链滴

joi 参数校验，过滤

作者：XPPA

原文链接：https://ld246.com/article/1540263244653

来源网站：链滴

前言

在后端开发中，后端的接口不应该过分信任前端提交的参数，这是保障接口安全的基本。而'隔栏'这设计是思路就是在接口的驱动层之前对所有前端的参数进行校验及过滤，默认为在经过这一层后我们有的数据都是干净的数据，不在业务层、逻辑层、数据层在对数据做任何校验。

在之前的项目参数校验中，主要封装了几个基本的校验方法，而对一些比较详细的校验(类型、长度等都是通过单独写一个函数进行校验。之后引入了joi这种可以将参数校验做成类似配置的工具,

```
ID: Joi.string().length(12).required()
INTEGER: Joi.number().integer().min(0).required()
```

使用

joi的安装这里就不在赘述了，在具体使用时，自己写了一个简单的方法，定义了一些常用的属性方便接使用，代码如下：

```
const Joi = require('joi');

// 通用字段
const joiCommonFields = {
    // 必填项
    required: {
        // 整数
        INTEGER: Joi.number().integer().required(),
        // 字符串
        STRING: Joi.string().required(),
        // 数组
        ARRAY: Joi.array().required(),
        // boolean类型值，兼容 'true', '1', 1
        BOOLEAN: Joi.boolean().truthy(['true', '1', 1]).required(),
        /** 最大长度200最小长度1的字符串 */
        URL: Joi.string().min(1).max(200).required(),
    },
    // 可选项
    optional: {
        // 整数
        INTEGER: Joi.number().integer().optional(),
        // 字符串
        STRING: Joi.string().optional(),
        // 数组
        ARRAY: Joi.array().optional(),
        // boolean类型值，兼容 'true', '1', 1
        BOOLEAN: Joi.boolean().truthy(['true', '1', 1]).optional(),
        /** 最大长度200最小长度1的字符串 */
        URL: Joi.string().min(1).max(200).optional(),
    },
};

/**
 * 参数校验
 * @param {Object} checkInfo 要校验的对象
 * @param {Object} schema 参数校验
```

```
 * @param {Object} option 可选项
 * @param {Boolean} option.abortEarly: 设置true，可以在检测到第一个错误时立即返回，默认fals
(检查全部)。推荐设置true
 * @param {Boolean} option.convert:设置true，可以尝试将值转换为所需的类型(例如，将字符串
换为数字）。默认为true。推荐采用默认
 * @param {Boolean} option.allowUnknown: 默认false，则允许对象包含被忽略的未知键。默认为
alse。推荐设置true
 * @param {Boolean} option.skipFunctions:如果为true，则忽略具有函数值的未知键。默认为fals
。推荐采用默认
 * @param {Boolean} option.stripUnknown: 默认为false,从对象和数组中删除未知的元素。默认为f
lse。也可以特殊的设置成 { objects: true , arrays: true }的形式，可以对象和数组分别处理。推荐采
默认
 * @param {String} option.presence: 设置默认的可选需求。支持的模式:’ optional’ ,’ required’
和 ’ forbidden’ 。默认为 ’ optional’ 。推荐采用默认
 * @param {Boolean} option.escapeHtml: 当为true时，出于安全目的，错误消息模板将特殊字符
义为html实体。默认为false。推荐采用默认
 * @param {Boolean} option.noDefaults:如果为true，则不应用默认值。默认为false。推荐采用默

 */
function joiValidate(checkInfo, schemaInfo, option = {stripUnknown: true}) {
    const schemaObj = Joi.object(schemaInfo);
    const {error, value} = Joi.validate(checkInfo, schemaObj, option);
    if (error || !schemaInfo || !checkInfo) {
        // 内部异常处理，可以替换为自己的异常处理方法
        throw new MError(MError.PARAMETER_ERROR, JSON.stringify(error));
    }
    return value;
}

exports.joiValidate = joiValidate;
exports.joiCommonFields = joiCommonFields;
```

参数定义:

```
const validatorSchema = {
    functionName: { // 方法名
        param_1: commonFields.required.ID,
        param_2: commonFields.required.INTEGER,
    },
};
```

调用方式:

```
    // 参数校验、过滤
    const legalData = joiValidate({param_1, param_2}, validatorSchema.functionName);
```

2018/11/10 补充

在使用过一段时间后补充一些用法

when

Joi中when的用法和if很像如下：通过判断a的值执行相应的校验方法。

```
const schema = {
    a: Joi.valid('a', 'b', 'other'),
    other: Joi.string()
        .when('a', { is: 'other', then: Joi.required() }),
};
```

上面的代码是官方的一个例子，when的写法很多，个人比较推荐上面的写法，when是可以用链式调多个，但在官方文档上没有说明。

Joi.date()

这个方法主要是时间判断,如下代码：

```
const schema = Joi.object({
  from: Joi.date().required(),
  to: Joi.date().min(Joi.ref('from')).required()
});
```

min、max可以设置最小时间，但是需要注意在获取时间参数到参数校验时，需要花费一点时间，使用Joi.date('now'))校验当前时间可能造成参数校验不通过。Joi.ref表示参数的引用。

附上官方文档连接:官方文档