Tutorial 4: Activity Lifecycle

Objectives

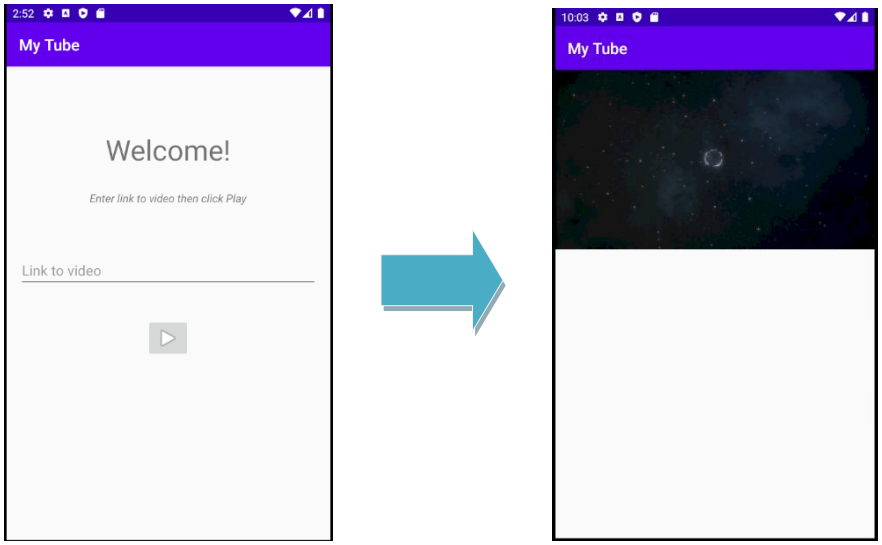
- Creating your first **multi-activity application** & navigating between activities
- Solving **view persistence problem** with mentioned but not practiced yet media widget: `VideoView`
- Understanding **Activity lifecycle** & ultilizing callback methods

Resources

- Tutorial instruction (this file)
- *[Optional]* Lecture slides & Lecture code example

Tutorial Exercises

In this tutorial, we will create an application named "My Tube", containing (2) activities. In the first activity, user enters link to video then click play, system starts the second activity & play the video with provided link.



Create a new project and complete the following exercises.

Exercise 1 – MainActivity: Enter link

- UI as mentioned below

- Functionality
    - User enters link link to video then click play
    - System navigates to ***Activity: Player*** to play video with entered link

Hint: for testing, when user click play button, let's display a Toast message to make sure that you got the link correctly before sending it to your next Activity.

- Adding a new activity for playing video named `PlayerActivity`

  Note:
    - Use ***File → New Activity → EmptyActivity***, this will automatically generate the layout file `res/layout/activity_player.xml` & update the `manifests/AndroidManifest.xml` file
    - Have a quick look on `manifests/AndroidManifest.xml` file

- Navigating to PlayerActivity

Hint: just remind

- Start a new activity from the current one

```
Intent intent = new Intent(MainActivity.this,
AnotherActivity.class);
startActivity(intent);
```

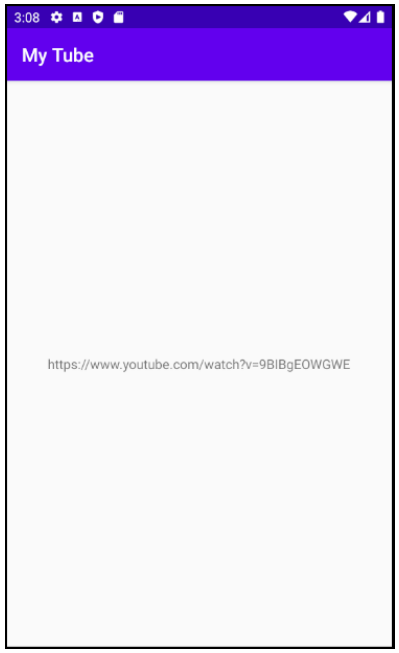
- Pass data to the target activity using Intent

```
// before startActivity
intent.putExtra("KEY", "mpr");
```

- Retrieve data in target activity from Intent

```
Intent intent = getIntent();
String key = intent.getStringExtra("KEY");
```

Hint: for testing, let's have a `TextView` to display the link to make sure that you passed the link correctly before playing it.

