PARTIE IV — Pour le développeur intégrateur

CHAPITRE 23 — Le catalogue produit dynamique en profondeur

Le catalogue produit dynamique transforme votre table `llx_product` en pages web générées automatiquement. Ce chapitre détaille l'ensemble du fonctionnement côté développeur : architecture, classes, configuration et points d'extension.

Architecture en couches

Couche	Rôle
StudioProductCatalog	Lit les produits Dolibarr (filtres, tris, multilingue) et expose un format normalisé.
StudioSolutionWrapper	Génère et supprime les wrappers Apache `solution-<ref>.php` en fonction des produits publiés.
Le gabarit solution-detail (page<X>.tpl.php)	Le gabarit unique qui affiche un produit. Une seule page Dolibarr Website pour N produits.
Trigger PRODUCT et CATEGORY	Régénère automatiquement les wrappers à chaque modification de produit ou de catégorie.
Tâche planifiée horaire	Filet de sécurité : relance la génération si un trigger a été manqué.

Le modèle de données produit

Chaque produit Dolibarr utilisé dans le catalogue exploite plusieurs tables :

Table	Données
`llx_product`	Champs natifs : ref, label, description, prix, tosell.

`llx_product_lang`	Traductions natives (label, description par langue).
`llx_product_extrafields`	Champs personnalisés canoniques : tagline, hero_image, badge, cta_label, cta_url, deployment, compatibility, support, languages, features (JSON), pricing_tiers (JSON), infrasstudio_published.
`llx_infrasstudio_product_translation`	Surcharges par langue des champs personnalisés traduisibles.
`llx_categorie_product`	Liens entre produits et catégories (utilisés pour la cartographie d'univers).

Provisionner les champs personnalisés

Les onze champs personnalisés nécessaires ne sont pas livrés en standard avec Dolibarr. Pour les créer en une commande, utilisez le script de provisionnement générique :

```
php htdocs/custom/infrasstudio/scripts/preset_default.php \
    htdocs/custom/infrasstudio/presets/keaticweb.json
```

Le fichier JSON livre les définitions par défaut (voir Chapitre 28 pour le format). Vous pouvez créer votre propre preset si vous avez besoin d'autres champs personnalisés.

Utiliser StudioProductCatalog

La classe expose plusieurs méthodes pour interroger les produits :

```
dol_include_once('/infrasstudio/class/studioproductcatalog.class.php');
$catalog = new StudioProductCatalog($db);

// Tous les produits publiés, dans la langue courante
$products = $catalog->fetchPublishedProducts(array(), $iso2);

// Filtres
$products = $catalog->fetchPublishedProducts(array(
    'univers' => 'supply-chain',
    'type'    => 'saas',
), $iso2);
```

```
// Un produit par référence
$product = $catalog->fetchProductBySlug('supplyflow-pro', $iso2);

// Univers utilisés (pour le filtre du catalogue)
$univers = $catalog->fetchUsedUnivers();
```

Format normalisé d'un produit

```
array(
    'id'           => 5,
    'ref'          => 'supplyflow-pro',
    'label'        => 'SupplyFlow Pro',                    // résolu selon la langue
    'description'  => '<p>Description...</p>',
    'price'        => 290.00,
    'badge'        => 'Nouveau',
    'hero_image'   => '/medias/...',
    'cta_label'    => 'Demander une démo',
    'cta_url'      => '/contact',
    'tagline'      => 'Optimisez votre supply chain...',
    'deployment'   => 'Cloud SaaS',
    'compatibility'=> 'SAP, Oracle, Sage',
    'support'      => '24/7',
    'languages'    => 'fr,en,de',
    'features'     => array('Prévision IA', 'Multi-entrepôts'),
    'pricing_tiers'=> array(...),
    'univers'      => 'supply-chain',
    'type'         => 'saas',
)
```

Cartographie catégorie vers univers

Les produits sont rattachés à des catégories Dolibarr. Le module associe chaque catégorie à un univers (concept éditorial : Supply Chain, Health, Legal, etc.).

Cartographie par défaut

```
// Les six univers livrés par défaut
'supply-chain'        => array(...catégories supply chain),
'health'              => array(...catégories santé),
'legal'               => array(...catégories juridique),
'digital-humanities'  => array(...catégories sciences humaines),
```

```
'transversal'         => array(...catégories outils transversaux),
'fsm'                 => array(...catégories field service)
```

Surcharge par JSON

Pour personnaliser, définissez la constante :

```
// Via dolibarr_set_const ou via la page de configuration admin
$conf->global->INFRASSTUDIO_PRODUCT_UNIVERS_MAP = json_encode(array(
    'mon-univers' => array(12, 13, 14),  // identifiants de catégories
    'autre'       => array(15, 16),
));
```

StudioSolutionWrapper

Cette classe gère la génération des wrappers Apache à partir des produits publiés.

Génération manuelle

```
dol_include_once('/infrasstudio/class/studiosolutionwrapper.class.php');
require_once DOL_DOCUMENT_ROOT . '/website/class/website.class.php';

$website = new Website($db);
$website->fetch(0, 'monsite');

$wrapper = new StudioSolutionWrapper($db);
$stats = $wrapper->rebuildAll($website, '/var/www/monsite');
print_r($stats);
// array('created' => 3, 'updated' => 2, 'deleted' => 1, 'skipped' => 5)
```

Anatomie d'un wrapper généré

```
<?php
// /var/www/monsite/solution-supplyflow-pro.php
// Generated by StudioSolutionWrapper — do not edit manually.
$solution_ref = 'supplyflow-pro';
$infrasstudio_current_product_ref = 'supplyflow-pro';
global $dolibarr_main_data_root, $conf;
if (empty($dolibarr_main_data_root)) {
    $res = include './page42.tpl.php';   // page solution-detail
} else {
```

```
    $res = include $dolibarr_main_data_root
        . ($conf->entity > 1 ? '/' . $conf->entity : '')
        . '/website/monsite/page42.tpl.php';
}
if ($res === false) { http_response_code(500); print 'Failed to make include'; }
```

> **Sécurité anti-collision —** Le module ne touche qu'aux wrappers qu'il a lui-même créés (présence du marqueur « StudioSolutionWrapper » dans l'en-tête du fichier). Les pages Dolibarr standards portant un slug commençant par `solution-` sont préservées.

Configuration du générateur de wrappers

Constante	Rôle
`INFRASSTUDIO_WEBSITE_KEY`	Référence du site cible.
`INFRASSTUDIO_PUBLIC_DOCROOT`	Docroot Apache absolu (par exemple `/var/www/monsite`).
`INFRASSTUDIO_SITE_<id>_WRAPPER_PREFIX`	Préfixe des wrappers (par défaut `solution-`). Exemples : `produit-` , `service-` .
`INFRASSTUDIO_SITE_<id>_WRAPPER_TEMPLATE_PAGEURL`	Slug du gabarit solution-detail (par défaut `solution-detail`).

