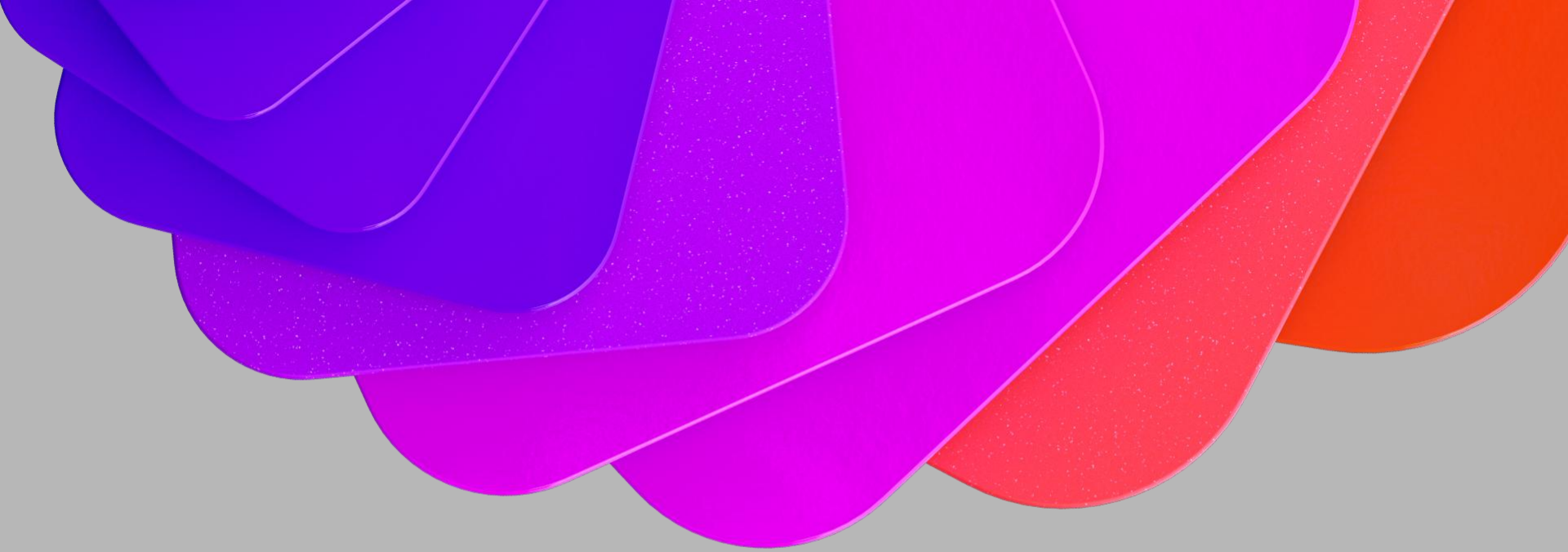

# 从喜欢到生产中使用，从安卓小玩具到后端应用：我与 Kotlin 的故事

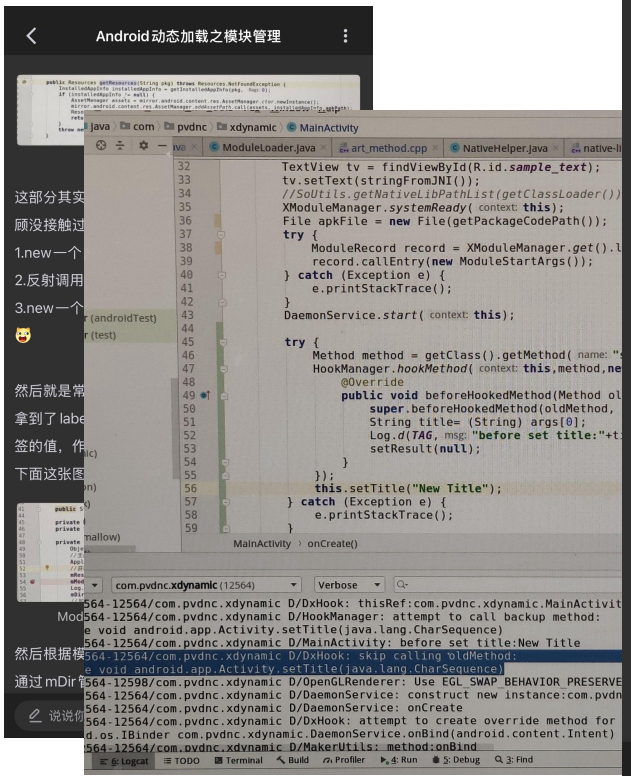
Any

Developed by JetBrains | @AkaAny



# 我与Kotlin的结缘

# 从前有座山.....





## Android 的 Kotlin 优先方法

本页内容

- 为什么要优先使用 Kotlin 进行 Android 开发？
- Kotlin 优先意味着什么？
- 我们也使用 Kotlin！

在 2019 年 Google I/O 大会上，我们宣布今后将优先采用 Kotlin 进行 Android 开发，并且也坚守了这一承诺。Kotlin 是一种富有表现力且简洁的编程语言，不仅可以减少常见代码错误，还可以轻松集成到现有应用中。如果您想构建 Android 应用，建议您从 Kotlin 开始着手，充分利用一流的 Kotlin 功能。

为了支持使用 Kotlin 进行 Android 开发，我们和另一组织联手创办了 Kotlin 基金会，不断投入人力物力来提高编译器性能和 build 速度。如需详细了解 Android 的 Kotlin 优先承诺，请参阅 Android 在 Kotlin 方面的承诺。

