Normes de développement DRUPAL

Author: Perroud Remi (DSE)
Date: 23 sept. 2015 08:35
URL: https://prod.etat-ge.ch/wikiadm/pages/viewpage.action?pageId=735085019

1 Généralités

#	Description	Références
GEN-1	Tout développement doit utiliser le master et l'un des profil qui a été fourni avec (internet ou intranet). Les modules à utiliser sont uniquement ceux du master. Si de nouveaux modules doivent être ajoutés, une demande doit être faite au chef de projet DGSI	
GEN-2	Le code source du noyau Drupal ne doit pas être modifié. Le fichier .htaccess doit correspondre à celui fourni avec le master. Toutefois, si une modification du noyau s'avère vraiment indispensable, un fichier patch accompagné d'une notice explicative peut être soumis au chef de projet DGSI pour validation et son éventuelle intégration au master.	
GEN-3	Le code source des modules contrib ne doit pas être modifié. Au besoin soumettre des patches sur drupal.org pour remonter les problèmes ou évolutions souhaitées à l'auteur du module et en informer la DGSI avec une notice explicative.	
GEN-4	Le projet doit s'adapter à la configuration serveur spécifiée par la DGSI (c.f. Dossier Infrastructure en référence). Si un paramétrage spécifique est nécessaire, par exemple ajout d'une extension PHP, ceci doit être discuté et validé par la DGSI. L'arborescence (répertoires, liens symboliques..) doit être la même que celle fourni avec le master. Le projet doit notamment tenir compte de : • la configuration du proxy dans le fichier settings.php pour gérer les appels http B2B. • la différence entre les urls front et bacck (cmsadmin.prod.etat-ge.ch et ge.ch) • la non fourniture du protocole de communication avec le client par le reverse proxy. (voir tableau de correspondance dans sites.php)	• https://prod.etat-ge.ch/wikiadm/x/w4DUD • phpinfo
GEN-5	Toute fonctionnalité doit être codée via une feature. Le nom de la feature, et du "projet" dans la feature, doivent contenir le terme "intranet" ou "internet" si il s'agit d'une fonctionnalité spécifique au profil intranet ou internet. Le module edg_installation, activé par défaut dans le master, fourni une liste de fonctionnalité (API) utilisable par les features pour facilité le développement	formation_technique.pptx

#	Description	Références
GEN-6	Tout site internet doit respecter la charte éditoriale de l'Etat de Genève et les normes d'accessibilité. Pour les sites Intranet, il faut rester dans la charte du thème "bestmobil**e"**	• Thème intranet : bestmobile original (démo) • Thème internet : Charte internet État
GEN-7	Tout développement doit être compatible avec les navigateurs suivants : • Firefox dernière version et avant dernière version + version 31ESR (version packagée pour l'Etat de Genève) • Internet Explorer 9 et + • Edge • Safari dernière version et avant dernière version • Chrome dernière version et avant dernière version Et fonctionner sur les systèmes d'exploitations les plus répandus : Windows, iOS, Macintosh, Android, Windows Phone	

2 Normes de codage

#	Description	Références
COD-1	Respecter les standards de codage Drupal : • Les fichiers doivent être au format UTF8 • Les commentaires ne doivent pas contenir d'accent. • Les retours à la ligne doivent être au format UNIX ("\n") • L'indentation se fait par 2 espaces	http://drupal.org/coding-standards
COD-2	Respecter l'architecture en couches : (voir l'architecture dans le document de formation) • aspect présentation sous la responsabilité des thèmes • aspect "métier" sous la responsabilité des modules/features Les thèmes notamment ne doivent pas faire d'accès base de données.	formation_technique.pptx
COD-3	Ne pas coder ce qui peut être fait par un module existant. Exemple : utiliser watchdog pour loguer et ne pas ré-écrire un système de log spécifique au projet.	https://www.drupal.org/node/299202
COD-4	Conserver un code lisible et bien découpé.	https://www.ufm.edu/images/0/04/Clean_Code.pdf
COD-5	Respecter les conventions de nommage Drupal.	https://www.drupal.org/node/299070
COD-6	Mettre en place des tests automatisés sur les développements complexes (complexité cyclomatique > 30).	http://drupal.org/simpletest
COD-7	Ne pas copier/coller du code et factoriser dès que possible	
COD-8	Commenter systématiquement : • chaque module custom ou feature doit contenir : 1 fichier readme.md et des commentaires dans le .info • chaque fichier : entête phpdoc avec le tag @file • chaque fonction : entête phpdoc avec les paramètres et résultat. • chaque implémentation de hook : entête phpoc avec commentaire "Implements hook_xxxx()" • tout code complexe qui mérite une explication	

