# Cơ bản tích hợp Thymeleaf + Spring Security

*Được viết bởi Jose Miguel Samper <jmiguelsamper AT users.sourceforge.net>*

Bạn đã chuyển sang Thymeleaf nhưng các trang đăng nhập và lỗi của bạn vẫn đang sử dụng JSP? Trong bài viết này, chúng tôi sẽ xem cách định cấu hình ứng dụng Spring của bạn để sử dụng Thymeleaf cho các trang đăng nhập và lỗi.

Tất cả các mã nhìn thấy ở đây đến từ một ứng dụng làm việc. Bạn có thể xem hoặc tải xuống mã nguồn từ repo GitHub của nó .

**Lưu ý** rằng các gói tích hợp Thymeleaf cho Spring Security hỗ trợ cả ứng dụng Spring MVC và Spring WebFlux kể từ Spring Security 5, nhưng bài viết này sẽ tập trung vào cấu hình Spring MVC.

## 1. Điều kiện tiên quyết

Chúng tôi cho rằng bạn đã quen thuộc với Thymeleaf và Spring Security và bạn có một ứng dụng hoạt động bằng các công nghệ này. Nếu bạn không biết Spring Security, bạn có thể thích đọc Tài liệu bảo mật mùa xuân .

## 2. Trang đăng nhập

Với Spring Security, bạn có thể chỉ định bất kỳ URL nào hoạt động như một trang đăng nhập, giống như:

```
@Override
protected void configure(final HttpSecurity http) throws Exception {
    http
        .formLogin()
        .loginPage("/login.html")
        .failureUrl("/login-error.html")
      .and()
        .logout()
        .logoutSuccessUrl("/index.html");
}
```

Bây giờ chúng ta phải khớp các trang này trong Bộ điều khiển mùa xuân:

```
@Controller
public class MainController {
```

```
  ...

  // Login form
  @RequestMapping("/login.html")
  public String login() {
    return "login.html";
  }

  // Login form with error
  @RequestMapping("/login-error.html")
  public String loginError(Model model) {
    model.addAttribute("loginError", true);
    return "login.html";
  }

}
```

Lưu ý rằng chúng tôi đang sử dụng cùng một mẫu **login.html** cho cả hai trang, nhưng khi có lỗi, chúng tôi đặt thuộc tính boolean vào mô hình.

Mẫu **login.html** của chúng tôi như sau:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Login page</title>
  </head>
  <body>
    <h1>Login page</h1>
    <p th:if="${loginError}" class="error">Wrong user or password</p>
    <form th:action="@{/login.html}" method="post">
      <label for="username">Username</label>:
      <input type="text" id="username" name="username" autofocus="autofocus" /> <br />
      <label for="password">Password</label>:
      <input type="password" id="password" name="password" /> <br />
      <input type="submit" value="Log in" />
    </form>
  </body>
</html>
```

## *3. Trang lỗi*

Chúng tôi cũng có thể định cấu hình trang lỗi dựa trên Thymeleaf. Trong trường hợp này, Spring Security hoàn toàn không liên quan, chúng ta chỉ cần thêm ExceptionHandler vào cấu hình Spring của mình như sau:

```
@ControllerAdvice
public class ErrorController {

    private static Logger logger = LoggerFactory.getLogger(ErrorController.class);
```

```
    @ExceptionHandler(Throwable.class)
    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
    public String exception(final Throwable throwable, final Model model) {
        logger.error("Exception during execution of SpringSecurity application", throwable);
        String errorMessage = (throwable != null ? throwable.getMessage() : "Unknown error");
        model.addAttribute("errorMessage", errorMessage);
        return "error";
    }

}
```

Mẫu **error.html** có thể giống như:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>Error page</title>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="css/main.css" th:href="@{/css/main.css}" />
    </head>
    <body th:with="httpStatus=${T(org.springframework.http.HttpStatus).valueOf(#response.status)}">
        <h1 th:text="|${httpStatus} - ${httpStatus.reasonPhrase}|">404</h1>
        <p th:utext="${errorMessage}">Error java.lang.NullPointerException</p>
        <a href="index.html" th:href="@{/index.html}">Back to Home Page</a>
    </body>
</html>
```

Lưu ý cách chúng tôi sử dụng `HttpStatus` enum của Spring để có được thông tin chi tiết về trạng thái phản hồi đã được đặt (trong trường hợp này sẽ luôn như vậy `500` , nhưng điều này cho phép chúng tôi sử dụng điều này `error.html` trong các tình huống báo cáo lỗi khác).

## *4. Phương ngữ bảo mật mùa xuân*

Trong môi trường Spring MVC, mô-đun tích hợp Spring Security hoạt động như một sự thay thế của taglib bảo mật Spring .

Chúng tôi sử dụng phương ngữ này trong ví dụ để in thông tin đăng nhập của người dùng và để hiển thị nội dung khác nhau cho các vai trò khác nhau.

Các **sec: ủy quyền cho** thuộc tính làm cho nội dung của nó khi các biểu hiện thuộc tính được đánh giá để **đúng** :

```
<div sec:authorize="isAuthenticated()">
  This content is only shown to authenticated users.
</div>
<div sec:authorize="hasRole('ROLE_ADMIN')">
  This content is only shown to administrators.
</div>
<div sec:authorize="hasRole('ROLE_USER')">
  This content is only shown to users.
</div>
```

Các **sec: xác thực** thuộc tính được sử dụng để in tên người dùng đăng nhập và vai trò:

```
Logged user: <span sec:authentication="name">Bob</span>
Roles: <span sec:authentication="principal.authorities">[ROLE_USER, ROLE_ADMIN]</span>
```

*Trên trang web này*

Trang Chủ
Tải xuống
Tài liệu
Hệ sinh thái
Câu hỏi thường gặp

Theo dõi vấn đề
Nhóm Thymeleaf
Ai đang sử dụng Thymeleaf?

*liên kết ngoại*

Diễn đàn
theo dõi chúng tôi trên Twitter
Ngã ba chúng tôi trên GitHub