

## Lab 9 : Spring Thymeleaf – Signup and Login

Usa Sammapun, Kasetsart University

ในแลปนี้ เราจะสร้างหน้าเว็บการ signup และ login เข้าเว็บร้านอาหาร

### I. Simple Homepage

1. ไปที่ <https://start.spring.io/>
  - ใส่ข้อมูล project ตามต้องการ นิสิตสามารถใช้ตามอาจารย์ได้เลย
  - **ต้องเลือก Spring Boot version ให้เป็น 2.6.11**
  - เลือก Java version ให้ตรงตามเครื่องคอมพิวเตอร์ของนิสิต
2. พิมพ์และเลือก dependencies ตามรูป แล้วกดปุ่ม GENERATE
3. จะได้เป็นไฟล์ .zip ให้ unzip ไฟล์

The screenshot shows the Spring Initializr web application interface. It is divided into several sections for configuring a new Spring project:

- Project:** Includes radio buttons for **Maven Project** (selected) and **Gradle Project**.
- Language:** Includes radio buttons for **Java** (selected), **Kotlin**, and **Groovy**.
- Spring Boot:** Includes radio buttons for versions 3.0.0 (SNAPSHOT), 3.0.0 (M4), 2.7.4 (SNAPSHOT), 2.7.3, 2.6.12 (SNAPSHOT), and **2.6.11** (selected).
- Project Metadata:** Includes input fields for Group (th.ac.ku), Artifact (restaurant), Name (restaurant), Description (Restaurant Web Application), and Package name (th.ac.ku.restaurant). It also has a **Packaging** section with **Jar** (selected) and **War** options, and a **Java** version section with 18, 17, **11** (selected), and 8 options.
- Dependencies:** A list of dependencies with checkboxes: **Spring Web** (WEB, selected), **Spring Data JPA** (SQL, selected), **H2 Database** (SQL, selected), **MySQL Driver** (SQL, selected), **Thymeleaf** (TEMPLATE ENGINES, selected), and **Spring Security** (SECURITY, selected). There is an **ADD DEPENDENCIES...** button with a plus icon.

At the bottom, there are three buttons: **GENERATE** (with a plus icon), **EXPLORE** (with a keyboard shortcut CTRL + SPACE), and **SHARE...**

4. เปิดโปรแกรม IntelliJ แล้วเลือก “Open”
5. เลือกโฟลเดอร์ที่เพิ่ง unzip
6. จะใช้เวลานานในการโหลดโค้ดครั้งแรก

## เชื่อมต่อกับ database

เลือกใช้ database ตามถนัด โดยกำหนดการเชื่อมต่อที่/src/main/resources/application.properties  
(ใช้ port ที่ไม่ตรงกับของ Menu API และเปลี่ยนชื่อ database)

- H2

```
server.port = 8091

# Enabling H2 Console
spring.h2.console.settings.web-allow-others=true
spring.h2.console.enabled=true

# Datasource
spring.datasource.url=jdbc:h2:mem:restaurant
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=test
spring.datasource.password=test

# JPA
spring.jpa.show-sql=true
Spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
```

