Supplementary:
Web Application Development

Information Security

Application classifications

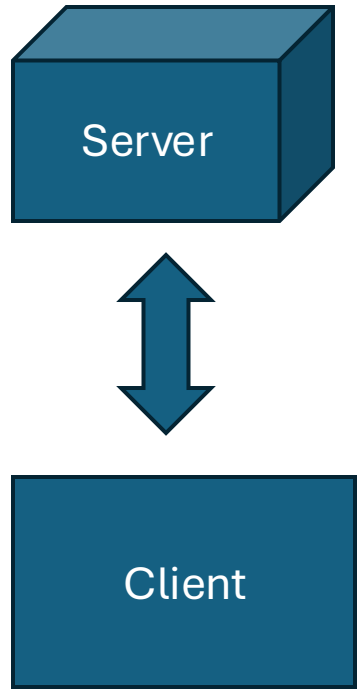
- Applications are different in many aspects:
  - Purposes
  - Users
  - *Runtime Environments*

Types of applications (by environment)

- Native Applications
  - Desktop Applications
  - Mobile Applications
- Scripting applications
- **Web Applications**
  - *Most important type of applications nowadays*

Web Application overview

- Client – Server
- Client : provide interactive user-facing interface (GUI)
- Server:
  - Provide logic according to user interactions
  - Store user data (centralized data management)



Why Web applications are IMPORTANT ?

- Ease of access and usage (for end users)
  - No software installation (just need browser)
  - Very nice GUI widgets
  - Very lightweight
- Ease of deployment and management (for service provider):
  - Centralized. Deploy once, serve many
  - Data centralization
- Excellent technology stacks
  - Highly customizable, flexible, reusable, easy to learn, quick in development

Web application technologies

- A web application = 2 programs + running at 2 separated locations
  - Frontend and backend
- Frontend: RUN INSIDE browser
- Backend: RUN INSIDE server
- Both are scripting technology

Frontend technologies

- ONE single standardized stack:
  - HTML: for representing content
  - CSS: for styling content
  - JavaScript: for handling client logics (animation, request sending, data visualization, user input)
- AGAIN: Run within the BROWSER!
  - Runtime environment = BROWSER
- DOM API: set of functions to interact with BROWSER programmatically

Backend Technologies

- MANY options to choose
  - Java, PHP, NodeJS, ... <-- called "*application server*"
  - RDBMS, noSQL <-- called "*database server*"
  - Supplementary components : redis , memcache, ...
- Easy to scale:
  - Easy to use load balancers : Haproxy, nginx/apache
- Optional only :
  - But are expected since it provides features.

Frontend – Backend Interaction

- Over HTTP
- Request – Reply cycles
- Each Request – Reply <=> 1 URL
- Each Request – Reply <=> 1 page / web object (as output)
- Example:
  - Client sends a Request (for an URL) --> backend produces output for the URL --> send the output in a Reply --> client views Reply content (render a page)

Backend Request Processing

- Backend processes request by request
  - Url by url
- No "main" function, there only multiple "request handlers"
  - Request handler = function that process a request
- Typical tasks:
  - Parsing request to get parameters (query string, headers, form, cookies, ...)
  - Query DB to fetch data or modify data in the DB
  - Format the queried data properly to make the reply

Frontend processing

- Actively sends requests
  - Manually type url in address bar
  - Javascript code sends requests (client logics): e.g. on click a button
- Wait for a reply (after sending a request)
- Process the reply:
  - Render a page
  - Manipulate the current page (AJAX)

Components of a Web Application Project

- A Web application may have:
  - Sub projects for backend + Sub projects for frontend
  - A single project that houses both frontend and backend
  - A single project for backend only (frontend code is automatically generated)
  - A single project for frontend only (static web, no backend)
- Projects' source code are managed by "source version control" (e.g.: Git)
  - GitHub/GitLab/Gerrit
- Team collaboration is supported and expected.

Frontend development skills

- Minimalist:
  - HTML + CSS
  - Javascript + DOM API
- Intermediate to Advance:
  - Minimalist and …
  - React/Angular/…

Backend Development Skills

- Choose a stack and master it:
  - PHP
  - Java
  - Python
  - NodeJS
  - ...
- SQL and database management
- Others : noSQL, Cache, load balancing, ...

Key skill: Control your ENVIRONMENTs

- Mastering your browser:
  - Developer Tools:
    - Console/Network/Element/Style
- Mastering your application server
  - How to install your stack: apt-get
  - How to configure your application server
    - edit configuration files
  - How to run your application server
    - Run app, systemctl , …
- Mastering your DB server, webserver and supplementary servers
- Mastering Git-based collaboration procedure

Lab challenge 1: Static Webserver

- Controlling nginx webserver
  - Question set 1:
    - What are command to install nginx?
    - After installation, what are commands to stop/start nginx?
    - After running nginx, how to show nginx's processes ?
    - After running nginx, how to check service ports that nginx is listening on?
  - Question set 2:
    - Where are configuration files for nginx ?
    - How many virtual servers are there by default?
    - Where is the default "document root"?
    - What is user account for running nginx ?
    - Create a new folder and make it the new "document root". Show commands and notices.

Lab challenge 2: Static Web on nginx

- Edit a simple HTML document and store it in "document root"
- Load the document via your browser.
- Questions:
  - Show content of your html
  - Show how to copy the file to nginx document root
  - What ownership and mode should you set to your html file ? Show command to change mode and ownership
  - Show screenshot that proves the html file is successfully loaded.
  - Change your html document content to add an image to the page. Show screenshot to prove your success.

Lab 3: Write Javascript code

- Go to this page and follows all the tutorial:
  - https://www.w3schools.com/js/

Lab 4: PHP-FPM application server

- Controlling PHP-FPM application server
  - Question set 1:
    - What are command to install php-fpm?
    - After installation, what are commands to stop/start php-fpm?
    - After running , how to show php-fpm's processes ?
    - After running, how to check service ports that php-fpm is listening on?
  - Question set 2:
    - What is user account for running php-fpm ?
    - User Google/GenAI to have nginx configuration for integrating with PHP-FPM. Show procedure of integration.
    - Test the stack (Nginx + PHP-FPM) with helloworld php file. Show the procedure.
    - Show PHP environment information with phpinfo() function

Control PHP environment

- Configure Nginx for PHP
- First PHP application
- PHP template (PHP tag):
  - Dynamic content vs static content
  - PHP execution and JavaScript execution
  - NEED to know what is run where
- PHP output with print()
- Configure for debugging : php.ini configuration
  - *phpinfo()* function

[Demo] Request Handling

- Request method
  - GET: get data from server
  - POST: upload data to server
  - Detecting request method with superglobal $_SERVER
- GET Request handling:
  - Working with "Query String"

[Demo] Request Handling: POST request

- Working with form
  - FORM action
  - Input element
- Process POST request

[Demo] Request Handling: Cookies

- How cookie work
  - Set-Cookie and Cookie headers
- header() function
- setcookie() function
- $_COOKIE superglobal

[Demo] Request Handling: Session in PHP

- Understand session in PHP
  - session_start()
  - session_id()
  - session_status()
  - session_destroy()

Sending request from browser

- Send request from address bar
- Send request with HTML
  - using FORM
  - Using other elements: <img>, <script>
- Send request with JavaScript
  - fetch() function

MySQL installation

- Install mysql-server in Ubuntu
- Access mysql server with *mysql* tool
- Create an account and grant privileges
- Install adminer

Working with database

- Reconfigure mysql server for listening at a network socket
- Practice:
  - Create a new database
  - Create a new user
  - Grant privileges

Working with database

- Create/Design tables in a database
- "account" table
  - Id, username, password, fullname, gender
- "friend" table:
  - account_id, friend_id

Social Network Application

- Simple web application
- Features:
  - Account Registration
  - Login/Logout
  - Make friend
  - View profile page (friend only)

Admin form

- Admin form for input user information : FOR USER ACCOUNT CREATION
- ADMIN Form Handling

- DONE

Signin

- Login Form (signin.php): Method=Post, action=our_php_script_to_handle_form (signin.php)
- Login Form handling

- PASSWORD processing:
  - Account creation:
    - PLAINTEXT_PASSWD --(password_hash)--> PASSWD_HASH (60 chars)
  - Password verify:
    - PLAINTEXT_PASSWD1 + PASSWD_HASH -- (password_verify) --> TRUE/FALSE

Implement "Home page"

