Initiation a Drupal

slides

Les slides de la présentation sont disponible à cet url

bit.ly/drupalini

initiation-a-drupal.pdf

Agenda

- Drush
- Module et hook
- API
- theme
- configuration

Drush

Drupal shell

Drush c'est le drupal shell, c'est l'outil en ligne de commande de drupal.

Documentation | drush.org

Project manager

Drush contiens un projet manager, qui peut télécharger et mettre à jours le core de drupal, les thèmes et les modules.

```
$ cd ~/Sites
$ drush pm-download
Project drupal (7.34) downloaded to /Users/dl/Sites/drupal-7.34
```

Installation

Drush permet d'installer drupal, ce qui créé ou écrase la base de données et les fichier de configuration.

"'bash $ cd /Users/dl/Sites/drupal-7.34 $ drush site-install standard my-template-name.tpl.php

```
  <h1><?php print $title ?></h1>
```

“‘

Mise à jours

On peut faire les mises à jour du code des modules et du core ainsi que la base de données par drush.

```
$ drush pm-updatecode
No code updates available.

$ drush updatedb
No database updates required
```

Variables

Drush permet d'inspecter et de modifier les variables de drupal.

```
$ drush variable-get name

$ drush variable-set site_name new-name
site_name was set to "new-name".
```

script

On peut utiliser drush pour executer des scripts php ce qui est pratique pour explorer l'api.

```
$ echo "<?php var_dump(menu_tree_page_data('main-menu'));" \
  > test.php
$ drush php-script test.php
 array(1) {
   '50000 Home 218' =>
   [...]
```

Gestion des modules

On peut aussi gérer les téléchargements, installations et mise à jour des modules

Avec la commande suivante on télécharge le projet example de drupal. Ce sont des exemples de module pour apprendre à travailler avec les api backend de drupal.

```
$ drush pm-download example
Project examples (7.x-1.x-dev) downloaded to sites/all/modules/examples.
```

Examples for Developers | drupal.org

block_example

Nous allons étudier le module 'block_example' qui démontre comment créer un block custom.

Au minimum un module contiens un fichier de configuration block_example.info et un fichier avec le code block_example.module.

Le fichier block_example.install contiens le script d'installation et le block_example.test contiens les tests d'intégration drupal avec Simple Test.

```
$ cd sites/all/modules/examples/block_example
$ ls
block_example.info
block_example.install
block_example.module
block_example.test
```

block_example.info

Le fichier info contiens l'information du module qui seront affiché dans l'interface d'administration ou dans la commande `drush pm-info`.

```
name = Block Example
description = An example outlining how a module can define blocks.
package = Example modules
core = 7.x
```

HOOK_block_info

Dans le fichier .module on définie des hook pour que drupal puisse appeler notre code. Les hook sont des fonction qui débute avec le nom du module.

Par exemple pour implémenter hook_block_info on définie une fonction avec le nom block_example_block_info().

```
/**
 * Implements hook_block_info().
 *
```

```
 * This hook declares what blocks are provided by the module.
 */
 function block_example_block_info() {
```

HOOK_block_info

Le hook block_info on retourne les informations de notre block à drupal.

```
  $blocks['example_uppercase'] = array(
    'info' => t('Example: uppercase this please'),
    'status' => TRUE,
    'region' => 'sidebar_first',
  );
  return $blocks;
}
```

documentation

Les valeurs possible de l'array de block est documenté sur le site de drupal.org

Et on as la listes des hook disponibles sur le lien suivant.

hook_block_info | drupal.org

hooks | drupal.org

example.local/admin/structure/block Lorsque drupal as les informations du block, on peut l'assigner à une région dans l'interface administrateur.

Figure 1:

hook_block_view

Le hook block view est appelé par drupal avant d'afficher les block et le machine name du block est passé en argument.

```
function block_example_block_view($delta = '') {
  switch ($delta) {
    case 'example_configurable_text':
```

example_empty

On as simplement à ajouter la string qu'on veux afficher comme valeur à la clé 'content' pour qu'elle s'affiche dans le block.

```
    case 'example_empty':
      $block['subject'] = t('Title of second block');
      $block['content'] = block_example_contents($delta);
    break;
```

hook_menu

Les routes dans drupal sont déclaré en utilisant le hook menu. Ça porte à confusion au début mais pour les développeur drupal un menu ce n'est pas un menu c'est une route, le path pour accéder à une page custom.

