Angular Routing

חדש

|  |
| --- |
| Routing  איך ראוטינג עובד:  מקשרים בין כפתור לנתיב על ידי הדיירקטיב routerLink.  כשהיוזר מקליק על הכפתור – הראוטר של אנגולר משנה את ה URL בדפדפן (נדמה לי שהוא עושה את זה על ידי שינוי של הפרופרטי location URL של הדפדפן) (ה URL הזה הוא כמו סיריאליזציה של state של הראוטר)  השינוי ב URL מפעיל את הראוטר.  הראוטר מסתכל בקונפיגורציה שלו ומחפש הגדרה עם path שמתאים ל URL. להגדרה הזו יש גם component.  הראוטר טוען את הטמפלייט של הקומפו' לתוך ה outlet.  את ה outlet מגדירים עם הדיירקטיב router-outlet  תיאור מחזור פעולה של הראוטר:  עבודת הראוטר מתחילה כשהוא רואה שינוי בשורת הכתובת של הדפדפן.  אז הראוטר מסתכל בקונפיגורציה שלו ומחפש הגדרה עם path שמתאים ל URL.  אם הראוטר לא מוצא path מתאים – הניווט נכשל. (לכן חשוב לשים \*\* כראוט אחרון דיפולטיבי)  הראוטר בודק ומטפל ב redirect  הראוטר בודק route guards כדי לוודא שיש הרשאה לנווט לראוט החדש  הראוטר מביא כל מידע שהראוט דורש (prefetch the data)  הראוטר מייצר אינסטנס חדש של הקומפו שמוגדרת עבור הראוט הזה (וכל קומפו בנות שלה)  הראוטר שותל את ה VIEW של הקומפו בתוך ה router outlet.  מחכה לשינוי נוסף בשורת הכתובת של הדפדפן.    היוזר יכול לנווט בתוך האפליקציה בכמה דרכים:  ללחוץ על כפתור / אופציה בתפריט וכו' - אשר מפעילים ראוט  לשנות ידנית את הכתובת בדפדפן  ללחוץ על כפתור back ו forward של הדפדפן  ניתן:  להעביר פרמטרים לראוט  להפעיל ראוט עם קוד מה - ts  להגן על ראוט עם guards |
| אנגולר מגיעה עם מודול שמבצע ראוטינג: RouterModule  הוא מספק סרביס לניהול ניווט ומניפולציות על ה URL,  קונפיגורציה לקנפג את הנתיבים,  ודיירקטיבים להפעלה והצגה של נתיבים: RouterLink, RouterLinkActive (מקשר סטייל), RouterOutlet |
| איך להוסיף ניווט לאפליקציה:  1. נוסיף את אלמנט base ל- head של הקובץ index.html.    2. נייבא את המודול של הראוטר של אנגולר אל המודול של האפליקציה.  3. נגדיר מערך של ראוטים (פירוט בהמשך)  4. נעביר את המערך כפרמטר למתודה forRoot() של ה RouterModule ב import של המודול של האפליקציה:    5. ניצור תפריט בקומפו ונגדיר בו ניווט בעזרת router-link דיירקטיב שמנווט אל הראוטים שהגדרנו במערך:  6. נגדיר בעזרת דיירקטיב router-link מיקום בדף (בקומפו הראשית) שבו נרצה להציג את התצוגות (הקומפו) שהיוזר מנווט אליהם: |
