

Lập trình web SPRING MVC



BÀI 3: LÀM VIỆC VỚI FORM

Mục tiêu :

01 Hiểu cơ chế buộc dữ liệu

02 Xây dựng form trong Spring

03 @ModelAttribute



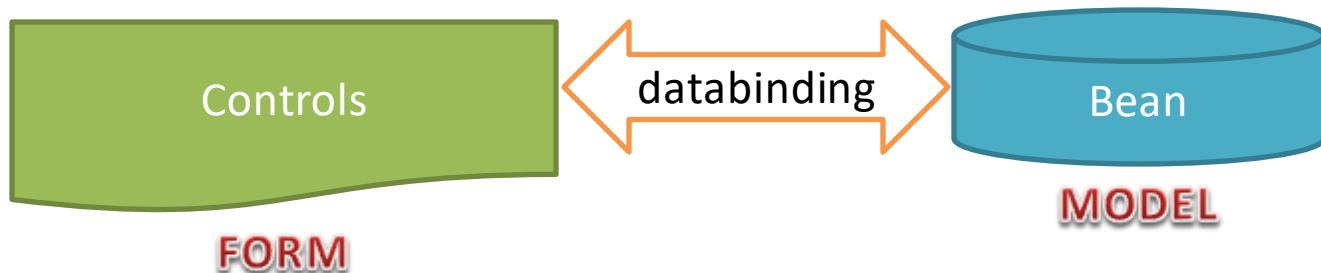


01

Giới thiệu Databinding?

GIỚI THIỆU DATABINDING?

- ❑ Databinding là sự kết nối dữ liệu của bean đặt trong model đến các điều khiển trên form.
- ❑ Khi thay đổi dữ liệu trong bean thì dữ liệu trên các điều khiển cũng thay đổi theo.
- ❑ Ràng buộc dữ liệu có thể là 1 chiều hoặc 2 chiều
 - ❖ Chiều lên: *chuyển dữ liệu từ các điều khiển vào các thuộc tính của bean*
 - ❖ Chiều về: *hiển thị dữ liệu từ các thuộc tính của bean lên các điều khiển của form*



BUỘC DỮ LIỆU VỚI CÁC THẺ HTML?

- ❑ Bạn có thể buộc dữ liệu từ các thuộc tính của bean vào các điều khiển HTML bằng cách sử dụng biểu thức EL

```
<form action="login.htm" method="post">  
  <div>User Name:</div>  
  <input name="id" value="${user.id}">  
  
  <div>Password:</div>  
  <input name="password" value="${user.password}">  
  
  <hr>  
  <button>Login</button>  
</form>
```

BUỘC DỮ LIỆU VỚI CÁC THẺ HTML?

❑ Dù chúng ta hoàn toàn có thể buộc dữ liệu từ bean trong model lên form với EL nhưng gặp phải một số hạn chế sau:

- ❖ Phải viết mã trên giao diện, dài dòng, khó quản lý
- ❖ Đổ dữ liệu vào các List Control trở nên phức tạp và khó khăn
 - Combox
 - Listbox
 - Radiobuttons
 - Checkboxes
- ❖ Kiểm và thông báo lỗi

SPRING FORM

- ❑ Spring MVC cung cấp thư viện thẻ giúp việc buộc dữ liệu từ bean vào các điều khiển trở nên dễ dàng hơn

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
```

- ❑ Sau khi khai báo thư viện thẻ ngay đầu trang JSP, chúng ta có thể tạo form và ràng buộc dữ liệu

```
<form:form action="login.htm" modelAttribute="user">
  <div>User Name:</div>
  <form:input path="id"/>
  <div>Password:</div>
  <form:input path="password"/>

