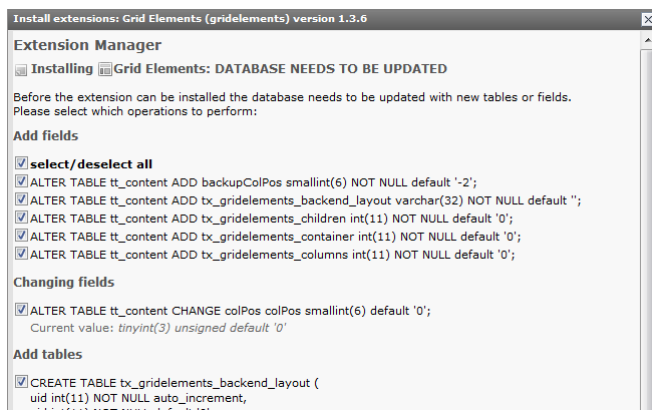
Important Note: The colPos field of the tt_content table will be changed from unsigned int(3) to smallint(6) to enable the usage of negative values. This should **never be reverted** by any upgrade script later on! Otherwise any child element will be moved from it's parent container to the default page 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.
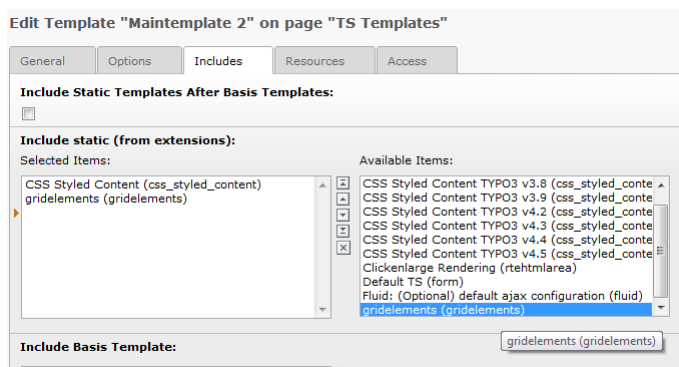
Create some CE backend layouts

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.

- If you want the children of the Grid container to appear side by side (i.e. to create a tab container or a twitter bootstrap row), you can activate the checkbox.

- 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.

Edit CE Backend Layout "Top + 3" on page "TS Templates"

General	Configuration	Access

Top Level Layout
☐ Enabled

Grid Configuration

```
backend_layout {
    colCount = 3
    rowCount = 2
    rows {
        1 {
            columns {
                1 {
                    name = Top
                    colspan = 3
                    colPos = 0
                }
            }
        }
        2 {
            columns {
                1 {
                    name = Left
                    colPos = 1
                }
                2 {
                    name = Middle
                    colPos = 2
                }
                3 {
                    name = Right
                    colPos = 3
                }
            }
        }
    }
}
```
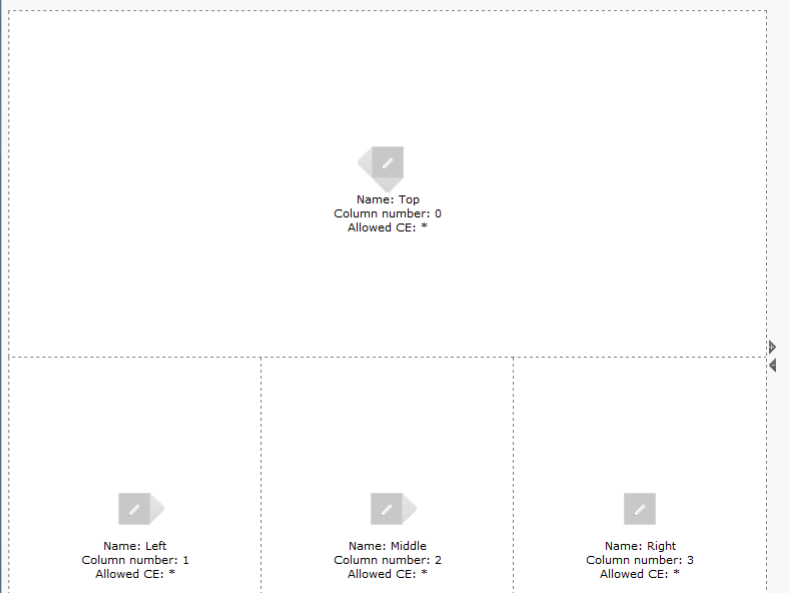
Flexform Configuration

Grid wizard - Google Chrome

grids.cmsbox.de/typo3conf/ext/gridelements/lib/wizard_gridelements_backend_layout.php?&P[fieldConfig][type]=text&P[fieldConfig][cols]=25...