- MySQL

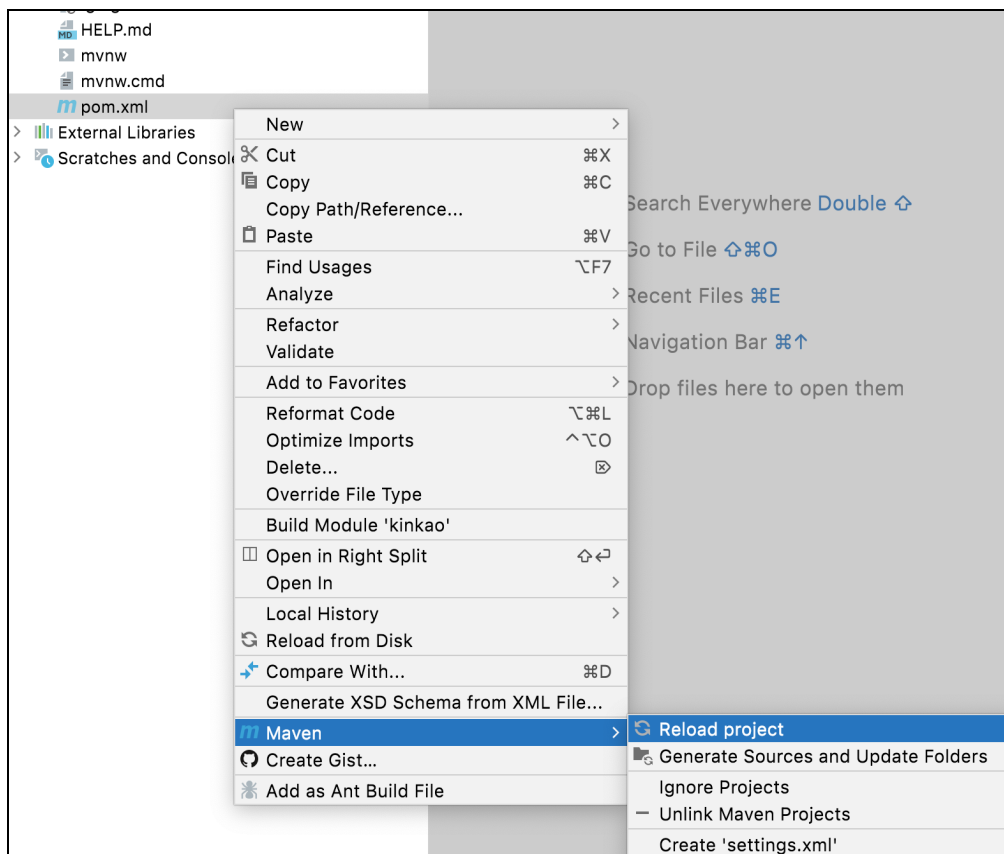
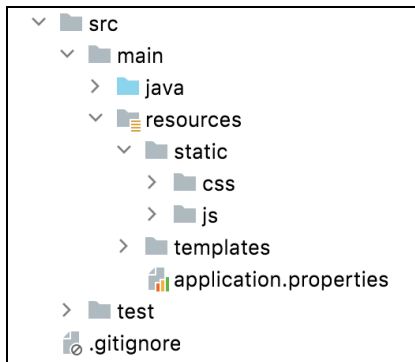
```
server.port = 8091

# Datasource
spring.datasource.url=jdbc:mysql://localhost:3306/restaurant
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=

# JPA
spring.jpa.show-sql=true
Spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

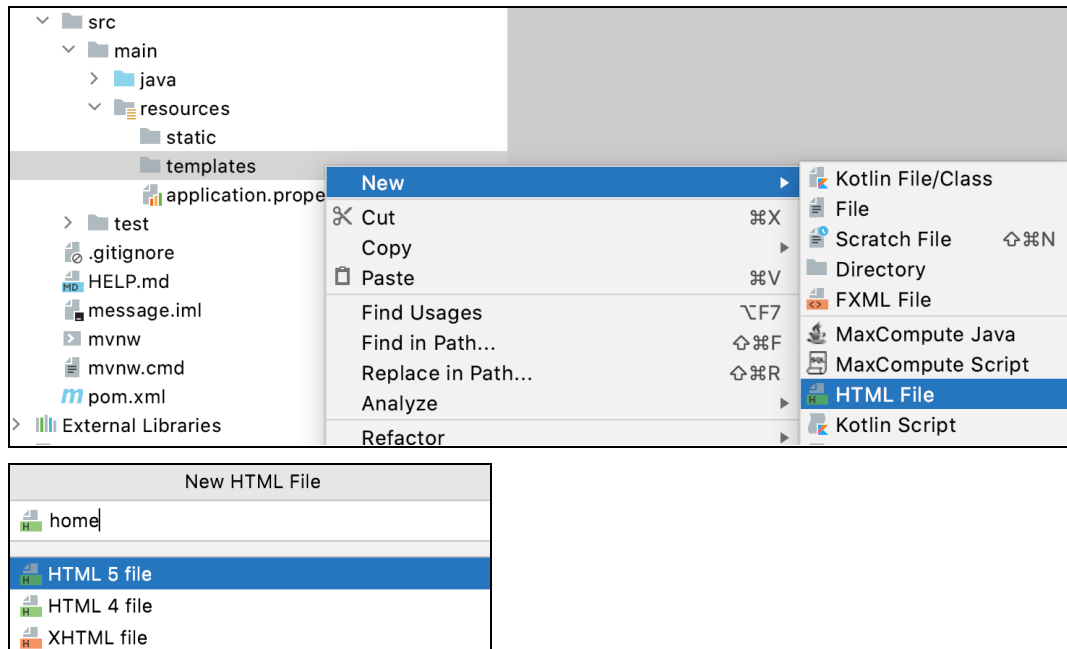
## เพิ่ม Bootstrap เพื่อให้หน้าเว็บสวยงาม

- ดาวน์โหลดไฟล์ Compiled CSS and JS จาก  
<https://getbootstrap.com/docs/5.2/getting-started/download/>  
จะได้ไฟล์ bootstrap-5.2.0-dist.zip จากนั้นให้ unzip ไฟล์
- นำ folder css และ js จากไฟล์ที่ unzip แล้ว ไปไว้ที่ **src/main/resources/static**
  - IntelliJ community version อาจต้อง reload pom.xml เพื่อให้ IntelliJ เห็น folder css/js



## เพิ่มหน้า html

### 9. เพิ่มไฟล์ HTML template ตั้งชื่อว่า home.html ใน `/src/main/resources/templates`



### 10. เพิ่มข้อความ ตามสีเหลือง

- ส่วน `xmlns:th="https://www.thymeleaf.org"` จะทำให้โปรแกรมรู้จัก syntax ของ Spring Thymeleaf
- เพิ่มลิงก์ bootstrap folder และใช้ attribute เช่น `display-6` ของ bootstrap ให้สวยงามขึ้น
- ใช้ `${greeting}` เพื่อระบุว่า จะใช้ตัวแปร `greeting` ที่จะส่งมาจาก controller

```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Restaurant Web Application</title>
  <link th:rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
  <script th:src="@{/js/bootstrap.min.js}"></script>
</head>
<body>
<div class="container">

  <h1 class="display-6">Welcome to Restaurant Web Application</h1>

  <p>
    <span th:text="${greeting}"></span>, you!
  </p>

</div>
```

```
</body>
</html>
```

## เพิ่ม Controller

11. สร้าง controller package

12. เพิ่มคลาส HomeController ใน package controller

- สั่งเกต annotation `@Controller` Spring จะสร้าง object นี้ให้อัตโนมัติ และมีลักษณะเป็น controller ในโครงสร้าง MVC
- สั่งเกต annotation `@RequestMapping` จะต่อด้วย endpoint หมายความว่า ไม่ว่าจะเป็ REST function ใดที่เรียก endpoint "/" ก็จะมาที่เมทอดนี้
- เมทอดนี้ จะคืนค่าเป็นชื่อไฟล์ HTML template (ไม่ต้องมีนามสกุล .html) จึงมี return type เป็น String
  - คลาส Model จะเก็บข้อมูลต่าง ๆ ที่ controller จะส่งไปให้ view
  - ตัวแปร `greeting` จะต้องมืชื่อที่ตรงกับไฟล์ home.html
  - ในโค้ดนี้ จะคืนค่าเป็น home.html ซึ่งเป็น HTML template

```
package th.ac.ku.restaurant.controller;

