



DECO: GRID-BASED LAYOUTS

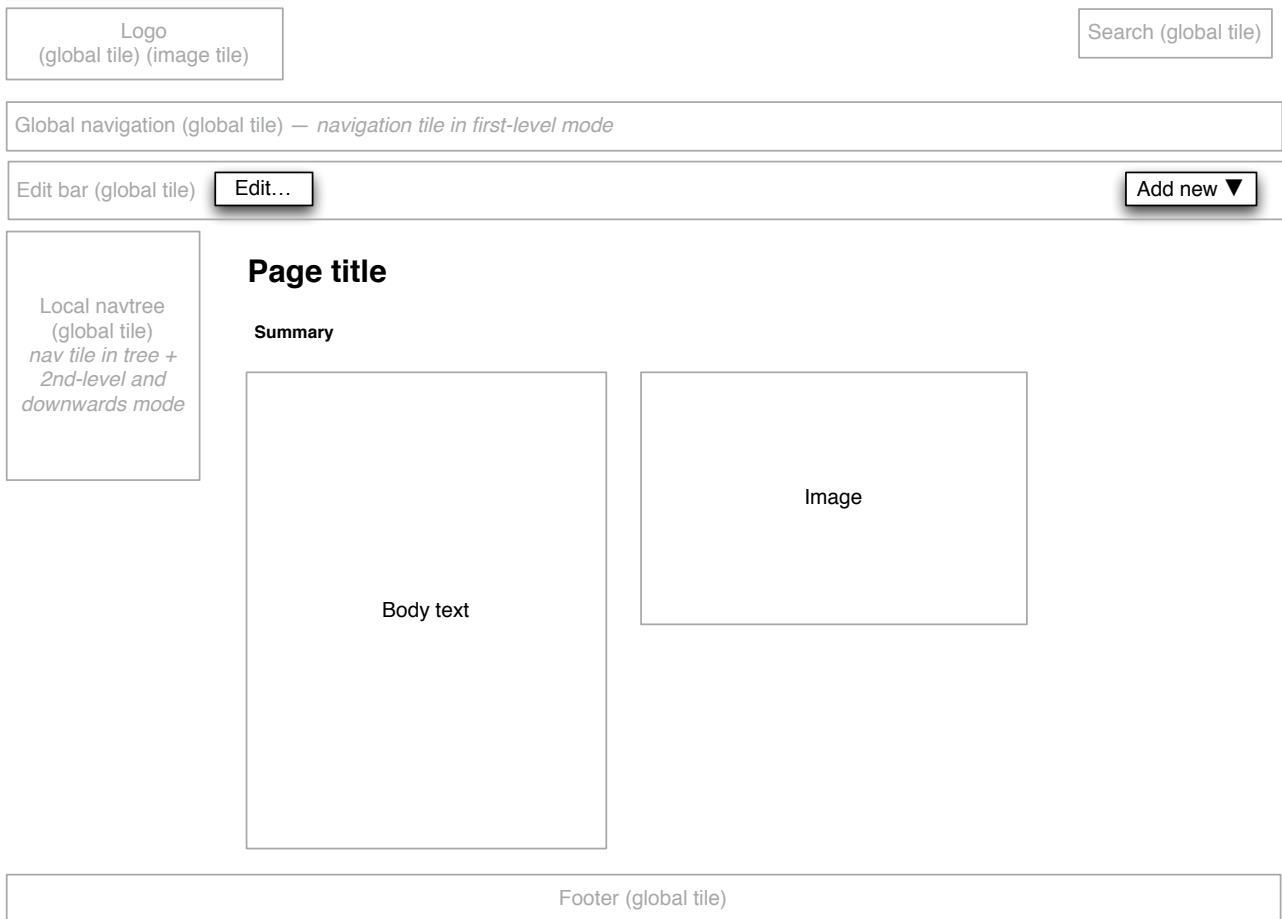
Martin Aspeli, Geir Bækholt, Laurence Rowe, Alexander Limi

