SPIP

SPIP **24/08/2017**

Référence SPIP/N/2017.002

GUIDE
LE PLUGIN N-CORE

SPIP

FICHE D'IDENTIFICATION

Rédacteur	Eric Lupinacci
Projet	SPIP
Étude	Le plugin N-Core
Nature du document	Guide
Date	24/08/2017
Nom du fichier	Guide N - Le plugin N-Core.docx
Référence	SPIP/N/2017.002
Confidentialité	
Langue du document	Français
Nombre de pages	16

SPIP

TABLE DES MATIERES

1. INTRODUCTION

Ce document a pour but de décrire les principes de base et les éléments de conception du plugin N-Core dont l'objectif est de fournir des API génériques de gestion des noisettes et de leur compilation dans un squelette.

N-Core ne fournit aucune interface utilisateur. Le noiZetier, dans sa version 3, utilise N-Core et offre une interface d'administration permettant de configurer et d'insérer au choix des noisettes dans les diverses pages publiques du site.

D'autres plugins pourront ainsi être développés à partir de N-Core en particulier pour fournir des interfaces utilisateur pour associer des noisettes et des squelettes.

2. PERIMETRE DE N-CORE

N-Core propose principalement 4 API :

- La gestion des types de noisettes, à savoir, le chargement des fichiers YAML, leur stockage et la lecture des informations stockées ;
- La gestion des noisettes, à savoir, un « CRUD-like » avec un stockage dédié ;
- La compilation des noisettes, à savoir, la gestion du contexte et des paramètres de chaque noisette et leur inclusion dans les squelettes au travers du pipeline `recuperer_fond()` ;
- La gestion de fichiers caches pour le stockage à accès rapide des types de noisettes et de certains éléments de contexte.

2.1 L'API de gestion des types de noisette

La gestion des types de noisette consiste à stocker les descriptions dans un espace à accès rapide et à permettre leur lecture et leur mise à jour.

API TYPES DE NOISETTE : INC/NCORE_TYPE_NOISETTE.PHP	
type_noisette_charger	Charge ou recharge les descriptions des types de noisette à partir des fichiers YAML. Les types de noisettes sont recherchés dans un répertoire relatif fourni en argument. La fonction optimise le chargement en effectuant uniquement les traitements nécessaires en fonction des modifications, ajouts et suppressions des types de noisettes identifiés en comparant les md5 des fichiers YAML.
type_noisette_lire	Retourne, pour un type de noisette, la description complète ou seulement un champ précis. Les champs textuels peuvent être fournis bruts ou avec un traitement typo.

type_noisette_repertorier	Renvoie une liste de types de noisette éventuellement filtrée sur certains champs.

2.2 L'API de gestion des noisettes

L'API de gestion des noisettes fournit une interface de type « CRUD étendue » pour associer des instances de type de noisette à un squelette HTML. L'interface utilisateur permettant le choix du squelette et des types de noisette ne fait pas partie de N-Core.

API NOISETTES : INC/NCORE_NOISETTE.PHP	
noisette_ajouter	Ajoute à un squelette, à un rang donné ou en dernier rang, une noisette d'un type donné. Si le rang n'est pas précisé, la noisette est ajoutée en dernier rang. La fonction renvoie si tout s'est bien passé l'identifiant unique de la noisette.
noisette_lire	Retourne, pour une noisette, la description complète ou seulement un champ précis.
noisette_parametrer	Met à jour les paramètres de configuration de la noisette destinés à son affichage.
noisette_supprimer	Supprime une noisette donnée et met à jour les rangs des autres noisettes si nécessaire.
noisette_deplacer	Déplace un noisette de sa position à une position destinatrice et met à jour les rangs des autres noisettes si nécessaire.
noisette_vider_grappe	Supprime toutes les noisettes associée à une grappe.

2.3 L'API de compilation des noisettes

L'API de compilation des noisette fournit une interface très simplifiée permettant de... à compléter.