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

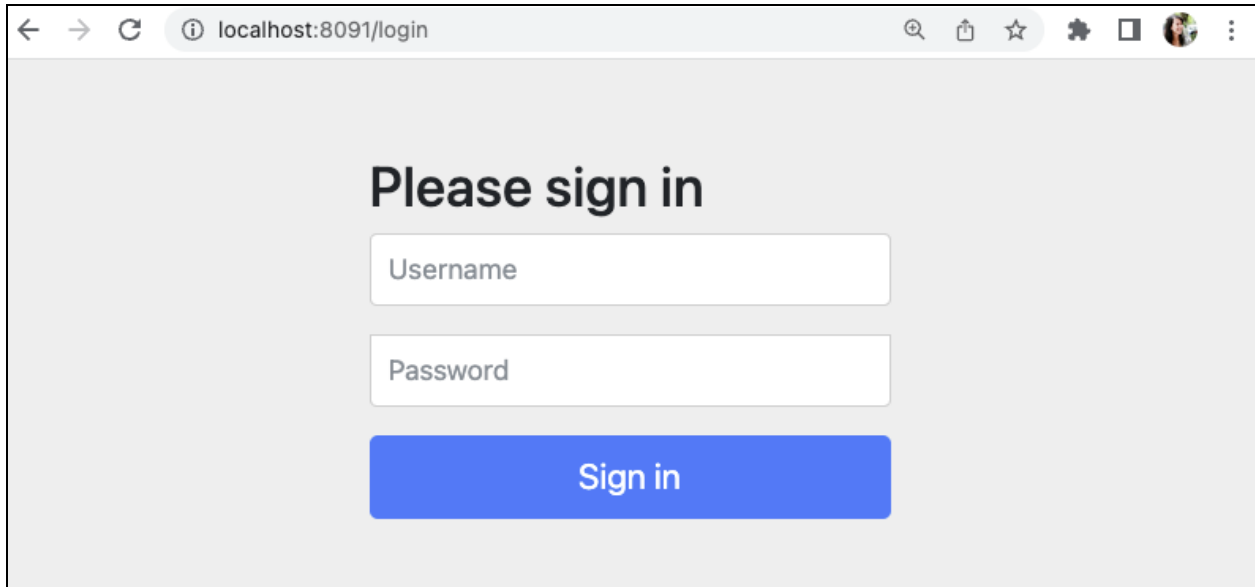
@Controller
public class HomeController {

    @RequestMapping("/")
    public String getHomePage(Model model) {
        model.addAttribute("greeting", "Sawaddee");
        // ต้องคืนค่าเป็นชื่อไฟล์ html template โดยในเมทอดนี้ คืนค่าเป็น home.html
        return "home";
    }
}
```

13. ลองรันโปรแกรม และไปที่หน้าเว็บ <http://localhost:8091/>

- จะถูก redirect ไปที่หน้า login
- หากไม่ได้ใช้ IntelliJ สามารถรันผ่าน command line ได้ ดังนี้

```
$ mvn spring-boot:run
```



#### 14. สร้าง security package

- สร้างคลาส SecurityConfig
- `.anyRequest().authenticated()` กำหนดว่า ต้อง login ก่อนมาที่หน้าใด ๆ

```
package th.ac.ku.restaurant.security;

import org.springframework.context.annotation.Configuration;
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;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

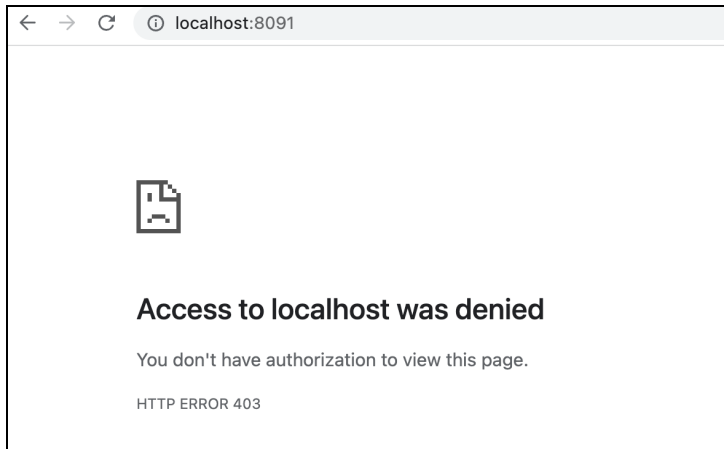
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .anyRequest().authenticated();
    }
}
```

#### 15. ลองรันโปรแกรม และไปที่หน้าเว็บ <http://localhost:8091/>

- จะได้หน้า error 403 Access denied

- หากไม่ได้ใช้ IntelliJ สามารถรันผ่าน command line ได้ ดังนี้

```
$ mvn spring-boot:run
```



#### 16. เพิ่มข้อยกเว้นสำหรับบางหน้าที่ไม่ต้อง login

- `.antMatchers("/", "/css/**", "/js/**").permitAll()` อนุญาตให้เข้าถึง path เหล่านี้ได้ โดยไม่ต้อง login

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/", "/css/**", "/js/**").permitAll()
            .anyRequest().authenticated();
    }
}
```

#### 17. ลองรันโปรแกรม และไปที่หน้าเว็บ <http://localhost:8091/> จะได้หน้า home



## II. เพิ่มการ sign-up

ในส่วนนี้ เราจะเพิ่มการ signup เพื่อเตรียมการลือคอิน โดยเราจะเพิ่ม ดังนี้

- สร้าง package model, repository
- เพิ่มคลาส User และ UserRepository ใน model, repository packages
- เพิ่ม signup.html ที่เป็น HTML template
- เพิ่มคลาส SignupController
- เพิ่มคลาส SignupService เพื่อช่วยประมวลผล user ก่อน save เข้าไปใน database รวมถึง การ hash password
- เพิ่มคลาส UserDetailsServiceImpl เพื่อทำ authentication
- ปรับ security configuration เพื่อให้เข้าหน้า signup ได้โดยไม่ต้อง login

### 1. เพิ่มคลาส User ใน model package

```
package th.ac.ku.restaurant.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import java.util.UUID;

@Entity
public class User {

    @Id
    @GeneratedValue
    private UUID id;

    private String username;
    private String password;
    private String firstName;
    private String lastName;
    private String role;

    // ให้ Generate..
    // - Getters และ Setters ทั้งหมด
}
```



