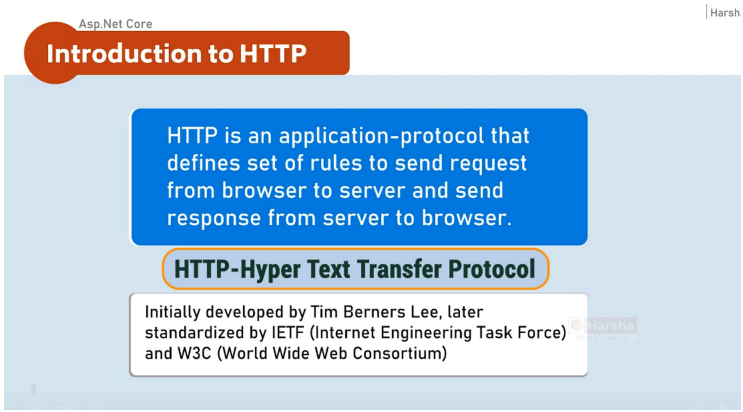
Section 3: HTTP [MVC and Web API]

29 December 2024 14:51

1. Introduction to HTTP



Asp.Net Core

Introduction to HTTP

HTTP is an application-protocol that defines set of rules to send request from browser to server and send response from server to browser.

Initially developed by Tim Berners Lee, later standardized by IETF (Internet Engineering Task Force) and W3C (World Wide Web Consortium)

Asp.Net Core

Introduction to HTTP

HTTP is an application-protocol that defines set of rules to send request from browser to server and send response from server to browser.

SSL-Security Socket Layer

Initially developed by Tim Berners Lee, later standardized by IETF (Internet Engineering Task Force) and W3C (World Wide Web Consortium)

Harsha

Asp.Net Core

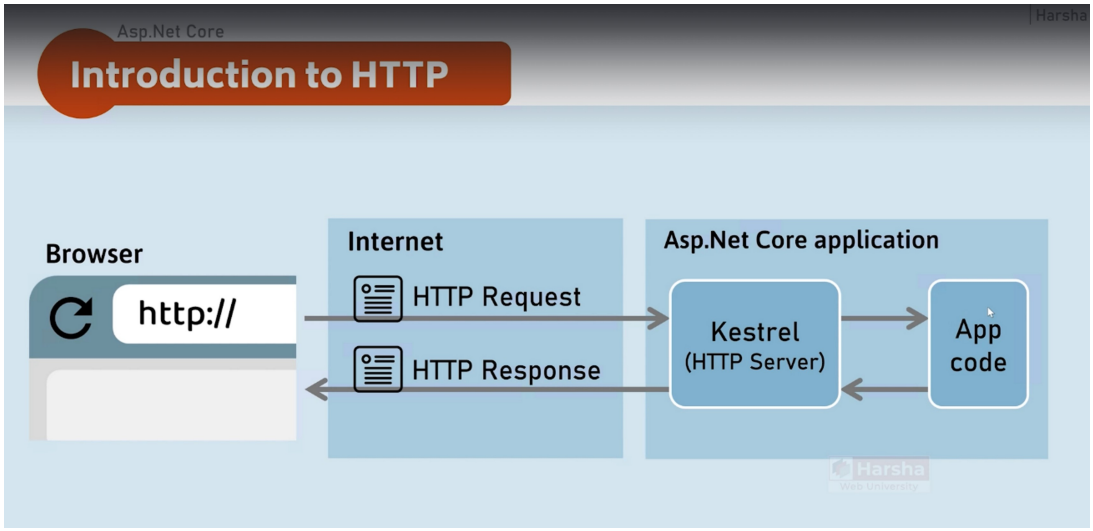
Introduction to HTTP

HTTP is an application-protocol that defines set of rules to send request from browser to server and send response from server to browser.

HTTPS

Initially developed by Tim Berners Lee, later standardized by IETF (Internet Engineering Task Force) and W3C (World Wide Web Consortium)

Harsha



If you can access websites on the internet, it is all through the HTTP protocol, which stands for Hypertext Transfer Protocol. It is a set of rules that enables browsers to send requests and servers to respond. Clients (browsers) send requests to servers, and servers receive those requests, process them, and send responses back to the browser. It's like a question-and-answer process: the browser asks a question (request), the server processes it, and provides an answer (response).

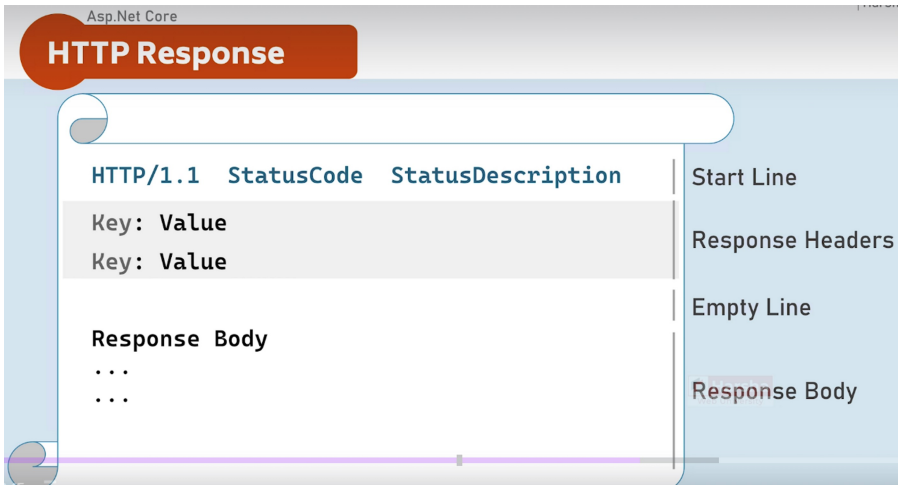
All websites on the internet use the HTTP protocol, with or without SSL (Secure Socket Layer). HTTP with SSL certificates is called HTTPS, which adds a security layer. HTTP was developed by Tim Berners-Lee in the 1990s and later standardized by the IETF for global use.

