Référence SPIP/N/2017.002 – Ed. 1.11

GUIDE
CONCEPTION DU PLUGIN N-CORE

FICHE D'IDENTIFICATION	
Rédacteur	Eric Lupinacci
Projet	SPIP
Étude	Conception du plugin N-Core
Nature du document	Guide
Date	22/02/2021
Nom du fichier	Guide N - Le plugin N-Core.docx
Référence	SPIP/N/2017.002 – Ed. 1.11
Dernière mise à jour	22/02/2021 18:34:04
Langue du document	Français
Nombre de pages	41

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 (version 0.17.4 et ultérieures) dont l'objectif est de fournir des API génériques de gestion des noisettes et de leur compilation.

N-Core ne fournit aucune interface utilisateur de gestion des noisettes si ce n'est le formulaire d'édition d'une noisette. Le noiZetier, dans sa version 3, utilise N-Core et offre une interface d'administration complète 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 ou d'autres objets.

2. CONCEPTS

2.1 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`).

2.2 Les types de noisettes

2.2.1 Définition

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). L'identifiant a la forme `[<type_page>-][<composition>-]nom_type_noisette`.

Un type de noisette est donc décrit par deux fichiers obligatoires :

- `type_noisette.yaml` pour la configuration,
- et `type_noisette.html` pour l'affichage,

et un fichier optionnel `type_noisette-preview.html` pour la prévisualisation, utilisée par exemple par le noiZetier pour visualiser la liste des noisettes dans le privé. Cet affichage est destiné à fournir

uniquement des informations de base du paramétrage de la noisette, les autres informations étant à exclure (comme le type et l'icône).

2.2.2 Catégories

Les types de noisette sont classifiés par catégorie, la liste des catégories n'étant pas fixée à l'avance. Les catégories sont affectées au travers du YAML du type de noisette (bloc « categories: »).

Une catégorie est définie par son identifiant , un libellé, une description et un icône. Il est possible d'affecter une ou plusieurs catégories à un type de noisette. Si aucune catégorie n'est définie dans le YAML, N-Core affecte d'autorité la catégorie d'identifiant réservé « defaut » au type de noisette.

2.3 Les noisettes

Une noisette est une **« instance paramétrée d'un type de noisette » incluse dans un conteneur** donné.

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.

Il est possible d'inclure plusieurs noisettes à la suite dans un conteneur. Il est donc nécessaire de gérer un rang pour chaque noisette incluse dans un conteneur.

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 de façon unique une noisette avec le conteneur auquel elle est associée et son rang. L'utilisation optimale de l'un ou l'autre identifiant dépend de la structure de stockage adoptée.

2.4 Les conteneurs

2.4.1 Définition

Dans les versions 1 et 2 du noiZetier, les noisettes sont associées à un squelette. N-Core, lui étend cette notion en permettant d'associer dans un ordre précis des noisettes à un **conteneur** qui peut être tout autre chose qu'un squelette.

Par exemple, le plugin noiZetier permet d'insérer des noisettes dans un bloc de page et ce pour un contenu précis (i.e. l'article 12). Ce faisant, il met en relation des noisettes, un squelette comme `content/article.html` et l'article concerné. Le couple (squelette, article) est en fait un conteneur.

Le conteneur peut aussi servir à choisir une liste de noisettes pour chaque utilisateur affichant une page ainsi composée. Dans ce cas, le conteneur est l'auteur concerné (objet auteur et identifiant `id_auteur`).

2.4.2 Noisette conteneur

Il est possible d'utiliser une noisette comme conteneur afin, par exemple, d'imbriquer du contenu dans une page. Néanmoins, une noisette ne peut être un conteneur que si son type possède la

propriété `conteneur` à 'oui' inscrite dans son YAML. Il est possible d'imbriquer plusieurs niveaux de noisette conteneur, N-Core n'imposant aucune limite.

Le fichier HTML d'une noisette conteneur inclut toujours une marque `<!--noisettes-->` pour indiquer où insérer le contenu compilé des noisettes incluses dans ce conteneur.

N-Core propose un type de noisette conteneur dont l'identifiant est `conteneur`. Ce type de noisette est suffisamment générique pour la plupart des usages car il permet de choisir la balise englobante et les styles à y attacher.

2.4.3 Structure de données

Un conteneur est matérialisé par un **tableau associatif**. Les index sont libres mais par convention l'index `squelette` désigne toujours un squelette comme conteneur ou comme élément d'un conteneur.

Une noisette conteneur est toujours identifiée par un tableau associatif formé par le couple (type_noisette, id_noisette) :

- l'index `type_noisette` permet de désigner le type de noisette d'une noisette conteneur et donc d'en déduire le squelette associé en utilisant le dossier configuré pour les types de noisettes ;
- l'index `id_noisette` identifie de façon unique la noisette conteneur.

Un conteneur possède aussi un identifiant unique de type chaine. Le calcul de cet identifiant doit être réversible et permettre de retrouver le tableau associatif du conteneur à partir de l'identifiant. N-Core calcule lui-même l'identifiant des noisettes conteneurs `conteneur|noisette|13`. Par contre, pour les autres conteneurs qui sont spécifiques au plugin utilisateur le calcul doit être effectué par le plugin lui-même (par exemple, `content/sommaire` ou `content/article|article|12` pour le noiZetier).

2.5 Les capsules

Une noisette peut être encapsulée systématiquement ou par configuration dans un markup HTML fourni par un fichier appelé une **capsule**. Les noisettes conteneur ne peuvent pas être encapsulée car elle agissent déjà comme une capsule.

Les capsules sont stockées dans le dossier `capsules/`. Une capsule est associée soit à un type de noisette auquel cas elle porte le nom du type de noisette, soit est générique pour toute noisette et porte le nom de `dist`.

Le fichier HTML d'une capsule inclut toujours une marque `<!--noisettes-->` pour indiquer où insérer le contenu compilé de la noisette à encapsuler.

N-Core ne fournit aucune capsule par défaut, c'est au plugin utilisateur de définir, si besoin, ses propres capsules y compris la capsule générique `dist`, si besoin. Par contre, N-Core est capable, si demandé, d'encapsuler une noisette dans une balise `<div>` même si aucune capsule n'est disponible pour la noisette. Le mécanisme d'encapsulation est décrit au paragraphe 4.6.2.

3. PERIMETRE DE N-CORE

N-Core propose plusieurs 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 étendu » avec un stockage dédié ;
- La gestion des conteneurs ;
- La compilation des noisettes, à savoir, la gestion du contexte et des paramètres de chaque noisette et leur affichage.

3.1 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 noisette sont recherchés dans un répertoire relatif fourni par la fonction de service `ncore_type_noisette_initialiser_dossier()`. Le type de noisette `conteneur` fournie par N-Core est systématiquement chargée. La fonction optimise le chargement en effectuant uniquement les traitements nécessaires en fonction des modifications, ajouts et suppressions des types de noisette 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 sérialisés en base sont fournis desérialisés. 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. Les données sont renvoyées brutes sous forme d'un tableau indexé par l'identifiant de chaque type de noisette.
type_noisette_decacher	Supprime tout ou partie des caches utilisés pour la compilation des noisettes.
type_noisette_repertorier_categories	Renvoie la description d'une ou de toutes les catégories de type de noisette.

3.2 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 conteneur. L'interface utilisateur permettant le choix du conteneur et des types de noisette ne fait pas partie de N-Core.

API NOISETTES : INC/NCORE_NOISETTE.PHP	
noisette_ajouter	Ajoute à un conteneur, à un rang donné ou en dernier rang, une noisette d'un type donné et met à jour le rang des autres noisettes du conteneur si nécessaire. Si le rang n'est pas précisé, la noisette est ajoutée en dernier rang. Si tout s'est bien passé la fonction renvoie l'identifiant unique de la noisette.
noisette_deplacer	Déplace une noisette de sa position au sein d'un conteneur à une nouvelle position au sein du même conteneur ou d'un autre conteneur. La fonction met aussi à jour le rang des noisettes du conteneur d'origine si nécessaire.
noisette_dupliquer	Duplique une noisette dans un conteneur destination. Si la noisette dupliquée est un conteneur toutes les noisettes incluses sont aussi dupliquées et ce, récursivement.
noisette_lire	Retourne, pour une noisette, la description complète ou seulement un champ précis. Les champs sérialisés en base sont fournis désérialisés. Les champs textuels peuvent subir un traitement typo si demandé.
noisette_parametrer	Met à jour les paramètres de configuration de la noisette destinés à son affichage. La fonction contrôle toujours que seuls les champs éditables sont modifiés (voir paragraphe 5.1.2).
noisette_supprimer	Supprime une noisette donnée et met à jour les rangs des autres noisettes du conteneur si nécessaire. Si la noisette est un conteneur les noisettes incluses sont aussi supprimées et ce récursivement.
API NOISETTES : NCORE_FONCTIONS.PHP	
noisette_repertorier	Renvoie une liste de noisettes, appartenant ou pas à un conteneur et éventuellement filtrée sur certains champs. Les données sont renvoyées brutes sous forme d'un tableau indexé soit par l'identifiant de chaque noisette, soit par le rang si le conteneur est précisé, soit par le couple (identifiant de conteneur, rang) si aucun conteneur n'est précisé.

