Référence SPIP/X/2020.002 – Ed. 1.1

GUIDE
CONCEPTION DU PLUGIN CHECK FACTORY

FICHE D'IDENTIFICATION	
Rédacteur	Eric Lupinacci
Projet	SPIP
Étude	Conception du plugin Check Factory
Nature du document	Guide
Date	12/11/2022
Nom du fichier	Guide - Le plugin Check Factory.docx
Référence	SPIP/X/2020.002 – Ed. 1.1
Dernière mise à jour	19/11/2022 13:49:20
Langue du document	Français
Nombre de pages	19

TABLE DES MATIERES

1. INTRODUCTION

Ce document a pour but de décrire les principes de base et les éléments de conception du plugin Check Factory (version 1.1.4 et ultérieures) dont l'objectif est de fournir des objets et des API génériques de gestion de contrôles et de leurs observations et de construction de dashboards de suivi.

Le plugin Check Factory fournit également une interface utilisateur limitée à l'administration des contrôles et des dashboards.

2. CONCEPTS

2.1 Les types de contrôle

2.1.1 Définition

Les types de contrôle sont des **fonctions** – au sens large – qui réalisent des **vérifications**, des **actions** ou des **états** et produisent des résultats sous forme d'**observations,** de **sorties physiques** (fichiers) et/ou d'**affichages**. Un type de contrôle peut concerner des objets ou des groupes d'objets du site sur lequel est installé le plugin Check Factory ou d'un site distant accessible, par exemple, au travers d'un service Web.

Un type de contrôle peut être associé à un fichier YAML, `type_controle.yaml`, qui décrit l'ensemble de ses caractéristiques. Son identifiant est alors le nom du fichier YAML sans extension (par exemple, `contrib_rubrique_prefixe`). L'identifiant doit être composé de caractères acceptables pour un nom de fonction PHP, donc en particulier éviter le tiret. Les fichiers YAML doivent être stockés dans le dossier `ezcheck/controles/` d'un plugin utilisateur.

Sinon, il est aussi possible de décrire les caractéristiques d'un type de contrôle via le pipeline `eztypecontrole_declarer` (voir paragraphe xx).

2.1.2 Composants d'un type de contrôle

Un type de contrôle est défini par :

- son **identification**, à savoir, son identifiant unique, son nom, sa description et son logo éventuel ;
- une éventuelle **fonction PHP d'exécution** du contrôle pour laquelle il est possible de préciser le nom, la localisation (fichier PHP) et des arguments qui seront fournis via un formulaire.
- une liste éventuelle d'**anomalies** (observation de type anomalie) pouvant être produites par le contrôle et pour lesquelles il est possible d'associer une fonction de correction automatique ;
- un **affichage complémentaire** définit par un squelette HTML auquel il est possible d'adjoindre un contexte fixe mais aussi une liste de paramètres fournis via un formulaire ;
- une liste de saisies de **paramètres** attachés soit à la fonction d'exécution, soit à l'affichage complémentaire.

Ces éléments font partie de la configuration du type de contrôle. La plupart sont facultatifs. Un type de contrôle sans fonction d'exécution ni paramètres est appelé un **état** : il se contente d'un affichage.

2.1.3 Fonctions d'exécution et de correction

Si le type de contrôle est basé sur une fonction PHP, celle-ci est identifiée par son nom et le fichier dans lequel elle est définie. Il en va de même pour les fonctions de corrections.

Par défaut, la fonction d'exécution porte un nom qui coïncide avec l'identifiant de son type de contrôle. Dans ce cas, il suffit d'indiquer la localisation de la fonction.

Les fonctions de correction possèdent un nom imposé composé de l'identifiant du type de contrôle et de l'identifiant de l'anomalie à corriger.

2.2 Les contrôles

Un contrôle est une **« instance d'un type de contrôle » exécutée à un instant t** suite à une demande d'un utilisateur autorisé ou à l'échéance d'un CRON (non disponible dans la version actuelle).

Outre son type, le contrôle possède un **statut d'exécution** qui rend compte du déroulement correct ou pas de la fonction associée. Ce statut n'a rien à voir avec les observations éventuellement générées pendant l'exécution du contrôle.

Tout contrôle possède un identifiant unique `id_controle`. Il est aussi possible d'identifier de façon unique un contrôle avec le type de contrôle auquel il est associé, la date d'exécution et l'auteur, mais cette identification est peu utilisée.

