[CaP CMS Software Architecture]

… The client's CMS

Basic Architecture Overview

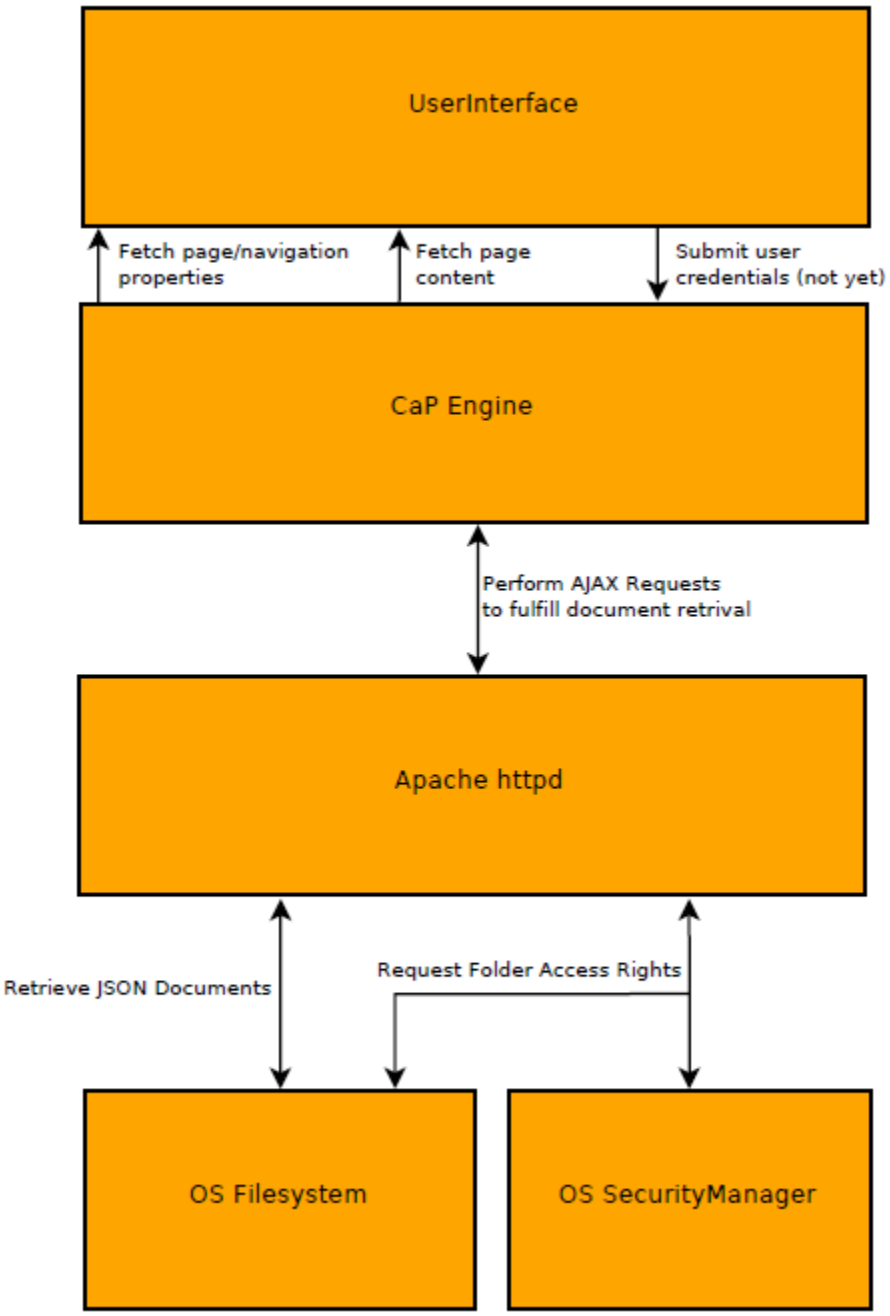

Figure 1: Shows the basic overall architecture of the CaP CMS System

The following section refers to Figure 1 in order to describe the overall architecture with the support of a graphical representation. The architecture will be discussed in top-down manner to keep the explanation structured. User Interface and CaP Engine are completely executed on the client's side.

