LEAD’S FACTORY

Documentation

Sommaire

Documentation LEADS FACTORY

Table des matières

[Introduction 5](#_Toc418421645)

[Guide fonctionnel 5](#_Toc418421646)

[Les Scopes 5](#_Toc418421647)

[Les Formulaires 5](#_Toc418421648)

[Les listes de références 5](#_Toc418421649)

[Les types 6](#_Toc418421650)

[Les statistiques 7](#_Toc418421651)

[Les Leads 8](#_Toc418421652)

[Les Exports 8](#_Toc418421653)

[Référence des tags utilisables 9](#_Toc418421654)

[Type Text 9](#_Toc418421655)

[Type TextArea 9](#_Toc418421656)

[Type Téléphone 9](#_Toc418421657)

[Type Email 9](#_Toc418421658)

[Type Checkbox 10](#_Toc418421659)

[Type Liste 10](#_Toc418421660)

[Type Listes liées (listes en cascade) 11](#_Toc418421661)

[Guide Technique 11](#_Toc418421662)

[Installation 11](#_Toc418421663)

[Installer Composer 11](#_Toc418421664)

[Mise à jour des ‘vendors’ 11](#_Toc418421665)

[Mise à jour du modèle de base de donnée 12](#_Toc418421666)

[Générer des données de démo 12](#_Toc418421667)

[Vider le cache de l’application 12](#_Toc418421668)

[Créer une liste de référence 13](#_Toc418421669)

[Cas d’une liste simple 14](#_Toc418421670)

[Cas d’une liste liée 15](#_Toc418421671)

[Créer un formulaire 16](#_Toc418421672)

[Configurer le formulaire 16](#_Toc418421673)

[Configurer les alertes et le monitoring 17](#_Toc418421674)

[Paramétrer le mail de confirmation 19](#_Toc418421675)

[Configurer un mail de notification 20](#_Toc418421676)

[Créer un nouveau type de champ de formulaire 21](#_Toc418421677)

[Paramétrer les validations de formulaire 21](#_Toc418421678)

[Insérez les formulaires dans une page de sites 21](#_Toc418421679)

[Pré-remplir des valeurs dans les champs. 21](#_Toc418421680)

[Utiliser les appels webservice 22](#_Toc418421681)

[Exclure des emails du domaine d’entreprise 22](#_Toc418421682)

[Créer une classe d’export 23](#_Toc418421683)

[Configuration du service d’export 23](#_Toc418421684)

[Création de la classe 24](#_Toc418421685)

[Paramétrage du formulaire 24](#_Toc418421686)

# Introduction

La lead’s factory est une application permettant la gestion des lead’s et la publication de ces données dans vos applications de back offices.

# Guide fonctionnel

## Les Scopes

Un scope est un ensemble de type ou formulaire. Les utilisateurs ne peuvent voir que le scope associé à leur compte. Cet attribut permet de séparer des sociétés dans un groupe.

## Les Formulaires

Le formulaire est une unité de contenu et de configuration qui sert à paramétrer le formulaire.

Un formulaire génère des lead’s qui génèrent des exports.

## Les listes de références

Une liste de référence est une série de données de ‘n’ niveaux permettant de remplir les listes des formulaires.

## Les types

Un type de formulaire est un attribut de regroupement des formulaires. Il permet de grouper les alertes et les statistiques des formulaires ensembles.

## Les statistiques

La lead’s factory permet de suivre des statistiques de pages vues / nombres de validations et taux de transformation des formulaires ou des groupes de formulaires.

## Les Leads

Un lead est un post de formulaire accépté par la lead’s factory. Ce sont les données brutes saisies sur le site.

## Les Exports

Un export est la conversion d’un lead pour un CRM. Un formulaire peux donc générer plusieurs leads qui vont eux meme générer plusieurs exports.

# Référence des tags utilisables

## Type Text

<field type="text" id="xxx"/>

**Description :**

C’est un champ texte à une ligne.

## Type TextArea

<field type="textarea" id="xxx"/>

**Description :**

C’est un champ texte multiligne de type textArea

**Aperçu :**

todo

## Type Téléphone

<field type="phone" id="xxx"/>

**Description :**

C’est un champ texte à une ligne utilisable pour la saisie d’un numéro de téléphone.

