Polyv Flutter Media Player - 项目文档

> 基于 保利威 iOS 点播播放器 SDK 封装的 Flutter 视频播放器插件，支持 iOS 和 Android 双平台。

目录

- 项目概述
- 架构设计
- 功能特性
- 项目结构
- 快速集成
- API 参考
- UI 组件
- 平台原生层
- 业务服务模块
- Demo 应用
- 开发指南
- 常见问题

项目概述

polyv-flutter-media-player-demo 是一个 Flutter Plugin 项目，将保利威原生播放器 SDK（iOS `PolyvMediaPlayerSDK`、Android `media-player-full`）封装为统一的 Dart API，供 Flutter 应用跨平台调用。

项目分为两层：

层级	目录	职责
Plugin（插件层）	`polyv_media_player/`	播放核心能力 + 可共享业务服务模块，无业务 UI

层级	目录	职责
Demo App（示例层）	`example/`	完整 UI 实现（播放器皮肤、控制栏、弹幕、字幕等），供客户参考复制

技术栈

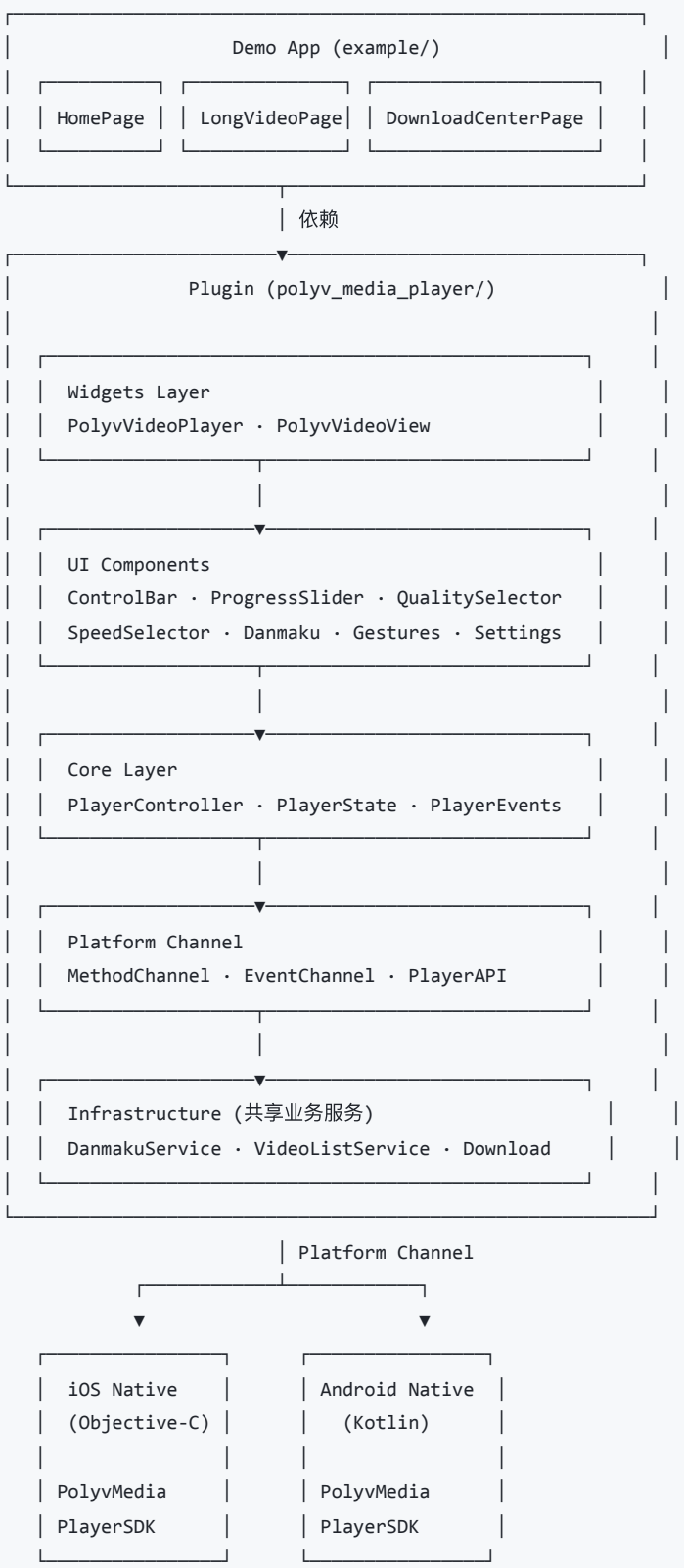
类别	技术
框架	Flutter (Dart SDK ^3.9.0)
状态管理	Provider (`ChangeNotifier` 模式)
iOS 原生 SDK	`PolyvMediaPlayerSDK ~> 2.7.2` (Objective-C)
Android 原生 SDK	`net.polyv.android:media-player-full:2.7.2` (Kotlin)
跨平台通信	Flutter Platform Channel (MethodChannel + EventChannel)
依赖	`http` , `crypto` , `shared_preferences` , `provider`

