ITSE412– Week 5

Node.JS – Web Server

Web Architecture

- Web applications are usually divided into four layers:
  - **Client layer**: consists of web browsers, or specific applications to send HTTP request to the server.
  - **Server layer**: consists of a Web Server which handle requests from the clients and send them the appropriate response.
  - **Business layer**: consists of application's business logic. This layer interacts with data layer.
  - **Data layer**: consists of databases or any source of data.

Node.js – Web Server

- A Web server is a software application which handles HTTP requests and returns Web content as a response to the clients.
- Web servers (e.g. Apache) usually deliver html pages along with images, style sheets and scripts.
- Most web servers support server side scripts using a scripting language to perform specific tasks such as getting data from database or performing complex logic etc. then return the output of the application server to client.

Web server and Path

- A Web server uses the URL, Uniform Resource Locator to find the appropriate path for the files. The file can be in the local file system or an external/internal program.
- A client makes a request using browser, URL: http://www.abc.com/login
- The request object will include many info as:

  Method : GET

  Endpoint: /login

  Protocol: HTTP/1.1

  HOST www.abc.com

Creating a web server using Node

- an HTTP server is created using the http.createServer() method.

```
const server= http.createServer(function(req, res) {   …….  } );
```

- This method takes as input a function with two parameters for request and response objects.
- In the function body we write Nodejs code to return a web content.
- The server.listen(3000) function defines the port to listen to.

Ex. 1. server.js

```
const http = require('http');
//creat http server
const server = http.createServer((req,res)=> {
    console.log(req.url);
    console.log(req.method);
    // set content type
    res.setHeader('Content-Type', 'text/plain');
            res.write('Home Page');
           res.write('You are in home page ');
           res.end();
server.listen(3000, () => {
    console.log('Server is running on port 3000.');
});
```

Ex. 1. server.js to handle routes use switch()

```
const server = http.createServer((req,res)=> {
    console.log(req.url);
    console.log(req.method);
    // set content type
    res.setHeader('Content-Type', 'text/html');
    switch(req.url){
        case '/':
            res.write('<h1>Home</h1>');
            res.write('<p> You are in home page</h1>');
            res.end();
            break;
```

continue to next page =>

Ex. 1. server.js to handle routes

```
        case '/about':
            res.write('<h1>About</h1>');
            res.write('<p> You are in About page</h1>');
            res.end();
            break;
        default:
            res.write('<h1>404 OOps!</h1>');
            res.write('<h2>Something went wrong!</h2>');
    }
});
```

Nodejs Frameworks

- Creating web applications using Node.js low level code is complex and time consuming.
- The Node community has created several Frameworks (e.g. Expres.js, Hapi, and Total.js) and provide them as NPM open source packages.
- A Framework is a package that you load to your Node application to make Web Application development easier.

Nodejs Frameworks

Node.js frameworks are mainly of three types

1- MVC frameworks: These frameworks follow the MVC design pattern to split the application development into three parts: model, view, and controller. Express.js is an MVC framework.

2- Full-Stack MVC Frameworks: these frameworks offer a range of app development capabilities such as scaffolding, libraries, template engines, etc...

Nodejs Frameworks

3- REST API frameworks: REpresentational State Transfer (REST) is a Web standards based architecture based on the HTTP Protocol. REST revolves around resources where every component is defined as a resource and a resource is accessed by a common interface (API) using HTTP standard methods.

REST uses various representations (such as JSON, XML, or plain text) to represent a resource, however JSON is the most common.

Nodejs Frameworks

REST Methods: The following four HTTP methods are commonly used in REST based architecture.

GET: is used to provide a read only access to a resource.

POST: is used to update a existing resource or create a new resource.

PUT: is used to create a new resource.

DELETE: is used to remove a resource.

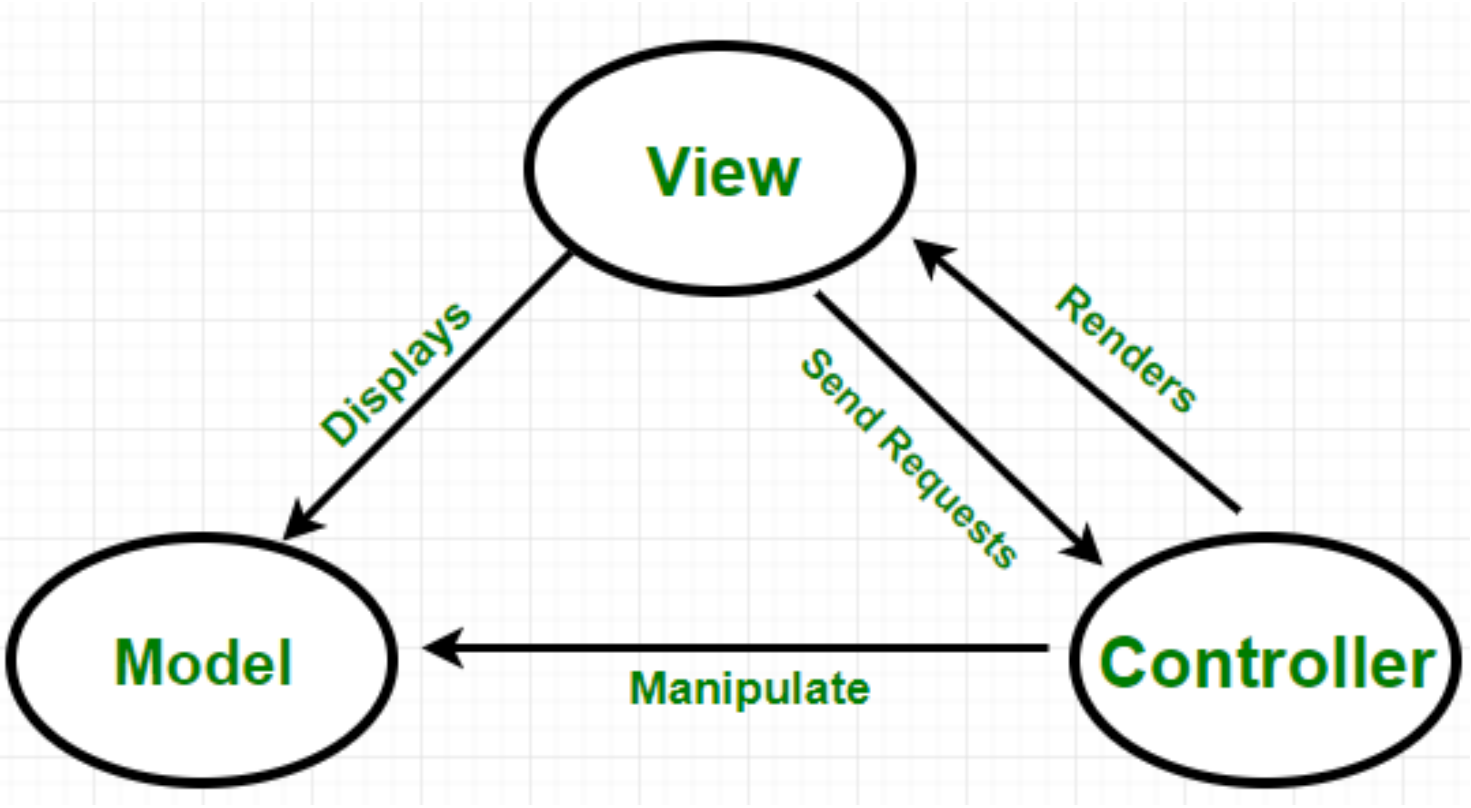
Model-View-Controller (MVC)

- **MVC** is a design pattern for software projects where application development is divided into three interconnected parts – Model, View and a Controller.
- The **MVC** model helps the developers to focus on a specific part of the application development at a time.
- **MVC** allows for efficient code reuse and speed up the development of the application.

Model-View-Controller (MVC)

- **Model**: the model represents the structure of data, and how it is stored. It maintains the data of the application.
- **View**: the view is what the application presents to the user. Views represent the user interface and send requests to the controller.
- **Controller**: controls the requests of the user. The user interacts with the View, which sends a request to the controller. The controller handles the requests and renders the appropriate view with the model data as a response.

Model-View-Control (MVC)



Model-View-Control (MVC)