DSL

~ dsl_slides
* about::contacts(me)

Aliaksandr Salnikau (Александр Сальников)

Company: EPAM

Position: Lead Software Engineer

Primary skill: Android

Contacts:

- email: aliaksandr_salnikau@epam.com
- skype: alexandersalnikov

Agenda

- About my current project
- Project challenges
- What is DSL? Is it a solution?
- DSL in Kotlin
- Kotlin features
- Example from my project

Circuit by Unify

Features:

- Private Conversations
- Commenting and reply
- Flagging
- Audio and video calling
- 1-2-1 and group calling
- Joining active calls
- Active speaker dynamic calls stage
- Mute
- Moving calls between Circuit devices
- Access to conference details
- PSTN dial in
- Search
- much more...

Circuit by Unify

- Android KitKat+
- Java 6
- Retrolambda
- RxJava
- Kotlin
- MVP
- Spek

Project challenges

Circuit by Unify

- Data classes often contains 700-1000 lines of code.
- Data classes generates from .proto files
- Complicated relations between classes
- Generating data for tests is hard

```
val messageItem = MessageItem(
        convId = "convid",
        title = "title",
        avatar = Avatar.EMPTY,
        creationTime = 42,
        muted = false,
        shortDescription = "",
        modificationTime = 42,
        unreadItemCount = 0,
        leftDescriptionDrawable = 0,
        rightDescriptionDrawable = 0,
        isFailed = false,
        convType = Conversation.ConversationType.DIRECT,
        favourited = false,
        isUserParticipant = true
)
```

```
val usersFirstNames = "User1, User2, User3, User4"
val conversationTopic = "Very interesting conversation"

it("Empty topic") {
    val titleGroupEmptyTopic = Conversation().apply {
        topic = ""
        participantFirstNames = usersFirstNames
        type = Conversation.ConversationType.GROUP
        participants = listOf(Participant())
    }.getTitle(mockedContext)

    Assert.assertEquals(EMPTY_GROUP_CONVERSATION,
titleGroupEmptyTopic)
}
```

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(context);
builder
        .setSmallIcon(R.drawable.icon_notification_circuit_logo_small)
        .setColor(ContextCompat.getColor(context, R.color.green))
        .setLargeIcon(largeIcon)
        .setContentTitle(userName)
        .setContentText(R.string.res_IsNowAvailable), userName)
        .setCategory(NotificationCompat.CATEGORY_SOCIAL)
        .setDefaults(NotificationCompat.DEFAULT_VIBRATE)
        .setAutoCancel(true)
        .setOnlyAlertOnce(true)
        .setContentIntent(contentIntent);
```

Fast@Furious Code Review

