CÓ 2 DẠNG TẬP HỢP

1. COLLECTION LÀ 1 TẬP HỢP CÁC PHẨN TỬ ĐƠN, NÓ LÀ MỞ RỘNG CỦA MẢNG BAO GỒM

🡪 LIST (INTERFACE), nếu new là Arraylist ( class)

🡪 SET: Set là 1 tâp hợp ko trùng -> Có KeySet -> lấy tất cả giá trị Key

1. MAP THÌ CÁC CẶP

**Key word**: Functional Programming

Protocol = giao thức

JDBC:

1. NẠP DRIVER
2. MỞ KẾT NỐI
3. TẠO PrepairedStatment
4. Truy vấn dư liệu và xử lí
5. ĐÓNG KẾT NỐI

Crt + shift + O 🡪 Import nhanh

Kiểu nguyên thủy int => new ra 🡪 cho ra là 0 boolean 🡪 cho ra false

Kiểu đóng bao Integer => new ra 🡪 cho ra là null Boolean 🡪 ra null

**GET:** ĐÓNG GÓI DỮ LIỆU CỦA FORM, GẮN VÀO URL ĐỂ GỬI ĐI, HẠN CHẾ DỮ LIỆU, KO UPLOAD ĐƯỢC HÌNH ẢNH, TÍNH BẢO MẬT ( HIỆN MẬT KHẨU TRÊN URL)

**POST**: ĐÀO 1 ĐƯỜNG NGẦM, MỞ 1 SOCKET RIÊNG , GỬI TỪ TRÌNH DUYÊT LÊN ĐẾN CHƯƠNG TRÌNH CỦA MÌNH

**@CONTROLLER chỉ chạy trong Packet ban đầu tạo ra..**

**Packet ban đầu com.demo 🡪 chạy được**

**Text

Description automatically generated**

**Trường hợp này ko chạy được:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**@RESTCONTROLLER:** CHỦ YẾU SỬ DỤNG TRONG WEB TĨNH, TRẢ VỀ KẾT QUẢ JSON

**@ResponseBody TRẢ VỀ NỘI DUNG**

Nếu ko có thì trả về nội dung trong file của đường dẫn

@**RequestParam(“name”) String fullname**

Để lấy giá trị tham số trong Form và xử lí trong Controller , ta dùng RequestParam và gán cho biến fullname kiểu String

* Lấy theo dạng Key và value ? localhost:8080/home/index?name=1;

**Optional<kiểu dữ liệu>** để xử lí các tham số tùy chọn (lúc có, lúc ko):

Vd: @RequestParam(“password”) Optional<String> opt;

String password = opt.orElse(“123456”);

Nếu người dùng ko nhập pass thì mặc định pass là 123456;

Ngoài ra còn IsEmty(): boolean, IsPresent(): Boolean; get().

@**RequestPart(“photo”) MultipartFile photo :** dùng để xử lí UpdateFile,

Bắt buộc dùng method: Post, enctype=”multipart/form-data”

**Entity** để miêu tả table của SQL

**Repository** dùng để truy xuất dữ liệu 🡪 tạo Interface extends JpaReposity<Entity, kiểu khóa chính>

**FindById()** : đọc dữ liệu database vào bộ nhớ ( chạy nhanh, tốn bộ nhớ)

**GetById()** : ko đọc, chỉ lấy link trỏ vào database ( ko đưa vào bộ nhớ)

**@Transactional:** có hiểu là giao dịch đơn, xem toàn bộ code bên trong method là 1 giao dịch. Chỉ cần 1 thao tác lưu sai thì sẽ hủy toàn bộ

**Thêm data từ form**

Text

Description automatically generated

**Khi field bean (java) trùng tên với name của input**

Text

Description automatically generatedText

Description automatically generated

Thì ta được rút gọn viết ngắn

Text

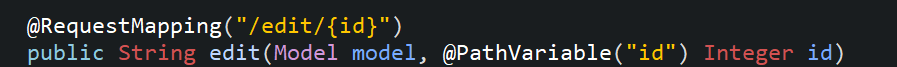
Description automatically generated with medium confidence

