General

TABLE OF CONTENTS

Get the current user

Note this will not get the user entity, but rather a user proxy with basic info but no fields or entity-specific data.

```
$user = \Drupal::currentUser();
```

```
$user = \Drupal\user\Entity\User::load(\Drupal::currentUser()->id());
```

Or

```
use \Drupal\user\Entity\User;
$user = User::load(\Drupal::currentUser()->id());
```

Get the logged in user name and email

```
$username = \Drupal::currentUser()->getAccountName();
```

or

```
$account_proxy = \Drupal::currentUser();
//$account = $account_proxy->getAccount();

// load user entity
$user = User::load($account_proxy->id());

$user = User::load(\Drupal::currentUser()->id());
$name = $user->get('name')->value;
```

Email

```
$email = \Drupal::currentUser()->getEmail();
```

or

```
$user = User::load(\Drupal::currentUser()->id());
$email = $user->get('mail')->value;
```

Check if you are on the Front page

```
$is_front = \Drupal::service('path.matcher')->isFrontPage();
```

The above statement will return either TRUE or FALSE. TRUE means you are on the front page.

Check if site is in system maintenance mode

```
$is_maint_mode = \Drupal::state()->get('system.maintenance_mode');
```

Get Node URL alias or Taxonomy Alias by Node id or Term ID

Sometimes we need a relative path and sometimes we need an absolute path. There is an $options parameter in the fromRoute() function where specify which you need.

Parameters:

- absolute true will return absolute path.
- absolute false will return relative path.

Returns the node alias. Note. If a nice url is not set using pathauto, you get `/node/1234`

```
use Drupal\Core\Url;
$options = ['absolute' => true];  //false will return relative path.

$url = Url::fromRoute('entity.node.canonical', ['node' => 1234], $options);
$url = $url->toString(); // make a string

// OR

$node_path = "/node/1";
$alias = \Drupal::service('path_alias.manager')->getAliasByPath($node_path);

// OR

$current_path = \Drupal::service('path.current')->getPath();
```

To get the full path with the host etc. this returns: https://ddev93.ddev.site/node/1

```
$host = \Drupal::request()->getSchemeAndHttpHost();
$url = \Drupal\Core\Url::fromRoute('entity.node.canonical',['node'=>$lab_home_nid]);
$url_alias = $url->toString();
$full_url = $host . $url->toString();
```

You can get the hostname, e.g. "drupal8.local", directly from the getHost() request with:

```
$host = \Drupal::request()->getHost();
```

Taxonomy alias

Return taxonomy alias

```
$options = ['absolute' => true];  //false will return relative path.
$url = Url::fromRoute('entity.taxonomy_term.canonical', ['taxonomy_term' => 1234], $options);
```

Get current Path

For node pages this will return `node/{node id}`, for taxonomy `taxonomy/term/{term id}`, for user `user/{user id}` if exists otherwise it will return the current request URI.

```
$currentPath  = \Drupal::service('path.current')->getPath();
```

Get current nid, node type and title

There are two ways to retrieve the current node – via the request or the route

```
$node = \Drupal::request()->attributes->get('node');
$nid = $node->id();
```

OR

```
$node = \Drupal::routeMatch()->getParameter('node');
if ($node instanceof \Drupal\node\NodeInterface) {
  // You can get nid and anything else you need from the node object.
  $nid = $node->id();
  $nodeType = $node->bundle();
  $nodeTitle = $node->getTitle();
}
```

If you need to use the node object in `hook_preprocess_page` on the preview page, you will need to use the `node_preview` parameter, instead of the

node parameter:

```
function mymodule_preprocess_page(&$vars) {

  $route_name = \Drupal::routeMatch()->getRouteName();

  if ($route_name == 'entity.node.canonical') {
    $node = \Drupal::routeMatch()->getParameter('node');
  }
  elseif ($route_name == 'entity.node.preview') {
    $node = \Drupal::routeMatch()->getParameter('node_preview');
  }
```

