CakePHPCheatSheet

v 2.3.*

CONTROLLERS back to top

Controller Actions

```
class RecipesController extends AppController {
    public function view($id) {
        //action logic goes here..
    }
}
```

Request Data

URL is: /posts/index?page=1&sort=title

```
$this->request->query['page'];
```

Accessing POST data from: input type with name: data[MyModel][title]

```
$this->request->data['MyModel']['title'];
```

Encode to JSON

```
$data = $this->request->input('json_decode');
```

Encode to XML

```
$data = $this->request->input('Xml::build', array('return' => 'domdocument'));
```

Interacting with Views

```
// First you pass data from the controller:
$this->set('color', 'pink');

// In View:
echo $color;
```

Modifying data

```
$this->request->data('Post.title', 'New post')
    ->data('Comment.1.author', 'Mark');
```

Components

Loading Components

```
public $components = array('Session', 'Math');
```

Creating Components

```
App::uses('Component', 'Controller');
class MathComponent extends Component {
    public function doComplexOperation($amount1, $amount2) {
        return $amount1 + $amount2;
```

Redirecting:

```
$this->redirect(array('controller' => 'orders', 'action' => 'thanks'));
```

Redirecting to website:

```
$this->redirect('http://www.example.com');
```

Render the element for ajax:

```
$this->render('/Elements/ajaxreturn');
```

Redirect to 404 not found:

```
throw new NotFoundException('404 Error - Page not found');
```

Before filter

```
public function beforeFilter()
{
    parent::beforeFilter();
    $this->Auth->allow('view');
}
```

Retrieving data from view

From Element view:

```
$comments = $this->requestAction('/comments/latest');
foreach ($comments as $comment) {
    echo $comment['Comment']['title'];
}
```

In Controller:

```
$this->requestAction(
    array('controller' => 'articles', 'action' => 'featured'),
    array('return')
);
```

Load Model

```
$this->loadModel('Article');
$Articles = $this->Article->find('all');

$this->loadModel('User', 2);
$user = $this->User->read();
```

```
    }
}
```

Returns: 3

```
$this->Math->doComplexOperation(1,2);
```

VIEWS back to top

Elements

Display Elemtent

```
echo $this->element('helpbox');
```

Passing Variables into an Element

```
echo $this->element('helpbox', array(
    "helptext" => "Oh, this text is very helpful."
));

// inside helpbox.ctp
echo $helptext;
```

Caching Elements

```
echo $this->element('helpbox', array(), array('cache' => true));
```

Page Titles

From Controller

```
$this->set('title_for_layout', 'View Active Users');
```

From View:

```
$this->set('title_for_layout', $titleContent);
```

Themes

```
class ExampleController extends AppController {
    public $theme = 'Example';
}
```

Set theme in your action:

```
$this->theme = 'AnotherExample';
```

Helpers

/app/View/Helper/HelloHelper.php

```
class ShowHelper extends AppHelper {
    public function hello($name) {
 return 'Hello: '. $name;
    }
}
```

Using your helper

```
echo $this->Show->hello('World');
```

Layouts

```
$this->layout = 'admin';
```

Fetch content

```
echo $this->fetch('content');
```

Retrieving data:

```
$this->requestAction('/articles/featured/limit:3');
$this->requestAction('/articles/view/5');
```

Insert Stylesheet

```
echo $this->Html->css('forms');
```

Insert Javascript

```
echo $this->Html->script('scripts');
```

Media view

Inside your controller action:

```
$this->viewClass = 'Media';
```

Download files from other location

```
$params = array(
    'id'        => 'example.zip',
    'name'      => 'example',
    'download'  => true,
    'extension' => 'zip',
    'path'      => APP . 'outside_webroot_dir' . DS
);
$this->set($params);
```

Response chache

```
$this->response->cache('-1 minute', '+5 days');
```

MODELS back to top

Simple model example:

```
App::uses('AppModel', 'Model');
class Ingredient extends AppModel {
    public $name = 'Ingredient';
}
```

Relations

Has One

```
public $hasOne = array(
    'Profile' => array(
        'className'    => 'Profile',
        'conditions'   => array('Profile.published' => '1'),
        'dependent'    => true
    )
);
```

BelongsTo Relation

```
public $belongsTo = array(
    'User' => array(
        'className'    => 'User',
        'foreignKey'   => 'user_id'
    )
);
```