Name: Top
Column number: 0
Allowed CE: *

Name: Left
Column number: 1
Allowed CE: *

Name: Middle
Column number: 2
Allowed CE: *

Name: Right
Column number: 3
Allowed CE: *

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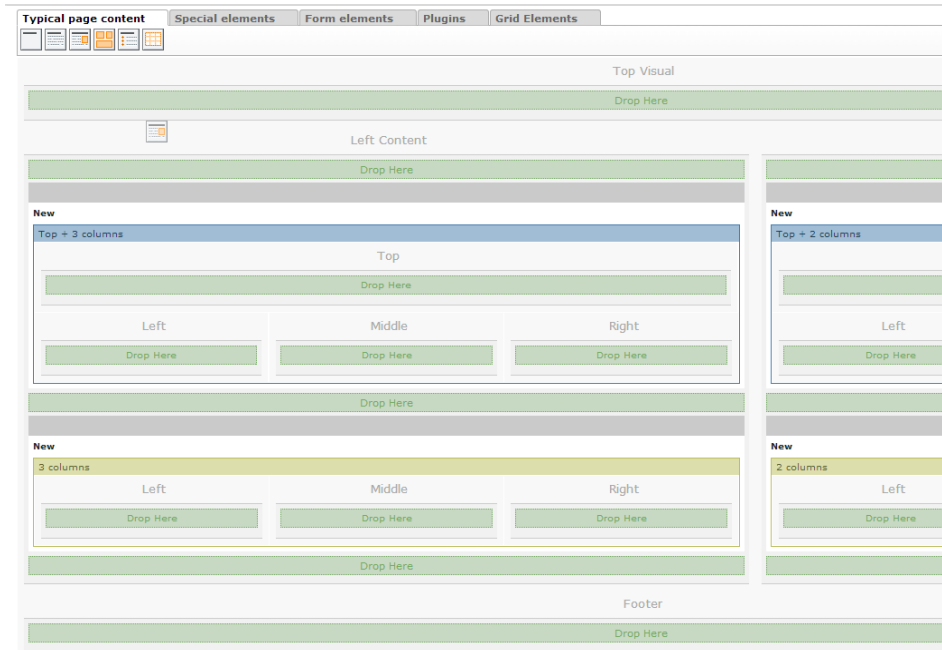
Note: Of course you can drag elements into the columns of a Grid Element as well, as soon as you got at least one of them on your page. So nesting can be done with the drag in wizard within just a few seconds as well.

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.
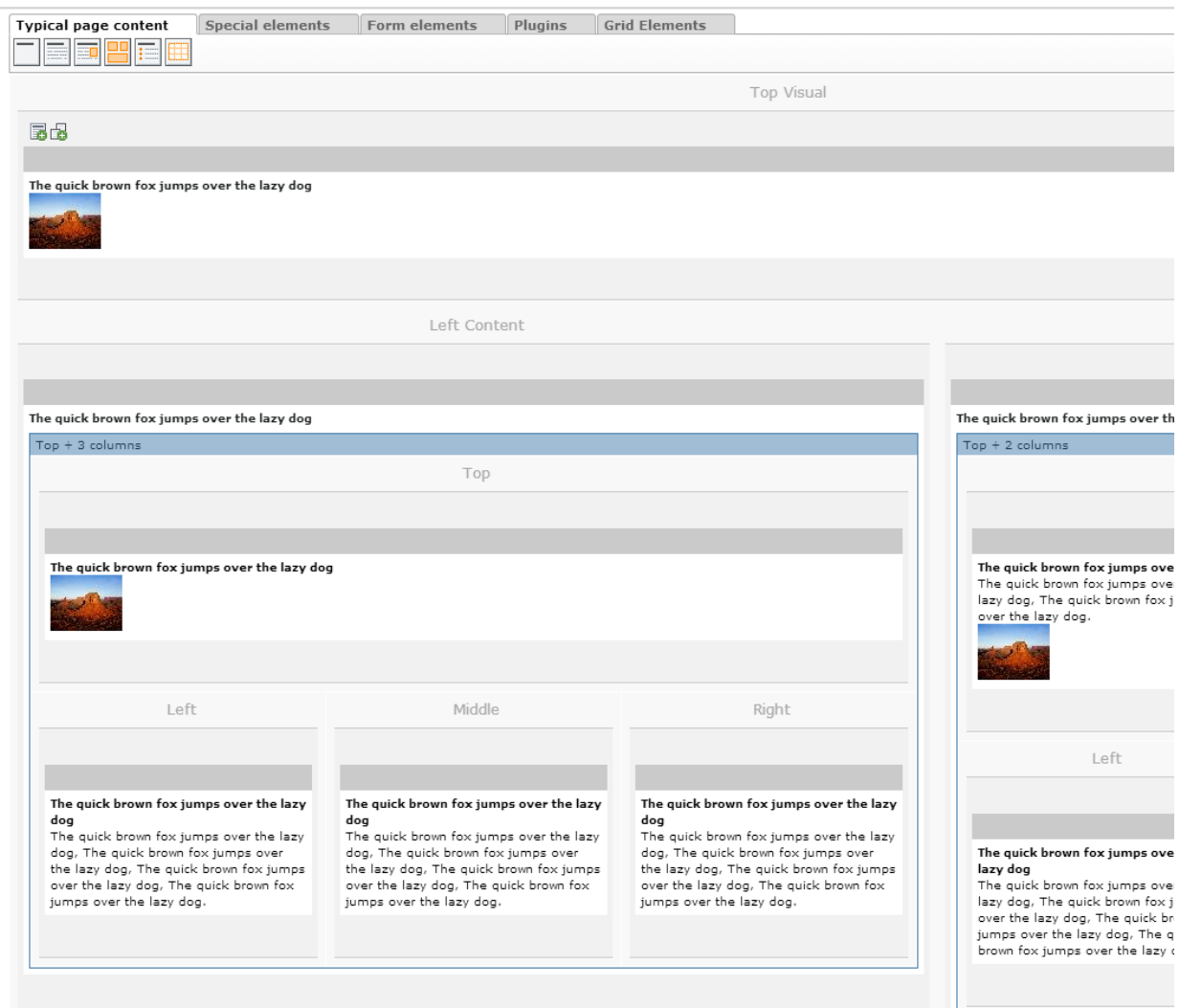
Include your own TypoScript

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 something 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 used to determine those content element types the user will be allowed to use within this column. You can use a comma separated list of Ctype values here and as soon as this contains at least one value, any other element type will be forbidden. 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 with only *text* and *text with image* allowed as a content type:

```
columns {
  1 {
    Name = Top
    colspan = 4
    colPos = 0
    allowed = text,textpic
  }
}
```

