
FINANTEQ®

FINANTEQ®

Synchroniczny Android ?

Autor: Kamil Kalisz

Porozmawiajmy

- Co to jest asynchroniczność
- Dlaczego mam z nią problem
- Czy da się coś zrobić w kontekście androida

Synchroniczność vs Asynchroniczność

```
String name = JOptionPane.showInputDialog(frame,"Enter your name");
// do something
```

```
JOptionPane.showInputDialog(frame,"Enter your name„,new InputListener(){
      void onInput(String input ){
      // do something
})
```

Główne Problemy podczas wytwarzania oprogramowania

- Cykl życia aktywności
- Cykl życia fragmentu
- Zdarzenia
- Dialogi

Rezultaty aktywności ?

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data){
    if (resultCode == eActivityResult.ACTIVITY_CLOSE_RESULT_IN_INTENT){
        String result = data.getStringExtra(eActivityResult.ACTIVITY_CLOSE_RESULT);
        if (result == null){
            return;
        }
        if (result.compareTo(eActivityResult.ACTIVITY_CLOSE_RESULT_OK) == 0) {...}
        else if (result.compareTo(eActivityResult.ACTIVITY_CLOSE_RESULT_ABORT) == 0){...}
        else if (result.compareTo(eActivityResult.ACTIVITY_CLOSE_RESULT_CANCEL) == 0){
            //DO NOTHING
        }
    }
    else if(requestCode == ACCOUNT_LIST){
        if(resultCode == AccountList.ACCOUNT_CHOSEN){
            Account accout = data.getParcelableExtra(AccountList.ACCOUNT_ID)
            onAccountChoose(accout);
        }
        else{ onCancelAccountChose();}
    }
    else if( requestCode == PICK_RECIPIENT_DATA){
        if(resultCode == RecipientList.RECIPEINT_CHHOSEN){
            if(data.hasExtra(RecipientList.RECIPIENT_ACCOUNT){...}
        }
        else if(data.hasExtra(....){..}
    }
    else super.onActivityResult(requestCode, resultCode, data);
}
```

Idealna koncepcja

- Prosta obsługa cyklu życia
- Łatwiejsza integracja zdarzeń
- Synchroniczność

Zarządcy – funkcje zwrotne

Zalety:

- Łatwiejsza obsługa konkretnych pojedynczych zdarzeń
- Możliwość odseparowanie części logiki
- Większa elastyczność

Wady:

- Powielany kod
- Konieczność zapisywanie się na wszystkie zdarzenia
- Skomplikowanie kodu rośnie przy dużej ilości logiki

Zarządy

```
public interface WindowManager
{
    void registerIntentHandler(@NonNull String key, @NonNull IntentHandler intentHandler);

    void startActivity(@NonNull String key, @NonNull Intent intent);
}
```

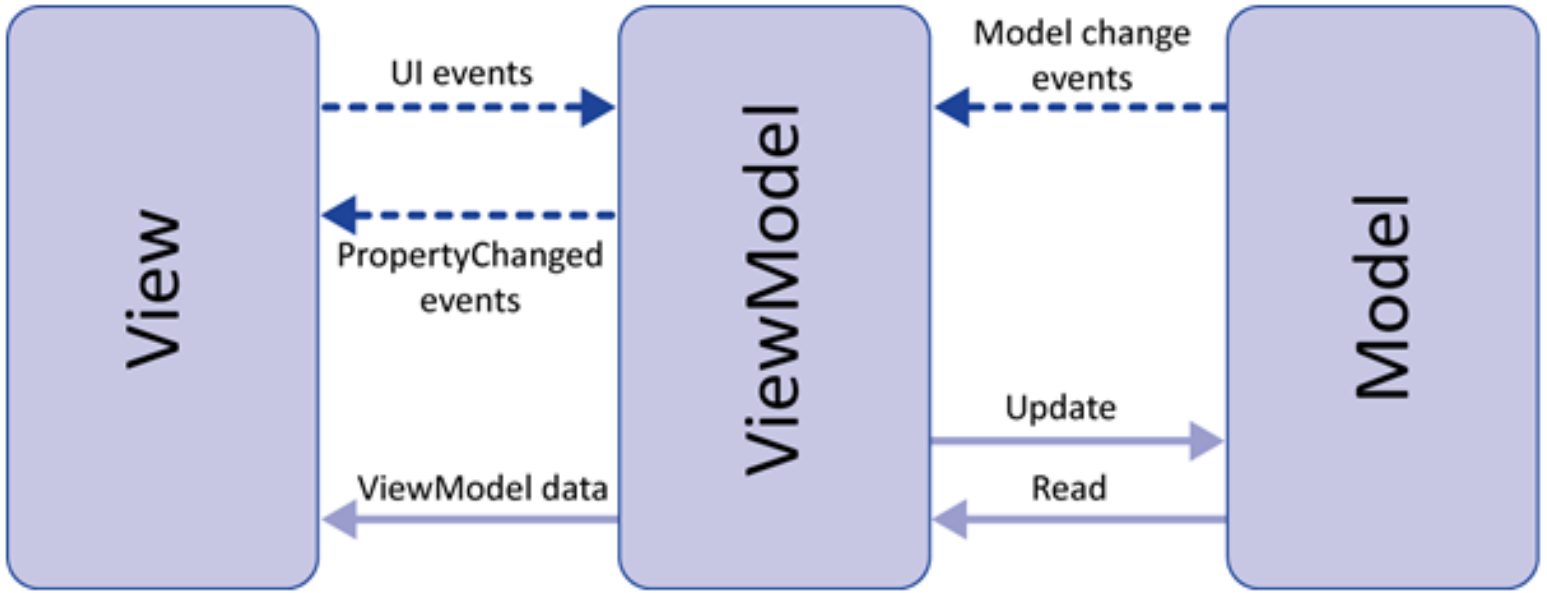
```
public interface  IntentHandler {

