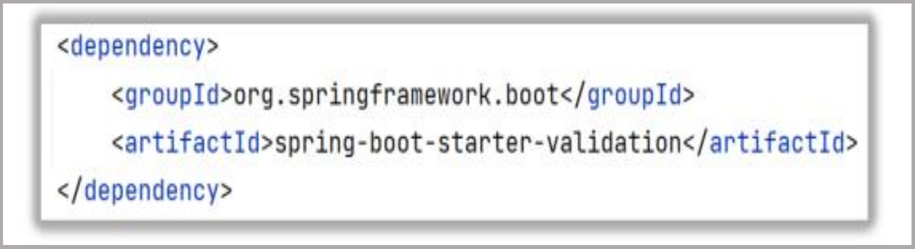
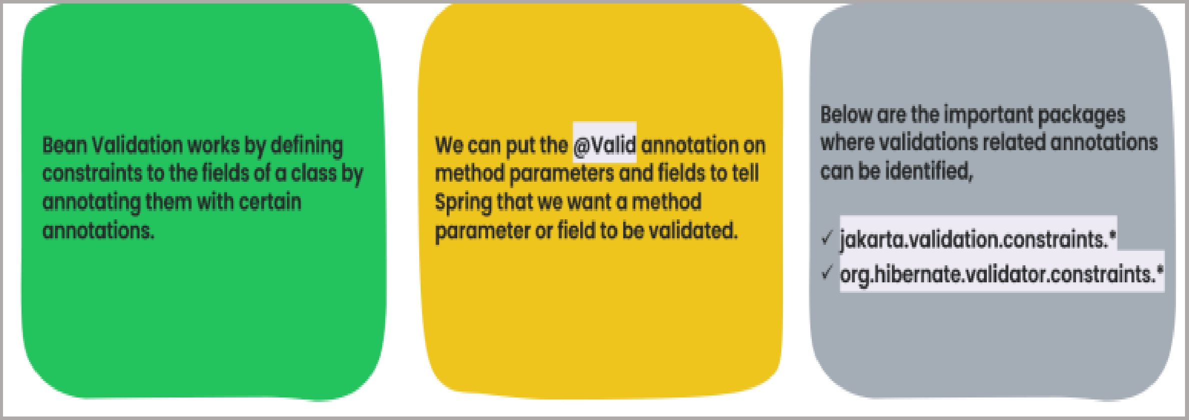
VALIDATION WITH SPRING BOOT

- Bean Validation (https://beanvalidation.org/) is the standard for implementing validations in the Java ecosystem. It's well integrated with Spring and Spring Boot.