Has Many

```
public $hasMany = array(
    'Comment' => array(
        'className'     => 'Comment',
        'foreignKey'    => 'user_id',
        'conditions'    => array('Comment.status' => '1'),
        'order'         => 'Comment.created DESC',
        'limit'         => '5',
        'dependent'     => true
    )
);
```

Mas Many and Belongs to

```
public $hasAndBelongsToMany = array(
'Ingredient' =>
    array(
        'className'              => 'Ingredient',
        'joinTable'              => 'ingredients_recipes',
        'foreignKey'             => 'recipe_id',
        'associationForeignKey'  => 'ingredient_id',
        'unique'                 => true
    )
);
```

Limit results hasMany

Limits the maximum number of results for the child hasMany model

```
$this->ModelName->hasMany['ChildModelName']['limit'] = 10;
```

Model Attributes

Database config

```
public $useDbConfig = 'alternate';
```

Table Prefix

```
public $tablePrefix = 'alternate_';
```

Primary Key

```
public $primaryKey = 'example_id';
```

Display field

```
public $displayField = 'fieldname';
```

Retrieving Data

Find all

```
$this->ModelName->find('all');
```

Find by name

```
$this->ModelName->findByName('name');
```

Find First

```
$this->ModelName->find('first');
```

Count

```
$this->ModelName->find('count');
```

Find List

```
$this->ModelName->find('list');
```

findAllBy

Product.order_status = 3

```
$this->Product->findAllByOrderStatus('3');
```

Recipe.type = 'Cookie'

```
$this->Recipe->findAllByType('Cookie');
```

User.last_name = 'Anderson'

```
$this->User->findAllByLastName('Anderson');
```

Cake.id = 7

```
$this->Cake->findAllById(7);
```

```
$this->ModelName->hasMany['ChildModelName']['limit'] = 10;
```

Validation

Single Field validation

```
public $validate = array(
    'fieldName1' => array(
        'rule'       => 'ruleName',
        'required'   => true,
        'allowEmpty' => false,
        'on'         => 'create',
        'message'    => 'Your Error Message'
    )
);
```

Multiple rules per field

```
public $validate = array(
    'login' => array(
        'loginRule-1' => array(
            'rule'    => 'alphaNumeric',
            'message' => 'Only alphabets and numbers',
        ),
        'loginRule-2' => array(
            'rule'    => array('minLength', 8),
            'message' => 'Minimum length of 8 characters'
        )
    )
);
```

Validate from controller

First, set the data to the model:

```
$this->ModelName->set($this->request->data);
```

Then, to check if the data validates, use the validates method of the model, which will return true if it validates and false if it doesn't:

```
if ($this->ModelName->validates()) {
    // it validated logic
} else {
    // didn't validate logic
    $errors = $this->ModelName->validationErrors;
}
```

Retrieve the validation errors in an array:

```
$errors = $this->ModelName->invalidFields();
```

Schema

```
public $_schema = array(
    'id' => array(
        'type' => 'integer',
        'length' => 12
    ),
    'title' => array(
        'type' => 'string',
        'length' => 40
    ),
    'created' => array(
        'type' => 'datetime'
    ),
    'message' => array('type' => 'text')
);
```

User.email = 'jhon' OR User.username = 'jhon';

```
$this->User->findAllByEmailOrUsername('jhon', 'jhon');
```

findBy

Product.order_status = 3

```
$this->Product->findByOrderStatus('3');
```

Recipe.type = 'Cookie'

```
$this->Recipe->findByType('Cookie');
```

Cake.id = 7

```
$this->Cake->findById(7);
```

Conditions

```
$conditions = array(
    'conditions' => array('Model.field' => $thisValue),
    'recursive' => 1, //int
    'fields' => array('Model.field', 'DISTINCT Model.field'),
    'order' => array('Model.created', 'Model.field3 DESC'),
    'group' => array('Model.field'),
    'limit' => n, //int
    'page' => n, //int
    'offset' => n, //int
    'callbacks' => true
);
```

Example for this:

```
$this->ModelName->find('all', $conditions);
```

Compact

```
$this->set(compact('categories'));
```

Query

```
$this->ModelName->query("SELECT * FROM pictures;");
```

Saving Data

```
if ($this->ModelName->save($this->request->data)) {
    $this->Session->setFlash('Data Saved!');
}
```