## 2. เพิ่ม interface UserRepository ใน repository package

```
package th.ac.ku.restaurant.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import th.ac.ku.restaurant.model.User;
import java.util.UUID;

@Repository
public interface UserRepository extends JpaRepository<User, UUID> {

    // SELECT * FROM User WHERE username = 'username in parameter'
    User findByUsername(String username);
}
```

## 3. เพิ่มหน้า signup.html

- `name="..."` attribute ใน `<input ..>` ต้องตรงกับชื่อ instance variable ในคลาส User

```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Restaurant Web Application</title>
    <link th:rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
    <script th:src="@{/js/bootstrap.min.js}"></script>
</head>
<body>

<div class="container w-50 p-3">
    <h1 class="display-5">Sign Up</h1>

    <form action="#" th:action="@{/signup}" method="POST">

        <div id="success-msg" class="alert alert-success"
            th:if="${signupSuccess}">
            Successfully signed up! Please <a th:href="@{/login}">login</a>.
        </div>
        <div id="error-msg" class="alert alert-danger"
            th:if="${signupError}">
            <span th:text="${signupError}"></span>
        </div>

        <div class="mb-3">
            <label for="inputFirstName">First Name</label>
            <input id="inputFirstName" type="input" class="form-control"
                name="firstName">
        </div>
        <div class="mb-3">
            <label for="inputLastName">Last Name</label>
            <input id="inputLastName" type="input" class="form-control"
                name="lastName">
        </div>
    </form>
</div>
```

```

        <div class="mb-3">
            <label for="inputUsername">Username</label>
            <input id="inputUsername" type="input" class="form-control"
                name="username">
        </div>
        <div class="mb-3">
            <label for="inputPassword">Password</label>
            <input id="inputPassword" type="password" class="form-control"
                name="password">
        </div>
        <div class="mb-3">
            <label for="inputRole">Role</label>
            <select id="inputRole" class="form-select" name="role">
                <option value="ROLE_USER" selected>User</option>
                <option value="ROLE_ADMIN">Admin</option>
            </select>
        </div>
        <button id="submit-button" type="submit" class="btn btn-primary">Sign
Up</button>
    </form>
</div>

</body>
</html>

```

#### 4. เพิ่มคลาส SignupController class

```

package th.ac.ku.restaurant.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import th.ac.ku.restaurant.model.User;
import th.ac.ku.restaurant.service.SignupService;

@Controller
public class SignupController {

    @Autowired
    private SignupService signupService;

    @GetMapping("/signup")
    public String getSignupPage() {
        return "signup"; // return หน้าฟอร์ม signup.html
    }

    @PostMapping("/signup")
    public String signupUser(@ModelAttribute User user, Model model) {

```

```

        if (signupService.isUsernameAvailable(user.getUsername())) {
            signupService.createUser(user);
            model.addAttribute("signupSuccess", true);
        } else {
            model.addAttribute("signupError", "Username not available");
        }
        return "signup";
    }
}

```

## 5. สร้าง service package

## 6. เพิ่มคลาส SignupService เพื่อช่วยประมวลผล user ก่อน save เข้าไปใน database

- คลาสนี้ ทำให้เราสามารถแยกหน้าที่การประมวลผล user ออกจากคลาส SignupController โดย
  - SignupController รองรับ requests จาก users
  - SignupService ประมวลผล user ก่อน save เข้าไปใน database และหลังจากค้นคืนมาจาก database
- โดยคลาส SignupService นี้ จะทำการ hash password ก่อน save ลง database โดยใช้ PasswordEncoder object ที่มีการสร้าง Bean ไว้ใน security config

```

package th.ac.ku.restaurant.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import th.ac.ku.restaurant.model.User;
import th.ac.ku.restaurant.repository.UserRepository;

@Service
public class SignupService {

    @Autowired
    private UserRepository repository;

    @Autowired
    private PasswordEncoder passwordEncoder;

    public boolean isUsernameAvailable(String username) {
        return repository.findByUsername(username) == null;
    }

    public void createUser(User user) {
        User record = new User();
        record.setFirstName(user.getFirstName());
        record.setLastName(user.getLastName());
        record.setRole(user.getRole());
        record.setUsername(user.getUsername());

        String hashedPassword = passwordEncoder.encode(user.getPassword());
        record.setPassword(hashedPassword);
    }
}

```

```

        repository.save(record);
    }

    public User getUser(String username) {
        return repository.findByUsername(username);
    }
}

```

## 7. เพิ่มคลาส UserDetailsServiceImpl ใน service package เพื่อทำ authentication เมื่อ login

- org.springframework.security.core.userdetails.User ตรวจสอบ username, password ให้เรา และจะ hash password ใน login form จากนั้นจะเปรียบเทียบกับ password ที่ hash แล้วใน database

```

package th.ac.ku.restaurant.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
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 th.ac.ku.restaurant.model.User;
import th.ac.ku.restaurant.repository.UserRepository;

import java.util.ArrayList;
import java.util.List;

@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

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

        if (user == null) {
            throw new UsernameNotFoundException("Could not find user");
        }

        List<SimpleGrantedAuthority> authorities = new ArrayList<>();
        authorities.add(new SimpleGrantedAuthority(user.getRole()));

        return new org.springframework.security.core.userdetails.User(
            user.getUsername(), user.getPassword(), authorities);
    }
}

