**BÀI 6.2: RESFUL WEB SERVICE TRONG SPRING MVC**

**Mối quan hệ One to Many và Many to One**

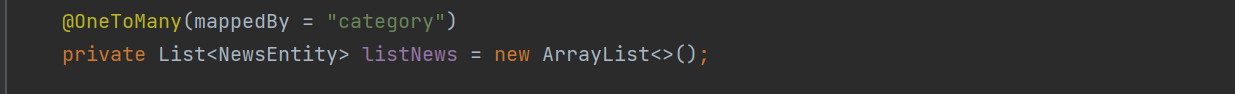
Đầu tiên, để tạo quan hệ mới thì phải đổi sang dạng create-drop. Sau đó thêm Entity và tiến hành khai báo Entity đó trong file dispascher.xml

Trong NewsEntity: ManyToOne

A black rectangle with white text

Description automatically generated

Trong CategoryEntity: OneToMany



Note: Nhớ thêm get set cho thuộc tính được thêm vào. FetchType.LAZY là để tránh trường hợp nó load lên cùng lúc với hệ thống, sẽ làm giảm hiệu suất

**Truyền dẫn dữ liệu qua Ajax**

**A screenshot of a computer program

Description automatically generated**

+ url: đường dẫn để truyền dữ liệu thông qua api

+ type: loại api

+ contentType: loại dữ liệu muốn đưa về cho server (từ client về server)

+ data: Là nội dung dữ liệu truyền về (ban đầu là data – object javascript, sau đó được stringify chuyển sang JSON)

+ dataType: loại dữ liệu muốn trả về cho client (từ server về client)

+ Các thông báo: sucsses hoặc errol

**Sơ lược lại về load dữ liệu ở bài trước**

**A computer screen shot of text

Description automatically generated**

Khi request được yêu cầu thông qua filter đúng với url này thì sẽ hàm showNewsList sẽ được thực thi. Hàm này tạo ra một modelandview với đường dẫn đến file list.jsp. Sau đó thì dữ liệu sẽ được chuyển vào file này thông qua việc gọi tới hàm fileAll được quy định ở Service để lấy về dữ liệu yêu cầu, tiếp tục thêm object model (get được từ http method) và cuối cùng trả ra view

**Các khái niệm phân trang:**

+ totalItem: tổng số item trả về từ database

+ limit: giới hạn số item trên trang

+ totalPage: tổng số trang chia được

+ offset: vị trí của phần tử bắt đầu tại trang đó

Dưa vào đây thì ta sẽ có 2 tham số truyền từ client về: page (cho biết đang đứng ở page nào) và limit (số limit có thể tùy biến dựa vào từng trường hợp)

3 tham số còn lại cần phải tính trong vùng Service. Dựa vào hàm dựng sẵn trong thư viện phân trang, ta có thể thấy 2 tham số quan trọng là visiblePage và totalPage.

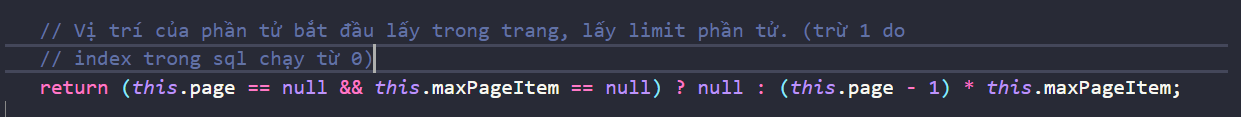
+ visiblePage: số trang hiển thị trên giao diện người dùng (có thể gán cứng)

+ totalPage: tổng số trang chia được (phải lấy từ server về). Do đó mà ở server trong phần servlet, chúng ta có một Interface chuyên trách cho việc này là Pageable

