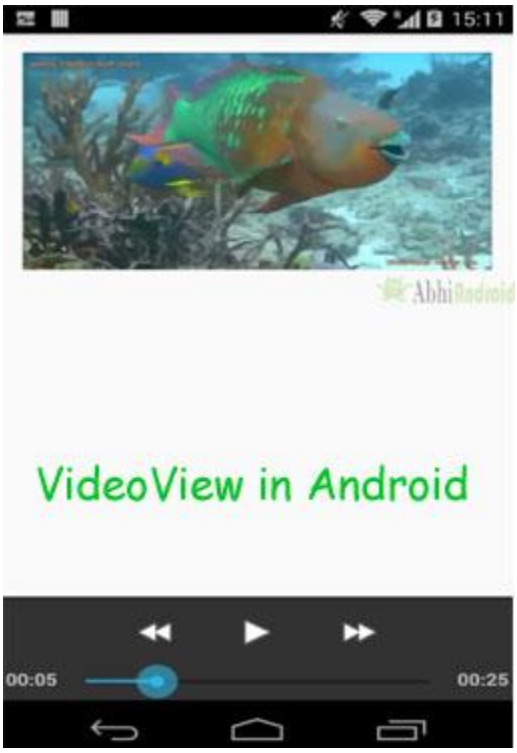
VideoView Tutorial With Example In Android Studio

In Android, VideoView is used to display a video file. It can load images from various sources (such as content providers or resources) taking care of computing its measurement from the video so that it can be used for any layout manager, providing display options such as scaling and tinting.



Important Note: VideoView does not retain its full state when going into the background. In particular it does not restore the current play position and play state. Applications should save and restore these in onSaveInstanceState(Bundle) and onRestoreInstanceState(Bundle).

VideoView code In XML Android:

```
<VideoView
android:id="@+id/simpleVideoView"
android:layout_width="fill_parent"
android:layout_height="fill_parent" />
```

Table Of Contents

Methods Used in VideoView:

Let’s we discuss some important methods of VideoView that may be called in order to manage the playback of video:

1. setVideoUri(Uri uri): This method is used to set the absolute path of the video file which is going to be played. This method takes a Uri object as an argument.

Below we set the uri of video which is saved in Android Studio:

Step 1: Create a new directory in res folder and name it raw

Step 2: Save a video name fishvideo in raw folder

Step 3: Now use the below code to set the path for the video using setVideoUri() method in VideoView.

```
// initiate a video view
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
simpleVideoView.setVideoURI(Uri.parse("android.resource://" +
getPackageName() + "/" + R.raw.fishvideo));
```

Setting Video From Online Web Source:

Step 1: First add internet permision in Manifest.xml file. We will need to add this so as to access the video through Internet. Open AndroidManifest.xml and add the below code

```
<!--Add this before application tag in AndroidManifest.xml-->
<uses-permission android:name="android.permission.INTERNET" />
```

Step 2: Add the basic VideoVideo XML code in activity_main.xml or activity.xml

Step 3: Use the below code to access the Video from our website

```
package coderbd.com.videofromwebsource;

import android.app.ProgressDialog;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.VideoView;
```

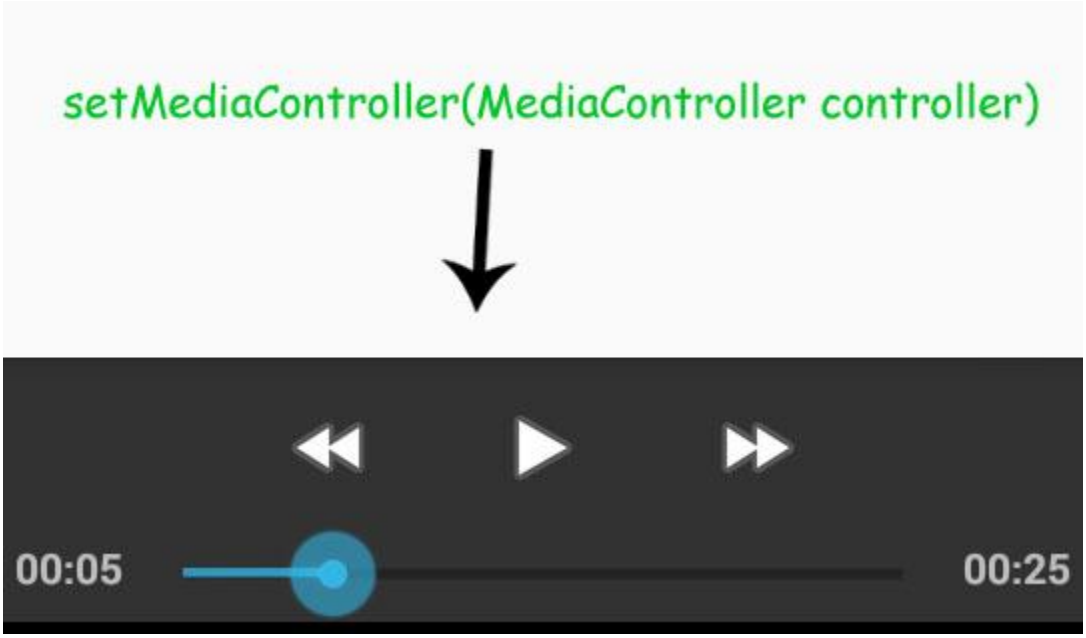
```
public class MainActivity extends AppCompatActivity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            Uri uri = Uri.parse("http://abhiandroid-8fb4.kxcdn.com/ui/wp-
content/uploads/2016/04/videoviewtestingvideo.mp4");
            VideoView simpleVideoView = (VideoView)
findViewById(R.id.simpleVideoView); // initiate a video view
            simpleVideoView.setVideoURI(uri);
            simpleVideoView.start();
    }
}
```

2. setMediaController(MediaController controller): This method of VideoView is used to set the controller for the controls of video playback.

Below we show how to set the media controller object for a video view.

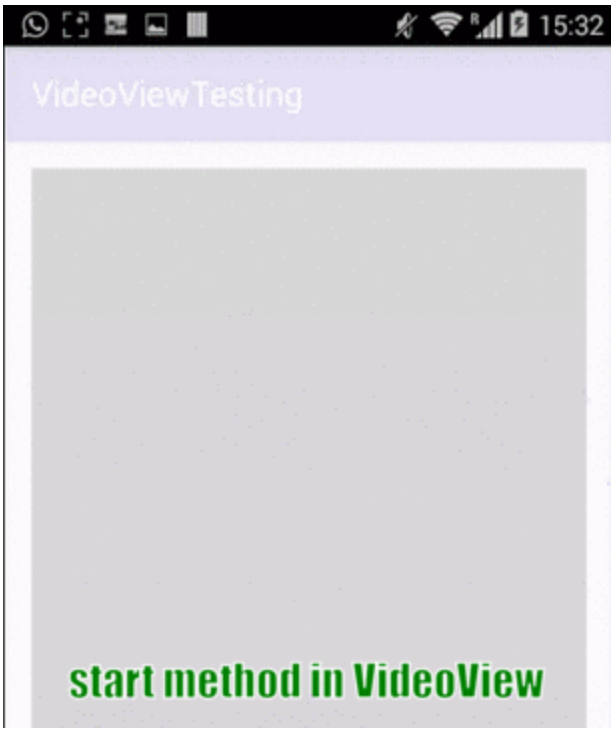
```
// create an object of media controller
MediaController mediaController = new MediaController(this);
// initiate a video view
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// set media controller object for a video view
simpleVideoView.setMediaController(mediaController);
```



3. start(): This method of VideoView is used to start the playback of video file.

Below we show how to start a video in video view.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
simpleVideoView.start(); // start a video
```



4. pause(): This method of video view is used to pause the current playback.

Below we shows how to pause a video in video view.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
simpleVideoView.pause(); // pause a video
```

5. canPause(): This method will tell whether VideoView is able to pause the video. This method returns a Boolean value means either true or false. If a video can be paused then it returns true otherwise it returns false.

