SPIP

SPIP **24/06/2018**

Référence SPIP/N/2017.002

GUIDE
CONCEPTION DU PLUGIN N-CORE

SPIP

FICHE D'IDENTIFICATION	
Rédacteur	Eric Lupinacci
Projet	SPIP
Étude	Conception du plugin N-Core
Nature du document	Guide
Date	24/06/2018
Nom du fichier	Guide N - Le plugin N-Core.docx
Référence	SPIP/N/2017.002
Dernière mise à jour	24/06/2018 14:23:57
Langue du document	Français
Nombre de pages	24

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 0.4.3 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. 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 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

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).

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).

2.3 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 le noiZetier, par défaut, 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.

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 à l'exception de l'index `squelette` qui 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, défini spécifiquement par le plugin utilisateur (par exemple, `content/article|article|12`). Le calcul de cet identifiant doit être réversible et permettre de retrouver le tableau associatif du conteneur à partir de son identifiant.

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 ;
- La gestion de fichiers caches pour le stockage à accès rapide des types de noisettes et de certains éléments de contexte.

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 une fonction de service. 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 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

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 et met à jour le rang des autres noisettes du conteneur si nécessaire.
noisette_lire	Retourne, pour une noisette, la description complète ou seulement un champ précis. 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_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 couple (identifiant de conteneur, rang).
noisette_supprimer	Supprime une noisette donnée et met à jour les rangs des autres noisettes du conteneur si nécessaire.

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.

API CONTENEURS : INC/NCORE_CONTENEUR.PHP	
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.

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

3.4 L'API de compilation des noisettes

3.4.1 L'affichage 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]}`.

Pour fonctionner, la balise `#NOISETTE_COMPILER` utilise des filtres nécessaires à la détermination du contexte de la noisette, de son inclusion en ajax , de son inclusion dynamique ou pas et de 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.
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. 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 uniquement un vue synthétique des paramètres de la noisette.

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}.
```

3.5 L'API de gestion des caches

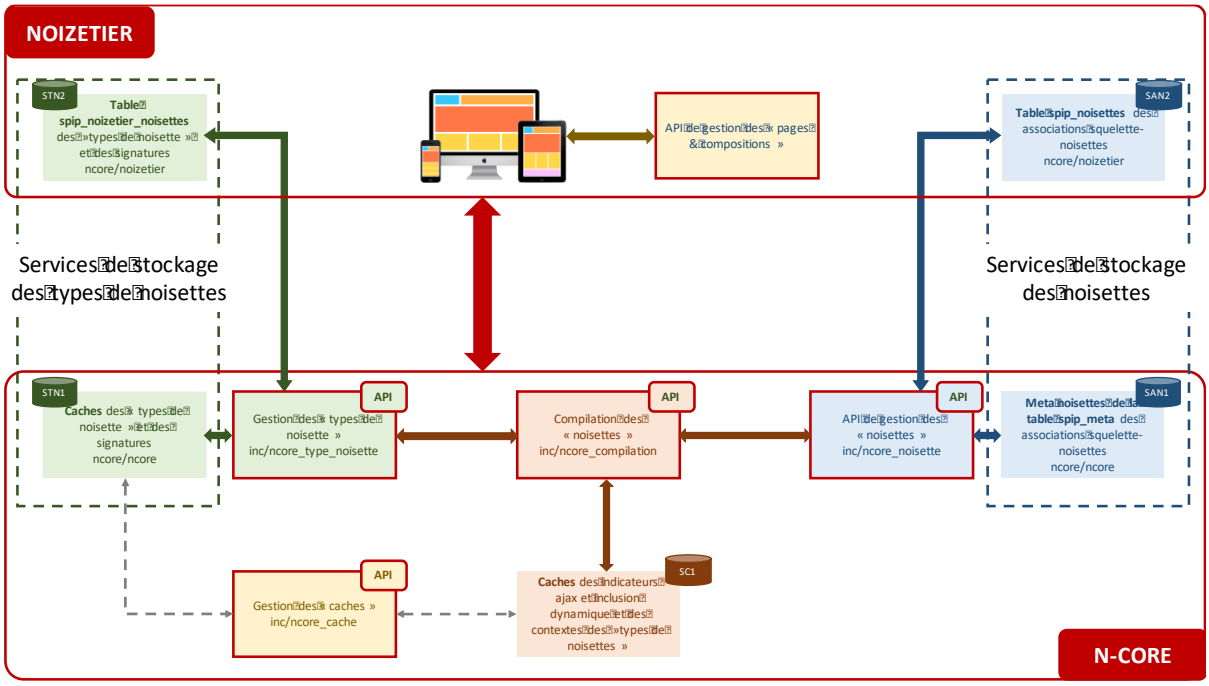
La gestion des caches est principalement à usage interne N-Core car celui-ci utilise de nombreux caches sécurisés comme moyen de stockage. Néanmoins, cette interface est exposée afin de rester utilisable par d'autres plugins.

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

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, conteneur et compilation). Néanmoins, un plugin utilisateur pourrait se passer des API de gestion des types de noisette, noisettes et conteneurs 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 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 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`.

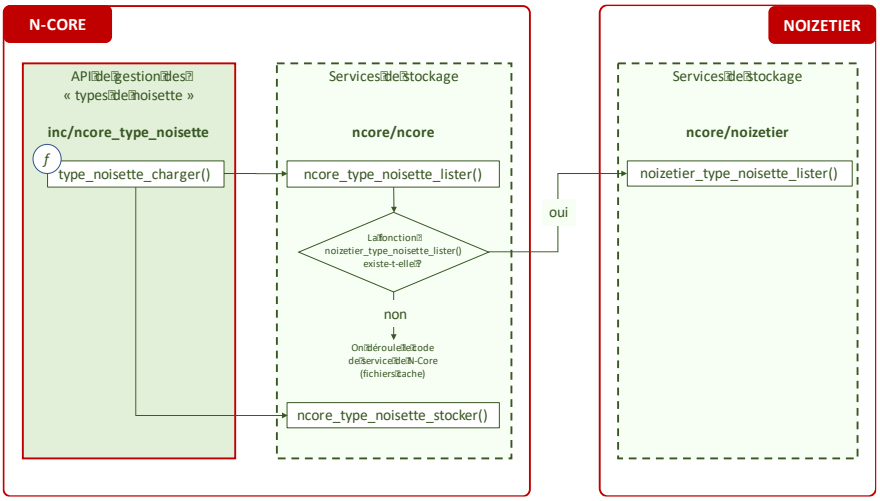
L'argument `$stockage`, lui, permet la réutilisation des services d'un plugin utilisateur par un autre plugin utilisateur ou de forcer l'utilisation des services N-Core (ce qui se fait plus simplement en omettant de créer le service homonyme dans le plugin appelant). Par exemple, le plugin utilisateur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

