$view = new ViewModel();

$this->layout('layout/scripts/layout');

==================================================================================

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$sql = "select um.\*,rm.role\_name,um.is\_active,umd.user\_name supervisor,umd.email\_id supervisoremail from user\_master as um

join user\_master\_details umd on um.user\_id=umd.user\_id and umd.is\_active=1 and umd.is\_delete=0

join role\_master rm on rm.role\_id = um.role\_id

where um.company\_id='$compid' and um.is\_approved=1

and um.is\_delete='0' and um.is\_active=1 order by umd.user\_name ASC";

$result = $db->query($sql)->execute();

$data = array();

foreach($result as $r){$data[] = $r;}

return $data;

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$sql = new Sql($db);

$select = $sql->select()

->from(array('tp'=>'training\_program'))

->join(array('tt' => 'training\_type'), 'tt.training\_type\_id = tp.training\_type\_id')

->join(array('im' => 'item\_master'), 'tp.item\_id = im.item\_id')

->where(array('im.is\_active'=>1))

->where(array('tp.is\_active'=>1))

->where(array('tp.company\_id'=>$compid))

->where(array('tp.training\_program\_id'=>$trainingProgId));

$statement = $sql->prepareStatementForSqlObject($select);

$results = $statement->execute();

==================================================================================

public function updateanywhere($mytable, array $data, $where) {

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$table = new TableGateway($mytable, $db);

$results = $table->update($data, $where);

}

public function deleteanywhere($mytable, $where) {

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$table = new TableGateway($mytable, $db);

$results = $table->delete($where);

}

public function insertanywhere($mytable, array $data) {

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$table = new TableGateway($mytable, $db);

$results = $table->insert($data);

}

public function lastinsertid($table, array $data) {

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$table = new TableGateway($mytable, $db);

$results = $table->insert($data);

$id = $table->getLastInsertValue();

return $id;

}

==================================================================================

$all\_params = $this->params()->fromRoute();

$all\_param = array(); $i=1;

foreach($all\_params as $key=>$value)

{ if($key=='id'.$i) {$all\_param[$value] = $all\_params['value'.$i]; $i++; } }

==================================================================================

$view->setTerminal(true);

==================================================================================

$this->redirect()->toUrl($basepath.'/Excel/'.$file);

==================================================================================

$basepath=$this->getServiceLocator()->get('Zend\View\Renderer\RendererInterface')->basePath();

=================================================================================

Model

$db =$this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$sql = "select \* from company\_cms\_supervisor where company\_id='$compid' and is\_active='1'";

$result1 = $db->query($sql)->execute(); //print\_r($result1);exit;

return $result1;

==================================================================================

<?php

namespace Application\Model\DbTable;

use Zend\Db\TableGateway\AbstractTableGateway;

use Zend\ServiceManager\ServiceLocatorAwareInterface;

use Zend\ServiceManager\ServiceLocatorInterface;

use Zend\Db\TableGateway\TableGateway;

use Zend\Db\Sql\Sql;

class Cmshistory extends AbstractTableGateway implements ServiceLocatorAwareInterface

{

protected $\_name = 'cms\_history';

public $serviceLocator;

public function setServiceLocator(ServiceLocatorInterface $serviceLocator){

$this->serviceLocator = $serviceLocator;

}

public function getServiceLocator(){

return $this->serviceLocator;

}

}

?>

==================================================================================

--this is the way for paginator without adapter----

$dataItems = array(

0 => array('id'=>1, 'name'=>'samsonasik'),

1 => array('id'=>2, 'name'=>'abdul malik ikhsan'),

2 => array('id'=>3, 'name'=>'superman'),

);

$paginator = new \Zend\Paginator\Paginator(new

\Zend\Paginator\Adapter\ArrayAdapter($dataItems)

);

$paginator->setCurrentPageNumber($page);

$paginator->setItemCountPerPage(2);

$view->setVariable('test', $paginator);

==================================================================================

---this is the way for paginator with adapter-----

$paginator = $this->getServiceLocator()->get('Application\Model\Abc')->fetchfunction();

$paginator->setCurrentPageNumber((int) $this->params()->fromQuery('page', 1));

$paginator->setItemCountPerPage(10);

Fetch function of model is as follows:

fetchfunction(){

$paginatorAdapter = new DbSelect( //DbSelect comes from use Zend\Paginator\Adapter\DbSelect

