

Kotlin 1.3 さらっと Contracts + a

LINE Developers Meetup #47 in Fukuoka

@dashimaki_dofu


プロフィール



- 名前: 高 嘉泰 (Yoshiyasu KO)
- 所属: ベガコーポレーション
- Androidエンジニア
- : @dashimaki_dofu

プロフィール



- 名前: 高 嘉泰 (Yoshiyasu KO)
- 所属: ベガコーポレーション
- Androidエンジニア
- : @dashimaki_dofu

「しんちゃんの人」 と覚えて帰ってください

What is Kotlin?

Kotlin (ことりん)



- 2013年 JetBrains社が開発したオブジェクト指向型言語
- JVM上で動作
- Android Studio 3.0より公式サポート

2018.10.30
Kotlin 1.3リリース！

Kotlin 1.3

- Coroutines 正式版
- Kotlin/Native β版
- Contracts
- when 引数のスコープ最適化 etc...

Kotlin 1.3

- Coroutines 正式版
 - Kotlin/Native β版
- Contracts
 - when 引数のスコープ最適化
- etc...

今日話すのはこちら

Contracts

Contracts = 契約

- 関数の振る舞いを「契約」する
- コンパイラに関数の振る舞いを教える

?????

Smart Cast

```
fun foo(s: String?) {  
    if (s != null) s.length  
}
```

Smart Cast

```
fun foo(s: String?) {  
    if (s != null) s.length  
}
```

s は Nullable

Smart Cast

```
fun foo(s: String?) {  
    if (s != null) s.length  
}
```

NotNullを保証

Smart Cast

```
fun foo(s: String?) {  
    if (s != null) s.length  
}
```

NotNullを保証

!! が不要

例: `isNullOrEmpty()`

~ Kotlin 1.2

```
fun foo(s: String?) {  
    if (!s.isNullOrEmpty()) s!!.length  
}
```


例: `isNullOrEmpty()`

~ Kotlin 1.2

```
fun foo(s: String?) {  
    if (!s.isNullOrEmpty()) s!!.length  
}
```

s が 「null or 空文字」 であればtrue
= null でも 空文字でもなければ false
= false であればNotNullが保証されるはず

例: isNullOrEmpty()

~ Kotlin 1.2

```
fun foo(s: String?) {  
    if (!s.isNullOrEmpty()) s!!.length  
}
```

コンパイラ は isNullOrEmpty の「振る舞い」が分からない
= Smart Cast が効かない・・・

例: `isNullOrEmpty()`

Kotlin 1.3

```
fun foo(s: String?) {  
    if (!s.isNullOrEmpty()) s.length  
}
```

`isNullOrEmpty` に実装された Contracts によって
Smart Castが効くようになる！

例: isNullOrEmpty()

Kotlin 1.3

```
@kotlin.internal.InlineOnly
public inline fun CharSequence?.isNullOrEmpty(): Boolean {
    contract {
        returns(false) implies (this@isNullOrEmpty != null)
    }

    return this == null || this.length == 0
}
```

例: isNullOrEmpty()

Kotlin 1.3

```
@kotlin.internal.InlineOnly
public inline fun CharSequence?.isNullOrEmpty(): Boolean {
    contract {
        returns(false) implies (this@isNullOrEmpty != null)
    }

    return this == null || this.length == 0
}
```

例: isEmpty()

Kotlin 1.3

DSL(後述)で記述される

```
@kotlin.internal.InlineOnly
public inline fun CharSequence?.isEmpty(): Boolean {
    contract {
        returns(false) implies (this@isEmpty != null)
    }

    return this == null || this.length == 0
}
```

例: isNullOrEmpty()

Kotlin 1.3

```
@kotlin.internal.InlineOnly
public inline fun CharSequence?.isNullOrEmpty(): Boolean {
    contract {
        returns(false) implies (this@isNullOrEmpty != null)
    }
    return this == null || this.length == 0
}
```

Diagram illustrating the `returns(false) implies (this@isNullOrEmpty != null)` contract:

- A blue box highlights the contract block.
- A blue box labeled **false を返す時** (When returning false) points to the `returns(false)` part of the contract.
- The contract implies that `this@isNullOrEmpty != null` when `false` is returned.

