
Bài 19

Session và Cookie trong Spring MVC

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài "Validation"

Tóm tắt lại các phần đã học từ bài "Validation"

- Trình bày được cơ chế quản lý cookie trong Spring
- Sử dụng được cookie trong Spring
- Trình bày được cơ chế quản lý session trong Spring
- Sử dụng được session trong Spring

Thảo

luận
Annotation @CookieValue
Ví dụ sử dụng Cookie trong Spring

- Annotation @CookieValue sử dụng để truy cập dữ liệu được đặt trong bất kỳ http cookie nào.
- Annotation @CookieValue có thể được sử dụng trong tham số phương thức của controller.
- Annotation @CookieValue sẽ tự động liên kết giá trị cookie với tham số phương thức

Annotation @CookieValue – Cú pháp

Cú pháp:

`(@CookieValue("cookieName") DataType`

`paramName)` Trong đó:

- `cookieName` là tên của cookie
- `DataType` là kiểu dữ liệu của tham số
- `paramName` là tên của tham số

Annotation @CookieValue – Ví dụ

Ví dụ:

```
public String hello(@CookieValue("hitCounter")  
Long countCookie) { //... }
```

Nếu không tìm thấy http cookie có tên "hitCounter", Spring sẽ liên kết tham số với giá trị mặc định. Cú pháp như sau:

```
public String hello(@CookieValue("hitCounter",  
defaultValue="0") Long countCookie) { ... }
```

Thiết lập http cookie trong Spring MVC

- Phương thức `addCookie()` của lớp `HttpServletResponse` sử dụng để thiết lập một cookie trong một ứng dụng dựa trên Spring MVC.
- Ví dụ:

```
public String hello(HttpServletResponse response) {  
    Cookie cookie = new Cookie("hitCounter", countCookie.toString());  
    response.addCookie(cookie);  
}
```


Thiết lập thời gian còn hiệu u

lực

- Phương thức `setMaxAge` xác định thời gian Cookie còn hiệu lực.
- Ví dụ sau sẽ thiết lập một Cookie trong 24 giờ:
`cookie.setMaxAge(60*60*24);`
- Trong đó:
`setMaxAge(seconds);`

Lấy giá trị của cookie

- Ví dụ:

```
<h2 th:text="${cookie.getValue()}"></h2>
```

Trong đó:

- `${cookie.getValue()}` sẽ lấy giá trị của cookie.

Xoá Cookie

- Xóa cookie theo 3 bước như sau:
 - Đọc một cookie đang tồn tại và lưu trong đối tượng Cookie.
 - Thiết lập tuổi của cookie về 0 sử dụng phương thức **setMaxAge()**.
 - Thêm cookie trở lại bên trong trường Response header.

Xoá Cookie – Ví dụ

```
Cookie cookie = null;
Cookie[] cookies = null;

cookies = request.getCookies();
if (cookies != null) {
    for (int i = 0; i < cookies.length; i++) {
        cookie = cookies[i];
        if (cookie.getName().compareTo("setUser") == 0) {
            cookie.setMaxAge(0);
            response.addCookie(cookie);
        }
    }
}
```

Cookie - Ví dụ: Đếm số lần truy cập trang



web
controller:

```
@Controller
public class HelloController {
    @RequestMapping("/hello")
    public String hello(
        @CookieValue(value = "hitCounter", defaultValue = "0") Long hitCounter,
        HttpServletResponse response, HttpServletRequest request, Model model) {

        // increment hit counter
        hitCounter++;

        // create cookie and set it in response
        Cookie cookie = new Cookie("hitCounter", hitCounter.toString());
        cookie.setMaxAge(24 * 60 * 60);
        response.addCookie(cookie);

        //get all cookies
        Cookie[] cookies = request.getCookies();
        //iterate each cookie
        for (Cookie ck : cookies) {
            //display only the cookie with the name 'hitCounter'
            if (ck.getName().equals("hitCounter")) {
                model.addAttribute("cookieValue", ck);
                break;
            }
        }
        return "hello";
    }
}
```

