Paramétrage du Front Office

Le but de ce document est de détailler la configuration du Front Office Workey. Elle peut être séparée en deux parties, une avec l'utilisation d'un fichier de paramètres JSON à définir dans l'interface de l'application et une autre, avec des paramètres Java, à définir dans le fichier de configuration du serveur *catalina.properties*.

La première partie, plus cossue, permet de définir la structure, l'agencement et la représentation de l'application.

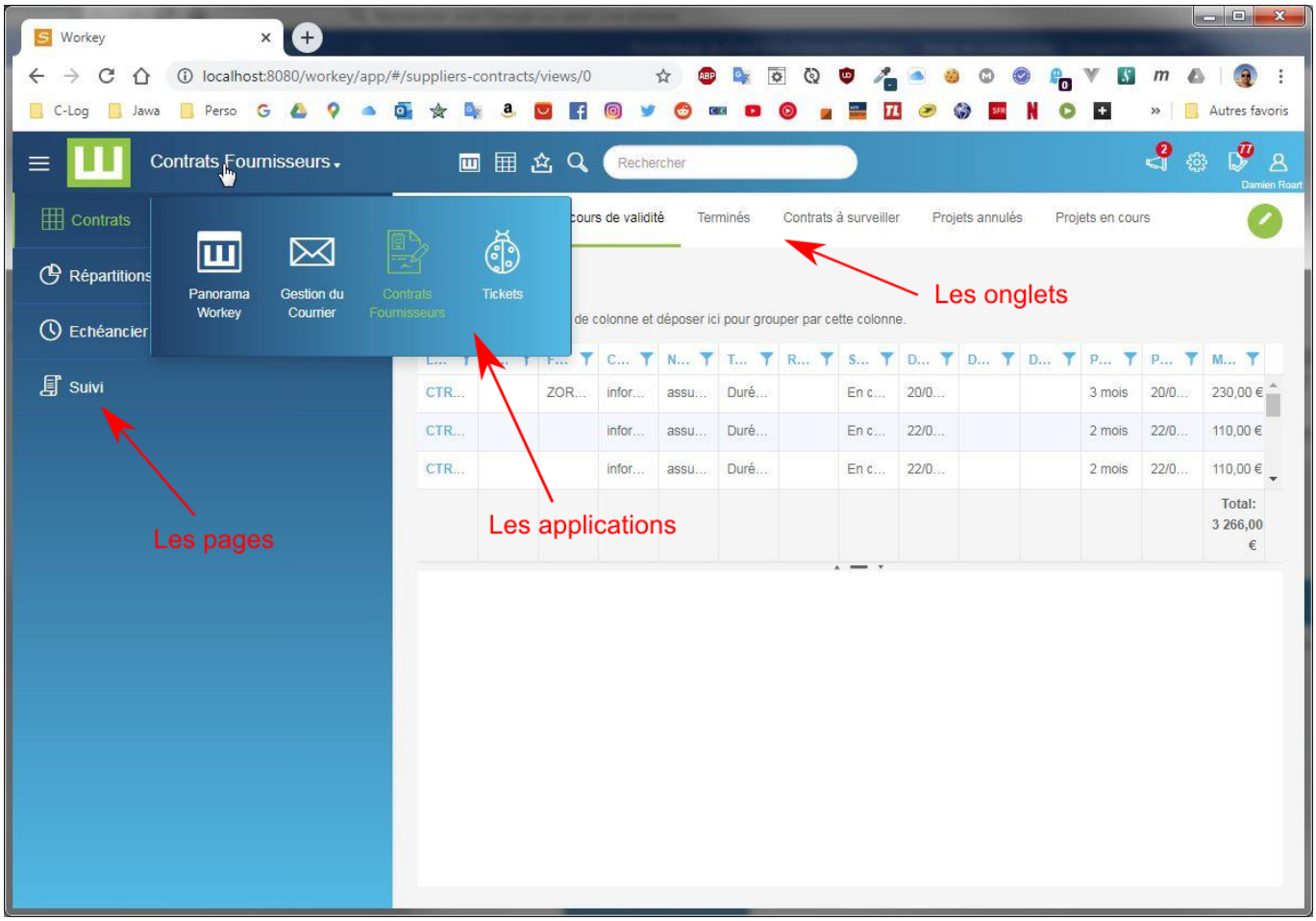
Elle se décompose en deux parties :

- Une partie générique qui est le panorama Workey transverse à tous les processus
- Une autre partie qui représente les processus métier sous formes d'applications verticales.

Les deux parties peuvent être utilisées simultanément. La partie Panorama Workey n'est rien d'autre qu'une application prédéfinie.

L'application se décompose donc en applications qui elles-mêmes se décomposent en pages qui contiennent des onglets.

Décomposition de l'écran :

