Eventos
en **Plugcore**

plugcore.com

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

https://github.com/plugcore/plugcore

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

https://twitter.com/plugcoreit

Eventos

1. **Introducción**
2. **Ejemplo con funciones**
3. **Importancia de los contextos**
4. **Eventos con servicios**

Introducción

- El sistema de eventos es una forma de comunicación desacoplada entre diferentes partes de nuestra aplicación.
- Las diferentes partes de plugcore pueden emitir diferentes eventos, por ejemplo el evento de que todos los servicios se han cargado, que puede ser útil si queremos hacer algo cuando la aplicación ya está lista.
- Está basada en la API de NodeJS de eventos, pero hay sólo un eventemitter para toda la aplicación.
- Para mejorar el rendimiento se ha utilizado la librería de eventemitter3.

Introducción

- La forma tradicional de comunicación en Javascript es simplemente llamar a una función pasando parámetros, por ejemplo si tenemos 2 objetos de 2 clases diferentes que se quieren comunicar entre ellos, primero de todo un objeto debe tener una referencia al otro objeto, y después llamar a un método de ese objeto.

- En cambio con los eventos, en cualquier parte de nuestra aplicación podemos lanzar un evento con un objeto asociado, y otra partes de la aplicación podrán escuchar ese evento si lo necesitan.

Ejemplo con funciones

```
// 1: Necesitamos el event dispatcher, que es el encargado de controlar
// la ejecución de todos los eventos.
const eventDispatcher = await Container.get(EventDispatcher);
// 2: Función que se va a ejecutar cuando pase un determinado evento
const listenerFunc = (name: string) => { console.log(`Nuevo usuario creado: ${name}`)
};
// 3: Registramos la función para que se ejecute en el evento
eventDispatcher.on(
    // Nombre del evento
    'newUser',
    // Función que se va a ejecutar
    listenerFunc
);
// 4: lanzamos el evento
eventDispatcher.emit('newUser', 'Jack');
```

Importancia de los contextos

- El ejemplo anterior funciona correctamente ya que es una función simple que no se ejecuta dentro de ningún contexto, por ejemplo el objeto de una clase.

- Si pasamos como “listener” de un evento una función de una clase, veremos que si ejecutamos “this” dentro de esa clase obtendremos valores inesperados. Esto se puede arreglar pasando la función con un “.bind(this)”, pero su ejecución va a ser relativamente más lenta, por ello podemos pasar el objeto de contexto que queramos. Ejemplo:

```
class ListenerClass {
    constructor(private appName: string) {}
    public listenerFunc(name: string) {
        console.log(`Nuevo usuario creado: ${name} en la app ${this.appName}`);
    }
}
```

Importancia de los contextos

```
// 1: Necesitamos el event dispatcher como antes
const eventDispatcher = await Container.get(EventDispatcher);
// 1: Creamos una instancia de nuestra clase
const listener = new ListenerClass('Test');
// 3: Registramos la función para que se ejecute en el evento
eventDispatcher.on(
    // Nombre del evento
    'newUser',
    // Función que se va a ejecutar
    listener.listenerFunc,
    // Contexto, que será el objeto que hemos creado
    listener
);
// 4: lanzamos el evento
eventDispatcher.emit('newUser', 'Jack');
```

Eventos en servicios

- Para hacer aún más sencillo este sistema se han creado un decorador que se puede aplicar a las funciones de los servicios para que esa función sea ejecutada en determinado evento.

```
@Service()
export class ListenerService {
    constructor(private log: Logger) {}
    @OnEvent('newUser') // Aquí ya registramos esta función para el evento
    public listenerFunc(name: string) {
        this.log.info(`Nuevo usuario creado: ${name}`);
    }
}
```

Eventos en servicios

```
// 1: Necesitamos el event dispatcher como antes
const eventDispatcher = await Container.get(EventDispatcher);
// 2: Vamos a aprovecharnos de un evento que lanza plugcore
// que nos indica que todos los servicios están listos
// para que así podamos lanzar el evento
eventDispatcher.on(PublicEvents.allServicesLoaded, () => {
    // 4: lanzamos el evento
    eventDispatcher.emit('newUser', 'Jack');
});
```

- Ver más en: https://github.com/plugcore/plugcore/wiki/Eventos

GRACIAS POR SU TIEMPO

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

plugcore.com