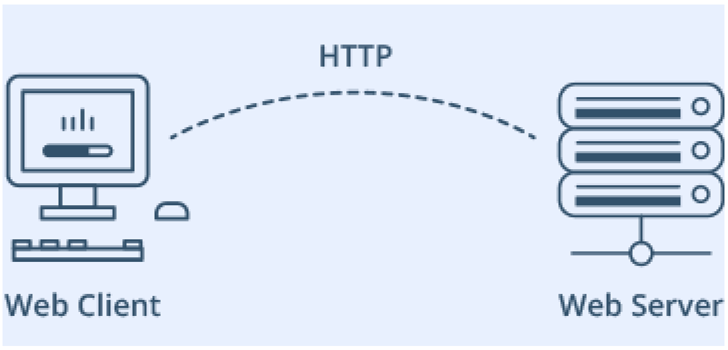
Hands on session 10: ESP32 Web Server

Web server and Web client

Web server is a place which stores, processes and delivers web pages to Web clients. Web client is nothing but a web browser on our laptops and smartphones. The communication between client and server takes place using a special protocol called Hypertext Transfer Protocol (HTTP).



In this protocol, a client initiates communication by making a request for a specific web page using HTTP and the server responds with the content of that web page or an error message if unable to do so (like famous 404 Error). Pages delivered by a server are mostly HTML documents.

How web server works?

- **Listens**
- **On a port**
- **For a request**
- **Sent via transport protocol**
- **Returns a response**
- **Containing requested resource**

Once a webserver application starts up it just sits there idle waiting for incoming requests. On a given port.

Default port for

- **HTTP:** 80
- **HTTPS:** 443

To work with web server on ESP-32 following libraries come in handy

- **Wifi.h** -> provides specific wifi methods to connect with the network
- **WebServer.h ->** provides methods to set up server and handle requests

To start working with WebServer.h we need to make an object of WebServer library

WebServer server(80);

Since 80 is the default port for HTTP, we will use this value. Now you can access the server without needing to specify the port in the URL.

In order to **handle incoming HTTP requests**, we need to specify which code to execute when a particular URL is hit. To do so, we use **on** method.

server.on("/",rootHandler);

This method takes two parameters.

1. URL path
2. Name of function which we want to execute when that URL is hit.

If the client requests any URL other than specified with **server.on().** It should respond with an HTTP status 404 (Not Found) and a message for the user.

We put this in a function as well, and use

Server.onNotFound(handlerFunction);

to tell it that it should execute it when it receives a request for a URL that wasn't specified with **server.on()**

To handle the actual incoming HTTP requests, we need to call the

handleClient();

method on the server object.

In order to respond to the HTTP request, we use the **send** method. Although the method can be called with a different set of arguments, its simplest form consists of the HTTP response code, the content type and the content.

server.send(200, "text/html", ResponseHTMLpage);

1. 200 (one of the HTTP status codes), which corresponds to the **OK** response.
2. specifying the content type as “text/html“,
3. Response HTML page.