

【Kotlin】Pair

Pair

在 Kotlin 中，`Pair` 是一個用於表示一對相關數據的數據結構。`Pair` 是一個簡單的二元組，包含兩個值，這兩個值可以是不同類型的。`Pair` 類的主要目的是為了方便存儲和傳遞成對的數據。

創建 Pair

你可以使用 `Pair` 類的構造函數來創建一個 `Pair` 對象。構造函數接受兩個參數，分別對應 `first` 和 `second` 值。

```
val pair = Pair("Hello", 42)
println(pair.first) // 輸出: Hello
println(pair.second) // 輸出: 42
```

Kotlin 也提供了一個方便的中綴函數 `to`，它可以用來創建一個 `Pair` 對象。這使得創建 `Pair` 更加簡潔。

```
val pair = "Hello" to 42
println(pair.first) // 輸出: Hello
println(pair.second) // 輸出: 42
```

使用 Pair

`Pair` 可以用於各種場景，比如返回多個值的函數，或者需要將兩個相關數據存儲在一起的情況。

```
// 函數返回一個 Pair
fun getPersonInfo(): Pair<String, Int> {
    return "Alice" to 30
}

val personInfo = getPersonInfo()
println("Name: ${personInfo.first}, Age: ${personInfo.second}")
// 輸出: Name: Alice, Age: 30
```

Pair 在 Map 中的應用

`Pair` 在處理 Map 時非常有用。例如，你可以使用 `Pair` 來初始化一個 Map，或者在操作 Map 時使用 `Pair`。

```
val map = mapOf("one" to 1, "two" to 2, "three" to 3)
println(map) // 輸出: {one=1, two=2, three=3}

// 使用 Pair 更新 Map
val updatedMap = map + ("four" to 4)
println(updatedMap) // 輸出: {one=1, two=2, three=3, four=4}
```

解構 Pair

Kotlin 提供了解構聲明，這使得從 `Pair` 中提取數據變得更加簡單。

```
val (name, age) = getPersonInfo()
println("Name: $name, Age: $age")
// 輸出: Name: Alice, Age: 30
```

Pair 類的其他方法

`Pair` 類還提供了一些其他有用的方法，比如 `toString()`、`equals()` 和 `hashCode()`，這些方法可以讓你更方便地使用 `Pair`。

```
val pair1 = "Hello" to 42
val pair2 = Pair("Hello", 42)

println(pair1.toString()) // 輸出: (Hello, 42)
println(pair1 == pair2)   // 輸出: true
println(pair1.hashCode()) // 輸出: 依據具體實現的哈希值
```

總的來說，`Pair` 是一個簡單但功能強大的數據結構，適合用於存儲和操作成對的數據。在 Kotlin 中，`Pair` 的使用非常直觀且方便。

🔄修訂版本 #2
★由 treeman 建立於 31 🇨🇹🇹🇹 2024 10:14:49
✍由 treeman 更新於 1 🇰🇪🇪🇪 2024 11:43:19