// our configured select object

$select,

// the adapter to run it against

$this->tableGateway->getAdapter(),

// the result set to hydrate

$resultSetPrototype

);

$paginator = new Paginator($paginatorAdapter); //Paginator comes from use Zend\Paginator\Paginator

return $paginator;

}

after this implement this in view and create a separate pagination file like this

view->

<table class="table">

<tr>

<th>Title</th>

<th>Artist</th>

<th>&nbsp;</th>

</tr>

<?php foreach ($this->paginator as $album) : // <-- change here! ?>

<tr>

<td><?php echo $this->escapeHtml($album->title);?></td>

<td><?php echo $this->escapeHtml($album->artist);?></td>

<td>

<a href="<?php echo $this->url('album',

array('action' => 'edit', 'id' => $album->id));?>">Edit</a>

<a href="<?php echo $this->url('album',

array('action' => 'delete', 'id' => $album->id));?>">Delete</a>

</td>

</tr>

<?php endforeach; ?>

</table>

<?php

// add at the end of the file after the table

echo $this->paginationControl(

// the paginator object

$this->paginator,

// the scrolling style

'sliding',

// the partial to use to render the control

'partial/paginator.phtml',

// the route to link to when a user clicks a control link

array(

'route' => 'album'

)

);

?>

paginatiion file->

<?php if ($this->pageCount): ?>

<div>

<ul class="pagination">

<!-- Previous page link -->

<?php if (isset($this->previous)): ?>

<li>

<a href="<?php echo $this->url($this->route); ?>?page=<?php echo $this->previous; ?>">

<<

</a>

</li>

<?php else: ?>

<li class="disabled">

<a href="#">

<<

</a>

</li>

<?php endif; ?>

<!-- Numbered page links -->

<?php foreach ($this->pagesInRange as $page): ?>

<?php if ($page != $this->current): ?>

<li>

<a href="<?php echo $this->url($this->route);?>?page=<?php echo $page; ?>">

<?php echo $page; ?>

</a>

</li>

<?php else: ?>

<li class="active">

<a href="#"><?php echo $page; ?></a>

</li>

<?php endif; ?>

<?php endforeach; ?>

<!-- Next page link -->

<?php if (isset($this->next)): ?>

<li>

<a href="<?php echo $this->url($this->route); ?>?page=<?php echo $this->next; ?>">

>>

</a>

</li>

<?php else: ?>

<li class="disabled">

<a href="#">

>>

</a>

</li>

<?php endif; ?>

</ul>

</div>

<?php endif; ?>

reference link - http://framework.zend.com/manual/current/en/tutorials/tutorial.pagination.html

==================================================================================

Controller

namespace Application\Controller;

use Zend\Mvc\Controller\AbstractActionController;

use Zend\View\Model\ViewModel;

use Zend\Session\Container;

use Zend\Authentication\AuthenticationService;

class ContentmanagementController extends AbstractActionController {

$view = new ViewModel();

return $view;

$this->redirect('any\_cont\_act', array('controller'=>'index', 'action'=>'home'));

$this->layout('layout/scripts/layout');

$view->setVariable('compid', $compid);

==================================================================================

how to implement different sql methods in \Zend\Db\Sql\Sql

'course\_count'=>new \Zend\Db\Sql\Expression('Count(\*)')

reference link - http://stackoverflow.com/questions/13808605/zend-framework-2-and-select-count-query

==================================================================================

**from a controller call another phtml which may be available in different module or controller then use this command. This command can also send parameters.**

$this->getServiceLocator()->get('viewmanager')->getRenderer()->render(

'<module>/<controller>/<view-name>',[

'param1' => $param1,

'param2' => $param2,

]);

$view->setTemplate('phtml\_name');//this code only renders the phtml of same module and controller without parameters

==================================================================================

what to do when we create a new controller;

**Create a new file in Controller folder which mentions the following.**

namespace Application\Controller;

use Zend\Mvc\Controller\AbstractActionController;

use Zend\View\Model\ViewModel;

use Zend\Session\Container;

use Zend\Authentication\AuthenticationService;

class IndexController extends AbstractActionController {

}

In IndexController class we will have our actions like this

class IndexController extends AbstractActionController {

public function indexAction{

}

}

If the action has its phtml (view) the every action will have this

public function indexAction{

$view = new ViewModel();

//code

return $view;

}

If the action does not have its view then we dont need to create instance of ViewModel

whenever you create a new controller put an entry in module.config.php of that module you have created new controller.

