Data binding on Android

How to use it in the real world

Agenda

- What, why, how?
- Two-way binding
- Custom attributes
- Event handlers
- Architecture
- Tips and tricks

What, why, how?

What is data binding?

- Connect data sources with views
- Automatic updates (requires some extra effort)
- Common approach in other frameworks
  - WPF & Windows Phone (C#)
  - Angular (JavaScript)

Why use data binding?

- Cleaner code
- Simple expressions in views
- Powerful custom attributes

Getting started – Enable data binding

```
android {
    ...
    dataBinding {
        enabled = true
    }
    ...
}
```

Getting started – Layout setup

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns="...">
    <data>
        <variable name="user"
                  type="com.databinding.User"/>
    </data>

    <TextView android:text="@{user.firstName}"/>
</layout>
```

Getting started – Activity setup

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

    User user = new User("George"));
    binding.setUser(user);
}
```

How does it work?

How does it work?

- Process layout files
- Parse expressions
- Generates Java code
- Traverses the view hierarchy once to find all views
- Keeps a reference to the views

Usage

- Use it to replace `findViewById` or Butter Knife

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

    binding.userName.setText(user.getName());
}
```

Usage

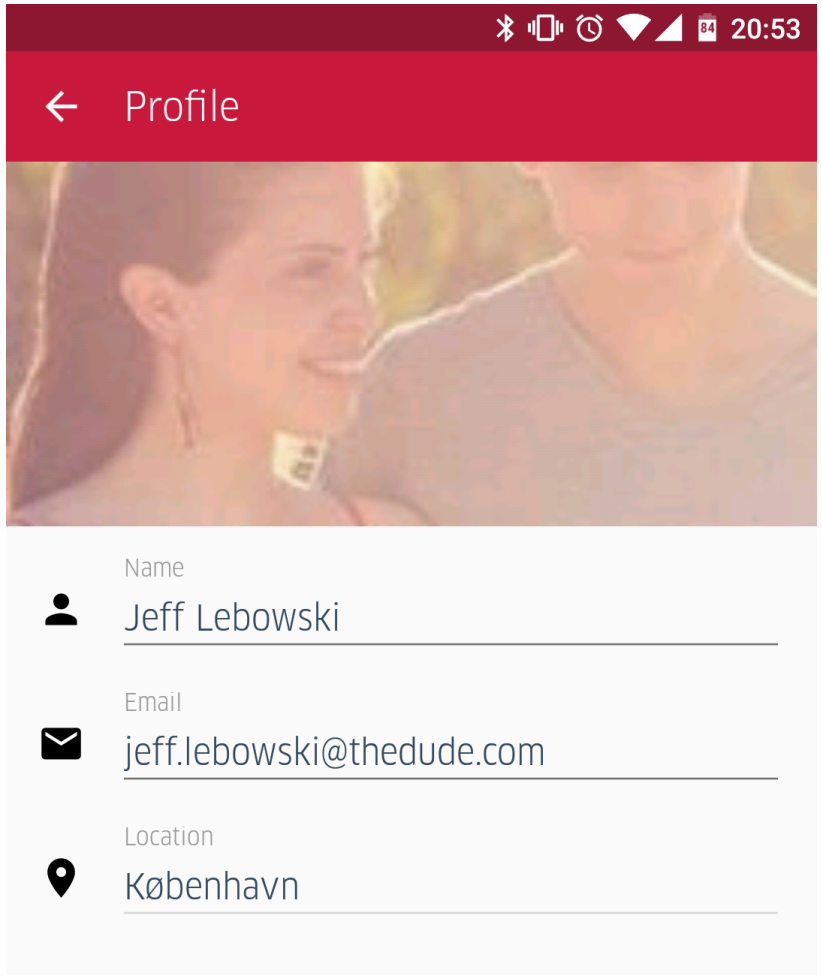
- Transforming text

```
android:text="@{String.valueOf(index + 1)}"
```

- Setting views visible based on certain conditions

```
android:visibility="@{user.age < 18 ? View.GONE
                                    : View.VISIBLE}"
```

Usage




