*MindMap Module4*

1. *[Tổng quan Spirng MVC](#MVC)*
2. *[Spring Controller](#Controller)*
3. *[View & ThymeLeaf](#thyme)*
4. *[DataBinding & Form](#databinding)*
5. *[Thao tác với DB và ORM](#orm)*
6. *[JPA](#JPa)*
7. *[Spring Data Repository](#SpringdataReository)*
8. *[Validation](#validation)*
9. *[AOP & Exception Handling](#AOP)*
10. *[Session & Cookie](#sessioncookie)*
11. *[WebService & RESTful](#webservice)*
12. *[jQuery & AJAX](#ajax)*
13. *[I18N](#I18N)*
14. *[Security , Authentication, Authorization](#Security)*

*Tổng quan Spring MVC*

1. Framework

* Là ứng dụng phần mềm có tính trừu tượng , cung cấp các tính năng chung, thông dụng có thể tùy biến để tạo nên những ứng dụng cụ thể khác nhau
* Mỗi framework cung cấp các phương pháp riêng biệt để xây dựng và triển khai ứng dụng
* Mỗi framework bao gồm 1 môi trường tổng thể, tái sử dụng được, nhằm cung cấp các chức năng và công cụ để hỗ trợ quá trình phát triển ứng dụng

1. Điểm khác biệt giữa framework và library :

* Là cơ chế Inversion Of Control ( IoC) .
* Framework sẽ nắm quyền điều khiển
* Library: lập trình viên nắm quyền điều khiển .

1. Spring Framework :

* Là 1 framework cung cấp mô hình đầy đủ để phát triển các hệ thống java lớn, giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử và tái sử dụng

1. Luồng thực thi của request trong mô hình Spring MVC :

* Request từ client sẽ gửi tới Front Controller ( đây là điểm duy nhất nhận request từ client và trả respone lại cho client )
* Tại Front Controller thì các request sẽ được chuyển tới các controller tương ứng thông qua cơ chế mapping mà chúng ta khai báo trong handle mapping
* Sau khi controller nhận request thì sẽ xử lí các nghiệp vụ logic và sau đó sẽ trả respone về cho front controller
* Front Controller nhận respone sẽ truyền tới view để tìm view tương ứng và sử dụng view cơ chế view resolve để truyền model vào view đó .
* View sẽ trả về cho front controller và front controller sẽ gửi trả về client để hiển thị cho người dùng

*Spring Controller*

1. Các cách tạo Controller

* Trước 2.5 thì implements interface Controller
* Dùng @Controller hoặc @RestController
* @Controller thường dùng để định nghĩa 1 class là controller trong web truyền thống. Và kết quả trả về là view
* @RestController thì được dùng trong webService kết quả trả về là XML hoặc JSON

1. RequestMapping

* @RequestMapping dùng để ánh xạ các request tới các action tương ứng của controller
* Dùng được cho cả lớp và phương thức
* 1 số biến thể :
* GetMapping
* PostMapping
* DeleteMapping
* PutMapping
* PatchMapping

1. So sánh @ResquestParam & @PathVariable

|  |  |
| --- | --- |
| @RequestParam | @PathVariable |
| * Đều có thể dùng để lấy dữ liệu từ client gửi đến server | |
| * Có thể nhận dữ liệu từ cả phương thức Get hoặc Post * Dữ liệu được lấy theo cặp key-value * Dùng nhiều trong web truyền thống | * Chỉ có thể lấy dữ liệu từ phương thức Get * Chỉ cần lấy value * Dùng nhiều trong web service |

*View & Thymeleaf*

1. View

* Là 1 trong 3 phần của Spring MVC , có nhiệm vụ hiển thị giao diện và tương tác với người dùng
* Trong Spring MVC View nằm ở phần Back-End
* ViewResolve là cơ chế xử lí tầng view của Spring MVC , ánh xạ tên của view sang đối tượng view tương ứng.
* ViewResolve là bộ phận quyết định Front Controller sẽ trả về view nào cho client

