Flutter：webview_flutter插件使用

原创 無_爲 已于 2022-11-10 12:27:15 修改 685 收藏 3 版权
分类专栏： Flutter 文章标签： flutter android

Flutter 专栏收录该内容 2 订阅 18 篇文章 订阅专栏

一、添加权限：

1.Android网络权限(工程/android/app/src/main/AndroidManifest.xml)：

```
<manifest ...>
    ...
    <uses-permission android:name="android.permission.INTERNET"/>
</manifest>
```

2.iOS添加使用说明(工程/ios/Runner/Info.plist)：

```
<dict>
    ...
    <key>io.flutter.embedded_views_preview</key>
    <string>YES</string>
</dict>
```

二、使用 WebView 显示网页：

1.添加webview_flutter插件依赖，在pubspec.yaml中：

```
dependencies:
  webview_flutter: ^3.0.4  #WebView插件
```

2.使用WebView显示网页：

```
class WebViewPage extends StatefulWidget { //网页显示界面
  const WebViewPage({Key? key, required this.url}) : super(key: key); //外部传入URL
  final String url;
  @override
  State<WebViewPage> createState() => _PageState();
}
class _PageState extends State<WebViewPage> {
  late WebViewController _webControl; //WebView控制类
  final CookieManager _cookieManager = CookieManager();
  late double progress = 0.01;  //H5加载进度值
  final double VISIBLE = 1;
  final double GONE = 0;
  late double progressHeight = GONE; //H5加载进度条高度
  //显示或隐藏进度条
  void setProgressVisible(double isVisible) {
    setState(() {
      progressHeight = isVisible;
    });
  }
  @override
  void initState() {
    super.initState();
    if (Platform.isAndroid)  WebView.platform = SurfaceAndroidWebView();  //androi
  }
  WebView _createWebView(){
    return WebView( //WebView组件
      initialUrl: widget.url,             //设置URL地址
      javascriptMode: JavascriptMode.unrestricted,  //启用JS
      onWebViewCreated: (WebViewController webControl) { //WebView创建时触发
        _webControl = webControl;
      },
      javascriptChannels: <JavascriptChannel>{getJSChannel1(context), getJSChannel
      navigationDelegate: (NavigationRequest request) { //控制页面是否进入下页
        if (request.url.startsWith('http://www.yyh.com')) {
          return NavigationDecision.prevent; //禁止跳下页
        }
        return NavigationDecision.navigate; //放行跳下页
      },
      onPageStarted: (String url) { //H5页面开始加载时触发
        setProgressVisible(VISIBLE); //显示加载进度条
      },
   onProgress: (int progress) { //加载H5页面时触发多次，progress值为0-100
    this.progress = progress.toDouble() / 100.0;  //计算成0.0-1.0之间的值
   },
   onPageFinished: (String url) { //H5页面加载完成时触发
    setProgressVisible(GONE); //隐藏加载进度条
   },
   gestureNavigationEnabled: true,  //启用手势
  );
 }
 //添加JS方法1，与H5交互数据
 JavascriptChannel getJSChannel1(BuildContext context) {
  return JavascriptChannel(
    name: 'jsMethodName1',  //方法名，与H5中的名称一致。H5中调用方式：js方法名1.pos
    onMessageReceived: (JavascriptMessage message) { //接收H5发过来的数据
     String json = message.message;
     print("H5发过来的数据1: $json");
    });
 }
 //添加JS方法2，与H5交互数据
 JavascriptChannel getJSChannel2(BuildContext context) {
  return JavascriptChannel(
    name: 'jsMethodName2',
    onMessageReceived: (JavascriptMessage message) {
     String json = message.message;
     print("H5发过来的数据2: $json");
    });
 }
 @override
 Widget build(BuildContext context) {
  return WillPopScope(
    onWillPop: () async {  //拦截页面返回事件
     if (await _webControl.canGoBack()) {
      _webControl.goBack(); //返回上个网页
      return false;
     }
     return true; //退出当前页
    },
    child: Scaffold(
      appBar: AppBar(title: const Text('WebView使用')),
      body: Stack(
       children: [
        _createWebView(), //WebView
        SizedBox(       //进度条
          height: progressHeight,
          child: LinearProgressIndicator(
           backgroundColor: Colors.white,
           valueColor: AlwaysStoppedAnimation(Colors.yellow),
           value: progress, //当前加载进度值
          ))
       ],
      )));
 }
}
```

三、WebView常用方法:

1.加载Html的几种方式:

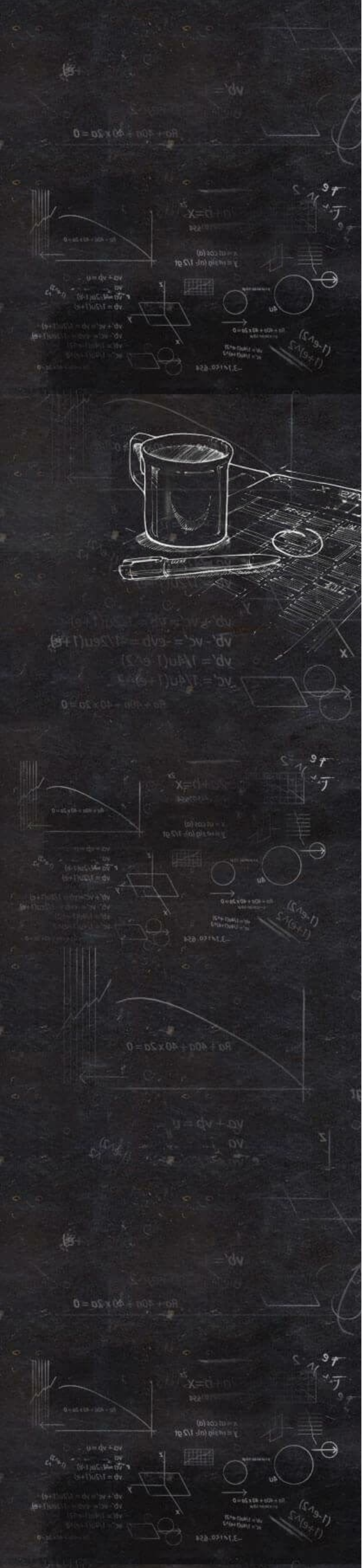
(1)加载Html字符串:

```
String htmlString = '''
 <!DOCTYPE html><html>
 <head><title>Navigation Delegate Example</title></head>
 <body>
 加载Html内容
 </body>
 </html>
 ''';
```

方式1:

```
Future<void> loadHtmlString1() async {
 String content = base64Encode(const Utf8Encoder().convert(htmlString));
 await _webControl.loadUrl('data:text/html;base64,$content');
}
```

方式2:



```
Future<void> loadHtmlString2() async {
  await _webControl.loadHtmlString(htmlString);
}
```

(2)加载Html文件：

资源文件方式：

```
Future<void> loadHtmlFile1() async {
  await _webControl.loadFlutterAsset('assets/www/index.html');
}
```

File方式：

```
Future<void> loadHtmlFile2() async {
  String dir = (await getTemporaryDirectory()).path;
  File file = File(<String>{dir, 'www', 'index.html'}.join(Platform.pathSeparator)
  await file.create(recursive: true);
  String filePath = file.path;
  await file.writeAsString(htmlString);  //将html字符串写入文件
  await _webControl.loadFile(filePath);  //加载html文件
}
```

(3)加载HTTP请求：

```
Future<void> loadHttpRequest(String url, String json) async {
  final WebViewRequest request = WebViewRequest(
    uri: Uri.parse(url), //设置URL
    method: WebViewRequestMethod.post,  //设置请求方式, post或get请求
    headers: <String, String>{'Content-Type': 'application/json'},  //请求头字段
    body: Uint8List.fromList(json.codeUnits),  //请求参数
  );
  await _webControl.loadRequest(request); //加载HTTP请求
}
```

2.运行JS代码：

```
Future<void> runJSCode() async { //运行JS代码, 模拟H5与Flutter交互数据
  await _webControl.runJavascript('jsMethodName1.postMessage("字符串");');  //此处模
}
```

3.Cookie添加/获取/清空：

```
//添加cookie
Future<void> addCookie() async {
  await _cookieManager.setCookie(const WebViewCookie(name: 'key1', value: 'value1'
}
//获取cookie列表
Future<List<String>> getCookieList() async {
  String cookies = await _webControl.runJavascriptReturningResult('document.cookie
  if (cookies == null || cookies == '""') <String>[];
  return cookies.split(';'); //获取cookie列表
}
//清空cookie
Future<void> clearCookies(BuildContext context) async {
  await _cookieManager.clearCookies();
}
```

4.缓存对象添加/获取/清空：

```
//添加缓存对象
Future<void> addCache() async {//caches.open()创建缓存对象, 存在时不创建
  await _webControl.runJavascript('caches.open("缓存对象名"); localStorage["key1"]
}
//获取缓存对象
Future<void> getCacheList() async {
  await _webControl.runJavascript('caches.keys()'
      '.then((cacheKeys) => JSON.stringify({"cacheKeys" : cacheKeys, "localStorage
      '.then((caches) => jsMethodName1.postMessage(caches))');  //获取H5缓存对象, 调
}
//清空缓存对象
Future<void> clearCache() async {
  await _webControl.clearCache();
}
```