Realm and Data binding

Android Data Binding

```
<RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:paddingLeft="16dp"
        android:onClick="@{() -> vm.onTrackSelected(track)}">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:text="@{track.name}"/>
    </RelativeLayout>
```

Android Data Binding

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);
    }
}
```

Android Data Binding: objects

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);

        Track track = new Track();
        track.setName("Platinum");
        binding.setTrack(track);
    }
}
```

Android Data Binding

```
<RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="48dp"
        android:paddingLeft="16dp"
        android:onClick="@{() -> vm.onTrackSelected(track)}">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:text="@{track.name}"/>
    </RelativeLayout>
```

Android Data Binding: binding

```
...
        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);

        PlayingPlaylistViewModel vm = new PlayingPlaylistViewModel();
        binding.setVm(vm);
    }

    public class PlayingPlaylistViewModel extends BaseObservable {
        public void onTrackSelected(Track track) {
            // Track changed
        }
    } ...
```

Realm

```
public class Track extends RealmObject {
    private long id;
    private String name;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
// id getter/setter
}
```

Realm

```
Realm realm = Realm.getDefaultInstance();
realm.executeTransaction(new Realm.Transaction() {
     @Override
     public void execute(Realm realm) {
         Track track = realm.createObject(Track.class);
         track.setId(System.currentTimeMillis());
         track.setName("Josh Pan - Platinum");
     }
});
```

Realm and Data Binding

```
public class MainActivity extends AppCompatActivity {
    PlayingPlaylistViewModel vm = new PlayingPlaylistViewModel();
    Realm realm = Realm.getDefaultInstance();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);

        binding.setTrack(realm.where(Track.class).findFirst());
        binding.setVm(vm);
     }
}
```

Realm and Data Binding: updating UI

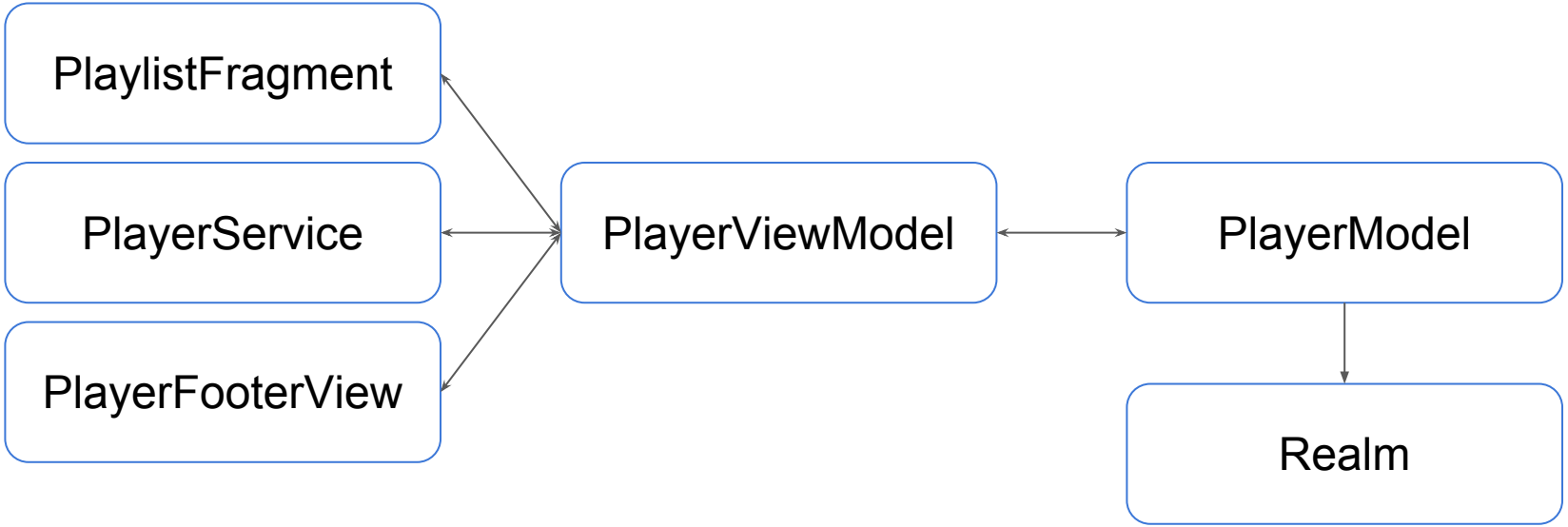
```
ActivityMainBinding binding = DataBindingUtil.setContentView(this, R.layout.activity_main);

Track track = realm.where(Track.class).findFirst();
track.addChangeListener(new RealmChangeListener() {
        @Override
        public void onChange() {
            binding.invalidateAll();
        }
});
binding.setTrack(track);
```