Tout type de contrôle possédant une fonction d'exécution ou à minima un affichage complémentaire avec des paramètres génère des contrôles. Seuls les états ne génèrent pas de contrôle car ils ne font qu'afficher un squelette HTML.

2.3 Les observations

Les observations sont des **logs qui consignent des résultats de l'exécution du contrôle**. Les observations ne sont générées que par la fonction d'exécution PHP au travers de l'API fournie par le plugin Check Factory. Leur utilisation est facultative.

Une observation possède un identifiant unique `id_observation` mais aussi un code unique au sein d'un type de contrôle donné. Un type de contrôle peut générer différents types d'observation au travers de ses vérifications.

Une observation est associée à un contrôle et concerne un objet au sens large (objet SPIP ou tout autre élément observable).

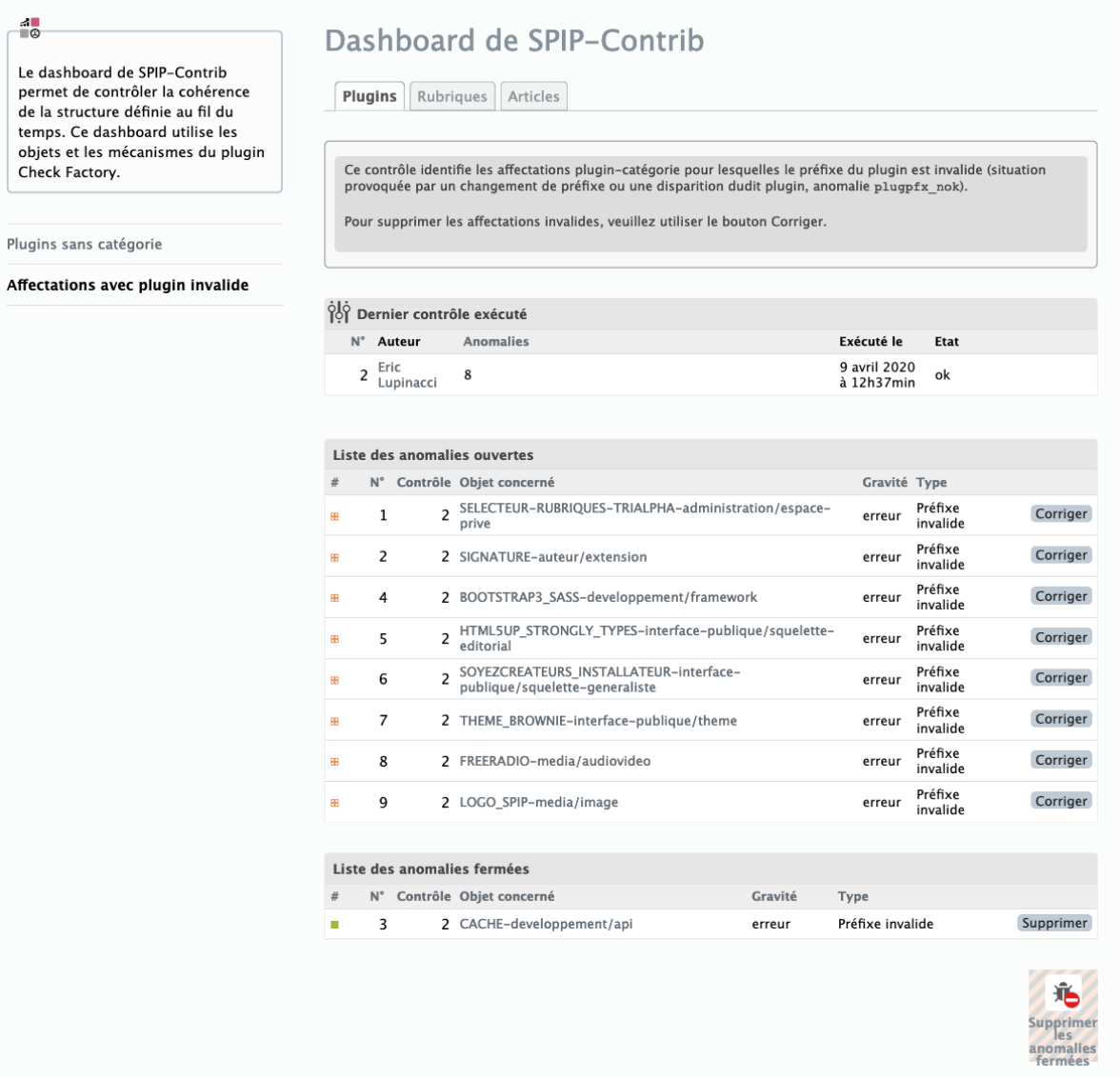
Une **anomalie** est une observation particulière qui possède un statut ouverte ou fermée et qui peut être soit acquittée manuellement soit corrigée automatiquement par l'exécution d'une fonction spécifique de correction : elle passe alors dans l'état fermée. Le suivi des anomalies est une fonction essentielle du plugin Check Factory.

