

Hybridutveckling med Flutter

15:54 : 15



Dagens agenda

Dagens agenda

1. Information om examinerande uppgifter

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden
5. Arkitektur

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden
5. Arkitektur
6. *Serialization*

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden
5. Arkitektur
6. Serialization
7. Lokal datalagring

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden
5. Arkitektur
6. Serialization
7. Lokal datalagring

Information om examinerande uppgifter

Information om examinerande uppgifter

1. Inlämningsdatum senare lagda till måndagar.

Information om examinerande uppgifter

1. Inlämningsdatum senarelagda till måndagar.
 - Fr.o.m. nästa måndag (2/10/2023) är det en inlämning varje måndag i 3 veckor.

Information om **examinerande uppgifter**

1. Inlämningsdatum senarelagda till måndagar.
 - Fr.0.m. nästa måndag (2/10/2023) är det en inlämning varje måndag i 3 veckor.
2. Kursen "Programmering i Dart" examineras med **tre praktiska uppgifter**

Information om examinerande uppgifter

1. Inlämningsdatum senarelagda till måndagar.
 - Fr.o.m. nästa måndag (2/10/2023) är det en inlämning varje måndag i 3 veckor.
2. Kursen "Programmering i Dart" examineras med **tre praktiska uppgifter**
 - Modelkod(library) för **tre olika applikationer (MVC)**

Information om examinerande uppgifter

1. Inlämningsdatum senarelagda till måndagar.
 - Fr.o.m. nästa måndag (2/10/2023) är det en inlämning varje måndag i 3 veckor.
2. Kursen "Programmering i Dart" examineras med **tre praktiska uppgifter**
 - ModelKod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (förslagsvis via länk till GitHub eller som ZIP av projektet)

Information om **examinerande uppgifter**

1. Inlämningsdatum senarelagda till måndagar.
 - Fr.o.m. nästa måndag (2/10/2023) är det en inlämning varje måndag i 3 veckor.
2. Kursen "Programmering i Dart" examineras med **tre praktiska uppgifter**
 - ModelKod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (förslagsvis via länk till GitHub eller som ZIP av projektet)
3. Kursen "Utveckling av Nativeapp med Flutter" examineras med **tre praktiska uppgifter**

Information om examinerande uppgifter

1. Inlämningsdatum senarelagda till måndagar.
 - Fr.o.m. nästa måndag (2/10/2023) är det en inlämning varje måndag i 3 veckor.
2. Kursen "Programmering i Dart" examineras med **tre** praktiska uppgifter
 - ModelKod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (förslagsvis via länk till GitHub eller som ZIP av projektet)
3. Kursen "Utveckling av Nativeapp med Flutter" examineras med **tre** praktiska uppgifter
 - Slutförande av de påbörjade applikationerna (**MVC**)

Information om examinerande uppgifter

1. Inlämningsdatum senarelagda till måndagar.
 - Fr.o.m. nästa måndag (2/10/2023) är det en inlämning varje måndag i 3 veckor.
2. Kursen "Programmering i Dart" examineras med **tre** praktiska uppgifter
 - ModelKod(library) för tre olika applikationer (**MVC**)
 - Kod lämnas in (förslagsvis via länk till GitHub eller som ZIP av projektet)
3. Kursen "Utveckling av Nativeapp med Flutter" examineras med **tre** praktiska uppgifter
 - Slutförande av de påbörjade applikationerna (**MVC**)
 - Kod lämnas in + kort video **demo**

Dagens agenda

1. Information om examinerande uppgifter
2. **Repetition**
3. Handledning
4. Antaganden
5. Arkitektur
6. Serialization
7. Lokal datalagring

Repetition - pedagogisk ansats

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att hålla er **intresserade** och på rätt spår

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. **Viktigt att kunna lära genom dokumentation och praktisk erfarenhet**

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. **Viktigt** att kunna lära genom dokumentation och praktisk erfarenhet
5. *"The only way to learn a new programming language is by writing programs in it."* - Dennis Ritchie (skapare av C)

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. **Viktigt** att kunna lära genom **dokumentation** och **praktisk erfarenhet**
5. *"The only way to learn a new programming language is by writing programs in it."* - Dennis Ritchie (skapare av C)
6. **Fullt möjligt** att viss **information** kan vara överflödig eller saknas

