









Для пр 5 на android (12)

[15:03] Глускер Александр Игоревич

package com.example.myapplication  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.activity.viewModels  
import androidx.compose.foundation.layout.Column  
import androidx.compose.foundation.layout.Row  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.widthIn  
import androidx.compose.foundation.lazy.LazyColumn  
import androidx.compose.foundation.lazy.items  
import androidx.compose.material.icons.Icons  
import androidx.compose.material.icons.filled.Delete  
import androidx.compose.material.icons.filled.Edit  
import androidx.compose.material3.Button  
import androidx.compose.material3.ExperimentalMaterial3Api  
import androidx.compose.material3.Icon  
import androidx.compose.material3.IconButton  
import androidx.compose.material3.MaterialTheme  
import androidx.compose.material3.Surface  
import androidx.compose.material3.Text  
import androidx.compose.material3.TextField  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.collectAsState  
import androidx.compose.runtime.getValue  
import androidx.compose.runtime.mutableStateOf  
import androidx.compose.runtime.saveable.rememberSaveable  
import androidx.compose.runtime.setValue  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.platform.testTag  
import androidx.compose.ui.res.stringResource  
import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
import androidx.lifecycle.ViewModel  
import androidx.navigation.NavHostController  
import androidx.navigation.NavType  
import androidx.navigation.compose.NavHost  
import androidx.navigation.compose.composable  
import androidx.navigation.compose.rememberNavController  
import androidx.navigation.navArgument  
import com.example.myapplication.ui.theme.MyApplicationTheme  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.flow.MutableStateFlow  
import kotlinx.coroutines.flow.asStateFlow  
import kotlinx.coroutines.launch  
  
class ListModel {  
 private val list = mutableListOf<Pair<Int,String>>()  
  
 private val scope = CoroutineScope(Dispatchers.Default)  
  
 private var id = 0  
  
 init {  
 (1..100).forEach {  
 list.add((++id) to it.toString())  
 }  
 }  
  
 fun add(s: String) {  
 list.add((++id) to s)  
 notifyFlow()  
 }  
  
 private fun notifyFlow() {  
 scope.launch {  
 mutableFlow.emit(list.toList())  
 }  
 }  
  
 fun drop(id: Int) {  
 list.removeIf {it.first == id}  
 notifyFlow()  
 }  
  
 fun change(id: Int,s: String) {  
 list.forEachIndexed { index, pair ->  
 if (pair.first == id) {  
 list[index] = id to s  
 }  
 }  
 notifyFlow()  
 }  
  
 private val mutableFlow = MutableStateFlow(list.toList())  
  
 val flow = mutableFlow.asStateFlow()  
  
  
}  
  
class ViewModelList: ViewModel() {  
  
 private val model = ListModel()  
  
 fun add(s: String) = model.add(s)  
  
 fun drop(id: Int) = model.drop(id)  
  
 fun change(id: Int,s: String) = model.change(id,s)  
  
 val flow = model.flow  
  
}  
  
  
class MainActivity : ComponentActivity() {  
  
 private val viewModel:ViewModelList by viewModels()  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContent {  
 MyApplicationTheme {  
 // A surface container using the 'background' color from the theme  
 Surface(  
 modifier = Modifier.fillMaxSize(),  
 color = MaterialTheme.colorScheme.background  
 ) {  
 ExampleList(viewModel)  
 }  
 }  
 }  
 }  
  
  
 companion object {  
 const val number1\_tag = "NUMBER1"  
 const val number2\_tag = "NUMBER2"  
 const val button = "BUTTON"  
 const val result = "RESULT"  
 }  
}  
  
@Composable  
fun ShowList(viewModel: ViewModelList, navController: NavHostController) {  
  
 val listState = viewModel.flow.collectAsState()  
  
 Column {  
 Button(onClick = {  
 navController.navigate("add")  
 }) {  
 Text("Добавить")  
 }  
  
 LazyColumn {  
  
 items(listState.value,  
 key = { it.first }  
 ) {  
 Row {  
 Text(it.second, modifier = Modifier.widthIn(max=300.dp))  
  
 IconButton(onClick = {  
 navController.navigate("edit/${it.first}")  
 }) {  
 Icon(Icons.Default.Edit, "Редактировать")  
 }  
  
 IconButton(onClick = {  
 viewModel.drop(it.first)  
 }) {  
 Icon(Icons.Default.Delete, "Удалить")  
 }  
  
 }  
 }  
 }  
 }  
  
}  
  
@Composable  
private fun ExampleList(viewModel: ViewModelList) {  
  
  
 //режим списка  
 //режим добавления  
 //режим изменения  
  
 //navigation  
  
 //контроллер  
 //хост  
  
 val navController = rememberNavController()  
 NavHost(navController = navController,  
 startDestination = "list"  
 ) {  
  
 composable("list") {  
 ShowList(viewModel,navController)  
 }  
 composable("add") {  
 AddToList(viewModel,navController)  
 }  
 composable("edit/{id}",  
 arguments = listOf(  
 navArgument("id") {  
 type = NavType.IntType  
 }  
 )  
 ) {  
  
 Edit(it.arguments!!.getInt("id"), viewModel,navController)  
 }  
  
 }  
  
}  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun Edit(id: Int, viewModel: ViewModelList, navController: NavHostController) {  
  
 val list = viewModel.flow.collectAsState()  
  
  
 Column {  
 var s by rememberSaveable {  
 mutableStateOf(list.value.first {it.first == id}.second)  
 }  
  
 TextField(value = s, onValueChange = {s=it}, label = {Text("Строка")})  
  
 Button(onClick={  
 viewModel.change(id,s)  
 navController.popBackStack()  
 // navController.navigate("list")  
 }) {Text("Изменить")}  
  