  <hr>
  <button>Login</button>
</form:form>
```

Thẻ trong thư
viện form

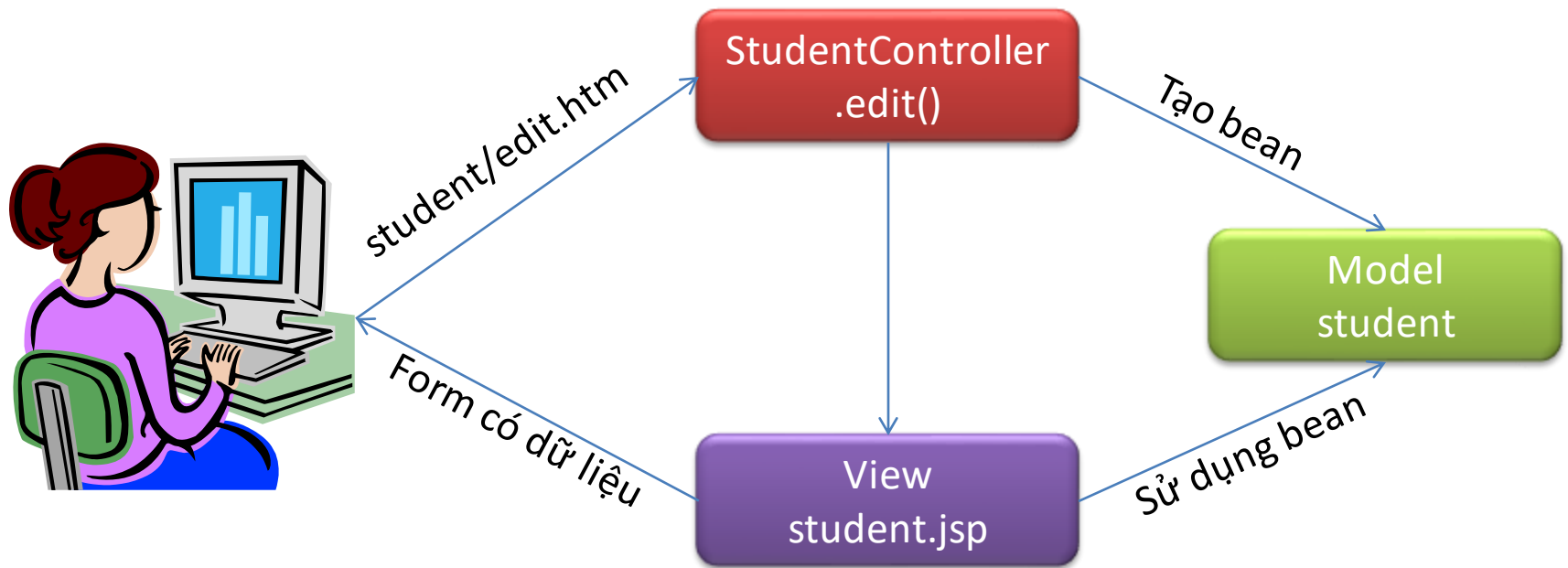
Tên thuộc tính
của bean user

Tên của bean
đặt trong model

ƯU ĐIỂM CỦA FORM SPRING?

- ❑ Cung cấp cơ chế buộc dữ liệu lên các điều khiển
- ❑ Form đơn giản, rõ ràng, dễ hiểu
- ❑ Khi thay đổi dữ liệu trong bean thì dữ liệu trên các điều khiển cũng thay đổi theo.
- ❑ Cấp dữ liệu vào các List Control trở nên rất đơn giản
- ❑ Kiểm và hiển thị lỗi một cách dễ dàng

TÌNH HUỐNG BUỘC DỮ LIỆU



- ❑ Người sử dụng yêu cầu `student/edit.htm`
- ❑ Phương thức `edit()` tạo bean và đặt vào model
- ❑ View chứa form buộc dữ liệu từ bean trong model lên các điều khiển của form

LỚP BEAN

```
package poly.bean;
```

```
public class Student {
```

```
    private String name;  
    private Double mark;  
    private String major;
```

Trường chứa dữ liệu