Save Many

```
$data = array(
    array('title' => 'title 1'),
    array('title' => 'title 2'),
);
$this->ModelName->saveAll($data);
```

Delete

```
$this->Model->delete($this->request->data('Model.id'));
```

Delete all

```
$this->Model->deleteAll(array('Model.spam' => true), false);
```

Field

echo the name for row id 22

```
$this->Post->id = 22;
echo $this->Post->field('name');
```

Behaviors

```
public $actsAs = array(
    'Cheat' => array(
        'option1_key' => 'option1_value'
    )
);
```

Cheat Behaviors

```
class CheatBehavior extends ModelBehavior {

    public function activate(Model $model, $method, $arg1) {
        //Logic..
    }
}
```

Example:

```
$this->Game->Cheat('counterstrike');
```

EMAIL back to top

Load Class

```
App::uses('CakeEmail', 'Network/Email');
```

Send Email example:

```
$Email = new CakeEmail();
$Email->from(array('me@example.com' => 'My Site'))
    ->to('you@example.com')
    ->subject('About')
    ->send('My message');
```

Configuration

```
class EmailConfig {
    public $gmail = array(
        'host' => 'ssl://smtp.gmail.com',
        'port' => 465,
        'username' => 'my@gmail.com',
        'password' => 'secret',
        'transport' => 'Smtp'
    );
}
```

Choosing the sender

```
$Email = new CakeEmail();
$Email->sender('app@example.com', 'MyApp emailer');
```

Configuration

```
$Email = new CakeEmail();
$Email->config('default');
```

Attachment

```
$Email->attachments(array(
    'photo.png' => array(
        'file' => '/full/some_hash.png',
        'mimetype' => 'image/png',
        'contentId' => 'my-unique-id'
    )
));
```

CACHING back to top

Cache Configuration example:

```
Cache::config('short', array(
    'engine' => 'File',
```

Using Groups

```
Cache::config('site_home', array(
    'engine' => 'Redis',
```

```
    'duration' => '+1 hours',
    'path' => CACHE,
    'prefix' => 'cake_short_'
));
```

Caching Query example:

```
public function newest() {
    $result = Cache::read('all_posts', 'short');
    if (!$result) {
        $result = $this->find('all'));
        Cache::write('all_posts', $result, 'short');
    }
    return $result;
}
```

Browser Caching

Cache for 5 Days

```
$this->response->cache('-1 minute', '+5 days');
```

Disable browser caching

```
$this->response->disableCache();
```

Expiration header

```
$this->response->expires('+5 days');
```

```
    'duration' => '+999 days',
    'groups' => array('comment', 'post')
));
```

Clearing Cache

```
public function afterSave($created) {
    if ($created) {
        Cache::clearGroup('post', 'site_home');
    }
}
```

Set Cache

```
Cache::set(array('duration' => '+30 days'));
```

Write Cache

```
Cache::write('results', $data);
```

Read Cache

```
$results = Cache::read('results');
```

Caching Elements

```
echo $this->element('helpbox', array(
        "helptext" => 'This will be $helptext',
    ),
    array(
        "cache" => "short",
        "callbacks" => true
    )
);
```

FORMS back to top

Creating Forms

```
echo $this->Form->create('Recipe');
```

Using Get Method

```
echo $this->Form->create('User', array('type' => 'get'));
```

With file upload

```
echo $this->Form->create('User', array('type' => 'file'));
```

Action to URL

```
echo $this->Form->create(null, array('url' => '/recipe/add'));
```

Action to controller:

```
echo $this->Form->create(null, array(
    'url' => array('controller' => 'recipe', 'action' => 'add')
));
```

Closing the form

Password:

```
echo $this->Form->input('password');
```

Username:

```
echo $this->Form->input('username', array('label' => 'Username'));
```

Birthday:

```
echo $this->Form->input('birth_dt', array(
    'label' => 'Date of birth',
    'dateFormat' => 'DMY',
    'minYear' => date('Y') - 70,
    'maxYear' => date('Y') - 18,
));
```

Email:

```
echo $this->Form->input('email', array('type' => 'email'));
```

File:

```
echo $this->Form->input('field', array('type' => 'file'));
```

```
echo $this->Form->end('Submit');
```

Buttons

```
echo $this->Form->button('Click me');
echo $this->Form->button('Button', array('type' => 'button'));
echo $this->Form->button('Reset', array('type' => 'reset'));
echo $this->Form->button('Submit', array('type' => 'submit'));
```