This is the complete example code

```
backend_layout {
  colCount = 4
  rowCount = 3
  rows {
    1 {
      columns {
        1 {
          name = Top
          colspan = 4
          colPos = 0
          allowed = text,textpic
        }
      }
    }
    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
        }
      }
    }
  }
}
```
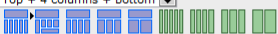
This is the visible result of the example code
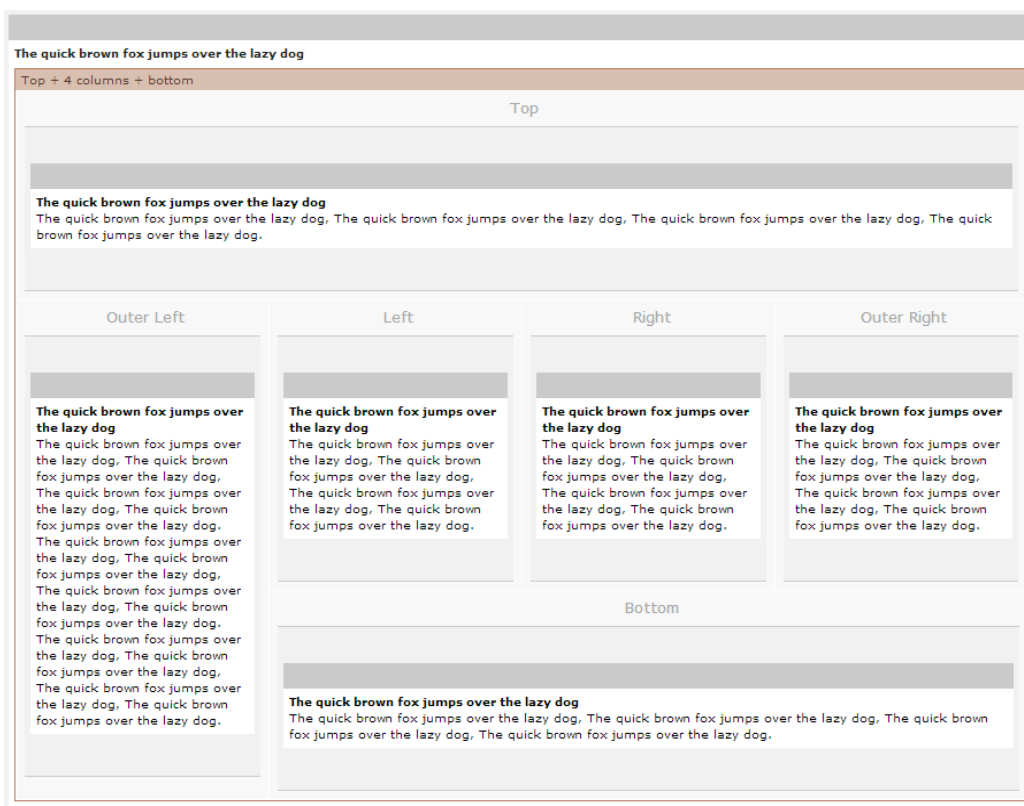
When you now edit this grid element, you can see how the child elements are connected to their parent grid via the core functions provided by Inline Relational Record Editing (IRRE). You will even be able to edit any element within a possible tree of nested grids and their children without having to 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 inside the editing form as well. But you still can sort elements by directly dragging and dropping them in the page module.

Edit Page Content "Nullam accumsan auctor magna" on page "Page 1.1"

| General | Appearance | Access | Extended |

Content Element

| Type | Column | Language |
| Grid Element | Content | Default |

Header

Header

Nullam accumsan auctor magna   [Lipsum]

| Type | Alignment | Date |
| Default | Default | |

Link

Grid Layout

Top + 4 columns + bottom

Plugin Options

Content elements

Donec pulvinar fringilla quam	ⓘ 💡 🗑
Donec molestie leo sed sem	ⓘ 💡 🗑
Nam non ipsum sit amet dui bibendum commodo	ⓘ 💡 🗑
Nullam porttitor elit a ante	ⓘ 💡 🗑
Nulla at neque vitae nunc feugiat vestibulum	ⓘ 💡 🗑
Praesent ultrices massa at libero	ⓘ 💡 🗑

Page Content

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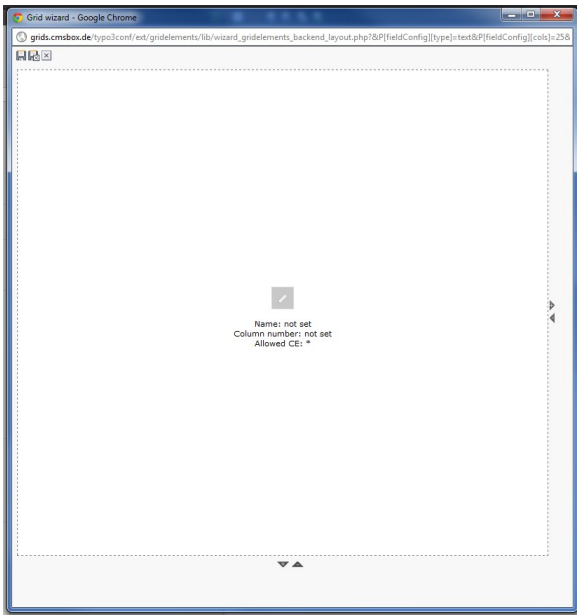
Otherwise it will show a visible representation of the structure provided in the textarea.

