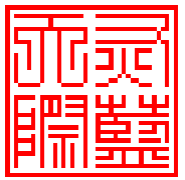
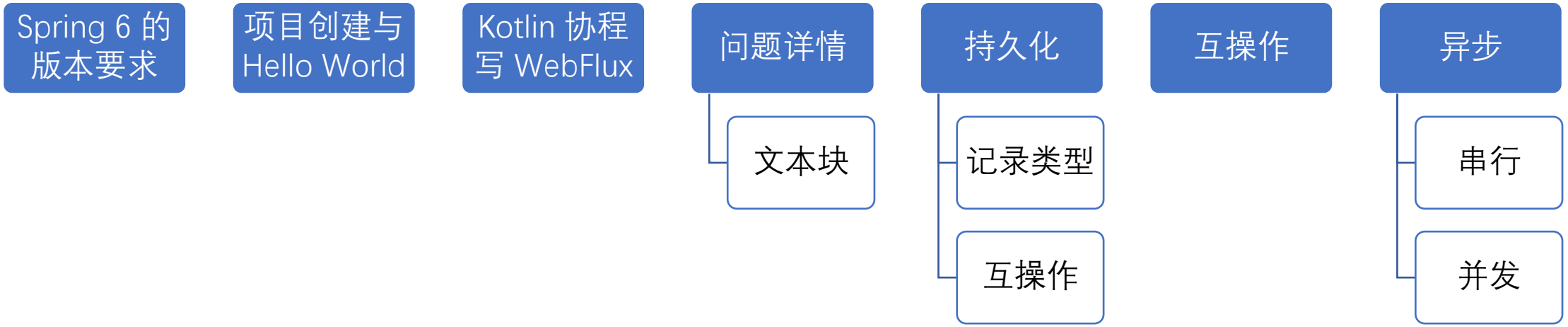
与时俱进
使用 Kotlin 尝鲜 Spring 6



贾彦伟
hltj.me

内容提要



插播：Koltin 中文站 GitBook 持续更新

- https://book.kotlincn.net/
- 已更新到当前最新（1.7.21）
- 期待大家参与翻译，翻译前请参考 翻译指南 https://github.com/hltj/kotlin-web-site-cn/issues/35

“鲜”

