# Réorganisation des fichiers
# - Tiny v1.5.3 -

Alexis DUBOIS - adubois@cudl-lille.fr -

# 1- Introduction

Ce document a pour but d'expliquer la mise en oeuvre de la réorganisation des fichiers dans la version 1.5.3 de Tiny.

Il est composé d'une explication des intérêts de la réorganisation, d'une explication sur la séparation des éléments du noyau (core), des éléments du back et des éléments du site.

Il est à noter que lors d'une mise à jour vers la version 1.5.3, seule la séparation des éléments du site (partie 5) est ***pénalisante***.

Dans le cas d'une première installation de Tiny à partir de la version 1.5.3, la réorganisation est déjà effective.

Dans tout le document la mention 'sid' correspond à l'identifiant d'un site.

# 2- Intérêts

Le premier intérêt de cette réorganisation est de séparer les éléments communs à tous les sites des éléments spécifiques à un site.

Le deuxième intérêt concerne l'accès aux différents répertoires des sites. Il est désormais possible de restreindre pour un développeur son espace de travail à un répertoire de site. ***Cela évite d'éventuelles modifications des éléments communs***. L'accès aux répertoires 'core/' et 'back/' pouvant être réservé aux personnes qui coordonnent les modifications et mises à jour des versions de Tiny.

# 3- Séparation des éléments du noyau

Organisation initiale :

- 'core/private/classes/' contient les objets de mapping communs et spécifiques.
- 'core/private/classes/portlets/' contient les portlets communs et spécifiques.

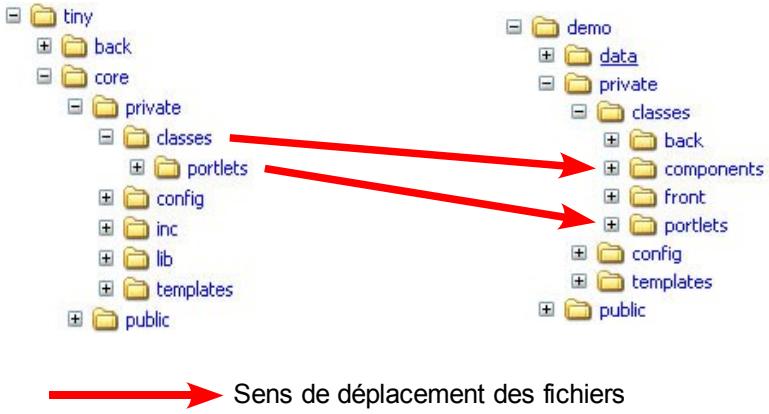
Nouvelle organisation :

Le but est de positionner dans le répertoire du site les éléments qui lui sont spécifiques. Les éléments communs à tous les sites restant à l'emplacement initial.

- 'sid/private/classes/components/' (à créer) contiendra les objets de mapping spécifiques.
- 'sid/private/classes/portlets/' (existant) contiendra les portlets spécifiques.

Arborescence :

Dans l'arborescence on considère sid = 'demo'.




Cette migration des fichiers du noyau vers le site courant a nécessité la mise en place d'un mécanisme automatique d'inclusion des classes métiers. Il n'est plus nécessaire de renseigner les différents « classes.inc » du noyau ou du site pour inclure les classes déplacées. C'est pourquoi ces fichiers contenus dans 'core/private/config/' et 'sid/private/config/' peuvent être modifiés (voir encadré ci-dessous).

Modification des fichiers « classes.inc » du noyau et du site :

```
<?php
/*
 * Fichier classes.inc
 * -------------------
 * Effectue les include des fichiers de configuration de l'application
 */
require_once C_CORE_CLASSES_PATH."Menu.class";
require_once C_CORE_CLASSES_PATH."Bloc.class";
require_once C_CORE_CLASSES_PATH."Faq.class";
require_once C_CORE_CLASSES_PATH."DhtmlMenu.class";
require_once C_CORE_CLASSES_PATH."Sondage.class";
require_once C_CORE_CLASSES_PATH."MiniAgenda.class";
require_once C_CORE_CLASSES_PATH."SitUtil.class";
require_once C_CORE_CLASSES_PATH."FlashInfo.class";
require_once C_CORE_CLASSES_PATH."Glossaire.class";
require_once C_CORE_CLASSES_PATH."CmdDoc.class";
require_once C_CORE_CLASSES_PATH."ObjetMetier1.class"; // Objet métier d'un site 1
require_once C_CORE_CLASSES_PATH."ObjetMetier2.class"; // Objet métier d'un site 2
?>
```

NB : l'inclusion automatique des classes ne concerne que le répertoire 'components/' du site.