Le gabarit solution-detail

Il s'agit d'une page Dolibarr Website classique (avec son slug et son tpl.php), mais utilisée par tous les wrappers. Elle reçoit la variable globale `$solution_ref` qui identifie le produit à afficher.

Squelette type

```
<?php
// page42.tpl.php (slug='solution-detail', type_container='page')
require_once __DIR__ . '/master.inc.php';
require_once DOL_DOCUMENT_ROOT . '/core/lib/website.lib.php';
```

```
require_once DOL_DOCUMENT_ROOT . '/core/website.inc.php';

dol_include_once('/infrasstudio/class/studioproductcatalog.class.php');
$catalog = new StudioProductCatalog($db);
$iso2 = infrasstudio_current_lang();
$product = $catalog->fetchProductBySlug($solution_ref, $iso2);
if (!$product) {
    http_response_code(404);
    print '<h1>Produit introuvable</h1>';
    exit;
}

ob_start();
try {
?>
<html lang="<?php echo $iso2; ?>">
<head>
    <title><?php echo dol_escape_htmltag($product['label']); ?></title>
</head>
<body>
    <section class="hero">
        <h1><?php echo dol_escape_htmltag($product['label']); ?></h1>
        <p><?php echo dol_escape_htmltag($product['tagline']); ?></p>
        <img src="<?php echo $product['hero_image']; ?>" alt="...">
    </section>
    <section class="features">
        <ul>
        <?php foreach ($product['features'] as $feature): ?>
            <li><?php echo dol_escape_htmltag($feature); ?></li>
        <?php endforeach; ?>
        </ul>
    </section>
</body>
</html>
<?php
} catch (Exception $e) { print $e->getMessage(); }
include dol_buildpath('/infrasstudio/core/tpl/website_output.tpl.php', 0);
```

Mélanger slots et données produit — Vous pouvez combiner les deux : les zones éditoriales (par exemple un titre marketing au-dessus de la grille) en slots, et les zones produit (label, prix,

fonctionnalités) lues via `StudioProductCatalog`. Les deux cohabitent sans conflit.

Le trigger PRODUCT et CATEGORY

Le module installe un trigger qui écoute :

- `PRODUCT_CREATE`
- `PRODUCT_MODIFY`
- `PRODUCT_DELETE`
- `PRODUCT_PRICE_MODIFY`
- `CATEGORY_LINK` (uniquement si l'objet lié est un produit)
- `CATEGORY_UNLINK`

À chaque événement, le trigger appelle `StudioSolutionWrapper::rebuildAll()` sur le site configuré.

> **Note —** Si `INFRASSTUDIO_WEBSITE_KEY` ou `INFRASSTUDIO_PUBLIC_DOCROOT` ne sont pas configurées, le trigger se termine silencieusement. Ni bruit, ni erreur. Cette discrétion convient aux instances Dolibarr qui n'utilisent pas le catalogue dynamique.

La tâche planifiée horaire (filet de sécurité)

Une tâche planifiée s'exécute toutes les heures et appelle `StudioSolutionWrapper::rebuildAllConfigured()`. Elle lit les constantes et relance la régénération.

Cette tâche apporte les bénéfices suivants :

- Rattrapage si un trigger a été manqué (import en masse, modification SQL directe, etc.).
- Synchronisation après une migration de serveur.
- Garantie de cohérence sans intervention manuelle.

Reconstruction manuelle en ligne de commande

```
php htdocs/custom/infrasstudio/scripts/rebuild_solution_wrappers.php \
    <ref-site-ou-id> <docroot-public> [entity]

# Exemple
php htdocs/custom/infrasstudio/scripts/rebuild_solution_wrappers.php \
    monsite /var/www/monsite 2
```

Récapitulatif

Vous savez désormais :

- Comprendre l'architecture du catalogue (StudioProductCatalog, StudioSolutionWrapper, gabarit, trigger, tâche planifiée).
- Provisionner les champs personnalisés produit avec le preset générique.
- Interroger les produits via `StudioProductCatalog`.
- Cartographier les catégories Dolibarr vers vos univers via `INFRASSTUDIO_PRODUCT_UNIVERS_MAP` .
- Comprendre comment `StudioSolutionWrapper::rebuildAll()` génère les wrappers Apache.
- Configurer le préfixe et le slug du gabarit par site.
- Écrire le gabarit `solution-detail` qui sert tous les produits.
- Comprendre le rôle du trigger automatique et de la tâche planifiée de sécurité.
- Lancer une reconstruction manuelle en ligne de commande.

Fin de la Partie IV — Vous maîtrisez désormais l'intégration côté développeur : préparer un site, annoter avec des slots, comprendre la grammaire, brancher des données Dolibarr, gérer le multilingue, créer vos gabarits et exploiter le catalogue dynamique. Vous êtes en mesure de livrer un site clé en main.

La Partie V aborde l'administration et la maintenance. Si vous êtes administrateur Dolibarr, c'est votre prochaine destination.

CHAPITRE 22 — Créer ses propres gabarits de page

Lorsque votre client clique sur « + Nouvelle page » dans le Studio, il choisit un gabarit de départ. Ce chapitre vous explique comment créer vos propres gabarits adaptés à la charte de votre site.

À quoi sert un gabarit

Un gabarit est un squelette de page qui comporte :

- Du HTML structuré (sections, classes CSS, balises sémantiques).
- Des slots pré-déclarés pour les zones éditables.
- Des valeurs par défaut, pour disposer d'une page complète dès la création.
- Des métadonnées (titre par défaut, slug, type de conteneur).

Lorsqu'un client crée une page depuis un gabarit, le module :

1. Crée une entrée dans la table `llx_website_page`.
2. Génère un fichier `page<N>.tpl.php` à partir du squelette.
3. Crée le wrapper Apache `<slug>.php`.
4. Lance un rescan des slots pour les détecter immédiatement.

Structure d'un gabarit

Un gabarit est un dossier dans `htdocs/custom/infrasstudio/templates/` :

```
templates/mon-gabarit/
├── meta.php             # Métadonnées du gabarit
└── skeleton.tpl.php     # Le squelette HTML avec slots
```

Le fichier meta.php

Il retourne un tableau de configuration :

```
<?php
return array(
    'code'          => 'mon-gabarit',                  // identifiant unique
    'label'         => 'Mon Gabarit',                  // affiché dans l'assistant
```

```
    'category'       => 'page',                          // page | landing | blog
    'description'    => "Description courte affichée sous le titre.",
    'icon'           => 'fa-file-lines',                 // icône FontAwesome
    'type_container' => 'page',                          // type Dolibarr Website
    'default_slug'   => 'nouvelle-page',                 // slug suggéré
);
```

Champ	Description
`code`	Identifiant unique. Doit correspondre au nom du dossier.
`label`	Texte affiché dans la grille de sélection de l'assistant.
`category`	Permet le regroupement visuel (page, landing, blog).
`icon`	Classe FontAwesome de l'icône de la tuile.
`type_container`	Valeur écrite dans `llx_website_page.type_container`. Valeurs standards : `page`, `blogpost`, `other`, `menu`.
`default_slug`	Slug suggéré dans le formulaire. Le client peut le modifier.

