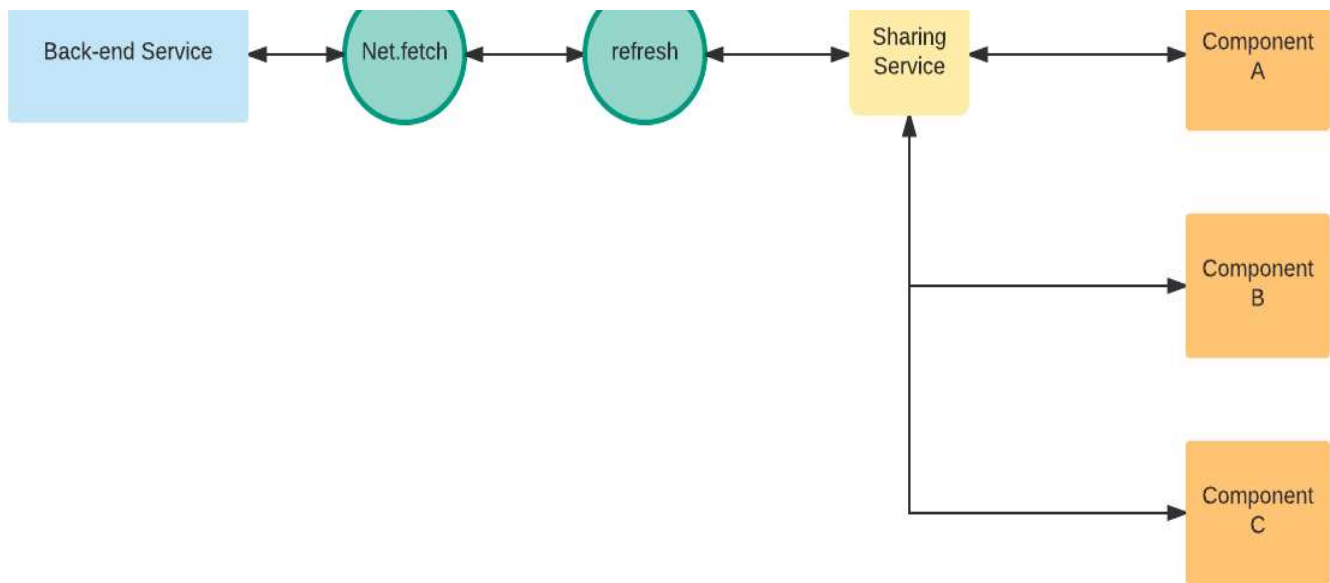


[Únete a la conversación en Slack](#)[Quiero Unirme!](#)[Todo](#) [Hecho En](#) [Tutoriales](#) [Noticias](#) [Tips](#)

Angular Y Observables: Como compartir información entre diferentes componentes de la aplicación de forma eficiente ?

tips angular2 observable rxjs reactive programming ChangeDetectionStrategy ChangeDetectionStrategy.OnPush performance - May 16, 2017 por [Javier González Rodríguez](#)
angular 4.0.0 rxjs 5.0.1

Una práctica común en las aplicaciones de **angular** para almacenar los datos para después ser utilizados en los diferentes componentes es guardarlo en sus **services** pues estos son inyectados como **singlenton**, hasta aquí vamos bien pues de cierta forma los datos no se repiten y tenemos un punto común de acceso a los datos desde todos los componentes. ¿Dónde nos llegan los dolores de cabeza? Es a la hora de propagar el cambio de estos datos a través de todos los componentes, normalmente lo hacemos utilizando los **events** pero conlleva a un esfuerzo bastante grande y de cierta forma la aplicación queda muy acoplada y frágil ante los cambios futuros ya sea por bugs o nuevos requerimientos. En este articulo intentaremos darle solución a esta problemática utilizando la **reactive programming** en particular la especificación para javascript **rxjs**.



