Software Development

Presented By:
Khaled Gamal

ASP.NET Core

Introduction

Outlines

- **Web Application Basic Concepts:**
  - **What is the meaning of web application**
  - **What are the components of web application**
  - **What is meaning of URL**
  - **Brief introduction of HTTP and Domain Name**
  - **What are static web pages and how they are processed**
  - **What are dynamic web pages and how they are processed**
- **What is ASP.NET Core?**
- **The structure of ASP.NET Core applications**
- **What types of applications can you build?**
- **Installing basic Environment and running Hello World Example**

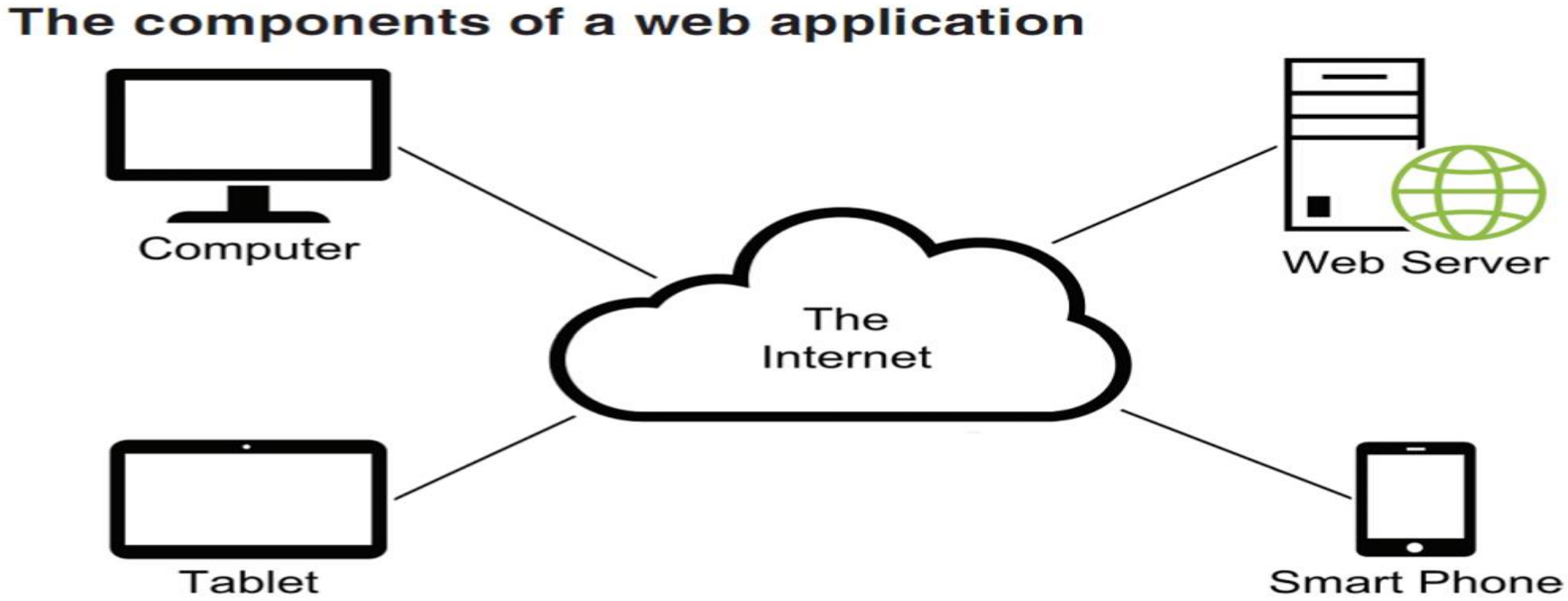
Web Application Basic Concepts

Web Application Basic Concepts

- **What is a web application?**
- A *web application* consists of a set of *web pages* that are generated in response to user requests and follows client-server model.