in module.config.php there are many arrays. in 'controller'=>array('invokables'=>array()); this array we are supposed to make entry. for example-

'controllers' => array(

'invokables' => array(

'Application\Controller\Index' => 'Application\Controller\IndexController',

),

),

==================================================================================

what to do when we create new plugin

Create a new file in Controller/Plugin folder which mentions the following.

namespace Application\Controller\Plugin;

use Zend\Mvc\Controller\Plugin\AbstractPlugin;

use Zend\Authentication\AuthenticationService;

use Zend\Http\Headers;

use Zend\Db\TableGateway\TableGateway;

use Zend\View\Model\JsonModel;

class CommonFunctions extends AbstractPlugin {

}

CommonFunctions will have function which we call from different controllers or views.

if we create new plugin we are supposed make entry in module.config.php in following mentioned method.

'controller\_plugins' => array(

'invokables' => array(

'CommonFunctions' => 'Application\Controller\Plugin\CommonFunctions',

),

),

==================================================================================

what to do when we create a new model

Create a new file in Model folder which mentions the following.

namespace Application\Model\Assessmentreports;

use Zend\Db\TableGateway\AbstractTableGateway;

use Zend\ServiceManager\ServiceLocatorAwareInterface;

use Zend\ServiceManager\ServiceLocatorInterface;

use Zend\Db\Sql\Sql;

use Zend\Db\TableGateway\TableGateway;

class UserTable extends AbstractTableGateway implements ServiceLocatorAwareInterface {

public $serviceLocator;

public function setServiceLocator(ServiceLocatorInterface $serviceLocator){

$this->serviceLocator= $serviceLocator;

}

public function getServiceLocator(){

return $this->serviceLocator;

}

}

we are supposed to make an entry in Module.php

public function getServiceConfig()

{

return array(

'factories' => array(

'Application\Model\UserTable' => function($sm) {

$tableGateway = $sm->get('ClassmasterGateway');

$table = new UserTable($tableGateway);

return $table;

},

'ClassmasterGateway' => function($sm){

$dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');

return new TableGateway('company\_announcement', $dbAdapter, null);

},

),

);

}

==================================================================================

index.php explaination

chdir(dirname(\_\_DIR\_\_));

//\_\_DIR\_\_ =>function return the path till current directory of current file. In our case we are in index.php & directory is public so it return D:\xampp\htdocs\zend2design\public

//

//dirname => function return parenet directory of current directory. so it return zend2design (project name)

//

//chdir => function changes the path of root directory. the new path will be the parameter passed in this function. in our case we have passed dirname(\_\_DIR\_\_), which means we have passed zend2design in this.

// Decline static file requests back to the PHP built-in webserver

if (php\_sapi\_name() === 'cli-server' && is\_file(\_\_DIR\_\_ . parse\_url($\_SERVER['REQUEST\_URI'], PHP\_URL\_PATH))) {

return false;

}

//php\_sapi\_name()->Returns the type of interface between web server and PHP

//Return values include aolserver, apache, apache2filter, apache2handler, caudium, cgi (until PHP 5.3), cgi-fcgi, cli, cli-server, continuity, embed, fpm-fcgi, isapi, litespeed, milter, nsapi, phttpd, pi3web, roxen, thttpd, tux, and webjames.

//parse\_url($\_SERVER['REQUEST\_URI'], PHP\_URL\_PATH)->return the url from root directory (zend2design) till current location

//var\_dump(\_\_DIR\_\_ . parse\_url($\_SERVER['REQUEST\_URI'], PHP\_URL\_PATH));die;

// Setup autoloading

require 'init\_autoloader.php';

//this command creates folders to install library if not installed. If installed it returns (return;).

//one more thing that this file calls require 'vendor/autoload.php'. This is imperative for composer to work. If composer does not find autoload.php then it cannot work.

// Run the application!

Zend\Mvc\Application::init(require 'config/application.config.php')->run();

//this command calls application.config.php file

//this file mentions the name of all modules in our application.

