

Hybridutveckling med Flutter

15:52:54



Hybridutveckling med Flutter

Kom ihåg att starta inspelning!

Dagens agenda

1. Repetition från senast:

- Layout
- Navigering
- FutureBuilder
- Dialog (popup)
- Modal (liknar en popup)
- Snackbar

Dagens agenda

1. Nytt innehåll (Livekodning):

- Mer om navigering
- Modal (fortsättning)
- Forms
- Mer Expandable
- Scrolling
- Hero animation
- PageTransition
- Theming

Repetition - Layout

1. Container : En mångsidig widget-behållare
 - Dekorativa egenskaper: kant, radie
 - Marginaler och Padding (utfyllnad)
 - Används för visuell och spatial separation
2. Column & Row : Linjära Layouter
 - Riktad Layout: vertikal & horisontell Justering och placering längst axlarna (x,y)
 - **Tips:** använd Expanded-widget för flexibel platsanvändning
3. Stack & Positioned : Överläggningslayouter
 - Stackar widgets ovanpå varandra
 - Positioned för att styra widgetens position inuti Stack

Repetition - Layout - fortsättning

1. **Wrap :** Flödeslayout
 - Adaptiv placering av barn baserad på barnens storlek
 - spacing & runSpacing för kontroll av gemensamma marginaler
2. **ListView :** Skrollbar lista
 - Används för långa listor med barn
 - **Tips:** använd ListView.builder för optimerad rendering
3. **GridView :** 2D rutnätslayout
 - Visar barn i ett rutnätsformat
 - Kontrollera antal barn per axel (x,y)

Repetition - navigering - demoapp

1. BottomNavigationBar : Lateral navigering
 - Används för att navigera mellan det primära innehållet i applikationen
 - Bör användas för att navigera inom samma hierarkiska nivå i applikationen
2. Navigator.push(MaterialPageRoute) : Forward navigation
 - Används för att öppna och navigera till en ny vy på stacken av vyer.
 - Används för att navigera djupare hierarkiskt, eller för att göra framsteg i ett "flöde".
3. Navigator.pop() : Reverse navigation
 - Används för att navigera uppåt i applikationens hierarki.
 - Kan innehärla att stänga en vy från stacken av vyer, eller stänga en popup, beroende på vilken vy som gör anropet.

Alternativa navigeringssval

1. Deep Linking : Öppna vyer i din app från en URL
 - Relevant t.ex. om du vill kunna dela innehåll i din app över andra plattformar.
 - T.ex. för att dela en länk till en vacker pixel-art i en pixel-art redigerings-app
2. Named Routes : Namngivna rutter
 - Koppla en vy till ett namn och använd Navigator.pushNamed(...)
 - **WARNING:** Rekommenderas av Flutter-teamet att undvikas 😊
3. Router : För avancerade navigeringsskrav. Alternativ till Navigator.
 - Främst relevant om du vill att all navigering ska motsvaras av URL:er likt en webbapplikation.
 - T.ex. att `/recipes/[id]/edit` går till en redigeringsida för ett recept.
 - Deras egna paket `go_router` rekommenderas av Flutter-teamet.

Förstå FutureBuilder

- Vad är en FutureBuilder?
 - En widget som returnerar en annan widget baserat på status av en Future (snapshot).
 - Designad för att fungera med asynkrona beräkningar och returnerar snapshots av antingen data eller ett fel.
 - Eliminerar behovet av boilerplate-kod för att hantera olika stader av asynkrona operationer.
- Användningsområden:
 - Hämta data från ett API.
 - Databas CRUD-operationer.
 - Alla uppgifter som kan slutföras senare i framtiden.

FutureBuilder - kodexempel

```
1 FutureBuilder(  
2   future: ingredientsFuture, // berätta vilken future som ska användas  
3   builder: (context, snapshot) { // builder-funktionen körs när status på vår future ändras  
4     if (snapshot.data != null) {  
5       // Visa innehållet i vår lyckade future  
6     } else if (snapshot.hasError) {  
7       // Visa felmeddelande från misslyckad future  
8     } else {  
9       // Visa en indikator att vi väntar på vår future t.ex. en CircularProgressIndicator  
10    }  
11  }  
12);
```

