



Android

DEVPRO VIETNAM

Nội dung

I. Data transfer

1. Truyền dữ liệu giữa các activity
2. Truyền dữ liệu giữa activity và fragment
3. Truyền dữ liệu giữa các fragment

II. Networking

1. API calling với Retrofit
2. Xử lý JSON
3. Basic JWT authen

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.1 Basic extra data

Với lượng dữ liệu nhỏ, đơn giản, ta có thể dùng `putExtra` của `intent` để đẩy trực tiếp các value này vào

```
// Từ Activity hiện tại
Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);
intent.putExtra("KEY_NAME", "John Doe");
intent.putExtra("KEY_AGE", 25);
intent.putExtra("KEY_IS_STUDENT", true);
startActivity(intent);
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.1 Basic extra data

sau đó ở bên phía activity nhận, ta lấy ra dữ liệu bằng cách gọi getIntent và gọi getIntentExtra, getStringExtra ... trên intent

```
// Trong TargetActivity - nhận dữ liệu
public class TargetActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_target);

        Intent intent = getIntent();
        String name = intent.getStringExtra("KEY_NAME");
        int age = intent.getIntExtra("KEY_AGE", 0);
        boolean isStudent = intent.getBooleanExtra("KEY_IS_STUDENT", false);

        // Sử dụng dữ liệu
        TextView textView = findViewById(R.id.textView);
        textView.setText("Name: " + name + ", Age: " + age);
    }
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.1 Basic extra data

Ngoài ra ta có thể đóng gói data vào một Bundle trước khi gửi vào intent

```
// Sử dụng Bundle để đóng gói dữ liệu
Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);
Bundle bundle = new Bundle();
bundle.putString("USER_NAME", "Alice");
bundle.putInt("USER_ID", 12345);
intent.putExtras(bundle);
startActivity(intent);
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.1 Basic extra data

Bên phía activity nhận

```
public class TargetActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_target);

        // Lấy Intent và dữ liệu từ Bundle
        Intent intent = getIntent();
        Bundle bundle = intent.getExtras();

        if (bundle != null) {
            String userName = bundle.getString("USER_NAME");
            int userId = bundle.getInt("USER_ID");

            // Sử dụng dữ liệu
            Log.d("TargetActivity", "User: " + userName + ", ID: " + userId);
        }
    }
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.2 Nhận kết quả trả về từ activity con

Để có thể start một activity và sau đó nhận một data trả về từ nó ta dùng hàm

`startActivityForResult`

Để xử lý kết quả trả về ta override hàm

`onActivityResult`

```
// Từ Activity gọi
private static final int REQUEST_CODE = 1;

private void startActivityForResult() {
    Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);
    intent.putExtra("REQUEST_DATA", "Data from main activity");
    startActivityForResult(intent, REQUEST_CODE);
}

// Xử lý kết quả trả về
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            String result = data.getStringExtra("RESULT_DATA");
            // Xử lý kết quả
            Toast.makeText(this, "Result: " + result, Toast.LENGTH_SHORT).show();
        }
    }
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.2 Nhận kết quả trả về từ activity con

Bên phía activity con

Ta trả kết quả về bằng cách sử dụng một intent, gói dữ liệu vào intent sau đó gọi hàm `setResult` để gửi kết quả về

Cờ báo `RESULT_OK` là hằng số để biểu thị quá trình xử lý ở activity con là thành công và kết quả trả về có thể sử dụng được

```
// Trong TargetActivity - trả về kết quả
// Nhận dữ liệu từ Activity gọi
Intent intent = getIntent();
String requestData = intent.getStringExtra("REQUEST_DATA");

// Xử lý và trả về kết quả
Button finishButton = findViewById(R.id.finishButton);
finishButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent resultIntent = new Intent();
        resultIntent.putExtra("RESULT_DATA", "Processed: " + requestData);
        setResult(RESULT_OK, resultIntent);
        finish(); // Đóng Activity hiện tại
    }
});
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.3 Activity navigation – Task stack

Task là ngăn xếp mà hệ thống dùng để quản lý việc điều hướng các activity. Mỗi activity khi được khởi tạo thì đều phải được đưa vào một Task nào đó. Các activity được quản lý theo một ngăn xếp để bảo đảm khi nhấn nút back ta có thể quay về đúng activity trước đó.

Ví dụ: activity A start activity B, sau đó B lại start C ta sẽ có ngăn xếp như sau:

$A \rightarrow B \rightarrow C$

Trong đó C ở đỉnh ngăn xếp, do vậy từ C khi back ta sẽ về B, ở B back sẽ về A

Tuy rằng trong đa số các trường hợp thứ tự start activity sẽ trùng khớp với thứ tự trong ngăn xếp. Tuy nhiên ta có thể can thiệp vào quá trình này để điều khiển việc điều hướng theo ý muốn.

Ví dụ sau khi user login thành công và vào được màn hình chính ta không muốn user nhấn back để quay lại màn hình login nữa. Khi này ta sẽ dùng đến các Intent FLAG hoặc launchMode để thiết lập.

Có các launchMode như sau: standard, singleTop, singleTask

launchMode sẽ được thiết lập ở thẻ <activity /> của file [AndroidManifest.xml](#)

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.3.1 launchMode

Các launchMode xác định cách mà activity được đưa vào Task stack khi nó được khởi chạy

standard: luôn tạo mới Activity và đẩy vào stack

với standard: nếu A start B, B start C, C start C, C start A ta sẽ có ngăn xếp $A \rightarrow B \rightarrow C \rightarrow C \rightarrow A$ activity có thể duplicate

singleTop: sử dụng lại activity (không tạo mới) nếu nó đang ở trên top của task stack. Nếu ko ở trên top, vẫn tạo mới và đẩy vào top.

với singleTop: nếu A start B, B start B, B start B trong stack chỉ có $A \rightarrow B$ bởi vì B đang ở trên top nên nó sẽ được dung luôn và không tạo duplicate B.

Nếu A gọi B, B gọi C, C gọi B thì B sẽ vẫn được tạo mới $A \rightarrow B \rightarrow C \rightarrow B$ lần này do B cũ đang không ở top nên B mới sẽ được tạo và đẩy vào top

singleTask: không tạo lại Activity nếu nó đã tồn tại trong task, nếu nó không ở trên top thì clear các activity khác trên nó để mang nó lên top. Đảm bảo không tạo duplicate instance của Activity, sử dụng cho main entry point, hoặc dashboard screen (màn hình mà không nên bị duplicate)

launchMode singleTask sẽ hữu ích khi sử dụng cho màn hình home trong trường hợp ta cần quay lại home từ một chuỗi rất sâu các activity trước đó, nhưng không muốn phải back từng cái mà muốn nhảy thẳng về.

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.3.2 Các flag điều khiển

Ta cũng có thể sử dụng các Intent flag để điều khiển ngăn xếp

```
// Sử dụng Flags để điều khiển behavior của Activity
Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);

// Xóa stack hiện tại và tạo stack mới
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);

// Hoặc xóa các Activity trên stack
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

startActivity(intent);
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.3.2 Các flag điều khiển – giải thích chi tiết

```
Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);  
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);  
startActivity(intent);
```

1. Intent.FLAG_ACTIVITY_CLEAR_TASK

Cờ này yêu cầu hệ thống **xóa tất cả các Activity** trong Task hiện tại đang giữ Activity gọi. Nó phải luôn được sử dụng cùng với FLAG_ACTIVITY_NEW_TASK. Nó đảm bảo rằng Task hiện tại sẽ hoàn toàn trống rỗng trước khi khởi động Activity mới.

2. Intent.FLAG_ACTIVITY_NEW_TASK

Cờ này yêu cầu Activity mới phải được khởi động trong một **Task (ngăn xếp Activity) mới**. Nếu đã có Task cho Activity này (dựa trên thuộc tính taskAffinity), Activity sẽ được thêm vào Task đó. Nếu không, một Task mới sẽ được tạo.

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.3.2 Các flag điều khiển – giải thích chi tiết

```
Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);  
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);  
startActivity(intent);
```

Khi sử dụng cả hai cờ này cùng nhau:

Hệ thống sẽ **xóa hoàn toàn Task hiện tại** (bao gồm tất cả các Activity đang chạy trong đó) do **FLAG_ACTIVITY_CLEAR_TASK**.

Sau đó, hệ thống sẽ **tạo một Task mới** (hoặc tái sử dụng một Task sẵn có) và **đặt Activity mục tiêu** (được chỉ định trong intent) làm Activity gốc (root) của Task mới đó, do **FLAG_ACTIVITY_NEW_TASK**.

Hai flag này có thể ứng dụng trong trường hợp Logout: Khi người dùng đăng xuất, ta muốn xóa toàn bộ lịch sử Activity cá nhân của họ và đưa họ đến màn hình **Đăng nhập (Login) mới tinh** mà không cho phép họ quay lại các màn hình trước đó bằng nút **Back**.

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.3.2 Các flag điều khiển – giải thích chi tiết

Hoặc trong trường hợp app được khởi động từ thông báo: Khi nhấp vào thông báo, ứng dụng có thể cần khởi động một màn hình cụ thể và ta muốn đảm bảo rằng màn hình đó là **điểm khởi đầu** của một Task mới, không bị chồng lên các Task cũ.

Hay khi user thay đổi cài đặt ngôn ngữ/đổi theme: Sau khi thay đổi các cài đặt quan trọng, ta cần khởi động lại ứng dụng từ màn hình chính để áp dụng thay đổi, và muốn **Task cũ bị xóa** hoàn toàn để tránh trạng thái không nhất quán.

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.3.2 Các flag điều khiển – giải thích chi tiết

```
Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);  
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  
startActivity(intent);
```

Khi gặp cờ **CLEAR_TOP**, hệ thống sẽ tìm kiếm trong ngăn xếp xem đã có một thể hiện (instance) của Activity đích chưa.

Nếu Activity đó đã tồn tại trong ngăn xếp:

Tất cả các Activity nằm trên Activity đó trong ngăn xếp sẽ bị kết thúc (finish) và bị xóa.

Activity mục tiêu sẽ được đưa lên trên cùng và được gọi lại.

Activity mục tiêu được gọi lại theo hai cách, tùy thuộc vào thuộc tính launchMode của nó trong tệp AndroidManifest.xml:

- Nếu launchMode là standard (mặc định), Activity đó sẽ bị xóa khỏi ngăn xếp và một thể hiện mới sẽ được tạo.
- Nếu launchMode là singleTop (hoặc singleTask), Activity đó sẽ được giữ lại, không bị tạo mới mà chỉ gọi phương thức onNewIntent() của nó.

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.4 Start activity với animation

Ta có thể custom animation cho intent bằng cách gọi hàm `overridePendingTransition` và truyền vào anim tương ứng

```
private void startActivityWithAnimation() {  
    Intent intent = new Intent(CurrentActivity.this, TargetActivity.class);  
    startActivity(intent);  
  
    // Thêm animation  
    overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);  
}
```

```
<!-- res/anim/slide_in_right.xml -->  
<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">  
    <translate android:fromXDelta="100%p" android:toXDelta="0"  
        android:duration="300"/>  
</set>
```

```
<!-- res/anim/slide_out_left.xml -->  
<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">  
    <translate android:fromXDelta="0" android:toXDelta="-100%p"  
        android:duration="300"/>  
</set>
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.5 Sử dụng ActivityResultLauncher

ActivityResultLauncher được khuyến dùng để thay thế cho startActivityForResult, bởi độ tin cậy và type safety cao hơn.

Ta khai báo và khởi tạo một launcher bằng hàm `registerForActivityResult` sau đó viết phần logic xử lý kết quả bằng một lambda. Biểu thức lambda này tương đương với hàm `onActivityResult`

Khi start activity đích ta dùng launcher vừa tạo và gọi hàm launch của nó

```
public class MainActivity extends AppCompatActivity {
    private ActivityResultLauncher<Intent> activityResultLauncher;
    private TextView tvResult;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvResult = findViewById(R.id.tvResult);
        Button btnStart = findViewById(R.id.btnStart);

        // Sử dụng Lambda expression
        activityResultLauncher = registerForActivityResult(
            new ActivityResultContracts.StartActivityForResult(),
            result -> {
                if (result.getResultCode() == RESULT_OK) {
                    Intent data = result.getData();
                    if (data != null) {
                        String returnedData = data.getStringExtra("RESULT_DATA");
                        tvResult.setText("Kết quả: " + returnedData);
                    }
                } else {
                    tvResult.setText("Hủy bỏ");
                }
            }
        );

        btnStart.setOnClickListener(v -> openSecondActivity());
    }

    private void openSecondActivity() {
        Intent intent = new Intent(this, SecondActivity.class);
        intent.putExtra("REQUEST_DATA", "Hello from MainActivity");
        activityResultLauncher.launch(intent);
    }
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.5 Sử dụng ActivityResultLauncher

Ở bên phía nhận data (activity đích)

```
public class SecondActivity extends AppCompatActivity {
    private EditText etInput;
    private Button btnSubmit, btnCancel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        etInput = findViewById(R.id.etInput);
        btnSubmit = findViewById(R.id.btnSubmit);
        btnCancel = findViewById(R.id.btnCancel);

        // Nhận dữ liệu từ MainActivity
        Intent intent = getIntent();
        String requestData = intent.getStringExtra("REQUEST_DATA");
        int number = intent.getIntExtra("NUMBER", 0);

        TextView tvReceived = findViewById(R.id.tvReceived);
        tvReceived.setText("Nhận được: " + requestData + " - " + number);

        setupClickListeners();
    }
}
```

```
private void setupClickListeners() {
    btnSubmit.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String inputText = etInput.getText().toString().trim();
            if (!inputText.isEmpty()) {
                returnResult(inputText);
            } else {
                Toast.makeText(...).show();
            }
        }
    });

    btnCancel.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            setResult(RESULT_CANCELED);
            finish();
        }
    });
}

private void returnResult(String data) {
    Intent resultIntent = new Intent();
    resultIntent.putExtra("RESULT_DATA", data);
    resultIntent.putExtra("RESULT_NUMBER", 42);
    resultIntent.putExtra("TIMESTAMP", System.currentTimeMillis());
    setResult(RESULT_OK, resultIntent);
    finish();
}
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.5 Sử dụng ActivityResultLauncher

Sử dụng contract để nhận result từ những activity hệ thống như xin cấp quyền.

ở đây ta set contract là

`ActivityResultContracts.RequestPermission`

Trong ví dụ này ta yêu cầu quyền truy cập camera

Khi cần yêu cầu nhiều quyền ta có thể dùng contract

`ActivityResultContracts.RequestMultiplePermissions`

```
public class MainActivity extends AppCompatActivity {
    private ActivityResultLauncher<String> launcherForPermission;
    private TextView tvPermissionResult;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        initView();
        setupLaunchers();
    }

    private void initView() {
        tvPermissionResult = findViewById(R.id.tvPermissionResult);
        findViewById(R.id.btnRequestPermission).setOnClickListener(v -> requestPermission());
    }

    private void setupLaunchers() {
        // Launcher cho yêu cầu quyền
        launcherForPermission = registerForActivityResult(
            new ActivityResultContracts.RequestPermission(),
            isGranted -> {
                tvPermissionResult.setText("Camera Permission: " +
                    (isGranted ? "GRANTED" : "DENIED"));
            });
    }

    private void requestPermission() {
        launcherForPermission.launch(Manifest.permission.CAMERA);
    }
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.5 Sử dụng ActivityResultLauncher

Các contract có thể dung:

Contract	Mục đích	Kết quả trả về (O)
TakePicture	Chụp ảnh bằng ứng dụng camera.	Boolean (True nếu ảnh được lưu thành công)
TakePicturePreview	Chụp ảnh và trả về bitmap nhỏ (thumbnail) mà không lưu vào bộ nhớ ngoài.	Bitmap
RequestMultiplePermissions	Yêu cầu một danh sách các quyền (Permissions) cùng lúc.	Map<String, Boolean> (Một Map ánh xạ tên quyền và trạng thái cấp phép)
GetContent	Mở ứng dụng chọn tệp/nội dung (ví dụ: thư viện ảnh) và chọn một tệp.	Uri (URI của tệp đã chọn)
GetMultipleContents	Mở ứng dụng chọn nội dung và chọn nhiều tệp.	List<Uri> (Danh sách URI các tệp đã chọn)
OpenDocument	Mở hộp thoại tệp của hệ thống để chọn một tệp bằng Storage Access Framework (SAF).	Uri
OpenMultipleDocuments	Mở hộp thoại tệp của hệ thống để chọn nhiều tệp bằng SAF.	List<Uri>
OpenDocumentTree	Cho phép người dùng chọn một thư mục (folder), cấp quyền truy cập lâu dài vào tất cả các tệp bên trong thư mục đó.	Uri (URI của thư mục gốc)
CreateDocument	Yêu cầu người dùng chọn vị trí và tên để tạo một tệp mới.	Uri

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.5 Sử dụng ActivityResultLauncher

Ví dụ sử dụng contract lấy ảnh từ thư viện, chụp ảnh, xin quyền

```
public class AdvancedActivity extends AppCompatActivity {

    private ActivityResultLauncher<String> pickImageLauncher;
    private ActivityResultLauncher<Intent> takePictureLauncher;
    private ActivityResultLauncher<String[]> multiplePermissionsLauncher;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_advanced);

        setupContracts();

        findViewById(R.id.btnPickImage).setOnClickListener(v -> pickImage());
        findViewById(R.id.btnTakePicture).setOnClickListener(v -> takePicture());
        findViewById(R.id.btnRequestPermissions).setOnClickListener(v -> requestMultiplePermissions());
    }
}
```

```
private void setupContracts() {
    // Chọn ảnh từ gallery
    pickImageLauncher = registerForActivityResult(
        new ActivityResultContracts.GetContent(),
        uri -> {
            if (uri != null) {
                // Xử lý URI ảnh
                Toast.makeText(this, "Image selected: " + uri.toString(), Toast.LENGTH_SHORT).show();
            }
        });

    // Chụp ảnh
    takePictureLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        result -> {
            if (result.getResultCode() == RESULT_OK) {
                Toast.makeText(this, "Picture taken successfully", Toast.LENGTH_SHORT).show();
            }
        });

    // Yêu cầu nhiều quyền
    multiplePermissionsLauncher = registerForActivityResult(
        new ActivityResultContracts.RequestMultiplePermissions(),
        permissions -> {
            boolean allGranted = true;
            for (Boolean isGranted : permissions.values()) {
                if (!isGranted) {
                    allGranted = false;
                    break;
                }
            }
            Toast.makeText(this, "All permissions granted: " + allGranted, Toast.LENGTH_LONG).show();
        });
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.5 Sử dụng ActivityResultLauncher

Ví dụ sử dụng contract lấy ảnh từ thư viện, chụp ảnh, xin quyền

```
private void pickImage() {
    pickImageLauncher.launch("image/*");
}

private void takePicture() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        takePictureLauncher.launch(takePictureIntent);
    }
}

private void requestMultiplePermissions() {
    String[] permissions = {
        Manifest.permission.CAMERA,
        Manifest.permission.READ_EXTERNAL_STORAGE,
        Manifest.permission.ACCESS_FINE_LOCATION
    };
    multiplePermissionsLauncher.launch(permissions);
}
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.5 Sử dụng `ActivityResultLauncher`

Các lưu ý khi dùng `ActivityResultLauncher`

- Không cần `onActivityResult()`
- Đăng ký launcher trong `onCreate()` hoặc `onStart()`
- Sử dụng các Contract có sẵn cho các tác vụ phổ biến
- Kiểm tra null khi nhận dữ liệu trả về

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.6 Sử dụng Serializable / Parcelable

Khi cần truyền cả một đối tượng phức tạp (nhiều property) intent cho phép đưa trực tiếp đối tượng đó vào với một key duy nhất. Ví dụ

```
User user = new User("John Doe", 25, "john@example.com");
```

```
Intent intent = new Intent(MainActivity.this, DetailActivity.class);  
intent.putExtra("USER_DATA", user);  
startActivity(intent);
```

Sau đó ở phía nhận, đối tượng có thể được lấy ra:

```
User user = (User) getIntent().getSerializableExtra("USER_DATA");
```

Để quá trình này hoạt động class `User` cần phải implement interface `Serializable`

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.6.2 Sử dụng Parcelable

Parcelable là một interface của Android, được tối ưu cho hiệu suất cao hơn Serializable. Nên dùng Parcelable nếu cần truyền nhiều đối tượng, đối tượng chứa lượng data lớn, phức tạp. Với parcelable class data cần code nhiều hơn:

```
public class User implements Parcelable {
    private String name;
    private int age;
    private String email;

    public User(String name, int age, String email) {
        this.name = name;
        this.age = age;
        this.email = email;
    }

    // Constructor từ Parcel
    protected User(Parcel in) {
        name = in.readString();
        age = in.readInt();
        email = in.readString();
    }
}
```

```
// Creator
public static final Creator<User> CREATOR = new Creator<User>() {
    @Override
    public User createFromParcel(Parcel in) {
        return new User(in);
    }

    @Override
    public User[] newArray(int size) {
        return new User[size];
    }
};

@Override
public int describeContents() {
    return 0;
}
```

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.6.2 Sử dụng Parcelable

Với parcelable class data cần code nhiều hơn:

```
@Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeInt(age);
        dest.writeString(email);
    }

    // Getter methods
    public String getName() { return name; }
    public int getAge() { return age; }
    public String getEmail() { return email; }
}
```

`writeToParcel` để ghi các property vào Parcel
ở đây có 3 property là name, age, và email

I. Data transfer

1. Truyền dữ liệu giữa các activity

1.6.2 Sử dụng Parcelable

Để giảm boilerplate code, có thể sử dụng thư viện Parceler

```
@Parcel
public class Book {
    String title;
    String author;
    int pages;

    public Book() {} // Constructor rỗng bắt buộc

    public Book(String title, String author, int pages) {
        this.title = title;
        this.author = author;
        this.pages = pages;
    }

    // Getter methods
    public String getTitle() { return title; }
    public String getAuthor() { return author; }
    public int getPages() { return pages; }
}
```

Với thư viện Parceler (add dependencies trong gradle) bây giờ chỉ cần thêm @Parcel phía trên khai báo class là có thể dùng được Parcelable

// Sử dụng

```
Book book = new Book("Android Development", "John Smith", 400);
Intent intent = new Intent(this, DetailActivity.class);
intent.putExtra("BOOK", Parcels.wrap(book));
startActivity(intent);
```

// Nhận

```
Book receivedBook = Parcels.unwrap(getIntent().getParcelableExtra("BOOK"));
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.1 Sử dụng bundle và setArguments

Đưa dữ liệu từ Activity vào fragment

```
// Tạo Fragment và truyền dữ liệu qua Bundle
MyFragment fragment = new MyFragment();
Bundle bundle = new Bundle();
bundle.putString("KEY_USER_NAME", "John Doe");
bundle.putInt("KEY_USER_AGE", 25);
bundle.putBoolean("KEY_IS_ACTIVE", true);
fragment.setArguments(bundle);
```

Tại activity (nơi gửi)

Ta đóng gói dữ liệu cần truyền vào một bundle, sau đó set bundle này trở thành argument của fragment

```
// Nhận dữ liệu từ Bundle
Bundle args = getArguments();
if (args != null) {
    userName = args.getString("KEY_USER_NAME", "Default Name");
    userAge = args.getInt("KEY_USER_AGE", 0);
    isActive = args.getBoolean("KEY_IS_ACTIVE", false);
}
```

Tại fragment (nơi nhận)

Trong hàm onCreate của fragment, lấy ra bundle bằng cách gọi `getArguments`

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.2 Gửi dữ liệu từ Fragment về Activity

2.2.1 Sử dụng một Interface khai báo trong Fragment, Trên activity implement interface này

```
public class InputFragment extends Fragment {  
    private EditText etInput;  
    private Button btnSend;  
  
    // Interface để giao tiếp với Activity  
    public interface OnDataSentListener {  
        void onDataSent(String data);  
        void onError(String message);  
    }  
  
    private OnDataSentListener listener;
```

```
@Override  
    public void onAttach(@NonNull Context context) {  
        super.onAttach(context);  
        // Kiểm tra Activity có implement interface không  
        if (context instanceof OnDataSentListener) {  
            listener = (OnDataSentListener) context;  
        } else {  
            throw new ClassCastException(context.toString() + " must impl  
            e ment OnDataSentListener");  
        }  
    }
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.2 Gửi dữ liệu từ Fragment về Activity

Xử lý gửi
data về khi
click vào nút

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_input, container, false);

    etInput = view.findViewById(R.id.etInput);
    btnSend = view.findViewById(R.id.btnSend);

    btnSend.setOnClickListener(v -> {
        String input = etInput.getText().toString().trim();
        if (!input.isEmpty()) {
            // Gửi dữ liệu về Activity
            if (listener != null) {
                listener.onDataSent(input);
            }
        } else {
            if (listener != null) {
                listener.onError("Input cannot be empty");
            }
        }
    });

    return view;
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.3 Sử dụng ViewModel để chia sẻ data

ViewModel có thể dùng để chia sẻ data hai chiều giữa Activity và Fragment

Trước hết định nghĩa một ViewModel

Giả sử data cần chia sẻ là một String và một số nguyên Integer

```
public class SharedViewModel extends ViewModel {  
    private final MutableLiveData<String> sharedData = new MutableLiveData<>();  
    private final MutableLiveData<Integer> sharedNumber = new MutableLiveData<>();  
  
    public void setSharedData(String data) {  
        sharedData.setValue(data);  
    }  
  
    public LiveData<String> getSharedData() {  
        return sharedData;  
    }  
  
    public void setSharedNumber(int number) {  
        sharedNumber.setValue(number);  
    }  
  
    public LiveData<Integer> getSharedNumber() {  
        return sharedNumber;  
    }  
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.3 Sử dụng ViewModel để chia sẻ data

Trong Activity khai báo
ViewModel, khởi tạo nó thông
qua `ViewModelProvider`

sau đó với dữ liệu nào muốn
nhận từ Fragment thì gọi hàm
set tương ứng và gọi `observe`
để gắn lambda xử lý data vào

Với data muốn gửi đi, chỉ cần
trực tiếp gọi hàm set tương ứng
trên viewModel

```
public class MainActivity extends AppCompatActivity {
    private SharedViewModel viewModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Lấy ViewModel
        viewModel = new ViewModelProvider(this).get(SharedViewModel.class);

        // Quan sát dữ liệu từ Fragment
        viewModel.getSharedData().observe(this, data -> {
            if (data != null) {
                TextView tvData = findViewById(R.id.tvData);
                tvData.setText("From Fragment: " + data);
            }
        });

        // Gửi dữ liệu đến Fragment
        Button btnSendToFragment = findViewById(R.id.btnSendToFragment);
        btnSendToFragment.setOnClickListener(v -> {
            viewModel.setSharedNumber(42);
        });
    }
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.3 Sử dụng ViewModel để chia sẻ data

Quá trình xử lý ở phía Fragment diễn ra tương tự:

Ta khai báo một instance của ViewModel, sau đó khởi tạo nó nhờ `ViewModelProvider`

Quá trình gửi và nhận diễn ra tương tự

Với chút khác biệt nhỏ khi truyền context: dùng `getViewLifecycleOwner` thay cho biến `this`

```
public class MyFragment extends Fragment {
    private SharedViewModel viewModel;
    private EditText etFragmentInput;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_my, container, false);

        etFragmentInput = view.findViewById(R.id.etFragmentInput);
        Button btnSendToActivity = view.findViewById(R.id.btnSendToActivity);

        // Lấy ViewModel từ Activity
        viewModel = new ViewModelProvider(requireActivity()).get(SharedViewModel.class);

        // Quan sát dữ liệu từ Activity
        viewModel.getSharedNumber().observe(getViewLifecycleOwner(), number -> {
            if (number != null) {
                TextView tvNumber = view.findViewById(R.id.tvNumber);
                tvNumber.setText("From Activity: " + number);
            }
        });

        btnSendToActivity.setOnClickListener(v -> {
            String input = etFragmentInput.getText().toString().trim();
            if (!input.isEmpty()) {
                viewModel.setSharedData(input);
            }
        });

        return view;
    }
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.5 Sử dụng Fragment Result API của AndroidX

Tại activity, trong hàm onCreate, gọi `setFragmentManagerListener` trên `FragmentManager` để thiết lập một `ResultListener` cho activity hiện tại

```
// Thiết lập FragmentManager để nhận kết quả từ Fragment
getSupportFragmentManager().setFragmentManagerListener("REQUEST_KEY", this,
    new FragmentResultListener() {
        @Override
        public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle bundle) {
            // Nhận dữ liệu từ Fragment
            String result = bundle.getString("DATA_KEY");
            int number = bundle.getInt("NUMBER_KEY", 0);

            TextView tvResult = findViewById(R.id.tvResult);
            tvResult.setText("Result: " + result + ", Number: " + number);
        }
    });
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.5 Sử dụng Fragment Result API của AndroidX

Quá trình gửi dữ liệu từ activity thực hiện bằng cách gọi `setFragmentResult` trên `FragmentManager`, dữ liệu gói trong một `Bundle`

```
// Gửi dữ liệu đến Fragment
private void sendDataToFragment() {
    Bundle result = new Bundle();
    result.putString("FROM_ACTIVITY", "Hello from Activity");
    getSupportFragmentManager().setFragmentResult("ACTIVITY_REQUEST_KEY", result);
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.5 Sử dụng Fragment Result API của AndroidX

Ở phía fragment, trong hàm onCreateView, cũng gọi `setFragmentManager()` và tạo một instance của `FragmentManager`

Chú ý: `ACTIVITY_REQUEST_KEY` trùng với phần gửi của activity

```
// Nhận dữ liệu từ Activity
getParentFragmentManager().setFragmentManager("ACTIVITY_REQUEST_KEY", this,
    new FragmentResultListener() {
        @Override
        public void onFragmentResult(@NonNull String requestKey, @NonNull Bundle bundle) {
            String dataFromActivity = bundle.getString("FROM_ACTIVITY");
            Toast.makeText(getContext(), dataFromActivity, Toast.LENGTH_SHORT).show();
        }
    });
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.5 Sử dụng Fragment Result API của AndroidX

Từ Fragment gửi data về activity cũng làm tương tự: gói data vào bundle, gọi `setFragmentResult` trên `FragmentManager`, với Request key trùng khớp

```
btnSend.setOnClickListener(v -> {  
    String input = etInput.getText().toString().trim();  
    if (!input.isEmpty()) {  
        // Gửi dữ liệu về Activity  
        Bundle result = new Bundle();  
        result.putString("DATA_KEY", input);  
        result.putInt("NUMBER_KEY", 123);  
  
        getParentFragmentManager().setFragmentResult("REQUEST_KEY", result);  
    }  
});
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.6 Sử dụng Event Bus

Thư viện EventBus cho phép truyền data một cách linh hoạt giữa nhiều phần của app: Activity, Fragment, Service. Để sử dụng EventBus, trước hết cần add thư viện trong gradle

```
dependencies {  
    // Phiên bản có thể thay đổi, hãy kiểm tra phiên bản mới nhất  
    implementation 'org.greenrobot:eventbus:3.3.1'  
}
```

Sau đó định nghĩa một Event class đóng vai trò làm Data Model. Loại hình dữ liệu mà ta cần gửi sẽ quyết định các property trong class này

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.6 Sử dụng Event Bus

Trong ví dụ này Event class sẽ cho phép gửi và nhận data dạng String, đồng thời có thêm một property source để biểu thị Event này được gửi từ đâu (không bắt buộc)

```
// Event class
public class MessageEvent {
    private String message;
    private String source;

    public MessageEvent(String message, String source)
    {
        this.message = message;
        this.source = source;
    }

    public String getMessage() { return message; }
    public String getSource() { return source; }
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.6 Sử dụng Event Bus

Trong activity cần đăng ký nhận Event trong onCreate và hủy đăng ký trong onDestroy

```
// Trong Activity
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Đăng ký EventBus
        EventBus.getDefault().register(this);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        // Hủy đăng ký EventBus
        EventBus.getDefault().unregister(this);
    }
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.6 Sử dụng Event Bus

Quá trình nhận Event trên activity thực hiện thông qua việc đặt một annotation `@Subscribe` trên một hàm. Hàm này có thể là hàm bất kỳ, chỉ cần nó có parameter với kiểu trùng với kiểu của Data Model class mà ta đang quan tâm. EventBus sẽ lọc event dựa trên kiểu của Data Model. Tùy chọn `threadMode = ThreadMode.MAIN` cho phép thực thi trên main thread

```
// Nhận event từ Fragment
@Subscribe(threadMode = ThreadMode.MAIN)
public void onMessageEvent(MessageEvent event) {
    Toast.makeText(this, "From Fragment: " + event.getMessage(), Toast.LENGTH_SHORT).show();
}

// Gửi event đến Fragment
private void sendEventToFragment() {
    EventBus.getDefault().post(new MessageEvent("Hello from Activity", "Activity"));
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.6 Sử dụng Event Bus

Có các threadMode như sau:

ThreadMode	Mục đích
POSTING	(Mặc định) Gọi trên cùng luồng gửi. Không được dùng để cập nhật UI.
MAIN	Gọi trên luồng chính (UI Thread). Thích hợp nhất để cập nhật giao diện người dùng.
BACKGROUND	Dùng cho các tác vụ tốn thời gian, được gọi trên luồng nền.
ASYNC	Dùng cho các tác vụ độc lập tốn nhiều thời gian, được gọi trên một luồng riêng biệt.

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.6 Sử dụng Event Bus

Ở phía Fragment ta đăng ký nhận Event trong `onStart` và hủy đăng ký trong `onStop`

```
// Trong Fragment
public class MyFragment extends Fragment {
    @Override
    public void onStart() {
        super.onStart();
        EventBus.getDefault().register(this);
    }

    @Override
    public void onStop() {
        super.onStop();
        EventBus.getDefault().unregister(this);
    }
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.6 Sử dụng Event Bus

Quá trình gửi và nhận ở phía Fragment diễn ra tương tự

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void onMessageEvent(MessageEvent event) {
    // Nhận event từ Activity
    if (event.getSource().equals("Activity")) {
        TextView tvMessage = getView().findViewById(R.id.tvMessage);
        tvMessage.setText(event.getMessage());
    }
}

private void sendEventToActivity() {
    EventBus.getDefault().post(new MessageEvent("Hello from Fragment", "Fragment"));
}
```

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.7 Tổng kết

Mỗi phương pháp truyền data đều có ưu – nhược điểm riêng. Việc quan trọng là chọn đúng giải pháp phù hợp với use-case để phát huy tối đa các ưu điểm và hạn chế nhược điểm của chúng

Phương pháp	Ưu điểm	Nhược điểm	Use Case
Bundle	Đơn giản, dễ sử dụng	Chỉ truyền được 1 lần	Truyền dữ liệu khởi tạo
Interface	Type-safe, rõ ràng	Code nhiều, phụ thuộc Activity	Giao tiếp 2 chiều
ViewModel	Live data, reactive	Phức tạp hơn	Chia sẻ dữ liệu real-time
Fragment Result	Hiện đại, ít phụ thuộc	API mới, ít linh hoạt	Giao tiếp 1 lần
EventBus	Linh hoạt, decouple	Thư viện third-party	Ứng dụng phức tạp

I. Data transfer

2. Truyền dữ liệu giữa activity và fragment

2.7 Tổng kết

Khuyến nghị:

- Dùng **Bundle** cho dữ liệu khởi tạo
- Dùng **ViewModel** cho chia sẻ dữ liệu real-time
- Dùng **Interface** cho giao tiếp cụ thể, type-safe
- Dùng **Fragment Result API** cho Android mới

I. Data transfer

3. Truyền dữ liệu giữa fragment fragment

VD: Sử dụng Interface

```
public interface FragmentCommunication {  
    void onMessageSent(String message);  
    void onDataSent(User user);  
    String onRequestData();  
}
```

```
public class MainActivity extends AppCompatActivity implements FragmentCommunication {  
    private String sharedData = "Default Data";  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) { ...}  
  
    @Override  
    public void onMessageSent(String message) {  
        // Chuyển message đến Fragment B  
        FragmentB fragmentB = (FragmentB) getSupportFragmentManager()  
            .findFragmentById(R.id.container_b);  
        if (fragmentB != null) {  
            fragmentB.onMessageReceived(message);  
        }  
    }  
  
    @Override  
    public void onDataSent(User user) {  
        // Chuyển user data đến Fragment B  
        FragmentB fragmentB = (FragmentB) getSupportFragmentManager()  
            .findFragmentById(R.id.container_b);  
        if (fragmentB != null) {  
            fragmentB.onUserReceived(user);  
        }  
    }  
  
    @Override  
    public String onRequestData() {  
        return sharedData;  
    }  
}
```