```java
package com.catdog2025.mediation.java;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import androidx.annotation.Nullable;
import android.util.Log;
import android.view.View;
import android.widget.FrameLayout;
import android.widget.TextView;

import com.bytedance.sdk.openadsdk.AdSlot;
import com.bytedance.sdk.openadsdk.CSJAdError;
import com.bytedance.sdk.openadsdk.CSJSplashAd;
import com.bytedance.sdk.openadsdk.CSJSplashCloseType;
import com.bytedance.sdk.openadsdk.TTAdNative;
import com.catdog2025.R;
import com.catdog2025.activity.CatDogActivity;
import com.catdog2025.config.TTAdManagerHolder;
import com.catdog2025.mediation.java.utils.Const;
import com.catdog2025.utils.UIUtils;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

/**
 * 融合demo，开屏广告使用示例。更多功能参考接入文档。
 *
 * 注意：每次加载的广告，只能展示一次
 *
 * 接入步骤：
 * 1、创建AdSlot对象
 * 2、创建TTAdNative对象
 * 3、创建加载、展示监听器
 * 4、加载广告
 * 5、加载并渲染成功后，展示广告
 * 6、在onDestroy中销毁广告
 */
```

```
40  public class MediationSplashActivity extends Activity {
41
42      private static final String TAG = "MediationSplashActivity";
43
44      private FrameLayout mSplashContainer;
45      private TextView mCountdownTextView; // 倒计时控件
46
47      private CSJSplashAd mCsjSplashAd;
48
49      private TTAdNative.CSJSplashAdListener mCSJSplashAdListener;
50
51      private CSJSplashAd.SplashAdListener
    mCSJSplashInteractionListener;
52
53      // 超时处理
54      private Handler mTimeoutHandler = new Handler(Looper.
    getMainLooper());
55      private Runnable mTimeoutRunnable;
56      private boolean isAdLoaded = false;
57
58      // 倒计时处理
59      private Handler mCountdownHandler = new Handler(Looper.
    getMainLooper());
60      private Runnable mCountdownRunnable;
61      private int mCountdownSeconds = 6; // 倒计时秒数
62      private boolean isCountdownRunning = false; //
    倒计时是否在运行
63
64      @Override
65      protected void onCreate(@Nullable Bundle savedInstanceState) {
66          super.onCreate(savedInstanceState);
67          Log.d(TAG, "onCreate: 创建开屏广告页面（优化版本）");
68          setContentView(R.layout.mediation_activity_splash);
69          mSplashContainer = findViewById(R.id.fl_content);
70          mCountdownTextView = findViewById(R.id.tv_countdown);
71
72          // 检查网络状态
73          checkNetworkStatus();
74
```

```
        // 检查本地配置支持情况
        checkLocalConfigSupport();

        // 启动倒计时
        startCountdown();

        // 加载并展示广告
        loadAndShowSplashAd();
    }

    /**
     * 检查网络状态
     */
    private void checkNetworkStatus() {
        ConnectivityManager cm = (ConnectivityManager) getSystemService(CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = cm.getActiveNetworkInfo();
        boolean isConnected = networkInfo != null && networkInfo.isConnected();

        Log.d(TAG, "□ 网络状态检查:");
        Log.d(TAG, "   连接状态: " + (isConnected ? "已连接" : "未连接"));
        if (networkInfo != null) {
            Log.d(TAG, "   网络类型: " + networkInfo.getTypeName());
        }

        if (!isConnected) {
            Log.w(TAG, "□□ 网络未连接，将依赖本地配置和兜底机制");
            if (mCountdownTextView != null) {
                mCountdownTextView.setText("网络异常，使用离线配置");
            }
        }
    }

    /**
     * 检查本地配置支持情况
     */
    private void checkLocalConfigSupport() {
```

```
111         boolean localConfigSupported = TTAdManagerHolder.
    isLocalConfigSupported();
112         boolean fallbackSupported = TTAdManagerHolder.
