

Mengimplementasikan Web Security Pada Spring Boot Framework

Pada tutorial ini akan diimplementasikan web security yang paling sederhana, yaitu proses otentikasi dan otorisasi. Proses otentikasi akan dilakukan dengan menerapkan fungsi login dan logout, sehingga dapat dibatasi user yang masuk ke dalam sistem. Sedangkan proses otorisasi dilakukan dengan menerapkan sistem “role” pada sistem, sehingga dapat dibatasi feature mana saja yang dapat diakses oleh user tertentu.

Untuk menerapkan proses otentikasi dan otorisasi, maka pertama sekali dibuat sebuah class dengan nama `WebSecurityConfig.java`. Dibuat sebuah fungsi “configure” untuk menentukan feature/fungsi mana yang dapat diakses oleh user/role tertentu. contohnya fungsi pada request (“/”) diberikan akses kepada semua user/role begitu juga dengan (/login) dan .logout.

Kemudian pada fungsi “configure global” disimpan username dan password beserta role user yang dapat digunakan untuk mengakses sistem.

Proses berikutnya dibuatkan sebuah halaman login dengan berisi dua inputan yaitu username dan password. pada controller juga ditambahkan sebuah fungsi login. Pada praktikum ini saya menambahkan fungsi login pada controller yang telah ada sebelumnya yaitu `StudentController`. Maka ketika dijalankan tampilan halaman login adalah sebagai berikut.

Latihan

1. Kemudian penambahan fungsi logout dilakukan dengan menambahkan tombol “sign out” pada halaman **`student/view/{npm}`**. Berikut tampilannya.

