ECE 448/528
Application Software Design

Lecture 14. Web Application

Spring 2025

Won-Jae Yi, Ph.D.

Department of Electrical and Computer Engineering
Illinois Institute of Technology

Web Application

Web Application

- A special kind of client/server application where the client runs in a web browser.
  - Allow clients to access resources on servers.
  - Allow clients to communicate with each other via servers.
- No need to install and maintain client programs on consumer computers.
  - Simplify the process to deliver newer versions of the application (and to revert if newer ones do not work well).
- Clients and servers communicate via the HTTP protocol.
  - Web servers: servers will receive HTTP requests and send HTTP responses.

Static Web Pages

- Static: as HTML files stored on web servers.
  - Plus other files like images and videos.
  - More similar to documents than applications.
- Retrieved by clients and interpreted by browsers.
  - As directed by users.
  - Or by other pages.
- Delivery can be optimized.
  - Preprocess and compress to save bandwidth requirement.
  - CDN (content delivery network) to bring pages closer to users.
  - Browsers may cache pages to eliminate network communication.

Server-Side Scripting

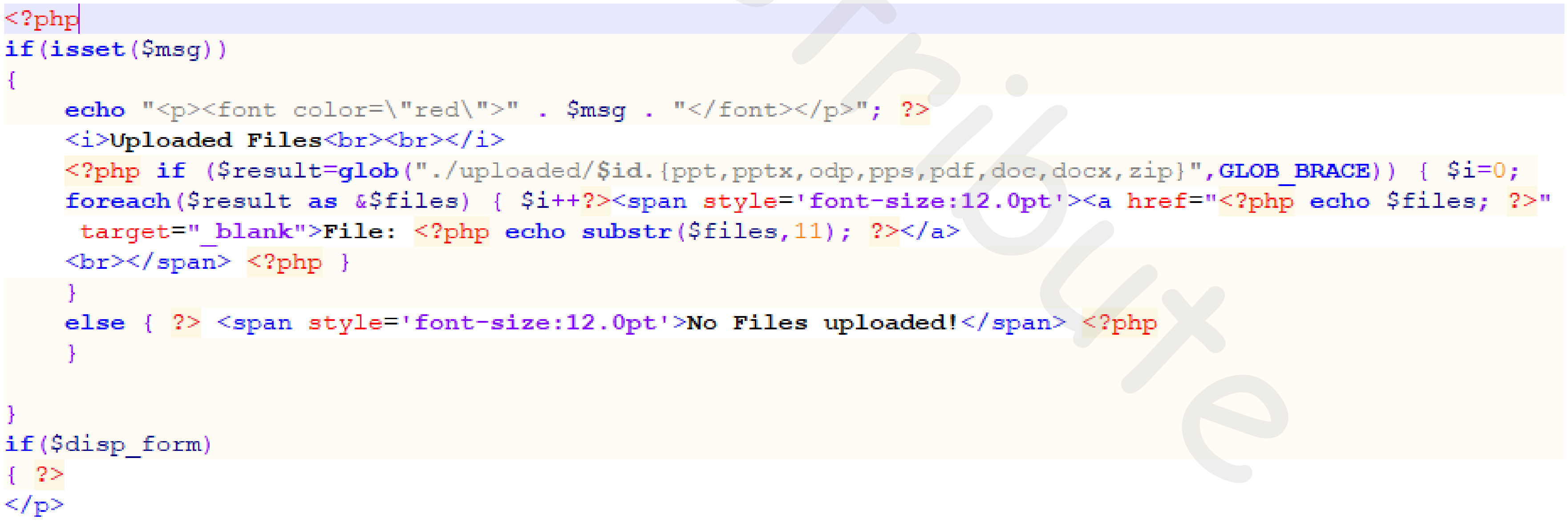
- Scripts run on web servers generate web pages on the fly as responses when requests from clients are received.
  - Incorporate data not available via a static file, e.g., from a database.
  - Allow finer control over authentication (who are you?) and authorization (what data can you access?).
  - Widely used and supported since the early days of the WWW.
- Almost all languages support a certain form of server-side scripting.
  - Early approaches usually use a separate program to generate web pages, e.g., CGI (Common Gateway Interface; middleware between WWW servers and external DBs)
  - Due to security and usability concerns, recent approaches are mostly based on templates.
- There are languages specifically designed for server-side scripting, e.g. PHP.
  - Many of them were introduced and matured in the late 90's.