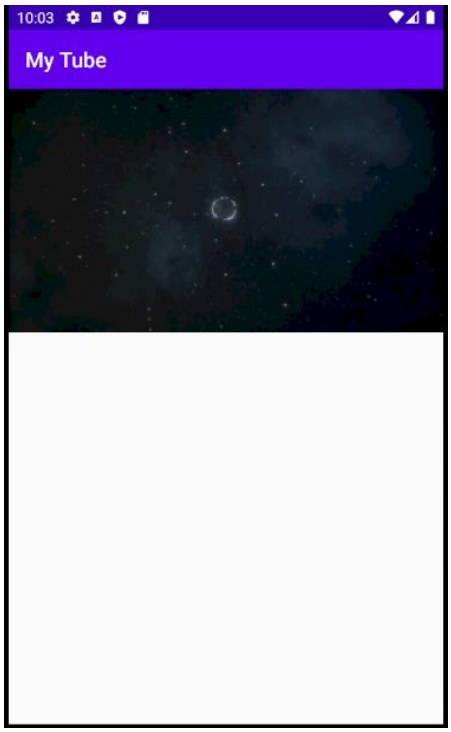


Exercise 2 – Activity: Player

- **IMPORTANT NOTE**: to play video from an external resource (URL), application need to have permission to access the Internet. In `AndroidManifest.xml` add this line just before `<application>`

```
<uses-permission android:name="android.permission.INTERNET" />
```

- UI as below



Note: the `MediaController` has position based on the container of the `VideoView`, so you may need to wrap it with a `LinearLayout`.

- Functionality (by built-in `MediaController`)
    - Play/ Pause
    - Seek forward/ backward

```
// example video
VideoView videoView = findViewById(R.id.videoView);
videoView.setVideoPath("https://download.ted.com/talks/ElonMusk_2017-480p.mp4");
videoView.setMediaController(new MediaController(this));
```

- Test video links
    - https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4
    - https://commondatastorage.googleapis.com/gtv-videos-bucket/sample/ElephantsDream.mp4
    - From https://test-videos.co.uk/
    - From https://www.ted.com/

- Optimize
    - Invoke `videoView.start()` in `onStart()` method
    - Avoid memory leak by invoking `videoView.stopPlayback()` in `onStop()` method

Exercise 3 – View persistence

Your application is now working nicely. However, try to rotate the screen. Observe what happen.

- What is the problem?
- Propose solution?

→ Maintaining the current position of `VideoView`

Hint: just remind

- Save state

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    outState.putString("KEY", "mpr");
}
```

- Restore state

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

    String key = savedInstanceState.getString("KEY");
}
```

- How to get current video playback position?

```
int currentPositionInMilliseconds = videoView.getCurrentPosition();
```

- & How to play video from some position?

```
// Display first frame rather than a black screen
videoView.seekTo(1);
```

- Does it works? 😉

→ Let's use `Log` to observe the current playback position in method

- `onSaveInstanceState()`

→ Let's use `Log` to observe the current playback position in different activity lifecycle methods

- `onPause()`
- `onStop()`

→ Solution: use a variable to store the current playback position

More about `VideoView`?

https://developer.android.com/codelabs/advanced-android-training-video-view

If you finish them all...

If you finish all the exercises, you can improve the apps above or your apps (with better input styles, images, functionality, handing exceptions…)

Posting them on our Facebook group with *#I'm_Android_developer* for discussion mark. **Note**: feel free if it somehow not so special – you know, we are all beginners :D

Good bye!