**ПРИЛОЖЕНИЕ Ж**

Листинг программных компонентов

**Клиентский скрипт отображающий панорамы и взаимодействующий с сервером**

(function($) {

$.fn.panorama = function(params) {

this.\_container = $(this);

this.cfg = {

width: 800,

height: 600,

startid: null,

host: '',

nopoints: false,

path: '/panorama',

click: null,

nocache: false,

nocontrolls: false,

locale: {}

};

for (var i in params) {

this.cfg[i] = params[i];

};

this.\_setSize = function(obj) {

obj.attr('width', this.cfg.width).attr('height', this.cfg.height).css({width: this.cfg.width, height: this.cfg.height});

};

this.\_prepareEvents = function() {

this.\_addStartDrag();

this.\_addStopDrag();

this.\_addOnMove();

this.\_addClick();

};

this.\_addStartDrag = function() {

var obj = this;

var handler = function(e) {

if (e.targetTouches && e.targetTouches.length > 0) {

var touch = event.targetTouches[0];

obj.lastX = touch.pageX;

} else {

obj.lastX = e.offsetX;

}

obj.drag = true;

return false;

};

this.canvas.bind('mousedown touchstart', handler);

};

this.\_addStopDrag = function() {

var obj = this;

var handler = function(e) {

if (e.targetTouches && e.targetTouches.length > 0) {

var touch = event.targetTouches[0];

obj.lastX = touch.pageX;

} else {

obj.lastX = e.offsetX;

}

obj.drag = false;

return false;

};

this.canvas.bind('mouseleave mouseup mouseout touchend', handler);

};

this.\_addOnMove = function() {

var obj = this;

var handler = function(e) {

if (obj.drag) {

var p = 0;

if (e.targetTouches && e.targetTouches.length > 0) {

var touch = event.targetTouches[0];

p = touch.pageX;

} else {

p = e.offsetX;

}

if (obj.lastX < p) {

obj.angle -= p - obj.lastX;

}

else {

obj.angle += obj.lastX - p;

}

if (obj.angle <= 0) {

obj.angle += 360;

} else if (obj.angle > 360) {

obj.angle -= 360;

}

obj.lastX = p;

obj.draw();

}

return false;

};

this.canvas.bind('mousemove touchmove', handler);

};

this.\_addClick = function() {

var obj = this;

if(typeof(obj.cfg.click) == 'function'){

var handler = function(e) {

var cx = Math.floor(obj.pWidth / 360 \* obj.angle);

var half = obj.cfg.width / 2 / obj.k;

cx = cx + Math.floor(e.offsetX / obj.k) - half;

if (cx > obj.pWidth) {

cx -= obj.pWidth;

} else if (cx < 0) {

cx += obj.pWidth;

}

var c = {x: Math.floor(cx / obj.pk), y: Math.floor(e.offsetY / obj.pk)}

obj.cfg.click(c);

};

this.canvas.bind('click', handler);

}

};

this.loadImage = function(src) {

this.spinner(true);

this.img = new Image();

this.img.src = this.cfg.host + src;

var obj = this;

this.img.onload = function() {

obj.pWidth = obj.img.width;

obj.pHeight = obj.img.height;

obj.k = obj.cfg.height / obj.pHeight;

obj.pk = obj.cfg.height / obj.oHeight;

obj.spinner();

obj.draw();

};

};

this.draw = function() {

var half = Math.floor(this.cfg.width / 2 / this.k);

var x1 = 0,

x2 = 0,

l1 = 0,

l2 = 0;

var p = Math.floor(this.pWidth / 360 \* this.angle);

if (p + half > this.pWidth) {

x1 = p - half;

l1 = this.pWidth - x1;

x2 = half - (this.pWidth - p);

l2 = x2;

var wl = l1 \* this.k;

this.context.drawImage(this.img, x1, 0, l1, this.pHeight, 0, 0, wl, this.cfg.height);

this.context.drawImage(this.img, 0, 0, l2, this.pHeight, wl, 0, l2 \* this.k, this.cfg.height);

} else if (p - half < 0) {

l1 = half - p;

x1 = this.pWidth - l1;

x2 = p + half;

l2 = x2;

var wl = l1 \* this.k;

this.context.drawImage(this.img, x1, 0, l1, this.pHeight, 0, 0, wl, this.cfg.height);

this.context.drawImage(this.img, 0, 0, l2, this.pHeight, wl, 0, l2 \* this.k, this.cfg.height);

} else {

this.context.drawImage(this.img, p - half, 0, this.cfg.width / this.k, this.pHeight, 0, 0, this.cfg.width, this.cfg.height);