Setting defaults for all the fields

```
echo $this->Form->inputDefaults(array(
        'label' => false,
        'div' => false,
        'class' => 'fancy'
    )
);
```

Hidden:

```
echo $this->Form->hidden('id');
```

Textarea:

```
echo $this->Form->textarea('notes');
```

Options:

```
echo $this->Form->input('field', array(
    'options' => array(1, 2, 3, 4, 5),
    'empty' => '(choose one)'
));
```

Checkbox:

```
echo $this->Form->checkbox('published', array('hiddenField' => false));
```

Radio:

```
$options = array('M' => 'Male', 'F' => 'Female');
$attributes = array('legend' => false);
echo $this->Form->radio('gender', $options, $attributes);
```

Year:

```
echo $this->Form->year('purchased', 2000, date('Y'));
```

Day

```
echo $this->Form->day('created');
```

HTML Helper back to top

Charset

```
echo $this->Html->charset('ISO-8859-1');
```

Stylesheet

```
echo $this->Html->css('forms');
```

Or multiple Stylesheets

```
echo $this->Html->css(array('forms', 'tables', 'menu'));
```

Javascript

```
echo $this->Html->script('scripts');
```

From remote:

```
$this->Html->script('http://code.jquery.com/jquery.min.js');
```

Favicon

```
echo $this->Html->meta(
    'favicon.ico',
    '/favicon.ico',
```

Hyperlink

```
echo $this->Html->link(
    'Click me',
    array('controller' => 'dashboards', 'action' => 'index')
);
```

or

```
echo $this->Html->link('Click', '/page/home');
```

Image

```
echo $this->Html->image('logo.png', array('alt' => 'Cake'));
```

Image with Link

```
echo $this->Html->image("recipes/6.jpg", array(
    "alt" => "Brownies",
    'url' => array('controller' => 'contr', 'action' => 'add')
));
```

Span tag

```
echo $this->Html->tag('span', 'Hello World.', array('class' => 'welcome'));
```

```
    array('type' => 'icon')
);
```

Page Description

```
echo $this->Html->meta(
    'description',
    'enter any meta description here'
);
```

Doctype

```
echo $this->Html->docType('html5');
```

Audio

```
echo $this->Html->media('audio.mp4');
```

Div

```
echo $this->Html->div('error', 'Please enter your credit card number.');
```

Table header

```
$this->Html->tableHeaders(array('Date', 'Title', 'Sweet'));
```

Table Cells

```
$this->Html->tableCells(array(
    array('Jul 7th, 2007', 'Best Brownies', 'Yes'),
    array('Jun 21st, 2007', 'Smart Cookies', 'Yes'),
    array('Aug 1st, 2006', 'Anti-Java Cake', 'No'),
));
```

PAGINATOR back to top

Paginate setup:

```
public $paginate = array(
    'limit' => 25,
    'order' => array(
        'Post.title' => 'asc'
    )
);
```

Paginator Example:

```
public function list_recipes() {
    $this->paginate = array(
        'conditions' => array('Recipe.title LIKE' => 'a%'),
        'limit' => 10
    );
    $data = $this->paginate('Recipe');
    $this->set(compact('data'));
);
```

Max Limit

```
public $paginate = array(
    // other keys here.
    'maxLimit' => 10
);
```

Paginate with GET

```
public $paginate = array(
    'paramType' => 'querystring'
);
```

```
$this->Paginator->settings['paramType'] = 'querystring';
```

Modifying the options

```
$this->Paginator->options(array(
    'url' => array(
        'sort' => 'email', 'direction' => 'desc', 'page' => 6,
```

Paginator in viewer

Sort

```
echo $this->Paginator->sort('user_id');
```

With title

```
echo $this->Paginator->sort('user_id', 'User account');
```

With custom HTML

```
echo $this->Paginator->sort('user_id', '<i>User account</i>', array('escape' => false));
```

Direction

```
echo $this->Paginator->sort('user_id', null, array('direction' => 'desc'));
```

Numbers

Display numbers

```
echo $this->Paginator->numbers();
```

Display first and last 2

```
echo $this->Paginator->numbers(array('first' => 2, 'last' => 2));
```

Previous

```
echo $this->Paginator->prev(' << ' . __('previous'), array(), null, array('class' => 'prev disabled'));
```