1. Thymeleaf

* Là bộ xử lí view được sử dụng trong các ứng dụng web và ứng dụng độc lập
* Có thể xử lý HTML CSS JS XMl và text
* Cung cấp 1 cách thức xây dựng template mẫu hiện đại và dễ bảo trì
* Được xây dựng phù hợp với tiêu chuẩn của web

1. Các biểu thức của thymeleaf

* Biểu thức với biến ${}
* Biểu thức với thuộc tính \*{}
* Biểu thức với message #{}
* Biểu thức với URL @{}
* Biểu thức với fragment ~{}

1. Layout

* Là bố cục cố định của trang web . thay đội nội dung thì bố cục vẫn không thay đổi

*DataBinding & Form*

1. DataBinding

* Là cơ chế liên kết dữ liệu đầu vào hoặc đầu ra với các đôi tượng model
* Giúp việc liên kết dữ liệu dễ dàng
* Sử dụng databinding thì các form đều được liên kết với đối tượng đc biểu diễn ở phía sau nó
* Khi tương tác với form dữ liệu sẽ được chuyển hóa thành thuộc tính của đối tượng liên kết với nó
* Hỗ trợ việc chuyển đổi dữ và validate dữ liệu

1. Cơ chế Databinding

* Dữ liệu do người dùng nhập vào thông qua cơ chế databinder sẽ chuyển thành Bean nếu như dữ liệu đúng và thành bindingresult nếu dữ liệu sai
* Databinder bao gồm : Validator , PropertyEditor , Formatter, Conversion service.

1. Các thành phần quan trọng của form trong databinding

* ModelAttribute : để liên kết đối tượng với form
* Path là tên thuộc tính của đối tượng
* Items là danh sách các mục của trường cần lựa chọn như select, radio, checkbox

*Thao tác với DB và ORM*

1. ORM ( Object relational mapping)

* Là kĩ thuật liên kết các đối tượng trong lập trình với các đối tượng trong cơ sở dữ liệu
* ORM cho phép truy xuất dễ dàng đến DB thông qua các đối tượng lập trình
* ORM giúp lập trình viên tập trung thao tác với đối tượng mà không cần quan tâm đến DB đang xử dụng.

1. Ưu nhược điểm của ORM

* Ưu điểm : quản lí dữ liệu tập trung ở code , các thao tác với dữ liệu được thực hiện tự động, tránh lỗi cú pháp SQL, hỗ trợ transaction, có thể cache dữ liệu giúp truy xuất nhanh hơn
* Nhược điểm : với dữ liệu lớn khả năng truy xuất bị hạn chế, dễ bị giảm hiệu năng của hệ thống khi việc truy xuất dữ liệu quá dễ dàng , đối với các thao tác phức tạp cần vẫn cần phải sử dụng đến SQL thuần.

1. Entity ?

* Là đối tượng đại diện cho dữ liệu trong ứng dụng, thường là các POJOs được sử dụng để ánh xạ tới 1 đối tượng trong DataBase

1. Điều kiện để trở thành Entity

* Được gán @Entity
* Lớp không được khai báo là final
* Có 1 constructor không có tham số
* Các thuộc tính không được khai báo access modifier là public

*JPA*

1. JPA ( Java persistence API )

* Là tập hợp các interface mô tả cách tạo entity, quản lí entity, cách sử dụng entity để truy vấn dữ liệu và quản lí mối quan hệ giữa cách entity.

1. Hibernate

* Là 1 ORM framework dùng để triển khai những gì mà JPA mô tả
* Giúp lập trình viên viết các ứng dụng java có thể map đối tượng lập trình với đối tượng trong db

1. Interface Entity Manager có nhiệm vụ :

* Khai baó phương thức để tương tác với persistence context ( tập hợp các thể hiện của entity được quản lí , tồn tại trong 1 kho dữ liệu )
* Tạo hóa xóa thể hiện của entity
* Tìm kiếm entity theo khóa chính
* Thực hện câu truy vấn lên Entity