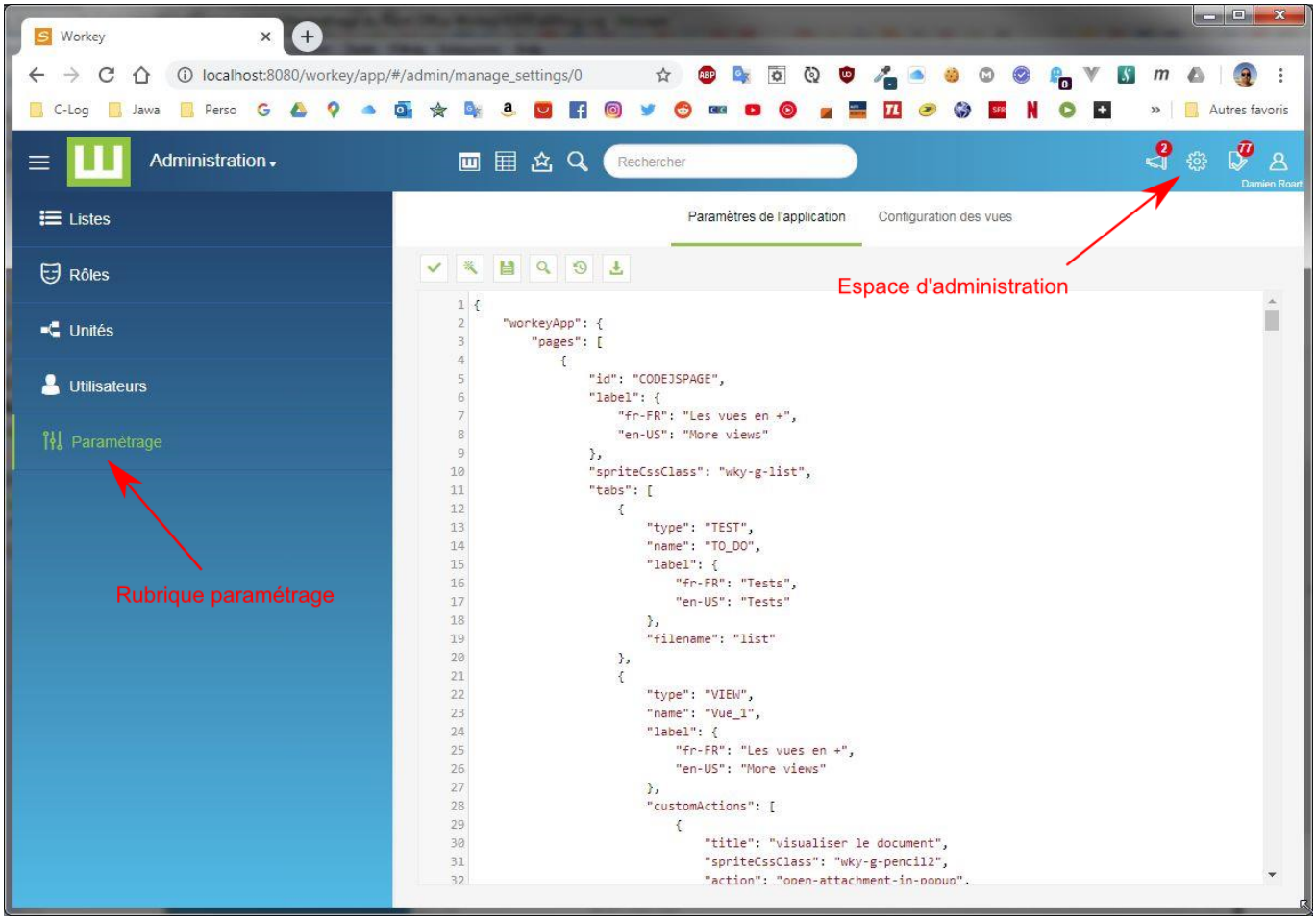


Emplacement du fichier de configuration

Le fichier se nomme settings.json et se situe sur le serveur là où est installé Workey, c'est-à-dire dans le répertoire des données de Workey. Le répertoire est le suivant *$WORKEY_DATA /settings*, *$WORKEY_DATA* étant la plupart du temps Workey_Data.

Modification du fichier

Un fichier JSON est tout simplement un fichier texte. Il est accessible en ligne au sein de l'application. Pour y accéder vous devez être connecté en tant qu'administrateur (Workflow Manager) et vous rendre dans l'espace d'administration, rubrique Paramétrage.



Pour l'éditer localement, vous pouvez aussi utiliser votre éditeur de texte préféré hormis les traitements de texte (Word, Libre Office etc). Il est doit être encodé au format UTF-8.

Il est recommandé d'utiliser l'édition en ligne qui offre un correcteur syntaxique, un correcteur grammatical, un backup à chaque modification et le retour à une configuration antérieure.

PRISE EN COMPTE DES MODIFICATIONS

Il vous suffit de remplacer l'ancien fichier settings.json par le nouveau. Il est fortement recommandé de garder l'ancien au cas où… L'édition en ligne

exécute et gère ces recommandations.

LA SYNTAXE DU JSON

JSON signifie JavaScript Object Notation et c'est un format de données. Autrement dit, c'est une façon de stocker des informations, un peu comme dans un fichier XML.

JSON est un format de données consistant en paires de nom/valeur (ou clé/valeur) ayant la forme de chaînes de caractères. Les noms et valeurs sont séparés par deux points : et chaque paire est séparée de la suivante par une virgule.

Certains caractères permettent de structurer les données dans un fichier. En voici la liste :

- {…} : les accolades définissent un objet.
- Les guillemets (double-quotes) et les double-points définissent un couple clé:valeur, ou propriété:valeur, ou attribut:valeur (on parle aussi de membre, *ex : "nom":"Jean-Pierre"*).
- […] : Les crochets définissent un tableau (ou array en anglais).
- {"id":1, "nom":"Jean-Pierre", "age":25} : Les virgules permettent de séparer les membres d'un tableau ou, comme ici, d'un objet. A noter : pas de virgule pour le dernier membre d'un objet, sinon, il ne sera pas valide et vous aurez des erreurs lors de l'analyse du fichier.

Il est possible de combiner les objets et les tableaux. Exemple d'un objet JSON contenant un tableau :

{"id":1, "nom":"Jean-Pierre", "age":25, "auteur":true, "enfants":["Julie", "Marius", Romain"]}

Exemple d'un tableau JSON contenant des objets:

[{"id":1, "nom":"Jean-Pierre"}, {"id":2, "nom":"Caroline"}, {"id":3, "nom":"Noah"}]

La valeur d'une paire peut être du texte, un nombre ou un booléen (true ou false).

C'est à peu près tout ce qu'il faut connaître pour la syntaxe du JSON.

Pour une meilleur lisibilité du fichier, il est préférable d'indenter le contenu en fonction du niveau d'introspection dans la structure de données, comme en XML.

```
{
        "id":1,
        "name":"Jean-Pierre",
        "age":33,
        "employed":true,
        "children": [
```

```
                "Julie",
                "Marius",
                "Romain"
        ]
}
```

Structure du fichier

Le fichier, définissant les paramètres de Workey, est composé un objet JSON qui contiendra toutes les données de configuration, par exemple :

```
{
        "verticalApps":[
                {
                        "id":"CONTRACT",
                        "route":"contract",
                        "label":"Contrats Fournisseurs",
                        "spriteCssClass":"wky-g-contract",
                        "availableDocuments":[
                                …
                        ],
                        "descendingDocuments":[
                                ...
                        ],
                        "pages":[
                                …
                        ],
                },
                {
                        "id":"TICKETS",
                        "route":"ticket",
                        "label":{
                                "fr-FR" :"Gestion des tickets",
                                "en-US" :"Ticket Management"
                        },
                        "spriteCssClass":"wky-g-ticket",
                        "availableDocuments":[
                                …
                        ],
                        "descendingDocuments":[
                                ...
                        ],
                        "pages":[
                                …
                        ],
                        "tables":[
                                …
                        ],
                }
        ],
```

```
        "extraPermissions":{
                "listsManagement": ["jrobert", "dmorgane"]
        },
        "workeyApp":{
                …
        }
}
```

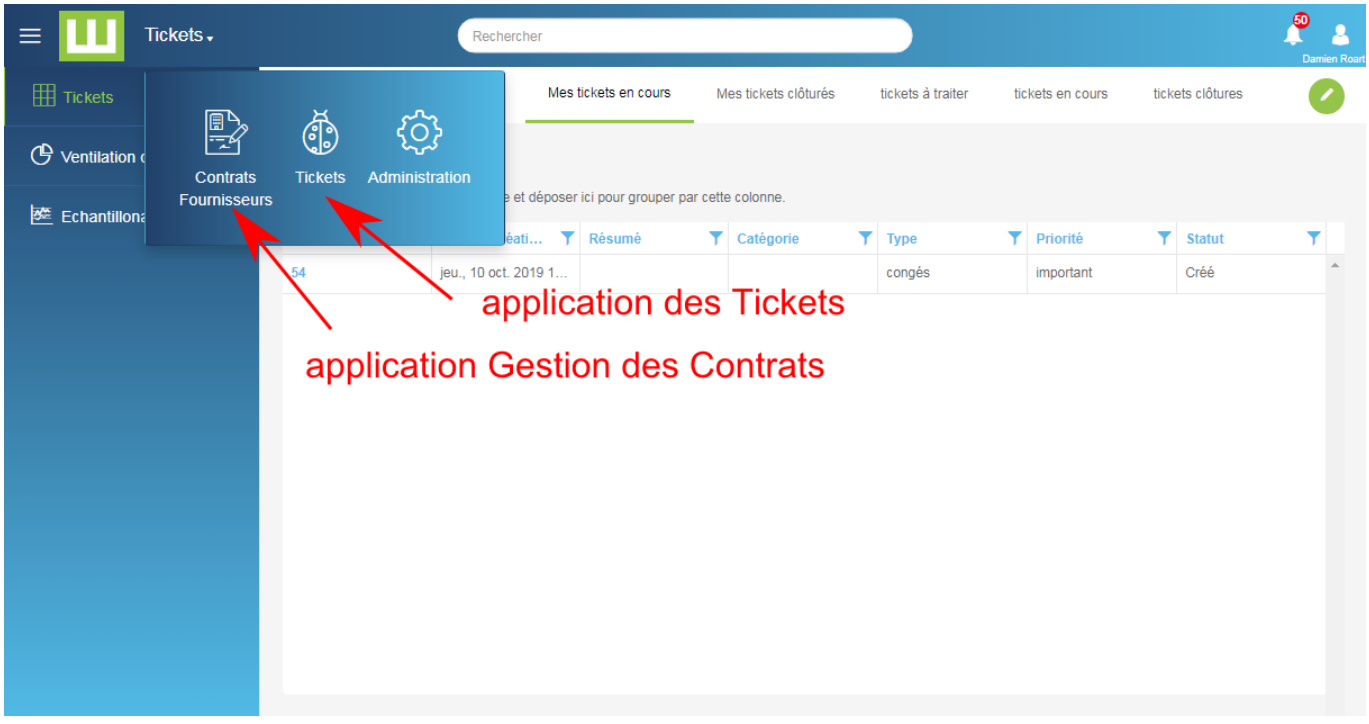
Cet objet contient trois propriétés, une pour les applications verticales, une pour l'application générique Workey (Panorama) et une autre pour les extra-permissions.

