Links, Aliases and URLs

TABLE OF CONTENTS

Create an external url

```
use Drupal\Core\Url
$url = Url::fromUri('http://testsite.com/go/here');
```

Create an internal url

First a simple URL:

```
use Drupal\Core\Url

$url = Url::fromUri('internal:/reports/search');

// Or.

$url = Url::fromUri('internal:/dashboard/achievements')

// Or Using link generator to create a GeneratedLink.
$url = = Url::fromUri('internal:/node/1');
$link = \Drupal::service('link_generator')->generate('My link', $url);

// OR
$url = Url::fromUri('internal:/');
// OR
'url' => Url::fromRoute('<front>'),
// OR
'url' => Url::fromRoute('hello_world.hello'),
```

Then something more complicated like this URL to `/reports/search?user=admin`

```
$option = [
  'query' => ['user' => 'admin'],
];
$url = Url::fromUri('internal:/reports/search', $option);
```

The Drupal Core Url Class

The `Drupal\Core\Url` class is often used to create URL's. Two important methods are:

`Url::fromRoute()` which takes a route name and parameters and

`Url::fromUri()` which takes an internal or external URL

See how these are used in some of the examples below.

The Drupal Core Link Class

Closely related and often used in conjunction with the Drupal Core URL class is the Drupal\Core\Link class.

You can generate links several different ways.

Create a link to a node

```
//Using link generator to create a GeneratedLink.
$url = Url::fromUri('internal:/node/1');
$link = \Drupal::service('link_generator')->generate('My link', $url);
```

Create a link to a path with parameters

To create a link to a path like `/reports/search?user=admin` use this code.

```
$option = [
 'query' => ['user' => 'admin'],
];
$url = Url::fromUri('internal:/reports/search', $option);

// use the Link class.
$link = Link::fromTextAndUrl('My link', $url);
$renderable_array = $link->toRenderable();
return $renderable_array;
```

Another way to create a link to a node:

```
$nid = $item->id();
$options = ['relative' => TRUE];  //could be absolute instead of relative.

$url = Url::fromRoute('entity.node.canonical',['node' => $nid], $options);
$link = \Drupal::service('link_generator')->generate('My link', $url);
```

Create a link from an internal URL

```
use Drupal\Core\Url

$url = Url::fromUri('internal:/reports/search');
$link = \Drupal::service('link_generator')->generate('My link', $url);

// ->toString() will extract the string of the URL.
$url_string = Url::fromUri('internal:/node/' . $id)->toString();
```

Check if a link field is empty

```
if (!$citation_node->field_link->uri) {
  // Empty.
}
```

Retrieve a link field from a node or a paragraph

The link field `field_link` is extracted from the node and a valid uri is extracted from that field.

```
$correction_node = Node::load($nid);
$current_url = $correction_node->get('field_link')->uri;
```

Or from a paragraph field

```
use Drupal\paragraphs\Entity\Paragraph;

$para = Paragraph::load($target_id);
$link = $para->field_link;
$link_uri = $para->field_link->uri;
```

Or a more convoluted example that extracts the url string for display from a link field.

```
if ($sf_contract) {
  // first() returns a Drupal\link\Plugin\FieldType\LinkItem
  $vendor_url = $sf_contract->field_vendor_url->first();
  if ($vendor_url) {
    // returns a Drupal\Core\Url.
    $vendor_url = $vendor_url->getUrl();
    $vendor_url_string = $vendor_url->toString();
  }
```

Removing `->first()` as in:

```
$vendor_url = $sf_contract->field_vendor_url;
```

returns a `Drupal\Core\Field\FieldItemList` which is a list of fields so you then would have to pull out the first field and extract the URI out of that. I'm not sure why Drupal considers it multiple values instead of just one. This was not set up as a multivalue field.

Retrieve a URL field

External links

You can get the URL (for external links) and then just the text part.