When you run an application, the browser sends an HTTP request to the server. This request follows specific rules, hence it's called an HTTP request. The request is received by the HTTP server (Kestrel), also known as the

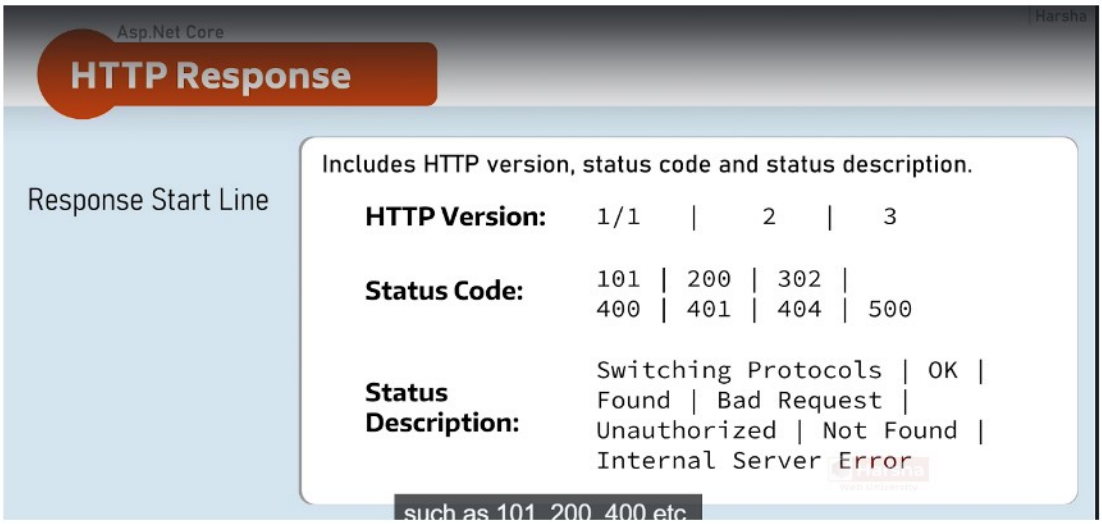
application server, which then invokes the application code. For example, a piece of code executes upon receiving a request at the root level (e.g., localhost port number). A string response is generated and sent back to the browser.

First, the response is received by Kestrel, which forwards it to the browser. This process is how the application runs every time, and it is also how all internet websites function.

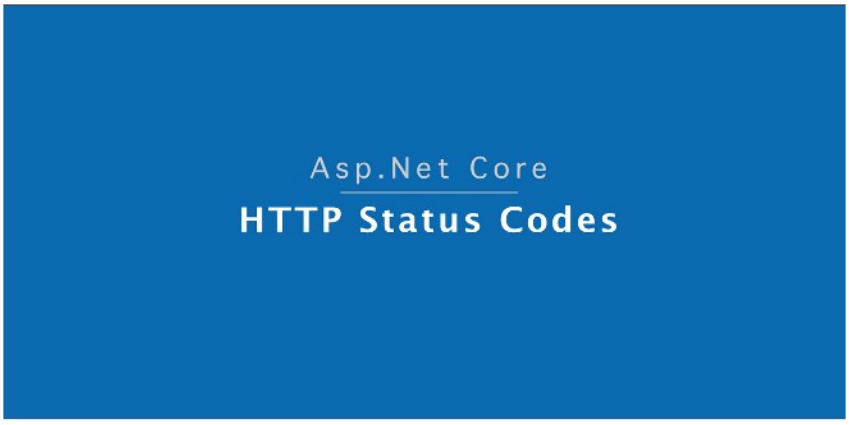
2. HTTP Response

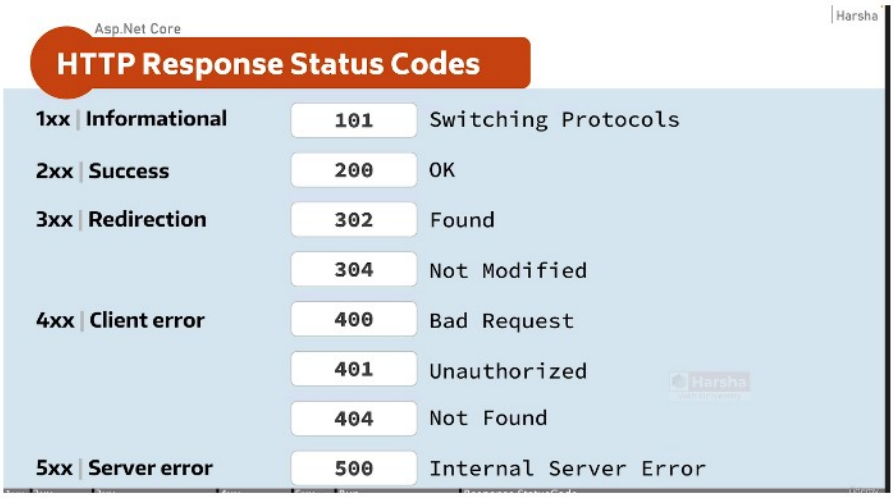


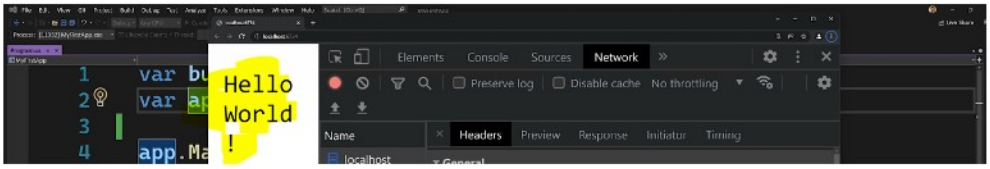
```
1  var builder = WebApplication.CreateBuilder(args);
2  var app = builder.Build();
3
4  app.MapGet("/", () => "Hello World!");
5
6  app.Run();
7
```