Web Application Basic Concepts

- **What are the components of a web application?**

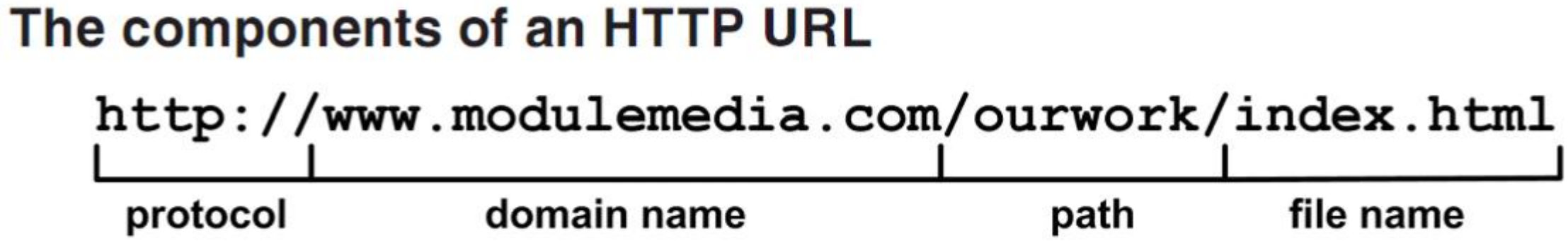


Web Application Basic Concepts

- **What are the components of a web application?**
  - *Clients*: are the computers, tablets, and mobile devices that use the web application by accessing or requesting its pages from web server through programs called *web browsers*. *(clients can also be a back end server)*
  - *A web server*: holds the files that make up the pages of a web application and responses with pages to browser's requests.
  - *A network*: is a system that allows clients and server to communicate, the *Internet* is a large network that consists of many smaller networks.

Web Application Basic Concepts

- To access a web page from a browser, you can type a *URL (uniform resource locator)* into browser's address area and press Enter.
- The URL consists of the *protocol* (usually, **HTTP**), domain name (or host name), folder or directory path and file name.
  - If you omit the protocol, HTTP is assumed.
  - If you omit the file name, the web server will look for a file named *index* or *default* *(with extension .html or .htm in static web pages)*.



Web Application Basic Concepts

- Domain name (or host name) uniquely identifies its location on the internet by mapping via the Domain Name Service (DNS) to an IP address.
  - Examples include microsoft.com, www.google.co.uk, and facebook.com.
- Hypertext Transfer Protocol (HTTP) is a stateless request-response protocol whereby a client machine sends a request to a server, which sends a response in turn.
- Every HTTP request consists of a ***verb*** *(GET,POST,PUT,PATCH,DELETE)* indicating the type of the request and a ***path*** indicating the resource to interact with.
- A request typically also includes ***headers***, which are key-value pairs, and in some cases a ***body***, such as the contents of a form, when sending data to the server.
- An HTTP response contains a ***status code***, indicating whether the request was successful, and optionally ***headers*** and a ***body***.