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:
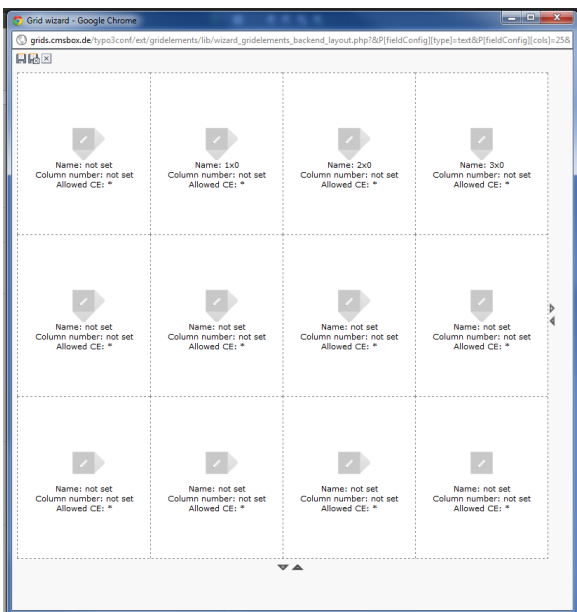
Spanning, naming and assigning cells

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 should click on the right triangle of the upper left cell first 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:
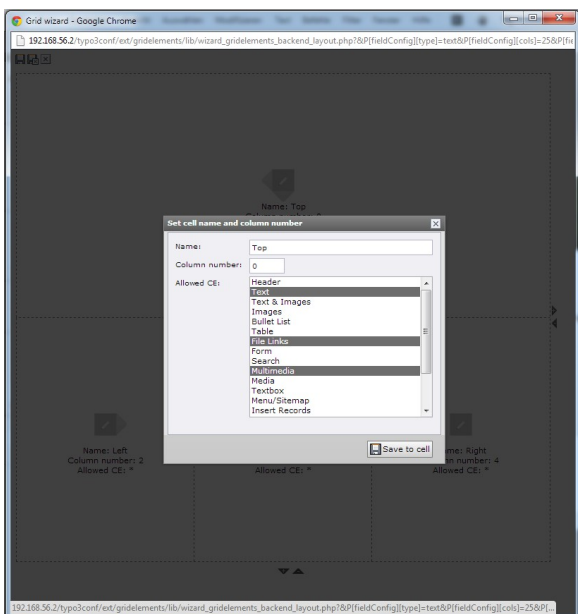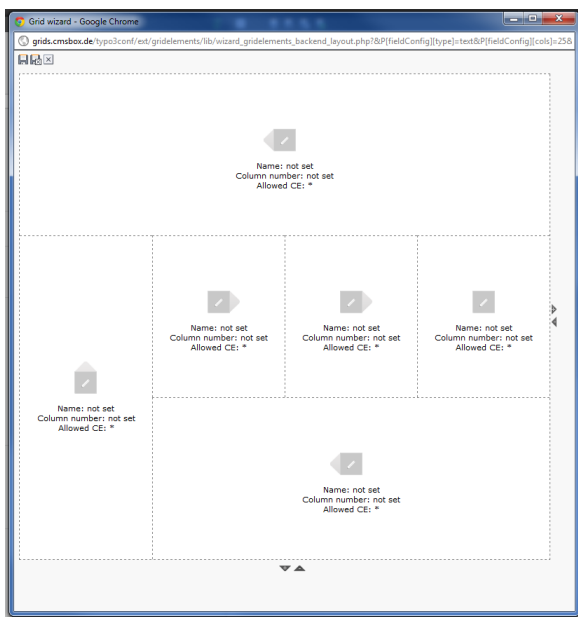
Finally you should give the cells a **name** and a number to be used as the value for the internal colPos within a grid element using this layout. And you should decide about the available content types for each cell. If you don't set the **column number**, the cell will be a placeholder that can not 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:
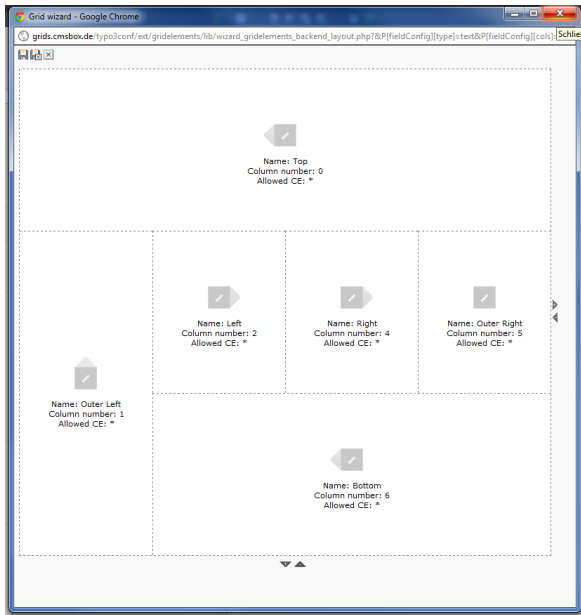
You can save it by clicking on the **disk symbol at the upper left corner** of the popup window. Depending on the names and column values you have been using, the result should be close to the example we have been using in the Grid TS structure section. When you open the wizard the next time, it will 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

Example 1 - Element with child content and handling settings (recommended)

