LẬP TRÌNH JAVA WEB SPRING MVC

Nguyễn Thái Sơn

SPRING BEAN

INJECT & AUTOWIRE

Spring Bean

- Bean: là các đối tượng được khởi tạo từ class, tồn tại trong khung làm việc của request, session, app context, … Có các loại
  - Singleton: 1 obj duy nhất trong IoC. Mặc định, hay dùng
  - Prototype: 1 obj mới đc tạo khi sử dụng bean. Ít dùng
  - Request: tồn tại trong một HTTP Request
  - Session và Global Session: tồn tại trong một HTTP Session
- Dependency Injection (DI) gán bean vào biến khi sử dụng trong chương trình. Autowire hoặc inject thông qua
  - Constructor: Gán giá trị cho biến thông qua constructor
  - Setter: Gán giá trị cho biến thông qua hàm setter

Chu kỳ Bean

- Init: gọi khi bắt đầu tạo bean
- Tạo thàng công: bean đc tạo thành công và quản lý bởi IoC
- Destroy: Gọi khi hủy bean

```
public class Person {
	private String name;
	private int age;
	public Person(String name, int age) {
		super();
		this.name = name;
		this.age = age;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}
	public void init() {
		System.out.println("Khoi tao bean");
	}
	public void destroy() {
		System.out.println("Bean da bi huy");
	}
}
```

Tạo Bean

- Tạo Bean bản chất là khởi tạo các đối tượng từ class
- XML config: khởi tạo thông qua beans namespace, dùng constructor hoặc thuộc tính với hàm set/get
- Java config: khởi tạo 1 obj thông qua từ khóa new và @Bean

Tạo Bean

```
<bean id="person" class="com.ezcloud.model.Person" init-method="init" destroy-method="destroy" scope="singleton">
  <constructor-arg name="name" value="Nguyen Van A"></constructor-arg>
  <constructor-arg name="age" value="20"></constructor-arg>
  <!-- <property name="name" value="Nguyen Van C"></property>
  <property name="age" value="20"></property> -->
  </bean>
  <bean id="order" class="com.ezcloud.model.Order" autowire="byName">
  <!-- <property name="person" ref="person"></property>
  <constructor-arg name="person" ref="person"></constructor-arg> -->
  </bean>
```

Tạo Bean

```
@Bean(name = "person", initMethod = "init", destroyMethod = "destroy")
@Scope(value = "prototype")
public Person person() {
Person person = new Person("Nguyen Van V", 30);
return person;
}
@Bean(name = "person2")
public Person person2() {
Person person = new Person("Nguyen Van C", 45);
return person;
}
@Bean
@Autowired
@Qualifier("person2")
public Order order(Person person) {
return new Order(person);
}
```

REQUESTMAPPING

CÁCH XỬ LÝ REQUEST

@RequestMapping

- Dùng cấu hình các request từ client gửi lên dựa vào đường dẫn URL tới một hàm bussiness xử lý ở Controller
- Trong Spring có nhiều cách viết mapping và hàm xử lý

```
public class HelloController {
    @RequestMapping("/hello")
    public String sayHello(ModelMap map, @RequestParam(name =
    "username", required = true) String user) {
        map.addAttribute("msg", "Xin chào: "+user);
        return "hello";
    }
    @RequestMapping("/hello2")
    public String sayHello2(HttpServletRequest request,
    @RequestParam(name = "username") String username) {
        request.setAttribute("msg", "helooooooooo "+username);
        return "hello";
    }
    @RequestMapping("/hello3")
    public ModelAndView sayHello3(HttpServletRequest request,
    @RequestParam(name = "username", required = true) String username) {
        request.setAttribute("msg", "Xin chào ....."+username);
        return new ModelAndView("hello");
    }
}
```

STATIC RESOURCE

SỬ DỤNG FILE CSS/JS/IMAGE

Spring Static Resource

- Static Resource: Bao gồm các file javascript(js), css, hình ảnh... sử dụng cho hệ thống website
- Để sử dụng các static resource chúng ta cần khai báo để map các request đến các files

REQUEST PARAM

REQUEST PARAMETER

Request Param

- Đọc các giá trị từ client gửi lên server thông qua Url hoặc Form
  - @RequestParam(name="id", required="true") int userId,
  - @RequestParam(name = "username", required = **true) String username,**

Path Variable

Path variable

Path Variable

- Đọc một giá trị trong đường dẫn Url
  - @RequestParam(name="id", required="true") int userId,
  - @RequestParam(name = "username", required = **true) String username,**

```
@RequestMapping("/hello/{name}/{id}")
public String pathVariable(HttpServletRequest request,
@PathVariable(name = "name") String name,
@PathVariable(name = "id") int id) {
    request.setAttribute("msg", "Xin chào: "+name+" id của bạn là: "+id);
    return "hello";
}
```

SPRING FORM

LÀM VIỆC VỚI FORM

Form Tag

- <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
- <form:form></form:form>
- <form:input/>
- <form:password/>
- <form:hidden/>
- <form:radiobutton/>
- <form:select></form:select >
- <form:textarean></form:textarean >
- <form:checkbox></form:checkbox >
- < form:checkboxes></form:checkbox >

Form Binding

- Tự động ghép các thuộc tính trong obj với các trường thông tin trên form thông qua spring form binding
- Bước 1: Trong controller tạo đối tượng và truyền sang view

```
public class User {
private String name;

@RequestMapping(value = "/add-user", method = RequestMethod.GET)
public String addUsser(HttpServletRequest request) {
    User user = new User("Spring");
    request.setAttribute("user", user);
    return "addUser";
```

Form Binding

- Bước 2: Trong view sử dụng các thẻ spring form tag để ghép trường thông tin với nhau

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

<c:url value="/add-user" var="url"></c:url>
<form:form modelAttribute="user" method="post" action="${url }">
<form:input path="name"/>
<button type="submit">Submit</button>
</form:form>
```

Form Binding

- Bước 3: Trong controller sẽ ghép thông tin từ form vào obj chứa thông tin

```
@RequestMapping(value = "/add-user", method = RequestMethod.POST)
public String addUser(HttpServletRequest request, @ModelAttribute("user") User user) {
    request.setAttribute("user", user);
    return "viewUser";
}
```

QUESTIONS?