

1. Introduction to HTTP

Asp.Net Core | Harsh

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.

HTTP-Hyper Text Transfer Protocol

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

Asp.Net Core | Harsh

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)

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)



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)

Introduction to HTTP

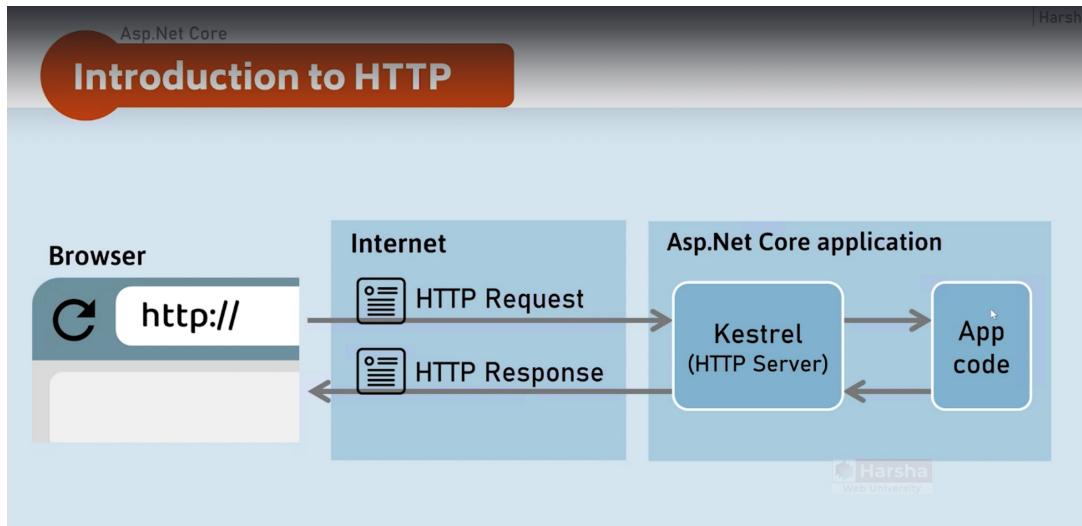
HTTP is an application-protocol that defines set of rules to send request from browser to server and send response from server to browser.

HTTPS

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



Introduction to HTTP



```

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

```

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).

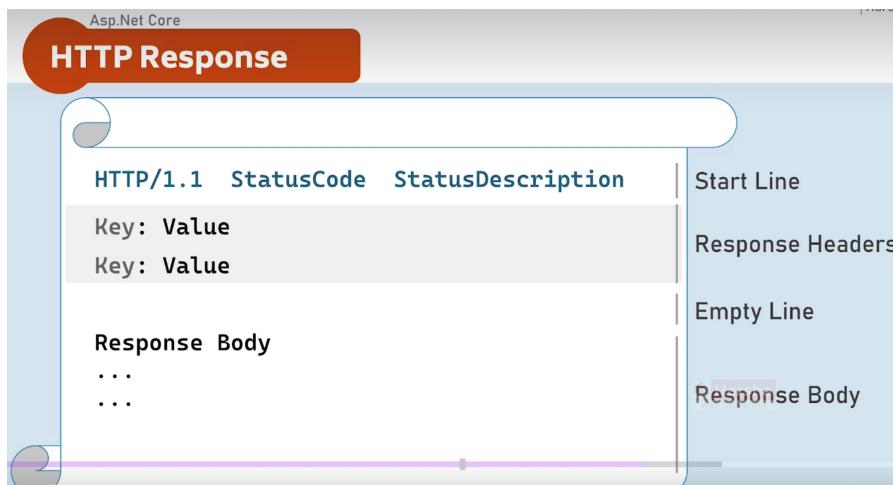
All websites on the internet use the HTTP protocol, with or without SSL (Secure Socket Layer). HTTP with SSL certificates is called HTTPS, which adds a security layer. HTTP was developed by Tim Berners-Lee in the 1990s and later standardized by the IETF for global use.

