|  |
| --- |
| CSA- 515 |

|  |
| --- |
| Web Development |

|  |
| --- |
| ( 3 credits ) - Year 2024-25 |

|  |
| --- |
| *Module 4 : HTTP and Middleware* |

|  |  |
| --- | --- |
| |  | | --- | | Faculty: H. H. Redkar | |

|  |
| --- |
| Master of Computer Applications (MCA) - Sem II  Computer Science Technology, Goa Business School, Goa University |

|  |  |
| --- | --- |
| |  | | --- | | Web Development : Module 4 | |

|  |  |
| --- | --- |
| HTTP & Middleware | |
| - | HTTP, Request & Response, methods & error code, headers, URL |

|  |  |
| --- | --- |
| - - | encoding & decoding  XML, data & XPath  JSON |

|  |
| --- |
| *(Link to Module 0)* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP & Middleware | |

|  |  |
| --- | --- |
| **Middleware** | |
| - | **Software tools** that act as ***intermediaries between different applications,*** |

|  |  |
| --- | --- |
| - | ***systems, or services,*** facilitating their communication and interaction.  **Middleware handle**various tasks such as***data translation, message queuing,*** |

***authentication, and connectivity***, making it easier to integrate and manage

|  |  |
| --- | --- |
| - | complex software environments.  Middleware typically***handles authentication, communications, data*** |

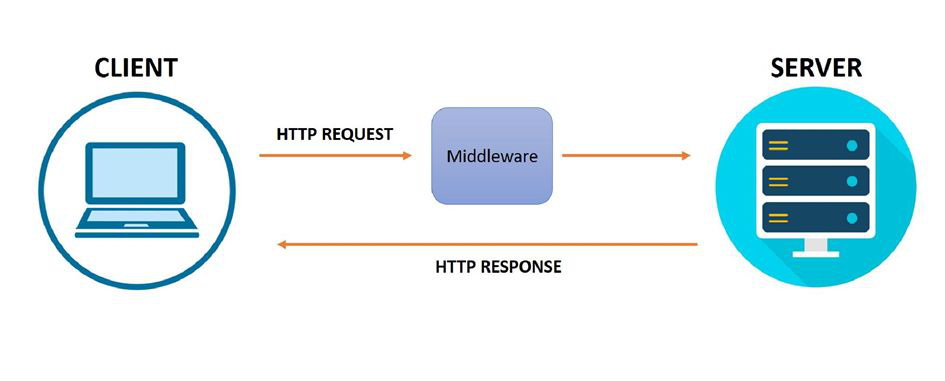
|  |  |
| --- | --- |
| - | ***management, application services, and API***management.  **Examples**include database middleware, web server middleware,and |

message-oriented middleware, cloud services of all kinds, enterprise application

|  |
| --- |
| integration, application runtimes, and**HTTP middleware.** |

|  |
| --- |
| *Source: <https://www.geeksforgeeks.org/what-is-middleware/>* |

|  |  |
| --- | --- |
| |  | | --- | | Middleware | |



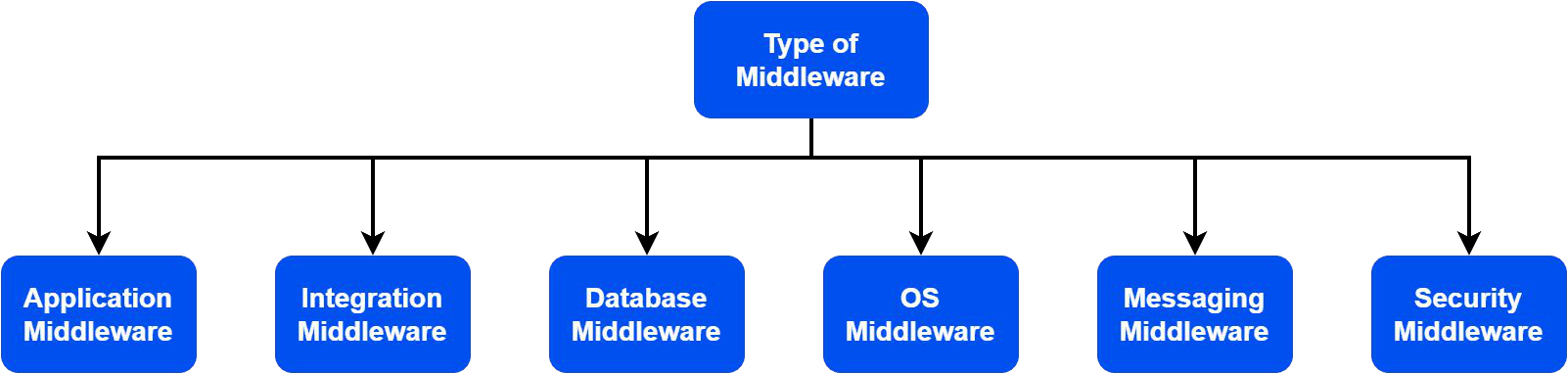
|  |
| --- |
| *Source: <https://medium.com/@priyaeswaran/middleware-in-javascript-8af7166494d3>* |