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. **Viktigt** att kunna lära genom **dokumentation** och **praktisk erfarenhet**
5. *"The only way to learn a new programming language is by writing programs in it."* - Dennis Ritchie (skapare av C)
6. Fullt möjligt att viss information kan vara överflödig eller saknas
7. Målet är att bli en bättre utvecklare generellt och apputvecklare specifikt

Repetition - pedagogisk ansats

1. Introducera koncept i en lämplig ordning för uppgifterna
 - Fullt möjligt att det finns en bättre ordning
2. Målet är att **hålla er intresserade** och på rätt spår
3. Uppgifter som en stark motivationsfaktor
4. **Viktigt att kunna lära genom dokumentation och praktisk erfarenhet**
5. *"The only way to learn a new programming language is by writing programs in it."* - Dennis Ritchie (skapare av C)
6. Fullt möjligt att viss information kan vara överflödig eller saknas
7. Målet är att bli en bättre utvecklare generellt och apputvecklare specifikt
 - Därför dyker det upp en del designmönster som kan anses **friivilliga men bidrar till högre kvalité (VG kriterier)** på uppgifterna

Repetition - innehåll tidigare föreläsning

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk
4. **Introduktion uppgift 1**

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk
4. Introduktion uppgift 1
 - Rollspel för fullt möjligt scenario

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk
4. Introduktion uppgift 1
 - Rollspel för fullt möjligt scenario
 - Uppmaning att börja designa systemet och besöka dokumentation

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk
4. Introduktion uppgift 1
 - Rollspel för fullt möjligt scenario
 - Uppmaning att börja designa systemet och besöka dokumentation
5. Kodexempel

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk
4. Introduktion uppgift 1
 - Rollspel för fullt möjligt scenario
 - Uppmaning att börja designa systemet och besöka dokumentation
5. Kodexempel
 - **Funktionsdeklarationer, alternativ för funktionsparametrar / argument**

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk
4. Introduktion uppgift 1
 - Rollspel för fullt möjligt scenario
 - Uppmaning att börja designa systemet och besöka dokumentation
5. Kodexempel
 - Funktionsdeklarationer, alternativ för funktionsparametrar / argument
 - Objektorienterade koncept, abstrakta klasser/interfaces

Repetition - innehåll tidigare föreläsning

1. Förhållande mellan Dart & Flutter
2. Varför Dart skapats och används i Flutter
3. Likheter mellan Dart & andra språk
4. Introduktion uppgift 1
 - Rollspel för fullt möjligt scenario
 - Uppmaning att börja designa systemet och besöka dokumentation
5. Kodexempel
 - Funktionsdeklarationer, alternativ för funktionsparametrar / argument
 - Objektorienterade koncept, abstrakta klasser/interfaces
 - Några datatyper, t.ex. List, Map

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. **Handledning**
4. Antaganden
5. Arkitektur
6. Serialization
7. Lokal datalagring

Handling

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna
 - Går också utmärkt att skriva på Teams. Jag är oftast tillgänglig och blir inte störd. Däremot ingen garanti att få svar direkt.

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna
 - Går också utmärkt att skriva på Teams. Jag är oftast tillgänglig och blir inte störd. Däremot ingen garanti att få svar direkt.
2. Senaste tillfället mycket diskussion och demonstration av koncept. Minimal individuell handledning. Därför också inspelat.

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna
 - Går också utmärkt att skriva på Teams. Jag är oftast tillgänglig och blir inte störd. Däremot ingen garanti att få svar direkt.
2. Senaste tillfället mycket diskussion och demonstration av koncept. Minimal individuell handledning. Därför också inspelat.
 - **Systemarkitektur - genomgång i dag**

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna
 - Går också utmärkt att skriva på Teams. Jag är oftast tillgänglig och blir inte störd. Däremot ingen garanti att få svar direkt.
2. Senaste tillfället mycket diskussion och demonstration av koncept. Minimal individuell handledning. Därför också inspelat.
 - Systemarkitektur - **genomgång idag**
 - **Diskussion designmönster - användning / behov**

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna
 - Går också utmärkt att skriva på Teams. Jag är oftast tillgänglig och blir inte störd. Däremot ingen garanti att få svar direkt.
2. Senaste tillfället mycket diskussion och demonstration av koncept. Minimal individuell handledning. Därför också inspelat.
 - Systemarkitektur - **genomgång idag**
 - Diskussion designmönster - användning / behov
 - Exempelkod liknande uppgift - **genomgång idag**

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna
 - Går också utmärkt att skriva på Teams. Jag är oftast tillgänglig och blir inte störd. Däremot ingen garanti att få svar direkt.
2. Senaste tillfället mycket diskussion och demonstration av koncept. Minimal individuell handledning. Därför också inspelat.
 - Systemarkitektur - **genomgång i dag**
 - Diskussion designmönster - användning / behov
 - Exempelkod liknande uppgift - **genomgång i dag**
 - **Skriva tester**

Handledning

1. Tillfälle för diskussion och praktisk hjälp med uppgifterna
 - Går också utmärkt att skriva på Teams. Jag är oftast tillgänglig och blir inte störd. Däremot ingen garanti att få svar direkt.
2. Senaste tillfället mycket diskussion och demonstration av koncept. Minimal individuell handledning. Därför också inspelat.
 - Systemarkitektur - **genomgång idag**
 - Diskussion designmönster - användning / behov
 - Exempelkod liknande uppgift - **genomgång idag**
 - Skriva tester
 - **Köra debuggern**

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. **Antaganden**
5. Arkitektur
6. Serialization
7. Lokal datalagring

Antaganden

Antaganden

1. Ni har skapat någon typ av databärande klass / interface

Antaganden

- Ni har skapat någon typ av databärande klass / interface

```
1 class Exercise {  
2   String id, name, description; // all 3 variables are now strings  
3   double? defaultWeight;  
4   int? defaultRepetitions;  
5   Exercise({  
6     required this.id, required this.name, required this.description,  
7     this.defaultWeight, this.defaultRepetitions,  
8   });  
9 }
```

Antaganden

1. Ni har skapat någon typ av databärande klass / interface

```
1 class Exercise {  
2   String id, name, description; // all 3 variables are now strings  
3   double? defaultWeight;  
4   int? defaultRepetitions;  
5   Exercise({  
6     required this.id, required this.name, required this.description,  
7     this.defaultWeight, this.defaultRepetitions,  
8   });  
9 }
```

2. Ni har beskrivit någon/några funktioner som hjälper till att implementera logiken för uppgiften.

Antaganden

- Ni har skapat någon typ av databärande klass / interface

```
1 class Exercise {  
2   String id, name, description; // all 3 variables are now strings  
3   double? defaultWeight;  
4   int? defaultRepetitions;  
5   Exercise({  
6     required this.id, required this.name, required this.description,  
7     this.defaultWeight, this.defaultRepetitions,  
8   });  
9 }
```

- Ni har beskrivit någon/några funktioner som hjälper till att implementera logiken för uppgiften.

```
1 class Workout {  
2   addExercise(Exercise exercise) {  
3     // TODO: Add exercise to workout  
4   }  
5 }
```

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden
5. **Arkitektur**
6. Serialization
7. Lokal datalagring

Architektur - Layered architecture

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - **Presenterar state** från **Lagret under**

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - **Skickar events till lagret under**

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - o Skapas i **flutter**.
 - o Presenterar state från lagret under
 - o Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.
 - Uppdaterar state.

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.
 - Uppdaterar state.
 - Kommunicerar med olika datakällor i lagret under

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.
 - Uppdaterar state.
 - Kommunicerar med olika datakällor i lagret under
 - I denna kurs implementerad med designmönstret **Bloc**

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.
 - Uppdaterar state.
 - Kommunicerar med olika datakällor i lagret under
 - I denna kurs implementerad med designmönstret **Bloc**
3. Persistence layer

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.
 - Uppdaterar state.
 - Kommunicerar med olika datakällor i lagret under
 - I denna kurs implementerad med designmönstret **Bloc**
3. Persistence layer
 - **CRUD** av lagrat data genom gemensamt designmönster: **Repository**

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.
 - Uppdaterar state.
 - Kommunicerar med olika datakällor i lagret under
 - I denna kurs implementerad med designmönstret **Bloc**
3. Persistence layer
 - CRUD av lagrat data genom gemensamt designmönster: **Repository**
4. Database layer

Arkitektur - Layered architecture

1. Presentation layer - Vyn användaren ser.
 - Skapas i **flutter**.
 - Presenterar state från lagret under
 - Skickar events till lagret under
2. Business layer - Implementerar applikationens logik.
 - Uppdaterar state.
 - Kommunicerar med olika datakällor i lagret under
 - I denna kurs implementerad med designmönstret **Bloc**
3. Persistence layer
 - CRUD av lagrat data genom gemensamt designmönster: **Repository**
4. Database layer
 - Lagringsmekanismen. T.ex. **SQL**, **MongoDB**, **CMS**(HTTP API?), **file**, ...