//This also tells the path to modules

//this file tells the path to global.php and local.php

==================================================================================

common commands in action

$view->setVariable('compid',$compid);

$this->redirect()->toRoute('any\_cont\_act', array('controller'=>'index', 'action'=>'home'));

$this->redirect()->toRoute('any\_cont\_act/default', array('controller'=>'index', 'action'=>'home', 'id1'=>$id1, 'id2'=>$id2, 'value1'=>$value1, 'value2'=>$value2));

**if there in no layout and phtml for action then use this command at the end of action**

return $this->response;

**if there in no layout for action then use this command at the end of action**

$view->setTerminal(true);

**this is how you call plugin in controller action**

$this->CommonFunctions()->getFieldName($compid);

**if you are recieving key and their values from url and the key values are separated from slash then use this**

$all\_params = $this->params()->fromRoute();

$all\_param = array(); $i=1;

foreach($all\_params as $key=>$value){if($key=='id'.$i){$all\_param[$value]=$all\_params['value'.$i];$i++;}}

**this is how you access the values for the particular key**

$mode = @$all\_param['key'];

**if you are recieving key and their values from url and the key values are separated from $ and = then use this.**

$value = $this->params()->fromQuery('key');

**if you are recieving key and their values from post method then use this**

$value = $this->params()->fromPost('key');

**if you want to access model from action use this**

$mymodel = $this->getServiceLocator()->get('Application\Model\Assessmentreports\Assessmentreports');

**and to call the function defined in that model, use this**

$mymodel->abcFunction($variable1, $variable2);

**if you call a function of plugin from action or other plugin and it return toRoute command but it is not redirecting. In that case try this at the receing end.**

$result = $this->HelperAuth()->somFunction($username);//someFunction return toRoute

if ($result instanceof Response) {

// $result is instance of response so i am redirecting here.

return $result;

}

to use this you need to do this at the top of your receiving page

use Zend\Stdlib\ResponseInterface as Response;

**to use session in action, use this**

$sesscompanyid = new Container('companyid');

$sesscompanyid->compid = $compid;

**to use this, you need to add this at the top**

use Zend\Session\Container;

**if you want to check whether the user is authenticated, then do this**

$auth = new AuthenticationService();

if($auth->hasIdentity()) {

$identity = $auth->getIdentity();

$user\_id = $identity->user\_id;

$company\_id=$identity->company\_id;

$roleid=$identity->role\_id;

}

**to use this you need to add this at the top**

use Zend\Authentication\AuthenticationService;

**if you want to access baseurl from anywher except phtml then use this**

$base\_url = $this->serviceLocator->get('Zend\View\Renderer\RendererInterface')->basePath();

**for phtml, use this**

$base\_url = $this->basePath();

**if you want to check whether the coming request is ajax request then**

$this->getRequest()->isXmlHttpRequest()//use this if

**to set layout for that particular action**

$this->layout('layout/scripts/userlayout');

**if you want to logout the user, then**

$auth = new Authenticationservice();

$auth->clearIdentity();

$this->redirect()->toRoute('any\_cont\_act', array('controller'=>'index', 'action'=>'index'));

==================================================================================

namespace Application\Controller\Plugin;

use Zend\Mvc\Controller\Plugin\AbstractPlugin;

use Zend\Authentication\AuthenticationService;

use Zend\ServiceManager\ServiceLocatorAwareInterface;

use Zend\ServiceManager\ServiceLocatorInterface;

class Assignmentfunctions extends AbstractPlugin{

};

**common commands in plugin**

**to access adapter in plugin**

$db = $this->getController()->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

to access any model

$itemDB = $this->getController()->getServiceLocator()->get('Application\Model\Item');

**to access controller or action name**

$controller = $this->getController()->getEvent()->getRouteMatch()->getParam('controller');

**to access basePath in plugin**

$baseUrl = $this->getController()->getServiceLocator()->get('Zend\View\Renderer\RendererInterface')->basePath();

**to access other plugins from plugins**

1. (recommended)

$this->getController()->getServiceLocator()->get('ControllerPluginManager')->get('Redirect')->toRoute('any\_cont\_act', array('controller'=>'index', 'action'=>'index'));

2.

$this->getController()->getPluginManager()->get('Layout')->setTemplate('layout/scripts/standard/'.$layout);

3.