Chạy lại request /inout

return "redirect:/inout";

**lấy đường dẫn**  
B1: bọc {name}

B2 dùng **@PathVariable**(“name”) kiểu tên biến)

****

**Forward: xảy ra phía server , action 1 chạy xong 🡪 action 2 chạy, Action 1 và action 2 chia sẻ dữ liệu qua Model**

**Redirect: Action1 , action2 hoạt động trên 2 request khác nhau chia sẻ qua tham số (RedirectAttribute params), Action 1 chạy xong trả về cho browser, browser mới gọi request 2**

**Vd ghi mình insert 1 bản ghi 🡪 nếu mình dùng Forward, người dùng vô tình bấm F5 thì sẽ insert thêm 1 bản nữa..**

**Dùng 🡪 Redirect, sau khi insert. Cho dù ng dùng F5 thì chỉ reload lại method hiển thị kết quả**

**THYMELEAF**

SỬ DỤNG GỌI MODEL

**1)** **[[ ${x} ]]** <h3> [[${x}]] </h3>

**HOẶC DÙNG th:text = “${x}” trong thẻ mở tag**

<h3 th:text="${x}"></h3>

**2) ĐỊNH DẠNG CHỮ : [(${x}]**

<h3> Unescape [(${x})] </h3>

**HOẶC DÙNG th:utext=”${x}”**

<h3 th:utext="${x}"></h3>

**U là unescape => ko mã hóa**

* 2 cách viết các khau: trên thì thêm chèn thêm chữ đầu và cuối được

Dưới thì ko.. muốn chèn phải thêm thẻ span ở giữa **: vd**

<h3>Thanh <span th:text="${x}"></span> Binh </h3>

**3) ĐƯA DỮ LIỆU VÀO THUỘC TÍNH**

<h1 th:title="${x}">Hello AE nhé</h1>

HOẶC

<h1 th:title="@{|Tieu de ${x}|}

**4) ĐƯA LINK HÌNH ẢNH**

<img th:src="@{|/image/Heart.png|}"