Attention, n’oubliez l’ajout des contrôles de validation sur ce champ.

**Aperçu :**

todo

## Type Email

<field type="email" id="xxx"/>

**Description :**

C’est un champ texte à une ligne utilisable pour la saisie d’un email.

Attention, n’oubliez l’ajout des contrôles de validation sur ce champ.

**Aperçu :**

todo

## Type Checkbox

<field type="checkbox" id="xxx"/>

**Description :**

C’est une case à cocher. Par défaut non cochée.

**Attribut possible :**

|  |  |
| --- | --- |
| @checked=’checked’ | Case cochée par défaut. |

**Aperçu :**

## Type Liste

<field type="reference-list" id="xxx" data-list="salutation"/>

**Description :**

Ceci affiche une liste de référence contenant les valeurs de la liste configurée dans le backoffice sous l’identifiant ‘salutation’.

La liste va pouvoir etre affichée sous une des formes suivantes :

* Liste déroulante type ‘select’
* Liste de case à cocher type ‘checkbox’
* Liste de choix type ‘options’

**Attribut possible :**

|  |  |
| --- | --- |
| @display | **select** : Affichage de type liste déroulante |
|  | **checkbox** : Affichage de type checkbox |
|  | **radio**: Affichage de type radio |

**Aperçu :**

## Type Listes liées (listes en cascade)

**Description :**

Les listes liées permettent la mise à jour en cascade de listes de références suivant les choix de l’utilisateur.

**Attribut possible :**

|  |  |
| --- | --- |
| @display | **select** : Affichage de type liste déroulante |
|  | **checkbox** : Affichage de type checkbox |
|  | **radio**: Affichage de type radio |
|  |  |
|  |  |

**Aperçu :**

# Guide Technique

## Installation

Téléchargement du projet vide de symfony

Configuration de la base de données

Initialisation de la base de données :

$ php app/console doctrine:database:create

## Installer Composer

$ php composer.phar update

## Mise à jour des ‘vendors’

Les vendors installés sur votre application peuvent être mis à jour via la commande suivante :

$ php composer.phar update

## Mise à jour du modèle de base de donnée

Le modèle de la base de donnée peut être simplement mis à jour en exécutant la commande suivante :

$ php app/console doctrine:schema:update --force

## Générer des données de démo

La commande suivante va générer des valeurs de démo du les différents formulaires configurés dans votre application

$ php app/console leadsfactory :demomode <formid>

- Formid est l’identifiant du formulaire.

## Vider le cache de l’application

$ php app/console cache :clear –env=prod

Videra le cache de prod. Omettre l’attribut –env videra l’ensemble des caches.

## Créer une liste de référence

Pour modifier une liste de référence dans la lead’s factory, il faut exporter son contenu en JSON puis le re-importer.

* Cliquez sur Générer le JSON pour Exporter.

Vous trouverez alors le fichier JSON dans le repertoire de travail de l’application « datas/json-lists »

* Cliquez sur Importer le JSON pour Importer vos modifications.

**Description du fichier JSON :**

### Cas d’une liste simple

**Le premier élément « lists »** permet de décrire les niveaux de listes utilisées dans le JSON. Dans le cas d’une liste simple, seul l’indice 0 va donner l’identifiant de la liste en édition. Le cas des listes liées est décrit plus tard dans la documentation.

**Le second élément « elements »** va décrire les valeurs de la liste.

Chaque valeur dispose d’un index de position comme clée, et contient possiblement trois valeurs :

**Id**: C’est l’identifiant de la valeur dans la lead’s. Si vous ajoutez une nouvelle valeur, vous omettrez simplement cette clée. Si vous ajoutez une clée ‘id’, lors de l’import la lead’s tentera de mettre à jour name et value de cette identifiant en base de donnée.

**name** : C’est la clée de votre valeur. Elle est utilisée dans l’attribut name des listes déroulantes.

**value :** C’est la valeur de votre entrée. Elle apparaît dans l’attribut value des listes déroulantes.

La suppression d’une entrée id/name/value, entrainera sa suppression en base de donnée.

### Cas d’une liste liée

Dans le cas d’une liste liée, le fichier JSON permet la modification de l’ensemble des listes.