|  |  |
| --- | --- |
| |  | | --- | | Middleware | |

|  |
| --- |
|  |

|  |
| --- |
| *Source: <https://www.baeldung.com/cs/middleware>* |

|  |  |
| --- | --- |
| |  | | --- | | Middleware | |



|  |
| --- |
| **Types of Middleware** |

|  |
| --- |
| We use **application middleware** in web applications to handle [HTTP](https://www.baeldung.com/cs/popular-network-protocols#:~:text=Hypertext%20Transfer%20Protocol%20(HTTP)&text=HTTP%20is%20an%20application%20layer,application%20like%20a%20web%20browser.) requests and responses. |

|  |
| --- |
| We can utilize them for tasks such as authentication, logging, and error handling. |

|  |
| --- |
| Examples of **application middleware** include [Express.js](https://en.wikipedia.org/wiki/Express.js), [Django framework](https://en.wikipedia.org/wiki/Django_(web_framework)), [Apache Tomcat](https://en.wikipedia.org/wiki/Apache_Tomcat), and [Microsoft IIS](https://en.wikipedia.org/wiki/Internet_Information_Services). |

|  |
| --- |
| Examples of **integration middleware** include the [enterprise service bus (ESB)](https://en.wikipedia.org/wiki/Enterprise_service_bus), [message queueing systems](https://en.wikipedia.org/wiki/Message_queue), and [API](https://en.wikipedia.org/wiki/API_management) |

|  |
| --- |
| [gateways](https://en.wikipedia.org/wiki/API_management). |

|  |
| --- |
| Examples of **database middleware** are [Oracle GoldenGate](https://en.wikipedia.org/wiki/Goldengate), [MySQL Replication](https://en.wikipedia.org/wiki/MySQL_Cluster), and [DBCP](https://en.wikipedia.org/wiki/Apache_Commons). |

|  |
| --- |
| *Source: <https://www.geeksforgeeks.org/what-is-middleware/>* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP & Middleware | |

|  |  |
| --- | --- |
| **Middleware** | |
| - | **Software tools** that act as ***intermediaries between different applications,systems, or*** |

|  |  |
| --- | --- |
| - | ***services,*** facilitating their communication and interaction.  **Middleware handle**various tasks such as***data translation, message queuing,*** |

***authentication, and connectivity***, making it easier to integrate and manage complex software

|  |  |
| --- | --- |
| - | environments.  Middleware typically***handles authentication, communications, data management,*** |

|  |  |
| --- | --- |
| - | ***application services, and API***management.  **Examples**include database middleware, web server middleware,and message-oriented |

middleware, cloud services of all kinds, enterprise application integration, application

|  |
| --- |
| runtimes, and**HTTP middleware.** |

|  |
| --- |
| *Source: <https://www.geeksforgeeks.org/what-is-middleware/>* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP | |

|  |  |
| --- | --- |
| HTTP : Hypertext Transfer Protocol | |
| - | HTTP is the **foundation of data communication on the World Wide Web** enabling web browsers and |

servers ***to exchange information,*** including HTML documents, images, and other web resources.

|  |
| --- |
|  |

|  |
| --- |
| *Source: Google, GeeksforGeeks* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP | |

|  |
| --- |
| HTTP : Hypertext Transfer Protocol |

|  |
| --- |
|  |

|  |
| --- |
| *Source: Google, GeeksforGeeks* |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | HTTP | | |
| - | **Functionality:** HTTP is an **application-layer protocol** that governs how information, such as web pages, |

|  |  |
| --- | --- |
| - | images, and multimedia content, are transferred between a web server and a client (e.g., a web browser). **Request-Response Model:** HTTP **follows a client-server model,** where a client (browser) makes a request |

|  |  |
| --- | --- |
| - | to a server for a resource, and the server responds by sending the requested data.  **Stateless Protocol:** HTTP is **stateless**, meaning each request is treated independently, and the server doesn't |

keep track of previous interactions between the client and server. Cookies and other mechanisms are used to

|  |  |
| --- | --- |
| - | maintain state across multiple requests.  **Versions:** HTTP has evolved through v**arious versions**, including HTTP/1.0, HTTP/1.1, HTTP/2, and the |

|  |  |
| --- | --- |
| - | latest, HTTP/3.  **Methods:** HTTP defines **various methods (or "verbs")** to perform different actions on a resource, including |

|  |  |
| --- | --- |
| - | GET (to retrieve data), POST (to submit data), PUT (to replace data), DELETE (to delete data), and others. **HTTPS:** HTTPS is the **secure version of HTTP**, using encryption to ensure secure communication by adding |

|  |  |
| --- | --- |
| - | the "s" for secure.  **HTTP/3:** The latest iteration of the HTTP protocol, known for improvements in performance and efficiency |

|  |
| --- |
| *Source: Google* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Middleware | |

|  |  |
| --- | --- |
| HTTP Middleware | |
| - | HTTP middleware **acts as a layer between incoming HTTP requests and the application's core** |

