23MX15 Web Technologies
Homework: Issue date: 20th September 2023
Date of Submission: 22nd September 2023

Lt Col. T Varadharajaperumal(Retd)
Student Id : 23MX128

Qsn 1. What do you mean by progressive web Apps? How is it different from normal web apps?

Ans 1. Progressive Web Apps (PWAs) are a type of web application that combines the best features of both traditional websites and native mobile applications. They are designed to provide a more engaging and app-like experience to users while retaining the accessibility and reach of the web. Some key differences and similarities between them: -

(a) **Offline Functionality:** PWAs use service workers, which are JavaScript files that run in the background, to cache essential resources. This means that even when users are not connected to the internet, a PWA can still load and display content. Normal web apps typically require an active internet connection to function.

(b) **App-Like Experience:** PWAs offer a more app-like experience with features such as smooth animations, gestures, and interactions. They can be installed on a user's device, which creates an icon on the home screen and allows users to launch the PWA from there, similar to native apps. This makes PWAs feel like a part of the user's device, whereas normal web apps are accessed through a web browser.

(c) **Responsive Design:** Both normal web apps and PWAs can be designed to be responsive, adapting to various screen sizes and devices. However, PWAs often put a stronger emphasis on responsive design to ensure a seamless experience across desktop and mobile devices.

(d) **Push Notifications:** PWAs can send push notifications to a user's device, even when the PWA is not actively open. This feature is similar to what native mobile apps offer and can help engage users and keep them informed. Normal web apps do not have this capability.

(e) **Offline Storage:** PWAs use local storage and indexed DB to store data on the user's device, enabling them to function even when there is no internet connection. This allows for a more continuous user experience. Normal web apps typically rely on server-side data and require an active internet connection to operate.

(f) **Installation:** PWAs can be installed on a user's device without going through an app store. Users can add them to their home screens directly from the web browser. Normal web apps are accessed by navigating to a website in a browser.

(g) **App Store Distribution (Optional):** While PWAs can be installed directly from the web, they can also be submitted to app stores (e.g., Google Play Store) as Progressive Web App equivalents. This can increase their discoverability. Normal web apps are not typically distributed through app stores.

(h) **Updates:** PWAs are updated automatically, so users always have the latest version. In contrast, users may need to manually update normal web apps by refreshing the web page.

(j) **Platform Independence:** PWAs are designed to work across different platforms and browsers, making them highly portable. Normal web apps also work on various platforms but may require adjustments for optimal performance and user experience.

Summary: Progressive Web Apps are a specific type of web application that offers enhanced user experiences, including offline functionality, push notifications, and a more app-like feel, while still leveraging the advantages of web technologies like cross-platform compatibility and ease of distribution. Normal web apps, on the other hand, are typically accessed through web browsers and may not offer the same level of functionality and integration as PWAs.

Qsn 2. Compare and contrast http and https.

Ans 2. HTTP (Hypertext Transfer Protocol) and HTTPS (Hypertext Transfer Protocol Secure) are both protocols used for transmitting data over the internet, but they differ significantly in terms of security and data protection. Following points are the comparisons and contrast of HTTP and HTTPS:

(a) **Security**:

(i) HTTP: HTTP is not secure. Data transmitted over HTTP is sent in plain text, which means that anyone with the right tools and access can intercept and read the data. This lack of security makes it vulnerable to various forms of attacks, such as eavesdropping and data tampering.

(ii) HTTPS**:** HTTPS is designed to provide security by encrypting the data transmitted between the client (e.g., web browser) and the server. It uses SSL (Secure Sockets Layer) or its successor, TLS (Transport Layer Security), to encrypt the data, ensuring that it cannot be easily intercepted or tampered with. This encryption provides a higher level of security and privacy for users.

(b) **Data Integrity:**

(i) HTTP**:** Since data is transmitted in plain text, there is no built-in mechanism to ensure data integrity. This means that data can be altered during transit without detection.

(ii) HTTPS**:** HTTPS ensures data integrity by using encryption and digital certificates. Any tampering or alteration of data during transmission is detected, and the connection is terminated if integrity checks fail.

(c) **Authentication**:

(i) HTTP: HTTP does not provide any authentication mechanism for verifying the identity of the website or server. This lack of authentication makes it susceptible to various forms of attacks, including man-in-the-middle attacks.

(ii) HTTPS**:** HTTPS uses digital certificates to authenticate the identity of the server. When you visit a website using HTTPS, your browser verifies the server's certificate to ensure that you are connecting to the legitimate server and not an impostor. This helps prevent phishing and other forms of online deception.

