# 使用WebSockets

Http协议是无状态的，只能由客户端主动发起，服务端再被动响应，服务端无法向客户端主动推送内容，并且一旦服务器响应结束，链接就会断开(见注解部分)，所以无法进行实时通信。WebSocket协议正是为解决客户端与服务端实时通信而产生的技术，现在已经被主流浏览器支持，所以对于Web开发者来说应该比较熟悉了，Flutter也提供了专门的包来支持WebSocket协议。

注意：Http协议中虽然可以通过keep-alive机制使服务器在响应结束后链接会保持一段时间，但最终还是会断开，keep-alive机制主要是用于避免在同一台服务器请求多个资源时频繁创建链接，它本质上是支持链接复用的技术，而并非用于实时通信，读者需要知道这两者的区别。

WebSocket协议本质上是一个基于tcp的协议，它是先通过HTTP协议发起一条特殊的http请求进行握手后，如果服务端支持WebSocket协议，则会进行协议升级。WebSocket会使用http协议握手后创建的tcp链接，和http协议不同的是，WebSocket的tcp链接是个长链接（不会断开），所以服务端与客户端就可以通过此TCP连接进行实时通信。有关WebSocket协议细节，读者可以看RFC文档，下面我们重点看看Flutter中如何使用WebSocket。

在接下来例子中，我们将连接到由[websocket.org提供的测试服务器](http://www.websocket.org/echo.html)。服务器将简单地返回我们发送给它的相同消息！

### 步骤

1. 连接到WebSocket服务器。
2. 监听来自服务器的消息。
3. 将数据发送到服务器。
4. 关闭WebSocket连接。

### 1. 连接到WebSocket服务器

[web\_socket\_channel](https://pub.dartlang.org/packages/web_socket_channel) package 提供了我们需要连接到WebSocket服务器的工具。该package提供了一个WebSocketChannel允许我们既可以监听来自服务器的消息，又可以将消息发送到服务器的方法。

在Flutter中，我们可以创建一个WebSocketChannel连接到一台服务器：

final channel = IOWebSocketChannel.connect('ws://echo.websocket.org');

### 2. 监听来自服务器的消息

现在我们建立了连接，我们可以监听来自服务器的消息，在我们发送消息给测试服务器之后，它会返回相同的消息。

我们如何收取消息并显示它们？在这个例子中，我们将使用一个[StreamBuilder](https://docs.flutter.io/flutter/widgets/StreamBuilder-class.html) 来监听新消息， 并用一个Text来显示它们。

new StreamBuilder(  
 stream: widget.channel.stream,  
 builder: (context, snapshot) {  
 return new Text(snapshot.hasData ? '${snapshot.data}' : '');  
 },  
);

#### 工作原理

WebSocketChannel提供了一个来自服务器的消息Stream 。该Stream类是dart:async包中的一个基础类。它提供了一种方法来监听来自数据源的异步事件。与Future返回单个异步响应不同，Stream类可以随着时间推移传递很多事件。该[StreamBuilder](https://docs.flutter.io/flutter/widgets/StreamBuilder-class.html) 组件将连接到一个Stream， 并在每次收到消息时通知Flutter重新构建界面。

### 3. 将数据发送到服务器

为了将数据发送到服务器，我们会add消息给WebSocketChannel提供的sink。

channel.sink.add('Hello!');

#### 工作原理

WebSocketChannel提供了一个[StreamSink](https://docs.flutter.io/flutter/dart-async/StreamSink-class.html)，它将消息发给服务器。

StreamSink类提供了给数据源同步或异步添加事件的一般方法。

### 4. 关闭WebSocket连接

在我们使用WebSocket后，要关闭连接：

channel.sink.close();

### 完整的例子

import 'package:flutter/material.dart';  
import 'package:web\_socket\_channel/io.dart';  
  
class WebSocketRoute extends StatefulWidget {  
 @override  
 \_WebSocketRouteState createState() => new \_WebSocketRouteState();  
}  
  
class \_WebSocketRouteState extends State<WebSocketRoute> {  
 TextEditingController \_controller = new TextEditingController();  
 IOWebSocketChannel channel;  
 String \_text = "";  
  
  
 @override  
 void initState() {  
 //创建websocket连接  
 channel = new IOWebSocketChannel.connect('ws://echo.websocket.org');  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return new Scaffold(  
 appBar: new AppBar(  
 title: new Text("WebSocket(内容回显)"),  
 ),  
 body: new Padding(  
 padding: const EdgeInsets.all(20.0),  
 child: new Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: <Widget>[  
 new Form(  
 child: new TextFormField(  
 controller: \_controller,  
 decoration: new InputDecoration(labelText: 'Send a message'),  
 ),  
 ),  
 new StreamBuilder(  
 stream: channel.stream,  
 builder: (context, snapshot) {  
 //网络不通会走到这  
 if (snapshot.hasError) {  
 \_text = "网络不通...";  
 } else if (snapshot.hasData) {  
 \_text = "echo: "+snapshot.data;  
 }  
 return new Padding(  
 padding: const EdgeInsets.symmetric(vertical: 24.0),  
 child: new Text(\_text),  
 );  
 },  
 )  
 ],  
 ),  
 ),  
 floatingActionButton: new FloatingActionButton(  
 onPressed: \_sendMessage,  
 tooltip: 'Send message',  
 child: new Icon(Icons.send),  
 ),  
 );  
 }  
  
 void \_sendMessage() {  
 if (\_controller.text.isNotEmpty) {  
 channel.sink.add(\_controller.text);  
 }  
 }  
  
 @override  
 void dispose() {  
 channel.sink.close();  
 super.dispose();  
 }  
}

上面的例子比较简单，不再赘述。我们现在思考一个问题，假如我们想通过WebSocket传输二进制数据应该怎么做（比如要从服务器接收一张图片）？我们发现StreamBuilder和Stream都没有指定接收类型的参数，并且在创建WebSocket链接时也没有相应的配置，貌似没有什么办法……其实很简单，要接收二进制数据仍然使用StreamBuilder，因为WebSocket中所有发送的数据使用帧的形式发送，而帧是有固定格式，每一个帧的数据类型都可以通过Opcode字段指定，它可以指定当前帧是文本类型还是二进制类型（还有其它类型），所以客户端在收到帧时就已经知道了其数据类型，所以flutter完全可以在收到数据后解析出正确的类型，所以就无需开发者去关心，当服务器传输的数据是指定为二进制时，StreamBuilder的snapshot.data的类型就是List<int>，是文本时，则为String。