Nous allons regarder les exemples fournis par drupal.

```
$cd ~/Sites/drupal-7.34/sites/all/modules/examples/menu_example

$ls
menu_example.info   menu_example.module menu_example.test
```

menu_example.module

Dans le fichier menu_example.module on définit le hook_menu que nous allons utiliser pour retourner les informations sur nos routes à drupal.

```
/**
 * Implements hook_menu().
 *
 * A simple example which defines a page callback and a menu entry.
 */
function menu_example_menu() {
```

example.local/examples/menu_example

Si on veux ajouter la route examples/menu_example à notre projet on ajoute la route comme clé à l'array que retourne le hook on déclare une fonction callback et une validation d'accès.

En retournant 'TRUE' on s'assure que tout les rôles ont accès à notre page.

```php
function menu_example_menu() {
 $items['examples/menu_example'] = array(
   'page callback' => '_menu_example_basic_instructions',
    'access callback' => TRUE,
    );
  return $items;
}
```

Callback

La fonction callback définie dans le hook_menu retourne le markup à afficher sur dans le contenu de la route.

Voici la documentation du hook menu et celle du hook_i18n_translate_path qui est pratique pour associer les routes des différentes langues pour que drupal puisse construire le language switcher.

```php
function _menu_example_basic_instructions($content = NULL) {
  $base_content = t(
  'This is the base page of the Menu Example. There are a number of examples
  here, from the most basic (like this one) to extravagant mappings of loaded
  placeholder arguments. Enjoy!');
  return '<div>' . $base_content . '</div><br /><div>' . $content . '</div>';
```

Documentation hook_menu

hook_menu | drupal.org

hook_i18n_translate_path | drupalcontrib.org

hook_theme

my_example.module

Le module theming_example.module démontre différent exemples d'utilisation du hook_theme. Souvent on l'utilise simplement pour créer un template custom dans un module.

On n'as qu'à retourner la clé de notre thème et le nom du fichier template au hook.

```php
function my_example_theme() {
  return [
    'my_theme_key' => [
```

```
      'template' => 'my-template-name'
    ]
  ];
}
```

template files

Dans notre exemple on déclare un fichier template my-template-name.tpl.php

```
  <h1><?php print $title ?></h1>
```

theme($key, $variables)

Puis lorsqu'on utilise cette clé dans la fonction thème ça nous retourne une chaine de caractère contenant le markup du fichier template my-template-name.tpl.php

Par exemple ici on utilise notre template dans un callback de menu pour l'afficher sur une page.

```
  function _menu_callback() {
    return theme('my_theme_key',['title'=> "variable value"]);
  }
```

Le deuxième argument de la fonciton thème contiens les variables du template et les clés seront les noms des variables.

hook_preprocess_hook

html.tpl.php

En déclarant le hook_preprocess_hook un modules peut modifier les variables que drupal passe à un template.

Comme premier exemple le html.tpl.php est le premier template de drupal, il contiens par exemple la balise

ou on aimerais ajouter l'id de Google Analytics

```
function my_example_preprocess_html( &$variables ) {
    $variables['googe_analytics_id'] = "1234543";
  }
```

page.tpl.php

Le template page.tpl.php contiens le layout général du site, les régions, les menus, le header, le footer. Dans cette exemple on définie une nouvelle variable $header qui contiens le markup de notre fonction _get_custom_header()

```
function my_example_preprocess_page( &$variables ) {
    $variables['header'] = _get_custom_header();
  }
```

On as aussi un preprocess pour les templates de nodes et on as le type de contenu dans l'array de variable pour agir sur les nodes d'un type de contenu spécifique.

```
function my_example_preprocess_node( &$var ) {
  if ($var['type'] == 'my_content_type'){
    $var['my_variables'] = _my_processing($var['node']);
  }
}
```

Variable

Drupal as un array de variable "persistante" qui est enregistré dans la base de donnés.