¿Qué tal lo estamos haciendo? Tomate 10 segundos en contestar una pregunta :)

Ir a la pregunta

Actualización (31/05/2017)

Paso 1: Iniciando el proyecto e incluir semantic-ui



```
ng new angular-observable
cd angular-observable
```

en el `index.html` incluir las dependencias de semantic-ui para darle algún estilo al componente.



```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>ServicesObservable</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">

  <link type="text/css" rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic
-ui/2.2.2/semantic.min.css">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.2.2/semantic.min.js"></scr
ipt>
```

```

</head>

<body>
  <app-root>Loading...</app-root>
</body>

</html>

```

Paso 2: Crear un componente y el Servicio que se utilizará como contenedor de los datos



```

ng generate component user-list
ng generate service user

```

Paso 3: Componente UserList

user-list-component.html



```

<table class="ui compact celled definition table">
  <thead>
    <tr>
      <th></th>
      <th>Name</th>
      <th>Registration Date</th>
      <th>E-mail address</th>
      <th>Premium Plan</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let user of users">
      <td class="collapsing">
        <div class="ui fitted slider checkbox">
          <input type="checkbox" [checked]="user.isPremium"> <label></label>
        </div>
      </td>
      <td>{{ user.name }}</td>
      <td>{{ user.registration }}</td>
      <td>{{ user.email }}</td>
      <td>{{ user.isPremium }}</td>
    </tr>
  </tbody>
  <tfoot class="full-width">
    <tr>
      <th></th>
      <th colspan="4">
        <div class="ui right floated small primary labeled icon button" (click)="createUser($event)">
          <i class="user icon" ></i> Add User
        </div>
        <div class="ui small button" (click)="approveAll($event)">

```

```

        Approve All
      </div>
    </th>
  </tr>
</tfoot>
</table>

```

El componente tiene como entrada la lista de los **users** y tiene dos eventos de salida uno para crear un nuevo usuario y otro para aprobar que todos los usuarios tengan una cuenta Premium.

user-list-component.ts



```

import { Component, OnInit, Input, Output, EventEmitter, ChangeDetectionStrategy } from '@angular/core';

@Component({
  selector: 'user-list',
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class UserListComponent implements OnInit {
  @Input() users;
  @Output() onCreateUser: EventEmitter<any> = new EventEmitter();
  @Output() onApproveAll: EventEmitter<any> = new EventEmitter();

  constructor() { }

  ngOnInit() {
  }

  createUser() {
    this.onCreateUser.emit({
      name: 'Prueba',
      email: 'prueba@gmail.com',
      registration: 'May 11, 2016',
      isPremium: false
    });
  }

  approveAll() {
    this.onApproveAll.emit();
  }
}

```

Paso 4: User Service

En el servicio utilizaremos BehaviorSubject que representara el mecanismo mediante el cual se va a mantener sincronizado los datos.



```
import { BehaviorSubject, Observable } from 'rxjs/Rx';
import { Injectable } from '@angular/core';

interface IUser {
  name: string;
  registration: string;
  email: string;
  isPremium: boolean;
}

export const DUMMY_DATA = [
  {
    name: 'John Lilki',
    registration: 'September 14, 2013',
    email: 'jhlilk22@yahoo.com',
    isPremium: true
  },
  {
    name: 'Jamie Harington',
    registration: 'January 11, 2014',
    email: 'jamieharingonton@yahoo.com',
    isPremium: true
  },
  {
    name: 'Jill Lewis',
    registration: 'May 11, 2014',
    email: 'jilsewris22@yahoo.com',
    isPremium: true
  }
];

@Injectable()
export class UserService {
  private usersSubject = new BehaviorSubject([]);
  private users: IUser[];

  constructor() { }

  getUsers(): Observable<IUser[]> {
    return this.usersSubject.asObservable();
  }

  private refresh() {
    // Emitir los nuevos valores para que todos los que dependan se actualicen.
    this.usersSubject.next(this.users);
  }

  createNewUser(user: IUser) {
    /**
     * Evitar hacer this.user.push() pues estaríamos modificando los valores directamente,
     * se debe generar un nuevo array !!!.
     */
    this.users = [...this.users, user];
    this.refresh();
  }

  loadDummyData() {
```

```

        this.users = DUMMY_DATA;
        this.refresh();
    }

    approveAll() {
        /**
         * Evitar hacer un forEach e ir modificando cada property !!! this.users.forEach(user => user.isPremium = true);
         *
         * Pudieramos Utilizar el .map pues siempre nos retorna un nuevo array pero si olvidamos el Object.assign( {}, ... )
         * siempre estaríamos tomando la referencia del objeto en memoria y estaríamos modificando nuevamente el valor
         * original en vez de crear una nueva copia o version del dato.
         *
         */

        this.users = this.users.map(user => Object.assign({}, user, { isPremium: true }));
        this.refresh();
    }
}

