

Adapt Authoring UI for adding / editing components

Mike and I started tackling the component editing issue in anger today. The main issue we are addressing is how we add and edit components.

Workflow

The flow across key pages should be as follows:

Illustration	Comments
<p>1) Add component 2) Select component type 3) Select placement/size 4) Choose whether to edit straight away or return to structure editing 5) Enter data into component Default view: Mandatory fields only 6) view all data input fields (optional) 7) enter content / make changes 8) save / cancel</p> <p>Page structure editing screen Component selection dialogue Component editing screen</p>	<p>In summary, this means there are three key screens we need to design. These are:</p> <ul style="list-style-type: none">• The page structure editing screen• The component selection dialogue• The component editing screen <p>We discussed the flow and agreed on two items:</p> <ul style="list-style-type: none">• Step 4 would enable the user to choose whether to go into data entry or skip this step for now and carry on with editing the structure• The detailed view for step 5 would show mandatory fields by default. There would be a vehicle to reveal the additional entry fields.

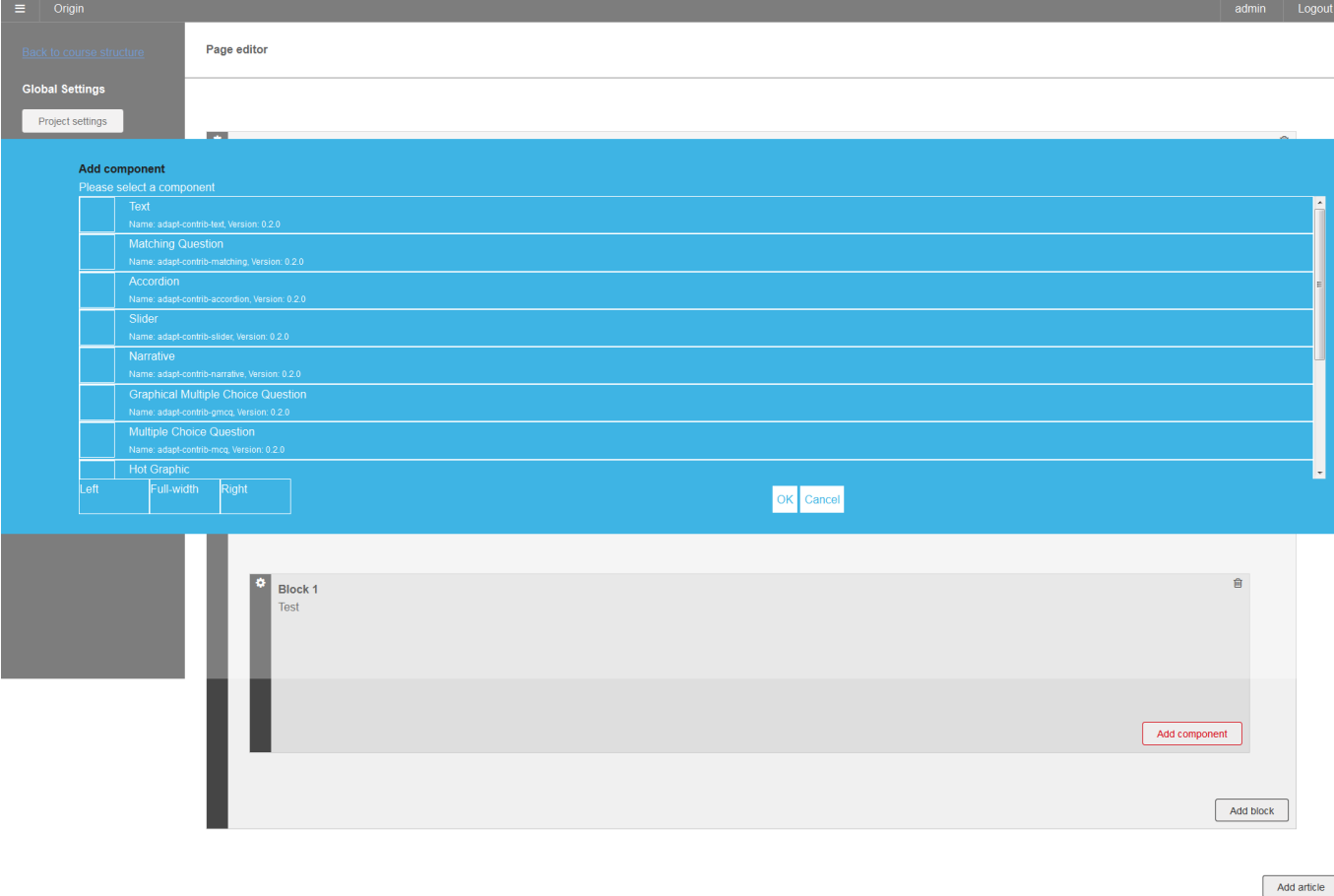
Page structure editing screen

The starting point for these designs is the page structure editing screen:

Relevant screenshot	Comments
<div><div><div><div>≡</div><div>Origin</div><div>admin</div><div>Logout</div></div><div>Back to course structure</div><div><div>Global Settings</div><div><div>Project settings</div><div>Configuration settings</div><div>Manage extensions</div><div>Preview</div><div>Publish</div><div>Close</div></div></div></div><div><div>Page editor</div><div><div><div>⚙</div><div>Article 2</div><div>This is the first article</div><div><div><div>⚙</div><div>Block 1</div><div>Test</div><div><div>⚙</div><div>[Text]</div><div>(Your new