2.4 Les dashboards

Un dashboard est une page HTML qui organise les types de contrôle, permet leur exécution, affiche les résultats et facilite la gestion des anomalies. Il est décrit par un fichier de configuration YAML ou JSON dont le nom coïncide avec l'identifiant du dashboard lui-même.

Les fichiers YAML ou JSON doivent être stockés dans le dossier `ezcheck/dashboards/` d'un plugin utilisateur.

Il est possible de créer autant de dashboard que l'on souhaite et d'y associer tous les types de contrôle nécessaires. Pour simplifier la présentation, les types de contrôle sont organisés par groupe (onglets). Un exemple de dashboard est fourni ci-dessous.



3. PERIMETRE DU PLUGIN CHECK FACTORY

Check Factory propose plusieurs API :

- La gestion des types de contrôle, à savoir, le chargement des fichiers YAML (ou du pipeline), leur stockage, la lecture des informations stockées et l'exécution des contrôles ;
- La gestion des dashboards, à savoir, le chargement, leur stockage et la lecture des informations de configuration ;
- La gestion des observations et, en particulier, la création des anomalies et leurs évolutions de statut jusqu'à la clôture ainsi que la gestion des corrections.

Un plugin utilisateur de Check Factory se contente en général de l'API des observations qu'il utilise dans ses fonctions de contrôle. Le reste des API n'est utile que si le plugin souhaite modifier la présentation des dashboards ou la gestion par défaut des concepts de Check Factory.

Check Factory propose également une interface minimale dans l'espace privé pour recharger les dashboards et les types de contrôle et aussi activer ou désactiver temporairement des types de contrôle.

3.1 L'API de gestion des types de contrôle

La gestion des types de contrôle consiste à stocker les descriptions en base de données et à permettre leur lecture, leur mise à jour et leur exécution au travers d'un contrôle.

API TYPES DE CONTROLES : INC/EZCHECK_TYPE_CONTROLE.PHP	
type_controle_charger	Charge ou recharge les descriptions des types de contrôle à partir des fichiers YAML ou du pipeline `eztypecontrole_declarer`. Les fichiers YAML sont recherchés dans un répertoire relatif `ezcheck/controles/`. La fonction optimise le chargement en effectuant uniquement les traitements nécessaires en fonction des modifications, ajouts et suppressions des types de contrôle identifiés en comparant les md5 des descriptions complètes.
type_controle_lire	Retourne, pour un type de contrôle, la description complète ou une liste de champs donnée. Les champs de type tableau sont fournis désérialisés.
type_controle_repertorier	Renvoie une liste de types de contrôle éventuellement filtrée sur certains champs. Les données sont renvoyées brutes sous forme d'un tableau indexé par l'identifiant de chaque type de contrôle. Il est possible de récupérer toute la description ou un seul champ pour chaque type de contrôle.

type_controle_executer	Exécute une fonction de contrôle d'un type de contrôle donné. Si la fonction n'existe pas c'est que l'exécution ne sert qu'à enregistrer les paramètres du formulaire servant uniquement à l'affichage d'un squelette. Dans tous les cas, cette fonction crée un contrôle du type concerné.
type_controle_identifier_dashboard	Renvoie l'identifiant du dashboard auquel est rattaché le type contrôle ou chaine vide sinon.

Check Factory propose une balise **`#TYPE_CONTROLE_DASHBOARD`** qui fournit l'identifiant du dashboard auquel est rattaché le type de contrôle. Cette balise est une encapsulation de la fonction d'API `type_controle_identifier_dashboard()`.

Le prototype de la balise est le suivant : `#TYPE_CONTROLE_DASHBOARD{type_controle}`.

3.2 L'API de gestion des observations

La gestion des observations consiste à gérer leur cycle de vie, de leur ouverture à leur clôture.