Kotlin

### 为什么要优先使用 Kotlin 进行 Android 开发？

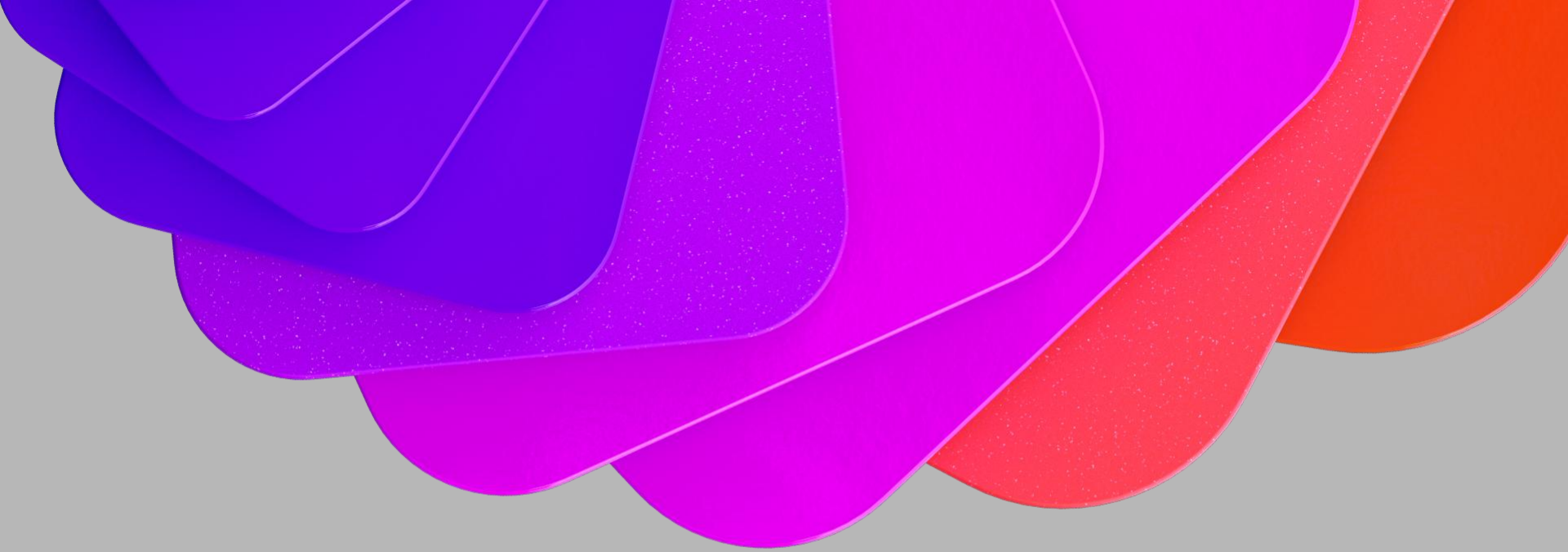
# 从java后端到Kotlin后端

# 广子 - 杭电助手

社员的助手，学生的助手

为学生提供了接触实际项目机会
实现个人与社团共赢

合作、社交、内推……

# “用了就回不去”
## ——我为什么喜欢Kotlin

# 流畅，激发创造欲的语法


open fun getUserProfile(userCode: String): UserProfileVO {
return JoinWrappers
.lambda(UserDO::class.java)
fun listByID(ids:Co
if(ids.isEmpty()
return empty
}
return JoinWrapp
.selectAll(U
.inWhenNotEr
.let {
return@
}
}
Create a Dp using an Int : val left = 10 val x = left.dp // -- or -- val y = 10.dp
@Stable
inline val Int.dp: Dp get() = Dp(value = this.toFloat())
Create a Dp using a Double : val left = 10.0 val x = left.dp // -- or -- val y = 10.0.dp
@Stable
inline val Double.dp: Dp get() = Dp(value = this.toFloat())
Create a Dp using a Float : val left = 10f val x = left.dp // -- or -- val y = 10f.dp
@Stable
inline val Float.dp: Dp get() = Dp(value = this)
ist<UserTagItemDO>>(
erDTO::activeAvatarHistoryItemDO)
java, it);

# 模板字符串+文本块+Language注解=?

兼具灵活性、通用性，
而且易读的查询!

```
fun listTopLevelNode(): List<TreeNodeDTO>{
    class ListTopLevelParamObject: IManualQuery
        override fun toVariableMap(): Map<Strin
            return emptyMap();
        }
    };
    @Language("Cypher")
    val query="""
        MATCH (s:${nodeName})
          WHERE NOT((s)<-[:${nextLevelEdgeName}]-(:${nodeName}))
        WITH CASE
          WHEN valueType(s) = 'NODE NOT NULL' THEN [s]
          WHEN valueType(s) = 'NULL' THEN []
          ELSE s
          END AS s_list
        UNWIND s_list AS s
          RETURN s AS node,NOT((s)-[:${nextLevelEdgeName}]->(:${nodeName})) AS leaf
    """.trimIndent()
    val paramObject= ListTopLevelParamObject();
    val dtoList= mutableListOf<TreeNodeDTO>();
    neo4jClient.query(query)
        .bindAll(paramObject.toVariableMap())
        .mappedBy{typeSystem,record->
            treeNodeDTOMappedByFunc(typeSystem,record,dtoList);
        }
        .all();
    return dtoList;
}
```

```
fun listNextLevel(nodeCode: String): List<TreeNodeDTO>{  Any *
    ramObject(val nodeCode:String): IManualQueryParamObject {
    eMap(): Map<String, Any> {  Any *
    ne to nodeCode
    @Language("Cypher")
    val query="""
        MATCH (s:${nodeName}{${nodeCodePropertyName}:${nodeCodeVar}})
        WITH s
        MATCH (s)-[:${nextLevelEdgeName}]->(c:${nodeName})
        WITH CASE
          WHEN valueType(c) = 'NODE NOT NULL' THEN [c]
          WHEN valueType(c) = 'NULL' THEN []
          ELSE c
          END AS c_list
        UNWIND c_list AS c
          RETURN c AS node,NOT((c)-[:${nextLevelEdgeName}]->(:${nodeName})) AS leaf
    """;
    val paramObject=ListNextLevelParamObject(nodeCode);
    val dtoList= mutableListOf<TreeNodeDTO>();
    neo4jClient.query(query)
        .bindAll(paramObject.toVariableMap())
        .mappedBy{typeSystem,record->
            treeNodeDTOMappedByFunc(typeSystem,record,dtoList);
        }
        .all();
    return dtoList;
}
```