$this->getController()->CommonFunctions()->checkjob\_titlename($job\_title,$compid, $item\_id);

==================================================================================

common commands in model

$db =$this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$sql="select cs.change\_password from company\_settings as cs where cs.company\_id='".$compid."' ";

$result =$db->query($sql)->execute()->current();//to fetch one row

return $result;

$db =$this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$sql="select cs.change\_password from company\_settings as cs where cs.company\_id='".$compid."' ";

$result =$db->query($sql)->execute();

$results = array();

foreach($result as $r){$results[]=$r;}//to fetch all data

return $results;

//Zend\Db\Sql\Sql

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$sql = new Sql($db);

$select = $sql->select()

->from(array('cci'=>'custom\_cms\_info'))

->columns(array('page\_title', 'page\_id'))

->where(array('cci.company\_id'=>$compid));

$statement = $sql->prepareStatementForSqlObject($select);

$results = $statement->execute();

//$results=$statement->execute()->getAffectedRows() to get affected rows

//Zend\Db\Sql\Sql

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$sql = new Sql($db);

$select = $sql->select()

->from(array('bt'=>'blog\_title'))

->columns(array('blog\_title\_id','blog\_cat\_id','title', 'desc', 'date\_add', 'date\_update','created\_by','last\_updated\_by'))

->join(array('bct'=>'blog\_cat\_topic') , 'bct.blog\_title\_id = bt.blog\_title\_id')

->join(array('bc'=>'blog\_comment') , 'bt.blog\_title\_id = bc.blog\_title\_id',array('comment\_date'=>'date\_add'), 'left')

->where(array('bt.active\_status'=>1, 'bt.company\_id', $compid))

->order('bt.date\_add DESC');

$select ->where

->in('bct.blog\_cat\_id', $blogids);

//look at the in operator of where how it is used and joins too

$select = $sql->select()

->from(array('bt'=>'blog\_title'));

$select ->where

->nest

->like('desc', '%'.$keyword.'%')

->or

->like('title', '%'.$keyword.'%')

->unnest;

//nest and unnest behaves like brackets ()

//Zend\Db\TableGateway\TableGateway

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$table = new TableGateway($mytable, $db);

$abc = $table->update($data, $where);

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$table = new TableGateway($mytable, $db);

$abc = $table->delete($where);

$db = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$table = new TableGateway($mytable, $db);

$abc = $table->insert($data);

$id = $table->getLastInsertValue();

$adapter = $this->getServiceLocator()->get('Zend\Db\Adapter\Adapter');

$db = new TableGateway('class\_master', $adapter);

$results = $db->select(array('company\_id'=>$compid, 'class\_id'=>$class\_id, 'is\_active'=>1));

$data = array();

foreach($results as $result){$data[] = $result;}

return $data;

// in this array of select works as where condition. You can select data from tablegateway only and only if you want data from only one table and where condition comprises only column and their values for which data needs to be fetched.

//if this is not the case with you then use Zend\Db\Sql\Sql. It provides so many functionalities like joins, huge variety of where conditions etc.

==================================================================================

common commands in phtml(view)

$this->basePath();

**//you cannot access plugin and models from phtml. You need to view helper for that.**

==================================================================================

<?php

namespace Application\View\Helper;

use Zend\View\Helper\AbstractHelper;

use Application\Model\MyTable;

use Zend\ServiceManager\ServiceLocatorInterface as ServiceLocator;

class Testhelper extends AbstractHelper

{

protected $serviceLocator;

public function \_\_construct(ServiceLocator $serviceLocator)

{

$this->serviceLocator = $serviceLocator;

}

public function getservicelocator()

{

return $this->serviceLocator;

}

public function getPlugin($pluginName)

{

$plugin = $this->getservicelocator()->get('ControllerPluginManager')->get($pluginName);

return $plugin;

}

public function common\_functions()

{

$plugin = $this->getservicelocator()->get('ControllerPluginManager')->get('CommonFunctions');

return $plugin;

}

public function acl()

{

$plugin = $this->getservicelocator()->get('ControllerPluginManager')->get('Acl2');

return $plugin;

}

public function contentmanagement()

{

$plugin = $this->getServiceLocator()->get('Application\Model\Contentmanagement\Contentmanagement');

return $plugin;

}

}

?>

//In above code i have written few cases

//You need to create a folder View at the same place where Controller directory is placed

//save view helpers in View folder.

//then write some code in module.php so that zend 2 comes to know about view helper, the code is

public function getViewHelperConfig()

{

return array(

'factories' => array(

'test\_helper' => function($sm) {

$serviceLocator = $sm->getServiceLocator();

return new \Application\View\Helper\Testhelper($serviceLocator);

},

'routeHelper' => function($sm) {

$serviceLocator = $sm->getServiceLocator();

return new \Application\View\Helper\RouteHelper($serviceLocator);

}

)

);

}

//here we are sending service manager in the class and this received in the \_\_construct function of that class. Look at the Testhelper construct function carefully. we are receiving service manager there.