- *verticalApps* : elle permet de spécifier les applications verticales accessibles sous forme d'un tableau JSON (Cf Définition des application verticales)
- *workeyApp* : cette propriété permet de spécifier la configuration propre à l'espace par défaut Workey. Il se compose d'un objet JSON (Cf Définition du Panorama Workey)
- *extraPermissions* : cette propriété permet de spécifier des permissions exceptionnelles à des groupes d'individus (Cf Définition du permissions exceptionnelles)

Définition des applications verticales

Ce tableau contient deux objets correspondant à deux applications, une qui gère les contrats et l'autre les tickets. Pour spécifier vos applications, consulter la Définition d'une application.

Résultat à l'écran :



Définition d’une application

Une application est un objet JSON qui contient diverses propriétés :

id : sert à identifier l’application, il doit être unique au sein du fichier.

route : sert à définir le morceau d’url qui permettra d’y accéder. Par exemple si vous lui donnez comme valeur ‘contract’, l’url http://www.lempire.com/workey/app/#/contract rend l’accès possible à l’application Contrats. Elle est par conséquent unique au sein de l’application.

label : présente le nom de l’application dans Workey.

Cette propriété peut être sous forme d’une chaîne de caractères.

Exemple :

“label”:”Gestion des contrats”

Si votre application est multi-langue, vous pouvez spécifier un objet JSON à la place de la chaîne de caractères cela permettra de définir les différentes langues. Par exemple pour le français et l’anglais:

“label” :{

“fr-FR”:”Gestion des contrats”,

“en-US”:”Contracts management”

}

Aujourd’hui Workey ne supporte que le français et l’anglais.

spriteCssClass : détermine l’icône à utiliser pour représenter l’application dans le menu de navigation en haut à gauche du front. Il existe une liste d’icônes prêtes à l’emploi (cf Icônes de l’application).

todoToUse : permet de spécifier une vue *A faire* à afficher à la place de celle par défaut. Sa valeur est le nom interne de la vue désirée, elle est affichée lors du click sur l’icône *Documents à traiter*

.

enable : sert à activer ou désactiver l’application. Cette propriété peut prendre *true* ou *false* comme valeur. La valeur *true* pour activer et *false* pour la cacher.

availableDocuments : spécifie les documents qui peuvent être créés dans l’application. Sa valeur est un tableau contenant des objets qui représentent les documents Workey candidats à la création. Si l’utilisateur connecté peut créer au moins un document, l’icône de création

apparait pour qu'il puisse créer un nouveau document. Chaque objet a une propriété 'names' qui contient la concaténation des ids du document Workey (*Process_Type_Name/Document_Type_Name/Role_Name*). La valeur de *names* est composée des id suivants :

- *Process_Type_Name* est le nom interne du processus dans le modélisateur Workey
- *Document_Type_Name* est le nom interne du document dans le modélisateur workey
- *Role_Name* est le nom interne du rôle qui crée le document Workey dans le processus au sein du modélisateur.

Par exemple :

"availableDocuments" :[

{"names":"Gestion_des_contrats/Contrat/Gestionnaire_de_contrat"},

{"names":"Gestion_des_contrats/Avenant/Controleur"}

]

Cette propriété permet d'exposer les documents à la création en adéquation avec l'application en cours.

descendingDocuments : cet attribut spécifie les documents fils qui peuvent être créés dans l'application. Ils sont issus des dérivations et leur représentation est identique à celle de l'attribut *availableDocuments (voir ci dessus). On notera que Role_Name n'est pas spécifié.*

Exemple :

"descendingDocuments":[

{"names":"Gestion_des_contrats/Facture"},

{"names":"Gestion_des_contrats/Engagement"}

]

pages : cette propriété va vous permettre d'agencer/de décomposer l'application en pages. Chaque page spécifiée se retrouve dans la colonne de gauche de Workey. Cette propriété est donc un tableau de page. L'ordre d'apparition dans le fichier est répercuté à l'écran.

Exemple :

"pages" :[

{…},

{…},

{…}

]

tables : Cette propriété permet de définir les domaines de valeur qui vont être gérés dans Workey. Les domaines sont regroupés par application, c'est pour cela qu'ils sont définis en sein de chaque application. Chaque domaine est spécifié dans un objet JSON contenu lui-même dans un tableau JSON.

Exemple :

"tables" :[

{…},

{…}

]

Le détail du contenu se trouve à la Gestion des domaines de valeurs.

tablesManagers : cette propriété permet de lister les utilisateurs qui peuvent modifier les listes de domaines de valeurs dans l'application sans qu'ils soient administrateurs. C'est un tableau contenant les DNs des utilisateurs.

Exemple :

"tablesManagers" : ["jdupont", "eleblanc", "pmartin"],

Les DNs des utilisateurs sont présents dans la liste des utilisateurs, qui elle-même est présente dans la gestion des utilisateurs issue de l'administration.

Attention : Pour un utilisateur connecté, une application est visible si et seulement ce dernier peut créer au moins un document ou accéder au moins à une vue.

Définition d'une page

Sur le même principe que l'application la page est un objet constitué de propriétés.

Id : sert à identifier la page, il doit être unique au sein d'une application.

label : présente le nom de la page dans l'application, on le retrouve comme entrée dans la colonne de gauche de Workey. Pour le spécifier, il faut faire la même chose que pour le label de l'application (cf label d'une application).

spriteCssClass : l'attribut spriteCssClass détermine l'icône de l'entrée dans la colonne de gauche. Pour le spécifier (cf Icône de l'application).

tabs : cette propriété regroupe les onglets qui seront affichés en haut dans la page. C’est un tableau de tab. L’ordre d’apparition dans le fichier est répercuté à l’écran.

