# תרגיל 3

## Spring MVC

## including: controllers, models, views with thymeleaf, beans, injection, thymeleaf, Ajax, mySQL database

*(ראו גם סרטון הסבר באתר)*

*בירוק: עידכונים*

מטרת התרגיל: ‏

לבנות אתר הכולל גם צד שרת ,וגם צד לקוח ‏בעזרת הטכנולוגיות שנלמדו: SpringBoot javascript, html, css ‏ בשילוב מסד נתוניםSQL .

זהו תרגיל מבוסס על אותו רעיון של לתרגיל 4 של סמסטר א׳: שימו לב שרוב העבודה בצד שרת (בניגוד למה שנעשה בסמסטר א׳) .

האתר מציג מידע ‏על משתמשים מתוך ‏האתר github, ‏וגם שומר היסטוריה של חיפושים (במסד נתונים).

## דף כניסה לאתר יציג מסך login

המשתמש יצטרך לעבור אימות לפני שיוכל לגשת לדפים אחרים (login/password). ‏

הגדירו את פרטי ה-login בתוך הקובץ שapplication.properties של Spring.

‏נגדיר משתמש קבוע:

user: admin

password: 1234

אחרי אימות, המשתמש יועבר לאתר שיכלול דפים בהתאם למבנה שתבחרו. ניתן לממש כל פעולה בדף בנפרד, או להפך לממש את כולן בדף אחד.

## דף חיפוש

בנו טופס‏ המאפשר להקליד שם משתמש ‏ולבצע חיפוש באתר github.

שימו לב: החיפוש מול github יתבצע **בצד שרת**, לפי login ‏של המשתמש .

אין חשיבות לאותיות גדולות או קטנות.

המטרה היא להציג מידע על עוקבים ו- repositoriesשל המשתמש.

החיפוש יציג את:

1. ‏ ה-**login** של המשתמש
2. **followers**: ‏מספר העוקבים

במידה והמשתמש לא נמצא, או אם ואחת הרשימות ריקה, יש להציג הודעה אינפורמטיבית (״משתמש לא קיים״, ״ ‏ ללא עוקבים״(‏

ריענון של דף החיפוש **יאפס את הטופס**: קלט ריק, ללא תוצאות חיפוש.

את חיפוש נרצה לשמור במסד נתונים: כלומר נרשום את שם המשתמש github ,קישור github לאתר שלו, ומספר פעמים שביצענו חיפוש עליו. ההיסטוריה תשמר כל פעם שנכנס לאתר שוב , גם אם נריץ שוב את הפרויקט.

## דף היסטוריה

נציג רק 10 החיפושים הפופולריים לפי סדר יורד (הפופולרי ביותר בראש הרשימה).

‏ניהול היסטוריה:

1. ‏ ‏במידה ומדובר בחיפוש חדש, יש להוסיף אותו במסד נתונים (עם מספר חיפושים = 1)
2. אם כבר חיפשנו, אז נעלה את ״המונה״ בהתאם.
3. ‏ SQL דואג לבעיות סנכרון (כלומר אין לעטוף ב-synchronize הכנסה/עידכון במסד נתונים) אבל יתכן וכן תצטרכו לדאוג ב-critical section בתוך הקוד שלכם, זה תלוי במבנה הפתרון שתבחרו.
4. 10 החיפושים הפופולריים: יש להגביל את הרשימה (מומלץ לעשות זאת ע״י שאילתה SQL) הרשימה צריכה לשקף בדיוק את התוכן בצד השרת. אין צורך לעדכן בזמן אמת את היסטוריה בדף (כלומר אם יש שינוי אחרי תצוגת ההיסטוריה, נצטרך לטעון שוב את הדף על מנת לראות תמונה עדכנית).

‏רשימת החיפושים הפופולריים תופיע בדף ותכלול רשימה של שמות משתמשים כאשר כל שם יהיה מקושר (תגית <a href=) לאתר github שלו. לדוגמא אם חיפשנו nickwalton 10 פעמים ואת hacker3 פעמים, אנחנו נציג:

[nickwalton](https://github.com/nickwalton) : 10

[hacker](https://github.com/hacker): 3

בנוסף, שימו כפתור איפוס של ההיסטוריה, כלומר מחיקה של כל ההיסטוריה במסד נתונים (וכמובן בדף עצמו כתוצאה מהפעולה).

## מבנה האתר

ניתן לחלק את האתר לדפים שונים כגון: טופס חיפוש, תוצאות של החיפוש, רשימת החיפושים הפופולריים. בכל מקרה יש להוסיף כפתורי ניווט בין דפים, וכפתור התנתקות  (logout) מהאתר )לחזור למסך (login.

דאגו לחסום גישה לכל הדפים כל עוד המשתמש לא עבר login או אם ה-session שלו כבר פג תוקף (ניתן לבדוק מצב זה ע״י מחיקה ידנית של הקוקיז). מומלץ אז להחזיר אותו למסך login או להציג דף error שמאפשר לחזור ל- login

כמו כן, משתמש שכבר עבר אימות לא יראה את הדף login ויועבר ישירות לדף הבא.

**דרישה: יש להשתמש בAjax/JSON פעם אחת בלבד באתר** (לדוגמא אפשר ב-login, חיפוש, או היסטוריה), כלומר צריך לממש controller המחזיר JSON + לכתוב קוד javascript שקורה לו. שאר ה- controllerימומשו ע״י thymeleaf.

יש לשמור על מבנה Spring MVC וארגון קבצים כהלכה.

הבדיקה תתבצע בהנחה שקיים מסד נתונים ריק (הטבלאות ייווצרו ע״י Spring/JPA).

שם מסד הנתונים יהיה מוגדר מראש (חשוב אחרת התרגיל לא יעבוד אצל הבודק!): **ex3**

הערה כללית:

‏

* דאגו כרגיל לבדוק את כל הדפים שלכם אחרי reload/back בדפדפן
* אין להשאיר על הדף אלמנטים שלא רלוונטיים לתוכן הדף (‏לדוגמא הודעת שגיאה כבר לא ‏תקפה).
* באופן כללי כאשר בונים אתר יש לדאוג לכול קלט לא חוקי או פעולה לא צפויה של המשתמש. המטרה היא לשמור על התוכן של האתר ‏ותמיד להציג מצב עדכני והודעות אינפורמטיבית. ניתן לדוגמא לנטרל רכיבים ‏כאשר שימוש בהם לא רלוונטי.
* דאגו להגיש פרויקט שיעבוד ללא שום מניפולציות: אם לדוגמא השתמשתם בספריה JSON, רשמו זאת בקובץ pom.xml והפרויקט יביא אותה אוטומטית.
* אם ברצונכם להשתמש בספריה אחרת חוץ מאלה שראינו בשיעור, יש לבקש אישור.

# GITHUB REST API

על מנת לחלץ את המידע אתם תשתמשו במימשק (REST API) של github.   
(השורה שהייתה כאן על fetch לא רלוונטית – חילוץ נתונים מול github נעשה ע״י פתיחה רגילה של URL)

**https://api.github.com/users/**${username}  
‏לדוגמא <https://api.github.com/users/nickwalton> ‏יחזיר את ה-JSON הבא :

{

"**login**": "nickwalton",

"id": 22206272,

"node\_id": "MDQ6VXNlcjIyMjA2Mjcy",

"avatar\_url": "https://avatars1.githubusercontent.com/u/22206272?v=4",

"gravatar\_id": "",

"url": "https://api.github.com/users/nickwalton",

**"html\_url": "https://github.com/nickwalton",**

"followers\_url": "https://api.github.com/users/nickwalton/followers",

"following\_url": "https://api.github.com/users/nickwalton/following{/other\_user}",

"gists\_url": "https://api.github.com/users/nickwalton/gists{/gist\_id}",

"starred\_url": "https://api.github.com/users/nickwalton/starred{/owner}{/repo}",

"subscriptions\_url": "https://api.github.com/users/nickwalton/subscriptions",

"organizations\_url": "https://api.github.com/users/nickwalton/orgs",

"repos\_url": "https://api.github.com/users/nickwalton/repos",

"events\_url": "https://api.github.com/users/nickwalton/events{/privacy}",

"received\_events\_url": "https://api.github.com/users/nickwalton/received\_events",

"type": "User",

"site\_admin": false,

"name": "Nick Walton",

"company": null,

"blog": "",

"location": null,

"email": null,

"hireable": null,

"bio": "Robotics, Simulation, Self Driving Vehicles.",

"public\_repos": 36,

"public\_gists": 0,

"**followers**": 37,

"following": 3,

"created\_at": "2016-09-14T23:27:36Z",

"updated\_at": "2019-12-07T22:26:00Z"

}

‏ קישור github: <https://developer.github.com/v3/users/#get-a-single-user>

כפי שאתם רואים יוחזר JSON data הכולל ‏מספר רב של תכונות. ‏מומלץ מאוד לחקור  
 (chrome inspector – network tab) את התקשורת בין הדף הלבין github על מנת להבין את מבנה התשובה.

אנחנו מעוניינים ב:

1. : **login** ‏זהו השם המשתמש
2. **html\_url** : קישור לאתר של המשתמש
3. **followers**: ‏מספר העוקבים