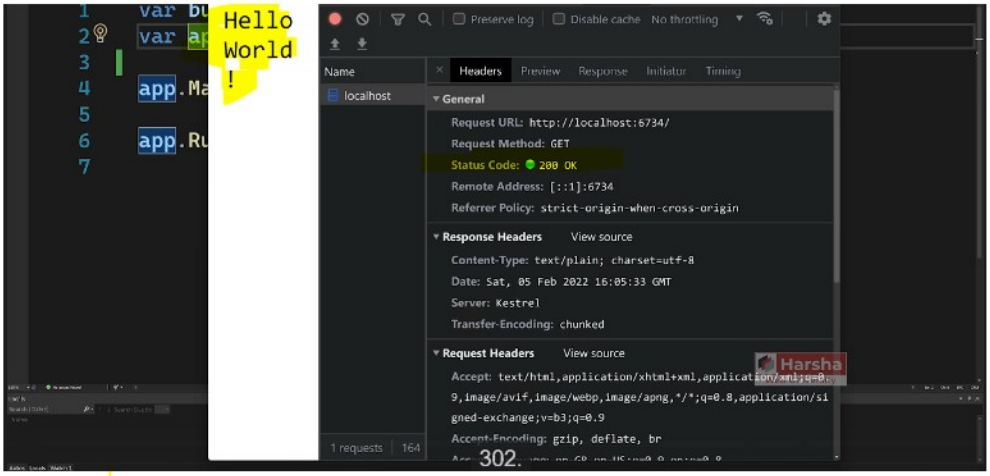


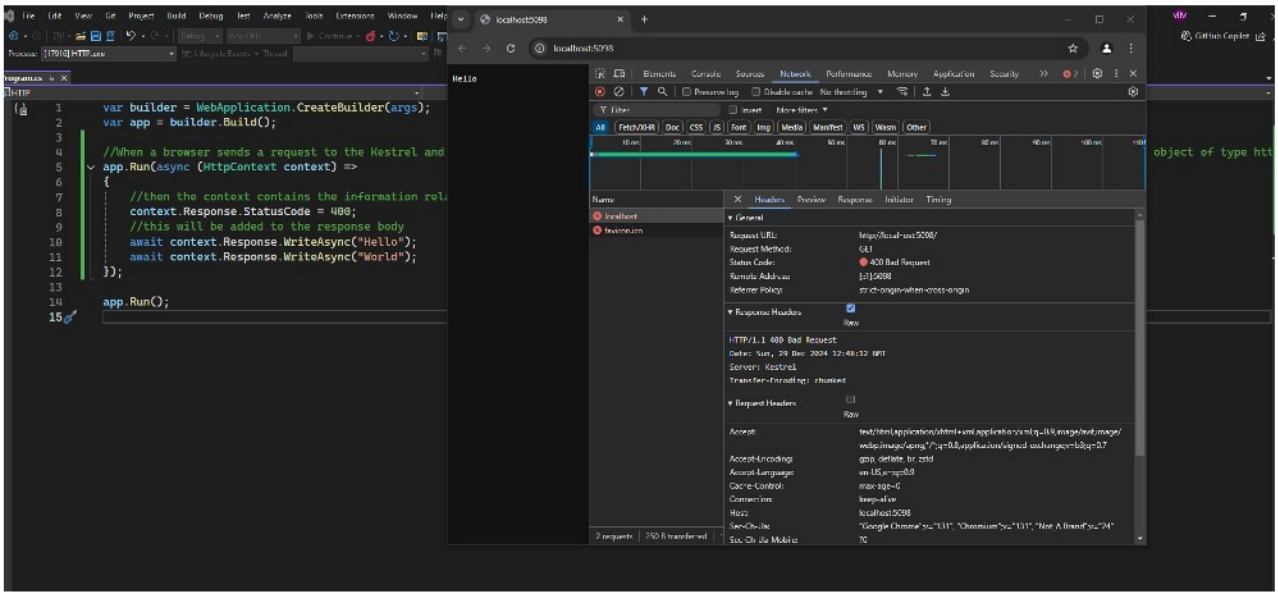
3. HTTP Status Code







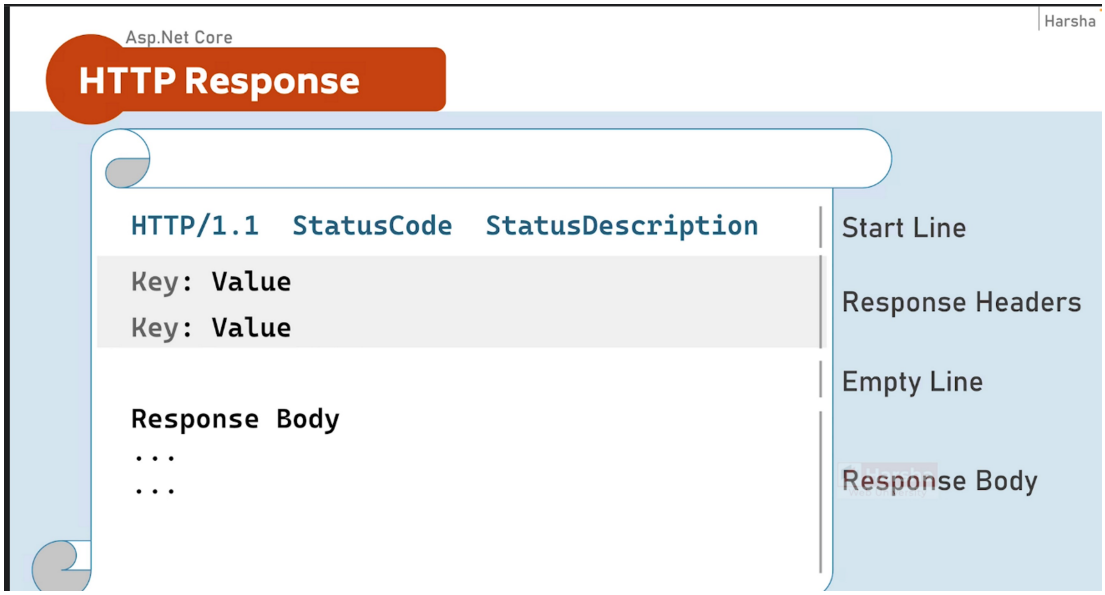


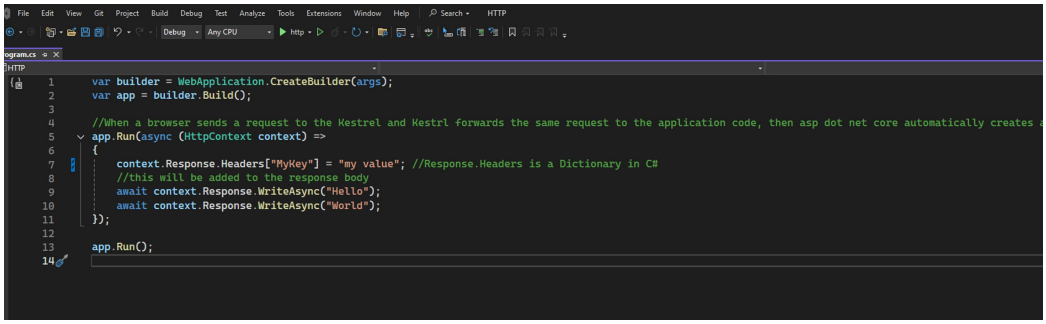


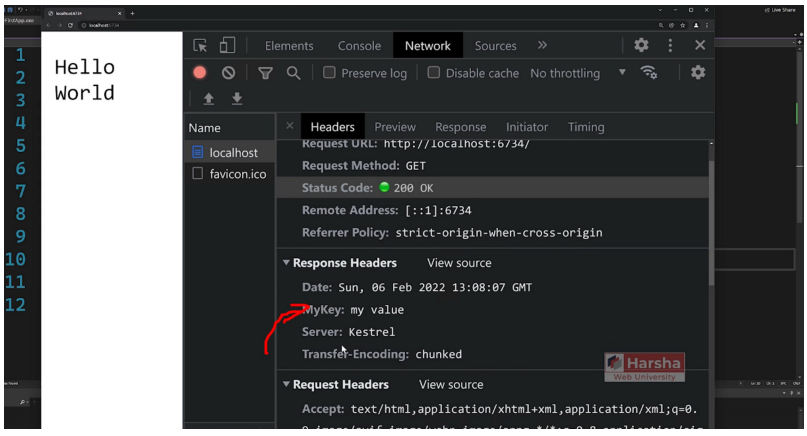
HTTP Response Headers

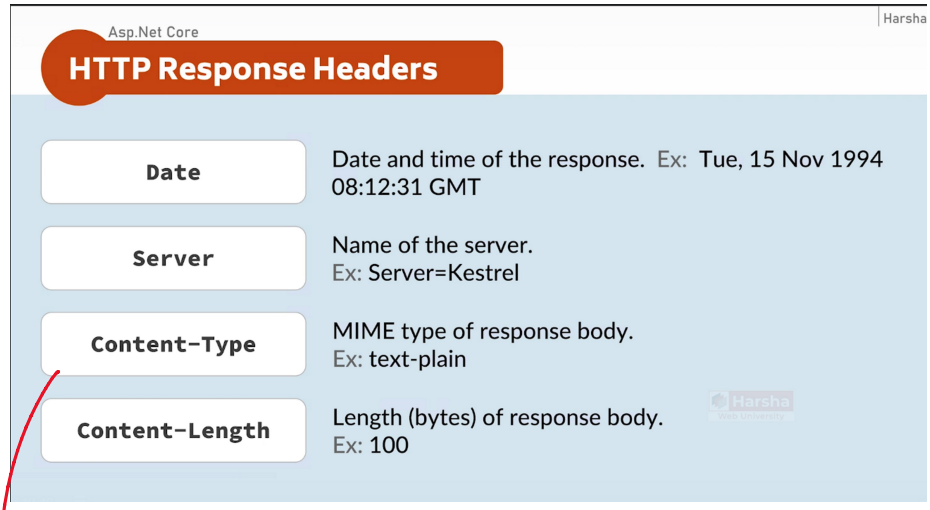
HTTP response headers are key-value pairs sent by the server to the client. They provide information about the response, such as its type and how it should be stored and managed on the client. These headers are typically not visible to the end user but are intended for information exchange between the server and client.

In this lecture, we will explore common response headers you may encounter and learn how to send response headers from an ASP.NET Core application.