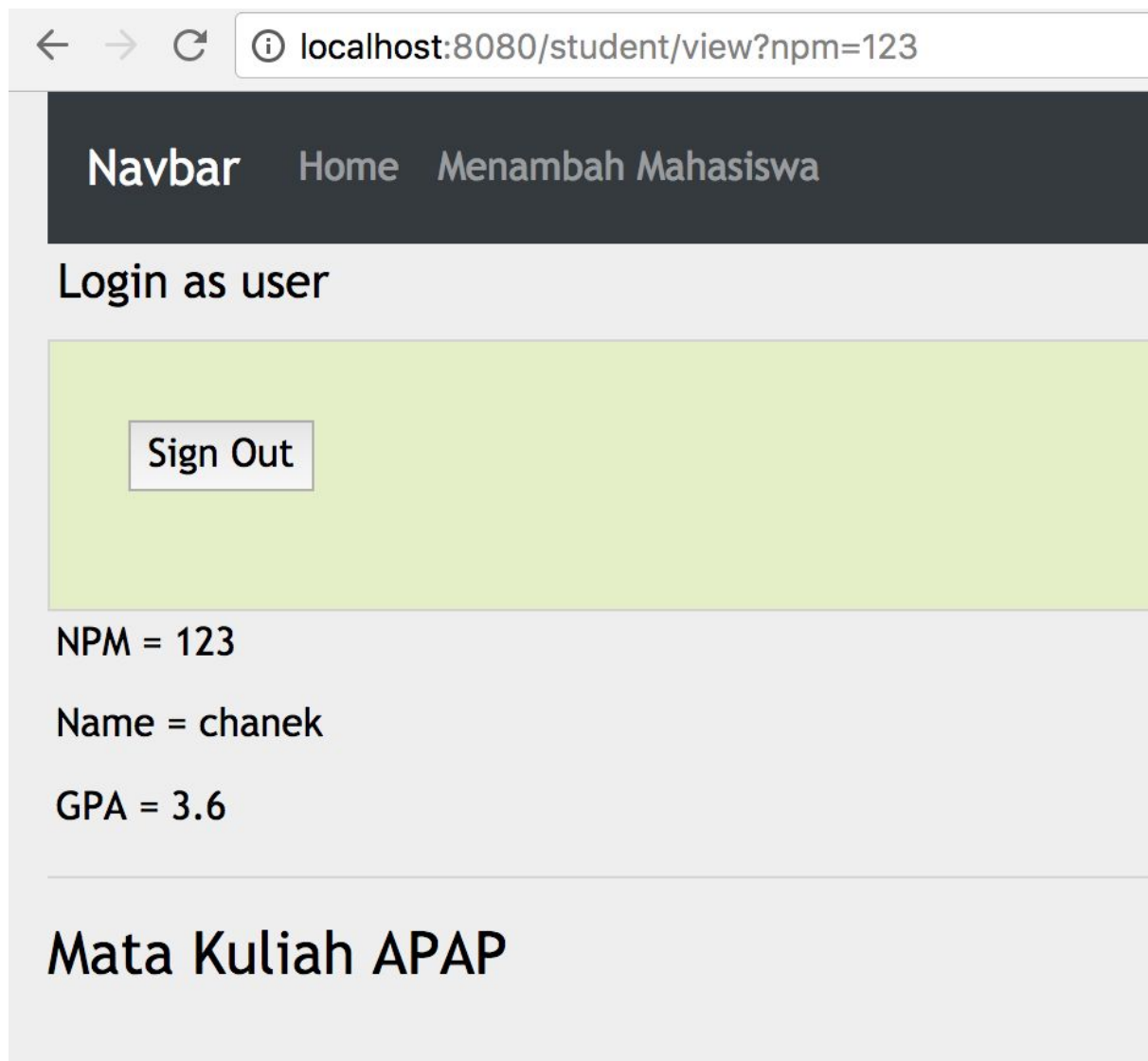


Nama : Laborawaty Rajagukguk
NPM : 1606954855

2. Kemudian dilakukan penambahan user dengan username “user”, password “user”, dan role “USER”. Berikut code yang ditambahkan pada fungsi configureGlobal di file WebSecurityConfig.java

```
@Autowired
public void configureGlobal (AuthenticationManagerBuilder auth) throws Exception
{
    auth.inMemoryAuthentication()
        .withUser("admin").password("admin")
        .roles("ADMIN")
        .and().withUser("user").password("user")
        .roles("USER");
}
```

Berikut tampilan login dengan menggunakan akun “user”



Nama : Laborawaty Rajagukguk
NPM : 1606954855

Selain itu penggunaan role dapat digunakan dalam pembuatan menu juga, sehingga role tertentu yang tidak dapat mengakses sebuah feature, maka menu feature tersebut akan dihilangkan.

Berikut tampilan ketika pengguna dengan akun “user” mengakses feature “viewall” student. Dimana sudah diatur dalam fungsi “configure” bahwa feature tersebut hanya dapat diakses oleh pengguna dengan role “ADMIN”.



Whitelabel Error Page

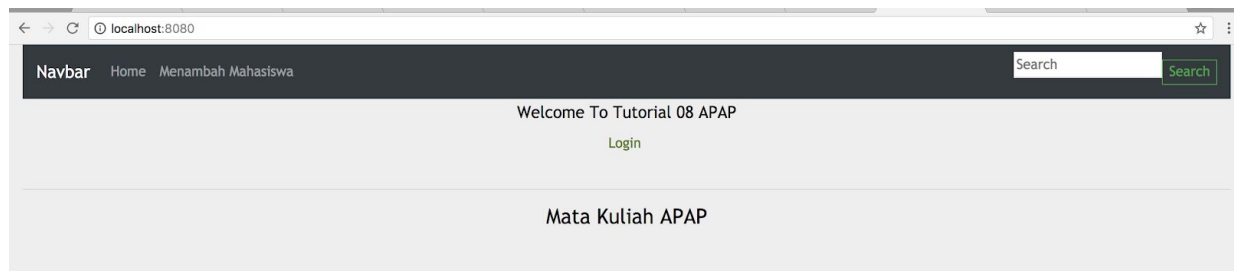
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Apr 24 20:26:13 WIB 2018

