Date : 12/01/2012
Version : 1.02.01
Etat : Validé
Confidentialité : Restreinte
Référence : J/JBC/167/DO/Drupal

Guide de développement Drupal Drupal

Document réalisé pour Jouve

Rédaction / Validation	Diffusion
Rédigée par :	**Destinataires client :**
Jean-Baptiste Combes	
Validée par :	**Destinataires Jouve :**
Sylvain Brochand	RCC
	Tous les développeurs Drupal

Version	État	Date	Auteur(s)	Modifications	Pages
1.00.00	Ebauche			Version originale	
1.01.00	Ebauche	12/01/2012	JBC		
1.01.00	Validé	12/01/2012	JBC		
1.02.00	Validé	12/01/2012	JBC	Traductions & déploiement	5
1.02.01	Validé	12/01/2012	SBR	Corrections	toutes

Guide de développement Drupal
Drupal

Table des matières

1 INTRODUCTION 3

2 RECOMMANDATIONS GENERALES 3

2.1 Données entrantes 3
2.1.1 Données passées par l'URL 3
2.1.2 Formulaires 3
2.1.3 Permissions 4
2.2 Affichage 4
2.3 Performances 4

3 THEMING 5

3.1 Style 6
3.2 Méthode 6
3.3 Maquette 6

4 DEVELOPPEMENT 6

4.1 Style à suivre 6
4.2 Accès aux données 6

1 INTRODUCTION

Ce document résume les consignes de développement Drupal.

2 RECOMMANDATIONS GENERALES

2.1 DONNEES ENTRANTES

Toutes les données d'entrée doivent être validées et contrôlées.

Il ne faut jamais utiliser une donnée fournie par l'utilisateur sans avoir vérifié qu'elle contient bien ce que l'on attend :

- Cela évite les injections SQL ;
- Cela évite (pour partie) les attaques de type XSS.

Les données entrantes peuvent être :

- Des données passées comme partie de l'URL
- Des données de formulaires, passées par GET
- Des données de formulaires passées par POST

Dans tous les cas, ces données sont aisément manipulables par l'utilisateur.

2.1.1 Données passées par l'URL

Toutes les données fournies sur l'URL doivent être contrôlées avant leur utilisation.

Par exemple, dans le cas d'une page avec l'URL `/mon_module/%mon_module_facture`, on s'attend à ce que `%mon_module_facture` soit une ID dans la base de données, c'est-à-dire un entier. Il est impératif de valider que c'est bien un entier avant de faire quoi que ce soit avec cette valeur.

Pour cela, utiliser le mécanisme d'auto-load de drupal, en définissant la fonction `mon_module_facture_load($ob_id)`. Cette fonction devra vérifier que l'argument dispose bien du format attendu et ensuite charger effectivement l'objet.

N.B : le nom de la variable doit être préfixé par le nom du module.

Voir la documentation de hook_menu : http://api.drupal.org/api/drupal/modules--system--system.api.php/function/hook_menu/7

2.1.2 Formulaires

Il est impératif d'utiliser la Form API pour tous les formulaires. Il faut aussi toujours utiliser les fonctions de callback Drupal pour la validation (fonctions *_validate) et les actions (fonctions *_submit).

2.1.3 Permissions

Il faut toujours vérifier que l'utilisateur a bien la permission d'accéder à l'objet demandé.

Dans l'exemple précédent, l'URL `mon_module/%mon_module_facture` permet d'accéder à une facture en fonction de son ID. S'il n'y a pas de contrôle, n'importe quel utilisateur peut afficher toutes les factures de tous les utilisateurs.

Il faut donc vérifier les permissions systématiquement avant de faire quelque action que ce soit (affichage, mise à jour, etc.).

2.2 AFFICHAGE

Toutes les données affichées doivent être :

- Nettoyées
- Traductibles dans le cas de chaînes en dur

Pour nettoyer les chaînes, la liste des fonctions disponibles est ici : http://api.drupal.org/api/drupal/includes--common.inc/group/sanitization/7

Pour générer des liens, vous devez utiliser la fonction `l()`, quel que soit le lien à générer.

Dans le cas de chaînes de caractère en dur dans les modules et les thèmes, vous devez utiliser la fonction `t()`, avec des variables de substitution. Les variables de substitution doivent commencer par « @ » ou « % » s'il faut mettre en valeur la variable.

Par exemple (`$title` contient du texte issu de la base de données) :

```
// Mauvais : ce n'est pas une chaîne en dur
// et $title est affiché sans nettoyage
print t("title is:" . $title);

// Mauvais : $title est affiché sans nettoyage
print t("title is: ") . $title;

// Correct : l'affichage peut être traduit,
// $title est nettoyé, empêchant le XSS
// et la destruction de la mise en page
print t("title is: @title", array("@title" => $title));
```

2.3 PERFORMANCES

Les performances doivent être prises en compte dès le début du projet. En particulier :

- Les vues doivent être paramétrées avec une limite sur le nombre de résultats. Elles ne doivent remonter que le nombre de résultats effectivement affichés ;
- Les requêtes sur la base de données doivent avoir une limite ;
- Aucun filtrage des résultats ne doit être fait dans le code ou les templates.