Arkitektur - Visuellt

Figur på draw.io...

Dagens agenda

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden
5. Arkitektur
6. **Serialization**
7. Lokal datalagring

Serialization

Serialization

1. **Serialization** - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.

Serialization

1. **Serialization** - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.
2. **Motsatsen kallas Deserialization** - när vi återskapar ursprungsobjekten/datastrukturen från det serialiseringade formatet.

Serialization

1. Serialization - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.
2. Motsatsen kallas Deserialization - när vi återskapar ursprungsobjekten/datastrukturerna från det serialiseringade formatet.
3. I kurser kommer vi använda:

Serialization

1. **Serialization** - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.
2. Motsatsen kallas **Deserialization** - när vi återskapar ursprungsobjekten/datastrukturerna från det serialiseringade formatet.
3. I kursen kommer vi använda:
 - **Till och från JSON**

Serialization

1. **Serialization** - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.
2. Motsatsen kallas **Deserialization** - när vi återskapar ursprungsobjekten/datastrukturen från det serialiseringade formatet.
3. I kursen kommer vi använda:
 - o Till och från JSON
 - o Till och från String (genom att först konvertera till JSON)

Serialization

1. **Serialization** - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.
2. Motsatsen kallas **Deserialization** - när vi återskapar ursprungsobjekten/datastrukturerna från det serialiseringade formatet.
3. I kursen kommer vi använda:
 - o Till och från JSON
 - o Till och från String (genom att först konvertera till JSON)
4. **JSON** - **JavaScript Object Notation**

Serialization

1. **Serialization** - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.
2. Motsatsen kallas **Deserialization** - när vi återskapar ursprungsobjekten/datastrukturerna från det serialiseringade formatet.
3. I kursen kommer vi använda:
 - o Till och från JSON
 - o Till och från String (genom att först konvertera till JSON)
4. **JSON** - JavaScript Object Notation

```
1 {  
2   "name": "John",  
3   "description": "Smith",  
4   "defaultWeight": null,  
5   "defaultRepetitions": 15  
6 }
```

Serialization

1. **Serialization** - Att konvertera datastrukturer eller objekt till ett format som kan lagras eller överföras.
2. Motsatsen kallas **Deserialization** - när vi återskapar ursprungsobjekten/datastrukturen från det serialiseringade formatet.
3. I kursen kommer vi använda:
 - Till och från JSON
 - Till och från String (genom att först konvertera till JSON)
4. **JSON** - JavaScript Object Notation

```
1 {  
2   "name": "John",  
3   "description": "Smith",  
4   "defaultWeight": null,  
5   "defaultRepetitions": 15  
6 }
```

- I Dart används datatyper **Map<String, dynamic>**

Serialization - Example

```
1 class Example {  
2   final String name;  
3  
4   Example({required this.name});  
5 }
```

Serialization - Example

```
1 class Example {  
2   final String name;  
3  
4   Example({required this.name});  
5  
6   Map<String, dynamic> toJson() => {  
7     'name': name,  
8   };  
9 }
```

Serialization - Example

```
1 class Example {  
2   final String name;  
3  
4   Example({required this.name});  
5  
6   Map<String, dynamic> toJson() => {  
7     'name': name,  
8   };  
9  
10  factory Example.fromJson(Map<String, dynamic> json) {  
11    return Example(name: json['name']);  
12  }  
13 }
```

Serialization - Example

```
1 class Example {  
2   final String name;  
3  
4   Example({required this.name});  
5  
6   Map<String, dynamic> toJson() => {  
7     'name': name,  
8   };  
9  
10  factory Example.fromJson(Map<String, dynamic> json) {  
11    return Example(name: json['name']);  
12  }  
13  
14  // first convert to json. then use built in json encoding to string  
15  String serialize() {  
16    final json = toJson();  
17    final string = jsonEncode(json);  
18    return string;  
19  }  
20}
```

