Warkham-L : cahier des charges

Version 5 du 15 janvier 2014

Changelog

— — —

Version 5

- Warkham::oracle : refonte et simplification
- ajout de Warkham::autocomplete, qui récupère une partie des fonctions des versions précédentes de Warkham::oracle

— — —

Version 4

- Warkham::list : ajout de la méthode setMaxSelectionSize(int)
- Warkham::list : renommage de setAvailableValues() en setDataset()
- refonte Warkham::taglist
- Warkham::textarea : ajout méthode setToolbar()

— — —

Version 3

- modification de l'API de Warkham::Oracle pour proposer un mode remote et un mode local de chargement des valeurs possibles

— — —

Version 2

- ajouts champs Warkham::list, Warkham::filelist et Warkham::imagelist
- modification attribut data-route en data-url
- positionnement de valeurs par défaut pour les méthodes avec un paramètre booléen

Comportements repris de Former

- Intégration à Bootstrap (wrappers, gestion des erreurs)
- Repopulation du formulaire (POST, ->forceValue(), populate(), …)
- Localisation

Méthodes communes à tous les champs

Les méthodes suivantes sont communes à tous les champs Warkham :

Méthode	Arguments	PHP	Javascript
enable(boolean e)	e : true / false	Positionne l'attribut « disabled » sur le champ, ou ajoute l'attribut data-disabled valué à true (selon le cas)	Désactive les champs avec data-disabled à true
setValue(String val)	val : la valeur, sous forme de String.		
forceValue(String val)			
addClass(String c)	c : la class à ajouter		

Tous les champs se construisent avec un nom (« name », obligatoire) et un texte de label facultatif, qui créé le <label> correspondant s'il est présent. Par exemple :

```
Warkham::taglist("couleurs", "Couleurs disponibles")
```

La plupart des champs présentent une interface gérée en javascript, ce dernier étant chargé de valuer le champ « hidden » qui sera réellement pris en compte dans le traitement du formulaire. Cette valuation se fera au moment du submit du formulaire, avec un hook jQuery.

Le code php communique au code javascript les paramètres du champ à l'aide d'attributs « data ».

Tous les champs doivent être entourés d'un wrapper avec comme class "wrapper-wkm-[typechamp]". Par exemple :

```
<div class="wrapper-wkm-text"> … </div>
```

Warkham::text

Il s'agit d'un simple input text, avec la class « wkm-text ». Le code javascript est chargé de gérer l'éventuel masque de saisie.

Méthode	Arguments	PHP	Javascript
mask(m)	m : un masque de saisie, qui peut être : - integer - number - currency - url_accepted - ou une regex	positionne l'attribut data-mask	Gère le masque de saisie

Warkham::textarea

Il s'agit d'un textarea, avec la class « wkm-textarea ». Le code javascript est chargé de gérer l'éventuelle interface de saisie.

Méthode	Arguments	PHP	Javascript
ui(String ui)	ui : une interface de saisie : - wysiwyg (basé sur http://mindmup.github.io/bootstrap-wysiwyg/ ?) - markdown (basé sur http://www.jqueryrain.com/?VYAvkzCv ?)	positionne l'attribut data-ui	Gère l'interface de saisie
setToolbar(Array t)	t : les outils présents en toolbar. Valeurs possibles : bold, italic, heading, link, image, list, undo	positionne l'attribut data-toolbar	Créé la toolbar

Warkham::checkbox

Il s'agit d'un input checkbox, avec la class « wkm-checkbox ». Le code javascript est chargé de gérer l'éventuelle interface de saisie.

Méthode	Arguments	PHP	Javascript
text(String t)	t : le texte attaché à la checkbox (cliquable)	Créé le <label> correspondant	
check([boolean c=true])	c : true / false		

Warkham::choice

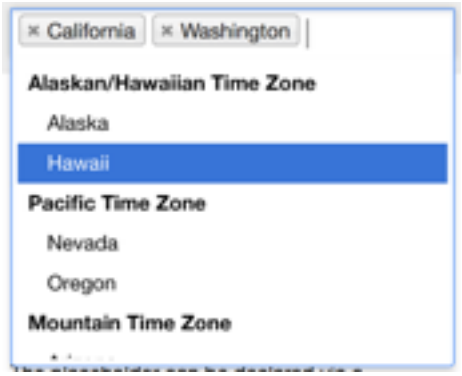
Selon le paramètre ui, il peut s'agit d'une liste de input radio, d'un select (cas par défaut) ou d'une checklist (checkboxes).