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
<?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>
```

16

Example 2 - Element with child content and additional input (partly recommended)

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
<?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.

Example 3 - a basic "FCE" - not recommended(!) but anyway here it is

An often asked task, especially if you migrate from TemplaVoila, will be to provide the editors with the predefined content elements (FCEs) they are used to. This can be done by completely defining the structure as a flexform and handling it via TypoScript. You should by any means avoid these FCEs, since you will lose any advantage of the normalized Grid Elements approach. Actually Flexform fields should **not** contain any real content elements but just configurational stuff like checkboxes, radiobuttons and selectboxes.

Still this setup will define some input fields to render a very basic teaser element in the frontend. It uses a headline, a link, an image and some text.

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<T3DataStructure>
  <ROOT type="array">
    <type>array</type>
    <el type="array">
      <headline type="array">
        <TCEforms type="array">
          <label>Headline</label>
          <config type="array">
            <type>input</type>
            <size>48</size>
            <eval>trim</eval>
          </config>
        </TCEforms>
      </headline>
```

```
<image type="array">
  <TCEforms type="array">
    <label>Image</label>
    <config type="array">
      <type>group</type>
      <internal_type>file</internal_type>
      <allowed>jpg,gif,png</allowed>
      <max_size>2000</max_size>
      <uploadfolder>uploads/tx_gridelements</uploadfolder>
      <show_thumbs>1</show_thumbs>
      <size>3</size>
      <minitems>0</minitems>
      <maxitems>1</maxitems>
    </config>
  </TCEforms>
</image>
<link type="array">
  <TCEforms type="array">
    <config type="array">
      <type>input</type>
      <eval>trim</eval>
      <wizards type="array">
        <_PADDING>2</_PADDING>
        <link type="array">
          <type>popup</type>
          <title>Link</title>
          <icon>link_popup.gif</icon>
          <script>
            browse_links.php?mode=wizard&amp;act=page
          </script>
          <params type="array">
            <blindLinkOptions>
              file,spec,email,folder
            </blindLinkOptions>
          </params>
          <JSopenParams>
            height=300,width=500,status=0,¬
              menubar=0,scrollbars=1
          </JSopenParams>
        </link>
      </wizards>
    </config>
    <label>Link</label>
  </TCEforms>
</link>
<text type="array">
  <TCEforms type="array">
```
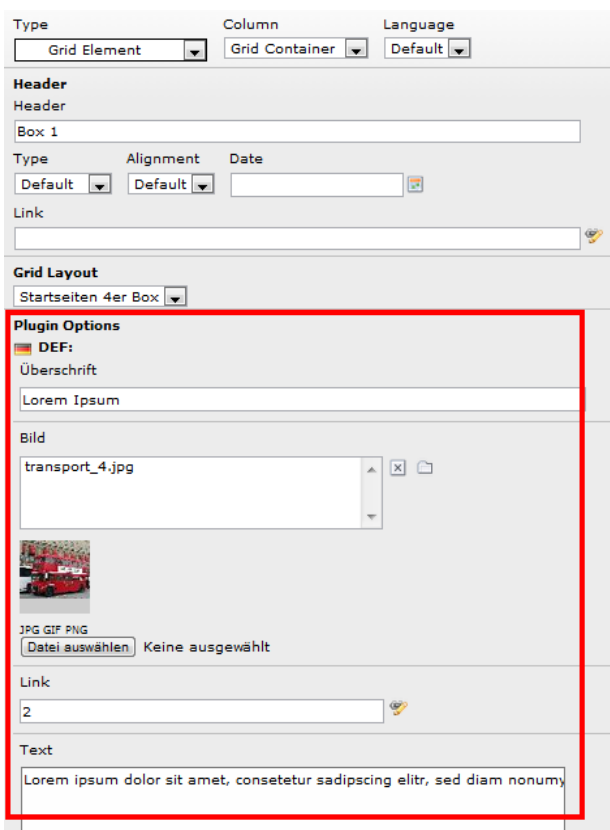
```
        <config type="array">
          <type>text</type>
          <cols>40</cols>
          <rows>10</rows>
          <wrap>off</wrap>
        </config>
        <label>Text</label>
      </TCEforms>
    </text>
  </el>
  </ROOT>
</T3DataStructure>
```

Be aware of wrapped lines. For a working, easy to copy scripting, have a look at https://gist.github.com/3294052.
The backend form of this setup will look like this:

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

TSconfig

It has been necessary to prevent children of Grid Elements from being visible in the list view, due to serious problems that occured when using the up and down arrows to move an element. **Due to some additional features we included in the list module this is not used anymore and can be removed from your settings**

```
TCEFORM.tt_content.tx_gridelements_backend_layout {

  removeChildrenFromList = 1

}
```

These are the other TSconfig option you will get to even configure Grid Elements completely without grid records:

Property:	Data type:	Description:	Default:
tx_gridelements.setup	Grid TS structure	Contains the setup of different grid layouts	
tx_gridelements.setup.123	Grid TS structure	Contains the setup of the grid layout with the ID 123 (see the GRID TS section for an example) – You can create the structure with the Grid Wizard and then copy it to your TSconfig.	
tx_gridelements.overruleRecords	boolean	Set this value to 1, when you want TSconfig settings to ave precendence over layout records with the same ID	0
tx_gridelements.excludeLayoutIds	List of integers	A list of IDs of layouts that should not be available in this branch of the page tree	
TCEFORM.tt_content. tx_gridelements_backend_layout. PAGE_TSCONFIG_ID	Integer	The ID of the page that contains your layout records	

TypoScript

This is the default TypoScript setting provided while including its setup in your TS template editor:

```
lib.gridelements.defaultGridSetup { // stdWrap functions being applied to each element
  columns {
    default {
      renderObj = COA
      renderObj {
        # You can use registers to i.e. provide different image settings for each column
        # 10 = LOAD_REGISTER
        20 =< tt_content
        # And you can reset the register later on
        # 30 = RESTORE_REGISTER
      }
    }
  }
  # if you want to provide your own templating, just insert a cObject here
  # this will prevent the collected content from being rendered directly
  # i.e. cObject = TEMPLATE or cObject = FLUIDTEMPLATE will be available from the core
  # the content will be available via fieldnames like
  # tx_gridelements_view_columns (an array containing each column)
  # or tx_gridelements_view_children (an array containing each child)
  # tx_gridelements_view_column_123 (123 is the number of the column)
  # or tx_gridelements_view_child_123 (123 is the UID of the child)
}

lib.tt_content.shortcut.pages = COA
lib.tt_content.shortcut.pages {
  10 = USER
  10 {
    userFunc = tx_gridelements_view->user_getTreeList
  }
  20 = CONTENT
  20 {
    table = tt_content
    select {
      pidInList.data = register:pidInList
      where = colPos >= 0
      orderBy = colPos,sorting
      orderBy.dataWrap = FIND_IN_SET(pid,'{register:pidInList}'),|
    }
  }
}
```

```
tt_content.shortcut.5 = LOAD_REGISTER
tt_content.shortcut.5 {
  tt_content_shortcut_recursive.field = recursive
}


tt_content.shortcut.20 {
  0 {
    tables := addToList(pages)
    conf.pages < lib.tt_content.shortcut.pages
  }
  1 {
    tables := addToList(pages)
    conf.pages < lib.tt_content.shortcut.pages
  }
}


tt_content.gridelements_pi1 >
tt_content.gridelements_pi1 = COA
tt_content.gridelements_pi1 {
  #10 =< lib.stdheader
  20 = COA
  20 {
    10 = USER
    10 {
      userFunc = tx_gridelements_view->main
      setup {
        default < lib.gridelements.defaultGridSetup
      }
    }
  }
}
```

As you can see, it is just based on the usual TypoScript and uses some internal keys, like columns, default and renderObj to define the setup for the columns. Anything else you want to use will be based on the official TypoScript syntax, so you won't have to hassle with any extension specific parameters.

