Dominique De Cooman
Monday, August 8, 2011 - 22:07
interface [1]
d7tip [2]
usability [3]

In drupal 7 we have something called contextual links. It is that little wheel you see when you hover over blocks so you can edit them in place. It is a great usability improvement but it is not always that clear how to implement them.

The contextual links functionality used to be a contrib module in d6 now it is a core module.

Reusing already defined local tasks on a custom element

For example on a recent project we have a listing of links by a theme. The links are grouped by a theme node, we use a link field to hold the links to be printed. Now we would like contextual links on our little block it so we can edit the node so we can change our links. Here is how to do it:

```
<?php
  //We have a render array defined in a custom module
  $block['#theme_wrappers'] = array('thema_block');
  $block['title'] = l($node->title, $url);
  $block['class'] = "themablock themablock-" . $block_count . " num-" . $block_in_row;
  $block['more'] =  theme('more_link', array('title' => 'Read more', 'url' => $url));

  //The contextual links we place on our element
  $block['#contextual_links']['thema_blocks'] = array('node', array($node->nid));

  render $block;
?>
```

Let me explain what happens. All the links with a type MENU_LOCAL_TASK (as defined in their respective hook_menu) and a context MENU_CONTEXT_INLINE underlying the path node are fetched. The simple entry in the #contextual_links array will fetch all the node local taks.
Look to how the node module defines node/%node/edit and node/%node/delete and it will be clear.

```
<?php
  $items['node/%node/edit'] = array(
    'title' => 'Edit',
    'page callback' => 'node_page_edit',
    'page arguments' => array(1),
    'access callback' => 'node_access',
    'access arguments' => array('update', 1),
    'weight' => 0,
    'type' => MENU_LOCAL_TASK,
    'context' => MENU_CONTEXT_PAGE | MENU_CONTEXT_INLINE,
    'file' => 'node.pages.inc',
  );
  $items['node/%node/delete'] = array(
    'title' => 'Delete',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('node_delete_confirm', 1),
    'access callback' => 'node_access',
    'access arguments' => array('delete', 1),
    'weight' => 1,
    'type' => MENU_LOCAL_TASK,
    'context' => MENU_CONTEXT_INLINE,
```

```
    'file' => 'node.pages.inc',
  );
?>
```

This means you **cannot** fetch just any link defined in hook_menu, only the local task with a context inline will work for contextual links.

Next we need a hook theme to implement our elements theme wrapper.

```
<?php
/**
 * Implements hook_theme().
 */
function glue_theme() {
  $items = array(
    'thema_block' => array(
      'render element' => 'element',
      'template' => 'tpl/thema_block',
    ),
  );

  return $items;
}
?>
```

In a preprocess file we can assign our variables.