# 语法糖铺子

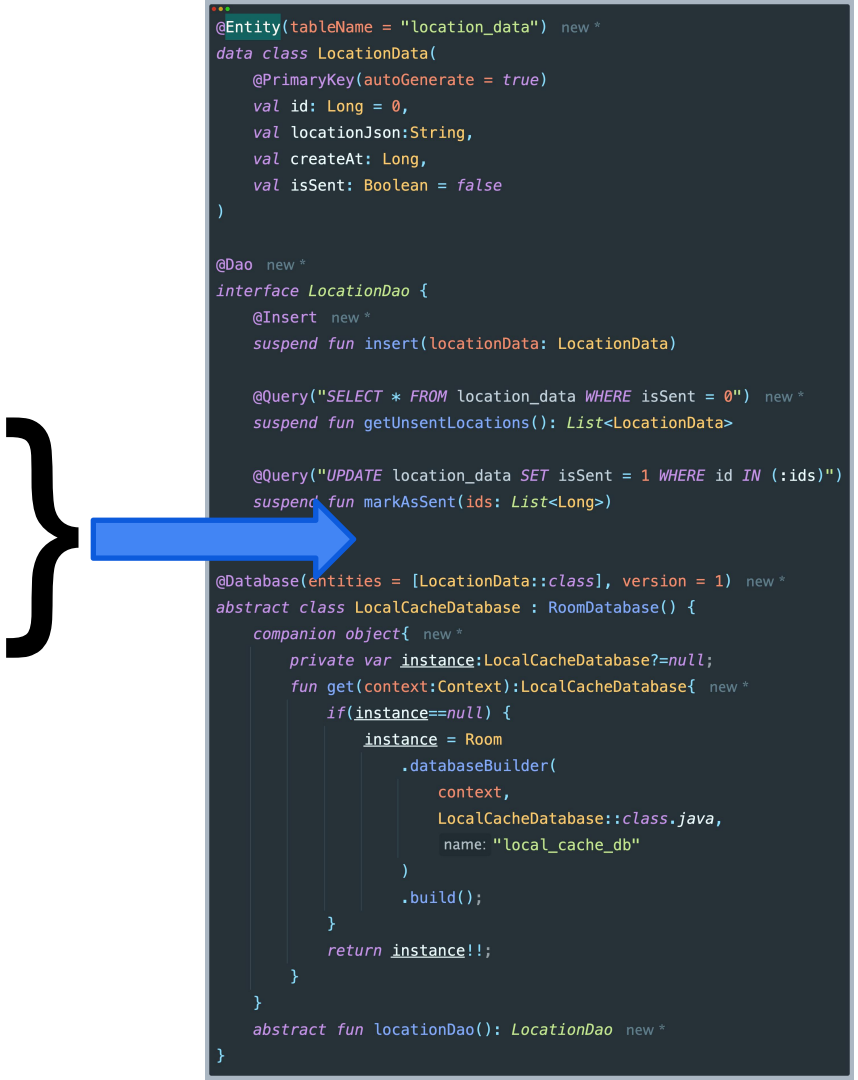
```kotlin
data class InQueryResult<T,K>(
    val foundList:List<T>,
    val notFoundKeySet:Set<K>
)

fun <T,K,Z:Collection<K>> listRecordIn(
    queryFn:(Z)->List<T>,
    keyGetter:(T)->K,
    keyIn:Z
):InQueryResult<T,K>{
    val foundList= queryFn(keyIn);
    val foundKeySet= foundList
        .map(keyGetter)
        .toSet();
    val notFoundKeySet=keyIn subtract foundKeySet;
    return InQueryResult(
        foundList= foundList,
        notFoundKeySet= notFoundKeySet
    )
}
```