| קונפיגורציה של ראוטים  איך לקנפג ראוטים:  כל ראוט צריך:  path - הגדרת הנתיב ב URL. לא לשים סלשים / בתחילת הנתיב  comopnent - הקומפו שתוצג. זה הפניה לקומפו עצמה. זה לא סטרינג ולא שמים אותו במרכאות.  דוגמה:  אדום- נתיב products. כחול- נתיב עם פרמטר- products שאחריו מפורט ה id של הפרודקט. ורוד- נתיב לדף הבית, תכלת- נתיב פתיחה- ישמש כשהאפליקציה עולה ולא נבחר נתיב. הוא עושה redirect לנתיב של דף הבית על ידי פרופרטי redirectTo. צהוב- wildcard - כותבים \*\*. מתאים לכל נתיב שהוא. הראוטר יעצור כאן אם לא מצא עד עכשיו נתיב שמתאים (לנתיב שרשום בשורת הכתובת בדפדפן)    חשוב:  להגדיר נתיב עם path ריק- (משמש כשהאפליקציה עולה ולא נבחר נתיב) (יכול להיות שהסדר חשוב והוא צריך לבוא לפני ה \*\*  להגדיר נתיב עם path - \*\* - ווילדכרד. יתאים לכל נתיב שלא נמצאה לו התאמה.  יש חשיבות לסדר הראוטים. הראוטר יקח את הראוט הראשון שמתאים. לכן- ראוטים ספציפיים צריכים לבוא לפני ראוטים כלליים יותר  הערות:  אין סלשים בתחילת הנתיב  זה case sensitive |
| איך להגדיר ניווט דרך כפתור (או תפריט, קישור וכו')  נוסיף לכפתור אטריביוט routerLink שמקבל מערך ב property binding.  הפריט הראשון במערך צריך להיות ה path של הראוט  אפשר להוסיף פריטים נוספים למערך- כמו פרמטרים אופציונליים  איך הראוטר בונה URL מהמערך: הוא משרשר ל- URL הבסיסי של האפליקציה את ה path, משרשר לו פרמטרים אם יש, וכו'  \*ה URL הבסיסי של האפליקציה מוגדר נדמה לי באלמנט base שבקובץ index.html  הראוטר ישנה את שורת הכתובת בדפדפן לפי ה URL שהוא יצר, ואז יעשה ניווט כרגיל- יאתר את הראוט הראשון המתאים בתוך הקונפיגורציה של הראוטים, יצור אינסטנס של הקומפו' שמוגדרת עבור הראוט הזה, וישתול את ה VIEW שלה ב router-outlet    נתיב אבסולוטי / יחסי  עם / - אבסולוטי. משרשר הנתיב רק לדומיין כלומר localhost:4200 (/customers)  בלי / א עםו ./ - יחסי. משרשר את הנתיב שרשום ב routerLink לנתיב העכשווי בבראוזר (customers) או (./customers)  ../ - מסיר את הנתיב העכשווי מהבראוזר (נדמה לי) ורושם חדש לפי הערך ב routerLink  כלומר: מתוך localhost:4200/users נקליק על לינק "departments" - נקבל localhost:4200/users/departments  אבל אם מתוך localhost:4200/users נקליק על לינק "/departments" - נקבל localhost:4200/departments כי הוא אבסולוטי |
| router-outlet  דיירקטיב שמגדיר לראוטר איפה להציג את התצוגות שניווטו אליהם |
| base דיירקטיב  מוסיפים כאלמנט בקובץ index.html. חובה לראוטינג  הראוטר צריך אותו כדי לדעת איך לייצר URLs  זה מציין את הנתיב הבסיסי של האפליקציה.  "/" - האפליקציה יושבת ב root folder של האתר  "/xxx" - נדמה לי שזה אומר שהאפליקציה יושבת בפולדר xxx בתוך ה root folder של האתר  דוגמה: |