Spring 6

10 天前
(2022-11-16)
正式发布

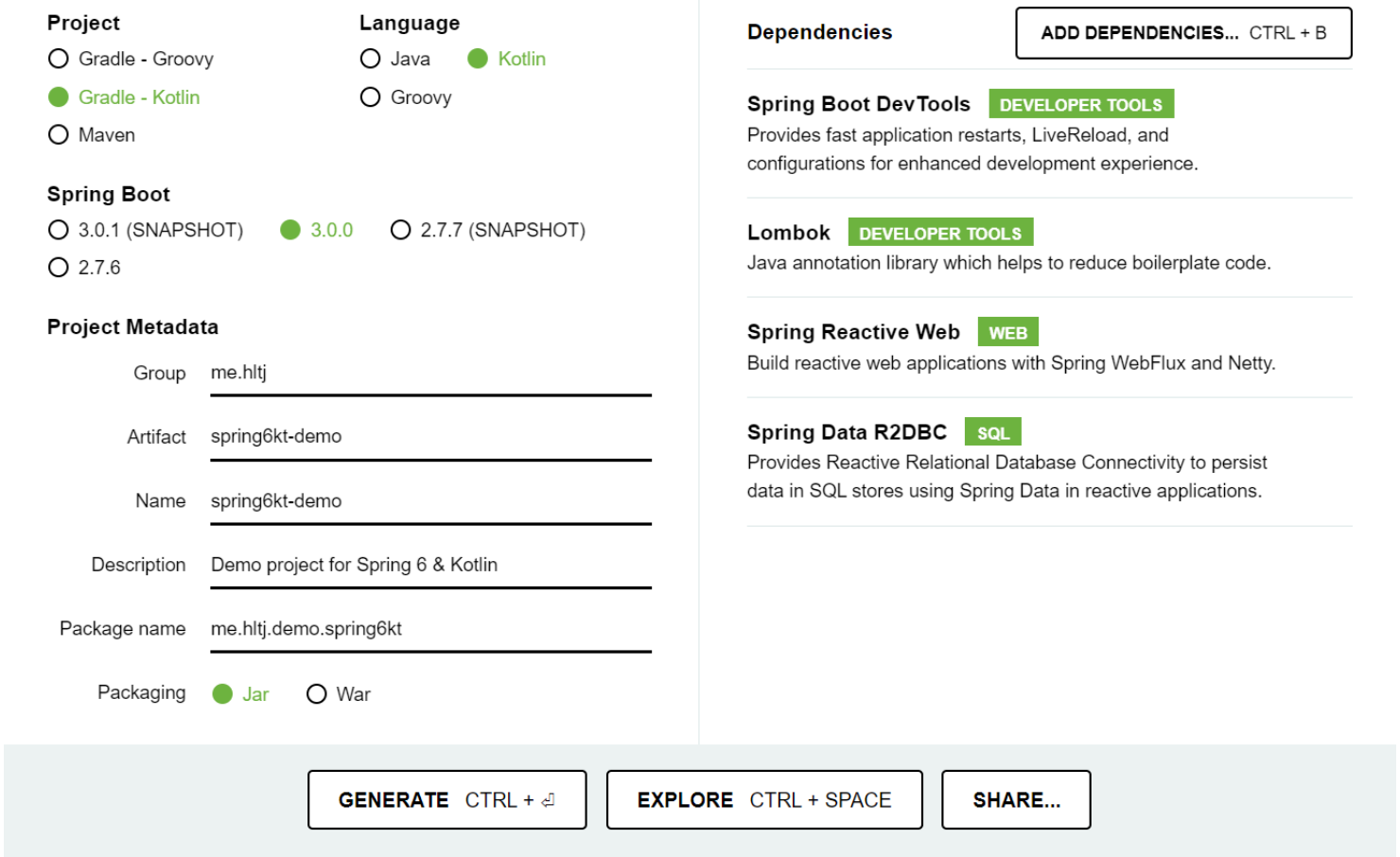
Spring Boot 3

2 天前
(2022-11-24)
正式发布

“鲜”

	版本
Java	**17+**
Kotlin	**1.7+**
Jakarta	9+
……	……

创建项目


Project
Gradle - Groovy
Gradle - Kotlin
Maven
Language
Java
Kotlin
Groovy
Spring Boot
3.0.1 (SNAPSHOT)
3.0.0
2.7.7 (SNAPSHOT)
2.7.6
Project Metadata
Group
me.hltj
Artifact
spring6kt-demo
Name
spring6kt-demo
Description
Demo project for Spring 6 & Kotlin
Package name
me.hltj.demo.spring6kt
Packaging
Jar
War
Dependencies
ADD DEPENDENCIES... CTRL + B
Spring Boot DevTools
DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
Lombok
DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.
Spring Reactive Web
WEB
Build reactive web applications with Spring WebFlux and Netty.
Spring Data R2DBC
SQL
Provides Reactive Relational Database Connectivity to persist data in SQL stores using Spring Data in reactive applications.
GENERATE CTRL + ⏎
EXPLORE CTRL + SPACE
SHARE...

Hello World

Java

```java
@RestController
@RequestMapping("/j")
public class JSimpleController {

    hltj
    @GetMapping("/hello")
    String hello() {
        return "Hello Spring 6 & Java 17";
    }

}
```

Kotlin

```kotlin
@RestController
@RequestMapping("/k")
class KSimpleController {

    hltj
    @GetMapping("/hello")
    fun hello() = "Hello Spring 6 & Kotlin 1.7"

}
```

Hello WebFlux

```java
@GetMapping("/3s")
Mono<String> delay3s() {
    return Mono.just("Delayed 3s")
            .delayElement(Duration.ofSeconds(3));
}
```

```kotlin
@GetMapping("/3s")
suspend fun delay3s() = with(Duration) { this: Duration
    delay(3.seconds)
    "Delayed 3s" ^with
}
```

```java
@GetMapping("/1sx3")
Flux<Integer> delay1s3times() {
    return Flux.just(1, 2, 3)
            .delayElements(Duration.ofSeconds(1))
            .doOnNext(i →
                    System.out.printf("Delayed 1s %d/3%n", i)
            );
}
```

```kotlin
@GetMapping("/1sx3")
fun delay1s3times() = (1 ≤ .. ≤ 3).asFlow().onEach { it: Int
    with(Duration) { delay(1.seconds) }
    println("Delayed 1s $it/3")
}
```