#	Description	Références
COD-9	Lors de la modification d'un module, d'une feature, ou d'un thème existant le numéro de version ne doit pas être modifié. • Si il s'agit d'une modification concernant le répertoire "all" le numéro de version sera mis à jour par la DGSI au moment du merge DEV /SCD • Lors de la modification d'un module, d'une feature, ou d'un thème existant dans le répertoire "**site_[monsite]**" le numéro de version ne doit pas être modifier. Le numéro de version sera mis à jour par la DGSI juste avant le déploiement en recette	
COD-10	Lors de la création d'un hook_update dans un fichier .install il faut préciser en commentaire "migration de 7.x-Y.Y vers nouvelle version" ou Y.Y est le numéro de version actuel dans le .info. Selon COD-9 la nouvelle version n'est pas encore connue a ce moment là. Il ne faut qu'un seul hook_update par module par lot livré (numéro de version).	
COD-11	Le numéro de version de l'application/site est défini par le lead technique de la DGSI (DEV) ainsi que les numéro de version des modules : • Lors de la procédure de mise en recette pour les projets standard (target 6 de release.xml) • Lors du packaging pour SCD lorsqu'il s'agit d'une évolution du master (fichier release/release.properties) COD-9 est donc réalisé juste avant COD-11	

3 Portabilité

#	Description	Références

POR-1	Le projet doit pouvoir être mis en place sur un autre environnement (pour autant qu'il respecte la configuration requise) et ne doit pas être dépendant d'une ressource spécifique "hardcodée" : chemin de filesystem, URL absolue, identifiant de contenu... sauf dans le répertoire config.
	
	L'interface d'administration étant accessible avec une URL différente du front-office, TOUTES LES URL DOIVENT ÊTRE RELATIVE. Dans le cas particulier des feeds, RSS, xmlsitemap et des webservices qui doivent contenir des URL absolues, il faut utiliser la variable $public_base_url .
	
	=> Si nécessaire, utiliser le fichier settings.php pour définir certaines variables de configuration en les commentant et décrire cela dans un document de livraison
	
	Dans la colonne référence sont présenté les variables déjà définis et utilisables.
		`global $base_root = protocol://hostname`
		`global $base_path = /alias`
		`global $base_url = $base_root.$base_path`
		`global $public_base_root = protocol://hostname`
		`global $public_base_url = $public_base_root.$base_path`

POR-2	Les modules et les thèmes développés peuvent un jour être traduits en différentes langues. Pour les parties back-office les textes doivent être écrit en français sans utilisation de la fonction t(). Pour les textes front-office écrire le texte en français dans une fonction t()	http://drupal.org/node/299085
POR-3	Les menus de configuration, et d'administration doivent être créés dans admin /config/edg/[ma page]. Une permission spécifique (hook_permission) doit être créé pour chaque page de ce type. Au moins ADMIN-SITE et ADMINISTRATEUR doivent avoir cette permission.	

#	Description	Références
POR-4	Pour la compatibilité avec les anciens sites de l'état il faut utiliser : • edg_installation_get_theme() pour déterminer si il s'agit du thème intranet (EDG_INTRANET_THEME) ou internet (EDG_INTERNET_THEME). • edg_installation_get_profile() pour déterminer si il s'agit d'un site intranet (EDG_INTRANET) ou internet (EDG_INTERNET)	

4 Performance

#	Description	Références
PER-1	Éviter d'activer trop de modules : au delà de 120 il y a sans doute des questions à se poser. Éviter d'activer notamment des modules qui ne sont pas utilisés. Supprimer de l'arborescence les modules qui ne sont pas activés	
PER-2	Respecter les recommandations Drupal pour écrire du code performant. Veiller notamment à ce que le site ne génère pas des milliers de requêtes SQL par page.	http://drupal.org/node/328206

5 Sécurité

#	Description	Références
SEC-1	Respecter les préconisations de Drupal relatives à la sécurité.	http://drupal.org/writing-secure-code
SEC-2	Dans le contexte DGSI l'accès à l'administration du site doit être géré via Gina. En prestation externe un mode mock est disponible pour simulation ainsi qu'un module edg_masquerade.	

