AI-SDK 逻辑层

说明

@polyv/live-watch-ai-sdk 是保利威直播观看 AI SDK，提供 AI 助手、智能大纲等 AI 功能的核心 JavaScript SDK。包含以下特性

- **AI 助手聊天**: 支持与 AI 助手进行实时对话，支持流式响应
- **AI 智能大纲**: 获取回放视频的智能大纲、字幕解析和互动答题
- **数字人支持**: 集成 AI 数字人功能，支持语音合成和视频播放
- **多语言支持**: 支持中文、英文、日文、韩文、俄文等多种语言
- **模块化设计**: 采用模块化架构，便于扩展和维护
- **事件驱动**: 基于事件驱动的 API 设计，便于状态管理

安装

```
npm install @polyv/live-watch-ai-sdk
# 或
yarn add @polyv/live-watch-ai-sdk
# 或
pnpm add @polyv/live-watch-ai-sdk
```

快速开始

基本使用

```
import { PolyvWatchAICore } from '@polyv/live-watch-ai-sdk';

// 初始化 SDK
const aiCore = new PolyvWatchAICore({
  lang: 'zh_CN',
  domainInfo: {
    polyvWatchApiDomain: 'https://watch-api.polyv.cn',
  },
  getViewerToken: async () => ({
    viewerToken: 'your-viewer-token',
  }),
  getChatToken: async () => ({
    chatToken: 'your-chat-token',
  }),
  getChannelInfo: async () => ({
    channelId: 'your-channel-id',
  }),
  getUserInfo: async () => ({
    userId: 'user-123',
    nick: '用户昵称',
    pic: 'https://example.com/avatar.jpg',
  }),
});

// 使用 AI 助手模块
const aiAssistant = aiCore.aiAssistant;

// 监听 AI 助手事件
aiAssistant.eventEmitter.on('AIAssistantChatMessage', ({ AIChatMsg }) => {
  console.log('收到 AI 消息:', AIChatMsg);
});

// 发送问题给 AI 助手
try {
  await aiAssistant.sendAIQuestionMsg({
    content: '你好，请介绍一下这个视频的主要内容',
  });
} catch (error) {
  console.error('发送问题失败:', error);
}
```

核心模块

PolyvWatchAICore

SDK 的核心类，管理所有模块和配置。

构造函数配置

```
interface AICoreConfig {
  /** 语言类型 */
  lang?: LangType;
  /** 域名信息 */
  domainInfo?: Partial<DomainInfo>;
  /** 获取用户令牌信息 */
  getViewerToken?: () => Promise<ViewerTokenData> | ViewerTokenData;
  /** 获取聊天室服务令牌信息 */
  getChatToken?: () => Promise<ChatTokenData> | ChatTokenData;
  /** 获取频道信息 */
  getChannelInfo?: () => Promise<ChannelInfo> | ChannelInfo;
  /** 获取频道配置 */
  getChannelConfig?: () => Promise<ChannelConfig> | ChannelConfig;
  /** 获取用户信息 */
  getUserInfo?: () => Promise<Partial<UserInfo>> | Partial<UserInfo>;
}
```

核心属性

- `aiAssistant`：AI 助手模块
- `aiSummary`：AI 智能大纲模块
- `domain`：域名模块

核心方法

- `getAppConfig()`：获取当前应用配置
- `updateAppConfig()`：更新应用配置
- `getViewerToken()`：获取观看者令牌
- `getChatToken()`：获取聊天令牌
- `getChannelId()`：获取频道 ID
- `getUserInfo()`：获取用户信息
- `destroy()`：销毁 SDK 实例

AI 助手模块

提供 AI 助手聊天功能，支持数字人交互。

主要功能

1. **AI 助手聊天**
   - 支持流式对话
   - 消息分组管理
   - 聊天状态管理
2. **数字人功能**
   - 数字人视频播放
   - 语音合成 (TTS)
   - 音频任务管理
3. **音频处理**
   - 语音识别 (ASR)
   - 音频转文字
   - 音频文件处理

使用示例

```
// 初始化 AI 助手
await aiCore.aiAssistant.setupAIAssistant(
  {
    aiAssistantId: 123,
    aiAssistantName: 'AI助手',
    aiAssistantCode: 'assistant-code',
  },
  { independent: true },
);

// 发送问题
await aiCore.aiAssistant.sendAIQuestionMsg({
  content: '请介绍一下这个视频',
});

// 获取聊天历史
const history = await aiCore.aiAssistant.getAIAssistantChatHistory();

// 创建音频任务（TTS）
const { taskId } = await aiCore.aiAssistant.createAiAssistantAudioTask({
  text: '你好，我是AI助手',
  rate: 1.0,
  ttsVoiceId: 'voice-001',
});

// 语音识别
const text = await aiCore.aiAssistant.recognizeAudioBlobToText({
  audioBlob: audioFile,
  audioType: 'wav',
});
```

AI 智能大纲模块

提供回放视频的智能分析功能。

主要功能

1. **回放大纲**
   - 获取视频摘要
   - 分段内容总结
   - 关键词提取
2. **字幕处理**
   - 获取 AI 生成字幕
   - SRT 格式解析

