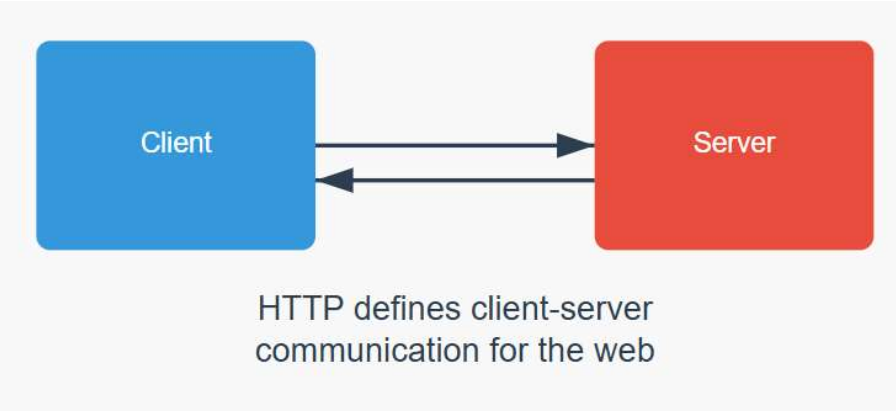
NMK30703 : Programming for Networks
HYPERTEXT TRANSFER PROTOCOL (HTTP)

Mohamed Elshaikh

Faculty of Electronics Engineering Technology – UniMAP (FTKEN-UniMAP)

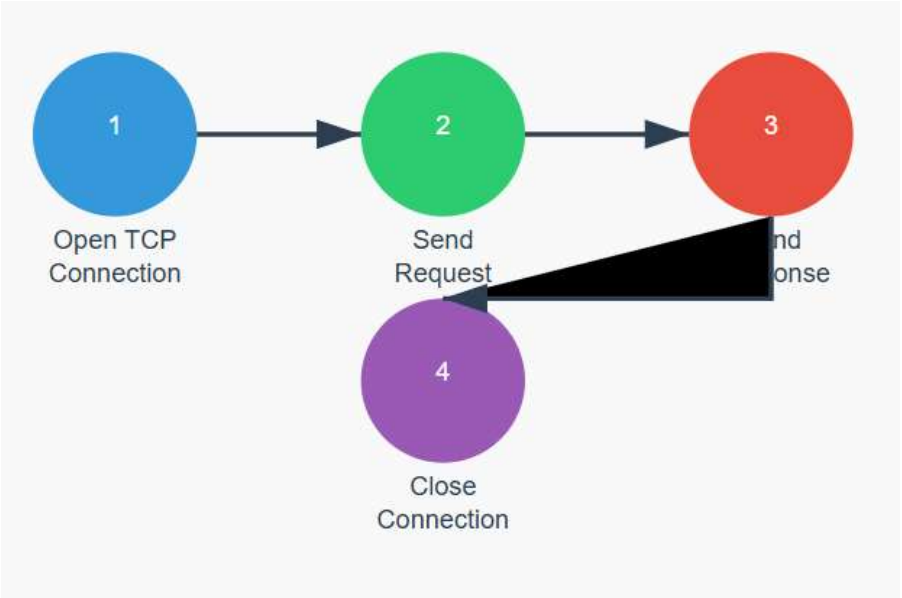
Introduction to HTTP

- **Content:**
  - HTTP defines client-server communication for the web
  - Transfers any data format (HTML, images, documents)
  - Deep understanding needed for programming



HTTP Protocol Basics

- Standard protocol for web browsers and servers
- Four step connection process:
  - Client opens TCP connection (port 80)
  - Client sends request
  - Server sends response
  - Server closes connection

