**Praktikum Menambahkan ItemOnClickListener**

**Pada Listview**

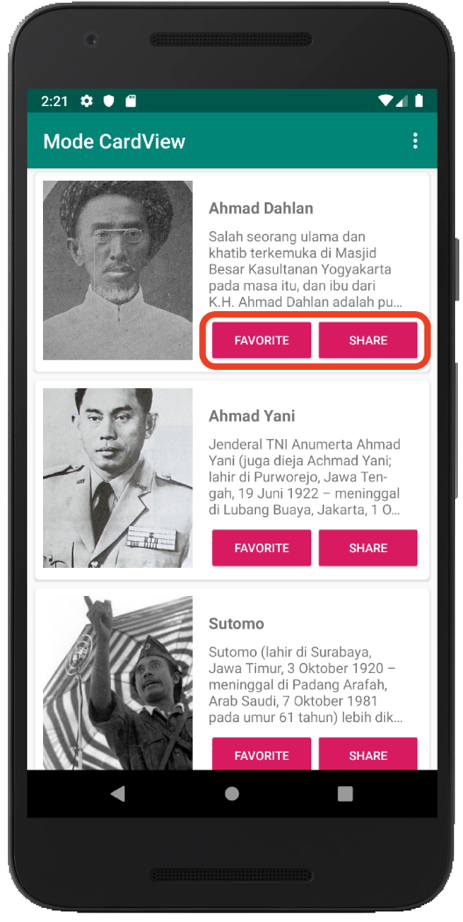
Listview memiliki listener untuk melakukan sebuah aksi ketika salah satu item pada list dipilih. Sementara itu, pada RecyclerView kita harus membuatnya secara manual.  
Pada modul ini, kita akan mengupas bagaimana caranya menangani kejadian ketika salah satu item pada list di RecyclerViewdipilih**.**

1. Bukalah kelas CardViewHeroAdapter dan perhatikan kode berikut:

|  |
| --- |
| Java |
| 1. holder.btnFavorite.setOnClickListener(new View.OnClickListener() { 2. @Override 3. public void onClick(View v) { 4. Toast.makeText(holder.itemView.getContext(), "Favorite " + 5. listHero.get(holder.getAdapterPosition()).getName(), Toast.LENGTH\_SHORT).show(); 7. } 8. }); 10. holder.btnShare.setOnClickListener(new View.OnClickListener() { 11. @Override 12. public void onClick(View v) { 13. Toast.makeText(holder.itemView.getContext(), "Share " + 14. listHero.get(holder.getAdapterPosition()).getName(), Toast.LENGTH\_SHORT).show(); 15. } 16. }); |

|  |
| --- |
| Kotlin |
| 1. holder.btnFavorite.setOnClickListener { 2. Toast.makeText(holder.itemView.context, "Favorite " + listHeroes[position].name, Toast.LENGTH\_SHORT).show() 3. } 5. holder.btnShare.setOnClickListener { 6. Toast.makeText(holder.itemView.context, "Share " + listHeroes[position].name, Toast.LENGTH\_SHORT).show() 7. } |

Jika diperhatikan, kode di atas digunakan untuk menambahkan onClick untuk btnFavorite dan btnShare.



Lalu bagaimana mengimplementasikan untuk item di dalam RecyclerView?

Tambahkan kode berikut di dalam CardViewHeroAdapter.

|  |
| --- |
| Java |
| 1. holder.**itemView**.setOnClickListener(new View.OnClickListener() { 2. @Override 3. public void onClick(View v) { 4. Toast.makeText(holder.itemView.getContext(), "Kamu memilih " + listHero.get(holder.getAdapterPosition()).getName(), Toast.LENGTH\_SHORT).show(); 5. } 6. }); |

|  |
| --- |
| Kotlin |
| 1. holder.**itemView**.setOnClickListener { 2. Toast.makeText(holder.itemView.context, "Kamu memilih " + listHeroes[position].name, Toast.LENGTH\_SHORT).show() 3. } |

Jika diperhatikan kembali metode setOnClickListener berada di dalam kelas Adapter, bagaimana cara menerapkan agar bisa kita tangani di kelas Activity?

