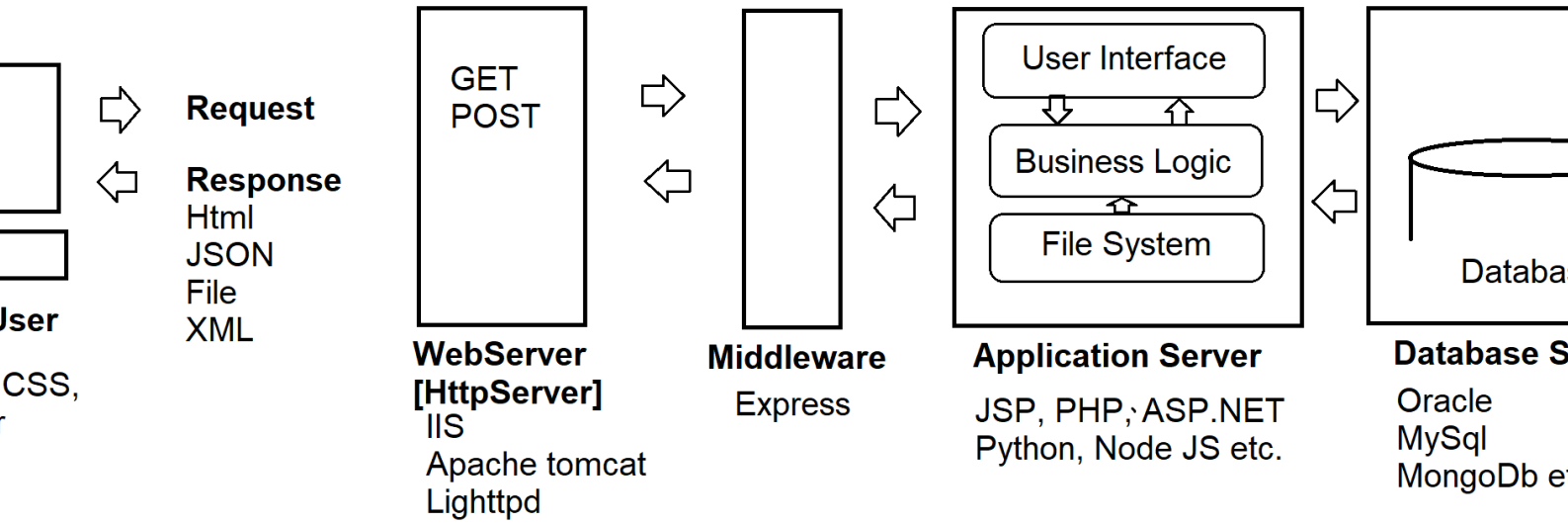
MEAN Stack - for End to End Application

Web Application Architecture



Database:

- MongoDb
  - Document Oriented
  - JavaScript Based
  - Open Source
  - Cross Platform
  - Non-Sql
  - Non-RDBMS

RDBMS	MongoDb
Database	Database
Table	Collection
Row	Document
Column/Field	Field
Join	Embedded Document

- **Setup Environment for MongoDb**
  - Download MongoDb Database Server
    https://www.mongodb.com/try/download/community
  - Start MongoDb Database Server
    - Go to “Services.msc” in window run option
    - Right Click on “MongoDb” server and Start
  - Open MongoDb Client
    - Open Command Prompt
    - Change to the following and execute the command “mongo.exe”
    C:\Program Files\MongoDB\Server\4.0\bin>mongo.exe

- **MongoDb Commands [Case Sensitive]**

Command	Description
show dbs	To View the list of all databases.
db	To View the active database.
use	To Switch to existing database or to create a new database. ➢ use new_DatabaseName
db.createCollection(“coll	Create a new Db Table

ectionName”)	Ex: db.createCollection("tblcategories") db.createCollection("tblproducts")
show collections	To View the list of Db tables
db.collectionName.insert({ }) db.collectionName.insert([{}, {}])	To insert one or many records into table (collection). Ex: db.tblcategories.insert([{CategoryId:1, CategoryName:'Electronics'},{CategoryId:2, CategoryName:'Footwear'}]) Ex: db.tblproducts.insert([{ProductId: 1, Name: 'Samsung TV', Price: 45000.55, CategoryId: 1},{ProductId:2, Name: 'Nike Casuals', Price: 4000.55, CategoryId: 2}, {ProductId: 3, Name: 'JBL Speaker', Price: 5000.55,

	CategoryId: 1}, {ProductId: 4, Name: 'Lee Cooper Boot', Price: 3000.55, CategoryId: 2},])
db.collectionName.find().pretty()	To View the records in table. Ex: db.tblcategories.find().pretty()

Server-Side Node JS

- Node JS is an open source, cross platform JavaScript Based Server-Side Scripting.
- No Buffering
- Asynchronous
- Single Threaded
- Node JS uses “JavaScript Programs” server-side [“.js”]

Node JS requires MongoDb Library to handle communication with database.

- Create a new folder in your projects “ServerSide”
- Install “Mongodb” library for your workspace.
  C:\Angular-workspace>npm install mongodb
- Create a new JavaScript file “testconnection.js” in “ServerSide” folder

Testconnection.js

```
//import MongoDb Library
var mongoClient = require("mongodb")
.MongoClient;

//Mongo Connection
var url = "mongodb://127.0.0.1:27017
";

mongoClient.connect(url, function(er
r, clientObj){
    if(!err) {
        console.log(`Connected to Mo
ngoDB`);
    } else {
        console.log(err);
    }
})
```

- Compile and Run Node JS Program by using Node Compile

  Command Prompt or Terminal:

  ➢ node testconnection.js

Ex: **To Read Records from a Database Table**

```
//import MongoDb Library
var mongoClient = require("mongodb").MongoCli
ent;

//Mongo Connection
var url = "mongodb://127.0.0.1:27017";

mongoClient.connect(url, function(err, client
Obj){
    if(!err) {
        var database = clientObj.db("apidb");
        database.collection("tblproducts").fi
nd().toArray(function(err, documents){
            if(!err){
                console.log(documents);
            } else {
                console.log(err);
            }
        })
```

```
    } else {
        console.log(err);
    }
})
```

Ex: Inserting record into Mogobd Collection

```
//import MongoDb Library
var mongoClient = require("mongodb").MongoClient;

//Mongo Connection
var url = "mongodb://127.0.0.1:27017";

mongoClient.connect(url, function(err, clientObj){
    if(!err) {
        var database = clientObj.db("apidb");
        var data = {
            ProductId: 5,
            Name: 'Earpods',
            Price: 5600.66,
            Category: 1
        };
```

```
        database.collection("tblproducts").insert(d
ata, function(err, result){
            if(!err) {
                console.log("Record Inserted");
            } else {
                console.log(err);
            }
        })
    } else {
        console.log(err);
    }
})
```

Middleware

- Install “Express” middleware in your workspace
- Middleware allows communication between client and server-side application
- You can create an API that handle communication.

Ex:
API.JS

```
var express = require("express");
var app = express();
```

```
app.get("/", function(req, res){
    res.send("Welcome to API");
})

app.listen(8080);
console.log("Server Started : http://127.0.0.1:8080
");
```

Business Layer - API

[Express – Node JS]

You need Packages

- npm install mongodb : To connect with MongoDb Database
- npm install express : To Create server-side API, which handles GET, POST etc.
- npm install body-parser : Converts the data into JSON

```
// Import Library
var mongoClient = require("mongodb").MongoClient;
var express = require("express");
var bodyParser = require("body-parser");
```

```
// Configure Connection String
var url = "mongodb://127.0.0.1:27017";

// Configure Middleware

var app = express();

// Configure Body Parser
app.use(bodyParser.urlencoded({
   extended: true
}));
app.use(bodyParser.json());

//Configure CORS [Cross Origin Resource Sharing]
app.use(function(req, res, next){
   res.header("Access-Control-All-Origin","*");
   res.header("Access-Control-Allow-Headers","Origin, X-
Requested-With, Content-Type, Accept");
   res.header("Access-Control-Allow-
Methods","GET","POST","PUT");
   next();
});
```

```
//Configure API Methods
app.get("/getproducts", function(req, res){
    mongoClient.connect(url, function(err, clientObj){
        if(!err) {
            var database = clientObj.db("apidb");
database.collection("tblproducts").find().toArray(function(err
, documents){
                if(!err){
                    res.send(documents);
                } else {
                    console.log(err);
                }
            })
        } else {
            console.log(err);
        }
    })
})
app.post("/addProducts", function(req, res){
   mongoClient.connect(url, function(err, clientObj){
      if(!err) {
```

```
        var database = clientObj.db("apidb");
        var data = {
            ProductId: req.body.ProductId,
            Name: req.body.Name,
            Price: req.body.Price,
            CategoryId: req.body.CategoryId
        };
        database.collection("tblproducts").insert(data,
function(err, result){
            if(!err){
                console.log("Record Inserted");
            } else {
                console.log(err);
            }
        })
      } else {
          console.log(err);
      }
    })
})

app.listen(8080);
```

```
console.log("Server Started: http://127.0.0.1:8080");
```

Consume in Angular

- Angular requires "**HttpClientModule**"
- It provides a set of properties and methods that are used to make an HttpRequest from client.
- It uses "**XmlHttpRequest**" object, which is use by JavaScript library to handle communication with an API.

JavaScript	**- XmlHttpRequest**
Jquery	- Ajax(), getJSON()
Angular JS	- $http
Angular	- HttpClient

Syntax:

Private http: HttpClient;

http.get()

http.post()

- **HttpErrorResponse** can track the activities performed through HttpRequest and can catch the Error Responses.
  - statusCode - 404
  - statusText - NotFound

RxJS Library

- It is Reactive Extension JavaScript Library
- It is used to configure Asynchronous calls from your client application.

Member	Description
Observable	It is used to identify the data returned on request.
Subscribe	It is used to execute any async method.
catchError	Catches the error – Handler block
throwError	Throws the error – Monitoring block
	RxJS Operators, RxJS

MEAN Example

1. BusinessLayer.js

```
// Import Library
var mongoClient = require("mongodb").MongoClient;
var express = require("express");
var bodyParser = require("body-parser");

// Configure Connection String
var url = "mongodb://127.0.0.1:27017";
```

```
// Configure Middleware

var app = express();

// Configure Body Parser
app.use(bodyParser.urlencoded({
   extended: true
}));
app.use(bodyParser.json());

//Configure CORS [Cross Origin Resource Sharing]
app.use(function(req, res, next){
   res.header("Access-Control-All-Origin","*");
   res.header("Access-Control-Allow-Headers","Origin, X-Requested-With, Content-Type, Accept");
   res.header("Access-Control-Allow-Methods","GET","POST","PUT");
   next();
});

//Configure API Methods
```

```
app.get("/getcategories", function(req, res){
    mongoClient.connect(url, function(err, clientObj){
        if(!err) {
            var database = clientObj.db("apidb");
database.collection("tblcategories").find().toArray(functio
n(err, documents){
                if(!err){
                    res.send(documents);
                } else {
                    console.log(err);
                }
            })
        } else {
            console.log(err);
        }
    })
})

app.get("/getproducts", function(req, res){
    mongoClient.connect(url, function(err, clientObj){
        if(!err) {
            var database = clientObj.db("apidb");
```

```
database.collection("tblproducts").find().toArray(function(
err, documents){
            if(!err){
               res.send(documents);
            } else {
               console.log(err);
            }
         })
      } else {
         console.log(err);
      }
   })
})
app.post("/addProducts", function(req, res){
  mongoClient.connect(url, function(err, clientObj){
     if(!err) {
       var database = clientObj.db("apidb");
       var data = {
          ProductId: req.body.ProductId,
          Name: req.body.Name,
          Price: req.body.Price,
          CategoryId: req.body.CategoryId
```

```
        };
        database.collection("tblproducts").insert(data,
function(err, result){
            if(!err){
                console.log("Record Inserted");
            } else {
                console.log(err);
            }
        })
      } else {
          console.log(err);
      }
   })
})

app.listen(8080);
console.log("Server Started: http://127.0.0.1:8080");
```

2. Create a new Angular Project

- Add new Contracts

  ICategory.cs

```
interface ICategory {
    CategoryId;
    CategoryName;
```

```
}
```

IProduct.cs

```
interface IProduct {
 ProductId;
 Name;
 Price;
 CategoryId;
}
```

- Add service

  > ng g service apidata –skipTests

Apidata.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpErrorResponse } from
'@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';

@Injectable({
  providedIn: 'root'
})
export class ApidataService {
  public getUrl = 'http://127.0.0.1:8080/getproducts';
  public postUrl = 'http://127.0.0.1:8080/addProducts';
  public getCategoriesUrl =
'http://127.0.0.1:8080/getcategories';

  constructor(private http: HttpClient) { }

  public GetCategories(): Observable<ICategory[]>{
    return this.http.get<ICategory[]>(this.getCategoriesUrl);
```

```
  }

  public GetProducts(): Observable<IProduct[]>{
    return this.http.get<IProduct[]>(this.getUrl);
  }

  public AddProduct(data){
     return this.http.post<any>(this.postUrl,
data).pipe(catchError(this.CatchError));
  }
  public CatchError(error: HttpErrorResponse){
     return throwError(error.statusText);
  }
}
```

- Add a new component
- **ProductsList.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { ApidataService } from '../apidata.service';

@Component({
  selector: 'app-productslist',
  templateUrl: './productslist.component.html',
  styleUrls: ['./productslist.component.css']
})
export class ProductslistComponent implements OnInit {

  public products = [];
  public categories = [];
  constructor(private data: ApidataService) { }

  ngOnInit(): void {
    this.data.GetProducts().subscribe(data=>this.products=data);
```

```
this.data.GetCategories().subscribe(data=>this.categories=data
);
  }
  public SubmitData(data) {
    this.data.AddProduct(data).subscribe(error=>
console.log('Something Went Wrong'));
    alert('Record Inserted');
    location.reload();
  }
}
```

ProductsList.component.ts

```
<div class="container-fluid">
 <div class="row">
  <div class="col-3">
    <h2>Register Product</h2>
    <form #frmRegister="ngForm" method="POST"
(submit)="SubmitData(frmRegister.value)">
      <dl>
        <dt>Product Id</dt>
        <dd><input name="ProductId" ngModel
#Products="ngModel" type="number" class="form-
control"></dd>
        <dt>Name</dt>
        <dd>
          <input name="Name" ngModel
#Name="ngModel" class="form-control" type="text"
required>
```

```
        <span *ngIf="Name.touched && Name.invalid"
class="text-danger">Name Required</span>
      </dd>
      <dt>Price</dt>
      <dd><input name="Price" ngModel
#Price="ngModel" class="form-control"
type="text"></dd>
      <dt>Category </dt>
      <dd>
        <select name="CategoryId" ngModel
#CategoryId="ngModel" class="form-control">
          <option *ngFor="let item of categories"
[value]="item.CategoryId">
            {{item.CategoryName}}
          </option>
        </select>
      </dd>
    </dl>
    <button [disabled]="frmRegister.invalid" class="btn
btn-primary btn-block">Register</button>
  </form>
 </div>
 <div class="col-9">
  <h2>
    Products List
    <select class="form-control">
      <option *ngFor="let item of categories">
        {{item.CategoryName}}
      </option>
```

```
    </select>
   </h2>
  <table class="table table-hover">
  <thead>
     <tr>
        <th>Product Id</th>
        <th>Name</th>
        <th>Price</th>
        <th>Category Id</th>
     </tr>
  </thead>
  <tbody>
     <tr *ngFor="let item of products">
       <td>{{item.ProductId}}</td>
       <td> {{item.Name}}</td>
       <td>{{item.Price | currency:'INR'}}</td>
       <td>{{item.CategoryId}}</td>
     </tr>
  </tbody>
</table>
  </div>
 </div>
</div>
```