And from https://drupal.stackexchange.com/questions/145823/how-do-i-get-the-current-node-id when you are using or creating a custom block then you have to follow this code to get current node id. Not sure if it is correct.

```
use Drupal\Core\Cache\Cache;

$node = \Drupal::routeMatch()->getParameter('node');
if ($node instanceof \Drupal\node\NodeInterface) {
  $nid = $node->id();
}

// for cache
public function getCacheTags() {
  //With this when your node changes your block will rebuild
  if ($node = \Drupal::routeMatch()->getParameter('node')) {
    //if there is node add its cachetag
    return Cache::mergeTags(parent::getCacheTags(), ['node:' . $node->id()]);
  }
  else {
    //Return default tags instead.
    return parent::getCacheTags();
  }
}

public function getCacheContexts() {
  //if you depend on \Drupal::routeMatch()
  //you must set context of this block with 'route' context tag.
  //Every new route this block will rebuild
  return Cache::mergeContexts(parent::getCacheContexts(), ['route']);
}
```

How to check whether a module is installed or not

```
$moduleHandler = \Drupal::service('module_handler');
$module_name = 'views';
if ($moduleHandler->moduleExists($module_name)) {
  echo "$module_name installed";
}
else {
  echo "$module_name not installed";
}
```

Get current Route name

Routes are in the form: view.files_browser.page_1, test.example or test.settings_form

E.g from test.routing.yml

```
test.example:
  path: '/test/example'
  defaults:
    _title: 'Example'
    _controller: '\Drupal\test\Controller\TestController::build'
  requirements:
    _permission: 'access content'

test.settings_form:
  path: '/admin/config/system/test'
  defaults:
    _title: 'Test settings'
    _form: 'Drupal\test\Form\SettingsForm'
  requirements:
    _permission: 'administer test configuration'
```

This will return Drupal route. It returns entity.node.canonical for the nodes, system.404 for the 404 pages, entity.taxonomy_term.canonical for the taxonomy pages, entity.user.canonical for the users and custom route name that we define in modulename.routing.yml file.

```
$current_route = \Drupal::routeMatch()->getRouteName();
```

Get the current page title

You can use this in a controller, to return the current page title.

```
$request = \Drupal::request();
  if ($route = $request->attributes->get(\Symfony\Cmf\Component\Routing\RouteObjectInterface::ROUTE_OBJECT)) {
$title = \Drupal::service('title_resolver')->getTitle($request, $route);}
```

Get the current user

Note this will not get the user entity, but rather a user proxy with basic info but no fields or entity-specific data.

```
$user = \Drupal::currentUser();
```

To get the user entity, use this which gets the user service (\Drupal::currentUser()), gets the uid (->id()), then calls load() to load the real user object.

```
$user = \Drupal\user\Entity\User::load(\Drupal::currentUser()->id());
```

Or

```
use \Drupal\user\Entity\User;
$user = User::load(\Drupal::currentUser()->id());
```

Check if you are on the Front page

This will return true for the front page otherwise false.

```
$is_front = \Drupal::service('path.matcher')->isFrontPage();
```

Check if site in system maintenance mode

From web/core/modules/system/src/Access/CronAccessCheck.php

```
if (\Drupal::state()->get('system.maintenance_mode')) {
```

Retrieve query and get or post parameters ($_POST and $_GET)

Old style was:

```
$name = $_POST['name'];
```

Now use this for post vars:

```
$name = \Drupal::request()->request->get('name');
```

And this for gets

```
$query = \Drupal::request()->query->get('name');
```

For all items in get:

```
$query = \Drupal::request()->query->all();
$search_term = $query['query'];
$collection = $query['collection'];
```

Be wary about caching. From https://drupal.stackexchange.com/questions/231953/get-in-drupal-8/231954#231954 the code provided only works the first time so it is important to add a '#cache' context in the markup.

```
namespace Drupal\newday\Controller;
use Drupal\Core\Controller\ControllerBase;

class NewdayController extends ControllerBase {
    public function new() {
      $day= [
        "#markup" => \Drupal::request()->query->get('id'),
      ];
      return $day;
    }
}
```

The request is being cached, you need to tell the system to vary by the query arg:

```
$day = [
    '#markup' => \Drupal::request()->query->get('id'),
    '#cache' => [
        'contexts' => ['url.query_args:id'],
    ],
];
```