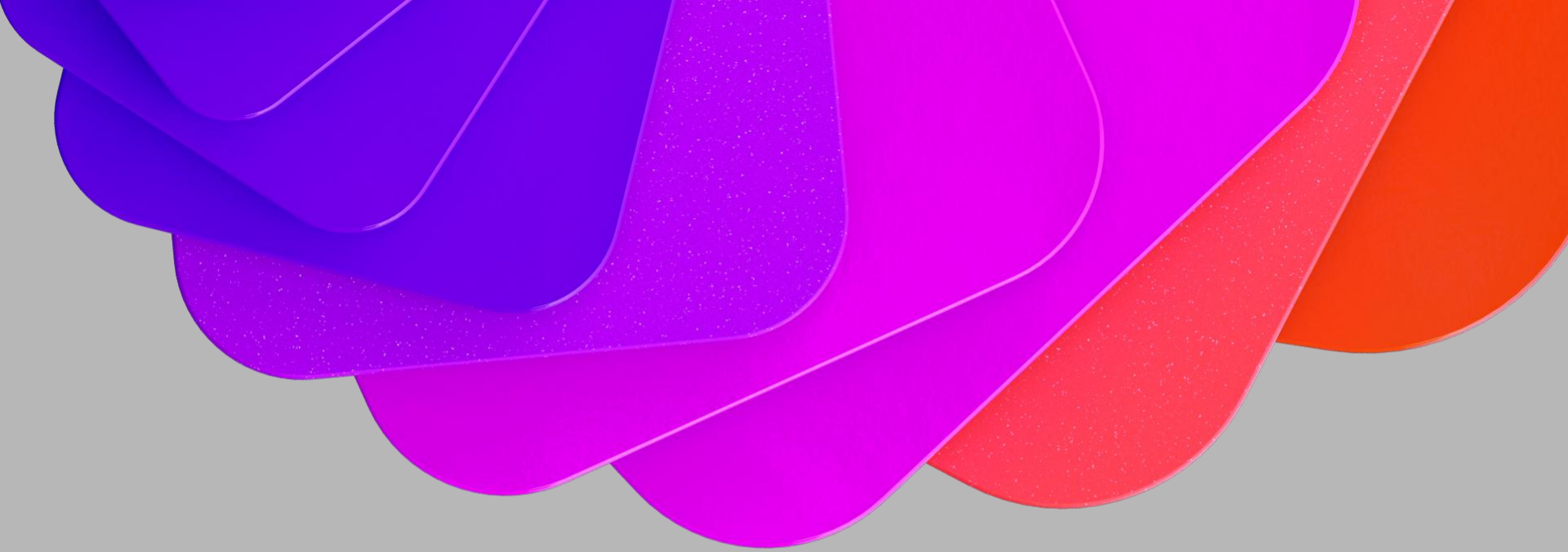
```kotlin
@Transactional(rollbackFor = [Throwable::class])
open fun listByUserCodeInOrThrow(userCodeIn: Set<String>):List<UserDTO>{
    val actualUserCodeIn= userCodeIn - setOf(USER_CODE_SUPER_ADMIN);
    val (foundDTOList,notFoundUserCodeSet)= listRecordIn(
        userMBPMapper::findAllByUserCodeIn,
        UserDTO::userCodeOf,
        actualUserCodeIn);
    if(notFoundUserCodeSet.isNotEmpty()){
        throw RuntimeException("找不到用户:${notFoundUserCodeSet}");
    }
    return foundDTOList;
}
```

```kotlin
fun printDisplayDimension(){
    val (width,height)=getDisplayDimension();
    println("width:${width} height:${height}");
}
```

