**this.userData$ = this.authService$.user$.pipe(**

**// 1. 사용자 객체가 변경될 때마다 새로운 내부 Observable로 전환**

**switchMap(firebaseUser => {**

**if (firebaseUser) {**

**const userUid = firebaseUser.uid;**

**const itemsRef = ref(this.db, `/${userUid}`);**

**return objectVal(itemsRef);**

**} else {**

**// 사용자가 로그아웃된 경우: null Observable을 반환합니다.**

**return of(null);**

**}**

**})**

**); >> Firebase API called outside injection context.**

**this.userData$ = objectVal(ref(this.db, 'qC53z7Q2F8QYZeGJEDQF8qlkz562')); is success ?**

**제공해주신 두 코드의 차이점을 분석해보면, 첫 번째 코드가 "Firebase API called outside injection context" 오류를 발생시키는 반면, 두 번째 코드는 성공하는 이유를 명확히 알 수 있습니다.**

**핵심은 objectVal() 함수가 실행되는 시점입니다.**

**🧐 오류 발생 원인 분석**

**❌ 오류 코드 (switchMap 내부)**

**TypeScript**

**this.userData$ = this.authService$.user$.pipe(**

**switchMap(firebaseUser => {**

**if (firebaseUser) {**

**// ...**

**const itemsRef = ref(this.db, `/${userUid}`);**

**// 💡 문제 발생 지점: objectVal()은 switchMap의 콜백 함수 내부에서 실행됨**

**return objectVal(itemsRef);**

**} else {**

**return of(null);**

**}**

**})**

**);**

1. **switchMap의 콜백 함수: switchMap의 인수로 전달된 함수는 RxJS 스트림에서 값이 방출될 때마다 실행되는 일반적인 JavaScript 함수입니다.**
2. **Injection Context 소멸: Angular의 Injection Context는 보통 서비스의 생성자나 컴포넌트의 필드 이니셜라이저가 실행될 때 활성화됩니다. 이 컨텍스트는 생성자가 완료되거나 비동기 작업이 시작되면 사라집니다.**
3. **지연된 호출: objectVal() 함수는 authService$.user$에서 값이 방출될 때 (즉, 비동기적으로 나중에) 실행됩니다. 이 시점은 Injection Context가 이미 사라진 후이므로, @angular/fire의 래퍼(objectVal 등)가 내부적으로 의존하는 Angular의 Zone.js 또는 Injection 로직이 제대로 작동하지 못해 "Firebase API called outside injection context" 오류가 발생합니다.**

**✅ 성공 코드 (생성자/이니셜라이저 내부)**

**TypeScript**

**this.userData$ = objectVal(ref(this.db, 'qC53z7Q2F8QYZeGJEDQF8qlkz562'));**

**이 코드가 서비스의 생성자나 필드 이니셜라이저에서 실행되었다고 가정하면:**

1. **즉시 실행: objectVal() 함수는 서비스가 처음 생성되는 시점에, 즉 Angular의 활성화된 Injection Context 내에서 즉시 실행됩니다.**
2. **컨텍스트 포착: objectVal() 함수는 이 활성화된 컨텍스트를 사용하여 Firebase API 호출을 Angular의 변경 감지 시스템(Zone.js)과 적절하게 연결합니다.**
3. **오류 없음: API가 허용된 시점에 호출되므로 오류가 발생하지 않고 성공적으로 동작합니다.**

**🛠️ switchMap 내부에서 오류를 해결하는 방법**

**이 문제를 해결하려면, objectVal()이 비동기적으로 호출되더라도 Angular의 Injection Context를 사용할 수 있도록 명시적으로 알려줘야 합니다.**

**최신 Angular 버전(v16 이상)에서는 \*\*Injector\*\*와 runInInjectionContext 함수를 사용하여 해결할 수 있습니다.**

**Injector 주입 및 runInInjectionContext 사용**

**서비스의 생성자에서 Injector를 주입받아 switchMap 내부의 Firebase 호출을 감싸줍니다.**

**TypeScript**

**import { Injectable, inject, Injector, runInInjectionContext } from '@angular/core';**

**import { Database, ref, objectVal } from '@angular/fire/database';**

**import { AuthService } from './auth.service'; // 가정**

**import { switchMap, of, Observable } from 'rxjs';**

**@Injectable({**

**providedIn: 'root'**

**})**

**export class FireDataService {**

**private db: Database = inject(Database);**

**private injector: Injector = inject(Injector); // 💡 Injector를 주입**

**private authService$: AuthService = inject(AuthService); // 가정**

**public userData$: Observable<any>;**

**constructor() {**

**this.userData$ = this.authService$.user$.pipe(**

**switchMap(firebaseUser => {**

**if (firebaseUser) {**

**const userUid = firebaseUser.uid;**

**const itemsRef = ref(this.db, `/${userUid}`);**

**// 🚀 해결책: runInInjectionContext로 objectVal 호출을 감쌉니다.**

**return runInInjectionContext(this.injector, () => {**

**return objectVal(itemsRef);**

**});**

**} else {**

**return of(null);**

**}**

**})**

**);**

**}**

**}**

**이 방법을 사용하면 비동기적인 시점(switchMap 내부)에서도 안전하게 Firebase Realtime Database의 모듈식 함수(objectVal)를 호출할 수 있습니다.**