```
<ImageView
    app:imageUrl="@{viewModel.user.pictureUrl}"
    />

<EditText
    android:text="@={viewModel.user.name}"/>

<EditText
    android:text="@={viewModel.user.email}"/>

<EditText
    android:text="@={viewModel.user.location}"
    />
```

Dealing with updates – Option 1

```
public class User {
    public final ObservableField<String>
                    firstName = new ObservableField<>();
}

user.firstName.set("Louise")
```

Dealing with updates – Option 2

```
public class User extends BaseObservable {
    private String lastName;

    @Bindable
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
        notifyPropertyChanged(BR.lastName);
    }
```

Dealing with updates - Internals

```
private  long mDirtyFlags = 0xffffffffffffffffL;

/* flag mapping
    flag 0 (0x1L): user.firstName
    flag 1 (0x2L): user
    flag 2 (0x3L): user.lastName
    flag 3 (0x4L): null
flag mapping end*/
```

Dealing with updates - Internals

```
public void setUser(User user) {
    updateRegistration(1, user);
    this.mUser = user;
    synchronized(this) {
        mDirtyFlags |= 0x2L;
    }
    notifyPropertyChanged(BR.user);
    super.requestRebind();
}
```

Dealing with updates - Internals

```
protected void executeBindings() {
    if ((dirtyFlags & 0xeL) != 0) {
        if (user != null) {
            // read user.lastName
            lastNameUser = user.getLastName();
        }
    }
}
```

Two-way binding

how to handle user input

Two-way binding

```
<EditText
    android:text="@{user.firstName}"
    … />
```

```
<EditText
    android:text="@={user.firstName}"
    … />
```

Custom attributes

aka binding adapters

Custom attributes – image loading

```
@BindingAdapter({"app:imageUrl", "app:error"})
public static void loadImage(ImageView view,
                             String url, Drawable error {

    Glide.with(view.getContext())
         .load(url)
         .error(error)
         .into(view);
}
```

Custom attributes – font

```
@BindingAdapter({"app:font"})
public static void setFont(TextView textView,
                           String fontName) {
    textView
      .setTypeface(FontCache.getInstance(context)
      .get(fontName));
}

<TextView
    app:font="@{'alegreya'}" />
```

https://github.com/lisawray/fontbinding

Event handlers

Event handlers

```
public class UserViewModel {
    …
    public View.OnClickListener clickListener;
    …
}
```

```
<View
    android:onClick="@{viewModel.clickListener}"
    … />
```

Event handlers

```
textChangedListener = new TextWatcher() {
    public void beforeTextChanged …
    public void onTextChanged …
    public void afterTextChanged …
}