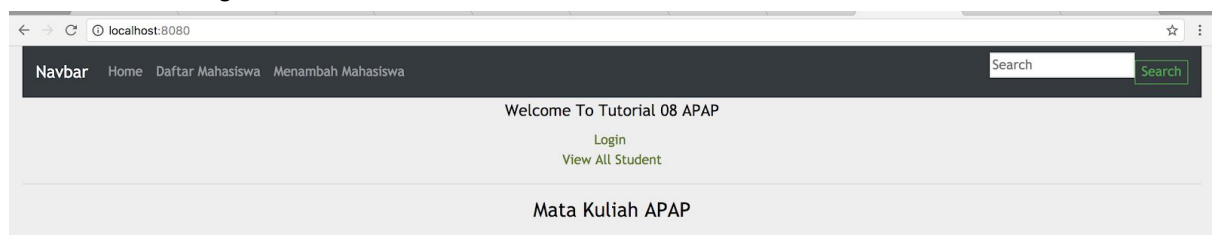
There was an unexpected error (type=Forbidden, status=403).

Access is denied

Dan berikut tampilan dimana menu dari fungsi viewall student tidak ditampilkan jika pengguna login dengan menggunakan akun “user”



Tetapi ketika pengguna login dengan menggunakan akun “admin” maka menu “viewall” student akan ditampilkan. Perubahan pada menu navbar dan juga link “View All Student” dibawah link “Login”



Menggunakan Database

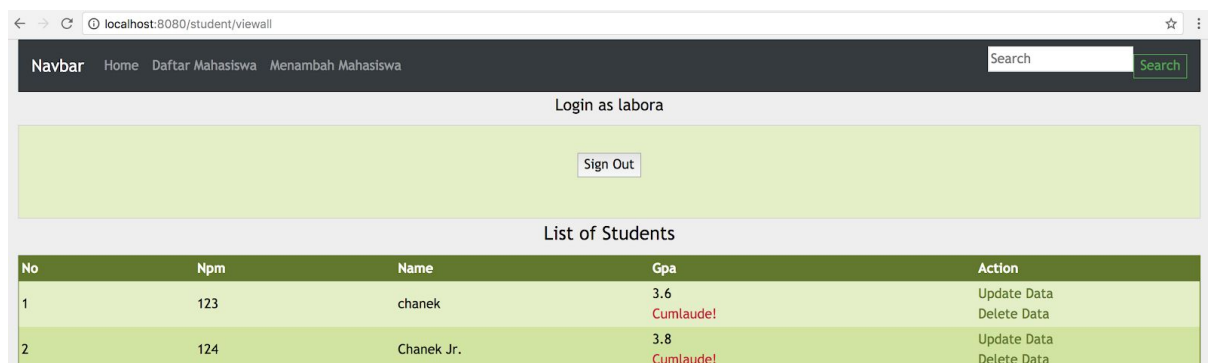
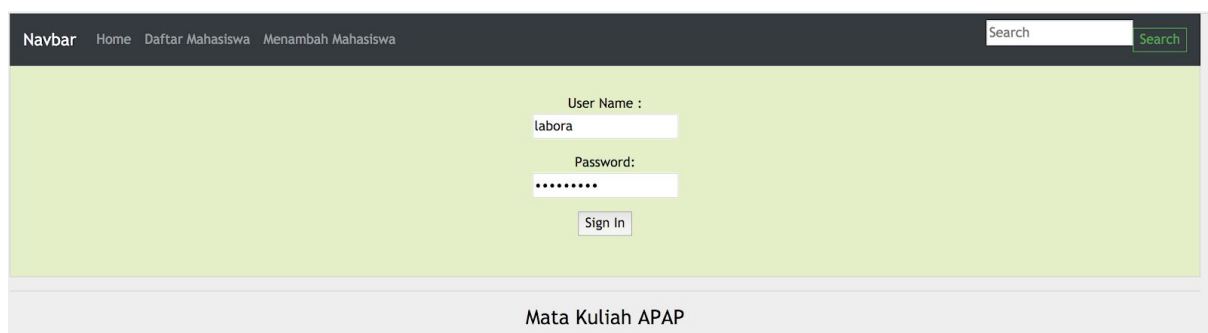
Cara lain untuk mendaftarkan pengguna adalah dengan menyimpan daftar pengguna pada table di database sistem yang digunakan. Hal ini akan mempermudah jika adanya

