Validación de objetos en **Plugcore**

Plug Soluciones TIC S.L.
info@plugcore.com


https://github.com/plugcore/plugcore


https://www.linkedin.com/company/plugcore


https://twitter.com/plugcoreit

Validación de objetos

1. **Introducción**
2. **Decoradores**
3. **Creación de schema**
4. **Validación**

Introducción

- Cuando tratamos con entradas de datos poco fiables como por ejemplo http, es necesario asegurarnos que los datos que nos llegan cumplan con las estructuras internas de datos que esperamos.

- Para este y muchos otros pasos se creó JSON SCHEMA, que es muy parecido a XSD (XML Schema definition) pero para objetos JSON, que son nativos en javascript.

- Los schemas nos permiten definir una serie de restricciones de cómo debe ser un objeto para ser válido, por ejemplo indicando qué propiedades debe haber y de qué tipo.

Introducción

- Aquí podemos ver un ejemplo sencillo de JSON schema.

- La creación manual de estos tipos de archivos puede ser tediosa y además muchas veces estarán desactualizados ya que el código puede cambiar y olvidarnos de actualizar los schemas.

- Por este motivo hemos creado una serie de decorators que podemos usar en nuestras clases de modelo, que nos ayudarán a definir estos archivos de una manera mucho más amena y fácil de actualizar.

Decoradores

- Los decorators creados hasta el momento son:
  - `@IsString()` Para indicar que la propiedad es string
  - `@IsNumber` Para indicar que la propiedad es number
  - `@IsBoolean` Para indicar que la propiedad es boolean
  - `@IsArray` Para indicar que la propiedad es array, podemos definir el tipo del array haciendo referencia a una clase definida con estos decorators.
  - `@IsObject` Para indicar que la propiedad es un objeto. Podemos hacer referencia a clases definidas con estos decorators
  - `@Required` Para indicar que la propiedad es obligatoria
  - `@ExtendsSchema` Nos permite hacer que una clase extienda del modelo de otra clase creada anteriormente.

Creación de schema

```
class Coordinates {
    @Required() @IsNumber()
    lat: number;
    @Required() @IsNumber()
    long: number
}

class MapPoint {
    @IsString()
    name?: string
    @Required() @IsObject(Coordinates)
    coordinates: Coordinates;
}
```

Creación de schema

- Con el ejemplo anterior podemos generar el json schema de la siguiente manera

```
// En formato objeto
const mapPointSchmea = ObjectValidatorUtils.generateJsonSchema(MapPoint);
console.log(JSON.stringify(mapPointSchmea))

// En formato array de objetos
const mapPointSchmeaAsArray = ObjectValidatorUtils.generateJsonSchema(MapPoint, {
asArray: true });
console.log(JSON.stringify(mapPointSchmeaAsArray));
```

Validación

- El punto anterior nos crearía el Objeto JSON schema, pero aún nos faltaría un sistema para validar los objetos, para ello existe el servicio "`ObjectValidator`" que podemos importarnos en cualquier servicio, y que ya se encargará de todo el proceso. por ejemplo:

```
@Service()
class ValidatorService {
    constructor(private objectValidator: ObjectValidator ) {}
    public validateMapPoint(object: MapPoint) {
        const mapPointValidator = this.objectValidator.
            createValidatorFromClass(MapPoint);
        const validationResult = mapPointValidator(object);
        return validationResult;
    }
}
```

Validación

- Y ya podemos ver un ejemplo de validación:

```
const validator = await Container.get(ValidatorService);
const validObject: MapPoint = { name: 'ex', coordinates: { lat: 1.6, long: 35.9 } };
const invalidObject: any = { name: 1, coordinates: { lat: 1.6 } };

console.log(JSON.stringify(
    validator.validateMapPoint(validObject), undefined, '\t'
));
console.log(JSON.stringify(
    validator.validateMapPoint(invalidObject), undefined, '\t'
));
```

- Ver más en: https://github.com/plugcore/plugcore/wiki/Validador-de-objectos

GRACIAS POR SU TIEMPO

Para más información
info@plugcore.com
germanml@plugcore.com
sergiolc@plugcore.com

plugcore.com