Chapter 01. 스프링의 기능을 활용해보자

Spring Boot Validation

Validation

Validation 이란 프로그래밍에 있어서 가장 필요한 부분 입니다. 특히 Java에서는 null 값에 대해서 접근 하려고 할 때 null pointer exception이 발생 함으로, 이러한 부분을 방지 하기 위해서 미리 검증을 하는 과정을 Validation 이라고 합니다.

단순하게는 아래와 같은 코드들 입니다.

```
public void run(String account, Spring pw, int age){

        if(account == null || pw == null){
            return
        }

        if(age == 0){
            return
        }

        // 정상 Logic
}
```

Validation

1. 검증해야 할 값이 많은 경우 코드의 길이가 길어 진다.
2. 구현에 따라서 달라 질 수 있지만 Service Logic과의 분리가 필요 하다.
3. 흩어져 있는 경우 어디에서 검증을 하는지 알기 어려우며, 재사용의 한계가 있다.
4. 구현에 따라 달라 질 수 있지만, 검증 Logic이 변경 되는 경우 테스트 코드 등 참조하는 클래스에서 Logic이 변경되어야 하는 부분이 발생 할 수 있다.

Validation

@Size	문자 길이 측정	Int Type 불가
@NotNull	null 불가	
@NotEmpty	null, “” 불가	
@NotBlank	null , “” , “ “ 불가	
@Past	과거 날짜	
@PastOrPresent	오늘이거나 과거 날짜	
@Future	미래 날짜	
@FutureOrPresent	오늘이거나 미래 날짜	
@Pattern	정규식 적용	
@Max	최대값	
@Min	최소값	
@AssertTrue / False	별도 Logic 적용	
@Valid	해당 object validation 실행	

Validation

1.gradle dependecies
implementation("org.springframework.boot:spring-boot-starter-validation")

2. bean validation spec
https://beanvalidation.org/2.0-jsr380/

3.핸드폰번호 정규식
"^\\d{2,3}-\\d{3,4}-\\d{4}$"

Chapter 01. 스프링의 기능을 활용해보자

Spring Boot Custom Validation

Validation

@Size	문자 길이 측정	Int Type 불가
@NotNull	null 불가	
@NotEmpty	null, “” 불가	
@NotBlank	null , “” , “ “ 불가	
@Past	과거 날짜	
@PastOrPresent	오늘이거나 과거 날짜	
@Future	미래 날짜	
@FutureOrPresent	오늘이거나 미래 날짜	
@Pattern	정규식 적용	
@Max	최대값	
@Min	최소값	
@AssertTrue / False	별도 Logic 적용	
@Valid	해당 object validation 실행	

Custom Validation

1. AssertTrue / False 와 같은 method 지정을 통해서 Custom Logic 적용 가능

2. ConstraintValidator 를 적용하여 재사용이 가능한 Custom Logic 적용 가능

Chapter 01. 스프링의 기능을 활용해보자

Spring Boot Validation 모범 사례

Chapter 01. 스프링의 기능을 활용해보자

Spring Boot Exception 처리

Exception 처리

Web Application 의 입장에서 바라 보았을때, 에러가 났을 때 내려줄 수 있는 방법은 많지 않다.

1. 에러 페이지

2. 4XX Error or 5XX Error

3. Client가 200 외에 처리를 하지 못 할 때는 200을 내려주고 별도의 에러 Message 전달

Exception 처리

@ControllerAdvice	Global 예외 처리 및 특정 package / Controller 예외처리
@ExceptionHandler	특정 Controller의 예외처리

Chapter 01. 스프링의 기능을 활용해보자

Spring Boot Filter 와 Interceptor

Filter

Filter란 Web Application에서 관리되는 영역으로써 Spring Boot Framework 에서 Client로 부터 오는 요청/응답에 대해서 최초/최종 단계의 위치에 존재하며, 이를 통해서 요청/응답의 정보를 변경하거나, Spring에 의해서 데이터가 변환되기 전의 순수한 Client의 요청/응답 값을 확인 할 수 있다.

유일하게 ServletRequest, ServletResponse 의 객체를 변환 할 수 있다.

주로 Spring Framework 에서는 request / response의 Logging 용도로 활용하거나,
인증과 관련된 Logic 들을 해당 Filter에서 처리 한다.

이를 선/후 처리 함으로써, Service business logic과 분리 시킨다.