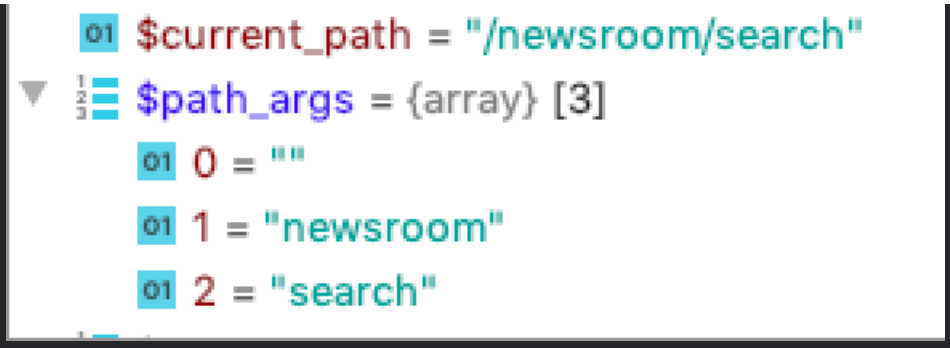
More about caching render arrays: https://www.drupal.org/docs/8/api/render-api/cacheability-of-render-arrays

Retrieve URL argument parameters

You can extract the url arguments with

```
$current_path = \Drupal::service('path.current')->getPath();
$path_args = explode('/', $current_path);
$term_name = $path_args[3];
```

For https://txg.ddev.site/newsroom/search/?country=1206



Get Current Language in a constructor

In dev1 - /modules/custom/iai_wea/src/Plugin/rest/resource/WEAResource.php we create the WeaResource class and using dependency injection, get the LanguageManagerInterface service passed in, then we call getgetCurrentLanguage(). This allows us to later retrieve the node

```
class WEAResource extends ResourceBase {

  /**
   * @var \Drupal\Core\Language\Language
   */
  protected $currentLanguage;

  /**
   * WEAResource constructor.
   *
   * @param array $configuration
   * @param string $plugin_id
   * @param mixed $plugin_definition
   * @param array $serializer_formats
   * @param \Psr\Log\LoggerInterface $logger
   * @param \Drupal\Core\Language\LanguageManagerInterface $language_manager
   */
  public function __construct(array $configuration, string $plugin_id, mixed $plugin_definition, array $serializer_formats, \Psr\Log\LoggerInterface $logger, LanguageManagerInte
    parent::__construct($configuration, $plugin_id, $plugin_definition, $serializer_formats, $logger);
    $this->currentLanguage = $language_manager->getCurrentLanguage();
  }
}
```

Later in the class, we can retrieve the correct language version of the node:

```
public function get($id) {
  if ($node = Node::load($id)) {
    $translatedNode = $node->getTranslation($this->currentLanguage->getId());
```

Of course, you can also get the language statically by using:

```
Global $language = Drupal::languageManager()->getLanguage(Language:TYPE_INTERFACE)
```

This is part of the packt publishing Mastering Drupal 8 module development video series: https://www.packtpub.com/product/mastering-drupal-8-development-video/9781787124493

Note. To test this in modules/custom/pseudo_client/get/

> php -S localhost:8888
>
> and put this in a browser:
>
> http://localhost:8888/get_item_from_drupal_core.php?domain=dev1&item=2716

> or
>
> http://localhost:8888/get_items_from_custom_code.php?domain=dev1
>
> OR just put this in browser without running php -S:
>
> http://dev1/iai_wea/actions/2716?_format=json

Add a variable to any page on the site

In the .theme file of the theme, add a `hook_preprocess_page` function like in themes/custom/dprime/dprime.theme:

```
function dprime_preprocess_page(&$variables) {
  $language_interface = \Drupal::languageManager()->getCurrentLanguage();

  $variables['footer_address1'] = [
    '#type'=>'markup',
    '#markup'=>'123 Disk Drive, Sector 439',
  ];
  $variables['footer_address2'] = [
    '#type'=>'markup',
    '#markup'=>'Austin, Texas 78759',
  ];
```

Then in the template file e.g. themes/custom/dprime/templates/partials/footer.html.twig

```
<div class="cell xlarge-3 medium-4">
  <address>
    <br />
    <br />
    Campus mail code: D9000<br />
    <a href="mailto:abc@example.com">abc@example.com </a>
  </address>
</div>
```

Add a variable to be rendered in a node.

From dev1 custom theme burger_burgler.

