World-Time API:

Now that we know how to fetch data from an external API using the http package, we’re now ready to fetch data from the actual site which has data useful to us. This is called the ‘World Time API’ at <http://worldtimeapi.org/>.

We follow all the previous steps, in the same getData function (which we’ll now call getTime() function) in the loading.dart file we create a Response object and store the output of the response of the request to the API. Now, the response of the API varies with the URL specified. So, if we wanted to query the time of ‘Kolkata’ suppose, we’ll need to query the url as: <http://worldtimeapi.org/api/timezone/Asia/Kolkata>

If we wanted Berlin, we need to query it as:

<http://worldtimeapi.org/api/timezone/Europe/Berlin>

Thus we see that whenever we change the last two sections of the url, we get different time-zones. The url basically has the format of:

<http://worldtimeapi.org/api/timezone/Continent/City>

We jsonDecode the body of this response object and store it in a map, as we did before.

Response response= **await** get(**'http://worldtimeapi.org/api/timezone/Asia/Kolkata'**);  
Map data=jsonDecode(response.**body**);  
print(data);

We get all the data dump as:

{"abbreviation":"IST","client\_ip":"182.70.91.217","datetime":"2020-05-10T21:25:14.489452+05:30","day\_of\_week":0,"day\_of\_year":131,"dst":false,"dst\_from":null,"dst\_offset":0,"dst\_until":null,"raw\_offset":19800,"timezone":"Asia/Kolkata","unixtime":1589126114,"utc\_datetime":"2020-05-10T15:55:14.489452+00:00","utc\_offset":"+05:30","week\_number":19}

Of these properties, we are interested in 2 specific ones:

1. DateTime
2. Utc\_offset

We’ll need to add the UTC offset to the dateTime in order to get the correct local time.

So, we extract those from our data and print them:

*//get properties from data*String dateTime=data[**'datetime'**];  
String offset=data[**'utc\_offset'**];

print(dateTime);  
print(offset);

We get the output as:

I/flutter(11285): 2020-05-10T21:25:14.489452+05:30

I/flutter(11285): +05:30

Currently however, these are a bunch of strings. It would be a million times better, if they were converted into the standard DART ‘DateTime’ object. Converting them into this format makes things much easier to extract data from. We can easily tap into the hours, minutes, seconds of the time as well as add or subtract hours, minutes and seconds, which is otherwise very difficult with strings.

This process is called ‘*parsing’.* Thus, we are parsing an ordinary string into a format compatible to be stored in the DateTime object in DART. This is done with the help of the method called ‘*parse’.*

Therefore, we write:

DateTime now=DateTime.*parse*(dateTime);

Now that we have the DateTime object, we can easily add the offset to it using the inbuilt method called add(). We are going to have to add the hours and minutes separately. Therefore, we need an instance of the Duration object to be specified in the add() method.

now=now.add(Duration(hours: ,minutes:);

We’ll need to extract the hours and minutes separately from the offset string, and we do that with the help of the substring method. We need to have 2 strings—one which stores the offset hours (with the sign), and the other which stores the offset minutes.

String offsetHours=data[**'utc\_offset'**].substring(0,3);// Sign is included  
String offsetMinutes=data[**'utc\_offset'**].substring(4,6);

Now, we’re ready to add the offset to the dateTime object. We’ll need to further parse these strings into integers with the help of the int.parse() method.

Our final time looks like:

now=now.add(Duration(hours: int.*parse*(offsetHours),minutes: int.*parse*(offsetMinutes)));

The final function looks like:

Void getTime() **async**{  
 *//make the request* Response response= **await** get(**'http://worldtimeapi.org/api/timezone/Asia/Kolkata'**);  
 Map data=jsonDecode(response.**body**);  
 *//print(data);  
   
 //get properties from data* String dateTime=data[**'datetime'**];  
 String offsetHours=data[**'utc\_offset'**].substring(0,3);  
 String offsetMinutes=data[**'utc\_offset'**].substring(4,6);  
 *//print(dateTime);  
 //print(offsetHours);  
 //print(offsetMinutes);  
   
 //create dateTime object* DateTime now=DateTime.*parse*(dateTime);  
 now=now.add(Duration(hours: int.*parse*(offsetHours),minutes: int.*parse*(offsetMinutes)));  
 print(now.**hour**);

}

The WorldTime Custom Class:

We have written quite a bit of logic in the getTime function which currently exists in loading.dart. In order to make our program more modular, we’ll have to place this logic into a separate place so that it can be reused by other classes.

We create a new package called ‘services’ into which we’ll add a new dart file called world\_time.dart.