Next

```
echo $this->Paginator->next(__('next').' >> ', array(), null, array('class' => 'next disabled'));
```

Current

```
echo $this->Paginator->current('Comment');
```

```
        'lang' => 'en'
    )
));
```

Paginator with Ajax

Add component

```
public $components = array('RequestHandler');
public $helpers = array('Js');
```

Include JQuery

```
echo $this->Html->script('jquery');
```

In your view:

```
$this->Paginator->options(array(
    'update' => '#content',
    'evalScripts' => true
));
```

Add this to the bottom of view

```
echo $this->Js->writeBuffer();
```

UTILITIES back to top

Numbers

Include Class

```
App::uses('CakeNumber', 'Utility');
```

Currency

```
echo CakeNumber::currency('1234.56', 'EUR');
echo $this->Number->currency('1234.56', 'EUR');
```

Precision

Outputs: 456.92

```
echo $this->Number->precision(456.91873645, 2);
```

Percentage

Outputs: 45.69%

```
echo $this->Number->toPercentage(45.691873645);
```

ReadableSize

Outputs: 1.26 MB

As NumberHelper

```
echo $this->Number->toReadableSize(1321205.76);
```

As CakeNumber

```
echo CakeNumber::toReadableSize(1321205.76);
```

Time

Formats

Outputs: August 22nd, 2011 11:53 AM

```
$this->Time->format('F jS, Y h:i A', '2011-08-22 11:53:00');
```

To Timestamp

Outputs: 1313971200

```
echo $this->Time->fromString('Aug 22, 2011');
```

Nice

Outputs: Mon, Aug 22nd 2011, 11:53

As TimeHelper

```
echo $this->Time->nice('2011-08-22 11:53:00');
```

As CakeTime

```
echo CakeTime::nice('2011-08-22 11:53:00');
```

Nice Short

Outputs: Aug 22nd, 11:53

As TimeHelper

```
echo $this->Time->niceShort('2011-08-22 11:53:00');
```

As CakeTime

Numberformat

Outputs: '¥ 123,456.79'

As NumberHelper

```
echo $this->Number->format('123456.7890', array(
    'places' => 2,
    'before' => '¥ ',
    'escape' => false,
    'decimals' => '.',
    'thousands' => ','
));
```

As CakeNumber

```
App::uses('CakeNumber', 'Utility');
echo CakeNumber::format('123456.7890', array(
    'places' => 2,
    'before' => '¥ ',
    'escape' => false,
    'decimals' => '.',
    'thousands' => ','
));
```

Sanitization

```
$badString = '<font size="99" color="#FF0000">HEY</font><script>...</script>';
```

Outputs: HEY<script>...</script>

```
echo Sanitize::html($badString);
```

Outputs: HEY...

```
echo Sanitize::html($badString, array('remove' => true));
```

Folder

Create Dir

```
$dir = new Folder('/path/to/folder');
```

Create dir with permission

```
$dir = new Folder('/path/to/folder', true, 0755);
```

Search for all .ctp:

```
$files = $dir->find('.*\.ctp');
```

Read file

```
$contents = $file->read();
```

Copy folder

```
$folder1 = new Folder('/path/to/folder1');
$folder1->copy('/path/to/folder2');
```

Delete

```
$folder = new Folder('foo');
if ($folder->delete()) {
    // Successfully deleted foo its nested folders
```

```
echo CakeTime::niceShort('2011-08-22 11:53:00');
```

timeAgoInWords

Outputs: 2 months, 2 weeks, 6 days ago

As TimeHelper

```
echo $this->Time->timeAgoInWords('Aug 22, 2011', array('format' => 'F jS, Y', 'end' => '+1 year'));
```

As CakeTime

```
echo CakeTime::timeAgoInWords('Aug 22, 2011', array('format' => 'F jS, Y', 'end' => '+1 year'));
```

Convert

Outputs: 1321038036

As TimeHelper

```
echo $this->Time->convert(time(), 'Asia/Jakarta');
```

As CakeTime

```
echo CakeTime::convert(time(), new DateTimeZone('Asia/Jakarta'));
```

Logging

Config

```
CakeLog::config('otherFile', array(
    'engine' => 'DatabaseLogger',
    'model' => 'LogEntry',
));
```

Results in this being appended to app/tmp/logs/error.log

```
$this->log("Something didn't work!");
```