    isFallbackSupported();
113
114         Log.d(TAG, "□ 配置兜底支持情况:");
115         Log.d(TAG, "   本地配置支持: " + (localConfigSupported ? "□
    支持" : "□ 不支持"));
116         Log.d(TAG, "   自定义兜底支持: " + (fallbackSupported ? "□
    支持" : "□ 不支持"));
117
118         if (localConfigSupported) {
119             Log.d(TAG, "   本地配置文件: " + TTAdManagerHolder.
    getLocalConfigFilePath());
120         }
121         if (fallbackSupported) {
122             Log.d(TAG, "   兜底代码位: " + TTAdManagerHolder.
    getSplashFallbackCodeId());
123         }
124
125         if (!localConfigSupported && !fallbackSupported) {
126             Log.i(TAG, "□ 当前为标准SDK，使用基础错误处理（功能正常
    ）");
127         }
128     }
129
130     private void loadAndShowSplashAd() {
131         /** 1、创建AdSlot对象 */
132         String splashMediaId = getResources().getString(R.string.
    splash_media_id);
133         Log.d(TAG, "loadAndShowSplashAd: 开始加载开屏广告，
    广告位ID: " + splashMediaId);
134
135         AdSlot adSlot = new AdSlot.Builder()
136                 .setCodeId(splashMediaId)
137                 .setImageAcceptedSize(UIUtils.getScreenWidthInPx(this),
    UIUtils.getScreenHeightInPx(this))
138                 .build();
139
```

```
140         /** 2、创建TTAdNative对象 */
141
142         TTAdNative adNativeLoader = TTAdManagerHolder.get().
        createAdNative(this);
143
144         /** 3、创建加载、展示监听器 */
145         initListeners();
146
147         /** 4、加载广告 */
148         if (adNativeLoader != null) {
149             // 设置超时处理
150             startTimeoutHandler();
151             // 增加广告加载超时时间从3.5秒到6秒，给广告更多加载机会
152             Log.d(TAG, "loadAndShowSplashAd: 开始加载广告，
        超时设置6000ms（增加到6秒）");
153             adNativeLoader.loadSplashAd(adSlot,
        mCSJSplashAdListener, 6000);
154         } else {
155             Log.e(TAG, "loadAndShowSplashAd: adNativeLoader为空
        ，无法加载广告");
156             jumpToCatDogActivity("广告加载器为空");
157         }
158     }
159
160     private void initListeners() {
161         // 广告加载监听器
162
163         this.mCSJSplashAdListener = new TTAdNative.
        CSJSplashAdListener() {
164             @Override
165
166             public void onSplashRenderSuccess(CSJSplashAd
        csjSplashAd) {
167                 /** 5、渲染成功后，展示广告 */
168                 Log.d(TAG, "广告渲染成功，开始展示");
169                 isAdLoaded = true;
170                 cancelTimeoutHandler();
171
172                 // 广告成功加载，更新倒计时显示
```

```
173             stopCountdown();
174             if (mCountdownTextView != null) {
175                 //mCountdownTextView.setText("广告中");
176                 mCountdownTextView.setVisibility(View.VISIBLE);
177             }
178
179             mCsjSplashAd = csjSplashAd;
180             csjSplashAd.setSplashAdListener(
    mCSJSplashInteractionListener);
181             View splashView = csjSplashAd.getSplashView();
182             UIUtils.removeFromParent(splashView);
183             mSplashContainer.removeAllViews();
184             mSplashContainer.addView(splashView);
185         }
186
187         public void onSplashLoadSuccess() {
188             Log.d(TAG, "splash load success");
189         }
190
191         @Override
192
193         public void onSplashLoadSuccess(CSJSplashAd csjSplashAd
    ) {
194
195         }
196
197         @Override
198
199         public void onSplashLoadFail(CSJAdError csjAdError) {
200             Log.e(TAG, "□ 开屏广告加载失败详情:");
201             Log.e(TAG, "   错误代码: " + csjAdError.getCode());
202             Log.e(TAG, "   错误信息: " + csjAdError.getMsg());
203
204             // 分析错误原因并记录
205             analyzeAdLoadError(csjAdError);
206
207             // 更新倒计时显示
208             if (mCountdownTextView != null) {
209                 mCountdownTextView.setText("广告加载失败，
```

```
209 使用兜底配置");