API OBSERVATIONS : INC/EZCHECK_OBSERVATION.PHP	
observation_ajouter	Crée une observation avec tous ses paramètres en base de données. Une observation de type anomalie se voit affecté d'un statut 'ouverte'.
observation_repertorier	Renvoie une liste d'observations éventuellement filtrée sur certains champs. Les données sont renvoyées brutes sous forme d'un tableau indexé par l'identifiant de chaque observation. Il est possible de récupérer toute la description ou un seul champ pour chaque observation.
observation_cloturer	Effectue les traitements adéquats pour clore une observation : acquitte une anomalie, corrige une anomalie ou supprime une observation. Acquitter et supprimer sont des actions qui ne requièrent que la mise à jour du statut. L'action corriger elle nécessite l'appel à la fonction de correction configurée.

3.3 L'API de gestion des dashboards

La gestion des dashboards consiste essentiellement à charger leur description et à les restituer sur demande au travers d'une fonction d'API ou d'une balise.

API DASHBOARDS : INC/EZCHECK_DASHBOARD.PHP	
dashboard_charger	Charge ou recharge la configuration des dashboards à partir de leur fichier YAML. Les fichiers YAML sont recherchés dans un répertoire relatif `ezcheck/dashboards/`. La fonction compile les dashboards dans un cache unique sécurisé.
dashboard_lire	Retourne la description complète du dashboard. Les champs textuels peuvent subir un traitement typo si demandé.
dashboard_repertorier	Renvoie l'information brute demandée pour l'ensemble des dashboards ou toute les descriptions si aucun champ n'est explicitement demandé. Les données sont renvoyées sous forme d'un tableau indexé par l'identifiant de chaque dashboard.
dashboard_contecxtualiser	Renvoie la configuration complète d'un dashboard ainsi que ainsi que des informations supplémentaires provenant de l'environnement de la page dashboard (groupe et type de contrôle à afficher par défaut).

Territoires propose aussi des balises utilisables dans l'espace public, à savoir :

- `#DASHBOARD` qui fournit, dans un tableau, la configuration complète d'un dashboard donné ou de tous les dashboards.
- `DASHBOARD_CONTEXTE` qui fournit, dans un tableau, la configuration complète d'un dashboard ainsi que ainsi que des informations supplémentaires provenant de l'environnement de la page dashboard. Cette balise est une encapsulation de l'API `dashboard_contextualiser()`.

Les prototypes des balises sont :

```
#DASHBOARD{[dashboard]}.
```

```
#DASHBOARD_CONTEXTE{dashboard[, groupe, type_controle]}.
```

3.4 L'interface utilisateur dans l'espace privé

3.4.1 L'administration des types de contrôle

L'interface utilisateur du plugin Check Factory propose, dans l'espace privé, une page de gestion sommaire des types de contrôle, `ezcheck&vue=type_controle`. Elle permet :

- d'afficher la liste des types de contrôle ;
- d'activer ou de désactiver temporairement des types de contrôle ;
- de recharger la liste des types de contrôle.

Chaque type de contrôle est présenté avec son identifiant, son libellé et, éventuellement, sa description et son icone. Si une erreur est détectée dans la configuration du type de contrôle (YAML ou pipeline), un message d'erreur est affiché dans la zone du type de contrôle.

3.4.2 L'administration des dashboards

L'interface utilisateur du plugin Check Factory propose, dans l'espace privé, une page de gestion sommaire des dashboard, `ezcheck&vue=dashboard`. Elle permet :

- d'afficher la liste des dashboards ;
- de recharger la liste des dashboards.

Chaque dashboard est présenté avec son identifiant, son libellé et, éventuellement, sa description et son icone.

3.4.3 Le dashboard d'exécution des contrôles

La page `dashboard` est la page d'exécution des types de contrôle et de gestion des anomalies pour un type de contrôle donné. Elle est construite automatiquement à partir des informations collectées dans les fichiers de configuration ou au travers des pipelines de chargement des dashboards et des types de contrôles.

La page est organisée avec un menu d'onglets, un onglet représentant un groupe de types de contrôle, un menu des types de contrôle du groupe sélectionné et une partie centrale fournissant l'ensemble des éléments constitutifs du contrôle exécuté ou à exécuter.