Dec 14, 2008

USER INTERFACE

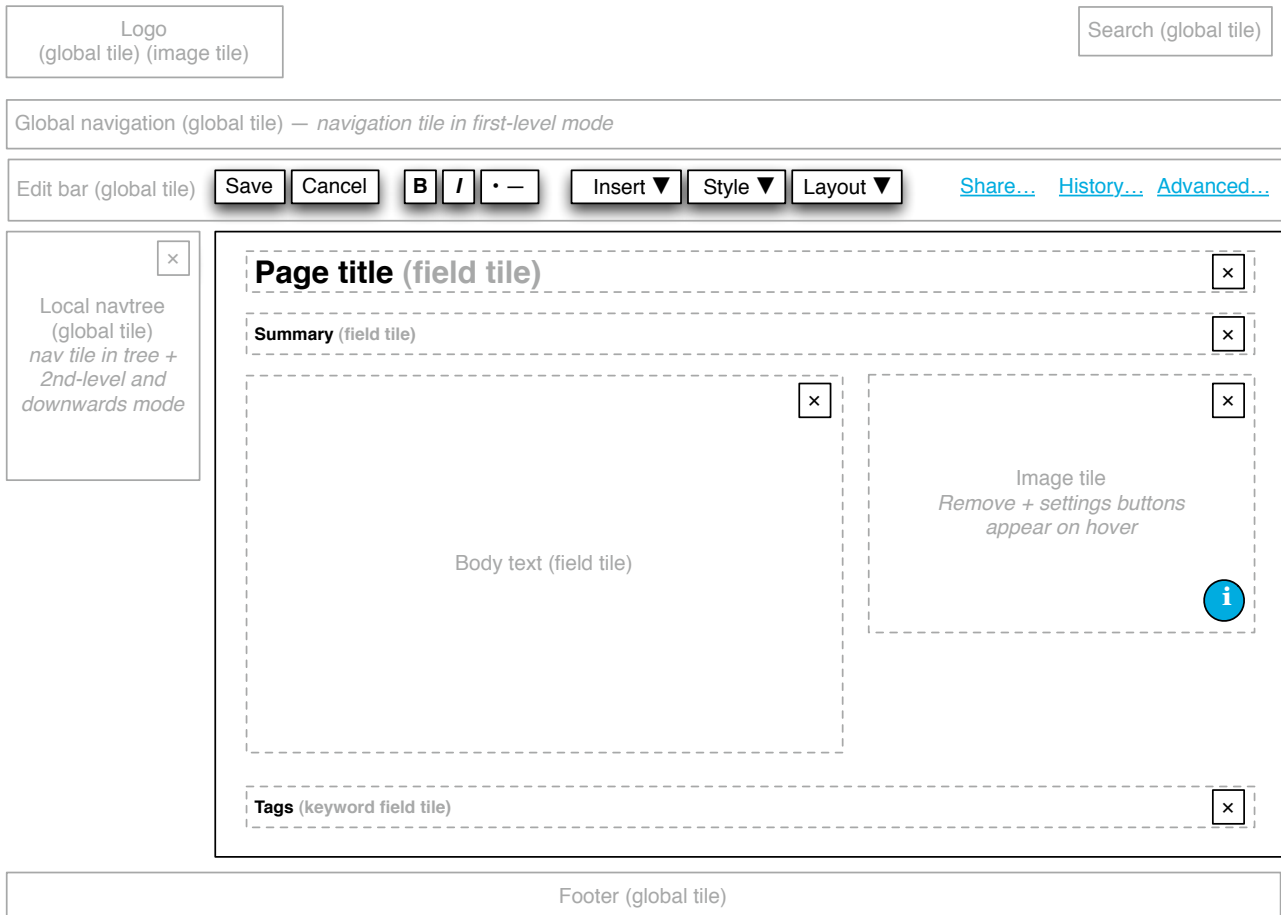
This section describes Deco from a user interface perspective. It demonstrates how the concepts of grid-based layouts and tiles are experienced by the user.

Viewing a page



- This is the new simplified UI, only contains Add and Edit.