```
    public Student(){  
    public Student(String name, Double mark, String major) {  
        this.name = name;  
        this.mark = mark;  
        this.major = major;  
    }
```

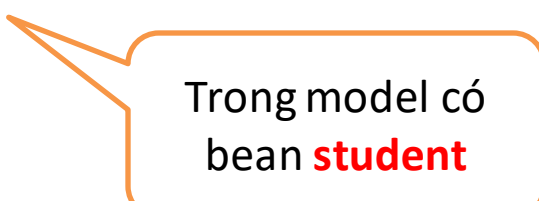
Các constructor

```
    public String getName() {return name;}  
    public void setName(String name) {this.name = name;}  
    public Double getMark() {return mark;}  
    public void setMark(Double mark) {this.mark = mark;}  
    public String getMajor() {return major;}  
    public void setMajor(String major) {this.major = major;}  
}
```

Các phương thức
getter/setter

LỚP STUDENTCONTROLLER

```
@Controller
@RequestMapping("/student/")
public class StudentController {
    @RequestMapping("edit")
    public String edit(ModelMap model) {
        Student sv = new Student("Nguyễn Văn Tèo", 9.5, "WEB");
        model.addAttribute("student", sv);
        return "student";
    }
}
```



Trong model có
bean **student**

- ❑ Khi gọi **student/edit.htm** thì phương thức action **edit()** sẽ chạy. **edit()** tạo một đối tượng **sv** và đặt vào **model** với tên là **student** để chuyển sang view **student.jsp**

THIẾT KẾ FORM CÓ RÀNG BUỘC DỮ LIỆU

- ❑ View student.jsp chứa form buộc các thuộc tính của bean vào các điều khiển

```
<%@ taglib uri = "http://www.springframework.org/tags/form" prefix = "form"%>
```

...

```
<form:form action = "student/update.htm" modelAttribute = "student">
```

```
  <div>Họ và tên</div>
```

```
  <form:input path = "name"/>
```

```
  <div>Điểm</div>
```

```
  <form:input path = "mark"/>
```

```
  <div>Chuyên ngành</div>
```

```
  <form:select path = "major">
```

```
    <form:option value = "APP">Ứng dụng phần mềm</form:option>
```

```
    <form:option value = "WEB">Thiết kế trang web</form:option>
```

```
  </form:select>
```

```
  <br>
```

```
  <button>Update</button>
```

```
</form:form>
```

Bean buộc dữ liệu
lên các điều khiển

Các thuộc tính của bean

BUỘC DỮ LIỆU LÊN FORM

Spring MVC - Databindin x

localhost:9999/Java5/student/edit.htm

Quản lý sinh viên

Họ và tên

Điểm

Chuyên ngành

Các thuộc tính của bean student được buộc với các điều khiển của form. Như vậy khi muốn thay đổi dữ liệu trên form bạn chỉ cần thay đổi bean trong model

BUỘC DỮ LIỆU CHIỀU LÊN

- ❑ Form sẽ submit dữ liệu đến action "**update.htm**".
Bạn cần bổ sung phương thức action update() vào StudentController để xử lý nút Update.

```
@RequestMapping("update")  
public String update(@ModelAttribute("student") Student student) {  
    return "student";  
}
```

- ❑ Dữ liệu form được chuyển vào các thuộc tính của đối số action student.
- ❑ **@ModelAttribute("student")** sẽ bổ sung một attribute có tên là student có giá trị là đối số student vào model. Attribute này sẽ buộc dữ liệu lên các điều khiển khi quay trở lại form

CÁC ĐIỀU KHIỂN FORM CỦA SPRING

| Điều khiển Spring | Sinh ra điều khiển HTML |
|----------------------|--------------------------|
| <form:form> | <form> |
| <form:input/> | <input type="text"/> |
| <form:textarea/> | <textarea/> |
| <form:checkbox/> | <input type='checkbox'/> |
| <form:radiobutton/> | <input type='radio'/> |
| <form:hidden/> | <input type='hidden'/> |
| <form:password/> | <input type='password'/> |
| <form:button/> | <button/> |
| <form:select/> | <select/> |
| <form:radiobuttons/> | Nhóm radio |
| <form:checkboxes/> | Nhóm checkbox |

Đây là các List Control cần được cấp dữ liệu từ Collection, Array hoặc Map

PHIẾU ĐĂNG KÝ ĐƯỜNG SẮT

Họ:

Tên:

Gender: ☐ Male ☐ Female

Meals: ☐ Ăn sáng ☐ Ăn trưa ☐ Ăn tối

Từ tỉnh/thành:

Đến tỉnh/thành:

Đăng Ký

<form:input/>

<form:radiobutton/>

<form:checkbox/>

<form:select/>



02

Sử dụng List Control

SỬ DỤNG LIST CONTROL

ĐỔ DỮ LIỆU VÀO LIST CONTROL

Tên nhân viên:

Quốc tịch:

Trình độ:
☐ Doctor ☒ Master ☐ University ☐ Other

Chức danh:

Sở thích:
☒ Travelling ☐ Music ☒ Sports ☐ Reading

❑ Các điều khiển được sử dụng để tạo List Control

<form:select>

<form:radiobuttons>

<form:select multiple>

<form:checkboxes>

Họ và tên

Nguyễn Văn Tèo

Điểm

9.5

Chuyên ngành

Ứng dụng phần mềm ▼

Update

Đổi từ nhập dữ liệu sang
chọn mục trong ComboBox

❑ Để đạt được điều mong muốn trên thì chúng ta cần thay đổi

- ❖ StudentController: phải cung cấp dữ liệu dạng Array, Collection hoặc Map vào model
- ❖ Student.jsp: phải thay điều khiển và đổ dữ liệu vào

ĐỒ DỮ LIỆU VÀO COMBOBOX

```
@ModelAttribute("majors")
public String[] getMajors() {
    String[] majors = {
        "Ứng dụng phần mềm",
        "Thiết kế trang web"
    };
    return majors;
}
```

String[]

```
<div>Chuyên ngành</div>
<form:select path="major" items="${majors}" />
```

```
<select id="major" name="major">
    <option value="Ứng dụng phần mềm">Ứng dụng phần mềm</option>
    <option value="Thiết kế trang web">Thiết kế trang web</option>
</select>
```

Ứng dụng phần mềm ▼
Ứng dụng phần mềm
Thiết kế trang web

ĐỔ DỮ LIỆU VÀO COMBOBOX

❑ Thay đổi StudentController

- ❖ Bổ sung phương thức `getMajors()`.
- ❖ **@ModelAttribute("majors")** sẽ đặt kết quả của phương thức này vào trong Model với tên là **majors**.
Dữ liệu này được sử dụng để đổ vào **ComboBox**

❑ Thay đổi view (student.jsp)

- ❖ Thay `<form:input path="major">` bằng `<form:select path="major" items="${majors}">`.
- ❖ Thuộc tính `items` chỉ ra dữ liệu (Collection, Map hay mảng) đặt trong Model để đổ vào ComboBox

@ModelAttribute

□ Trong Spring MVC @ModelAttribute được sử dụng để bổ sung attribute vào model trong 2 trường hợp:

❖ @ModelAttribute(name) argument

- Sẽ bổ sung attribute có tên là name và có giá trị là giá trị của đối số phương thức action
- Tương đương: `model.addAttribute(name, argument)`

❖ @ModelAttribute(name) method

- Sẽ bổ sung attribute có tên là name và có giá trị là kết quả của phương thức
- Tương đương: `model.addAttribute(name, method())`

□ Trong view bạn có thể sử dụng nó như một attribute bình thường: buộc vào form, sử dụng EL, đổ vào ListControl

ĐỔ DỮ LIỆU VÀO COMBOBOX

```
@ModelAttribute("majors")
```

```
public List<String> getMajors() {  
    List<String> majors = new ArrayList<>();  
    majors.add("Ứng dụng phần mềm");  
    majors.add("Thiết kế trang web");  
    return majors;  
}
```

List<String>