La partie centrale est composée de différents éléments distincts concernant l'exécution du contrôle et les résultats de l'exécution. Les éléments présentés diffèrent selon la configuration du type de contrôle mais on distinguer :

- Le **formulaire de lancement de l'exécution**, qui affiche toujours une explication sur le contrôle et, éventuellement, une liste de saisies nécessaires pour exécuter le contrôle et un bouton de lancement de l'exécution d'un nouveau contrôle ;
- Le **dernier contrôle** effectué avec, en particulier, la date et le statut d'exécution ;
- La **liste des contrôles précédents** dont les anomalies n'ont pas toutes été résolues (dépend du type de contrôle et du contexte d'exécution) ;
- La liste des **anomalies ouvertes** (dépend du type de contrôle et du contexte d'exécution);
- La liste des **anomalies fermées** et non encore effacées (dépend du type de contrôle et du contexte d'exécution) ;
- Un **affichage spécifique** optionnel fournissant des résultats divers sur l'exécution du contrôle au travers d'un squelette.

4. FONCTIONNEMENT DE CHECK FACTORY

4.1 La chargement des configurations

4.1.1 Les types de contrôle

Tbc.

4.1.2 Les dashboards

Tbc.

4.2 L'interface d'exécution des types de contrôle

Tbc.

5. DONNEES DE CHECK FACTORY

5.1 La structure des données

5.1.1 Les types de contrôle

Un type de contrôle est un objet SPIP matérialisé par un enregistrement dans la table `spip_types_controles`. Les champs de cette table sont listés ci-dessous avec leur valeur par défaut.

TABLE SPIP_TYPES_CONTROLES	
Données de base issues du fichier YAML ou du pipeline	
identifiant	Identifiant de l'objet type de contrôle au format chaine [a-z0-9_] ; correspond au nom du fichier de configuration YAML ou de l'index du tableau issu du pipeline `eztypecontrole_declarer`.
nom	Libellé du type de contrôle. Par défaut, est initialisé avec l'identifiant.
description	Texte descriptif du type de contrôle.
icone	Icône type de contrôle au format png ou svg. Si aucun icone n'est défini, l'icône `contrôle_defaut-24.svg` est utilisé.
necessite	Liste des préfixes des plugins nécessités par le type de contrôle pour assurer son exécution ou tableau vide sinon. Le tableau est stocké sérialisé.
include	Chemin relatif du fichier contenant la fonction de contrôle ou chaine vide sinon.
fonction	Nom de la fonction d'exécution. Par défaut, en l'absence de précision, coïncide avec l'identifiant du type de contrôle.
parametres	Paramètres à fournir en entrée de la fonction ou du squelette sous forme de saisies. Possède à minima deux index, l'un pour les paramètres de la fonction, 'fonction', l'autre pour les paramètres du squelette, 'squelette'. Le tableau est stocké sérialisé.
anomalies	Liste des identifiants d'anomalies rangée suivant les actions acquitter ou corriger et chemin de l'include des fonctions de correction. Le tableau donc les index 'acquitter', 'corriger' et 'include'. Le tableau est stocké sérialisé.
squelette	Chemin relatif du squelette HTML permettant un affichage complémentaire spécifique au contrôle ou chaine vide sinon.
contexte	Tableau de contexte spécifique au squelette.
Données complémentaires calculées lors du chargement	
est_etat	Indique que le type de contrôle est un état.

actif	Indicateur d'activité du contrôle. Si différent de 'oui', aucun contrôle de ce type ne peut être réalisé. Dans ce cas, la valeur permet de déterminer le type de l'erreur constatée au chargement qui peut valoir : • 'not' : squelette spécifique introuvable • 'nof' : fonction d'exécution introuvable • 'noc' : fonction de correction introuvable • 'nop' : plugin nécessité désactivé • 'nok' : autre erreur de configuration
signature	MD5 de la description du type de contrôle (YAML ou pipeline).
maj	Timestamp automatique de l'objet.

5.1.2 Les contrôles

Un contrôle est un objet SPIP matérialisé par un enregistrement dans la table `spip_controles`. Les champs de cette table sont listés ci-dessous avec leur valeur par défaut.

TABLE SPIP_CONTROLES	
id_controle	Identifiant incrémental du contrôle.
type_controle	Identifiant du type de contrôle dont le contrôle est une instance.
parametres	Tableau, éventuellement vide, définissant les valeurs des paramètres du type de contrôle saisis dans le formulaire d'exécution.
date	Date de l'exécution
etat_execution	Statut retourné par la fonction d'exécution. Vaut 'exec_ok' ou 'exec_nok_defaut'.
id_auteur	Identifiant de l'auteur ayant exécuté le contrôle ou 0 si le lancement a été automatique (CRON, non disponible actuellement).
nb_anomalies	Nombre d'anomalies encore dans l'état ouvert.
maj	Timestamp automatique de l'objet.

