Get It Done, To-Do-List Uygulaması

Özet

-State yönetimi, local ve genel state yonetımı

-State Management paketlerinin önemi için callbacklerle state yonetımı yapmaya calıstım.

-Provider kütüphanesi ile state management

-Provider, Changenotifier provider, multiprovider sınıfları.

-Shared prefernces

Uygulama Splash Screen ile açılıyor.

Settings kısmından tema değişikliği yapılabiliyor. Bu değişiklik provider paketi ile bütün widget ağacına yayınlanıyor ve state management bu şekilde yapılyor. State management provider paketi ile kullanıcı tuşa basıp yeni görev eklediği zaman, gorevlerin yapılıp yapılmadığını tikleyıp işaretleyebiliyor ve bu değişiklikler provider paketi ile sağlanıyor. Kullanıcı, kaydırma işlemi ile görev silebiliyor ve state yönetimi ile uygulamanın ilgili widget'ları kendini yeniden inşa edebiliyor. Tema ve görev kartı bilgilerini cihazın shared prefernces'ına kaydedebiliyoruz ve uygulama kapatıp açıldığı zaman kullanıcının yapmış olduğu işlemler kaybedilmiyor. Veri tabanı yönetimi en basit hali ıle shared preferences ile sağlanmıştır.

Callback Kullanımı ile State Yönetimi

Provider Paketi

Change Notifier sınıfının notifyListeners() adında değişiklikleri bildirme özelliğini kullandım.

Statless widgetleri provider kullanarak değişmesini sağladım. Aynı zamanda yalnızca istediğimiz widgetlar rebuild oldu. Optimizasyon ve hızlı calısan bı uygulama ıcın onemlı bır durumdur.

Provider: Consumer Widget

Widget ağacının sadece ihtiyaç olunan kısmın rebuild olması için Consumer kullanılır.

Consumer nerede ise o kısım kendini build ediyor. Ayrıca tek tek provider of yazmak yerine Consumer widgetinin build parametresine verılen orta değeri kullanarak kodu kısaltmış oluruz.

Temalandırma ve Ana Sayfa

primarySwatch: ThemeData'nın bir property'idir. verilen rengin tonlarını kullanarak bir renk oluşturur.

Text typhography material.io sitesinden faydalanarak subtitle ve headlines ları kullandım.

Column içerisinde Flex özellikli 2 expanded widget ile ekranı paylaştırarak yapıyı oluşturdum..

Görev Kartları Oluşturma

ItemCard adında *Card* widgetı return eden bir widget oluşturdum. Card içerisinde child olarak *ListTile* widgetı kullandım ve bu widget ile text ve checkbox iconu ekledim.

ListView.builder kullanarak home sayfasında ItemCard() widgetımı eklemiş oldum.

Görevler Veritabanı Oluşturma Provider Kullanımı

State bilgisini tutan ıtemdata sınıfına changenotfier özelliğini kazandırıdım.

Myapp ana widgetımı ChangeNotifierProvider ile sarmaladım ve ItemData verilerini yayınlattım.

Veriyi ekrana çizen listview builder bilgilerini provider ile dinamik bir forma ulaştırdım.

Item sayısını veren ilk expanded kısmı için text verisini ınterpolation ve provider ile dinamik hale getirdim.

Görev Tamamlama Ayarları

Uygulama stateımde, veritabanında bir şey değiştiği zaman ekran otomatik yenilenmeli.

toggleStatus ile durum değişikliğini işleyip checkboxların tıklanma animasyonunu sağladık.

Card widgetına elevation: isDone?1 : 5 ile check box tıklanınca yapışması animasyonunu kattık.

Yeni Görev Ekleme

Floating action buttona tıklandığında state verimi tutan listeme eklenen item ı eklemem gerekir.

BottomSheet class ını kullanarak animasyonlu pencere sağlandı. *Persistent:* kapanması için geri tusuna basmanız gerekır, arka ekrana dokununca kapanmaz.

Modal: pencere arka plana dokununca kapanıyor. Bu uygulamada *Model*'den yararlandık.

Kullanıcının TextField a girdiği input görevini text fieldın controller property'sinden yararlanrak tanımladığımız textController değişkeni içerisinde tuttuk.

Bu değişkeni items data kısmına yeni bir veri olarak provider ile (add butonuna basıldığı zaman gerçekleşmesi gerektiği için Flatbutton içerisindeki onpressed fonksiyonunun içerisinde) gönderdik.