```
~ dsl_slides
* about::contacts(me)
```

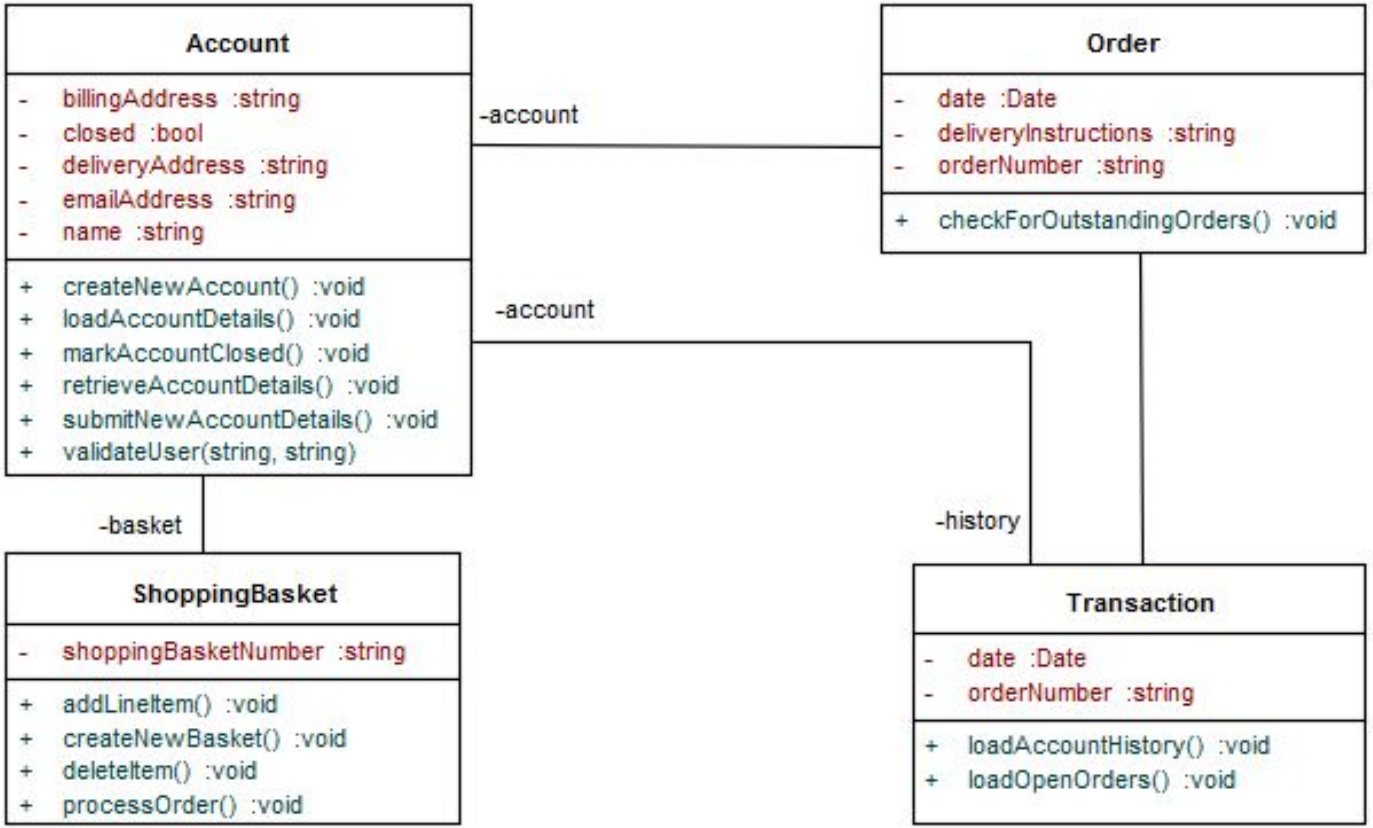
git commit -m ‘~ dsl_slides’

::	Подсистема
=	Создал новое. Аналогия: int a = 30;
+	Добавил метод/функционал
-	Удалил (устарело к примеру)
~	Изменил оставив совместимым(малый рефакторинг)
*	BugFix

What is DSL?

- Original Language (UML, Cucumber)
- Target Language (Gradle build script)

UML


Account
- billingAddress :string
- closed :bool
- deliveryAddress :string
- emailAddress :string
- name :string
+ createNewAccount() :void
+ loadAccountDetails() :void
+ markAccountClosed() :void
+ retrieveAccountDetails() :void
+ submitNewAccountDetails() :void
+ validateUser(string, string)
-account
Order
- date :Date
- deliveryInstructions :string
- orderNumber :string
+ checkForOutstandingOrders() :void
-account
-basket
-history
ShoppingBasket
- shoppingBasketNumber :string
+ addLineItem() :void
+ createNewBasket() :void
+ deleteItem() :void
+ processOrder() :void
Transaction
- date :Date
- orderNumber :string
+ loadAccountHistory() :void
+ loadOpenOrders() :void

Cucumber

```
# feature/hello_cucumber.feature
Feature: Hello Cucumber
As a product manager
I want our users to be greeted when they visit our site
So that they have a better experience

Scenario: User sees the welcome message
When I go to the homepage
Then I should see the welcome message
```

https://cucumber.io/

Gradle build script

```
android {
    compileSdkVersion 25
    buildToolsVersion "25.0.3"
    defaultConfig {
        applicationId "by.bkug.app"
        minSdkVersion 21
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

DSL in Kotlin

Anko

https://github.com/Kotlin/anko

```xml
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:padding="64dp">

  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="32dp"
    android:text="@string/login_form"
    android:textSize="32sp"/>

  <EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email"
    android:inputType="textEmailAddress"/>

  <EditText
    android:id="@+id/password"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/password"
    android:inputType="textPassword"/>
</LinearLayout>
```

```
verticalLayout {
    padding = dip(64)

    textView(R.string.login_form) {
        textSize = sp(32).toFloat()
    }.lparams {
        width = MATCH_PARENT
        bottomMargin = dip(16)
    }

    editText {
        hintResource = R.string.email
    }

    editText {
        hintResource = R.string.password
    }
}
```


Kotlinx

Kotlinx.html

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        h1 { +"XML encoding with Kotlin" }
        p { +"this format can be used as an alternative markup to XML" }
        // an element with attributes and text content
        a(href = "http://kotlinlang.org") { +"Kotlin" }
        p { +"some text" }

        // content generated by
        p {
            for (arg in args)
                +arg
        }
    }
}
```

https://github.com/Kotlin/kotlinx.html

Kotlinx.coroutines

```
fun main(args: Array<String>) {
    launch(CommonPool) {
        delay(1000L)
        println("World!")
    }
    println("Hello,")
    Thread.sleep(2000L)
}
```

https://github.com/Kotlin/kotlinx.coroutines

Kotlinx.*

Potential candidates

- kotlinx.*http*
- kotlinx.*logging*
- kotlinx.*json*
- kotlinx.*xml*
- kotlinx.*compression*
- kotlinx.*calendar*
- kotlinx.*drawing*

Gradle Script Kotlin