N-Core propose aussi une balise utilisable dans l'espace public `#NOISETTE_REPERTORIER` qui fournit, dans un tableau indexé par rang ou par id de noisette, la description de toutes les noisettes incluses dans un conteneur donné. Cette balise est une encapsulation de l'API `noisette_repertorier()` mais limitée à un conteneur et sans les options de filtrage ni d'index (toujours le rang).

Le prototype de la balise est : `#NOISETTE_REPERTORIER{plugin, conteneur[, stockage]}`.

3.3 L'API de gestion des conteneurs

L'API de gestion des conteneurs fournit une interface fonctionnelle aujourd'hui limitée au vidage des noisettes d'un conteneur et au calcul de son identifiant unique.

API CONTENEURS : INC/NCORE_CONTENEUR.PHP	
conteneur_construire	Renvoie, pour un conteneur donné, son tableau exact calculé à partir de son identifiant unique au format chaine. Cette fonction est une encapsulation exacte de la fonction de service `ncore_conteneur_construire()`.
conteneur_identifier	Renvoie, pour un conteneur donné, son identifiant calculé à partir de ses éléments. L'identifiant est une chaine de caractères unique non vide. Cette fonction est une encapsulation de la fonction de service `ncore_conteneur_identifier()`. A ceci près qu'elle vérifier le tableau du conteneur en appelant le service `ncore_conteneur_verifier()` au préalable.
conteneur_est_noisette	Détermine si un conteneur, connu par son identifiant sous sa forme tabulaire ou chaine, est une noisette ou pas.
conteneur_vider	Supprime toutes les noisettes incluses dans un conteneur et ce de façon récursive si des noisettes conteneur sont imbriquées.

N-Core propose aussi une balise utilisable dans l'espace public **`#CONTENEUR_IDENTIFIER`** qui fournit l'identifiant textuel du conteneur à partir de sa description tabulaire. Cette balise est une encapsulation exacte de l'API `conteneur_identifier()`.

Le prototype de la balise est : `#CONTENEUR_IDENTIFIER{plugin, conteneur[, stockage]}`.

3.4 L'API de compilation des noisettes

3.4.1 L'affichage public d'une noisette

La véritable API de compilation des noisettes est la balise **`#NOISETTE_COMPILER`** qui construit l'affichage de la noisette concernée.

La **balise devant être appelée dans une boucle de noisettes**, la plupart des données de la noisette en cours de compilation sont accessibles via la fonction `champ_sql()`. Néanmoins, elle requiert un argument obligatoire, l'identifiant de la noisette, et un argument optionnel, le stockage spécifique si celui-ci diffère de celui du plugin appelant.

Le prototype de la balise est donc : `#NOISETTE_COMPILER{id_noisette[, stockage]}`.

La balise gère la **récursivité nécessaire à la compilation des noisettes conteneur** et l'appel à l'encapsulation des noisettes.

Pour fonctionner, la balise `#NOISETTE_COMPILER` utilise des filtres nécessaires à la détermination du contexte de la noisette, à son comportement Ajax , à son inclusion dynamique ou pas, à son encapsulation et à la localisation du type de noisette.

API COMPILATION : NCORE_FONCTIONS.PHP	
noisette_contextualiser	Construit le contexte d'une noisette donnée à partir de la configuration de son type, de l'environnement et de ses identifiants – id_noisette et couple (conteneur, rang). La configuration du contexte des types de noisette est stockée dans un cache dédié qui est géré par la fonction.
noisette_encapsuler	Inclut le contenu compilé d'une noisette non conteneur dans une capsule ou insère la compilation des noisettes incluses dans une noisette conteneur dans le HTML dudit conteneur (la noisette conteneur se comporte comme une capsule).
type_noisette_ajaxifier	Détermine si une noisette doit être incluse en ajax ou pas en fonction de la configuration de son type de noisette et de la configuration générale du plugin. L'indicateur ajax de chaque type de noisette est stocké dans un cache dédié géré par la fonction.
type_noisette_dynamiser	Détermine si une noisette doit être incluse dynamiquement ou pas en fonction de la configuration de son type de noisette et de la configuration générale du plugin. L'indicateur d'inclusion dynamique de chaque type de noisette est stocké dans un cache dédié géré par la fonction.
type_noisette_localiser	Renvoie le dossier relatif des types de noisette pour le plugin appelant ou la localisation relative du type de noisette demandé. Cette fonction gère le cas particulier de la noisette conteneur fournie par N-Core qui est, elle, toujours dans le dossier par défaut de N-Core.

Ces filtres sont présentés comme une API car ils peuvent éventuellement servir à un plugin utilisateur qui souhaiterait surcharger la balise `#NOISETTE_COMPILER`.

3.4.2 La prévisualisation d'une noisette

N-Core propose un autre affichage d'une noisette disponible uniquement si le type de noisette possède un fichier de prévisualisation (`type_noisette-preview.html`). Le but de cette prévisualisation est de fournir une vue synthétique des paramètres de la noisette utilisée dans les interfaces de configuration des noisettes.

Comme pour l'affichage classique, N-Core propose une balise **`#NOISETTE_PREVIEW`** qui construit la prévisualisation de la noisette concernée.

En outre, comme un type de noisette peut être chargé mais rester inactif du fait qu'au moins un des plugins qu'il nécessite est désactivé, la balise gère aussi ce cas en affichant un message d'erreur à la place de la prévisualisation.

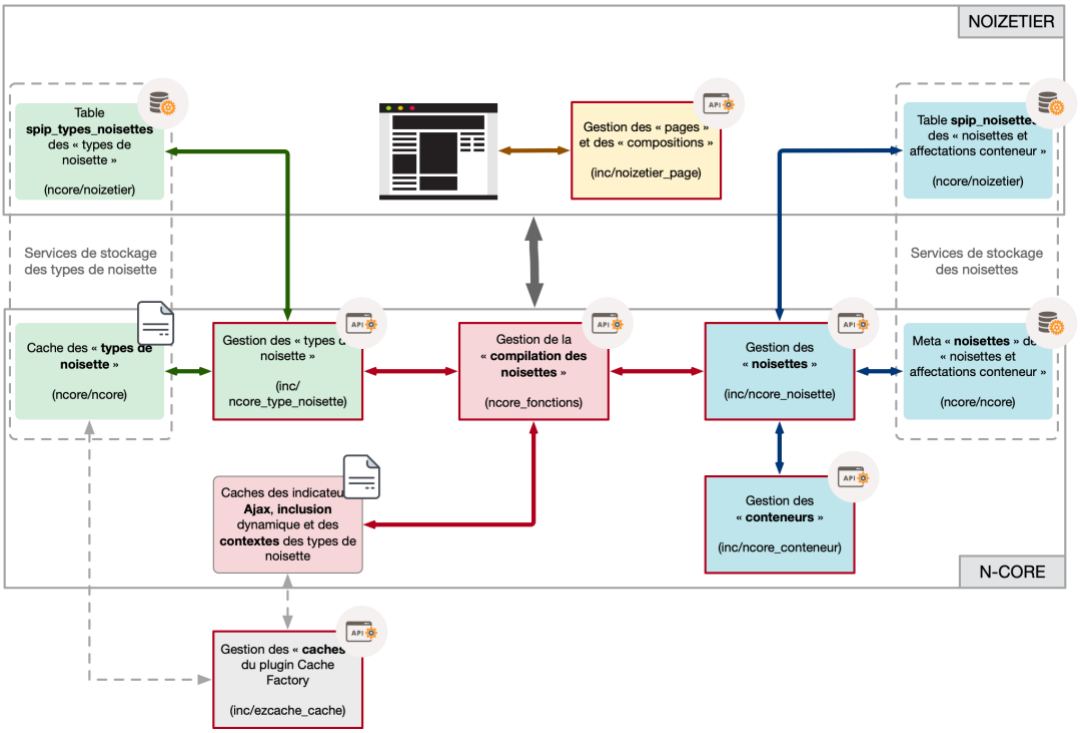
Le prototype de la balise est :

```
#NOISETTE_PREVIEW{id_noisette, type_noisette_actif, plugins_necessites}.
```

4. FONCTIONNEMENT DE N-CORE

4.1 Schéma de principe

Le fonctionnement global du plugin N-Core est illustré ci-dessous en regard de l'utilisation qu'en fait 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, conteneur et compilation). Néanmoins, un plugin utilisateur pourrait se passer de certaines API 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 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 service 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 et utiliser le stockage N-Core.