The **User Interface** will be referred to as UIManager in the following. The UI Manager handles the entire front end management and user interaction based behaviour. Therefore it is located on top of the CaP Engine and uses its Application Programming Interface (API) in order to manage the page's visualization. The layout and component placement, as well as the page's design is UIManager main task. In a future version the components and content is only provided when the right user credentials have been submitted.

The **CaP Engine** itself represents the core of the whole Content Management System (CMS). It is the first script the client's browser retrieves and deals with initial configuration, the invocation of the UIManager, is the component classloader and provides the capability to load data items. Hence all traffic between client and server is initiated by this architectural unit.

On the server side only simple instance of **Apache's** web server is needed to serve the CaP CMS. Although there are several features which need to be configured by the administrator on the server side.
The most important configuration option which needs to be enabled is the auto indexing of directories. This is necessary for the engine to be able to access all entries within a certain directory in order to fetch a list of those items.
The CaP Engine is capable of parsing Apache's standard auto indexing output, but also the *mod_jsonindex* directory representation, which is way faster. A further module for the Apache web server is called *mod_upload,* which enables the CaP Engine to fulfill file uploads. In case the user wants to create articles or for any other post this module needs to handle the incoming data. Unfortunately this module is not working to convenient extend.
All items (containing configuration and payload) are stored as Java Script Object Notation (JSON) files.They are retrieved and served from the server's file system. The User Rights Management is provided by the web server or operating system, but is not used yet.