问题详情

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json
Content-Language: zh-CN

{
  "type": "/problem/invalid-params",
  "title": "参数错误",
  "status": 400,
  "detail": "请求所传 age 与 color 参数不正确",
  "instance": "/k/problem-demo"
  "invalid_params": [
    {
      "name": "age",
      "reason": "must be a positive integer"
    },
    {
      "name": "color",
      "reason": "must be 'red', 'green' or 'blue'"
    }
  ]
}
```

问题详情

```yaml
spring:
  webflux:
    problemdetails:
      enabled: true
```

问题详情

Java

```java
@GetMapping("/problem-demo")
Mono<Void> problemDemo() {
    var pd = ProblemDetail.forStatusAndDetail(HttpStatus.BAD_REQUEST, "请求所传 age 与
    pd.setType(URI.create("/problem/invalid-params"));
    pd.setTitle("参数错误");
    pd.setProperty("invalid_params", OBJECT_MAPPER.readValue("""
            [
              {
                "name": "age",
                "reason": "must be a positive integer"
              },
              {
                "name": "color",
                "reason": "must be 'red', 'green' or 'blue'"
              }
            hltj
            ]""", new TypeReference<List<LinkedHashMap<String, Object>>>() {
    }));
    return Mono.just(true).delayElement(Duration.ofSeconds(3)).flatMap(x ->
            Mono.error(new ErrorResponseException(HttpStatus.BAD_REQUEST, pd, null))
    );
}
```

Kotlin

```kotlin
@GetMapping("/problem-demo")
suspend fun problemDemo() {
    val pd = ProblemDetail.forStatusAndDetail(HttpStatus.BAD_REQUEST, detail: "请求所传
    pd.type = URI.create("/problem/invalid-params")
    pd.title = "参数错误"
    pd.setProperty(
        "invalid_params",
        objectMapper.readValue<List<LinkedHashMap<String, Any>>>("""
            [
              {
                "name": "age",
                "reason": "must be a positive integer"
              },
              {
                "name": "color",
                "reason": "must be 'red', 'green' or 'blue'"
              }
            ]""".trimIndent()
        )
    )
    with(Duration) { delay(3.seconds) }
    throw ErrorResponseException(HttpStatus.BAD_REQUEST, pd, null)
}
```

(问题详情)：文本块

Java

```
private static final String INVALID_PARAMS_JSON_STR = """
        [
          {
            "name": "age",
            "reason": "must be a positive integer"
          },
          {
            "name": "color",
            "reason": "must be 'red', 'green' or 'blue'"
          }
        ]""";
```

Kotlin

```
private val invalidParamsJsonStr = """
        [
          {
            "name": "age",
            "reason": "must be a positive integer"
          },
          {
            "name": "color",
            "reason": "must be 'red', 'green' or 'blue'"
          }
        ]""".trimIndent()
```

https://github.com/bennyhuo/Kotlin-Trim-Indent

(问题详情)：JSON 解析

Java

```
var invalidParamsJson = OBJECT_MAPPER.readValue(
        INVALID_PARAMS_JSON_STR,
        hltj
        new TypeReference<List<LinkedHashMap<String, Object>>>() {
        });
```

Kotlin

```
val invalidParamsJson =
    objectMapper.readValue<List<LinkedHashMap<String, Any>>>(
        invalidParamsJsonStr
    )
```

(问题详情)：JSON 解析

Java

```
var invalidParamsJson = OBJECT_MAPPER.readValue(
        INVALID_PARAMS_JSON_STR,
        hltj
        new TypeReference<List<LinkedHashMap<String, Object>>>() {
        });
```

Kotlin

```
val invalidParamsJson: List<LinkedHashMap<String, Any>> =
    objectMapper.readValue(invalidParamsJsonStr)
```

问题详情：构造响应

Java

```
return Mono.just(true).delayElement(Duration.ofSeconds(3)).flatMap(x →
        Mono.error(new ErrorResponseException(HttpStatus.BAD_REQUEST, pd, null))
);
```

Kotlin

```
with(Duration) { delay(3.seconds) }
throw ErrorResponseException(HttpStatus.BAD_REQUEST, pd, null)
```

问题详情：Spring 6 自带