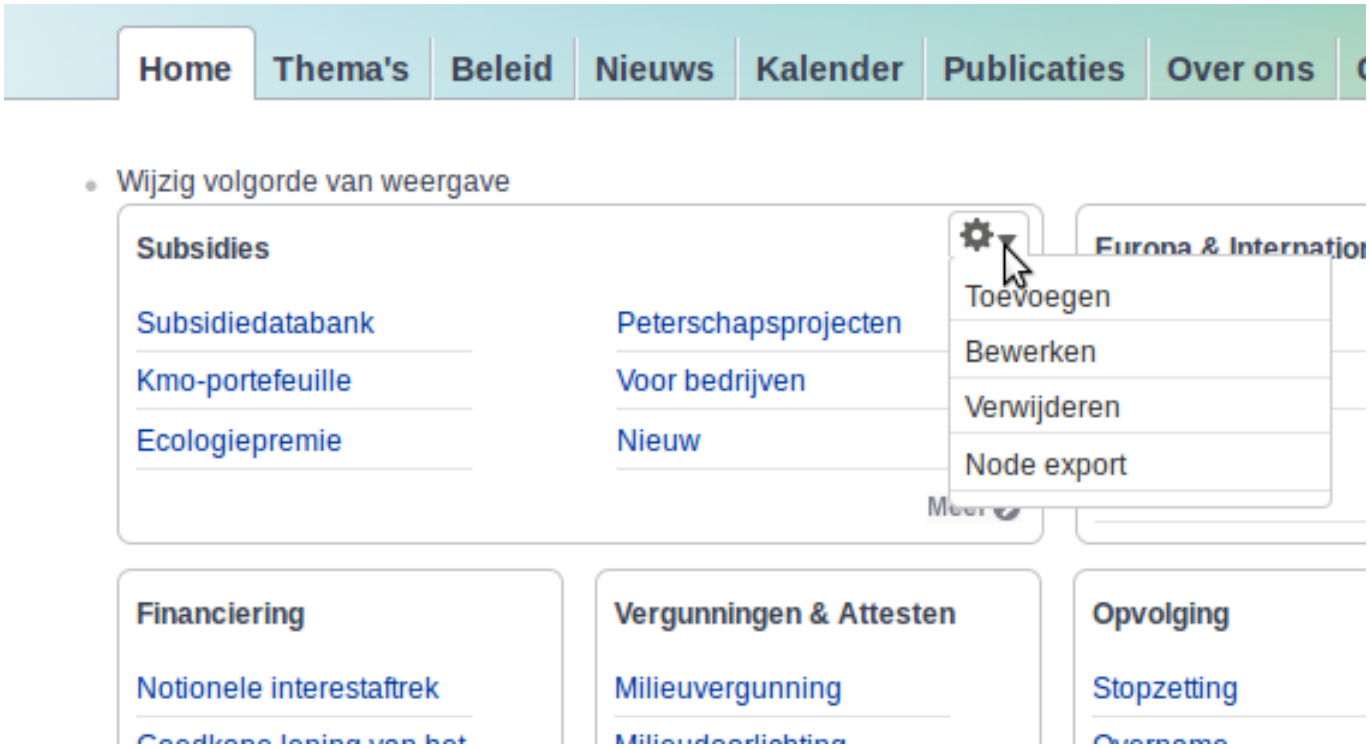
```
<?php
/**
 * Implements preprocess_thema_block()
 */
function glue_preprocess_thema_block(&$variables) {
  $variables['classes_array'][] = $variables['element']['class'];
  $variables['title'] = $variables['element']['title'];
  $variables['content'] = $variables['element']['#children'];
  $variables['more'] = $variables['element']['more'];
}
?>
```

Now the only thing we need a little template file to print it all.

```
<?php
<div class="<?php print $classes; ?>" <?php print $attributes; ?>>
  <?php print render($title_prefix); ?>
  <h2 <?php print $title_attributes; ?>><?php print $title;?></h2>
  <div class="content"<?php print $content_attributes; ?>>
    <?php print $content ?>
  </div>
  <?php print $more ?>
  <?php print render($title_suffix); ?>
</div>
?>
```

Here we can see we printed the $classes which will contain the contextual link classes. The contextual links are filled in by the render function. The actual links you ll find in $title_prefix, also put in place by the render function. By

rendering that array the html for the links will be printed. The jquery added by the contextual_links module will transform all arrays with the correct classes to the little wheel you can click.



Your own contextual links

Now if we want our own contextual links we need to create a hook_menu and define our own items as local tasks with an inline context. Here is an example found on the adding new contextual links page on http://drupal.org/node/1089922 [4]

In the example we will add our links in the hook_menu and with a hook_block_view_alter we will change the render array of the blocks and add our links to it.

```
<?php
  // An example contextual link menu item.
  $items['contextual/%/information'] = array(
    'title' => 'Block information',
    'type' => MENU_LOCAL_ACTION,
    'context' => MENU_CONTEXT_INLINE,
    'page callback' => 'contextual_example_page',
    'page arguments' => array(1),
    'access callback' => TRUE,
  );
  // To use local task menu items, there must be a parent page.
  $items['contextual'] = array(
    'title' => 'The contextual example page',
    'page callback' => 'contextual_example_page',
    'page arguments' => array(1),
    'access callback' => TRUE,
  );
?>
```

```
<?php
  /**
  * Implements hook_block_view_alter().
```

```
 */
 function contextual_example_block_view_alter(&$data, $block) {
   // Contextual links can be added as a renderable element to the content of
   // a render array. We check if the block has content, and if so add a
   // contextual link to it.
   if (isset($data['content']) && is_array($data['content'])) {
     $contextual_links = array(
       'contextual',
       array($block->module),
     );

     $data['content']['#contextual_links']['contextual_example'] = $contextual_links;
 }
}
?>
```

As you can see it works perfectly. The path in

```
<?php
$contextual_links = array('contextual', array($block->module));
?>
```

points to what we defined in the hook_menu fetches everything underneath the path 'contextual' and the $block->module is the argument passed.