Editing a page that supports layout mode



Logo (global tile) (image tile)

Search (global tile)

Global navigation (global tile) — navigation tile in first-level mode

Edit bar (global tile) Save Cancel B / •— Insert ▼ Style ▼ Layout ▼ [Share...](#) [History...](#) [Advanced...](#)

Local navtree (global tile) nav tile in tree + 2nd-level and downwards mode

Page title (field tile)

Summary (field tile)

Body text (field tile)

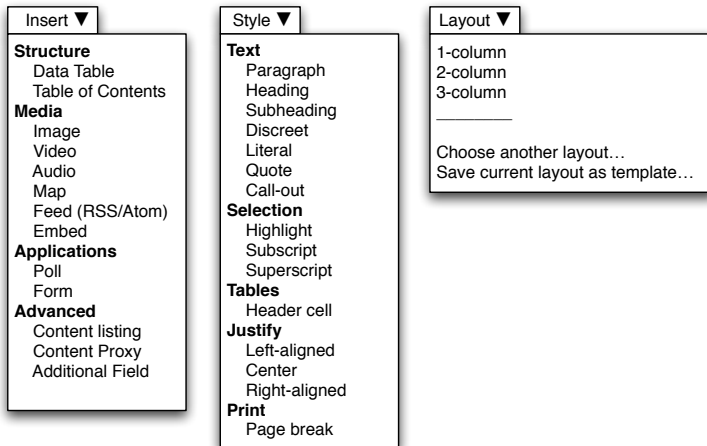
Image tile Remove + settings buttons appear on hover

Tags (keyword field tile)

Footer (global tile)

- Clicking the “Edit” button acts as a toggle, and the layout is kept the same, but with the editable parts indicated with a dashed outline.
- Navigation:
 - The cursor is auto-focused inside the first field (which is technically speaking also a tile).
 - If you click inside any of the other tiles that are text edit fields, you are able to edit them directly.
 - If you click outside of the editing area, the page is in layout mode, and you can drag/drop the tiles.
 - To get back into editing mode, you double-click any of the tiles. For text edit fields, you are back in edit mode, for things like the image tile, you get the settings dialog.
 - We’ll most likely add an icon/button that you can click to enter edit mode too, since I don’t like relying on double-click — although it’s fine as a shortcut.
 - The “Close” and “Info” icons show up when hovering over a tile that supports them.
- Note that you can remove any page element. These will then show up in the “Advanced” screen, and if you want them back in the layout, you use the “Insert Field” option (covered later).
- You can also “blacklist” global tiles, e.g. if you wanted to get rid of the navtree on this particular page. This is still undergoing refinements wrt. how we handle it in the UI — i.e. we might handle it in the Layout menu options instead. Taking this to its logical conclusion, you could even manage “splash pages” this way, by blacklisting everything but a few elements.

Inserting tiles



- To insert additional tiles, you use the Insert menu.