```

## 8. ปรับคลาส SecurityConfig และเพิ่มโค้ดที่ highlight เอาไว้

- UserDetailsServiceImpl จะทำ authentication
- BCryptPasswordEncoder เป็นการ hash แบบ bcrypt
  - อ่านเพิ่มเติมได้ที่
  - <https://docs.spring.io/spring-security/site/docs/current/api/org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder.html>
- @Bean บอก Spring ให้สร้าง object ให้เรา
- .ignoring().antMatchers("/h2-console/\*\*") อนุญาตให้เข้าถึง H2 database ได้ที่ url /h2-console

```
package th.ac.ku.restaurant.security;

import th.ac.ku.service.UserDetailsServiceImpl;
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.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
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.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsServiceImpl userDetailsService;

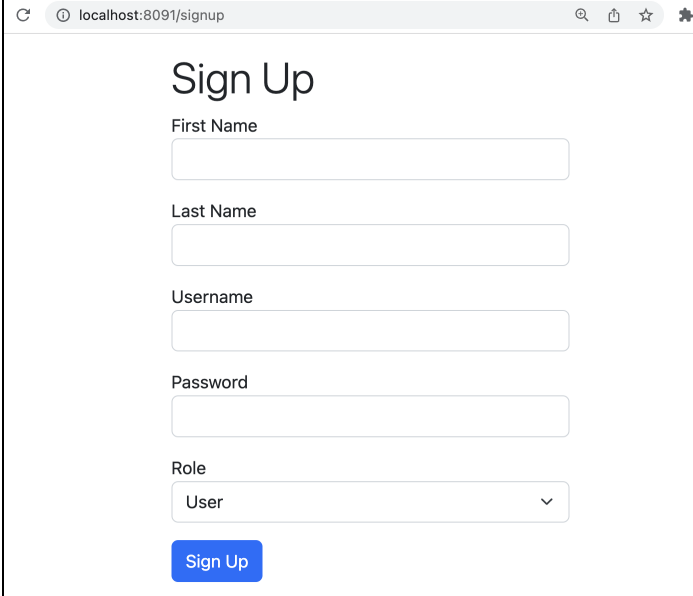
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/", "/signup",
                "/css/**", "/js/**").permitAll()
            .anyRequest().authenticated();
    }

    @Bean
    public PasswordEncoder encoder() {
        return new BCryptPasswordEncoder(12);
    }

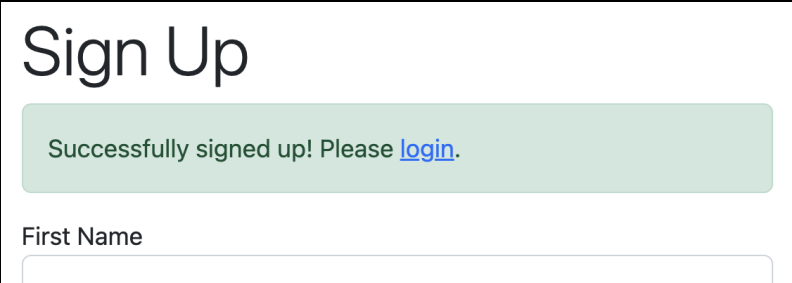
    @Override
    public void configure(WebSecurity web) throws Exception {
```

```
web
    .ignoring()
    .antMatchers("/h2-console/**");
}
```

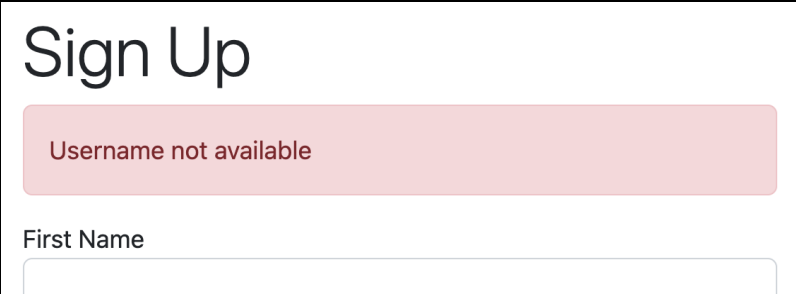
9. รันโปรแกรม และไปที่หน้า <http://localhost:8091/signup> และให้ลอง signup



A screenshot of a web browser window showing a "Sign Up" form. The browser's address bar displays "localhost:8091/signup". The form includes input fields for "First Name", "Last Name", "Username", and "Password", and a dropdown menu for "Role" with "User" selected. A blue "Sign Up" button is at the bottom.



A screenshot of the "Sign Up" form after a successful registration. A green message box states "Successfully signed up! Please [login](#)." Below the message, the "First Name" input field is visible.



A screenshot of the "Sign Up" form showing an error. A red message box displays "Username not available". The "First Name" input field is visible below the message.

### III. เพิ่มการ login

#### 1. เพิ่มหน้า login.html template.

- (ไม่ต้องมี logout template เนื่องจากจะให้ redirect ไปที่หน้า login หลังจาก logout แล้ว)

```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Restaurant Web Application</title>
  <link th:rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
  <script th:src="@{/js/bootstrap.min.js}"></script>
</head>
<body>

<div class="container w-50 p-3">
  <h1 class="display-5">Login</h1>

  <form th:action="@{/login}" method="POST">
    <div id="error-msg" th:if="${param.error}" class="alert alert-danger">
      Invalid username or password
    </div>
    <div id="logout-msg" th:if="${param.logout}" class="alert alert-success">
      You have been logged out
    </div>
    <div class="mb-3">
      <label for="inputUsername">Username</label>
      <input type="text" name="username" id="inputUsername"
        class="form-control" required>
    </div>
    <div class="mb-3">
      <label for="inputPassword">Password</label>
      <input type="password" name="password" id="inputPassword"
        class="form-control" required>
    </div>
    <button id="submit-button" type="submit" class="btn
btn-primary">Login</button>
  </form>
</div>

</body>
</html>
```

#### 2. เพิ่มคลาส controller สำหรับการ login/logout

```
package th.ac.ku.restaurant.controller;

import org.springframework.security.core.Authentication;
import
org.springframework.security.web.authentication.logout.SecurityContextLogoutHa
ndler;
```



```

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

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@Controller
public class AuthController {

    @GetMapping("/login")
    public String loginView() {
        return "login"; // return login.html
    }

    @GetMapping("/logout")
    public String logout(HttpServletRequest request,
                        HttpServletResponse response,
                        Authentication auth) {

        if (auth != null) {
            new SecurityContextLogoutHandler()
                .logout(request, response, auth);
        }
        // You can redirect wherever you want, but generally it's a good
        // practice to show the login screen again.
        return "redirect:/login?logout";
    }
}