Here two vars `stock_field` and `my_custom_field` are added and will be rendered by a normal node twig file. The function hook_preprocess_node is in the .theme file at `themes/custom/burger_burgler/burger_burgler.theme`.

```
function burger_burgler_preprocess_node(&$variables) {

  $variables['content']['stock_field'] = [
    '#type'=>'markup',
    '#markup'=>'stock field here',
  ];

  $variables['content']['my_custom_field'] = [
    '#type' => 'markup',
    '#markup' => 'Hello - custom field here',
  ];
}
```

If you've tweaked your node twig template, you'll need to reference like this:

```
<div class="stock-field-class">
  {{ content['stock_field'] }}
</div>
```

Note. You can always just add a variable like

```
$variables['abc'] = 'hello';
```

which can be referenced in the template as {{ abc }} (or {{ kint(abc) }})

Add a bunch of variables to be rendered in a node

You can easily grab the node from the $variables with:

```
$node = $variables['node'];
```

Then to access a field in the node, you can just specify them by:

```
$node->field_ref_aof
$node->field_ref_topic
```

Here we grab a bunch of variables, cycles through them (for multi-value fields, which most of them are and build an array that can be easily rendered by twig:

From: themes/custom/txg/txg.theme

```
function txg_preprocess_node(&$variables) {
  $view_mode = $variables['view_mode']; // Retrieve view mode
  $allowed_view_modes = ['full']; // Array of allowed view modes (for performance so as to not execute on unneeded nodes)
  $node = $variables['node'];
  if (($node->getType() == 'news_story') && ($view_mode == 'full')) {
    $aofs = _txg_multival_ref_data($node->field_ref_aof, 'aof', 'target_id');
    $units = _txg_multival_ref_data($node->field_ref_unit, 'unit', 'target_id');
    $audiences = _txg_multival_ref_data($node->field_ref_audience, 'audience', 'target_id');
    $collections = _txg_multival_ref_data($node->field_ref_program_collection, 'collection', 'target_id');
    $topics = _txg_multival_ref_data($node->field_ref_topic,  'topic', 'target_id', 'taxonomy');
    $continents = _txg_multival_ref_data($node->field_continent, 'continent', 'value', 'list');
    $countries = _txg_multival_ref_data($node->field_ref_country, 'country', 'target_id');
    $related_news_items = array_merge($topics, $aofs, $units, $audiences, $collections, $continents,  $countries);
    $variables['related_news_items'] = $related_news_items;
  }
}

/**
 * Returns array of data for multivalue node reference fields
 * ref_field = entity reference field
 * param_name = parameter name to be passed as get value
 * value_type = indicates which field to retrieve from database
 * field_ref_type = variable to determine type of reference field
 * field_term_category =
 */
function _txg_multival_ref_data($ref_field, $param_name, $value_type, $field_ref_type = 'node') {
  $values = [];
  foreach($ref_field as $ref) {
    if ($field_ref_type == 'taxonomy') {
      $term = Drupal::entityTypeManager()->getStorage('taxonomy_term')->load($ref->$value_type);
      $title = $term->getName();
    }
    else {
      $title = $value_type == 'value' ? $ref->$value_type : $ref->entity->title->value;
    }
    $id = $ref->$value_type;
    $values[] = [
      'title' => $title,
      'id' => str_replace(' ', '+', $id),
      'param_name' => $param_name,
    ];
  }
  return $values;
}
```

Grabbing entity reference fields in hook_preprocess_node for injection into the twig template

You can easily pull in referenced fields by referring to them as

```
$node->field_sf_contract_ref->entity->field_how_to_order->value;
```

Where field_sf_contract_ref is the reference field, which points to an entity which has a field called field_how_to_order. Then we can jam it into the $variables array and refer to it in the twig template as ``

From web/themes/custom/dirt_bootstrap/dirt_bootstrap.theme

In function dirt_bootstrap_preprocess_node(&$variables)

```
if ($type === 'contract') {
  if ($view_mode === 'full') {
    $how_to_order_lookup = $node->field_sf_contract_ref->entity->field_how_to_order_lookup->value;
    $variables['how_to_order_lookup'] = $how_to_order_lookup;
    $contract_type = $node->get('field_contract_type')->value;
    if ($how_to_order_lookup === "Custom Text") {
      if ($contract_type === "DIRT") {
        $variables['how_to_order'] = $node->field_sf_contract_ref->entity->field_how_to_order->value;
      }
      else {
        $variables['how_to_order'] = $node->field_sf_contract_ref->entity->field_how_to_order_custom->value;
      }
    }
  }
}
```