# 4- Séparation des éléments du back

Organisation initiale :

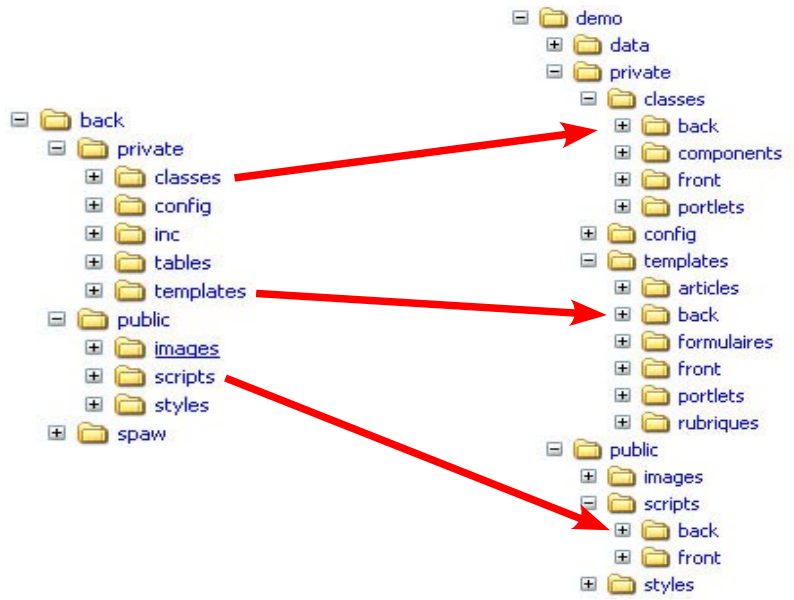
- 'back/private/classes/' contient les classes d'administration communes et spécifiques
- 'back/private/templates/' contient les templates des fonctionnalités communes et spécifiques.
- 'back/public/scripts/' contient les scripts des fonctionnalités communes et spécifiques.

Nouvelle organisation :

- 'sid/private/classes/back/' (à créer) contiendra les classes spécifiques.
- 'sid/private/templates/back/' (à créer) contiendra les templates spécifiques.
- 'sid/public/scripts/back' (à créer)  contiendra les scripts spécifiques.

Arborescence :

Dans l'arborescence on considère sid = 'demo'.




Sens de déplacement des fichiers

Modification des fichiers « sid/private/classes/back/*.class » :

Il faut aussi intervenir sur les fichiers « *.class » déplacés. En effet, ceux-ci contiennent des variables qui font référence à l'emplacement des templates et javascripts initial. Le choix a été fait de repasser sur tous ces fichiers pour modifier le code (voir encadré ci dessous).

```
<?
// Inclusion du fichier javascript
$this->oEnv->setJavascript("javascriptsite1", "<script language='JavaScript' src='".$this->oEnv-
>getConstante("SITE_JAVASCRIPTS_MANAGER_PATH")."JavascriptSite1.js' type='text/javascript'></script>");


// Association du template principal
$this->setTemplate($this->oEnv->getConstante("SITE_TEMPLATES_MANAGER_PATH")."templatesite1_manager_modifier.htm");
?>
```

# 5- Séparation des éléments du site

Suite aux réorganisations précédentes, nous avons choisi aussi de réorganiser le répertoire du site lui-même. Nous avons créé les répertoires :

- 'sid/private/classes/front/' contiendra les classes du front.
- 'sid/private/templates/front/' contiendra les templates du front.
- 'sid/public/scripts/front/' contiendra les scripts du front.

Ici contrairement aux deux points précédents, un simple déplacement des fichiers est nécessaire. Il faut prendre les fichiers 'sid/private/classes/*.class' et les placer dans 'sid/private/classes/front/', les templates de 'sid/private/templates/*.htm' vers 'sid/private/templates/front/' et enfin les fichiers javascripts de 'sid/public/scripts/*.js' vers 'sid/public/scripts/front/'.

Si vous ne souhaitez  pas effectuer cette réorganisation. Dans ce cas la seule modification à faire se trouve dans le fichiers 'core/private/config/host.inc'. Il faut modifier la valeur de la variable SUFIX_FRONT  .

```
<?
SUFIX_FRONT = "front/"
?>
```