==================================================================================

common commands in Module.php

<?php

/\*\*

\* Zend Framework (http://framework.zend.com/)

\*

\* @link http://github.com/zendframework/ZendSkeletonApplication for the canonical source repository

\* @copyright Copyright (c) 2005-2015 Zend Technologies USA Inc. (http://www.zend.com)

\* @license http://framework.zend.com/license/new-bsd New BSD License

\*/

namespace Application;

use Zend\Mvc\ModuleRouteListener;

use Zend\Mvc\MvcEvent;

use Zend\Db\ResultSet\ResultSet;

use Zend\ModuleManager\Feature\AutoloaderProviderInterface;

use Zend\ModuleManager\Feature\ConfigProviderInterface;

use Application\Model\User;

use Application\Model\UserTable;

use Zend\Db\TableGateway\TableGateway;

use Zend\ModuleManager\Feature\ViewHelperProviderInterface;

class Module implements AutoloaderProviderInterface, ConfigProviderInterface, ViewHelperProviderInterface

{

public function onBootstrap(MvcEvent $e)

{

$eventManager = $e->getApplication()->getEventManager();

$eventManager->attach('route', array($this, 'aclConfiguration'), 2);

$sm = $e->getApplication()->getServiceManager();

$moduleRouteListener = new ModuleRouteListener();

$sharedManager = $eventManager->getSharedManager();

$moduleRouteListener->attach($eventManager);

}

public function getControllerPluginConfig() {

return array(

'invokables' => array(

'HelperAuth' => 'Application\Controller\Plugin\HelperAuth',

'CommonFunctions' => 'Application\Controller\Plugin\CommonFunctions',

'Acl2' => 'Application\Controller\Plugin\Acl2',

'Trainingcatelogallplace' => 'Application\Controller\Plugin\Trainingcatelogallplace',

),

);

}//there array tells us that we can define are plugin in module.php instead of modul.config.php. Choice is ours

//it is recommended to declare plugins in module.config.php

//module.php may consist code which may be required to talk to other modules.

public function getConfig()

{

return include \_\_DIR\_\_ . '/config/module.config.php';

}

public function getAutoloaderConfig()

{

return array(

'Zend\Loader\ClassMapAutoloader' => array(

\_\_DIR\_\_ . '/autoload\_classmap.php',

),

'Zend\Loader\StandardAutoloader' => array(

'namespaces' => array(

\_\_NAMESPACE\_\_ => \_\_DIR\_\_ . '/src/' . \_\_NAMESPACE\_\_,

),

),

);

}

public function getServiceConfig()

{

return array(

'factories' => array(

'Application\Model\UserTable' => function($sm) {

$tableGateway = $sm->get('ClassmasterGateway');

$table = new UserTable($tableGateway);

return $table;

},

'Application\Model\Jobmanagement\Jobmanagement'=>function($sm){

$tableGateway = $sm->get('ClassmasterGateway');

return new Model\Jobmanagement\Jobmanagement($tableGateway);

},

'ClassmasterGateway' => function($sm){

$dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');

return new TableGateway('company\_announcement', $dbAdapter, null);

},

),

);

}

public function bootstrapSession($e)

{

$session = $e->getApplication()

->getServiceManager()

->get('Zend\Session\SessionManager');

$session->start();

$container = new Container('initialized');

if (!isset($container->init)) {

$serviceManager = $e->getApplication()->getServiceManager();

$request = $serviceManager->get('Request');

$session->regenerateId(true);

$container->init = 1;

$container->remoteAddr = $request->getServer()->get('REMOTE\_ADDR');

$container->httpUserAgent = $request->getServer()->get('HTTP\_USER\_AGENT');

$config = $serviceManager->get('Config');

if (!isset($config['session'])) {

return;

}

$sessionConfig = $config['session'];

if (isset($sessionConfig['validators'])) {

$chain = $session->getValidatorChain();

foreach ($sessionConfig['validators'] as $validator) {

switch ($validator) {

case 'Zend\Session\Validator\HttpUserAgent':

$validator = new $validator($container->httpUserAgent);

break;

case 'Zend\Session\Validator\RemoteAddr':

$validator = new $validator($container->remoteAddr);

break;

default:

$validator = new $validator();

}

$chain->attach('session.validate', array($validator, 'isValid'));