# 保持专注

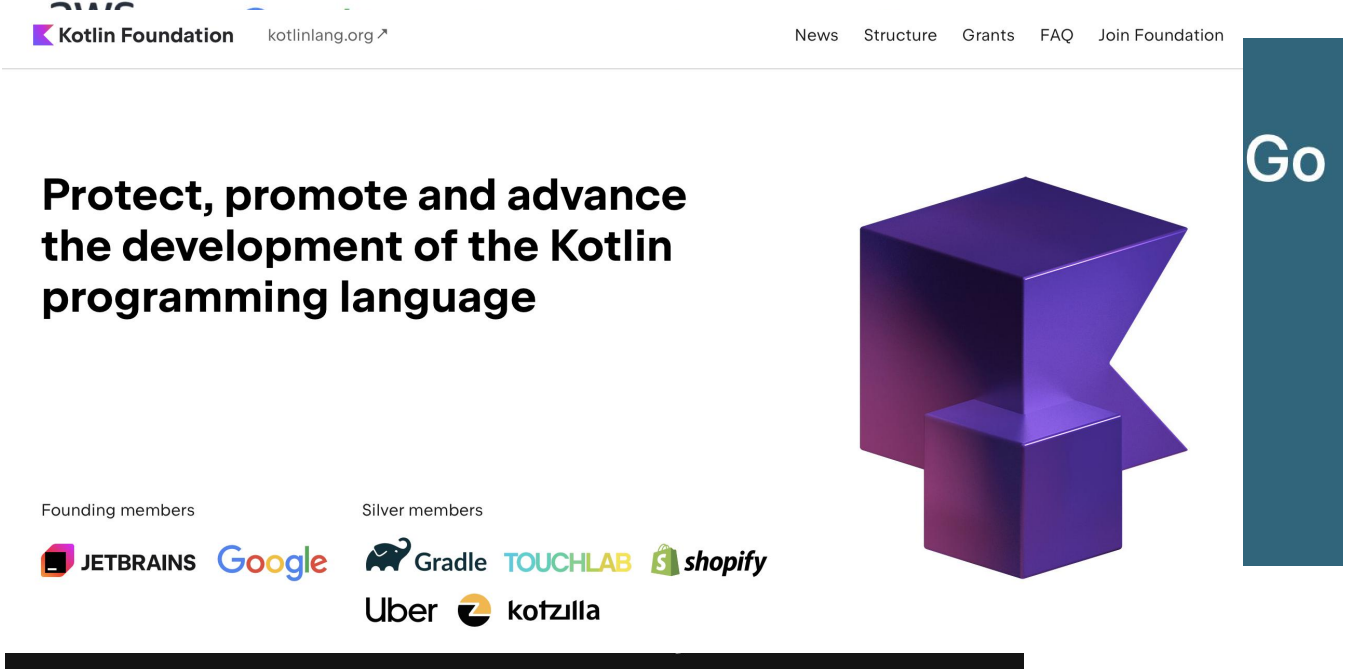

@Entity(tableN
data class Loc
@PrimaryKe
val id: Lo
val locati
val create
val isSent
)
@Database(entities = [LocationData::class], version = 1)
abstract class LocalCacheDatabase : RoomDatabase() {
companion object{ new *
private var instance:LocalCacheDatabase?=null;
fun get(context:Context):LocalCacheDatabase{ nev
if(instance==null) {
instance = Room
.databaseBuilder(
context,
LocalCacheDatabase::class.java,
name: "local_cache_db"
)
.build();
}
return instance!!;
}
}
abstract fun locationDao(): LocationDao new *
}
@Dao new *
interface Locatio
@Insert new *
suspend fun i
@Query("SELEC
suspend fun g
@Query("UPDAT
suspend fun m
}


@Entity(tableName = "location_data") new *
data class LocationData(
@PrimaryKey(autoGenerate = true)
val id: Long = 0,
val locationJson:String,
val createAt: Long,
val isSent: Boolean = false
)
@Dao new *
interface LocationDao {
@Insert new *
suspend fun insert(locationData: LocationData)
@Query("SELECT * FROM location_data WHERE isSent = 0") new *
suspend fun getUnsentLocations(): List<LocationData>
@Query("UPDATE location_data SET isSent = 1 WHERE id IN (:ids)")
suspend fun markAsSent(ids: List<Long>)
@Database(entities = [LocationData::class], version = 1) new *
abstract class LocalCacheDatabase : RoomDatabase() {
companion object{ new *
private var instance:LocalCacheDatabase?=null;
fun get(context:Context):LocalCacheDatabase{ new *
if(instance==null) {
instance = Room
.databaseBuilder(
context,
LocalCacheDatabase::class.java,
name: "local_cache_db"
)
.build();
}
return instance!!;
}
}
abstract fun locationDao(): LocationDao new *
}



# 我为什么愿意学习并将Kotlin投入生产？

# JetBrains的支持



# JetBrains的支持

Kotlin Foundation kotlinlang.org↗ News Structure Grants FAQ Join Foundation

**Protect, promote and advance the development of the Kotlin**

Android Developers > Get started > Kotlin > 指南

Android 

本页内容

为什么要优先使用 Kotlin 进

Kotlin 优先意味着什么?

我们也使用 Kotlin!

在 2019 年 Google I/O 大
是一种富有表现力且简洁
Android 应用，建议您从

为了支持使用 Kotlin 进行
译器性能和 build 速度。如需详细了解 Android 的 Kotlin 优先承诺，请参阅 Android 在 Kotlin 方面的承诺。

Kotlin

A concise multiplatform language developed

All News Releases Multiplatform

News

The New JV

```
package com.example.composepluginizedemo

import ...

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            ComposePluginizeDemoTheme {
                Scaffold(modifier = Modifier.fillMaxSiz
                    Greeting(
                        name = "PluginDemo1",
                        modifier = Modifier.padding(in
                    )
                }
            }
        }
    }
}

@Composable
fun Greeting(name: S
    Text(
```