    void onWindowResult(String requestCode, int resultCode, Intent resultData);
}
```

Pomysł ?

- LifeCycle
- ViewModel
- Kotlin
- Coroutines

ViewModel, LiveData, Lifecycle

- Komponenty architektury
- Wzorzec MVVM
- Obsługa zmiany konfiguracji

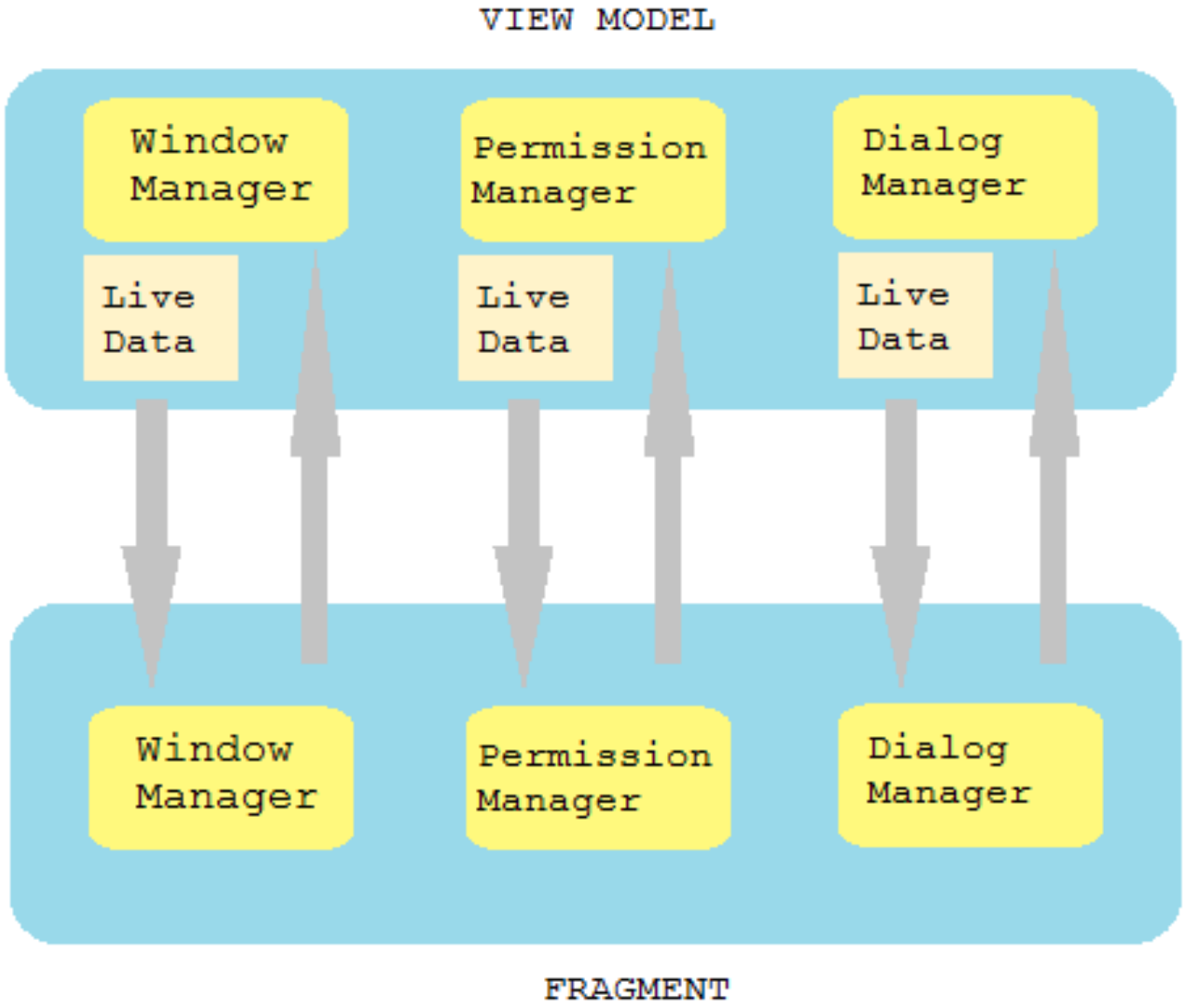


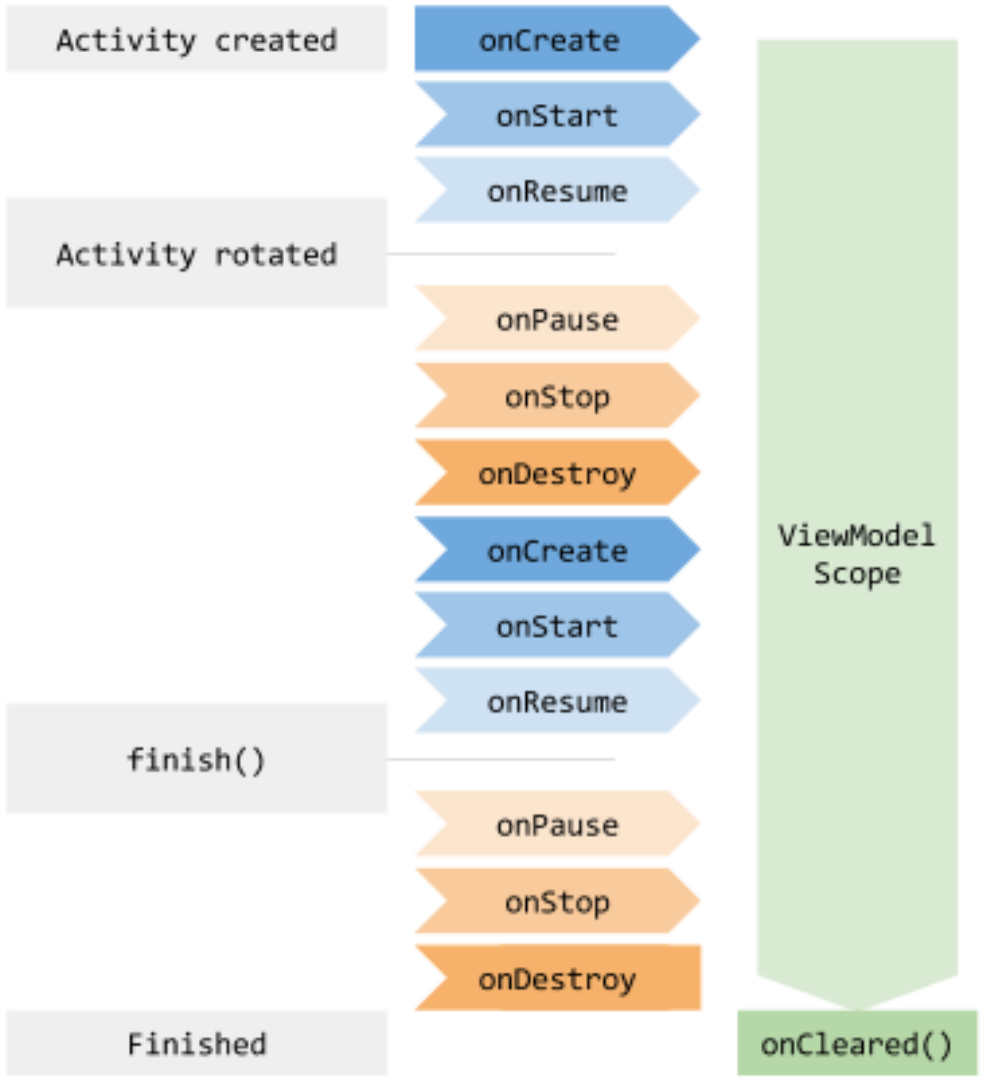
ViewModel