**logic,** enabling tasks like authentication, logging, and routing to be handled efficiently without cluttering

|  |  |
| --- | --- |
| - | the main application code.  HTTP middleware ***examine the incoming HTTP request***, modify it, or modify the response before it's |

sent back to the client

|  |
| --- |
|  |

|  |
| --- |
| *Source: Google* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Middleware | |

|  |  |
| --- | --- |
| HTTP Middleware | |
| - | **Examples of Middleware Tasks:** |

|  |  |  |
| --- | --- | --- |
| **-** | - | **Authentication:** Verifying user credentials. |

|  |  |
| --- | --- |
| - - | **Authorization:** Determining user permissions. |

|  |  |
| --- | --- |
| - - | **Logging:** Recording HTTP requests and responses. |

|  |  |
| --- | --- |
| - - | **CORS:** Handling Cross-Origin Resource Sharing. |

|  |  |
| --- | --- |
| - - | **Request/Response Modification:** Altering the request or response before/after processing. |

|  |  |
| --- | --- |
| - - | **Routing:** Mapping URLs to specific handlers. |

|  |  |
| --- | --- |
| - - | **Data Security:** Protecting sensitive data. |

|  |
| --- |
| *Source: Google* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Request / Response | |

|  |
| --- |
| HTTP Request and HTTP Response |

|  |  |
| --- | --- |
| Communication between clients and servers is done by **requests** and **responses**: | |
| ● | A client (a browser) sends an **HTTP request** to the web |
| ● | A web server receives the **request** |
| ● | The **server runs an application** to process the request |
| ● | The s**erver returns an HTTP response** (output) to the browser |
| ● | The client (the browser) r**eceives the response** |

|  |
| --- |
| **Example:** |

|  |  |
| --- | --- |
| A typical HTTP request / response circle: | |
| 1. | The browser requests an HTML page. The server returns an HTML file. |
| 2. | The browser requests a style sheet. The server returns a CSS file. |
| 3. | The browser requests an JPG image. The server returns a JPG file. |
| 4. | The browser requests JavaScript code. The server returns a JS file |
| 5. | The browser requests data. The server returns data (in XML or JSON). |

|  |
| --- |
| *Source: <https://www.w3schools.com/whatis/whatis_http.asp>* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Request/Response | |

|  |
| --- |
| HTTP Request and HTTP Response |

|  |
| --- |
| XHR - XML Http Request |

|  |
| --- |
| All browsers have a **built-in XMLHttpRequest Object (XHR).** |

|  |
| --- |
| **XHR is a JavaScript object** that is used to transfer data between a web browser and a web server. |

|  |
| --- |
| XHR is often used to request and receive data f**or the purpose of modifying a web page.** |

Despite the XML and Http in the name, XHR is used with other protocols than HTTP, and the data can be of many different types like [HTML](https://www.w3schools.com/whatis/whatis_html.asp),[CSS](https://www.w3schools.com/whatis/whatis_css.asp),[XML](https://www.w3schools.com/whatis/whatis_xml.asp),[JSON](https://www.w3schools.com/whatis/whatis_json.asp), and plain text.

|  |  |
| --- | --- |
| The XHR Object is a Web Developers Dream, because you can: | |
| ● | Update a web page without reloading the page |
| ● | Request data from a server - after the page has loaded |
| ● | Receive data from a server - after the page has loaded |
| ● | Send data to a server - in the background |

|  |
| --- |
| The XHR Object is the underlying concept of [AJAX](https://www.w3schools.com/whatis/whatis_ajax.asp)and [JSON](https://www.w3schools.com/whatis/whatis_json.asp): |

|  |
| --- |
| *Source: <https://www.w3schools.com/whatis/whatis_http.asp>* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Request/Response | |

|  |
| --- |
| HTTP Request and HTTP Response |

|  |
| --- |
| XHR - XML Http Request |

|  |
| --- |
| The XHR Object is the underlying concept of [AJAX](https://www.w3schools.com/whatis/whatis_ajax.asp)and [JSON](https://www.w3schools.com/whatis/whatis_json.asp): |

