

1. Ringkasan Materi

Pada kali tutorial 8 kali ini dipelajari terkait web security dimana setiap halaman web mempunyai hak akses hanya untuk beberapa role saja. Untuk membuat suatu web security config telah disediakan class `WebSecurityConfigurerAdapter` yang dapat digunakan. Terdapat 2 tipe role yang digunakan yaitu role admin dan role user dimana secara umum role admin mempunyai hak akses yang lebih luas dibanding role user. Dipelajari juga bagaimana melakukan konfigurasi untuk menampilkan menu berdasarkan dengan role ketika login.

2. Penjelasan Method Latihan

1. Menambahkan tombol Sign Out dan cetak nama user untuk halaman student/view/{npm}

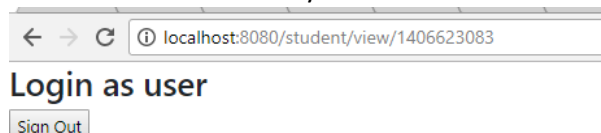
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>View Student by NPM</title>
  </head>
  <body>
    <h2 th:text="'Login as ' + ${httpServletRequest.remoteUser}">Login as</h2>
    <form th:action="@{/logout}" method="post">
      <input type="submit" value="Sign Out"/>
    </form>
    <div th:replace = "fragments/fragment :: header" ></div>
    <h3 th:text="'NPM = ' + ${student.npm}">Student NPM</h3>
    <h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
    <h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
  </body>
</html>
```

Sama seperti pada halaman view all yang ada pada modul tutorial. Untuk menambahkan cetak nama user maka ditambahkan `#httpServletRequest.remote user`. Sedangkan untuk menambahkan tombol login dan kemudiang dilakukan direct ke halaman login maka pada halaman view ditambahkan dengan form dengan type input submit dan action `\logout`.

2. Menambahkan user baru dengan username "user", password "user", dan role "USER". Mencoba apakah bisa login dengan akun tersebut.

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

Dilakukan penambahan pada method `configureGlobal` dimana sebelumnya telah ada user admin. Sama seperti user admin maka ditambahkan juga untuk `auth.inMemoryAuthentication` untuk username user. `Withuser` adalah nama username yang akan digunakan ketika login, password adalah password yang akan dilakukan proses verifikasi dengan username yang telah dideklarasikan sebelumnya dan roles adalah nama rolesnya.



3. Menampilkan 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

```
<div th:if="${#httpServletRequest.isUserInRole('USER')}">
  <form th:action="@{/student/view/}" method="post" >
    <div><label> NPM : <input type="text" name="npm"/></label></div>
    <div><input type="submit" value="Lihat Data Student"/></div>
  </form>
```

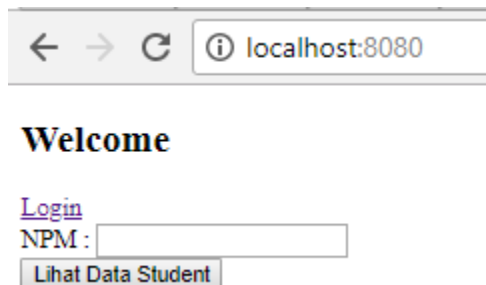
Pada bagian html index ditambahkan terlebih dahulu fitur untuk memasukkan npm dengan tag input dan button dimana action yang dilakukan setelah submit adalah mengakses controller student/view. Selain itu dilakukan pengecekan apakah user yang login adalah user atau bukan. Jika role user maka baru akan menampilkan index dengan inputan npm.

```
@RequestMapping("/student/view")
public String view (Model model,
    @RequestParam(value = "npm", required = false) String npm)
{
    return "redirect:/student/view/" + npm;
}

@RequestMapping("/student/view/{npm}")
public String viewPath (Model model,
    @PathVariable(value = "npm") String npm)
{
    StudentModel student = studentDAO.selectStudent (npm);

    if (student != null) {
        model.addAttribute ("student", student);
        return "view";
    } else {
        model.addAttribute ("npm", npm);
        return "not-found";
    }
}
```

Pada bagian controller ketika pada halaman index dilakukan submit maka akan direct ke method view. Untuk dapat menampilkan npm pada url maka dari method view dilemparkan ke halaman yang menerima path variable. Karena method pada index adalah post maka npm tidak akan muncul dan tetap dianggap sebagai requestmapper.



← → ↻ ⓘ localhost:8080

Welcome

[Login](#)

NPM:

Tampilan index sebagai user.

Tampilan setelah role user submit npm.

4. Membuat agar user dengan role ADMIN dapat melihat view all student dan view student by npm juga.

```
<div th:if="${#httpServletRequest.isUserInRole('USER')} or #httpServletRequest.isUserInRole('ADMIN')}">
  <form th:action="@{/student/view/}" method="post" >
    <div><label> NPM : <input type="text" name="npm"/></label></div>
    <div><input type="submit" value="Lihat Data Student"/></div>
  </form>
</div>
```

Jika pada nomor 3 hanya di set pada tampilan index input npm hanya muncul ketika login dengan role user maka pada kali ini pada index dilakukan penambahan kondisi yaitu jika login dengan role user dan dengan role admin dengan `nth:if`. Kemudian lainnya sama dengan tahap nomor 3.

```
@Override
protected void configure(HttpSecurity http) throws Exception
{
    http
        .authorizeRequests()
        .antMatchers("/").permitAll()
        .antMatchers("/student/viewall").hasRole("ADMIN")
        .antMatchers("/student/view/**").hasAnyRole("USER", "ADMIN")
        .anyRequest().authenticated()
        .and()
        .formLogin()
        .loginPage("/login")
        .permitAll()
        .and()
        .logout()
        .permitAll();
}
```

Pada `WebConfigSecurity` dilakukan perubahan pada `anyMacher` untuk path `student/view/` dimana awalnya menggunakan method `hasRole` karena hanya dapat satu role user saja. Pada nomor 4 ini dilakukan perubahan untuk role yang dapat mengakses path tersebut lebih dari satu sehingga method yang digunakan diubah menjadi `hasAnyRole`.