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
```

A screenshot of a code editor showing a C# file named "Program.cs". The code defines a simple ASP.NET Core application that maps the root URL to a "Hello World!" response and runs the application.

Asp.Net Core Harsha

HTTP Response

Response Start Line

Includes HTTP version, status code and status description.

HTTP Version:	1/1		2		3
Status Code:	101		200		302
	400		401		404
					500

Status Description:	Switching Protocols		OK		
	Found		Bad Request		
	Unauthorized		Not Found		
	Internal Server Error				

such as 101, 200, 400 etc.

3. HTTP Status Code

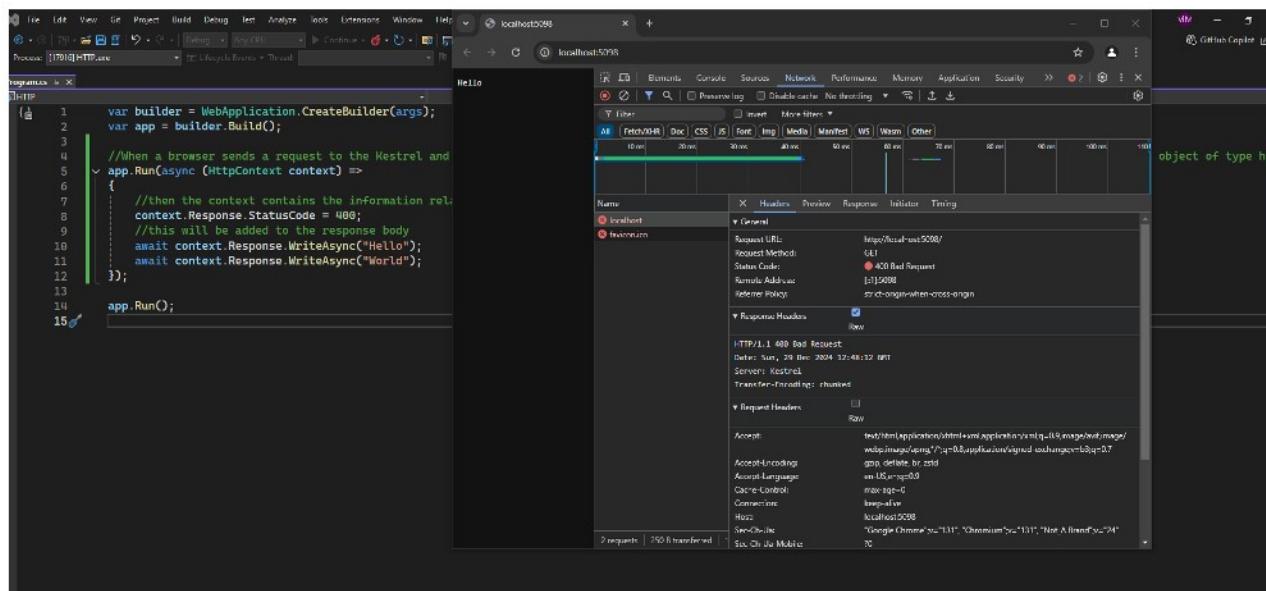
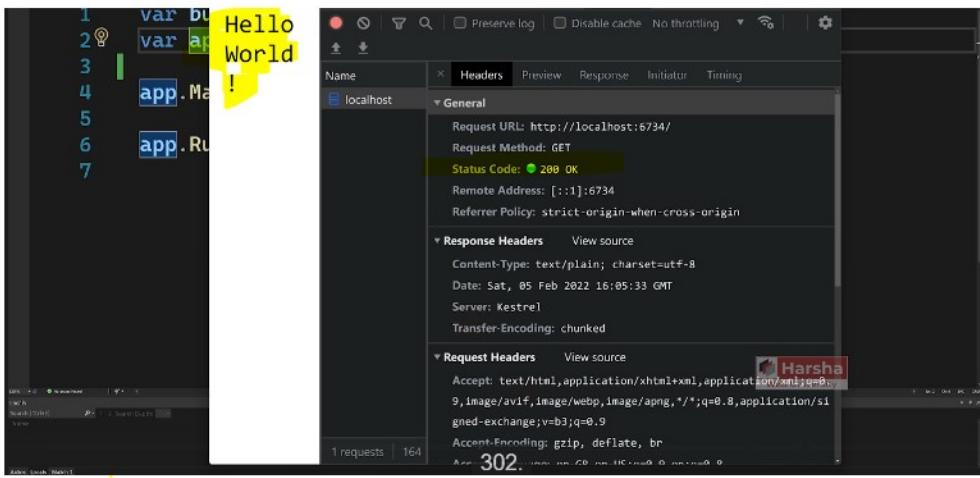
Asp.Net Core
HTTP Status Codes

Asp.Net Core Harsha

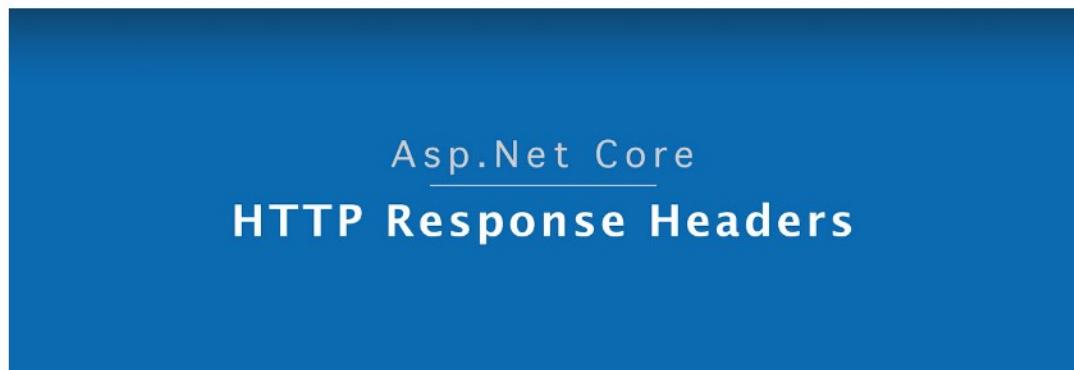
HTTP Response Status Codes

1xx Informational	101	Switching Protocols
2xx Success	200	OK
3xx Redirection	302	Found
	304	Not Modified
4xx Client error	400	Bad Request
	401	Unauthorized
	404	Not Found
5xx Server error	500	Internal Server Error

The screenshot shows a browser's developer tools Network tab. A request to 'localhost' is listed with a status code of '200 OK'. The response body contains the text 'Hello World!'. The browser's code editor on the left shows some JavaScript code starting with 'var bu'.

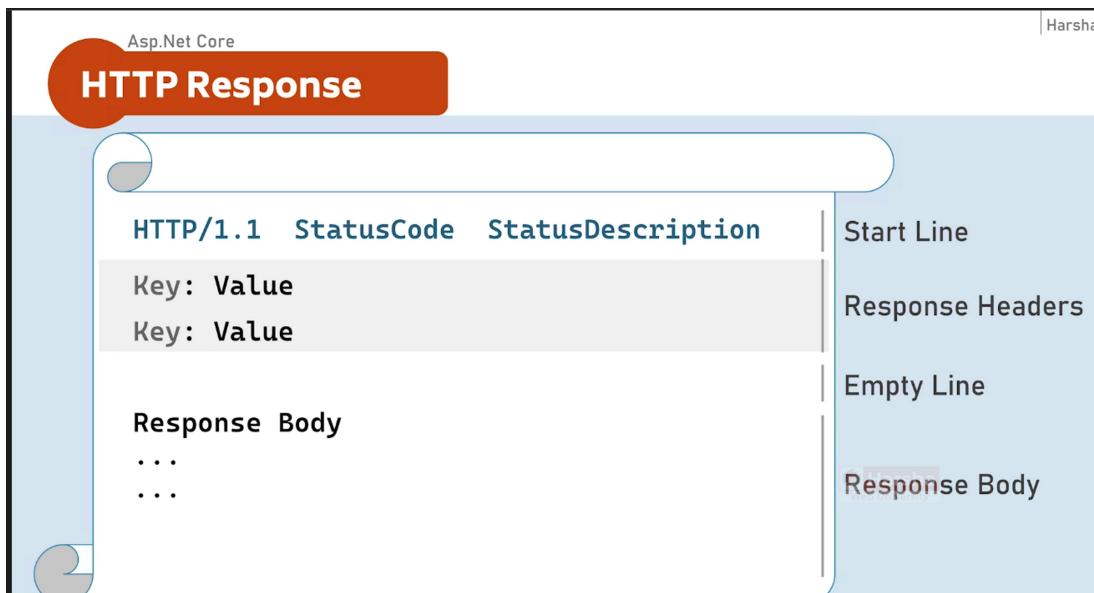


HTTP Response Headers

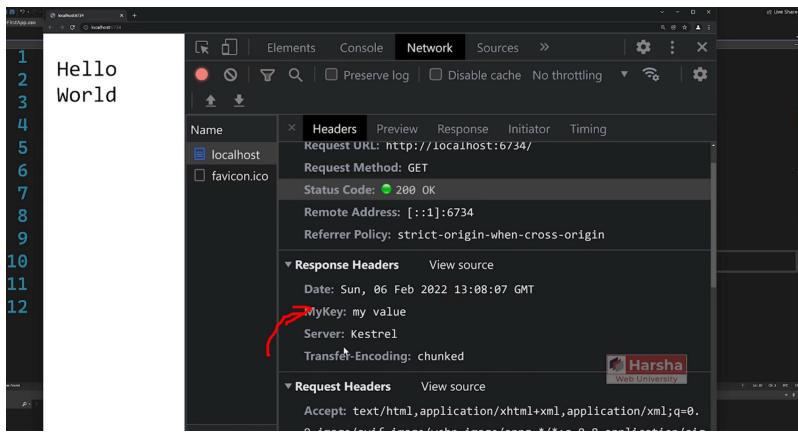


HTTP response headers are key-value pairs sent by the server to the client. They provide information about the response, such as its type and how it should be stored and managed on the client. These headers are typically not visible to the end user but are intended for information exchange between the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



