Integrate CodeIgniter with Yar

levin.lin@ringcentral.com

Purpose: Introduce Yar into CodeIgniter to enable RPC ability.
Sessions: (1) install Yar extension; (2) change CI framework to support Yar

Session 1. Install Yar extension

Yar is an excellent RPC framework, which support PHP/C, GET/POST, SYNC/ASYNC ways.

Here is a link of introduction: http://www.laruence.com/2012/09/15/2779.html.

Source code and installation guide can be found from github: https://github.com/laruence/yar.

Personal installation experience described as follow:

Step 1. Download Yar source code and extract to a certain directory.

```
unzip yar-master.zip
cd yar-master
```

Step 2. Phpize.

```
which php
/usr/local/opt/php54/bin/php
/usr/local/opt/php54/bin/phpize
Configuring for:
PHP Api Version:         20100412
Zend Module Api No:      20100525
Zend Extension Api No:   220100525
Cannot find autoconf. Please check your
autoconf installation and the

$PHP_AUTOCONF environment variable.  Then,
rerun this script.
brew install autoconf
==> Downloading
https://homebrew.bintray.com/bottles/autoconf
-2.69.yosemite.bottle.1.tar.gz
#############################################
########################### 100.0%
==> Pouring autoconf-
2.69.yosemite.bottle.1.tar.gz
Error: The `brew link` step did not complete
successfully
The formula built, but is not symlinked into
/usr/local
Could not symlink bin/autoconf
/usr/local/bin is not writable.

You can try again using:
  brew link autoconf
==> Summary
     /usr/local/Cellar/autoconf/2.69: 70
files, 3.1M
```

```
export
PATH=/usr/local/Cellar/autoconf/2.69/bin:$PAT
H
echo $PATH
/usr/local/Cellar/autoconf/2.69/bin:/usr/loca
l/Cellar/mysql/5.6.25/bin:/usr/local/opt/php5
4/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:
/sbin
sudo /usr/local/opt/php54/bin/phpize
Configuring for:
PHP Api Version:         20100412
Zend Module Api No:      20100525
Zend Extension Api No:   220100525
```

Step 3. Configure & make

```
./configure  --with-php-
config=/usr/local/opt/php54/bin/php-config
make
```

Step 4. Install & enable

```
make install
Installing shared extensions:
/usr/local/Cellar/php54/5.4.43_2/lib/php/exte
nsions/no-debug-non-zts-20100525/
vi /usr/local/etc/php/5.4/php.ini
extension=yar.so
sudo apachectl restart (restart mod_php)
```

Session 2. Change CI framework to support Yar

Step 1. Basic design

Brother Laruence introduce here adding two lines of code to use Yar in an OOP PHP script. It is simple enough, but unrealistic for an existing large project.

We prefer a simple additional class called `Rpc` and function called `call()` to initialize any customized class and its functions without doing much changes to existing code.

Step 2. `Rpc` class implement

```
<?php
/**
 * levin.lin
 */

defined('BASEPATH') OR exit('No direct script
access allowed');

class Rpc extends CI_Controller {
    public function __construct(){
        parent::__construct();
    }
    public  function call(){
        $controller_name = ucfirst($this-
>input->post_get("controller",TRUE));
        $method = $this->input-
>post_get("method",TRUE);
        $this->load-
>controller($controller_name, false);
        $controller =
"My_{$controller_name}";
        $service = new Yar_Server($this-
>$controller);
        $service->handle();
    }
}
```

Step 3. Extend CI_Loader to enable controller loader

```
vi /path/to/application/core/MY_Loader.php
```

```
<?php
/*
   enable controller loader
   add by levin 2015.11.20
   usage: $this->load-
>controller($object_name,$callByCI);
          $this->my_{$object_name}->index();
*/
defined('BASEPATH') OR exit('No direct script
access allowed');
```

```
class MY_Loader extends CI_Loader{
    public function __construct(){
        parent::__construct();
    }
    public function
controller($path_file_name,
$calledByCI=true){ //enable path/to/file_name
        $CI = & get_instance();
        $file_path =
APPPATH.'controllers/'.$path_file_name.'.php'
;
        $file_path_info =
explode("/",$path_file_name);
        $object_name = end($file_path_info);
        $my_object_name = 'My_'.$object_name;
//add prefix "My_" avoid conflict
        $class_name = ucfirst($object_name);
        if (isset($CI->$my_object_name)){
//conflict check
            show_error('The controller name
you are loading is the name of a resource
that is already being used: '.$object_name);
        }
        if(file_exists($file_path)){
            require_once($file_path);
            $CI->$my_object_name = new
$class_name($calledByCI);
        }else{
            show_error("Unable to load the
requested controller class: ".$class_name);
        }
    }
}
?>
```

Step 4. Make slight changes to system function load_class() to avoid some errors

```
function &load_class($class, $directory = 'libraries',
$param = NULL)
{
    ……
    foreach (array(APPPATH, BASEPATH) as $path)
    {
        if
(file_exists($path.$directory.'/'.$class.'.php'))
        {   //original code doesn't have if branch
            if($path == BASEPATH){
                $name = 'CI_'.$class;
            }else{
                $name = $class;
            }
            ……
        }
        ……
    }
    ……
}
```

Step 5. Make slight changes to basic controller construct function to avoid Session error

```php
public function __construct($calledByCI=true)
{
    self::$instance =& $this;
    foreach (is_loaded() as $var => $class)
    {
        if($class == "Session" && !$calledByCI){
        }else{
            $this->$var =& load_class($class);
        }
    }
    ......
}
```

Step 6. Add default parameter `calledByCI=true` to constructor of any controller aimed to invoked by Yar, just like:

```php
public function __construct($calledByCI=true){
    parent::__construct($calledByCI);
}
```

Step 7. Do the happy Rpc
(1) Single Call Example:

```php
<?php
$client = new Yar_Client("http://localhost:8088/rpc/call?controller=test");
/* the following setopt is optinal */
$client->SetOpt(YAR_OPT_CONNECT_TIMEOUT, 1000);

/* call remote service */
$result = $client->testForRpc();
var_dump($result);
?>
```

(2) Multi Call Example:

```php
<?php
$time=1;
$result = array();
$begin = time();
function callback($retval, $callinfo) {
    global $time;
    global $result;
    global $begin;
    if($callinfo == NULL){
        echo "all async calls sent\n";
    }else{
        echo "this is the $time time call\n";
        $time++;
        $result[] = $retval;
    }
    if(count($result) == 3){
        $end = time();
        var_dump($result);
        echo "total time cost: ".($end-$begin)."\n";
    }
}

Yar_Concurrent_Client::call("http://localhost:8088/rpc/call?controller=test",
"testForRpc", array("controller"=>"test"), "callback");
Yar_Concurrent_Client::call("http://localhost:8088/rpc/call?controller=test",
"testForRpc", array(), "callback");
Yar_Concurrent_Client::call("http://localhost:8088/rpc/call?controller=test",
"testForRpc", array(), "callback");
Yar_Concurrent_Client::loop(); //send
?>
```