<EditText
    android:addTextChangedListener=
            "@{viewModel.textChangedListener}"/>
```

Event handlers

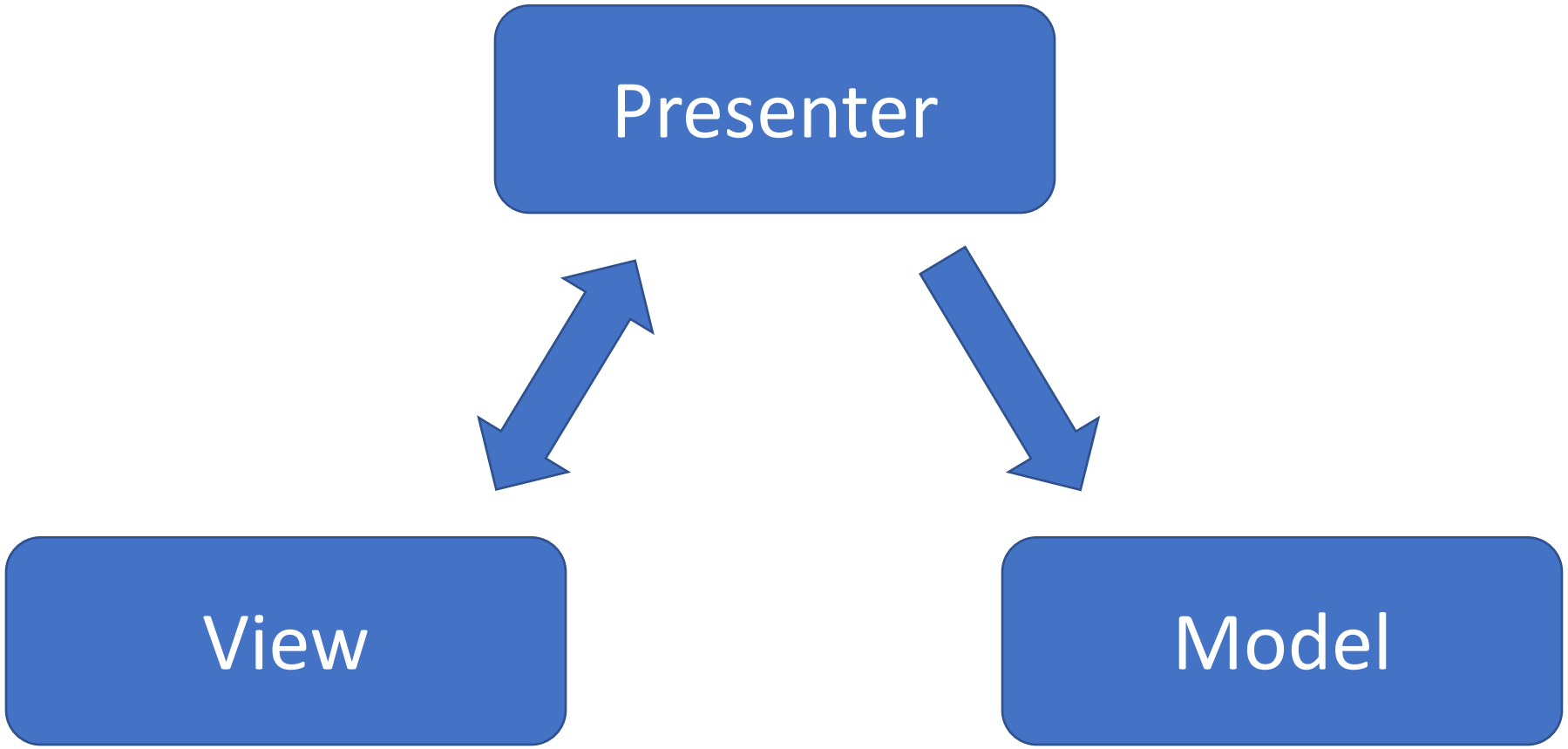
```
<EditText
    android:addTextChangedListener=
                "@{viewModel.textChangedListener}"
    … />