component)</div></div><div><div>⚙</div><div>[Text]</div><div>(Your new component)</div></div></div><div>Add block</div></div><div><div><div>⚙</div><div>Article 1</div><div>This is the first article</div><div><div><div>⚙</div><div>Block 1</div><div>Test</div><div><div>⚙</div><div>[Text]</div><div>(Your new component)</div></div></div><div>Add block</div></div><div>Add article</div></div></div></div></div></div></div>	<p>The page structure editing screen is largely in keeping with the designs in the design review project (which is missing some detail).</p> <p>In each block, there is an ‘Add component’ button, which is not visible in the screenshot to the left. Given that there is one of these buttons per block, the user determines the location of where to add the component by choosing the button.</p>

The component selection dialogue

The component selection dialogue is the interim screen, which covers steps 2-4 in the flow above.

Relevant screenshot	Comments
	<p>The component selection dialogue should be implemented as a modal dialogue and cover the following:</p> <ul style="list-style-type: none">• Provide a list of all the components installed in the authoring tool, which the user has access to• The user can cancel out of the dialogue at any time. This would mean no component will be added.• The user selects a component• The user selects whether this is to be added as a full width component or as a single width component. If the latter, the user chooses whether the component displays on the left or right of the block.• The user can then select whether to to:<ul style="list-style-type: none">○ Add the component and return to the page structure editing screen. (If so, default or dummy values are added to the mandatory component data fields)○ Add the component and go straight into the content entry / editing screen. <p>The look & feel shown here is not in keeping with the clean design and will need to be reviewed and incorporated into the design review project.</p>