Le niveau hiérarchique de liaison des listes doit etre indiqué dans la section « lists ».

Dans l’exemple, la liste principale est la liste avec l’identifiant 3, et la liste liée (liste de niveau 1) est la liste avec l’identifiant 2.

Les attributs faisant appel à la liste liée, doivent définir une entrée ‘children’, permettant de lister les valeurs liée au parent.

Une valeur de liste liée peut appartenir à « N » parent. Une liste ne peut posséder qu’une seule liste de niveau n-1.

La modification des valeurs et nom d’attributs se fait de la même façon qu’une liste simple.

## Créer un formulaire

Pour créer un formulaire il faut remplir les differents champs de la page « formulaire » :

|  |  |
| --- | --- |
| **Nom** | **Description** |
| **Name** | Nom du formulaire |
| **Description** | Description courte du formulaire |
| **Code** | Code technique relatif au formulaire. Ce code est utilisé dans les tags d’appels. |
| **Code de sécurité** | Code utilisable dans le webservice de consultation des données |
| **Code action par défaut** | Code action utilisée par l’application en cas d’absence de code présenté au TAG. |
| **Scope** | Portée du formulaire. Il ne sera visible que pour les utilisateurs du scope. |
| **Type de formulaire** | Groupement fonctionnel du formulaire |
| **Source pseudo HTML** | Source HTML du formulaire, remplacant les tags de formulaire par des champs @field de la lead’s factory |
| **Javascript** | Zone permettant de joindre un javascript au formulaire |
| **Configuration** | Configuration JSON du formulaire. |
| **Alertes** | Configuration JSON des alertes |
| **Confirmation Email** | Source HTML du mail de confirmation au client si besoin. |

Une fois enregistré, le formulaire est visible immédiatement des clients.

## Configurer le formulaire

|  |  |
| --- | --- |
| **Nom** | **Description** |
| **Export ‘obligatoire’** | Nom du plugin d’export à utiliser. Le paramétrage du plugin est relatif à ce dernier. |
| **notification ‘optionnel’** | Paramétrage d’un email de notification lors du post de formulaires.  **From :** ‘email expediteur’ : ‘nom de l’expediteur’  **To :** ‘email destinataire: ‘nom du destinataire’  Il est possible d’ajouter de multiples destinataires.  **Subject :**  Le mail contiendra la liste des champs du formulaire. |
| **redirect ‘obligatoire’** | **url\_success**: Redirection en cas de succès d’enregistrement de la demande  **url\_error** : Redirection en cas d’erreur d’enregistrement de la demande  **redirect\_with\_id**: (true|false). Si à true, la lead’s dirigera sur la page de succès en passant l’attribut **lead\_id** et **key** permettant de contact ensuite le webservice. |
| **confirmation\_email ‘optionnel’** | Email de confirmation déstiné au client.  **From :** ‘email expediteur’ : ‘nom de l’expediteur’  **To :** Configuration du destinataire  **Email\_input\_id :** Identifiant du champs de formulaire contenant l’email du client.  **Firstname\_input\_id :** Identifiant du champ de formulaire contenant le prénom du client  **Lastname\_input\_id :** Identifiant du champ contenant le nom de famille du client. |

## Configurer les alertes et le monitoring

Le calcul des alertes de monitoring fonctionne en comparant deux jours :

* La veille du jour courant.
* Le jour courant – 9 jours.

|  |  |
| --- | --- |
| **Nom** | **Description** |
| **rules** | Configuration encapsulant les éléments de paramétrage des alertes |
| **warning** | Zone définissant un niveau d’alerte pour attirer l’attention des utilisateurs sur le formulaire/type |
| **error** | Zone définissant un niveau d’erreur pour attirer l’attention des utilisateurs sur le formulaire/type |
| **min** | Valeur minimale du nombre de leads à avoir pour ne pas générer d’alerte/erreur. |
| **Max** | Valeur maximum du nombre de leads à ne pas dépasser pour ne pas générer d’alerte/erreur. |
| **delta** | Variation maximum avant de déclencher une alerte/erreur |

## Paramétrer le mail de confirmation

Le mail de confirmation est un email envoyé au client permettant de lui confirmer la bonne réception de sa demande.

