Trường Đại học Công nghệ Thông tin – Đại học Quốc gia Hồ Chí Minh
Bộ môn An toàn thông tin

Web Server

Ths. Trần Tuấn Dũng – dungtran@uit.edu.vn

Introduction

A web server is specialised software that responds to client (I.e. web browser) requests

Every web site requires a web server to process client requests and ‘serve up’ the pages

Web servers used to service Internet, intranets and extranets

Note that web server in this context is software. Server machine is also referred to as the web server.

System architecture

A web server is part of a multi-tier application

Functionality is divided into separate tiers or groupings

Tiers can be on same computer or on separate computers

Web applications are often three tiered:

- Information tier (also called data tier)
- Middle tier
- Client tier (user interface tier)

Common web system architecture

Client tier

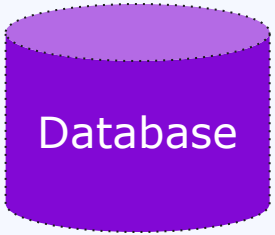
Application User interface. The client interacts with the middle tier to make requests and to retrieve data from the information tier

---

Middle tier

Controls the interactions between the application clients and application data. Enforces business rules. implements presentation logic. **Web server** typically supports this tier.

Information tier



Maintains data for the application. Data typically stores in a relational database management system (RDBMS)

Hyper Text Transfer Protocol

Basic function of web server is to act as **HTTP server**

Web servers communicate with clients using a Response-Request protocol: HTTP

Client-Server model and HTTP

- A request is generated by a client (by browser software)
  - Most common requests are “Get” and “Post”
- Request reaches the appropriate web-server
- Request is processed by the web-server
- A response is formulated by the web server and sent back to the client (e.g. web page contents)

Client-Server model and HTTP

- HTTP is the standard for transferring World Wide Web documents
- Usually to port 80 (or 443 for HTTPs)
- HTTP messages (requests and responses) between client and server are human readable

Http: Requests from client

- Request
  - Get resource
  - Type of browser
  - Name of host
  - etc
- First line is request-line. Contains the nature of the Request
  - GET: Get a file from the server
  - POST: Post data to the server
  - PUT: Update some resource

Requests from client: HTML

```
<body>
  <form method= "post" action ="process.php">
     Word to look up: <Input type="text" name="word">
   <input types="submit">
  </form>
</body>
```

Indicates a **post** request
Data in form is posted to the server

Response from web server

- Response
  - 200 = Status code
    - All's well
  - Type of server
  - Other contents etc

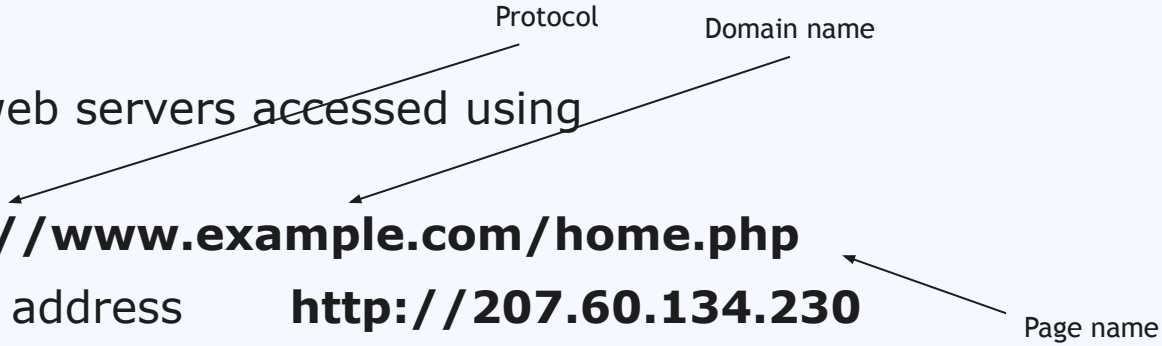
```
HTTP/1.1 200 OK
Date: Thu, 22 July 1998 18:40:55 GMT
Server: Apache 1.3.5 (Unix) PHP/3.0.6
Last-Modified: Mon, 19 July 1997 16:03:22 GMT
Content-Type: text/html
Content-Length: 12987
...
```

HTTP Response Status Codes

- 1XX: Provide information to the client
- 2XX: Correct response has occurred.
- 3XX: Browser must carry out some further action in order for the request to be successful. For example, the code 301 indicates that the resource that was requested has been permanently moved to another location.
- 4XX: Something has gone wrong; for example, the most frequent status code that is returned is 404 which indicates that the resource that has been requested cannot be found.
- 5XX: Server has experienced a problem. For example, the status code 503 indicates that the service requested has not been able to be carried out.

