
myplanet

Coding for Config

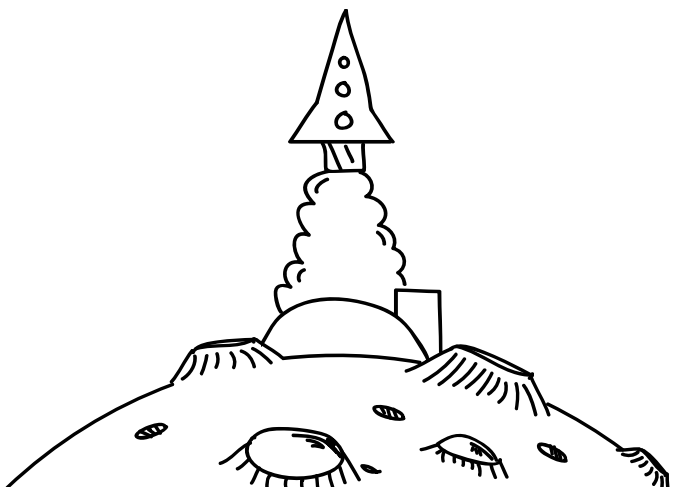
Install Profile Development Using Features

What is an install profile?

Distributions provide site features and functions for a specific type of site as a single download containing Drupal core, contributed modules, themes, and pre-defined configuration.

They make it possible to quickly set up a complex, use-specific site in fewer steps than if installing and configuring elements individually.

Distributions vs. Install Profiles



myplanet

Distribution Requirements

- Only GPL-compatible libraries
- No cloned sandboxes
- No includes[] directive
- Only d.o modules and themes
- Must use drupal-org-core.make
- Must define git branches

myplanet

Install Profile Requirements

- Provide site features and functions for a specific type of site as a single download

Hard and Soft Configuration

Hard config

ery Field

Show row weights

Label	Machine name	Field type	Widget	Operations
Title	title	Node module element		
Body field	body	Long text and summary	Text area with a summary	edit delete
Upload a file	every_field_file	File	File	edit delete
Upload an image	every_field_image	Image	Image	edit delete
Boolean	every_field_list_boolean	Boolean	Single on/off checkbox	edit delete
List (integer)	every_field_list_integer	List (integer)	Select list	edit delete
List (float)	every_field_list_float	List (float)	Check boxes/radio buttons	edit delete
List (text)	every_field_list_text	List (text)	Select list	edit delete
Integer	every_field_integer	Integer	Text field	edit delete
Decimal	every_field_decimal	Decimal	Text field	edit delete
Float	every_field_float	Float	Text field	edit delete
Tags	every_field_term_reference	Term reference	Autocomplete term widget (tagging)	edit delete
Text	every_field_text	Text	Text field	edit delete
Text, long	every_field_text_long	Long text	Text area (multiple rows)	edit delete
Add new field				