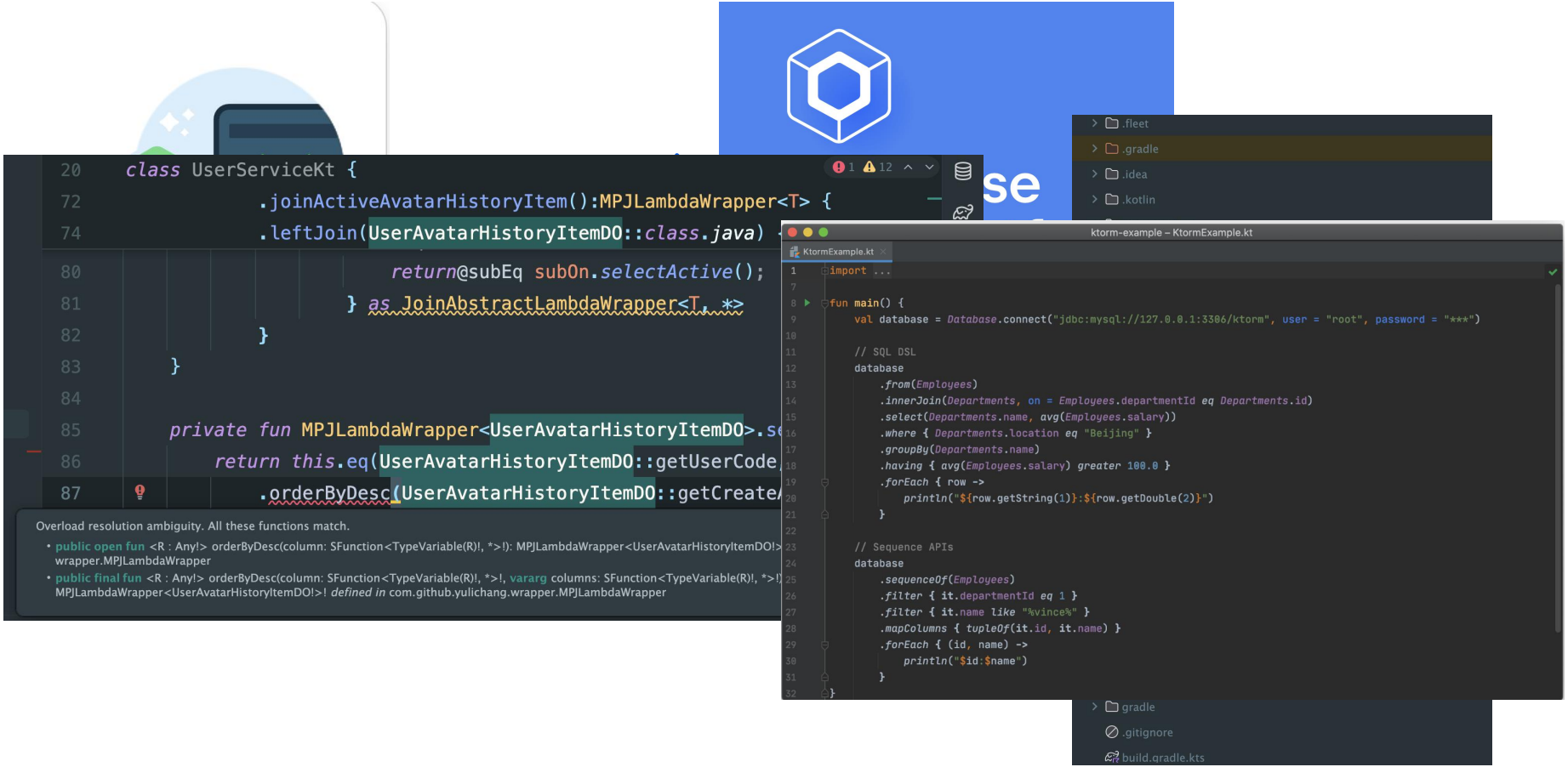
GreetingPreview

Hello PluginDemo1!

ComposePluginizeDemoTheme(Boolean, Boolean, @Composable () -> Unit) ComposePluginizeDemo.app.main 1/1

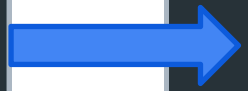
```
@Composable
fun ComposePluginizeDemoTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
```

# 社区和生态的稳步提升


class UserServiceKt {
.joinActiveAvatarHistoryItem():MPJLambdaWrapper<T> {
.leftJoin(UserAvatarHistoryItemDO::class.java)
return@subEq subOn.selectActive();
} as JoinAbstractLambdaWrapper<T, *>
}
}
private fun MPJLambdaWrapper<UserAvatarHistoryItemDO>.s
return this.eq(UserAvatarHistoryItemDO::getUserCode,
.orderByDesc(UserAvatarHistoryItemDO::getCreate
Overload resolution ambiguity. All these functions match.
public open fun <R : Any!> orderByDesc(column: SFunction<TypeVariable(R)!, *>!): MPJLambdaWrapper<UserAvatarHistoryItemDO!>
wrapper.MPJLambdaWrapper
public final fun <R : Any!> orderByDesc(column: SFunction<TypeVariable(R)!, *>!, vararg columns: SFunction<TypeVariable(R)!, *>!)
MPJLambdaWrapper<UserAvatarHistoryItemDO!>! defined in com.github.yulichang.wrapper.MPJLambdaWrapper
.fleet
.gradle
.idea
.kotlin
ktorm-example – KtormExample.kt
KtormExample.kt
import ...
fun main() {
val database = Database.connect("jdbc:mysql://127.0.0.1:3306/ktorm", user = "root", password = "***")
// SQL DSL
database
.from(Employees)
.innerJoin(Departments, on = Employees.departmentId eq Departments.id)
.select(Departments.name, avg(Employees.salary))
.where { Departments.location eq "Beijing" }
.groupBy(Departments.name)
.having { avg(Employees.salary) greater 100.0 }
.forEach { row ->
println("${row.getString(1)}:${row.getDouble(2)}")
}
// Sequence APIs
database
.sequenceOf(Employees)
.filter { it.departmentId eq 1 }
.filter { it.name like "%vince%" }
.mapColumns { tupleOf(it.id, it.name) }
.forEach { (id, name) ->
println("$id:$name")
}
}
gradle
.gitignore
build.gradle.kts