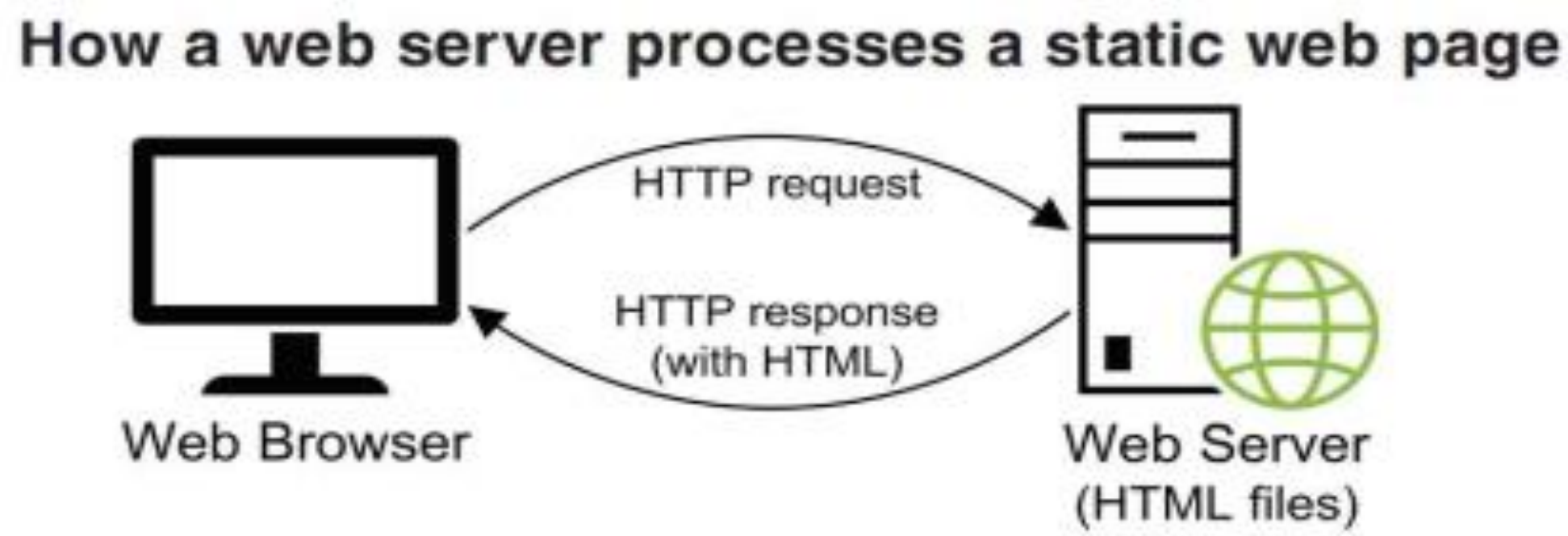
Static Web Pages

Static Web Pages

- **What is *static web* pages and how are they processed?**
- A *static web page*: is a web page built from or designed by an *HTML* document that is stored on the web server and doesn't change each time it is requested.
- The file names for static web pages usually have *.htm* or *.html* extensions.
- This type of web page is sent directly from web server to the web browser when the browser requests it.
- **Note**: *Not only HTML but may be used also CSS (or CSS frameworks like bootstrap, Twilind.css, and etc), JavaScript (or JavaScript libraries (Jquery, Ajax, and etc) or SPA Frameworks like Angular, React, vue.js or Blazor)).*

Static Web Pages

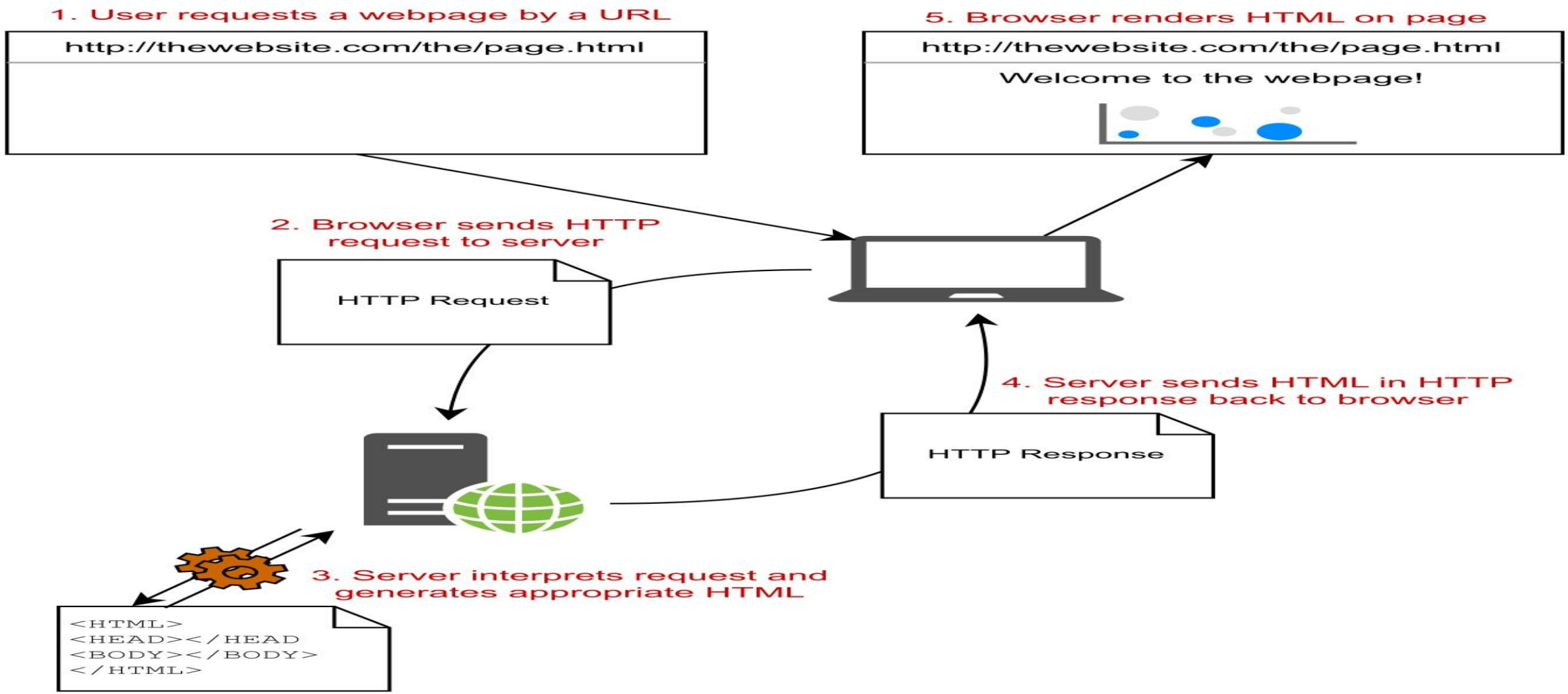
- **What is *static web* pages and how are they processed?**



Static Web Pages

- **What is *static web* pages and how are they processed?**
- When the user requests a static web page (through typing URL or clicking link), the browser sends an *HTTP request* to the web server that includes the name of the file that being requested.
- When the web server receives the request, it retrieves the HTML for the web page (from its file system) and sends it back to the browser as part of an *HTTP response*.
- When the browser receives the HTTP response, it *renders* the HTML into a web page that is displayed in the browser.