Klavye boyutu farklılığından dolayı container a yalnızca alttan klavyenın kapladığı kadar padding verdim. *(bottom: MediaQuery.of(context).viewInsets.bottom)*

Text Field a satır atlayabilmesi için min ve max lines propert'lerini kullandık

Görev Silme

Cardlara *title* ile *key* vererek *Dissmisible* widget kullandım. Kaydırıp yok edilen card nesnesi veri tabanından ve widget ağacından da yok edilmeli. ListView.builder içinde removeAt kullanan deleteItem fonksyonunu provider ile kullandım. Aynı zamanda Dismissible widgetın onDismissed property'sine de bu fonksiyonu tanımladım.

Cardları GestureDetector widget ile sarmalayıp onTap, doubleTap, onVericalDragDown gibi property'lere deleteItem fonksiyonunu tanımlayabilir ve silinme animasyonunu değiştirebilirdim.

Setting Page ve MultiProvider kullanımı

Ayrı bir settings_page oluşturup body sine

SwitchCard adında Card Widgetı kullanan ayrı oluşturduğum bir statefull widget verdim. Child olarak *SwitchListTile* classını kullandım. Local bir stateManagement yönetebilmek için böyle yaptım.

Home page e ekledıgım settıngs butonunun onchanged property'sine Navigator.push metotunu kullanarak, tıklanıldığında SettingsPage sayfasına gitmesini sağladım.

SwitchListTile card ı ayrı bır yerde statefull wıdget olarak tanımlayarak local state Management yapmıs oldum. Butonun ıcınde calısan her sey yalnızca o butonun setstate ını etkılıyor.

SwitchListTile üstünde iki ayrı Text tanımlayarak subtitle property'sine ternary operator ile ekledim.

Color_theme_data sayfasında yeni bir class oluşturup (with ChangeNotifier) iki adet tema tanımladım. selectedThemeData değişkenime default olarak purple ThemeData atadım.

selectedThemeData yı değiştirebilecek bir fonksiyon tanımladım ve içerisindeki private değişkenimin get metodunu yazdım.

Widget agacının en tepesından asagı MaterialApp ın görebileceği şekilde yayınlamam gerek.

1.ChangeNotifierProvider = listelerle ilgili veriyi alan ItemData.

2 ChangeNotifierProvider = tema rengi ile ilgili veriyi alan ItemData.

Birden fazla veri provide etmek için MultiProvider kullandım.

MaterialApp içerisindeki theme property'sini Provider ile değişip yayınlanmasını sağladım.

SwitchListTile içindeki butonuna, onchanged property'si içerisinde provider ile switchTheme metotunu ekleyerek fonksyonalite kazandırdım.

Final Refactoring

ListView.builderı consumer widget ile sarmalayıp tüm widgetn yanı scaffoldun yerıne sadece consumer kısmının provider ile rebuild olmasını sağladım. Local state yonetımı yaptım.

ListView.builder ı Align ile sarıp shrankwrap ve reverse ile eklenen liste elemanlarının üstten alta doğru eklenmesini sağladım.

İtem Data sınıfımdaki items değişkenimi private yapıp get ile aynı adlı kopyasını oluşturarak ve *UnmodifiableListView* kullanarak dışarıdan erişilemez hale getırdım.

Kodun daha rahat okunabilir olması için, appbar widgetını home_page kısmında metot olarak extract ettım.

Shared Preferences

Kullanıcı uygulamadan çıkıp tekrar gırdıgında lısteleri hazır olarak gelmeliydi.

Verının nasıl bır verı olduguna baglı olarak verıyı tutmak ıcın kullancagınız yontemler degısıklık gosterır. Uygulama içerisinde saklanılacak tema değimi ve login bilgileri gibi verileri tutan özel dosyalar vardır. Bu dosyaya kalıcı olarak kaydedılen verılere shared preferences denır. Krıtık bılgılerın burada saklanılması önerılmez. Bilgiye ulasılma ıslemı işlemciye gore daha yavas olacagından asynchronously yapı gerekır.

Provider ile basitçe bir state yönetimi yaptım.

Not: Parantez içerisindeki argüman gövde içerisinde kullanılmıyor ama yine de parantez içerisinde parametre olarak belirtilmesi gererkiyorsa "_" kullanabiliriz.

Shared Preferences + Provider Paketini Birlikte Kullanma

Provider ile yönetilen bilgimizin cihaz üzerinde kalıcı saklanması gerekiyor. Bu işlem için 2 adet fonksiyona ihtiyaç duyuyorum:

1. Fonksiyon: Cihazadaki SharedPreferences dosyasına git/aç ve yazma işlemi yap.

2.Fonksiyon: Yazılan veriyi oku.

SharedPrefernces objesi tanımladım. Bu objeyı kullanmamızı sağlayan *createSharedPrefObject* adınfa Future döndüren bir asyn fonksiyon oluşturdum. Bu fonksiyon getInstance() statik metotunu barındırır.

Uygulama açıldığında splash screen koymak, veriyi okuyup işleyebilmek için vakit kazandırır.