| איך להעביר פרמטרים לראוט  1. נקנפג בראוט את ה path עם: סלש, נקודותיים, שם. דוגמה:    2. באטריביוט routerLink נוסיף את הפרמטר במערך:    3. כדי לקבל את הפרמטר בקומפו:  נשתמש בסרביס ActivatedRoute של מודול ראוטר. יש להוסיף dependency לקומפו.  ניגש לפרופרטי snapshot של הסרביס, ודרכו אל המתודה paramMap.get() שמקבלת את שם הפרמטר. במקרה שלנו- זה id  \*אפשר לגשת אל הפרמטר id ב- 2 דרכים:  דרך ה snapshot - מתאים אם הפרמטר לא יכול להשתנו באותו דף. כלומר- בהכרח תיווצר אינסטנס חדש של הקומפו עם פרמטר חדש. הפרמטר ישתנה רק ברגע היצירה ואתחול של קומפו חדשה  דרך observable - מתאים אם הפרמטר יכול להשתנות בלי שעזבנו את הדף (כלומר על אותו אינסטנס של קומפו- הפרמטר יכול להשתנות). למשל- אם יש לי על הקומפו כפתור next שמציג את הפרודקט הבא בתור (ה id מתחלף) בלי שטענו מחדש את כל הקומפו. לשם כך צריך observable  דוגמה- ורוד- הופכים את ה id מסטרינג למספר (חשוב!) |
| איך לנווט מהקוד:  1. נוסיף בקומפו' dependency לסרביס Router של RouterModule    2. נקרא למתודה navigate() של ה router סרביס. היא מקבלת מערך כמו האטריביוט routerLink |
| איך לעשות redirect  ה \*\* זה wildcard שפירושו – לתפוס את הכל. חובה לשים בסוף, אחרת כל הנתיבים יעצרו עליו. |

XX

|  |
| --- |
| Guards  אם רוצים להגביל גישה לראוט (למשל בגלל הרשאות)  אם רוצים להגביל שינוי ראוט (למשל- אם היוזר הכניס מידע לטופס ועדיין לא שמר והוא מנסה לנתב החוצה מהראוט- נרצה להוסיף guard על זה ולשאול אותו אם הוא בטוח שהוא רוצה לנווט בלי לשמור את המידע?)  ה RouterModule מספק כמה guards:  CanActivate - שומר מפני ניווט אל ראוט (למשל בגלל הרשאות)  CanDeactivate - שומר מפני שינוי ראוט (למשל אם יש היוזר נוטש תהליך ורוצים לוודא שהוא לא עושה זאת בטעות)  Resolve - כשמנווטים לראוט מסוים - עושה pre-fetch לפני שהראוט הוא אקטיב  CanLoad - מונע ניווט אסינכרוני |
| דוגמה איך לממש guard  \*עושים אותו דבר עבור כל סוג של guard  נבנה guard מסוג CanActivate שבודק אם הפרמטר (id) שמועבר לראוט הוא תקין (גדול מאפס) אם כן- יאפשר ניווט לראוט. אם לא - ינווט לראוט אחר  1. נכתוב סרביס חדש המממש את האינטרפייס CanActivate (שם האינטרפייס - כשם הסוג של ה guard)  יש לו מתודה יחידה canActivate(). (שם המתודה - כשם האינטרפייס)  היא יודעת להחזיר (ורוד) boolean או observable<boolean> אם צריך לעבוד אסינכרוני  היא מקבלת 2 פרמטרים:  activatedRouteSnapshot - (כחול) מספקת מידע על הראוט הנוכחי - בזמן PT  routerStateSnapshot - (אדום) מספקת מידע על מצב הראוט ( state)  ירוק- פרמטר next של הפונקציה (מסוג ActivateedRouteSnapshot) מכיל פרופרטי url - זהו מערך שמכיל את חלקי הראוט. למשל- אם לראוט יש path שהוא customers/:id , והיוזר מנווט לראוט ככה: customers/5, נקבל ב url מערך של 2 סטרינגים - הראשון יכיל את "customers" והשני יכיל את "5". לכן ניגשים לפריט השני במערך, כי הוא זה שמחזיק את הפרמטר id של הראוט. וניגשים ל path שלו כדי לקבל את הערך "5". שים לב שממירים אותו מסטרינג למספר על ידי שימוש בקיצור של JS - שהוא פשוט הוספת + לפני הסטרינג.  