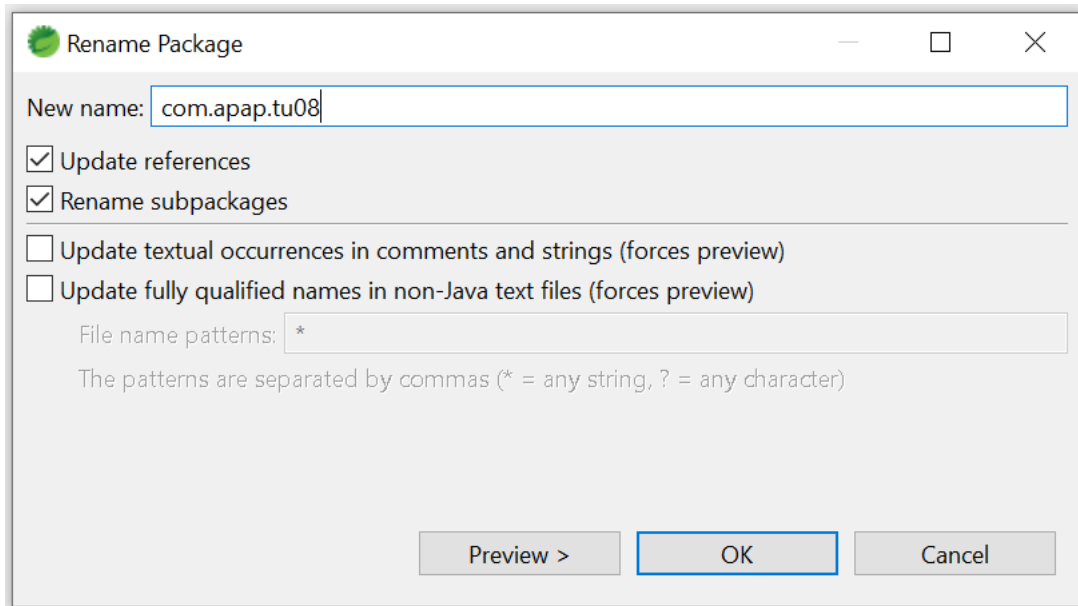


Tutorial 08 – Web Security

Tutorial ini mengasumsikan penggunaan pada *Windows*, tanpa melarang penggunaan OS lain, silakan menyesuaikan.

1. Pendahuluan

- Unduh Maven Project (Tutorial-06) yang ada di Scele lalu ekstrak (tip: pada *workspace*). Agar berbeda dengan *project* sebelumnya ubah nama folder menjadi tutorial-08.
- Jalankan STS, lalu *import project* tersebut.
- Ubah nama package dari *.tu07 menjadi *.tu08, klik kanan package >> Refactor >> Rename



- Tambahkan *dependency* Spring Boot Security pada pom.xml sbb.:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```
- Buat DB dengan nama tu08, lalu import SQL yang ada di Scele (Tutorial-06 schema dan data).

2. Login (Global)

- Buatlah *package* baru dengan nama com. apap.tu08.security
- Pada *package* tersebut buatlah sebuah *class* bernama WebSecurityConfig.java dengan spesifikasi sbb.:

```
package com.apap.tu08.security;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;
```

```

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/css/**").permitAll()
            .antMatchers("/js/**").permitAll()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .loginPage("/login")
            .permitAll()
            .and()
            .logout().logoutRequestMatcher(new AntPathRequestMatcher("/logout")).logoutSuccessUrl("/login")
            .permitAll();
    }

    @Autowired
    public void configureGlobal (AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication()
            .passwordEncoder(encoder())
            .withUser("cokicoki").password(encoder().encode("enaksekali"))
            .roles("USER");
    }

    @Bean
    public BCryptPasswordEncoder encoder() {
        return new BCryptPasswordEncoder();
    }
}

```

Kode pada *method* `configure` menjelaskan bahwa *form* untuk *login* berada di `"/login"` dan dapat diakses siapapun. Selain pada `"/login"` perlu dilakukan autentikasi.