```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help | Search -> HTTP
logams.cs | x
HTTP
1 var builder = WebApplication.CreateBuilder(args);
2 var app = builder.Build();
3
4 //When a browser sends a request to the Kestrel and Kestrel forwards the same request to the application code, then asp dot net core automatically creates a
5 app.Run(async (HttpContext context) =>
6 {
7     context.Response.Headers["MyKey"] = "my value"; //Response.Headers is a Dictionary in C#
8     //this will be added to the response body
9     await context.Response.WriteAsync("Hello");
10    await context.Response.WriteAsync("World");
11 });
12
13 app.Run();
14
```



HTTP Response Headers

Date

Date and time of the response. Ex: Tue, 15 Nov 1994
08:12:31 GMT

Server

Name of the server.
Ex: Server=Kestrel

Content-Type

MIME type of response body.
Ex: text/plain

Content-Length

Length (bytes) of response body.
Ex: 100



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

HTTP Response Headers

Cache-Control

Indicates number of seconds that the response can be
cached at the browser. Ex: max-age=60

Set-Cookie

Contains cookies to send to browser.
Ex: x=10

Access-Control-Allow-Origin

Used to enable CORS (Cross-Origin-Resource-Sharing)
Ex: Access-Control-Allow-Origin: http://www.example.com

Location

Contains url to redirect.
Ex: http://www.example-redirect.com



Further reading: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

HTTP Request

Asp.Net Core

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.

A screenshot of a browser's developer tools Network tab. The request for "hello" is selected, showing its headers:

- MyKey: my value**
- Server: My server**
- Transfer-Encoding: chunked**

Below the headers, the Request Headers section is expanded, showing the raw HTTP request:

```
GET /hello HTTP/1.1
Host: localhost:6734
Connection: keep-alive
sec-ch-ua: " Not;A Brand";v="99" "Google Cl
hromium";v="97"
sec-ch-ua-mobile: ?0
```

A red arrow points from the left margin towards the "Request Headers" section.

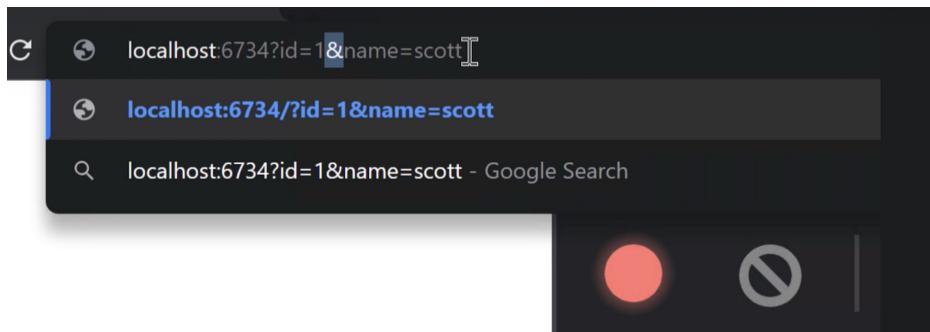
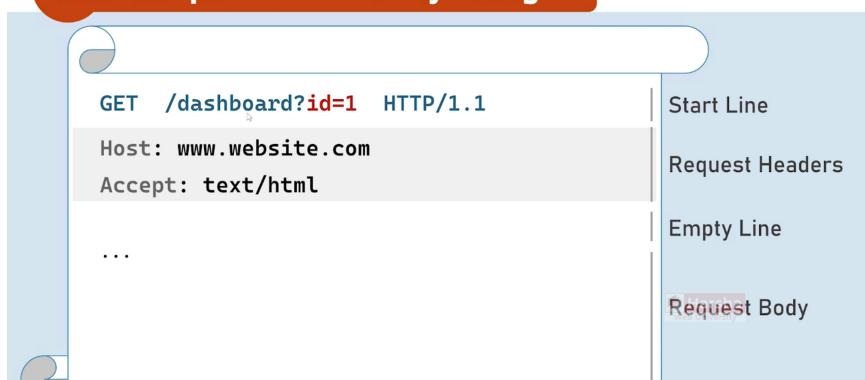
Asp.Net Core

Query String

Asp.Net Core

Harsha

HTTP Request - with Query String



A screenshot of the Network tab in a developer tools browser extension, likely Fiddler or Chrome DevTools. The request details are as follows:

- Request URL:** `http://localhost:6734/?id=1&name=scott`
- Request Method:** GET
- Status Code:** 200 OK
- Remote Address:** [::1]:6734
- Referrer Policy:** strict-origin-when-cross-origin

The Response Headers section shows:

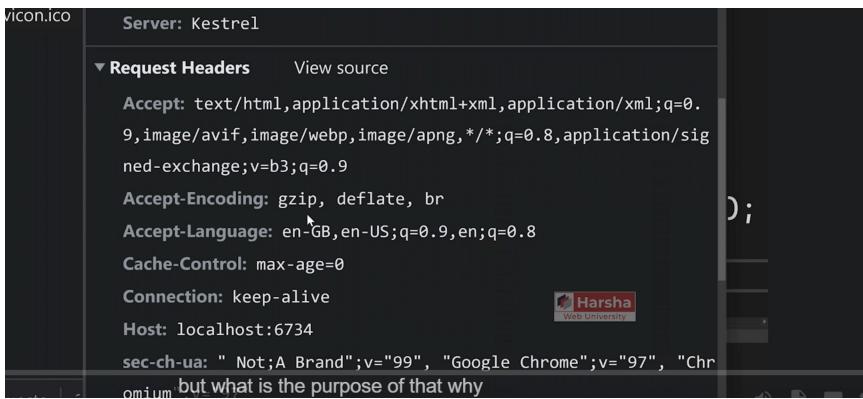
- Content-Type:** text/html

A note at the bottom states: "Data: 6 - 26.5 just like how you read id in the same".

Asp.Net Core

HTTP Request Headers

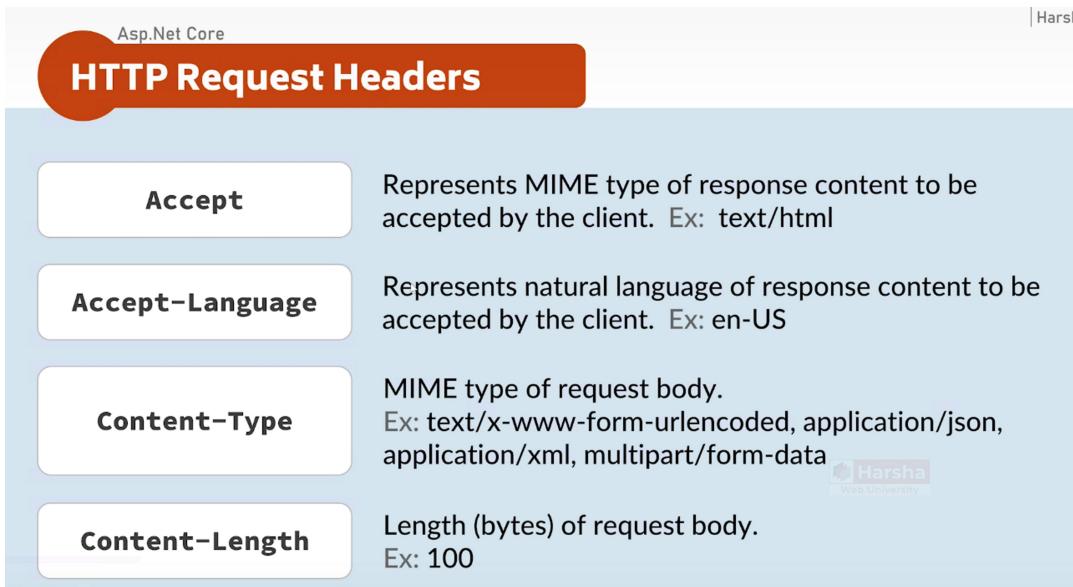
Request Headers are the key value pairs that are sent from the browser to server.



The screenshot shows the 'Request Headers' section of a browser's developer tools. The headers listed are:

- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
- Accept-Encoding: gzip, deflate, br
- Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
- Cache-Control: max-age=0
- Connection: keep-alive
- Host: localhost:6734
- sec-ch-ua: " Not A Brand";v="99", "Google Chrome";v="97", "Chromium" but what is the purpose of that why

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



HTTP Request Headers

- Accept** Represents MIME type of response content to be accepted by the client. Ex: text/html
- Accept-Language** Represents natural language of response content to be accepted by the client. Ex: en-US
- Content-Type** MIME type of request body.
Ex: text/x-www-form-urlencoded, application/json, application/xml, multipart/form-data
