

# Creating A Titanium Module using Claude

Prompt:

**I'm trying to wrap this library <https://github.com/PierfrancescoSoffritti/android-youtube-player> into a Titanium module. Can you create the code to do this?**

I'll help you create a Titanium module to wrap the android-youtube-player library. I'll walk you through creating the necessary files and code for the module.

I've created a basic Titanium module that wraps the android-youtube-player library. Here's a breakdown of the key components:

1. The manifest file defines the module's basic information and requirements
2. build.gradle includes the YouTube Player library dependency
3. YouTubePlayerModule.java is the main module class
4. YouTubePlayerProxy.java handles the creation and management of the YouTube player view

To use this module:

1. Create a new Titanium module project
2. Copy these files into the appropriate directories
3. Add the android-youtube-player dependency to your build.gradle
4. Build the module using:

```
```bash
ti build -p android --build-only
```
```

The module can then be used in your Titanium app as shown in the example code.

Some additional features you might want to add:

1. Event handlers for player states (playing, paused, ended)
2. Methods for controlling playback (play, pause, seek)
3. Quality control options
4. Playlist support

Would you like me to expand on any of these additional features or would you like to know about implementing specific functionality?

**I'm getting this error for YouTubePlayerProxy.java when I try to build: error: incompatible types: Activity cannot be converted to TiViewProxy youTubePlayerView = new YouTubePlayerView(activity);**

Ah, I see the issue. There's a naming conflict between our inner class `YouTubePlayerView` and the YouTube player library's `YouTubePlayerView` class. Let's fix this by renaming our inner class and properly casting the activity.

I've made the following changes to fix the compilation error:

1. Renamed our inner class from `YouTubePlayerView` to `TiYouTubePlayerView` to avoid the naming conflict
2. Used the fully qualified name for the YouTube player library's `YouTubePlayerView` class
3. Updated all references to the class name throughout the file

The error occurred because there was ambiguity between our custom view class and the library's view class. By using the fully qualified name for the library class and renaming our custom class, we've resolved this conflict.

Try building the module again with these changes. Let me know if you encounter any other issues!

**I'm getting another error when building: error: method getString in class KrollDict cannot be applied to given types; String videold = proxy.getProperties().getString("videold", ""); required: String found: String,String reason: actual and formal argument lists differ in length**

Ah, I see the issue. The `getString()` method in `KrollDict` only accepts a single parameter, not a default value like some other APIs. Let's fix this by updating how we handle the video ID property.

I've made two key changes:

1. Removed the default value parameter from `getString()` since `KrollDict` doesn't support it
2. Added proper null checking for the `videoId` value

The `KrollDict.getString()` method only takes the key name as a parameter and returns null if the key doesn't exist. We now properly handle this by checking both for null and empty string values.

Try building again with these changes. Let me know if you encounter any other issues!

**This works. How can I change it so it doesn't auto play on load**

I'll modify the code to prevent autoplay when loading a video. We'll need to use the `cueVideo` method instead of `loadVideo`, as `loadVideo` automatically starts playing the video.

I've changed `loadVideo()` to `cueVideo()` in the `onReady` listener. The `cueVideo()` method will load the video but won't start playing it automatically. The user will need to click the play button to start the video.

You can also add a method to explicitly load and play a video if needed. Would you like me to add that functionality?