Explications liées au programme DemoAppValidationBean

<dependency> **<groupId>** **org.springframework.boot** **</groupId>** **<artifactId>** **spring-boot-starter-validation** **</artifactId>** **</dependency>**	Fichier Pom → il faut ajouter les dépendances à l'API Bean Validation que Spring intègre

@Data **@NoArgsConstructor** **@AllArgsConstructor** **@Component** **public class Employe implements Serializable {** **@Positive(message="Le Id doit être un entier positif !")** **private int id;** **@NotBlank(message="Le nom ne peut pas être à null ou un chaîne vide")** **@Pattern(regexp="[A-Z][A-Za-z]*", message="Non respect des règles pour le nom... ")** **private String nom;** /* Autres exemples @NotBlank(message="Le prénom ne peut peut^^etre à null ou un chaîne vide") String prenom; @Min(value=1000, message="La valeur minimum doit être de 1000") @Max(value=20000, message="La valeur maximum doit être de 20000") private float bareme; @Pattern(regexp="[A-Z][A-Za-z-0-9]*", message="Non respect des règles pour le passwd") private String passwd; */ **}**	On peut préciser les règles de validation des propriétés d'un bean par les annotations suivantes : • **@NotNull** • **@NotBlank** • **@Min/ @Max** • **@Positive** • **@Future** • **@Past** • **@Size** • **@Email** • **@NotEmpty**

@Controller **public class MyControllerInscription {** **@Autowired** **Employe emp;** **@GetMapping("/inscription")** **public String inscription(Model model)** **{ model.addAttribute("emp", emp);** **return "inscription";** **}** **}**	Exemple d'utilisation d'une classe DAO pour un méthode seulement

@Controller **public class MyControllerSuite {** **@PostMapping("/suite")** **public String suite(@Valid Employe emp,** **Errors errors , Model model)** **{** **if(errors.hasErrors())** **{ // Traitement éventuel des erreurs, ici on les affiche** **System.out.println("Erreurs !!!");** **List<ObjectError> list=errors.getAllErrors();** **for(ObjectError obj:list)** **{ System.out.println("errors:"+obj.toString()); }** **}** **else System.out.println("pas d'erreurs");** **model.addAttribute("emp", emp);** **if(errors.hasErrors()) return "inscription";** **else return "suite";** **}** **}**	Le paramètre Employe doit être annoté par **@Valid** La méthode reçoit un objet **Errors** qui contient les éventuelles erreurs de validation On teste si l'objet **Errors** contient des erreurs de validation

Fichier Inscription.html

```
<html>
  <body>
   <form th:action="@{/suite}" th:object="${emp}" method="post">
     Id:    <input type="text" th:field="*{id}" th:errorclass="errorField"/> <br>
     <span th:errors="*{id}"  class="errorMsg">Erreur Id</span><br>

     Nom:   <input type="text" th:field="*{nom}"  th:errorclass="errorField"  /> <br>
     <span th:errors="*{nom}" class="errorMsg">Erreur Nom</span><br>
    <button type="submit" > Envoyer</button>
   </form>
  </body>
</html>
```

Fichier Suite.html

```
<html>
  <body>
    <div>TODO write content</div>
     <span th:text="${emp.id}"/><br>
     <span th:text="${emp.nom}"/><br>
  </body>
</html>
```