API NOISETTES : INC/NCORE_COMPILATION.PHP	
noisette_repertorier	
noisette_contextualiser	

2.4 L'API de gestion des caches

La gestion des caches est principalement à usage interne N-Core car celui-ci utilise de nombreux caches comme moyen de stockage. Néanmoins, cette interface est utilisable aussi par d'autres plugins.

API CACHES : INC/NCORE_CACHE.PHP	
cache_lire	Lit le cache spécifié et renvoie le contenu sous forme de tableau éventuellement vide.
cache_ecrire	Ecrit le contenu d'un tableau dans le cache spécifié.
cache_supprimer	Supprime le cache spécifié.

3. CONCEPTS

3.1 Les squelettes

La mise en page d'un site SPIP est effectuée au moyen de **gabarits au format HTML** nommés **squelettes**, contenant des instructions simplifiées permettant d'indiquer où et comment se placent les informations tirées de la base de données dans la page web.

Un squelette est donc un fichier HTML installé dans un site SPIP qui affiche une page comme `article.html` ou une partie d'une page web comme `content/article.html`.

L'identifiant d'un squelette est son chemin relatif sans extension (par exemple content/article).

3.2 Les types de noisettes

Les types de noisette sont les composants de base de N-Core. Un **type de noisette est un squelette HTML** autonome, suffisamment **générique** pour être **réutilisable** dans différentes pages ou sites et pouvant être **configurable**. Il est toujours associé à un fichier YAML qui décrit l'ensemble de ses caractéristiques.

L'identifiant d'un type de noisette est le nom du fichier associé sans extension (par exemple, article-cartouche).

3.3 Les noisettes

Une noisette est une **« instance paramétrée d'un type de noisette » incluse dans un squelette** donné. L'inclusion se fait soit en fin du squelette, soit à un endroit précis repéré par un pattern particulier.

Outre son type, la noisette se distingue par son paramétrage (valeur de chaque paramètre attaché au type de noisette) et par un ensemble d'informations de contexte qui permettent de la compiler lors de son inclusion dans le squelette.

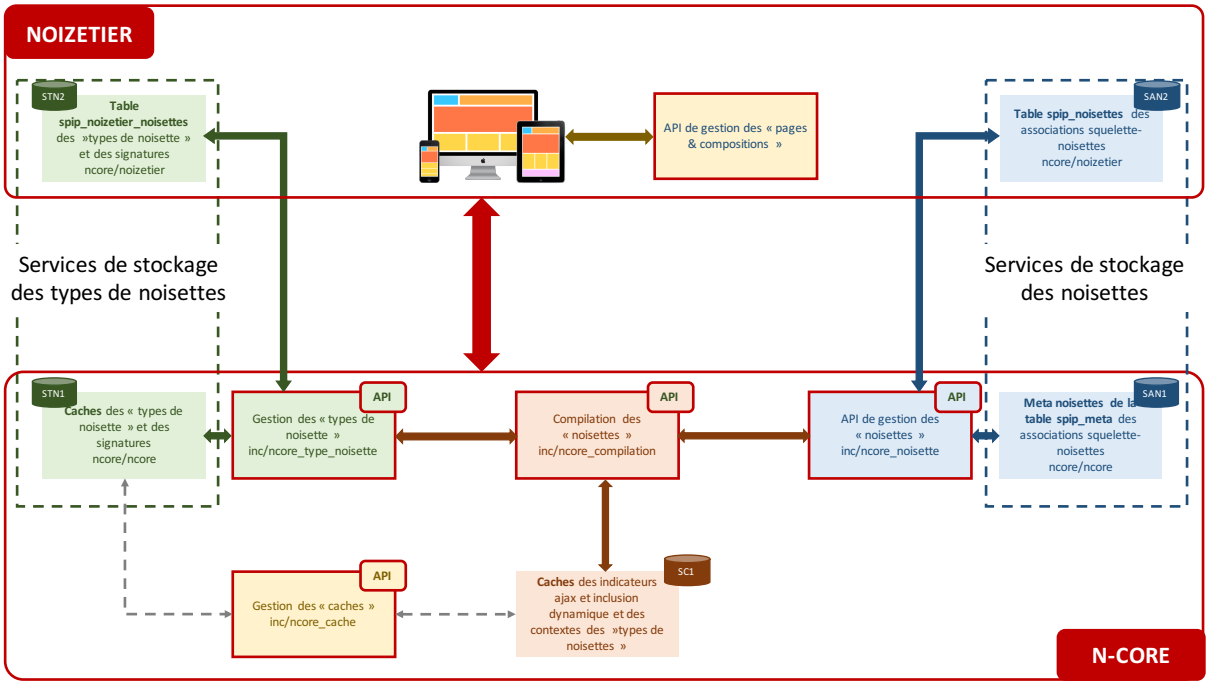
Il est possible d'inclure plusieurs noisettes à la suite dans un squelette (voire de construire complètement le squelette avec des noisettes). Il est donc nécessaire de gérer un rang pour chaque noisette incluse dans un squelette.

