Kotlin Coroutines

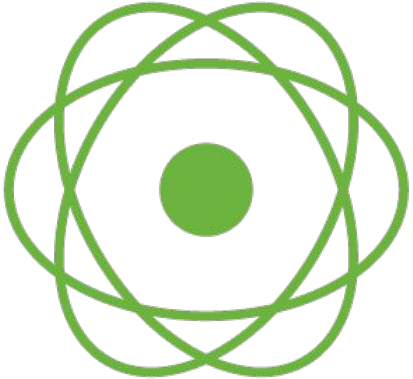
Часть 2

Practical

ktor

Anko

VERT.X



Http Client + Coroutines

```
suspend fun HttpAsyncClient.execute(request: HttpUriRequest): HttpResponse {
    return suspendCancellableCoroutine { cont: CancellableContinuation<HttpResponse> →
        val future :Future<HttpResponse!>! = this.execute(request, object : FutureCallback<HttpResponse> {
            override fun completed(result: HttpResponse) { cont.resume(result) }
            override fun cancelled() {}
            override fun failed(ex: Exception) { cont.resumeWithException(ex) }
        })

        cont.cancelFutureOnCompletion(future);
        Unit
    }
}
```

Http Client + Coroutines

```kotlin
suspend fun geoIp(url: String): String {
    val request = HttpGet( uri: "http://freegeoip.net/json/$url")
    return httpClient.execute(request).text()
}
```

Coroutines + Spring

```
@RestController
class BlogController(
    val blogService: BlogService
) {

    @PostMapping("/{token}")
    fun createArticle(@PathVariable token: String, article: Article): CompletableFuture<String> {
        return future {
            val result = blogService.post(token, article)

            when (result) {
                is Success → result.data
                is Fail → throw RuntimeException(result.message)
            }
        }
    }
}
```

Http Server + Coroutines

```
class CoroutinesHandler(
    private val context: CoroutineContext = singleThreadContext,
    private val handler: suspend (HttpServerExchange) → Unit
) : HttpHandler {
    override fun handleRequest(exchange: HttpServerExchange) {
        exchange.dispatch(Runnable {
            launch(context = context, start = CoroutineStart.UNDISPATCHED) {
                handler(exchange)
            }
        })
    }
}
```

Http Server + Coroutines

```
val httpHandler = CoroutinesHandler { exchange →
    delay( time: 1000)
    exchange.responseSender.send( data: "Hello, World!")
}
```

Demo

CoroutineContext

```
coroutineContext[ContinuationInterceptor.Key]
coroutineContext[Job.Key]
coroutineContext[CoroutineName.Key]
coroutineContext[CoroutineExceptionHandler.Key]
```

Pro Tip

```kotlin
suspend fun getCoroutineContext(): CoroutineContext =
    suspendCoroutine { continuation →
        continuation.resume(continuation.context)
    }
```

Pro Tip

```
suspend fun client() {
    val coroutinesContext : CoroutineContext = getCoroutineContext()

    val interceptor : ContinuationInterceptor? = coroutinesContext[ContinuationInterceptor.Key]
}
```

Pro Tip

```
suspend fun client(continuation: Continuation<T>) {
    val coroutinesContext : CoroutineContext = getCoroutineContext()

    val interceptor : ContinuationInterceptor? = coroutinesContext[ContinuationInterceptor.Key]
}
```

Bytecode

```
package by.heap.komodo.samples.coroutines.bytecode

import kotlinx.coroutines.experimental.delay

suspend fun fetch() {
    delay(1000)
}
```

Bytecode

```
-rw-r--r-- 1 yoda yoda 1342 Jun  1 08:03 ExampleKt.class
-rw-r--r-- 1 yoda yoda 1833 Jun  1 08:03 ExampleKt$fetch$1.class
```

Bytecode

```
public final class ExampleKt {
  public static final Object fetch(
    Continuation<? super Unit>
  );
}
```

Bytecode

```
public final class ExampleKt {
    @Nullable
    public static final Object fetch(@NotNull final Continuation<? super
Unit> $continuation) {
    Intrinsics.checkParameterIsNotNull((Object)$continuation,
"$continuation");
    return new
ExampleKt$fetch.ExampleKt$fetch$1((Continuation)$continuation).doResume((Ob
ject)Unit.INSTANCE, (Throwable)null);
    }
}
```

Bytecode

```
final class ExampleKt$fetch$1 extends CoroutineImpl {
  public final Object doResume(Object, Throwable);
  ExampleKt$fetch$1(Continuation);
}
```

Bytecode

```
static final class ExampleKt$fetch$1 extends CoroutineImpl {
      @Nullable
      public final Object doResume(@Nullable final Object data, @Nullable final Throwable throwable) {
      final Object coroutine_SUSPENDED = IntrinsicsKt.getCOROUTINE_SUSPENDED();
      switch (super.label) {
            case 0: {
            ...
            break;
            }
            case 1: {
                  ...
                  break;
            }
            default: {
                  throw new IllegalStateException("call to 'resume' before 'invoke' with coroutine");
            }
      }
      return Unit.INSTANCE;
      }
}
```

buildSequence

kotlin.coroutines.experimental.

buildSequence

```
public fun <T> buildSequence(
    builderAction: suspend SequenceBuilder<T>.() -> Unit
): Sequence<T> = Sequence { buildIterator(builderAction) }
```

kotlin.coroutines.experimental.buildSequence

```
val lazySeq: Sequence<Int> = buildSequence {
    for (i in 1..100) {
        yield(i)
    }
}

lazySeq.take(3).forEach { print(it) }
// 123
```

kotlin.coroutines.experimental.

buildSequence

```
val lazySeq: Sequence<Int> = buildSequence {
    for (i in 1..100) {
        delay(1000)
        yield(i)
    }
}
```

Error:(22, 9) Kotlin: Restricted suspending functions can only invoke member or extension suspending functions on their restricted coroutine scope

```
public fun <T> buildSequence(
    builderAction: suspend SequenceBuilder<T>.() -> Unit
): Sequence<T> = Sequence { buildIterator(builderAction) }
```

kotlin.coroutines.experimental.

SequenceBuilder

```
@RestrictsSuspension
public abstract class SequenceBuilder<in T> internal constructor() {
    public abstract suspend fun yield(value: T)
    public abstract suspend fun yieldAll(iterator: Iterator<T>)
    ...
}
```

kotlin.coroutines.experimental.

SequenceBuilder

```
suspend fun SequenceBuilder<Int>.answer() {
    this.yield(42)
}

val ultimateAnswerSeq: Sequence<Int> = buildSequence {
    while (true) {
        answer()
    }
}
```

Performance

Performance and Benchmarks

Lib	Remote PingPong	Inproc PingPong	SkyNet
Proto.Actor Kotlin	~2 700 000 msg/sec	~160 000 000 msg/sec	~0.31 sec
Proto.Actor C#	~2 500 000 msg/sec	~125 000 000 msg/sec	~0.8 sec
Proto.Actor Go	~2 400 000 msg/sec	~120 000 000 msg/sec	~1.5 sec
Akka	?	~38 000 000 msg/sec	?
Akka.NET	~38 000 msg/sec	~30 000 000 msg/sec	~12 sec
Erlang	~200 000 msg/sc	~12 000 000 msg/sec	~0.75 sec

http://proto.actor/docs/performance#performance-and-benchmarks

Q&A

KOMODO

Ruslan Ibragimov

Belarus Kotlin User Group: https://bkug.by/

Java Professionals BY: http://jprof.by/

Awesome Kotlin: https://kotlin.link/

Slides: https://goo.gl/XThZxE