1. Câu truy vấn tĩnh / động.

* Phương thức createQuery() của interface EntityManager tạo ra câu truy vấn động
* Phương thức createNameQuery() tạo ra câu truy vấn tĩnh
* Sự khác nhau giữa 2 câu truy vấn này : Câu truy vấn tĩnh thì được query 1 lần và kết quả sẽ được sử dụng tới khi tắt chương trình. Câu truy vấn động thì mỗi lần gọi thì sẽ sinh ra 1 câu query mới và đem đi truy vấn

*Spring Data Repository*

1. Spring data repository

* Là thành phần bổ trợ cho JPA thực hiện các thao tác CRUD, sắp xếp , phân trang.
* Giúp giảm thiểu lượng code lặp đi lặp lại trong qua trình truy xuất dữ liệu

1. Các Interface chính :

* Repository : đánh dấu cho các interface kế thừa sẽ có chức năng , nhiệm vụ tương tác với database
* CRUD repository : khai báo các phương thức CRUD
* PagingAndSorting : khai báo các phương thức sử dụng cho sắp xếp và phân trang
* JPA repository :hỗ trợ trả về kiểu List thay cho Iterable và object thay cho Optional , hỗ trợ đa luồng , flush()

1. Các cách tạo câu truy vấn :

* Sử dụng @Query : có thể sử dụng HQL , hoặc SQL với khai báo kèm theo nativeQuery = true
* Sử dụng tên phương thức

1. So sánh Pageable vs slice

|  |  |
| --- | --- |
| Pageable | Slice |
| * Đều được sử dụng để hỗ trợ phân trang | |
| * Biết được tổng số trang * Không thích hợp khi sử dụng cho dữ liệu lớn Vì cần phải đếm từng record để tính ra được tổng số trang làm chậm ứng dụng | * Không biết được tổng số trang * Thích hợp sử dụng cho dữ liệu lớn khi phân trang vì không cần phải đếm số record để tính ra tống số trang. |

1. Converter/ Formatter

|  |  |
| --- | --- |
| Coverter | Formatter |
| * Hỗ trợ chuyển đổi dữ liệu nhập vào sang kiểu dữ liệu thích hợp | |
| * Là thành phần sử dụng chung cho toàn hệ thống. | * Sử dụng ở tầng web (databinder) |

*Validation*

1. Validation là gì ?

* Xác minh tính đúng đắn, hợp lệ của dữ liệu mà người dùng nhập vào
* Các tầng có thể validate :
* UI ( HTML , JS)
* Tầng Bussiness ( Spring hỗ trợ)
* Tầng Database ( các ràng buộc )

1. Các cách validate ?

* Sử dụng các annotation có sẵn
* Sử dụng custom validation
* Sử dụng annotation tự tạo

*AOP & Exception Handling*

1. AOP ? (Aspect oriented programming)

* Hay còn gọi là lập trình hướng khía cạnh . là 1 kỹ thuật lập trình ( tương tự OOP) nhằm mục đích phân tách chương trình thành nhiều các module riêng lẻ, không phụ thuộc vào nhau
* Khi hoạt động thì chương trình tự kết hợp các module lại và chạy
* Khi thay đổi 1 chức năng chỉ cần thay đổi 1 module
* Dùng để bổ trợ OOP

1. Concern

* Core - concern là yêu cầu chính và logic nghiệp vụ chính của chương trình
* Cross cutting concern là những xử lý phụ được thực hiện của chương trình khi core-concern được gọi

1. Một số thuật ngữ
2. Aspect :

* Là những mối quan tâm xuyên suốt trong AOP
* AOP sử dụng để phân tách các module, mỗi module đảm nhận 1 nhiệm vụ nhất định
* Aspect là dùng để xây dựng những nhiệm vụ này
* Mỗi aspect sẽ bao gồm các advice và join point

1. Advice :

* Là một hành động cụ thể của aspect
* Những xử lí phụ được thêm vào xử lý chính. Code của các xử lí đó được gọi là advice.
* Các loại advice : before advice, after advice, after returning advice, after throwing advice, around advice

1. Join point :

* Là điểm nào đó trong luồng thực thi của chương trình mà các advice đc đan vào .