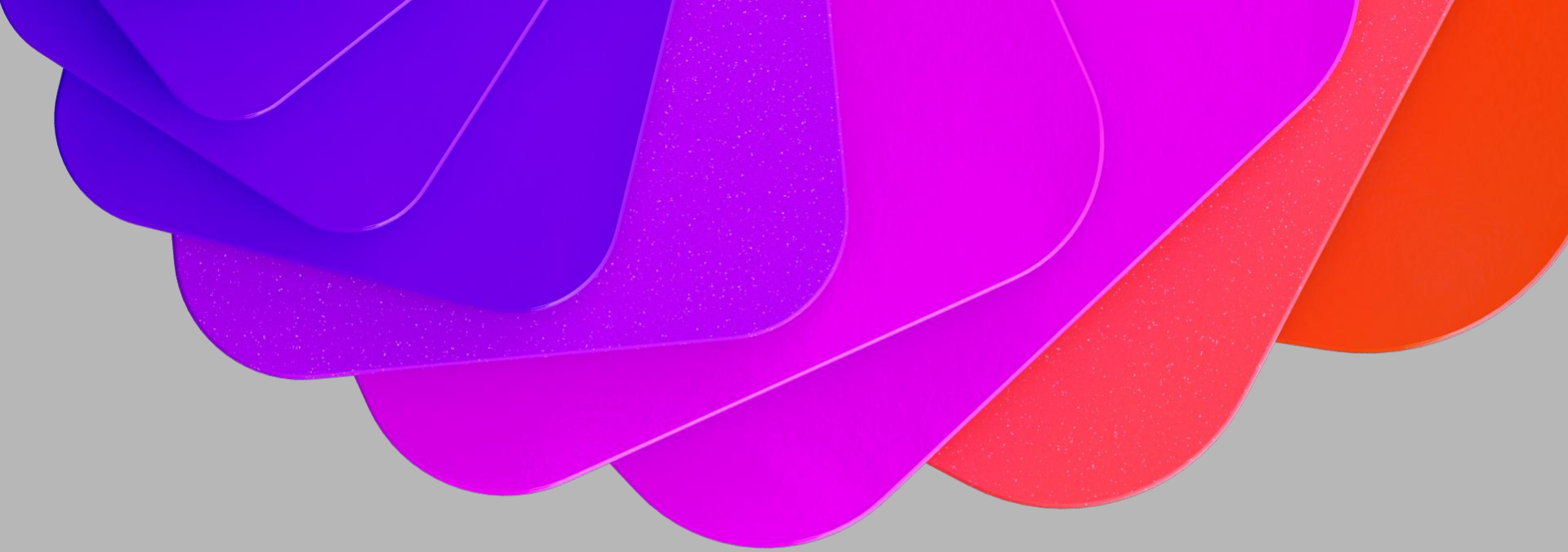
# 从css binding到统一的Modifier API

```
fun main() {
    var count: Int by mutableStateOf( value: 0)

    renderComposable(rootElementId = "root") {
        Button (){

        }
        Div({ style { padding(25.px); } }) {
            Button(attrs = {
                onClick { count -= 1 }
            }) {
                Text( value: "-1")
            }
            Span({ style { padding(15.px); } }) {
                Text( value: "$count")
            }

            Button(attrs = {
                onClick { count += 1 }
            }) {
                Text( value: "+1")
            }
        }
        NameCard( name: "any", description: "")
    }
}
```



```
@Composable
@Preview
fun App() {
    MaterialTheme {
        var showContent by remember { mutableStateOf(false) }
        Column(Modifier.fillMaxWidth(), horizontalAlignment = Alignment.CenterHorizontally) {
            Button(onClick = { showContent = !showContent }) {
                Text("Click me!")
            }
            AnimatedVisibility(showContent) {
                val greeting = remember { Greeting().greet() }
                Column(Modifier.fillMaxWidth(), horizontalAlignment = Alignment.CenterHorizontally) {
                    Image(painterResource(Res.drawable.compose_multiplatform), null)
                    Text("Compose: $greeting")
                }
            }
        }
    }
}
```

```
@OptIn(ExperimentalComposeUiApi::class)
fun main() {
    ComposeViewport(document.body!!) {
        App()
    }
}
```



# 实际后端开发中，Kotlin的优势

# 更简洁、流畅

```kotlin
fun newFindUntilDeepestByPathQueryKt(node: String,  new *
                                     property: String,
                                     nextLevelEdgeName: String,
                                     namePath: List<String>): String{
    val pathVar="path".asCypherVariable();
    @Language("Cypher")
    val queryStr="""
    WITH ${pathVar} AS path
    MATCH (n0:${node}{${property}:path[0]})
        WHERE NOT((:${node})-[:${nextLevelEdgeName}]->(n0))
    WITH path,n0,[n0] AS res
    ${
        (1 ≤ until < namePath.size)
            .map { i->
                val escapedNameItem=namePath[i]
                    .escapeForCypherQuery();
                @Language("Cypher")
                val itemQueryStr= """
                //查询路径项:\'${escapedNameItem}\'
                OPTIONAL MATCH (n${i-1})-[:${nextLevelEdgeName}]->(n$i:${node}{${property}:path[${i}]})
                WITH path , n${i} , coalesce(res+[n${i}],res) AS res
                """;
                return@map itemQueryStr;
            }
            .joinToString()
    }
    UNWIND res AS x
    WITH x
      WHERE x IS NOT NULL
    RETURN collect(x) AS res
    """
    return queryStr;
}
```

```java
public static String newFindUntilDeepestByPathQuery(  no usages  new *
        String node,
        String property,
        String nextLevelEdgeName,
        List<String> namePath
) {
    StringBuilder queryBuilder = new StringBuilder();

    queryBuilder.append("WITH $path AS path\n")
            .append("MATCH (n0:").append(node)
            .append("{").append(property).append(":path[0]})\n")
            .append("WHERE NOT((:").append(node).append(")-[:")
            .append(nextLevelEdgeName).append("]->(n0))\n")
            .append("WITH path, n0, [n0] AS res\n");

    String dynamicPart = IntStream.range(1, namePath.size()) IntStream
            .mapToObj(i -> {
                String escapedNameItem = namePath.get(i).replace( target: "\n",  replacement: "");
                return String.format("// 查询路径项: '%s'\n"
                                + "OPTIONAL MATCH (n%d)-[:%s]->(n%d:%s{%s:path[%d]})\n"
                                + "WITH path, n%d, coalesce(res + [n%d], res) AS res\n",
                        escapedNameItem, i - 1, nextLevelEdgeName, i, node, property, i, i, i);
            }) Stream<String>
            .collect(Collectors.joining());

    queryBuilder.append(dynamicPart);
    queryBuilder.append("UNWIND res AS x\n")
            .append("WITH x\n")
            .append("WHERE x IS NOT NULL\n")
            .append("RETURN collect(x) AS res\n");

    return queryBuilder.toString();
}
```