```

Paso 4: App Component

app.component.html



```

<div class="ui container" style="margin-top: 5%">
    <user-list [users]="users$ | async" (onCreateUser)="createUser($event)" (onApproveAll)="approveAll($event)"></user-list>
</div>

```

app.component.ts



```

import { Observable } from 'rxjs/Rx';
import { UserService } from '../providers/user.service';
import { Component, ChangeDetectionStrategy } from '@angular/core';

@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css'],
    changeDetection: ChangeDetectionStrategy.OnPush
})
export class AppComponent {
    private users$: Observable<any[]>;

    constructor(private userService: UserService) { }

    ngOnInit() {
        this.users$ = this.userService.getUsers();
        this.userService.loadDummyData();
    }
}

```

```
}  
  
createUser(user) {  
  this.userService.createNewUser(user);  
}  
  
approveAll() {  
  this.userService.approveAll();  
}  
}
```

De esta forma no solo creamos un mecanismo que nos ayuda a mantener los datos de la aplicación de forma consistente, también desacoplamos la dependencia de los componentes y aumentamos el performace de la aplicación pues podemos desactivar el **ChangeDetectionStrategy** Cambiándolo a **OnPush** .

Resultado

	Name	Registration Date	E-mail address	Premium Plan
<input type="checkbox"/>	John Lilki	September 14, 2013	jhlilk22@yahoo.com	true
<input type="checkbox"/>	Jamie Harington	January 11, 2014	jamieharingonton@yahoo.com	true
<input type="checkbox"/>	Jill Lewis	May 11, 2014	jilsewris22@yahoo.com	true
<input type="button" value="Approve All"/>				<input type="button" value="Add User"/>

Te ha gustado el Post? No te olvides dejarnos una [Calificación](#) en Facebook y darle Like al [Facebook Page](#). Espero sea de utilidad y sigan programando :)

Por: [Javier González Rodríguez](#)

¡Compártelo!

13 Comentarios

Ng-Classroom

 1 Acceder ▾

 Recomendar

 Compartir

Ordenar por los mejores ▾



Únete a la conversación...

INICIAR SESIÓN CON

O REGISTRARSE CON DISQUS

Nombre



Dámaris Suárez Corrales • hace 5 días

Buenas Javier, muchas gracias por tu post, tengo una duda, esto no sirve para compartir data entre componentes en el caso de que navegues a otra ruta no? Es que estoy tratando de construir mi primera app en Angular y no sé qué usar en caso de navegar a traves de varios pasos de un mismo formulario, para mantener los datos que el usuario mete, pero sin guardarlos aun en BD, muchas gracias!

^ | v • Responder • Compartir ›



Javier González Rodríguez Moderador ➔ Dámaris Suárez Corrales • hace un día

Saludos.

Como los servicios son singlentos es posible utilizar la data en diferentes rutas y componentes, para este caso en especifico no te recomendaría usar varias rutas para los datos de un mismo formulario, sino un solo formulario reactivo con un wizard así mantienes la data dentro de un único objeto formulario y lo puedes validar y transformar a gusto con mayor facilidad.

si utilizas rutas tendrias que de alguna forma validar rutas, guardar la data por ejemplo en el local storage para cuando entre directo a un step que no sea el inicial tenga la data.

digamos algo así

```
this.formGroup = this.formBuilder.group({
  personalData: this.formBuilder.group({
    name: ['', Validators.required],
    lastName: ['', Validators.required],
    age: [16, [ Validators.required, Validators.min(16), Validators.minLength(2) ] ]
  }),
```

[ver más](#)

^ | v • Responder • Compartir ›



J_avila • hace 3 meses

Me gusto bastante y entendí este tema que me tiene algo... fastidiado.