Le fichier skeleton.tpl.php

Il s'agit du squelette HTML, identique à un fichier tpl.php standard, à la différence qu'il contient des marqueurs qui seront remplacés au moment de la création.

Marqueurs disponibles

Marqueur	Remplacé par
`@@PAGEID@@`	L'identifiant Dolibarr de la nouvelle page (par exemple 42).

`@@PAGEURL@@`	Le slug URL final (par exemple `about-keatic`).
`@@ISO2@@`	Le code ISO2 de la langue principale (`fr`, `en`).

Exemple complet — gabarit page-libre

meta.php

```
<?php
return array(
    'code'           => 'page-libre',
    'label'          => 'Page libre',
    'category'       => 'page',
    'description'    => 'Une page simple avec un titre et un grand champ de texte riche.',
    'icon'           => 'fa-file-lines',
    'type_container' => 'page',
    'default_slug'   => 'nouvelle-page',
);
```

skeleton.tpl.php

```
<?php // BEGIN PHP
$websitekey = basename(__DIR__);
if (! defined('USEDOLIBARRSERVER') && ! defined('USEDOLIBARREDITOR')) {
    require_once __DIR__ . '/master.inc.php';
}
require_once DOL_DOCUMENT_ROOT . '/core/lib/website.lib.php';
require_once DOL_DOCUMENT_ROOT . '/core/website.inc.php';
ob_start();
try {
// END PHP ?>
<html lang="@@ISO2@@">
<head>
    <title>{{slot:page_title|type=text|default=Nouvelle page|label=Titre
SEO|group=seo}}</title>
    <meta name="description"
```

```
content="{{slot:page_meta_description|type=text|default=|label=Meta description|group=seo}}"
/>
    <link rel="canonical" href="<?php echo $website->virtualhost; ?>/@@PAGEURL@@.php" />
</head>
<body>
    <?php includeContainer('header'); ?>
    <main class="container">
        <h1>{{slot:page_h1|type=text|default=Titre principal|label=H1|group=hero}}</h1>
        <div class="content">
            {{slot:page_body|type=richtext|default=<p>Contenu de la page.</p>|label=Contenu}}
        </div>
    </main>
    <?php includeContainer('footer'); ?>
</body>
</html>
<?php // BEGIN PHP
} catch (Exception $e) { print $e->getMessage(); }
include dol_buildpath('/infrasstudio/core/tpl/website_output.tpl.php', 0);
// END PHP ?>
```

Le bloc final — N'oubliez pas la ligne `include dol_buildpath('/infrasstudio/core/tpl/website_output.tpl.php', 0);`. C'est elle qui déclenche la résolution des slots et shortcodes au moment du rendu.

Localiser vos gabarits hors du module

Si vous souhaitez livrer des gabarits avec votre projet client sans modifier le module, définissez la constante :

```
// htdocs/conf/conf.php ou via dolibarr_set_const
$conf->global->INFRASSTUDIO_TEMPLATE_EXTRA_DIR = '/var/www/monsite/templates';
```

Le module scanne ce répertoire en complément de `htdocs/custom/infrasstudio/templates/`.

Gabarits livrés par défaut

Code	Description
`page-free`	Page libre avec titre et un grand champ texte riche. Pour les pages ponctuelles.

`blog-standard`	Article de blog générique avec hero, introduction, corps et appel à l'action.
`example-blog`	Article de blog au design moderne (hero CSS, accroche en italique, image secondaire, articles liés). Adaptable à votre charte.
`example-landing`	Page de destination produit complète (environ 70 slots). Hero, problème, solution, fonctionnalités, contact, FAQ.

> **Conseil —** Inspirez-vous de `example-landing` pour comprendre comment structurer un gabarit complexe avec environ 70 slots organisés en sections.

Bonnes pratiques pour vos gabarits

- Préfixez tous les slots du gabarit par un identifiant commun (par exemple `landing_`) pour éviter les collisions entre gabarits.
- Regroupez les slots avec `group=` par section logique (hero, fonctionnalités, contact, etc.).
- Donnez des valeurs par défaut représentatives. Le rédacteur dispose ainsi d'un exemple à modifier plutôt que d'une page vide intimidante.
- Incluez les slots SEO (`page_title`, `page_meta_description`) dans tout gabarit de type page.
- Incluez les balises Open Graph dans l'en-tête HTML pour le partage social.
- Incluez le helper hreflang si le site est multilingue.
- Testez le gabarit en créant une page réelle depuis l'assistant et vérifiez le rendu public.

Récapitulatif

Vous savez désormais :

- Comprendre l'utilité d'un gabarit (squelette, slots, valeurs par défaut).
- Créer un dossier `templates/<code>/` avec `meta.php` et `skeleton.tpl.php`.
- Renseigner les métadonnées (code, label, category, icon, type_container, default_slug).
- Utiliser les marqueurs `@@PAGEID@@`, `@@PAGEURL@@` et `@@ISO2@@`.
- Localiser vos gabarits en dehors du module via `INFRASSTUDIO_TEMPLATE_EXTRA_DIR`.
- Suivre les bonnes pratiques (préfixe, regroupement, valeurs par défaut, SEO, multilingue).

Le dernier chapitre de la Partie IV détaille le catalogue produit dynamique en profondeur.

CHAPITRE 21 — Gérer le multilingue (pages sœurs)

Dolibarr Website propose deux modèles multilingues différents. Identifier le modèle que vous utilisez est une étape préalable à toute écriture de code. Ce chapitre vous présente les deux options et la manière dont le module s'y intègre.

Les deux modèles multilingues

Modèle	Caractéristique
A. Slot par langue (recommandé)	Un seul fichier tpl.php sert toutes les langues. Les slots disposent de surcharges par langue.
B. Pages sœurs (legacy)	Un fichier tpl.php par langue (`about.php` , `about-en.php` , `about-de.php`). Modèle hérité des sites Dolibarr Website classiques.

> **Recommandé —** Adoptez le modèle A pour tout nouveau site. Le modèle B reste pris en charge pour la migration progressive de sites existants. Le module fournit un outil de consolidation B vers A.