Available tiles

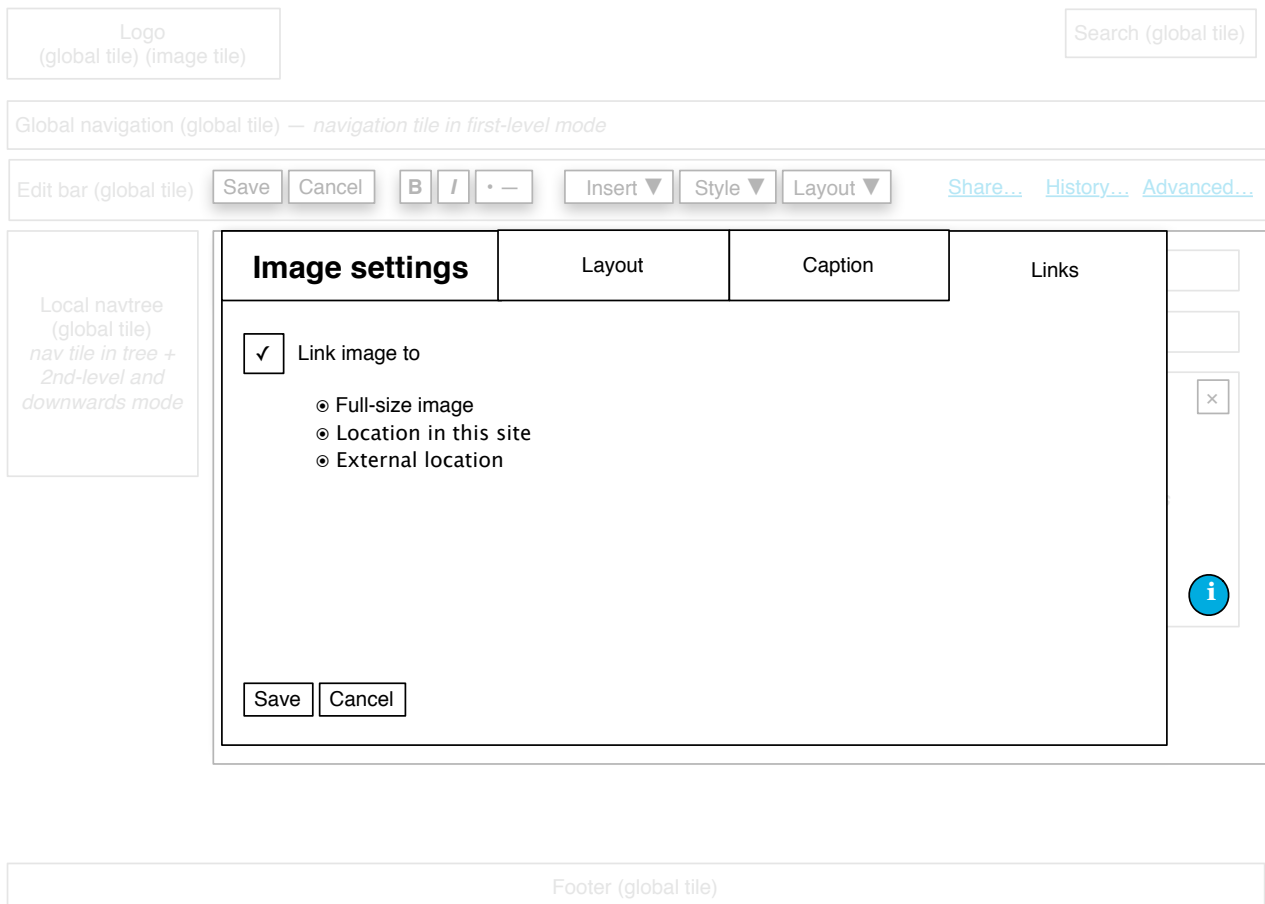
(some of these might only be visible to non-newbie users, we may introduce a Newbie/Novice role to help with simplifying the UI in 4.0)

- **Data Table** — Your plain old average table. The reasoning for moving it to a tile is that table editing in every browser is different, and always has really bad UI. This way, we can give tables a “Settings” page to add in more control. The actual editing works the same way, we just try to move most of the things you can change with a table to a Settings panel instead of having 4x4 pixel icons like in the Mozilla edit controls.
- **Table of Contents** — An always-updated Table of Contents tile. This uses JS like the current Table of Contents, but we might be able to add some more options — deciding how many levels of headlines it should care about, for example.
- **Image** — Similar to turning tables into tiles, there are good reasons for turning images into tiles too. Defining size, captions and copyrights, as well as options for linking to a full-size version of the image, to a location in the site, or to an external URL. I’m sure there are more.
- **Audio/Video** — This inserts a tile with a player, something like collective.flowplayer for self-hosted audio/videos. It also supports pasting in `<embed>` or `<object>` tags, as well as the HTML5 tags `<video>` and `<audio>`.
- **Map** — This inserts a Google Maps tile. Some integration with the Location field would be cool.
- **Feed** — Inserts a tile that takes its content from an RSS or Atom feed, similar to the current RSS portlet. It might make sense to make this a mode of the Content Listing tile instead
- **Embed** — This is a catch-all tile that simply accepts an embed statement. The reason this is useful is that most sites use this as their way to add content from external sites, and they always call it “embed”. YouTube, Maps, Widgets/Gadgets, etc.



- **Applications: Survey/Poll/Form** — This is just an example to illustrate how third-party tiles — say, a poll product — would integrate with the new layout model. It would have its own settings, where you would configure the poll questions, single/multiselect, etc.
- **Content Listing** — This is the new way to handle both “Collections” and “Folder listings”. We’re most likely getting rid of the Folder concept, since it was only used for attaching particular views on a set of objects in the first place. Under the new approach, all types would have the ability to contain other objects (note that I don’t want to ever see the word “folderish” ever again, use container-like or container :). A content listing can either list all the objects under itself, or anything you can configure using a query-builder similar to the one you have in Trac custom reports, Mac OS X Finder, iTunes or similar. This query building is done in the tile’s Settings page.
- **Content Proxy** — This is the way to display the same content in multiple locations. It’s a simple tile that just displays the content from a different page in the site inside a tile. That content is not editable, but there is of course a way to open the original page from the tile, so it can be changed (if you have edit permissions).
- **Additional Field** — This is where you can grab any field from the type definition — say, “Location” or “Contributors” — and add it to the page as a “field tile” similar to how Title and Description work by default. This is also how you build pre-defined layout templates, once you have added the fields you want to the page, you can save it as a template. A classic example would be to create a Press Release template from a standard Page — the only difference being a Location and EffectiveDate field in the beginning of the text, and maybe a Contributors field at the end.