Toute noisette possède un identifiant unique qui se nomme `id_noisette` et qui peut être un entier ou une chaine de caractères pour autant qu'il soit unique pour une utilisation donnée. Il est aussi possible d'identifier une noisette avec le squelette auquel elle est associée et son rang. L'utilisation de l'un ou l'autre dépend de la structure de stockage adoptée.

4. FONCTIONNEMENT DE N-CORE

4.1 Schéma de principe

Le fonctionnement global du plugin N-Core est illustré ci-dessous en regard de ce que pourrait être le plugin noiZetier v3.



4.2 La dissociation API – Services

De façon générale, un plugin utilisateur comme le noiZetier va s'appuyer sur l'ensemble des API publiques N-Core (types de noisette, noisettes et compilation). Néanmoins, un plugin utilisateur pourrait se passer des API de gestion des types de noisette et noisettes mais pas de l'API de compilation.

Si un plugin utilisateur choisit d'utiliser les API de gestion, il doit définir le stockage qu'il souhaite pour ses types de noisette ou ses noisettes. Par conception, **N-Core de dissocie la fonction de gestion d'un objet de l'espace et des services de stockage de ce même objet**.

Dans le code de ses API, N-Core appelle des fonctions de services qui :

- si la fonction de service homonyme existe dans le plugin utilisateur, va l'appeler et utiliser le stockage propre au plugin ;
- sinon, va dérouler la fonction de N-Core avec son propre stockage.

Mais N-Core va plus loin en permettant à un **plugin utilisateur d'exposer ses fonctions de service et leur stockage comme une librairie**. Le stockage et les fonctions de service d'un plugin utilisateur sont alors réutilisables par un autre plugin utilisateur à l'instar de celui de N-Core. Par exemple, un plugin

« préfixe » pourrait utiliser le stockage des noisettes du plugin noiZetier, à savoir, la table « spip_noizetier ». Cette fonctionnalité impose bien entendu des contraintes aux fonctions de service qui sont détaillées ci-après.

Toute fonction d'API de N-Core possède deux arguments incontournables, `$plugin` qui est obligatoire et `$stockage` qui est optionnel, comme on peut le voir sur le prototype de `type_noisette_charger()` :

```
function type_noisette_charger($plugin, $dossier = 'noisettes/', $recharger = false, $stockage = '')
```

L'argument $plugin qualifie le module appelant, généralement un plugin comme le noiZetier. Il est donc recommandé d'utiliser le **préfixe du plugin** comme identifiant unique. Cet argument permet de distinguer les espaces de stockage d'un plugin utilisateur par rapport à d'autres. Par exemple, N-Core utilise un cache pour stocker les types de noisette dont le chemin dans `_DIR_CACHE` est `ncore/${plugin}/types_noisette_description.php`.