Modèle A — Slot par langue (le standard moderne)

Fonctionnement

Une seule page Dolibarr Website existe par concept (par exemple `about`). Le fichier tpl.php est unique. Les slots possèdent leur valeur canonique (FR par défaut) et des surcharges par langue stockées dans `llx_infrasstudio_slot`.

```
<!-- about.tpl.php — UN SEUL fichier pour FR/EN/DE/ES/IT/PT/NL/PL -->
<h1>{{slot:about_title|type=text|default=À propos de nous|label=Titre About}}</h1>
```

```
<p>{{slot:about_lead|type=textarea|default=Notre histoire en quelques
mots.|label=Accroche}}</p>
```

Récupérer la langue du visiteur dans le gabarit

Le module résout les slots automatiquement selon la langue détectée. Pour vos propres traitements PHP (afficher la date dans la bonne langue, choisir une image différente par langue), récupérez la langue via :

```
<?php
// helpers fournis par le module
$iso2 = infrasstudio_current_lang();         // 'fr', 'en', 'de', ...
$locale = infrasstudio_current_locale();     // 'fr_FR', 'en_US', 'de_DE', ...
?>
```

Cascade de détection

La fonction `infrasstudio_current_lang()` détecte la langue dans cet ordre :

1. Constante `INFRASSTUDIO_LANG_ISO` définie par le site (souvent dans `lang.inc.php`).
2. Paramètre URL `?lang=xx`.
3. Suffixe sur le slug (`about-en.php` donne en).
4. Cookie de persistance (`INFRASSTUDIO_LANG_COOKIE`).
5. Valeur de `$langs->defaultlang`.
6. En-tête HTTP `Accept-Language` du visiteur.

Modèle B — Pages sœurs (legacy)

Il s'agit du modèle Dolibarr Website classique : pour chaque page, vous créez un fichier tpl.php par langue.

```
about.php           # FR (canonique)
about-en.php        # EN
about-de.php        # DE
about-es.php        # ES
...
```

Si vous reprenez un site existant qui suit ce modèle, deux options s'offrent à vous :

1. Conserver le modèle avec des stubs du module.
2. Migrer vers le modèle A grâce à l'outil de consolidation en ligne de commande.

Conserver le modèle B avec des stubs

Le module fournit un helper `sister_stub.tpl.php` qui réduit chaque page sœur à trois lignes et permet de partager le HTML avec la canonique.

Procédure

La page canonique reste un tpl.php classique :

```
// page5.tpl.php — canonique FR
<?php // bootstrap dolibarr ... ?>
<html>...<h1>{{slot:about_title|type=text|default=À propos|...}}</h1>...</html>
```

Chaque page sœur devient un stub :

```
<?php
// page42.tpl.php — sister EN
$_infrasstudio_sister_canonical = 'page5.tpl.php';
$_infrasstudio_sister_lang      = 'en';
include dol_buildpath('/infrasstudio/core/tpl/sister_stub.tpl.php', 0);
```

Le stub force `$_GET['lang'] = 'en'` puis délègue le rendu à la canonique. Les surcharges EN sont automatiquement utilisées pour résoudre les slots.

Migrer du modèle B vers A en ligne de commande

Un script consolide une famille de pages sœurs en une page canonique et plusieurs stubs :

```
php htdocs/custom/infrasstudio/scripts/consolidate_sister_pages.php <ref-site> \
    [--entity=N] \
    [--base-slug=about] \
    [--dry-run] \
    [--extractor=/path/to/extractor.php]
```

Ce que fait le script

1. Détecte les groupes de pages sœurs parmi les codes ISO2 pris en charge (en, de, es, it, pt, nl, pl).
2. Pour chaque groupe :
   - Conserve la page tpl FR comme canonique.
   - Extrait les valeurs traduites depuis chaque page sœur (par expression régulière ou via un extracteur personnalisé).
   - Insère les surcharges dans `llx_infrasstudio_slot`.
   - Réécrit la canonique avec des slots `{{slot:...}}`.

- Remplace chaque page sœur par un stub de trois lignes.

3. Sauvegarde de chaque tpl original avec l'extension `.bak`.

> **Conseil —** Lancez d'abord avec `--dry-run` pour visualiser ce que le script va faire sans rien écrire. Lancez en mode réel uniquement après vérification.

hreflang : déclarer les alternates au navigateur

Pour que Google sache que vos pages sont des traductions les unes des autres, vous devez émettre des balises `<link rel="alternate" hreflang="...">` dans l'en-tête HTML.

Le module fournit un helper qui les génère automatiquement :

```
<head>
    ...
    <?php echo infrasstudio_hreflang_tags($website, $WEBSITE_PAGE); ?>
    ...
</head>
```

Sortie type :

```
<link rel="alternate" hreflang="fr" href="https://exemple.com/about.php" />
<link rel="alternate" hreflang="en" href="https://exemple.com/about-en.php" />
<link rel="alternate" hreflang="de" href="https://exemple.com/about-de.php" />
<link rel="alternate" hreflang="x-default" href="https://exemple.com/about.php" />
```

> **Compatibilité avec les deux modèles —** Pour le modèle A, le helper émet une seule URL canonique avec les langues différenciées par `?lang=`. Pour le modèle B, il émet une URL par page sœur.

Sélecteur de langue côté gabarit

Pour permettre au visiteur de changer de langue, vous devez exposer un sélecteur. Le helper `infrasstudio_translated_url($iso2)` génère l'URL équivalente de la page courante dans une autre langue :

```
<nav class="lang-switcher">
    <?php foreach (array('fr', 'en', 'de', 'es') as $iso): ?>
        <a href="<?php echo infrasstudio_translated_url($iso); ?>">
```

```
            <?php echo strtoupper($iso); ?>
        </a>
    <?php endforeach; ?>
</nav>
```

Charger les fichiers de langue du site

Si votre site utilise des fichiers `.lang` Dolibarr (par exemple pour les libellés de menu), chargez-les en début de tpl :

```
<?php
require_once __DIR__ . '/lang.inc.php';
$langs->loadLangs(array('website_main', 'website_blog'));
?>
```

Et utilisez les clés via `$langs->trans('Key')` ou via le slot avec `default=@lang:Key`.

Récapitulatif

Vous savez désormais :

- Distinguer le modèle A (slot par langue, recommandé) du modèle B (pages sœurs, legacy).
- Récupérer la langue du visiteur via `infrasstudio_current_lang()`.
- Convertir un site existant utilisant le modèle B en utilisant le helper `sister_stub.tpl.php`.
- Migrer en bloc B vers A grâce à l'outil `consolidate_sister_pages.php`.
- Émettre les balises hreflang automatiquement avec `infrasstudio_hreflang_tags`.
- Construire un sélecteur de langue avec `infrasstudio_translated_url`.

CHAPITRE 20 — Insérer des données Dolibarr (shortcodes)

Les slots stockent du contenu éditable. Les **shortcodes**, eux, affichent des données Dolibarr lues en direct (produits, catégories, informations société, médias). Ils sont résolus au moment du rendu, à chaque consultation de page.

Distinction entre slot et shortcode

Slot	Shortcode
Stocke du **contenu éditable** par le client.	Affiche des **données Dolibarr** lues en direct.
Persistance dans `llx_infrasstudio_slot`.	Aucune persistance — résolu à chaque requête.
Exemple : `{{slot:hero_title|type=text}}`	Exemple : `{{product:ref=supplyflow.label}}`

Syntaxe générale

```
{{<namespace>:<sélecteur>.<champ>}}
```

- **namespace** : la source de données (`product`, `category`, `dict`, `mysoc`, `extrafield`, `media`).
- **sélecteur** : la manière d'identifier l'élément (le plus souvent `ref=xxx` ou `id=N`).
- **champ** : la donnée à extraire de l'élément.

