Constraint

Nếu ở DB bị lỗi constraint sqlize sẽ throw SequelizeUniqueConstraintError

const User = sequelize.define("user", {

username: {

type: DataTypes.TEXT,

allowNull: false,

unique: true

}

});

Validation

sequelize.define('foo', {

bar: {

type: DataTypes.STRING,

validate: {

is: /^[a-z]+$/i, // matches this RegExp

is: ["^[a-z]+$",'i'], // same as above, but constructing the RegExp from a string

not: /^[a-z]+$/i, // does not match this RegExp

not: ["^[a-z]+$",'i'], // same as above, but constructing the RegExp from a string

isEmail: true, // checks for email format (foo@bar.com)

isUrl: true, // checks for url format (http://foo.com)

isIP: true, // checks for IPv4 (129.89.23.1) or IPv6 format

isIPv4: true, // checks for IPv4 (129.89.23.1)

isIPv6: true, // checks for IPv6 format

isAlpha: true, // will only allow letters

isAlphanumeric: true, // will only allow alphanumeric characters, so "\_abc" will fail

isNumeric: true, // will only allow numbers

isInt: true, // checks for valid integers

isFloat: true, // checks for valid floating point numbers

isDecimal: true, // checks for any numbers

isLowercase: true, // checks for lowercase

isUppercase: true, // checks for uppercase

notNull: true, // won't allow null

isNull: true, // only allows null

notEmpty: true, // don't allow empty strings

equals: 'specific value', // only allow a specific value

contains: 'foo', // force specific substrings

notIn: [['foo', 'bar']], // check the value is not one of these

isIn: [['foo', 'bar']], // check the value is one of these

notContains: 'bar', // don't allow specific substrings

len: [2,10], // only allow values with length between 2 and 10

isUUID: 4, // only allow uuids

isDate: true, // only allow date strings

isAfter: "2011-11-05", // only allow date strings after a specific date

isBefore: "2011-11-05", // only allow date strings before a specific date

max: 23, // only allow values <= 23

min: 23, // only allow values >= 23

isCreditCard: true, // check for valid credit card numbers

// Examples of custom validators:

isEven(value) {

if (parseInt(value) % 2 !== 0) {

throw new Error('Only even values are allowed!');

}

}

isGreaterThanOtherField(value) {

if (parseInt(value) <= parseInt(this.otherField)) {

throw new Error('Bar must be greater than otherField.');

}

}

}

}

});

## Model-wide validation

class Place extends Model {}

Place.init({

latitude: {

type: DataTypes.INTEGER,

validate: {

min: -90,

max: 90

}

}

}, {

sequelize,

validate: {

bothCoordsOrNone() {

if (this.latitude === null) {

throw new Error('Either both latitude and longitude, or neither!');

}

}

}

})

## Validating

const someplace = new Place(…)

somePlace.validate()

Result

{

'latitude': ['Invalid number: latitude'],

'bothCoordsOrNone': ['Either both latitude and longitude, or neither!']

}

## Custom Message

isInt: {

msg: "Must be an integer number of pennies"

}

With args   
 isIn: {

args: [['en', 'zh']],

msg: "Must be English or Chinese"

}

With allowNull Constraint

allowNull: false,

validate: {

notNull: {

msg: 'Please enter your name'

}

}