Illustration

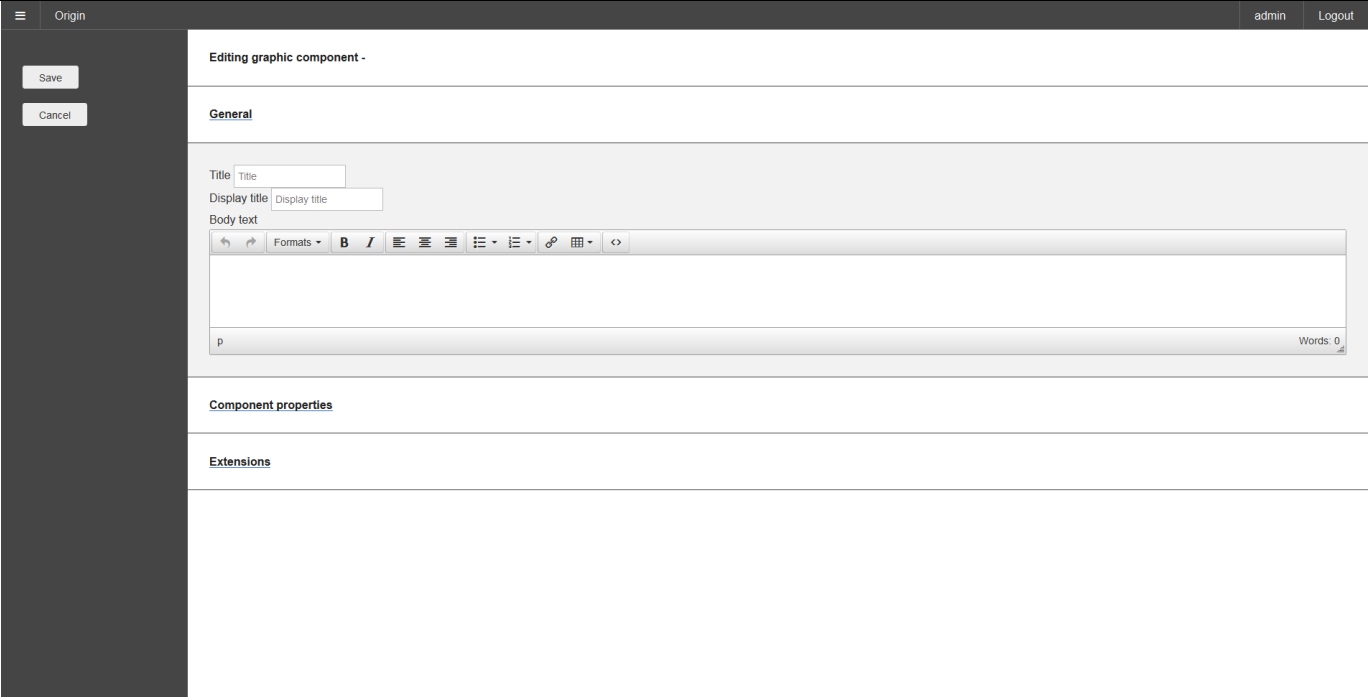
A hand-drawn illustration of a user interface for adding components. The interface is enclosed in a rounded rectangle. At the top left, the text "Add component" is underlined. To its right is a "Search" label followed by a rounded input field containing a magnifying glass icon. In the top right corner of the interface is a red circle with a white 'X' inside. Below the search bar is a list of five component types: "Text", "Graphic", "Narrative", "Slider", and "Graphical MCQ". The "Graphic" item is highlighted with a blue background and has a black arrow pointing to it. To the right of this list is a vertical scrollbar with a hatched pattern. Below the list, separated by a dashed line, is a "Position:" label followed by three buttons: "Left", "Full-width", and "Right". At the bottom, also separated by a dashed line, are three buttons: "Cancel" (outlined in red), "Add and return" (outlined in green), and "Add and edit" (outlined in green).

Comments

See comments on screenshot above

The component editing screen

The component editing screen is where the user inputs data for the content and configuration of the component.

Relevant screenshots	Comments
	<p>Component editing happens in a full page, which behaves like a dialogue. There are several actions available on this page (not represented in screenshot) including:</p> <ul style="list-style-type: none">• Save / cancel• Preview• Close and return to page structure <p>The screenshot shows the editing page. The main content area shows an accordion to separate out types of properties – in this example the General accordion ‘tab’ is open.</p> <p>Separation via accordion control The separation of General, component properties and extensions is derived from the technical structure of the schema files. I believe this is not suitable for separation from a usability point of view. I also believe that the accordion vehicle is not correct and that users would be better served with a long scrolling page and maybe an index to in the sidebar to skip to items.</p> <p>Design actions:</p> <ul style="list-style-type: none">• Add available action buttons to sidebar• Change accordion control in to long scrollable page• Initial view to show mandatory fields only• Add index to sections in scrolling page on sidebar

The screenshot shows the editing page. The main content area shows an accordion to separate out types of properties – in this example, the Properties accordion ‘tab’ is open.