- Below is the maven dependency that we can add to implement Bean validations in any Spring/SpringBoot project,



```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```

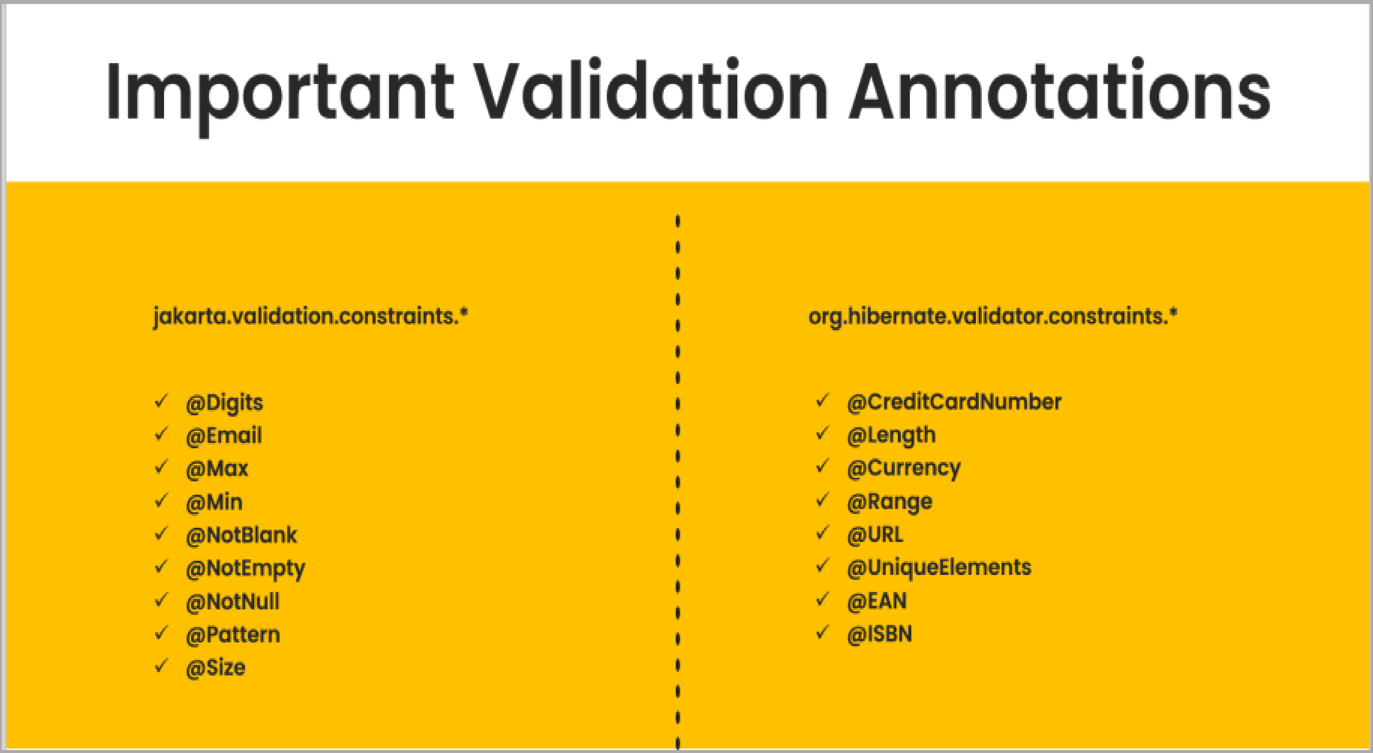


Bean Validation works by defining constraints to the fields of a class by annotating them with certain annotations.

We can put the @Valid annotation on method parameters and fields to tell Spring that we want a method parameter or field to be validated.

Below are the important packages where validations related annotations can be identified,

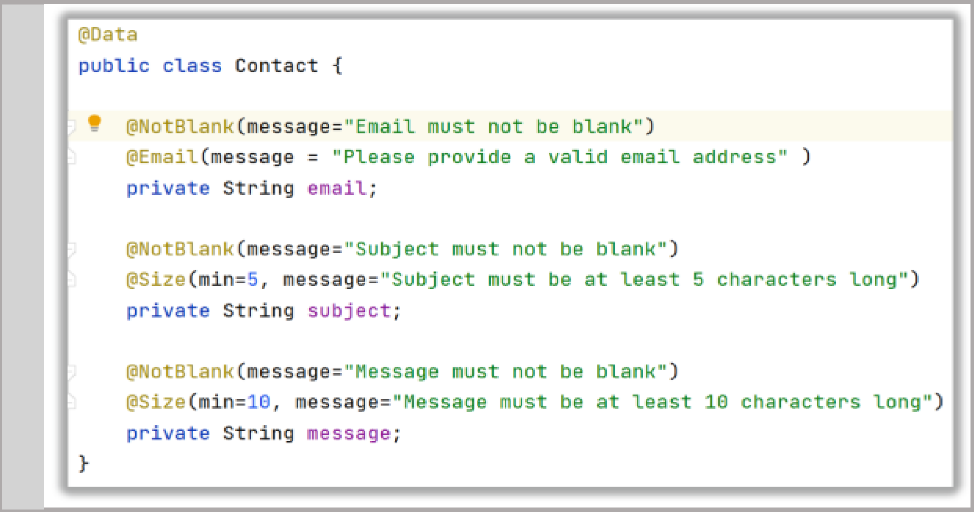
✓ jakarta.validation.constraints.*
✓ org.hibernate.validator.constraints.*



Important Validation Annotations

jakarta.validation.constraints.*	org.hibernate.validator.constraints.*
✓ @Digits	✓ @CreditCardNumber
✓ @Email	✓ @Length
✓ @Max	✓ @Currency
✓ @Min	✓ @Range
✓ @NotBlank	✓ @URL
✓ @NotEmpty	✓ @UniqueElements
✓ @NotNull	✓ @EAN
✓ @Pattern	✓ @ISBN
✓ @Size	

- Sample Validations declaration inside a java pojo class,



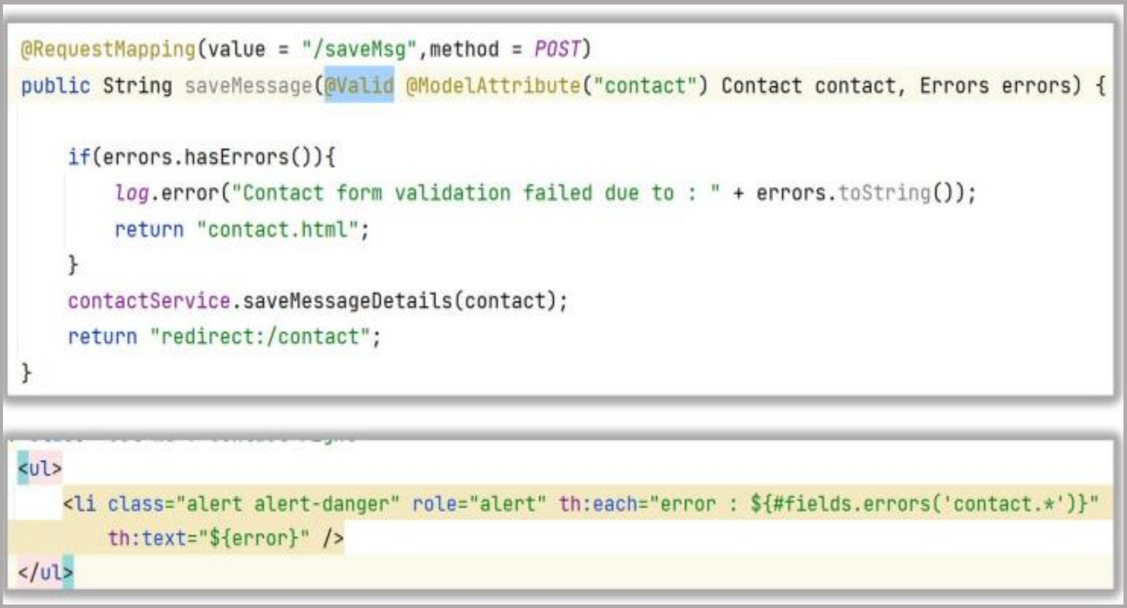
```java
@Data
public class Contact {

    @NotBlank(message="Email must not be blank")
    @Email(message = "Please provide a valid email address" )
    private String email;

    @NotBlank(message="Subject must not be blank")
    @Size(min=5, message="Subject must be at least 5 characters long")
    private String subject;

    @NotBlank(message="Message must not be blank")
    @Size(min=10, message="Message must be at least 10 characters long")
    private String message;
}
```

- We can put @Valid Annotation on method parameters to tell Spring Framework that we want a particular POJO objects needs to be validated based on the validation annotation configurations. For any issues, Framework populates the error details inside the Errors object . The Errors can be used to display on the UI to the user. Sample example code is below,



```java
@RequestMapping(value = "/saveMsg",method = POST)
public String saveMessage(@Valid @ModelAttribute("contact") Contact contact, Errors errors) {

    if(errors.hasErrors()){
        log.error("Contact form validation failed due to : " + errors.toString());
        return "contact.html";
    }
    contactService.saveMessageDetails(contact);
    return "redirect:/contact";
}
```

```html
<ul>
    <li class="alert alert-danger" role="alert" th:each="error : ${#fields.errors('contact.*')}"
        th:text="${error}" />
</ul>
```