Fpt University

FPT POLYTECHNIC

LẬP TRÌNH JAVA 5

BÀI 2: CONTROLLER

www.poly.edu.vn

Mục tiêu

- Sử dụng thành thạo @RequestMaping
  - Ánh xạ nhiều action
  - Ánh xạ phân biệt POST|GET
  - Ánh xạ phân biệt tham số
- Nắm vững phương pháp nhận tham số
  - Sử dụng HttpServletRequest
  - Sử dụng @RequestParam
  - Sử dụng JavaBean
  - Sử dụng @PathVariable để nhận dữ liệu từ URL
- Sử dụng @CookieValue để nhận cookie
- Hiểu rõ kết quả của phương thức action

@RequestMapping (1)

- Annotation **@RequestMapping** được sử dụng để ánh xạ một action đến một phương thức action trong Controller

```
@RequestMapping("say-hello")
public String sayHello() {
        return "hello";
}
```

- Khi người dùng đưa ra yêu cầu **say-hello.htm** thì phương thức action sayHello() sẽ thực hiện
- Trong một lớp @Controller có thể chứa nhiều phương thức action.

@RequestMapping (2)

- @RequestMapping("say-hello") là cách viết thu gọn của @RequestMapping(**value=**"say-hello")
- @RequestMapping() có thể được sử dụng để **đặt trên lớp Controller** để ánh xạ chung cho nhiều action method

```
@Controller
@RequestMapping("/home/")
public class HomeController{
        @RequestMapping("index")
        public String index() {
                return "home/index";
        }
        @RequestMapping("about")
        public String about() {
                return "home/about";
        }
}
```

home/index.htm

home/about.htm

@RequestMapping (3)

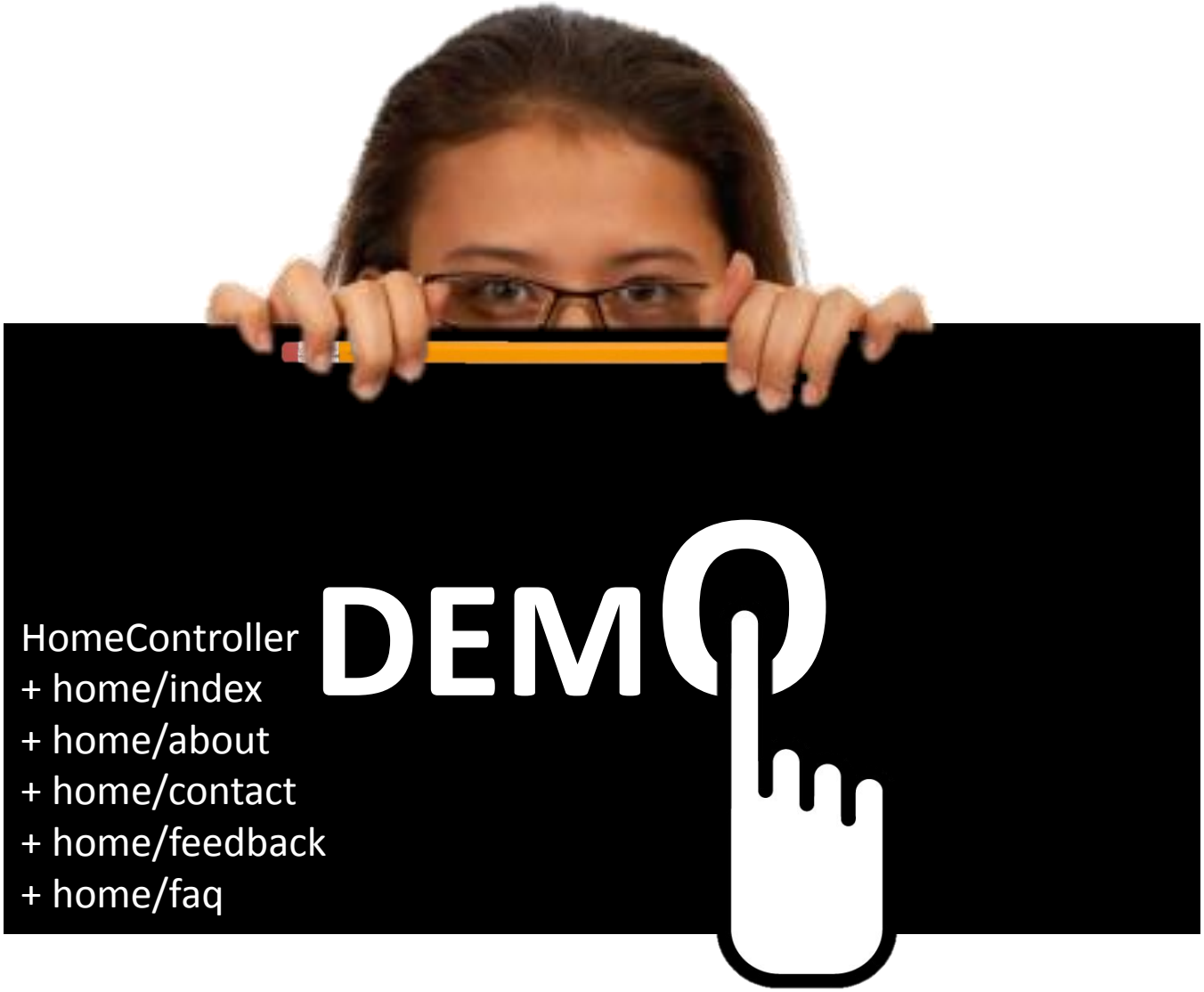
```
@Controller
public class HomeController{
        @RequestMapping("/home/index")
        public String index() {
                return "home/index";
        }
        @RequestMapping("/home/about")
        public String about() {
                return "home/about";
        }
}
```

