Express Validator

The express-validator library is a middleware that wraps all the validators and sanitizers found in **validator.js**. Its main feature is creating validation chains by allowing these validators and sanitizers to be combined in any order. It determines whether a request is valid or not.

The express-validator may work with libraries that aren't part of express.js. It only requires that our server model its HTTP requests in a way that is similar to express.js. [1]

Overview:

- The **query()** function can be used to check whether a URL / request body parameter's value. [2]
- Validators don't automatically report errors to users. This is because if the validators were part of a validation chain, then the order in which the errors would be reported cannot be determined by the validators themselves. [2]
- Thus we need to collect the errors (if any) by passing **req** to the **validationResult()** function. Here, the error is returned as a JSON object which contains the **errors** array. Each error in that array is a JSON object which has the following keys: **location** (Where the error occurred), **msg** (Error Message), **path** (name) and **type** (What the path belongs to). [2]
- The **escape()** sanitizer can be used to convert HTML characters (Eg - <, >) into text. This is to prevent XSS attacks. [2]
- The **matchedData()** function returns all the data that express may have validated / sanitized. [2]

Validation Chains:

- A validation involves chaining several methods together. In this chain, a value is wrapped by validations and changed by sanitizations that are returned from the respective methods. [3]
- A validation chain is also a middleware since it can be passed to route handlers. [3]
- A validation chain has 3 kinds of methods: **validators**, **sanitizers** and **modifiers**. [3]
- Validators validate a value by checking if it is of the right data type and format. [3]
- Sanitizers remove unwanted values from an input, cast it to the right JS type and may also provide some line of defence against XSS. [3]
- Modifiers control the behavior of validation chains including when and whether they should run and also control error messages. [3]
- All the validators and sanitizers present in validator.js are called **standard validators**. [3]
- Since **validator.js** works only with strings, therefore express-validator converts all values used with standard validators to string first. [3]

Chaining Order:

- The methods used in a validation chain run in the order in which they are called. An exception to this is the **optional()** method which can be placed anywhere in the chain and will mark the chain optional in the same way. [3]

Reusing Validation Chains:

- A validation chain is always mutable. More methods can be added to it. [3]

Field Selection:

- In express-validator, a **field** is any value that is either sanitized / validated. [4]
- Every function / value returned by express-validator references fields in some way. [4]

Syntax:

- A field's path is always a string. We use the path of a field to reference it. [4]
- Each word-like sequence of characters is a **segment**. They're similar to JS Object properties. [4]
- Fields nested under objects can be selected by separating two segments with a dot. [4]

Whole-Body Selection:

- The entire body of a request can be selected by omitting the field path or by using an empty string. [4]

Advanced Features:

- The **Wildcard (*)** can be used to apply the same rules to all keys of an object. [4]
- The wildcard can be used in place of any segment, which will correctly select all indices of the array or keys of the object it's located in. [4]
- Each matched field is returned as a different instance; that is, it's validated or sanitized independently from the others. [4]
- If the array or object that the wildcard is placed in is empty, then nothing is validated.
- **Globstars (**)** extend **Wildcards** to an infinitely deep level. They are used when there are a large number / unknown number of nested fields and we want to validate / sanitize all of them in the same way. [4]

Custom Validators & Sanitizers:

- A Custom Validator is a simple field that receives a value and some information about it and checks whether that value is valid or not. [5]
- A Custom Sanitizer transforms the value of a field. [5]
- Both of them can be asynchronous if necessary. [5]

Error Messages:

- Whenever a field value is invalid, an error message is recorded for it. Both standard validators and custom validators can be given custom error messages. [5]
- A field-level message is set when you create the validation chain. It's used as a fallback message when a validator doesn't override its error message. [5]

Schema Validation:

- Schemas are plain JavaScript objects that you pass to the **checkSchema() function**, where you specify which fields to validate as the keys, and the schema of the field as the value. [6]
- In turn, the field schemas contain the validators, sanitizers, and any options to modify the behavior of the internal validation chain. [6]

REFERENCES

[1] https://express-validator.github.io/docs/
[2] https://express-validator.github.io/docs/guides/getting-started/
[3] https://express-validator.github.io/docs/guides/validation-chain
[4] https://express-validator.github.io/docs/guides/field-selection
[5] https://express-validator.github.io/docs/guides/customizing
[6] https://express-validator.github.io/docs/guides/schema-validation