```

### 3. ปรับคลาส SecurityConfig โดยเพิ่มโค้ดที่ highlight ไว้

- `.formLogin().loginPage("/login")` บอก Spring ให้ใช้ /login ในการล็อกอิน
- `.defaultSuccessUrl()` กำหนด path ที่ให้ Spring ไปเมื่อ login สำเร็จ
- `.logout().logoutUrl("/logout")` บอก Spring ให้ใช้ /logout ในการล็อกเอาท์

```

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsServiceImpl userDetailsService;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/", "/signup",
                        "/css/**", "/js/**").permitAll()
            .anyRequest().authenticated()

            .and()

```

```
        .formLogin().loginPage("/login")
        .defaultSuccessUrl("/", true)
        .permitAll()
    .and()
        .logout().logoutUrl("/logout")
        .clearAuthentication(true)
        .invalidateHttpSession(true)
        .deleteCookies("JSESSIONID", "remember-me")
        .permitAll();
}

@Bean
public PasswordEncoder encoder() {
    return new BCryptPasswordEncoder(12);
}

@Override
public void configure(WebSecurity web) throws Exception {
    web
        .ignoring()
        .antMatchers("/h2-console/**");
}
}
```

#### 4. เพิ่มการแสดงผลชื่อ user ในหน้า Home โดยการเพิ่มโค้ด ดังนี้

```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org"
      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity5">
<head>
  <meta charset="UTF-8">
  <title>Restaurant Web Application</title>
  <link th:rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
  <script th:src="@{/js/bootstrap.min.js}"></script>
</head>
<body>

<div class="container">
  <h1 class="display-6">Welcome to Restaurant Web Application</h1>

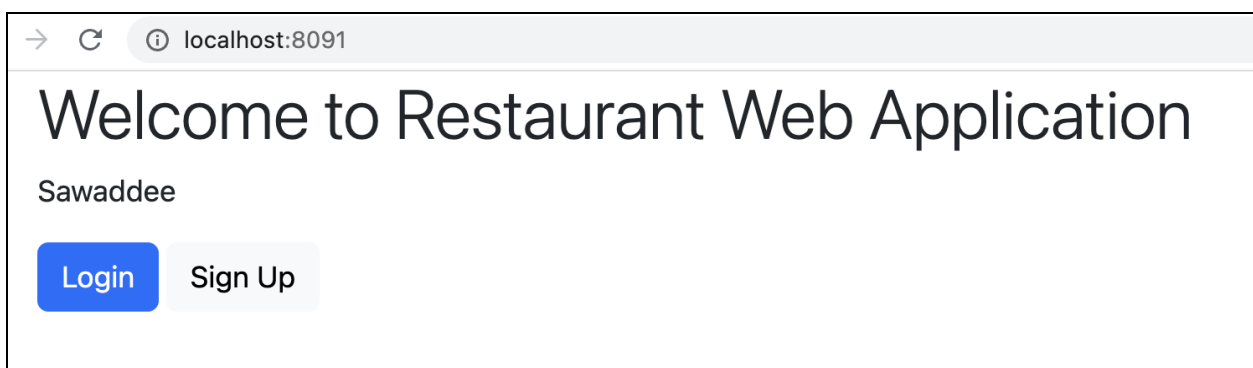
  <p><span th:text="${greeting}"></span></p>

  <div sec:authorize="isAuthenticated()">
    <p>Khun <b><span sec:authentication="name"></span></b></p>
  </div>
  <div sec:authorize="isAnonymous()">
    <a class="btn btn-primary" th:href="@{/login}">Login</a>
    <a class="btn btn-light" th:href="@{/signup}">Sign Up</a>
  </div>
</div>

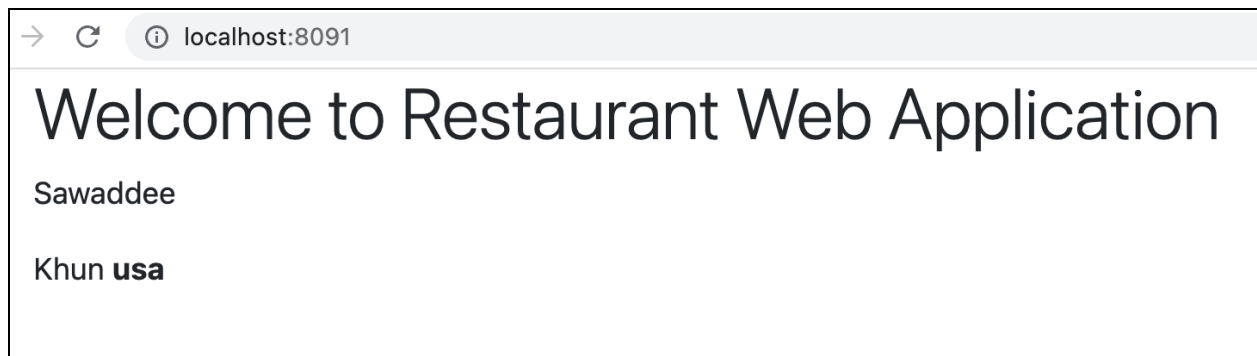
</body>
</html>
```