<EditText
    android:afterTextChanged=
                  "@{viewModel.afterTextChanged}"
    … />
```

Event handlers with parameters

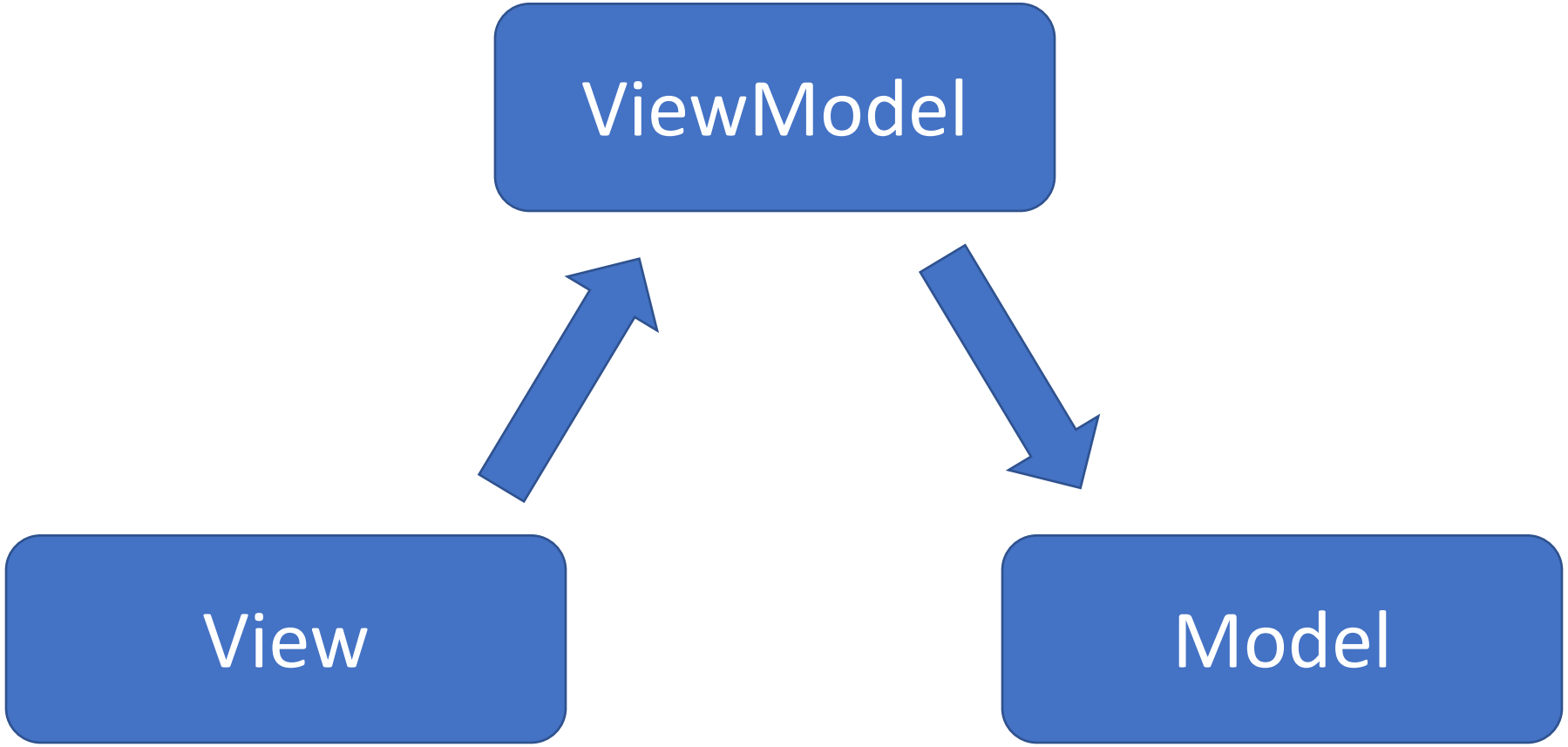
```
public class UserViewModel {
    public void save(User user)
    {
        userRepository.save(user);
    }
}