תכלת- עושים בדיקה אם הערך שקיבלנו הוא מספר (עם פונקצית isNaN()), ואם המספר גדול מאפס.  אם הבדיקה נכשלת- צהוב- מנווטים לראוט אחר (וגם מחזירים false)  אם הבדיקה עברה - מחזירים true. אם מחזירים true הניווט לראוט יסתיים בהצלחה.    2. בקונפיג של הראוט - נוסיף את ה guard החדש שיצרנו לקונפיגורציה של הראוט שרוצים להגן עליו.  \*הראוט מקבל מערך של guards מסוג canActivate (או מכל סוג אחר)    3. נבדוק את הניווט. נקליד כתובת בדפדפן ונבדוק מה קורה:  הניווט הזה יצליח כי 10 הוא מספר וגם גדול מאפס:    הניווט הזה ייכשל כי saw זה לא מספר: |
|  |
|  |

XX

|  |
| --- |
| Route Resolver  משמש לטעינה של מידע דינמי מהשרת – לפני שהראוט מופעל (לפני שהקומפו' מוצגת)  משפר את חווית המשתמש (בלי ריזולבר- הקומפו תיטען (תעשה initו) תציג מידע חסר, אחכ תטען את המידע, וכשהמידע מגיע מהשרת- תעדכן את התצוגה)  איך זה עובד?   1. הראוטר בודק אם יש לראוט ריזולבר, ואם יש- מפעיל אותו לפני הראוט. 2. הריזולבר יטען את המידע מהשרת על ידי HTTP. 3. כשהמידע מגיע מהשרת- אז הראוטר מפעיל את הראוט והקומפו מופעלת כאשר היא populated לגמרי במידע שקיבלה מהריזולבר. בלי ריזולבר – הקומפו' תציג קודם כל בלי המידע (init), תטען בעצמה את המידע מהשרת, ורק כשהוא יגיע תעדכן את התצוגה ותציג את המידע השלם.   כדי להשתמש בריזולבר צריך:   1. לבנות ולרשום את הריזולבר. הריזולבר הוא סרביס 2. להוסיף לקונפיג של הראוט את הפרופרטי resolve עם הריזולבר סרביס שהגדרנו בצעד 1.   \*אפשר לקנפג הרבה ריזולבים לראוט יחיד. יש לתת לכל אחד שם יוניקי.   1. בקומפו- נקרא את המידע שקיבלנו מהריזולבר מהסרביס activatedRoute (כמו שקוראים את הפרמטרים של ראוט)   \*על ידי לפררופרטי data של הסנפשט של ה activatedRoute', ולפי השם של הריזולב (השם מהקונפיגורציה)  דוגמה בכתום: |
| דוגמה:   1. ניצור סרביס חדש (עשה EXPORT למחלקה, נקשט עם @INJECTABLE() דקורייטור, נייבא מה שצריך עם IMPORT. 2. נממש את האינטרפייס Resolve<T>. זה אינארפייס גנרי, ומגדירים לו איזה סוג נתונים הוא מחזיר   לאינטרפייס יש מתודה resolve (ורוד) שמקבלת 2 פרמטרים:  ActivatedRouteSnapshot – מכיל מידע על הראוט האקטיבי  RouterStateSnapshot – מייצג את ה state של ראוט ברגע מסוים. זהו עץ של ActivatedRoute סנפשוטים.  מחזירה Observable, או סתם מידע.  הקוד במתודה resolve מביא את המידע מהשרת ומחזיר אותו לראוט  כשהראוט מופעל- הראוטר מפעיל את פונקציית ה resolve ומחכה ל observable שיסיים לפני שהוא מפעיל את הקומפו' של הראוט.     1. נקנפג את הראוט עם הריזולבר. ניתן לקנפג הרבה ריזולברים לראוט יחיד. יש לתת שם לריזולבר, עם השם הזה ניגש למידע שהריזולבר החזיר – בקומפו'      1. בקומפו – ניגש למידע שהריזולבר החזיר (סגול) דרך הפרופרטי data של הסנפשוט של ה ActivatedRoute |
