[Thực hành] Lưu tên người dùng sử dụng Cookie

Mục tiêu

Luyện tập thao tác với Cookie trong ứng dụng Spring MVC.

Mô tả

Trong phần này, chúng ta sẽ phát triển một tính năng cho phép người dùng đăng nhập vào ứng dụng và ghi nhận lại tên của người dùng nếu đăng nhập thành công.

Nếu lần sau quay lại đăng nhập thì tên của người dùng sẽ được tự động điền vào.

Các bước hoạt động của tính năng này bao gồm:

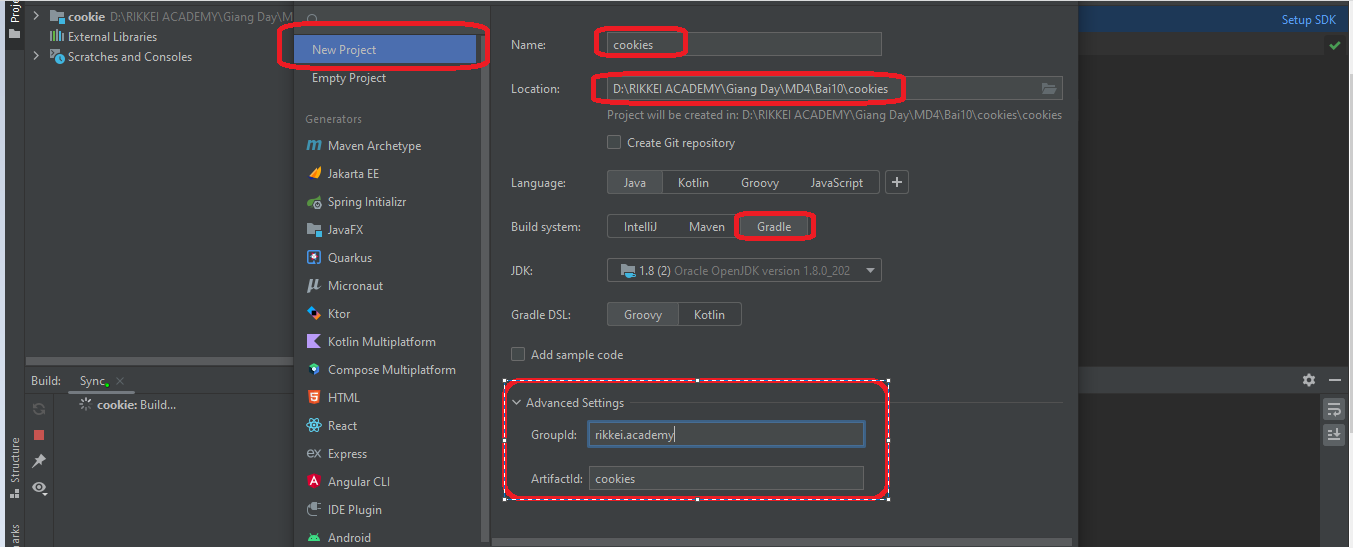
1. Người dùng đăng nhập vào hệ thống
2. Người dùng đăng xuất khỏi hệ thống (bằng cách nhấn nút đăng xuất hoặc tắt trình duyệt)
3. Người dùng mở trang đăng nhập, tên của người dùng đã được nhập sẵn

**Để hoàn thành bài thực hành, học viên cần:**

* Đưa mã nguồn lên GitHub

Hướng dẫn

* Tạo project với thư viện : gradle



* Thêm thư viện vào **build.gradle:**

compileOnly('javax.servlet:javax.servlet-api:4.0.1')  
implementation group: 'org.springframework', name: 'spring-webmvc', version: '5.3.2'  
implementation group: 'org.thymeleaf', name: 'thymeleaf-spring5', version: '3.0.11.RELEASE'

* Tạo package: **rikkei.academy.config:**
* Tạo class: **AppConfig** trong package: **config:**

package rikkei.academy;  
  
import org.springframework.beans.BeansException;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.thymeleaf.spring5.SpringTemplateEngine;  
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
import org.thymeleaf.templatemode.TemplateMode;  
@Configuration  
@EnableWebMvc  
@ComponentScan("rikkei.academy.controller")  
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware

{  
 private ApplicationContext applicationContext;  
  
