Routing

lesson #basic06

James L. Parry
B.C. Institute of Technology

Agenda

Routing

Segment-based routing is the normal CodeIgniter way of resolving the handling of an incoming request.

There are alternatives: wildcard routing, regular expression routing, callback routing, and HTTP verb routing.

For the truly adventurous you can even remap requests on the fly.

We'll look at each of these in turn.

Routing Rules

The controller folder convention can be over-ridden by specifying routing rules, in `application/config/routes.php`

An example such rule lets you change the default controller:
`$route['default_controller'] = 'welcome';`

If you specify multiple rules, they are tested consecutively until one fits.

Any reserved route rules must come before any wildcard or regular expression rules!

Wildcard Routing

A routing rule can use a "wildcard" token, `(:num)` to match a numeric segment value, or `(:any)` to match any segment value.

Specify an expression using these as the "key" for a routing rule, and specify the proper destination as the "value".

You can use the substitution token `$n` to reference a URI segment in the original request.

Some examples of routing rules:

```
$route['blog/joe'] = "blogs/users/34";

$route['product/(:num)'] =
"catalog/product_lookup_by_id/$1";

$route['page/(:any)'] = 'welcome/page/$1';

$route['secret/(:any)/(:any)'] =
'youllneverfindme/$1/$2';
```

Regular Expression Routing

You can also use a regular expression in a routing rule.

For instance:
`$route['products/([a-z]+)/(\d+)'] = "$1/id_$2";`
would remap `/products/banana/25` to `/banana/id_25`

Another example:
`$route['([a-z]+)/register'] = 'assimilate/$0';`
would remap `/jim/register` to `/assimilate/jim`

Callback Routing

If you are using PHP >= 5.3 you can use callbacks in place of the normal routing rules to process the back-references.

For instance:

```
$route['products/([a-zA-Z]+)/edit/(\d+)'] =
function ($product_type, $id)
{
  return 'catalog/product_edit/' . strtolower($product_type) . '/' . $id;
};
```

This is code to execute, NOT a controller. In the above example, the two matched tokens would be passed as the parameters `$product_type` and `$id`.

HTTP Verb Routing

You can specify routing rules that apply to specific HTTP request types. This would be applicable to utility and service controllers.

Some examples, in a RESTful fashion:

```
$route['products']['PUT'] = 'product/insert';
$route['products/(:num)']['DELETE'] = 'product/delete/$1';
```

Remapping On the Fly

Separate from any routing rules, a controller itself can affect the handling of an incoming request, through a `_remap` method.

The _remap method lets you overwrite the normal behaviour for controller methods. If your controller has this method, it is always called.

An example which hides the real name of a method ...

```
public function _remap($method, $parms=array() )
    if ($method == 'work') $this->guessMeIfYouCa
    else return call_user_func_array($method,$pa
}
```

Remapping for Localization

Remapping could also be used to handle of having a language code between the domain name and the "normal" part of a URL, for instance "https://msdn.microsoft.com/en-us/library/windows/desktop".

This way, URLs have the form `/idiom/class/method/parm1/...`, where "idiom" is a language identifier.

You need a routing rule to swap the idiom and the intended controller:
`$routes['(:any)/(:any)/(.+)'] = '$1/$0/$2';`

This would redirect the example above to "/library/en-us/windows/desktop".

Then, in your `Library` or base controller...

```
public function _remap($method, $parms=array() )
    $idiom = $method; // the alleged method is r
    $this->lang->load('translations_for_this_con
    $real_method = array_shift($parms); // grab
    return call_user_func_array($real_method,$pa
}
```

The above would handle `/fr/find/apple` by calling `Find::apple(...)` with the `fr` translations loaded.

See the user guide for more details.

Hooks

The CodeIgniter framework, internally, performs the following steps to handle a request:

1. Apply routing rules to determine the controller and method to use
2. Instantiate the controller
3. Invoke the appropriate method, capturing output
4. Return the output to the browser

CodeIgniter provides "hooks", to let you inject processing before or after each of the milestones shown to the left.

Refer to the user guide for the gory details ... this is just a taste!

Hook Points

The following are some of the "hook points" that you can use:

- pre_system
- pre_controller
- post_controller_constructor
- post_controller
- display_override
- post_system

Hooks are configured similarly to routes, and you can have multiple hooks for the same hook point.

Adding Hooks

Configure your hooks in `application/config/hooks`

An example:

```
$hookie = array(
  'class'=>...,
  'function'=>...,
  'filename'=>...,
  'filepath'=>...,
  'params'=>...
);

$hooks[entrypoint][] = $hookie;
```

Loose Ends

Upcoming lessons and tutorials may address some more exotic controller issues:

- Webapp error handling (avoid getting fired)
- Handling AJAX requests
- Handling service requests
- Handling plugins for additional resources

Caution: Before using a feature, eg. hooks or routing, RTFM!

Coding Conventions

Required:

- Class and file naming - "ucfirst"

Allowed:

- deviations from the suggested, for good reason
- multiple classes (related) in file

Bad ideas:

- PHP namespaces (for now), unless you know what you are doing

Suggested (for methods & variables):

- words separated by underscores
- underscores in front of internal items
- Allman style braces & indenting
- commenting, PHPdoc style!
- value & type comparison (===)!
- don't use closing PHP tag at end of file!

Congratulations!

You have completed lesson #basic06: Routing

If you would take a minute to provide some feedback, we would appreciate it!

The next activity in sequence is: lab04 Working With Controllers

You can use your browser's back button to return to the page you were on before starting this activity, or you can jump directly to the course homepage, organizer, or reference page.