PHP

- PHP: PHP Hypertext Preprocessor
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags, allowing the programmer to embed PHP scripts within HTML pages
- Interpreted language, scripts are parsed at run-time
- Executed on the server-side
- Source-code not visible by client
- Various built-in functions allow for fast development
- Compatible with many popular database

PHP Code

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag
  - <?php …. ?>
  - can be used along with HTML codes



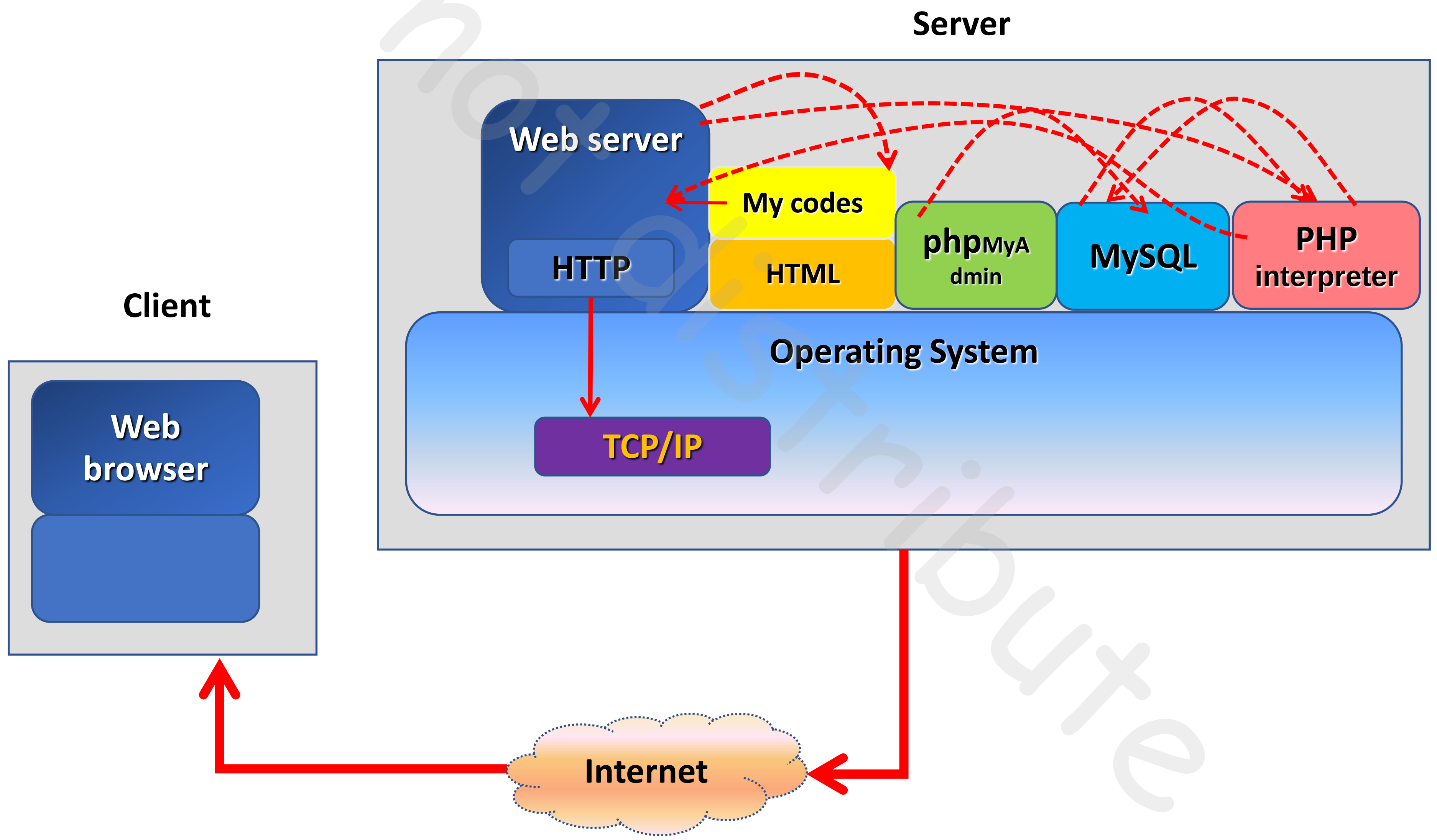
```
<?php
if(isset($msg))
{
    echo "<p><font color=\"red\">" . $msg . "</font></p>"; ?>
    <i>Uploaded Files<br><br></i>
    <?php if ($result=glob("./uploaded/$id.{ppt,pptx,odp,pps,pdf,doc,docx,zip}",GLOB_BRACE)) { $i=0;
    foreach($result as &$files) { $i++?><span style='font-size:12.0pt'><a href="<?php echo $files; ?>"
     target="_blank">File: <?php echo substr($files,11); ?></a>
    <br></span> <?php }
    }
    else { ?> <span style='font-size:12.0pt'>No Files uploaded!</span> <?php
    }

}
if($disp_form)
{ ?>
</p>
```