}

if (this.marks != null && this.marks.length > 0) {

for(var i in this.marks) {

var ppp = this.marks[i];

var px = Math.floor(parseInt(this.marks[i].x) \* this.pk);

var h = parseInt(this.marks\_obj[i].css('width')) / 2;

var d = false;

if (px - h < 0) {

px = this.pWidth + px - h;

} else {

px -= h;

}

if (p - half < 0) {

if (px > x1 && px <= this.pWidth) {

px -= x1;

d = true;

} else if (px >= 0 && px < x2) {

px += l1;

d = true;

}

} else if (p + half > this.pWidth) {

if (px > x1 && px <= this.pWidth) {

px -= x1;

d = true;

} else if (px >= 0 && px <= x2) {

px += l1;

d = true;

}

} else if (px > p - half && px < p + half) {

px -= Math.abs(p - half);

d = true;

}

if (d) {

px = Math.floor(this.k \* px);

if (px + parseInt(this.marks\_obj[i].css('width')) <= this.cfg.width) {

var y = Math.floor(this.marks[i].y \* this.pk);

if(this.marks[i].index){

y += this.marks[i].index \* 35;

}

this.marks\_obj[i].css({top: y, left: px, display: 'block'});

d = false;

continue;

}

}

this.marks\_obj[i].css({top: 0, left: 0, display: 'none'});

}

}

};

this.spinner = function(show) {

if (this.\_spinner && show) {

this.\_spinner.css({height: this.cfg.height, display: 'block'});

} else {

this.\_spinner.css({display: 'none'});

}

};

this.view = function(src, angle, points) {

this.angle = this.offset = angle;

this.loadImage(src);

for (var i = 0; i < this.marks\_obj.length; i++) {

this.marks\_obj[i].remove();

delete this.marks\_obj[i];

}

this.marks\_obj = [];

var marks = [];

var index = 0;

var p = [];

for (var i = 0; i < points.length; i++) {

if(points[i].rtype <= 1){

var element = this.\_markHtml(points[i]);

p[index] = points[i];

this.marks\_obj[index] = element;

this.container.append(element);

index++;

} else if(points[i].rtype == 2){

if(points[i].data.length > 0){

var element = this.\_markHtml(points[i]);

p[index] = points[i];

this.marks\_obj[index] = element;

this.container.append(element);

index++;

for(var j = 0; j < points[i].data.length; j++){

var mark = points[i].data[j];

if(mark.to == this.current\_point){

continue;

}

mark.x = points[i].x;

mark.y = points[i].y;

mark.index = j + 1;

var element = this.\_markHtml(mark);

p[index] = mark;

this.marks\_obj[index] = element;

this.container.append(element);

index++;

}

}

}

}

this.marks = p;

if(this.\_route){

this.route(true);

}

this.draw();

};

this.\_markHtml = function(mark){

var obj = this;

var element = $('<div/>', {'class': 'point pano-type-' + mark.type});

var label = $('<div/>', {'class': 'pano-label'});

var img = $('<img/>', {'src': mark.icon, 'class': 'point-icon'});

var name = $('<span/>').text(mark.name);

label.append(img).append(name);

element.append(label);

if (mark.type == 1 && mark.rtype != 2) {

var f = function(topoint) {

var handler = function(e) {

obj.load(topoint);

};

label.bind('click', handler);

};

f(mark.to);

}

if (mark.description.length > 0) {

var toggle = $('<span/>', {'class': 'point-togle-desc'}).html('&#9660;');

label.append(toggle);

var description = $('<div/>', {'class': 'point-desc'}).html(mark.description);

element.append(description);

var f = function(d, t) {

t.bind('click', function(e) {

e.stopPropagation();

d.toggle();

});

};

f(description, toggle);

}

return element;

};

this.\_ajax = function(action, param, callback) {

var obj = this;

$.ajax({

type: 'post',

url: obj.\_url(action) + '/',

data: param,

success: function(result) {

if (typeof (callback) == 'function') {

callback(result);

}

}

});

};

this.\_url = function() {

var url = this.cfg.host + this.cfg.path + '/';

for (var i = 0; i < arguments.length; i++) {

url += arguments[i];

if (i + 1 != arguments.length) {

url += '/';

}

}

return url;

}

this.load = function(id) {

if (typeof(id) == 'undefined') {

id = parseInt(this.cfg.startid);

}

this.current\_point = id;

var obj = this,p;

if (p = this.getCache('points',id)) {

obj.oHeight = p[0][2];

obj.view(p[0][0], p[0][1], p[1]);