Detailed Overview

UI Manager

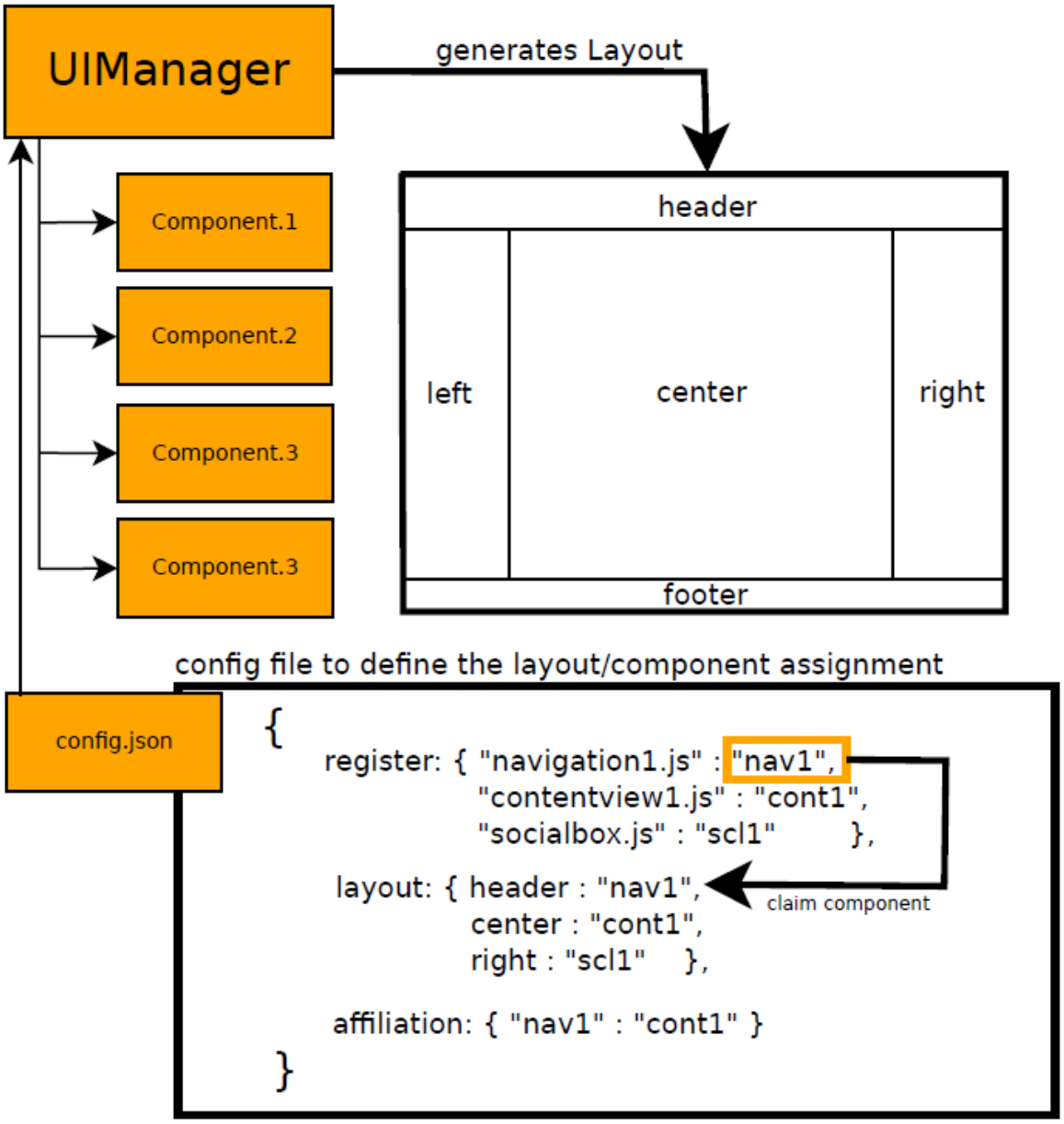

Figure 2: Shows a more detailed overview of the UIManager

Layout Rendering:

The UIManager is configured within the initial config.JSON, which needs to be deployed next to the index.html (CaP Delivery Page). As defined in this configuration file the UIManager creates the page layout. There are four sections available: Header, Left, Right, Center and Footer. It is also possible to declare more complex layout structures in which a new layout is nested. For example one could define a layout like “center.left”, that means a component is placed in the center section on the left side. Additionally there are several properties definable for those layout configurations. (Those specific configuration properties can be read in another documentation file)

```
{
	home:{
		register:{
			"contentviewer.js":"cont1",
			"topnavigation.js":"nav3",
			"navigation.js":"nav1",
		},
		layout:{
			header:{
			component:"nav3",
			},
			center:{
				width:"768px",
				position:"centered",
				center:{
					component:"cont1",
				},
			},
		},
		look:{
			background:"white",
		},
		affiliation:{
			"nav3":"cont1",

		},
		paths:{
			"nav3":"src/resources/ui/navigation/",
			"cont":"src/resources/content/article/"
		},
	},
	paths:{
	components:"src/resources/ui/components/"
	},
}
```

Figure 3: Shows a demo configuration file for the environment “home”

A layout is configured by its layout tag within the configuration file, which is exposed in figure three.

Environments:
In order to generate layouts for several different pages, the “Environment” feature was introduced. As figure three shows the configuration for just the “home” environment, to configure more environments the administrator has to define a new name in the same JSON-level like “home”. Then everything can be reconfigured, what has already been configured in “home”.

Component Loading:
As one can see in figure three the administrator is also able to configure several components, containing things like the navigation bar or any other visualizations the page’s content. Components have to be registered in the “register” section and named as well. The UIManager then creates instances of that component in order to place it once or several times.