Exemple d’une page

“pages”: [

{

“id”:”PAGE01″,

“label”:{

“fr-FR”:”Contrats”,

“en-US”:”Contracts”

},

“spriteCssClass”:”wky-g-view2″,

“tabs”:[

…

]

},

{

“id”:”PAGE02″,

…

}

]

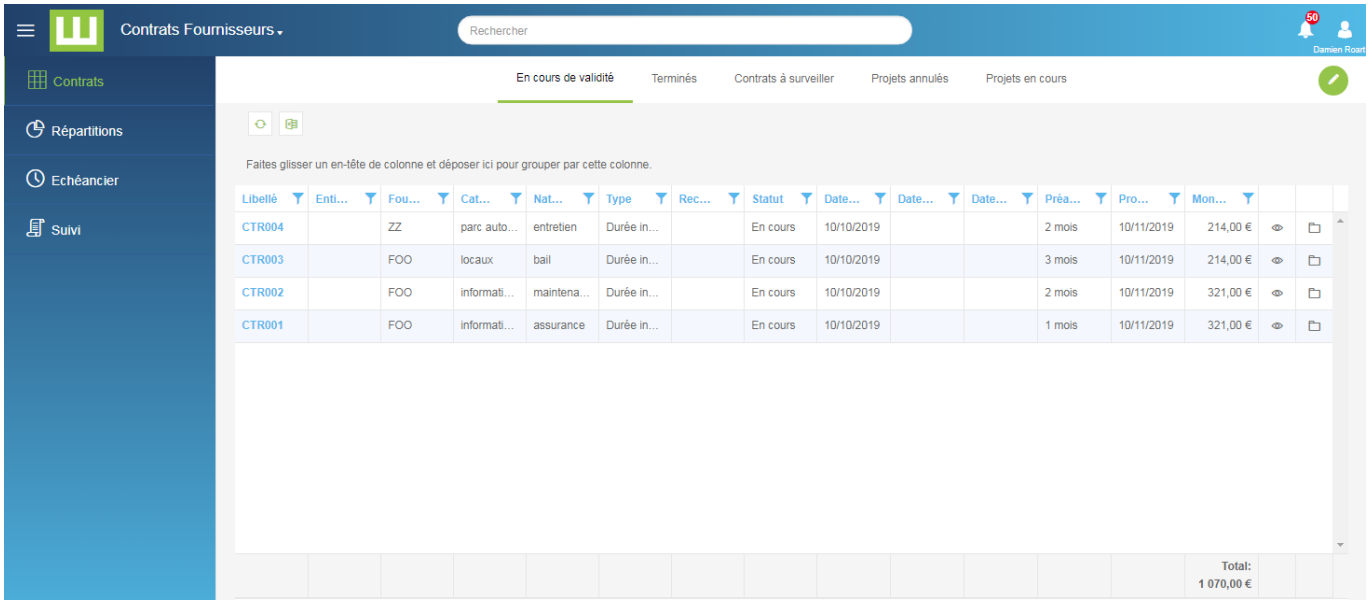
Définition d’un onglet

Pour un onglet, vous allez afficher une vue ou un graphe (view, pie chart ou line chart).

Définition d’un onglet de type vue

Comme son nom l’indique un onglet de type vue affiche une vue Workey créée avec le modéliser Workey Designer.

Exemple du vue :



Pour un onglet de type vue comme pour l'application ou la page, vous allez spécifier un objet avec des propriétés.

type : il doit être valorisé à 'VIEW', attention vous devez respecter la casse.

label : présente le nom de l'onglet dans la page, on le retrouve comme entrée dans la barre d'onglets en haut de la page. Pour le spécifier, il faut faire la même chose que pour le label de l'application (cf label d'une application).

name : est le nom interne de la vue Workey à afficher.

agreggates : permet de grouper les données d'une colonne pour effectuer une action dessus. Vous pouvez déterminer seulement un aggregate par colonne. Sa valeur est un tableau d'aggregate. Un aggregate se décompose en 4 propriétés :

- *field* : spécifie la colonne sur laquelle les données sont groupées
- *aggregate*: cet attribut réalise l'action souhaité et peut prendre comme valeur sum, count ou average (somme, comptage ou moyenne).
- *format* : cette propriété formate le résultat de l'action cf Définition d'un format
- *label* : labellise le résultat de l'aggregate.

Vous obtenez la définition d'un aggregate suivant :

"aggregates":[

{

"field":"Montant_echeance",

"aggregate":"sum",

"format":"{0:c}",

“label”:{

“fr-FR” :”Total: “,

“en-US”:”Total: ”

}

}

]

L’aggregate ci-dessus affichera le total des échéances en bas de la colonne Montant_echeance de la vue.

customActions : cet attribut permet de rajouter des actions personnalisées sur les lignes des vues. Sa valeur est un objet JSON contenant différents attributs :

- *title* : texte affiché en tooltip sur l’icône en passant la souris dessus.
- *spriteCssClass* : permet de spécifier une icône qui représente l’action à effectuer.
- *action* : cette propriété a comme valeur le nom d’une fonction prédéfinie cf Liste des actions prédéfinies.
- *params* : est un objet JSON qui spécifie les paramètres passés à la fonction (*action*). Ils sont issus de la ligne affichée dans la vue (le paramètre passé correspond au nom interne d’une colonne). Chaque clé/valeur correspond au nom du paramètre attendu (clé) et la valeur de la colonne (valeur).

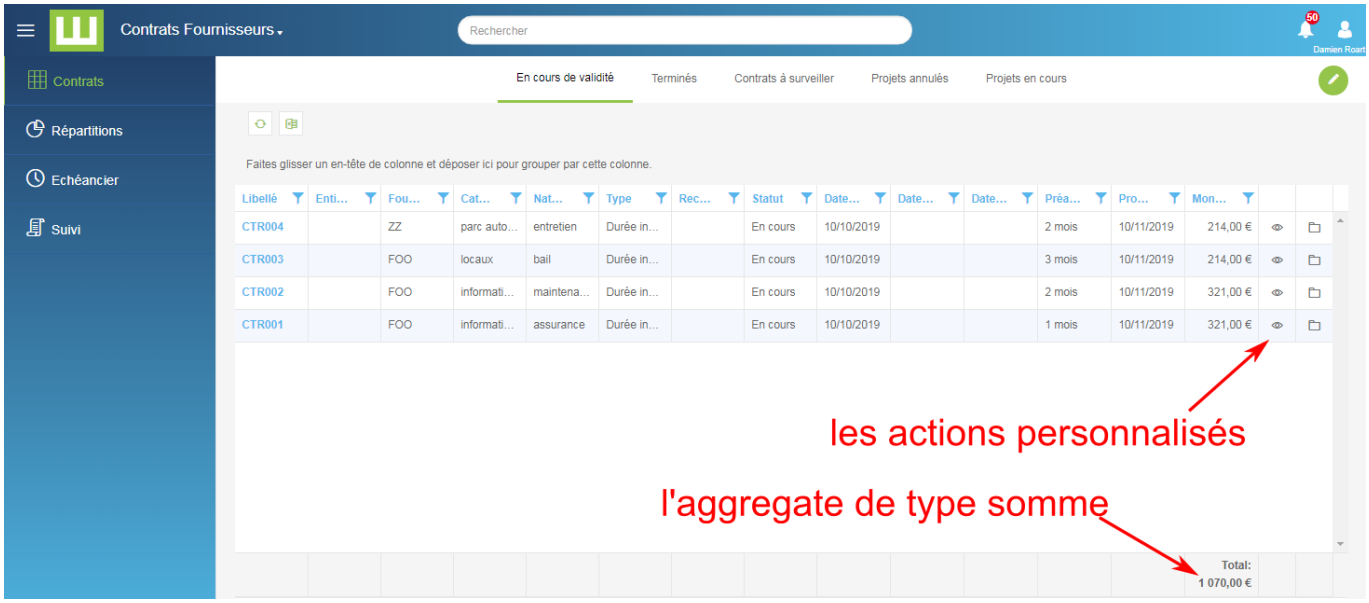
Finalement le JSON correspondant aux customActions est le suivant

```
[
                {
                        "title":"Visualiser le document dans un popup",
                        "spriteCssClass":"wky-g-eye",
                        "action": "open-attachment-in-popup",
                        "params": {
                                "uuid" : "piece_jointe",
                                "title" : {
                                                "fr-FR" :"Titre de la
fenêtre",
                                        "en-US":"Window title: "
                                }
                        }
                },
                {
                        "title":"Visualiser le document dans un nouvel
onglet",
                        "spriteCssClass":"wky-g-folder",
                        "action": "open-attachment-in-window",
```

```
                "params": {
                        "uuid" : "annexe"
                }

        }
]
```

