אז איך מאמתים את הטוקן בכל בקשה?

אז היה לנו את הפילטר הראשון הוא – JwtUsernameAndPasswordAuthenticationFilter

עכשיו נוסיף עוד פילטר שהוא יהיה – JwtTokenVerifier

אז ניצור את המחלקה הזאת

```
public class JwtTokenVerifier extends OncePerRequestFilter {
    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException, IOException {

    }
}
```

המחלקה תירש מהמחלקה האבסטרקטית OncePerRequestFilter והיא מצהירה על מתודה אבסטרקטית אחת שהיא doFilterInternal.

אנחנו יורשים את המחלקה כדי שהפיטלר הזה יעבור על בקשה רק פעם אחת ולא יותר

אז אנחנו לוקחים את הטוקן מתוך הheader

ואז בודקים אותו בעזרת המחלקה Strings אם הוא null או ריק או אם הוא לא מתחיל ב Bearer (אנחנו הוספנו את זה מקודם),

אז אנחנו בעצם מפלטרים את הטוקן ומסיימים את המתודה.

```
if (Strings)
Strings com.google.common.base
```

```
@Override
protected void doFilterInternal(HttpServletRequest request,
                                HttpServletResponse response,
                                FilterChain filterChain) throws ServletException, IOException {

    String authorizationHeader = request.getHeader( s: "Authorization"); // מקבלים את הטוקן
    if (Strings.isNullOrEmpty(authorizationHeader) || !authorizationHeader.startsWith("Bearer ")) { // אם הוא ריק או לא מתחיל עם Bearer אז הוא לא תקין
        filterChain.doFilter(request, response); // מפלטרים את הבקשה
        return;
    }
    String token = authorizationHeader.replace( target: "Bearer ",  replacement: ""); // מורידים את ה Bearer מהמחרוזת
    try {
        String key = "secure_secure_secure_secure_secure_secure_secure_secure_";

        Jws<Claims> claimsJws = Jwts.parser()
```

אם עברנו את הסינון הראשוני אז אנחנו מחלצים את הטוקן -> מורידים לו את ה Bearer שהוספנו לו

מביאים את המפתח שעזר לא לקודד את הטוקן ואז

מייצרים אובייקט שאיתנו אנחנו יכולים לגשת למידע של בטוקן בעזרת הטוקן עצמו והמפתח,

מוצאים ממנו את המידע שהכנסנו לתוך הטוקן, ואז נחלץ מתוך הbody את הusername והauthorities (הוא מתקבל בתור רשימה של מפה של סטרינגים)

ואז כדי להמיר את זה לסט נעשה על זה פונקציית מיפוי.

ואז נעשה אותנטיקציה למידע

אם זה נפל בדרך אז נזרוק שגיאה שהטוקן לא אמין

```
        Jws<Claims> claimsJws = Jwts.parser()
                .setSigningKey(Keys.hmacShaKeyFor(key.getBytes()))
                .parseClaimsJws(token);

        Claims body = claimsJws.getBody();

        String username = body.getSubject();
        var authorities = (List<Map<String, String>>) body.get("authorities");

        Set<SimpleGrantedAuthority> simpleGrantedAuthorities = authorities
                .stream()
                .map(m -> new SimpleGrantedAuthority(m.get("authority"))
                ).collect(Collectors.toSet());
        Authentication authentication = new UsernamePasswordAuthenticationToken(username, credentials: null, simpleGrantedAuthorities);
        SecurityContextHolder.getContext().setAuthentication(authentication);

    } catch (JwtException e) {
        throw new IllegalStateException("Token: " + token + " cannot be trusted");
    }
}
```

אחרי הכל כדי שזה ימשיך בשרשרת של הפילטרים נוסיף את השורה הבאה, ואז הכל יעבוד כמו שצריך

```
    } catch (JwtException e) {
        throw new IllegalStateException("Token
    }

    filterChain.doFilter(request,response);
```

בהגדרות שלנו נוסיף את הפילטר הזה ונגיד לו שהוא יתבצע אחרי הפילטר הראשון שיצרנו

```
.addFilter(new JwtUsernameAndPasswordAuthenticationFilter(authenticationManager()))
.addFilterAfter(new JwtTokenVerifier(),JwtUsernameAndPasswordAuthenticationFilter.class)
```