Router

To route requests there is special class **ClickBlocks\Net\Router** in the framework. This class associates regular expressions (patterns) that correspond URL of the current request with any action specified in the form of a delegate. Besides of URL patterns the class also allows you to set any actions for certain HTTP-methods.

Router class is used by POM and API modules.

URL patterns

This regular expression is superset Perl-compatible regular expressions (PCRE) in which some characters have the special meaning:

- Symbol #. It is used for moving of some URL part into a variable. For example, template **category/#category#/#ID#** will match the following URL:

  - category/my_first_category/5
  - category/my_second_category/10
  - category/my_third_category/15
  - ...

  In this case the following values will correspond the variable "category": my_first_category, my_second_category, my_third_category and so on. The variable "ID" will take the values: 5, 10, 15 and so on.

  If it is necessary to use symbol # without special meaning then you need to escape it by backslash: **category/#category#/#ID##fragment**.

- Symbol | is used for setting of regular expression that bound with template variable. For example, if we want to limit the variable "ID" from previous example only integer numbers then we can write such regular expression: **category/#category#/#ID|[0-9].#**. If after symbol | we use another one then it will be treated as regular symbol of alternative in regular expressions.

The binding of delegates with URL patterns

You can bind a delegate with some URL pattern by using method **bind()**:

```
$router = new Router();
$router->bind('add/#a#/#b#', 'MyClass->foo', 'GET|POST');
```

In this example URL pattern **/add/#a#/#b#** is bound with delegate **MyClass->foo**. And besides, this binding will work only if the current HTTP method is GET or POST. When the delegate is invoked it takes values of pattern variables **#a#** и **#b#** as parameters.

Besides of method **bind()** there are two more methods allowing to redirect if URL of the current request matches the given URL pattern.

The following method redirects to the specified URL:

```
$router = new Router();
$router->redirect('articles/#article#', '/posts/#article#', 'GET');
```

The first parameter of the method is URL pattern to match. The second one is URL to redirect. And besides you can use pattern variables in the URL to replace its parts. For example above, request with URL like **/articles/route_engine** will redirect to address **/posts/route_engine**.

Method **redirect()** has also the fourth parameter that determines a delegate which will be invoked instead of redirect to the specified address. The delegate takes, as parameter, the fully formed URL to redirect. For example:

```
// Creates our delegate that simply outputs URL to redirect.
$redirect = function($url)
{
  return $url;
}

// Creates an object of the router.
$router = new Router();
// Binds the delegate with an URL pattern.
```

```
$router->redirect('articles/#article#', '/posts/#article#', 'GET', $redirect);
```

By using method **secure()** you can, according to URL pattern, change HTTP protocol to HTTPS and vice versa. For example, we want that all URL containing the path "/secure/" will redirect to the same address but with protocol HTTPS. In this case, the code could be so:

```
$router = new Router();
$router->secure('#.+/secure/.+#i', true, 'GET|POST|PUT|DELETE');
```

> Note that the first parameter of the method is the usual Perl-compatible regular expression.

The second method parameter determines the protocol type. If it equals TRUE, the redirect will change the current HTTP protocol to HTTPS. Otherwise, the redirect changes the protocol to HTTP.

> Redirect will not happen if the current protocol as desired.

The third method parameter determines HTTP method or methods to match URL pattern.

Also as in the case of method **redirect()**, method **secure()** takes, as the fourth parameter, a delegate that will be invoked instead of the redirect. The delegate will take the URL to redirect.

Additional routing options

Besides specifying the HTTP-methods, you can also set additional options affecting the routing. First, using method **component()** you can specify which part of the current URL should be used to match URL pattern.

```
$router = new Router();
// Binds some action with URL pattern.
$router->bind('my.host.(com|net|org)/posts/#post#', 'MyClass->foo', 'GET|POST')
// Determines which part of the URL should be matched the pattern.
       ->component(URL::HOST | URL::PATH);
```

Parameter of method **component()** is a number corresponding some constant (or constant combination) of class **URL()**. Each constant corresponds to the defined URL part. Here are the full list of these constants:

- **ALL** - corresponds to the whole URL.
- **SCHEME** - corresponds to the URL scheme.
- **HOST** - corresponds to the URL host.
- **PATH** - corresponds to the URL path.
- **QUERY** - corresponds to URL query.
- **FRAGMENT** - corresponds to the URL fragment.

