Server Side Scripting

IF3110 – Web-based Application Development
School of Electical Engineering and Informatics
Institut Teknologi Bandung

Typical Web Application Processing

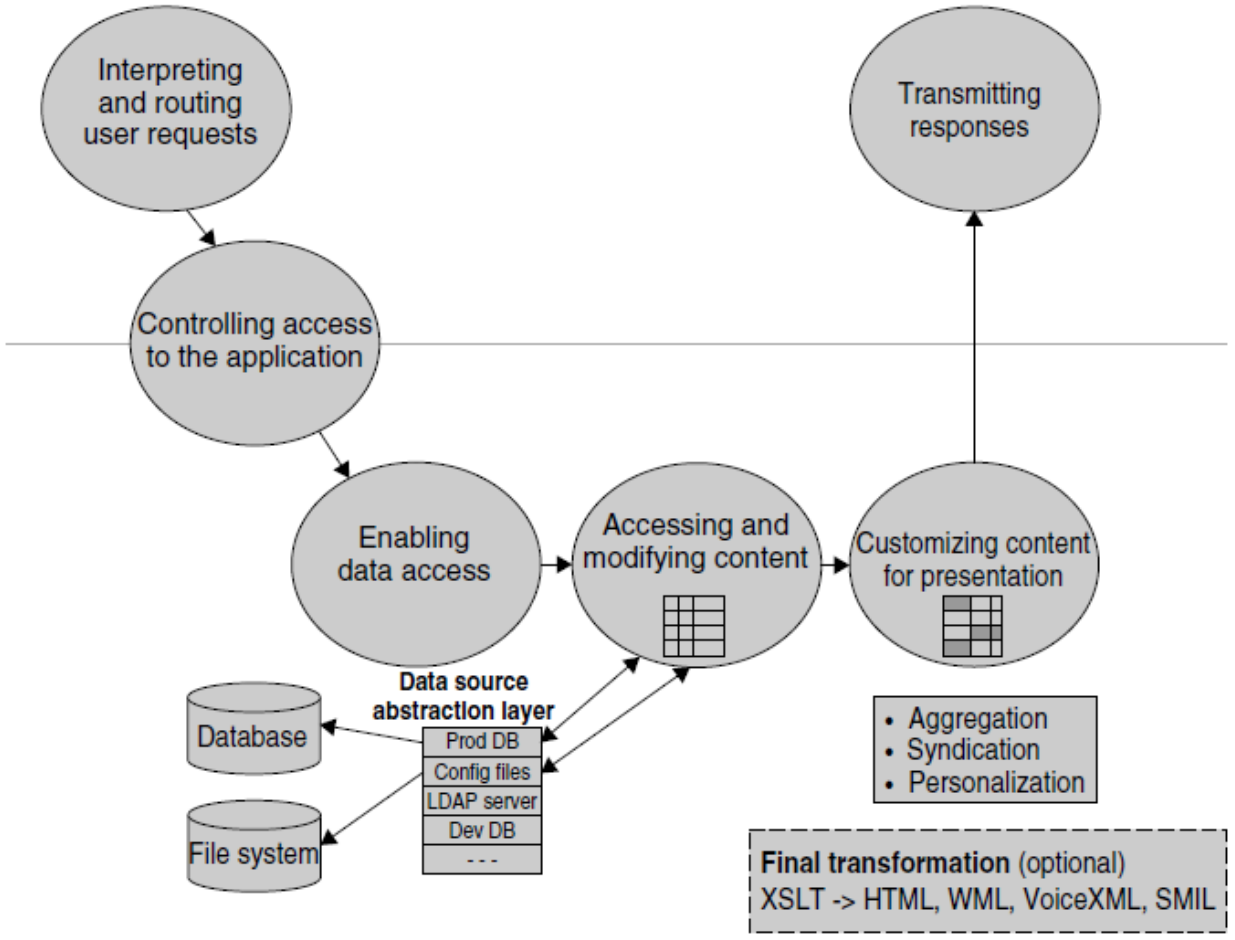


Figure 8.3 Processing flow in a typical Web application (Above the grey line—Web server; below the grey line—Web application)

Web Stack

A collection of software required to develop and run web applications

Contains all software layers

- Operating System
- Web Server
- Programming (& Markup) Language (frontend & backend)
- Web Framework (frontend & backend)
- Database Server

Examples:

- LAMP, MEAN, …

Common Stack

- MEAN/MERN/MEVN
  - MongoDB, ExpressJS, Angular/React/Vue.js, NodeJS
- WINS
  - Windows Server, IIS, .NET, (Microsoft) SQL Server
- LAMP
  - Linux, Apache, MySQL/MariaDB, PHP
- XAMPP
  - (X), Apache, MySQL/MariaDB, PHP & Perl
- etc.

Server Side Scripting Languages

- PHP (Laravel, Zend, CodeIgniter, etc.)
- JavaScript/TypeScript (Express, Nest, Hapi, etc.)
- Python (Flask, Django)
- Ruby (Rails)
- Java (Spring, etc.)
- C# (ASP.NET)
- Scala (Play)
- Go
- etc.