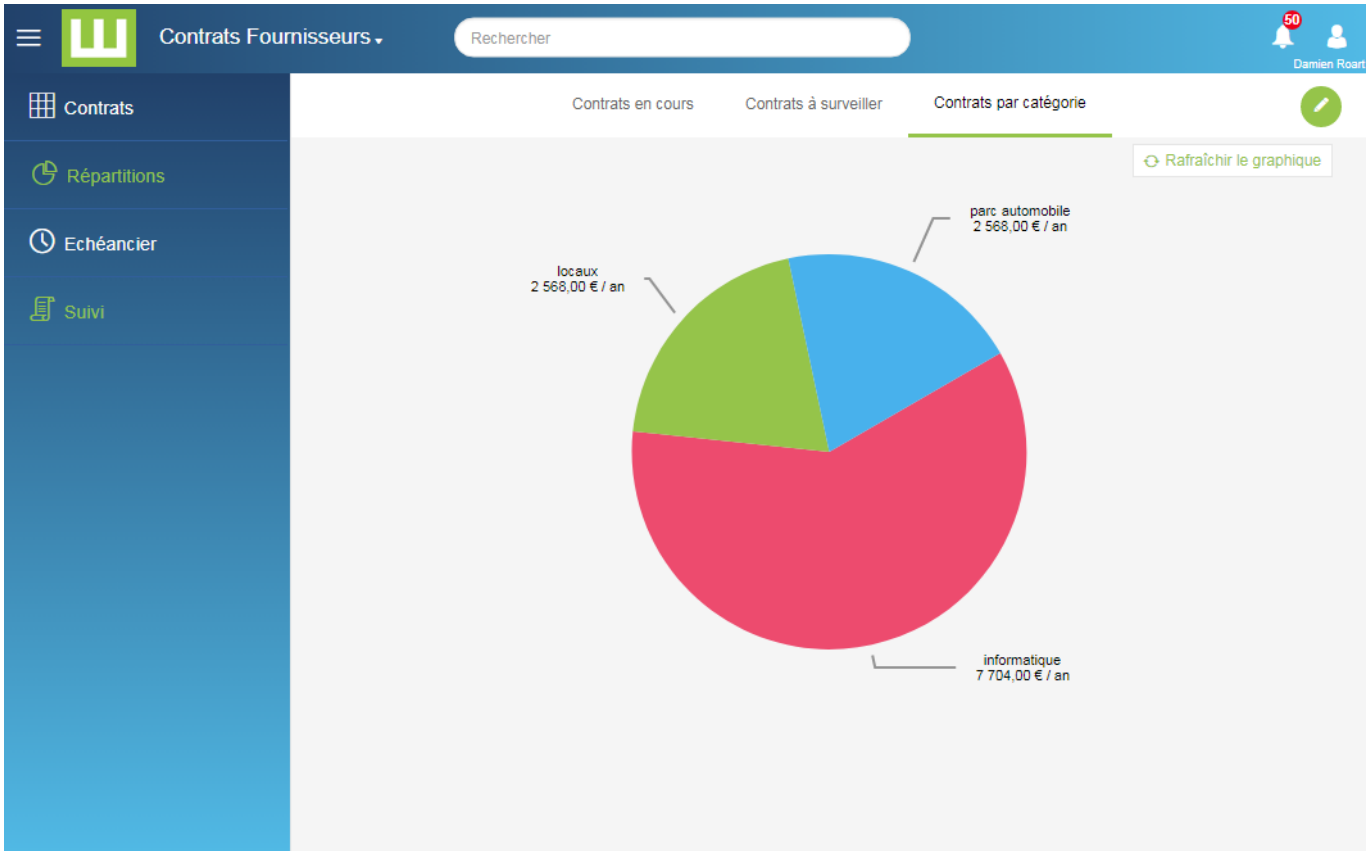
Exemple d'implémentation de l'aggregate et de customs actions :



Définition d'un onglet de type diagramme circulaire

Pour un onglet de type graphe, c'est presque la même chose que pour les vues ci-dessus. Vous allez afficher une vue Workey mais changer sa représentation graphique.

Représentation en diagramme circulaire :



Pour déclarer ce graphe vous allez devoir spécifier ses attributs.

type : il doit être valorisé à 'PIE_CHART', attention vous devez respecter la casse.

label : cf Définition d'un onglet View

name : cf Définition d'un onglet View

groupBy : cette propriété permet de former des groupes de données selon sa valeur. C'est un tableau de colonnes issues de la vue spécifiée par l'attribut name ci-dessus. Vous devez donc spécifier un tableau de noms internes déterminés dans le modélisateur Workey. Cette attribut fonction comme le GROUP BY des requêtes SQL.

Exempe :

"groupBy":["Categorie"]

L'attribut *groupBy* doit être combiné avec la propriété *aggregate* qui va réaliser l'action souhaitée.

- *aggregate* : cette propriété réalise l'action souhaitée sur les groupes de données (*groupBy*). Elle est représentée par un object JSON qui est composé de différents membres :
- *action* : cet attribut réalise l'action souhaité et peut prendre comme valeur sum, count ou average (somme, comptage ou moyenne).
- *format* : cette propriété formate le résultat de l'action cf Définition d'un format
- *field* : ce membre spécifie le nom de la colonne sur laquelle l'action se

déroule

Exemple :

“aggregate”:{

“action” :”sum”,

“format” :”{0:c}”,

“field” :”Total_annuel”

}

labelTemplate : cette propriété format le libellé affiché en regard du quartier.

Exemple :

“labelTemplate” : “#= category #\n#=dataItem.formattedValue# / an”

Finalement, pour la définition d’un Pie Chat , vous obtenez le JSON suivant :

{

“type”:”PIE_CHART”,

“name”:”Repartition_Montants_annuels_par_categorie”,

“label”:”Contrats par catégorie”,

“groupBy”:[“Categorie”],

“aggregate”: {

“action”:”sum”,

“format”:”{0:c}”,

“field”:”Total_annuel”