6 Les features

#	Description	Références

FEATURE-1	• Pour les liens pointant vers des vues le lien doit être créé dans la vue.
	• Pour les liens pointant vers des page custom de modules ils doivent être créé via le hook_menu
	• Pour les autres liens il ne faut pas utiliser le composant menu_link de feature car l'utilisateur doit pouvoir modifier le menu sans que cela entraine un état "supplanté" de la feature. Il faut donc utiliser dans le hook_install de la feature et la fonction menu_link_save ou edg_installation_queue_menu_link_save si la page vers lequel pointe le lien n'existe pas encore au moment de l'installation (voir ci joint).
		utilisation de menu_link_save
		`// lien à créer`
		`$item = array(`
		`'link_path' => 'lien_de_la_page',`
		`'router_path' => 'lien_de_la_page',`
		`'link_title' => 'Titre du lien dans le menu',`
		`'expanded' => 1,`
		`'customized' => 1,`
		`'module' => 'Nom_du_module,`
		`'menu_name' => 'main-menu',`
		`);`
		`$mlid = menu_link_save($item);`
		
		utilisation de edg_installation_menu_link_save
		`// lien a creer`
		`$item = array(`
		`'expanded' => 1,`
		`'link_path' => 'exemple',`
		`'menu_name' => 'main-menu',`
		`);`
		`// on doit attendre que le parent exemple0 existe avant de creer le lien`
		`$paths = array(`
		`'parent_path' => 'exemple0',`
		`);`
		`$menus = array(`
		`'parent_path' => 'main-menu',`
		`);`
		`edg_installation_queue_menu_link_save($paths, $menus, $item);`

#	Description	Références
FEATURE-2	Les éléments suivants ne doivent pas être embarqués dans les features : • Les menus fourni par le socle (edg_installation) • Les items de menu (voire FEATURE-1) • L'item de menu lié au menu navigation quand la feature embarque un content type. Le menu sera re-créé automatiquement • La variable "menu_parent_[content_type]" si elle dépend d'un menu défini par la feature elle même (voir REG_TP-4) • La variable "additional_settings__active_tab_[mon_content_type]" qui indique quel est l'onglet actif dans la page d'admin du content type. Aucun intérêt et introduit des états "supplanté"	
FEATURE-3	Pour les features **intranet** il faut appeler la fonction edg_installation_add_bestmobile_icon dans le hook_install pour définir les icônes applicable aux pages publique défini par la feature (view, node...).	

FEATURE-4	Pour les features **intranet**, tous les menus générés pour le menu principal doivent avoir "expended=1" donc il faudra utiliser edg_installation_queue_menu_link_save dans le hook_enable pour modifier les liens de menu créé par la feature.	**utilisation de edg_installation_menu_link_save**
		`// lien a modifier`
		`$item  = array('expanded' => 1);`
		`$paths = array('link_path' => 'exemple');`
		`$menus = array('link_path' => 'main-menu');`
		`edg_installation_queue_menu_link_save($paths, $menus, $item);`

FEATURE-5	Si une feature embarque des modules en dépendance qui créent des vues inutilisées, il faudra les désactivé avec edg_installation_deactivate_views	

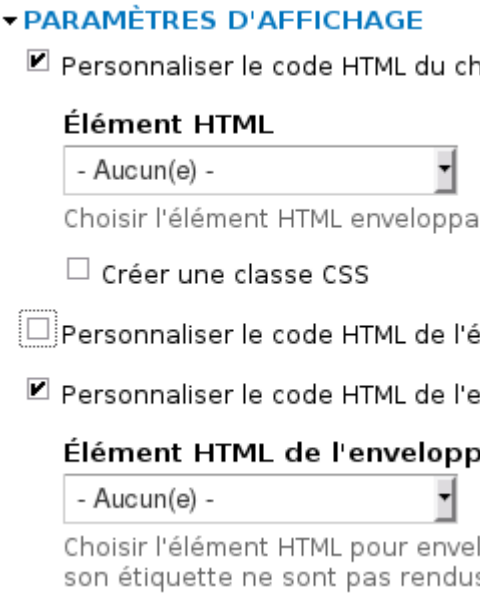
7 Communication entre features

#	Description	Références
COM-1	Chaque feature doit prendre en charge la communication avec les features transverses	Les features transverses au • edg_personnalisation_ • edg_activites_recente • edg_workflow
COM-2	Lors de la création d'une feature transverse, celle-ci doit embarquer les informations concernant les modification des vues et types de contenus définis par edg_installation et doit fournir un hook pour que les autres features puissent donner leurs instructions. Les hook défini ainsi ne contiendront pas le mot "edg" ni le mot "intranet" afin de faire plus court. Exemple correct : hook_personnalisation_instructions() => edg_actualites_intranet_personnalisation_instructions() Exemple incorrect : hook_edg_personnalisation_intranet_instructions() => edg_actualites_intranet_edg_personnalisation_intranet_instructions()	Exemple dans edg_personna • **edg_personnalisatio** hook_personnalisatio • **edg_personnalisatio** ce même hook pour e edg_installation_pers • **edg_actualites_intra** même hook pour son edg_actualites_intran
COM-3	Chaque nouveau hook créé dans une feature dois être déclaré et expliqué dans le fichier [ma_feature].api.php	