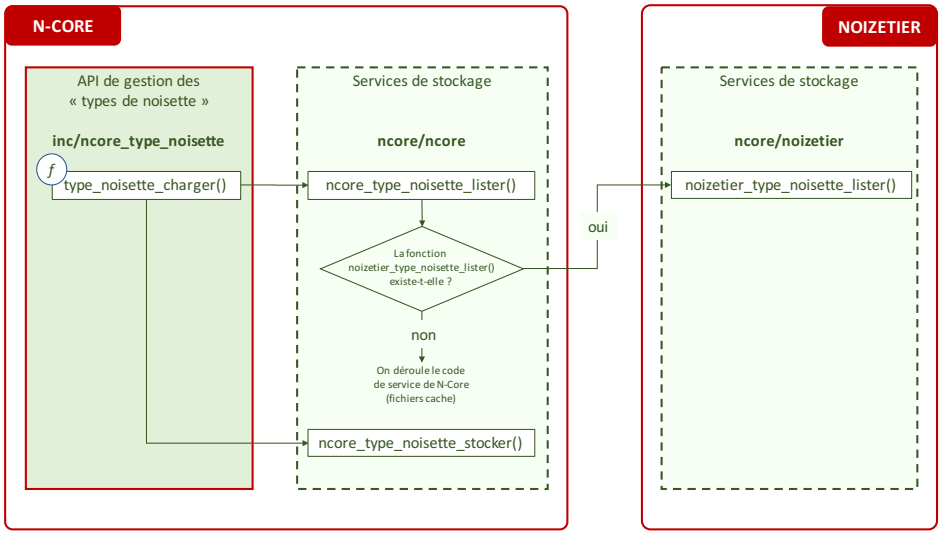
L'argument `$stockage`, lui, permet la réutilisation des services d'un plugin utilisateur par un autre plugin utilisateur ou de forcer l'utilisation des services N-Core (ce qui se fait plus simplement en omettant de créer le service homonyme dans le plugin appelant). Par exemple, le plugin utilisateur « préfixe » pourrait appeler le stockage du noiZetier pour charger ses types de noisette :

```
type_noisette_charger('prefixe', 'types_noisette/',  false, 'noizetier');
```

Il est donc impératif que les plugins utilisateur prévoit toujours cette possibilité dans la conception de leur espace de stockage. Par exemple, pour le noiZetier, une colonne « plugin » a été rajoutée dans la table « spip_noizetier_noisettes » de stockage des types de noisette.

4.3 L'aiguillage des services

Par conception et pour des raisons de lisibilité du code, les fonctions d'API de N-Core appellent systématiquement les fonctions de service de N-Core. Ce sont ces **fonctions de service de N-Core qui réalisent l'aiguillage vers le service souhaité**.



Le schéma ci-dessus illustre de déroulement du code suite à l'appel par le plugin noiZetier :

```
type_noisette_charger('noizetier', 'noisettes/');
```

La fonction `type_noisette_charger()` de l'API fait appel à une fonction de service de N-Core, `ncore_type_noisette_lister()`, qui doit renvoyer la liste des signatures des fichiers YAML des types de noisettes. Cette fonction de service va déterminer quelle fonction appliquer, celle du noiZetier ou elle-même. Pour cela, elle appelle une fonction utilitaire `ncore_chercher_service()` qui lui retourne le nom de la fonction de service ou vide sinon :

```
// Initialisation du tableau de sortie
$types_noisettes = array();

// On cherche le service de stockage à utiliser.
include_spip('inc/ncore_utils');
if ($lister = ncore_chercher_service($plugin, 'type_noisette_lister', $stockage)) {
   // On passe le plugin appelant à la fonction car cela permet ainsi de mutualiser
les services de stockage.
   $types_noisettes = $lister($plugin, $information);
} else {
   // Le plugin ne propose pas de fonction propre ou le stockage N-Core est
explicitement demandé.
   // ...suite du code de la fonction de service N-Core.
```

Le code de la fonction utilitaire `ncore_chercher_service()` est le suivant :

```
function ncore_chercher_service($plugin, $service, $stockage = '') {

   $fonction_trouvee = '';

   // Si le stockage n'est pas précisé on cherche la fonction dans le plugin
appelant.
   if (!$stockage) {
      $stockage = $plugin;
   }

   // Eviter la réentrance si on demande explicitement le stockage N-Core
   if ($stockage != 'ncore') {
      include_spip("ncore/${stockage}");
      $fonction_trouvee = "${stockage}_${service}";
      if (!function_exists($fonction_trouvee)) {
         $fonction_trouvee = '';
      }
   }

   return $fonction_trouvee;
}
```

4.4 Les services

Les fonctions d'API de N-Core font donc appel à des services dont la liste précise est fournie ci-après.