```
apply {
    plugin("com.android.application")
    plugin("kotlin-android")
}

android {
    buildToolsVersion("25.0.0")
    compileSdkVersion(23)

    defaultConfig {
        minSdkVersion(15)
        targetSdkVersion(23)

        applicationId = "com.example.kotlingradle"
        versionCode = 1
        versionName = "1.0"
    }
}
```

Teamcity configuration DSL

```
object Project : Project({
    uuid = "8efe17c7-00c3-4f37-b304-71a46c42f7e4"
    extId = "GitExtensions"
    parentId = "_Root"
    name = "Git Extensions"
    description = "https://github.com/gitextensions/gitextensions"

    vcsRoot(GitExtensions_HttpsGithubComJetbrainsGitextensions)

    buildType(GitExtensions_Main)
})
```

https://blog.jetbrains.com/teamcity/2016/11/kotlin-configuration-scripts-an-introduction/

Spek

```
describe("playing rock paper scissor") {
    context("when player one plays rock") {
        beforeEachTest {
            game.recordMove(player = Player.ONE, move = Move.ROCK)
        }
        context("and player two plays scissors") {
            it("declares player one the winner") {
                assertEquals("Player One wins!", game.getResult())
            }
        }
...
```

http://spekframework.org/

Kotlin Features

Lambda expressions

```
view.setOnClickListener(object: View.OnClickListener {
    override fun onClick(v: View?) {
        // do something
    }
})
```

Lambda expressions

```
view.setOnClickListener { /* do something */ }
```

Extension functions

```
fun View.visible() {
    visibility = View.VISIBLE
}
```

Extension properties

```
val ViewGroup.children: List<View>
    get() = (0..childCount -1).map { getChildAt(it) }
```

Infix functions

```
val Aliaksandr = User()
val Dzmitry = User()

Aliaksandr write "Hello, ${Dzmitry.firstName}!"
```

Infix functions

```
infix fun User.write(topic: String): ConversationItem {
    return ConversationItem().apply {
        this.title = topic
    }
}
```

Higher-Order Functions

```
fun toBeSynchronized() = sharedResource.operation()

val result = lock(lock, ::toBeSynchronized)
```

Higher-Order Functions

```
fun <T> lock(lock: Lock, body: () -> T): T {
    lock.lock()
    try {
        return body()
    }
    finally {
        lock.unlock()
    }
}
```

Operator overloading

```
val dreamTeam: List<User> = Aliaksandr + Dzmitry + Hanna
```

Operator overloading

```
operator fun User.plus(user: User): List<User> {
    return listOf(this, user)
}
```

Operator overloading

Expression	Translated to
`a + b`	`a.plus(b)`
`a - b`	`a.minus(b)`
`a * b`	`a.times(b)`
`a / b`	`a.div(b)`
`a..b`	`a.rangeTo(b)`
`much more...`	

http://kotlinlang.org/docs/reference/operator-overloading.html

Function Literals with Receiver

```
User.() -> Unit
```

Function Literals with Receiver

```
fun Int.(other: Int): Int = this + other
```

Lambda with Receiver

```
fun User.default(block: User.() -> Unit): User {
    userId = UUID.randomUUID().toString()
    emailAddress = "$firstName.$lastName@gmail.com"
    block()
    return this
}
```

Lambda with Receiver

```
val Aliaksandr = User().default {
    firstName = "Aliaksandr"
    lastName = "Salnikau"
}
```

Default Arguments

```
fun a(href: String = "", init: A.() -> Unit) {
    val a = initTag(A(), init)
    a.href = href
}
```

Default Arguments

```
a(href = "http://kotlinlang.org") { +"Kotlin" }

a { +"Kotlin" }
```

Type-Safe Builders

```
html {
    head {
        title { +"XML encoding with Kotlin" }
    }
    body {
        h1 { +"XML encoding with Kotlin" }
        p { +”Some text here" }
    }
}
```

Show me the code

Conclusions

- DSLs in Kotlin are easy to build
- DSLs in Kotlin work ideally as configuration API
- They can work as a powerful abstraction
- Custom DSL is not a silver bullet

Some links

- Cucumber - https://cucumber.io/
- Kotlinx.dom - https://github.com/Kotlin/kotlinx.dom
- Kotlin XML data binding DLS - https://github.com/jonnyzzz/kotlin.xml.bind
- TeamCity2DSL - https://github.com/jonnyzzz/TeamCity2DSL
- Kotlin links - https://kotlin.link/?q=dsl
- Систематизация коммитов - https://habrahabr.ru/post/157317/