이미지 출처 : https://justforchangesake.files.wordpress.com/2014/05/spring-request-lifecycle.jpg

Filter

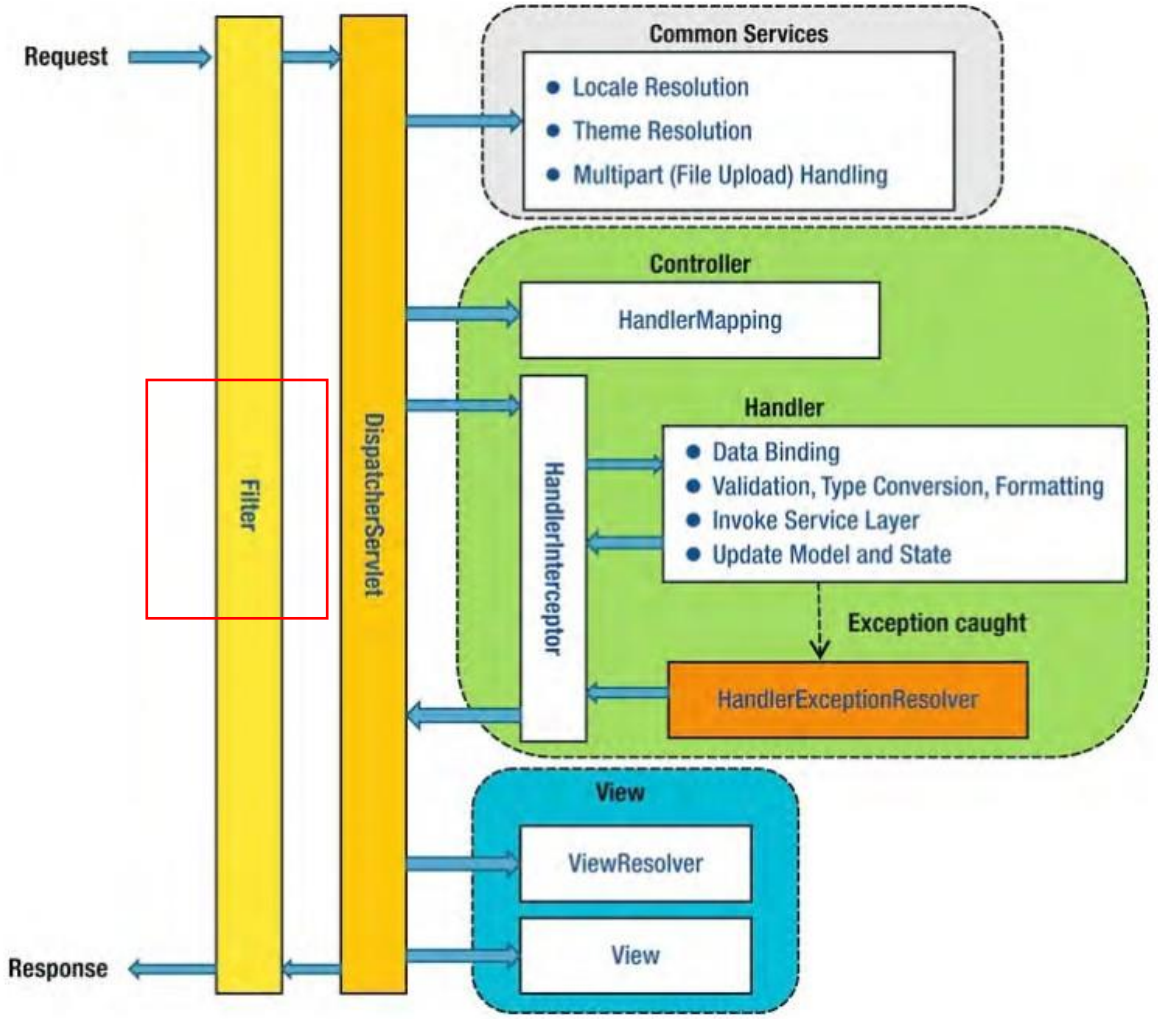

Figure 17-3. Spring MVC request life cycle

이미지 출처 : https://justforchangesake.files.wordpress.com/2014/05/spring-request-lifecycle.jpg

Interceptor

Interceptor 란 Filter와 매우 유사한 형태로 존재 하지만, 차이점은 Spring Context에 등록 된다.

AOP와 유사한 기능을 제공 할 수 있으며,

주로 인증 단계를 처리 하거나, Logging 를 하는 데에 사용한다.

