# FutureBuilder<T>

<https://stackoverflow.com/questions/51983011/flutter-future-builder-when-should-i-use-it>

<https://stackoverflow.com/questions/56359049/setting-provider-value-in-futurebuilder>

Widget that builds itself based on the latest snapshot of interaction with a Future.

FutureBuilder **removes** some of the **boilerplate codes**.

Lets say you want to fetch data from backend on launch of page and show loader till data comes.

**Tasks for ListBuilder:**

* have two state variables 1.dataFromBackend 2.isLoadingFlag
* On launch, set isLoadingFlag = true and based on which show loader.
* Once data arrival, set data with what you get from backend and set isLoadingFlag = false (inside setState obviously)
* We need to have a if-else in widget creation. If isLoadingFlag is true, show loader else show the data. If failure, show error message.

**Tasks for FutureBuilder:**

* give the async task in future of Future Builder
* based on connectionState, show message (loading, active(streams), done)
* based on data(snapshot.hasError) show view

**Pros of FutureBuilder**

* no two flags and no setState
* reactive programming (FutureBuilder will take care of updating the view on data arrival)

**Example:**

new FutureBuilder<String>(

future: \_fetchNetworkCall, // async work

builder: (BuildContext context, AsyncSnapshot<String> snapshot) {

switch (snapshot.connectionState) {

case ConnectionState.waiting: return new Text('Loading....');

default:

if (snapshot.hasError)

return new Text('Error: ${snapshot.error}');

else

return new Text('Result: ${snapshot.data}');

}

},

)

**Performance impact:**

I just looked into the FutureBuilder code to understand the **performance** impact of using this.

* FutureBuilder is just a StatefulWidget whose state variable is \_snapshot
* Intial state is \_snapshot = new AsyncSnapshot<T>.withData(ConnectionState.none, widget.initialData);
* It is subscribing to future which we send in constructor and updating the state based on that.
* widget.future.then<void>((T data) {
* if (\_activeCallbackIdentity == callbackIdentity) {
* setState(() {
* \_snapshot = new AsyncSnapshot<T>.withData(ConnectionState.done, data);
* });
* }
* }, onError: (Object error) {
* if (\_activeCallbackIdentity == callbackIdentity) {
* setState(() {
* \_snapshot = new AsyncSnapshot<T>.withError(ConnectionState.done, error);
* });
* }

});

So the FutureBuilder is a wrapper/boilerplate of what we do typically. So there should not be any performance impact.