| טיפול בשגיאות:  כדאי להכניס את הטיפול בשגיאות כדי השקומפו של הראוט תוכל להימנע מזה.  דוגמה למתודת ריזולב עם טיפול בשגיאות.   1. אדום- בדיקת הפרמטר ID. במקרה של שגיאה- ה ID לא תקין- ננווט לראוט אחר בעזרת route.navigate() והאובזרבבל מחזיר NULL 2. תכלת- קוראים ל map() כדי לגשת למידע לפני שמעבירים אותו. זה מאפשר בדיקת NULL על המידע. ה map() אופרטור מחזיר אובזרבבל אז לא צריך ליצור אובזרבבל חדש. 3. ורוד- נשתמש באופרטור CATCH על האובזררבל שקיבלנו מ getProduct(). |

XX

ישן:

|  |
| --- |
| ראוטינג Routing |
| 1. נוסיף routerLinkActive דיירקטיב כדי לצבוע את האלמנט כאקטיבי. אפשר על האלמנט אבא או על ה a ב NAV BAR.   כדי להימנע מלסמן את נתיב הבית כפעיל כל הזמן (כי הוא חלק מכל נתיב אחר כרגע) נשתמש ב וrouterLinkActiveOptions ונגדיר ה exact:true . זה יעשה אקטיבי את הנתיב הספציפי הזה רק אם הוא כל הנתיב ולא חלק מנתיב אחר.      כדי לנווט רלטיבי: נוסיף פרמטר עם relativeTo. הוא לא מקבל סטרינג אלא אוביקט מסוג ActivatedRoute       1. הערה: כשמנווטים מה HTML בעזרת routerLink דיירקטיב – אם הנתיב לא קיים נקבל שגיאת ניווט (נדמה לי שרק ברלטיבי)   כשמנווטים מהקוד בעזרת navigate() – לעולם לא נקבל שגיאה על ניווט רלטיבי גם אם הוא לא תקין. אלא אם כן נשתמש ב relativeTo   1. ניתן להגדיר ריבוי פרמטרים על ראוט: (כולם יהיו זמינים ב snapshot.params אחר כך (סעיף קודם))          1. חשוב! איך לרפרש הפרמטרים בקומפו': אם רוצים לטעון את אותה קומפו' מתוך עצמה: איך להגיב לשינוי בפרמטרים (הטכניקה בסעיף הקודם לא תעבוד)(אם רהותה קומפו' מתוךינויים במידזה לא המקרה- אין טעם בטכניקה של סעיף זה)   מה קורה כשטוענים נתיב חדש: אנגולר מסתכלת בנתיבים שהגדרנו, מוצאת הנתיב הדרוש, טוענת הקומפו' שלו, מאתחלת את הקומפו', נותנת את המידע מהראוט זמין ב ActiveRoute.snapshot  אבל!!!!!!!!!!!!! אם אני כבר נמצאת על הקומפו' הזאת- אנגולר לא תאתחל שוב את הקומפו'!!!!!!!!!!!!  כדי להירשם לעדכונים של הפרמטרים אם הקומפו' כבר טעונה: ניגש לפרופרטי params של ActiveRoute. הוא observableואפשר להירשם לשינויים שבו ע"י subscribe(). סבסקרייב מקבל 3 פרמטרים, הראשון- פונקציה שתופעל במקרה של שינויים במידע הפרמטרים.       1. פרמטרים של queryString      * 1. איך להעביר:      1. בטמפלייט:       queryString params – נעביר באמצעות queryParams (פרופרטי של RouterLink דיירקטיב). מקבל אובייקט עם paramName:paramVal :      איך להוסיף # בסוף ה queryString:       * + 1. בקוד:      * 1. איך לקבל:       הערה: אם אני כבר בקומפו' והפרמטרים משתנים – אני לא אדע וזה לא יתרפרש. לשם כך דרוש :observable יש לרשום בתוך ה subscribe() את פונקצית הטיפול במידע חדש |
