Spring

Validation

Spring: Validation

- En un Model podemos agregar instrucciones de validación, para que Spring las realice por nosotros.
- Estas instrucciones las insertamos a través de anotaciones de javax.validation.constraints.*.
- Por ejemplo @NotEmpty, se encarga de que el String nombre no se provea vacio “” .

```
import
javax.validation.constraints.Not
Empty;

public class Usuario{
    @NotEmpty
    private String nombre;
}
```

Spring: Validation

- Para poder importar las restricciones de validación de javax, se debe agregar en el pom una dependencia, a partir de Spring 2.3.*

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

Spring: Validation

- Para solicitar la validación de los atributos de un modelo por parte de Spring. Se debe agregar la anotacion @Valid.

```
@Controller
public class SiteController{

    @GetMapping("/usuario")
    public String usuario(@Valid
Usuario usuario, Model model)
{
 model.addAttribute("usuario",
usuario);
        return "ver";
    }
}
```

Spring: View Errors

- Ahroa bien con @Valid solicitamos la validación, pero si algo sale mal debemos informar cuales son los campos que han fallado y por que han fallado.
- Para eso debemos usar BindingResult.

```
@Controller
public class SiteController{

    @GetMapping("/usuario")
    public String usuario(@Valid
Usuario usuario, BindingResult
result, Model model){
 model.addAttribute("usuario",
usuario);
        if(result.hasErrors())
            return "index";
        return "ver";
    }
}
```

Spring: View Errors

- Luego para mostrar los errores en el archivo index.html. Debemos usar etiquetas de Thymeleaf.

```
<form action="#" th:action="@{/ver}"
th:object="${usuario}" method="post">
    <label for="nombre">Nombre:</label>
    <input id="nombre" type="text"
th:field="*{nombre}" />
    <span
  th:if="{#fields.hasErrors('nombre')}"
  th:errors="*{nombre}">Error en
  nombre</span></br>
</form>
```

```
@Controller
public class SiteController{

    @GetMapping("/usuario")
    public String ver(@Valid
Usuario usuario, BindingResult
result, Model model){
 model.addAttribute("usuario",
usuario);
        if(result.hasErrors())
            return "index";
        return "ver";
    }
}
```

Spring: Validation Annotation

- Hay mas tipos de validación. ¡Claro que si!

- AssertFalse
- AssertFalse.List
- AssertTrue
- AssertTrue.List
- DecimalMax
- DecimalMax.List
- DecimalMin
- DecimalMin.List
- Digits
- Digits.List
- Email
- Email.List
- Future
- Future.List
- FutureOrPresent
- FutureOrPresent.List
- Max
- Max.List
- Min
- Min.List
- Negative
- Negative.List
- NegativeOrZero.List
- NotBlank
- NotBlank.List
- NotEmpty
- NotEmpty.List
- NotNull
- NotNull.List
- Null
- Null.List
- Past
- Past.List
- PastOrPresent
- PastOrPresent.List
- Pattern
- Pattern.List
- Positive
- Positive.List
- PositiveOrZero
- PositiveOrZero.List
- Size
- Size.List

Spring: @Pattern Validation

- Vamos a revisar @Pattern ya que trabaja con expresiones regulares. Y con eso se convierte en una herramienta muy poderosa.

```
import
javax.validation.constraints.Pa
ttern;

public class Usuario{
    @Pattern(regex="^\\
d{1,2}\\.\\d{3}\\.\\d{3}\\-(\\d|k|K)
$")
    private String run;
}
```

Spring: Custom Validation

- Crearemos una validación para el RUN. Pero esta vez vamos a validar mas que el formato. Vamos a validar parte de su semántica.

```
import
org.springframework.validation
.Validator;

public class RUNValidation
implements {

	@Override
	public boolean supports
(Class<?> clazz) {}
	@Override
	public void validate
(Object target, Errors errors) {}
}
```

Spring: Custom Validation

- El el método validate incluiremos la llamada a un método privado que se encargara de validar el RUN.

```
public class RUNValidation
implements {
      @Override
      public void validate
(Object target, Errors errors) {
         If(!validateRUN())
          errors.rejectValue(“run”,
null, ”run no valido”);
      }
      private boolean
validateRUN(String run) {
      //validation code here
}
}
```