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 dédiée

à cet usage. Cette fonctionnalité impose bien entendu des contraintes aux fonctions d'API et 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, $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`.

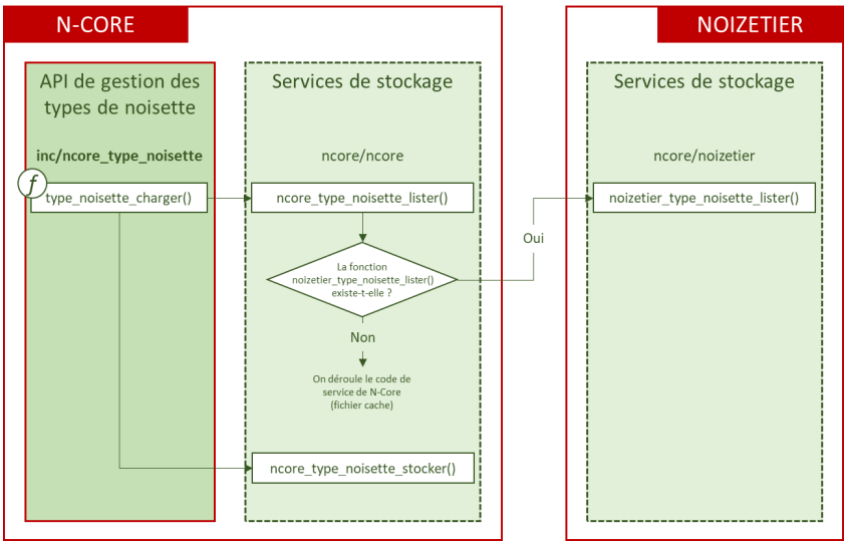
L'argument `$stockage`, lui, permet la réutilisation des services d'un plugin utilisateur par un autre plugin utilisateur ou de forcer l'utilisation des services N-Core (ce qui se fait plus simplement en omettant de créer le service homonyme dans le plugin appelant). Par exemple, le plugin utilisateur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

```
type_noisette_charger('prefixe', 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 les tables de stockage des types de noisette et des noisettes.

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** ce qui leur permet aussi d'effectuer des traitements génériques comme la récursivité sur la suppression des noisettes conteneur imbriquées et donc de limiter la complexité pour les plugins utilisateur.



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

```
type_noisette_charger('noizetier');
```

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 si aucune fonction n'est définie dans le plugin appelant :

```
// 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 pipelines

Certains services ne se comportent pas comme ceux décrits ci-dessus. En effet, le service de N-Core ne fait pas appel à un service du plugin utilisateur via la fonction `ncore_chercher_service()` mais appelle un pipeline homonyme. Cela permet de prolonger la chaine d'appel au-delà du plugin utilisateur ce qui est parfois nécessaire.

Les pipelines concernés sont `type_noisette_completer_description`, `type_noisette_lister_categories`, `type_noisette_styler`, `noisette_completer_description` et `noisette_completer_action`.

Par exemple, N-Core propose un service permettant de compléter la description d'une noisette avant son ajout : `ncore_noisette_completer_description()`. Ce service fait appel au pipeline homonyme `noisette_completer_description` de la façon suivante :

```
function ncore_noisette_completer_description($plugin, $description, $action, $stockage='') {
  // Si le plugin utilisateur veut compléter la description il utilise le pipeline homonyme.
  $flux = array(
    'args' => array(
      'plugin'   => $plugin,
      'action'   => $action,
      'stockage' => $stockage
    ),
    'data' => $description
  );
  $description = pipeline('noisette_completer_description', $flux);

  return $description;
}
```

Il est conseillé d'inclure ces fonctions dans le même fichier que les autres services.

4.5 Récapitulatif des services

Les fonctions d'API de N-Core font donc appel à des services dont la liste exacte est fournie ci-après. Le nom des fonctions est amputé du préfixe du plugin appelant.

Ces fonctions de service ne doivent pas être appelées par le plugin appelant qui doit utiliser exclusivement les fonctions d'API. Le plugin appelant peut définir ses propres services qui seront appelés par ceux de N-Core, mais en aucun cas les utiliser dans son code.

SERVICES	
Groupe des services de gestion des conteneurs	
conteneur_identifier (o)	Renvoie, pour un conteneur donné (tableau associatif), son identifiant calculé à partir de ses index. L'identifiant est une chaine de caractères unique non vide.
conteneur_construire (o)	Construit le conteneur sous forme de tableau associatif à partir de son identifiant unique (service inverse de `conteneur_identifier`.
conteneur_est_noisette	Détermine si le conteneur est une noisette et renvoie true ou false. **Ce service est le seul à n'être jamais surchargé par un plugin utilisateur**.
conteneur_destocker	Retire, de l'espace de stockage, toutes les noisettes d'un conteneur et ce de façon récursive si des noisettes conteneur sont incluses.
conteneur_verifier (o)	Vérifie la conformité des index du tableau associatif représentant le conteneur et supprime les index inutiles, si besoin.

Groupe des services de gestion des types de noisette	
type_noisette_completer_description[p]	Complète si nécessaire la description d'un type de noisette issue de la lecture de son fichier YAML avant son stockage avec des champs spécifiques au plugin utilisateur. Ce service facultatif appelle le pipeline `type_noisette_completer_description`.
type_noisette_decrire	Renvoie la description brute d'un type de noisette (sauf le timestamp `maj`) sans traitement typo des champs textuels ni désérialisation des champs de type tableau sérialisé.
type_noisette_initialiser_dossier	Renvoie la configuration par défaut du dossier relatif du PATH SPIP accueillant les types de noisette à charger. N-Core initialise cette valeur à 'noisettes/'.
type_noisette_initialiser_ajax	Renvoie la configuration par défaut de l'ajax à appliquer pour la compilation des noisettes. Cette information est utilisée si la description YAML d'un type noisette ne contient pas de tag ajax ou contient un tag ajax à 'defaut'.
type_noisette_initialiser_inclusion	Renvoie la configuration par défaut de l'inclusion à appliquer pour la compilation des noisettes (statique ou dynamiauqe). Cette information est utilisée si la description YAML d'un type noisette ne contient pas de tag inclusion ou contient un tag inclusion à 'defaut'.
type_noisette_lister_categories[p]	Renvoie la liste des catégories et leur description. Ce service facultatif appelle le pipeline `type_noisette_lister_categories`.
type_noisette_lister	Renvoie, pour l'ensemble des types de noisette utilisés par le plugin appelant, le champ demandé ou la description complète (sauf le timestamp `maj`) si aucun 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.
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_styler[p]	Renvoie la liste des saisies permettant de compléter le champ 'css' des styles affectés à une capsule ou à

	une noisette conteneur. Par défaut, N-Core fournit une saisie unique de type input texte. Ce service appelle le pipeline `type_noisette_styler`.
type_noisette_traiter_typo	Traite avec la fonction `typo()`, si ils existent, les champs textuels de la description d'un type de noisette spécifiques au plugin appelant. Ce service est facultatif et N-Core traite toujours les champs nom et description.
Groupe des services de gestion des noisettes	
noisette_changer_conteneur	Transfère une noisette d'un conteneur vers un autre à un rang donné. Le rang destination n'est pas vérifié lors du rangement dans le conteneur destination, il convient à l'appelant de vérifier que le rang est libre. La profondeur dans le conteneur destination est fournie en argument et doit être mise à jour par l'appelant.
noisette_completer_action[p]	Complète si nécessaire le traitement effectué sur une noisette (ajout, paramétrage, suppression, déplacement ou duplication). Ce service facultatif appelle le pipeline `noisette_completer_action`.
noisette_completer_description[p]	Complète si nécessaire la description d'une noisette avec des champs spécifiques au plugin utilisateur. Ce service facultatif appelle le pipeline `noisette_completer_description`.
noisette_decrire	Renvoie la description brute d'une noisette sans traitement typo des champs textuels ni désérialisation des champs de type tableau sérialisé.
noisette_destocker	Efface de l'espace de stockage la description d'une noisette donnée.
noisette_initialiser_encapsulation	Renvoie la configuration par défaut de la capsule à utiliser dans le cas où le champ balise de la noisette vaut 'defaut'. Les valeurs sont 'oui' ou 'non'. N-Core initialise l'encapsulation à 'oui'.
noisette_initialiser_parametrage[p]	Renvoie la liste des champs éditables d'une noisette en distinguant le cas d'une noisette conteneur ou non. Ce service facultatif appelle le pipeline `noisette_initialiser_parametrage`.

