

In Model

```
var $validate = array(
    'fieldName' => array(
        'ruleName1' => array(
            'rule' => 'ruleName' / array('ruleName', $args),
            'required' => boolean,
            'allowEmpty' => boolean,
            'on' => 'update' / 'create'
        ),
        'ruleName2' => array(
            'rule' => 'ruleName',
            ...
        )
    )
    'anotherFieldName' => array(
        ....
    )
);
```

Rules

alphaNumeric**between**, min, max**blank****cc**, type, deep (bool), regex**comparison**, (>=, >, =, <=, <, !=), value**date** [, format]**decimal** [,places]**email** [, verify host (bool)]**equalTo**, value**extension**, ext/array**file** (?)**ip****maxLength**, value**money** (?)**isUnique**,**minLength**, value,**inList**, array**numeric**,**notEmpty**: field is not empty**phone**, regex/null, country (us)**postal**, regex/null, country (us, ca, uk, it, de, be)**range**, min, max (non inclusive)**ssn**, regex/null, country**url**, [strict (bool)]

Custom Validation

Regex Validation

```
'rule' => array('custom', '/[a-z0-9]{3,}$/i'),
```

Method Validation

```
'rule' => array('method', $params...)
write a Model::method($value, $params...) that returns boolean
```

`$value[field]` contains the value to validate

You can access model data in `$this->data` to compare fields

You can override existing validation methods

Validation in the Controller

1. Set the data to the model
`$this->modelName->set($this->data);`
2. Call `Model::validates()` boolean
`$this->modelName->validates()`
3. See errors
`$errors = $this->modelName->invalidFields();`