```
HTTP/1.1 400 Bad Request
Content-Type: application/problem+json

{
  "type": "about:blank",
  "title": "Bad Request",
  "status": 400,
  "detail": "Required query parameter 'name' is not present.",
  "instance": "/k/users"
}
```

持久化

Java

```java
@Table("users")
public record JUserEntity(@Id Integer id, @Nonnull String name, @Nonnull Instant regTime) {
}
```

```java
public interface JUserRepository extends ReactiveCrudRepository<JUserEntity, @NonNull Integer> {
}
```

```java
@Override
public Mono<JUser> register(@Nonnull String name) {
    return jUserRepository.save(new JUserEntity(null, name, Instant.now())).map(entity ->
            new JUser(entity.id(), entity.name(), entity.regTime())
    );
}
```

Kotlin

```kotlin
@Table("users")
@JvmRecord
data class KUserEntity(@Id val id: Int?, val name: String, val regTime: Instant)
```

```kotlin
interface KUserRepository : ReactiveCrudRepository<KUserEntity, Int>
```

```kotlin
override suspend fun register(name: String): KUser =
    kUserRepository.save(KUserEntity( id: null, name, Instant.now())).map { it: KUserEntity!
        KUser(it.id!!, it.name, it.regTime)
    }.awaitSingle()
```

持久化：记录类型

Java

```java
@Table("users")
public record JUserEntity(@Id Integer id,
}
```

Kotlin

```kotlin
@Table("users")
@JvmRecord
data class KUserEntity(@Id val id: Int?,
```

持久化： ReactiveCrudRepository

Java

```java
public interface JUserRepository extends
        ReactiveCrudRepository<JUserEntity, @NonNull Integer> {
}
```

Kotlin

```kotlin
interface KUserRepository :
    ReactiveCrudRepository<KUserEntity, Int>
```

(持久化)：Mono → suspend

Java

```java
@Override
public Mono<JUser> register(@Nonnull String name) {
    return jUserRepository.save( [illegible]
            [illegible]
    );
}
```

Kotlin

```kotlin
override suspend fun register(name: String): KUser =
    kUserRepository.save( [illegible]
        [illegible]
    }.awaitSingle()
```

持久化：自定义方法

Java

```java
public interface JUserRepository extends ReactiveCrudRepository<
    1 usage    hltj
    Mono<Integer> countByName(@NonNull String name);

    1 usage    hltj
    Flux<JUser> findAllByName(@NonNull String name);
}
```

Kotlin

```kotlin
interface KUserRepository : ReactiveCrudRepository<
    hltj
    suspend fun countByName(name: String): Int

    hltj
    fun findAllByName(name: String): Flow<KUser>
}
```

互操作：suspend → Mono

Java

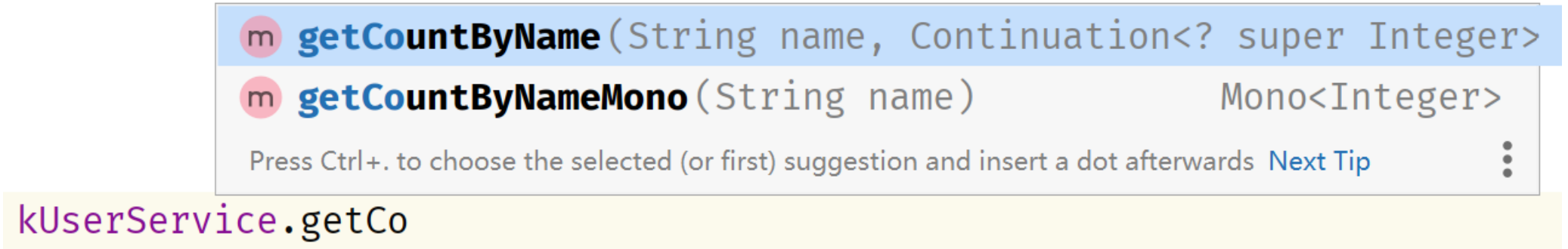
```java
@GetMapping("/_k1_count")
Mono<Integer> getKt1CountByName(@RequestParam String name) {
    return kUserService.getCountByNameMono(name);
}
```

Kotlin

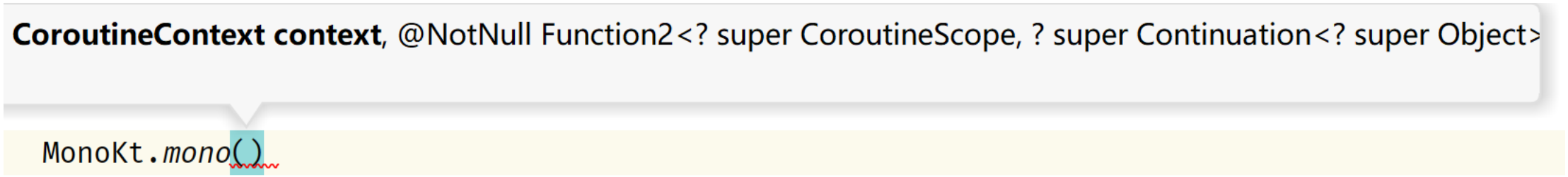
```kotlin
override suspend fun getCountByName(name: String): Int =
    kUserRepository.countByName(name)