Configuring tiles



Logo (global tile) (image tile)

Search (global tile)

Global navigation (global tile) — navigation tile in first-level mode

Edit bar (global tile) Save Cancel B / • — Insert ▼ Style ▼ Layout ▼ Share... History... Advanced...

Local navtree (global tile) nav tile in tree + 2nd-level and downwards mode

Image settings Layout Caption Links

Link image to

- Full-size image
- Location in this site
- External location

Save Cancel

Footer (global tile)

- Every tile with the exception of the text edit / field tiles is likely to have a settings page.
- There's an *i* icon that will bring up the settings for a given tile. The Remove and Settings icons show on hover, but double-clicking the tile will also bring you to the settings page.
- If the tile has several aspects to its settings, it supports this via an approach similar to the fieldsets we use in 3.x.

Saving layout templates

- When you save a Layout Template, it gets filed under the category your page is using.
- Categories can have a default layout template associated with them.
- We might want to restrict the layout saving functionality to the non-newbie users.

“Advanced” screen

Logo
(global tile) (image tile)

Search (global tile)

Global navigation (global tile) — *navigation tile in first-level mode*

Edit bar (global tile)
Save
Cancel
B
/
• —
1 —
Link
Insert ▼
Style ▼
[Share...](#)
[History...](#)
[Advanced...](#)

Local navtree
(global tile)
*nav tile in tree +
2nd-level and
downwards mode*

Categorization	Ownership	Dates	Settings
<div style="display: flex; align-items: center; gap: 10px;"> <input checked="" type="checkbox"/> Exclude from navigation </div>			
<div style="display: flex; gap: 10px;"> Save Cancel </div>			

Footer (global tile)

- All fields that are not in the main layout are on the “Advanced” screen that is a modal dialog that you bring up by clicking the link.
- This is where the more esoteric metadata and page settings are located, as well as any fields that have been removed from the main layout.

Adding additional schema fields to the layout

- Use the Insert → Additional Field menu.

Changing layouts

(Unfinished: What happens with existing layout? Possibility of choosing a layout that overrides the default. No panel merging!)

Categories and keywords

- Keywords are (finally!) renamed to tags, and use an autocomplete widget similar to `plone.formwidget.autocomplete`.
- We introduce the concept of “categories”, which allows people to do useful things without necessarily building a new type for everything. This helps us avoid type explosion, which is a problem in most Plone sites. This also removes the need for most of the use cases for approaches like subtyping.
 - Categories can be used as listing criteria (navigation, content listings).
 - An object can only be in one category (I think ~limi ;-)
 - In the Press Release example mentioned earlier, this would likely be a Page type with the category “Press Release” and an associated saved Layout Template.
 - Another example: If you want to create a Blog, you’d probably just have a Content Listing that listed all Pages with the Category “Blog entry” in reverse chronological order.
 - When you save a Layout Template, it gets filed under the category your page is using.
 - Categories can have a default layout template associated with them.
 - Basic rule is: You create new types when you need different workflow/permissions, if you need additional fields that need to be indexed/queryable. If not, a Category with an associated layout will probably give you all you need.
- With regards to restricting types in particular locations, I envision that it should be possible to set the default Category that something shows up as when you add it, but not enforce this. If you want to enforce what can be added in a certain container, that might be better suited for a dedicated type. I’m open for suggestions on this, though.

Editing a content object that uses forms-based editing

For types that don't opt in to the new layout model, there will be few or no changes compared to Plone 3. The main one is moving the fieldsets to the "Advanced" screen.

Building navigation

Under the new approach, there is no such thing as a Folder type, and Pages are containers too. So how do you build structure in the new approach?