210             }
211
212             // 检查是否有兜底机制可用
213             if (TTAdManagerHolder.isFallbackSupported()) {
214                 Log.i(TAG, "□□ 尝试使用自定义兜底机制");
215             } else {
216                 Log.w(TAG, "□□ 无兜底机制可用，等待超时处理");
217             }
218
219             // 继续等待6秒超时机制处理，确保开屏页面展示时间
220             Log.d(TAG, "□ 广告加载失败，继续等待超时机制处理跳转");
221         }
222
223         @Override
224
225         public void onSplashRenderFail(CSJSplashAd csjSplashAd,
    CSJAdError csjAdError) {
226             Log.d(TAG, "广告渲染失败, 错误代码: " + csjAdError.
    getCode() + ", 错误信息: " + csjAdError.getMsg());
227             // 注意：广告渲染失败时不立即跳转，让超时机制处理，
    确保开屏页面展示时间
228             Log.d(TAG, "广告渲染失败，
    但继续等待6秒超时机制处理跳转，确保足够展示时间");
229             // 不调用 jumpToCatDogActivity，让6秒超时机制处理
230         }
231     };
232     // 广告展示监听器
233
234     this.mCSJSplashInteractionListener = new CSJSplashAd.
    SplashAdListener() {
235         @Override
236
237         public void onSplashAdShow(CSJSplashAd csjSplashAd) {
238             Log.d(TAG, "splash show");
239         }
240
241         @Override
242
```

```
243         public void onSplashAdClick(CSJSplashAd csjSplashAd) {
244             Log.d(TAG, "splash click");
245         }
246
247         @Override
248
249         public void onSplashAdClose(CSJSplashAd csjSplashAd, int
    closeType) {
250
251             if (closeType == CSJSplashCloseType.CLICK_SKIP) {
252                 Log.d(TAG, "开屏广告点击跳过");
253
254             } else if (closeType == CSJSplashCloseType.
    COUNT_DOWN_OVER) {
255                 Log.d(TAG, "开屏广告点击倒计时结束");
256
257             } else if (closeType == CSJSplashCloseType.CLICK_JUMP
    ) {
258                 Log.d(TAG, "点击跳转");
259             }
260
261             // 跳转到Cat&Dog展示页面
262             jumpToCatDogActivity("广告正常结束");
263         }
264     };
265   }
266
267   /**
268    * 分析广告加载错误原因
269    * @param error 错误信息
270    */
271   private void analyzeAdLoadError(CSJAdError error) {
272     int errorCode = error.getCode();
273     String errorMsg = error.getMsg();
274
275     Log.d(TAG, "□ 广告错误分析:");
276
277     // 常见错误代码分析
278     switch (errorCode) {
```

```
279             case 40001:
280                 Log.d(TAG, "   原因: 配置拉取失败（网络问题或服务器异常
    ）");
281                 Log.d(TAG, "   建议: 本地配置和兜底机制将发挥作用");
282                 break;
283             case 40002:
284                 Log.d(TAG, "   原因: 广告位配置错误");
285                 Log.d(TAG, "   建议: 检查广告位ID是否正确");
286                 break;
287             case 40003:
288                 Log.d(TAG, "   原因: 无广告填充");
289                 Log.d(TAG, "   建议: 正常现象，兜底机制处理");
290                 break;
291             case 20005:
292                 Log.d(TAG, "   原因: 全部代码位请求失败（
    聚合广告位无可用资源）");
293                 Log.d(TAG, "   说明: 所有聚合的广告平台都无法返回广告");
294                 Log.d(TAG, "   建议: 正常现象，超时机制保证用户体验");
295                 break;
296             case -8:
297                 Log.d(TAG, "   原因: 网络超时");
298                 Log.d(TAG, "   建议: 本地配置将提供保障");
299                 break;
300             default:
301                 Log.d(TAG, "   原因: 其他错误 (" + errorCode + ")");
302                 Log.d(TAG, "   描述: " + errorMsg);
303                 break;
304         }
305
306         // 检查网络状态
307         ConnectivityManager cm = (ConnectivityManager)
    getSystemService(CONNECTIVITY_SERVICE);
308         NetworkInfo networkInfo = cm.getActiveNetworkInfo();
309         boolean isConnected = networkInfo != null && networkInfo.