Render a list created in the template_preprocess_node()

Here we create a list in the preprocess_node custom theme burger_burgler):

```
function burger_burgler_preprocess_node(&$variables) {

  $burger_list = [
    ['name' => 'Cheesburger'],
    ['name' => 'Mushroom Swissburger'],
    ['name' => 'Jalapeno bugburger'],
  ];
  $variables['burgers'] = $burger_list;

}
```

and render it in the twig template node--article--full.html.twig

```
<ol>
  {% for burger in burgers %}
    <li> {{ burger['name'] }} </li>
  {% endfor %}
</ol>
```

Indexing paragraphs so you can theme the first one

Posted on https://www.drupal.org/project/paragraphs/issues/2881460#comment-13291215

From themes/custom/dprime/dprime.theme

Add this to the theme:

```
/**
 * Implements hook_preprocess_field.
 *
 * Provides an index for these fields referenced as
 * in twig template.
 *
 * @param $variables
 */
function dprime_preprocess_field(&$variables) {
  if($variables['field_name'] == 'field_video_accordions'){
    foreach($variables['items'] as $idx => $item) {
      $variables['items'][$idx]['content']['#paragraph']->index = $idx;
    }
  }
}
```

field_video_accordions is the name of the field that holds the paragraph you want to count.

In the twig template for that paragraph, you can use the value paragraph.index as in:

```
{% if paragraph.index == 0 %}
  <li class="accordion-item is-active" data-accordion-item="">
{% else %}
  <li class="accordion-item" data-accordion-item="">
{% endif %}
```

Add meta tags using template_preprocess_html

Also covered at https://drupal.stackexchange.com/questions/217880/how-do-i-add-a-meta-tag-in-inside-the-head-tag

If you need to make changes to the <head> element, the hook_preprocess_html is the place to do it in the .theme file. Here we check to see that the content type is contract and then we create a fake array of meta tags and jam them into the $variables['page']['#attached']['html_head'] element. They are then rendered on the page.

```
/**
 * Implements hook_preprocess_html().
 */
function dir_bootstrap_preprocess_html(&$variables) {

  $node = \Drupal::routeMatch()->getParameter('node');
  if ($node instanceof \Drupal\node\NodeInterface) {
    if ($node->getType() == 'contract') {

      $brand_meta_tag[] = [[
        '#tag' => 'meta',
        '#attributes' => [
          'name' => 'brand',
          'content' => 'Dell',
        ]],
        'Dell',
      ];
..

$variables['page']['#attached']['html_head'][] = $brand_meta_tag;
```

Note that the extra “Dell” low down in the array appears to be a description of some kind – it isn’t rendered. If you don’t include the second “Dell” you could rather use

```
$page['#attached']['html_head'][] = [$description, 'description'];
```

For multiple tags, I had to do this version:

```
$brand_meta_tags[] = [[
  '#tag' => 'meta',
  '#attributes' => [
    'name' => 'brand',
    'content' => 'Dell',
  ]],
  'Dell',
];
$brand_meta_tags[] = [[
  '#tag' => 'meta',
  '#attributes' => [
    'name' => 'brand',
    'content' => 'Apple',
  ]],
  'Apple',
];

foreach ($brand_meta_tags as $brand_meta_tag) {
  $variables['page']['#attached']['html_head'][] = $brand_meta_tag;
}
```

And here I do a query and build some new meta tags from `themes/custom/dirt_bootstrap/dirt_bootstrap.theme`.

```
$brand_meta_tags = [];
$contract_id = $node->field_contract_id->value;
if ($contract_id) {

  //Lookup dirt store brand records with this contract id.
  $storage = \Drupal::entityTypeManager()->getStorage('node');
  $query = \Drupal::entityQuery('node')
    ->condition('type', 'sf_store_brands')
    ->condition('status', 1)
    ->condition('field_contract_id', $contract_id);
  $nids = $query->execute();
  foreach ($nids as $nid) {
    $store_brand_node = Node::load($nid);
    $brand = $store_brand_node->field_brand->value;
    if ($brand) {
      $brand_meta_tags[] = [[
        '#tag' => 'meta',
        '#attributes' => [
          'name' => 'brand',
          'content' => $brand,
        ]],
        $brand,
      ];
    }
  }
  foreach ($brand_meta_tags as $brand_meta_tag) {
    $variables['page']['#attached']['html_head'][] = $brand_meta_tag;
  }
}
```

