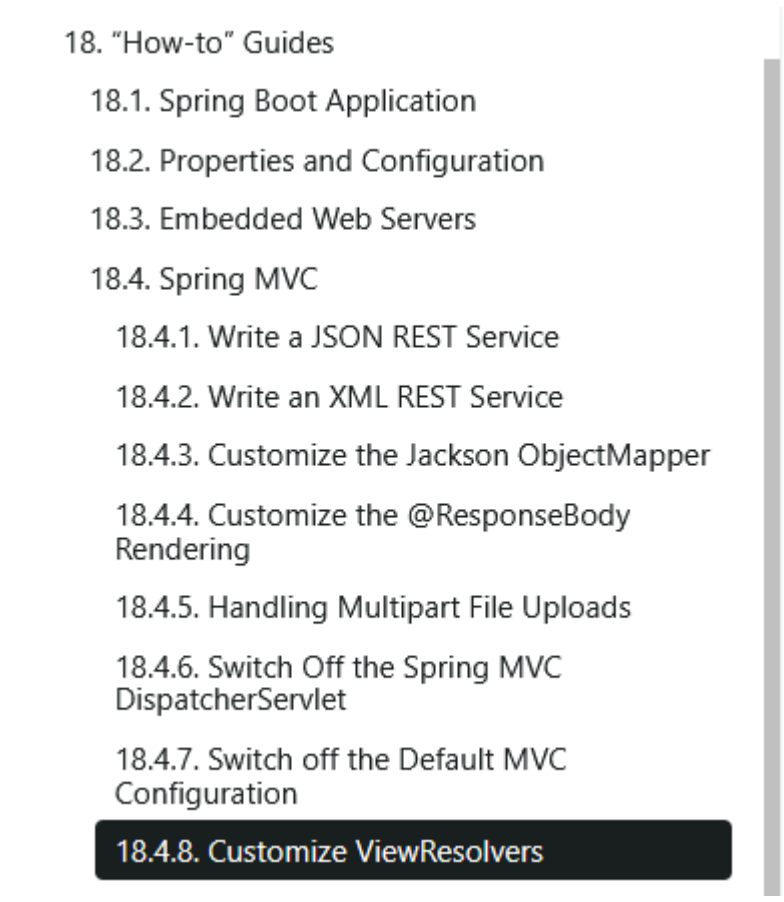
처음에 문제는 아래와 관련해서 시작되었다.

어떻게 함수에서 String형 반환값을 반환하면 그것을 가지고 그 "반환되는 문자열.html" 라는 이름의 페이지에 접근을 하는가? 답은 thymeleaf template engine이고 이 템플릿엔진의 설정을 바꾸면 resources/templates 라는 폴더에서 해당 파일명을 찾지 않고 다른 폴더에서 찾을수 있도록 변경할 수 있다는 것이 강사의 설명이었다. 그러면서 스프링 부트의 메뉴얼을 보고 설정을 변경할 수 있다고 말하였고 나는 이런 설정을 직접 한번 변경해 보는 과정이 큰 공부가 될것이라고 생각하여 한번 변경해보기로 하였고 그 결과는 성공적이었고 많이 배울수 있는 계기가 되었다. 아래는 그 과정에서 내가 알게 된것을 정리한것이다.

1. 첫번째 난관은 메뉴얼의 버전문제와 검색어 문제였다. 방대한 부트 메뉴얼에서 어떻게 검색할지 몰라 처음에는 위에서 나와있는 것과 같이 viewName changing등과 같이 검색하였더니 원하는것이 나오지 않았다. 문제는 검색어에도 있었지만(검색어는 안되면 다른 관련된 것으로 여럿 검색해 보면 결국엔 찾게 된다. 예를들면 thymeleaf) 메뉴얼의 버전상 교제에서 안내해준것에는 관련 자료가 없었던 것이다. 스프링 부트 메뉴얼은 이게 현재 최신이다. 그리고 그냥 구글에 spring boot manual이라고 당연한 검색어로 검색하면 제일 상단에 뜬다.
https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

2. 적절한 검색어(thymeleaf)로 아래 목차에서 적절한 안내글을 발견하였다.



딱봐도 유용한 정보 넘칠듯

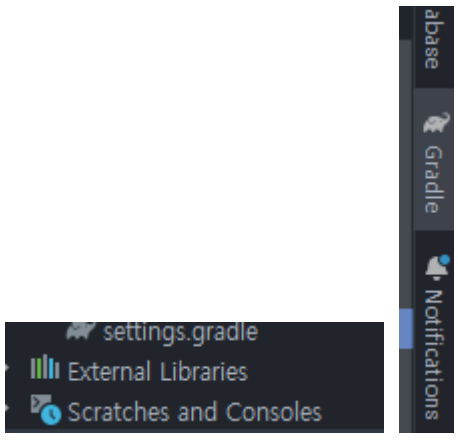
- If you use Thymeleaf, you also have a `ThymeleafViewResolver` named 'thymeleafViewResolver'. It looks for resources by surrounding the view name with a prefix and suffix. The prefix is `spring.thymeleaf.prefix`, and the suffix is `spring.thymeleaf.suffix`. The values of the prefix and suffix default to 'classpath:/templates/' and '.html', respectively. You can override `ThymeleafViewResolver` by providing a bean of the same name.