SERVICES

Groupe des services de gestion des types de noisette	
type_noisette_stocker	Stocke les descriptions des types de noisette en distinguant les types de noisette obsolètes, les types de noisettes modifiés et les nouveaux types de noisettes. Chaque description de type de noisette est un tableau associatif dont tous les index possibles - y compris la signature - sont initialisés quel que soit le contenu du fichier YAML.
type_noisette_decrire	Renvoie la description brute d'un type de noisette sans traitement typo des champs textuels ni désérialisation des champs de type tableau sérialisé.
type_noisette_lister	Renvoie, pour l'ensemble des types de noisette utilisés par le plugin appelant, le champ demandé ou la description complète si aucune champ n'est explicitement spécifié. Les données renvoyées sont brutes sous forme d'un tableau indexé par l'identifiant (nom du fichier YAML sans extension) de chaque type de noisette.
Groupe des services de gestion des noisettes	
noisette_stocker	Stocke la description d'une nouvelle noisette ou modifie les paramètres d'affichage d'une noisette existante.
noisette_decrire_grappe	Phrase l'identifiant d'une grappe et en retourne les éléments constitutifs dont le squelette.
noisette_ranger	Modifie le rang d'une noisette existante.
noisette_destocker	Efface de l'espace de stockage la description d'une noisette donnée ou de toutes les noisettes d'une grappe.
noisette_decrire	Renvoie la description brute d'une noisette sans traitement typo des champs textuels. Les champs de type tableau sérialisé sont désérialisés.
noisette_lister	Renvoie, pour l'ensemble des noisettes utilisées par le plugin utilisateur, le champ demandé ou la description complète si aucun champ n'est explicitement spécifié. Les données sont renvoyées brutes sous forme d'un tableau indexé soit par l'identifiant (`id_noisette`) de chaque noisette soit par le couple (`id_grappe`, `rang`).
Groupe des services de compilation des noisettes	

Les services avec une étoile doivent toujours être définies par le plugin utilisateur, les autres étant optionnels. Les services sont toutefois indissociable par « groupe » : si on définit un service comme `type_noisette_stocker()`, il faut alors définir tous les services du même groupe, à savoir, dans ce cas ceux gérant les types de noisette.

N-Core propose l'ensemble de ces services dans son fichier `ncore/ncore.php` associé à ses propres espaces de stockage ce qui permet de minimiser les développements pour la plupart des plugins utilisateur. Le noiZetier v3, par contre, propose l'ensemble des services dans son fichier `ncore/noizetier.php` car il utilise ses propres espaces de stockage.

4.5 La compilation des noisettes

N-Core utilise le pipeline `recuperer_fond` pour intercepter les appels à un squelette `bloc/page.html`. Suivant la configuration d'inclusion des noisettes dans la page concernée, le NoiZetier ajoute à la fin du squelette une inclusion du squelette `noizetier-generer-bloc.html` qui se charge d'inclure les noisettes proprement dites.

Sauf si la configuration de la noisette précise le contraire, le NoiZetier insère les noisettes avec le mode ajax activé.

A COMPLETER

5. DONNEES DE N-CORE

5.1 La structure des données

5.1.1 Les types de noisette

Les données relatives aux types de noisettes proviennent principalement des fichier YAML associés. La description d'un type de noisette est structurée dans N-Core dans un tableau associatif dont tous les champs possibles sont initialisés.

