Set up the configuration

PlatformUI Configuration files

JavaScript modules in `yui.yml`

Each component of the PlatformUI application is written as a YUI module. The YUI module system comes with a dependency system which is used in the PlatformUI application. For instance, the PlatformUI application has a module called `ez-templatebasedview` which provides a base view class to ease the usage of a template. This module is a dependency of most of the views in the application and has itself some dependencies, like for instance the view module from YUI. Those dependencies are expressed in the `yui.yml` configuration and this configuration is meant to be extended / overridden in others bundles. For instance:

Excerpt of yui.yml

```
system:
    default:
        yui:
            modules:
                ez-templatebasedview:
                    requires: ['ez-texthelper', 'ez-view', 'handlebars', 'template']
                    path: %ez_platformui.public_dir%/js/views/ez-templatebasedview.js
                ez-loginformview:
                    requires: ['ez-templatebasedview', 'node-style',
'loginformview-ez-template']
                    path: %ez_platformui.public_dir%/js/views/ez-loginformview.js
```

This configuration defines two modules `ez-templatebasedview` and `ez-loginformview`:

- `ez-templatebasedview` requires the modules `ez-texthelper`, `ez-view`, `handlebars` and `template`. The source code of this module is in `%ez_platformui.public_dir%/js/views/ez-templatebasedview.js` on the disk;
- `ez-loginformview` requires the modules `ez-templatebasedview`, `node-style` and `loginformview-ez-template`. The source code of this module is in `%ez_platformui.public_dir%/js/views/ez-loginformview.js` on the disk.

Note that the order of module definitions is not important, `ez-loginformview` can be defined before the `ez-templatebaseview` module even if the latter is a dependency of the former.

Most of the PlatformUI application views use a Handlebars template to generate the HTML markup. In the application the templates are also handled as YUI modules but those modules are special because they are dynamically generated from the regular template files on the disk. For this to work, the YUI module corresponding to a template must have the `type` flag set to `template`. To complete the example above, the template module `loginforview-ez-template` should be defined in the following way:

```
loginformview-ez-template:
    type: 'template'
    path: %ez_platformui.public_dir%/templates/loginform.hbt
```

CSS files in `css.yml`

The CSS files used by the application are also listed in `css.yml`, but in the case of CSS files, it's much simpler as this file just lists the CSS files to load, example:

Excerpt of css.yml

```
system:
    default:
        css:
            files:
                - '@eZPlatformUIBundle/Resources/public/css/views/field.css'
                -
'@eZPlatformUIBundle/Resources/public/css/views/fields/view/relation.css'
```

Injecting the extension custom configuration files

To be able to extend the PlatformUI application configuration, the extension bundle has to set up its own configuration handling. For this, you only have to tweak the extension class generated in the bundle in the first step to load the extension bundle's `yui.yml` and `css.yml` and to process their respective content with PlatformUI's configuration. The default empty `yui.yml` and `css.yml` also have to be created so that eZ Platform is still usable.

If you want to get more details on this operation, you can read the How to expose SiteAccess aware configuration for your bundle cook book page.

After this, you just have to make sure the bundle has a `Resources/public` directory to install the assets with the following command:

```
$ app/console assets:install --symlink
```

Results and next step:
The resulting code can be seen in the 2_configuration tag on GitHub (the interesting part is the EzSystemsExtendingPlatformUIConferenceExtension class), this step result can also be viewed as a diff between tags `1_bundle` and `2_configuration`.

After these two first steps, PlatformUI should remain perfectly usable and unchanged. The next step is then to tweak the PlatformUI Application routing to start implementing the new feature.