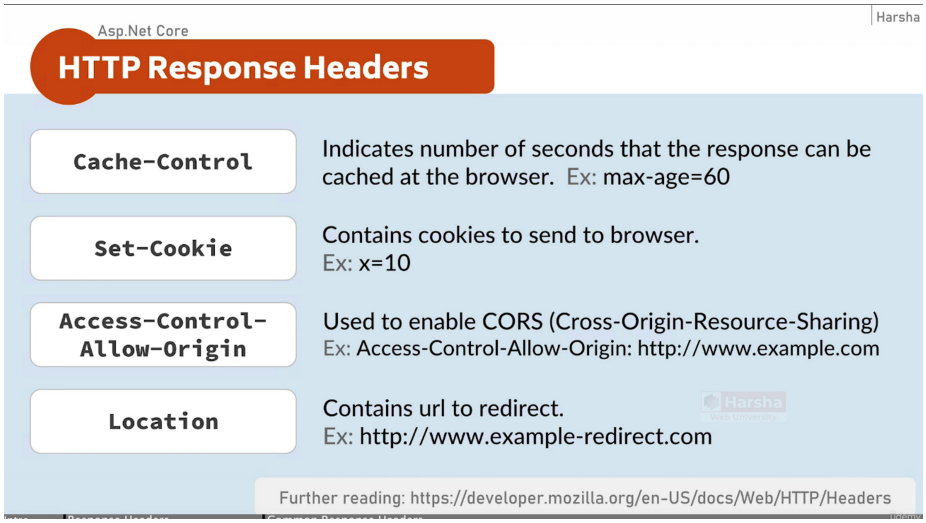




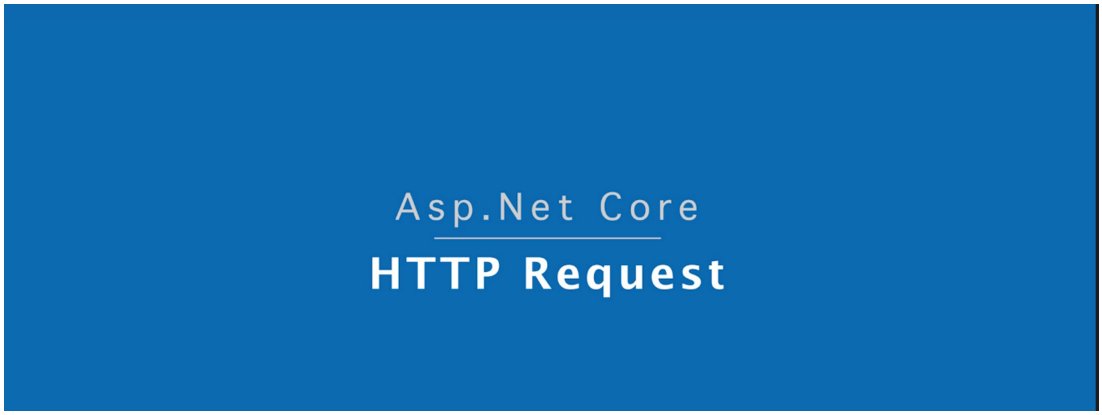




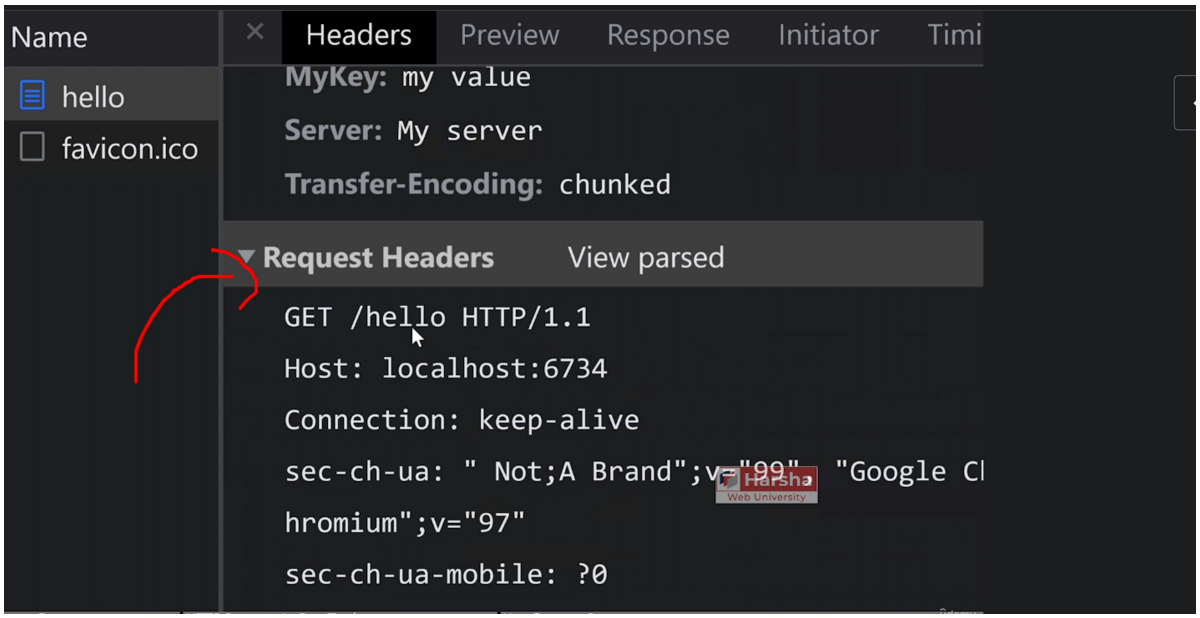
What type of response we are sending from server to the client



HTTP Request

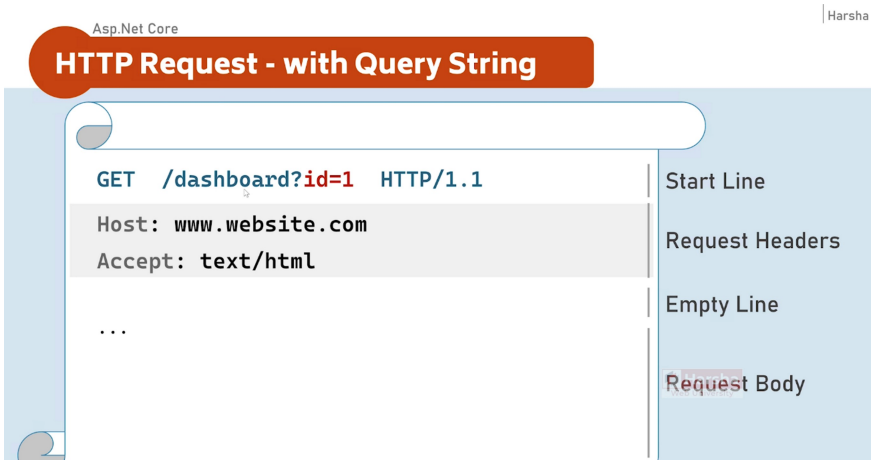


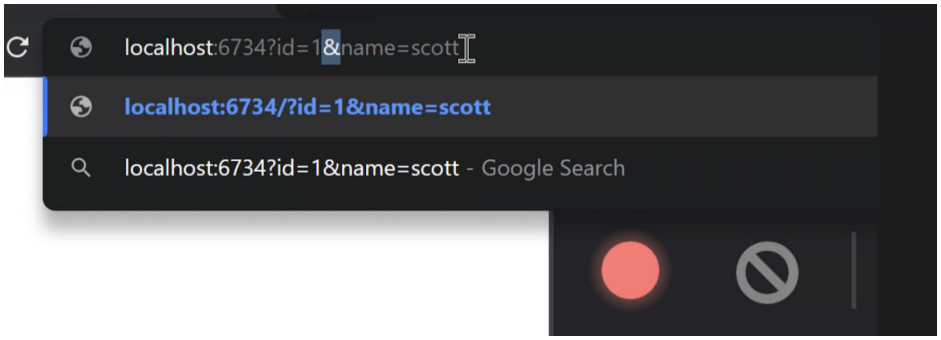
An HTTP request is a message sent from the browser to the server. For example, if I were the browser, I might ask the server, "Hey server, please give me a book" or "Please give me a page." These are examples of general requests.

