



**گام نخست : آماده سازی برای توسعه کامپوننت صفحه بندی ساده با  
استفاده از Node.js و پایگاه داده MongoDB**

علی عسگری

فروردین 1401



عنوان: گام نخست ،

آماده سازی توسعه یک کامپوننت صفحه بندی ساده با استفاده از Node.js و پایگاه داده MongoDB

نگارش : علی عسگری

نام درس : معماری نرم افزار

استاد درس : دکتر مصطفی فخر احمد

## مقدمه :

در این داکيومنت قصد داریم یک برنامه صفحه بندی ساده در Node JS و Mongo DB ایجاد کنیم. برای سادگی، از چارچوب Express استفاده خواهیم کرد. این برنامه بسیار ساده خواهد بود و چارچوب ما را برای پیاده سازی تمیز تر و مناسب تر در بستر یک پروژه واقعی که در گام دوم به آن میپردازیم آماده خواهد کرد.

## پایگاه داده Mongo DB

یک پایگاه داده با نام "pagination\_nodejs\_mongodb" ایجاد کردیم و سپس باید تعدادی دیتا درون آن وارد کنیم. من از نرم افزار MongoDBCompass برای وارد کردن دیتا استفاده کردم. از آنجایی که این پایگاه داده غیر رابطه ای و مبتنی بر فایل های json است ، یک رکورد از دیتا های وارد شده به شکل زیر است.

```
{
  "_id": {
    "$oid": "619d0074e229e9235ca11f08"
  },
  "admin": true,
  "age": 25,
  "email": "ali8asgari@gmail.com",
  "password": "$2a$15$M86yHx7ISbCyL6A6nnOzOuRyPahiOpCFGD2D9wTAKyhqrV5Xt2.Ki",
  "createdAt": {
    "$date": {
      "$numberLong": "1637679164346"
    }
  },
  "updatedAt": {
    "$date": {
      "$numberLong": "1637679164346"
    }
  },
  "__v": 0,
  "name": "ehsanrezaei"
}
```

## راه اندازی پروژه

یک پوشه خالی جدید ایجاد کردم و با دستورات زیر از ترمینال وارد آن شدم :

```
mkdir pagination-nodejs-mongodb  
cd pagination-nodejs-mongodb
```

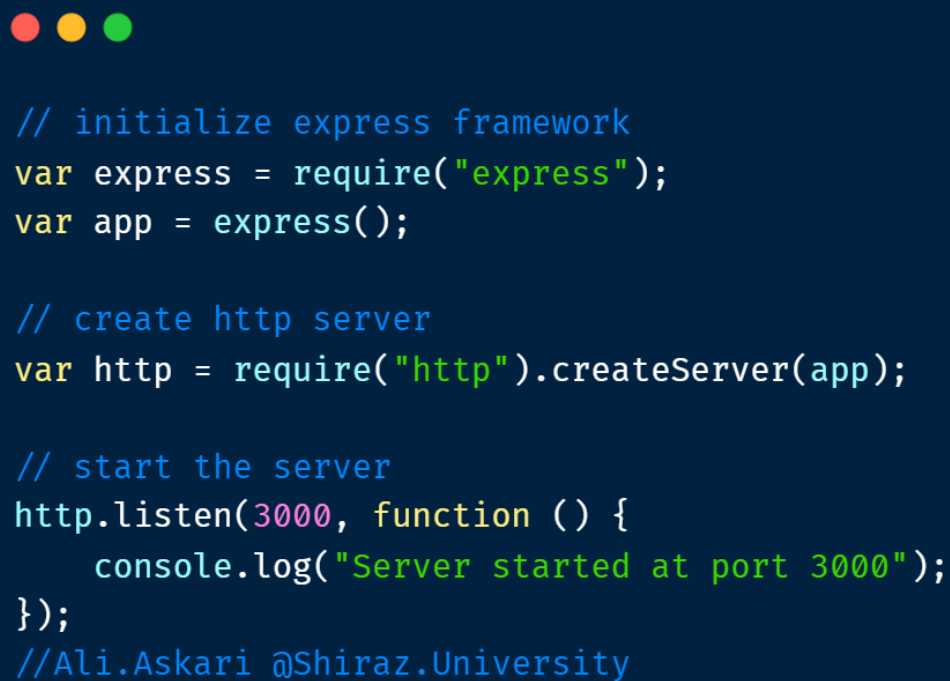
پس از ورود به پوشه، ماژول های مورد نیاز را با اجرای دستورات زیر نصب کردم :

```
npm install express http mongodb ejs  
npm install -g nodemon
```

## روند پروژه

چارچوب اکسپرس برای ایجاد برنامه های کاربردی وب و API ها استفاده می شود و ما یک API برای واکنش داده ها برای صفحه بندی ایجاد خواهیم کرد. ماژول HTTP برای راه اندازی سرور در یک پورت خاص استفاده خواهد شد. ماژول Mongo DB برای اتصال و تعامل با پایگاه داده Mongo DB استفاده خواهد شد. و در نهایت از EJS برای رندر فایل های HTML استفاده خواهد شد.

یک فایل به نام server.js ایجاد شده است کدهای زیرمربط به آن است:



```
// initialize express framework
var express = require("express");
var app = express();

// create http server
var http = require("http").createServer(app);

// start the server
http.listen(3000, function () {
    console.log("Server started at port 3000");
});
//Ali.Askari @Shiraz.University
```

تصویر 1 - کدهای فایل server.js

دستور زیر را برای راه اندازی سرور اجرا کنید.

```
nodemon server.js
```

البته گزینه دیگر استفاده از دستورات زیر میباشد.

```
Npm install
```

```
Npm start
```

با این کار سرور در پورت 3000 راه اندازی می شود. اگر ترمینال خود را باز کنید، پیام زیر را مشاهده خواهید کرد.

"Server started at port 3000"

## کامپوننت صفحه بندی

اکنون باید ماژول Mongo DB را اضافه کنیم، به خطوط زیر دقت کنید.

```
// include mongo DB module
var mongodb = require("mongodb");
var MongoClient = mongodb.MongoClient;
var ObjectId = mongodb.ObjectId;
//Ali.Askari @Shiraz.University
```

تصویر 2 - نمایی از فراخوانی ماژول مونگو دی بی

سپس می توانیم به سرور MongoDB متصل شویم. حال ما باید خطوط زیر را در داخل تابع `http.listen` (پارامتر دوم) بنویسیم:

```
// connect with mongo DB server and database
mongoClient.connect("mongodb://localhost:27017", {
  useUnifiedTopology: true
}, function (error, client) {
  var database = client.db("pagination_nodejs_mongodb");
  console.log("Database connected.");
});
//Ali.Askari @Shiraz.University
```

تصویر 3 - اتصال به مونگو دی بی و پایگاه داده

اگر اکنون ترمینال خود را بررسی کنیم ، پیام دومی را خواهیم دید که می گوید پایگاه داده شما متصل شده است. اکنون باید یک مسیر GET ایجاد کنیم که در آن مقدمات صفحه بندی ایجاد شده است :  
قصد داریم مشخصات ، نام و سنی مربوط به دو کاربر را در هر صفحه نشان دهیم ، یعنی هر صفحه دو آیتم را به نمایش میگذارد.

```

// create a GET HTTP route
app.get("/", async function (request, result) {

    // number of records you want to show per page
    var perPage = 2;

    // total number of records from database
    var total = await database.collection("users").count();

    // Calculating number of pagination links required
    var pages = Math.ceil(total / perPage);

    // get current page number
    var pageNumber = (request.query.page == null) ? 1 : request.query.page;

    // get records to skip
    var startFrom = (pageNumber - 1) * perPage;

    // get data from mongo DB using pagination
    var users = await database.collection("users").find({})
        .sort({ "id": -1 })
        .skip(startFrom)
        .limit(perPage)
        .toArray();

    // render an HTML page with number of pages, and users data
    result.render("index", {
        "pages": pages,
        "users": users
    });
});
//Ali.Askari @Shiraz.University

```

تصویر 4 - منطق صفحه بندی

کد های بالا در تصویر 4 ، نمایان گر منطق صفحه بندی ما هستند و کامنت ها توضیحات هر بخش را به وضوح بیان کرده اند. برای نمایش فایل HTML، باید به فریم ورک Express بگوییم که از engine یا موتور EJS استفاده خواهیم کرد. قبل از تابع http.listen خط کد زیر را قرار میدهیم :



```
// set the view engine as EJS for displaying HTML files
app.set("view engine", "ejs");
//Ali.Askari @Shiraz.University
```

تصویر 5 - هندل کردن EJS

پوشه ای به نام `views` در پروژه موجود است. در این پوشه `views`، فایل `index.ejs` به نام `index.ejs` ایجاد وجود دارد که محتوای فایل `"index.ejs"` در تصویر 7 خواهد بود، این فایل، بخش ویو و نمای صفحه مد نظر ما را هندل میکند، همچنین تصویر شماره 6 خروجی را نمایش میدهد. دقت نمایید که با کلیک کردن بر روی اعداد 1 و 2 و 3، دو رکورد از دیتابیس و جدول یوزر بیرون کشیده شده و نام و سنشان نمایش داده میشود. تنها 6 رکورد در دیتابیس درج شده است بنابراین تنها 3 صفحه خواهیم داشت.

| Name Age       |    |
|----------------|----|
| alimazandarani | 23 |
| ehsanrezaei    | 25 |

- [1](#)
- [2](#)
- [3](#)

تصویر 6 - خروجی

```
<table>
  <tr>
    <th>Name</th>
    <th>Age</th>
  </tr>

  <tbody>
    <% for (var a = 0; a < users.length; a++) { %>
      <tr>
        <td><%= users[a].name %></td>
        <td><%= users[a].age %></td>
      </tr>
    <% } %>
  </tbody>
</table>

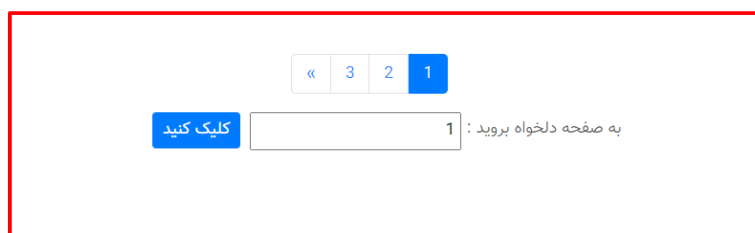
<ul class="pagination">
  <% for (var a = 1; a ≤ pages; a++) { %>
    <li>
      <a href="?page=<%= a; %>">
        <%= a; %>
      </a>
    </li>
  <% } %>
</ul>
//Ali.Askari @ShirazUniversity
```

این داکيومنت و مینی پروژه ، مقدمات کار را برای ما آماده کردند ، در گام دوم همچنان به استفاده از node.js و mongoDB پایبند خواهیم بود اما در بستر یک وبسایت و پروژه واقعی کامپوننتمان را پیاده سازی میکنیم و از همان کامپوننت درون پروژه استفاده میکنیم.

در گام بعدی به علت پیچیده تر شدن ابعاد پروژه از Mongoose هم کمک میگیریم اما دقت نمایید که Mongoose یک ORM نیست و بود و نبودش در کامپوننت ما تاثیری ندارد .

به طور دقیق تر منظور من از پیچیده تر شدن ابعاد پروژه این است که در گام دوم ، صفحه اصلی داریم ، صفحه مربوط به جست و جو خواهیم داشت و حتی پنل کاربری ایجاد خواهیم کرد و کامپوننت مان را درون آن پروژه مورد استفاده و تست قرار خواهیم داد.

تصویر 8 هدف نهایی ما در رابطه با کامپوننت برای گام دوم خواهد بود.



وبسایت برای تست کامپوننت صفحه بندی میباشد

2022

تصویر 8 - کامپوننت پیاده سازی شده در گام دوم

با احترام و تشکر به خاطر وقتی که برای مطالعه گذاشتید  
همچنین از کارکرد سیستم ویدیوی کوتاهی در فایل پروژه قرار گرفته است  
علی عسگری - فروردین 1401