底层 SDK 参考

本插件封装的保利威原生 SDK 文档：

- **iOS 点播 SDK 文档**: https://help.polyv.net/index.html#/vod/ios_player_sdk/
- **iOS SDK Demo**: https://github.com/polyv/polyv-ios-vod-sdk
- **Android SDK**: 通过阿里云私有 Maven 仓库分发
- **开发者中心**: https://www.polyv.net/dev/

架构设计



核心设计原则

1. **Plugin 只提供核心能力**：播放控制、状态管理、Platform Channel 封装

2. **业务逻辑在 Dart 层统一实现**：不在原生层调用 Polyv 业务 HTTP 接口
3. **原生层只封装播放器 SDK**：暴露底层能力（播放、下载、字幕等），不包含业务决策
4. **清晰度切换进度恢复**：由原生层负责（Dart 层仅触发切换、消费事件）

功能特性

播放核心

功能	说明
视频播放	通过 VID 播放保利威点播视频
播放控制	播放、暂停、停止、重播
进度控制	Seek 到指定位置，进度实时回调
倍速播放	支持 0.5x ~ 2.0x 变速播放
清晰度切换	多码率切换（流畅、高清、超清等），自动恢复进度
字幕系统	多语言字幕轨道、双语字幕、字幕开关
离线播放	自动检测已下载视频，优先本地播放
播放进度记忆	自动保存和恢复上次播放进度

视频下载

功能	说明
下载管理	创建、暂停、继续、重试、删除下载任务
状态持久化	App 重启后自动同步下载状态
进度回调	实时下载进度通知

弹幕系统

功能	说明
弹幕渲染	支持滚动弹幕层
弹幕服务接口	可插拔的 DanmakuService / DanmakuSendService
HTTP 弹幕	内置保利威弹幕 API 实现
弹幕设置	透明度、速度、字体大小等
弹幕发送	全屏模式下的弹幕输入框

交互体验

功能	说明
手势控制	滑动调节进度/音量/亮度
双击全屏	双击播放区域切换全屏
锁屏模式	全屏时锁定控制栏
控制栏自动隐藏	可配置隐藏延迟
横竖屏适配	全屏/非全屏布局自适应