phpMyAdmin

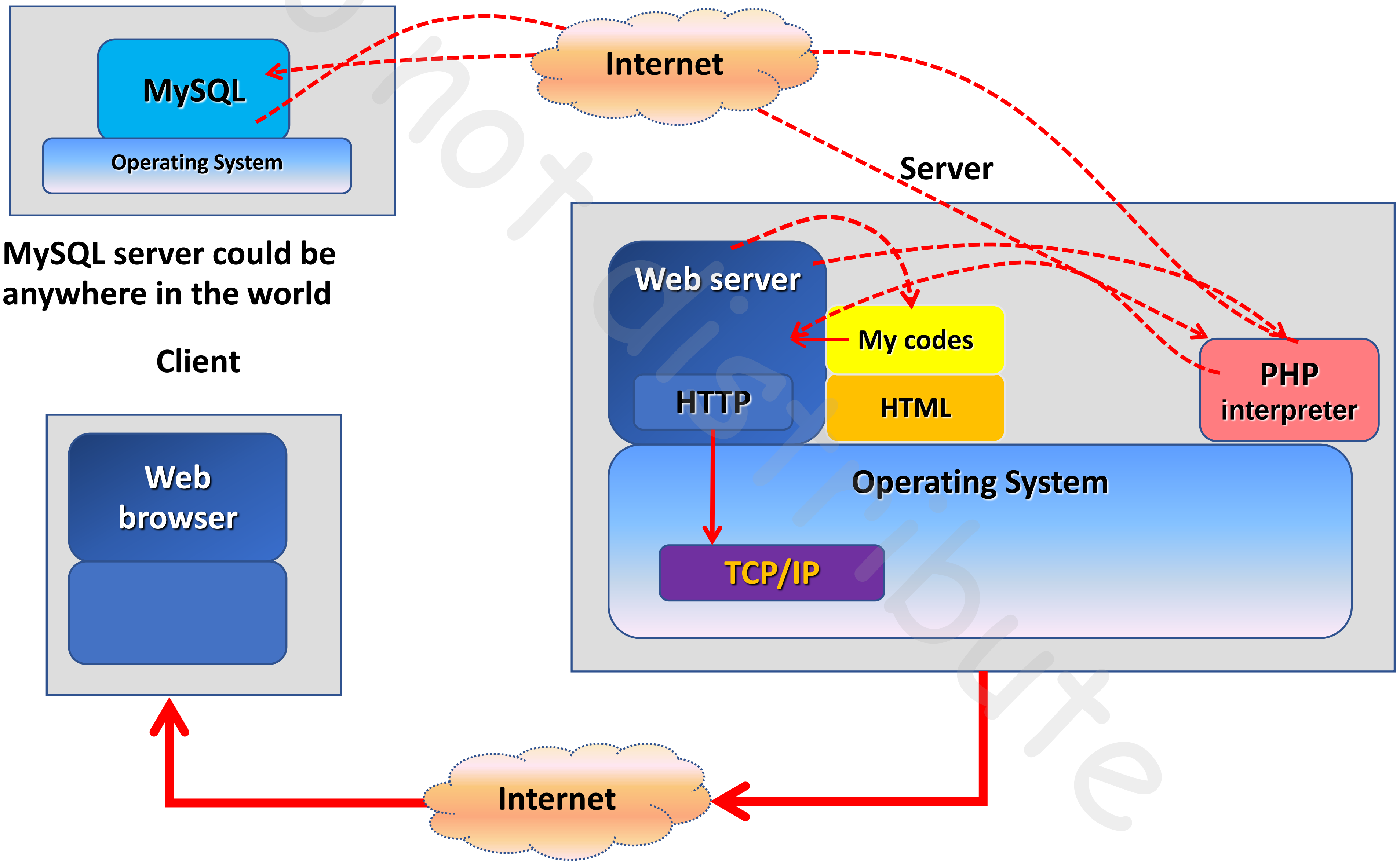
- One of the most popular applications for MySQL database management
- Free tool written in PHP
- Can create, alter, drop, delete, import and export MySQL database tablets
- Can run MySQL queries, optimize, repair and check tables, change collation and execute other database management commands
- User-friendly web interface
- Support for most MySQL functions
- Import data from CSV and SQL files
- Export data to various formats (CSV, SQL, XML, PDF, Word, Excel, LATEX, etc..)
- Searching globally in a database or a subset of it