Results in this being appended to app/tmp/logs/activity.log

```
CakeLog::write('activity', 'A special message for you!');
```

Routing

```
Router::connect(
    '/myprofile',
    array('controller' => 'users', 'action' => 'view', 1)
);
```

Encryption

Cipher

Encrypt your text with my_key

```
$secret = Security::cipher('hello world', 'my_key');
```

Decrypt your text

```
$nosecret = Security::cipher($secret, 'my_key');
```

rijndael

Encrypt

```
$encrypted = Security::rijndael('a secret', Configure::read('Security.key'), 'encry
```

```
}
```

String

Include class

```
App::uses('String', 'Utility');
```

Insert

Outputs: My name is Bob and I am 65 years old.

```
String::insert('My name is :name and I am :age years old.', array('name' => 'Bob', 'age' => '65'));
```

Highlight

```
echo $this->Text->highlight($lastSentence, 'using', array('format' => '<span class="highlight">\1</span>'));
```

Truncate

Outputs: The killer crept...

As TextHelper

```
echo $this->Text->truncate(
    'The killer crept forward and tripped on the rug.',
    22,
    array(
        'ellipsis' => '...',
        'exact' => false
    )
);
```

As String

```
echo String::truncate(
    'The killer crept forward and tripped on the rug.',
    22,
    array(
        'ellipsis' => '...',
        'exact' => false
    )
);
```

Sessions

read the session

```
$green = $this->Session->read('Person.eyeColor');
```

Delete session

```
$this->Session->delete('Person.eyeColor');
```

```
$this->Session->delete('Person');
```

Destroy

```
$this->Session->destroy();
```

User Agent

```
$this->Session->read('Config.userAgent');
```

Auth

```
pt');
```

Decrypt

```
$decrypted = Security::rijndael($encrypted, Configure::read('Security.key'), 'decrypt');
```

CakePHP Salt

Sha1

```
$sha1 = Security::hash('CakePHP Framework', 'sha1', true);
```

md5

```
$md5 = Security::hash('CakePHP Framework', 'md5', 'my-salt');
```

hash

```
$hash = Security::hash('CakePHP Framework');
```

XML

Import XML

```
$xml = Xml::build('http://bakery.cakephp.org/articles/rss');
```

Old method

```
$xml = new Xml('http://bakery.cakephp.org/articles/rss');
```

Flash

Display good message

```
$this->Session->setFlash('Bad!.', 'default', array(), 'bad');
```

Display bad message

```
$this->Session->setFlash('Good.', 'default', array(), 'good');
```

Cookie

Write cookie

```
$this->Cookie->write('name', 'Larry');
```

Expire after 1 hour

```
$this->Cookie->write('name', 'Larry', false, 3600);
```

Read cookie

```
echo $this->Cookie->read('name');
```

Delete Cookie

```
$this->Cookie->delete('name');
```

```
if ($this->Auth->user('role') === 'admin') {
    $this->theme = 'Admin';
}
```

CONSOLE back to top

Creating Shell

```
class HelloShell extends AppShell {
    public function main() {
        $this->out('Hello world.');
    }
}
```

Using Models in your Shell

```
public $uses = array('User');
```

Creating Tasks

In HelloShell:

```
public $tasks = array('Template');
```

Task:

```
class FileGeneratorTask extends Shell {
    public function execute() {
     //Logic..
    }
}
```

Argument

```
$parser->addArgument('model', array('help' => 'Help!'));
```

```
$parser->addArgument('type', array(
    'help' => 'The type of node to interact with.',
    'required' => true,
    'choices' => array('aro', 'aco')
));
```

Options

```
$parser->addOption('connection', array(
    'short' => 'c',
    'help' => 'connection',
    'default' => 'default',
));
```

Export path

```
export PATH="$PATH:/Users/mark/cakephp/lib/Cake/Console"
```

Echo output

```
$this->out('Hello world.');
```

Styling

```
$this->stdout->styles('red', array('text' => 'red'));
$this->stdout->styles('green', array('text' => 'green'));
```

Description

```
$parser->description(array('line one', 'line two'));
$parser->description();
```

Cronjob

```
*/5 *  *  *  * /full/path/to/cakeshell myshell myparam -cli /usr/bin -console /cakes/2.x.x/lib/Cake/Console -app /full/path/to/app
```