项目结构

```
polyv-flutter-media-player-demo/          # Git 仓库根目录
├── polyv_media_player/                   # Flutter Plugin 插件
│   ├── lib/
│   │   ├── polyv_media_player.dart       # 主入口（导出所有公共 API）
│   │   ├── core/                         # 核心层
│   │   │   ├── player_controller.dart    # 播放器控制器（ChangeNotifier）
│   │   │   ├── player_state.dart         # 播放器状态模型
│   │   │   ├── player_events.dart        # 事件类型定义
│   │   │   ├── player_event_parser.dart  # 原生事件解析器
│   │   │   ├── player_exception.dart     # 异常类
│   │   │   ├── player_config.dart        # 配置类
│   │   │   ├── subtitle_selection_policy.dart  # 字幕自动选择策略
│   │   │   ├── offline_playback_decider.dart   # 离线播放决策
│   │   │   └── system_locale_provider.dart     # 系统语言检测
│   │   ├── platform_channel/             # Platform Channel 封装
│   │   │   ├── player_api.dart           # Channel 名称 + 方法常量
│   │   │   ├── method_channel_handler.dart  # MethodChannel 处理
│   │   │   └── event_channel_handler.dart   # EventChannel 处理
│   │   ├── services/                     # 服务层
│   │   │   ├── polyv_config_service.dart # 账号配置管理
│   │   │   ├── player_initializer.dart   # 播放器初始化
│   │   │   ├── video_progress_service.dart  # 播放进度记忆
│   │   │   └── subtitle_preference_service.dart  # 字幕偏好
│   │   ├── infrastructure/               # 基础设施（共享业务服务）
│   │   │   ├── danmaku/                  # 弹幕系统
│   │   │   │   ├── danmaku_model.dart    # 弹幕数据模型
│   │   │   │   └── danmaku_service.dart  # 弹幕服务接口 + 实现
│   │   │   ├── download/                 # 下载管理
│   │   │   │   ├── download_task.dart    # 下载任务模型
│   │   │   │   ├── download_task_status.dart  # 下载状态枚举
│   │   │   │   ├── download_state_manager.dart  # 下载状态管理
│   │   │   │   ├── download_event_handler.dart  # 下载事件处理
│   │   │   │   └── download_native_repository.dart  # 原生下载能力封装
│   │   │   ├── video_list/               # 视频列表
│   │   │   │   ├── video_list_models.dart  # 视频列表数据模型
│   │   │   │   ├── video_list_service.dart  # 视频列表服务
│   │   │   │   └── video_list_api_client.dart  # API 客户端
│   │   │   └── polyv_api_client.dart     # Polyv API 通用客户端
│   │   ├── widgets/                      # Widget 层
│   │   │   ├── polyv_video_player.dart   # 全功能播放器 Widget
│   │   │   └── polyv_video_view.dart     # 原生视频视图（PlatformView）
│   │   ├── ui/                           # 内置 UI 组件
│   │   │   ├── control_bar.dart          # 控制栏
│   │   │   ├── control_bar_state_machine.dart  # 控制栏状态机
│   │   │   ├── player_colors.dart        # 播放器颜色常量
│   │   │   ├── double_tap_detector.dart  # 双击检测
│   │   │   ├── progress_slider/          # 进度条组件
│   │   │   ├── quality_selector/         # 清晰度选择器
│   │   │   ├── speed_selector/           # 倍速选择器
│   │   │   ├── subtitle_toggle.dart      # 字幕开关
│   │   │   ├── danmaku/                  # 弹幕 UI 组件
│   │   │   │   ├── danmaku_layer.dart    # 弹幕渲染层
```

```
│   │   │   │   ├── danmaku_toggle.dart   # 弹幕开关
│   │   │   │   ├── danmaku_input_overlay.dart  # 弹幕输入浮层
│   │   │   │   └── danmaku_settings.dart # 弹幕设置面板
│   │   │   ├── gestures/                 # 手势系统
│   │   │   │   ├── player_gesture_controller.dart
│   │   │   │   ├── player_gesture_detector.dart
│   │   │   │   └── seek_preview_overlay.dart
│   │   │   └── settings_menu/            # 设置菜单
│   │   └── utils/                        # 工具类
│   │       └── plv_logger.dart           # 日志工具
│   ├── ios/                              # iOS 原生代码
│   │   ├── Classes/
│   │   │   ├── PolyvMediaPlayerPlugin.m  # 插件入口
│   │   │   ├── PLVFlutterMethodRouter.m  # Method 路由分发
│   │   │   ├── PLVFlutterPlayerSession.m # 播放器会话管理
│   │   │   ├── PLVFlutterEventEmitter.m  # 事件发射器
│   │   │   ├── PLVFlutterDownloadMonitor.m # 下载监控
│   │   │   ├── PLVFlutterSubtitleCoordinator.m  # 字幕协调
│   │   │   ├── PLVVideoViewFactory.m     # PlatformView 工厂
│   │   │   └── ...字幕解析相关文件
│   │   └── polyv_media_player.podspec    # CocoaPods 配置
│   ├── android/                          # Android 原生代码
│   │   └── src/main/kotlin/
│   │       ├── PolyvMediaPlayerPlugin.kt # 插件入口
│   │       ├── MethodRouter.kt           # Method 路由
│   │       ├── PlayerCoordinator.kt      # 播放协调
│   │       ├── DownloadCoordinator.kt    # 下载协调
│   │       ├── SubtitleCoordinator.kt    # 字幕协调
│   │       ├── PlaybackEventEmitter.kt   # 播放事件发射
│   │       ├── DownloadEventEmitter.kt   # 下载事件发射
│   │       └── PolyvVideoViewFactory.kt  # PlatformView 工厂
│   └── test/                             # 单元测试
├── example/                              # Demo App
│   ├── lib/
│   │   ├── main.dart                     # 应用入口
│   │   ├── config/
│   │   │   └── app_config.dart           # 账号配置（环境变量注入）
│   │   ├── pages/
│   │   │   ├── home_page.dart            # 首页（长视频/下载中心入口）
│   │   │   ├── long_video_page.dart      # 长视频播放页
│   │   │   └── download_center/          # 下载中心
│   │   │       ├── download_center_page.dart
│   │   │       └── downloading_task_item.dart
│   │   └── player_skin/
│   │       └── video_list/               # 视频列表组件
│   └── pubspec.yaml
└── docs/                                 # 项目文档
```