If we would want to add these links to our custom element in previous example the only thing needed would be to adding them to the array.

```
<?php
 //We have a render array defined in a custom module
 $block['#theme_wrappers'] = array('thema_block');
 $block['title'] = l($node->title, $url);
 $block['class'] = "themablock themablock-" . $block_count . " num-" . $block_in_row;
 $block['more'] =  theme('more_link', array('title' => 'Read more', 'url' => $url));

 //The contextual links we place on our element
 $block['#contextual_links']['thema_blocks'] = array('node', array($node->nid));

 $block['#contextual_links']['whatever'] = array('contextual', array($something_usefull));
 //On $something_usefull you ll need to put something so your function contextual/%/information  knows what to do in the given context.

 render $block;
?>
```

Altering

Yet another method of adding contextual links is the alter method. This is taken from the api page [5]

```
function hook_menu_contextual_links_alter(&$links, $router_item, $root_path) {
// Add a link to all contextual links for nodes.
<?php
 if ($root_path == 'node/%') {
   $links['foo'] = array(
     'title' => t('Do fu'),
     'href' => 'foo/do',
     'localized_options' => array(
```

```
      'query' => array(
        'foo' => 'bar',
      ),
    ),
  );
 }
}
?>
```

Of course this will only work for existing paths on existing elements.

On views rows

We can also add contextual links to views rows. In our first example we want to add a contextual link to our slides of our slideshow view. Here is what we did:

```
<?php
/**
 *  Contextual links maker
 */
function glue_make_contextual_links($output, $nid) {
  $render_array =
      array(
        'children' => $output,
        '#theme_wrappers' => array('contextual_container'),
        '#contextual_links' => array(
          'glue' => array('node', array($nid)),
        ),
  );
  return render($render_array);
}

/**
 * Adds contextual links to views templates
 */
function glue_preprocess_views_view_field(&$vars) {
  if (isset($vars['field']->field_info['field_name']) && $vars['field']->field_info['field_name'] == 'field_slide_image') {
    $vars['output'] = glue_make_contextual_links($vars['output'], $vars['row']->nid);
  }
}
?>
```

We implemented the views_view_field preprocess hook and we wrapped the contextual links around our field we are displaying in the interface. Since the slide in slideshow is a node we can use the nid as argument to call for the correct contextual links allowing us to edit/delete the slide shown.

To be complete here is the theme_hook and the preprocess

```
<?php
/**
 * Implements hook_theme().
 */
function glue_theme() {
  $items = array(
    'contextual_container' => array(
      'render element' => 'element',
      'template' => 'tpl/contextual_container',
```

```
    ),
  );

  return $items;
}

/**
 * Implements hook_preprocess_contextual_container()
 */
function glue_preprocess_contextual_container(&$variables) {
  $variables['content'] = $variables['element']['children'];
}
?>
```

And the template file:

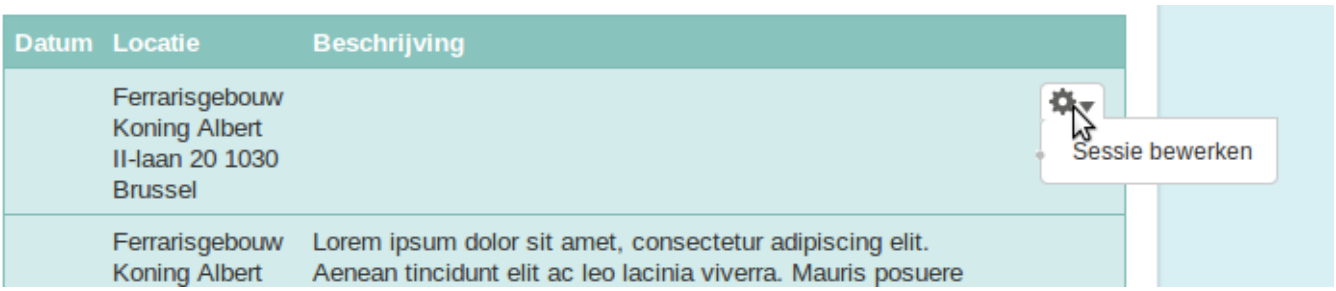
```
<?php
<div class="<?php print $classes; ?>" <?php print $attributes; ?>>
  <div <?php print $content_attributes; ?>>
    <?php print $content ?>
  </div>
  <div class="custom-contextual-links">
    <?php print render($title_suffix); ?>
  </div>
</div>
?>
```

In our next example we did it different. We have table and we want to add a contextual link field. We added a node edit link field to our view table display. In the template of that field named : we ve put this:

```
<?php
<div class="contextual-links-region">
```

```
  <div class="contextual-links-wrapper">
    <ul class="contextual-links">
      <li>
        <?php print $output; ?>
      </li>
    </ul>
  </div>
</div>
?>
```

This one is less clean because you can only add one item like this. In theory it would be possible to add it to the table row but this mean adapting the jquery from contextual links to also target tr elements, in the core it only targets div elements, putting them around tr elements is not valid html.



