Routing

Overview

All components can be accessed through a query string by using the option parameter which will equate to the name of the component. For example, to access the "Blog" component, you could type http://yourhub.org/index.php?option=com_blog.

When SEF URLs are being employed, the first portion after the site name will almost always be the name of a component. For the URL http://yourhub.org/blog, the first portion after the slash translates to the component com_blog. If a matching component cannot be found, routing will attempt to match against an article section, category, and/or page alias.

While not required, most components will have more detailed routing instructions that allow SEF URLs to be made from and converted back into query strings that pass necessary data to the component. This is done by the inclusion of a file called router.php.

The Router

Every router.php file has a class with two methods: build() which takes a query string and turns it into a SEF URL and parse() which deconstructs a SEF URL back into a query string to be passed to the component.

```
<?php
namespace ComponentsExampleSite;

use HubzeroComponentRouterBase;

class Router extends Base
{
 public function build(&$query)
 {
  $segments = array();

  if (!empty($query['task']))
  {
   $segments[] = $query['task'];
   unset($query['task']);
  }
  if (!empty($query['id']))
  {
   $segments[] = $query['id'];
   unset($query['id']);
  }
```

```
  if (!empty($query['format']))
  {
   $segments[] = $query['format'];
   unset($query['format']);
  }

  return $segments;
 }

 public function parse($segments)
 {
  $vars = array();

  if (empty($segments))
  {
   return $vars;
  }
  if (isset($segments[0]))
  {
   $vars['task'] = $segments[0];
  }
  if (isset($segments[1]))
  {
   $vars['id'] = $segments[1];
  }
  if (isset($segments[2]))
  {
   $vars['format'] = $segments[2];
  }

     return $vars;
 }
}
```

The build() Method

This method is called when using Route::url(). Route::url() passes the query string (minus the option={componentname} portion) to the method which returns an array containing the necessary portions of the URL to be constructed *in the order* they need to appear in the final SEF URL.

```
// $query = 'task=view&id=123&format=rss'
public function build(&$query)
```

```
{
 $segments = array();

 if (!empty($query['task']))
 {
  $segments[] = $query['task'];
  unset($query['task']);
 }
 if (!empty($query['id']))
 {
  $segments[] = $query['id'];
  unset($query['id']);
 }
 if (!empty($query['format']))
 {
  $segments[] = $query['format'];
  unset($query['format']);
 }

 return $segments;
}
```

Will return:

```
Array(
 'view',
 '123',
 'rss'
);
```

This will in turn be passed back to Route::url() which will construct the final SEF URL of example/view/123/rss.

The parse() Method

This method is automatically called on each page view. It is passed an array of segments of the SEF URL that called the page. That is, a URL of example/view/123/rss would be separated by the forward slashes with the first segment automatically being associated with a component name. The rest are stored in an array and passed to parse() which then associates each segment with an appropriate variable name based on the segment's position in the array.

```
public function parse($segments)
{
 $vars = array();

 if (empty($segments))
 {
  return $vars;
 }
 if (isset($segments[0]))
 {
  $vars['task'] = $segments[0];
 }
 if (isset($segments[1]))
 {
  $vars['id'] = $segments[1];
 }
 if (isset($segments[2]))
 {
  $vars['format'] = $segments[2];
 }

    return $vars;
}
```

Note: Position of segments is very important here. A URL of example/view/123/rss could yield completely different results than a URL of example/rss/view/123.