DESCRIPTION D'UN TYPE DE NOISETTE	
Données issues du fichier YAML	
noisette	Identifiant du type de noisette. Correspond au nom du fichier YAML sans extension.
nom	Titre du type de noisette sous forme textuelle ou d'un item de langue. Par défaut coïncide avec l'identifiant du type de noisette.
description	Texte ou item de langue décrivant le rôle du type de noisette.
icon	Nom du fichier d'icône représentant le type de noisette (sans chemin). Par défaut, prend la valeur `noisette-24.png`.

necessite	Liste des plugins – préfixes – nécessairement actifs pour utiliser le type de noisette. Ce champ est un tableau, éventuellement vide, de format « [] = préfixe ».
contexte	Liste des variables de contexte à fournir à la noisette lors de la compilation. Ce champ est un tableau, éventuellement vide, de format « [] = variable ». Les mots-clés « aucun » ou « env » peuvent être utilisés.
ajax	Indicateur d'inclusion en ajax de la noisette lors de la compilation. Prend les valeurs « defaut » (par défaut), « oui », « non ».
inclusion	Indicateur d'inclusion dynamique de la noisette lors de la compilation. Prend les valeurs « statique » (par défaut), « dynamique ».
parametres	Tableau, éventuellement vide, définissant le paramétrage du type de noisette et permettant la génération automatique du formulaire via l'API du plugin Saisies.
Données complémentaires	
plugin	L'identifiant du plugin utilisateur, à savoir, en général, son préfixe.
signature	md5 du fichier YAML calculé lors de son chargement.

Ces données sont initialisées par N-Core qui transmet la description au service de stockage. Il convient au plugin utilisateur de compléter ou pas cette description avant stockage. Ces données ne sont jamais modifiées unitairement mais complètement lors d'un chargement du fichier YAML.

5.1.2 Les noisettes

Les données relatives aux noisettes proviennent du type de noisette, de la localisation de son inclusion dans un squelette et du paramétrage choisi pour la noisette. La description d'une noisette est structurée dans N-Core dans un tableau associatif dont tous les champs possibles sont initialisés.

DESCRIPTION D'UNE NOISETTE	
Données de localisation	
noisette	Identifiant du type de noisette. Correspond au nom du fichier YAML sans extension.
squelette	Chemin relatif du squelette sans son extension (content/article, prive/squelettes/content/accueil, rubrique, etc).
rang	Position de la noisette dans la liste des noisettes incluses dans le même squelette. Ce champ est un entier supérieur ou égal à 1.
Données de paramétrage	
parametres	Tableau, éventuellement vide, définissant les valeurs des paramètres du type de noisette saisis dans le formulaire d'édition de la noisette.

balise	Indicateur d'inclusion de la noisette dans une balise `<div>` englobante. Prend les valeurs « defaut » (par défaut), « oui », « non ».
css	Styles CSS à affecter à la balise `<div>` englobante si celle-ci est requise pour la noisette.
Données complémentaires	
plugin	L'identifiant du plugin utilisateur, à savoir, en général, son préfixe.
id_noisette	Identifiant unique de la noisette retourné lors de la création d'une nouvelle noisette. Ce champ est soit un entier (id d'une table SPIP) soit une chaine auquel cas il est calculé en utilisant la fonction PHP `uniqid()` avec le préfixe `${plugin}_` (cas de N-Core).

Ces données sont initialisées par N-Core qui transmet la description au service de stockage. Il convient au plugin utilisateur de compléter ou pas cette description avant stockage. Seules les données de paramétrage et le rang (en rouge) peuvent être modifiées unitairement après la création globale de la noisette. Les autres données sont, elles, statiques après création.

5.2 Les espaces de stockage

5.2.1 Les types de noisette

N-Core stocke les descriptions des types de noisettes telles que définies au paragraphe 5.1.1 dans un cache sécurisé, installé dans un sous-dossier `ncore/${plugin}/` de `_DIR_CACHE` et nommé `type_noisette_descriptions.php`.

Le cache contient le tableau sérialisé de tous les types de noisette détectés par N-Core ou le plugin utilisateur. La description complète est incluse dans le cache et la clé d'index est l'identifiant du type de noisette (qui est aussi inclus dans la description).