Namespace product

Lit un produit dans la table `llx_product`.

```
<h2>{{product:ref=supplyflow-pro.label}}</h2>
<p>{{product:ref=supplyflow-pro.description}}</p>
<span class="price">{{product:ref=supplyflow-pro.price}} €</span>
<span>{{product:ref=supplyflow-pro.ef_tagline}}</span>
```

```
<img src="{{product:ref=supplyflow-pro.ef_hero_image}}" alt="...">
```

Champs disponibles

Champ	Source
`label`, `description`, `note`	Champs natifs (traduits selon la langue du visiteur via `llx_product_lang`)
`price`, `price_ttc`, `cost_price`	Prix HT, TTC, coût (formaté selon la langue)
`ref`, `tosell`	Référence, statut commercialisable
`ef_<slug>`	Tout champ personnalisé (`ef_tagline`, `ef_hero_image`, etc.)

Le marqueur $current

Pour un même gabarit utilisé par plusieurs produits (par exemple `solution-detail` servi par les wrappers `solution-<ref>.php`), utilisez `ref=$current` :

```
<h1>{{product:ref=$current.label}}</h1>
<p>{{product:ref=$current.ef_tagline}}</p>
```

Le module résout `$current` via la variable globale `$infrasstudio_current_product_ref`, posée par le wrapper.

Namespace category

Lit une catégorie dans la table `llx_categorie`.

```
<h3>{{category:id=5.label}}</h3>
<p>{{category:id=5.description}}</p>
```

Namespace dict

Lit une entrée d'un dictionnaire Dolibarr (`c_country`, `c_currencies`, etc.).

```
<span>{{dict:c_country.code=FR.label}}</span>
<span>{{dict:c_currencies.code_iso=EUR.label}}</span>
```

Namespace mysoc

Lit les informations de la société courante (`$mysoc`).

```
<footer>
    © {{mysoc.name}} — {{mysoc.address}}, {{mysoc.zip}} {{mysoc.town}}
    Tél : {{mysoc.phone}} — Courriel : {{mysoc.email}}
    SIRET {{mysoc.idprof2}} — TVA {{mysoc.tva_intra}}
</footer>
```

Champs disponibles

`name`, `address`, `zip`, `town`, `country_code`, `phone`, `fax`, `email`, `url`, `capital`, `tva_intra`, `idprof1` à `idprof6`, `logo`, `logo_small`, `logo_squarred`.

Namespace extrafield

Lit un champ personnalisé d'un objet Dolibarr quelconque.

```
{{extrafield:table=product|ref=supplyflow.field=tagline}}
{{extrafield:table=societe|id=42|field=segment}}
{{extrafield:table=product|ref=$current|field=tagline}}
```

Plus générique que `{{product:ref=X.ef_tagline}}` mais plus verbeux. À utiliser lorsque le namespace dédié n'existe pas (par exemple pour `societe` ou `contact`).

Namespace media

Lit un média de la bibliothèque du module (`llx_infrasstudio_media`).

```
<img src="{{media:ref=hero-1.url}}" alt="{{media:ref=hero-1.alt}}">
<img src="{{media:ref=hero-1.thumb}}">
<img src="{{media:ref=hero-1.card}}">
<img src="{{media:ref=hero-1.wide}}">
```

Champs disponibles

Champ	Description
`url`	URL du fichier original
`thumb`	Variante 200 × 200
`card`	Variante 640 × 480
`wide`	Variante 1600 × 1200
`alt`	Texte alternatif (résolu selon la langue)
`label`, `width`, `height`	Métadonnées

Combiner shortcodes et slots

Vous pouvez intégrer un shortcode dans la valeur par défaut d'un slot, ou dans le contenu d'un slot de type texte riche :

```
<!-- Shortcode dans le default d'un slot -->
<p>{{slot:greeting|type=text|default=Bienvenue chez {{mysoc.name}}}}</p>

<!-- Shortcode dans un slot richtext (l'éditeur peut le saisir librement) -->
<div>{{slot:long_intro|type=richtext|default=Notre équipe à {{mysoc.town}} vous
accompagne.}}</div>
```

> **Cas d'usage —** Une page de remerciements après commande qui dit « Merci, votre commande sera livrée à... ». Le HTML reste statique côté gabarit, mais le rendu est personnalisé pour chaque visiteur.

Étendre avec un namespace personnalisé

Pour ajouter un namespace propre à votre projet, déposez un fichier dans `htdocs/custom/infrasstudio/shortcodes/` :

```
// shortcodes/myorg.shortcode.php
function infrasstudio_shortcode_myorg_resolve($args, $context)
```

```
{
    global $db;
    $id    = isset($args['id']) ? (int) $args['id'] : 0;
    $field = isset($args['_field']) ? $args['_field'] : 'name';
    if ($id <= 0) { return ''; }

    $sql = "SELECT name, sector FROM ".MAIN_DB_PREFIX."myorg WHERE rowid = ".$id;
    $resql = $db->query($sql);
    if (!$resql) { return ''; }
    $obj = $db->fetch_object($resql);
    $db->free($resql);
    return isset($obj->$field) ? (string) $obj->$field : '';
}
```

Utilisable dans n'importe quel gabarit :

```
<h2>{{myorg:id=12.name}}</h2>
<p>Secteur : {{myorg:id=12.sector}}</p>
```

Pièges et performance

Shortcode dans une boucle — Si vous résolvez cent fois le même shortcode dans une page, vous effectuez cent requêtes SQL. Mettez vos données en cache dans une variable locale avant la boucle, ou utilisez les fonctions natives Dolibarr (`Product::fetch`) en PHP.

Shortcode introuvable — Si `{{product:ref=inexistant.label}}` ne résout rien, le module retourne une chaîne vide silencieusement. Aucune erreur n'est affichée. Testez en local avant la livraison.

Shortcodes dans les attributs — Les shortcodes fonctionnent dans tous les contextes HTML, attributs compris : `<img src="{{media:ref=X.url}}">`, `<a href="{{slot:cta_url|...}}">`, etc.

Récapitulatif

Vous savez désormais :

- Distinguer slots (contenu éditable) et shortcodes (données Dolibarr).
- Utiliser les six namespaces livrés (product, category, dict, mysoc, extrafield, media).
- Utiliser `$current` dans les gabarits partagés entre plusieurs produits.
- Combiner shortcodes et slots.
- Ajouter votre propre namespace via un fichier dans `shortcodes/`.
- Anticiper les pièges (boucles, valeurs absentes).

CHAPITRE 19 — La grammaire des slots en détail

Référence exhaustive de tous les types de slot et de leurs attributs. Gardez ce chapitre à portée de main lors de l'annotation d'un template.