home/index.htm

home/about.htm

```
@Controller
@RequestMapping("/home/")
public class HomeController{
        @RequestMapping("index")
        public String index() {
                return "home/index";
        }
        @RequestMapping("about")
        public String about() {
                return "home/about";
        }
}
```

- Hai cách ánh xạ này hoàn toàn tương đương nhau


HomeController
+ home/index
+ home/about
+ home/contact
+ home/feedback
+ home/faq
DEMO

PHÂN BIỆT POST|GET (1)

- Trong Servlet khi yêu cầu từ người dùng gửi đến server với phương thức web là GET thì phương thức doGet() của Servlet được thực hiện, ngược lại nếu phương thức web là POST thì doPost() được thực hiện
- Chú ý:
  - Trường hợp POST duy nhất là khi bạn submit một form có thuộc tính method="POST".
  - Các trường hợp GET thường gặp
    - Nhập url vào ô địa chỉ của trình duyệt web
    - Nhấp vào liên kết
    - Submit form với method="GET"

PHÂN BIỆT POST|GET (2)

- Trong Spring MVC phân biệt POST|GET thông qua tham số method của phương thức action

```
@RequestMapping(value="login", method=RequestMethod.GET)
public String login() {
        return "user/login";
}

@RequestMapping(value="login", method=RequestMethod.POST)
public String login(ModelMap model, HttpServletRequest request) {
        return "user/login";
}
```

- Như vậy khi yêu cầu user/login.htm được gửi đến server, Spring MVC sẽ gọi phương thức login() nào là tùy thuộc vào phương thức web **GET** hay **POST**

PHÂN BIỆT POST|GET (3)

- Thông thường GET là để vào giao diện còn POST được sử dụng để xử lý các nút chức năng

```
@RequestMapping(value="login", method=RequestMethod.GET)
public String login() {
        return "user/login";
}

@RequestMapping(value="login", method=RequestMethod.POST)
public String login(ModelMap model, HttpServletRequest request) {
        String id = request.getParameter("id");
        String pw = request.getParameter("password");
        if(id.equals("fpt") && pw.equals("polytechnic")){
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        else{
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        return "user/login";
}
```


DEMO
UserController
+ GET: user/login
+ POST: user/login

PHÂN BIỆT THAM SỐ (1)

- Trong Spring MVC không những hỗ trợ gọi phương thức action phân biệt theo phương thức web mà còn cho phép phân biệt theo tham số truyền theo

```
@RequestMapping(value="say-hello", params="mvc")
public String sayHello() {
        return "hello";
}
```

say-hello.htm?mvc

- Với định nghĩa này khi gọi say-hello.htm phải có tham số mvc thì phương thức sayHello() mới được thực hiện

PHÂN BIỆT THAM SỐ (2)

- Khi yêu câu student.htm thì phương thức nào sẽ thực hiện?