Nama : Laborawaty Rajagukguk
NPM : 1606954855

penambahan pengguna terus menerus, sehingga tidak diperlukan pengubahan code, hanya perlu menambahkan row pada tabel pengguna dan tabel role di database.

Pertama sekali ditambahkan tabel pengguna (users) dan juga tabel role (user_role) pada database. Kemudian fungsi "configureGlobal" tidak diperlukan lagi dan digantikan dengan menggunakan fungsi "configureAuthentication". Fungsi tersebut akan melakukan query ke database untuk menemukan dan membandingkan username yang digunakan oleh pengguna benar atau tidak serta menemukan role dari user tersebut. Role dari user tersebut dapat ditemukan dengan adanya relasi antara table users dan user_role.

Sebagai contoh saya menambahkan user pada tabel database dengan username "labora" dan password "qwerty123" dan pada tabel user_roles menambahkan role "ROLE_ADMIN" untuk username "labora". Sehingga pengguna dapat login dengan username "labora" dan password "qwerty123" serta mendapat role sebagai ADMIN



No	Npm	Name	Gpa	Action
1	123	chanek	3.6 Cumlaude!	Update Data Delete Data
2	124	Chanek Jr.	3.8 Cumlaude!	Update Data Delete Data

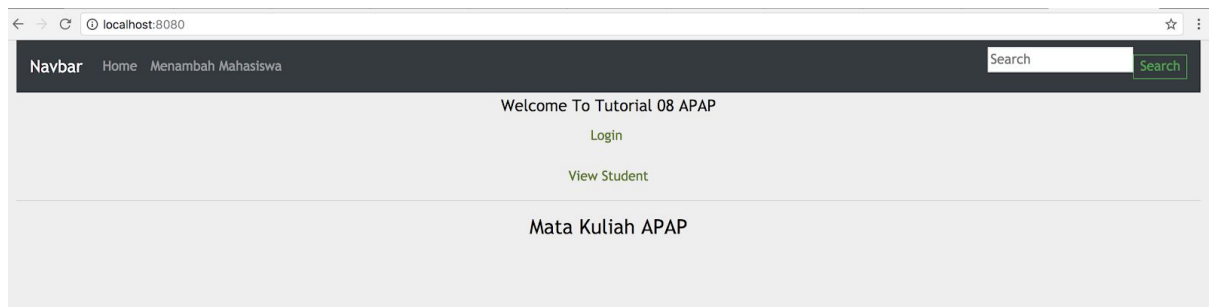
Latihan

- Menu view student by npm beserta input field dan button untuk memasukan nomor npm student yang ingin dicari pada halaman index user dengan role USER. Berikut tampilannya.
- Tambahkan link untuk menuju view student by npm dan tambahkan pada link penggunaan role untuk mengakses menu tersebut.

