날짜 : 2025-01-16

태그 : #Flutter #WebView

> 메모
>
> WebView 로 개발을 진행한다면 껍데기만 flutter 를 사용하기 때문에 데이터관리나 세션 관리 등등을 신경 쓸 일이 많지않아 앱 쪽에서 개발 영역이 많지 않고 네이티브 설정을 위한 부분이 상대적으로 많다.

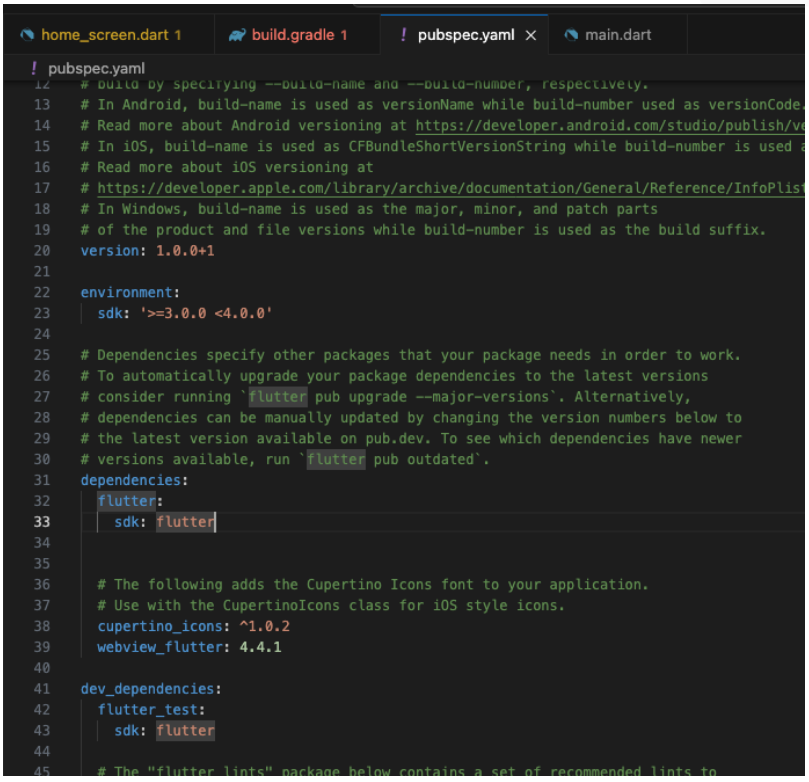
코드 구현

1. 프로젝트 생성

[flutter 프로젝트 생성 (VSCode)]

2. pubspec.yaml 설정하기

> pubspec.yaml 파일 설명 (pubspec.yaml 자세한 설명)
> dependencies 에 webview_flutter 플러그인 추가 및 `flutter pub get` 명령어 던지기



3. 권한 및 네이티브 설정하기

> 웹뷰를 사용하려면 몇 가지 네이티브 설정이 필요합니다.
> 인터넷 사용 권한을 추가하고 https 프로토콜뿐만 아니라 http 프로토콜도 이용할 수 있게 수정해야합니다.

- 안드로이드 설정

> 1. 안드로이드 설정 파일은 andorid/app/src/main/AndoridManifest.xml 입니다. 자세한 설명 -> 권한 설정
> 2. android/build.gradle 파일은 안드로이드의 빌드 툴인 그레들 설정 파일입니다. 자세한 설명 -> 안드로이드 버전 설정

- android/app/src/main/AndroidMenifest.xml (인터넷 허용 권한 추가)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.blog_web_app">

    <!-- 웹뷰를 실행할 때 인터넷을 사용해야하니 인터넷 권한 추가 -->
    <uses-permission android:name="android.permission.INTERNET" />

    ... 생략 ...
</manigest>
```

- android/app/build.gradle (안드로이드 버전 변경)

```
android {

    // 버전을 33버전으로 변경
    compileSdkVersion 33

    ... 생략 ...

    defaultConfig {

    // TODO: Specify your own unique Application ID
(https://developer.android.com/studio/build/application-id.html).

    applicationId "com.example.blog_web_app"

    // You can update the following values to match your application
needs.

    // For more information, see:
https://docs.flutter.dev/deployment/android#reviewing-the-build-
configuration.
```

```
17
18      // 20 버전으로 변경
19      minSdkVersion 20
20
21      targetSdkVersion flutter.targetSdkVersion
22
23      versionCode flutterVersionCode.toInteger()
24
25      versionName flutterVersionName
26
27  }
28
29      ... 생략 ...
30
31  }
```

> webview_flutter 플러그인을 사용하려면 안드로이드 최소 버전을 20 이상으로 설정해야합니다.

- android/app/src/main/AndroidManifest.xml (http 프로토콜 설정)

```
1   <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2   package="com.example.blog_web_app">
3
4   <uses-permission android:name="android.permission.INTERNET" />
5
6   <application
7       android:label="blog_web_app"
8       android:name="${applicationName}"
9       android:icon="@mipmap/ic_launcher"
10      android:usesCleartextTraffic="true">. -- http 프로토콜 허용 설정
11
12      ... 생략 ...
13  </manifest>
```

- IOS 설정

> 1. ios/Runner/Info.plist 파일에 작업 자세한 내용

- Info.plist (http 프로토콜을 사용하는 설정)

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
    "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
3   <plist version="1.0">
4   <dict>
```

```
    ... 생략 ...
    <!-- 추가한 코드 -->
    <key>NSAppTransportSecurity</key>
    <dict>
        <key>NSAllowsLocalNetworking</key>
        <true/>
        <key>NSAllowsArbitraryLoadsInWebContent</key>
        <true/>
    </dict>
</dict>
</plist>
```

