קורס Web Services- תרגיל 1 להגשה: שלב ראשון, צד השרת

חובת הגשה. הגשה מיד אחרי פסח ללא DB. אפשרות לשיפור ציון לאחר מכן, להוספת DB.

בשלב השני סטודנט X יקבל לתכנת את צד הלקוח עבור WS של סטודנט Y.

ציון:

1. 81-100 [אם ירוץ תקין אצלי על המחשב + הדאטה יגיע מה- DB]
2. 60-80 [אם ירוץ תקין אצלי על המחשב. דאטה מקובץ json סטטי על השרת]
3. נכשל [אם יהיו הודעות שגיאה ולא יהיה פלט נכון]

Deployment:

1. **GitHub**:
   - מעלים ל- GitHub את הקוד לבדיקתי.
   - שמים **במודל** את הקישור.
   - באתר שלי יש מצגת נפרדת
2. **Heroku**:
   - מעלים ל- Heroku את הקוד להרצת ה- WS מהענן.
   - שמים **במודל** את הקישור **להרוקו** ול- **API**:
     - שם ה- WS ומטרתו
     - שמות 3 הפונקציות, תיאור הפרמטרים ששולחים אליהן, הפרמטרים שמקבלים בחזרה
     - תיאור אפשרות לדיבאגינג למתכנת שיעבוד בצד הלקוח
   - באתר שלי יש מצגת לעבודה עם הרוקו

לא יתקבלו תרגילים באיחור. נקודות למחשבה: קוד יעיל, מינימליסטי, מסודר, בעל משמעות, עם הערות בקוד

הוראות התרגיל: WS לניהול סטודנטים בשנקר

1. יש ליצור WS, שיספק לאפליקציה בצד הלקוח 3 נתונים מסוימים מתוך אובייקט JSON ששמור ב- mLab.
   - יש להגדיר את המטרה של ה- WS שלכם. לדוגמה, ExcellenceStudents או StudentsGrades
     1. פונקציית get לאובייקט ללא פרמטר, כמו getAllExcellenceStudent
     2. פונקציית get לנתון לפי פרמטר שמגיע, כמו getStudGrade לפי id
     3. פונקציית get מורכב יותר, כמו getExcellenceByYear שיחזיר את פרטי הסטודנטים עבור שנה מסוימת שיגיע כפרמטר (הצלבת נתונים והחזרת רשימת נתונים).
2. לכל פונקציה ניתן לגשת על ידי פנייה ל- route מסוים.
3. הפונקציות חייבות להיות מיוצאות ממודול
4. האובייקט המוחזר חייב להיות אובייקט JSON תקני

5. **העמוד המפעיל** את ה- WS יקרא לכל הפונקציות שבמודול.
6. המידע יישלח לדפדפן באובייקט response בצורת JSON. כדי לבדוק את עצמכם, צפו בתוצאה בדפדפן וראו אם אתם מקבלים JSON תקני, בו יוכל מתכנת אחר לעשות שימוש בשלב השני.

7. קובץ package חייב להכיל את השם שלכם, את שם ה- WS ואת כל ה- dependencies שעשיתם בהם שימוש, במידה ויש כאלו.

8. אני ממליצה להתחיל עם json סטטי בספריית data ורק כשיעבוד, תעלו את ה- json ל- mLab.

2 commits | 1 branch | 0 releases | 1 contributor

Branch: master | Books / +

Create README.md

KerenyX authored on May 20 — latest commit 1bc9d46e2c

node_modules/express	Books	3 months ago
Procfile	Books	3 months ago
README.md	Create README.md	3 months ago
books_ws.js	Books	3 months ago
index.js	Books	3 months ago
package.json	Books	3 months ago

README.md

Books

https://children-books-ws.herokuapp.com/ // the welcome page

https://children-books-ws.herokuapp.com/booksNumber // function that only returns the number of books exist

https://children-books-ws.herokuapp.com/bookByYear?year=2010 // function that get the year (like 2010) and return the name of the first book it find. // if it doesn't find anything, the name will be '0'

https://children-books-ws.herokuapp.com/booksByRating/8 // function that get a number (like 8 here) and returns the books that got rating equal or over this number

https://children-books-ws.herokuapp.com/error // route that returns an 'internal server error'

debugging: in chrome - response headers you can see the name of the executed function and 'ok'.

https://index-ws.herokuapp.com

hello folks, please type those url's in the web browser, here's an examples...

index page:

https://index-ws.herokuapp.com/

---

book by id:

https://index-ws.herokuapp.com/id/1

---

list of books:

https://index-ws.herokuapp.com/list

---

longest book from the book list:

https://index-ws.herokuapp.com/longest

---

best book by rate:

https://index-ws.herokuapp.com/rate

---

book type, example: fantazy,drama,comedy...:

https://index-ws.herokuapp.com/find?book_type=drama

---

https://ws-ex1-dudi.herokuapp.com/api

Book Store Web Service API

this web application created to give numerous ways to retrieve text books from our inventory.

the books are returned in JSON objects containing an array of text books

all the requests should be in GET method

example of JSON object with one book:

```
{
"textbooks" : [
        {
            "ISBN":"1118008189",
            "name":"HTML and CSS: Design and Build Websites",
            "author":"Jon Duckett",
            "publisher":"Wiley",
            "year":2011
        }
    ]
}
```

Obtaining all Text Books

in order to obtain every text books one needs to add the following link:

```
https://ws-ex1-dudi.herokuapp.com/textbooks
```

the result will be an JSON object containing an "textbooks" key and an array with all the text books objects.

Obtaining one Text Book by ISBN

in order to obtain a textbook bt a certain ISBN one need to add the following link:

```
https://ws-ex1-dudi.herokuapp.com/isbn/:isbn
```

where ":isbn" should be the ISBN of the book

the result will be an JSON object containing an "textbooks" key and an array with the text book with that ISBN

if the book was not found the result will be an empty array

Obtaining all text books by the same publisher

in order to obtain textbooks by a certain publisher one need to add the following link:

noamrom89.github.io/Best-Books-Sellers/

Best-books-sellers- API

View on GitHub | Download .zip | Download .tar.gz

Best Sellers Books - API specification

This is a web service for third-party who want to use the Best-Sellers-Books app.

In this app we will use Node js for the basic functionality.

First thing we will need to do is to make the server (server.js) running...

URL	Description	Example
https://best-sellers-books.herokuapp.com/	Showing all the best sellers books by Json objects	{ Name: "Ronaldo", year: "1989", BSMonth: "March" },{...},{...}..
https://best-sellers-books.herokuapp.com/name/BOOKID	Showing the requested book with the specific name of it (BOOKID)	{ Name: "Ronaldo", year: "1989", BSMonth: "March" }
https://best-sellers-books.herokuapp.com/BSForMonth/bookmonth	Showing array of books with the requested month of them/it (bookmonth)	{ Name: "Gerrard", year: "2010", BSMonth: "June" }, { Name: "Zidan", year: "2011", BSMonth: "June" }

Best-books-sellers- API is maintained by NoamRom89.

This page was generated by GitHub Pages using the Cayman theme by Jason Long.

https://sportbooks.herokuapp.com

WELCOME TO SPORT BOOKS WEB SERVICE

show all sport books:

if you want to get all the availble sportbooks click on the link bellow:
https://sportbooks.herokuapp.com/showAllSportBooks

show book by specific ID:

if you want to get a specific book you need you use the link bellow and you can change the id number in the query string in this ex
https://sportbooks.herokuapp.com/id?*id=111*

show book in range of prices:

if you want to get books in a specific price range you need you use the link bellow and you can change the from and to numbers in
https://sportbooks.herokuapp.com/betweenPrice?*from=2&to=80*

אבישי חג'בי:

https://ws-course.herokuapp.com

Shoes store

structure: id, size, company, useFor

GET getAllShoes
example: /getAllShoes

GET getShoeById
example: /getShoeById/?id=123

GET insertNewShoe
example: /insertNewShoe/?id=123&size=42&company=nike&useFor=running

GET deleteShoeById
example: /deleteShoeById/?id=123

GET findShow
example: /findShoe/?id=123&size=42&company=nike&useFor=running

GET updateShoe
example: /updateShoe/?id=123&size=42&company=nike&useFor=running

any time you see a multiple params you can ignore the params you want

Github optional folder structure:

shenkar web services course -- ping pong game

12 commits | 1 branch | 0 releases | 1 contributor

Branch: master | ws-course / +

change query syntax

avishayhajbi authored on May 22 — latest commit d453465f4d

public	ws 2	3 months ago
shoes	change query syntax	3 months ago
views	ws 2	3 months ago
.gitignore	Update .gitignore	3 months ago
README.md	Initial commit	3 months ago
app.json	insert printing to screen	3 months ago
index.js	done functionality	3 months ago
npm-debug.log	fix new line	3 months ago
package.json	ws 2	3 months ago

README.md

ws-course