Not: Uygulama mimarisinde, her iş ve servisin uzmanı sınıflar farklı olmalı ve looselycoupled, yani birbirlerine en zayıf şekilde bağlı olmalılardır. Arayüz ekranlarının olabildiğince bu tip servis ve verilerden ayırılması gerekir.

Anlık değeri kaydedip tutan *saveThemeToSharedPref* adında bir fonksiyon tanımladım. Buton ile her toggle yapıldıgında bu fonksiyon çağrılır.

Uygulama ilk çalıştığında kayıtlı verinin okunması için *loadThemeFromSharedPref* adında veriyi uygulama ilk açıldığında hemen çağırıp getiren bir fonksiyon oluşturdum. Statefull widget olsaydı initState içerisinde çağırabilirdim. Fakat statless widget kullanıldığı için MaterialApp return edilmeden hemen önce Provider ile çağırdım.

Not: Syntactic Sugar Önerisi: "*??*", yazılan değerin null olması halinde devamına yazılan değeri atama işlemi yapar. Yani null koşulunu/kontrolünü gerçekleştirir.

Shared Preferences: main( ) async

Shared Prefernces'dan gidip çekilen veri önem taşıyor ve hiçbir şey oluşmadan önce o bilginin elimizde olmasını istiyorsak, main fonksiyonun içerisinde, runApp çalıştırılmadan önce asyn olarak veri çeken bir fonksiyon tanımlarım. Yani aslında main fonksiyonunu bir asyn fonksiyona çevirmiş olurum. Önce veri gelir, sonra runApp çalışır.

Not: => rotasyonu yalnızca bir satırlık returnler için kullanılan bir kısaltma işlemidir.

Main fonksiyonu içerisinde createSharedPrefObject fonksiyonumu await ile çağırdım. Bu kullanım için fonksiyonumun objesiz de kullanabilir bir yapıda olması gerekir çünkü henüz obje üretilmeden çağırıp kullanmak istiyorum. Objenin üretildiği yere static anahtar kelimesi kullanarak bu işlemi gerçekleştirdim. Bu işlemin gerçekleştirilmesi için öncesinde WidgetsFlutterBinding sınıfının ensureInitialized() metotunu kullanmamız gerekir.

Not: "static" anahtar kelimesi, bir sınıfın sahip olduğu verilerin farklı nesneleri arasındaki değerleri kalıcı hale getirmesine olanak tanır. Elimizdeki sınıfın herhangi bir obje oluşturulmadan kullanılabilecek, hemen ulaşılabilecek bir instances field tanımlayabilmemizi sağlar.

SwitchTileList Hata Düzeltme

Uygulamada settings kısmında ListTile ile Local StateManagement yapıp setState kullandık. Kullanıcının tema değiştirme butonuna tıklaması, yalnızca metni ve butonun görüntüsünü etkilemiyor. Tüm widget ağacını etkiliyor. Settings kısmını izole edemedik. SwitchCard Widgetımızın stateful olmasının bir anlamı kalmadı. Stateless yapıp set state i sildik.

Tema Bilgisini Kaydetme

Uygulamanın tema rengi bilgisini ve yapılacaklar listesinin elemanlarını cihazın shared prefernces'ına kaydetmek ve uygulama yeniden açıldığında bu bilginin tutuluyor olmuş olması. Kullanıcnın uygulamayı başlattığı an bu iki bilgi hazır olmuş olmalı.

createPrefObject, *saveThemeToSharedPref* ve *loadThemeFromSharedPref* adında sırasıyla, obje oluşturup getiren, objenin verisini kaydeden ve uygulama ilk açıldığında SharedPreferencesten veri alıp çekme görevleri olan üç adet fonksiyon oluşturdum.

Kullanıcı her switchTheme ile temayı değiştiğinde bu veriyi ShrPrf'a kaydedebiliriz. Her değişiklikte kayıt edilmesi daha güvenli olur. saveThemeToSharedPref(selected);

Main fonksiyonumuzu asyn bir hale getirip asyn bir fonksıyon cagırdım ve WidgetsFlutterBinding çağırdık.

MaterialApp return edilmeden loadThemeFromSharedPref fonksiyonum ile uygulama baslamadan once tema bılgısını edındırdım.

Uygulama async calısıp obje olusturup bu objenın loadThemeFromSharedPref metotu ile veri alınıp isPurple a atanacak ardından materialApp return edilip tema bilgisindeki selectedThemeData çağrılılcak, o da _selectedThemeData'yı okuyup color_theme_data içerisinde tema kontrolünün yapıldığı yere gidicek.