 @Override  
 public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {  
 this.applicationContext = applicationContext;  
 }  
  
 //Cấu hình Thymleaf  
 @Bean  
 public SpringResourceTemplateResolver templateResolver() {  
 SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();  
 templateResolver.setApplicationContext(applicationContext);  
 templateResolver.setPrefix("/WEB-INF/views");  
 templateResolver.setSuffix(".html");  
 templateResolver.setTemplateMode(TemplateMode.*HTML*);  
 templateResolver.setCharacterEncoding("UTF-8");  
 return templateResolver;  
 }  
  
 @Bean  
 public SpringTemplateEngine templateEngine() {  
 SpringTemplateEngine templateEngine = new SpringTemplateEngine();  
 templateEngine.setTemplateResolver(templateResolver());  
 return templateEngine;  
 }  
  
 @Bean  
 public ThymeleafViewResolver viewResolver() {  
 ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();  
 viewResolver.setTemplateEngine(templateEngine());  
 viewResolver.setCharacterEncoding("UTF-8");  
 return viewResolver;  
 }  
  
}

* Tạo class: **AppInit** trong package: **config**

package rikkei.academy.config;  
  
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;  
  
public class AppInit extends AbstractAnnotationConfigDispatcherServletInitializer {  
 @Override  
 protected Class<?>[] getRootConfigClasses() {  
 return new Class[]{AppConfig.class};  
 }  
  
 @Override  
 protected Class<?>[] getServletConfigClasses() {  
 return new Class[0];  
 }  
  
 @Override  
 protected String[] getServletMappings() {  
 return new String[]{"/"};  
 }  
}

* Tạo package: **model** => Tạo class: **User** trong package này:
* package rikkei.academy.model;  
    
  public class User {  
   private String email;  
   private String password;  
    
   public User() {  
   }  
    
   public User(String email, String password) {  
   this.email = email;  
   this.password = password;  
    
   }  
    
   public String getEmail() {  
   return email;  
   }  
    
   public void setEmail(String email) {  
   this.email = email;  
   }  
    
   public String getPassword() {  
   return password;  
   }  
    
   public void setPassword(String password) {  
   this.password = password;  
   }  
    
    
  }

Trong class User, chúng ta khai báo thuộc tính email, password sử dụng để điền thông tin vào form đăng nhập.

* Tạo package: **controller** => Tạo class: **UserController** trong package này:

Trong đó:

Annotation @SessionAttributes("user") được sử dụng để lưu trữ thông tin của model attribute có tên là user.

Đoạn mã sau đây sẽ add User vào trong model attribute:

@ModelAttribute("user")  
public User setUpUserForm() {  
 return new User();  
}

Annotation @ModelAttribute("user") sẽ nhận user trả về từ view, sau đó đưa vào session:

@ModelAttribute("user")

model sẽ trả message về cho view trong trường hợp login thành công hoặc thất bại.

Annotation @CookieValue để ràng buộc giá trị của cookie HTTP với tham số phương thức trong controller.

response sẽ trả cookie về cho view sử dụng phương thức: response.addCookie(cookie);

if (user.getEmail().equals("admin@gmail.com") && user.getPassword().equals("12345")) {  
 if (user.getEmail() != null)  
 setUser = user.getEmail();  
  
 // create cookie and set it in response  
 Cookie cookie = new Cookie("setUser", setUser);  
 cookie.setMaxAge(24 \* 60 \* 60);  
 response.addCookie(cookie);

Đoạn mã trên sẽ so sánh email truyền vào có bằng với "admin@gmail.com" và password truyền vào có bằng với "12345" hay không. Nếu đúng sẽ thực hiện các công việc sau:

* Gán email truyền vào cho biến setUser
* Tạo cookie và trả về cho client

Cookie[] cookies = request.getCookies();  
//iterate each cookie  
for (Cookie ck : cookies) {  
 //display only the cookie with the name 'setUser'  
 if (ck.getName().equals("setUser")) {  
 model.addAttribute("cookieValue", ck);  
 break;  
 } else {  
 ck.setValue("");  
 model.addAttribute("cookieValue", ck);  
 break;  
 }  
}

Đoạn mã trên sẽ duyệt danh sách cookie và lấy cookie có tên 'setUser' sau đó truyền vào model.