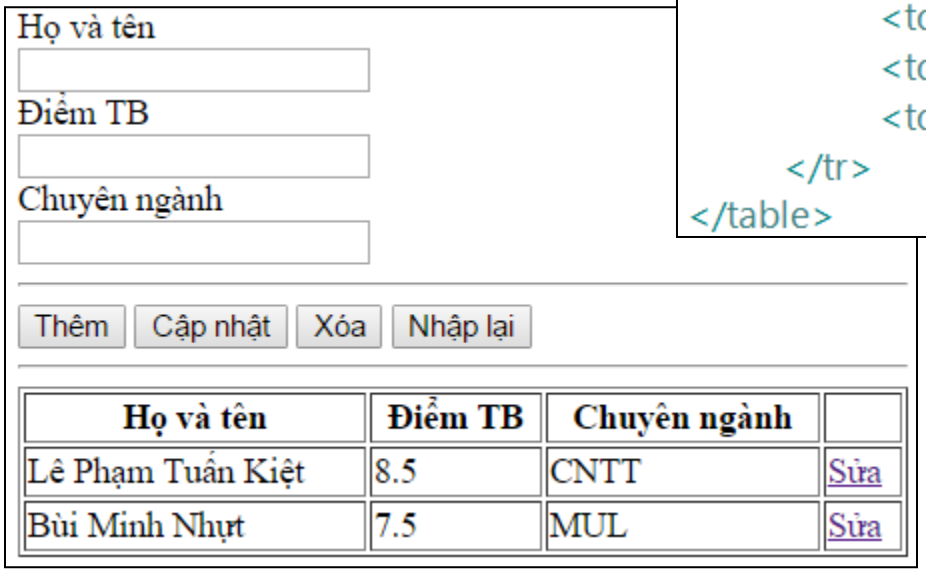
```
@Controller
@RequestMapping("student")
public class StudentController {
    @RequestMapping()
    public String index(ModelMap model) {...}

    @RequestMapping(params="btnInsert")
    public String insert(ModelMap model) {...}

    @RequestMapping(params="btnUpdate")
    public String update(ModelMap model) {...}

    @RequestMapping(params="btnDelete")
    public String delete(ModelMap model) {...}

    @RequestMapping(params="lnkEdit")
    public String edit(ModelMap model) {...}
}
```

- Nếu có tham số **btnInsert**, **btnUpdate**, **btnDelete** hoặc **lnkEdit** thì các phương thức **insert**(), **update**(), **delete**() hoặc **edit**() sẽ thực hiện
- Nếu không có các tham số trên thì **index**() sẽ thực hiện
- Nếu có nhiều hơn 1 tham số trên thì sẽ báo lỗi

TRANG SAU SẼ GỌI STUDENT.HTM

- [Thêm]=>insert()
- [Cập nhật]=>update()
- [Xóa]=>delete()
- [Nhập lại]=>index()
- [Sửa]=>edit()

```
<form action="student.htm" method="post">
        ...
        <button name="btnInsert">Thêm</button>
        <button name="btnUpdate">Cập nhật</button>
        <button name="btnDelete">Xóa</button>
        <button name="btnReset">Nhập lại</button>
</form>
<hr>
<table border="1" style="width:100%">
    <tr>
        <td>Lê Phạm Tuấn Kiệt</td>
        <td>8.5</td>
        <td>CNTT</td>
        <td><a href="student.htm?lnkEdit">Sửa</a></td>
    </tr>
</table>
```




DEMO
Chạy và giải thích student.htm

Fpt University

FPT POLYTECHNIC

Lập trình Java 5

Phần 2

www.poly.edu.vn

XỬ LÝ THAM SỐ NGƯỜI DÙNG

- Tham số là dữ liệu truyền đến server khi có yêu cầu từ người dùng dưới dạng các trường của form hoặc chuỗi truy vấn của liên kết
- Ví dụ
  - Khi nhấp vào liên kết sau thì các tham số **mark** và **name** sẽ được truyền đến phương thức action

```
<a href="say-hello.htm?mark=5&name=Phương">Hello</a>
```