1. Point cut :

* Là tập hợp các join point để AOP đan advice vào nghiệp vụ chính

1. Exception Handling

* Mục đích là không hiển thị các exception ra client
* Bắt được ngoại lệ đúng lúc và có hướng xử lí ngoại lệ phù hợp

1. Các loại ngoại lệ :

* Loại 1 : khi người dùng thao tác lại sẽ không sinh ra ngoại lệ nữa .. các ngoại lệ này xảy ra thường là do thư viện và nghiệp vụ .
* Loại 2 : khi gặp lại ngoại lệ vẫn xảy ra là những ngoại lệ từ hệ thống
* Loại 3 : ngoại lệ từ chối request là do ngoại lệ từ framework

1. Các cách xử lí ngoại lệ :

* Sử dụng try-catch
* Sử dụng @ExceptionHandler
* Sử dụng HandlerExceptionResolve
* Sử dụng error-page

*Session & Cookie*

1. Cookie

* Là tập tin được trang web mà người dùng truy cập tạo ra
* Cookie giúp người dùng trải nghiệm tốt hơn = cách lưu lại thông tin duyệt web
* Với Cookie các trang web có thể duy trì trạng thái đăng nhập , giỏ hàng v.v….
* Chỉ có thể lưu dữ liệu dạng chuỗi

1. Session

* Gọi chung là phiên làm việc giữa client và server
* 1 session bắt đầu khi client thực hiện 1 request đầu tiên tới server và kết thúc khi client dừng làm việc
* Trong phiên làm việc có thể có biến Session được tạo ra và lưu trên server
* Người dùng có thể truy cập giá trị của biến session khi : Không đóng trình duyệt ,không log out , và không tương tác với trình duyệt ít hơn thgian mà admin quy định ( thường là 30 phút )

1. So sánh

|  |  |
| --- | --- |
| Session | Cookie |
| * Đều tạo ra dữ liệu mà người dùng có thể sử dụng khi tương tác với web | |
| * Được lưu ở trên server * Bảo mật tốt hơn * Có thể lưu được nhiều kiểu dữ liệu * Thời hạn là trong 1 phiên làm việc * Hoạt động dựa cookie | * Được lưu ở client * Bảo mật kém hơn * Chỉ lưu được dạng chuỗi * Thời hạn sống có thể được thiết lập thông qua setMaxAge |

*Webservice & RESTful*

1. Webservice

* Là thành phần ứng dụng được hiển thị dưới dạng các dịch vụ trên www
* Có thể được sử dụng để tích hợp các ứng dụng được viết bằng các ngôn ngữ khác nhau và chạy trên các nền tảng khác nhau
* Xây dựng các chuẩn mở và các giao thức mở để giao tiếp
* Hoạt động như 1 server trong mô hình client - server chỉ sử dụng các giao thức HTTTP/HTTPs và chỉ thực hiện 1 tác vụ cụ thể
* Với dữ liệu đầu vào xác định thì webservice xử lý nghiệp vụ và trả về kết quả đảm bảo là mọi ứng dụng có thể hiểu và sử dụng mà không cần quan tâm đến thiết bị , hệ điều hành , ngôn ngữ sử dụng
* Kiểu dữ liêu trả về phổ biến là XML hoặc JSON

1. Phân biệt Webservice và web truyền thống

|  |  |
| --- | --- |
| Web | Web Service |
| * Có giao diện người dùng * Hoạt động đa nền tảng * Được sử dụng bởi con người * Được truy cập qua các thành phần của HTML như form,button,link v.v…. * Là 1 ứng dụng đầu cuối .Người dùng truy cập thông qua URL và kết quả nhận về những dữ liệu dạng text , hình ảnh , âm thanh vv.v… | * Không có giao diện người dùng * Độc lập nền tảng * Các ứng dụng tương tác với nhau qua internet * Được truy cập bởi các phương thức Get,Post,Put,Delete,Patch * Dữ liệu trả về thường là JSON hoặc XML người dùng thường khó hiểu nên được webservice chế biến thành dữ liệu mà người dùng có thể hiểu được |