Realm + DataBinding

+ Possible
+ Good for small objects
+ No middleware classes
- No fields binding
- Transactions in main thread
- Queries in activity/fragment
- Hard to maintain

Music player application



DB

```
public class Player extends RealmObject {

    @PrimaryKey
    private long id;
    private Playlist playlist;
    private Track currentTrack;
    private int currentTrackProgress;
    private boolean isPlaying;
}
```

Model

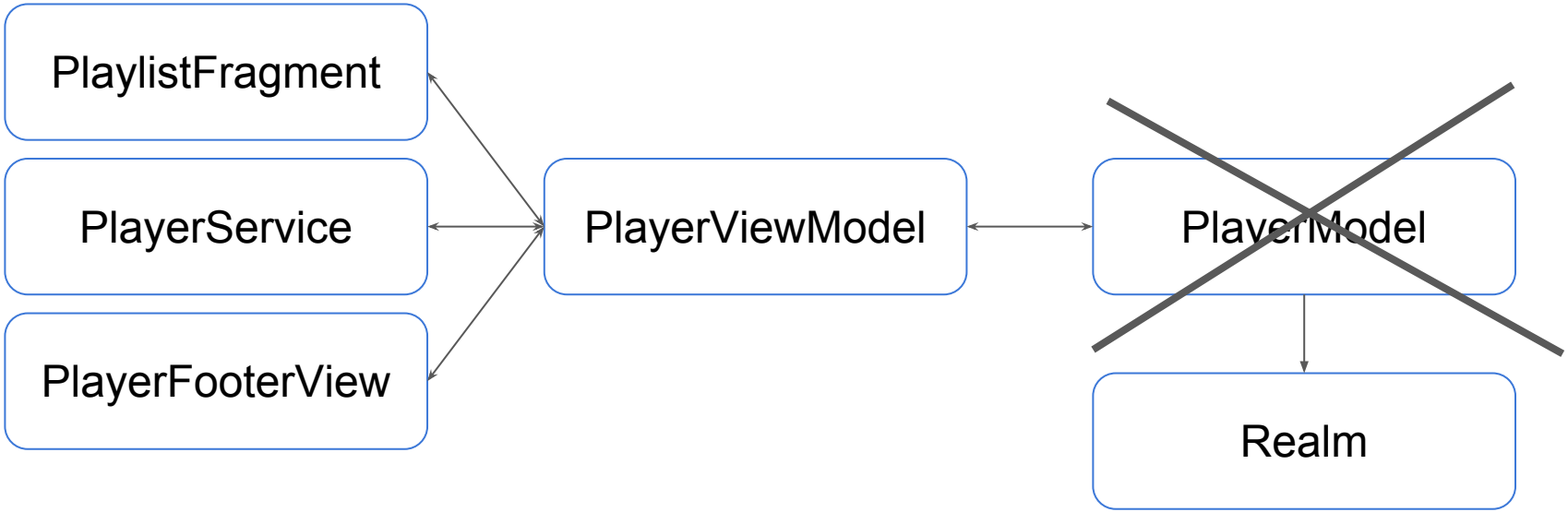
```
public class PlayerModel extends RealmModel {
    public void setPlaying(final boolean playing) {
        realm.executeTransaction({
                getPlayer().setPlaying(playing);
            }});
    }
    public void setCurrentTrack(final Track track) {
        realm.executeTransaction({
                Player player = getPlayer();
                player.setCurrentTrack(track);
            }});
    }
```

ViewModel

```
public class PlayerViewModel extends ViewModel {
// model field and constructor

    @Bindable
    public boolean isPlaying() {
        return model.isPlaying();
    }
    public void setPlaying(boolean playing) {
        model.setPlaying(playing);
        notifyPropertyChanged(BR.playing);
    }
}
```