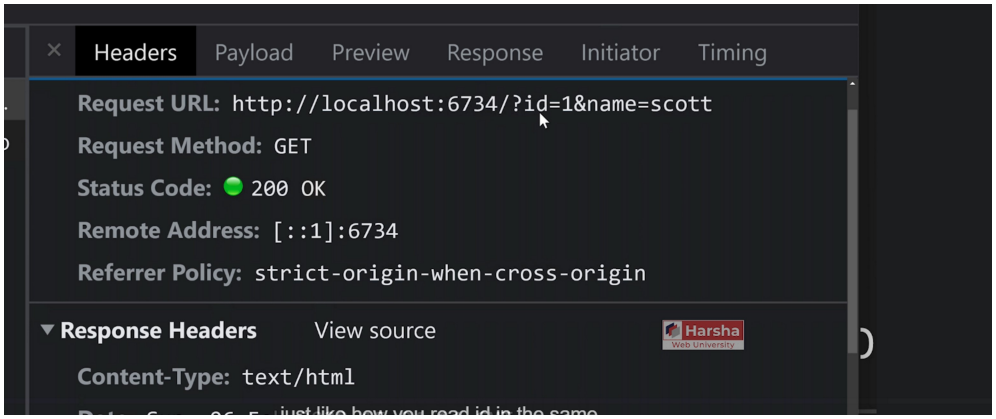


Asp.Net Core

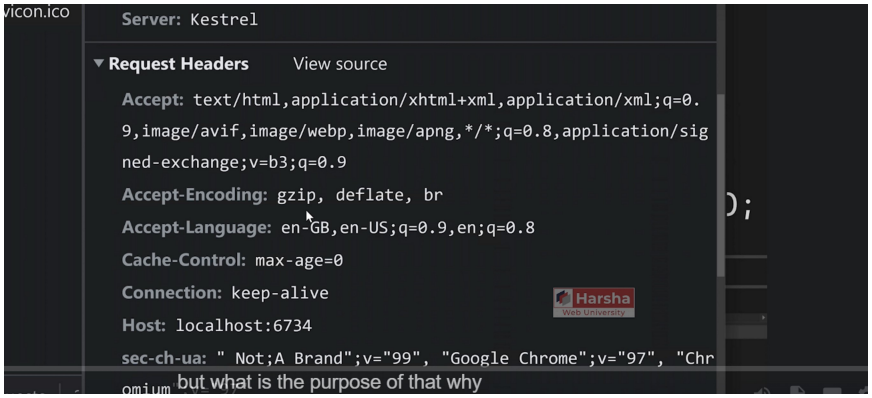
Query String



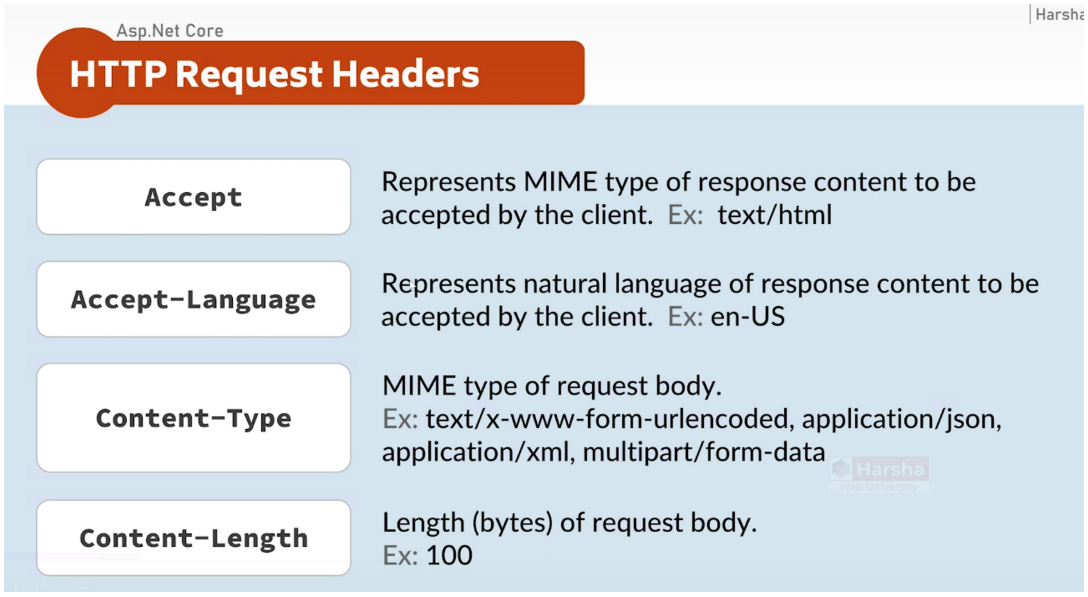


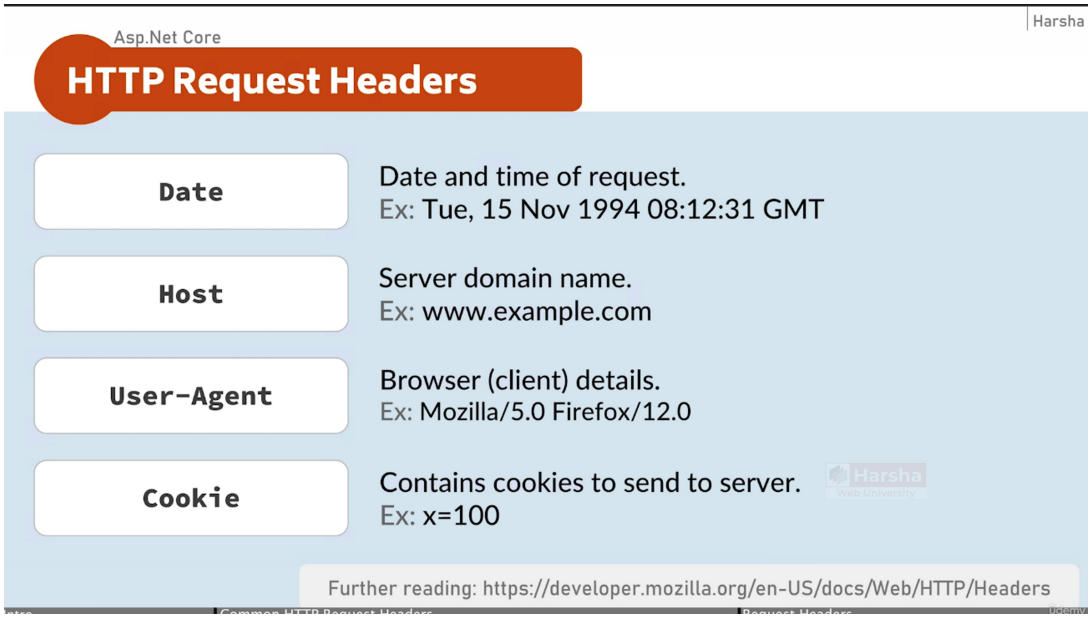


Request Headers are the key value pairs that are sent from the browser to server.

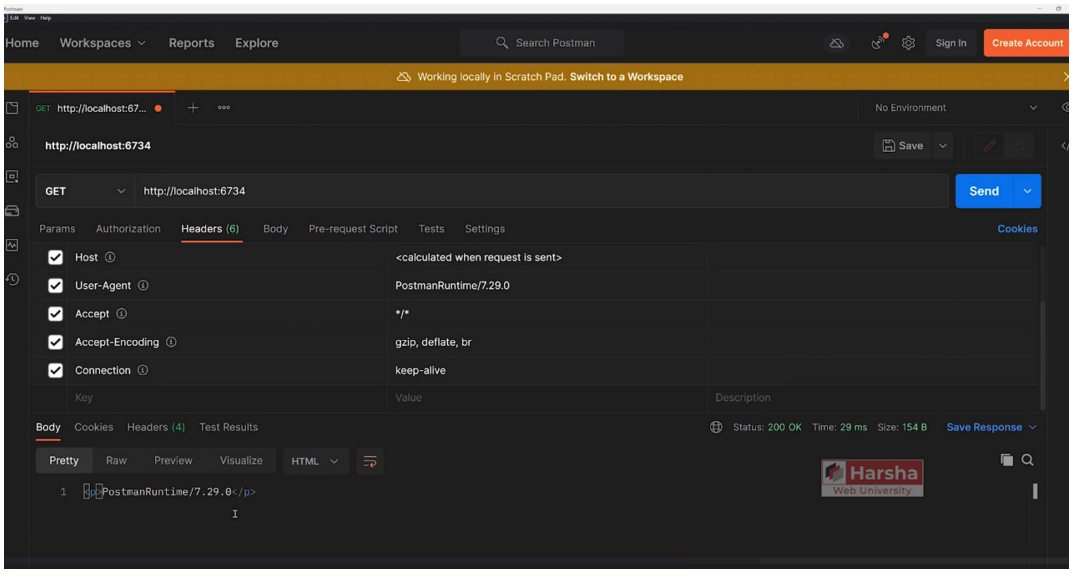


It is a way how the browser talk to the server.