Accessing web servers

- Must know host name on which web server resides
- Remote web servers accessed using
  - URL: **http://www.example.com/home.php**
  - OR IP address **http://207.60.134.230**



- Local web servers (on same machine) accessed using machine name or *localhost*

Web server functionality

HTTP Server (at a minimum)

But usually includes many other functions such as:

- File Transfer Protocol (FTP) server
- Simple Mail Transfer Protocol (SMTP) server (for Email)
- Web development and publishing functionality
- Support for specific server side technologies e.g. JSP, SSIs
- Security features
- And more

Apache web server

Most popular web server (almost 70% of all web sites use it)

Freeware.

Frequent updates. Maintained by the Apache Software Foundation - Website www.apache.org
Welcome! - The Apache Software Foundation.htm

Runs on Unix, Linux and Windows operating systems

Supports a range of server-side technologies, but can require additional software installation

We learn how to use Apache with Xampp in previous lesson

Apache HTTP Server

- The Apache HTTP server is a **software (or program)** that runs in the background under an appropriate operating system, which supports **multi-tasking**, and provides services to the client web browsers.
- It is an open-source software with cross-platform functionality.
- What is **daemon?**

*In multitasking computer operating systems, a daemon is a **computer program that runs as a background process, rather than being under the direct control of an interactive user**.*

Apache HTTP Server

- Apache can be installed on a variety of operating systems. Regardless of the platform used, a hosted website will typically have four main directories:

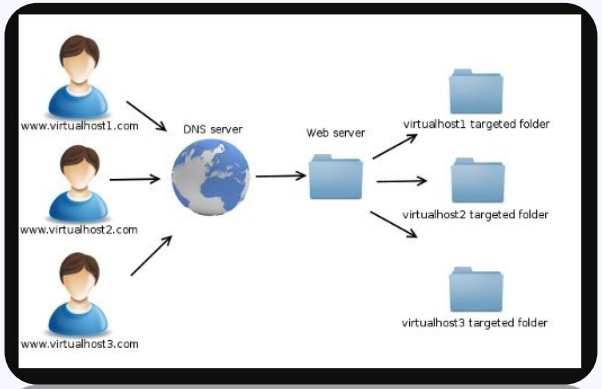
htdocs	config	logs	cgi-bin
• Static page • Dynamic content	• Plain text	• Server log • Error log	• Cgi script • Cgi program

Virtual host

- An Apache web server can host multiple websites on the SAME server. You do not need separate server machine and apache software for each website. This can achieved using the concept of Virtual Host or VHost.
- Any domain that you want to host on your web server will have a separate entry in apache configuration file.

Types of Virtual Host :

- ✔ Name based Virtual Host
- ✔ IP based Virtual Host



Configuring .htaccess

Control what user can do, where user can go on your website

Must very careful and have deep understanding

Misconfiguration can lead to security issue

Path Traversal!!

Configuring .htaccess – Example

How website redirect routes: Assuming we rename a file from **foo.html -> bar.html.** But users are familiar with **foo** so we need to make sure user access with **foo.html** still be able to access same content

We write */foo.html -> */bar.html

```
RewriteEngine  on
RewriteRule    "^/foo\.html$"  "/bar.html"
```

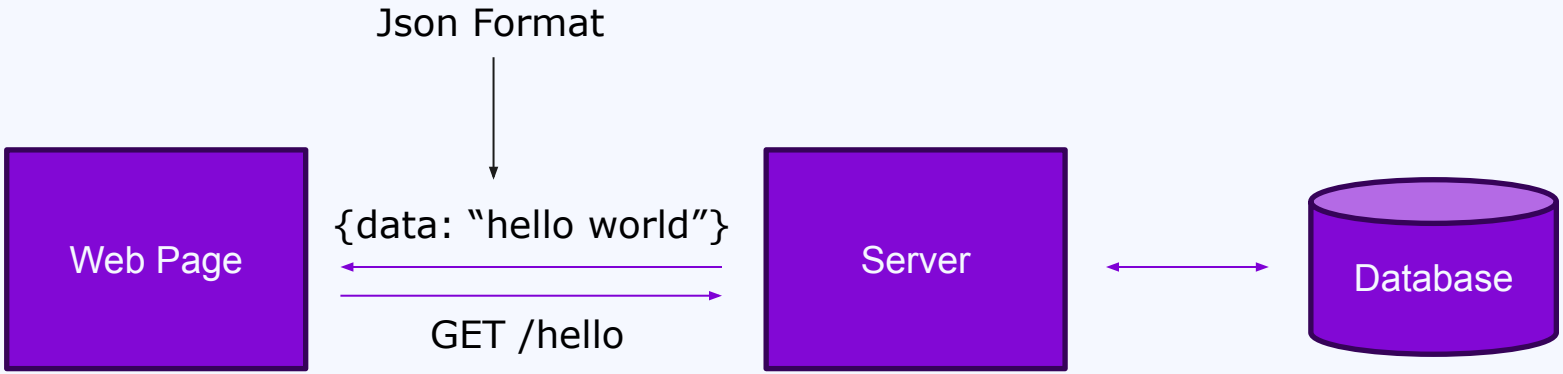
API Server

Sometimes we need a server is able to provide data and handle requests for multiple website with various technology