Kode pada *method* `configure Global` menyimpan autentikasi *in-memory* dengan *username* "cokicoki", *password* "enaksekali", dan *role*-nya "USER".

- Buatlah halaman `login.html`, *template* tersedia di Scele.
- Pada `home.html`, ubah *tag* pada tulisan Hello, menjadi sbb.:

```
<h2 th:text=" 'Hello ' + ${#httpServletRequest.remoteUser} + ' !' ">Login as</h2>
```

- Buatlah *class* `PageController.java` dengan spesifikasi sbb.:

```

package com.apap.tu08.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class PageController {
    @RequestMapping("/")
    public String home () {
        return "home";
    }

    @RequestMapping("/login")
    public String login () {
        return "login";
    }
}

```

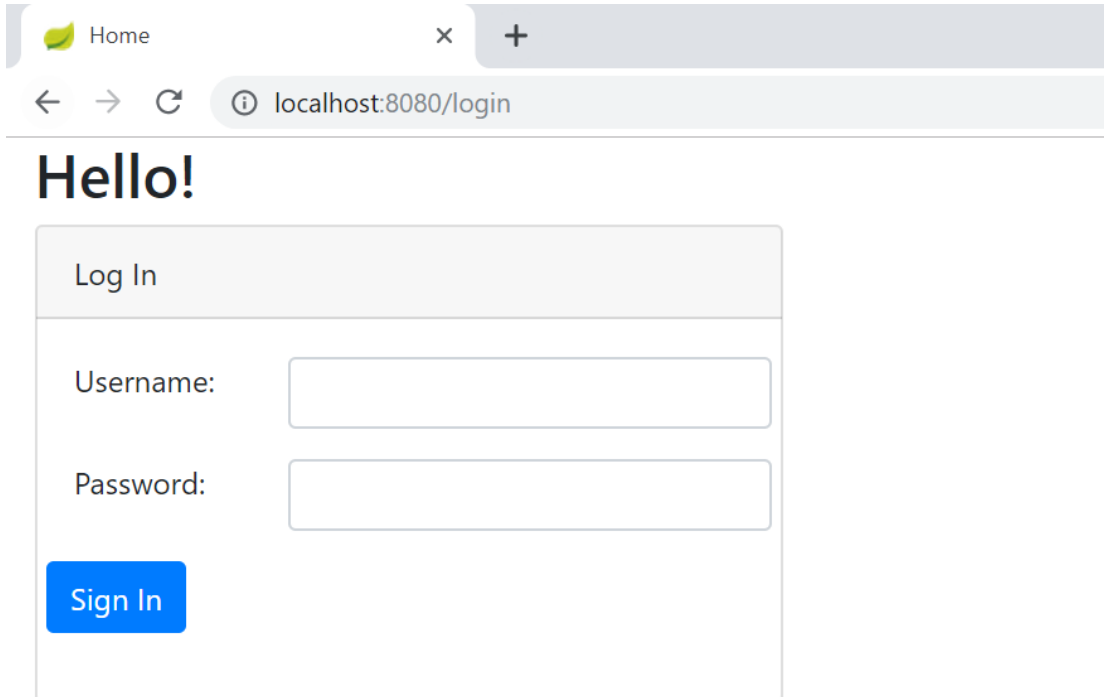
- Hapus routing (`"/"`) yang ada pada `PilotController.java` agar tidak terjadi ambigui.

```

// @RequestMapping("/")
// private String home() {
//     return "home";
// }

```

- Jalankan program dan buka localhost:8080 maka tampilannya adalah sbb.:



Home x +

← → ↻ ⓘ localhost:8080/login

Hello!

