Casting results from Observable.forkJoin to their respective types in Angular 2

Asked 2 years, 2 months ago Active 1 year, 7 months ago Viewed 9k times

13

2

Let say I have a component in Angular 2 that needs to load 2 different things from the server before the page is displayed. I'd like all of those things to fire off and call one event handler when they come back telling the page isLoaded = true. Let's say I have a service class that looks like this.

```
export class MyService {
   getStronglyTypedData1(): Observable<StrongData1[]>{
      return this.http.get('http://...').map((response:Response) =>
<StrongData1[]>response.json());
   }
   getStronglyTypedData2(): Observable<StrongData2[]>{
         return this.http.get('http://...').map((response:Response) =>
<StrongData2[]>response.json());
   }
}
```

Then I have a component that uses that service class like this.

```
export class MyComponent implements OnInit {
   isLoaded = false;
   stronglyTypedData1: StrongData1[];
   stronglyTypedData2: StrongData2[];

   constructor(private myService:MyService){ }

   ngOnInit(){
      var requests [
         this.myService.getStronglyTypedData1(),
         this.myService.getStronglyTypedData2()
      ];
      Observable.forkJoin(requests).subscribe(
         results => {
            this.stronglyTypedData1 = results[0];
```

```
}
```

The TypeScript compiler is complaining that it cant convert type object to type StrongData1[]. If I change StrongData1 and StrongData2 to "any", everything works fine. I'd rather not do that though because I'm losing the benefit of TypeScript's strong typings.

How do I cast the results from forkJoin to their respective types?

angular typescript rxjs

asked Jul 25 '17 at 14:30

Chris Lees
1,209 2 17 37

2 Answers

20

for me it always works when i add the requests directly to the Observable.forkJoin and then use es6 destructing for the result array.

so your code could look like this

```
Observable
    .forkJoin(this.myService.getStronglyTypedData1(),
this.myService.getStronglyTypedData2())
    .subscribe(
        ([typeData1, typeData2]) => {
            this.stronglyTypedData1 = typeData1;
            this.stronglyTypedData2 = typeData2;
            this.isLoaded = true;
        }
    );
```

answered Jul 25 '17 at 14:46

Nicolas Gehlert
1,388 11 30

`this.myService.getStronglyTypedData2()` then `Observable.forkJoin(request1, request2).subscribe(results => { type1 = results[0];`

`type2 = results[1] })` or es6 destructing `Observable.forkJoin(request1, request2).subscribe(([type1, type2]) => { type1 = type1; type2 = type2})` – locnguyen Aug 17 '17 at 19:25

2 In case you're like me and you don't see it: don't pass in an array to `forkJoin([typedReq1, typedReq2])` , instead directly add the observables: `forkJoin(typedReq1, typedReq2)` and the ES6 destructing variables will be typed. – Ranil Wijeyratne Dec 8 '17 at 20:15

This appears to be deprecated in RxJs 6 – crush May 28 at 18:43

With rxjs 6 you should directly use forkJoin import from rxjs instead of Observable.forkJoin – Nicolas Gehlert May 28 at 19:07

try

3

```
(results:[StrongData1[], StrongData2[]]) =>
```

answered Jul 25 '17 at 14:45

kit

3,895 2 17 19

it throw exception like "results must be StrongData1[]". you can write Array<StrongData1 | StrongData2> . But @Nicolas-Gehlert answer work fine. – Alexey Prokopenko Oct 5 '17 at 9:19