

# Angular Observable, sharing data within components

Asked 1 year, 1 month ago   Active 1 year, 1 month ago   Viewed 855 times



I'm making my first App on Angular. I trying to filter a list of book objects by the gender attribute. I am having difficulty sharing data between components: the `filteredData` variable and the list of books `FireBaseList` .

2



I am trying to pass the `filteredData` variable from `fixed-nav.component` to `books-grid.component` . From my research, I know I need to make the `books-grid.component` "bookList" observable of any changes and have the `bookList` on `fixed-nav.component.ts` emit an event on any changes on `this.bookList`



1

However, I am unable to achieve this. I know there are many questions about observables already but none use Firebase. Hope anyone out there can help me, THANKS!!

fixed-nav.component that I use as filter

```
import { Component, OnInit } from '@angular/core';
import { BookService } from '../services/book.service';
import { Book } from '../models/book';

@Component({
  selector: 'fixed-nav',
  templateUrl: './fixed-nav.component.html',
  styleUrls: ['./fixed-nav.component.css']
})
export class FixedNavComponent implements OnInit {

  Genders: string[];
  bookList: Book[];

  constructor(private bookService: BookService) {

    this.Genders = ["Historia", "Literatura", "Infantil",
    "Juvenil", "Poesía", "Narrativa"];
  }

  filterByGender(gender: string){

    this.bookService.getBooks() // FIREBASE
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

    item.forEach(element => {
      let x = element.payload.toJSON();
      x["$key"] = element.key;
      this.bookList.push(x as Book)
    })
    const filteredData = this.bookList.filter( function(element){
      return element.gender == gender;
    });
    return filteredData; // I WANT TO SHARE THIS DATA
  })
}

ngOnInit() {
}
}

```

### books-grid.component.ts

```

import { Component, OnInit } from '@angular/core';

import { BookService } from '../services/book.service';
import { Book } from '../models/book';

@Component({
  templateUrl: './books-grid.component.html',
  styleUrls: ['./books-grid.component.css']
})

export class BooksGridComponent implements OnInit {

  bookList: Book[];

  constructor(private bookService: BookService) {}

  ngOnInit() {
    this.bookService.getBooks() // FIREBASE
      .snapshotChanges()
      .subscribe(item => {
        this.bookList = [];
        item.forEach(element => {
          let x = element.payload.toJSON();
          x["$key"] = element.key;

```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
    })  
  }  
}
```

book.service.ts

```
import { Injectable } from '@angular/core';  
  
import { AngularFireDatabase, AngularFireList } from 'angularfire2/database';  
  
import { Book } from '../models/book'; // Model  
  
@Injectable({  
  providedIn: 'root'  
})  
  
export class BookService {  
  
  bookList: AngularFireList<any>;  
  selectedBook: Book = new Book(); // Temporally save selected Book  
  
  constructor(private firebase: AngularFireDatabase) { }  
  
  getBooks(){  
    return this.bookList = this.firebase.list('books');  
  }  
}
```

books.model.ts

```
export class Book {  
  $key: string;  
  title: string;  
  author: string;  
  gender: string;  
  language: string;  
  image: string;  
  likes: number;  
  price: number;  
  new: boolean;  
}
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

edited Jul 15 '18 at 0:48

asked Jul 14 '18 at 21:47



coder

4,056

12

23

36



Sergi

115

1

8

Think about moving any filtering code, or shared lists into the `BookService`. That way, you can share the full or filtered list from there. Any component that uses the `BookService` can use them. – [R. Richards](#) Jul 14 '18 at 22:31

## 2 Answers



4



In your `book.service.ts` create a `filteredData` variable and function to set `filteredData` like below. Import `Subject` using `import { Subject } from 'rxjs';`

```
filteredData = new Subject<any>();

setFilteredData(data) {
  this.filteredData.next(data);
}
```

And then in your `fixed-nav.component.ts` after done filtration set that filtered data like below


```
this.bookService.setFilteredData(filteredData);
```

Finally subscribe to the `filteredData` in `book.service.ts` from `books-grid.component.ts` like below and assign the data into variable that you want.

```
constructor(private bookService: BookService) {
  bookService.filteredData.subscribe((data)=>{
    // Assign the data into variable that you want e.g this.filteredData = data;
  })
}
```

Hope this will helps you!

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

- 
- 1 thanks a lot, this is working! gonna try the option above too, do you think it its better option? so i can use that service in any other component – [Sergi](#) Jul 15 '18 at 1:31
- 
- 1 @Sergi glad to hear that :) . yes, you could use this service other component as well it will work fine – [coder](#) Jul 15 '18 at 3:00
- 
- hey i have a problem, everytime i subscribe to changes, it stacks,like its subscribing to all events, how to instance Subject<any>() so it only subscribe once – [Sergi](#) Jul 16 '18 at 12:30
- 
- 1 implements OnDestroy on your component, and inside ngOnDestroy() unsubscribe from the observable. e.g. first declare a subscription: ISubscription; variable and assign this.subscription = bookService.filteredData.subscribe((data)=>{ }) and inside ngOnDestroy() use like this this.subscription.unsubscribe(); – [coder](#) Jul 17 '18 at 2:43 
- 
- 1 thanks a lot, you just rocks! – [Sergi](#) Jul 17 '18 at 3:06
- 

you could utilize another shared service, where you update the booklist and then **subscribe** to any change event of this filtered booklist from your **BooksGridComponent** or any other component in need. I edited your code a little bit.

1

**Something like this:**

Create a new Service, where you return the filtered BooksList as an Observable, so any component can subscribe to it

```
import { Injectable } from '@angular/core';
import { Observable, Subject } from 'rxjs'; // For rxjs 6

@Injectable({
  providedIn: 'root'
})
export class SharedDataService {

  private filteredBookList: Subject<Book[]> = new Subject<Book[]>();

  constructor() {}

  public getFilteredBookList(): Observable<Book[]> {
    return this.filteredBookList.asObservable();
  }

  public setFilteredBookList(books: Book[]): void {
    this.filteredBookList.next(books);
  }
}
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

**fixed-nav.component.ts:**

Here you simply call the SharedDataService and set the filteredBookList.

```
this.sharedDataService.setFilteredBookList(filteredData);
```

**And now in your books-grid.component.ts:**

Here you are going to subscribe to any changes of the filteredBookList

```
this.sharedDataService.getFilteredBookList().subscribe((books: Book[]) => {  
  // Do Something with the filtered BookList from SharedDataService  
})
```

edited Jul 15 '18 at 21:08

answered Jul 15 '18 at 1:03



sagat

423 4 9

---

having some troubles importing 'rxjs/Rx'; – [Sergi](#) Jul 15 '18 at 2:14

---

It should work fine without that import. I edited the answer! – [sagat](#) Jul 15 '18 at 20:38

---