**CÁCH TỐT NHẤT @{| BIỂU THỨC |}**

**Databinding và Validation**

**Databinding: Là sự kết nối dữ liệu từ Model với các điều khiển trên giao diện và ngược lại**

**Validation: ràng buộc dữ liệu, kiểm soát dữ liệu đầu vào ( kiểm tra cách lỗi người dùng nhập vào)**

**@NotBlank: ko để trống (dùng cho chuỗi)**

**Blank -> length >0 hoặc chuỗi chỉ chứa khoảng trắng**

**@NotEmty: (dùng cho chuỗi) (ngoài ra dùng cho tập hợp, vd Mảng, List,)**

**Emty ->Length =0; đã lưu địa chỉ bộ nhớ của biến**

**@NotNull: ko được Null ( ko dùng cho chuỗi) chưa lưu địa chỉ bộ nhớ của biến**

**( null là object)**

**@Min: set số Min**

**@Max: set số Max**

**@Email: điền đúng dạng email**

**Sự khác nhau của Null và Emty**

**Khi cho chuỗi = null 🡪 in ra sẽ tạo ra ngoại lệ con trỏ Null**

* **Null được sử dụng để chỉ ra KHÔNG CÓ GÌ**

**Khi cho chuỗi= (emty) 🡪 in ra sẽ cho kết quả độ dài = 0**

* **Emty sẽ cho độ dài của chuỗi = 0**

**\*Note: Biểu thức chính quy ( regular Expression) cần tìm hiểu**

**LOMBOK**

**@Data:** TẠO GETTER, SETTER TỰ ĐỘNG.

**@AllArgsContructor: sinh ra CONTRUCTOR với tất cả tham số**

**@NoArgsContructor : sinh ra CONTRUCTOR ko tham số**

1. **Gọi lấy Object**

**Trường hợp 1 :**

**CÁCH 1:**

<li>HO TEN: [[${u.getFullname()}]] </li>

**CÁCH 2: GẮN GỌN- BỎ GET/SET VÀ VIẾT HOA THƯỜNG CHỮ ĐẦU**

<li>MA: [[${u.id}]] </li>

**Trường hơp 2:**

<ul th:object="${u}">

**-------🡪 GỌI** <li>HO TEN: [[\*{fullname}]] </li>

**2) KHAI BÁO BIẾN TÍNH TOÁN TRONG GIAO DIỆN**

<ul th:object="${u}" th:with="ss =\*{salary\*salary}, xy=100">

**-------🡪 GỌI** <li>LUONG: [[${ss}]] va [[${xy}]] </li>

**5) LẤY OBJECT TRONG TABLE**

**Text

Description automatically generated**

**HOẶC DÙNG th:object  
Text

Description automatically generated**

**LẤY VỊ TRÍ CỦA PHẦN TỬ ( TÍNH TỪ 0)**

<tr th:each="item,state: ${staffs}">

<td>[[${state.index}]]</td>

**LẤY SỐ LƯỢNG (TÍNH TỪ 1)**

<td>[[${state.count}]]</td>

**Scope**

**Model:** là 1 Scope giống Request.. phạm vi chia sẻ dữ liệu rất nhỏ.. chỉ chia sẻ giữa Controller và View

**Cookie:** mẫu text nhỏ (<4kb) **được lưu trên Client do trình duyệt quản lí** và cho phép truyền thông với Server.

Có thể xem Cookie là 1 phạm vi chia sẻ dữ liệu cho nhiều phiên làm việc trên 1 máy khách của 1 website trong 1 khoảng thời gian vì dữ liệu được duy trì ngay cả khi shutdown máy   
- cách xóa Cookie: ở server tạo Cookie có thời hạn = 0 gửi về CLient

|  |  |  |
| --- | --- | --- |
| **Request:**  **HttpServerletRequest** | **Session:**  **HttpSession** | **Application:**  **ServletContext** |
| **request.setAttribute(); request.getAttribute(); request.removeAttribute();** | **session.setAttribute(); session.getAttribute(); session.removeAttribute();** | **application.setAttribute(); application.getAttribute(); application.removeAttribute();** |
| **Chia sẻ dữ liệu giữa các Component hoạt động trên cùng 1 Request**  **request sẽ được giải phóng khỏi bộ nhớ sau khi Request kết thúc** | **Chia sẻ dữ liệu giữa các Component hoạt động trên cùng 1 phiên làm việc**  **Nơi lưu trữ dữ liệu theo phiên làm việc ( có hiệu lực từ lúc kết nối website 🡪 đóng trình duyêt, hoặc sau khoảng tgian ko truy xuất vào website)**   * **20p sau sẽ timeout** | **Nơi lưu trữ dữ liệu dùng chung cho các hoạt động trên toàn ứng dụng web (bất kì Component nào cũng có thể tạo và đọc Attribute)** |

(Ioc) Tại sao người ta viết = interface mà ko viết = class để nạp vào môi trường ?

* Tạo cho ta 1 ứng dụng có thể tháo lắp các chức năng mà ko cần phải lập trình lại từ đầu

Security

Authentication: kiểu như cái cửa chặn lại, muốn vào phải check

Authorizing: giống như thẩm quyền, vào để làm gì ? ngoài ra liên quan tới đóng 1 vai trò, nhắc tới author thì nó bao hàm luôn cả Authen

Phải có @Configuration

Extends WebSecurityConfigurerAdapter

@EnableWebSecurity

Nếu mún bảo vệ bên trong method thì bật

@EnableGlobalMethodSecurity(prePostEnabled = true)

* Controller method có PreAuthorize(“isAuthenticated()”)