noisette_lister	Renvoie, pour un conteneur ou 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 de chaque noisette soit par le couple (identifiant de conteneur, rang).
noisette_ranger	Positionne une noisette à un rang différent de celui qu'elle occupe dans le conteneur.
noisette_stocker	Stocke la description d'une nouvelle noisette ou modifie les paramètres d'affichage d'une noisette existante.
noisette_traiter_typo	Traite avec la fonction `typo()`, si ils existent, les champs textuels de la description d'une noisette spécifiques au plugin appelant. Ce service est facultatif, N-Core n'ayant aucun champ à traiter.

Les services notés [p] sont des services-pipeline.
Les services notés (o) doivent toujours être définies par le plugin utilisateur, les autres sont optionnels. Les services sont toutefois indissociables par « groupe » : si on définit un service comme `type_noisette_stocker()`, il faut alors définir tous les services non facultatifs du même groupe, à savoir, dans ce cas ceux gérant les types de noisette à l'exception de `type_noisette_completer()` et `type_noisette_traiter_typo()` si rien n'est à rajouter.

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. Pour les noisettes conteneur, N-Core fournit le code des services `conteneur_identifier()`, `conteneur_verifier()` et `conteneur_construire()` et ne fait jamais appel au plugin utilisateur. Par contre, N-Core considère que la gestion des identifiants tableau ou chaine des autres conteneurs est toujours un sujet propre au plugin appelant.

Le noiZetier v3, propose, lui, l'ensemble des services dans son fichier `ncore/noizetier.php` car il utilise ses propres espaces de stockage.

4.6 La compilation des noisettes

4.6.1 Le calcul du contenu de la noisette

N-Core utilise une balise nommée `#NOISETTE_COMPILER` pour calculer le contenu d'une noisette donnée. La balise fait appel à la fonction `recuperer_fond()` en tenant compte de plusieurs paramètres :

- La **localisation du type de noisette** en gérant le cas particulier du type de noisette `conteneur` fourni par N-Core qui sera toujours localisée dans le dossier relatif de N-Core. Il est conseillé de ne pas la surcharger sauf besoin impérieux ;

- Le **contexte de la noisette** qui est une combinaison de l'environnement de la page dans laquelle est incluse la noisette, de la configuration du contexte du type de noisette concerné et des variables spécifiques liées à la noisette comme son identifiant. A minima, les deux identifiants de la noisette (id_noisette et le couple id_conteneur, rang) font toujours partie du contexte fourni ;
- l'**indicateur d'inclusion dynamique** qui est une combinaison du paramétrage défini au niveau de chaque type de noisette et du paramétrage global de l'inclusion pour le plugin appelant ;
- l'**indicateur de comportement Ajax** qui est une combinaison du paramétrage défini au niveau de chaque type de noisette et du paramétrage global de l'Ajax pour le plugin appelant.

Chacun de ces paramètres est géré par une fonction d'API décrites au paragraphe 3.4.1. A l'exception de la fonction de localisation du type de noisette, chacune de ces fonctions gère un cache propre pour accélérer les traitements de compilation. Ces caches sont décrits au paragraphe 5.2.

4.6.2 L'encapsulation d'une noisette non conteneur

Par défaut, le contenu compilé de chaque noisette non conteneur est systématiquement inclus dans une balise `<div>` ou dans un squelette plus sophistiqué lié au type de noisette. Cette opération s'appelle **l'encapsulation** de la noisette et le squelette englobant une **capsule**. Cette encapsulation est réalisée par un filtre appelé dans le code de la balise `#NOISETTE_COMPILER`.

Pour conserver un fonctionnement historique, il est possible de débrayer l'encapsulation automatique et de laisser l'utilisateur choisir pour chaque noisette si une encapsulation doit être appliquer ou pas. Pour passer dans ce mode manuel, il suffit de positionner la constante `_NCORE_ENCAPSULATION_AUTO` à `false`. Dans ce cas, c'est le champ « encapsulation » de la structure de données d'une noisette qui le définit. Les valeurs possibles sont alors :

- « oui », qui indique d'appliquer une encapsulation. Dans ce cas, il est possible d'agrémenter le HTML d'encapsulation par des classes CSS additionnelles (champ « css » de la noisette) ;
- « non », qui indique qu'aucune encapsulation n'est à réaliser ;
- « defaut », qui indique que l'encapsulation dépend de la valeur par défaut retournée par la fonction de service `noisette_initialiser_encapsulation()`. Pour N-Core la fonction de service renvoie « oui ».

Quand une encapsulation est requise pour une noisette, la fonction N-Core `noisette_encapsuler()` applique le protocole suivant :

1. elle cherche d'abord si une capsule propre au type de noisette concernée existe et si c'est le cas l'applique au contenu de la noisette ;
2. sinon, elle cherche la capsule générique `dist` et l'applique si elle existe ;
3. et sinon, si aucune capsule n'existe, elle encapsule le contenu de la noisette dans une balise `<div>` sans passer par une capsule (pseudo-capsule non matérialisée par un HTML).

4.6.3 La compilation d'une noisette conteneur

La balise `#NOISETTE_COMPILER` assure aussi la compilation des noisettes conteneur qui requiert un traitement spécifique tant pour la compilation que pour l'encapsulation. Quand une noisette conteneur est compilée, le schéma suivant est appliqué :

- Plutôt que de compiler la noisette conteneur elle-même en appelant `recuperer_fond()` sur le fichier HTML `type_noisette.html`, la balise compile la liste des noisettes incluses en appelant `recuperer_fond()` sur le squelette `conteneur_compiler.html` en fournissant l'environnement aux noisettes incluses.
- Ensuite, la balise encapsule le HTML précédemment calculé en utilisant pour capsule la noisette conteneur elle-même, sachant qu'un conteneur n'est jamais encapsulé mais agit comme une capsule.

Le squelette `conteneur_compiler.html` dont le code est fourni ci-dessous est fourni par N-Core mais peut-être surchargé par un plugin utilisateur pour mieux s'adapter à l'espace de stockage utilisé.

```
#SET{noisettes, #NOISETTE_REPERTORIER{#ENV{plugin}, #ENV{id_conteneur}, #ENV{stockage}}}
<BOUCLE_noisettes_conteneur(DATA) {source table, #ENV{noisettes, #ARRAY}}{plugin}{par rang_noisette}>
  [(#NOISETTE_COMPILER{#VALEUR{id_noisette}, #ENV{stockage, ''}})]
</BOUCLE_noisettes_conteneur>
```

4.6.4 L'affichage d'une liste de noisettes

Étant donné que la balise `#NOISETTE_COMPILER` prend en charge la compilation des noisettes conteneur (éventuellement imbriquées) et l'encapsulation, l'affichage d'une liste de noisettes se réduit à une boucle triviale ne faisant appel qu'à la balise `#NOISETTE_COMPILER` pour chaque noisette de la liste.
En outre, la liste à afficher correspond toujours aux noisettes incluses dans un conteneur spécifique, ce qui fait que le squelette `conteneur_compiler.html` peut aussi être utiliser.