} else {

this.spinner(true);

this.\_ajax('point', {'id': id, height: this.cfg.height}, function(data) {

if (data.length != 0) {

var p = JSON.parse(data);

if (p.length > 0) {

if (obj.cfg.nopoints == true) {

p[1] = [];

}

obj.spinner(false);

obj.oHeight = p[0][2];

obj.view(p[0][0], p[0][1], p[1]);

obj.setCache('points',id,p);

}

}

});

}

};

this.\_ = function(text){

if(typeof(this.cfg.locale[text]) == 'string'){

text = this.cfg.locale[text];

}

return text;

};

this.searchResults = function(show){

var sr = $('#panoram-search-result');

if(show){

sr.show();

} else {

sr.hide();

}

};

this.route = function(show){

var sr = $('.panoram-controls-route');

if(show){

this.\_route = true;

sr.show();

var la = $('.panoram-route-row-active');

la.removeClass('panoram-route-row-active');

if(this.\_route\_obj[this.current\_point]){

this.\_route\_obj[this.current\_point].addClass('panoram-route-row-active');

}

var item = this.\_route\_items[this.\_route\_end];

if(item && item.id != this.current\_point){

for(var i in this.marks){

if(this.\_route\_items[this.marks[i].to]){

this.marks\_obj[i].addClass('pano-route');

}

}

}

} else {

this.\_route = false;

$('.pano-route').each(function(){

$(this).removeClass('pano-route');

});

sr.hide();

}

};

this.getRoute = function(point){

var obj = this;

obj.spinner(true);

obj.searchResults(false);

var routeContainer = $('.panoram-route-container');

routeContainer.empty();

var f = function(data) {

if (data.length != 0) {

var rc = JSON.parse(data);

var bctp = function(item,id){

item.bind('click',function(){

obj.load(id);

});

};

if (rc.length > 0) {

obj.\_route\_obj = [];

obj.\_route\_items = [];

for(var key in rc){

var row = $('<div/>',{'class':'panoram-route-row'})

.append($('<div/>',{'class':'panoram-route-name'}).text(rc[key].name))

routeContainer.append(row);

obj.\_route\_obj[rc[key].id] = row;

obj.\_route\_items[rc[key].id] = rc[key];

if(rc[key].type == 1){

bctp(row,rc[key].id);

}

obj.\_route\_end = rc[key].id;

}

}

obj.route(true);

obj.spinner(false);

}

};

if(this.getCache('route',obj.current\_point + '\_' + point)){

f(this.getCache('route',obj.current\_point + '\_' + point));

} else {

this.\_ajax('path', {from:obj.current\_point,to:point}, function(data){

f(data);

obj.setCache('route',obj.current\_point + '\_' + point, data);

});

}

};

this.getCache = function(type,id){

if(this.cfg.nocache){

return null;

}

if(this.cache[type] && this.cache[type][id]){

return this.cache[type][id];

}

return null;

};

this.setCache = function(type,id,value){

if(typeof(this.cache[type]) == 'undefined'){

this.cache[type] = [];

}

this.cache[type][id] = value;

};

this.search = function(q,param){

if(q.length > 0){

var obj = this;

this.spinner(true);

var f = function(data){

if (data.length != 0) {

var resultContainer = $('.panoram-search-result-container');

resultContainer.empty();

var rc = JSON.parse(data);

if (rc.length > 0) {

var bgr = function(item,id){

item.bind('click',function(){

obj.getRoute(id);

});

};

var ssp = function (item,id) {

item.bind('click',function(){

obj.load(id);

});

};

for(var key in rc){

if(rc[key].id == obj.current\_point) continue;

var srp, srn;

var row = $('<div/>',{'class':'panoram-search-result-row'})

.append( srn = $('<div/>',{'class':'panoram-search-result-name'}).text(rc[key].name))

.append(srp = $('<span/>',{'class':'panoram-search-result-path'}).html('&#9658'));

resultContainer.append(row);

ssp(srn,rc[key].id);

bgr(srp,rc[key].id);

}

}

}

obj.searchResults(true);

obj.spinner(false);

};

if(this.getCache('search',q+'\_'+param)){

f(this.getCache('search',q+'\_'+param));

} else {

this.\_ajax('search', {q:q,param:param}, function(data){

f(data);

obj.setCache('search',q+'\_'+param,data);

});