Pero cómo podría aplicar por ejemplo para pasar de un servicio un valor (el curretnuser en localsotrage) al parent component?

^ | v • Responder • Compartir ›



Carlos Rojas Moderador ➔ J_avila • hace 3 meses



^ | v • Responder • Compartir »



^ | v • Responder • Compartir ›



^ | v • Responder • Compartir »



^ | v • Responder • Compartir ›



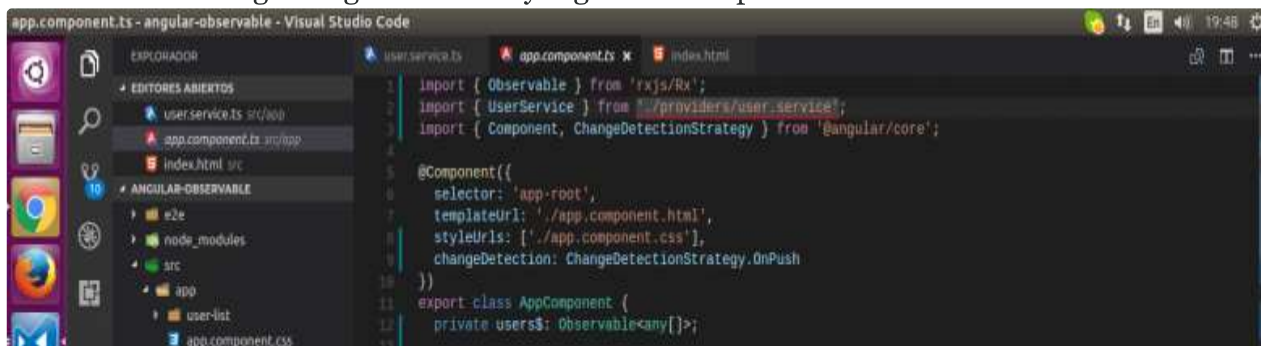
^ | v • Responder • Compartir ›

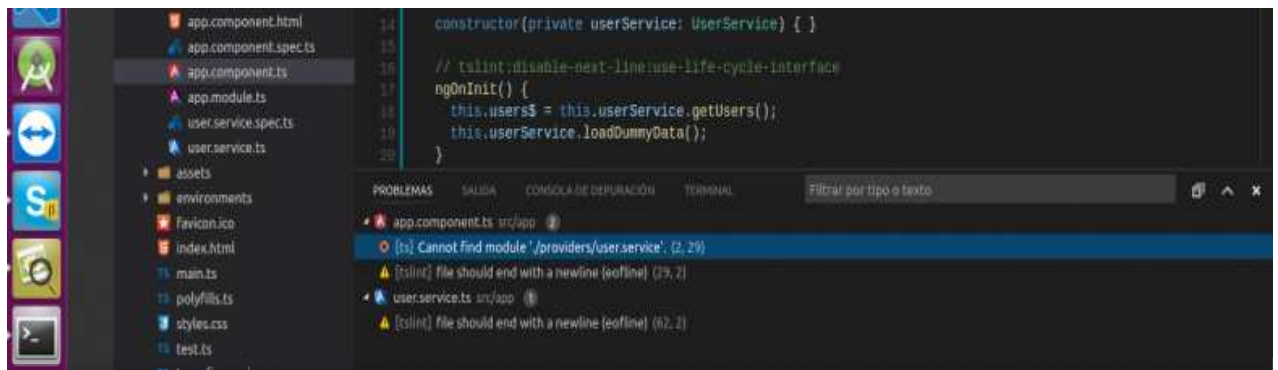


^ | v • Responder • Compartir ›



buenas a todos tengo el siguiente error y segui el tuto a perfeccion



[ver más](#)

^ | v • Responder • Compartir ›



Javier González Rodríguez Moderador ➔ Nestor Marquez • hace un año

Saludos.