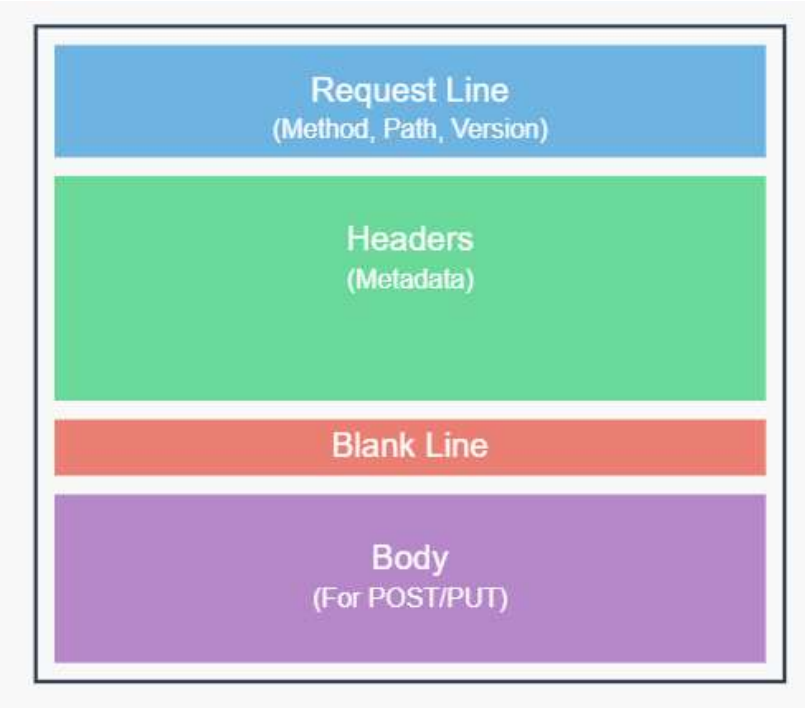


HTTP 1.0 vs 1.1

- HTTP 1.0: New connection per request
- HTTP 1.1: Persistent connections (Keep-Alive)
- Multiple requests/responses per connection