5.1.3 Les observations

Une observation est un objet SPIP matérialisé par un enregistrement dans la table `spip_observations`. Les champs de cette table sont listés ci-dessous avec leur valeur par défaut.

TABLE SPIP_OBSERVATIONS	
id_observation	Identifiant incrémental de l'observation.
id_controle	Identifiant du contrôle ayant remonté l'observation.

type_controle	Identifiant du type de contrôle. Cette information est reprise du contrôle afin d'optimiser les boucles.
objet	Type d'élément sur lequel porte l'observation. Peut désigner un type d'objet SPIP ou un nom quelconque sinon.
id_objet	Identifiant de l'objet SPIP sur lequel porte l'observation ou 0 sinon.
url_objet	URL de l'élément sur lequel porte l'observation si celui-ci n'est pas un objet SPIP.
est_anomalie	Indique si l'observation est une anomalie, 'oui', ou pas, 'non'.
gravite	Indique si l'observation est une information, 'i', ou une erreur 'e'. Une anomalie est toujours considérée comme une erreur.
code	Identifiant textuel relatif de l'observation.
statut	Statut de l'observation. Une anomalie est créée avec le statut 'ouverte', peut passer à 'fermee' après correction ou acquittement puis être mise à la poubelle. Une observation non constitutive d'une anomalie est créée avec le statut 'fermee' directement car c'est juste un log.
parametres	Tableau, éventuellement vide, permettant d'expliquer précisément l'anomalie en fournissant des paramètres utilisables dans l'item de langue explicatif.
date	Date du passage dans le statut courant de l'observation.
maj	Timestamp automatique de l'objet.

5.1.4 Les dashboards

La description d'un dashboard est structurée dans un tableau associatif dont tous les champs possibles sont initialisés.

DESCRIPTION D'UN DASHBOARD	
Structure générale	
identifiant	Identifiant du dashboard au format chaine [a-z0-9_] ; correspond au nom du fichier de configuration YAML ou de l'index du tableau issu du pipeline `ezdashboard_declarer`.
nom	Libellé du dashboard. Si non défini est initialisé avec l'identifiant.
description	Texte descriptif du dashboard.
icone	Icône du dashboard png ou svg. Si aucun icone n'est défini, l'icône `dashboard_defaut-24.svg` est utilisé.
groupes	Tableau de configuration des groupes. Le tableau est indexé par l'identifiant de chaque groupe (cf. note [1]).

Structure du bloc groupes[1]	
identifiant	Identifiant du groupe au format chaine [a-z0-9_].
nom	Libellé du groupe qui est affiché comme titre de l'onglet correspondant.
controles	Liste des identifiants des types de contrôle associés au groupe.

5.2 Les espaces de stockage

Check Factory utilise la base de données SPIP comme espace de stockage privilégié. Ainsi, les types de contrôles, les contrôles et les observations sont stockées dans une table SPIP nommée respectivement `spip_types_controles`, `spip_controles` et `spip_observations`.

Les dashboards ne sont pas stockés en base de données mais dans un cache sécurisé géré par le plugin Cache Factory et nommé `tmp/dashboards.php`. Les données du cache sont sérialisées et ne sont jamais périmées ; c'est à l'utilisateur de recharger manuellement les dashboards.

6. MISE EN PLACE D'UN DASHBOARD UTILISATEUR

6.1 Configurer un type de contrôle

Tbc.

7. REGLES DE CODAGE

7.1 Nommage des fonctions

Le nommage des fonctions appartenant à l'API de Check Factory suit des règles strictes qui simplifient l'identification de l'objet et de l'action appliquée. Le nom de chaque fonction est donc composée ainsi : **`<objet>_<verbe_infinitif>`**. Par exemple, la fonction de lecture d'un type de contrôle se nomme `type_controle_lire()`.

7.2 Arguments et variables standardisés

Les arguments ou variables locales dépendent de chaque fonction mais leur nommage est toujours le même d'une fonction à une autre.

Par exemple, l'argument `$id_type_controle` désigne toujours l'identifiant d'un type de contrôle. De façon identique, les identifiants de contrôle, d'observation ou de dashboard, qu'ils soient numériques ou chaines de caractères seront toujours précédés du préfixe `$id_` suivi du nom de l'objet en question.

Les variables ou arguments désignant la description de l'objet lui-même sont nommées avec le nom de l'objet comme `$type_controle`. Un tableau de descriptions est nommés de la même façon avec un caractère `s` final comme `$dashboards`.