On peut utiliser la fonciton variable_get() avec la clé de la variable pour obtenir sa valeur.

```
  $variables['googe_analytics_id'] = variable_get('my_ga_id');
```

variable.module

il y as un module nommé variable qui permet de déclarer des page de configuration dans l'interface administrateur de drupal pour les variables.

```
$drush dl variable
Project variable (7.x-2.5) downloaded to sites/all/modules/variable.
Project variable contains 6 modules: variable_views, variable_store, variable_realm, variabl
$ cd ~/sites/drupal-7.34/sites/all/modules/variable/variable_example
```

hook_variable_info

Pour obtenir une page de configuration pour notre variable on premièrement le hook_variable_info pour configurer les métadonnées de notre variable et on l'associe à un groupe de variable.

```
//variable_example.variable.inc
function variable_example_variable_info($options) {
 $variables['variable_example_text'] = array(
    'type' => 'text',
    'title' => t('Simple text', array(), $options),
#   [...]
   'group' => 'variable_example',
```

hook_variable_group_info

Ensuite on déclare les informations du groupe de variable qu'on vas afficher sur notre page de configuration et on l'associe à un path.

```
function variable_example_variable_group_info() {
 $groups['variable_example'] = array(
    'title' => t('Examples'),
    'description' => t('Variable examples of different types.'),
    'access' => 'administer site configuration',
   'path' => array('admin/config/system/variable/example'),
```

Finalement on déclare la route custom pour la page d'administration de notre group de variables.

On ajoute le callback drupal_get_form et le groupe de notre variable aux arguments de cette route.

```
//variable_example.module
function variable_example_menu() {
    'path' => array('admin/config/system/variable/example'),
    'title' => 'Variable example',
    'description' => 'Example of auto generated settings form.',
   'page callback' => 'drupal_get_form',
   'page arguments' => array('variable_group_form', 'variable_example'),
    'access arguments' => array('administer site configuration'),
  );
```

EntityFieldQuery

Les nodes et les users sont des entitées et l'objet EntityFieldQuery permet de faire des requêtes qui retourne ces entités. ### node de type custom_type

```
$query = new \EntityFieldQuery();
$query->entityCondition('entity_type', 'node')
  ->entityCondition('bundle','custom_type')
  ->propertyCondition('status', 1)
  ->propertyCondition('language', 'fr');
```

node_load

```
$queryResult = $query->execute();
$nodeArray = isset($queryResult['node'])? $queryResult['node']:array();
$nodesId = array_keys($nodeArray);
$nodes=  node_load_multiple($nodesId);
```

documentation

How to use EntityFieldQuery | drupal.org

Entity Metadata

Le entity metadata wrapper permet de faciliter la manipulation des nodes en php.

Sans le Entity metadata accéder aux valeurs des field est un peu complexe. Il faut utiliser la clé de la langue et celle de l'index, même si c'est un champ à valeur unique dans un type de contenu non traduit.

```
$value = $node->field_number[LANGUAGE_NONE][0]['value']
```

getter

Lorsqu'on ajoute le wrapper à une node on peut simplement appeler le getter sur le field et le wrapper d'occupe de la langue courante et de nous retourner une valeur unique ou un array selon le cas.

```
$wrapper = entity_metadata_wrapper('node', $node);
$value = $wrapper->field_number->value();
```

setter

On as aussi accès à un setter.

```
$node_wrapper->field_number->set(1);
```

property information

Le wrapper ne contiens aucunes valeurs, pour avoir la liste des fields et autres propriétés d'une entity on peux utiliser la méthode getPropertyInfo

```
var_dump($wrapper->getPropertyInfo());
```

theme

Mothership

Drupal est à la base un projet porté par une communauté de développeur backend. On entend rarement des développeurs front end vanter les bon coté du système de template de drupal et les critiques sont plus que fréquentes.

L'ambition de mothership est d'offrir des solutions à certaines de ces critiques.

> "If you really like the markup & CSS options that Drupal provides - this theme is probably not for you" -Mothership

obtenir mothership

On peut utiliser le project manager de drupal pour obtenir, et installer un thème.

```
$ drush dl mothership
Project mothership (7.x-2.10) downloaded to sites/all/themes/mothership.
$ drush en mothership
The following extensions will be enabled: mothership
Do you really want to continue? (y/n): y
mothership was enabled successfully.
$ drush cc all
```

Sous thème

Une fois activé Mothership offre une commande drush qui permet de créer un sous thème.

```
$ drush |grep  mothership
Other commands: (make,mothership)
 mothership            Create a mothership sub-theme.
$ drush mothership test
 Mothership subtheme "test" created in: /Users/dl/Sites/drupal-7.34/sites/all/themes/test
 Visit your themes settings page and configure it to your liking: http://default/admin/appea
```

Override Core template

Les templates sont premièrement défini dans les modules, ils sont overridé si un fichier avec le même nom se trouve à l'intérieur du folder du thème.