> Note that almost all methods of class **ClickBlocks\Net\Router** returns an object of this class. It means that you call multiple methods chain (fluent interface), making your code more readable.

The following method allows you to specify a regular expression applied to a pattern variable.

```
$router = new Router();
// Binds some action with an URL pattern.
$router->bind('posts/#post#', 'MyClass->foo', 'GET|POST')
// Sets a regular expression for pattern variable "post".
       ->validation('/^[a-zA-Z0-9_]+$/i');
```

You can also specify regular expressions for random variables directly in the URL pattern. However, the advantage of the method **validation()** is that we can specify an arbitrarily complex regular expressions are not clogging up the main URL pattern.

When the delegate associated with some URL pattern will be invoked it takes, as parameters, values of the pattern variables. To pass additional parameters to the delegate there is method **args()**:

```
$router = new Router();
// Binds some action with an URL pattern.
$router->bind('posts/#post#', 'MyClass->foo', 'GET|POST')
// Determines additional parameters of the delegate.
```

```
       ->args(['arg1' => 'val1', 'arg2' => 'val2']);
```

> Note that if the name of the pattern variable is the same name as an additional parameter, then the value of a pattern variable will be passed to the delegate instead of the appropriate additional parameter.

By default the variable names of the pattern does not correspond to the names of delegate variables. However, using method **coordinateParameterNames()** you can demand that argument names of the delegate will correspond to pattern variable names. Here's how it does:

```
$router = new Router();
// Binds some action with an URL pattern.
$router->bind('posts/#post#', 'MyClass->foo', 'GET|POST')
// Turns on synchronization with delegate parameters.
       ->coordinateParameterNames();
```

Sometimes it is more convenient to pass pattern variables to the delegate as an associative array as an argument of the delegate. This can be done using method **extra()**:

```
$router = new Router();
// Binds some action with an URL pattern.
$router->bind('posts/#post#', 'MyClass->foo', 'GET|POST')
// We specify the name of the delegate parameter,
// which will take an associative array of pattern variables.
       ->extra('foo');
```

But what happens when the pattern match is found, but the delegate can not be called (e.g. there is no required class or method). By default, in this case an exception occurs. However, by using **ignoreWrongDelegate()** we can disable throwing exceptions. In such a case, if the delegate can not be called it will be ignored.

```
$router = new Router();
// Binds some action with an URL pattern.
$router->bind('posts/#post#', 'MyClass->foo', 'GET|POST')
// Turns on ignoring of invalid delegates.
       ->ignoreWrongDelegate();
```

Routing

Once all of the URL patterns can be defined we can launch the routing request. To do it you should use method **route()**.

The first method parameter is HTTP method or methods for routing. If this parameter is not defined, the method of the current HTTP-request will be used.

The second parameter is an URL string (or an object of class **ClickBlocks\Net\URL**), which will be analyzed for matching the URL pattren. If this parameter is not specified, URL of the current request will be taken.

Example of the routing of the current request to the server:

```
$router = new Router();
// Sets redirect to secured protocol for some URL.
$router->secure('#.+/admin.+#i', true, '*');
// Sets redirect for the specified URL.
$router->redirect('article/#article#', '/posts/#article#', 'GET');
// Binds delegates with URL patterns.
$router->bind('category/#category#', 'MyClass->getCategory', 'GET|POST')
       ->args(['type' => 'general']);
$router->bind('posts/#post#', 'MyClass->getPost', 'GET|POST')
       ->ignoreWrongDelegate()
       ->extra('params');
// Routing
$router->route();
```

Method **route()** returns an array like `['success' => ..., 'result' => ...]`. Where the first element informs us whether the matching with the pattern is happened (the value TRUE) or not (FALSE). If the matching is happened then the second element of the array contains the result of the

delegate execution.