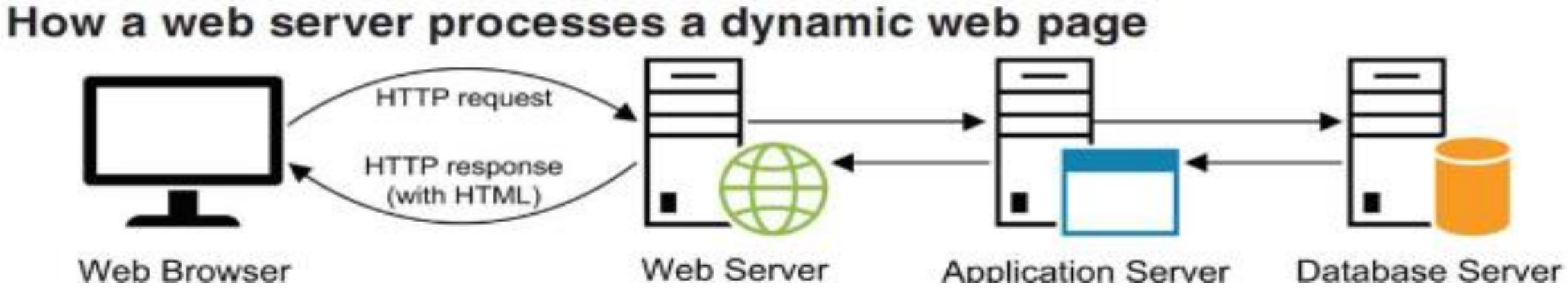
Static Web Pages



Dynamic Web Pages

Dynamic Web Pages

- **What is *dynamic web* pages and how are they processed?**
- A *dynamic web page*: is a web page that is created by a *program* *(handler)* on an *application server* running on a web server.
- This program uses the data that is sent with the HTTP request to generate the HTML that is returned to the browser.



Dynamic Web Pages

- **What is *dynamic web* pages and how are they processed?**
- The process begins when the user requests a page in a web browser through typing URL, clicking a link, or clicking a button that submits a form that contains the data that the dynamic page should process.
- In either case, the web browser builds an HTTP request, which includes whatever data the application needs for processing the request, and sends it to the web server.
- When a web server receives a request for a dynamic web page, it looks up the extension of the requested file to identify the *application server* and passes the request to that appropriate *application server* for processing.

Dynamic Web Pages

- **What is *dynamic web* pages and how are they processed?**
- Next, the application server retrieves the appropriate program (let call it *Handler*) and loads any form data that user submitted.
- Then, it executes the program that will generate the HTML for the web page.
- If necessary, the program will also request data from a database server and use that data as part of the web page it is generating.
- After program finishing, the application server sends the dynamically generated HTML back to the web server.

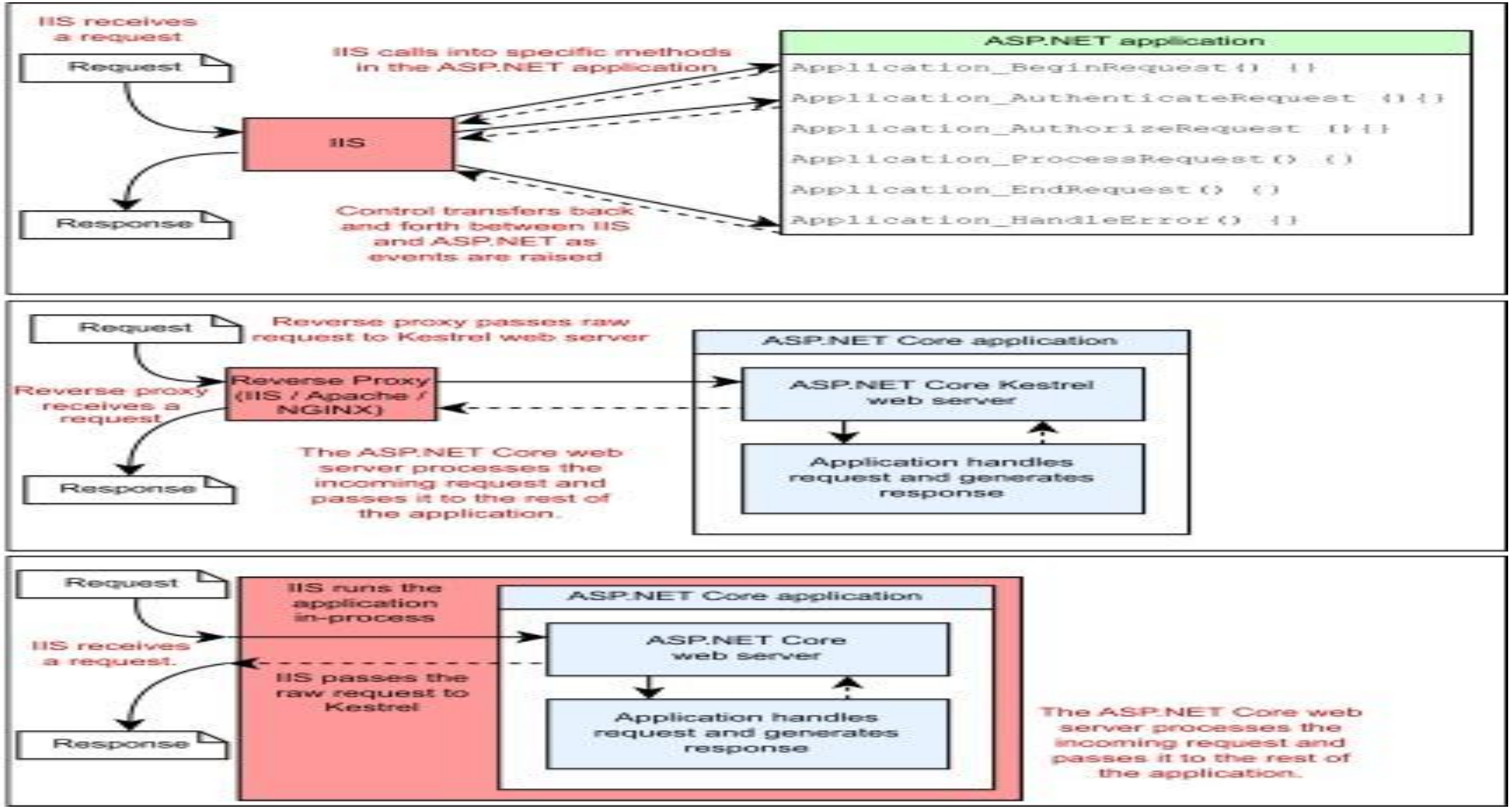
Dynamic Web Pages

- **What is *dynamic web* pages and how are they processed?**
- Then, the web server sends the HTML back to the browser in an HTTP response.
- When the web browser receives the HTTP response, it renders the HTML and displays the web page, with no knowledge whether the HTML was for a static or a dynamic page.

Dynamic Web Pages

- When you build ASP.NET core applications, the ***web server*** or ***web host*** is:
  - *Kestrel (already built-in, asp.net core application is self-hosted by it)*
  - *You can build your own*
- But you can host your application on several web servers or use them as ***reverse proxy*** like:
  - *Internet Information Services (IIS)*
  - *Nginx*
  - *Apache*
- ***A reverse proxy*** is software responsible for receiving requests and forwarding them to the appropriate web server.
- ***The reverse proxy*** is exposed directly to the internet, whereas the underlying web server is exposed only to the proxy.
- This setup has several benefits, primarily security and performance for the web servers.

Dynamic Web Pages



Dynamic Web Pages

- *ASP.NET core* is used for the application server *(Program, the code you will write with the help of ASP.NET different paradigms)*.
- You're also likely to use Microsoft's SQL Server for the DBMS (Database Management System).

What is ASP.NET Core?