<Button
    android:onClick=
            "@{()->userViewModel.save(user)}"/>
```

Architecture

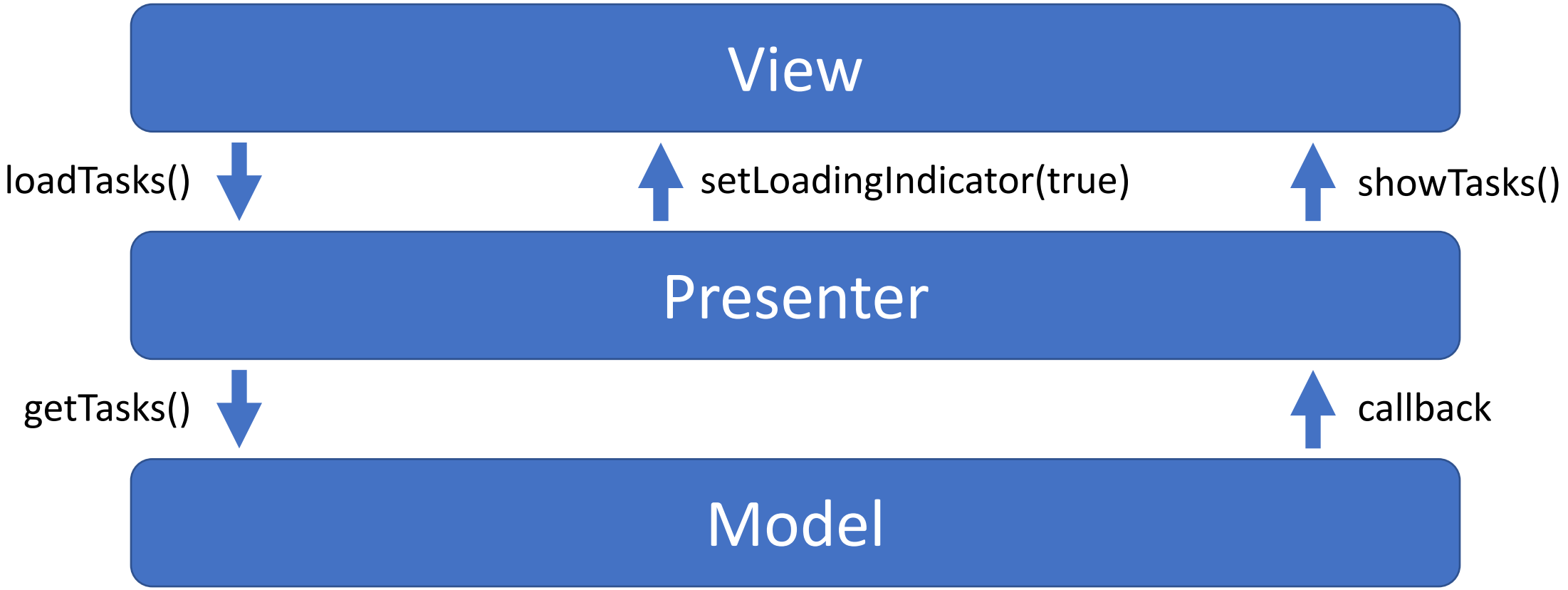
MVP



MVVM

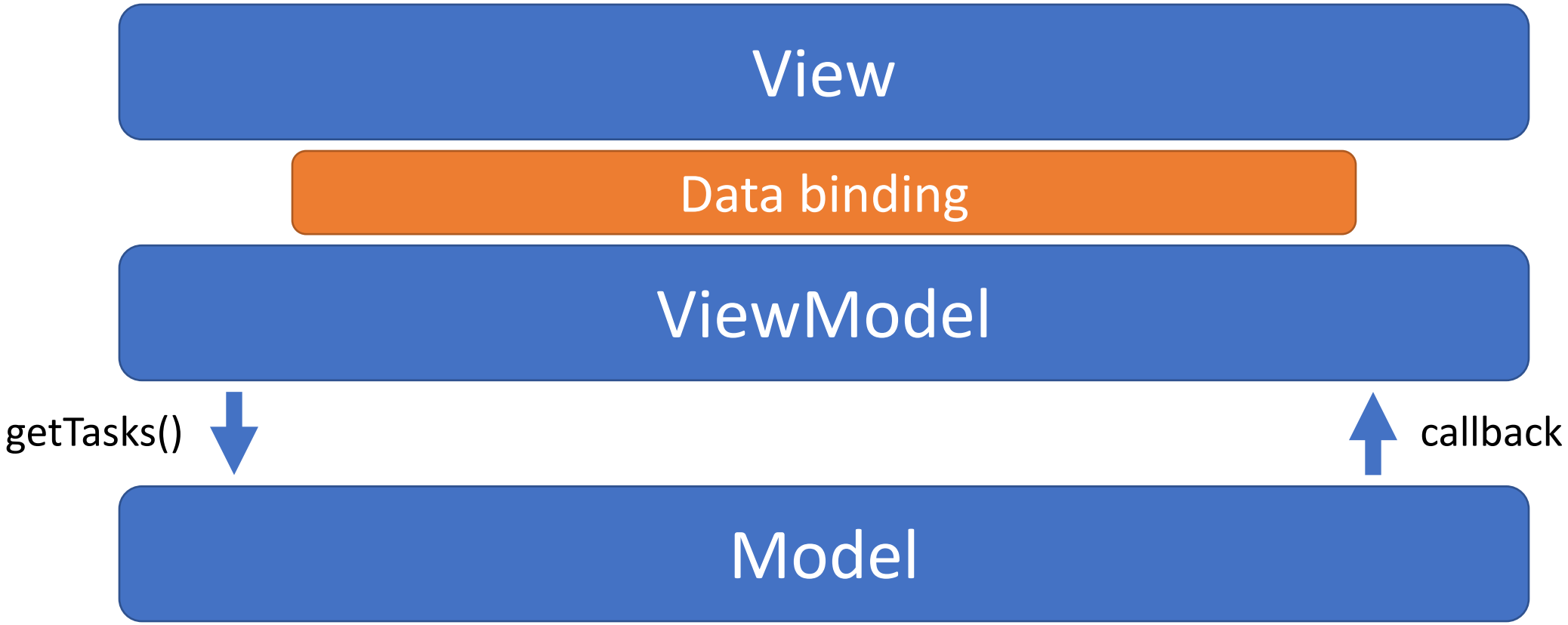


MVP vs. MVVM



https://github.com/googlesamples/android-architecture/blob/todo-mvp-clean

MVP vs. MVVM



Tips & tricks

RecyclerView