}

}

};

this.\_addSpinner = function(){

var sl = $('<div/>', {'class': 'panoram-spinner'});

this.\_setSize(sl);

var ol = $('<div/>', {'class': 'panoram-overlay'});

this.\_setSize(ol);

this.\_spinner = $('<div/>', {'class': 'panoram-spinner-container'});

this.\_spinner.append(ol).append(sl);

this.container.append(this.\_spinner);

};

this.\_init = function() {

this.canvas = $('<canvas/>', {id: 'panorama', 'class': 'panoram-container'}).text('Your browser does not support the canvas element');

this.\_setSize(this.canvas);

this.container = $('<div/>', {'class': 'panoram-items-container', id: 'panoram-items-container'});

this.\_setSize(this.container);

this.container.append(this.canvas);

this.\_container.append(this.container);

if($.browser.msie && $.browser.version < 9 && G\_vmlCanvasManager) {

G\_vmlCanvasManager.initElement(this.canvas[0]);

}

this.context = this.canvas[0].getContext('2d');

this.\_prepareEvents();

if(!this.cfg.nocontrolls){

var cc = $('<div/>',{'class': 'panoram-controls-container panoram-controls-container-s'});

cc.append($('<div/>',{'class': 'panoram-controls-bg',id:'panoram-controls-bg','height':this.cfg.height}));

var sb,srh,rh,src,srhc,rhc;

var cf = $('<div/>',{'class': 'panoram-controls-search'})

.append($('<input/>',{type:'text',id:'panoram-search','class':'panoram-search'}))

.append($('<select/>',{id:'panoram-search-select','class':'panoram-search-select'}))

.append(sb = $('<button/>',{id:'panoram-search-button','class':'panoram-search-button'}).text(this.\_('Search')))

.append($('<div/>',{id:'panoram-search-result','class':'panoram-search-result',style:'display:none'})

.append(srh = $('<div/>',{'class':'panoram-search-result-header'}).text(this.\_('Search results:')))

.append(src = $('<div/>',{'class':'panoram-search-result-container'}))

);

var cr = $('<div/>',{'class': 'panoram-controls-route',style:'display:none'})

.append(rh = $('<div/>',{'class':'panoram-route-header'}).text(this.\_('Route:')))

.append($('<div/>',{'class':'panoram-route-container'}));

cc.append(cf);

cc.append(cr);

var obj = this;

srh.append(srhc = $('<span/>').text('x'));

rh.append(rhc = $('<span/>').text('x'));

src.css({'max-height':this.cfg.height - 84});

srhc.bind('click',function(){obj.searchResults(false)});

rhc.bind('click',function(){obj.route(false)});

this.container.append(cc);

this.\_addSpinner();

this.\_ajax('getmtypes', {}, function(data) {

if (data.length != 0) {

var select = $('#panoram-search-select');

select.append($('<option/>'));

var p = JSON.parse(data);

if (p.length > 0) {

for(var key in p){

select.append($('<option/>',{id:p[key].id}).text(p[key].name));

}

}

}

});

sb.bind('click',function(){

var param = $('#panoram-search-select').val();

var q = $('#panoram-search').val();

obj.route(false);

obj.search(q,param);

return false;

});

cc.bind('click',function(e){

$(this).toggleClass('panoram-controls-container-s');

});

var f = function(e){

e.stopPropagation();

};

cf.bind('click',f);

cr.bind('click',f);

} else {

this.\_addSpinner();

}

this.load();

};

this.cache = [];

this.\_route\_obj = [];

this.marks\_obj = [];

this.\_init();

};

})(jQuery);

**Класс Linker**

class Linker

{

protected static $\_obj = array();

public function \_\_construct() {

$this->loadParts();

}

protected function loadParts() {

$parts = $this->findParts();

usort($parts,array('Linker','\_compareNames'));

$parts = array\_merge($parts,$this->findParts('app/model'));

foreach($parts as $item) {

include\_once($item);

}

}

protected function findParts($path = 'app/core') {

$parts = array();

$handle = opendir($path);

while (false !== ($file = readdir($handle))) {

if ($file != '.' && $file != '..') {

$file = $path . '/' . $file;

if (is\_file($file)) {

$parts[] = $file;

} else {

$parts = array\_merge($parts, $this->findParts($file));

}

}

}

closedir($handle);

return $parts;

}

public static function \_compareNames($name1,$name2){

$a1 = explode('/',$name1);

$a2 = explode('/',$name2);

$c1 = count($a1);

$c2 = count($a2);

if($c1 == $c2){

return !strcmp(end($a1),end($a2));

} else {

return ($c1 > $c2) ? -1 : 1;