Serialization - Example

```
1 class Example {  
2   final String name;  
3  
4   Example({required this.name});  
5  
6   Map<String, dynamic> toJson() => {  
7     'name': name,  
8   };  
9  
10  factory Example.fromJson(Map<String, dynamic> json) {  
11    return Example(name: json['name']);  
12  }  
13  
14  // first convert to json, then use built in json encoding to string  
15  String serialize() {  
16    final json = toJson();  
17    final string = jsonEncode(json);  
18    return string;  
19  }  
20  
21  // first convert string to json, then create class from json  
22  factory Example.deserialize(String serialized) {  
23    return Example.fromJson(jsonDecode(serialized));  
24  }
```

Serialization - forstsättning

Serialization - fortsättning

- För att serialisera ett objekt måste alla fält/attribut/instancesvariabler i objektet också vara **serializable**

Serialization - fortsättning

- För att serialisera ett objekt måste alla fält/attribut/instansvariabler i objektet också vara serializerbara

```
1 class ExampleHolder {  
2   List<Example> examples;  
3   ExampleHolder({  
4     this.examples = const [],  
5   });  
6 }
```

Serialization - fortsättning

- För att serialisera ett objekt måste alla fält/attribut/instansvariabler i objektet också vara serializerbara

```
1 class ExampleHolder {  
2   List<Example> examples;  
3  
4   ExampleHolder({  
5     this.examples = const [],  
6   });  
7  
8   Map<String, dynamic> toJson() => {  
9     'examples': examples.map((example) => example.toJson()) .toList(),  
10   };  
11 }
```

Serialization - fortsättning

- För att serialisera ett objekt måste alla fält/attribut/instansvariabler i objektet också vara serializerbara

```
1 class ExampleHolder {  
2   List<Example> examples;  
3  
4   ExampleHolder({  
5     this.examples = const [],  
6   });  
7  
8   Map<String, dynamic> toJson() => {  
9     'examples': examples.map((example) => example.toJson()) .toList(),  
10    };  
11  
12   factory ExampleHolder.fromJson(Map<String, dynamic> json) {  
13     return ExampleHolder(  
14       examples: (json['examples'] as List)  
15         .map((json) => Example.fromJson(json))  
16         .toList(),  
17     );  
18   }  
19 }
```

Tack för idag!

1. Information om examinerande uppgifter
2. Repetition
3. Handledning
4. Antaganden
5. Arkitektur
6. Serialization
7. **Lokal datalagring**

Lokal datalagring

Lokal datalagring

1. Vi vill kunna spara data lokalt i appen, som finns kvar om vi stänger appen och kommer tillbaka senare.

Lokal datalagring

1. Vi vill kunna spara data lokalt i appen, som finns kvar om vi stänger appen och kommer tillbaka senare.
2. Vårat **Repository** kommunicerar med den **valda mekanismen för lokal datalagring**.

Lokal datalagring

1. Vi vill kunna spara data lokalt i appen, som finns kvar om vi stänger appen och kommer tillbaka senare.
2. Vår **Repository** kommunicerar med den valda mekanismen för lokal datalagring.
 - Innehåller CRUD-operationer. `create`, `read`, `update`, `delete` (och `readAll`)

Lokal datalagring

1. Vi vill kunna spara data lokalt i appen, som finns kvar om vi stänger appen och kommer tillbaka senare.
2. Vårat **Repository** kommunicerar med den valda mekanismen för lokal datalagring.
 - Innehåller CRUD-operationer. `create`, `read`, `update`, `delete` (och `readAll`)
3. Vi kommer använda oss av **HiveDB** som ni hittar på:
 - <https://pub.dev/packages/hive>

ExampleRepository

```
1 class ExampleRepository {
2     /* Från Hive får vi en Box som lagrar värden av typen String.
3     Allt som lagras i en box har en nyckel som är en String.
4     Vår Box är praktiskt taget en Map<String, String> */
5     late Box<String> _exampleBox;
6
7     ExampleRepository() {
8         // nit (petig) : användaren kanske själv vill konfigurera var datat ska sparas
9         Directory directory = Directory.current;
10        Hive.init(directory.path);
11    }
12
13    Future<void> initialize() async {
14        // asynkron operation, tar obestämd (men snabb) tid att öppna vår datalagring.
15        _exampleBox = await Hive.openBox('examples');
16    }
17
18 }
```