We’ll need to import the http package as well as the dart:convert package so as to jsonconvert the data:

**import 'package:http/http.dart'**;  
**import 'dart:convert'**;

We’ll create a new class and call it WorldTime. It will have the following properties:

**class** WorldTime{  
  
 String **location**; *//location name for UI* String **time**;*//time in that location* String **flag**; *//url to a flag icon* String **url**;*//location url for API endpoint* WorldTime({**this**.**location**, **this**.**flag**, **this**.**url**});

}

Now, we bring up the getTime function from the loading.dart file, and place it here in this class. Also, instead of hard-coding the location, we’ll pass the url property to the http request:

**class** WorldTime{  
  
 String **location**; *//location name for UI* String **time**;*//time in that location* String **flag**; *//url to a flag icon* String **url**;*//location url for API endpoint* String **eventOfDay**;*//decides if morning,eveninng,night,etc*  
  
 **void** getTime() **async** {  
 *//make the request*  Response response= **await** get(**'http://worldtimeapi.org/api/timezone/**$**url'**);  
 Map data=jsonDecode(response.**body**);  
 *//print(data);  
   
 //get properties from data* String dateTime=data[**'datetime'**];  
 String offsetHours=data[**'utc\_offset'**].substring(0,3);  
 String offsetMinutes=data[**'utc\_offset'**].substring(4,6);  
 *//print(dateTime);  
 //print(offsetHours);  
 //print(offsetMinutes);  
   
 //create dateTime object* DateTime now=DateTime.*parse*(dateTime);  
 now=now.add(Duration(hours: int.*parse*(offsetHours),minutes: int.*parse*(offsetMinutes)));  
 *//print(now.hour);*

}

Now, if we look at the class, we see that the time property is a string. But what we’ve got here at the end of the getTime() function is an instance of the DateTime class with the name ‘now’. We’ll need to convert it into a string. So, we simply add a now.toString() and set it equal to Time:

*//set the time property*

**time**=now.toString();

Now, we’ll have a basic idea of how our WorldTime class works. Whenever we create an instance of this class, we’ll need to pass in the ‘location’, the ‘url’ and the ‘flag’ property while instantiating the class. After receiving these properties, if we call the getTime function, the value of the ‘time’ property will be set.

So, we appropriately create a constructor for it:

WorldTime({**this**.**location**, **this**.**flag**, **this**.**url**});

Our class now looks like:

**import 'package:http/http.dart'**;  
**import 'dart:convert'**;  
**import 'package:intl/intl.dart'**;  
  
**class** WorldTime{  
  
 String **location**; *//location name for UI* String **time**;*//time in that location* String **flag**; *//url to a flag icon* String **url**;*//location url for API endpoint* String **eventOfDay**;*//decides if morning,eveninng,night,etc* WorldTime({**this**.**location**, **this**.**flag**, **this**.**url**});  
  
  
 **void** getTime() **async** {  
 *//make the request* Response response= **await** get(**'http://worldtimeapi.org/api/timezone/**$**url'**);  
 Map data=jsonDecode(response.**body**);  
 *//print(data);  
   
 //get properties from data* String dateTime=data[**'datetime'**];  
 String offsetHours=data[**'utc\_offset'**].substring(0,3);  
 String offsetMinutes=data[**'utc\_offset'**].substring(4,6);  
 *//print(dateTime);  
 //print(offsetHours);  
 //print(offsetMinutes);  
   
 //create dateTime object* DateTime now=DateTime.*parse*(dateTime);  
 now=now.add(Duration(hours: int.*parse*(offsetHours),minutes: int.*parse*(offsetMinutes)));  
 *//print(now.hour);  
 //set the time property*   
 **time**=now.toString();

}  
  
 }

Now that we’re done with the class, we head on to the loading screen, and then import this class into the loading.dart file:

**import 'package:chronograph/services/world\_time.dart'**;

Inside the loading screen, we’ll create a new async function called ‘setupWorldTime’ which will initialise what time the app shows when the user first loads the app. This will of course be hard coded. In here, we’ll create a new instance of the WorldTime class, and specify the properties like ‘location’, ‘url’ and ‘flag’, then we’ll call the getTime function in an awaited manner in order to set the time property of the instance. The ‘await’ here is important as we don’t want anything to be executed before we fetch the time from the API.

**void** setupWorldTime() **async**{  
 WorldTime instance=WorldTime(location: **'Kolkata'**,flag: **'india.png'**,url: **'Asia/Kolkata'**);  
 **await** instance.getTime();  
   
}

Now that we’re awaiting the getTime() method, and if we want to use the await keyword, we need to remember that the request can also return an error. In that case, we’re not sure of what datatype our request returns. This datatype will therefore be resolved in the future—when the request completes. Since we’re unaware of the returned datatype here, we’ll wrap the void return type of the getTime function with the future keyword. It just signifies that—at the time of writing the code, the return type is unknown, *and will be resolved in the future.* So, the current return type is Future which is a placeholder value until the function is complete.