(d) **Search Engine Ranking:**

(i) HTTP: Search engines like Google have started to prioritize websites that use HTTPS in their search rankings. This means that HTTP may negatively impact a website's visibility and search engine optimization (SEO).

(ii) HTTPS: Using HTTPS can help improve a website's ranking in search results, which can be beneficial for online businesses and organizations.

(e) **Performance:**

(i) HTTP: HTTP may have a slight performance advantage over HTTPS because it does not require the overhead of encryption and decryption. However, the performance impact of HTTPS is often negligible for most websites, especially with modern hardware and optimized configurations.

(ii) HTTPS: While HTTPS may introduce a small amount of additional latency due to encryption and decryption, the performance impact is usually not noticeable for most users. The benefits of security and data protection outweigh this minor performance difference.

Summary: HTTPS is the secure and recommended version of the HTTP protocol for transmitting data over the internet. It provides encryption, data integrity, and authentication, making it essential for protecting sensitive information and ensuring the security of online communication. HTTP, on the other hand, is less secure and not suitable for transmitting sensitive data over the internet.

Qsn 3. How are stateless protocols different from stateful protocols? Name a few protocols in each category.

Ans 3. Stateless protocols and Stateful protocols are two different approaches to handling communication between devices or systems. They differ in how they manage and remember information about the current state of the communication session. Here is how they are different, along with some examples of protocols in each category:

(a) **Stateless Protocols:**

(i) **No Session Information**: Stateless protocols do not maintain any information about the current state or context of the communication session between two devices. Each request is treated independently, and the protocol does not remember previous interactions.

(ii) **Simplicity**: Stateless protocols are often simpler and more lightweight because they do not need to manage session state. This simplicity can make them more suitable for certain types of communication where maintaining state is unnecessary.

(iii) **Examples**:

HTTP (Hypertext Transfer Protocol): The standard web browsing protocol is stateless. Each HTTP request/response pair is independent, and the server does not inherently remember previous requests from the same client.

DNS (Domain Name System): DNS is a stateless protocol used for translating domain names into IP addresses. Each DNS query is processed independently, and the server does not maintain session state between queries.

(b) **Stateful Protocols:**

(i) **Session State**: Stateful protocols maintain information about the ongoing communication session between devices. They remember the context and state of previous interactions, allowing for more complex and interactive communication.

(ii) **Complex Interactions**: Stateful protocols are used when it is necessary to maintain session-specific information, such as user authentication or the progression of a multi-step transaction.

(iii) **Examples:**

FTP (File Transfer Protocol): FTP is a stateful protocol commonly used for file transfers. It maintains session information, including the user's login credentials and the current directory.

SMTP (Simple Mail Transfer Protocol): SMTP is used for sending email. It maintains the state of the email session, ensuring that messages are delivered properly and that communication with the mail server is consistent.

SSH (Secure Shell): SSH is a secure remote login and file transfer protocol that maintains session state. It allows users to interact with remote systems while maintaining a secure and persistent connection.

TCP (Transmission Control Protocol): While not a standalone application-layer protocol, TCP is a transport-layer protocol that provides stateful, reliable communication by maintaining connection state, acknowledging data receipt, and retransmitting lost packets.

Summary: The key difference between stateless and stateful protocols lies in their approach to maintaining session information. Stateless protocols treat each interaction as independent, while stateful protocols maintain session state, allowing for more complex and interactive communication scenarios. The choice between these two types of protocols depends on the specific requirements of the communication and the desired level of complexity and state management.

Qsn 4. Compare and contrast two-tier and three-tier web architecture.

Ans 4. Two-tier and three-tier web architectures are two common approaches for designing the structure of web applications. They differ in terms of how they distribute the components and responsibilities within the application. Here is a comparison and contrast of two-tier and three-tier architectures:

Two-Tier Web Architecture:

1. **Components:**
   - (a) Client: In a two-tier architecture, the client (usually a web browser) handles the presentation layer and user interface. It interacts directly with the backend database or data store for data retrieval and storage.
   - (b) Server: The server, also known as the application server or database server, combines the business logic and data access layers into a single unit. It directly communicates with the database to perform CRUD (Create, Read, Update, Delete) operations.
2. **Scalability and Maintenance:**
   - (a) Scalability: Two-tier architectures can be less scalable than three-tier architectures because both the presentation and business logic are tightly coupled on the server. Scaling requires scaling the entire server, which can be inefficient.
   - (b) Maintenance: Maintenance can be more challenging in two-tier architectures because changes to the business logic often require modifications to both the client and server components.
3. **Flexibility:**
   - (a) Flexibility: Two-tier architectures can be less flexible when it comes to adapting to changing requirements or integrating with different client types (e.g., mobile apps) because the business logic is closely tied to the client application.

