Tests
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

Tests

1. **Introducción**
2. **Ejecución básica**
3. **@BeforeTests() y @AfterTests()**
4. **Focused tests**

Introducción

- Para poder hacer los tests correspondientes sobre los servicios que creemos y además poder beneficiarnos de sistemas que ya existen en plugcore como la inyección de dependencias o el sistema de configuración, se ha creado un mini-framework nuevo de tests.

- Está basado en clases de Typescript decoradas, que son igual que los servicios que vemos en el inyector de dependencias.

- Todos los tests son async por defecto.

- Usa el módulo de Assert de NodeJs para indicar si los tests se están ejecutando correctamente.

Ejecución básica

- Lo primero que tenemos que hacer es crear nuestra clase de test, para ello podemos utilizar la carpeta “test” que crea el cli de plugcore.
- Dentro de esta carpeta crearemos un archivo ts, por ejemplo “example.test.ts”
- Para ejecutar los tests podemos ejecutar el comando “**npm test**” que ya crea el cli, esto lo que hará es compilar el código con “tsc” y después buscará todos los tests que estén así anotados dentro de la carpeta de “test” y los ejecutará.
- A continuación enseñamos un ejemplo sencillo de test:

Ejecución básica

```
@TestService()
// Además para tener acceso a la propiedad "assert" debemos extender de esta clase
export class ExampleTest extends AsserterService {
    constructor(
        // Esta clase es un servicio normal y podemos importar lo que sea necesario
        private log: Logger
    ) { super(); }
    // Todos los métodos que queramos que se ejecuten deben anotarse con "Test"
    @Test()
    public async determineModel() { // Todos los tests pueden ser "async"
        this.log.info('My first test');
        this.assert.equal(1, 1);
    }

}
```

Ejecución básica

- Al ejecutar el test anterior nos saldrá el siguiente output:
- La sección de **Registered test** indica un listado de todas las clases y métodos de tests encontrados
- **Tests results** indica todas las assertions ejecutadas.
- **Final results** es el recuento final.

```
------------------------------------------------
-- Registered tests
------------------------------------------------

ExampleTest
- firstTest

------------------------------------------------
-- Tests execution
------------------------------------------------

[2020-05-11 08:01:22.598 +0000] INFO : My first test
* ExampleTest * (ok)

------------------------------------------------
-- Tests results
------------------------------------------------

ExampleTest: ( 1 successful tests ) and ( 0 test with errors )
- firstTest: ( 1 successful assertions ) and ( 0 assertions with errors )

--- Final results ---
Execution time: 52 ms
Tests with errors: 0
Successful tests: 1
Tests result: Success
--------------------
```

@BeforeTests() y @AfterTests()

- Hay muchos casos en los que nos puede ser interesante inicializar algunas variables antes de la ejecución principal de nuestros tests, por ejemplo conectarnos a algún servidor, crear algún objeto de ayuda o parecido.

- De igual manera, después de ejecutar los tests podemos querer ejecutar alguna acción como eliminar elementos creados en una base de datos.

- Para ello existen @BeforeTests() y @AfterTests(), que nos permiten decorar métodos asíncronos para que se ejecuten antes que ningún test de esa clase.

@BeforeTests() y @AfterTests()

```
@TestService()
export class ExampleTest extends AsserterService {
    private myString: string;
    constructor(private log: Logger) { super(); }
    @BeforeTests()
    public async beforeTests() { this.myString = 'My first test'; }
    @AfterTests()
    public async afterTests() { this.log.info('After tests'); }
    @Test()
    public async firstTest() {
        this.log.info(this.myString);
        this.assert.equal(1, 1);
    }

}
```

Focused tests

- Cuando estamos creando alguna funcionalidad nueva, normalmente no queremos que se nos ejecuten todos los tests posible ya que ya sabemos que funcionan, y sólo queremos ejecutar los tests sobre los que estamos trabajando.

- Para ello, en cualquier decorador "`@TestService()`" ó "`@Test()`" podemos pasarle un objeto de configuración que indique que sólo queremos ejecutar los tests dentro de esta clase, o este test en concreto: "`{ testThisOnly: true }`"

Focused tests

```
@TestService()
export class ExampleTest extends AsserterService {
    constructor(private log: Logger) { super(); }

    @Test()
    public async firstTest() {
        this.log.info('First');
        this.assert.equal(1, 1);
    }
    @Test({ testThisOnly: true })
    public async secondTest() {
        this.log.info('Second'); // Sólo este se ejecutará
        this.assert.equal(1, 1);
    }
}
```

GRACIAS POR SU TIEMPO

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

plugcore.com