Par exemple, le noiZetier affiche une page en appelant la compilation de chaque bloc Z constituant la page. Pour ce faire, il fait appel pour chaque bloc (qui est un conteneur spécifique) au squelette `conteneur_compiler.html` qu'il surcharge pour tenir compte de son propre espace de stockage des noisettes comme illustré ci-dessous.

```
<BOUCLE_noisettes_conteneur(NOISETTES) {plugin=noizetier}{id_conteneur=#ENV{id_conteneur}}{par rang_noisette}>
  #NOISETTE_COMPILER{#ID_NOISETTE}
</BOUCLE_noisettes_conteneur>
```

4.7 La gestion des caches

La gestion des caches est à usage interne N-Core car celui-ci utilise de nombreux caches sécurisés comme moyen de stockage. Pour gérer ses caches, N-Core utilise le plugin Cache Factory qui offre une API et un fonctionnement en services comparable à celle de N-Core.

Pour une description complète de l'API du plugin Cache Factory il est conseillé de lire le guide de conception associé.

4.7.1 La configuration des caches

L'utilisation du plugin Cache Factory requiert en premier lieu de définir la configuration standard des caches de N-Core, à savoir, leur localisation, leur nom et leurs attributs comme la sérialisation des

donnés incluses et la sécurisation du cache. Les caches de N-Core sont tous du même type appelé « stockage ». La configuration contient les données suivantes :

CONFIGURATION DES CACHES N-CORE		
Localisation		
racine	`'_DIR_CACHE'`	Nom de la constante et pas sa valeur (noter les guillemets autour de la constante)
sous_dossier	`true`	Le sous-dossier portera le nom du plugin utilisateur de N-Core ('noizetier' par exemple)
Nommage		
nom_obligatoire	`array('objet', 'fonction')`	Le composant objet pour 'type_noisette' et 'fonction' pour 'ajax', 'inclusions'…
nom_facultatif	`array()`	Pas de composant facultatif
extension	`'.php'`	Obligatoire pour un fichier sécurisé
separateur	`'-'`	Tiret pour éviter la confusion avec le souligné de type_noisette.
Attributs		
securisation	`true`	Les caches N-Core sont tous sécurisés
serialisation	`true`	Les données des caches sont des tableaux sérialisés
conservation	`0`	Les caches n'ont pas de durée de péremption. Ils peuvent être supprimés manuellement (vidage du cache SPIP).

De façon plus prosaïque, les caches de N-Core sont stockés dans un dossier `tmp/cache/ncore/<plugin_utilisateur_ncore>/` et se nomment `<objet>-<fonction>.php`. Cela donne pour le cache Ajax du noiZetier : `tmp/cache/ncore/noizetier/type_noisette-ajax.php`.

4.7.2 Les services Cache Factory propres à N-Core

Etant donné les besoins limités de N-Core concernant les caches, seul le service de configuration `cache_configurer()` est surchargé, ce qui revient pour N-Core à coder la fonction `ncore_cache_configurer()` dans le fichier `cache/ncore.php`. Le service se limite à remplir et renvoyer un tableau PHP des données précitées.

4.7.3 L'utilisation de l'API Cache Factory

N-Core se contente d'utiliser les API `cache_lire()` et `cache_ecrire()` : la fonction `type_noisette_ajaxifier()` donne un exemple d'utilisation. Pour les besoins des plugins utilisateurs et comme les caches sont tous associés à l'objet `type_noisette`, N-Core fournit une fonction de vidage des caches de compilation, `type_noisette_decacher()`. Cette fonction est utile pour forcer le calcul des caches au moment propice défini par le plugin utilisateur.

4.8 Les pipelines de gestion d'un type de noisette

4.8.1 Le pipeline `type_noisette_completer_description`

Ce pipeline est appelé au travers du service `ncore_type_noisette_completer_description()` juste avant qu'un type de noisette soit stocké dans l'espace de données du plugin utilisateur après la lecture de son YAML. Le plugin utilisateur à l'occasion de compléter la description par des champs spécifiques ou des modifications de champs existant. C'est le cas du plugin noiZetier qui complète le type de noisette avec le type et la composition éventuellement associée.

Les informations fournies en entrée du pipeline sont les suivantes :

PIPELINE TYPE_NOISETTE_COMPLETER_DESCRIPTION	
Index args	
plugin	Le préfixe du plugin utilisateur.
stockage	Identifiant de la méthode de stockage ou chaine vide si celle du plugin utilisateur est bien utilisée.
Index data	
	Le tableau de la description du type de noisette complété en sortie du pipeline. N-Core fournit au départ le tableau de base avant le stockage dans l'espace requis.

4.8.2 Le pipeline `type_noisette_styler`

Le champ « css_saisies » d'un type de noisette est destiné à recevoir la liste des saisies permettant de calculer le champ css de la noisette. Par défaut, N-Core initialise ce champ avec une saisie de type input texte. Néanmoins, il est possible de personnaliser cette saisie en la complétant ou en la remplaçant par une ou plusieurs autres saisies.

Pour cela, un plugin utilisateur doit coder son propre service-pipeline `type_noisette_styler` pour retourner sa version des saisies pour le type de noisette concerné. Ce service est un pipeline, le plugin utilisateur doit donc le définir comme tel.

Par défaut, la constante `_NCORE_COMPLEMENT_SAISIES_CSS` est initialisée à `true` ce qui indique que les saisies fournies en retour du pipeline doivent être ajoutées à la saisie par défaut et non la remplacer.

Les informations fournies en entrée du pipeline sont les suivantes :

PIPELINE TYPE_NOISETTE_STYLER	
Index args	
plugin	Le préfixe du plugin utilisateur.
stockage	Identifiant de la méthode de stockage ou chaine vide si celle du plugin utilisateur est bien utilisée.
Index data	
	Le tableau des saisies. N-Core fournit au départ un tableau vide mais renvoie un tableau avec une unique saisie de type texte si aucun plugin ne fournit des saisies alternatives.

Un plugin utilisateur a la possibilité d'ajouter des saisies qui s'appliquent à l'ensemble des types de noisette ou à un type de noisette donné. Un exemple est donné ci-dessous :

```
// Saisies communes à tous les types de noisette
$saisies['*'] = array(
  array(
    'saisie' => 'selection',
    'options' => array(
      'nom' => 'couleur',
      'label' => 'Classe : couleur (toutes les noisettes)',
      'data' => array(...),
    ),
  ),
);
// Saisies spécifiques au type de noisette "cartouche d'article".
$saisies['article-cartouche'] = array(
  array(
    'saisie' => 'selection',
    'options' => array(
      'nom' => 'taille',
      'label' => 'Classe : taille (uniquement pour cartouche article)',
      'data' => array(...),
    ),
  ),
);

// On complète les saisies fournies en entrée.
$flux['data'] = array_merge_recursive($flux['data'], $saisies);
```

4.8.3 Le pipeline `type_noisette_lister_categories`

Ce pipeline est appelé au travers du service `ncore_type_noisette_lister_categories()` qui relaye l'API `type_noisette_repertorier_categories()`. C'est le seul moyen de fournir la description de toutes les catégories utilisables car il n'existe pas de définition YAML ou autre des catégories.

N-Core fournit par défaut la description de la catégorie « defaut ». Le plugin noiZetier, par exemple, complète la liste avec les catégories « type » et « composition » qui définissent la compatibilité d'un type de noisette.

Les informations fournies en entrée du pipeline sont les suivantes :

PIPELINE TYPE_NOISETTE_LISTER_CATEGORIES	
Index args	
plugin	Le préfixe du plugin utilisateur.
Index data	
	Le tableau des descriptions de catégories indexé par l'identifiant de chaque catégorie et complété en sortie du pipeline. N-Core fournit au départ le tableau initialisé avec la catégorie « defaut ».

N-Core déclare la description de la catégorie « defaut » de la manière suivante :

```
$categories = array(
  'defaut' => array(
    'nom'         => '<:ncore:type_noisette_categorie_defaut_label:>',
    'description' => '<:ncore:type_noisette_categorie_defaut_description:>',
    'icone'       => 'defaut-24.png'
  ),
);
```

4.9 Les pipelines de complétion d'une noisette

4.9.1 Le pipeline `noisette_completer_description`

Ce pipeline est appelé au travers du service `ncore_noisette_completer_description()` juste avant qu'une nouvelle noisette soit stockée dans l'espace de données du plugin utilisateur ou qu'une noisette existante soit mise à jour (paramétrage). Le plugin utilisateur à l'occasion de compléter la description par des champs spécifiques. C'est le cas du plugin noiZetier qui complète la noisette avec les éléments de la page ou de l'objet qui l'accueille (champs type, composition, objet, id_objet et bloc).