Label

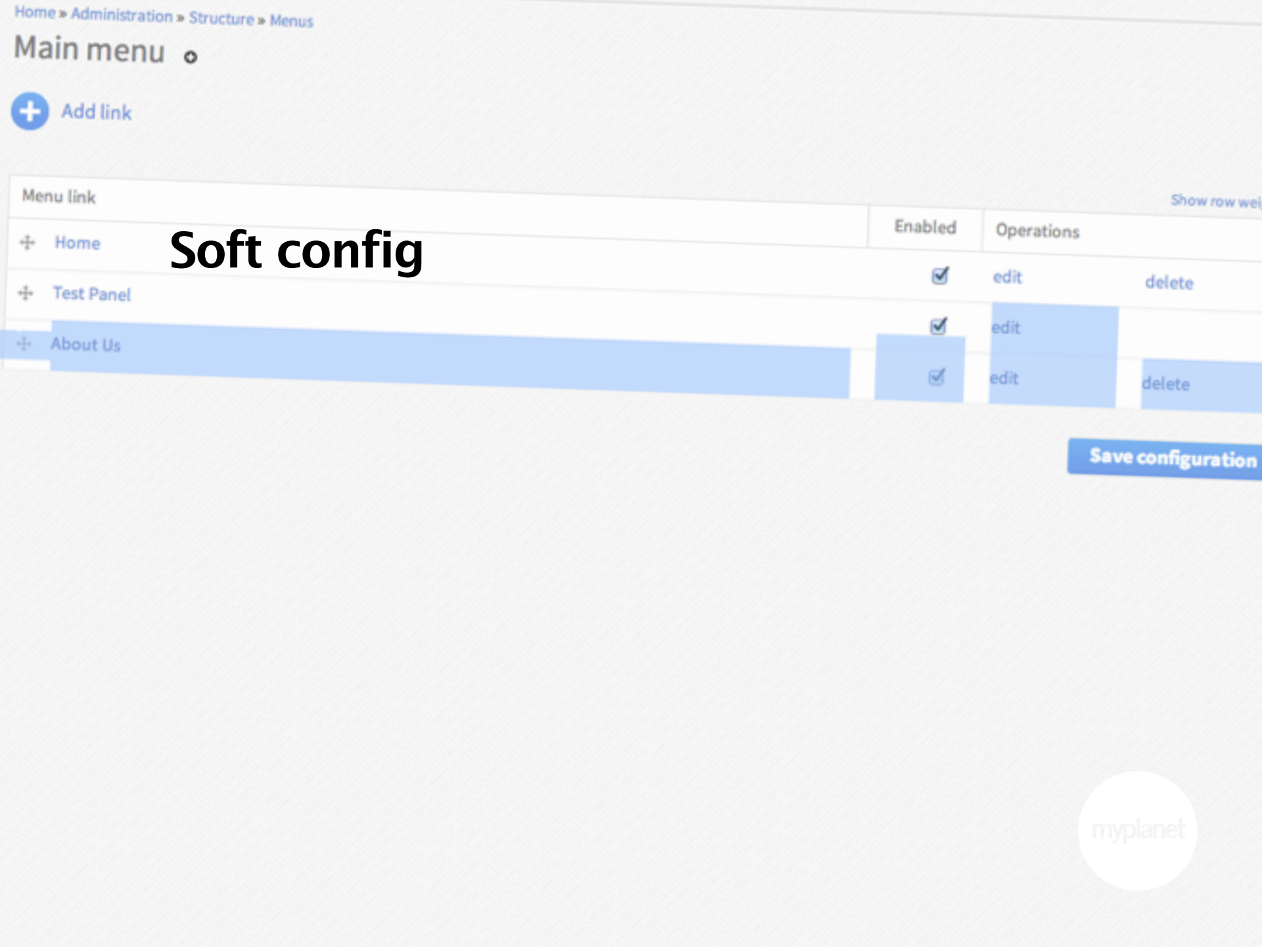
- Select a field type -

Type of data to store.

- Select a widget -

Form element to edit the data.

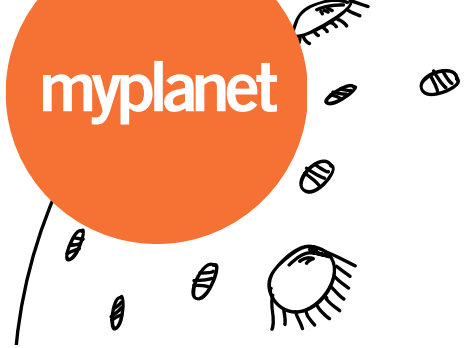
It has a machine name. It is easy to export.


Home » Administration » Structure » Menus
Main menu
Add link
Menu link
Enabled
Operations
Home
Test Panel
edit
delete
About Us
edit
edit
delete
Save configuration

Soft config

It has a numeric ID. It is hard to export.

When? Where? Why?



How do you make an install profile?

You'll need...

```
[profile].info
[profile].profile
[profile].install
```

```
build-[profile].make
drupal-org.make
drupal-org-core.make
```

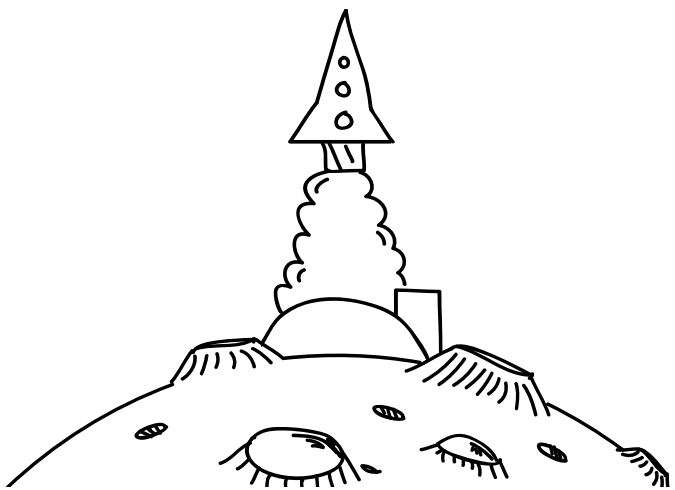
```
modules/
   contrib/
   custom/
   features/
```

```
themes/
  contrib/
  custom/
```

```
tmp/
   docs/
   tests/
   snippets/
   scripts/
    build-dev.sh
    build-prod.sh
```

```
[profile].info
[profile].profile
[profile].install

build-[profile].make
drupal-org.make
drupal-org-core.make

modules/
  custom/
  features/
  contrib/

libraries/

themes/
  custom/
  contrib/

tmp/
  docs/
  tests/
  snippets/
  scripts/
    build-dev.sh
    build-prod.sh
```