Server – Responds to Client

- Webserver supports HTTP



Server – Responds to Clients



MySQL server could be anywhere in the world

Server-Side Scripting: Advantages

- For templates and specifically designed languages, developers usually write the programs as a mixture of code running on servers and clients.
    - Easy to work with.
    - Data are available on the server directly, usually through a database.
    - No interactions: These programs need to generate outputs from inputs.
- Browsers “just” work.
    - As if the pages are static web pages.

Server-Side Scripting: Disadvantages

- As demanded by users, today's web applications are much more complicated than the ones 10-30 years ago.
- Impacts on developers
  - Creating eye-catching web pages requires a different skill set than programming.
  - More features in a program lead to more bugs.
  - A mixture of server- and client-side code makes it hard for developers to implement authorization correctly, causing security issues (also made worse by bugs).
- Impacts on servers
  - Need additional server resources to process more data and to obtain data from multiple sources.
  - Need to support more users with fewer servers to reduce operational cost.
  - Hard to cache generated pages.
- Impacts on users.
  - Any updates to the web page need a round-trip to the web server – latency is noticeable or even not acceptable depending on the type of applications.

Client-Side Scripting

- Allow clients to update web pages by themselves.
  - Shift certain work from web servers to browsers.
- JavaScript was created in the 90s to run web pages in browsers.
  - Though the performance of JavaScript was a concern.
- Since 2008, browsers with an optimized JavaScript engine, like Google's V8 have become widely available.
  - Web pages can now be updated much faster with JavaScript.
- Note that browsers still retrieve all initial web pages, JavaScript code, etc. from web servers.

Client-Side Scripting: Architectural Implications

- If web servers no longer need to generate web pages on the fly, what should they do?
  - In addition to serving static web pages, JavaScript code, etc.
- Separation of responsibility
  - Client-side: control UI appearance using HTML/CSS and support event-driven UI updates via JavaScript.
  - Server-side: serve data instead of web pages via RESTful services.
- Advantages
  - Developers may choose to focus on either front end (client-side) or back end (server-side).
  - Easier to cache data at server-side.
  - Applications are more responsive as many UI updates no longer need round-trips to servers, and even if a round-trip is required, much fewer bytes need to be transferred.