As described in the commented part, you will find some additional virtual fields in your data, that will contain stuff that has been used during the rendering process. These come in handy, when you want to use a TEMPLATE or FLUIDTEMPLATE element to produce your output.

Any of the internal keys and the default settings will of course be passed to the stdWrap method, so you can assign almost anything to any part of your setup.

The two setups for the shortcut cObject are used to render the references properly, so you should probably not change them unless you really know what you are doing.

Example 3 - a basic "FCE" continued

This part shows the TypoScript to render the output for the frontend. The expected result will look like this:

Lorem Ipsum

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam more...

The TypoScript is like this:

```
tt_content.gridelements_pi1.20.10.setup {
  1 < lib.gridelements.defaultGridSetup
  1 {
    prepend = COA
    prepend {
      10 = TEXT
      10 {
        data = field:flexform_headline
        typolink.parameter.data = field:flexform_link
        wrap = <h3>|</h3>
      }
      20 = IMAGE
      20 {
          file.import.data = field:flexform_image
          file.width = 200
          file.height = 133c
      }
      30 = COA
      30 {
        10 = TEXT
        10 {
          data = field:flexform_text
          crop = 150 |   | 1
        }
        20 = TEXT
        20 {
          value =  more...
          typolink.parameter.data = field:flexform_link
        }
        wrap = <p>|</p>
```

```
      }
    }
  }
}
```

The easy to copy scripting can be found at https://gist.github.com/3294347.

The "1" in line 2 and 3 reflect the uid of the gridelements record, fit this to your needs.

TypoScript Reference

Property:	Data type:	Description:	Default:
setup	Internal /stdWrap	The setup handed over to the userFunc that is responsible for the rendering process	
setup.default	Internal / stdWrap	The default setup used by any Grid Element layout that has not got its own setup available. Layouts are assigned by their UID It usually contains the columns parameter and might use additional stdWrap methods. If you provide a cObject as stdWrap method, this will override the default output and give you the option to make use of the special virtual fields containing prerendered stuff.	
setup.123	Internal / stdWrap	This setup will be used for the Grid Element layout with the UID value 123	
columns	Internal / stdWrap	This contains the setup for the default column and any other column that might differ from the default. Columns are assigned by their column value.	
columns.default	Internal / stdWrap	This will be the setup used for any column that has not got its own setup available	
columns.123	Internal / stdWrap	This will be used for the column with the column value 123	
renderObj	cObject / stdWrap	Can be any kind of TypoScript content object, like TEXT or TEMPLATE	COA
flexform_fieldname	Flexform field	Get the value of a field defined in the flexform of the grid layout of the current cObj	
parentgrid_fieldname	Value of the parent container	Get the value of a field of the parent container of the current cObj	
tx_gridelements_view_children	Array	An array containing all the children as raw data	
tx_gridelements_view_columns	Array	An array containing the prerendered columns	
tx_gridelements_view_column_123	HTML	The prerendered content of column 123	
tx_gridelements_view_child_123	HTML	The prerendered child with the uid 123	

FAQ

Question: Are Grid Elements a full replacement for TemplaVoila?

Answer: Grid Elements are a replacement for the backend related features of TemplaVoila, since they will provide you with structured views of containers and additional functionality like drag & drop, references and handling of unused elements. But they will not provide any kind of mapping tool to connect existing HTML elements to these structures.

Question: How can I connect my HTML templates to the structures?

Answer: They can be connected just like the page templates by using the TEMPLATE or FLUIDTEMPLATE objects of TypoScript. You will even get prerendered virtual fields containing arrays or sets of elements to put into the subparts and markers of a TEMPLATE or into some variables of a FLUIDTEMPLATE.

Question: Can I still use my TemplaVoila FCEs?

Answer: Yes you can, since there is a flexform field, that Grid Elements make use of, although it is not recommended for the creation of new elements. But you will have to provide your own TypoScript setups for the frontend output of these FCEs.

Question: Why is it not recommended to use the flexform for the creation of new content elements?

Answer: The major problem will be to get the content out of these flexform structures when you try to collect it while you are not on the same page. Since there will be no normalised datastructure it can be very hard to just create things like teaser menus or other kinds of collections.

Question: But how should I create new content elements with individual input fields then?

Answer: It's very easy to provide a new CType for the tt_content table with the TYPO3 API since this table already offers you lots of different fields to make use of. Just create the TCA structure and add a new content type via an extension or maybe via extTables.php, provide some TypoScript for the frontend output and – voila – here you go with a new content element.

Question: Is it possible to create new elements based on grid containers?

Answer: Yes – but again this is currently not recommended, since you would have to provide a fully fledged tt_content record for each and every part of this element. We are working on a concept with a drastcially reduced table though, that can be connected using Grid Containers. With this approach it will be possible to create new elements based on basic content snippets like Headline, Image, Text, Table and so on.

Question: Why do you reload the page after D&D actions but the TYPO3 core does not?