Future <**void**> getTime() **async**{

*//make the request* Response response= **await** get(**'http://worldtimeapi.org/api/timezone/**$**url'**);  
 Map data=jsonDecode(response.**body**);  
 *//print(data);  
   
 //get properties from data* String dateTime=data[**'datetime'**];  
 String offsetHours=data[**'utc\_offset'**].substring(0,3);  
 String offsetMinutes=data[**'utc\_offset'**].substring(4,6);  
 *//print(dateTime);  
 //print(offsetHours);  
 //print(offsetMinutes);  
   
 //create dateTime object* DateTime now=DateTime.*parse*(dateTime);  
 now=now.add(Duration(hours: int.*parse*(offsetHours),minutes: int.*parse*(offsetMinutes)));  
 *//print(now.hour);  
 //set the time property*   
 **time**=now.toString();

}

We’ll call up this method inside the initState() method of the loading screen:

@override  
**void** initState() {  
 **super**.initState();  
 setupWorldTime();  
}

Now, as a test, we can actually print the obtained time on the UI in a text widget. In order to do that, we’ll create a dummy string whose initial value will be ‘loading’ and after the getTime function runs, we’ll fill it in with the time obtained.

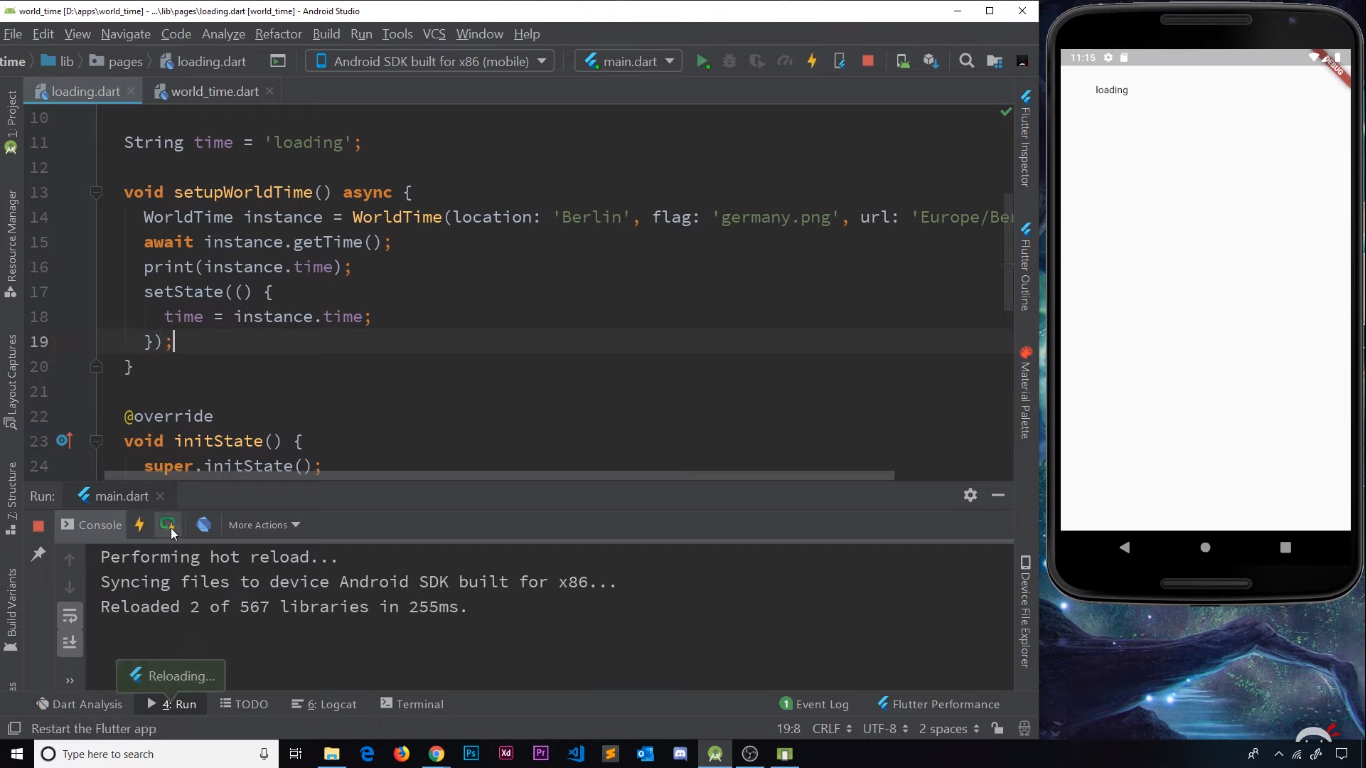
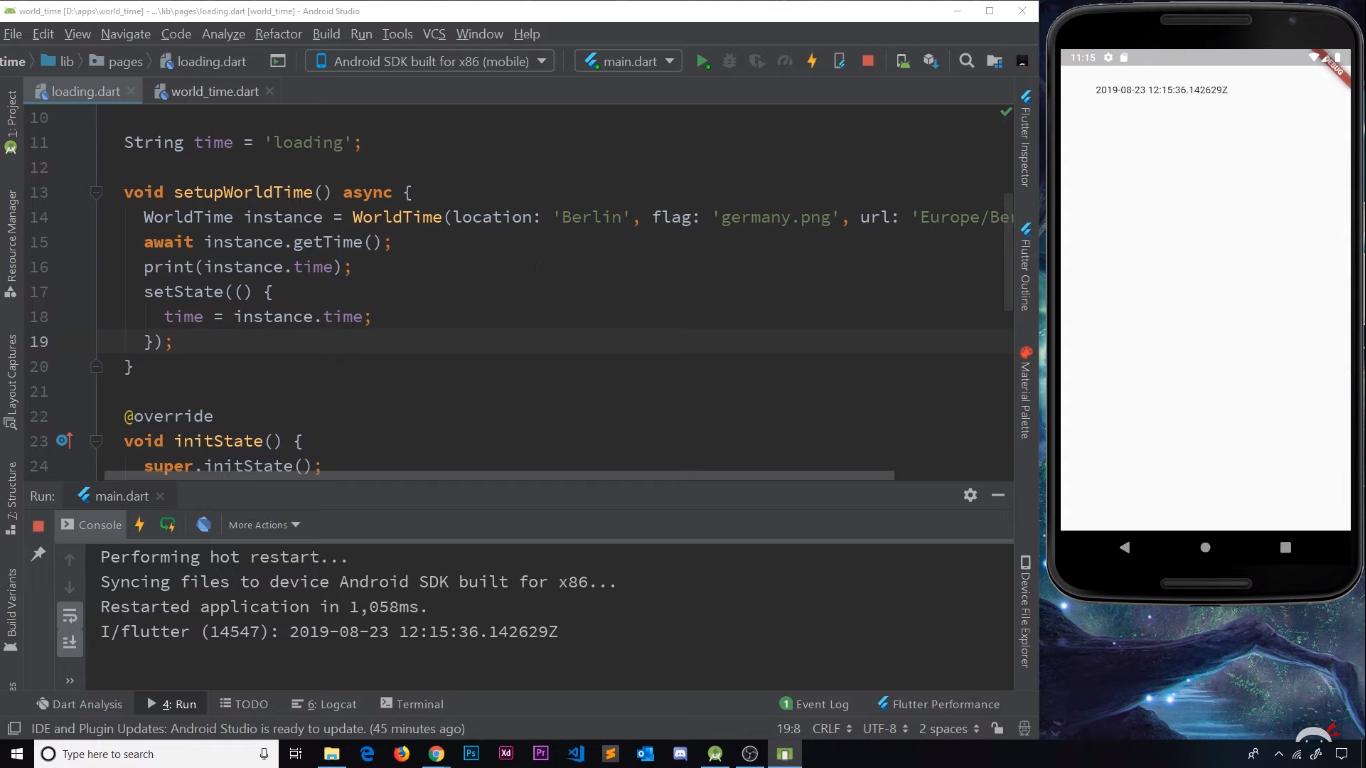
**String** time= ‘Loading’;

**void** setupWorldTime() **async**{  
 WorldTime instance=WorldTime(location: **'Kolkata'**,flag: **'india.png'**,url: **'Asia/Kolkata'**);  
 **await** instance.getTime();

setState((){

time=instance.getTime();

});  
 }

Error Handling:

There might be cases where an incorrect URL for the time zone will cause the entire app to crash. Because the appropriate JSON file is not found, we’ll not be able to map it to DART map, then the dateTime will not be created, and finally the time property cannot be set. In such cases we’ll need to manually handle the error. We do this with a simple try/catch block.

We’ll just update the ‘time’ text, to an error message it the catch block.

**try** {

//code of getTime function

}

**catch** (e) {  
 print(**'Caught Error** $e**'**);  
 **time**=**'Failed to get the time data'**;  
}

So, if we have some wrong url, we’ll see:

