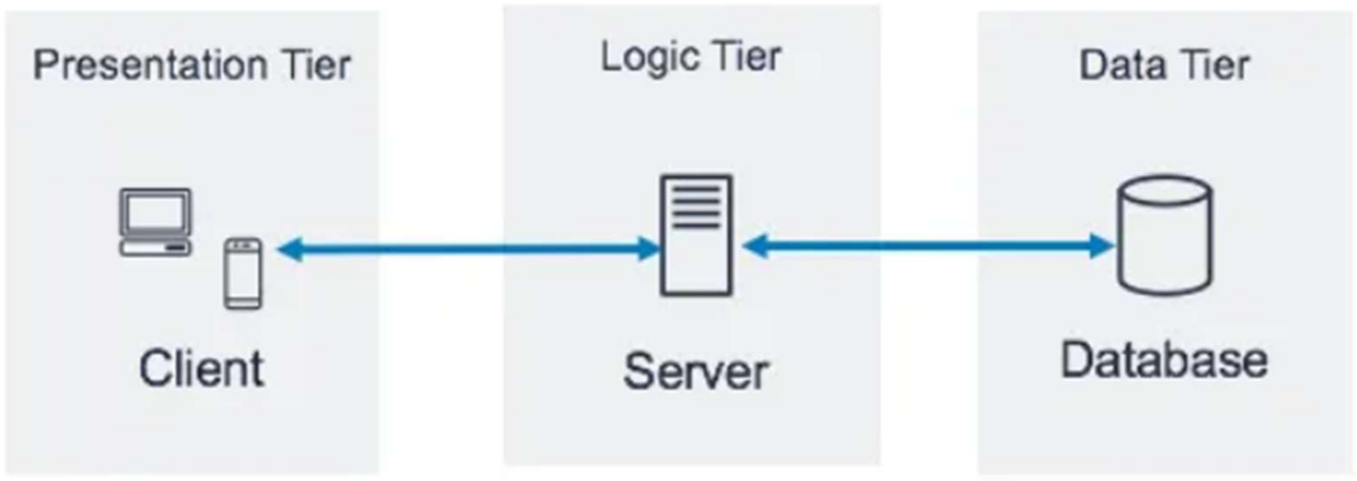
Web Fundamentals and HTTP Protocol

Web Development

Raed Felfel - 2025

Introduction to Web Applications

- What is a web application?
- How web applications differ from desktop applications
- Key characteristics:
    - Accessed through web browsers
    - Centralized hosting
    - No installation required
    - Cross-platform compatibility



- Three-tier architecture:
  - Client tier (Presentation layer)
  - Server tier (Application logic)
  - Database tier (Data storage)