For more detail, see the following sections:

- `WebMvcAutoConfiguration`
- `ThymeleafAutoConfiguration`
- `FreeMarkerAutoConfiguration`
- `GroovyTemplateAutoConfiguration`

강의에서 **prefix, suffix**와 같은 키워드를 주었기에 망정이지 혼자서는 찾는데 애좀 먹었을것같다. 만약 이러한 키워드가 없었다면 **customize ViewResolvers**라는 것을 보고 여기쯤 있겠구나~ 하고 유추하는 방법이 제일 무난할듯하다.

또한 위의 글을 보면 **ThymeleafViewResolver**을 오버라이딩하면 커스터마이징 할수 있다고 나오는데 여기서 말하는 오버라이딩은 그동안 내가 알아왔던 오버라이딩의 개념하고는 좀 다르다. 같은 이름의 빈을 제공하여 **ThymeleafViewResolver**를 오버라이딩한다는 의미이다. 여기서 또 중요한 삽질을 했는데 나는 아래와 같은 **External Libraries**혹은 **Gradle**의 라이브러리같은 곳에 직접들어가 그 라이브러리 안의 코드를 내가 변경해야 하는줄 알았다. 하지만 이건 매우 어린 생각이다. 외부라이브러리는 적어도 여기서는 수정할 수 없게 되어있다.



그래서 어쩔수 없이 위의 글에서 처럼 주워진 단서인

- If you use Thymeleaf, you also have a `ThymeleafViewResolver` named 'thymeleafViewResolver'. It looks for resources by surrounding the view name with a prefix and suffix. The prefix is `spring.thymeleaf.prefix`, and the suffix is `spring.thymeleaf.suffix`. The values of the prefix and suffix default to 'classpath:/templates/' and '.html', respectively. You can override `ThymeleafViewResolver` by providing a bean of the same name.

를 가지고 주워진 링크로 들어갔다(For more detail, see the following sections:). 그래서 발견하게 된 코드가 전의를 상실시키게 만드는 아래의 코드이다.

https://github.com/spring-projects/spring-boot/blob/v3.2.1/spring-boot-project/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/thymeleaf/ThymeleafAutoConfiguration.java

확인해 보면 알겠지만 이것은 그 방대한 타임리프 라이브러리이다. 여기서 어떻게 앞으로 더 나아갈지 도저히 가늠이 되지 않아 방향을 틀어 구글링을 시작했다. 검색어는 **"ThymeleafViewResolver** 오버라이드**"** 했더니 찾은 글이 아래 글이었고

https://eblo.tistory.com/54

나와있는 설명대로 하였더니 성공적으로 **Thymeleaf engine**이 리소스를 찾는 경로를 변경할 수 있었다. 이후에 여기서 멈추지 않았다. 나는 어떻게든 정형화된 경로로 해결방안을 찾아야 했다. 언제까지 이런 설정변경이 나올 때마다 정처없이 헤메이며 구글링을 할수는 없는 것이니 말이다. 이런 불안정한것이 의욕을 없애는 것은 물론이다**!!**