ExampleRepository

```
1 class ExampleRepository {
2   late Box<String> _exampleBox;
3
4   ExampleRepository() {
5     Directory directory = Directory.current;
6     Hive.init(directory.path);
7   }
8
9   Future<void> initialize() async {
10   _exampleBox = await Hive.openBox('examples');
11 }
12
13 }
```

ExampleRepository

```
1 class ExampleRepository {
2   late Box<String> _exampleBox;
3
4   ExampleRepository() {
5     Directory directory = Directory.current;
6     Hive.init(directory.path);
7   }
8
9   Future<void> initialize() async {
10   _exampleBox = await Hive.openBox('examples');
11 }
12
13 bool create(Example example) // ...
14
15 Example? read(String id) // ...
16
17 Example update(Example example) // ...
18
19 bool delete(String id) // ...
20
21 List<Example> readAll() // ...
22
23 }
```

ExampleRepository - Create

```
1 class ExampleRepository {  
2  
3     bool create(Example example) {  
4         var existing = _exampleBox.get(example.id);  
5         if (existing != null) {  
6             return false;  
7         }  
8         _exampleBox.put(example.id, example.serialize());  
9         return true;  
10    }  
11  
12 }
```

ExampleRepository - Create

```
1 class ExampleRepository {  
2  
3     bool create(Example example) {  
4         var existing = _exampleBox.get(example.id);  
5         if (existing != null) {  
6             return false;  
7         }  
8         _exampleBox.put(example.id, example.serialize());  
9         return true;  
10    }  
11  
12 }
```

ExampleRepository - Create

```
1 class ExampleRepository {  
2  
3     bool create(Example example) {  
4         var existing = _exampleBox.get(example.id);  
5         if (existing != null) {  
6             return false;  
7         }  
8         _exampleBox.put(example.id, example.serialize());  
9         return true;  
10    }  
11  
12 }
```

ExampleRepository - Create

```
1 class ExampleRepository {  
2  
3     bool create(Example example) {  
4         var existing = _exampleBox.get(example.id);  
5         if (existing != null) {  
6             return false;  
7         }  
8         _exampleBox.put(example.id, example.serialize());  
9         return true;  
10    }  
11  
12 }
```

ExampleRepository - Create

```
1 class ExampleRepository {
2
3   bool create(Example example) {
4     var existing = _exampleBox.get(example.id);
5     if (existing != null) {
6       return false;
7     }
8     _exampleBox.put(example.id, example.serialize());
9     return true;
10   }
11
12 }
```

ExampleRepository - Read

```
1 class ExampleRepository {  
2  
3     Example? read(String id) {  
4         var serialized = _exampleBox.get(id);  
5         return serialized != null ? Example.deserialize(serialized) : null;  
6     }  
7  
8 }
```

ExampleRepository - Read

```
1 class ExampleRepository {  
2  
3     Example? read(String id) {  
4         var serialized = _exampleBox.get(id);  
5         return serialized != null ? Example.deserialize(serialized) : null;  
6     }  
7  
8 }
```

ExampleRepository - Read

```
1 class ExampleRepository {  
2  
3     Example? read(String id) {  
4         var serialized = _exampleBox.get(id);  
5         return serialized != null ? Example.deserialize(serialized) : null;  
6     }  
7  
8 }
```

ExampleRepository - Read

```
1 class ExampleRepository {  
2  
3     Example? read(String id) {  
4         var serialized = _exampleBox.get(id);  
5         return serialized != null ? Example.deserialize(serialized) : null;  
6     }  
7  
8 }
```

ExampleRepository - Update

```
1 class ExampleRepository {  
2  
3     Example update(Example example) {  
4         var existing = _exampleBox.get(example.id);  
5         if (existing == null) {  
6             throw Exception('Example not found');  
7         }  
8         _exampleBox.put(example.id, example.serialize());  
9         return example;  
10    }  
11  
12 }
```

ExampleRepository - Update

```
1 class ExampleRepository {
2
3   Example update(Example example) {
4     var existing = _exampleBox.get(example.id);
5     if (existing == null) {
6       throw Exception('Example not found');
7     }
8     _exampleBox.put(example.id, example.serialize());
9     return example;
10   }
11
12 }
```

