Servicios REST
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

Servicios REST

1. **Introducción**
2. **¿Cómo registrar una ruta?**
3. **Objetos Request y Response**
4. **Configuración**

Introducción

- Todo lo que vamos a comentar hoy está en el paquete @plugcore/web, con lo cual si no lo tenemos instalado lo vamos a necesitar.

- Internamente usa Fatify, que a día de hoy ofrece uno de los frameworks rest más ligeros, flexibles y rápidos que hay.

- Tiene integración con el sistema de logging, con lo cual podremos tratar de la misma manera los logs de nuestra app que los que queramos que genere fastify.

- Por defecto sólo soporta content type json y text/plain, pero se le puede agregar soporte para otros, por ejemplo plugcore ha agregado un módulo para form multipart.

¿Cómo registrar una ruta?

- Para registrar rutas lo primero que necesitamos es tener una clase servicio del inyector de dependencias, pero en vez de estar decorada con `@Service()` deberá tener el decorador `@Controller()`.

- Dentro de ese servicio podremos decorar a cualquier método con uno de los decoradores de ruta: `@Get()`, `@Post()`, `@Patch()`, `@Delete()`, etc. Esto hará que se registre una ruta con ese método http. Dentro de cada uno de estos controladores podríamos definir la ruta que debería registrar, por ejemplo '/create'.

- A nivel de `@Controller()` podremos definir una url base para todas las rutas que se vayan a registrar dentro, por ejemplo si es un controlador de usuarios podría ser: '/api/users'.

¿Cómo registrar una ruta?

- Vamos a ver un ejemplo sencillo, aquí simplemente vamos a registrar la ruta '/api/example/say-hello' que devolverá un string.

```
@Controller({ urlBase: '/api/example' })
export class MyController {
    constructor(@InjectLogger() private log: Logger) {}

    @Get('/say-hello')
    public async sayHello() {
        this.log.info('Api called!')
        return 'Hello there!'
    }

}
```

Objetos Request y Response

- Dentro de los métodos que registran rutas (Get, Post, etc.) vamos a recibir 2 parámetros:
    - **Request**: Representa toda la información que nos está llegado a esa ruta, como por ejemplo el body de la petición, headers, etc.
    - **Response**: Representa la respuesta que vamos a devolver, normalmente no hacer falta usar este objeto ya que toda la implementación de la respuesta se gestiona automáticamente, pero por ejemplo podremos devolver algunos códigos HTTP especiales, o por ejemplo devolver un header.

Objetos Request y Response

- Antes de seguir hay que mencionar que además de poder definir rutas cómo *'/say-hello'*, podemos definir **parámetros en la url**, como por ejemplo el id de una entidad: *'/api/article/***:articleId**', esto hará que "articleId" esté disponible en la request.

- Podemos ver un mejor detalle de todas las propiedades de la request en la documentación de fastify, pero lo más importante sería:
  - **query**: Un objeto que representa los parámetros que llegan por el query string (ej: "*?param1=val1&param2=val2*" => { *param1: 'val1', param2: 'val2'* })
  - **body**: El cuerpo de la petición, por defecto si es un JSON ya vendrá parseado.
  - **params**: En caso de definir parámetros por la url
  - **headers**: Los headers de la petición

Objetos Request y Response

- Ejemplo de request y response:

```
@Controller({ urlBase: '/api/article' })
export class ArticleController {
    constructor(@InjectLogger(ArticleController.name) private log: Logger) {}
    @Patch('/:articleId')
    public async editArticle(req: Request, res: Response) {
        this.log.info('<Query>: ' + JSON.stringify(req.query, null, '\t'));
        this.log.info('<Body>: ' + JSON.stringify(req.body, null, '\t'));
        this.log.info('<Params>: ' + JSON.stringify(req.params, null, '\t'));
        this.log.info('<Headers>: ' + JSON.stringify(req.headers, null, '\t'));
        res.code(202);
        res.header('custom', 'header');
        return { txt: 'Hello there!' };
    }
}
```

Configuración

- Al igual que con otras partes de plugcore, se puede usar el sistema de configuración para configurar algunos aspectos de la ejecución del servidor web.

- El objeto en concreto lo podemos ver en aquí aunque lo que más nos va a interesar son las propiedades de:

```
{
    "web": {
        "server": {
            "port": 3000,
            "host": "localhost"
        }
    }
}
```

GRACIAS POR SU TIEMPO

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

plugcore.com