|  |  |
| --- | --- |
| Data binding | * Là cơ chế liên kết dữ liệu đầu vào hoặc đầu ra với model, giúp việc tương tác với dữ liệu dễ dàng hơn * Khi tương tác với form dữ liệu thì dữ liệu trên form được tự động chuyển đổi thành các thuộc tính của đối tượng liên kết với nó (model) * Hỗ trợ chuyển đổi dữ liệu và validate dữ liệu |
| Thẻ form | * Được sử dụng để tạo form * Khai báo bằng “taglib form” * Hỗ trợ tạo các thẻ button, input, hidden,.. (tương tự như trong html) * Thuộc tính quan trọng trong thẻ form:   + path: tên thuộc tính được liên kết với trường hiện tại  + item: danh sách để hiển thị đối với thẻ có đa dữ liệu (select, option,…)   * ModelAttribute dùng để liên kết model với 1 form |
|  | Thymeleaf là gì?   * Thymeleaf là một bộ xử lý view được sử dụng cho các ứng dụng web và các ứng dụng độc lập. Dùng để xử lý HTML, XML, JavaScript, CSS và cả Text. * Thymeleaf cung cấp một cách thức xây dựng template (mẫu) hiện đại và dễ bảo trì. * Thymeleaf được xây dựng phù hợp với các tiêu chuẩn của web, đặc biệt là HTML5. |
|  | Các loại biểu thức trong Thymeleaf   * Biểu thức vs biến: ${} * Biểu thức với thuộc tính: \*{} * Biểu thức với Message: #{} * Biểu thức với URL: @{} * Biểu thức với phân đoạn: ~{} |
|  | Cú pháp vòng lặp trong Thymeleaf   * Gồm 2 biến: phần tử, trạng thái * Biến trạng thái iState:   Index: chỉ số vòng lặp (bắt đầu 0)  Count: số phần tử đã đc xử lý  Size: tổng số phần tử trong danh sách  Even/odd: kiểm tra index của vòng lặp hiện tại là chẵn hay lẽ  First: kiểm tra có phải lần lặp đầu tiên k  Last: kiểm tra có phải lần lặp cuối hay k  <**someHtmlTag** th:each="item, iState : ${items}">  .....  </**someHtmlTag**> |
|  | Cách tạo layout trong Thymeleaf   * Sử dụng thuộc tính th:fragment để tạo * Sử dụng th:insert, th:replace, th:include để sử dụng * Sử dụng fragment có 6 tham số, các tham số này giúp định hình nên giao diện đầy đủ của một trang. |
|  | Add ViewResolver cho Thymeleaf (Config Thymeleaf bằng Java) |