Cookie - Ví dụ: Đếm số lần truy cập trang web

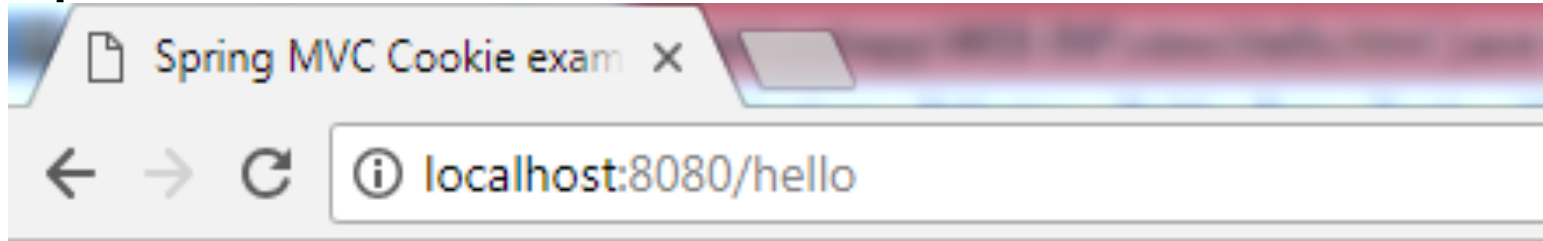


View:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Spring MVC Cookie example</title>
</head>
<body>
<h1>Page hit counter:</h1>
<h2 th:text="${cookieValue.value}"></h2>
</body>
</html>
```

Cookie - Ví dụ: Đếm số lần truy cập trang web

Chạy ứng dụng, refresh trang 2 – 3 lần và quan sát kết quả:



Page hit counter:

5

Demo

Quản lý cookie

Thảo luận

Cơ chế quản lý session trong Spring

Ví dụ sử dụng session trong Spring

- Session được gọi là một phiên làm việc giữa client và server.
- Một session bắt đầu khi client thực hiện request đầu tiên đến server và kết thúc khi client dừng làm việc với server.
- Thông thường chúng ta sử dụng Session để lưu thông tin đăng nhập, giỏ hàng hoặc những dữ liệu mang tính chất tạm thời và mỗi client sẽ có dữ liệu khác nhau.
- Session tồn tại khi các yêu cầu sau đây còn thoả mãn:
 - User không đóng browser
 - User không log-out khỏi website
 - Nếu user không tương tác gì với Website thì sau một khoảng thời gian thường 30 phút (do admin ấn định bên server), session sẽ bị đóng.

session

- Phương pháp 1: Inject trong HttpSession nơi được yêu cầu.
- Phương pháp 2: Sử dụng như một tham số
- Phương pháp 3: Tạo một bean và giới hạn cho session làm việc.
- Phương pháp 4: Sử dụng annotation @SessionAttributes.

Session - Inject trong HttpSession nơi được yêu cầu.

Ví dụ:

```
@Service
public class ShoppingCartService {

    @Autowired
    private HttpSession httpSession;

    ...
}
```

Ví dụ:

```
@Controller
public class ShoppingCartController {
    @RequestMapping("/addToCart")
    public String addToCart(long productId, HttpSession httpSession) {
        //do something with the httpSession
    }
}
```

Session - Tạo một bean và giới hạn cho session làm việc.

Ví dụ:

```
@Component
@Scope(proxyMode=ScopedProxyMode.TARGET_CLASS, value="session")
public class ShoppingCart implements Serializable{

    ...
}
```

Ví dụ:

```
@SessionAttributes("shoppingCart")
public class OrderFlowController {
    public String step1(@ModelAttribute("shoppingCart") ShoppingCart shoppingCart) {
        //...
    }
}
```

Session - Ví dụ: Ứng dụng đăng nhập

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; }  
}
```


Session - Ví dụ: Ứng dụng đăng nhập

Controller:

```
@Controller
@SessionAttributes("user")
public class LoginController {
    @ModelAttribute("user")
    public User setUpUserForm() { return new User(); }

    @GetMapping("/")
    public String Index() { return "index"; }

    @PostMapping("/dologin")
    public String doLogin(@ModelAttribute("user") User user, Model model) {
        if (user.getEmail().equals("admin@example.com")
        && user.getPassword().equals("12345")) {
            model.addAttribute("message", "Login success. Welcome: ");
        } else {
            model.addAttribute("message", "Login failed. Try again.");
        }
        return "index";
    }
}
```

Session - Ví dụ: Ứng dụng đăng nhập

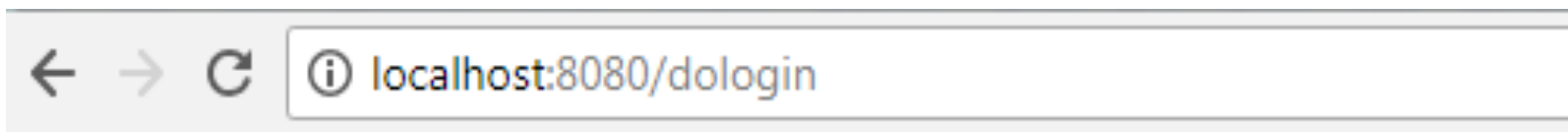


View

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

Session - Ví dụ: Ứng dụng đăng nhập

Kết quả khi chạy chương trình và đăng nhập thành công:



← → ↻ ⓘ localhost:8080/dologin

User Login

Email

Password

Login

Login success. Welcome: admin@example.com

Demo

Quản lý session

- Annotation `@CookieValue` sử dụng để truy cập dữ liệu được đặt trong bất kỳ http cookie nào.
- Session được coi là một phiên làm việc giữa client và server.
- Thông thường chúng ta sử dụng Session để lưu thông tin đăng nhập, giỏ hàng hoặc những dữ liệu mang tính chất tạm thời và mỗi client sẽ có dữ liệu khác nhau.
- Các cơ chế quản lý session trong Spring:
 - Inject trong `HttpSession` nơi được yêu cầu.
 - Sử dụng như một tham số.
 - Tạo một bean và giới hạn cho session làm việc.
 - Sử dụng annotation `@SessionAttributes`.

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: Web service và RESTful web service