File contents



myplanet





SPARK



myplanet

```
[profile].info
[profile].profile
[profile].install
```

```
name = Spark
description = Cutting-edge authoring experience improvements for
Drupal.
version = "7.x-1.0-alpha5"
core = 7.x
php = 5.3

; The distribution_name property is used in the installer and other
places as
; a label for the software being installed.
distribution_name = Spark

; Required modules (Core)
; Copy/pasted from standard.info
dependencies[] = block
dependencies[] = color
dependencies[] = comment
dependencies[] = contextual
dependencies[] = dblog
dependencies[] = field_ui
dependencies[] = file
dependencies[] = help
dependencies[] = image
dependencies[] = list
dependencies[] = menu
dependencies[] = number
```

```
<?php
/**
 * @file
 * Enables modules and site configuration for a standard site
installation.
 */

/**
 * Implements hook_form_FORM_ID_alter() for install_configure_form().
 *
 * Allows the profile to alter the site configuration form.
 */
function spark_form_install_configure_form_alter(&$form, $form_state) {
  // Pre-populate the site name with the server name.
  $form['site_information']['site_name']['#default_value'] =
$_SERVER['SERVER_NAME'];
}
```

```
<?php
/**
 * @file
 * Install, update and uninstall functions for the spark install
profile.
 */

/**
 * Implements hook_install().
 *
 * Perform actions to set up the site for this profile.
 *
 * @see system_install()
 */
function spark_install() {
  ...
}
```

```php
<?php
/**
 * @file
 * Install, update and uninstall functions for the spark install
profile.
 */

/**
 * Implements hook_install().
 *
 * Perform actions to set up the site for this profile.
 *
 * @see system_install()
 */
function spark_install() {
  ...
}

/**
 * Implements hook_update_N().
 *   Description of hook update.
 */
function spark_update_10001(&$sandbox) {
  ...
}
```

```php
<?php
/**
 * @file
 * Install, update and uninstall functions for the spark install
profile.
 */

/**
 * Implements hook_install().
 *
 * Perform actions to set up the site for this profile.
 *
 * @see system_install()
 */
function spark_install() {
  ...
}

/**
 * Implements hook_update_N().
 *   Description of hook update.
 */
function spark_update_10001(&$sandbox) {
  ...
}
```

Example install hooks

Set Variables

```
/**
 * Implements hook_update_N().
 *   Enables default and admin themes.
 */
function profilename_update_10001(&$sandbox) {

  theme_enable(array('default_theme'));

  variable_set('theme_default', 'default_theme');

  variable_set('node_admin_theme', '1');

  return t('The admin and default themes have been set.');
}
```

Populate Menus

```
/**
 * Implements hook_update_N().
 *   Generate menu links
 */
function profilename_update_10002(&$sandbox) {
  cache_clear_all();
  //generate the menu links array
  //for the header menu

  $menu_links = array();
  $menu_links['home'] = array(
    'link_path' => '<front>',
    'link_title' => t('Home'),
    'weight' => 0,
  );
  $menu_links['about'] = array(
    'link_path' => 'about-us',
    'link_title' => t('About Us'),
    'weight' => 0,
  );
```

Populate Menus

```
  //run menu_link_save on all of the items.
  foreach ($menu_links as $menu_link) {
    $menu_link['menu_name'] = 'menu-header';
    $menu_link['module'] = 'menu';
    menu_link_save($menu_link);
  }

  menu_cache_clear_all();

  // Success!
  return t('Menu links have been generated');
}
```