快速集成

环境要求

要求	版本
Flutter	>= 3.3.0
Dart SDK	^3.9.0
iOS	>= 13.0
Android minSdk	>= 21

步骤 1: 添加依赖

将 `polyv_media_player` 目录复制到项目中（与 `lib/` 平级），然后在 `pubspec.yaml` 中添加：

```
dependencies:
  polyv_media_player:
    path: polyv_media_player
```

执行：

```
flutter pub get
```

步骤 2: iOS 配置

在 iOS 项目的 `Podfile` 中确保平台版本 >= 13.0：

```
platform :ios, '13.0'
```

然后执行：

```
cd ios && pod install
```

插件会自动通过 CocoaPods 引入以下依赖：

```
PolyvMediaPlayerSDK (~> 2.7.2)
PLVFoundationSDK/AbstractBase (~> 1.30.2)
PLVFDB (~> 1.0.5)
PLVLOpenSSL (~> 1.1.12101)
SSZipArchive (~> 2.0)
```

步骤 3: 初始化 SDK

在 `main.dart` 中初始化保利威账号配置：

```
import 'package:flutter/material.dart';
import 'package:polyv_media_player/polyv_media_player.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await PolyvMediaPlayer.initialize(
    userId: 'your_user_id',
    secretKey: 'your_secret_key',
    readToken: 'your_read_token',   // 可选
    writeToken: 'your_write_token', // 可选
  );

  runApp(const MyApp());
}
```

> 账号配置信息可在 保利威后台 注册后获取。

步骤 4: 使用播放器

最简用法，一行代码即可播放视频：

```
PolyvVideoPlayer(
  vid: 'your_video_id',
  autoPlay: true,
)
```

API 参考

SDK 初始化

```
await PolyvMediaPlayer.initialize(
  userId: String,     // 必填：保利威用户 ID
  secretKey: String,  // 必填：密钥
  readToken: String?, // 可选：读取 Token
  writeToken: String?,// 可选：写入 Token
);

// 检查是否已初始化
bool initialized = PolyvMediaPlayer.isInitialized;

// 获取用户 ID
String userId = await PolyvMediaPlayer.userId;
```

PlayerController

`PlayerController` 是播放器的核心控制类，继承自 `ChangeNotifier`，通过 Provider 模式驱动 UI 更新。

播放控制

方法 / 属性	说明
`loadVideo(vid, {autoPlay})`	加载视频
`play()`	播放
`pause()`	暂停
`stop()`	停止
`replay()`	重播（回到开头重新播放）
`seekTo(position)`	跳转到指定位置（毫秒）

设置

方法	说明
`setPlaybackSpeed(speed)`	设置倍速（0.5 ~ 2.0）

方法	说明
`setQuality(index)`	切换清晰度
`setSubtitle(index)`	设置字幕（-1 关闭）
`toggleSubtitle()`	切换字幕开关
`setSubtitleWithKey({enabled, trackKey})`	通过 key 设置字幕