|  |
| --- |
| *Source: <https://www.w3schools.com/whatis/whatis_http.asp>* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Methods | |

|  |
| --- |
| **HTTP methods are the different ways to request and modify data from a server.** |

|  |  |
| --- | --- |
| **GET**  ● ● ● | The most common HTTP method  Used to request data from a resource without modifying it Considered safe because it doesn't change the server's state |

|  |  |
| --- | --- |
| **POST** | |
| ● | Used to submit data to a resource for processing |
| ● | Includes data in the request body, unlike GET |
| ● | Suitable for sending large amounts of data or sensitive information |

|  |  |
| --- | --- |
| **PUT**● ● | Used to replace or update an existing resource on the server Used in RESTful APIs and web services |

|  |
| --- |
| *Source: Google* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Methods | |

|  |  |
| --- | --- |
| **HEAD** | |
| ● | Returns metadata about a resource on the server |
| ● | Returns all the headers associated with a resource at a given URL, but does not return the resource |

|  |  |
| --- | --- |
| **TRACE** | |
| ● | Used for diagnosis purposes |
| ● | Creates a loop-back test with the same request body that the client sent to the server before |

|  |  |
| --- | --- |
| **CONNECT** | |
| ● | Used to establish a TCP/IP tunnel to the origin server |
| ● | Used to secure two-way communication through one or more HTTP proxies with SSL/TLS |

|  |
| --- |
| *Source: Google* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Error Codes | |

|  |
| --- |
| All HTTP response status codes are separated into five classes or categories. |

|  |
| --- |
| The **first digit** of the status code defines the class of response, |

|  |
| --- |
| while the **last two digits** do not have any classifying or categorization role. |

|  |  |
| --- | --- |
| There are five classes defined by the standard: | |
| ● | ***1xx informational response***– the request was received, continuing process |
| ● | ***2xx successful*** – the request was successfully received, understood, and |

|  |  |
| --- | --- |
| ● | accepted  ***3xx redirection***– further action needs to be taken in order to complete the |

|  |  |
| --- | --- |
| ● ● | request  ***4xx client error*** – the request contains bad syntax or cannot be fulfilled ***5xx server error***– the server failed to fulfil an apparently valid request |

|  |
| --- |
| *Source: Google, <https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>* |

|  |  |
| --- | --- |
| |  | | --- | | HTTP Error Codes | |

|  |
| --- |
| HTTP status codes, with some common errors including 400 (Bad Request), 401 (Unauthorized), 403 |

|  |
| --- |
| (Forbidden), 404 (Not Found), 429 (Resource Exhausted), and 500 (Internal Server Error). |

|  |  |
| --- | --- |
| Common HTTP Status Codes: | |
| ● | **200 OK:** The request was successful. |
| ● | **400 Bad Request:** The request is malformed or contains invalid data. |
| ● | **401 Unauthorized:** Authentication is required or the provided credentials are invalid. |
| ● | **403 Forbidden:** The user does not have permission to access the requested resource. |
| ● | **404 Not Found:** The requested resource does not exist. |
| ● | **429 Resource Exhausted:** You've exceeded the rate limit or quota. |
| ● | **500 Internal Server Error:** An unexpected error occurred on server side. |
| ● | **503 Service Unavailable:** The service is temporarily unavailable or overloaded. |

|  |
| --- |
| *Source: Google, <https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>* |

|  |  |
| --- | --- |
| |  | | --- | | HTML Headers | |

|  |
| --- |
| **HTTP headers are key-value pairs sent in HTTP requests and responses,**providing essential information |

about the communication between the client and server.

They include details such as c***ontent type, encoding, cache control, authentication, and more,*** helping

|  |
| --- |
| manage the behavior of HTTP transactions. |

|  |  |
| --- | --- |
| **There are four kinds of headers context-wise:** | |
| ● | **General Header:** This type of header is applied on Request and Response headers both but without |

|  |  |
| --- | --- |
| ● ● | affecting the database body.  **Request Header:** This type of header contains information about the fetched request by the client. **Response Header:** This type of header contains the location of the source that has been requested |

|  |  |
| --- | --- |
| ● | by the client.  **Entity Header:** This type of header contains information about the body of the resources like MIME |

|  |
| --- |
| type and content. |

|  |
| --- |
| *Source: Google, <https://blog.postman.com/what-are-http-headers/>* |

|  |  |  |
| --- | --- | --- |
| |  | | --- | | Web Development : Next Class | | |
| - | |  | | --- | | URL encoding & decoding | |
| - | XML, data & XPath |
| - | JSON |

|  |  |
| --- | --- |
| |  | | --- | | Thank You | |