8 Les blocs

#	Description	Références
BLOC-1	Le positionnement d'un bloc dans une région ne doit pas être fait avec le module FE_block mais avec un appel à edg_installation_insert_or_update_blocs. FE_block est lent, peu fiable, et nécessite un patch : • Créer un fichier « nom_de_la_feature_X.block.inc » à la racine de la fonctionnalité • Inclure ce fichier (include_once) dans le hook_install de la fonctionnalité (nom_de_la_feature_X.install) • Utiliser l'api EDG_Installation afin de placer correctement les blocs (cf edg_installation.api.inc)	
BLOC-2	Toujours spécifier les conditions d'affichage d'un bloc si ce bloc ne s'affiche pas sur toutes les pages même si la région où il s'affiche n'existe pas sur toutes les pages.	
BLOC-3	Une fois qu'une région a été livré en production, il est interdit de modifier sont nom, car cela engendre de gros problème de mise à jour que l'on veut éviter	

9 Les vues

#	Description	Références
REG_VUE-1	Rendre statique les modèles de champs de vue pour faciliter la tâche au designer (views-fields-template.tpl.php) • Remplacer les foreach par la liste des champs de la vue dans les templates qui seront sous la responsabilité du designer.	
REG_VUE-2	Pas de HTML / CSS dans la configuration des vues	
REG_VUE-3	Les vues doivent utiliser un cache de type "rules_trigered_cache" sauf dans le cas ou il y a des filtres exposé ou l'utilisation d'ajax. Ne pas oublier d'activer le cache de bloc pour les vues blocs sauf si le bloc dépend de paramètre	
REG_VUE-4	Un bloc généré par une vue ne doit pas avoir de nom encrypté de type (exemple : bloc-AHSGZISNAGQODHZHQG5648). Ce cas ce produit lorsque le nom machine du bloc est trop long, qu'il contient des accents, ou qu un autre bloc d'une autre vue porte déjà le même nom.	

#	Description	Références
REG_VUE-5	intranet : Tous les filtres des vues ont été placés dans la colonne latérale. Afin de respecter cette convention, il est nécessaire de choisir l'option « Filtre exposé dans un bloc » à partir des vues possédants des filtres exposés.	
REG_VUE-6	Il faut toujours accompagné une vue d'une règle de clear cache automatique. Dans le cas standard (une vue dont le cache doit être clear lorsque l'on modifie des noeud d'un certain type) on utilisera edg_installation_queue_add_clean_view_rules dans le hook_enable. Dans le cas où cette rule n'est pas suffisante il faudra embarquer dans la feature des rules complémentaires.	
REG_VUE-7	Lorsque l'on utilise edg_installation_queue_add_clean_view_rules il faut également utiliser edg_installation_queue_remove_clean_view_rules dans le hook_disable	

10 Les types de contenus

#	Description	Références
REG_TP-1	Règle concernant les noms de champs de types de contenus, il doivent comporter : • Un nom court correspondant au type de contenu • Le nom du champs • Le type intra ou rien (pour inter ou commun)	Exemple pour la création d'un champ de type Image dans le type de contenu Galerie Photos en intranet : field_galerie_photos_images_int
REG_TP-2	Pour les champs **images** ne pas oublier de compléter : • La taille minimum autorisée • Cocher la case "alt" (accessibilité) • Donner un nom de répertoire correspondant à la feature • Un widget multi upload est a utilisé lorsqu'il s'agit d'un type de contenu embarquant beaucoup d'images (exemple feature galerie photo) sinon un widget type image configuré correctement avec epsa crop est a utiliser obligatoirement pour faciliter le travail du rédacteur.	voir feature actualites
REG_TP-3	Pour les champs **files** ne pas oublier de compléter : • Donner un nom de répertoire correspondant à la feature	voir feature publications