Note this doesn't work for internal links. Note also this slightly convoluted example has a reference field field_sf_contract_ref which has a link to another entity and the field_vendor_url->first()->getUrl() is the important part. Also note, this is a single-value field (not a multivalue field – so the first() call may be a little disturbing to those who expect things to be a little clearer.)

```
$vendor_url = $node->field_sf_contract_ref->entity->field_vendor_url->first()->getUrl();
if ($vendor_url) {
  $vendor_url = $vendor_url->getUri();
  //OR
  $vendor_url = $vendor_url->toString();
}
```

A slightly simpler example from a form

```
$citation_link = $citation->get('field_link');
if (!$citation_link->isEmpty()) {
  $citation_link = $citation->field_link->first()->getUrl()->toString();
}
```

Internal links

For internal links, use getUrl()for the URL and ->title for the title.

```
$instructions_node = Node::load($order_type_instructions_nid);
if ($instructions_node) {
  $order_link = $instructions_node->field_link->first();
  if ($order_link) {
    $uri = $order_link->uri;
    $variables['order_link_title'] = $order_link->title;
    $order_url = $order_link->getUrl();
    if ($order_url) {
      $variables['order_type_link'] = $order_url;
    }
  }
}
```

Get the NID from a URL Alias

To get the nid for a node, you can pass the URL alias to getPathByAlias.

```
// Given "/test-node, returns "/node/32".
$alias = "/test-node";
$path = \Drupal::service('path_alias.manager')->getPathByAlias($alias);
```

OR

If you have a URL for a node and you want its nid

```
$route = $url->getRouteParameters();
// first check if it's a node.
if (isset($route['node'])) {
  $nid = $route["node"];
}
```

Get the Taxonomy Term ID from a URL alias

Returns taxonomy/term/5

```
$term_path_with_tid = \Drupal::service('path_alias.manager')->getPathByAlias('/hunger-strike');
```

Get URL alias for a taxonomy term

This returns term/5 if no alias is set, otherwise it returns the alias.

```
$term5_url = Url::fromRoute('entity.taxonomy_term.canonical', ['taxonomy_term' => 5], $options);
$term5_alias = $term5_url->toString();
```

Get the User ID from a URL alias

Returns "/user/2"

```
//User
$user_path_with_uid = \Drupal::service('path_alias.manager')->getPathByAlias('/selwyn-the-chap');
```

Get the URL alias for a node

If no alias is set, this will return "/node/32". Note. If there are multiple aliases, you will get the most recently created one.

```
$node_path = '/node/32';
$node32_alias = \Drupal::service('path_alias.manager')->getAliasByPath($node_path);
```

Use this code if you need the absolute URL . If node/32 has a URL alias set to "/test-node" it returns "https://d9book2.ddev.site/test-node" . If you specify `absolute => FALSE`, it returns "/test-node" .

```
use Drupal\Core\Url;

// Note. If a pathauto url alias is not set, it returns '/node/32'
$nid = 32;
$options = ['absolute' => TRUE];
$url = Url::fromRoute('entity.node.canonical', ['node' => $nid], $options);
// make a string
$url_string = $url->toString();
```

Create a Node Alias

URL aliases are entities so you create them like you would any entity. Be sure to save() them.

```
$node_path = "/node/32";
$new_alias = "/test-node";

/** @var \Drupal\path_alias\PathAliasInterface $path_alias */
$my_node_alias = \Drupal::entityTypeManager()->getStorage('path_alias')->create([
  'path' => $node_path,
  'alias' => $new_alias,
  'langcode' => 'en',
]);
$my_node_alias->save();
```

Get the current Path

This returns the current relative path. For node pages, the return value will be in the form "/node/32" For taxonomy "taxonomy/term/5", for user "user/2" if it exists otherwise it will return the current request URI.

```
$currentPath  = \Drupal::service('path.current')->getPath();
// Or with alias and query string.
$alias = \Drupal::request()->getRequestUri();
// Or
$url_string = Url::fromRoute('<current>')->toString();
```

Get current nid, node type and title

There are two ways to retrieve the current node – via the request or via 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 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 get current Route name

A Drupal route is returned in the form of a string e.g. view.files_browser.page_1

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

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.