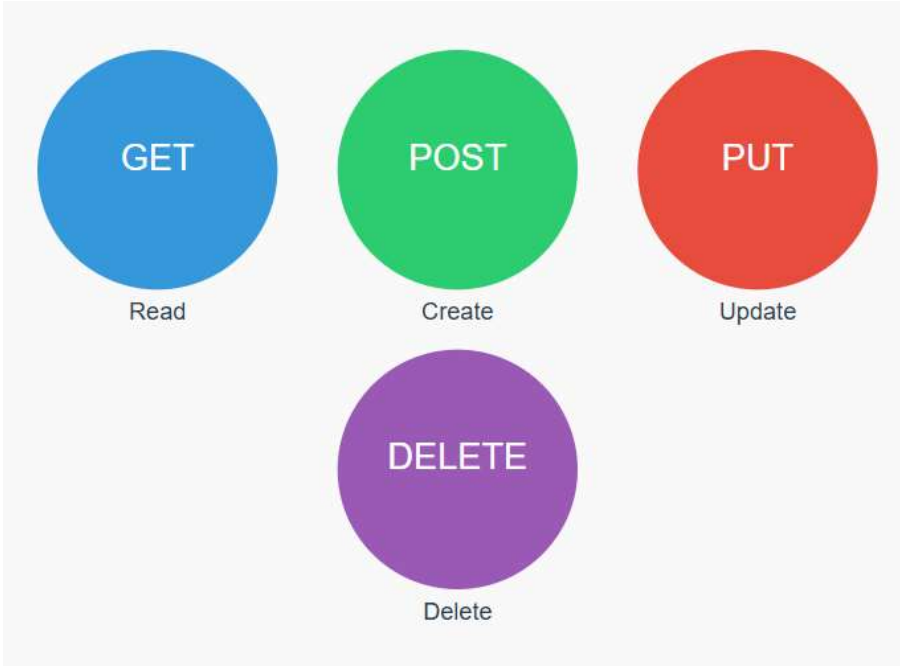
HTTP Request Structure

1. Request line (method, path, version)
2. Header fields (metadata)
3. Blank line
4. Body (for POST/PUT)



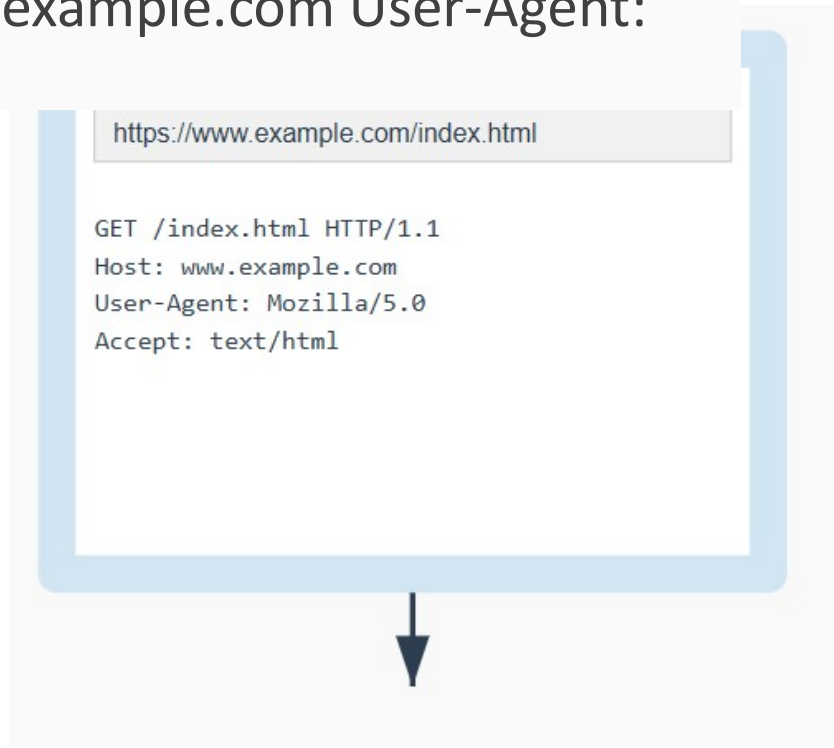
: HTTP Methods

- GET: Retrieve resource (safe, idempotent)
- POST: Submit data (not idempotent)
- PUT: Upload resource (idempotent)
- DELETE: Remove resource (idempotent)



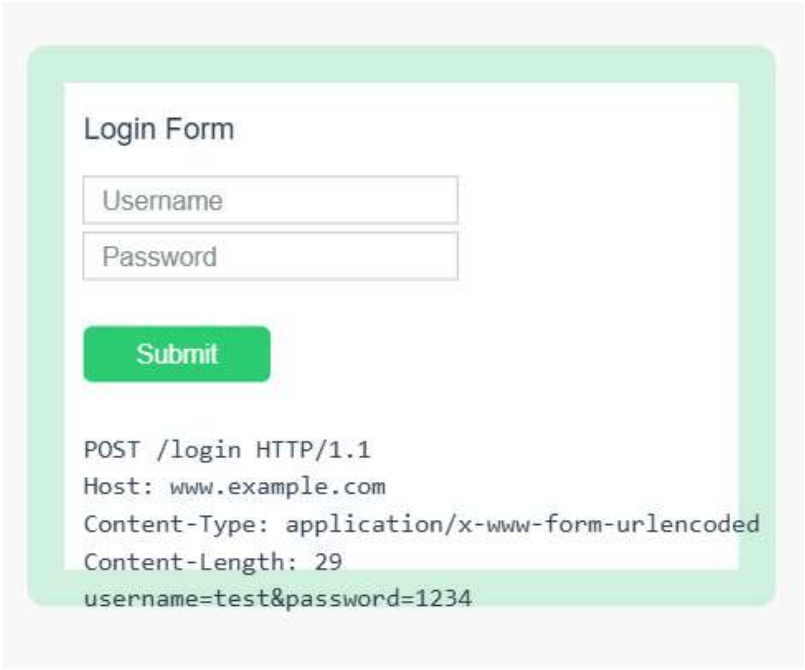
GET Request Example

- GET /index.html HTTP/1.1 Host: www.example.com User-Agent: Mozilla/5.0 Accept: text/html



