

**UJIAN TENGAH SEMESTER
PEMROGRAMAN MOBILE**



Dosen Pengampu: Nova Agustina, ST., M.Kom.

Disusun Oleh:

Bintang Akmal Kurniawan (23552011195)

Essay

1. Apa fungsi findViewById?
2. Apa syarat pemanggilan method findViewById? Buat contohnya dan screenshot source code nya!
3. Error apa yang terjadi jika file kotlin salah menginisialisasi findViewById atau objek pada xml belum diinisialisasi?
4. Buat sebuah contoh program untuk menampilkan pesan error Resources.NotFoundException! Screenshot logcat-nya!
5. Kumpulkan dalam bentuk pdf di Elearning (Soal essay digabung dengan soal studi kasus cek point 7 Studi Kasus)

Jawaban:

1. findViewById adalah fungsi buat menghubungkan komponen UI dari file XML layout ke kode Kotlin. Fungsi ini mencari View berdasarkan ID yang ditentukan.
2. - Komponen harus memiliki ID (android:id) di XML
- Untuk Activity: langsung panggil findViewById()
- Pemanggilan harus setelah setContentView()

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:scaleType="centerCrop"
    android:layout_marginEnd="8dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

Gambar di atas menunjukkan komponen XML yang memiliki id @+id/imageView.

```
class ViewHolder(@NonNull itemView: View) : RecyclerView.ViewHolder(itemView) {
    val judul: TextView = itemView.findViewById(R.id.judul)
    val subJudul: TextView = itemView.findViewById(R.id.subJudul)
    val imageView: ImageView = itemView.findViewById(R.id.imageView)
}
```

Dan gambar ini menunjukkan pemanggilan id di file kotlin.

3. Error Jika Salah Inisialisasi

NullPointerException: Jika ID tidak ditemukan di XML, `findViewById` mengembalikan null

Resources.NotFoundException: Jika layout XML yang dimaksud tidak ada

ClassCastException: Jika type casting salah (misal: `TextView` di-cast sebagai `Button`)

4. Contoh Program:

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        enableEdgeToEdge()  
        setContentView(R.layout.activity_main)  
  
        val angka = resources.getString(id: 12)  
    }  
}
```

Error di Logcat:

```
E FATAL EXCEPTION: main (Ask Gemini)  
Process: com.example.apaaja, PID: 30372  
java.lang.RuntimeException: Unable to start activity ComponentInfo{com.example.apaaja/com.example.apaaja.MainActivity}: android.content.res.Resources$NotFoundException: String  
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3782)  
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3922)  
    at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:103)  
    at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:139)  
    at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:96)  
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2443)
```

Penjelasan Source Code:

1) SplashActivity

```
<ImageView
    android:id="@+id/logoImage"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:src="@drawable/logo"
    android:layout_centerInParent="true" />
```

Menampilkan logo yang sudah di buat di halaman activity_splash.

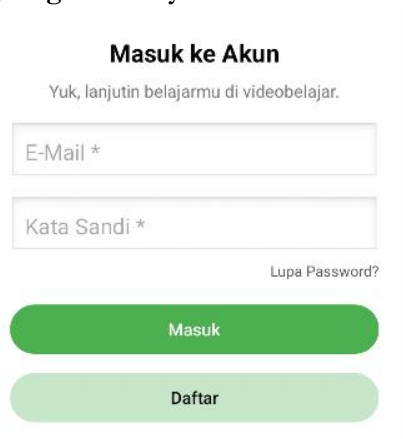
```
class SplashActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

        Handler(Looper.getMainLooper()).postDelayed({
            startActivity(Intent( packageContext: this, LoginActivity::class.java))
            finish()
        }, delayMillis: 2000)
    }
}
```

Dengan memanggil setContentView "activity_splash" selama 2000 (2 Detik) setelah itu akan menampilkan LoginActivity yang dimana didalamnya memanggil halaman "activity_login".

2) LoginActivity



The screenshot shows a login interface with the title "Masuk ke Akun". Below the title is a subtitle "Yuk, lanjutin belajarmu di videobelajar.". There are two input fields: "E-Mail *" and "Kata Sandi *". Below the password field is a link "Lupa Password?". At the bottom, there are two buttons: "Masuk" (green) and "Daftar" (light green).

Dari gambar di atas terdapat banyak Element seperti EditText dan Button yang dimana setiap element mempunyai Id nya masing-masing, antara lain:

- EditText (E-Mail), dengan ID **emailInput**
- EditText (Password), dengan ID **passwordInput**
- Button (Masuk), dengan ID **loginButton**
- Button (Daftar), dengan ID **registerButton**

```
val etEmail = findViewById<EditText>(R.id.emailInput)
val etPassword = findViewById<EditText>(R.id.passwordInput)
val btnLogin = findViewById<Button>(R.id.loginButton)
val btnDaftar = findViewById<Button>(R.id.registerButton)
```

- **Fungsi:** Menghubungkan komponen XML ke dalam kode Kotlin agar bisa digunakan.

```
btnLogin.setOnClickListener {
    val email = etEmail.text.toString()
    val password = etPassword.text.toString()

    // Gabungkan semua user (dummy + hasil register)
    val allUsers = RegisterActivity.registeredUsers

    val isValidUser = allUsers.any { it.first == email && it.second == password }

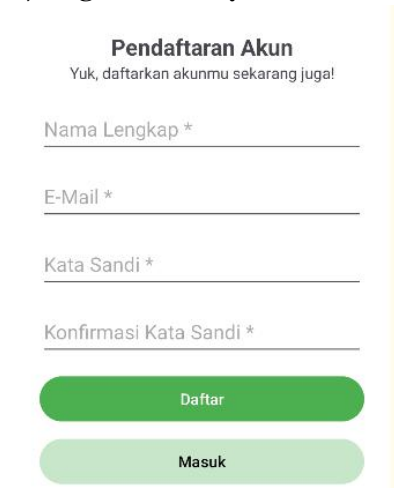
    if (isValidUser) {
        Toast.makeText(context: this, text: "Login berhasil!", Toast.LENGTH_SHORT).show()
        startActivity(Intent(packageContext: this, HomeActivity::class.java))
        finish()
    } else {
        Toast.makeText(context: this, text: "Email atau password salah!", Toast.LENGTH_SHORT).show()
    }
}
```

- **Fungsi:** Dieksekusi saat tombol login diklik.
- **etEmail.text.toString():** Mengambil teks dari input email.
- **RegisterActivity.registeredUsers:** Mengambil daftar user yang terdaftar (berasal dari RegisterActivity).
- **any { it.first == email && it.second == password }:**
 - Mengecek apakah ada user dengan email dan password yang cocok.
 - **it.first:** email, **it.second:** password dari tuple (**Pair<String, String>**).
- Jika cocok (**isValidUser == true**):
 - Menampilkan toast "Login berhasil!".
 - Berpindah ke halaman HomeActivity dan mengakhiri LoginActivity.
- Jika tidak cocok:
 - Menampilkan pesan kesalahan login.

```
btnDaftar.setOnClickListener {
    startActivity(Intent(packageContext: this, RegisterActivity::class.java))
}
```

- **Fungsi:** Saat tombol daftar diklik, pindah ke halaman registrasi.
- **Intent(this, RegisterActivity::class.java):** Membuat intent untuk membuka activity (halaman register).

3) RegisterActivity



- EditText (Nama Lengkap), dengan ID **etNama**
- EditText (E-Mail), dengan ID **etEmail**
- EditText (Kata Sandi), dengan ID **etPassword**
- EditText (Konfirmasi Kata Sandi), dengan ID **etConfirmPassword**
- Button (Daftar), dengan ID **btnDaftar**
- Button (Masuk), dengan ID **btnMasuk**

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_register)  
}
```

- **Fungsi:** Menghubungkan komponen XML ke dalam kode Kotlin agar bisa digunakan.

```
val etNama = findViewById<EditText>(R.id.etNama)  
val etEmail = findViewById<EditText>(R.id.etEmail)  
val etPassword = findViewById<EditText>(R.id.etPassword)  
val etConfirmPassword = findViewById<EditText>(R.id.etConfirmPassword)  
val btnDaftar = findViewById<Button>(R.id.btnDaftar)  
val btnMasuk = findViewById<Button>(R.id.btnMasuk)
```

- Menghubungkan komponen UI di XML ke variabel Kotlin agar bisa digunakan.
- EditText → untuk mengambil input dari user (nama, email, password).
- Button → tombol aksi (daftar dan masuk).


```

btnDaftar.setOnClickListener {
    val nama = etNama.text.toString()
    val email = etEmail.text.toString()
    val password = etPassword.text.toString()
    val confirmPassword = etConfirmPassword.text.toString()

    if (nama.isEmpty() || email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty()) {
        Toast.makeText(context, this, text: "Semua kolom wajib diisi!", Toast.LENGTH_SHORT).show()
    } else if (password != confirmPassword) {
        Toast.makeText(context, this, text: "Kata sandi tidak cocok!", Toast.LENGTH_SHORT).show()
    } else {
        // Simpan ke list dummy
        registeredUsers.add(Pair(email, password))
        Toast.makeText(context, this, text: "Pendaftaran berhasil!", Toast.LENGTH_SHORT).show()
        startActivity(Intent(context, LoginActivity::class.java))
        finish()
    }
}

```

- **Fungsi:** Dieksekusi saat tombol daftar diklik.
- Lalu mengambil input yang sudah di inputkan
- Menampilkan pesan jika ada kolom yang kosong.
- Memastikan konfirmasi password cocok.
- Simpan data ke registeredUsers.
- Tampilkan pesan berhasil.
- Arahkan user ke halaman LoginActivity dan akhiri activity saat ini (finish()).

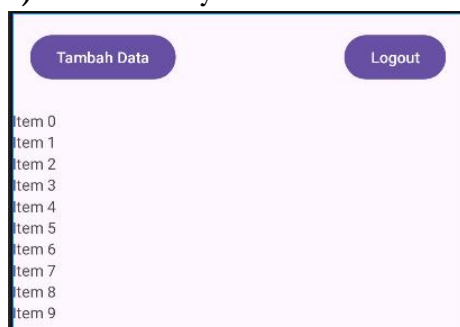
```

btnMasuk.setOnClickListener {
    startActivity(Intent(context, LoginActivity::class.java))
    finish()
}

```

- Ketika user sudah punya akun, tombol "Masuk" akan mengarahkan langsung ke halaman login.

4) HomeActivity



- Button (Tambah Data), dengan ID **btnAddData**
- Button (Logout), dengan ID **btnLogout**
- RecyclerView (Item..), dengan ID **recyclerView**

```
val btnAddData: Button = findViewById(R.id.btnAddData)
btnAddData.setOnClickListener {
    showAddDialog()
}
```

- **Fungsi:** Tombol untuk menambahkan data baru (item berupa gambar, judul, dan subjudul) ke dalam daftar RecyclerView.
- Saat diklik, tombol ini akan memanggil fungsi showAddDialog(), layoutnya ada di file **dialog_add_item.xml**
- Fungsi tersebut akan memunculkan dialog input (menggunakan AlertDialog) yang meminta pengguna memasukkan judul dan subjudul.

```
val btnLogout: Button = findViewById(R.id.btnLogout)

btnLogout.setOnClickListener {
    val sharedPref = getSharedPreferences( name: "login_session", MODE_PRIVATE)
    sharedPref.edit().clear().apply() // Hapus semua data login

    val intent = Intent( packageContext: this, LoginActivity::class.java)
    intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
    startActivity(intent)
    finish()
}
```

- **Fungsi:** Tombol untuk keluar dari akun dan kembali ke halaman login.
- Saat diklik:
 - Menghapus data login yang disimpan di SharedPreferences dengan nama "login_session".
 - Membuat Intent untuk berpindah ke LoginActivity.
 - Menambahkan flag agar semua activity sebelumnya dihapus dari backstack.
 - Memulai activity login dan menutup halaman Home (finish()).

```
itemList.add(HomeItem(R.drawable.logo, title: "Judul 1", subtitle: "Subjudul 1"))
itemList.add(HomeItem(R.drawable.logo, title: "Judul 2", subtitle: "Subjudul 2"))

recyclerView = findViewById(R.id.recyclerView)
recyclerView.layoutManager = LinearLayoutManager( context: this)

adapter = ItemAdapter(itemList,
    onEditClick = { position -> editItem(position) },
    onDeleteClick = { position -> deleteItem(position) }
)

recyclerView.adapter = adapter
```

- **Fungsi:** Menampilkan isi dari itemList menggunakan ItemAdapter.
- **itemList.add(HomeItem..)**, menjadi data default
- Setiap item bisa diedit melalui editItem(position) atau dihapus dengan deleteItem(position), tergantung tombol yang ditekan dalam setiap item.

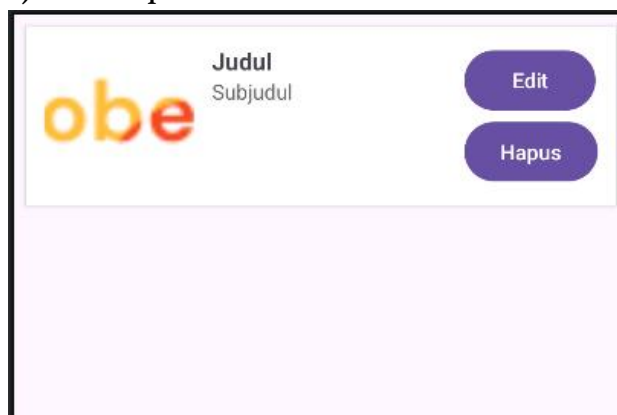
- **LinearLayoutManager** digunakan agar data ditampilkan secara vertikal satu per satu seperti daftar.

5) HomeItem

```
data class HomeItem(
    var imageResId: Int,
    var title: String,
    var subtitle: String
)
```

- Menjadi record dan type data setiap element data yang akan di tambah atau yang sudah ada.

6) ItemAdapter



- ImageView (Menampilkan Data Gambar), dengan ID **ivImage**
- TextView (Menampilkan Data Judul), dengan ID **tvTitle**
- TextView (Menampilkan Data SubJudul), dengan ID **tvSubtitle**
- Button (tombol Mengedit Data), dengan ID **btnEdit**
- Button (tombol Menghapus Data), dengan ID **btnDelete**

```
class ItemViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
    val imageView: ImageView = itemView.findViewById(R.id.ivImage)
    val titleTextView: TextView = itemView.findViewById(R.id.tvTitle)
    val subtitleTextView: TextView = itemView.findViewById(R.id.tvSubtitle)
    val btnEdit: View = itemView.findViewById(R.id.btnEdit)
    val btnDelete: View = itemView.findViewById(R.id.btnDelete)
}
```

- Ini adalah variable untuk menginisialisasi Id yang ada pada file XML.

```

override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
    val currentItem = itemList[position]
    if (currentItem is HomeItem) {
        holder.imageView.setImageResource(currentItem.imageResId)
        holder.titleTextView.text = currentItem.title
        holder.subtitleTextView.text = currentItem.subtitle
    }

    holder.btnEdit.setOnClickListener {
        onEditClick(position)
    }

    holder.btnDelete.setOnClickListener {
        onDeleteClick(position)
    }
}

```

- **Fungsi: imageView** Menampilkan gambar pada setiap item di RecyclerView.
- **Fungsi: titleTextView** Menampilkan judul dari item.
- **Fungsi: subTilteTextView** Menampilkan subjudul atau deskripsi singkat dari item.
- **Fungsi: btnEdit** Tombol untuk mengedit data item tersebut.
- **Fungsi: btnDelete** Tombol untuk menghapus data item tersebut.

Edit Data

Judul 1

Subjudul 1

BATAL UPDATE

- Edit/Update

```

private fun editItem(position: Int) {

    val item = itemList[position] // Ambil item yang ingin diedit
    val dialogView = inflater.inflate(R.layout.dialog_add_item, root: null)
    val titleEditText = dialogView.findViewById<EditText>(R.id.etTitle)
    val subtitleEditText = dialogView.findViewById<EditText>(R.id.etSubtitle)

    titleEditText.setText(item.title)
    subtitleEditText.setText(item.subtitle)

    AlertDialog.Builder(context: this)
        .setTitle("Edit Data")
        .setView(dialogView)
        .setPositiveButton(text: "Update") { _, _ ->

            val updatedTitle = titleEditText.text.toString()
            val updatedSubtitle = subtitleEditText.text.toString()

            // Perbarui item dalam list
            item.title = updatedTitle
            item.subtitle = updatedSubtitle

            // Notify adapter agar tampilan di RecyclerView diperbarui
            adapter.notifyDataSetChanged(position)
        }
        .setNegativeButton(text: "Batal", listener: null)
        .show()
}

```

- Delete

```

private fun deleteItem(position: Int) {
    itemList.removeAt(position)
    adapter.notifyItemRemoved(position)
}

```