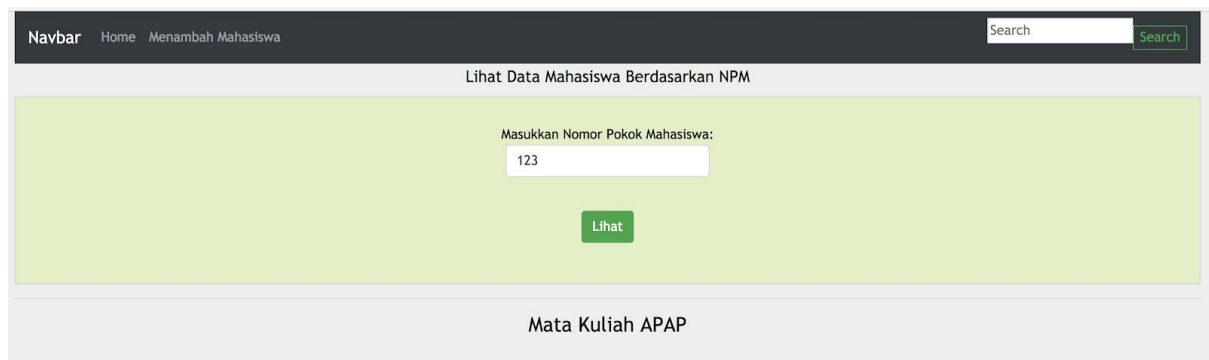
```
<a th:if="${#httpServletRequest.isUserInRole('USER')}" href="/student/search">View Student</a><br/>
```

Nama : Laborawaty Rajagukguk
NPM : 1606954855

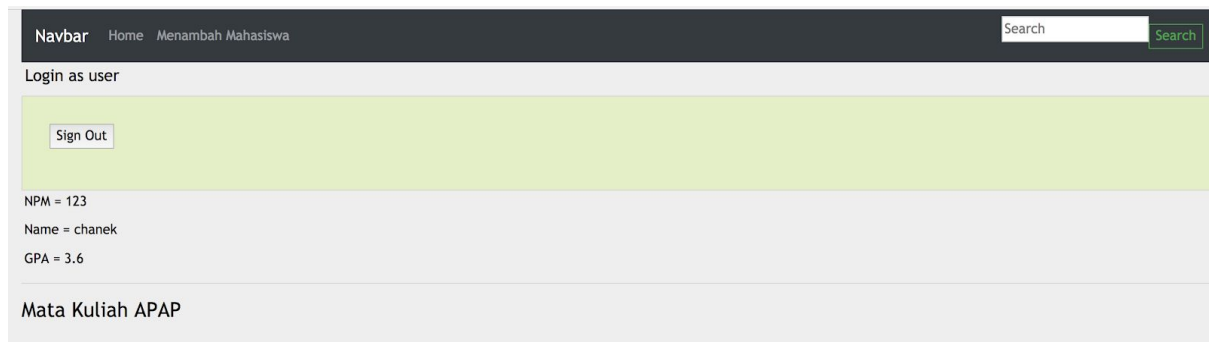
Berikut tampilan link “view student” pada index pengguna dengan role USER



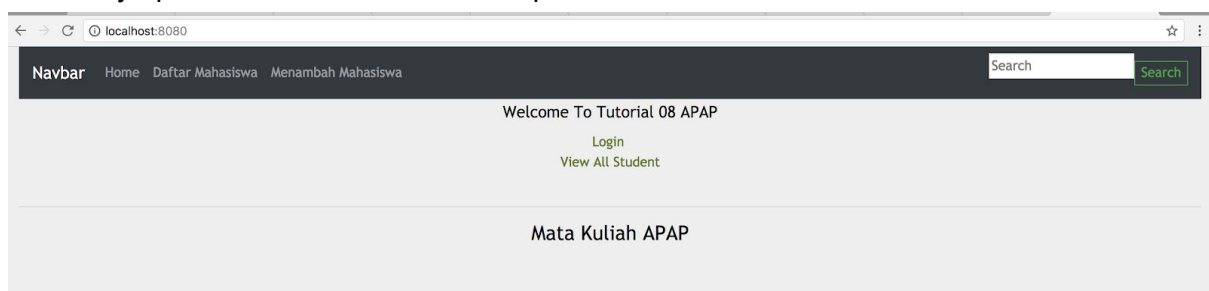
- Kemudian tambahkan sebuah halaman yang berisi field inputan untuk memasukkan npm student yang ingin dicari datanya