Music player application



Music player application

ViewModel

- Application-wide
- Handles user actions
- Directly uses Realm
- Not connected with Activity lifecycle

JUST ADD DAGGER 2

VM: Module

```
@Module
public class ViewModelModule {
    private final Realm realm;
    public ViewModelModule(Realm realm) {
        this.realm = realm;
    }

    @Provides @Singleton
    PlayerViewModel providesPlayerViewModel() {
        return new PlayerViewModel(realm);
    }
}
```

VM:Component

```
@Singleton
@Component(modules = {ViewModelModule.class})
public interface ViewModelComponent {

    void inject(PlayerService service);
    void inject(PlaylistFragment fragment);
    void inject(MainActivity activity);
}
```

VM: Handles user actions & uses realm

```
public void onChangeTrack(final Track track) {
    realm.executeTransactionAsync(new Realm.Transaction() {
        @Override
        public void execute(Realm realm) {
            getPlayer().setCurrentTrack(track);
        }
    }, new Realm.Transaction.OnSuccess() {
        @Override
        public void onSuccess() {
            notifyPropertyChanged(BR.currentTrack);
        }});
}
```

ViewModel + Realm

```
public class PlayerViewModel extends ViewModel {
    @Bindable
    public boolean isPlaying() {
        return getPlayer().isPlaying();
    }

    public void setPlaying(final boolean playing) {
        realm.executeTransactionAsync( {
                getPlayer().setPlaying(playing);
            }
        }, {
                notifyPropertyChanged(BR.playing);
            }});
    }
```

Bonus: adapters and binding

- We actually can use RealmChangeListener for RealmResults<T>
- Binding works only for elements
- VM also handles user actions

MORE CODE FOR THE CODE GOD

i’m really sorry...

Single type adapter

```
public abstract class SingleTypeAdapter<Binding extends ViewDataBinding>
        extends RecyclerView.Adapter<SingleTypeAdapter<Binding>.ViewHolder> {

    // context, inflater fields
    private final int itemLayoutId;
    // constructor

    public abstract void onBindItem(Binding binding, int position);

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new ViewHolder(inflater.inflate(itemLayoutId, parent, false));
    }
```

Single type adapter

```
@Override
public void onBindViewHolder(SingleTypeAdapter<Binding>.ViewHolder holder, int position) {
    onBindItem(holder.binding, position);
}

public class ViewHolder extends RecyclerView.ViewHolder {
    public final Binding binding;
    public ViewHolder(View itemView) {
        super(itemView);
        binding = DataBindingUtil.bind(itemView);
    }
}
```

Single type Realm adapter

```
public abstract class SingleTypeRealmAdapter
<T extends RealmObject, Binding extends ViewDataBinding>
extends SingleTypeAdapter<Binding> implements RealmChangeListener {

    private final RealmResults<T> queryResults;

    public SingleTypeRealmAdapter(Context context, int itemLayoutId, RealmResults<T> queryResults){
        super(context, itemLayoutId);
        this.queryResults = queryResults;
    }
}
```

RealmChangeListener trick

```
@Override
public void onViewAttachedToWindow(ViewHolder holder) { // call superclass
    this.queryResults.addChangeListener(this);
}
@Override
public void onViewDetachedFromWindow(ViewHolder holder) { // call superclass
    this.queryResults.removeChangeListener(this);
}
@Override
public void onChange() {
    notifyDataSetChanged();
}
```

Realm object <-> Binding

```
public abstract void onBindItem(Binding binding, T item);

@Override
public void onBindItem(Binding binding, int position) {
    onBindItem(binding, queryResults.get(position));
}

@Override
public int getItemCount() {
    return queryResults.size();
}
```

Usage

```
public class PlaylistTracksAdapter extends SingleTypeRealmAdapter<Track, ItemTrackBinding> {
    private final PlaylistViewModel vm;

    public PlaylistTracksAdapter(Context context, PlaylistViewModel vm) {
        super(context, R.layout.item_track, vm.getTracks());
        this.vm = vm;
    }

    @Override
    public void onBindItem(ItemTrackBinding binding, Track item) {
        binding.setTrack(item);
        binding.setVm(vm);
    }
}
```

Thank you for your attention

Andrey Khitryy, Trinity Digital

ak@trinitydigital.ru

github.com/arkty