#### 5. รันโปรแกรม และไปหน้า home <http://localhost:8091/>

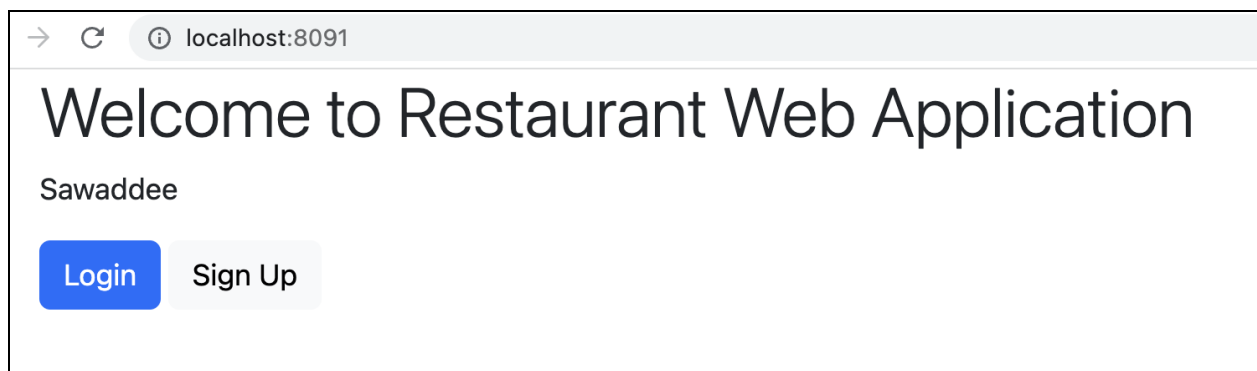
- ถ้ายังไม่ได้ login จะได้หน้า ดังนี้



- ถ้า login แล้ว จะได้หน้า ดังนี้ โดยจะแสดง username ให้นำหน้า home



- และลอง logout จะกลับมาหน้านี้ <http://localhost:8091/logout>



6. ลองดูข้อมูล user ใน database

- ถ้าใช้ H2 ให้มาที่หน้า: <http://localhost:8091/h2-console>
- ให้ป้อน JDBC URL ให้เป็น `jdbc:h2:mem:restaurant` และ login

The screenshot shows the H2 console Login dialog box. At the top, there is a language dropdown set to 'English' and links for 'Preferences', 'Tools', and 'Help'. The 'Login' section has a 'Saved Settings' dropdown set to 'Generic H2 (Embedded)'. Below it, the 'Setting Name' is also 'Generic H2 (Embedded)', with 'Save' and 'Remove' buttons. The 'Driver Class' is 'org.h2.Driver'. The 'JDBC URL' is 'jdbc:h2:mem:restaurant'. The 'User Name' is 'test', and the 'Password' field is empty. At the bottom are 'Connect' and 'Test Connection' buttons.

7. เมื่อเข้า database ได้ ทั้ง MySQL หรือ H2 ให้ลอง `SELECT * FROM User` ดู

- จะเห็นว่า password มีการ hash ด้วย bcrypt encoder แล้ว

The screenshot shows the H2 console interface. The left sidebar displays the database structure: 'jdbc:h2:mem:restaurant' (selected), 'USER', 'INFORMATION\_SCHEMA', 'Users', and 'H2 1.4.200 (2019-10-14)'. The main area shows the SQL statement 'SELECT \* FROM USER' entered in the 'SQL statement:' field. Below the statement, the results are displayed in a table format. The table has four columns: 'ID', 'FIRST\_NAME', 'LAST\_NAME', and 'PASSWORD'. The first row contains the values: '4d76e2b6f8b54c08af25be857baf8ddc', 'Usa', 'Sammapun', and '\$2a\$12\$i8.TJUGqc2wj16vJXBq8XOZ/'. Below the table, it indicates '(1 row, 24 ms)'.

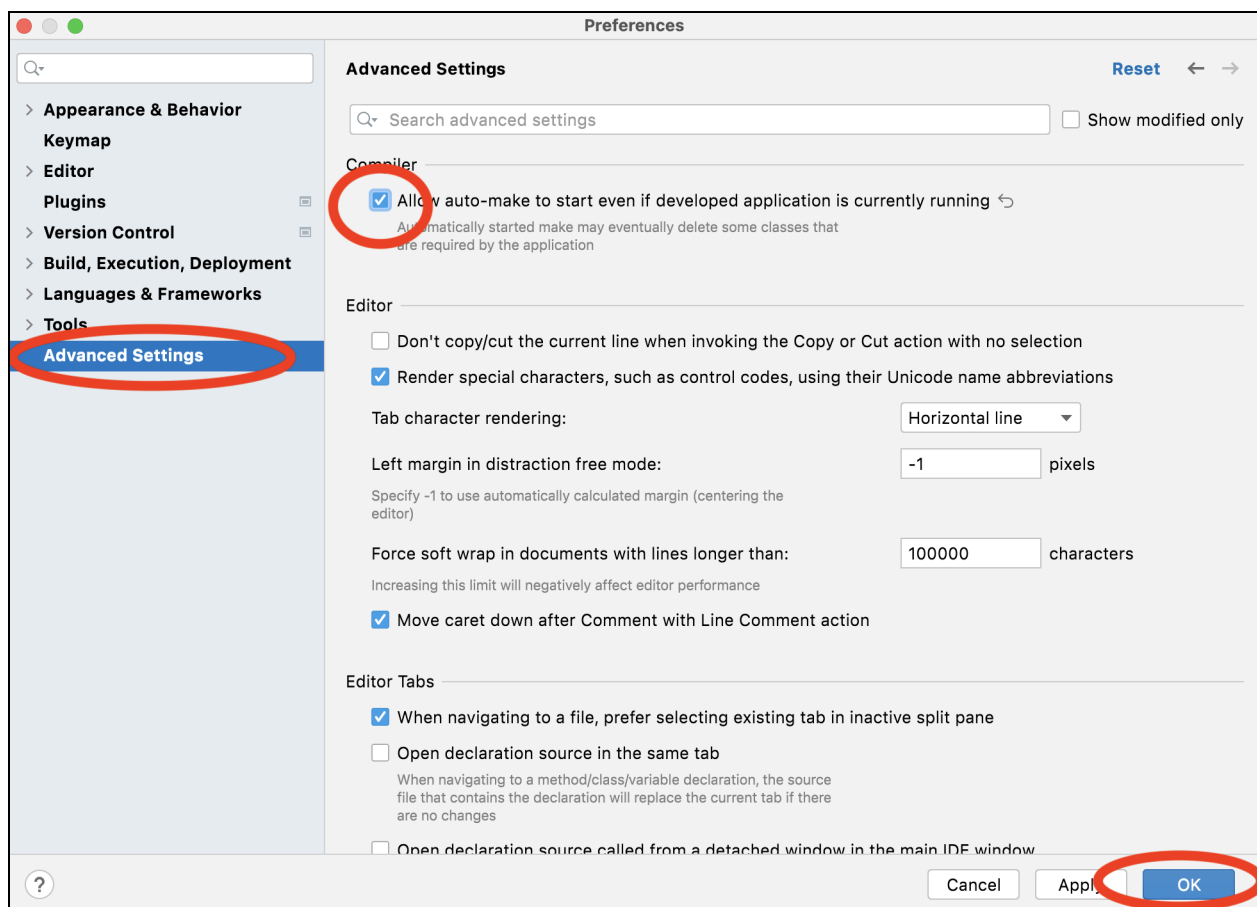
ID	FIRST_NAME	LAST_NAME	PASSWORD
4d76e2b6f8b54c08af25be857baf8ddc	Usa	Sammapun	\$2a\$12\$i8.TJUGqc2wj16vJXBq8XOZ/

