# 11.3 Http请求-Dio http库

通过上一节介绍，我们可以发现直接使用HttpClient发起网络请求是比较麻烦的，很多事情得我们手动处理，如果再涉及到文件上传/下载、Cookie管理等就会非常繁琐。幸运的是，Dart社区有一些第三方http请求库，用它们来发起http请求将会简单的多，本节我们介绍一下目前人气较高的[dio](https://github.com/flutterchina/dio)库。

dio是一个强大的Dart Http请求库，支持Restful API、FormData、拦截器、请求取消、Cookie管理、文件上传/下载、超时等。dio的使用方式随着其版本升级可能会发生变化，如果本节所述内容和dio官方有差异，请以dio官方文档为准。

### 引入

引入dio:

dependencies:  
 dio: ^x.x.x #请使用pub上的最新版本

导入并创建dio实例：

import 'package:dio/dio.dart';  
Dio dio = Dio();

接下来就可以通过 dio实例来发起网络请求了，注意，一个dio实例可以发起多个http请求，一般来说，APP只有一个http数据源时，dio应该使用单例模式。

### 示例

发起 GET 请求 :

Response response;  
response=await dio.get("/test?id=12&name=wendu")  
print(response.data.toString());

对于GET请求我们可以将query参数通过对象来传递，上面的代码等同于：

response=await dio.get("/test",queryParameters:{"id":12,"name":"wendu"})  
print(response);

发起一个 POST 请求:

response=await dio.post("/test",data:{"id":12,"name":"wendu"})

发起多个并发请求:

response= await Future.wait([dio.post("/info"),dio.get("/token")]);

下载文件:

response=await dio.download("https://www.google.com/",\_savePath);

发送 FormData:

FormData formData = new FormData.from({  
 "name": "wendux",  
 "age": 25,  
});  
response = await dio.post("/info", data: formData)

如果发送的数据是FormData，则dio会将请求header的contentType设为“multipart/form-data”。

通过FormData上传多个文件:

FormData formData = new FormData.from({  
 "name": "wendux",  
 "age": 25,  
 "file1": new UploadFileInfo(new File("./upload.txt"), "upload1.txt"),  
 "file2": new UploadFileInfo(new File("./upload.txt"), "upload2.txt"),  
 // 支持文件数组上传  
 "files": [  
 new UploadFileInfo(new File("./example/upload.txt"), "upload.txt"),  
 new UploadFileInfo(new File("./example/upload.txt"), "upload.txt")  
 ]  
});  
response = await dio.post("/info", data: formData)

值得一提的是，dio内部仍然使用HttpClient发起的请求，所以代理、请求认证、证书校验等和HttpClient是相同的，我们可以在onHttpClientCreate回调中设置，例如：

(dio.httpClientAdapter as DefaultHttpClientAdapter).onHttpClientCreate = (client) {  
 //设置代理   
 client.findProxy = (uri) {  
 return "PROXY 192.168.1.2:8888";  
 };  
 //校验证书  
 httpClient.badCertificateCallback=(X509Certificate cert, String host, int port){  
 if(cert.pem==PEM){  
 return true; //证书一致，则允许发送数据  
 }  
 return false;  
 };   
 };

注意，onHttpClientCreate会在当前dio实例内部需要创建HttpClient时调用，所以通过此回调配置HttpClient会对整个dio实例生效，如果你想针对某个应用请求单独的代理或证书校验策略，可以创建一个新的dio实例即可。

怎么样，是不是很简单，除了这些基本的用法，dio还支持请求配置、拦截器等，官方资料比较详细，故本书不再赘述，详情可以参考dio主页：https://github.com/flutterchina/dio 。 下一节我们将使用dio实现一个分块下载器。

### 实例

我们通过Github开放的API来请求flutterchina组织下的所有公开的开源项目，实现：

1. 在请求阶段弹出loading
2. 请求结束后，如果请求失败，则展示错误信息；如果成功，则将项目名称列表展示出来。

代码如下：

class \_FutureBuilderRouteState extends State<FutureBuilderRoute> {  
 Dio \_dio = new Dio();  
  
 @override  
 Widget build(BuildContext context) {  
  
 return new Container(  
 alignment: Alignment.center,  
 child: FutureBuilder(  
 future: \_dio.get("https://api.github.com/orgs/flutterchina/repos"),  
 builder: (BuildContext context, AsyncSnapshot snapshot) {  
 //请求完成  
 if (snapshot.connectionState == ConnectionState.done) {  
 Response response = snapshot.data;  
 //发生错误  
 if (snapshot.hasError) {  
 return Text(snapshot.error.toString());  
 }  
 //请求成功，通过项目信息构建用于显示项目名称的ListView  
 return ListView(  
 children: response.data.map<Widget>((e) =>  
 ListTile(title: Text(e["full\_name"]))  
 ).toList(),  
 );  
 }  
 //请求未完成时弹出loading  
 return CircularProgressIndicator();  
 }  
 ),  
 );  
 }  
}