- Content-Length** Length (bytes) of request body.
Ex: 100

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

Further reading: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

Asp.Net Core Postman

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Edit', 'View', 'Help', 'Home', 'Workspaces', 'Reports', 'Explore', a search bar, and 'Sign In' and 'Create Account' buttons. Below the navigation is a banner stating 'Working locally in Scratch Pad. Switch to a Workspace'. The main area shows a request card for a 'GET' request to 'http://localhost:6734'. The 'Headers' tab is selected, showing the following configuration:

Key	Description	Value
Host	<calculated when request is sent>	PostmanRuntime/7.29.0
User-Agent	<calculated when request is sent>	PostmanRuntime/7.29.0
Accept	<calculated when request is sent>	/*
Accept-Encoding	<calculated when request is sent>	gzip, deflate, br
Connection	<calculated when request is sent>	keep-alive

At the bottom of the interface, there are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The status bar at the bottom right shows 'Status: 200 OK', 'Time: 29 ms', 'Size: 154 B', and a 'Save Response' button. The footer features the 'Harsha Web University' logo.

Asp.Net Core

HTTP Get [vs] Post

Part 1

Asp.Net Core

Harsha

HTTP Request Methods

GET

Requests to retrieve information (page, entity object or a static file).

Post

Sends an entity object to server; generally it will be inserted into the database.

Put

Sends an entity object to server; generally updates all properties (full-update) it in the database.

Patch

Sends an entity object to server; generally **updates** few properties (partial-update) it in the database.

Delete

Requests to delete an entity in the database.

Asp.Net Core

Harsha

HTTP Request

GET Url HTTP/1.1

Key: Value

Key: Value

Request Body is empty

Start Line

Request Headers

Empty Line

Request Body

is empty.

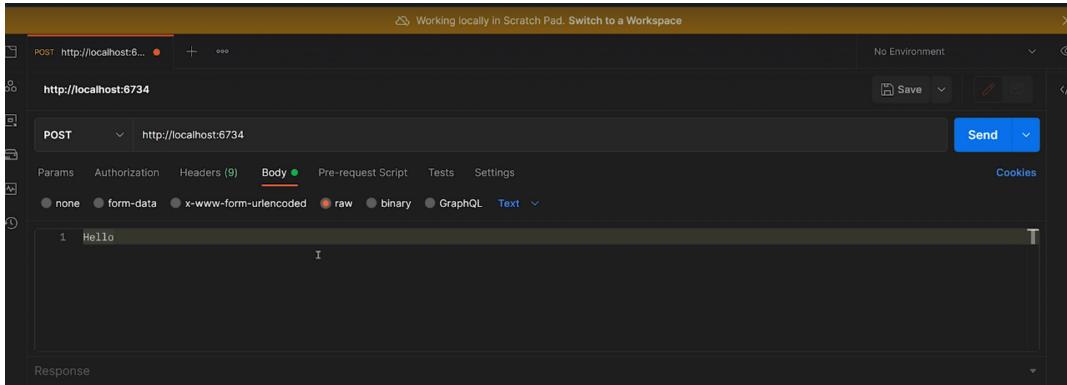


ram.cs