```
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.



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 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.

SERVICES	
Groupe des services de gestion des conteneurs	
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.
conteneur_destocker	Retire, de l'espace de stockage, toutes les noisettes d'un conteneur.
Groupe des services de gestion des types de noisette	
type_noisette_completer	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 est facultatif.
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_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_lister	Renvoie, pour l'ensemble des types de noisette utilisés par le plugin appelant, le champ demandé ou la description complète 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.
Groupe des services de gestion des noisettes	
noisette_completer	Complète si nécessaire la description d'une noisette avec des champs spécifiques au plugin utilisateur. Ce service est facultatif.

noisette_decrire	Renvoie la description brute d'une noisette sans traitement typo des champs textuels ni desérialisation des champs de type tableau sérialisé.
noisette_destocker	Efface de l'espace de stockage la description d'une noisette donnée.
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.

Les services notés (*) 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()` 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. Néanmoins, le service `conteneur_identifier()` de N-Core ne retourne rien par défaut ; N-Core considère que la construction de l'identifiant est toujours un sujet spécifique 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.

4.5 La compilation des noisettes

4.5.1 L'appel à `recuperer_fond()`

N-Core utilise une balise nommée `#NOISETTE_COMPILER` pour afficher 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 ;
- Le **contexte de la noisette** qui est une combinaison de l'environnement de la page dans laquelle sont inclues 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 défini au niveau de chaque type de noisette ;
- l'**indicateur d'inclusion en 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 0. 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.

4.5.2 L'inclusion optionnelle dans une balise

Pour chaque noisette il est possible de configurer si elle sera ou pas incluse dans une balise englobante lors de son affichage. Dans le cas où une balise englobante est configurée, il est possible de l'agrémenter par des classes CSS additionnelles.

Pour l'instant **N-Core ne gère pas la balise `<div>` englobante** lors de la compilation : il convient au plugin appelant de s'en charger (voir le paragraphe suivant).