  - Khi nhấp vào nút Hello của form sau thì các tham số **mark** và **name** sẽ được truyền đến phương thức action

```
<form action="say-hello.htm">
    <input name="mark">
    <input name="name">
    <button>Hello</button>
</form>
```

XỬ LÝ THAM SỐ

- Spring MVC cung cấp các phương pháp nhận tham số sau đây
  - Sử dụng **HttpServletRequest** tương tự Servlet
  - Sử dụng **@RequestParam**
  - Sử dụng **JavaBean**
  - Sử dụng **@PathVariable** để nhận một phần trên URL

SỬ DỤNG HTTPSERVLETREQUEST

- Chỉ cần thêm đối số HttpServletRequest vào phương thức action là có thể nhận được tham số người dùng như Servlet

```
@RequestMapping(value="login", method=RequestMethod.POST)
public String login(HttpServletRequest request) {
        String id = request.getParameter("id");
        String pw = request.getParameter("password");
        // ...
        return "login";
}
```

SỬ DỤNG @REQUESTPARAM (1)

- Sử dụng **@RequestParam** thể hiện tính chuyên nghiệp hơn và có thể chuyển đổi tự động sang kiểu mong muốn.
- Ví dụ sau được sử dụng để nhận các tham số có tên là id và password

```
@RequestMapping(value="login", method=RequestMethod.POST)
public String login(@RequestParam("id") String id,
                    @RequestParam("password") String password) {
        // ...
        return "login";
}
```

SỬ DỤNG @REQUESTPARAM (2)

- @RequestParam(value, defaultValue, required) là dạng đầy đủ với ý nghĩa của các tham số:
  - value: chỉ ra tên tham số muốn nhận
  - defaultValue: là giá trị mặc định của tham số khi tham số không tồn tại
  - required: tham số có bắt buộc hay không
- Ví dụ với khai báo nhận tham số sau
  - @RequestParam(value="tuoi", defaultValue="20", required=false) **Integer age**
    - Tên tham số là **tuoi** sẽ được nhận vào đối số là **age**
    - Nếu không có tham số thì giá trị của **age** là 20
    - Tham số **tuoi** là không bắt buộc

Ví dụ

```
@RequestMapping(value="login", method=RequestMethod.POST)
public String login(ModelMap model, HttpServletRequest request) {
        String id = request.getParameter("id");
        String pw = request.getParameter("password");
        if(id.equals("fpt") && pw.equals("polytechnic")){
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        else{
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        return "user/login";
}
```

```
@RequestMapping(value="login", method=RequestMethod.POST)
public String login(ModelMap model,
        @RequestParam("id") String id, @RequestParam("password") String pw) {
        if(id.equals("fpt") && pw.equals("polytechnic")){
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        else{
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        return "user/login";
}
```

SỬ DỤNG JAVABEAN (1)

- Lớp JavaBean là lớp thỏa mãn các qui ước sau
  - Phải được định nghĩa là public
  - Phải có constructor không tham số
  - Đọc ghi dữ liệu thông qua getter/setter

```
public class User {
    private String id;
    private String password;

    public String getId() {return id;}
    public void setId(String id) {this.id = id;}
    public String getPassword() {    return password;}
    public void setPassword(String password) {this.password = password;}
}
```

SỬ DỤNG JAVABEAN (2)

- Thuộc tính của bean được xác định từ các getter và setter bằng cách
  - Bỏ get và set và đổi ký tự đầu tiên của phần còn lại sang ký tự thường
- Ví dụ lớp User có 2 thuộc tính cho phép đọc/ghi là id và password
  - Thuộc tính id được xác định từ get**Id**() và set**Id**()
  - Thuộc tính password được xác định từ get**Password**() và set**Password**()
- *Chú ý quan trọng: các trường dữ liệu không phải là thuộc tính của bean*

SỬ DỤNG JAVABEAN (3)

- Spring MVC cho phép sử dụng JavaBean để nhận các tham số **cùng tên** với các thuộc tính của bean.

```
@RequestMapping(value="login", method=RequestMethod.POST)
public String login(ModelMap model, User user) {
        if(user.getId().equals("fpt") && user.getPassword().equals("polytechnic")){
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        else{
                model.addAttribute("message", "Sai thông tin đăng nhập !");
        }
        return "user/login";
}
```

- Với ví dụ này thì các thuộc tính id và password của đối số user sẽ nhận các giá trị từ các tham số cùng tên là id và password.