Log In

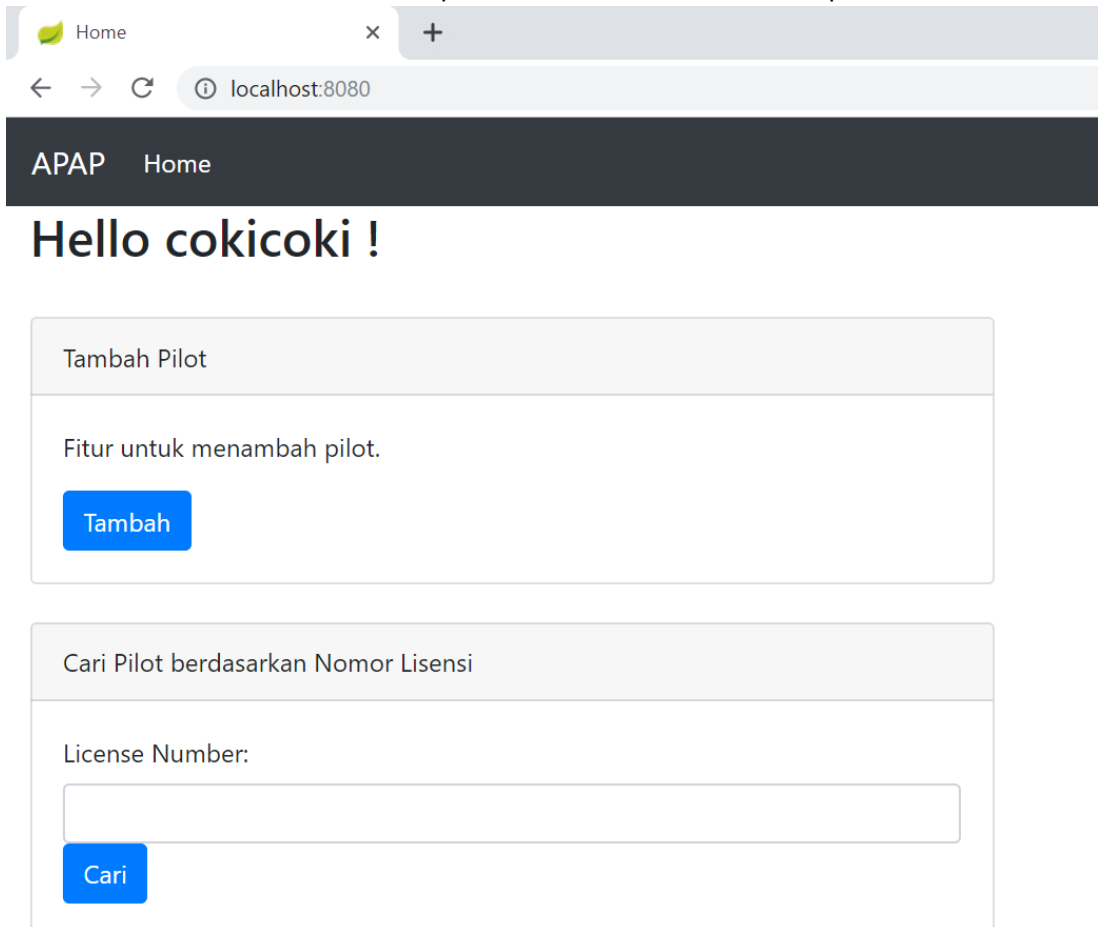
Username:

Password:

Sign In

Anda akan dialihkan ke halaman login secara otomatis.

- Masukkan username “cokicoki” dan password “enaksekali” akan tampil sbb.:



Home x +

← → ↻ ⓘ localhost:8080

APAP Home

Hello cokicoki !

Tambah Pilot

Fitur untuk menambah pilot.

Tambah

Cari Pilot berdasarkan Nomor Lisensi

License Number:

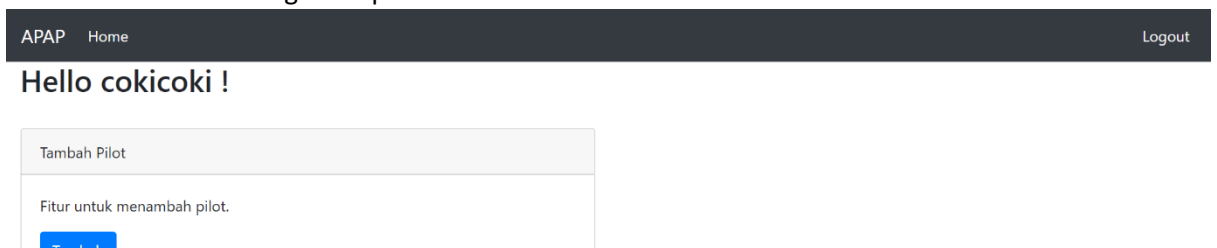
Cari

3. Logout

- Tambahkan tombol logout pada fragment.html, misalnya sbb.:

```
<body>
  <nav th:fragment="navbar" class="navbar navbar-expand-lg navbar-dark bg-dark">
    <a class="navbar-brand" href="#">APAP</a>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" th:href="@{/}">Home</a>
        </li>
      </ul>
      <ul class="navbar-nav pull-right">
        <li class="nav-item active">
          <a class="nav-link" th:href="@{/Logout}">Logout</a>
        </li>
      </ul>
    </div>
  </nav>
</body>
```

- Akan muncul tombol logout seperti sbb.:



Jika Anda melakukan logout, maka akan dialihkan ke halaman login.

4. Otentikasi dan Otorisasi Menggunakan Database – Create User

- Buatlah *class* UserRoleModel dengan spesifikasi sbb.:

```
package com.apap.tu08.model;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Lob;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

@Entity
@Table(name="user_role")
public class UserRoleModel {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @NotNull
    @Size(max=50)
    @Column(name="username", nullable=false)
    private String username;

    @NotNull
    @Lob
    @Column(name="password", nullable=false)
    private String password;

    @NotNull
    @Size(max=50)
    @Column(name="role", nullable=false)
    private String role;
}
```

Lengkapi dengan getter dan setter.

- Untuk menambah fitur *user* baru ke dalam DB buatlah *class* UserRoleService.java dengan spesifikasi sbb.:

```
package com.apap.tu08.service;

import com.apap.tu08.model.UserRoleModel;

public interface UserRoleService {
    UserRoleModel addUser(UserRoleModel user);
    public String encrypt(String password);
}
```

- Lalu buat *class* UserRoleDB.java dengan spesifikasi sbb.:

```
package com.apap.tu08.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.apap.tu08.model.UserRoleModel;

@Repository
public interface UserRoleDb extends JpaRepository<UserRoleModel, Long> {
    UserRoleModel findByUsername(String username);
}
```

- Buat UserRoleServiceImpl.java dengan spesifikasi sbb.:

```
package com.apap.tu08.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import com.apap.tu08.model.UserRoleModel;
import com.apap.tu08.repository.UserRoleDb;

@Service
public class UserRoleServiceImpl implements UserRoleService {
    @Autowired
    private UserRoleDb userDb;

    @Override
    public UserRoleModel addUser(UserRoleModel user) {
        String pass = encrypt(user.getPassword());
        user.setPassword(pass);
        return userDb.save(user);
    }

    @Override
    public String encrypt(String password) {
        BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
        String hashedPassword = passwordEncoder.encode(password);
        return hashedPassword;
    }
}
```

- Tambahkan UserRoleController.java dengan spesifikasi sbb.:

```
package com.apap.tu08.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.apap.tu08.model.UserRoleModel;
import com.apap.tu08.service.UserService;

@Controller
@RequestMapping("/user")
public class UserRoleController {
    @Autowired
    private UserRoleService userService;

    @RequestMapping(value="/addUser", method=RequestMethod.POST)
    private String addUserSubmit(@ModelAttribute UserRoleModel user) {
        userService.addUser(user);
        return "home";
    }
}
```

- Pada home.html, tambahkan sbb.:

```
<body>
    <nav th:replace="fragments/fragment :: navbar"></nav>

    <div class="container-fluid">
        <h2 th:text=" 'Hello ' + ${httpServletRequest.remoteUser} + ' !'">Login as</h2>

        <div class="row">
            <div class="col-md-6">
                <div class="card">
                    <div class="card-header">
                        Tambah User Baru
                    </div>
                    <div class="card-body">
                        <form th:action="@{/user/addUser}" method="post">
                            <div class="row form-group">
                                <label class="col-sm-4">User Name</label>
                                <input class="col-sm-8 form-control input-sm" type="text" name="username"/>
                            </div>
                            <div class="row form-group">
                                <label class="col-sm-4">Password: </label>
                                <input class="col-sm-8 form-control input-sm" type="password" name="password"/>
                            </div>
                            <div class="row form-group">
                                <label class="col-sm-4">Role:</label>
                                <input class="col-sm-8 form-control input-sm" type="text" name="role"/>
                            </div>
                            <div>
                                <button type="submit" class="btn btn-primary">Simpan</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
        <br>
        <div class="row">
```

- Tambahkan akun pilot dengan username viktor, password 1234, dan role PILOT.