이를 선/후 처리 함으로써, Service business logic과 분리 시킨다.

이미지 출처 : https://justforchangesake.files.wordpress.com/2014/05/spring-request-lifecycle.jpg

Interceptor

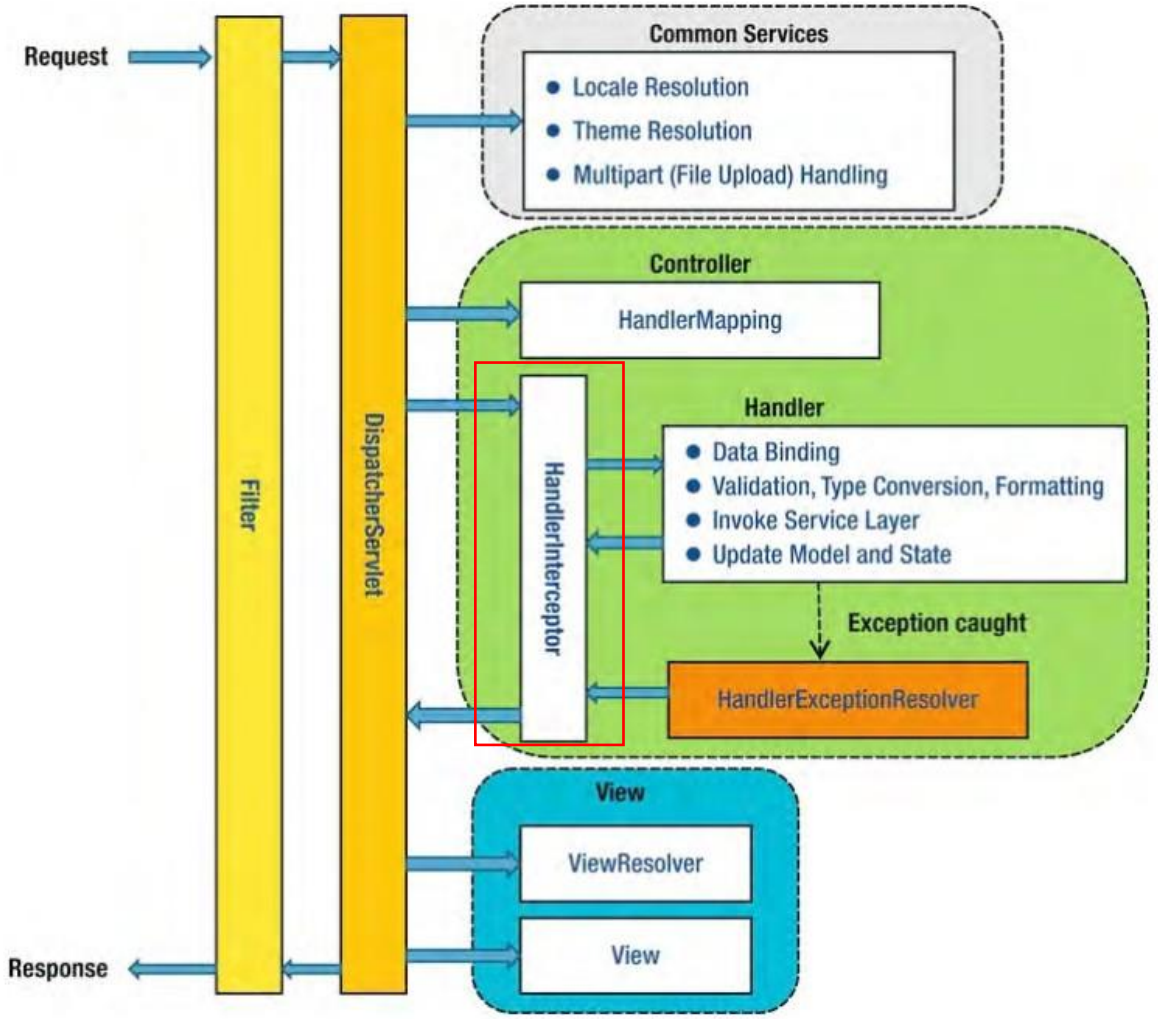

Figure 17-3. Spring MVC request life cycle

이미지 출처 : https://justforchangesake.files.wordpress.com/2014/05/spring-request-lifecycle.jpg

Chapter 06. 스프링의 기능을 활용해보자

비동기 처리하기