State Management 09.09.2024

- Mümkün olan en geniş anlamda, bir uygulamanın durumu(state), uygulama çalışırken bellekte var olan her şeydir. Bu, uygulamanın varlıklarını(app's assets), Flutter çerçevesinin kullanıcı arayüzü hakkında tuttuğu tüm değişkenleri, animasyon durumunu, dokuları, yazı tiplerini vb. içerir.Bu mümkün olan en geniş durum tanımı geçerli olsa da, bir uygulamanın mimarisi için pek kullanışlı değildir.

- İlk olarak, bazı durumları (dokular gibi) yönetmiyorsunuz bile. Framework bunları sizin için halleder. Dolayısıyla, durumun(state) daha kullanışlı bir tanımı "kullanıcı arayüzünüzü herhangi bir zamanda yeniden oluşturmak için ihtiyaç duyduğunuz her türlü veri" şeklindedir.İkinci olarak, kendi kendinize yönettiğiniz durum iki kavramsal türe ayrılabilir: geçici durum(ephemeral state) ve uygulama durumu(app state).

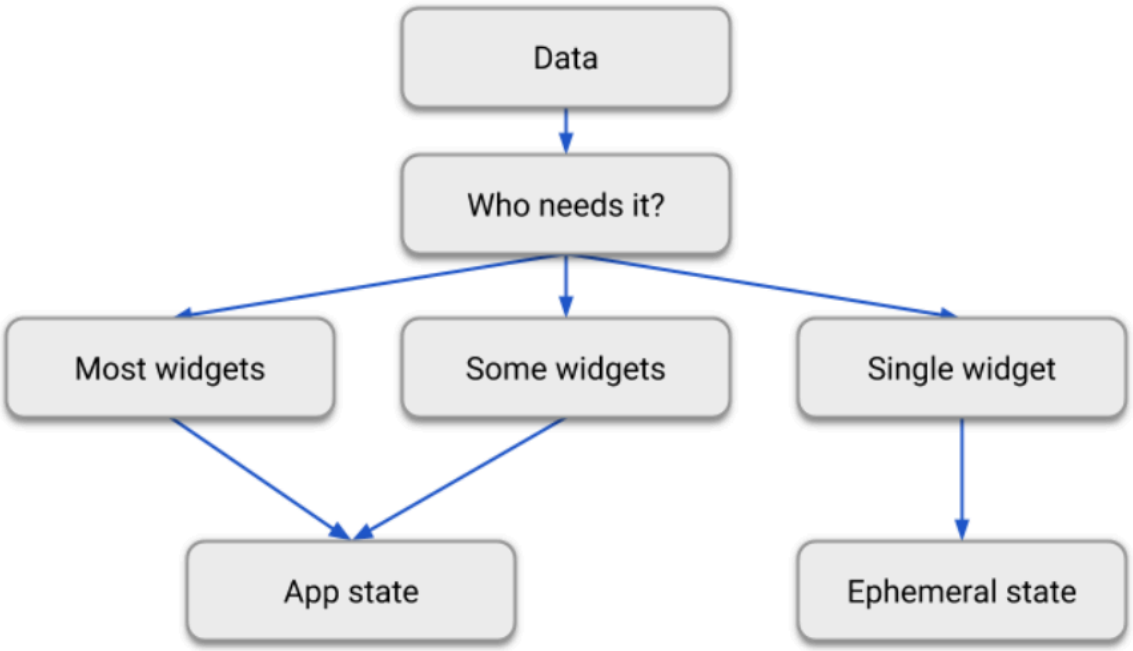
- Geçici durum (bazen UI state veya local state olarak da adlandırılır), tek bir widget'ta düzgün bir şekilde barındırabileceğiniz durumdur. Örnek vermek gerekirse PageView'de geçerli sayfanın durumunu tutan `int _index = 0;` değişkeni veya BottomNavigationBar'da geçerli seçili sekme (onunda ilk tanımlamasının `int _index = 0;` şeklinde olduğunu varsayarsak) uygulama kapatılıp açılsada _index değişkeni her zaman yeniden yaratılacaktır. Bu tip değerler anlık olarak kullanılır ve geçicidir.

- Geçici olmayan, uygulamanızın birçok bölümünde paylaşmak istediğiniz ve kullanıcı oturumları arasında saklamak istediğiniz durum, uygulama durumu(App State) (bazen paylaşılan durum(Shared State) olarak da adlandırılır) olarak adlandırdığımız şeydir. Örnek vermek gerekirse kullanıcı tercihleri, giriş bilgileri, bir sosyal ağ uygulamasındaki bildirimler, bir e-ticaret uygulamasındaki alışveriş sepeti içeriği, haber uygulamasındaki makalelerin okundu/okunmadı durumları diyebiliriz. Uygulama durumunu yönetmek için seçeneklerinizi araştırmak isteyeceksiniz. Seçiminiz, uygulamanızın karmaşıklığına ve doğasına, ekibinizin önceki deneyimine ve diğer birçok hususa bağlıdır.

- Açık olmak gerekirse, uygulamanızdaki tüm durumları yönetmek için State ve setState() işlevlerini kullanabilirsiniz. Aslında, Flutter ekibi bunu birçok basit uygulama örneğinde (her flutter oluşturma ile birlikte aldığınız başlangıç uygulaması dahil) yapıyor. Bunun tersi de geçerlidir. Örneğin, uygulamanızın bağlamında, bottom navigation barda seçili sekmenin geçici bir durum olmadığına karar verebilirsiniz. Bunu sınıfın dışından değiştirmeniz, oturumlar arasında tutmanız vb. gerekebilir. Bu durumda, _index değişkeni uygulama durumudur(app state).

- Belirli bir değişkenin geçici(ephemeral state) mi yoksa uygulama durumu(app state) mu olduğunu ayırt etmek için kesin ve evrensel bir kural yoktur. Bazen birini diğerine dönüştürmeniz gerekebilir. Örneğin, açıkça geçici bir state ile başlarsınız, ancak uygulamanızın özellikleri arttıkça, app state'e taşınması gerekebilir.

For that reason, take the following diagram with a large grain of salt:



- Dan Abramov: “Temel kural şudur: Hangisi daha az garipse onu yapın.”

- Özetle, herhangi bir Flutter uygulamasında iki kavramsal durum türü vardır. Geçici durum, State ve setState() kullanılarak uygulanabilir ve genellikle tek bir widget için yereldir. Gerisi sizin uygulama durumunuzdur. Her iki türün de herhangi bir Flutter uygulamasında yeri vardır ve ikisi arasındaki ayrım kendi tercihinize ve uygulamanın karmaşıklığına bağlıdır.

- Flutter'da state'i, onu kullanan widget'ların üzerinde tutmak mantıklıdır.

- Neden mi? Flutter gibi bildirimsel(declarative) frameworklerde, kullanıcı arayüzünü değiştirmek istiyorsanız, yeniden oluşturmanız gerekir. MyCart.updateWith(somethingNew) yapmanın kolay bir yolu yoktur. Başka bir deyişle, bir widget'ı dışarıdan bir metot çağırarak zorunlu olarak değiştirmek zordur. Ve bunu yapabilseniz bile, framework'ün size yardımcı olmasına izin vermek yerine onunla savaşıyor olursunuz.