En el post ya esta comentado como solucionar el bug que comentas, gracias por utilizar ion-book.com, cualquier duda nos comentas.

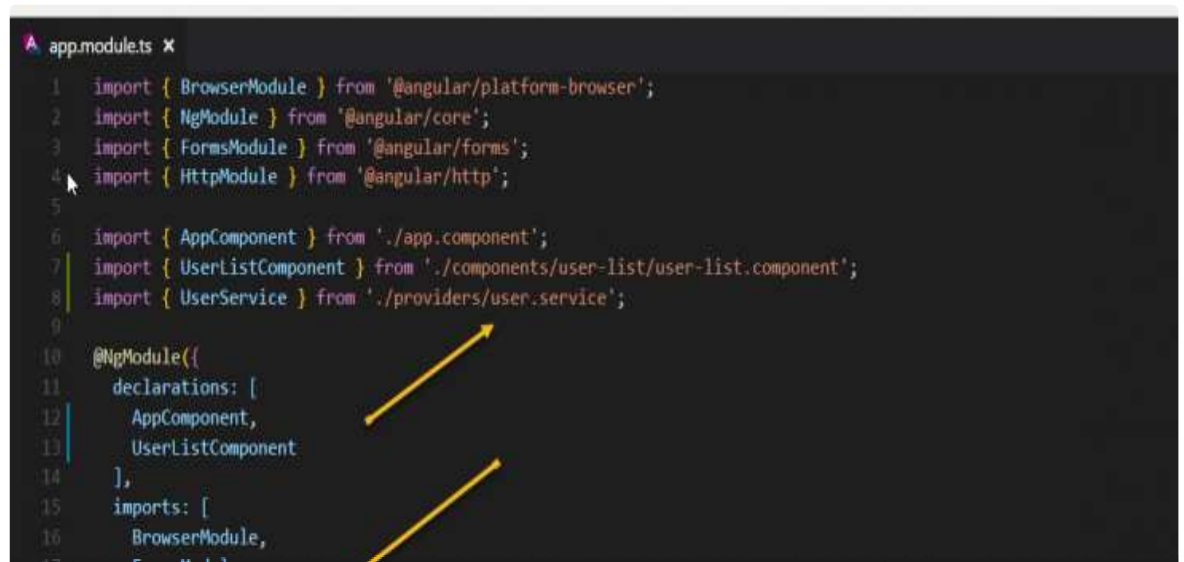
^ | v • Responder • Compartir ›



Javier González Rodríguez Moderador ➔ Nestor Marquez • hace un año

Saludos.

El servicio puedes ponerlo en en cualquier folder para el ejemplo quedaría de esta forma y debes activarlo en el app.module.ts. cualquier duda me comentas.

[ver más](#)

^ | v • Responder • Compartir ›



Daniel Barrios Cardoso ➔ Nestor Marquez • hace un año


Saludos,

Referencia el servicio desde la ubicación generada por el @angular/cli , e incluye-lo en el array de providers del componente.

```
import { Observable } from 'rxjs/Rx';
import { UserService } from './user.service';
import { Component, ChangeDetectionStrategy } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  providers: [UserService],
  changeDetection: ChangeDetectionStrategy.OnPush
})
```

^ | v • Responder • Compartir ›

✉ Suscríbete  Añade Disqus a tu sitio webAñade Disqus Añadir

 Política de privacidad de DisqusPolítica de privacidadPrivacidad

Artículos relacionados:



Micro Frontends.

Peticiones en paralelo con RXJS (ForkJoin)

Como Publicar tu App en App Store.



Únete a nuestra comunidad

Únete a nuestro canal de slack y podras interactuar con toda la comunidad hispana en ionic.

[Unete](#)

Somos un pequeño pedazo del Internet empoderado por la Comunidad de Ionic en Español. Aprende a crear apps con HTML, CSS y Javascript con nuestros Tutoriales, Libro, Slack y Publicaciones.

Inicio

Cursos

Autores

Podcast

Codigo de Conducta