hltj
override fun getCountByNameMono(name: String): Mono<Int> =
    mono { getCountByName(name) }
```

互操作：suspend → Mono



互操作： suspend → Mono



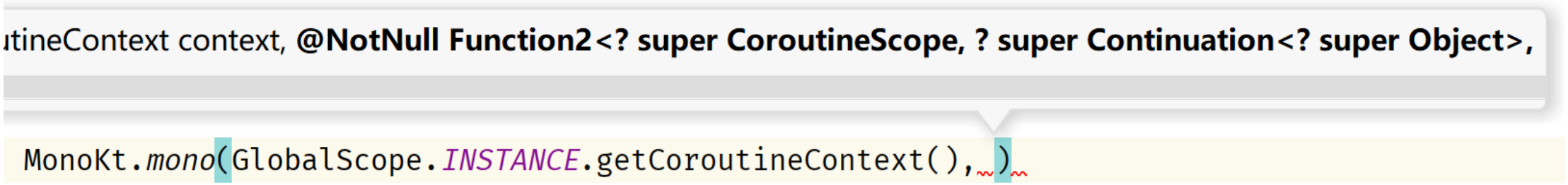
互操作：suspend → Mono

```
public fun <T> mono(
    context: CoroutineContext = EmptyCoroutineContext,
    block: suspend CoroutineScope.() → T?
): Mono<T> {
    require( value: context[Job] == null) { "Mono contex
            "Its lifecycle should be managed via Dispos
    return monoInternal(GlobalScope, context, block)
}
```

互操作：suspend → Mono

```
public object GlobalScope : CoroutineScope {
    Returns EmptyCoroutineContext.
    override val coroutineContext: CoroutineContext
        get() = EmptyCoroutineContext
}
```

互操作：suspend → Mono



互操作：suspend → Mono

```
@GetMapping("/_k2_count")
Mono<Integer> getKt2CountByName(@RequestParam String name) {
    return MonoKt.mono(
            GlobalScope.INSTANCE.getCoroutineContext(),
            (scope, continuation) →
                    kUserService.getCountByName(name, continuation)
    );
}
```

互操作： suspend → Mono

Java

```java
@GetMapping("/_k1_count")
Mono<Integer> getKt1CountByName(@RequestParam String name) {
    return kUserService.getCountByNameMono(name);
}
```

```java
@GetMapping("/_k2_count")
Mono<Integer> getKt2CountByName(@RequestParam String name) {
    return MonoKt.mono(
            GlobalScope.INSTANCE.getCoroutineContext(),
            (scope, continuation) →
                    kUserService.getCountByName(name, continuation)
    );
}
```

Kotlin

```kotlin
override suspend fun getCountByName(name: String): Int =
    kUserRepository.countByName(name)

hltj
override fun getCountByNameMono(name: String): Mono<Int> =
    mono { getCountByName(name) }
```

(互操作)：协程

15:10-15:50 Gray Lo

- 从零开始欣赏 Coroutine 的精湛设计

互操作： Flux → Flow

```kotlin
@GetMapping("")
fun getMulti(@RequestParam name: String): Flow<KUser> =
    kUserService.getAllByName(name)

hltj
@GetMapping("/_j")
fun getJMulti(@RequestParam name: String): Flow<JUser> =
    jUserService.getAllByName(name).asFlow()
```

互操作： Flow → Flux

Java

```java
@GetMapping("/_k1")
Flux<KUser> getK1Multi(@RequestParam String name) {
    return kUserService.getAllByNameFlux(name);
}
```