* Tạo **webapp/WEB-INF/views** dưới thư mục **main: (**Chú ý: trước khi tạo thêm id ‘war’ vào build.gradle)

plugins **{** id 'java'  
 id 'war'  
**}**

* Tạo file: **login.html** trong thư mục **views:**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">  
 <title>Cookie Demo</title>  
</head>  
<body>  
<h1>User Login</h1>  
<form action="#" th:action="@{/dologin}" th:object="${user}" method="post">  
 <table border="0">  
 <tr>  
 <td><label>Email</label></td>  
 <td><input type="text" id="email" name="email"  
 th:value="${cookieValue.value}"/></td>  
 </tr>  
 <tr>  
 <td><label>Password</label></td>  
 <td><input type="password" th:field="\*{password}"/></td>  
 </tr>  
 </table>  
 <input type="submit" value="Login"/>  
</form>  
<br>  
<span th:text="${message}"></span>  
<span th:text="${user.email}"></span>  
</body>  
</html>

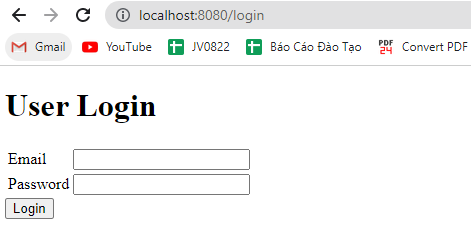
Cú pháp gán giá trị cookie vào textbox như sau:

<input type="text" id="email" name="email"  
 th:value="${cookieValue.value}"/>

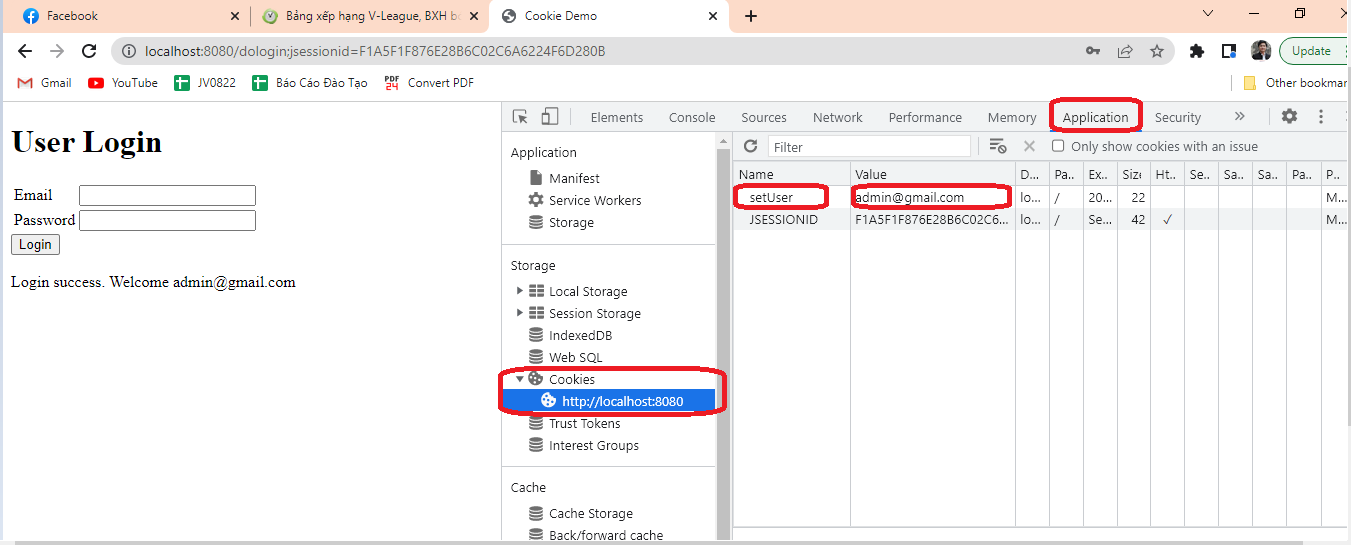
Trong đó:

* th:value = "${cookieValue.value}" để gán giá trị cookie có tên 'setUser' cho value của textbox
* Add Tomcat và chạy ứng dụng:

Chạy ứng dụng, nhập vào thanh địa chỉ như sau: http://localhost:8080/login

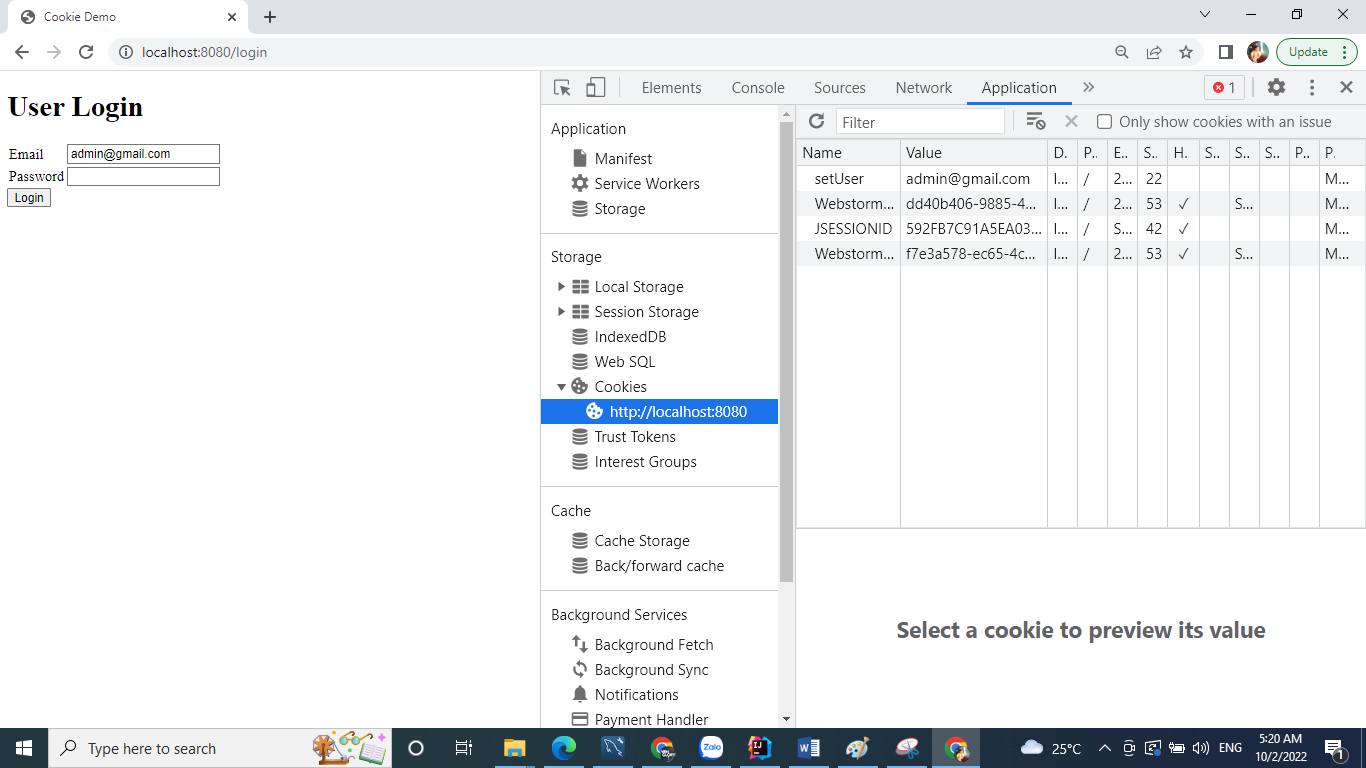


Nhập Email: admin@gmail.com. Password: 12345 và nhấn Login.



* Quan sát cách lưu cookie ở trên trình duyệt web.

Tắt trình duyệt, sau đó bật lại trình duyệt và truy cập lại vào: http://localhost:8080/login



Email đã được nhập sẵn trong textbox.

* ***Khác với session, cookie được lưu ở phía trình duyệt và cho dù chúng ta có tắt trình duyệt thì cookie vẫn tồn tại, nó chỉ mất khi chúng ta xóa nó hoặc set thời gian tồn tại cho cookie (Theo Chính NĐ)***

### https://github.com/nguyendongminhtri/MD4-Bai10-TH2-Cookie.git

### Hướng dẫn nộp bài:

Up code lên github.

Paste link github vào phần nộp bài.

### Grading summary