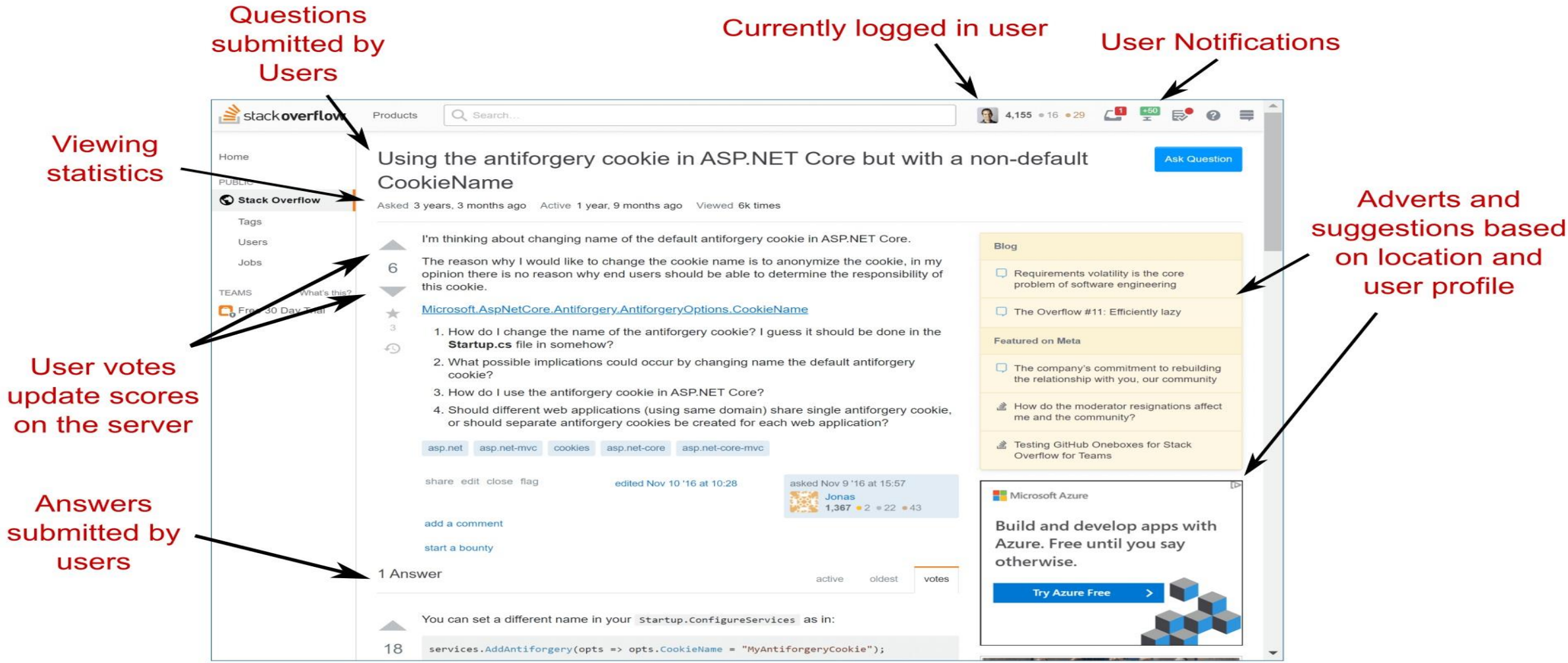
What is ASP.NET Core?

- ASP.NET Core is a cross-platform, open-source application framework that you can use to build dynamic web applications quickly.
- ASP.NET Core runs on .NET 8, which is the latest version of .NET Core—a high-performance, cross-platform, open-source runtime.
- ASP.NET Core provides structure, helper functions, and a framework for building applications, which saves you from having to write a lot of this code yourself, which makes writing web applications faster, easier, and more secure than trying to build everything from scratch.

What is ASP.NET Core?

- It contains libraries for common things like:
  - Creating dynamically changing web pages
  - Letting users log in to your web app
  - Letting users use their Facebook accounts to log in to your web app
  - Providing a common structure for building maintainable applications
  - Reading configuration files
  - Serving image files
  - Logging requests made to your web app
- Without a dynamic framework, it wouldn’t be possible to log in to websites or to display any sort of personalized data on a page.

What is ASP.NET Core?



What is ASP.NET Core?

