Dart中的Mixins

Nicholas68 LV.3
2021年11月29日 15:55 · 阅读 548

关注

上篇文章, 我们学习了Dart基础语法中的单继承关键字abstract、extends关键字

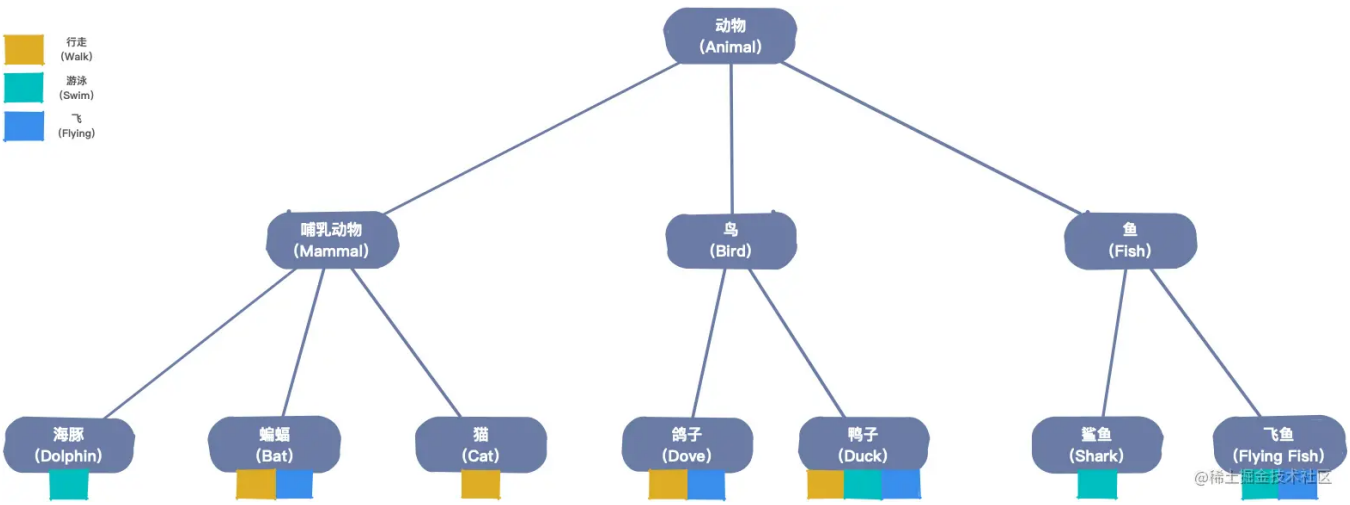
这篇文章, 我们一起来学习下, Dart基础语法中的 `mixin` 关键字。

Mixins

官方文档解释:

- Mixin 是一种在多重继承中复用某个类中代码的方法模式。
- 使用 `with` 关键字并在其后跟上 Mixin 类的名字来使用 Mixin 模式
- 想要实现一个 `mixin` , 那么, 这个类必须使用关键字 `mixin` , 必须继承 `Object` , 并且未声明构造函数
- 可以使用关键字 `on` 来指定哪些类可以使用该 `Mixin` 类

那么, Mixin 使用场景是什么？ Mixin 具体如何使用？下面, 我们以一个官方例子说明下。



在上图中，有一个类—— `Animal` ， 它有三个子类—— `Mammal` 、 `Bird` 及 `Fish` ， 而这三个类也有其对应的子类。

上面的关系图, 用Dart代码实现

scala 复制代码

```scala
/// 声明Animal基类
abstract Animal {...}
/// 哺乳动物类-->Animal
class Mammal extends Animal {...}
/// 鸟类-->Animal
class Bird extends Animal {...}
/// 鱼类--Animal
class Fish extends Animal {...}
/// 海豚-->哺乳动物类
class Dolphin extends Mammal {...}
/// 蝙蝠-->哺乳动物类
class Bat extends Mammal {...}
/// 猫-->哺乳动物类
class Cat extends Mammal {...}
/// 鸽子-->鸟类
class Dove extends Bird {...}
/// 鸭子-->鸟类
class Duck extends Bird {...}
/// 鲨鱼-->鱼类
class Shark extends Fish {...}
/// 飞鱼-->鱼类
class FlyingFish extends Fish {...}
```