Format général

```
{{slot:<name>|type=<type>|default=<value>|label=<text>|group=<section>|help=<text>|maxlength=<int>|options=<csv>}}
```

Tous les attributs, à l'exception de `name` et `type`, sont facultatifs. L'ordre est libre. Le séparateur est le caractère pipe.

Règles syntaxiques

Règle	Détail
Identifiant	`[a-z0-9_]+`, 64 caractères maximum. Pas de tirets, pas de majuscules.
Unicité	Unique par page. Deux slots avec le même nom sur la même page produisent un avertissement au scan.
Pas d'espaces	Aucun espace de part et d'autre du signe `=`. Aucun espace dans les noms d'attributs.
Échappement	Le pipe à l'intérieur d'une valeur n'est pas pris en charge. Évitez-le dans les valeurs `default`.
Une seule ligne	Un token de slot tient sur une seule ligne. Aucun saut de ligne ne doit y être présent.

Les dix types de slot

Type text — Texte court

Champ d'une seule ligne, sans mise en forme.

```
<h1>{{slot:hero_title|type=text|default=Bienvenue|label=Titre principal|maxlength=80}}</h1>
```

- Interface : champ de saisie simple.
- Idéal pour : titres, libellés, boutons d'appel à l'action, copyright.
- Compteur de caractères affiché si `maxlength` est défini.

Type textarea — Texte long brut

Multi-lignes, sans mise en forme HTML.

```
<p class="lead">{{slot:hero_lead|type=textarea|default=Une
accroche.|label=Accroche|maxlength=300}}</p>
```

- Interface : zone de texte multi-lignes.
- Les retours à la ligne sont automatiquement convertis en balises `<br>` au rendu.
- Idéal pour : accroches, paragraphes simples.

Type richtext — Texte riche WYSIWYG

Mise en forme complète via CKEditor.

```
<div class="post-body">
    {{slot:post_body|type=richtext|default=<p>Contenu de l'article.</p>|label=Corps de
l'article}}
</div>
```

- Interface : éditeur CKEditor (Format, gras, italique, listes, liens, couleurs, alignement, etc.).
- Idéal pour : corps d'articles, descriptions longues, conditions générales.
- La valeur stockée contient du HTML (`<p>`, `<strong>`, etc.).

Type image — Média image

```
<img src="{{slot:hero_image|type=image|default=/medias/hero.jpg|label=Image hero}}" alt="...">
```

- Interface : champ texte, bouton de sélection dans la bibliothèque, bouton de suppression et vignette d'aperçu.
- Stocke soit une URL absolue, soit un shortcode `{{media:ref=X.url}}`.
- Non traduisible par défaut : la même image est servie dans toutes les langues.

Type url — Lien

```
<a href="{{slot:hero_cta_url|type=url|default=/contact|label=URL du bouton}}">...</a>
```

- Interface : champ texte avec validation URL légère.
- Idéal pour : URL de bouton, liens vers les réseaux sociaux, URL de partage.

Type icon — Icône FontAwesome

```
{{slot:feature_1_icon|type=icon|default=fa-solid fa-rocket|label=Icône feature 1}}
```

- Interface : champ classe, sélecteur de couleur, vingt icônes proposées en accès rapide, aperçu en direct.
- Stockage en JSON : `{"class":"fa-solid fa-star","color":"#fbbf24"}`.
- Rendu : `<i class="fa-solid fa-star" style="color:#fbbf24"></i>`.
- Sécurisé : liste blanche sur la classe (a-zA-Z0-9_-) et expression régulière hexadécimale sur la couleur.

Type color — Couleur

```
<section style="background-color:{{slot:section_bg|type=color|default=#19052d|label=Couleur de fond}}">
```

- Interface : sélecteur de couleur HTML5, champ hexadécimal, bouton de retour à la valeur par défaut.
- Format strict : `#RRGGBB` ou `#RRGGBBAA` (avec transparence).
- Non traduisible.

Type number — Nombre

```
<span class="stat">{{slot:stats_clients|type=number|default=42|label=Nombre de clients}}</span>
```

Type bool — Booléen

```
<?php if ('{{slot:hero_show_badge|type=bool|default=1|label=Afficher le badge}}'): ?>
    <span class="badge">Nouveau</span>
<?php endif; ?>
```

- Interface : case à cocher.
- Valeur : 0 ou 1.

Type select — Liste déroulante

```
<section class="hero hero-{{slot:hero_layout|type=select|options=light,dark,gradient|default=light|label=Style du
```

```
hero}}">
```

- Interface : menu déroulant.
- Les valeurs disponibles sont définies dans `options` (CSV obligatoire).

Valeurs par défaut spéciales

default=@lang:KEY — Résolution via les fichiers de langue

Lorsque votre site utilise déjà des fichiers `.lang` Dolibarr (cas typique pour les sites multilingues existants), vous pouvez réutiliser les clés au lieu de les dupliquer dans les slots :

```
<h1>{{slot:hero_title|type=text|default=@lang:HeroTitle|label=Titre du hero}}</h1>
```

Au moment du rendu, le module résout via `$langs->trans('HeroTitle')` selon la langue du visiteur. Les fichiers `.lang` du site présents dans `DOL_DATA_ROOT/<entity>/website/<ref>/langs/` sont chargés automatiquement.

> **Cas d'usage —** Migration progressive d'un site existant : vous conservez vos fichiers `.lang` en l'état, vous ajoutez les slots, et le client peut soit éditer dans le Studio (surcharge), soit conserver la traduction du `.lang` (canonique).

Récapitulatif des types et de leur traduisibilité

Type	Interface	Traduisible
`text`	Champ de saisie	Oui
`textarea`	Zone de texte	Oui
`richtext`	CKEditor	Oui
`image`	Sélecteur de média	Non
`url`	Champ de saisie	Non
`icon`	Classe et sélecteur de couleur	Non
`color`	Sélecteur de couleur	Non
`number`	Champ numérique	Oui (rare)
`bool`	Case à cocher	Non

`select`	Menu déroulant	Non

Pièges fréquents

Slots dans des attributs JSON intégrés — Si vous avez du JSON intégré dans une balise (par exemple `data-config="{...}"`), n'y placez pas de slot. Les doubles accolades sont aussi un délimiteur Mustache et risquent de casser les bibliothèques JavaScript qui parseraient ce JSON.

Identifiant trop long — La limite est de 64 caractères. Au-delà, le scanner rejette silencieusement, ce que met en évidence l'option `--lint`.

Espaces autour des = — Le scanner rejette `type = text`. Utilisez toujours `type=text`.

Caractère pipe dans une valeur — `default=A|B` casse l'analyseur. Utilisez une autre séparation, ou laissez `default` vide et saisissez la valeur via le Studio.