Microservices

- With client-side scripting, instead of requiring whole web pages containing all the data, browsers can request pieces of data from different URLs.
- There is no need to store all the data on a single server.
  - There is no need to have a single service that serves everything.
- Microservices: to have many less complicated services.
  - Manageable by a small team.
  - Services are RESTful services that can be used by browsers directly.
  - RESTful services can be used by other services and mobile applications.
- Better decoupling leads to better software!

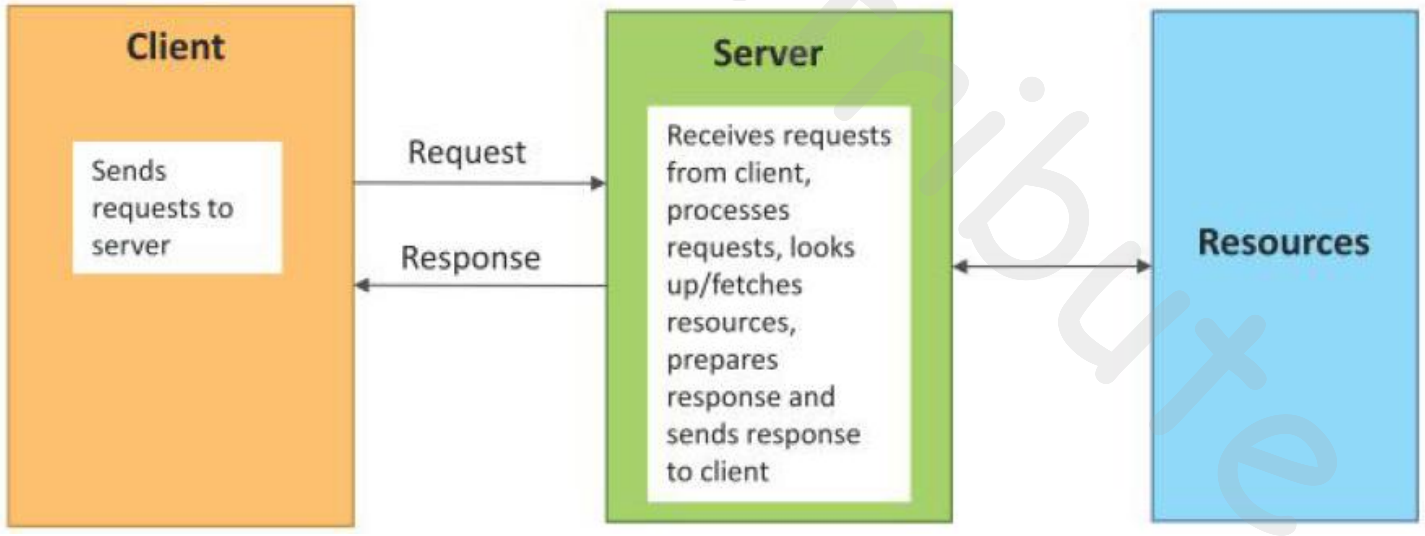
RESTful Web Services

REST: Representational State Transfer

- A software architecture for web services.
  - For clients to access data on servers via request/response.
  - For the whole system to gain desirable properties like scalability and simplicity.
- Stateless: requests from clients can be understood by servers in isolation.
  - e.g., to send a single request to switch plug X on, instead of sending a first request to select plug X, and a second request to turn it on.
- Layered: Intermediate systems may be introduced.
  - To enhance security, performance, scalability, etc.
  - e.g., proxy and reversed proxy.
- Uniform interface: conventions to map HTTP methods and paths into desired functionality.
  - Allow to better utilize existing web infrastructure, e.g., to cache GET responses since they usually do not change frequently.

