Reference: Model rules validation

1. How validation works
2. Validation rules reference
3. Selected readings

This is a reference to be used for Model rule validation and is compiled from the Yii documentation and code. The purpose is to have all the information gathered in one place instead of scattered. This reference is not an intro. See The Definitive Guide to Yii, Declaring Validation Rules for a tutorial.

How validation works

The [CModel] class uses a method named [CModel::rules()] to return an array with the rules for validation.

```
public function rules()
{
    return array(
        array('username, password', 'required'),
        array('password_repeat', 'required', 'on'=>'register'),
        array('password', 'compare', 'compareAttribute'=>'password_repeat', 'on'
    );
}
```

The code above is an example of what the [CModel::rules()] function may look like. In the main array, each line is an array that declare a new validation rule (described in next section).

These rules are applied by the [CModel::validate()] method which returns a boolean value. By default, the method [CActiveRecord::save()] automatically calls this validation, and requires it to succeed before it tries to save the record.

Parameters of a validator

```
array(
    // mandatory arguments
        'attribute list',
        'validator name',
        // optional parameters
        'on'=>'scenario name',
        'message'=>'The attribute didn\'t validate!',
        ...validation parameters...
);
```

- *attribute list*: specifies the attributes (separated by commas) to be validated;
- *validator name*: specifies the validator to be used. See the next section for details.
- *on*: this specifies the scenarios when the validation rule should be performed. Separate different scenarios with commas. If this option is not set, the rule will be applied in any scenario. See the section [Scenarios][#Scenarios] for details.
- *message*: replaces the default error message if validation fails.

- *...validation parameters...*: any number of extra parameters to be used by the specified validator.

Note: You may get into trouble when using model's attributes as additional parameters for validators. Take a look at: https://github.com/yiisoft/yii/issues/3001

Choice of validators

Yii will look for validators in the specific order:

1. A method in the model class with the same name as the validator specified.
2. A built-in validator in Yii that extends the [CValidator] class.
3. A path/alias to a home made class (extending [CValidator]) that is not built in.

Here is an example of these 3 kinds:

```
public function rules()
{
    return array(
        array('password', 'validateLogin'), // Custom validation method in this
        array('username, password', 'required'), // built-in alias for CRequired
        array('username', 'ext.MyValidators.MyLoginValidator'), // Custom valida
    );
}

public function validateLogin() {
// [...]
}
```

Scenarios

Each model object has a scenario property. Some scenarios are built-in and will be assigned automatically by Yii, but you can define your own.

For instance, a [CActiveRecord] read from the database has the "update" scenario, while a new record has the "insert" scenario.

```
$modelA = User::model()->findByPk(1); // $model->scenario = 'update'
$modelB = new User();                 // $model->scenario = 'insert'
$modelB->scenario = 'light';          // custom scenario
if ($modelB->validate()) {            // will only apply rules of the "light" sc
```

As shown in the example of the first section, the rules can be restricted to a specific scenario through the `"on" => "scenario"` parameter.

Validation rules reference

For each rule, the list of its specific arguments is given with the eventual default value *(in italics)*.

`boolean` : [CBooleanValidator]

Validates that the attribute value is either trueValue or falseValue.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **falseValue**, the value representing false status. *(0)*

3. **strict**, whether the comparison to trueValue and falseValue is strict. *(false)*
4. **trueValue**, the value representing true status. *(1)*

`captcha` : [CCaptchaValidator]

Validates that the attribute value is the same as the verification code displayed in the CAPTCHA.

1. **allowEmpty**, whether the attribute value can be null or empty. *(false)*
2. **caseSensitive**, whether the comparison is case sensitive. *(false)*

`compare` : [CCompareValidator]

Compares the specified attribute value with another value and validates if they are equal.