A black square with white text

Description automatically generated with medium confidence

Nó sẽ chứa các hàm tự buid đáp ứng yêu cầu về việc phân trang (và cho cả việc sắp xếp). Chú ý hàm getOffset, hàm này có nhiệm vụ là lấy ra vị trí bắt đầu của phần tử trang đó. Nghĩa là nếu quy định mỗi trang có 3 phần tử, và đang muốn truy xuất vào trang 2 thì hàm này sẽ đi tìm phần tử thứ 4 trong tổng số các phần tử trả về (lưu ý maxPageItem chính là limit)



Sau khi lấy được tất cả các giá trị cần thiết thì tại controller, tiến hành gán vào để truyền các giá trị đó lên cho thư viện twbsPaging để tiến hành phân trang

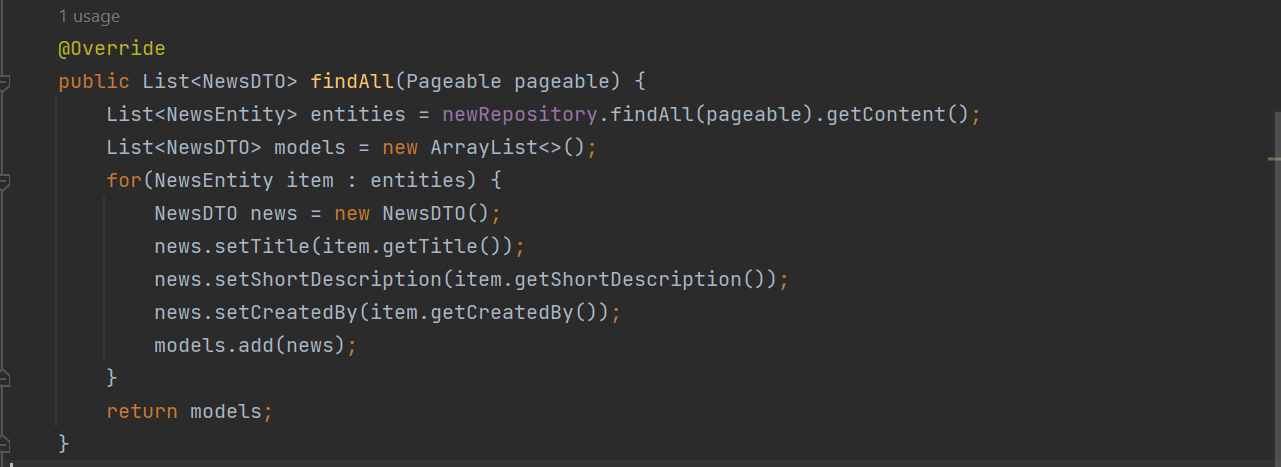
A screen shot of a computer program

Description automatically generated

**Phân trang trong Spring Data JPA**

Spring Data JPA có hỗ trợ thư viện để tiến hành phân trang, cụ thể:

+ Tại NewsService, bỏ phụ thuộc với NewsDAO đi, thay vào đó sử dụng newRepository vì cái này nó hỗ trợ sẵn hàm findAll() để lấy ra tất cả các thông tin của các news. Tuy nhiên cái chúng ta cần là một list NewsDTO (để truyền phát dữ liệu ra bên ngoài) mà hàm findAll() lại trả ra một list Entity. Do đó cần convert lại sang DTO, đơn giản nhất là sử dụng vòng lặp for. Hoặc để tối ưu code, thì tạo thêm một hàm mapper nhận vào tham số là một entity và tiến hành map qua sau đó trả về list DTO (sẽ làm sau)



+ Để chuẩn bị thì cần phải lấy 2 giá trị quan trọng từ client trả về là page (page mà client cần) và limit. Chúng ta sẽ lấy 2 giá trị này ở controller sau đó set vào cho model (nhờ vào @RequestParam) để tiến hành tính toán phân trang bên phía service. Sau đó tạo đối tượng Pageable (như ở servlet) để truyền xuống cho service (lý do trừ đi 1 là vì