Les informations fournies en entrée du pipeline sont les suivantes :

PIPELINE NOISETTE_COMPLETER_DESCRIPTION	
Index args	
plugin	Le préfixe du plugin utilisateur.
action	Désigne l'action effectuée sur la noisette, soit 'ajouter' ou 'parametrer'. C'est en fait le verbe de la fonction d'API appelante.
stockage	Identifiant de la méthode de stockage ou chaine vide si celle du plugin utilisateur est bien utilisée.
Index data	

	Le tableau de la description de la noisette complétée en sortie du pipeline. N-Core fournit au départ le tableau de base avant le stockage dans l'espace requis.

4.9.2 Le pipeline `noisette_completer_action`

Ce pipeline est appelé au travers du service `ncore_noisette_completer_action()` juste après un traitement effectué avec succès sur une noisette : c'est un pipeline de post-traitement. Le plugin utilisateur à l'occasion de compléter le traitement effectué sur la noisette sachant que celui-ci a réussi. Il peut concerner la noisette elle-même ou d'autres noisettes liées.

Les informations fournies en entrée du pipeline sont les suivantes :

PIPELINE NOISETTE_COMPLETER_ACTION	
Index args	
plugin	Le préfixe du plugin utilisateur.
noisette	Description de la noisette concernée par l'action.
action	Désigne l'action effectuée sur la noisette, soit 'ajouter', 'parametrer', 'deplacer', 'dupliquer' et 'supprimer'. C'est en fait le verbe de la fonction d'API appelante.
stockage	Identifiant de la méthode de stockage ou chaine vide si celle du plugin utilisateur est bien utilisée.
Index data	
	Tableau vide, l'index 'data' n'a aucune utilité pour le pipeline.

4.10 L'édition d'une noisette

4.10.1 Formulaire d'édition

Le plugin N-Core ne fournit aucune interface utilisateur complète, seulement des éléments de base. C'est le cas du formulaire d'édition d'une noisette qui est entièrement géré par N-Core et que les plugins utilisateur peuvent utiliser sans surcharge car il possède nativement des capacités de personnalisation.

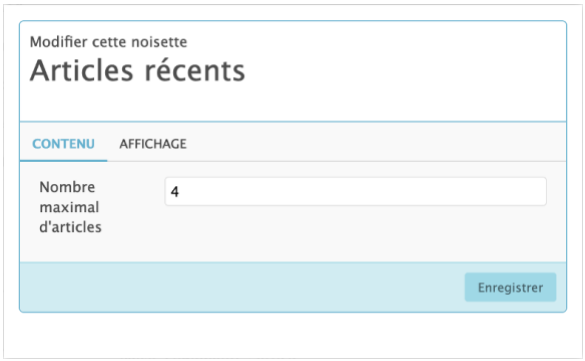
Le prototype d'appel du formulaire est :

```
#FORMULAIRE_EDITER_NOISETTE{plugin, id_noisette[, redirect]}.
```

Par défaut, le formulaire d'édition est organisé en plusieurs groupes sans que le plugin utilisateur n'ait à préciser quoi que ce soit :

- un premier groupe d'identifiant « **contenu** », accueillant toutes les saisies définies dans le bloc « parametres : » du YAML du type de noisette ;
- un second groupe d'identifiant « **affichage** », accueillant la saisie des classes CSS (champ « css » de la noisette) qui sont affectées à la capsule de la noisette ou à la noisette elle-même si celle-ci est un conteneur ;
- un troisième groupe d'identifiant « **avance** », accueillant l'indicateur d'encapsulation si le mode d'encapsulation automatique a été débrayé.

Les groupes sont matérialisés par des saisies de type fieldset qui englobent les autres saisies des paramètres. A partir de la version 3.46.0 du plugin Saisies, ces groupes sont affichés comme des onglets (voir ci-dessous), sinon ils sont affichés comme des fieldsets les uns sous les autres.



La liste des saisies du bloc « parametres: » du YAML peut être simple ou comporter des imbrications de fieldsets sans limitation. Le formulaire d'édition est entièrement construit en PHP au travers de la fonction `formulaires_editer_noisette_saisies_dist`.

4.10.2 Personnalisation des groupes de saisies

Il est possible de modifier la répartition des saisies dans les groupes de saisies proposés par défaut. Cela permet de ranger des paramètres de la noisette comme paramètres d'affichage ou paramètres avancés et ainsi présenter le formulaire d'édition de façon plus logique pour l'utilisateur.

Pour ce faire, il suffit d'insérer dans le bloc « parametres: » du YAML du type de noisette concerné, des clés correspondant aux identifiants de groupes. Cette organisation sera ensuite reprise pour l'affichage du formulaire. Un exemple est fourni ci-après.

```
parametres:
 contenu:
  -
   saisie: 'input'
   options:
    nom: 'lieu'
    label: '<:rainette:noisette_label_lieu:>'
    explication: '<:rainette:noisette_explication_lieu:>'
  -
   saisie: 'fieldset'
   options:
    nom: 'fieldset_intermediaire'
```

```
    label: 'Service'
  saisies:
   -
    saisie: 'service_meteo'
    options:
     nom: 'service'
     label: '<:rainette:noisette_label_service:>'
     defaut: ''
     cacher_option_intro: 'oui'
affichage:
 -
  saisie: 'modele_24h'
  options:
   nom: 'modele'
   label: '<:rainette:noisette_label_modele:>'
   defaut: 'previsions_24h'
   cacher_option_intro: 'oui'
```

Il est aussi possible de rajouter des groupes personnalisés (avec des identifiants spécifiques) mais cette pratique n'est pas conseillée car elle rompt l'organisation standard mise en place.

4.10.3 La personnalisation des saisies CSS

Une noisette est principalement stylée via sa capsule sauf si elle est elle-même un conteneur. Pour réaliser le stylage de la noisette, lors de son encapsulation, la capsule ou la noisette conteneur reçoit le champ « css » de la noisette.

Ce champ « css » est éditable via une saisie incluse dans le groupe « affichage », et qui, par défaut, est de type input de texte. Néanmoins, il est possible de personnaliser cette saisie en la remplaçant par une ou plusieurs autres saisies qui in fine sont compilées dans le champ « css ». Pour cela, un plugin utilisateur doit coder son propre service-pipeline `type_noisette_styler` pour retourner sa version des saisies. Ce pipeline est décrit au paragraphe 4.8.2.

5. DONNEES DE N-CORE

5.1 La structure des données

5.1.1 Les types de noisette

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	
type_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 ».
conteneur	Indicateur précisant si le type de noisette peut être un conteneur de noisettes ou pas. Prend les valeurs « oui » ou « non » (par défaut).
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. L'absence de conexte dans le fichier YAML est traduite par un contexte « env ». Pour un conteneur on force le contexte à « aucun ».
ajax	Indicateur d'inclusion en ajax de la noisette lors de la compilation. Prend les valeurs « defaut » (par défaut), « oui » ou « non ».
inclusion	Indicateur d'inclusion dynamique de la noisette lors de la compilation. Prend les valeurs « defaut » (par défaut), « statique » ou « 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.
categories	Liste au format ',cat1,cat2,cat3,' des catégories affectées au type de noisette. Le format spécial est nécessaire pour effectuer des recherches simplifiées dans le cas où le stockage est une base de données SQL.

Données complémentaires	
groupes	Tableau, éventuellement vide, définissant le regroupement des paramètres du type de noisette en fieldsets ou en onglets lors de l'édition d'une noisette du type concerné.
css_saisies	Tableau construit au chargement à partir du service `ncore_type_noisette_styler()` et contenant les saisies constitutives du champ « css » d'une noisette.
plugin	L'identifiant du plugin utilisateur, à savoir, en général, son préfixe.
actif	Indicateur précisant si les plugins nécessités par le type de noisette sont tous activés ou pas. Prend les valeurs « oui » (par défaut) ou « non ». Si aucun plugin n'est nécessité, l'indicateur vaut toujours « oui ».
signature	md5 du fichier YAML calculé lors de son chargement.