Dans les cas « radio » et « checklist », un champ « hidden » contenant la valeur ou la liste des valeurs est créé en plus, avec le nom indiqué.

Selon le cas, le ou les champs prendront les class « wkm-choice », « wkm-checklist » ou « wkm-radios ».

Méthode	Arguments	PHP	Javascript
ui(String ui)	ui : « radio », « checklist » ou « list »	Créé les input radio, checkbox ou le select en fonction de la valeur du champ (select par défaut)	
setDataset(Array values)	values : un Array [id => value]	positionne l'attribut dataset	
multiple([boolean m=true])	m : true / false. Seulement pris en compte dans le cas d'une ui=list	positionne l'attribut multiple à true	
setMaxSelectionSize(int max)	max : le nombre max d'éléments (cas « multiple » uniquement)	positionne l'attribut data-maxselectionsize	

L'attribut « multiple » génère une interface[1] du type :



La différence avec un taglist avec setAvailableValues(array, true) — voir plus loin — est qu'on gère ici une liste [id, value], et pas simplement une liste de tags, et que les tags permettent d'entrer un élément qui n'est pas dans de le dataset.

[1] http://ivaynberg.github.io/select2/

Warkham::oracle

Il s'agit d'un champ <select> dont la liste des valeurs possibles est récupérée soit en asynchrone, depuis un appel serveur, soit en local dans un dataset.

Méthode	Arguments	PHP	Javascript
setValue(id[, text])		setValue est redéfinie afin d'avoir l'id et le texte à afficher de l'élément courant sans appel serveur. Le text n'est pris en compte que dans le cas de remote = true.	
forceValue(id, text)		idem	
setRemoteRoute(String route)	route : le nom de la route vers la méthode de renvoi asynchrone des valeurs possibles	positionne l'attribut data-remoteurl	gère l'envoi de la requête et le retour des valeurs trouvées
setQueryMinLength(int min)	min : un entier définissant le nb min de caractères qu'il faut entrer pour lancer la recherche	positionne l'attrbiut data-queryminlength	
setTemplate()	gestion d'un template de présentation des valeurs possibles.	A DETAILLER SELON SOLUTION TECHNIQUE	

La méthode de renvoi des valeurs possibles (serveur) prend en argument un « term » de recherche, et doit retourner une liste [id => val]. Elle est à implémenter par le code fonctionnel.

Warkham::autocomplete

Il s'agit d'un champ <input type=text> dont le contenu fait l'objet d'une aide d'autocompletion.

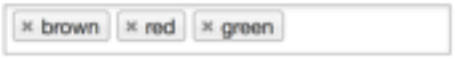
Méthode	Arguments	PHP	Javascript
setDataset(Array data)	data : un tableau sous la forme [id => value] qui liste les valeurs possibles.	A voir (dépend solution PHP -> JSON)	
setRemoteRoute(String route)	route : le nom de la route vers la méthode de renvoi asynchrone des valeurs possibles	positionne l'attribut data-remoteurl	gère l'envoi de la requête et le retour des valeurs trouvées
setTemplate()	gestion d'un template de présentation des valeurs possibles.	A DETAILLER SELON SOLUTION TECHNIQUE	

Dans le cas où seul le dataset est positionné, le chargement se fait une fois au départ ; si seul remoteRoute est positionné, le chargement est « live » (à la saisie) ; et enfin si les deux paramètres sont valués, dataset sert de « prefetch » (valeurs disponibles tout de suite, sans appel), et remoteRoute vient s'ajouter.

La méthode remote (code fonctionnel) a les mêmes caractéristiques que celle de Warkham::oracle.

Warkham::taglist

Le champ visible est représenté comme un input text contenant une liste de tags :



Ce champ est nommé avec un nom unique généré, et possède la class « wkm-taglist » ; un champ « hidden » est également créé avec le nom indiqué.

Le code javascript peut être basé sur http://ivaynberg.github.io/select2/.

NB : les méthodes setValue et forceValue prennent en argument une liste de tags séparés par une virgule.

