Get started Samples & tutorials Flutter Gallery [running app] Flutter Gallery [repo] Sample apps on GitHub ☑ Cookbook **Codelabs** <u>Tutorials</u> <u>Development</u> User interface Data & backend Accessibility & internationalization Platform integration Packages & plugins Add Flutter to existing app Tools & features Migration notes Testing & debugging Performance & optimization **Deployment** <u>Resources</u>

Who is Dash?

<u>Reference</u>

<u>Widget index</u>

API reference ☑

flutter CLI reference

Package site

Update data over the internet

Work with WebSockets

Cookbook > Networking > Work with WebSockets

Contents

- 1. Connect to a WebSocket server
- 2. Listen for messages from the server

How this works

3. Send data to the server

How this works

4. Close the WebSocket connection

Complete example

In addition to normal HTTP requests, you can connect to servers using WebSockets. WebSockets allow for two-way communication with a server without polling.

Persist data with SQLite

In this example, connect to a <u>test server provided by websocket.org</u>. The server sends back the same message you send to it. This recipe uses the following steps:

- 1. Connect to a WebSocket server.
- 2. Listen for messages from the server.
- 3. Send data to the server.
- 4. Close the WebSocket connection.

1. Connect to a WebSocket server

The web_socket_channel package provides the tools you need to connect to a WebSocket server.

The package provides a WebSocketChannel that allows you to both listen for messages from the server and push messages to the server.

In Flutter, use the following line to create a WebSocketChannel that connects to a server:

```
final channel = WebSocketChannel.connect(
   Uri.parse('wss://echo.websocket.org'),
);
```

2. Listen for messages from the server

Now that you've established a connection, listen to messages from the server.

After sending a message to the test server, it sends the same message back.

In this example, use a <u>StreamBuilder</u> widget to listen for new messages, and a <u>Text</u> widget to display them.

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

How this works

The WebSocketChannel provides a <u>Stream</u> of messages from the server.

The Stream class is a fundamental part of the dart:async package. It provides a way to listen to async events from a data source. Unlike Future, which returns a single async response, the Stream class can deliver many events over time.

The <u>StreamBuilder</u> widget connects to a <u>Stream</u> and asks Flutter to rebuild every time it receives an event using the given builder function.

<u>Get started</u>	~
Samples & tutorials	_
Flutter Gallery [running app] 🛮	
Flutter Gallery [repo] 🗷	
Sample apps on GitHub ☑	
Cookbook	
<u>Codelabs</u>	
<u>Tutorials</u>	
<u>Development</u>	^
▶ <u>User interface</u>	
▶ <u>Data & backend</u>	
▶ <u>Accessibility & internationalization</u>	
▶ <u>Platform integration</u>	
▶ <u>Packages & plugins</u>	
▶ Add Flutter to existing app	
▶ <u>Tools & features</u>	
▶ <u>Migration notes</u>	
Testing & debugging	~
Performance & optimization	~
<u>Deployment</u>	~
Resources	~
<u>Reference</u>	^
Who is Dash?	
<u>Widget index</u>	
API reference 🗷	

flutter CLI reference

Package site

3. Send data to the server

To send data to the server, add() messages to the sink provided by the WebSocketChannel.

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

How this works

The WebSocketChannel provides a <u>StreamSink</u> to push messages to the server.

The StreamSink class provides a general way to add sync or async events to a data source.

4. Close the WebSocket connection

After you're done using the WebSocket, close the connection:

```
channel.sink.close();
```

Complete example

```
<u>Get started</u>
<u>Samples & tutorials</u>
  Flutter Gallery [running app] 🗷
  Flutter Gallery [repo]
  Sample apps on GitHub ☑
  Cookbook
  <u>Codelabs</u>
  <u>Tutorials</u>
                                             ^
<u>Development</u>
▶ <u>User interface</u>
Data & backend

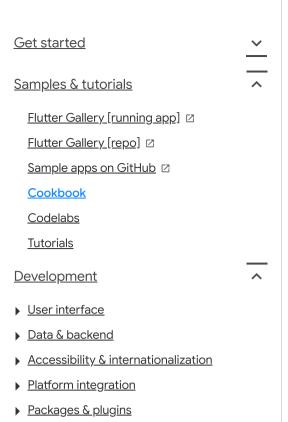
    Accessibility & internationalization

▶ Platform integration
Packages & plugins

    Add Flutter to existing app

Tools & features
Migration notes
Testing & debugging
Performance & optimization
<u>Deployment</u>
Resources
Reference
  Who is Dash?
  Widget index
  <u>API reference</u> ☑
  flutter CLI reference
  Package site ☑
```

```
import 'package:web_socket_channel/web_socket_channel.dart';
import 'package:flutter/material.dart';
void main() => runApp(const MyApp());
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    const title = 'WebSocket Demo';
    return const MaterialApp(
      title: title,
      home: MyHomePage(
        title: title,
      ),
    );
  }
class MyHomePage extends StatefulWidget {
  const MyHomePage({
    Key? key,
    required this.title,
  }) : super(key: key);
  final String title;
  @override
  _MyHomePageState createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
  final TextEditingController _controller = TextEditingController();
  final _channel = WebSocketChannel.connect(
    Uri.parse('wss://echo.websocket.org'),
  );
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Form(
              child: TextFormField(
                controller: _controller,
                decoration: const InputDecoration(labelText: 'Send a message'),
              ),
            ),
            const SizedBox(height: 24),
            StreamBuilder(
              stream: _channel.stream,
              builder: (context, snapshot) {
                return Text(snapshot.hasData ? '${snapshot.data}' : '');
              },
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _sendMessage,
        tooltip: 'Send message',
        child: const Icon(Icons.send),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
  void _sendMessage() {
    if (_controller.text.isNotEmpty) {
      _channel.sink.add(_controller.text);
  }
  @override
  void dispose() {
    _channel.sink.close();
    _controller.dispose();
    super.dispose();
}
```



0 **▼** 4 7 2:11 **WebSocket Demo** Hello Flutter Glitter Flutter Fluttering $q^1 w^2 e^3 r^4 t^5 y^6 u^7 i^8 o^9 p^0$ $a^{\circ} s^{\dagger} d^{\circ} f^{\overline{}} g^{\circ} h^{\overline{}} j^{\dagger} k^{\circ} l^{\circ}$ z x c v b n m 🕰 ?123 lacksquare

(<u>Update data over the internet</u>

Persist data with SQLite >



▶ Tools & features ▶ Migration notes

Add Flutter to existing app

flutter-dev@・terms・ brand usage・ security・ privacy・ español・ 社区中文资源・ 한국어・ We stand in solidarity with the Black community. Black Lives Matter.

Except as otherwise noted, this work is licensed under a Creative Commons Attribution 4.0 International License, and code samples are licensed under the BSD License.

Resources

<u>Reference</u>

Who is Dash?

Widget index

<u>API reference</u> ☑

flutter CLI reference

Package site