1. Bukalah kelas ListHeroAdapter, tambahkan interface berikut dan implementasikan onClick ke kelas interface berikut:

|  |
| --- |
| Java |
| 1. public class ListHeroAdapter extends RecyclerView.Adapter<ListHeroAdapter.ListViewHolder> { 2. ... 4. **private OnItemClickCallback onItemClickCallback;** 5. **public void setOnItemClickCallback(OnItemClickCallback onItemClickCallback) {** 6. **this.onItemClickCallback = onItemClickCallback;** 7. **}** 9. ... 11. @Override 12. public void onBindViewHolder(@NonNull **final** ListViewHolder holder, int position) { 13. ... 15. **holder.itemView.setOnClickListener(new View.OnClickListener() {** 16. **@Override** 17. **public void onClick(View v) {** 18. **onItemClickCallback.onItemClicked(listHero.get(holder.getAdapterPosition()));** 19. **}** 20. **});** 21. } 23. ... 25. **public interface OnItemClickCallback {** 26. **void onItemClicked(Hero data);** 27. **}** 28. } |

|  |
| --- |
| Kotlin |
| 1. class ListHeroAdapter(val listHeroes: ArrayList<Hero>) : RecyclerView.Adapter<ListHeroAdapter.ListViewHolder>() { 2. **private lateinit var onItemClickCallback: OnItemClickCallback** 4. **fun setOnItemClickCallback(onItemClickCallback: OnItemClickCallback) {** 5. **this.onItemClickCallback = onItemClickCallback** 6. **}** 8. ... 10. override fun onBindViewHolder(holder: ListViewHolder, position: Int) { 11. ... 13. **holder.itemView.setOnClickListener { onItemClickCallback.onItemClicked(listHero[holder.adapterPosition]) }** 14. } 16. ... 18. **interface OnItemClickCallback {** 19. **fun onItemClicked(data: Hero)** 20. **}** 21. } |

Kemudian bukalah MainActivity dan tambahkan satu metode ini untuk menandakan item mana yang dipilih.

|  |
| --- |
| Java |
| 1. private void showSelectedHero(Hero hero) { 2. Toast.makeText(this, "Kamu memilih " + hero.getName(), Toast.LENGTH\_SHORT).show(); 3. } |

|  |
| --- |
| Kotlin |
| 1. private fun showSelectedHero(hero: Hero) { 2. Toast.makeText(this, "Kamu memilih " + hero.name, Toast.LENGTH\_SHORT).show() 3. } |

Kemudian tambahkan pada masing-masing metode di list seperti berikut:

|  |
| --- |
| Java |
| 1. private void showRecyclerList() { 2. rvHeroes.setLayoutManager(new LinearLayoutManager(this)); 3. ListHeroAdapter listHeroAdapter = new ListHeroAdapter(list); 4. rvHeroes.setAdapter(listHeroAdapter); 6. **listHeroAdapter.setOnItemClickCallback(new ListHeroAdapter.OnItemClickCallback() {** 7. **@Override** 8. **public void onItemClicked(Hero data) {** 9. **showSelectedHero(data);** 10. **}** 11. **});** 12. } |

|  |
| --- |
| Kotlin |
| 1. private fun showRecyclerList() { 2. rvHeroes.layoutManager = LinearLayoutManager(this) 3. val listHeroAdapter = ListHeroAdapter(list) 4. rvHeroes.adapter = listHeroAdapter 6. **listHeroAdapter.setOnItemClickCallback(object : ListHeroAdapter.OnItemClickCallback {** 7. **override fun onItemClicked(data: Hero) {** 8. **showSelectedHero(data)** 9. **}** 10. **})** 11. } |

Lakukan juga untuk metode grid!