Méthode	Arguments	PHP	Javascript
setTags(Array tags)	tags : liste de tags (String)	met à jour : - la valeur de l'attribut data-tags du champ text - et la valeur du champ hidden avec la liste des tags séparés par une virgule.	Initialise le champ avec les tags précisés.
setDataset(Array values)	values : un Array des tags disponibles force : si true, impossible de saisir un nouveau tag	positionne l'attribut dataset	propose les ces valeurs en fonction de ce que tape l'utilisateur
setMaxSelectionSize(int max)	max : nombre maximum de tags	positionne l'attribut data-maxselectionsize	gère une limite au nombre de tags
allowCreate([boolean c=true])	c : si true, la création de nouveaux tags et permise.	positionne l'attribut data-allowcreate	

Warkham::file

Créé un <input type='file'> avec un nom généré et la class « wkm-file ». Un champ « hidden » est créé avec le nom indiqué. Le code Javascript est chargé de gérer l'upload asynchrone, et de valuer le hidden avec le path du fichier uploadé. Quand un fichier est uploadé (ou que le champ est déjà valué) :

- le nom du fichier uploadé apparaît, ou sa vignette (selon le paramètre thumbnail)
- un bouton « supprimer » permet de réinitialiser le champ (effacer le nom ou la vignette et réafficher le champ d'upload)

Le code javascript peut être basé sur https://github.com/blueimp/jQuery-File-Upload/wiki/Basic-plugin.

Concernant le handler d'upload côté PHP, on peut se baser sur quelque chose comme https://github.com/blueimp/jQuery-File-Upload/blob/master/server/php/UploadHandler.php (à voir comment l'intégrer dans un Controller Laravel ?).

Méthode	Arguments	PHP	Javascript
accept(types)	types : soit un Array de mime ou d'extensions, soit « image », « video » ou « audio »	positionne l'attribut « accept »	
multiple([boolean m=true])		positionne l'attribut « multiple »	
max(int size, String unit)	size : entier unit : MB, Ko, TB, …	créé le hidden MAX_FILE_SIZE correspondant	
uploadRoute(String name)	name : nom de la route du Controller d'upload	met à jour l'attribut data-url	
thumbnail(int width, [int height, String className])	width : largeur de la vignette height : hauteur optionnelle className : class du tag <img> généré	positionne les attributs : - data-thumbnail (true / false) - data-thumbnailwidth - data-thumbnailheight - data-thumbnailclass	indique au Controller d'upload qu'il doit générer une vignette ; créé le tag <img> correspondant au retour
progress([boolean p=true])	p : true/false	positionne l'attribut data-progress à true / false	gère l'affichage d'une barre de progression

Warkham::date

Créé un input text avec un nom généré et la class « wkm-date », et un champ « hidden » avec le nom indiqué. Si le paramètre « hours » est à true, un <select> des heures est ajouté. Idem pour le paramètre « minutes ».

Le champ « hidden » doit être valué selon la norme RFC 2822[2].

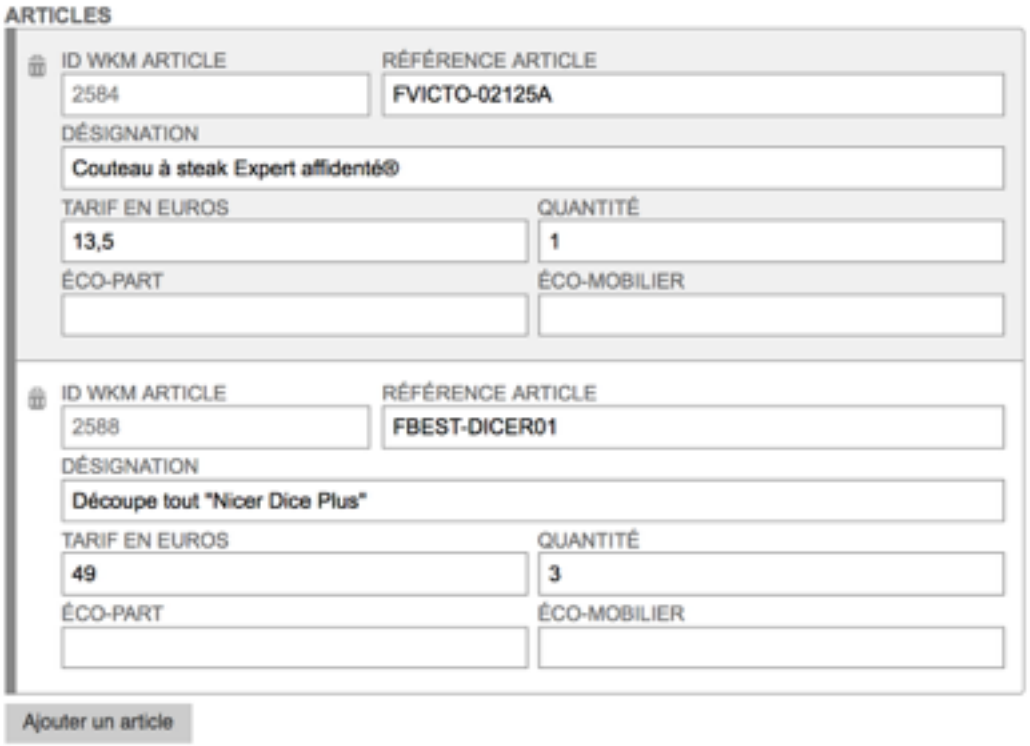
Le code Javascript est chargé d'afficher un calendrier au clic sur la zone, et de valuer le champ « hidden ».