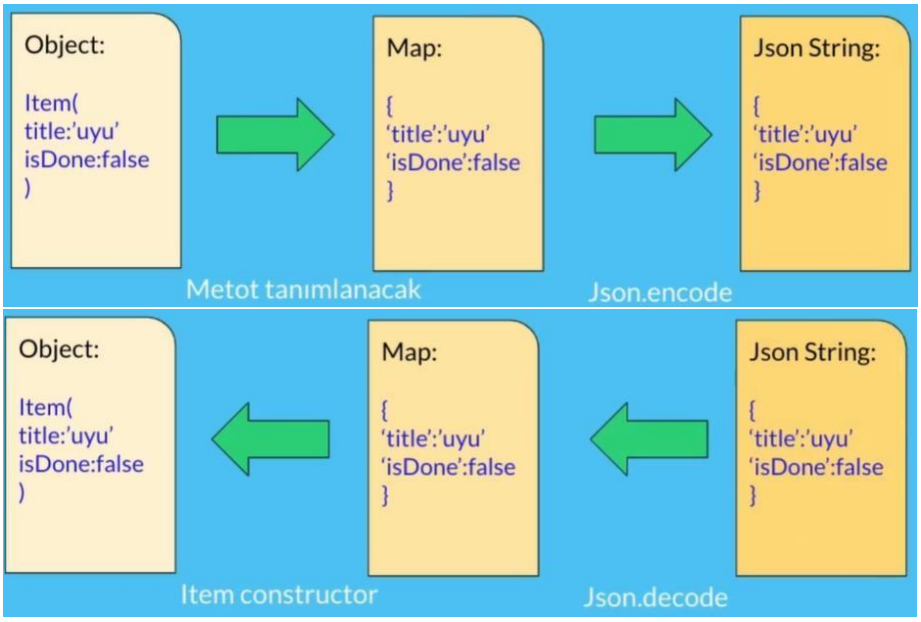
Nesneleri String'e Çevirme

SharedPreferences a Item sınıfından oluşturulan nesnelerin title ve isDone property'si, instances variable'ı oluyor. SharedPrefernces sadece primitive ve stringlerden oluşan listeleri saklama becerisine sahip. Bu özel sınfıtan oluşan özel objeleri SharedPrefernces'a kaydetmek için her Item verimizi string olarak liste içerisinde saklamamız gerekir.

Her objeyi stringe çevrilip String tipinde bir listeye kaydedilip SharedPrefernces'a göndereceğim. Okuyacağım zaman ise string listesini tekrardan objeye dönüştürüp listviewbuilder'ın kullanabileceği Item lardan oluşan listeye dönüştürmem gerekir.

Herhangi bir objeyi stringe nasıl dönüştürürüz? JSON, veri değişimini kolaylaştıran bir metin biçimidir.

Yapacağımız işlem:



Gerekli tanımlamaları object_map_json dosyası altında yaptım.

Item objesının map hali için Map döndren, keyleri string ve value değerleri ise birden çok türde olabileceğinden dynamic tiğinde olan *toMap* adında bir çevirici fonksiyon tanımladım. Bu sınıftan oluşan her obje kendisne ait bu metotu kullanarak kendisinin map e dönüşmüş halini return edebildi.

Map'e dönüşmüş objeyi JSON'a dönüştürmek için json.encode ve json.decode kullandım.

Not: Herhangi bir sınıftan obje oluştururken Named Constructor ile istedğimiz kadar constructor oluşturabiliriz.

Item listesini kullanarak stringlerden oluşan bir liste oluşturup her bir elemanı alıp dönüştürerek başka bir liste içerisine yazdım. Her yeni eleman ekleme, silme ve işaretleme sonrası listeyi hemen oluşturup sharedpreferences'a kaydetme işlemini gerçekleştirdim. Uygulama ilk çalıştığında load metotu ile stringlerden olusan listeyi shared preferencstan alıp her öğeyi çevirip items listesi olusturup listview builder da bu items listesini kullandım.

Not: Dart’da herhangi bir değişkenin ardından . runtimeType metotu kullanılırsa uygulama çalıştığında o objenin veri tipini öğrenebilirsiniz.

Görev Listesini Shared Preferences'a kaydetme

Consumer Widget

Widget agacının aynı kısmında birden fazla provider yayını dinlemek gerekebilir. Consumer Widgeti maksimum 6 yayına kadar, yayınların aynı anda dinlenebilmesini sağlar. Child property’si ile sadece özel olarak build edilecek kısımları belirtebiliriz.

Aynı isimli iki görev oluşurve biri silmek istenirse hata aldım. Sebebi her ikisinin de key değerinin aynı olmasıydı. Dismissible return eden item_card sayfasının key değerini Uniquekey() olarak değiştirince bu sorun flutter tarafından çözülmüş oldu.

Splash Screen

Splash Screen return eden ayrı bir SplashWidget oluşturdum. Splash Screen widgeti ve property’leri ile animasyonu hazırladım. Material app içerisinde bu widgeti return ettim.