1. **allowEmpty**, whether the attribute value can be null or empty. *(false)*
2. **compareAttribute**, the name of the attribute to be compared with.
3. **compareValue**, the constant value to be compared with.
4. **operator**, the operator for comparison. *("==")*
5. **strict**, whether the comparison is strict (both value and type must be the same). *(false)*

`date` : [CDateValidator]

Validates that the attribute value is a valid date, time or datetime.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **format**, the format pattern that the date value should follow. This can be either a string or an array representing multiple formats. The formats are described in [CDateTimeParser] API. *('MM/dd/yyyy')*
3. **timestampAttribute**, name of the attribute that will receive date parsing result. *(null)*

NOTE: The date validator was added in Yii **1.1.7** - it's not available prior to that

Example:

```
array('org_datetime', 'date', 'format'=>'yyyy-M-d H:m:s'),
```

`default` : [CDefaultValueValidator]

Sets the attributes with the specified value. It does not do validation. Its existence is mainly to allow specifying attribute default values in a dynamic way.

1. **setOnEmpty**, whether to set the default value only when the attribute value is null or empty string. *(true)*
2. **value**, the default value to be set to the specified attributes.

`email` : [CEmailValidator]

Validates that the attribute value is a valid email address.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **allowName**, whether to allow name in the email address. *(false)*
3. **checkMX**, whether to check the MX record for the email address. *(false)*
4. **checkPort**, whether to check port 25 for the email address. *(false)*
5. **fullPattern**, the regular expression used to validate email addresses with the name part. Requires that the property "allowname" is on.
6. **pattern**, the regular expression used to validate the attribute value.

`exist` : [CExistValidator]

Validates that the attribute value exists in a table.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **attributeName**, the ActiveRecord class attribute name that should be used to look for the attribute value being validated. *(null, meaning same attribute name)*

3. **className**, the ActiveRecord class name that should be used to look for the attribute value being validated. *(null, meaning same class name)*
4. **criteria**, additional query criteria. *(array())*

`file` : [CFileValidator]

Verifies if an attribute is receiving a valid uploaded file.

1. **allowEmpty**, whether the attribute requires a file to be uploaded or not. *(false)*
2. **maxFiles**, the maximum file count the given attribute can hold. *(1)*
3. **maxSize**, the maximum number of bytes required for the uploaded file. *(null, meaning no limit)*
4. **minSize**, the minimum number of bytes required for the uploaded file. *(null, meaning no limit)*
5. **tooLarge**, the error message used when the uploaded file is too large.
6. **tooMany**, the error message used if the count of multiple uploads exceeds limit.
7. **tooSmall**, the error message used when the uploaded file is too small.
8. **types**, a list of file name extensions that are allowed to be uploaded. *(null, meaning all extensions)*
9. **wrongType**, the error message used when the uploaded file has an extension name that is not listed among extensions.
10. **mimeTypes**, a list of MIME-types of the file that are allowed to be uploaded. *(requires fileinfo PHP extension)*
11. **wrongMimeType**, the error message used when the uploaded file has a MIME-type that is not listed among mimeTypes.

`filter` : [CFilterValidator]

Transforms the data being validated based on a filter.

1. **filter**, the filter method.

Examples:

- With a custom function

```
array('mycode','filter','filter'=>'strtoupper'),
```

- With a custom method

```
array('eanupc','filter','filter'=>array($this,'keepProductCode')),
array('eanupc','filter','filter'=>array(self,'formatProductCode')),
...
public function keepProductCode($code){
        // returns a new value (uses $this) ...
}
static public function formatProductCode($code){
        // returns a new value (cannot use $this) ...
}
```

NOTE: The filter validator maybe use with other validators for the same attribute. In this case one must pay attention to the order of the validators list as the filter validator will change the attribute value.

`in` : [CRangeValidator]

Validates that the attribute value is among the list (specified via range).

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*

2. **not**, whether to invert the validation logic. Since Yii 1.1.5. *(false)*
3. **range**, list of valid values that the attribute value should be among.
4. **strict**, whether the comparison is strict (both type and value must be the same). *(false)*

Example:

```
array('language','in','range'=>array('en','fr','zn'),'allowEmpty'=>false),
```

`length` : [CStringValidator]

Validates that the attribute value is of certain length.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **encoding** , string encoding. *(null, meaning unchecked)*
3. **is**, exact length. *(null, meaning no limit)*
4. **max**, maximum length. *(null, meaning no limit)*
5. **min**, minimum length. *(null, meaning no limit)*
6. **tooLong**, user-defined error message used when the value is too long.
7. **tooShort**, user-defined error message used when the value is too short.

