Angular inifinite loop calling function

Asked 1 year, 6 months ago Active 1 year, 6 months ago Viewed 110 times

1

In a few Angular projects I have the same problem, whenever I try to call a function inside my HTML (that retrieves some value from the api) it triggers an infinite loop. In the example below it's getUser() that triggers the loop.

HTML

```
<ul>
    <li *ngFor="let order of orders">
        {{ getUser(order.orderNr).emailAddress }}
    </li>
</ul>
```

component

```
private getOrdersList() {

    this.orderService.getAll().subscribe(
        orders => {
            this.orders = orders;
        }
    );
}

public getUser(orderNr: number) {
    return this.orderService.getUser(orderNr);
}
```

service

```
public getAll(): Observable<Order[]> {
    return this.api.get<Order[]>('orders');
}

public getUser(orderNr: number) {
    return this.api.get<void>('orders/'+orderNr);
}
```

I think it has something to do with the way Angular handles data but I'm fairly new to Angular and unsure how to retrieve this data without causing the loop. Perhaps someone more experienced can provide some help?

javascript angular

asked Jan 13 '18 at 16:05

CS_student
42 2 8

1 Answer

2

It's not an infinite loop, it's just Angulars change detection.

In development each change detection run also follows a 2nd turn. Change detection is run when any async call completes (event handlers, timeout, ...) and can therefore happen quite often.

Binding to functions in the view should generally be avoided, instead assign the result to a field and bind to that field instead. Angular is extremely efficient checking whether field values have changed.

answered Jan 13 '18 at 16:09

Günter Zöchbauer
357k 82 1132 1033

Alright, that makes sense but I'm not sure how to accomplish this considering I have to do this while looping through orders. – CS_student Jan 13 '18 at 17:31

In `getOrderList` when you receive the orders, iterate over them and put them into an array like `this.orders = orders.map(order => { order: order, user: getUser(order.orderId) });` and then in the binding use `{{ order.user.emailAddress }}` – Günter Zöchbauer Jan 13 '18 at 17:35

I used this: this.orders = orders.map(order => { return { order: order, user: this.getUser(order.orderNr)}; }); Which sounds like a great solution but unfortunately it doesn't seem to work for me. User remains undefined and order is no longer visible in my html. – CS_student Jan 13 '18 at 19:05

1 Order is actually visible by just using {{ order.order.orderNr }}. – CS_student Jan 13 '18 at 20:27

1 Yes exactly. The reason I thought your method wasn't working was also because my html showed intake.intake.attribute was false even though it was working. – CS_student Jan 13 '18 at 21:05