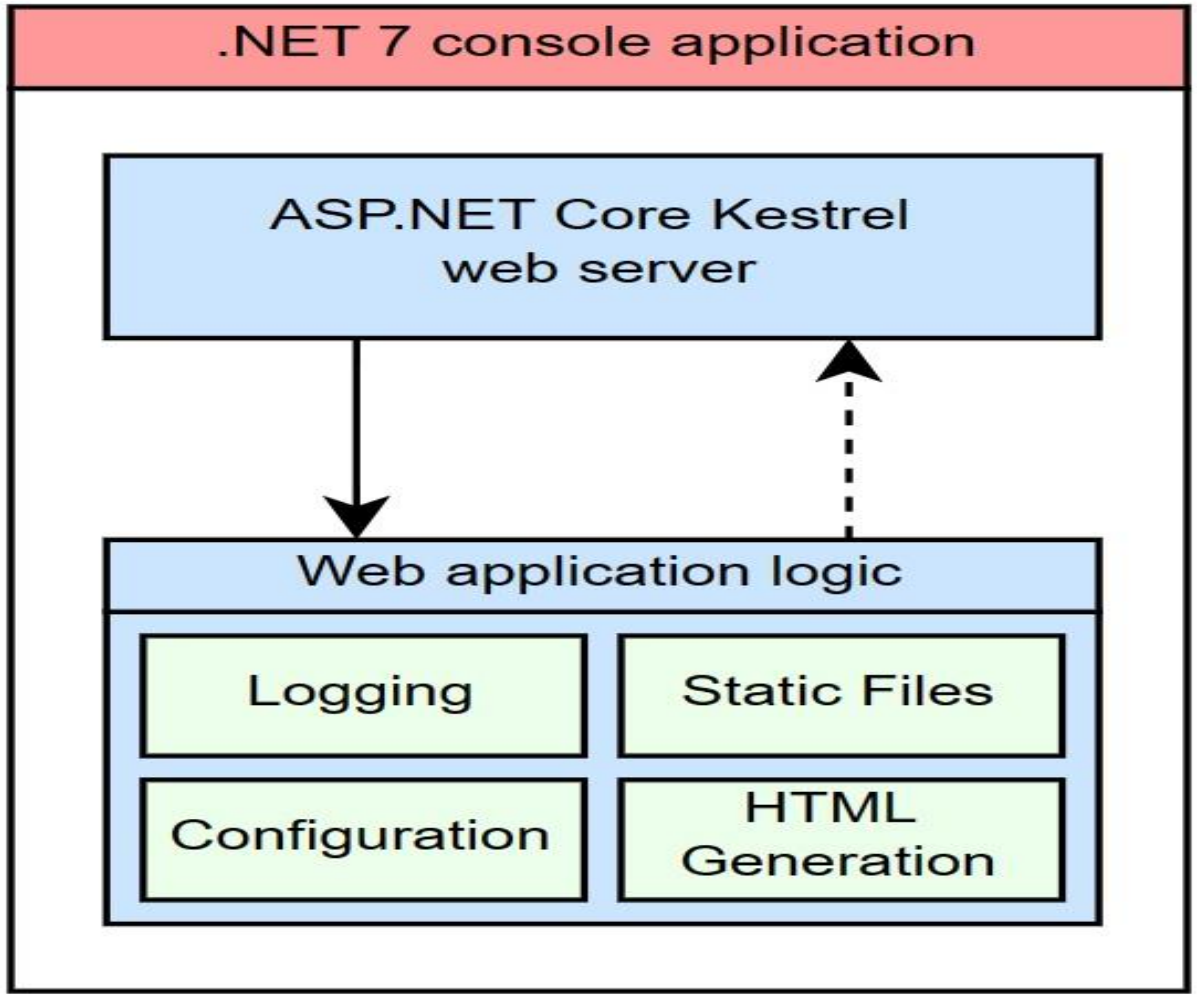
- With .NET, it’s possible to build console applications that run cross platform.
- Microsoft created ASP.NET Core to be an additional layer on top of console applications so that converting to a web application involves adding and composing libraries (Adding a web server library converts this model to an ASP.NET Core web app. You can add other features, such as configuration and logging, using various libraries).

What is ASP.NET Core?

You write a .NET 7 console app that starts up an instance of an ASP.NET Core web server.

Microsoft provides a cross-platform web server by default, called Kestrel.

Your web application logic is run by Kestrel. You'll use various libraries to enable features such as logging and HTML generation as required.