}

}

public static function getClass($name){

$class = null;

$name[0] = strtoupper($name[0]);

if(class\_exists($name)){

$class = new $name();

}

return $class;

}

public static function getObject($name){

if(!isset(self::$\_obj[$name])){

self::$\_obj[$name] = self::getClass($name);

}

return self::$\_obj[$name];

}

public static function registerObject($obj){

self::$\_obj[get\_class($obj)] = $obj;

return $obj;

}

}

**Класс Map**

final class Map extends Linker

{

public function \_\_construct() {

parent::\_\_construct();

$this->registerObject(new SystemConfig());

$this->registerObject(new Session());

$this->registerObject(new DataBase());

$front = new Front();

$front->process();

unset($front);

}

public static function getRootPath() {

return $\_SERVER['DOCUMENT\_ROOT'];

}

public static function L() {

$locale = self::getObject('Locale');

$args = func\_get\_args();

$args = array\_map(array($locale, 'L'), $args);

return implode(' ', $args);

}

}

**Класс DataBase**

class DataBase

{

protected $\_connection = null;

protected $\_errors = array();

public function \_\_construct()

{

}

public function connect(){

if(is\_null($this->\_connection)){

$config = Map::getObject('SystemConfig');

try {

$this->\_connection = new PDO('mysql:host='.$config->db\_host.':'.$config->db\_port.';dbname='.$config->db\_name, $config->db\_user, $config->db\_pswd);

$this->\_connection->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} catch(PDOException $ex){

$this->addError($ex);

}

}

}

public function escape($var){

if(is\_array($var)){

array\_map(array($this,'escape'), $var);

} else {

$var = mysql\_real\_escape\_string($var);

}

return $var;

}

public function query($query,$objName = null){

if(strlen($query) > 0){

$result = null;

try {

$this->connect();

$result = $this->\_connection->query($query);

} catch(PDOException $ex){

$this->addError($ex);

}

if($result){

if(is\_null($objName)){

$result->setFetchMode(PDO::FETCH\_ASSOC);

} else {

$result->setFetchMode(PDO::FETCH\_CLASS, $objName);

}

if($result->rowCount() > 1){

$rc = array();

while($row = $result->fetch()){

$rc[] = $row;

}

return $rc;

} else {

$rc = $result->fetch();

if(!$rc) $rc = array();

return $rc;

}

}

}

return null;

}

public function insert($query){

try{

$this->connect();

$result = $this->\_connection->exec($query);

if($result > 0){

return $this->\_connection->lastInsertId();

}

}

catch (PDOException $ex){

$this->addError($ex);

}

return -1;

}

public function update($query){

try{

$this->connect();

$result = $this->\_connection->exec($query);

if($result >= 0){

return true;

}

}

catch (PDOException $ex){

$this->addError($ex);

}

return -1;

}

public function delete($query){

try{

$this->connect();

$result = $this->\_connection->exec($query);

if($result >= 0){

return true;

}

}

catch (PDOException $ex){

$this->addError($ex);

}

return -1;

}

public function beginTransaction(){

$rc = false;

try{

$this->connect();

$rc = $this->\_connection->beginTransaction();

}

catch(PDOException $ex){

$this->addError($ex);

}

return $rc;

}

protected function addError($exception){

$this->\_errors[] = $exception;

}

public function commit(){

$rc = false;

try{

$rc = $this->\_connection->commit();

}

catch(PDOException $ex){

$this->addError($ex);

}

return $rc;

}

public function rollback(){

$rc = false;

try{

$rc = $this->\_connection->rollback();

}

catch(PDOException $ex){

$this->addError($ex);

}

return $rc;

}

public function getErrors(){

return $this->\_errors;

}

}

**Класс Object**

abstract class Object

{

protected $\_data = array();

public function \_\_construct()

{

}

public function \_\_get($name)

{

if(self::\_\_isset($name)) {

return $this->\_data[$name];

} else {

return null;

}

}

public function \_\_set($name, $value)

{

$this->\_data[$name] = $value;

}

public function \_\_isset($name)

{

return isset($this->\_data[$name]);

}

public function getData(){

return $this->\_data;

}

public function setData($data){

$this->\_data = $data;

}

}

**Класс Model**

class Model extends Object

{

protected $\_columns = array();

protected $\_data = array();

protected $\_types = array('text','int','email','bool','time','password','image','hidden');

protected $\_isValid = false;

protected $\_errors = array();

protected $\_query = null;

protected $\_tableName = '';

public function \_\_construct() {

$this->\_query = new Query();

if(empty($this->\_tableName)){

$this->\_tableName = strtolower(get\_class($this));