Get current Document root path

This will return the current document root path like "/var/www/html/project1".

```
$image_path = \Drupal::service('file_system')->realpath();
```

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

Retrieve query and GET or POST parameters ($_POST and $_GET)

For get variables

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

For POST variables:

```
$name = \Drupal::request()->request->get('name');
//or
$name  = $_POST['abc'];
```

For all items in a 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 arguments:

```
$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

Modify URL Aliases programmatically with hook_pathauto_alias_alter

The pathauto contrib module includes a nice hook that you can use to modify url aliases on the fly.

You just do your necessary checks (the current entity is stored in `\context['data']`) and change the alias that is passed. Pathauto does the rest.

As implemented in a module file.

```php
/**
 * Implements hook_pathauto_alias_alter().
 *
 * Note. This function is a stopgap measure to handle pathauto
 * token failing to return the parent menu item alias.
 * Using the pattern:
 * [node:menu-link:parent:url:path]/[node:title]
 * never returns the parent alias for some inexplicable reason.
 * This works fine on a fresh Drupal 9 site. Much research
 * yielded no results so this function hijacks pathauto's
 * efforts to create the alias.  It checks for the presence
 * of the parent menu alias.
 * That parent alias is inserted in the path if it isn't found.
 * If the parent alias is correctly inserted, it should not impact
 * the correct functioning of pathauto and token.
 */
function dirt_pathauto_alias_alter(&$alias, array &$context) {

  // Change the alias if it doesn't include the parent item.
  /** @var Drupal\pathauto\Entity\PathautoPattern $pattern*/
  $pattern = $context["pattern"];
  /* The pattern will be something like this
   * [node:menu-link:parent:url:path]/[node:title]
   * so I could test for it
   * but it doesn't seem necessary.
   * Of course, if they change the pattern later this
   * pattern will overwrite it.
   *
   */
  $bundles = ['page', 'audience', 'overview', 'program_area'];
  if (in_array($context["bundle"], $bundles)) {
    $op = $context['op'];
    if (in_array($op, ['insert', 'update', 'bulkupdate'])) {
      $parts = explode('/', $alias);
      if (count($parts) == 2) {
        if ($parts[0] == "") {
          //Missing the parent link.
          $id = $pattern->id();
          //Is this the default pathauto pattern?
          if ($id == "default") {
            //Is there a parent link?
            $nid = $context["data"]["node"]->id();
            $parent_link = _findParentMenuItem($nid);
            if ($parent_link) {
              $parent_alias = $parent_link['#url']->toString();
              //Update the alias to include the parent alias.
              if ($parent_alias) {
                $alias = $parent_alias . $alias;
              }
            }
          }
        }
      }
    }
  }
}
```

Also see an example at https://makedrupaleasy.com/articles/drupal-version-7-9-how-update-alias-programmatically-using-value-field

Drupal l() is deprecated

DEPRECATED

The l() method (lower case letter L) was a convenience wrapper for the link generator service's generate() method. So do this instead:

```
use Drupal\Core\Url;
use Drupal\Core\Link;

$url = Url::fromRoute('entity.node.edit_form', ['node' => NID]);
$project_link = Link::fromTextAndUrl(t('Open Project'), $url);
$project_link = $project_link->toRenderable();
// If you need some attributes.
$project_link['#attributes'] = ['class' => ['button', 'button-action', 'button--primary', 'button--small']];
print render($project_link);
```

Reference links

- Good reference from 2017 for creating links in Drupal
- #! code: Drupal 9: Programmatically Creating And Using URLs And Links, March 2022

Back to top

Page last modified: Jul 23 2023.

Edit this page on GitHub