Harsha
Asp.Net Core
HTTP Request Headers
Date
Date and time of request.
Ex: Tue, 15 Nov 1994 08:12:31 GMT
Host
Server domain name.
Ex: www.example.com
User-Agent
Browser (client) details.
Ex: Mozilla/5.0 Firefox/12.0
Cookie
Contains cookies to send to server.
Ex: x=100
Harsha
Web University
Further reading: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

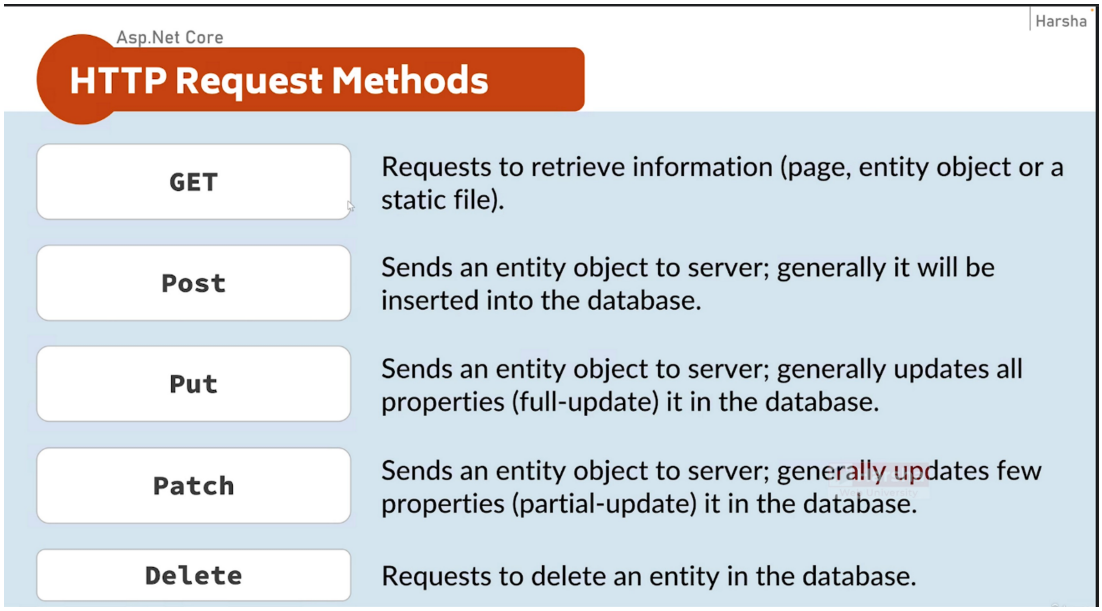

Asp.Net Core
Postman

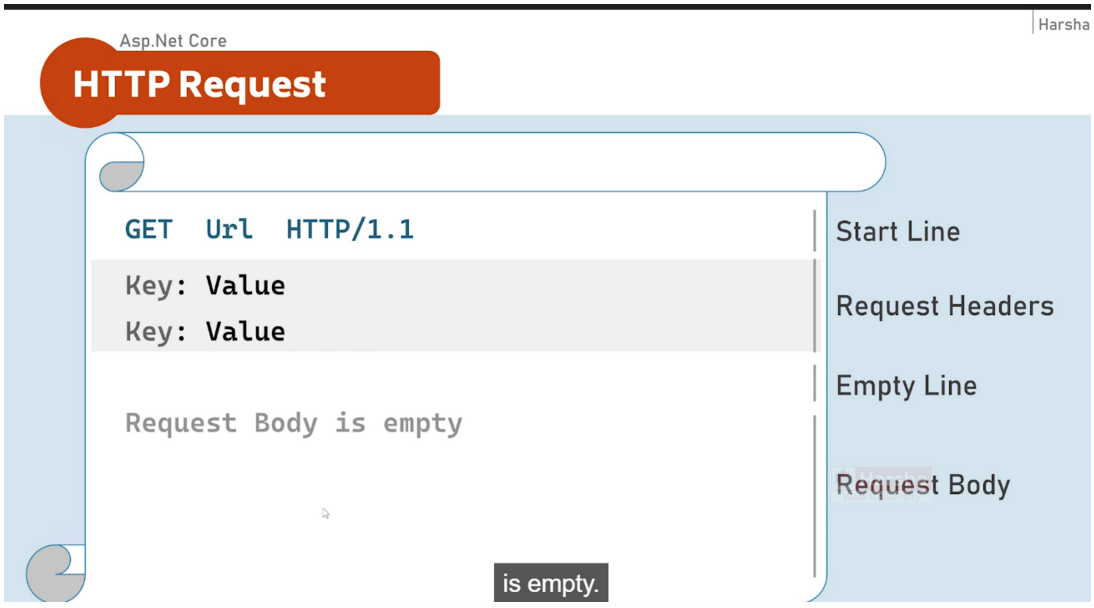

Home
Workspaces
Reports
Explore
Search Postman
Sign In
Create Account
Working locally in Scratch Pad. Switch to a Workspace
GET http://localhost:67...
No Environment
http://localhost:6734
Save
GET
http://localhost:6734
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Host
<calculated when request is sent>
User-Agent
PostmanRuntime/7.29.0
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 29 ms
Size: 154 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 PostmanRuntime/7.29.0
Harsha
Web University