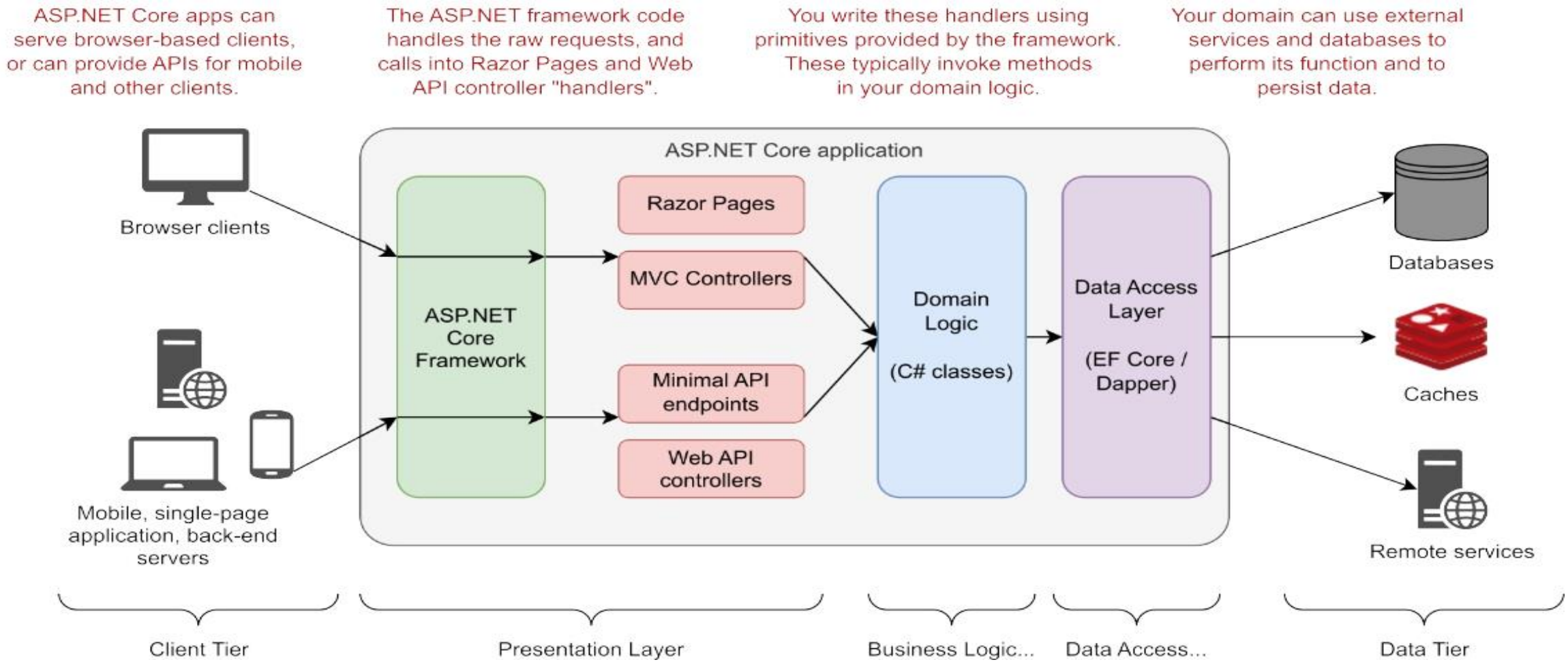


The structure of ASP.NET Core applications

The structure of ASP.NET Core applications

- The ASP.NET Core framework code handles requests from a client, dealing with the complex networking code.
- Then the framework calls in to ***handlers*** (Razor Pages and Web API controllers, for example) that you write using primitives provided by the framework.
- Finally, these handlers call in to your application's domain logic—typically, C# classes and objects that doesn't depend directly on ASP.NET Core
- You can interact with other services here, such as databases or remote APIs.

The structure of ASP.NET Core applications



What types of applications can you build?

What types of applications can you build?

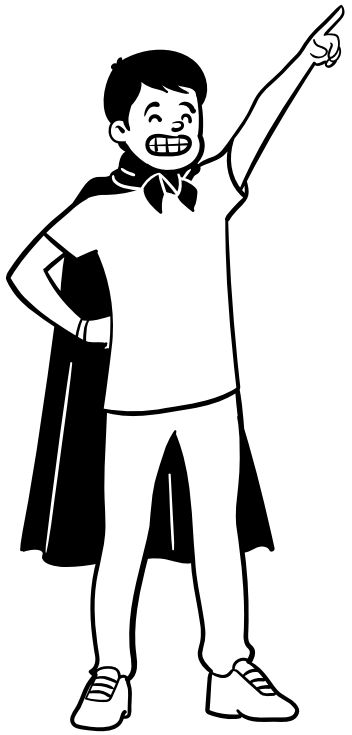
- ASP.NET Core includes APIs that support many paradigms:
  - ***Minimal APIs***: Simple *HTTP APIs* and *RESTful-APIs* that can be consumed by mobile applications or browser-based single-page applications.
  - ***Web APIs***: An alternative approach to building *HTTP APIs* and *RESTful-APIs* that adds more structure and features than minimal APIs.
  - *gRPC APIs*: Used to build efficient binary APIs for server-to-server communication using the gRPC protocol.
  - *Real-time web app (SignalR)*: using ASP.NET core signalR library to enable server-side code to push content to clients instantly.
  - ***Razor Pages***: Used to build page-based server-rendered applications.

NOTE: ***REST*** stands for *representational state transfer*.
RESTful applications typically use lightweight and stateless HTTP calls to read, post (create/update), and delete data.

What types of applications can you build?

- ASP.NET Core includes APIs that support many paradigms:
  - ***MVC controllers***: Similar to Razor Pages. Model-View-Controller (MVC) controller applications are for server-based applications but without the page-based paradigm.
  - *Blazor WebAssembly*: A browser-based single-page application framework that uses the WebAssembly standard, similar to JavaScript frameworks such as Angular, React, and Vue.
  - *Blazor Server*: Used to build stateful applications, rendered on the server, that send UI events and page updates over WebSockets to provide the feel of a client-side single-page application but with the ease of development of a server-rendered application.

NOTE: You don't need to choose only one of these paradigms; ASP.NET Core can combine multiple paradigms within a single application.



Any Questions?


THANK
YOU!

Installing basic Environment and running Hello World Example