状态

属性	类型	说明
`state`	`PlayerState`	当前播放器完整状态
`qualities`	`List<QualityItem>`	可用清晰度列表
`availableSubtitles`	`List<SubtitleItem>`	可用字幕列表
`effectiveIsPlaying`	`bool`	播放状态（推荐用于 UI）

生命周期

方法	说明
`dispose()`	释放资源（必须在 Widget dispose 时调用）

PlayerState

```
class PlayerState {
  PlayerLoadingState loadingState;  // idle/loading/prepared/playing/paused/buffering/completed/error
  int position;           // 当前位置（毫秒）
  int duration;           // 总时长（毫秒）
  int bufferedPosition;   // 缓冲位置（毫秒）
  double playbackSpeed;   // 播放速度
  String? errorMessage;   // 错误信息
  String? vid;            // 当前视频 VID
  bool subtitleEnabled;   // 字幕是否开启
  String? currentSubtitleId;  // 当前字幕 ID
  List<SubtitleItem> availableSubtitles;  // 可用字幕列表
}
```

事件类型

事件	说明	数据
`stateChanged`	播放状态变化	新状态 (playing/paused/buffering...)
`progress`	进度更新	position, duration, bufferedPosition
`error`	错误	code, message
`qualityChanged`	清晰度变化	qualities[], currentIndex
`subtitleChanged`	字幕变化	subtitles[], currentIndex
`playbackSpeedChanged`	倍速变化	speed
`completed`	播放完成	-

Platform Channel 常量

```
MethodChannel: com.polyv.media_player/player
EventChannel:  com.polyv.media_player/events
DownloadEvent: com.polyv.media_player/download_events
```

PolyvVideoPlayer Widget

参数	类型	默认值	说明
`vid`	`String`	必填	视频 ID
`autoPlay`	`bool`	`true`	是否自动播放
`showControls`	`bool`	`true`	是否显示控制栏
`enableDanmaku`	`bool`	`true`	是否启用弹幕
`enableGestures`	`bool`	`true`	是否启用手势
`enableDoubleTapFullscreen`	`bool`	`true`	是否启用双击全屏

参数	类型	默认值	说明
`isFullscreen`	`bool`	`false`	是否全屏模式
`showLockButton`	`bool`	`false`	全屏时显示锁屏按钮
`showDanmakuSend`	`bool`	`false`	全屏时显示弹幕发送
`showTopBar`	`bool`	`false`	全屏时显示顶部栏
`videoTitle`	`String?`	`null`	全屏顶部栏标题
`aspectRatio`	`double`	`16/9`	视频宽高比
`backgroundColor`	`Color`	`Colors.black`	背景色
`autoHideDuration`	`Duration`	`3s`	控制栏自动隐藏时长
`controller`	`PlayerController?`	`null`	外部控制器
`danmakuService`	`DanmakuService?`	`null`	弹幕数据服务
`danmakuSendService`	`DanmakuSendService?`	`null`	弹幕发送服务
`danmakuHeightFactor`	`double`	`0.6`	弹幕显示区域高度比例
`onFullscreenChanged`	`Function?`	`null`	全屏切换回调
`onLoaded`	`Function?`	`null`	加载完成回调
`onPlayingChanged`	`Function?`	`null`	播放状态变化回调
`onCompleted`	`Function?`	`null`	播放完成回调
`onError`	`Function?`	`null`	错误回调

UI 组件

插件内置了完整的播放器 UI 组件库，可独立使用或组合使用。

可用组件

组件	导入路径	说明
`ControlBar`	`ui/control_bar.dart`	完整控制栏（进度条 + 播放按钮 + 倍速 + 清晰度）
`ProgressSlider`	`ui/progress_slider/`	进度条组件（含缓冲进度）
`QualitySelector`	`ui/quality_selector/`	清晰度选择器
`SpeedSelector`	`ui/speed_selector/`	倍速选择器
`SubtitleToggle`	`ui/subtitle_toggle.dart`	字幕开关
`DanmakuLayer`	`ui/danmaku/danmaku.dart`	弹幕渲染层
`DanmakuToggle`	`ui/danmaku/danmaku_toggle.dart`	弹幕开关
`DanmakuInputOverlay`	`ui/danmaku/danmaku_input_overlay.dart`	弹幕发送输入框
`DanmakuSettings`	`ui/danmaku/danmaku_settings.dart`	弹幕设置面板
`SettingsMenu`	`ui/settings_menu/`	设置菜单（清晰度 + 倍速 + 弹幕）
`PlayerGestureDetector`	`ui/gestures/gestures.dart`	手势检测（滑动进度/音量/亮度）
`SeekPreviewOverlay`	`ui/gestures/seek_preview_overlay.dart`	Seek 预览浮层
`PlayerColors`	`ui/player_colors.dart`	播放器颜色常量

