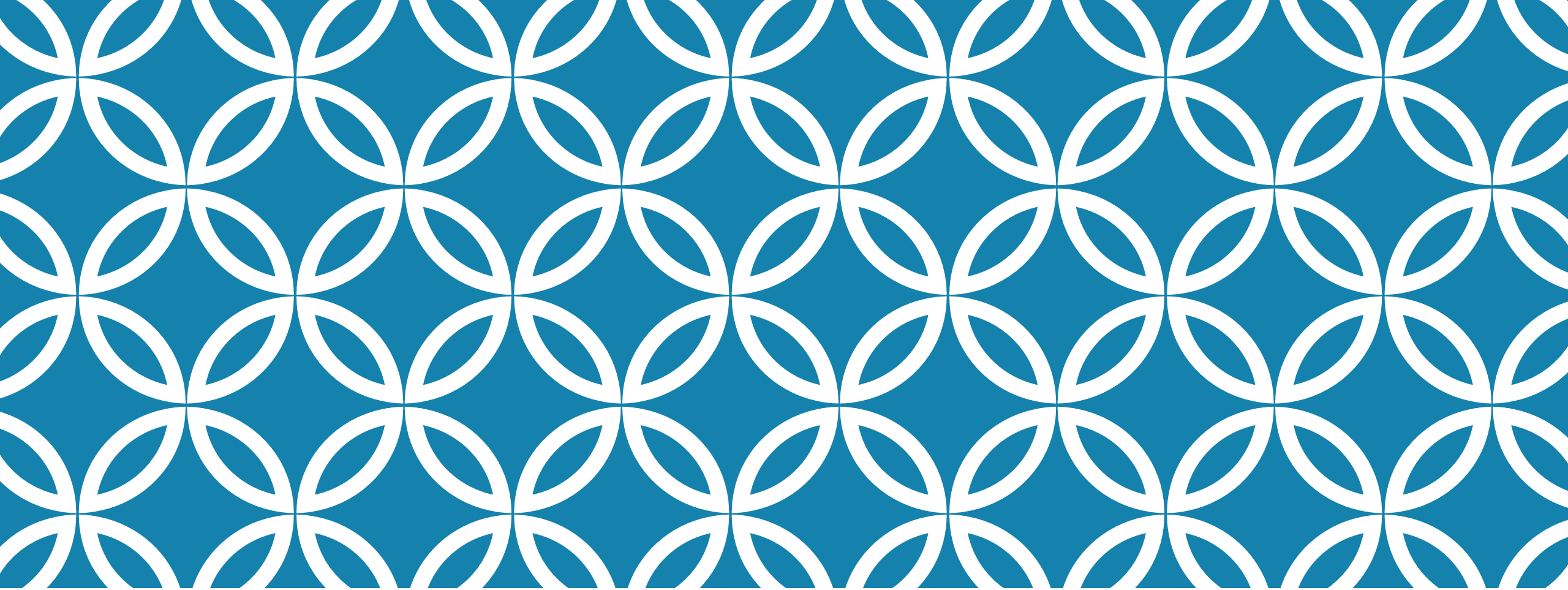
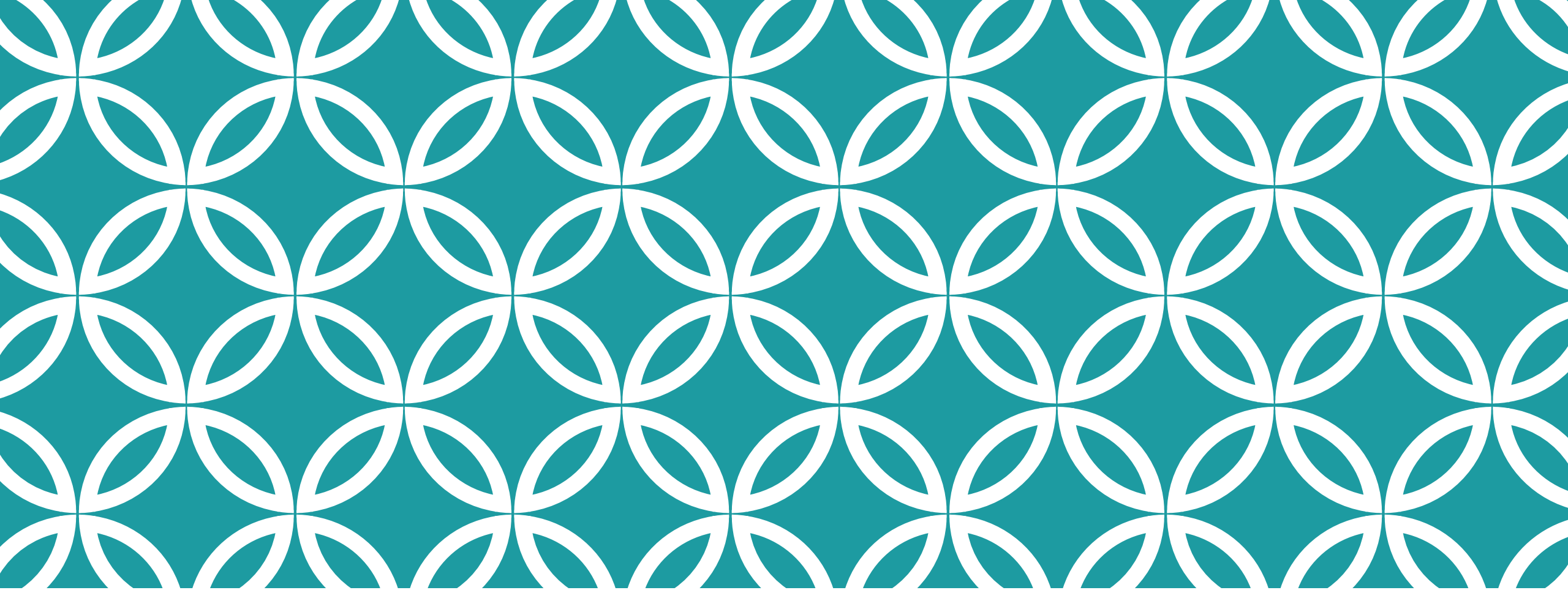