```
public class NameViewModel extends ViewModel {

    // Create a LiveData with a String
    private MutableLiveData<String> mCurrentName;

    public MutableLiveData<String> getCurrentName() {
        if (mCurrentName == null) {
            mCurrentName = new MutableLiveData<String>();
        }
        return mCurrentName;
    }

// Rest of the ViewModel...
}
```

ViewModel

```
// Get the ViewModel.
mModel = ViewModelProviders.of(this).get(NameViewModel.class);

// Create the observer which updates the UI.
final Observer<String> nameObserver = new Observer<String>() {
      @Override
      public void onChanged(@Nullable final String newName) {
           // Update the UI, in this case, a TextView.
            mNameTextView.setText(newName);
      }
 };

// Observe the LiveData, passing in this activity as the LifecycleOwner and the observer.
 mModel.getCurrentName().observe(this, nameObserver);
```


VIEW MODEL
Window Manager
Permission Manager
Dialog Manager
Live Data
Live Data
Live Data
Window Manager
Permission Manager
Dialog Manager
FRAGMENT


Activity created
onCreate
onStart
onResume
Activity rotated
onPause
onStop
onDestroy
onCreate
onStart
onResume
finish()
onPause
onStop
onDestroy
Finished
ViewModel Scope
onCleared()

Kotlin coroutines

- Możliwość zapisania asynchronicznych operacji w synchroniczny sposób
- Koncepcja „suspending function”
- Możliwość wykonywania operacji w różnych wątkach

```
fun showToastAfterClick() {
    view.setOnClickListener({ v ->
        Toast.makeText(activity,"click",Toast.LENGTH_SHORT).show() })
}
```

```
fun showToastAfterClick() {
    launch ( UI ) {
        waitForClick(view)
        Toast.makeText(activity,"click",Toast.LENGTH_SHORT).show()
    }
}

suspend fun waitForClick(view: View) =
        suspendCoroutine<Any?> { cont: Continuation<Any?> ->
            view.setOnClickListener({ cont.resume(null) })
     }
```

```
fun showToastAfterClick() {
    view.setOnClickListener({ v ->
        Toast.makeText(activity,"click",Toast.LENGTH_SHORT).show() })
}
```

```
fun showToastAfterClick() {
    launch ( UI ) {
        waitForClick(view)
        Toast.makeText(activity,"click",Toast.LENGTH_SHORT).show()
    }
}

suspend fun waitForClick(view: View) =
        suspendCoroutine<Any?> { cont: Continuation<Any?> ->
            view.setOnClickListener({ cont.resume(null) })
    }
```

```
fun showToastAfterClick() {
    view.setOnClickListener({ v ->
        continuation()
    })
}

fun continuation() {
    Toast.makeText(activity, "click", Toast.LENGTH_SHORT).show()
}
```

```
fun showToastAfterClick() {
    launch(UI) {
        waitForClick(view)
        Toast.makeText(activity, "click", Toast.LENGTH_SHORT).show()
    }
}

suspend fun waitForClick(view: View) =
        suspendCoroutine<Any?> { cont: Continuation<Any?> ->
            view.setOnClickListener({ cont.resume(null) })
        }
```

Suspend, Continuation

```
suspend fun startActivityForResult(intent: Intent): IntentResult =
        suspendCoroutine { cont: Continuation<IntentResult> ->

            val tag = System.currentTimeMillis().toString() + intent.toString()
            windowHelper.windowManager.startActivity(tag, intent)
            windowHelper.windowManager.registerIntentHandler(tag, {
                requestCode, resultCode, resultData ->
                        cont.resume(IntentResult(requestCode, resultCode, resultData)) })
        }
```

Suspend, Continuation

```
suspend fun startActivityForResult(intent: Intent): IntentResult =
        suspendCoroutine { cont: Continuation<IntentResult> ->

            val tag = System.currentTimeMillis().toString() + intent.toString()
            windowHelper.windowManager.startActivity(tag, intent)
            windowHelper.windowManager.registerIntentHandler(tag, {
                requestCode, resultCode, resultData ->
                        cont.resume(IntentResult(requestCode, resultCode, resultData)) })
        }
```

Suspend, Continuation