| תקלות אפשריות:  הפרמטר הוא מספר, אבל נחשב לסטרינג כי כל הראוט זה סטרינג. פיתרון: להפוך אותו למספר עם + : |
| Nested routes / child reoutes   1. נוסיף children routes:      1. נוסיף עוד אלמנט של אאוטלט      1. כדי לנווט רלטיבי: נוסיף פרמטר עם relativeTo. הוא לא מקבל סטרינג אלא אוביקט מסוג ActivatedRoute |
|  |
|  |
| איך להעביר מידע לראוט  מידע סטטי:  עם פרופרטי data  אני יכולה להוסיף לו איזה פרופרטיס שבא לי. במקרה זה: message.      מידע דינמי: |
|  |
| נוציא את קוד ה routing למודול נפרד:  באותה ספריה כמו app.module.ts- ניצור קובץ חדש app-routing.module.ts  נוסיף את השורה:    נכתוב בו CLASS חדש AppRoutingModule ונעשה לו אספורט.  נקשט אותו עם דקורטור @NgModule    נעשה imprt ל ראוטר מודול:     * נעשה CUT PASTE למשתנה appRoutes מסוג Routes שכתבנו בקובץ app.module.ts     יש לעשות import לכל קומפוננטה.  לא להוסיף את הקומפו' האלה לפרופרטי declerationsשל הדקורטור @NgModule כי הן כבר מופיעות ב ב declerations של המודול העיקרי- app.module.ts וזה יעיף שגיאות.  נייבא את המודול החדש שלי אל תוך app.module.ts: |

|  |
| --- |
| Route Guards (שומרים) - זה פ'יצר של ראוטינג שמאפשר לאנגולר להריץ קוד לפני שעוברים לראוט / לפני שעוזבים ראוט.  יש כל מיני Guards, למשל: CanActivate, CanActivateChild, CanDeactivate, resolver  כל guard (שומר) נדמה לי יודע לקבל סרביס (למעשה מערך של סרביסים) שאותו הוא מריץ. הסרביס צריך לממש אינטרפייס וכך- לממש מתודה מסוימת שתורץ על ידי הראוטר. מקנפגים את הסרביסים בראוט. |
| CanActivate – guard שרץ לפני שהראוט מרונדר, ויכול להחליט אם לרנדר אותו (מחזיר true) או לא (מנתב לנתיב אחר או מחזיר false ?)  מתאים לאותנטיקציה.  יוצרים סרביס שמממש את אינטרפייס CanActivate, שלו מתודה יחידה CanActivate().  מממשים את המתודה הזו. אם היא מחזירה true – אנגולר תאפשר להגיע לראוט. אם לא - ??? ניתן גם לנתב מהפונציה הזו לנתיב אחר באמצעות הראוטר.  המתודה הזו יודעת להחזיר אחד מ 3 סוגים: פרומיס או observable (אם רוצים א-סינכ(, או פשוט boolean.  אם מחזירה true – אנגולר תעביר את היוזר לראוט.  אם מחזירה false - ???  אפשר גם לנווט מתוך הפונקציה לנתיב אחר (למשל לדף לוגין)  אח"כ נקנפג הראוטים שרוצים להציב עליהם שמירה – עם פרופרטי canActivate שיודע לקבל מערך של guards אשר ירוצו לפני שהיוזר מנותב לנתיב הזה.  הקונפיגורציה של ראוט מתיישמת אוטו' לכל הראוטים בנים שלו.  ...אנגולר תבצע את הקוד הזה לפני שהיא טוענת את הראוט.  לדוגמה – נוסיף guard שמבצע בדיקה של אותנטיקציה, ולא מאפשר ליוזרים לא מורשים לראות את הנתיב.  