4.5.3 L'affichage d'une liste de noisettes

N-Core fournit un squelette nommé `compiler_noisettes.html` qui affiche une liste de noisettes passées en argument en faisant appel à la balise `#NOISETTE_COMPILER` et propose une gestion de la balise `<div>` englobante ainsi que de l'imbrication des noisettes conteneur.

Ce squelette peut être utilisé si le besoin du plugin appelant est satisfait sinon il convient de créer son propre squelette sur ce modèle (c'est le cas du plugin noiZetier). A minima, ce squelette peut servir d'exemple pour un plugin appelant, le code est donc présenté ci-dessous.

N-Core propose aussi un squelette nommé `compiler_conteneur.html` qui répertorie les noisettes d'un conteneur et appel le squelette `compiler_noisettes.html` avec comme argument la liste de noisettes répertoriées.

```
[(#REM) <!--  #COMPILER NOISETTES
   Compile une liste de noisettes fournies en paramètres de l'inclusion.

   @api
   @param string plugin
          Identifiant qui permet de distinguer le module.
   @param array  noisettes
          Liste des descriptions de noisette à compiler dans l'ordre de leur rang.
   @param string defaut balise
          Valeur à utiliser si la noisette est paramétrée avec la valeur `defaut`.
   @param string stockage
          Identifiant du service de stockage à utiliser si précisé.
-->]
<BOUCLE_noisettes(DATA) {source table, #ENV{noisettes, #ARRAY}}{plugin}{par rang noisette}>
   #SET{conteneur, #ARRAY{type_noisette, #VALEUR{type_noisette}, id_noisette,
#VALEUR{id_noisette}}}
   <B_noisettes_enfants>
      #SET{balise_conteneur, #VALEUR{conteneur_balise}|sinon{div}}
      [<(#GET{balise_conteneur})[ class="(#VALEUR{conteneur_css})"]>]
   <BOUCLE_noisettes_enfants(DATA)
      {source table, #NOISETTE_REPERTORIER{#ENV{plugin}, #GET{conteneur}, #ENV{stockage}}}
      {par rang_noisette}>
      #SET{conteneur, #ARRAY{type_noisette, #VALEUR{type_noisette}, id_noisette,
#VALEUR{id_noisette}}}

<BOUCLE_noisettes_petits_enfants(BOUCLE noisettes enfants)></BOUCLE_noisettes_petits_enfants>
      #SET{avec_div, #VALEUR{balise}|=={defaut}|?{#ENV{defaut_balise, oui}, #VALEUR{balise}}}
      [[(#GET{avec_div}|oui)<div class="noisette[ noisette_(#VALEUR{type_noisette})][
(#VALEUR{css})]">]
         (#NOISETTE_COMPILER{#VALEUR{id_noisette}, #ENV{stockage, ''}})
      [(#GET{avec div}|oui)</div>]]
   </BOUCLE_noisettes_enfants>
      [</(#GET{balise_conteneur})>]
   </B_noisettes_enfants>
   #SET{avec_div, #VALEUR{balise}|=={defaut}|?{#ENV{defaut_balise, oui}, #VALEUR{balise}}}
   [[(#GET{avec_div}|oui)<div class="noisette[ noisette_(#VALEUR{type_noisette})][
(#VALEUR{css})]">]
      (#NOISETTE_COMPILER{#VALEUR{id_noisette}, #ENV{stockage, ''}})
   [(#GET{avec_div}|oui)</div>]]
</BOUCLE_noisettes>
```

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.
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.
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 éventuellement limité à l'index `squelette` dans lequel est inclus la noisette Sa composition dépend totalement du plugin utilisateur.
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.
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.

Le champ conteneur est fourni par N-Core au plugin appelant alors qu'il est redondant avec le champ id_conteneur dont le calcul doit être réversible. Il peut donc être omis dans le stockage spécifique du plugin appelant. Par exemple, N-Core le stocke dans sa meta alors que le noiZetier l'exclut de sa table `spip_noisettes`.

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. Seules les données de paramétrage et le rang (en rouge) peuvent être modifiées unitairement après la création de la noisette. Les autres données sont, elles, statiques après création.

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 `defaut`, une configuration globale permet de déterminer cette 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.

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`. 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 `$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, 'statique' si absent
# - '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 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'
  -
    saisie: 'input'
    options:
      nom: 'param_2'
      label: '<:ncore:label_param_2:>'
```