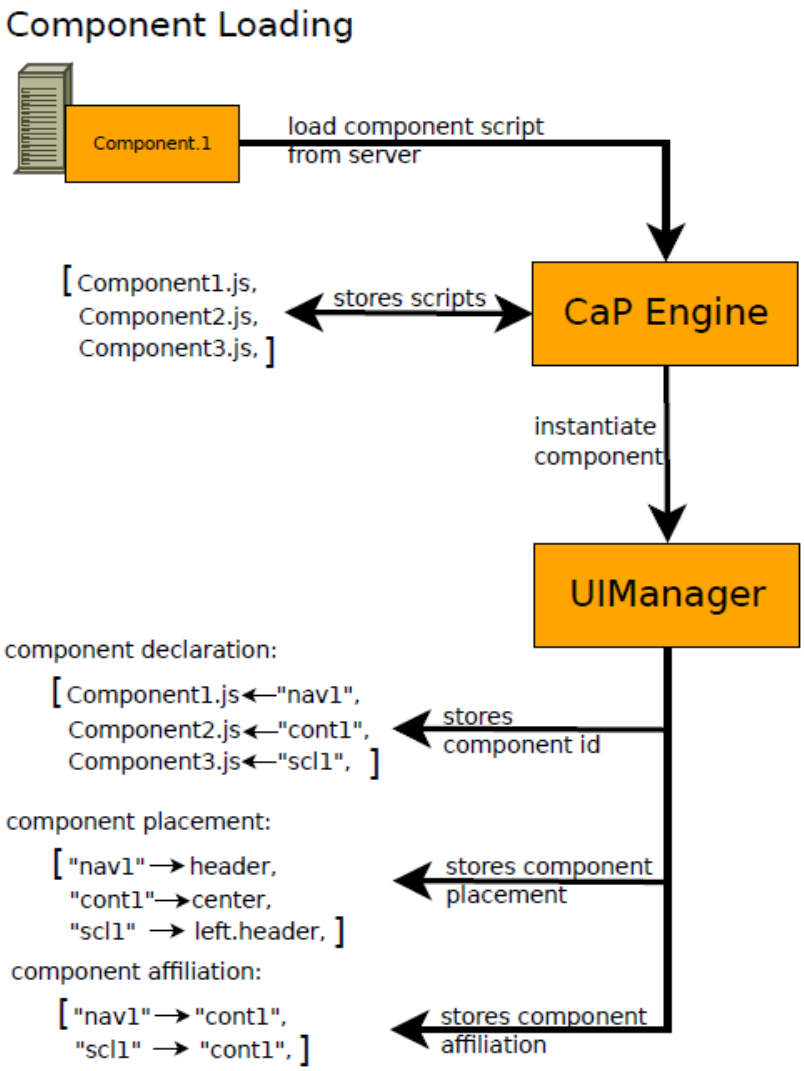

Figure 4: Shows how a component can be loaded into the page

Once the UIManager requests the initialisation of the component it calls the related functions in the CaP Engine and the Engine retrieves the script from the Server (only if needed, not every script is already loaded). This means that the CaP Engine only caches the raw script, but also also calls the class loader instructions. After that the UIManager stores the properties “component id”, “component placement” and “component affiliation”. A full life cycle of a component starts, when it is loaded from the web server, then stored in the CaP Engine and in the end initiated by the UIManager as figure four exposes.

Component Placement:

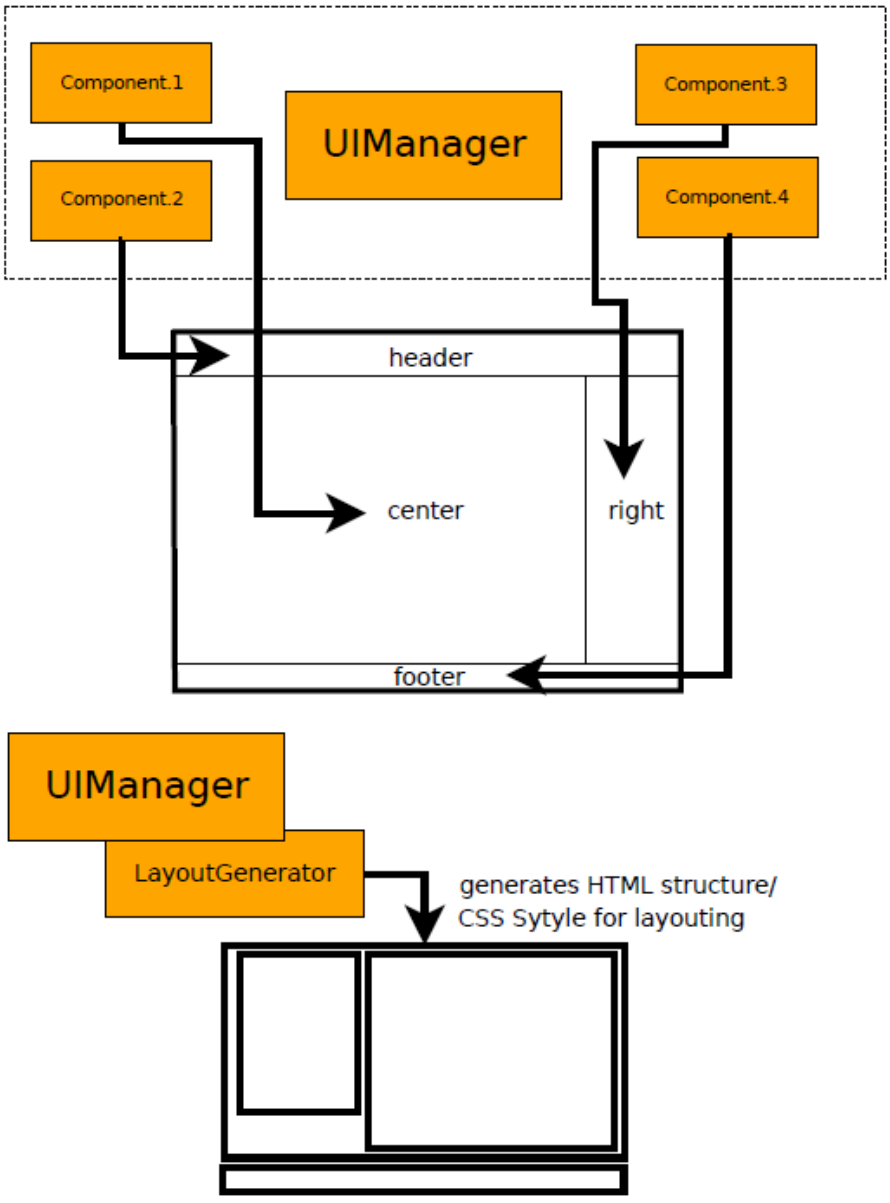

Figure 5: Shows the component placement and generation of HTML and CSS Style layouting

As already mentioned before and illustrated in figure five, the components are placed within the automatically generated HTML structure. It is not the administrators but rather the developer's business to create components, which can be deployed afterwards. There are some specifications and functions, that need to be implemented by the developer, in order to make the component work in the CMS's environment.

Component Relation:
The UIManager provides the capability of component affiliation, which means enabling the component communication among each other. As figure three depicts the component affiliation is configurable by the registered name as well.

CaP Engine

The CaP Engine itself is one the one hand a Java Script Class but on the other hand an architectural unit as well.

Environment handling:
As soon as the UIManager initiates an environment change, the CaP Engine destroys the entire surface and recreates a UIManager with the new parameters.

Provided Classes:
The CaP Engine provides several helpful classes:

- ComponentLoader
- ComponentManager
- Component
- DataItemLoader
- SecurityManager

Those classes can be used by the component itself and the UIManager and are mandatory for the functionality of the CaP CMS.

Security Manager:
The Security is not implemented yet. It deals with User Identification and attempts to post articles to a configured path on the server to determine whether the user is an administrator or not. This necessary to enable the editing of articles and to specify the page's properties. The Security Manager is straightly connected to the *mod_upload* module.