Below we checks whether a video is able to pause or not.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
Boolean canPauseVideo = simpleVideoView.canPause(); // check whether a video
is able to pause or not
```

6. canSeekForward(): This method will tell whether video is able to seek forward. This method returns a Boolean value i.e. true or false. If a video can seek forward then it returns true otherwise it returns false.

Below we checks whether a video is able to seek forward or not.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
Boolean canSeekForward = simpleVideoView.canSeekForward(); // checks whether
a video view is able to seek forward  or not
```

7. canSeekBackward(): This method will tell whether video is able to seek backward. This method returns a Boolean value i.e. true or false. If a video can seek backward then it return true otherwise it return false.

Below we checks whether a video is able to seek backward or not.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
Boolean canSeekBackword = simpleVideoView.canSeekBackward(); // checks
whether a video view is able to seek backword or not
```

8. getDuration(): This method is used to get the total duration of VideoView. This methods return an integer value.

Below we get the total duration of a video view.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
int duration =simpleVideoView.getDuration();// get the total duration of the
video
```

9. getCurrentPosition(): This method is used to get the current position of playback. This method returns an integer value.

Below we get the current position of a playback.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
int currentPosition = simpleVideoView.getCurrentPosition(); // get the
current position of the video play back
```

10. isPlaying(): This method tells whether a video is currently playing or not. This method returns a Boolean value. It returns true if video is playing or false if it's not.

Below we check whether a video view is currently playing or not

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
Boolean isPlaying = simpleVideoView.isPlaying(); // check whether a video
view is currently playing or not
```

11. stopPlayback(): This method of VideoView is used to stop the video playback.

Below we show how to stop a video in video view.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
simpleVideoView.stopPlayback(); // stop a video
```

12. setOnPreparedListener(MediaPlayer.OnPreparedListener): This is a listener which allows a callback method to be called when the video is ready to play.

Below we show the use of setOnPreparedListener event of a video view.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view

// perform set on prepared listener event on video view
simpleVideoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
@Override
public void onPrepared(MediaPlayer mp) {

// do something when video is ready to play

}
});
```

13. setOnErrorListener(MediaPlayer.OnErrorListener): This listener allows a callback method to be called when an error occurs during the video playback.

Below we show the use of setOnErrorListener event of a video view.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view

// perform set on error listener event on video view
simpleVideoView.setOnErrorListener(new MediaPlayer.OnErrorListener() {
@Override
public boolean onError(MediaPlayer mp, int what, int extra) {


// do something when an error is occur during the video playback
return false;
```

```
}
});
```

14. setOnCompletionListener(MediaPlayer.OnCompletionListener): This listener allow a callback method to be called when the end of the video is reached.

Below we shows the use of setOnCompletionListener event of a video view.

```
VideoView simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);
// initiate a video view
// perform set on completion listener event on video view
simpleVideoView.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
@Override
public void onCompletion(MediaPlayer mp) {
// do something when the end of the video is reached
}
});
```

MediaController In VideoView

MediaController is a class which is used to provide the controls for the video playback. If a video is simply played using the VideoView class then the user will not be given any control over the playback of the video which will run until the end of the video is reached. This issue can be addressed by attaching an instance of the MediaController class to the VideoView instance. The Media Controller will then provide a set of controls allowing the user to manage the playback (such as seeking backwards/forwards and pausing in the video timeline).

Methods Of MediaController:

Let's we discuss some important methods of MediaController class that may be called in order to control for the playback.

1. setAnchorView(View view): setAnchorView is used to designates the view to which the controller is to be anchored. This controls the location of the controls on the screen.

Below we show how to use setanchorview() method of a MediaController class.

```
MediaController mediaController = new MediaController(this); // create an
object of media controller
mediaController.setAnchorView(simpleVideoView); // set anchor view for video
view
```

2. show(): This method is used to show the controller on the screen.

Below we show the controller on the screen.

```
MediaController mediaController = new MediaController(this); // create an
object of media controller
mediaController.show(); // show the controller on the screen
```

3. show(int timeout): This method is used to set the time to show the controller on the screen.

Below we set the time for showing the controller on the screen.