#	Description	Références
REG_TP-4	Un type de contenu doit être "rangé" sous la vue qui liste l'ensemble des contenus de ce type si elle existe, ou dans le menu où il doit apparaitre si il existe. soit via : • le module menu force si le contenu doit apparaître comme menu et un appel à "edg_installation_queue_set_menu_parent" (exemple : feature galeries photos) • la création de rules menu position via un appel à "edg_installation_queue_menu_position_add_rule_ct" si le contenu ne doit pas apparaître comme menu. Dans ce cas menu position permettra d'avoir un breadcrumb correct et un menu sélectionné correct lorsque l'on est sur la page du noeud. (exemple : feature actualités).	Exemple : il existe une vue "actualités" qui se trouve dans le menu "main-menu:actualités" et qui liste l'ensemble des actualité le type de contenu "actualites_intra" doit donc être rangé dans le menu : "main-men actualités/[node:title]"
REG_TP-5	Un type de contenu doit toujours avoir un pattern (motif) pathauto en correspondance avec la règle REG_TP-4 : • Si une vue existe avec un path X, les contenu devront avoir un pattern X/[node:title] (exemple : feature actualités) • Si un menu existe le pattern doit suivre les parents du menu [menu:parents]/[node:title] (exemple content type page de base) • Si il n'y a pas de rangement, le pattern devra se baser sur le nom du content type : [node:type]/[node: title] (exemple content type bloc)	
REG_TP-6	Concernant la visibilité des nœuds en tant que page (node /XXX) d'un type de contenu : • Soit les nœuds sont visibles : dans ce cas il faut ajouter dans le hook_install de la feature un appel à la fonction edg_installation_add_ct_to_linkit pour déclarer l'existence de ces pages à linkit (exemple : feature actualités) • Soit les noeuds ne sont pas sensés être visibles directement via une page propre (sous entendu, ils ne sont visibles qu'à travers des vues) : dans ce cas il faut ajouter une règle "page manager" pour rediriger l'affichage de ce type de noeud vers la vue les affichant	voir feature news_flash

#	Description	Références
REG_TP-7	Pour les champs **texte long, texte long avec résumé** ne pas oublier de compléter : • les format de texte utilisable (better format) : • la largeur : bloc, étroit, large (en général étroit ET large) • les fonctionnalités (images, lien, aucun des 2, les 2)	voir feature actualites

11 La taxonomie

#	Description	Références
TAX-1	Comme pour REG_TP-6 les vocabulaires qui ne sont pas accessibles directement doivent faire l'objet d'une règle "page manager" pour bloquer l'accès à la vue taxonomy/term pour ces vocabulaires	features faq
TAX-2	Afin d'éviter les conflits entre features, une feature n'a pas le droit d'utiliser la vue taxonomy/term par défaut. Elle doit utiliser sa propre vue avec sa propre url (lié a pathauto). Ceci peut être réalisé à l'aide du module page manager et page manager views	
TAX-3	Lorsque vous embarquez une règle page manager pour taxonomy/term, il faut ajouter la configuration de la variable "page_manager_term_view_disabled" dans le hook_install avec un variable set et nom pas via feature pour éviter les conflit	variable_set ("page_manager_term_view_disabled", FALSE);

12 Les permissions et droits d'accès

#	Description	Références
PERM-1	L'organisation des droits est défini dans le fichier "gestion des droits" accessible depuis la page d'accueil du Projet indus dans le wiki.	
PERM-2	edg_installation_user_roles() : est une fonction de l'API edg_installation équivalente à user_roles et qui permet de récupérer la liste des roles avec les noms techniques des rôles au lieu des noms traduits.	

13 Documentation

#	Description	Références
DOC-1	Chaque feature doit s'accompagner d'un fichier readme.md décrivant les composants de la fonctionnalité. Ce fichier doit être maintenu à jour au fur et à mesure des développements.	
DOC-2	Il doit être possible d'identifier et fournir les différentes releases (même anciennes) du projet ce qui implique l'utilisation d'un gestionnaire de sources. Bonnes pratiques pour les commits (en interne) : • Toujours commiter du code stable • Indiquer le numéro de demande dans le message : ticket JIRA ou identifiant de fonctionnalité (ceci facilite notamment les revues de code) • Les fichiers impactés par le commit ne doivent concerner que la demande traitée. Il est possible de faire plusieurs commits pour une même demande • Ne pas commiter les fichiers "personnels" comme certains fichiers de config propres à l'environnement local, cela facilitera le travail des collègues qui vont mettre à jour • Eviter de commiter des modifications d'espace ou de sauts de ligne qui polluent l'historique • Utiliser crucible pour les revues de code. A noter qu'il est possible de créer automatiquement des revues en rajoutant des mots-clés dans le message de commit, exemple svn commit -m "CMS-XXX correction du bug X +review DRUPAL @perroudr " => Ceci créera automatiquement une revue, dans le projet DRUPAL, avec comme reviewer perroudr	https://www.drupal.org/node/299067
DOC-3	Lorsqu'une feature déclare un nouveau hook un exemple doit être créer dans le fichier [ma_feature].api.php comme l'exige les bonnes pratiques Drupal	