Answer: The reason for this decision is the behaviour of the TYPO3 core while working on page content. When an editor is working at some content, others will be notified only, when they open the editing form of that particular content element. While working with D&D drop within a larger editorial team it happens quite often, that people change content of the same page at the same time. Just imagine the mess that could happen without reloading the current state of that page. So as long as there is no locking of content available in TYPO3, we will stick to this behaviour.

ToDo

Migrate the code to be compatible to the way TYPO3 6.0 handles classes, hooks and interfaces.

Deactivate the TYPO3 6.0 default D&D features and reactivate the icon that triggers the Drag In Wizard.

ToDo

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!

Changelog

2.1.1 => 2014-07-29

Fixed: Issue #60661 Wizard crashes in Gridelements 2.1.0

2.1.0 => 2014-07-28

Feature: Added horizontal positioning of children to layout setups via "Group children: Side by side"
Backport: Fixes from master especially regarding namespaces and coding guidelines
Fixed: Fixed problems due to core patch fixing "Movements pollute colPos value of content elements"

2.0.3 => 2014-05-27

Security Fix, PLEASE UPDATE For details read http://typo3.org/teams/security/security-bulletins/typo3-extensions/typo3-ext-sa-2014-008/ ~hh

2.0.2 => 2013-06-17

Fixed: Wrong dependency setting in ext_emconf.php
Fixed: Remove superfluous JS comma that breaks IE8
Fixed: Problems due to restructuring and namespaces
Fixed: JavaScript error for read-only content elements
Fixed: Issue #48484 JavaScript error thrown in IE8

2.0.1 => 2013-05-22

Fixed: Issue #48410 New content element doesn't limit allowed types
Fixed: Issue #48371 Fatal error in page view when running on master / 6.2-to-be
Fixed: Issue #48136 Can't move Element to Position Zero
Fixed: Issue #48134 Click Menu does not work when GE 2.0 is installed

2.0.0 => 2013-05-12

Done: Task #44373 TYPO3 6.x compatibility
Fixed: Issue #48081 Select the Gridelement Layout "invalid value" gridelements 2-0
Fixed: Issue #47612 Folder "itemsprocfuncs" should be UpperCamelCase
Fixed: Issue #47574 Fatal Error: Argument 6 passed to ProcessCmdmap::processCmdmap() is None
Fixed: Issue #47544 Error "Calculated absolute path to t3lib directory does not exist." when opening the grid wizard
Fixed: Issue #47508 PHP fatal on creating new content element because of wrong folder name
Fixed: Issue #47506 PHP fatal in list module when ext:version is not loaded
Fixed: Issue #47497 Folder "itemsprocfuncs" should be camel cased
Fixed: Issue #47218 PHP Error Message
Fixed: Issue #46861 No Icons in Backend for CEs after installing gridelements
Fixed: Issue #46827 Old XCLASS method used
Fixed: Issue #46721 Fatal error on backend_layout update
Fixed: Issue #46678 Missing toolbar icons on mouseover
Fixed: Issue #46459 Fit namespaces of TYPO3 6.0
Fixed: Issue #46394 JS D'n'D errors
Fixed: Issue #46227 Layout wizard not working
Fixed: Issue #44534 The content element wizard always show ALL CEs
Fixed: Issue #44533 Modified TYPO3-Icons
Fixed: Issue #44530 Problem with Drag and Drop
Fixed: Issue #44529 missing icons and wrong color

1.5.2 => 2014-07-26

Fixed: Fixed problems due to core patch fixing "Movements pollute colPos value of content elements"
Fixed: Fixed wrong path to class in XCLASS section
Fixed: Issue #59689 Content elements are moved accidentally to column -2 when backend layout is updated

1.5.1 => 2014-05-27

Security Fix, PLEASE UPDATE For details read http://typo3.org/teams/security/security-bulletins/typo3-extensions/typo3-ext-sa-2014-008/ ~hh

1.5.0 => 2014-05-03

Fixed: Issue #58326 Restrict language selectors of children to match the language of their container
Fixed: Issue #53951Translated copies of children being put into the wrong container
Fixed: Issue #48863 Error in languages mode when drag&drop CE
Fixed: Issue #57502 "Hide column headers", which should not remove dropzones and icons
Fixed: Issue #54961 Respect enable fields for child translations.
Backported: Fix for containers with language "all"
Fixed: Issue #33884 Missing children with sys_language_overlay = 0
Fixed: Issue #49053 Broken insertion of copies via column header icons and small CGL fixes
Fixed: csvColumns can be 0, so the check must be against an empty string
Backported: Fixes from 3.0 master
Fixed: #50094 Translation problems and problems with paste icons
Fixed: #44243 Show only translated subset of children for language "all"
Fixed: locallang_db.xml
Added: Feature alias field to get IDs independently from autoincremented UIDs
Merged: Fixes and features of the 2.x branch
Fixed: Frontend output of flexform sections
Fixed: Avoid PHP warnings
Reverted: Latest revision changed file of a tag instead of trunk - reverted and applied to the correct file again
Backported: Fixes while moving or importing elements
Fixed: Broken "get me a copy from another page" icon behaviour

1.4.1 => 2013-05-09

Fixed: Issue #44243 Content not shown in FE without default language of it
Fixed: Issue #32784 New CE wizard – be_group
Fixed: Issue #47418 copy Element out of a gridelement "No valid ajaxID parameter given"
Fixed: Issue #46397 Kopieren nicht möglich
Fixed: Issue #46868 Allowed params is not respected if the colPos is the same
Fixed: Issue #46944 Allowed = * not working in BE-Layout
Fixed: Issue #46331 Paste icon is showed when files are in clipboard
Fixed: Issue #46312 CSS bug if TYPO3 is in subdirectory
Fixed: Issue #47415 Filtering of allowed CTypes fails due to error in sql syntax
Fixed: Issue #46408 "Paste Copy" and "Paste Reference" under pagemodul languageview not possible
Fixed: Issue #46175 t3lib_utility_Math not available in TYPO3 4.5
Fixed: Issue #46946 Creating new CE after an existing element failes
Fixed: Issue #47256 Problem with a foreach in tx_gridelements_view
Fixed: Issue #46766 Drag&Drop function not working after update to svn version 4572
Fixed: Issue #46254 Version number in em_conf.php in v1.4.0 tag incorrect
Fixed: Issue #46175 t3lib_utility_Math not available in TYPO3 4.5
Fixed: Issue #44784 required file wrapperForT3libBeFunc.php in Tests directory