# 减少“削足适履”

```java
public String readFileJava(String fileName) throws IOException {  no usages
    StringBuilder stringBuilder=new StringBuilder();
    try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
```

```java
boolean findElementJava(List<List<Integer>> matrix, int target) {  nc
    try {
        for (int i = 0; i < matrix.size(); i++) {
            List<Integer> row = matrix.get(i);
            for (int j = 0; j < row.size(); j++) {
                if (row.get(j) == target) {
                    System.out.println("Found at " + i + ", " + j);
                    throw new FoundException(); // 通过抛出异常中断
                }
            }
        }
    } catch (FoundException e) {
        return true; // 处理异常并返回结果
    }
    return false;
}

static class FoundException extends RuntimeException {}  2 usages
```

```kotlin
fun readFileKotlin(fileName: String): String{
    val file= File(fileName);
    return try{
        file.bufferedReader()
            .use {
                return@use it.readText();
            }
    }catch (e:IOException){
        throw IOException("按行读取文件:${fileName}出错",e);
    }
}
```

```kotlin
fun findElementKotlin(matrix: List<List<Int>>, target: Int): Boolean {
    matrix.forEachIndexed { rowIndex, row ->
        row.forEachIndexed { _, value ->
            if (value == target) {
                println("Found at `$rowIndex, $`colIndex")
                return true // 找到目标后提前返回
            }
        }
    }
    return false
}
```

# 在优化时保持简洁

```
fun processUserDataSync(userCodeIn: Set<String>):List<UserDTO>{
    val userDOList= listUserByUserCodeInSync(userCodeIn);
    return userDOList.map { userDO->
        val tagItemDOList =
            listUserTagItemByUserCodeSync(userDO.userCode);
        return@map UserDTO().apply {
            this.userDO = userDO;
            this.tagDOList = tagItemDOList;
        }
    }
}
```

```
fun processUserData(userCodeIn: Set<String>):List<UserDTO> = runBlocking {
    val userDOList = listUserByUserCodeIn(userCodeIn)
    return@runBlocking userDOList
        .map { userDO ->
            async {
                val tagItemDOList =
                    listUserTagItemByUserCode(userDO.userCode);
                return@async UserDTO().apply {
                    this.userDO = userDO;
                    this.tagDOList = tagItemDOList;
                }
            }
        }
        .awaitAll();
}
```

# 在优化时保持简洁


private final ExecutorService e
Executors.newFixedThre
List<CompletableFuture<UserDTO>> futures = userDOList.parallelStream() Stream<UserDO>
.map(userDO -> listUserTagItemByUserCode(userDO.getUserCode())
thenApply(tagItemDOList -> {
public CompletableFuture
return CompletableFu
List<UserDO> fou
Set<String> foun
.map(Use
.collect
Set<String> notF
.filter(c
.collect
if (!notFoundUse
public CompletableFu
return Completab
LambdaQueryW
queryWrapper
return tagIt
}, executor);
}
public CompletableFuture<List<UserDTO>> processUserData(Set<String> userCodeIn) { no usages
return listUserByUserCodeIn(userCodeIn).thenCompose(userDOList -> {
List<CompletableFuture<UserDTO>> futures = userDOList.parallelStream() Stream<UserDO>
.map(userDO -> listUserTagItemByUserCode(userDO.getUserCode())
.thenApply(tagItemDOList -> {
UserDTO userDTO = new UserDTO();
userDTO.setUserDO(userDO);
userDTO.setTagDOList(tagItemDOList);
return userDTO;
})) Stream<CompletableFuture<...>>
.toList();
CompletableFuture<Void> allOf = CompletableFuture.allOf(futures.toArray(new CompletableFuture[0]));
return allOf.thenApply(v -> futures.parallelStream() Stream<CompletableFuture<...>>
.map(CompletableFuture::join) Stream<UserDTO>
.toList());
});
}
ableFuture[0]));

# 与java互操作


@Entity 17 usages
@Table(name = UserDO.TABLE_NAME)
@TableName(UserDO.TABLE_NA
@Data
publi
p
@
@
@
p
@
@
p
}
package com.example.kotl
import com.example.kotli
import com.example.kotli
import com.example.kotli
import com.github.yulich
interface UserMBPMapper:
MPJBaseMapper<UserDO
interface UserTagItemMBP
MPJBaseMapper<UserTa
interface UserAvatarHist
MPJBaseMapper<UserTa
@Data 13 usages
public class UserDTO {
open fun getUserProfile(userCode: String): UserProfileVO {
return JoinWrappers
.lambda(UserDO::class.java)
.selectAssociation(UserDO::class.java, UserDTO::userDO)
.selectCollection<UserDTO, UserTagItemDO, UserTagItemDO, List<UserTagItemDO>>(
UserTagItemDO::class.java, UserDTO::tagDOList
)
.selectAssociation(UserAvatarHistoryItemDO::class.java, UserDTO::activeAvatarHistoryItemDO)
.optionalJoinTagAndAvatarHistory(
includeTag= true,
includeAvatarHistory = true
)
.eq(UserDO::getUserCode, userCode)
.let {
return@let userMBPMapper.selectJoinOne(UserDTO::class.java, it);
}
?.wrapToVO()
?:throw RuntimeException("未找到对应的用户:${userCode}")
}

# Thanks!
# Have a
# Nice Kotlin

Kotlin

@AkaAny

# 问答环节

Kotlin 中文开发者大会