```
public class ViewHolder extends RecyclerView.ViewHolder
{
    private final ListItemBinding listItemBinding;

    public ViewHolder(ListItemBinding binding) {
        super(binding.getRoot());
        this.listItemBinding = binding;
    }

    …
}
```

RecyclerView

```
public class ViewHolder extends RecyclerView.ViewHolder
{
    private final ListItemBinding listItemBinding;

    …

    public void bind(User user) {
        listItemBinding.setUser(user);
        listItemBinding.executePendingBindings();
    }
}
```

RecyclerView

```
public class UserAdapter extends RecyclerView.Adapter {
    private List<User> users;

    …

    @Override
    public void onBindViewHolder(ViewHolder holder, int
                                                position) {
        User user = users.get(position);
        holder.bind(user);
    }
}
```

RecyclerView

```
public class MainActivity extends … {
    @Override
    protected void onCreate() {
        ActivityMainBinding binding = DataBindingUtil
            .setContentView(this, R.layout.activity_main);

        binding.users.setLayoutManager(
                              new LinearLayoutManager(this));
        binding.users.setAdapter(new UserAdapter(users));
    }
    …
}
```

RecyclerView advanced

```
@BindingAdapter("items")
public static <T> void setItems(RecyclerView
            recyclerView, Collection<T> items)
{
    BindingRecyclerViewAdapter<T> adapter =
        (BindingRecyclerViewAdapter<T>)
            recyclerView.getAdapter();
    adapter.setItems(items);
}
```

RecyclerView advanced

```
<android.support.v7.widget.RecyclerView
  …
  app:items="@{usersVM.users}"
  app:itemViewBinder="@{usersVM.itemViewBinder}"
  app:clickHandler="@{usersVM.click}"
  app:longClickHandler="@{usersVM.longClick}"
  />
```

https://github.com/radzio/android-data-binding-recyclerview

String formatting

```
<resources>
    <string name="greeting">Hello, %s</string>
</resources>
```

```
<TextView
    android:text="@{@string/greeting(`Michael`)}"
/>
```

String formatting

```
<resources>
    <string name="greeting">Hello, %s</string>
</resources>
```

```
<TextView
    android:text=
            "@{@string/greeting(user.firstName)}"
/>
```

Math in expressions

```
<TextView
    android:padding="@dimen/padding"
    android:padding="@{@dimen/padding}"
    android:padding="@{@dimen/padding * 2}"
    android:padding="@{@dimen/padding +
                                @dimen/padding}"
    android:padding="@{largeScreen ?
            @dimen/padding * 2 : @dimen/padding}"
/>
```

Binding conversions

```
@BindingConversion
public static int
convertBooleanToVisibility(boolean visible) {
    return visible ? View.VISIBLE : View.GONE;
}
```

```
<TextView
    …
    android:visibility="@{user.isAdult}"/>
```

Resources

- Official documentation
  https://developer.android.com/topic/libraries/data-binding/index.html
- George Mount
  https://medium.com/@georgemount007
- Advanced data binding (Google IO 2016)
  https://www.youtube.com/watch?v=DAmMN7m3wLU
- Data binding & RecyclerView
  https://github.com/radzio/android-data-binding-recyclerview

Data binding on Android

twitter.com/kevinpelgrims

kevinpelgrims.com