```java
@GetMapping("/_k2")
Flux<KUser> getK2Multi(@RequestParam String name) {
    return ReactorFlowKt.asFlux(
            kUserService.getAllByName(name)
    );
}
```

Kotlin

```kotlin
override fun getAllByName(name: String): Flow<KUser> =
    kUserRepository.findAllByName(name)
```

hltj

```kotlin
override fun getAllByNameFlux(name: String): Flux<KUser> =
    getAllByName(name).asFlux()
```

异步

Java

```java
Mono<Integer> countByName(@NonNull String name);

1 usage  hltj
Mono<JUser> getFirst1ByName(@NonNull String name);
```

```java
record JPreview(int count, @Nullable JUser first) {
}
```

Kotlin

```kotlin
suspend fun countByName(name: String): Int

hltj
suspend fun findFirst1ByName(name: String): KUser?
```

```kotlin
@JvmRecord
data class KPreview(val count: Int, val first: KUser?)
```

异步：串行

Java

```java
@Override
public Mono<JPreview> serialPreview(@Nonnull String name) {
    return
    jUserRepository.countByName(name).flatMap(count →
            jUserRepository.getFirst1ByName(name).map(first →
                    new JPreview(count, first)
            ).defaultIfEmpty(new JPreview(count, null))
    );
}
```

Kotlin

```kotlin
override suspend fun serialPreview(name: String): KPreview {
    val count = kUserRepository.countByName(name)
    val first = kUserRepository.findFirst1ByName(name)
    return KPreview(count, first)
}
```

异步：并发

Java

```java
@Override
public Mono<JPreview> concurrentPreview(String name) {
    var countMono = jUserRepository.countByName(name);
    var firstOptMono = jUserRepository.getFirst1ByName(name)
            .map(Optional::of)
            .defaultIfEmpty(Optional.empty());
    return Mono.zip(countMono, firstOptMono, (count, firstOpt) ->
            new JPreview(count, firstOpt.orElse(null))
    );
}
```

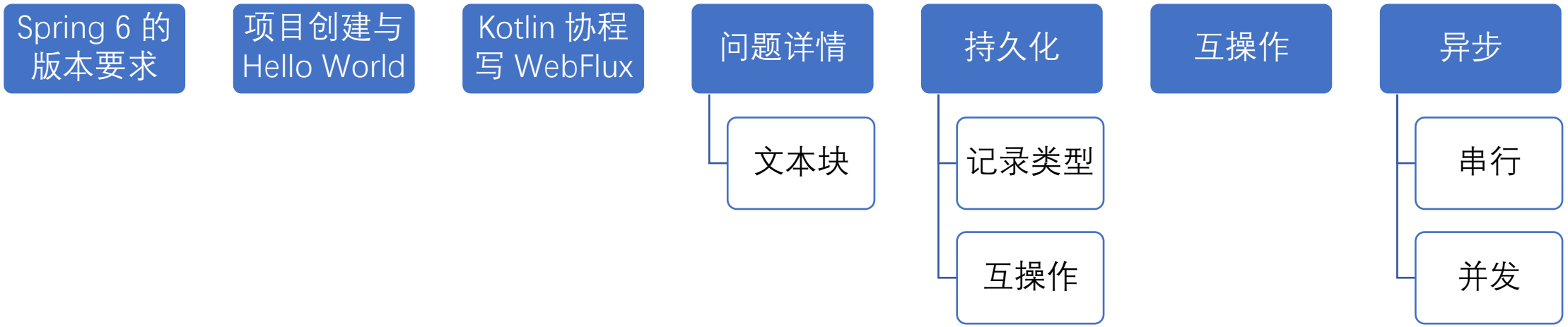
Kotlin

```kotlin
override suspend fun concurrentPreview(name: String) = coroutineScope {
    val asyncCount = async { kUserRepository.countByName(name) }
    val asyncFirst = async { kUserRepository.findFirst1ByName(name) }
    KPreview(asyncCount.await(), asyncFirst.await())^coroutineScope
}
```

示例代码

- https://github.com/hltj/spring6kt-demo

小结


Spring 6 的版本要求
项目创建与 Hello World
Kotlin 协程写 WebFlux
问题详情
文本块
持久化
记录类型
互操作
互操作
异步
串行
并发

Q & A

公众号《灰蓝时光》	微博：灰蓝天际