

Denny's Blog

[Home](#) [Archives](#) [Tags](#) [About](#)

Dart - Mixin

2019-04-29 · dart

前言

mixin可以簡單的理解為“已經實作的interface”。以下範例:

Code

```
1  mixin CryAbility{
2      void cry() => print("cry very loud");
3  }
4
5  class Baby with CryAbility{
6      int age = 0;
7  }
8
9  // 使用時
10 var baby = Baby();
11 baby.cry();
```

是否還蠻好懂得呢?接下來更細緻一點介紹mixin的使用方法。

舊版宣告

mixin在舊版的dart中是直接宣告為class，然後只要在想加上mixin的class中使用with關鍵字，就會自動被解析為mixin。

Code

```
1  // abstract 拿掉亦可執行
2  abstract class CryAbility{
3      void cry() => print("cry very loud");
4  }
5
6  class Baby with CryAbility{
7      int age = 0;
8  }
9
10 // 使用時
11 var baby = Baby();
12 baby.cry();
```

不過這樣造成語意不清，身為mixin的class應該只能被with，不應該被實體化。

於是多數人會在mixin class前面加上abstract，以提醒使用者。

為了可讀性，還是盡可能使用新版的mixin關鍵字比較好。

多重實作

如同一個class可以實作多個interface，一個class同樣也可以實作多個mixin。以下範例。

Code

```
1  mixin CryAbility{
2      void cry() => print("cry very loud");
3  }
```



```

4
5  mixin EatAbility{
6      void eat() => print("eat eat eat");
7  }
8
9  class Baby with CryAbility, EatAbility{
10     int age = 0;
11 }
12
13 // 使用時
14 var baby = Baby();
15 baby.cry();
16 baby.eat();

```

限制實作對象

如果今天想限制只有某些class可以實作mixin，比如說，假如想限制只有Baby可以cry，其他class(例如IronMan)不能cry。那可以用以下的寫法：

Code

```

1  // on 可以實作的對象
2  mixin CryAbility on Baby{
3      void cry() => print("cry very loud");
4      // 可以使用on對象上的物件
5      void cryWithAge() => print("A $age year-old baby cry very loud");
6  }
7
8  // ERROR: 注意!!!此處不能with，原因後述。
9  class Baby {
10     int age = 0;
11 }
12
13 // 沒問題，GrowthBaby是Baby的一種
14 class GrowthBaby extends Baby with CryAbility{
15     int age = 1;
16 }
17
18 // ERROR: IronMan 不屬於Baby，編譯器會報錯。
19 class IronMan with CryAbility{
20     int iron = 10;
21 }
22
23 // 使用時
24 var baby = GrowthBaby();
25 baby.cryWithAge();

```

當限制CryAbility只能on Baby時，CryAbility就有了存取Baby field的能力。因此cryWithAge才能使用age這個Baby的field。

值得一提的是，IronMan報錯不難理解，但為何無法在原始的Baby class上with CryAbility呢？這牽扯到了mixin這個關鍵字如何被轉譯成舊版的宣告形式的。以下範例

Code

```

1  mixin CryAbility on Baby{
2      void cry() => print("cry very loud");
3  }
4
5  // 等價於
6  abstract class CryAbility extends Baby{
7      void cry() => print("cry very loud");
8  }

```

所以為何CryAbility能使用Baby field? 因為他繼承了Baby。
為何Baby不能with CryAbility? 因為他是CryAbility的super class。
這點在使用上要特別小心。

實作同一個方法

如果實作多個mixin，剛好有同樣的method name，那到底誰會被呼叫呢？
簡單結論是：最後一個被with的會被呼叫。

Code

```
1  mixin CryAbility1{
2    void cry() => print("cry1");
3  }
4
5
6  mixin CryAbility2{
7    void cry() => print("cry2");
8  }
9
10 class Baby12 with CryAbility1, CryAbility2{}
11
12 class Baby21 with CryAbility2, CryAbility1{}
13
14 main(){
15   Baby12().cry(); // 輸出: cry2
16   Baby21().cry(); // 輸出: cry1
17 }
```

更詳細內容請看參考資料: [【译】Dart | 什么是Mixin](#)

Reference

- 官網教學
- Implementation plan for super mixins
- [【译】Dart | 什么是Mixin](#)

- 前言
- 舊版宣告
- 多重實作
- 限制實作
- 實作同一個
- Reference

#dart

◀ [\[讀書心得\] Java 8 Lambdas 技術手冊 \(Java 8 Lambdas: Pragmatic Functional Programming\)](#)

[Dart Class Constructor](#) ▶



0 Comments Denny's blog

 Login ▾ Recommend Tweet Share

Sort by Best ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Be the first to comment.

ALSO ON DENNY'S BLOG

[讀書心得] 圖解密碼學與比特幣原理 - Denny's Blog

1 comment • 2 years ago



李元魁 — 對! 這本很好讀

.net的RSA加解密與UWP - Denny's Blog

2 comments • 2 years ago




鄭翔元 (肥冬瓜) — 可以 請便

Swift 不同Controller互動 與 Navigation Bar - Denny's Blog

1 comment • 2 years ago



Richard Hsieh — 我覺得寫的蠻清楚的~~^^

 Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy Policy

%(EXTRA string=Hugo) | - Even

© 2015 - 2019 ♥ Denny Cheng

