Aplicaciones Open Source

Semana10
26/05/18

Carlos A. Quinto Cáceres
pcsicqui@upc.edu.pe

Agenda

- Validaciones
- Consultas

Logros del día

- Al final de la clase, los alumnos podrán:
  - Realizar validaciones en las entradas de formulario.
  - Definir consulta a reutilizar en las entidades y repositorio.

Validaciones

Concepto

- Cuando nuestras aplicaciones reciben la entrada de datos por parte de los usuarios, debemos aplicar algunas reglas de validación.
- Es importante porque nos ayudan a mantener la integridad de los datos y la lógica de la aplicación.
- Para el caso de JPA, contamos con anotaciones que nos permiten aplicar algunas reglas a los valores.

Anotaciones

Validación	Detalle	Ejemplo
@Size	Validar el tamaño de la propiedad y debe coincidir con los límites especificados.	@Size(min=3, max=75)
@NotEmpty	El valor no puede ser vacío o no nulo.	@NotEmpty(message="mensaje personalizado")
@NotBlank	El valor no puede ser vacío o no nulo. No toma en cuenta espacios vacíos.	@NotBlank
@Digits	El número debe estar en el rango especificado. Integer para dígitos enteros y Fraction para dígitos fraccionarios.	@Digits(integer=6, fraction=2)
@Max	Debe ser menor o igual al valor	@Max(200)
@Min	Debe ser mayor o igual al valor	@Min(100)
@Email	Valida que tenga el formato de un correo	@Email

Mensajes de error

```
@RequestMapping(value="/autor/guardar", method=RequestMethod.POST)
public String guardar(@ModelAttribute @Valid Autor objAutor, BindingResult bindingResult,
    if (bindingResult.hasErrors()) {
        return "admin/autor_agregar";
    }else{
        Autor objResultado = serviceAutor.save(objAutor);
```

```
<div class="form-group">
    <label th:text="Nombres" for="campo01"></label>
    <input th:field="*{nombres}" type="text" class="form-control" id="campo01" placeholder="ingre nombres" />
    <p class="text-danger" th:if="${#fields.hasErrors('nombres')}" th:errors="*{nombres}">Error en el nombre</p>
</div>
<div class="form-group">
    <label th:text="Apellidos" for="campo02"></label>
    <input th:field="*{apellido}" type="text" class="form-control" id="campo02" placeholder="ingrese apellidos" />
    <p class="text-danger" th:if="${#fields.hasErrors('apellido')}" th:errors="*{apellido}">Error en el nombre</p>
</div>
```

Consultas

NamedQuery

- Es una consulta definida de forma manual con una cadena de consulta predefinida.
- Se declaran en las entidades.
- Permite mejorar la organización del código al separar las cadenas de consulta JPQL del código Java.
- Se puede hacer uso de parámetros de consulta.

Ejemplos

```
@Entity
@NamedQuery(name = "Autor.buscarPorNacionalidad",
            query = "select a from Autor a where a.nacionalidad = ?1")
```

```
@NamedQueries({
    @NamedQuery(name = "Autor.buscarPorNacionalidad",
                query = "select a from Autor a where a.nacionalidad = ?1"),
    @NamedQuery(name = "Autor.buscarPorApellido",
                query = "select a from Autor a where a.apellido = ?1"),
})
```

Query

- Se declaran en el método que lo va ejecutar .
- Tienen prioridad sobre los NamedQuery.

```
@Query("select a from Autor a where a.nombres = ?1")
Autor buscarPorNombre01(String nombre);
```

```
@Query("select a from Autor a where a.apellido like CONCAT('%',:apellidos,'%')")
List<Autor> buscarApellidos(@Param("apellidos") String apellidos);
```