    isConnected();
310         Log.d(TAG, "   当前网络: " + (isConnected ? "正常" : "异常"));
311
312         // 兜底机制状态
313         Log.d(TAG, "   本地配置: " + (TTAdManagerHolder.
```

```
313  isLocalConfigSupported() ? "可用" : "不可用"));
314          Log.d(TAG, "   自定义兜底: " + (TTAdManagerHolder.
     isFallbackSupported() ? "可用" : "不可用"));
315      }
316
317      /**
318       * 启动倒计时
319       */
320      private void startCountdown() {
321          if (isCountdownRunning) {
322              return; // 防止重复启动
323          }
324
325          isCountdownRunning = true;
326          mCountdownSeconds = 6; // 重置倒计时为6秒
327          updateCountdownDisplay();
328
329          mCountdownRunnable = new Runnable() {
330              @Override
331              public void run() {
332                  if (mCountdownSeconds > 0 && !isFinishing()) {
333                      mCountdownSeconds--;
334                      updateCountdownDisplay();
335
336                      if (mCountdownSeconds > 0) {
337                          // 继续倒计时
338                          mCountdownHandler.postDelayed(this, 1000);
339                      } else {
340                          // 倒计时结束
341                          Log.d(TAG, "倒计时结束，准备跳转");
342                          isCountdownRunning = false;
343                      }
344                  } else {
345                      isCountdownRunning = false;
346                  }
347              }
348          };
349
350          // 1秒后开始第一次倒计时更新
```

```
351         mCountdownHandler.postDelayed(mCountdownRunnable,
    1000);
352     }
353
354     /**
355      * 停止倒计时
356      */
357     private void stopCountdown() {
358         if (mCountdownHandler != null && mCountdownRunnable
     != null) {
359             mCountdownHandler.removeCallbacks(
    mCountdownRunnable);
360             isCountdownRunning = false;
361         }
362     }
363
364     /**
365      * 更新倒计时显示
366      */
367     private void updateCountdownDisplay() {
368         if (mCountdownTextView != null) {
369             if (mCountdownSeconds > 0) {
370                 mCountdownTextView.setText(mCountdownSeconds + "
    s");
371                 mCountdownTextView.setVisibility(View.VISIBLE);
372             } else {
373                 mCountdownTextView.setText("跳转中...");
374             }
375         }
376     }
377
378     /**
379      * 开始超时处理
380      */
381     private void startTimeoutHandler() {
382         mTimeoutRunnable = new Runnable() {
383             @Override
384             public void run() {
385                 if (!isAdLoaded && !isFinishing()) {
```

```
386                 Log.d(TAG, "开屏页面6秒展示完成，超时跳转");
387                 jumpToCatDogActivity("开屏展示超时");
388             }
389         }
390     };
391     // 设置6秒超时，确保开屏页面至少展示6秒（无论广告成功还是失败）
392     mTimeoutHandler.postDelayed(mTimeoutRunnable, 6000);
393 }
394
395 /**
396  * 取消超时处理
397  */
398 private void cancelTimeoutHandler() {
399     if (mTimeoutHandler != null && mTimeoutRunnable != null) {
400         mTimeoutHandler.removeCallbacks(mTimeoutRunnable);
401     }
402 }
403
404 /**
405  * 统一的跳转方法，确保最小展示时间
406  * @param reason 跳转原因
407  */
408 private void jumpToCatDogActivity(String reason) {
409     if (isFinishing()) {
410         return; // 防止重复跳转
411     }
412
413     // 停止倒计时
414     stopCountdown();
415
416     Log.d(TAG, reason + "，立即跳转到CatDogActivity");
417     Intent intent = new Intent(MediationSplashActivity.this, CatDogActivity.class);
418     startActivity(intent);
419     finish();
420 }
421
422 @Override
```

```
423     protected void onDestroy() {
424         super.onDestroy();
425         // 清理超时处理器
426         cancelTimeoutHandler();
427         // 清理倒计时处理器
428         stopCountdown();
429 
430         /** 6、在onDestroy中销毁广告 */
431         if (mCsjSplashAd != null && mCsjSplashAd.
        getMediationManager() != null) {
432             mCsjSplashAd.getMediationManager().destroy();
433         }
434     }
435 }
436 
437 
```