Plutôt que de charger les entités (nœuds, etc.) avec une requête manuelle sur la base, il faut utiliser les fonctions de chargement (`node_load`, etc.) qui permettent de bénéficier des mécanismes de cache de Drupal.

Afin de détecter d'éventuels problèmes, l'intégration doit être faite avec une base de données représentative des données de production, avec un volume de données proche de celui attendu.

2.4 TRADUCTIONS

Les traductions doivent être exportées dans des fichiers .po et importées à l'installation ou à la mise à jour. Le socle dispose d'un mécanisme permettant de réaliser cette opération automatiquement. Les fichiers .po doivent se trouver dans `<racine du projet>/translations/<groupe de chaines>/<code langue>.po`

Par exemple, les traductions pour le groupe « Interface utilisateur » en français se trouveront dans `<racine du projet>/translations/default/fr.po`.

3 DEPLOIEMENT

3.1 METHODE DE DEPLOIEMENT

Le déploiement du projet doit se baser sur les technologies suivantes :

- **Drush** pour l'exécution de commandes ;
- **Features** et **Strongarm** pour la gestion de la configuration ;
- **Migrate** pour l'initialisation des données.

3.2 INSTALLATION

Le projet doit comporter un module `master` qui active la toute la configuration et les modules nécessaires. Les commandes à exécuter pour l'installation doivent être programmées dans l'installeur `installer/install.php`. En particulier, le positionnement des variables doit être fait avec `drush vset`. Les variables qui sont modifiables par le client doivent être configurées dans le fichier de configuration externe.

L'installation doit se faire sans intervention sur le back-office.

3.3 MISE A JOUR

L'activation du module master suivi d'un revert des features doit déployer la nouvelle configuration. Les commandes à exécuter pour la mise à jour doivent être programmées dans l'installeur `installer/install.php`.

L'installation doit se faire sans intervention sur le back-office.

3.4 INITIALISATION DES DONNEES

Les données doivent être initialisées par Migrate. Les commandes nécessaires à l'import des données lors de l'installation doivent être programmées dans l'installeur `installer/install.php`.

L'initialisation des données doit se faire sans intervention sur le back-office.

4 THEMING

4.1 STYLE

Le style doit suivre les consignes générales de Drupal : http://drupal.org/node/1965

4.2 METHODE

- Partir de la maquette HTML
- Créer un thème vierge
- Extraire les CSS et JavaScript pour qu'ils soient gérés par Drupal
- Découper les pages de la maquette en région
  - Ne pas hésiter à créer des régions supplémentaires en cours de route
- Remplacer le contenu exemple par les variables Drupal

4.3 MAQUETTE

La maquette HTML doit être sous SVN. Elle doit respecter les règles suivantes :

- Tous les fichiers CSS, images et JavaScript doivent être dans un répertoire `assets`
- Les classes CSS doivent être préfixées par « jv- »

Le répertoire `assets` doit être placé dans le thème Drupal, avec une référence `svn:externals` dans la maquette.

5 DEVELOPPEMENT

5.1 STYLE A SUIVRE

Le standard de codage à respecter est détaillé ici : http://drupal.org/coding-standards

Ces règles de codage s'appliquent aussi aux CSS et JavaScript.

Une attention particulière doit être apportée aux conventions de nommage. Il est impératif de préfixer les noms des fonctions et des variables persistantes par le nom du module. Les noms des fonctions privées du module doivent commencer par un underscore « _ » suivi du nom du module.

5.2 ACCES AUX DONNEES

Les accès aux données doivent utiliser la DB API : http://drupal.org/developing/api/database

Toutes les requêtes modifiant des données (`INSERT`, `UPDATE`, `DELETE`) doivent utiliser les requêtes dynamiques ; les `SELECT` doivent de préférence utiliser des requêtes dynamiques.

Toutes les requêtes, même les plus simples, doivent utiliser des variables de remplacement :

```
//Mauvais
```

```
db_query("SELECT n.nid, n.title FROM {node} n WHERE
n.type = '$type'");


//Correct
db_query("SELECT nid, title FROM {node} WHERE type =
:type", array(
  ':type' => 'page',
));
```

Pour les requêtes remontant des listes de résultats, il faut utiliser les requêtes dynamiques et le mécanisme de modification des requêtes. Ceci permet de s'assurer que les restrictions de permissions seront bien prises en compte par Drupal. Cela permet d'exclure lors de la requête les nœuds auquel l'utilisateur n'a pas accès.

En particulier :

```
// Requête accédant à des nœuds
$query->addTag('node_access');

// Requête accédant à des termes de taxonomie
$query->addTag('term_access');
```

Guide de développement Drupal

J/JBC/167/DO/Drupal

Version : 1.02.01	Nouveauté / ~~Evolution~~	**Version remplacée :**	1.02.00

Modification(s) du document

Page/Paragraphe	Eléments modifiés

Relecture

Page/Paragraphe	Remarques

Relu par :	Date :	Visa :	Relecture après correction :
			oui ☐ non ☐

A RETOURNER A JBC AVANT LE 26/01/2012