- We should be able to know if an user is logged in or not!
  - If user is logged in: Show content specific to the user
  - If no user is logged in: Redirect user to signin form

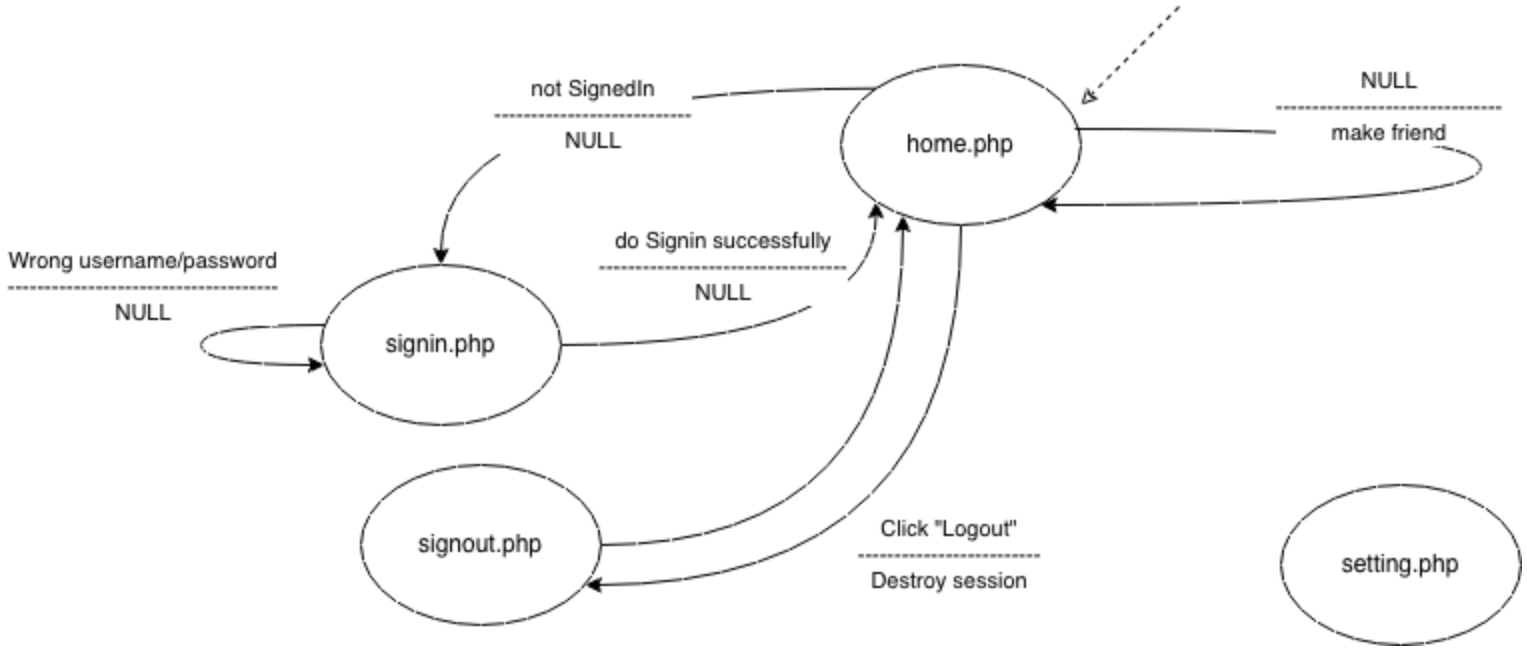
Working with "session" (phiên)

- In Signin form:
  - Successfully login ==> store username into $_SESSION
- In Home page:
  - Check if username is in $_SESSION or not. If username in $_SESSION --> already logged in.
  - Otherwise: not login

- How to work with SESSION:
  - https://www.w3schools.com/php/php_sessions.asp

Application flows

- Think about "Page transition diagram"



Page navigation -- Redirection

- If you want to bring user to another page . Use "Redirection with php"
  - Using header("Location: ..."); exit();

- PHP file content reusage:
  - require/require_once
  - Include/include_once

Working with SELECT query

- Open connection
- Execute query
- Fetch results <-- a loop may be needed.

Home page: Making friend

- Menu bar:
  - Edit profile
  - Sign out
- 2 lists:
  - List of strangers
  - List of friends
- Stranger can become Friend : via make friend button
- Friend's profile can be viewed

Profile page