**NodeJS – Controllers**

מהו controller:  
controller הוא קובץ המכיל מספר פונקציות שבעזרתן נוכל לבצע מניפולציות ב DB בקריאות שונות לשרת. לדוגמא: יצירת פריט / עדכון פריט / מחיקת פריט. זהו החלק בו אנו מחליטים מה לעשות עם הקריאה שהגיעה מהclient.

איך בונים controller?

בואו נבנה api שמציג מקומות תיירותיים.

בדוגמא הזו client יבקש לקבל מקומות לפי id

1. ראשית, על מנת שלא נצטרך לעצור באמצע נבנה כבר עכשיו פונקציה לניהול שגיאות. Error מכיל בתוכו שני מרכיבים עיקריים – errorCode ו - message.
2. בואו ניצור תיקייה בשם models ובתוכה ניצור קובץ HttpError.ts
3. בתוך קובץ השגיאות שלנו נפתח interface ונגדיר שHttpError extends Error

ובתוך הinterface נרצה להגיד שנקבל code לerrorCode מסוג number.

export interface HttpError extends Error {

  code: number;

}

1. לאחר מכן נפתח פונקציה שנקראת HttpError והיא תקבל message, errorCode

ובתוכה נפתח constant בשם error שמטרתו לזרוק error חדש כאשר יהיה צורך.

ולאחריו נגדיר error.code = errorCode ונחזיר error. בסוף הקובץ נעשה export default וזה יראה כך:

export interface HttpError extends Error {

  code: number;

}

function HttpError(message: string, errorCode: number): HttpError {

  const error = new Error(message) as HttpError;

  error.code = errorCode;

  return error;

}

export default HttpError;

1. עכשיו כשזה מאחורינו, בתוך תקיית src נפתח תקייה בשם controllers ובתוכה נפתח קובץ places\_controller.ts
2. ניצור דאטה לצורך תרגול – בניתי קובץ Json עם מקומות תיירותיים בפרופיל GitHub שלי בכתובת מקוצרת - <https://rb.gy/0cfzu> הכנסו ועשו העתק הדבק לתוך קובץ Json חדש בשם dummyPlaces.json או הורידו את הקובץ ישירות מהGitHub.
3. בתוך הקובץ places\_controller.ts נייבא את הtypes שאנו צריכים לקריאות api (Request, Response, NextFunction) ואת המידע שנרצה שהשתמש בו וזה יראה כך:

import DUMMY\_PLACES from "../dummyPlaces.json";

import { NextFunction, Request, Response } from "express";

1. כעת עלינו ליצור instance לdata שלנו נפתח constant בשם places ועליו נעבוד.

let places = DUMMY\_PLACES

1. ניצור פונקציה שמקבלת מיקום לפי id נקרא לה getPlaceById.

נפתח משתנה constant ונמשוך מהURL את הid שביקש הלקוח להציג לו.

נוכל לגשת למידע ע"י כניסה לrequest.params.pid לדוגמא: בדוגמא הזו הid של המקום הוא p1

  const placeId = req.params.pid; // { pid: 'p1' }

1. השלב הבא הוא למצוא את המיקום הספציפי שביקש הclient ע"י פונקציית find בתוך places לפי id זה יראה כך:

  const place = places.find((p) => { return p.id === placeId });

1. עכשיו, עלינו טפל בשגיאות בעזרת הפונקציה שפתחנו בהתחלה. נעשה לה import כך:

import HttpError from "../models/Http\_Error";

1. ונפתח תנאי – אם אין place זרוק error ובתוך הerror נכתוב את הerror שנרצה להציג. לדוגמא:

  if (!place) {

    throw HttpError("Could not find a place for the provided id.", 404);

  }

1. עכשיו, אחרי שהתנאי עבר ולא נזרק error הדבר האחרון שנרצה לעשות זה להחזיר את התשובה מהשרת לclient ע"י כניסה לresponse.json({place}).

Place זה קיצור של place: place.

1. זוהי הפונקציה הראשונה שלנו והיא נראית ככה:

export const getPlaceById = (req: Request, res: Response, next:NextFunction) => {

  const placeId = req.params.pid; // { pid: 'p1' }

  const place = places.find((p) => { return p.id === placeId });

  if (!place) {

    throw HttpError("Could not find a place for the provided id.", 404);

  }

  res.json({place}); // => { place } => { place: place }

};

**זהו – בנינו את הcontroller הראשון שלנו!**