SỬ DỤNG @PATHVARIABLE

- Spring MVC cho phép nhận một phần dữ liệu từ đường dẫn URL
- Ví dụ action edit() sau đây sẽ lấy được tên sinh viên từ URL student/**Nguyễn Văn Tèo**.htm

```
@Controller
@RequestMapping("/student")
public class StudentController {
        @RequestMapping(value="/{name}", params="lnkEdit")
        public String edit(ModelMap model, @PathVariable("name") String name) {
                ...
                return "student";
        }
}
```

```
<a href="student/Nguyễn Văn Tèo.htm?lnkEdit">Sửa</a>
```

NHẬN GIÁ TRỊ COOKIE

- Trong Servlet bạn có thể nhận cookie thông qua HttpServletRequest. Phương pháp này viết mã khá dài dòng, phức tạp.
- Trong Spring MVC bạn có thể sử dụng **@CookieValue** để nhận dữ liệu từ cookie

```
@RequestMapping("say-hello")
public String sayHello(@CookieValue("userid") String id) {
        return "hello";
}
```

- Ví dụ này cho phép sử dụng đối số **id** để nhận giá trị của cookie có tên là **userid**

@CookieValue

- @CookieValue(**value**, **defaultValue**, **required**) có 3 tham số và ý nghĩa như sau
  - Value: tên cookie muốn nhận dữ liệu
  - defaultValue: giá trị mặc định của cookie
  - Required: có bắt buộc cookie userid có tồn tại hay không
- Ví dụ
  - @CookieValue(value="userid", defaultValue="poly", required=false) String id
    - Sử dụng đối số id để nhận giá trị của cookie có tên là userid
    - Nếu cookie không tồn tại thì giá trị của id là poly
    - Cookie này cho phép không tồn tại


DEMO
Đăng nhập 2 có ghi nhớ tài khoản

ĐẦU RA CỦA PHƯƠNG THỨC ACTION

- Return của phương thức action không đơn thuần phải là tên của view mà có thể là 1 trong 3 trường hợp sau
  - Tên view => ViewResolver sẽ xử lý để xác định view
    - return **"<tên view>"**
  - Nội dung => được trả trực tiếp về client mà không qua ViewResolver. Trường hợp này phương thức action phải được chú thích bởi **@ResponseBody**
    - return **"<Nội dung>"**
  - Lời gọi một action khác
    - return **"redirect:/<action>"**

ĐẦU RA CỦA PHƯƠNG THỨC ACTION

```
@RequestMapping("say-hello")
public String sayHello() {
        return "hello";
}
```
→ /WEB-INF/views/hello.jsp

```
@ResponseBody
@RequestMapping("say-hello")
public String sayHello() {
        return "Hello World";
}
```
→ Hello World

Trường hợp này rất hữu ích cho tương tác JSON, JavaScript, XML…

```
@RequestMapping("say-hello")
public String sayHello() {
        return "redirect:/home/index.htm";
}
```
→ @RequestMapping(“/home/index”)

FPT POLYTECHNIC

TỔNG KẾT NỘI DUNG BÀI HỌC

- ☑ Sử dụng thành thạo @RequestMaping
  - ☑ Ánh xạ nhiều action
  - ☑ Ánh xạ phân biệt POST|GET
  - ☑ Ánh xạ phân biệt tham số
- ☑ Nắm vững phương pháp nhận tham số
  - ☑ Sử dụng HttpServletRequest
  - ☑ Sử dụng @RequestParam
  - ☑ Sử dụng JavaBean
  - ☑ Sử dụng @PathVariable để nhận dữ liệu từ URL
- ☑ Biết cách nhận Cookie với @CookieValue
- ☑ Hiểu rõ kết quả của phương thức action


TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
Cảm ơn