נוסיף סרביס שמממש את אינטרפייס CanActiate: מהpackage של @angular/router. האינטרפייס הזה מכריח אותי לממש פונקציה CanActivate() שמקבלת 2 פרמטרים: ActivatedRouteSnapshot, RouterStateSnapshot, ומחזירה Observable<boolean> (יכולה גם להחזיר פרומיס או פשוט boolean למקרה שלא צריכים א-סינכרוניות (למשל אם כל הקוד רץ בקליינט לא צריך א-סינכ, אבל אם פונים לסרבר כדאי לממש א-סינכ)    נקפג את הראוטים שאני רוצה להגן עליהם. ראוט שצריך אותנטיקציה: נוסיף לו guard Can Activate שמקבל מערך של כל השומרים (guards) שאני רוצה להריץ.    זה הסרביס שבדוגמה עושה אותנטיקציה ומחזיר boolean האם היוזר עשה לוגין או לא (על ידי פרומיס): |
| CanActivateChild – דומה מאוד ל CanActivate. יש לממש את אינטרפייס CanActivateChild (אפשר באותו סרביס כמובן), ולממש את מתודה canActivateChild() שמקבלת ומחזירה אותם פרמטרים כמו CanActivate()  משתמשים אם רוצים להגן רק על הילדים של ראוט מסוים (ולא על עצמו). לא משפיע על הראוט עצמו, רק על הילדים שלו. מקנפגים בראוט עצמו.      במקרה הזה פשוט קוראים למתודה אחרת שנמצאת בסרביס- canActivate() כי היא עושה בדיוק אותו דבר. |
| CanDeactivate Guard –  נגדיר אינטרפייס שיש לו פונקציה יחידה CanComponentDeactivate() שלא מקבלת פרמטרים, ומחזירה Observable<boolean> (או פרומיס או בוליאן רגיל אם זה סינכרוני)    נכתוב את הסרביס שמממש אינטרפייס ופונקציה canDeactivate() שמקבלת קומפו' עם פונקציה canDeactivate() (כי הקומפו מממשת את האינטרפייס החדש שכתבנו)    נקנפג את הראוט:    עכשיו הקומפו' EditServerComponent – בכל פעם שהיוזר ירצה לנווט החוצה ממנה – לפני שעוזבים אותה הראוטר יפעיל את סרביס CanDeactivateGuard – את המתודה שלו CanDeactivate – שתפעיל את המתודה canDeactivate() של הקומפו.  נממש את האינטרפייס שכתבו בקומפו: |
| data – איך להעביר מידע סטטי לראוט  נקנפג בראוט:    נשתמש במידע בקומפו. יש פה 2 דוגמאות: עם סנפשוט (בהערה. למצב שהמידע לא ישתנה) או עם observable (למצב שהמידע ישתנה) |
| resolver – משמש לבצע פעולה הקודמת להצגת הנתיב. לדוגמה- אם רוצים להביא מידע דינמי מהשרת (למשל פרטים אישיים) כדי להציג בקומפו. אפשר לעשות את זה גם ב OnInit של הקומפו', אבל אם רוצים לטעון את המידע לפני שבכלל מציגים את הראוט – משתמשים בריזולבר.  Resolver הוא גנרי, צריך להגדיר לו איזה סוג מידע הוא מחזיר.  הריזולבר ירוץ בכל פעם שמנתבים לראוט הזה. הערה: אין צורך להירשם ל observable של ראוט, כי הריזולבר ירוץ בכל פעם עם המידע החדש. לכן מספיק להשתמש ב סנפשוט של הראוט.  בדוגמה: הפלוס לפני route.params משמש כדי להפוך את הערך מסטרינג למספר.    ריזולבר לא מקבל מערך של סרביסים כמו שאר ה guards, אלא אובייקט עם פרופרטיס, כדי שנוכל בקומפו לגשת אל המידע שקיבלנו מהריזולבר. איך? דרך שם הפרופרטי שקינפגנו באובייקט. בדוגמה- דרך server |
|  |
|  |