}

}

}

}

public function aclConfiguration(MvcEvent $e)

{

$application = $e->getApplication();

$sm = $application->getServiceManager();

$sharedManager = $application->getEventManager()->getSharedManager();

$router = $sm->get('router');

$request = $sm->get('request');

$matchedRoute = $router->match($request);

$controller =$matchedRoute->getParam('controller');

if($controller=='token'){

$sharedManager->attach('Zend\Mvc\Controller\AbstractActionController','dispatch',function($e) use ($sm) {

$sm->get('ControllerPluginManager')->get('HelperAuth')->authenticate($e);

},2

);

}

if (null !== $matchedRoute) {

$sharedManager->attach('Zend\Mvc\Controller\AbstractActionController','dispatch',

function($e) use ($sm) {

$sm->get('ControllerPluginManager')->get('Acl2')

->doAuthorization($e); //pass to the plugin...

},2

);

}

}

public function getViewHelperConfig()

{

return array(

'factories' => array(

'test\_helper' => function($sm) {

$serviceLocator = $sm->getServiceLocator();

return new \Application\View\Helper\Testhelper($serviceLocator);

},

'routeHelper' => function($sm) {

$serviceLocator = $sm->getServiceLocator();

return new \Application\View\Helper\RouteHelper($serviceLocator);

}

)

);

}

}

=====================================================================================

common commands in module.config.php

<?php

/\*\*

\* Zend Framework (http://framework.zend.com/)

\*

\* @link http://github.com/zendframework/ZendSkeletonApplication for the canonical source repository

\* @copyright Copyright (c) 2005-2015 Zend Technologies USA Inc. (http://www.zend.com)

\* @license http://framework.zend.com/license/new-bsd New BSD License

\*/

return array(

'modules'=>array(

'Application',

),

'router' => array(

'routes' => array(

'hom' => array(

'type' => 'Zend\Mvc\Router\Http\Literal',

'options' => array(

'route' => '/',

'defaults' => array(

'controller' => 'Application\Controller\Index',

'action' => 'index',

),

),

),

'any\_cont\_act' => array( //use this route to redirect to any controller or action

'type' => 'Segment',

'options' => array(

'route' => '/[:controller[/:action]]',

'defaults' => array(

'\_\_NAMESPACE\_\_' => 'Application\Controller',

'controller' => 'Index',

'action' => 'index',

),

),

'may\_terminate' => true,

'child\_routes' => array(

'default' => array(

'type' => 'Segment',

'options' => array(

'route' => '/[:id1][/][:value1][/][:id2][/][:value2][/][:id3][/][:value3][/][:id4][/][:value4][/][:id5][/][:value5][/][:id6][/][:value6][/][:id7][/][:value7]',

// 'constraints' => array(

//'id' => '[a-zA-Z][a-zA-Z0-9\_-]\*',//defining rules for id, same can be done for others

//'more\_id' => '[a-zA-Z][a-zA-Z0-9\_-]\*',

// ),

'defaults' => array(

'id1' =>'',

'id2' =>'',

'id3' =>'',

'id4' =>'',

'id5' =>'',

'id6' =>'',

'id7' =>'',

'value1'=>'',

'value2'=>'',

'value3'=>'',

'value4'=>'',

'value5'=>'',

'value6'=>'',

'value7'=>'',

),

),

),

),

),

// The following is a route to simplify getting started creating

// new controllers and actions without needing to create a new

// module. Simply drop new controllers in, and you can access them

// using the path /application/:controller/:action

'application' => array(

'type' => 'Segment',

'options' => array(

'route' => '/application[/:controller[/:action]]',

'defaults' => array(

'\_\_NAMESPACE\_\_' => 'Application\Controller',

'controller' => 'Index',

'action' => 'index',

),

),

'may\_terminate' => true,

'child\_routes' => array(

'default' => array(

'type' => 'Segment',

'options' => array(

'route' => '[/:id][/:id\_value]',

'constraints' => array(

'id' => '[a-zA-Z][a-zA-Z0-9\_-]\*',

'id\_value' => '[a-zA-Z][a-zA-Z0-9\_-]\*',

),

'defaults' => array(

),

),

),

),

),

'pagination' => array(

'type' => 'Segment',

'options' => array(

'route' => '/[:controller[/:action]]',

'defaults' => array(

'\_\_NAMESPACE\_\_' => 'Application\Controller',

'controller' => 'Index',

'action' => 'index',

),

),

'may\_terminate' => true,

'child\_routes' => array(

'default' => array(

'type' => 'Segment',

'options' => array(

'route' => '/[:id1][/][:value1][/][:id2][/][:value2]',

'constraints' => array(

'id' => '[a-zA-Z][a-zA-Z0-9\_-]\*',

'id\_value' => '[a-zA-Z][a-zA-Z0-9\_-]\*',

),

'defaults' => array(

'id1' =>'',

'value1' =>'',

'id2' =>'',

'value2' =>''