Par exemple ici on copie le template par défaut de page dans notre thème.

```
$ cd ~/Sites/drupal-7.34/sites/all/themes/test
$ cp ~/Sites/drupal-7.34/modules/system/page.tpl.php .
```

Template mothership

Mothership override la plus part des templates du core de drupal, on peut se baser sur ceux ci pour notre override.

Drupal commence par prendre le fichier du thème actif, s'il n'est pas défini, il prend celui défini dans le thème parent et finalement il prend celui par défaut défini dans le module.

```
$ cp ../mothership/mothership/templates/page.tpl.php .
```

templates folder

Drupal cherche les templates dans le thème de façons récursive, on peut donc avoir une hiéarchie de dossier pour organiser ces templates.

```
$ mkdir templates
$ mv page.tpl.php templates
```

node

On peut déclarer des template différent pour chaque type de node. Il suffit d'ajouter le machine name du type de contenu après la node.

```
$ cp ../mothership/mothership/templates/node.tpl.php \
```

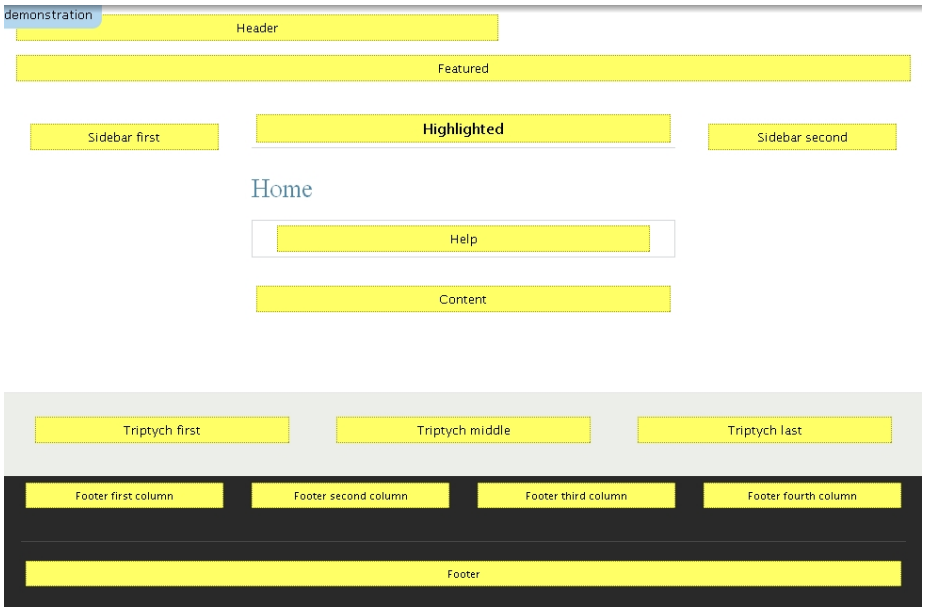

Figure 2:

Drupal définie des régions dans les quels on peut afficher les blocks. Ici on voie les région par défaut.

region

On peux définir une région custom en ajoutant le machine name dans le fichier de configuration de notre thème test.info et en appelant la fonction render() sur la région qui est ajouté aux variables $page à l'intérieur du page.tpl.php

```
;test.info
regions[header] = Header

//page.tpl.php
<?php print render($page['header']); ?>
```

Création d'un type de content

Utilisation du module field collection

Configuration drupal par script automatisés

```
 fichier.install
hook_update_n
```

gestion des variables et des modules ## gestion des roles et permissions ## Export de la configuration par features ## Module d'intégration

Fonction des différentes branches/environnement

Gestions des librairies et autoloader (composer)

Exemples de code et test automatisé (phpspec)

Traitement des variables des éléments de conteu

```
Drupal adapter
```

Arrays functions Fields functions Menu