Message Passing Between Fragments

By Muhammad Haris Bin Abid

Overview

- Fragments often need to communicate with each other
- Two primary methods: Safe Args and Bundle Arguments
- Both methods work with the Android Navigation Component

- For more info visit https://developer.android.com/guide/navigation/use-graph/pass-data

Safe Args

What is Safe Args?

- Type-safe way to pass data between fragments
- Part of the Android Navigation Component
- Generates helper classes based on your navigation graph
- Prevents runtime errors through compile-time checking

Using Safe Args:

- Ensures arguments have a valid type
- Lets you provide default values
- Generates a `<SourceDestination>Directions` class with methods for every action in that destination
- Generates a class to set arguments for every named action
- Generates a `<TargetDestination>Args` class providing access to the destination's arguments

Safe Args: Setup

Add dependencies in build.gradle (Module level):

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.navigation.safeargs.kotlin) // Add this line
}
dependencies {
    // Navigation Component (already in your build.gradle)
    implementation(libs.androidx.navigation.fragment.ktx)
    implementation(libs.androidx.navigation.ui.ktx)
}
```

Update your libs.versions.toml file to include:

[plugins]

navigation-safeargs-kotlin = { id = "androidx.navigation.safeargs.kotlin", version = "2.7.6" }

Safe Args: Navigation Graph Setup

```
<fragment
    android:id="@+id/destinationFragment"
    android:name="com.example.DestinationFragment">
    <argument
        android:name="userId"
        app:argType="integer" />
    <argument
        android:name="userName"
        app:argType="string" />
</fragment>
```

Safe Args: Sending Data

```
// In SourceFragment.kt

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

    super.onViewCreated(view, savedInstanceState)

    buttonNavigate.setOnClickListener {

        // Using the generated Directions class

        val action = SourceFragmentDirections

         .actionSourceToDestination(

            userId = 123,

            userName = "John Doe” )

      findNavController().navigate(action)

    }

}
```

Safe Args: Receiving Data

```
// In DestinationFragment.kt

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

    super.onViewCreated(view, savedInstanceState)


    // Using the generated Args class

    val args: DestinationFragmentArgs by navArgs()


    val userId = args.userId        // 123

    val userName = args.userName    // "John Doe"


    // Use the arguments

    userIdTextView.text = userId.toString()

    userNameTextView.text = userName

}
```

Safe Args: Advantages

- Type safety: compiler catches errors
- Auto-generated classes: reduces boilerplate code
- Argument validation at compile time
- IDE auto-completion support
- Easier refactoring
- Default values and nullability can be specified

Supported Arguments type

Type	**Type Syntax** `app:argType=<type>`	**Supports Default Values**	**Supports Null Values**
Integer	`"integer"`	Yes	No
Float	`"float"`	Yes	No
Long	`"long"`	Yes	No
Boolean	`"boolean"`	Yes (`"true"` or `"false"`)	No
String	`"string"`	Yes	Yes
Array	above type + `"[]"` (for example, `"string[]"` `"long[]"`)	Yes (only `"@null"`)	Yes
Enum	Fully qualified name of the enum	Yes	No
Resource reference	`"reference"`	Yes	No

Supported argument types: Custom classes

Type	**Type Syntax** `app:argType=<type>`	**Supports Default Values**	**Supports Null Values**
Serializable	Fully qualified class name	Yes (only "`@null`")	Yes
Parcelable	Fully qualified class name	Yes (only "`@null`")	Yes

Bundle Arguments

- Traditional way to pass data between fragments
- Uses Bundle objects to store key-value pairs
- Works with Navigation Component and traditional fragment transactions
- More flexible but less type-safe than Safe Args

Bundle Arguments: Sending Data

```
// In SourceFragment.kt
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
   buttonNavigate.setOnClickListener {
        // Create bundle with data
        val bundle = Bundle().apply {
            putInt("userId", 123)
            putString("userName", "John Doe")
        }
        // Navigate with bundle
        findNavController().navigate(
            R.id.destinationFragment,
            bundle
        )
    }
}
```

Bundle Arguments: Receiving Data

```
// In DestinationFragment.kt
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    // Get arguments from bundle
    arguments?.let { bundle ->
        val userId = bundle.getInt("userId", -1)    // 123
        val userName = bundle.getString("userName")  // "John Doe"

        // Use the arguments
        userIdTextView.text = userId.toString()
        userNameTextView.text = userName
    }
}
```

Bundle Arguments: Advantages and Disadvantages

Advantages:

- Works without Navigation Component
- Flexible data types
- Compatible with legacy code
- No code generation step required

Disadvantages:

- Not type-safe (runtime errors possible)
- String keys prone to typos
- No compile-time validation
- More boilerplate code

Comparison: Safe Args vs Bundle Arguments

Feature	Safe Args	Bundle Arguments
Type Safety	✅ Yes	❌ No
Compile-time Validation	✅ Yes	❌ No
Code Generation	✅ Yes (auto)	❌ No
Works without Navigation	❌ No	✅ Yes
IDE Auto-completion	✅ Yes	❌ No
Learning Curve	Medium	Low
Recommended for	New projects	Legacy compatibility

Thank You