自定义 UI 示例

使用底层组件组装自定义播放器：

```
class CustomVideoPage extends StatefulWidget {
  @override
  State<CustomVideoPage> createState() => _CustomVideoPageState();
}

class _CustomVideoPageState extends State<CustomVideoPage> {
  late final PlayerController _controller;

  @override
  void initState() {
    super.initState();
    _controller = PlayerController();
    _controller.loadVideo('your_video_id');
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: Stack(
        children: [
          // 原生视频视图
          const PolyvVideoView(),
          // 自定义控制栏
          Align(
            alignment: Alignment.bottomCenter,
            child: ControlBar(controller: _controller),
          ),
        ],
      ),
    );
  }
}
```

平台原生层

iOS 原生架构

iOS 原生代码采用模块化拆分，每个组件职责单一：

组件	文件	职责
Plugin 入口	`PolyvMediaPlayerPlugin.m`	持有 Flutter registrar/channel，注册 Plugin
Method 路由	`PLVFlutterMethodRouter`	MethodChannel 方法分发
播放器会话	`PLVFlutterPlayerSession`	播放器实例生命周期管理、SDK 调用封装
事件发射	`PLVFlutterEventEmitter`	统一封装 player/download EventChannel 事件发送
下载监控	`PLVFlutterDownloadMonitor`	下载状态轮询与下载事件发送
字幕协调	`PLVFlutterSubtitleCoordinator`	字幕轨道事件、label 维护
视频视图	`PLVVideoViewFactory`	PlatformView 工厂，创建原生视频渲染视图
字幕解析	`PLVVodMediaSubtitleParser` 等	SRT/ASS 字幕文件解析

iOS 依赖（通过 CocoaPods）

```
s.dependency 'PolyvMediaPlayerSDK', '~> 2.7.2'
s.dependency 'PLVFoundationSDK/AbstractBase', '~> 1.30.2'
s.dependency 'PLVFDB', '~> 1.0.5'
s.dependency 'PLVLOpenSSL', '~> 1.1.12101'
s.dependency 'SSZipArchive', '~> 2.0'
s.platform = :ios, '13.0'
```

Android 原生架构

组件	文件	职责
Plugin 入口	`PolyvMediaPlayerPlugin.kt`	注册 MethodChannel、EventChannel
Method 路由	`MethodRouter.kt`	方法分发
播放协调	`PlayerCoordinator.kt`	播放器实例管理
下载协调	`DownloadCoordinator.kt`	下载任务管理

组件	文件	职责
字幕协调	`SubtitleCoordinator.kt`	字幕轨道管理
播放事件	`PlaybackEventEmitter.kt`	播放事件发送到 Flutter
下载事件	`DownloadEventEmitter.kt`	下载事件发送到 Flutter
视频视图	`PolyvVideoViewFactory.kt`	PlatformView 工厂

Android 依赖

```
implementation("net.polyv.android:media-player-full:2.7.2")
implementation("net.polyv.android:media-player-sdk-addon-business:2.7.2")
implementation("net.polyv.android:media-player-sdk-addon-download:2.7.2")
```

业务服务模块

插件在 `infrastructure/` 目录下提供了可跨端共享的业务服务：

弹幕服务 (Danmaku)

```
// 弹幕数据模型
class Danmaku {
  final String id;
  final String content;
  final int time;       // 毫秒
  final String color;
  final String type;    // scroll/top/bottom
}

// 弹幕服务接口
abstract class DanmakuService {
  Future<List<Danmaku>> fetchDanmakus(String vid);
}

// 弹幕发送服务接口
abstract class DanmakuSendService {
  Future<void> sendDanmaku(String vid, String content, {String? color});
}
```