Ces données, qui proviennent principalement des fichiers YAML associés, 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 via le service prévu à cet effet. Ces données ne sont jamais modifiées unitairement mais complètement lors du premier chargement ou d'un rechargement 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 conteneur 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 d'identification et de localisation	
id_conteneur	Identifiant unique du conteneur au format chaine de caractères. La fonction de calcul de cet identifiant à partir des éléments du conteneur doit être réversible de façon à facilement en déduire ces mêmes éléments.
rang_noisette	Position de la noisette dans la liste des noisettes incluses dans le même squelette. Ce champ est un entier supérieur ou égal à 1.
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).
conteneur	Tableau associatif représentatif du conteneur dans lequel est inclus la noisette. Sa composition dépend totalement du plugin utilisateur sauf pour une noisette conteneur dont les index sont toujours le type et l'id.
type_noisette	Identifiant du type de noisette. Correspond au nom du fichier YAML sans extension.

est_conteneur	Indicateur précisant si la noisette peut être un conteneur pour d'autres noisettes ou pas. Si oui, l'affichage de la noisette tiendra compte des éventuelles noisettes incluses. Cet indicateur est une recopie du champ conteneur du type de noisette.
profondeur	Niveau de profondeur de la noisette. Ce champ est un entier supérieur ou égal à 0. La valeur 0 indique que la noisette est dans le conteneur de plus haut niveau et une valeur supérieure indique que la noisette est dans un conteneur noisette.
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.
encapsulation	Indicateur d'encapsulation de la noisette dans une capsule. Prend les valeurs « defaut » (par défaut), « oui » ou « non ». Pour une noisette conteneur, l'encapsulation vaut toujours « non ». Utilisé uniquement si `_NCORE_ENCAPSULATION_AUTO` est à `false`.
css	Styles CSS à affecter à la capsule ou à la noisette conteneur au format chaine.
Données complémentaires	
css_saisies	Styles CSS (duplication du champ « css ») au format tableau.
plugin	L'identifiant du plugin utilisateur, à savoir, en général, son préfixe.

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 via le service prévu à cet effet.

5.2 Les espaces de stockage obligatoires de N-Core

N-Core utilise plusieurs caches (fichiers PHP sécurisé) pour accéder rapidement à des données utiles à la compilation des noisettes. Ces caches ne peuvent pas être surchargés par le plugin appelant.

5.2.1 Les éléments de contexte des types de noisette

Ce cache se contente de consolider pour chaque type de noisette le tableau des éléments de contexte qui seront à minima fournis à la compilation. Cette information est configurée dans le fichier YAML caractérisant le type de noisette sous le tag `contexte:`.

Ce cache est géré par la fonction `noisette_contextualiser()`.

5.2.2 L'indicateur d'inclusion Ajax des types de noisette

Ce cache consolide pour chaque type de noisette l'indication d'inclusion de la noisette en Ajax ou pas. Cette information est tout d'abord configurée dans le fichier YAML caractérisant le type de noisette sous le tag `ajax:` et peut prendre les valeurs `defaut`, `oui` ou `non`. Si la valeur est absente ou

vaut `defaut`, une configuration globale permet de déterminer la valeur par défaut. L'information dans le cache est un booléen qui indique si le type de noisette doit être utilisé en Ajax ou pas.

Ce cache est géré par la fonction `type_noisette_ajaxifier()`.

5.2.3 L'indicateur d'inclusion dynamique des types de noisette

Ce cache consolide pour chaque type de noisette l'indication du mode d'inclusion des noisettes de ce type, à savoir statique ou dynamique. Cette information est configurée dans le fichier YAML caractérisant le type de noisette sous le tag `inclusion:`. L'information dans le cache est un booléen qui indique si le type de noisette doit être utilisé dynamiquement ou pas.

Ce cache est géré par la fonction `type_noisette_dynamiser()`.

5.3 Les espaces de stockage optionnels de N-Core

Pour simplifier le développement d'un plugin manipulant des noisettes, N-Core propose par défaut un espace de stockage pour les types de noisettes et un autre pour les noisettes. Les plugins appelant peuvent utiliser l'un ou l'autre ou les deux espaces proposés par N-Core ou développer leur propre espace de stockage comme le fait le plugin noiZetier.

5.3.1 Les types de noisette

N-Core stocke les descriptions des types de noisette telles que définies au paragraphe 5.1.1 dans un cache (fichier PHP 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 (fichier PHP 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. La signature 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()`.

5.3.2 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 conteneurs utilisés par le plugin utilisateur. Chaque affectation de noisette est un tableau indexé par identifiant de conteneur et par rang dans le conteneur. L'identification d'une noisette par le couple (identifiant de conteneur, rang) est donc optimale pour le stockage N-Core.

5.4 La configuration de N-Core

La configuration du plugin est limitée à deux constantes qui sont expliquées ci-dessous. La valeur par défaut est indiquée dans la colonne de droite.

SOUS-BLOC REQUETE		
_NCORE_ENCAPSULATION_AUTO	Indicateur d'encapsulation automatique. Par défaut indique que toute les noisettes non conteneur sont incluses dans une capsule.	`true`
_NCORE_COMPLEMENT_SAISIES_CSS	Mode d'utilisation du pipeline `type_noisette_styler`. Par défaut, implique que les saisies en retour du pipeline complètent la saisie libre positionnée par défaut.	`true`

6. REGLES DE CODAGE

6.1 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 traduit 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()`.

6.2 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`. 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 `$information` désigne toujours un champ de la description d'une noisette ou d'un type de noisette. L'argument `$type_noisette` désigne toujours l'identifiant d'un type de noisette qui coïncide avec le nom du fichier YAML sans extension.

De même `$id_noisette` et `$id_conteneur` représentent toujours respectivement l'identifiant unique d'une noisette ou d'un conteneur.

Par contre, l'argument `$noisette` s'il identifie bien de façon unique une noisette, peut revêtir deux formes : celle d'un *id* unique - i.e. `$id_noisette` - ou celle d'un couple *(identifiant de conteneur, rang)* – i.e. `$id_conteneur` et `$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.

Enfin, l'argument `$conteneur` identifie le conteneur soit par son identifiant unique – i.e. `$id_conteneur` – soit de façon explicite par son tableau associatif.

7. LES FICHIERS YAML

7.1 Les types de noisettes

La description d'une noisette est toujours fournie par un fichier YAML dont le nom correspond à l'identifiant du type de noisette. Le modèle d'un fichier descriptif de noisette - (`noisettes/type_noisette.yaml_template`) est présenté ci-dessous.

```yaml
# Titre du type de noisette
# - obligatoire
# - texte ou item de langue
nom: '<:ncore:type_noisette_xxxx_nom:>'
# Description du rôle du type de noisette
# - facultatif, vide si absent
# - texte ou item de langue
description: '<:ncore:type_noisette_xxxx_description:>'
# Nom de l'icône représentant le type de noisette sans chemin
# - facultatif, 'noisette-24.png' si absent
icon: 'xxxx-24.png'
# Indique si la noisette est un conteneur ou pas
# - facultatif, 'non' si absent
# - 'oui' ou 'non'
conteneur: 'oui'
# Liste des variables de contexte à passer à la noisette
# - facultatif, 'env' si absent
# - 'aucun', 'env' ou le tableau des noms de variables
contexte: 'aucun'
# Indique la méthode ajax à appliquer à ce type de noisette
# - facultatif, 'defaut' si absent
# - 'defaut', 'oui' ou 'non'
ajax: 'non'
# Indique la méthode d'inclusion de ce type de noisette
# - facultatif, 'defaut' si absent
# - 'defaut', 'statique' ou 'dynamique'
inclusion: 'statique'
# Liste des plugins nécessités pour le fonctionnement de la noisette
# - facultatif, [] si absent
# - tableau des préfixes de plugin
necessite: ['prefixe1', 'prefixe2']
# Liste des catégories associées au type de noisette (facultatif, ['defaut'] si absent)
categories: ['categorie1', 'categorie2']
# Liste des paramètres de la noisette qui seront proposés dans un formulaire
# - facultatif, [] si absent
# - tableau des configuration de saisies des paramètres (cf. plugin SAISIES)
parametres:
 -
  saisie: 'selection'
  options:
   nom: 'param_1'
   label: '<:ncore:label_param_1:>'
   defaut: 'data1'
   datas:
    d1: 'data1'
    d2: 'data2'
```

Un schéma JSON (`noisettes/type_noisette.schema.json`) est aussi fourni et pourrait être utilisé à terme pour valider les YAML des types de noisettes.