User friendliness of input fields

The data input fields are read in from the component schemas and rendered by the authoring tool. This example shows that the authoring tool uses the variable names from the schema rather than descriptive labels. Using schema data alone, also creates an issue in terms of being able to translate the labels into other languages. Finally, for ease of use, we also ought to add context sensitive help for each data field in order to make the authoring tool as easily usable as possible.

Design actions:

- Add language-string display labels (Ability to change authoring too interface language)
- Add context sensitive help files
- Change input fields to better represent input data type (e.g. longer input boxes, textareas with formatting, file pickers)

The screenshot shows the editing page. The main content area shows an accordion to separate out types of properties – in this example, the Properties accordion ‘tab’ is open.

User friendliness of input fields

The data input fields are read in from the component schemas and rendered by the authoring tool. This example shows that the authoring tool uses the variable names from the schema rather than descriptive labels. Using schema data alone, also creates an issue in terms of being able to translate the labels into other languages. Finally, for ease of use, we also ought to add context sensitive help for each data field in order to make the authoring tool as easily usable as possible.

Design actions:

- Add language-string display labels (Ability to change authoring too interface language)
- Add context sensitive help files
- Change input fields to better represent input data type (e.g. longer input boxes, textareas with formatting, file pickers)

≡

Origin

admin

Logout

Save

Cancel

Editing graphic component -

General

Component properties

Extensions

root

_pageLevelProgress

_jsEnabled

false ▾

In this example, the Extensions accordion ‘tab’ is open.

No additional comments

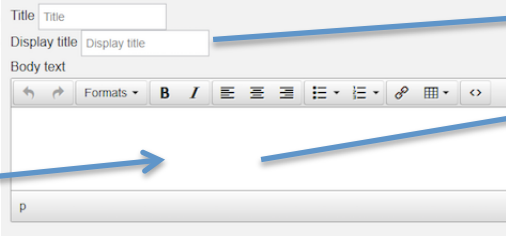

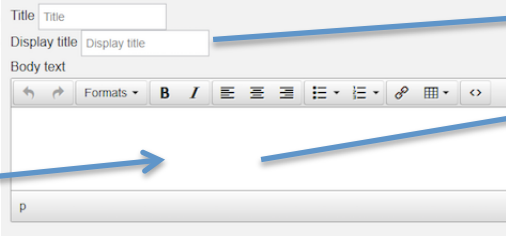

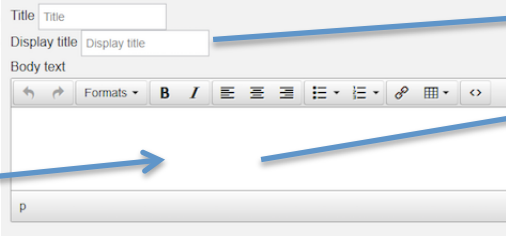

Research into use of schemas for input rendering

In order to provide the correct design input, we have done a review of the data available to the authoring tool. This has included the following:

Review of the existing schema files

There is a hierarchy of schema files, which the editor uses to render the editing interface:

Illustration	Comment
<pre>{ "type": "object", "\$schema": "http://json-schema.org/draft-04/schema", "id": "http://jsonschema.net", "\$ref": "http://localhost/system/tenantObject.schema", "properties": { "_type": { "type": "string",</pre>	<div>basicContent.schema</div> <div>This schema is the highest in the inheritance structure. It applies to all components. Most importantly, this means that all components have the following data entry fields as far as the authoring tool is concerned (highlighted in red):<ul style="list-style-type: none">titledisplayTitle</div>

