GLOT
Compositional Web Framework

Widget package

At a minimum, **widget package** defines a class in **PHP or NodeJS** with a render method that returns GLOT or HTML code for the given parameters of a widget. The package can also define client-side JavaScript, CSS, and general assets such as fonts and images.

File structure

```
MyWidget
├── assets
│   ├── images
│   │   └── example.svg
│   └── fonts
├── css
│   └── MyWidget.scss
│   └── MyWidget.css
├── dependencies
│   ├── customLibs.json
│   └── externalLibs.json
├── dictionaries
│   ├── markup.json
│   ├── profile.json
│   └── panel.json
├── js
│   └── MyWidget.js
├── settings
│   ├── widget.json
│   ├── params.json
│   └── xyz.json
├── src
│   └── MyWidget.php
├── store
│   ├── assets
│   │   └── widget_icon.svg
│   └── profile.md
├── LICENSE
├── README.md
└── composer.json
```

Examples & Explanation

`css/MyWidget.css`

```
.MyWidget {
    background-color: white;
}
```

The widget package allows for other CSS Pre-Processors enabled via optional plugins. Developers can write stylings in `css/MyWidget.scss` or `css/MyWidget.less` in those case. During development, the Builder generates their minimized css file.

`dependencies/customLibs.json`

```
{
    "js": ["MyWidget.js", "xyz.js"],
    "css": ["MyWidget.scss"]
}
```

This file declares the local libraries that this widget package needs. When rendering the HTML, these local libraries are included in order. Javascript libraries will be placed at the end of the body element and CSS libraries will be placed in the head element.

`dependencies/externalLibs.json`

```
{
    "js": [
        {
            "lib": "jquery",
            "version": "3.5.1",
            "url": "https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"
        },
        {
            "lib": "twitter-bootstrap",
            "version": "4.4.1",
            "url": "https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.4.1/js/bo
        }
    ],
    "css": [
        {
            "lib": "twitter-bootstrap",
            "version": "4.4.1",
            "url": "https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.4.1/css/b
        }
    ],
    "polyfills": ["IntersectionObserver", "Array.from"]
}
```

This file declares the external libraries that this widget package needs. These external libraries will be placed before local libraries. Polyfills are pre-set JavaScript libraries declared in polyfill.io and they will be placed before the normal external libraries.

Different widgets may need same libraries in different order or different versions. Only the latest version of the same libraries will be included. During rendering the HTML, the weights of the dependency between libraries will be calculated and they will be used for including libraries in most reasonable order.

`js/MyWidget.js`

```
class MyWidget extends __Widget {
    /**
     * Called once per page.
     */
    static initClass() {
        // init generic options
    }

    /**
     * Called once per object instance in a page.
     */
    render() {
        console.log(this.holder.id);
        console.log(this.options);
    }
}
```

`src/MyWidget.php`:

```
namespace X\Y;

/**
 * Example entry-point class for the component.
 */
class MyWidget extends Widget
{
    public function render($data, $params)
    {
        // Add a call to the JS 'render' method into the "document ready"
        // event of the webpage. It does nothing if there is no JS code.
        $this->initJavaScriptWidget($params, 'render');

        return ['tag' => 'div', 'data' => $data];
    }
}
```

`settings/widget.json`

```
{
    "version": "1.0.44",
    "name": "MyWidget",
    "label": "My Widget",
    "icon": "widget_icon.svg",
    "storeImage": "widget_card.svg",
    "storeSlides": [],
    "description": "Description of the widget",
    "category": "",
    "tags": [],
    "role": "",
    "childConstraints": {
        "mandatory": [],
        "optional": [],
        "exclude": [],
        "disallowChildren": "1"
    },
    "parentConstraints": {
        "mandatory": [],
        "optional": [],
        "exclude": []
    },
    "implementedConstraints": ["navbar_child"],
    "needRefresh": true,
    "publishedTime": 1600824350
}
```

Head over to the Basic settings for more information of properties.

`settings/params.json`

```
[
    {
        "name": "label",
        "type": "text",
        "label": {
            "en": "Label"
        },
        "editor": true,
        "live": true,
        "linkOption": false,
        "notranslate": false
    }
];
```

Widget parameter schema

Parameter type	Properties	CSS
text		
simpleText		
href.json		
dropdown	options, multiSelect, extendableOptions, privilege,	
switch		
media		
file		
simpleNumber		
number		
slide		
submenu		
table		
font		
systemPanel		
color		yes

The widget parameters have types. Some types are localized so that the **params** argument received by the **render()** method might not be the same that the value of the params property of the widget.

Explain the schema. No underscore prefixes allows. Those are reserved.