Pour optimiser certains traitements, N-Core utilise un autre cache sécurisé installé dans le même dossier et nommé `type_noisette_signatures.php`. Ce cache contient uniquement le tableau sérialisé des signatures des fichiers YAML indexé par l'identifiant du type de noisette. Cette information est aussi présente dans le cache des descriptions.

Ces deux caches sont créés ou mis à jour simultanément lors de l'appel à la fonction d'API `type_noisette_charger()`.

5.2.2 Les noisettes

N-Core stocke les affectations de noisettes telles de définies au paragraphe 5.1.2 dans une meta nommée `${plugin}_noisettes`.

Cette meta contient le tableau sérialisé de toutes les noisettes affectées à divers squelettes utilisés par le plugin utilisateur. Chaque affectation de noisette est un tableau indexé par squelette, puis pour un squelette par rang. L'identification d'une noisette par le couple (squelette, rang) est donc plus adaptée pour le stockage N-Core.

5.2.3 Les éléments de contexte des noisettes

6. CONCEPTION ET UTILISATION DES API N-CORE

6.1 Principes généraux

6.1.1 Nommage des fonctions

Le nommage des fonctions appartenant aux différentes API de N-Core 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 de la description d'un type de noisette se nomme `type_noisette_lire()` et la fonction d'ajout d'une noisette se nomme `noisette_ajouter()`.

En outre, la même action se traduite par le même verbe à l'infinitif quel que soit l'objet concerné. Par exemple, la fonction de lecture de la description d'une noisette se nomme `noisette_lire()`.

6.1.2 Arguments standardisés

Toutes les fonctions des API N-Core possèdent à minima deux arguments récurrents , à savoir, `$plugin` et `$stockage`.

L'argument **obligatoire** `$plugin` est toujours le **premier** argument du prototype des fonctions d'API. C'est une chaine de caractères qui **identifie le module utilisant la fonction** qui est dans tous les cas ou presque, un plugin à l'instar du noiZetier. Pour un plugin, l'utilisation du préfixe est recommandée. Cet argument est principalement utilisé pour distinguer les espaces de stockage d'un plugin utilisateur par rapport à d'autres.

L'argument **facultatif** `$stockage` est toujours le **dernier** argument du prototype des fonctions d'API. C'est une chaine de caractères qui est initialisée à vide si l'argument n'est pas fourni et qui identifie le type de stockage à utiliser en priorité indépendamment du plugin appelant `$plugin`. C'est normalement le préfixe du plugin fournissant le stockage souhaité. Cet argument permet la réutilisation des services d'un plugin utilisateur par un autre plugin utilisateur.

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

Par exemple, l'argument facultatif `$information` désigne toujours un champ de la description d'une noisette ou d'un type de noisette.

L'argument `$type_noisette` désigne toujours l'identifiant d'un type de noisette qui coïncide avec le nom du fichier YAML sans extension.

De même, l'argument `$noisette` désigne toujours l'identifiant d'une noisette et peut revêtir deux formes : celle d'un *id* unique (entier ou chaine) ou celle d'un couple *(squelette, rang)*. Cette souplesse

permet d'optimiser l'adressage de la noisette en fonction du format de stockage. La conséquence est qu'il sera demandé aux fonction de services de supporter les deux adressages.

6.2 L'API de gestion des types de noisette

6.2.1 Conception

6.2.2 Utilisation

6.3 L'API de gestion des noisettes

6.3.1 Conception

Les fonctions de l'API de gestion des noisettes sont orientées de façon à faciliter la mise en œuvre des actions des utilisateurs, à savoir :

- l'ajout d'une noisette à un squelette,
- la suppression d'une noisette,
- le déplacement d'une noisette d'un emplacement à un autre au sein du même squelette,
- la modification des paramètres d'affichage d'une noisette,
- et la lecture des données d'une noisette, souvent pour son affichage.

6.3.2 Utilisation