```
MediaController mediaController = new MediaController(this); // create an
object of media controller
mediaController.show(500); // set the time to show the controller on the
screen
```

4. hide(): This method is used to hide the controls from the screen.

Below we hide the control from the screen

```
MediaController mediaController = new MediaController(this); // create an
object of media controller
mediaController.hide(); // hide the control from the screen
```

5. isShowing(): This method returns a Boolean value indicating whether the controls are currently visible to the user or not.

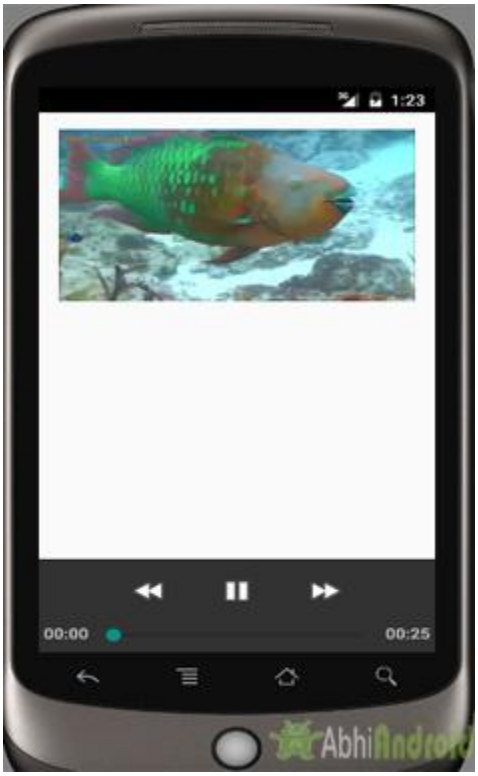
Below we checks whether the controls are currently visible or not.

```
MediaController mediaController = new MediaController(this); // create an
object of media controller
Boolean isShowing = mediaController.isShowing(); // checks whether the
controls are currently visible or not
```

VideoView Example In Android Studio:

Below is the example of VideoView in Android in which we play a video in a video view by using Media Controller and perform set on error and completion listener events and display Toast when the video is completed or an error occur while playing thee video.

In this example we create a folder named raw in our project and store the video file in that folder and then set the uri for the video in our activity in which we display the video view.



Step 1 create new project in Android Studio and name it **VideoViewExample**

Step 2: Open res -> layout -> xml (or) main.xml and add following code :

In this step we open an xml file and add the code to display a VideoView in our activity.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <VideoView
        android:id="@+id/simpleVideoView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />


</RelativeLayout>
```

Step 3: Open src -> package -> MainActivity.java

In this step we open MainActivity and add the code to initiate the video view and create an object of MediaController to control the video playback.

In this class we also set the uri for the video and perform set on error and completion listener events and display st message when video is completed or an error is occur while playing thee video.

Also make sure to create a new directory in res folder and name it raw. Save a video name fishvideo in raw folder. We will be setting path to this Video in setVideoURI() method.

```
package example. coderbd.videoviewexample;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.res.Configuration;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnPreparedListener;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.MediaController;
import android.widget.Toast;
import android.widget.VideoView;

public class MainActivity extends Activity {

    VideoView simpleVideoView;
    MediaController mediaControls;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Find your VideoView in your video_main.xml layout
        simpleVideoView = (VideoView) findViewById(R.id.simpleVideoView);

        if (mediaControls == null) {
            // create an object of media controller class
            mediaControls = new MediaController(MainActivity.this);
            mediaControls.setAnchorView(simpleVideoView);
        }
        // set the media controller for video view
        simpleVideoView.setMediaController(mediaControls);
        // set the uri for the video view
        simpleVideoView.setVideoURI(Uri.parse("android.resource://" +
getPackageName() + "/" + R.raw.fishvideo));
        // start a video
        simpleVideoView.start();

        // implement on completion listener on video view
        simpleVideoView.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
```

```
                Toast.makeText(getApplicationContext(), "Thank You...!!!",
Toast.LENGTH_LONG).show(); // display a toast when an video is completed
            }
        });
        simpleVideoView.setOnErrorListener(new MediaPlayer.OnErrorListener()
{
            @Override
            public boolean onError(MediaPlayer mp, int what, int extra) {
                Toast.makeText(getApplicationContext(), "Oops An Error Occur
While Playing Video...!!!", Toast.LENGTH_LONG).show(); // display a toast
when an error is occured while playing an video
                return false;
            }
        });
    }

}
```