A screen shot of a computer program

Description automatically generated

+ Ở serivce thì Spring Data JPA có hỗ trợ luôn một tham số Pageable (không cần build thủ công) có chức năng tương tự trong khóa servlet. Và do đã nhận được từ controller truyền xuống nên nó sẽ có sẵn page và limit, chúng ta sẽ dùng 2 tham số này để tính toán phân trang và trả lại cho client

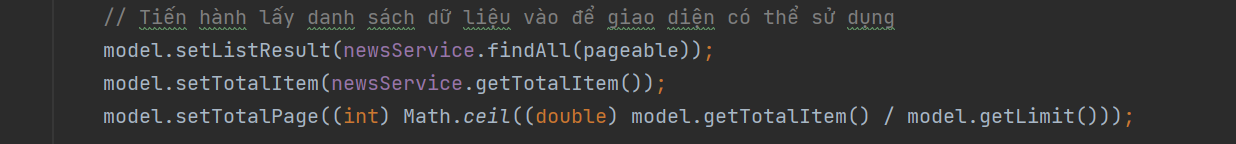


(chỉ việc truyền tham số vào và gọi hàm getContent() là xong – lưu ý phiên bản Spring)

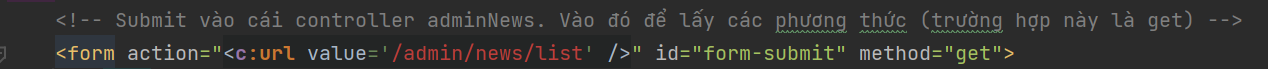
+ Tạo thêm hàm đếm tổng số item ở Service để có thể tính được số phân tử trên trang

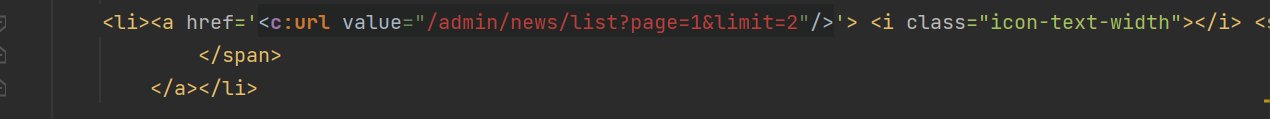


+ Và ngay tại controller, lấy ra số item này và thực hiện tính toán sau đó gán tất cả các giá trị cần thiết truyền vào cho model trước khi trả về client để hiển thị (tổng số trang phân được bằng tổng số item lấy được chia cho giới hạn item mỗi trang)

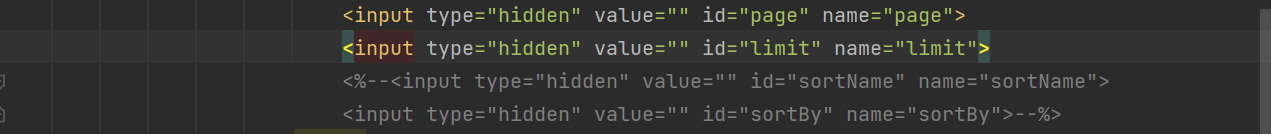


+ Sau đó ở trang list, truyền vào url dẫn đến việc phân trang (phải bao gồm page và limit). Trang nav ở common (link dẫn tới trang list) cũng truyền vào. Mục đích là khi form submit thì form sẽ biết mình phải submit vào controller nào để lấy về dữ liệu phân trang (page và limit sẽ được @RequestParam nhận)





+ Để truyền về được cho controller xử lý thì cần 2 input dạng hiden để đưa xuống controller khi form submit, và 2 input này sẽ được gán giá trị tại sự kiện onclick. Khi submit form thì nó sẽ đi theo data để xuống gặp controller



+ Lưu ý là chỉ submit khi mà page click vào(page) khác với page đang đứng(currentPage), chứ kh sẽ tạo ra vòng lặp vô hạn

A screen shot of a computer code

Description automatically generated

**Cốt lõi của phân trang**

**A screen shot of a computer code

Description automatically generated**

Lý do mà có thể phân trang được chính là do đoạn sql kia, về cơ bản thì thư viện cũng build giống giống thế này.