Récapitulatif

Vous savez désormais :

- La syntaxe complète d'un token de slot.
- Les dix types disponibles et leurs interfaces respectives.
- La résolution `@lang:KEY` pour réutiliser les fichiers .lang Dolibarr.
- Quels types sont traduisibles ou non.
- Les pièges syntaxiques fréquents.

Le chapitre suivant aborde les shortcodes, qui injectent des données Dolibarr dans le HTML.

CHAPITRE 18 — Annoter un template avec des slots

Le slot est l'unité de base du module. Ce chapitre vous montre comment transformer un HTML statique en HTML annoté, prêt à être édité par le client. Il s'agit de l'opération la plus fréquente dans votre travail d'intégrateur.

Le principe en deux phrases

1. Vous écrivez votre HTML normalement, comme vous le feriez sans le module.
2. Aux endroits que vous voulez rendre éditables, vous remplacez le contenu en dur par un token `{{slot:...}}`.

C'est tout. Le module se charge du reste : détection automatique, persistance, rendu, interface d'édition.

Premier exemple complet

Avant — page non éditable

```
<section class="hero">
    <h1>Bienvenue chez Keatic</h1>
    <p>Votre partenaire numérique de confiance depuis 2010.</p>
    <a href="/contact" class="btn">Nous contacter</a>
</section>
```

Après — page éditable via le Studio

```
<section class="hero">
    <h1>{{slot:hero_title|type=text|default=Bienvenue chez Keatic|label=Titre du
hero|group=hero}}</h1>
    <p>{{slot:hero_lead|type=textarea|default=Votre partenaire numérique de confiance depuis
2010.|label=Accroche|group=hero}}</p>
    <a href="{{slot:hero_cta_url|type=url|default=/contact|label=URL du bouton|group=hero}}"
class="btn">
        {{slot:hero_cta_label|type=text|default=Nous contacter|label=Libellé du
bouton|group=hero}}
    </a>
```

```
</section>
```

> **Résultat —** Le client voit dans le Studio un panneau « Hero » comportant quatre champs (titre, accroche, URL du bouton, libellé du bouton). Aucun HTML à modifier de son côté.

Anatomie d'un token de slot

```
{{slot:<identifiant>|type=<type>|attribut1=valeur1|attribut2=valeur2}}
```

Élément	Obligatoire	Description
`{{slot:`	Oui	Préfixe fixe qui déclare un slot.
Identifiant	Oui	Unique par page. Caractères acceptés : minuscules, chiffres, tiret bas. 64 caractères maximum.
`type=`	Oui	text, textarea, richtext, image, url, number, select, bool, icon, color.
`default=`	Recommandé	Valeur de repli si le slot n'est pas encore édité.
`label=`	Recommandé	Libellé affiché à l'éditeur dans le Studio.
`group=`	Facultatif	Regroupe les slots dans une section de l'interface.
`help=`	Facultatif	Aide affichée sous le champ.
`maxlength=`	Facultatif	Limite de caractères pour text et textarea.
`options=`	Si type=select	Liste CSV des valeurs possibles.

Conseil — La grammaire complète est détaillée au Chapitre 19.

Où placer un slot

Un slot peut être placé à de nombreux endroits dans votre HTML.

Dans le contenu d'une balise (cas le plus fréquent)

```
<h1>{{slot:title|type=text|default=Bienvenue}}</h1>
```

Dans un attribut HTML

```
<img src="{{slot:hero_image|type=image|default=/medias/hero.jpg}}" alt="...">
<a href="{{slot:cta_url|type=url|default=/contact}}">...</a>
<section style="background-color:{{slot:bg_color|type=color|default=#19052d}}">
```

Dans une chaîne de classes CSS

```
<i class="fa-solid {{slot:hero_icon|type=text|default=fa-rocket}}"></i>
```

> **À ne pas faire —** Ne placez pas un slot dans un commentaire HTML, dans un bloc `<script>`, ou dans du JSON intégré. Le scanner détecte tous les tokens et peut générer des slots fantômes.

Convention de nommage des identifiants

L'identifiant est libre, mais une convention claire facilite la maintenance.

Schéma recommandé

`<section>_<champ>` ou `<page>_<section>_<champ>`

Bon	À éviter
`hero_title`	`title` (trop générique, risque de collision)
`features_grid_card_3_label`	`truc3` (incompréhensible)
`footer_copyright`	`FooterCopyright` (les majuscules sont rejetées par le scanner)

> **Recommandé —** Si votre site comporte cinq sections (hero, features, stats, testimonials, footer), préfixez tous vos slots par leur section. Le client retrouvera ainsi plus rapidement ce qu'il cherche dans le Studio.

Utiliser le groupe pour structurer l'interface

L'attribut `group` détermine sous quelle section le slot apparaît dans le panneau « Contenu de la page » du Studio.

```
{{slot:hero_title|type=text|group=hero|...}}
{{slot:hero_lead|type=textarea|group=hero|...}}
{{slot:hero_image|type=image|group=hero|...}}

{{slot:features_card_1_title|type=text|group=features|...}}
{{slot:features_card_2_title|type=text|group=features|...}}
{{slot:features_card_3_title|type=text|group=features|...}}

{{slot:footer_copyright|type=text|group=footer|...}}
```

Dans le Studio, ces slots seront regroupés visuellement sous trois sections : Hero, Features et Footer.

Le rescan après ajout

Ajouter un slot dans un fichier `tpl.php` ne suffit pas : le module doit le détecter et l'enregistrer en base. Pour cela, lancez un rescan.

Méthode A — Via l'interface Studio

1. Outils → InfraSStudio → Contenu des pages.
2. Sélectionnez votre site.
3. Cliquez sur **Rescanner** en haut.
4. Le module parcourt tous les fichiers `tpl.php` et synchronise les slots.

Méthode B — En ligne de commande

```
php htdocs/custom/infrasstudio/scripts/rescan_slots.php <ref-du-site> --entity=<N>
```

Sortie typique :

```
Rescanning website #1 — monsite (entity 2)
Pages scanned   : 22
Slots added     : 3
Slots updated   : 40
Slots orphaned  : 0
```

Mode de validation — Ajoutez l'option `--lint` pour détecter les erreurs de syntaxe avant de les valider :
`php rescan_slots.php <ref> --lint`

Le code de sortie est 0 en l'absence d'erreur, 1 en présence d'avertissements, 2 en cas d'erreurs. Cette commande peut être intégrée dans un crochet de pré-commit.

Slots orphelins

Lorsque vous supprimez un slot d'un fichier `tpl.php`, son enregistrement en base passe au statut « orphelin ». Le module le conserve pendant 30 jours avant de le supprimer automatiquement (via une tâche planifiée).

Cette politique présente un avantage : si vous restaurez le slot dans le HTML pendant cette période, sa valeur précédente est récupérée.