Förstå popups (Dialog) i Flutter

- Vad är en dialog i Flutter?
 - Ett användargränssnittslement som informerar användaren eller frågar användaren om beslut.
 - Kan användas för att visa meddelanden, bekräftelser, val, inmatningar och mer.
 - Genom att använda `showDialog`-metoden kan du visa en dialog ovanpå appens dåvarande innehåll.
- Användningsområden:
 - Informera användaren om en händelse eller uppgift som behöver uppmärksamhet.
 - Fråga användaren om bekräftelse före en åtgärd, t.ex. radera en post.
 - Insamling av användarinput eller beslut.
- När dialogen stängs kan data skickas till den komponent som öppnade dialogen via `Navigator.pop([result])`

Dialog - kodexempel

```
1 IconButton(  
2   icon: Icon(Icons.delete),  
3   onPressed: () {  
4     // visa dialog när någon klickar på något i gränssnittet  
5     showDialog<void>( // void indikerar att dialogen inte returnerar något värde via Navigator.pop()  
6       context: context,  
7       barrierDismissible: false, // användaren måste trycka på en knapp för att stänga dialogen med Navigator.pop()  
8       builder: (BuildContext context) {  
9         // Titel och innehåll i dialogen  
10        // Användaren kan trycka på knapparna för att utföra olika åtgärden  
11        // Dialogen stängs automatiskt när Navigator.pop() anropas  
12        return AlertDialog(  
13          title: const Text('Raderingsdialog'),  
14          content: Text("bekräfta ingrediensradering"),  
15          actions: <Widget>[  
16            TextButton(  
17              child: const Text('avbryt'),  
18              onPressed: () {  
19                Navigator.of(context).pop();  
20              }, // ...  
21            ],  
22          );  
});
```

Förstå Modals i Flutter

- Vad är en modal i Flutter?
 - Ett överlappande användargränssnittselement som visas ovanpå appens primära innehåll.
 - Använts oftast för tillfälliga uppgifter som inte kräver en ny skärm, men som behöver användarens uppmärksamhet.
 - Lämpar sig för något mer omfattande uppgifter än en simpel popup dialog.
- Genom att använda `showModalBottomSheet`-metoden kan du visa en modal nedanför skärmen.
- Användningsområden:
 - Visa ytterligare information eller val utan att navigera bort från den nuvarande skärmen.
 - Få användarens respons på en specifik fråga eller uppgift, t.ex. välja ett datum från en kalender.

Dialog - kodexempel

```
1 // visa dialog när någon klickar på något i gränssnittet
2 final result = await showModalBottomSheet<String>(
3   context: context,
4   builder: (BuildContext context) {
5     return Container(
6       height: MediaQuery.of(context).size.height * 0.5, // 50% av skärmens höjd
7       color: Colors.amber,
8       child: Center(
9         child: TextButton(
10           child: const Text('return random stuff'),
11           onPressed: () {
12             // returnera string "random stuff" som tilldelas till result i komponenten som öppnade modalen
13             Navigator.of(context).pop("random stuff");
14           },
15         );
16       },
17     print(result); // prints "random stuff"
18   ),
19 );
```

Förstå Snackbar i Flutter

- Vad är en `SnackBar` i Flutter?
- Ett temporärt meddelande som visas nedest på skärmen.
- Använts för att informera användaren om en åtgärd som har inträffat, vanligtvis som ett resultat av en användares åtgärd.
- Visas med ` ScaffoldMessenger.of(context).showSnackBar` metod.
- Användningsområden:
 - Ge snabb feedback till användaren.
 - Bekräfta åtgärder som att ett objekt har sparats eller raderats.
 - Visa kortvariga meddelanden som inte stör användaren men ger viktig information.

SnackBar - Kodexample

```
1 // random example button
2 ElevatedButton(
3   child: const Text('Add new'),
4   onPressed: () {
5     // Logic to add new ingredient... such as IngredientRepository.create(....)
6     // ...
7     ScaffoldMessenger.of(context).showSnackBar(
8       // show snackbar with green background and text "Successfully added ingredient"
9       const SnackBar(backgroundColor: Colors.green, content: Text("Successfully added ingredient")),
10      );
11    },
12  );
```

Nu: Livekodning

1. Modal (fortsättning)
2. Forms
3. Mer Expandable
4. Scrolling
5. Hero animation
6. PageTransition
7. Theming