How to strip % characters from a string

```
$str = "threatgeek/2016/05/welcome-jungle-tips-staying-secure-when-you%E2%80%99re-road";
echo $str . "\n";
//echo htmlspecialchars_decode($str) . "\n";

echo (urldecode($str)) . "\n";
echo urlencode("threatgeek/2016/05/welcome-jungle-tips-staying-secure-when-you're-road");

echo urldecode('We%27re%20proud%20to%20introduce%20the%20Amazing');
```

Remote media entities

For this project, I had to figure out a way to make media entities that really were remote images. i.e. the API provided images but we didn't want to store them in Drupal

I started by looking at https://www.drupal.org/sandbox/nickhope/3001154 which was based on https://www.drupal.org/project/media_entity_flickr.

I tweaked the nickhope module (media_entity_remote_file) so it worked but it had some trouble with image styles and thumbnails

A good solution (thanks to Hugo) is:

https://www.drupal.org/project/remote_stream_wrapper_widget

https://www.drupal.org/project/remote_stream_wrapper

Hugo suggests using this to do migration:

```
$uri = 'http://example.com/somefile.mp3';
$file = File::Create(['uri' => $uri]);
$file->save();
$node->field_file->setValue(['target_id' => $file->id()]);
$node->save();
```

There was no documentation so I added some at https://www.drupal.org/project/remote_stream_wrapper/issues/2875444#comment-12881516

Deprecated functions like drupal_set_message

Note. `drupal_set_message()` has been removed from the codebase so you should use `messenger()` but you can also use `dsm()` which is provided by the devel contrib module. This is useful when working through a problem if you want to display a message on a site during debugging.

From https://github.com/mglaman/drupal-check/wiki/Deprecation-Error-Solutions

Before

```
drupal_set_message($message, $type, $repeat);
```

After

```
\Drupal::messenger()->addMessage($message, $type, $repeat);
```

Read more.

Block excessive crawling of Drupal Views or search results

From Acquia.com

Sometimes, robot webcrawlers (like Bing, Huwaei Cloud, Yandex, Semrush, etc) can attempt to crawl a Drupal View's search results pages, and could also be following links to each of the view's filtering options. This places extra load on your site. Additionally, the crawling (even if done by legitimate search engines) may not be increasing your site's visibility to users of search engines.

Therefore, we suggest blocking or re-routing this traffic to reduce resource consumption at the Acquia platform, avoid overages to your

Acquia entitlements (for Acquia Search, Views & Visits, etc.), and to generally help your site perform better.

```
# EXAMPLE ROBOT BLOCKING CODE for Search pages or views.
# From: https://support-acquia.force.com/s/article/4408794498199-Block-excessive-crawling-of-Drupal-Views-or-search-results
#   NOTE: May need editing depending on your use case(s).
#
# INSTRUCTIONS:
# PLACE THIS BLOCK directly after the "RewriteEngine on" line
#   on your docroot/.htaccess file.
#
# This will block some known robots/crawlers on URLs when query arguments are present.
#   DOES allow basic URLs like /news/feed, /node/1 or /rss, etc.
#   BLOCKS only when search arguments are present like
#     /news/feed?search=XXX or /rss?page=21.
# Note: You can add more conditions if needed.
#   For example, to only block on URLs that begin with '/search', add this
#   line before the RewriteRule:
#     RewriteCond %{REQUEST_URI} ^/search
#
RewriteCond %{QUERY_STRING} .
RewriteCond %{HTTP_USER_AGENT} "11A465|AddThis.com|AdsBot-Google|Ahrefs|alexa site audit|Amazonbot|Amazon-Route53-Health-Check-Service|ApacheBench|AppDy
RewriteRule ^.* - [F,L]
```

Back to top

Page last modified: Jul 31 2023.

Edit this page on GitHub