- 时间轴对齐

3. **互动答题**
   - 获取视频中的互动问题
   - 题目类型识别
   - 答案验证

使用示例

```
// 获取回放大纲
const outline = await aiCore.aiSummary.getPlaybackOutline({
  id: 'video-123',
  type: 'playback', // 'record' 或 'playback'
});

if (outline) {
  console.log('视频摘要:', outline.introduction);
  console.log('分段数量:', outline.outlineContent.length);
}

// 获取并解析字幕
const subtitleContent = await aiCore.aiSummary.getAISubtitleContent(
  'https://example.com/subtitle.srt',
);
const parsedSubtitles = aiCore.aiSummary.parseAISubtitleContent(subtitleContent);

parsedSubtitles.forEach(item => {
  console.log(`时间: ${item.start}-${item.end}ms, 内容: ${item.text}`);
});

// 获取互动答题
const questionData = await aiCore.aiSummary.getAIPlaybackQuestionData(
  'https://example.com/questions.json',
);
```

事件系统

AI 助手事件

```
enum AIAssistantEvents {
  // AI 助手设置完成
  AIAssistantChatSetupedIndependent = 'AIAssistantChatSetupedIndependent',
  AIAssistantChatSetupedComplete = 'AIAssistantChatSetupedComplete',

  // 聊天状态变化
  AIAssistantChatStatusChange = 'AIAssistantChatStatusChange',

  // 聊天消息
  AIAssistantChatMessage = 'AIAssistantChatMessage',
  ReplaceAIAssistantChatMessage = 'ReplaceAIAssistantChatMessage',
}
```

聊天状态

```
enum PolyvAIAssistantChatStatus {
  Wait = 0, // 等待用户输入
  Replaying = 1, // AI 正在回答
  Busy = 2, // AI 繁忙
}
```

事件监听示例

```
// 监听 AI 助手状态变化
aiCore.aiAssistant.eventEmitter.on('AIAssistantChatStatusChange', ({ status }) => {
  console.log('AI 助手状态变化:', status);
});

// 监听聊天消息
aiCore.aiAssistant.eventEmitter.on('AIAssistantChatMessage', ({ AIChatMsg }) => {
  console.log('收到消息:', AIChatMsg);
});

// 监听消息更新（用于流式响应）
aiCore.aiAssistant.eventEmitter.on('ReplaceAIAssistantChatMessage', ({ id, AIChatMsg }) => {
  console.log('消息更新:', AIChatMsg.content);
});
```

工具函数

调试工具

```
import { setDebugMode } from '@polyv/live-watch-ai-sdk';

// 开启调试模式
setDebugMode(true);
```

实用工具

```
import { plvInterval, plvWait, plvParseJson } from '@polyv/live-watch-ai-sdk';

// 定时器
const timer = plvInterval(
  () => {
    console.log('定时执行');
  },
  1000,
  { maxCount: 10 },
);

// 延迟执行
await plvWait(3000);

// JSON 解析
const result = plvParseJson<{ data: string }>('{"data": "test"}');
if (result.success) {
  console.log(result.data);
}
```

最佳实践

性能优化

1. **延迟加载模块**: 按需初始化模块，减少初始加载时间

```
const aiCore = new PolyvWatchAICore(config);

if (needAIAssistant) {
  await aiCore.aiAssistant.setupAIAssistant(assistantConfig);
}
```

2. **事件监听器管理**: 及时清理事件监听器，避免内存泄漏

```
const messageHandler = ({ AIChatMsg }) => {
  console.log('收到消息:', AIChatMsg);
};

aiAssistant.eventEmitter.on('AIAssistantChatMessage', messageHandler);

// 组件销毁时清理
function cleanup() {
  aiAssistant.eventEmitter.off('AIAssistantChatMessage', messageHandler);
}
```

状态管理

```
// 在 Vue/React 中，将 SDK 状态与组件状态绑定
const messages = ref([]);
const chatStatus = ref(PolyvAIAssistantChatStatus.Wait);

aiAssistant.eventEmitter.on('AIAssistantChatMessage', ({ AIChatMsg }) => {
  messages.value.push(AIChatMsg);
});

aiAssistant.eventEmitter.on('AIAssistantChatStatusChange', ({ status }) => {
  chatStatus.value = status;
});
```

常见问题

1. 如何获取必要的令牌?

SDK 需要以下令牌才能正常工作：

- `viewerToken`：观看者令牌，用于 API 认证
- `chatToken`：聊天令牌，用于 AI 助手聊天

这些令牌需要通过业务后端接口获取，并在初始化时通过回调函数提供给 SDK。

2. AI 助手不响应怎么办?

检查以下配置：

1. 确保 `aiAssistantCode` 正确设置

2. 验证 `getChatToken` 返回有效的聊天令牌
3. 检查网络连接和域名配置
4. 查看浏览器控制台是否有错误信息

3. 如何自定义域名?

```
const aiCore = new PolyvWatchAICore({
  domainInfo: {
    polyvWatchApiDomain: 'https://your-custom-domain.com',
  },
});
```