 }  
  
  
}  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun AddToList(viewModel: ViewModelList, navController: NavHostController) {  
  
 Column {  
 var s by rememberSaveable {  
 mutableStateOf("")  
 }  
  
 TextField(value = s, onValueChange = {s=it}, label = {Text("Строка")})  
  
 Button(onClick={  
 viewModel.add(s)  
 navController.popBackStack()  
 // navController.navigate("list")  
 }) {Text("Добавить")}  
  
 }  
  
}  
  
  
  
  
@Composable  
fun ShowAnswer(num: Int?,modifier: Modifier = Modifier) {  
 Text("Ответ: "+num.toString(),  
 modifier = modifier.testTag(MainActivity.result))  
}  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun EnterNumber(  
 numberChanged:(Int?)->Unit,  
 tag: String,  
 label: String,  
 modifier: Modifier= Modifier) {  
 val number= rememberSaveable { mutableStateOf("") }  
 TextField(number.value, {  
 number.value=if (it.toIntOrNull()!=null||it=="-") {  
 numberChanged(it.toIntOrNull())  
 it  
 }  
 else number.value}, label = {  
 Text(label)  
 },modifier = modifier.testTag(tag)  
 )  
  
}  
  
  
@Composable  
fun Add(viewModel: AddViewModel,modifier: Modifier = Modifier) {  
 Column(modifier) {  
  
 var num1 by rememberSaveable { mutableStateOf<Int?>(null) }  
 var num2 by rememberSaveable { mutableStateOf<Int?>(null) }  
  
 EnterNumber({num1 = it},MainActivity.number1\_tag,stringResource(R.string.number1))  
 EnterNumber({num2 = it},MainActivity.number2\_tag,stringResource(R.string.number2))  
  
 Button(onClick = {  
 val n1 = num1  
 val n2 = num2  
 if (n1!=null&&n2!=null) {  
 viewModel.add(n1,n2)  
 }  
 },modifier = Modifier.testTag(MainActivity.button)  
 ) {  
 Text("Сложить")  
  
 }  
  
 val result = viewModel.resultStateFlow.collectAsState()  
 if (result.value!=null) {  
 ShowAnswer(result.value)  
 }  
 }  
  
  
  
  
}  
  
@Composable  
fun Greeting(name: String, modifier: Modifier = Modifier) {  
 Text(  
 text = "Hello $name!",  
 modifier = modifier  
 )  
}  
  
  
//@Preview(showBackground = true)  
//@Composable  
//fun AddPreview() {  
// MyApplicationTheme {  
// Add()  
// }  
//}  
  
@Preview(showBackground = true)  
@Composable  
fun GreetingPreview() {  
 MyApplicationTheme {  
 Greeting("Android")  
 }  
}  
  
class AddModel {  
 fun add(x:Int,y: Int) = x+y  
}  
  
class AddViewModel private constructor() {  
  
  
  
 private val resultMutableStateFlow = MutableStateFlow<Int?>(null)  
 val resultStateFlow = resultMutableStateFlow.asStateFlow()  
  
 private val scope = CoroutineScope(Dispatchers.Default)  
  
 fun add(x:Int,y:Int) {  
 val res = AddModel().add(x,y)  
 scope.launch {  
 resultMutableStateFlow.emit(res)  
 }  
 }  
  
 companion object {  
 private var instance:AddViewModel? = null  
  
 fun getInstance():AddViewModel {  
 if (instance==null) instance = AddViewModel()  
 return instance!!  
 }  
 }  
}

[15:04] Глускер Александр Игоревич

plugins {  
 id("com.android.application")  
 id("org.jetbrains.kotlin.android")  
}  
  
android {  
 namespace = "com.example.myapplication"  
 compileSdk = 34  
  
 defaultConfig {  
 applicationId = "com.example.myapplication"  
 minSdk = 24  
 targetSdk = 34  
 versionCode = 1  
 versionName = "1.0"  
  
 testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
 vectorDrawables {  
 useSupportLibrary = true  
 }  
 }  
  
 buildTypes {  
 release {  
 isMinifyEnabled = false  
 proguardFiles(  
 getDefaultProguardFile("proguard-android-optimize.txt"),  
 "proguard-rules.pro"  
 )  
 }  
 }  
 compileOptions {  
 sourceCompatibility = JavaVersion.VERSION\_1\_8  
 targetCompatibility = JavaVersion.VERSION\_1\_8  
 }  
 kotlinOptions {  
 jvmTarget = "1.8"  
 }  
 buildFeatures {  
 compose = true  
 }  
 composeOptions {  
 kotlinCompilerExtensionVersion = "1.4.3"  
 }  
 packaging {  
 resources {  
 excludes += "/META-INF/{AL2.0,LGPL2.1}"  
 }  
 }  
}  
  
dependencies {  
  
 implementation("androidx.core:core-ktx:1.12.0")  
 implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.2")  
 implementation("androidx.activity:activity-compose:1.8.0")  
 implementation(platform("androidx.compose:compose-bom:2023.03.00"))  
 implementation("androidx.compose.ui:ui")  
 implementation("androidx.compose.ui:ui-graphics")  
 implementation("androidx.compose.ui:ui-tooling-preview")  
 implementation("androidx.compose.material3:material3")  
 testImplementation("junit:junit:4.13.2")  
 androidTestImplementation("androidx.test.ext:junit:1.1.5")  
 androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")  
 androidTestImplementation(platform("androidx.compose:compose-bom:2023.03.00"))  
 androidTestImplementation("androidx.compose.ui:ui-test-junit4")  
 debugImplementation("androidx.compose.ui:ui-tooling")  
 debugImplementation("androidx.compose.ui:ui-test-manifest")  
  
 implementation("androidx.navigation:navigation-compose:2.7.4")  
}