}

}

protected function addColumn($column){

if(is\_array($column)){

if(isset($column['name']) && !empty($column['name'])){

$col = array('name' => $column['name']);

if(isset($column['length']) && is\_numeric($column['length'])){

$col['length'] = $column['length'];

} else {

$col['length'] = 0;

}

if(isset($column['minlength']) && is\_numeric($column['minlength'])){

$col['minlength'] = $column['minlength'];

} else {

$col['minlength'] = 0;

}

if(isset($column['required']) && (bool)$column['required']){

$col['required'] = 1;

} else {

$col['required'] = 0;

}

if(isset($column['type']) && !empty($column['type']) && in\_array($column['type'], $this->\_types)){

$col['type'] = $column['type'];

} else {

$col['type'] = '';

}

if(isset($column['title']) && !empty($column['title'])){

$col['title'] = $column['title'];

} else {

$col['title'] = $column['name'];

}

if(isset($column['default']) && strlen($column['default']) > 0){

$col['default'] = $column['default'];

$this->$column['name'] = $column['default'];

} else {

$col['default'] = '';

}

if(isset($column['select']) && is\_object($column['select'])){

$col['select'] = $column['select'];

} else {

$col['select'] = null;

}

$this->\_columns[$column['name']] = $col;

}

}

}

protected function addColumns($array){

foreach ($array as $item){

$this->addColumn($item);

}

}

public function load($param = null){

if(is\_numeric($param) && $param >= 0){

return $this->loadById($param);

} else if(is\_array ($param) && count($param) > 0){

return $this->loadByAttr($param);

} else if(is\_null($param)){

return $this->loadAll();

}

return null;

}

protected function loadById($id){

$this->setData(Map::getObject('DataBase')->query($this->\_query->select(array\_keys($this->\_columns),$this->\_tableName,array(key($this->\_columns),'=',$id))->getQuery()));

return $this;

}

protected function loadByAttr($attr){

$collection = array();

if(is\_array($attr)){

$where = array();

foreach ($attr as $key => $item) {

if(is\_array($item)){

$where[] = array($key,'IN',$item);

}

else {

$where[] = array($key,'=',$item);

}

}

$data = Map::getObject('DataBase')->query($this->\_query->select(array\_keys($this->\_columns),$this->\_tableName,$where)->getQuery());

if(count($data) == 0){

return $this;

}

if(isset($data[key($this->\_columns)])){

$this->prepareData($data);

$collection[] = $this;

} else {

foreach ($data as $row){

$item = clone $this;

$item->setData($row);

$collection[] = $item;

}

}

}

return $collection;

}

protected function loadAll(){

$collection = array();

$query = $this->\_query->select(array\_keys($this->\_columns),$this->\_tableName);

$data = Map::getObject('DataBase')->query($query->getQuery());

if(count($data) == 0){

return $this;

}

if(isset($data[key($this->\_columns)])){

$this->prepareData($data);

$collection[] = $this;

} else {

foreach ($data as $row){

$item = clone $this;

$item->setData($row);

$collection[] = $item;

}

}

return $collection;

}

public function prepareJoins($query){

}

public function range($from = 0, $to = 20){

$data = Map::getObject('DataBase')->query($this->\_query->select(array\_keys($this->\_columns),$this->\_tableName)->limit($from,$to)->getQuery());

$collection = array();

if(isset($data[key($this->\_columns)])){

$this->setData($data);

$collection[] = $this;

} else {

foreach ($data as $row){

$item = clone $this;

$item->setData($row);

$collection[] = $item;

}

}

return $collection;

}

public function count($attr = null){

$count = 0;

if(is\_array ($attr) && count($attr) > 0){

$where = array();

foreach ($attr as $key => $item) {

$where[] = array($key,'=',$item);

}

$rc = array\_values(Map::getObject('DataBase')->query($this->\_query->count($this->\_tableName,$where)->getQuery()));

$count = array\_shift($rc);

} else if(is\_null($attr)){

$rc = array\_values(Map::getObject('DataBase')->query($this->\_query->count($this->\_tableName)->getQuery()));

$count = array\_shift($rc);

}

return $count;

}

public function getColumns(){

return $this->\_columns;

}

public function prepareData($data){

if(is\_array($data)){

$columns = array\_keys($this->\_columns);

foreach($data as $key => $item){

if(in\_array($key, $columns)){

$this->$key = $item;

}

}

}

return $this;

}

public function validate(){

$valid = true;