4. 프로젝트 초기화 하기

> 프로그래밍을 하다 보면 폴더와 파일의 정리가 매우 중요합니다.
> 프로젝트가 작을 때는 원하는 코드나 특정 파일을 찾는게 어렵지 않지만 프로젝트가 커질수록 복잡해집니다. 이번 프로젝트를 포함하여 앞으로 진행할 모든 프로젝트에서는 화면과 관련된 모든 위젯을 [screen] 폴더에 모아둡니다.

- todo
  1. 우선 [lib] 폴더 위에서 마우스 우클릭
  2. [screen] 폴더를 생성
  3. [screen] 폴더 위에 다시 마우스 올려 우클릭
  4. 앱의 기본 홈 화면으로 사용할 (사용자 지정 위젯인) 홈스크린 위젯을 생성할 home_screen.dart 파일을 생성해줍니다.
  5. 그리고 statelessWidget 을 생성합니다.
  6. 마지막으로 main.dart 에서 home: 영역에 만들어준 class 를 사용한다.

5. 웹뷰 구현하기

- body에 웹뷰로 그려줄 영영에 WevViewWidget 으로 감싸준다.

```dart
import 'package:flutter/material.dart';
import 'package:grove_flutter/screen/home_screen.dart';
import 'package:webview_flutter/webview_flutter.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

```

```
12    // This widget is the root of your application.
13    @override
14    Widget build(BuildContext context) {
15      return const MaterialApp(
16        title: 'Grove Lecture',
17        home: WebViewWidget(
18        controller: webViewController,  // 순서대로 진행하면
19                                //webview contoller 가 선언되지 않아 에러가 발생함
20        ),
21      );
22    }
23  }
```

6. 웹뷰 컨트롤러 설정

> 웹뷰 컨트롤러는 웹뷰 위젯을 제어하는 역할을 합니다.
> 웹뷰 위젯은 화면에 웹뷰를 렌더링해서 웹사이트를 보여주는 역할을 합니다.
> 웹뷰 컨트롤러의 함수를 실행해서 웹뷰 위젯의 다양한 설정을 제어하고 웹사이트로 이동할 수 있습니다.

- lib/screen/home_screen.dart

```
1   // URI/URL을 생성하는데 도움을 주는 클래스
2   final uri = Uri.parse('http://3.35.61.27:8085');
3
4   /// screen 쪽에 webView 로 통신을 할 수있는 controller 생성
5   WebViewController controller = WebViewController()
6     ..setJavaScriptMode(JavaScriptMode.unrestricted)
7     ..setNavigationDelegate(NavigationDelegate(onPageFinished: (String url)
    {
8       print(url);
9     }))
10    ..loadRequest(uri); // ❶ 컨트롤러 변수 생성
```

> 1. WebViewController 타입인 webViewController 변수를 선언합니다.
> 2. WebViewWidget에 미리 webViewController 변수를 입력해뒀기 때문에 자동으로 컨트롤러가 입력됩니다.
> 3. JavascriptMode.unrestricted 를 입력하면 웹페이지에서 제한 없이 Javascript가 실행될 수 있도록 합니다. (반대의 기능은 JavascriptMode.restricted)
> 4. loadRequest() 함수는 웹뷰 위젯의 가장 중요한 함수 입니다. : loadRequest() 함수는 Uri 객체를 매개변수로 입력받으며 이 입력받은 값을 통해 지정한 사이트로 이동하는 기능입니다
> 5. Uri 객체를 매개변수로 입력받으며 이 입력받은 값을 통해 지정한 사이트로 이동합니다. : Uri.parse() static 함수가 존재하는데 이 함수에 이동하고 싶은 사이트의 URL 을 입력하면 해당 URL 이 Uri 로 자동 변경합니다.

7. main.dart 파일 수정

> 플러터 프로젝트를 실행하는 runApp() 함수는 내부적으로 WidgetFullterBinding.ensureInitialized() (이하 ensureInitialized() 함수) 함수를 실행하고 있습니다. 일반적으로 개발자가 직접 이 함수를 실행할 필요는 없지만 StatelessWidget에서 WebViewController 를 프로퍼티로 직접 인스턴스화 하려면 ensureInitizlied() 함수를 직접 실행해주는 작업을 해야합니다
> 만약, ensureInitizlied() 함수를 직접 실행하지 않고 WebViewController 를 정상적으로 인스턴스화하는 방법은 StateFullWidget의 initState() 함수에서 진행하게 됩니다.

- lib/main.dart

```
1
2  void main() {
3    /// 플러터 프레임워크가 앱을 실행할 준비가 될 때까지 기다림
4  WidgetsFlutterBinding.ensureInitialized();
5
6  ... 생략 ...
```

GIT 주소 연결

- WebView Flutter Git 레파지토리

생각 (질문)