},

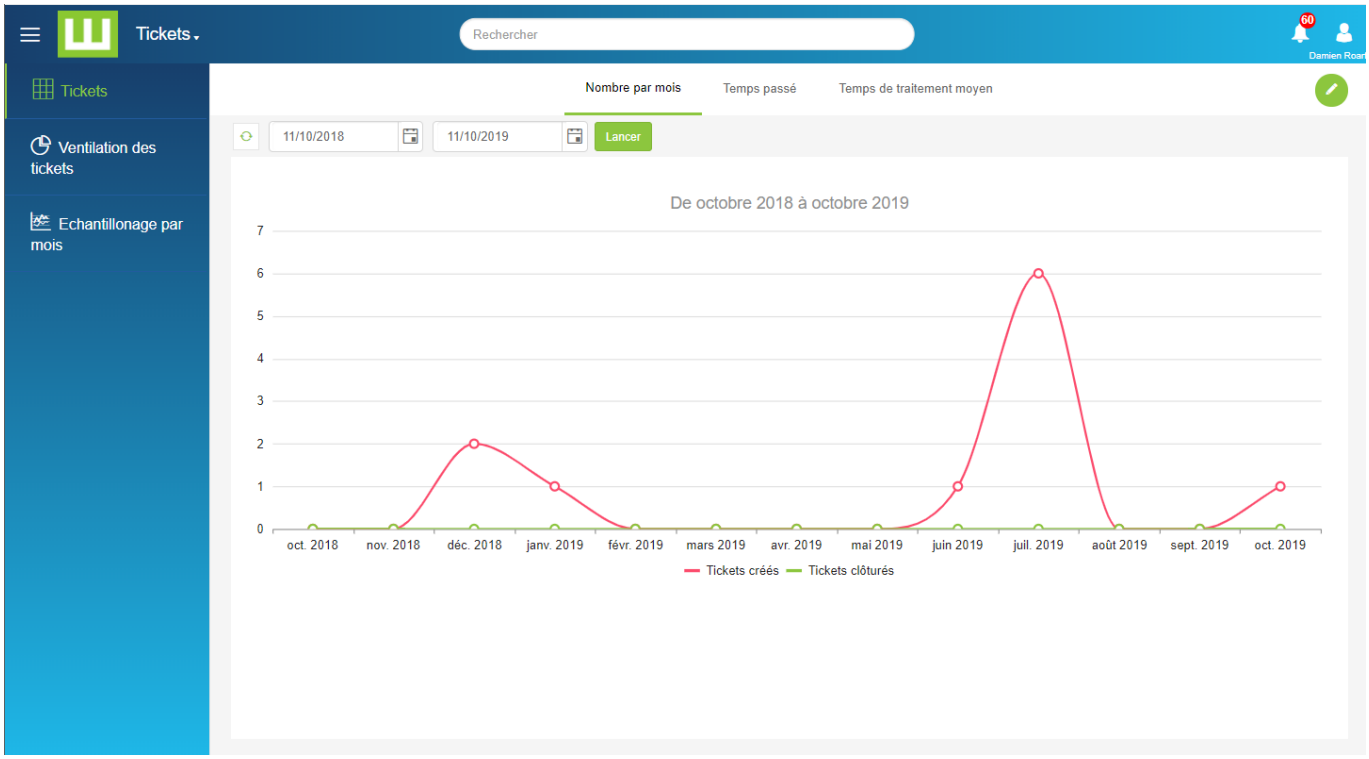
“labelTemplate”:”#= category #\n#=dataItem.formattedValue# / an”

}

Définition d’un onglet de type graphique en ligne

Le graphique en ligne permet de représenter une vue Workey en l’agençant différemment.

Exemple de graphe :



Comme pour le graphe PIE CHART, vous allez devoir déclarer différents attributs :

id : cf Définition d'un onglet View

type : il doit être valorisé à 'LINE_CHART', attention vous devez respecter la casse.

label : cf Définition d'un onglet View

name : cf Définition d'un onglet View

lines : cette propriété va vous permettre de déclarer les lignes souhaitées dans le graphe. Chaque ligne est composée de points qui vont concerner 0 ou n lignes de la vue, c'est sur ce groupement que vous allez effectuer une action. Cette propriété est représentée sous forme d'un tableau de lignes, qui elles-mêmes sont déclarées sous forme d'objet JSON composés de propriétés :

- *field* : spécifie le champ d'une colonne utilisé pour le regroupement
- *label* : détermine un libellé affiché sur les points de la ligne lorsque la souris passe dessus.

Exemple :

"lines":[

{

"field":"Date_de_creation",

"label":"Tickets créés"

},

{

“field”:”Date_de_cloture”,

“label”:”Tickets clôturés”

}

]

aggregate : Cette propriété va effectuer une action sur les groupes déterminés par l’attribut *lignes* ci-dessus. L’action est un objet JSON composé de propriétés :

- action : cet attribut réalise l’action souhaité et peut prendre comme valeur sum, count ou average (somme, comptage ou moyenne).
- *field* : contient le nom du champ des lignes sur lesquelles l’action est réalisée.

Définition du Panorama Workey

Cette espace est prédéfini par défaut dans le front, il ne nécessite pas de configuration par défaut, par contre Il est customisable et peut être enrichi avec de nouvelles pages. Il se compose d’un object JSON, en regard de la propriété *workeyApp,* contenant différentes propriétés :

label : cette propriété définit le libellé du Panorama Workey. Par défaut sa valeur est donc *Panorama Workey*, pour la modifier, consultez la propriété label d’une application (Cf Définition d’une application).

spriteCssClass : Cette propriété modifie l’icône utilisée pour le panorama (Définition d’une application).

pages : Cette propriété permet d’ajouter des pages au panorama Workey qui s’ajoutent en dessous de la page Panorama Workey. Elle se compose d’un tableau JSON qui contient les différentes pages (Cf Définition d’une page).

tables : Cette propriété permet de définir les domaines de valeur qui vont être gérés dans le panorama Workey. Chaque domaine est spécifié dans un objet JSON contenu lui-même dans un tableau JSON.

Le détail du contenu se trouve à la Gestion des domaines de valeurs.

Définition des permissions exceptionnelles

Il se compose d’un objet JSON, en regard de la propriété *extraPermissions,* contenant différentes propriétés définissants des permissions. Chaque permission définit un tableau JSON contenant les DNs des utilisateurs concernés.

Liste des permissions

listsManagment : Cette propriété contient les utilisateurs non administrateur qui peuvent gérer les domaines de valeurs dans l'application Workey.

GESTION DES DOMAINES DE VALEURS

Les domaines de valeurs peuvent être gérés au sein de l'administration de Workey, dans la rubrique Listes. Chaque Liste / Domaine de valeur est le reflet d'une table en base de données sur laquelle vous allez pouvoir ajouter ou supprimer des valeurs.

Il faut bien-sûr créer au préalable la table dans la data source adéquate.

Les domaines de valeurs sont représentés sous forme de tableaux JSON contenus dans la propriété *tables* d'une application (cf Définition d'une application).

L'objet JSON de déclaration est formé de plusieurs propriétés :

name : Nom de la table dans la data source

label : Libellé localisé de la table plus descriptif que le nom de la table

fields : Liste des champs liés à la gestion, sa valeur est un tableau JSON contenant n champs, chaque champ est objet JSON composé d'attributs :

- *name* : Nom de la colonne dans la table
- *label* : Libellé localisé de la colonne plus descriptif que le nom de la colonne
- *type* : Type de donnée de la colonne. Pour le moment cette propriété peut prendre comme valeur : string, integer ou float.

