Configure the navigation

Tutorial path

PlatformUI Navigation Hub

As written in the PlatformUI technical introduction, the Navigation Hub gives access to 4 navigation zones which have a set of navigation items. Each Navigation Item is actually a View in the Navigation Hub which can generate one or more links in the menu. Most Navigation Items can even be seen as a View of a given application route. A Navigation Item View is also responsible for handling its *selected state*. This means that the Navigation Items are notified when the application matches a new route and the view can then decide to react accordingly.

PlatformUI comes with 3 different implementations of Navigation Item. They all generate a link to a route with a given anchor text and they differ by the ability to check if the newly matched route in the application is the route they represent:

- the base implementation is `Y.eZ.NavigationItemView`. When the matched application route changes, it sets its selected state if the navigation item route name matches the name of the new matched route in the application
- the `Y.eZ.NavigationItemParameterView` implementation adds a check on a route parameter. So to appear selected, the route names must match and a given route parameter should be the same in both the application matched route and in the route the view is representing
- the `Y.eZ.NavigationItemSubtreeView` also adds a match on a route parameter, but in this case it considers the `id` route parameter and checks whether the matched id in the application route is a descendant of a given Location id.

The default structure of the Navigation Hub is defined in the Navigation Hub view service attributes.

Adding a new navigation item

Plugin for the Navigation Hub view service

Since the menu structure is defined in the Navigation Hub view service, we need to write a plugin for the Navigation Hub view service. Again, we'll create a module that will define a plugin. So the first thing to do is to declare our new module in `yui.yml`:

yui.yml

```
ezconf-navigationplugin:
    requires: ['ez-pluginregistry',
'ez-viewservicebaseplugin'] # ez-viewservicebaseplugin
instead of plugin, base for plugins for view services
    dependencyOf: ['ez-navigationhubviewservice']
    path:
%extending_platformui.public_dir%/js/views/services/plug
ins/ezconf-navigationplugin.js
```

View service plugins are a bit special, they need to follow a specific interface provided by `Y.eZ.Plugin.ViewServiceBase` which is defined in the `ez-viewservicebaseplugin` module, so our module needs to require it.

Then, the base plugin can be written on disk. It is very close to the base plugin written in the Alter the JavaScript Application routing step:

ezconf-navigationplugin.js

```
YUI.add('ezconf-navigationplugin', function (Y) {
    Y.namespace('eZConf.Plugin');

    // view service plugins must extend
Y.eZ.Plugin.ViewServiceBase
    // Y.eZ.Plugin.ViewServiceBase provides several
method allowing to deeply
    // hook into the view service behaviour
    Y.eZConf.Plugin.NavigationPlugin =
Y.Base.create('ezconfNavigationPlugin',
Y.eZ.Plugin.ViewServiceBase, [], {
        initializer: function () {
            var service = this.get('host'); // the
plugged object is called host

            console.log("Hey, I'm a plugin for
NavigationHubViewService");
            console.log("And I'm plugged in ", service);
        },
    }, {
        NS: 'ezconfNavigation'
    });

    Y.eZ.PluginRegistry.registerPlugin(
        Y.eZConf.Plugin.NavigationPlugin,
['navigationHubViewService']
    );
});
```

At this point, if you refresh you browser, the navigation hub should remain the same but you should see new messages in the console.

Adding a new navigation item

Now that we have plugin plugged in the Navigation Hub View service, we can change the menu structure. Among others methods, the Navigation Hub view service has an `addNavigationItem` method to add a navigation item in a given zone, so we can use it in our plugin to add a new item:

ezconf-navigationplugin.js

```
YUI.add('ezconf-navigationplugin', function (Y) {
    Y.namespace('eZConf.Plugin');

    Y.eZConf.Plugin.NavigationPlugin =
Y.Base.create('ezconfNavigationPlugin',
Y.eZ.Plugin.ViewServiceBase, [], {
        initializer: function () {
            var service = this.get('host');

            console.log("Hey, I'm a plugin for
NavigationHubViewService");
            console.log("And I'm plugged in ", service);

            console.log("Let's add the navigation item
in the Content zone");
            service.addNavigationItem({
                Constructor: Y.eZ.NavigationItemView,
                config: {
                    title: "List contents",
                    identifier: "ezconf-list-contents",
                    route: {
                        name: "eZConfList" // same route
name of the one added in the app plugin
                    }
                }
            }, 'platform'); // identifier of the zone
called "Content" in the UI
        },
    }, {
        NS: 'ezconfNavigation'
    });

    Y.eZ.PluginRegistry.registerPlugin(
        Y.eZConf.Plugin.NavigationPlugin,
['navigationHubViewService']
    );
});
```

At this point, if you refresh you browser, you should see a new entry in the *Content* zone called *List contents*. Clicking on this link should even get you to the page defined in the previous step. And you can also notice that the navigation item gets a special style (a green bottom border) when the `eZConfList` route is matched and that it loses this style if you navigate elsewhere.

Results and next step:
The resulting code can be seen in the 5_navigation tag on GitHub, this step result can also be viewed as a diff between tags `4_view` and `5_navigation`.

The next step is then to build and display the content list.