Set blocks to regions

```
/**
 * Implements hook_update_N().
 *   Set copyright blocks to its region.
 */
function profilename_update_10003(&$sandbox) {
  //pull in the theme name
  global $theme_key;

  // Set our block values
  $values = array(
    array(
      'module' => 'custom_module',
      'delta'  => 'copyright',
      'region' => 'footer',
      'theme'  => $theme_key,
      'pages'  => '',
      'cache'  => 8,
      'status' => 1,
      'weight' => 0,
    ),
  );
```

Set blocks to regions

```
// If a db_entry for these blocks exists, delete it
foreach ($values as $position => $record) {
  $result = db_select('block', 'b')
    ->fields('b')
    ->condition('module', $record['module'], '=')
    ->condition('delta', $record['delta'], '=')
    ->condition('theme', $theme_key, '=')
    ->execute()
    ->fetchAssoc();
  if (!empty($result['bid'])) {
    db_delete('block')
    ->condition('module', $record['module'], '=')
    ->condition('delta', $record['delta'], '=')
    ->condition('theme', $theme_key, '=')
    ->execute();
  }
}
```

Set blocks to regions

```
  // Insert into the database any block values that aren't already
present
  foreach ($values as $record) {
    $query = db_insert('block')->fields(array('module', 'delta',
'theme', 'status', 'weight', 'region', 'pages', 'cache'));
    $query->values($record);
    $insert_result = $query->execute();
  }

 // Success!
  return t('Copyright blocks has been placed in the footer.');
}
```

Populate a Vocabulary

```
function profilename_requirements($phase) {
  $requirements = array();
  // Ensure translations don't break at install time
  $t = get_t();
  if ($phase == 'install' || $phase == 'update') {
    // Load taxonomy.module during install & updates
    if (!is_null(module_load_include('module', 'taxonomy'))) {
       $requirements['taxonomy'] = array(
        'title' => $t('Taxonomy Module'),
        'value' => $t('Taxonomy module loaded'),
        'severity' => REQUIREMENT_OK,
      );
    }
    else {
      $requirements['taxonomy'] = array(
        'title' => $t('Taxonomy Module'),
        'value' => $t('Taxonomy module not loaded'),
        'severity' => REQUIREMENT_ERROR,
      );
    }

  }
  return $requirements;
}
```

myplanet

Populate a Vocabulary

```
/**
 * Implements hook_update_N().
 *   Create vocabulary and taxonomy terms
 */
function profilename_update_10004(&$sandbox) {
  // Get the vocabulary
  $vocab = taxonomy_vocabulary_machine_name_load('terms');

  // If it's not created, do so (expected result)
  if(empty($vocab)) {
    $vocab = array(
      'name' => 'Terms',
      'machine_name' => 'terms',
      'description' => 'A list of technical terms',
      'hierarchy' => '0',
      'module' => 'taxonomy',
      'weight' => '0',
    );
    taxonomy_vocabulary_save((object)$vocab);
  }
```

Populate a Vocabulary

```
// Confirm that we've created it
$vocab = taxonomy_vocabulary_machine_name_load('terms');

// If we haven't created it, stop
if (empty($vocab)) {
  return t('Failed to create vocabulary.');
}

// Else if the "terms" vocabulary exists, populate it.
$terms = array(
  'Drupal' => 'A content management system',
  'Module' => 'A functional component of Drupal',
);
```

Populate a Vocabulary

```
  //  For each term defined in the above array, add it to the
"terms" vocabulary.
  foreach ($terms as $term_name => $term_description) {
    $term = new stdClass();
    $term->vid = $vocab->vid;
    $term->name = $term_name;
    $term->description = $term_description;
    taxonomy_term_save($term);
  }

  // Success!
  return t('Created vocabulary and taxonomy terms');
}
```

```
build-[profile].make
drupal-org.make
drupal-org-core.make
```

build-spark.make

```
api = 2
core = 7.x
; Include the definition for how to build Drupal core directly,
including patches:
includes[] = drupal-org-core.make

; Download the Spark install profile and recursively build all its
dependencies:
projects[spark][type] = profile
projects[spark][download][type] = git
projects[spark][download][branch] = 7.x-1.x
```

build-spark.make - with GIT!

```
api = 2
core = 7.x
; Include the definition for how to build Drupal core directly,
including patches:
includes[] = drupal-org-core.make

; Download the Spark install profile and recursively build all its
dependencies:
projects[spark][type] = profile
projects[spark][download][git]= git
projects[spark][download][url] = http://git.drupal.org/project/spark.git
projects[spark][download][revision] = master
```