```
suspend fun startActivityForResult(intent: Intent): IntentResult =
        suspendCoroutine { cont: Continuation<IntentResult> ->

            val tag = System.currentTimeMillis().toString() + intent.toString()
            windowHelper.windowManager.startActivity(tag, intent)
            windowHelper.windowManager.registerIntentHandler(tag, {
                requestCode, resultCode, resultData ->
                        cont.resume(IntentResult(requestCode, resultCode, resultData)) })
        }
```

Suspend, Continuation

```
suspend fun startActivityForResult(intent: Intent): IntentResult =
        suspendCoroutine { cont: Continuation<IntentResult> ->

            val tag = System.currentTimeMillis().toString() + intent.toString()
            windowHelper.windowManager.startActivity(tag, intent)
            windowHelper.windowManager.registerIntentHandler(tag, {
                requestCode, resultCode, resultData ->
                        cont.resume(IntentResult(requestCode, resultCode, resultData)) })
        }
```

Anko

```
fun getData(): Data { ... }
fun showData(data: Data) { ... }

async(UI) {
    val data: Deferred<Data> = bg { // Runs in background
        getData()
    }

    // This code is executed on the UI thread
    showData(data.await(),)


}
```

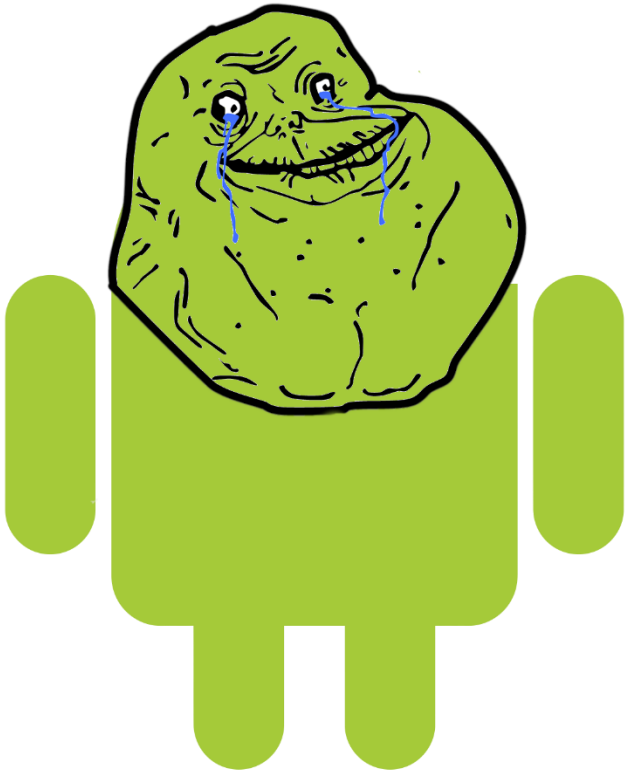
MetaLab async/await

```
async {
    progressBar.visibility = View.VISIBLE
    // Release main thread and wait until text is loaded in background thread
    val loadedText = await { loadFromServer() }
    // Loaded successfully, come back in UI thread and show the result
    txtResult.text = loadedText
    progressBar.visibility = View.INVISIBLE
}
```

Rezultat

```
fun pickPhoneNumber()
{
    launch(UI) {
        val permissionResult = requestPermissions(listOf(Manifest.permission.READ_CONTACTS))
        if(permissionResult.wasAllPermissionsGranted()) {
            val result = startActivity(createPickPhoneIntent())
            val phone = extractPhoneFromResult(getApplication(),result.resultCode,result.resultData)
            if(phone != null){
                val dialogDefinition = AlertDialogDefinition("Czy ……?", phone, "TAK", "NIE")
                val dialogResult = showDialog(dialogDefinition)
                if(dialogResult == AlertDialogResult.POSITIVE) {
                    phoneNumber.value = phone
                }
            }
        }
    }
}
```

Tak



Przydatne Strony

- https://developer.android.com/topic/libraries/architecture/viewmodel.html
- https://developer.android.com/topic/libraries/architecture/livedata.html
- https://kotlinlang.org/docs/reference/coroutines.html
- https://github.com/Kotlin/anko
- https://github.com/metalabdesign/AsyncAwait
- https://github.com/kkalisz/android-manager

FINANTEQ SA