5. Otentikasi dan Otorisasi Menggunakan Database - Login

- Buat *class* UserDetailsServiceImpl.java dengan spesifikasi sbb.:

```
package com.apap.tu08.security;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import com.apap.tu08.model.UserRoleModel;
import com.apap.tu08.repository.UserRoleDb;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {
    @Autowired
    private UserRoleDb userRoleDb;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        UserRoleModel user = userRoleDb.findByUsername(username);

        Set<GrantedAuthority> grantedAuthorities = new HashSet<GrantedAuthority>();
        grantedAuthorities.add(new SimpleGrantedAuthority(user.getRole()));

        return new User(user.getUsername(), user.getPassword(), grantedAuthorities);
    }
}
```

- Pada *class* WebSecurityConfig ubah dan tambahkan kode sbb.:

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/css/**").permitAll()
            .antMatchers("/js/**").permitAll()
            .antMatchers("/flight/**").hasAnyAuthority("PILOT")
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .loginPage("/login")
            .permitAll()
            .and()
            .logout().logoutRequestMatcher(new AntPathRequestMatcher("/logout")).logoutSuccessUrl("/login")
            .permitAll();
    }

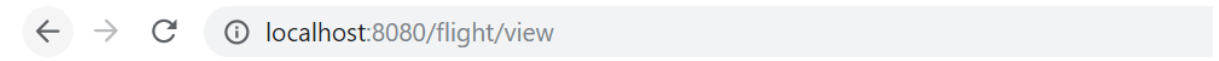
    @Bean
    public BCryptPasswordEncoder encoder() {
        return new BCryptPasswordEncoder();
    }

    @Autowired
    private UserDetailsService userDetailsService;

    @Autowired
    public void configureAuthentication(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(encoder());
    }
}
```

Tip: Hapus/comment pada method configureGlobal untuk menghilangkan akun "cokicoki."

- Coba jalankan dan login menggunakan akun pilot yang anda tambahkan tadi.
- Coba tambahkan akun baru dengan role UMUM ke dalam DB lalu coba akses halaman flight menggunakan akun tersebut, maka akan tampil error berikut:



Whitelabel Error Page

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

Tue May 07 14:45:10 WIB 2019

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

Forbidden

Hal ini dikarenakan seluruh route ("/flight/") hanya bisa diakses oleh user yang memiliki role PILOT.

Latihan

1. Tambahkan *user* dengan *role* ADMIN, agar fitur *add*, *view*, *update* dan *delete* pilot hanya dapat diakses oleh admin.
2. Tambahkan fitur *update password* yang dapat diakses oleh semua *role user*. Contoh tampilan:

A screenshot of a web form titled 'Update password'. The form has three input fields: 'Password Lama', 'Password Baru:', and 'Konfirmasi Password Baru:'. Below the input fields is a blue button labeled 'Simpan'.

3. Pada saat pembuatan *user* baru dan *update password user*, tambahkan ketentuan agar *password user* mengandung angka dan huruf serta minimal memiliki 8 karakter.

Pengumpulan

1. Penjelasan pengerjaan tutorial dan latihan mengenai apa yang Anda pelajari. Tambahkan *screen capture (output* pada *browser* dan/atau tampilan DB) yang menunjukkan fitur/method berjalan dengan sesuai. Pengumpulan dituliskan dalam satu file dengan format **npm_nama-lengkap-anda.pdf** dan unggah ke submission slot yang disediakan di Scele.
2. Folder tutorial-08 dikumpulkan dengan cara push ke GitHub (<https://github.com/achmad-f-abka/apap>).