**Configuration :**

|  |  |
| --- | --- |
| **Nom** | **Description** |
| **Confirmation\_email** | Section de la configuration permettant de paramétrer l’email de confirmation client. |
| **from** | Expediteur du mail. La première section contient l’email, la seconde le texte du nom à afficher par le client mail.  ‘email@domaine.fr’ :’Mon nom’ |
| **to** | Section de configuration du destinataire du mail. |
| **email\_input\_id** | Identifiant du champ de formulaire avec l’email du client. |
| **firstname\_input\_id** | Identifiant du champ de formulaire avec le prénom du client. |
| **lastname\_input\_id** | Identifiant du champ de formulaire avec le nom de famille du client. |

**Source HTML :**

Le mail envoyé au client est au format HTML. Il est possible d’inserer les champs de formulaire sous forme de variable en entourant leur identifiant de { et }.

## Configurer un mail de notification

Le mail de notification est un mail envoyé par la lead’s factory aux utilisateurs pour les informer de l’arrivée d’une nouvelle lead.

|  |  |
| --- | --- |
| **Nom** | **Description** |
| **notification** | Section de la configuration permettant de paramétrer l’email de confirmation client. |
| **from** | Expediteur du mail. La première section contient l’email, la seconde le texte du nom à afficher par le client mail.  ‘email@domaine.fr’ :’Mon nom’ |
| **to** | Section de configuration du destinataire du mail.  Chaque destinataire est indiqué avec un paramètre en deux parties :  La première section contient l’email, la seconde le texte du nom à afficher par le client mail.  ‘email@domaine.fr’ :’Mon nom’ |
| **subject** | Sujet du mail de notification |

## Créer un nouveau type de champ de formulaire

Un type de formulaire (ou groupement de formulaire) est un groupe permettant de regrouper fonctionnellement les leads. Le but est d’obtenir une vue dans les alertes et les statistiques plus précise.

## Paramétrer les validations de formulaire

Les validations de la lead’s factory sont coté client.

La librairie utilisée est : <http://posabsolute.github.io/jQuery-Validation-Engine/>

Reportez-vous directement à la documentation de JQuery validation engine pour comprendre le fonctionnement.

## Insérez les formulaires dans une page de sites

Inserez dans votre page la librairie de la leads et la librairie de validation JS, puis réalisez l’appel à la lead’s.

L’appel à la librairie javascript de la lead’s se réalise de la façon suivante :

<script type="text/javascript"

src="http://local.dev/leadsfactory/web/app\_dev.php/client/form/js/<code formulaire>/<code action>" >

</script>

Le tag prend en attributs :

* Le code du formulaire que vous avez indiqué dans le back office.
* Le code action de la page.

## Pré-remplir des valeurs dans les champs.

Pour chaque champs, la lead’s factory génère un getter et un setter permettant d’y accéder et de modifier son contenu.

Les getter et setter sont de la forme : set<Id> ou get<Id>. Prenons un identifiant ‘name’, nous aurons alors les methodes getName() et setName(value).

## Utiliser les appels webservice

Une application de votre SI peut lire des informations dans la lead’s via le webservice.

La route du webservice est : /lead/{id}/{key}

**Ou :**

* id est l’identifiant de la lead’s à lire.
* key est la clée fournie par la lead’s lors de la redirection avec le paramètre de configuration **‘redirect\_with\_id’**.

## Exclure des emails du domaine d’entreprise

Le paramètre de configuration suivant permet d’exclure les domaines d’emails des statistiques de la lead’s factory

## Créer une classe d’export

Une classe d’export permet de connecter un CRM (ou un outil equivalent) à la lead’s factory.

### Configuration du service d’export

Une déclaration de service doit être réalisée. Le service doit prendre un paramètre setContainer avec le container symfony.

Le service dispose d’un tag :

* nommé : export.method => Permettant à la lead’s factory d’identifier ce service comme un export.
* Alias : Le nom utilisé dans la configuration pour faire référence à ce service.

### Création de la classe

La classe du service doit simplement étendre la classe abstraite Tellaw\LeadsFactoryBundle\Utils\Export\AbstractMethod

Cette classe rend obligatoire l’implémentation de la methode ‘export ($jobs,$form)’ prenant en parametre deux objets :

* Le job actuel à exporter.
* Le formulaire source du job d’export.