(1 row, 24 ms)

## IV. วิธีทำให้ IntelliJ รัน live reload

คือ หากแก้โค้ดใด ๆ หรือแก้ไฟล์ .html จะทำให้ IntelliJ reload หรือ restart โปรแกรมให้เรารู้อัตโนมัติ โดยไม่ต้องกดปุ่ม Stop และ Start ค่ะ (แต่การ reload จะช้าๆ เป็นบางครั้ง)

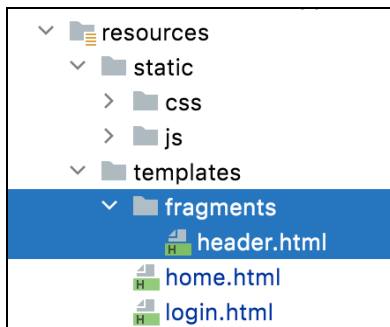
1. **Windows:** ไปที่เมนู File > Settings...
2. **MacOS:** ไปที่เมนู IntelliJ IDEA > Preferences
3. เลือกที่ Advanced Settings
4. ดึงที่ Allow auto-make to start .....
5. กดปุ่ม OK



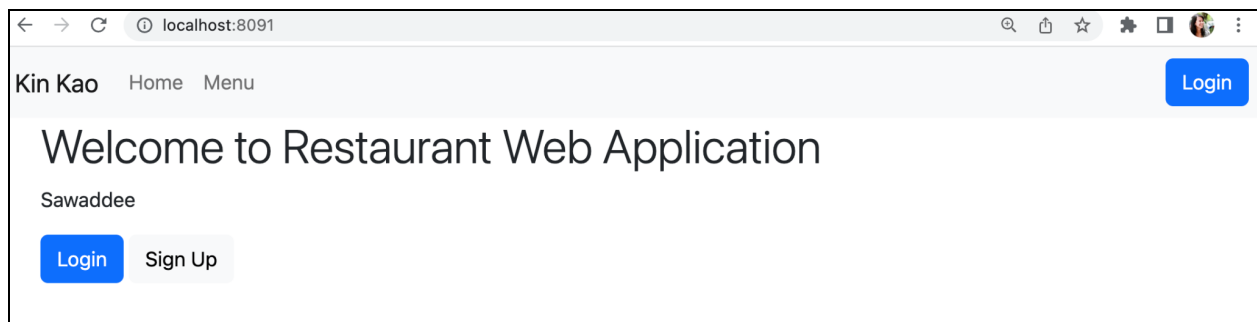
## V. จัดการ layout

### เพิ่ม header

- สร้าง folder ชื่อ fragments ใน templates/
- สร้างไฟล์ header.html ใน fragments folder



- เราจะเพิ่มเมนู navbar หน้าตาแบบนี้
- (เราทำ navbar menu เตรียมไว้สำหรับสัปดาห์หน้า)



```
<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org"
      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity5">
<head>
  <meta charset="UTF-8">
  <link th:rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
  <script th:src="@{/js/bootstrap.min.js}"></script>
</head>

<nav class="navbar navbar-expand-lg navbar-light bg-light"
  th:fragment="header">

  <div class="container-fluid">
    <a class="navbar-brand" th:href="@{/}">Kin Kao</a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
  data-bs-target="#navbarSupportedContent"
  aria-controls="navbarSupportedContent" aria-expanded="false"
```

```

aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
            <a class="nav-link" th:href="@{/}">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" th:href="@{/menu}">Menu</a>
        </li>
    </ul>
</div>
<div sec:authorize="isAuthenticated()">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
            <a class="nav-link" href="#">สวัสดี <b><span
sec:authentication="name"></span></b></a>
        </li>
        <li class="nav-item">
            <a class="btn btn-primary" th:href="@{/logout}">Logout</a>
        </li>
    </ul>
</div>
<div sec:authorize="isAnonymous()">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
            <a class="btn btn-primary" th:href="@{/login}">Login</a>
        </li>
    </ul>
</div>
</div>
</nav>

</nav>
</body>
</html>

```

## ปรับไฟล์ html ให้ใช้ header

### 8. ปรับไฟล์ .html ทุกหน้า

- เพิ่ม <div th:insert="fragments/header :: header"></div> ไว้บรรทัดแรกหลัง <body>

```

<!DOCTYPE html>
<html lang="en" xmlns:th="https://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">

```



```

<title>Restaurant Web Application</title>
<link th:rel="stylesheet" th:href="@{/css/bootstrap.min.css}">
<script th:src="@{/js/bootstrap.min.js}"></script>
</head>
<body>

<div th:insert="fragments/header :: header"></div>

<div class="container">
    ...
</div>

</body>
</html>

```

### 9. รีรันโปรแกรม และไปที่หน้าเว็บ จะเห็นเมนูอยู่ด้านบน

- ก่อน login จะมีปุ่มให้ login ได้
- หลัง login จะมีชื่อ username และปุ่ม logout

