02

O pattern publisher/subscriber

Transcrição

O pattern *Publish/Subscribe* cria uma barramento muitas vezes chamado de *Broker*. Este barramento recebe mensagens dos publishers e as reencaminha para subscribers cadastrados. Com este pattern, estaremos acoplados ao barramento que possui apenas dois métodos:

- *publish(topico, dado)*: notifica todos que estejam inscritos ao tópico passado como primeiro parâmetro. O segundo parâmetro é qualquer informação que queiramos fornecer para os inscritos.
- *subscribe(topico)*: realiza a inscrição para um tópico. É possível haver um ou mais inscritos e todos eles serão notificados toda vez que houver uma publicação para este tópico.

No entanto, utilizamos outra nomenclatura, a mesma utilizada pela plataforma Node.js.

Sobre EventEmitter

Nessa plataforma, há a classe `EventEmitter` que implementa o pattern *Publisher/Subscribe*. Ele possui os métodos `emit` e `on` que equivalem respectivamente aos métodos `publish` e `subscribe`. Em suma, faremos um simples clone do `EventEmitter` para o browser.

No context do nosso problema, queremos no final um código nesta escrutura:

```
// app/app.js
// código anterior omitido
// exemplo apenas, ainda não entra em nosso código
const action = operations(() =>
    retry(3, 3000, () => timeoutPromise(1000, service.sumItems('2143')))
    .then(total => EventEmitter.emit('itensTotalizados', total))
    .catch(console.log)
);
// código posterior omitido
```

Reparem que não há mais a chamada de `.then(sumItemsWithCode)`, pois todo o código que lida com as notas também será isolado de `app/app.js` além do código que renderiza uma lista com as datas de todas as notas. Se tivermos outros interessados no tópico `itensTotalizados`, basta o interessado se inscrever neste tópico para ser notificado toda vez que houver uma publicação. Vejamos um exemplo:

```
// módulo A
EventEmitter.on('itensTotalizados', total => console.log(total);

// módulo B
EventEmitter.on('itensTotalizados', total => alert(total);
```

Os módulos `A` e `B` executaram operações distintas para uma mesma publicação.

Agora que já temos uma visão geral do pattern, vamos implementá-lo.