On models

On our site we also use the model module which provides "a container entity". It is an entity with just a title and its fieldable. Instead of implementing lots of hooks yourself you could take this module and have this entity working right away. (http://drupal.org/project/model [6]) We use the model entity store information that doesnt require the extra a node offers, like workflow, authoring, etc.. We just want to store stuff in fields. Perfect to store some header images and the path they need to be displayed on. Using this header_image bundle we can show a different header on the specified paths. Now that done wouldnt it be great to have contextual links on it? This is how we did it:

```
<?php
function glue_menu() {
  $items['admin/content/models/model/%/add'] = array(
    'title' => 'Add',
    'type' => MENU_LOCAL_ACTION,
    'context' => MENU_CONTEXT_INLINE,
    'page callback' => 'glue_model_add_type',
    'page arguments' => array(4),
    'access callback' => TRUE,
  );

  return $items;
}

/**
 * goto the adding page for the model
 */
function glue_model_add_type($mid) {
  $destination = drupal_get_destination();
  $query = db_select('model', 'm');
  $result = $query
        ->condition('m.model_id', $mid, '=')
        ->fields('m', array('type'))
        ->execute()
```

```
        ->fetch();

  $params = array(
    'query' => array(
      'destination' => $destination['destination'],
    )
  );
  //where to go next
  $_GET['destination'] = url('admin/content/models/add/' . $result->type, $params);
  drupal_goto('admin/content/models/add/' . $result->type, $params);
}

function create_logo() {
  $model = model_load($item->model_id);
  $logo = field_view_field('model', $model, 'field_logo_logo', 'full');
  $logo[0]['#image_style'] = 'logo_style';
  $logo_and_link = array(
    '#type' => 'link',
    '#title' => render($logo[0]),
    '#href' => '',
    '#options' => array('html' => TRUE, 'attributes' => array('class' => $css_class)),
    '#contextual_links' => array(
      'logo' => array('admin/content/models/model', array($model->model_id)),
    )
  );
  if (isset($model->field_logo_link['und'][0]['url'])) {
    $header_and_link['#href'] = $model->field_logo_link['und'][0]['url'];
  }
  return $logo_and_link;
}
?>
```

We added a menu callback to register local actions to add each model. In the callback we query the type and we make it go to the add page. This way when we create a render array the contextual links to not only edit and delete but also the add link will be present.

Finaly how to put the node/add/%type in a contextual link trick

The same trick we did with the models we can do with the nodes:

```
<?php
function glue_menu() {
  $items['node/%/add'] = array(
    'title' => 'Add',
    'type' => MENU_LOCAL_ACTION,
    'context' => MENU_CONTEXT_INLINE,
    'page callback' => 'glue_node_add_type',
    'page arguments' => array(1),
    'access callback' => TRUE,
  );

  return $items;
}

/**
 * goto the adding page for the node
 */
function glue_node_add_type($nid) {
  $destination = drupal_get_destination();
  $query = db_select('node', 'n');
  $result = $query
          ->condition('n.nid', $nid, '=')
          ->fields('n', array('type'))
          ->execute()
          ->fetch();

  $params = array(
    'query' => array(
      'destination' => $destination['destination'],
    )
  );
  //where to go next
  $_GET['destination'] = url('node/add/' . $result->type, $params);
  drupal_goto('node/add/' . $result->type, $params);
}
?>
```

The slides are nodes so in the screenshot above you can "Toevoegen" Which means "Add" in dutch. This was caused by the last piece of code.

I m sure they are lots of other methods to add contextual links. So I you have any post them in the comments.

Source URL: https://dominiquedecooman.com/blog/drupal-7-tip-add-contextual-links-anything

Links:
[1] https://dominiquedecooman.com/blog-topics/interface
[2] https://dominiquedecooman.com/blog-topics/d7tip
[3] https://dominiquedecooman.com/blog-topics/usability
[4] http://drupal.org/node/1089922
[5] http://api.drupal.org/api/drupal/modules--system--system.api.php/function/hook_menu_contextual_links_alter/7
[6] http://drupal.org/project/model