SPRING

Jérémy PERROUAULT



VALIDATION

Validation

VALIDATION

Le principe de la validation

- Vérifier si les champs obligatoires sont remplis
- Vérifier si une valeur est comprise entre x et y
- Ces validations sont à faire, une à une, dans la méthode POST

Spring MVC et l'API de validation vont nous éviter tout ça !

Utilisation de l'annotation @**Valid**

VALIDATION

Ajouter @**Valid** devant @**ModelAttribute**

Active la validation api-validator (hibernate-validator)

```
@PostMapping("/produit/nouveau")
public String ajouterProduit(@Valid @ModelAttribute("produti") Produit produit, BindingResult result, Model model) {
  if (result.hasErrors()) {
    System.out.println("Le produit n'a pas été validé ...");
    return "form-produit";
  }

  return "redirect:/produits";
}
```

Permet de connaître les erreurs lors du Bind, si erreur il y a

Si utilisation de plusieurs @**ModelAttribute**

- Il faut placer un BindingResult juste après un @**ModelAttribute**
  - Celui qui suit @**ModelAttribute** lui correspond

VALIDATION

2 options sont possibles pour la validation

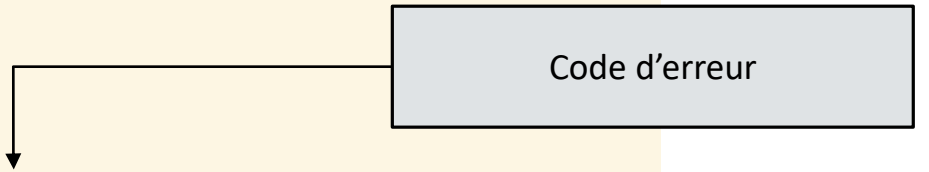
- Utiliser une classe qui implémente l’interface "Validator"
- Utiliser Hibernate-Validation

VALIDATION – VALIDATOR

Implémenter une classe qui implémente *Validator* (SpringFramework)

```
public class ProduitValidator implements Validator {
  @Override public boolean supports(Class<?> cls) {
    return Produit.class.equals(cls);
  }

  @Override public void validate(Object obj, Errors e) {
    ValidationUtils.rejectIfEmptyOrWhitespace(e, "nom", "nom.empty", "Le nom doit être saisi");
  }
}
```



VALIDATION – VALIDATOR

La méthode @**RequestMapping** nécessite un Validator

- Soit en première instruction de la méthode (dans ce cas, @**Valid** n'est plus nécessaire)

```
@PostMapping("/produit/nouveau")
public String ajouterProduit(@Valid @ModelAttribute("produit") Produit produit, BindingResult result, Model model) {
  new ProduitValidator().validate(produit, result);
  //...
}
```

- Soit via une méthode annotée de @InitBinder (directement dans le contrôleur)

```
@InitBinder
protected void initBinder(WebDataBinder binder) {
  binder.addValidators(new ProduitValidator());
}
```

VALIDATION – HIBERNATE-VALIDATOR

Il suffit d'annoter les propriétés de la classe

```
public class Produit {
  @NotEmpty(message = "Le nom est obligatoire")
  private String nom;
}
```

VALIDATION

Pour afficher les messages d'erreur dans la page JSP

```
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
```

```
<form:form method="POST" modelAttribute="produit">
  <div>
    <form:label path="libelle">Libellé :</label>
    <form:input path="libelle" type="text" />
    <span><form:errors path="libelle" /></span>
  </div>
</form:form>
```

VALIDATION

Pour afficher les messages d'erreur dans la page JSP

```
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring" %>
```

```
<form:form method="POST" modelAttribute="produit">
  <spring:hasBindErrors name="produit">
    <ul>
      <c:forEach items="${ errors.allErrors }" var="error">
        <li><spring:message message="${ error }" /></li>
      </c:forEach>
    </ul>
  </spring:hasBindErrors>
</form:form>
```

EXERCICE

Modifier le CRUD « produit »

- Utiliser @**ModelAttribute** et la validation
- Message si le nom / prix n'est pas saisi ou mal saisi