下面就根据上图来分别给这些类添加行为—— `Walk` ， `Swim` 及 `Flying` 。由于这些行为并不是所有类通用的，所以不能将这些行为放在父类。因为Dart是单继承的, 没有多继承。所以, 我们可以用 `Mixin` 来实现上述需求。通过 `Mixin` 将上面的一些行为加入到各自对应的类中。

scala 复制代码

```scala
//行走
mixin Walker {
  void walk(){...}
}
//游泳行为
mixin Swim {
  void swim(){...}
}
//飞翔行为，由于这个是抽象方法，所以必须在要实现，不能调用super.flying()
mixin Flying {
  void flying();
}
//海豚可以游泳
class Dolphin extends Mammal with Swim {
  @override
  void swim() {
    super.swim();
  }
}
```

```
//蝙蝠可以飞、行走
class Bat extends Mammal with Flying, Walk {
  @override
  void flying() {...}
  //覆盖Walk类中的walk方法
  @override
  void walk() {
    super.walk();
  }
}
//猫可以行走，这里没有重写Walk中的方法
class Cat extends Mammal with Walk{}
//鸽子可以行走、飞
class Dove extends Bird with Flying,Walk{

  @override
  void flying() {...}
}
//鸭子可以行走、飞及游泳
class Duck extends Bird with Walk,Flying,Swim{
  @override
  void flying() {...}

  @override
  void walk() {...}
}
//鲨鱼可以游泳
class Shark extends Fish with Swim{...}
//飞鱼可以飞及游泳
class FlyingFish extends Fish with Flying,Swim{
  @override
  void flying() {...}
}
```

我们发现 `mixin` 里有方法的具体实现，这样可以避免接口的方法必须在子类实现从而导致的代码冗余。

mixin之线性化

在上面的示例中，我们发现 `with` 关键字后有多个类。那么这里就产生了一个问题——如果 `with` 后的多个类中有相同的方法，那么当调用该方法时，会调用哪个类里的方法? 由于距离 `with` 关键字越远的类会重写前面类中的相同方法，因此分为以下两种情况

scss 复制代码

```
class A {
  printMessage() => print('A');
}

mixin B on A {
  printMessage() {
    super.printMessage();
    print('B');
  }
}

mixin C on B {
  printMessage() {
        print('C0');
    super.printMessage();
        print('C1');
  }
}

class D with A, B, C {
  printMessage() {
   print('D0');
   super.printMessage();
   print('D1');
  };
}

void main() {
  D().printMessage();
}
```

输出结果:

```
D0
C0
A
B
C1
D1
```

由上面程序可知, 当对象 `D` 调用 `printMessage` 方法时, 先执行的是类 `D` 中的方法。再执行 `super.printMessage();` , 因为类 `D` 使用 `with` 关键字混合Mixins了 `A, B, C` 。并且 `A, B, C` 类中都有 `printMessage()` 方法, 那么, 当在 `D` 中执行 `super.printMessage();` 时, 程序会先执行 `A, B, C` 中的哪个类的 `printMessage()` 方法呢? 从程序的输出结果, 很清晰地看到, 程序执

行顺序是 `A-->B-->C-->D` 。也就是, 当 `with` 混合多个 `mixin` 时, 程序先从最后面的 `mixin` 类开始执行。

我们得出重要的结论:

- **with 后面的类会覆盖前面的类的同名方法**
- **如果当前使用类重写了该方法，就会调用当前类中的方法。**
- **如果当前使用类没有重写了该方法，则会调用距离 `with` 关键字最远类中的方法。**

参考资料

使用 Mixin 为类添加功能

深入理解 Dart 中的 Mixin

Dart之Mixin详解

分类: 前端 标签: 前端