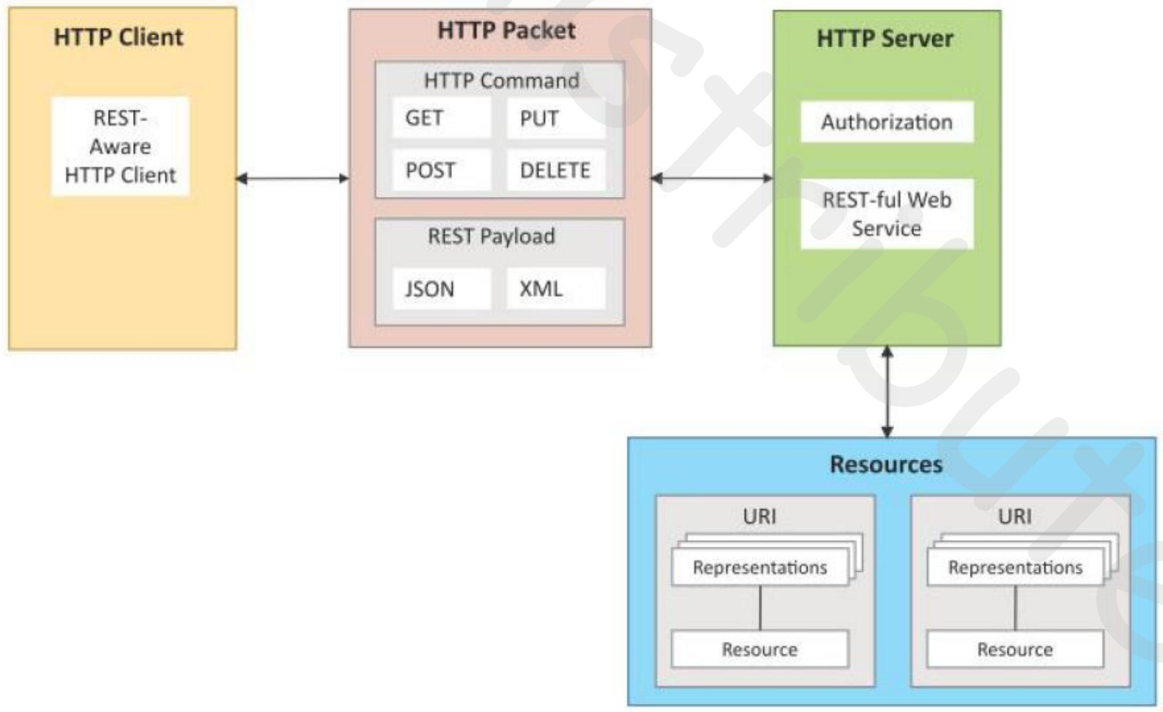
Request-Response Communication Model

- Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests.
- When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client.


Client
Sends requests to server
Request
Response
Server
Receives requests from client, processes requests, looks up/fetches resources, prepares response and sends response to client
Resources

REST-based Communication APIs

- Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.
- REST APIs follow the request-response communication model.
- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.



RESTful Web Service Practices

- Most RESTful web services today use JSON to exchange data.
  - Allow JavaScript code to process request and response directly.
  - As a request body to encode complex parameters.
  - As a response body to encode complex returned value.
  - Content-type: application/json
- RESTful web services can be accessed in most programming languages and via command line tools.
  - Possible to build RESTful web services utilizing other services using any language you are comfortable with.
- Interface design follows conventions in general.
  - We may also call these interfaces RESTful API.

CRUD and HTTP Methods

- CRUD: typical operations to manipulate data
  - Create, Read, Update, Delete.
- HTTP methods: GET, POST, PUT, DELETE, etc.
- By mapping HTTP methods to CRUD operations, we may manipulate a piece of data on the server via a single path.
  - GET → Read
  - POST → Create
  - PUT → Update
  - DELETE → Delete
  - Simplify interface design and thus make it easier to communicate such design to other developers.

References

- SQL Tutorial (https://www.w3schools.com/sql/default.asp)
- MySQL Tutorial (https://www.tutorialspoint.com/mysql/index.htm)
- PHP Tutorial (https://www.w3schools.com/php/default.asp)
- PHP MySQL Tutorial (https://www.w3schools.com/php/php_mysql_intro.asp)
- XAMPP Tutorial (https://blog.udemy.com/xampp-tutorial/)
- phpMyAdmin (https://www.phpmyadmin.net/)