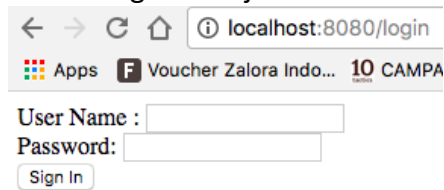


Nama: Muhammad Hasby Rosyadi  
NPM: 1306386642  
APAP-A

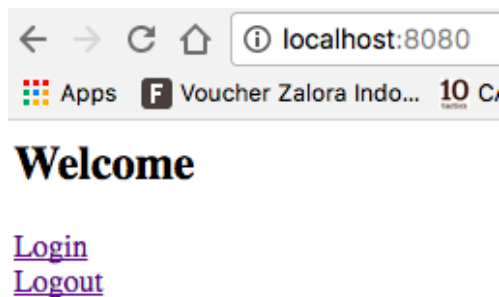
1. Halaman login. Menjadi sarana autentikasi setiap member yang masuk.



2. pembuatan role baru untuk admin pada kasus ini masih menggunakan in memory.

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

pada tutor ini. Penulis menambahkan fitur logout berupa link, yang bertujuan agar memudahkan pengguna ketika ingin langsung keluar. Seperti pada gambar dibawah



untuk menjalankan fitur tersebut perlu adanya tambahan pada class WebSecurityConfig yaitu logoutRequestMatcher dan LogoutSuccessUrl

```
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/")
            .permitAll()
            .antMatchers("/course/**").hasRole("ADMIN")
            .antMatchers("/student/**").hasAnyRole("ADMIN", "USER")
            .anyRequest()
            .authenticated()
            .and()
            .formLogin()
            .loginPage("/login")
            .permitAll()
            .and()
            .logout()
            .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
            .logoutSuccessUrl("/login")
            .permitAll();
    }
}
```

Nama: Muhammad Hasby Rosyadi  
NPM: 1306386642  
APAP-A

3. untuk menyelesaikan bagian ini, hal-hal yang perlu dilakukan adalah

- membuat student dapat diakses oleh 2 role yaitu admin dan user dengan menggunakan hasAnyRole.

```
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {  
    @Override  
    protected void configure(HttpSecurity http) throws Exception {  
        http  
            .authorizeRequests()  
            .antMatchers("/").  
            .permitAll()  
            .antMatchers("/course/**").hasRole("ADMIN")  
            .antMatchers("/student/**").hasAnyRole("ADMIN", "USER")  
            .anyRequest()  
            .authenticated()  
            .and()  
            .formLogin()  
            .loginPage("/login")  
            .permitAll()  
            .and()  
            .logout()  
            .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))  
            .logoutSuccessUrl("/login")  
            .permitAll();  
    }  
}
```

- membaca username dan password dari database yang dibuat yaitu dengan menambahkan kode dibawah pada class WebSecurityConfig

```
@Autowired  
DataSource dataSource;  
@Autowired  
public void configAuthentication (AuthenticationManagerBuilder auth) throws Exception {  
    auth  
        .jdbcAuthentication()  
        .dataSource(dataSource)  
        .usersByUsernameQuery("select username, password, enabled from users where username =? ")  
        .authoritiesByUsernameQuery("select username, role from user_roles where username =? ");  
}
```

- selanjutnya membuat link pada index untuk hak akses yang diberikan pada setiap role yaitu

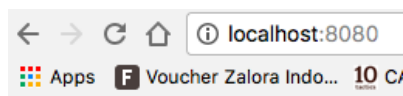
- user hanya dapat mengakses ViewAllStudent
- admin hanya dapat mengakses ViewAllStudent dan ViewAllCourse yang mana akses diberikan berupa link, ketika berhasil login.