> **Note —** Pour forcer la purge immédiate des slots orphelins :
> `php rescan_slots.php <ref> --purge-orphans`

Liste de contrôle d'annotation

Avant de livrer une page annotée :

- Chaque slot possède un `type` explicite.
- Chaque slot possède une valeur `default`, garantissant la lisibilité de la page si la base venait à se vider.
- Chaque slot possède un `label` en français clair, destiné à l'éditeur.
- Les slots sont regroupés par section avec `group`.
- Les identifiants suivent la convention `<section>_<champ>`.
- Le rescan a été lancé sans avertissement ni erreur.
- Vous avez ouvert la page dans le Studio et vérifié la présence de tous les slots.

Le chapitre suivant détaille la grammaire complète des slots, avec tous les types et leurs comportements.

CHAPITRE 17 — Préparer un site Dolibarr Website

Avant de pouvoir éditer un site via le module, ce site doit exister côté Dolibarr Website. Ce chapitre s'adresse au développeur qui démarre un nouveau projet : créer le site, le câbler proprement, et préparer le terrain pour le Studio.

Étape 1 — Créer le site dans Dolibarr Website

1. Connectez-vous à Dolibarr en tant qu'administrateur.
2. Rendez-vous dans Accueil → Sites web.
3. Cliquez sur **Nouveau site**.
4. Renseignez les informations suivantes :
   - **Référence** : identifiant court sans espace, par exemple `monsite` ou `keaticweb`. C'est l'identifiant interne, utilisé partout (URLs des wrappers, dossiers de données, constantes de configuration).
   - **Description** : libellé facultatif.
   - **Virtualhost principal** : URL publique, par exemple `https://monsite.com`.
   - **Langue principale** : code de la locale, par exemple `fr_FR`.
   - **Autres langues** : valeurs séparées par des virgules, par exemple `en_US,de_DE,es_ES`.
5. Enregistrez.

> **Convention de référence —** Privilégiez un identifiant court et stable. Cette référence apparaîtra dans les chemins de fichiers, les URLs internes et les constantes de configuration. La modifier ultérieurement nécessite plusieurs ajustements.

Étape 2 — Configurer le virtualhost Apache

Le module Website ne configure pas Apache pour vous. Vous devez créer un VirtualHost qui pointe sur le docroot du site.

Arborescence type

```
/var/www/monsite/                          # docroot Apache du site
├── index.php                               # wrapper page d'accueil
├── <alias>.php                             # autres wrappers générés
├── master.inc.php                          # bootstrap (inclut DOL_DOCUMENT_ROOT)
├── medias  -> lien symbolique vers DOL_DATA_ROOT/<entity>/medias/
```

```
└── ...

DOL_DATA_ROOT/<entity>/website/monsite/      # dossier de données du site
├── page1.tpl.php                            # gabarits des pages
├── page2.tpl.php
├── ...
└── htmlheader.html                          # en-tête commun éventuel
```

Exemple de VirtualHost minimal

```
<VirtualHost *:443>
    ServerName monsite.com
    DocumentRoot /var/www/monsite

    SSLEngine on
    SSLCertificateFile      /etc/letsencrypt/live/monsite.com/fullchain.pem
    SSLCertificateKeyFile   /etc/letsencrypt/live/monsite.com/privkey.pem

    <Directory /var/www/monsite>
        Options FollowSymLinks
        AllowOverride None
        Require all granted
    </Directory>

    # PHP-FPM dédié recommandé
    <FilesMatch \.php$>
        SetHandler "proxy:unix:/run/php/php8.2-fpm.monsite.sock|fcgi://localhost"
    </FilesMatch>
</VirtualHost>
```

> **Note —** Un pool PHP-FPM dédié est recommandé pour isoler les performances et les variables d'environnement par site. Le pool peut tourner sous un utilisateur spécifique pour faciliter la gestion des permissions.

Étape 3 — Le lien symbolique medias

Pour que les images téléversées soient servies directement par Apache (mode média « native »), il faut un lien symbolique :

```
cd /var/www/monsite
ln -sfn /mnt/data/dolibarr/<entity>/medias medias
ls -la medias  # doit pointer sur le dossier de données Dolibarr
```

> **Conseil —** Si vous ne pouvez pas créer ce lien symbolique, basculez le site en mode média *module* dans la configuration du module. Les images seront servies via `document.php`, avec un coût de performance modéré.

Étape 4 — Le master.inc.php multicompany

Si votre installation Dolibarr fonctionne en multicompany, le fichier `master.inc.php` du site doit définir `DOLENTITY` avant le chargement :

```
<?php
// /var/www/monsite/master.inc.php
define('DOLENTITY', 2);   // l'entity correspondant au client
require_once '/var/www/dolibarr/htdocs/master.inc.php';
```

> **Avertissement —** Sur une instance multicompany sans `DOLENTITY`, le site renverra une erreur 503 ou affichera les données de la mauvaise entity. C'est l'erreur la plus fréquemment rencontrée lors de la mise en service.

Étape 5 — Créer la page d'accueil

Toujours dans le module Website Dolibarr :

1. Sélectionnez votre site.
2. Cliquez sur **Nouvelle page**.
3. Renseignez :
   - **Page URL** : `home`, `index` ou autre.
   - **Title** : Accueil.
   - **Type container** : `page`.
   - **Lang** : `fr_FR`.
   - **Status** : Brouillon (le passage à Publié interviendra plus tard).
4. Enregistrez.
5. Dolibarr crée automatiquement les fichiers `/var/www/monsite/home.php` et `page<N>.tpl.php`.

Étape 6 — Activer le site dans le module

La procédure complète est détaillée au Chapitre 7. En résumé :

1. Outils → InfraSStudio → Configuration.
2. Cochez votre site dans la liste.
3. Choisissez le mode média (*native* recommandé).
4. Enregistrez.

Étape 7 — Annoter les premières pages avec des slots

Cette étape fait l'objet du Chapitre 18. En quelques mots : modifiez le fichier `page<N>.tpl.php` pour ajouter des tokens `{{slot:nom|type=...}}` aux endroits que vous souhaitez rendre éditables.

Étape 8 — Vérifier avec la page Diagnostic

Avant de livrer le site au client, lancez le diagnostic (Outils → InfraSStudio → Diagnostic) :

- Tous les voyants verts dans Environnement, Schéma SQL, Stockage et Intégration Dolibarr.
- Le site géré apparaît dans la section Sites avec docroot résolu, mode média correct et dossier de données accessible en écriture.

Récapitulatif

Votre site est prêt si :

- Le site est créé dans Dolibarr Website (référence, virtualhost, langues).
- Le VirtualHost Apache pointe sur le bon docroot.
- Le lien symbolique `medias` est en place (mode native), ou la constante `INFRASSTUDIO_SITE_<id>_MEDIA_MODE=module` est définie.
- Le fichier `master.inc.php` définit `DOLENTITY` en multicompany.
- Au moins une page existe (l'accueil).
- Le site est coché dans la configuration du module.
- La page Diagnostic est entièrement verte.

Le chapitre suivant aborde l'annotation des templates avec des slots.