1. Ưu nhược điểm của webservice ?
   1. Ưu điểm :

* Vì độc lập nền tảng nên có thể hoạt động được trên nhiều nền tảng , nhiều hệ điều hành, ngôn ngữ khác nhau
* Khả năng tái sử dụng cao
* Tạo mối liên kết mềm dẻo giữa các phần mềm nên dễ dàng phát triển ứng dụng phân tán
* Giảm thời gian,chi phí phát triển hệ thống
  1. Nhược điểm :
* Khi 1 web service chết hoặc dừng hoạt động sẽ gây lỗi và thiệt hại lớn đến toàn bộ hệ thống đang sử dụng web service đó
* Cần quan tâm đến vấn đề an toàn . bảo mật nhiều hơn
* Có quá nhiều chuẩn => khó nắm bắt

1. Các loại cơ bản :
   1. SOAP ( Simple object access protocol) : là giao thức sử dụng XML để định nghĩa dữ liệu thuần văn bản (plaintext) và truyền dữ liệu thông qua HTTP
   2. REST (Representational state transfer) : là 1 kiểu cấu trúc cung cấp các quy tắc để xây dựng web service . REST định nghĩa dữ liệu thông qua XML hoặc JSON và truyền thông qua internet sử dụng HTTP. Webservice xây dựng dựa trên REST gọi là RESTful . Chúng chủ yếu xử lí các hoạt động CRUD

*jQuery & AJAX*

1. jQuery

* Là thư viện phổ biến của Javascript. Hỗ trợ thao tác với HTML , DOM , CSS , xử lý các event, hỗ trợ các hiệu ứng, hỗ trợ ajax, và hỗ trợ hầu hết các trình duyệt
* Châm ngôn của jQuery WRITE LESS - DO MORE

1. AJAX

* Viết tắt của Asynchronous Javascript and XML
* Là một nhóm các công nghệ web được sử dụng để phát triển các ứng dụng web giàu tính tương tác, nhanh hơn, mượt hơn với sự giúp đỡ của JS , HTML , CSS và XML
* AJAX cho phép web cập nhật 1 cách bất đồng bộ nhờ việc trao đổi 1 lượng dữ liệu nhỏ mà không cần tải lại
* Với AJAX người dùng có thể tiếp tục sử dụng ứng dụng trong khi các chương trình trên client gửi request đến server

1. Các thành phần của AJAX

* HTML CSS : dùng để hiển thị dữ liệu
* JS
* DOM (Document Object Model) : tương tác với thông tin được hiển thị thông qua JS
* XML Http Request : Trao đổi dữ liệu 1 cách bất đồng bộ với server
* XML/JSON định dạng dữ liệu nhận và gửi

1. Ưu/ Nhược của AJAX

* Ưu : Chỉ yêu cầu câp nhật các nội dung cần thiết nên giảm lượng lớn băng thông và tgian load trang. Sử dụng bất đồng bộ nên tăng trải nghiệm , tương tác cao hơn vì không cần load lại trang. Giảm kết nối đến server
* Nhược : Khó khăn cho việc bookmark 1 trạng thái nào đó của web.K hông lưu lại lịch sử duyệt web.Các trình duyệt không hỗ trợ hoặc tắt JS sẽ không sử dụng đc AJAX. Dễ bị tấn công bởi các mã độc

*I18N*

1. I18N ( Internationalization ) :

* Là quá trình thiết kế ứng dụng có thể đáp ứng nhiều ngôn ngữ và khu vực khác nhau mà không cần phải thay đổi kĩ thuật
* Còn được gọi là Translation / localization enablement

1. L10N ( Localization) :

* Là quá trình điều chỉnh phần mềm đã được quốc tế hóa cho 1 ngôn ngữ hoặc khu vực cụ thể bằng cách chỉ định ngôn ngữ hoặc khu vực sau đó dịch văn bản
* I18N + L10N = G11N ( Globalization )