- Pada halaman index.html tambahkan kode dibawah untuk menampilkan link seperti permintaan diatas

```
<h2>Welcome</h2>  
<a href="/login">Login</a><br/>  
<a href="/logout">Logout</a><br/>  
<a th:href="${#httpServletRequest.isUserInRole('ADMIN')}" href="/course/viewall"> View All Course </a><br/>  
<a th:href="${#httpServletRequest.isUserInRole('USER')} or ${#httpServletRequest.isUserInRole('ADMIN')}" href="/student/viewall"> View All Student </a><br/>
```

- Selanjutnya login dengan username: user dan pass: password. Maka akan menampilkan view seperti dibawah

Nama: Muhammad Hasby Rosyadi  
NPM: 1306386642  
APAP-A



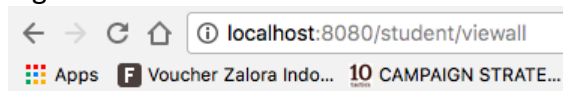
## Welcome

[Login](#)

[Logout](#)

[View All Student](#)

- Ketika view all student maka akan menampilkan halaman seperti dibawah dengan login as user



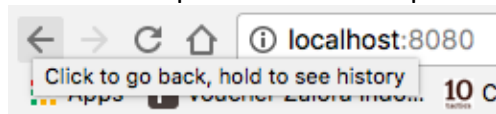
## Login as user

[Sign Out](#)

## All Students

### No. 1

- Kemudian logout dan coba menggunakan username: admin dan pass: admin. Maka akan menampilkan halaman seperti dibawah



## Welcome

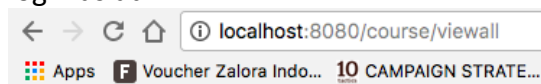
[Login](#)

[Logout](#)

[View All Course](#)

[View All Student](#)

- Ketika View All Course dipilih maka akan menampilkan view seperti dibawah dengan login as admin



## Login as admin

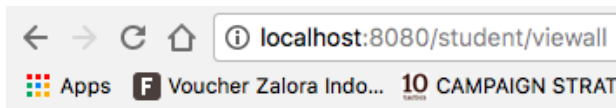
[Sign Out](#)

## All Course

### No. 1

- Kemudian back. Dan pilih View All Student, maka akan menampilkan view seperti dibawah dengan login as admin

Nama: Muhammad Hasby Rosyadi  
NPM: 1306386642  
APAP-A



## Login as admin

[Sign Out](#)

## All Students

### No. 1

- Pada bagian ini penulis melakukan percobaan pada menampilkan nama user yang login dari controller yang dilempar ke halaman html, percobaan dilakukan dengan menggunakan SecurityContextHolder untuk mendapatkan username

```
User user = (User)SecurityContextHolder.getContext().getAuthentication().getPrincipal();
String name = user.getUsername();

model.addAttribute("name", name);
```

potongan kode diatas di masukan pada StudentController pada bagian ViewallStudent dan ViewAllCourse.
- Selanjutnya ganti bagian login as dengan name pada halaman view all student

```
<h2 th:text="'Login as ' + ${name}">
    Login as</h2>
<form th:action = "@{/logout}" method = "post">
    <input type = "submit" value = "Sign Out"/></form>
<h1>All Students</h1>
```
- Selanjutnya ganti bagian login as dengan name pada halaman view all course

```
<body>
    <h2 th:text="'Login as ' + ${name}">
        Login as</h2>
    <form th:action="@{/logout}" method="post">
        <input type="submit" value="Sign Out" />
    </form>
    <h1>All Course</h1>
```
- Maka akan menampilkan hasil yang sama dengan tutorial yang dibuat.

### Lesson Learn

1. Mengetahui cara pembuatan autentikasi.
2. Mengetahui cara konfigurasi otorisasi berdasarkan role.
3. Mengetahui pembuatan metode pembuatan user yang berhak masuk kedalam sistem dalam database.
4. Mengetahui cara mengambil user yang login dari controller.
5. Mengetahui cara membuat logout tanpa form.