나는 그래서 위에 나온 소스코드에서

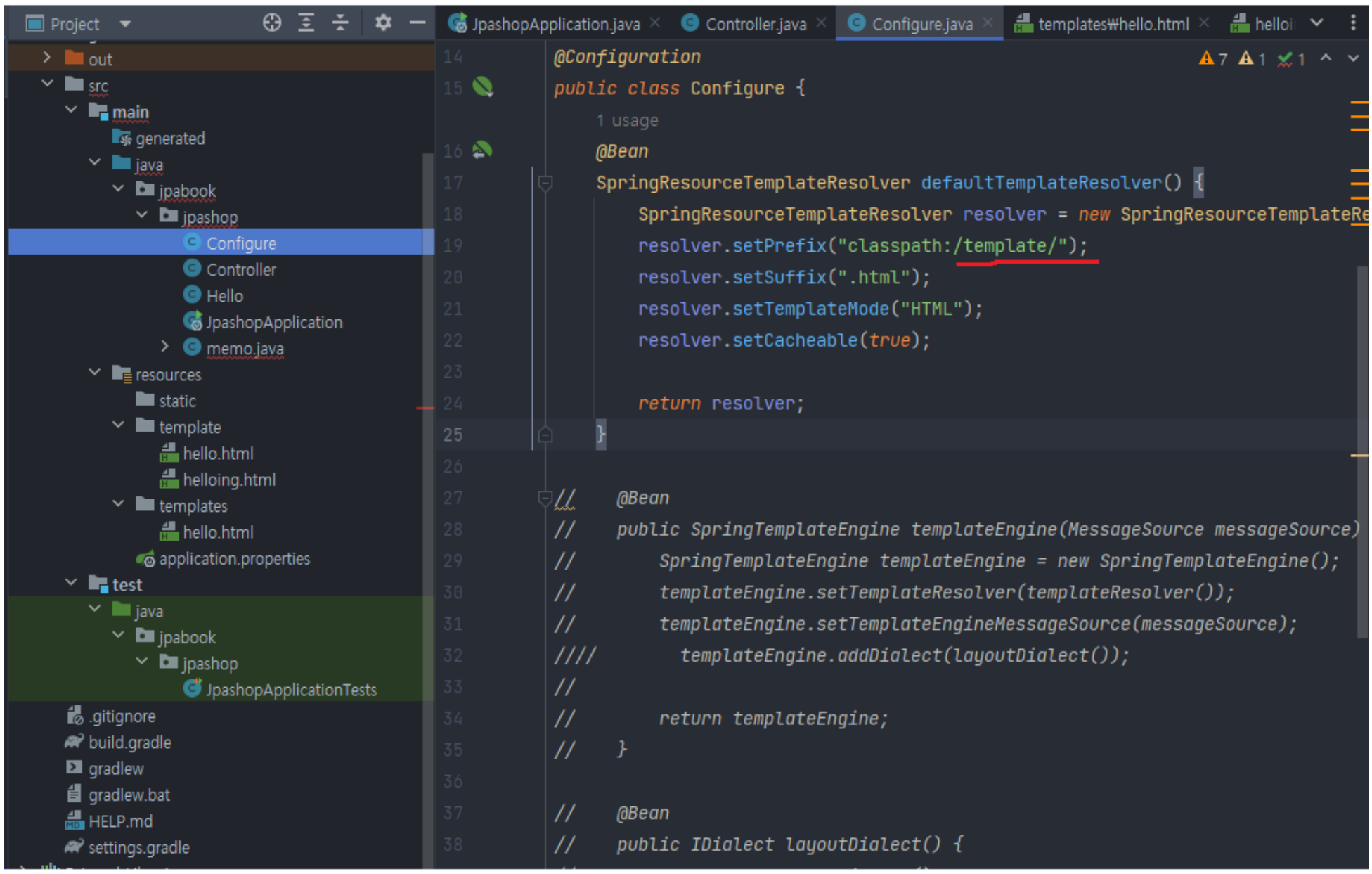
```
@Bean
public SpringResourceTemplateResolver templateResolver() {
    SpringResourceTemplateResolver  templateResolver = new SpringResourceTempl
    templateResolver.setPrefix("classpath:template/");
    templateResolver.setCharacterEncoding("UTF-8");
    templateResolver.setSuffix(".html");
    templateResolver.setTemplateMode("LEGACYHTML5");
    return templateResolver;
}
```

이 부분만을 제외하고 모두 삭제해 버렸다**(**사실 이 안에서도 더 삭제한 것이 있다**).** 사실 스프링부트 **Document**의 설명대로라면 **ThymeleafViewResolver**에 관한 코드는 삭제하면 안된다. 그런데 다행이도 정상적으로 동작하였다**(**이건 어디까지나 내 짧은 경험에 근거하여 맞다고 평가한 것이지 항상 이렇게 전부 삭제해도 될거라는 생각은 하지 않는다**).**

내가 https://github.com/spring-projects/spring-boot/blob/v3.2.1/spring-boot-project/spring-boot-autoconfigure/src/main/java/org/springframework/boot/autoconfigure/thymeleaf/ThymeleafAutoConfiguration.java 에 나와있는 것을 손볼수 없었던 것은 서로 연관되어서 어느 하나를 삭제하면 다른 곳에서 빨간불들어오고 하기때문에 도저히 손을 댈 염두가 나지 않은것이다.하지만 어떻게든 스프링부트 다큐먼트에 나와있는 설명을 참고해서 해당코드를 손보아야 한다. 어쩔수 없는것이다. 다만 이번경험으로 부터 얻은 팁을 좀 적자면 일단 삭제해 가기 전에 해당 변수 혹은 메서드가 어느 것과 연관되어 있는지 끝까지 가보는 노력이 필요하다. 그렇게 하면 결국엔 어떠한 상수값에 수렴하게 되는 경우가 이번에는 거진 전부였다. 그렇게 알아낸 상수값으로 변수를 대체하고 필요한 **setPrefix, setSuffix**함수가 들어있는 **Bean**으로 등록된 함수만 빼고 모두 삭제 했을시에**(**물론 스프링부트 문서에 나와있는 **ThymeleafViewResolver**관련 코드를 삭제하고도 무난히 굴러가는 것은 운이라고 본다**)** 빨간불은 남아있지 않고 웹에서 찾은 코드와 동일한 형태로 만들수 있었다.

이렇게 해야 한다. 웹에서는 찾을 수도 있고 찾지 못할 수도 있는 불확실성의 연속인 것이다. 하지만 메뉴얼에 나와있는 설명을 참고하고 관련코드를 우선은 내가 적절한 형태로 만들어 내는 훈련이 되어야 한다. 그후에도 안되면 또 구글링 하는 것이 맞는 방법이다.

결과화면



위 그림과 같이 나는 경로를 변경했고 만족할 만한 결과를 얻을 수 있었다.