goto

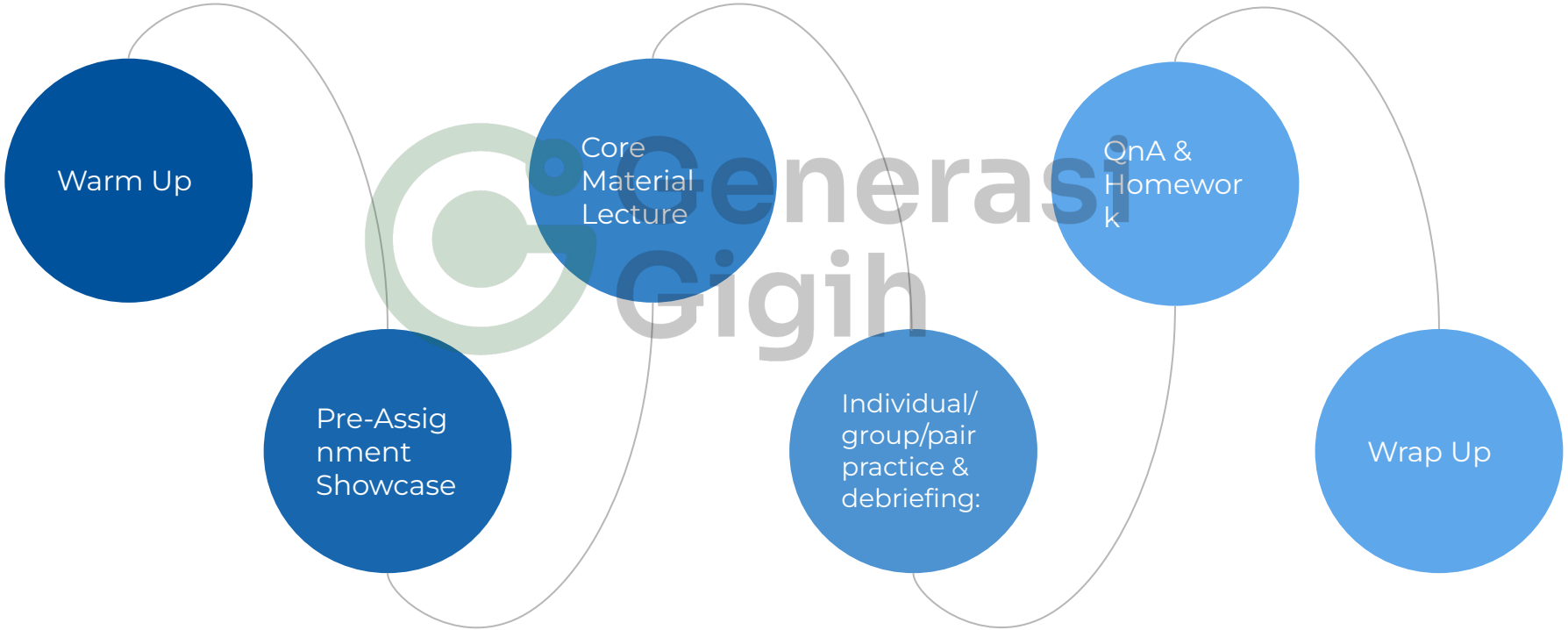
goto impact foundation

Generasi Gigih

Full Stack Engineer

Module 2.2: Backend Development with ExpressJS

Our Agenda


Warm Up
Pre-Assignment Showcase
Core Material Lecture
Individual/ group/pair practice & debriefing:
QnA & Homework
Wrap Up

Generasi Gigih

Let’s Warm Up!