### Paramétrage du formulaire

Dans le formulaire il faut référencer classe d’export par son alias de tag.

# API Taches planifiées

L'api de taches planifiées de la lead's factory permet la programmation et le monitoring de commandes symfony.

## Les commandes Symfony

Symfony 2 permet la création de commandes dans sa console. Ses commandes utilisent le framework et le container Symfony comme environnement d'execution.

Dans la lead's factory, une commande se trouve par convention dans un dossier **Command**.

C'est une classe qui etend généralement une des classes suivantes :

ContainerAwareCommand (Symfony\Bundle\FrameworkBundle\Command) Command (Symfony\Component\Console\Command)

Vous pouvez lister les commandes existantes en utilisant en ligne de commande :

php app/console

## Rendre un commande compatible

Toutes les commandes Symfony 2 sont compatibles avec le planificateur de taches de la lead's factory. Seul le chargement automatique dans la lead's factory demande la configuration d'un service supplémentaire (voir chapitre : Pré-configurer une tache)

## Pré-configurer une tache dans l'admin

Il suffit de créer un service implémentant l'interface **IScheduledJob (Tellaw\LeadsFactoryBundle\Utils)** taggé avec le tag **scheduled.job**

L'execution du cronjob de la lead's factory **CronRunnerCommand** va automatiquement initialiser ce nouveau Job.

**Exemple :**

Pour initialiser automatiquement une tâche planifiée avec monitoring sur le cronjob **StatusHistoryUpdateCommand**, le service suivant doit etre déclaré.

*Le service d'initialisation :*

<?php

namespace Tellaw\LeadsFactoryBundle\Utils\ScheduledJobs;

use Symfony\Bundle\FrameworkBundle\Routing\Router;

use Tellaw\LeadsFactoryBundle\Utils\IScheduledJob;

class StatusHistoryScheduler implements IScheduledJob

{

public function getExpression()

{

return "1 \* \* \* \* \*";

}

public function getName()

{

return "Core\_AlertStatusHistory\_Job";

}

public function getCommands()

{

return array ('leadsfactory:statusHistory:update');

}

public function getEnabled()

{

return true;

}

}

Après la première initilisation, les valeurs sont modifiables dans l'administration de la lead's. Pour éviter l'écrasement de paramétrage, seule la 'commande' sera mise à jour si sa valeur change dans le service d'initialisation.

## Les Crons & la planification

Le planificateur utilise un format cron expression pour parametrer l'interval d'execution de la tâche.

Certains intervals sont pré-configuré dans la lead's :

@yearly, @annually) - Run once a year, midnight, Jan. 1 - 0 0 1 1 \*

@monthly - Run once a month, midnight, first of month - 0 0 1 \* \*

@weekly - Run once a week, midnight on Sun - 0 0 \* \* 0

@daily - Run once a day, midnight - 0 0 \* \* \*

@hourly - Run once an hour, first minute - 0 \* \* \* \*

Syntaxe d'une cron expression :

Chaque entrée de la table (chaque ligne) correspond à une tâche à exécuter et doit respecter cette notation :

mm hh jj MMM JJJ tâche

mm représente les minutes (de 0 à 59)

hh représente l'heure (de 0 à 23)

jj représente le numéro du jour du mois (de 1 à 31)

MMM représente l'abréviation du nom du mois (jan, feb, ...) ou bien le numéro du mois (de 1 à 12)

JJJ représente l'abréviation du nom du jour ou bien le numéro du jour dans la semaine :

0 = Dimanche

1 = Lundi

2 = Mardi

...

6 = Samedi

7 = Dimanche (représenté deux fois pour les deux types de semaine)

Pour chaque valeur numérique (mm, hh, jj, MMM, JJJ) les notations possibles sont :

\* : à chaque unité (0, 1, 2, 3, 4...)

5,8 : les unités 5 et 8

2-5 : les unités de 2 à 5 (2, 3, 4, 5)

\*/3 : toutes les 3 unités (0, 3, 6, 9...)

10-20/3 : toutes les 3 unités, entre la dixième et la vingtième (10, 13, 16, 19)

*Extrait de Wikipedia*

## Les logs des commandes

Les logs des commandes sont visibles dans l'interface d'administration.