```

1 var builder = WebApplication.CreateBuilder(args);
2 var app = builder.Build();
3
4 //When a browser sends a request to the Kestrel and Kestrel forwards the same request to the application code, then asp dot net core automatically creates an object of type
5 app.Run(async (HttpContext context) =>
6 {
7     System.IO.StreamReader reader = new StreamReader(context.Request.Body);
8     string body = await reader.ReadToEndAsync();
9
10    if(body == "Hello")
11    {
12        Console.WriteLine(
13    });
14 });
15
16 app.Run();
17

```



http://localhost:6734

POST http://localhost:6734

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL Text

1 firstName=scott&age=20

Harsha
Web University

localhost:6734

http://localhost:6734

Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL Text

firstName=scott&age=20&age=30

Harsha
Web University

```
1 1 using Microsoft.Extensions.Primitives;
2 2 [using System.IO;
3 3
4 4 var builder = WebApplication.CreateBuilder(args);
5 5 var app = builder.Build();
6 6
7 7 app.Run(async (HttpContext context) =>
8 8 {
9 9     StreamReader reader = new StreamReader(context.Request.Body);
10 10    string body = await reader.ReadToEndAsync();
11 11
12 12    Dictionary<string, StringValues> queryDict =
13 13        Microsoft.AspNetCore.WebUtilities.QueryHelpers.ParseQuery
14 14        (body);
15 15});
```

Harsha
Web University