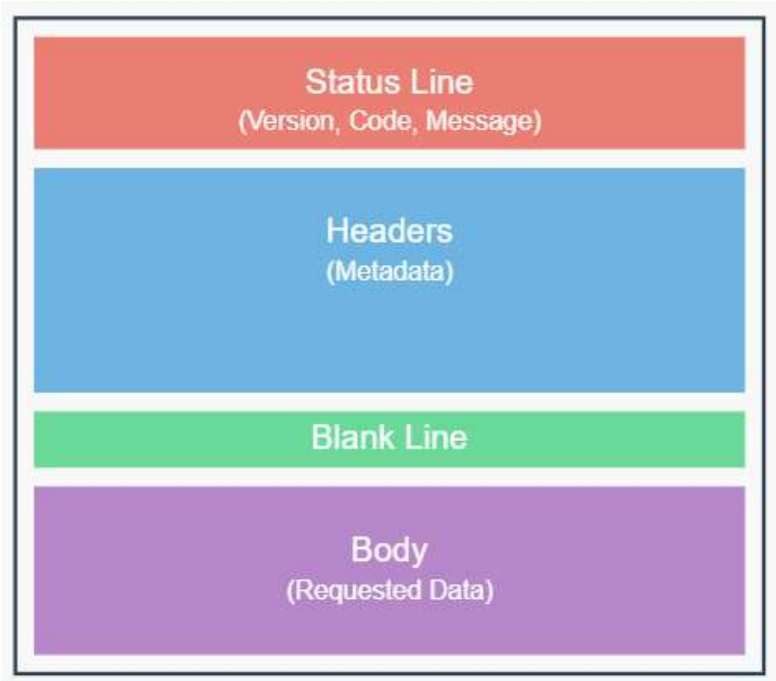
POST Request Example

- POST /login HTTP/1.1 Host: www.example.com Content-Type: application/x-www-form-urlencoded Content-Length: 29 username=test&password=1234



HTTP Response Structure

1. Status line (version, code, message)
2. Header fields
3. Blank line
4. Body



HTTP Status Codes

1xx: Informational

2xx: Success (200 OK)

3xx: Redirection

4xx: Client errors (404 Not Found)

5xx: Server errors

Common Status Codes

200 OK: Success

301 Moved Permanently

400 Bad Request

403 Forbidden

404 Not Found

500 Internal Server Error

Image: [SVG of common error pages]

Keep-Alive Connections

- Reduces connection overhead
- Enabled with Connection: Keep-Alive
- Java system properties control behavior

Java Keep-Alive Settings

- System.setProperty("http.keepAlive", "true");
- System.setProperty("http.maxConnections", "5");

HTTP Headers

- Key-value metadata
- Examples:
  - Content-Type: text/html
  - Content-Length: 348
  - Cache-Control: no-cache

MIME Types

- Format: type/subtype
- Examples:
  - text/html
  - image/jpeg
  - application/json

Cookies Introduction

- Small text strings for client state
- Sent in Set-Cookie header
- Returned in Cookie header

Cookie Attributes

- Domain: .example.com
- Path: /products
- Expires: Wed, 21 Oct 2025
- Secure: HTTPS only
- HttpOnly: No JavaScript access

Cookie Example

- Set-Cookie: sessionid=1234;
- Domain=.example.com;
- Path=/;
- Secure;
- HttpOnly;
- Expires=Wed, 21 Oct 2025]

Java Cookie Management

- CookieManager manager = new CookieManager();
- CookieHandler.setDefault(manager);

Cookie Policies

- ACCEPT_ALL
- ACCEPT_NONE
- ACCEPT_ORIGINAL_SERVER

Custom Cookie Policy

```
public class NoGovCookies implements CookiePolicy {
    public boolean shouldAccept(URI uri, HttpCookie c)
    {
    return !uri.getAuthority().endsWith(".gov");
    }
}
```

CookieStore Operations

- add() - Add new cookie
- get() - Retrieve cookies
- remove() - Delete cookie
- removeAll() - Clear all

HttpCookie Class

- Encapsulates cookie attributes
- Methods: setDomain(), setPath()
- Parsing: parse()

HTTP/2.0 Features

- Header compression
- Multiplexed connections
- Server push

Security Considerations

- Secure flag for sensitive cookies
- HttpOnly prevents XSS
- SameSite attribute

Best Practices

- Use GET for safe operations
- POST for actions with side effects
- Proper cookie scoping

Common Mistakes

- Overusing POST
- Not setting cookie expiration
- Ignoring status codes

Debugging HTTP

- Inspect headers
- Check status codes
- Verify cookies

Tools for HTTP

- Browser DevTools
- curl
- Postman
- Wireshark

Summary

- HTTP is stateless request-response
- Methods define operation types
- Headers carry metadata
- Cookies maintain state

Q&A