),

),

),

),

),

),

),

'service\_manager' => array(

'abstract\_factories' => array(

'Zend\Cache\Service\StorageCacheAbstractServiceFactory',

'Zend\Log\LoggerAbstractServiceFactory',

),

'aliases' => array(

'translator' => 'MvcTranslator',

),

),

'translator' => array(

'locale' => 'en\_US',

'translation\_file\_patterns' => array(

array(

'type' => 'gettext',

'base\_dir' => \_\_DIR\_\_ . '/../language',

'pattern' => '%s.mo',

),

),

),

'controllers' => array(

'invokables' => array(

'Application\Controller\Index' => 'Application\Controller\IndexController',

'Application\Controller\Trainingcatalog' => 'Application\Controller\TrainingcatalogController',

),

),

'controller\_plugins' => array(

'invokables' => array(

'HelperAuth' => 'Application\Controller\Plugin\HelperAuth',

)

),

'view\_manager' => array(

'display\_not\_found\_reason' => true,

'display\_exceptions' => true,

'doctype' => 'HTML5',

'not\_found\_template' => 'error/404',

'exception\_template' => 'error/index',

'template\_map' => array(

'layout/layout' => \_\_DIR\_\_ . '/../view/layout/layout.phtml',

'adminheader' => \_\_DIR\_\_ . '/../view/layout/scripts/adminheader.phtml',

'standard/adminheader' => \_\_DIR\_\_ . '/../view/layout/scripts/standard/adminheader.phtml',

'header' => \_\_DIR\_\_ . '/../view/layout/scripts/header.phtml',

'leftnav' => \_\_DIR\_\_ . '/../view/layout/scripts/leftnav.phtml',

'footer' => \_\_DIR\_\_ . '/../view/layout/scripts/footer.phtml',

'application/index/index' => \_\_DIR\_\_ . '/../view/application/index/index.phtml',

'error/404' => \_\_DIR\_\_ . '/../view/error/404.phtml',

'error/index' => \_\_DIR\_\_ . '/../view/error/index.phtml',

),

'template\_path\_stack' => array(

'application'=> \_\_DIR\_\_ . '/../view',

),

),

'service\_manager'=>array(

'initializers' => array(

function ($instance, $sm) {

if ($instance instanceof \Zend\Db\Adapter\AdapterAwareInterface) {

$instance->setDbAdapter($sm->get('Zend\Db\Adapter\Adapter'));

}

}

),

// 'invokables' => array(

// 'Application\Model\UserTable' => 'Application\Model\UserTable'

// )

),

);

=====================================================================================

common commands in application.config.php

return array(

// This should be an array of module namespaces used in the application.

'modules' => array(

'Application',

//more modules will come here

),

==================================================================================

common commands in global.php and local.php

global.php

return array(

'db' => array(

'driver' => 'Pdo',

'dsn' => 'mysql:dbname=beta;host=localhost',

'driver\_options' => array(

PDO::MYSQL\_ATTR\_INIT\_COMMAND => 'SET NAMES \'UTF8\''

),

),

'service\_manager' => array(

'factories' => array(

'Zend\Db\Adapter\Adapter'

=> 'Zend\Db\Adapter\AdapterServiceFactory',

),

),

'session' => array(

'config' => array(

'class' => 'Zend\Session\Config\SessionConfig',

'options' => array(

'name' => 'zend2design',

),

),

'storage' => 'Zend\Session\Storage\SessionArrayStorage',

'validators' => array(

'Zend\Session\Validator\RemoteAddr',

'Zend\Session\Validator\HttpUserAgent',

),

),

);

local.php

return array(

'db' => array(

'username' => 'root',

'password' => '',

),

);

==================================================================================

common commands in layout

$this->content;

$this->partial("header"); //header is an alias which we created in module.config.php

$this->partial("footer"); //footer is an alias which we created in module.config.php

$this->headTitle('Pinnacle Dealer Academy');ss