ExampleRepository - Update

```
1 class ExampleRepository {
2
3   Example update(Example example) {
4     var existing = _exampleBox.get(example.id);
5     if (existing == null) {
6       throw Exception('Example not found');
7     }
8     _exampleBox.put(example.id, example.serialize());
9     return example;
10   }
11
12 }
```

ExampleRepository - Update

```
1 class ExampleRepository {
2
3   Example update(Example example) {
4     var existing = _exampleBox.get(example.id);
5     if (existing == null) {
6       throw Exception('Example not found');
7     }
8     _exampleBox.put(example.id, example.serialize());
9     return example;
10   }
11
12 }
```

ExampleRepository - Update

```
1 class ExampleRepository {
2
3   Example update(Example example) {
4     var existing = _exampleBox.get(example.id);
5     if (existing == null) {
6       throw Exception('Example not found');
7     }
8     _exampleBox.put(example.id, example.serialize());
9     return example;
10   }
11
12 }
```

ExampleRepository - Delete

```
1 class ExampleRepository {  
2  
3     bool delete(String id) {  
4         var existing = _exampleBox.get(id);  
5         if (existing != null) {  
6             _exampleBox.delete(id);  
7             return true;  
8         }  
9         return false;  
10    }  
11  
12 }
```

ExampleRepository - Delete

```
1 class ExampleRepository {
2
3     bool delete(String id) {
4         var existing = _exampleBox.get(id);
5         if (existing != null) {
6             _exampleBox.delete(id);
7             return true;
8         }
9         return false;
10    }
11
12 }
```

ExampleRepository - Delete

```
1 class ExampleRepository {  
2  
3     bool delete(String id) {  
4         var existing = _exampleBox.get(id);  
5         if (existing != null) {  
6             _exampleBox.delete(id);  
7             return true;  
8         }  
9         return false;  
10    }  
11  
12 }
```

ExampleRepository - Delete

```
1 class ExampleRepository {  
2  
3     bool delete(String id) {  
4         var existing = _exampleBox.get(id);  
5         if (existing != null) {  
6             _exampleBox.delete(id);  
7             return true;  
8         }  
9         return false;  
10    }  
11  
12 }
```

ExampleRepository - Delete

```
1 class ExampleRepository {
2
3     bool delete(String id) {
4         var existing = _exampleBox.get(id);
5         if (existing != null) {
6             _exampleBox.delete(id);
7             return true;
8         }
9         return false;
10    }
11
12 }
```

ExampleRepository - readAll

```
1 class ExampleRepository {  
2  
3     List<Example> readAll() {  
4         return _exampleBox.values  
5             .map((serialized) => Example.deserialize(serialized)) // for each serialized object, perform deserialization  
6             .toList(); // and we are returning a list :-)  
7     }  
8  
9 }
```

ExampleRepository - readAll

```
1 class ExampleRepository {  
2  
3     List<Example> readAll() {  
4         return _exampleBox.values  
5             .map((serialized) => Example.deserialize(serialized)) // for each serialized object, perform deserialization  
6             .toList(); // and we are returning a list :-)  
7     }  
8  
9 }
```

ExampleRepository - readAll

```
1 class ExampleRepository {  
2  
3     List<Example> readAll() {  
4         return _exampleBox.values  
5             .map((serialized) => Example.deserialize(serialized)) // for each serialized object, perform deserialization  
6             .toList(); // and we are returning a list :-)  
7     }  
8  
9 }
```

ExampleRepository - readAll

```
1 class ExampleRepository {  
2  
3     List<Example> readAll() {  
4         return _exampleBox.values  
5             .map((serialized) => Example.deserialize(serialized)) // for each serialized object, perform deserialization  
6             .toList(); // and we are returning a list :-)  
7     }  
8  
9 }
```

ExampleRepository - readAll

```
1 class ExampleRepository {  
2  
3     List<Example> readAll() {  
4         return _exampleBox.values  
5             .map((serialized) => Example.deserialize(serialized)) // for each serialized object, perform deserialization  
6             .toList(); // and we are returning a list :-)  
7     }  
8  
9 }
```

Tack för idag!

Det var allt!