drupal-org-core.make

```
; A separate drupal-org-core.make file makes it so we can apply core
patches if we need to.

api = 2
core = 7.x
projects[drupal][type] = core
projects[drupal][version] = 7.15

; CORE PATCHES
; Hide the profiles under /profiles, so Spark is the only one. This
allows the installation to start at the Language selection screen,
bypassing a baffling and silly choice, especially for non-native
speakers.
projects[drupal][patch][1780598] = http://drupal.org/files/spark-
install-1780598-5.patch
; This requires a core bug fix to not show the profile selection page
when only one profile is visible.
projects[drupal][patch][1074108] = http://drupal.org/files/1074108-
skip-profile-16-7.x-do-not-test.patch
```

drupal-org.make

```
; This is a standard make file for packaging the distribution along
with any contributed modules/themes or external libraries. Some
examples are below.
; See http://drupal.org/node/159730 for more details.

api = 2
core = 7.x

; Contributed modules; standard.

projects[responsive_bartik][type] = theme
projects[responsive_bartik][version] = 1.x-dev
projects[responsive_bartik][subdir] = contrib

projects[ctools][type] = module
projects[ctools][version] = 1.2
projects[ctools][subdir] = contrib
; Fix incompatibilities with jQuery 1.7.
projects[ctools][patch][1494860] = "http://drupal.org/files/ctools-
dependent-js-broken-with-jquery-1.7-1494860-30.patch"
```

```
tmp/
   docs/
   tests/
   snippets/
   scripts/
    build-dev.sh
    build-prod.sh
```

build-dev.sh

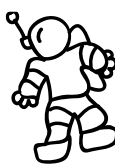
```
#!/bin/bash

# USAGE
# Run this script as given.
#
# $ bash build-dev.sh [ /fullpath/to/project.make ] [ /fullpath/to/
build/project ]
#
# (Default paths are for Vagrant VM.)

# Bail if non-zero exit code
set -e

PROJECT=profilename

# Set from args
BUILD_FILE="$1"
BUILD_DEST="$2"
```

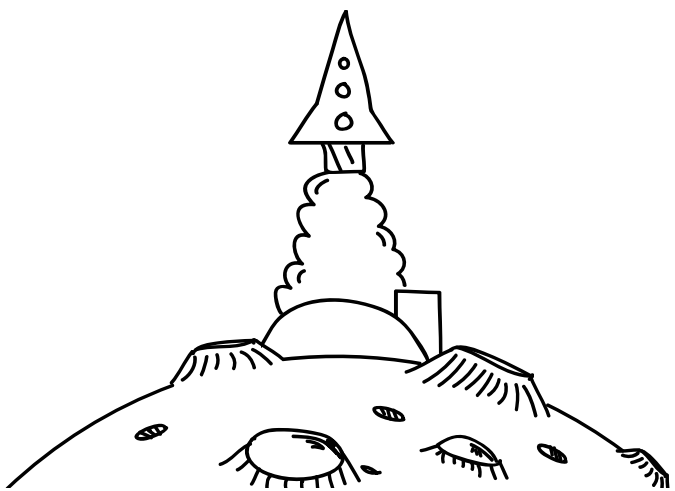
build-dev.sh

```
# Drush make the site structure
drush make ${BUILD_FILE} ${BUILD_DEST} \
  --working-copy \
  --prepare-install \
  --yes \
  --no-gitinfofile

chmod 666 ${BUILD_DEST}/sites/default/settings.php

echo "Appending settings.php snippets..."
for f in ${BUILD_DEST}/profiles/${PROJECT}/tmp/snippets/*.settings.php
do
  # Concatenate newline and snippet, then append to settings.php
  echo "" | cat - $f | tee -a ${BUILD_DEST}/sites/default/settings.php
> /dev/null
done

chmod 444 ${BUILD_DEST}/sites/default/settings.php
cd ${BUILD_DEST}
drush cc all
```

```
[profile].info
[profile].profile
[profile].install

build-[profile].make
drupal-org.make
drupal-org-core.make

modules/
  custom/
  features/
  contrib/

libraries/

themes/
  custom/
  contrib/

tmp/
  docs/
  tests/
  snippets/
  scripts/
    build-dev.sh
    build-prod.sh
```

Debate & discussion.





myplanet

Contact us:

Erin Marchak
Developer
erin@myplanetdigital.com

TF: 1.866.232.7456
Myplanetdigital.com

207-90C Centurian Dr
Markham, Ontario
L3R 8C5