```
<div>Chuyên ngành</div>  
<form:select path="major" items="${majors}" />
```

```
<select id="major" name="major">  
    <option value="Ứng dụng phần mềm">Ứng dụng phần mềm</option>  
    <option value="Thiết kế trang web">Thiết kế trang web</option>  
</select>
```

Ứng dụng phần mềm ▼
Ứng dụng phần mềm
Thiết kế trang web

ĐỔ DỮ LIỆU VÀO COMBOBOX

```
@ModelAttribute("majors")
public Map<String, String> getMajors() {
    Map<String, String> majors = new HashMap<>();
    majors.put("APP", "Ứng dụng phần mềm");
    majors.put("WEB", "Thiết kế trang web");
    return majors;
}
```

Map<String, String>

```
<div>Chuyên ngành</div>
<form:select path="major" items="${majors}" />
```

```
<select id="major" name="major">
    <option value="APP">Ứng dụng phần mềm</option>
    <option value="WEB">Thiết kế trang web</option>
</select>
```

Ứng dụng phần mềm ▼
Ứng dụng phần mềm
Thiết kế trang web

ĐỔ DỮ LIỆU VÀO COMBOBOX

```
@ModelAttribute("majors")  
public List<Major> getMajors() {  
    List<Major> majors = new ArrayList<>();  
    majors.add(new Major("APP", "Ứng dụng phần mềm"));  
    majors.add(new Major("WEB", "Thiết kế trang web"));  
    return majors;  
}
```

List<Major>

```
public class Major {  
    private String id;  
    private String name;  
  
    public Major() {}  
    public Major(String id, String name) {}  
  
    public String getName() {}  
    public void setName(String name) {}  
    public String getId() {}  
    public void setId(String id) {}  
}
```

```
<div>Chuyên ngành</div>  
<form:select path="major" items="${majors}"  
    itemLabel="name" itemValue="id"/>
```

```
<select id="major" name="major">  
    <option value="APP">Ứng dụng phần mềm</option>  
    <option value="WEB">Thiết kế trang web</option>  
</select>
```

Ứng dụng phần mềm ▼
Ứng dụng phần mềm
Thiết kế trang web

ĐỔ DỮ LIỆU VÀO LIST CONTROL

❑ `<form:select path="property" items="{items}"
itemValue="prop1" itemLabel="prop2">`

- ❖ items: chỉ ra tập dữ liệu đổ vào ComboBox
- ❖ itemValue và itemLabel chỉ được sử dụng khi tập items là Collection<Bean>
 - itemValue: chỉ ra tên thuộc tính để làm giá trị
 - itemLabel: chỉ ra tên thuộc tính để làm nhãn (nhìn thấy)

❑ List Control khác có cùng cú pháp với select

- ❖ `<form:radiobuttons path="property" items="{items}"
itemValue="prop1" itemLabel="prop2">`
- ❖ `<form:checkboxes path="property" items="{items}"
itemValue="prop1" itemLabel="prop2">`

❑ Đổ dữ liệu vào các List Control là như nhau

CÁC THUỘC TÍNH THƯỜNG DÙNG

❑ Thẻ Spring `<form:tag>` có một số thuộc tính thường dùng sau:

- ❖ `cssClass`: thay cho thuộc tính class trong HTML
- ❖ `disabled`: thay cho thuộc tính disabled trong HTML
- ❖ `readonly`: thay cho thuộc tính readonly trong HTML
- ❖ `cssErrorClass`: cho ra class định dạng thông báo lỗi

❑ Ví dụ:

- ❖ `<form:input path="id" readonly="true"/>`
- ❖ `<form:input path="name" cssClass="form-control">`

TỔNG KẾT NỘI DUNG BÀI HỌC

- ✓ Tìm hiểu cơ kết buộc dữ liệu 2 chiều
- ✓ Sử dụng thuộc tính `modelAttribute` để kết nối attribute trong model với form
- ✓ Sử dụng `path="property"` để buộc thuộc tính của bean vào các điều khiển form
- ✓ Đổ dữ liệu vào List Control
- ✓ Biết cách sử dụng `@ModelAttribute`
- ✓ Khai thác một số thuộc tính khác của các điều khiển Spring