One way to solve this might be to do what Google Sites does. When you add a new page, it asks you the following on object creation, before you get to the edit page itself:

- What the title of the page is.
- The URL (ID) is generated and shown, and you can edit it immediately if you want.
- Where it should be located:
 - It defaults to same level as the page you came from.
 - You can also choose to put it "under" (inside) the Page, thereby nesting it a level deeper.
 - You can also use a picker to choose a different location entirely.

Other interesting implications of this approach is that they ask what kind of page you are adding (their equivalent of types). Expanding this to Plone, we could ask for:

- Type
- Category

That way, the "Add new..." menu could be reduced to a single button instead of a pulldown, and it's also cheaper to compute the page, since we don't have to check for addable types — then again, I'm not sure if it will make a noticeable performance difference. And yes, this is what CMF used to do. :-)

Editing site layout



- You get to the Site Layout screen via Site Setup.
- It gives you an edit bar at the top, and allows you to add tiles that will apply globally, as well as defining where the main content area is located.
- It uses the exact same grid system and layout engine as the edit screens on a type, but doesn't concern itself with what's inside the content area — this is a simple placeholder.
- The tiles have settings, can be removed (the X) and added (using the “Add new tile” menu) — and can be dragged and dropped for positioning.
- Note that you can also position UI elements like the Edit bar, so if you prefer that on top of the page, it's a simple drag and drop away.

Section-specific site layouts

It can be useful to have certain layouts apply in certain sections of the site by default. As an example, consider the Support section on plone.org — we might want to make this section not have any navtree, and there might be a sections of the site where we want to get rid of pretty much all the global tiles and just have a splash page with a graphic and a few calls to action.

This is handled by using the same layout mechanism as the Site Layout. Notice how we can add “Section layouts” in the menu that displays what you are currently editing. These would be associated with a particular location in the site, and usually inherit to all objects under it — although it should be possible to choose either.



Missing bits

Stuff that isn't in the mock-ups and/or descriptions yet:

- Categories aren't handled in the current UI mock-ups — suggestions appreciated, although I have some ideas I want to try.
- Handling of state changes in layout mode. (One option is to ask on save?)
- Change note + significant edit indicator (might also be done on save)
- Description of the new Draft saving mechanism
- Pulldown for multilingual (“translate into Swahili”)
- State change without editing (current proposal is to integrate it into the by-line, as in “published ▼ by Jon Stahl at 08:46.”)

FREQUENTLY ASKED QUESTIONS

What if I want a variable-width layout?

The grid system we are currently looking at — Emastic — does offer some level of support for variable-width layouts, but more experimentation is needed before we know whether we'll pick that over the de-facto standard in grid layouts, Blueprint. We might have to be as opinionated as to say that you can't use the new layout model with fluid layouts, which I'm fine with — but I know it will lead to complaints. We'll see. :-)

What about accessibility?

There's nothing about the new approach that is less accessible than the previous approach in Plone 3. You can always choose to give up the layout model and write HTML/STX/reST/Markdown etc. instead, which is probably a better approach at least for blind people. That being said, you can tab between different tiles when you edit, and it should be possible to get most of it working even from an accessibility perspective.

What about using alternative visual editors?

We'll be using two terms to talk about this, just so we don't get confused:

- **Formatting:** Simple text formatting tasks. Examples: Bold/italics, headlines, lists. Explicitly **not** about things like images, and possibly not even something as complex as tables.
- **Layout:** Adding richer content like images, videos, audio, using "tiles" — essentially the unification of portlets and viewlets. Layout also deal with positioning of these tiles using drag and drop, and is implemented in the DOM, outside of `contentEditable`.

In a nutshell, we're moving as much as we can of the **layout** responsibilities outside of the `contentEditable` (think Kupu/TinyMCE) implementations in the different browsers.

So what does this mean? It means that pretty much any editor can perform the required tasks, since we are de-scoping its responsibilities quite extensively. We may elect to use a stripped-down

How will tiles be customized?

I assume that this will work the same way as any other customization of views, since we need to straighten out that story anyway in Plone 4. GloWorm is a likely candidate for locating the correct view template for the integrator that is trying to do it. I'll let Martin elaborate on what this question means, since he asked it. :)