1.4.0 => 2013-03-07

Fixed: Issue #46060 Allowed param in backend_layout is not respected
Fixed: Issue #46027 tx_gridelements_layoutsetup and Singleton
Fixed: Issue #45912 Save flexform when creating gridelements CE with drag&drop feature
Fixed: Issue #45911 "Invalid value" when adding a gridelements CE in another gridelements CE
Fixed: Issue #45768 Wrong position of icons
Fixed: Issue #45650 Content element type select shows "INVALID VALUE"
Fixed: Issue #45594 No drop zone when create a grid CE with the new content icon at the top of the page
Fixed: Issue #45593 cType is lost when adding new element from the "New record after this one" icon
Added: Feature #45592 Add an option to show empty column in frontend
Fixed: Issue #45546 "Copy content from another page" Bug
Fixed: Issue #45461 Column order is broken in frontend
Fixed: Issue #45385 Content element type select shows "INVALID VALUE" after installing gridelements
Fixed: Issue #45344 TCAdefaults not taken into Account
Fixed: Issue #45229 Collapse icon for long tables doubled

Fixed: Issue #45080 Allow to xclass existing xclass
Fixed: Issue #44782 Gridelements removes all CTypes
Fixed: Issue #44780 MySQL error at opening an CE
Fixed: Issue #44707 Backend List View on page with Grid FCE List View Blank
Fixed: Issue #44486 New Content Element - select type [INVALID VALUE ("text")]
Fixed: Issue #44437 Problems when configuring more than one grid as "not assigned"
Fixed: Issue #43553 Database is not extended upon installation within V 6.0.1dev
Fixed: Issue #43202 child elements in grid containers, not visible in workspace
Fixed: Issue #42675 CEs not shown in Web -> List
Fixed: Issue #42637 Gridelements disable alle BE Item previews...
Added: Suggestion #42494 Performance: Add index to tt_content::tx_gridelements_container
Fixed: Issue #42137 cObj:parentRecordNumber remains 1
Fixed: Issue #41937 Wrong cell insertion for colPos > 255
Fixed: Issue #41588 Errors in documentation
Added: Feature #41390 Dutch translation locallang_db.xml
Added: Feature #41342 "Language"-View
Fixed: Issue #40875 FE, Call to undefined method tslib_cObj::getCurrentTable
Fixed: Issue #40839 Fileupload in Flexform stores absolute filepath
Fixed: Issue #40049 tx_gridelements_TCEmainHook->getAvailableColumns throws errors
Added: Feature #40008 Flexform input can be file, use this or the raw input
Added: Feature #40007 accept include typoscript file references
Added: Feature #40006 Allow rendering of content using Fluid
Added: Feature #39887 Semantic grid by awareness of content
Fixed: Issue #39693 Exception from tx_gridelements_drawItemHook when reference is deleted
Added: Feature #39672 flexform typoscript example
Fixed: Issue #39601 new_content_el Wizard does not work if TYPO3 is installed in subfolder of webroot
Fixed: Issue #39333 Class tx_gridelements_layoutsetup does not exist
Fixed: Issue #39259 Hidden content element can not be moved with drag and drop
Fixed: Issue #39257 JS error with copy/paste, when backend column is "Nicht zugewiesen"
Fixed: Issue #39251 Images fails in 6.0
Fixed: Issue #39094 col Pos gets lost after cut and paste
Fixed: Issue #39064 "Top Level Layout" has no effect while d'n'd
Fixed: Issue #38943 Gridelement column lost on element edit with no right on the field colpos
Fixed: Issue #37999 Integration of CSH-File ominous
Fixed: Issue #37883 Unable to un-hide CE after translation in gridlement
Fixed: Issue #37878 Backend: Column relation lost after translation
Added: Feature #37876 Backend: Pagemodul language view
Fixed: Issue #37486 Conflict with foreign_table and itemsProcFunc
Fixed: Issue #37371 Drag'n Drop and Workspaces
Added: Feature #37350 Get a reference to content from a given page
Fixed: Issue #37232 Rendering problem grids ans typoscript
Added: Feature #36948 Make nested content elements collapsible
Fixed: Issue #36910 Problems with translation
Fixed: Issue #36803 possible to define a storagepage for the gridelements
Added: Feature #36725 Improve rendering
Fixed: Issue #36551 Assignment of CE grid elements to grid container not working
Added: Feature #36332 Default 'NEW' header for grid element
Fixed: Issue #36330 Warning in page module with TYPO3 4.7
Fixed: Issue #35997 Copy & paste into grid columns does not work for translated elements
Added: Feature #35994 Have the page jump to the drop position after reloading
Added: Feature #35967 Make path to skin CSS configurable in EM
Fixed: Issue #35958 Error Drag & drop in 6.0
Added: Feature #35948 Implement a "get me a copy from another page" function
Added: Feature #35946 Resize the drop zones to make them accessible
Added: Feature #35670 Add flexform configuration as a file
Fixed: Issue #34109 PHP Warning: strcmp() expects parameter 1 to be string
Added: Feature #33940 Multiple selection for "Allowed CE"
Fixed: Issue #33329 Content not visible in backend or frontend grid after creating it in the backend grid
Added: Feature #32388 Sorting of CE
Added: Feature #30227 Show grid elements as folders containing elements when in list view

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.