8. GRAPHE D'APPEL API – SERVICES

API – SERVICE	
API des types de noisette	
type_noisette_charger	ncore_type_noisette_initialiser_dossier ncore_type_noisette_lister ncore_type_noisette_styler [p] ncore_type_noisette_completer_description [p] ncore_type_noisette_stocker
type_noisette_lire	ncore_type_noisette_decrire ncore_type_noisette_traiter_typo
type_noisette_repertorier	ncore_type_noisette_lister
type_noisette_decacher	cache_supprimer
type_noisette_repertorier_categories	ncore_type_noisette_lister_categories [p]
API des noisettes	
noisette_ajouter	type_noisette_lire ncore_conteneur_verifier ncore_conteneur_identifier ncore_conteneur_construire ncore_conteneur_est_noisette ncore_noisette_decrire ncore_noisette_completer_description [p] ncore_noisette_lister ncore_noisette_ranger ncore_noisette_stocker ncore_noisette_completer_action [p]
noisette_deplacer	ncore_noisette_decrire ncore_conteneur_verifier ncore_conteneur_identifier ncore_noisette_lister ncore_conteneur_construire ncore_conteneur_est_noisette ncore_noisette_changer_conteneur ncore_noisette_ranger ncore_noisette_completer_action [p]

noisette_dupliquer	ncore_noisette_decrire noisette_ajouter noisette_parametrer ncore_noisette_lister noisette_dupliquer ncore_noisette_completer_action [p]
noisette_lire	ncore_noisette_decrire ncore_noisette_traiter_typo
noisette_parametrer	ncore_noisette_decrire ncore_noisette_completer_description [p] ncore_noisette_stocker ncore_noisette_completer_action [p]
noisette_supprimer	ncore_noisette_decrire ncore_conteneur_destocker ncore_noisette_lister ncore_noisette_ranger ncore_noisette_completer_action [p]
noisette_repertorier	ncore_conteneur_verifier ncore_noisette_lister
API des conteneurs	
conteneur_construire	ncore_conteneur_construire
conteneur_identifier	ncore_conteneur_verifier ncore_conteneur_identifier
conteneur_est_noisette	ncore_conteneur_construire ncore_conteneur_verifier ncore_conteneur_est_noisette
conteneur_vider	ncore_conteneur_construire ncore_conteneur_verifier ncore_conteneur_destocker

9. PROTOTYPES DES API

9.1 API des types de noisette

---

boolean **type_noisette_charger**(*string* $plugin , *boolean* $recharger = false , *string* $stockage = '')

Charge ou recharge les descriptions des types de noisette à partir des fichiers YAML. La fonction optimise le chargement en effectuant uniquement les traitements nécessaires en fonction des modifications, ajouts et suppressions des types de noisette identifiés en comparant les md5 des fichiers YAML.

---

array **type_noisette_decacher**(*string* $plugin , *array* $fonctions = array())

Supprime tout ou partie des caches liés à la compilation.

---

array | *string* **type_noisette_lire**(*string* $plugin , *string* $type_noisette , *string* $information = '' , *boolean* $traiter_typo = false , *string* $stockage = '')

Retourne, pour un type de noisette donné, la description complète ou seulement un champ précis. Les champs textuels peuvent subir un traitement typo si demandé.

---

array **type_noisette_repertorier**(*string* $plugin , *array* $filtres = array() , *string* $stockage = '')

Renvoie une liste de descriptions de types de noisette éventuellement filtrée sur certains champs fournis en argument.

---

array **type_noisette_repertorier_categories**(*string* $plugin, *string* $id_categorie = '')

Renvoie la liste des catégories de type de noisette et leur description. Il est possible de demander toutes les catégories ou juste une seule désignée par son identifiant.

9.2 API des noisettes

---

integer | *string* | *boolean* **noisette_ajouter**(*string* $plugin , *string* $type_noisette , *array* $conteneur , *integer* $rang = 0 , *string* $stockage = '')

Ajoute dans un conteneur, à un rang donné ou en dernier rang, une noisette d'un type donné. La fonction met à jour les rangs des autres noisettes si nécessaire.

boolean **noisette_deplacer**(*string* $plugin , *mixed* $noisette , *array* | *string* $conteneur_destination , *integer* $rang_destination , *string* $stockage = '')

Déplace une noisette donnée au sein d'un même conteneur ou dans un autre conteneur. La fonction met à jour les rangs des autres noisettes si nécessaire.

boolean **noisette_dupliquer**(*string* $plugin , *mixed* $noisette , *array* | *string* $conteneur , *integer* $rang = 0 , *null* | *array* $parametrage = null , *string* $stockage = '')

Duplique une noisette donnée dans un conteneur destination et, si cette noisette est un conteneur, duplique aussi les noisettes contenues de façon récursive.

array | *string* | *integer* **noisette_lire**(*string* $plugin , *mixed* $noisette , *string* $information = '' , *boolean* $traiter_typo = false , *string* $stockage = '')

Retourne, pour une noisette donnée, la description complète ou seulement un champ précis. Les champs textuels peuvent subir un traitement typo si demandé.

boolean **noisette_parametrer**(*string* $plugin , *mixed* $noisette , *array* $modifications , *string* $stockage = '')

Met à jour les paramètres éditables d'une noisette donnée. La fonction contrôle la liste des champs modifiables.

array **noisette_repertorier**(*string* $plugin , *array* | *string* $conteneur = array() , *string* $cle = 'rang_noisette' , *array* $filtres = array() , *string* $stockage = '')

Renvoie une liste de descriptions de noisettes appartenant à un conteneur donné ou pas et éventuellement filtrée sur certains champs. Le tableau retourné est indexé soit par identifiant de noisette soit par identifiant du conteneur et rang de la noisette.

boolean **noisette_supprimer**(*string* $plugin , *mixed* $noisette , *string* $stockage = '')

Supprime une noisette donnée du conteneur auquel elle est associée et, si cette noisette est un conteneur, le vide de ses noisettes au préalable. La fonction met à jour les rangs des autres noisettes si nécessaire.

9.3 API des conteneurs

array **conteneur_construire**(*string* $plugin , *string* $id_conteneur , *string* $stockage = '')

Reconstruit le conteneur sous forme de tableau canonique à partir de son identifiant unique (fonction inverse de conteneur_identifier). Cette fonction est juste un wrapper pour le service ncore_conteneur_construire() mais est très utilisée par les plugins utilisateur.

boolean **conteneur_est_noisette**(*string* $plugin , *string* | *array* $conteneur , *string* $stockage = '')

Détermine si un conteneur est une noisette ou pas.

string **conteneur_identifier**(*string* $plugin , *array* $conteneur , *string* $stockage = '')

Calcule l'identifiant unique pour le conteneur sous forme de chaine. Cette fonction est juste un wrapper pour le service ncore_conteneur_identifier(). Elle est utilisée par les balises #NOISETTE_COMPILER et #CONTENEUR_IDENTIFIER.

boolean **conteneur_vider**(*string* $plugin , *array* | *string* $conteneur , *string* $stockage = '')

Supprime toutes les noisettes d'un conteneur. L'éventuelle imbrication de conteneurs est gérée dans la fonction de service ncore_conteneur_destocker().

9.4 API de compilation

array **noisette_contextualiser**(*string* $plugin , *array* $noisette , *string* $type_noisette , *integer* $profondeur , *array* $environnement , *string* $stockage = '')

Renvoie le contexte de la noisette sous la forme d'un tableau éventuellement vide. Cette fonction gère un cache des contextes non valorisés des types de noisette disponibles.

string **noisette_encapsuler**(*string* $plugin , *string* $contenu , *string* $encapsulation , *array* $parametres)

Encapsule, si demandé, le contenu compile d'une ou d'un ensemble de noisettes dans un balisage HTML plus ou moins complexe appelé une capsule. Une noisette conteneur est considérée comme une capsule et donc traitée en tant que tel.

boolean **type_noisette_ajaxifier**(*string* $plugin , *string* $type_noisette , *string* $stockage = '')

Détermine si le type de noisette spécifié doit être inclus en AJAX ou pas. Cette fonction gère un cache des indicateurs ajax.

boolean **type_noisette_dynamiser**(*string* $plugin , *string* $type_noisette , *string* $stockage = '')

Détermine si la noisette spécifiée doit être incluse dynamiquement ou pas. Cette fonction gère un cache des indicateurs d'inclusion dynamique.

string **type_noisette_localiser**(*string* $plugin , *string* $type_noisette = '')

Renvoie le dossier relatif des types de noisette pour le plugin appelant ou la localisation du type de noisette demandé. Cette fonction gère le cas particulier de la noisette conteneur fournie par N-Core qui est, elle, toujours dans le dossier par défaut de N-Core.