Méthode	Arguments	PHP	Javascript
withHours(boolean h=true, [int min, int max])	h : true / false min : heure minimale sélectionnable max : heure maximale sélectionnable	Génère le <select> correspondant, avec la class « wkm-date-hours »	
withMinutes(boolean m=true, [int step=10, int min, int max])	m : true / false step : paliers d'affichage (toutes les 10 min. par défaut) min : minute minimale sélectionnable max : minute maximale sélectionnable	Génère le <select> correspondant, avec la class « wkm-date-minutes »	
min(String d)	d : une date sous la forme yyyy-mm-dd	positionne l'attribut data-min à la valeur indiquée	gère cette date comme minimum sélectionnable dans le composant calendrier
max(String d)	d : une date sous la forme yyyy-mm-dd	positionne l'attribut data-max à la valeur indiquée	gère cette date comme maximum sélectionnable dans le composant calendrier

[2] http://www.faqs.org/rfcs/rfc2822.html

Warkham::list

Ce composant est le plus complexe : il s'agit de gérer une liste d'items, chacun pouvant être constitué de plusieurs champs.



L'interface doit permettre (selon le paramétrage) d'ajouter, supprimer et déplacer les items. Les champs inclus dans un item peuvent être de n'importe que type Warkham.

Méthode	Arguments	PHP	Javascript
setValue(Array values)	values : un tableau sous la forme [idchamp => value]		
sortable([boolean m=true])	m : true / false	positionne l'attribut data-sortable	Gère le fait que les items soient triables en drag and drop ou non
addable([boolean m=true, String text="Ajouter"])	m : true / false text : le libellé du bouton d'ajout	- positionne l'attribut data-addable - créé le bouton d'ajout avec le bon libellé - ajoute le « template » d'item caché (voir plus bas)	gère l'ajout d'item
removeable([boolean m=true, String text="Supprimer"])	m : true / false text : le libellé du bouton de suppression	- positionne l'attribut data-removeable - créé le bouton de suppression avec le bon libellé sur chaque item - ajoute le bouton de suppression dans le template de création	gère la suppression d'item

Le code HTML généré doit correspondre à celui des « list group » Bootstrap (http://getbootstrap.com/components/#list-group).

Si la liste est créée avec ->addable(true), un caché avec comme id « {nom de la liste}-template » doit être créé juste après la liste, contenant le markup d'un item, qui servira au code JS pour la génération d'un item.

Warkham::filelist

Il s'agit d'une liste dont les items ne sont composés que d'un champ : un Warkham::file : c'est un simple raccourci.

Méthode	Arguments	PHP	Javascript
setValue(Array values)	values : un tableau sous la forme [idchamp => value] ; idchamp est facultatif, étant donné qu'il n'y a qu'un champ.		
sortable([boolean m=true])	Idem Warkham::list		
addable([boolean m=true, String text="Ajouter"])			
removeable([boolean m=true, String text="Supprimer"])			

Warkham::imagelist

Il s'agit d'une spécialisation de Warkham::filelist : les champs Warkham::file n'acceptent que des images.

Une autre différence se situe au niveau de l'affichage : plutôt qu'une liste verticale, les photos sont affichées en vignette, avec un seul champ « file »: