סדנת Spring ו- REST API



<תמונת הטכנולוגיה>

לפני שמתחילים

**מטרות**

* להבין מבנה כללי של פרוייקט web, ובעיקר על מה זה backend
* להכיר ולהבין מה זה REST API
* להכיר את Spring
* להתנסות במימוש REST API באמצעות Spring

**מבנה הסדנה**

* נדבר על מבנה אפליקציית web גנרית, ואיפה נמצא השרת הסיפור
* נסקור את REST API, ואיך הוא מתקשר לשרת
* נקרא על Spring בקצרה
* נפתח פרויקט Spring ובו ניצור שרת HTTP שיברך אותנו לשלום
* נפתח על השרת REST API שייתן לנו ערך מבצעי אמיתי בצבא ההגנה מהבאגים.

**לפני שמתחילים וודאו שעשיתם את הדברים הבאים:**

כאן נרשום תנאים שצריכים להתקיים על מנת שמבצע הסדנה יוכל לבצעה, לדוגמה

* גרסאת JDK מותקנת (מומלץ – AdaptOpenJDK)
* Maven מותקן (ניתן לוודא ע"י לפתוח cmd ולהקליד mvn -v)
* סביבת הפיתוח IntelliJ IDEA מותקנת.
* ידע והכרת שפת Java, מומלץ להכיר חלק מהפיצ'רים של Java 8
* מומלץ – להכיר בקטנה ארכיטקטורת WEB ומה זה REST API (בגדול)

אז עכשיו שהכל סגור – בואו נצא לדרך!

מה זה פיתוח ב WEB?

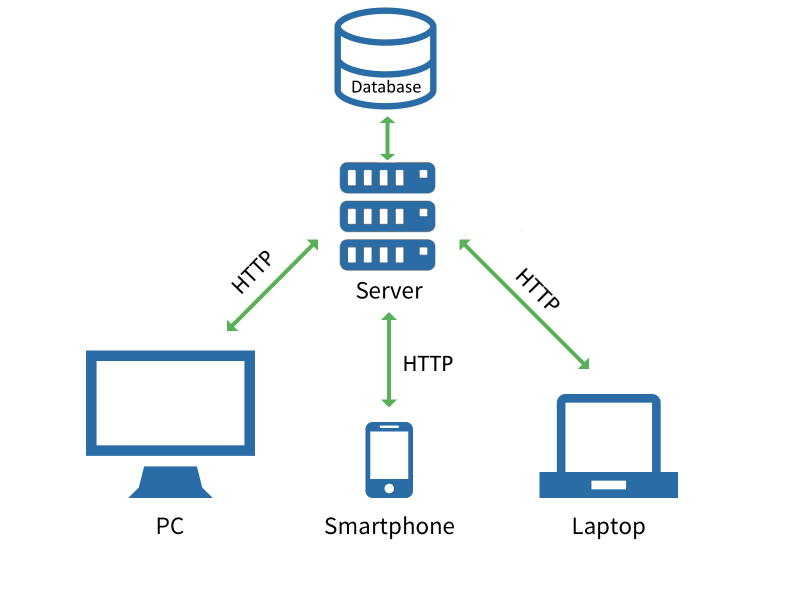
פיתוח web זה פיתוח תוכנות שירוצו על גבי הדפדפן. זה הופך אותן לנגישות מכל מחשב שיש בו דפדפן – בין אם הוא מריץ ווינדוס, לינוקס, מק או אפילו אנדרואיד. מעבר לכך – זה גם נגיש בפלאפונים שלנו.

אפליקציית WEB בסיסית מורכבת מהדברים הבאים:

* Front-end – התוכן, העיצוב, ובפועל כל התכנה שרצה אצל הלקוח – כלומר על הדפדפן.
* Back-end – כינוי למכלול כל השרתים והשירותים שאליהם ניגשת התוכנה שרצה על הדפדפן. למשל – שירות התחברות וניהול משתמשים, שרת שנותן גישה למסד נתונים, ושרתי אחסון ששומרים עבורנו את כל התמונות שאנחנו מעלים.

באפליקציה בסיסית מאוד – ה backend שלנו יהיה מורכב ממסד נתונים, בו ישמרו נתוני המשתמשים, ושרת (מחשב שמריץ קוד שכתבנו) שייחצן ממשק שייתן לנו גישה מנוהלת למסד הנתונים, מערכת התחברות, לוגיקה לכל מיני פעולות (למשל הרשאות גישה לנתונים) וכו..

אז איך בעצם זה עובד?



האפליקצייה בצד לקוח פונה דרך האינטרנט לשרת שלנו – והוא מחזיר את התשובות הרלוונטיות בהתאם לבקשה. למשל, כשאנחנו פותחים את פייסבוק, הוא קודם כל טוען את קבצי התכנה של האתר של פייסבוק, ואז מריץ אותם – והם בתורם פונים ל backend של פייסבוק, מוודאים שאנחנו מחוברים, ואם כן – פונים שוב לשרתים ומבקשים את כל התמונות של החתולים שהעלו החברים שלנו, וכמובן את הפרסומות שאנחנו כל כך אוהבים.

*\*זה תיאור לא כ"כ מדויק לגבי איך פייסבוק עובד – אבל לא ניכנס לזה עכשיו 😊*

ועכשיו לחלק המעניין - השרת

בגדול, צד השרת הוא קוד שרץ על מחשב (לרוב מרוחק), ומספק שירותים לתכנה שלנו (או לכל מישהו אחר שנגדיר) דרך הרשת.

איך בדיוק זה קורה?

בעקרון – התקשורת מבוצעת בשכבת האפליקציה, בפרוטוקול HTTP. בגדול מה שזה אומר הוא שהפניות לשרת הן בפורמט טקסטואלי – אם ננסה לתרגם בקשת HTTP גנרית לעברית, זה יהיה משהו בסגנון: "שלום, אני דפדפן פיירפוקס, בגרסה 5, מבין תווים בשפת UTF-8. אתה יכול לתת לי בבקשה את התמונה שנמצאת בתיקייה cats, בעמוד 2"?

והשרת בתגובה עונה – "מצאתי את התמונה – האורך שלה הוא 1600 פיקסלים – קח" וממשיך בשליחת התמונה, או לסרוגין "לא מצאתי את התמונה, קוד 404 (המפורסם)"

אז בעצם – HTTP הוא ה"שפה" שבה מדברים עם שרתי אינטרנט.

מה זה REST API

REST הוא ארכיטקטורת תוכנה, בעצם סט של חוקים וכללים שנלך בעקבותם כדי ליצור שירותי WEB. בגדול המטרה של פיתוח ה REST היא ליצור צורה אחידה שבה נתקשר עם אותם שרתים, בצורה כזאת שתיתן לנו יתרונות כמו ביצועים באינטרקציות בין רכיבים שונים, סקלביליות שתאפשר להרבה רכיבים כאלו לעבוד יחד מבלי שיפריעו אחד לשני, ממשק פשוט ואחיד, יכולת לעדכן את אותם רכיבים לפי הצורך (גם כאשר האפליקציה רצה), נראות התקשורת בין אותם רכיבים, אמינות וחסינות בפני כשלונות של רכיבי המערכת, חיבורים או מידע.

רגע, מה?

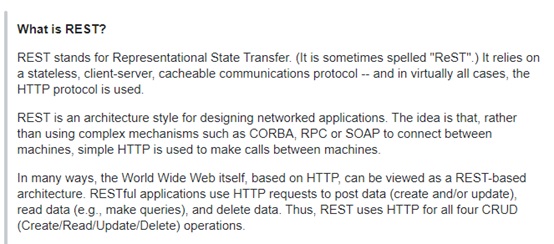
בגדול – אפשר לאכול את הראש עם הסברים כאלה וזה יהיה ממש ממש תאורטי ולא יעזור לנו – אנחנו הרי אנשים יעילים, קצינים אוטוטו, רוצים תכלס!! מה עכשיו אתה מספר לנו חיבורים רכיבים ארכיטקטורה אינטרקציות?

תכלס – צודקים.

אז מה המשמעות הפרקטית של REST עבורנו?

בגדול,כמו שאמרתי קודם – זה איזשהו אוסף של כללים שאמורים ליצור אחידות באופן שבו אנחנו מדברים עם שרתים.

אם נלך ל StackOverflow – נמצא את התשובה הזאת:



כמו שנאמר, במידה רבה נוכל לראות את האינטרנט שלנו כיום, עם פרוטוקול ה HTTP כארכיטקטורה מבוססת REST. לכן, באופן פרקטי – אם נתמקד בלהבין איך נראית תקשורת מעל HTTP ונפעל לפי העקרונות האלו אז דיי סגרנו את הפינה של REST.

אזזז – נמשיך בהסבר קצר על HTTP:

אם נקביל את השרת WEB שלנו לדואר – אז בקשת HTTP היא מעטפה שבה הפרטים רשומים מבחוץ, וגוף הבקשה (body) הוא מה שבתוך המעטפה.  
דברים חשובים שיש על גבי המעטפה זה סוג הבקשה והכתובת.  
סוגי בקשות העיקריות ב HTTP הם GET, POST, PUT, PATCH, DELETE, ולהן משמעויות סמנטיות שונות.  
לבקשה מסוג GET (כתוב על גבי המעטפה) אין מידע בגוף הבקשה (אין מכתב בתוך המעטפה) – המטרה שלה היא לבקש מידע כלשהו מכתובת מסוימת.  
בקשת PATCH אמורה "לעדכן חלק ממשהו כבר קיים", בקשת POST מוסיפה מידע חדש, בקשת PUT מוסיפה מידע אם הוא לא קיים, ובקשת DELETE מוחקת מידע.

ההסברים האלו פחות או יותר סוגרים את הרקע הנדרש ל WEB, HTTP וREST.

עכשיו מתחיל החלק המעניין והפרקטי – אנחנו נמשיך ל Spring, ונממש באמצעותו שרת HTTP.

**Common error HTTP status codes include:**

* 400 Bad Request – This means that client-side input fails validation.
* 401 Unauthorized – This means the user isn’t not authorized to access a resource. It usually returns when the user isn’t authenticated.
* 403 Forbidden – This means the user is authenticated, but it’s not allowed to access a resource.
* 404 Not Found – This indicates that a resource is not found.
* 500 Internal server error – This is a generic server error. It probably shouldn’t be thrown explicitly.
* 502 Bad Gateway – This indicates an invalid response from an upstream server.
* 503 Service Unavailable – This indicates that something unexpected happened on server side (It can be anything like server overload, some parts of the system failed, etc.).

Spring

בקצרה – זה framework לשפת java שמיועד ליצירת שרתי WEB. זאת אומרת שאם אנחנו רוצים ליצור שרת HTTP ב java, אז לא נצטרך להתעסק בחיבור ה TCP, בקריאת המידע, פירוש המידע לפי פרוטוקול ה HTTP, טיפול בתקלות פנימיות, ובתכלס מימוש כל פרוטוקול ה HTTP.

נוכל פשוט להשתמש ב Spring, שיעשה את כל הדברים האלה בשבילנו – ואנחנו נתעסק בעיקר, במה שאנחנו רוצים שהשרת יעשה בפועל.

בואו נתחיל לתרגל - PostgreSQL

לפני שמתחילים, אנא עיינו ב[קוד סטייל של טוויטר](https://github.com/twitter-archive/commons/blob/master/src/java/com/twitter/common/styleguide.md) וקראו את מסמך הסטנדרטים של המדור.

כעת קראו את [המדריך](https://www.javaguides.net/2019/01/springboot-postgresql-jpa-hibernate-crud-restful-api-tutorial.html) וממשו זאת גם בפרויקט משלכם.

תחילה ממשו את ה-controller ואת האובייקט של עובד כפי שמוצג בתרגיל.

בשלב השני, הוסיפו לאובייקט בקוד שלכם את השדה salary ושדה מסוג מערך  
שנקרא languages אשר מייצג את השפות שאותן העובד מדבר.

הוסיפו למערכת את ה-Routes הבאים:

1. מצא את כל העובדים שלא מדברים אנגלית.

2. שלוף את כל השפות המדוברות ורשום כמה מדברים כל שפה . האובייקט שמוחזר יראה כך:

[

{

“Language”: “English”,

“count”: 4

},

{

“Language”: “Arabic”,

“count”: 2s

}

]

3. שלוף את כל העובדים שמדברים לפחות 3 שפות ומרוויחים מעל 12K.

הערות

1. השתמשו ב-Java Spring Bean Validation
2. דונו עם החופף שלכם על מבנה התיקיות

מקורות מידע נוספים

* <https://spring.io/quickstart>
* <https://spring.io/guides/gs/rest-service/>
* גוגל הוא חברכם הטוב
* אה וגם stackoverflow 😉

בואו נתחיל לתרגל – מונגו (באישור רשצ)

אנא עיינו [במדריך](https://medium.com/javarevisited/building-a-rest-service-with-spring-boot-and-mongodb-part-1-2de01e4f434d) וממשו זאת גם בפרויקט משלכם.

וודאו שיש כאן את השדה year שמייצג את שנת הלימודים שנדרשתם להוסיף בפרק הקודם על מונגו.

הוסיפו למערכת את ה-Routes הבאים:

1. מצא את כל הסטודנטים שלא לומדים את קורס X

2. שלוף את כל הקורסים שנלמדים ורשום כמה לומדים כל קורס

3. שלוף את כל הסטודנטים שלומדים לפחות 3 קורסים, שהם בשנה א' ויש להם ממוצע 90+