Three-Tier Web Architecture:

1. **Components:**
   - (a) Client: The client handles the presentation layer and user interface, just like in a two-tier architecture.
   - (b) Application Server (Middle Tier): The application server, also known as the middle tier, is responsible for hosting the business logic. It acts as an intermediary between the client and the data store. It processes client requests, interacts with the database, and returns data to the client.
   - (c) Database Server (Data Tier): The database server stores and manages the data. It is separate from the application server and is responsible for data storage, retrieval, and management.
2. **Scalability and Maintenance:**

(a) Scalability: Three-tier architectures are more scalable because you can scale each tier independently. If the database server experiences high load, you can scale it separately from the application server. This flexibility improves performance and resource utilization.

(b) Maintenance: Maintenance is often more straightforward in three-tier architectures because changes to the business logic can be made in the application server without impacting the client. It also allows for easier updates and maintenance of the database schema without affecting the client.

3. **Flexibility:**

(a) Flexibility: Three-tier architectures are more flexible and adaptable to changing requirements. The separation of concerns between layers allows for easier integration with different client types and the ability to modify the business logic independently of the presentation layer.

Summary: The main difference between two-tier and three-tier architectures is the level of separation and distribution of components. Two-tier architectures combine the business logic and data access layers into a single unit on the server, while three-tier architectures introduce a middle tier that separates the business logic from the data layer. Three-tier architecture offers advantages in terms of scalability, maintenance, and flexibility, making them a popular choice for building modern web applications.

Qsn 5. What is a web server? How is it different from an application server?

Ans 5. A web server and an application server are both critical components in the architecture of web applications, but they serve different purposes and have distinct roles within the application stack. Below is a brief comparison of the two servers:

Web Server:

1. Purpose:
   A web server is responsible for handling HTTP requests and responses. It primarily deals with static content and acts as an intermediary between the client (typically a web browser) and the web application.
2. Functionality:
   It serves static files such as HTML, CSS, JavaScript, images, and other assets directly to clients without any dynamic processing.
   It may also handle URL routing to direct requests to specific files or routes within the application.
   Web servers often support features like load balancing, caching, and SSL/TLS termination to improve performance and security.
3. Examples:
   Apache HTTP Server
   Nginx
   Microsoft Internet Information Services (IIS)
   LiteSpeed
   Caddy
4. Use Cases:
   Web servers are suitable for serving static websites or web applications where the content does not change frequently.
   They can also be used in conjunction with application servers to serve static assets and offload some of the HTTP-related tasks.

Application Server:

1. Purpose:
   An application server is responsible for executing the business logic of a web application. It processes dynamic content and interacts with databases, external services, and other components to generate responses to client requests.
2. Functionality:
   It processes application-specific logic, such as user authentication, data validation, and complex computations.
   Application servers communicate with databases to retrieve or update data.

They manage session state and user sessions for stateful applications.
Application servers often support multiple programming languages and frameworks.

3. Examples:
   - Tomcat (for Java-based applications)
   - Ruby on Rails (Ruby)
   - Django (Python)
   - Express.js (Node.js)
   - ASP.NET (C#)
4. Use Cases:
   Application servers are essential for web applications that require dynamic content generation, user authentication, database interactions, and complex server-side logic.
   They are responsible for processing and responding to HTTP requests after the web server forwards them.

Key Differences:

- **Responsibilities**: A web server primarily serves static content and manages HTTP-related tasks, while an application server focuses on executing the application's business logic, processing dynamic requests, and interacting with databases.
- **Content:** Web servers handle static files, while application servers generate dynamic content.
- **Layer:** Web servers are typically part of the presentation layer, whereas application servers are part of the application layer in a multi-tier architecture.
- **Examples:** Web servers include Apache and Nginx, while application servers include Tomcat, Ruby on Rails, and Django.

In many modern web applications, both web servers and application servers are used together to create a complete web stack. The web server handles the initial request, serves static assets, and forwards dynamic requests to the application server, which then processes them and generates the appropriate responses. This separation of concerns allows for better scalability, flexibility, and maintainability in web application development.