Let’s Discuss

[Session outline]

Generasi Gigih

Let’s Talk About The Materials

Generasi Gigih

Back to Basic: Input-Process-Output

(In the context of Server App)

Input-Process-Output of Web Server

Input: Client requests (HTTP requests) received by the server

Process:

- The web server receives the request, parses the URL and HTTP headers, and identifies the requested resource.
- The server looks for the requested resource in its file system or database.
- If the requested resource is a dynamic content, the server executes the corresponding server-side code and generates the response.
- The server performs any necessary authentication and authorization checks.
- The server generates the response by assembling the HTTP headers and the requested resource (if any) and sends the response back to the client.

Output: HTTP response sent back to the client

* According to ChatGPT (asked on 14 Apr 2023)

Generasi Gigih

Input: HTTP Request Handling

Input Handling

In the context of Server, input is typically a **HTTP Request** generated by the client.

There are several parameters in HTTP Request:

1. Method
2. Path
3. Query Parameters
4. Headers
5. Body

HTTP Method

HTTP Method indicate **the action that the client** wants to perform. Each method has specific purpose, the 4 most important methods.

POST	Create a resource
GET	Read resource from server
PUT	Update resource in server
DELETE	Delete resource from server

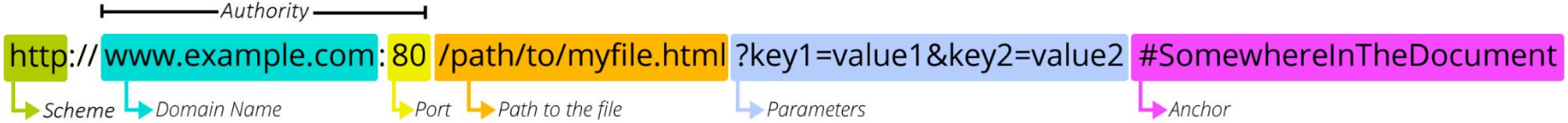
CRUD is the fundamental operation in web development especially regarding data processing.

HTTP Method handling in Express

```
app.get('/', (req, res) => {
   res.send('This is GET response');
});

app.post('/', (req, res) => {
   res.send('This is POST response');
});

app.put('/', (req, res) => {
   res.send('This is PUT response');
});

app.delete('/', (req, res) => {
   res.send('This is DELETE response');
});
```

Express able to parse and understand the HTTP Method.

Path



Path was initially used to indicate the file path of the requested document, but nowadays it can also be used for abstract resource as well.

2 Types of path:

Static	Fixed, defined in development	`/about`
Dynamic	Only the template defined in development	`/users/:email`

Path in Express

```
app.get('/about', (req, res) => {
   res.send('Return about page');
});


app.get('/users/:email', (req, res) => {
   let email = req.params.email
   res.send(`Return user detail with email:
${email}`);
});


app.get('/products/:sku', (req, res) => {
   let sku = req.params.sku
   res.send(`Return product detail with SKU:
${sku}`);
});
```

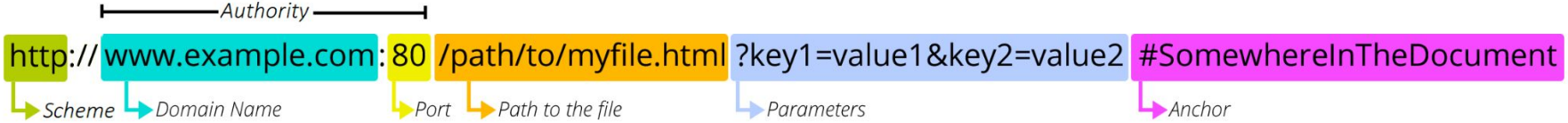
Dynamic path only define the template of a path, i.e. `/users/:email`. This makes the `req.params` to have `email` attribute.

For example:

`GET /users/john.doe@gigih.com`

Then `email` attribute will have value of `john.doe@gigih.com`

Query Param



Query Param or URL Variable is the path after ? in URL.

```
app.get('/', (req, res) => {
   let page = req.query.page
   res.send(`Return page ${page}`)
})
```

Query params is not statically defined in code. This method usually indicate that the params is **optional**.

HTTP Header

HTTP Header is part of HTTP Request that usually contains metadata, authentication information, or cookies.

Cookie is a data that set by server, stored in user's browser. Browser will send cookies back to server on each request.

```
GET /profiles HTTP/1.1
Host: localhost:3000
User-Agent: curl/7.84.0
Accept: */*
cookie: user=johndoe@gigih.com
```

HTTP Headers is a set of key-value pairs, separated by colon.

HTTP Header in Express

```
app.get('/profiles', (req, res) => {
   let cookie = req.headers.cookie
   res.send(`Return user profile for
${cookie}`)
})
```

With express, headers can be accessed in `req.headers`.

Generasi Gigih

HTTP Body

HTTP Body is the payload or content of an HTTP message, which contains the data that is being sent from the client to the server. It can contain any type of data such as text, JSON, or binary data.

HTTP Body is only available for the HTTP methods that support it, such as POST and PUT.

```
GET /profiles HTTP/1.1
Host: localhost:3000
User-Agent: curl/7.84.0
Accept: */*

{
    "name": "John Doe",
    "email": "johndoe@gigih.com"
}
```

HTTP Body can be formatted in many format. JSON is among the most popular one.

HTTP Body in Express

```
const express = require('express');
const app = express();

app.use(express.json());

app.post('/users', (req, res) => {
   let body = req.body
   res.send(`created user`)
})
```

Body can be accessed with req.body.

For JSON formatted body, a middleware needs to be used for Express to be able parse it.

Generasi Gigih

Output: HTTP Response Writing

HTTP Response

There are 3 parameters of HTTP Response that can be written:

1. Response Code:
   a. 2xx → Success (200 OK, 201 Created, etc)
   b. 3xx → Further action required (301 Moved Permanently)
   c. 4xx → User Error (400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, etc)
   d. 5xx → Server Error (500 Internal Server Error, 504 Gateway Timeout)
2. HTTP Response Header: Similar to HTTP Request
3. HTTP Response Body

HTTP Response Writing in Express

```
const express = require('express');
const app = express();

app.use(express.json());

app.get('/users', (req, res) => {
   res.send(`list of user`)
})
```

The code beside means that our server will return code 200 by default with default headers

HTTP Response Writing in Express

```
const express = require('express');
const app = express();

app.use(express.json());

app.post('/users', (req, res) => {
   res.set('Custom-Header', 'this is a
custom header')
   res.status(201)
   res.send(`created user`)
})
```

We can also explicitly set headers and status code.

REST API

API (Application Programming Interface) is a contract between two software systems, specifying how they can talk to each other and what they can do. It defines the methods, parameters, and data formats that developers can use to access the functionality or data of a particular software application, service, or platform.

REST (Representational State Transfer) API currently is still the most popular standards to be used for developing API both between frontend and backend development or from backend app to external systems.

Typically REST API use HTTP as its protocol and JSON as the data format.

Added Constraint of REST API to HTTP

Rules of REST API:

1. REST is based on the resource or noun instead of action or verb based. It means that a URI of a REST API should always end with a noun. Example: /api/users is a good example, but /api?type=users is a bad example of creating a REST API.
2. HTTP verbs are used to identify the action. Some of the HTTP verbs are – GET, PUT, POST, DELETE, GET, PATCH.
3. A web application should be organized into resources like users and then uses HTTP verbs like – GET, PUT, POST, DELETE to modify those resources. And as a developer it should be clear that what needs to be done just by looking at the endpoint and HTTP method used.`

Generasi Gigih

Q&A!

Generasi Gigih

Homework

Generasi Gigih

Simple Spotify Playlist Server

Create an HTTP Server to store your playlist

1. Add song to your playlist
   Attributes: Title, Artists (can have multiple artist), URL (from spotify URL)
2. Play song from your playlist
3. Get List of songs from your playlist

Generasi Gigih

Finally, Let’s Wrap Up!