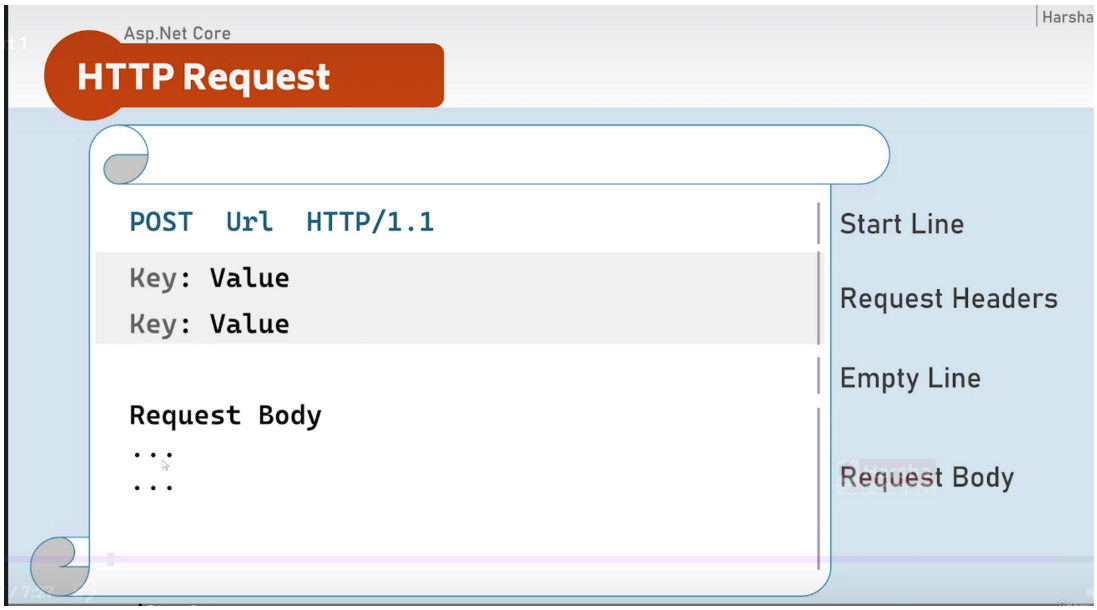
Asp.Net Core

HTTP Get [vs] Post

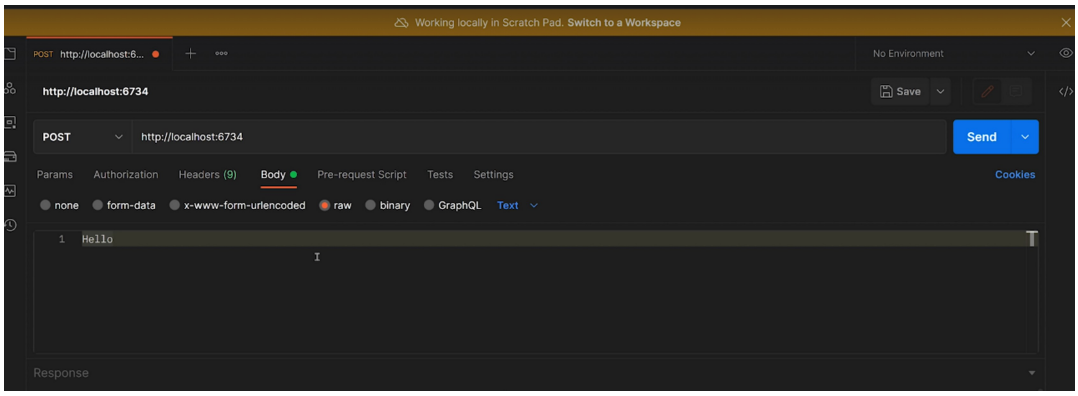
Part 1

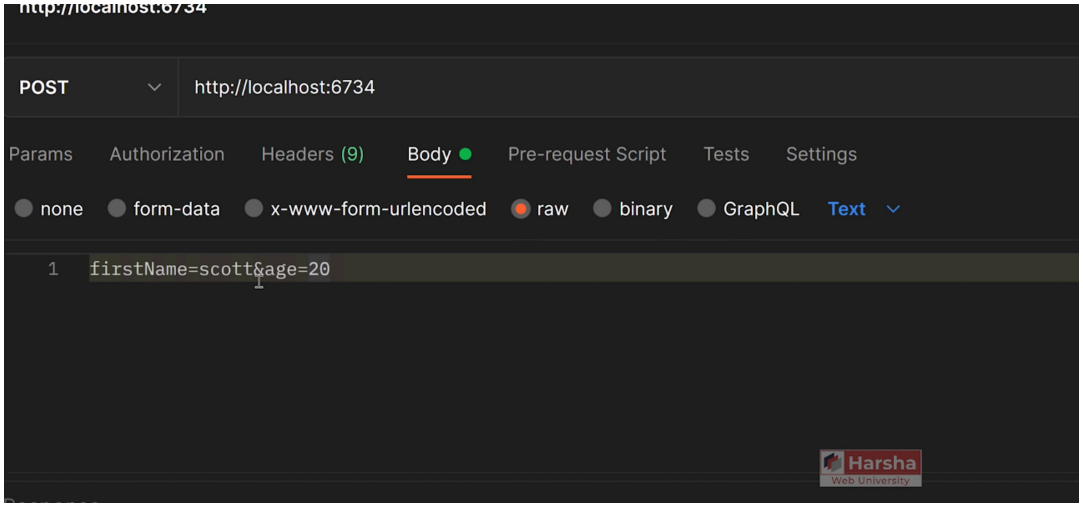


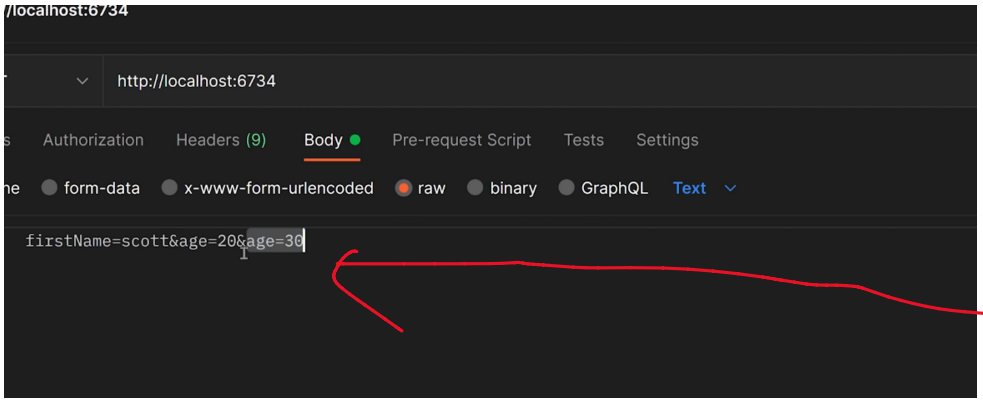





```
var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

//When a browser sends a request to the Kestrel and Kestrl forwards the same request to the application code, then asp dot net core automatically creates an object of type
app.Run(async (HttpContext context) =>
{
    System.IO.StreamReader reader = new StreamReader(context.Request.Body);
    string body = await reader.ReadToEndAsync();

    if(body == "Hello")
    {
        Console.WriteLine(
    }
});

app.Run();
```








```
using Microsoft.Extensions.Primitives;
using System.IO;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.Run(async (HttpContext context) =>
{
    StreamReader reader = new StreamReader(context.Request.Body);
    string body = await reader.ReadToEndAsync();

    Dictionary<string, StringValues> queryDict =
      Microsoft.AspNetCore.WebUtilities.QueryHelpers.ParseQuery
      (body);

});
```