- Setelah menekan tombol “Lihat” maka akan ditampilkan data student sesuai npm yang dimasukkan



- Ketika login sebagai pengguna dengan role ADMIN, maka link menu “view student” by npm tersebut tidak akan ditampilkan



4. User dengan role ADMIN dapat melihat view all student dan view student by npm juga, dilakukan dengan menghapus code berikut :

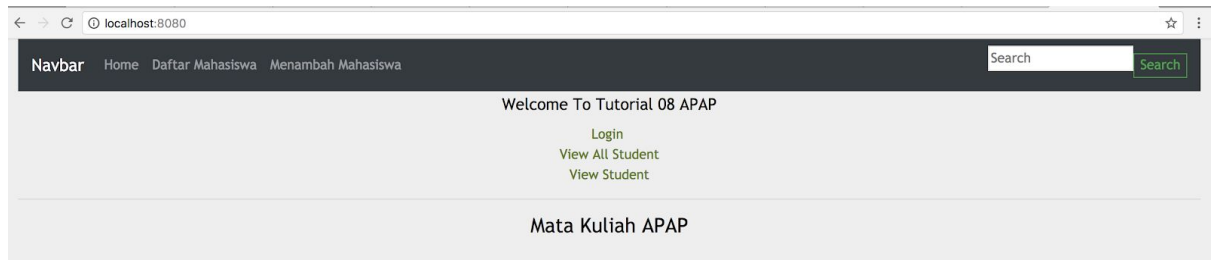
Nama : Laborawaty Rajagukguk
NPM : 1606954855

```
.antMatchers("/student/view/**").hasRole("USER")
```

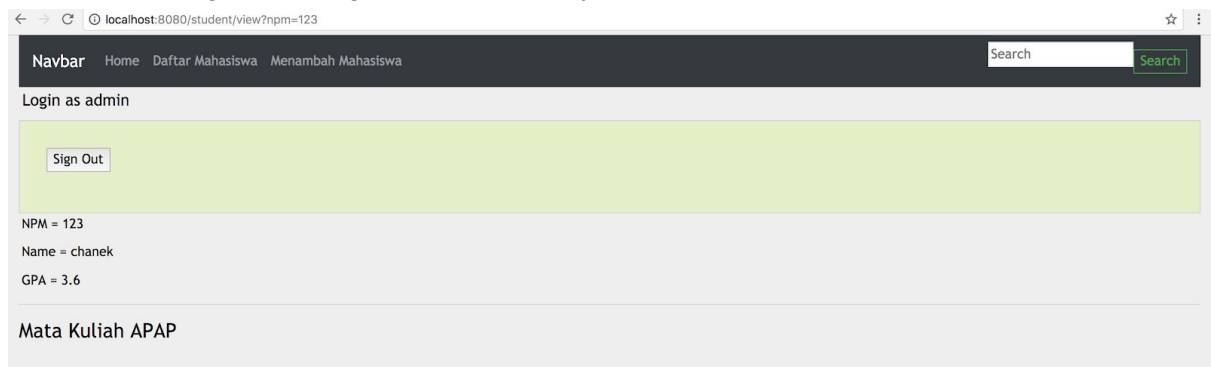
dan juga code pada link menu view student

```
th:if="${#httpServletRequest.isUserInRole('USER')}"
```

Maka tampilan index jika menggunakan akun pengguna dengan role ADMIN akan seperti gambar dibawah.



Dan dapat mengakses fungsi view student by npm.



Catatan :

Terdapat masalah pada tampilan bootstrap setelah mejalankan fungsi logout, tetapi jika halaman index di refresh kembali, bootstrap akan tampil seperti semula dan fungsi berjalan seperti yang diharapkan.