Web Architecture

Evolution of Web Applications

1990s

Web 1.0 (1990s): Static HTML pages, limited interactivity

2000s

Web 2.0 (2000s): Dynamic content, AJAX, social media

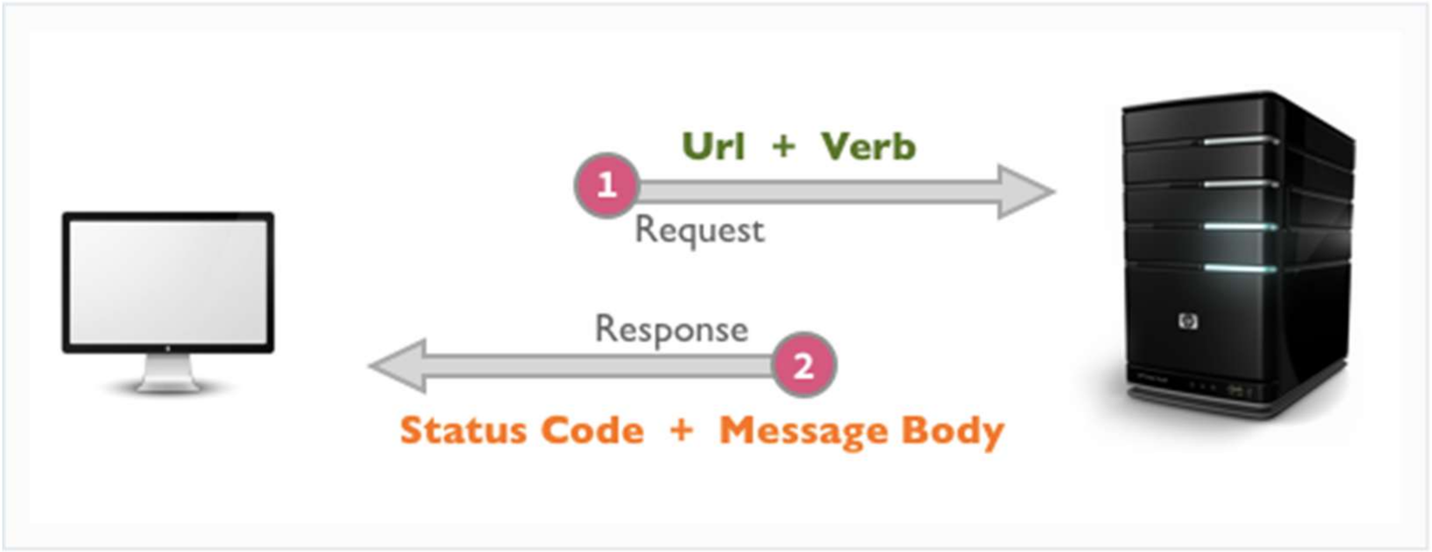
2010s-Present

Modern Web

- Responsive design
- Single Page Applications (SPAs)
- Progressive Web Apps (PWAs)
- Microservices architecture

HTTP Protocol - Overview

- HTTP = Hypertext Transfer Protocol
- Foundation of data communication on the web
- Stateless protocol - each request/response is independent
- Client initiates communication
- Based on request-response model



- Client initiates request to server
- Server processes request
- Server returns response
- Client renders or processes response

HTTP Request-Response Cycle

HTTP Methods

Method	Purpose	Example
GET	Request data	Retrieving a webpage
POST	Submit data	Submitting a form
PUT	Update existing resource	Updating user profile
DELETE	Remove a resource	Deleting an account
PATCH	Partial update	Changing one field
HEAD	Get headers only	Checking if resource exists
OPTIONS	Available communications options	CORS preflight

HTTP Status Codes

Range	Meaning and example
1xx - Informational	100 means Continue
2xx - Success	200 OK 201 Created 204 No Content
3xx - Redirection	301 Moved Permanently 302 Found (Temporary Redirect)
4xx - Client Error	400 Bad Request 401 Unauthorized 403 Forbidden 404 Not Found
5xx - Server Error	500 Internal Server Error 503 Service Unavailable

HTTP Headers and Content Types

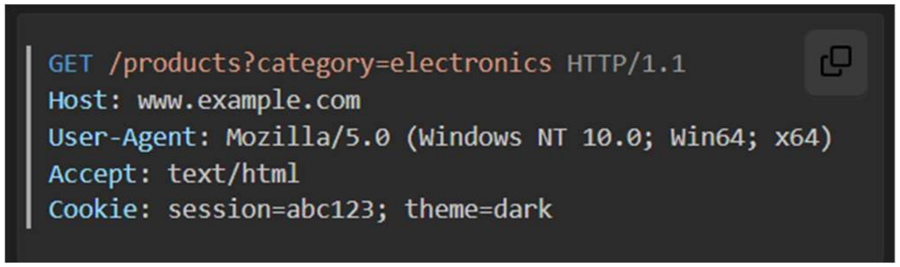
Common Request Headers:

- **User-Agent**: Browser/client information
- **Accept**: Content types client can process
- **Content-Type**: Type of data being sent
- **Authorization**: Authentication credentials
- **Cookie**: Stored browser cookies

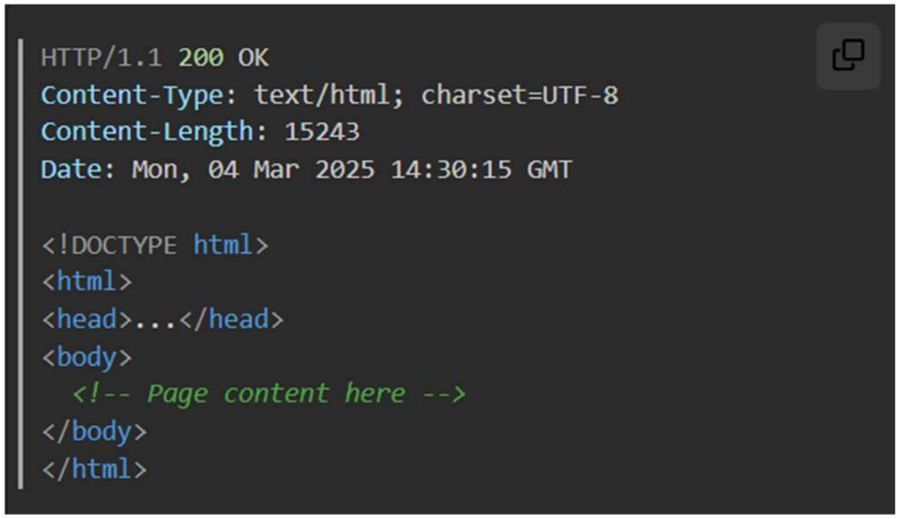
Common Response Headers:

- **Content-Type**: Format of response data
- **Content-Length**: Size of response
- **Set-Cookie**: Send cookies to client
- **Cache-Control**: Caching instructions

HTTP Example - Request and Response



```
GET /products?category=electronics HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html
Cookie: session=abc123; theme=dark
```



```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 15243
Date: Mon, 04 Mar 2025 14:30:15 GMT

<!DOCTYPE html>
<html>
<head>...</head>
<body>
  <!-- Page content here -->
</body>
</html>
```

Statelessness and State Management

HTTP is stateless - each request is independent

Challenges for web applications:

- User authentication
- Shopping carts
- Multi-step processes
- User preferences

Common state management techniques:

- Cookies (client-side data)
- Session state (server-side storage)
- Hidden form fields
- Query strings
- Local/Session storage (HTML5)
- Token-based authentication

Security Considerations in HTTP

HTTP vs HTTPS (secure HTTP)

- HTTP: unencrypted, vulnerable to eavesdropping
- HTTPS: encrypted using TLS/SSL

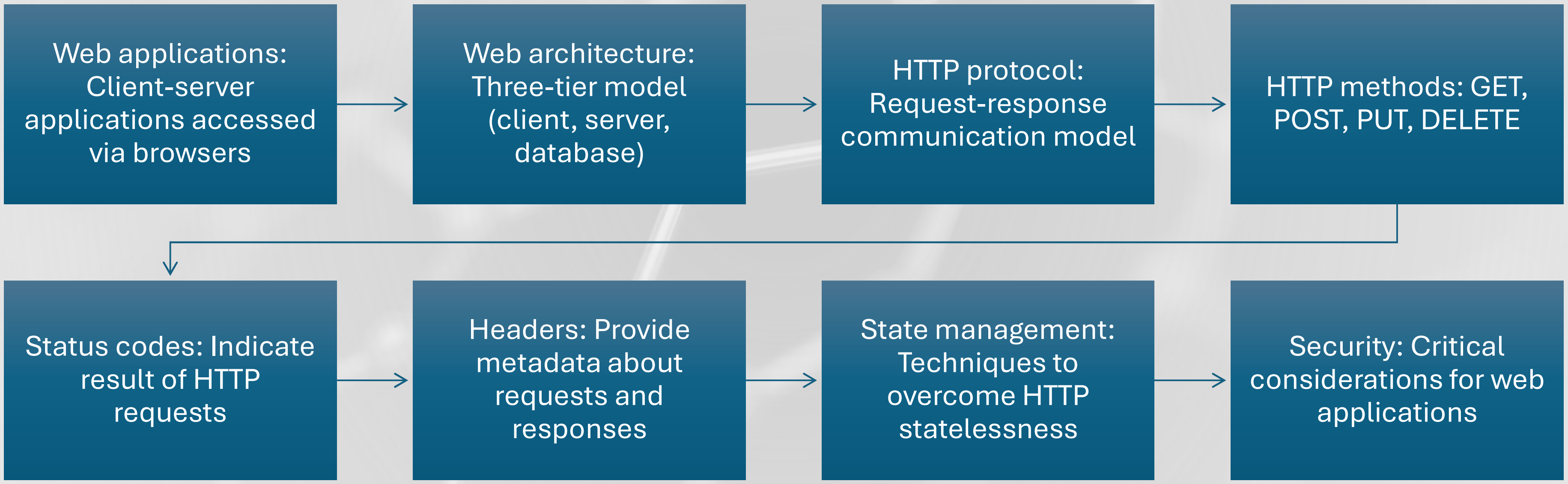
Common web security concerns:

- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- SQL Injection
- Man-in-the-Middle attacks
- Session hijacking

ASP.NET MVC security features:

- CSRF protection tokens
- Input validation and encoding
- Authentication and authorization frameworks

Summary


Web applications: Client-server applications accessed via browsers
Web architecture: Three-tier model (client, server, database)
HTTP protocol: Request-response communication model
HTTP methods: GET, POST, PUT, DELETE
Status codes: Indicate result of HTTP requests
Headers: Provide metadata about requests and responses
State management: Techniques to overcome HTTP statelessness
Security: Critical considerations for web applications