`numerical` : [CNumberValidator]

Validates that the attribute value is a number.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **integerOnly**, whether the attribute value can only be an integer. *(false)*
3. **max**, upper limit of the number. *(null, meaning no limit)*
4. **min**, lower limit of the number. *(null, meaning no limit)*
5. **tooBig**, user-defined error message used when the value is too big.
6. **tooSmall**, user-defined error message used when the value is too small.

Example:

```
array('quantity','numerical',
    'integerOnly'=>true,
    'min'=>1,
    'max'=>250,
    'tooSmall'=>'You must order at least 1 piece',
    'tooBig'=>'You cannot order more than 250 pieces at once'),
```

`match` : [CRegularExpressionValidator]

Validates that the attribute value matches to the specified regular expression.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **not**, whether to invert the validation logic. Since Yii 1.1.5. *(false)*
3. **pattern**, the regular expression to be matched with.

`required` : [CRequiredValidator]

Validates that the specified attribute does not have null or empty value.

1. **requiredValue**, the desired value that the attribute must have. *(null)*
2. **strict**, whether the comparison to "requiredValue" is strict. *(false)*

`safe` : [CSafeValidator]

Marks the associated attributes to be safe for massive assignments.

`type` : [CTypeValidator]

Verifies if the attribute is of the type specified by type. (integer, float, string, date, time, datetime). Since 1.1.7 you should use [CDateValidator] to validate dates.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **dateFormat**, the format pattern that the date value should follow. *('MM/dd/yyyy')*
3. **datetimeFormat**, the format pattern that the datetime value should follow. *('MM/dd/yyyy hh:mm')*
4. **timeFormat**, the format pattern that the time value should follow. *('hh:mm')*
5. **type**, the data type that the attribute should be. *('string')*

Example of time rule;

```
array('org_starttime, org_finishtime', 'type', 'type'=>'time', 'timeFormat'=>'hh
```

`unique` : [CUniqueValidator]

Validates that the attribute value is unique in the corresponding database table.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **attributeName**, the ActiveRecord class attribute name that should be used to look for the attribute value being validated. *(null, meaning same attribute name)*
3. **caseSensitive**, whether the comparison is case sensitive. *(true)*
4. **className**, the ActiveRecord class name that should be used to look for the attribute value being validated. *(null, meaning same class name)*
5. **criteria**, additional query criteria. *(array())*
6. **skipOnError**, whether this validation rule should be skipped if when there is already a validation error for the current attribute. *(true)*

By default, [CUniqueValidator] works with a single attribute that's presumed to be unique across the whole model table, but it can work with multi-attribute unique constraints as well. See [CUniqueValidator::c2215] for an example.

`unsafe` : [CUnsafeValidator]

Marks the associated attributes to be unsafe so that they cannot be massively assigned.

`url` : [CUrlValidator]

Validates that the attribute value is a valid "http" or "https" URL.

1. **allowEmpty**, whether the attribute value can be null or empty. *(true)*
2. **defaultScheme**, the default URI scheme. It will be prepended to the input, if the input doesn't contain one. Since Yii 1.1.7. *(null, meaning the url must contain a scheme part)*
3. **pattern**, the regular expression used to validates the attribute value.
4. **validSchemes**, the allowed URI scheme. *(array('http', 'https'))*

Selected readings

- The Definitive Guide to Yii: Working with Forms - Creating Model
- [CModel::validate()]
- [CModel::rules()]
- [CModel::$scenario]
- [CValidator]
- [CInlineValidator]

Built in validators in Yii

The validators are all the classes that inherit from [CValidator], so the API documentation contains an always up-to-date list in the subclasses description of CValidator.

- [CBooleanValidator]

- [CCaptchaValidator]
- [CCompareValidator]
- [CDefaultValueValidator]
- [CEmailValidator]
- [CExistValidator]
- [CFileValidator]
- [CFilterValidator]
- [CNumberValidator]
- [CRangeValidator]
- [CRegularExpressionValidator]
- [CRequiredValidator]
- [CSafeValidator]
- [CStringValidator]
- [CTypeValidator]
- [CUniqueValidator]
- [CUnsafeValidator]
- [CUrlValidator]