$primary = key($this->\_columns);

/\*if(!is\_numeric($this->$primary)){

$primary = null;

}\*/

foreach ($this->\_columns as $key => $column) {

if($key != $primary){

$value = $this->\_\_get($column['name']);

if($column['required'] && empty($value)){

$this->addError($column['name'], sprintf(Map::L('%s is required!'),$column['title']));

} else {

$len = strlen($value);

$error = '';

if((!$column['required'] && $len > 0) && ($column['minlength'] > 0 && $len < $column['minlength'])){

$error .= sprintf(Map::L('Minimum length is %d.'),$column['minlength']);

}

if($column['length'] > 0 && $len > $column['length']){

$error .= sprintf(Map::L('Maximum length is %d.'),$column['length']);

}

if($error != ''){

$this->addError($column['name'], $error);

$valid = false;

break;

}

switch ($column['type']){

case 'text':{

if(!is\_string($value)){

$this->addError($column['name'], sprintf(Map::L('%s must be a string!'),$column['title']));

$valid = false;

}

break;

}

case 'int':{

if((int)$value != $value && !filter\_var($value, FILTER\_VALIDATE\_INT) && $value !== 0){

$this->addError($column['name'], sprintf(Map::L('%s must be a integer!'),$column['title']));

$valid = false;

}

break;

}

case 'email':{

if(!filter\_var($value,FILTER\_VALIDATE\_EMAIL)){

$this->addError($column['name'], sprintf(Map::L('invalid email!'),$column['title']));

$valid = false;

}

$valid = true;

break;

}

case 'bool':{

break;

}

case 'image':{

if(!is\_string($value)){

$this->addError($column['name'], sprintf(Map::L('%s must be a string!'),$column['title']));

$valid = false;

}

break;

}

}

}

}

}

return $valid;

}

public function addError($column, $msg){

if(!isset($this->\_errors[$column]) || !is\_array($this->\_errors[$column])){

$this->\_errors[$column] = array();

}

$this->\_errors[$column][] = $msg;

}

public function getErrors(){

return $this->\_errors;

}

public function save(){

$rc = $this->validate();

if($rc){

$data = $this->getData();

$keys = array\_keys($this->\_columns);

$primary = $keys[0];

unset($data[$primary]);

if(is\_numeric($this->$primary) && $this->$primary > 0){ // update

$result = Map::getObject('DataBase')->update($this->\_query->update($data,$this->\_tableName,array($primary,'=',$this->$primary))->getQuery());

} else { // insert

$result = Map::getObject('DataBase')->insert($this->\_query->insert($data,$this->\_tableName)->getQuery());

if($result > 0){

$this->$primary = $result;

$rc = true;

} else {

$rc = false;

}

}

}

return $rc;

}

public function delete($ids = null){

$rc = false;

$keys = array\_keys($this->\_columns);

$primary = $keys[0];

$result = 0;

if(is\_null($ids)){

if(is\_numeric($this->$primary) && $this->$primary > 0){

$result = Map::getObject('DataBase')->delete($this->\_query->delete($this->\_tableName,array($primary,'=',$this->$primary))->getQuery());

}

} else {

$result = Map::getObject('DataBase')->delete($this->\_query->delete($this->\_tableName,array($primary,'IN',$ids))->getQuery());

}

if($result > 0){

$rc = true;

}

return $rc;

}

}

**Класс Controller**

abstract class Controller

{

protected $\_notify = array();

protected $\_protected = false;

public function \_\_construct() {

}

protected function addNotification($type, $msg){

if($type == 'error' || $type == 'success' || $type = 'warning'){

$this->\_notify[] = array('type' => $type, 'message' => $msg);

}

}

public function getNotifications(){

return $this->\_notify;

}

protected function redirect($url, $code = 301){

Map::getObject('Session')->notifications = $this->getNotifications();

@header('Location: '.$url, true, $code);

die();

}

public function isProtected(){

return $this->\_protected;

}

}

**Класс PageView**

abstract class PageView

{

protected $\_data = array();

protected $\_layout = 'app/view/\_layout.html';

protected $\_partialPath = 'app/view/partial/';

protected $\_templateExt = 'html';

public function \_\_construct($data = null,$view = null){

$this->\_data['data'] = $data;

$this->\_data['scripts'] = array();

$this->\_data['rscripts'] = array();

$this->\_data['css'] = array();

$this->\_data['title'] = 'Empty page';

$this->\_data['content\_type'] = 'text/html';

$this->\_data['view'] = $view;

$this->\_data['body\_attr'] = array();