1. Interceptor :

* Là đối tượng cho phép đính kèm vào vòng đời của request có khả năng xử lí tiền , hậu request được dùng phổ biến trong I18N , được dùng phổ biến để intertionalize

1. LocaleResolver

* Là interface để triển khai phân giải các message theo locale trên nền web. Có thể phân giải thông tin về locale dựa trên request, session cookie

*Spring Security*

1. Authenticaction ( xác thực )

* Là hành động nhằm vào việc thiết lập hoặc chứng thực 1 thông điệp, đối tượng nào đó là đáng tin cậy
* Cũng có nghĩa là công nhận nguồn gốc của đối tượng. Để xác thực thì người dùng có thể sử dụng nhiều ứng dụng khác nhau

1. Authorization ( phân quyền )

* Là quá trình xác định xem 1 người dùng có quyền truy cập 1 tài nguyên cụ thể nào đó để thực hiện một số hành động hay không
* Cấp quyền chỉ xảy ra sau khi được xác thực

1. Các cơ chế xác thực :
   1. HTTP Basic :

* Là phương thức mà client gửi username + password khi thực hiện yêu cầu
* Để sử dụng HTTP Basic thì client phải gửi tiêu đề cấp phép và các yêu cầu với server. Username + password sẽ không được mã hóa nhưng được xây dựng bằng cách nối username vs password lại với nhau chuỗi này mã hóa basic64 từ khoá basic được đặt trước chuỗi
  1. Cookie :
* Lưu thông tin đăng nhập = cookie để gửi lên server trong mỗi request. Dễ bị tấn công CSRF và không tương thích với web service
  1. Signature
* Client gửi request + public key tới server , server sẽ dùng private key để giải mã
  1. Certificate
* Tương tự với signature nhưng cặp key này được cơ quan chứng thực ( Certification Authority ) có thẩm quyền cung cấp cho người dùng,
  1. OTP ( One Time Password ) :
* Mật khẩu dùng 1 lần có thời gian nhất định
* Thường được dùng như xác thực 2 yếu tố trong các ứng dụng
  1. OAuth2
* Là một giao thức mở . được sử sụng để người dùng có thể nhờ bên thứ 3 xác thực.
  1. JWT
* Là 1 tiêu chuẩn mở ( RFC-7519 ) định nghĩa cách thức trao đổi thông tin an toàn = JSON . TIn này được xác thực và đánh dấu tin cậy = chữ kí của nó . chữ kí của nó thường được mã hóa HMAC hoặc RSA
* JWT gồm 3 phần :
* Header : gồm 2 phần chính loại token là gì , và thuật toán dùng để mã hóa .
* Payload : thường chứa nội dung , ngoài ra còn lưu thêm 1 số data phụ trợ như khi nào token hết hạn v.v..
* Signature : là 1 chuỗi được mã hóa bao gồm header, payload cùng với 1 chuỗi bí mật theo nguyên tắc :sẽ dùng thuật toán mã hóa base64UrlEncode cho header sau đó cộng với chuỗi mã hóa base64UrlEncode của payload và secret ) cuối cùng sẽ dùng thuật toán HMACSHA256 để mã hóa toàn bộ chuỗi này

1. Spring security

* Là 1 dự án trong hệ sinh thái spring framework cung cấp các dịch vụ bảo mật cho các ứng dụng
* Cung cấp authentication và authorization

1. Các thành phần của Spring security

* SecurityContext : là interface lưu trữ tất cả các chi tiết liên quan đến bảo mật. được kích hoạt cùng lúc với spring framework
* Lớp SecurityContextHolder là lớp được sử dụng để truy cập vào SecurityContext. Lớp này lưu trữ SecurityContext hiện tại của ứng dụng ,gồm chi tiết của các principal đang tương tác với ứng dụng
* Đối tượng Authentication biển diễn thông tin của principal
* UserDetails là Interface đại diện cho 1 principal