内置实现：

- `HttpDanmakuService` - 通过保利威弹幕 API 获取弹幕
- `HttpDanmakuSendService` - 通过保利威 API 发送弹幕
- `MockDanmakuService` - Mock 实现（用于调试）

视频列表服务 (VideoList)

```
class VideoListService {
  Future<VideoListResult> fetchVideoList({
    int? page,
    int? pageSize,
    String? categoryId,
  });
}
```

下载管理 (Download)

```
// 下载任务状态
enum DownloadTaskStatus { pending, downloading, paused, completed, failed, canceled }

// 下载状态管理（单例）
class DownloadStateManager extends ChangeNotifier {
  static final DownloadStateManager instance = DownloadStateManager._();
  Future<void> syncFromNative();  // 从原生层同步下载列表
  List<DownloadTask> get tasks;
}

// 通过 PlayerController 的 Platform Channel 调用原生下载能力
// startDownload / pauseDownload / resumeDownload / retryDownload / deleteDownload
```

Demo 应用

`example/` 目录包含完整的示例应用，展示了插件的所有功能。

页面结构

页面	文件	功能
首页	`pages/home_page.dart`	深色渐变背景，长视频/下载中心两个入口按钮

页面	文件	功能
长视频页	`pages/long_video_page.dart`	视频播放 + 视频列表 + 弹幕 + 控制栏
下载中心	`pages/download_center/`	下载任务列表、状态管理、操作控制

账号配置

Demo 通过 `--dart-define` 环境变量注入账号配置：

```
flutter run \
  --dart-define=POLYV_USER_ID=your_user_id \
  --dart-define=POLYV_SECRET_KEY=your_secret_key \
  --dart-define=POLYV_READ_TOKEN=your_read_token \
  --dart-define=POLYV_WRITE_TOKEN=your_write_token
```

配置由 `config/app_config.dart` 读取并注入到 `PolyvConfigService`。

运行 Demo

```
cd polyv_media_player
fvm flutter pub run example  # 或 flutter run
```

开发指南

命名规范

类型	规范	示例
Class	PascalCase	`PlayerController`
Method	camelCase	`seekTo()`
Variable	camelCase	`currentPosition`
Private	前缀 `_`	`_nativeChannel`

类型	规范	示例
File	snake_case	`player_controller.dart`

状态管理

使用 Provider + ChangeNotifier 模式：

```
// 在 Widget 中监听 PlayerController
Consumer<PlayerController>(
  builder: (context, controller, child) {
    return Text('${controller.state.position}');
  },
)
```

错误处理

所有 Platform Channel 调用必须捕获异常：

```
try {
  await _channel.invokeMethod('playVideo', {'vid': vid});
} on PlatformException catch (e) {
  throw PlayerException(
    code: e.code ?? 'UNKNOWN_ERROR',
    message: e.message ?? 'An error occurred',
  );
}
```

运行测试

```
cd polyv_media_player
fvm flutter test           # 插件单元测试
fvm flutter test example   # Demo 测试
```

代码分析

```
cd polyv_media_player
fvm flutter analyze
```

常见问题

Q1: iOS 编译失败?

1. 确保运行 `cd ios && pod install`
2. 检查 iOS 最低版本 >= 13.0
3. 确保 CocoaPods 版本兼容

Q2: Android 编译失败?

1. 确保项目 Android `minSdkVersion` >= 21
2. 检查 Gradle 配置中的 Maven 仓库可访问

Q3: 视频无法播放?

1. 检查 `userId` 和 `secretKey` 是否正确
2. 确认视频 VID 有效且已转码
3. 查看控制台日志中的错误信息

Q4: 如何获取保利威账号配置?

1. 前往 保利威官网 注册账号
2. 在点播后台获取 `userId`、`secretKey`、`readToken`、`writeToken`

Q5: 如何更新原生 SDK 版本?

- iOS: 修改 `ios/polyv_media_player.podspec` 中的版本号
- Android: 修改 `android/build.gradle` 中的依赖版本号

参考链接

- 保利威帮助中心
- 保利威 iOS 点播 SDK 文档
- 保利威开发者中心
- iOS SDK Demo (GitHub)
- SDK 下载页面

- 点播 SDK 功能页