$this->\_data['notify'] = array();

$this->\_data['meta'] = array('<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />');

}

protected function generateHeader(){

$header = "<!DOCTYPE html>\n<html>\n<head>\n<title>".$this->getTitle()."</title>".PHP\_EOL;

$header.=$this->generateMeta();

$header.=$this->generateCSS();

$header.=$this->generateScript();

$header.="</head>\n";

return $header;

}

protected function generateMeta(){

return implode(PHP\_EOL, $this->\_data['meta']).PHP\_EOL;

}

protected function generateScript(){

$script = '';

foreach($this->\_data['scripts'] as $s){

if(is\_array($s)){

$script.='<!--[if '.$s[1].']><script src="/scripts/'.$s[0].'" type="text/javascript"></script><![endif]-->'.PHP\_EOL;

} else {

$script.='<script src="/scripts/'.$s.'" type="text/javascript"></script>'.PHP\_EOL;

}

}

if(count($this->\_data['rscripts']) > 0){

$r = '<script type="text/javascript">$(document).ready(function(){'.PHP\_EOL;

$r.=implode(PHP\_EOL, $this->\_data['rscripts']);

$t.='});'.PHP\_EOL;

}

return $script;

}

protected function generateCSS(){

$css = '';

foreach($this->\_data['css'] as $c){

$css.='<link href="/styles/'.$c.'" rel="stylesheet" type="text/css" />'.PHP\_EOL;

}

return $css;

}

protected function generateFooter(){

return $this->renderBlock($this->partial('footer',false));

}

protected function generateBody(){

$layout = $this->getLayout();

$body = "<body";

if(count($this->\_data['body\_attr']) > 0){

foreach($this->\_data['body\_attr'] as $key => $attr){

$body .= sprintf(' %s="%s"',$key,$attr);

}

}

$body.=">\n";

$body.=$layout.$this->generateFooter();

$body.="</body>\n";

return $body;

}

protected function getLayout(){

return $this->renderBlock($this->\_layout);

}

public function body(){

$path = 'app/view/';

if($this->\_data['view'] == null){

$request = Map::getObject('Request');

if(!empty($request->area)){

$path .= $request->area.'/';

}

$path .= $request->controller.'/'.$this->prepareExt($request->action);

} else {

$path .= $this->prepareExt($this->\_data['view']);

}

return $this->renderBlock($path);

}

protected function renderBlock($path){

$block = '';

if(file\_exists($path)){

ob\_start();

require($path);

$block = ob\_get\_contents();

ob\_end\_clean();

}

return $block;

}

public function setLayout($file){

$path = 'app/view/'.$file;

if(file\_exists($path)){

$this->\_layout = $path;

}

}

public function getData(){

return $this->\_data['data'];

}

public function setData($data){

$this->\_data['data'] = $data;

}

public function getTitle(){

return $this->\_data['title'];

}

public function setTitle($title){

$this->\_data['title'] = $title;

}

public function render(){

$body = $this->generateBody();

$response = $this->generateHeader().$body;

echo $response;

}

public function addScript($path, $if = null) {

if (is\_null($if)) {

$this->\_data['scripts'][] = $path;

} else {

$this->\_data['scripts'][] = array($path, $if);

}

}

public function addInitScript($script){

$this->\_data['rscripts'][] = $script;

}

public function addCSS($path){

$this->\_data['css'][] = $path;

}

public function addBodyAttr($attr = array()){

if(is\_array($attr)){

$this->\_data['body\_attr'] = array\_merge($this->\_data['body\_attr'],$attr);

}

}

public function addMeta($key, $value){

$this->\_data['meta'][] = '<meta name="'.$key.'" content="'.$value.'"/>';

}

protected function prepareExt($path){

return $path.'.'.$this->\_templateExt;

}

public function setNotifications($n){

$this->\_data['notify'] = $n;

}

protected function notifications(){

$notify = '';

if(count($this->\_data['notify']) > 0){

$notify='<div class="alerts">'.PHP\_EOL;

foreach ($this->\_data['notify'] as $n) {

if(isset($n['type']) && isset($n['message'])){

$notify.=sprintf('<div class="alert alert-%s"><button type="button" class="close" data-dismiss="alert">&times;</button>%s</div>',$n['type'],$n['message']);

}

}

$notify.='</div>'.PHP\_EOL;

}

return $notify;

}

public function partial($name,$require = true){

$path = $this->prepareExt($this->\_partialPath.$name);

if($require && file\_exists($path)){

require $path;

}

return $path;

}

}