Exemple :

"tables":[

{

"name":"contrat",

"label":{

"fr-FR" : "Liste des catégories",

"en-US" :"List of categories"

},

"fields":[

{

"name":"Categorie",

“type”:”string”

},

{

“name”:”Nature”,

“type”:”string”

}

]

},

{

“name”:”liste_societe”,

“label”:”Sociétés”,

“fields”:[

{

“name”:”societe”,

“type”:”string”

}

]

}

]

LISTE DES ACTIONS PREDEFINIES

Les “custom actions” sont des actions effectuées sur les documents présents dans une vue. Elles sont déclarées dans la définition d’une vue. Vous trouvez ci-dessous la liste des actions prédéfinies.

open-attachment-in-window

Cette action affiche une pièce jointe Workey dans une nouvelle fenêtre ou onglet du navigateur. Il faut lui passer en paramètre la valeur suivante (via la propriété *params*) :

- uuid : c’est l’identifiant d’une pièce jointe Workey

open-attachment-in-popup

Cette action affiche une pièce jointe Workey dans une boîte de dialogue modale (l'utilisateur doit la fermer avant de pouvoir continuer dans Workey). Il faut lui passer en paramètre les paramètres suivants (via la propriété *params*) :

- uuid : c'est l'identifiant d'une pièce jointe Workey
- title : titre localisé du popup

mg-open-viewer

Cette action ouvre un popup de visualisation d'un document Multigest. Vous devez lui passer le paramètre suivant nécessaires au bon fonctionnement (via la propriété *params*) :

- uuid : l'id du dossier MG

mg-open-dossier

Cette action ouvre un popup de visualisation d'un dossier Multigest. Vous devez lui passer le paramètre suivant nécessaires au bon fonctionnement (via la propriété *params*) :

- uuid : id du dossier MG

ICONES DE L'APPLICATION

Workey embarque une font d'icônes, elle est constituée de différents glyphes prédéfinis. Pour visualiser son contenu il faut accéder au système de fichiers du serveur Workey. Sa localisation est la suivante :

Workey_Engine\webapps\workey\css\fonts\workey-glyphicons.html

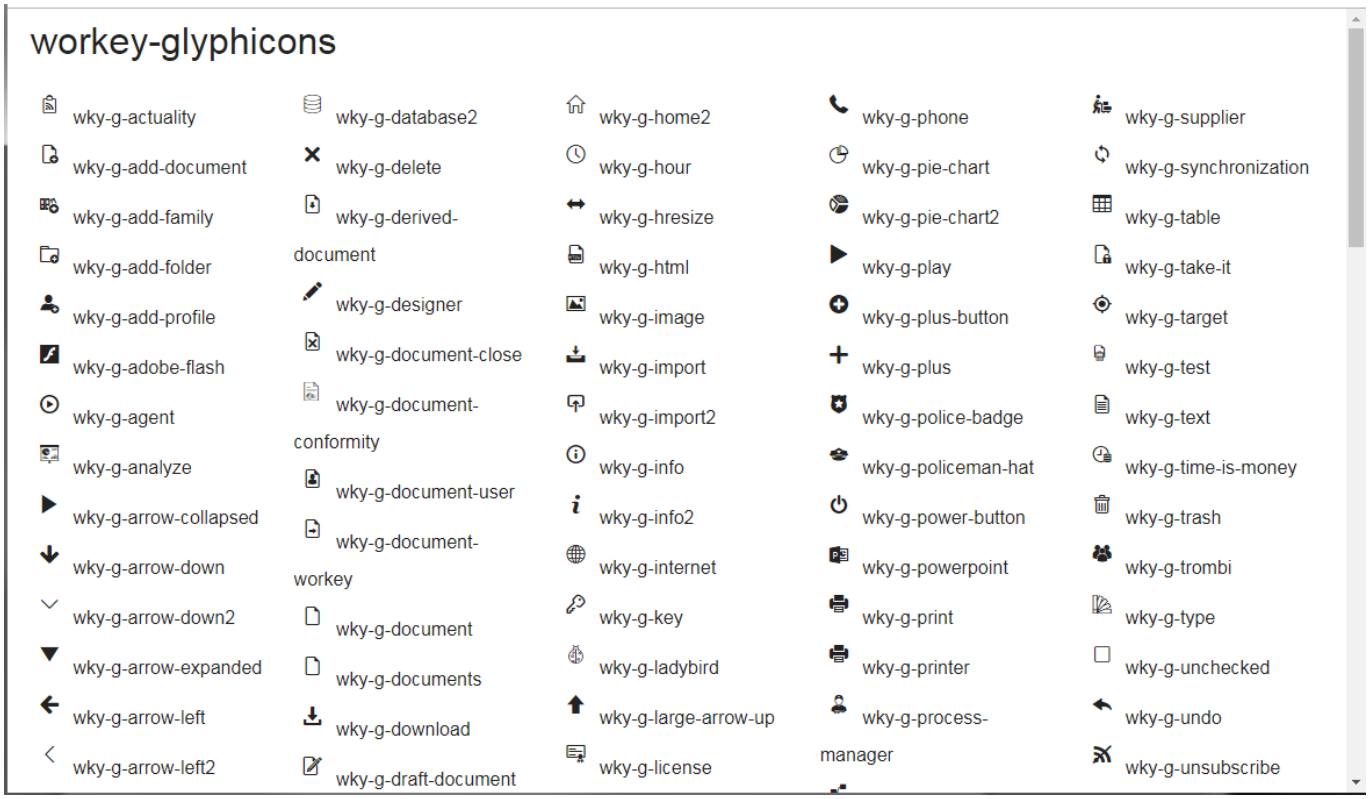
Pour utiliser une icône, il faut utiliser son nom qui est sous la forme wky-g-xxx.

Ainsi pour :

wky-g-pencil vous obtenez l'icône


wky-g-mail vous obtenez l'icône


Aperçu de la font d'icônes dans un navigateur:



DEFINITION D'UN FORMAT

Il est possible de formater une valeur (nombre) issue d'une action sur un groupe de données lors de son affichage. Il est souvent contenu dans la propriété *format*.

Exemple

"format":"{0:c}"

Voici quelques exemples.

Formatage	Valeur	Résultat
{0:n}	1234	1 234,00
{0:n1}	123	123,0
{0:n1}	123,55	123,6
{0:n0}	123.55	124
{0:c}	1234	1 234,00 €
{0:c0}	-123,55	-124 €
{0:c0}	123,55	124 €
{0:p}	0,1	10,00 %
{0:p0}	0,1	10 %
{0:p3}	200	20 000,000 %
Le prix est de {0:c}	23,6	Le prix est de 23,60 €

Paramètres Java Back Office

Les paramètres suivants se définissent dans le fichier de configuration *catalina.properties* situé dans le répertoire *conf* du serveur Tomcat. Pour qu'ils soient pris en compte, il faut que le serveur soit redémarré.

com.clog.workey.app.running.mode

Description : ce paramètre permet de spécifier le mode de fonctionnement du Front Office (mode SAAS ou On Permise)

Valeur(s) possible(s) : SAAS ou ON_PREMISE

Valeur par défaut : SAAS

com.clog.workey.app.enable

Description : ce paramètre active le Front Office pour qu'il soit accessible

valeur(s) possible(s) : true ou false

valeur par défaut : false

com.clog.workey.app.workey.app.enable

Description : ce paramètre active l'affichage de la partie Workey par défaut (Panorama Workey)

valeur(s) possible(s) : true ou false

valeur par défaut : false

com.clog.workey.app.vertical.apps.enable

Description : ce paramètre active l'affichage de la partie Application verticales

Valeur(s) possible(s) : true ou false

Valeur par défaut : false

com.clog.workey.app.custom.app.name

description : ce paramètre permet de spécifier un nom pour l'application, il permet d'effacer l'identité du produit Workey au profit d'un nom choisi par le client.

valeur(s) possible(s) : Chaine de charactères

com.clog.workey.app.custom.no.logo

Description : ce paramètre permet de ne pas afficher le logo Workey sur la page de login, il permet d'effacer l'identité du produit Workey.

