Transformación 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

Transformación de objetos

1. **Introducción**
2. **Cómo usarlo**
3. **Ejemplo avanzado**

Introducción

- La transformación de objetos (Object mapping) nos ayuda a la hora de comunicarnos con servicios externos, ya que normalmente las estructuras de los objetos internos y los objetos de los servicios es muy diferente.

- Lo que nos permite es que en vez de crear una función en la que hacemos la transformación manualmente, simplemente tenemos que crear un objeto que es una definición de la transformación

- Una vez esta definición está hecha, ya podemos usarla para pasarle objetos con la estructura de entrada (source) y que lo convierta a un objeto con la estructura de salida (target)

Cómo usarlo

- Si tenemos por ejemplo un objeto de entrada (source) de esta estructura

```
const sourceObject = {
    sourceString: 'value1',
    sourceNumber: 2,
    sourceObject: {
            otherString: 'value3'
    }
};
```

- Y queremos un objeto de salida (target) con esta estructura

```
const targetObject = {
    myProps: {
        stringProp1: 'value1',
        numProp: 2,
        stringProp2: 'value3'
    }
};
```

Cómo usarlo

- Lo que tenemos que hacer es importarnos la dependencia "`ObjectMapper`" y ejecutar lo siguiente:

```
// El Object mapper es un servicio que nos podemos importar
const objectMapper = await Container.get(ObjectMapper);
// Objeto de definición de la transformación
const mapping = {
    sourceString: 'myProps.stringProp1',
    sourceNumber: 'myProps.numProp',
    ['sourceObject.otherString']: 'myProps.stringProp2'
};
// Aplicamos transformación
const targetObject = objectMapper.map(mapping, sourceObject);
// Ya podemos usar el objeto
console.log(JSON.stringify(targetObject, undefined, '\t'));
```

Ejemplo avanzado

- Hay algunas utilidades más como por ejemplo tratamiento de arrays, funciones de transformación y usar un atributo de entrada para más de un atributo de salida, vamos a ver un ejemplo que implemente todo estos casos.

- Si tenemos por ejemplo un objeto de entrada (source) de esta estructura

```
const sourceObject = {
    sourceArray: [
        {stringProp: 's1'},
        {stringProp: 's2'}
    ],
    sourceString: 'base'
};
```

- Y queremos un objeto de salida (target) con esta estructura

```
const targetObject = {
    firstArrayString: 's1',
    allArrayStrings: ['s1', 's2'],
    targetString: 'base',
    targetString1: 'base1'
};
```

Ejemplo avanzado

- Para poder realizar esta transformación vamos a utilizar un objeto de transformación cómo:

```
const mapping = {
    // Podemos hacer referencia a posiciones del array
    ['sourceArray[0].stringProp']: 'firstArrayString',
    // Ó podemos hacer referencia a un atributo de los objetos del array
    ['sourceArray[].stringProp']: 'allArrayStrings',
    // Como valor de un atributo podemos definir un array,
    // para que se aplique más de 1 transformación
    sourceString: [
        // Transformación simple a un atributo de salida
        'targetString',
        // Si ponemos un objeto como este, podemos definir
        // una función de transformación
        {  key: 'targetString1', transform: (val: string) => `${val}1` }
    ] };
```

Ejemplo avanzado

- Además podemos guardar esa transformación en una función para reutilizar cuando queramos sin necesidad de hacer referencia al objeto de mapping:

```
// Esto lo podemos definir en algún sitio de la clase
private customMapping: ObjectMapping<SourceObject, TargetObject>;
// Y después en el constructor por ejemplo, creamos la función
this.customMapping = this.objectMapper.createMapping(mapping);
// Y ya podemos usarla donde queramos
const sourceObject = {
    sourceArray: [
        {stringProp: 's1'},
        {stringProp: 's2'}
    ],
    sourceString: 'base'
};
const targetObject = this.customMapping(sourceObject);
console.log(JSON.stringify(targetObject, undefined, '\t'));
```

GRACIAS POR SU TIEMPO

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

plugcore.com