Server-side Processing

Web Programming and Testing

Mario Simaremare, S.Kom., M.Sc.
Program Studi Sarjana Sistem Informasi
Institut Teknologi Del

Objectives

- The objective of this session is the following:
  - The students are able to elaborate the role of server-side processing.
  - The students get to know one instance of server-side programming language.

Outlines

1. Motivation
2. Server-side Programming Language

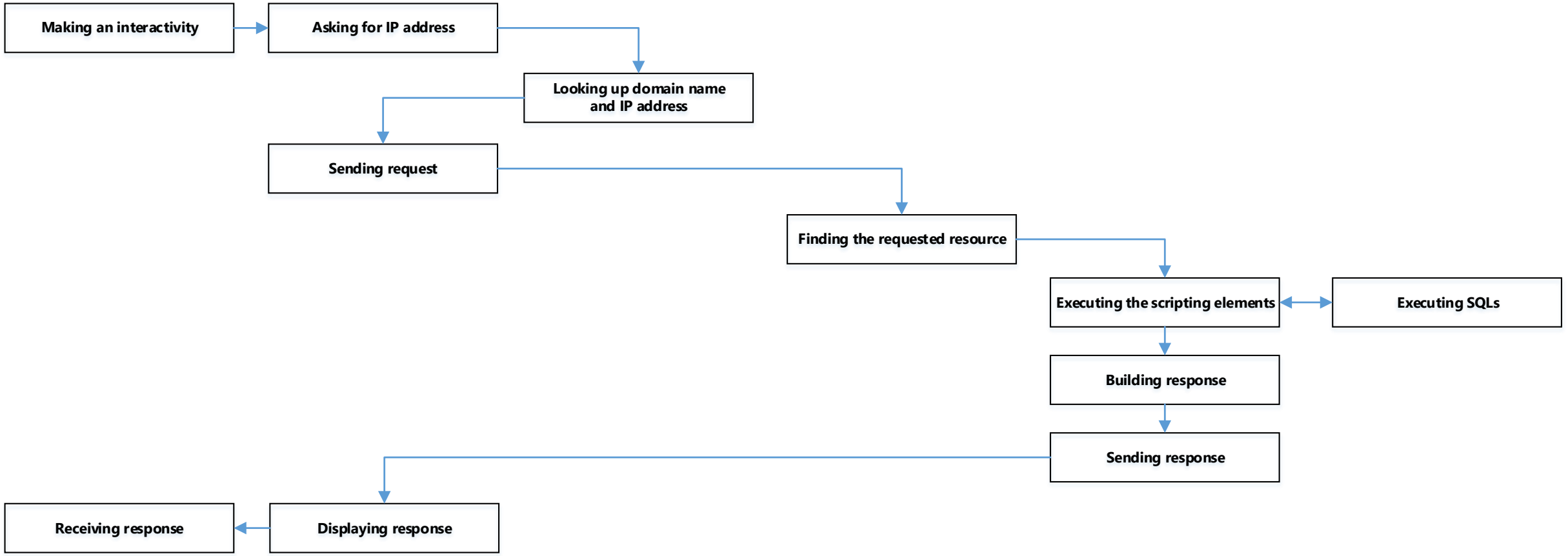
Motivation

Web 2.0: Interactivity

- Web 2.0 is minimal: dynamic content.
  - User should be able to contribute.

- Server-side processing is needed.
  - + persistence data storage.

General Workflow



Server-side Programming Language

Server-side Programming Language

- There are numerous programming languages that can work as server-side processor.

- In this course, we will use PHP.
  - You can use other language.

  Brace yourself, this course is **NOT** about PHP.

- Other platform offers a more broader ecosystem.
  - Jakarta EE, .NET, Node.JS, Phyton, Dart, etc.

PHP, A Scripting Language

- PHP Hypertext Preprocessor.
  - It is considered as relatively easy to learn.
  - Used in more than 60% web applications.

- PHP module is registered in the web server.
  - Files with `.php` extension will be preprocessed.

Please visit: https://www.php.net/

PHP, A Scripting Language

- PHP codes are embeddable.
  - It can be inserted any where.

- PHP codes are written inside the PHP tag.
  - The parser would search for the tag and execute the codes inside the tag.

```
<?php
   // codes
?>
```

Language Features

- Variables, operators, data type.
- Control structure (branching & iteration).
- Function, magic methods, namespacing, etc.
- File manipulation, accepting user input, and state management.
- Database connectivity.
- It supports both procedural and OO paradigm.
- Dependency management (Composer).

Variables

- PHP variable identifier follows these rules:
  - Started with a dollar ($) sign.
  - The first character of the identifier must be any alphabetic character or underscore.
  - The next character could be any alphanumeric character or underscore, but not white space character (tab, space, carriage-return).
  - Not case-sensitive.
  - e.g. `$name`, `$_last_name`, `$_58`

Data Types

- Four scalar types:
  - boolean, integer, float (or double), and string.
- Four compound types:
  - array, object, callable (and callback), iterable
- Two special types:
  - resource, NULL

Operators

- Arithmetic Operators
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Error Control Operators
- Execution Operators
- Incrementing/ Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Type Operators

- Operator Precedence

Array

- In PHP, array is designed to be very flexible.
  - It support indexed, associative, or mixed array.
  - It accepts multiple data types.

```
$arr = [
  “age” => 34
];
$arr[1] = 2;
$arr[“name”] = “Jaka Sembung”;
$arr[] = [];
$arr[2][] = “Nasi Padang”;
```

Control Structure: Branching

- PHP supports `if` and `switch-case` variants.

```
$x = true;
$y = ...;

if($x && $y === 0){
  // $y is a numeric with value equals to 0
} else {
 // something else
}
```

```
$x = true;
$y = ...;

switch($y){
  case 0:
    // $y is 0
  case 1:
    // $y is 1
  ...
  default:
    // other things
}
```

Control Structure: Iteration

- PHP supports `for`, `foreach`, `while` and `do-while` variants.
  - The `foreach` variant is very useful to iterate an array with key and value entry set.

```
$arr = [];
// filling $x with some values

foreach($arr as $key=>$value){
  echo("{$key}:{$value}");
}
```

Function

- Function is a callable type.
  - A function may have parameters (with type hinting).
  - Some parameters could have its default value.
    - Such parameters are specified later in the parameter list.

```
function f(string $_param1, $_param2 = []){
  // do something
}

f(“Jaka Sembung”);
```

Function

- Function is a callable type.
  - Anonymous function.

```
$g = function(...$_params){
  foreach($_params as $value){
    // do something
  }
}

$g(1, “hallo”, true, 3.14, []);
```

Function

- Function is a callable type.
  - Callback function.

```
$g = function(...$_params){
  // do something
}

function f($_callme){
  $_callme();
}

f($g);
```

File Handling

- Reading file content.
  - `file()`, `file_get_contents()`, `fopen()`, ... and many more.

- Writing string into file.
  - `file_put_contents()`, `fwrite()`, ... and many more.

Object-Orientation

- Initially, PHP was not designed to be used in OO approach.
  - As time goes, it now fully supports OO without leaving out procedural approach.
  - All OO principles are applicable.
    - Encapsulation, polymorphism, and inheritance.

Database Connectivity

- Capability to interact with database is crucial.
  - PHP supports tons of industry standard database platforms.

- Vendor Specific Database Extensions
  - CUBRID
  - DB++
  - dBase
  - filePro
  - Firebird/InterBase
  - FrontBase
  - IBM DB2 — IBM DB2, Cloudscape and Apache Derby
  - Informix
  - Ingres — Ingres DBMS, EDBC, and Enterprise Access Gateways
  - MaxDB
  - Mongo — MongoDB driver (legacy)
  - MongoDB — MongoDB driver
  - mSQL
  - Mssql — Microsoft SQL Server
  - MySQL — MySQL Drivers and Plugins
  - OCI8 — Oracle OCI8
  - Paradox — Paradox File Access
  - PostgreSQL
  - SQLite
  - SQLite3
  - SQLSRV — Microsoft SQL Server Driver for PHP
  - Sybase

Database Connectivity

- Connectivity can be achieved via two approach:
  - Native driver, plainly using the vendor's library.
  - PDO (PHP Data Object), an extra layer on top of the native driver to promote higher level of abstraction.

- Benefit of using PDO:
  - Feature rich and vendor agnostic.
  - Fully object-oriented, foundation for Object relational mapping (ORM).

Some Useful Function

- Debugging:
  - `var_dump()` – to see what's inside a variable.
  - `print_r()` – like `var_dump()` with a better formatting.
  - `die()` to stop script execution.

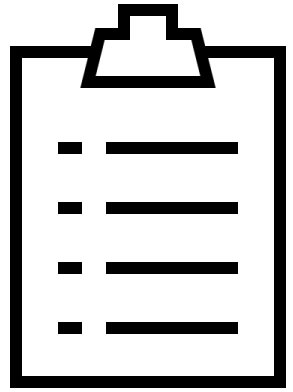
- Sending HTTP Header:
  - `header()` – to send HTTP Header to the client.
    - E.g. 'Location: http://www.example.com/index.php');

To-dos

1. Understand how the server-side processing works.

2. Dig the documentation.
   - This session is surely not enough.

3. In the near feature:
   - We will use some frameworks.



References

Srinivasan, M. (2012). Web Technology: Theory and Practice. Pearson.

Tatroe, K., et. al. (2020). Programming PHP. O'Reilly.

PHP Manual https://www.php.net/manual/en/

Thank you