Server Side Aspects

- Routing
- Templating
- Form Processing
- Data Access

Interpreting & processing request

Browser communicate the request via HTTP (info sent by the browser called *request context*)

- URL, query string, request headers, request body, session information

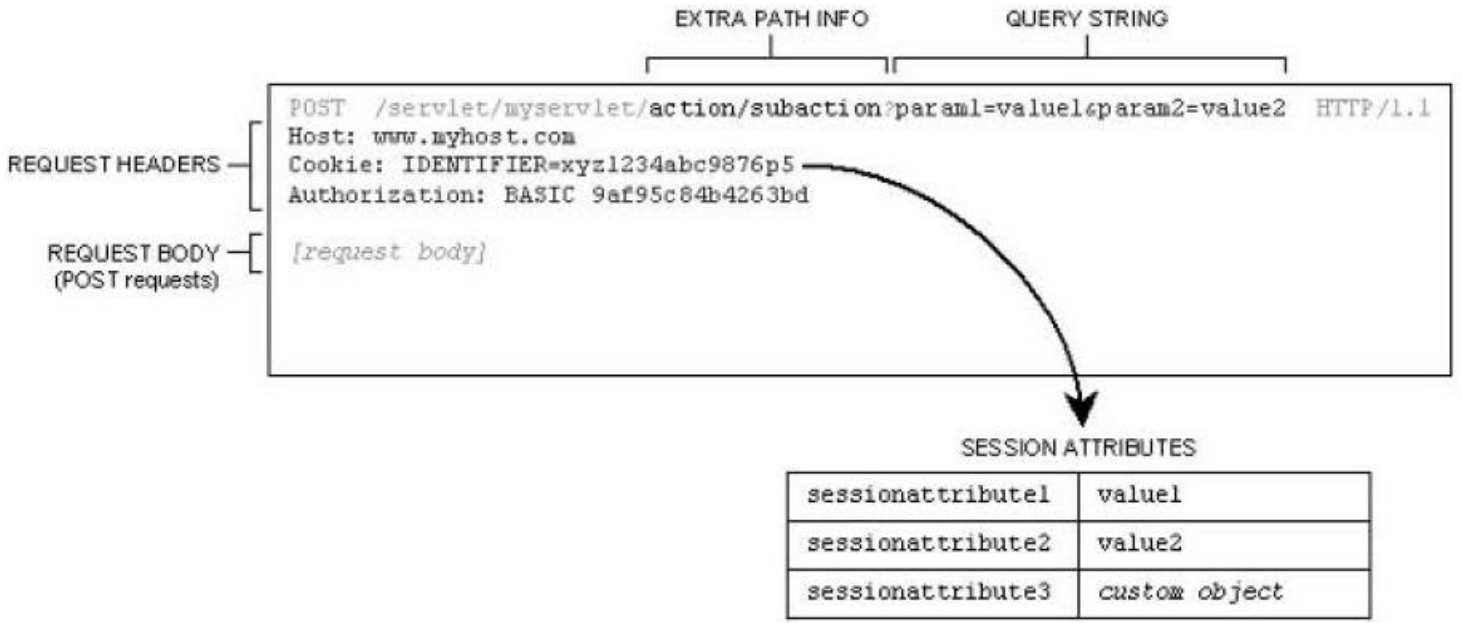


Table 8.1 Selection of server processing modules

Approach	Configuration	URL Examples
CGI	1. Server provides mechanism for defining CGI path mappings. 2. Server provides mechanism to map URL file name extensions to CGI processing.	`http://host/cgi-bin/script?...` `http://host/.../script.cgi?...`

Auxiliary processing modules (scripting, template, hybrid)	1. Server provides mechanism for registering processing modules for files with predefined name extensions.
	2. Native support may be built into server (e.g., ASP on IIS).
		`http://host/.../modulename.php?...`
		`http://host/.../modulename.cfm?...`
		`http://host/.../modulename.asp?...`

J2EE Webapp	1. Server provides mechanism for defining application path mappings.
	2. Server provides mechanism to map URL file name extensions to be processed as JSP pages.
		`http://host/servlet/servletname/...`
		`http://host/webappname/servletname/...`
		`http://host/webappname/modulename.jsp?...`
		`http://host/anotherjsp.jsp`

Routing

- Map the specific URL with an associated function
- URI syntax
  - scheme:[//authority]path[?query][#fragment]
  - authority = [userinfo@]host[:port]
  - https://website.org/one/certain/path.html
  - https://myweb.app/view
- Without routing through web application, that simply returns an HTML file
- Using a web application, an HTML file *may* be returned, among other return types.

Routing Example

- Python + Flask

```
from flask import Flask
from flask import jsonify, render_template, request

app = Flask(__name__)

@app.route(“/”)
def hello_world():
  return "<p>Hello, World!</p>“

@app.route(“/view”)
def view_route():
  return render_template(“view.html”)

@app.route(“/analyze”, methods=[POST])
def analyze_sentiment():
  text = request.form[‘text’]
  #analyze sentiment using NLP library and save to a variable “sentiment”
  return jsonify({‘text’: text, ‘sentiment’: sentiment})
```

Template

- Templates are files that contain static data as well as placeholders for dynamic data.
- A template is rendered with specific data to produce a final document.

```
from flask import Flask
from flask import jsonify, render_template

app = Flask(__name__)

@app.route(“/analyze”, methods=[POST])
def analyze_sentiment():
  text = request.form[‘text’]
  #analyze sentiment using NLP library and save to a variable “sentiment”
  return render_template(“result.html”, sentiment=sentiment, text=text)
```

Template (2)

```
<!--pre-rendered result.html-->

<html>
<head><!--necessary header components--></head>
<body>
  <h2>Text</h2>
  <p>{{ text }}</p>
  <h2>Sentiment</h2>
  <p>{{ sentiment }}</p>
</body>
</html>
```

Form Processing

- It is a usual practice for POST-type form requests to be processed using server-side programs
- The data values coming out of the form are passed to the function to be further processed

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != 'admin' or request.form['password'] != 'admin':
            error = 'Invalid Credentials. Please try again.'
        else:
            return redirect(url_for('home'))
    return render_template('login.html', error=error)
```

Form Processing (2)

```
<form action="" method="post">
  <input type="text" placeholder="Username" name="username"
    value="{{ request.form.username }}">
  <input type="password" placeholder="Password" name="password"
    value="{{ request.form.password }}">
  <input class="btn btn-default" type="submit" value="Login">
</form>
```

PHP Programming

What is PHP?

•PHP (recursive acronym for *PHP: Hypertext Preprocessor*) is a widely-used ***open source general-purpose scripting language*** that is especially suited for web development and ***can be embedded into HTML***.

•What distinguishes PHP from something like client-side JavaScript is that the code is **executed on the server**, generating HTML which is then sent to the client.

Relation between the versions

- PHP/FI 2.0 is an early and no longer supported version of PHP. PHP 3 is the successor to PHP/FI 2.0 and is a lot nicer.

- PHP 5 uses the Zend engine 2 which, among other things, offers many additional OOP features.

- PHP 6 was experimental, and never released
- PHP 7 offers major improvements, twice faster than PHP 5
- PHP 8 is the latest major version has JIT compilers, better memory management and 10-20% faster to PHP 7

What can PHP do?

- Anything.

- PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies. But PHP can do much more.

Table 1-1. Sampling of major websites that use PHP

Website name	Description	URL
Facebook	Social networking	*http://www.facebook.com*
Flickr	Photograph sharing	*http://www.flickr.com*
Wikipedia	Online collaborative encyclopedia	*http://www.wikipedia.org*
SugarCRM	Customer relationship management tool	*http://www.sugarcrm.com*
Dotproject	Project management tool	*http://www.dotproject.org*
Drupal	Website construction template engine	*http://drupal.org*
Interspire	Newsletter and email marketing product	*http://www.interspire.com*

Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

hello.php

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <p>Hello World</p>
 </body>
</html>
```

Other examples

Example #1 Printing a variable (Array element)

```php
<?php
echo $_SERVER['HTTP_USER_AGENT'];
?>
```

A sample output of this script may be:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

Example #2 Example using control structures and functions

```php
<?php
if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== FALSE) {
    echo 'You are using Internet Explorer.<br />';
}
?>
```

A sample output of this script may be:

```
You are using Internet Explorer.<br />
```

Example #3 Mixing both HTML and PHP modes

```
<?php
if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE') !== FALSE) {
?>
<h3>strpos() must have returned non-false</h3>
<p>You are using Internet Explorer</p>
<?php
} else {
?>
<h3>strpos() must have returned false</h3>
<p>You are not using Internet Explorer</p>
<?php
}
?>
```

A sample output of this script may be:

```
<h3>strpos() must have returned non-false</h3>
<p>You are using Internet Explorer</p>
```

Example #1 A simple HTML form

```
<form action="action.php" method="post">
 <p>Your name: <input type="text" name="name" /></p>
 <p>Your age: <input type="text" name="age" /></p>
 <p><input type="submit" /></p>
</form>
```

Example #2 Printing data from our form

```
Hi <?php echo htmlspecialchars($_POST['name']); ?>.
You are <?php echo (int)$_POST['age']; ?> years old.
```

A sample output of this script may be:

```
Hi Joe. You are 22 years old.
```

Language Overview

- script tag

```
<?php  php_statements..  ?>
<? php_statements..  ?>
```

- comment

```
// komentar
/* komentar */
# komentar
```

- statement
  - ended by “;”

```
$nama = "amir";
```

Data Types

- Scalar Type:
  - boolean, integer, float (or double), string
- Compound Type:
  - array, object
- Special Type:
  - resource, NULL
- pseudo-types:
  - mixed, number, callback

Examples

```php
<?php
$a_bool = TRUE;   // a boolean
$a_str  = "foo";  // a string
$a_str2 = 'foo';  // a string
$an_int = 12;     // an integer

echo gettype($a_bool); // prints out:  boolean
echo gettype($a_str);  // prints out:  string

// If this is an integer, increment it by four
if (is_int($an_int)) {
    $an_int += 4;
}

// If $bool is a string, print it out
// (does not print out anything)
if (is_string($a_bool)) {
    echo "String: $a_bool";
}
?>
```

Array

- An array in PHP is actually an ordered map.
- A map is a type that associates ***values*** to ***keys***.
- This type is optimized for several different uses; it can be treated as an:
  - array,
  - list (vector),
  - hash table (an implementation of a map),
  - dictionary, collection, stack, queue, and probably more.
- As array values can be other arrays, trees and multidimensional arrays are also possible.

Example

```php
<?php
$arr = array("foo" => "bar", 12 => true);

echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

```php
<?php
$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));

echo $arr["somearray"][6];    // 5
echo $arr["somearray"][13];   // 9
echo $arr["somearray"]["a"];  // 42
?>
```

```php
<?php
// This array is the same as ...
array(5 => 43, 32, 56, "b" => 12);

// ...this array
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Array operations

Array access

```
$buah[0] = "apel";
$buah[1] = "mangga";
```

Adding array element

```
$buah[] = "apel";
$buah[] = "mangga";
```

length of array

```
$len = count( $buah );
```

associative array

```
$pengguna["nama"] = "amir";
$pengguna["alamat"] = "ganesha 10";
```

Array Init

- Intialization

```
 $hari = array("senin", "selasa", "rabu", "kamis",
"jumat", "sabtu", "minggu");

 $hari = array(1=>"senin", "selasa", "rabu", "kamis",
"jumat", "sabtu", "minggu");

 $days = array("mon"=>"monday", "tue"=>"tuesday",
   "wed"=>"wednesday", "thu"=>"thursday",
"fri"=>"friday",
   "sat"=>"saturday", "sun"=>"sunday");
```

Array access (1)

```
<?php
    $Cities = array(
        "Jawa Barat"=>array(
            "Bandung",
            "Cianjur",
            "Cirebon"
            ),
        "Jawa Tengah"=>array(
            "Semarang",
            "Magelang"
            )
        );
    print($Cities["Jawa Barat"][1]);
?>
```

Array access (2)

```
<?php

  $buah[0] = "apel";
  $buah[1] = "mangga";
  $buah[2] = "jambu";
  $len = count( $buah );
  for( $i = 0; $i < $len; $i++) {
    echo $buah[$i], "<br>";
  }
?>
```

Array access (3)

```
<?php

$days = array("mon"=>"monday", "tue"=>"tuesday", "wed"=>"wednesday",
"thu"=>"thursday", "fri"=>"friday", "sat"=>"saturday",
"sun"=>"sunday");

  foreach( $days as $key=>$value) {
    echo "key: ", $key, ", value: ", $value, "<br>";
  }
?>
```

Function

definition:

```
function f($param1, $param2 ..) { statements.. }
```

return value

```
return $val;
```

parameter by reference

```
function f(&$param1, &$param2 ..) { statements.. }
```

return by reference

```
function &f($param1, $param2 ..) { .. return $v }
```

dynamic parameter access

```
func_get_arg($i), func_num_args()
```

Function

- variable scope
  - use `global` to access global variable with in a function

```
<?php
  function assignName() {
    // // global $name;
   echo $nglobal $name = "Zeev";
 ame;
 }
  global $name;
  $name = "Leon";
  assignName();
  print($name);
?>
```

Function

- static variable
  - store variable state with in a function
- dynamic function call
  - function execution can be done dynamically, by storing the function name in a variable and call the variable as a function

```php
<?php
    function printBold($text)
    {
        print("<b>$text</b>");
    }

    print("This Line is not Bold<br>\n");
    printBold("This Line is Bold");
    print("<br>\n");
    print("This Line is not Bold<br>\n");
?>
```

```php
<?php
    function makeBold($text)
    {
        $text = "<b>$text</b>";
        return($text);
    }

    print("This Line is not Bold<br>\n");
    print(makeBold("This Line is Bold") . "<br>\n");
    print("This Line is not Bold<br>\n");
?>
```

```php
<?php
    function stripCommas(&$text)
    {
        $text = str_replace(",", "", $text);
    }

    $myNumber = "10,000";

    stripCommas($myNumber);
    print($myNumber);
?>
```

```
<?
    function useColor()
    {
        static $ColorValue = "#00FF00";

        if($ColorValue == "#00FF00"){
            $ColorValue = "#CCFFCC";
        } else {
            $ColorValue = "#00FF00";
        }

        return($ColorValue);
    }

    print("<table width=\"300\">\n");
    for($count=0; $count < 10; $count++) {
      $RowColor = useColor();

      print("<tr>" .
            "<td style=\"background: $RowColor\">" .
            "Row number $count" .
            "</td>" .
            "</tr>\n");
    }
    print("</table>\n");
?>
```

```php
<?php
    function write($text)
    {
        print($text);
    }

    function writeBold($text)
    {
        print("<b>$text</b>");
    }

    $myFunction = "write";
    $myFunction("Hello!");
    print("<br>\n");

    $myFunction = "writeBold";
    $myFunction("Goodbye!");
    print("<br>\n");
?>
```

Object

- Starting with PHP 5, the object model was rewritten to allow for better performance and more features. This was a major change from PHP 4. PHP 5 has a full object model.
- Among the features in PHP 5 are the inclusions of visibility, abstract and final classes and methods, additional magic methods, interfaces, cloning and typehinting.
- PHP treats objects in the same way as references or handles, meaning that each variable contains an object reference rather than a copy of the entire object. See Objects and References

Example

```php
<?php
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

Resource

- A resource is a special variable, holding a reference to an external resource.
- Resources are created and used by special functions.
- As resource variables hold special handlers to opened files, database connections, image canvas areas and the like.

Example #1 mysql_connect() example

```php
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
if (!$link) {
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_close($link);
?>
```

NULL

- The special NULL value represents a variable with no value. NULL is the only possible value of type NULL.
- A variable is considered to be null if:
  - assigned the constant NULL
  - has not been set to any value yet
  - has been unset()

Pseudo-types

- mixed indicates that a parameter may accept multiple (but not necessarily all) types
- number indicates that a parameter can be either integer or float

•callback – a callable function or method

- void – indicates no value is returned
- Some functions like call_user_func() or usort() accept user-defined callback functions as a parameter

Type Juggling

- PHP does not require (or support) explicit type definition in variable declaration; a variable's type is determined by the context in which the variable is used. That is to say, if a string value is assigned to variable $var, $var becomes a string. If an integer value is then assigned to $var, it becomes an integer.

```php
<?php
$foo = "0";  // $foo is string (ASCII 48)
$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
```

```php
<?php
$foo = 10;   // $foo is an integer
$bar = (boolean) $foo;   // $bar is a boolean
?>
```

Variable

- Scope default: local
- To access global variable use **global** keyword
- Support **static** variable
- Can be variable of variable:
  $a=”hallo”;
  $$a=”world”; // sama dg $hallo

Predefined Variable

- $_REQUEST: variabel http request
- $_GET: variabel http GET
- $_POST: variabel http POST
- $_FILES: http file upload
- $_SESSION: variabel sesi
- $_COOKIE: http cookie
- $_ENV: variabel environment
- $_SERVER: variabel server

Input Data

- Alternative of data input source:
  - Parameter URL: $_GET, $_REQUEST
  - Form handling: $_POST, $_REQUEST, $_FILES
  - Cookie: $_COOKIE
  - Session: session_start(), $_SESSION
  - File: fopen(), fread(), fclose(), dll
  - Database: connect, select_db, query, fetch

Output

- Output alternative:
  - HTML: echo
  - Image: imagejpeg(), imagegif(), imagepng()
  - File: fopen(), fwrite(), fclose()
  - Cookie: setcookie()
  - Session: session_start(), $_SESSION
  - Database: connect, select_db, query
  - JSON or XML data (for APIs)

- Used to indicate which link is clicked by the user
- Each link represents data/action

HTML

```
<html>
1. Jaket <a href='go.php?id=1&act=edit'>Edit</a>
         <a href='go.php?id=1&act=delete'>Delete</a><br>
2. Sepatu <a href='go.php?id=2&act=edit'>Edit</a>
          <a href='go.php?id=2&act=delete'>Delete</a>
</html>
```

Browser

1. Jaket Edit Delete
2. Sepatu Edit Delete

PHP: go.php

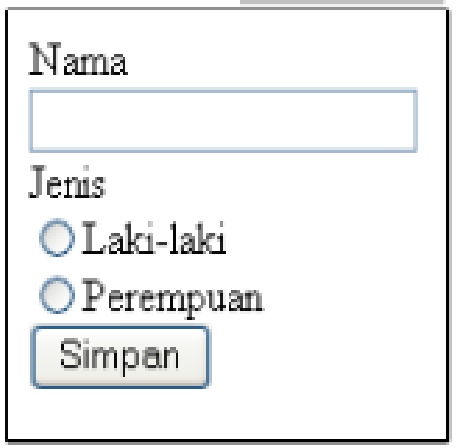
```
<?
$aksi = $_GET["act"];
$id = $_GET["id"];
if ($aksi == "edit") {
    //lakukan edit terhadap data dengan ID = $id
}
else if ($aksi == "Delete") {
    //lakukan delete terhadap data dengan ID = $id
}
?>
```

Input from HTML form

HTML

```
<html>
 <form action='save.php' method='POST'>
   Nama<br>
     <input type='text' name='nama'><br>
   Jenis<br>
     <input type='radio' name='jenis' value='L'>Laki-laki<br>
     <input type='radio' name='jenis' value='P'>Perempuan<br>
   <input type='submit' value='Simpan'>
 </form>
</html>
```

Browser



PHP: save.php

```
<?
$nama = $_POST["nama"];  //berisi string nama
$jenis = $_POST["jenis"];  //berisi "L" atau "P"

//simpan data $nama dan $jenis

?>
```

Input from Cookie

- Dapat digunakan untuk mendapatkan data yang dimasukkan oleh user pada halaman sebelumnya

HTML: login.html

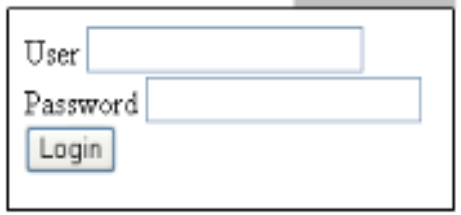
```
<html>
 <form action='login.php' method='POST'>
   User <input type='text' name='user'><br>
   Password <input type='text' name='pass'><br>
   <input type='submit' value='Login'>
 </form>
</html>
```

PHP: login.php

```
<?
$user = $_POST["user"];  //berisi string username
$pass = $_POST["pass"];  //berisi string password

if (UserDanPasswordOK($user, $pass)) {
    //simpan $user di cookie
    setcookie("login", $user);
}
?>
```

Browser



PHP: anypage.php

```
<?
$user = $_COOKIE["login"];  //berisi string username
if ($user == "") {    //belum melakukan login
    header("Location: login.html");  //redirect ke halaman login
}
else {
   // User sudah login, boleh melakukan sesuatu
}
?>
```

Input from Session

HTML: login.html

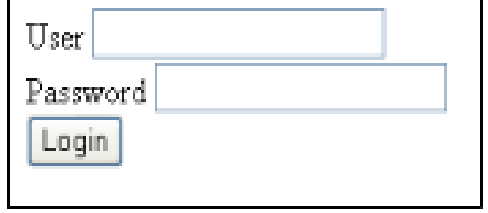
```
<html>
 <form action='login.php' method='POST'>
   User <input type='text' name='user'><br>
   Password <input type='text' name='pass'><br>
   <input type='submit' value='Login'>
 </form>
</html>
```

PHP: login.php

```
<?
$user = $_POST["user"];  //berisi string username
$pass = $_POST["pass"];  //berisi string password

if (UserDanPasswordOK($user, $pass)) {
    //simpan $user di session
    session_start();
    $_SESSION["login"] = $user;
}
?>
```

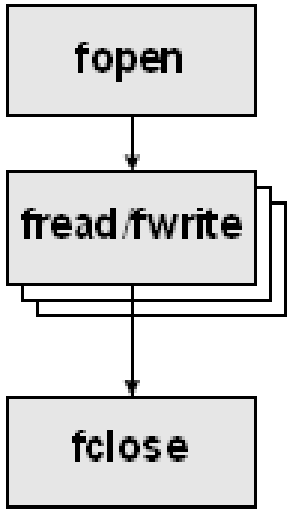
Browser



PHP: anypage.php

```
<?
session_start();
$user = $_SESSION["login"];  //berisi string username
if ($user == "") {   //belum melakukan login
    header("Location: login.html");  //redirect ke halaman login
}
else {
   // User sudah login, boleh melakukan sesuatu
}
?>
```

File Access



PHP

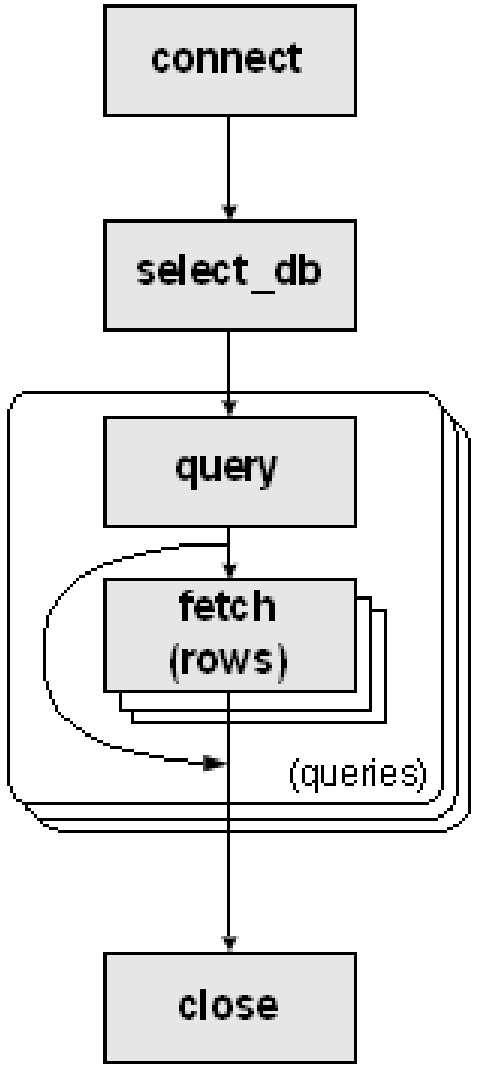
```php
<?
$namafile = "log.txt";

//Contoh menulis ke file
$fw = fopen($namafile, "w"); //buka untuk ditulisi
fwrite($fw, "2006-02-12 User Budi melakukan login\n");
fwrite($fw, "2006-02-15 User Toni melakukan login\n");
fwrite($fw, "2006-02-17 User Budi menambah data\n");
fclose($fw);


//Contoh membaca file
$fr = fopen($namafile, "r"); //buka untuk dibaca
while ($line = fread($fr)) {
    echo $line;
}
fclose($fr);


//Contoh membaca isi file dan memasukkan isinya ke sebuah variabel
$isifile = file_get_contents($namafile);
?>
```

Database Access



PHP

```php
$server = "167.205.1.2"; //database server
$userid = "tedi";
$password = "asdf";
$basisdata = "mhs";
$link = mysql_connect($server, $userid, $password);

mysql_select_db($basisdata, $link);

//contoh menyimpan data
$query = "insert into t_mahasiswa values('135', 'Budi', 'L')";
mysql_query($query);

//contoh membaca data
$query = "select nim, nama, jenis from t_mahasiswa";
$result = mysql_query($query, $link);
while ($row = mysql_fetch_array($result)) {
    echo $row["nama"]."<br>";
}

mysql_close($link);
```

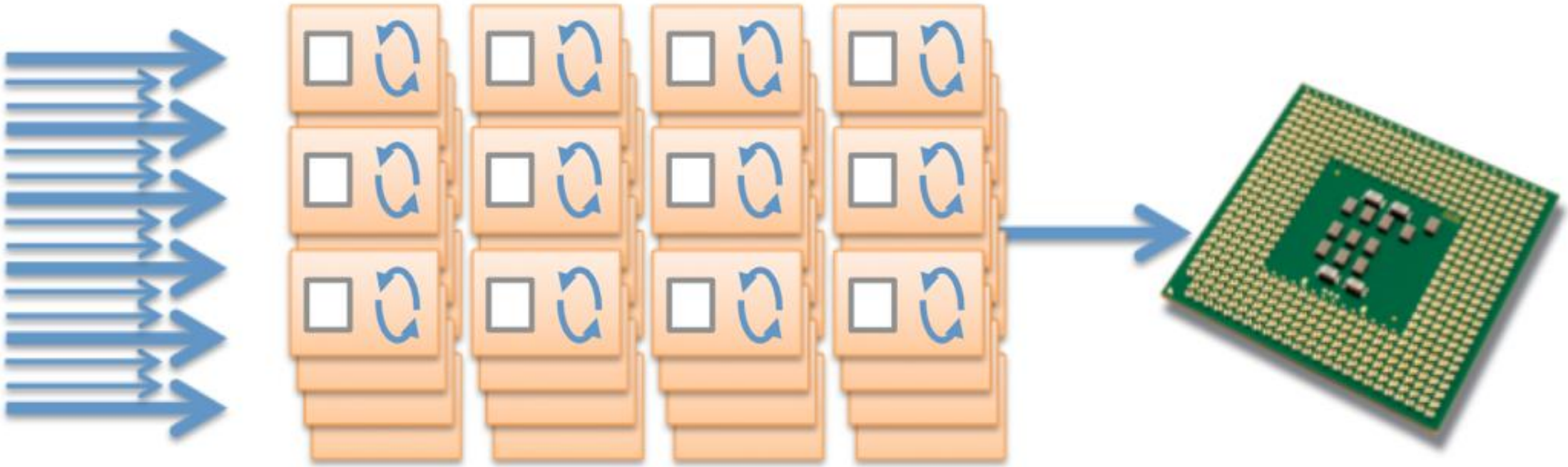
Image

PHP

```php
<?
//HTTP header yang menyatakan bahwa output mempunyai format GIF
header("Content-type: image/gif");
//membuat image baru di memory
$im = imagecreate(100, 100);  //width,height
//definisi warna pertama untuk background
$backgroundcolor = imagecolorallocate($im, 255, 0, 255); //purple
//contoh definisi warna lainnya sesuai kebutuhan
$redcolor = imagecolorallocate($im, 255, 0, 0);        //red
$greencolor = imagecolorallocate($im, 0, 200, 0);      //green
$bluecolor = imagecolorallocate($im, 0, 0, 255);       //blue
$yellowcolor = imagecolorallocate($im, 255, 255, 0);   //yellow
//contoh menggambar persegi panjang
imagefilledrectangle($im, 5, 5, 80, 25, $redcolor);    //x1,y1,x2,y2,color
//contoh menggambar lingkaran
imagefilledellipse($im, 60, 40, 50, 50, $greencolor); //xcenter,ycenter,width,height
//contoh menggambar poligon
$points = array(30,10,60,60,30,50,10,70);              //x1,y1,x2,y2,x3,y3,x4,y4
imagefilledpolygon($im, $points, 4, $bluecolor);       //arraypoints,numpoints,color
//contoh menggambar teks
imagestring($im, 5, 8, 8, "Contoh", $yellowcolor);     //fontsize,x,y,color
//outputkan ke browser
imagegif($im);
//hapus dari memory
imagedestroy($im);
?>
```

Browser

PHP Execution Model: Apache HTTPD



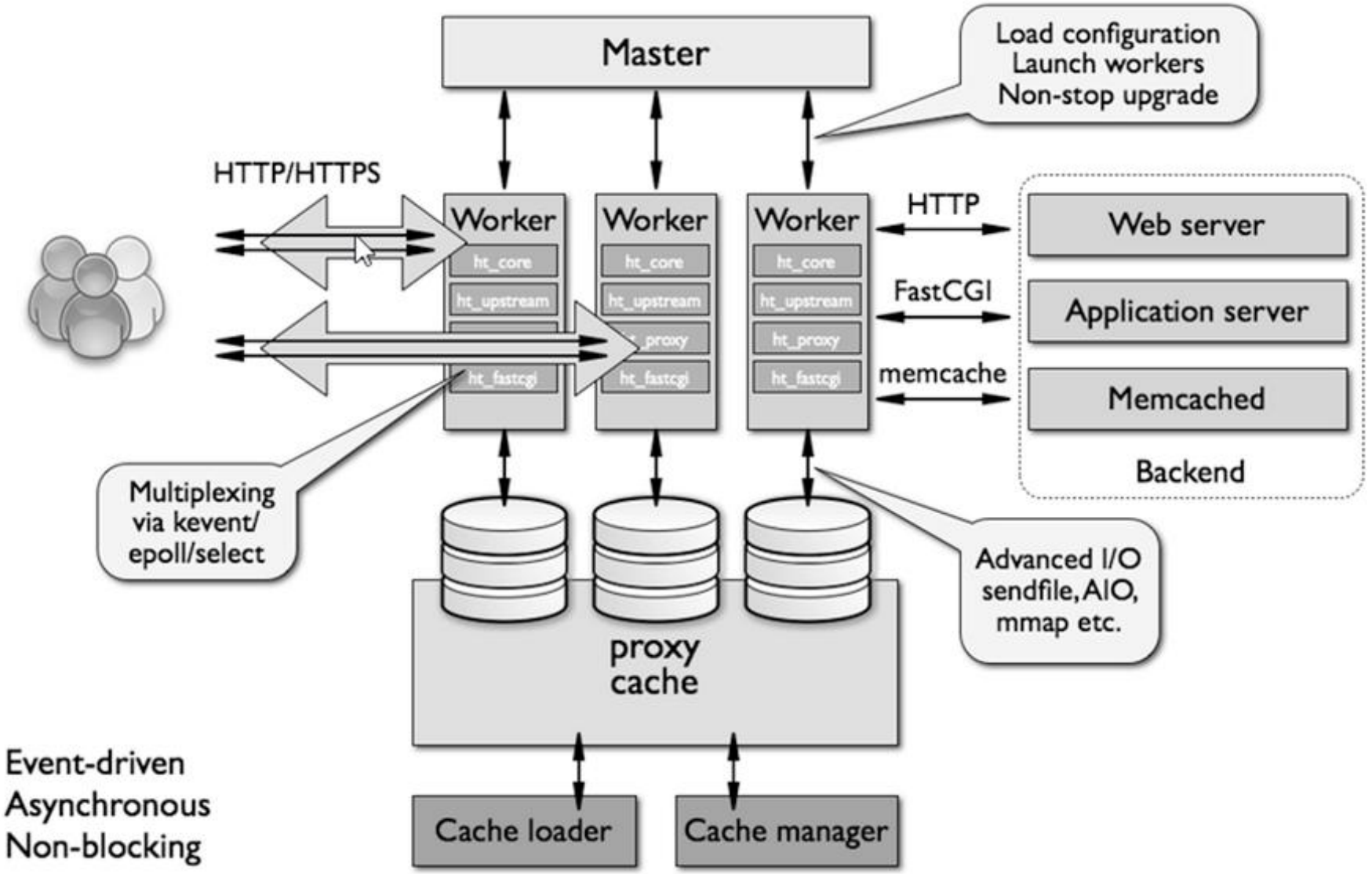
Hundreds of concurrent connections...

require hundreds of heavyweight threads or processes...

competing for limited CPU and memory

- Process-driven Approach
- Create a new thread for each request

PHP Execution Model: NGINX



- Event-driven
- Asynchronous
- Non-blocking

Additional Stuffs in PHP 7

- Speed: benchmarks for PHP 7 consistently show speeds twice as fast as PHP 5.6, and even faster
- Optional strict typing
- Error/exception handling
- New operators
  - ”Spaceship” operator
  - Null coalesce operator

```
$compare = 2 <=> 1
2 < 1? return -1
2 = 1? return 0
2 > 1? return 1
```

```
$name = $firstName ??  "Guest";
```

PHP 8

PHP 8 introduces another increase in performance for PHP, continuing the progressive improvements of each minor release of PHP 7. While the performance increase is not as stark as the difference between PHP 5 and PHP 7, there is a steady increase in performance with each release.

1. **New Language Syntax**
2. **Excellent Load Bearing Capacity**
3. **Help with Null**
4. **Return Types**
5. **Asynchronous Programming Support**