<pre> "id": "http://jsonschema.net/_type" }, "title": { "type": "string", "required": false, "default": "" }, "displayTitle": { "type": "string", "required": false, "default": "" }, "body": { "type": "string", "default": "" } } } </pre>	<ul style="list-style-type: none"> body <p>To illustrate this...</p> <table border="1"> <thead> <tr> <th data-bbox="1043 252 1579 292">in the authoring tool...</th><th data-bbox="1579 252 2152 292">and the output.</th></tr> </thead> <tbody> <tr> <td data-bbox="1043 292 1579 831">  </td><td data-bbox="1579 292 2152 831"> <p>Imagery helps to tell a story</p> <p>For content that needs a visual effect we have a graphic component. To optimise this for various devices, this component swaps out images based upon your screen size. Giving the user the best experience possible according to their device's spec.</p>  </td></tr> </tbody> </table>	in the authoring tool...	and the output.		<p>Imagery helps to tell a story</p> <p>For content that needs a visual effect we have a graphic component. To optimise this for various devices, this component swaps out images based upon your screen size. Giving the user the best experience possible according to their device's spec.</p> 
in the authoring tool...	and the output.				
	<p>Imagery helps to tell a story</p> <p>For content that needs a visual effect we have a graphic component. To optimise this for various devices, this component swaps out images based upon your screen size. Giving the user the best experience possible according to their device's spec.</p> 				
<pre> { "type": "object", "\$schema": "http://json-schema.org/draft-04/schema", "id": "http://jsonschema.net", "\$ref": "http://localhost/system/basicContent.schema", "properties": { "_classes": { "type": "string", "default": "" }, "_parentId": { "type": "objectid", "required": true }, "_courseId": { "type": "objectid", </pre>	<p>Content.schema</p> <p>This schema inherits from the basicContent.schema, meaning it builds upon it and carries the values the former has already defined as well as the ones listed in the code to the left.</p> <p>This schema does not add any fields that should be input into the authoring tool by the end user. The authoring tool handles the assignment of these data values internally.</p>				

<pre>"required":true, "editorOnly": true } } }</pre>	
<pre>{ "type":"object", "\$schema": "http://json-schema.org/draft-04/schema", "id": "http://jsonschema.net", "\$ref": "http://localhost/system/content.schema", "properties": { "_componentType": { "type": "objectid", "required": true, "ref": "componenttype", "editorOnly": true }, "_component": { "type": "string", "required": true, "default": "" }, "_layout": { "type": "string" }, "_extensions": { "type":"object" }, "properties" : { "type": "object" } } }</pre>	<p>Model.schema</p> <p>This schema builds upon the two previous ones and defines further internal variables, which are not seen or modified in the authoring tool by the end user.</p> <p>For reference, some properties carry an underscore prefix. These properties are values, which are language independent and would not change regardless of authoring tool language chosen.</p>
<pre>{ "type":"object", "\$schema": "http://json-schema.org/draft-04/schema",</pre>	<p>An example component schema</p> <p>In order to get to rendering the input fields for a component, each component</p>

```

"id": "http://jsonschema.net",
"$ref": "http://localhost/plugins/content/component/model.schema",
"properties": {
  "_graphic": {
    "type": "object",
    "required": true,
    "properties": {
      "alt": {
        "type": "string",
        "required": false
      },
      "large": {
        "type": "string",
        "required": true
      },
      "medium": {
        "type": "string",
        "required": true
      },
      "small": {
        "type": "string",
        "required": true
      },
      "title": {
        "type": "string",
        "required": false
      }
    }
  }
}

```

has a schema, too, which builds upon the previous three schemas. The example here is a Graphic component, which is relatively simple. All properties are explained below for reference:

[Adapt-contrib-graphic – properties.schema](#)


This schema is responsible for the following data input rendering interface:

Component properties

Data fields and their meanings:

- `_graphic` – this is essentially a group of data values, which contains the following attributes:
- `alt` – the alternative text for the graphic
- `large` – the relative URL to the large graphic file (desktop display). This would ideally be input via a file chooser dialog
- `medium` – the relative URL to the medium size graphic file (tablet display).
- `small` – the relative URL to the small graphic file (smartphone display).
- `title` – *I am unsure where this is used to be honest. It could be a duplication.*

And an example of this looks like so:

Example JSON data...	and the output.
<pre>{ "_id": "c-70", "_parentId": "b-35", "_type": "component", "_component": "graphic", "_classes": "", "_layout": "right", "title": "Imagery helps to tell a story", "body": "For content that needs a visual effect we have a graphic component. To optimise this for various devices, this component swaps out images based upon your screen size. Giving the user the best experience possible according to their device's spec.", "instruction": "", "graphic": { "alt": "alt text", "title": "title text", "large": "course/en/images/origami-menu- two.jpg", "medium": "course/en/images/origami-menu- two.jpg", "small": "course/en/images/origami-menu- two.jpg" }, "_pageLevelProgress": { "_isEnabled": true } },</pre>	<p>Imagery helps to tell a story</p> <p>For content that needs a visual effect we have a graphic component. To optimise this for various devices, this component swaps out images based upon your screen size. Giving the user the best experience possible according to their device's spec.</p> 

Requirements for data entry / editing

We have defined some basic data input UI requirements:

- Ability to display a descriptive language-string label other than the variable name
- Ability to display a language-string context sensitive help popup per data item
- Ability to display mandatory fields only by default
- Ability to display all data entry fields
- Ability to group related data entry fields in the editor
- Ability to apply basic formatting to relevant data types
- Ability to call up a file-chooser dialogue for assets
- Ability to limit arrays of items to sensible values / what the code can handle via the schema
- Ability to render a data input form field best suited to the data type

Data types and rendering characteristics

Data type	Rendering characteristics
Single line unformatted text string (short)	
Single line unformatted text string (long)	
Single line formatted text string (short)	
Single line formatted text string (long)	
Multi line unformatted text string	
Multi line formatted text string	
Boolean	
Integer	
Decimal	
CSV	
Single select from list	
Relative link URL	
Fully qualified link URL	
Relative asset URL	
(Array)	

Appendix A – Adapt core bundle definition

The core bundle is defined as follows. This identifies the scope of the plug-ins we need to consider for the initial design task

Type	Repository / plug-in	Schemas or datafiles	Status
Core	adapt_framework	course/config.json	Seems to be a data file. Cannot find schema.
		course/<lang>/course.json	Seems to be a data file. Cannot find schema.
		course/<lang>/course.json	Seems to be a data file. Cannot find schema.
		course/<lang>/articles.json	Seems to be a data file. Cannot find schema.
		course/<lang>/blocks.json	Seems to be a data file. Cannot find schema.
Components	adapt-contrib-accordion	properties.schema	Schema in place. Where and how do we express limitations of the array of items?
	adapt-contrib-blank	?	Missing or is this deliberately omitted?
	adapt-contrib-gmcq	properties.schema	Schema in place. [Issues: array limitations, Shouldn't there be a different data type for asset URLs?]
	adapt-contrib-graphic	properties.schema	Schema in place. Shouldn't there be a different data type for asset URLs?
	adapt-contrib-hotgraphic	properties.schema	Schema in place.
	adapt-contrib-matching	properties.schema	
	adapt-contrib-mcq	properties.schema	
	adapt-contrib-media	properties.schema	
	adapt-contrib-narrative	properties.schema	
	adapt-contrib-slider	properties.schema	
	adapt-contrib-text	properties.schema	
	adapt-contrib-textInput	properties.schema	
Extensions	adapt-contrib-assessment	example.json	Seems to be a data file. Cannot find schema.
	adapt-contrib-pageLevelProgress	example.json	Seems to be a data file. Cannot find schema.
	adapt-contrib-resources	example.json	Seems to be a data file. Cannot find schema.
	adapt-contrib-spoor	?	Missing altogether. Incorporated into config.json?
	adapt-contrib-trickle	example.json	Seems to be a data file. Cannot find schema.
	adapt-contrib-tutor	?	Missing altogether.
Theme	adapt-contrib-vanilla	?	Missing altogether.
Menu	adapt-contrib-boxMenu	course/<lang>/contentObjects.json	Seems to be a data file. Cannot find schema.
Developer tools	adapt-cli	n/a	n/a