1. Bukalah GridHeroAdapter tambahkan kode berikut di dalamnya:

|  |
| --- |
| Java |
| 1. public class GridHeroAdapter extends RecyclerView.Adapter<ListHeroAdapter.ListViewHolder> { 2. ... 4. **private OnItemClickCallback onItemClickCallback;** 5. **public void setOnItemClickCallback(OnItemClickCallback onItemClickCallback) {** 6. **this.onItemClickCallback = onItemClickCallback;** 7. **}** 9. ... 11. @Override 12. public void onBindViewHolder(@NonNull **final** ListViewHolder holder, int position) { 13. ... 15. **holder.itemView.setOnClickListener(new View.OnClickListener() {** 16. **@Override** 17. **public void onClick(View v) {** 18. **onItemClickCallback.onItemClicked(listHero.get(holder.getAdapterPosition()));** 19. **}** 20. **});** 21. } 23. ... 25. **public interface OnItemClickCallback {** 26. **void onItemClicked(Hero data);** 27. **}** 28. } |

|  |
| --- |
| Kotlin |
| 1. class GridHeroAdapter(val listHeroes: ArrayList<Hero>) : RecyclerView.Adapter<ListHeroAdapter.ListViewHolder>() { 2. **private lateinit var onItemClickCallback: OnItemClickCallback** 4. **fun setOnItemClickCallback(onItemClickCallback: OnItemClickCallback) {** 5. **this.onItemClickCallback = onItemClickCallback** 6. **}** 8. ... 10. override fun onBindViewHolder(holder: ListViewHolder, position: Int) { 11. ... 13. **holder.itemView.setOnClickListener { onItemClickCallback.onItemClicked(listHero[holder.adapterPosition]) }** 14. } 16. ... 18. **interface OnItemClickCallback {** 19. **fun onItemClicked(data: Hero)** 20. **}** 21. } |

Kemudian bukalah MainActivity dan tambahkanlah metode grid seperti berikut:

|  |
| --- |
| Java |
| 1. private void showRecyclerList() { 2. rvHeroes.setLayoutManager(new GridLayoutManager(this, 2)); 3. GridHeroAdapter gridHeroAdapter = new GridHeroAdapter(list); 4. rvHeroes.setAdapter(gridHeroAdapter); 6. **gridHeroAdapter.setOnItemClickCallback(new GridHeroAdapter.OnItemClickCallback() {** 7. **@Override** 8. **public void onItemClicked(Hero data) {** 9. **showSelectedHero(data);** 10. **}** 11. **});** 12. } |

|  |
| --- |
| Kotlin |
| 1. private fun showRecyclerGrid() { 2. rvHeroes.layoutManager = GridLayoutManager(this, 2) 3. val gridHeroAdapter = GridHeroAdapter(list) 4. rvHeroes.adapter = gridHeroAdapter 6. **gridHeroAdapter.setOnItemClickCallback(object : GridHeroAdapter.OnItemClickCallback {** 7. **override fun onItemClicked(data: Hero) {** 8. **showSelectedHero(data)** 9. **}** 10. **})** 11. } |

1. Sekarang jalankan kembali aplikasinya. Klik salah satu item pada RecyclerView bentuk list dan grid. Seharusnya sebuah obyek Toast akan tampil ketika salah satu item ditekan.

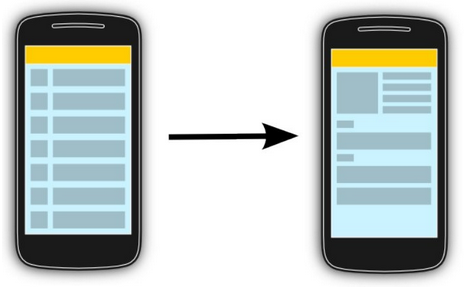


Selesai! Sejauh ini Anda telah berhasil menerapkan beragam bentuk dari RecyclerView**.** Pendekatan ini sangat berguna karena ke depannya Anda akan berhadapan dengan koleksi data yang harus ditampilkan ke dalam bentuk list.

### ****Bedah Kode****

#### **RecyclerView**

Pendekatan umum dalam mengembangkan aplikasi mobile terletak pada bentuk list-to-detail. Pendekatan ini menampilkan sejumlah informasi dalam bentuk list. Ketika pengguna menekan salah satu item untuk mendapatkan detail informasi, maka aplikasi akan menampilkannya secara detail.



Pendekatan tersebut sangat lumrah dikarenakan keterbatasan ukuran layar. Keterbatasan ini justru memudahkan penggunaan dua jempol sebagai pointer untuk melakukan sebuah aksi pada aplikasi.

Android sendiri memiliki beberapa mekanisme menampilkan data dalam jumlah yang banyak, yaitu dalam bentuk ListView, GridView, maupun RecyclerView. Namun semenjak adanya material design, para developer dituntut untuk memahami suksesor dari komponen ListView ini.

Kelebihan utama dari RecyclerView terletak pada performa dan fleksibilitasnya.

Dari sisi performa, secara default adapter dari RecyclerView haruslah mengimplementasikan ViewHolder pattern. Tujuan dari pendekatan ini adalah agar RecyclerView tetap responsif dengan mempertahankan peforma terbaik ketika menampilkan koleksi data dalam jumlah yang banyak.  
Ini jelas berbeda dengan ListView yang tidak mewajibkan pattern tersebut secara default. Pada salah satu contoh adapter yang kita tuliskan di atas, pattern ViewHolder diimplementasikan pada bagian berikut:

|  |
| --- |
| Java |
| 1. public class ListHeroAdapter extends RecyclerView.Adapter<ListHeroAdapter.ListViewHolder> { 2. private ArrayList<Hero> listHero; 3. private OnItemClickCallback onItemClickCallback; 5. public void setOnItemClickCallback(OnItemClickCallback onItemClickCallback) { 6. this.onItemClickCallback = onItemClickCallback; 7. } 8. public ListHeroAdapter(ArrayList<Hero> list) { 9. this.listHero = list; 10. } 12. @NonNull 13. @Override 14. public ListViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) { 15. View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.item\_row\_hero, viewGroup, false); 16. return new ListViewHolder(view); 17. } 19. @Override 20. public void onBindViewHolder(@NonNull final ListViewHolder holder, int position) { 21. Hero hero = listHero.get(position); 23. Glide.with(holder.itemView.getContext()) 24. .load(hero.getPhoto()) 25. .apply(new RequestOptions().override(55, 55)) 26. .into(holder.imgPhoto); 28. holder.tvName.setText(hero.getName()); 29. holder.tvFrom.setText(hero.getFrom()); 31. holder.itemView.setOnClickListener(new View.OnClickListener() { 32. @Override 33. public void onClick(View v) { 34. onItemClickCallback.onItemClicked(listHero.get(holder.getAdapterPosition())); 35. } 36. }); 37. } 39. @Override 40. public int getItemCount() { 41. return listHero.size(); 42. }  45. class ListViewHolder extends RecyclerView.ViewHolder { 46. ImageView imgPhoto; 47. TextView tvName, tvFrom; 49. ListViewHolder(View itemView) { 50. super(itemView); 51. imgPhoto = itemView.findViewById(R.id.img\_item\_photo); 52. tvName = itemView.findViewById(R.id.tv\_item\_name); 53. tvFrom = itemView.findViewById(R.id.tv\_item\_from); 54. } 55. } 57. public interface OnItemClickCallback { 58. void onItemClicked(Hero data); 59. } 60. } |

|  |
| --- |
| Kotlin |
| 1. class ListHeroAdapter(private val listHero: ArrayList<Hero>) : RecyclerView.Adapter<ListHeroAdapter.ListViewHolder>() { 2. private lateinit var onItemClickCallback: OnItemClickCallback 4. fun setOnItemClickCallback(onItemClickCallback: OnItemClickCallback) { 5. this.onItemClickCallback = onItemClickCallback 6. } 8. override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int): ListViewHolder { 9. val view: View = LayoutInflater.from(viewGroup.context).inflate(R.layout.item\_row\_hero, viewGroup, false) 10. return ListViewHolder(view) 11. } 13. override fun onBindViewHolder(holder: ListViewHolder, position: Int) { 14. val (name, from, photo) = listHero[position] 16. Glide.with(holder.itemView.context) 17. .load(photo) 18. .apply(RequestOptions().override(55, 55)) 19. .into(holder.imgPhoto) 21. holder.tvName.text = name 22. holder.tvFrom.text = from 24. holder.itemView.setOnClickListener { onItemClickCallback.onItemClicked(listHero[holder.adapterPosition]) } 25. } 27. override fun getItemCount(): Int { 28. return listHero.size 29. }  32. inner class ListViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) { 33. var imgPhoto: ImageView = itemView.findViewById(R.id.img\_item\_photo) 34. var tvName: TextView = itemView.findViewById(R.id.tv\_item\_name) 35. var tvFrom: TextView = itemView.findViewById(R.id.tv\_item\_from) 37. } 39. interface OnItemClickCallback { 40. fun onItemClicked(data: Hero) 41. } 42. } |

Setiap kali kita melakukan scroll pada RecyclerView,ia akan memeriksa memori apakah item view yang hendak ditampilkan tertentu sudah berada di memori  atau belum. Jika belum, maka akan dijalankan sebuah proses yang cukup mahal dari segi memori, yaitu dijalankannya onCreateViewHolder(). Di dalam metode ini, terjadi sebuah casting view (findViewById) yang akan menampilkan koleksi data dalam format tampilan yang ditentukan, baris per baris jika pada bentuk list atau baris dan kolom pada bentuk grid.

Hubungan antara satu adapter dengan ViewHolder adalah satu ke banyak. Artinya, satu kelas adapter bisa memiliki lebih dari satu ViewHolder.

|  |
| --- |
| Java |
| 1. rvHeroes.setHasFixedSize(**true**); |
| Kotlin |
| 1. rvHeroes.setHasFixedSize(**true**) |

Baris di atas menjelaskan bahwa bila fixed size bernilai **true**, maka RecyclerView dapat melakukan optimasi ukuran lebar dan tinggi secara otomatis. Nilai lebar dan tinggi RecyclerViewmenjadi konstan. Terlebih jika kita memiliki koleksi data yang dinamis untuk proses penambahan, perpindahan, dan pengurangan item dari koleksi data.

Pada contoh kode di atas, kita menggunakan library bernama Glide untuk menampilkan foto secara asynchronous dari URL yang diberikan.

Pada sisi fleksibilitas, RecyclerView memiliki beragam bentuk yang disesuaikan dengan design yang diinginkan. Kita hanya perlu menentukan nilai pada metode setLayoutManager() saja untuk menentukan bagaimana RecyclerView ditampilkan.

|  |
| --- |
| Java |
| 1. rvHeroes.setLayoutManager(newLinearLayoutManager(**this**)); |
| Kotlin |
| 1. rvHeroes.layoutManager = LinearLayoutManager(**this**) |

Atau

|  |
| --- |
| Java |
| 1. rvHeroes.setLayoutManager(newGridLayoutManager(**this**, 2)); |
| Kotlin |
| 1. rvHeroes.layoutManager = GridLayoutManager(**this**, 2) |

Terakhir, untuk membuat kamu lebih paham tentang **RecyclerView** beberapa link di bawah ini wajib kamu baca : [RecylerView](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html), [Using the RecyclerView](https://guides.codepath.com/android/using-the-recyclerview), [Fundamentals for ListView](https://www.bignerdranch.com/blog/recyclerview-part-1-fundamentals-for-listview-experts/), dan [RecyclerView Animator](https://github.com/wasabeef/recyclerview-animators).

#### **Glide**

Perhatikan kode berikut:

|  |
| --- |
| Java |
| 1. Glide.with(holder.itemView.getContext()) 2. .load(listHero.get(position).getPhoto()) 3. .apply(new RequestOptions().override(55, 55)) 4. .into(holder.imgPhoto); |

|  |
| --- |
| Kotlin |
| 1. Glide.with(holder.itemView.context) 2. .load(listHeroes[position].photo) 3. .apply(RequestOptions().override(55, 55)) 4. .into(holder.imgPhoto) |

Glide digunakan untuk memuat sebuah gambar, baik yang sudah Anda siapkan di drawable maupun berada di server. Glide terdiri dari banyak fungsi:

1. with: digunakan untuk memasukkan context.
2. load: digunakan untuk memasukkan sumber gambar, contohnya menggunakan url.
3. apply: digunakan untuk melakukan memanipulasi gambar, contohnya menggunakan RequestOption untuk me-*override*ukuran gambar.
4. into: digunakan untuk memasukkan imageView sebagai obyek penampil gambar.

Anda bisa baca referensi tentang Glide di tautan berikut: <https://github.com/bumptech/glide> dan <https://bumptech.github.io/glide/>.