Valeur(s) possible(s) : true ou false

Valeur par défaut : false

com.clog.workey.app.manage.users.enable

Description : ce paramètre permet d'activer la gestion des utilisateurs dans la partie administration.

Valeur(s) possible(s) : true ou false

Valeur par défaut : true

com.clog.workey.app.manage.users.crud.enable

Description : ce paramètre permet d'activer la mise à jour des comptes LDAP via la gestion des utilisateurs dans la partie administration.

Valeur(s) possible(s) : true ou false

Valeur par défaut : false

com.clog.workey.app.no.js.error.notification

description : ce paramètre permet de désactiver l'affichage des erreurs Javascript issues de l'utilisation de la librairie WKYJS.

valeur(s) possible(s) : true ou false

valeur par défaut : false

Edition des documents PDF

Dans les composants pièces jointes Workey, il est possible d'éditer les documents PDF l'aide de PDFTron. Pour ce faire, il faut activer et configurer ce service à l'aide des paramètres ci-dessous.

com.clog.workey.pdftron.enable

description : ce paramètre permet d'activer l'édition des documents contenus dans les pièces jointes Workey à l'aide de PDFTron.

valeur(s) possible(s) : true ou false

valeur par défaut : false

com.clog.workey.pdftron.candidate.docs

description : ce paramètre permet de spécifier les types de document candidats à l'édition dans PDFTron.

valeur(s) possible(s) : liste des extensions de fichier séparées par le caractère | de format *.ext1*|*.ext2*|*.ext3* par exemple .pdf|.pdfa

valeur par défaut : .pdf

com.clog.workey.pdftron.available.creation

description : ce paramètre permet la création de document PDF depuis le composant pièce jointe.

valeur(s) possible(s) : true ou false

valeur par défaut : false

Exemple de configuration minimale pour mettre en place PDFTron

com.clog.workey.pdftron.enable=true

Edition des documents Office

Dans les composants pièces jointes Workey, il est possible de modifier les documents bureautiques de type Office l'aide d'OnlyOffice. Pour ce faire, il faut :

- Un serveur OnlyOffice accessible par le serveur Workey (Tomcat). Rapprochez-vous de votre administrateur le cas échéant.
- Activer et configurer ce service à l'aide des paramètres ci-dessous.

com.clog.workey.onlyoffice.enable

description : ce paramètre permet d'activer l'édition des documents contenus dans les pièces jointes Workey à l'aide d'OnlyOffice.

valeur(s) possible(s) : true ou false

valeur par défaut : false

com.clog.workey.onlyoffice.filesize.max

description : ce paramètre permet de spécifier la taille maximale des documents Office pouvant être édités.

valeur(s) possible(s) : Nombre représentant le nombre d'octets maximal

valeur par défaut : 5242880 ce qui correspond à 5Mo

com.clog.workey.onlyoffice.docservice.viewed.docs

description : ce paramètre permet de spécifier les types de documents qui seront ouverts en lecture seule.

valeur(s) possible(s) : liste des extensions de fichier séparées par le caractère | de format *.ext1*|*.ext2*|*.ext3* par exemple .pdf|.pdfa

valeur par défaut : .pdf|.djvu|.xps

com.clog.workey.onlyoffice.docservice.edited.docs

description : ce paramètre permet de spécifier les types de documents qui seront ouverts en écriture.

valeur(s) possible(s) : liste des extensions de fichier séparées par le caractère | de format *.ext1*|*.ext2*|*.ext3* par exemple .pdf|.pdfa

valeur par défaut : .docx|.xlsx|.csv|.pptx|.txt

com.clog.workey.onlyoffice.docservice.convert.docs

description : ce paramètre permet de spécifier les types de documents qui seront convertis avant édition.

valeur(s) possible(s) : liste des extensions de fichier séparées par le caractère | de format *.ext1*|*.ext2*|*.ext3* par exemple .pdf|.pdfa

valeur par défaut :

.docm|.dotx|.dotm|.dot|.doc|.odt|.fodt|.ott|.xlsm|.xltx|.xltm|.xlt|.xls|.#ods|.fods|.ots|.pptm|.ppt|.ppsx|.ppsm|.pps|.potx|.potm|.pot|.odp|.fodp|.otp|.rtf|.mht|.html|.htm|.epub

com.clog.workey.onlyoffice.docservice.url.base

description : ce paramètre permet de spécifier le serveur qui héberge OnlyOffice, toutes les urls du service *docService* sont préfixées par cette url.

valeur(s) possible(s) : une url de *type http://server:port*

valeur par défaut : à définir obligatoirement

com.clog.workey.onlyoffice.docservice.url.converter

description : ce paramètre permet de spécifier l’url du convertisseur de document.

valeur(s) possible(s) :

un chemin d’url complétant *com.clog.workey.onlyoffice.docservice.url.base*

valeur par défaut : /ConvertService.ashx

com.clog.workey.onlyoffice.docservice.url.tempstorage

description : ce paramètre permet de spécifier l’url du stockage temporaire.

valeur(s) possible(s) :

un chemin d’url complétant *com.clog.workey.onlyoffice.docservice.url.base*

valeur par défaut : /ResourceService.ashx

com.clog.workey.onlyoffice.docservice.url.api

description : ce paramètre permet de spécifier l'url de l'API JS d'OnlyOffice.

valeur(s) possible(s) :

un chemin d'url complétant *com.clog.workey.onlyoffice.docservice.url.base*

valeur par défaut : /web-apps/apps/api/documents/api.js

com.clog.workey.onlyoffice.docservice.url.preloader

description : ce paramètre permet de spécifier l'url du preloader.

valeur(s) possible(s) :

un chemin d'url complétant *com.clog.workey.onlyoffice.docservice.url.base*

valeur par défaut : /web-apps/apps/api/documents/cache-scripts.html

com.clog.workey.onlyoffice.docservice.timeout

description : ce paramètre permet de spécifier le timeout des appels aux services

valeur(s) possible(s) : Un nombre de millisesondes

valeur par défaut : 120000

com.clog.workey.onlyoffice.docservice.secret

description : ce paramètre permet de spécifier les identifiants de connexion vers l'appelant aux services. Ils sont encodés dans les entêtes HTTP.

valeur par défaut :

com.clog.workey.onlyoffice.docservice.header

description : ce paramètre permet de spécifier des entêtes HTTP.

valeur par défaut : Authorization

com.clog.workey.onlyoffice.available.creations

description : ce paramètre permet la création de document Office depuis le composant pièce jointe.

valeur(s) possible(s) : énumération avec comme caractère séparateur |, par exemple word|excel|powerpoint

valeur par défaut : word

Exemple de configuration minimale pour mettre en place OnlyOffice

com.clog.workey.onlyoffice.enable=true

com.clog.workey.onlyoffice.docservice.url.base=http://192.168.1.29:8081

Customisation

Insertion de feuilles de style à l'application

Il est possible de rajouter vos propres CSS à l'application, pour ce faire il faut rajouter vos fichiers dans le répertoire *$WORKEY_DATA/custom-css/*.

La lecture et incorporation des fichiers sera effectuée selon l'ordre alphabétique de leur nom.

Insertion de code Javascript à l'application

Il est possible de rajouter vos propres librairie/script Javascript à l'application, pour ce faire il faut rajouter vos fichiers dans le répertoire *$WORKEY_DATA/custom-js/*.

La lecture et incorporation des fichiers sera effectuée selon l'ordre alphabétique de leur nom.

Attention le nom de fichier *__all_processes.js* est réservé et ne doit pas être utilisé.