Web server like **Apache** will not be suitable since it require complex configuration

Flexible Server => Only need to transfer enough data for end user site

Json API


Json Format
Web Page
{data: "hello world"}
GET /hello
Server
Database

Restful API

API **Request/Response Standard**

REST (REpresentational **S**tate **T**ransfer**)** defines a set of constraints for how the architecture of a distributed, Internet-scale hypermedia system, such as the Web, should behave

Control how we use methods: **GET, POST, PUT, DELETE,...**

Control how we use status codes: 200, 400,…

Restful API

API **Request/Response Standard**

- **GET:** Retrieve data from server
- **POST**: Send data to server
- **PUT:** Update, store data on server
- **DELETE:** Remove data from server

Restful API

API **Request/Response Standard**

- 200 OK: Success Request, no error
- 201 Created: POST request and data was stored successfully
- 404 Not found: No resource found
- 400 Bad request: Wrong data posted, malform request,…
- 401 Unauthorized, 403 Forbidden: No permission
- …

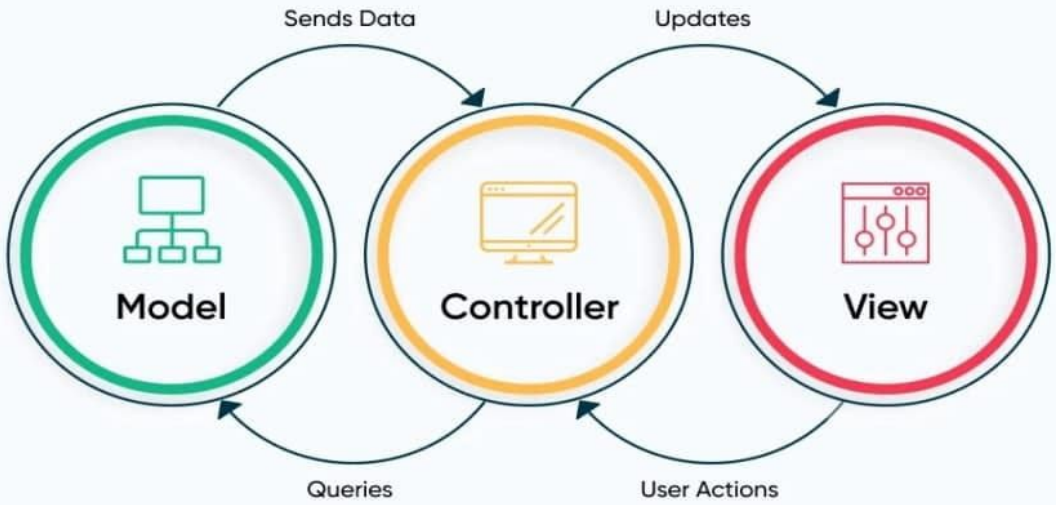
Restful API

API **Request/Response Standard**

Plural form **users**

- **GET** /users: Get all users
- **GET** /users/:id: Get user with id
- **POST** /users: Create users
- **PUT** /users/:id: Update all fields all specific user
- **PATCH** /users/:id: Update few fields
- **DELETE** /users/:id: Remove user

MVC Architecture



Laravel MVC – Models

```php
<?php namespace App;

use Illuminate\Database\Eloquent\Model;

class Car extends Model {
    //
}
```

Laravel MVC – Controllers

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Requests;
use App\Http\Controllers\Controller;
class CarController extends Controller {
/** * Display a listing of the resource. * *
@return Response */
public function index() { // }
/** * Show the form for creating a new resource. * *
@return Response */
public function create() { // }
...
```

Laravel MVC – Routes

```
Route::resource('cars', 'CarController');
```

Automatically map with functions in controllers

Laravel MVC – Views

```
public function show($id) {
     $car = Car::find($id);
     return view('cars.show', array('car' => $car));
}
```

Modify the controller to return view

```
<!DOCTYPE html>
<html>
<head>
<title>Car {{ $car->id }}</title>
</head>
<body>
<h1>Car {{ $car->id }}</h1>
<ul>
<li>Make: {{ $car->make }}</li>
<li>Model: {{ $car->model }}</li>
<li>Produced on: {{ $car->produced_on }}</li> </ul>
</body>
</html>
```

View are just HTML