例: isEmpty()

Kotlin 1.3

```
@kotlin.internal.InlineOnly
public inline fun CharSequence?.isEmpty(): Boolean {
    contract {
        returns(false) implies (this@isEmpty != null)
    }
    return this == null || this.isEmpty()
}
```

Diagram illustrating the contract of the `isEmpty()` function:

- returns(false) implies (this@isEmpty != null)**: This contract ensures that if the function returns `false`, it guarantees that `this` is not `null`.
- false を返す時**: When the function returns `false`, it implies that the receiver is not `null`.
- NotNullを保証することを契約する**: The contract guarantees that the receiver is not `null` when the function returns `false`.

例: isEmpty()

Kotlin 1.3

```
@kotlin.internal.InlineOnly
public inline fun CharSequence?.isEmpty(): Boolean {
    contract {
        returns(false) implies (this@isEmpty != null)
    }
    return this != null
}
```

Diagram illustrating the contract of the `isEmpty()` function:

- `returns(false)` implies `this@isEmpty != null`
- `false` を返す時
- `this`
- NotNullを保証することを契約する

コンパイラに関数の振る舞いを伝えることができる

Contracts DSL

- `returns()`
 - 関数の実行に成功した時
- `returns(Boolean?)`
 - 関数が引数の値を返した時
- `returnsNotNull()`
 - 関数がNotNullを返した時
- `callsInPrace(block, InvocationKind)`
 - `block` が呼ばれる回数を `InvocationKind` で保証する

Contracts DSL

- `returns()`
 - 関数の実行に成功した時
- `returns(Boolean?)`
 - 関数が引数の値を返した時
- `returnsNotNull()`
 - 関数がNotNullを返した時
- `callsInPrace(block, InvocationKind)`
 - `block` が呼ばれる回数を `InvocationKind` で保証する

自分でContractを作る
= Custom Contracts

when 引数のスコープ最適化

when 引数のスコープ最適化

~ Kotlin 1.2

```
fun Request.getBody() {  
    val response = executeRequest()  
    return when (response) {  
        is Success -> response.body  
        is HttpError -> throw HttpException(response.status)  
    }  
}
```

when 引数のスコープ最適化

~ Kotlin 1.2

```
fun Request.getBody() {  
    val response = executeRequest()  
    return when (response) {  
        is Success -> response.body  
        is HttpError -> throw HttpException(response.status)  
    }  
}
```

ローカル変数をここで
定義する必要がある

when 引数のスコープ最適化

~ Kotlin 1.2

```
fun Request.getBody() {  
    val response = executeRequest()  
    return when (response) {  
        is Success -> response.body  
        is HttpError -> throw HttpException(response.status)  
    }  
}
```

ローカル変数をここで
定義する必要がある

when だけで使う変数のスコープが広がってしまう

when 引数のスコープ最適化

Kotlin 1.3

```
fun Request.getBody() =  
    when (val response = executeRequest()) {  
        is Success -> response.body  
        is HttpError -> throw HttpException(response.status)  
    }
```


when 引数のスコープ最適化

Kotlin 1.3

引数内でローカル変数が
定義できる


```
fun Request.getBody() =  
    when (val response = executeRequest()) {  
        is Success -> response.body  
        is HttpError -> throw HttpException(response.status)  
    }
```

when 引数のスコープ最適化

Kotlin 1.3

引数内でローカル変数が
定義できる

```
fun Request.getBody() =  
    when (val response = executeRequest()) {  
        is Success -> response.body  
        is HttpError -> throw HttpException(response.status)  
    }
```



引数で利用する変数のスコープを
when ブロック内で収めることができる

今日話したこと

- **Contract**
 - コンパイラに関数の「振る舞い」を伝える
 - スタンダードライブラリにも組み込まれている
 - DSL で記述される
 - Custom Contracts も定義できる
- **when 文の引数のスコープ最適化**
 - ローカル変数を引数内で定義できる
 - when ブロック内にスコープが収められる

まとめ

かゆい所に
手が届いたゾ！



ほほーい



じゃ
そゆことで～